GX7400
GX7400A

PXI Programmable DC Power Supply

User's Guide

Last updated October 31, 2011

GEOTEST

MARVIN TEST SYSTEMS, INC.

Safety and Handling

Each product shipped by Geotest 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

Geotest products are warranted against defects in materials and workmanship for a period of 12 months. Geotest 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 Geotest Customer Technical Services Department via http://www.geotestinc.com/magic/ - the Geotest on-line support system.

If You Need Help

Visit our web site at http://www.geotestinc.com for more information about Geotest 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.geotestinc.com/magic/

You can also use Geotest 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 Geotest 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 Geotest has been advised of the possibility for such damages.

Copyright

Trademarks

Table of Contents

Safety and Handling i
Warranty i
If You Need Help i
Disclaimer i
Copyright i
Trademarks ii

Table of Contents iii

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
Board Description 4
Features 5
Architecture 6
Block Diagram 6
Bus Interface, Control and Logic 6
Power Conditioning Unit 6
Software 7
Function Library 7
Virtual Panel Description 8
Virtual Panel Initialize Dialog 9
Virtual Panel Setup Page 10
Virtual Panel 11

Chapter 3 - Setup and Installation 12

Getting Started 12
Packing List 12
Unpacking and Inspection 12
System Requirements 12
Installation of the GXPS Driver 13
Overview of the GXPS Software 14
Configuring Your PXI System using the PXI/PCI Explorer 15
Board Installation 16

Before you Begin 16
Electric Static Discharge (ESD) Precautions 16
Installing a Board 16
Plug & Play Driver Installation 17
Removing a Board 17
Installation Directories 18
GXPS Driver Files Description 18
Driver File and Virtual Panel 18
GXPS On-line Help and Manual 19
Example Programs 19
Setup Maintenance Program 20
Chapter 4 - Programming the Power Supply 21
The GXPS Driver 21
Programming Using C/C++ Tools 21
Programming Using Visual Basic 21
Programming Using Pascal/Delphi 22
Programming GXPS Boards Using ATEasy® 22
Using the GXPS driver functions 23
Initialization, HW Slot Numbers and VISA Resource 23
Board Handle 24
Reset 25
Error Handling 25
Driver Version 25
Panel 25
Distributing the Driver 25
Sample Program 26
Sample Program Listing 26
Chapter 5 - In-System Calibration 31
Calibration Interval 31
In-System Calibration license setup 32
Calibration required instruments 33
Connections diagram 33
Running In-System calibration from the virtual panel 36
Front Panel Calibration dialog 37
Calibration dialog tutorial 38
Step 1: Pre-calibration settings 38
Steps 2 through 9: Measure voltage 39

Step 10: Save Channel's Calibration data ... 39
In-System Calibration Example program ... 40
Sample User Calibration Program Listing ... 40
Chapter 6 - Functions Reference ... 45
Introduction ... 45
Parameter Prefix Names ... 45
Functions by Category ... 45
GxPsGetBoardSummary ... 48
GxPsGetCalibrationDate ... 49
GxPsGetCalibrationInfo ... 50
GxPsGetChannelCalibrationDate ... 52
GxPsGetCurrent ... 53
GxPsGetCurrentLimit ... 54
GxPsGetDriverSummary ... 55
GxPsGetErrorString ... 56
GxPsGetExtendedSerialNumber ... 59
GxPsGetStandby ... 60
GxPsGetState ... 61
GxPsGetType ... 62
GxPsGetVoltage ... 63
GxPsInitialize ... 64
GxPsInitializeVisa ... 65
GxPsPanel ... 66
GxPsRecallSettings ... 67
GxPsReset ... 68
GxPsSetCurrentLimit ... 69
GxPsSetStandby ... 70
GxPsSetStandbyImmediate ... 71
GxPsSetVoltage ... 72
GxPsSetVoltageAndWaitUntilReady ... 73
GxPsStoreSettings ... 74
GxPsUpgradeFirmware ... 75
GxPsUpgradeFirmwareStatus ... 76
GxPsUserCalReadMeasuredVal ... 78
GxPsUserCalRestoreFactoryCalibration ... 79
GxPsUserCalSetup ... 80
GxPsUserCalSetupForMeasurement ... 81

GxPsUserCalStoreCalibrationData 82
GxPsUserCalWriteMeasuredVal 83
Appendix A– Connectors and Cables 85
Connections 85
Channel 1 Connector Pins 85
Channel 2 Connector Pins 85
Connectors and Accessories 86
Appendix B – Specifications 87
Modules 87
Output Characteristics 87
Input AC Power 87
Power Requirements 88
Protection Circuitry 88
Physical 88
Index 89

Chapter 1 - Introduction

This manual contains instructions for installation, testing, programming, and controlling the Geotest power supply instrument on a 6U PXI/cPCI platform.

Manual Scope and Organization

Manual Scope

This GX7400/GX7400A User’s Guide provides all the information needed to install, program, and use Geotest’s GX7400 power supply. This manual assumes that the user has a general knowledge of PC-based computers, Windows operating systems and an electronics background. Some knowledge of programming and development tools will permit computer program control of the power supply. Unless specifically stated otherwise, reference to the GX7400 features and capabilities applies to the GX7400A version as well.

Manual Organization

The GX7400 manual is organized in the following manner:

Chapter	**Content**
Chapter 1	Introduces this GX7400 User’s Guide.
Chapter 2	Summarizes power supply features, architecture, hardware and software.
Chapter 3	Furnishes step-by-step directions for installing and setting up the hardware and GXPS software.
Chapter 4	Presents details on how to program the power supply by using the supplied driver and example files.
Chapter 5 – In-System Calibration	Describes the procedure used to perform the in-system calibration.
Chapter 6	Contains an alphabetical list of driver functions including syntax, variables, programming comments and samples.
Appendix A	Describes the power supply connectors and cables.
Appendix B	Summarizes GX7400 specifications.
Index	A keyword list of important topics and concepts in this manual.

Conventions Used in This Manual

Naming conventions used throughout this manual are:

Example	**Description**
Copy or **Paste**	Commands are indicated in bold type.
Shift+F1	Keys are often used in combinations. The example to the left instructs the user to hold down the shift key while pressing the FI key at the same time. When key combination instructions are separated by commas, such as ALT+D, A, hold the ALT key while pressing D, then press A.
Direction Keys	Refer to the up arrow (↑), down arrow (↓), right arrow (→), and left arrow (←) keys.
cd bold	Bolded text must be entered from the keyboard exactly as shown.
cd *directory name*	Italicized text is a placeholder for variables or other items that the user must define and enter from the keyboard.

Example	**Description**
`examples`	Examples and source code are indicated in Courier, a fixed pitch font.
0x*hexnumber*	An integer in hexadecimal notation, E.g., 0x10A equals 266 in decimal.

Chapter 2 - Overview

Introduction

The GX7400 is a dual programmable DC power supply designed for use in any 6U CompactPCI (cPCI) or PXI instrumentation chassis. It provides power for a Unit Under Test (UUT) and it is fully compatible with other ATE tools. The GX7400 occupies three slots of CompactPCI or PXI chassis and is controlled by the host computer bus. AC power is provided using a panel-mounted receptacle. Both channels are fully isolated and can be externally connected in series in order to provide a higher output voltage. The input AC power supports input voltages of 115VAC and 230VAC making it suitable for worldwide use. The GX7400 model's input AC voltage is selectable to be either 115VAC or 230VAC range. The GX7400A model supports an input voltage range from 115VAC to 230VAC (nominal).

Three DC modules are available for the GX7400:

- 0 to 15 V @ 10 A
- 0 to 30 V @ 5 A
- 0 to 60 V @ 2.5 A

Programmable current limits, programmable isolation relays, over-voltage and over-current, and remote sense, are standard for each of the modules.

The fixed modules provide a configurable fixed DC output level rated up to 100W (Geotest no-longer sells the fixed modules), available voltage levels are:

- 3.3VDC @ 10A
- 5VDC @ 10A
- 12VDC @ 8.4A
- 15VDC @ 7.6A
- 28VDC @ 3.6A.

Outputs are routed through an on-board isolation relay and include remote sensing capability.

An illustration of the power supply is shown in Figure 2-1: GX7400 Power Supply.

The GX7400 receives input AC line power from a connector on the front panel and may be operated from either 115VAC or 230VAC input selected via side-mounted selection switch (GX7400 only).

Note: This switch ***must*** be set to the proper operating voltage in order to prevent damage to the instrument.

The GX7400 plugs into Geotest GX7000 or any 6U cPCI/PXI chassis and is controlled via the host computer bus. DC output connectors for the UUT are also located on the front panel of the unit, along with an On/Off switch. The GX7400 is fully enclosed and shielded in order to prevent any Electro Magnetic Interference (EMI) to other instruments in the system.

Board Description

Figure 2-1: GX7400 Power Supply shows the GX7400 with its two front panel connectors. The GX7400 receives input AC line power from a connector on the front panel and may be operated from either 115VAC or 230VAC input selected via side-mounted selection switch. Outputs are routed through an on-board isolation relay and include remote sensing capability.

The GX7400 is a modular power supply that can accommodate two power modules: programmable or fixed. The power section is optically isolated from the PXI bus and the outputs are isolated from each other. A 14-bit DAC and a 16-bit A/D provide extremely accurate programming and readback capabilities.

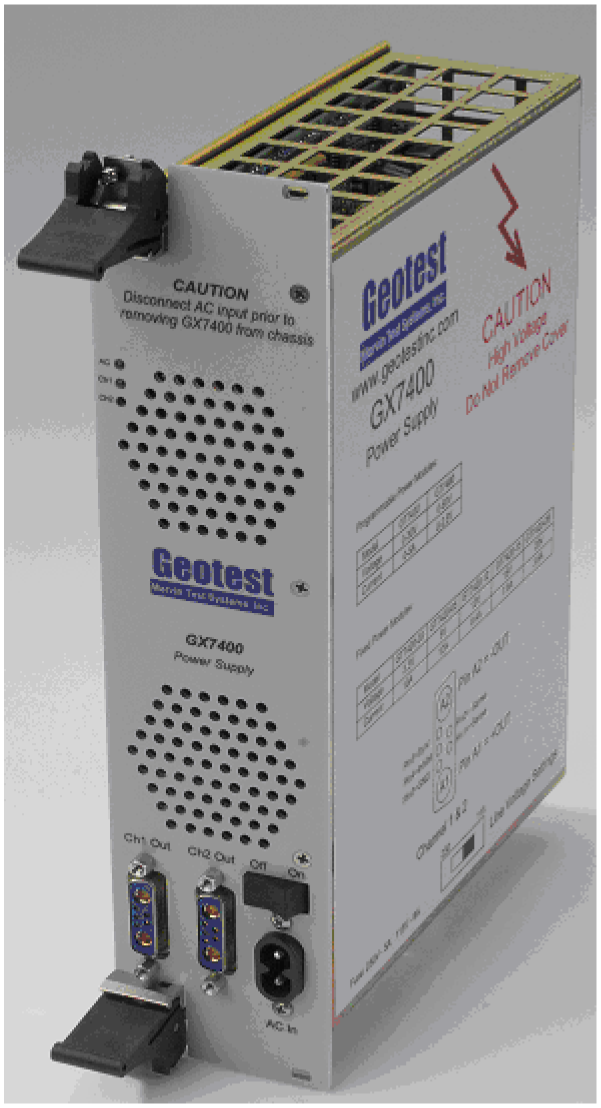


Figure 2-1: GX7400 Power Supply

Features

- Up to two independent DC power modules
- 115VAC or 230VAC Input (GX7400 only switch selectable). GX7400A offers a wide range input.
- All power interfaces, Power Switch, AC Input, DC Output connectors conveniently located on the faceplate.
- AC and DC on/off power indicators located on the faceplate
- Real time output voltage and current for complete control
- Over-voltage and over-current protection
- Over-temperature and short circuit protection
- Remote inhibit via faceplate output connector
- All modules have isolated power outputs
- Per module In-System-Calibration capability
- Fast Turn On and Standby times

Applications

- Power Supply Testing
- Automated Test Equipment (ATE)
- R&D Verification Systems

Architecture

Block Diagram

The GX7400 consists of a triple cPCI/PXI slot unit housing the bus interface, power conditioning and rectification circuitry and power conversion stages. The GX7400 utilizes chassis power for the bus interface and AC line power for all other needs. Figure 2-2 below shows a block diagram of the GX7400.

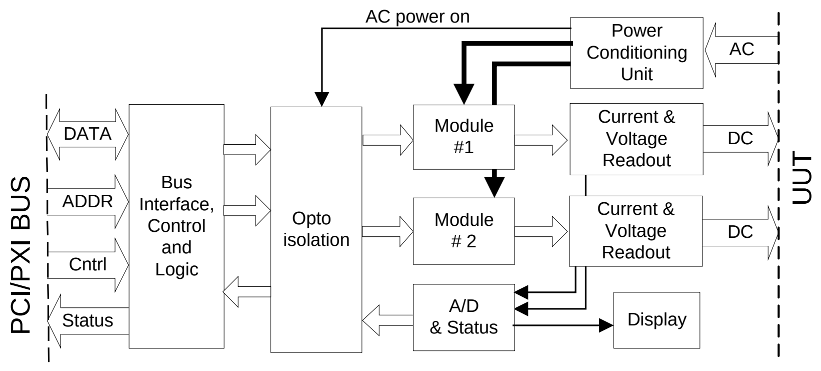


Figure 2-2: GX7400 Block Diagram

Bus Interface, Control and Logic

The GX7400 Bus Interface, Control and Logic unit consists of PCI bus interface and power supply control logic.

The AC Power On signal is sent to the control section bus whenever the external AC line power switch is on. The control unit has an on-board EEPROM storing calibration data for each module. Each module is controlled by dedicated logic, which handles the A/D and D/A unit and control/protection of the module.

Power Conditioning Unit

The 115/230V AC line voltage is converted to DC voltage by a power-conditioning unit. The power-conditioning unit accommodates up to two power modules. Power modules are completely isolated from each other, from the line input, and from the controlling logic, and communicate with the power supply control through a bi-directional serial link. Protection circuits protect against over-voltage, over-current, over-temperature and shorted output lines.

Available power modules are described in detail in Appendix B, “Specifications.”

Software

The GX7400 software package, also referred to as the GxPs package, includes the following features:

- Function Library (DLL and LIB files).
- Virtual Panel (stand-alone executable).
- Programming language interface files for C, VB, ATEasy, LabView and more
- Programming Examples
- Windows Help File
- Installation and Setup utility.

Function Library

This part of the module contains functions that perform the following:

Initialize	Initializing the board and retrieving the board handle.
Reset	Resetting the power supply to the default state.
Store	Stores the power supply voltage and current setting
Recall	Retrieve the previously stored power supply voltage and current setting.
Set Current	Apply the current setting to the specified channel.
Set Voltage	Apply the voltage setting to the specified channel.
On/Standby	Sets the specified channel to On (channel is active) or to Stand by.

The driver is supplied as a Windows 32-bitand 64 bit DLLs that are compatible with Windows. A separate software package (GtLinux) provide interface to Linux,

Exported functions are declared using Pascal calling conventions to provide access from various development tools. Interface files and examples are provided for various development tools and programming languages, such as VC++, VB, Pascal, Borland, and Geotest’s ***ATEasy***.

Virtual Panel Description

The **GXPS** includes a virtual panel program, which provides full access to the various configuration settings and operating modes. To understand the front panel operation, it is best to become familiar with the functionality of the board.

To open the virtual panel application, select **GX7400 Panel** from the **Geotest**, **GXPS** menu under the **Start** menu. The GX7400 virtual panel opens as shown here:

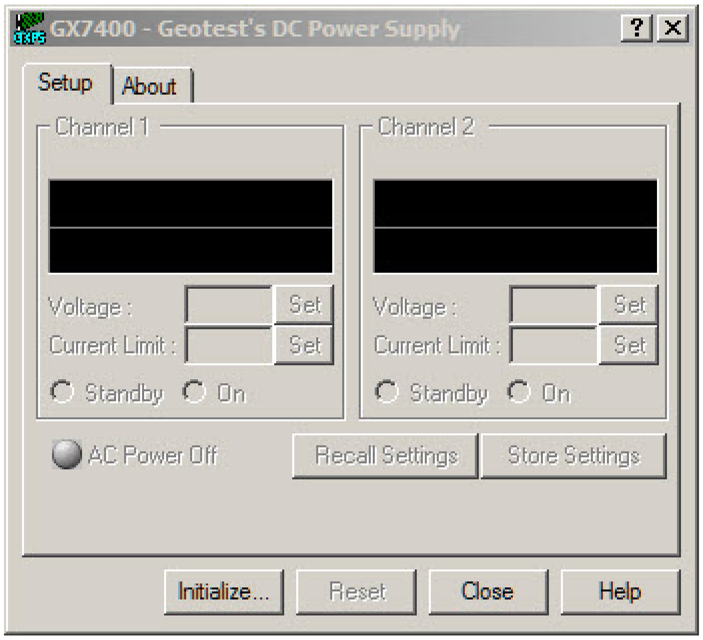


Figure 2-3: GX7400 Virtual Panel

The function of the panel button controls are shown below:

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

Reset - resets the PXI board settings to their default state and clears the reading.

Apply – applies changed settings to the board

Close - closes the panel. Closing the panel **does not affect** the board settings.

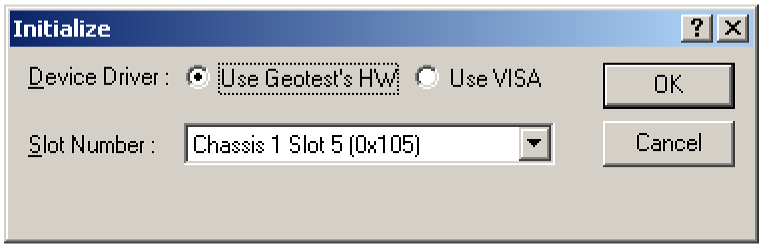
Help - opens the on-line help window. In addition to the help menu, the caption shows a **What's This Help** button (?) button. This button can be used to obtain help on any control that is displayed in the panel window. To display the What's This Help information click on the (?) button and then click on the control – a small window will display the information regarding this control.

Virtual Panel Initialize Dialog

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

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

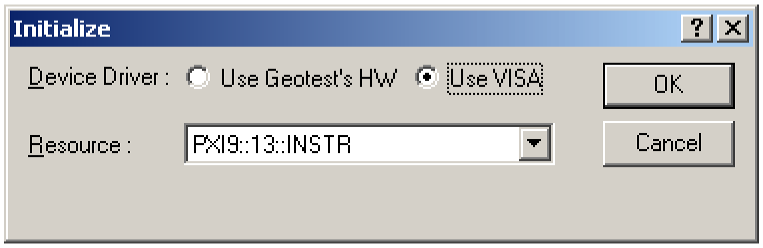
1. **Use Geotest's HW** – this is the device driver installed by the setup program and is the default driver. When selected, the **Slot Number** list displays the available counter boards installed in the system and their slots. The chassis, slots, devices and their resources are also displayed by the 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 0x105 (chassis 1 Slot 5). This is the slot number argument (*nSlot*) passed by the panel when calling the driver **GxPsInitialize** function used to initialize driver with the specified board.



Device Driver : [✓ Use Geotest's HW] Use VISA
Slot Number : Chassis 1 Slot 5 (0x105)

Figure 2-4: Initialize Dialog Box using Geotest's 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 Geotest **PXI/PCI Explorer**. The following figure shows PXI9::13::INSTR as the VISA resource (PCI bus 9 and Device 13). This is VISA resource string argument (*szVisaResource*) passed by the panel when calling the driver **GxPsInitializeVisa** function to initialize the driver with the specified board.



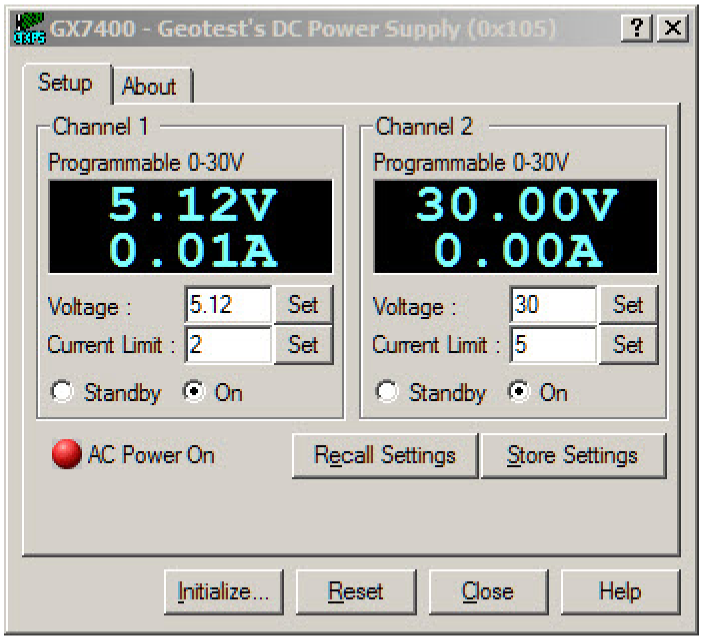
Device Driver : Use Geotest's HW [✓ Use VISA]
Resource : PXI9::13::INSTR

Figure 2-5: 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:

The following controls are shown in the Setup page:



GX7400 - Geotest's DC Power Supply (0x105)

Setup | About

Channel 1
Programmable 0-30V
5.12V
0.01A
Voltage : 5.12 Set
Current Limit : 2 Set
Standby [✓ On]

Channel 2
Programmable 0-30V
30.00V
0.00A
Voltage : 30 Set
Current Limit : 5 Set
Standby [✓ On]

● AC Power On

Figure 2-6: GX7400 Virtual Panel with Options Enabled

Channel Voltage: Sets the voltage for a channel. Enter a positive numeric value (nn.nn format) in the channel's Voltage text box and click Set. Disabled if channel is a fixed module.

Channel Current Limit: Sets the channel's maximum allowable current. Enter a positive numeric value (n.nn format) in the channel's Current Limit text box and click Set. Disabled if channel is a fixed module. . If current limit is set to zero and output is set to On, output voltage is not defined.

Channel Standby/On: Sets/display channel's Standby or On state. When set to ON the output voltage is measured and displayed. When in Standby the output voltage is zero.

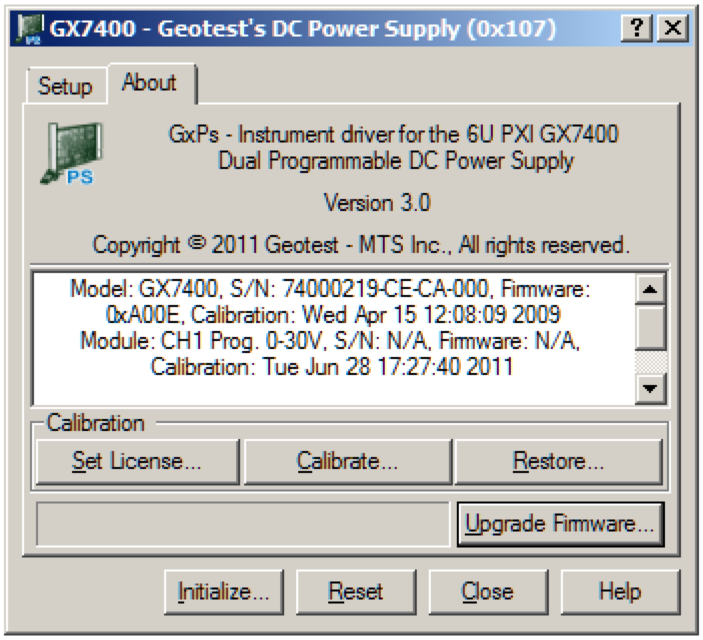
Recall Settings: Recalls and applies all installed channels parameters from the on-board EEPROM. The EEPROM stores the voltage, current limit and the On/Standby state for all installed channels when pushing the Store button.

Store Settings: Stores all installed channels parameters to the on-board EEPROM. The EEPROM will store the voltage, current limit and the On/Standby state for all installed channels.

AC Power On: Indicate whether the AC power switch is either set to On or Off position.

Virtual Panel

Clicking on the **About** tab will show the **About** page as shown in Figure 2-7



GxPs - Instrument driver for the 6U PXI GX7400
Dual Programmable DC Power Supply

Version 3.0

Copyright © 2011 Geotest - MTS Inc., All rights reserved.

Model: GX7400, S/N: 74000219-CE-CA-000, Firmware: 0xA00E, Calibration: Wed Apr 15 12:08:09 2009
Module: CH1 Prog. 0-30V, S/N: N/A, Firmware: N/A, Calibration: Tue Jun 28 17:27:40 2011

Figure 2-7: GX7400 Virtual Panel – About Page

The top part of the **About** page displays version and copyright of the GXPS driver. The bottom part displays the board summary, including the board Revision, calibration date and installed channels description. 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 Geotest 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.

Calibration (group box):

Set License… button opens the **License Setup** dialog, see "Chapter 5: In-System Calibration for details.

Calibration…button opens the calibration by user dialog. see "Chapter 5: In-System Calibration for details.

Restore…. button restores the specified channel's calibration data back to the manufacture data.

Chapter 3 - Setup and Installation

Getting Started

This section includes general hardware installation procedures for all GXPS power-supply instruments and installation instructions for the GXPS software. Before proceeding, please refer to the appropriate chapter to become familiar with the board being installed.

To Find Information on..	**Refer to..**
Hardware Installation	This Chapter
GXPS Driver Installation	This Chapter
Programming the board	Chapter 4
In-system Calibration	Chapter 5
GXPS Reference Functions	Chapter 6

Packing List

All GX7400 boards have the same basic packing list, which includes:

1. GX7400 Board
2. GX7400 Driver Disk

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 GX7400 instrument boards are designed for use with a 6U cPCI or PXI compatible chassis. The software is compatible with any computer system running a 32-bit or 64-bit Windows operating systems. In addition, Microsoft Windows Explorer version 4.0 or above is required to view the online help.

Each board requires three unoccupied 6U PXI bus slots.

Installation of the GXPS Driver

Before installing the board it is recommended to install the GXPS driver as described in this section. To install the GXPS driver follow the instruction described here:

1. Insert the Geotest CD-ROM and locate the **GXPS.EXE** setup program. If you computer's Auto Run is configured, when inserting the CD a browser will show several options, select the Geotest 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). You can also download the file from Geotest web site (www.geotestinc.com).
2. Run the GXPS setup and follow the instruction on the Setup screen to install the GXPS driver.

   Note: When installing under Windows NT/2000/XP/VISTA, you may be required to restart the setup after logging-in as a user with an Administrator privileges. This is required in-order to upgrade your system with newer Windows components and to install kernel-mode device drivers (HW.SYS and HWDEVICE.SYS) required by the GXPS 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 GXPS 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\Geotest\`GXPS.
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, GXPS menu located in the Windows Taskbar.

Overview of the GXPS Software

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

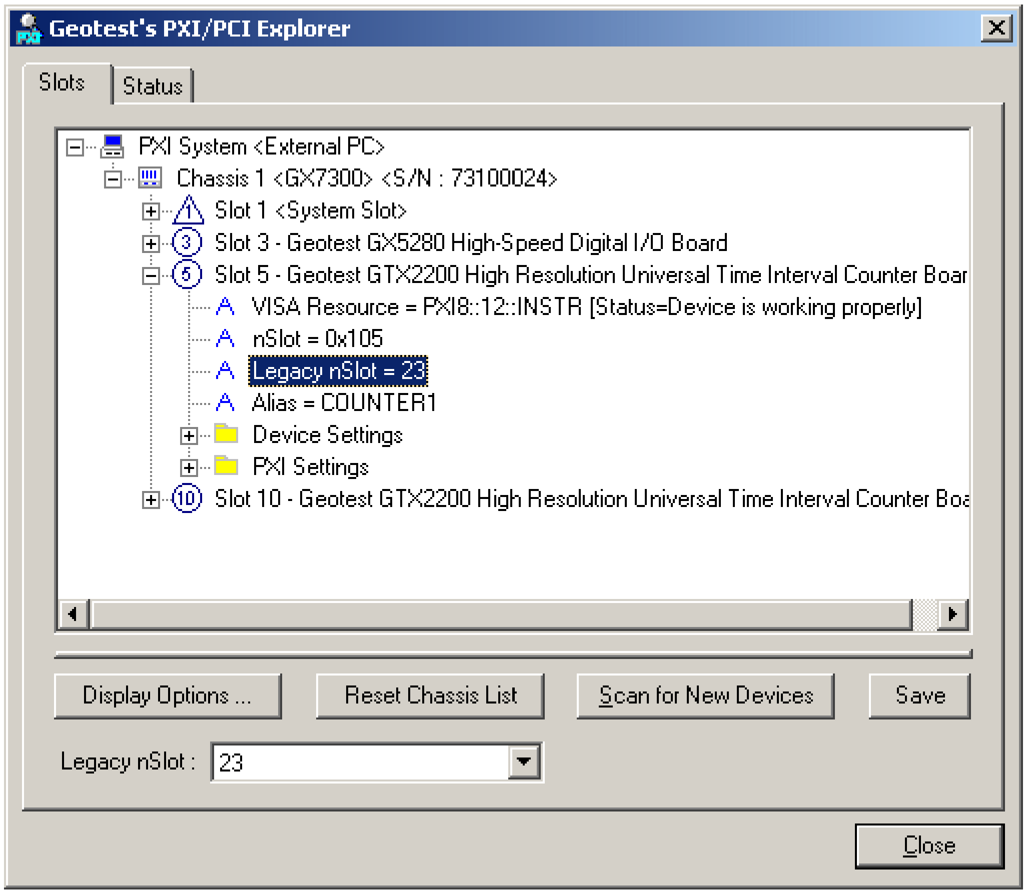
- **PXI/PCI Explorer applet** – use to configure the PXI chassis, controllers and devices. This is required for accurate identification of your PXI instruments later on when installed in your system. The applet configuration is saved to PXISYS.ini and PXIeSYS.ini that are used by Geotest instruments, 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.vxipnp.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 Geotest's PXI/PCI Explorer).
- **GXPS Panel** – use to configure the smart chassis features includes over-temperature behavior, control the system fans, measure slot temperature and system power supply usage and program trigger lines direction and connection between PXI bus segments.
- **GXPS driver** - a DLL (GXPS.DLL located in the Windows System folder) used to program and control the board.
- **Programming files and examples** – interface files and libraries for various programming tools, see later in this chapter for a complete list of files and development tools supported by the driver.
- **Documentation** – On-Line help and User's Guide.

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, **Geotest**, **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 Geotest 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 shown 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 Geotest or VISA drivers. Alias names can provide a way to address a PXI device using your logical name (e.g. “DMM1”). For more information regarding these numbers see the **GxPsInitialize** and **GxPsInitializeVisa** 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).


Geotest's PXI/PCI Explorer

```
PXI System <External PC>
  Chassis 1 <GX7300> <S/N : 73100024>
    Slot 1 <System Slot>
    Slot 3 - Geotest GX5280 High-Speed Digital I/O Board
    Slot 5 - Geotest GTX2200 High Resolution Universal Time Interval Counter Boar
      VISA Resource = PXI8::12::INSTR [Status=Device is working properly]
      nSlot = 0x105
      Legacy nSlot = 23
      Alias = COUNTER1
      Device Settings
      PXI Settings
    Slot 10 - Geotest GTX2200 High Resolution Universal Time Interval Counter Boa
```

Legacy nSlot : 23

Figure 3-1: PXI/PCI Explorer

Board Installation

Before you Begin

- Install the GXPS 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 GX7400 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. Install first the GXPS Driver as explain in the next section.
2. Turn off the PXI chassis and unplug the power cord.
3. Locate 3 empty consecutive PXI slots in the PXI chassis.
4. Place the module edges into the PXI chassis rails (top and bottom).
5. Carefully slide the PXI board to the rear of the chassis, make sure that the ejector handles are pushed **<u>out</u>** (as shown in Figure 3-2).

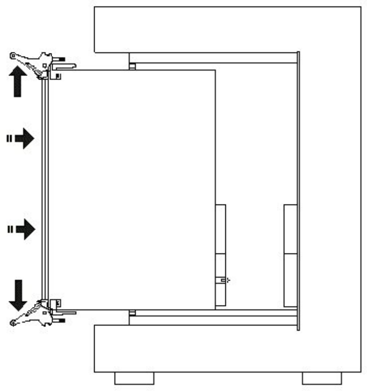


Figure 3-2: Ejector handles position during module insertion

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

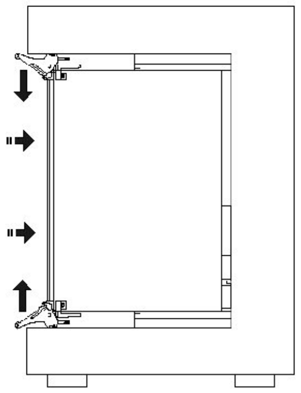


Figure 3-3: Ejector handles position after module insertion

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

Plug & Play Driver Installation

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

If another Geotest board software package was already installed, Windows will suggest using the driver information file: HW.INF. The file is located in your Program Files\Geotest\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 GXPS driver was not installed prior to the board installation), you may install the GXPS driver 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\Geotest\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 GXPS 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 GXPS driver.

Installation Directories

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

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

Name	Purpose / Contents
…\Geotest\GxPs	The GX7400 directory. Contains panel programs, programming libraries, interface files and examples, on-line help files and other documentation.
…\Geotest\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. GXPS Driver and example are copied to this directory only if ***ATEasy*** is installed to your machine.
…\Windows\System (Windows 9x/Me), or …\Windows\System32 when running Windows 2000/XP/Vista/7	Windows System directory. Contains the GXPS DLL driver and some upgraded system components, such as the HTML help viewer, etc.

GXPS Driver Files Description

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

Driver File and Virtual Panel

- GxPs.dll - 32-Bit MS-Windows DLL for applications running under Windows, 9x – Windows 7 or above.
- GxPsPanel.exe – An instrument front panel program for all GXPS supported boards.

Interface Files

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

- GXPS.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++
- GxPs.lib - Import library for GXPS.DLL (used when linking C/C++ application that uses GXPS.DLL).
- GsPsBC.lib - Import library for GXPS.DLL (used when linking Borland C/C++ application that uses GsPs.dll).
- GsPS.pas - interface file to support Borland Pascal Borland Delphi.
- GXPs.bas - Supports Microsoft Visual Basic 4.0, 5.0 and 6.0.
- GXPS.vb - Supports Microsoft Visual Basic .NET.
- GX7400.drv - ATEasy driver File for GX7400.GXPS Virtual Panel Program.
- GxPs.llb – LabView library.

GXPS On-line Help and Manual

GsPs.chm – On-line version of the GX7400 User's Guide. The help file is provided in a Windows Compiled HTML help file (.CHM). The file contains information about the GX7400 board, programming reference and panel operation.

GX7400.pdf – On line, printable version of the GX7400 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 GXPS manuals. You can view and/or print this file using the Windows NOTEPAD.EXE or 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:

- GxPsExampleC.cpp - Source file
- GxPsExampleC.ico - Icon file
- GxPsExampleC.rc - Resource file
- GxPsExampleC.vcproj - VC++ .NET project file
- GxPsExampleC.exe - Example executable

Microsoft Visual C++ 6.0 example files:

- GxPsExampleC.cpp - Source file
- GxPsExampleC.ico - Icon file
- GxPsExampleC.rc - Resource file
- GxPsExampleC.dsp - VC++ project file
- GxPsExampleC.exe - Example executable

Borland C++ example files:

- GxPsExampleC.cpp - Source file
- GxPsExample.ico - Icon file
- GxPsExampleC.rc - Resource file
- GxPsExampleC.bpr - Borland project file
- GxPsExampleC.exe - Example executable

Microsoft Visual Basic .NET example files:

- GxPsExampleVB.vb - Example form.
- GxPsExampleVB.resx - Example form resource.
- GxPsExampleVBapp.config - Example application configuration file.
- GxPsExampleVBAssebleyInfo.vb - Example application assembly file
- GxPsExampleVB.vbproj - Project file
- GxPsExampleVB.exe - Example executable

Microsoft Visual Basic 6.0 example files:

- GxPsExampleVB6.frm - Example form
- GxPsExampleVB6.frx - Example form binary file
- GxPsExampleVB6.vbp - Project file
- GxPsExampleVB6.exe - Example executable.

ATEasy driver and examples files (ATEasy Drivers directory):

- GsPs.prj - example workspace
- GX7400.prj - example project
- GX7400.sys - example system
- GX7400.prg - example program

Setup Maintenance Program

You can run the Setup again after GXPS 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 GXPS installation. The following options are available in Maintenance mode:

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

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

Follow the instruction on the screen until Setup is complete.

Chapter 4 - Programming the Power Supply

This chapter contains information about how to program the switching instruments using the GXPS driver. The GXPS driver contains functions to initialize, reset, and control the switching instruments. A brief description of the functions, as well as how and when to use them, is included in this chapter. Chapter 5 and the specific instrument User's Guide contain a complete and detailed description of the available programming functions.

The driver supports many development tools. Using these tools with the driver is described in this chapter. In addition, the GX7400 directory contains examples written for these development tools. Refer to Chapter 3 for a list of the available examples.

An example using the DLL driver with Microsoft Visual C++ 6.0 is included at the end of this chapter. Since the driver functions and parameters are identical for all operating systems and development tools, the example can serve as an outline for other programming languages, programming tools, and other GX7400 driver types.

The GXPS Driver

The GXPS driver is a 32 bit Windows DLL file: GXPS.DLL or a 64-bit DLL: GXPS64.DLL. The DLL is used with 32 or 64 bit applications running under Windows 95/98/ME and Windows NT/2000/XP/Vista or Windows 7. The DLL uses a device driver to access the board resources. The device driver HW.SYS or HW64.sys (on Windows NT/2000/XP/VISTA or Windows 7) or HW.VXD (on Windows 9x/Me) is installed by the setup program and is shared by other Geotest products (ATEasy, GXDIO, etc).

The DLL can be used with various development tools such as Microsoft Visual C++, Borland C++ Builder, Microsoft Visual Basic, Borland Pascal or Delphi, ATEasy 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 GX7400 driver with C/C++ development tools:

- Include the GXPS.H header file in the C/C++ source file that uses the GX7400 function. This header file is used for all driver types. The file contains function prototypes and constant declarations to be used by the compiler for the application.
- Add the required .LIB file to the projects. This can be import library GXPS.LIB for Microsoft Visual C++ and GXPSBC.LIB for Borland C++. 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 GX7400 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 4.0, 5.0 or 6.0 (for 32-bit applications), the user must include the GXPS.BAS to the project. For Visual Basic .NET use the GXPS.VB.

The file can be loaded using *Add File* from the Visual Basic *File menu*. The GXPS.BAS/.VB 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 GXPS.PAS to the project. The GXPS.PAS file contains a **unit** with function prototypes for the DLL functions. Include the GX7400 unit in the **uses** statement before making calls to the GX7400 functions.

Programming GXPS Boards Using ATEasy®

The GX7400 package is supplied with a separate ATEasy driver for each of board types. The ATEasy driver uses the GXPS.DLL to program the board. In addition, each driver is supplied with an example that contains a program and a system file pre-configured with the ATEasy driver. Use the driver shortcut property page from the System Drivers sub-module to change the PCI slot number before attempting to run the example.

Using commands declared in the ATEasy driver are easier to use than using the DLL functions directly. The driver commands will also generate exception that allows the ATEasy application to trap errors without checking the status code returned by the DLL function 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 *nHandle* parameter is omitted. The driver handles this parameter automatically. ATEasy uses driver logical names instead i.e. PRES1, PRES2 for GX7400.
- The *nStatus* parameter was omitted. Use the Get Status commands instead of checking the status. After calling a DLL function the ATEasy driver will check the returned status and will call the error statement (in case of an error status) to generate exception that can be easily trapped by the application using the **OnError** module event or using the **try-catch** statement.

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 commands are self-documented. Their syntax is similar to English. In addition, you may generate the commands from the code editor context menu or by using the ATEasy's code completion feature instead of typing them directly.

Using the GXPS driver functions

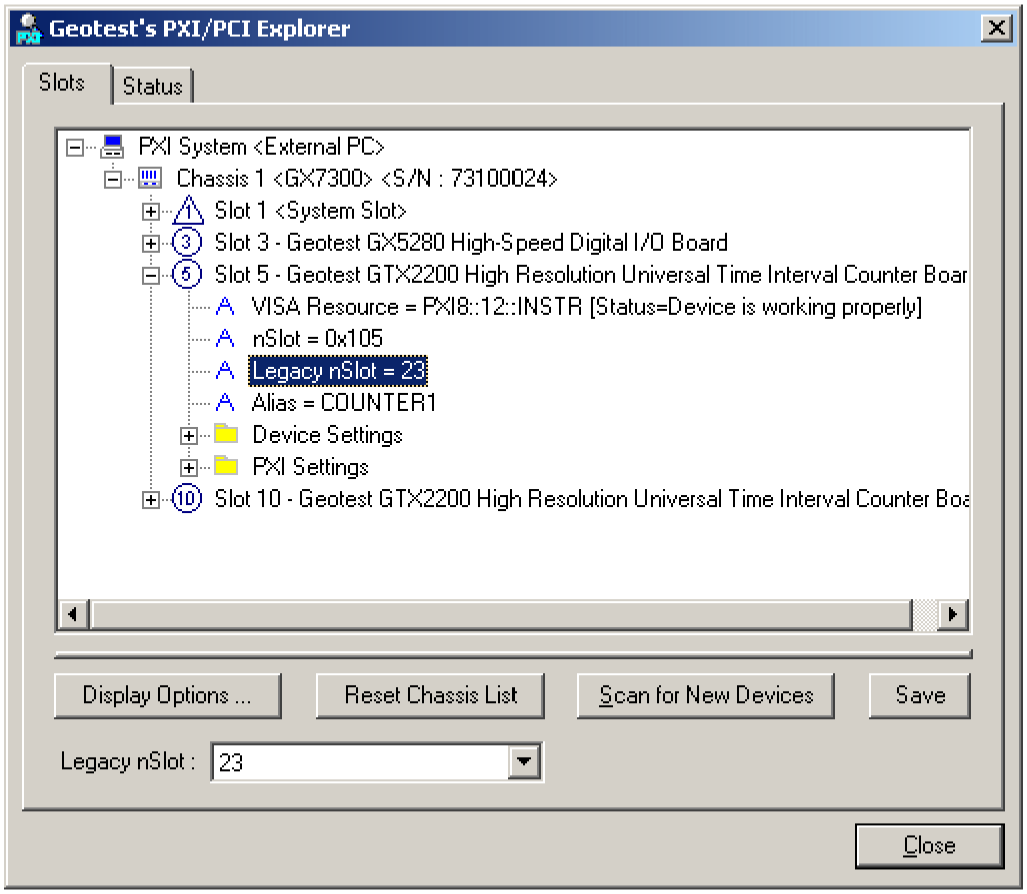
The GXPS driver contains a set of functions for the GX7400. Functions names that starts with the **GxPs** prefix applies to all GXPS boards (i.e. **GxPsGetDriverSummary**). The GXPS functions are designed with consistent set of arguments and functionality. All boards have a function that initializes the GXPS 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 GXPS driver supports two device drivers HW and VISA which are used to initialize, identify and control the board. The user can use the **GxPsInitialize** to initialize the board ‘s driver using HW and **GxPsInitializeVisa** to initialize using VISA. The following describes the two different methods used:

1. **Geotest’s HW** - the default device driver that is installed by the GXPS driver. To initialize and control the board using the HW use the **GxPsInitialize**(*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.



```
PXI System <External PC>
  Chassis 1 <GX7300> <S/N : 73100024>
    Slot 1 <System Slot>
    Slot 3 - Geotest GX5280 High-Speed Digital I/O Board
    Slot 5 - Geotest GTX2200 High Resolution Universal Time Interval Counter Boar
      VISA Resource = PXI8::12::INSTR [Status=Device is working properly]
      nSlot = 0x105
      Legacy nSlot = 23
      Alias = COUNTER1
      Device Settings
      PXI Settings
    Slot 10 - Geotest GTX2200 High Resolution Universal Time Interval Counter Boa
```

Legacy nSlot : 23

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). GXPS setup installs a VISA compatible driver for the GXPS board in-order to be recognized by the VISA provider. Use the GXPS function **GxPsInitializeVisa** (*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 Geotest **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 **GxPsInitialize** 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 **GxPsInitialize** or the **GxPsInitializeVisa** functions is a short integer (16 bits) number. It is used by the GXPS 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 **GxPsInitialize** function. But there is no need to destroy the handle. Calling **GxPsInitialize** 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 causes the driver to change all settings to their default state. The application software issue a Reset after the initializing the Counter, but a Reset can be issued any time. All counter boards have the **GxPsReset**(*nHandle, nStatus*) function. See the Function Reference for more information regarding the specific board.

Error Handling

All GXPS 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 **GxPsGetErrorString** function to return a string representing the error. The **GxPsGetErrorString** reference contains possible error numbers and their associated error strings.

Driver Version

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

Panel

Calling the **GxPsPanel** 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 **GxPsPanel** function is also used by the GXPSPANEL.EXE panel program that is 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 (GXPS.dll, GXPS64.dll and the HW device driver files) can be shipped with the application. Typically, the GXPS.dll 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 GXPS driver files (see Geotest\HW folder) or a standalone setup HW.exe. Alternatively, you can provide the GXPS.exe setup to be installed along with the board.

Sample Program

The following example demonstrates how to program the board using the C programming language under Windows. The example shows how to close or open a relay.

To run, Enter the following command line:

GxPsExample *<SlotNumber> <Command> <Module#> <Value>*

Where:

<SlotNumber>	Pci slot number where the board is installed, e.g.: 3
<command>	Specify whether you want to set current limit or voltage • C=current limit • V=Voltage • SUM= Displays power supply information
<Module#>	1 – output 1 2 - output 2
<Value>	Value to set

Sample Program Listing

```
/***************************************************************************

   FILE      : GxPsExampleC.cpp

   PURPOSE      : WIN32/LINUX example program for GX2200 boards
              using the GxPs driver.

   CREATED      : Mar 2002

   COPYRIGHT    : Copyright 2002-2011 GEOTEST - MTS Inc.

   COMMENTS :

   To compile the example:

   1. Microsoft VC++
      Load GxPsExampleC.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 GxPsExampleC.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 -fGxPsExampleC.mk [CFG=Release[64] | Debug[64]] [rebuild |
            clean]

***************************************************************************/
#ifndef __GNUC__
#include "windows.h"
#endif
```

```
#include "GxPs.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

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

//****************************************************************************
//    DisplayMsg
//****************************************************************************
void DisplayMsg(PCSTR lpszMsg)
{
#ifndef __GNUC__
   MessageBeep(0);
   MessageBox(0, lpszMsg, "GxPs example program", MB_OK);
#else
    printf("\r\nGxPs 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 "
      "assign current limit/voltage to a certain channel\r\n"

      "Usage:\r\n"
        "GxPsExampleC <slot|address> <command> [<channel#> <Value>]"
      "\r\n\r\nWhere:\r\n"

        "<slot> - Under Windows: PCI/PXI slot number as shown by the PXI \
            explorer\r\n"
        "<address> - Under Linux: Bus/device as displayed with lspci \
            utility\r\n"
        "<command> C | V | SUM: specify whether you want to set \
            current limit, voltage or display PS summary\r\n"

        "<channel#> channel number : 1 or 2\r\n"
      "<value>=value to set\r\n"

      "\nExample 1 assigning 20.5V to channel 1 using 1 as slot number\r\n"
      "GxPsExampleC 1 V 1 20.5 \r\n"

      "\nExample 2 setting channel 2's current limit to be 1.2A using 1 \
            as slot number:\r\n"
      "GxPsExampleC 1 C 2 1.2 \r\n"

      "\nExample 3 assigning 0V to channel 2 using 1 as slot number:\r\n"
      "GxPsExampleC 1 V 1 0 \r\n"

      "\nExample 4 display board summary:\r\n"
      "GxPsExampleC 1 SUM \r\n"

      "\r\nTo change command line under Windows \r\n"
      "Select File/Properties from Program Manager \r\n"
      "Menu and change the command line edit box as \r\n"
      "shown above."
      );
   exit(1);
}

//***************************************************************************
//      CheckStatus
//***************************************************************************
void CheckStatus(SHORT nStatus)
{
   CHAR    sz[1024];

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

//***************************************************************************
//  This main function receives four user command line parameters:
//
//  GxPs board slot number (e.g. 1)
//  command specify whether you want to set current
//  limit or voltage, C=current limit, V=voltage
//  Channel# 1 or 2
//  Value to set
//***************************************************************************
int main(int argc, char **argv)
{
   SHORT nSlotNumber;      // Slot number
   char *   pszOperation;     // Board Operation (C/V/SUM)
   SHORT nChannel;           // Channel number
   DOUBLE   dVal,dVal1,dVal2;
   SHORT nHandle;        // Board handle
   SHORT nStatus;        // Returned status
   char  acBuf[64];
    char    sz[1024];

   // ** Check number of arguments rcvd
   if (argc < 3) DisplayUsage();

   // ** Parse command line parameters
   nSlotNumber=(SHORT)strtol(*(++argv), NULL, 0);
   pszOperation=*(++argv);

   // ** Process operation (Current/Voltage)
    __strupr(pszOperation);
   if (!strcmp(pszOperation, "C") && !strcmp(pszOperation, "V") &&
        !strcmp(pszOperation, "SUM"))
      DisplayUsage();

   // ** Initialize, and retrieve a handle.
   GxPsInitialize (nSlotNumber, &nHandle, &nStatus);
   CheckStatus(nStatus);

   switch (*pszOperation)
   {   case 'C':
         nChannel=(SHORT)strtol(*(++argv), NULL, 0);
         dVal=atof(*(++argv));
         GxPsSetCurrentLimit(nHandle, nChannel, dVal, &nStatus);
         CheckStatus(nStatus);
         break;
      case 'V':
         nChannel=(SHORT)strtol(*(++argv), NULL, 0);
         dVal=atof(*(++argv));
         GxPsSetVoltage(nHandle, nChannel, dVal, &nStatus);
         CheckStatus(nStatus);
         break;
      case 'S':
            // print board summary
            GxPsGetBoardSummary(nHandle, sz, sizeof sz, &nStatus);
          CheckStatus(nStatus);
          printf("Board Summary: %s.\n", sz);
   }

    // print other PS settings
    // module 1
   GxPsGetVoltage(nHandle,1,&dVal,&nStatus);
   GxPsGetCurrent(nHandle,1,&dVal1,&nStatus);
   GxPsGetCurrentLimit(nHandle,1,&dVal2,&nStatus);
   sprintf(acBuf,"\nModule 1: %3.2fV, %3.2fA, current limit: %3.2fA",dVal,
        dVal1, dVal2);
   DisplayMsg (acBuf);
    // module 2
   GxPsGetVoltage(nHandle,2,&dVal,&nStatus);
   GxPsGetCurrent(nHandle,2,&dVal1,&nStatus);
   GxPsGetCurrentLimit(nHandle,2,&dVal2,&nStatus);
   sprintf(acBuf,"\nModule 2: %3.2fV, %3.2fA, current limit: %3.2fA",dVal,
        dVal1, dVal2);
   DisplayMsg (acBuf);
   return 0;
```

```
}

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

Chapter 5 - In-System Calibration

Geotest offers Automatic In-System Calibration software for the power supplies boards. The In-System Calibration requires purchasing a license from Geotest. The Automatic In-System Calibration can be performed through the virtual panel or through the driver function calls. The support for the Automatic In-System Calibration needs to be activated by running the **GXPS-VCAL** license setup in order to be used by the virtual panel and the driver functions.

The calibration program provides an easy-to-use Automatic In-System Calibration procedure, which uses a calibration front panel or Driver function calls. This allows for calibration without the need to remove the power supply board from a rack or system. The calibration virtual front panel provides an interactive step-by-step process that ensures proper and accurate calibration.

Note – It is recommended to calibrate the power supply every 12 months.

The following provides information regarding:

- Installation of the **GXPS-CAL** license setup.
- Calibration required instruments for the calibration.
- Connections diagram.
- Running calibration from the virtual panel.
- Calibration Example program (running calibration using API calls).

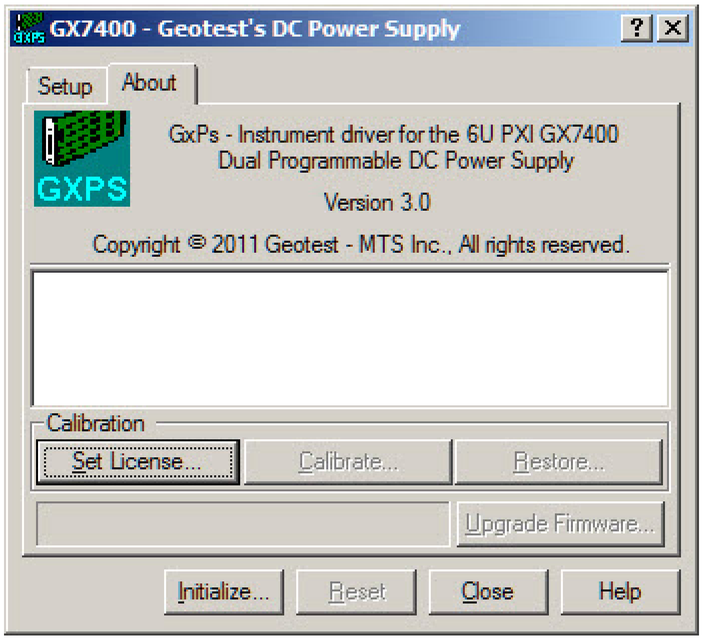
Calibration Interval

The instrument should be calibrated on a regular interval determined by the measurement accuracy requirements of your application. A 1-year interval is adequate for most applications. Accuracy specifications are warranted only if adjustment is made at regular calibration intervals. Accuracy specifications are not warranted beyond the 1-year calibration interval. Geotest does not recommend extending calibration intervals beyond 2 years for any application.

In-System Calibration license setup

The **GxPs Calibration** license string needs to be installed in order to activate and enable calibration option through the virtual front panel or through the driver function calls. The following procedure describes how to setup the calibration license:

1. Open the Panel application, **About** page:



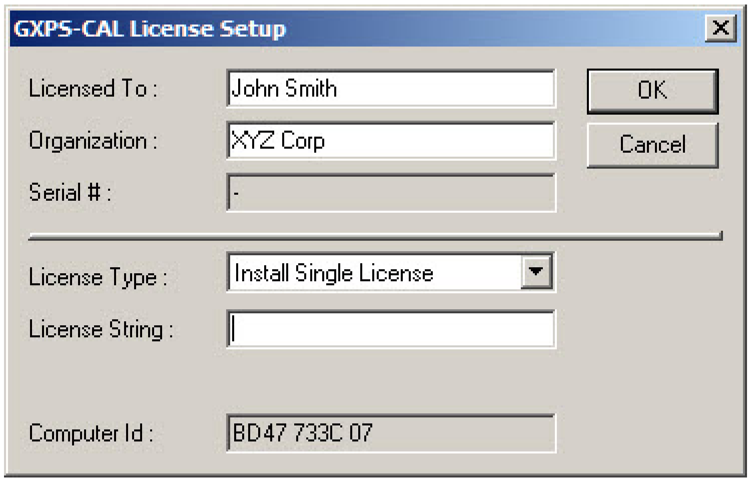
GxPs - Instrument driver for the 6U PXI GX7400
Dual Programmable DC Power Supply

Version 3.0

Copyright © 2011 Geotest - MTS Inc., All rights reserved.

Figure 5-1: GX7400 Virtual Panel About Page

2. Click on the **Calibration License…** button to open the **License Setup** dialog:



Licensed To : John Smith
Organization : XYZ Corp
Serial # : -
License Type : Install Single License
License String :
Computer Id : BD47 733C 07

Figure 5-2: GXPS-CAL License Setup Dialog

3. Request a license from Geotest. You may need to create or log into your magic account at http://www.geotestinc.com/magic. Then create an incident requesting a license specifying your product serial

number (as it appears on your purchase order or packing list) and the Computer Id shown in the **License Setup** dialog (in this example "C338 ECEE 05".).

4. Once you receive the license string (by email), enter it to the edit box in the **License Setup** dialog along with your name and organization and click **OK**.

You can now 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, Geotest and GXCNT menu located in the Windows Taskbar. In the virtual panel click the "Initialize" button, then click on the About tab. The Calibration By User.. button should be enabled.

Calibration required instruments

In order to calibrate the power supply board the following list of instruments is required.

- Calibrated multimeter with 61/2 digits.
- Alligator clip leads with AWG16.
- Calibration Recommended Load Range

The following table lists the Calibration Recommended Load Range in order to ensure best results. Best results are achieved when load values are close to the half of the module's total output power.

The load must either be cooled properly, or thermal-coefficient resistors must be used.

Module type	**Output Voltage range**	**Calibration Recommended Load Range**
GT7430	0-30 V @ 5 A Programmable Power Supply module	12Ω ±5% @ 75Watt
GT7460	0-60 V @ 2.5 A Programmable Power Supply module	48Ω ±5% @ 75Watt
GT7420-03	3.3V @10A fixed PS module	1.65Ω ±5% @ 16.5Watt
GT7420-05	5V @10A fixed PS module	1Ω ±5% @ 25Watt
GT7420-12	12V @8.4A fixed PS module	2.85Ω ±5% @ 50Watt
GT7420-15	15V @7.6A fixed PS module	4Ω ±5% @ 56Watt
GT7420-28	28V @3.6A fixed PS module	15.5Ω ±5% @ 50Watt

Connections diagram

The power supply outputs are routed through an on-board isolation relay, and include remote sensing capability. The following diagram demonstrates the GX7400 connector with a high-power load. The positive output voltage should be connected to the load with the positive sense line: the negative output voltage should be connected to the load with the negative sense line.

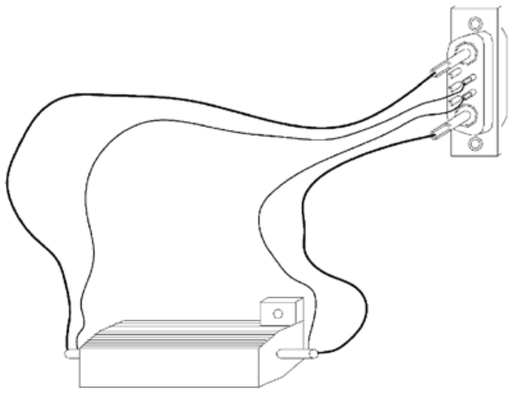


Figure 5-3: GX7400 connector with a high-power load

The following diagram shows the GX7400 front panel connectors.

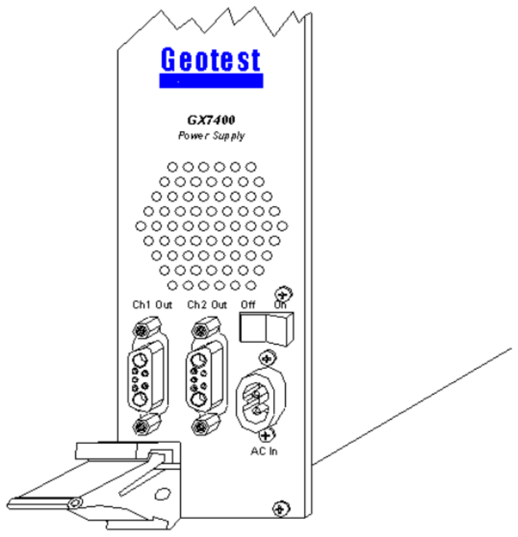


Figure 5-4: GX7400 front panel connectors

The following diagram shows the front panel output connector of Channel 1 connected to the load.

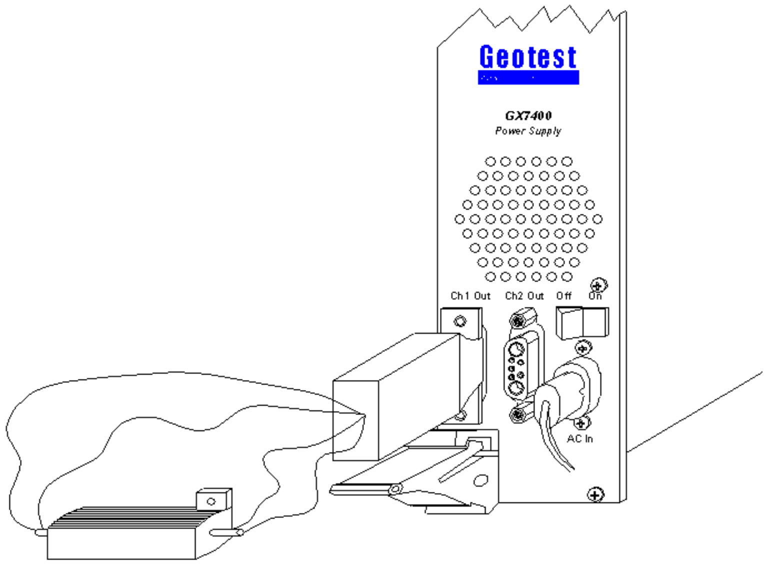


Figure 5-5: GX7400 Channel 1 connected to the load

Running In-System calibration from the virtual panel

The **GXPS** includes a virtual panel program, which provides full access to the various configuration settings and operating modes. To open the virtual panel application, select **GX7400 Panel** from the **Geotest**, **GXPS** menu under the **Start** menu. The GX7400 virtual panel opens as shown here:

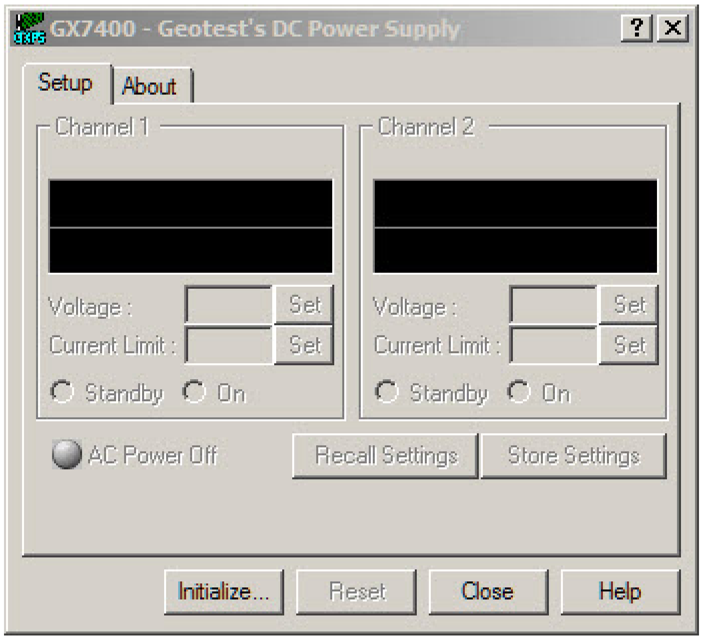
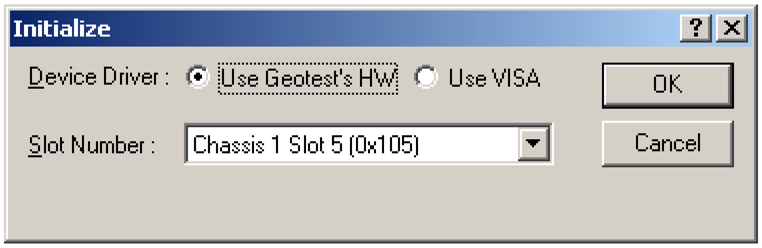


Figure 5-6: GX7400 Virtual Panel (not Initialized)

The function of the panel button controls are shown below:

Initialize allows the user to associate the front panel with a specific PXI counter. The settings of the chosen counter **will not change**. Then panel will reflect the current settings of the counter after the Initialize dialog closes.

The Initialize button opens an Initialize dialog window. The user may select the slot where the PXI slot location for a PXI counter. Slot 0x105 has a PXI counter installed and is available for selection in the Initialize dialog box as shown in Figure 5-7:



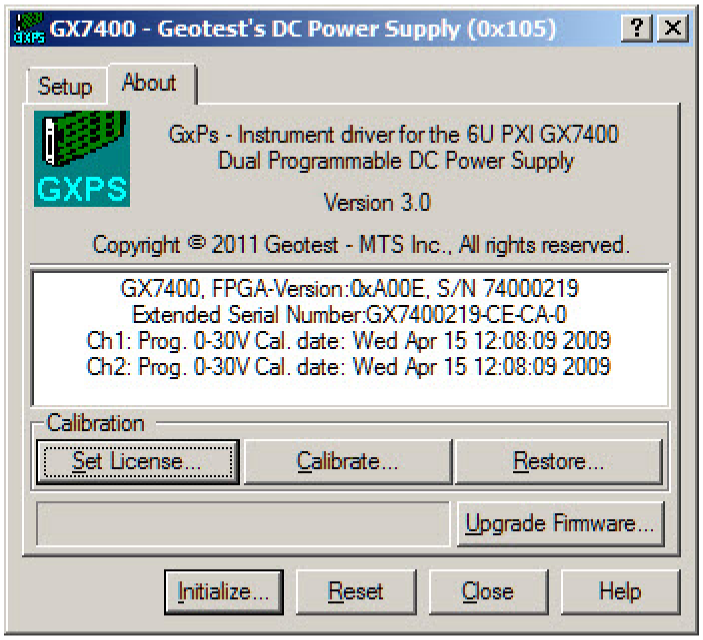
Device Driver : [✓ Use Geotest's HW] Use VISA

Slot Number : Chassis 1 Slot 5 (0x105)

Figure 5-7: Initialize Dialog Box

The **Slot Number:** in the Initialize dialog box refers to the PXI slot in which the Gx7400 board is installed. Select the slot from the drop down list. The list displays the slots where Gx7400 boards are installed. The specified slot number can also be reviewed or set by using the **PXI/PCI Explorer** applet located in the Windows Control Panel. Select the board slot number and click **OK** to initialize the driver for the specified board.

After the board is initialized the panel is enabled and will display the current setting of the board, click on the **About** tab. The **Calibrate…** button should be enabled.



GxPs - Instrument driver for the 6U PXI GX7400
Dual Programmable DC Power Supply
Version 3.0
Copyright © 2011 Geotest - MTS Inc., All rights reserved.

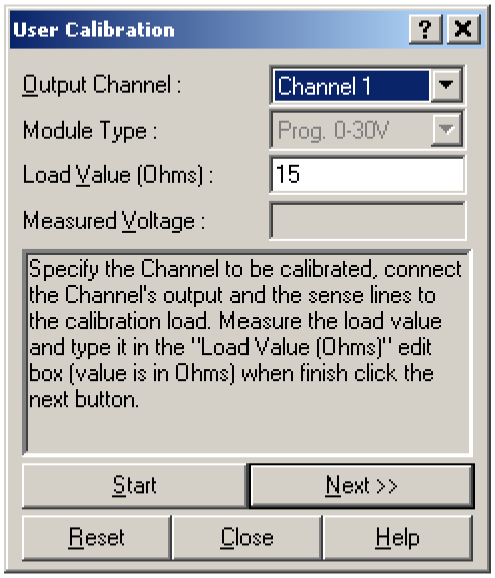
GX7400, FPGA-Version:0xA00E, S/N 74000219
Extended Serial Number:GX7400219-CE-CA-0
Ch1: Prog. 0-30V Cal. date: Wed Apr 15 12:08:09 2009
Ch2: Prog. 0-30V Cal. date: Wed Apr 15 12:08:09 2009

Figure 5-8: GX7400 Virtual Panel – About Page

Clicking on the "**Calibrate…**" button will open the User Calibration dialog as follows:

Front Panel Calibration dialog

The Front Panel Calibration dialog applies only to Nonvolatile Calibration, Figure 5-9 shows the **User Calibration** dialog settings:



Output Channel : Channel 1
Module Type : Prog. 0-30V
Load Value (Ohms) : 15
Measured Voltage :

Specify the Channel to be calibrated, connect the Channel's output and the sense lines to the calibration load. Measure the load value and type it in the "Load Value (Ohms)" edit box (value is in Ohms) when finish click the next button.

Figure 5-9: Calibration Dialog

Output Channel (dropdown box): Sets/displays the channel to be calibrated.

Module Type (dropdown box): Sets/displays the specified module type.

Load Value (Edit Box): Sets/displays the load value. The load value should be as close as possible to the recommended value.

Measured Voltage (Edit Box): Sets/displays the measured voltage at the specified channel output value (value provided by user).

Start Button: Click this button if settings need to be changed at any time during calibration.

Next Button: Advances to the next step in the calibration process.

Close Button: Closes the Calibration dialog.

Reset Button: Resets the GX7400 board (useful if need to disconnect the power supply immediately).

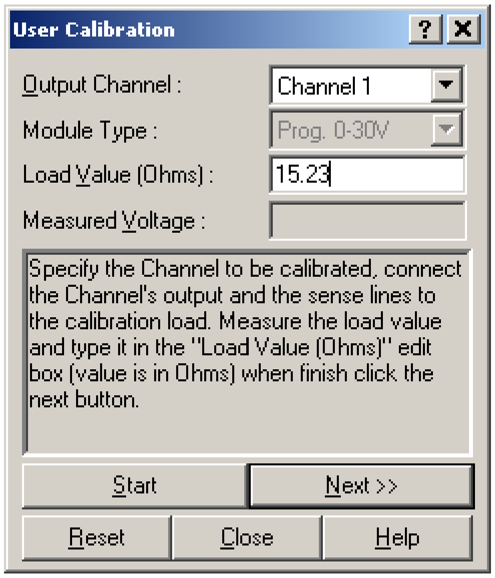
Help Button: Opens the Calibration topics from the GXPS on-line help.

Calibration dialog tutorial

The following tutorial demonstrates the calibration process. It uses the Front Panel Calibration dialog to calibrate Channel A with Programmable 30V.

Step 1: Pre-calibration settings

This step detects the specified Channel Module type and sets the recommended load. Specify the output channel to be calibrated with the dropdown menu and then connect the Channel's output and sense lines to the calibration load. Measure the load value (in ohms) and type it in the "Load Value (Ohms)" edit box. Click **Next**.



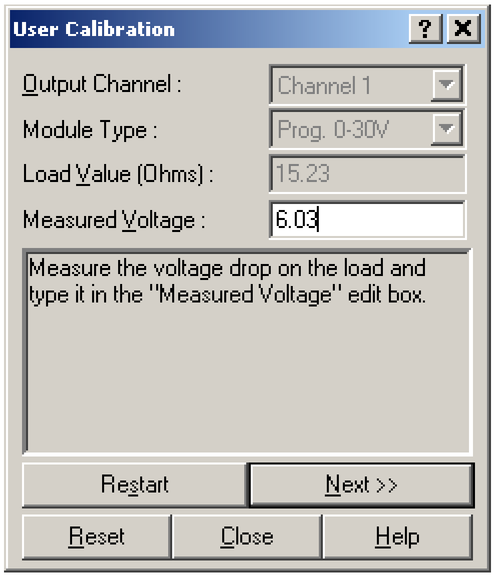
Output Channel : Channel 1
Module Type : Prog. 0-30V
Load Value (Ohms) : 15.23
Measured Voltage :

Specify the Channel to be calibrated, connect the Channel's output and the sense lines to the calibration load. Measure the load value and type it in the "Load Value (Ohms)" edit box (value is in Ohms) when finish click the next button.

Figure 5-10: Calibration Dialog - Step 1: Pre-calibration settings

Steps 2 through 9: Measure voltage

Type the measured voltage at the output channel into the edit box and click **Next** (**Enter**), click **Restart** to restart the calibration process or the **Close** button to terminate the calibration procedure.



User Calibration

Output Channel : Channel 1

Module Type : Prog. 0-30V

Load Value (Ohms) : 15.23

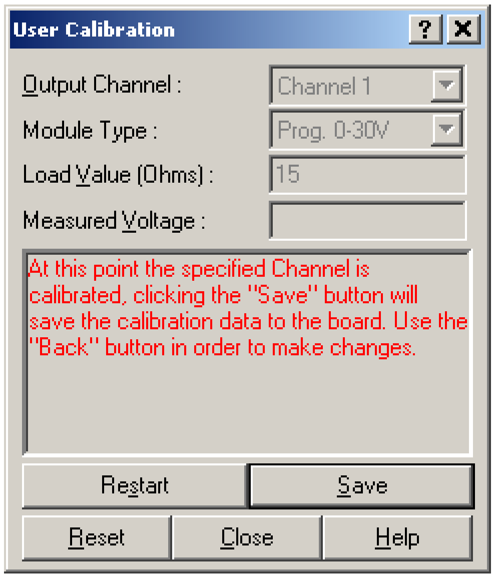
Measured Voltage : 6.03

Measure the voltage drop on the load and type it in the "Measured Voltage" edit box.

Figure 5-11: Calibration Dialog - Steps 2 through 9: Measure voltage

Step 10: Save Channel's Calibration data

The Channel is calibrated and the calibration data is ready to be saved to the on-board EEPROM. Click **Save** (or **Enter**) to save the Channel calibration data to the on-board EEPROM. To change any previous settings, click **Restart**. To terminate the calibration procedure, click **Cancel.**



User Calibration

Output Channel : Channel 1

Module Type : Prog. 0-30V

Load Value (Ohms) : 15

Measured Voltage :

At this point the specified Channel is calibrated, clicking the "Save" button will save the calibration data to the board. Use the "Back" button in order to make changes.

Figure 5-12: Calibration Dialog - Step 10: Save Channel's Calibration data

In-System Calibration Example program

The following example demonstrates how to calibrating the GX7400 boards using the GXPS driver and C programming language under Windows, to run, enter the following command line:

To run, Enter the following command line:

GxPsExample <*PciSlot*> <Channel> <sOperation>

Where:

<*PciSlot*>	Pci slot number where the board is installed, e.g.: 3
<Channel>	Channel 1 or 2
<sOperation>	specify whether you want to calibrate the channel or to restore the channel's calibration data.

Sample User Calibration Program Listing

```
/****************************************************************************

    FILE     : GxPsExampleUserCal.C

    PURPOSE  : Using the GxPs driver User Calibration functions.

    COPYRIGHT: Copyright 2005 GEOTEST Inc.

    COMMENTS :

   To compile the Windows 32 bit DLL example:

   To compile the example:

   1. Microsoft VC++
      Load GxPsExampleC.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 GxPsExampleC.bpr from the Project/Open
         Project... menu
      Select Project/Build all from the menu

****************************************************************************/

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "GxPs.h"

//***************************************************************************
//      DispMsg
//***************************************************************************
void DispMsg(LPSTR lpszMsg)
{
   //if windows use MessageBox
   MessageBeep(0);
   MessageBox(0, lpszMsg,"GxPs", MB_OK);
   return;
}

//***************************************************************************
//      DispUsage
//***************************************************************************
void DispUsage(void)
{
   DispMsg(
      "\r\nThis example shows how to use the GxPs"
      "GxPs driver User Calibration functions.\r\n"

      "Usage:\r\n"
      "GxPsExampleUserCal <SlotNumber> <Channel> <sOperation>"
      "\r\n\r\nWhere:\r\n"
      "<SlotNumber>"
      "\t(example : 1)\r\n"
      "<Channel> 1 or 2\r\n"
      "<sOperation> specify whether you want to calibrate the channel \
            or to\r\n"
      "restore the channel's calibration data to the factory \
            calibration.\r\n"
      "Where Command: CALIBRATE=Calibrate Channel, RESTORE=Voltage\r\n"

      "\nExample1 calibrating channel 1 using 1 as slot number\r\n"
      "GxPsExampleUserCal 1 1 CALIBRATE\r\n"

      "\r\nTo change command line under Windows \r\n"
      "Select File/Properties from Program Manager \r\n"
      "Menu and change the command line edit box as \r\n"
      "shown above."
      );
   exit(1);
}

//***************************************************************************
//      CheckStatus
//***************************************************************************
void CheckStatus(SHORT nStatus)
{
   CHAR    sz[256];

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

//***************************************************************************
//  This main function receives four user command line parameters:
//  - GxPs board slot number (e.g. 1)
//  - Channel# 1 or 2
//  - Command CALIBRATE or RESTORE
//***************************************************************************
int main(int argc, char **argv)
{
```

```
SHORT nSlotNumber;      // Slot number
SHORT nChannel;         // Channel number
FLOAT fVoltage=0.0;     // Measured voltage
FLOAT fLoad=0.0;        // Measured load
SHORT nHandle;       // Board handle
SHORT nStatus;       // Returned status
char* sOperation;       // Board Command Operation
SHORT nStepNum;         // Step number

// ** Check number of arguments rcvd
if (argc < 4) DispUsage();

// ** Parse command line parameters
nSlotNumber=(SHORT)strtol(*(++argv), NULL, 0);
nChannel = (SHORT)strtol(*(++argv), NULL, 0);
sOperation = strupr(*(++argv));

// ** Initialize, and retrieve a handle.
GxPsInitialize (nSlotNumber, &nHandle, &nStatus);
CheckStatus(nStatus);

if(!strcmp(sOperation, "CALIBRATE"))
{  /*
   The following provides a step-by-step Calibration example program
   Initializing the specified Channel for calibration, detects the
     specified Channel Module type (unless Fixed type), and pass the
     measured load value (in ohms). See the User's Guide for recommended
     loads values.
   */
   printf("Enter the measured load value in Ohms\n");
   scanf( "%f", &fLoad);
   GxPsUserCalSetup(nHandle, nChannel, fLoad, NULL, &nStatus);
   CheckStatus(nStatus);
   for (nStepNum=GXPS_USER_CAL_STEP1; nStepNum<=GXPS_USER_CAL_STEP9;
         nStepNum++)
   {
      // Set the board for Measurement
      GxPsUserCalSetupForMeasurement(nHandle, nStepNum, &nStatus);
      CheckStatus(nStatus);
      printf("Calibration Step number %i, enter the measured voltage \
                value\n", nStepNum);
      scanf( "%f", &fVoltage);
      GxPsUserCalWriteMeasuredVal(nHandle, nStepNum, fVoltage,
             &nStatus);
      CheckStatus(nStatus);
   }

   // Store the Channel's Calibration Data to the on-board EEPROM.
   GxPsUserCalStoreCalibrationData(nHandle, &nStatus);
   CheckStatus(nStatus);
}
else if (!strcmp(sOperation, "RESTORE"))
{
   // Restore the specified channel calibration data back to the
     // factory calibration.
   GxPsUserCalRestoreFactoryCalibration(nHandle, nChannel, &nStatus);
}
```

```
   else
      DispUsage();

   return 0;
}

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

Chapter 6 - Functions Reference

Introduction

The functions reference chapter organizes the list of GXPS driver functions in an 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 and parameter syntax follow 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 **GxPsInitialize** 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 non-zero on error. The description of the error is available using the **GxPsGetErrorString** function or by using a predefined constant, defined in the driver interface files: GXPS.H, GXPS.BAS, GXPS.VB, GXPS.PAS or GXPS.DRV.

Parameter Prefix Names

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*

Functions by Category

The following list is a summary of functions available for the GX7400:

Driver Functions	**Description**
General	
GxPsInitialize	Initializes the GX7400 driver for the specified slot.
GxPsInitializeVisa	Initializes the GX7400 driver for the specified slot using VISA resources.

Driver Functions	**Description**
GxPsPanel	Opens a virtual panel used to interactively control the GX7400.
GxPsReset	Resets the GX7400 board to its default settings.
GxPsGetDriverSummary	Returns the driver name and version.
GxPsGetErrorString	Returns the error string associated with the specified error number.
GxPsGetExtendedSerialNumber	Returns the Extended board's Serial Number information.
Functions	
GxPsGetBoardSummary	Returns the board summary from the EEPROM
GxPsGetCalibrationDate	Returns the board calibration date from the on-board EEPROM.
GxPsGetCalibrationInfo	Returns the calibration information.
GxPsGetChannelCalibrationDate	Returns the specified channel calibration date from the on-board EEPROM.
GxPsGetCurrent	Returns the specified channel's current.
GxPsGetCurrentLimit	Returns the specified channel's current limit.
GxPsGetDriverSummary	Returns the driver description string and version number.
GxPsGetErrorString	Returns the error string as specified by the error number.
GxPsGetStandby	Returns the specified channel's state (Standby/On).
GxPsGetState	Returns the specified channel's status (Inhibit/Fail).
GxPsGetType	Returns the specified channel's module configuration.
GxPsGetVoltage	Returns the specified channel's voltage.
GxPsRecallSettings	Recalls and apply the specified channel's parameters from the on-board EEPROM.
GxPsSetCurrentLimit	Sets the specified channel's current limit.
GxPsSetStandby	Sets the specified channel to Standby or On.
GxPsSetStandbyImmediate	Sets the specified channel to Standby or On in the shortest time possible.
GxPsSetVoltage	Sets the specified channel's voltage.
GxPsSetVoltageAndWaitUntilReady	Sets the specified channel's voltage. The function return when the specified voltage settings was verified or in case of a timeout.
GxPsStoreSettings	Store the specified channel's parameters to the on-board EEPROM.
User Calibration Functions	
GxPsUserCalReadMeasuredVal	Returns the specified calibration step voltage value as was set by the user.
GxPsUserCalRestoreFactoryCalibration	Restore the specified channel's calibration data back to the manufacture data.
GxPsUserCalSetup	Initialize the specified channel's for calibration and sets the load value.
GxPsUserCalSetupForMeasurement	Sets the currently under calibration channel output voltage according to the specified calibration step.
GxPsUserCalStoreCalibrationData	Stores the calibration data to the on-board EPROM.
GxPsUserCalWriteMeasuredVal	Sets the specified calibration step voltage value as was measured by the user.

Driver Functions	**Description**
Upgrade firmware functions	
GxPsUpgradeFirmware	Upgrades the board’s firmware.
GxPsUpgradeFirmwareStatus	Monitor the firmware upgrade process.

GxPsGetBoardSummary

Purpose

Returns the board summary from the EEPROM.

Syntax

GxPsGetBoardSummary (*nHandle*, *pszBoardSum*, *nSumMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
pszBoardSum	LPSTR	Buffer to contain the returned board info string (null terminated string)
nSumMaxLen	SHORT	Size of the buffer contains the error string.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board info string reads from the on-board EEPROM the following data in that order:

- Name
- Hardware Version
- Firmware version
- Serial Number
- Calibration date
- Channels description

For example the returned string look as follow:

```
“Model: GX7400, S/N: 74000219-CE-CA-000, Firmware: 0xA00E, Calibration: Wed
Apr 15 12:08:09 2009
Module: CH1 Prog. 0-30V, S/N: N/A, Firmware: N/A, Calibration: Tue Jun 28
17:27:40 2011
Module: CH1 Prog. 0-30V, S/N: N/A, Firmware: N/A, Calibration: Wed Apr”
```

Example

The following example returns the board summary.

```
SHORT nHandle, nStatus;
CHAR szBoardSum[256];
GxPsGetBoardSummary(nHandle, szBoardSum, sizeof (szBoardSum), &nStatus);
```

See Also

GxPsGetErrorString, **GxPsInitialize**

GxPsGetCalibrationDate

Purpose

Returns the date when the board was calibrated by the manufacture from the on-board EEPROM.

Syntax

GxPsGetCalibrationDate (*nHandle, pszCalibrationDate, nSumMaxLen, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
pszCalibrationDate	LPSTR	Buffer to contain the returned calibration date (null terminated string)
nSumMaxLen	SHORT	Size of the buffer contains the error string.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calibration date has the mm/dd/yyyy format, for example:

"03/20/2002"

Example

The following example returns the board calibration date.

```
SHORT nHandle, nStatus;
CHAR szCalibrationDate [256];
GxPsGetCalibrationDate (nHandle, szCalibrationDate, sizeof (szCalibrationDate), &nStatus);
```

See Also

GxPsGetCalibrationInfo, **GxPsGetBoardSummary, GxPsGetErrorString**, **GxPsInitialize**

GxPsGetCalibrationInfo

Purpose

Returns the calibration information.

Syntax

GxPsGetCalibrationInfo (*nHandle, pszCalibrationInfo, nInfoMaxLen, pnDaysUntilExpire, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a counter board.
pszCalibrationInfo	PSTR	Buffer to contain the returned board's calibration information (null terminated) string.
nSumMaxLen	SHORT	Size of the buffer to contain the error string.
pnDaysUntilExpire	PSHORT	Returns the number of days until or from expiration, if number is > 0 then calibration is current otherwise past due.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The returned board's calibration information has the following fields:

Model: model number, e.g. "GX7400"

Serial Number: serial number, e.g. 216

Control Number: Geotest control number, e.g. "*-CH-CB-0"

Production Calibration Date: Wed Oct 24 12:30:25 2010

Calibration Date: Wed Oct 24 12:31:58 2010

Recommended Interval: 1 year

Next Calibration Date: Fri Oct 24 12:31:58 2011

Status: calibration status can be either "Expired" followed by the number of days past expiration or "Current" followed by number of days until expire.

Calibration License: can be either "Installed" with the calibration license number or "Not Installed".

Example

The following example returns the board’s calibration information string:

```
SHORT   nStatus;
char    szCalibrationInfo[1024];
BOOL    bExpired;
GxCntGetCalibrationInfo(nHandle, szCalibrationInfo, sizeof szCalibrationInfo,
&bExpired,&nStatus);
```

```
szCalibrationInfo string printout:
```

```
Model: Gx7400
Serial Number: 74000219
Control Number: *-CE-CA-0
Production Calibration Date: Wed Apr 15 12:08:09 2009
Calibration Date: Wed Apr 15 12:08:09 2009
Recommended Interval: 1 year
Next Calibration Date: Thu Apr 15 12:08:09 2010
Status: Expired (432 days past expiration)
Calibration License: Installed license number 999998
```

See Also

GxPsGetBoardSummary, GxPsGetErrorString, **GxPsInitialize**

GxPsGetChannelCalibrationDate

Purpose

Returns the specified channel calibration date from the on-board EEPROM.

Syntax

GxPsGetChannelCalibrationDate (*nHandle, nChannel pszCalibrationDate, nSumMaxLen, pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel number from which to get the current: 1 – channel A 2 – channel B
pszCalibrationDate	LPSTR	Buffer to contain the returned calibration date (null terminated string)
nSumMaxLen	SHORT	Size of the buffer contains the error string.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calibration date has the mm/dd/yyyy format, for example:

"03/20/2002"

Example

The following example returns the channel A calibration date.

```
SHORT nHandle, nStatus;

CHAR szCalibrationDate [256];

GxPsGetChannelCalibrationDate (nHandle, 1, szCalibrationDate, sizeof (szCalibrationDate),
&nStatus);
```

See Also

GxPsGetCalibrationInfo, **GxPsGetBoardSummary, GxPsGetErrorString**, **GxPsInitialize**

GxPsGetCurrent

Purpose

Returns the specified channel's current.

Syntax

GxPsGetCurrent (*nHandle, nChannel, pdCurrent, pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel number from which to get the current: 1 – channel A 2 – channel B
pdCurrent	LPDOUBLE	Returned current. Programmable 0-15V: 0.00A-10.00A Programmable 0-30V: 0.00A-5.00A Programmable 0-60V: 0.00A-2.5A Fixed Module: The measured current at the sense lines.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the specified channel is in standby mode, the returned current is 0.

Programmable modules current resolution is:

0-15V module: 76.75 µA/step, 0-30V module: 153.5 µA/step, 0-60V module: 305.1 µA/step.

If total power exceeds 150 W (maximum output power), output voltages will drop to satisfy the power limit.

Example

The following example returns the current from channel A.

```
SHORT nHandle, nStatus;
DOUBLE dCurrent;
GxPsGetCurrent (nHandle, 1, &dCurrent, &nStatus);
```

See Also

GxPsGetVoltage, GxPsGetCurrentLimit, GxPsSetCurrentLimit, GxPsGetErrorString

GxPsGetCurrentLimit

Purpose

Returns the specified channel's current limit.

Syntax

GxPsGetCurrentLimit (*nHandle, nChannel, pdCurrentLimit, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
Channel	SHORT	Specifies the Channel for which to get the Current Limit: 1 – channel A 2 – channel B
pdCurrentLimit	LPDOUBLE	Returned current limit. Programmable 0-15V: 0.00A-10.00A Programmable 0-30V: 0.00A-5.00A Programmable 0-60V: 0.00A-2.5A Fixed Module: The maximum rated current for the installed module.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

Programmable modules current resolution is:

0-15V module: 76.75 µA/step, 0-30V module: 153.5 µA/step, 0-60V module: 305.1 µA/step.

If total power exceeds 150 W (maximum output power), output voltages will drop to satisfy the power limit.

Example

The following example gets the Current Limit from Channel B.

```
SHORT nHandle, nStatus;
DOUBLE dCurrentLimit;
GxPsGetCurrentLimit (nHandle, 2, &dCurrentLimit, &nStatus);
```

See Also

GxPsSetCurrentLimit, GxPsGetVoltage, **GxPsGetCurrent GxPsSetVoltage, GxPsGetErrorString**, **GxPsInitialize**

GxPsGetDriverSummary

Purpose

Returns the driver description string and version number.

Syntax

GxPsGetDriverSummary (*szSummary*, *nSummaryMaxLen*, *pdwVersion*, *pnStatus*)

Parameters

Name	**Type**	**Comments**
pszSummary	LPSTR	Buffer to receive the summary string.
nSummaryMaxLen	SHORT	Buffer size passed by pszSummary.
pdwVersion	LPDWORD	Driver version
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns the driver summary.

```
SHORT nHandle, nStatus;

DWORD dwVersion;

CHAR szSummary[128];

GxPsGetDriverSummary(szSummary, 128, &dwVersion, &nStatus);

After the function call the parameter SzSummary will be set to "GxPs Driver for GX7400, Version
1.0 2001, Copyright (c) Geotest Marvin Test Systems Inc."
```

See Also

GxPsGetErrorString

GxPsGetErrorString

Purpose

Returns the error string associated with the specified error number.

Syntax

GxPsGetErrorString (*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:

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
-26	Parameter is out of the allowed range
-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
-37	Unable to create synchronization object
-38	This function is not available under LabView/Real-Time
-39	Unable to lock a board resource, resource is used by another process or thread
-40	Unable to execute this function. The function requires special license to be installed

Board Specific Errors

-50	Fatal error: Unable to set the Mode register
-51	Fatal error: Power supply not found!
-52	Fatal error: EEPROM not calibrated!
--53	Fatal error: Invalid DAC source
-54	Fatal error: EEPROM offset is out of range
-55	Fatal error: EEPROM gain is out of range
-56	Fatal error: Fail to set channel On
-57	Fatal error: Fail to set channel to Standby
-58	Fatal error: Channel generated a fault
-59	Fatal error: Channel must be set to ON prior calling this function
-60	Illegal timeout or not enough time to finish settings the specified voltage, try increasing the current limit
-61	Error: Timeout reading the Analog to Digital convertor
-62	Error: Timeout setting the Digital to Analog to convertor voltage
-63	Error: Timeout setting the Digital to Analog to convertor current

User Calibration

-70	In-System-Calibration Error: Invalid load value, load value is out of the allowed range, see User's Guide for details
-71	In-System-Calibration Error: Illegal calibration sequence
-72	In-System-Calibration Error: Invalid Fixed Module type
-73	In-System-Calibration Error: Invalid voltage value
-74	Invalid Calibration By User step number
-75	Calibration did not start

Board Parameters error

-80	Invalid DAQ Buffer Pointer Selection

-81	Invalid DAQ Buffer Count Selection
-82	Invalid HIRES Buffer Pointer Selection
-83	Invalid HIRES Buffer Count Selection
-84	Invalid Aperture Selection

Unknown error code

-99	Invalid or unknown error number

Example

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

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

GxPsGetExtendedSerialNumber

Purpose

Returns the Extended board’s Serial Number information.

Syntax

GxPsGetExtendedSerialNumber (*nHandle, pszSerialNum, nSerialNumMaxLen, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a counter board.
pszSerialNum	PSTR	Buffer to contain the returned Extended board’s Serial Number string.
nSerialNumMaxLen	SHORT	Size of the buffer to contain the error string.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The returned Extended board’s Serial Number string is provides information regarding hardware changes and modifications of the specified board. The information can be used by Geotests’s Customer Support department and by the user in order to determine what functionalities are supported by the board.

For example, the returned string could look like the following: “GX7400154-CA-CA-10”.

The function is supported only for power supplies with firmware version 0xA00D and above, calling **GetBoardSummary** can retrieve the firmware version**.**

See Also

GxPsGetBoardSummary, GxPsGetDriverSummary, **GxPsInitialize**, **GxPsGetErrorString**

GxPsGetStandby

Purpose

Returns the specified channel's state (Standby/On).

Syntax

GxPsGetStandby (*nHandle, nChannel, pnStdby, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel number: 1 – Channel A 2 – Channel B
pnStdby	LPSHORT	Returned value is as follow. 0. GXPS_OUPUT_ON- The channel is in "On" mode. 1. GXPS_OUPUT_STANDBY - The channel is in "Standby" mode.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

In Standby mode, board voltages and current limits can be set; however, outputs are disconnected.

Example

The following example returns the standby status of channel A.

```
SHORT nHandle, nStatus, nStdby;
GxPsGetStandby (nHandle, 1, &nStdby, &nStatus);
```

See Also

GxPsSetStandby, GxPsGetErrorString, **GxPsInitialize**

GxPsGetState

Purpose

Returns the channel's state (Inhibit/Fail).

Syntax

GxPsGetState (*nHandle*, *nChannel*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel number: 1 – Channel A 2 – Channel B
pnStatus	LPSHORT	Returned status is as follow: 0. GT_NO_ERROR - If neither failed nor inhibited 1. GXPS_CHANNEL_FAIL - Channel's status is FAIL 2. GXPS_CHANNEL_INHIBIT - Channel's status is INHIBIT

Example

The following example gets channel A's status.

```
SHORT nHandle, nStatus;
GxPsGetState (nHandle, 1, &nStatus);
```

See Also

GxPsGetErrorString, **GxPsInitialize**

GxPsGetType

Purpose

Returns the specified channel's module configuration.

Syntax

GxPsGetType (*nHandle, nModule, pnType, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nModule	SHORT	Specifies the Channel number: 1 – channel A 2 – channel B
pnType	LPSHORT	Returned values: 0. GXPS_TYPE_NONE - Module not installed 1. GXPS_MOD_PMP30 - Programmable module 0-30V. 2. GXPS_MOD_PMP60 - Programmable module 0-60V. 3. GXPS_MOD_PMC3 - 3V Fixed voltage module. 4. GXPS_MOD_PMC5 - 5V Fixed voltage module. 5. GXPS_MOD_PMC12 - 12V Fixed voltage module. 6. GXPS_MOD_PMC15 - 15V Fixed voltage module. 7. GXPS_MOD_PMC28 - 28V Fixed voltage module. 8. GXPS_MOD_PMP30 - Programmable module 0-15V.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

See Appendix B, “Specifications,” for a full specification of each GX7400 power supply modules.

Example

The following example gets channel A's configuration.

```
SHORT nHandle, nStatus, nType;
GxPsGetType (nHandle, 1, &nType, &nStatus);
```

See Also

GxPsGetErrorString, **GxPsInitialize**

GxPsGetVoltage

Purpose

Returns the specified channel's voltage.

Syntax

GxPsGetVoltage (*nHandle, nChannel, pdVoltage, pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel from which to get the voltage: 1 – Channel A 2 – Channel B
pdVoltage	LPDOUBLE	Returned voltage. Programmable 0-15V: 0.00V-15.00V Programmable 0-30V: 0.00V-30.00V Programmable 0-60V: 0.00V-60.00V Fixed Module: The measured voltage at the sense lines.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the specified channel is in standby mode, the returned voltage is 0.

Programmable modules current resolution is:

0-15V module: 0.915 mV/step

0-30V module: 1.83 mV/step

0-60V module: 3.66 mA/step.

If total power exceeds 150 W (maximum output power), output voltages will drop to satisfy the power limit.

Example

The following example returns the voltage from channel A.

```
SHORT nHandle, nStatus;
DOUBLE dVoltage;
GxPsGetVoltage (nHandle, 1, &dVoltage, &nStatus);
```

See Also

GxPsSetVoltage ,GxPsGetCurrentLimit, GxPsGetCurrent, GxPsSetCurrentLimit, GxPsGetErrorString, GxPsInitialize

GxPsInitialize

Purpose

Initializes the driver for the power supply at the specified base address.

Syntax

GxPsInitialize (*nSlotNumber*, *pnHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nSlotNumber	SHORT	Denotes the PCI slot number for location of GX7400 board.
pnHandle	LPSHORT	Returned handle for the power supply. The handle is set to zero on error and <> 0 on success.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Geotest HW device driver is installed with the driver and is the default device driver. The function returns a handle that for use with other Counter functions to program the board. The function does not change any of the board settings.

The specified PXI slot number is displayed 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. The function accepts two types of slot numbers:

- A combination of chassis number (chassis # x 256) with the chassis slot number. For example 0x105 (chassis 1 slot 5).
- 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 (1-255).

The **GxPsInitialize** function verifies whether the power supply does or does not exist. However, the function does not have any effect on the power supply settings.

Example

The following example initializes the GX7400 at slot number 1.

```
SHORT nHandle, nStatus;
GxPsInitilize (1, &nHandle, &nStatus);
if ( nHandle==0)
{   printf("Unable to Initialize the power supply")
    return;
}
```

See Also

GxPsGetErrorString, **GxPsGetBoardSummary**

GxPsInitializeVisa

Purpose

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

Syntax

GxPsInitializeVisa (*szVisaResource, pnHandle, pnStatus*)

Parameters

Name	Type	Comments
szVisaResource	LPCTSTR	String identifying the location of the specified board in order to establish a session.
pnHandle	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 **GxPsInitializeVisa** 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 Geotest 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: “PS1”. 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 Counter boards at PXI bus 5 and device 11.

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

See Also

GxPsInitialize, GxPsGetErrorString, GxPsReset

GxPsPanel

Purpose

Opens a virtual panel used to interactively control the GX7400.

Syntax

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

Parameters

Name	**Type**	**Comments**
pnHandle	LPSHORT	GX7400 slot number. This number may be zero if the power supply is to be initialized by the panel window.
hwndParent	DWORD	Sets the 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—Modeless window. 1—Modal window.
phwndPanel	LPDWORD	Returned window handle for the panel (for modeless panels only).
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

The panel window may be opened as a modal or a modeless window depending on the *nMode* parameter value.

If the mode is set to modal (*nMode*=1), the panel disables the parent window (*hwndParent*) and the function returns only after the window was closed by the user. In that case, *pnHandle* may return the handle created by the user using the panel Initialize dialog.

If the mode is set to modeless (*nMode*=0), the function returns immediately after creating the panel window, returning the window handle (*phwndPanel*) to the panel. It is the responsibility of the calling program to dispatch Windows messages to this window so that the window can respond to messages.

This function is supplied only with the DLL versions of the driver.

GxPsRecallSettings

Purpose

Recalls all the specified channel's parameters from the on-board EEPROM.

Syntax

GxPsRecallSettings (*nHandle, nChannel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel to recall the parameters for: 1. GXPS_CHANNEL1: Channel 1 2. GXPS_CHANNEL2: Channel 2
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Comments

If there were no prior calling to **GxPsStoreSettings,** the voltage and current limits are zero and the output state is set to **Standby.**

The AC Power must be connected in order to apply settings correctly.

Example

The following example recalls Channel 1 settings.

```
SHORT nHandle, nStatus;
GxPsRecallSettings (nHandle, GXPS_CHANNEL1, &nStatus);
```

See Also

GxPsSetVoltage, GxPsGetVoltage, **GxPsSetCurrentLimit, GxPsGetCurrent, GxPsGetCurrentLimit, GxPsSetStandby, GxPsGetStandby, GxPsGetErrorString, GxPsInitialize**

GxPsReset

Purpose

Sets all channels to standby, opens the isolation relays and sets the voltages and current limits to zero.

Syntax

GxPsReset (*nHandle*, *pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	GX7400 slot number.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example resets the power supply:

```
SHORT nHandle, nStatus;
GxPsReset(nHandle, &nStatus);
```

See Also

GxPsGetErrorString, **GxPsInitialize**

GxPsSetCurrentLimit

Purpose

Sets the specified channel's current limit.

Syntax

GxPsSetCurrentLimit (*nHandle, nChannel, dCurrentLimit, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel for which the current limit is to be set: 1 – Channel A 2 – Channel B
dCurrentLimit	DOUBLE	Current to be set. Programmable 0-15V: 0.00A-10.00A Programmable 0-30V: 0.00A-5.00A Programmable 0-60V: 0.00A-2.5A
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

Programmable modules current resolution is:

0-15V module: 76.75 µA/step

0-30V module: 153.5 µA/step

0-60V module: 305.1 µA/step.

If total power exceeds 150 W (maximum output power), output voltages will drop to satisfy the power limit.

This function is not supported by fixed modules.

Example

The following example sets the current limit to Channel B to 2 amps.

```
SHORT nHandle, nStatus;
GxPsSetCurrentLimit (nHandle, 2, 2.0, &nStatus);
```

See Also

GxPsGetCurrentLimit, GxPsGetVoltage, GxPsGetCurrent, GxPsSetVoltage, GxPsGetErrorString, **GxPsInitialize**

GxPsSetStandby

Purpose

Sets the specified channel to STANDBY or ON.

Syntax

GxPsSetStandby (*nHandle, nChannel, bStdby, pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel number: 1 – Channel A 2 – Channel B
nStdby	SHORT	Operation to perform. 0. The channel is in ON mode. 1. The channel is in STANDBY mode.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

When setting the specified channel to ON, this function will apply the voltage and current limit settings gradually.

In Standby mode board voltages and current limits can be set; however, the outputs are disconnected.

Example

The following example sets channel A to standby mode.

```
SHORT nHandle, nStatus;
GxPsSetStandby (nHandle, 1, GXPS_OUPUT_STANDBY, &nStatus);
```

See Also

GxPsGetStandby, GxPsGetErrorString, **GxPsInitialize**

GxPsSetStandbyImmediate

Purpose

Sets the specified channel to STANDBY or ON at the shortest time possible.

Syntax

GxPsSetStandbyImmediate (*nHandle, nChannel, bStdby, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel number:
		1 – Channel A
		2 – Channel B
nStdby	SHORT	Operation to perform.
		0. The channel is in ON mode.
		1. The channel is in STANDBY mode.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

When setting the specified channel to ON, this function will apply the voltage and current limit settings at the shortest time possible.

When setting the specified channel to STANDBY, this function will set the voltage and currents limit to zero and disconnect the outputs at the shortest time possible.

NOTE: Supported by power supplies with Serial Numbers greater then 74000140 or older models that were upgraded to support that functionality.

In Standby mode board voltages and current limits can be set; however, the outputs are disconnected.

Example

The following example sets channel A to standby mode.

```
SHORT nHandle, nStatus;
GxPsSetStandbyImmediate (nHandle, 1, GXPS_OUPUT_STANDBY, &nStatus);
```

See Also

GxPsGetStandby, GxPsGetErrorString, **GxPsInitialize**

GxPsSetVoltage

Purpose

Sets the specified channel's voltage.

Syntax

GxPsSetVoltage (*nHandle, nChannel, dVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel for which to set the voltage: 1 – Channel A 2 – Channel B
dVoltage	DOUBLE	Voltage. Programmable 0-15V: 0.00V-15.00V Programmable 0-30V: 0.00V-30.00V Programmable 0-60V: 0.00V-60.00V
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

Programmable modules current resolution is:

0-15V module: 0.915 mV/step

0-30V module: 1.83 mV/step

0-60V module: 3.66 mA/step.

If total power exceeds 150 W (maximum output power), output voltages will drop to satisfy the power limit.

This function is not supported by fixed modules.

Example

The following example sets the voltage in Channel A to 3.42 volts.

```
SHORT nHandle, nStatus;
GxPsSetVoltage (nHandle, 1, 3.42, &nStatus);
```

See Also

GxPsSetVoltageAndWaitUntilReady, **GxPsGetVoltage**, **GxPsSetCurrentLimit, GxPsGetCurrent, GxPsGetCurrentLimit, GxPsGetErrorString, GxPsInitialize**

GxPsSetVoltageAndWaitUntilReady

Purpose

Sets the specified channel's voltage, and wait until the specified voltage is set.

Syntax

GxPsSetVoltageAndWaitUntilReady (*nHandle, nChannel, dVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel for which to set the voltage: 1. GXPS_CHANNEL1: Channel 1 2. GXPS_CHANNEL2: Channel 2
dVoltage	DOUBLE	Voltage. • Programmable 0-15V: 0.00V-15.00V • Programmable 0-30V: 0.00V-30.00V • Programmable 0-60V: 0.00V-60.00V
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Comments

The function sets the specified channel's voltage and wait for the specified voltage to be reached. The function may return with a warning (GXPS_SET_VOLTAGE_TIMEOUT) if the specified voltage cannot be set, e.g. low current limit will restrict the output voltage.

If the specified channel was in Standby (output disabled) while calling this function an internal flag will be set. The next time the channel's output is enabled the driver will wait until the voltage is stable before returning.

Programmable modules current resolution is:

1. 0-15V module: 0.915 mV/step, 0-30V module: 1.83 mV/step, 0-60V module: 3.66 mA/step.
2. If total power exceeds 150 W (maximum output power), output voltages will drop to satisfy the power limit.

Not supported by fixed modules.

Example

The following example sets the voltage in Channel 1 to 3.42 volts.

```
SHORT nHandle, nStatus;
GxPsSetVoltageAndWaitUntilReady (nHandle, GXPS_CHANNEL1, 3.42, &nStatus);
```

See Also

GxPsSetVoltage, GxPsGetVoltage, **GxPsSetCurrentLimit, GxPsGetCurrent, GxPsGetCurrentLimit, GxPsSetStandby, GxPsGetStandby, GxPsGetErrorString, GxPsInitialize**

GxPsStoreSettings

Purpose

Stores all the specified channel's parameters to the on-board EEPROM. The EEPROM will store the channel voltage, current limit and the On/Standby state.

Syntax

GxPsStoreSettings (*nHandle, nChannel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel to store it parameters: 1. GXPS_CHANNEL1: Channel 1 2. GXPS_CHANNEL2: Channel 2
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Comments

The AC Power must be connected in order to store current settings propely.

Example

The following example stores Channel 1 settings.

```
SHORT nHandle, nStatus;
GxPsStoreSettings (nHandle, GXPS_CHANNEL1, &nStatus);
```

See Also

GxPsRecallSettings, GxPsSetVoltage, GxPsGetVoltage, **GxPsSetCurrentLimit, GxPsGetCurrent, GxPsGetCurrentLimit, GxPsSetStandby, GxPsGetStandby, GxPsGetErrorString, GxPsInitialize**

GxPsUpgradeFirmware

Purpose

Upgrades the board’s firmware.

Syntax

GxPsUpgradeFirmware (*nHandle, szFile ,nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX7400 board.
szFile	PCSTR	Path and file name of the firmware file. The firmware file extension is JAM.
nMode	SHORT	The upgrading firmware mode can be as follows: 0. GT_FIRMWARE_UPGRADE_MODE_SYNC: the function returns when upgrading firmware is done or in case of an error. 1. GT_ FIRMWARE_UPGRADE_MODE_ASYNC: the function returns immediately. The user can monitor the progress of upgrading firmware using the **GxPsUpgradeFirmwareStatus** API.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function used in order to upgrade the board’s firmware. The firmware file can only obtained by request from Geotest.

Note: Loading an incorrect firmware file to the board can permanently damage the board.

Example

The following example loads Upgrades the board’s firmware using synchronous mode:

```
GxPsUpgradeFirmware (nHandle, “C:\\Gx7400Fw.JAM”, GT_LOAD_MODE_SYNC, &nStatus);
```

See Also

GxPsUpgradeFirmwareStatus, GxPsGetErrorString

GxPsUpgradeFirmwareStatus

Purpose

Monitor the firmware upgrade process.

Syntax

GxPsUpgradeFirmwareStatus (*nHandle*, *pszMsg*, *nMsgMaxLen*, *pnProgress*, *pbIsDone*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX7400 board.
pszMsg	PSTR	Buffer to contain the message from the firmware upgrade process.
nMsgMaxLen	SHORT	*pszMsg* buffer size .
pnProgress	PSHORT	Returns the firmware upgrades progress.
pbIsDone	PBOOL	Returned TRUE if the firmware upgrades is done.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function is used in order to monitor the firmware upgrade process whenever the user called **GxPsUpgradeFirmware** API with GT_ FIRMWARE_UPGRADE_MODE_ASYNC mode.

Note: In order to prevent CPU over load if the function is called form within a loop, a delay of about 500mSec will be activated if the time differences between consecutive calls are less than 500mSec.

Example

The following example loads Upgrades the board's firmware using asynchronous mode, and ten monitors the firmware upgrade process:

```
CHAR    sz[256];
CHAR    szMsg[256];
BOOL    bIsDone=FALSE;
GxPsUpgradeFirmware (nHandle, “C:\\Gx7400Fw.JAM”, GT_UPGRADE_FIRMWARE_MODE_ASYNC, &nStatus);
if (nStatus<0)
{   GxPsGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz);// prints the error string returns
}
While (bIsDone==FALSE || nStatus<0)
{   GxPsUpgradeFirmwareStatus (nHandle, szMsg, sizeof szMsg, &nProgress, &bIsDone, &nStatus);
    printf(“Upgrade Progress %i”, nProgress);
    sleep(1000);
}
if (nStatus<0)
{   GxPsGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz);// prints the error string returns
}
```

See Also

GxPsUpgradeFirmware, GxPsGetErrorString

GxPsUserCalReadMeasuredVal

Purpose

Returns the specified calibration step voltage value as was set by the user.

Syntax

GxPsUserCalReadMeasuredVal (*nHandle, nCalStep, pdVal, pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	GX7400 slot number.
nCalStep	SHORT	Calibration step number: 1. GXPS_USER_CAL_STEP1 2. GXPS_USER_CAL_STEP2 3. GXPS_USER_CAL_STEP3 4. GXPS_USER_CAL_STEP4 5. GXPS_USER_CAL_STEP5 6. GXPS_USER_CAL_STEP6 7. GXPS_USER_CAL_STEP7 8. GXPS_USER_CAL_STEP8 9. GXPS_USER_CAL_STEP9
pdVal	PDOUBLE	Measured voltage value as was set by the user.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Example

The following example returns calibration step 2 voltage.

```
SHORT nHandle, nStatus;
DOUBLE  dVal;
GxPsUserCalReadMeasuredVal (nHandle, GXPS_USER_CAL_STEP2, &dVal, &nStatus);
```

See Also

GxPsUserCalWriteMeasuredVal, GxPsUserCalRestoreFactoryCalibration, GxPsUserCalSetup, GxPsUserCalSetupForMeasurement, GxPsUserCalStoreCalibrationData, GxPsGetErrorString

GxPsUserCalRestoreFactoryCalibration

Purpose

Restore the specified channel’s calibration data back to the manufacture data.

Syntax

GxPsUserCalRestoreFactoryCalibration (*nHandle, nChannel, pnStatus*)

Parameters

Name	Type	Comments
NHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel for which to set the voltage: 1. GXPS_CHANNEL1: Channel 1 2. GXPS_CHANNEL2: Channel 2
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Comments

The function overwrites the current calibration data with the manufacture’s calibration data.

Example

The following example restore channel 1 calibration data back to the manufacture data.

```
SHORT nHandle, nStatus;
GxPsUserCalRestoreFactoryCalibration (nHandle, GXPS_CHANNEL1, &nStatus);
```

See Also

GxPsUserCalReadMeasuredVal, GxPsUserCalSetup, GxPsUserCalSetupForMeasurement, GxPsUserCalStoreCalibrationData, GxPsUserCalWriteMeasuredVal, GxPsGetErrorString

GxPsUserCalSetup

Purpose

Initialize the specified channel's for calibration and sets the load value.

Syntax

GxPsUserCalSetup (*nHandle, nChannel, dLoadVal, nFixedModuleType, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nChannel	SHORT	Specifies the Channel for which to set the voltage: 1. GXPS_CHANNEL1: Channel 1 2. GXPS_CHANNEL2: Channel 2
dLoadVal	DOUBLE	Load value connected to the channel measured by the user.
nFixedModuleType	SHORT	In case of a Fixed Module user need to specify the Module Type to be one of the followings: 3. GXPS_TYPE_PMC3: Fixed Voltage 3.3V 4. GXPS_TYPE_PMC5: Fixed Voltage 5V 5. GXPS_TYPE_PMC12: Fixed Voltage 12V 6. GXPS_TYPE_PMC15: Fixed Voltage 15V 7. GXPS_TYPE_PMC28: Fixed Voltage 28V
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Comments

The function initialize the channel's for calibration and sets the load value. The load value needs to be measured by the user before calling this function. See **Preparing the Load** section for details on selecting the right load value for the channel type.

Example

The following example initialize the channel 1 for calibration with a load of 15.6Ohms.

sets the voltage in Channel 1 to 3.42 volts.

```
SHORT nHandle, nStatus;
GxPsUserCalSetup (nHandle, GXPS_CHANNEL1, 15.6, NULL, &nStatus);
```

See Also

GxPsUserCalReadMeasuredVal, GxPsUserCalRestoreFactoryCalibration, GxPsUserCalStoreCalibrationData, GxPsUserCalWriteMeasuredVal, GxPsGetErrorString

GxPsUserCalSetupForMeasurement

Purpose

Sets the currently under calibration channel output voltage according to the specified calibration step.

Syntax

GxPsUserCalSetupForMeasurement (*nHandle, nCalStep, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nCalStep	SHORT	Calibration step number: 1. GXPS_USER_CAL_STEP1 2. GXPS_USER_CAL_STEP2 3. GXPS_USER_CAL_STEP3 4. GXPS_USER_CAL_STEP4 5. GXPS_USER_CAL_STEP5 6. GXPS_USER_CAL_STEP6 7. GXPS_USER_CAL_STEP7 8. GXPS_USER_CAL_STEP8 9. GXPS_USER_CAL_STEP9
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Comments

The function sets the currently under calibration channel output voltage for measurement according to the calibration step. The following step will be write the measured voltage back to the driver using the **GxPsUserCalWriteMeasuredVal** function.

Example

The following example sets the currently under calibration channel output for measurement according to calibration steps 3.

```
SHORT nHandle, nStatus;

GxPsUserCalSetupForMeasurement (nHandle, GXPS_USER_CAL_STEP3, &nStatus);
```

See Also

GxPsUserCalReadMeasuredVal, GxPsUserCalRestoreFactoryCalibration, GxPsUserCalSetup, GxPsUserCalStoreCalibrationData, GxPsUserCalWriteMeasuredVal, GxPsGetErrorString

GxPsUserCalStoreCalibrationData

Purpose

Stores the calibration data to the on-board EPROM.

Syntax

GxPsUserCalStoreCalibrationData (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Comments

The function conclude the calibration process and stores the calibrated channel's data to the on-board EPROM. Users can only call this function after all six calibration steps were executed in order. See the **Front Panel Calibration** and **Sample Calibration Program** sections for details.

Example

The following example Stores the calibration data to the on-board EPROM.

```
SHORT nHandle, nStatus;
GxPsUserCalStoreCalibrationData (nHandle, &nStatus);
```

See Also

GxPsUserCalReadMeasuredVal, GxPsUserCalRestoreFactoryCalibration, GxPsUserCalSetup, GxPsUserCalSetupForMeasurement, GxPsUserCalWriteMeasuredVal, GxPsGetErrorString

GxPsUserCalWriteMeasuredVal

Purpose

Sets the specified calibration step voltage value as was measured by the user.

Syntax

GxPsUserCalWriteMeasuredVal (*nHandle, nCalStep, dVal, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX7400 slot number.
nCalStep	SHORT	Calibration step number: 1. GXPS_USER_CAL_STEP1 2. GXPS_USER_CAL_STEP2 3. GXPS_USER_CAL_STEP3 4. GXPS_USER_CAL_STEP4 5. GXPS_USER_CAL_STEP5 6. GXPS_USER_CAL_STEP6 7. GXPS_USER_CAL_STEP7 8. GXPS_USER_CAL_STEP8 9. GXPS_USER_CAL_STEP9
dVal	DOUBLE	Measured voltage value as was set by the user.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure, >0 for warning.

Comments

The function sets the specified calibration step voltage value as was measured by the user. This function follows the **GxPsUserCalSetupForMeasurement** function.

Example

The following example sets calibration step 2 voltage as was measured by the user to be 23.05V.

```
SHORT nHandle, nStatus;
DOUBLE  dVal;
GxPsUserCalWriteMeasuredVal (nHandle, GXPS_USER_CAL_STEP2, 23.05, &nStatus);
```

See Also

GxPsUserCalReadMeasuredVal, GxPsUserCalRestoreFactoryCalibration, GxPsUserCalSetup, GxPsUserCalSetupForMeasurement, GxPsUserCalStoreCalibrationData, GxPsGetErrorString

Appendix A– Connectors and Cables

Connections

Connections to the GX7400 are made using two standard D-type male connector; one for each channel.

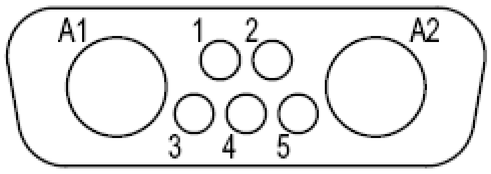


Figure A-1: Power supply connector

Channel 1 Connector Pins

Pin	Name	Function
A1	OUT1+	Positive Output Power for Module 1
A2	OUT1-	Power Return for Module 1
1	SNS1+	Sense for Positive Output of Module 1
2	SNS1-	Sense for Negative Output of Module 1
3	GND	Signal ground (reference for the Inhibit input)
4	Inhibit	Inhibit input, active low.
5	SyncL0	For future use, do not connect

Table A-1: Channel 1 Connector Pins

Channel 2 Connector Pins

Pin	Name	Function
A1	OUT1+	Positive Output Power for Module 2
A2	OUT1-	Power Return for Module 2
1	SNS1+	Sense for Positive Output of Module 2
2	SNS1-	Sense for Negative Output of Module 2
3	GND	Signal ground (reference for the Inhibit input)
4	Inhibit	Inhibit input, active low
5	SyncL0	For future use, do not connect

Table A-2: Channel 2 Connector Pins

Connectors and Accessories

The following accessories are available from Geotest for GX7400 power supply.

Part / Model Number	Description
GX97601	Mating connector for GX7400
GX97602	Mating connector for GX7400 with a 6' unterminated harness
GX97603	3' harness for GX7400, D-Type connector on Both ends (Male to Female)
GX97607	3' harness for GX7400, D-Type connector on Both ends (Male to Male)
GX97604	Calibration service for GX7400
GX97605	Calibration Software for GX7400
GX97606	6' harness for GX7400, D-Type connector on Both ends (Male to Female)
GX97608	6' harness for GX7400, D-Type connector on Both ends (Male to Male)

Table A-2: Spare Connectors and Accessories

Appendix B – Specifications

Modules

	GX7415	GX7430	GX7460	GX7420-03	GX7420-05	GX7420-12	GX7420-15	GX7420-28
Type	Prog. output	Prog. output	Prog. output	Fixed. output	Fixed. output	Fixed. output	Fixed. output	Fixed. output
Voltage (V)	0 - 15	0-30	0-60	3.3	5	12	15	28
Power	150	150	150	33	50	100	100	100
Max Current	10A	5A	2.5A	10A	10A	8.4A	7.6A	3.6A
Programming Accuracy	+/- 10 mV	10mV	20mV	N/A	N/A	N/A	N/A	N/A
Voltage Mode Noise and Regulation								
RMS	8 mV	8mV	8mV	N/A	N/A	N/A	N/A	N/A
P-P(mV)	20	30	30	100	100	150	150	280
Load Regulation (mV)	+/- 10	10	10	*0.8% or 40	*0.8% or 40	*0.8% or 40	*0.8% or 40	*0.8% or 40
Line Regulation (mV)	+/- 10	10	10	*0.4% or 20	*0.4% or 20	*0.4% or 20	*0.4% or 20	*0.4% or 20
Readback Accuracy (16 bit resolution)								
Voltage	+/- 2 mV	+/- 3 mV	+/- 5 mV					
Current	+/- 1 mA	+/- 1 mA	+/- 1 mA					

* The greater of the two

Output Characteristics

Modes of Operation:	Constant Voltage & Current Limit
GX7430/60:	Current Limit
Programming Resolution:	14 bits
Read back Resolution:	16 bits
Remote/Local Sense:	Up to 2 VDC may be dropped across sense loads. Drop decreases voltage available at load.

Input AC Power

Configuration:	Single Phase
Voltage:	GX7400: 115 VAC or 230 VAC (selectable to either 115VAC or 230VAC range)
	GX7400A: 85 to 230 VAC (auto-selectable)
Frequency:	47-63 Hz
Efficiency:	Typ.>75% (depends on programmed volt. & current)

Power Factor:	>0.85

Power Requirements

3.3V Power	0.1A Typical, 0.3A Max
5V Power	0.1A Typical, 0.1A Max

Protection Circuitry

Over Voltage:	105% of full scale.
Current Limit:	Programmable from 0 to FS. Output will automatically switch into constant current mode when limit is reached.
Current Trip:	105% of full scale (fixed).
Over Temperature:	Automatically disables output if maximum allowable temperature is exceeded.
UUT Discharge:	Active when output is down programmed. Circuit will discharge at a constant rate.
Short Circuit:	Outputs are protected in the event of a short across output terminals.
Isolation Relays:	Independently controllable for each output. Circuitry is provided to protect relays from switching under load.
Remote Inhibit:	Provides for more remote hardware shutdown of outputs via front panel interlock pins.

Physical

Format:	6U triple slot
Size:	6.7”(h) x 2.4”(w) x 15.5”(d)
Approximate Weight:	7 lbs.
Operating Temp:	0 to 55°C
Humidity:	20% - 95% non-condensing
Cooling:	PXI Chassis airflow plus one fan per programmable module

Index

.

.NET ii

A

Applications........ 5

ATEasy........ ii, 18, 19, 21, 22

B

Block diagram 6

Board Description........ 4

Board Handle........ 24

Borland .... ii, 18, 21, 22

Borland-Delphi 22

Bus Interface, Control and Logic........ 6

C

C/C++ 18, 19, 21

C++........ 21

Calibration dialog tutorial........ 38

Calibration Interval........ 31

Calibration required instruments 33

Channel 1 Connector Pins 85

Channel 2 Connector Pins 85

Connections 85

Connections diagram 33

Connectors........ 17

Connectors & Cables 85

Connectors and Accessories 86

Control........ 6

Control and Logic........ 6

Conventions

  notation........ 1

Corrupt files........ 20

D

Delphi .... ii, 18, 21, 22

Development tools........ 7

Directories 18

Distributing the Driver........ 25

DLL 7

Driver

  Directory........ 18

  Files 18

Driver Version 25

E

Error Handling........ 25

Error-Handling 25

Example........ 19

F

Front Panel Calibration dialog........ 37

Function library 7

G

GT7400

  Introduction 3

GT7420-03 87

GT7420-05 87

GT7420-12 87

GT7420-15 87

GT7420-28 87

GT7430........ 87

GT7460........ 87

GX7400 1

  Architecture 6

  Features........ 5

  programming 21

  software 7

GXPS........13
  Driver-Description........21
  Header-file........21
  Help-File-Description........19
  Panel-File-Description........18
  programming........21
  software........7
  Supported-Development-Tools........21
GXPS.BAS........18, 21
GXPS.DLL........18, 21, 22
GXPS.EXE........13
GXPS.H........18, 21
GXPS.LIB........18, 21
GXPS.PAS........22
GXPSBC.LIB........18, 21
GxPsGetBoardSummary........48
GxPsGetCalibrationDate........49
GxPsGetCalibrationInfo........50
GxPsGetChannelCalibrationDate........52
GxPsGetCurrent........53
GxPsGetCurrentLimit........54
GxPsGetDriverSummary........23, 25, 55
GxPsGetErrorString........25, 45, 56
GxPsGetExtendedSerialNumber........59
GxPsGetStandby........60
GxPsGetStatus........61
GxPsGetType........62
GxPsGetVoltage........63
GxPsInitialize........9, 23, 24, 25, 64
GxPsInitializeVisa........9, 23, 24, 65
GxPsPanel........25, 66
GXPSPANEL.EXE........18, 25
GxPsRecallSettings........67
GxPsReset........25, 68
GxPsSetCurrentLimit........69
GxPsSetStandby........70
GxPsSetStandbyImmediate........71
GxPsSetVoltage........72
GxPsSetVoltageAndWaitUntilReady........73
GxPsStoreSettings........74
GxPstInitialize........45
GxPsUpgradeFirmware........75
GxPsUpgradeFirmware........47
GxPsUpgradeFirmwareStatus........47
GxPsUpgradeFirmwareStatus........76
GxPsUserCalReadMeasuredVal........78
GxPsUserCalRestoreFactoryCalibration........79
GxPsUserCalSetup........80
GxPsUserCalSetupForMeasurement........81
GxPsUserCalStoreCalibrationData........82
GxPsUserCalWriteMeasuredVal........83

H

Handle........16, 23, 24, 25
HW........13, 17, 18, 21, 25

I

Installation........12
Installation Directories........18
Installation:........16, 17
In-System Calibration Example program........40
In-System Calibration license setup........32

L

LabView........18
Listing........26, 40

N

nHandle........22, 24, 25

O

OnError........22

P

Panel........8, 11, 18, 20, 23, 25, 36
Pascal........18, 21, 22

PCI 18
Plug & Play 17
pnStatus 25, 45
Power conditioning unit 6
Program-File-Descriptions 18
Programming
  Borland-Delphi 22
  Error-Handling 25
  Panel-Program 25
Programming languages 7
Programming the GX7400 21
PXI 1, 12, 15, 16, 17, 23, 24, 36
PXI/PCI Explorer 9, 15, 23, 24, 36, 64, 65
PXIeSYS.INI 9
PXISYS.INI 9

R

README.TXT 18, 19
Readme-File 19
Required instruments 33
Reset 25
Running calibration using API calls 40
Running In-System calibration from the virtual panel 36

S

Sample 26
Sample User Calibration 40
Setup 13, 18, 20
Setup Maintenance 20
Setup-and-Installation 12
slot 26, 40
Slot 9, 16, 17, 23, 36
Software 7
  supported development tools 7
Specifications 87
Step 1
  Pre-calibration settings 38
Step 10
  Save Channel’s Calibration data 39
Step 2 through Step 9
  Measure voltage 39
System
  Directory 18
System-Requirements 12

U

Using the GXPS driver functions 23

V

Virtual Panel 8, 9, 10, 11, 13, 18, 23, 36
  About Page 11
  Initialize Dialog 9
  Setup Page 10
VISA 9, 23, 24, 64, 65
Visual Basic ......ii, 21
Visual C++ ......ii, 18, 21