

GX1164

PXI Programmable Resistors Board GXPRES Software

User's Guide

Last Updated: January 29, 2016

Safety and Handling

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

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

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

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

Warranty

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

If You Need Help

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

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

Disclaimer

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

Copyright

Copyright © 2002-2016 Marvin Test Solutions, Inc. All rights reserved. No part of this document can be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Marvin Test Solutions.

Trademarks

ATEasy®, CalEasy, DIOEasy®, DtifEasy, WaveEasy	Marvin Test Solutions, Inc., Geotest – Marvin Test Systems, Inc (prior company name)
C++ Builder, Delphi	Embarcadero Technologies Inc.
LabView, LabWindows™/CVI	National Instruments
Microsoft Developer Studio, Microsoft Visual C++, Microsoft Visual Basic,, .NET, Windows 95, 98, NT, ME, 2000, XP, VISTA, 7, 8 and 10	Microsoft Corporation

All other trademarks are the property of their respective owners.

Table of Contents

Safety and Handling.....	i
Warranty	i
If You Need Help.....	i
Disclaimer	i
Copyright	i
Chapter 1 - Introduction	1
Manual Scope and Organization	1
Manual Scope.....	1
Manual Organization.....	1
Conventions Used in this Manual	1
Chapter 2 - Overview	3
Introduction.....	3
Features.....	3
Applications.....	4
Board Description	4
Architecture	5
Specifications.....	6
Channel Specifications (GX1164)	6
Channel Specifications (GX1164-2)	6
Channel Specifications (GX1164-4)	6
Channel Specifications (GX1164-8)	6
Channel Specifications (GX1164-0.25)	7
Power Requirements	7
Environmental.....	7
Physical	7
Calibration	7
Software and Documentation.....	7
Virtual Panel Description.....	8
Virtual Panel Initialize Dialog	9
Virtual Panel Group A/B Pages	10
Virtual Panel About Page.....	11
Chapter 3 - Installation and Connections	13
Getting Started	13
Packing List	13
Unpacking and Inspection.....	13

System Requirements.....	13
Installation of the GXPRES Driver.....	14
Overview of the GXPRES Software.....	15
Configuring Your PXI System using the PXI/PCI Explorer.....	16
Board Installation.....	17
Before you Begin.....	17
Electric Static Discharge (ESD) Precautions.....	17
Installing a Board.....	17
Plug & Play Driver Installation.....	18
Removing a Board.....	18
Connectors and Jumpers.....	19
J3 Connector.....	19
Connectors and Accessories.....	20
Installation Directories.....	20
GXPRES Driver Files Description.....	21
Driver File and Virtual Panel.....	21
Interface Files.....	21
GXPRES On-line Help and Manual.....	21
ReadMe File.....	21
Example Programs.....	21
Setup Maintenance Program.....	23
Chapter 4 - Programming the Board.....	25
The GX1164 Driver.....	25
Programming Using C/C++ Tools.....	25
Programming Using Visual Basic.....	25
Programming Using Visual C#.....	25
Programming Using Pascal/Delphi.....	26
Programming GXPRES Boards Using ATEasy®.....	26
Programming Using LabView and LabView/Real Time.....	26
Using and Programming under Linux.....	26
Using the GXPRES driver functions.....	27
Initialization, HW Slot Numbers and VISA Resource.....	27
Board Handle.....	28
Reset.....	28
Error Handling.....	28
Driver Version.....	28
Panel.....	29

Distributing the Driver	29
Sample Programs	29
Sample Program Listing	30
Chapter 5 - Functions Reference.....	33
Introduction.....	33
GX1164 Functions	34
Gx1164GetBoardSummary.....	35
Gx1164GetChannelMode	36
Gx1164GetChannelRelays.....	37
Gx1164GetChannelResistanceRange	38
Gx1164GetDoubleChannelResistance.....	39
Gx1164GetSingleChannelResistance	40
Gx1164GetWiperMode	41
Gx1164Initialize	42
Gx1164InitializeVisa	43
Gx1164Panel.....	44
Gx1164Reset.....	45
Gx1164SetDoubleChannelResistance	46
Gx1164SetChannelRelays	48
Gx1164SetSingleChannelResistance	49
Gx1164SetWiperMode	50
GxPResGetDriverSummary.....	51
GxPResGetErrorStrings.....	52
Index	55

Chapter 1 - Introduction

Manual Scope and Organization

Manual Scope

The purpose of this manual is to provide all the necessary information to install, use, and maintain the GX1164 instrument. This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and some understanding of digital I/O.





This manual also provides programming information using the GX1164 driver (referred in this manual **GXPRES**). Therefore, good understanding of programming development tools and languages may be necessary.

Manual Organization

The GX1164 manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX1164 manual. Lists all the supported board and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX1164 features, board description, its architecture, specifications and the panel description and operation.
Chapter 3 –Installation and Connections	Provides instructions on how to install the GX1164 Board and its accompanying GXPRES software.
Chapter 4 – Programming the Board	Provides a listing of GX1164 driver files, general purpose/generic driver functions, and programming methods. Discusses various supported operating systems and development tools.
Chapter 5 – Functions Reference	Contains a listing of the general GX1164 functions. Each function is described along with its syntax, parameters, and special programming comments. Samples are given for each function.

Conventions Used in this Manual

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

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

Chapter 2 - Overview

Introduction

GX1164 is a 3U cPCI/PXI instrument board that provides 8 programmable resistance channels arranged in two groups of four. The channels can be configured by software commands to a specified resistance within the channel's range. Each channel consists of eight resistors and eight relays to shunt each resistor. Channel 1, 3, 5 and 7 can be connected to an adjacent one (channels 2, 4, 6 and 8) provide better accuracy and higher resistance. Five different models are available: GX1164 with resolution of 1 Ω , GX1164-2 with resolution of 2 Ω , GX1164-4 with resolution of 4 Ω , GX1164-8 with resolution of 8 Ω and the GX1164-0.25 with a resolution of 1/4 Ω .

Features

The GX1164 is a Programmable Resistors card with the following features:

- Four or eight Programmable Resistor Channels.
- Actual resistor values and relay resistance are stored in the on-board EEPROM.
- Each of channels 1, 3, 5 and 7 can be connected to their adjacent channel (channels 2, 4, 6 and 8) to provide better accuracy and higher resistance.
- Each group of four channels can be used in wiper mode.
- Five different models are supported: Gx1164 with resolution of 1 Ω , Gx1164-2 with resolution of 2 Ω , Gx1164-4 with resolution of 4 Ω , Gx1164-8 with resolution of 8 Ω and GX1164-0.25 with a resolution of 1/4 ohm.
- The Gx1164 model supports the following programmable configurations:
 - 4 Channels each ranges from 1 Ω to 64K Ω with 1 Ω resolution.
 - 2 Channels each ranges from 1 Ω to 128K Ω with 1 Ω resolution.
 - 1 Channel ranges from 1 Ω to 256K Ω with 1 Ω resolution.
 - 2 Channels each ranges from 1 Ω to 64K Ω Potentiometer with a Wiper and 1 Ω resolution.
- The Gx1164-2 model supports the following programmable configurations:
 - 4 Channels each ranges from 2 Ω to 128K Ω with 2 Ω resolution.
 - 2 Channels each ranges from 2 Ω to 256K Ω with 2 Ω resolution.
 - 1 Channel ranges from 2 Ω to 512K Ω with 2 Ω resolution.
 - 2 Channels each ranges from 2 Ω to 128K Ω Potentiometer with a Wiper and 2 Ω resolution.
- The Gx1164-4 model supports the following programmable configurations:
 - 4 Channels each ranges from 4 Ω to 256K Ω with 4 Ω resolution.
 - 2 Channels each ranges from 4 Ω to 512K Ω with 4 Ω resolution.
 - 1 Channel ranges from 4 Ω to 1024K Ω with 4 Ω resolution.
 - 2 Channels each ranges from 4 Ω to 256K Ω Potentiometer with a Wiper and 4 Ω resolution.

- The Gx1164-8 model supports the following programmable configurations:
 - 4 Channels each ranges from 8Ω to $512K\Omega$ with 8Ω resolution.
 - 2 Channels each ranges from 8Ω to 1024Ω with 8Ω resolution.
 - 1 Channel ranges from 8Ω to $2048K\Omega$ with 8Ω resolution.
 - 2 Channels each ranges from 8Ω to $512K\Omega$ Potentiometer with a Wiper and 8Ω resolution.
- The GX1164-0.25 module supports the following programmable configurations:
 - 4 channels, each supports values from $\frac{1}{4}$ ohm to 16K ohm
 - 2 channels, each supports values from $\frac{1}{4}$ ohm to 32K ohm
 - 1 channel, supporting a value from $\frac{1}{4}$ ohm to 64K ohm
 - 2 channels, each supporting values from $\frac{1}{4}$ ohm to 32K ohm
- The board can be configured to accommodate different resistors to accommodate customers' specifications.
- Channels are connected via a 25-pin D-type connector.

Applications

- Automatic Test Equipment (ATE) and Functional Test
- Data Acquisition
- Process Control
- Factory Automation

Board Description

Figure 2-1 shows the GX1164 board and its connector. J6 a 25 pin, D type, provides access to the 8 resistor channels. Two additional cPCI connectors (J1 and J2) are used to connect to the back plane and to pass in the PCI and PXI signals to the board.

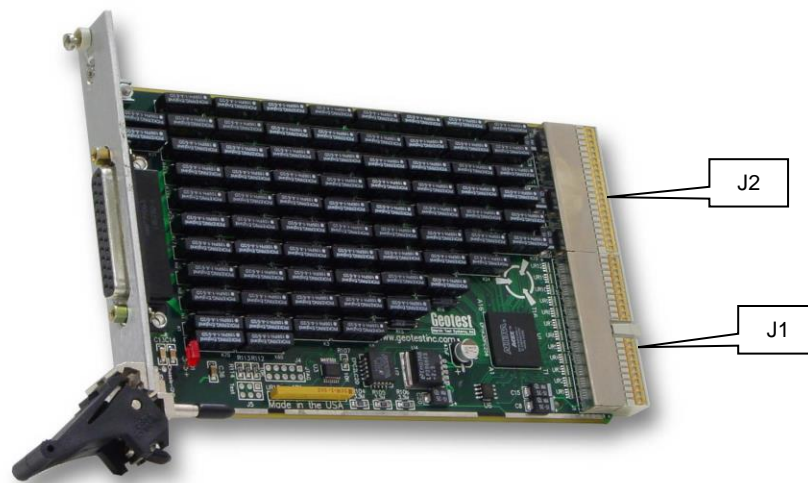


Figure 2-1: GX1164 Board (side view)

Architecture

The GX1164 provides 8 channels with 8-steps of binary programmable resistors. The 8 channels are divided into 2 groups of 4 channels each (channel 1 through 4 for the first group and channel 5 through 8 as the second group).

The following explanation refers to the Gx1164, all the other modes (Gx1164-2, Gx1164-4, Gx1164-8 and GX1164-0.25) have the same architecture but with different resistors values.

Figure 2-2 illustrates the configuration of a single group. Channels 1 and 3 resistors' (CH_x and CH_{x+2}) have binary ladder values from 1Ω to 128Ω and can therefore simulate resistors from 1Ω to 255Ω. Channels 2 and 4 resistors' (CH_{x+1} and CH_{x+3}) have binary ladder values from 256Ω to 32KΩ and can therefore simulate resistors from 256Ω to 64KΩ.

Three relays connect channels within groups to allow wider resistance ranges and the use of a group as a potentiometer with a wiper. There are also bypass relays to increase accuracy when selecting low resistance values.

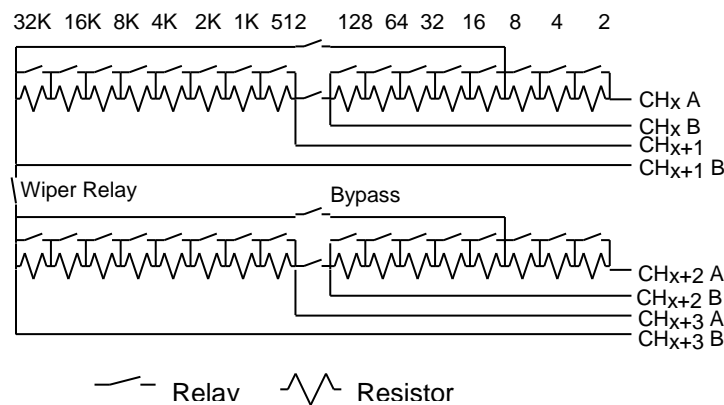


Figure 2-2: GX1164 Typical Group

The GX1164 resistance range of each group is as follows:

Single channel mode:

- Four programmable resistor channels of 1Ω to 255Ω (this range applies to channels 1,3, 5 and 7)
- Four programmable resistor channels of 256Ω to 65,535Ω (this range applies to channels 2, 4, 6 and 8)

Each group may also be used as double channel mode providing four programmable resistor channels of 1Ω to 65,535Ω (by combining channels 1 through 8 as shown in figure 2-2)

Or in Wiper Mode, where each group can emulate a variable resistor (potentiometer):

- One programmable resistor channel of 1Ω to 128KΩ (by combining channels 1, 2, 3 and 4 as shown in figure 2-2) with 'wiper' connection between channels 2 and 3.
- One programmable resistor channel of 1Ω to 720KΩ (by combining channels 5, 6, 7 and 8 as shown in figure 2-2) with 'wiper' connection between channels 6 and 7.

By selecting which resistor will be used (corresponding relay open) or shorted (relay closed), the GX1164 can provide any resistance value in the programmable range with resolution equal to the module's resistance ladder value – 0.25, 1, 2, 4, or 8 ohms.

Specifications

The following table outlines the specifications for all the GX1164 models.

Channel Specifications (GX1164)

Number of channels	8
Max Voltage on channel	200V
Max Power on channel	0.5W
Resistance setup time	5ms
Resolution	1 ohm
Programming Accuracy	+/- (1% of target value + 1 ohm)

Channel Specifications (GX1164-2)

Number of channels	8
Max Voltage on channel	200V
Max Power on channel	0.5W
Resistance setup time	5ms
Resolution	2 ohms
Programming Accuracy	+/- (1% of target value + 1 ohm)

Channel Specifications (GX1164-4)

Number of channels	8
Max Voltage on channel	200V
Max Power on channel	0.5W
Resistance setup time	5ms
Resolution	4 ohms
Programming Accuracy	+/- (1% of target value + 1 ohm)

Channel Specifications (GX1164-8)

Number of channels	8
Max Voltage on channel	200V
Max Power on channel	0.5W
Resistance setup time	5ms
Resolution	8 ohms
Programming Accuracy	+/- (1% of target value + 1 ohm)

Channel Specifications (GX1164-0.25)

Number of channels	8
Max Voltage on channel	200V
Max Power on channel	0.5W
Resistance setup time	5ms
Resolution	0.25 ohms
Programming Accuracy	+/- (1% of target value + 1 ohm)

Power Requirements

3.3 VDC Current	0.1A Typical, 0.3A Max.
5.0 VDC Current	0.4A Typical, 0.8A Max

Environmental

Temperature: Operating:	0 to +55°C -20 to +70°C
----------------------------	----------------------------

Physical

Size	3U cPCI/PXI
Weight	250g

Calibration

Board must be calibrated by the manufacturer every two years. Last calibration date is displayed in the board virtual panel and programmatically using the Gx1164GetBoardSummary function. Contact Marvin Test Solutions to arrange for calibration.

Software and Documentation

The GX1164 software module, also referred to as the GXPRES module, includes the following:

- Setup program
- Various interface file and libraries to support various development tools and programming languages such as C, VB, ATEasy and more
- Programming examples
- Documentation including with on-line HTML help file and PDF manual /(requires Adobe Acrobat Reader)
- Virtual Panel to display and control the board settings (described later in this chapter).

Refer to chapters 2, 3, 4 and 5 for more information about these software components.

Virtual Panel Description

The GX1164 includes a virtual panel program, which enables full utilization of the various configurations and controlling modes. To fully understand the front panel operation, it is best to become familiar with the functionality of the board.

To open the virtual panel application, select **GX1164 Panel** from the **Marvin Test Solutions, GXPRES** menu under the **Start** menu. The GX1164 virtual panel opens as shown here:

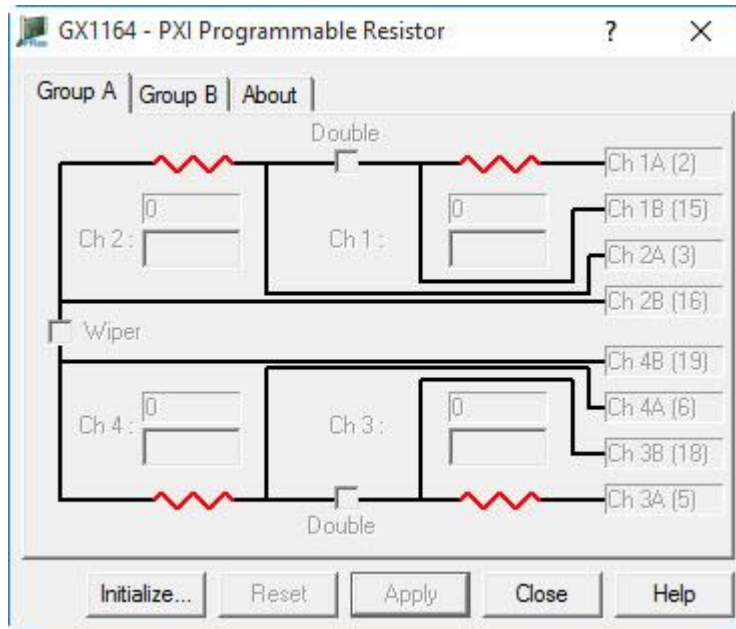


Figure 2-3: GX1164 Virtual Panel

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 counter 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 Marvin Test Solutions' 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 **Gx1164Initialize** function used to initialize driver with the specified board.

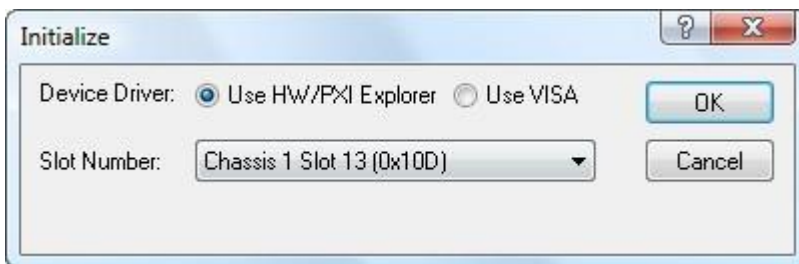


Figure 2-4: Initialize Dialog Box using Marvin Test Solutions' HW driver

2. **Use VISA** – this is a third party device driver usually provided by National Instrument (NI-VISA). When selected, the **Resource** list displays the available boards installed in the system and their VISA resource address. The chassis, slots, devices and their resources are also displayed by the VISA resource manager, **Measurement & Automation** (NI-MAX) and in Marvin Test Solutions **PXI/PCI Explorer**. The following figure shows 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 **Gx1164InitializeVisa** function to initialize the driver with the specified board.



Figure 2-5: Initialize Dialog Box using VISA resources

Virtual Panel Group A/B Pages

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

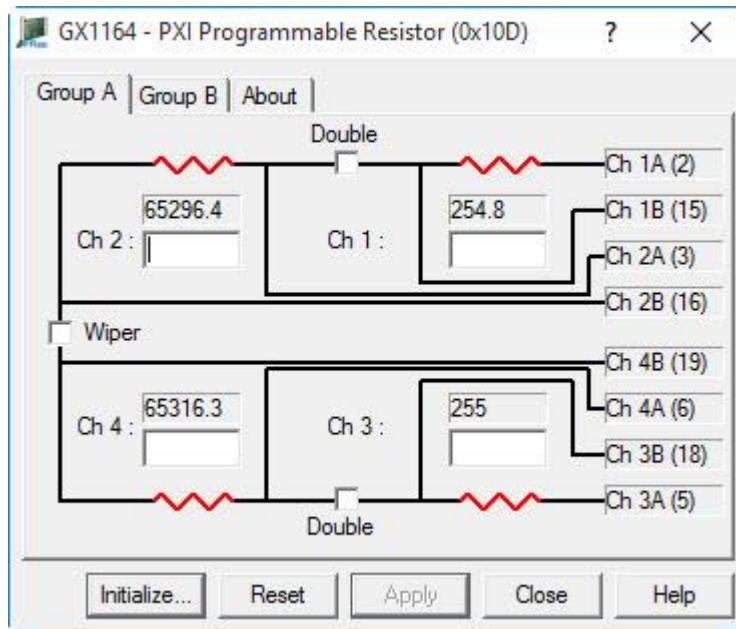


Figure 2-5: GX1164 Virtual Panel – Group A page

The following controls are shown in the **Group A** page:

Single Channel Ch 1, 2, 3 4 and double channels Ch 1 / 2 and 3 / 4: Each channel is displayed with edit box that is used for typing the required resistance value. The actual resistance value is displayed in the text box below the edit control. To set the channel value: type in the required resistance value and click on the **Apply** button.

Double Check Boxes: Two check boxes for channel 1 / 2 and 3 / 4 . When checked the channel become Double Channel (e.g. Ch 1 / 2). When the box is unchecked the channel are in Single mode. Figure 2-5 shows channel 1 and 2 in single mode and channel 3 / 4 in double mode. Clicking on the check box will show the group block diagram in red in the background.

Wiper Check box: Use to set the group to wiper mode.

Ch xA/B Text Boxes: Displays the pin name and number in the board connector of the specified resistor channel.

Similar page is available for group B.

Virtual Panel About Page

Clicking on the **About** tab will show the **About page** as shown in Figure 2-6:

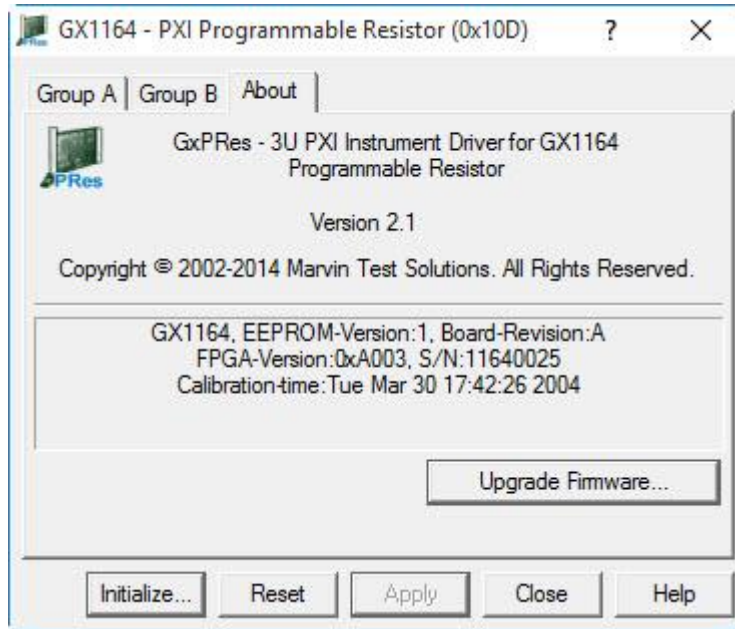


Figure 2-6: GX1164 Virtual Panel – About Page

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

Chapter 3 - Installation and Connections

Getting Started

This section includes general hardware installation procedures for the GX1164 board and installation instructions for the GX1164 (GXPRES) software. Before proceeding, please refer to the appropriate chapter to become familiar with the board being installed.

To Find Information on...	Refer to...
GXPRES Driver Installation	This Chapter
Hardware/Board Installation	This Chapter
Programming	Chapter 4
GX1164 Function Reference	Chapter 5

Packing List

All GX1164 board have the same basic packing list, which includes:

1. GX1164 Board
2. Disk with GXPRES Software

Unpacking and Inspection

After removing the board from the shipping carton:



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

Remove the board from the static bag by handling only the metal portions.

Be sure to check the contents of the shipping carton to verify that all of the items found in it match the packing list.

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

The GX1164 instrument boards are designed for use with a 3U or 6U cPCI or PXI compatible chassis. The software is compatible with any computer system running Windows (32-bit or 64 bit) that was released prior to the release date of this software.

Each board requires one unoccupied 3U PXI bus slot.

Installation of the GXPRES Driver

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

1. Insert the Marvin Test Solutions CD-ROM and locate the **GXPRES.EXE** setup program. If your computer's Auto Run is configured, when inserting the CD a browser will show several options, select the Marvin Test Solutions Files option, then locate the setup file. If Auto Run is not configured you can open the Windows explorer and locate the setup files (usually located under \Files\Setup folder). You can also download the file from Marvin Test Solutions web site (www.MarvinTest.com).

2. Run the GXPRES setup and follow the instruction on the Setup screen to install the GXPRES driver.

Note: You may be required to restart the setup after logging-in as a user with Administrator privileges. This is required in-order to upgrade your system with newer Windows components and to install the HW kernel-mode device drivers that are required by the GXPRES 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 GXPRES is to be installed. Either click Browse to set up a new folder, or click Next to accept the default entry of C:\Program Files\Marvin Test Solutions\GXPRES for 32 bit Windows and C:\Program Files (x86)\Marvin Test Solutions\GXPRES for 64 bit Windows.
5. Select the type of Setup you wish and click **Next**. You can choose between **Typical**, **Run-Time** and **Custom** setups. **Typical** setup type installs all files. **Run-Time** setup type will install only the files required for controlling the board either from its driver or from its virtual panel. **Custom** setup type lets you select from the available components.

The program will now start its installation. During the installation, Setup may upgrade some of the Windows shared components and files. The Setup may ask you to reboot after it complete if some of the components it replaced were 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, GXPRES** menu located in the Windows Taskbar.

Overview of the GXPRES 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 PXIE SYS.ini that are used by Marvin Test Solutions instruments, the VISA provider and VISA based instruments drivers. In addition, the applet can be used to assign chassis numbers, Legacy Slot numbers and instruments alias names.

VISA is a standard maintained by the VXI Plug & Play System Alliance and the PXI Systems Alliance organizations (<http://www.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 Marvin Test Solutions' PXI/PCI Explorer).

GXPRES 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.

GXPRES driver - a DLL (GXPRES.DLL or GXPRES64.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, **Marvin Test Solutions, HW, PXI/PCI Explorer**.
2. **Identify Chassis and Controllers.** After the PXI/PCI Explorer started it will scan your system for changes and will display the current configuration. The PXI/PCI Explorer automatically detects systems that have Marvin Test Solutions controllers and chassis. In addition, the applet detects PXI-MXI-3/4 extenders in your system (manufactured by National Instruments). If your chassis is not 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 Marvin Test Solutions 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 **Gx1164Initialize** and **Gx1164InitializeVisa** functions.
4. **Save you work.** PXI Explorer saves the configuration to the following files located in the Windows folder: PXISYS.ini, PXIeSYS.ini and GxPxiSys.ini. Click on the **Save** button to save you changes. The PXI/Explorer prompt you to save the changes if changes were made or detected (an asterisk sign ' * ' in the caption indicated changes).

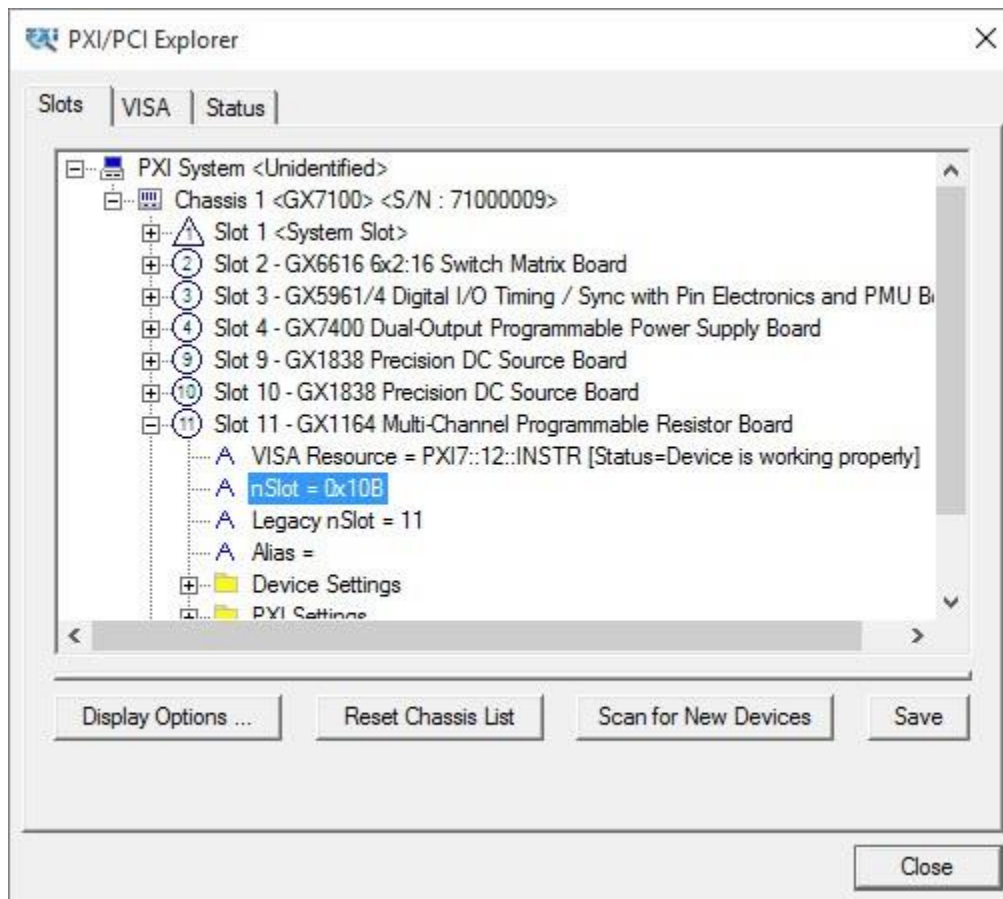


Figure 3-1: PXI/PCI Explorer

Board Installation

Before you Begin

Install the GXPRES 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 GX1164 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 GXPRES Driver as explain in the next section.
2. Turn off the PXI chassis and unplug the power cord.
3. Locate a PXI empty slot on 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 **out** (as shown in Figure 3-2).

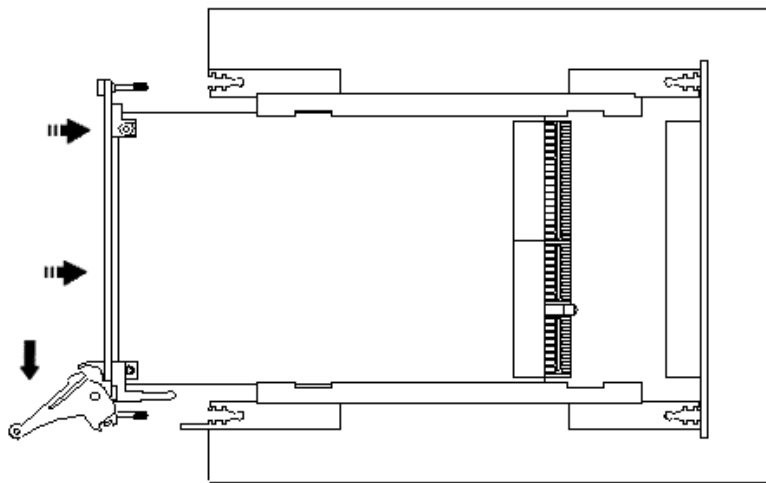


Figure 3-2: Ejector handles position during module insertion

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

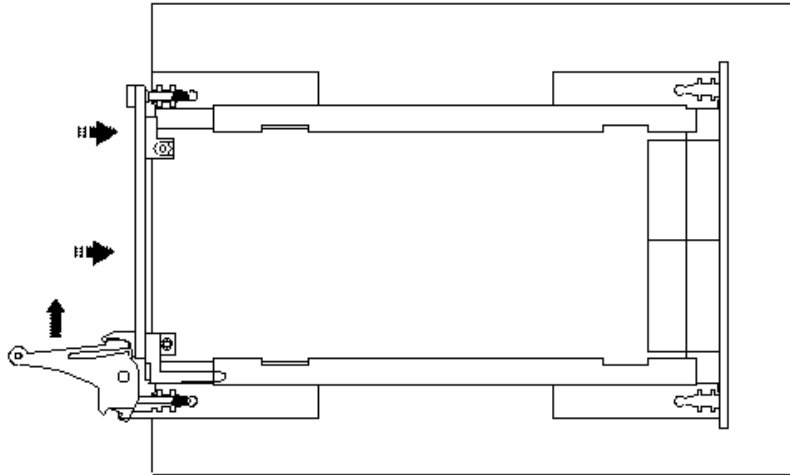


Figure 3-3: Ejector handles position after module insertion

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

Plug & Play Driver Installation

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

If the operating system was unable to find the driver (since the GXPRES driver was not installed prior to the board installation), you may install the GXPRES 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\Marvin Test Solutions\HW.

If you are unable to locate the driver click **Cancel** to the found New Hardware wizard and exit the New Hardware Found Wizard, install the GXPRES 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:

- Turn off the PXI chassis and unplug the power cord.
- Locate a PXI slot on the PXI chassis.
- Disconnect and remove any cables/connectors connected to the board.
- Unscrew the module's front panel screws
- Push out the ejector handles and slide the PXI board away from the chassis.
- Optionally - uninstall the GXPRES driver.

Connectors and Jumpers

Figure 3-4 shows the available GX1164 board connectors and jumpers followed by a description of J3. Two additional cPCI connectors (J1 and J2) are used to connect to the back plane and to pass in the PCI and PXI signals to the board.

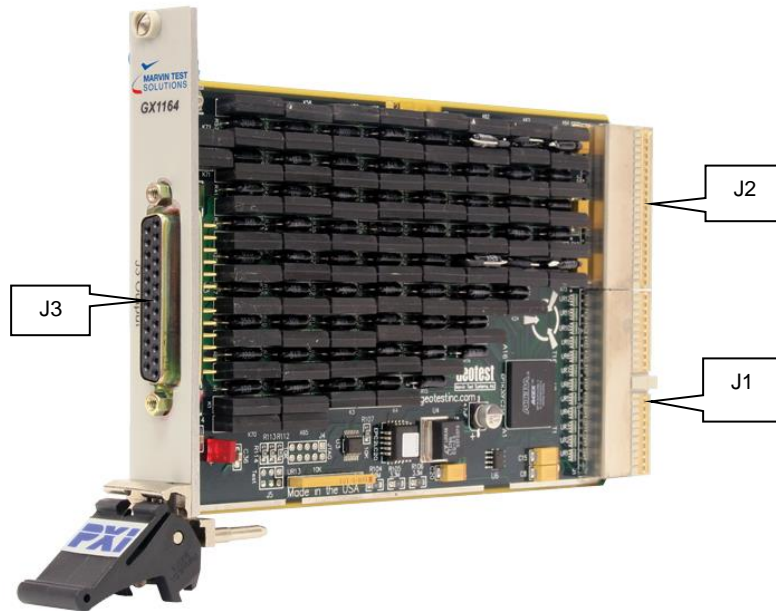


Figure 3-4: GX1164 Connectors and Jumpers

J3 Connector

The GX1164 has a single 25 pin female D-Type connector. The following table outlines the 25-pin connector and the function of each pin:

Pin #	Function	Pin #	Function
1	Chassis	14	Ground
2	CH1 A	15	CH1 B
3	CH2 A	16	CH2 B
4	Reserved	17	Reserved
5	CH3 A	18	CH3 B
6	CH4 A	19	CH4 B
7	Reserved	20	CH5 B
8	CH5 A	21	CH6 B
9	CH6 A	22	Reserved
10	Reserved	23	CH7 B
11	CH7 A	24	CH8 B
12	CH8 A	25	Ground
13	Chassis		

Connectors and Accessories

The following accessories are available from Marvin Test Solutions for GX1164 switching board.

Part / Model Number	Description
GX91801	25 Pin Male mating connector for GX1838/GX1164
GX91802	25 Pin Male mating connector for GX1838/GX1164 with a 3' un-terminated harness
GX91803	3' harness for GX1838/GX1164, 25 pin male/female connectors on both ends

Installation Directories

The GXPRES driver files are installed in the default directory C:\Program Files\Marvin Test Solutions\GXPRES for 32 bit Windows and C:\Program Files (x86)\Marvin Test Solutions\GXPRES for 64 bit Windows. You can change the default GXPRES directory to one of your choosing at the time of installation.

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

Name	Purpose / Contents
...\Marvin Test Solutions\GXPRES	The GX1164 directory. Contains panel programs, programming libraries, interface files and examples, on-line help files and other documentation.
...\Marvin Test Solutions\HW	HW device driver. Provide access to your board hardware resources such as memory, IO ports and PCI board configuration. See the README.TXT located in this directory for more information.
...\ATEasy\Drivers	ATEasy drivers directory. GXPRES Driver and example are copied to this directory only if <i>ATEasy</i> is installed to your machine.
Windows System Folders	Windows System directory. Contains the GXPRES DLL or GXPRES64.DLL driver and some upgraded system components, such as the HTML help viewer, etc.

GXPRES Driver Files Description

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

Driver File and Virtual Panel

- GXPRES.DLL and GXPRES64.DLL - 32/64-Bit MS-Windows DLLs for 32/64-bit applications running under Windows
- GXPRESPANEL.EXE and GXPRESPANEL64.DLL – An instrument front panel program for all GXPRES supported boards.

Interface Files

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

- GXPRES.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++
- GXPRES.LIB/GXPRES64.LIB - 32/64 bit import library for GXPRES.DLL/GXPRES64.DLL (used when linking C/C++ application).
- GXPRESBC.LIB - Import library for GXPRES.DLL (used when linking Borland C/C++ application that uses GXPRES.DLL).
- GXPRES.PAS - interface file to support Borland Pascal Borland Delphi.
- GXPRES.BAS - Supports Microsoft Visual Basic 4.0, 5.0 and 6.0.
- GXPRES.VB - Supports Microsoft Visual Basic .NET.
- GXPRES.CS - Supports Microsoft Visual C#.
- GX1164.DRV - ATEasy driver File for GX1164.GXPRES Virtual Panel Program.
- GXPRES.LLB – LabView library.

GXPRES On-line Help and Manual

GXPRES.CHM – On-line version of the GX1164 User's Guide. The help file is provided in a Windows Compiled HTML help file (.CHM). The file contains information about the GX1164 board, programming reference and panel operation.

GX1164.PDF – On line, printable version of the GX1164 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 GXPRES 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:

- GXPRESExampleC.cpp - Source file

- GXPRESExampleC.ico - Icon file
- GXPRESExampleC.rc - Resource file
- GXPRESExampleC.vcproj - VC++ .NET project file
- GXPRESExampleC.exe - Example executable

Microsoft Visual C++ 6.0 example files:

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

Borland C++ example files:

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

Microsoft Visual Basic .NET example files:

- GXPRESExampleVB.vb - Example form.
- GXPRESExampleVB.resx - Example form resource.
- GXPRESExampleVBapp.config - Example application configuration file.
- GXPRESExampleVBAssemblyInfo.vb - Example application assembly file
- GXPRESExampleVB.vbproj - Project file
- GXPRESExampleVB.exe - Example executable

Microsoft Visual C# example files:

- GXPRESExampleCS.cs - Example source
- GXPRESExampleCS.csproj - Project file
- GXPRESExampleCS.exe - Example executable

Microsoft Visual Basic 6.0 example files:

- GXPRESExampleVB6.frm - Example form
- GXPRESExampleVB6.frx - Example form binary file
- GXPRESExampleVB6.vbp - Project file
- GXPRESExampleVB6.exe - Example executable.

ATEasy driver and examples files (ATEasy Drivers directory):

- Gx1164.prj - example project
- GX1164.sys - example system
- GX1164.prg - example program

LabView example:

- GXPresExample.vi

Setup Maintenance Program

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

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

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

Follow the instruction on the screen until Setup is complete.

Chapter 4 - Programming the Board

This chapter contains information about how to program the switching instruments using the GXPRES driver. The GXPRES 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 GX1164 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++ 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 GX1164 driver types.

The GX1164 Driver

The GX1164 driver is a 32 bit Windows DLL file: GXPRES.DLL and a 64-bit DLL: GXPRES64.DLL. The DLL is used with 32 bit applications running under Windows. The DLL uses a device driver to access the board resources. The device driver HW.SYS (on Windows NT/2000/XP) or HW.VXD (on Windows 9x/Me) is installed by the setup program and is shared by other Marvin Test Solutions products (ATEasy, GXDIO, etc.).

The 32-bit 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 GX1164 driver with C/C++ development tools:

- Include the GXPRES.H header file in the C/C++ source file that uses the GX1164 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 the import library GXPRES.LIB for Microsoft Visual C++ for 32-bit applications, GXPRES64.LIB for 64-bit applications and GXPRESBC.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 GX1164 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 GXPRES.BAS to the project. For Visual Basic .NET use the GXPRES.VB.

The file can be loaded using *Add File* from the Visual Basic *File menu*. The GXPRES.BAS/.VB contains function declarations for the DLL driver.

Programming Using Visual C#

To use the driver with Visual C#, the user must include the GXPRES.CS.

Programming Using Pascal/Delphi

To use the driver with Borland Pascal or Delphi, the user must include the GXPRES.PAS to the project. The GXPRES.PAS file contains a **unit** with function prototypes for the DLL functions. Include the GX1164 unit in the **uses** statement before making calls to the GX1164 functions.

Programming GXPRES Boards Using ATEasy®

The GX1164 package is supplied with a separate ATEasy driver for each of board types. The ATEasy driver uses the GXPRES.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 GX1164.
- 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.

Programming Using LabView and LabView/Real Time

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

Using and Programming under Linux

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

Using the GXPRES driver functions

The GXPRES driver contains a set of functions for the GX1164. Functions names that starts with the **GxPRes** prefix applies to all GXPRES boards (i.e. **GxPResGetDriverSummary**). The GXPRES functions are designed with consistent set of arguments and functionality. All boards have a function that initializes the GXPRES 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 GXPRES driver supports two device drivers HW and VISA which are used to initialize, identify and control the board. The user can use the **Gx1164Initialize** to initialize the board's driver using HW and **Gx1164InitializeVisa** to initialize using VISA. The following describes the two different methods used:

1. **Marvin Test Solutions' HW** - the default device driver that is installed by the GXPRES driver. To initialize and control the board using the HW use the **Gx1164Initialize(*nSlot*, *pnHandle*, *pnStatus*)** function. The function initializes the driver for the board at the specified PXI slot number (*nSlot*) and returns a board handle. The **PXI/PCI Explorer** applet in the Windows Control Panel displays the PXI slot assignments. You can specify the *nSlot* parameter in the following way:

A combination of chassis number (chassis # x 256) with the chassis slot number, e.g. 0x105 for chassis 1 and slot 5. Chassis number can be set by the **PXI/PCI Explorer** applet.

Legacy *nSlot* as used by earlier versions of HW/VISA. The slot number contains no chassis number and can be changed using the **PXI/PCI Explorer** applet: 23 in this example.

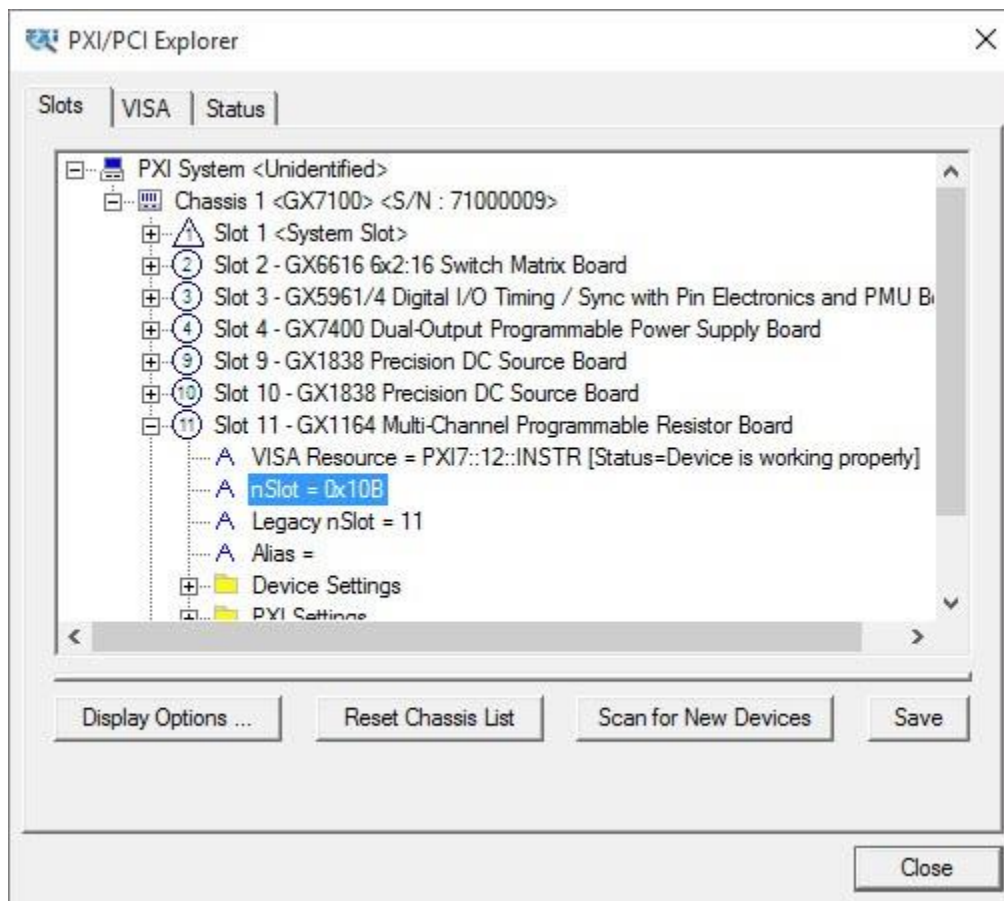


Figure 4-1: PXI/PCI Explorer

2. **VISA** – this is a third party library usually by National Instruments (NI-VISA). You must ensure that the VISA installed supports PXI and PCI devices (not all VISA providers supports PXI/PCI). GXPRES setup installs a VISA compatible driver for the GXPRES board in-order to be recognized by the VISA provider. Use the GXPRES function **Gx1164InitializeVisa** (*szVisaResource*, *pnHandle*, *pnStatus*) to initialize the driver board using VISA. The first argument *szVisaResource* is a string that is displayed by the VISA resource manager such as **NI Measurement and Automation (NI_MAX)**. It is also displayed by Marvin Test Solutions **PXI/PCI Explorer** as shown in the prior figure. The VISA resource string can be specified in several ways as the following examples:

Using chassis, slot: “PXI0::CHASSIS1::SLOT5”

Using the PCI Bus/Device combination: “PXI9::13::INSTR” (bus 9, device 9).

Using alias: “COUNTER1”. Use the PXI/PCI Explorer to set the device alias.

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

The **Gx1164Initialize** 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 **Gx1164Initialize** or the **Gx1164InitializeVisa** functions is a short integer (16 bits) number. It is used by the GXPRES 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 **Gx1164Initialize** function. But there is no need to destroy the handle. Calling **Gx1164Initialize** 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 **Gx1164Reset**(*nHandle*, *nStatus*) function. See the Function Reference for more information regarding the specific board.

Error Handling

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

Driver Version

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

Panel

Calling the **Gx1164Panel** 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 **Gx1164Panel** function is also used by the GXPRESANEL.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 (GXPRES.DLL or GXPRES64.DLL) and the HW device driver files located in the HW folder) can be shipped with the application. Typically, the DLLs should be copied to the Windows System directory. The HW device driver files should be installed using a special setup program HWSETUP.EXE that is provided with GXPRES driver files. Alternatively, you can provide the GXPRES disk to be installed along with the board.

Sample Programs

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

To run, Enter the following command line:

GXPRES <PciSlot> <specific_command>

Where:

<slot>	PCI/PXI Explorer slot number where the board resides.
<channel>	Channel number, 1 to 8.
<operation>	Operation code: SS=read port direction SD=write port direction RS=read port value RD=write port value
<type>	Channel type 1 for single channel, 2 double channel.
<value>	Channel resistance value.

Sample Program Listing

```

/*****
FILE           : GxPResExampleC.cpp

PURPOSE       : WIN32/LINUX example program for GX1164 boards
                using the GXPRES driver.

CREATED       : Mar 2002

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

COMMENTS      :

To compile the example:

1. Microsoft VC++
   Load GxPResExampleC.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 GxPResExampleC.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 -fGxPResExampleC.mk [CFG=Release[64] | Debug[64]] [rebuild |
   clean]
*****/

#ifdef __GNUC__
#include "windows.h"
#endif
#include "GxPRes.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#ifdef __BORLANDC__
#pragma hdrstop
#include <condefs.h>
USELIB("GxPResBC.lib");
USERC("GxPResExampleC.rc");
#endif

/*****
//
//           DisplayMsg
//
void DisplayMsg(PCSTR lpszMsg)
{
#ifdef __GNUC__
    MessageBeep(0);
    MessageBox(0, lpszMsg, "GxPRes example program", MB_OK);
#else
    printf("\r\nGxPRes 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(
        "This example shows how to set the GX1164 resistance channels:\r\n\r\n"

        "Usage:\r\n"
        "GxPResExample <slot> <channel> <type> <value>"

        "\r\n\r\nWhere : \r\n"
        "<slot> - PCI/PXI slot number as shown by the PXI explorer\r\n"
        "<channel> - channel number 1 to 8\r\n"
        "<type> - 1 for single 2 for double channel\r\n"
        "<value> - resistance value\r\n"

        "\r\nTo change command line under Windows:\r\n"
        "\tRight click on the example shortcut from the start menu\r\n"
        "\tand type the new command line\r\n"
    );
    exit(1);
}

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

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

//*****
//          MAIN
//
// This main functin receives three parameters
//
// Board (PXI/PXI Explorer) slot number:
//                               1 - 64
// Channel number:
//                               1 - 8
// Channel type:
//                               1=single channel
//                               2=double channel
// Resistance value:
//                               floating point
//
// Example:
//                               GxPResExampleC 1 2 1234
// Sets double channel 1/2 to 1234 ohm
//*****
int main(int argc, char ** argv)
{
    short    nSlot;                // Board slot number
    short    nChannel;            // channel number
    short    nDouble;             // single or double
    double   dValue, dActual;     // resistance value or
    short    nHandle;            // Board handle
    short    nStatus;            // Returned status

```

```

// Check number of arguments rcvd
if (argc!=4) DisplayUsage();

// Parse command line parameters
nSlot=(SHORT)strtol(++argv, NULL, 0);
nChannel=(SHORT)strtol(++argv, NULL, 0);
nDouble=(SHORT)strtol(++argv, NULL, 0);
dValue=strtod(++argv, NULL);

if (nSlot<0) DisplayUsage();
if (nChannel<1 || nChannel>8) DisplayUsage();
if (nDouble!=1 && nDouble!=2) DisplayUsage();

Gx1164Initialize(nSlot, &nHandle, &nStatus);
CheckStatus(nStatus);

switch (nDouble)
{
    case 1:
        // Set single channel
        Gx1164SetSingleChannelResistance(nHandle, nChannel, dValue, &nStatus);
        CheckStatus(nStatus);
        Gx1164GetSingleChannelResistance(nHandle, nChannel, &dActual, &nStatus);
        printf("Set Single Channel %d to %f, Actual Resistance is %f.\n",\
            nChannel, dValue, dActual);
        break;
    case 2:
        // Set double channel
        Gx1164SetDoubleChannelResistance(nHandle, nChannel, dValue, &nStatus);
        CheckStatus(nStatus);
        Gx1164GetDoubleChannelResistance(nHandle, nChannel, &dActual, &nStatus);
        printf("Set Double Channel %d to %f, Actual Resistance is %f.\n",\
            nChannel, dValue, dActual);
        break;
}
return 0;
}

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

```


Chapter 5 - Functions Reference

Introduction

The GX1164 driver functions reference chapter is organized in alphabetical order. Each function description contains the function name; purpose, syntax, parameters description and type followed by Comments, an Example (written in C), and a See Also sections.

All function parameters syntax follows the same rules:

- Strings are ASCIIZ (null or zero character terminated).
- The first parameter of most functions is *nHandle* (16-bit integer). This parameter is required for operating the board and is returned by the **Gx1164Initialize** function. The *nHandle* is used to identify the board when calling a function for programming and controlling the operation of that board.
- All functions return a status with the last parameter named *pnStatus*. The *pnStatus* is zero if the function was successful, or less than a zero on error. The description of the error is available using the **GxPRESGetErrorString** function or by using a predefined constant, defined in the driver interface files: GXPRES.H, GXPRES.BAS, GXPRES.VB, GXPRES.PAS or GX1164.DRV.
- Parameter name are prefixed as follows:

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

Table 5-1: Parameter Name Prefixes

GX1164 Functions

The following list is a summary of functions available for the GX1164:

Driver Functions	Description
General	
Gx1164Initialize	Initializes the driver for the specified slot using the HW device driver.
Gx1164InitializeVisa	Initializes the driver for the specified slot using VISA.
Gx1164Panel	Opens a virtual panel used to interactively control the GX1164.
Gx1164Reset	Resets the GX1164 board to its default settings.
GxPResGetDriverSummary	Returns the driver summary and version.
GxPResGetErrorString	Returns the error string associated with the specified error number.
Resistance Functions	
Gx1164GetBoardSummary	Returns a string describing the board S/N, firmware version and calibration date.
Gx1164GetChannelMode	Returns the channel mode: single or double.
Gx1164GetChannelRelays	Returns the specified channel resistors relay settings.
Gx1164GetChannelResiatnceRange	Returns the single or double channel resistance range.
Gx1164GetDoubleChannelResistance	Returns the double channel resistance.
GX1164GetSingleChannelResistance	Returns the single channel resistance.
Gx1164GetWiperMode	Returns whether the group is in wiper mode.
Gx1164SetChannelRelays	Connect the specified channel resistors relays.
Gx1164SetDoubleChannelResistance	Sets the double channel resistance.
Gx1164SetSingleChannelResistance	Sets the single channel resistance.
Gx1164SetWiperMode	Sets the group wiper mode.

Gx1164GetBoardSummary

Purpose

Returns the board S/N, firmware version and revision, and calibration time.

Syntax

Gx1164GetBoardSummary (*nHandle*, *pszSummary* *nSummaryMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>pszSummary</i>	PSTR	Buffer to the returned driver summary string.
<i>nSummaryMaxLen</i>	SHORT	The size of the summary string buffer.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The returned string is: " GX1164, Version 0.1, Revision 0, S/N 1, last calibration time Tue Jul 23 18:22:53 2002".

Example

The following example prints the driver version:

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

Gx1164GetBoardSummary (nHandle, sz, sizeof sz, &nStatus);
printf("Board Summary: %s", sz);
```

See Also

GxPresGetDriverSummary, **GxPresGetErrorString**

Gx1164GetChannelMode

Purpose

Returns the channel mode: single or double.

Syntax

Gx1164GetChannelMode (*nHandle*, *nChannel*, *pbDouble*, *pnStatus*);

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nChannel</i>	SHORT	Channel number: 1-8.
<i>pbDouble</i>	PBOOL	Channel mode: 1 (TRUE) for double, 0 (FALSE) for single.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx1164GetChannelMode** to determine if the channel is set to single mode or double. In Double mode the odd number channel and the consecutive channel numbers are combined to provide a more accurate resistance.

Example

The following example determines if channel 1 and 2 are in single mode:

```
Gx1164GetChannelMode(nHandle, 2, &bDouble, nStatus);
if (bDouble==0)
    printf("Channel 1 and 2 are in single mode");
```

See Also

Gx1164SetSingleChannelResistance, **Gx1164SetDoubleChannelResistance**

Gx1164GetChannelRelays

Purpose

Returns the specified channel resistors relays settings.

Syntax

Gx1164SetChannelRelays (*nHandle*, *nChannel*, *pucRelays*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nChannel</i>	SHORT	Channel number: 1-8.
<i>pucRelays</i>	PBYTE	Channel resistors relays (1-8), each bit represents a relay state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function provides control over the settings of each resistor relay in the channel to be set On or Off.

Example

The following example returns channel 1 resistor relays settings:

```
BYTE ucRelays;
Gx1164GetChannelRelays (nHandle, 1, &ucRelays, &nStatus)
```

See Also

Gx1164SetChannelRelays, **Gx1164GetSingleChannelResistance**, **Gx1164SetDoubleChannelResiatnce**

Gx1164GetChannelResistanceRange

Purpose

Returns the single or double channel resistance range.

Syntax

Gx1164GetChannelResistanceRange (*nHandle*, *nChannel*, *bDouble*., *pdMin*, *pdMax*, *pnStatus*);

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nChannel</i>	SHORT	Channel number: 1-8.
<i>bDouble</i>	BOOL	Channel mode: 1 (TRUE) for double, 0 (FALSE) for single.
<i>pdMin</i>	PDOUBLE	Returned the smaller value of the range.
<i>pdMax</i>	PDOUBLE	Returned the highest value of the range.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx1164GetChannelResistanceRange** to determine the channel range. Typically, the board factory resistor are set to 1-65535 for double channels, 1-255 for single channel 1,3, 5 and 7, and 256-65280 for single channels 2, 4, 6 and 8. Different value can be set upon special order, call Marvin Test Solutions for more details.

Example

The following example determines if channel 1 and 2 are in single mode:

```
Gx1164GetChannelResistanceRange(nHandle, 2, TRUE, &dMin, &dMax, nStatus);
Printf("Channel 1/2 resistance is %g to %g", dMin, dMax);
```

See Also

Gx1164SetSingleChannelResistance, **Gx1164SetDoubleChannelResistance**

Gx1164GetDoubleChannelResistance

Purpose

Returns the double channel resistance.

Syntax

Gx1164GetDoubleChannelResistance (*nHandle*, *nChannel*, *pdwValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nChannel</i>	SHORT	Channel number: 1-8.
<i>pdwValue</i>	PDOUBLE	Returned resistance value.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx1164GetDoubleChannelResistance** to retrieve the actual channel resistance value.

Example

The following example sets the channel resistance and verifies its actual value:

```
Gx1164SetSingleChannelResistance(nHandle, 1, 131.0, nStatus);
Gx1164GetSingleChannelResistance(nHandle, 1, &dValue, nStatus);
if (131-dValue>3)
    printf("Resistance is not accurate");
```

See Also

Gx1164SetSingleChannelResistance

Gx1164GetSingleChannelResistance

Purpose

Returns the single channel resistance.

Syntax

Gx1164GetSingleChannelResistance (*nHandle*, *nChannel*, *pdValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nChannel</i>	SHORT	Channel number: 1-8.
<i>pdValue</i>	PDOUBLE	Returned resistance value.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx1164GetSingleChannelResistance** to retrieve the actual double channel resistance value.

Example

The following example sets the double channel 3/4 resistance and verifies its actual value:

```
Gx1164SetDoubleChannelResistance(nHandle, 3, 12131.0, nStatus);
Gx1164GetSingleChannelResistance(nHandle, 3, &dValue, nStatus);
if (12131-dValue>5)
    printf("Resistance is not accurate");
```

See Also

Gx1164SetDoubleChannelResistance

Gx1164GetWiperMode

Purpose

Returns whether the group is in wiper mode.

Syntax

Gx1164GetWiperMode (*nHandle*, *nGroup*, *pbWiper*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nGroup</i>	SHORT	Group number: 0 for channels 1 to 4 and 1.for channels 5-8
<i>pbWiper</i>	PBOOL	Returned mode: 0 (FALSE): Off. 1 (TRUE): On.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx1164GetWiperMode** to determine if the specified group is in wiper mode.

Example

The following example sets the double channel 3/4 resistance and verifies its actual value:

```
Gx1164SetWiperMode(nHandle, 0, 1, nStatus);
Gx1164GetWiperMode(nHandle, 0, &bWiper, nStatus);
if (bWiper!=1)
    printf("Error : Wiper mode not set...");
```

See Also

Gx1164SetWiperMode

Gx1164Initialize

Purpose

Initializes the driver for the specified PXI slot using the HW device driver.

Syntax

Gx1164Initialize (*nSlot*, *pnHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nSlot</i>	SHORT	GX1164 board slot number.
<i>pnHandle</i>	PSHORT	Returned Handle for a GX1164 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Marvin Test Solutions 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).

Example

The following example initializes two GX1164 boards at slot 1 and 2.

```
SHORT nHandle1, nHandle2, nStatus;
Gx6616Initialize (1, &nHandle1, &nStatus);
Gx6616Initialize (2, &nHandle2, &nStatus);
if (nHandle1==0 || nHandle2==0)
{
    printf("Unable to Initialize the boards")
    return;
}
```

See Also

Gx1164InitializeVisa, **GxPresGetErrorString**, **Gx1164Reset**

Gx1164InitializeVisa

Purpose

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

Syntax

Gx1164InitializeVisa (*szVisaResource*, *pnHandle*, *pnStatus*)

Parameters

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

Comments

The **Gx1164InitializeVisa** opens a VISA session to the specified resource. The function uses the default VISA provider configured in your system to access the board. You must ensure that the default VISA provider support PXI/PCI devices and that the board is visible in the VISA resource manager before calling this function.

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

Using chassis, slot: "PXI0::CHASSIS1::SLOT5"

Using the PCI Bus/Device combination: "PXI9::13::INSTR" (bus 9, device 9).

Using alias: "COUNTER1". 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;
Gx1164InitializeVisa("PXI5::11::INSTR", &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

Gx1164Initialize, **GxPResGetErrorString**, **Gx1164Reset**

Gx1164Panel

Purpose

Opens a virtual panel used to interactively control the GX1164.

Syntax

Gx1164Panel (*pnHandle hwndParent, nMode, phwndPanel, pnStatus*)

Parameters

Name	Type	Comments
<i>pnHandle</i>	PSHORT	Handle to a GX1164 board.
<i>hwndParent</i>	HWND	Panel parent window handle. A value of 0 sets the desktop as the parent window.
<i>nMode</i>	SHORT	The mode in which the panel main window is created. 0 for modeless window and 1 for modal window.
<i>phwndPanel</i>	HWND	Returned window handle for the panel.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

If the mode is set to modal dialog (*nMode=1*), the panel will disable the parent window (*hwndParent*) and the function will return only after the window was closed by the user. In that case, the *pnHandle* may return the handle created by the user using the panel Initialize dialog. This handle may be used when calling other GX1164 functions.

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

Example

The following example opens the panel in modal mode:

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

Gx1164Panel(&nHandle, 0, 1, &dwPanel, &nStatus);
```

See Also

Gx1164Initialize, GxPResGetErrorString

Gx1164Reset

Purpose

Resets the GX1164 board to its default settings.

Syntax

Gx1164Reset (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

After reset channels are set to maximum resistance (all relays are opened) in single mode. Wiper mode is off.

Example

The following example initializes and resets the GX1164 board:

```
Gx1164Initialize (1, &nHandle, &nStatus);  
Gx1164Reset (nHandle, &nStatus);
```

See Also

Gx1164Initialize

Gx1164SetDoubleChannelResistance

Purpose

Sets the double channel resistance.

Syntax

Gx1164SetDoubleChannelResistance (*nHandle*, *nChannel*, *dValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nChannel</i>	SHORT	Channel number: 1-8.
<i>dValue</i>	DOUBLE	Resistance value. See comments.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The following table describes the range and resolution for each of the available double channels:

Model	Channel	Range	Resolution
GX1164	1 / 2	1Ω-64KΩ	1Ω
	3 / 4	1Ω-64KΩ	1Ω
	5 / 6	1Ω-64KΩ	1Ω
	7 / 8	1Ω-64KΩ	1Ω
GX11642	1 / 2	2Ω-128KΩ	2Ω
	3 / 4	2Ω-128KΩ	2Ω
	5 / 6	2Ω-128KΩ	2Ω
	7 / 8	2Ω-128KΩ	2Ω
GX1164-4	1 / 2	4Ω-256KΩ	4Ω
	3 / 4	4Ω-256KΩ	4Ω
	5 / 6	4Ω-256KΩ	4Ω
	7 / 8	4Ω-256KΩ	4Ω
GX1164-8	1 / 2	8Ω-512KΩ	8Ω
	3 / 4	8Ω-512KΩ	8Ω
	5 / 6	8Ω-512KΩ	8Ω
	7 / 8	8Ω-512KΩ	8Ω

*Passing 1 or 2 to the *nChannel* parameter is the same.

Use the **Gx1164GetDoubleChannelResistance** to determine the actual value set. When double channel resistance is used, intermediate pins are still connected. For example when passing *nChannel* as 1 or 2 pins 15 and 3 are still connected to the resistor.

Example

The following example sets channel 1 / 2 to 1271 Ω

```
Gx1164SetDoubleResistance(nHandle, 3, 1271.0, &nStatus)
```

See Also

Gx1164GetDoubleChannelResistance, Gx1164SetSingleChannelResistance

Gx1164SetChannelRelays

Purpose

Sets the specified channel resistors relays.

Syntax

Gx1164SetChannelRelays (*nHandle*, *nChannel*, *ucRelays*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nChannel</i>	SHORT	Channel number: 1-8.
<i>ucRelays</i>	BYTE	Channel resistors relays (1-8), each bit represents a relay state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function provides control over the settings of each resistor relay in the channel to be set On or Off.

Example

The following example close resistor relays 1-3 in channel 1:

```
Gx1164SetChannelRelays (nHandle, 1, 7, &nStatus)
```

See Also

Gx1164GetChannelRelays, **Gx1164GetSingleChannelResistance**, **Gx1164SetDoubleChannelResistance**

Gx1164SetSingleChannelResistance

Purpose

Sets the single channel resistance.

Syntax

Gx1164SetSingleChannelResistance (*nHandle*, *nChannel*, *dValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nChannel</i>	SHORT	Channel number: 1-8.
<i>dValue</i>	DOUBLE	Resistance value. See comments.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The following table describes the range and resolution for each of the available double channels:

Model	Channels	Range	Resolution
GX1164	1, 3, 5, 7	1Ω-256Ω	1Ω
	2, 4, 6, 8	256Ω-64KΩ	1Ω
GX11642	1, 3, 5, 7	2Ω-512KΩ	2Ω
	2, 4, 6, 8	512Ω-128KΩ	2Ω
GX1164-4	1, 3, 5, 7	4Ω-1024Ω	4Ω
	2, 4, 6, 8	1024Ω-256KΩ	4Ω
GX1164-8	1, 3, 5, 7	8Ω-2048Ω	8Ω
	2, 4, 6, 8	2048Ω-512KΩ	8Ω

Use the **Gx1164GetSingleChannelResistance** to determine the actual value set.

Example

The following example sets channel 1 / 2 to 1271 Ω

```
SetDoubleResistance(nHandle, 3, 1271.0, &nStatus)
```

See Also

Gx1164GetSingleChannelResistance, **Gx1164SetDoubleChannelResistance**

Gx1164SetWiperMode

Purpose

Sets the group wiper mode.

Syntax

Gx1164SetWiperMode (*nHandle*, *nGroup*, *pbWiper*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX1164 board.
<i>nGroup</i>	SHORT	Group number 0-1 for A and B.
<i>pbWiper</i>	PBOOL	Returned mode: 0 (FALSE): Off. 1 (TRUE): On.
<i>pnStatus</i>	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx1164SetWiperMode** when working in potentiometer mode. When in wiper mode the 2 lower channels are connected to the upper channels of the group.

Example

The following example sets group 0 (A) to wiper mode

```
Gx1164GetWiperMode(nHandle, 0, &bWiper, &nStatus)
If (bWiper)
    Printf("We are in Wiper Mode");
```

See Also

Gx1164GetWiperMode

GxPResGetDriverSummary

Purpose

Returns the driver name and version.

Syntax

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

Parameters

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

Comments

The returned string is: "GXPRES Driver for GX1164. Version 1.0, Copyright © Marvin Test Solutions 2002".

Example

The following example prints the driver version:

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

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

See Also

Gx1164GetBoardSummary, **GxPResGetErrorString**

GxPResGetErrorStrings

Purpose

Returns the error string associated with the specified error number.

Syntax

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

Parameters

Name	Type	Comments
<i>nError</i>	SHORT	Error number.
<i>pszMsg</i>	PSTR	Buffer to the returned error string.
<i>nErrorMaxLen</i>	SHORT	The size of the error string buffer.
<i>pnStatus</i>	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

- 1 Board does not exist in this slot
- 2 Unable to open the HW device/Service
- 3 Different board exist in the specified PCI slot
- 4 PCI slot not configured properly. You may configure it by using the **PCIExplorer** from the Control Panel
- 5 Unable to register the PCI device
- 6 Unable to allocate system resource or memory for the PCI device
- 7 Too many boards
- 8 Unable to create panel
- 9 Unable to create a Windows timer
- 10 Invalid board EEPROM
- 11 Not in calibration mode
- 12 Board is not calibrated

General Parameter Errors

- 20 Invalid error
- 20 Invalid parameter
- 21 Invalid PCI slot number
- 22 Invalid board handle
- 23 Invalid string length number

Parameter Errors

- 40 Invalid channel number
- 41 Invalid group number
- 42 Invalid mode
- 43 Invalid value

Example

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

```
CHAR    sz[256];
SHORT   nStatus, nHandle;
..
Gx1164Initialize (3, &Handle, &Status);
if (nStatus<0)
{   GxPResGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz);          // prints the error string returns
}
```


Index

- .
- .NET ii
- A**
- Applications.....4
- Architecture 1, 5
- ATEasyii, 20, 21, 25, 26
- B**
- Board Description..... 1, 4
- Board Handle.....28
- Borland 21, 25, 26
- Borland-Delphi26
- C**
- C#25
- C/C++21, 25
- C++25
- Calibration7
- Connectors..... 18, 19, 20
- Corrupt files.....23
- D**
- Delphiii, 21, 25, 26
- Directories20
- Distributing.....29
- Driver
 - Directory.....20
 - Files20
- Driver Version28
- E**
- Error Handling.....28
- Error-Handling28
- Example21
- F**
- Features.....3
- G**
- GX1164 1, 4, 5
 - Software.....7
 - Gx1164GetBoardSummary35
 - Gx1164GetChannelMode36
 - Gx1164GetChannelRelays37
 - Gx1164GetChannelResistanceRange38
 - Gx1164GetDoubleChannelResistance.....39
 - Gx1164GetSingleChannelResistance40
 - Gx1164GetWiperMode41
 - Gx1164Initialize9, 27, 28, 33, 34, 42
 - Gx1164InitializeVisa.....9, 27, 28, 34, 43
 - Gx1164Panel29, 44
 - Gx1164Reset28, 45
 - Gx1164SetChannelRelays48
 - Gx1164SetDoubleChannelResistance46
 - Gx1164SetSingleChannelResistance.....49
 - Gx1164SetWiperMode.....50
- GXPRES..... 1, 14
 - Driver-Description.....25
 - Header-file25
 - Help-File-Description21
 - Panel-File-Description.....21
 - Supported-Development-Tools.....25
- GXPRES.BAS21, 25
- GXPRES.DLL21, 25, 26
- GXPRES.EXE14
- GXPRES.H.....21, 25
- GXPRES.LIB21, 25
- GXPRES.lib.....26
- GXPRES.PAS21, 26
- GXPRES64.LIB25
- GXPRESBC.LIB21
- GxPResGetDriverSummary27, 28, 51
- GxPResGetErrorString28, 33, 52, 53
- GXPRESANEL.EXE21, 29
- GXPS.BAS25

H

Handle 17, 18, 27, 28

HW 18, 20, 25, 29

I

Installation Directories 20

Installation: 17, 18

J

J3 Connector 19

J7 19

Jumpers 19

L

LabView 21, 23, 26

LabView/Real Time 26

Linux 26

Listing 30

N*nHandle* 26, 28**O**

OnError 26

P

Panel 8, 10, 11, 21, 23, 27, 29, 44, 52

Part / Model Number 20

Pascal 21, 25, 26

PCI 20

Plug & Play 18

pnStatus 28, 33

port byte 50

Program-File-Descriptions 21

Programming

Borland-Delphi 26

Error-Handling 28

Panel-Program 29

Programming the Board 25

PXI 13, 16, 17, 18, 27

PXI/PCI Explorer 9, 16, 27, 28, 42, 43

PXIeSYS.INI 9

PXISYS.INI 9

R

README.TXT 20, 21

Readme-File 21

Reset 28

S

Sample 29, 30

SCSI 19

Setup 14, 20, 21, 23

Setup Maintenance 23

Setup-and-Installation 13

Slot 9, 13, 17, 18, 27

Software 7

Specifications 1, 6

System

Directory 20

System-Requirements 13

U

Using the GXPRES driver functions 27

V

Virtual Panel 8, 9, 10, 11, 14, 21, 27

Initialize Dialog 9

VISA 9, 27, 28, 34, 42, 43

Visual Basic ii, 25

Visual C# 25

Visual C++ ii, 21, 25

Voltage Rails

Range 5