United Electronic Industries

The High-Performance Alternative

PowerDNA DNA-AI-205 Analog Input Layer – User Manual

Simultaneous Sampling, 18-bit, 4-channel, 250 kS/s per channel, Analog Input layer for the PowerDNA Cube

July 2007 Edition
Version 3.4
PN Man-DNA-AI-205-0707

Contacting United Electronic Industries

Mailing Address:

27 Renmar Avenue
Walpole, MA 02081
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:	(508) 921-4600
Fax:	(508) 668-2350

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

Table of Figures

Chapter 1 Introduction

This document outlines the feature set and use of the DNA-AI-205 layer. This layer is an analog input module for the PowerDNA I/O Cube.

1.1 Organization of this manual

This PowerDNA AI-205 User Manual is organized as follows:

- **Introduction**
  This chapter provides an overview of PowerDNA Analog Input Series board features, the various models available, and what you need to get started.
- **Chapter 1 — The AI-205 layer**
  This chapter provides an overview of the device architecture, connectivity, and logic of the AI-205 layer.
- **Programming Using the UeiDaq Framework High-Level API**
  This chapter provides an overview of the how to create a session, configure the session for analog input, and interpret results on the AI-205 series layer.
- **Programming Using the Low-Level API**
  This chapter describes Low-level API commands for configuring and using the AI-205 series layer.
- **Appendix A: Accessories**
  This appendix provides a list of accessories available for AI-205 layer(s).
- **Appendix B: Calibration**
  This appendix outlines a layer calibration procedure for the AI-205 series layer.
- **Index**
  This is an alphabetical listing of the topics covered in this manual.

1.2 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.

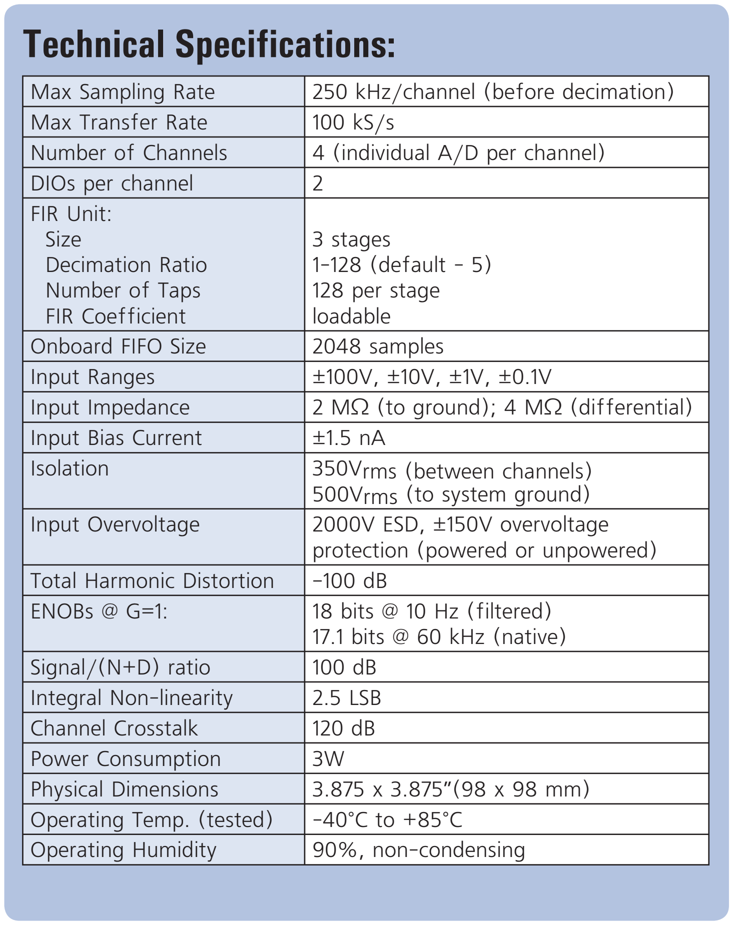
Text formatted in bold typeface generally represents text that should be entered verbatim. For instance, it can represent a command, as in the following example: "**You can instruct users how to run setup using a command such as setup.exe**."

1.3 The AI-205 Layer

AI-205 is an analog input layer with the following features:

- Maximum sampling rate of 250kHz per channel
- ±100V max input range
- 18-bit resolution, no missing codes
- Simultaneous sampling
- Polyphase filtering (three 128-tap hardware FIR filters with post-decimators and bypass mode)
- Four (4) analog input differential channels fully isolated from the system ground (up to 500V) – and isolated between channels (up to 350V)
- Gains of 1/10/100/1000, per-channel selectable, effective ranges: ±100V, ±10V, ±1V, ±0.1V
- Over-voltage protection (150V)
- 2kV Electrostatic Shock Discharge (ESD) protection
- Two bidirectional lines for every analog input channel. Line state is encoded into analog input data
- 2048 sample input FIFO with 32-bit per-sample timestamp
- Interrupt request on any position in the input or channel list FIFO
- On-layer EEPROM to store configuration and calibration data
- Power consumption ~ 1.6/2.2W

1.4 Specifications



Technical Specifications:

Max Sampling Rate	250 kHz/channel (before decimation)
Max Transfer Rate	100 kS/s
Number of Channels	4 (individual A/D per channel)
DIOs per channel	2
FIR Unit: Size Decimation Ratio Number of Taps FIR Coefficient	 3 stages 1-128 (default - 5) 128 per stage loadable
Onboard FIFO Size	2048 samples
Input Ranges	±100V, ±10V, ±1V, ±0.1V
Input Impedance	2 MΩ (to ground); 4 MΩ (differential)
Input Bias Current	±1.5 nA
Isolation	350V$_{rms}$ (between channels) 500V$_{rms}$ (to system ground)
Input Overvoltage	2000V ESD, ±150V overvoltage protection (powered or unpowered)
Total Harmonic Distortion	-100 dB
ENOBs @ G=1:	18 bits @ 10 Hz (filtered) 17.1 bits @ 60 kHz (native)
Signal/(N+D) ratio	100 dB
Integral Non-linearity	2.5 LSB
Channel Crosstalk	120 dB
Power Consumption	3W
Physical Dimensions	3.875 x 3.875"(98 x 98 mm)
Operating Temp. (tested)	-40°C to +85°C
Operating Humidity	90%, non-condensing

Figure 1-1. Technical Specifications

Figure 1-2 is a photo of the DNA-AI-205 Analog Input layer.

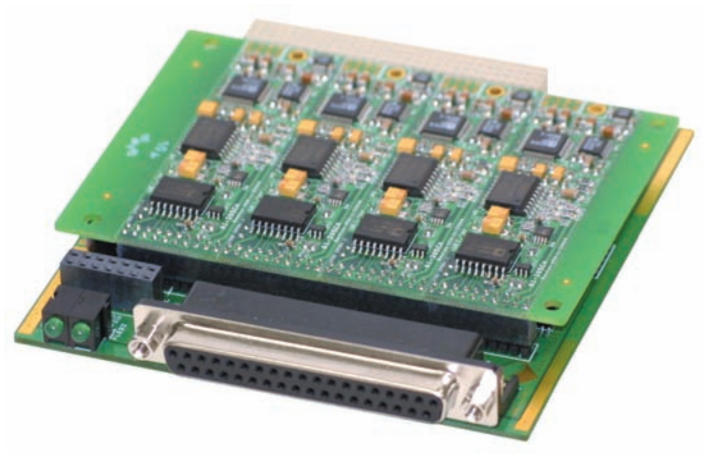


Figure 1-2. Photo of DNA-AI-205

1.5 Device Architecture

The AI-205 Layer has a PL-60x base (FPGA layer) and a 205 daughter card with A/D converters and optical isolation.

Every channel has a dedicated FIR unit. Each FIR unit consists of three identical in-hardware FIR filters and a decimator. An FIR filter has the ability to perform multiplication and accumulation operations in one clock cycle. Each FIR filter can be set into bypass state.

A FIR filter can accept up to 128 taps with 16-bit resolution. A Decimator can decimate output data with coefficients from 1 to 32. Please see the FIR section for further details.

The AI-205 employs a successive approximation 18-bit converter per channel with no pipeline delay.

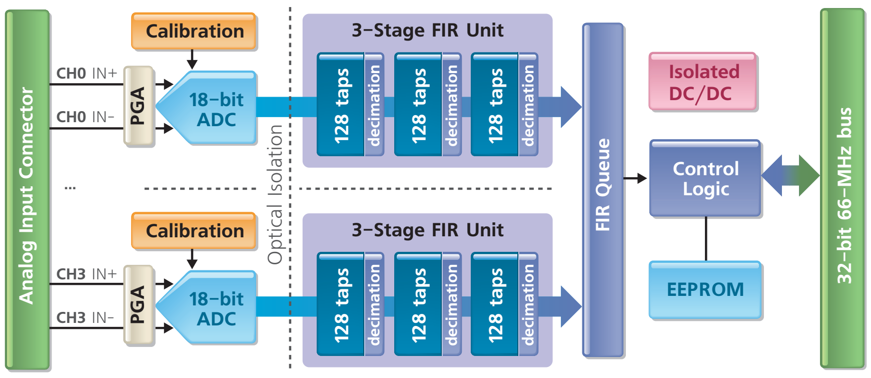


Figure 1-3. DNA-AI-205 Block Diagram

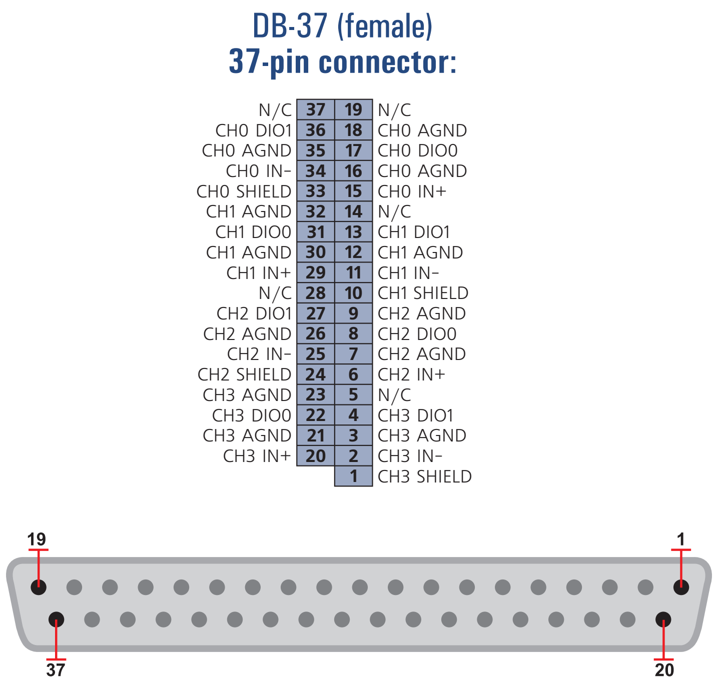
1.6 Layer Connectors and Wiring

Every layer can accept up to four differential signals. A signal line is marked as CHx IN+ and a return as CHx IN-. Because every layer is isolated and has a separate A/D converter, a single-ended configuration is not supported implicitly. If signals you wish to digitize are single-ended (have common ground), connect the return (CHx IN-) line to the common ground.

CHx AGND line is an isolated ground for this channel. Digital I/O lines are also referenced to this ground.

CHx DIO0 and CHx DIO1 are bidirectional digital lines. DIO0 is an output and DIO1 is an input by default.

CHx SHIELD is a special line to connect to the differential pair shield.The CHx SHIELD line always maintains a potential equal to a common-mode voltage.



N/C	37	19	N/C
CH0 DIO1	36	18	CH0 AGND
CH0 AGND	35	17	CH0 DIO0
CH0 IN-	34	16	CH0 AGND
CH0 SHIELD	33	15	CH0 IN+
CH1 AGND	32	14	N/C
CH1 DIO0	31	13	CH1 DIO1
CH1 AGND	30	12	CH1 AGND
CH1 IN+	29	11	CH1 IN-
N/C	28	10	CH1 SHIELD
CH2 DIO1	27	9	CH2 AGND
CH2 AGND	26	8	CH2 DIO0
CH2 IN-	25	7	CH2 AGND
CH2 SHIELD	24	6	CH2 IN+
CH3 AGND	23	5	N/C
CH3 DIO0	22	4	CH3 DIO1
CH3 AGND	21	3	CH3 AGND
CH3 IN+	20	2	CH3 IN-
		1	CH3 SHIELD

Figure 1-4. DNA-AI-205 Pinout

1.7 Layer Capabilities

The AI-205 layer is capable of acquiring analog input voltages in the ±100V range with gains of 1, 10, 100 and 1000.

The layer is capable of generating its own CL and CV clocks and trigger and can also accept an external trigger from the SYNCx bus only.

The layer does not have hardware capability of analog triggering at present, but will have a digital implementation (after conversion data analysis) in a future revision.

Table 1-1. Gains

Card	Gain	Range	Noise, LSB	Resolution, (Noise Limited at High Gains)
DNA-AI-205	1	± 100V	0.81	762µV
	10	±10V	1.05	76.2µV
	100	±1V	1.58	20µV
	1000	±100mV	3.32	10µV

Analog four-pole anti-aliasing filtering is tuned to provide roll-off at 150kHz (half of the maximum sampling frequency.)

1.8 Data Representation

The DNA-AI-205 layer is equipped with four 18-bit A/D converters. The layer can return 18-bit straight binary data in 32-bit words combined with levels on general-purpose digital I/O lines.

The 18-bit data is represented as follows:

Table 1-2. 18-bit Data Representation

Bit	Name	Description	Reset State
17-0	ADCDATA	Upper 18 bits of data, straight-binary	<pos>

<pos> represents a position in the output buffer. Upon reset, every entry in the output buffer is filled with its relative position number. If you start receiving consecutive data from the layer, such as 0,1,2, etc., it means that either the layer is not initialized properly or the layer is damaged.

To convert data into floating point, use the following formula (at a gain of 1):

```
Volts = (Raw & 3ffff) * (200V/2^18) – 100V
```

32-bit data has a different representation, as follows:

Table 1-3. 32-bit Data Representation

Bit	Name	Description	Reset State
31-28	CHN	AI-205 channel #	0
27	DIO1	Input level of DIO1 line. This line is an input by default	0
26	DIO0	Input level of DIO0 line. This line is an output by default, user should switch it to input before use	0
25-24	INFO	Additional information bits. Reserved, in the future will include status of level trigger	0
23-18	RSV	Reserved, should be ignored by user application	
17-0	ADCD	ADC conversion result	

Because all channels have a separate converter and, potentially, a different decimation ratio, the user application should rely on the CHN bits to determine the channel to which a data point belongs.

Chapter 2 Programming with the High Level API

This section describes how to program the PowerDNA AI-205 using the UeiDaq Framework API.

UeiDaq Framework is object oriented and its objects can be manipulated in the same manner from different development environments such as Visual C++, Visual Basic, or LabVIEW.

Although the following section focuses on the C++ API, the concept is the same no matter what programming language you use.

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

2.1 Creating a Session

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

```
CUeiSession session;
```

2.2 Configuring the Channels

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

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

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

For example, the following resource string selects analog input channels 0,2,3,4 on device 1 at IP address 192.168.100.2: “pdna://192.168.100.2/Dev1/Ai0,2,3,4”

The gain to apply on each channel is specified using low and high input limits.

For example, the AI-205 available gains are 1, 10,100,1000 and the maximum input range is [-100V, 100V].

To select the gain of 100, you need to specify input limits of [-1V, 1V].

```
// Configure channels 0,1 to use a gain of 100 in
// differential mode
session.CreateAIChannel(“pdna://192.168.100.2/Dev0/Ai0,1”, -1.0, 1.0,
UeiAIChannelInputModeDifferential);
```

2.3 Configuring the Timing

You can configure the AI-205 to run 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 AI-205 on-board clock.

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

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

2.4 Configuring the FIR Filters

Each AI-205 analog input channel is equipped with a three-stage FIR filter and decimator. You can control those filters using custom properties (Please read the “UeiDaq Framework User Manual Appendix B).

Note that each property must be written in the sequence described below:

- “**channel**”: An integer representing the channel for which you want to configure the FIR filter.
- “**stage**”: An integer set to 0, 1 or 2 representing the stage to configure for the selected channel.
- “**decimation**”: An integer representing the decimation for the selected stage.
- “**tap**”: An array of floating-point values representing the taps for the selected stage. The maximum number of taps is 128.

Note that setting a decimation value greater than 1 will slow down the rate at which your application will receive data from the AI-205. You need to adjust the session timeout parameter accordingly.

The following sample code shows how to program the first stage of the FIR filter on channel 0:

```
int firChannel = 0;
int firStage = 0;
int decimation = 1;
double taps[8]= {…};
MySession.SetCustomProperty(“channel”, sizeof(int), &firChannel);
MySession.SetCustomProperty(“stage”, sizeof(int), &firStage);
MySession.SetCustomProperty(“decimation”, sizeof(int), &decimation);
MySession.SetCustomProperty(“tap”, 8*sizeof(double), taps);
```

2.5 Reading Data

Reading data from the AI-205 is done using a reader object. There is a reader object to read raw data coming straight from the A/D converter. There is also a reader object to read data already scaled to volts.

The following sample code shows how to create a scaled reader object and read samples.

```
// Create a reader and link it to the session’s stream
CueiAnalogScaledReader reader(session.GetDataStream());

// read one scan, the buffer must be big enough to contain
// one value per channel
double data[2];
reader.ReadSingleScan(data);
```

2.6 Cleaning-up the Session

The session object will clean itself up when it goes out of scope or when it is destroyed. However, you can manually clean up the session (to reuse the object with a different set of channels or parameters).

```
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 the PowerDNA DAQBios protocol and also allows you to access device registers directly.

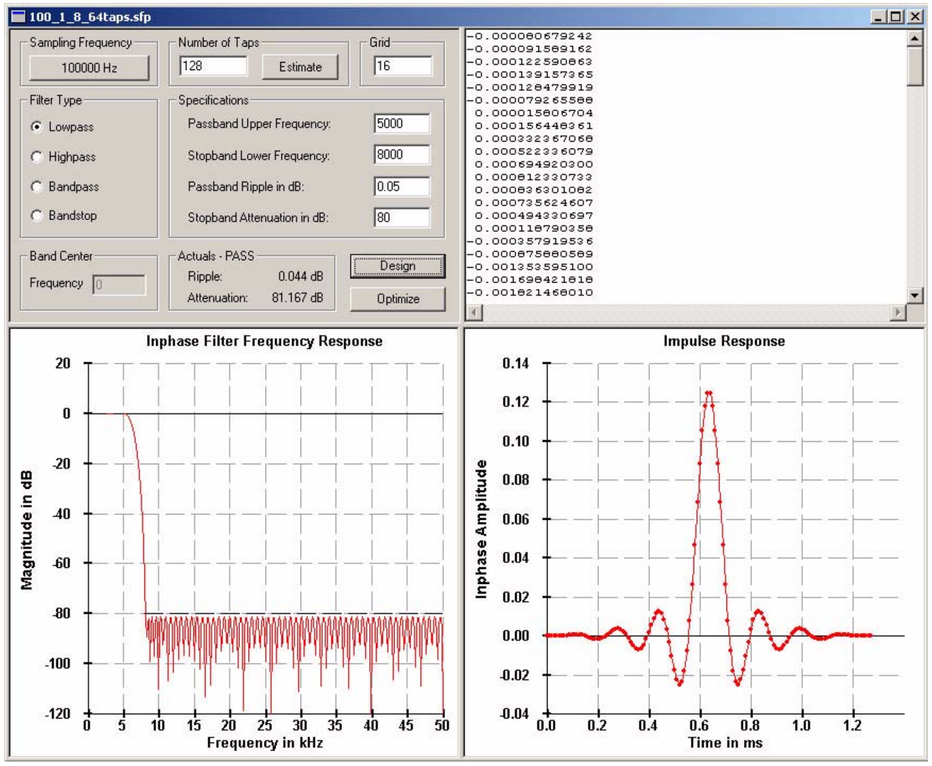
We recommend that you use the UeiDaq Framework (*see Chapter 2*), which is easier to use.

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

3.1 FIR[1] Programming

We recommend use of an external application capable of generating filter coefficients and visualizing filter data.

For example, you can use ScopeFIR from IOWegian Corp., which can produce a display as shown in **Figure 3-1**.



Sampling Frequency: 100000 Hz
Number of Taps: 128
Grid: 16
Filter Type: Lowpass
Specifications
Passband Upper Frequency: 5000
Stopband Lower Frequency: 8000
Passband Ripple in dB: 0.05
Stopband Attenuation in dB: 80
Band Center Frequency: 0
Actuals - PASS
Ripple: 0.044 dB
Attenuation: 81.167 dB

```
-0.000080679242
-0.000091589162
-0.000122590863
-0.0001391573 65
-0.000128479919
-0.000079265588
 0.000015806704
 0.000156448361
 0.000332367068
 0.000522336079
 0.000694920300
 0.000812330733
 0.000836301082
 0.000735624607
 0.000494330697
 0.000118790358
-0.000357919536
-0.000875880589
-0.001353595100
-0.001698421818
-0.001821468010
```

Figure 3-1. ScopeFIR Display of Low-Pass Filter Characteristics

To design an FIR filter, you should first decide on a base sampling frequency. The higher the base sampling frequency, the more taps you need to filter lower frequency signals. On the other hand, a higher sampling frequency allows better anti-aliasing filtering. A polyphase filter solves this problem.

1. Finite Input Response digital filter.

The second consideration is the filter type. A FIR filter can be programmed to perform low-pass, high-pass, band-pass and band-stop filtering. A Low-pass filter (shown) allows signals with frequencies lower than selected to pass. A High-pass filter does the opposite. A Band-pass filter allows only certain frequencies to pass through and a band-stop filter blocks them.

Pass-band frequency defines which frequencies the filter should pass without attenuation. Stop-band defines at what frequency the desired attenuation should be achieved. The filter requires more taps to achieve sharper roll-off curve.

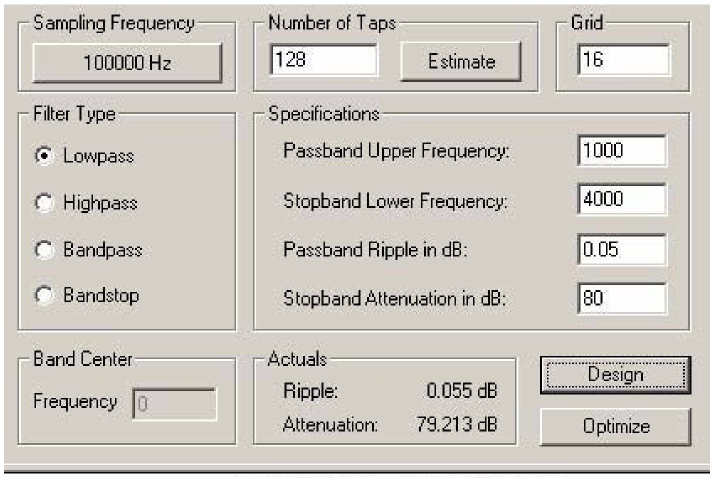
Pass-band ripple defines deviations of the signal amplitude on signal frequency. To decrease pass-band ripple, use more filter taps.

Stop-band attenuation defines the remaining signal level at stop-band frequencies.

To define a band-pass or a band-stop filter, you should define center frequency and pass-band width (symmetrical to center frequency).

Each AI-205 filter has 128 taps. It might not be enough to filter a user signal when a low pass-band frequency is required along with steep roll-off. To achieve this kind of filtering, you should design a polyphase filter in FIR unit filters and decimators.

For example, if you want to sample at 100 kHz (to avoid picking up aliases) but the signal of interest lies below 50Hz, use the filter displayed in **Figure 3-2**.



Sampling Frequency: 100000 Hz
Number of Taps: 128 Estimate
Grid: 16
Filter Type: [✓ Lowpass] Highpass Bandpass Bandstop
Specifications
Passband Upper Frequency: 1000
Stopband Lower Frequency: 4000
Passband Ripple in dB: 0.05
Stopband Attenuation in dB: 80
Band Center
Frequency 0
Actuals
Ripple: 0.055 dB
Attenuation: 79.213 dB
Design
Optimize

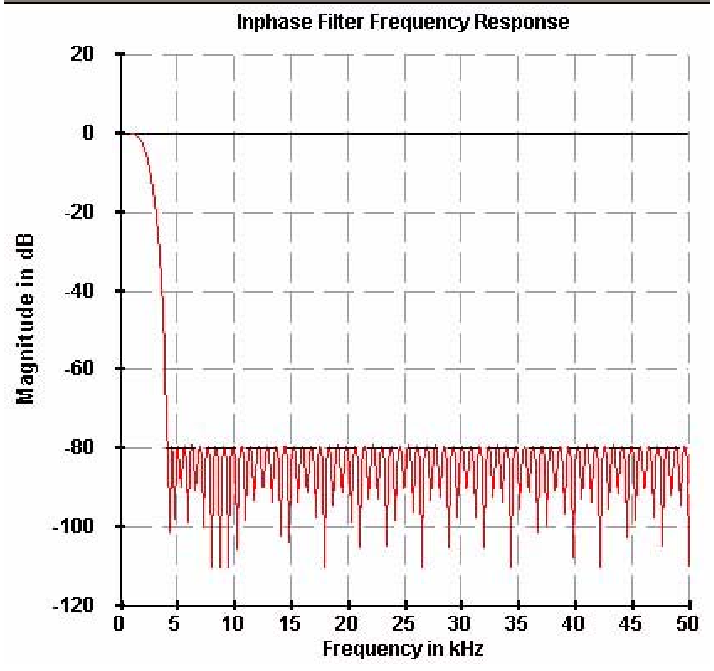


Figure 3-2. Polyphase Filter with Cutoff above 4 kHz

First, the FIR filter cuts out frequencies above 4 kHz.

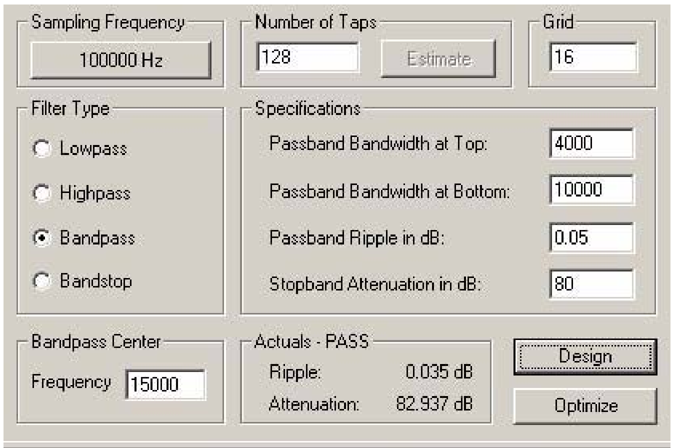
Now you can safely remove all the signal frequencies above 5 kHz by decimating the original signal with a decimation ratio of 10. The effective sampling rate becomes 10 kHz.

By setting the same parameters for the second and third filter in the FIR unit, the resulting signal will have a sampling frequency of 100Hz with 10Hz pass-band and 40Hz stop-band.

It is impossible to achieve this kind of filter characterizing using a single-phase filter with 3*128 = 384 taps. Polyphase filters have a significant advantage over single-phase filters with the same cumulative number of taps.

The PowerDNA API provides `DqAdv205LoadCoeff()`to set up filter coefficients for an FIR unit.

The following picture shows a band-pass filter. This filter allows signal frequencies from 13 kHz to 17 kHz to pass through the filter without attenuation and completely stops frequencies below 10 kHz and above 20 kHz.



Sampling Frequency: 100000 Hz

Number of Taps: 128

Grid: 16

Filter Type: ○ Lowpass ○ Highpass [✓ Bandpass] ○ Bandstop

Specifications:

Passband Bandwidth at Top:	4000
Passband Bandwidth at Bottom:	10000
Passband Ripple in dB:	0.05
Stopband Attenuation in dB:	80

Bandpass Center Frequency: 15000

Actuals - PASS

Ripple: 0.035 dB

Attenuation: 82.937 dB

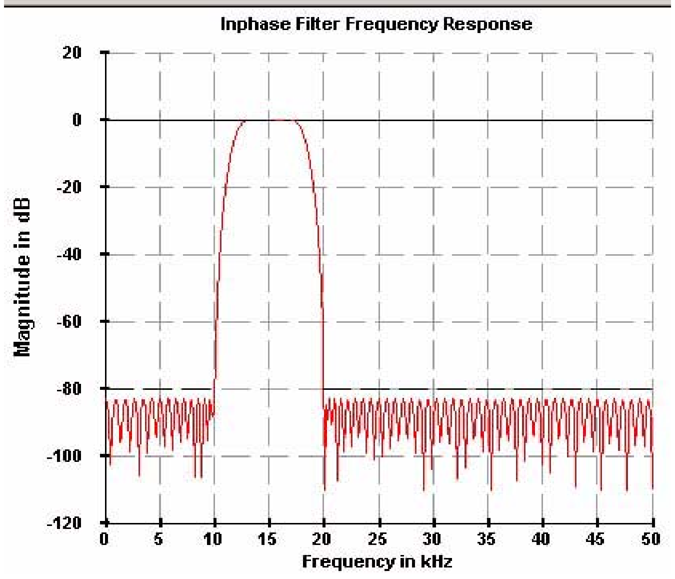


Figure 3-3. Band Pass Filter

Different channels can have different decimation ratios and different data output rates as a result.

On a hardware reset, filters are loaded with default filter coefficients and decimation ratios. A default filter has a pass-band of 20% of the sampling frequency with a stop-band at 25%. Stop-band attenuation is –80dB and pass-band ripple 0.00001 dB. Default decimation ratio is 5.

Thus, without changing filter settings, the AI-205 delivers data at 1/125 of the original frequency. For example, if the user sets the conversion clock at 125 kHz, the output data rate will be 1 kHz with a pass-band at 200Hz and a stop-band at 250Hz.

3.2 Configuration Settings

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

Not all configuration bits apply to the AI-205 layer.

The following bits are used:

```
#define DQ_FIFO_MODEFIFO (2L << 16) // continuous acquisition with
                                        // FIFO
#define DQ_LN_MAPPED     (1L<<15)   // For WRRD (DMAP) devices
#define DQ_LN_STREAMING  (1L<<14)   // For RDFIFO devices - stream the
                                     // FIFO data
                                    // automatically
                                   // 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_LN_ENABLE enables all operations with the layer
DQ_LN_CVCKSRC0 selects the internal channel list clock (CL) source as
a timebase. The AI-205 supports the CV clock.
DQ_LN_ACTIVE is needed to switch on “STS” LED on CPU layer.
```

The user can select either the CL or CV clock as a timebase. Because of the parallel architecture of AI-205 layer, either clock triggers all four converters.

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

Acquisition rate cannot be selected on per-channel basis. To select a different resulting rate for different channels, you should program proper decimators in the FIR unit.

3.3 Channel liSt Settings

AI-205 layer has a very simple channel list structure:

Table 3-1. Channel List Structure

Bit	Name	Purpose	Macro
31	DQ_LNCL_NEXT	Tells firmware that there is next entry in the channel list	
20	DQ_LNCL_TSRQ	Request timestamp as a next data point	
11..8		Gain	DQ_LNCL_GAIN()
7..0		Channel number	

Gains are different for different options of AI-205 layer

Layer Type	Range	Gain	Gain Number
DNA=AI-205	±100V	1	0
	±10V	10	1
	±1V	100	2
	±100V	1000	3

NOTE: Despite having the same conversion rate across all channels, data output rate can vary, depending on decimation. Even when the decimation ratio is identical, there is no guarantee in which order data will be put into the output buffer. That's why you should not rely on the order of channels he specified in the channel list, but rather strip output data into channel data based on channel numbers embedded in the data itself.

3.4 Layer-specific Commands and Parameters

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

- **DqAdv205Read**()

  This function works using the underlying `DqReadAIChannel()`, but converts data using internal knowledge of input range and gain of every channel. It uses the `DQCMD_IOCTL` command with the `DQIOCTL_CVTCHNL` function under the hood.

  When this function is called for the first time, the firmware stops any ongoing operation on the device specified and reprograms it accordingly with the channel list supplied. The function uses the preprogrammed CL update frequency – 10Hz. You can reprogram the update frequency by calling `DqCmdSetClk()`after the first call to `DqAdv205Read()`.

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

If you specify a short timeout delay, this function can time out when called for the first time, because it is executed as a pending command and layer programming takes up to 10ms.

Once this function is called, the layer continuously acquires data and every call to the function returns the latest acquired data.

If you would like to cancel ongoing sampling, call the same function with 0xffffffff as a channel number.

- **DqAdv205LoadCoeff**()

  This function loads the coefficient table.

3.5 Data Represent-ation in ACB and DMap

Output layer data are presented in ACB in raw or floating point format in voltages. Raw data are represented as 32-bit words and floating point data are represented as structures.

```
typedef struct {
    uint32 raw;// raw part of the data including channel number
    double data;// converted data
} DQly205_double, *pDQly205_double;
```

3.6 Using Layer in ACB Mode

This is a pseudo-code example that highlights the functions needed in sequence to use ACB on the 205 layer. A complete example with error checking can be found in the directory *SampleACB205*.

Note: the AI-205 layer is not guaranteed to return the channels in the correct order. This is why data returned from the `DqAcbGetScansCopy()` function is an array of `DQly205_float` structures, which contain both the raw and converted channel values. The upper 8 bits of the raw value tell you which channel the value came from. Our example does not use this value to sort the data; it just dumps out the data in the order received.

```
#include "PDNA.h"

// unit configuration word
#define CFG205          (DQ_LN_ENABLED \
                        |DQ_LN_ACTIVE \
                        |DQ_LN_GETRAW \
                        |DQ_LN_IRQEN \
                        |DQ_LN_CVCKSRC0 \
                        |DQ_LN_STREAMING \
                        |DQ_AI205_MODEFIFO)
    uint32 Config = CFG205;
```

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,
                 &CVSize,
                 0,      //float*  fCLClk,
                 CL,
                 0,      //uint32* ScanBlock,
                 &acb);

    printf("Actual clock rate: %f\n", fCVClk);

    // Now set up events
    DqeSetEvent(bcb,
DQ_eFrameDone|DQ_ePacketLost|DQ_eBufferError|DQ_ePacketOOB);
```

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) {
            minrq = acb.framesize;
            avail = minrq;
            while (TRUE) {
                DqAcbGetScansCopy(bcb, data, acb.framesize,
acb.framesize,
                  &size, &avail);
                samples += size*CHANNELS;

                for (i = 0; i < size * CHANNELS; i++) {
                    fprintf(fo, "%f\t", ((DQly205_float*)data + i)-
>data);
                    if ((i % CHANNELS) == (CHANNELS - 1)) {
                         fprintf(fo, "\n");
                    }
                }

                printf("eFD:%d scans received (%d samples) min=%d
avail=%d\n", size,
                  samples, minrq, avail);
                if (avail < minrq) {
                    break;
                }
            }
        }
    }
```

STEP 5: Stop operation.

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

STEP 6: Clean up.

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

3.7 Using 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|DQ_ePacketOOB);
```

STEP 4: Start operation.

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

STEP 5: Process data.

```
    while (keep_looping) {

        ret = DqeWaitForEvent(&pBcb, 1, FALSE, timeout, &eventsin);

        if (eventsin & DQ_eDataAvailable) {
            // read input or write output here
            for (i = 0; i < CHANNELS; i++) {
                printf("%08x ", *(uint32*)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
```

Appendix

A. Appendix A - Accessories

The following cables and STP boards are available for the AI-205 layer.

DNA-CBL-37

3ft, 37-way flat ribbon cable; connects DNA-AI-205 to panels

DNA-STP-37

37-way screw terminal panel

DNA-STP-37D

37-way direct-connect screw terminal panel

B. Appendix B – Calibration

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

Calibration should be performed with a microvolt-resolution precision voltage source with low (1Ohm or less) output impedance. Calibration assumes use of the single-ended mode. Signals are measured relative to AGND.

To perform layer calibration, you should have a precision voltage source attached to all four channels and running a serial terminal program attached to the IOM serial port.

Use the “simod 1” command to calibrate the layer. Calibration is performed at an input gain of 10 to minimize requirements to the voltage source equipment.

Calibration Procedure

The recommended calibration procedure (using a serial port terminal program) is as follows:

STEP 1: Apply 0V on all channels

STEP 2: Type “simod 1”

STEP 3: Select the proper device from the device table to be calibrated.

STEP 4: Select calibration DAC 2 to calibrate offset on the first channel.

STEP 5: Adjust offset by pressing “[” and “]” keys (current DAC values are displayed). Use “{” and “}” keys to decrease or increase value of calibration DAC by 10.

STEP 6: Use calibration DAC4 for the second channel, DAC6 for the third and DAC8 for the fourth.

STEP 7: Apply 9.9V.

STEP 8: Select calibration DAC 1 to calibrate gain on the first channel.

STEP 9: Adjust gain the same way as offset.

STEP 10: Use calibration DAC3 for the second channel, DAC5 for the third and DAC7 for the fourth.

STEP 11: Press “Esc” and reply “y” if you want to save calibration values into E^2PROM.

STEP 12: Reset the PowerDNA cube to verify calibration.

Notes:

1. The calibration program uses FIR and MAW filters to improve resolution. Press “c” to clear filter history

2. We recommend you calibrate offset by applying 0 volts from the signal source rather than by shorting inputs
3. We recommend calibrating layer gain as close to the end of the scale as possible. A 9.9V calibration point is ideal for layer calibration.
4. You can verify calibration after resetting the PowerDNA cube using the same “simod 1” routine, but do not save results. “simod 2” shows raw acquired data without filtering.
5. For AI-205 layers, we recommend annual factory recalibration at UEI

Index