

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

GX1838

Multifunction DC Source PXI Board

GXPDO SOFTWARE

User's Guide

Last Updated August 5, 2019

MARVIN TEST SOLUTIONS

Safety and Handling

Each product shipped by Marvin Test Solutions is carefully inspected and tested prior to shipping. The shipping box provides protection during shipment, and can be used for storage of both the hardware and the software when they are not in use.

The circuit boards are extremely delicate and require care in handling and installation. Do not remove the boards from their protective plastic coverings or from the shipping box until you are ready to install the boards into your computer.

If a board is removed from the computer for any reason, be sure to store it in its original shipping box. Do not store boards on top of workbenches or other areas where they might be susceptible to damage or exposure to strong electromagnetic or electrostatic fields. Store circuit boards in protective anti-electrostatic wrapping and away from electromagnetic fields.

Be sure to make a single copy of the software CD for installation. Store the original CD in a safe place away from electromagnetic or electrostatic fields. Return compact disks (CD) to their protective case or sleeve and store in the original shipping box or other suitable location.

Warranty

Marvin Test Solutions products are warranted against defects in materials and workmanship for a period of 12 months. Marvin Test Solutions shall repair or replace (at its discretion) any defective product during the stated warranty period. The software warranty includes any revisions or new versions released during the warranty period. Revisions and new versions may be covered by a software support agreement. If you need to return a board, please contact Marvin Test Solutions Customer Technical Services Department via http://www.marvintest.com/magic/- the Marvin Test Solutions on-line support system.

If You Need Help

Visit our web site at http://www.marvintest.com for more information about Marvin Test Solutions products, services and support options. Our web site contains sections describing support options and application notes, as well as a download area for downloading patches, example, patches and new or revised instrument drivers. To submit a support issue including suggestion, bug report or questions please use the following link: http://www.marvintest.com/magic/

You can also use Marvin Test Solutions technical support phone line (949) 263-2222. This service is available between 8:30 AM and 5:30 PM Pacific Standard Time.

Disclaimer

In no event shall Marvin Test Solutions or any of its representatives be liable for any consequential damages whatsoever (including unlimited damages for loss of business profits, business interruption, loss of business information, or any other losses) arising out of the use of or inability to use this product, even if Marvin Test Solutions has been advised of the possibility for such damages.

Copyright

Copyright © 2001-2019 Marvin Test Solutions. All rights reserved. No part of this document can be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Marvin Test Solutions.

Trademarks

ATEasy®, CalEasy, DIOEasy®, DtifEasy, WaveEasy	Marvin Test Solutions (prior name is Geotest - Marvin Test Systems Inc.)
C++ Builder, Delphi	Embarcadero Technologies Inc.
LabView, LabWindows™/CVI	National Instruments
Microsoft Developer Studio, Microsoft Visual C++, Microsoft Visual Basic,, .NET, and Windows	Microsoft Corporation

All other trademarks are the property of their respective owners.

Table of Contents

Safety and Handling....i
Warranty....i
If You Need Help....i
Disclaimer....i
Copyright....i
Trademarks....ii
Chapter 1 - Introduction....1
Manual Scope and Organization....1
Manual Scope....1
Manual Organization....1
Conventions Used in this Manual....1
Chapter 2 - Overview....3
Introduction....3
Features....3
Applications....3
Board Description....4
Architecture....6
Block Diagram....6
Output Channels....6
Board Rails....7
Rail Output Voltage With Different Loads....7
Specifications....8
Power Requirements....8
Virtual Panel Description....9
Virtual Panel Initialize Dialog....10
Virtual Panel Setup Page....11
Virtual Panel About Page....12
Chapter 3 - Installation and Connections....13
Getting Started....13
Packing List....13
Unpacking and Inspection....13
System Requirements....13
Installation of the GXPDO Software....14
Overview of the GXPDO Software....15
Configuring Your PXI System using the PXI/PCI Explorer....16

Board Installation....17
Before you Begin....17
Electric Static Discharge (ESD) Precautions....17
Installing a Board....17
Plug & Play Driver Installation....18
Removing a Board....19
Connectors and Accessories....19
Connectors....20
Output Channels Connector(J3)....20
Installation Folders....21
GXPDO Driver Files Description....21
Driver File and Virtual Panel....21
Interface Files....21
On-line Help and Manual....22
ReadMe File....22
Example Programs....22
Setup Maintenance Program....23
Chapter 4 - Programming the Board....25
The GXPDO Driver....25
Programming Using C/C++ Tools....25
Programming Using Visual Basic....25
Programming Using C#....25
Programming Using Pascal/Delphi....26
Programming Using ATEasy®....26
Programming Using LabView and LabView/Real Time....26
Using and Programming under Linux....26
Using the GXPDO driver functions....27
Initialization, HW Slot Numbers and VISA Resource....27
Board Handle....28
Reset....28
Error Handling....28
Driver Version....28
Panel....28
Distributing the Driver....29
Sample Programs....29
Sample Program Listing....30
Chapter 5 - Functions Reference....37

Introduction....37
GX1838 Functions....38
Gx1838CalByUserDeleteCalibration....39
Gx1838CalByUserReadMeasuredVoltage....40
Gx1838CalByUserSetup....41
Gx1838CalByUserSetupDacForMeasurement....42
Gx1838CalByUserStoreDacCalibrationData....43
Gx1838CalByUserWriteMeasuredVoltage....44
Gx1838GetBoardSummary....45
Gx1838GetChannelRail....46
Gx1838GetDacOutputRange....47
Gx1838GetDacCalibrationDataSource....48
Gx1838GetDacCalibrationDataSources....49
Gx1838GetDacCalibrationDataString....50
Gx1838GetDacVoltage....51
Gx1838GetRailSource....52
Gx1838Initialize....53
Gx1838InitializeVisa....54
Gx1838Panel....55
Gx1838Reset....56
Gx1838SetChannelRail....57
Gx1838SetDacCalibrationDataSource....58
Gx1838SetDacVoltage....60
Gx1838SetRailSource....61
GxPdoGetDriverSummary....62
GxPdoGetErrorString....63
Index....67

Chapter 1 - Introduction

Manual Scope and Organization

Manual Scope

The purpose of this manual is to provide all the necessary information to install, use, and maintain the GX1838 instrument. This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and some understanding of digital I/O.

This manual also provides programming information using the GX1838 driver (referred in this manual **GXPDO**). Therefore, good understanding of programming development tools and languages may be necessary.

Manual Organization

The GX1838 manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX1838 manual. Lists all the supported board and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX1838 features, board description, its architecture, specifications and the panel description and operation.
Chapter 3 –Installation and Connections	Provides instructions on how to install the GX1838 Board and its accompanying GXPDO software.
Chapter 4 – Programming the Board	Provides a listing of GX1838 driver files, general purpose/generic driver functions, and programming methods. Discusses various supported operating systems and development tools.
Chapter 5 – Functions Reference	Contains a listing of the general GX1838 functions. Each function is described along with its syntax, parameters, and special programming comments. Samples are given for each function.

Conventions Used in this Manual

Symbol Convention	Meaning
	Static Sensitive Electronic Devices. Handle Carefully.
	Warnings that may pose a personal danger to your health. For example, shock hazard.
	Cautions where computer components may be damaged if not handled carefully.
	Tips that aid you in your work.

Formatting Convention	Meaning
`Monospaced Text`	Examples of field syntax and programming samples.
Bold type	Words or characters you type as the manual instructs. For example: function or panel names.
Italic type	Specialized terms. Titles of other reference books. Placeholders for items you must supply, such as function parameters

Chapter 2 - Overview

Introduction

The GX1838 is a 3U PXI Precision DC Source with eight analog channels. Each of the analog channels can be connected to one of the three multifunction DC source rails. The rails can be connected to one of three programmable high current voltage sources, one fixed load and three external inputs. Therefore, each of the eight analog channels can be connected to any one of these functions through a corresponding rail. The GX1838 Precision Multifunction DC Source was designed for ATE, data acquisition, or process control systems. The card can be ordered in its standard version with an output voltage range of -10V to +32V or the GX1838-20 version with an output voltage range of -20V to +20V. The GX1838 is a PXI hybrid slot compatible.

Features

The GX1838 is a Precision Multifunction DC Source card with:

- Eight analog channels each can be connected to any of the three multifunction DC source rails.
- Three multifunction DC source rails, each rail has its own set of available sources.
- All rails have a high-resolution programmable high current voltage source. Voltages can be set from –10V to +32V @ 500mA for the standard version or –20V to +20V @ 500mA for the GX1838-20 version.
- All rails can be connected to an external source via the external 25-pin D connector.
- Rail C has an additional 500 Ohm fixed load.
- A single 25-pin D-Type connector for the eight Output Channels, three external signals inputs and ground signals.
- Each DAC can have up to three calibration data sources: Factory and two user defined calibration data sets, which can be non-volatile (stored to the board) and volatile (dynamic).
- Calibration data source for each DAC is set by software.
- Calibration for each DAC can be done using MTS' calibration service or using **CalEasy** software. Calibration of the card using **CalEasy** requires purchasing of a special software license.

Applications

- Simulation of discrete levels, common in avionics testing.
- UUT power applications requiring less than 500 mA of current.
- Requirements for multiple programmable, precision DC sources.

Board Description

The GX1838 is a Precision Multifunction DC Source with 8 channels. Each channel can be connected to one of three rails at a time.

The card has three individual rails (Rail A, Rail B, and Rail C). Each rail has a set of resources that it may be connected to.

Sources for Rail A:

- Open (not connected).
- High-resolution programmable high current voltage source. Voltages can be set from –10V to +32V @ 500mA for the standard version or –20V to +20V @ 500mA for the GX1838-20 version.
- External source via the 25-pin connector.

Sources for Rail B:

- Open (not connected).
- High-resolution programmable high current voltage source. Voltages can be set from –10V to +32V @ 500mA for the standard version or –20V to +20V @ 500mA for the GX1838-20 version.
- External source via the 25-pin connector.

Sources for Rail C:

- Open (not connected).
- High-resolution programmable high current voltage source. Voltages can be set from –10V to +32V @ 500mA for the standard version or –20V to +20V @ 500mAforn the GX1838-20 version.
- External source via the 25-pin connector.
- Fixed load of 500 Ohms.

Figure 2-1 shows the GX1838 with its J3 connector.

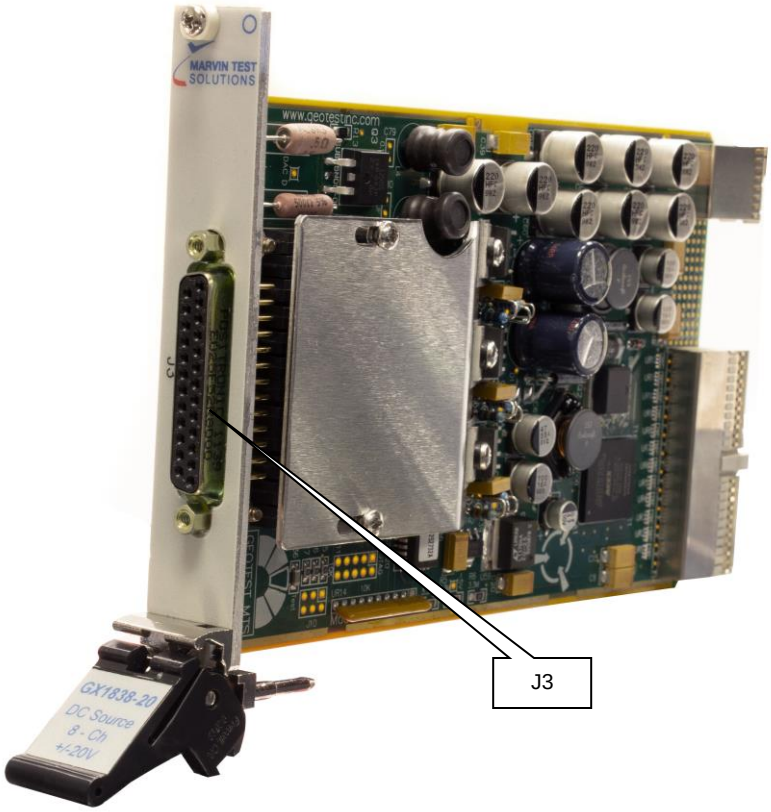


Figure 2-1: GX1838 Board, Component Side

Architecture

Block Diagram

Figure 2-2:GX1838 Block Diagram illustrates the GX1838 architecture. The board is programmed by software using the PXI interface to set the DACs' voltages; the Rails' source and the channels' rail connections. The board connector on the right (J3) provides the interface to the output the channels' signals and external input signals.

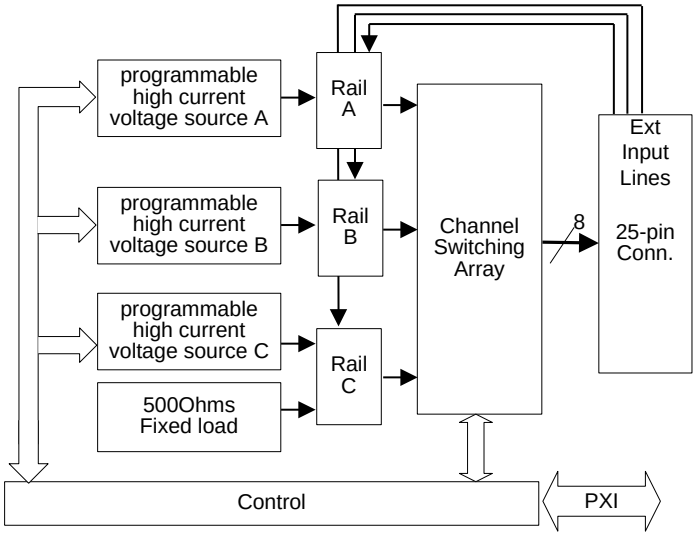


Figure 2-2:GX1838 Block Diagram

Output Channels

The GX1838 provides eight parallel analog output channels which interface through a 25-pin D-Type connector (J3). Each channel can be connected to one of the three rails or disconnected from all rails. Channel settings are done through a software driver in order to set the channel's source. Figure 2-3 shows a typical output channel diagram:

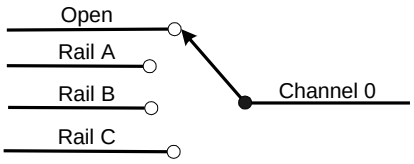


Figure 2-3: Typical Output Channel

Board Rails

The GX1838 contains three individual rails. Each rail can connect to its own set of resources. Figure 2-4: GX1838 Rail Sources, presents a diagram for the GX1838 rail connections.

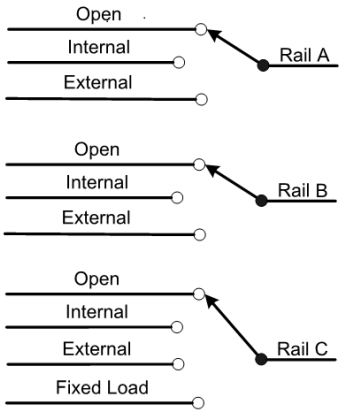


Figure 2-4: GX1838 Rail Sources

Rail Output Voltage With Different Loads

All rails (A, B and C) are calibrated with a load of 1000 0hms over the full range of –10V to 32V or -20V to +20V. A slight shift in output value may occur if the actual load is different from the specified 1000 Ohm load.

Specifications

The following table outlines the specifications of the GX1838.

Output channels	8
Voltage Rails	3
Voltage sources	3
Output Voltage Range (all voltage sources)	-10 V to +32 VDC (Standard version) -20 V to +20 VDC (GX1838-20 version)
Resolution	14-bit
Accuracy	± 14mV (with 1000 Ohm load)
Slew Rate	6 V/µs
Max Load (min. resistance) per channel	50Ω
Max. Current per channel	500 mA
Max. Current per rail	500 mA
Max. Current per GX1838	500 mA
Output Noise	< 4 mV rms, 1K ohm load, 20 – 300 KHz measurement BW
Output Protection	Short Circuit (Ground)
Power ON State	All Channels / Rails open
Operating Temperature	0 to+55°C
Storage Temperature	-20 to+70°C
Size	3U PXI, hybrid slot compatible
Weight	14 oz.

Power Requirements

3.3 VDC Current	2.1A, max
5 VDC Current	2.9A, max

Calibration

Calibration Interval	1 year

Virtual Panel Description

The GX1838 includes a virtual panel program, which enables full utilization of the various configurations and controlling modes. To fully understand the front panel operation, it is best to become familiar with the functionality of the board.

To open the virtual panel application, select **GX1838 Panel** from the **Marvin Test Solutions**, **GXPDO** menu under the **Start** menu. The GX1838 virtual panel opens as shown here:

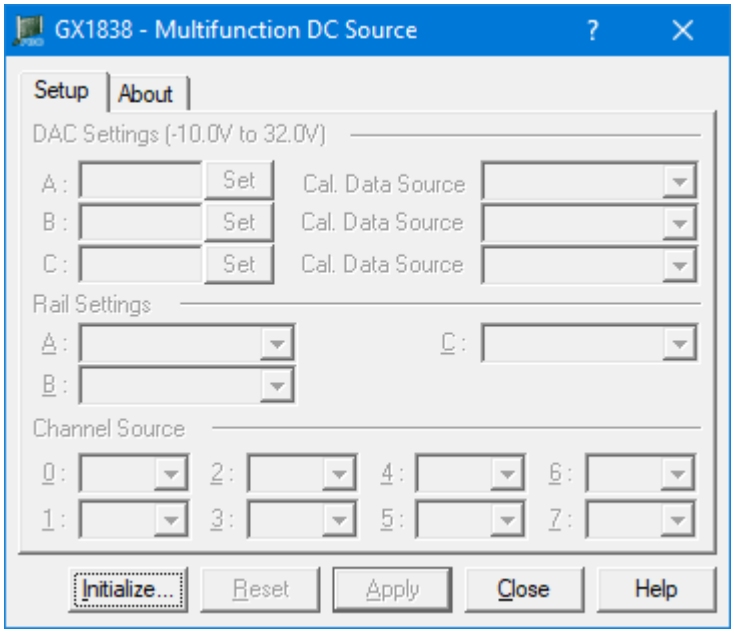


Figure 2-5: GX1838 Virtual Panel (not initialized)

The following controls are shown:

Initialize: Opens the Initialize Dialog (see Initialize Dialog paragraph) in order to initialize the board driver. The current settings of the selected board **will not change after calling initialize**. The panel will reflect the current settings of the board after the Initialize dialog closes.

Reset: Resets the board to its default state. The reset disconnects all rails from their sources, disconnect all channels from rails and sets all DACs to 0 volts.

Close: Closes (exits) the GX1838 panel.

Help: Opens the GX1838 on-line help window.

Virtual Panel Initialize Dialog

The Initialize dialog initializes the driver for the selected board. The board settings **will not change** after initialize is called. Once initialize, the panel will reflect the current settings of the board.

The Initialize dialog supports two different device drivers that can be used to access and control the board:

1. **Use HW/PXIExplorer** – this is the device driver installed by the setup program and is the default driver. When selected, the **Slot Number** list displays the available boards installed in the system and their slots. The chassis, slots, devices and their resources are also displayed by the HW resource manager, **PXI/PCI Explorer** applet that can be opened from the Windows Control Panel. The PXI/PCI Explorer can be used to configure the system chassis, controllers, slots and devices. The configuration is saved to PXISYS.INI and PXIeSYS.INI located in the Windows folder. These configuration files are also used by VISA. The following figure shows the slot number 0x107 (chassis 1 Slot 7). This is the slot number argument (*nSlot*) passed by the panel when calling the driver **Gx1838Initialize** function used to initialize driver with the specified board.

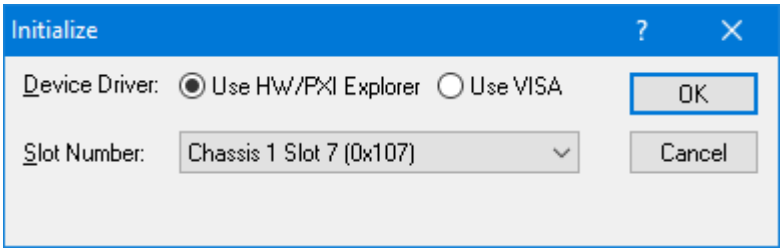


Figure 2-6: Initialize Dialog Box using Marvin Test Solutions' HW driver

2. **Use VISA** – this is a third-party device driver usually provided by National Instrument (NI-VISA). When selected, the **Resource** list displays the available boards installed in the system and their VISA resource address. The chassis, slots, devices and their resources are also displayed by the VISA resource manager, **Measurement & Automation** (NI-MAX) and in Marvin Test Solutions **PXI/PCI Explorer**. The following figure shows PXI5::10::INSTR as the VISA resource (PCI bus 5 and Device 10). This is VISA resource string argument (*szVisaResource*) passed by the panel when calling the driver **Gx1838InitializeVisa** function to initialize the driver with the specified board.

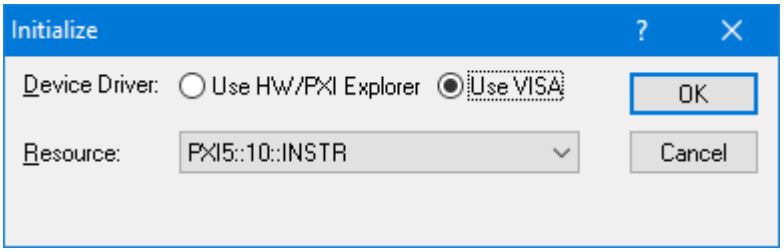


Figure 2-7: Initialize Dialog Box using VISA resources

Virtual Panel Setup Page

After the board is initialized the panel is enabled and will display the current setting of the board. The following picture shows the **Setup page** settings:

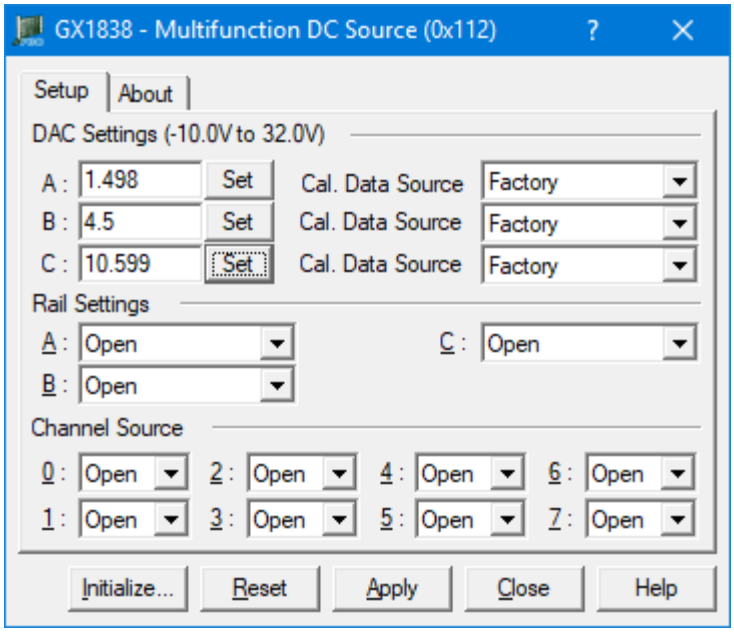


Figure 2-8: GX1838 Virtual Panel (Initialized)

The following controls are shown in the Setup page:

DAC Setting (Text Box A-C): Sets/display the correspondent DAC value. Each DAC is displayed with edit box for typing the required voltage value and the actual value is displayed in the next text box. Type in a new value in order to set channel's voltage and click on the **Set** button.

Cal Data Source (DAC A – C drop down list): Set/display the specified DAC Calibration data source.

Rail Settings (Rail A – C drop down list): Select/display rail source settings.

Channel Source Group (CH0-CH7 drop down list): Select/display the correspondent channel rail source settings.

Virtual Panel About Page

Clicking on the **About** tab will show the **About page** as shown in Figure 2-9:

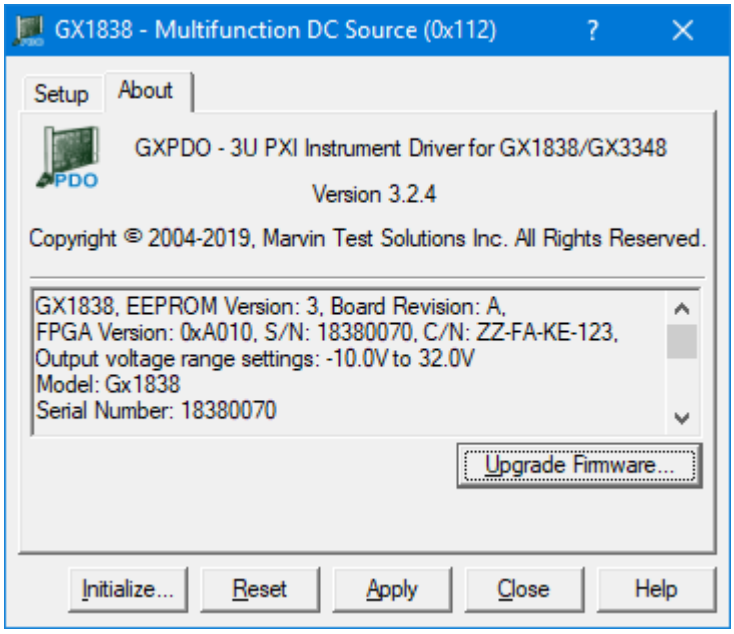


Figure 2-9:GX1838 Virtual Panel – About Page

The top part of the **About** page displays version and copyright of the GX1838 driver. The bottom part displays the board summary, including the EEPROM version, the board Revision, the FPGA version, the board serial number and the calibration time. The **About** page also contains a button **Upgrade Firmware…** used to upgrade the board FPGA. This button maybe used only when the board requires upgrade as directed by Marvin Test Solutions support. The upgrade requires a firmware file (.jam) that is written to the board FPGA. After the upgrade is complete you must shut down the computer to recycle power to the board.

Chapter 3 - Installation and Connections

Getting Started

This section includes general hardware installation procedures for the GX1838 board and installation instructions for the GX3348 - GXPDO software. Before proceeding, please refer to the appropriate chapter to become familiar with the board being installed.

To Find Information on…	Refer to…
GXPDO Software Installation	This Chapter
Hardware/Board Installation	This Chapter
Programming	Chapter 4
Software Function Reference	Chapter 5

Packing List

All GX1838 boards have the same basic packing list, which includes:

1. GX1838 Board
2. GXPDO Software CD

Unpacking and Inspection

After removing the board from the shipping carton:

 **Caution -** Static sensitive devices are present. Ground yourself to discharge static.

1. Remove the board from the static bag by handling only the metal portions.
2. Be sure to check the contents of the shipping carton to verify that all of the items found in it match the packing list.
3. Inspect the board for possible damage. If there is any sign of damage, return the board immediately. Please refer to the warranty information at the beginning of the manual.

System Requirements

All GX1838 instrument boards are designed to run on PXI compatible computer running 32/64 bit, LabView/Real-Time or Linux.

Each board requires one unoccupied 3U PXI bus slot. The board is a PXI hybrid slot compatible.

Installation of the GXPDO Software

This section describes installation for Windows, for LabView/Real-Time, see the ReadMe.txt file installed under the product main folder. For Linux installation, see the GtLinux software package ReadMe.txt file. Before installing the board, it is recommended to install the software as described in this section:

1. Insert the Marvin Test Solutions DVD and locate the **GXPDO.EXE** setup program. If your computer's Auto Run is configured, when inserting the DVD, a browser will show several options, select the Marvin Test Solutions Files option, then locate the setup file. If Auto Run is not configured, you can open the Windows explorer and locate the setup files (usually located under \Files\Setup folder). Alternatively, you can also download and install the latest GXPDO software package from Marvin Test Solutions web site (www.marvintest.com).
2. Run the setup and follow the instruction on the Setup screen to install the software.

   Note: You may be required to restart the setup after logging-in as a user with Administrator privileges. This is required in-order to upgrade your system with newer Windows components and to install the HW kernel-mode device drivers that are required by the GXPDO driver to access resources on your board.
3. The first setup screen to appear is the Welcome screen. Click **Next** to continue.
4. Enter the folder where the software is to be installed. Either click **Browse** to set up a new folder, or click **Next** to accept the default entry of `C:\Program Files\Marvin Test Solutions\GXPDO` or `C:\Program Files\MarvinTestSolutions (x86)\GXPDO` under x64 Windows.
5. Select the type of Setup you wish and click **Next.** You can choose between **Typical**, **Run-Time** and **Custom** setups. **Typical** setup type installs all files. **Run-Time** setup type will install only the files required for controlling the board either from its driver or from its virtual panel. **Custom** setup type lets you select from the available components.

The program will now start its installation. During the installation, Setup may upgrade some of the Windows shared components and files. The Setup may ask you to reboot after it complete if some of the components it replaced where used by another application during the installation – do so before attempting to use the software.

You can now continue with the installation to install the board. After the board installation is complete, you can test your installation by starting a panel program that let you control the board interactively. The panel program can be started by selecting it from the Start, Programs, GXPDO menu located in the Windows Taskbar.

Overview of the GXPDO Software

Once the software installed, the following tools and software components are available:

- **PXI/PCI Explorer applet** – use to configure the PXI chassis, controllers and devices. This is required for accurate identification of your PXI instruments later on when installed in your system. The applet configuration is saved to PXISYS.ini and PXIeSYS.ini that are used by Marvin Test Solutions instruments, the VISA provider and VISA based instruments drivers. In addition, the applet can be used to assign chassis numbers, Legacy Slot numbers and instruments alias names.
- **VISA** is a standard maintained by the VXI Plug & Play System Alliance and the PXI Systems Alliance organizations (http://www.ivifoundation.org/, http://www.pxisa.org/). VISA provides a standard way for instrument manufacturers and users to write and use instruments drivers. The VISA resource managers such as National Instruments **Measurement & Automation** (NI-MAX) can display and configure instruments and their address (similar to Marvin Test Solutions' PXI/PCI Explorer).
- **GXPDO Panel** – use to configure, control and display the board settings.
- **GXPDO driver** - A DLL based function library (32/64-bit versions of GXPDO.DLL, located in the Windows System folders) used to program and control the board. The driver uses Marvin Test Solutions' HW driver or VISA supplied by third party vendor to access and control the GXPDO board.
- **Programming files and examples** – interface files and libraries for various programming tools, see later in this chapter for a complete list of files, programming languages and development tools supported by the driver.
- **Documentation** – On-Line help (compiled HTML help file - chm file) and User's Guide (Adobe Acrobat - pdf).

Configuring Your PXI System using the PXI/PCI Explorer

To configure your PXI/PCI system using the **PXI/PCI Explorer** applet follow these steps:

1. **Start the PXI/PCI Explorer applet**. The applet can be start from the Windows Control Panel or from the Windows Start Menu, **Marvin Test Solutions**, **HW**, **PXI/PCI Explorer**.
2. **Identify Chassis and Controllers**. After the PXI/PCI Explorer started, it will scan your system for changes and will display the current configuration. The PXI/PCI Explorer automatically detects systems that have Marvin Test Solutions controllers and chassis. In addition, the applet detects PXI-MXI-3/4 extenders in your system (manufactured by National Instruments). If your chassis is not displayed in the explorer main window, use the Identify Chassis/Controller commands to identify your system. Chassis and Controller manufacturers should provide INI and driver files for their chassis and controllers to be used by these commands.
3. **Change chassis numbers, PXI devices Legacy Slot numbering and PXI devices Alias names.** These are optional steps to be performed if you would like your chassis to have different numbers. Legacy slots numbers are used by older Marvin Test Solutions or VISA drivers. Alias names can provide a way to address a PXI device using your logical name (e.g. "COUNTER1"). For more information regarding these numbers, see the **GxPdoInitialize** and **GxPdoInitializeVisa** functions.
4. **Save you work**. PXI Explorer saves the configuration to the following files located in the Windows folder: PXISYS.ini, PXIeSYS.ini and GxPxiSys.ini. Click on the **Save** button to save you changes. The PXI/Explorer prompt you to save the changes if changes were made or detected (an asterisk sign ' *' in the caption indicated changes).

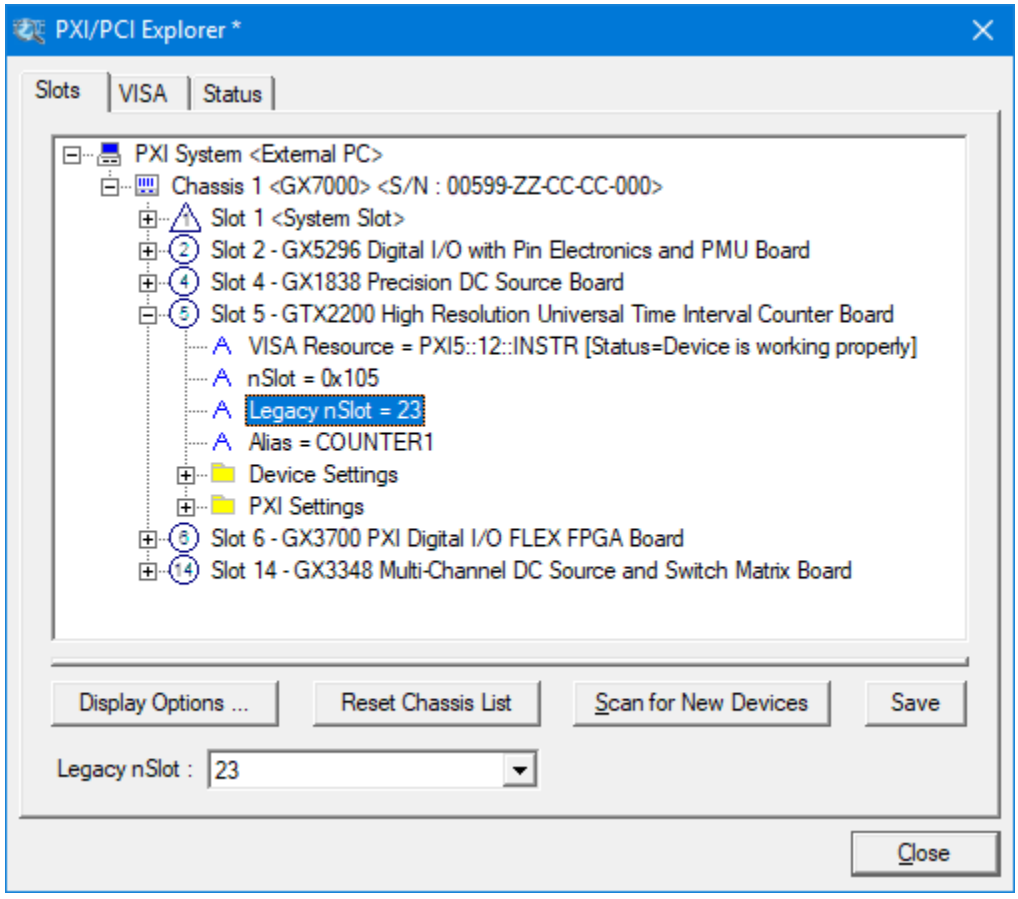


Figure 3-1: PXI/PCI Explorer

Board Installation

Before you Begin

- Install the software driver as described in the prior section.
- Configure your PXI/PC system using **PXI/PCI Explorer** as described in the prior section.
- Verify that all the components listed in the packing list (see previous paragraph) are present.

Electric Static Discharge (ESD) Precautions

To reduce the risk of damage to the board, the following precautions should be observed:

- Leave the board in the anti-static bags until installation requires removal. The anti-static bag protects the board from harmful static electricity.
- Save the anti-static bag in case the board is removed from the computer in the future.
- Carefully unpack and install the board. Do not drop or handle the board roughly.
- Handle the board by the edges. Avoid contact with any components on the circuit board.

Caution - Do not insert or remove any board while the computer is on. Turn off the power from the PXI chassis before installation.

Installing a Board

Install the board as follows:

1. Turn off the PXI chassis and unplug the power cord.
2. Locate a PXI empty slot on the PXI chassis.
3. Place the module edges into the PXI chassis rails (top and bottom).
4. Carefully slide the PXI board to the rear of the chassis, make sure that the ejector handles are pushed **out** (as shown in Figure 3-2).

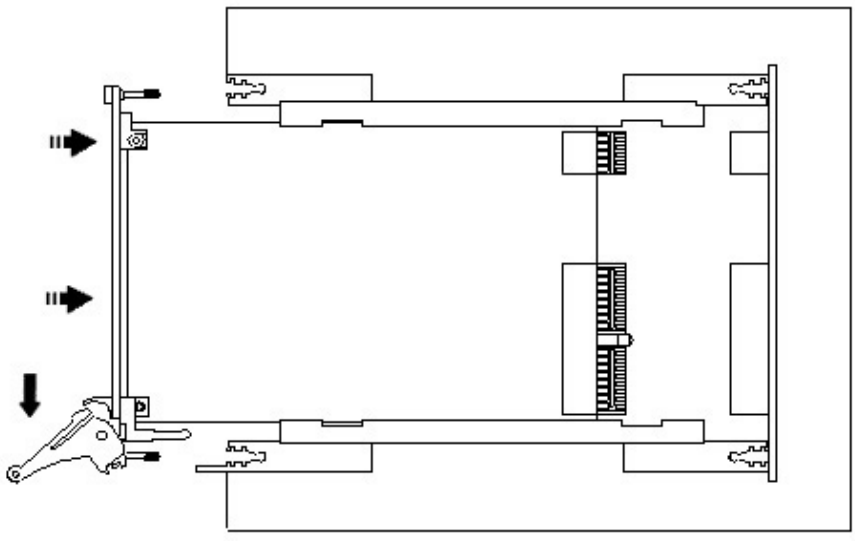


Figure 3-2: Ejector handles position during module insertion

5. After you feel resistance, push in the ejector handles as shown in Figure 3-3 to secure the module into the frame.

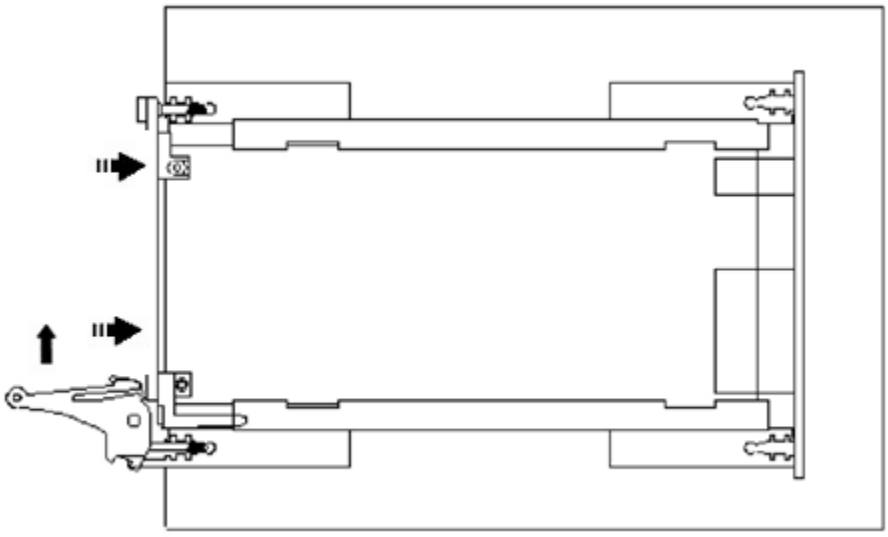


Figure 3-3: Ejector handles position after module insertion

6. Tighten the module's front panel to the chassis to secure the module in.
7. Connect any necessary cables to the board.
8. Plug the power cord in and turn on the PXI chassis.

Plug & Play Driver Installation

Plug & Play operating systems such as Windows notifies the user that a new board was found using the **New Hardware Found** wizard after restarting the system with the new board.

If another Marvin Test Solutions board software package was already installed, Windows will suggest using the driver information file: HW.INF. The file is located in your Program Files\Marvin Test Solutions\HW folder. Click **Next** to confirm and follow the instructions on the screen to complete the driver installation.

If the operating system was unable to find the driver (since the driver was not installed prior to the board installation), you may install the software as described in the prior section, then click on the **Have Disk** button and browse to select the HW.INF file located in C:\Program File\Marvin Test Solutions\HW.

If you are unable to locate the driver click **Cancel** to the found New Hardware wizard and exit the New Hardware Found Wizard, install the GXPDO driver, reboot your computer and repeat this procedure.

The Windows Device Manager (open from the System applet from the Windows Control Panel) must display the proper board name before continuing to use the board software (no Yellow warning icon shown next to device). If the device is displayed with an error, you can select it and press delete and then press F5 to rescan the system again and to start the New Hardware Found wizard.

Removing a Board

Remove the board as follows:

1. Turn off the PXI chassis and unplug the power cord.
2. Locate a PXI slot on the PXI chassis.
3. Disconnect and remove any cables/connectors connected to the board.
4. Un-tighten the module's front panel screws to the chassis.
5. Push out the ejector handles and slide the PXI board away from the chassis.
6. Optionally - uninstall the software by running the setup again (or from the Windows Control Pane, Programs and Features or Add Remove Programs applet) and selecting Remove/Uninstall.

Connectors and Accessories

The following accessories are available from Marvin Test Solutions for your switching board.

Part / Model Number	Description
GT25-CON	Connector, D-type 25 Male with solder pins.
GT25-MCON	Military Grade Connector, D-type 25 Male with solder pins.

Table 3-1: Spare Connectors and Accessories

Connectors

Figure 3-4: GX1838 J3 Output Channels Connector shows the available GX1838 board connector with description:

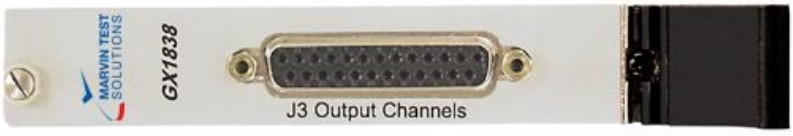


Figure 3-4: GX1838 J3 Output Channels Connector

Output Channels Connector(J3)

The following are the connector signal pin assignments for GX1838 module:

Pin#	Signal	Pin#	Signal	Pin#	Signal
1	Shield	10	Ext A	19	GND
2	I/O Ch0	11	Ext B	20	GND
3	I/O Ch1	12	Ext C	21	GND
4	I/O Ch2	13	Shield	22	GND
5	I/O Ch3	14	GND	23	GND
6	I/O Ch4	15	GND	24	GND
7	I/O Ch5	16	GND	25	GND
8	I/O Ch6	17	GND		
9	I/O Ch7	18	GND		

Table 3-2: J3 – Output Channels Connector

Legend

I/O Ch0 – Ch7: Channel input output.

Ext A - Ext C: Specified Rail External input.

GND: Ground.

Installation Folders

The GXPDO driver files are installed in the default directory `C:\Program Files\Marvin Test Solutions\GXPDO`. You can change the default `GXPDO` directory to one of your choosing at the time of installation.

During the installation, `GXPDO` Setup creates and copies files to the following directories:

Name	Purpose / Contents
…\Marvin Test Solutions\GXPDO	The GXPDO directory. Contains panel programs, programming libraries, interface files and examples, on-line help files and other documentation.
…\Marvin Test Solutions\HW	HW device driver. Provide access to your board hardware resources such as memory, IO ports and PCI board configuration. See the README.TXT located in this directory for more information.
…\ATEasy\Drivers	ATEasy drivers directory. GXPDO Driver and example are copied to this directory only if ***ATEasy*** is installed to your machine.
Windows System Folders	Windows System directories (32/64-bit versions of GXPDO.DLL driver, HW driver shared files and some upgraded system components, such as the HTML help viewer, etc.

GXPDO Driver Files Description

The Setup program copies the GXPDO driver, a panel executable, the GXPDO help file, the README.TXT file, and driver samples. The following is a brief description of each installation file:

Driver File and Virtual Panel

- GXPDO.DLL - 32/64-bit Windows DLLs for 32/64-bit applications running under Windows.
- GXPDOPANEL.EXE and GXPDOPANEL64.EXE – An instrument front panel program for all GXPDO supported boards.

Interface Files

The following GXPDO interface files are used to support the various development tools:

- GXPDO.H - header file for accessing the DLL functions using the C/C++ programming language. The header file compatible with the following 32-bit development tools:
  - Microsoft Visual C++, Microsoft Visual C++ .NET
  - Borland C++
- GXPDO.LIB and GXPDO64.LIB - Import library for 32 or 64-bit GXPDO.DLL (used when linking C/C++ application).
- GXPDOBC.LIB - Import library for GXPDO.DLL (used when linking C/C++ Builder application that uses GXPDO.DLL).
- GXPDO.PAS - interface file to support Pascal or Delphi.
- GXPDO.VB - Supports Microsoft Visual Basic .NET.
- GX1838.drv - ATEasy driver File for GX1838.
- GXPDO.llb – LabView library.

On-line Help and Manual

GXPDO.CHM – On-line version of the GXPDO User's Guide. The help file is provided in a Windows Compiled HTML help file (.CHM). The file contains information about the GX1838 board, programming reference and panel operation.

GXPDO.PDF – On line, printable version of the GXPDO User's Guide in Adobe Acrobat format. To view or print the file you must have the reader installed. If not, you can download the Adobe Acrobat reader (free) from http://www.adobe.com.

ReadMe File

README.TXT – Contains important last-minute information not available when the manual was printed. This text file covers topics such as a list of files required for installation, additional technical notes, and corrections to the GXPDO manuals. You can view and/or print this file using the Windows NOTEPAD.EXE or any other text file editors.

Example Programs

The sample program includes a C/C++ sample compiled with various development tools, Visual Basic example and an ATEasy sample. Other examples may be available for other programming tools.

Microsoft Visual C++ .NET example files:

- GXPDOExampleC.cpp - Source file
- GXPDOExampleC.ico - Icon file
- GXPDOExampleC.rc - Resource file
- GXPDOExampleC.vcproj - VC++ .NET project file
- GXPDOExampleC.exe - 32-bit example executable
- GXPDOExampleC64.exe - 64-bit example executable

Embarcadero/Borland C++ Builder example files:

- GXPDOExampleC.cpp - Source file
- GXPDOExample.ico - Icon file
- GXPDOExampleC.rc - Resource file
- GXPDOExampleC.bpr - Borland project file
- GXPDOExampleC.exe - Example executable

Microsoft Visual Basic .NET example files:

- GXPDOExampleVB.vb - Example form.
- GXPDOExampleVB.resx - Example form resource.
- GXPDOExampleVBapp.config - Example application configuration file.
- GXPDOExampleVBAssembleyInfo.vb - Example application assembly file
- GXPDOExampleVB.vbproj - Project file
- GXPDOExampleVB.exe - Example executable

Microsoft Visual C# .NET example files:

- GXPDOExampleCS.cs - Example form.
- GXPDOExampleCS.csproj - Project file
- GXPDOExampleCS.exe - Example executable

ATEasy driver and examples files (ATEasy Drivers directory):

- GX1838.prj - example project
- GX1838.sys - example system
- GX1838.prg - example program
- GX1838.prj - example project
- GX1838.sys - example system
- GX1838.prg - example program

Setup Maintenance Program

You can run the Setup again after GXPDO has been installed from the original disk or from the Windows Control Panel – Add Remove Programs applet. Setup will be in the Maintenance mode when running for the second time. The Maintenance window show below allows you to modify the current GXPDO installation. The following options are available in Maintenance mode:

- **Modify.** When you want to add or remove GXPDO components.
- **Repair.** When you have corrupted files and need to reinstall.
- **Remove.** When you want to completely remove GXPDO.

Select one of the options and click **Next**.

Follow the instruction on the screen until Setup is complete.

Chapter 4 - Programming the Board

This chapter contains information about how to program the GX1838 using the GXPDO driver. The driver contains functions to initialize, reset, and control the PXI board. This chapter includes a brief description of the functions, as well as how and when to use them, with code examples. Chapter 5 contains a complete and detailed description of the available programming functions.

The driver supports many development tools. This chapter describes using these tools with the driver. The GXPDO installation directory contains examples written for these development tools. Chapter 3 lists the available examples.

An example using the DLL driver with Microsoft Visual C++ is at the end of this chapter. The driver functions and parameters are identical for all operating systems and development tools, and this example serves as a guide for use other programming languages.

The GXPDO Driver

The driver is a 32 and 64-bit Windows DLL file: GXPDO.DLL. The DLLs are used with 32-bit or a 64-bit application running under Windows. The HW device driver is installed by the setup program and is shared by other Marvin Test Solutions products (ATEasy, GXPIO, GXSW, etc.). The DLLs can also use VISA (provided by a third party) to access the board hardware instead of the provided HW driver.

The DLL can be used with various development tools such as Visual C++, C++ Builder, Microsoft Visual Basic, Pascal or Delphi, ATEasy, LabView and more. The following paragraphs describe how to create an application that uses the driver with various development tools. Refer to the paragraph describing the specific development tool for more information.

Programming Using C/C++ Tools

The following steps are required to use the GXPDO driver with C/C++ development tools:

- Include the GXPDO.H header file in the C/C++ source file that uses the GX1838 function. The file contains function prototypes and constant declarations to be use by the C/C++ compilers for your application.
- Add the required .LIB file to the projects. This can be the import library GXPDO.LIB for 32-bit applications Microsoft Visual C+ (or other C/C++ development such as LabWindows/CVI) for 32-bit applications, GXPDO64.LIB for 64-bit applications. Use GXPDOBC.LIB for C++ Builder Windows-based. Applications that explicitly load the DLL by calling the Windows **LoadLibrary** API should not include the .LIB file in the project.
- Add code to call the Gx1838 as required by the application.
- Build the project.
- Run, test, and debug the application.

Programming Using Visual Basic

To use the driver with Visual Basic .NET developers must use the GXPDO.VB.

The file can be loaded within the integrated development environment (IDE) using *Add File* from the Visual Basic IDE *File menu*. The GXPDO.VB contains function declarations for the DLL driver.

Programming Using C#

To use the driver with Microsoft Visual c# .NET developers must use the GXPDO.cs.

The file can be loaded within the integrated development environment (IDE) using *Add File* from the Visual Basic IDE *File menu*. The GxPdo.cs contains function declarations for the DLL driver.

Programming Using Pascal/Delphi

To use the driver with Borland Pascal or Delphi, the user must include the GXPDO.PAS to the project. The GXPDO.PAS file contains a **unit** with function prototypes for the DLL functions. Include the GX1838 unit in the **uses** statement before making calls to the GX1838 functions.

Programming Using ATEasy®

The GXPDO package supplies a separate ATEasy driver for the PXI board. The ATEasy driver uses the same GXPDO.DLL to program the board for all development environments. An example project that contains a program and a system file pre-configured with the ATEasy driver is also provided. Use the property dialog to change the driver shortcut property in the System Drivers sub-module and change the PCI slot number to reflect your current installation before running the example.

The ATEasy driver plain-languages commands are easier to use than using the DLL functions directly. The driver commands will generate an exception if the function call fails. That allows the ATEasy application to trap errors without directly checking the status code returned by the DLL function. The driver commands check the status parameter after each function call.

The ATEasy driver contains commands that are similar to the DLL functions in name and parameters, with the following exceptions:

- The driver handles the *nHandle* parameter automatically. ATEasy handles board instances as driver logical names i.e. GXPDO1, GXPDO2 for GX1838. Direct access to the parameter is possible using the property operator (the "dot" operator) once the parameter is set public in the ATEasy driver.
- The driver handles the *nStatus* parameter automatically. The Get Status command in the ATEasy Driver provides access to the board's status. After calling a DLL function, the ATEasy driver will check the returned status and will call the CheckError procedure. An error status generates an exception that can be easily trapped by the application using the **OnError** module event or using the **try-catch** statement. ATEasy will notify the user or developer with an error dialog if the condition is not handled by other ATEasy code. Direct access to the parameter is possible using the property operator (the "dot" operator) once the parameter is set public in the ATEasy driver.

Some ATEasy drivers contain additional commands to permit easier access to the board features. For example, parameters for a function may be omitted by using a command item instead of typing the parameter value. The plain-language commands are self-documenting. Their syntax is similar to an English language statement. In addition, you can use the commands from the code editor context menu or by using the ATEasy's code completion feature instead of typing them directly.

Programming Using LabView and LabView/Real Time

To use the driver with LabView use the provided lab view library GXPDO.llb. The library is located in the GXPDO folder. An example for LabView is also provided in the Examples folder. A DLL located in the LabViewRT folder can be used for deployment with LabView/Real-Time.

Using and Programming under Linux

Marvin Test Solutions provides a separate software package with a Linux driver (Marvin Test Solutions Drivers Pack for Linux). The software package can be download from the Marvin Test Solutions website. See the ReadMe.txt in that package for more information regarding using and programming the driver under Linux.

Using the GXPDO driver functions

The GXPDO driver contains a set of functions for the GX1838. Functions names that starts with the **GXPDO** prefix applies to all GXPDO boards (i.e. **Gx1838Reset**). The GXPDO functions are designed with consistent set of arguments and functionality. All boards have a function that initializes the GXPDO driver for a specific board, reset the board, and display the virtual panel. All the functions use handles to identify and reference a specific board and all functions return status and share the same functions to handle error codes.

Initialization, HW Slot Numbers and VISA Resource

The GXPDO driver supports two device drivers HW and VISA, which are used to initialize, identify and control the board. The user can use the **Gx1838Initialize** to initialize the board's driver using HW and **Gx1838InitializeVisa** to initialize using VISA. The following describes the two different methods used:

1. **Marvin Test Solutions' HW** - the default device driver that is installed by the GXPDO driver. To initialize and control the board using the HW use the **Gx1838Initialize** (*nSlot, pnHandle, pnStatus*) function. The function initializes the driver for the board at the specified PXI slot number (*nSlot*) and returns a board handle. The **PXI/PCI Explorer** applet in the Windows Control Panel displays the PXI slot assignments. You can specify the *nSlot* parameter in the following way:

   - A combination of chassis number (chassis # x 256) with the chassis slot number, e.g. 0x105 for chassis 1 and slot 5. Chassis number can be set by the **PXI/PCI Explorer** applet.
   - Legacy nSlot as used by earlier versions of HW/VISA. The slot number contains no chassis number and can be changed using the **PXI/PCI Explorer** applet: 23 in this example.

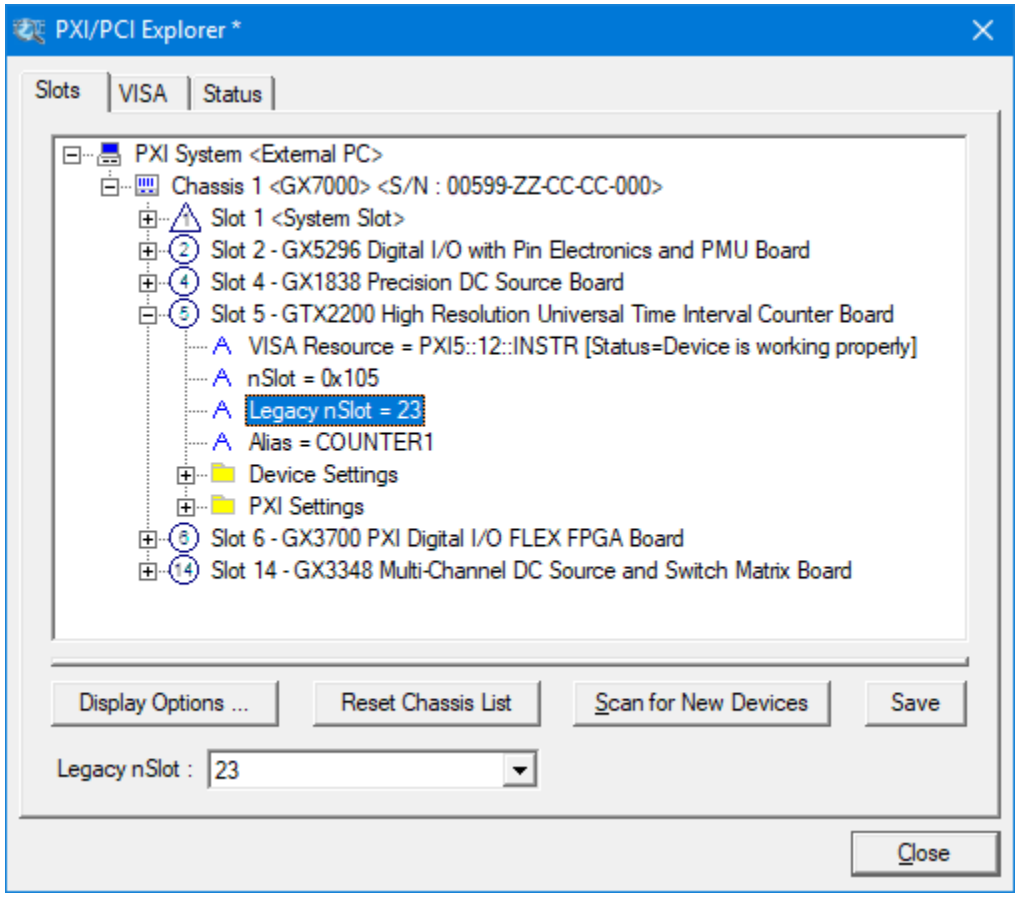


Figure 4-1: PXI/PCI Explorer

2. **VISA** – this is a third-party library usually by National Instruments (NI-VISA). You must ensure that the VISA installed supports PXI and PCI devices (not all VISA providers supports PXI/PCI). GXPDO setup installs a VISA compatible driver for the GXPDO board in-order to be recognized by the VISA provider. Use the GXPDO function **Gx1838InitializeVisa** (*szVisaResource, pnHandle, pnStatus*) to initialize the driver board using VISA. The first argument *szVisaResource* is a string that is displayed by the VISA resource manager such as NI **Measurement and Automation** (NI_MAX). It is also displayed by Marvin Test Solutions **PXI/PCI Explorer** as shown in the prior figure. The VISA resource string can be specified in several ways as the following examples:

   - Using chassis, slot: "PXI0::CHASSIS1::SLOT5"
   - Using the PCI Bus/Device combination: "PXI9::13::INSTR" (bus 9, device 9).
   - Using alias: "COUNTER1". Use the PXI/PCI Explorer to set the device alias.

   Information about VISA is available at http://www.pxisa.org.

The **Gx1838Initialize** function returns a handle that is required with other driver functions to program the board. This handle is usually saved in the program in a global variable for later use when calling other functions. The initialize function does not change the state of the board or its settings.

Board Handle

The board handle argument, *nHandle* , passed (by reference) to the parameter *pnHandle* of the **Gx1838Initialize** or the **Gx1838InitializeVisa** functions is a short integer (16 bits) number. It is used by the GXPDO driver functions to identify the board being accessed by the application. Since the driver supports many boards at the same time, the *nHandle* argument is required to uniquely identify which board is being programmed.

The *nHandle* is created when the application calls the **Gx1838Initialize** function. There is no need to destroy the handle. Calling **Gx1838Initialize** with the same slot number will return the same handle.

Once the board is initialized the handle can be used with other functions to program the board.

Reset

The Reset function **Gx1838Reset**(*nHandle, nStatus*), causes the driver to change all settings to their default state. See the Function Reference for more information regarding the specific board.

Error Handling

All GXPDO functions pass a fail or success status - *pnStatus* - in the last parameter. A successful function call passes zero in the status parameter upon return. If the status is non-zero, then the function call fails. This parameter can be later used for error handling. When the status is error, the program can call the **GxPdoGetErrorString** function to return a string representing the error. The **GxPdoGetErrorString** reference contains possible error numbers and their associated error strings.

Driver Version

The **GxPdoGetDriverSummary** function can be used to return the current GXPDO driver version. It can be used to differentiate between the driver versions. See the Function Reference for more information.

Panel

Calling the **Gx1838Panel** will display the instrument's front panel dialog window. The panel can be used to initialize and control the board interactively. The panel function may be used by the application to allow the user to directly interact with the board.

The **Gx1838Panel** function is also used by the GXPDOPANEL.EXE. or the GXPDOPANEL64.EXE applications that are supplied with this package and provides a stand-alone Windows application that displays the instrument panel.

Distributing the Driver

Once the application is developed, the driver files (32 or 64-bit version of GXPDO.DLL and the HW device driver files located in the HW folder) can be shipped with the application. Typically, the DLLs should be copied to the Windows System directory. The HW device driver files should be installed using a special setup program HWSETUP.EXE that is provided with GXPDO driver files. Alternatively, you can provide the GXPDO disk to be installed along with the board.

Sample Programs

The following example demonstrates how to program the board using the C programming language under Windows. The example shows how to apply certain voltage to a specified rail.

To run, Enter the following command line:

GxPdoExample <*PciSlot*> <*operation*> <*channel* | *dac* | *rail*> <*rail_number* | *voltage* | *rail_source*>

Where:

<*PciSlot*>	PCI/PXI Explorer slot number where the board reside.
<*operation*>	Operation code: SCR=Set Channel Rail GCR=Get Channel Rail SDV=Set Dac Voltage GDV=Get Dac Voltage SRS=Set Rail Source
< *channel*	*dac*	*rail* >	Depends on the operation: Channel number: 0-7 for SCR/GCR operations DAC number: 0-2 for SDV/GDV operations Rail number: 0-2 for SRS/GRS operations
<*rail_numbe*|*voltage*|*rail_source*>	Depends on operation: Rail_number: 0 to 2 Voltage: -10V to 32V Rail_source: 0 to 4

Sample Program Listing

```
/*********************************************************************
      FILE        : GxPdoExampleC.cpp

      PURPOSE     : WIN32/LINUX example program for GX1838/GX3348
                    boards using the GXPDO driver.

      CREATED     : May 2002

      COPYRIGHT   : Copyright 2002-2016 MARVIN TEST SOLUTIONS - MTS Inc.

      COMMENTS    :

      To compile the example:

      1. Microsoft VC++
            Load GxPdoExampleC.dsp, .vcproj or .mak, depends on
            the VC++ version from the Project\File/Open... menu
            Select Project/Rebuild all from the menu

      2. Borland C++ Builder
            Load GxPdoExampleC.bpr from the Project/Open
                  Project... menu
            Select Project/Build all from the menu

      3. Linux (GCC for CPP and Make must be available)
            make -fGxPdoExampleC.mk [CFG=Release[64] | Debug[64]] [rebuild |
            clean]

*********************************************************************/
#ifndef __GNUC__
#include "windows.h"
#endif
#include "GxPdo.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#if defined(__BORLANDC__)
#pragma hdrstop
#include <condefs.h>
USELIB("GxPdoBC.lib");
USERC("GxPdoExampleC.rc");
#endif

//*********************************************************************
//          DisplayMsg
//*********************************************************************
void DisplayMsg(PCSTR lpszMsg)
{
#ifndef __GNUC__
      MessageBeep(0);
      MessageBox(0, lpszMsg, "GxPdo example program", MB_OK);
#else
```

```
    printf("\r\nGxPdo example program: %s\r\n", lpszMsg);
#endif
      return;
}

//*********************************************************************
//          __strupr
//*********************************************************************
char * __strupr(char * sz)
{
    int i;

    for (i=0; sz[i]; i++)
        sz[i]=toupper(sz[i]);
    return sz;
}

//*********************************************************************
//          DisplayUsage
//*********************************************************************
void DisplayUsage(void)
{
      DisplayMsg(
            "\r\nThis example shows how to use the GX1838/Gx3348:\r\n\r\n"

            "Usage:\r\n"
            "For 1838:\r\n"
            "GxPdoExampleC 1838 <slot_number> <operation> <channel|dac|"
                  "rail> <rail_number|voltage|rail_source>"
            "\r\n\r\nWhere : \r\n"
            "<model> - Model of board, 1838\r\n"
            "<slot_number> - PCI/PXI slot number as shown by the PXI "
                  "explorer\r\n"
            "<operation> - one of the followings :\r\n"
            "\tSCR=Set Channel Rail\r\n"
            "\tGCR=Get Channel Rail\r\n"
            "\tSDV=Set DAC Voltage\r\n"
            "\tGDV=Get DAC Voltage\r\n"
            "\tSRS=Set Rail Source\r\n"
            "\tGRS=Get Rail Source\r\n\r\n"
            "<channel|dac|rail numbers> - depends on the operation:\r\n"
            "\tchannel number:\t0-7 for SCR/GCR operations\r\n"
            "\tdac number :   \t0-2 for SDV/GDV operations\r\n"
            "\trail number :   \t0-2 for SRS/GRS operations\r\n\r\n"
            "For 3348:\r\n"
            "GxPdoExampleC 3348 <slot_number> <operation> <group> <channel>"
                  "<dac> <state|voltage>\r\n"
            "\r\n\r\nWhere : \r\n"
            "<model> - Model of board, 3348\r\n"
            "<slot_number> - PCI/PXI slot number as shown by the PXI "
                  "explorer\r\n"
            "<operation> - one of the followings :\r\n"
            "\tSDV=Set DAC Voltage\r\n"
            "\tGDV=Get DAC Voltage\r\n"
            "\tSRE=Set Relay State (Open or Close)\r\n"
            "\tGRE=Get Relay State (Open or Close)\r\n"
            "\tMES=Measure channel voltage\r\n"
```

```
            "<group> - channel group to set or measure (0-2)\r\n"
            "<channel> - channel to set or measure within a group(0-15)\r\n"
            "<dac> - dac to set (0-2)\r\n"
            "<state|voltage> - depends on the operation:\r\n"
            "\tSDV : 0-3    \tDAC Voltage (-20 to 30V)\r\n"
            "\tSRE : 0-3    \tRelay state, 1 for close and 0 for open\r\n"
            "\r\nTo change command line under Windows:\r\n"
            "\tRight click on the example shortcut from the start menu\r\n"
            "\tand type the new command line\r\n"
            );
      exit(1);
}

//*****************************************************************
//          CheckStatus
//*****************************************************************
void CheckStatus(SHORT nStatus)
{
    CHAR    sz[512];

      if (!nStatus) return;
      GxPdoGetErrorString(nStatus, sz, sizeof sz, &nStatus);
      DisplayMsg(sz);
      DisplayMsg("Aborting the program...");
      exit(nStatus);
}

//*****************************************************************
//          MAIN
//
// This main function receives five parameters
//
// GXPDO model (e.g. 1838 or 3348)
// GXPDO board slot number (e.g. 1)
// GX1838 operations (e.g. SCR=Set Channel Rail,
//                                  GCR=Get Channel Rail,
//                                    SDV=Set Dac Voltage,
//                                    GDV=Get Dac Voltage,
//                                    SRS=Set Rail Source,
//                                    GRS=Get Rail Source
// GX1838 channel dac rail number depends on operation
//                                    0 to 7 for channel
//                                    0 to 2 for dac
//                                    0 to 2 for rail
// rail_number voltage rail_source depends on operation
//                                    0 to 2 for rail_number
//                                    -10V to 32V for voltage
//                                    0 to 4 for rail_source
// GX3348 operations (e.g. SDV=Set Dac Voltage,
//                                  SDV=Get Dac Voltage,
//                                    SRE=Set Relay state,
//                                    GRE=Get Relay state,
//                                    MES=Measure channel voltage
// GX3348 group, channel, rail
//                                    0 to 3 for group
//                                    0 to 15 for channel
//                                    0 to 2 for dac
```

```
//*********************************************************************
int main(int argc, char **argv)
{
	short nBoardType;			// Board model
	short nSlotNum;				// Board slot number
	char* sOperation;			// Board Operation
	short nChannelOrDacOrRail;// Channel or DAC or Rail number
	short nHandle;				// Board handle
	short nStatus;				// Returned status
	short nRailOrRailSource;	// Rail number or rail source
	double	dVoltage;				// Dac voltage
	short nGroup;					// Group number
	short nChannel;				// Channel number
	short nDac;					// Dac number
	short nState;					// State of channel relay
	BOOL  bState;					// State of channel relay

	// Check number of arguments recived
	if (argc<4) DisplayUsage();

	// Parse command line parameters
	nBoardType=(SHORT)strtol(*(++argv), NULL, 0);
	nSlotNum=(SHORT)strtol(*(++argv), NULL, 0);
	sOperation=__strupr(*(++argv));

	if (nSlotNum<0)
		DisplayUsage();

	switch(nBoardType)
	{
		case 1838:
			nChannelOrDacOrRail=(SHORT)strtol(*(++argv), NULL, 0);
			Gx1838Initialize(nSlotNum, &nHandle, &nStatus);
			CheckStatus(nStatus);

			if(!strcmp(sOperation, "SCR"))
			{	// set channel rail
				if (argc<5) DisplayUsage();
				nRailOrRailSource=(SHORT)strtol(*(++argv), NULL, 0);
				Gx1838SetChannelRail(nHandle, nChannelOrDacOrRail,
					nRailOrRailSource, &nStatus);
				CheckStatus(nStatus);
				printf("Set Channel %i rail to %i.\n",
					nChannelOrDacOrRail, nRailOrRailSource);
			}
			else if(!strcmp(sOperation, "GCR"))
			{	// get channel rail
				Gx1838GetChannelRail(nHandle, nChannelOrDacOrRail,
					&nRailOrRailSource, &nStatus);
				CheckStatus(nStatus);
				printf("Channel %i rail is %i.\n",
					nChannelOrDacOrRail, nRailOrRailSource);
			}
			else if(!strcmp(sOperation, "SDV"))
			{	// set dac voltage
				if (argc<5) DisplayUsage();
				dVoltage=strtod(*(++argv), NULL);
```

```
                        Gx1838SetDacVoltage(nHandle, nChannelOrDacOrRail,
                              dVoltage, &nStatus);
                        CheckStatus(nStatus);
                        printf("Set DAC %i voltage to %f.\n",
                              nChannelOrDacOrRail, dVoltage);
                  }
                  else if(!strcmp(sOperation, "GDV"))
                  {     // get dac voltage
                        Gx1838GetDacVoltage(nHandle,
                              nChannelOrDacOrRail, &dVoltage, &nStatus);
                        CheckStatus(nStatus);
                        printf("Dac %i voltage is %f.\n",
                              nChannelOrDacOrRail, dVoltage);
                  }
                  else if(!strcmp(sOperation, "SRS"))
                  {     // set rail source
                        if (argc<5) DisplayUsage();
                        nRailOrRailSource=(SHORT)strtol(*(++argv), NULL, 0);
                        Gx1838SetRailSource(nHandle, nChannelOrDacOrRail,
                              nRailOrRailSource, &nStatus);
                        CheckStatus(nStatus);
                        printf("Set rail %i source to %i.\n",
                              nChannelOrDacOrRail, nRailOrRailSource);
                  }
                  else if(!strcmp(sOperation, "GRS"))
                  {     // get rail source
                        Gx1838GetRailSource(nHandle, nChannelOrDacOrRail,
                              &nRailOrRailSource, &nStatus);
                        CheckStatus(nStatus);
                        printf("Rail %i source is %i.\n",
                              nChannelOrDacOrRail, nRailOrRailSource);
                  }
                  break;
            case 3348:
                  Gx3348Initialize(nSlotNum, &nHandle, &nStatus);
                  CheckStatus(nStatus);
                  if(!strcmp(sOperation, "SDV"))
                  {
                        if (argc<6) DisplayUsage();
                        //Set Dac Voltage
                        nGroup=(SHORT)strtol(*(++argv), NULL, 0);
                        nChannel=(SHORT)strtol(*(++argv), NULL, 0);
                        nDac=(SHORT)strtol(*(++argv), NULL, 0);
                        dVoltage=strtod(*(++argv), NULL);

                        Gx3348SetDacVoltage(nHandle, nDac, dVoltage, false,
                              &nStatus);
                        CheckStatus(nStatus);
                  }
                  else if(!strcmp(sOperation, "GDV"))
                  {
                        if (argc<5) DisplayUsage();
                        //Get Dac Voltage
                        nGroup=(SHORT)strtol(*(++argv), NULL, 0);
                        nChannel=(SHORT)strtol(*(++argv), NULL, 0);
                        nDac=(SHORT)strtol(*(++argv), NULL, 0);
```

```
                        Gx3348GetDacVoltage(nHandle, nDac, &dVoltage, false,
                              &nStatus);
                        CheckStatus(nStatus);
                        printf("DAC %i voltage is %f.\n", nDac, dVoltage);
                  }
                  else if(!strcmp(sOperation, "SRE"))
                  {
                        if (argc<6) DisplayUsage();
                        //Set relay state
                        nGroup=(SHORT)strtol(*(++argv), NULL, 0);
                        nChannel=(SHORT)strtol(*(++argv), NULL, 0);
                        nDac=(SHORT)strtol(*(++argv), NULL, 0);
                        nState=(SHORT)strtol(*(++argv), NULL, 0);

                        Gx3348SetRelay(nHandle, nDac, nGroup, nChannel,
                              nState==1, &nStatus);
                        CheckStatus(nStatus);
                  }
                  else if(!strcmp(sOperation, "GRE"))
                  {
                        if (argc<6) DisplayUsage();
                        //Get relay state
                        nGroup=(SHORT)strtol(*(++argv), NULL, 0);
                        nChannel=(SHORT)strtol(*(++argv), NULL, 0);
                        nDac=(SHORT)strtol(*(++argv), NULL, 0);
                        nState=(SHORT)strtol(*(++argv), NULL, 0);
                        Gx3348GetRelay(nHandle, nDac, nGroup, nChannel,
                              &bState, &nStatus);
                        CheckStatus(nStatus);

                        if(bState)
                              printf("%d Channel in Group %d is connected "
                                    "to DAC %d", nChannel, nGroup, nDac);
                        else
                              printf("%d Channel in Group %d is "
                                    "disconnected to DAC %d", nChannel,
                                    nGroup, nDac);
                  }
                  else if(!strcmp(sOperation, "MES"))
                  {
                        if (argc<6) DisplayUsage();
                        //Measure channel voltage
                        nGroup=(SHORT)strtol(*(++argv), NULL, 0);
                        nChannel=(SHORT)strtol(*(++argv), NULL, 0);
                        Gx3348Measure(nHandle, nGroup, nChannel, &dVoltage,
                              &nStatus);
                        CheckStatus(nStatus);
                        printf("Measured voltage on Channel %d in Group %d "
                              "is %fV.\n", nChannel, nGroup, dVoltage);
                  }
                  break;
            default:
                  DisplayUsage();
      }

      return 0;
}
```

```
//******************************************************************
//          End Of File
//******************************************************************
```

Chapter 5 - Functions Reference

Introduction

The GX1838 driver functions reference chapter is organized in alphabetical order. Each function description contains the function name; purpose, syntax, parameters description and type followed by Comments, an Example (written in C), and a See Also sections.

All function parameters syntax follows the same rules:

- Strings are ASCIIZ (null or zero character terminated).
- The first parameter of most functions is *nHandle* (16-bit integer). This parameter is required for operating the board and is returned by the **Gx1838Initialize** function. The *nHandle* is used to identify the board when calling a function for programming and controlling the operation of that board.
- All functions return a status with the last parameter named *pnStatus*. The *pnStatus* is zero if the function was successful, or less than a zero on error. The description of the error is available using the **GxPdoGetErrorString** function or by using a predefined constant, defined in the driver interface files: GXPDO.H, GXPDO.BAS, GXPDO.VB, GXPDO.PAS or GX1838.DRV.
- Parameter name are prefixed as follows:

Prefix	Type	Example
a	Array - prefix this before the simple type.	*anArray* (Array of Short)
b	BOOL – Boolean, 0 for FALSE; <>0 for TRUE	*bUpdate*
d	DOUBLE - 8 bytes floating point	*dReading*
dw	DWORD - double word (unsigned 32-bit)	*dwTimeout*
hwnd	Window handle (32-bit integer).	*hwndPanel*
l	LONG - (signed 32-bit)	*lBits*
n	SHORT - (signed 16-bit)	*nMode*
p	Pointer - Usually used to return a value. Prefix this before the simple type.	*pnStatus*
sz	Null - (zero value character) terminated string	*szMsg*
uc	BYTE - (8 bits) unsigned.	*ucValue*
w	WORD - Unsigned short (unsigned 16-bit)	*wParam*

Table 5-1: Parameter Name Prefixes

GX1838 Functions

The following list is a summary of functions available for the GX1838:

Driver Functions	Description
General	
Gx1838Initialize	Initializes the GX1838 driver for the specified slot.
Gx1838InitializeVisa	Initializes the GX1838 driver for the Visa resource
Gx1838Panel	Opens a virtual panel used to interactively control the GX1838.
Gx1838Reset	Resets the GX1838 board to its default settings.
GxPdoGetDriverSummary	Returns the driver name and version.
GxPdoGetErrorString	Returns the error string associated with the specified error number.
Functions	
Gx1838GetBoardSummary	Returns the Gx1838 version, serial number and calibration date.
Gx1838GetChannelRail	Returns the specified channel rail connection.
Gx1838GetDacCalibrationDataSource	Returns the specified DAC calibration data source currently in use.
Gx1838GetDacCalibrationDataSources	Returns the specified DAC calibration data sources available to be used.
Gx1838GetDacCalibrationDataString	Returns the specified DAC calibration source data string.
Gx1838GetDacVoltage	Returns the specified DAC voltage.
Gx1838GetRailSource	Returns the specified rail source
Gx1838SetChannelRail	Sets the specified channel to the specified rail.
Gx1838SetDacCalibrationDataSource	Returns the specified DAC calibration data source currently in use.
Gx1838SetDacVoltage	Sets the specified DAC voltage.
Gx1838SetRailSource	Set the specified rail source.
Calibration	
Gx1838CalByUserDeleteClibration	Delete the specified calibration data source.
Gx1838CalByUserReadMeasuredVoltage	Returns the measured value of the specified voltage level used in the calibration process for the DAC currently under calibration.
Gx1838CalByUserSetup	Initiates a new calibration process and sets up the specified DAC for calibration.
Gx1838CalByUserSetupDacFor Measurement	Sets the DAC currently under calibration to the specified min/max voltage level.
Gx1838CalByUserStoreDacCalibration Data	Stores the calibration data of the DAC currently under calibration.
Gx1838CalByUserWriteMeasuredVoltage	Writes the measured value by the user for the specified voltage level used in the calibration process for the DAC currently under calibration.

Gx1838CalByUserDeleteCalibration

Purpose

Delete the specified calibration data source.

Syntax**Gx1838CalByUserDeleteClibration** (*nHandle, nDac, nCalDataSource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nDac	SHORT	Specified DAC number to be calibrated. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
nCalDataSource	SHORT	Calibration data source are as follow: 2. GX1838_CAL_DATA_USER_NONVOLATILE 4. GX1838_CAL_DATA_USER_VOLATILE
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calling this function will delete the specified calibration data source and its data.

Calibration data sources:

- GX1838_CAL_DATA_USER_NONVOLATILE: Exists only if the user calibrated the specified DAC using the **Calibration By User** set of functions. This calibration is a nonvolatile type (save to the on-board EEPROM).
- GX1838_CAL_DATA_USER_VOLATILE: Exists only if the user calibrated the specified DAC using the **Calibration By User** set of functions. This calibration data source is a volatile type (not saved to the board). The calibration data is valid only if the current application (ATEasy or executable) used for calibration was not terminated.

Example

The following example deletes the calibration by user nonvolatile data from Dac B:

```
SHORT   nStatus;

Gx1838CalByUserDeleteClibration (nHandle, GX1838_DAC_B, GX1838_CAL_DATA_USER_NONVOLATILE,
&nStatus);
```

See Also

Gx1838CalByUserSetupDacForMeasurement, Gx1838CalByUserWriteMeasuredVoltage, Gx1838CalByUserStoreDacCalibrationData, Gx1838SetDacCalibrationDataSource, Gx1838GetDacCalibrationDataSources, GxPdoGetErrorString

Gx1838CalByUserReadMeasuredVoltage

Purpose

Returns the measured value of the specified voltage level used in the calibration process for the DAC currently under calibration.

Syntax

Gx1838CalByUserReadMeasuredVoltage (*nHandle, nVoltageLevel, pdVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nVoltageLevel	SHORT	Specified voltage level: 0. GX1838_CAL_LOW_VOLTAGE 1. GX1838_CAL_HIGH_VOLTAGE
pdVoltage	PDOUBLE	Returned value that was supplied and measured by the user for the specified voltage level used in the calibration process.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function is valid only if calibration is in process.

The returned value is the voltage that was measured by the user for the specified calibration voltage level and written to the driver using the **Gx1838CalByUserWriteMeasuredVoltage** function. The voltage always refers to the currently calibrated DAC.

The function will return an error if currently not in calibration process.

Example

The following example returns the measured low voltage level used in the calibration process:

```
SHORT   nStatus;
DOUBLE  dVoltage;

Gx1838CalByUserReadMeasuredVoltage (nHandle, GX1838_CAL_LOW_VOLTAGE, &dVoltage, &nStatus);
```

See Also

Gx1838CalByUserSetupDacForMeasurement, Gx1838CalByUserWriteMeasuredVoltage, Gx1838CalByUserStoreDacCalibrationData, Gx1838SetDacCalibrationDataSource, Gx1838GetDacCalibrationDataSources, GxPdoGetErrorString

Gx1838CalByUserSetup

Purpose

Initiates a new calibration process and sets up the specified DAC for calibration.

Syntax

Gx1838CalByUserSetup (*nHandle, nDac, nCalDataTarget, nChannel, dCalLoadVal, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nDac	SHORT	Specified DAC number to be calibrated. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
nCalDataTarget	SHORT	Specified calibration data target to be applied to as follow: 2. GX1838_CAL_DATA_USER_NONVOLATILE 4. GX1838_CAL_DATA_USER_VOLATILE
nChannel	SHORT	Specified output channel number 0-7where the calibration load is connected.
dCalLoadVal	DOUBLE	Calibration load value in ohms. Value is measured by the user, the minimum can be 64Ω.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calling this function will imitate a calibration process and set a path between the calibrated DAC and the load connected on the output connector (J3). The specified calibration data target will be applied to either the dynamic calibration that is not stored on the on-board EEPROM or user calibration that at the end of the calibration process will be stored to the on-board EEPROM for the specified DAC:

Following this function, the user must call the **Gx1838CalByUserSetupDacForMeasurement** function in order to follow the calibration sequence.

Example

The following example sets up DAC B for calibration using a 200Ω load that is connected to channel 3, the calibration type is User Calibration (stored to the on-board EEPROM):

```
SHORT   nStatus;

Gx1838CalByUserSetup (nHandle, GX1838_DAC_B, GX1838_CAL_DATA_USER_NONVOLATILE, 3, 200, nStatus);
```

See Also

Gx1838CalByUserSetupDacForMeasurement, Gx1838CalByUserWriteMeasuredVoltage, Gx1838CalByUserReadMeasuredVoltage, Gx1838CalByUserStoreDacCalibrationData, Gx1838SetDacCalibrationDataSource, Gx1838GetDacCalibrationDataSources, GxPdoGetErrorString

Gx1838CalByUserSetupDacForMeasurement

Purpose

Sets the DAC currently under calibration to the specified min/max voltage level.

Syntax

Gx1838CalByUserSetupDacForMeasurement (*nHandle, nVoltageLevel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nVoltageLevel	SHORT	Specified voltage level: 0. GX1838_CAL_LOW_VOLTAGE 1. GX1838_CAL_HIGH_VOLTAGE
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function sets the DAC currently under calibration to either low voltage (GX1838_CAL_LOW_VOLTAGE) or high voltage (GX1838_CAL_HIGH_VOLTAGE). Following this function, the user needs to measure the voltage across the output load, as was set in the **Gx1838CalByUserSetup** function and write the value to the driver using **Gx1838CalByUserWriteMeasuredVoltage** function.

Example

The following example sets up the internal DAC currently under calibration for measuring its high voltage:

```
SHORT   nStatus;

Gx1838CalByUserSetupDacForMeasurement (nHandle, GX1838_CAL_HIGH_VOLTAGE, &nStatus);
```

See Also

Gx1838CalByUserWriteMeasuredVoltage, Gx1838CalByUserReadMeasuredVoltage, Gx1838CalByUserStoreDacCalibrationData, Gx1838SetDacCalibrationDataSource, Gx1838GetDacCalibrationDataSources, GxPdoGetErrorString

Gx1838CalByUserStoreDacCalibrationData

Purpose

Stores the calibration data of the DAC currently under calibration.

Syntax

Gx1838CalByUserStoreDacCalibrationData (*nHandle, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function concludes the calibration process for the DAC currently under calibration. The function analyzes the data previously gathered, zero voltage and positive full-scale voltage in relationship to the specified load and to the board's specifications. If the analysis succeeds the calibration data is then stored to the calibration data source as was specified in the **Gx1838CalByUserSetup** function.

Note: Storing the calibration data to the on-board EEPROM may take a minute or so (if calibration data source was specified as User).

Example

The following example stores the calibration data of the DAC currently under calibration:

```
SHORT   nStatus;

Gx1838CalByUserStoreDacCalibrationData (nHandle, &nStatus);
```

See Also

Gx1838CalByUserSetupDacForMeasurement, Gx1838CalByUserWriteMeasuredVoltage, Gx1838CalByUserReadMeasuredVoltage, Gx1838SetDacCalibrationDataSource, Gx1838GetDacCalibrationDataSources, GxPdoGetErrorString

Gx1838CalByUserWriteMeasuredVoltage

Purpose

Writes the measured value by the user for the specified voltage level used in the calibration process for the DAC currently under calibration.

Syntax

Gx1838CalByUserReadMeasuredVoltage (*nHandle, nVoltageLevel, dVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nVoltageLevel	SHORT	Specified voltage level: 0. GX1838_CAL_LOW_VOLTAGE 1. GX1838_CAL_HIGH_VOLTAGE
dVoltage	DOUBLE	Supplied voltage value measured by the user for the specified voltage level used in the calibration process.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function is valid only if calibration is in process.

The voltage value is the voltage that was measured by the user for the specified calibration voltage level. The written voltage value can be read back using the **Gx1838CalByUserReadMeasuredVoltage** function. The voltage always refers to the DAC currently calibrated.

The function will return an error if currently not in calibration process.

Example

The following example writes the measured low voltage value:

```
SHORT   nStatus;

Gx1838CalByUserWriteMeasuredVoltage (nHandle, GX1838_CAL_LOW_VOLTAGE, 0.12, &nStatus);
```

See Also

Gx1838CalByUserSetupDacForMeasurement, Gx1838CalByUserReadMeasuredVoltage, Gx1838CalByUserStoreDacCalibrationData, Gx1838SetDacCalibrationDataSource, Gx1838GetDacCalibrationDataSources, GxPdoGetErrorString

Gx1838GetBoardSummary

Purpose

Returns the board summary from the on-board EEPROM.

Syntax

Gx1838GetBoardSummary (*nHandle, szSummary, nSumMaxLen, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
szSummary	PSTR	Buffer to contain the returned board info (null terminated) string.
nSumMaxLen	SHORT	Size of the buffer to contain the error string.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX1838 summary string provides the following data from the on-board EEPROM in the order shown:

- Instrument Name (e.g., GX1838)
- EEPROM format version (e.g., 1.00)
- PCB revision (e.g., 'A')
- FPGA version (e.g. 0xA003)
- Serial Number (e.g., 1838008)
- Calibration time and date

For example, the returned string could look like the following:

```
GX1838, Version 1.00, Revision A, S/N 1838008, last calibrated time Mon May 08 14:39:02
2002
```

Example

The following example returns the board summary:

```
SHORT nHandle, nStatus;
CHAR szSummary [256];

Gx1838GetBoardSummary(nHandle, szSummary, sizeof(szSummary), &nStatus)
```

See Also

GxPdoGetDriverSummary, Gx1838Initialize, GxPdoGetErrorString

Gx1838GetChannelRail

Purpose

Returns the specified channel rail connection.

Syntax

Gx1838GetChannelRail (*nHandle, nChannel, pnRail, pnStatus*)

Parameters

Name	Type	Comments
NHandle	SHORT	Handle to a GX1838 board.
nChannel	SHORT	Specified channel number 0-7.
PnRail	PSHORT	Returned Rail connections are as follow: 0. GX1838_RAIL_NONE 1. GX1838_RAIL_A 2. GX1838_RAIL_B 3. GX1838_RAIL_C
PnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns the rail connection for channel two:

```
SHORT nRail, nStatus;

Gx1838GetChannelRail(nHandle, 2, &nRail, &nStatus);
```

See Also

Gx1838SetChannelRail, **Gx1838SetRailSource, Gx1838GetRailSource, GxPdoGetErrorString**

Gx1838GetDacOutputRange

Purpose

Returns the specified DAC output voltage range.

Syntax

Gx1838GetDacOutputRange (*nHandle, nDAC, pdMinVoltage, pdMaxVoltage, pnStatus*)

Parameters

Name	Type	Comments
NHandle	SHORT	Handle to a GX1838 board.
Ndac	SHORT	Specified DAC number. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
PdMinVoltage	PDOUBLE	Returns the specified DAC minimum output voltage range.
PdMaxVoltage	PDOUBLE	Returns the specified DAC maximum output voltage range.
PnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example returns DAC A output voltage range:

```
SHORT   nStatus;
DOUBLE  dMinVoltage, dMaxVoltage;

Gx1838GetDacOutputRange (nHandle, GX1838_DAC_A, &dMinVoltage, &dMaxVoltage, &nStatus);
```

See Also

Gx1838GetBoardSummary, GxPdoGetErrorString

Gx1838GetDacCalibrationDataSource

Purpose

Returns the specified DAC calibration data source currently in use.

Syntax

Gx1838GetDacCalibrationDataSource (*nHandle*, *nDac*, *nDataSource*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nDac	SHORT	Specified DAC number. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
nDataSource	SHORT	Returns calibration data source currently in use, data source are as follow: 1. GX1838_CAL_DATA_FACTORY 2. GX1838_CAL_DATA_USER_NONVOLATILE 4. GX1838_CAL_DATA_USER_VOLATILE
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calibration data source can be changed to any of the currently available data source using the function. The user needs to use the function prior of using either the GX1838_CAL_DATA_USER_NONVOLATILE or GX1838_CAL_DATA_USER_VOLATILE data source in order to verify their existence.

Calibration data sources:

- GX1838_CAL_DATA_FACTORY: Always exists for all three DACs.
- GX1838_CAL_DATA_USER_NONVOLATILE: Exists only if the user calibrated the specified DAC using the **Calibration By User** set of functions. This calibration is a nonvolatile type (save to the on-board EEPROM).
- GX1838_CAL_DATA_USER_VOLATILE: Exists only if the user calibrated the specified DAC using the **Calibration By User** set of functions. This calibration data source is a volatile type (not saved to the board). The calibration data is valid only if the current application (ATEasy or executable) used for calibration was not terminated.

Example

The following example returns the currently used calibration data source for DAC C:

```
SHORT nDataSource, nStatus;

Gx1838GetDacCalibrationDataSource (nHandle, GX1838_DAC_C, &nDataSource, &nStatu);
```

See Also

Gx1838SetDacCalibrationDataSource, Gx1838GetDacCalibrationDataSources, Gx1838CalByUserReadMeasuredVoltage, Gx1838CalByUserSetup, Gx1838CalByUserSetupDacForMeasurement, Gx1838CalByUserStoreDacCalibrationData, Gx1838CalByUserWriteMeasuredVoltage, Gx1838GetBoardSummary, GxPdoGetErrorString

Gx1838GetDacCalibrationDataSources

Purpose

Returns the specified DAC calibration data sources available to be used.

Syntax

Gx1838GetDacCalibrationDataSources (*nHandle, nDac, nDataSources, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nDac	SHORT	Specified DAC number. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
nDataSources	SHORT	Returned calibration data sources is a combination of the follow: 1 GX1838_CAL_DATA_FACTORY 2 GX1838_CAL_DATA_USER_NONVOLATILE 4 GX1838_CAL_DATA_USER_VOLATILE
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function returns the calibration data sources currently available to use. The returned value is a combination of the following:

Calibration data sources:

- GX1838_CAL_DATA_FACTORY: Always exists for all three DACs.
- GX1838_CAL_DATA_USER_NONVOLATILE: Exists only if the user calibrated the specified DAC using the **Calibration By User** set of functions. This calibration is a nonvolatile type (save to the on-board EEPROM).
- GX1838_CAL_DATA_USER_VOLATILE: Exists only if the user calibrated the specified DAC using the **Calibration By User** set of functions. This calibration data source is a volatile type (not saved to the on-board EEPROM). The calibration data is valid only if the current application (ATEasy or executable) used for calibration was not terminated.

Example

The following example returns the currently available to use calibration data sources for DAC C:

```
SHORT nDataSources, nStatus;

Gx1838GetDacCalibrationDataSources (nHandle, GX1838_DAC_C, &nDataSources, &nStatus);
```

See Also

Gx1838GetDacCalibrationDataSource, Gx1838SetDacCalibrationDataSource, Gx1838CalByUserReadMeasuredVoltage, Gx1838CalByUserSetup, Gx1838CalByUserSetupDacForMeasurement, Gx1838CalByUserStoreDacCalibrationData, Gx1838CalByUserWriteMeasuredVoltage, Gx1838GetBoardSummary, GxPdoGetErrorString

Gx1838GetDacCalibrationDataString

Purpose

Returns the specified DAC calibration source data string.

Syntax

Gx1838GetDacCalibrationDataString (*nHandle, nDac, nDataSource, pszCalData, nSumMaxLen, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nDac	SHORT	Specified DAC number. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
nDataSource	SHORT	Specified calibration data source as follow: 1. GX1838_CAL_DATA_FACTORY 2. GX1838_CAL_DATA_USER_NONVOLATILE 4. GX1838_CAL_DATA_USER_VOLATILE
pszCalData	*PSTR*	Buffer to contain the returned board info (null terminated) string.
nSumMaxLen	SHORT	Size of the buffer to contain the error string.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

See the **Gx1838SetDacCalibrationDataSource** function for details of each of the calibration data source. The user needs to check if the specified *nDataSource* supports by the board by calling **Gx1838GetDacCalibrationAvailableDataSource** function.

The DAC calibration source data string provides detailed information for each DACs' calibration parameters. The returned string provides the following data in the order shown:

- DAC number (A through C)
- Load Value:(e.g., 1234.56)
- Calibration time and date (e.g. Thu Oct 09 10:24:31 2003)

Example

The following is an example of DAC B with calibration source set as

```
GX1838_CAL_DATA_USER_NONVOLATILE:

"DAC B, Load Value:1234.56, Calibration time: Thu Oct 09 10:24:31 2003";
```

See Also

Gx1838GetDacCalibrationDataSources, Gx1838GetDacCalibrationDataSource, Gx1838SetDacCalibrationDataSource, Gx1838GetBoardSummary GxPdoGetErrorString

Gx1838GetDacVoltage

Purpose

Returns the specified DAC voltage.

Syntax

Gx1838GetDacVoltage (*nHandle, nDac, pdVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nDac	SHORT	Specified DAC number. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
pdVoltage	PDOUBLE	Returned DAC voltage, voltage can be between 32V to –10V
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns DAC A voltage:

```
SHORT nStatus;
DOUBLE dVoltage;

Gx1838GetDacVoltage (nHandle, DAC_A, &dVoltage, &nStatus);
```

See Also

Gx1838SetDacVoltage, Gx1838SetRailSource, Gx1838GetRailSource, GxPdoGetErrorString

Gx1838GetRailSource

Purpose

Returns the specified rail source.

Syntax

Gx1838GetRailSource (*nHandle*, *nRail*, *pnSource*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nRail	SHORT	Specified rail number as follow: 1. GX1838_RAIL_A 2. GX1838_RAIL_B 3. GX1838_RAIL_C
pnSource	PSHORT	Specified rail source, see comments for details.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Rails sources are as follow:

RAIL_A:

0. GX1838_SOURCE_OPEN
1. GX1838_SOURCE_INTERNAL (DAC A only)
2. GX1838_SOURCE_EXTERNAL (pin 10 on Connector J3).

RAIL_B:

0. GX1838_SOURCE_OPEN
1. GX1838_SOURCE_INTERNAL (DAC B only)
2. GX1838_SOURCE_EXTERNAL (pin 11 on Connector J3).

RAIL_C:

0. GX1838_SOURCE_OPEN
1. GX1838_SOURCE_INTERNAL (DAC C only)
2. GX1838_SOURCE_EXTERNAL (pin 12 on Connector J3).
4. GX1838_SOURCE_FIXED_LOAD (Fixed load of 500Ohms 1W).

Example

The following example returns the source for rail B:

```
SHORT nSource, nStatus;

Gx1838GetRailSource (nHandle, RAIL_B, &nSource, &nStatus)
```

See Also

Gx1838SetRailSource, Gx1838SetDacVoltage, Gx1838GetDacVoltage, GxPdoGetErrorString

Gx1838Initialize

Purpose

Initializes the GX1838 driver for the specified PXI slot.

Syntax

Gx1838Initialize (*nSlot, pnHandle, pnStatus*)

Parameters

Name	Type	Comments
nSlot	SHORT	GX1838 board slot number.
pnHandle	PSHORT	Returned Handle for a GX1838 board.
pnStatus	PSHORT	Returned status: 0 on success, 1 on failure.

Comments

The function returns a handle for use with other GX1838 functions to program the board. The function does not change any of the board settings.

The specified PXI slot number can be retrieved by the **PXI/PCI Explorer** applet that can be opened from the Windows **Control Panel**. You may also use the label on the chassis below the PXI slot where the board is installed.

Example

The following example initializes a GX1838 board at PXI bus slot 1.

```
SHORT nHandle, nStatus;

Gx1838Initialize(1, &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

GxPdoGetErrorString, Gx1838Reset

Gx1838InitializeVisa

Purpose

Initializes the driver for the specified PXI slot using the default VISA provider.

Syntax

Gx1838InitializeVisa (*szVisaResource, pnHandle, pnStatus*)

Parameters

Name	Type	Comments
szVisaResource	PCSTR	String identifying the location of the specified board in order to establish a session.
panhandle	PSHORT	Returned Handle (session identifier) that can be used to call any other operations of that resource
pnStatus	PSHORT	Returned status: 0 on success, 1 on failure.

Comments

The **Gx1838InitializeVisa** opens a VISA session to the specified resource. The function uses the default VISA provider configured in your system to access the board. You must ensure that the default VISA provider support PXI/PCI devices and that the board is visible in the VISA resource manager before calling this function.

The first argument *szVisaResource* is a string that is displayed by the VISA resource manager such as NI Measurement and Automation (NI_MAX). It is also displayed by Marvin Test Solutions PXI/PCI Explorer as shown in the prior figure. The VISA resource string can be specified in several ways as follows:

- Using chassis, slot: "PXI0::CHASSIS1::SLOT5"
- Using the PCI Bus/Device combination: "PXI9::13::INSTR" (bus 9, device 9).
- Using alias: "PDO1". Use the PXI/PCI Explorer to set the device alias.

The function returns a board handle (session identifier) that can be used to call any other operations of that resource. The session is opened with VI_TMO_IMMEDIATE and VI_NO_LOCK VISA attributes. On terminating the application the driver automatically invokes **viClose**() terminating the session.

Example

The following example initializes a DC Power Supply board at PXI bus 5 and device 11.

```
SHORT nHandle, nStatus;

Gx1838InitializeVisa("PXI5::11::INSTR", &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

Gx1838Initialize, Gx1838GetErrorString, Gx1838Reset

Gx1838Panel

Purpose

Opens a virtual panel used to interactively control the GX1838.

Syntax

Gx1838Panel (*pnHandle, hwndParent, nMode, phwndPanel, pnStatus***)**

Parameters

Name	Type	Comments
pnHandle	PSHORT	Handle to a GX1838 board.
hwndParent	HWND	Panel parent window handle. A value of 0 sets the desktop as the parent window.
nMode	SHORT	The mode in which the panel main window is created. 0 for modeless window and 1 for modal window.
phwndPanel	HWND	Returned window handle for the panel.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function is used to create the panel window. The panel window may be open as a modal or a modeless window depending on the *nMode* parameters.

If the mode is set to modal dialog (*nMode*=1), the panel will disable the parent window (*hwndParent*) and the function will return only after the window closes. In that case, the *pnHandle* may return the handle created by the user using the panel Initialize dialog. This handle may be used when calling other GX1838 functions.

If a modeless dialog was created (*nMode*=0), the function returns immediately after creating the panel window returning the handle to the panel - *phwndPanel*. It is the responsibility of calling program to dispatch windows messages to this window so that the window can respond to messages.

Example

The following example opens the panel in modal mode:

```
DWORD dwPanel;
SHORT nHandle=0, nStatus;

Gx1838Panel(&nHandle, 0, 1, &dwPanel, &nStatus);
```

See Also

Gx1838Initialize, GxPdoGetErrorString

Gx1838Reset

Purpose

Resets the GX1838 board to its default settings.

Syntax

Gx1838Reset (*nHandle, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX1838 board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function does the following:

- Disconnect all rails from their sources.
- Disconnect all channels from rails.
- Sets all DACs to 0 volts.

Example

The following example initializes and resets the GX1838 board:

```
Gx1838Initialize (1, &nHandle, &nStatus);
Gx1838Reset (nHandle, &nStatus);
```

See Also

Gx1838Initialize, GxPdoGetErrorString

Gx1838SetChannelRail

Purpose

Sets the specified channel rail connection.

Syntax

Gx1838SetChannelRail (*nHandle, nChannel, nRail, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nChannel	SHORT	Specified channel number 0-7.
nRail	SHORT	Rail connections are as follow: 0. GX1838_RAIL_NONE 1. GX1838_RAIL_A 2. GX1838_RAIL_B 3. GX1838_RAIL_C
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example connect channel two to Rail B:

```
SHORT nRail, nStatus;

Gx1838SetChannelRail(nHandle, 2, RAIL_B, &nStatus);
```

See Also

Gx1838GetChannelRail, **Gx1838SetRailSource, Gx1838GetRailSource, GxPdoGetErrorString**

Gx1838SetDacCalibrationDataSource

Purpose

Returns the specified DAC calibration data source currently in use.

Syntax

Gx1838GetDacCalibrationDataSource (*nHandle*, *nDac*, *nDataSource*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nDac	SHORT	Specified DAC number. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
nDataSource	SHORT	Returns calibration data source currently in use, data sources are as follow: 1. GX1838_CAL_DATA_FACTORY 2. GX1838_CAL_DATA_USER_NONVOLATILE 4. GX1838_CAL_DATA_USER_VOLATILE
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calibration data source can be changed to any of the currently available data source using the function. The user needs to use the function prior of using either the GX1838_CAL_DATA_USER_NONVOLATILE or GX1838_CAL_DATA_USER_VOLATILE data source in order to verify their existence.

Calibration data sources:

- GX1838_CAL_DATA_FACTORY: Always exists for all three DACs.
- GX1838_CAL_DATA_USER_NONVOLATILE: Exists only if the user calibrated the specified DAC using the **Calibration By User** set of functions. This calibration is a nonvolatile type (save to the on-board EEPROM).
- GX1838_CAL_DATA_USER_VOLATILE: Exists only if the user calibrated the specified DAC using the **Calibration By User** set of functions. This calibration data source is a volatile type (not saved to the on-board EEPROM). The calibration data is valid only if the current application (ATEasy or executable) used for calibration was not terminated.

Example

The following example set DAC C calibration data source for GX1838_CAL_DATA_USER_NONVOLATILE:

```
SHORT nStatus;

Gx1838SetDacCalibrationDataSource (nHandle, GX1838_DAC_C, GX1838_CAL_DATA_USER_NONVOLATILE,
&nStatu);
```

See Also

Gx1838GetDacCalibrationDataSources, Gx1838GetDacCalibrationDataSource, Gx1838CalByUserReadMeasuredVoltage, Gx1838CalByUserSetup, Gx1838CalByUserSetupDacForMeasurement, Gx1838CalByUserStoreDacCalibrationData, Gx1838CalByUserWriteMeasuredVoltage, Gx1838GetBoardSummary, GxPdoGetErrorString

Gx1838SetDacVoltage

Purpose

Sets the specified DAC voltage.

Syntax

Gx1838SetDacVoltage (*nHandle, nDac, dVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nDac	SHORT	Specified DAC number. 0. GX1838_DAC_A 1. GX1838_DAC_B 2. GX1838_DAC_C
dVoltage	DOUBLE	DAC voltage, voltage can be between –10V to 32V
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets DAC A to 10V:

```
SHORT nStatus;

Gx1838SetDacVoltage (nHandle, DAC_A, 10.0, &nStatus);
```

See Also

Gx1838GetDacVoltage, Gx1838SetRailSource, Gx1838GetRailSource, GxPdoGetErrorString

Gx1838SetRailSource

Purpose

Sets the specified rail source.

Syntax

Gx1838SetRailSource (*nHandle, nRail, nSource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX1838 board.
nRail	SHORT	Specified rail number as follow: 0. GX1838_RAIL_A 1. GX1838_RAIL_B 2. GX1838_RAIL_C
nSource	SHORT	Specified rail source, see comments for details.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Rails sources are as follow:

RAIL_A:

0. GX1838_SOURCE_OPEN
1. GX1838_SOURCE_INTERNAL (DAC_A only)
2. GX1838_SOURCE_EXTERNAL (pin 10 on Connector J3).

RAIL_B:

0. GX1838_SOURCE_OPEN
1. GX1838_SOURCE_INTERNAL (DAC_B only)
2. GX1838_SOURCE_EXTERNAL (pin 11 on Connector J3).

RAIL_C:

0. GX1838_SOURCE_OPEN
1. GX1838_SOURCE_INTERNAL (DAC_C only)
2. GX1838_SOURCE_EXTERNAL (pin 12 on Connector J3).
4. GX1838_SOURCE_FIXED_LOAD (Fixed load of 500Ohms 1W).

Example

The following example connects rail B to DAC_B:

```
SHORT nStatus;

Gx1838SetRailSource (nHandle, RAIL_B, GX1838_SOURCE_INTERNAL, &nStatus)
```

See Also

Gx1838GetRailSource, Gx1838SetDacVoltage, Gx1838GetDacVoltage, GxPdoGetErrorString

GxPdoGetDriverSummary

Purpose

Returns the driver name and version.

Syntax

GxPdoGetDriverSummary (*pszSummary ,nSummaryMaxLen, pdwVersion, pnStatus*)

Parameters

Name	Type	Comments
pszSummary	PSTR	Buffer to the returned driver summary string.
nSummaryMaxLen	SHORT	The size of the summary string buffer.
pdwVersion	LPDWORD	Returned version number. The high order word specifies the major version number where the low order word specifies the minor version number.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The returned string is: "GXPDO Driver for GX1838. Version 1.0, Copyright © Marvin Test Solutions 2002"

Example

The following example prints the driver version:

```
CHAR   sz[128];
DWORD  dwVersion;
SHORT  nStatus;

GxPdoGetDriverSummary (sz, sizeof sz, &dwVersion, &nStatus);
printf("Driver Version %d.%d", (INT)(dwVersion>>16), (INT)
    dwVersion &0xFFFF);
```

See Also

GxPdoGetErrorString

GxPdoGetErrorString

Purpose

Returns the error string associated with the specified error number.

Syntax

GxPdoGetErrorString (*nError*, *pszMsg*, *nErrorMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
nError	SHORT	Error number.
pszMsg	PSTR	Buffer to the returned error string.
nErrorMaxLen	SHORT	The size of the error string buffer.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the error string associated with the *nError* as returned from other driver functions.

The following table displays the possible error values; not all errors apply to this board type:

Warnings

12 Measurement Value is out of range

11 Unable to find suitable filter value AutoFilter mode

10 Measurement timeout, no signal was detected. Timeout value needs to be greater than Gate Time.

Resource Errors

0 No error has occurred

-1 Unable to open the HW driver. Check if HW is properly installed

-2 Board does not exist in this slot/base address

-3 Different board exist in the specified PCI slot/base address

-4 PCI slot not configured properly. You may configure using the PciExplorer from the Windows Control Panel

-5 Unable to register the PCI device

-6 Unable to allocate system resource for the device

-7 Unable to allocate memory

-8 Unable to create panel

-9 Unable to create Windows timer

-10 Bad or Wrong board EEPROM

-11 Not in calibration mode

-12 Board is not calibrated

-13 Function is not supported by the specified board

General Parameter Errors

-20 Invalid or unknown error number

-21 Invalid parameter

-22 Illegal slot number

-23 Illegal board handle

-24 Illegal string length

-25 Illegal operation mode

VISA Errors

-30 Unable to Load VISA32/64.DLL, make sure VISA library is installed

-31 Unable to open default VISA resource manager, make sure VISA is properly installed

-32 Unable to open the specified VISA resource

-33 VISA viGetAttribute error

-34 VISA viInXX error

-35 VISA ViMapAddress error

Misc Errors

-38 Unable to lock a board resource, resource is used by another process or thread

Parameter Errors

-40 Invalid Channel number (0 or 1)

-41 Invalid Channel rail

-42 Invalid DAC

-43 Invalid DAC Voltage

-44 Invalid Rail

-45 Invalid Rail Source

-46 Invalid Group

-47 Invalid Gain Value

-48 Invalid Offset Value

-49 Invalid Calibration Data Source

-50 EEPROM Busy Timeout

Board/execution errors

-60 Unable to Set DAC

Calibration errors

-80 Calibration by user, sequence error

-81 Calibration by user, invalid calibration data

-82 Calibration by user, invalid load value

-83 Calibration by user, change calibration source

Example

The following example initializes the board. If the initialization failed, the following error string is printed:

```
CHAR    sz[256];
SHORT   nStatus, nHandle;
..
Gx1838Initialize (3, &Handle, &Status);
if (nStatus<0)
{   GxPdoGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz); // prints the error string returns
}
```

Index

.

.NET ii

3

3U PXI 8

A

Accessories 19

Architecture 1, 6

ATEasy ..... ii, 21, 22, 23, 25, 26

B

Block Diagram 6

Board Description 1, 4

Board Handle 28

Board Rails 7

Borland 21, 26

Borland-Delphi 26

C

c# 25

C/C++ 21, 22, 25

C++ 25

CalEasy 3

Calibration 38, 64

Connector 20

Connectors 3, 19, 20

Connectors and Accessories 19

Corrupt files 23

D

Delphi ..... ii, 21, 25, 26

Distributing 29

Driver

- Directory 21
- Files 21

Driver Version 28

E

Error Handling 28

Error-Handling 28

Example 22

F

Functions Reference 37

G

GX1838 ..... 1, 5

GX1838 Functions 38

Gx1838CalByUserDeleteClibration 39

Gx1838CalByUserReadMeasuredVoltage 40

Gx1838CalByUserSetup 41

Gx1838CalByUserSetupDacForMeasurement 42

Gx1838CalByUserStoreDacCalibrationData 43

Gx1838CalByUserWriteMeasuredVoltage 44

Gx1838GetBoardSummary 45

Gx1838GetChannelRail 46

Gx1838GetDacCalibrationDataSource 48

Gx1838GetDacCalibrationDataSources 49

Gx1838GetDacCalibrationDataString 50

Gx1838GetDacOutputRange 47

Gx1838GetDacVoltage 51

Gx1838GetRailSource 52

Gx1838Initialize 10, 27, 28, 37, 38, 53

Gx1838InitializeVisa 10, 27, 28, 38, 54

Gx1838Panel 55

Gx1838Reset 27, 28, 56

Gx1838SetChannelRail 57

Gx1838SetDacCalibrationDataSource 58

Gx1838SetDacVoltage 60

Gx1838SetRailSource 61

GXPDO 1, 14

- Driver-Description 25
- Header-file 25
- Help-File-Description 22
- Panel-File-Description 21

GXPDO.DLL 21, 26

GXPDO.EXE 14

GXPDO.H 21, 25
GXPDO.LIB 21, 25
GXPDO.llb 26
GXPDO.PAS 21, 26
GXPDO.VB 21
GXPDO64.LIB 21, 25
GXPDOBC.LIB 21
GxPdoGetDriverSummary 28, 62
GxPdoGetErrorString 28, 37, 63
GXPDOPANE64L.EXE 21
GXPDOPANEL.DLL 21
GXPDOPANEL.EXE 28
GXPDOPANEL64.EXE 28

H

Handle 17, 18, 27, 28
HW 18, 21, 25

I

Installation: 17, 19

J

J3 .....3, 6, 20

L

LabView 26
LabView/Real Time 26
Linux 26

N

nHandle 26, 28

O

OnError 26
Output Channels 6
Output Channels Connector 20

P

Panel 9, 12, 23, 27, 28, 55
Pascal 21, 25, 26
PCI 21
Plug & Play 18
pnStatus 28, 37
Program-File-Descriptions 21
Programming
  Borland-Delphi 26
  Error-Handling 28
  Panel-Program 28
Programming Using Visual Basic 25
Programming Using Visual c# 25
PXI 3, 6, 16, 17, 18, 19, 27
PXI/PCI 53
PXI/PCI Explorer 10, 16, 27, 28, 54
PXIeSYS.INI 10
PXISYS.INI 10

R

Rails Output Voltage With Different Loads 7
Rails Sources Diagram 7
README.TXT 21, 22
Readme-File 22
Reset 28

S

Sample 29, 30
Setup 14, 21, 23
Setup Maintenance 23
Setup-and-Installation 13
Slot 10, 17, 19, 27
Specifications 1, 8
Supported-Development-Tools 25
System
  Directory 21

T

The GX3348 Driver 25
Typical Output Channel Diagram 6

U

Using the GXPDO driver functions 27

V

Virtual Panel 9, 10, 11, 12, 14, 27, 55
  About Page 12
  Initialize Dialog 10
  Setup Page 11

VISA.....10, 27, 28, 54, 64
Visual Basic.....ii, 21, 22, 25, 26
Visual C++ .....ii, 21, 25

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP