GX7600

GX7610

GX7600 3U PXI Express Instrumentation Platform Series

GxChassis Software

User's Guide

Last updated September 2, 2014

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

Trademarks

ATEasy®, CalEasy, DIOEasy®, DtifEasy, WaveEasy	Marvin Test Solutions (prior name is Geotest Inc Test Systems Inc.)
C++ Builder, Delphi	Embarcadero Technologies Inc.
LabView, LabWindows^{tm}/CVI	National Instruments
Microsoft Developer Studio, Microsoft Visual C++, Microsoft Visual Basic,, .NET, Windows 95, 98, NT, ME, 2000, XP, VISTA and Windows 7	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
GX76xx Features....4
The PXI Standard....5
PXI and PCI Express Overview....5
GX76xx PXI Express Features....6
GX76xx Models....6
GX76xx Chassis Description – Front View....7
GX76xx Chassis Description – Rear View....10
Optional Equipment....11
Gx76xx Boards....12
3U PXI and Compact PCI Boards....12
3U PXI Hybrid Boards....13
3U PXI Express Boards....13
3U PXI Express System Timing Boards....14
3U PXI Express System Boards....14
PXI Bus Segments....15
System Controller Slot....15
PXI Express System Timing Slot....15
Local Bus....15
Trigger Bus....16
Star Trigger Lines....17
System Reference Clocks....17

System Power Supply 17
Power Distribution 18
GxChassis Driver Features 19
Virtual Panel Description 20
Virtual Panel Initialize Dialog 21
Virtual Panel Temperature Settings 22
Virtual Panel PXI Trigger Lines 24
Virtual Panel Advanced page 25
Virtual Panel About Page 26
Chapter 3 - Setup and Installation 27
Unpacking and Inspecting the Chassis 27
Mounting Information 27
Line Voltage Selection and Fuse Values 27
GX7600 and GX7610 Chassis Installation 27
GX7600 Master and GX7610 Slave Configurations 28
Installation of the GxChassis Software 29
Overview of the GxChassis Software 30
Configuring Your PXI System using the PXI/PCI Explorer 31
GX7600 and GX7610 PXI Instrument Installation 32
PXI Instrument Removal 33
Using External Instruments 33
Installation Directories 34
Driver Files Description 34
Driver File and Virtual Panel 34
Interface Files 34
On-line Help and Manual 35
ReadMe File 35
Program Example 35
Setup Maintenance Program 36
Chapter 4 - Programming the Chassis 37
Overview 37
The GxChassis Driver 37
Programming Using C/C++ Tools 37
Programming Using Visual Basic 37
Programming Using Pascal/Delphi 38
Programming GxChassis Boards Using ATEasy ® 38
Using the GxChassis Driver Functions 38

Chassis Handle 38
Error Handling 38
Driver Version 38
Virtual Front Panel Window 39
Distributing the Driver 39
Sample Programs 40
Sample Program Listing 40
Chapter 5 - Functions Reference 47
Introduction 47
GxChassis Functions 48
GxChassisGetAlarmMode 50
GxChassisGetAlarmTemperature 51
GxChassisGetBoardSummary 52
GxChassisGetDriverSummary 53
GxChassisGetErrorString 54
GxChassisGetPowerSuppliesVoltages 56
GxChassisGetFanSpeed 57
GxChassisGetFanThresholdTemperatures 58
GxChassisGetPxiTriggerLine 59
GxChassisGetPxiTriggerLineLevels 61
GxChassisGetShutdownTemperature 62
GxChassisGetSlotsTemperatures 63
GxChassisGetSlotsTemperaturesStates 64
GxChassisGetSlotsTemperaturesStatistics 65
GxChassisGetSlotTemperature 66
GxChassisGetTemperatureScale 67
GxChassisGetTemperatureThresholdMode 68
GxChassisInitialize 69
GxChassisPanel 70
GxChassisRecallSettings 71
GxChassisResetPxiTriggerLines 72
GxChassisSetAlarmMode 73
GxChassisSetAlarmTemperature 74
GxChassisGetFanSpeed 75
GxChassisSetFanThresholdTemperatures 76
GxChassisSetPxiTriggerLine 77
GxChassisSetShutdownTemperature 79

GxChassisSetSlotsTemperaturesStates 80
GxChassisSetTemperatureScale 81
GxChassisSetTemperatureThresholdMode 82
Appendix A – GX7600/GX7610 System Specifications 83
AC Input Power 83
Power Supply 83
Cooling 83
Temperature Monitoring 84
Power Supply Monitoring 84
PXI Clocks 84
External 10 MHz Clock Input 84
10 MHz Clock Output 84
Slots 84
Physical Dimensions 85
Environmental 85
Appendix B –PXI Slots Pin Outs 87
Bus Signal Names 88
Back Plane Connector Layouts by Slot Type 89
XP4 (XJ4) Connector Pin Out for System Controller Slot 90
XP3 (XJ3) Connector Pin Out for System Controller Slot (4-Link Configuration) 90
XP2 (XJ2) Connector Pin Out for System Controller Slot (4-Link Configuration) 91
XP1 (XJ1) Connector Pin Out for System Controller Slot 91
XP4 (XJ4) Connector Pin Out for System Timing Slot 92
XP3 (XJ3) Connector Pin Out for System Timing Slot 92
TP2 (TJ2) Connector Pin Out for System Timing Slot 93
XP4 (XJ4) Connector Pin Out for Hybrid Slots 94
XP3 (XJ3) Connector Pin Out for Hybrid Slots 94
P1 (J1) Connector Pin Out for Hybrid Slots 95
P1 (J1) Connector Pin Out for PXI-1 Peripheral Slots 96
P2 (J2) Connector Pin Out for PXI-1 Peripheral Slots 97
Appendix C – Rear Panel Connector Layout 99
Ethernet Connector 99
USB Connector 99
Appendix D – Model Numbers 101
Chassis and Controller Model Numbers 101
Chassis Accessory Model Numbers 101
Index 103

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 GX7600 PXI Express chassis. This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and some understanding of functional test systems.

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

Manual Organization

The GX7600 PXI chassis manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX7600 PXI Express chassis manual. Lists all the supported boards and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX7600 PXI Express 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 GX7600'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 – GX7600/GX7610 System Specifications	Provides the GX76XX specifications.
Appendix B – PXI Slots Pin Outs	Describes the P1 and P2 connector pin outs for the GX7600 and GX7610 backplanes.
Appendix C – Rear Panel Connector Layout	Provides information on the rear panel connectors of the GX7600.
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 GX7600 or the GX7610 PXI Express instrumentation chassis' (GX76xx). This GX76xx chassis is designed for test, data acquisition, process control, and factory automation applications. The GX76xx is a 3U, 9-slot PXI Express chassis for desktop or bench top applications. The GX76xx is based on the CompactPCI Express™ (cPCIe) and PXI Express™ (PCI eXtentions for Instrumentation Express) standards and accommodates up to (5) 3U PXI or cPCI instruments, (2) 3U PXI or cPCI Hybrid instruments and (1) 3U PXI Express or cPCI Express instruments, and a PXI Express controller or expander is slot 1. The design of the GX76xx allows integration of PXI, cPCI, PXI Express and cPCI Express boards from any vendor.

The GX76xx PXI instrumentation chassis, when supplied with a controller (GX7600), comes with the GxChassis driver preinstalled. This driver supports the chassis' Smart functions and includes the monitoring of chassis slot temperatures, power supply voltage monitoring, monitoring/controlling of fan speed as well as programming/routing and driving/monitoring of the PXI trigger lines. The GxChassis driver provides a complete set of API (Application Programming Interface) calls for controlling all of the PXI chassis' Smart features as well as providing a soft front panel for interactive control and monitoring of the Smart features. The GxChassis driver can initiate alarm or shut down based on preset temperature limits. These limits can be saved to an on-board EEPROM which can then be used as default settings on power up. The driver can be download from Marvin Test Solutions web site and is included with the Marvin Test Solutions Product CD, which is supplied with every chassis and can be installed on an external PC controller for slave chassis applications.

The following figures show the GX7600 master chassis with a Marvin Test Solutions controller and the GX7610 slave chassis with a PXI expander that is typically connected and controlled by an external desktop PC or another PXIe chassis with a controller such as the GX7600.

.

Figure 2-1: GX7600 Instrumentation Chassis

Figure 2-2: GX7610 Instrumentation Chassis

GX76xx Features

The GX7600 and GX7610 models offer the following features:

- Full compliance with PXI Express Hardware Specification revision 1.0 and the accompanying ECN-1 revision 2.0. Supports features such as trigger bus, star trigger, differential triggers, local bus, and system clock.
- Interoperability with 32-bit, 33MHz, CompactPCI and Compact PCI Express.
- Front-loading configuration. Boards are inserted from the front for simplified maintenance.
- Two 79 CFM fans provide airflow for all plug-in instruments.
- One fan provides cooling for the power supply.
- A single power supply providing 560 watts of system power.
- PXI Express back plane which incorporates the local bus, trigger buses, and 10 & 100 MHz reference clocks.
- Support for external instrumentation and devices using the built-in USB and Ethernet (GX7600 only) interfaces. Support for additional interfaces such as IEEE-488 (GPIB) and RS-232 are available via peripheral cPCI cards.
- GxChassis software package to provide control over the chassis smart features and innovative PXI/PCI Explorer ™ software provides easy configuration tools for the chassis and instruments.

When bundled with ***ATEasy***™, Marvin Test Solutions' award-winning software development environment, the GX76XX provides a complete system for creating a variety of test and measurement applications.

The PXI Standard

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

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

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

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

PXI and PCI Express Overview

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

PXI Express leverages the electrical features defined by the widely adopted PCI Express specification for data movement. This is accomplished by PXI Express Modules complying with the CompactPCI Express specification, which combines the PCI Express electrical specification with rugged euro card mechanical packaging and high-performance differential connectors. This allows measurement and automation Systems based on PXI Express to have a data throughput of 4 GBytes/sec in each direction. PXI Express also offers two-way interoperability with CompactPCI Express products.

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

GX76xx PXI Express Features

The PXI Express (PXIe) and PCI Express (PCIe) features incorporated into the GX7600 and GX7610 represent a significant performance improvement compared to PXI. Key features of PXI Express include::

- Dramatic improvement in available PCI bus bandwidth from 132 MB/s to 4 GB/s, offering a 30X improvement in bandwidth while maintaining software and hardware compatibility with PXI modules.
- Preservation of compatibility with existing PXI based systems.
- Additional timing and synchronization features including a 100 MHz PXI clock signal and differential Star triggers.
- Point to point data bus architecture eliminates data bus sharing, allowing the implementation of data streaming applications for continuous data acquisition applications.
- 2 - 3U hybrid slots offer the flexibility to use PXI Express or PXI modules in the same slot.
- 5 3U PXI or cPCI slots
- 1 3U PXIe or cPCI Express slot
- Support for x1 or x4 PCI Express lane configurations

GX76xx Models

The GX76xx models are available in two configurations:

- GX7600: This chassis includes a DVD-RW and hard disk drive. The GX7600 is designed to operate with the GX7940 family of embedded controllers. Via a rear I/O connection, the controller interfaces to the chassis' peripheral devices and peripheral interfaces located on the rear panel of the GX7600, minimizing the number of connections to the controller's front panel.

  NOTE: The GX7940 controller is provided with documentation that describes its available connections and configuration separately.

- GX7610: This chassis is designed to operate with PXI Express bus expanders or with a MXI Express interface. This configuration allows the use of a external desktop PC with PCI Express or another PXI Express chassis as the system controller.

GX76xx Chassis Description – Front View

The GX7600 model is configured as a modular master 3U PXI Express chassis. Slot 1 contains a controller (in this case GX7940). Slot 2, 6, 7, 8, 9 are PXI slots, Slot 3 and 5 are hybrid slot and can accept PXI or PXIe instruments and slot 4 is a PXIe slot. Figure 2-3 shows the front view features.

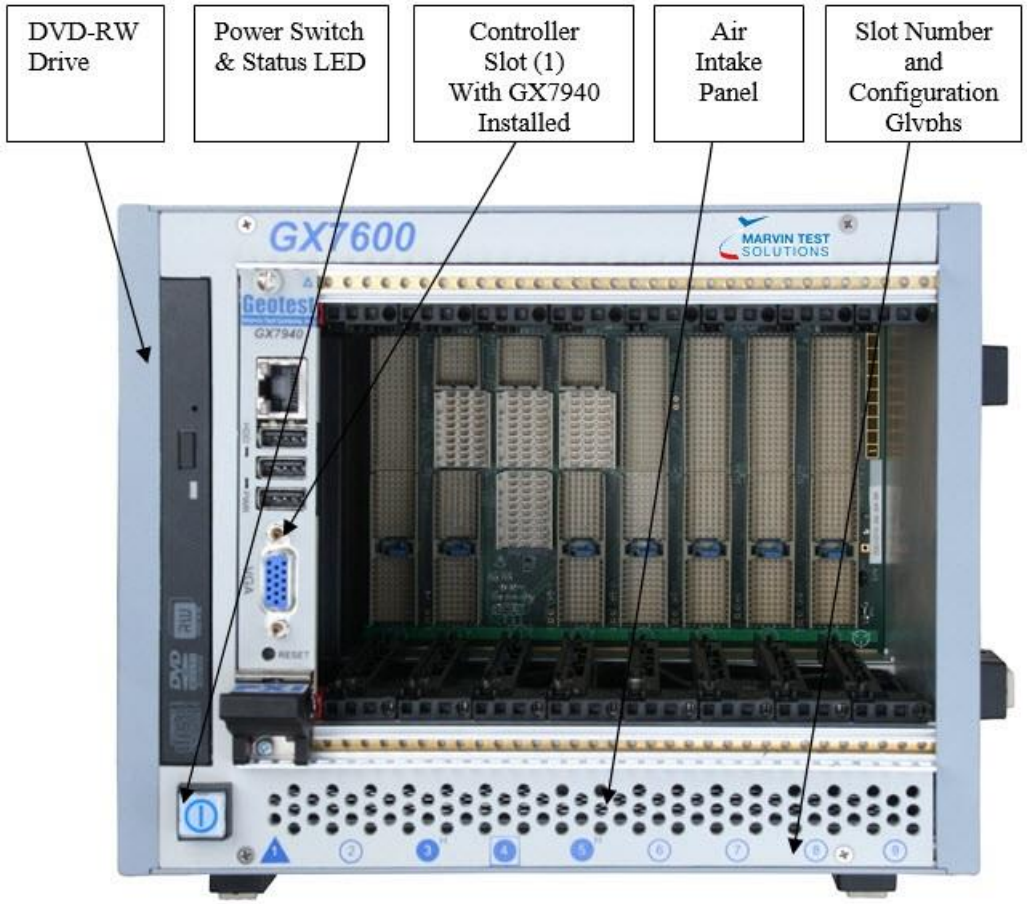

Figure 2-3: GX7600 with Controller Installed

The GX7610 model is configured as a modular slave 3U PXI Express chassis. Slot 1 contains PXIe expander usually connected to a desktop PC or to another PXIe master chassis such as the 7600. Slot 2, 6, 7, 8, 9 are PXI slots, Slot 3 and 5 are hybrid slot and can accept PXI or PXIe instruments and slot 4 is a PXIe slot. Figure 2-4 shows the front view features.

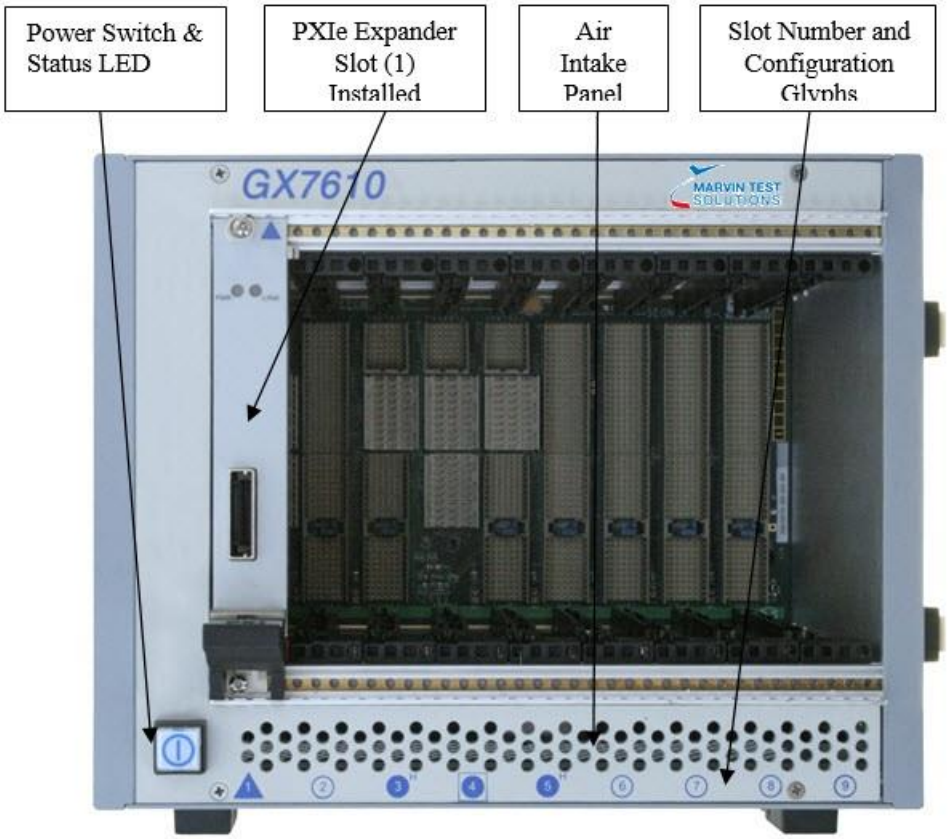

Figure 2-4: GX7610 Front View

The following paragraphs provide more information about the chassis front view callouts:

Power Switch

Momentary power control switch with an integrated green LED to denote power status. Shutting down the chassis power requires holding the button pressed down until the chassis turns off.

Slot Number and Glyph Designations

	Represents reserved PXIe controller slot (slot #1). The System slot is used for embedded or remote controllers. The GX7600 is designed to be compatible with the GX7940 controller which provides rear I/O connections to the built-in hard drive, DVD and rear panel I/O.
	Represents PXI slots, (slots #2, 6, 7, 8 and 9), any PXI or cPCI board can be installed in these slots.
	Represents PXIe Hybrid slots (#3 and 5), any PXIe, cPCIe or PXI/cPCI hybrid compatible board can be installed in these slots. Hybrid compatible PXI boards are cards that J2 does not exist or XJ4 connector is used (instead of J2).
	Represents PXIe System Timing slot (#4). Either a System Timing Module or any PXIe/cPCIe instrument can use the System Timing slot.

Air Intake Panel

This panel, located below the card cage, provides the intake for cooling the GX7600. **DO NOT BLOCK THIS PANEL.**

DVD-RW Drive

A DVD-RW Drive (GX7600 only) is available for use in conjunction with the GX7940 family of embedded controllers.

GX76xx Chassis Description – Rear View

Figure 2-5and Figure 2-6 shows the GX7600 and the GX7610 rear view. Both chassis have power switch, power outlet, variable fan with Fan Speed control that can be set to Auto or High position, 10 MHz clock BNC input and output that are used for system timing signals and distribution, exhaust panel for airflow and chassis ground connection (bottom right). The GX7600 also have Ethernet and USB ports.

Figure 2-5: GX7600 Rear View

Figure 2-6: GX7610 Rear View

The following table describes the rear view features of the GX76xx chassis:

Name	Description
Chassis Air Exhaust Panel	Chassis Air Exhaust Panel
Fan Speed Switch	Setting this switch to "Auto" will allow the system to control the fan speed automatically. The GxChassis driver can be used to control the system's fan speed. Setting this switch to "High" places the fans at full speed and can be used to manually override the system's control of the fans.
Power Switch	Main power switch
Input Power Receptacle	This receptacle connects to the power cord provided.
Power Supply Cooling Fan	Independent fan for power supply cooling.
Ethernet	A 10/100 BaseT Ethernet port
USB	USB port.
BNC Out	10 MHz signal out. Used for system timing signals and distribution.
BNC In	10 MHz signal in. Used for system timing signals and distribution.
Ground	Chassis Ground connection

Optional Equipment

Marvin Test Solutions offers a variety of products to use with your GX76xx chassis:

- PXI Express Embedded Controllers
- PXI Express Remote Controllers or expanders
- 3U PXI Express and PXI instruments and switches
- Blank panels for non-occupied slots

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

Gx76xx Boards

The GX7600/GX7610 supports several types of boards that can be inserted to the chassis. The following paragraphs describe the board's types and characteristics.

3U PXI and Compact PCI Boards

3U PXI can be installed in slots 2, 6, 7, 8 and 9 as shown in Figure 2-7.

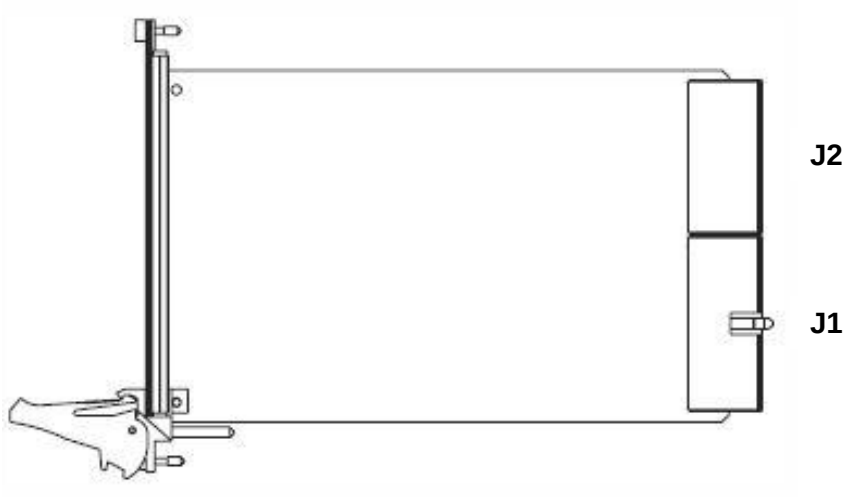

Figure 2-7: 3U PXI Boards

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

If J2 does not exist in the board, the card is considered Compact PCI board (cPCI) board as shown in Figure 2-8. 3U Compact PCI boards can be plugged to PXI Slot as well to 3U PXI Express Hybrid slot (slot 3 and 5 on the GX76xx chassis).

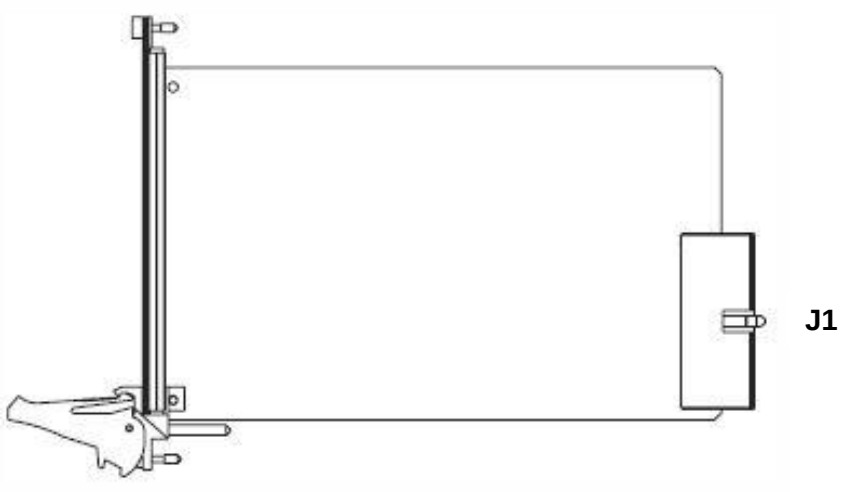

Figure 2-8: 3U Compact PCI Boards

3U PXI Hybrid Boards

PXI hybrid boards consist of a J1 connector and XJ4 connector as shown in Figure 2-9. The J1 connector supports the PCI bus signals and the XJ4 connector supports the PXI trigger bus resources. Slots 3 and 5 on the GX76xx support 3U PXI Express Hybrid Boards and 3U Compact PCI boards as shown in Figure 2-8.

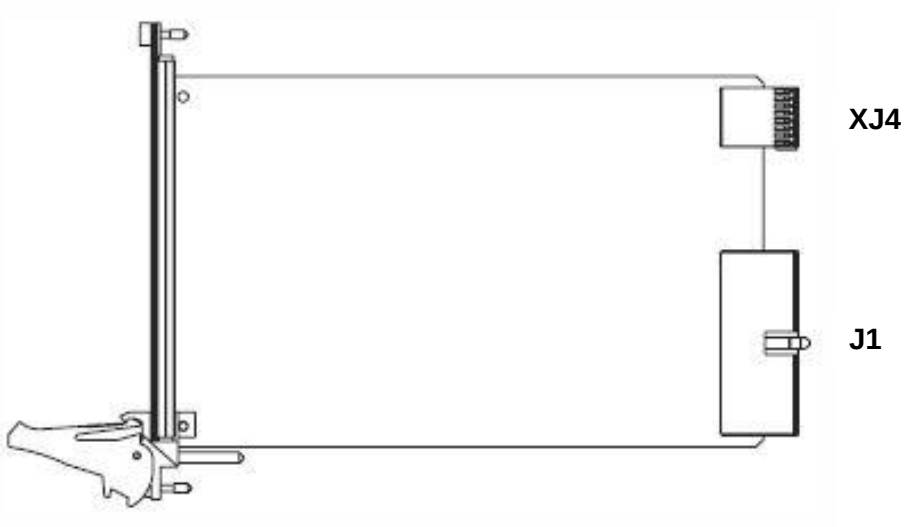

Figure 2-9: 3U PXI Hybrid Board

3U PXI Express Boards

3U PXI Express boards consist of an XJ4 and XJ3 connector as shown in Figure 2-10. The XJ4 connector supports power and the PXI trigger bus resources and the XJ3 connector supports the PCI Express bus connections and PXI Express trigger / clock resources. Slot 4 on the GX76XX supports a PXI Express board.

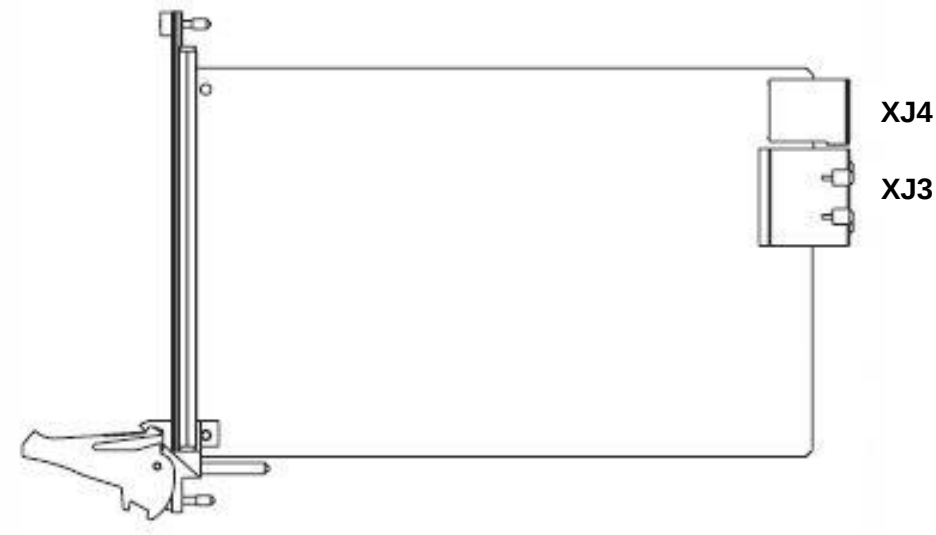

Figure 2-10: 3U PXI Express Board

3U PXI Express System Timing Boards

3U PXI Express System Timing boards is shown in Figure 2-11, similar to the 3U PXI board consist of an XJ4 and XJ3 connectors to provide power, reference clocks and triggering signals. In addition, TP2 and TP1 connectors support PXI express three differential star triggers to each of the chassis boards. Each one of the connectors support Slot 4 on the GX76XX supports a PXI Express board.

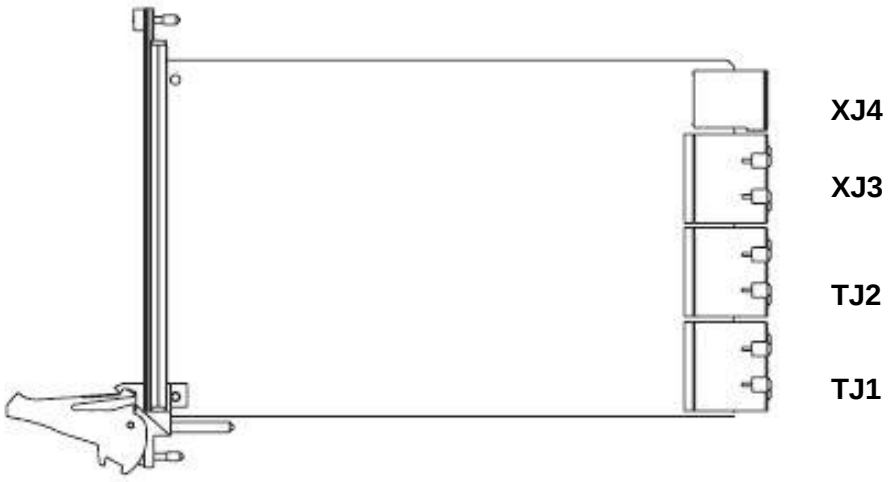

Figure 2-11: 3U PXI Express Timing Board

3U PXI Express System Boards

PXI Express System Board is shown in Figure 2-12. The GX76xx System Slot is slot 1. The GX7600 master chassis PXI Express System Board is a CPU module that contains all the controller ports including VGA, Ethernet, and USB ports. The GT7610 slave chassis will accept a PXI Express System Board (bus expander) that can be connected to an external desktop PC or to another PXI Express chassis such as the GX7600.

	Check with Marvin Test Solutions support before inserting PXI Express system board/controller that is not supplied by Marvin Test Solutions specifically for the GX7600. Third party PXI Express System Controllers boards may not be compatible with the GX7600 System Slot connections and may damage your chassis.

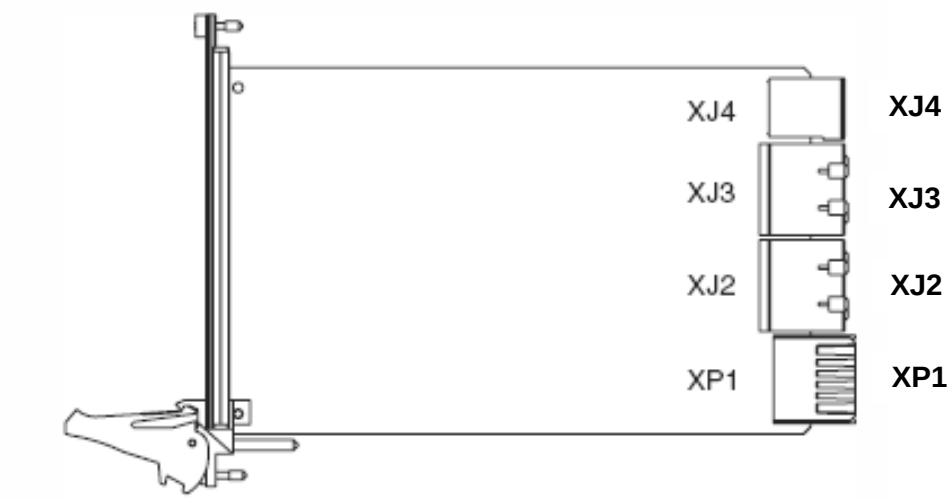

Figure 2-12: PXI Express System Board

PXI Bus Segments

The GX7600's backplane is divided into two bus segments (see Figure 2-10), which are interconnected using PCI-PCI bridge technology. The PCI bridge device presents one PCI load on each of the bus segments. The left bus segment supports the System Slot, 1 PXI slot and 2 Hybrid slots. The second segment supports PXI slots 6 to 9. The PCI Express connections (slots 3-5) and the PCIe-PCI Bridge are connected directly to the System Controller via PCI Express lanes.

System Controller Slot

The System Controller slot is located in slot #1 of the chassis and occupies a single slot width. Slot numbers are clearly labeled below each slot where slot #1 is the left-most slot and slot #9 is the right most. The GX76XX can accept 3U embedded controllers that are 1-slot wide. The system controller slot supports (4) x1 or (4) x4 PCI Express lane configurations.

PXI Express System Timing Slot

Slot 4 is the PXI Express System Timing slot (4th from the left). This slot has a dedicated star trigger line going to slots 2 through 9 and dedicated differential star trigger lines going to slots 3-5. The Star Trigger is used to synchronize between 8 instruments and it utilizes back plane traces that are of equal length, providing for a skew of less than 1nSec between slots. The differential star trigger lines are used to send high-speed clocks and triggers between the System Timing Module (Slot 4) and Express Peripheral Modules (slots 3 and 5). If you do not need a System Timing Module, any PXIe or cPCIe instrument can be used in this slot. See Figure 2-14 in this chapter for more information about the available trigger architectures.

Local Bus

The PXI local bus is a daisy-chained bus connecting peripheral slots in the same bus segment. The local bus in segment A is 1 user-defined line wide and the local bus in segment B is 13 lines wide, as defined in the PXI Express Specification. Each local bus can be used to pass analog or digital signals between two adjacent modules or to support a high-speed side-band digital communication path that is independent of the PXI of PXI Express bus bandwidth.

Local bus signals can support voltages from 0 to 42V DC and up to 200 mA DC of current.

Figure 2-13 schematically shows the GX7600 / GX7610's local bus configuration.

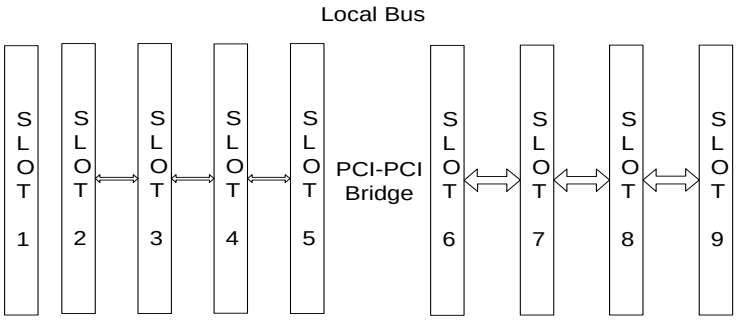

Figure 2-13: PXI Local Bus Routing

Trigger Bus

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

The PXI trigger bus typically provides connectivity only within a single bus segment preserving the high performance characteristics of the trigger bus within a segment. The GX7600 backplane partitions instruments into logical groups as show in Figure 2-14. A feature of the GX7600 is its ability to programmatically connect, set direction or isolate trigger lines between segments A and B. This feature is controlled by the GxChassis driver.

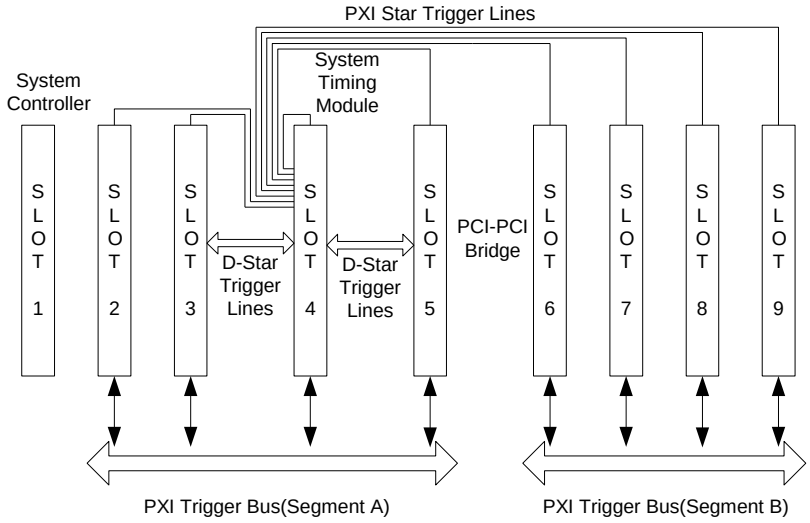

Figure 2-14: PXI Trigger Architecture

Star Trigger Lines

Eight PXI Star Trigger lines are connected to slots 2 to 9 and are sourced from the system time slot (slot 4). The PXI Star Trigger lines can be used to synchronize the operation of several different PXI peripheral modules.

The PXI Star Trigger Bus offers high performance synchronization features to users of PXI systems. The Star Trigger Bus implements a dedicated trigger line between the System Timing Slot (Slot #4) and the other peripheral slots. A System Timing Module can be installed in this slot and can be used to provide very precise trigger signals to other peripheral modules. Systems that do not require this advanced trigger can install any standard express peripheral module in this slot. Through the required use of line-length equalization techniques for routing the star triggers, PXI systems can meet demanding triggering requirements for which bused triggers are not appropriate. Note that the star trigger can be used to communicate information back to the star trigger controller, as in the case of reporting a slot's status, as well as responding to information provided by the star trigger controller.

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

Three sets of three PXIe Differential Star Trigger Lines are connected to slots 3 to 5. The Differential Start Trigger (DSTAR) Lines can be used to pass clocks and triggers between the System Timing Slot (Slot #4) and the peripheral slots. The first DSTAR line, DSTARA, is a fast-switching LVPECL clock, generated by the System Timing Module, for precise timing. The second DSTAR line, DSTARB, is a fast-switching LVDS line carrying clock or trigger signals from the System Timing Module to the peripheral module. The third DSTAR line, DSTARC, is a fast-switching LVDS line carrying clock or trigger signals from the peripheral module to the System Timing Module. All lines are length matched.

System Reference Clocks

The PXIe 100MHz differential reference clock is distributed to the Hybrid Slots (3 and 5) and the System Timing Slot (4). A differential signal, PXIe_SYNC100, is also distributed to the Hybrid Slots (3 and 5) and the System Timing Slot (4). The PXIe_SYNC100 signal is derived from the PXIe 100 MHz differential reference clock and has the default behavior of pulsing at 10MHz, with a pulse width of approximately 5ns. The pulse rate can be slower than 10 MHz or non-periodic, allowing events to be synchronized between chassis that may be located far apart. The PXI 10 MHz system clock (PXI_CLK10) is derived from the 100 MHz differential reference clock and is distributed to all slots of the GX7600. These common reference clocks can be used for synchronization of multiple instruments in a measurement or control systems. The internal 10 MHz clock is also available on the GX76XX's rear panel via a BNC connector.

The GX76XX has the ability to synchronize to an external 10MHz source providing a more stable/accurate system clock. The external 10MHz clock can be provided through the System Timing Slot or through the BNC connector on the back of the GX76XX.

System Power Supply

A single power supply provides system power to all slots of the GX76XX. A total power of 560 watts is available.

Power Distribution

The GX76XX meets or exceeds the requirements of the PXI specifications regarding the power provided to each slot. Table 2-1 lists the power per slot required by the PXI specification:

Slot \ Voltage	5V	3.3V	+12V	-12V	5Vaux
PXIe System Slot, no expansion slots	1A	3A	2A	N/A	1A
PXI Instrument Slot	2A	2A	0.5A	0.25A	N/A
PXI Hybrid Slot	2A	3A	2A	0.25A	0A
PXIe Slot/System Timing Slot	N/A	3A	2A	N/A	0A
Minimum required current for a GX76XX configuration	15A	22A	10.5A	1.75A	1.5A*

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

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

The maximum current provided by the GX7600 and GX7610 for any PXI slot is listed in the Table 2-3.

Slot \ Voltage	VIO (5V)	5V	3.3V	+12V	-12V	5Vaux
PXIe System Slot	0A	15A	15A	30A	0A	1A
PXI Instrument Slot	11A	6A	6A	1A	1A	0A
PXI Hybrid Slot	5A	6A	6A	4A	1A	1A
PXIe Slot/System Timing Slot	0A	0A	6A	4A	0A	1A
Total available current for the GX76XX	60A*	60A*	40A*	32A	3A	2A

*60A total of the 5V and 3.3V current. Total power cannot exceed 560 watts.

Table 2-3: Available GX7600 / GX7610 Power

GxChassis Driver Features

The GxChassis driver has the following features:

- Program the PXI chassis' over-temperature shutdown level.
- Program the PXI chassis' over-temperature alarm level.
- Control and monitor system fans
- 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 (30 A max.), +5V (50 A max.), +12V (27 A max.), -12V (3 A max.).
- 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 between segments.
- Selectable temperature scale.
- Save settings to an on-board EEPROM to be used as defaults.
- API calls controlling all of the PXI chassis' capabilities.
- Front panel control of all of the PXI chassis' capabilities.

Virtual Panel Description

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

To open the virtual panel application, select **GxChassis Panel** from the **Marvin Test Solutions**, **GxChassis** menu under the **Start** menu. The GxChassis virtual panel opens as shown in Figure 2-9:

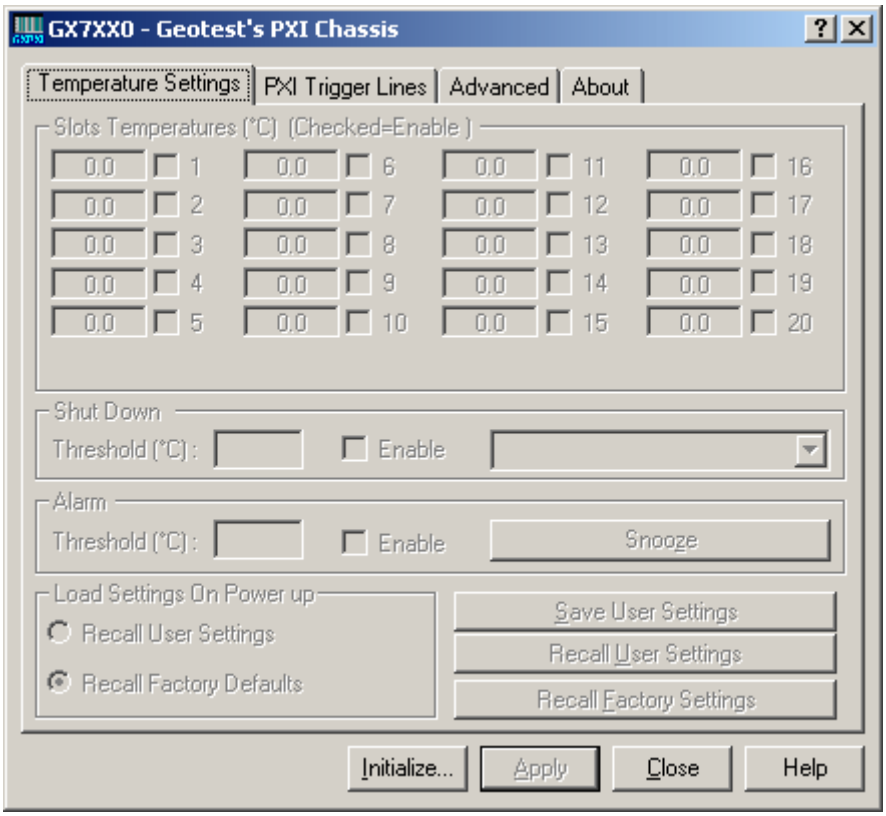

Figure 2-15: GxChassis Virtual Panel – Temperature Settings (not Initialized)

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.

Marvin Test Solutions' HW (Figure 2-16):

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 2-16: Initialize Dialog Box using Marvin Test Solutions' HW driver

Virtual Panel Temperature Settings

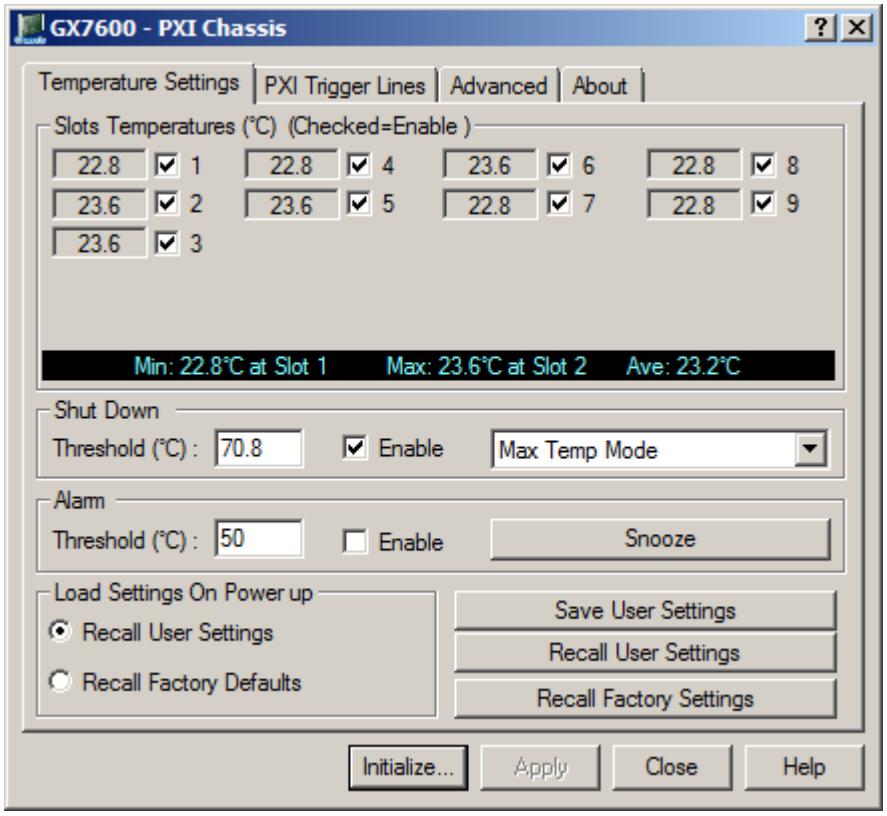

Figure 2-17: GxChassis Virtual Panel – Temperature Settings

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 affect the alarm or shutdown threshold limits.

Shut Down (Group Box)

Threshold (Edit box): Sets/displays the shutdown Temperature for 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 operation mode. The temperature threshold operation mode dictates how the alarm and shutdown thresholds will be activated. When set to Max Temp Mode, the shutdown and alarm temperature conditions will be activated when any of the active slots' temperature are 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 is set to On) both backplane buzzers will beep simultaneously in intervals of 10 seconds.

Enable (Button): Checking this box enables the threshold temperature alarm at which point, the alarm will turn on.

Snooze (Button): Checking this box will 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 those settings which 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 2-18:

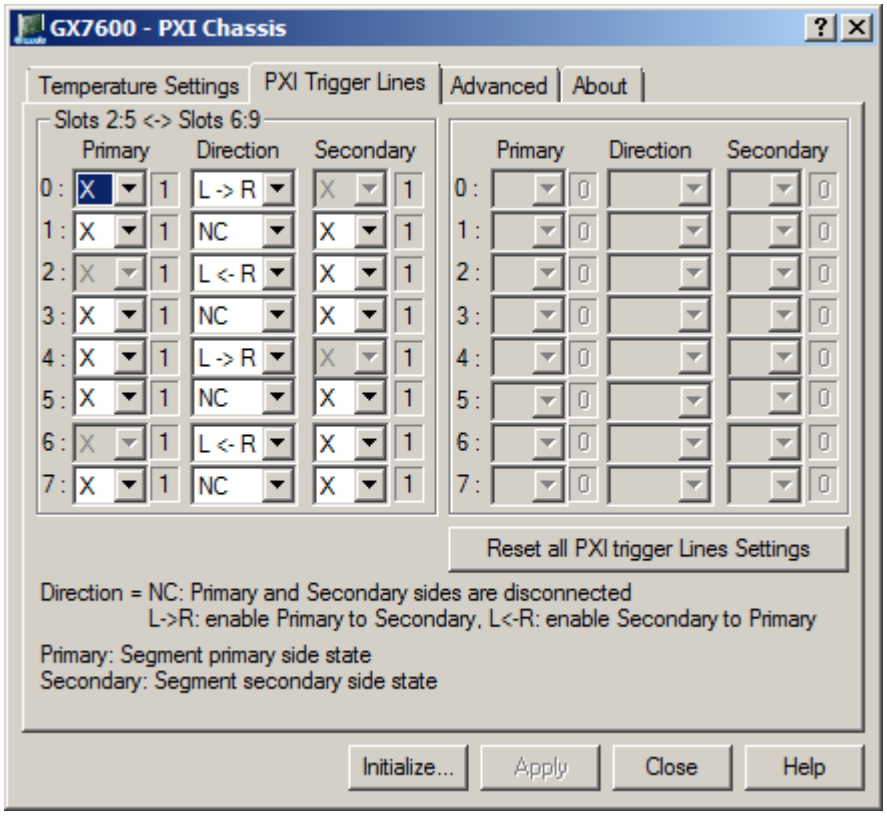

Figure 2-18: GxChassis Virtual Panel – Pxi Trigger Lines

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

Slots 2:7 <-> Slots 8:13 and Slots 8:13 <-> Slots 14:20 (Group Box)

Primary (Dropdown List): If enabled (depending on the Direction settings) sets/displays the specified PXI trigger line's primary side state: X=Monitor, 0=Driving low, 1=Driving high.

Direction (Dropdown List): Sets/displays the specified PXI trigger line direction, NC= Primary side is disconnected from Secondary side. L->R = Set trigger direction from Primary side to the Secondary side, L<-R = Set trigger direction from Secondary side to the Primary side.

Secondary (Dropdown List): If enabled (depending on the Direction settings) sets/displays the specified PXI trigger line's secondary side state: X=Monitor, 0=Driving low, 1=Driving high.

Virtual Panel Advanced page

Clicking on the **Advanced** tab will show the **Advanced page** as shown in Figure 2-19:

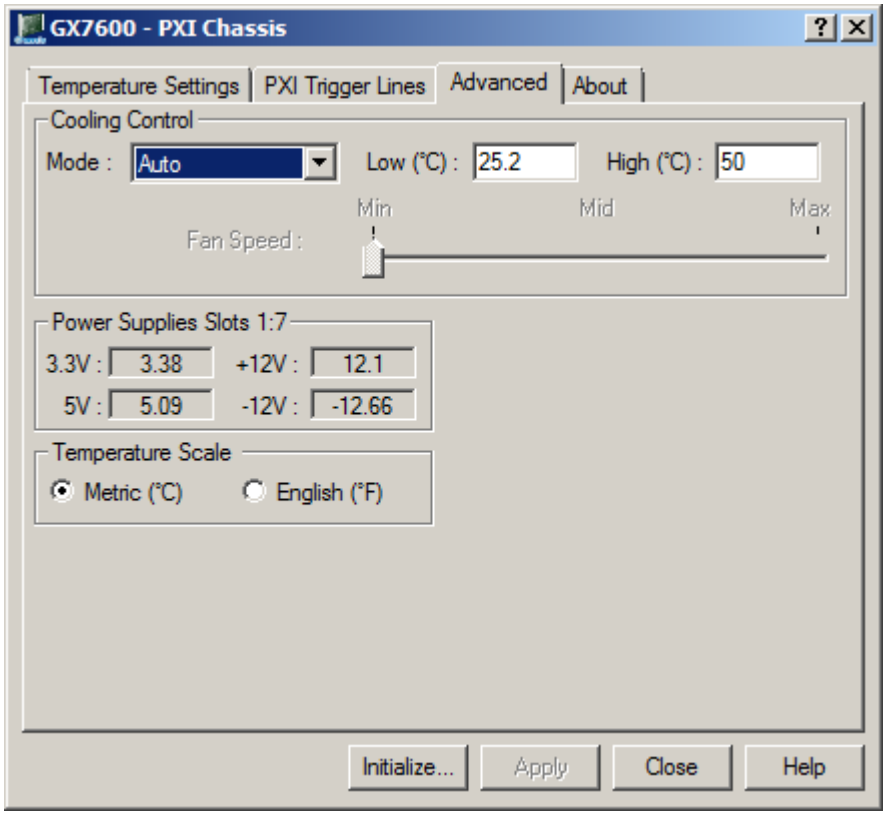

Figure 2-19: GxChassis Virtual Panel – Advanced page

The following controls are shown in the Advanced tab:

Cooling Control (Group Box)

Mode (dropdown list): Set/displays the fan's operational mode: Auto: Chassis sets the fan speed automatically, User Defined: User sets the Fan speed using the slider.

Fan Speed (Slider): If in Auto mode, the fan speed will be displayed. If in User Defined mode, the user can set the fan speed (slider enabled).

Low: Sets/displays the low threshold temperature when in Auto mode.

High: Sets/displays the high threshold temperature when in Auto mode.

Power Supplies Slots 1:10 (Group Box)

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

Power Supplies Slots 11:20 (Group Box)

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

Temperature Scale (Group Box)

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

Virtual Panel About Page

Clicking on the **About** tab will show the **About page** as shown in Figure 2-20:

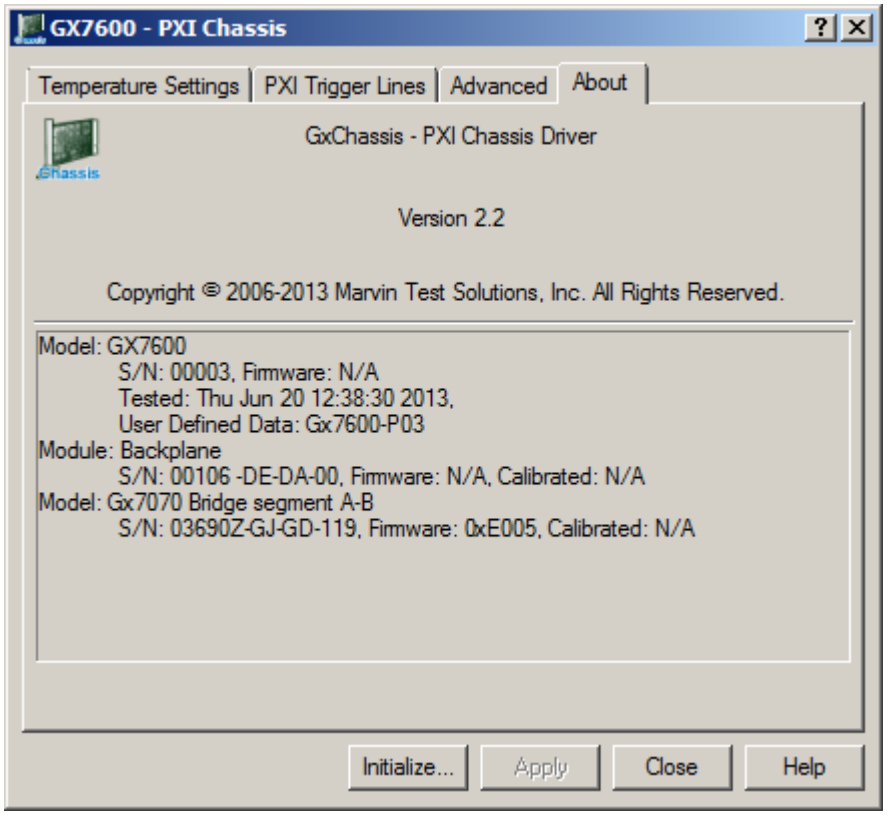

Figure 2-20: GxChassis Virtual Panel – About Page

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 GX7600 and GX7610 chassis and boards.

Unpacking and Inspecting the Chassis

1. Before unpacking the chassis, 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

Follow the appropriate installation instructions for your GX7600 or GX7610.

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

Line Voltage Selection and Fuse Values

The voltage selection of the GX7600and GX7610 is automatic. The chassis power supplies will automatically select the input voltage. The fuse should be correct for operation in your geographical region. Check the voltage/current requirements on the chassis' rear panel decal and make sure you have the correct fuse type. The GX7600 or GX7610 can operate with line voltages of 110 to 240 VAC.

GX7600 and GX7610 Chassis Installation

Follow these steps to install the chassis; more information regarding these steps is described later in this chapter:

1. Place the chassis on a sturdy, level surface. Leave space behind the chassis for ventilation.
2. Connect the power cable to the chassis and an outlet.
3. Install an embedded controller (master configuration, GX7600) or a remote controller (slave configuration, GX7610) to slot #1 if not installed.
4. Turn on the chassis power and the optional external system (for slave installation turn on the sale first)
5. Install the **GxChassis** software
6. Configure your system using the **PXI/PCI Explorer** applet.
7. Install any additional drivers for PXI instruments.
8. Turn off the system.
9. Install PXI modules into the chassis as described in the next procedure.

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

GX7600 Master and GX7610 Slave Configurations

The chassis is provided with two main configurations:

1. GX7600 – master configuration, slot 1 in this case contains embedded controller such as GX7940. The embedded controller contains a single slot computer with CPU, Memory, USB and Ethernet ports, hard drive and DVD/CD-RW drive.
2. GX7610 – slave configuration, slot 1 contains PXI Express expander card connected to an external PC or another PXI Express chassis such as GX7600. The controller resides on the external PC/chassis. Multiple slave chassis can be connected to the controller in a start or daisy chained configuration.

Figure 3-1 outlines a slave GX7610 (slave) configuration with a desktop PC (master) being the system controller.

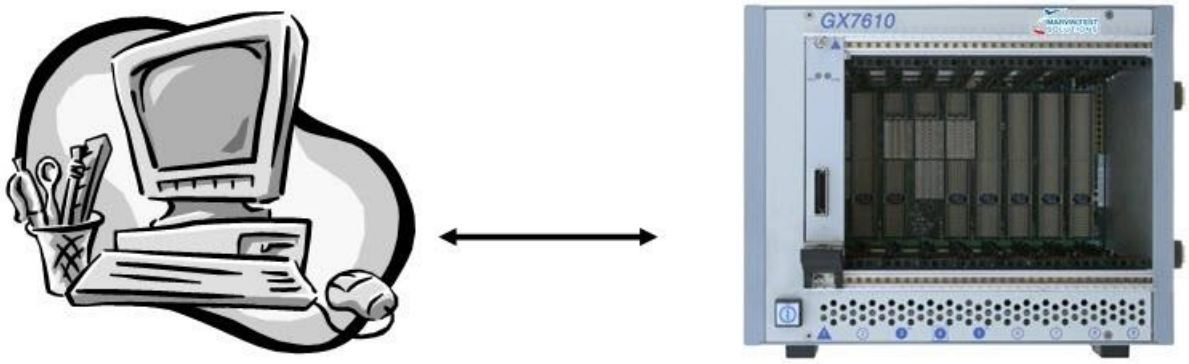

Figure 3-1: Slave GX-7600 Configuration with a Desktop PC

Figure 3-2 shows GX7600 chassis (master) connected to a GX7610 chassis (slave).

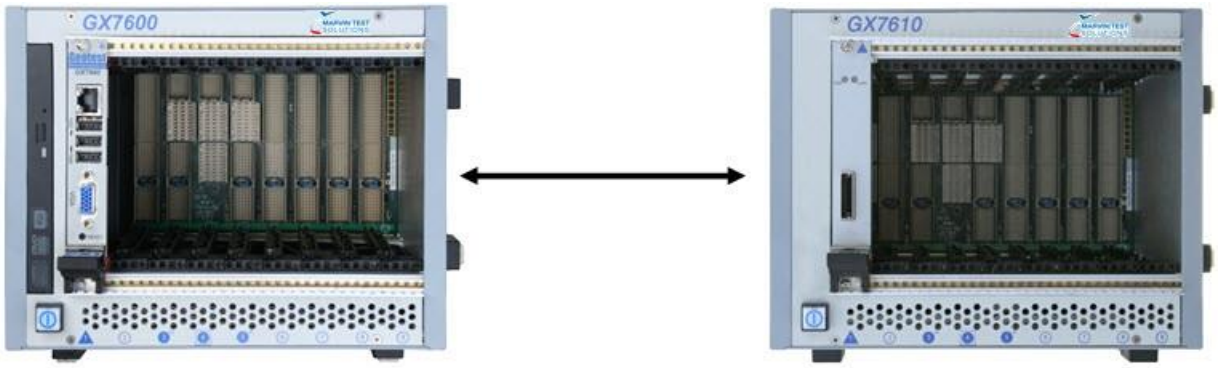

Figure 3-2: Master-Slave, GX7600 to GX7610 Configuration with Two PXI Express Chassis

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 GX7600 master configuration or on the external PC or the chassis where the controller reside (for GX7610 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 you 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 NT/2000/XP/VISTA, you may be required to restart the setup after logging-in as a user with an Administrator privileges. This is required in-order to upgrade your system with newer Windows components and to install kernel-mode device drivers (HW.SYS and HWDEVICE.SYS) required by the GxChassis driver to access resources on your chassis.

3. The first setup screen to appear is the Welcome screen. Click **Next** to continue.
4. Enter the folder where GxChassis is to be installed. Either click **Browse** to set up a new folder, or click **Next** to accept the default entry of `C:\Program Files\Marvin Test Solutions\GxChassis`.
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.

Overview of the GxChassis Software

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

- **PXI/PCI Explorer applet** – use to configure the PXI chassis, controllers and devices. This is required for accurate identification of your PXI instruments later on when installed in your system. The applet configuration is saved to PXISYS.ini and PXIeSYS.ini that are used by Marvin Test Solutions instruments, the VISA provider and VISA based instruments drivers. In addition, the applet can be used to assign chassis numbers, Legacy Slot numbers and instruments alias names.
- **VISA** is a standard maintained by the VXI Plug & Play System Alliance and the PXI Systems Alliance organizations (http://www.vxipnp.org/, http://www.pxisa.org/). VISA provides a standard way for instrument manufacturers and users to write and use instruments drivers. The VISA resource managers such as National Instruments **Measurement & Automation** (NI-MAX) can display and configure instruments and their address (similar to Marvin Test Solutions' PXI/PCI Explorer).
- **GxChassis Panel** – use to configure the smart chassis features includes over-temperature behavior, control the system fans, measure slot temperature and system power supply usage and program trigger lines direction and connection between PXI bus segments.
- **GxChassis driver** - a DLL (GxChassis.DLL located in the Windows System folder) used to program and control the board.
- **Programming files and examples** – interface files and libraries for various programming tools, see later in this chapter for a complete list of files and development tools supported by the driver.
- **Documentation** – On-Line help and User's Guide.

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).

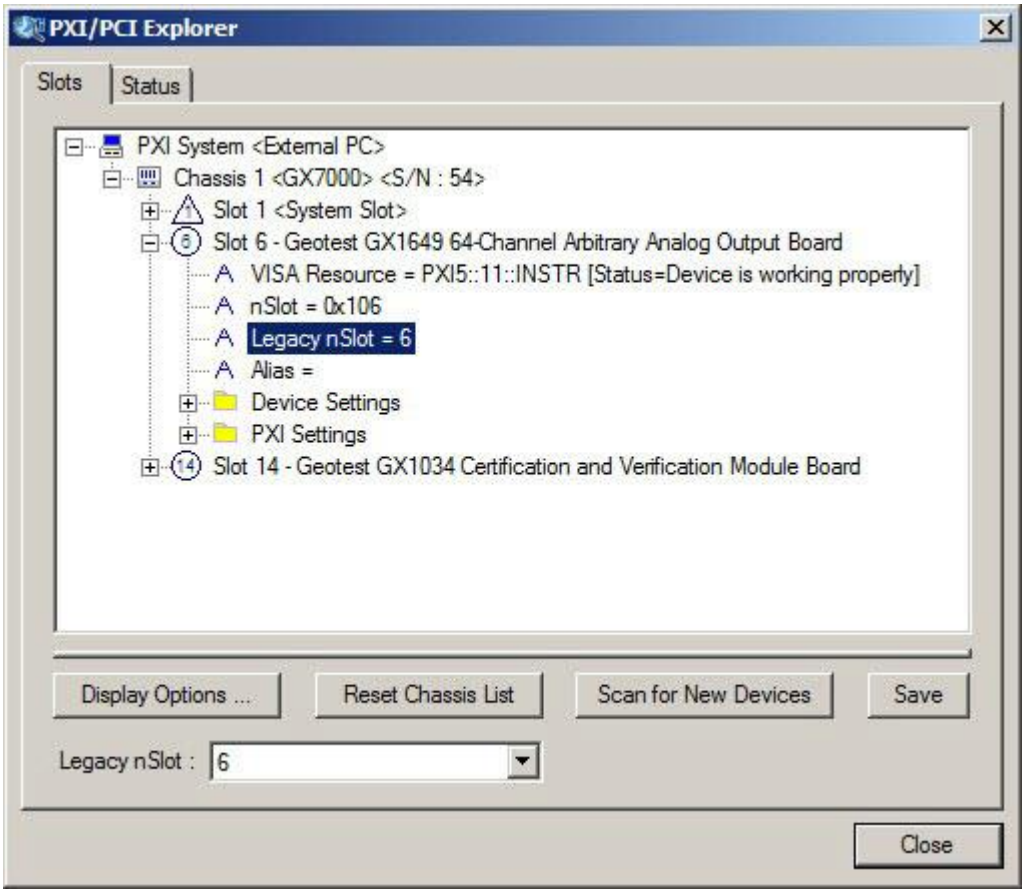

Figure 3-3: PXI/PCI Explorer

GX7600 and GX7610 PXI Instrument Installation

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

1. Before installing any PXI board, install the GxChassis software package. 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 GX76XX and other PXI module.

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

4. Locate an empty PXI/Hybrid/Express slot on the chassis, dependant on board type.
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 3-3).

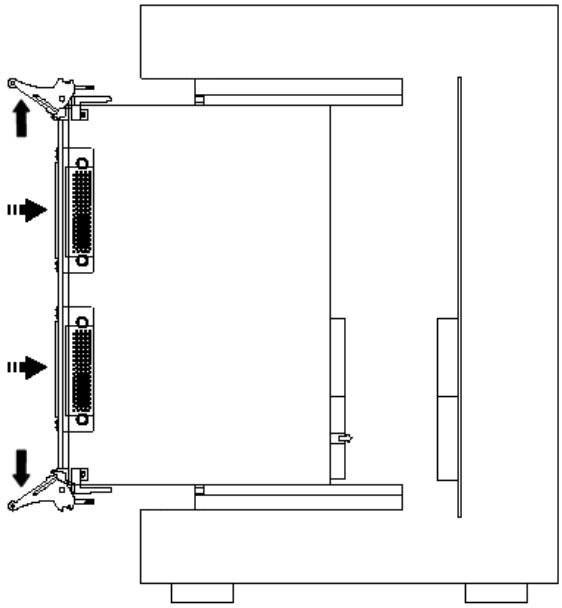

Figure 3-4: Ejector Handles Position During Module Insertion

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

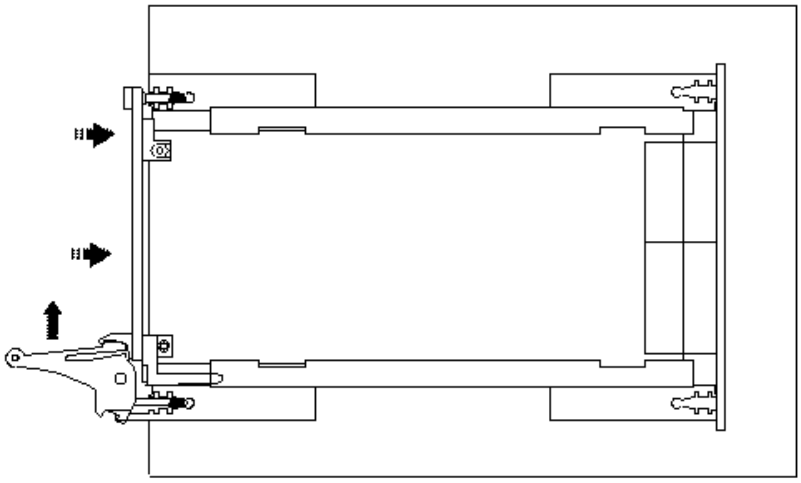

Figure 3-5: Ejector Handles Position After Module Insertion

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 Instrument Removal

Remove a PXI instrument board as follows:

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

Using External Instruments

Your GX76XX chassis supports all PXI and cPCI instruments and provides interfaces to USB or Ethernet devices. In some cases however, you may need to connect additional instruments to the GX7600. These additional instruments are typically GPIB (IEEE-488) or VXI. To use external instruments, you will need a Plug-in PXI module that provides an interface to GPIB or to VXI (MXI-2). Such interfaces are available from numerous vendors.

Installation Directories

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

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

Name	Purpose / Contents
…\Marvin Test Solutions\GxChassis	The GxChassis directory. Contains panel programs, programming libraries, interface files and examples, on-line help files and other documentation.
…\Marvin Test Solutions\HW	HW device driver. Provides access to your board hardware resources such as memory, IO ports and PCI board configuration. See the README.TXT located in this directory for more information.
…\ATEasy\Drivers	ATEasy drivers directory. GxChassis Driver and examples are copied to this directory only if ***ATEasy*** is installed on your machine.
…\Windows\System (Windows 9x/Me), or …\Windows\System32 when running Windows NT/2000/XP/Vista	Windows System directory. Contains the GxChassis DLL driver and some upgraded system components, such as the HTML help viewer, etc.

Driver Files Description

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

Driver File and Virtual Panel

- GxChassis.dll - 32-Bit MS-Windows DLL for applications running under Windows, 95, 98, Me, NT, 2000, XP or Vista.
- GxChassisPanel.exe – An instrument front panel program for all GxChassis supported boards.

Interface Files

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

- GxChassis.h - header file for accessing the DLL functions using the C/C++ programming language. The header file compatible with the following 32 bit development tools:
  - Microsoft Visual C++, Microsoft Visual C++ .NET
  - Borland C++
- GxChassis.LIB - Import library for GxChassis.dll (used when linking C/C++ application that uses GxChassis.dll).
- GxChassisBC.LIB - Import library for GxChassis.dll (used when linking Borland C/C++ application that uses GxChassis.dll).
- GxChassis.pas - interface file to support Borland Pascal or Borland Delphi.
- GxChassis.bas - Supports Microsoft Visual Basic 4.0, 5.0 and 6.0.

- GxChassis.vb - Supports Microsoft Visual Basic .NET.
- GxChassis.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 board, programming reference and panel operation.

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

ReadMe File

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

Program Example

The sample program includes a C/C++ sample compiled with various development tools, a Visual Basic example and an ATEasy example.

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 Pascal/Delphi

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

Programming GxChassis Boards Using ATEasy ®

The GxChassis package is supplied with an ATEasy driver. The ATEasy driver uses the GxChassis.dll to program the chassis' functions. The ATEasy driver includes an example that contains a program and a system file for use with the ATEasy driver. Plain language commands declared in the ATEasy driver are easier to use than using the DLL functions directly. The driver commands will also generate exception that allows the ATEasy application to trap errors without checking the status code returned by the DLL function after each function call.

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

- The *nHandle* parameter is omitted. The driver handles this parameter automatically. ATEasy uses driver logical names instead i.e. GxChassis1, GxChassis2 for GxChassis.
- 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.

Virtual Front Panel Window

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 which can be installed with the deployed system controller / chassis.

Sample Programs

The following example demonstrates how to program the board using the C programming language under Windows. The example shows how to get or set a group or channel voltage.

To run enter the following command line parameters:

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

Sample Program Listing

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

      FILE        : GxChassisExampleC.cpp

      PURPOSE           : WIN32/LINUX example program for GX7xxx chassis
                          using the GXCNT driver.

      CREATED           : Dec 2005

      COPYRIGHT   : Copyright 2002-2013, Marvin Test Solutions, Inc.

      COMMENTS    :

      To compile the example:

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

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

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

*****************************************************************************/
#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 - Functions Reference

Introduction

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

All function parameter syntax follows the same rules:

- Strings are ASCIIZ (null or zero character terminated).
- The first parameter of most functions is *nHandle* (16-bit integer). This parameter is required for accessing the chassis and is returned by the **GxChassisInitialize** function. The *nHandle* is used to identify the chassis when calling a function for programming and controlling the operation of the chassis. .
- All functions return a status with the last parameter named *pnStatus*. The *pnStatus* is zero if the function was successful, or less than zero for error conditions. The description of the error is available using the **GxChassisGetErrorString** function or by using a predefined constant, defined in the driver interface files: GxChassis.h, GxChassis.bas, GxChassis.pas or GxChassis.drv.
- Parameter names are prefixed as follows:

Prefix	Type	Example
a	Array, prefix this before the simple type.	*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 four power supplies voltages.
GxChassisGetPxiTriggerLine	Returns the specified PXI trigger line bridge direction mode and its direction configuration (left or right).
GxChassisGetPxiTriggerLineLevel	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.
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 **GX76XX**mainframe Chassis.
pnMode	PSHORT	Alarm mode is one of the following: 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 the backplane buzzer 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 **GX76XX** mainframe 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 **GX76XX** mainframe 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: GX7600, 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[128];
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
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 four power supplies voltages.

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX76XX** mainframe 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[8];
GxChassisGetPowerSuppliesVoltages (nHandle, adVoltage, &nStatus);
```

See Also

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

GxChassisGetFanSpeed

Purpose

Returns the fan speed and control settings.

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX76XX** mainframe 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 bridgeboard revisions G and above.

Example

The following example returns fan speed and control settings:

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

See Also

GxChassisSetFanSpeed, **GxChassisGetFanThresholdTemperatures, GxChassisGetErrorString**

GxChassisGetFanThresholdTemperatures

Purpose

Returns the fan low and high threshold temperatures.

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX76XX** mainframe 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 bridgeboard revisions G and above.

Example

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

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

See Also

GxChassisSetFanThresholdTemperatures, GxChassisSetFanSpeed,, GxChassisGetErrorString

GxChassisGetPxiTriggerLine

Purpose

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

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX73XXA/GX73XXA** mainframe 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: 0. GXCHASSIS_SEGMENT_0_TO_SEGMENT_1 – Segment Slots 2:7 connecting to Segment Slots 8:13 (chassis left side bridge).
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 **GX76XX** mainframe 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: 0. GXCHASSIS_SEGMENT_0_TO_SEGMENT_1 – Segment Slots 2:7 connecting to Segment Slots 8:13 (chassis left side bridge).
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 bridgeboard revisions G and above.

Example

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

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

See Also

GxChassisSetPxiTriggerLine, **GxChassisGetErrorString**

GxChassisGetShutdownTemperature

Purpose

Returns the shutdown temperature and shutdown state.

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX76XX** mainframe 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 **GX76XX** mainframe 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

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 9 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 **GX76XX** mainframe Chassis.
pdwStates	PDWORD	Returns slots with active or enabled temperature monitoring, bits 0 through 8 represents slots 1 through 9. • 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 **GX76XX** mainframe 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.

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 **GX76XX** mainframe Chassis.
nSlot	SHORT	Specified slot temperature. Slot number can be from 1 to 9.
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

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 **GX76XX** mainframe 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 **GX76XX** mainframe 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 GX7600 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 **GX76XX** mainframe 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 **GX76XX** mainframe 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 **GX76XX** mainframe 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 **GX76XX** mainframe Chassis.
nMode	SHORT	Over Temperature Alarm mode is one of the following: 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) backplane buzzer will beep 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 **GX76XX** mainframe 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**

GxChassisGetFanSpeed

Purpose

Sets the fan speed and control settings.

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX76XX** mainframe 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 bridgeboard revisions G and above.

Example

The following example sets the fan speed to Auto:

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

See Also

GxChassisGetFanSpeed, **GxChassisSetFanThresholdTemperatures, GxChassisGetErrorString**

GxChassisSetFanThresholdTemperatures

Purpose

Sets the fan low and high threshold temperatures.

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX76XX** mainframe 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 bridgeboard revisions G and above.

Example

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

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

See Also

GxChassisGetFanThresholdTemperatures, GxChassisSetFanSpeed,, GxChassisGetErrorString

GxChassisSetPxiTriggerLine

Purpose

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

Syntax

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

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a **GX76XX** mainframe 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: 0. GXCHASSIS_SEGMENT_0_TO_SEGMENT_1 – Segment Slots 2:7 connecting to Segment Slots 8:13 (chassis left side bridge).
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).

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 **GX76XX** mainframe 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 Over 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, **GxChassisGetOverTemperatureAlarmThreshold, GxChassisSetShutdownTemperature**, **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 **GX76XX** mainframe Chassis.
dwStates	DWORD	Defines slots that will be actively monitored for temperature. Bits 0 through 8 represents slots 1 through 9. • 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 **GX76XX** mainframe 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 **GX76XX** mainframe 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 are 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 – GX7600/GX7610 System Specifications

AC Input Power

110VAC to 240 VAC, 47 –63 Hz

10 A max, (PFC)

Input AC power is filtered using a line filter.

Power Supply

One 560W DC power supply providing:

+5 VDC @ 60 (Max)

+3.3 VDC @ 40A (Max)

+12 VDC @ 32A (Max)

-12 VDC @ 3A (Max)

+5Vaux @ 2A (Max)

Total power cannot exceed 560 watts.

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	- 3 %	+ 5 %	50	mV	50	mV
2	+5.0V	2.5 A	60.0 A	- 3 %	+ 5 %	50	mV	50	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	- 5 %	+ 5 %	150	mV	200	mV
5	SB +5.0V	0.0A	2.0A	-5%	+5%	100	mV	100	mV

*****Noise Bandwidth: DC – 20 MHz

Cooling

Two 79 CFM system fans mounted below the slots. One 50 CFM fan for system power supply cooling. System fan speed can be controlled and monitored via GxChassis driver software. Fans operate in automatic or hi-speed mode via a selector switch on the GX76XX's rear panel.

Temperature Monitoring

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

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

Power Supply Monitoring

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

Accuracy: +/- 2% of reading

PXI Clocks

Integrated 10 and 100 MHz system clocks with auto-detect function. Presence of an external 10 MHz PXI clock will cause the 100 MHz PXIe clock to synchronize to the external 10 MHz clock source.

100 MHz clock accuracy: +/- 30 ppm

External 10 MHz Clock Input

An external 10 MHz clock source (TTL) can be provided via a rear panel BNC or via the PXI Express System Timing Controller

10 MHz Clock Output

10 MHz clock is available via a rear panel BNC connector, TTL compatible level

Slots

9 slots (8 instrument slots max):

- 1 System Controller Slot
- 1 System Timing Slot with Star and DSTAR Triggers (can be used by any PXIe/cPCIe instrument)
- 5 PXI Slots with Star Trigger (can be used by any PXI/cPCI instrument)
- 2 Hybrid Slots with Star and DSTAR Triggers (Can be used by any PXIe/cPCIe or PXI/cPCI Hybrid compliant instrument)

Physical Dimensions

Empty Weight:	GX7600: 16 lbs GX7610: 13 lbs
Dimensions:	8.9” Wide 7” High (4U) 18” Deep

Environmental

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

Appendix B –PXI Slots Pin Outs

This appendix describes the P1 and P2 connector pin outs for the GX7600 and GX7610 backplanes, and defines the bus signal names. Also, use Figure B-1 for orientation of connector layouts when performing slot pin outs.

- Table B-1 exhibits the bus signal names.

  To help in reviewing the tables in this section and locating the appropriate specification for signal requirements, Table B-1 lists all signals alphabetically by original specification (PXI, CompactPCI, or PCI).
- Table B-2 shows the P4 (J4) connector pin out for the System Controller slot.
- Table B-3 shows the P3 (J3) connector pin out for the System Controller slot.
- Table B-4 shows the P2 (J2) connector pin out for the System Controller slot (4-link configuration).
- Table B-5 shows the P1 (J1) connector pin out for the System Controller slot.
- Table B-6 shows the P4 (J4) connector pin out for the System Timing slot.
- Table B-7 shows the P3 (J3) connector pin out for the System Timing slot.
- Table B-8 shows the P2 (J2) connector pin out for the System Timing slot.
- Table B-9 shows the P4 (J4) connector pin out for the Hybrid slots.
- Table B-10 shows the P3 (J3) connector pin out for the Hybrid slots.
- Table B-11 shows the P1 (J1) connector pin out for the Hybrid slots.
- Table B-12 shows the P1 (J1) connector pin out for the PXI-1 Peripheral slots.
- Table B-13 shows the P2 (J2) connector pin out for the PXI-1 Peripheral slots.

Bus Signal Names

System	Signals		
PXI Express	PXIe_CLK100[p,n][3..1]	PXIe_DSTAR[C..A][p,n][0,3,5]	PXIe_SYNC_CTRL
	PXIe_SYNC100[p,n][3..1		
CompactPCI Express	SYSEN#	ALERT#	PWR_OK
	PS_ON#	LINKCAP	PWRBTN#
	ATNLED	ATNSW#	PWREN#
	MPWRGD		
PCI Express	[4..1]PE[T,R][p,n][3..0]	[4..1]RefClk[p,n]	PERST#
	SMDAT	SMBCLK	WAKE#
	PRSNT#		
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

Back Plane Connector Layouts by Slot Type

See Figure B-1 for connector layouts used for performing slot pin out examinations. This figure is simplified for clarity.

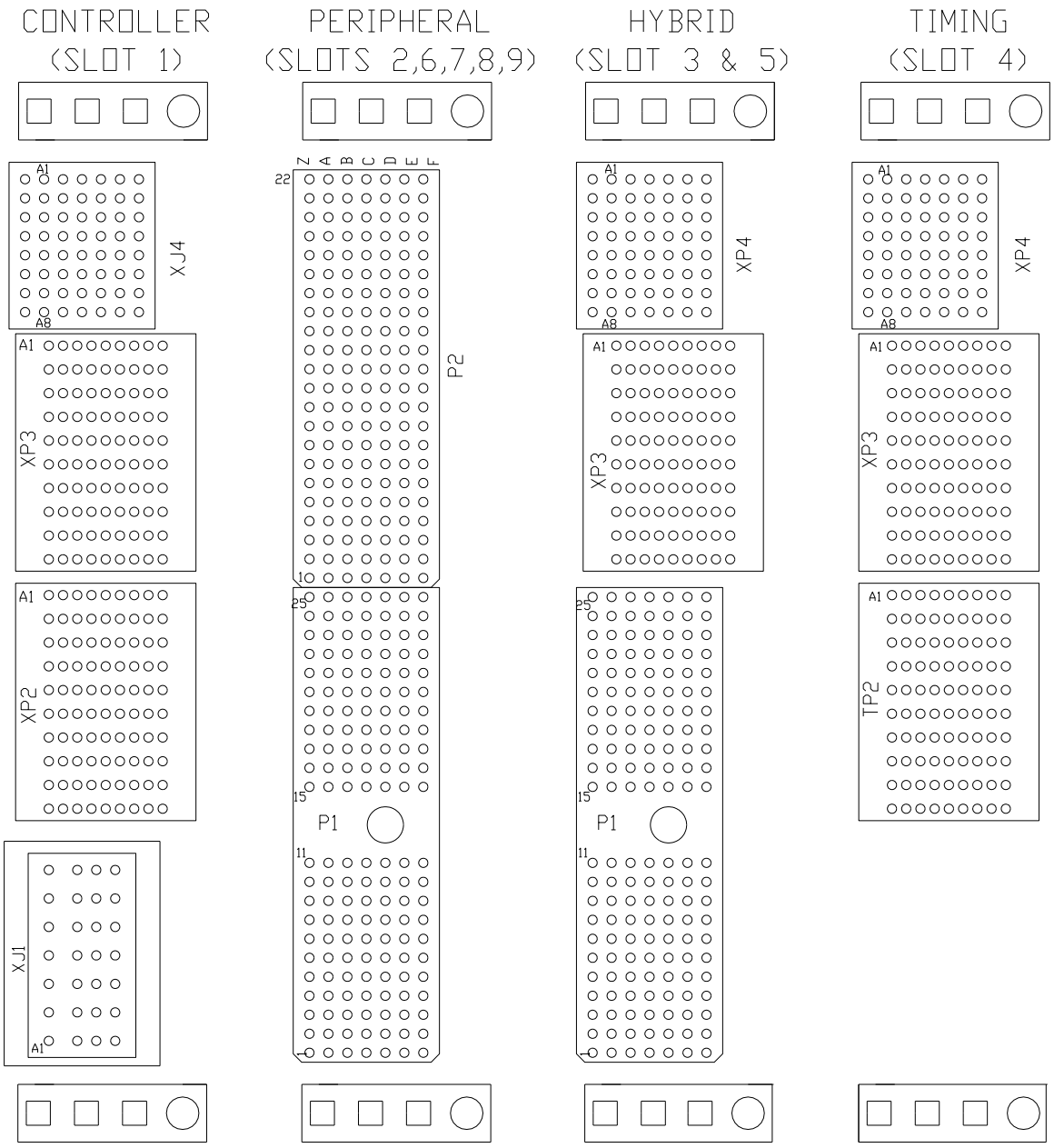

Figure B-1: Connector Layouts by Slot Type

XP4 (XJ4) Connector Pin Out for System Controller Slot

Pin	Z	A	B	C	D	E	F
1	GND	GA4	GA3	GA2	GA1	GA0	GND
2	GND	5Vaux	GND	SYSEN#	WAKE#	ALERT#	GND
3	GND	LPC_CLK_33MHz(1)	+5V(1)	SMB2_CLK/Tx(1)	SMB2_DAT/Tx(1)	SATA_TX+(1)	GND
4	GND	GND(1)	LPC_LFRAME#(1)	+3V3(1)	(BOOT)(1)	SATA_TX-(1)	GND
5	GND	LPC_LAD0(1)	LPC_RST#(1)	ETH_TX+(1)	+3V3(1)	SATA_RX+(1)	GND
6	GND	LPC_LAD1(1)	LPC_SMI#(1)	ETH_TX-(1)	GND(1)	SATA_RX-(1)	GND
7	GND	LPC_LAD2(1)	LPC_SERIRQ(1)	ETH_RX+(1)	GND	USB1+(1)	GND
8	GND	LPC_LAD3(1)	LPC_LDRQ#(1)	ETH_RX-(1)	+5V(1)	USB1-(1)	GND

1) These signals can only be used on the rear I/O interface, and are supported in conjunction with the GX7940-RIO module when used with the GX7940.

Table B-2: XP4 (XJ4) Connector Pin Out for the System Controller Slot

XP3 (XJ3) Connector Pin Out for System Controller Slot (4-Link Configuration)

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

B-3: XP3 (XJ3) Connector Pin Out for the System Controller Slot

XP2 (XJ2) Connector Pin Out for System Controller Slot (4-Link Configuration)

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

Table B-4: XP2 (XJ2) Connector Pin Out for the System Controller Slot

XP1 (XJ1) Connector Pin Out for System Controller Slot

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

Table B-5: XP1 (XJ1) Connector Pin Out for the System Controller Slot

XP4 (XJ4) Connector Pin Out for System Timing Slot

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

Table B-6: XP4 (XJ4) Connector Pin Out for the System Timing Slot

XP3 (XJ3) Connector Pin Out for System Timing Slot

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

*Pins not used in 4-Link Configuration

Table B-7: XP3 (XJ3) Connector Pin Out for the System Timing Slot

TP2 (TJ2) Connector Pin Out for System Timing Slot

Pin	A	B	AB	C	D	CD	E	F	EF
1	PXIe_DSTARC0+	PXIe_DSTARC0-	GND	PXIe_DSTARC8+*	PXIe_DSTARC8-	GND	PXIe_DSTARB8+*	PXIe_DSTARB8-*	GND
2	PXIe_DSTARA0+	PXIe_DSTARA0-	GND	PXIe_DSTARC9+*	PXIe_DSTARC9-	GND	PXIe_DSTARA8+*	PXIe_DSTARA8-*	GND
3	PXIe_DSTARB0+	PXIe_DSTARB0-	GND	PXIe_DSTARC1+*	PXIe_DSTARC1-	GND	PXIe_DSTARA9+*	PXIe_DSTARA9-*	GND
4	PXIe_DSTARB1+*	PXIe_DSTARB1-*	GND	PXI_STAR0*	PXI_STAR1*	GND	PXIe_DSTARB9+*	PXIe_DSTARB9-*	GND
5	PXIe_DSTARA1+*	PXIe_DSTARA1-*	GND	PXI_STAR2	PXI_STAR3	GND	PXIe_DSTARC10+	PXIe_DSTARC10-	GND
6	PXIe_DSTARC2+*	PXIe_DSTARC2-*	GND	PXI_STAR4*	PXI_STAR5	GND	PXIe_DSTARA10+	PXIe_DSTARA10-	GND
7	PXIe_DSTARB2+*	PXIe_DSTARB2-*	GND	PXI_STAR6	PXI_STAR7	GND	PXIe_DSTARB10+	PXIe_DSTARB10-	GND
8	PXIe_DSTARA2+*	PXIe_DSTARA2-*	GND	PXI_STAR8	PXI_STAR9	GND	PXIe_DSTARC11+	PXIe_DSTARC11-	GND
9	PXIe_DSTARC3+*	PXIe_DSTARC3-*	GND	PXI_STAR10*	PXI_STAR11*	GND	PXIe_DSTARA11+	PXIe_DSTARA11-	GND
10	PXIe_DSTARB3+*	PXIe_DSTARB3-*	GND	PXIe_DSTARC16+*	PXIe_DSTARC16-*	GND	PXIe_DSTARB11+	PXIe_DSTARB11-	GND

*Not used in the Gx7600, see the *.ini file for trigger routing

Table B-8: TP2 (TJ2) Connector Pin Out for the System Timing Slot

XP4 (XJ4) Connector Pin Out for Hybrid Slots

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

B-9: XP4 (XJ4) Connector Pin Out For The Hybrid Slots

XP3 (XJ3) Connector Pin Out for Hybrid Slots

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

*Pins not used in 4-Link Configuration

Table B-10: XP3 (XJ3) Connector Pin Out for the Hybrid Slots

P1 (J1) Connector Pin Out for Hybrid Slots

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

Table B-11: P1 (J1) Connector Pin Out for the Hybrid Slots

P1 (J1) Connector Pin Out for PXI-1 Peripheral Slots

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

Table B-12: P1 (J1) Connector Pin Out for the PXI-1 Peripheral Slots

P2 (J2) Connector Pin Out for PXI-1 Peripheral Slots

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

Table B-13: P2 (J2) Connector Pin Out for the PXI-1 Peripheral Slot

Appendix C – Rear Panel Connector Layout

This section provides information on the rear panel connectors of the GX7600 (when used in conjunction with the GX7900 embedded controller).

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	--

Appendix D – Model Numbers

Chassis and Controller Model Numbers

The following are the PXI chassis and controller model numbers:

Model #	Description
GX7600	3U, 9 Slot PXI Chassis with built-in DVD-RW drive and Hard Disk drives.
GX7610	3U, 9 Slot PXI Chassis for use with GX7990 PXI Bus Expander
GX7940-141024	CPU Plug-in cPCI controller for GX7600. 1.4 GHz/1GB RAM, Pentium M
GX7940-201024	CPU Plug-in cPCI Express controller for GX7600. 2 GHz/1 GB RAM, Pentium M
PCIE-PXIE (X4)	PCIe to PXIe Bus Expander for GX7600. Two card set with cable. X4 lane configuration.
PCIE-PXIE (X1)	PCIe to PXIe Bus Expander for GX7600. Two card set with cable. X1 lane configuration.

Chassis Accessory Model Numbers

The following are the PXI chassis accessory model numbers:

Model #	Description
GX97111	Blank Panel for GX7600, 1-slot wide
GX97112	Blank Panel for GX7600, 2-slots wide
GX97114	Blank Panel for GX7600, 4-slots wide

Index

3

3U Boards....12

A

AC Input Power....81
Air Intake Panel....9
architecture....5
ATEasy....ii, 4, 5, 34, 35, 36, 37, 38

B

Back Plane Connector Layouts By Slot Type....87
BNC In....11
BNC Out....11
Borland....ii, 34, 35, 37, 38
Borland C++ Example Files:....35
Bus Signal Names....86

C

Chassis Accessory Model Numbers....99
Chassis Air Intake Panel....11
Chassis and Controller Model Numbers....99
Chassis Handle....38
Clock....82
CompactPCI Express™....3
controlling fan speed....3
Cooling....81
Copyright....i
Corrupt files....36

D

default settings on power up....3
Disclaimer....i
Distributing The Driver....39
Driver File And Virtual Panel....34
Driver Files Description....34
Driver Version....38
DVD-RW Drive....9

E

EEPROM....3
Environmental....83
Error Handling....38
Ethernet....11
Ethernet Connector....97

F

Fan Speed Switch....11
Functions Reference....45
Functions Reference....1

G

Ground....11
GX7600 And GX7610 Chassis Installation....27
GX7600 And GX7610 PXI Instrument Installation 32
GX7600 Chassis Description – Rear View....10
GX7600 Slave Configuration....28
GX7600/GX7610 Features....4
GX7600/GX7610 System Specifications....81
GxChassis driver....1, 3, 11, 19, 34, 37, 38, 39
GxChassis Driver....19
GxChassis driver (Master)....3
GxChassis Functions....46
GxChassisGetAlarmMode....47
GxChassisGetAlarmTemperature....48
GxChassisGetBoardSummary....49
GxChassisGetDriverSummary....50
GxChassisGetErrorString....51
GxChassisGetFanSpeed....54, 72
GxChassisGetFanThresholdTemperatures....55
GxChassisGetPowerSuppliesVoltages....53
GxChassisGetPxiTriggerLine....56, 58
GxChassisGetShutdownTemperature....59
GxChassisGetSlotsTemperatures....60
GxChassisGetSlotsTemperaturesStates....61
GxChassisGetSlotsTemperaturesStatistics....62
GxChassisGetSlotTemperature....63
GxChassisGetTemperatureScale....64
GxChassisGetTemperatureThresholdMode....65
GxChassisInitialize....66
GxChassisPanel....67
GxChassisRecallSettings....68
GxChassisResetPxiTriggerLines....69
GxChassisSetAlarmMode....70
GxChassisSetAlarmTemperature....71
GxChassisSetFanThresholdTemperatures....73
GxChassisSetPxiTriggerLine....74
GxChassisSetShutdownTemperature....76
GxChassisSetSlotsTemperaturesStates....77
GxChassisSetTemperatureScale....78
GxChassisSetTemperatureThresholdMode....79
GXCNT....29
GXCNT.EXE....29

H

HW....29

I

If You Need Help....i

Input Power Receptacle 11
Installation Directories 34
Interface Files 34
Introducing PXI And PCI Express 5, 6

L

LabView 36
Line Voltage Selection And Fuse Values 27
Local Bus 15

M

Manual Organization 1
Manual Scope 1
Microsoft Visual Basic .NET Example Files: 35
Microsoft Visual Basic 6.0 Example Files: 36
Microsoft Visual C++ .NET Example Files: 35
Microsoft Visual C++ 6.0 Example Files: 35
Model Numbers 99
Mounting Information Instrument Rack System 27

O

On-line Help And Manual 35
Optional Equipment 11

P

Panel 20, 21, 22, 24, 25, 26, 36
Physical Dimensions 83
Power Distribution 18
power supply 3, 4, 17, 19, 27, 81
Power Supply 81
Power Supply Cooling Fan 11
Power Supply Monitoring 82
Power Switch 11
Program Example 35
Programming GxChassis Boards Using ATEasy 38
Programming The Chassis 37
Programming Using C/C++ Tools 37
Programming Using Pascal/Delphi 38
Programming Using Visual Basic 37
PXI 21, 31
PXI Bus Segments 12, 13, 15
PXI Express System Timing Slot 15
PXI Express™ 3
PXI Instrument Removal 33
PXI Slots Pinouts 85
PXI Systems Alliance 5
Pxi Trigger Lines 24
PXI trigger lines 3
PXI/Express chassis 1
PXI/PCI Explorer 21, 31

R

Rear Panel Connector Layout 97

S

Safety and Handling i
Sample Program Listing 40
Sample Programs 40
Setup 29, 36
Setup And Installation 27
Setup Maintenance Program 36
Slots 82
Star Trigger Lines 17
System Controller Slot 15
System Power Supply 17
System Reference Sources 17

T

Temperature Monitoring 82
Temperature Settings 22
The GxChassis Driver 37
The PXI Standard 5
Trademarks ii
Trigger Bus 16

U

Unpacking And Inspecting The Chassis 27
USB 11
USB Connector 97
Using External Instruments 33
Using The GxChassis Driver Functions 38

V

Virtual Front Panel Window 39
Virtual Instrument Software Architecture 5
Virtual Panel 20, 21, 22, 24, 25, 26, 29
  About Page 26
  Advanced page 25
  Initialize Dialog 21
Virtual Panel Description 20
VXI plug & play System Alliance 5

W

Warranty i