United Electronic Industries

The High-Performance Alternative

DNA-AO-308-420 4-20mA Current Analog Output Layer User Manual

Release 1.0

November 2006 Edition

PN Man-DNA-AO-308-420-1106

Contacting United Electronic Industries

Mailing Address:

611 Neponset Street
Canton, MA 02021
U.S.A.

For a list of our distributors and partners in the US and around the world, please see
http://www.ueidaq.com/partners/

Support:

Telephone: (781) 821-2890
Fax: (781) 821-2891

Also see the FAQs and online “Live Help” feature on our web site.

Internet Support:

Support: support@ueidaq.com
Web-Site: www.ueidaq.com
FTP Site: ftp://ftp.ueidaq.com

Product Disclaimer:

WARNING!

DO NOT USE PRODUCTS SOLD BY UNITED ELECTRONIC INDUSTRIES, INC. AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS.

Products sold by United Electronic Industries, Inc. are not authorized for use as critical components in life support devices or systems. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. Any attempt to purchase any United Electronic Industries, Inc. product for that purpose is null and void and United Electronic Industries, Inc. accepts no liability whatsoever in contract, tort, or otherwise whether or not resulting from our or our employees' negligence or failure to detect an improper purchase.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

This document outlines the feature set and use of the DNA-AO-308-420 4-20mA current analog output layer when used with the PowerDNA Core Module. This manual describes the following products:

- DNA-AO-308-420 4-20mA current, 16-bit, 8-channel, Analog Output Layer with per-channel digital offset and gain calibration, buffered output, excellent linearity, and low output noise. Designed for use in industrial and control applications to interface with standard 4-20mA sensors.
- Accessory modules such as cables, screw terminals, and cooling fans.

1.1 Organization

This DNA-AO-308-420 User Manual is organized as follows:

1.1.1 Introduction

This chapter provides an overview of DNA-AO-308-420 board/layer features, accessories, and what you need to get started.

1.1.2 DNA-AO-308-420 Layer

This chapter provides an overview of the device architecture, connectivity, logic, and accessories for the DNA-AO-308-420 layer board.

1.1.3 Programming with the High-Level API

This chapter provides a general overview of procedures that show how to create a session, configure the session, and generate output on a DNA-AO-308-420 layer, working with the UEIDAQ Framework High-Level API.

1.1.4 Programming with the Low-Level API

This chapter describes the Low-Level API commands for configuring and using a DNA-AO-308-420 layer.

Appendices

A. Accessories This appendix provides a list of accessories available for use with a DNA-AO-308-420 layer.

B. Calibration This appendix outlines the recommended calibration procedure for a DNA-AO-308-420 board/layer.

Index This is an alphabetical index of topics covered in this manual.

NOTE: A glossary of terms used with the PowerDNA and layers can be viewed and/or downloaded from www. ueidaq.com.

Manual Conventions

To help you get the most out of this manual and our products, please note that we use the following conventions:

Tips are designed to highlight quick ways to get the job done, or reveal good ideas you might not discover on your own.

NOTE: ***Notes alert you to important information.***

CAUTION! *Caution advises you of precautions to take to avoid injury, data loss, and damage to your boards or a system crash.*

Text formatted in **bold** typeface generally represents text that should be entered verbatim. For instance, it can represent a command, as in the following example: “You can instruct users how to run setup using a command such as **setup.exe**.”

1.2 The DNA-AO-308 Series Analog Output Layers

The DNA-AO-308 series of Analog Output boards/layers includes the following products:

- DNA-AO-308 16-bit, 8-channel, ±10V Analog Output Board/Layer
- DNA-AO-308-350 16-bit, 8-channel, ±10V, High Current Analog Output Board/Layer
- DNA-AO-308-353 16-bit, 8-channel, ±40V, High Voltage Analog Output Board/Layer
- DNA-AO-308-420 16-bit, 8-channel, 4-20 mA Current Analog Output Board/Layer

This manual describes the DNA-AO-308-420 4-20mA current, 16-bit, 8-channel, Analog Output Board/Layer only The other products in the series are described in separate documents.

The technical specifications for the DNA-AO-308-420 4-20mA current Analog Output Layer are listed in **Table 1-1**.

Table 1-1. DNA-AO-308-420 Technical Specifications

Number of Channels	8
Resolution	16 bits
Max Update Rate: @ 16-bit resolution @ 12-bit resolution @ 9-bit resolution	 100 kHz/channel (800kHz max aggregate) 200 kHz/channel (800kHz max aggregate) 400 kHz/channel (800kHz max aggregate)
Buffer Size	1K samples
Type of D/A	double-buffered
INL (no load)	±1 LSB (0.003%)
DNL (no load)	±1 LSB (0.003%)
Monotonicity Over Temperature	16 bits
Gain Linearity Error	0.002%
Gain Calibration Error	±244 µA
Offset Calibration Error	±244 µA
Offset Drift	5ppm/°C
Gain Drift	5ppm/°C
Output Range	4-20mA
Output Coupling	DC
Output Impedance	0.1Ω max
Capacitive Loads	500 pF
Settling Time	10 µs to 16 bits
Maximum Load[1]	450 - 800 Ohms
Isolation	350Vrms
Power Consumption[2]	1.8W - 6W
Physical Dimensions	3.875" x 3.875" (98 x 98 mm)
Operating Temp. (tested)	-40°C to +85°C
Operating Humidity	90%, non-condensing

[1] Refer to the Typical Performance Characteristics for more details.
[2] If the total power consumption of the layer is over the 4.5W, the DNA-FANx rear-mount cooling fan is required. Refer to the Typical Performance Characteristics for more details.

Figure 1-1 is a photo of the DNA-AO-308-420 Layer board.

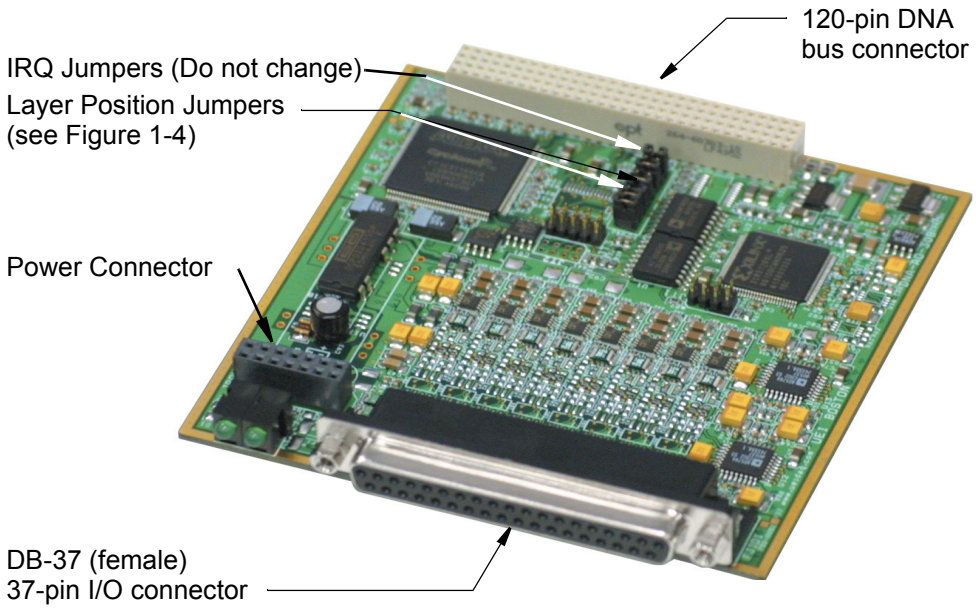

Figure 1-1. DNA-AO-308-420 Board

1.3 Device Architecture

The DNA-AO-308-420 4-20mA current Analog Output Layer board has eight individual analog output channels. A Block Diagram of the board/layer is shown in **Figure 1-2**.

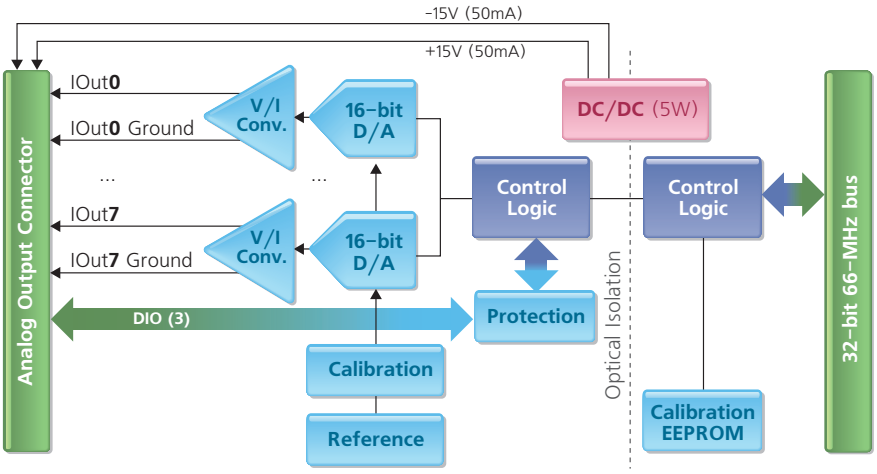

Figure 1-2. Block Diagram of DNA-AO-308-420 Device Architecture

1.4 Layer Connectors and Wiring

Since the DNA-AO-308-420 4-20mA current Analog Output board is designed with output buffers, separate sense lines are not provided. To minimize error due to differences in lead resistance, be sure to use equal length signal and return lines.

Be sure to short the signal and return lines whenever the output is not connected to an external circuit.

NOTE: ***If signal and return lines are left open-circuited, the output amplifier feedback circuitry is open, causing the signal to drift either to the positive or to the negative side of the power supply rail.***

1.4.1 Connectors

The pinout of the 37-pin connector for the DNA-AO-308-420 Layer board is shown in **Figure 1-3**. A physical layout of the board is shown in **Figure 1-4**.

.

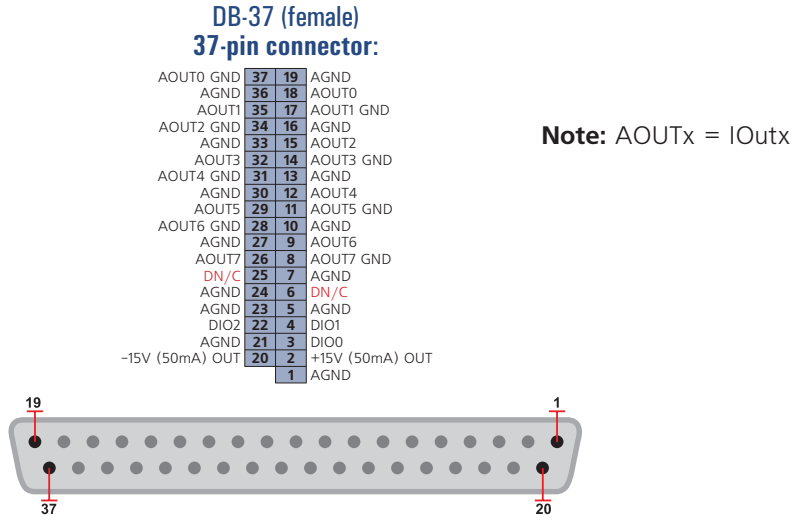

Figure 1-3 DB-37 I/O Connector Pinout

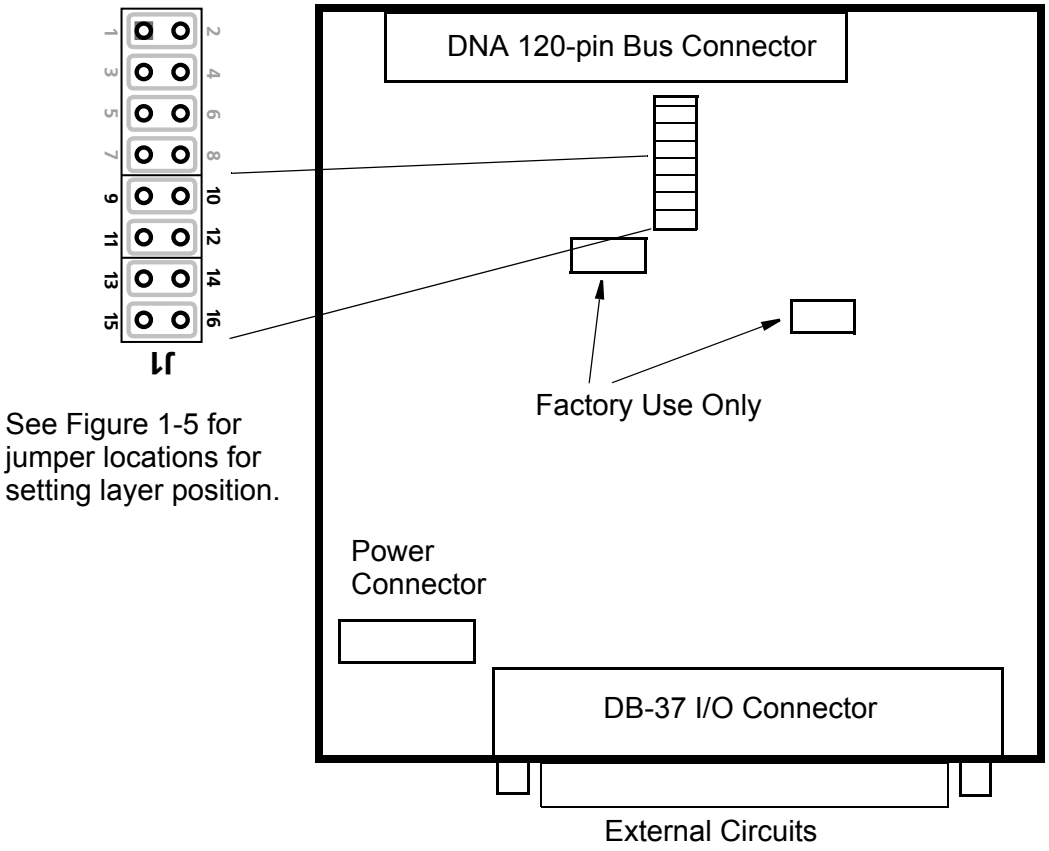

Figure 1-4. Physical Layout of DNA-AO-308-420 Layer Board

1.4.1.1 Jumper Settings

A diagram of the jumper block is shown in **Figure 1-4**. To set the layer position jumpers, place jumpers as shown in **Figure 1-5**.

		Layer's Position as marked on the Faceplate*					
		I/O **1**	I/O **2**	I/O **3**	I/O **4**	I/O **5**	I/O **6**
Jx Pins	9-10	● ●	○ ○	● ●	○ ○	● ●	○ ○
	11-12	● ●	● ●	○ ○	○ ○	● ●	● ●
	13-14	● ●	● ●	● ●	● ●	○ ○	○ ○
	15-16	● ●	● ●	● ●	● ●	● ●	● ●

* *All I/O Layers are sequentially enumerated from top to the bottom of the Cube*

○ ○ - **Open** ● ● - **Closed**

Figure 1-5. Diagram of DNA-AO-308-420 Layer Position Jumper Settings

Chapter 2 Programming with the High-Level API

2.1 Programming with the Ueidaq Framework API

This section describes how to program the DNA-AO-308x (AO-308, AO-308-350, AO-308-353, AO-308-420) board/layer using the UeiDaq Framework High-Level API.

The UeiDaq Framework is object-oriented. Its objects can be manipulated in the same manner within various development environments, such as Visual C++, Visual Basic, or LabVIEW.

Although the following section focuses on the C++ API, the concept is the same for any programming language you use.

Please refer to the “UeiDaq Framework User Manual” for more information on using other programming languages.

Please refer to the examples that come with the UeiDaq Framework. They contain detailed and commented code that can be compiled and executed.

2.1.1 Creating a Session

The Session object controls all operations on your PowerDNA device. Therefore, the first task is to create a session object, by entering:

```
CUeiSession session;
```

2.1.2 Configuring the Channels

Framework uses resource strings to select which device, subsystem, and channels you use within a session. The resource string syntax is similar to a web URL, as:

```
<device class>://<IP address>/<Device Id>/
<Subsystem><Channel list>
```

For PowerDNA, the device class is **pdna**.

For example, the following resource string selects analog output channels 0,1 on device 1 at IP address 192.168.100.2: “pdna://192.168.100.2/Dev1/Ao0:1”

```
// Configure channels 0,1 with an output
// range of ±10V (or 4-20 mA for the AO-420)
session.CreateAOChannel("pdna://192.168.100.2/
Dev0/ao0:1",-10.0, 10.0);
```

2.1.3 Configuring the Timing

You can configure the DNA-AO-308x layer to run either in simple mode (point by point) or buffered mode (ACB mode).

In simple mode, the delay between samples is determined by software on the host computer.

In buffered mode, the delay between samples is determined by the DNA-AO-308x on-board clock.

The following sample shows how to configure the simple mode. Please refer to the “UeiDaq Framework User Manual” to learn how to use the other timing modes.

```
session.ConfigureTimingForSimpleIO();
```

2.1.4 Writing Data

Writing data to the DNA-AO-308x board/layers is done with a writer object. You can create a writer object that writes raw data straight to the D/A converter. You can also create a writer object that writes data scaled to volts. Framework automatically performs a conversion to binary code before sending the data to the D/A converter.

The following sample code shows how to create a scaled writer object and write a sample.

```
// Create a reader and link it to the
// session's stream
CueiAnalogScaledWriter
writer(session.GetDataStream());

// write one scan, the buffer must contain
// one value
// for each channel
double data[2] = {0.0, 0.0};
writer.WriteSingleScan(data);
```

Similarly, you can create a raw writer object by entering the following:

```
// Create a reader and link it to the session's stream

CUeiAnalogRawWriter writer(session.GetDataStream());

// write one scan, the buffer must contain one value

// for each channel

uInt16 data[2] = {0x1234, 0x5678};

writer.WriteSingleScan(data);
```

All the 308x analog output layers are programmed the same way.

2.1.5 Cleaning-up the Session

The session object cleans itself up when it goes out of scope or when it is destroyed. If you want to reuse the object with a different set of channels or parameters, you can manually clean up the session with the following:

```
session.CleanUp();
```

Chapter 3 Programming with the Low-Level API

This section describes how to program the PowerDNA cube using the low-level API. The low-level API offers direct access to PowerDNA DAQBIOS protocol and also allows you to access device registers directly.

We recommend that, where possible, you use the UeiDaq Framework high-level API (*see* “Programming with the Ueidaq Framework API” on page 7), which is easier to use than the low-level API.

You need to use the low-level API only if you are using an operating system other than Windows.

3.1 Configuration Settings

Configuration settings are passed in `DqCmdSetCfg()` and `DqAcbInitOps()` functions.

Not all configuration bits apply to DNA-AO-308x series (AO-308, AO-308-350, AO-308-353, AO-308-420) boards/layers, however. The following bits make sense:

```
#define DQ_FIFO_MODEFIFO (2L << 16)
    // continuous acquisition with FIFO
#define DQ_LN_MAPPED     (1L<<15)
    // For WRRD (DMAP) devices
    //(automatically selected)
#define DQ_LN_STREAMING  (1L<<14)
   // For RDFIFO devices - stream the FIFO data
   //(automatically selected) For WRFIFO - do NOT
   //send reply to WRFIFO unless needed
#define DQ_LN_IRQEN    (1L<<10) //enable layer irqs
#define DQ_LN_PTRIGEDGE1 (1L<<9)
   // stop trigger edge MSB
#define DQ_LN_PTRIGEDGE0 (1L<<8)
    // stop trigger edge: 00 - software,
    // 01 - rising, 02 – falling
#define DQ_LN_STRIGEDGE1 (1L<<7)
   // start trigger edge MSB
#define DQ_LN_STRIGEDGE0 (1L<<6)
   // start trigger edge: 00 - software,
  //01 -rising, 02 - falling
#define DQ_LN_CVCKSRC1   (1L<<5)
  // CV clock source MSB
#define DQ_LN_CVCKSRC0   (1L<<4)
  // CV clock source 01 - SW,10 -  HW, 11 –EXT
#define DQ_LN_CLCKSRC1   (1L<<3)
  // CL clock source MSB
#define DQ_LN_CLCKSRC0   (1L<<2)
 // CL clock source 01 - SW,10 - HW,11 –EXT
#define DQ_LN_ACTIVE     (1L<<1)
 // "STS" LED status
```

```
#define DQ_LN_ENABLED    (1L<<0)
                                  // enable operations
```

For streaming operations with hardware clocking, select the following flags:

```
DQ_LN_ENABLE | DQ_LN_CVCKSRC0 | DQ_LN_STREAMING |
DQ_LN_IRQEN | DQ_LN_ACTIVE | DQ_AO308_BI10
```

DNA-AO-308x has a range of layer-specific settings - as follows:

The following modes are reserved for future use:

```
#define DQ_AO308_MODEFIFO  (1L << 19)
     // continuous output with FIFO
#define DQ_AO308_MODECONT  (2L << 19)
    // waveform mode – continuous
#define DQ_AO308_MODECYCLE (3L << 19)
    // waveform mode – regenerate
#define DQ_AO308_MODEWFGEN (4L << 19)
   // waveform mode - hardware
```

`DQ_LN_ENABLE` enables all operations with the layer. `DQ_LN_CVCKSRC0` selects the internal channel list clock (CL) source as a timebase. AO-308 supports CV clock.
`DQ_LN_ACTIVE` is needed to switch on “STS” LED on the CPU layer.

You can select either the CL or CV clock as a timebase. Because of the parallel architecture of AO-308x layer, either clock triggers all converters.

```
Aggregate rate = Per-channel rate * Number of
channels
```

Note that acquisition rate cannot be selected on per-channel basis.

3.2 Channel List Settings

The DNA-AO-308x layers have the following channel list structure:

Bit	Name	Purpose	Comments
31	LNCL_NEXT	Tells firmware that there is a next entry n the channel list	
21	DQ_LNCL_UPDALL	Check update line to update all DACs.	Reserved
20	DQ_LNCL_WRITE	Write data into the DAC, but do not update.	Reserved
7...0		Channel number	

Table 3-1. DNA-AO-308 Layer Channel List Structure

3.3 Data Representation

DNA-AO-308x has 16-bit straight binary data representation, as shown in **Table 3-2**.

Layer	Range/ Value	0x0	0x8000	0xFFFF	Span	Offset
AO-308, 350,	±10V	-10V	0	+10V	20V	0
AO-353	±40V	-40V	0	+40V	80V	0
AO-420	4-20mA	4mA	12mA	20mA	16mA	4mA

Table 3-2. DNA-AO-308x Layer Channel List Structure

To convert voltage into an A/D representation, use the following formula:

```
Raw = (Volt+Offset)/(Span/0xFFFF),
```

where Volt is the desired level in volts.

To convert current into A/D representation (AO-308-420 only), use the following formula:

```
Raw = (mA+Offset)/(Span/0xFFFF),
```

where mA is the desired level in mA.

3.4 Layer-specific Commands and Parameters

Layer-specific functions are described in the *DaqLibHL.h* file.

`DqAdv30xWrite()`

This function works using underlying `DqCmdIoctl()`. It uses the `DQCMD_IOCTL` command with the `DQIOCTL_CVTCHNL` function.

When this function is called for the first time, the firmware terminates any ongoing operation on the device.

Then, the firmware parses the channel list and writes the passed values one by one.

Therefore, you cannot perform this function call when the layer is involved in any streaming or data mapping operations.

Every write to the channel takes approximately 3.3µs. Thus, execution time for this function depends on the number of channels in the channel list.

3.5 Using the Layer in ACB Mode

This is a pseudo-code example that highlights the sequence of functions needed to use ACB on the 308x layers. A complete example with error checking can be found in the directory *SampleACB30x*.

Note that we use the `#defines` for a 30x for a DNA-AO-308x layer.

```
#include "PDNA.h"
// unit configuration word
#define CFG308          (DQ_LN_ENABLED \
                        |DQ_LN_ACTIVE \
                        |DQ_LN_GETRAW \
                        |DQ_LN_IRQEN \
                        |DQ_LN_CVCKSRC0 \
                        |DQ_LN_STREAMING \
                        |DQ_AI30x_MODEFIFO
                        |DQ_AO30x_BI10)
                        uint32 Config = CFG30x;
```

STEP 1: Start DQE engine.

```
#ifndef _WIN32
    DqInitDAQLib();
#endif
    // Start engine
    DqStartDQEngine(1000*1, &pDqe, NULL);
    // Open communication with IOM
    hd0 = DqOpenIOM(IOM_IPADDR0, DQ_UDP_DAQ_PORT,
TIMEOUT_DELAY, &RdCfg);
    // Receive IOM crucial identification data
    DqCmdEcho(hd0, DQRdCfg);

    // Set up channel list
    for (n = 0; n < CHANNELS; n++) {
        CL[n] = n;
    }
```

STEP 2: Create and initialize host and IOM sides.

```
    // Now we are going to test device
    // DqAcbCreate(pDqe, hd0, DEVN, DQ_SS0IN, &bcb);
    // Let's assume that we are dealing with AI-201
      //device
dquser_initialize_acb_structure();
    // Now call the function
    DqAcbInitOps(bcb,
                 &Config,
                 0,      //TrigSize,
                 NULL,   //pDQSETTRIG TrigMode,
                 &fCLClk,
                 0,      //float*  fCVClk,
                 &CLSize,
                 CL,
                 0,      //uint32* ScanBlock,
                 &acb);
    printf("Actual clock rate: %f\n", fCLClk);
    // Now set up events
    DqeSetEvent(bcb,
DQ_eFrameDone|DQ_ePacketLost|DQ_eBufferError|DQ_eP
acketOOB);
    // Allocate data buffer
    datta = dquser_allocatebuffer();
    // Pre-fill ACB with raw data
    dquser_prefillbuffer(data);
    DqAcbPutScansCopy(bcb, data, // buffer
                      bufsize,   // buffer size in
                                 //scans
                      bufsize,   // minimum size
                      &size,     // actual copied
                                 //size (from user
                                // buffer into ACB)
                      &avail);

                                // available free
                                 // space in buffer
```

STEP 3: Start operation.

```
    // Start operations
    DqeEnable(TRUE, &bcb, 1, FALSE);
```

STEP 4: Process data.

```
        // We will not use event notification at first
        // - just retrieve scans
        while (keep_looping) {
            DqeWaitForEvent(&bcb, 1, FALSE,
EVENT_TIMEOUT, &events);
            if (events & DQ_eFrameDone) {
                // fill buffer with more data
                dquser_prefillbuffer(data);
                DqAcbPutScansCopy(bcb, data,// buffer
                                  bufsize,  // buffer size
                                  MINRQ,   // minimum size
                                     &size,   // actual
                                  //copied size from
                                  //user buffer into
                                  //ACB &avail);
                                  // available free space
                                     //in buffer
            }
        }
```

STEP 5: Stop operation.

```
        DqeEnable(FALSE, &bcb, 1, FALSE);
```

STEP 6: Clean up.

```
        DqAcbDestroy(bcb);
        DqStopDQEngine(pDqe);
        DqCloseIOM(hd0);
#ifndef _WIN32
        DqCleanUpDAQLib();
#endif
```

3.6 Using the Layer in DMap Mode

```
#include "PDNA.h"
```

STEP 1: Start DQE engine.

```
#ifndef _WIN32
    DqInitDAQLib();
#endif
    // Start engine
    DqStartDQEngine(1000*10, &pDqe, NULL);
    // open communication with IOM
    hd0 = DqOpenIOM(IOM_IPADDR0, DQ_UDP_DAQ_PORT,
TIMEOUT_DELAY, &DQRdCfg);
    // Receive IOM crucial identification data
    DqCmdEcho(hd0, DQRdCfg);
```

```
    for (i = 0; i < DQ_MAXDEVN; i++) {
        if (DQRdCfg->devmod[i]) {
            printf("Model: %x Option: %x\n",
DQRdCfg->devmod[i], DQRdCfg->option[i]);
} else {
            break;
}
    }
```

STEP 2: Create and initialize host and IOM sides.

```
DqDmapCreate(pDqe, hd0, &pBcb, UPDATE_PERIOD,
&dmapin, &dmapout);
```

STEP 3: Add channels into DMap.

```
    for (i = 0; i < CHANNELS; i++) {
        DqDmapSetEntry(pBcb, DEVN, DQ_SS0IN, i,
DQ_ACB_DATA_RAW, 1, &ioffset[i]);
    }
    DqDmapInitOps(pBcb);
    DqeSetEvent(pBcb,
DQ_eDataAvailable|DQ_ePacketLost|DQ_eBufferError|D
Q_ePacketOOB);
```

STEP 4: Start operation.

```
    DqeEnable(TRUE, &pBcb, 1, FALSE);
```

STEP 5: Process data.

```
    while (keep_looping) {
        DqeWaitForEvent(&pBcb, 1, FALSE, timeout,
&eventsin);
        if (eventsin & DQ_eDataAvailable) {
            printf("\ndata ");
            for (i = 0; i < CHANNELS; i++) {
                printf("%04x ",
*(uint16*)ioffset[i]);
            }
        }
    }
```

STEP 6: Stop operation.

```
    DqeEnable(FALSE, &pBcb, 1, FALSE);
```

STEP 7: Clean up.

```
    DqDmapDestroy(pBcb);
    DqStopDQEngine(pDqe);
    DqCloseIOM(hd0);
#ifndef _WIN32
```

```
    DqCleanUpDAQLib();
#endif
```

Appendices

A. Accessories

The following accessory items are available for use with the DNA-AO-308-420 layer.

DNA-CBL-37

This is a 3 ft., 37-way flat ribbon cable with one 37-pin male and one 37-pin D-sub connector. Used to connect the DNA-AO-308-420 board to a 37-terminal panel such as the DNA-STP-37.

DNA-STP-37

This is a 37-way screw terminal panel that can be used for making external connections to the DNA-AO-308-420 layer and DNA-CBL-37 cable.

NOTE: If the total power consumption of the layer exceeds 4.5W, a rear mount cooling fan such as the DNA-FAN5 (for 3-layer Cube) or DNA-FAN8 (for 5-layer Cube) should be added to the DNA Cube.

B. Layer Calibration

Calibration should be performed with a milliammeter.

To calibrate the layer, first open-circuit all eight channel signal and return lines.

Run a serial terminal program attached to the IOM serial port.

Use “simod 1” command to calibrate the layer.

Please note that once you perform layer calibration yourself, the factory calibration warranty is void.

Calibration Procedure

The calibration procedure for the DNA-AO-308-420 Layer (using a serial port terminal) is:

STEP 1: Type “simod 1.”

STEP 2: Select the device to be calibrated from the device table.

STEP 3: Insert a milliammeter into the output circuit.

STEP 4: Repeat this step for each channel and output 4mA to each by entering command “a” and value 0.

STEP 5: Adjust the calibration of DAC 0 by entering command 0 and a hexadecimal value to set this DAC to (0x0 – 0xff).

STEP 6: Adjust the calibration of DAC 0 to output 20mA on channels 0 through 7.

STEP 7: Output 20mA on all channels by entering command “a” and value FFFF.

STEP 8: Attach milliammeter to channel 0 and adjust calibration to reach 20mA on all channels (with an error not to exceed .01mA).

STEP 9: When calibration is complete, enter “q” command and reply “y” if you want to save calibration values into EEPROM.

STEP 10: Reset the PowerDNA cube to verify calibration.

For all DNA-AO-308-420 layer boards, we recommend annual factory recalibration at UEI.

Index