DNx-MUX-461 Series
User Manual

Multiplexer Boards
for the PowerDNA Cube and RACK Series Chassis

October 2023

PN Man-DNx-MUX-461 Series

© Copyright 1998-2023 United Electronic Industries, Inc. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form by any means, electronic, mechanical, by photocopying, recording, or otherwise without prior written permission.

Information furnished in this manual is believed to be accurate and reliable. However, no responsibility is assumed for its use, or for any infringement of patents or other rights of third parties that may result from its use.

All product names listed are trademarks or trade names of their respective companies.

See the UEI website for complete terms and conditions of sale:

http://www.ueidaq.com/cms/terms-and-conditions

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.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

© Copyright 2023
United Electronic Industries, Inc.

© Copyright 2023
United Electronic Industries, Inc.

List of Figures

© Copyright 2023
United Electronic Industries, Inc.

List of Tables

© Copyright 2023
United Electronic Industries, Inc.

Chapter 1 Introduction

This manual outlines the feature set and use of UEI's DNx-MUX-461 Series Multiplexer Boards. These multiplexer boards are designed for a variety of switching and digital control applications.

The DNx-MUX-461 Series includes the following multiplexer offerings:

- the DNx-MUX-461 26-channel reed relay multiplexer
- the DNx-MUX-461-350 24-channel high voltage, solid-state relay multiplexer designed for use in a wide variety of switching applications.

The information presented in this manual will apply to both DNx-MUX-461 Series boards except where specifically noted. Keep in mind the difference in the number of available channels between the two boards.

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 Series User Manual*" is organized as follows:

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

© Copyright 2023
United Electronic Industries, Inc.

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 DNx-MUX-461 Series includes the standard DNx-MUX-461 boards and the high voltage DNx-MUX-461-350 boards.

Both the standard and high-voltage versions of the DNx-MUX-461 Series boards are compatible with UEI chassis as follows:

- DNA-MUX-461 and DNA-MUX-461-350 boards are compatible with UEI Cube chassis.
- DNR-MUX-461 and DNR-MUX-461-350 boards are compatible with UEI RACKtangle chassis.
- DNF-MUX-461 and DNF-MUX-461-350 boards are compatible with UEI FLATRACK chassis.

The DNx-MUX-461 boards are electronically identical to each other and the DNx-MUX-461-350 boards are electronically identical to each other. Within each group, the boards differ only in mounting hardware.

Throughout this manual, the terms DNx-MUX-461, DNx-MUX-461-350, and DNx-MUX-461 Series refer 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**

© Copyright 2023
United Electronic Industries, Inc.

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 *"Firmware Update Procedures.pdf"* for instructions on checking and updating the firmware. This document is located in the following locations:

- On Linux: *PowerDNA_Linux_<x.y.z>/docs*
- On Windows:
  Start » All Programs » UEI » DNx Firmware Update Procedures

1.6 DNx-MUX-461 Features

DNx-MUX-461 Series Multiplexer Boards provide 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 for standard DNx-MUX-461 boards (24 two-wire or 12 four-wire channels for DNx-MUX-461-350 boards)
- Up to five DNx-MUX-461 Series boards may be daisy chained together within a Cube or Rack chassis thus providing up to 130 two-wire or 65 four-wire channels for standard DNx-MUX-461 boards. For DNx-MUX-461-350 boards, the maximum number of channels is 120 two-wire or 60 four-wire.
- connects to DNx-DMM-261 without external wiring
- maximum operating voltage:
  - ±170 VDC or VAC for standard DNx-MUX-461 boards
  - ±350 VDC or VAC for high voltage DNx-MUX-461-350 boards
- contact ON impedance
  - 0.5 Ω (not including cabling) for DNx-MUX-461 boards
  - 4.0 Ω (not including cabling) for DNx-MUX-461-350 boards

© Copyright 2023
United Electronic Industries, Inc.

- 500 mA continuous load current rating
- Relay operation counter tracks relay life cycles
- 500 Hz 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 (24 two-wire or 12 four-wire channels using a DNx-MUX-461-350). A channel can be programmed for Voltage, Current, 2-Wire Resistance, or 4-Wire Resistance measurements. Two and four-wire resistance measurements can be mixed.

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. With a Cube chassis, a set of internal connectors connect the DNx-MM-261 and DNx-MUX-461 Series boards. In a RACKTangle chassis, a DNR-CBL-461-x cable is used (see Section A.1).

1.6.2 Switch Conditions

DNx-MUX-461 boards employ 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).

DNx-MUX-461-350 boards feature solid-state relays that support switching rates of up to 500 Hz. Each channel is capable of switching voltages up to ±350 VDC or AC waveforms with peaks less than ±350 VDC. Each channel is rated for continuous operation at 500 mA DC or AC rms with a switch resistance of less than 4.0 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

The standard DNx-MUX-461boards utilize a built-in counter that counts the number of switch cycles for each reed relay, allowing the age of contacts to be tracked. Each relay is rated for 1 million cycles at 24 VDC / 50 mA or 12 VDC / 100 mA.

The solid-state relays on the DNx-MUX-461-350 boards are not limited by a maximum number of operations so tracking of relay life is not required.

© Copyright 2023
United Electronic Industries, Inc.

1.6.4 Synchro-nization

The DNx-MUX-461 can synchronize relay switching via the SYNC IN and SYNC OUT pins.

When SYNC IN is enabled, a pulse on the SYNC IN pin can be used as a gate for switching relays when the SYNC IN mode and polarity conditions are met.

The SYNC OUT pin can be configured to change state when there is a relay state change (write or ready).

1.6.5 Diagnostics

Users have the capability of reading internal supply voltages, the relay driver voltage, and on-board 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 Series boards offer 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 designed to operate to 5 g vibration and 100 g shock. The board has been tested from -40 to +85 °C 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 DNx-MUX-461 Series boards to UEI's DNA-STP-62 screw terminal panel via the DNA-CBL-62 cables.

In a Rack chassis, a DNR-CBL-461-x cable connects a DNR-DMM-261 board to as many as five DNR-MUX-461 Series boards. The cable plugs into the JPOW1 connectors on the PCBs in a daisy-chain configuration as described in the "*DNx-DMM-261 User Manual*". Five different options of the DNR-CBL-461-x cable are available, where x corresponds to the number of MUX boards in the system.

1.6.9 Software Support

DNx-MUX-461 Series boards include a software suite that supports 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.

© Copyright 2023
United Electronic Industries, Inc.

1.7 Technical Specifications

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

Table 1-1 Technical Specifications

Specification	DNx-MUX-461	DNx-MUX-461-350
Configuration	26 two-wire or 13 four-wire multiplexers. Two-wire and four-wire measurements may be mixed.	24 two-wire or 12 four-wire multiplexers. Two-wire and four-wire measurements may be mixed.
Switch Specifications		
Rated load (switching)	500 mA (-40 to +85 °C)	500 mA (-40 to +85 °C)
Rated load (carry)	1 A	
Max operating voltage	170 VDC or VAC (other version available as special order)	350 VDC (peak) or VAC
Contact type	Reed relay	Solid-state / MOSFET relays
Contact ON impedance	0.5 Ω max (at the I/O connector)	4.0 Ω max (at the I/O connector)
Contact OFF impedance	>10 GΩ	>10 MΩ
OFF leakage current	< 20 nA	< 10 µA max, 0.05 µA tyupical
Carrying current	1 A	
Max update rate	500 Hz	500 Hz
Turn-off time	< 0.35 ms typical (A/B channel relays) < 1 ms typical (D2 V± relays)	< 1 ms typical
Turn-on time	< 0.25 ms typical (A/B channel relays) < 2 ms typical (D2 V± relays)	< 1 ms typical
Max operating rate	500 Hz	500 Hz
Rated contact life	1 million operations at 24 VDC / 50 mA 1 million operations at 12 VDC / 100 mA	
Power up / reboot state	All switches off	All switches off
Power dissipation	< 5 W not including output switching	< 5 W not including output switching
Isolation	350 Vrms	350 Vrms
Operating temperature range	Tested -40 to +85 °C	Tested -40 to +85 °C
Operating humidity	95%, non-condensing	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	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	100 g, 3 ms half sine, 18 shocks @ 6 orientations 30 g, 11 ms half sine, 18 shocks @ 6 orientations
MTBF	450,000 hours	800,000 hours

© Copyright 2023
United Electronic Industries, Inc.

Chapter 2 Device Overview

This section describes the device architecture and hardware of the DNx-MUX-461 Series Multiplexer Boards. 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)
- Parallel Multiplexers (Section 2.5)
- Indicators and Connectors (Section 2.6)
- Pinout (Section 2.7)

2.1 Relay Architecture

Figure 2-1 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 (24 for the DNx-MUX-461-350) evenly divided among the A bus and B bus. An A and B channel pair may be used together as a four-wire channel, thereby providing up to 13 four-wire channels (12 for the DNx-MUX-461-350).

When programming the relays, users select a channel number between 0...12, the A or B bus (for two-wire measurements only), and a DMM Mode. **Table 2-1** summarizes which relays are closed for the programmed DMM Mode and channel. Note that four-wire resistance measurements use the B bus as the sense line and the A bus as the excitation line.

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

© Copyright 2023
United Electronic Industries, Inc.

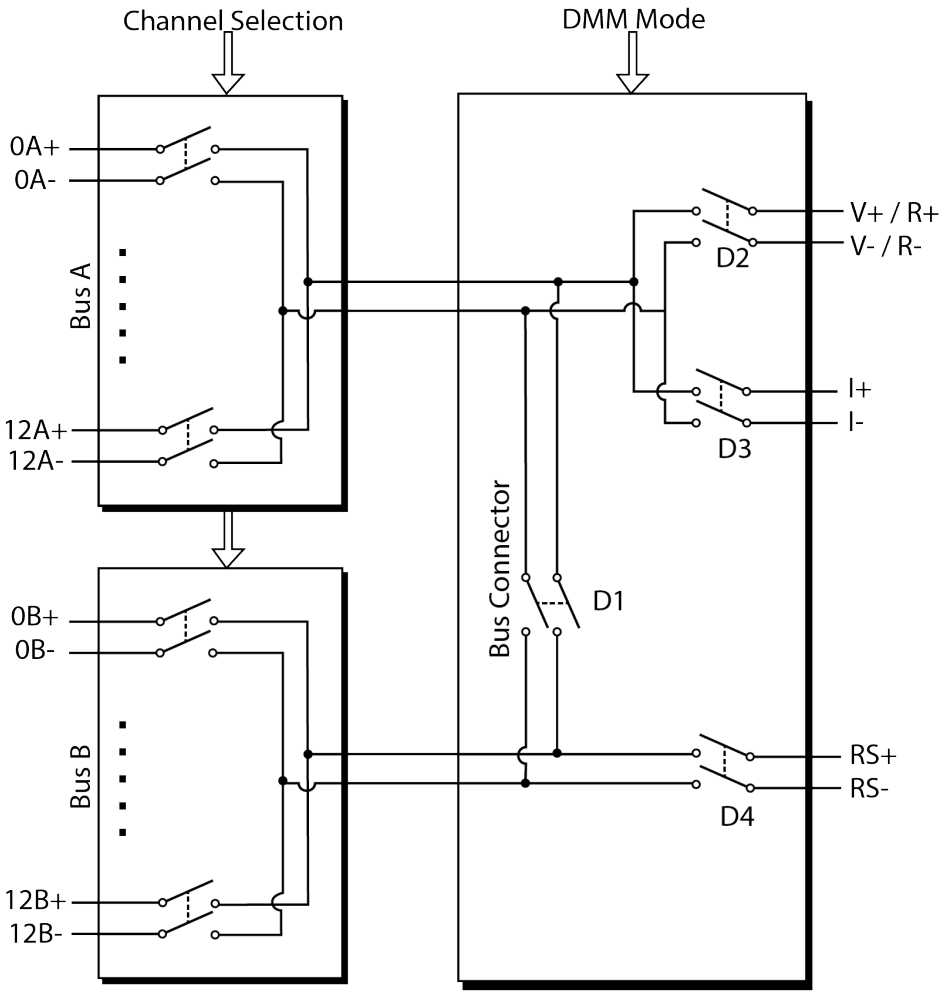

Figure 2-1 Simplified Schematic of DNx-MUX-461

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	●	●		●		●

© Copyright 2023
United Electronic Industries, Inc.

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 is referred to as "break-before-make" and 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 the reed relays on the standard DNx-MUX-461.

2.3 Synchro-nizaton

Synchronization functionality is provided via the SYNC IN and SYNC OUT pins on the DB-62 connector or via internal chassis bus lines.

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 level/rising edge or low level/falling edge), is user configurable.

When enabled, the SYNC OUT pin can be configured to change state when there is a relay state change (write or ready). The following functionality is available:

- Drive constant logic '0' or '1' on SYNC OUT
- Generate high or low pulse on relay write. The pulse width is also user configurable via user application software.
  NOTE: To ensure that the pulses are evenly spaced, it is recommended that the "on delay" time be set to at least 1.5 ms.
- Change state (high or low) when relays are ready.

For more information on configuring SYNC IN and SYNC OUT, refer to the "*PowerDNA API Reference Manual*" or the "*UeiDaq Framework Reference Manual*".

2.4 Diagnostics

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

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

© Copyright 2023
United Electronic Industries, Inc.

2.5 Parallel Multiplexers

Multiple DNx-MUX-461 Series boards can be arranged in parallel to increase the total number of channels. Up to five DNx-MUX-461 Series boards may be placed in parallel within the Cube/RACK chassis providing up to 130 two-wire or 65 four-wire channels in a single chassis. Any configuration that contains one or more DNx-MUX-461-350 boards will have that maximum number of channels reduced since a single DNx-MUX-461-350 board is limited to 24 two-wire or 12 four-wire channels. Larger systems are possible although that will require the DNx-DMM-261 to DNx-MUX-461 Series interconnection be external to the chassis.

In a Cube chassis, internal connectors connect the DNx-MUX-461 Series boards to each other and to the DNx-DMM-261. Within a Rack chassis, connections are made using a DNR-CBL-461-x cable.

2.6 Indicators and Connectors

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

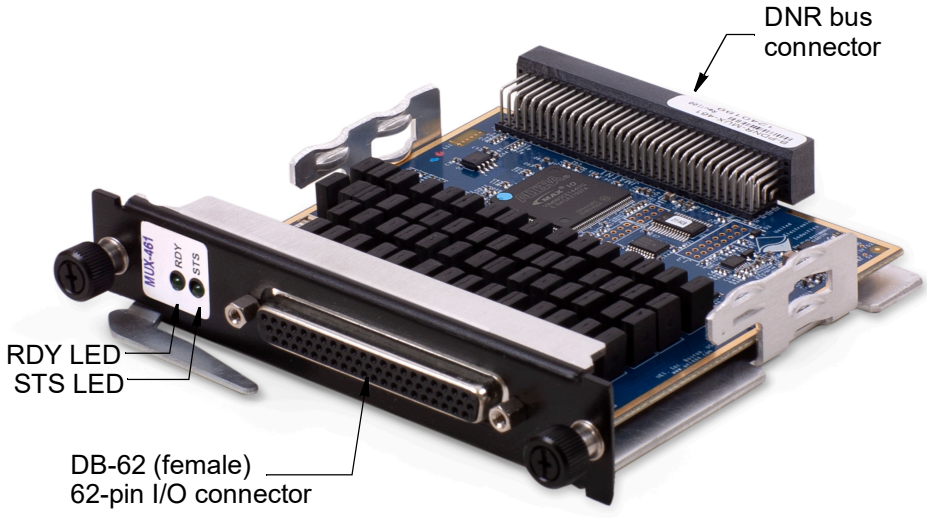

Figure 2-2 Photo of DNR-MUX-461 Board

Table 2-2 DNx-MUX-461 LED Indicators

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

2.7 Pinout

External connections for DNx-MUX-461 Series boards are made through a standard DB-62 female connector. **Figure 2-3** illustrates the pin numbering for the DB-62 connector. If using DNx-MUX-461 Series boards with the DNx-DMM-261, the chassis is shipped with DMM connections made internally.

© Copyright 2023
United Electronic Industries, Inc.

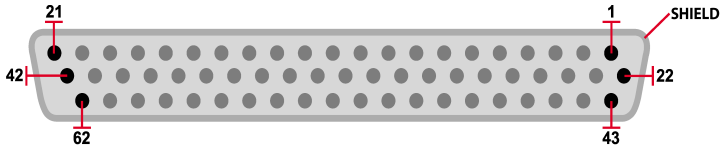

Figure 2-3 Pin Numbering for DB-62 Connector

Figures containing pinout descriptions for each measurement type supported by the DNx-MUX-461 Series Multiplexer Boards are shown in **Figure 2-4** through **Figure 2-7**. Table 2-3 shows the pin name and signal descriptions common to all the figures.

Table 2-3 DNx-MUX-461 Series DMM and Sync Pin Descriptions

Pin Name	Signal
DMM I+ DMM I-	DMM current measurement pins
DMM V+ / R+ DMM V- / R-	DMM voltage measurement pins or resistance input
DMM RS+ DMM RS-	DMM resistance sense pins
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.

© Copyright 2023
United Electronic Industries, Inc.

Pin	Signal	Pin	Signal	Pin	Signal
1	Sync +3.75V	22	Sync Gnd	43	I 12A- (nc for -350)
2	Sync Out	23	Sync In	44	I 12A+ (nc for -350)
3	I 12B+ (nc for -350)	24	I 12B- (nc for -350)	45	I 11B-
4	I 11A+	25	I 11A-	46	I 11B+
5	I 10A+	26	I 10A-	47	I 9A-
6	I 10B+	27	I 10B-	48	I 9A+
7	I 9B+	28	I 9B-	49	I 8B-
8	I 8A+	29	I 8A-	50	I 8B+
9	I 7A+	30	I 7A-	51	I 6A-
10	I 7B+	31	I 7B-	52	I 6A+
11	I 6B+	32	I 6B-	53	I 5B-
12	I 5A+	33	I 5A-	54	I 5B+
13	I 4A+	34	I 4A-	55	I 3A-
14	I 4B+	35	I 4B-	56	I 3A+
15	I 3B+	36	I 3B-	57	I 2B-
16	I 2A+	37	I 2A-	58	I 2B+
17	I 1A+	38	I 1A-	59	I 0A-
18	I 1B+	39	I 1B-	60	I 0A+
19	I 0B+	40	I 0B-	61	DMM V-
20	DMM RS-	41	DMM RS+	62	DMM V+
21	DMM I-	42	DMM I+		

Figure 2-4 DNx-MUX-461 Pinout for Current Measurements

Pin	Signal	Pin	Signal	Pin	Signal
1	Sync +3.75V	22	Sync Gnd	43	V 12A- (nc for -350)
2	Sync Out	23	Sync In	44	V 12A+ (nc for -350)
3	V 12B+ (nc for -350)	24	V 12B- (nc for -350)	45	V 11B-
4	V 11A+	25	V 11A-	46	V 11B+
5	V 10A+	26	V 10A-	47	V 9A-
6	V 10B+	27	V 10B-	48	V 9A+
7	V 9B+	28	V 9B-	49	V 8B-
8	V 8A+	29	V 8A-	50	V 8B+
9	V 7A+	30	V 7A-	51	V 6A-
10	V 7B+	31	V 7B-	52	V 6A+
11	V 6B+	32	V 6B-	53	V 5B-
12	V 5A+	33	V 5A-	54	V 5B+
13	V 4A+	34	V 4A-	55	V 3A-
14	V 4B+	35	V 4B-	56	V 3A+
15	V 3B+	36	V 3B-	57	V 2B-
16	V 2A+	37	V 2A-	58	V 2B+
17	V 1A+	38	V 1A-	59	V 0A-
18	V 1B+	39	V 1B-	60	V 0A+
19	V 0B+	40	V 0B-	61	DMM V-
20	DMM RS-	41	DMM RS+	62	DMM V+
21	DMM I-	42	DMM I+		

Figure 2-5 DNx-MUX-461 Pinout for Voltage Measurements

© Copyright 2023
United Electronic Industries, Inc.

Pin	Signal	Pin	Signal	Pin	Signal
1	Sync +3.75V	22	Sync Gnd	43	R 12A- (nc for -350)
2	Sync Out	23	Sync In	44	R 12A+ (nc for -350)
3	R 12B+ (nc for -350)	24	R 12B- (nc for -350)	45	R 11B-
4	R 11A+	25	R 11A-	46	R 11B+
5	R 10A+	26	R 10A-	47	R 9A-
6	R 10B+	27	R 10B-	48	R 9A+
7	R 9B+	28	R 9B-	49	R 8B-
8	R 8A+	29	R 8A-	50	R 8B+
9	R 7A+	30	R 7A-	51	R 6A-
10	R 7B+	31	R 7B-	52	R 6A+
11	R 6B+	32	R 6B-	53	R 5B-
12	R 5A+	33	R 5A-	54	R 5B+
13	R 4A+	34	R 4A-	55	R 3A-
14	R 4B+	35	R 4B-	56	R 3A+
15	R 3B+	36	R 3B-	57	R 2B-
16	R 2A+	37	R 2A-	58	R 2B+
17	R 1A+	38	R 1A-	59	R 0A-
18	R 1B+	39	R 1B-	60	R 0A+
19	R 0B+	40	R 0B-	61	DMM V-
20	DMM RS-	41	DMM RS+	62	DMM V+
21	DMM I-	42	DMM I+		

Figure 2-6 DNx-MUX-461 Pinout for 2-wire Resistance Measurements

Pin	Signal	Pin	Signal	Pin	Signal
1	Sync +3.75V	22	Sync Gnd	43	R 12A- (nc for -350)
2	Sync Out	23	Sync In	44	R 12A+ (nc for -350)
3	RS 12B+ (nc for -350)	24	RS 12B- (nc for -350)	45	RS 11B-
4	R 11A+	25	R 11A-	46	RS 11B+
5	R 10A+	26	R 10A-	47	R 9A-
6	RS 10B+	27	RS 10B-	48	R 9A+
7	RS 9B+	28	RS 9B-	49	RS 8B-
8	R 8A+	29	R 8A-	50	RS 8B+
9	R 7A+	30	R 7A-	51	R 6A-
10	RS 7B+	31	RS 7B-	52	R 6A+
11	RS 6B+	32	RS 6B-	53	RS 5B-
12	R 5A+	33	R 5A-	54	RS 5B+
13	R 4A+	34	R 4A-	55	R 3A-
14	RS 4B+	35	RS 4B-	56	R 3A+
15	RS 3B+	36	RS 3B-	57	RS 2B-
16	R 2A+	37	R 2A-	58	RS 2B+
17	R 1A+	38	R 1A-	59	R 0A-
18	RS 1B+	39	RS 1B-	60	R 0A+
19	RS 0B+	40	RS 0B-	61	DMM V-
20	DMM RS-	41	DMM RS+	62	DMM V+
21	DMM I-	42	DMM I+		

Figure 2-7 DNx-MUX-461 Pinout for 4-wire Resistance Measurements

© Copyright 2023
United Electronic Industries, Inc.

Chapter 3 PowerDNA Explorer

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

- Introduction (Section 3.1)
- MUX-461 Configuration (Section 3.2)
- Relay Counters (Section 3.3)

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 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).

NOTE: Up to five DNx-MUX-461 Series Multiplexer Boards may be connected to a DNx-DMM-261. However, you must ensure that only one MUX board at a time is sending output to the DMM in order to avoid relay damage in both the MUX and DMM.

© Copyright 2023
United Electronic Industries, Inc.

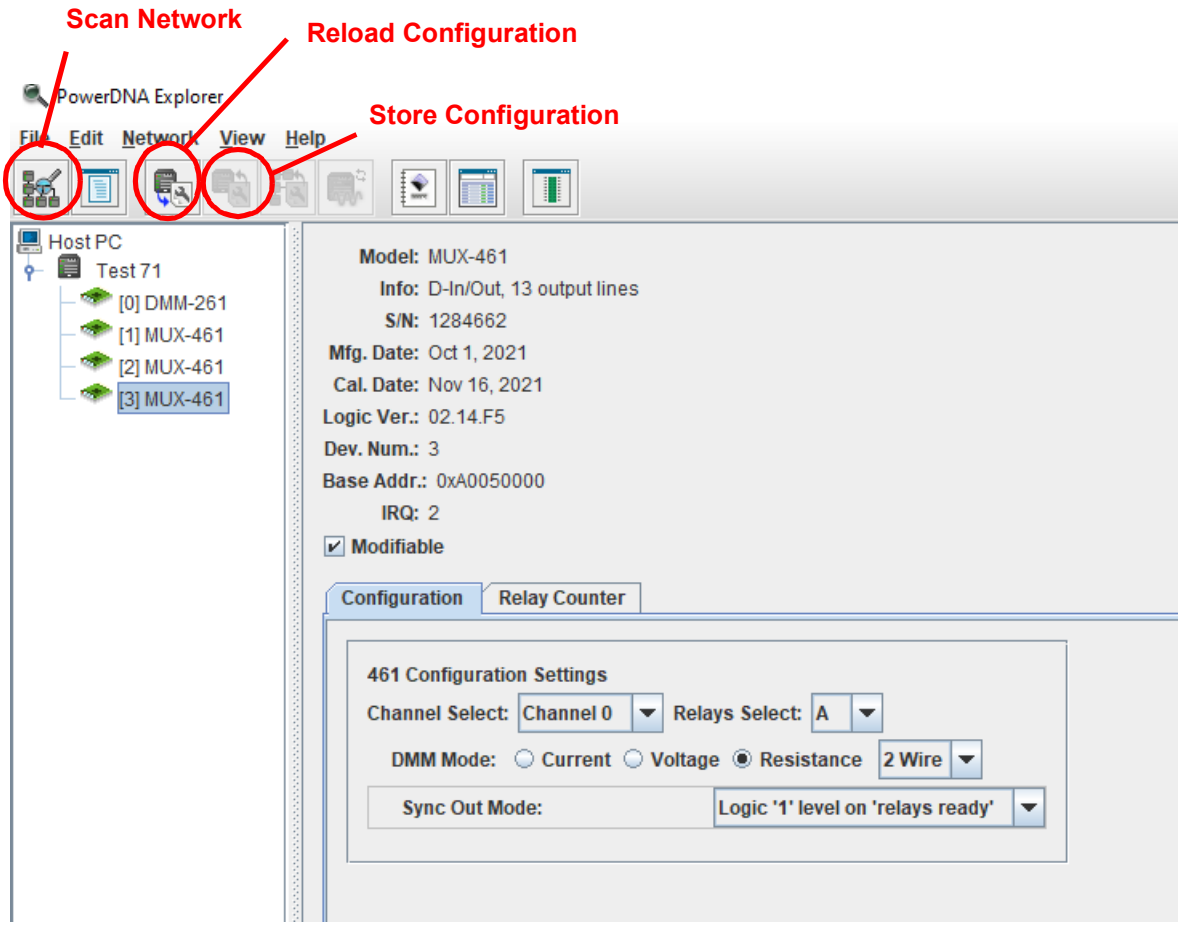

Figure 3-1 PowerDNA Explorer for DNx-MUX-461

When using PowerDNA Explorer for the DNx-MUX-461 Series, the right-hand panel contains a single Configuration tab. The Configuration tab allows selection of the MUX channel and relay to connect to the DNx-DMM-261.

NOTE: DNx-MUX-461 Series Multiplexer Boards are supported in PowerDNA version 5.2+.

3.2 MUX-461 Configuration

The **Configuration** tab, shown in **Figure 3-1**, contains the following configuration items:

- **Channel Select**: Drop-down to select the channel number (0-12) on the selected bus. Users can select either the A bus or B bus (see "**Relays Select**") so there are up to 26 channels available (24 for DNx-MUX-461-350 boards). Selecting channel 12 will have no effect for DNx-MUX-461-350 boards.
- **Relays Select**: Drop-down to select either A bus or B bus relays. Select "Off" to open all the channel relays.
- **DMM Mode**: selects the DMM measurement mode and controls which DMM pins the channel is connected to (switches "D" relays). If "4 wire" Resistance mode is selected, the Relays Select drop-down will be disabled and both A and B relays will be closed for the selected channel.
- **Sync Out Mode**: signal to output on SYNC OUT pin.

The new settings will not take effect until the configuration is saved by clicking the **Store Configuration** button.

Note that for a valid configuration, DNx-MUX-461 boards must be seated in adjacent slots starting with the slot immediately to the right of the DNx-DMM-261. In a cube chassis, the DNx-MUX-461 boards must be seated in adjacent slots starting with the slot immediately below the DNx-DMM-261. An example of a valid configuration is shown in Figure 3-2.

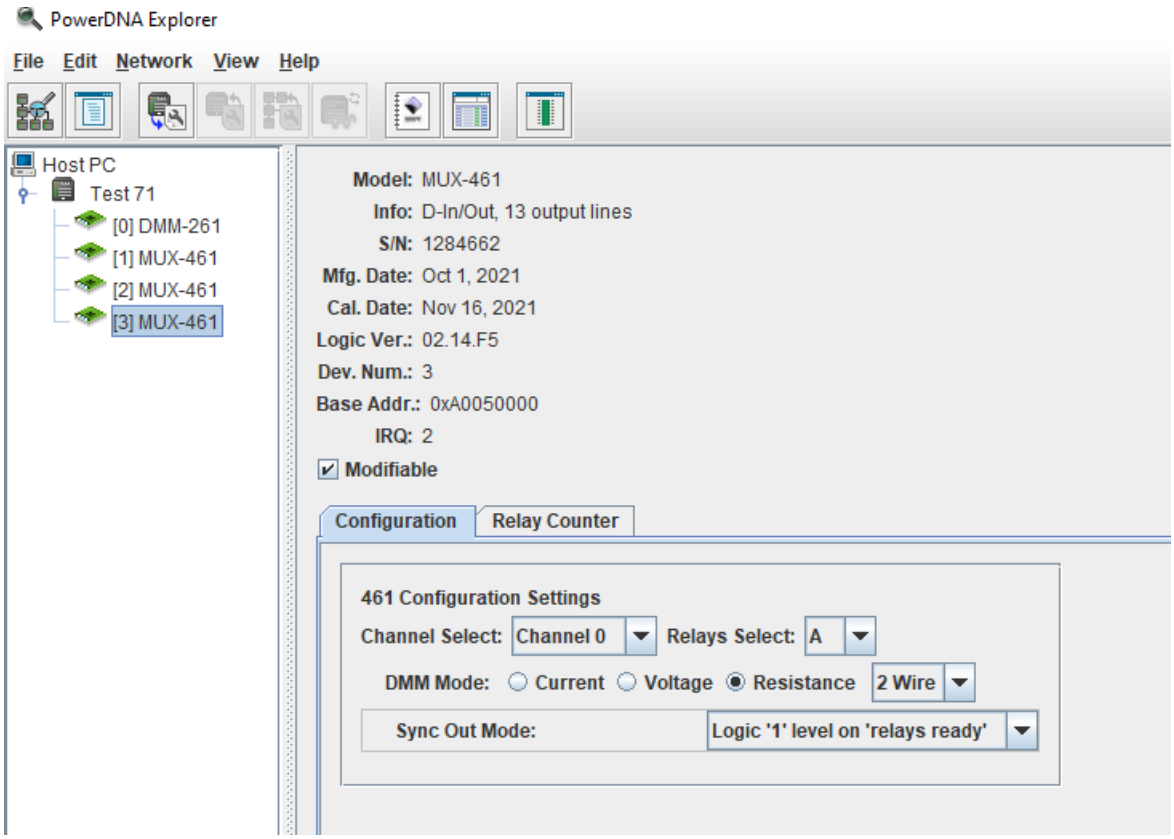

Figure 3-2 PowerDNA Explorer Configuration Tab

3.3 Relay Counters

The **Relay Counter** tab (**Figure 3-3**) shows the number of times each relay has been energized. The A/B relays are for channel selection and the D relays are used for DMM mode selection.

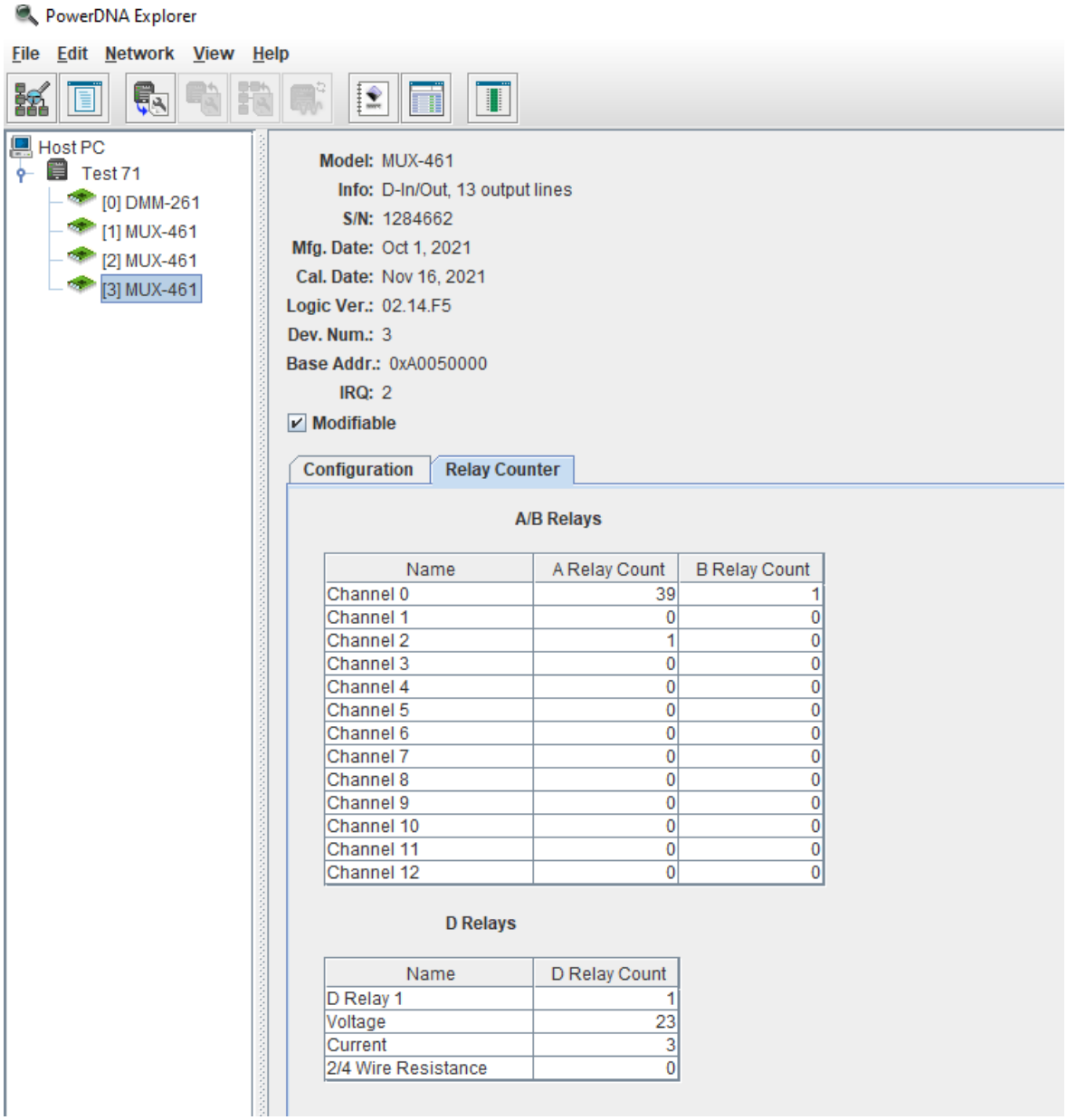

Figure 3-3 PowerDNA Explorer Relay Counter Tab

Chapter 4 Programming with the High-level API

This chapter provides the following information about programming DNx-MUX-461 Series boards using the UeiDaq Framework (high-level) API:

- About the High-level API (Section 4.1)
- Example Code (Section 4.2)
- Framework Methods (Section 4.3)
- Create a Session (Section 4.4)
- Resource String (Section 4.5)
- Configure the Sessions (Section 4.6)
- Configure the Timing (Section 4.7)
- Start the Sessions (Section 4.8)
- Write Data (Section 4.9)
- Read Diagnostic Data (Section 4.10)
- Stop the Session (Section 4.11)

4.1 About the High-level API

The 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 Framework 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

For information on the Framework's classes, structures, and constants, please see the *"UeiDaq Framework Reference Manual"* located under:

Start » All Programs » UEI

4.2 Example Code

The UeiDaq Framework is bundled with examples for supported programming languages. The example code is located under the appropriate language-specific folder under:

Start » All Programs » UEI

For example:

Start » All Programs » UEI » Visual C++ Examples

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

© Copyright 2023
United Electronic Industries, Inc.

Each UeiDaq Framework application can be created using the following 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 the 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.

Up to five DNx-MUX-461 Series boards can be used together in a UEI chassis to multiplex up to 130 two-wire or 65 four-wire channels to a DNx-DMM-261. The UeiDaq Framework API provides functionality that allows the creation of a "MuxGroup". This simplifies the programming needed to manage safe connection between DNx-MUX-461 Series boards and a DNx-DMM-261 so that applications are prevented the from accidentally connecting multiple signals to the DMM.

This chapter will provide information on how to use the UeiDaq Framework to program both individual DNx-MUX-461 Series boards as well as DNx-MUX-461 Series boards that are part of a MuxGroup providing input signals to a DNx-DMM-261.

4.3 Framework Methods

Code snippets presented in this chapter show how to use various UeiDaq Framework classes to configure DNx-MUX-461 Series boards and create a MuxGroup for connection to a DNx-DMM-261 board. For additional information and a complete list of relevant classes and methods, refer to the *"UeiDaq Framework Reference Manual"*.

4.3.1 CUeiMuxPort

The `CUeiMuxPort` class is used to configure and manage DNx-MUX-461 Series boards. The `CUeiMuxPort` class implements the methods listed in **Table 4-1**.

Table 4-1 CUeiMuxPort Class Methods

Get Methods	Set Methods
`IsBreakBeforeMakeEnabled`	`EnableBreakBeforeMake`
`IsSyncInputEnabled`	`EnableSyncInput`
`IsSyncInputEdgeModeEnabled`	`EnableSyncInputEdgeMode`
`GetSyncInputEdgePolarity`	`SetSyncInputEdgePolarity`
`GetSyncOutputMode`	`SetSyncOutputMode`
`GetSyncOutputPulseWidth`	`SetSyncOutputPulseWidth`

© Copyright 2023
United Electronic Industries, Inc.

Table 4-1 CUeiMuxPort Class Methods

Get Methods	Set Methods
`GetOnDelay`	`SetOnDelay`
`GetOffDelay`	`SetOffDelay`

4.3.2 MuxGroup API

The UeiDaq Framework provides the ability to create a MuxGroup for safely connecting up to five MUX devices to a DMM. The MuxGroup API ensures that within a MuxGroup, only one MUX channel at a time is connected to a DMM.

The MuxGroup API is enabled by calling `CUeiDMMChannel::SetMuxDeviceCount()`. See Section 4.6 for information on creating a `CUeiDMMChannel` object.

A MUX device connection to a DMM is configured by calling `CUeiDMMWriter::WriteDMMMuxConfig (muxDeviceNum, muxChannel, muxRelaySelect)`.

4.3.3 Reading and Writing MUX Data

The UeiDaq Framework provides the `CUeiMuxWriter` class for reading from and writing to DNx-MUX-461 Series boards. The methods provided by the `CUeiMuxWriter` class are listed in **Table 4-2**.

Table 4-2 CUeiMuxWriter Class Methods

CUeiMuxWriter Methods	Description
`WriteRelays`	Set relays individually
`WriteMux`	Close relays on a set of channels
`WriteMuxRaw`	Set low-level format MUX values
`WriteMuxDmm`	Set relay state and DMM mode on selected MUX channel
`ReadStatus`	Get status of each relay on the MUX device
`ReadADC`	Read diagnostic ADC values
`ReadRelayCounts`	Read current count of times each relay has been energized

4.4 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 for each device.

```
//create a session object

CUeiSession muxSession; // if directly configuring a MUX-461
CUeiSession dmmSession; // if creating a MuxGroup for a DMM-261
```

© Copyright 2023
United Electronic Industries, Inc.

Because each session is dedicated to a specific device and subsystem, the application must create an additional session for the DNx-DMM-261 in order to create a MuxGroup. In a rack chassis, the Framework assumes that the DNx-MUX-461 boards are seated in adjacent slots starting with the slot immediately to the right of the DNx-DMM-261. In a cube chassis, the Framework assumes that the DNx-MUX-461 boards are seated in adjacent slots starting with the slot immediately below the DNx-DMM-261.

If there are no specific configuration or status reads for the DNx-MUX-461, then only a DMM session is needed. The DMM session will handle tasks such as setting relays.

4.5 Resource String

The 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. The syntax and components of a resource string are as follows:

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

- *<device class>* – By default, Framework examples 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 Series boards are supported by the `Mux` subsystem. DNx-DMM-261 boards are supported by the `Ai` subsystem.
- *<channel list>* – desired channels within the selected subsystem. All DNx-MUX-461 channels are included in channel 0. All DNx-DMM-261 measurements use channel 0.

4.6 Configure the Sessions

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

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

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

The input parameters are:

- `resource` – specifies the board and subsystem (see Section 4.5).
- `breakBeforeMake` – enables or disables break-before-make functionality on “D” relays (see Section 4.6.1).

© Copyright 2023
United Electronic Industries, Inc.

In order to create a MuxGroup in a system with a DNx-DMM-261, a `CUeiDMMChannel` object must be created using a separate session and then calling `SetMuxDeviceCount()`.

```
//Configure session for DMM channel and create a MuxGroup with 5 MUXes

CUeiDMMChannel* dmmChannel = dmmSession.CreateDMMChannel
                                ("pdna://192.168.100.2/Dev0/ai0",
                                 10,
                                 UeiDMMModeCalibratedDCVoltage);

dmmChannel->SetMuxDeviceCount(5);
```

Note the following about the MuxGroup API:

- break-before-make is handled internally by the MuxGroup API
- A MuxGroup may only be configured for one type of DMM measurement. Therefore, when switching measurement types, the existing MuxGroup must be destroyed before a new one is created. This can be done by either ending the DMM Channel session or setting the MUX device count to zero.

4.6.1 Break-before-Make and Port On Delays

By default, DNx-MUX-461 Series boards implement 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") and the delay before a new relay write command is accepted ("on delay"). You can also optionally disable break-before-make for "D" relays only. The DNx-MUX-461 Series boards support delay times between 100 microseconds and 25.6 milliseconds in 100 microsecond increments. See **Table 1-1** for relay turn-off time and turn-on time. Be aware not to set the "off delay" and "on delay" to a time greater than the application wait time to avoid skipping a write to the relays.

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

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

You can enable or disable break-before-make for the "D" relays when you first create the MUX port, or you can use the `EnableBreakBeforeMake()` method shown below. Disabling break-before-make is useful when your application does not need to change the DMM mode, as it can extend the life of the relays. This is more important for the reed relays used in the standard DNx-MUX-461 boards.

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

port->EnableBreakBeforeMake(false);
```

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

© Copyright 2023
United Electronic Industries, Inc.

4.6.2 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 4.6.2.1. Then, configure and enable SYNC IN on the second board as described in Section 4.6.2.2. 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.

4.6.2.1 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
- `UeiMuxSyncOutputRelaysReadyLogic0` – drive logic '0' level while relays are settled

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

```
//Set SYNC OUT pulse width to 1 millisecond. Argument is in microseconds.
port->SetSyncOutputPulseWidth(1000);
```

4.6.2.2 Sync In

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

```
//Enable SYNC IN.
port->EnableSyncInput(true);
```

© Copyright 2023
United Electronic Industries, Inc.

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 4.11 and Section 4.8). This allows the latest relay writes to go through. All subsequent writes are handled normally.

4.7 Configure the Timing

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

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

muxSession.ConfigureTimingForSimpleIO();
```

4.8 Start the Sessions

After each session is configured, you can start a session manually:

```
//Start the sessions.

muxSession.Start();
dmmSession.Start();
```

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

4.9 Write Data

Relays can be switched on and off either by using a `CUeiMuxWriter` object for directly writing to a DNx-MUX-461 Series board or by using a `CUeiDMMWriter` object when a MuxGroup has been created.

4.9.1 MuxWriter

A `CUeiMuxWriter` object is created as follows. A complete list of `CUeiMuxWriter` methods is provided in **Table 4-2**.

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

After creating the writer object, call the `WriteMuxDmm()` method to connect a multiplexer channel to the DMM. A break-before-make (Section 4.6.1) ensures that a maximum of one channel will be connected at any given time.

```
//Select channel number 12 on bus B.
int channel = 12;
tUeiMuxRelaySelect relaySelect = UeiMuxRelaySelectB;
//Close the B12 relay (V 25+ and V 25-).
//Output to DMM V+ and DMM V- by closing D1 and D2 relays.
writer.WriteMuxDmm(channel, relaySelect, UeiMuxDmmMeasV);
```

The `tUeiMuxRelaySelect` type is defined as follows:

```
typedef enum _tUeiMuxRelaySelect
{
   UeiMuxRelaySelectDisable = 0, ///< Disable relays on selected channel
   UeiMuxRelaySelectA = 1,       ///< Enable A relay
   UeiMuxRelaySelectB = 2,       ///< Enable B relay
   UeiMuxRelaySelectAB = 3,      ///< Enable A and B relay (4-wire mode)
} tUeiMuxRelaySelect;
```

The supported DMM modes are defined in `tUeiMuxDmmMode`:

```
typedef enum _tUeiMuxDmmMode
{
   UeiMuxDmmDisable,   ///< disconnect all channels from DMM
   UeiMuxDmmMeasI,     ///< current measurement
   UeiMuxDmmMeasV,     ///< voltage measurement
   UeiMuxDmmMeasRes2,  ///< two-wire resistance measurement
   UeiMuxDmmMeasRes4   ///< four-wire resistance measurement
                       ///< closes both A and B relays - relay
                       ///< selection is ignored
} tUeiMuxRelaySelect;
```

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

4.9.2 DMMWriter (MuxGroup)

If a MuxGroup has been created, relays can be safely connected to a DMM by calling

```
CUeiDMMWriter::WriteDMMMuxConfig
              (muxDeviceNum, muxChannel, muxRelaySelect.
```

Data can be read from a DNx-DMM-261 by creating a `DMMReader` object and calling one of its methods, e.g., `ReadSingleScan()`. The `DMMReader` methods are described in the "*UeiDaq Framework Reference Manual*". After the call to `WriteDMMMuxConfig()`, your application should wait at least the amount of time specified by `SetOnDelay`() before the DMM data is read. After the DMM data is read, the application should then wait at least the amount of time specified by `SetOffDelay`() before the next write.

4.10 Read Diagnostic Data

DNx-MUX-461 Series boards monitor DC internal supply voltages and temperature using an on-board ADC. It can also read back the current state, status, and number of cycles for each relay. You can read from the diagnostic ADC channels described in **Table 4-3**.

Table 4-3 Diagnostic Channels

Channel #	Description
0	Voltage of internal 24 V supply (standard MUX-461 only)
1	Voltage of internal 3.3 V supply
2	Internal relay driver voltage (standard MUX-461 only)
3	Temperature in degrees C
4	Status (uint32, see **Table 4-4**)
5	Timestamp

Create a `CUeiMuxWriter` object (Section 4.9) and use its `ReadADC()` method:

```
//Read all 6 diagnostic channels.

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

4.10.1 Relay States and Status

You can monitor current relay states and board status with the `CUeiMuxWriter` 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.

© Copyright 2023
United Electronic Industries, Inc.

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

Table 4-4 Relay Status

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

4.10.2 Relay Counts

The `ReadRelayCounts()` method in the `CUeiMuxWriter` 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;
}
```

© Copyright 2023
United Electronic Industries, Inc.

4.11 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();
```

© Copyright 2023
United Electronic Industries, Inc.

Chapter 5 Programming with the Low-level API

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

- About the Low-level API (Section 5.1)
- Example Code (Section 5.2)
- Data Acquisition Modes (Section 5.3)
- Point-by-Point API (Section 5.4)
- MuxGroup API (with DMM-261) (Section 5.5)

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, 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**

NOTE: The DNx-MUX-461 Series is supported in PowerDNA version 5.2+. If you are unsure if your version supports tour board, please contact Technical Support at support@ueidaq.com

5.2 Example Code

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

- On Linux: *<PowerDNA-x.y.z>/src/DAQLib_Samples*
- On Windows: *C:\Program Files (x86)\UEI\PowerDNA\SDK\Examples*

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

© Copyright 2023
United Electronic Industries, Inc.

5.3 Data Acquisition Modes

The following data acquisition (DAQ) mode is available for transferring data between the DNx-MUX-461 Series boards 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 file locations listed in Section 5.1.

5.4 Point-by-Point API

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

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

Function	Description
`DqAdv461SetChannel`	Select a multiplexer channel and connect output to a DMM; optionally enable sync in/out for the write
`DqAdv461ReadADC`	Read diagnostic internal voltages and board temperature
`DqAdv461SetCfg`	Configure break-before-make and sync in/out functionality
`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 example code and the API reference manual.

© Copyright 2023
United Electronic Industries, Inc.

5.4.1 Configuration Settings

DNx-MUX-461 Series boards are configured using the `DqAdv461Cfg()` function:

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

The function takes in the following inputs and updates the board's configuration.

- `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 subsequent call can add or change a parameter in the board's configuration. Refer to the "*PowerDNA API Reference Manual*" for a complete description of member of the configuration structure.

```
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;
```

5.4.2 Programming Relays

This section describes how to use the `DqAdv461SetChannel()` API function to connect a single 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 - 0...12 for standard DNx-MUX-461 boards, 0...11 for DNx-MUX-461-350 boards. There are two banks of relays. The `relay_select` parameter selects which bank by specifying either the A bus or B bus thereby providing up to 26 channels (up to 13 channels for 4-wire resistance measurements).
- `uint32 relay_select` – open or close relay on bus A and/or bus B
- `uint32 dmm_mode` – measure voltage, current, two-wire resistance, or four-wire resistance
- `uint32 sync` – flags to enable SYNC IN and SYNC OUT pins

The break-before-make feature 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. The “D” relays are used to select the programmed DMM mode and can remain closed as long as the DMM mode doesn’t change, thereby prolonging the life of the reed relays on standard DNx-MUX-461 boards. This is done by setting `d_bbm_mode` to 0 during configuration.

Example: Two-wire Measurement

```
//Close the B12 relay (V 25+ and V 25-).
//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: Four-wire Measurement

```
//Close the A12 and B12 relays (V 12A+, V 12A-, RS 12B+, RS 12B-).
//Connect A12 to DMM V+ and DMM V- by closing D2 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);
```

5.4.3 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.

© Copyright 2023
United Electronic Industries, Inc.

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 on DEVN0 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 on DEVN1 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);
```

5.5 MuxGroup API (with DMM-261)

When one or more DNx-MUX-461 Series Multiplexer Boards are installed in your system with a DNx-DMM-261 board, the MuxGroup API can be used to ensure that the MUX boards safely connect to the DMM. The MuxGroup API ensures that within a MuxGroup, only one MUX channel at a time is connected to a DNx-DMM-261.

The low-level API MuxGroup functions are listed in **Table 5-2**. See the "*DNx-DMM-261 User Manual*" for more information on the DNx-DMM-261. See the "*PowerDNA API Reference Manual*" for more information on the MuxGroup API functions.

See `Sample261MuxGroup` for a an example of setting up a MuxGroup and reading DMM measurements. The example includes typical initialization, error handling, and usage of these functions. The remainder of this chapter is intended as a supplement to the example code and the API reference manual.

Table 5-2 MuxGroup API for using DNx-MUX-461 Boards with a DNx-DMM-261

Function	Description
`DqAdv261MuxGroupCreate`	Link a DMM-261 and up to five MUX layers into a group.
`DqAdv261MuxGroupDestroy`	Release DMM and all MUX layers from a MuxGroup.
`DqAdv261MuxGroupDisconnect`	Disconnect all MUX channels from the DMM-261 without connecting a new input channel or destroying the MuxGroup.
`DqAdv261MuxGroupSet`	Safely configure the MuxGroup relays for the specified MUX input channel and DMM mode before a call to `DqAdv261MuxGroupRead()`.
`DqAdv261MuxGroupRead`	Read data from a DMM that is part of a MuxGroup.

© Copyright 2023
United Electronic Industries, Inc.

Appendix A

Accessories

A.1 Cables

The following cables are available for DNx-MUX-461 Series Multiplexer Boards.

DNA-CBL-62 Cables

The DNA-CBL-62 family of cables consists of 62-conductor, round, heavy-shielded cables available in several lengths. The cables feature 62-pin male D-sub connectors on both ends. The available options are listed in Table A-1.

Table A-1 DNA-CBL-62 Product Options

Cable	Length
DNA-CBL-62	2.5 ft. (0.76 m)
DNA-CBL-62-6	6 ft. (1.83 m)
DNA-CBL-62-10	10 ft. (3.05 m)
DNA-CBL-62-20	20 ft. (6.1 m)

DNR-CBL-461-x (x can be 1, 2, 3, 4, or 5)

The DNR-CBL-461-x cable connects a DNR-DMM-261 board to as many as five DNx-MUX-461 Series boards inside of a Rack chassis. The cable plugs into the JPOW1 connectors on the PCBs in a daisy-chain configuration as described in the "*DNx-DMM-261 User Manual*". Five different cable options are available (**Figure A-1**), where x corresponds to the number of MUX boards in the system.

Figure A-1 DNR-CBL-461-x Cables

© Copyright 2023
United Electronic Industries, Inc.

A.2 Screw Terminal Panel

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 (with standoffs) are 4” w x 3.8” d x1.2” h (10.2 x 9.7 x 3 cm). The weight of the STP-62 board is 3.89 ounces (110 grams).

A photo of the STP-62 board is shown in **Figure A-2**. A pinout diagram is shown in **Figure A-3**.

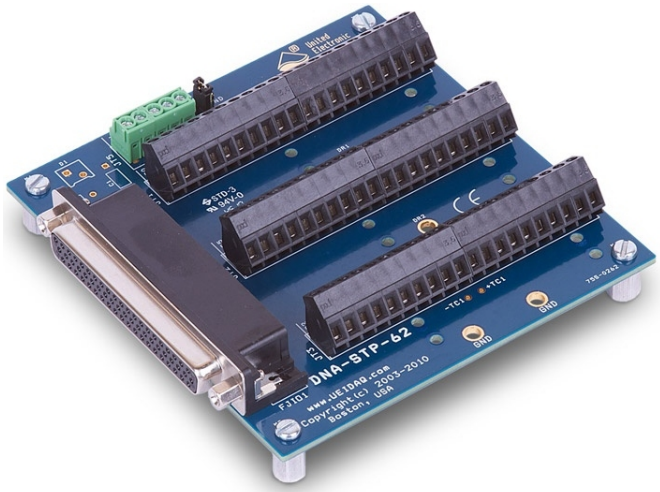

Figure A-2 Photo of DNA-STP-62 Screw Terminal Panel

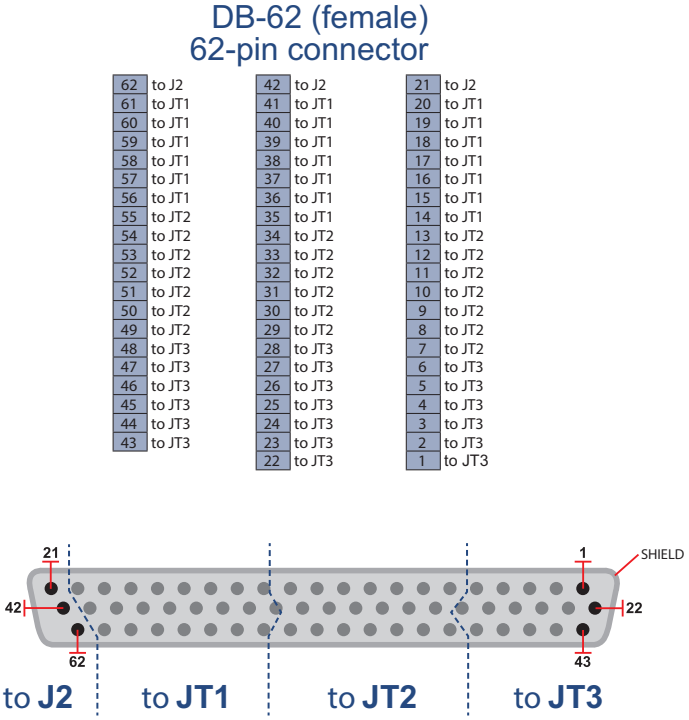

Figure A-3 Pinout of DNA-STP-62 Screw Terminal Panel