United Electronic Industries

The High-Performance Alternative

DNx-MUX-461
User Manual

26-Channel Multiplexer Board
for the PowerDNA Cube and RACK series chassis

September 2021

PN Man-DNx-MUX-461

CE

Contacting United Electronic Industries

Mailing Address:
249 Vanderbilt Avenue
Norwood, MA 02062
U.S.A.

Shipping Address:
24 Morgan Drive
Norwood, MA 02062
U.S.A.

For a list of our distributors and partners in the US and around the world, please contact a member of our support team:

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
Website: 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.

Specifications in this document are subject to change without notice. Check with UEI for current status.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

This manual outlines the feature set and use of the DNx-MUX-461, a 26-channel multiplexer designed for a variety of switching and digital control applications.

The following sections are provided in this chapter:

- Organization of this Manual (Section 1.1)
- Manual Conventions (Section 1.2)
- Naming Conventions (Section 1.3)
- Related Resources (Section 1.4)
- Before You Begin (Section 1.5)
- DNx-MUX-461 Features (Section 1.6)
- Technical Specifications (Section 1.7)

1.1 Organization of this Manual

This DNx-MUX-461 User Manual is organized as follows:

- **Introduction**
  Chapter 1 summarizes the features and specifications of the DNx-MUX-461.
- **Device Overview**
  Chapter 2 describes the device architecture, logic, and connectivity of the DNx-MUX-461.
- **PowerDNA Explorer**
  Chapter 3 shows how to explore DNx-MUX-461 features through a GUI-based application.
- **Programming with High-level API**
  Chapter 4 describes how to create a session, configure the session, and interpret results with the Framework API.
- **Programming with Low-level API**
  Chapter 5 provides an overview of PowerDNA commands for configuring and using the DNx-MUX-461.
- **Appendix A - Accessories**
  This appendix provides a list of accessories available for use with the DNx-MUX-461.

1.2 Manual Conventions

The following conventions are used throughout this manual:

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

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

NOTE: Notes alert you to important information.

Typeface	Description	Example
bold	field or button names	Click **Scan Network**
»	hierarchy to get to a specific menu item	**File » New**
`fixed`	source code to be entered verbatim	`session.CleanUp()`
<brackets>	placeholder for user-defined text	`pdna://<IP address>`
italics	path to a file or directory	*C:/Program Files*

1.3 Naming Conventions

The DNA-MUX-461, DNR-MUX-461, and DNF-MUX-461 board versions are compatible with the UEI Cube, RACKtangle, and FLATRACK chassis respectively. These boards are electronically identical and differ only in mounting hardware. The DNA version stacks in a Cube chassis, while the DNR and DNF versions plug into the backplane of a Rack chassis. Throughout this manual, the term DNx-MUX-461 refers to both Cube and Rack products.

1.4 Related Resources

This manual only covers functionality specific to the DNx-MUX-461. To get started with the UEI IOM, please see the documentation included with the software installation. On Windows, these resources can be found from the desktop by clicking **Start » All Programs » UEI**

UEI's website includes other user resources such as application notes, FAQs, tutorials, and videos. In particular, the glossary of terms may be helpful when reading through this manual: https://www.ueidaq.com/glossary

Additional questions? Please email UEI Support at support@ueidaq.com or call 508-921-4600.

1.5 Before You Begin

No Hot Swapping!

Before plugging any I/O connector into the Cube or RACKtangle, be sure to remove power from all field wiring. Failure to do so may cause severe damage to the equipment.

Check Your Firmware

Ensure that the firmware installed on the Cube or Rack CPU matches the UEI software version installed on your PC. The IOM is shipped with pre-installed firmware and a matching software installation. If you upgrade your software installation, you must also update the firmware on your Cube or RACK CPU. See “FirmwareUpdatingProcedures” for instructions on checking and updating the firmware. These instructions are located in the following directories:

- On Linux: *<PowerDNA-x.y.z>/docs*
- On Windows: **Start » All Programs » UEI » PowerDNA » Documentation**

1.6 DNx-MUX-461 Features

The DNx-MUX-461 Multiplexer Board is a high-speed switch interface designed to increase the number of inputs connected to a measurement instrument such as the DNx-DMM-261 multimeter. Its features include:

- 26 two-wire or 13 four-wire channels
- Connects to DMM-261 without external wiring
- ±170 VDC or VAC maximum operating voltage
- 0.5 Ω resistance (not including cabling)
- 500mA continuous load current rating
- Relay operation counter tracks relay life cycles
- 500Hz update rate

1.6.1 Channel Configuration

The DNx-MUX-461 multiplexes 26 two-wire or 13 four-wire channels to a 6-pin digital multimeter. A channel can be programmed for Voltage, Current, 2-Wire Resistance, or 4-Wire Resistance measurement mode. When used with the DNx-DMM-261, all DMM connections are made inside the Cube or RACKtangle so the only connections you need to make are to the various input channels.

1.6.2 Switch Conditions

The board employs reed relays that support switching rates of up to 500 Hz. Each channel is capable of switching voltages up to ±170 VDC or AC waveforms with peaks less than ±170 VDC.

Each channel is rated for continuous operation at 500 mA DC or AC rms with a switch resistance of less than 0.5 Ω (typical, not including external cables).

A break-before-make ensures that only one channel is closed at a time. All relays default to “open” on power up/reset.

1.6.3 Switch Counter

A built-in counter counts the number of switch cycles for each relay, allowing the age of contacts to be tracked. Each relay is rated for 1 million cycles at 24VDC/ 50 mA or 12VDC/100 mA.

1.6.4 Synchro-nization

The DNx-MUX-461 can synchronize relay switching via the SYNC IN and SYNC OUT pins. The SYNC OUT pin can be configured to change state when programming a relay state change, and the SYNC IN pin can be used to initiate channel switches.

1.6.5 Diagnostics

Users have the capability of reading internal supply voltages, the relay driver voltage, and onboard temperature. Users can also read back the current state and status of each relay.

1.6.6 Isolation & Over-voltage Protection

The DNx-MUX-461 offers 350 Vrms of isolation between itself and other I/O boards as well as between the I/O connections and the chassis.

1.6.7 Environmental Conditions

Like all UEI I/O boards, the board offers operation in extreme environments and has been tested to 5g vibration, 100g shock, from -40 to +85 °C temperatures and will function at altitudes up to 70,000 feet.

1.6.8 Accessories

All field-wiring connections are made through a standard 62-pin D connector, allowing OEM users to build custom cabling systems through off-the-shelf components. Users may also connect the DNx-MUX-461 board to UEI's DNA-STP-62 screw terminal panel via the DNA-CBL-62 cables.

1.6.9 Software Support

The DNx-MUX-461 includes a software suite supporting Windows, Linux, QNX, VXWorks, RTX, and most other popular real-time operating systems. Windows users may use the UEIDAQ Framework, which provides a simple and complete software interface to Windows programming languages and DAQ applications (e.g., LabVIEW, MATLAB). All software support includes extensive example programs that make it easy to cut-and-paste the I/O software into your applications.

1.7 Technical Specifications

Table 1-1 lists the technical specifications for the DNx-MUX-461 board. All specifications are for a temperature of 25°C unless otherwise stated.

Table 1-1 Technical Specifications

Output configuration	26 two-wire or 13 four-wire multiplexers
Output Specifications	
Rated load (switching)	500mA (-40 to +85°C)
Rated load (carry)	1 A
Max operating voltage	170 VDC or VAC (other version available as special order)
Contact type	Reed relay
Contact ON impedance	0.5 Ohm max (at the I/O connector)
Contact OFF impedance	>10 GOhm
OFF leakage current	< 20 nA
Max update rate	500 Hz
Turn-off time	< 0.35 ms typical
Turn-on tme	< 0.25 ms typical
Max operating rate	500 Hz
Rated contact life	1 million operations at 24VDC / 50 mA 1 million operations at 12 VDC / 100 mA
Power up / reboot state	All switches off
Power dissipation	< 5 W not including output switching
Isolation	350 Vrms
Operating temperature range	Tested -40 to +85°C
Operating humidity	95%, non-condensing
Vibration IEC 60068-2-6 IEC 60068-2-64	5 g, 10-500 Hz, sinusoidal 5 g (rms), 10-500 Hz, broadband random
Shock IEC 60068-2-27	100 g, 3 ms half sine, 18 shocks @ 6 orientations 30 g, 11 ms half sine, 18 shocks @ 6 orientations
MTBF	tbd

Chapter 2 Device Overview

This section describes the device architecture and hardware of the DNx-MUX-461 Multiplexer Board. The following sections are provided in this chapter:

- Relay Architecture (Section 2.1)
- Break-before-Make (Section 2.2)
- Synchronizaton (Section 2.3)
- Diagnostics (Section 2.4)
- Cascading Multiplexers (Section 2.5)
- Indicators and Connectors (Section 2.6)
- Pinout (Section 2.7)

2.1 Relay Architecture

Figure 2-4 depicts the relay architecture within the DNx-MUX-461. The relays are categorized as “A” relays, “B” relays, or “D” relays. A and B relays select the desired channel, while D relays route the channel to the DMM pins. The relays are SPST (Form A) with +/- pairs controlled by the same signal. Users can configure the DC/DC voltage used to switch and hold the relays.

The board provides 26 two-wire channels evenly divided among the A and B bus. An A and B channel pair may be used together as a four-wire channel, therefore providing up to 13 four-wire channels.

When programming the relays, users select a channel number between 0...12, the A or B bus (for two-wire mode only), and a DMM Mode. **Table 2-1** summarizes which relays are closed for the programmed DMM Mode and channel. Note that in “4-Wire Resistance Mode”, the B bus is used as the sense line and the A bus is used as the excitation line.

Please refer to Chapter 4 and Chapter 5 for more information about programming relays.

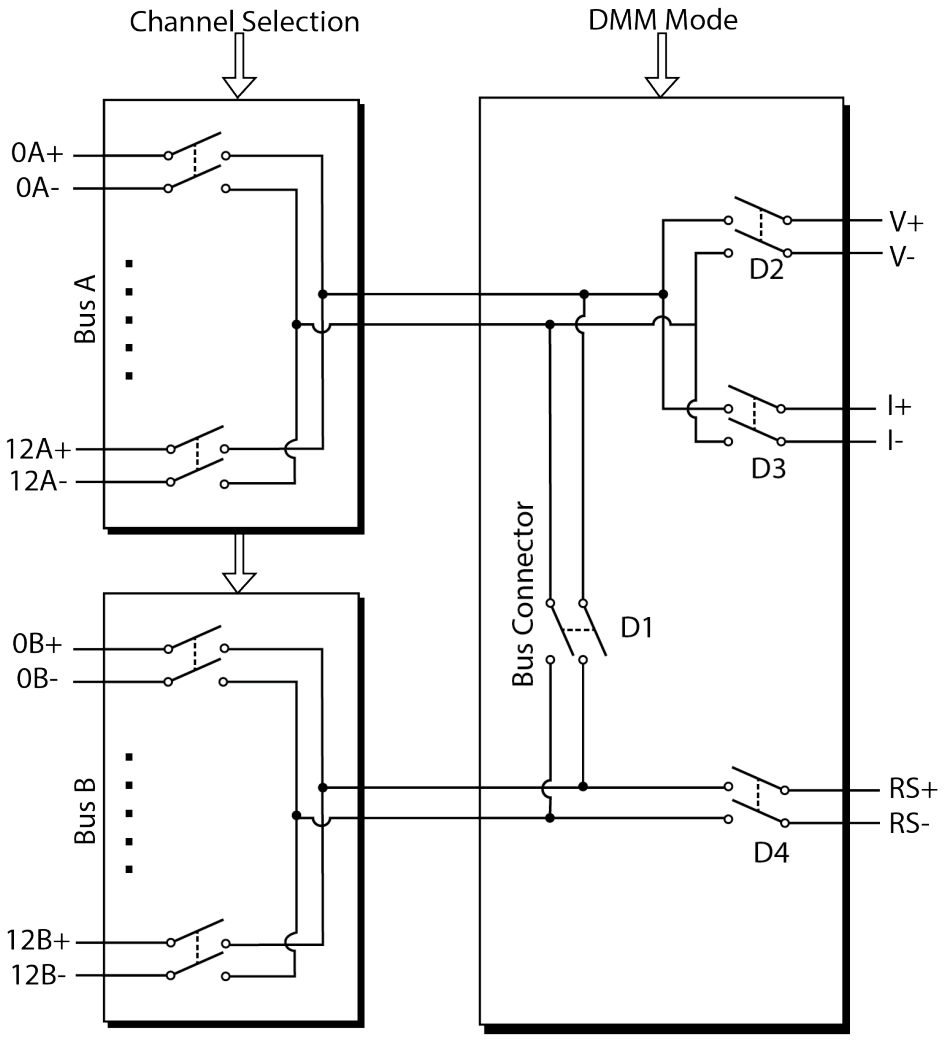


Figure 2-1 Simplified Schematic of DNx-MUX-461

Table 2-1 Closed Relays for Selected DMM Mode and Channel (n = 0...12)

DMM Mode	nA	nB	D1	D2	D3	D4
Voltage	●			●		
Voltage		●	●	●		
Current	●				●	
Current		●	●		●	
2-Wire Resistance	●			●		
2-Wire Resistance		●	●	●		
4-Wire Resistance	●	●		●		●

2.2 Break-before-Make

By default, when users write a new relay state, all closed relays open before the programmed relay is closed. This functionality ensures that only one channel is connected to the DMM at a time. The break-before-make time (i.e., the time when all relays are open) is user configurable.

Break-before-make can be disabled for D relays only. This setting is useful if the DMM Mode stays the same, as it can extend the lifetime of these relays.

2.3 Synchro-nizaton

Users have access to synchronization functionality via the SYNC IN and SYNC OUT pins on the DB-62 connector or via internal chassis bus lines.

The SYNC OUT pin can be programmed to change state when the relays change state. The pulse width and pulse type are user configurable.

If sync in functionality is enabled, relay state changes are delayed until a pulse is received on the SYNC IN pin. Triggering on a level or edge change, as well as the polarity (high/rising or low/falling), is user configurable.

2.4 Diagnostics

Users can read back the following diagnostic information from the DNx-MUX-461:

- Voltage of internal 24V supply
- Voltage of internal 3.3V supply
- Internal relay driver voltage
- Onboard temperature
- Status of the relay write
- Current state of relays and SYNC IN pin
- Number of times relays have been energized

2.5 Cascading Multiplexers

Multiple DNx-MUX-461 boards can be cascaded to increase the total number of channels. Up to five of the DNx-MUX-461 boards may be daisy chained within the Cube/RACK chassis providing up to 130 two-wire or 65 four-wire channels in a single chassis. Larger systems are possible though will require the DNx-DMM-261 to DNx-MUX-461 interconnection external to the chassis itself.

The example shown in **Figure 2-2** expands the channel count to 52 two-wire channels or 26 four-wire channels. Because four-wire mode always expects the Sense lines on relay bus B, RS+ and RS- outputs on mux #1 and #2 are wired to the B inputs on mux #3.

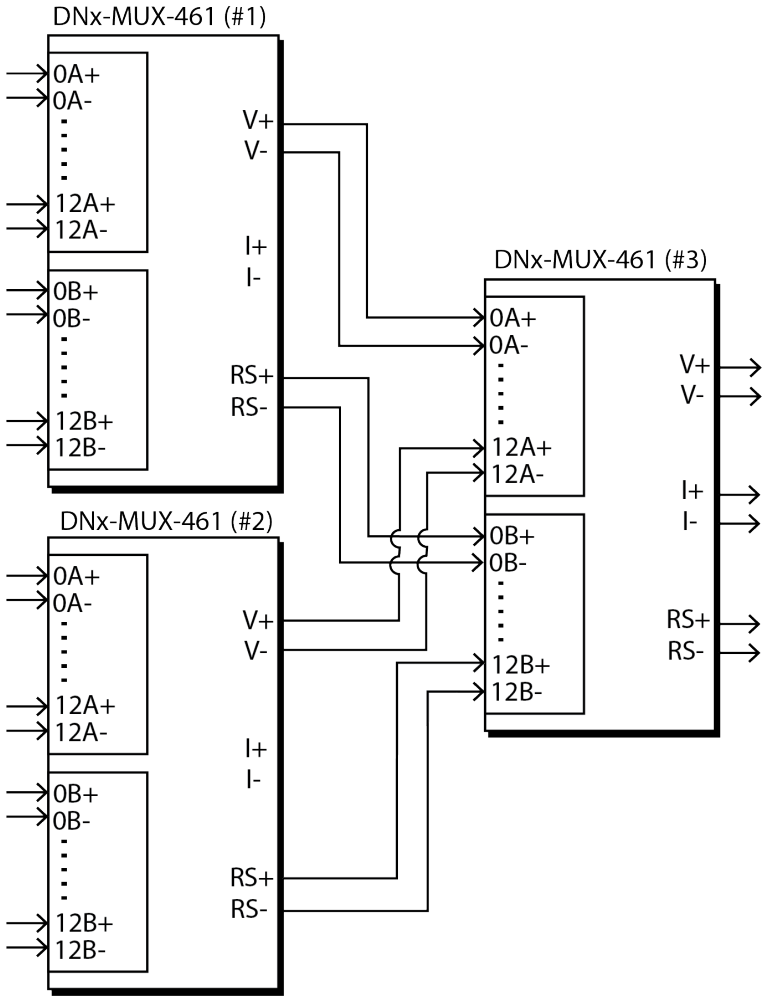


Figure 2-2 Cascading DNx-MUX-461 Boards

Example 1: Current Measurement on Channel 51

Program mux #2 for relay 12B, Voltage Mode. Then, program mux #3 for relay 12A, Current Mode. The device under test (DUT) will now be connected to DMM I+/-.

Example 2: Four-Wire Resistance Measurement on Channel 12

Program mux #1 to close relays 12A and 12B, Four-Wire Resistance Mode. Then, program mux #3 to close relays 0A and 0B, Four-Wire Resistance Mode. The excitation and sense leads on the DUT will now be connected to DMM V+/- and DMM RS+/- respectively.

2.6 Indicators and Connectors

Figure 2-3 shows the locations of the LEDs and connectors on the DNx-MUX-461. The LED indicators are described below in **Table 2-2**.

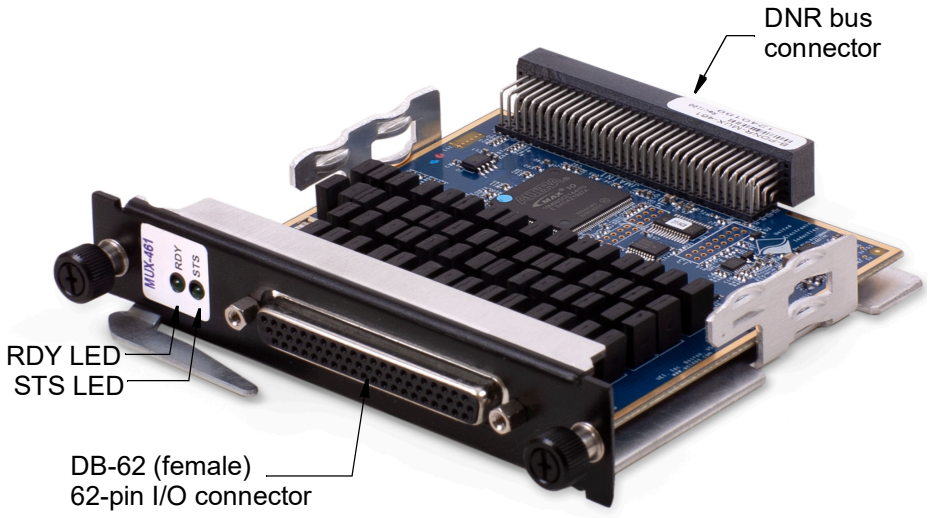


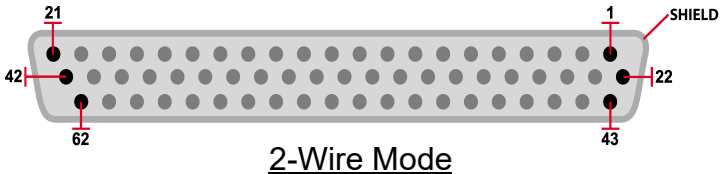
Figure 2-3 Photo of DNR-MUX-461 Board

Table 2-2 DNx-MUX-461 LED Indicators

LED Name	Description
RDY	Board is powered up and operational
STS	reserved

2.7 Pinout

Figure 2-4 illustrates the pin configuration for the DNx-MUX-461 board. External connections are made through a standard DB-62 female connector. If using the DNx-MUX-461 with the DNx-DMM-261, the chassis is shipped with DMM connections made internally.



2-Wire Mode

Pin	*Signal*	*Pin*	*Signal*	*Pin*	*Signal*
1	Sync +3.75V	22	Sync Gnd	43	IN 12-
2	Sync Out	23	Sync In	44	IN 12+
3	IN 25+	24	IN 25-	45	IN 24-
4	IN 11+	25	IN 11-	46	IN 24+
5	IN 10+	26	IN 10-	47	IN 9-
6	IN 23+	27	IN 23-	48	IN 9+
7	IN 22+	28	IN 22-	49	IN 21-
8	IN 8+	29	IN 8-	50	IN 21+
9	IN 7+	30	IN 7-	51	IN 6-
10	IN 20+	31	IN 20-	52	IN 6+
11	IN 19+	32	IN 19-	53	IN 18-
12	IN 5+	33	IN 5-	54	IN 18+
13	IN 4+	34	IN 4-	55	IN 3-
14	IN 17+	35	IN 17-	56	IN 3+
15	IN 16+	36	IN 16-	57	IN 15-
16	IN 2+	37	IN 2-	58	IN 15+
17	IN 1+	38	IN 1-	59	IN 0-
18	IN 14+	39	IN 14-	60	IN 0+
19	IN 13+	40	IN 13-	61	DMM V-
20	DMM RS-	41	DMM RS+	62	DMM V+
21	DMM I-	42	DMM I+		

4-Wire Mode

Pin	*Signal*	*Pin*	*Signal*	*Pin*	*Signal*
1	Sync +3.75V	22	Sync Gnd	43	IN 12A-
2	Sync Out	23	Sync In	44	IN 12A+
3	IN 12B+	24	IN 12B-	45	IN 11B-
4	IN 11A+	25	IN 11A-	46	IN 11B+
5	IN 10A+	26	IN 10A-	47	IN 9A-
6	IN 9B+	27	IN 10B-	48	IN 9A+
7	IN 19B+	28	IN 9B-	49	IN 8B-
8	IN 8A+	29	IN 8A-	50	IN 8B+
9	IN 7A+	30	IN 7A-	51	IN 6A-
10	IN 7B+	31	IN 7B-	52	IN 6A+
11	IN 6B+	32	IN 6B-	53	IN 5B-
12	IN 5A+	33	IN 5A-	54	IN 5B+
13	IN 4A+	34	IN 4A-	55	IN 3A-
14	IN 4B+	35	IN 4B-	56	IN 3A+
15	IN 3B+	36	IN 3B-	57	IN 2B-
16	IN 2A+	37	IN 2A-	58	IN 2B+
17	IN 1A+	38	IN 1A-	59	IN 0A-
18	IN 1B+	39	IN 1B-	60	IN 0A+
19	IN 0B+	40	IN 0B-	61	DMM V-
20	DMM RS-	41	DMM RS+	62	DMM V+
21	DMM I-	42	DMM I+		

Figure 2-4 Pinout Diagram for DNx-MUX-461

Table 2-3 DNx-MUX-461 Pinout Descriptions

Pin Name	2-Wire Mode	4-Wire Mode
DMM I+ DMM I-	DMM current measurement pins	n/a
DMM V+ DMM V-	DMM voltage measurement pins	DMM excitation current pins
DMM RS+ DMM RS-	n/a	DMM voltage sense pins
In 0+...25+ In 0-...25-	Connect to device under test (DUT)	n/a
In 0A+...12A+ In 0A-...12A-	n/a	Connect to Excitation leads on DUT
In 0B+...12B+ In nB-...12B-	n/a	Connect to Sense leads on DUT
Sync In	Synchronization input signal	
Sync Out	Synchronization output signal	
Sync +3.75V	Outputs constant +3.75V (for diagnosing power supply)	
Sync Gnd	Return for Sync In, Sync Out, and Sync +3.75V	

No Hot Swapping!

Before plugging any I/O connector into the Cube or RACKtangle, be sure to remove power from all field wiring. Failure to do so may cause severe damage to the equipment.

Unused Pins

Unused pins can be left open/disconnected.

Chapter 3 PowerDNA Explorer

This chapter provides the following information about exploring the DNx-MUX-461 with the PowerDNA Explorer application.

- Introduction (Section 3.1)
- Selecting a Channel (Section 3.2)
- Configuring the Channel (Section 3.3)
- Configuring Initialization State (Section 3.4)

3.1 Introduction

PowerDNA Explorer is a GUI-based application for communicating with your RACK or Cube system. You can use it to start exploring a system and individual boards in the system. PowerDNA Explorer can be launched from the Windows startup menu:

Start » All Programs » UEI » PowerDNA » PowerDNA Explorer

When using PowerDNA Explorer to configure DNx-MUX-461 boards, resetting the IOM or changing the DNx-MUX-461 configuration outside of PowerDNA Explorer (e.g., via C code or Labview) is not recommended; PowerDNA Explorer will not display changed parameters until **Scan Network** or **Reload Configuration** is clicked again (see **Figure 3-1** for button locations).

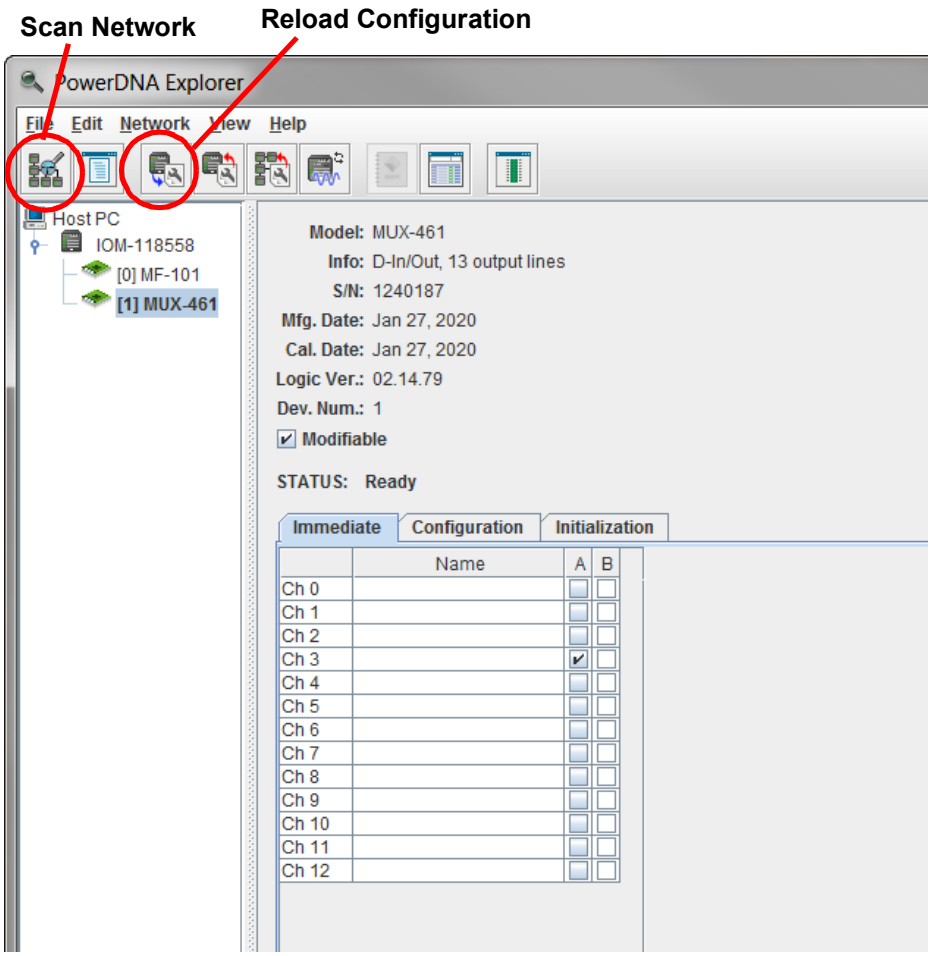


Figure 3-1 PowerDNA Explorer for DNx-MUX-461

When using PowerDNA Explorer for the DNx-MUX-461, the right-hand panel contains three tabs:

- **Immediate**: Select a multiplexer channel for immediate output.
- **Configuration**: Configures the channel in the Immediate tab.
- **Initialization**: Configures the initial state of the board at power-up.

Pressing **Reload Configuration** will read the current board state, which is useful if you just restarted PowerDNA Explorer.

NOTE: The DNx-MUX-461 is supported in PowerDNA version 5.0+.

3.2 Selecting a Channel

The **Immediate** tab, shown **Figure 3-2** below, contains the following columns:

- **Ch X**: read-only display of the channel number.
- **Name**: a name or note that you wish to give to the channel.
- **A** and **B**: Column A is active when relay bus A has been selected in the **Configuration** tab (Section 3.3). Column B is active when relay bus B has been selected. You may check off ONE box in this grid of 26 channels. The DNx-MUX-461 only allows one channel to be connected at any given time.

NOTE: When DMM Mode is configured for a 4-Wire resistance measurement, both A and B relays are closed for a particular channel number, even if Explorer only shows one checkmark. It does not matter whether you check off a box in the A or B column. The other will be automatically closed.

You must save the configuration in order for the new settings to take effect. To save the configuration, click the **Store Configuration** button.

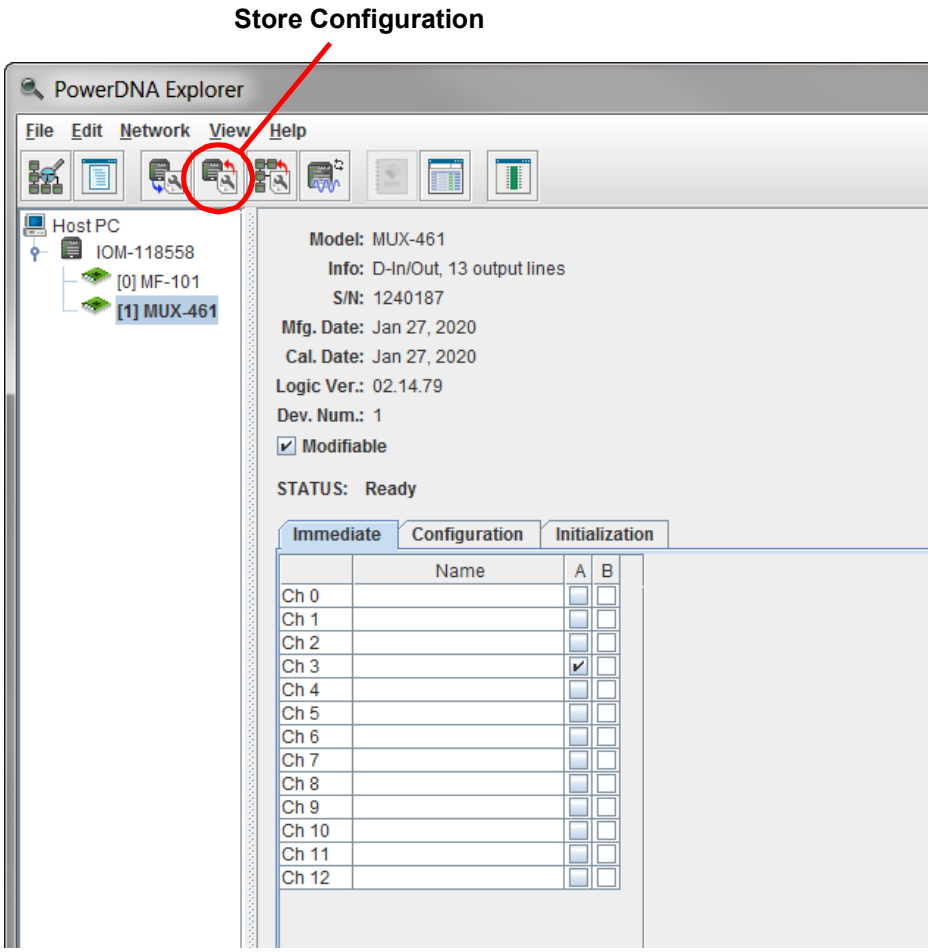


Figure 3-2 PowerDNA Explorer for DNx-MUX-461: Immediate Tab

3.3 Configuring the Channel

The **Configuration** tab, shown in **Figure 3-3**, includes the following configuration parameters:

- **DMM Mode**: controls which DMM pins the channel is connected to (switches “D” relays). If you change the DMM Mode but keep the same channel number, you will need to deselect and reselect the channel number in the **Immediate** tab in order for the command to take effect.
- **Select Relays**: allows channel to be selected on the A bus or B bus (ignored for 4-wire DMM Mode). Set the channel number in the **Immediate** tab as described in Section 3.2.
- **Relay Holding Voltage**: DC/DC voltage used for relay holding.
- **Relay Switching Voltage**: DC/DC voltage used for relay switching.
- **Break Before Make:** enables/disables break-before-make on D relays only. Break-before-make is always enabled for A and B relays.
- **On Delay**: breaking time of break-before-make in units of 100us.
- **Off Delay**: time before a new channel can be programmed, in units of 100us.
- **DI Mode**: gate relay switching with a level or edge on the SYNC IN pin.
- **DI Polarity**: polarity of the SYNC IN trigger selected in DI Mode.
- **Sync Out PW**: Pulse width for SYNC OUT modes 6 and 7.
- **Sync Out Mode**: signal to output on SYNC OUT pin.

You must save the configuration in order for the new settings to take effect. To save the configuration, click the **Store Configuration** button. To read back the configuration to confirm the settings, click the **Reload Configuration** button.

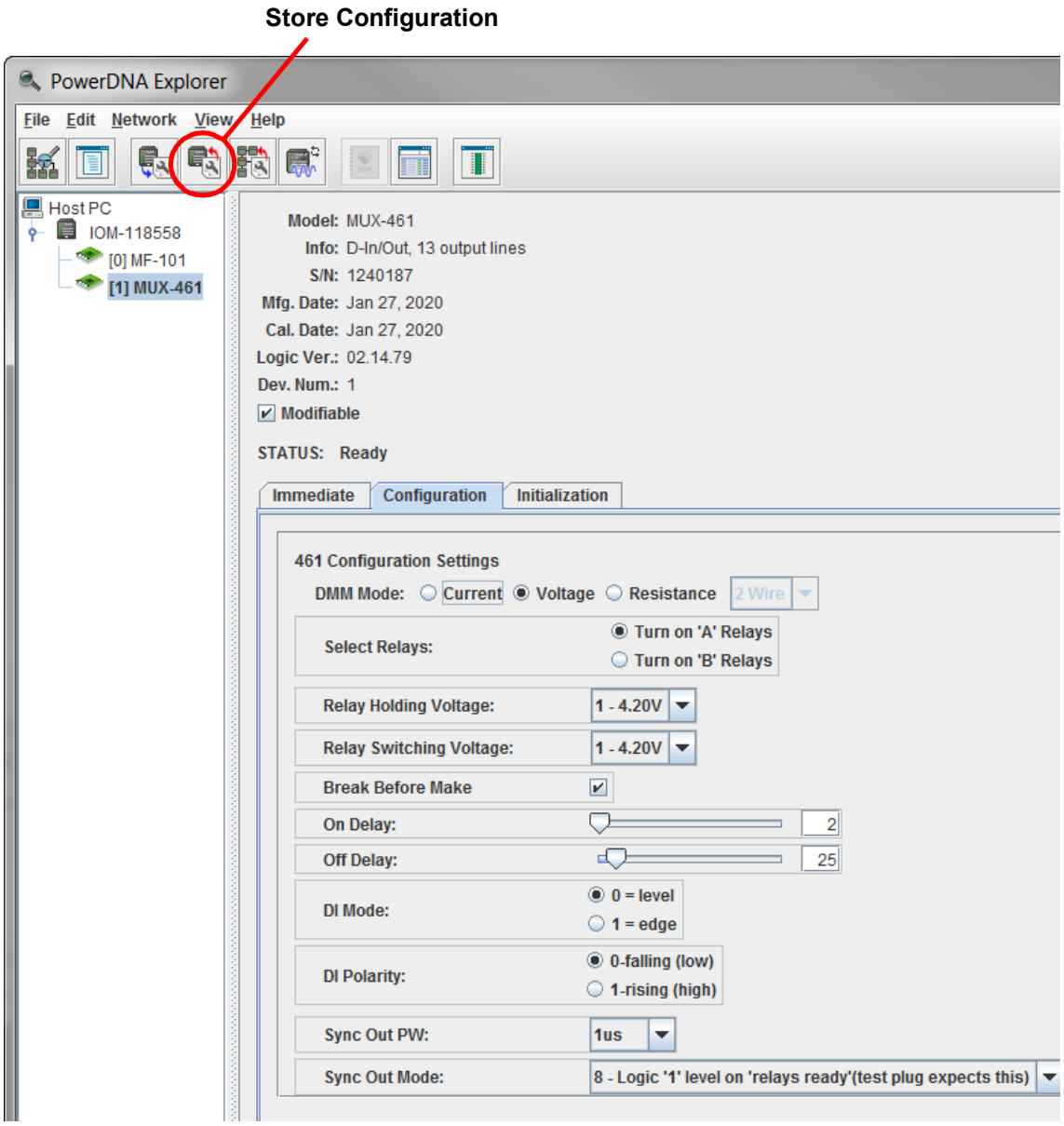


Figure 3-3 PowerDNA Explorer for DNx-MUX-461: Configuration Tab

3.4 Configuring Initialization State

The **Initialization** tab is a combination of the **Immediate** and **Configuration** tabs shown in **Figure 3-2** and in **Figure 3-3** respectively. However, instead of writing the relay states immediately to RAM, the **Initialization** tab stores the configuration into the EEPROM. The **Initialization** settings take effect upon power-up.

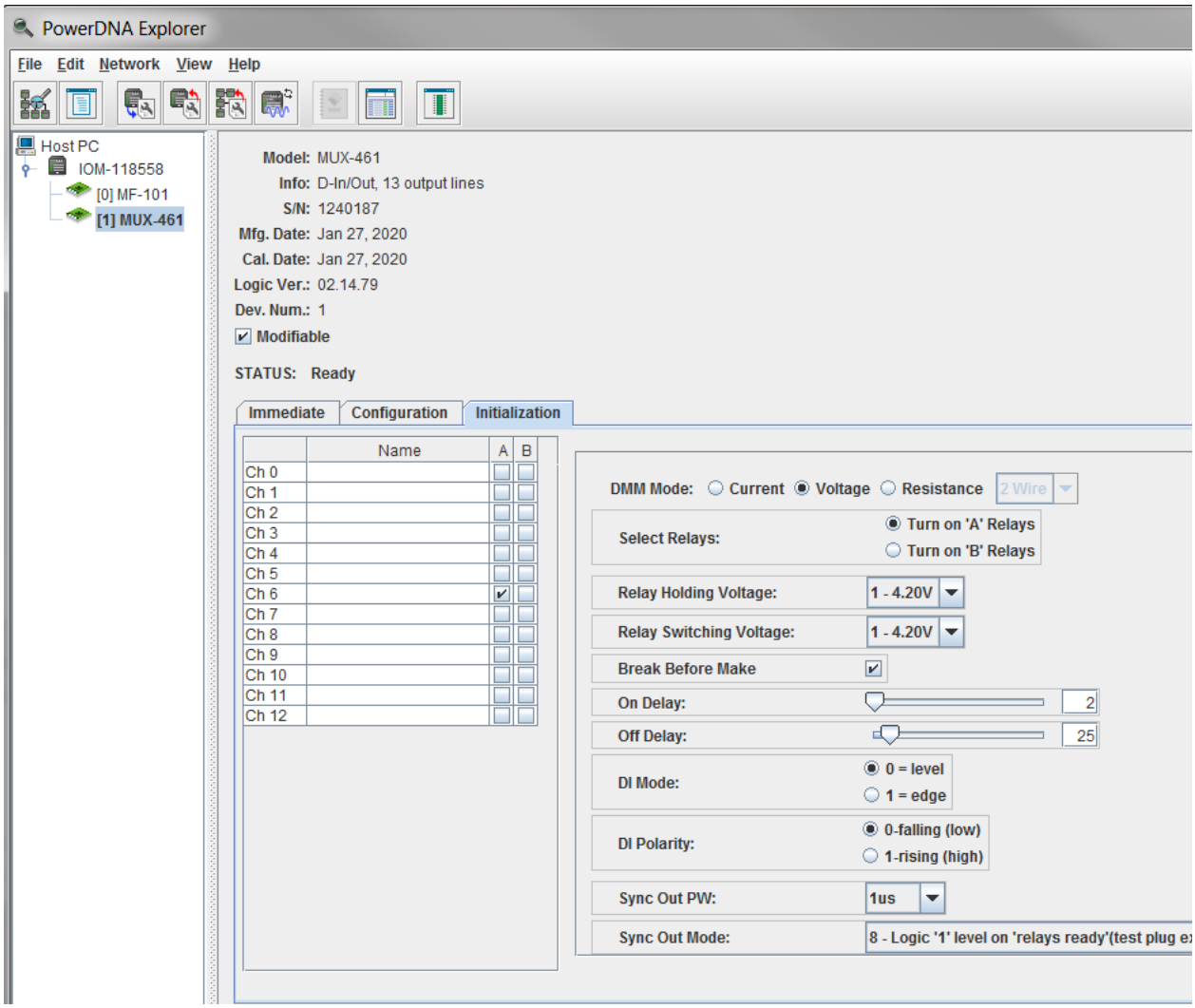


Figure 3-4 PowerDNA Explorer for DNx-MUX-461: Initialization Tab

Chapter 4 Programming with High-level API

This chapter provides the following information about programming the DNx-MUX-461 using the UeiDaq Framework API:

- About the High-level API (Section 4.1)
- Sample Code (Section 4.2)
- Create a Session (Section 4.3)
- Configure the Mux Port (Section 4.4)
- Configure the Timing (Section 4.5)
- Start the Session (Section 4.6)
- Write Data (Section 4.7)
- Read Diagnostic Data (Section 4.8)
- Stop the Session (Section 4.9)

4.1 About the High-level API

UeiDaq Framework is object oriented and its objects can be manipulated in the same manner from different development environments, such as Visual C++, MATLAB, LabVIEW, and more. The Framework is supported in Windows 7 and up. It is generally simpler to use compared to the low-level API, and it includes a generic simulation device to assist in software development. Therefore, we recommend that Windows users use the high-level API unless unconventional functionality is required. Users programming in Linux or a real-time operating system should instead use the low-level API (Chapter 5).

For more detail regarding the Framework's architecture, please see the "UeiDaq Framework User Manual" located under:

Start » All Programs » UEI » Framework » Documentation

4.2 Sample Code

UeiDaq Framework is bundled with examples for supported programming languages. The sample code is located under:

Start » All Programs » UEI » Framework » Examples

Unlike the low-level samples, Framework samples are board-agnostic. For instance, the "Mux" sample applies to all UEI multiplexer boards including the DNx-MUX-461.

Each high-level sample follows the same basic structure:

1. Create a session.
2. Configure the session for a particular device and subsystem.
3. Configure the timing.
4. Start the session.
5. Read or write data.
6. Stop the session.

This chapter presents examples using C++ API, but the concepts are the same no matter which programming language you use. The “UeiDaq Framework User Manual” provides additional information about programming in other languages.

4.3 Create a Session

The session object manages all communications with the DNx-MUX-461. Therefore, the first step is always to create a new session.

```
//create a session object

CUeiSession mySession;
```

Because each session is dedicated to a specific device and subsystem, the application should use separate sessions for the DNx-MUX-461 and DNx-DMM-261.

4.4 Configure the Mux Port

The following example code configures the session to access all channels on the DNx-MUX-461 and obtains a pointer to the mux port object.

```
//Configure session to access all relay channels.

CUeiMuxPort* port = mySession.CreateMuxPort(“pdna://192.168.100.2/Dev1/
Mux0”, true);
```

The input parameters are:

- `resource` – specifies the board and subsystem (see Section 4.4.1).
- `breakBeforeMake` – enables or disables break-before-make functionality on “D” relays (see Section 4.4.2).

4.4.1 Resource String

Framework uses a resource string to link the session to the hardware. The resource string syntax is similar to a web URL; it should not have any spaces and is case insensitive.

“<device class>://<IP address>/<device number>/<subsystem><channel list>”

- *<device class>* – By default, Framework samples open with a generic simulated device. To use the DNx-MUX-461, set the device class to `pdna`.
- *<IP address>* – IP address of the IOM.
- *<device number>* – position of the DNx-MUX-461 within the chassis, relative to the other I/O boards.
- <subsystem> – DNx-MUX-461 only supports the `Mux` subsystem.
- *<channel list>* – desired port(s) within the selected subsystem. All DNx-MUX-461 channels are included in port 0.

4.4.2 Break-before-Make and Port On Delays

By default, the DNx-MUX-461 implements a break-before-make during relay write commands. All relays are opened before the programmed relays are closed. You can set the breaking time of the break-before-make ("off delay"), the delay before a new relay write command is accepted ("on delay"), and optionally disable break-before-make for "D" relays only. The DNx-MUX-461 supports delay times between 100uS and 25.6ms in 100uS increments.

```
//Keep relays open for 400uS during a relay write.
//New relay writes are accepted after 200uS.

port->SetOffDelay(400);
port->SetOnDelay(200);
```

You can enable or disable break-before-make when you first create the mux port, or you can use the EnableBreakBeforeMake() method shown below.

```
//Disable break-before-make on "D" relays.

port->EnableBreakBeforeMake(false);
```

NOTE: Break-before-make cannot be disabled for "A" and "B" relays.

4.4.3 Sync Modes

You can optionally gate relay switches with a signal on the SYNC IN pin. You can also generate a pulse on the SYNC OUT pin whenever the relays switch.

As an example, you could synchronize one DNx-MUX-461 board to another DNx-MUX-461 board. To achieve this functionality, wire the SYNC OUT pin on the first board to the SYNC IN pin on the second board (with SYNC GND connected). Configure and enable SYNC OUT on the first board as described in Section 4.4.3.1. Then, configure and enable SYNC IN on the second board as described in Section 4.4.3.2. Relay channels on both boards are programmed on/off as usual. However, the second board delays the actual hardware switching until it detects the SYNC IN trigger.

4.4.3.1 Sync Out

The following example configures the SYNC OUT pin to pulse high after the relays have finished switching.

```
//Set SYNC OUT mode to pulse high after a relay write.

port->SetSyncOutputMode(UeiMuxSyncOutputRelaysReadyPulse1);
```

The following SYNC OUT modes are supported:

- `UeiMuxSyncOutputLogic0` – drive with constant logic '0'
- `UeiMuxSyncOutputLogic1` – drive with constant logic '1'
- `UeiMuxSyncOutputLine0` through `UeiMuxSyncOutputLine3` – drive with internal SYNC BUS[0] through [3]
- `UeiMuxSyncOutputRelaysReadyPulse1` – output positive transitioning pulse when relays have settled (normally low)
- `UeiMuxSyncOutputRelaysReadyPulse0` – output negative transitioning pulse when relays have settled (normally high)

- `UeiMuxSyncOutputRelaysReadyLogic1` – drive logic '1' level while relays are settled (used with UEI's DNx-TADP-461 test adapter)
- `UeiMuxSyncOutputRelaysReadyLogic0` – drive logic '0' level while relays are settled

The pulse width for `UeiMuxSyncOutputRelaysReadyPulse0` or `UeiMuxSyncOutputRelaysReadyPulse1` mode defaults to 100us. You can use `SetSyncOutputPulseWidth()` to set the pulse width to 1us, 100us, or 1ms:

```
//Set SYNC OUT pulse width to 1ms.

port->SetSyncOutputPulseWidth(1000);
```

4.4.3.2 Sync In

To use SYNC IN, the functionality must first be enabled:

```
//Enable SYNC IN.

port->EnableSyncInput(true);
```

You can either trigger relay writes on edges or levels. To switch relays on an edge, enable edge detection mode and select the desired edge polarity (`UeiDigitalEdgeRising` or `UeiDigitalEdgeFalling`).:

```
//Trigger relay writes on a rising edge.

port->EnableSyncInputEdgeMode(true);
port->SetSyncInputEdgePolarity(UeiDigitalEdgeRising);
```

To switch relays on a logic level, you can disable edge mode (less commonly used). `UeiDigitalEdgeRising` corresponds to logic level '1', and `UeiDigitalEdgeFalling` corresponds to logic level '0'.

```
//Trigger relay writes when SYNC IN is at logic level 1.

port->EnableSyncInputEdgeMode(false);
port->SetSyncInputEdgePolarity(UeiDigitalEdgeRising);
```

You can release the mux from waiting for a SYNC IN trigger by restarting the session with a `Stop` followed by a `Start` (see Section 4.9 and Section). This allows the latest relay writes to go through. All subsequent writes are handled normally.

4.5 Configure the Timing

Only Point-by-Point data acquisition mode can be used to transfer data between a Framework application and the DNx-MUX-461.

Point-by-Point mode transfers one sample at a time to/from each configured channel of the I/O board. The delay between samples is controlled by the host application (e.g. by using a Sleep function), thus limiting the data transfer rate to a maximum of 100 Hz. This mode is also known as immediate mode or simple mode.

```
//configure session to use Point-by-Point DAQ mode
mySession.ConfigureTimingForSimpleIO();
```

4.6 Start the Session

After the session is configured, you can start the session manually:

```
//Start the session.
mySession.Start();
```

If you don't explicitly start the session, it will start automatically the first time you try to transfer data.

4.7 Write Data

Switching relays on/off is done using a `MuxWriter` object:

```
//Create a writer object and link it to the session's data stream.
CUeiMuxWriter writer(mySession.GetDataStream());
```

After creating the writer object, call the `WriteMuxDmm()` method to connect a multiplexer channel to the DMM. A break-before-make (Section 4.4.2) ensures that a maximum of one channel will be connected at any given time.

```
//Select channel number 12 on bus B.
int channel = 12;
int relaySelect = 2;
//Close the B12 relay (IN26+ and IN26-).
//Output to DMM V+ and DMM V- by closing D1 and D2 relays.
writer.WriteMuxDmm(channel, relaySelect, UeiMuxDmmMeasV);
```

The `relaySelect` parameter is defined as follows:

- 0 – open relay on bus A and B
- 1 – close relay on bus A
- 2 – close relay on bus B

The supported DMM Modes are:

- `UeiMuxDmmDisable` – disconnect all channels from DMM
- `UeiMuxDmmMeasV` – voltage measurement
- `UeiMuxDmmMeasI` – current measurement
- `UeiMuxDmmMeasRes2` – 2-wire resistance measurement

- `UeiMuxDmmMeasRes4` – 4-wire resistance measurement (closes both A and B relays; `relaySelect` is ignored)

Refer to **Figure 2-1** for a diagram of the input channels, DMM output pins, and relay architecture.

4.8 Read Diagnostic Data

The DNx-MUX-461 monitors DC internal supply voltages and temperature using an onboard ADC. It can also readback the current state, status, and number of cycles for each relay.

4.8.1 Voltage and Temperature

You can read from the diagnostic ADC channels described in **Table 4-1**.

Table 4-1 Diagnostic Channels

Channel #	Description
0	Voltage of internal 24V supply
1	Voltage of internal 3.3V supply
2	Internal relay driver voltage
3	Temperature in degrees C
4	Status (uint32, see **Table 4-2**)
5	Timestamp

Create a `MuxWriter` object (Section 4.7) and use its `ReadADC()` method:

```
//Read all 6 diagnostic channels.

double adcBuffer[6];
writer.ReadADC(6, adcBuffer, NULL);
```

4.8.2 Relay States and Status

You can monitor current relay states and board status with the `MuxWriter` object's `ReadStatus()` method:

```
//Read current state of A, B, D relays and return their status.

uInt32 stRelayA, stRelayB, stRelayD, status;
writer.ReadStatus(&stRelayA, &stRelayB, &stRelayD, &status);
```

The relay state is returned as a bitwise representation where 1 = closed and 0 = open. For example, `stRelayA` = 0x10 indicates that relay A4 is closed.

The `status` word is described in **Table 4-2**. All other bits are reserved.

Table 4-2 Relay Status

Bit #	Description
17	'1' means new unread data is ready from the ADC

Table 4-2 Relay Status

Bit #	Description
16	‘1’ means overrun (a relay write occurred while busy)
3	‘1’ means state machine is busy
2	‘1’ means output state machine is waiting for the external SYNC IN (if configured) or for “relays ready”
1	‘1’ means relays are ready
0	reports the logic state of the SYNC IN pin

4.8.3 Relay Counts

The `ReadRelayCounts()` method in the `MuxWriter` class returns the number of times each relay has been energized. The counts are updated internally every 11 minutes. You can return the count from up to 30 relays (A0:A12, B0:B12, and D1:D4).

```
//Get the number of relay cycles for all relays.

int relaycounts[30];
writer.ReadRelayCounts(30, relaycounts, NULL);

//Print results for A and B relays.

for (int j = 0; j < 13; j++)
{
   std::cout << "A" << j << "=" << relaycounts[j]
          << "   B" << j << "=" << relaycounts[j+13] << std::endl;
}

//Print results for D relays.

for (int j = 0; j < 4; j++)
{
   std::cout << "D" << j+1 << "=" << relaycounts[j+26] << std::endl;
}
```

4.9 Stop the Session

The session will automatically stop and clean itself up when the session object goes out of scope or when it is destroyed. To manually stop the session:

```
//Stop the session.

mySession.Stop();
```

To reuse the object with a different set of channels or parameters, you can manually clean up the session as follows:

```
//clean up session and free resources

mySession.CleanUp();
```

Chapter 5 Programming with Low-level API

This chapter provides the following information about programming the DNx-MUX-461 using low-level API:

- About the Low-level API (Section 5.1)
- Sample Code (Section 5.2)
- Data Acquisition Modes (Section 5.3)
- Point-by-Point API (Section 5.4)

5.1 About the Low-level API

The low-level API provides direct access to the DAQBIOS protocol structure and registers in C. The low-level API is intended for speed-optimization, when programming unconventional functionality, or when programming under Linux or real-time operating systems.

When programming in Windows OS, we recommend that you use the UeiDaq high-level Framework API (see Chapter 4). The Framework simplifies the low-level API, making programming easier and faster while still providing access to the majority of low-level API features. Additionally the Framework supports a variety of programming languages and the use of scientific software packages such as LabVIEW and MATLAB.

For additional information regarding low-level programming, refer to the "PowerDNA API Reference Manual" located in the following directories:

- On Linux: *<PowerDNA-x.y.z>/docs*
- On Windows: **Start » All Programs » UEI » PowerDNA » Documentation**

NOTE: The DNx-MUX-461 is supported in PowerDNA version 5.0+. If you're unsure if your version supports the board, please contact Technical Support at support@ueidaq.com

5.2 Sample Code

Application developers are encouraged to explore the self-documented source code examples to get started programming UEI products. The sample code is located in the following directories:

- On Linux: *<PowerDNA-x.y.z>/src/DAQLib_Samples*
- On Windows: **Start » All Programs » UEI » PowerDNA » Examples**

The I/O board number is embedded in the name of the sample code. For example, the Sample461 folder contains sample code specific to the DNx-MUX-461. The sample code should run out of the box after inputting the IOM's IP address and the board's Device Number (DEVN).

5.3 Data Acquisition Modes

The following data acquisition (DAQ) mode is available for transferring data between the DNx-MUX-461 and the low-level user application:

- **Point-by-Point:** Transfers one data point at a time to/from each configured channel of a single I/O board. Timing is controlled by the user application, which limits the transfer rate to 100 Hz. This mode is also known as immediate mode or simple mode.

Please refer to “FAQ - Data Acquisition Modes” for an overview and comparison of all the different acquisition modes offered by UEI. The “PowerDNx Protocol Manual” includes more detailed information about the protocols. Both of these documents are located in the directories listed in Section 5.1.

5.4 Point-by-Point API

Table 5-1 summarizes the low-level API used to configure, read from, and write to the DNx-MUX-461 in Point-by-Point DAQ mode.

Table 5-1 Low-level DNx-MUX-461 API

Function	Description
`DqAdv461SetChannel`	Select a multiplexer channel and connect output to DMM; optionally enable sync in/out for the write
`DqAdv461SetCfg`	Configure break-before-make and sync in/out functionality
`DqAdv461ReadADC`	Read diagnostic internal voltages and board temperature
`DqAdv461ReadStatus`	Read positions of all relays, the most recent write, and the current board status
`DqAdv461GetRelayCounts`	Read the number of times each relay has been energized

The functions and parameters are described in detail in the “PowerDNA API Reference Manual”. Please see *Sample461* for a comprehensive example which includes typical initialization, error handling, and usage of these functions. The remainder of this chapter is intended as a supplement to the sample code and the API reference manual.

5.4.1 Programming Relays

Use the `DqAdv461SetChannel()` API to connect ONE channel to the DMM output pins.

```
int DqAdv461SetChannel(int hd, int devn, uint32 channel_num, uint32
relay_select, uint32 dmm_mode, uint32 sync);
```

- `int hd` - handle to the IOM
- `int devn` – device number in the layer stack
- `uint32 channel_num` – channel number from 0...12
- `uint32 relay_select` – select relay on bus A and/or bus B
- `uint32 dmm_mode` – measure voltage, current, 2-wire resistance, or 4-wire resistance
- `uint32 sync` – flags to enable SYNC IN and SYNC OUT pins

A break-before-make automatically opens all relays before closing the programmed relays. Therefore, only one channel can be connected at any given time. Refer to **Figure 2-1** for a diagram of the input channels, DMM outputs, and relay buses.

Example: 2-wire measurement

```
//Close the B12 relay (IN26+ and IN26-).
//Output to DMM V+ and DMM V- by closing D1 and D2 relays.
//Disable sync in and sync out.

DqAdv461SetChannel(hd, devn, 12, DQ_MUX461_SETCHAN_RL_B,
   DQ_MUX461_SETCHAN_DMM_V, 0);
```

Example: 4-wire measurement

```
//Close the A12 and B12 relays (IN12A+, IN12A-, IN12B+, IN12B-).
//Connect A12 to DMM V+ and DMM V- by closing D1 relay.
//Connect B12 to DMM RS+ and DMM RS- by closing D4 relay.
//Disable sync in and sync out.

DqAdv461SetChannel(hd, devn, 12, DQ_MUX461_SETCHAN_RL_A_B,
   DQ_MUX461_SETCHAN_DMM_RES4, 0);
```

5.4.2 Configuration Settings

The DNx-MUX-461 is configured using the `DqAdv461Cfg()` function:

```
int DqAdv461SetCfg(int hd, int devn, pDQ461CFG pCfg);
```

The function takes in the following inputs and populates a configuration card.

- `int hd` - handle to the IOM
- `int devn` – device number in the layer stack
- `pDQ461CFG pCfg` – structure containing configuration settings

Before calling `DqAdv461SetCfg()`, initiate and fill out the fields in a `DQ461CFG` structure (shown below). Any uninitialized fields are set to 0. Configuration calls can be additive, i.e. each following call adds or changes a parameter on the configuration card. Refer to the “PowerDNA API Reference Manual” for a complete description of each parameter.

```
typedef struct {

//Relay switching delays
   uint32 d_bbm_mode;    // enable break-before-make for D relays
   uint32 on_delay;      // time before next command is accepted
   uint32 off_delay;     // breaking time of break-before-make

//Sync modes
   uint32 sync_in_mode;       // SYNC IN pin operation mode
   uint32 sync_in_polarity;   // polarity of SYNC IN strobe
   uint32 sync_out_pw;   // SYNC OUT pulse length
   uint32 sync_out_mode; // SYNC OUT pin operation mode
   uint32 sync_skip;     // release previous command from sync in waiting

} DQ461CFG, *pDQ461CFG;
```

5.4.3 Sync In/Out Handshaking

The following example programs a DNx-MUX-461 located at DEVN=0 to switch at the same time as a DNx-MUX-461 located at DEVN=1. This example assumes that the SYNC OUT pin on DEVN1 is wired to the SYNC IN pin on DEVN0.

5.4.3.1 Configure SYNC OUT

First, configure DEVN0 to generate a pulse on the SYNC OUT pin whenever a relay state is written.

```
//Initialize a configuration data structure.

DQ461CFG r_cfg;

//SYNC OUT is normally low and pulses high on a relay write.
//Set pulse width to 1ms.

r_cfg.sync_out_mode = DQ_MUX461_SETCFG_SYNC_OUT_MODE_HIGH_PULSE;
r_cfg.sync_out_pw = DQ_MUX461_SETCFG_SYNC_OUT_PW_1MS;

//Set the configuration on DEVN0.

DqAdv461SetCfg(hd, devn0, &r_cfg);
```

5.4.3.2 Configure SYNC IN

Next, configure SYNC IN pin as a gating device on DEVN1. Relay switching will be delayed until the SYNC IN signal is detected.

```
//Wait until SYNC IN is at a logic '1' level before switching relays.

r_cfg.sync_in_mode = DQ_MUX461_SETCFG_SYNC_IN_MODE_LEVEL;
r_cfg.sync_in_polarity = DQ_MUX461_SETCFG_SYNC_IN_POLARITY_HIGH;

//Set the configuration on DEVN1.

DqAdv461SetCfg(hd, devn1, &r_cfg);
```

5.4.3.3 Writing Relay States

Use the `sync` parameter to enable SYNC IN and/or SYNC OUT for the write command.

```
//Close B12 relay and generate positive-transitioning SYNC OUT pulse.

DqAdv461SetChannel(hd, devn0, 12, DQ_MUX461_SETCHAN_RL_B,
   DQ_MUX461_SETCHAN_DMM_V, DQ_MUX461_SETCHAN_SYNC_OUT);

//Close A10 relay after SYNC IN pin is detected as high.

DqAdv461SetChannel(hd, devn1, 10, DQ_MUX461_SETCHAN_RL_A
   DQ_MUX461_SETCHAN_DMM_V, DQ_MUX461_SETCHAN_SYNC_IN);
```

Appendix A

Accessories

A.1 Cables and STP Boards

The following cables and screw terminal panels are available for the DNx-MUX-461 board.

DNA-CBL-62

This is a 62-conductor round shielded cable with 62-pin male D-sub connectors on both ends. It is made with round, heavy-shielded cable; 2.5 ft (75 cm) long, weight of 9.49 ounces or 269 grams; up to 10ft (305cm) and 20ft (610cm).

DNA-STP-62

The STP-62 is a Screw Terminal Panel with three 20-position terminal blocks (JT1, JT2, and JT3) plus one 3-position terminal block (J2). The dimensions of the STP-62 board are 4w x 3.8d x1.2h inch or 10.2 x 9.7 x 3 cm (with standoffs). The weight of the STP-62 board is 3.89 ounces or 110 grams.

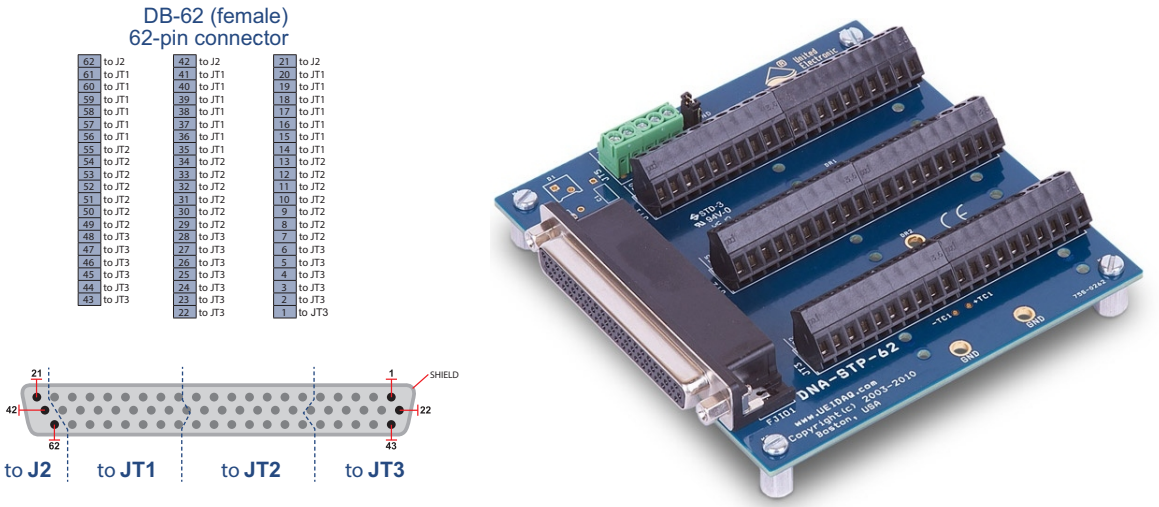


Figure A-1 Pinout and Photo of DNA-STP-62 Screw Terminal Panel