

***GX7200***

***GX7210***

***GX7202***

***GX7212***

**GX7200 3U PXIe Chassis Series  
GxChassis Software**

***User's Guide***

Last updated: Jan, 26, 2017





## 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 diskette for installation. Store the original diskette 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. Software products and accessories are warranted for 3 months. Unless covered by software support or maintenance agreement. Marvin Test Solutions shall repair or replace (at its discretion) any defective product during the stated warranty period. The software warranty includes any revisions or new versions released during the warranty period. Revisions and new versions may be covered by a software support agreement. If you need to return a board, please contact Marvin Test Solutions Customer Technical Services department via <http://www.marvintest.com/magic> the Marvin Test Solutions on-line support system.

## If You Need Help

---

Visit our web site at <http://www.marvintest.com> for more information about Marvin Test Solutions products, services and support options. Our web site contains sections describing support options and application notes, as well as a download area for downloading patches, example, patches and new or revised instrument drivers. To submit a support issue including suggestion, bug report or question 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 7:30 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 © 2003-2017 by 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 (prior name is Geotest - Marvin Test Systems Inc.)
C++ Builder, Borland C++, Pascal and Delphi	Embarcadero Technologies Inc.
LabView, LabWindows™/CVI	National Instruments
Microsoft Developer Studio, Microsoft Visual C++, Microsoft Visual Basic, .NET, and Windows.	Microsoft Corporation

All other trademarks are the property of their respective owners.

# Table of Contents

Safety and Handling.....	i
Warranty .....	i
If You Need Help.....	i
Disclaimer .....	i
Copyright .....	i
Trademarks .....	ii
<b>Table of Contents.....</b>	<b>iii</b>
<b>Chapter 1 - Introduction .....</b>	<b>1</b>
Manual Scope and Organization.....	1
Manual Scope.....	1
Manual Organization.....	1
Conventions Used in this Manual .....	2
<b>Chapter 2 - Overview .....</b>	<b>3</b>
Introduction.....	3
GX7200 Series Features .....	4
The PXI Standard.....	5
PXI and PCI Express Overview.....	5
GX72xx PXI Express Features .....	6
GX72xx Models.....	6
Optional Equipment.....	6
Chassis Description – Front View (GX7200 & GX7202) .....	7
Chassis Description – Rear View.....	9
PXI/PXIe Slots.....	11
Gx72xx Boards .....	12
3U PXI and Compact PCI Boards.....	12
3U PXI Hybrid Boards.....	13
3U PXI Express Boards .....	13
3U PXI Express System Timing Boards .....	14
3U PXI Express System Board .....	14
PXI Bus Segments .....	15
System Controller Slot.....	15
PXI Express System Timing Slot .....	15
Local Bus .....	16
Trigger Bus .....	17
Star Trigger Lines .....	18

System Reference Clock .....	19
System Power Supply – GX7200 / GX7210 / GX7202 / GX7212 .....	19
Power Distribution .....	19
Overview of the GxChassis Software .....	20
GxChassis driver Features .....	20
Virtual Panel Description.....	21
Virtual Panel Initialize Dialog .....	22
Virtual Panel Temperature Settings .....	23
Slots Temperatures (Group Box) .....	23
Virtual Panel PXI Trigger Lines .....	25
Virtual Panel Advanced page.....	26
Virtual Panel About Page.....	27
<b>Chapter 3 - Setup and Installation.....</b>	<b>29</b>
Unpacking and Inspecting the Chassis.....	29
Mounting Information.....	29
Line Voltage Selection.....	29
Chassis Installation .....	29
GX72xx Master and Slave Configurations .....	30
Installation of the GxChassis Software .....	31
Configuring Your PXI System using the PXI/PCI Explorer.....	32
Installing PXI Instruments .....	33
PXI Instrument Removal .....	34
Using External Instruments.....	34
Installation Directories.....	35
Driver Files Description.....	35
Driver File and Virtual Panel .....	35
Interface Files.....	35
On-line Help and Manual.....	36
ReadMe File.....	36
Example Programs .....	36
Setup Maintenance Program .....	37
<b>Chapter 4 - Programming the Chassis.....</b>	<b>39</b>
Overview.....	39
The GxChassis Driver.....	39
Programming Using C/C++ Tools .....	39
Programming Using Visual Basic.....	39
Programming Using Pascal/Delphi .....	39

Programming GxChassis Boards Using ATEasy®.....	40
Using the GxChassis Driver Functions .....	41
Chassis Handle .....	41
Error Handling .....	41
Driver Version.....	41
Panel.....	41
Distributing the Driver .....	41
Sample Programs .....	42
Sample Program Listing .....	42
<b>Chapter 5 - Functions Reference.....</b>	<b>49</b>
Introduction.....	49
GxChassis Functions.....	50
GxChassisGetAlarmMode .....	51
GxChassisGetAlarmTemperature .....	52
GxChassisGetBoardSummary .....	53
GxChassisGetDriverSummary.....	54
GxChassisGetErrorString .....	55
GxChassisGetPowerSupplies Voltages.....	57
GxChassisGetFanSpeed.....	58
GxChassisGetFanThresholdTemperatures.....	59
GxChassisGetPxiTriggerLine .....	60
GxChassisGetPxiTriggerLineLevels .....	62
GxChassisGetShutdownTemperature .....	63
GxChassisGetSlotsTemperatures.....	64
GxChassisGetSlotsTemperaturesStates .....	65
GxChassisGetSlotsTemperaturesStatistics .....	66
GxChassisGetSlotTemperature .....	67
GxChassisGetTemperatureScale.....	68
GxChassisGetTemperatureThresholdMode .....	69
GxChassisInitialize .....	70
GxChassisPanel .....	71
GxChassisRecallSettings .....	72
GxChassisResetPxiTriggerLines .....	73
GxChassisSetAlarmMode.....	74
GxChassisSetAlarmTemperature.....	75
GxChassisGetFanSpeed.....	76
GxChassisSetFanThresholdTemperatures .....	77

GxChassisSetPxiTriggerLine.....	78
GxChassisSetShutdownTemperature.....	80
GxChassisSetSlotsTemperaturesStates.....	81
GxChassisSetTemperatureScale.....	82
GxChassisSetTemperatureThresholdMode.....	83
<b>Appendix A – Specifications.....</b>	<b>85</b>
Cooling.....	86
GX7200 / GX7210 / GX7202 / GX7212.....	86
Temperature Monitoring.....	86
Power Supply Monitoring.....	86
PXI 10 MHz Clock and PXIe 100 MHz Clock.....	86
Slots.....	86
Physical Characteristics.....	87
Environmental and Compliance.....	87
<b>Appendix B – PXI Slots, Pin Outs.....</b>	<b>89</b>
Bus Signal Names.....	90
Backplane Connector Layouts by Slot Type.....	91
XJ4 Connector Pin Out for System Controller Slot.....	92
XP3 Connector Pin Out for System Controller Slot (4-Link Configuration).....	92
XP2 Connector Pin Out for System Controller Slot (4-Link Configuration).....	93
XJ1 Connector Pin Out for System Controller Slot.....	93
XP4 Connector Pin Out for System Timing Slot.....	94
XP3 Connector Pin Out for System Timing Slot.....	94
TP2 Connector Pin Out for System Timing Slot.....	95
XP4 Connector Pin Out for Hybrid Slots.....	96
XP3 Connector Pin Out for Hybrid Slots.....	96
P1 Connector Pin Out for Hybrid Slots.....	97
P1 Connector Pin Out for PXI-1 Peripheral Slots.....	98
P2 Connector Pin Out for PXI-1 Peripheral Slots.....	99
<b>Appendix C – Rear Panel Connector Layout.....</b>	<b>101</b>
Ethernet Connector.....	101
USB Connector.....	102
<b>Appendix D – Model Numbers.....</b>	<b>103</b>
Chassis and Controller Model Numbers.....	103
Chassis Accessory Model Numbers.....	103
<b>Index.....</b>	<b>105</b>



# 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 GX72xx PXI chassis. This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and some understanding of digital to analog conversion.

This manual also provides programming information using the GxChassis driver. Therefore, good understanding of programming development tools and languages may be necessary.





### Manual Organization

The GX7200 PXIe chassis manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX72xx PXIe chassis manual. Lists all the supported boards and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX72xx PXIe chassis features, chassis description, its architecture, specifications and the GxChassis panel description and operation.
Chapter 3 – Installation and Connections	Provides instructions on how to install the GX7200's accompanying GxChassis software.
Chapter 4 – Programming the Board	Provides a listing of GxChassis 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 GxChassis functions. Each function is described along with its syntax, parameters, and special programming comments. Samples are given for each function.
Appendix A – Specifications GX72xx	Provides the GX72xx specifications.
Appendix B – PXI Slots Pin Outs GX72xx	Describes the P1, P2, XJ1, XP2, XP3, XP4, TP1, and TP2 connector pin outs for the GX72xx backplane.
Appendix C – Rear Panel Connector Layout GX72xx	Provides information on the rear panel connectors of the GX7200, GX7202, GX7210, and GX7212,
Appendix D – Model Numbers GX72xx	Describes the chassis model numbers.

## 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.
<b>Bold type</b>	Words or characters you type as the manual instructs, programming function names and window control 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

---

Thank you for selecting the GX7200 series (GX72xx) PXI Express chassis, (GX7200, GX7210, GX7202 and GX7212). The GX72xx chassis is designed for test, data acquisition, process control, and factory automation applications. The GX72xx is a 3U, 21-slot PXI Express chassis for benchtop or rack mount applications. The GX72xx is based on the CompactPCI Express™ (cPCIe) and PXI Express™ (PCI eXtensions for Instrumentation Express) standards and accommodates up to (8) 3U PXI or cPCI instruments, (8) 3U PXI or cPCI Hybrid instruments, (4) 3U PXI Express or cPCI Express instruments, and a PXI Express controller or bus expander in slot 1. The design of the GX72xx allows integration of PXI, cPCI, PXI Express and cPCI Express boards from any vendor. The GX72xx PXI master chassis (GX7200 or GX7202) is supplied with the pre-installed GxChassis software. The GxChassis software supports the chassis' Smart functions, which includes the monitoring of chassis temperature and power supplies as well as programming/routing of the PXI trigger lines. The GxChassis software provides API functions for controlling all of the PXI chassis' capabilities as well as providing a soft front panel that supports these same capabilities. The software is also included with the Marvin Test Solutions' Product CD, which is supplied with every chassis.

The GxChassis software driver supports the programming and settings of shutdown and alarm temperature limits, slot temperature measurement, control of the PXI trigger lines' directions and states, measurement of system power supply voltages and monitoring / control of the system fans. In addition, the driver enables the user to save those settings to an on-board EEPROM that can then be used as default settings on power up. The user can also set the temperature scale used for programming or monitoring of any temperature value.

The following figures show the GX7200 master chassis with a Marvin Test Solutions controller.



**Figure 2-1: GX7200 Instrumentation Chassis**

## GX7200 Series Features

---

The GX72xx models offer the following features:

- 21PXI/cPCI, 3U slots. Slot 1 is dedicated for an embedded controller or for a remote controller. Slots 2, 9, 10, and 21 are dedicated slots for use with PXIe/cPCIe x4 peripheral modules. Slots 2 through 8 and 11 through 20 support the PXI Star Trigger. Slot 9 supports the PXIe timing slot controller.
- Flexible slot configuration offers 8 PXI, 8 PXI/PXIe Hybrid, and 4 PXIe slots.
- 4x4 PXIe lane architecture supports 8 GB/s system bandwidth and peer to peer communication.
- Built-in hard disk drive for embedded controller configurations (GX7200 / GX7202).
- Integral Smart functions provide per slot temperature monitoring, system power supply monitoring, fan monitoring / control and PXI trigger mapping.
- 755 W power supply
- Full compliance with PXI Express Hardware Specification revision 1.0 and the accompanying ECN-1 revision 2.0. Supports features such as trigger bus, star trigger, differential triggers, local bus, and system clock.
- Interoperability with 32-bit 33MHz CompactPCI.
- Front-loading configuration. Boards are inserted from the front for simplified maintenance.
- Board connectors face the front side of the platform enabling easy access to board connectors and cables and a short path to the interface.
- (4) high capacity fans located under the card cage ensures adequate airflow for all plug-in instruments
- Separate fan provides cooling for the chassis power supply
- PXI Express back plane incorporates the local bus, trigger buses, and 10 & 100 MHz reference clocks.
- Support for external instrumentation and devices using the built-in serial, USB and Ethernet interfaces.
- Additional chassis may be daisy-chained using a PXI bus expander.
- Innovative PXI-Explorer™ and GxChassis Software and Panel provides easy configuration and programming tools for the chassis and instruments.

When bundled with *ATEasy*™, Marvin Test Solutions' award-winning software development environment, the GX72xx provides a complete system for any test and measurement application.

## The PXI Standard

---

PCI eXtensions for Instrumentation (PXI) modular instrumentation delivers a rugged, PC-based, and high-performance measurement and automation system. With PXI you benefit from the low cost, performance, and flexibility of the latest PC technology and the benefits of an open industry standard. PXI combines standard PC technology from the CompactPCI™ specification with integrated timing and triggering to deliver a rugged platform with up to a 10X performance improvement over older architectures. PXI has become the industry standard for measurement and automation applications.

The PXI standard was developed in response to the needs of test systems developers and users who required a new platform that is high-performance, functional and reliable, yet easy to integrate and use. By leveraging the PCI, CompactPCI, Microsoft Windows, and VXI standards, PXI combines technologies for PC-based test and measurement, instrumentation, and industrial automation. Since PXI is a PC-based platform, it maintains software compatibility with industry-standard personal computers, as well as all PC-Based operating systems, software tools, and instrument drivers. PXI is fully compatible with existing operating systems and software; it also employs the Virtual Instrument Software Architecture (VISA) standard that was created by the VXI plug & play System Alliance (see <http://www.vxi.org/>). VISA is used to locate and communicate with PXI, serial, VXI, and GPIB peripheral modules and is supported by test development software packages such as *ATEasy*™, LabVIEW™, LabWindows/CVI™ and Agilent VEE™.

PXI expands upon the PCI bus and allows PXI users to realize all the benefits of PCI and cPCI including the mechanical, electrical and software features of these standards. These features allow PXI to successfully address test & measurement, data acquisition, industrial instrumentation and factory automation applications.

The PXI standard is defined and maintained by the PXI Systems Alliance (see <http://www.pxisa.org>). Manufacturers of PXI products are members of the alliance and sub-committees are assigned to manage different aspects of the specifications. Consequently, PXI users experience full interoperability between devices as all are designed to the same standards.

## PXI and PCI Express Overview

---

PXI Express builds on the successful PXI standard and the PCI Express architecture. Similar to the PXI standard, existing industry standards are leveraged by PXI Express to benefit from high component availability at lower costs. PXI Express also continues to maintain software compatibility with industry-standard personal computers, allowing customers to use the same software tools and environments with which they are familiar.

PXI Express leverages the electrical features defined by the widely-adopted PCI Express specification for data movement. This is accomplished by PXI Express Modules complying with the CompactPCI Express specification, which combines the PCI Express electrical specification with rugged euro card mechanical packaging and high-performance differential connectors. This allows measurement and automation systems based on PXI Express to have a data throughput of up to 8GBytes/sec in each direction. PXI Express also offers two-way interoperability with CompactPCI Express products.

Instrumentation capabilities within PXI Express can reach a new level of performance by providing point-to-point differential triggers, point-to-point differential variable clocks, and a 100 MHz differential System clock. PXI-1's bussed triggers, point-to-point triggers, and 10 MHz clock defined in the PXI specification are maintained. This allows PXI Express Module designers to make optimized cost versus performance tradeoffs when implementing instrumentation features. The PXI Express standard is defined and maintained by the PXI Systems Alliance (see <http://www.pxisa.org>).

## GX72xx PXI Express Features

---

The PXI Express (PXIe) and PCI Express (PCIe) features incorporated into the GX72xx chassis represent a significant performance improvement compared to PXI. Key features of PXI Express include:

- Dramatic improvement in available PCI bus bandwidth from 132 MB/s to 4 GB/s, offering a 30X improvement in bandwidth while maintaining software and hardware compatibility with PXI modules.
- Preservation of compatibility with existing PXI based systems.
- Additional timing and synchronization features including a 100 MHz PXI clock signal and differential Star triggers.
- Point to point data bus architecture eliminates data bus sharing, allowing the implementation of data streaming applications for continuous data acquisition applications.
- 8 3U hybrid slots offer the flexibility to use PXI Express or PXI modules in the same slot.
- 8 3U PXI or cPCI slots
- 4 3U PXIe or cPCI Express slots
- Support for x1, or x4 PCI Express lane configurations

## GX72xx Models

---

The GX72xx is available in several configurations for maximum flexibility:

- GX7200: This innovative chassis includes a hard disk drive. The GX7200 / GX7202 is designed to operate with the GX7944 embedded controller (additional controllers may be available). Internal circuits connect the embedded controller to the peripheral devices and many of the controller's interfaces (i.e. USB, Ethernet, VGA, etc.) are routed to the rear-panel of the GX7200 / GX7202, minimizing the number of connections required at the front of the chassis.  
**NOTE:** Your GX7900 controller is provided with documentation that describes its available connections and configuration separately.
- GX7210: This chassis is designed to operate with a PXIe bus expander such as a MXI Express interface. This configuration allows the use of a desktop PC or another PXI chassis as the system controller.
- GX7200R and GX7210R: Both models are available in desktop or rack-mount configurations. Rack mount configurations have model designations GX7200R and GX7210R.
- GX7202 and GX7212: GX7200R or GX7210R with an integrated cable tray and a hinged front panel with removable covers for creating custom interfaces or adapting to a mass interconnect receiver. Overall height of the GX7202 and GX7212 is 6U. Figure 2-1 shows a front view of the GX7202 chassis.

## Optional Equipment

---

Marvin Test Solutions offers a variety of products to use with your GX72xx chassis as follows:

- Embedded Controllers
- Remote Controllers
- 3U PXI instruments and switches
- Rack mount kits
- Blank panels

For part numbers, refer to Appendix B or call the office nearest you.

## Chassis Description – Front View (GX7200 & GX7202)

The GX7200 and GX7202 are modular 3U PXIe chassis.

Figure 2-2 shows the front view of the GX7200 and Figure 2-2.5 shows the front view of the GX7202 which includes a hinged front panel and integrated cable tray.

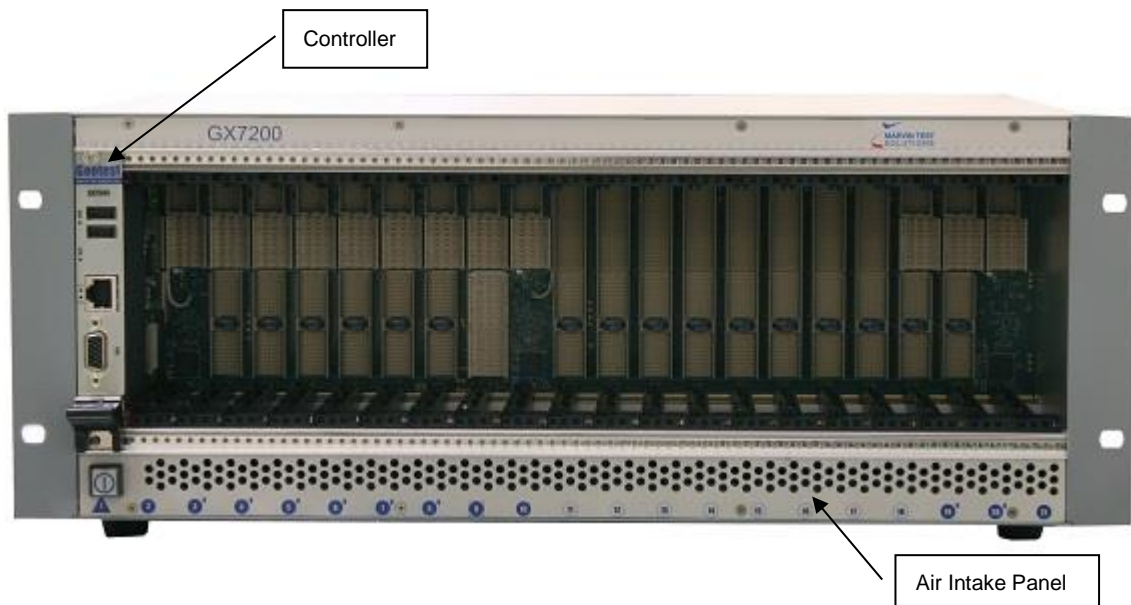


Figure 2-2: GX7200R with Controller Front View

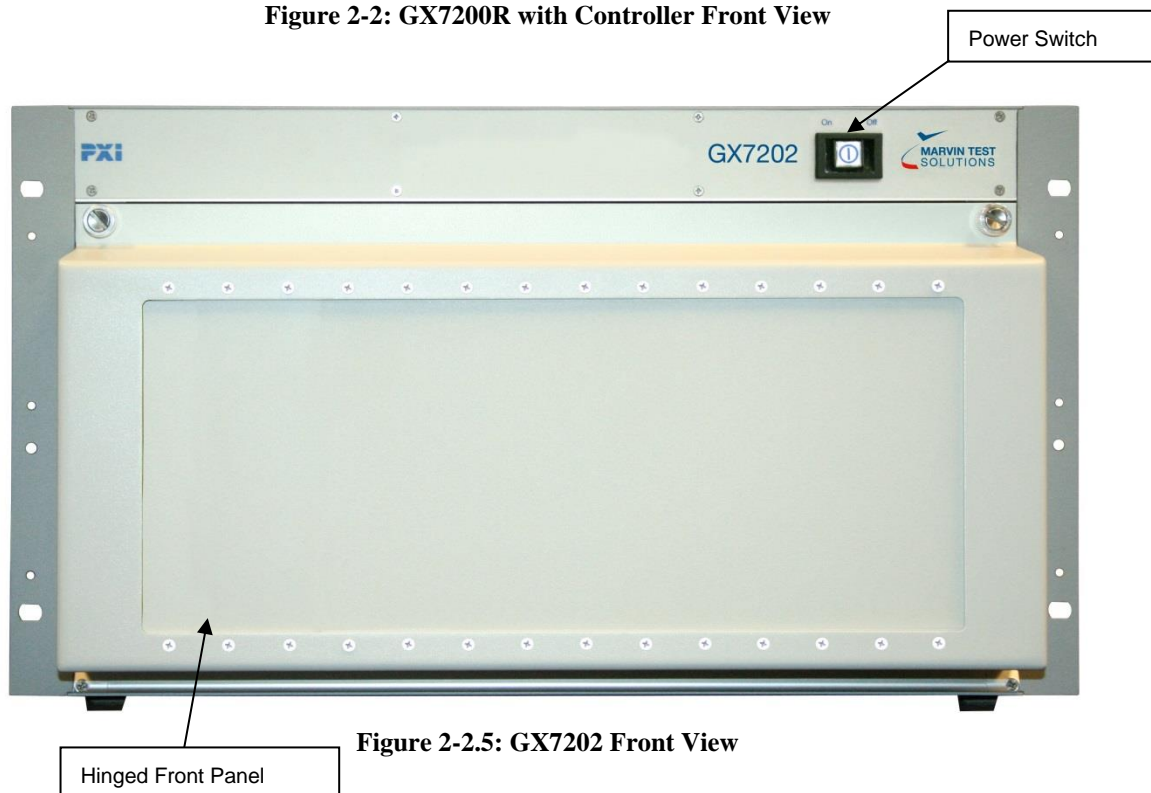


Figure 2-2.5: GX7202 Front View

### Power Switch

Lighted On/Off switch.

**System Slot (Slot #1)**

The System Slot is the leftmost slot and is used for embedded or remote controllers. The system slot can accept any embedded controller that is 1 slot wide although only the GX794x provides built-in connections to the built-in drives and rear panel I/O.






**Star Trigger Controller Slot (Slot #2)**

Either a Star Trigger controller or any PXI/cPCI instrument can use the Star Trigger Controller slot.

**Air Intake Panel**

This panel, located below the card cage, provides the intake for cooling the GX72xx. **DO NOT BLOCK THIS PANEL.** The use of an integral mass interconnect receiver with a GX7202 or GX7212 chassis will not compromise air flow within the chassis.

**Slot Number and Glyph Designations**

	Represents reserved PXIe controller slot (slot #1). The System slot is used for embedded or remote controllers. The GX7200 is designed to be compatible with the GX7944 controller which provides rear I/O connections to the built-in hard drive, DVD and rear panel I/O.
	Represents PXI slots, (slots #11, 12, 13, 14, 15, 16, 17 and 18), any PXI or cPCI board can be installed in these slots.
	Represents PXIe Hybrid slots (slots #3,4, 5, 6, 7, 19 and 20), any PXIe, cPCIe or PXI/cPCI hybrid compatible board can be installed in these slots. Hybrid compatible PXI boards are cards that J2 does not exist or XJ4 connector is used (instead of J2).
	Represents PXIe System Timing slot (#4). Either a System Timing Module or any PXIe/cPCIe instrument can use the System Timing slot.
	Represents PXIe (only) slot (slots #2, 10, and 21), any PXIe board can be installed in these slots.



## Chassis Description – Rear View

When used in conjunction with the GX794x embedded controllers, many of the controller's peripheral I/O are available through the rear panel. Error! Reference source not found. shows the rear view of the GX7200 and Figure 2-3.5 shows a rear view of the GX7202.

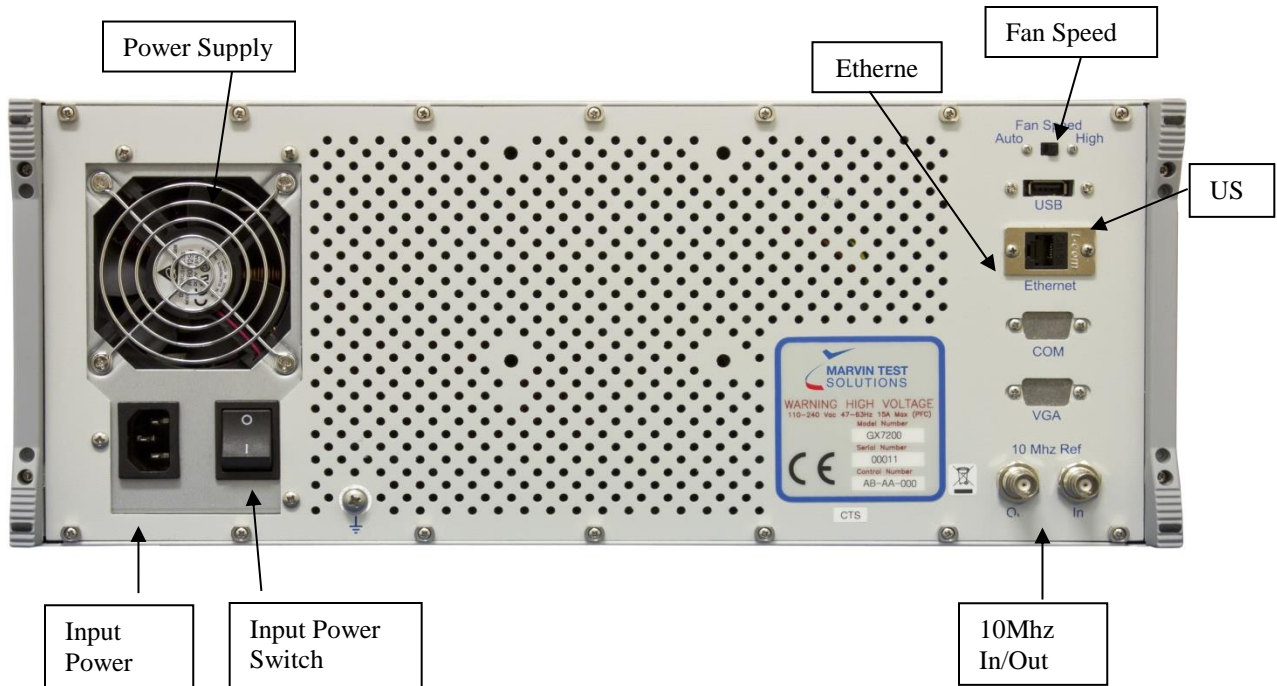


Figure 2-3: GX7200 Rear View

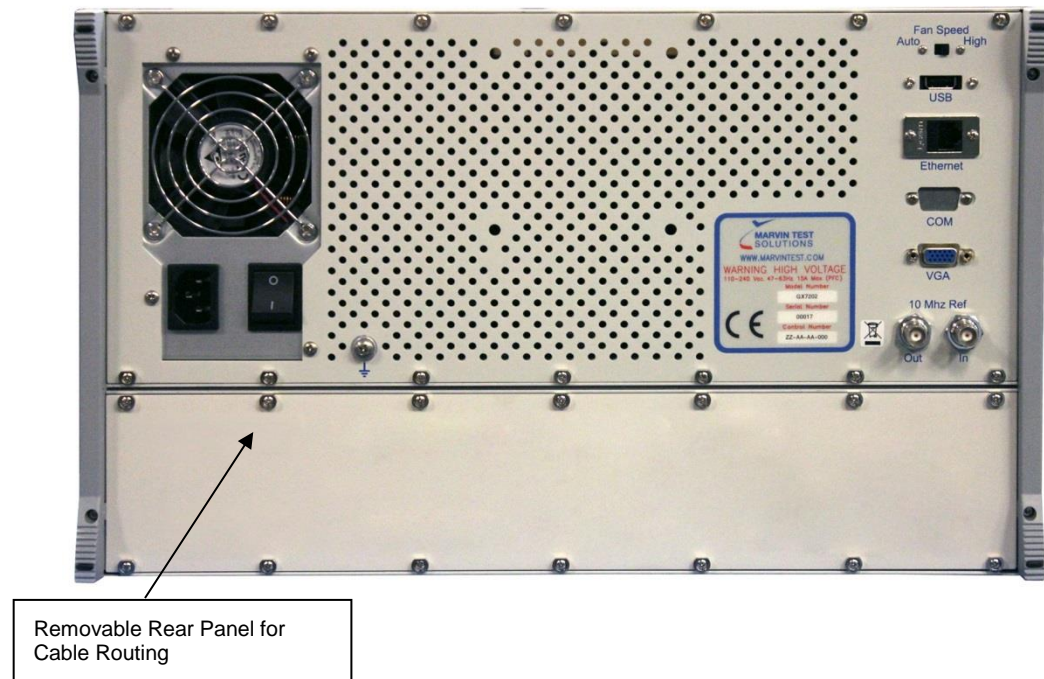


Figure 2-3.5: GX7202 Rear View

### **Auto / High Fan Speed Control**

The fan speed control allows the user to select the fan control to be automatic (controlled by the GxChassis software based on internal chassis temperature) or for high power dissipation applications, the switch can be set to high which will set the fans to operate at high speed continuously

### **PXI 10 MHz Input and Output Connections**

An external 10 MHz clock can be provided to the chassis via this connection. When present, the chassis will automatically select this input as the 10 MHz reference for the PXI backplane. The 10 MHz output connection provides a buffered 10 MHz PXI clock output.

### **Input Power Receptacle**

This receptacle connects to the power cord provided.

Several connections are available only on the GX7200 / GX7202 and only if used with the GX7944 Embedded Controller. The connections are marked on the rear I/O panel. These connections are:

### **USB Connectors**

1 USB ports

### **Ethernet Connector**

A 10/100 BaseT Ethernet port. By default, this port is enabled to the front panel. You must change the CMOS configuration in order to use this port from the rear

Note: The VGA rear connection is optional and only available with the GX7202.

## PXI/PXIe Slots

The GX72xx contains 21 3U slots numbered 1 to 21 as shown in Figure 2-4. Slot number 1 is dedicated for an embedded controller or for a bus extender such as the MXI Express. Slot 9 can be used by a PXI Star Trigger Controller or by a PXI/cPCI Express instrument. Slots 2 to 8 and 11 to 20 support the PXI Star Trigger.

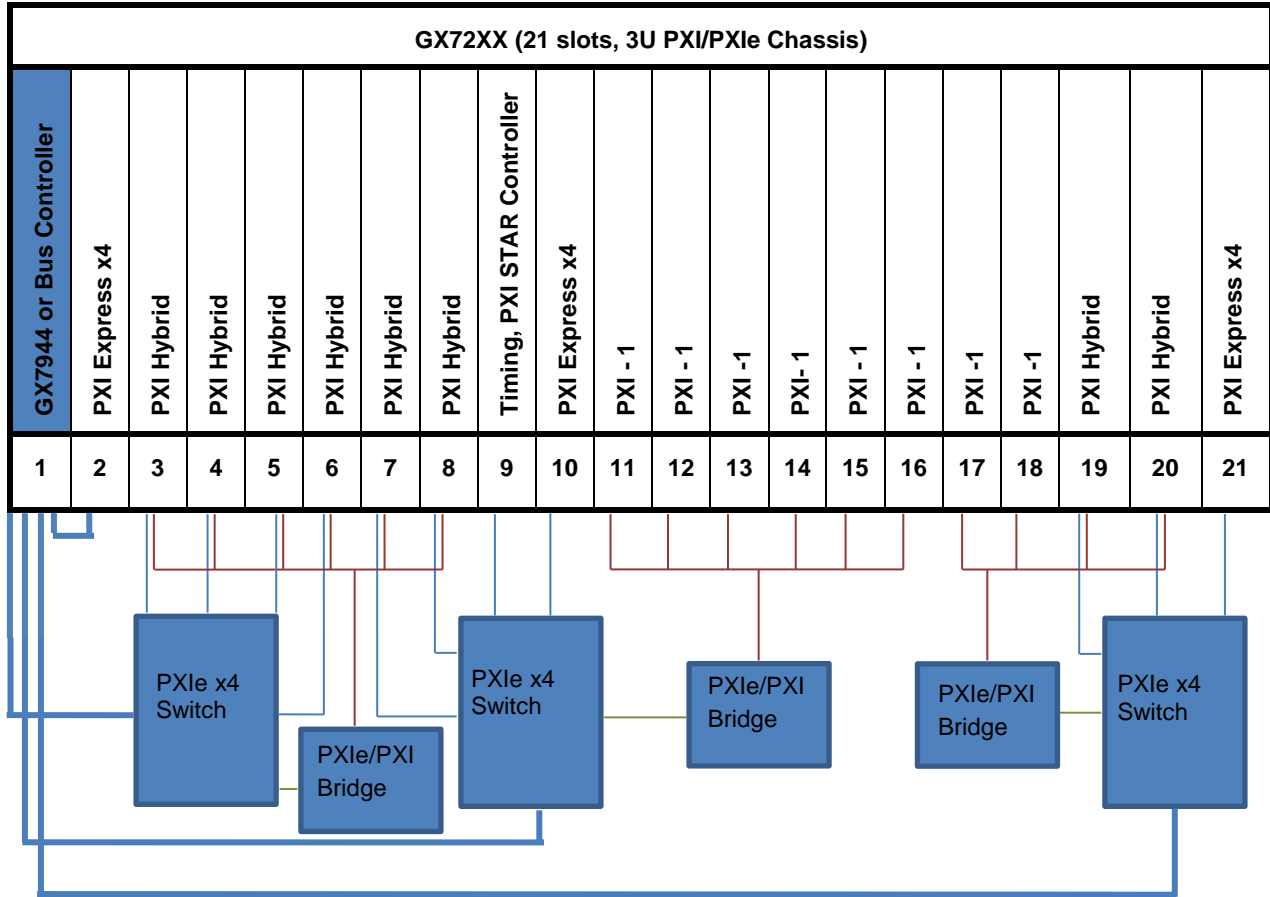


Figure 2-4: GX72xx Slot Configuration

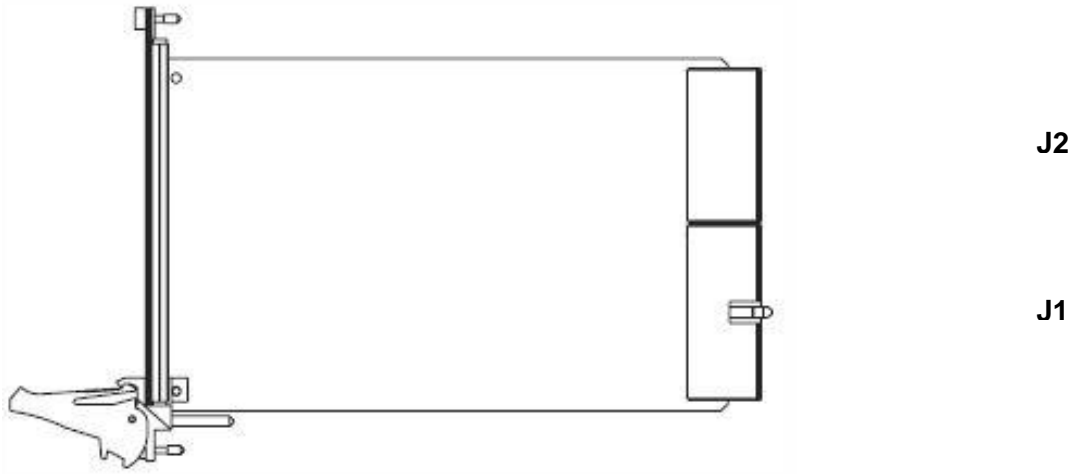
## Gx72xx Boards

---

The GX72xx supports several types of boards that can be inserted to the chassis. The following paragraphs describe the board types and characteristics.

### 3U PXI and Compact PCI Boards

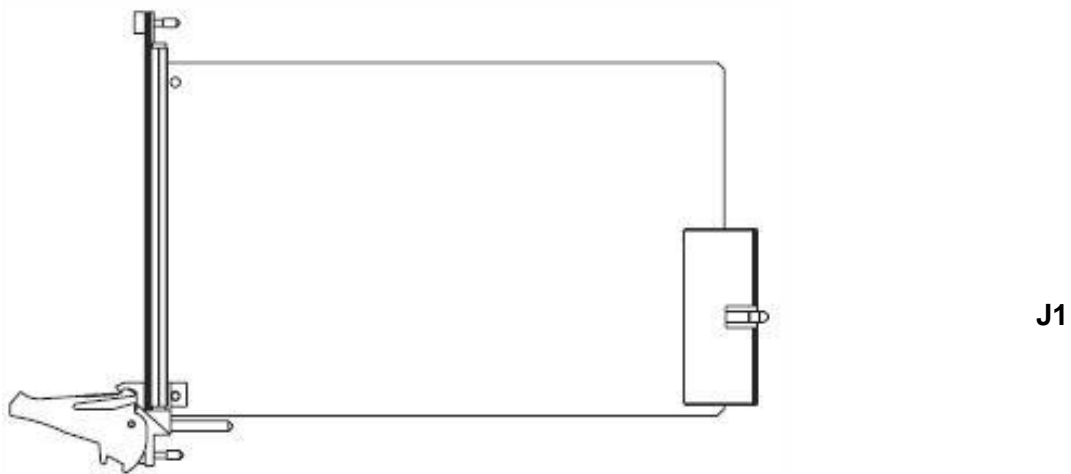
3U PXI-1 cards can be installed in slots 11, 12, 13, 14, 15, 16, 17 and 18 as shown in Figure 2-5.



**Figure 2-5: 3U PXI -1 Boards**

The 3U PXI-1 board has two rear connectors J1 and J2. J1 supports the PCI bus signals, and J2 supports the PXI bus signals. PXI signals include the local bus, star trigger signals and trigger bus signals.

If J2 does not exist in the board, the card is a Compact PCI board (cPCI) board as shown in Figure2-6. 3U Compact PCI boards can be plugged into PXI slots as well as 3U PXI Express Hybrid slots (slots 3 – 8 and 19 & 20 on the GX72xx chassis).



**Figure2-6: 3U Compact PCI Boards**

### 3U PXI Hybrid Boards

PXI hybrid boards consist of a J1 connector and XJ4 connector as shown in Figure 2-7. The J1 connector supports the PCI bus signals and the XJ4 connector supports the PXI trigger bus resources. Slots 3, 4, 5, 6, 7, 8, 19 and 20 on the GX72xx support 3U PXI Express Hybrid Boards and 3U Compact PCI boards as shown in Figure 2-7:

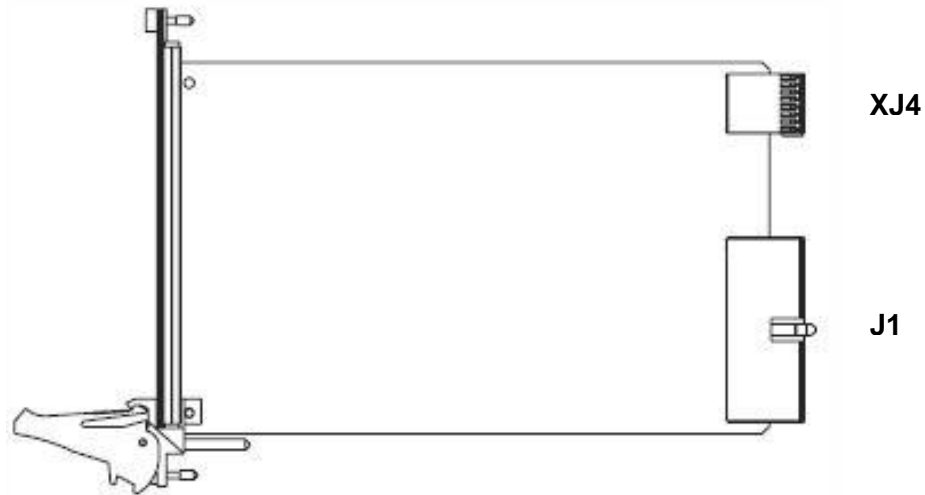


Figure 2-7: 3U PXI Hybrid Board

### 3U PXI Express Boards

3U PXI Express boards consist of an XJ4 and XJ3 connector as shown in Figure 2-8. The XJ4 connector supports power and the PXI trigger bus resources and the XJ3 connector supports the PCI Express bus connections and PXI Express trigger/clock resources. Slots 2, 9, 10, and 21 on the GX72xx support a PXI Express board.

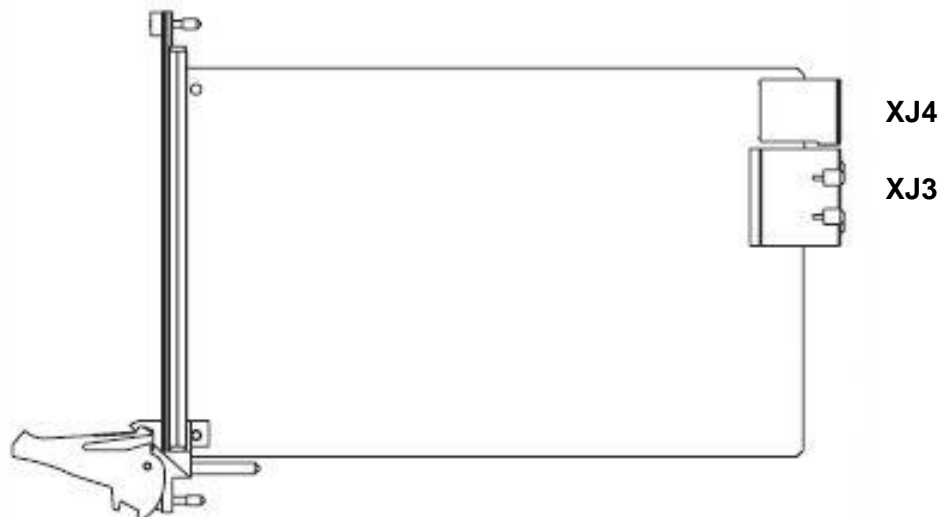
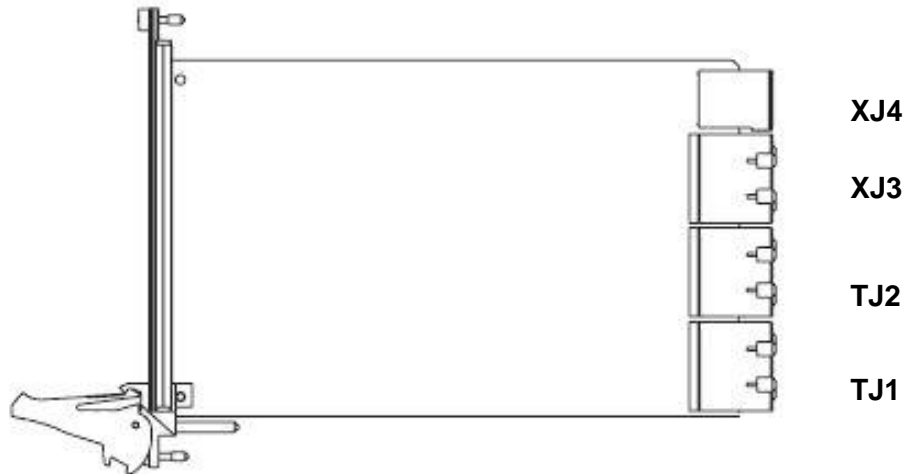


Figure 2-8: 3U PXI Express Board

### 3U PXI Express System Timing Boards


The 3U PXI Express System Timing board is shown in Figure 2-9, which is similar to the 3U PXI board and consists of XJ4 and XJ3 connectors which provide power, reference clocks and triggering signals. In addition, TJ2 and TJ1 connectors support PXI express three differential star triggers which are routed to each PXI express and PXI hybrid slot. Slot 9 supports a PXI Express system timing board.

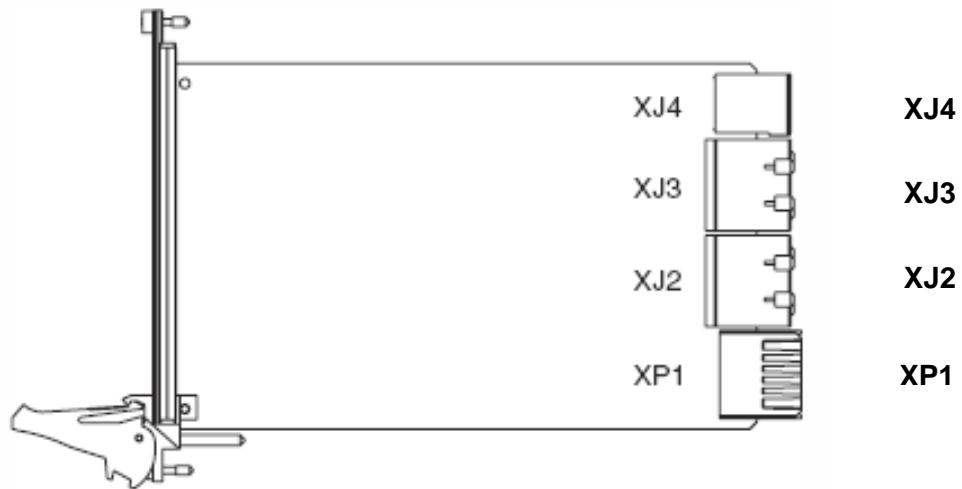


**Figure 2-9: 3U PXI Express Timing Board**

### 3U PXI Express System Board

The PXI Express System Board is shown in Figure 2-10. The GX72xx System Slot is slot 1. The GX7200 / GX7202 master chassis' PXI Express System Board is a CPU module that supports all of the controller's peripheral ports including VGA, Ethernet, and USB. The GX7210 / GX7212 slave chassis will accept a PXI Express System Board (bus expander) that can be connected to an external desktop PC or to another PXI Express chassis such as the GX7200.

	<p>Check with Marvin Test Solutions support before inserting a PXI Express system board/controller that is not supplied by Marvin Test Solutions specifically for the GX72xx. Third party PXI Express System Controllers boards may not be compatible with the GX7200 / GX7202 System Slot connections and may damage your chassis.</p>
---	---



**Figure 2-10: PXI Express System Board**

## PXI Bus Segments

---

The GX72xx's backplane employs (3) PCIe bus switches as well as (3) PCIe to PCI bridges to provide communication and control to all 21 slots. The backplane supports the Gen 2 4x4 PCIe architecture. Referring to Figure 2-4, slot 2 is connected to one x4 lane configuration. The remaining (3) x4 lanes employ the three PCIe switches to support the (9) PXIe slots and the (3) PCIe to PCI bridges. Peer to peer communication can be supported between slots that share a PCIe switch (see Figure 2-4).

## System Controller Slot

---

The System Controller slot is located in slot 1 of the chassis and occupies a single slot width. Slot numbers are clearly labeled below each slot where slot 1 is the left-most slot and slot 21 is the right most. The GX72XX can accept 3U embedded controllers that are 1-slot wide. The system controller slot supports (4) x1 or (4) x4 PCI Express lane configurations.

## PXI Express System Timing Slot

---

Slot 9 is the PXI Express System Timing slot (9<sup>th</sup> from the left). This slot has a dedicated star trigger line going to slots 2 through 8 and 11 through 20. Dedicated differential star trigger lines go to slots 2 through 8 and slots 10, 19, 20, and 21. The Star Trigger is used to synchronize between 8 instruments and it utilizes back plane traces that are of equal length, providing for a skew of less than 1nsec between slots. The differential star trigger lines are used to send high-speed clocks and triggers between the System Timing Module (Slot 9) and Express Peripheral Modules. If you do not need a System Timing Module, any PXIe or cPCIe instrument can be used in this slot.

## Local Bus

The PXI local bus is a daisy-chained bus connecting peripheral slots within the same bus segment. The local bus in segment A is single line which can interconnect slots 2 – 8. The local bus in segment B is 13 lines wide and interconnect slots 9 – 14. Each local bus can be used to pass analog or digital signals between two adjacent modules or to support a high-speed side-band digital communication path that is independent of the PXI or PXI Express bus bandwidth. Segment C interconnects the local bus for slots 15– 21. Local bus signals can support voltages from 0 to 42V DC and up to 200 mA DC of current.

Local Bus Routing																				
Bus Controller	PXI Express x4	PXI Hybrid	PXI Hybrid	PXI Hybrid	PXI Hybrid	PXI Hybrid	PXI Hybrid	PXI STAR Controller	PXI Express x4	PXI - 1	PXI - 1	PXI - 1	PXI - 1	PXI - 1	PXI - 1	PXI - 1	PXI - 1	PXI Hybrid	PXI Hybrid	PXI Express x4
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Local Bus Signals                      Symbol  
 1 (LOCAL BUS #6)                      →  
 13 (LOCAL BUS #0 to #12)            ↗

Figure 2-11 schematically shows the GX7200 / GX7210's local bus configuration.

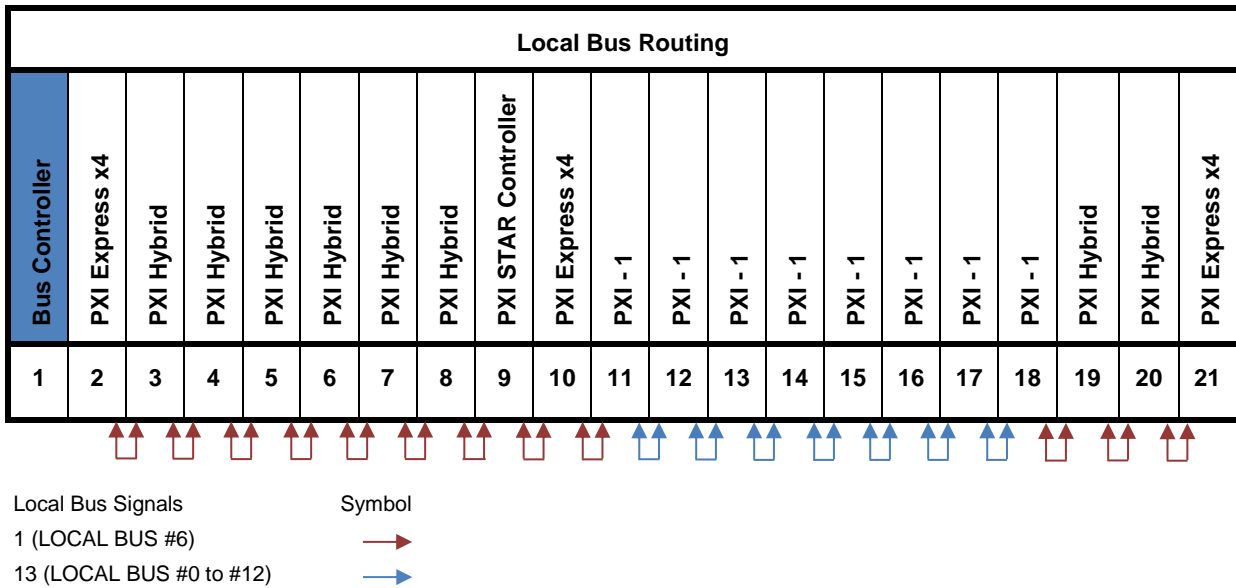


Figure 2-11: PXI Local Bus Routing



## Trigger Bus

The eight PXI bus trigger lines can be used for inter-module triggering or for sequencing measurement events. For example, triggers can be used to synchronize or sequence the operation of several different PXI peripheral modules. In other applications, one module can control precisely timed sequences for several modules within the system. Triggers may be passed from one module to another, allowing precisely timed responses to asynchronous external events that are being monitored or controlled. The number of triggers that a particular application requires varies with the complexity and number of events involved.

The PXI trigger bus typically provides connectivity only within a single bus segment preserving the high-performance characteristics of the trigger bus within a segment. The GX72xx backplane partitions instruments into logical groups as show in

Figure 2-12. A feature of the GX72xx is its ability to programmatically connect, set direction or isolate trigger lines between segments A, B, and C. This feature is controlled by the GxChassis driver.

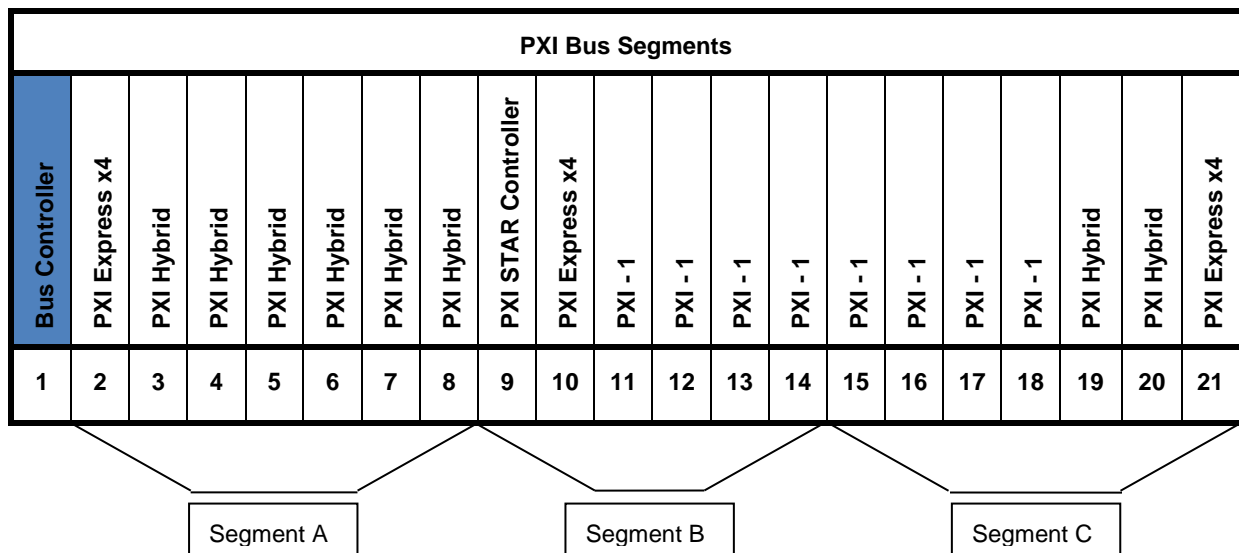


Figure 2-12: PXI Local Bus Segments

## Star Trigger Lines

---

17 PXI Star Trigger lines are connected to slots 2 to 8, and 11 to 20. They are sourced from the system timing slot (slot 9). The PXI Star Trigger lines can be used to synchronize the operation of several different PXI peripheral modules.

The PXI Star Trigger Bus offers high performance synchronization features to users of PXI systems. The Star Trigger Bus implements a dedicated trigger line between the System Timing Slot (Slot 9) and the other peripheral slots. A System Timing Module can be installed in this slot and can be used to provide very precise trigger signals to other peripheral modules. Systems that do not require this advanced trigger can install any standard express peripheral module in this slot. Through the required use of line-length equalization techniques for routing the star triggers, PXI systems can meet demanding triggering requirements for which bused triggers are not appropriate. Note that the star trigger can be used to communicate information back to the star trigger controller, as in the case of reporting a slot's status, as well as responding to information provided by the star trigger controller.

This trigger architecture for PXI gives two unique advantages in augmenting the bused trigger lines. The first advantage is a guarantee of a unique trigger line for each module in the system. For large systems, this eliminates the need to combine multiple module functions on a single trigger line or to artificially limit the number of trigger times available. The second advantage is the low-skew (1nsec) connection from a single trigger point. The PXI backplane defines specific layout requirements such that the star trigger lines provide matched propagation time from the star trigger slot to each module for very precise trigger relationships between each module.

For PXI Express slots, there are three pairs of PXIe Differential Star Trigger Lines. These are connected to slots 2 through 8, slot 10, and slots 19 through 21. The Differential Start Trigger (DSTAR) Lines can be used to pass clocks and triggers between the System Timing Slot (Slot #9) and the peripheral slots. The first DSTAR line, DSTARA, is a fast-switching LVPECL clock, generated by the System Timing Module, for precise timing. The second DSTAR line, DSTARB, is a fast-switching LVDS line carrying clock or trigger signals from the System Timing Module to the peripheral module. The third DSTAR line, DSTARC, is a fast-switching LVDS line carrying clock or trigger signals from the peripheral module to the System Timing Module. All lines are length matched.

## System Reference Clock

The PXIe 100MHz differential reference clock is distributed to all PXI Express, Hybrid Slots, and the System Timing Slot. A differential signal, PXIe\_SYNC100, is also distributed to all Hybrid, and PXI Express slots. The PXIe\_SYNC100 signal is derived from the PXIe 100 MHz differential reference clock and has the default behavior of pulsing at 10MHz, with a pulse width of approximately 5ns. The pulse rate can be slower than 10 MHz or non-periodic, allowing events to be synchronized between chassis that may be located far apart. The PXI 10 MHz system clock (PXI\_CLK10) is derived from the 100 MHz differential reference clock and is distributed to all slots of the GX7200. These common reference clocks can be used for synchronization of multiple instruments in a measurement or control system. The internal 10 MHz clock is also available on the GX72xx's rear panel via a BNC connector.

The GX72xx has the ability to synchronize to an external 10MHz source providing a more stable/accurate system clock. The external 10MHz clock can be provided through the System Timing Slot or through the BNC connector on the rear panel of the GX72xx.

## System Power Supply – GX7200 / GX7210 / GX7202 / GX7212

One power supply provides system power to all slots of the GX72xx. A total power of 755 watts is available.

### Power Distribution

The GX72xx meets or exceeds the requirements of the PXI specifications regarding the power provided to each slot. The table below lists current per slot as required by the PXI specifications:

Slot \ Voltage	5V	3.3V	+12V	-12V	5Vaux
PXIe System Slot, no expansion slots	1A	3A	2A	N/A	1A
PXI Instrument Slot	2A	2A	0.5A	0.25A	N/A
PXI Hybrid Slot	2A	3A	2A	0.25A	0A
PXIe Slot/System Timing Slot	N/A	3A	2A	N/A	0A
Minimum required current for a GX72XX configuration	33A	53A	30A	4A	1.5A*

\*There must be 0.5A of the 5Vaux current available to the peripheral slots.

**Table 2-1: Power per Slot per the PXI Specification**

The maximum current provided by the GX72xx for any PXI slot is listed in Table 2-2.

Slot \ Voltage	5V	3.3V	+12V	-12V	5Vaux
PXIe System Slot	15A	15A	30A	0A	1A
PXI Instrument Slot	6A	6A	1A	1A	0A
PXI Hybrid Slot	6A	6A	4A	1A	1A
PXIe Slot/System Timing Slot	0A	6A	4A	0A	1A
Total available current for the GX72XX	60A	60A	30A	5A	2A

Total power cannot exceed 755 watts.

**Table 2-2: Available GX7200 / GX7210 / GX7202 / GX7212 Power**

## Overview of the GxChassis Software

---

Once the GxChassis 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).

- **GxChassis 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.
- **GxChassis driver** - a DLL (GxChassis.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.

## GxChassis driver Features

---

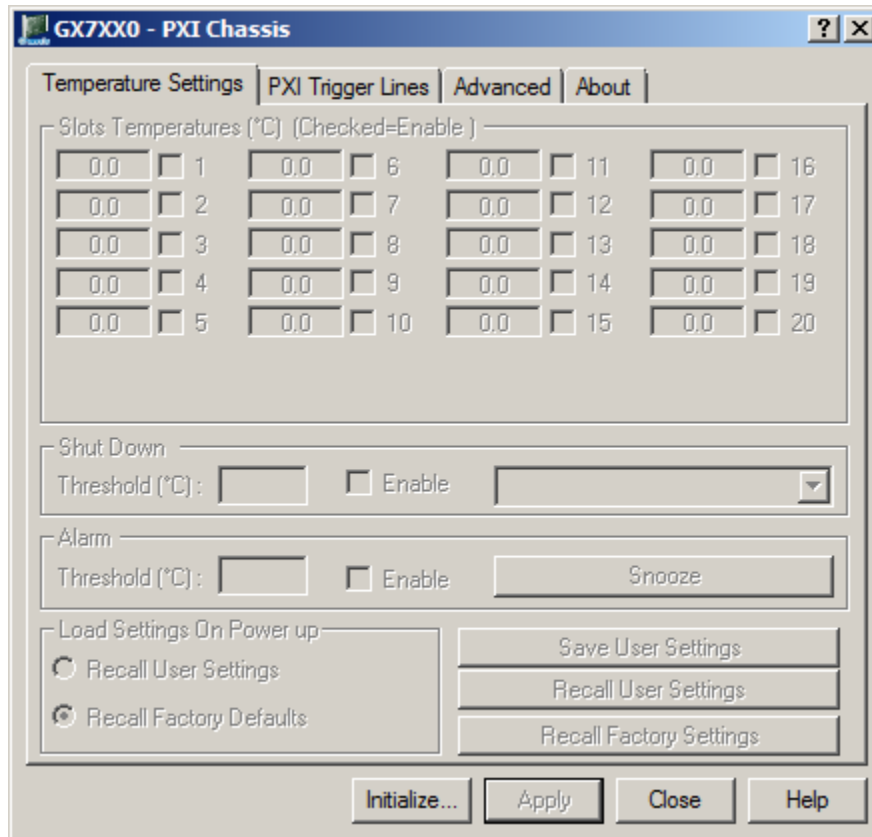
The GxChassis driver has the following features:

- Program the PXI chassis' over-temperature shutdown level.
- Program the PXI chassis' over-temperature alarm level.
- Measure all PXI chassis slot temperatures.
- Enable/disable each of the PXI chassis slots' temperature measurements.
- Measure all PXI chassis backplane power supply voltages (+3.3V, +5V, +12V, -12V).
- Measure the PXI chassis backplane voltage level supplied to the VIO pins.
- Program each PXI trigger lines' direction.
- Enable/disable each of the eight PXI trigger lines.
- Selectable temperature scale.
- Monitor / control system fans
- Save settings to an on-board EEPROM to be used as defaults.
- Complete API calls controlling all of the PXI chassis' capabilities.
- Front panel control of all of the PXI chassis' capabilities.

## Virtual Panel Description

The GxChassis driver includes a virtual panel program, which allows 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 chassis.

To open the virtual panel application, select **GxChassis Panel** from the **Marvin Test Solutions, GxChassis** menu under the **Start** menu. The GxChassis virtual panel opens as shown in **Figure 2-13**:

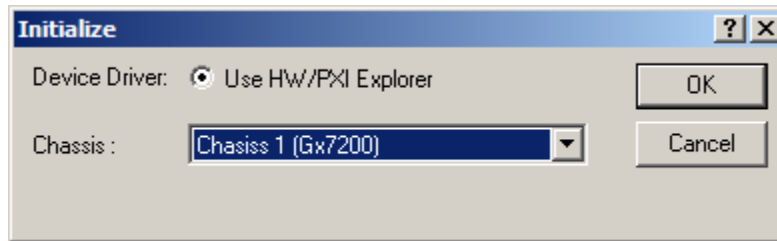


**Figure 2-13: GxChassis Virtual Panel – Temperature Settings (not Initialized)**

### Virtual Panel Initialize Dialog

The Initialize Dialog initializes the chassis while the settings of the chosen chassis **will not change**. The panel will reflect the current settings of the board after the Initialize dialog closes.

The Marvin Test Solutions' **Chassis** number and the model in the Initialize dialog box refer to the PXI **Chassis** number in which it was set. Select the chassis from the drop down list. The list displays all the Marvin Test Solutions' chassis that the PXI Explorer found. The chassis number can also be reviewed or set by using the **PXI/PCI Explorer** applet located in the Windows Control Panel. Select the chassis number and click **OK** to initialize the driver for the specified chassis.



**Figure 2-14: Initialize Dialog Box using Marvin Test Solutions' HW driver**

The GxChassis driver includes a virtual panel program, which allows 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 chassis.

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

## Virtual Panel Temperature Settings

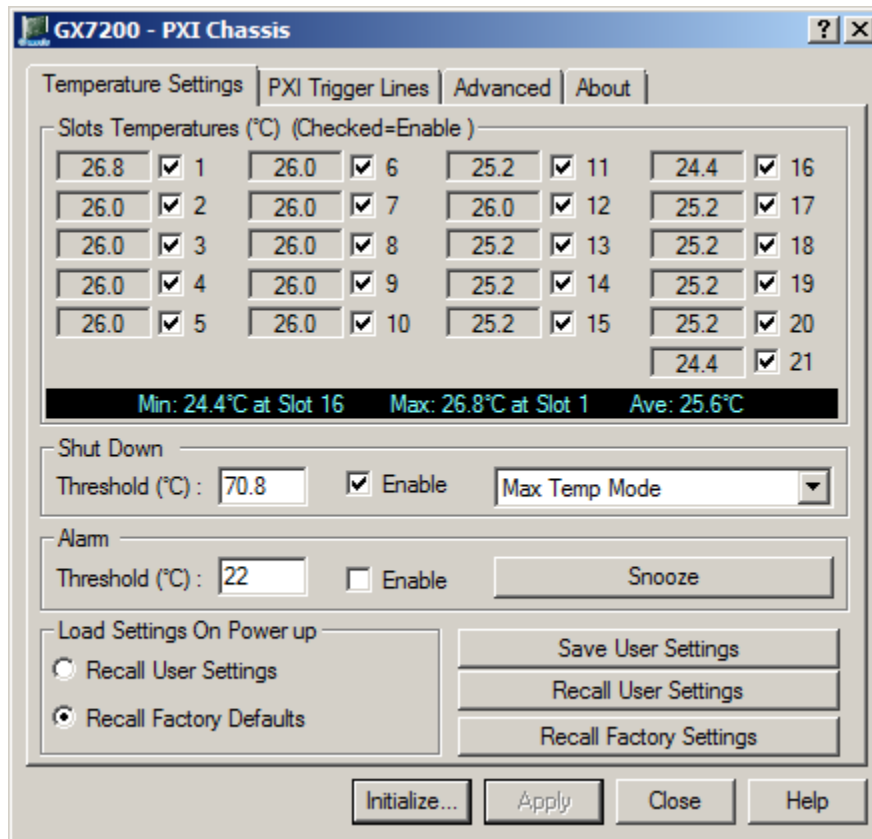


Figure 2-15: GxChassis Virtual Panel – Temperature Settings

The following controls are shown in the Temperature Settings page:

### Slots Temperatures (Group Box)

Displays measurement of all active slots' temperatures and sets/displays slots' active states. Only active (enabled) slots determine if the alarm or shutdown thresholds' conditions were met.

### Shut Down (Group Box)

**Threshold (Edit box):** Sets/displays the shutdown Temperature to any value between 20°C and +70°C. The programmed threshold can be saved to the onboard EEPROM and be automatically loaded on the next system power up.

**Enable (Button):** If checked, the shutdown Temperature is enabled.

**Mode (Combo dropdown list):** Sets/displays the temperature threshold operational mode. The temperature threshold operational mode dictates how the alarm and shutdown thresholds will be activated. When set to Max Temp Mode, the shutdown and alarm temperature will be activated when any of the active slots temperature is above the threshold. When set to Average Temp Mode, the alarm will be activated when the average of all active slots' temperatures are above the alarm threshold.

### Alarm (Group Box)

**Threshold (Edit box):** Sets the Alarm state. When the Alarm is on (threshold condition was met or set to On) both backplane buzzers will beep simultaneously in intervals of 10 seconds.

**Enable (Button):** Check enables the threshold temperature at which point, the alarm will turn on..

**Snooze (Button):** Snooze the Alarm when it is on. If the alarm condition reoccurs, the alarm will reactivate.

**On Power up (Group Box)**

Sets/displays the source settings to be loaded or saved.

**Save User Settings (Button):** Saves all current settings to the onboard EEPROM as well as which settings will be loaded on the next power up as was specified in the On Power up (Group Box).

**Recall User Settings (Button):** Loads and applies the last saved user's settings from the onboard EEPROM.

**Recall Factory Settings (Button):** Loads and applies the factory default settings.

**Apply:** Applies current settings.

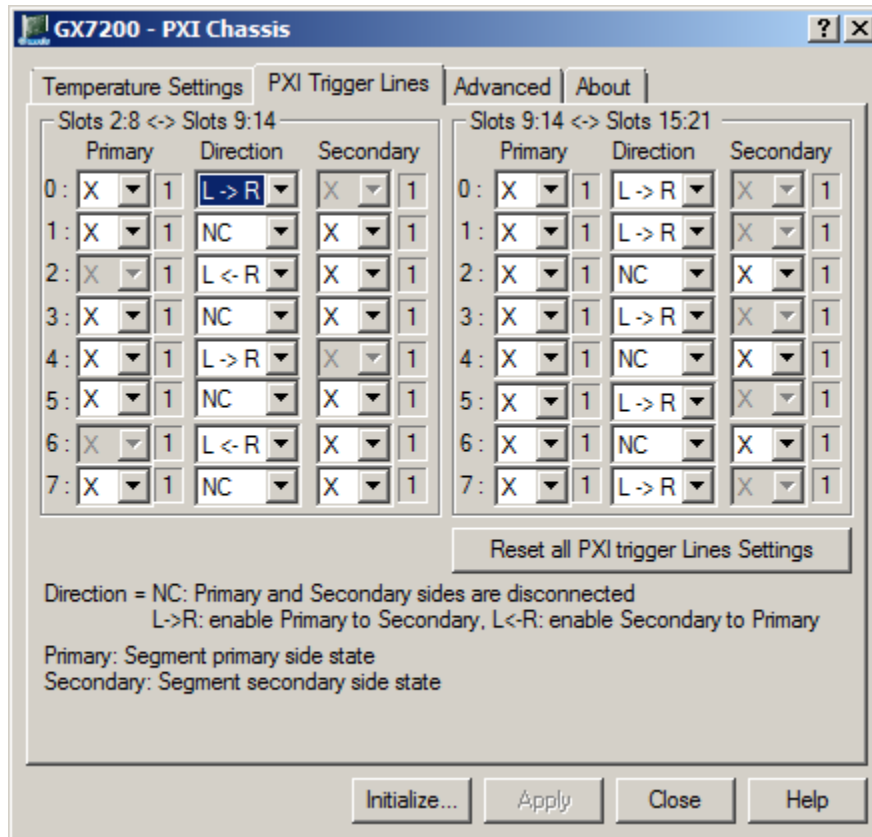
**Close:** Closes (exits) the GxChassis panel.

**Help:** Opens the GxChassis on-line help window.



## Virtual Panel PXI Trigger Lines

Clicking on the PXI Trigger Line tab will show the **PXI Trigger Line** page as shown in Figure 2-16:



**Figure 2-16: GxChassis Virtual Panel – Pxi Trigger Lines**

The following controls are shown in the PXI Trigger Lines page:

### Slots 2:8 <-> Slots 9:14 (Group Box)

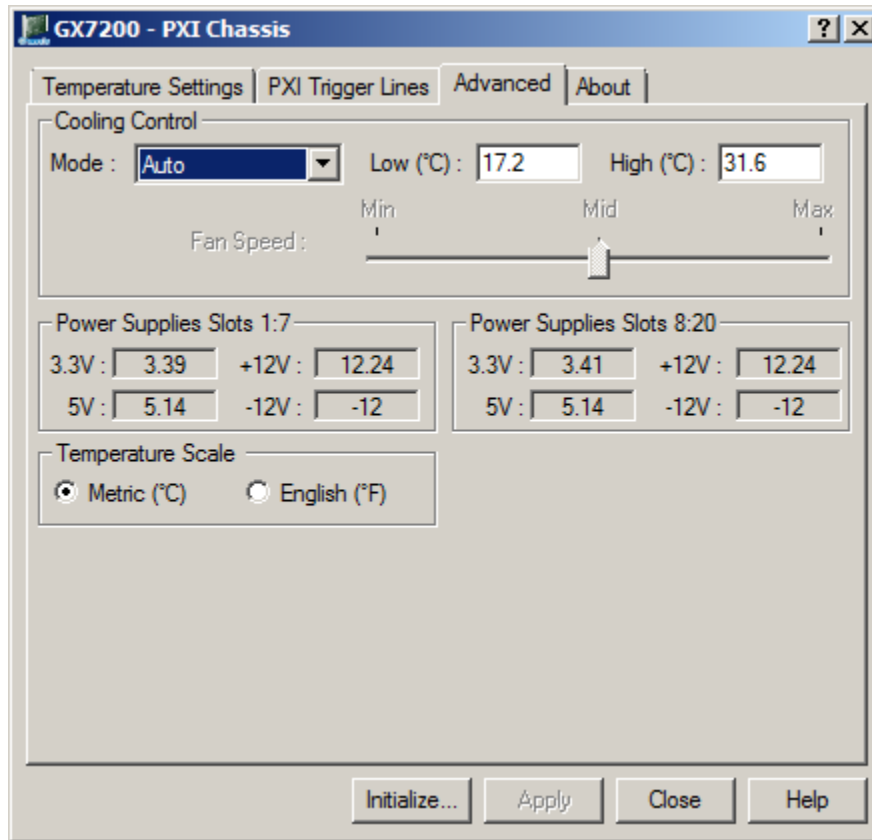
Sets/Displays PXI trigger line states and directions between slots 2:8 and slots 9:14.

### Slots 9:14 <-> Slots 15:21 (Group Box)

Sets/Displays PXI trigger line states and directions between slots 9:14 and slots 15:21.

## Virtual Panel Advanced page

Clicking on the **Advanced** tab will show the **Advanced page** as shown in Figure 2-17:



**Figure 2-17: GxChassis Virtual Panel – Advanced page**

The following controls are shown in the Advanced page:

### **Power Supplies Slots 1:10 (Group Box)**

Displays the measured +3.3V, +5V, +12V and -12V backplane voltages for slots 1 through 10.

### **Power Supplies Slots 11:21 (Group Box)**

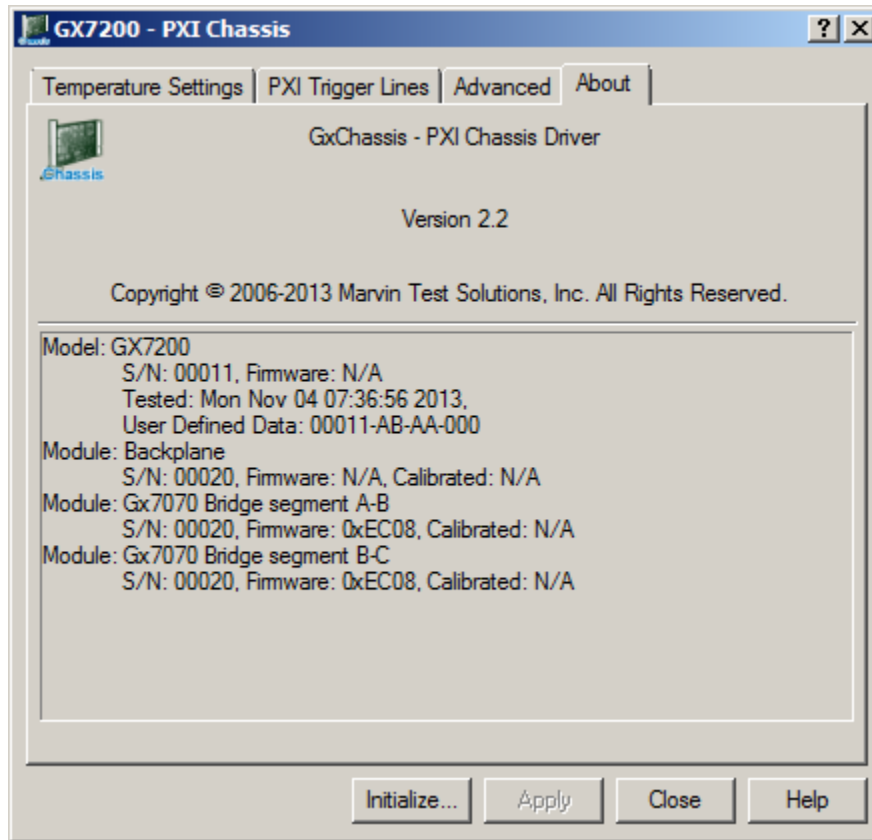
Displays the measured +3.3V, +5V, +12V and -12V backplane voltages for slots 11 through 20.

### **Temperature Scale (Group Box)**

Sets/Displays the temperature scale to Metric or English used for setting or getting any temperature value. Once the temperature scale is set, the same scale will be applied to all temperature values, e.g. shutdown temperature. The temperature scale setting is saved to the host computer.

## Virtual Panel About Page

Clicking on the **About** tab will show the **About** page as shown in Figure 2-18:



**Figure 2-18: GxChassis Virtual Panel – About Page**

The top part of the **About** page displays version and copyright of the GxChassis driver. The bottom part displays the board summary, including the EEPROM version; board Revision, FPGA version, board serial number and the calibration time.



## Chapter 3 - Setup and Installation

This chapter describes how to set up the GX72xx chassis and boards.

### Unpacking and Inspecting the Chassis

---

1. Before unpacking the GX72xx, check the outside of the shipping package for damage. Note any damage on the shipping bill.
2. Remove the chassis from the shipping carton.
3. Read the packing list to ensure all listed items are enclosed, including hardware, power cords, manuals, etc.
4. Inspect the unit. If any missing items, defects, or damage are noticed, notify Marvin Test Solutions immediately.

### Mounting Information

---

The GX72xx is designed to operate on a bench or within an instrument rack system. Follow the appropriate installation instructions for your GX72xx.

Openings in the rear and along the bottom-front panel of the chassis facilitate power supply and instrument cooling. This is very important to the operation of your GX72xx. Make sure to place your GX72xx on a bench top or in an instrument rack so the air intake openings in the front and the air outlet openings along the rear panel are not blocked. Keep other equipment a minimum of 3 inches away from the air intake and outlets.

Rack-mount applications require the optional rack-mount kit available from Marvin Test Solutions. Refer to the rack-mount kit documentation to install your GX72xx in an instrument rack.

### Line Voltage Selection

---

The line voltage input selection of the GX72xx is automatic. The GX72xx chassis can operate with line voltages from 90 to 264 VAC, 47 to 63 Hz. Maximum input current for the GX72xx / GX7210 is 12 A (PFC).

### Chassis Installation

---

Follow these steps to install the GX72xx chassis:

1. Place the GX72xx chassis on a sturdy, level surface. Leave space behind the chassis for ventilation.
2. Turn off the power switches.
3. Connect the power cable to the chassis and an outlet.
4. Install an embedded controller (master configuration, GX72xx) or a remote controller (slave configuration, GX7210) to slot #1 if not installed.
5. For a master chassis, connect a keyboard, mouse and VGA monitor to the controller using the controller's front panel connections or the peripheral connections located on the rear of the chassis.
6. Turn on the chassis power and the optional external system (for slave installation turn on the slave first)
7. Install the **GxChassis** software.
8. Configure your system using the **PXI/PCI Explorer** applet.
9. Install any additional drivers for PXI instruments.
10. Turn off the system.
11. Install PXI modules into the chassis as described in the next procedure.

12. Turn on the chassis power switch and follow the Found New Hardware Wizard instructions for new instruments installed.

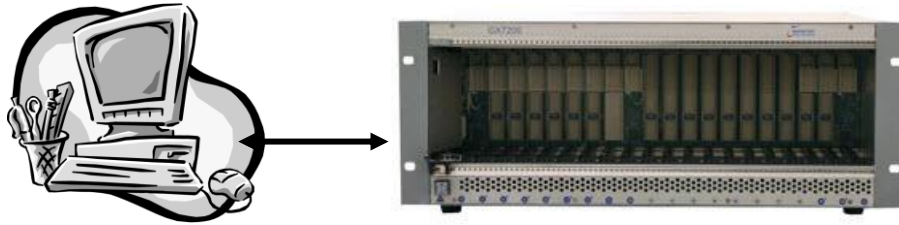
## **GX72xx Master and Slave Configurations**

---

The chassis is provided with two configurations:

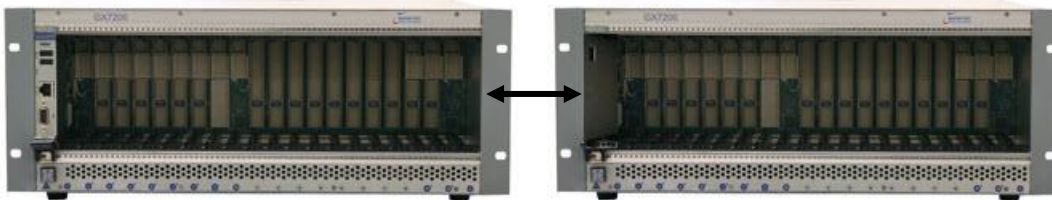
1. Master configuration: slot 1 in this case contains embedded controller such as the GX7944. The embedded controller contains a single slot computer with CPU, Memory, USB and Ethernet ports, and hard drive.
2. Slave configuration: slot 1 contains PXIe expander card (such as a MXI express interface) connected to an external PC or another PXI chassis such as a GX72xx. The controller resides on the external PC/chassis. Multiple slave chassis can be connected to the controller in a start or daisy chained configuration.

Figure 3-1 outlines a remote configuration with a desktop PC being the system controller.



**Figure 3-1: GX7210 Slave Configuration with a Desktop PC**

Figure 3-2 outlines a master-slave configuration with another PXI chassis being the system controller.



**Figure 3-2: GX7200 (master) Connected to GX7210 (slave)**

## Installation of the GxChassis Software

---

Before installing any board in the chassis, it is recommended to install the GxChassis software as described in this section. The software is installed on the chassis controller (for GX7200) or on the external PC or the chassis where the controller resides (such as a GX7210 slave configuration). To install the GxChassis driver follow the instruction described here:

1. Insert the Marvin Test Solutions CD-ROM and locate the **GxChassis.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 (GxChassis.EXE). 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](http://www.marvintest.com)).
2. Run the GxChassis setup and follow the instruction on the Setup screen to install the GxChassis driver.

---

**Note:** When installing under Windows 2000/XP/VISTA /Windows 7/ Windows 8, 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 GxChassis driver to access resources on your chassis.

---

3. The first setup screen to appear is the Welcome screen. Click **Next** to continue.
4. Enter the folder where GxChassis 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\GxChassis for 32-bit Windows and C:\Program Files (x86)\Marvin Test Solutions\GxChassis 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 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, GxChassis** menu located in the Windows Taskbar.

## 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 **GxXXXInitialize** and **GxXXXInitializeVisa** 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).

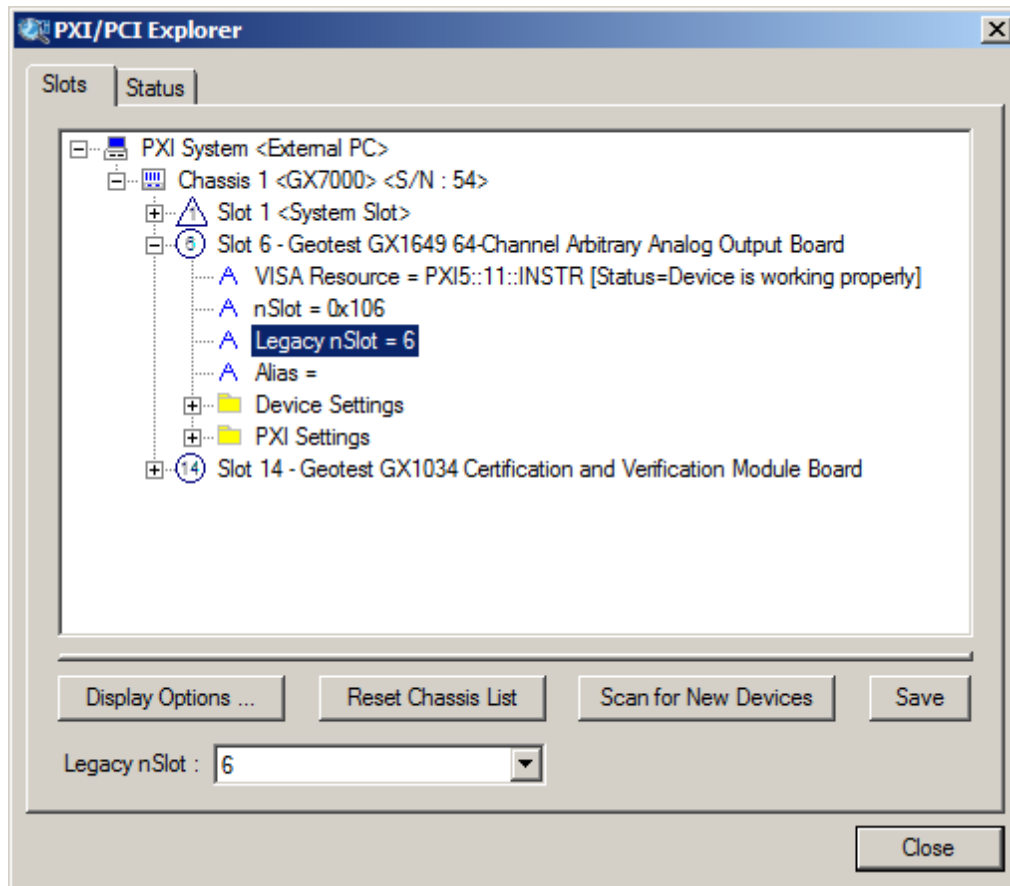


Figure 3-3: PXI/PCI Explorer



## Installing PXI Instruments

---

Install a PXI Instrument board (PXI module) as follows:

1. Turn off the PXI chassis and unplug the power cord.
2. Set the board switches and jumpers if required.

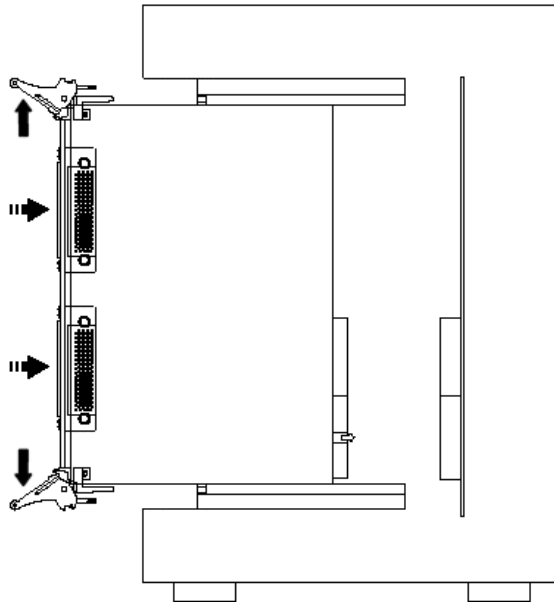


**Caution** - Electrostatic discharge can damage components on the GX72xx and PXI modules.

---

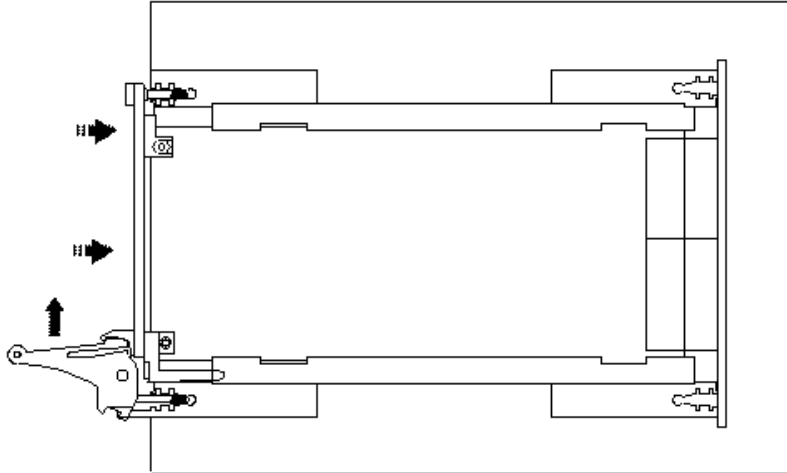
Check the board documentation for details on jumpers and switch settings before the installation.

3. Locate an empty PXI Slot on the chassis. Ensure that the module is compatible with the PXI slot – i.e. PXI Express, PXI Hybrid, or PXI-1.
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 3-4).



**Figure 3-4: Ejector handles position during module insertion**

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



**Figure 3-5: Ejector handles position after module insertion**

7. Tighten the board's front panel screws 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' power switch.

## PXI Instrument Removal

---

Remove a PXI instrument board as follows:

1. Shut down your system from the Windows Start menu.
2. Turn off the PXI chassis (master and slaves) and unplug the power cord.
3. Disconnect and remove any cables/connectors connected to the board.
4. Unscrew the module's front panel screws from the chassis.
5. Push **outward** the ejector handles and pull the PXI board away from the chassis.

## Using External Instruments

---

Your GX72xx chassis supports all PXI and cPCI instruments.. In some cases, however, you may need to connect additional instruments to the GX72xx. These additional instruments are typically GPIB (IEEE-488). To use an external GPIB instrument, you will need a Plug-in PXI module that provides an interface to GPIB. Such interfaces are available from numerous vendors.

## Installation Directories

---

The GxChassis driver files are installed in the default directory C:\Program Files [(x86)]\Marvin Test Solutions\GxChassis. You can change the default GxChassis directory to one of your choosing at the time of installation.

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

Name	Purpose / Contents
...\Marvin Test Solutions\GxChassis	The GxChassis directory. Contains panel programs, programming libraries, interface files and examples, on-line help files and other documentation.
...\Marvin Test Solutions\HW	HW device driver. Provides 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. GxChassis Driver and examples are copied to this directory only if <i>ATEasy</i> is installed on your machine.
Windows System folder (...\Windows\System32 or ...\Windows\SysWOW64)	Windows System directory. Contains the GxChassis DLL driver and some upgraded system components, such as the HTML help viewer, etc.

## Driver Files Description

---

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

### Driver File and Virtual Panel

GxChassis.dll - 32-Bit MS-Windows DLL for applications running under Windows, 95, 98, Me, NT, 2000, XP, Windows 7, 8 or above.

- GxChassispanel.exe – An instrument front panel program for all GxChassis supported boards.

### Interface Files

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

- GxChassis.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++
- GxChassis.LIB - Import library for GxChassis.dll (used when linking C/C++ application that uses GxChassis.dll).
- GxChassisBC.LIB - Import library for GxChassis.dll (used when linking Borland C/C++ application that uses GxChassis.dll).
- GxChassis.pas - interface file to support Borland Pascal or Borland Delphi.
- GxChassis.bas - Supports Microsoft Visual Basic 4.0, 5.0 and 6.0.
- GxChassis.vb - Supports Microsoft Visual Basic .NET.
- GxChassis.driv - ATEasy driver File for GxChassis Virtual Panel Program

## On-line Help and Manual

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

GxChassis.pdf – On line, printable version of the GxChassis 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 GxChassis manuals. You can view and/or print this file using the Windows NOTEPAD.EXE or any other text file editors.

## Example Programs

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

### Microsoft Visual C++ .NET example files:

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

### Microsoft Visual C++ 6.0 example files:

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

### Borland C++ example files:

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

**Microsoft Visual Basic .NET example files:**

- GxChassisExampleVB.vb - Example form.
- GxChassisExampleVB.resx - Example form resource.
- GxChassisExampleVBapp.config - Example application configuration file.
- GxChassisExampleVBAssemblyInfo.vb - Example application assembly file
- GxChassisExampleVB.vbproj - Project file
- GxChassisExampleVB.exe - Example executable

**Microsoft Visual Basic 6.0 example files:**

- GxChassisExampleVB6.frm - Example form
- GxChassisExampleVB6.frx - Example form binary file
- GxChassisExampleVB6.vbp - Project file
- GxChassisExampleVB6.exe - Example executable.

**ATEasy driver and examples files (ATEasy Drivers directory):**

- GxChassis.drv - driver
- GxChassis.prj - example project
- GxChassis.sys - example system
- GxChassis.prg - example program

**LabView Driver**

- GxChassis.llb – LabView library

**Setup Maintenance Program**

---

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

**Modify.** Use when you want to add or remove GxChassis components.

**Repair.** Use to reinstall.

**Remove.** Use when you want to completely remove GxChassis.

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

Follow the instruction on the screen until Setup is complete.



# Chapter 4 - Programming the Chassis

## Overview

---

This chapter contains information about how to program the Chassis' functions using the GxChassis driver. The GxChassis driver contains functions to initialize, control and retrieve information and settings from the Chassis. 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 GxChassis driver supports many development tools. Using these tools with the driver is described in this chapter. In addition, the GxChassis 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 GxChassis driver types.

## The GxChassis Driver

---

The GxChassis driver is a Windows DLL file: GxChassis.dll. 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 GxChassis driver with C/C++ development tools:

- Include the GxChassis.h header file in the C/C++ source file that uses the GxChassis 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 an import library GxChassis.lib for Microsoft Visual C++ and GxChassisBC.lib or 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 GxChassis 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 6.0 the user must include the GxChassis.bas to the project. For Visual Basic .NET use the GxChassis.vb.

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

## Programming GxChassis Boards Using ATEasy®

---

The GxChassis package is supplied with an ATEasy driver. The ATEasy driver uses the GxChassis.dll to program the chassis' functions. . The ATEasy driver includes an example that contains a program and a system file for use with the ATEasy driver. Plain language 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 commands 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. CHASSIS1, CHASSIS2.
- 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.



## Using the GxChassis Driver Functions

---

The GxChassis driver contains a set of functions that support all of the chassis' Smart features. The **GxChassisInitialize** function returns a handle that must be used with other driver functions to program the chassis. This handle is usually saved in the program as a global variable for later use when calling other functions. The initialize function does not change the state of the chassis. .

### Chassis Handle

The chassis handle argument *nHandle* passed (by reference) to the parameter *pnHandle* of the **GxChassisInitialize** is a short integer (16-bit) number. It is used by the GxChassis driver functions to identify the chassis being accessed by the application. Since the driver can support multiple chassis at the same time, the *nHandle* argument is required to identify which chassis is being programmed.

The *nHandle* is created when the application calls the **GxChassisInitialize** function. There is no need to destroy the handle. Once the driver is initialized the handle can be used with other function calls to program the chassis.

### Error Handling

All the **GxChassis** functions return a status named *pnStatus* as the last parameter. This parameter can be later used for error handling. The status is zero for success, less than zero for failure or error. When the status is error, the program can call the **GxChassisGetErrorString** function to return a string representing the error. The **GxChassisGetErrorString** reference contains possible error numbers and their associated error strings.

### Driver Version

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

### Panel

Calling the **GxChassisPanel** will display the instrument virtual front panel window. The panel can be used to display its current setting and to control the board interactively. The panel function may be used by the application to allow the user to directly interact with the board.

The **GxChassisPanel** function is also used by the GxChassis.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 (GxChassis.dll and the HW device driver files located in the HW folder) can be shipped with the application. Typically, the GxChassis.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 GxChassis driver files. Alternatively, you can provide the GxChassis 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 get or set a group or channel voltage.

To run enter the following command line parameters:

**GxChassisExample**<chassis number><operation><param1><param2><param3><param4><param5>

## Sample Program Listing

---

```

/*****
FILE      : GxChassisExampleC.cpp
PURPOSE   : WIN32/LINUX example program for GX7xxx chassis
           using the GXCNT driver.
CREATED   : Dec 2005
COPYRIGHT : Copyright 2002-2013, Marvin Test Solutions, Inc.
COMMENTS  :

To compile the example:

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

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

#ifdef __BORLANDC__
#pragma hdrstop
#include <condefs.h>
USELIB("GxChassisBC.lib");
USERC("GxChassisExampleC.rc");
#endif // defined(__BORLANDC__)

```

```

//*****
//          DisplayMsg
//*****
void DisplayMsg(PCSTR lpszMsg)
{
#ifdef __GNUC__
    MessageBeep(0);
    MessageBox(0, lpszMsg, "GxChassis example program", MB_OK);
#else
    printf("\r\nGxChassis 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 use the GxChassis driver:\r\n"
        "Usage: GxChassisExample <chassis number> <operation> <param1> \\  

        <param2> <param3> <param4> <param5>\r\n"
        "\r\nWhere : \r\n"
        "chassis number: chassis number as was set by PXI Explorer\r\n"
        "operation one of the followings :\r\n"
        "  GetAlarmState = Get Alarm State (no parameters)\r\n"
        "  SetAlarmState = Set Alarm State, parameters:\r\n"
        "                  <param1>: 0-Disable, 1-Enable\r\n"
        "  GetAlarmTemp = Get Alarm Temperature (no parameters)\r\n"
        "  SetAlarmTemp = Set Alarm Temperature, parameters:\r\n"
        "                  <param1>: Alarm Temperature\r\n"
        "  GetShutdownTemp = Returns Shutdown Temperature (no \  

        parameters)\r\n"
        "  SetShutdownTemp = Set Shutdown Temperature, parameters:\r\n"
        "                  <param1>: 0-Disable, 1-Enable\r\n"
        "                  <param2>: Shutdown Temperature\r\n"
        "  GetVoltages = Get Power Supplies Voltages (no \  

        parameters)\r\n"
        "  GetTemps = Get Slots Temperatures (no parameters)\r\n"
        "  GetPxiTrigLine = Returns the specified PXI Trigger Line \  

        direction and mode\r\n"
        "  SetPxiTrigLine = Sets the specified PXI Trigger Line \  

        direction and mode\r\n"
        "                  <param1>: Trigger line 0-7\r\n"
    );
}

```

```

"          <param2>: Chassis Segments: 0-segment 0 to 1, \
                1-segment 1 to 2\r\n"
"          <param3>: Direction, 0-Disconnect, 1-Connect \
                Left to Right, 2-Connect Right to \
                Left\r\n"
"          <param4>: Primary side mode, 0-Monitor, \
                1-Drive Low, 2-Drive High\r\n"
"          <param5>: Secondary side mode, 0-Monitor, \
                1-Drive Low, 2-Drive High\r\n"
"  SUM = Print board summary\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"
);
exit(1);
}

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

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

/*****
MAIN

This main function receives between 0 to 2 parameters

GxChassis operation (e.g. SetShutdownTemp=Set Shutdown Temperature)

GetAlarmState:    Returns Alarm State (no parameters)
SetAlarmState:    Set Alarm State, parameters:
                  <param1>: 0-Disable, 1-Enable
GetAlarmTemp:     Returns Alarm Temperature (no parameters)
SetAlarmTemp:     Set Alarm Temperature, parameters:
                  <Temperature>: Alarm Threshold
GetShutdownTemp: Returns Shutdown Temperature (no parameters)
SetShutdownTemp: Set Shutdown Temperature, parameters:
                  <param1>: Shutdown Temperature
                  <param2>: 0-Disable, 1-Enable
GetVoltages:      Returns Power Supplies Voltages (no parameters)
GetTemps:         Returns Slots Temperatures (no parameters)
GetPxiTrigLine:   Returns the specified PXI Trigger Line direction and mode
SetPxiTrigLine:   Sets the specified PXI Trigger Line direction and mode,
                  parameters:
                  <param1>: Trigger line 0-7
                  <param2>: Chassis Segment: 0-segment 0 to 1,
                          1-segment 1 to 2
                  <param3>: Direction, 0-Disconnect, 1-Connect Left to

```

```

        Right, 2-Connect Right to Left
        <param4>: Primary side mode, 0-Monitor, 1-Drive Low,
                2-Drive High
        <param5>: Secondary side mode, 0-Monitor,
                1-Drive Low, 2-Drive High
SUM:          Print board summary
*****/
int main(int argc, char **argv)
{
    CHAR* szOperation;          // Board Operation
    SHORT nChassisNum;         // Chassis number
    SHORT nHandle;             // Board handle
    SHORT nStatus;             // Returned status
    SHORT nMode;
    SHORT nTrigline;           // PXI trigger bus line number
    SHORT nChassisSeg;         // Chassis Segment
    SHORT nDirection;          // PXI trigger bus direction
    SHORT nPrimSideMode;       // PXI trigger bus Primary side mode
    SHORT nSecSideMode;        // PXI trigger bus Secondary Side Mode
    DOUBLE dThreshold;
    BOOL bEnable;
    INT i;
    char sz[512];              // board summary

    // Check number of arguments received
    if (argc<2) DisplayUsage();
    nChassisNum=(SHORT)strtol(*(++argv), NULL, 0);
    szOperation = __strupr(*(++argv));

    GxChassisInitialize(nChassisNum, &nHandle, &nStatus);
    CheckStatus(nStatus);

    if (!strcmp(szOperation, "GETALARMMODE"))
    {
        GxChassisGetAlarmMode(nHandle, &nMode, &nStatus);
        CheckStatus(nStatus);
        printf("Alarm Mode is %s\r\n", nMode==0? "disabled": "enabled");
    }
    else if (!strcmp(szOperation, "SETALARMSTATE"))
    {
        // Check number of arguments received
        if (argc<3) DisplayUsage();
        nMode=(SHORT)strtol(*(++argv), NULL, 0);
        GxChassisSetAlarmMode(nHandle, nMode, &nStatus);
        CheckStatus(nStatus);
        printf("Alarm Mode is %s\r\n", nMode==0? "disabled": "enabled");
    }
    else if (!strcmp(szOperation, "GETALARMTEMP"))
    {
        GxChassisGetAlarmTemperature(nHandle, &dThreshold, &nStatus);
        CheckStatus(nStatus);
        printf("Temperature Alarm Threshold is %0.1f\r\n", dThreshold);
    }
    else if (!strcmp(szOperation, "SETALARMTEMP"))
    {
        // Check number of arguments received
        if (argc<3) DisplayUsage();
        dThreshold=(SHORT)strtol(*(++argv), NULL, 0);
        GxChassisSetAlarmTemperature(nHandle, dThreshold, &nStatus);
        CheckStatus(nStatus);
        GxChassisGetAlarmTemperature(nHandle, &dThreshold, &nStatus);
    }
}

```

```

        printf("Temperature Alarm Threshold is %0.1f\r\n", dThreshold);
    }
else if(!strcmp(szOperation, "GETSHUTDOWNTEMP"))
{
    GxChassisGetShutdownTemperature(nHandle, &bEnable, &dThreshold,
    &nStatus);
    CheckStatus(nStatus);
    printf("Shutdown Temperature is %0.1f\r\n", dThreshold);
}
else if(!strcmp(szOperation, "SETSHUTDOWNTEMP"))
{
    // Check number of arguments received
    if (argc<4) DisplayUsage();
    bEnable=(INT)strtol(*(++argv), NULL, 0);
    dThreshold=(SHORT)strtol(*(++argv), NULL, 0);
    GxChassisSetShutdownTemperature(nHandle, bEnable, dThreshold,
    &nStatus);
    CheckStatus(nStatus);
    GxChassisGetShutdownTemperature(nHandle, &bEnable, &dThreshold,
    &nStatus);
    printf("Shutdown Temperature is %0.1f\r\n", dThreshold);
}
else if(!strcmp(szOperation, "GETVOLTAGES"))
{
    DOUBLE adVoltage[8];
    GxChassisGetPowerSuppliesVoltages(nHandle, adVoltage, &nStatus);
    CheckStatus(nStatus);
    for (i=0; i<2; i++)
    {
        printf("%s +12V Power Supply Voltage=%0.1f\r\n", i==0?
        "Slots 1:10":"Slots 11:20", adVoltage[i*4]);
        printf("%s -12V Power Supply Voltage=%0.1f\r\n", i==0?
        "Slots 1:10":"Slots 11:20", adVoltage[i*4+1]);
        printf("%s 3.3V Power Supply Voltage=%0.1f\r\n", i==0?
        "Slots 1:10":"Slots 11:20", adVoltage[i*4+2]);
        printf("%s 5V Power Supply Voltage=%0.1f\r\n", i==0?
        "Slots 1:10":"Slots 11:20", adVoltage[i*4+3]);
    }
}
else if(!strcmp(szOperation, "GETTEMPS"))
{
    DOUBLE adTemp[20];
    GxChassisGetSlotsTemperatures(nHandle, adTemp, &nStatus);
    CheckStatus(nStatus);
    for (i=0; i<20; i++)
        printf("Slot %i Temperature=%0.1f\r\n", i+1, adTemp[i]);
}
else if(!strcmp(szOperation, "GETPXITRIGLINE"))
{
    // Check number of arguments received
    if (argc<5) DisplayUsage();
    nTrigline=(SHORT)strtol(*(++argv), NULL, 0);
    nChassisSeg=(SHORT)strtol(*(++argv), NULL, 0);
    GxChassisGetPxiTriggerLine(nHandle, nTrigline, nChassisSeg,
    &nDirection, &nPrimSideMode, &nSecSideMode, &nStatus);
    CheckStatus(nStatus);
    printf("PXi Trigger Line %i Segment %i settings: Direction=%i, \
    Primary Side Mode=%i, Secondary Side Mode=%i\r\n",
    nTrigline, nChassisSeg, nDirection, nPrimSideMode,
    nSecSideMode);
}
else if(!strcmp(szOperation, "SETPXITRIGLINE"))
{
    // Check number of arguments received

```

```

    if (argc<8) DisplayUsage();
    nTrigline=(SHORT)strtol(++argv, NULL, 0);
    nChassisSeg=(SHORT)strtol(++argv, NULL, 0);
    nDirection=(SHORT)strtol(++argv, NULL, 0);
    nPrimSideMode=(SHORT)strtol(++argv, NULL, 0);
    nSecSideMode=(SHORT)strtol(++argv, NULL, 0);
    GxChassisSetPxiTriggerLine(nHandle, nTrigline, nChassisSeg,
        nDirection, nPrimSideMode, nSecSideMode, &nStatus);
    CheckStatus(nStatus);
    printf("PXI Trigger Line %i Segment %i settings: Direction=%i, \
        Primary Side Mode=%i, Secondary Side Mode=%i\r\n",
        nTrigline, nChassisSeg, nDirection, nPrimSideMode,
        nSecSideMode);
}
else if (!strcmp(szOperation, "SUM"))
{
    // print board summary
    GxChassisGetBoardSummary(nHandle, sz, sizeof sz, &nStatus);
    CheckStatus(nStatus);
    printf("Board Summary: %s.\n", sz);
}
else
    DisplayUsage();

return 0;
}

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

```





# Chapter 5 - Functions Reference

## Introduction

---

The GxChassis 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 section.

All function parameter 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 accessing the chassis and is returned by the **GxChassisInitialize** function. The *nHandle* is used to identify the chassis when calling a function for programming and controlling the operation of the chassis. .
- All functions return a status with the last parameter named *pnStatus*. The *pnStatus* is zero if the function was successful, or less than zero for error conditions. The description of the error is available using the **GxChassisGetErrorString** function or by using a predefined constant, defined in the driver interface files: GxChassis.h, GxChassis.bas, GxChassis.pas or GxChassis.drv.
- Parameter names are prefixed as follows:

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

**Table 5-1: Parameter Name Prefixes**

## GxChassis Functions

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

Driver Functions	Description
<b>GxChassisGetAlarmMode</b>	Returns the Alarm Mode.
<b>GxChassisGetAlarmTemperature</b>	Returns the Alarm Temperature threshold settings.
<b>GxChassisGetBoardSummary</b>	Returns the board summary.
<b>GxChassisGetDriverSummary</b>	Returns the driver name and version.
<b>GxChassisGetErrorString</b>	Returns the error string associated with the specified error number.
<b>GxChassisGetFanSpeed</b>	Returns the fan speed and control settings
<b>GxChassisGetFanThresholdTemperatures</b>	Returns the fan low and high threshold temperatures.
<b>GxChassisGetPowerSuppliesVoltages</b>	Returns the backplane's eight power supplies voltages.
<b>GxChassisGetPxiTriggerLine</b>	Returns the specified PXI trigger line bridge direction mode and its direction configuration (left or right).
<b>GxChassisGetPxiTriggerLineLevel</b>	Returns the specified PXI trigger line segment's logic levels.
<b>GxChassisGetShutdownTemperature</b>	Returns the shutdown Temperature and active mode.
<b>GxChassisGetSlotsTemperatures</b>	Returns all slot temperature values
<b>GxChassisGetSlotsTemperaturesStates</b>	Returns all active slot temperature values.
<b>GxChassisGetSlotsTemperaturesStatistics</b>	Returns the slot with the lowest temperature, slot with the highest temperature and the average temperature of active slots.
<b>GxChassisGetSlotTemperature</b>	Returns the specified slot temperature value
<b>GxChassisGetTemperatureScale</b>	Returns the temperature scale used for setting or getting any temperature value.
<b>GxChassisGetTemperatureThresholdMode</b>	Returns the Temperature threshold operation mode.
<b>GxChassisInitialize</b>	Initializes the driver.
<b>GxChassisPanel</b>	Opens a virtual panel used to interactively control the GxChassis.
<b>GxChassisRecallSettings</b>	Loads and applies the settings as specified by the settings source parameter.
<b>GxChassisResetPxiTriggerLines</b>	Resets all PXI trigger lines for the specified segment
<b>GxChassisSetAlarmMode</b>	Sets the Alarm mode.
<b>GxChassisSetAlarmTemperature</b>	Sets the Alarm Temperature threshold.
<b>GxChassisSetFanSpeed</b>	Sets the fan speed and control settings
<b>GxChassisSetFanThresholdTemperatures</b>	Sets the fan low and high threshold temperatures.
<b>GxChassisSetPxiTriggerLine</b>	Sets the specified PXI trigger line bridge direction mode and its direction (Left or Right mode).
<b>GxChassisSetShutdownTemperature</b>	Sets the shutdown Temperature and active mode.
<b>GxChassisSetSlotsTemperaturesStates</b>	Sets the active state for slots monitoring temperature
<b>GxChassisSetTemperatureScale</b>	Sets the temperature scale used for setting or getting any temperature value.
<b>GxChassisSetTemperatureThresholdMode</b>	Sets the Temperature threshold operational mode.

## GxChassisGetAlarmMode

---

### Purpose

Returns the alarm mode.

### Syntax

**GxChassisGetAlarmMode** (*nHandle*, *pnMode*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pnMode</i>	PSHORT	Alarm mode is one of the following: <ol style="list-style-type: none"> <li>0. GXCHASSIS_OVER_TEMPERATURE_ALARM_DISABLE – Alarm disabled.</li> <li>1. GXCHASSIS_OVER_TEMPERATURE_ALARM_ENABLE – Alarm enabled (default).</li> <li>2. GXCHASSIS_OVER_TEMPERATURE_ALARM_ON – Alarm is on.</li> <li>3. GXCHASSIS_OVER_TEMPERATURE_ALARM_SNOOZE – Silence the Alarm after the Alarm threshold condition is met. If the alarm condition reoccurs, the buzzer will be activated again.</li> </ol>
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

When the Alarm is on (threshold condition was met, or set to On) both backplane buzzers will beep simultaneously in intervals of 10 seconds.

### Example

The following example returns the Alarm state:

```
SHORT  nMode, nStatus;
GxChassisGetAlarmMode (nHandle, &nMode, &nStatus);
```

### See Also

**GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetTemperatureThresholdMode**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisGetAlarmTemperature

---

### Purpose

Returns the alarm temperature threshold setting.

### Syntax

**GxChassisGetAlarmTemperature** (*nHandle*, *pdTemp*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pdTemp</i>	PDOUBLE	Alarm temperature threshold setting.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The Alarm temperature can be programmed to any value between  $-20^{\circ}\text{C}$  and  $+70^{\circ}\text{C}$ . The programmed temperature can be saved to the onboard EPROM and automatically loaded on the next system power up (using the front panel only).

The temperature resolution is 0.8 degree.

**Note:** Manufacture default Alarm temperature setting is  $+50^{\circ}\text{C}$ .

### Example

The following example returns the Alarm Temperature:

```
SHORT  nStatus;
DOUBLE dTemp
GxChassisGetAlarmTemperature(nHandle, &dTemp, &nStatus);
```

### See Also

**GxChassisSetAlarmTemperature**, **GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisGetBoardSummary

---

### Purpose

Returns a summary of chassis backplane information.

### Syntax

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

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>szSummary</i>	PSTR	Buffer to contain the returned board info (null terminated with 512 bytes).
<i>nSumMaxLen</i>	SHORT	Size of the buffer to contain the board info string.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The board summary retrieves an array of information from the chassis and the chassis backplane bridges. The information ranges from the component's serial number to firmware versions.

For example, the returned board info can be as follows:

“Model: GX7200, S/N: 00063, Firmware: N/A, Tested: Tue Jun 24 13:40:39 2008, User Defined Data: GX7200-0063

Module: Backplane, S/N: 00583\*-CN-CA-00, Firmware: N/A, Calibrated: N/A

Module: Gx7070 Bridge Segment A-B, S/N: 03251\*-GH-GD-00, Firmware: 0xE004, Calibrated: N/A”

Module: Gx7070 Bridge Segment B-C, S/N: 03252\*-GH-GD-00, Firmware: 0xE004, Calibrated: N/A”

### Example

```
CHAR sz[512];
SHORT nStatus;
```

```
GxChassisGetBoardSummary(nHandle, sz, sizeofsz, &nStatus);
```

### See Also

**GxChassisGetDriverSummary**, **GxChassisGetErrorString**

## GxChassisGetDriverSummary

---

### Purpose

Returns the driver name and version.

### Syntax

**GxChassisGetDriverSummary** (*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 and 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: GxChassis - Marvin Test Solutions's PXI Chassis Driver for the Gx7XX0 family, Version 1.0, Copyright © 2013Marvin Test Solutions, All rights reserved".

### Example

The following example prints the driver version:

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

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

### See Also

**GxPxiGetBoardSummary**, **GxChassisGetErrorString**

## GxChassisGetErrorString

---

### Purpose

Returns the error string associated with the specified error number.

### Syntax

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

### Parameters

Name	Type	Comments
<i>nError</i>	SHORT	Error number as returned by the <i>pnStatus</i> of any of the driver functions. See table below for possible values. The number should be a negative number, otherwise the function returns the “No error has occurred” string.
<i>pszMsg</i>	LPSTR	Buffer containing the returned error string (null terminated string).
<i>nErrorMaxLen</i>	SHORT	Size of the buffer <i>pszMsg</i> .
<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 type of driver.

Resource Errors	Description
-2	Unable to open the HW device/Service
-5	Unable to register the PCI device
-6	Unable to allocate system resource or memory for the PCI device
-8	Unable to create panel
-9	Unable to create a Windows timer
<b>Parameter Errors</b>	
-20	Invalid parameter
-22	Invalid board handle
-25	Invalid mode
-27	Invalid string length
<b>Board specific parameter error</b>	
-50	Invalid over temperature threshold value
-51	Invalid trigger direction, settings will result in a conflicting trigger lines direction.
-52	Invalid PXI trigger bus line direction, settings will result in a conflicting trigger lines direction
-53	Invalid PXI trigger bus line mode, settings will result in a conflicting trigger lines direction
-54	Invalid PXI trigger bus segment
-55	Invalid number of fan poles
-56	Invalid chassis type
<b>Board Errors/Warnings</b>	
-60	Controller is busy, return on timeout
-61	Controller communication error

-62	Error backplane left bridge, unable to communicate with the backplane left bridge
-63	Error backplane right bridge, unable to communicate with the backplane right bridge
-64	Error backplane bridges, unable to communicate with any of the backplane bridges
-65	Error backplane bridges, unable to detect any of the backplane bridges
-66	Error backplane left bridge, unable to detect with the backplane left bridge
-67	Error backplane right bridge, unable to detect with the backplane right bridge
<b>Miscellaneous Errors</b>	
-99	Invalid or unknown error number

**Example**

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

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

GxChassisInitialize(0, &Handle, &Status);
if (nStatus<0)
{
    GxChassisGetErrorString(nStatus, sz, sizeofsz, &nStatus);
    printf(sz);          // prints the error string returns
}
```



## GxChassisGetPowerSuppliesVoltages

---

### Purpose

Returns the backplane's eight power supplies voltages.

### Syntax

**GxChassisGetPowerSuppliesVoltages** (*nHandle*, *pdVoltage*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pdVoltage</i>	PDOUBLE	An array containing the backplane power supply voltages.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The returned eight power supply voltages are arranged as follows:

Array index	Power supplies voltage
0	+12V of Slots 1-10
1	-12V of Slots 1-10
2	3.3V of Slots 1-10
3	5V of Slots 1-10
4	+12V of Slots 11-21
5	-12V of Slots 11-21
6	3.3V of Slots 11-21
7	5V of Slots 11-21

### Example

The following example returns the backplane's eight power supplies voltages:

```
SHORT  nStatus;
DOUBLE adVoltage[8];
GxChassisGetPowerSuppliesVoltages (nHandle, adVoltage, &nStatus);
```

### See Also

**GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisGetFanSpeed

---

### Purpose

Returns the fan speed and control settings.

### Syntax

**GxChassisGetFanSpeed** (*nHandle*, *pnSpeedControl*, *pnSpeed*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pnSpeedControl</i>	PSHORT	Returns the fan speed control mode as follows: <ol style="list-style-type: none"> <li>GXCHASSIS_FAN_SPEED_MODE_AUTO: Fan speed is automatically controlled by the chassis. When mode is set to Auto the user can specify fan speed based on user defined high and low temperature thresholds.</li> <li>GXCHASSIS_FAN_SPEED_MODE_USER_DEFINED: Fan speed is specified by the user (<i>pnSpeed</i> value).</li> </ol>
<i>pnSpeed</i>	PSHORT	Returns the fan speed as follows: <ol style="list-style-type: none"> <li>GXCHASSIS_FAN_SPEED_MIN: Fan speed is at the minimum operational range.</li> <li>GXCHASSIS_FAN_SPEED_MID: Fan speed is at the middle operational range.</li> <li>GXCHASSIS_FAN_SPEED_MAX: Fan speed is at the maximum operational range.</li> </ol>
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

When the fan speed control is set to Auto (GXCHASSIS\_FAN\_SPEED\_MODE\_AUTO), the user can specify the temperature threshold range (low and high). When threshold is  $\leq$  low temp then the fan speed will be set to low, when threshold is  $\geq$  high temp the fan speed will be set to high. In between those threshold points the chassis will set the fan speed relative to the measured chassis temperature, e.g. if the fan's low threshold temperature is set to 20 and the high threshold temperature is set to 40 and the chassis temperature is 30 then the fan speed will be set to the medium speed.

When fan speed control is set to user defined (GXCHASSIS\_FAN\_SPEED\_MODE\_USER\_DEFINED) then the fan speed will stay constant according to the programmed *pnSpeed* value.

**Note:** this functionality is supported by bridgeboard revisions G and above.

### Example

The following example returns fan speed and control settings:

```
SHORT  nStatus;
SHORT  nSpeedControl, nSpeed;
GxChassisGetFanSpeed (nHandle, &nSpeedControl, &nSpeed, &nStatus);
```

### See Also

**GxChassisSetFanSpeed**, **GxChassisGetFanThresholdTemperatures**, **GxChassisGetErrorString**

## GxChassisGetFanThresholdTemperatures

---

### Purpose

Returns the fan low and high threshold temperatures.

### Syntax

**GxChassisGetFanThresholdTemperatures**(*nHandle*, *pdMinThreshold*, *pdMaxThreshold*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pdMinThreshold</i>	PDOUBLE	Returns the fan's low threshold temperature speed, value is either in Fahrenheit or Celsius as was set by the GxChassisSetTemperatureScalefunction call.
<i>pdMaxThreshold</i>	PDOUBLE	Returns the fan's high threshold temperature speed, value is either in Fahrenheit or Celsius as was set by the GxChassisSetTemperatureScalefunction call.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

When the fan speed control is set to Auto (GXCHASSIS\_FAN\_SPEED\_MODE\_AUTO), the user can specify the temperature threshold range (low and high). When the threshold is  $\leq$  low temp then the fan speed will be set to low, when the threshold is  $\geq$  high temp the fan speed will be set to high. In between those threshold points the chassis will set the fan speed relative to the measured chassis temperature, e.g. if the fan's low threshold temperature is set to 20 and the fan's high threshold temperature is set to 40 and the chassis temperature is 30 then the fan speed will be set to the medium speed.

The temperature resolution is 0.8 degree.

**Note:** this functionality is supported by bridgeboard revisions G and above.

### Example

The following example returns the fan low and high threshold temperatures.

```
SHORT  nStatus;
DOUBLE dMinThreshold, dMaxThreshold;
GxChassisGetFanThresholdTemperatures (nHandle, &dMinThreshold, &dMaxThreshold, &nStatus);
```

### See Also

**GxChassisSetFanThresholdTemperatures, GxChassisSetFanSpeed, GxChassisGetErrorString**

## GxChassisGetPxiTriggerLine

---

### Purpose

Returns the specified PXI trigger line bridge direction mode and its Left and Right mode.

### Syntax

**GxChassisGetPxiTriggerLine** (*nHandle*, *nLine*, *nSegment*, *pnDirection*, *pnPrimaryMode*, *pnSecondaryMode*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nLine</i>	SHORT	Specified PXI trigger line of the specified PXI chassis Segment: 0. GXCHASSIS_PXI_TRIGGER_BUS_LINE0 - PXI trigger line 0 1. GXCHASSIS_PXI_TRIGGER_BUS_LINE1 - PXI trigger line 1 2. GXCHASSIS_PXI_TRIGGER_BUS_LINE2 - PXI trigger line 2 3. GXCHASSIS_PXI_TRIGGER_BUS_LINE3 - PXI trigger line 3 4. GXCHASSIS_PXI_TRIGGER_BUS_LINE4 - PXI trigger line 4 5. GXCHASSIS_PXI_TRIGGER_BUS_LINE5 - PXI trigger line 5 6. GXCHASSIS_PXI_TRIGGER_BUS_LINE6 - PXI trigger line 6 7. GXCHASSIS_PXI_TRIGGER_BUS_LINE7 - PXI trigger line 7
<i>nSegment</i>	SHORT	Specified PXI chassis Segments: 0. GXCHASSIS_SEGMENT_0_TO_SEGMENT_1 – Segment Slots 2:8 connecting to Segment Slots 9:14 (chassis left side bridge). 1. GXCHASSIS_SEGMENT_1_TO_SEGMENT_2 - Segment Slots 9:14 connecting to Segment Slots 15:21 (chassis right side bridge).
<i>pnDirection</i>	PSHORT	Returns the Specified PXI trigger line segment direction as follows: 0. GXCHASSIS_PXI_TRIGGER_BUS_LINE_DISCONNECT - Disconnect the PXI trigger line from the Right segment and the Left segment. I.e. PXI trigger line is not connected to either segment. 1. GXCHASSIS_PXI_TRIGGER_BUS_LINE_LEFT_TO_RIGHT - Connect the PXI trigger line direction to cross from Left segment to the Right segment. 2. GXCHASSIS_PXI_TRIGGER_BUS_LINE_RIGHT_TO_LEFT - Connect the PXI trigger line direction to cross from Right segment to the Left segment.
<i>pnPrimaryMode</i>	PSHORT	Returns the Specified PXI trigger line primary side mode, modes are as follows: 0. GXCHASSIS_PXI_TRIGGER_BUS_LINE_MONITOR: the primary segment side (left) does not drive the specified trigger line (default). 1. GXCHASSIS_PXI_TRIGGER_BUS_LINE_DRIVE_LOW: the primary segment side (left) drives the specified trigger line low (default). 2. GXCHASSIS_PXI_TRIGGER_BUS_LINE_DRIVE_HIGH: the primary segment side (left) drives the specified trigger line high (default). <b>Note:</b> this functionality is supported by bridgeboard revisions G and above; previous bridgeboard revision will not be affected.
<i>pnSecondaryMode</i>	PSHORT	Returns the Specified PXI trigger line secondary side mode, modes are as follows:

0. GXCHASSIS\_PXI\_TRIGGER\_BUS\_LINE\_MONITOR: the secondary segment side (right) does not drive the specified trigger line (default).
1. GXCHASSIS\_PXI\_TRIGGER\_BUS\_LINE\_DRIVE\_LOW: the secondary segment side (right) drives the specified trigger line low (default).
2. GXCHASSIS\_PXI\_TRIGGER\_BUS\_LINE\_DRIVE\_HIGH: the secondary segment side (right) drives the specified trigger line high (default).

**Note:** this functionality is supported by bridgeboard revisions G and above; previous bridgeboard revision will not be affected.

*pnStatus*                    PSHORT    Returned status: 0 on success, negative number on failure.

### Comments

The user can monitor the specified trigger line level, high or low, using the **GxChassisGetPxiTriggerLineLevels** (supported by bridgeboard revisions G and above).

### Example

The following example returns PXI trigger line 0 Segment Slots 2:7 connecting to Segment Slots 8:13 settings:

```
SHORT  nStatus;
SHORT  nDirection, nPrimaryMode, nSecondaryMode;
GxChassisGetPxiTriggerLine (nHandle, GXCHASSIS_PXI_TRIGGER_BUS_LINE0,
GXCHASSIS_SEGMENT_0_TO_SEGMENT_1, &nDirection, &nPrimaryMode, &nSecondaryMode, nStatus);
```

### See Also

**GxChassisSetPxiTriggerLine, GxChassisGetPxiTriggerLineLevels, GxChassisGetErrorString**

## GxChassisGetPxiTriggerLineLevels

---

### Purpose

Returns the specified PXI trigger line segment Left and Right logic levels.

### Syntax

**GxChassisGetPxiTriggerLineLevels** (*nHandle*, *nLine*, *nSegment*, *pnPrimary*, *pnSecondary*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nLine</i>	SHORT	Specified PXI trigger line of the specified PXI chassis Segment: 0. GXCHASSIS_PXI_TRIGGER_BUS_LINE0 - PXI trigger line 0 1. GXCHASSIS_PXI_TRIGGER_BUS_LINE1 - PXI trigger line 1 2. GXCHASSIS_PXI_TRIGGER_BUS_LINE2 - PXI trigger line 2 3. GXCHASSIS_PXI_TRIGGER_BUS_LINE3 - PXI trigger line 3 4. GXCHASSIS_PXI_TRIGGER_BUS_LINE4 - PXI trigger line 4 5. GXCHASSIS_PXI_TRIGGER_BUS_LINE5 - PXI trigger line 5 6. GXCHASSIS_PXI_TRIGGER_BUS_LINE6 - PXI trigger line 6 7. GXCHASSIS_PXI_TRIGGER_BUS_LINE7 - PXI trigger line 7
<i>nSegment</i>	SHORT	Specified PXI chassis Segments: 0. GXCHASSIS_SEGMENT_0_TO_SEGMENT_1 – Segment Slots 2:8 connecting to Segment Slots 9:14 (chassis left side bridge). 1. GXCHASSIS_SEGMENT_1_TO_SEGMENT_2 - Segment Slots 9:14 connecting to Segment Slots 15:21 (chassis right side bridge).
<i>pnPrimary</i>	PSHORT	Returns the Specified PXI trigger line primary side logic level: 0. The primary segment side (left) specified trigger line is low. 1. The primary segment side (left) specified trigger line is high..
<i>pnSecondary</i>	PSHORT	Returns the Specified PXI trigger line secondary side logic level: 0. The secondary segment side (right) specified trigger line is low. 1. The secondary segment side (right) specified trigger line is high.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

**Note:** this functionality is supported by bridge board revisions G and above.

### Example

The following example returns PXI trigger line 0 Segment Slots 2:7 connecting to Segment Slots 8:13 levels:

```
SHORT nStatus;
SHORT nPrimary, nSecondary;
GxChassisGetPxiTriggerLine (nHandle, GXCHASSIS_PXI_TRIGGER_BUS_LINE0,
GXCHASSIS_SEGMENT_0_TO_SEGMENT_1, &nPrimary, &nSecondary, nStatus);
```

### See Also

**GxChassisSetPxiTriggerLine**, **GxChassisGetErrorString**

## GxChassisGetShutdownTemperature

---

### Purpose

Returns the shutdown temperature and shutdown state.

### Syntax

**GxChassisGetShutdownTemperature** (*nHandle*, *pbEnable*, *pdThreshold*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pbEnable</i>	PBOOL	Shutdown state: 0. Disabled. 1. Enabled (default).
<i>pdThreshold</i>	PDOUBLE	Shutdown Temperature threshold settings, value can be between +20°C to +70°C.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The programmable over temperature shutdown can be programmed to any value between  $-20^{\circ}\text{C}$  and  $+70^{\circ}\text{C}$ . The programmed temperature can be saved to the onboard EEPROM and be automatically loaded on the next system power up (using the front panel only).

The temperature resolution is 0.8 degree.

**Note:** the manufacture default threshold is programmed to  $+70^{\circ}\text{C}$ .

### Example

The following example returns the shutdown temperature and active mode:

```
SHORT  nStatus;
BOOL   bEnable
DOUBLE dThreshold;
GxChassisGetShutdownTemperature (nHandle, &bEnable, &dThreshold, &nStatus);
```

### See Also

**GxChassisSetTemperatureThresholdMode, GxChassisSetAlarmMode, GxChassisSetAlarmTemperature, GxChassisSetShutdownTemperature, GxChassisGetErrorString**

## GxChassisGetSlotsTemperatures

---

### Purpose

Returns all slot temperatures.

### Syntax

**GxChassisGetSlotsTemperatures** (nHandle, pdTemp, pnStatus)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pdTemp</i>	SHORT	Array holding all measured slot temperatures. Measured temperature of slot 1 returned in array index 0.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The temperature resolution is 0.8 degree.

**Note:** Slots' temperatures are measured regardless if the slots are active or not. See the **GxChassisSetSlotsTemperaturesStates** function for details.

### Example

The following example returns all twenty measured slots' temperatures into an array:

```
SHORT  nStatus;
DOUBLE adTemp[20];
GxChassisGetSlotsTemperatures(nHandle, adTemp, &nStatus);
```

### See Also

**GxChassisSetSlotsTemperaturesStates**, **GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**



## GxChassisGetSlotsTemperaturesStates

---

### Purpose

Returns all active (enabled) slot temperatures.

### Syntax

**GxChassisGetSlotsTemperaturesStates** (*nHandle*, *pdwStates*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pdwStates</i>	PDWORD	Returns slots with active or enabled temperature monitoring, bits 0 through 19 represents slots 1 through 20. <ul style="list-style-type: none"> <li>• Bit high – specified slot is enabled.</li> <li>• Bit low – specified slot is disabled.</li> </ul>
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Only active (enabled) slots determine if alarm threshold or shutdown threshold conditions are met.

### Example

The following example returns the slots' temperatures' active states:

```
SHORT  nStatus;
DWORD  dwStates;
GxChassisGetSlotsTemperaturesStates(nHandle, &dwStates, &nStatus);
```

### See Also

**GxChassisSetSlotsTemperaturesStates**, **GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisGetSlotsTemperaturesStatistics

---

### Purpose

Returns the slot with the lowest temperature, the slot with the highest temperature and the average temperature of the active slots.

### Syntax

**GxChassisGetSlotsTemperaturesStatistics** (*nHandle*, *pnMinTempSlot*, *pdMinTemp*, *pnMaxTempSlot*, *pdMaxTemp*, *pdAveTemp*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pnMinTempSlot</i>	PSHORT	The slot number with the lowest temperature out of all active slots.
<i>pdMinTemp</i>	PDOUBLE	The temperature of the slot number with the lowest temperature out of all active slots.
<i>pnMaxTempSlot</i>	PSHORT	The slot number with the highest temperature out of all active slots.
<i>pdMaxTemp</i>	PDOUBLE	The temperature of the slot number with the highest temperature out of all active slots.
<i>pdAveTemp</i>	PDOUBLE	The average temperature of all active slots.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

This function returns all the slots' temperatures and out of the active slots determines the slot with the lowest temperature, the slot with the highest temperature and the average temperature.

The function can be most useful to determine shutdown threshold and alarm threshold settings as well as monitoring the slots' temperatures range.

The temperature resolution is 0.8 degree.

### Example

The following example returns the minimum, maximum and average temperatures of the active slots:

```
SHORT  nMinTempSlot, nMaxTempSlot, nStatus;
DOUBLE dMinTemp, dMaxTemp, dAveTemp;
GxChassisGetSlotsTemperaturesStatistics(nHandle, &nMinTempSlot, &dMinTemp, &nMaxTempSlot,
&dMaxTemp, &dAveTemp, &nStatus);
```

### See Also

**GxChassisSetSlotsTemperaturesStates**, **GxChassisSetTemperatureThresholdMode**,  
**GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetShutdownTemperature**,  
**GxChassisGetErrorString**

## GxChassisGetSlotTemperature

---

### Purpose

Returns the specified slot temperature.

### Syntax

**GxChassisGetSlotTemperature** (*nHandle*, *nSlot*, *pdTemp*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nSlot</i>	SHORT	Specified slot temperature. Slot number can be from 1 to 21.
<i>pdTemp</i>	SHORT	Array holding all measured slots' temperatures. Measured temperature of slot 1 returned in array cell number 0.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The temperature resolution is 0.8 degree.

**Note:** Slots' temperatures are measured regardless if the slots are active or not. See the **GxChassisSetSlotsTemperaturesStates** function for details.

### Example

The following example returns slot number 2's temperature:

```
SHORT  nStatus;
DOUBLE aTemp;
GxChassisGetSlotTemperature(nHandle, 2, &dTemp, &nStatus);
```

### See Also

**GxChassisSetSlotsTemperaturesStates**, **GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisGetTemperatureScale

---

### Purpose

Returns the temperature scale used for setting or getting any temperature value.

### Syntax

**GxChassisGetTemperatureScale** (*nHandle*, *pnScale*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pnScale</i>	PSHORT	Temperature scale: 0. GXCHASSIS_TEMPERATURE_SCALE_METRIC 1. GXCHASSIS_TEMPERATURE_SCALE_ENGLISH
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Once the temperature scale is set, the same scale will be applied to all temperature values, e.g. shutdown temperature. The temperature scale setting is saved to the host computer.

### Example

The following example returns the temperature scale:

```
SHORT  nScale, nStatus;
GxChassisGetTemperatureScale(nHandle, &nScale, &nStatus);
```

### See Also

**GxChassisSetTemperatureScale**, **GxChassisSetSlotsTemperaturesStates**,  
**GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**,  
**GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisGetTemperatureThresholdMode

---

### Purpose

Returns the Temperature threshold operation mode.

### Syntax

**GxChassisGetTemperatureThresholdMode** (*nHandle*, *pnMode*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pnMode</i>	PSHORT	Temperature threshold operation modes are: 0. GXCHASSIS_OVER_TEMPERATURE_MODE_MAX_SLOT (default) 1. GXCHASSIS_OVER_TEMPERATURE_MODE_AVERAGE_SLOTS
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The temperature threshold operational mode dictates how the alarm threshold and shutdown threshold will be activated. The modes are: GXCHASSIS\_OVER\_TEMPERATURE\_MODE\_MAX\_SLOT:

- Shutdown activated when any of the enabled slots' temperature is above the shutdown temperature.
- Alarm activated when any of the enabled slots' temperature is above the alarm temperature.

GXCHASSIS\_OVER\_TEMPERATURE\_MODE\_AVERAGE\_SLOTS:

- Shutdown activated when the average temperature of all active slots are above the shutdown temperature.
- Alarm activated when the average temperature of all active slots are above the alarm temperature.

### Example

The following example returns the Temperature threshold operational mode:

```
SHORT nMode, nStatus;
GxChassisGetTemperatureThresholdMode(nHandle, &nMode, &nStatus);
```

### See Also

**GxChassisSetTemperatureThresholdMode, GxChassisSetAlarmMode, GxChassisSetAlarmTemperature, GxChassisSetShutdownTemperature, GxChassisGetErrorString**

## GxChassisInitialize

---

### Purpose

Initialize the driver for the specified chassis number.

### Syntax

**GxChassisInitialize** (*nChassis*, *pnHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nChassis</i>	SHORT	PXI Chassis number.
<i>pnHandle</i>	PSHORT	Returned handle for a <b>GX72XX</b> Chassis. The handle is set to zero on error and <> 0 on success.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

This function returns a handle that can be used with other GxChassis functions to program the chassis. The function does not change any of the chassis' settings.

### Example

The following example initializes two PXI Chassis 1 and 2.

```
SHORT  nHandle1, nHandle2, nStatus;

GxChassisInitialize(1, &nHandle1, &nStatus);
if(nHandle1==0)
    printf("Unable to Initialize the board");

GxChassisInitialize(2, &nHandle2, &nStatus);
if(nHandle2==0)
    printf("Unable to Initialize the board");
```

### See Also

**GxChassisGetErrorString**

## GxChassisPanel

---

### Purpose

Opens a virtual panel used to interactively control the GxChassis mainframe.

### Syntax

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

### Parameters

Name	Type	Comments
<i>pnHandle</i>	PSHORT	Handle to a GX72XX Chassis. This number may be zero if the board is to be initialized by the panel window.
<i>hwndParent</i>	HWND	Sets the 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 and 1 for modal window.
<i>phwndPanel</i>	LPHWND	Returned window handle for the panel (for modeless panel only).
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

This function is used to create the virtual panel. The panel window may be opened 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 user closes the window. In that case the *pnHandle* returns the handle created by the user using the panel Initialize dialog. This handle then may be used when calling other GxChassis 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 the calling program to dispatch window messages to this window, so that the window can respond to messages.

### Example

The following example opens the panel in modal mode:

```
HWND    hwndPanel;
SHORT   nHandle=0, nStatus;
...
GxChassisPanel (&nHandle, 0, 1, &hwndPanel, &nStatus);
```

### See Also

**GxChassisInitialize**, **GxChassisGetErrorString**

## GxChassisRecallSettings

---

### Purpose

Loads and applies the settings as specified by the settings source parameter.

### Syntax

**GxChassisRecallSettings** (*nHandle*, *nSettingSource*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nSettingSource</i>	SHORT	Recall Settings source are: 0. GXCHASSIS_RECALL_FACTORY_SETTINGS - Loads and applies the factory default settings 1. GXCHASSIS_RECALL_USER_SETTINGS - Loads and applies the last saved users' settings from the onboard EEPROM.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Factory default settings are:

- All slots' temperatures enabled.
- Temperature threshold mode is set to Max Temp Mode.
- Shutdown temperature is 70°C.
- Shutdown temperature state enabled
- Alarm temperature is 50°C.
- Alarm state disabled.
- PXI Trigger lines are all disabled.

**Note:** Users can only save their settings to the on-board EEPROM when running the front panel.

### Example

The following example loads and applies the last saved user settings:

```
SHORT nStatus;
GxChassisRecallSettings(nHandle, GXCHASSIS_RECALL_USER_SETTINGS, &nStatus);
```

### See Also

**GxChassisGetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetTemperatureThresholdMode**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**



## GxChassisResetPxiTriggerLines

---

### Purpose

Resets all PXI trigger lines in a specified segment.

### Syntax

**GxChassisResetPxiTriggerLines** (*nHandle*, *nSegment*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nSegment</i>	SHORT	Specified PXI chassis Segments: <ol style="list-style-type: none"> <li>0. GXCHASSIS_SEGMENT_0_TO_SEGMENT_1 – Segment Slots 2:8 connecting to Segment Slots 9:14 (chassis left side bridge).</li> <li>1. GXCHASSIS_SEGMENT_1_TO_SEGMENT_2–Segment Slots 9:14 connecting to Segment Slots 15:21 (chassis right side bridge).</li> </ol>
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

After calling this function the specified segment settings will be as follows:

Direction: A segment's primary and secondary sides are disconnected  
(GXCHASSIS\_PXI\_TRIGGER\_BUS\_LINE\_DISCONNECT).

Primary side: monitor state (GXCHASSIS\_PXI\_TRIGGER\_BUS\_LINE\_MONITOR).

Secondary side: monitor state (GXCHASSIS\_PXI\_TRIGGER\_BUS\_LINE\_MONITOR).

### Example

The following example resets the first segment:

```
SHORT nStatus;
GxChassisResetPxiTriggerLines (nHandle, GXCHASSIS_SEGMENT_0_TO_SEGMENT_1, &nStatus);
```

### See Also

**GxChassisSetPxiTriggerLine**, **GxChassisgetPxiTriggerLine**, **GxChassisGetErrorString**

## GxChassisSetAlarmMode

---

### Purpose

Sets the over temperature alarm mode.

### Syntax

**GxChassisSetAlarmMode** (*nHandle*, *nMode*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nMode</i>	SHORT	Over Temperature Alarm mode is one of the following: <ol style="list-style-type: none"> <li>0. GXCHASSIS_OVER_TEMPERATURE_ALARM_DISABLE – Alarm disabled.</li> <li>1. GXCHASSIS_OVER_TEMPERATURE_ALARM_ENABLE – Alarm enabled.</li> <li>2. GXCHASSIS_OVER_TEMPERATURE_ALARM_ON – Alarm is on.</li> <li>3. GXCHASSIS_OVER_TEMPERATURE_ALARM_SNOOZE – Silence the Alarm after the Alarm threshold condition is met. If the alarm condition reoccurs, the buzzer will activate again.</li> </ol>
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

When the Alarm is on (threshold condition was met or set to On) both backplane buzzers will beep simultaneously in intervals of 10 seconds.

### Example

The following example enables the Over Temperature Alarm:

```
SHORT  nStatus;
GxChassisSetAlarmMode (nHandle, GXCHASSIS_OVER_TEMPERATURE_ALARM_ENABLE, &nStatus);
```

### See Also

**GxChassisGetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetTemperatureThresholdMode**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisSetAlarmTemperature

---

### Purpose

Sets the alarm temperature threshold.

### Syntax

**GxChassisSetAlarmTemperature** (*nHandle*, *dTemp*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>dTemp</i>	DOUBLE	Alarm temperature threshold settings
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The programmable Alarm temperature can be programmed to any value between  $-20^{\circ}\text{C}$  and  $+70^{\circ}\text{C}$ . The programmed temperature can be saved to the onboard EEPROM and be automatically loaded on the next system power up (using the front panel only).

The temperature resolution is 0.8 degree.

**Note:** Manufacturer default Alarm temperature is  $+50^{\circ}\text{C}$ .

### Example

The following example sets the Alarm temperature to  $45^{\circ}\text{C}$ :

```
SHORT nStatus;
GxChassisSetAlarmTemperature (nHandle, 45, &nStatus);
```

### See Also

**GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisGetFanSpeed

---

### Purpose

Sets the fan speed and control settings.

### Syntax

**GxChassisSetFanSpeed** (*nHandle*, *nSpeedControl*, *nSpeed*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nSpeedControl</i>	SHORT	Sets the fans speed control mode as follows: <ol style="list-style-type: none"> <li>GXCHASSIS_FAN_SPEED_MODE_AUTO: fan speed is automatically controlled by the chassis. When mode is set to Auto the user can specify fan speed based on user defined high and low temperature thresholds.</li> <li>GXCHASSIS_FAN_SPEED_MODE_USER_DEFINED: Fans speed is specified by the user (<i>pnSpeed</i> value).</li> </ol>
<i>nSpeed</i>	SHORT	Sets the fans speed as follows: <ol style="list-style-type: none"> <li>GXCHASSIS_FAN_SPEED_MIN: Fan speed is at the minimum operational range.</li> <li>GXCHASSIS_FAN_SPEED_MID: Fan speed is at the middle operational range.</li> <li>GXCHASSIS_FAN_SPEED_MAX: Fan speed is at the maximum operational range.</li> </ol>
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

When the fan speed control is set to Auto (GXCHASSIS\_FAN\_SPEED\_MODE\_AUTO), the user can specify the temperature threshold range (low and high). When threshold is  $\leq$  low temp then the fan speed will be set to low, when threshold is  $\geq$  high temp the fan speed will be set to high. In between these threshold points the chassis will set the fan speed relative to the measured chassis temperature, e.g. if the fan's low threshold temperature is set to 20 and the high threshold temperature is set to 40 and the chassis temperature is 30 then the fan speed will be set to the medium speed.

When fan speed control is set to user defined (GXCHASSIS\_FAN\_SPEED\_MODE\_USER\_DEFINED) then the fan speed will stay constant according to the programmed *pnSpeed* value.

**Note:** this functionality is supported by bridgeboard revisions G and above.

### Example

The following example sets the fan speed to Auto:

```
SHORT nStatus;
GxChassisSetFanSpeed (nHandle, GXCHASSIS_FAN_SPEED_MODE_AUTO, 0, &nStatus);
```

### See Also

**GxChassisGetFanSpeed**, **GxChassisSetFanThresholdTemperatures**, **GxChassisGetErrorString**

## GxChassisSetFanThresholdTemperatures

---

### Purpose

Sets the fan low and high threshold temperatures.

### Syntax

**GxChassisSetFanThresholdTemperatures**(*nHandle*, *dMinThreshold*, *dMaxThreshold*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>dMinThreshold</i>	DOUBLE	Fan's low threshold temperature speed. Value is either in Fahrenheit or Celsius as was set by the GxChassisSetTemperatureScale function call.
<i>dMaxThreshold</i>	DOUBLE	Fan's high threshold temperature speed, Value is either in Fahrenheit or Celsius as was set by the GxChassisSetTemperatureScale function call.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

When the fan speed control is set to Auto (GXCHASSIS\_FAN\_SPEED\_MODE\_AUTO), the user can specify the temperature threshold range (low and high). When threshold is <=low temp then the fan speed will be set to low, when the threshold is >=high temp the fan speed will be set to high. In between these threshold points the chassis will set the fan speed relative to the measured chassis temperature, e.g. if fan low threshold temperature is set to 20 and the high threshold temperature is set to 40 and the chassis temperature is 30 then the fan speed will be set to the medium speed.

The temperature resolution is 0.8 degree.

**Note:** this functionality is supported by bridgeboard revisions G and above.

### Example

The following example sets the fan low and high threshold temperatures in Celsius.

```
SHORT nStatus;
GxChassisSetFanThresholdTemperatures (nHandle, 20, 40, &nStatus);
```

### See Also

**GxChassisGetFanThresholdTemperatures, GxChassisSetFanSpeed, GxChassisGetErrorString**

## GxChassisSetPxiTriggerLine

---

### Purpose

Sets the specified PXI trigger line bridge direction mode and the Left and Right mode.

### Syntax

**GxChassisSetPxiTriggerLine** (*nHandle*, *nLine*, *nSegment*, *ucDirection*, *nPrimaryMode*, *nSecondaryMode*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nLine</i>	SHORT	Specified PXI trigger line of the specified PXI chassis Segment: 0. GXCHASSIS_PXI_TRIGGER_BUS_LINE0 - PXI trigger line 0 1. GXCHASSIS_PXI_TRIGGER_BUS_LINE1 - PXI trigger line 1 2. GXCHASSIS_PXI_TRIGGER_BUS_LINE2 - PXI trigger line 2 3. GXCHASSIS_PXI_TRIGGER_BUS_LINE3 - PXI trigger line 3 4. GXCHASSIS_PXI_TRIGGER_BUS_LINE4 - PXI trigger line 4 5. GXCHASSIS_PXI_TRIGGER_BUS_LINE5 - PXI trigger line 5 6. GXCHASSIS_PXI_TRIGGER_BUS_LINE6 - PXI trigger line 6 7. GXCHASSIS_PXI_TRIGGER_BUS_LINE7 - PXI trigger line 7
<i>nSegment</i>	SHORT	Specified PXI chassis Segments: 0. GXCHASSIS_SEGMENT_0_TO_SEGMENT_1 – Segment Slots 2:7 connecting to Segment Slots 8:14 (chassis left side bridge). 1. GXCHASSIS_SEGMENT_1_TO_SEGMENT_2 - Segment Slots 8:14 connecting to Segment Slots 15:21(chassis right side bridge).
<i>nDirection</i>	SHORT	Specified PXI trigger line segment direction as follows: 0. GXCHASSIS_PXI_TRIGGER_BUS_LINE_DISCONNECT - Disconnect the PXI trigger line from the Right segment and the Left segment. I.e. PXI trigger line is isolated between the left and right segment.. 1. GXCHASSIS_PXI_TRIGGER_BUS_LINE_LEFT_TO_RIGHT - Connect the PXI trigger line direction to cross from the Left segment to the Right segment. 2. GXCHASSIS_PXI_TRIGGER_BUS_LINE_RIGHT_TO_LEFT - Connect the PXI trigger line direction to cross from the Right segment to the Left segment.
<i>nPrimaryMode</i>	SHORT	Specified PXI trigger line primary side mode, modes are as follows: 0. GXCHASSIS_PXI_TRIGGER_BUS_LINE_MONITOR: the primary segment side (left) does not drive the specified trigger line (default). 1. GXCHASSIS_PXI_TRIGGER_BUS_LINE_DRIVE_LOW: the primary segment side (left) drives the specified trigger line low (default). 2. GXCHASSIS_PXI_TRIGGER_BUS_LINE_DRIVE_HIGH: the primary segment side (left) drives the specified trigger line high (default). <b>Note:</b> this functionality is supported by bridgeboard revisions G and above; previous bridgeboard revision will not be affected.
<i>nSecondaryMode</i>	SHORT	Specified PXI trigger line secondary side mode, modes are as follows: 0. GXCHASSIS_PXI_TRIGGER_BUS_LINE_MONITOR: the secondary segment side (right) does not drive the specified trigger line (default).

1. `GXCHASSIS_PXI_TRIGGER_BUS_LINE_DRIVE_LOW`: the secondary segment side (right) drives the specified trigger line low (default).
2. `GXCHASSIS_PXI_TRIGGER_BUS_LINE_DRIVE_HIGH`: the secondary segment side (right) drives the specified trigger line high (default).

**Note:** this functionality is supported by bridgeboard revisions G and above; previous bridge board revision will not be affected.

*pnStatus*                    PSHORT    Returned status: 0 on success, negative number on failure.

### Comments

The user can monitor the specified trigger line level, high or low, using the `GxChassisGetPxiTriggerLineLevels` (supported by bridgeboard revisions G and above).

### Example

The following example sets PXI trigger line 0 Segment Slots 2:7 connecting to Segment Slots 8:13 settings:

```
SHORT  nStatus;
SHORT  nDirection, nPrimaryMode, nSecondaryMode;
GxChassisSetPxiTriggerLine (nHandle, GXCHASSIS_PXI_TRIGGER_BUS_LINE0,
                           GXCHASSIS_SEGMENT_0_TO_SEGMENT_1,
                           GXCHASSIS_PXI_TRIGGER_BUS_LINE_LEFT_TO_RIGHT,
                           GXCHASSIS_PXI_TRIGGER_BUS_LINE_MONITOR,
                           GXCHASSIS_PXI_TRIGGER_BUS_LINE_MONITOR, nStatus);
```

### See Also

`GxChassisSetPxiTriggerLine`, `GxChassisGetPxiTriggerLineLevels`, `GxChassisGetErrorString`

## GxChassisSetShutdownTemperature

---

### Purpose

Sets the shutdown temperature and shutdown state.

### Syntax

**GxChassisSetShutdownTemperature** (*nHandle*, *bEnable*, *dThreshold*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>bEnable</i>	BOOL	Shutdown state: 0. Disabled. 1. Enabled (default).
<i>dThreshold</i>	DOUBLE	Shutdown Temperature threshold settings, value can be between +20°C to +70°C.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The programmable over temperature shutdown can be programmed to any value between +20°C and +70°C. The programmed temperature can be saved to the onboard EEPROM and be automatically loaded on the next system power up (using the front panel only).

The temperature resolution is 0.8 degree.

**Note:** Manufacturer default threshold is programmed to +70°C.

### Example

The following example sets the shutdown temperature to 50°C and enables the shutdown:

```
SHORT  nStatus;
GxChassisSetShutdownTemperature (nHandle, TRUE, 50, &nStatus);
```

### See Also

**GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisGetAlarmTemperature**, **GxChassisGetShutdownTemperature**, **GxChassisGetErrorString**



## GxChassisSetSlotsTemperaturesStates

---

### Purpose

Sets (enables) all slots for active temperature monitoring

### Syntax

**GxChassisSetSlotsTemperaturesStates** (*nHandle*, *dwStates*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>dwStates</i>	DWORD	Defines slots that will be actively monitored for temperature. Bits 0 through 20 represents slots 1 through 21. <ul style="list-style-type: none"> <li>• Bit high – specified slot enabled.</li> <li>• Bit low – specified slot disabled.</li> </ul>
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Only active (enabled) slots determine if alarm threshold or shutdown threshold conditions are met.

### Example

The following example enables slots 1 through 6 only:

```
SHORT  nStatus;
GxChassisSetSlotsTemperaturesStates (nHandle, 0x3F, &nStatus);
```

### See Also

**GxChassisGetSlotsTemperaturesStates**, **GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**, **GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisSetTemperatureScale

---

### Purpose

Sets the temperature scale used for setting or getting any temperature value.

### Syntax

**GxChassisGetTemperatureScale** (*nHandle*, *nScale*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>nScale</i>	SHORT	Temperature scale: 0. GXCHASSIS_TEMPERATURE_SCALE_METRIC 1. GXCHASSIS_TEMPERATURE_SCALE_ENGLISH
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Once the temperature scale is set, the same scale will be applied to all temperature values, e.g. shutdown temperature. The temperature scale setting is saved to the host computer.

### Example

The following example sets the temperature scale used for setting or getting any temperature value to the English scale:

```
SHORT  nStatus;
GxChassisSetTemperatureScale (nHandle, GXCHASSIS_TEMPERATURE_SCALE_ENGLISH, &nStatus);
```

### See Also

**GxChassisGetTemperatureScale**, **GxChassisSetSlotsTemperaturesStates**,  
**GxChassisSetTemperatureThresholdMode**, **GxChassisSetAlarmMode**, **GxChassisSetAlarmTemperature**,  
**GxChassisSetShutdownTemperature**, **GxChassisGetErrorString**

## GxChassisSetTemperatureThresholdMode

---

### Purpose

Sets the Temperature threshold operational mode.

### Syntax

**GxChassisSetTemperatureThresholdMode**(*nHandle*, *nMode*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX72XX Chassis.
<i>pnMode</i>	PSHORT	Temperature threshold operational modes are: 0. GXCHASSIS_OVER_TEMPERATURE_MODE_MAX_SLOT (default) 1. GXCHASSIS_OVER_TEMPERATURE_MODE_AVERAGE_SLOTS
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The temperature threshold operational mode dictates how the alarm threshold and shutdown threshold will be activated. The modes are:

GXCHASSIS\_OVER\_TEMPERATURE\_MODE\_MAX\_SLOT:

- Shutdown activated when any of the enabled slots' temperature is above the shutdown temperature.
- Alarm activated when any of the enabled slots' temperature is above the alarm temperature.

GXCHASSIS\_OVER\_TEMPERATURE\_MODE\_AVERAGE\_SLOTS:

- Shutdown activated when the average temperature of all active slots is above the shutdown temperature.
- Alarm activated when the average temperature of all active slots is above the alarm temperature.

### Example

The following example sets the Temperature threshold operational mode to average:

```
SHORT nStatus;
GxChassisSetTemperatureThresholdMode(nHandle, GXCHASSIS_OVER_TEMPERATURE_MODE_AVERAGE_SLOTS,
&nStatus);
```

### See Also

**GxChassisGetTemperatureThresholdMode, GxChassisSetAlarmMode, GxChassisSetAlarmTemperature, GxChassisSetShutdownTemperature, GxChassisGetErrorString**



## Appendix A – Specifications

### AC Input Power

---

#### GX7200/GX7210/GX7202/GX7212

90VAC to 264 VAC, 12 A max, (PFC)

47 – 63 Hz

### System Power Supply

---

#### GX7200/GX7210/GX7202/GX7212 System Power

Total DC power is 755W.

#### Power Supply Load, Regulation, Ripple, and Noise Specifications

	Output Voltage	Maximum Output Current.	Regulation		Ripple & Noise* Max. mV P-P
			Min.	Max.	
1	+3.3V	60.0* A	- 3 %	+ 5 %	50 mV
2	+5.0V	60.0* A	- 3 %	+ 5 %	50 mV
3	+12.0V	30.0 A	- 3 %	+ 5 %	120 mV
4	-12.0V	5.0 A	- 5 %	+ 5 %	120 mV

\*\*Noise Bandwidth: 10 Hz – 20 MHz

\* Bandwidth: DC – 20 MHz

## Cooling

---

### **GX7200 / GX7210 / GX7202 / GX7212**

Four 79 CFM fans located under the card cage provide cooling for all PXI modules. A separate fan provides cooling for the system power supply. Fan speed control and monitoring is automatic or can be controlled / monitored via the GxChassis software.

## Temperature Monitoring

---

Integrated temperature monitoring via an on-board microcontroller with audible and software notification when preset temperature limits are exceeded.

Temperature Monitoring features:

- Per slot monitoring, 1 reading/sec/slot
- 4 second moving average value
- User selectable alarm criteria: Maximum slot temperature, Average slot temperature accuracy: +/- 2 ° C
- Default warning and shutdown limits: +50 ° C & +70° C
- Warning and shutdown limits programmable via software driver
- Status: Query via software driver and audible alarm for a warning limit condition

## Power Supply Monitoring

---

Monitored voltages: 3.3, 5, +12, -12, VIO value

Accuracy: +/- 2% of reading

## PXI 10 MHz Clock and PXIe 100 MHz Clock

---

Integrated 10 and 100 MHz clock with an auto detect function. The presence of an external 10 MHz PXI clock will synchronize the 100 MHz clock to the external 10 MHz source.

10 MHz clock accuracy: +/- 100 ppm

100 MHz clock accuracy: ± 30 ppm

## Slots

---

The GX72xx has a total of 21 slots:

- 1 System Controller Slot
- 1 PXIe Star Trigger Controller Slot (can be used by any PXIe/cPCIe instrument)
- 8 Hybrid Instruments with (Star Trigger)
- 8 PXI-1/cPCI Instruments (with Star Trigger)
- 1 PXIe/cPCIe Instrument (with Star Trigger)
- 2 PXIe/cPCIe Instrument (without Star Trigger)

## Physical Characteristics

---

Empty Weight:	GX7200: 29 lbs GX7210: 26 lbs GX7202: 35 lbs GX7212: 32 lbs
Dimensions:	GX7200 / GX7210: 4U (7") H, x 17.6" W x 16.5" D GX7202 / GX7212: 6U (10.5") H x 17.6" W x 23" D

## Environmental and Compliance

---

Operating Temperature Range:	0°C to 50°C
Storage Temperature Range:	-20°C to +60°C
Operating relative humidity:	10 to 90%, non-condensing
Storage relative humidity:	5 to 95%, non-condensing
Emissions:	EN 55011:1991 Group 1 Class A at 10 m FCC Class A at 10 m
CE compliance:	EN61010-1 EN61326
PXI compliance	PXI-5, PXI Express Hardware Specification





## Appendix B – PXI Slots, Pin Outs

This appendix describes the P1, P2, XJ1, XJ4, TP2, XP1, XP2, XP3, and XP4 connector pin outs for the GX72xx backplane, and defines the bus signal names. Figure B-1 provides a visual representation of the connector pin outs.

Table B-1 lists the various bus signal names associated with specific bus functionality.

To help in reviewing the tables in this section and locating the appropriate specification for signal requirements, Table B-1 lists all signals alphabetically by original specification (PXI, CompactPCI, or PCI).

- Table B-2 shows the XJ4 connector pin out for the System Controller slot.
- Table B-3 shows the XP3 connector pin out for the System Controller slot.
- Table B-4 shows the XP2 connector pin out for the System Controller slot (4-link configuration).
- Table B-5 shows the XJ1 connector pin out for the System Controller slot.
- Table B-6 shows the XP4 connector pin out for the System Timing slot.
- Table B-7 shows the XP3 connector pin out for the System Timing slot.
- Table B-8 shows the TP2 connector pin out for the System Timing slot.
- Table B-9 shows the XP4 connector pin out for the Hybrid slots.
- Table B-10 shows the XP3 connector pin out for the Hybrid slots.
- Table B-11 shows the P1 connector pin out for the Hybrid slots.
- Table B-12 shows the P1 connector pin out for the PXI-1 Peripheral slots.
- Table B-13 shows the P2 connector pin out for the PXI-1 Peripheral slots.

## Bus Signal Names

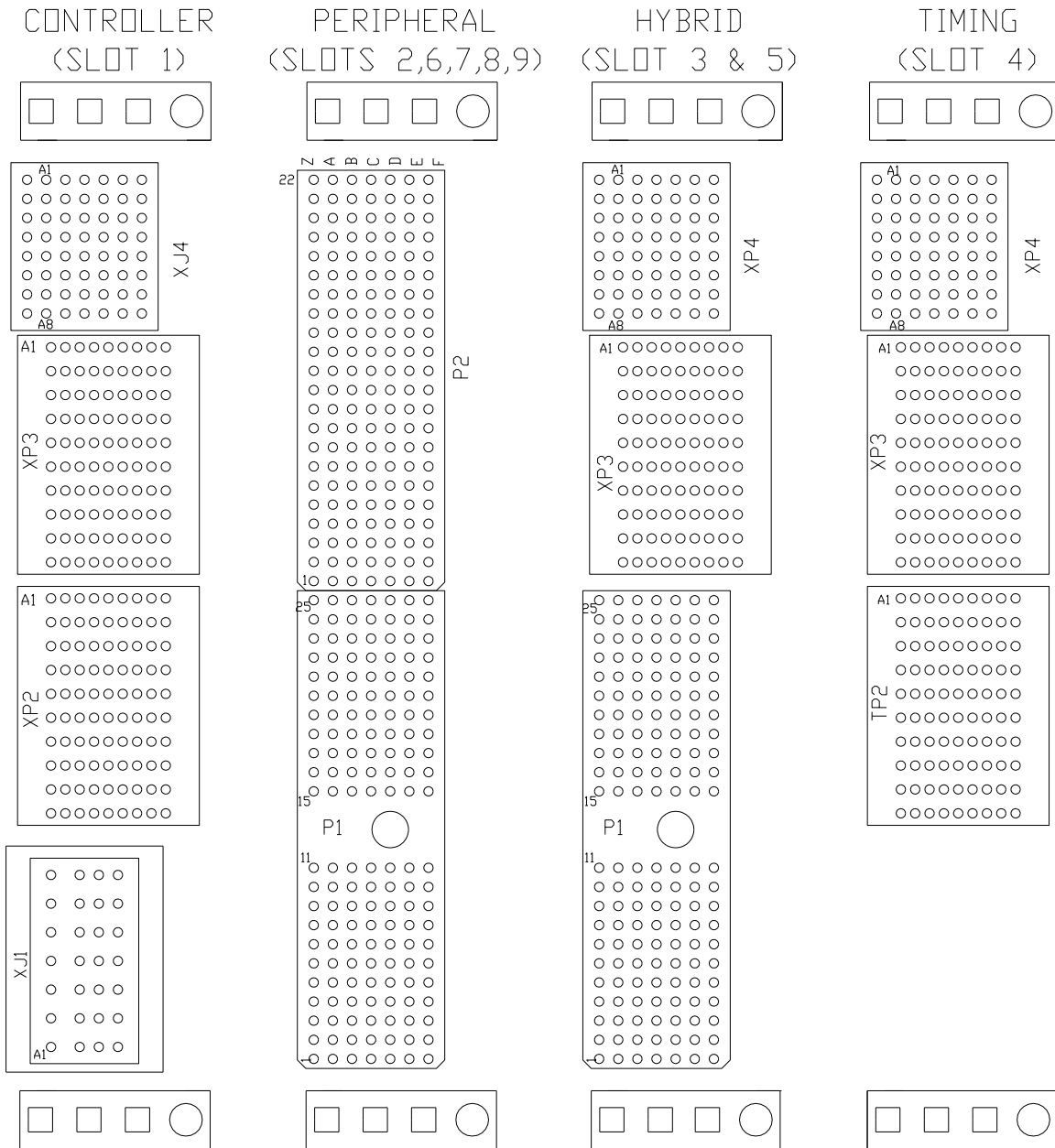
The table below lists all signals alphabetically as defined by the original specification (PXI, CompactPCI, or PCI).

Function	Bus Signal Names		
<b>PXI Express</b>	PXIe_CLK100[p,n][3..1] PXIe_SYNC100[p,n][3..1]	PXIe_DSTAR[C..A][p,n][0,3,5]	PXIe_SYNC_CTRL
<b>CompactPCI Express</b>	SYSEN# PS_ON# ATNLED MPWRGD	ALERT# LINKCAP ATNSW#	PWR_OK PWRBTN# PWREN#
<b>PCI Express</b>	[4..1]PE[T,R][p,n][3..0] SMDAT PRSNT#	[4..1]RefClk[p,n] SMBCLK	PERST# WAKE#
<b>PXI</b>	PXI_BRSV PXI_CLK10 PXI_CLK10_IN	PXI_LBL[0:12] PXI_STAR[0:12] PXI_TRIG[0:7]	PXI_STAR[0:12] PXI_TRIG[0:7]
<b>CompactPCI</b>	BD_SEL# BRSV CLK[0:6] DEG# ENUM# FAL# GA0-GA4 GNT#[0:6]	HEALTHY# INTP INTS IPMB_PWR IPMB_SCL IPMB_SDA PRST#	REQ#[0:6] RSV SYSEN# SMB_ALERT# SMB_SCL SMB_SDA UNC
<b>PCI</b>	ACK64# CLK GND INTA# INTD# M66EN PERR# RST# TCK TMS V(I/O) +12 V	AD[0:63] DEVSEL# GNT# INTB# IRDY# PAR REQ# SERR# TDI TRDY# 3.3 V -12 V	C/BE[0:7]# FRAME# IDSEL INTC# LOCK# PAR64 REQ64# STOP# TDO TRST# 5 V

**Table B-1: Signal Names Grouped By BUS**

## Backplane Connector Layouts by Slot Type

Figure B-1 provides the connector layouts for the backplane slots. This figure is simplified for clarity.



**Figure B-1: Connector Layouts By Slot Type**

## XJ4 Connector Pin Out for System Controller Slot

Pin	Z	A	B	C	D	E	F
1	GND	GA4	GA3	GA2	GA1	GA0	GND
2	GND	5Vaux	GND	SYSEN#	WAKE#	ALERT#	GND
3	GND	LPC_CLK_33M Hz <sup>(1)</sup>	+5V <sup>(1)</sup>	SMB2_CLK/Tx <sup>(1)</sup>	SMB2_DAT/Tx <sup>(1)</sup>	SATA_TX+ <sup>(1)</sup>	GND
4	GND	GND <sup>(1)</sup>	LPC_LFRAME# <sup>(1)</sup>	+3V3 <sup>(1)</sup>	(BOOT) <sup>(1)</sup>	SATA_TX- <sup>(1)</sup>	GND
5	GND	LPC_LAD0 <sup>(1)</sup>	LPC_RST# <sup>(1)</sup>	ETH_TX+ <sup>(1)</sup>	+3V3 <sup>(1)</sup>	SATA_RX+ <sup>(1)</sup>	GND
6	GND	LPC_LAD1 <sup>(1)</sup>	LPC_SMI# <sup>(1)</sup>	ETH_TX- <sup>(1)</sup>	GND <sup>(1)</sup>	SATA_RX- <sup>(1)</sup>	GND
7	GND	LPC_LAD2 <sup>(1)</sup>	LPC_SERIRQ <sup>(1)</sup>	ETH_RX+ <sup>(1)</sup>	GND	USB1+ <sup>(1)</sup>	GND
8	GND	LPC_LAD3 <sup>(1)</sup>	LPC_LDRQ# <sup>(1)</sup>	ETH_RX- <sup>(1)</sup>	+5V <sup>(1)</sup>	USB1- <sup>(1)</sup>	GND

1) These signals can only be used on the rear I/O interface, and are supported in conjunction with the GX7944-RIO module when used with the GX7944.

**Table B-2: XJ4 Connector Pin Out for the System Controller Slot**

## XP3 Connector Pin Out for System Controller Slot (4-Link Configuration)

Pin	A	B	AB	C	D	CD	E	F	EF
1	RSV	RSV	GND	RSV	RSV	GND	RSV	RSV	GND
2	RSV	RSV	GND	PWR_OK	PS_ON#	GND	LINKCAP	PWRBTN#	GND
3	SMBDAT	SMBCLK	GND	4RefClk+	4RefClk-	GND	2RefClk+	2RefClk-	GND
4	RSV	PERST#	GND	3RefClk+	3RefClk-	GND	1RefClk+	1RefClk-	GND
5	1PETp0	1PETn0	GND	1PERp0	1PERn0	GND	1PETp1	1PETn1	GND
6	1PETp2	1PETn2	GND	1PERp2	1PERn2	GND	1PERp1	1PERn1	GND
7	1PETp3	1PETn3	GND	1PERp3	1PERn3	GND	2PETp0	2PETn0	GND
8	2PETp1	2PETn1	GND	2PERp1	2PERn1	GND	2PERp0	2PERn0	GND
9	2PETp2	2PETn2	GND	2PERp2	2PERn2	GND	2PETp3	2PETn3	GND
10	3PETp0	3PETn0	GND	3PERp0	3PERn0	GND	2PERp3	2PERn3	GND

**B-3: XP3 Connector Pin Out for the System Controller Slot**

### XP2 Connector Pin Out for System Controller Slot (4-Link Configuration)

Pin	A	B	AB	C	D	CD	E	F	EF
1	3PETp1	3PETn1	GND	3PERp1	3PERn1	GND	3PETp2	3PETn2	GND
2	3PETp3	3PETn3	GND	3PERp3	3PERn3	GND	3PERp2	3PERn2	GND
3	4PETp0	4PETn0	GND	4PERp0	4PERn0	GND	4PETp1	4PETn1	GND
4	4PETp2	4PETn2	GND	4PERp2	4PERn2	GND	4PERp1	4PERn1	GND
5	4PETp3	4PETn3	GND	4PERp3	4PERn3	GND	RSV	RSV	GND
6	RSV	RSV	GND	RSV	RSV	GND	RSV	RSV	GND
7	RSV	RSV	GND	RSV	RSV	GND	RSV	RSV	GND
8	RSV	RSV	GND	RSV	RSV	GND	RSV	RSV	GND
9	RSV	RSV	GND	RSV	RSV	GND	RSV	RSV	GND
10	RSV	RSV	GND	RSV	RSV	GND	RSV	RSV	GND

Table B-4: XP2 Connector Pin Out for the System Controller Slot

### XJ1 Connector Pin Out for System Controller Slot

Pin	
G	GND
F	+12V
E	+12V
D	GND
C	+5V
B	+3.3V
A	GND

Table B-5: XJ1 Connector Pin Out for the System Controller Slot

### XP4 Connector Pin Out for System Timing Slot

Pin	Z	A	B	C	D	E	F
1	GND	GA4	GA3	GA2	GA1	GA0	GND
2	GND	5Vaux	GND	SYSEN#	WAKE#	ALERT#	GND
3	GND	+12V	+12V	GND	GND	GND	GND
4	GND	GND	GND	+3.3V	+3.3V	+3.3V	GND
5	GND	PXI_TRIG3	PXI_TRIG4	PXI_TRIG5	GND	PXI_TRIG6	GND
6	GND	PXI_TRIG2	GND	ATNLED	PXI_CLK10_IN	PXI_CLK10	GND
7	GND	PXI_TRIG1	PXI_TRIG0	ATNSW#	GND	PXI_TRIG7	GND
8	GND	PXIe_SYNC_CTRL	GND	RSV	PXI_LBL6	PXI_LBR6	GND

Table B-6: XP4 Connector Pin Out for the System Timing Slot

### XP3 Connector Pin Out for System Timing Slot

Pin	A	B	AB	C	D	CD	E	F	EF
1	PXIe_CLK100+	PXIe_Clk100-	GND	PXIe_SYNC100+	PXIe_SYNC100-	GND	PXIe_DSTARC+	PXIe_DSTARC-	GND
2	PRSNT#	PWREN#	GND	PXIe_DSTARB+	PXIe_DSTARB-	GND	PXIe_DSTARA+	PXIe_DSTARA-	GND
3	SMBDAT	SMBCLK	GND	RSV	RSV	GND	RSV	RSV	GND
4	MPWRGD#	PERST#	GND	RSV	RSV	GND	1RefClk+	1RefClk-	GND
5	1PETp0	1PETn0	GND	1PERp0	1PERn0	GND	1PETp1	1PETn1	GND
6	1PETp2	1PETn2	GND	1PERp2	1PERn2	GND	1PERp1	1PERn1	GND
7	1PETp3	1PETn3	GND	1PERp3	1PERn3	GND	1PETp4*	1PETn4*	GND
8	1PETp5*	1PETn5*	GND	1PERp5*	1PERn5*	GND	1PERp4*	1PERn4*	GND
9	1PETp6*	1PETn6*	GND	1PERp6*	1PERn6*	GND	1PETp7*	1PETn7*	GND
10	RSV	RSV	GND	RSV	RSV	GND	1PERp7*	1PERn7*	GND

\*Pins not used in 4-Link Configuration

Table B-7: XP3 Connector Pin Out for the System Timing Slot

## TP2 Connector Pin Out for System Timing Slot

Pin	A	B	AB	C	D	CD	E	F	EF
1	PXIe_DSTA RC0+	PXIe_DST ARC0-	G N D	PXIe_DSTA RC8+*	PXIe_DSTA RC8-	G N D	PXIe_DSTA RB8+*	PXIe_ DST ARB8 -*	GND
2	PXIe_DSTA RA0+	PXIe_DST ARA0-	G N D	PXIe_DSTA RC9+*	PXIe_DSTA RC9-	G N D	PXIe_DSTA RA8+*	PXIe_ DST ARA 8-*	GND
3	PXIe_DSTA RB0+	PXIe_DST ARB0-	G N D	PXIe_DSTA RC1+*	PXIe_DSTA RC1-	G N D	PXIe_DSTA RA9+*	PXIe_ DST ARA 9-*	GND
4	PXIe_DSTA RB1+*	PXIe_DST ARB1-*	G N D	PXI_STAR0 *	PXI_STAR1 *	G N D	PXIe_DSTA RB9+*	PXIe_ DST ARB9 -*	GND
5	PXIe_DSTA RA1+*	PXIe_DST ARA1-*	G N D	PXI_STAR2	PXI_STAR3	G N D	PXIe_DSTA RC10+	PXIe_ DST ARC1 0-	GND
6	PXIe_DSTA RC2+*	PXIe_DST ARC2-*	G N D	PXI_STAR4 *	PXI_STAR5	G N D	PXIe_DSTA RA10+	PXIe_ DST ARA 10-	GND
7	PXIe_DSTA RB2+*	PXIe_DST ARB2-*	G N D	PXI_STAR6	PXI_STAR7	G N D	PXIe_DSTA RB10+	PXIe_ DST ARB1 0-	GND
8	PXIe_DSTA RA2+*	PXIe_DST ARA2-*	G N D	PXI_STAR8	PXI_STAR9	G N D	PXIe_DSTA RC11+	PXIe_ DST ARC1 1-	GND
9	PXIe_DSTA RC3+*	PXIe_DST ARC3-*	G N D	PXI_STAR1 0*	PXI_STAR1 1*	G N D	PXIe_DSTA RA11+	PXIe_ DST ARA 11-	GND
10	PXIe_DSTA RB3+*	PXIe_DST ARB3-*	G N D	PXIe_DSTA RC16+*	PXIe_DSTA RC16-*	G N D	PXIe_DSTA RB11+	PXIe_ DST ARB1 1-	GND

\*Not used in the Gx72xx, see the \*.ini file for trigger routing

### B-8: TP2 Connector Pin Out for the System Timing Slot

## XP4 Connector Pin Out for Hybrid Slots

Pin	Z	A	B	C	D	E	F
1	GND	GA4	GA3	GA2	GA1	GA0	GND
2	GND	5Vaux	GND	SYSEN#	WAKE#	ALERT#	GND
3	GND	+12V	+12V	GND	GND	GND	GND
4	GND	GND	GND	+3.3V	+3.3V	+3.3V	GND
5	GND	PXI_TRIG3	PXI_TRIG4	PXI_TRIG5	GND	PXI_TRIG6	GND
6	GND	PXI_TRIG2	GND	ATNLED	PXI_STAR	PXI_CLK10	GND
7	GND	PXI_TRIG1	PXI_TRIG0	ATNSW#	GND	PXI_TRIG7	GND
8	GND	RSV	GND	RSV	PXI_LBL6	PXI_LBR6	GND

**B-9: XP4 Connector Pin Out for The Hybrid Slots**

## XP3 Connector Pin Out for Hybrid Slots

Pin	A	B	AB	C	D	CD	E	F	EF
1	PXIe_C LK100+	PXIe_Cl k100-	GND	PXIe_S YNC100 +	PXIe_S YNC100 -	GND	PXIe_D STARC +	PXIe_D STARC-	GND
2	PRSENT#	PWREN #	GND	PXIe_D STARB +	PXIe_D STARB-	GND	PXIe_D STARA +	PXIe_D STARA-	GND
3	SMBDATA	SMBCLK	GND	RSV	RSV	GND	RSV	RSV	GND
4	MPWRGD#	PERST#	GND	RSV	RSV	GND	1RefClk +	1RefClk -	GND
5	1PETp0	1PETn0	GND	1PERp0	1PERn0	GND	1PETp1	1PETn1	GND
6	1PETp2	1PETn2	GND	1PERp2	1PERn2	GND	1PERp1	1PERn1	GND
7	1PETp3	1PETn3	GND	1PERp3	1PERn3	GND	1PETp4 *	1PETn4 *	GND
8	1PETp5 *	1PETn5 *	GND	1PERp5 *	1PERn5 *	GND	1PERp4 *	1PERn4 *	GND
9	1PETp6 *	1PETn6 *	GND	1PERp6 *	1PERn6 *	GND	1PETp7 *	1PETn7 *	GND
10	RSV	RSV	GND	RSV	RSV	GND	1PERp7 *	1PERn7 *	GND

\*Pins not used in 4-Link Configuration

**Table B-10: XP3 Connector Pin Out for the Hybrid Slots**



## P1 Connector Pin Out for Hybrid Slots

Pin	Z	A	B	C	D	E	F
25	GND	5V	REQ64#	ENUM#	3.3V	5V	GND
24	GND	AD[1]	5V	V(I/O)	AD[0]	ACK64#	GND
23	GND	3.3V	AD[4]	AD[3]	5V	AD[2]	GND
22	GND	AD[7]	GND	3.3V	AD[6]	AD[5]	GND
21	GND	3.3V	AD[9]	AD[8]	M66EN	C/BE[0]#	GND
20	GND	AD[12]	GND	V(I/O)	AD[11]	AD[10]	GND
19	GND	3.3V	AD[15]	AD[14]	GND	AD[13]	GND
18	GND	SERR#	GND	3.3V	PAR	C/BE[1]#	GND
17	GND	3.3V	IPMB_SCL	IPMB_SDA	GND	PERR#	GND
16	GND	DEVSEL#	GND	V(I/O)	STOP#	LOCK#	GND
15	GND	3.3V	FRAME#	IRDY#	BD_SEL#	TRDY#	GND
12–14	Key	Area					
11	GND	AD[18]	AD[17]	AD[16]	GND	C/BE[2]#	GND
10	GND	AD[21]	GND	3.3V	AD[20]	AD[19]	GND
9	GND	C/BE[3]#	IDSEL	AD[23]	GND	AD[22]	GND
8	GND	AD[26]	GND	V(I/O)	AD[25]	AD[24]	GND
7	GND	AD[30]	AD[29]	AD[28]	GND	AD[27]	GND
6	GND	REQ#	GND	3.3V	CLK	AD[31]	GND
5	GND	BRSVP1A5	BRSVP1B5	RST#	GND	GNT#	GND
4	GND	IPMB_PWR	HEALTHY#	V(I/O)	INTP	INTS	GND
3	GND	INTA#	INTB#	INTC#	5V	INTD#	GND
2	GND	TCK	5V	TMS	TDO	TDI	GND
1	GND	5V	-12V	TRST#	+12V	5V	GND

**Table B-11: P1 Connector Pin Out for the Hybrid Slots**

## P1 Connector Pin Out for PXI-1 Peripheral Slots

Pin	Z	A	B	C	D	E	F
25	GND	5V	REQ64#	ENUM#	3.3V	5V	GND
24	GND	AD[1]	5V	V(I/O)	AD[0]	ACK64#	GND
23	GND	3.3V	AD[4]	AD[3]	5V	AD[2]	GND
22	GND	AD[7]	GND	3.3V	AD[6]	AD[5]	GND
21	GND	3.3V	AD[9]	AD[8]	M66EN	C/BE[0]#	GND
20	GND	AD[12]	GND	V(I/O)	AD[11]	AD[10]	GND
19	GND	3.3V	AD[15]	AD[14]	GND	AD[13]	GND
18	GND	SERR#	GND	3.3V	PAR	C/BE[1]#	GND
17	GND	3.3V	IPMB_SCL	IPMB_SDA	GND	PERR#	GND
16	GND	DEVSEL#	GND	V(I/O)	STOP#	LOCK#	GND
15	GND	3.3V	FRAME#	IRDY#	BD_SEL#	TRDY#	GND
12–14	Key	Area					
11	GND	AD[18]	AD[17]	AD[16]	GND	C/BE[2]#	GND
10	GND	AD[21]	GND	3.3V	AD[20]	AD[19]	GND
9	GND	C/BE[3]#	IDSEL	AD[23]	GND	AD[22]	GND
8	GND	AD[26]	GND	V(I/O)	AD[25]	AD[24]	GND
7	GND	AD[30]	AD[29]	AD[28]	GND	AD[27]	GND
6	GND	REQ#	GND	3.3V	CLK	AD[31]	GND
5	GND	BRSVP1A5	BRSVP1B5	RST#	GND	GNT#	GND
4	GND	IPMB_PWR	HEALTHY#	V(I/O)	INTP	INTS	GND
3	GND	INTA#	INTB#	INTC#	5V	INTD#	GND
2	GND	TCK	5V	TMS	TDO	TDI	GND
1	GND	5V	-12V	TRST#	+12V	5V	GND

**Table B-12: P1 Connector Pin Out for the PXI-1 Peripheral Slots**

## P2 Connector Pin Out for PXI-1 Peripheral Slots

Pin	Z	A	B	C	D	E	F
22	GND	GA4	GA3	GA2	GA1	GA0	GND
21	GND	PXI_LBR0	GND	PXI_LBR1	PXI_LBR2	PXI_LBR3	GND
20	GND	PXI_LBR4	PXI_LBR5	PXI_LBL0	GND	PXI_LBL1	GND
19	GND	PXI_LBL2	GND	PXI_LBL3	PXI_LBL4	PXI_LBL5	GND
18	GND	PXI_TRIG3	PXI_TRIG4	PXI_TRIG5	GND	PXI_TRIG6	GND
17	GND	PXI_TRIG2	GND	RSV	PXI_STAR	PXI_CLK10	GND
16	GND	PXI_TRIG1	PXI_TRIG0	RSV	GND	PXI_TRIG7	GND
15	GND	PXI_BRSVA15	GND	RSV	PXI_LBL6	PXI_LBR6	GND
14	GND	AD[35]	AD[34]	AD[33]	GND	AD[32]	GND
13	GND	AD[38]	GND	V(I/O)	AD[37]	AD[36]	GND
12	GND	AD[42]	AD[41]	AD[40]	GND	AD[39]	GND
11	GND	AD[45]	GND	V(I/O)	AD[44]	AD[43]	GND
10	GND	AD[49]	AD[48]	AD[47]	GND	AD[46]	GND
9	GND	AD[52]	GND	V(I/O)	AD[51]	AD[50]	GND
8	GND	AD[56]	AD[55]	AD[54]	GND	AD[53]	GND
7	GND	AD[59]	GND	V(I/O)	AD[58]	AD[57]	GND
6	GND	AD[63]	AD[62]	AD[61]	GND	AD[60]	GND
5	GND	C/BE[5]#	GND	V(I/O)	C/BE[4]#	PAR64	GND
4	GND	V(I/O)	PXI_BRSVB4	C/BE[7]#	GND	C/BE[6]#	GND
3	GND	PXI_LBR7	GND	PXI_LBR8	PXI_LBR9	PXI_LBR10	GND
2	GND	PXI_LBR11	PXI_LBR12	UNC	PXI_LBL7	PXI_LBL8	GND
1	GND	PXI_LBL9	GND	PXI_LBL10	PXI_LBL11	PXI_LBL12	GND

**Table B-13: P2 Connector Pin Out for the PXI-1 Peripheral Slot**



## Appendix C – Rear Panel Connector Layout

This section provides information on the rear panel connectors for the GX7200 and GX7202.

### Ethernet Connector

---

Connector Type: RJ45

Mating Connector: RJ45, Male

Pin #	Signal Name	Signal Function	Direction
1	TX+	Transmit +	Out
2	TX-	Transmit –	Out
3	RX+	Receive +	In
4	NC	--	--
5	NC	--	--
6	RX-	Receive –	In
7	NC	--	--
8	NC	--	--

**Table C-1: Ethernet Connector Pin out**

## USB Connector

---

Connector Type: USB

Mating Connector: USB

Pin #	Signal Name	Signal Function	Direction
1	VCC	VCC	--
2	UV0-	Differential USB-	In/Out
3	UV0+	Differential USB+	In/Out
4	GND	GND	--

**Table C-2: USB Connector Pin out**

## Appendix D – Model Numbers

### Chassis and Controller Model Numbers

---

The following are the PXI chassis and controller model numbers:

Model #	Description
GX7200	3U, 21 Slot PXIe Chassis with built-in Hard Disk drive.
GX7210	3U, 21 Slot PXIe Chassis for use with a remote controller
GX7200R	3U, 21 slot PXIe chassis with build-in Hard Disk drive with rack mount
GX7210R	3U, 21 slot PXIe chassis with rack mount. For use with PXI bus expander
GX7202	3U. 21 slot PXIe chassis with front panel and cable routing, with built in HDD
GX7212	3U, 21 slot PXIe chassis with front panel and cable routing, For use with PXI bus expander.
GX7944-2162048	CPU Plug-in controller for GX7200. 2.16GHz/2 GB RAM Core2Duo

### Chassis Accessory Model Numbers

---

The following are the PXI chassis accessory model numbers:

Model #	Description
GX97111	Blank Panel for GX7200, 1-slot wide
GX97112	Blank Panel for GX7200, 2-slots wide
GX97114	Blank Panel for GX7200, 4-slots wide





# Index

.	
.NET .....	ii
<b>3</b>	
3U Boards .....	7, 11, 12
<b>6</b>	
6U Boards .....	103
<b>A</b>	
<b>AC Input Power</b> .....	85
Air Intake Panel .....	8
architecture .....	5
Architecture .....	1
ATEasy .....	ii, 4, 5, 35, 36, 39, 40
<b>B</b>	
Back Plane Connector Layouts By Slot Type.....	91
Backplane Connector.....	91
Board Handle.....	41
Borland .....	ii, 35, 39
Borland-Delphi .....	39
Bus Signal Names.....	90
<b>C</b>	
C/C++ .....	35, 36, 39
C++ .....	39
Chassis Accessory Model Numbers .....	103
Chassis Configuration.....	29
Chassis Description .....	1
Chassis Description .....	7
Chassis Description .....	9
Chassis Installation .....	29
Chassis Model Numbers .....	103
Clock .....	86
CompactPCI .....	4
Configuring.....	32
Cooling .....	86
Corrupt files.....	37
cPCI.....	8, 11, 34, 86
<b>D</b>	
Delphi .....	ii, 35, 39
Dimensions, GX7200 .....	87
Directories .....	35
Distributing.....	41
Driver	
Directory.....	35
Files .....	35
Driver Files .....	35
Driver Functions .....	41
<b>E</b>	
Environmental .....	87
Error-Handling .....	41
Ethernet Connector .....	10, 101
Example.....	36
<b>F</b>	
Fan.....	86
Features.....	4
Files .....	35
Front View.....	7
Functions .....	41
Functions Reference .....	49
<b>G</b>	
GPIO .....	34
GX7200 .....	3
GX7200 PXI chassis.....	1
GX7210 .....	3, 6, 103
GX72xx .....	3
GX72xx Models .....	6
GX7900 .....	6, 8, 9, 10, 103
GX7990 .....	4, 6, 8, 11, 30, 103
GxChassis .....	31
Driver-Description.....	39
Header-file .....	39
Help-File-Description .....	36
Panel-File-Description.....	35
Supported-Development-Tools.....	39
GxChassis Driver.....	39
GxChassis driver Features .....	20
GxChassis Functions .....	50
GxChassis Software.....	20
GxChassis.bas.....	35
GxChassis.BAS .....	39
GxChassis.dll.....	35, 39, 40
GxChassis.exe .....	41
GxChassis.h .....	35, 39
GxChassis.lib.....	39
GxChassis.LIB.....	35
GxChassis.pas.....	35, 39
GxChassis.vb .....	35
GxChassisBC.lib .....	39
GxChassisBC.LIB .....	35
GxChassisGetAlarmMode.....	51
GxChassisGetAlarmTemperature .....	52

GxChassisGetBoardSummary .....	53
GxChassisGetDriverSummary .....	41
GxChassisGetDriverSummary .....	54
GxChassisGetErrorString .....	41, 55
GxChassisGetFanSpeed.....	58, 76
GxChassisGetFanThresholdTemperatures .....	59
GxChassisGetPowerSupplies Voltages .....	57
GxChassisGetPxiTriggerLine.....	60, 62
GxChassisGetShutdownTemperature.....	63
GxChassisGetSlotsTemperatures .....	64
GxChassisGetSlotsTemperaturesStates .....	65
GxChassisGetSlotsTemperaturesStatistics .....	66
GxChassisGetSlotTemperature.....	67
GxChassisGetTemperatureScale .....	68
GxChassisGetTemperatureThresholdMode.....	69
GxChassisInitialize .....	41, 70
GxChassisPanel .....	41, 71
GxChassispanel.exe.....	35
GxChassisRecallSettings .....	72
GxChassisResetPxiTriggerLines .....	73
GxChassisSetAlarmMode.....	74
GxChassisSetAlarmTemperature.....	75
GxChassisSetFanThresholdTemperatures .....	77
GxChassisSetPxiTriggerLine .....	78
GxChassisSetShutdownTemperature.....	80
GxChassisSetSlotsTemperaturesStates.....	81
GxChassisSetTemperatureScale .....	82
GxChassisSetTemperatureThresholdMode .....	83

**H**

handle .....	33
Handle .....	41
HW .....	31, 35, 41

**I**

Input Power Receptacle .....	10
Inspecting the GX7200.....	29
Installation .....	31
Chassis .....	29
Mounting information.....	29
PXI Module .....	33
Installation Directories .....	35
Introducing PXI And PCI Express .....	5, 6
Introduction .....	3, 49

**L**

LabView .....	37
Line Voltage Selection .....	29
Local Bus.....	16

**M**

Master .....	30
Model Numbers .....	103
Models .....	6

Monitoring.....	86
-----------------	----

**N**

<i>nHandle</i> .....	40, 41
----------------------	--------

**O**

OnError.....	40
Optional Equipment.....	6
Overview .....	20, 39

**P**

P1 (J1) Connector .....	97, 98
P2 (J2) Connector .....	99
Panel .....	21, 22, 23, 25, 26, 27, 35, 37, 41
Pascal.....	35, 39
PCI.....	35
Physical Dimension .....	87
<i>pnStatus</i> .....	41
Power Distribution.....	19
<b>Power Supplies</b> .....	85
Power Supply.....	19
Power Supply Monitoring.....	86
Power Switch.....	7
Program-File-Descriptions .....	35
Programming .....	39
Borland-Delphi .....	39
Error-Handling .....	41
Panel-Program .....	41
PXI.....	7, 11, 22, 32, 33, 34
PXI Bus Segments .....	12, 13, 15
PXI Express System Timing Slot .....	15
PXI Instrument Removal .....	34
PXI Module Installation .....	33
PXI Slots.....	11
PXI Slots Pinouts.....	89
PXI Standard .....	5
PXI Systems Alliance .....	5
Pxi Trigger Lines .....	25
PXI/PCI Explorer .....	22, 32

**R**

README.TXT .....	35, 36
Readme-File .....	36
Rear Panel.....	101
RS-232.....	6, 34

**S**

Sample Program Listing .....	42
Sample Programs .....	42
Serial.....	4
Setup .....	31, 35, 37
Setup Maintenance Program.....	37
Slave .....	30

Slot.....	8, 33, 86, 103
Slot Number.....	8
Slots .....	11, 86
Star Trigger.....	8
System .....	8, 86
Specifications .....	1, 85
Stand Alone Configuration .....	30
Star Trigger Lines .....	18
System	
Directory.....	35
System Controller Slot.....	15
System Reference Clock.....	19

**T**

Temperature Monitoring.....	86
Temperature Settings .....	23
The PXI Standard .....	5
TP2 (TJ2) Connector .....	95
Trigger Bus.....	17

**U**

Unpacking the GX7200.....	29
---------------------------	----

USB Connector.....	102
USB Connectors .....	10
Using External Instruments .....	34

**V**

Virtual Instrument Software Architecture .....	5
Virtual Panel.....	21, 22, 23, 25, 26, 27, 31, 35
About Page .....	27
Advanced page .....	26
Initialize Dialog .....	22
Virtual Panel Description .....	21
Visual Basic.....	ii, 39
Visual C++ .....	ii, 35, 39
VXI.....	34
VXI plug & play System Alliance .....	5

**X**

XP2 (XJ2) Connector .....	93
XP3 (XJ3) Connector .....	96
XP4 (XJ4) Connector .....	92, 94, 96

