

ARTISAN®
TECHNOLOGY GROUP

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

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

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

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

Custom engineering
so your equipment works exactly as you specify.

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

Expert team | Trust guarantee | 100% satisfaction

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

Visit our website - *Click HERE*

GX7100A/B
GX7110A/B
GX7102A/B
GX7112A/B
GX7111A/B

GX71xx 3U/6U PXI Instrumentation Platform Series

GxChassis Software

User's Guide

Last updated July 9, 2020

MARVIN TEST SOLUTIONS

Safety and Handling

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

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

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

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

Warranty

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

If You Need Help

Visit our web site at https://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: https://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 © 2003-2020 by Marvin Test Systems, 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)
C++ Builder, Delphi	Embarcadero Technologies Inc.
LabView, LabWindows^{tm}/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
Table of Contents iii
Chapter 1 - Introduction 1
Manual Scope and Organization 1
Manual Scope 1
Manual Organization 1
Conventions Used in this Manual 2
Chapter 2 - Overview 3
Introduction 3
Features 4
The PXI Standard 5
Chassis Models 5
Optional Equipment 6
Chassis Description – Front View 7
Gx71xxA Chassis Description – Rear View 10
Gx71xxB Chassis Description – Rear View 11
PXI Slots 12
PXI Bus Segments 12
System Controller Slot 12
Star Trigger Controller Slot 12
Peripheral Slots 13
3U and 6U Boards 13
Local Bus 13
Trigger Bus 14
Star Trigger Lines 15
System Reference Clock 15
Chassis Orientation (Gx71xxB models and above) 15
System Power Supplies 16
Power Distribution 16
Overview of the GxChassis Software 17

GxChassis driver Features....17
Virtual Panel Description....18
Virtual Panel Initialize Dialog....19
Virtual Panel Temperature Settings....19
Virtual Panel PXI Trigger Lines....21
Virtual Panel Advanced page....22
Virtual Panel About Page....23
Chapter 3 - Setup and Installation....25
Unpacking and Inspecting the Chassis....25
Mounting Information....25
Line Voltage Selection and Current Values....25
Chassis Installation....25
GX7100 Master and GX7110 Slave Configurations....26
Installation of the GxChassis Software....27
Configuring Your PXI System using the PXI/PCI Explorer....28
Installing a PXI Instrument....29
PXI Board Software Installation....30
Removing a PXI Instrument....31
Using External Instruments....31
Installation Folders....32
Driver Files Description....32
Driver File and Virtual Panel....32
Interface Files....32
On-line Help and Manual....33
ReadMe File....33
Example Programs....33
Setup Maintenance Program....34
Chapter 4 - Programming the Chassis....35
Overview....35
The GxChassis Driver....35
Programming Using C/C++ Tools....35
Programming Using Visual Basic....35
Programming Using Visual C#....36
Programming Using Pascal/Delphi....36
Programming GxChassis mainframes Using ATEasy®....36
Using the GxChassis driver functions....36
Chassis Handle....36

Error Handling 36
Driver Version 37
Panel 37
Distributing the Driver 37
Sample Programs 37
Sample Program Listing 38
Chapter 5 - Function Reference 45
Introduction 45
GxChassis Functions 46
GxChassisGetAlarmMode 48
GxChassisGetAlarmTemperature 49
GxChassisGetBoardSummary 50
GxChassisGetDriverSummary 51
GxChassisGetErrorString 52
GxChassisGetPowerSuppliesVoltages 54
GxChassisGetFanSpeed 55
GxChassisGetFanThresholdTemperatures 56
GxChassisGetOrientation 57
GxChassisGetPxiTriggerLine 58
GxChassisGetPxiTriggerLineLevels 60
GxChassisGetShutdownTemperature 61
GxChassisGetSlotsTemperatures 62
GxChassisGetSlotsTemperaturesStates 63
GxChassisGetSlotsTemperaturesStatistics 64
GxChassisGetSlotTemperature 65
GxChassisGetTemperatureScale 66
GxChassisGetTemperatureThresholdMode 67
GxChassisInitialize 68
GxChassisPanel 69
GxChassisRecallSettings 70
GxChassisResetPxiTriggerLines 71
GxChassisSetAlarmMode 72
GxChassisSetAlarmTemperature 73
GxChassisSetFanSpeed 74
GxChassisSetFanThresholdTemperatures 75
GxChassisSetPxiTriggerLine 76
GxChassisSetShutdownTemperature 78

GxChassisSetSlotsTemperaturesStates 79
GxChassisSetTemperatureScale 80
GxChassisSetTemperatureThresholdMode 81
Appendix A – Specifications 83
AC Input Power 83
GX7100/GX7110/GX7102/GX7112/GX7111 83
Power Supply 83
GX7100/GX7102/GX7111/GX7110/GX7112 83
560W Power Supply Load, Regulation, Ripple, and Noise Specifications 83
Cooling 83
B Models or Newer 84
Temperature Monitoring 84
Power Supply Monitoring 84
PXI 10 MHz Clock 84
A Series 84
B Models or Newer 84
Slots 85
Physical Dimensions 85
Environmental 85
Appendix B –PXI Slots Pin Outs 87
P1 Connector Pin Out for System Controller Slot 88
P2 Connector Pin Out for System Controller Slot 89
P1 Connector Pin Out for the Star Trigger Slot 90
P2 Connector Pin Out for the Star Trigger Slot 91
P1 Connector Pin Out for the Peripheral Slot 92
P2 Connector Pin Out for the Peripheral Slot 93
Appendix C – Rear Panel Connector Layout 95
Serial Port Connector 95
Keyboard Connector 95
Mouse Connector 96
Ethernet Connector 96
USB Connector 96
Optional SCSI Connector 97
Parallel Port Connector 98
Appendix D – Model Numbers 99
Chassis Model Numbers 99
Chassis Accessory Model Numbers 100

Index 101

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 GX7100, GX7102, GX7110, GX7112, and GX7111 PXI chassis (B series and the older model A series). This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and some understanding of digital to analog conversion. Throughout this manual, the designation GX71xx may be used when describing features that are common to all chassis versions.

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 manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX71xx PXI chassis manual. Lists all the supported versions and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX71xx PXI 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 GX71xx'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	Provides the GX71xx specifications.
Appendix B – PXI Slots Pin Outs	Describes the P1 and P2 connector pin outs for the GX71xx backplane.
Appendix C – Rear Panel Connector Layout	Provides information on the rear panel connectors of the GX710x.
Appendix D – Model Numbers	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.
Bold type	Words or characters you type as the manual instructs, programming function names and window control names.
Italic type	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 GX71xx PXI instrumentation chassis family. This state-of-the-art chassis is designed for test, data acquisition, process control, and factory automation applications. The GX71xx is a 4U, 19" PXI chassis available in both desktop and rack-mount configurations. The GX71xx00 is based on the CompactPCI™ (cPCI) and PXI™ (PCI eXtentions for Instrumentation) standards. The chassis accommodates seven 3U PXI or CompactPCI instruments and seven additional 3U or 6U PXI or cPCI instruments for a total of 14 instruments. When a 6U CPU module is installed the maximum instruments that can be installed is 13 (7 3U and 6 6U/3U). The design of the GX71xx allows integration of PXI and cPCI boards from any vendor. The Gx71xxB has the capability to get the chassis' orientation in terms of pitch and roll (see "Chassis Orientation" section for details).

The GX71xx PXI instrumentation chassis comes with a GxChassis driver that 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 driver provides complete API calls for controlling all of the PXI chassis capabilities as well as providing a soft front panel that supports these same capabilities. This driver is also included with the Marvin Test Solutions Product CD that is supplied with every chassis.

The GxChassis driver supports the programming of shutdown and alarm temperature limits, measures slot temperatures, measures the system power supply voltages. In series B and above the driver also controls the PXI trigger lines' directions and states, as well as monitors and controls 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.



Figure 2-1: GX7100A Instrumentation Chassis



Figure 2-2: GX7100B Instrumentation Chassis

Features

The GX71xx models offer the following features:

- Controller module, which includes CPU module, a CD/DVD-RW, hard-disk drives and Floppy drive and (Floppy drive not available in B model or newer).
- 14 PXI/cPCI, 3U or 6U slots. Slots 1 to 7 on the left are 6U and 8-14 on the right are 3U. Slot 1 is dedicated for an embedded controller or for a GX7990 receiver (GX7110). Slot 2 can be used by a PXI Star Trigger Controller or by any PXI/cPCI instrument. Slots 3 through 14 supports the PXI Star Trigger.
- Full compliance with PXI specifications revision 2.2. Supports features such as trigger bus, star trigger, local bus, and system clock.
- Interoperability with 32-bit 33MHz CompactPCI.
- Front-loading design. Boards are inserted from the front for simplified maintenance.
- Board user interface connectors face the front side of the platform enabling easy access to board connectors and cables and a short path to the interface.
- Two, four- fan assemblies located on the left and center wall assemblies of the chassis provide positive airflow for all instruments.
- Programmatically control the fans' speed (B models and newer).
- Programmatically control the PXI trigger lines direction and states per trigger line settings (B models and newer).
- Programmatically retrieve the chassis' pitch and roll in degrees in order to determine the cassis orientation (B models and newer).
- Separate fan provides cooling for the chassis power supplies.
- 560W power supply offers wide range input (100 –250 VAC) with power factor correction (PFC).
- Backplane incorporates a local bus, trigger bus, and a 10MHz reference clock.
- Support for external instrumentation and devices via the built-in serial (RS-232, RS-422 or RS-485), parallel, USB and Ethernet interfaces. Support for additional interfaces such as IEEE-488 (GPIB) or VXI is available.
- Additional chassis may be daisy-chained using the PXI to PXI bus expander (GX7990-1 or MXI-4).
- Innovative PXI-Explorer ™ software provides easy configuration tools for the chassis and instruments.
- The Gx71xxB has the capability to get the chassis' orientation in terms of pitch and roll (see "Chassis Orientation" section for details).

When bundled with ***ATEasy***™, Marvin Test Solution's award-winning test executive and software development environment, the GX71xx provides a complete system for creating any test and measurement application.

The PXI Standard

The PXI standard was developed in response to the needs of test system 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 brings together the right technologies for PC-based test and measurement, instrumentation, and industrial automation. Further, 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. Not only is PXI fully compatible with existing operating systems and software, it further implements the Virtual Instrument Software Architecture (VISA) standard that was created by the VXIplug&play System Alliance (see http://www.ivifoundation.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 resulting in PXI users receiving all the benefits of PCI and cPCI within an architecture that also supports mechanical, electrical and software features. These features are typically focused on test & measurement, data acquisition, industrial instrumentation and factory automation applications.

The PXI standard is defined 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. Compared to PC-based products, PXI products are subjected to higher and more carefully defined levels of environmental performance – a necessity in today's industrial environments.

Chassis Models

The chassis is available in several master (with single board computer) and slave configurations, providing maximum flexibility for a variety of applications. The following models are available:

- GX7100: This innovative chassis includes a CD/DVD-RW, floppy disk and hard disk drives. The GX7100 is designed to operate with the GX79xx family of embedded controllers. Internal circuits connect the embedded controllers to the peripheral's devices and many of the controller's interfaces (i.e. USB, RS-232, Ethernet, VGA, etc.) are routed to the rear-panel of the GX7100, minimizing the number of connections required at the front of the chassis. See Figure 2-2.

  NOTE: Your GX79xx controller is provided with documentation that describes its available connections and configuration separately.

- GX7110: This chassis is designed to operate with the GX799x PXI bus expanders or with a MXI-4 interface. This configuration allows the use of desktop PC or another PXI chassis as the system controller. See Figure 2-3.
- GX7102: This chassis is similar to the GX7100 but is a rack mountable unit that also incorporates an integrated cable tray and hinged front panel for mass interconnect applications. See Figure 2-4.
- GX7112: This chassis is similar to the GX7110 but is a rack mountable chassis that also incorporates an integrated cable tray and hinged front panel for mass interconnect applications. See Figure 2-5.
- GX7111: This chassis is similar to the GX7110 but offers a two-inch recessed card cage, facilitating cable management and interfacing to instrument front panel connections. See Figure 2-6.

All models are available in a desktop or rack-mount configurations. For a rack-mount configuration, add the GX7100 rack mount kit (P/N 97000).

Optional Equipment

Marvin Test Solutions offers a variety of products to use with the GX71xx chassis family:

- Embedded Controllers
- Remote Controllers
- 3U & 6U PXI instruments
- Rack mount kits
- Blank panels
- 3U to 6U panel adapters allowing a 3U instrument to fit into a 6U chassis

For part numbers, refer to Appendix B or contact Marvin Test Solutions customer support.

Chassis Description – Front View

The following figures show the front view in master (GX7100A) and slave (GX7110A) configuration:

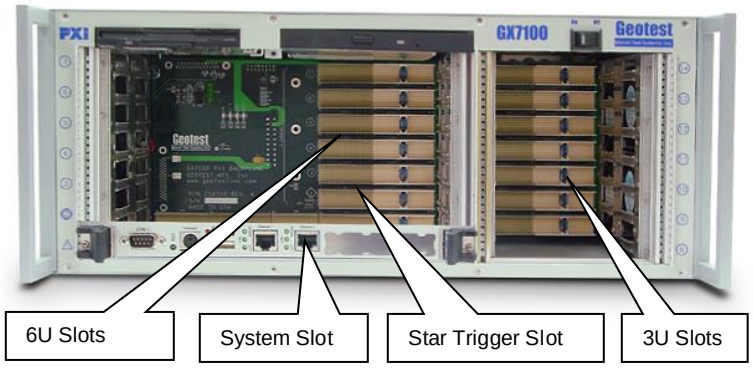


Figure 2-3: GX7100A Front View

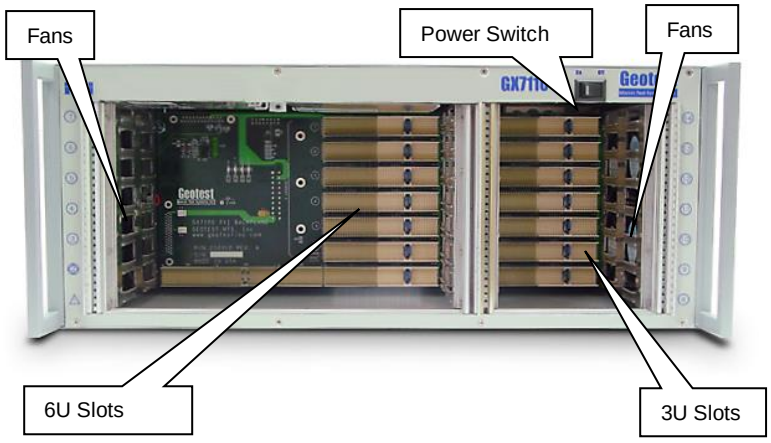


Figure 2-4: GX7110A Front View

The chassis' main features include:

Power Switch

On/Off switch with a power On LED.

System Slot (Slot #1)

The System Slot is the bottom left bottom slot and is used for embedded or remote controllers (GX7910 or GX7990). The system slot can accept any embedded controller that is 1slot wide although only the GX7910 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.

CD/DVD-RW

A CD/DVD-RW optical drive, capable of reading and writing CDs or DVDs. Connections are internal (built-in) when used in conjunction with the GX7910 family of embedded controllers.

Floppy Disk Drive

A 1.44MB Floppy Disk Drive. Connections are internal (built-in) when used in conjunction with the GX7910 family of embedded controllers. Floppy drive is not available in B or newer models.

3 and 6 U Slots

7 6U slots and 7 3U PXI slots are available for PXI instruments

System Slot

Used for the CPU module in the GX7100 or the PXI bus expender in the GX7110

The following figure shows the GX102 chassis (master configuration) with the integrated cable tray and hinged front panel:



Figure 2-5: GX7102A

The following figure shows the GX7112 chassis (slave configuration) with the integrated cable tray and hinged front panel:



Figure 2-6: GX7112A

The following figure shows the GX111 chassis (slave configuration) with the two-inch recessed card cage:



Figure 2-7: GX7111A

Gx71xxA Chassis Description – Rear View

When used in conjunction with the GX7910 embedded controllers, many of the controller's peripheral I/O are available through the rear panel. Figure 2-8 shows the rear view of the GX7100A followed by Figure 2-9 that shows the rear view of the GX7110A.

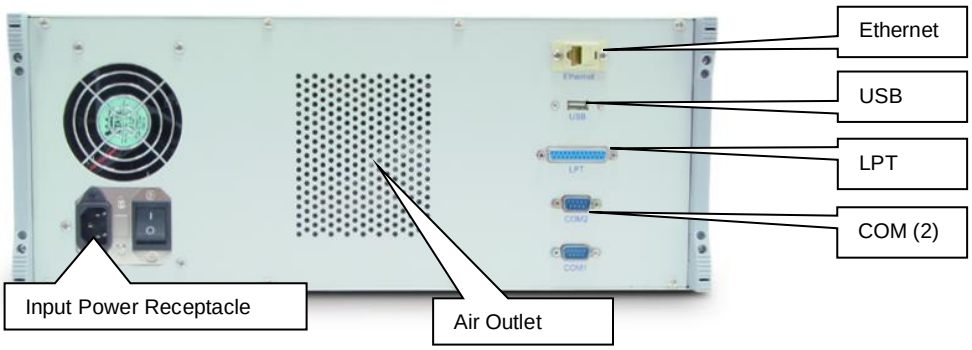


Figure 2-8: GX7100A Rear

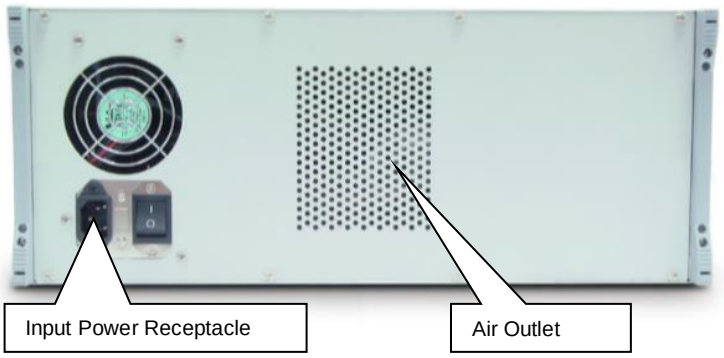


Figure 2-9: GX7110A Rear View

Several connections are available only on the GX7100A and only if used with the GX79xx Embedded Controller. The chassis back side main features are:

- Input Power Receptacle. This receptacle connects to the power cord provided.
- LPT - Parallel (Printer) Connector.
- COM - Serial COM port Connectors. Two Serial Ports are available: These ports may be routed to the rear or front panel and are factory preset to RS-232 mode. By default, COM1 is available at the front panel and COM2 at the rear panel. COM1 may be routed to the rear panel by using the CMOS setup. Furthermore, COM1 when routed to the back can be configured as RS-422 or RS-485. Contact Marvin Test Solutions for more details regarding this configuration.
- USB Connector
- 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 panel.
- Keyboard Connector. A PS2 keyboard port.

Gx71xxB Chassis Description – Rear View

When used in conjunction with the GX7910 embedded controllers, many of the controller's peripheral I/O are available through the rear panel. Figure 2-10 shows the rear view of the GX7100B followed by Figure 2-11 that shows the rear view of the GX7110B.

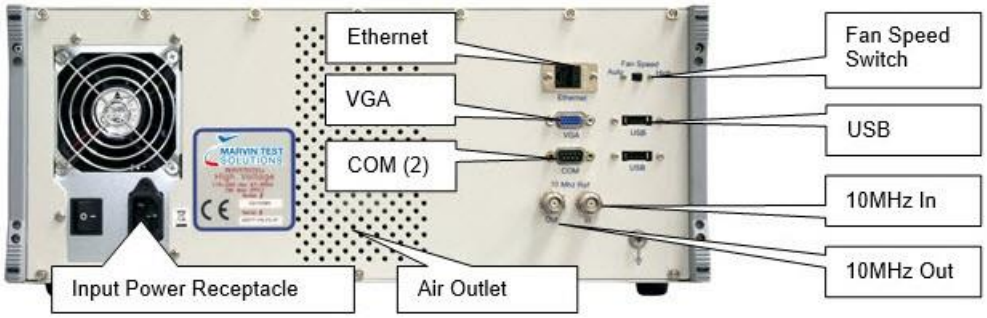


Figure 2-10: GX7100B Rear

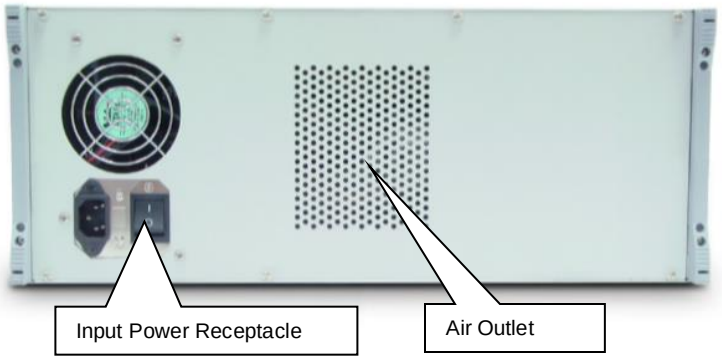


Figure 2-11: GX7110B Rear View

Several connections are available only on the GX7100B and only if used with the GX79xx Embedded Controller. The chassis back side main features are:

Input Power Receptacle. This receptacle connects to the power cord provided.

- COM - Serial COM port Connectors. Two Serial Ports are available: These ports may be routed to the rear or front panel and are factory preset to RS-232 mode. By default, COM1 is available at the front panel and COM2 at the rear panel. COM1 may be routed to the rear panel by using the CMOS setup. Furthermore, COM1 when routed to the back can be configured as RS-422 or RS-485. Contact Marvin Test Solutions for more details regarding this configuration.
- USB Connector
- 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 panel.
- Keyboard Connector. A PS2 keyboard port.

PXI Slots

The GX71xx contains 7 6U slots numbered 1 to 7 and additional 7 3U slots numbered 8 to 14 as shown in Figure 2-12. Slot 1 is dedicated for an embedded controller or for a GX7990 receiver. Slot 2 can be used by a PXI Star Trigger Controller or by a PXI/cPCI instrument. Slots 3 through 14 supports the PXI Star Trigger.

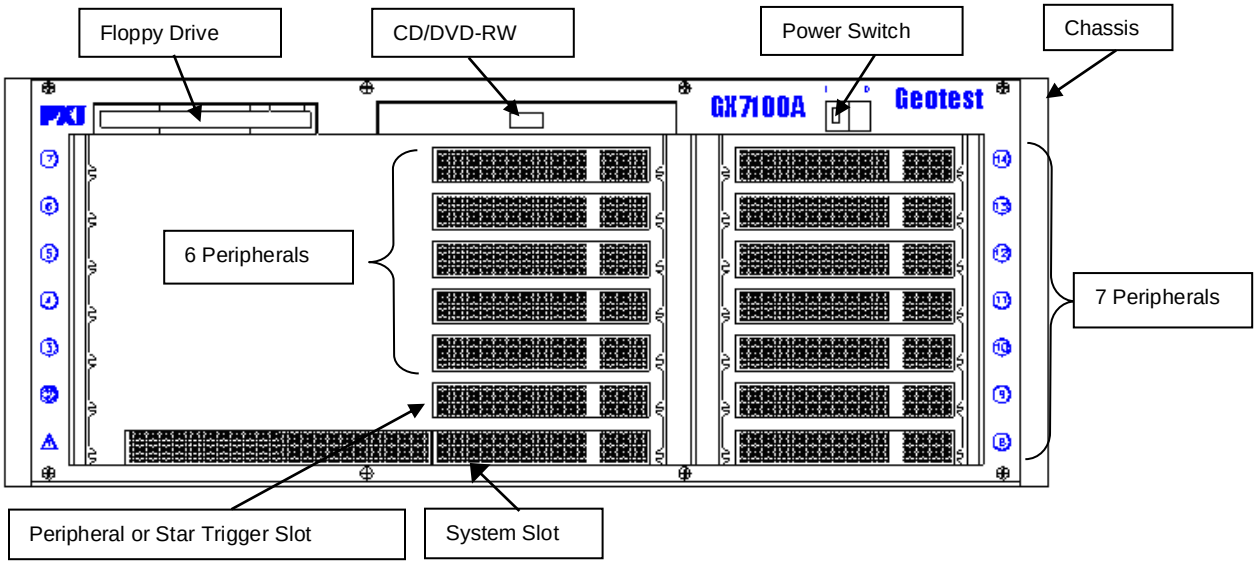


Figure 2-12: GX7100A/GX7110A Slots

PXI Bus Segments

The GX71xx slots are divided between two bus segments, which are connected using PCI-PCI bridge technology. The bridge device takes up one PCI load on each of the bus segments that it links together. The left bus segment supports the System Slot, the Star Trigger Slot and 5 more peripheral slots (Slot 3 to 7). The second segment supports slots 8 to 14.

System Controller Slot

The System Controller slot is located in slot #1 of the chassis. Slot numbers are clearly labeled on both sides of the chassis. Slot #1 at the bottom left through slot #7 at the top left, slot #8 at the bottom right through slot #14 on the top right. The GX71xx can accept either 3U or 6U CPU module that is 1 slot wide (the GX7910 embedded controller family is 6U).

Star Trigger Controller Slot

Slot 2 is the Star Trigger (ST) Controller slot (2nd from the bottom left). This slot has a dedicated trigger line going to slots 3 through 14. The Star Trigger is used to synchronize between 12 instruments and it utilizes back plane traces that are of equal length, providing for a skew of less than 1nSec between slots. If you do not need a Star Trigger Controller, any PXI or cPCI instrument can be used in this slot. See Figure 2-15 in this chapter for more information about the available trigger architectures.

Peripheral Slots

The GX71xx has 12 peripheral slots located in slots 3 through 14 as shown in Figure 2-15. Slots 3 through 14 supports the Star Trigger.

3U and 6U Boards

The GX71xx chassis family supports both form factors of PXI instruments: 3U (100 by 160 mm, or 3.94 by 6.3 in.) and 6U (233.35 by 160 mm, or 9.19 by 6.3 in.). These two form factors are shown in Figure 2-13:

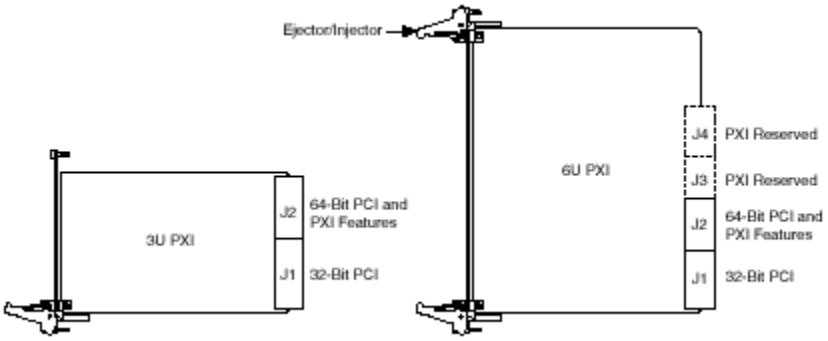


Figure 2-13: 3U and 6U PXI boards

The PXI board has two rear connectors J1 and J2. J1 supports the PCI signals, and J2, supports the PXI signals. PXI signals include the local bus, star trigger signals and trigger bus signals. They are described later in this chapter.

The GX71xx backplane includes the *interface connectors* (P1 and P2) which provide the interconnection between the controller and peripheral modules.

Local Bus

The PXI local bus is a daisy-chained bus connecting peripheral slots in the same bus segment. Each local bus has 13 user-defined lines which are available to connect to each adjutant slot. The lines can be used to pass analog or digital signals between modules or to provide a high-speed side-band digital communication path that does not affect the PXI bandwidth.

Local bus signals can support voltages from 0 to 42V DC and up to 200 mA DC current into any local bus line.

The local bus lines for the leftmost peripheral slot of a PXI back plane (slot 2) are used for the star trigger. Figure 2-14: shows a complete PXI system demonstrating the local buses.

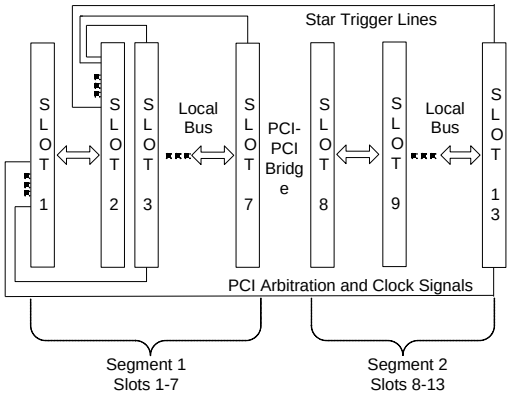


Figure 2-14: PXI Local Bus Routing

Trigger Bus

The eight PXI bus trigger lines can be used in a variety of ways. For example, triggers can be used to synchronize the operation of several different PXI peripheral modules. In other applications, one module can control carefully timed sequences of operations with other modules in 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 provides connectivity only within a single bus segment and does not allow physical connection to an adjacent bus segment. This maintains the high-performance characteristics of the trigger bus and allows the GX71xx to partition instruments into logical groups as show in Figure 2-15. However, logical connections of the trigger bus are allowed. Under software control, the GX71xx's trigger lines can be isolated or interconnected to each segment's eight trigger lines. Signal direction between segments is also controlled by software for each trigger line.

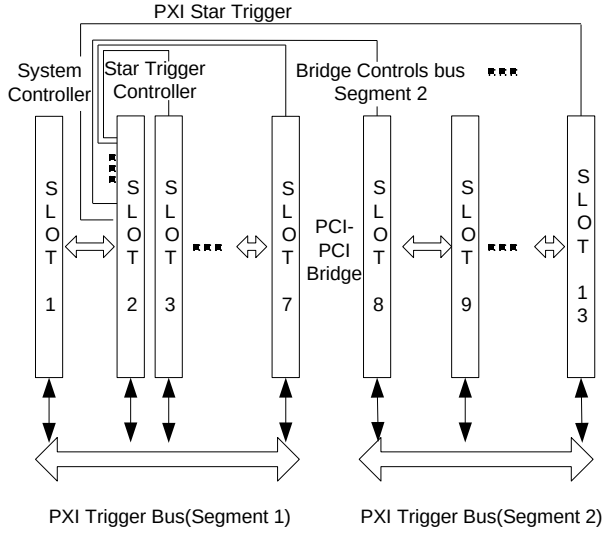


Figure 2-15: PXI Trigger Architecture

Star Trigger Lines

Twelve PXI star trigger lines are connected between slot 2 and each of the slots 3 to 14. The PXI star trigger lines can be used to synchronize the operation of several different PXI peripheral modules.

The PXI star trigger bus offers ultra-high-performance synchronization features to users of PXI systems. The star trigger bus implements a dedicated trigger line between the first peripheral slot (adjacent to the system slot) and the other peripheral slots. A star trigger controller 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 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 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 resulting in very precise trigger relationships between each module.

System Reference Clock

The PXI 10 MHz system clock (PXI_CLK10) is distributed to all slots of the GX71xx. The signal is distributed to each slot using equal length traces and a low jitter clock driver. This common reference clock can be used for synchronization of multiple instruments in measurement or control systems. The star trigger controller in slot two can supply an external 10MHz reference clock. When the GX71xx detects the presence of an external reference clock it automatically switches from the internal reference source to the external source.

Chassis Orientation (Gx71xxB models and above)

The Gx71xxB models and above have the capability to retrieve the chassis' orientation through the driver. The returned chassis' orientation gives the pitch and roll values of the Chassis in degrees. The Pitch and roll angles are referenced to the local horizontal plane, which is perpendicular to the Earth's gravity.

Pitch: defined as the angle between the front of the chassis and the horizontal plane. If the front of the chassis is displaced upwards creating an angel between the front of the chassis and the horizontal plane then it is positive pitch. If the front of the chassis is displaced downwards creating a negative angel between the front of the chassis and the horizontal plane then it is positive pitch. See Figure 2-16:

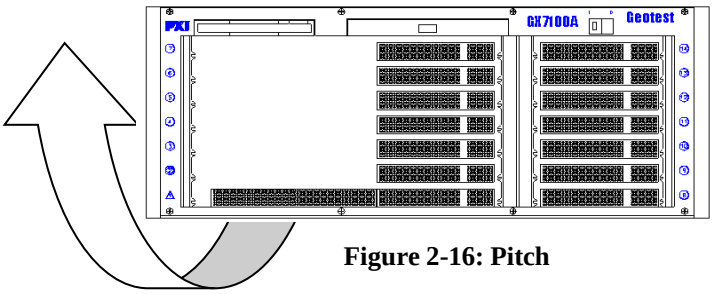


Figure 2-16: Pitch

Roll: defined as the rotation of the chassis. If the chassis is rotating clockwise then it is positive roll. If the chassis is rotating counterclockwise then it is positive roll. See Figure 2-17:

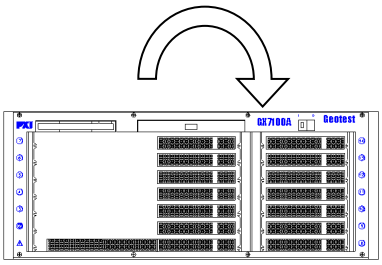


Figure 2-17: Roll

System Power Supplies

One power supply provides operating power to all slots of the GX71xx chassis. 560 watts of system power is available.

Power Distribution

The GX7100/GX7110 meets or exceeds the requirements of the PXI specifications regarding the power provided to each slot. The table below lists the power per slot required by the PXI specifications:

Slot \ Voltage	5V	3.3V	+12V	-12V
System Slot	4A	6A	0.5A	0.1A
Instrument Slot	2A	0A	0.5A	0.1A
Total for a 14-Slot Chassis	22A	6A	10A	2A

Table 2-1: Power per slot required by the PXI standard

The power provided by the GX71xx is listed in the Table 1-3 below:

Slot \ Voltage	5V	3.3V	+12V	-12V
System Slot	8A	6A	2A	0.1A
Instrument Slot	3.2A	1.5A	1A	0.1A
Total for GX7100/GX7110	60A	40A	32A	3A

Table 2-2: Power provided by the GX71xx

Note: Total 5V and 3.3V power cannot exceed 300 watts. Total power cannot exceed 560 watts.

Overview of the GxChassis Software

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

- **GxChassis Panel** – Configures and controls the various features via an interactive user interface.
- **GxChassis driver** - A DLL based function library (GxChassis.DLL (32 or 64-bit) located in the Windows 32/64-bit System folders)) used to program and control the board.
- **Programming files and examples** – Interface files and libraries for support of various programming tools such as C#, C++, VB, VB6, and LabVIEW. A complete list of files and development tools supported by the driver is included in subsequent sections of this manual.
- **Documentation** – On-Line help and User's Guide for the supported chassis, panel application, and driver programming.
- HW driver and PXI/PCI Explorer applet – HW driver allows the GxChassis to access and program the supported chassis and boards. The explorer applet configures the PXI chassis, controllers and devices. This is required for accurate identification of your PXI instruments later on when installed in your system. The applet configuration is saved to PXISYS.ini and PXIeSYS.ini and is used by Marvin Test Solutions instruments HW driver and VISA. The applet can be used to assign chassis numbers, Legacy Slot numbers and instrument alias names. The HW driver is installed and shared with all Marvin Test Solutions products to support accessing the PC resources. Similar to HW driver, VISA provides a standard way for instrument manufacturers and users to write and use instruments drivers. VISA is a standard maintained by the VXI Plug & Play System Alliance and the PXI Systems Alliance organizations (http://www.ivifoundation.org, http://www.pxisa.org/). The VISA resource manager such as National Instruments Measurement & Automation (NI-MAX) displays and configures instruments and their address (similar to Marvin Test Solutions' PXI/PCI Explorer). The driver can work with either HW or VISA to control an access the supported boards.

GxChassis driver Features

The GxChassis software 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.
- 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 here:

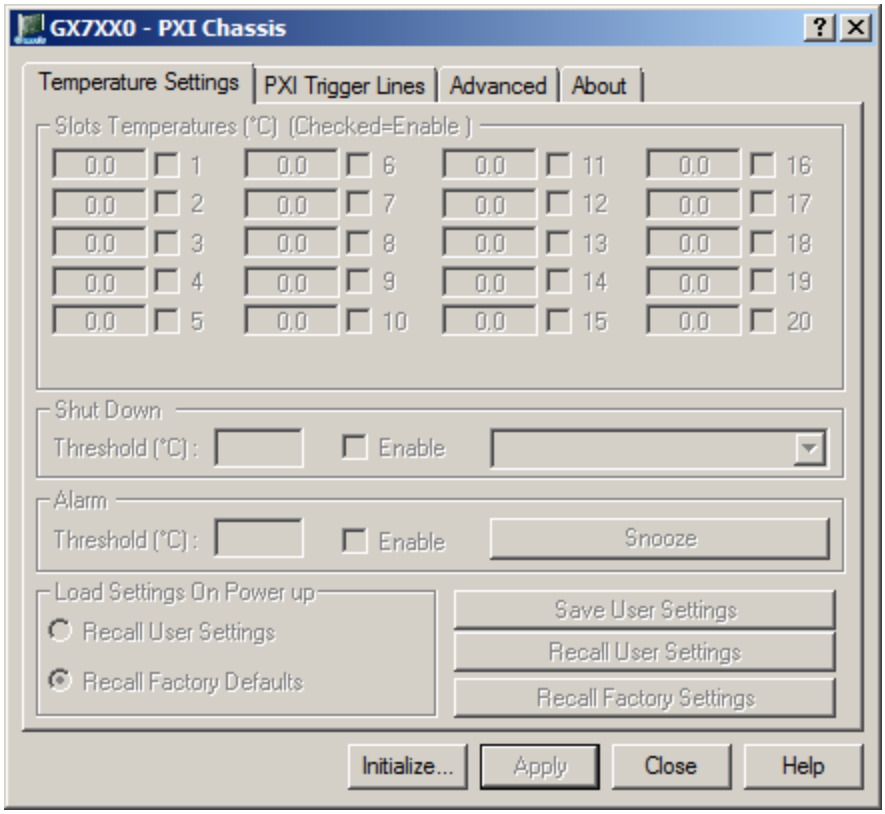


Figure 2-18: GxChassis Virtual Panel – Temperature Settings (not Initialized)

Virtual Panel Initialize Dialog

The Initialize Dialog initializes the card while the settings of the chosen board **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 2-19: Initialize Dialog Box using Marvin Test Solutions' HW driver

Virtual Panel Temperature Settings

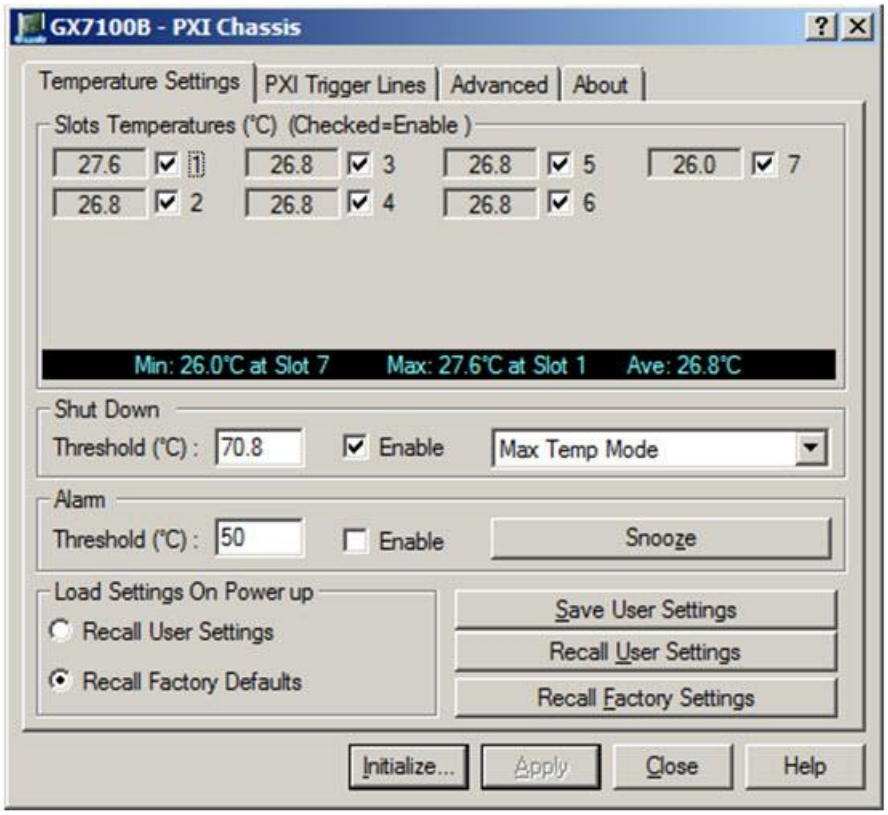


Figure 2-20: GxChassis Virtual Panel – Temperature Settings

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 slot's 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 2-21:

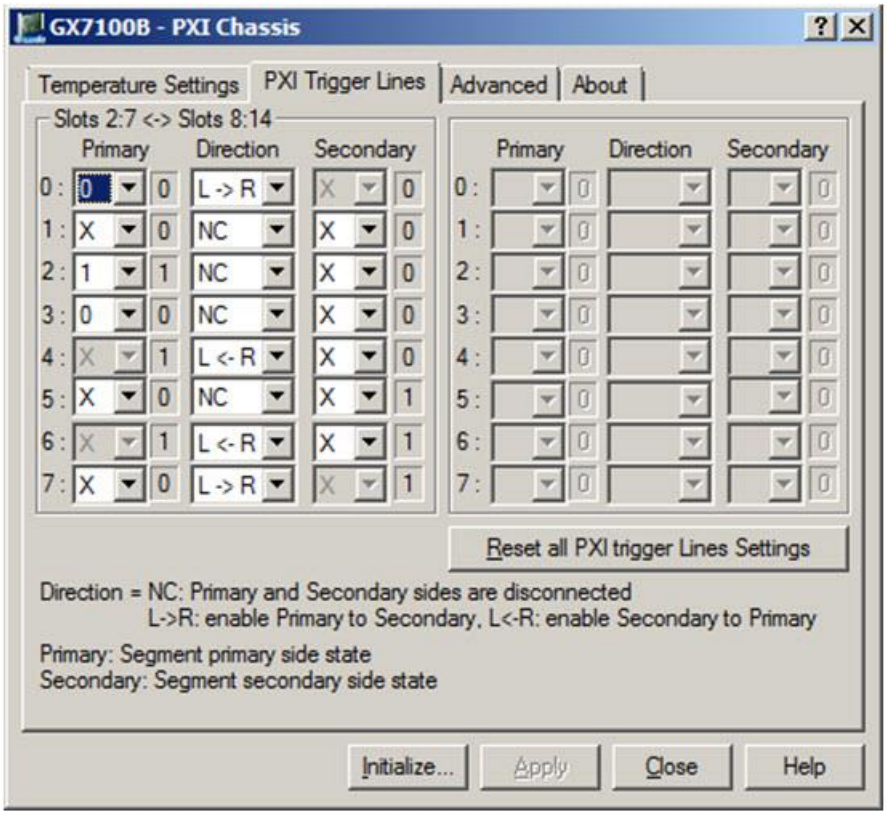


Figure 2-21: GxChassis Virtual Panel – PXI Trigger Lines

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

Slots 1:7 <-> Slots 8:14(Group Box)

Gx71xxA only: Displays PXI trigger lines states and sets/displays directions between slots 1:7 and slots 8:14.

Gx71xxB models and above: Sets/Displays PXI trigger lines states and directions between slots 1:7 and slots 8:14.

Virtual Panel Advanced page

Clicking on the **Advanced** tab will show the **Advanced page** as shown in Figure 2-22:

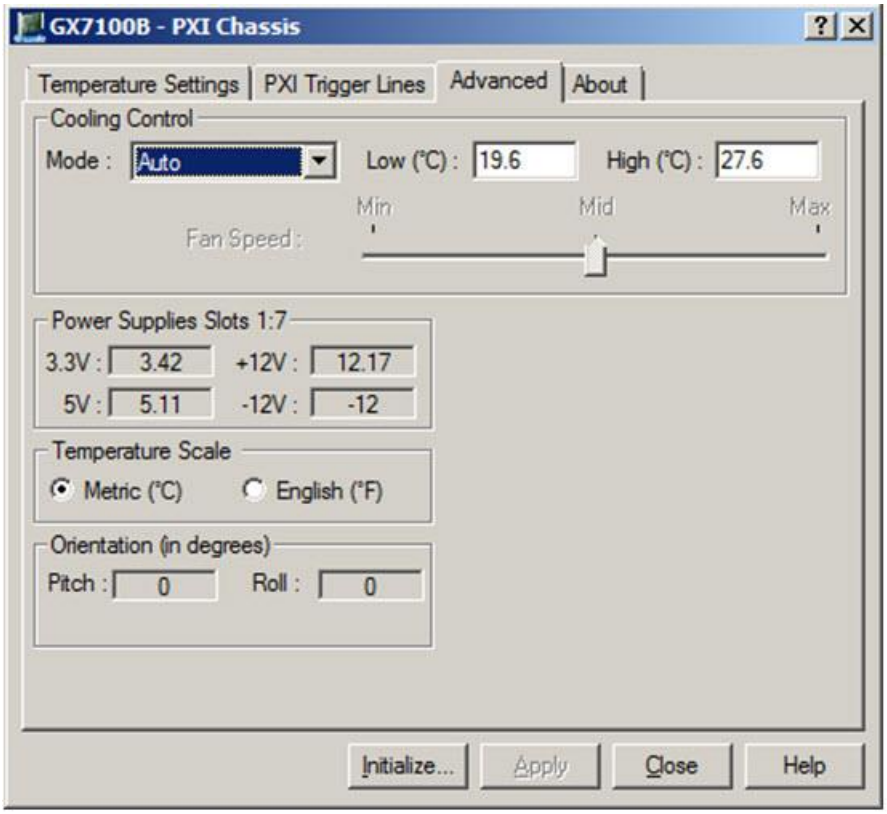


Figure 2-22: GxChassis Virtual Panel – Advanced page

The following controls are shown in the Advanced page:

Cooling Control (Group Box)

Gx71xxB models and above: controls the cooling fans operation.

Low (edit box): Sets/displays the low temperature range for the Auto mode, enabled only in Auto mode.

High (edit box): Sets/displays the high temperature range for the Auto mode, enabled only in Auto mode.

Fan Speed slider: set/displays the fan speed, one of three states: min, mid and max. The control is enabled whenever the Mode is set to User.

Power Supplies Slots 1:14 (Group Box)

Displays the measured +3.3V, +5V, +12V and -12V backplane voltages.

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.

Orientation (Group Box)

Gx71xxB models and above: Displays the chassis' pitch and roll in degrees in order to determine the cassis orientation

Virtual Panel About Page

Clicking on the **About** tab will show the **About page** as shown in Figure 2-23:

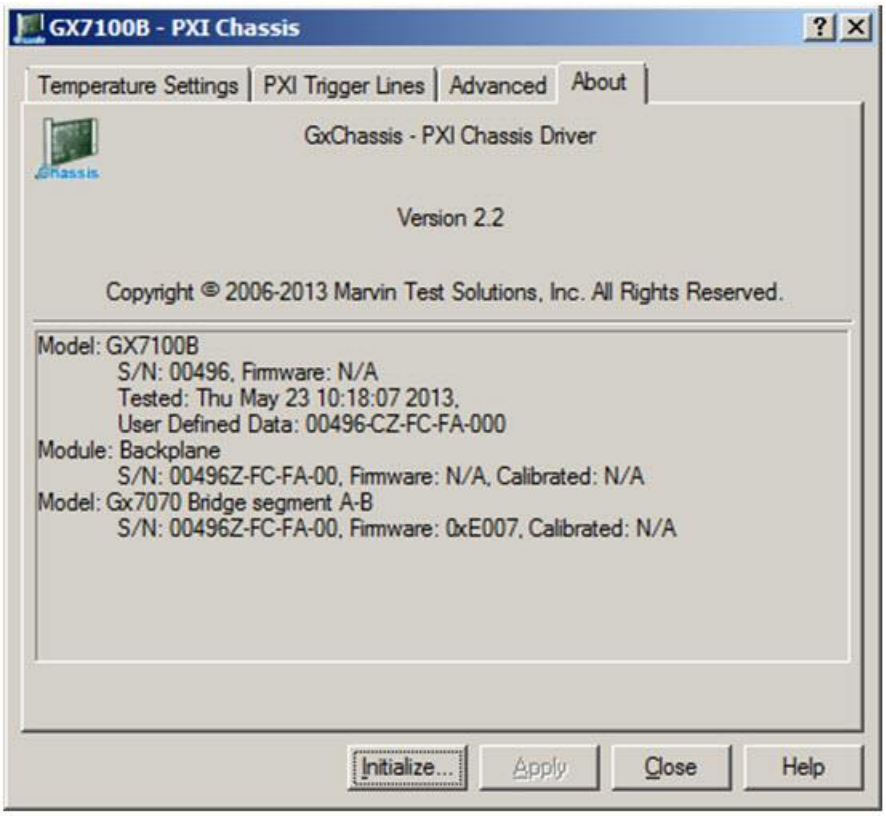


Figure 2-23: GxChassis Virtual Panel – About Page

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 GX71xx chassis and boards.

Unpacking and Inspecting the Chassis

1. Before unpacking the GX71xx, 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 GX71xx is designed to operate on a bench or within an instrument rack system. Follow the appropriate installation instructions for your GX71xx chassis.

Openings on the left and right-side panels and rear panel of the chassis facilitate power supply and instrument cooling. This is very important to the operation of your GX71xx. Make sure to place your unit on a bench top or in an instrument rack so the air intake openings on the sides 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 GX71xx in an instrument rack.

Line Voltage Selection and Current Values

The voltage selection for proper operation of the GX71xx is automatic. The chassis power supply will automatically select the proper input voltage configuration. Check the voltage/current requirements on the GX71xx's rear panel decal to verify that you have the correct circuit breaker. The GX70xxA chassis can operate with line voltages from 100 to 250 VAC.

Chassis Installation

Follow these steps to install the GX71xx chassis:

1. Place the GX7100 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, GX7100) or a remote controller (slave configuration, GX7110) to slot #1 if not installed.
5. Turn on the chassis power and the optional external system (for slave installation turn on the sale first)
6. Install the **GxChassis** software.
7. Configure your system using the **PXI/PCI Explorer** applet.
8. Install any additional drivers for PXI instruments.

9. Turn off the system.
10. Install PXI modules into the chassis as described in the next procedure.
11. Turn on the chassis power switch and follow the Found New Hardware Wizard instructions for new instruments installed.

GX7100 Master and GX7110 Slave Configurations

The GX71xx can be used as a complete system using the GX79xx family of embedded controllers. In this configuration, a GX79xx is used in Slot #1 to provide the system controller for the chassis and the PXI modules.

Many applications require that the CPU be external to the chassis. In some cases, a desktop PC is the controller of the system while in others; another PXI chassis is the controller. In both cases, a remote PXI controller is required (GX7990, GX7990-1, or a MXI-4 interface)

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

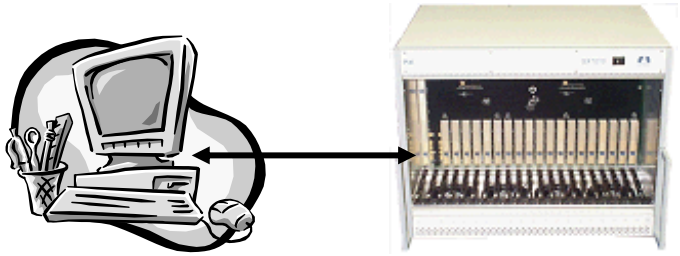


Figure 3-1: Remote configuration with a desktop PC (GX7112A / GX7111A/GX7110A)

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

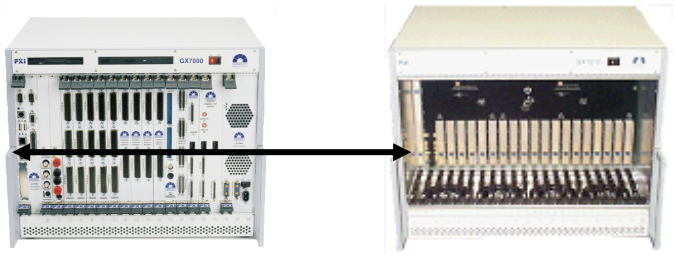


Figure 3-2: Remote configuration with another PXI

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 GX7300 master configuration or on the external PC or the chassis where the controller reside (for GX7310 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/).**
2. Run the GxChassis setup and follow the instruction on the Setup screen to install the GxChassis driver.

   Note: When installing under Windows, you may be required to restart the setup after logging-in as a user with Administrator privileges. This is required in-order to upgrade your system with newer Windows components and to install kernel-mode device drivers (HW driver) which are required by the GxChassis driver to access resources.

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. The default entry for 32-bit machines is C:\Program Files\Marvin Test Solutions\GxChassis, and for 64-bit Windows C:\Program Files (x86)\Marvin Test Solutions\GxChassis.
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.

Refer to Overview of the GxChassis Software in chapter 2 to learn more about the GxChassis software components.

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).

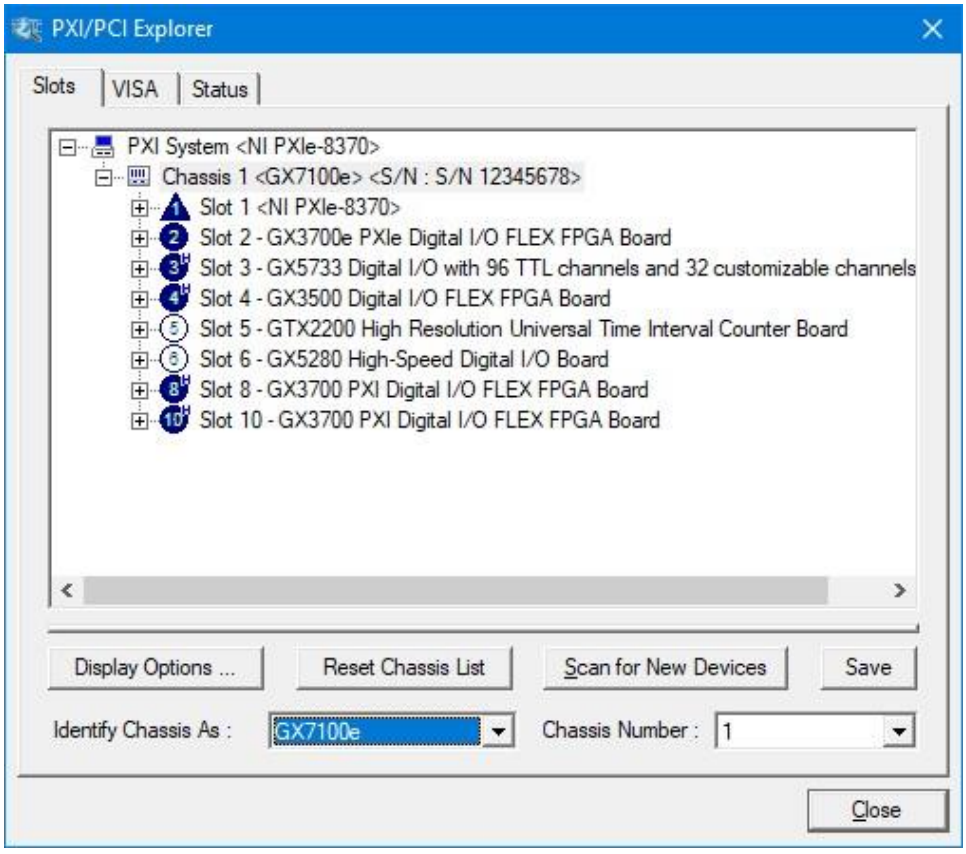


Figure 3-3: PXI/PCI Explorer

Installing a PXI Instrument

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

1. Before installing any PXI board, install the GxChassis software package. Make sure that the latest software package for your board was installed (the latest product setup can be found at https://www.marvintest.com/downloads.aspx). The GxChassis package contains chassis and controller drivers as well as general support for Marvin Test Solutions instruments (HW – Marvin Test Solutions Hardware Access Driver).
2. Turn off the PXI chassis and unplug the power cord.
3. Set the board switches and jumpers if required.

   Caution - Electrostatic discharge can damage components on the GX71xx and PXI modules.

   Check the board documentation for details on jumpers and switch settings before the installation.
4. Locate an empty PXI Slot on the chassis.
5. Place the module edges into the PXI chassis rails (top and bottom).
6. 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).

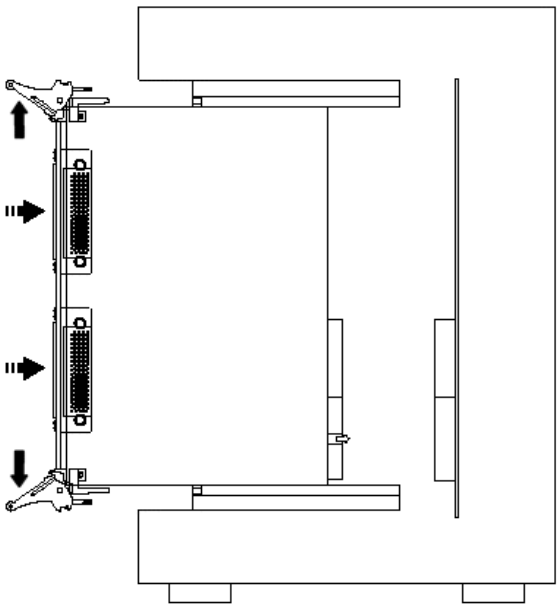


Figure 3-4: Inserting PXI board

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

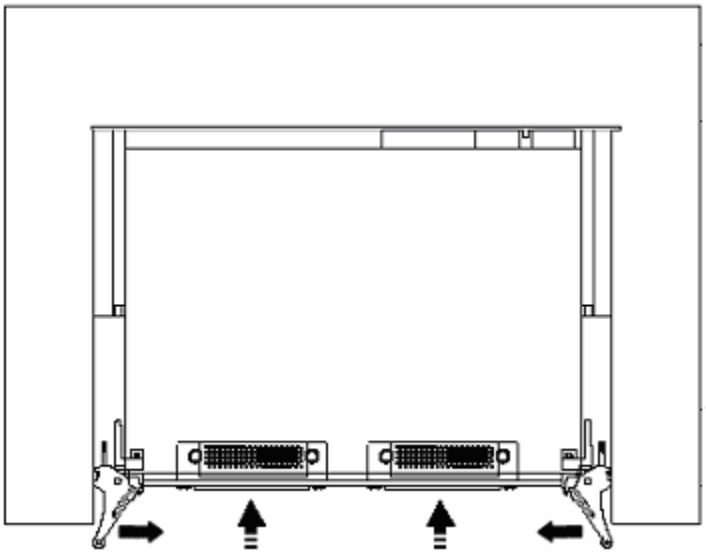


Figure 3-5: Ejector handles position after module insertion

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

PXI Board Software Installation

Plug & Play operating systems, such as Windows, notifies the user whenever a new hardware was found after restarting the system using the Windows **New Hardware Found wizard**. The following describes the installation of Marvin Test Solutions board:

1. **Before installing any new Marvin Test Solutions board, make sure that the latest software package for your board was installed (the latest product setup can be found at https://www.marvintest.com/downloads.aspx).**
2. **Power down the machine, Insert the new hardware to your system and restart the machine.**
3. **Windows will notify the user that a new hardware was found using the Windows** New Hardware Found wizard**. If no wizard is displayed, proceed to step 7.**
4. **Select Browse my computer for driver software.**
5. **Click the Browse button and locate the driver file under C:\Program Files\Marvin Test Solutions\HW folder, or on 64-bit machines C:\Program Files (x86)\Marvin Test Solutions\HW.**
6. **Click** Next **to confirm and follow the instructions on the screen to complete the driver installation.**

7. **Verify that the board driver is installed correctly: Open the appropriate board panel application from the Windows Start menu and use the** Initialize… **button. Select the board PXI slot and press** OK**. If no error is displayed and the panel pages are enabled and displays the correct board settings, then installation is complete. If the no board PXI slot is displayed in the Initialize dialog or initialization failed, proceed to the next step.**

8. **If initializing the new Marvin Test Solutions board fails the driver might have been installed incorrectly by windows:**

   a. **Open the Windows** Device Manager**.**

   b. **The Windows Device Manager should display the board name and description under the** Marvin Test Solutions **entry. If the board is displayed with a yellow or red icon next to it, or the board description is generic and not described the board correctly (i.e. PCI Data Acquisition and Signal Processing Controller is displayed in the device list), follow these steps:**

      1) Right click on the PCI Data Acquisition and Signal Processing Controller, and select Update Driver from the context menu.
      2) Select Browse my computer for driver software.
      3) Click the Browse button and locate the driver file under C:\Program Files\Marvin Test Solutions\HW folder, or on 64-bit machines C:\Program Files (x86)\Marvin Test Solutions\HW.
      4) Click Next to confirm and follow the instructions on the screen to complete the driver installation.
      5) Repeat step 7 and verify that the board driver is installed correctly.

Removing a PXI Instrument

Remove a PXI instrument board as follows:

1. Turn off the PXI chassis and unplug the power cord.
2. Disconnect and remove any cables/connectors connected to the board.
3. Un-tighten the module's front panel screws from the chassis.
4. Push **outward** the ejector handles and pull the PXI board away from the chassis.

Using External Instruments

Your chassis supports all PXI and cPCI instruments. In some cases, however, you may need to connect additional instruments to the chassis. 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 Folders

The GxChassis driver files are installed in the default directory `C:\Program Files\Marvin Test Solutions\GxChassis` under 32-bit Windows, or `C:\Program Files (x86)\Marvin Test Solutions\GxChassis` under 64-bit Windows. 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 folders:

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 ATEasy is installed on your machine.
Windows System Folder	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/GxChassis64.dll – 32/64-bit Windows DLL for 32/64-bit applications running under Windows.
- GxChassisPanel.exe/ GxChassisPanel64.exe – 32/64 bit Windows instrument front panel application for all GxChassis supported chassis.

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.drv - 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 mainframe, 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
- GxChassisExample.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.
- GxChassisExampleVBAssembleyInfo.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 Visual C#

To use the driver with Visual C# the user must include the GxChassis.cs in the project. The file can be loaded using *Add File* from the Visual Basic *File menu*. The GxChassis.cs contains function declarations for the DLL driver.

Programming Using Pascal/Delphi

To use the driver with Borland Pascal or Delphi, the user must include the 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 mainframes 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 GxChassis 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]

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

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

//*************************************************************************
//          DisplayMsg
//*************************************************************************
void DisplayMsg(PCSTR lpszMsg)
{
#ifndef __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 recived
      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 recived
            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 recived
            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 recived
      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 recived
      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 recived
      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 - Function 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.	*anArray* (Array of Short)
n	Short (signed 16-bit)	*nMode*
d	Double - 8 bytes floating point	*dReading*
dw	Double word (unsigned 32-bit)	*dwTimeout*
hwnd	Window handle (32-bit integer).	*hwndPanel*
l	Long (signed 32-bit)	*lBits*
p	Pointer. Usually used to return a value. Prefix this before the simple type.	*pnStatus*
sz	Null (zero value character) terminated string	*szMsg*
w	Unsigned short (unsigned 16-bit)	*wParam*

Table 5-1: Parameter Name Prefixes

GxChassis Functions

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

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

Driver Functions	Description
GxChassisSetTemperatureThresholdMode	Sets the Temperature threshold operational mode.

GxChassisGetAlarmMode

Purpose

Returns the alarm mode.

Syntax

GxChassisGetAlarmMode *(nHandle, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX71xx** mainframe Chassis.
pnMode	PSHORT	Alarm mode is one of the followings: 0. GXCHASSIS_OVER_TEMPERATURE_ALARM_DISABLE – Alarm disabled. 1. GXCHASSIS_OVER_TEMPERATURE_ALARM_ENABLE – Alarm enabled (default). 2. GXCHASSIS_OVER_TEMPERATURE_ALARM_ON – Alarm is on. 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.
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
pdTemp	PDOUBLE	Alarm temperature threshold setting.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Alarm temperature can be programmed to any value between –20°C and +70°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).

Note: Manufacture default Alarm temperature setting is +50°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
nHandle	SHORT	Handle to a GX71xx Chassis.
szSummary	PSTR	Buffer to contain the returned board info (null terminated with 512 bytes).
nSumMaxLen	SHORT	Size of the buffer to contain the board info string.
pnStatus	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: GX7100, S/N: 00063, Firmware: N/A, Tested: Tue Jun 24 13:40:39 2008, User Defined Data: GX7600-0063

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"

Example

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

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

See Also

GxChassisGetDriverSummary, GxChassisGetErrorString

GxChassisGetDriverSummary

Purpose

Returns the driver name and version.

Syntax

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

Parameters

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

Comments

The returned string is: GxChassis - Marvin Test Solutions's PXI Chassis Driver for the Gx7XX0 family, Version 1.0, Copyright © 2006 Marvin Test Solutions, All rights reserved".

Example

The following example prints the driver version:

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

GxChassisGetDriverSummary (sz, sizeof sz, &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
nError	SHORT	Error number as returned by the *pnStatus* 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.
pszMsg	LPSTR	Buffer containing the returned error string (null terminated string).
nErrorMaxLen	SHORT	Size of the buffer *pszMsg*.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the error string associated with the *nError* as returned from other driver functions. The following table displays the possible error values. Not all errors apply to this 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
Parameter Errors	
-20	Invalid parameter
-22	Invalid board handle
-25	Invalid mode
-27	Invalid string length
Board specific parameter error	
-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

Resource Errors	Description
Board Errors/Warnings	
-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
Miscellaneous Errors	
-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, sizeof sz, &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
nHandle	SHORT	Handle to a GX71xx Chassis.
pdVoltage	PDOUBLE	An array containing the backplane power supply voltages.
pnStatus	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

Example

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

```
SHORT   nStatus;
DOUBLE  adVoltage[4];
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
nHandle	SHORT	Handle to a GX71xx Chassis.
pnSpeedControl	PSHORT	Returns the fan speed control mode as follows: 0. 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. 1. GXCHASSIS_FAN_SPEED_MODE_USER_DEFINED: Fan speed is specified by the user (*pnSpeed value*).
pnSpeed	PSHORT	Returns the fan speed as follows: 0. GXCHASSIS_FAN_SPEED_MIN: Fan speed is at the minimum operational range. 1. GXCHASSIS_FAN_SPEED_MID: Fan speed is at the middle operational range. 2. GXCHASSIS_FAN_SPEED_MAX: Fan speed is at the maximum operational range.
pnStatus	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 threshold is >=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 PCB revisions D 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
nHandle	SHORT	Handle to a GX71xx Chassis.
pdMinThreshold	PDOUBLE	Returns the fan's low threshold temperature speed, value is either in Fahrenheit or Celsius as was set by the GxChassisSetTemperatureScale function call.
pdMaxThreshold	PDOUBLE	Returns the fan's high threshold temperature speed, value is either in Fahrenheit or Celsius as was set by the GxChassisSetTemperatureScale function call.
pnStatus	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 <=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 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.

Note: this functionality is supported by PCB revisions D 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

GxChassisGetOrientation

Purpose

Returns the chassis' pitch and roll in degrees in order to determine the cassis orientation.

Syntax

GxChassisGetOrientation (*nHandle, nSegment, pnPitch, pnRoll, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX71xx Chassis.
nSegment	SHORT	Specified PXI chassis Segments, always 0
pnPitch	PSHORT	Returns the chassis pitch in degrees, the pitch angle resolution is 1°.
pnRoll	PSHORT	Returns the chassis roll in degrees, the roll angle resolution is 1°.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function is used to help the user determine if the chassis is in the correct orientation as was specified. The best way to use this function is to set the chassis at its correct orientation and save the pitch and roll of that position in the user's program. Those absolute angles can later be used to compare against pitch and roll angles to determine if the chassis is at the correct orientation.

The returned chassis' orientation gives the pitch and roll values of the Chassis in degrees. The Pitch and roll angles are referenced to the local horizontal plane, which is perpendicular to the Earth's gravity.

Pitch: defined as the angle between the front of the chassis and the horizontal plane. If the front of the chassis is displaced upwards creating an angel between the front of the chassis and the horizontal plane then it is positive pitch. If the front of the chassis is displaced downwards creating a negative angel between the front of the chassis and the horizontal plane then it is positive pitch.

Roll: defined as the rotation of the chassis. If the chassis is rotating clockwise then it is positive roll. If the chassis is rotating counterclockwise then it is positive roll.

Example

The following example returns the chassis pitch and roll in degrees:

```
SHORT   nStatus;
SHORT   nPitch, nRoll;
GxChassisGetOrientation (nHandle, GXCHASSIS_PXI_TRIGGER_BUS_LINE0,
                         GXCHASSIS_SEGMENT_0_TO_SEGMENT_1, &nPitch, &nRoll, nStatus);
```

See Also

GxChassisGetErrorString

GxChassisGetPxiTriggerLine

Purpose

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

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX71xx Chassis.
nLine	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
nSegment	SHORT	Specified PXI chassis Segments, always 0
pnDirection	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.
pnPrimaryMode	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). **Note**: this functionality is supported by bridgeboard revisions G and above; previous bridgeboard revision will not be affected.
pnSecondaryMode	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
nHandle	SHORT	Handle to a GX71xx Chassis.
nLine	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
nSegment	SHORT	Specified PXI chassis Segments, always 0
pnPrimary	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.
pnSecondary	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.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Note: this functionality is supported by PCB revisions D 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
nHandle	SHORT	Handle to a GX71xx Chassis.
pbEnable	PBOOL	Shutdown state: 0. Disabled. 1. Enabled (default).
pdThreshold	PDOUBLE	Shutdown Temperature threshold settings, value can be between +20°C to +70°C.
pnStatus	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: the manufacture default threshold is programmed to +70°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
nHandle	SHORT	Handle to a GX71xx Chassis.
pdTemp	SHORT	Array holding all measured slot temperatures. Measured temperature of slot 1 returned in array index 0.
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
pdwStates	PDWORD	Returns slots with active or enabled temperature monitoring, bits 0 through 19 represents slots 1 through 20. • Bit high – specified slot is enabled. • Bit low – specified slot is disabled.
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
pnMinTempSlot	PSHORT	The slot number with the lowest temperature out of all active slots.
pdMinTemp	PDOUBLE	The temperature of the slot number with the lowest temperature out of all active slots.
pnMaxTempSlot	PSHORT	The slot number with the highest temperature out of all active slots.
pdMaxTemp	PDOUBLE	The temperature of the slot number with the highest temperature out of all active slots.
pdAveTemp	PDOUBLE	The average temperature of all active slots.
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
nSlot	SHORT	Specified slot temperature. Slot number can be from 1 to 20.
pdTemp	SHORT	Array holding all measured slots' temperatures. Measured temperature of slot 1 returned in array cell number 0.
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
pnScale	PSHORT	Temperature scale: 0. GXCHASSIS_TEMPERATURE_SCALE_METRIC 1. GXCHASSIS_TEMPERATURE_SCALE_ENGLISH
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
pnMode	PSHORT	Temperature threshold operation modes are: 0. GXCHASSIS_OVER_TEMPERATURE_MODE_MAX_SLOT (default) 1. GXCHASSIS_OVER_TEMPERATURE_MODE_AVERAGE_SLOTS
pnStatus	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
nChassis	Short	PXI Chassis number.
pnHandle	PSHORT	Returned handle for the to a GX71xx mainframe Chassis. The handle is set to zero on error and <> 0 on success.
pnStatus	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
pnHandle	PSHORT	Handle to a GX71xx Chassis. This number may be zero if the board is to be initialized by the panel window.
hwndParent	HWND	Sets the panel parent window handle. A value of 0 sets the desktop as the parent window.
nMode	SHORT	The mode in which the panel main window is created. 0 for modeless and 1 for modal window.
phwndPanel	LPHWND	Returned window handle for the panel (for modeless panel only).
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
nSettingSource	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.
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
nSegment	SHORT	Specified PXI chassis Segments, always 0.
pnStatus	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 segement's primery 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
nHandle	SHORT	Handle to a GX71xx Chassis.
nMode	SHORT	Over Temperature Alarm mode is one of the followings: 0. GXCHASSIS_OVER_TEMPERATURE_ALARM_DISABLE – Alarm disabled. 1. GXCHASSIS_OVER_TEMPERATURE_ALARM_ENABLE – Alarm enabled. 2. GXCHASSIS_OVER_TEMPERATURE_ALARM_ON – Alarm is on. 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.
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
dTemp	DOUBLE	Alarm temperature threshold settings
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The programmable Alarm temperature 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 Alarm temperature is +50°C.

Example

The following example sets the Alarm temperature to 45°C:

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

See Also

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

GxChassisSetFanSpeed

Purpose

Sets the fan speed and control settings.

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX71xx Chassis.
nSpeedControl	SHORT	Sets the fans speed control mode as follows: 0. 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. 1. GXCHASSIS_FAN_SPEED_MODE_USER_DEFINED: Fans speed is specified by the user (*pnSpeed value*).
nSpeed	SHORT	Sets the fans speed as follows: 0. GXCHASSIS_FAN_SPEED_MIN: Fan speed is at the minimum operational range. 1. GXCHASSIS_FAN_SPEED_MID: Fan speed is at the middle operational range. 2. GXCHASSIS_FAN_SPEED_MAX: Fan speed is at the maximum operational range.
pnStatus	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 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 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 PCB revisions D 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
nHandle	SHORT	Handle to a GX71xx Chassis.
dMinThreshold	DOUBLE	Fan's low threshold temperature speed. Value is either in Fahrenheit or Celsius as was set by the GxChassisSetTemperatureScale function call.
dMaxThreshold	DOUBLE	Fan's high threshold temperature speed, Value is either in Fahrenheit or Celsius as was set by the GxChassisSetTemperatureScale function call.
pnStatus	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.

Note: this functionality is supported by PCB revisions D 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
nHandle	SHORT	Handle to a GX71xx Chassis.
nLine	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
nSegment	SHORT	Specified PXI chassis Segments, always 0.
nDirection	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.
nPrimaryMode	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). **Note**: this functionality is supported by bridgeboard revisions G and above; previous bridgeboard revision will not be affected.
nSecondaryMode	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 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).

Note: this functionality is supported by PCB revisions D 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
nHandle	SHORT	Handle to a GX71xx Chassis.
bEnable	BOOL	Shutdown state: 0. Disabled. 1. Enabled (default).
dThreshold	DOUBLE	Shutdown Temperature threshold settings, value can be between +20°C to +70°C.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The programmable shutdown temperature 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
nHandle	SHORT	Handle to a GX71xx Chassis.
dwStates	DWORD	Defines slots that will be actively monitored for temperature. Bits 0 through 19 represents slots 1 through 20. • Bit high – specified slot enabled. • Bit low – specified slot disabled.
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
nScale	SHORT	Temperature scale: 0. GXCHASSIS_TEMPERATURE_SCALE_METRIC 1. GXCHASSIS_TEMPERATURE_SCALE_ENGLISH
pnStatus	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
nHandle	SHORT	Handle to a GX71xx Chassis.
pnMode	PSHORT	Temperature threshold operational modes are: 0. GXCHASSIS_OVER_TEMPERATURE_MODE_MAX_SLOT (default) 1. GXCHASSIS_OVER_TEMPERATURE_MODE_AVERAGE_SLOTS
pnStatus	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 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

GX7100/GX7110/GX7102/GX7112/GX7111

100 – 250 VAC, 47 – 63 HZ, 10 A max. With PFCInput AC power is filtered using a line filter.

Power Supply

GX7100/GX7102/GX7111/GX7110/GX7112

560W DC power supply providing:

- +5 VDC @ 60A (Max)
- +3.3 VDC @ 40A (Max)
- +12 VDC @ 32A (Max)
- -12 VDC @ 3A (Max)

560W Power Supply Load, Regulation, Ripple, and Noise Specifications

Output Voltage		Load Range		Regulation		Ripple Max		Ripple & Noise*	
		Min.	Max.	Min.	Max.	mV	P-P	Max.	mV P-P
1	+3.3V	0.2 A	40.0 A**	- 5 %	+ 5 %	50	mV	100	mV
2	+5.0V	2.5 A	60.0 A**	- 5 %	+ 5 %	50	mV	100	mV
3	+12.0V	0.5 A	32.0 A	- 5 %	+ 5 %	100	mV	150	mV
4	-12.0V	0.0 A	3.0 A	- 10 %	+ 10 %	150	mV	200	mV

*****Noise Bandwidth: DC – 20 MHz

**Total 5V and 3.3V power cannot exceed 300 watts.

Cooling

Four 20 CFM fans are mounted to the left wall and four fans are located in the center wall of the card cage, providing a positive airflow. Air exhaust is through the right chassis side panel.

An additional 25CFM fan provides cooling for the power supply.

B Models or Newer

Fan speed control and monitoring is automatic or can be controlled / monitored via the GxChassis software.

Temperature Monitoring

Integrated temperature monitoring via on-board microcontroller with audible and software notification when preset temperature limits are exceeded. Temperature monitoring features include:

- 7 sensors, one per each 3U/6U horizontal slot,
- 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 system power supply voltages: 3.3 5, +12, =12, VIO value

Accuracy: +/- 2% of reading

PXI 10 MHz Clock

A Series

Integrated 10MHz PXI system clock with auto-detect function. Presence of an external 10 MHz PXI clock will disable the internal clock source.

10 MHz clock accuracy: +/- 100 ppm

B Models or Newer

The chassis includes an integrated 10MHz PXI system clock with auto-detect function. Presence of an external 10 MHz PXI clock will disable the internal clock source. External source can be from the Star Trigger Controller Slot (slot 2) or an external input. Precedence is as follows:

1. **10 MHz source from Slot 2 timing slot**
2. **10 MHz source from external input (rear panel)**
3. **10 MHz source from the backplane**

10 MHz clock accuracy: +/- 100 ppm

External 10 MHz Clock (rear panel):

External 10 MHz input: TTL compatible

External 10 MHz output: TTL compatible

Slots

The GX71xx has a total of 14 slots:

- 1 System Controller Slot
- 1 PXI Star Trigger Controller Slot (can be used by any PXI/cPCI instrument)
- 12 PXI/cPCI Instruments with Star Trigger

Physical Dimensions

Empty Weight:	GX7100: 38 lbs.
	GX7102: 28 lbs.
	GX7110: 23 lbs.
	GX7111: 23 lbs.
	GX7112: 25 lbs.
Dimensions:	GX7100, GX7110, GX7111: 4U (7") H x 17.6" W x 16" D
	GX7102, GX7112: 6U (10.5") H x 17.6" W x 23" D

Environmental

Operating Temperature Range:	0ºC to 55ºC
Storage Temperature Range:	-20ºC to +85ºC
Operating relative humidity:	10 to 90%, non-condensing
Storage relative humidity:	5 to 95%, non-condensing
Emissions:	EN61010-1, EN61236

Appendix B –PXI Slots Pin Outs

This appendix describes the P1 and P2 connector pin outs for the GX7100/GX7110 backplane.

- Table B-1 shows the P1 connector pin out for the System Controller slot.
- Table B-2 shows the P2 connector pin out for the System Controller slot.
- Table B-3 shows the P1 connector pin out for the Star Trigger slot.
- Table B-4 shows the P2 connector pin out for the Star Trigger slot.
- Table B-5 shows the P1 connector pin out for the peripheral slots.
- Table B-6 shows the P2 connector pin out for the peripheral slots.

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).

System	Signals		
PXI	PXI_BRSV	PXI_LBL[0:12]	PXI_STAR[0:12]
	PXI_CLK10	PXI_STAR[0:12]	PXI_TRIG[0:7]
	PXI_CLK10_IN	PXI_TRIG[0:7]	
CompactPCI	BD_SEL#	HEALTHY#	REQ#[0:6]
	BRSV	INTP	RSV
	CLK[0:6]	INTS	SYSEN#
	DEG#	IPMB_PWR	SMB_ALERT#
	ENUM#	IPMB_SCL	SMB_SCL
	FAL#	IPMB_SDA	SMB_SDA
	GA0-GA4	PRST#	UNC
	GNT#[0:6]		
PCI	ACK64#	AD[0:63]	C/BE[0:7]#
	CLK	DEVSEL#	FRAME#
	GND	GNT#	IDSEL
	INTA#	INTB#	INTC#
	INTD#	IRDY#	LOCK#
	M66EN	PAR	PAR64
	PERR#	REQ#	REQ64#
	RST#	SERR#	STOP#
	TCK	TDI	TDO
	TMS	TRDY#	TRST#
	V(I/O)	3.3 V	5 V
	+12 V	-12 V	

Table B-1: Signal Names grouped by BUS

P1 Connector Pin Out for System Controller Slot

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#	GND	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]#	GND	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	REQ0#	GND	3.3V	CLK0	AD[31]	GND
5	GND	BRSVP1A5	BRSVP1B5	RST#	GND	GNT0#	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-2: P1 Connector Pin Out for the System Controller Slot

P2 Connector Pin Out for System Controller Slot

Pin	Z	A	B	C	D	E	F
22	GND	GA4	GA3	GA2	GA1	GA0	GND
21	GND	CLK6	GND	RSV	RSV	RSV	GND
20	GND	CLK5	GND	RSV	GND	RSV	GND
19	GND	GND	GND	SMB_SDA	SMB_SCL	SMB_ALERT#	GND
18	GND	PXI_TRIG3	PXI_TRIG4	PXI_TRIG5	GND	PXI_TRIG6	GND
17	GND	PXI_TRIG2	GND	PRST#	REQ6#	GNT6#	GND
16	GND	PXI_TRIG1	PXI_TRIG0	DEG#	GND	PXI_TRIG7	GND
15	GND	PXI_BRSVA15	GND	FAL#	REQ5#	GNT5#	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	CLK4	GND	GNT3#	REQ4#	GNT4#	GND
2	GND	CLK2	CLK3	SYSEN#	GNT2#	REQ3#	GND
1	GND	CLK1	GND	REQ1#	GNT1#	REQ2#	GND

Table B-3: P2 Connector Pin Out for the System Controller Slot

P1 Connector Pin Out for the Star Trigger Slot

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-4: P2 Connector Pin Out for the System Controller Slot

P2 Connector Pin Out for the Star Trigger Slot

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_STAR0	GND	PXI_STAR1	GND
19	GND	PXI_STAR2	GND	PXI_STAR3	PXI_STAR4	PXI_STAR5	GND
18	GND	PXI_TRIG3	PXI_TRIG4	PXI_TRIG5	GND	PXI_TRIG6	GND
17	GND	PXI_TRIG2	GND	RSV	PXI_CLK10_IN	PXI_CLK10	GND
16	GND	PXI_TRIG1	PXI_TRIG0	RSV	GND	PXI_TRIG7	GND
15	GND	PXI_BRSVA15	GND	RSV	PXI_STAR6	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_STAR7	PXI_STAR8	GND
1	GND	PXI_STAR9	GND	PXI_STAR10	PXI_STAR11	PXI_STAR12	GND

Table B-5: P2 Connector Pin Out for the Star Trigger Slot

P1 Connector Pin Out for the Peripheral Slot

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-6: P1 Connector Pin Out for the Peripheral Slot

P2 Connector Pin Out for the Peripheral Slot

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-7: P2 Connector Pin Out for the Peripheral Slot

Appendix C – Rear Panel Connector Layout

This section provides information on the rear panel connectors of the GX710x (when used in conjunction with the GX79XX embedded controller).

Serial Port Connector

Connector Type: D-Sub, 9 pins

Mating Connector: 9-pin D-Sub, Female

Pin #	Signal Name	Signal Function	Direction
1	DCD	Data carrier detect	In
2	RXD	Receive data	In
3	TXD	Transmit data	Out
4	DTR	Data terminal ready	Out
5	GND	Signal ground	--
6	DSR	Data send request	In
7	RTS	Request to send	Out
8	CTS	Clear to send	In
9	RI	Ring indicator	In

Keyboard Connector

Connector Type: PS/2 mini DIN

Mating Connector: PS/2, Male

Pin #	Signal Name	Signal Function	Direction
1	KDATA	Keyboard data	In/out
2	MDATA	Mouse data	In/out
3	GND	Ground signal	--
4	VCC	VCC signal	--
5	KCLK	Keyboard clock	Out
6	MCLK	Mouse clock	Out

Mouse Connector

Connector Type: PS/2 mini DIN

Mating Connector: PS/2, Male

Pin #	Signal Name	Signal Function	Direction
1	MDATA	Mouse data	In/out
2	NC	No Connection	--
3	GND	Ground signal	--
4	VCC	VCC signal	--
5	MCLK	Mouse clock	Out
6	NC	No Connection	--

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	--	--

USB Connector

Connector Type: USB

Mating Connector: USB

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

Optional SCSI Connector

Connector Type: SCSI

Mating Connector: SCSI-3

Pin	Signal	Pin	Signal	Pin	Signal
1	GND	24	GND	47	-DB(7)
2	GND	25	GND	48	-DB(P)
3	GND	26	GND	49	GND
4	GND	27	GND	50	CONN-Detect
5	GND	28	GND	51	TERMPWR
6	GND	29	GND	52	TERMPWR
7	GND	30	GND	53	RESERVED
8	GND	31	GND	54	GND
9	GND	32	GND	55	-ATN
10	GND	33	GND	56	GND
11	GND	34	GND	57	-BSY
12	GND	35	-DB(12)	58	-ACK
13	GND	36	-DB(13)	59	-RST
14	GND	37	-DB(14)	60	-MSG
15	GND	38	-DB(15)	61	-SEL
16	GND	39	-DB(P1)	62	-C/D
17	TERMPWR*	40	-DB(0)	63	-REQ
18	TERMPWR*	41	-DB(1)	64	-I/O
19	RESERVED*	42	-DB(2)	65	-DB(8)
20	GND	43	-DB(3)	66	-DB(9)
21	GND	44	-DB(4)	67	-DB(10)
22	GND	45	-DB(5)	68	-DB(11)
23	GND	46	-DB(6)		

Parallel Port Connector

Connector Type: D-Sub, 25 Pins

Mating Connector: 25 pin D-Sub, Male

Pin #	Signal Name	Signal Function	Direction	Pin #	Signal Name	Signal Function	Direction
1	-STB	Strobe data	Out	14	-AFD	Auto feed	Out
2	PD0	LSB of printer data	Out	15	-ERR	Printer error	In
3	PD1	Printer data 1	Out	16	-INIT	Initialize printer	Out
4	PD2	Printer data 2	Out	17	-SLIN	Select printer	Out
5	PD3	Printer data 3	Out	18	GND	Signal ground	N/A
6	PD4	Printer data 4	Out	19	GND	Signal ground	N/A
7	PD5	Printer data 5	Out	20	GND	Signal ground	N/A
8	PD6	Printer data 6	Out	21	GND	Signal ground	N/A
9	PD7	Printer data 7	Out	22	GND	Signal ground	N/A
10	-ACK	Character accepted	In	23	GND	Signal ground	N/A
11	BSY	Busy	In	24	GND	Signal ground	N/A
12	PE	Paper end	In	25	GND	Signal ground	N/A
13	SLCT	Ready to receive	In				

Appendix D – Model Numbers

Chassis Model Numbers

The following are the PXI chassis model numbers:

Model #	Description
GX7100A	6U/3U, 14 Slot PXI *Smart* Master Desktop Combo Chassis w/CD-ROM, Hard Disk and Floppy Disk Drives
GX7100AR	6U/3U, 14 Slot PXI *Smart* Master Rack Mount Combo Chassis w/CD-ROM, Hard Disk and Floppy Disk Drives
GX7100B	6U/3U, 14 Slot PXI *Smart* Master Desktop Combo Chassis w/CD-ROM, Hard Disk.
GX7100BR	6U/3U, 14 Slot PXI *Smart* Master Rack Mount Combo Chassis w/CD-ROM, Hard Disk.
GX7110A/B	6U/3U, 14 Slot PXI *Smart* Slave Desktop Combo Chassis for use with GX7990 PXI Bus Expander
GX7110A/B R	6U/3U, 14 Slot PXI *Smart* Slave Rack Mount Combo Chassis for use with GX7990 PXI Bus Expander
GX7102A/B	PXI *Smart* Master Combo Chassis with an integrated Cable Tray and a Hinged front panel for Mass Interconnect (rack mount configuration with handles)
GX7112A/B	PXI *Smart* Slave Combo Chassis with an integrated Cable Tray and a Hinged front panel for Mass Interconnect (rack mount configuration with handles)
GX7111A/B	PXI *Smart* Slave Combo Chassis with a 2" recessed instrument card cage
GX7111A/B R	PXI *Smart* Slave Combo Chassis with a 2" recessed instrument card cage w/rack-mount
GX79XX	CPU Plug-in controller for GX7100A/GX7102A/GX7100B/GX7102B. Contract factory for specific model numbers.
GX7990	PXI Bus Expander for GX7110A/GX7111A/GX7112A/GX7110B/GX7111B/GX7112B. Two cards set with a 4 ft. cable.

Chassis Accessory Model Numbers

The following are the PXI chassis accessory model numbers:

Model #	Description
GX97000	Rack mount kit for GX7100A/B
GX97003	Rack mount kit with handles for GX7100A/B
GX97001	Blank Panel for GX71X0A/B, 1-slot wide
GX97002	Blank Panel for GX71X 0A/B, 2-slots wide
GX97004	Blank Panel for GX71X 0A/B, 4-slots wide
GX97005	3U to 6U Panel Adapter (allows a 3U instrument to fit into a 6U chassis)
GX79XX-NT	Upgrade any GX79XX to Windows NT Operating System
GX79XX-64M	64MB Memory Upgrade to any GX79XX
GX79XX-128M	128MB Memory Upgrade to any GX79XX

Index

.

.NET ii

3

3U Boards 4, 6, 12, 13, 100

6

6U Boards 4, 6, 12, 13, 100

A

AC Input Power 83
Architecture 1
ATEasy ..... ii, 4, 32, 33, 35, 36

B

Board Handle 36
Borland 32, 35, 36
Borland-Delphi 36

C

C/C++ 32, 33, 35
C++ 35
CD-ROM 4, 8, 99
Chassis Accessory Model Numbers 100
Chassis Configuration 25
Chassis Description 1
Chassis Description 7
Chassis Installation 25
Chassis Model Numbers 99
Chassis Models 5
Chassis Orientation 15
Clock 84
COM 10, 11
CompactPCI 3, 4, 5, 87
Corrupt files 34
cPCI 3, 4, 5, 7, 12, 85

D

Delphi ..... ii, 32, 35, 36
Dimensions 85
Distributing 37
Driver
- Directory 32
- Files 32

E

Environmental 85
Error-Handling 36
Ethernet Connector 10, 11, 96
Example 33

F

Fan 83
Features 4
Floppy Disk Drive 8
Folders 32
Functions Reference 45

G

GPIB 5
GX7010A 5
GX7100A 99
GX7100AR 99
GX7100B 99
GX7100BR 99
GX71020A 5
GX7102A/B 99
GX7110 4
GX7110A 4
GX7110A/B 99
GX7110A/B R 99
GX7111A/B 99
GX7111A/B R 99
GX7112A/B 99
GX71xx 16
Gx71xxA Chassis Description 10
Gx71xxB Chassis Description 11
GX7900 ..... 5, 7, 8, 10, 11, 12, 26, 95, 99, 100
GX7990 ..... 4, 5, 7, 12, 26, 99
GX97000 100
GX97001 100
GX97002 100
GX97003 100
GX97004 100
GX97005 100
GxChassis 27
- Driver-Description 35
- Header-file 35
- Help-File-Description 33
- Supported-Development-Tools 35

GxChassis driver Features 17
GxChassis Functions 46
GxChassis.bas 32
GxChassis.BAS 35
GxChassis.cs 36
GxChassis.dll 32, 35, 36
GxChassis.exe 37
GxChassis.h 32, 35

GxChassis.lib....35
GxChassis.LIB....32
GxChassis.pas....32, 36
GxChassis.vb....33
GxChassisBC.lib....35
GxChassisBC.LIB....32
GxChassisGetAlarmMode....48
GxChassisGetAlarmTemperature....49
GxChassisGetBoardSummary....50
GxChassisGetDriverSummary....37
GxChassisGetDriverSummary....51
GxChassisGetErrorString....36
GxChassisGetFanSpeed....55, 74
GxChassisGetFanThresholdTemperatures....56
GxChassisGetOrientation....57
GxChassisGetPowerSuppliesVoltages....54
GxChassisGetPxiTriggerLine....58, 60
GxChassisGetShutdownTemperature....61
GxChassisGetSlotsTemperatures....62
GxChassisGetSlotsTemperaturesStates....63
GxChassisGetSlotsTemperaturesStatistics....64
GxChassisGetSlotTemperature....65
GxChassisGetTemperatureScale....66
GxChassisGetTemperatureThresholdMode....67
GxChassisInitialize....36, 68
GxChassisPanel....37, 69
GxChassisRecallSettings....70
GxChassisResetPxiTriggerLines....71
GxChassisSetAlarmMode....72
GxChassisSetAlarmTemperature....73
GxChassisSetFanThresholdTemperatures....75
GxChassisSetPxiTriggerLine....76
GxChassisSetShutdownTemperature....78
GxChassisSetSlotsTemperaturesStates....79
GxChassisSetTemperatureScale....80
GxChassisSetTemperatureThresholdMode....81

H

handle....29
Handle....36
HW....32, 37

I

Input Power Receptacle....10, 11
Inspecting the GX71xx....25
Installation
  Chassis....25
  Mounting information....25
  Procedures-All-Boards....31
  PXI Module....29, 31
Installation Folders....32

J

J1....*See* P1
J2....91, *See* P2

K

Keyboard Connector....10, 11, 95

L

LabView....34
Line Voltage Selection....25
Local Bus....13
LPT....4, 10, 98

M

Model Numbers....99
Mouse Connector....96

N

nHandle....36

O

OnError....36
Optional Equipment....6
Optional SCSI Connector....97

P

P1....13, 88, 90, 92, 97
P2....13, 89, 91, 93
Panel....18, 19, 21, 22, 23, 32, 33, 34, 37
Parallel Port....4, 10, 98
Pascal....32, 35, 36
PCI....32
Peripheral....13
Pinouts....87
pnStatus....36
Power Distribution....16
Power Supply....83
Power Switch....7
Program-File-Descriptions....32
Programming
  Borland-Delphi....36
  Error-Handling....36
  Panel-Program....37
PXI....3, 5, 19, 28, 29, 30, 31
PXI Board Software Installation....30
PXI Bus Segments....12
PXI Module Installation....29, 31
PXI Slots....12
PXI Trigger Lines....21
PXI/PCI Explorer....19, 28

R

README.TXT....32, 33
Readme-File....33

Rear Panel.....95
RS-232.....4, 5

S

Sample Program Listing.....38
Sample Programs.....37
Serial.....4
Serial COM ports.....10, 11
Serial Port Connector.....95
Setup.....27, 32, 34
Setup Maintenance Program.....34
Slot..... 4, 7, 12, 16, 26, 29, 85, 90, 91, 92, 93, 99
Slots.....4, 12, 13, 85, 87
  Peripheral.....13
  Star Trigger.....7, 12
  System.....7, 12, 16, 85, 88, 89
Specifications.....1, 83
Stand Alone Configuration.....26
System
  Directory.....32
System Reference Clock.....15, 16

T

Temperature Settings.....19
The PXI Standard 5
Trigger Bus.....14, 15

U

Unpacking the GX71xx.....25
USB Connector.....10, 11, 96
Using External Instruments 31

V

Virtual Panel..... 18, 19, 21, 22, 23, 27, 32, 33
  About Page.....23
  Advanced page 22
  Initialize Dialog.....19
Virtual Panel Description 18
Visual Basic.....ii, 35, 36
Visual C#.....36
Visual C++.....ii, 32, 35
VXI.....5

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

Gold-standard solutions

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

We buy equipment

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

Learn more!

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

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

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

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

ARTISAN® TECHNOLOGY GROUP