

# ***GX5732***

## **Static Digital I/O PXI Board GXPIO Software**

### ***User's Guide***

Last updated July 2, 2015



---

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 board, please contact Marvin Test Solutions Customer Technical Services Department via <http://www.marvintest.com/magic/>- the Marvin Test Solutions on-line support system.

## If You Need Help

---

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

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

## Disclaimer

---

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

## Copyright

---

Copyright © 2010-2015 by Marvin Test Solutions, Inc. All rights reserved. No part of this document can be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Marvin Test Solutions.

## Trademarks

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

All other trademarks are the property of their respective owners.

# Table of Contents

Safety and Handling.....	i
Warranty .....	i
If You Need Help.....	i
Disclaimer .....	i
Copyright .....	i
Trademarks .....	ii
<b>Chapter 1 - Introduction .....</b>	<b>1</b>
Manual Scope and Organization .....	1
Manual Scope.....	1
Manual Organization.....	1
Conventions Used in this Manual .....	1
<b>Chapter 2 - Overview .....</b>	<b>3</b>
Introduction.....	3
Features.....	3
Applications.....	3
Board Description .....	4
Architecture .....	5
Digital I/O .....	5
Counters .....	5
Specifications.....	7
Channel Specifications.....	7
Power Requirements .....	7
Environmental .....	7
Physical .....	7
<b>Chapter 3 - Installation and Connections .....</b>	<b>9</b>
Getting Started .....	9
Packing List .....	9
Unpacking and Inspection.....	9
System Requirements.....	9
Installation of the GXPIO Software.....	10
Setup Maintenance Program .....	10
Overview of the GXPIO Software .....	11
Installation Folders .....	11
Configuring Your PXI System using the PXI/PCI Explorer.....	12
Board Installation.....	13

Before you Begin .....	13
Electric Static Discharge (ESD) Precautions .....	13
Installing a Board .....	13
Plug & Play Driver Installation .....	15
Removing a Board .....	15
GXPIO Driver Files Description.....	16
Driver File and Virtual Panel .....	16
Interface Files.....	16
GXPIO On-line Help and Manual.....	16
ReadMe File.....	16
Example Programs .....	16
Connectors and Jumpers .....	18
JP1 – Chassis Ground Jumper.....	19
JB1-JB12 – Factory Installed Jumpers.....	19
Connections .....	20
J6 – Counters Connector.....	20
J7-J13 – Ports 0-6 Digital I/O Connectors .....	21
Connectors and Accessories .....	22
<b>Chapter 4 - Instrument Software Panel.....</b>	<b>23</b>
Virtual Panel Description.....	23
Virtual Panel Initialize Dialog .....	24
Virtual Panel Digital I/O Page .....	25
Virtual Panel Counters Page .....	26
Virtual Panel Advanced Page .....	27
Virtual Panel About Page.....	28
<b>Chapter 5 - Programming the Board .....</b>	<b>29</b>
The GXPIO Driver.....	29
Programming Using C/C++ Tools .....	29
Programming Using Visual Basic.....	29
Programming Using Pascal/Delphi.....	30
Programming GXPIO Boards Using ATEasy®.....	30
Using the GXPIO driver functions.....	30
Initialization, HW Slot Numbers and VISA Resource .....	31
Board Handle .....	32
Reset.....	32
Error Handling .....	32
Driver Version.....	32

Panel .....	32
Programming Examples .....	32
Distributing the Driver .....	32
Sample Programs .....	33
Sample Program Listing .....	34
<b>Chapter 6 - Functions Reference.....</b>	<b>41</b>
Introduction.....	41
GX5732 Functions .....	42
Gx5732GetBoardSummary.....	44
Gx5732GetCounterClock .....	45
Gx5732GetCounterEnable .....	46
Gx5732GetCounterGate .....	47
Gx5732GetCounterMode.....	48
Gx5732GetCounterPort .....	49
Gx5732GetCounterValue .....	50
Gx5732GetInternalClock .....	51
Gx5732GetPort .....	52
Gx5732GetPortBit .....	53
Gx5732GetPortByte.....	54
Gx5732GetPortByteDirection.....	55
Gx5732GetPortDirection .....	56
Gx5732GetPortWord .....	57
Gx5732GetTerminalCountPortConnection.....	58
Gx5732Initialize .....	59
Gx5732InitializeVisa .....	60
Gx5732LoadCounterCounterPort .....	61
Gx5732Panel.....	62
Gx5732Reset.....	63
Gx5732ResetCounter .....	64
Gx5732ResetPort .....	65
Gx5732SetCounterClock .....	66
Gx5732SetCounterEnable.....	67
Gx5732SetCounterGate .....	68
Gx5732SetCounterMode .....	69
Gx5732SetCounterPort .....	70
Gx5732SetCounterValue .....	71
Gx5732SetInternalClock.....	72

Gx5732SetPort.....	73
Gx5732SetPortBit.....	74
Gx5732SetPortByte .....	75
Gx5732SetPortByteDirection .....	76
Gx5732SetPortDirection.....	77
Gx5732SetPortWord.....	78
Gx5732SetTerminalCountPortConnection .....	79
GxPioGetDriverSummary.....	80
GxPioGetErrorString .....	81
Resource Errors.....	81
General Parameter Errors .....	81
Parameter Errors .....	82
<b>Index .....</b>	<b>83</b>



# Chapter 1 - Introduction

## Manual Scope and Organization

### Manual Scope

The purpose of this manual is to provide all the necessary information to install, use, and maintain the GX5732 instrument. This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and some understanding of digital I/O.





This manual also provides programming information using the GX5732 driver (referred in this manual **GXPIO**). Therefore, good understanding of programming development tools and languages may be necessary.

### Manual Organization

The GX5732 manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX5732 manual. Lists all the supported board and shows warning conventions used in the manual.
Chapter 2 - Overview	Describes the GX5732 features, board description, its architecture, specifications and the panel description and operation.
Chapter 3 - Installation and Connections	Provides instructions on how to install the GX5732 Board and its accompanying GXPIO software.
Chapter 4 - Instrument Software Panel	Provides instruction how to open and use the instrument front panel application in order to view and control the instrument settings.
Chapter 5 – Programming the Board	Provides a listing of GX5732 driver files, general purpose/generic driver functions, and programming methods. Discusses various supported operating systems and development tools.
Chapter 6 – Functions Reference	Contains a listing of the general GX5732 functions. Each function is described along with its syntax, parameters, and special programming comments. Samples are given for each function.

## Conventions Used in this Manual

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

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

# Chapter 2 - Overview

## Introduction

---

The GX5732 is a 6U PXI instrument board that provides 7 ports of 32 channels parallel input and output for a total of 224 channels. Designed for ATE, data acquisition, or process control systems where a large number of discrete I/O channels are required, the GX5732 offers the highest density in the industry for a single PXI plug-in board. The 224 channels have TTL levels, and the direction of each group of eight channels can be programmed as Input or Output.

The board also contains four 8 bit counters. The counters can be parallel loaded by using the provided software driver or externally from the SCSI connector. The four 8-bit, 50-MHz counters can provide counter and timing functions.

## Features

---

The GX5732 is a Static Digital Input or Output card with:

- Seven 32-bit ports for a total of 224 Input or Output channels
- TTL Levels
- Programmable direction (input or output) in groups of 8 pins
- Four 8-bit, 50-MHZ counters.

## Applications

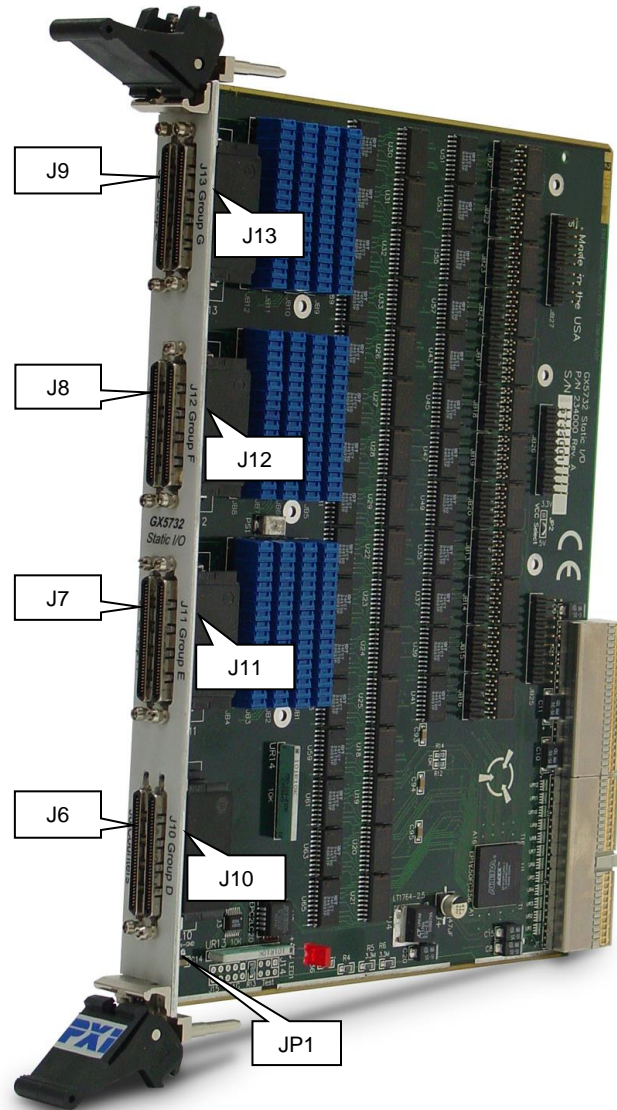
---

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

## Board Description

The GX5732 is a static digital input or output card with 224 I/O channels (pins) made out of 7 ports of 32 channels each. Each port is divided to 4 groups of 8 channels where each group can be programmed to output or input at a time.

In addition, the GX5732 has four 8-bit (1 byte), bi-directional, 50-MHz counters. An eight bit counter port can be used as input or output to any of the counters. An additional 8 outputs can be used as output of any of the counters, or as a Terminal Count (TC) for all counters. Six programmable inputs are provided for counter control: clock, gate, or load. The counters may be daisy chained to create two 16-bit counters, or one 32-bit counter.



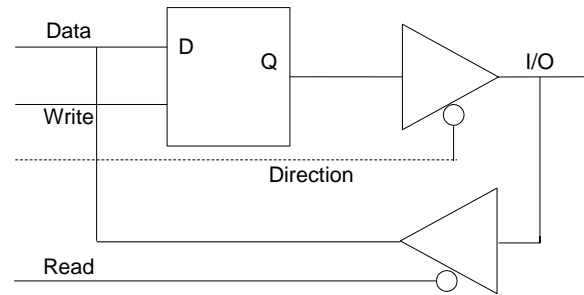
**Figure 2-1: GX5732 Board Side**

The board has eight 68 pins SCSI connectors. J6 is used for the counters' signals. J7 to J13 are used for digital I/O signals, JP2 factory installed and a jumper JP1, that is used for chassis ground. Additional jumper blocks not shown here are located on the board and are used for future expansion, the jumper are factory installed. Additional connectors, J1 and J2 in the back of the board are used to pass the PCI/PXI signals.

## Architecture

### Digital I/O

The GX5732 provides 224 digital Inputs or Outputs and 4 counters. The 224 TTL channels' direction is programmable and can be set for Input or Output in groups of eight. The GX5732 has no on-board memory and it uses software driver to set or get the channels level and direction. Figure 2-2 shows a typical I/O channel block diagram:



**Figure 2-1: Typical I/O Channel**

The GX5732 consist of seven 32-bit parallel input/output ports. Each port interfaces through a 68 pins SCSI connector (J7 to J13).

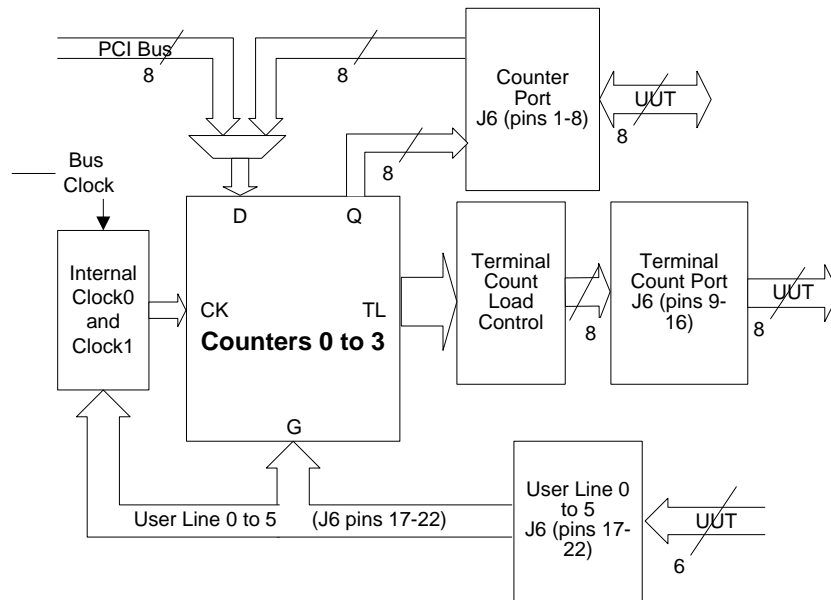
### Counters

The GX5732 contains four 8-bit counters. The counters can be loaded from the PXI bus or from the 8-bit Counter Port accessible from J6 (CP0-7 pins in J6 connector). Each counter has a Terminal Count signal that changes when the counter overflows. The signal can be routed to the TCP0-3 pins in JP6 connector or to the Carryout signal that goes to the next counter to form a 16, 24 or 32 bit counter using sibling counter(s).

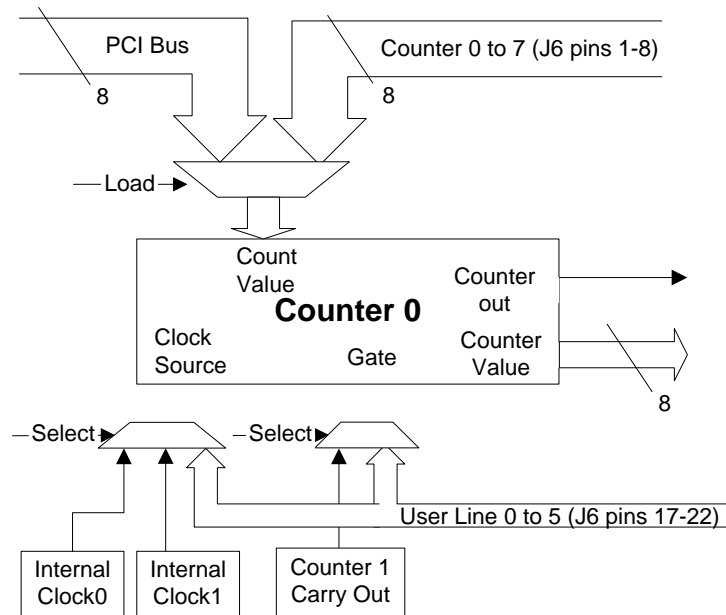
Two internal clock sources can be used to drive the counter. Each of the clocks can be configured by the 10MHz PXI clock, or the 33 MHz PCI Bus clock or any one of the 6 User Lines pins on the J6 connector (UL0-5).

The 6 User Lines input from J6 connector can be configured to drive the counter gate or clock.

The following figures: **Figure 2-2**, **Figure 2-3** and **Figure 2-4** presents block diagram for the GX5732 counters.



**Figure 2-2: GX5732 Counters**



**Figure 2-3: GX5732 Counter 0 (Typical)**

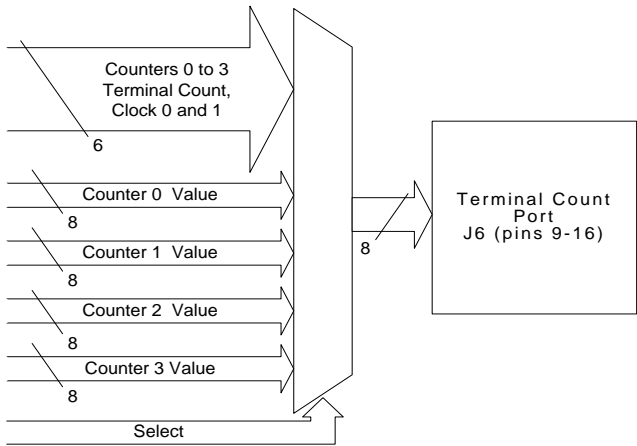


Figure 2-4: GX5732 Terminal Count Port

Specifications

The following table outlines the specifications of the GX5732.

Channel Specifications

TTL I/O Levels		
Low	0 V Min.	0.8V Max.
High	2.0V Min.	5.0V Max.
Number of channels	224	
Number of Counters	Four 8-bit	
Max. Counter Rate	50 MHz	

Power Requirements

3.3 VDC Current	0.5 A
5 VDC Current	1.3 A

Environmental

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

Physical

Size	6U PXI
Weight	18 oz.





## Chapter 3 - Installation and Connections

### Getting Started

---

This section includes general hardware installation procedures for the GX5732 board and installation instructions for the GX5732 (GXPIO) software. Before proceeding, please refer to the appropriate chapter to become familiar with the board being installed.

To Find Information on..	Refer to..
Hardware Installation	This Chapter
GX5732 Driver Installation	This Chapter
Programming the boards	Chapter 5
GX5732 Reference Functions	Chapter 6

### Packing List

All GX5732 boards have the same basic packing list, which includes:

1. GX5732 Board
2. GX5732 Driver Disk

### Unpacking and Inspection

After removing the board from the shipping carton:



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

---

1. Remove the board from the static bag by handling only the metal portions.
2. Be sure to check the contents of the shipping carton to verify that all of the items found in it match the packing list.
3. Inspect the board for possible damage. If there is any sign of damage, return the board immediately. Please refer to the warranty information at the beginning of the manual.

### System Requirements

The GX5732 Instrument board is designed to run on PXI compatible computer running Windows 2000, XP and above. In addition, Microsoft Windows Explorer version 4.0 or above is required to view the online help.

The board requires one unoccupied 6U PXI bus slot.

## Installation of the GXPIO Software

---

Before installing the board it is recommended that you install the GXPIO software as described in this section. To install the GXPIO software, follow the instruction described below:

1. Insert the Marvin Test Solutions CD-ROM and locate the **GXPIO.EXE** setup program. If your computer's Auto Run is configured, when inserting the CD a browser will show several options. Select the Marvin Test Solutions Files option and then locate the setup file. If Auto Run is not configured you can open the Windows explorer and locate the setup files (usually located under \Files\Setup folder). You can also download the file from Marvin Test Solutions' web site ([www.marvintest.com](http://www.marvintest.com)), downloading the setup from the website is preferred since it ensures that the latest software version is installed.
2. Run the GXPIO setup and follow the instruction on the Setup screen to install the GXPIO driver.

---

**Note:** You may be required to restart the setup after logging-in as a user with Administrator privileges. This is required in-order to upgrade your system with newer Windows components and to install the HW kernel-mode device drivers that are required by the GXPIO driver to access resources on your board.

---

3. The first setup screen to appear is the Welcome screen. Click **Next** to continue.
4. Enter the folder where GXPIO 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\GXPIO under 32-bit Windows or C:\Program Files (X86)\Marvin Test Solutions\GXPIO under 64-bit Windows.
5. Select the type of Setup you wish and click **Next**. You can choose between **Typical**, **Run-Time** and **Custom** setups types. The **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. The **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 completion if some of the components it replaced were used by another application during the installation – do so before attempting to use the software.

You can now continue with the installation to install the board. After the board installation is complete you can test your installation by starting a panel program that lets you control the board interactively. The panel program can be started by selecting it from the Start, Programs, GXPIO menu located in the Windows Taskbar.

## Setup Maintenance Program

---

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

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

Select one of the options and click **Next** and follow the instruction on the screen until Setup is complete.

## Overview of the GXPIO Software

---

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

- **GXPIO Panel** – Configures and controls the GXPIO board various features via an interactive user interface.
- **GXPIO driver** – A DLL based function library (GXPIO.DLL for 32-bit applications or GXPIO64.DLL for 64-bit applications, located in the Windows System folder) used to program and control the board. The driver uses Marvin Test Solutions' HW driver or VISA supplied by third party vendor to access and control the GXPIO boards.
- **Programming files and examples** – Interface files and libraries for support of various programming tools. A complete list of files and development tools supported by the driver is included in subsequent sections of this manual.
- **Documentation** – On-Line help and User's Guide for the board, GXPIO driver and panel.
- **HW driver and PXI/PCI Explorer applet** – HW driver allows the GXPIO driver to access and program the supported boards. The explorer applet configures the PXI chassis, controllers and devices. This is required for accurate identification of your PXI instruments later on when installed in your system. The applet configuration is saved to PXISYS.ini and PXIE SYS.ini and is used by Marvin Test Solutions instruments HW driver and VISA. The applet can be used to assign chassis numbers, Legacy Slot numbers and instrument alias names. The HW driver is installed and shared with all Marvin Test Solutions products to support accessing the PC resources. Similar to HW driver, VISA provides a standard way for instrument manufacturers and users to write and use instruments drivers. VISA is a standard maintained by the VXI Plug & Play System Alliance and the PXI Systems Alliance organizations (<http://www.vxipnp.org/>, <http://www.pxisa.org/>). The VISA resource manager such as National Instruments **Measurement & Automation** (NI-MAX) displays and configures instruments and their address (similar to Marvin Test Solutions' PXI/PCI Explorer). The GXPIO driver can work with either HW or VISA to control an access the supported boards.

## Installation Folders

---

The GX5732 driver files are installed in the default folder C:\Program Files\Marvin Test Solutions\GXPIO under 32-bit Windows or C:\Program Files (X86)\Marvin Test Solutions\GXPIO under 64-bit Windows.

You can change the default GXPIO folder to one of your choosing at the time of installation.

During the installation, GXPIO Setup creates and copies files to the following folders:

Name	Purpose / Contents
...\Marvin Test Solutions\GXPIO	The GXPIO folder. Contains panel programs, programming libraries, interface files and examples, on-line help files and other documentation.
...\Marvin Test Solutions\HW	HW device driver. Provide access to your board hardware resources such as memory, IO ports and PCI board configuration. See the README.TXT located in this directory for more information.
...\ATEasy\Drivers	ATEasy drivers folder. GXPIO Driver and example are copied to this directory only if ATEasy is installed to your machine.
Windows System Folders	Windows System directory. Contains the GXPIO.DLL or GXPIO64.DLL driver, HW driver shared files and some upgraded system components, such as the HTML help viewer, etc.

## 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 is 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 which are used by these commands.
3. **Change chassis numbers, PXI devices Legacy Slot numbering and PXI devices Alias names.** These are optional steps and can 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 a logical name (e.g. "PIO1"). For more information regarding slot numbers and alias names, see the **Gx5732Initialize** and **Gx5732InitializeVisa** functions.
4. **Save your 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 your changes. The PXI/Explorer will prompt you to save the changes if changes were made or detected (an asterisk sign '\*' in the caption indicated changes).

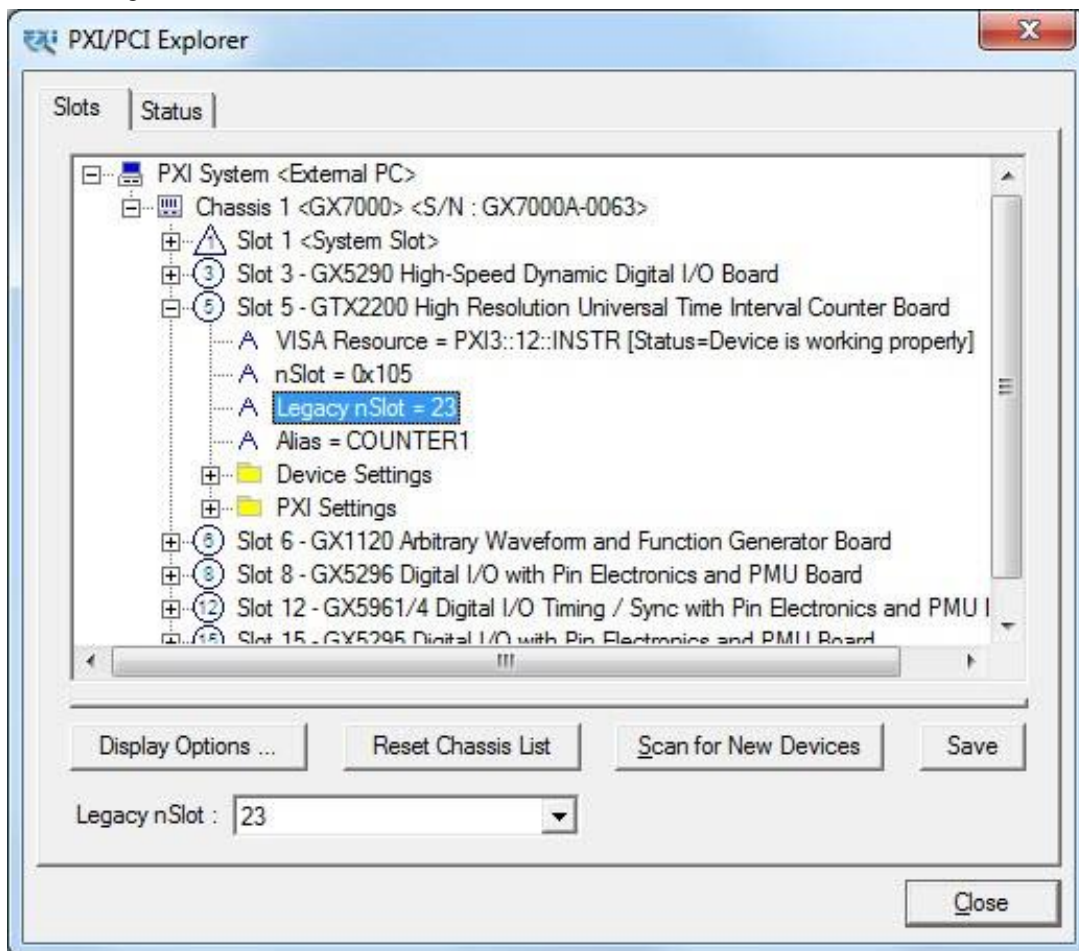


Figure 3-1: PXI/PCI Explorer

## Board Installation

---

### Before you Begin

- Install the GXPIO driver as described in the prior section.
- Configure your PXI/PC system using **PXI/PCI Explorer** as described in the prior section.
- Verify that all the components listed in the packing list (see previous section in this chapter) are present.

### Electric Static Discharge (ESD) Precautions

To reduce the risk of damage to the GX5732 board, the following precautions should be observed:

- Leave the board in the anti-static bags until installation requires removal. The anti-static bag protects the board from harmful static electricity.
- Save the anti-static bag in case the board is removed from the computer in the future.
- Carefully unpack and install the board. Do not drop or handle the board roughly.
- Handle the board by the edges. Avoid contact with any components on the circuit board.



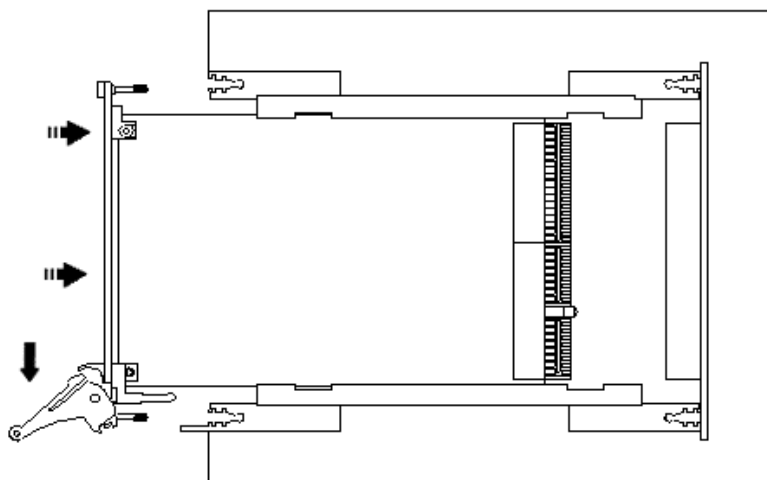
**Caution** – Do not insert or remove any board while the computer is on. Turn off the power from the PXI chassis before installation.

---

### Installing a Board

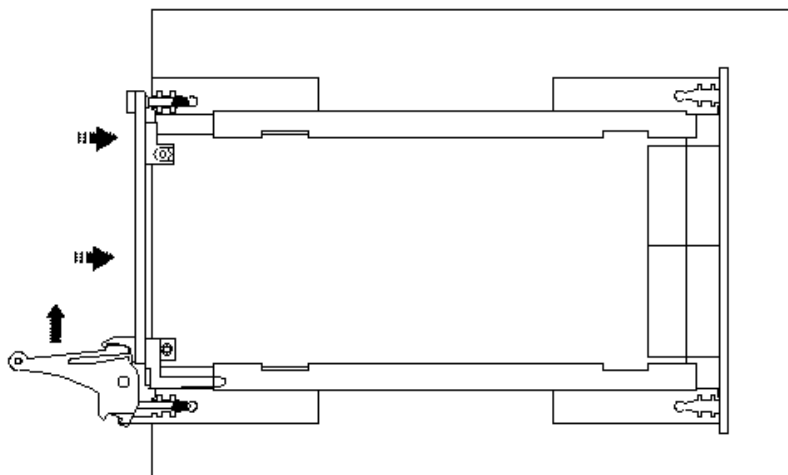
Install the board as follows:

1. Install first the GXPIO Driver as described in the next section.
2. Turn off the PXI chassis and unplug the power cord.
3. Locate a PXI empty slot on the PXI chassis.
4. Place the module edges into the PXI chassis rails (top and bottom).
5. Carefully slide the PXI board to the rear of the chassis, make sure that the ejector handles are pushed **out** (as shown in Figure 3-2).



**Figure 3-2: Ejector handles position during module insertion**

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



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

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

## Plug & Play Driver Installation

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

If another Marvin Test Solutions board software package was already installed, Windows will suggest using the driver information file: HW.INF. The file is located in your computer Program Files under \Marvin Test Solutions\HW folder. Click **Next** to confirm and follow the instructions on the screen to complete the driver installation.

If the operating system was unable to find the driver (since the GXPIO driver was not installed prior to the board installation), you may install the GXPIO driver as described in the prior section, then click on the **Have Disk** button and browse to select the HW.INF file located in your computer Program File under \Marvin Test Solutions\HW.

If you are unable to locate the driver click **Cancel** to the found New Hardware wizard and exit the New Hardware Found Wizard, install the GXPIO driver, reboot your computer and repeat this procedure.

The Windows Device Manager (open from the System applet from the Windows Control Panel) must display the proper board name before continuing to use the board software (no Yellow warning icon shown next to device). If the device is displayed with an error you can select it and press delete and then press F5 to rescan the system again and to start the New Hardware Found wizard.

## Removing a Board

Remove the board as follows:

1. Turn off the PXI chassis and unplug the power cord.
2. Locate a PXI slot on the PXI chassis.
3. Disconnect and remove any cables/connectors connected to the board.
4. Un-tighten the module's front panel screws to the chassis.
5. Push out the ejector handles and slide the PXI board away from the chassis.
6. Optionally – uninstall the GXPIO driver.

## GXPIO Driver Files Description

---

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

### Driver File and Virtual Panel

- GXPIO.DLL and GXPIO64.DLL – 32/64 bit Windows DLLs for 32/64 bit applications running under Windows.
- GXPIOPANEL.EXE and GXPIOPANEL64.EXE– 32/64 bit Windows executable provides an instrument front panel program for all GXPIO supported boards.

### Interface Files

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

- GXPIO.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++
- GXPIO.LIB and GXPIO64.LIB - Import library for GXPIO.DLL and GXPIO64.DLL (used when linking C/C++ 32/64 bit application).
- GXPIOBC.LIB - Import library for GXPIO.DLL (used when linking Borland C/C++ application that uses GXPIO.DLL).
- GXPIO.PAS - interface file to support Borland Pascal Borland Delphi.
- GXPIO.BAS - Supports Microsoft Visual Basic 4.0, 5.0 and 6.0.
- GXPIO.VB - Supports Microsoft Visual Basic .NET.
- GX5732.DRV - ATEasy driver File for GX5732.GXPIO Virtual Panel Program

### GXPIO On-line Help and Manual

GXPIO.CHM – On-line version of the GX5732 User's Guide. The help file is provided in a Windows Compiled HTML help file (.CHM). The file contains information about the GX5732 board, programming reference and panel operation.

GX5732UG.PDF – On line, printable version of the GX5732 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 GXPIO manuals. You can view and/or print this file using the Windows NOTEPAD.EXE or other text file editors.

### Example Programs

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

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

- GXPIOExampleC.cpp - Source file
- GXPIOExampleC.ico - Icon file
- GXPIOExampleC.rc - Resource file



- GXPIOExampleC.vcproj - VC++ .NET project file
- GXPIOExampleC.exe - 32 bit example executable
- GXPIOExampleC64.exe - 64 bit example executable

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

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

**Borland C++ example files:**

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

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

- GxPioExampleVB.vb - Example form.
- GxPioExampleVB.resx - Example form resource.
- GxPioExampleVBApp.config - Example application configuration file.
- GxPioExampleVBAssemblyInfo.vb - Example application assembly file
- GXPIOExampleVB.vbproj - Project file
- GXPIOExampleVB.exe - Example executable

**Microsoft Visual Basic 6.0 example files:**

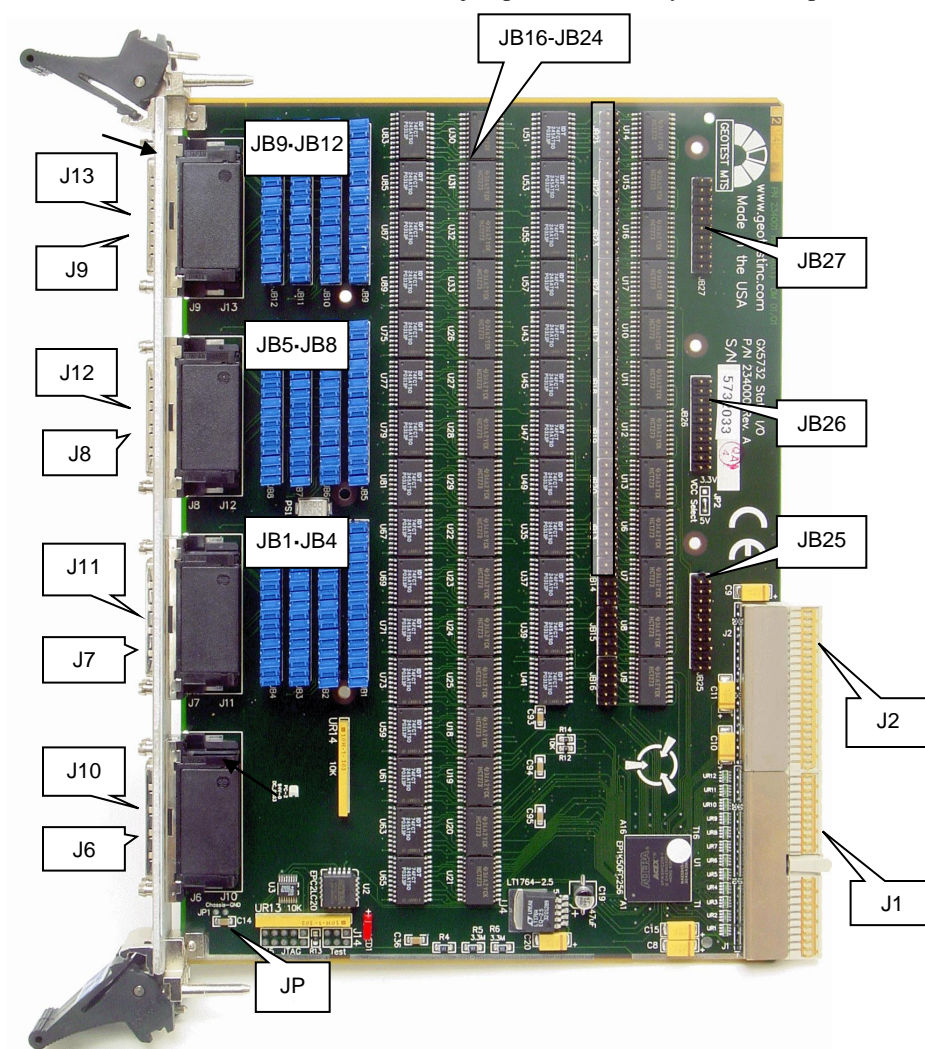
- GxPioExampleVB6.frm - Example form
- GxPioExampleVB6.frx - Example form binary file
- GXPIOExampleVB6.vbp - Project file
- GXPIOExampleVB6.exe - Example executable.

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

- Gx5732.prj - example project
- GX5732.sys - example system
- GX5732.prg - example program

## Connectors and Jumpers

Figure 3-4 shows the available GX5732 board connectors and jumpers followed by their description:



**Figure 3-4: GX5732 Connectors and Jumpers**

Connector/Jumpers	Description
J1	PCI/PXI Connector
J2	PCI/PXI Connector
J6	Counter connector
J7	Port 0 digital I/O Connector
J8	Port 1 digital I/O Connector
J9	Port 2 digital I/O Connector

Connector/Jumpers	Description
J10	Port 3 digital I/O Connector
J11	Port 4 digital I/O Connector
J12	Port 5 digital I/O Connector
J13	Port 6 digital I/O Connector
JP1	Chassis ground jumper
JP2	VCC Select. Factory installed, Do Not Remove
JB1-JB12	Factory Installed Jumpers do not Remove
JB16-JB24	Unpopulated Jumpers for future use

### **JP1 – Chassis Ground Jumper**

---

JP1 jumper connects the chassis ground to the logic ground. When jumper is not installed (default), chassis and logic ground are separate. This does not preclude them being connected elsewhere, such as in the power supplies. When the jumper is installed, chassis and logic ground are connected on the board.

### **JB1-JB12 – Factory Installed Jumpers**

---

The Jumper Blocks of JP1 through JB12 (groups of JP1-JP4, JP5-JP8, JP9-JP12) are factory installed jumpers and should not be removed. All jumpers should be installed in order for the board to work properly.

## Connections

Connections to the GX5732 may be made with eight 68 pins SCSI male plug connector. Shielded cables with matching connectors are available from Marvin Test Solutions. Table 3-1 and Table 3-2 list the connectors' pins.

### J6 – Counters Connector

The following are the connector signal pin assignments for GX5732 module:

#	Signal	#	Signal	#	Signal	#	Signal
1	CP0	18	UL1	35	GND	52	GND
2	CP1	19	UL2	36	GND	53	GND
3	CP2	20	UL3	37	GND	54	GND
4	CP3	21	UL4	38	GND	55	GND
5	CP4	22	UL5	39	GND	56	GND
6	CP5	23	NC	40	GND	57	GND
7	CP6	24	NC	41	GND	58	GND
8	CP7	25	NC	42	GND	59	GND
9	TCP0	26	NC	43	GND	60	GND
10	TCP1	27	NC	44	GND	61	GND
11	TCP2	28	NC	45	GND	62	GND
12	TCP3	29	NC	46	GND	63	GND
13	TCP4	30	NC	47	GND	64	GND
14	TCP5	31	NC	48	GND	65	GND
15	TCP6	32	NC	49	GND	66	GND
16	TCP7	33	VCC	50	GND	67	VCC
17	UL0	34	GND	51	GND	68	GND

**Table 3-1: J6 – Counters Connector**

CPx	Counter Port, Input or Output (8 pins)
TCPx	Terminal Count Port, Output (8 pins)
ULx	User Lines 0-5, (6 pins)
GND	Ground
NC	Not connected or used
VCC	+5V

## J7-J13 – Ports 0-6 Digital I/O Connectors

The following are the connector signal pin assignments for GX5732 module:

#	Signal	#	Signal	#	Signal	#	Signal
1	I/On0	18	I/On17	35	GND	52	GND
2	I/On1	19	I/On18	36	GND	53	GND
3	I/On2	20	I/On19	37	GND	54	GND
4	I/On3	21	I/On20	38	GND	55	GND
5	I/On4	22	I/On21	39	GND	56	GND
6	I/On5	23	I/On22	40	GND	57	GND
7	I/On6	24	I/On23	41	GND	58	GND
8	I/On7	25	I/On24	42	GND	59	GND
9	I/On8	26	I/On25	43	GND	60	GND
10	I/On9	27	I/On26	44	GND	61	GND
11	I/On10	28	I/On27	45	GND	62	GND
12	I/On11	29	I/On28	46	GND	63	GND
13	I/On12	30	I/On29	47	GND	64	GND
14	I/On13	31	I/On30	48	GND	65	GND
15	I/On14	32	I/On31	49	GND	66	GND
16	I/On15	33	VCC	50	GND	67	VCC
17	I/On16	34	GND	51	GND	68	GND

**Table 3-2: J7 – J13: GX5732 Port 0-6 Digital I/O Connectors**

I/On0 to I/On31	I/O Data Lines, Port number ( <i>n</i> ) can be 0 to 2
VCC	+5V
GND	Ground

## Connectors and Accessories

---

The following accessories are available from Marvin Test Solutions for GX5732 switching board.

Part / Model Number	Description
GX95401	Extra GX5732 User Manual
GT95014	Connector Interface for GX5xxx/GX5732, SCSI to 100 Mil Grid, Single Ended
GT95021	2' shielded cable for GX5xxx/GX5732 (68 Pin)
GT95022	3' shielded cable for GX5xxx/GX5732 (68 Pin)
GT95028	10' shielded cable for GX5xxx/GX5732 (68 Pin)
GT95031	6' shielded cable for GX5xxx/GX5732 (68 Pin)

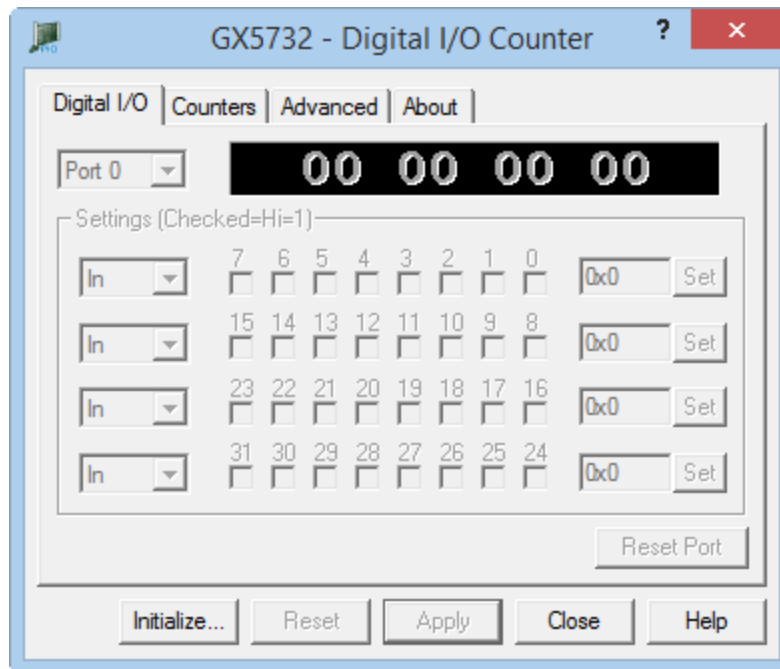
## Chapter 4 - Instrument Software Panel

Calling the **GX5732Panel** function will display the instrument front panel in a window. The panel can be used to initialize 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 **Gx5732Panel** function is also used by the GXPIOPANEL.EXE or GXPIOPANEL64.EXE (with the **/5732** command line argument) panel programs that is supplied with this package and provides a standalone Windows application that displays the instrument panel.

### Virtual Panel Description

The **GX5732** includes a virtual panel program, which enables full utilization of the various configurations and controlling modes. To fully understand the front panel operation, it is best to become familiar with the functionality of the board.

To open the virtual panel application, select **GX5732 Panel** from the **Marvin Test Solutions, GXPIO** menu under the **Start** menu. The GX5732 virtual panel opens as shown here:



**Figure 4-1: GX5732 Virtual Panel**

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

**Reset** – Resets the PXI board settings to their default state and clears the reading.

**Apply** – Applies changed settings to the board.

**Close** – Closes the panel. Closing the panel **does not affect** the board settings.

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

## Virtual Panel Initialize Dialog

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

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

1. **Use Marvin Test Solutions' HW** – This is the device driver installed by the setup program and is the default driver. When selected, the **Slot Number** list displays the available **GX5732** boards installed in the system and their slots. The chassis, slots, devices and their resources are also displayed by the HW resource manager, **PXI/PCI Explorer** applet that can be opened from the Windows Control Panel. The **PXI/PCI Explorer** can be used to configure the system chassis, controllers, slots and devices. The configuration is saved to PXISYS.INI and PXIeSYS.INI located in the Windows folder. These configuration files are also used by VISA. The following figure shows the slot number 0x105 (chassis 1 Slot 5). This is the slot number argument (*nSlot*) passed by the panel when calling the driver **Gx5732Initialize** function which is used to initialize the driver for the specified board.

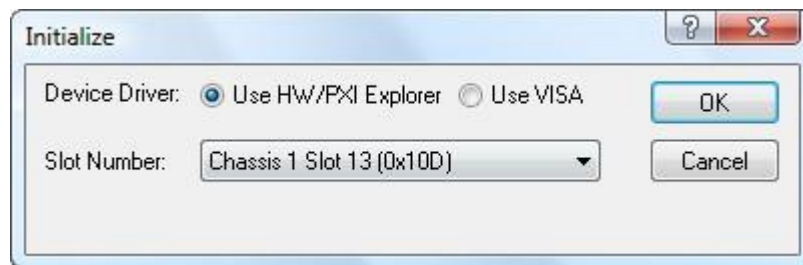


Figure 4-2: Initialize Dialog Box using Marvin Test Solutions' HW driver

2. **Use VISA** – This is a third party device driver usually provided by National Instrument (NI-VISA). When selected, the **Resource** list displays the available boards installed in the system and their VISA resource address. The chassis, slots, devices and their resources are also displayed by the VISA resource manager, **Measurement & Automation** (NI-MAX) and by Marvin Test Solutions **PXI/PCI Explorer**. The following figure shows PXI9::13::INSTR as the VISA resource (PCI bus 9 and Device 13). This is a VISA resource string argument (*szVisaResource*) which is passed by the panel when calling the driver **Gx5732InitializeVisa** function which initializes the driver for the specified board.



Figure 4-3: Initialize Dialog Box using VISA resources



## Virtual Panel Digital I/O Page

After the board is initialized the panel is enabled and will display the current setting of the board. The following picture shows the **Digital I/O page** settings:

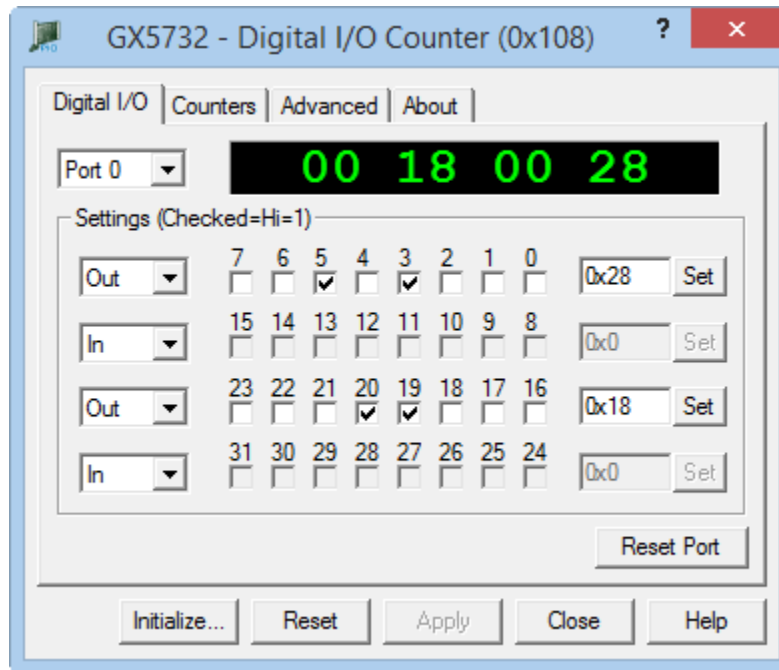


Figure 4-4: GX5732 Virtual Panel – Digital I/O page

The following controls are shown in the Digital I/O page:

**Port drop-down box (Ports 0 – 6):** Selects a port to display its current settings in this page. All other controls in this page show the selected port settings.

**Port Value display area:** Displays the current selected port value. The value is displayed in hexadecimal. The right two digits correspond to byte #0 and the left two digits to byte #3.

**Reset Port button:** Resets the current selected port. Clicking on the button will set all the port groups to input mode.

**Port Direction drop-down boxes (Byte0-3):** Displays the current port's direction each check box represents 8 channels: Byte0 for channels 0-7, Byte1 for 8-15, Byte2 16-23 and Byte3 for 24-31.

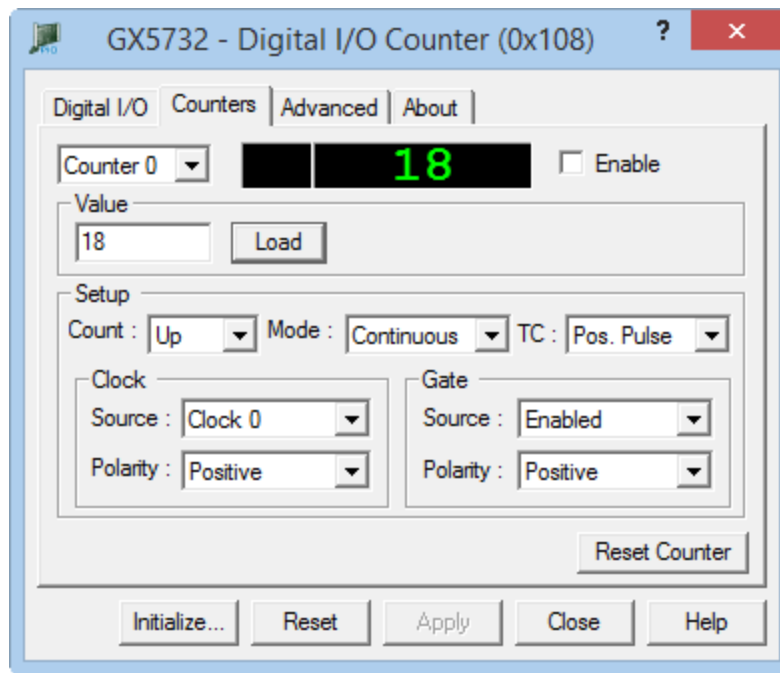
**Channel Value check boxes (0-31):** Displays the current port channels state. If the channel box is checked the channel is in Hi state. The check boxes display the current state and are used to change the channel state when in output mode. The checkbox are in read-only (disabled) when the channel in input.

**Byte Value edit box (Byte0-3):** Displays the ports byte in hexadecimal. When the byte is in the output direction, you may type a value directly in hexadecimal (prefix with 0x) or decimal and press the Set button to set the value to the output channels. These controls are disabled when the byte in input direction.

**Set button:** Sets the value in the adjacent edit box to the output byte channels. This control is enabled only when the byte is in output direction.

## Virtual Panel Counters Page

Clicking on the **Counters** tab will show the **Counters** page as shown in **Error! Reference source not found.**:



**Figure 4-5: GX5732 Virtual Panel – Counters Page**

The following controls are shown in the Counters page:

**Counter drop-down box (0-3):** Selects a counter to display its current settings in this page. All other controls in this page show the selected counter settings.

**Counter Value display area:** Displays the counter value decimal value (0-255) and the counter terminal state. If the counter terminal state is overflow the **Ovr** is shown.

**Enable check box:** Enables (checked) or disable the counter operation.

**Value edit box:** The value to load to the counter. Set the value in decimal or hexadecimal (prefix with 0x) and press the load button.

**Load button:** Loads a value to the counter.

**Mode drop-down box:** Displays the counter modes: Continuous or One-Shot. In continuous mode when the counter overflows and reaches 0 or 255 (depends if it counts down or up) the counter continues to counts starting with 255 or 0 accordingly.

**Counts drop-down box:** Displays weather the counter counts up incrementing its value by 1or down (decrementing by 1).

**TC drop-down box:** Displays the action performed by the counter when the counter got to overflow state. Possible values are: Positive pulse, negative Pulse, Low to High or High to low signals.

**Clock Source drop-down box:** Specifies the counter clock polarity used to increment or decrement its value. Selections can be: Internal clock 0 or 1 or User line 0 to 5.

**Clock Polarity drop-down box:** Specifies the counter clock source used to increment or decrement its value. Selections can be: Positive or Negative. When Positive the counter counts when the clock changes from low to high. When negative the counter counts when the clock changes from high to low.

**Gate Source drop-down box:** Specifies the counter gate source mode. The gate is used to control when the counter is enable. Possible values are: Enabled, Carryout and user line 0 to 5. When enabled the counter always enabled. When Carryout is selected, the counter is enabled when the carry signal from another counter changes to high.

**Gate Polarity drop-down box:** Specifies the counter gate source mode. The gate is used to control when the counter is enable. Possible values are: Positive or Negative. When Positive the counter counts when the gate changes from low to high. When negative the counter counts when the gate changes from high to low.

**Reset Counter button:** Resets the current counter. This will set the Value source to software

## Virtual Panel Advanced Page

Clicking on the **Advanced** tab will show the **Advanced page** as shown in **Error! Reference source not found.:**

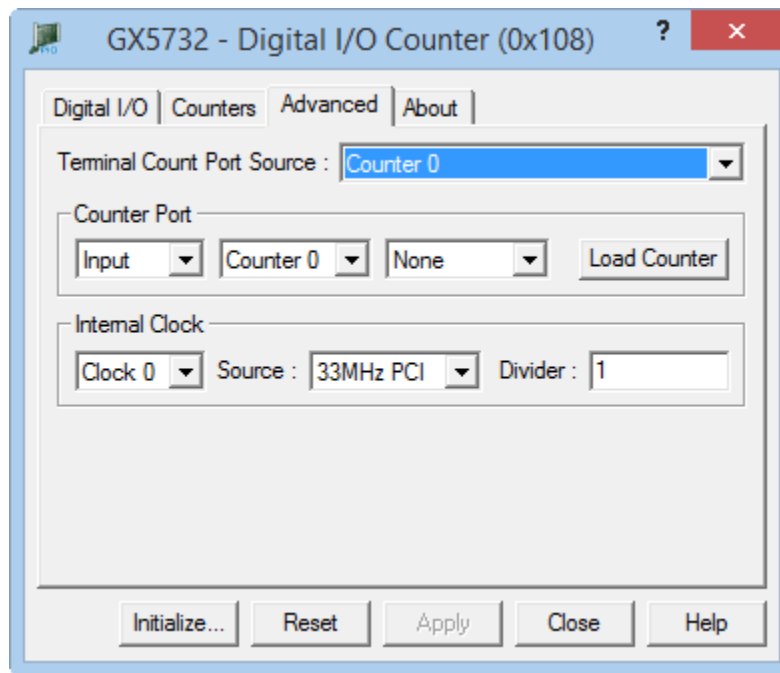


Figure 4-6: GX5732 Virtual Panel – Advanced Page

The following controls are shown in the Advanced page:

**Terminal Count Port Source drop-down box:**

**Counter Port Direction drop-down box:** The Counter Port direction. Select Input or Output direction.

**Counter Port Counter drop-down box:** The counter connected to the Counter Port. Select None or Counter 0 to 3.

**Counter Port Load Control Source drop-down box:** Displays the counter Load Control Source. Possible selections are: None or User Line 0 to 5. When a User Line is selected the Counter port will load when the line signal changes.

**Load Counter button:** Loads the counter with value stored in the Counter Port. This control is only visible when the Counter port is in Input mode.

**Internal Clock drop down box:** Selects the internal clock that its settings are displayed in the Source and Divider controls.

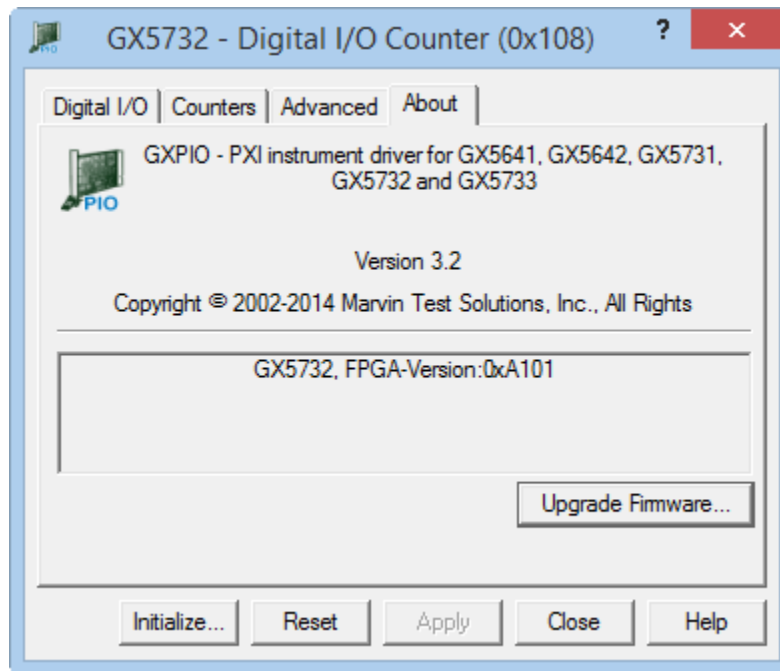
**Internal Clock Source drop down box:** The source that is used as an input to the selected internal clock. This can be the 2 on board clocks 10 and 33MHz or User Lines 0 to 5.

**Divider edit box:** The clock divider. Type a number between 1 to 0xFFFFFFFF. This number will be used to divide the Clock the input from the clock source.

## Virtual Panel About Page

---

Clicking on the **About** tab will show the **About** page as shown in **Error! Reference source not found.**:



**Figure 4-7: GX5732 Virtual Panel – About Page**

The About page displays version and copyright of the GXPIO driver.

## Chapter 5 - Programming the Board

This chapter contains information about how to program the board using the GXPIO driver. The GXPIO driver contains functions to initialize, reset, and control the board. 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 GXPIO driver supports many development tools. Using these tools with the driver is described in this chapter. In addition, the GXPIO 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 GXPIO driver types.

### The GXPIO Driver

---

The GXPIO driver is provided with a 32-bit Windows DLL file GXPIO.DLL and a 64-bit GXPIO64.DLL. The 32-bit DLL is used with 32-bit or a 64-bit applications running under Windows. The DLLs use the provided HW device driver to access the board resources. The HW device driver is installed by the setup program and is shared by other Marvin Test Solutions products (ATEasy, GTDIO, etc). The DLLs can also use VISA (provided by a third party) to access the board hardware instead of the provided HW driver.

The DLLs 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 GX5732 driver with C/C++ development tools:

- Include the GXPIO.H header file in the C/C++ source file that uses the GX5732 function. This header file is used for all driver types. The file contains function prototypes and constant declarations to be used by the compiler for the application.
- Add the required .LIB file to the projects. This can be import library GXPIO.LIB for Microsoft Visual C++ for 32-bit applications, GXPIO64.LIB for 64 bit applications and GXPIOBC.LIB for Borland C++. Windows based applications that explicitly load the DLL by calling the Windows **LoadLibrary** API should not include the .LIB file in the project.
- Add code to call the GX5732 as required by the application.
- Build the project.
- Run, test, and debug the application.

### Programming Using Visual Basic

---

To use the driver with Visual Basic 4.0, 5.0 or 6.0 (for 32-bit applications), the user must include the GXPIO.BAS to the project. For Visual Basic .NET use the GXPIO.VB.

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

## Programming Using Pascal/Delphi

---

To use the driver with Borland Pascal or Delphi, the user must include the GXPIO.PAS to the project. The GXPIO.PAS file contains a **unit** with function prototypes for the DLL functions. Include the GX5732 unit in the **uses** statement before making calls to the GX5732 functions.

## Programming GXPIO Boards Using ATEasy®

---

The GXPIO package is supplied with a separate ATEasy driver for each of board types. The **GX5732.drv** ATEasy driver uses the GXPIO.DLL to program the board. In addition, each driver is supplied with an example that contains a program and a system file pre-configured with the ATEasy driver. Use the driver shortcut property page from the System Drivers sub-module to change the PCI slot number before attempting to run the example.

Using commands declared in the ATEasy driver are easier to use than using the DLL functions directly. The driver commands will also generate exception that allows the ATEasy application to trap errors without checking the status code returned by the DLL function after each function call.

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

- The *nHandle* parameter is omitted. The driver handles this parameter automatically. ATEasy uses driver logical names instead i.e. PIO1, PIO2 for GX5732.
- The *nStatus* parameter was omitted. Use the Get Status commands instead of checking the status. After calling a DLL function the ATEasy driver will check the returned status and will call the error statement (in case of an error status) to generate exception that can be easily trapped by the application using the **OnError** module event or using the **try-catch** statement.

Some ATEasy drivers contain additional commands to permit easier access to the board features. For example parameters for a function may be omitted by using a command item instead of typing the parameter value. The commands are self-documented. Their syntax is similar to English. In addition, you may generate the commands from the code editor context menu or by using the ATEasy's code completion feature instead of typing them directly.

## Using the GXPIO driver functions

---

The GXPIO driver contains a set of functions for each of the supported instrument boards. The function name also starts with the board type. For example the GX5732 board uses the Gx5732Xxxx functions. Other functions are available as general purpose functions that apply to all boards (i.e **GxPioGetDriverSummary**).

Each of the board types has similar functions that initialize the board driver, reset the board, and display the instrument virtual panel. In addition, all the board types use handles (see below) to access the boards and use the same error handling method. The following paragraphs describe the steps required to program the boards.

## Initialization, HW Slot Numbers and VISA Resource

The GXPIO driver supports two device drivers HW and VISA which are used to initialize, identify and control the board. The user can use the **Gx5732Initialize** to initialize the board's driver using HW and **Gx5732InitializeVisa** to initialize using VISA. The following describes the two different methods used to initialize:

1. **Marvin Test Solutions' HW** – This is the default device driver that is installed by the GXPIO driver. To initialize and control the board using the HW use the **Gx5732Initialize**(*nSlot*, *pnHandle*, *pnStatus*) function. The function initializes the driver for the board at the specified PXI slot number (*nSlot*) and returns boards handle. The **PXI/PCI Explorer** applet in the Windows Control Panel displays the PXI slot assignments. You can specify the *nSlot* parameter in the following way:
  - A combination of chassis number (chassis # x 256) with the chassis slot number, e.g. 0x105 for chassis 1 and slot 5. The chassis number can be set by the **PXI/PCI Explorer** applet.
  - Legacy nSlot is used by earlier versions of HW/VISA. The slot number contains no chassis number and can be changed using the **PXI/PCI Explorer** applet: 23 in this example.

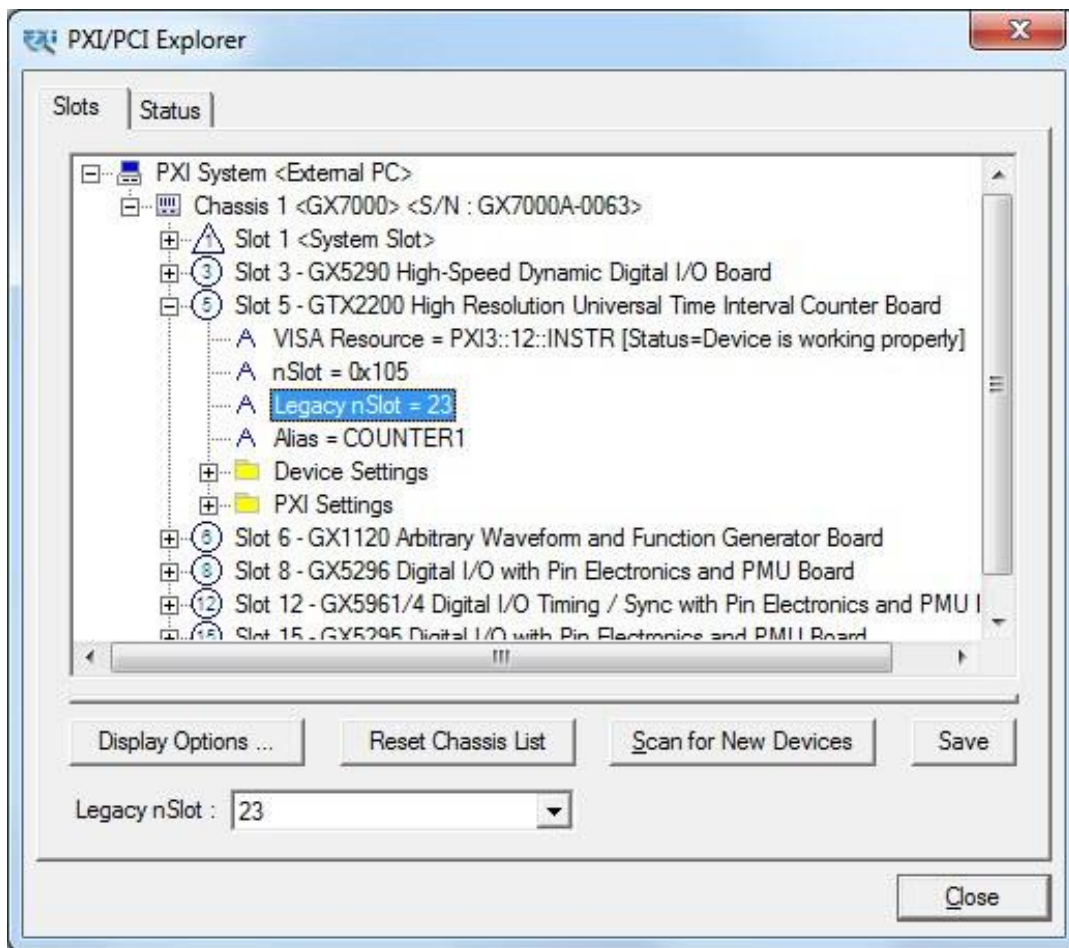


Figure 5-1: PXI/PCI Explorer

2. **VISA** – This is a third party library usually supplied by National Instruments (NI-VISA). You must ensure that the VISA installed supports PXI and PCI devices (not all VISA providers supports PXI/PCI). GXPIO setup installs a VISA compatible driver for the GXPIO board in-order to be recognized by the VISA provider. Use the GXPIO function **Gx5732InitializeVisa** (*szVisaResource*, *pnHandle*, *pnStatus*) to initialize the driver's board using VISA. The first argument *szVisaResource* is a string that is displayed by the VISA resource manager such as NI **Measurement and Automation** (NI\_MAX). It is also displayed by Marvin Test Solutions **PXI/PCI**

**Explorer** as shown in the prior figure. The VISA resource string can be specified in several ways as the following examples demonstrate:

- Using chassis, slot: “PXI0::CHASSIS1::SLOT5”
- Using the PCI Bus/Device combination: “PXI9::13::INSTR” (bus 9, device 9).
- Using the alias: for example “COUNTER1”. Use the PXI/PCI Explorer to set the device alias.

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

## Board Handle

The **Gx5732Initialize** and the **Gx5732InitializeVisa** functions return a handle that is required by other driver functions in order to program the board. This handle is usually saved in the program as a global variable for later use when calling other functions. The initialize functions do not change the state of the board or its settings.

## Reset

The Reset function sets the board to a known default state. A reset is usually performed after the board is initialized. See the Function Reference for more information regarding the reset function.

## Error Handling

All the GXPIO functions returns status - *pnStatus* - in the last parameter. This parameter can be later used for error handling. The status is zero for success status or less than zero for errors. When the status is error, the program can call the **GxPioGetErrorString** function to return a string representing the error. The **GxPioGetErrorString** reference contains possible error numbers and their associated error strings.

## Driver Version

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

## Panel

---

Calling the **Gx5732Panel** will display the instrument front panel in a window. The panel can be used to initialize 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 **Gx5732Panel** function is also used by the GXPIOPANEL.EXE or GXPIOPANEL64.EXE panel programs that is supplied with this package and provides a standalone Windows application that displays the instrument panel.

## Programming Examples

---

The README.txt located on the GXPIO folder contains a list of the GXPIO programming examples provided with the GXPIO software. Examples are provided for various programming languages including C, VB.NET, VB (6.0), ATEasy and more.

## Distributing the Driver

---

Once the application is developed, the driver files (GXPIO.DLL or GXPIO64.DLL and the HW device driver files located in the HW folder) can be shipped with the application. Typically, the DLLs should be copied to the Windows System directory. The HW device driver files should be installed using a special setup program HWSETUP.EXE that is provided with GXPIO driver files. Alternatively, you can provide the GXPIO disk to be installed along with the board.



## Sample Programs

---

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

To run, Enter the following command line:

**GXPIOEX** <PciSlot> <specific\_command>

Where:

<slot>	PCI/PXI Explorer slot number where the board reside.
<port   counter numbers>	The digital i/o port number (0-6) or the counter number (0-3).
<operation>	Operation code: RD=read port direction WD=write port direction RP=read port value      WP=write port value      RC=read counter WC=write counter
<direction   value   count>	Direction for WP operation: 0-15=for the port 4 groups. Each bit controls the direction of a byte of the port. 1 for output, 0 for input. Value for WP operation: 0 - 0xFFFFFFFF for port value Count for WC operation: 0 - 255 for counter value

## Sample Program Listing

---

```

/*****
FILE      : GxPIOExample.cpp
PURPOSE   : WIN32 sample program for GX5731 and GX5732
            boards using the GXPIO driver.
CREATED   : March 2003
COPYRIGHT : Copyright 2003 MARVIN TEST SOLUTIONS, Inc.
COMMENTS  :
To compile the WIN32 example:
1. Microsoft VC++
    Load GxPIOExampleC.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 GxPIOExampleC.bpr from the Project/Open
    Project... menu
    Select Project/Build all from the menu

*****/

#include "windows.h"
#include "GxPIO.h"
#include "stdio.h"

// Borland C++ Builder compat. block
#ifdef __BORLANDC__
#pragma hdrstop
#include <condefs.h>
USELIB("GxPIOBC.lib");
USERC("GxPioExampleC.rc");
//-----
#endif // defined(__BORLANDC__)
/*****
//      DisplayMsg
//*****/
void DisplayMsg(PSTR lpszMsg)
{
    MessageBeep(0);
    MessageBox(0, lpszMsg, "GXPIO", MB_OK);
    return;
}
/*****
//      DisplayUsage
//*****/
void DisplayUsage(void)
{
    DisplayMsg(
        "This example shows how to use the GX5731 or GX5732\r\n"
        "Usage:\r\n"
        "GxPIOExample <board_type> <slot_number> <port|counter numbers> <operation>\r\n"
        "<direction|value|count>\r\n\r\n"
        "Where: \r\n\r\n"
        "<board_type> - Board type: 5731=GX5731, 5732=GX5732\r\n\r\n"
    );
}

```



```

//      RP=read port value
//      WP=write port value
//      RC=read counter
//      WC=write counter
//      RB=read buffer
//      WB=write buffer
//      RBF=read buffer empty/full flag
//      SCLK_SRC=set internal clock source and divider
//      GCLK_SRC=get internal clock source
//      SCLK_DIV=set internal clock divider
//      GCLK_DIV=Get internal clock divider
// Direction/value/counter/clock number (depends on operation):
//      0 - 15 for direction. This is 4-bits direction.
//      Each bit controls the direction of a byte of the port. 1 for
//      output, 0 for input.
//      0 - 0xFFFFFFFF for value
//      0 - 255 for counter
//      0 - 1 for Internal clock 0 or 1
//*****
int main(int argc, char **argv)
{
    SHORT    nSlotNum;           // Board slot number
    SHORT    nBoardType;        // Board type
    char*    sOperation;        // Board Operation
    SHORT    nPortOrCount;      // Port or counter number
    DWORD    dwValue;           // Direction, port value or count
    DWORD    dwSize;            // Write/Read buffer data Size
    BOOL     bTerminalCountState; // returned terminal count state
    SHORT    nHandle;           // Board handle
    SHORT    nStatus;           // Returned status
    DWORD    dwDivider;         // Read Clock Divider
    SHORT    nSource;           // Read Clock Source

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

    // Parse command line parameters
    nSlotNum=(SHORT)strtol(++argv, NULL, 0);
    nBoardType=(SHORT)strtol(++argv, NULL, 0);
    nPortOrCount=(SHORT)strtol(++argv, NULL, 0);
    sOperation = strdup(++argv);

    // Check parameters
    if (nBoardType!=5731 && nBoardType!=5732) DisplayUsage();
    if (nPortOrCount<0 || nPortOrCount>7) DisplayUsage();

    // Borad type is GX5731
    if (nBoardType==5731)
    {
        Gx5731Initialize(nSlotNum, &nHandle, &nStatus);
        CheckStatus(nStatus);

        if(!strcmp(sOperation, "WD")) // Write port direction
        {
            if (nPortOrCount>6) DisplayUsage();
            if (argc<5) DisplayUsage();

```

```

        dwValue=(DWORD)strtol(++argv, NULL, 0);
        Gx5731SetPortDirection(nHandle, nPortOrCount, SHORT(dwValue), &nStatus);
        CheckStatus(nStatus);
        printf("Writes port direction OK\n");
    }
    else if (!strcmp(sOperation, "RD")) // read port direction
    {
        if (nPortOrCount>6) DisplayUsage();
        Gx5731GetPortDirection(nHandle, nPortOrCount, ((PSHORT)&dwValue), &nStatus);
        CheckStatus(nStatus);
        printf("Reads port direction: %d\n", SHORT(dwValue));
    }
    else if (!strcmp(sOperation, "WP")) // write port data
    {
        if (nPortOrCount>6) DisplayUsage();
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(++argv, NULL, 0);
        Gx5731SetPort(nHandle, nPortOrCount, dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Writes port value OK\n");
    }
    else if (!strcmp(sOperation, "RP")) // read port data
    {
        if (nPortOrCount>6) DisplayUsage();
        Gx5731GetPort(nHandle, nPortOrCount, &dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Reads port value: 0x%X\n", dwValue);
    }
}

////////////////////////////////////
else if (!strcmp(sOperation, "WB")) // write buffer
{
    if (argc<5 || (nPortOrCount<0 || nPortOrCount>2)) DisplayUsage();
    dwValue=(DWORD)strtol(++argv, NULL, 0);
    dwSize=1;
    Gx5731ModuleBufferWriteData(nHandle, nPortOrCount, &dwValue, &dwSize, &nStatus);
    CheckStatus(nStatus);
    printf("Writes data to buffer OK\n");
}
else if (!strcmp(sOperation, "RB")) // read buffer
{
    if (nPortOrCount<0 || nPortOrCount>2) DisplayUsage();
    dwSize=1;
    Gx5731ModuleBufferReadData(nHandle, nPortOrCount, &dwValue, &dwSize, &nStatus);
    CheckStatus(nStatus);
    printf("Reads data to buffer: %d\n", dwValue);
}
else if (!strcmp(sOperation, "RBF")) // read buffer empty/full flag
{
    if (argc<5 || (nPortOrCount<0 || nPortOrCount>2)) DisplayUsage();
    dwValue=(DWORD)strtol(++argv, NULL, 0);
    Gx5731ModuleBufferGetState(nHandle, nPortOrCount, ((PSHORT)&dwValue), &nStatus);
    CheckStatus(nStatus);
    printf("Module Buffer Flag state is: %d\n", dwValue);
}
else if (!strcmp(sOperation, "SCLK_SRC")) // set internal clock source
{
    if (argc<5 || (nPortOrCount<0 || nPortOrCount>2)) DisplayUsage();
    dwValue=(DWORD)strtol(++argv, NULL, 0);
    Gx5731GetInternalClock(nHandle, nPortOrCount, &nSource, &dwDivider, &nStatus);

```

```

        Gx5731SetInternalClock(nHandle, nPortOrCount, SHORT(dwValue), dwDivider, &nStatus);
        CheckStatus(nStatus);
        printf("Set internal clock source OK\n");
    }
    else if (!strcmp(sOperation, "GCLK_SRC"))    // get internal clock source
    {
        if (nPortOrCount<0 || nPortOrCount>2) DisplayUsage();
        Gx5731GetInternalClock(nHandle, nPortOrCount, ((PSHORT)&dwValue), &dwDivider,
            &nStatus);
        CheckStatus(nStatus);
        printf("Internal clock source is: %d\n", dwValue);
    }
    else if (!strcmp(sOperation, "SCLK_DIV"))    // set internal clock divider
    {
        if (argc<5 || (nPortOrCount<0 || nPortOrCount>2)) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5731GetInternalClock(nHandle, nPortOrCount, &nSource, &dwDivider, &nStatus);
        Gx5731SetInternalClock(nHandle, nPortOrCount, nSource, dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Set internal clock divider OK\n");
    }
    else if (!strcmp(sOperation, "GCLK_DIV"))    // get internal clock divider
    {
        if (nPortOrCount<0 || nPortOrCount>2) DisplayUsage();
        Gx5731GetInternalClock(nHandle, nPortOrCount, &nSource, &dwDivider, &nStatus);
        CheckStatus(nStatus);
        printf("Internal clock divider is: %d\n", dwDivider);
    }
    else
        DisplayUsage();
}
// Borad type is GX5732
else
{
    Gx5732Initialize(nSlotNum, &nHandle, &nStatus);
    CheckStatus(nStatus);

    if(!strcmp(sOperation, "WD")) // Write port direction
    {
        if (nPortOrCount>6) DisplayUsage();
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5732SetPortDirection(nHandle, nPortOrCount, SHORT(dwValue), &nStatus);
        CheckStatus(nStatus);
        printf("Writes port direction OK\n");
    }
    else if (!strcmp(sOperation, "RD")) // read port direction
    {
        if (nPortOrCount>6) DisplayUsage();
        Gx5732GetPortDirection(nHandle, nPortOrCount, ((PSHORT)&dwValue), &nStatus);
        CheckStatus(nStatus);
        printf("Reads port direction: %d\n", SHORT(dwValue));
    }
    else if (!strcmp(sOperation, "WP")) // write port data
    {
        if (nPortOrCount>6) DisplayUsage();
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5732SetPort(nHandle, nPortOrCount, dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Writes port value OK\n");
    }
}

```

```

    }
    else if (!strcmp(sOperation, "RP")) // read port data
    {
        if (nPortOrCount>6) DisplayUsage();
        Gx5732GetPort(nHandle, nPortOrCount, &dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Reads port value: 0x%X\n", dwValue);
    }
    else if (!strcmp(sOperation, "WC")) // write counter
    {
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(++argv, NULL, 0);
        Gx5732SetCounterValue(nHandle, nPortOrCount, (BYTE)dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Writes counter OK\n");
    }
    else if (!strcmp(sOperation, "RC")) // read counter
    {
        Gx5732GetCounterValue(nHandle, nPortOrCount, (BYTE*)&dwValue, &bTerminalCountState,
            &nStatus);
        CheckStatus(nStatus);
        printf("Reads counter: %d, terminal count state: %d\n", (BYTE)dwValue,
            bTerminalCountState);
    }
    else
        DisplayUsage();
}
return 0;
}
//*****
//      End Of File
//*****

```





## Chapter 6 - Functions Reference

### Introduction

---

The GX5732 driver functions reference chapter is organized in alphabetical order. Each function description contains the function name; purpose, syntax, parameters description and type followed by Comments, an Example (written in C), and a See Also sections.

All function parameters syntax follows the same rules:

- Strings are ASCIIZ (null or zero character terminated).
- Most function's first parameter is *nHandle* (16-bit integer). This parameter is required for operating the board and is returned by the **Gx5732Initialize** or the **Gx5732InitializeVisa** functions. The *nHandle* is used to identify the board when calling a function for programming and controlling the operation of that board.
- All functions return a status with the last parameter named *pnStatus*. The *pnStatus* is zero if the function was successful, or less than a zero on error. The description of the error is available using the **GxPioGetErrorString** function or by using a predefined constant, defined in the driver interface files: GXPIO.H, GXPIO.BAS, GXPIO.VB, GXPIO.PAS or GX5732.DRV.
- Parameter name are prefixed as follows:

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

**Table 6-1 : Parameter Name Prefixes**

## GX5732 Functions

The following list is a summary of functions available for the GX5732:

Driver Functions	Description
<b>General</b>	
<b>Gx5732GetBoardSummary</b>	Returns the board summary.
<b>Gx5732Initialize</b>	Initializes the driver for the board at the specified slot number using HW. The function returns a handle that can be used with other GX5732 functions to program the board
<b>Gx5732InitializeVisa</b>	Initializes the driver for the specified slot using VISA. The function returns a handle that can be used with other GX5732 functions to program the board.
<b>Gx5732Panel</b>	Opens a virtual panel used to interactively control the GX5732.
<b>Gx5732Reset</b>	Resets the GX5732 board to its default settings.
<b>GxPioGetDriverSummary</b>	Returns the driver name and version.
<b>GxPioGetErrorString</b>	Returns the error string associated with the specified error number.
<b>Digital I/O Functions</b>	
<b>Gx5732ResetPort</b>	Rest the specified digital I/O port.
<b>Gx5732GetPort</b>	Reads the port value.
<b>Gx5732SetPort</b>	Writes a value to a port.
<b>Gx5732GetPortWord</b>	Reads a port word value.
<b>Gx5732SetPortWord</b>	Writes a value to the port word.
<b>Gx5732GetPortByte</b>	Reads a port byte value.
<b>Gx5732SetPortByte</b>	Writes a value to the port byte.
<b>Gx5732GetPortBit</b>	Reads a port bit value.
<b>Gx5732SetPortBit</b>	Writes a value to a port byte.
<b>Gx5732GetPortDirection</b>	Returns the direction of port bytes.
<b>Gx5732SetPortDirection</b>	Sets the direction for port bytes.
<b>Gx5732GetPortByteDirection</b>	Returns the direction of a port byte.
<b>Gx5732SetPortByteDirection</b>	Sets the direction of a port byte.
<b>Counters Functions</b>	
<b>Gx5732ResetCounter</b>	Resets the specified counter.
<b>Gx5732GetCounterEnable</b>	Returns whether the counter is enabled.
<b>Gx5732SetCounterEnable</b>	Enables or disables the counter to count.
<b>Gx5732GetCounterValue</b>	Return the counter value and the terminal count state.
<b>Gx5732SetCounterValue</b>	Loads the counter with a value.

Driver Functions	Description
<b>Gx5732GetCounterMode</b>	Returns whether the counter settings.
<b>Gx5732SetCounterMode</b>	Changes the counter settings. Including whether the counter runs in One-shot mode or continuously, if the counter counts up or down, and the terminal count output mode.
<b>Gx5732GetCounterClock</b>	Returns the counter clock source and polarity.
<b>Gx5732SetCounterClock</b>	Sets the counter clock source and polarity.
<b>Gx5732GetCounterGate</b>	Returns the counter gate source and polarity.
<b>Gx5732SetCounterGate</b>	Sets the counter gate source and gate polarity.
<b>Gx5732GetTerminalCountPortConnection</b>	Returns which counter or all counters terminal count is connected to the Terminal Count Port
<b>Gx5732SetTerminalCountPortConnection</b>	Connects the specified counter or all counters terminal count to the Terminal Count Port
<b>Gx5732GetCounterPort</b>	Returns the Counter Port settings. Including the Counter Port direction (input or Output), The counter connected to it and the port load control.
<b>Gx5732SetCounterPort</b>	Changes the Counter Port settings. Including the Counter Port direction (input or output), the counter connected to it and the port load control.
<b>Gx5732LoadCounterCounterPort</b>	Loads the Counter with the Counter Port value
<b>Gx5732GetInternalClock</b>	Returns the internal clock source and divider.
<b>Gx5732SetInternalClock</b>	Sets internal clock source and divider.

## Gx5732GetBoardSummary

---

### Purpose

Returns the board summary.

### Syntax

**Gx5732GetBoardSummary** (*nHandle*, *szSummary*, *nSumMaxLen*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>szSummary</i>	PSTR	Buffer to contain the returned board info (null terminated) string.
<i>nSumMaxLen</i>	SHORT	Size of the buffer to contain the error string.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The GX5732 summary string provides the following data:

- Instrument name (e.g., GX5731)
- FPGA version of main board (e.g. 0xA003)

For example, the returned string could look like the following:

```
GX5731, FPGA Version:0xA000
```

### Example

The following example returns the board summary:

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

Gx5732GetBoardSummary(nHandle, szSummary, sizeof(szSummary), &nStatus);
```

### See Also

**GxPioGetDriverSummary, Gx5732Initialize, GxPioGetErrorString**

## Gx5732GetCounterClock

---

### Purpose

Returns the counter clock source and polarity.

### Syntax

**Gx5732GetCounterClock** (*nHandle*, *nCounter*, *pnClockSource*, *pbClockPolarity*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>pnClockSource</i>	PSHORT	Returned counter clock source: 0. GX5731_CLOCK0: Internal clock 0 1. GX5731_CLOCK1: Internal clock 1 2. GX5732_COUNTER_TO_USER_LINE0: User Line 0. 3. GX5732_COUNTER_TO_USER_LINE1: User Line 1. 4. GX5732_COUNTER_TO_USER_LINE2: User Line 2. 5. GX5732_COUNTER_TO_USER_LINE3: User Line 3. 6. GX5732_COUNTER_TO_USER_LINE4: User Line 4. 7. GX5732_COUNTER_TO_USER_LINE5: User Line 5.
<i>pbClockPolarity</i>	PBOOL	Returned clock polarity: 0. GX5732_POSITIVE_EDGE: Count on rising clock edge. 1. GX5732_NEGATIVE_EDGE: Count on falling clock edge.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetCounterClock** to set the clock source and polarity.

### Example

The following example returns the clock source and polarity for counter 0:

```
SHORT nClockSource, nStatus;
BOOL bClockPolarity;
Gx5732GetCounterClock (nHandle, GX5732_COUNTER1, &nClockSource, &bClockPolarity, &nStatus);
```

### See Also

**Gx5732SetCounterClock**, **Gx5732SetInternalClock**, **Gx5732SetCounterEnable**

## Gx5732GetCounterEnable

---

### Purpose

Returns whether the counter is enabled.

### Syntax

**Gx5732GetCounterEnable** (*nHandle*, *nCounter*, *pbEnable*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. GX5732_COUNTER3: Counter 3.
<i>pbEnable</i>	PBOOL	Enabled state: 0. FALSE: Counter is disable 1. TRUE: Counter is enabled.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetCounterEnable** to enable the counter.

### Example

The following check if counter 3 is enabled:

```
SHORT nStatus;
BOOL bEnable;
Gx5732GetCounterEnable (nHandle, GX5732_COUNTER3, &bEnable, &nStatus);
```

### See Also

**Gx5732SetCounterEnable**

## Gx5732GetCounterGate

---

### Purpose

Returns the counter gate source and polarity.

### Syntax

**Gx5732GetCounterGate** (*nHandle*, *nCounter*, *pnGateSource*, *pbGatePolarity*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>pnGateSource</i>	PSHORT	Returned gate source: 0. GX5732_COUNTER_GATE_ENABLE_ALWAYS: Always enabled. 1. GX5732_COUNTER_GATE_CARRY_COUNTER1: Terminal Count Carryout. 2. GX5732_COUNTER_GATE_USER_LINE0: User Line 0. 3. GX5732_COUNTER_GATE_USER_LINE1: User Line 1. 4. GX5732_COUNTER_GATE_USER_LINE2: User Line 2. 5. GX5732_COUNTER_GATE_USER_LINE3: User Line 3. 6. GX5732_COUNTER_GATE_USER_LINE4: User Line 4. 7. GX5732_COUNTER_GATE_USER_LINE5: User Line 5.
<i>pbGatePolarity</i>	PBOOL	Returned gate polarity: 0. GX5732_POSITIVE_EDGE: Count when gate signal is high. 1. GX5732_NEGATIVE_EDGE: Count when gate signal is low.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetCounterGate** to setup the counter gate.

### Example

The following example reads counter 3 gate settings:

```
SHORT nGateSource, nGatePolarity, nStatus;
Gx5732GetCounterGate (nHandle, GX5732_COUNTER3, &nGateSource, &nGatePolarity, &nStatus);
```

### See Also

**Gx5732SetCounterGate**, **Gx5732SetCounterClock**, **Gx5732SetCounterEnable**

## Gx5732GetCounterMode

---

### Purpose

Returns the counter settings and mode.

### Syntax

**Gx5732GetCounterMode** (*nHandle*, *nCounter*, *pbOneShot*, *pbUpDown*, *pnTerminalCounterMode*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>pbOneShot</i>	PBOOL	Returned mode: 0. FALSE: Continuous. 1. TRUE: One shot.
<i>pbUpDown</i>	PBOOL	Returned count direction: 0. FALSE: Up. 1. TRUE: Down.
<i>pnTerminalCounterMode</i>	PSHORT	Returned terminal count mode: 0. GX5732_TERMINAL_COUNT_POS_PULSE: Positive pulse. 1. GX5732_TERMINAL_COUNT_NEG_PULSE: Negative pulse. 2. GX5732_TERMINAL_COUNT_POS_SQUARE: Toggle output, start with low. 3. GX5732_TERMINAL_COUNT_NEG_SQUARE: Toggle output, start with high.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetCounterMode** to setup the counter modes.

The function returns whether the counter runs in One-shot mode or continuously, if the counter counts up or down, and the terminal count output mode.

### Example

The following example retrieves counter 3 settings:

```
BOOL bOneShot, bDown, nStatus;
SHORT nTerminalCount;
Gx5732GetCounterMode (nHandle, GX5732_COUNTER3, &bOneShot, &bDown, &nTerminalCount, &nStatus);
```

### See Also

**Gx5732SetCounterMode**, **Gx5732SetCounterEnable**



## Gx5732GetCounterPort

---

### Purpose

Returns the counter connected to the Counter Port, and direction and the loading mode of the port.

### Syntax

**Gx5732GetCounterPort** (*nHandle*, *pnCounter*, *pbInOut*, *pnInLoadControl*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>pbInOut</i>	PBOOL	Returned Counter Port direction: 0. GX5732_COUNTER_PORT_DIRECTION_IN: Input 1. GX5732_COUNTER_PORT_DIRECTION_OUT: Output
<i>pnCounter</i>	PSHORT	Returned Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>pnInLoadControl</i>	PSHORT	When input, returns the signal causes the Counter Port to load: 0. GX5732_COUNTER_PORT_LOAD_CONTROL_NONE: None. 1. GX5732_COUNTER_PORT_LOAD_CONTROL_IMMEDIATE: load Immediate (software control). 2. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE0: User Line 0. 3. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE1: User Line 1. 4. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE2: User Line 2. 5. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE3: User Line 3. 6. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE4: User Line 4. 7. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE5: User Line 5.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetCounterPort** to change these settings.

### Example

The following example reads the Counter Port settings:

```
SHORT nCounter, nInLoadControl, nStatus;
BOOL bInOut;
Gx5732GetCounterPort(nHandle, &nCounter, &bInOut, &nInLoadControl, &nStatus);
```

### See Also

**Gx5732SetCounterPort**, **Gx5732LoadCounterCounterPort**

## Gx5732GetCounterValue

---

### Purpose

Returns the counter value and the terminal count state.

### Syntax

**Gx5732GetCounterValue** (*nHandle* *nCounter*, *pucValue*, *pbTerminalCountState*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>pucValue</i>	PBYTE	Returned counter value: 0-255
<i>pbTerminalCountState</i>	PBOOL	Returned terminal counter state: 0. FALSE: Not overflow. 1. TRUE: Overflow.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetCounterValue** to initialize the counter value.

### Examples

The following example reads counter 2 value:

```
SHORT nStatus;
BOOL bTCState;
BYTE ucValue;
Gx5732GetCounterValue (nHandle, GX5732_COUNTER2, &ucValue, &bTCState, &nStatus);
```

### See Also

**Gx5732SetCounterValue**

## Gx5732GetInternalClock

---

### Purpose

Returns the internal clock source and divider.

### Syntax

**Gx5732GetInternalClock** (*nHandle*, *nClock*, *pnSource*, *pdwDivider*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nClock</i>	SHORT	Clock number: 0. GX5731_CLOCK0: Clock 0. 1. GX5731_CLOCK1: Clock 1.
<i>pnSource</i>	PSHORT	Returned source: 0. GX5732_INTERNAL_CLOCK_T0_PCI_CLOCK_33MHZ: 33MHz PCI Clock. 1. GX5732_INTERNAL_CLOCK_T0_PXI_CLOCK_10MHZ: 10MHz PXI Clock. 2. GX5732_INTERNAL_CLOCK_T0_USER_LINE0: User Line 0. 3. GX5732_INTERNAL_CLOCK_T0_USER_LINE1: User Line 1. 4. GX5732_INTERNAL_CLOCK_T0_USER_LINE2: User Line 2. 5. GX5732_INTERNAL_CLOCK_T0_USER_LINE3: User Line 3. 6. GX5732_INTERNAL_CLOCK_T0_USER_LINE4: User Line 4. 7. GX5732_INTERNAL_CLOCK_T0_USER_LINE5: User Line 5.
<i>pdwDivider</i>	PDWORD	Divider value: 1-0x1000000 (16777216)
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetInternalClock** to setup the internal clock.

### Example

The following example reads clock 0 setup:

```
SHORT nSource, nStatus;
DWORD dwDivider;
Gx5732GetInternalClock (nHandle, GX5731_CLOCK0, &nSource, &dwDivider, &nStatus);
```

### See Also

**Gx5732SetInternalClock**, **Gx5732SetCounterClock**

## Gx5732GetPort

---

### Purpose

Reads a port value.

### Syntax

**Gx5732GetPort** (*nHandle*, *nPort*, *pdwValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>dwValue</i>	PDWORD	Returned port value, 32 bit integer where each bit represents a channel. Bit 0 for channel 0 and bit 31 for channel 31.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetPort** to write data to the port when in Output direction.

### Example

The following example reads port 1 value:

```
SHORT nStatus;
DWORD dwValue;
Gx5732GetPort (nHandle, GX5732_PORT1, &dwValue, &nStatus);
```

### See Also

**Gx5732SetPort**, **Gx5732GetPortBit**, **Gx5732GetPortByte**, **Gx5732GetPortWord**, **Gx5732SetPortDirection**

## Gx5732GetPortBit

---

### Purpose

Reads a port bit value.

### Syntax

**Gx5732GetPortBit** (*nHandle*, *nPort*, *nBit*, *pucValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>nBit</i>	SHORT	Port bit number: 0-31. Where 0 is used for low order bit and 31 for the high order bit.
<i>pucValue</i>	PBYTE	Returned channel value: 0 for low state and 1 for hi state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetPortBit** to write data to the port bit when in Output direction.

### Example

The following example reads port 1, bit 30 value:

```
SHORT nStatus;
BYTE ucValue;
Gx5732GetPortBit(nHandle, GX5732_PORT1, 30, &ucValue, &nStatus);
```

### See Also

**Gx5732SetPortBit**, **Gx5732GetPortByte**, **Gx5732GetPortWord**, **Gx5732GetPort**

## Gx5732GetPortByte

---

### Purpose

Reads a port byte value.

### Syntax

**Gx5732GetPortByte** (*nHandle*, *nPort*, *nByte*, *pucValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>nByte</i>	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte. 0. GX5732_PORT_BYTE0 1. GX5732_PORT_BYTE1 2. GX5732_PORT_BYTE2 3. GX5732_PORT_BYTE3
<i>ucValue</i>	PBYTE	Returned value 0-255. Each bit represents the specific channel state bit 0 for lowest channel number.
<i>pnStatus</i>	LPSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetPortByte** to write data to the port byte when in Output direction.

### Example

The following example reads port 1, byte 2 value:

```
SHORT nStatus;
BYTE ucValue;
Gx5732GetPortByte (nHandle, GX5732_PORT1, GX5732_PORT_BYTE2, &ucValue, &nStatus);
```

### See Also

**Gx5732SetPortByte**, **Gx5732GetPortBit**, **Gx5732GetPortWord**, **Gx5732GetPort**

## Gx5732GetPortByteDirection

---

### Purpose

Returns the direction of a port byte.

### Syntax

**Gx5732GetPortByteDirection** (*nHandle*, *nPort*, *nByte*, *bInOut*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>Nhandle</i>	SHORT	Handle to a GX5732 board.
<i>NPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>Nbyte</i>	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte: 0. GX5732_PORT_BYTE0 1. GX5732_PORT_BYTE1 2. GX5732_PORT_BYTE2 3. GX5732_PORT_BYTE3
<i>BInOut</i>	PBOOL	Returned Byte direction: 0. FALSE: Input. 1. TRUE: Output.
<i>PnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Each digital I/O port is divided to four groups or bytes. Each group contains eight channels and can have its own direction, input or output.

### Example

The following example returns the byte direction of port 1, byte 3:

```
SHORT nStatus;
BOOL bOut;
Gx5732GetPortByteDirection (nHandle, GX5732_PORT1, GX5732_PORT_BYTE3, &bOut, nStatus);
```

### See Also

**Gx5732SetPortByteDirection**, **Gx5732GetPortDirection**

## Gx5732GetPortDirection

---

### Purpose

Returns the direction of port bytes.

### Syntax

**Gx5732GetPortDirection** (*nHandle*, *nPort*, *pnDirection*, *pnStatus*)

### Parameters

Name	Type	Description
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>pnDirection</i>	PSHORT	Returned port bytes directions where bit 0 corresponds to byte 0, bit 1 to byte 1, bit 2 byte 2 and bit 3 to byte 0. A bit is set to hi '1' for output and low '0' for input.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Each digital I/O port is divided to four groups or bytes. Each group contains eight channels and can have its own direction, input or output.

### Example

The following example returns the direction of port 1:

```
SHORT nDirection, nStatus;
Gx5732GetPortByteDirection (nHandle, GX5732_PORT1, &nDirection, &nStatus);
```

### See Also

**Gx5732SetPortDirection, Gx5732GetPortByteDirection**



## Gx5732GetPortWord

---

### Purpose

Reads a port word value.

### Syntax

**Gx5732GetPortWord** (*nHandle*, *nPort*, *nWord*, *pwValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>nWord</i>	SHORT	Port word number: 0. GX5732_PORT_LOW_WORD: low order word (bytes 0 and 1). 1. GX5732_PORT_HIGH_WORD: high order word (bytes 2 and 3).
<i>pwValue</i>	PWORD	Returned port word value: 0 to 65,535 (0-0xFFFF). Where bit 0 corresponds to channel 0 and bit 15 to channel 15.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetPortWord** to write data to the port word when in Output direction.

### Example

The following example reads port 1, word 0 value:

```
SHORT nStatus;
WORD wValue;
Gx5732GetPortByte (nHandle, GX5732_PORT1, GX5732_PORT_LOW_WORD, &wValue, &nStatus);
```

### See Also

**Gx5732SetPortWord**, **Gx5732GetPortBit**, **Gx5732GetPortByte**, **Gx5732GetPort**

## Gx5732GetTerminalCountPortConnection

---

### Purpose

Returns which counter or all counters terminal count is connected to the Terminal Count Port.

### Syntax

**Gx5732GetTerminalCountPortConnection** (*nHandle*, *pnCounterOrAllTerminalCountsAndClocks*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>pnCounterOrAllTerminalCountsAndClocks</i>	PSHORT	Returned Counter Port connection: 0. GX5732_TERMINAL_COUNT_OUPUT_COUNTER0: Counter 0. 1. GX5732_TERMINAL_COUNT_OUPUT_COUNTER1: Counter 1. 2. GX5732_TERMINAL_COUNT_OUPUT_COUNTER2: Counter 2. 3. GX5732_TERMINAL_COUNT_OUPUT_COUNTER3: Counter 3. 4. GX5732_TERMINAL_COUNT_OUPUT_ALLCOUNTERS_AND_CLOCKS: All counters terminal and clock 0 and 1 are connected
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetTerminalCountPortConnection** to change the Counter Port connection.

### Example

The following example returns the Terminal Port connection:

```
SHORT nConnection, nStatus;
Gx5732GetTerminalCountConnection(nHandle, &nConnection, &nStatus);
```

### See Also

**Gx5732SetTerminalCountPortConnection**, **Gx5732GetCounterValue**

## Gx5732Initialize

---

### Purpose

Initializes the GX5732 driver for the specified board slot.

### Syntax

**Gx5732Initialize** (*nSlot*, *pnHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nSlot</i>	Short	GX5732 board slot number.
<i>pnHandle</i>	PSHORT	Returned Handle for a GX5732 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, 1 on failure.

### Comments

The **Gx5732Initialize** function verifies whether or not the GX5732 board exists in the specified PXI slot. The function does not change any of the board settings. The function uses the HW driver to access and program the board.

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

The specified PXI slot number is displayed by the **PXI/PCI Explorer** applet that can be opened from the Windows **Control Panel**. You may also use the label on the chassis below the PXI slot where the board is installed. The function accepts two types of slot numbers:

- A combination of chassis number (chassis # x 256) with the chassis slot number. For example 0x105 (chassis 1 slot 5).
- Legacy nSlot as used by earlier versions of HW/VISA. The slot number contains no chassis number and can be changed using the **PXI/PCI Explorer** applet (1-255).

The returned handle *pnHandle* is used to identify the specified board with other GX5732 functions.

### Example

The following example initializes a GX5732 boards at PCI bus slot 1.

```
SHORT nHandle, nStatus;

Gx5732Initilize(1, &nHandle, &nStatus);
if (nHandle==0)
{   printf("Unable to Initialize the board")
    return;
}
```

### See Also

**GxPioGetErrorString**, **Gx5732Reset**

## Gx5732InitializeVisa

---

### Purpose

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

### Syntax

**Gx5732InitializeVisa** (*szVisaResource*, *pnHandle*, *pnStatus*)

### Parameters

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

### Comments

The **Gx5732InitializeVisa** opens a VISA session to the specified resource. The function uses the default VISA provider configured in your system to access the board. You must ensure that the default VISA provider support PXI/PCI devices and that the board is visible in the VISA resource manager before calling this function.

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

- Using chassis, slot, for example: "PXI0::CHASSIS1::SLOT5"
- Using the PCI Bus/Device combination, for example: "PXI9::13::INSTR" (bus 9, device 9).
- Using alias, for example: "COUNTER1". Use the PXI/PCI Explorer to set the device alias.

The function returns a board handle (session identifier) that can be used to call any other operations of that resource. The session is opened with VI\_TMO\_IMMEDIATE and VI\_NO\_LOCK VISA attributes. On terminating the application the driver automatically invokes **viClose()** terminating the session.

### Example

The following example initializes a GX5732 boards at PXI bus 5 and device 11.

```
SHORT nHandle, nStatus;
Gx5732InitializeVisa ("PXI5::11::INSTR", &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

### See Also

**Gx5732Initialize**, **Gx5732Reset**, **GxPIOGetErrorString**

## Gx5732LoadCounterCounterPort

---

### Purpose

Loads the Counter with the Counter Port value

### Syntax

**Gx5732LoadCounterCounterPort** (*nHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The Counter Port direction must be set to Input prior to calling this function.

### Example

The following set the counter port and loads it value to counter 1.

```
Gx5732SetCounterPort(nHandle, GX5732_COUNTER_PORT_DIRECTION_IN GX5732_COUNTER0,
                    GX5732_COUNTER_PORT_LOAD_CONTROL_IMMEDIATE, &nStatus);
Gx5732LoadCounterCounterPort(nHandle, &nStatus);
```

### See Also

**Gx5732GetCounterPort**, **Gx5732SetCounterPort**

## Gx5732Panel

---

### Purpose

Opens a virtual panel used to interactively control the GX5732.

### Syntax

**Gx5732Panel** (*pnHandle*, *hwndParent*, *nMode*, *phwndPanel*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>pnHandle</i>	PSHORT	Handle to a GX5732 board.
<i>hwndParent</i>	HWND	Panel parent window handle. A value of 0 sets the desktop as the parent window.
<i>nMode</i>	SHORT	The mode in which the panel main window is created. 0 for modeless window and 1 for modal window.
<i>phwndPanel</i>	HWND	Returned window handle for the panel.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

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

If the mode is set to modal dialog (*nMode*=1), the panel will disable the parent window (*hwndParent*) and the function will return only after the window was closed by the user. In that case, the *pnHandle* may return the handle created by the user using the panel Initialize dialog. This handle may be used when calling other GX5732 functions.

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

### Example

The following example opens the panel in modal mode:

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

Gx5732Panel(&nHandle, 0, 1, &dwPanel, &nStatus);
```

### See Also

**Gx5732Initialize**, **GxPioGetErrorString**

## Gx5732Reset

---

### Purpose

Resets the GX5732 board to its default settings.

### Syntax

**Gx5732Reset** (*nHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5732 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The function resets all digital I/O port and counters. Digital I/O ports are set to input direction. Counters are disabled and set to continuous mode, 0 value, clock source is zero, gate is enabled, counter port is set to input from counter 0 with load control set to None, Terminal Counter set to Counter 0, Clock are set to 33Mhz with divider set to 1.

### Example

The following example initializes and resets the GX5732 board:

```
Gx5732Initialize (1, &nHandle, &nStatus);  
Gx5732Reset (nHandle, &nStatus);
```

### See Also

**Gx5732Initialize**, **Gx5732ResetPort**, **Gx5732ResetCounter**

## Gx5732ResetCounter

---

### Purpose

Resets the specified counter.

### Syntax

**Gx5732ResetCounter** (*nHandle*, *nCounter*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Counters are disabled and set to continuous mode, 0 value, clock source is zero, gate is enabled, counter port is set to input from counter 0 with load control set to None, Terminal Counter set to Counter 0, Clock are set to 33Mhz with divider set to 1.

### Example

The following example resets counter 1:

```
Gx5732ResetCounter (nHandle, GX5732_COUNTER1, &nStatus);
```

### See Also

**Gx5732Reset**, **Gx5732ResetPort**, **Gx5732Initialize**



## Gx5732ResetPort

---

### Purpose

Resets the specified digital I/O port.

### Syntax

**Gx5732ResetPort** (*nHandle*, *nPort*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

After the port is reset all port byte directions are set to input state.

### Example

The following example resets port 1:

```
Gx5732ResetPort (nHandle, GX5732_PORT1, &nStatus);
```

### See Also

Gx5732Reset, Gx5732ResetCounter, Gx5732Initialize

## Gx5732SetCounterClock

---

### Purpose

Sets the counter clock source and polarity.

### Syntax

**Gx5732SetCounterClock** (*nHandle*, *nCounter*, *nClockSource*, *bClockPolarity*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>nClockSource</i>	SHORT	Counter clock source: 0. GX5731_CLOCK0: Internal clock 0 1. GX5731_CLOCK1: Internal clock 1 2. GX5732_COUNTER_TO_USER_LINE0: User Line 0. 3. GX5732_COUNTER_TO_USER_LINE1: User Line 1. 4. GX5732_COUNTER_TO_USER_LINE2: User Line 2. 5. GX5732_COUNTER_TO_USER_LINE3: User Line 3. 6. GX5732_COUNTER_TO_USER_LINE4: User Line 4. 7. GX5732_COUNTER_TO_USER_LINE5: User Line 5.
<i>bClockPolarity</i>	BOOL	Clock polarity: 0. GX5732_POSITIVE_EDGE: Count on rising clock edge. 1. GX5732_NEGATIVE_EDGE: Count on falling clock edge.
<i>pnStatus</i>	LPSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732SetCounterClock** to set the clock source and polarity.

### Example

The following example sets the clock source and polarity for counter 2:

```
Gx5732SetCounterClock (nHandle, GX5732_COUNTER2, GX5731_CLOCK1, GX5732_POSITIVE_EDGE, &nStatus);
```

### See Also

**Gx5732GetCounterClock**, **Gx5732SetInternalClock**, **Gx5732SetCounterEnable**

## Gx5732SetCounterEnable

---

### Purpose

Enables or disables the counter to count.

### Syntax

**Gx5732SetCounterEnable** (*nHandle*, *nCounter*, *pucValue*, *pbTerminalCountState*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>bEnable</i>	BOOL	Enabled state: 0. FALSE: Disable the counter. 1. TRUE: Enable the counter.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732GetCounterEnable** to retrieve whether the counter is enabled.

### Example

The following enables counter 3:

```
Gx5732SetCounterEnable (nHandle, GX5732_COUNTER3, TRUE, &nStatus);
```

### See Also

**Gx5732GetCounterEnable**

## Gx5732SetCounterGate

---

### Purpose

Sets the counter gate source and gate polarity.

### Syntax

**Gx5732SetCounterGate** (*nHandle*, *nCounter*, *nGateSource*, *bGatePolarity*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>nGateSource</i>	SHORT	Gate source: 0. GX5732_COUNTER_GATE_ENABLE_ALWAYS: Always enabled. 1. GX5732_COUNTER_GATE_CARRY_COUNTER1: Terminal Count Carryout. 2. GX5732_COUNTER_GATE_USER_LINE0: User Line 0. 3. GX5732_COUNTER_GATE_USER_LINE1: User Line 1. 4. GX5732_COUNTER_GATE_USER_LINE2: User Line 2. 5. GX5732_COUNTER_GATE_USER_LINE3: User Line 3. 6. GX5732_COUNTER_GATE_USER_LINE4: User Line 4. 7. GX5732_COUNTER_GATE_USER_LINE5: User Line 5.
<i>bGatePolarity</i>	BOOL	Gate polarity: 0. GX5732_POSITIVE_EDGE: Count when gate signal is high. 1. GX5732_NEGATIVE_EDGE: Count when gate signal is low.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732GetCounterEnable** to retrieve the counter gate settings. Use the Carryout gate source to create 16, 24 or 32 bit counter by cascading the counter.

### Example

The following creates a 16-bit counter by setting the gate of counter 1 to carryout:

```
Gx5732SetCounterGate (nHandle, GX5732_COUNTER1, GX5732_COUNTER_GATE_CARRY_COUNTER1,
                     GX5732_POSITIVE_EDGE, &nStatus);
```

### See Also

**Gx5732GetCounterGate**, **Gx5732SetCounterClock**, **Gx5732SetCounterEnable**

## Gx5732SetCounterMode

---

### Purpose

Sets the counter settings and mode.

### Syntax

**Gx5732SetCounterMode** (*nHandle*, *nCounter*, *bOneShot*, *bUpDown*, *nTerminalCounterMode*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>bOneShot</i>	BOOL	Mode: 0. FALSE: Continuous. 1. TRUE: One shot
<i>bUpDown</i>	BOOL	Count direction: 0. FALSE: Up. 1. TRUE: Down.
<i>nTerminalCounterMode</i>	SHORT	Terminal count mode: 1. GX5732_TERMINAL_COUNT_POS_PULSE: Positive pulse. 0. GX5732_TERMINAL_COUNT_NEG_PULSE: Negative pulse. 1. GX5732_TERMINAL_COUNT_POS_SQUARE: Toggle output, start with low. 2. GX5732_TERMINAL_COUNT_NEG_SQUARE: Toggle output, start with high.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732GetCounterMode** to retrieve the counter setup modes.

The function sets whether the counter runs in one-shot mode or continuously, if the counter counts up or down, and the terminal count output mode.

### Example

The following example sets counter 3 settings to be one shot, count down, and set terminal count mode to 1:

```
Gx5732SetCounterMode (nHandle, GX5732_COUNTER3, TRUE, TRUE, GX5732_TERMINAL_COUNT_NEG_PULSE,
                      &nStatus);
```

### See Also

**Gx5732GetCounterMode**, **Gx5732SetCounterEnable**

## Gx5732SetCounterPort

---

### Purpose

Connects the counter to be connected to the port, and direction and the loading mode of the port.

### Syntax

**Gx5732SetCounterPort** (*nHandle*, *bInOut*, *nCounter*, *nInLoadControl*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>bInOut</i>	PBOOL	Counter Port direction: 1. GX5732_COUNTER_PORT_DIRECTION_IN: Input 0. GX5732_COUNTER_PORT_DIRECTION_OUT: Output
<i>nCounter</i>	PSHORT	Counter number: 1. GX5732_COUNTER0: Counter 0. 0. GX5732_COUNTER1: Counter 1. 1. GX5732_COUNTER2: Counter 2. 2. GX5732_COUNTER3: Counter 3.
<i>nInLoadControl</i>	PSHORT	When input, sets the signal causes the Counter Port to load: 0. GX5732_COUNTER_PORT_LOAD_CONTROL_NONE: None. 1. GX5732_COUNTER_PORT_LOAD_CONTROL_IMMEDIATE: load Immediate (software control). 2. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE0: User Line 0. 3. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE1: User Line 1. 4. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE2: User Line 2. 5. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE3: User Line 3. 6. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE4: User Line 4. 7. GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE5: User Line 5.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732GetCounterPort** to retrieve these settings.

### Example

The following example connects counter 1 to Counter Port and set the port to input. The counter will be load when a signal received from UL0:

```
Gx5732SetCounterPort (nHandle, GX5732_COUNTER_PORT_DIRECTION_IN GX5732_COUNTER1,
GX5732_COUNTER_PORT_LOAD_CONTROL_USERLINE0, &nStatus);
```

### See Also

**Gx5732GetCounterPort**, **Gx5732LoadCounterCounterPort**

## Gx5732SetCounterValue

---

### Purpose

Loads the counter with a value.

### Syntax

**Gx5732SetCounterValue** (*nHandle* *nCounter*, *ucValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nCounter</i>	SHORT	Counter number: 0. GX5732_COUNTER0: Counter 0. 1. GX5732_COUNTER1: Counter 1. 2. GX5732_COUNTER2: Counter 2. 3. GX5732_COUNTER3: Counter 3.
<i>ucValue</i>	BYTE	Counter value: 0-255
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732GetCounterValue** to retrieve the counter value.

### Examples

The following example writes a value (250) to counter 2:

```
Gx5732SetCounterValue (nHandle, GX5732_COUNTER2, 250, &nStatus);
```

### See Also

**Gx5732GetCounterValue**, **Gx5732LoadCounterCounterPort**

## Gx5732SetInternalClock

---

### Purpose

Sets internal clock source and divider.

### Syntax

**Gx5732SetInternalClock** (*nHandle*, *nClock*, *nSource*, *dwDivider*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5732 board.
<i>nClock</i>	SHORT	Clock number: 0. GX5731_CLOCK0: Clock 0. 1. GX5731_CLOCK1: Clock 1.
<i>nSource</i>	SHORT	Internal Clock source: 0. GX5732_INTERNAL_CLOCK_T0_PCI_CLOCK_33MHZ: 33MHz PCI Clock. 1. GX5732_INTERNAL_CLOCK_T0_PXI_CLOCK_10MHZ: 10MHz PXI Clock. 2. GX5732_INTERNAL_CLOCK_T0_USER_LINE0: User Line 0. 3. GX5732_INTERNAL_CLOCK_T0_USER_LINE1: User Line 1. 4. GX5732_INTERNAL_CLOCK_T0_USER_LINE2: User Line 2. 5. GX5732_INTERNAL_CLOCK_T0_USER_LINE3: User Line 3. 6. GX5732_INTERNAL_CLOCK_T0_USER_LINE4: User Line 4. 7. GX5732_INTERNAL_CLOCK_T0_USER_LINE5: User Line 5.
<i>dwDivider</i>	DWORD	Divider value: 1-0x1000000 (16777216)
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732GetInternalClock** to retrieve the internal clock settings.

### Example

The following example sets the internal clock 0 to 10Mhz and divider to 20000:

```
Gx5732SetInternalClock (nHandle, GX5731_CLOCK0, GX5732_INTERNAL_CLOCK_T0_PXI_CLOCK_10MHZ, 20000,
                        &nStatus);
```

### See Also

**Gx5732GetInternalClock**, **Gx5732SetCounterClock**



## Gx5732SetPort

---

### Purpose

Writes whole data to a port.

### Syntax

**Gx5732SetPort** (*nHandle*, *nPort*, *dwValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>dwValue</i>	DWORD	Port value to set: 0 to 4,294,967,295 (0-0xFFFFFFFF), where each bit represents a channel. Bit 0 for channel 0 and bit 31 for channel 31.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Only bytes that are set to output direction will be written.

Use the **Gx5732GetPort** to read data from the port.

### Example

The following example writes 0xFF00FF to port 1:

```
Gx5732SetPort (nHandle, GX5732_PORT1, 0xFF00FF, &nStatus);
```

### See Also

**Gx5732GetPort**, **Gx5732SetPortBit**, **Gx5732SetPortByte**, **Gx5732SetPortWord**, **Gx5732SetPortDirection**

## Gx5732SetPortBit

---

### Purpose

Writes a specific bit of data to a port channel.

### Syntax

**Gx5732SetPortBit** (*nHandle*, *nPort*, *nBit*, *ucValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>nBit</i>	SHORT	Port channel number: 0 to 31.
<i>ucValue</i>	BYTE	Channel value: 0 for low state and 1 for hi state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Only bit that reside in a group set to output direction will be written.

Use the **Gx5732GetPortBit** to read data from a port bit.

### Example

The following example set port 1, bit 30 value to hi state:

```
Gx5732GetPort (nHandle, GX5732_PORT1, 30, 1, &nStatus);
```

### See Also

**Gx5732GetPortBit**, **Gx5732SetPortByte**, **Gx5732SetPortWord**, **Gx5732SetPort**

## Gx5732SetPortByte

---

### Purpose

Writes a specific byte of data to a port group.

### Syntax

**Gx5732SetPortByte** (*nHandle*, *nPort*, *nByte*, *ucValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>nByte</i>	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte. 0. GX5732_PORT_BYTE0 1. GX5732_PORT_BYTE1 2. GX5732_PORT_BYTE2 3. GX5732_PORT_BYTE3
<i>ucValue</i>	BYTE	Value to set 0-255. Each bit represents the specific channel state bit 0 for lowest channel number.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Only byte set to output direction will be written.

Use the **Gx5732GetPortByte** to read data from a port byte.

### Example

The following example writes 0x5 (bit 16 and 19 hi and bit 17, 18, 20-23 to low state) to port 1, byte 2 value:

```
Gx5732GetPortByte (nHandle, GX5732_PORT1, GX5732_PORT_BYTE2, 0x5, &nStatus);
```

### See Also

**Gx5732GetPortByte**, **Gx5732SetPortBit**, **Gx5732SetPortWord**, **Gx5732SetPort**

## Gx5732SetPortByteDirection

---

### Purpose

Sets a specific byte of the direction for a port group.

### Syntax

**Gx5732SetPortByteDirection** (*nHandle*, *nPort*, *nByte*, *bInOut*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>nByte</i>	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte: 0. GX5732_PORT_BYTE0 1. GX5732_PORT_BYTE1 2. GX5732_PORT_BYTE2 3. GX5732_PORT_BYTE3
<i>bInOut</i>	BOOL	Byte direction: 0. FALSE: Input. 1. TRUE: Output.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Each digital I/O port is divided to four groups or bytes. Each group contains eight channels and can have its own direction, input or output.

### Example

The following example set the byte direction of port 1, byte 3 to output:

```
Gx5732SetPortByteDirection (nHandle, GX5732_PORT1, GX5732_PORT_BYTE3, TRUE, nStatus);
```

### See Also

**Gx5732GetPortByteDirection, Gx5732SetPortDirection**

## Gx5732SetPortDirection

---

### Purpose

Sets a direction for a port groups.

### Syntax

**Gx5732SetPortDirection** (*nHandle*, *nPort*, *nDirection*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>nDirection</i>	SHORT	Port bytes directions where bit 0 corresponds to byte 0, bit 1 to byte 1, bit 2 byte 2 and bit 3 to byte 0. A bit is set to hi '1' for Input and low '0' for output.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Each digital I/O port is divided to four groups or bytes. Each group contains eight channels and can have its own direction, input or output.

### Example

The following example sets the direction of port 1, byte 0 to input and byte 1 to 3 to output:

```
Gx5732GetPortByteDirection (nHandle, GX5732_PORT1, 6, nStatus);
```

### See Also

**Gx5732GetPortDirection, Gx5732SetPortByteDirection**

## Gx5732SetPortWord

---

### Purpose

Writes a specific word of data to a port.

### Syntax

**Gx5732SetPortWord** (*nHandle*, *nPort*, *nWord*, *wValue*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0. GX5732_PORT0: Port 0. 1. GX5732_PORT1: Port 1. 2. GX5732_PORT2: Port 2. 3. GX5732_PORT3: Port 3. 4. GX5732_PORT4: Port 4. 5. GX5732_PORT5: Port 5. 6. GX5732_PORT6: Port 6.
<i>nWord</i>	SHORT	Port word number: 0. GX5732_PORT_LOW_WORD: low order word (bytes 0 and 1) 1. GX5732_PORT_HIGH_WORD: high order word (bytes 2 and 3).
<i>wValue</i>	WORD	Port word value: 0 to 65,535 (0-0xFFFF). Where bit 0 corresponds to channel 0 and bit 15 to channel 15.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Each digital I/O port contains two words. Each word contains two bytes or 16 channels. Set the word byte directions to output before using this function.

### Example

The following example writes 0xFF12 to port 1 low order word (bytes 0 and 1) value:

```
Gx5732SetPortWord (nHandle, GX5732_PORT1, GX5732_PORT_LOW_WORD, 0xFF12, &nStatus);
```

### See Also

**Gx5732GetPortWord, Gx5732SetPortBit, Gx5732SetPortByte, Gx5732SetPort**

## Gx5732SetTerminalCountPortConnection

---

### Purpose

Connects the specified counter or all counters terminal count to the Terminal Count Port.

### Syntax

**Gx5732SetTerminalCountPortConnection** (*nHandle*, *nCounterOrAllTerminalCountsAndClocks*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5732 board.
<i>nCounterOrAllTerminalCountsAndClocks</i>	SHORT	Counter Port connection: 0. GX5732_TERMINAL_COUNT_OUPUT_COUNTER0: Counter 0. 1. GX5732_TERMINAL_COUNT_OUPUT_COUNTER1: Counter 1. 2. GX5732_TERMINAL_COUNT_OUPUT_COUNTER2: Counter 2. 3. GX5732_TERMINAL_COUNT_OUPUT_COUNTER3: Counter 3. 4. GX5732_TERMINAL_COUNT_OUPUT_ALLCOUNTERS_AND_CLOCKS: All counters terminal and clock 0 and 1 are connected
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

Use the **Gx5732GetTerminalCountPortConnection** to retrieve the Counter Port connection.

### Example

The following example connects counter 2 to the Terminal Port connection:

```
Gx5732SetTerminalCountConnection (nHandle, GX5732_TERMINAL_COUNT_OUPUT_COUNTER2, &nStatus);
```

### See Also

**Gx5732GetTerminalCountPortConnection**, **Gx5732GetCounterValue**

## GxPioGetDriverSummary

---

### Purpose

Returns the driver name and version.

### Syntax

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

### Parameters

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

### Comments

The returned string is: "GXPIO Driver for GX5732. Version 1.0, Copyright © Marvin Test Solutions 2006.".

### Example

The following example prints the driver version:

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

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

### See Also

**GxPioGetErrorString**



## GxPioGetErrorString

---

### Purpose

Returns the error string associated with the specified error number.

### Syntax

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

### Parameters

Name	Type	Comments
<i>nError</i>	SHORT	Error number.
<i>pszMsg</i>	PSTR	Buffer to the returned error string.
<i>nErrorMaxLen</i>	SHORT	The size of the error string buffer.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The function returns the error string associated with the *nError* as returned from other driver functions.

The following table displays the possible error values; not all errors apply to this board type:

### Resource Errors

- 0 No error has occurred
- 1 Unable to open the HW driver. Check if HW is properly installed
- 2 Board does not exist in this slot/base address
- 3 Different board exist in the specified PCI slot/base address
- 4 PCI slot not configured properly. You may configure using the PciExplorer from the Windows Control Panel
- 5 Unable to register the PCI device
- 6 Unable to allocate system resource for the device
- 7 Unable to allocate memory
- 8 Unable to create panel
- 9 Unable to create Windows timer
- 10 Bad or Wrong board EEPROM
- 11 Not in calibration mode
- 12 Board is not calibrated
- 13 Function is not supported by the specified board

### General Parameter Errors

- 20 Invalid or unknown error number
- 21 Invalid parameter
- 22 Illegal slot number
- 23 Illegal board handle
- 24 Illegal string length
- 25 Illegal operation mode

-26 Parameter is out of the allowed range

### Parameter Errors

-40 Invalid port

-41 Invalid word

-42 Invalid byte

-43 Invalid bit

-44 Invalid counter

-45 Invalid input load control

-46 Invalid counter or all terminal counts and clocks

-47 Invalid terminal count mode

-48 Invalid clock source

-49 Invalid clock internal number

-50 Invalid clock internal source

-51 Invalid gate source

-52 Invalid clock divider value

### Example

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

```
CHAR    sz[256];
SHORT   nStatus, nHandle;
..
Gx5731Initialize (3, &Handle, &Status);
if (nStatus<0)
{   GxPioGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz); // prints the error string returns
}
```

# Index

## A

Architecture ..... 1, 5  
ATEasy ..... 11, 16, 29, 30

## B

Block Diagram ..... 5, 6  
Board Description ..... 1, 4  
Board Handle ..... 32  
Borland ..... 16, 29, 30  
Borland-Delphi ..... 30

## C

C/C++ ..... 16, 29  
C++ ..... 29  
Connectors ..... 15, 18, 20, 21, 22  
Corrupt files ..... 10  
Counter Clock ..... 26, 45, 66  
Counter Port ..... 5, 27, 49, 58, 61, 70, 79

## D

Delphi ..... 16, 29, 30  
Digital Input or Output ..... 4  
I/O ..... 4  
Directories ..... 11  
Distributing ..... 32  
Driver  
Directory ..... 11  
Files ..... 11

## E

Error-Handling ..... 32  
Example ..... 16

## F

Functions  
Boards-Supported ..... 30

## G

Gx5732GetBoardSummary ..... 44  
Gx5732GetCounterClock ..... 45  
Gx5732GetCounterEnable ..... 46

Gx5732GetCounterGate ..... 47  
Gx5732GetCounterMode ..... 48  
Gx5732GetCounterPort ..... 49  
Gx5732GetCounterValue ..... 50  
Gx5732GetInternalClock ..... 51  
Gx5732GetPort ..... 52  
Gx5732GetPortBit ..... 53  
Gx5732GetPortByte ..... 54  
Gx5732GetPortByteDirection ..... 55  
Gx5732GetPortDirection ..... 56  
Gx5732GetPortWord ..... 57  
Gx5732GetTerminalCountPortConnection ..... 58  
Gx5732Initialize ..... 24, 59  
Gx5732InitializeVisa ..... 12, 24, 60  
Gx5732LoadCounterCounterPort ..... 61  
Gx5732Panel ..... 32, 62  
Gx5732Reset ..... 63  
Gx5732ResetCounter ..... 64  
Gx5732ResetPort ..... 65  
Gx5732SetCounterClock ..... 66  
Gx5732SetCounterEnable ..... 67  
Gx5732SetCounterGate ..... 68  
Gx5732SetCounterMode ..... 69  
Gx5732SetCounterPort ..... 70  
Gx5732SetCounterValue ..... 71  
Gx5732SetInternalClock ..... 72  
Gx5732SetPort ..... 73  
Gx5732SetPortBit ..... 74  
Gx5732SetPortByte ..... 75  
Gx5732SetPortByteDirection ..... 76  
Gx5732SetPortDirection ..... 77  
Gx5732SetPortWord ..... 78  
Gx5732SetTerminalCountPortConnection ..... 79  
GXPIO ..... 1, 10  
Driver-Description ..... 29

Header-file .....	29	JP1 .....	19
Help-File-Description .....	16	JP2 .....	19
Panel-File-Description .....	16	Jumpers .....	18
Supported-Development-Tools .....	29	<b>L</b>	
GXPIO.BAS .....	16, 29	Listing .....	34
GXPIO.DLL .....	11, 16, 29, 30, 32	<b>N</b>	
GXPIO.EXE .....	10	<i>nHandle</i> .....	30
GXPIO.H .....	16, 29	<b>O</b>	
GXPIO.LIB .....	29	OnError .....	30
GXPIO.PAS .....	16, 30	<b>P</b>	
GXPIO64.DLL .....	11, 32	Panel .....	10, 16, 23, 25, 26, 27, 28, 31, 32, 62
GXPIOBC.LIB .....	16, 29	Part / Model Number .....	22
GxPioGetDriverSummary .....	30, 32, 80	Pascal .....	16, 29, 30
GxPioGetErrorString .....	32, 41, 81, 82	PCI .....	11
GXPIOPANEL.EXE .....	32	Plug & Play .....	15
GxXXXXInitialize .....	31, 32	<i>pnStatus</i> .....	41
GxXXXXInitializeVisa .....	31	port byte .....	42, 54, 55, 65, 75
<b>H</b>		Program-File-Descriptions .....	16
Handle .....	13, 14, 31, 32	Programming	
HW .....	11, 15, 29, 32	Borland-Delphi .....	30
<b>I</b>		Error-Handling .....	32
I/O .....	4, 42	Panel-Program .....	32
Installation Directories .....	11	PXI .....	3, 5, 7, 9, 12, 13, 14, 15, 31
Installation: .....	13, 15	PXI/PCI Explorer .....	12, 24, 31, 32, 59, 60
<b>J</b>		PXIeSYS.INI .....	24
J1 .....	18	PXISYS.INI .....	24
J10 .....	19	<b>R</b>	
J11 .....	19, 21	README.TXT .....	11, 16
J12 .....	19, 21	Readme-File .....	16
J13 .....	5, 19, 21	Reset .....	32
J2 .....	18	<b>S</b>	
J6 .....	18, 20	Sample .....	33, 34
J7 .....	5, 18, 21	SCSI .....	5
J8 .....	18	Setup .....	10, 11, 16
J9 .....	18	Setup Maintenance .....	10
JB16-JB24 .....	19	Setup-and-Installation .....	9
JB1-JB12 .....	19	Slot .....	9, 13, 15, 24, 31, 33

Specifications .....1, 7

System

    Directory ..... 11

System-Requirements .....9

## T

Terminal Count Port. ....58

TTL.....3, 5

## V

Virtual Panel ..... 10, 16, 23, 24, 25, 26, 27, 28, 62

    Initialize Dialog .....24

VISA..... 11, 12, 24, 31, 32, 42, 59, 60

Visual Basic.....29

Visual C++ ..... 16, 29

