GX5731

Digital I/O PXI Board
with 128 TTL Channels
and 3x32 Customizable Channels
GXPIO Software

User's Guide

Last updated June 22, 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

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

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 GX5731 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 GX5731 driver (referred in this manual **GXPIO**). Therefore, good understanding of programming development tools and languages may be necessary.

Manual Organization

The GX5731 manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX5731 manual. Lists all the supported board and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX5731 features, board description, its architecture, specifications and the panel description and operation.
Chapter 3 –Installation and Connections	Provides instructions on how to install the GX5731 Board and its accompanying GXPIO software.
Chapter 3 –Installation and Connections	Provides instruction how to open and use the instrument front panel application in order to view and control the instrument settings
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 GX5731 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 GX5731 functions. Each function is described along with its syntax, parameters, and special programming comments. Samples are given for each function.

Conventions Used in this Manual

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

	Tips that aid you in your work.

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

Chapter 2 - Overview

Introduction

The GX5731 is a 6U PXI instrument board that provides seven ports of 32 channels parallel input and output for a total of 224 channels. The seven ports are composed of four Digital I/O ports and three ports that can be configured using variety of I/O Modules interfacing different UUTs.

The four TTL level Digital I/O ports channels have TTL levels, and the direction of each group of eight channels can be programmed as Input or Output.

The three I/O Modules ports expand the ability of the GX5731 by mounting different I/O Modules. Those I/O Module enable the GX5731 to acquire or transmit 32 channels vectors each by using high-speed FIFO as well as interfacing different voltage levels, compare data on the fly and output high power signals to name a few.

All installed I/O Modules can be programmed to run using internal or external strobe. When using the internal strobe, each the I/O Module can use any of eight available signals, two Internal programmable clock signals and six external user clock lines.

When programmed for external strobe, all I/O Modules the hardware Strobed Output/Input Handshaking Signals mechanism. This Handshake mechanism allows the I/O Modules to communicate with the UUT using Handshaking protocol and control by the hardware.

Features

The GX5731 is a Digital Input or Output card with:

- Total of seven 32-bit ports for a total of 224 Input or Output channels.
- 128 I/O TTL Levels with Programmable direction (input or output) in groups of 8 pins.
- Three Digital I/O ports that can be configured using variety of I/O Modules interfacing different UUTs.
- Available I/O Modules are:
    a) GX5701: Digital Input Latch (DOL) Module
    b) GX5702: Digital Output Latch (DOL) Module
    c) GX5704: Digital Power Output (DPO) Module
    d) GX5709: RS422 Port I/O
    e) GX5711: Bidirectional LVDS-TTL Converter Module
    f) GX5712: Bidirectional RS422-TTL Converter Module

Applications

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

Board Description

The GX5731 is a 6U PXI instrument board that provides seven ports of 32 channels parallel input and output for a total of 224 channels. The seven ports are composed of four Digital I/O ports and three ports that can be configured using variety of I/O Modules interfacing different UUTs.

The four TTL level Digital I/O ports channels have TTL levels, and the direction of each group of eight channels can be programmed as Input or Output.

The three I/O Modules ports expand the ability of the GX5731 by mounting different I/O Modules. Those I/O Module enable the GX5731 to acquire or transmit 32 channels vectors by using high-speed FIFO as well as interfacing different voltage levels, compare data on the fly and output high power signals to name a few.

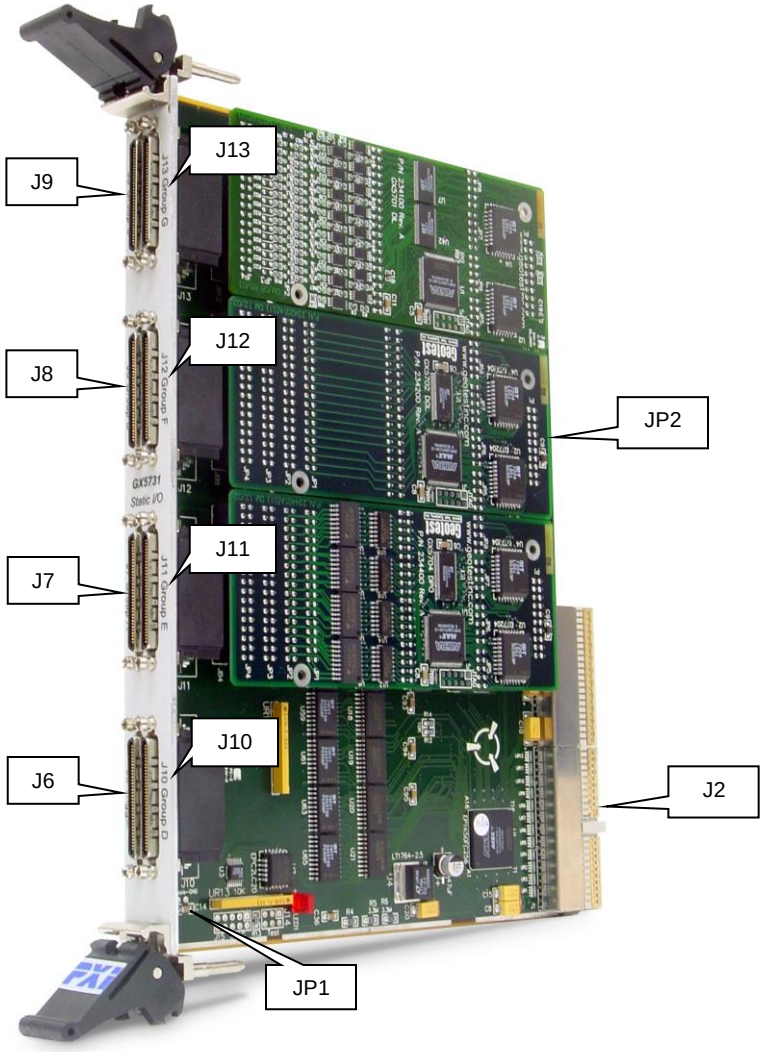

Figure 2-1: GX5731 Board Side View with three I/O Modules installed

The board has eight 68 pins SCSI connectors describes as follow:

J6	External triggers, External User Clock inputs, and internal Clock 0 and 1 output.
J7-J9	I/O Modules signals
J10 - J13	Digital I/O signals
JP1	Factory installed chassis ground jumper.
JP2	Factory installed jumper sets board VCC level (3.3V or 5V).
J1-J2	PXI Bus connector at the back of the board.

Architecture

Digital I/O

The GX5731 provides 128 digital Inputs or Outputs and 4 counters. The 128 TTL channels' direction is programmable and can be set for Input or Output in groups of eight. The GX5731 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:

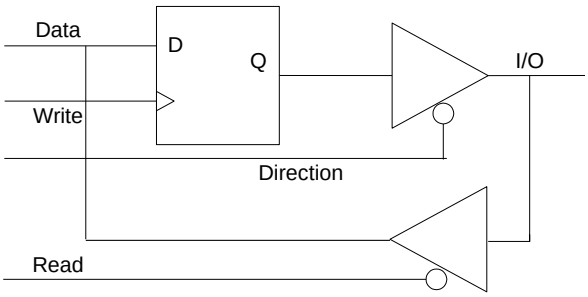

Figure 2-2: Typical I/O Channel

The GX5731 consist of four 32-bit parallel input/output ports. Each port interfaces through a 68 pins SCSI connector (J10 to J13).

Module Strobe Signals

The GX5731 can have up to three Modules installed. Each installed module has an independent set of strobe signals controlled by software and hardware. The module strobe source can be set using software to one of the following sources:

- Internal Clock0
- Internal Clock1
- User Line 0 to 5 (J6 pins 17-22)
- External Strobe Input (pin 67)

When selecting the External Strobe Input the user can control the External Strobe Polarity and External Strobe Enable Polarity for negative or positive edge.

Figure 2-3 is a block diagram for the GX5731 Module's Strobe Signals:

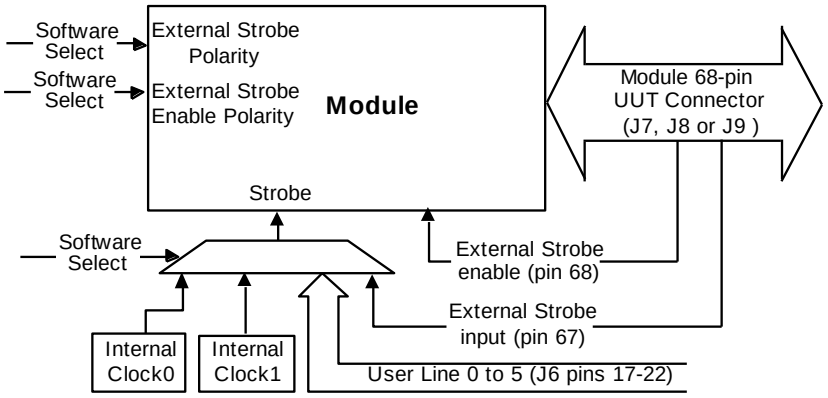

Figure 2-3: Module Strobe Signals

Internal Clock Source

The GX5731 contains two internal clock sources that can be used to drive any of the installed modules. Each of the internal clocks can be configured to use the 10MHz PXI clock, 33 MHz PCI Bus clock or any of External Input User Lines 2-5 (J6 pins 19-22).

Figure 2-4: GX5731 Internal Clock Source is a block diagram for the GX5731 Internal Clock Source:

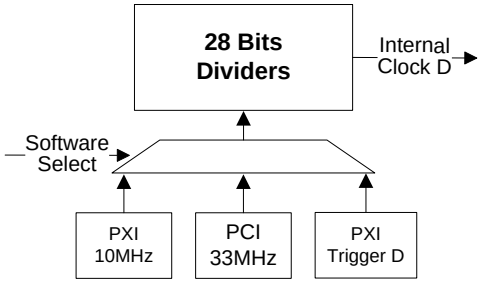

Figure 2-4: GX5731 Internal Clock Source

Module Strobed Output/Input Handshaking

The GX5731 can have up to three Modules installed. Each installed module can use Direct Output/Input or Strobed Output/Input with handshaking.

When using Direct Output/Input on each strobe signal transition (software controlled positive edge or negative edge)

When using Strobed Output/Input with handshaking the hardware controls the Output/Input sequences. The handshaking mechanism is implemented using input and output control and strobe signals.

GX5701: Digital Input Latch (DIL) Module

The GX5701 Digital Input Latch (DIL) Module has 32 input channels with 4K of high-speed buffer. The channels have programmable threshold of –30 to +30 volts with resolution of less then 1mV. The GX5701 can be triggered using internal or external strobe (software controlled). When using external strobe the module supports Strobed Input with Handshaking.

Features:

- Programmable threshold from –30 to +30 VDC.
- Direct input or Strobed Input with Handshaking
- Input trigger source may be internal, external or controlled by the user (step mode).
- High-speed buffer 4K deep.

Figure 2-5: demonstrates the Handshaking process. Each signal number (1 to 9) is explained in table. The description of each timing segment (numbered 1-9) explained in Table 2-1.

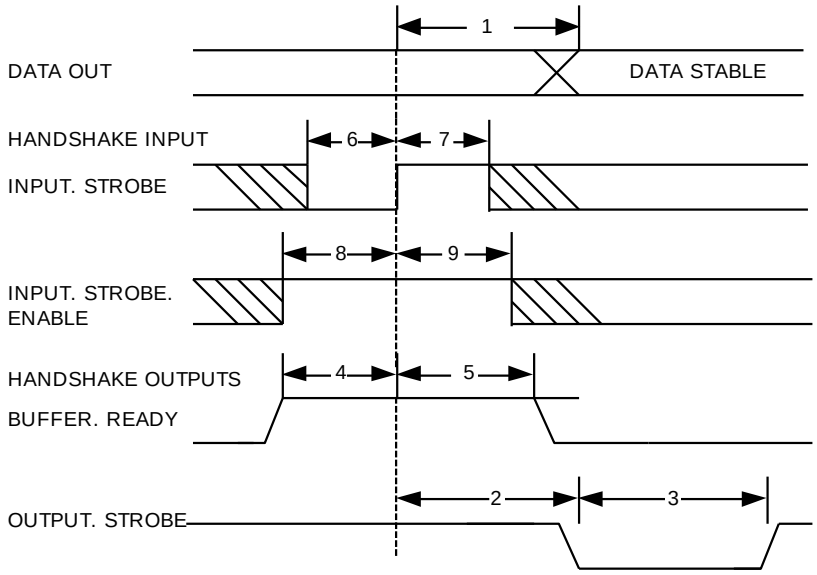

Figure 2-5: Strobed Input with Handshaking timing diagram

The following table describes each timing segment (numbered 1-9):

Signal #	Description	Time (uSec)		
		Min	Typ	Max
1	Data stable before strobe input is TRUE.	250		
2	Data stable after Strobe input is TRUE.	20		
3	Strobe input Pulse Width.	20		
4	Buffer Ready FALSE after Strobe TRUE (set to TRUE when issue a software RUN command).			15
5	Buffer ready TRUE after Strobe input is FALSE.	100		150
6	Strobe output delay after Strobe input is FALSE.	100		150
7	Strobe input pulse low Output strobe width (programmable with 50nS resolution).	50		750

Table 2-1: GX5701: Strobed Input with Handshaking timing diagram symbol description

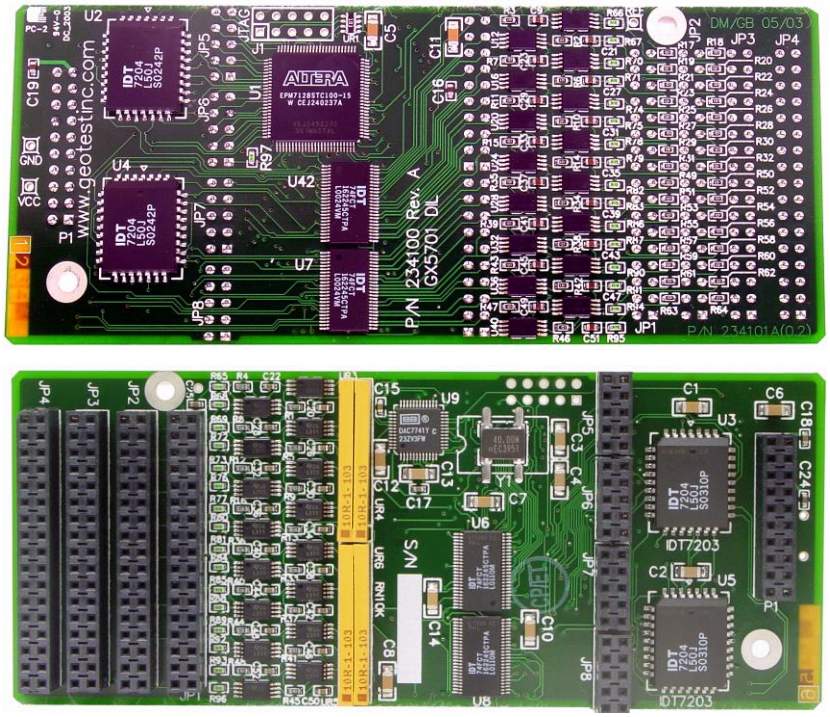

Figure 2-6: GX5701: Digital Input Latch (DIL) Module

GX5702: Digital Output Latch (DOL) Module

The GX5702 Module Digital Output Latch (DOL) drives 32 output-latched channels with 4K of high-speed buffer.. Data can be latched to the output connector using

The GX5702 can be triggered using internal or external strobe (software controlled). When using external strobe the module supports Strobed Input with Handshaking

Features:

- 32-channels of TTL or CMOS compatible outputs.
- Direct input or Strobed Input with Handshaking
- Output trigger source may be internal, external or controlled by the user (step mode).
- Programmable strobe delay.
- External Strobe input enable/disable by software.
- Up to 4096 (4K) can be stored in a High-speed output buffer.

Figure 2-7: demonstrates the Handshaking process. The description of each timing segment (numbered 1-9) explained in Table 2-2.

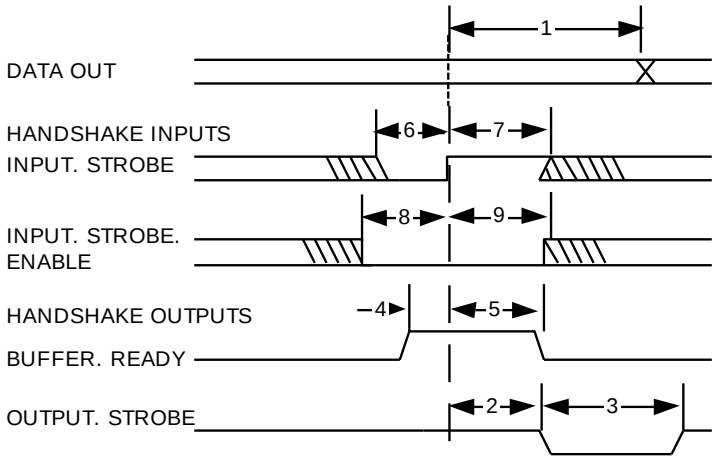

Figure 2-7: GX5702 Strobed Output with Handshaking timing diagram

The following table describes each timing segment (numbered 1-9):

Signal #	Description	Time (uSec)		
		Min	Typ	Max
2	Delay from Strobe input is TRUE to output valid.	25		
3	Strobe input Pulse Width.	20		
4	Buffer ready FALSE after Strobe TRUE (set to TRUE when issue a software RUN command).			15
5	Buffer ready TRUE after Strobe input is FALSE.	100		150
6	Strobe output delay after Strobe input is FALSE.	100		150
7	Strobe input pulse low Output strobe width (programmable with 50nS resolution)	50		750

Table 2-2: GX5702 Strobed Output with Handshaking timing diagram symbol description

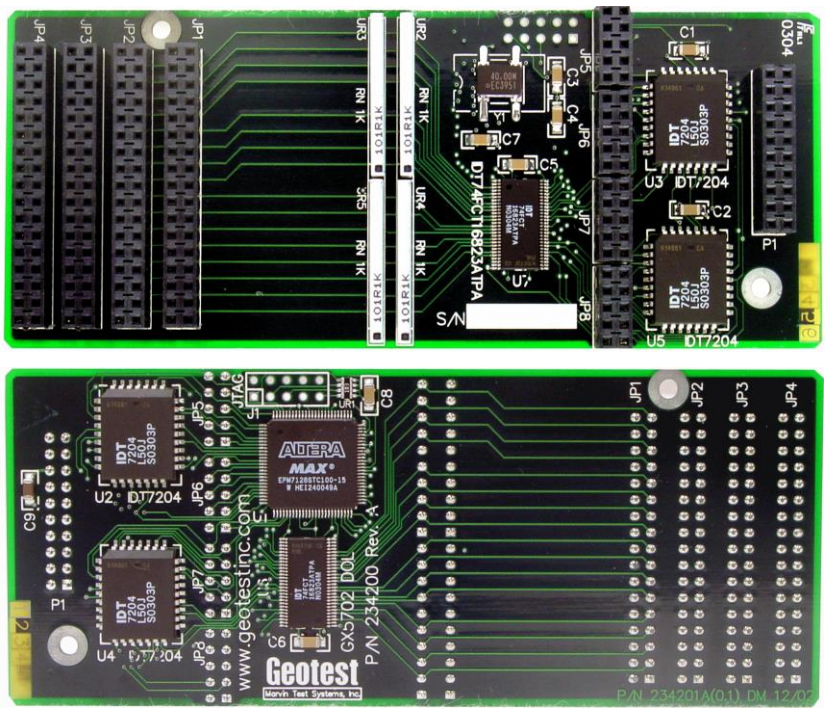

Figure 2-8: GX5702: Digital Output Latch (DOL) Module

GX5704: Digital Power Output (DPO) Module

The GX5704 Digital Power Output (DPO) Module has 32 output channels with 4K of high-speed buffer.

The GX5701 can be triggered using internal or external strobe (software controlled). When using external strobe the module supports Strobed Input with Handshaking

Features:

- 32 open collector channels outputs.
- Direct input or Strobed Input with Handshaking
- Output trigger source may be internal, external or controlled by the user (step mode).
- Programmable strobe delay.
- External Strobe input enable/disable by software.
- Up to 4096 (4K) can be stored in a High-speed output buffer.

Figure 2-9: demonstrates the Handshaking process. The description of each timing segment (numbered 1-9) explained in Table 2-3.

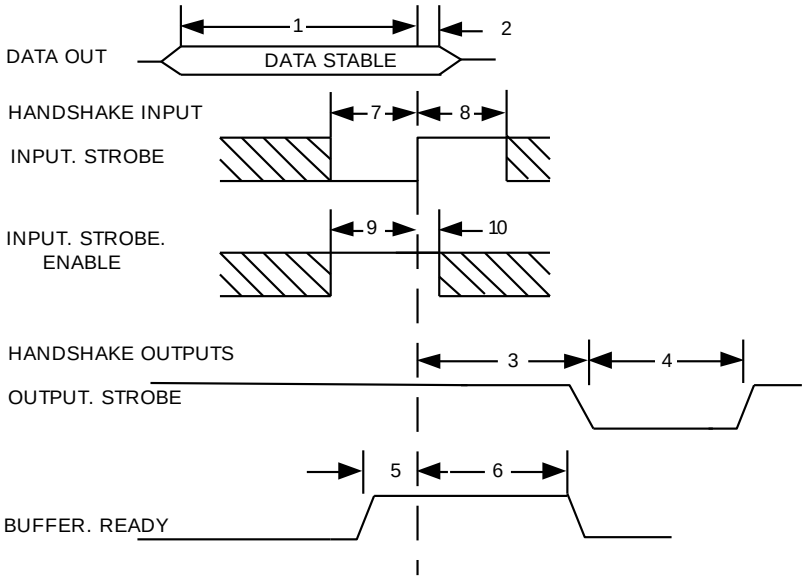

Figure 2-9: GX5704 Strobed Output with Handshaking timing diagram

The following table describes each timing segment (numbered 1-10):

Signal #	Description	Time (uSec)		
		Min	Type	Max
2	Delay from Strobe input is TRUE to output valid.	5000		
3	Strobe input Pulse Width.	20		
4	Buffer ready FALSE after Strobe TRUE (set to TRUE when issue a software RUN command).			15
5	Buffer ready TRUE after Strobe input is FALSE.	100		150
6	Strobe output delay after Strobe input is FALSE.	100		150
7	Strobe input pulse low Output strobe width (programmable with 50nS resolution)	50		750

Table 2-3: GX5704 Strobed Output with Handshaking timing diagram symbol description

Figure 2-10: GX5704- Digital Power Output (DPO) Module

GX5709: RS422 Port I/O

The GX5709 RS422 Port I/O Module is a differential RS422 32 I/O channels module. The direction and of each group of eight channels can be programmed as Input or Output. The of each. The termination of each group of eight channels can be programmed to be On or Off.

Features:

- 32 differential RS422 I/O channels Module.
- Programmable direction of channels in groups of eight.
- Programmable termination of channels in groups of eight.

Architecture

The RS422 I/O channels' direction is programmable and can be set for Input or Output in groups of eight. The GX5709 Module uses software driver to set or get the channels logic level and direction. Figure 2-11 shows a typical I/O channel block diagram:

Block Diagram

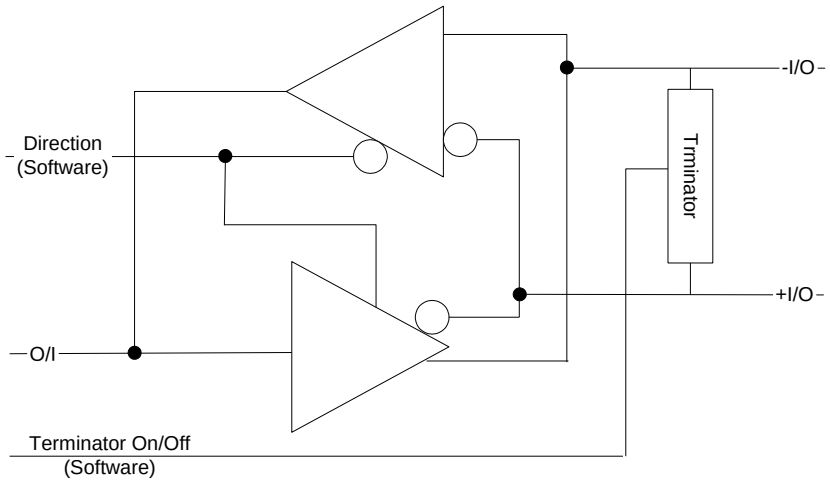

Figure 2-11: Typical RS422 I/O Channel Block Diagram

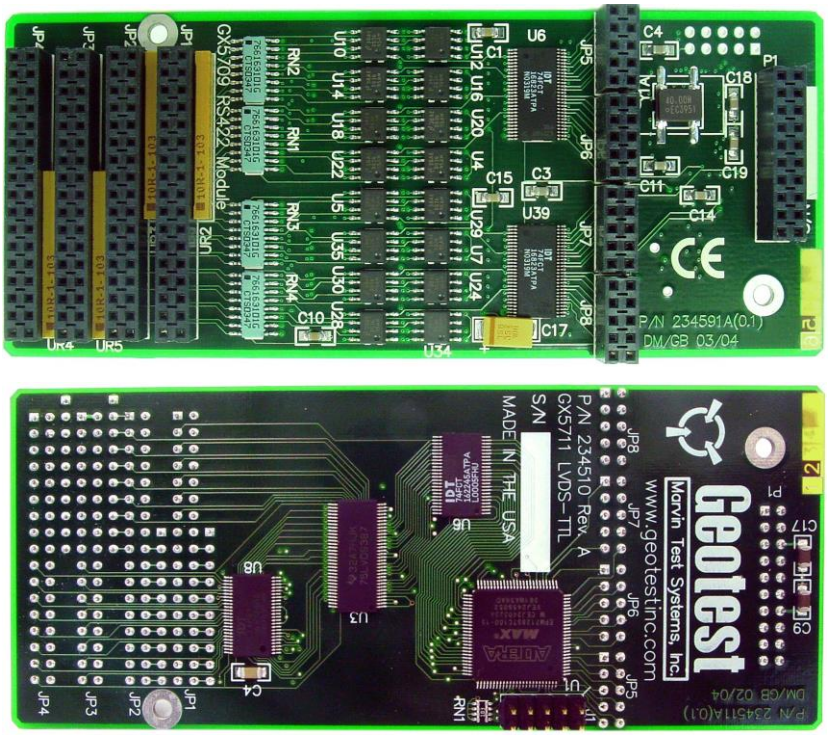

Figure 2-12: GX5709: RS422 Port I/O Module

GX5711: Bidirectional LVDS-TTL Converter Module

The GX5711 Bidirectional LVDS-TTL Converter Module is a 16 channels module converting signals from LVSD to TTL or vise versa. The direction of each group of eight channels can be programmed as LVDS to TTL or TTL to LVDS. Each group conversion direction source can be either internal (software) or external. All 16 differential channels can have termination can be programmed to be On or Off.

Features:

- 16 Bidirectional LVDS-TTL channels Module.
- Programmable direction of channels in groups of eight.
- Programmable conversion direction source as either internal (software) or external.
- Direction Programmable direction of channels in groups of eight.
- Programmable termination of channels.

Block Diagram

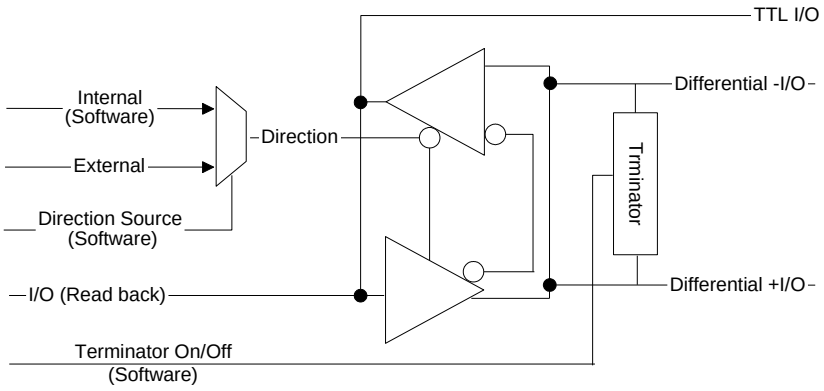

Figure 2-13: GX5711: Typical Bidirectional LVDS-TTL channel Block Diagram

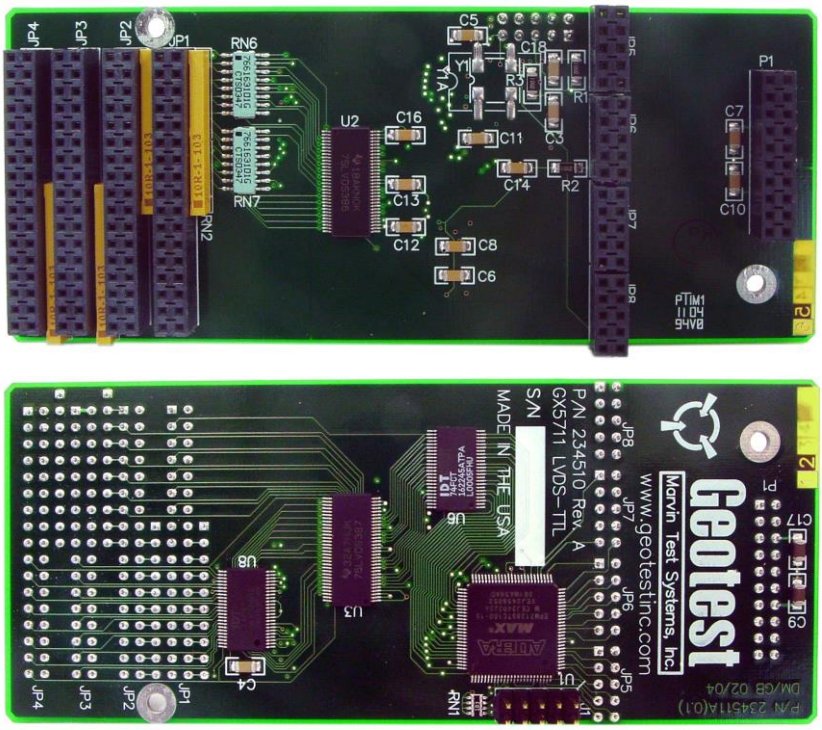

Figure 2-14: GX5711: Bidirectional LVDS-TTL Converter Module

GX5712: Bidirectional RS422-TTL Converter Module

The GX5712 Bidirectional RS422-TTL Converter Module is a 16 channels module converting signals from RS422 to TTL or vise versa. The direction of each group of eight channels can be programmed as RS422 to TTL or TTL to RS422. Each group conversion direction source can be either internal (software) or external. All 16 differential channels can have termination can be programmed to be On or Off.

Features:

- 16 Bidirectional RS422-TTL channels Module.
- Programmable direction of channels in groups of eight.
- Programmable conversion direction source as either internal (software) or external.
- Direction Programmable direction of channels in groups of eight.
- Programmable termination of channels.

Block Diagram

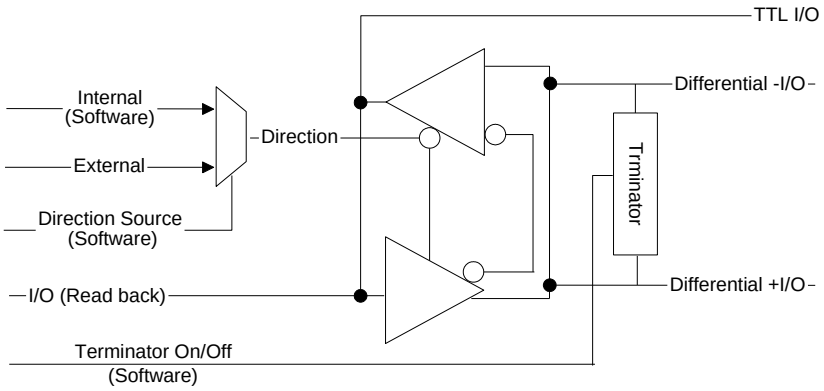

Figure 2-15: GX5712 - Typical Bidirectional RS422-TTL channel Block Diagram

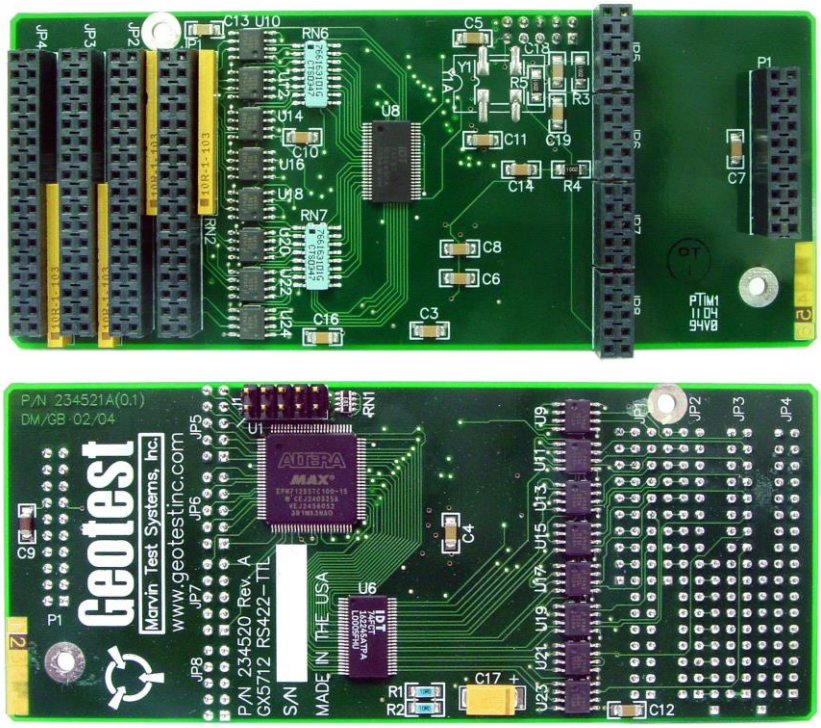

Figure 2-16: GX5712: Bidirectional RS422-TTL Converter Module

Specifications

The following table outlines the specifications of the GX5731.

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	0.1A typical, 0.3A max
5 VDC Current	1A typical, 4.5A max

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 GX5731 board and installation instructions for the GX5731 (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
GX5731 Driver Installation	This Chapter
Programming the boards	Chapter 5
GX5731 Reference Functions	Chapter 6

Packing List

All GX5731 boards have the same basic packing list, which includes:

1. GX5731 Board
2. GX5731 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 GX5731 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 you 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), 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 PXIeSYS.ini and is used by Marvin Test Solutions instruments HW driver and VISA. The applet can be used to assign chassis numbers, Legacy Slot numbers and instrument alias names. The HW driver is installed and shared with all Marvin Test Solutions products to support accessing the PC resources. Similar to HW driver, VISA provides a standard way for instrument manufacturers and users to write and use instruments drivers. VISA is a standard maintained by the VXI Plug & Play System Alliance and the PXI Systems Alliance organizations (http://www.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 GX5731 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 **Gx5731Initialize** and **Gx5731InitializeVisa** 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).

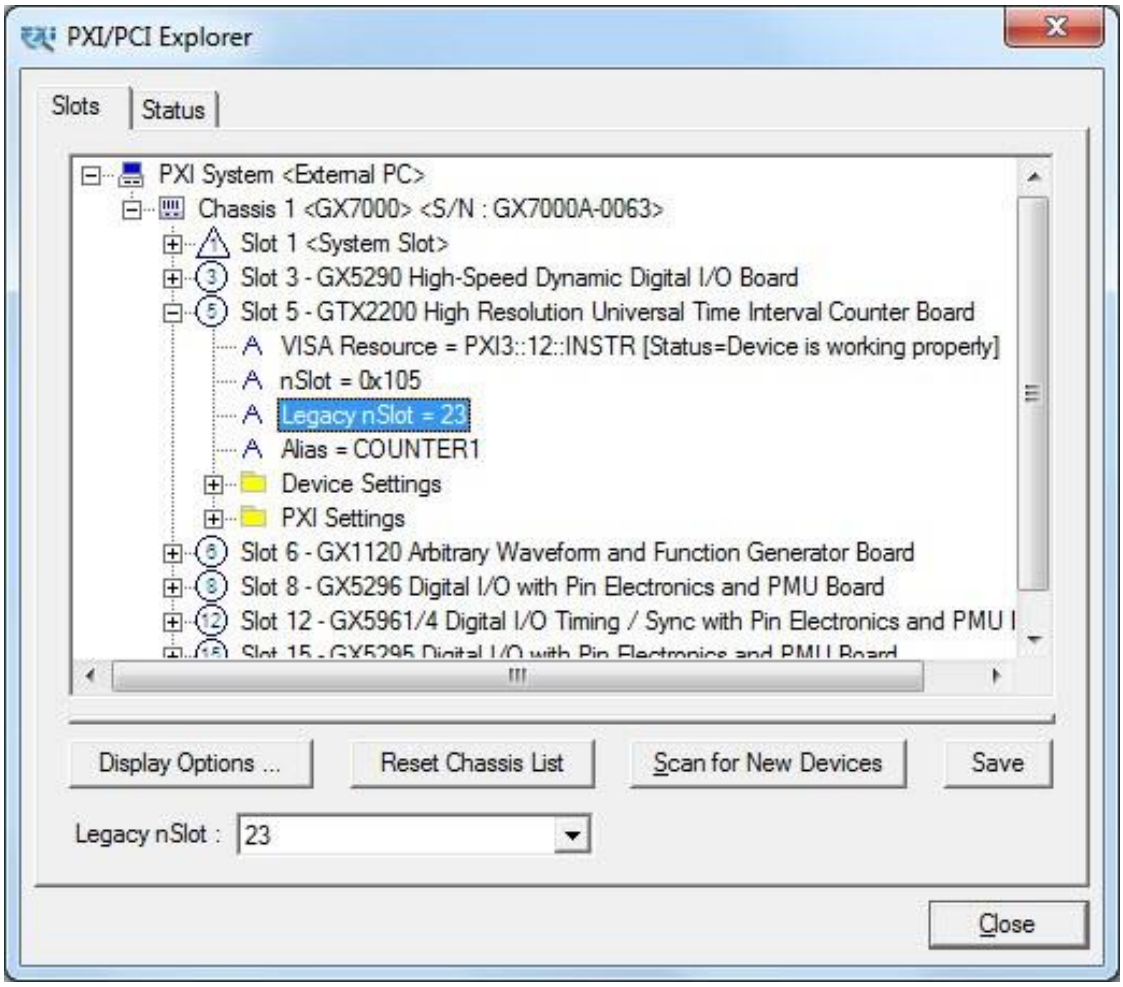

Figure 3-1: : PXI/PCI Explorer

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 GX5731 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).

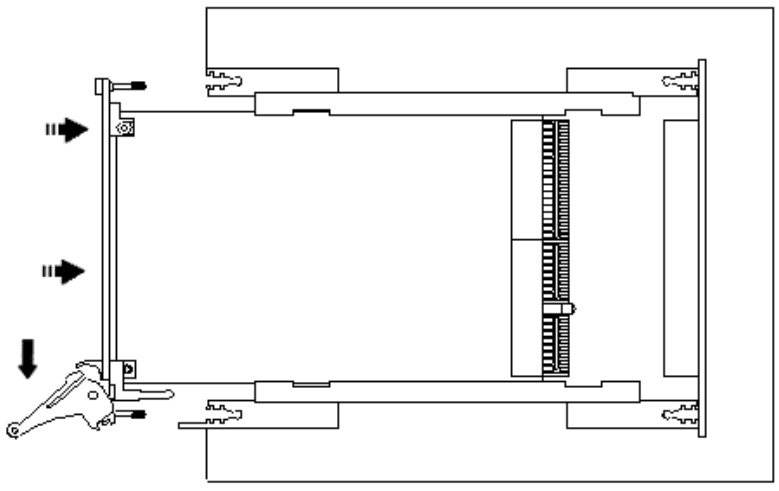

Figure 3-2: Ejector handles position during module insertion

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

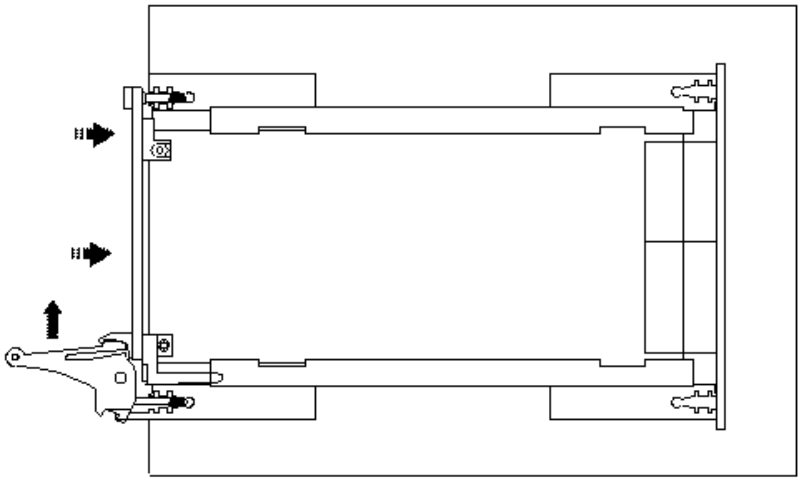

Figure 3-3: Ejector handles position after module insertion

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.
- GX5731.DRV - ATEasy driver File for GX5731.GXPIO Virtual Panel Program

GXPIO On-line Help and Manual

GXPIO.CHM – On-line version of the GX5731 User's Guide. The help file is provided in a Windows Compiled HTML help file (.CHM). The file contains information about the GX5731 board, programming reference and panel operation.

GX5731.PDF – On line, printable version of the GX5731 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
- GXPIOExample.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.
- GxPioExampleVBAssebleyInfo.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):

- Gx5731.prj - example project
- GX5731.sys - example system
- GX5731.prg - example program

Connectors and Jumpers

Figure 3-4 shows the available GX5731 board connectors and jumpers followed by their description:

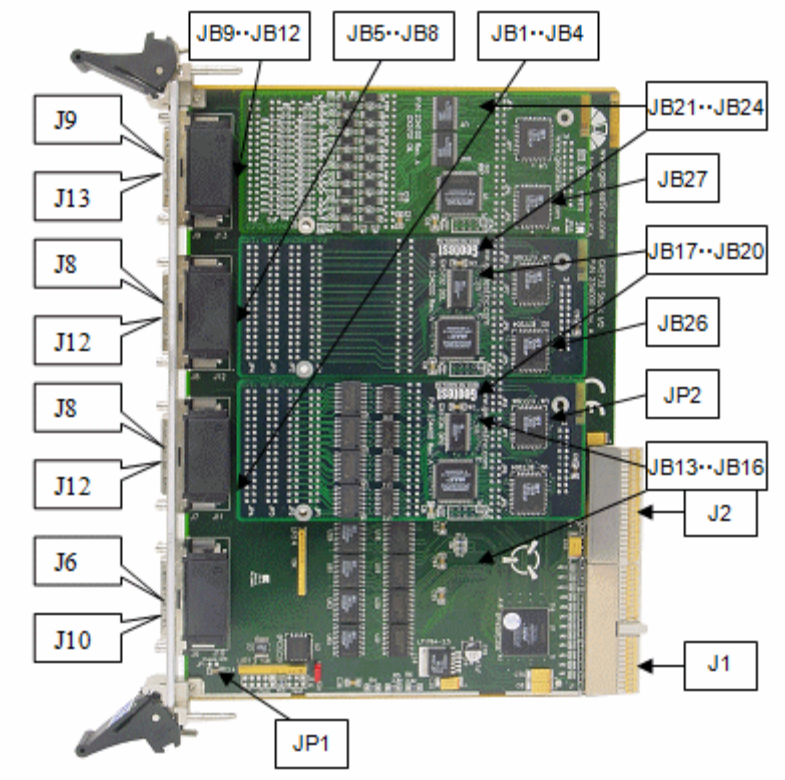

Figure 3-4: GX5731 Connectors and Jumpers

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
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 are divided into three groups, each group of Jumper Block pertain to a different I/O module:

- Jumper Block JP1-JP4: for I/O module number 0
- Jumper Block JP5-JP8: for I/O module number 1
- Jumper Block JP9-JP12: for I/O module number 2

In case that no I/O module is installed on any of the Jumper Blocks, those Jumper Blocks should be fully populated in order for them to function as a Digital Input Only ports.

Connections

Connections to the GX5731 may be made with eight 68 pins SCSI male plug connector. Shielded cables with matching connectors are available from Marvin Test Solutions.

J6 – External Strobe, External Clock, Internal Clock Out and PXI Trigger Connector

The following are the connector signal pin assignments for GX5731 module:

#	Signal	#	Signal	#	Signal	#	Signal
1	IC0	18	ClkIn1	35	GND	52	GND
2	IC1	19	ClkIn2	36	GND	53	GND
3	NU	20	ClkIn3	37	GND	54	GND
4	NU	21	ClkIn4	38	GND	55	GND
5	ASTrg	22	ClkIn5	39	GND	56	GND
6	BSTrg	23	NU	40	GND	57	GND
7	CSTrg	24	NU	41	GND	58	GND
8	NU	25	NC	42	GND	59	GND
9	PXTrg0	26	NC	43	GND	60	GND
10	PXTrg1	27	NC	44	GND	61	GND
11	PXTrg2	28	NC	45	GND	62	GND
12	PXTrg3	29	NC	46	GND	63	GND
13	PXTrg4	30	NC	47	GND	64	GND
14	PXTrg5	31	NC	48	GND	65	GND
15	PXTrg6	32	NC	49	GND	66	GND
16	PXTrg7	33	VCC	50	GND	67	VCC
17	ClkIn0	34	GND	51	GND	68	GND

Table 3-1: J6: External Strobe, External Clock, Internal Clock Out and PXI Trigger Connector

I	Input
O	Output
NC	Not Connected
NU	Not Used
GND	Ground
IC	Internal Clock
ASTrg, BSTrg, CSTrg	I/O Module Strobe Trigger
PXTrg0, PXTrg1, PXTrg2, XTrg3, PXTrg4, PXTrg5, PXTrg6, XTrg7,	PXI Trigger
ClkIn0, ClkIn1, ClkIn2, ClkIn3, ClkIn4, ClkIn5,	Clock Input

J7-J9: GX5731 with GX5701 Module Port 0-2 Connectors

The following are the connector signal pin assignments for GX5731 with GX5701 Module:

#	Signal	#	Signal	#	Signal	#	Signal
1	IN*n*0	18	IN*n*17	35	GND	52	GND
2	IN*n*1	19	IN*n*18	36	GND	53	GND
3	IN*n*2	20	IN*n*19	37	GND	54	GND
4	IN*n*3	21	IN*n*20	38	GND	55	GND
5	IN*n*4	22	IN*n*21	39	GND	56	GND
6	IN*n*5	23	IN*n*22	40	GND	57	GND
7	IN*n*6	24	IN*n*23	41	GND	58	GND
8	IN*n*7	25	IN*n*24	42	GND	59	GND
9	IN*n*8	26	IN*n*25	43	GND	60	GND
10	IN*n*9	27	IN*n*26	44	GND	61	GND
11	IN*n*10	28	IN*n*27	45	GND	62	GND
12	IN*n*11	29	IN*n*28	46	GND	63	GND
13	IN*n*12	30	IN*n*29	47	GND	64	GND
14	IN*n*13	31	IN*n*30	48	GND	65	GND
15	IN*n*14	32	IN*n*31	49	GND	66	GND
16	IN*n*15	33	BufRdy	50	GND	67	XStrb
17	IN*n*16	34	OutStrb	51	GND	68	XStrbEn

Table 3-2: J7-J9: GX5731 with GX5701 Module Port 0-2 Connectors

GND	Ground
IN*n*0 to IN*n*31	Input Data Lines, Port Number (*n*) can be 0-2
BufRdy	Buffer Ready Handshake line
OutStrb	Output Strobe Handshake line
XStrb	External Strobe input Handshake line
XStrbEn	External Strobe Enable Handshake line

J7-J9: GX5731 with GX5702 or GX5704 Module Port 0-2 Connectors

The following are the connector signal pin assignments for GX5731 with GX5702 or GX5704 Module:

#	Signal	#	Signal	#	Signal	#	Signal
1	OUT*n*0	18	OUT*n*17	35	GND	52	GND
2	OUT*n*1	19	OUT*n*18	36	GND	53	GND
3	OUT*n*2	20	OUT*n*19	37	GND	54	GND
4	OUT*n*3	21	OUT*n*20	38	GND	55	GND
5	OUT*n*4	22	OUT*n*21	39	GND	56	GND
6	OUT*n*5	23	OUT*n*22	40	GND	57	GND
7	OUT*n*6	24	OUT*n*23	41	GND	58	GND
8	OUT*n*7	25	OUT*n*24	42	GND	59	GND
9	OUT*n*8	26	OUT*n*25	43	GND	60	GND
10	OUT*n*9	27	OUT*n*26	44	GND	61	GND
11	OUT*n*10	28	OUT*n*27	45	GND	62	GND
12	OUT*n*11	29	OUT*n*28	46	GND	63	GND
13	OUT*n*12	30	OUT*n*29	47	GND	64	GND
14	OUT*n*13	31	OUT*n*30	48	GND	65	GND
15	OUT*n*14	32	OUT*n*31	49	GND	66	GND
16	OUT*n*15	33	BufRdy	50	GND	67	XStrb
17	OUT*n*16	34	OutStrb	51	GND	68	XStrbEn

Table 3-3: J7-J9: GX5731 with GX5702 or GX5704 Module Port 0-2 Connectors

GND	Ground
OUT*n*0 to OUT*n*31	Out Data Lines, Port Number (*n*) can be 0-2
BufRdy	Buffer Ready Handshake line
OutStrb	Output Strobe Handshake line
XStrb	External Strobe Input Handshake line
XStrbEn	External Strobe Enable Handshake line

J7-J9: GX5731 with GX5709 Module Port 0-2 Connectors

The following are the connector signal pin assignments for GX5731 with GX5709 Module:

#	Signal	#	Signal	#	Signal	#	Signal
1	I/O*n*D0+	18	I/O*n*D17+	35	I/O*n*D0-	52	I/O*n*D17-
2	I/O*n*D1+	19	I/O*n*D18+	36	I/O*n*D1-	53	I/O*n*D18-
3	I/O*n*D2+	20	I/O*n*D19+	37	I/O*n*D2-	54	I/O*n*D19-
4	I/O*n*D3+	21	I/O*n*D20+	38	I/O*n*D3-	55	I/O*n*D20-
5	I/O*n*D4+	22	I/O*n*D21+	39	I/O*n*D4-	56	I/O*n*D21-
6	I/O*n*D5+	23	I/O*n*D22+	40	I/O*n*D5-	57	I/O*n*D22-
7	I/O*n*D6+	24	I/O*n*D23+	41	I/O*n*D6-	58	I/O*n*D23-
8	I/O*n*D7+	25	I/O*n*D24+	42	I/O*n*D7-	59	I/O*n*D24-
9	I/O*n*D8+	26	I/O*n*D25+	43	I/O*n*D8-	60	I/O*n*D25-
10	I/O*n*D9+	27	I/O*n*D26+	44	I/O*n*D9-	61	I/O*n*D26-
11	I/O*n*D10+	28	I/O*n*D27+	45	I/O*n*D10-	62	I/O*n*D27-
12	I/O*n*D11+	29	I/O*n*D28+	46	I/O*n*D11-	63	I/O*n*D28-
13	I/O*n*D12+	30	I/O*n*D29+	47	I/O*n*D12-	64	I/O*n*D29-
14	I/O*n*D13+	31	I/O*n*D30+	48	I/O*n*D13-	65	I/O*n*D30-
15	I/O*n*D14+	32	I/O*n*D31+	49	I/O*n*D14-	66	I/O*n*D31-
16	I/O*n*D15+	33	VCC	50	I/O*n*D15-	67	VCC
17	I/O*n*D16+	34	GND	51	I/O*n*D16-	68	GND

Table 3-4: J7-J9: GX5731 with GX5709 Module Digital I/O Port 0-2 Connectors

I/O*n*D0 to I/O*n*D31	Diffrerential I/O Data Lines, Port number (*n)* can be 0 to 2
VCC	+5V
GND	Ground

J7-J9: GX5731 with GX5711 or GX5712 Module Port 0-2 Connectors

The following are the connector signal pin assignments for GX5731 with GX5711 or GX5712 Module:

#	Signal	#	Signal	#	Signal	#	Signal
1	I/*On*D0+	18	I/*On*T1	35	I/*On*D0-	52	GND
2	I/*On*D1+	19	I/*On*T2	36	I/*On*D1-	53	GND
3	I/*On*D2+	20	I/*On*T3	37	I/*On*D2-	54	GND
4	I/*On*D3+	21	I/*On*T4	38	I/*On*D3-	55	GND
5	I/*On*D4+	22	I/*On*T5	39	I/*On*D4-	56	GND
6	I/*On*D5+	23	I/*On*T6	40	I/*On*D5-	57	GND
7	I/*On*D6+	24	I/*On*T7	41	I/*On*D6-	58	GND
8	I/*On*D7+	25	I/*On*T8	42	I/*On*D7-	59	GND
9	I/*On*D8+	26	I/*On*T9	43	I/*On*D8-	60	GND
10	I/*On*D9+	27	I/*On*T10	44	I/*On*D9-	61	GND
11	I/*On*D10+	28	I/*On*T11	45	I/*On*D10-	62	GND
12	I/*On*D11+	29	I/*On*T12	46	I/*On*D11-	63	GND
13	I/*On*D12+	30	I/*On*T13	47	I/*On*D12-	64	GND
14	I/*On*D13+	31	I/*On*T14	48	I/*On*D13-	65	GND
15	I/*On*D14+	32	I/*On*T15	49	I/*On*D14-	66	GND
16	I/*On*D15+	33	DIR0	50	I/*On*D15-	67	VTH
17	I/*On*T0	34	DIR1	51	GND	68	GND

Table 3-5: J7-J9: GX5731 with GX5711 or GX5712 Module Ports 0-2 C*On*nector

I/O*n*D0 to I/O*n*D15	Differential I/O Data Lines, Port Number n (*n*) can be 0 to 2
I/O*n*T0 to I/O*n*T15	TTL I/O Data Lines, Port Number n can be 0 to 2
DIR0/ DIR1	Direction Control of data flow for group 0 and 1
VTH	High terminators voltage. (Usually VCC)
VCC	+5V
GND	Ground

J10-J13 – Ports 3-6 Digital I/O Connectors

The following are the connector signal pin assignments for GX5731 (Ports 3-6) Digital I/O module:

#	Signal	#	Signal	#	Signal	#	Signal
1	I/O*n*0	18	I/O*n*17	35	GND	52	GND
2	I/O*n*1	19	I/O*n*18	36	GND	53	GND
3	I/O*n*2	20	I/O*n*19	37	GND	54	GND
4	I/O*n*3	21	I/O*n*20	38	GND	55	GND
5	I/O*n*4	22	I/O*n*21	39	GND	56	GND
6	I/O*n*5	23	I/O*n*22	40	GND	57	GND
7	I/O*n*6	24	I/O*n*23	41	GND	58	GND
8	I/O*n*7	25	I/O*n*24	42	GND	59	GND
9	I/O*n*8	26	I/O*n*25	43	GND	60	GND
10	I/O*n*9	27	I/O*n*26	44	GND	61	GND
11	I/O*n*10	28	I/O*n*27	45	GND	62	GND
12	I/O*n*11	29	I/O*n*28	46	GND	63	GND
13	I/O*n*12	30	I/O*n*29	47	GND	64	GND
14	I/O*n*13	31	I/O*n*30	48	GND	65	GND
15	I/O*n*14	32	I/O*n*31	49	GND	66	GND
16	I/O*n*15	33	VCC	50	GND	67	VCC
17	I/O*n*16	34	GND	51	GND	68	GND

Table 3-6: GX5731: Ports 3-6 Digital I/O Connectors

I/O*n*0 to I/O*n*31	Bi-directional I/O, Port number (*n*) can be 3 to 6, channels data lines (0-31)
VCC	+5V
GND	Ground

Connectors and Accessories

The following accessories are available from Marvin Test Solutions for GX5731 switching board.

Part / Model Number	Description
GX95401	Extra GX5731 User Manual
GT95014	Connector Interface for GX5xxx/GX5731, SCSI to 100 Mil Grid, Single Ended
GT95021	2' shielded cable for GX5xxx/GX5731 (68 Pin)
GT95022	3' shielded cable for GX5xxx/GX5731 (68 Pin)
GT95028	10' shielded cable for GX5xxx/GX5731 (68 Pin)
GT95031	6' shielded cable for GX5xxx/GX5731 (68 Pin)

Chapter 4 - Instrument Software Panel

Calling the **GX5731Panel** 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 **Gx5731Panel** function is also used by the GXPIOPANEL.EXE or GXPIOPANEL64.EXE (with the **/5731** 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 **GX5731** 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 **GX5731 Panel** from the **Marvin Test Solutions**, **GXPIO** menu under the **Start** menu. The GX5731 virtual panel opens as shown here:

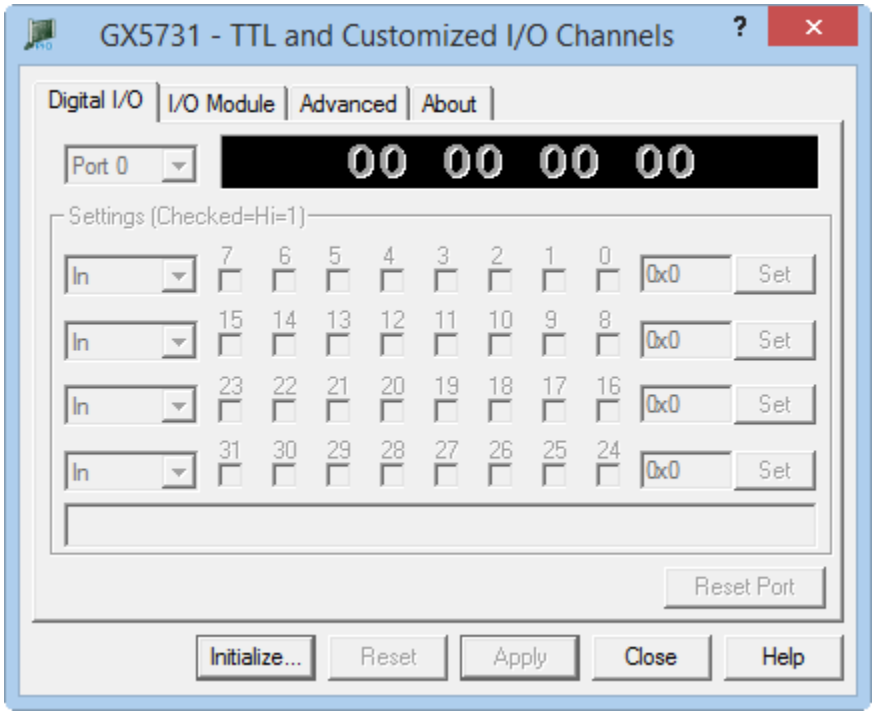

Figure 4-1: GX5731 Virtual Panel

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 **GX5731** 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 **Gx5731Initialize** function which is used to initialize the driver for the specified board.

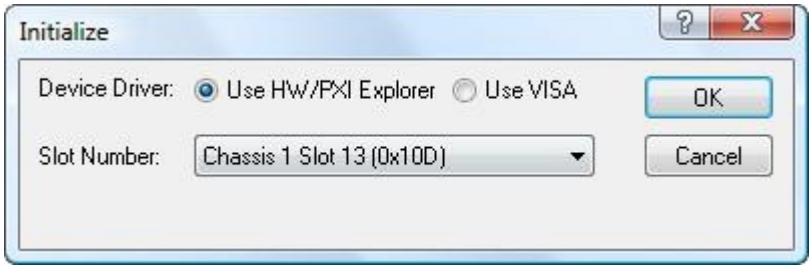

Figure 4-2: Initialize Dialog Box using Marvin Test Solutions' HW driver

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 **Gx5731InitializeVisa** function which initializes the driver for the specified board.

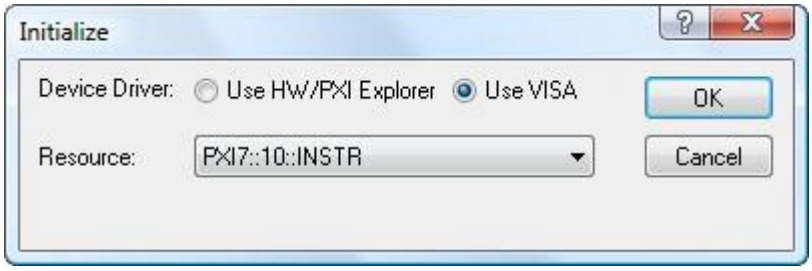

Figure 4-3: Initialize Dialog Box using VISA resources

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:

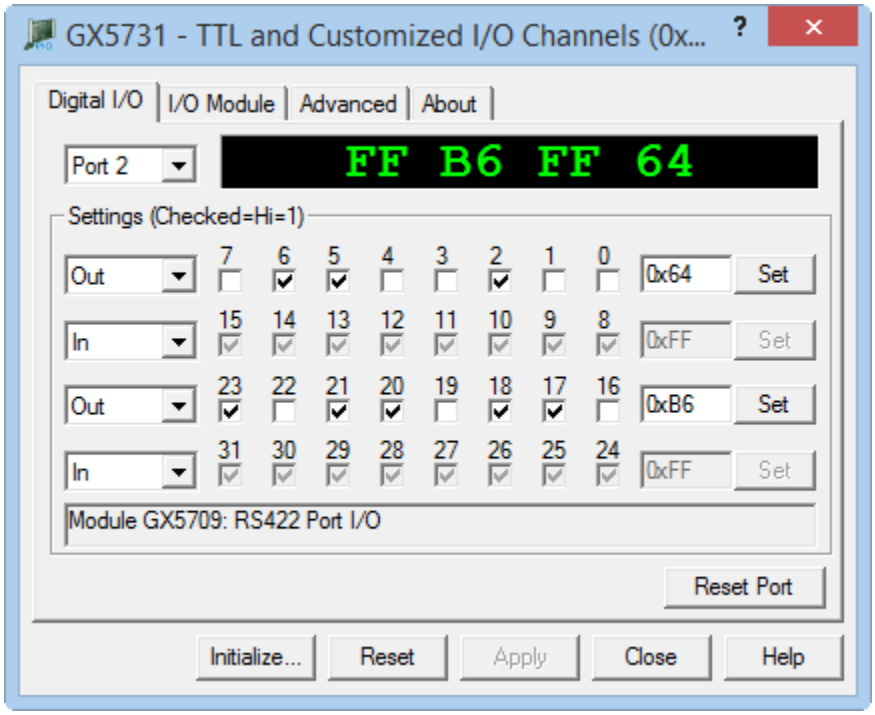

Figure 4-4: GX5731 Virtual Panel – Digital I/O page

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 Byte 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 for24-31.

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 Advanced Page

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

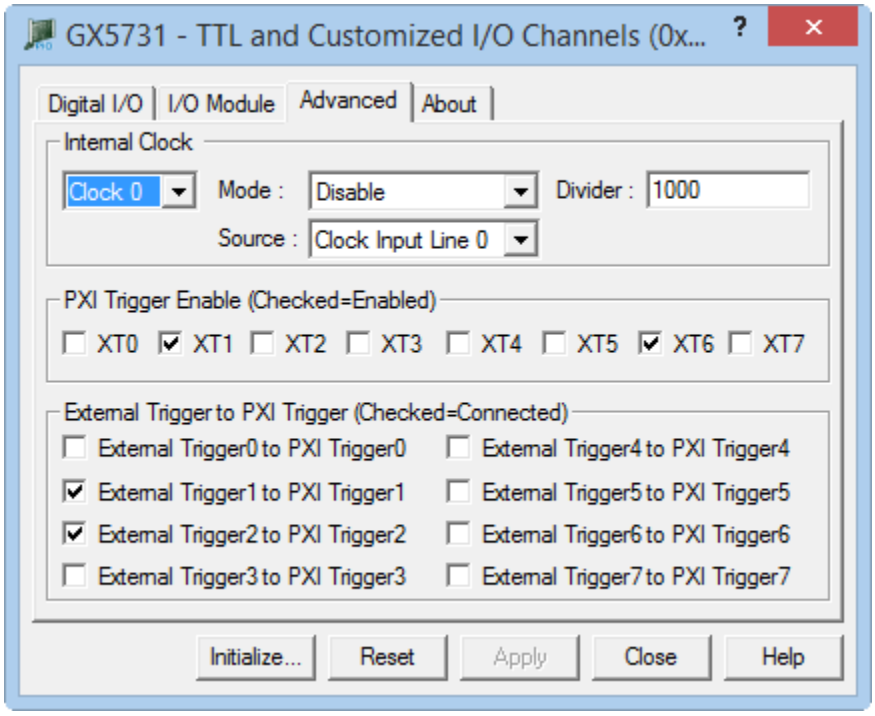

Figure 4-5: GX5731 Virtual Panel – Advanced Page

The following controls are shown in the Advanced page:

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

Internal Clock Mode drop down box: Sets/display the Internal Clock Mode, Enable or Disable. The Clock will not be start until enabled.

Internal Clock Divider edit box: Sets/display the selected clock divider. Type a number between 1 to 0x1000000 (16,777,216). This number will be used to divide the Clock the input from the clock source.

Internal Clock Source drop down box: Sets/display **t**he source that is used as an input to the selected internal clock.

PXI Trigger Enable: Enable (checked) or disable (unchecked) the specified PXI Trigger.

External Trigger to PXI Trigger: Connect (checked) or disconnect (unchecked) the specified External Trigger line to PXI Trigger line.

Virtual Panel About Page

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

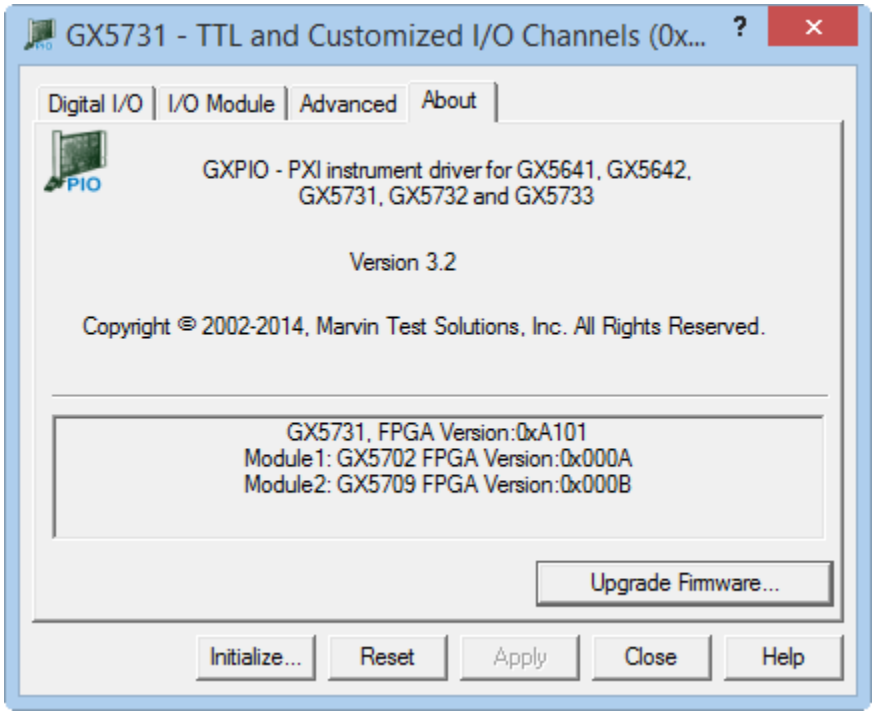

Figure 4-6: GX5731 Virtual Panel – About Page

The top part of the **About** page displays version and copyright of the GX1838 driver. The bottom part displays the board summary, including the main board FPGA version an each installed I/O Module FPGA version. The **About** page also contains a button **Upgrade Firmware…** used to upgrade the board FPGA. This button maybe used only when the board requires upgrade as directed by Marvin Test Solutions support. The upgrade requires a firmware file (.jam) that is written to the board FPGA. After the upgrade is complete you must shut down the computer to recycle power to the board.

Chapter 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 a 32-bit Windows DLL file GXPIO.DLL and a 64-bit DLL GXPIO64.DLL. The DLL is used with 32-bit or a 64-bit applications running under Windows. 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 GX5731 driver with C/C++ development tools:

- Include the GXPIO.H header file in the C/C++ source file that uses the GX5731 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 GX5731 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 GX5731 unit in the **uses** statement before making calls to the GX5731 functions.

Programming GXPIO Boards Using ATEasy®

The GXPIO package is supplied with a separate ATEasy driver for each of board types. The **GX5731.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 GX5731.
- 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 GX5731 board uses the Gx5731Xxxx 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 **Gx5731Initialize** to initialize the board's driver using HW and **Gx5731InitializeVisa** 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 **Gx5731Initialize**(*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.

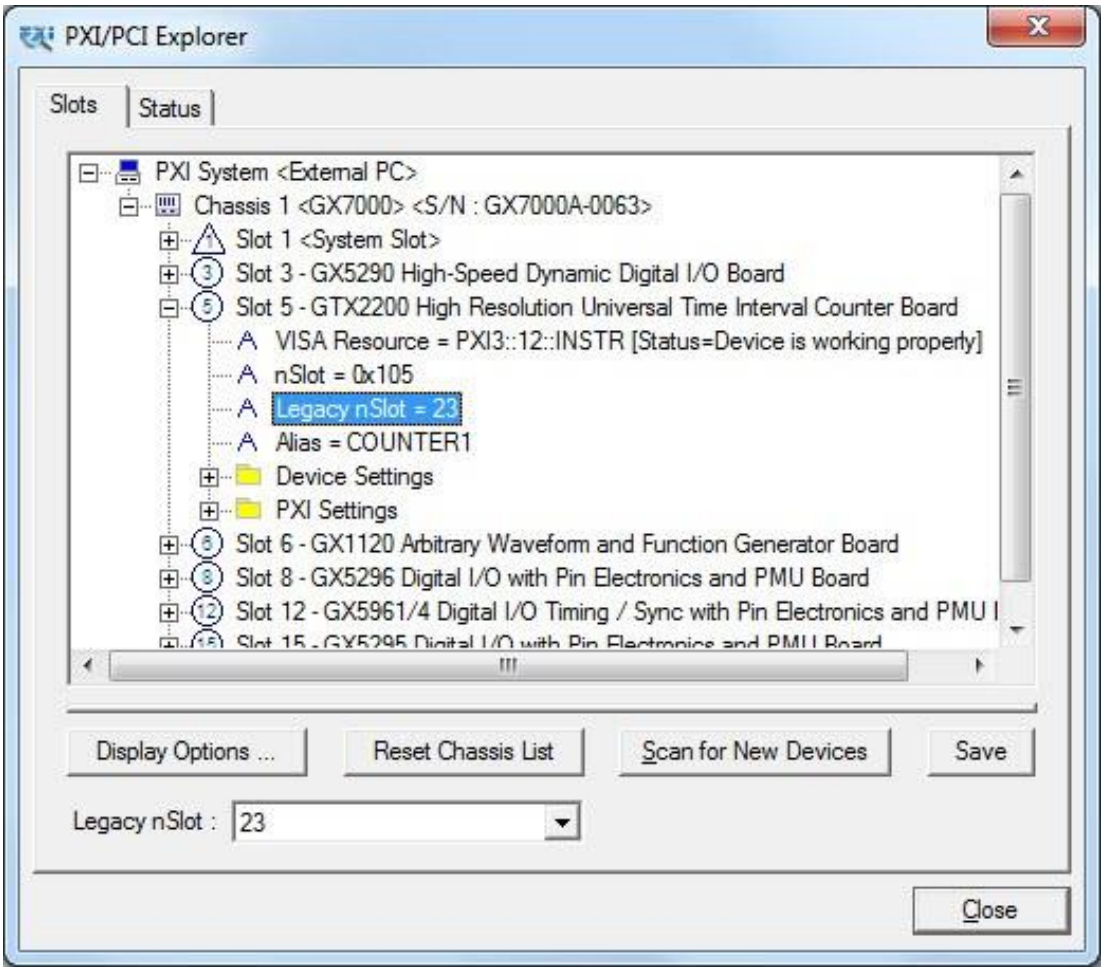

Figure 5-1: PXI/PCI Explorer

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 **Gx5731InitializeVisa** (*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 **Gx5731Initialize** and the **Gx5731InitializeVisa** 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 **Gx5731Panel** 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 **Gx5731Panel** 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 - MTS 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
#if defined(__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>
        <direction|value|count>\r\n\n"
        "Where: \r\n"
        "<board_type> - Board type: 5731=GX5731, 5732=GX5732\r\n\n"
```

```
        "<slot_number> - PCI/PXI slot number as shown by the PXI explorer\r\n\n"
        "<port|counter numbers> - depends on the operation:\r\n"
        "\tport number:    \t0-6 for WD/RD/WP/RP operations\r\n"
        "\tcounter number: \t0-7 for WC/RC operations\r\n\n"
        "<operation> - one of the followings :\r\n"
        "\tWD=Write port direction\t\tRD=Read port direction\r\n"
        "\tWP=Write port value    \t\tRP=Read port value\r\n"
        "\tWC=Write counter       \t\tRC=Read counter\r\n"
        "\tRB=read buffer              \tWB=write buffer\r\n"
        "\tRBF=read buffer empty/full flag\r\n"
        "\tSCLK_SRC=set internal clock source GCLK_SRC=get internal clock source\r\n"
        "\tSCLK_DIV=set internal clock divider       GCLK_DIV=Get internal clock divider\r\n\n"

        "<direction|value|counter> - depends on the operation:\r\n"
        "\tWD : 0-15        \t\t4 bits direction. each bit controls a byte direction\r\n"
        "\t             \t\t in the port, 0 for input and 1 for output\r\n"
        "\tWP : 0-0xFFFFFFFF\t\tPort value\r\n"
        "\tWC : 0-255       \t\tCounter value to load\r\n\n"

        "To change command line under Windows:\r\n"
        "\tRight click on the example shortcut from the start menu\r\n"
        "\tand type the new command line"
        );
    exit(1);
}
//*********************************************************
//      CheckStatus
//*********************************************************
void CheckStatus(SHORT nStatus)
{
    CHAR   sz[128];

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

//*********************************************************
//      MAIN
//
// This main functin receives three to four parameters
//
// Board type: 5731 - GX5731
//             5732 - GX5732
// PIO board slot number (e.g. 1)
// Port/counter number (depends on operation):
//         0 - 6 for port
//         0 - 7 for counter
// Operation:
//         RD=read port direction
//         WD=write port direction
```

```
//          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 = strupr(*(++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 GX5731 driver functions reference chapter is organized in alphabetical order. Each function is presented starting with the syntax of the function, a short description of the function parameters description and type followed by a Comments, an Example (written in C), and a See Also sections.

All function parameters follow 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 required for operating the board and is returned by the **Gx5731Initialize** or the f **Gx5731InitializeVisa** 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.PAS or GX5731.DRV.
- Parameter name are prefixed as follows:

Prefix	Type	Example
a	Array, prefix this before the simple type.	*anArray* (Array of Short)
n	SHORT (signed 16-bit)	*nMode*
d	DOUBLE - 8 bytes floating point	*dReading*
dw	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

GX5731 Functions

The following list is a summary of functions available for the GX5731:

Driver Functions	Description
General	
Gx5731GetBoardSummary	Returns the board summary.
Gx5731Initialize	Initializes the driver for the board at the specified slot number using HW. The function returns a handle that can be used with other GX5731 functions to program the board
Gx5731InitializeVisa	Initializes the driver for the specified slot using VISA. The function returns a handle that can be used with other GX5731 functions to program the board.
Gx5731Panel	Opens a virtual panel used to interactively control the GX5731.
Gx5731Reset	Resets the GX5731 board to its default settings.
GxPioGetDriverSummary	Returns the driver name and version.
GxPioGetErrorString	Returns the error string associated with the specified error number.
Digital I/O Functions	
Gx5731GetPort	Reads the port value.
Gx5731GetPortBit	Reads a port bit value.
Gx5731GetPortByte	Reads a port byte value.
Gx5731GetPortByteDirection	Returns the direction of a port byte.
Gx5731GetPortDirection	Returns the direction of port bytes.
Gx5731GetPortWord	Reads a port word value.
Gx5731ResetPort	Reset the specified digital I/O port.
GX5731SetPort	Writes a value to a port.
Gx5731SetPortBit	Writes a value to a port byte.
Gx5731SetPortByte	Writes a value to the port byte.
Gx5731SetPortByteDirection	Sets the direction of a port byte.
Gx5731SetPortDirection	Sets the direction for port bytes.
Gx5731SetPortWord	Writes a value to the port word.
Module Functions	
Gx5731ModuleBufferClear	Clear the Module's Buffer.
Gx5731ModuleBufferGetMode	Returns the Module's Buffer Full or Half-Full Mode.
Gx5731ModuleBufferGetState	Returns the Module's Buffer Full or Half-Full State.
Gx5731ModuleBufferReadData	Returns the Module's buffer.
Gx5731ModuleBufferSetMode	Set the Module's Buffer Full or Half-Full Mode.

Driver Functions	Description
Gx5731ModuleBufferTest	Test the specified Module buffer fro reading and writing
Gx5731ModuleBufferWriteData	Write a buffer to the Module's Buffer.
Gx5731ModuleGetDirection	Returns the Module's channels direction
Gx5731ModuleGetDirectionSource	Returns the Module's channels direction source
Gx5731ModuleGetExtStrobeEnablePolarity	Returns the Module's External Strobe Enable Polarity.
Gx5731ModuleGetExtStrobeEnableState	Returns the Module's External Strobe Enable State.
Gx5731ModuleGetExtStrobePolarity	Returns the Module's External Strobe Polarity.
Gx5731ModuleGetExtStrobeState	Returns the Module's External Strobe State.
Gx5731ModuleGetHandshakeOutputMode	Returns the Module's Handshake output mode.
Gx5731ModuleGetInternalStrobeSource	Returns the Module's Internal Strobe Clock source.
Gx5731ModuleGetOperationMode	Returns the Module's Operation mode.
Gx5731ModuleGetOutputMode	Returns the Module's Output mode.
Gx5731ModuleGetOutputStrobeWidth	Returns the Module's Output Strobe Width.
Gx5731ModuleGetPort	Returns the Module's Port Data.
Gx5731ModuleGetPxiTriggerBus	Returns all specified Module Strobe to Pxi Trigger lines connection mode.
Gx5731ModuleGetPxiTriggerLine	Returns the specified Module's Strobe to Pxi Trigger Line connection mode.
Gx5731ModuleGetRevision	Returns the Module's Revision.
Gx5731ModuleGetState	Returns the Module's State.
Gx5731ModuleGetStrobeSource	Returns the Module's Strobe Source.
Gx5731ModuleGetTermination	Returns the Module termination mode
Gx5731ModuleGetType	Returns the Module's type.
Gx5731ModuleGetVThreshold	Returns the Module's Threshold voltage.
Gx5731ModuleHalt	Sets the specified Module to Halt state.
Gx5731ModuleRun	Sets the specified Module to Run mode.
Gx5731ModuleSetDirection	Sets the Module's channels direction
Gx5731ModuleSetDirectionSource	Sets the Module's channels direction source
Gx5731ModuleSetExtStrobeEnablePolarity	Sets the Module's External Strobe Enable Polarity.
Gx5731ModuleSetExtStrobePolarity	Sets the Module's External Strobe Polarity.
Gx5731ModuleSetHandshakeOutputMode	Sets the Module's Handshake output mode.
Gx5731ModuleSetInternalStrobeSource	Sets the Module's Internal Strobe Clock source.
Gx5731ModuleSetOperationMode	Sets the Module's Operation mode.
Gx5731ModuleSetOutputMode	Sets the Module's Output mode.
Gx5731ModuleSetOutputStrobeWidth	Sets the Module's Output Strobe Width.

Driver Functions	Description
Gx5731ModuleSetPort	Sets the Module's Port Data.
Gx5731ModuleSetPxiTriggerBus	Sets all specified Module Strobe to Pxi Trigger lines connection mode.
Gx5731ModuleSetPxiTriggerLine	Sets the specified Module's Strobe to Pxi Trigger Line connection mode.
Gx5731ModuleSetStrobeSource	Sets the Module's Strobe Source.
Gx5731ModuleSetVThreshold	Sets the Module's Threshold voltage.
Gx5731ModuleStep	Run the specified port Module for given number of steps.
Gx5731ModuleSetTermination	Sets the Module termination mode
Clock Functions	
Gx5731GetInternalClock	Returns the internal clock source and divider.
Gx5731GetInternalClockEnable	Returns the specified internal clock enable mode
Gx5731SetInternalClock	Sets internal clock source and divider.
Gx5731SetInternalClockEnable	Sets the specified internal clock enable mode
Pxi Trigger Functions	
Gx5731GetExtTriggerToPxiTriggerBus	Returns the connections for all External Trigger lines to their corresponding Pxi Triggers lines number.
Gx5731GetExtTriggerToPxiTriggerLine	Returns the connection for the specified External Trigger to Pxi Trigger line
Gx5731GetStrobeToPxiTriggerBusState	Returns all Strobe to Pxi Trigger lines Output Mode.
Gx5731GetStrobeToPxiTriggerLineState	Returns the specified Strobe to Pxi Trigger Line output mode..
Gx5731SetExtTriggerToPxiTriggerBus	Sets the connections for all External Trigger lines to their corresponding Pxi Triggers lines number.
Gx5731SetExtTriggerToPxiTriggerLine	Sets the connection for the specified External Trigger to Pxi Trigger line
Gx5731SetStrobeToPxiTriggerBusState	Sets all Strobe to Pxi Trigger lines Output Mode
Gx5731SetStrobeToPxiTriggerLineState	Sets the specified Strobe to Pxi Trigger Line output mode

Gx5731GetBoardSummary

Purpose

Returns the board summary.

Syntax

Gx5731GetBoardSummary(*nHandle, szSummary, nSumMaxLen, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
szSummary	PSTR	Buffer to contain the returned board info (null terminated) string.
nSumMaxLen	SHORT	Size of the buffer to contain the error string.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX5731 summary string provides the following data:

- Instrument name (e.g., GX5731)
- FPGA version of main board (e.g. 0xA003)
- Each installed Module's FPGA version.

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

```
GX5731, FPGA Version:0xA000, Module0: GX5701 FPGA Version:0x000A, Module1: GX5702 FPGA
Version:0x000A, Module2: GX5704 FPGA Version:0x000A
```

Example

The following example returns the board summary:

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

Gx5731GetBoardSummary(nHandle, szSummary, sizeof(szSummary), &nStatus)
```

See Also

GxPioGetDriverSummary, Gx5731Initialize, GxPioGetErrorString

Gx5731GetExtTriggerToPxiTriggerBus

Purpose

Returns the connections for all External Trigger lines to their corresponding Pxi Triggers lines number.

Syntax

Gx5731GetExtTriggerToPxiTriggerBus (*nHandle, pnExtTrigToXTrig, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
pnExtTrigToXTrig	PSHORT	Returns the connections for all External Trigger lines to their corresponding Pxi Triggers lines number. Each of bits 0-7 represents a External Trigger lines, bit 0 for line 0 and bit 7 for line 7.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

External Trigger lines can only be connected to their corresponding Pxi Triggers lines number. I.e. External Trigger line 0 can only be connected to Pxi Triggers line 0 and so forth.

Use the **Gx5731SetExtTriggerToPxiTriggerLine**to set a specified External Trigger line to Pxi Trigger Line.

Example

The following example returns the connection of all External Trigger lines to their corresponding Pxi Triggers lines Returns:

```
SHORT nExtTrigToXTrig, nStatus;

Gx5731SetExtTriggerToPxiTriggerBus (nHandle, &nExtTrigToXTrig, &nStatus);
```

See Also

Gx5731SetExtTriggerToPxiTriggerBus, **Gx5731SetExtTriggerToPxiTriggerLine**, **Gx5731GetExtTriggerToPxiTriggerLine, Gx5731SetStrobeToPxiTriggerBusState**

Gx5731GetExtTriggerToPxiTriggerLine

Purpose

Returns the connection for the specified External Trigger to Pxi Trigger line.

Syntax

Gx5731GetExtTriggerToPxiTriggerLine (*nHandle, nExtTrigToXTrigLine, pbMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nExtTrigToXTrigLine	SHORT	External Trigger to Pxi Trigger Line numbers are: 0 GX5731_EXT_TRIGGER_LINE0: External Trigger 0. 1 GX5731_EXT_TRIGGER_LINE1: External Trigger 1. 2 GX5731_EXT_TRIGGER_LINE2: External Trigger 2. 3 GX5731_EXT_TRIGGER_LINE3: External Trigger 3. 4 GX5731_EXT_TRIGGER_LINE4: External Trigger 4. 5 GX5731_EXT_TRIGGER_LINE5: External Trigger 5. 6 GX5731_EXT_TRIGGER_LINE6: External Trigger 6. 7 GX5731_EXT_TRIGGER_LINE7: External Trigger 7.
bMode	BOOL	Pxi trigger line mode are: 0 GX5731_XTRIGGER_LINE_DISCONNECTED: Pxi trigger line is disconnected. 1 GX5731_XTRIGGER_LINE_CONNECTED: Pxi trigger line is Connected
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetExtTriggerToPxiTriggerBus** to set External Trigger lines can only be connected to their corresponding Pxi Triggers lines number.

Example

The following example returns External Trigger lines 2 to Pxi Triggers line 2 connection:

```
SHORT nStatus;
BOOL bMode;
Gx5731GetExtTriggerToPxiTriggerLine (nHandle, GX5731_EXT_TRIGGER_LINE2, &bMode, &nStatus);
```

See Also

Gx5731SetExtTriggerToPxiTriggerLine, **Gx5731SetExtTriggerToPxiTriggerBus**, **Gx5731GetExtTriggerToPxiTriggerLine, Gx5731SetStrobeToPxiTriggerBusState**

Gx5731GetInternalClock

Purpose

Returns the internal clock source and divider.

Syntax

Gx5731GetInternalClock (*nHandle, nClock, pnSource, pdwDivider, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nClock	SHORT	Clock number: 0 GX5731_CLOCK0: Internal Clock 0 1 GX5731_CLOCK1: Internal Clock 1
nSource	SHORT	Internal Clock 0 and Internal Clock 1 sources are listed in comment.
dwDivider	DWORD	Divider value: 1-0x8000000
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Internal Clock 0 sources are:

0 GX5731_INTERNAL_CLOCK0_T0_CLOCK_IN0: Internal Clock 0 connected to Clock Input 1.

1 GX5731_INTERNAL_CLOCK0_T0_PCI_CLOCK_33MHZ: Internal Clock 0 connected to 33MHz PCI Clock.

2 GX5731_INTERNAL_CLOCK0_T0_PXI_CLOCK_10MHZ: Internal Clock 0 connected to 10MHz PXI Clock.

3 GX5731_INTERNAL_CLOCK0_T0_STAR_TRIGGER: Internal Clock 0 connected to the PXI Start Trigger.

4 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER0: Internal Clock 0 connected to the PXI External Trigger Line 0.

5 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER1: Internal Clock 0 connected to the PXI External Trigger Line 1.

6 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER2 Internal Clock 0 connected to the PXI External Trigger Line 2.

7 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER3 Internal Clock 0 connected to the PXI External Trigger Line 3.

Internal Clock 1 sources are:

0 GX5731_INTERNAL_CLOCK0_T0_CLOCK_IN1: Internal Clock 1 connected to Clock Input 0.

1 GX5731_INTERNAL_CLOCK0_T0_PCI_CLOCK_33MHZ: Internal Clock 1 connected to 33MHz PCI Clock.

2 GX5731_INTERNAL_CLOCK0_T0_PXI_CLOCK_10MHZ: Internal Clock 1 connected to 10MHz PXI Clock.

3 GX5731_INTERNAL_CLOCK0_T0_STAR_TRIGGER: Internal Clock 1 connected to the PXI Start Trigger.

4 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER4: Internal Clock 1 connected to the PXI External Trigger Line 4.

5 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER5: Internal Clock 1 connected to the PXI External Trigger Line 5.

6 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER6 Internal Clock 1 connected to the PXI External Trigger Line 6.

7 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER7 Internal Clock 1 connected to the PXI External Trigger Line 7.

Use the **Gx5731SetInternalClock** to setup the internal clock.

Example

The following example reads clock 0 setup:

```
SHORT nSource, nStatus;
DWORD dwDivider;

Gx5731GetInternalClock (nHandle, 0, &nSource, &dwDivider, &nStatus);
```

See Also

Gx5731SetInternalClock

Gx5731GetInternalClockEnable

Purpose

Return the specified internal clock enable mode.

Syntax

Gx5731GetInternalClockEnable (*nHandle, nClock, pnEnable, dwDivider, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5731 board.
nClock	SHORT	Clock number: 0 GX5731_CLOCK0: Internal Clock 0 1 GX5731_CLOCK1: Internal Clock 1
pnEnable	PSHORT	Clock enable mode: GX5731_DISABLE_INTERNAL_CLOCK: Disable the specified clock. GX5731_ENABLE_INTERNAL_CLOCK: Enable the specified clock.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

```
SHORT nEnable, nStatus;
```

The following example return the internal clock 0 enable state:

```
Gx5731GetInternalClockEnable (nHandle, GX5731_CLOCK0, &nEnable, &nStatus);
```

See Also

Gx5731SetInternalClockEnable, Gx5731GetInternalClock

Gx5731GetPort

Purpose

Reads a port value.

Syntax

Gx5731GetPort (*nHandle, nPort, pdwValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 3-6. • Any port number 0 to 2 that has no I/O module installed.
pdwValue	PDWORD	Returned port value, 32 bit integer where each bit represents a channel. Bit 0 for channel 0 and bit 31 for channel 31.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetPort** to write data to the port when in Output direction.

Example

The following example reads port 1 value:

```
SHORT nStatus;
DWORD dwValue;

Gx5731GetPort (nHandle, 1, &dwValue, &nStatus);
```

See Also

Gx5731SetPort, **Gx5731GetPortBit**, **Gx5731GetPortByte**, **Gx5731GetPortWord, Gx5731SetPortDirection**

Gx5731GetPortBit

Purpose

Reads a port bit value.

Syntax

Gx5731GetPortBit (*nHandle, nPort, nBit, pucValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 3-6. • Any port number 0 to 2 that has no I/O module installed.
nBit	SHORT	Port bit number: 0-31. Where 0 is used for low order bit and 31 for the high order bit.
pucValue	PBYTE	Returned channel value: 0 for low state and 1 for hi state.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetPortBit** 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;

Gx5731GetPort (nHandle, 1, 30, &ucValue, &nStatus);
```

See Also

Gx5731SetPortBit, **Gx5731GetPortByte**, **Gx5731GetPortWord**, **Gx5731GetPort**

Gx5731GetPortByte

Purpose

Reads a port byte value.

Syntax

Gx5731GetPortByte (*nHandle, nPort, nByte, pucValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 3-6. • Any port number 0 to 2 that has no I/O module installed.
nByte	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte.
pucValue	PBYTE	Returned value 0-255. Each bit represents the specific channel state bit 0 for lowest channel number.
pnStatus	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetPortByte** 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;

Gx5731GetPortByte (nHandle, 1, 2, &ucValue, &nStatus);
```

See Also

Gx5731SetPortByte, Gx5731GetPortBit, **Gx5731GetPortWord**, **Gx5731GetPort**

Gx5731GetPortByteDirection

Purpose

Returns the direction of a port byte.

Syntax

Gx5731GetPortByteDirection (*nHandle, nPort, nByte, bInOut, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 3-6. • Any port number 0 to 2 that has no I/O module installed.
nByte	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte.
bInOut	PBOOL	Returned byte direction: 0. (FALSE) – Input. 1. (TRUE) – Output.
pnStatus	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;

Gx5731GetPortByteDirection (nHandle, 1, 3, &bOut, nStatus);
```

See Also

Gx5731SetPortByteDirection, Gx5731GetPortDirection

Gx5731GetPortDirection

Purpose

Returns the direction of port bytes.

Syntax

Gx5731GetPortDirection (*nHandle, nPort, pnDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 3-6. • Any port number 0 to 2 that has no I/O module installed.
pnDirection	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 3. A bit is set to hi '1' for output and low '0' for input.
pnStatus	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;

Gx5731GetPortByteDirection (nHandle, 1, &nDirection, nStatus);
```

See Also

Gx5731SetPortDirection, Gx5731GetPortByteDirection

Gx5731GetPortWord

Purpose

Reads a port word value.

Syntax

Gx5731GetPortWord (*nHandle, nPort, nWord, pwValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 3-6. • Any port number 0 to 2 that has no I/O module installed.
nWord	SHORT	Port word number: 0-1. Where 0 is for low order word (bytes 0 and 1) and 1 for the high order word (bytes 2 and 3).
pwValue	PWORD	Returned port word value: 0 to 65,535 (0-0xFFFF). Where bit 0 corresponds to channel 0 and bit 15 to channel 15.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetPortWord** 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;
Gx5731GetPortByte (nHandle, 1, 0, &wValue, &nStatus);
```

See Also

Gx5731SetPortWord, Gx5731GetPortBit, Gx5731GetPortByte, Gx5731GetPort

Gx5731GetStrobeToPxiTriggerBusState

Purpose

Returns all Strobe to Pxi Trigger lines Output Mode.

Syntax

Gx5731GetStrobeToPxiTriggerBusState (*nHandle, pnBusState, pnStatus*)

Parameters

Name	Type	Comments
Nhandle	SHORT	Handle to a GX5731 board.
pnBusState	PSHORT	Returned all eight Strobe to Pxi Trigger Bus lines Output mode, 8 bits integer where each bit represents a Pxi Trigger line. Bit 0 for line 0 and bit 7 for line 7. A high ('1') means the corresponding Pxi Trigger line is enable, low ('0') means the corresponding Pxi Trigger line is disabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetStrobeToPxiTriggerBusState** to set a specified Pxi Trigger Line output.

Example

The following example Returned all eight Strobe to Pxi Trigger Bus lines output mode:

```
SHORT nBusState, nStatus;

Gx5731GetStrobeToPxiTriggerBusState (nHandle, &nBusState, &nStatus);
```

See Also

Gx5731SetStrobeToPxiTriggerBusState, Gx5731SetStrobeToPxiTriggerLineState, Gx5731GetExtTriggerToPxiTriggerLine,

Gx5731GetStrobeToPxiTriggerLineState

Purpose

Returns the specified Strobe to Pxi Trigger Line output mode.

Syntax

Gx5731GetStrobeToPxiTriggerLineState (*nHandle, nLine, pbMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nLine	SHORT	Pxi Trigger Line numbers are: 0 GX5731_XTRIGGER_LINE0: Line 0 1 GX5731_XTRIGGER_LINE1: Line 1 2 GX5731_XTRIGGER_LINE2: Line 2 3 GX5731_XTRIGGER_LINE3: Line 3 4 GX5731_XTRIGGER_LINE4: Line 4 5 GX5731_XTRIGGER_LINE5: Line 5 6 GX5731_XTRIGGER_LINE6: Line 6 7 GX5731_XTRIGGER_LINE7: Line 7
pbMode	PBOOL	Returned Pxi trigger line mode are: 0 GX5731_XTRIGGER_LINE_DISCONNECTED: Pxi trigger line is disconnected. 1 GX5731_XTRIGGER_LINE_CONNECTED: Pxi trigger line is Connected
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetStrobeToPxiTriggerBusState** function to set all Pxi-Trigger lines at once.

Example

The following example reads Pxi Trigger line 1mode:

```
SHORT nStatus;
BOOL bMode;

Gx5731GetStrobeToPxiTriggerLineState(nHandle, GX5731_XTRIGGER_LINE1, &bMode, &nStatus);
```

See Also

Gx5731SetStrobeToPxiTriggerBusState, Gx5731SetStrobeToPxiTriggerLineState, Gx5731GetExtTriggerToPxiTriggerLine, Gx5731GetExtTriggerToPxiTriggerBus

Gx5731Initialize

Purpose

Initializes the GX5731 driver for the specified board slot.

Syntax

Gx5731Initialize (*nSlot, pnHandle, pnStatus*)

Parameters

Name	Type	Comments
nSlot	SHORT	GX5731 board slot number.
pnHandle	PSHORT	Returned Handle for a GX5731 board.
pnStatus	PSHORT	Returned status: 0 on success, 1 on failure.

Comments

The **Gx5731Initialize** function verifies whether or not the GX5731 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 GX5731 functions.

Example

The following example initializes a GX5731 boards at PCI bus slot 1.

```
SHORT nHandle, nStatus;

Gx5731Initilize(1, &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

GxPioGetErrorString, Gx5731Reset

Gx5731InitializeVisa

Purpose

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

Syntax

Gx5731InitializeVisa (*szVisaResource, pnHandle, pnStatus*)

Parameters

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

Comments

The **Gx5731InitializeVisa** 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 GX5731 boards at PXI bus 5 and device 11.

```
SHORT nHandle, nStatus;
Gx5731InitializeVisa (“PXI5::11::INSTR”, &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

Gx5731Initialize, Gx5731Reset, **GxPIOGetErrorString**

Gx5731ModuleBufferClear

Applies To

GX5701, GX5702, GX5704

Purpose

Clear the Module's Buffer.

Syntax

Gx5731ModuleBufferClear (*nHandle*, *nPort*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0. GX5731_PORT0: Module port 0 1. GX5731_PORT1: Module port 1 2. GX5731_PORT2: Module port 2
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calling this function clears the Module's Buffer. All Module's Buffer data will be erased.

Example

The following example return the clears Module 2 Buffer:

```
SHORT nStatus;
Gx5731ModuleBufferClear (nHandle, GX5731_PORT2, &nStatus);
```

See Also

Gx5731ModuleBufferReadData, Gx5731ModuleBufferWriteData

Gx5731ModuleBufferGetMode

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Buffer Full or Half-Full Mode.

Syntax

Gx5731ModuleBufferGetMode (*nHandle, nPort, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0. GX5731_PORT0: Module port 0 1. GX5731_PORT1: Module port 1 2. GX5731_PORT2: Module port 2
pnMode	PSHORT	Returned specified Module Buffer mode are as follow: 0 GX5731_MODULE_BUFFER_MODE_FULL: The module FIFO flag is set to detect a Full or Empty state. 1 GX5731_MODULE_BUFFER_MODE_HALFFULL: The module Buffer flag is set to detect a Half-Full or Half-Empty state.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example return the Module's 0 Buffer mode:

```
SHORT nStrobeWidth, nStatus;
Gx5731ModuleBufferGetMode (nHandle, GX5731_PORT0, &nStrobeWidth, &nStatus);
```

See Also

Gx5731ModuleBufferSetMode, Gx5731ModuleBufferGetState

Gx5731ModuleBufferGetState

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Buffer Full or Half-Full State.

Syntax

Gx5731ModuleBufferGetState (*nHandle*, *nPort*, *pnState*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnState	PSHORT	Returned Buffer state can be as follow: 0 GX5731_MODULE_BUFFER_EMPTY: buffer is empty (reading from the buffer) or not full (writing to the buffer). 1 GX5731_MODULE_BUFFER_FULL: buffer is full (writing to the buffer) or buffer is empty (reading from the buffer).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When reading or writing to the buffer the state is as follow:

Writing to the buffer:

0. Indicates the buffer in not yet full.
1. The buffer is full. In order to write new data the buffer needs to be cleared first by calling **Gx5731ModuleBufferClear** or by reading data from the buffer using **Gx5731ModuleBufferReadData**.

Reading from the buffer:

0. Indicates the buffer in not empty.
1. The buffer is empty.

Example

The following example returns Module 0 buffer flag state:

```
SHORT nState, nStatus;
Gx5731ModuleBufferGetState (nHandle, GX5731_PORT0, &nState, &nStatus);
```

See Also

Gx5731ModuleBufferWrite, Gx5731ModuleBufferRead, Gx5731ModuleBufferSetMode

Gx5731ModuleBufferReadData

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's buffer.

Syntax

Gx5731ModuleBufferReadData (*nHandle, nPort, pdwVector, pdwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
dwVector	PDWORD	Array of double words (32-bit)
pdwSize	PWORD	Number of requested of double words to read. Returns the actual number of double words read. The Array size can be between 0 and 4095.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function will continue to read data from the Module's buffer until the number of requested of double words to read is reached or if the buffer empty flag is set to TRUE. See the **Gx5731ModuleGetBufferFlagState** for more details.

Example

The following example reads 18 double words from module port 0:

```
SHORT nStatus;
DWORD adwVector[64];
DWORD dwSize=18;
Gx5731GetPortByte (nHandle, GX5731_PORT0, adwVector &dwSize, &nStatus);
```

See Also

Gx5731ModuleBufferWriteData, Gx5731ModuleBufferGetState

Gx5731ModuleBufferSetFlagMode

Applies To

GX5701, GX5702, GX5704

Purpose

Set the Module's Buffer Full or Half-Full Flag Mode.

Syntax

Gx5731ModuleBufferSetFlagMode (*nHandle, nPort, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnMode	SHORT	Module Buffer Flag mode are as follow: 0 GX5731_MODULE_BUFFER_FLAG_MODE_FULL: The module Buffer flag is set to detect a Full or Empty state. 1 GX5731_MODULE_BUFFER_FLAG_MODE_HALFFULL: The module Buffer flag is set to detect a Half-Full or Half-Empty state.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example return the Buffer flag mode of module port 0 :

```
SHORT nMode, nStatus;
Gx5731ModuleBufferSetFlagMode (nHandle, GX5731_PORT0, &nMode, &nStatus);
```

See Also

Gx5731ModuleBufferWriteData, Gx5731ModuleBufferReadData, Gx5731ModuleBufferGetState

Gx5731ModuleBufferSetMode

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Buffer Full or Half-Full Mode.

Syntax

Gx5731ModuleBufferSetMode (*nHandle, nPort, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nMode	SHORT	Specified Module Buffer mode are as follow: 0 GX5731_MODULE_BUFFER_MODE_FULL: The module FIFO flag is set to detect a Full or Empty state. 1 GX5731_MODULE_BUFFER_MODE_HALFFULL: The module Buffer flag is set to detect a Half-Full or Half-Empty state.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets the Module's 0 Buffer mode:

```
SHORT nStatus;
Gx5731ModuleBufferGetMode (nHandle, GX5731_PORT0, GX5731_MODULE_BUFFER_MODE_HALFFULL, &nStatus);
```

See Also

Gx5731ModuleBufferGetMode, Gx5731ModuleBufferGetState

Gx5731ModuleBufferTest

Applies To

GX5701, GX5702, GX5704

Purpose

Test the specified Module's Buffer.

Syntax

Gx5731ModuleBufferTest (*nHandle, nPort, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function runs an internal test to check the specified module's buffer for read and write operations. The specified buffer is empty upon return.

Example

The following example return the Buffer flag mode of module port 0 :

```
SHORT nMode, nStatus;
Gx5731ModuleBufferTest (nHandle, GX5731_PORT0, &nStatus);
```

See Also

Gx5731ModuleBufferWriteData, Gx5731ModuleBufferReadData, Gx5731ModuleBufferGetState

Gx5731ModuleBufferWriteData

Applies To

GX5701, GX5702, GX5704

Purpose

Write a buffer to the Module's Buffer.

Syntax

Gx5731ModuleBufferWriteData (*nHandle, nPort, pdwVector, pdwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
dwVector	PDWORD	Array of double words (32-bit)
pdwSize	PWORD	Number of requested of double words to write. Returns the actual number of double words written. The Array size can be between 0 and 4095.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function will continue to write data from the Module's buffer until the number of requested of double words to write is reached or if the buffer full flag is set to TRUE. See the **Gx5731ModuleGetBufferFlagState** for more details.

Example

The following example writes 18 double words to module port 0:

```
SHORT nStatus;
DWORD adwVector[64];
DWORD dwSize=18;
Gx5731ModuleBufferWriteData (nHandle, GX5731_PORT0, adwVector &dwSize, &nStatus);
```

See Also

Gx5731ModuleBufferReadData, Gx5731ModuleBufferGetState

Gx5731ModuleGetDirection

Applies To

GX5711, GX5712

Purpose

Returns the Module's channels direction.

Syntax

Gx5731ModuleGetDirection (*nHandle, nPort, nGroup, pnDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
NPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
NGroup	SHORT	Specified Module's port group number: 0 GX5731_MODULE_GROUP0: Module port group 0. 1 GX5731_MODULE_GROUP1: Module port group 1.
pnDirection	PSHORT	Returns Module's channels direction mode: 0 GX5731_MODULE_DIRECTION_DIFFERENTIAL_TO_TTL: Module's channels direction is from differential to TTL. 1 GX5731_MODULE_DIRECTION_TTL_TO_DIFFERENTIAL: Module's channels direction is from TTL to differential.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The data that flows through the Module can be monitor by calling **Gx5731GetPort.**

Example

The following example returns the module's port 0 group 0 direction:

```
SHORT nDirection, nStatus;
Gx5731ModuleGetDirection (nHandle, GX5731_PORT0, GX5731_MODULE_GROUP0,&nDirection, &nStatus)
```

See Also

Gx5731ModuleSetDirection, Gx5731ModuleSetDirectionSource, Gx5731GetPort, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleGetDirectionSource

Applies To

GX5711, GX5712

Purpose

Returns the Module's channels direction source.

Syntax

Gx5731ModuleGetDirectionSource (*nHandle*, *nPort*, *pnS*ource, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
*pnS*ource	PSHORT	Returned Module's channels direction source: 0 GX5731_MODULE_DIRECTION_SOURCE_INTERNAL: Module's channels direction is set internally (software). 1 GX5731_MODULE_DIRECTION_SOURCE_EXTERNAL: Module's channels direction is set externally.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The data that flows through the Module can be monitor by calling **Gx5731GetPort.**

Example

The following example returns module's port 0 direction source:

```
SHORT nSource, nStatus;
Gx5731ModuleGetDirectionSource (nHandle, GX5731_PORT0, &nSource, &nStatus)
```

See Also

Gx5731ModuleSetDirectionSource, Gx5731ModuleGetDirection, Gx5731GetPort, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleGetExtStrobeEnablePolarity

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's External Strobe Enable Polarity.

Syntax

Gx5731ModuleGetExtStrobeEnablePolarity (*nHandle, nPort, pnPolarity, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnPolarity	PSHORT	Returned External Strobe Enable Polarity mode: 0 GX5731_MODULE_STROBE_POSITIVE_EDGE: External Strobe will be enabled on positive edge. 1 GX5731_MODULE_STROBE_NEGATIVE_EDGE: External Strobe will be enabled on negative edge.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns modules port 0 external strobe enable polarity:

```
SHORT nPolarity, nStatus;
Gx5731ModuleGetExtStrobeEnablePolarity (nHandle, GX5731_PORT0, &nPolarity, &nStatus);
```

See Also

Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleGetExtStrobeEnableState

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's External Strobe Enable State.

Syntax

Gx5731ModuleGetExtStrobeEnableState (*nHandle, nPort, pnState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnState	PSHORT	Returned External Strobe Enable State mode: 0 External Strobe is disabled ('0'). 1 External Strobe is enabled ('1').
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function returns the state of the External Strobe Enable input signal on the connector J6 pin 68. This input is used by the Handshake mechanism and can be read at any time.

Example

The following example returns modules port 0 external strobe enable state:

```
SHORT nState, nStatus;
Gx5731ModuleGetExtStrobeEnableState (nHandle, GX5731_PORT0, &nState, &nStatus);
```

See Also

Gx5731ModuleGetExtStrobeState, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleGetExtStrobePolarity

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's External Strobe Polarity.

Syntax

Gx5731ModuleGetExtStrobePolarity (*nHandle, nPort, pnPolarity, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnPolarity	PSHORT	Returned External Strobe Polarity mode: 0 GX5731_MODULE_STROBE_POSITIVE_EDGE: External Strobe is active on positive edge. 1 GX5731_MODULE_STROBE_NEGATIVE_EDGE: External Strobe is active on negative edge.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns modules port 0 external strobe polarity:

```
SHORT nPolarity, nStatus;
Gx5731ModuleGetExtStrobePolarity (nHandle, GX5731_PORT0, &nPolarity, &nStatus);
```

See Also

Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleGetExtStrobeState

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's External Strobe State.

Syntax

Gx5731ModuleGetExtStrobeState (*nHandle*, *nPort*, *pnState*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnState	PSHORT	Returned External Strobe Enable State mode: 0 External Strobe state is low ('0'). 1 External Strobe state is high ('1').
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function returns the state of the External Strobe input signal on the connector J6 pin 67. This input is used by the Handshake mechanism and can be read at any time.

Example

The following example returns modules port 0 external strobe state:

```
SHORT nState, nStatus;
Gx5731ModuleGetExtStrobeState (nHandle, GX5731_PORT0, &nState, &nStatus);
```

See Also

Gx5731ModuleGetExtStrobeEnableState, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleGetHandshakeOutputMode

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Handshake output mode.

Syntax

Gx5731ModuleGetHandshakeOutputMode (*nHandle*, *nPort*, *pnMode*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnMode	PSHORT	Handshake output mode: 0 GX5731_MODULE_HANDSHAKE_OUTPUT_DISABLE: Handshake output mode disabled. 1 GX5731_MODULE_HANDSHAKE_OUTPUT_ENABLE: Handshake output mode inabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX5731 can have up to three Modules installed. Each installed module can use Direct Output/Input or Strobed Output/Input with handshaking.

When using Direct Output/Input on each strobe signal transition (software controlled positive edge or negative edge).

When using Strobed Output/Input with handshaking the hardware controls the Output/Input sequences. The handshaking mechanism is implemented using input and output control and strobe signals. See **Chapter 2: Module Strobed Output/Input Handshaking** section for typical handshaking events in order to output a single double word of data.

Example

The following example returns modules port 0 Handshake Output mode:

```
SHORT nMode nStatus;
Gx5731ModuleGetHandshakeOutputMode (nHandle, GX5731_PORT0, &nMode, &nStatus);
```

See Also

Gx5731ModuleSetHandshakeOutputMode, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleGetInternalStrobeSource

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Internal Strobe Clock source.

Syntax

Gx5731ModuleGetInternalStrobeSource (*nHandle, nPort, pnSource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nSource	SHORT	Returned source: 0 GX5731_CLOCK0: Internal Clock0. 1 GX5731_CLOCK1: Internal Clock1 2 GX5731_PXI_CLOCK_USER_LINE2: User External input line2. 3 GX5731_PXI_CLOCK_USER_LINE3: User External input line3. 4 GX5731_PXI_CLOCK_USER_LINE4: ser External input line4.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns modules port 0 internal strobe source:

```
SHORT nSource, nStatus;
Gx5731ModuleGetInternalStrobeSource (nHandle, GX5731_PORT0, &nSource, &nStatus);
```

See Also

Gx5731ModuleSetInternalStrobeSource, Gx5731SetInternalClock, Gx5731SetInternalClockEnable, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetStrobeSource

Gx5731ModuleGetOperationMode

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Operation mode.

Syntax

Gx5731ModuleGetOperationMode (*nHandle*, *nPort*, *pnMode*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnMode	PSHORT	Returns the Module's Operation mode as follow: 0 GX5731_MODULE_OPERATION_BUFFERED: Default operation mode, in this mode the Module is using its buffer to read/write (depend on its type input or output). 1 GX5731_MODULE_OPERATION_PORT_IO: The user can use the Module's as a static port in or port out (depend on its type input or output)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If Module type is either GX5702 or GX5704 and the Operation mode is Port I/O (GX5731_MODULE_OPERATION_PORT_IO) the outputs can be written statically.

If Module type is GX5701 and the Operation mode is Port I/O (GX5731_MODULE_OPERATION_PORT_IO) the inputs can be read statically.

Example

The following example return the module port 0 operation mode:

```
SHORT nMode, nStatus;

Gx5731ModuleGetOperationMode (nHandle, GX5731_PORT0, &nMode, &nStatus);
```

See Also

Gx5731ModuleSetOperationMode, **Gx5731ModuleSetPort, Gx5731ModuleGetPort, Gx5731moduleSetOutputMode, Gx5731moduleSetStrobeSource**

Gx5731ModuleGetOutputMode

Applies To

GX5702, GX5704

Purpose

Returns the Module's Output mode.

Syntax

Gx5731ModuleGetOutputMode (*nHandle, nPort, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnMode	PSHORT	Returns the Module's Output mode as follow: 0 GX5731_MODULE_OUTPUT_DISABLE:All the Module's 32 channels are in Tri-State mode. 1 GX5731_MODULE_OUTPUT_ENABLE: All the Module's 32 channels are enabled to output data..
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Outputs are set to Tri-State when in disabled.

Example

The following example return the module port 0 output mode:

```
SHORT nMode, nStatus;

Gx5731ModuleGetOutputMode (nHandle, GX5731_PORT0, &nMode, &nStatus);
```

See Also

Gx5731moduleSetOutputMode, Gx5731moduleSetStrobeSource

Gx5731ModuleGetOutputStrobeWidth

Applies To

GX5702, GX5704

Purpose

Returns the Module's Output Strobe Width.

Syntax

Gx5731ModuleGetOutputStrobeWidth (*nHandle, nPort, pnStrobeWidth, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnStrobeWidth	PSHORT	Returned integer value representing the Output Strobe Width in nSec. Strobe Width value range is 50 to 5000 with resolution of 50nSec..
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example return the module port 0 strobe width:

```
SHORT nStrobeWidth, nStatus;

Gx5731ModuleGetOutputStrobeWidth (nHandle, GX5731_PORT0, &nStrobeWidth, &nStatus);
```

See Also

Gx5731moduleSetOutputStrobeWidth, Gx5731moduleSetStrobeSource

Gx5731ModuleGetPort

Applies To

GX5701

Purpose

Returns the Module's Port Data.

Syntax

Gx5731ModuleGetPort (*nHandle, nPort, pdwData, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pdwData	PDWORD	Returned port double word value: 0 to -0xFFFFFFFF. Where bit 0 corresponds to channel 0 and bit 31 to channel 31.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows the module to be used as a port in. Prior to calling this function the module's operating mode needs to be set to Port I/O (GX5731_MODULE_OPERATION_PORT_IO). Use the **Gx5731ModuleSetOpeartionMode** to set the module's operating mode.

If no Module is installed in the specified Module's port, the port is functioning as Digital Port Input only. Calling this function will return the Port input data

Note: Calling this function will clear the Module's buffer.

Example

The following example reads the module port 0 data:

```
SHORT nStatus;
DWORD dwData;
Gx5731ModuleGetPort (nHandle, GX5731_PORT0, &dwData, &nStatus);
```

See Also

Gx5731ModuleSetOpeartionMode, Gx5731ModuleGetOpeartionMode Gx5731ModuleSetPort

Gx5731ModuleGetPxiTriggerBus

Purpose

Returns all specified Module Strobe to Pxi Trigger lines connection mode.

Syntax

Gx5731ModuleGetPxiTriggerBus (*nHandle, pnPxiTrigBus, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnPxiTrigBus	PSHORT	Returns the specified Module's eight output Strobe lines connections to the Pxi Trigger Bus lines. The Strobe lines connections are 8 bits integer where each bit represents a Pxi Trigger line. Bit 0 for line 0 and bit 7 for line 7. A high ('1') means the corresponding Pxi Trigger line is connected; low ('0') means the corresponding Pxi Trigger line is disconnected.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Connections to the Pxi Trigger Bus lines from the Module eight output Strobe lines can be done ion parallel, i.e. from any of currently installed Modules.

Example

The following example return all eight Strobe to Pxi Trigger Bus lines to 0x5:

```
SHORT nPxiTrigBus, nStatus;

Gx5731ModuleGetPxiTriggerBus (nHandle, &nPxiTrigBus, &nStatus);
```

See Also

Gx5731ModuleSetPxiTriggerBus, Gx5731ModuleSetPxiTriggerLine, Gx5731GetStrobeToPxiTriggerBusState, Gx5731SetStrobeToPxiTriggerLineState, Gx5731GetExtTriggerToPxiTriggerLine,

Gx5731ModuleGetPxiTriggerLine

Purpose

Returns the specified Module's Strobe to Pxi Trigger Line connection mode.

Syntax

Gx5731ModuleGetPxiTriggerLine (*nHandle, nPxiTrigLine, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nPxiTrigLine	SHORT	Pxi Trigger Line numbers are: 0 GX5731_XTRIGGER_LINE0: Line 0 1 GX5731_XTRIGGER_LINE1: Line 1 2 GX5731_XTRIGGER_LINE2: Line 2 3 GX5731_XTRIGGER_LINE3: Line 3 4 GX5731_XTRIGGER_LINE4: Line 4 5 GX5731_XTRIGGER_LINE5: Line 5 6 GX5731_XTRIGGER_LINE6: Line 6 7 GX5731_XTRIGGER_LINE7: Line 7
pnMode	PSHORT	Pxi trigger line mode are: 0 GX5731_XTRIGGER_LINE_DISCONNECTED: Pxi trigger line is disconnected. 1 GX5731_XTRIGGER_LINE_CONNECTED: Pxi trigger line is Connected
PnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731ModuleSetPxiTriggerBus** function to set all Pxi-Trigger lines at once.

Example

The following example returns Pxi Trigger line 1 connection mode:

```
SHORT nMode, nStatus;

Gx5731ModuleGetPxiTriggerLine (nHandle, GX5731_XTRIGGER_LINE1, &nMode, &nStatus);
```

See Also

Gx5731ModuleSetPxiTriggerLine, Gx5731ModuleSetPxiTriggerBus, Gx5731GetStrobeToPxiTriggerBusState, Gx5731SetStrobeToPxiTriggerLineState, Gx5731GetExtTriggerToPxiTriggerLine,

Gx5731ModuleGetRevision

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Revision.

Syntax

Gx5731ModuleGetRevision (*nHandle*, *nPort*, *pnRevision*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnRevision	PSHORT	Return the Module's revision.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example reads module port 0 revision:

```
SHORT nRevision, nStatus;

Gx5731ModuleGetRevision (nHandle, GX5731_PORT0, &nRevision, &nStatus);
```

See Also

Gx5731GetModuleType

Gx5731ModuleGetState

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's State.

Syntax

Gx5731ModuleGetState (*nHandle, nPort, pnState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnState	PSHORT	Return the Module State as follow: 0 GX5731_MODULE_STATE _HALT: The Module is in Halt state. 1 GX5731_MODULE_STATE_RUN: The Module is in Run state.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the current module state. Use this function in order to monitor the module's state

Example

The following example returns module port 0 state:

```
SHORT nState, nStatus;

Gx5731ModuleGetState (nHandle, GX5731_PORT0, &nState, &nStatus);
```

See Also

Gx5731ModuleRun, Gx5731ModuleHalt, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleGetStrobeSource

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Strobe Source.

Syntax

Gx5731ModuleGetStrobeSource (*nHandle*, *nPort*, *pnSource*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnSource	PSHORT	Return the Strobe Source as follow: 0 GX5731_MODULE_STROBE_INTERNAL: Strobe source is internal. 1 GX5731_MODULE_STROBE_EXTERNAL: Strobe source is external.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the Strobe Source is set to Internal see **Gx5731ModuleSetStrobeSource** functions for more details.

If the Strobe Source is set to External see **Gx5731ModuleSetExtStrobePolarity** and Gx5731ModuleSetExtStrobeEnablePolarity functions for more details.

The following example set module port 0 Strobe Source to external:

```
SHORT nSource, nStatus;

Gx5731ModuleGetStrobeSource (nHandle, GX5731_PORT0, &nSource, &nStatus);
```

See Also

Gx5731ModuleSetStrobeSource, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleGetExtStrobePolarity, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleGetExtStrobeEnablePolarity

Gx5731ModuleGetTermination

Applies To

GX5709, GX5711, GX5712

Purpose

Returns the Module termination mode.

Syntax

Gx5731ModuleGetTermination (*nHandle, nPort, pnTermination, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnTermination	PSHORT	GX5709 module: Module bytes Terminations where bit 0 corresponds to byte 0, bit 1 to byte 1, bit 2 byte 2 and bit 3 to byte 3. A bit is set to hi '1' for Termination On and low '0' for Termination Off. GX5711 and GX5712 modules: Module Terminations, bit 0 is set to hi '1' for Termination On and low '0' for Termination Off.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Settings the terminators to be On will only be active only when outputting data through the differential lines.

Example

The following example returns Module port 0 terminators mode:

```
SHORT nTermination, nStatus;
Gx5731ModuleGetTermination (nHandle, GX5731_PORT0, &nTermination, &nStatus);
```

See Also

Gx5731ModuleSetTermination, **Gx5731ModuleGetType**

Gx5731ModuleGetType

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's type.

Syntax

Gx5731ModuleGetType (*nHandle, nPort, pnType, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnType	PSHORT	Returned Module types, see comments for details.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

5700 GX5731_NO_MODULE_INSTALLED: No Module installed in the specified port.

5701 GX5731_DIGITAL_INPUT_LATCH: 32-channel digital input module with variable input threshold and rapid data comparison.

5702 GX5731_DIGITAL_OUTPUT_LATCH: 32-channel digital output module with TTL levels that can be used for digital control and propagation delay testing.

5704 GX5731_DIGITAL_POWER_OUTPUT_LATCH: Module capable of driving loads or driving signals over long distances.

5709 RS422 Port I/O Module is a differential RS422 32 I/O channels module.

5711 Bidirectional LVDS-TTL Converter Module is a 16 channels module converting signals from LVSD to TTL or vise versa.

5712 Bidirectional RS422-TTL Converter Module is a 16 channels module converting signals from RS422 to TTL or vise versa.

5714 GX5731_DIGITAL_PORT_IO: Digital port I/O.

5715 GX5731_PORT_IN_ONLY: This is the most basic configuration. The port is only capable to serve as input.

When the module type is GX5731_PORT_IN_ONLY there are certain jumpers that needs to be installed.

Example

The following example reads module port 0 type:

```
SHORT nType, nStatus;
Gx5731GetPortByte (nHandle, GX5731_PORT0, &nType, &nStatus);
```

See Also

Gx5731GetRevision

Gx5731ModuleGetVThreshold

Applies To

GX5701

Purpose

Returns the Module's Threshold voltage.

Syntax

Gx5731ModuleGetVThreshold (*nHandle, nPort, pdVThreshold, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pdVThreshold	PDOUBLE	Returned Module's threshold voltage. Threshold voltage range is from –30 to +30 volts with resolution of 16-bits
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example return the threshold voltage of module port 0:

```
SHORT nStatus;
DOUBLE dVThreshold;
Gx5731ModuleSetVThreshold (nHandle, GX5731_PORT0, &dVThreshold, &nStatus);
```

See Also

Gx5731ModuleSetVThreshold, Gx5731ModuleGetType

Gx5731ModuleHalt

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the specified Module to Halt state.

Syntax

Gx5731ModuleHalt (*nHandle, nPort, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calling this function will take the Module from Run to Halt state. Use **Gx5731ModuleGetState** in order to monitor the module state.

Example

The following example set module port 0 to HALT mode:

```
SHORT nPolarity, nStatus;
Gx5731ModuleHalt (nHandle, GX5731_PORT0, &nStatus);
```

See Also

Gx5731ModuleGetState, Gx5731ModuleRun, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleRun

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the specified Module to Run mode.

Syntax

Gx5731ModuleRun (*nHandle, nPort, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Once in Run mode Module will start running depend on the strobe settings (i.e. Internal or External). Use **Gx5731ModuleGetState** in order to monitor the module state.

Example

The following example set module port 0 to RUN mode:

```
SHORT nPolarity, nStatus;
Gx5731ModuleRun (nHandle, GX5731_PORT0, &nStatus);
```

See Also

Gx5731ModuleGetState, Gx5731ModuleHalt, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleSetDirection

Applies To

GX5711, GX5712

Purpose

Sets the Module's channels direction.

Syntax

Gx5731ModuleSetDirection (*nHandle, nPort, nGroup, nDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nGroup	SHORT	Specified Module's port group number: 0 GX5731_MODULE_GROUP0: Module port group 0. 1 GX5731_MODULE_GROUP1: Module port group 1.
nDirection	SHORT	Module's channels direction mode: 0 GX5731_MODULE_DIRECTION_DIFFERENTIAL_TO_TTL: Module's channels direction is from differential to TTL. 1 GX5731_MODULE_DIRECTION_TTL_TO_DIFFERENTIAL: Module's channels direction is from TTL to differential.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The data that flows through the Module can be monitor by calling **Gx5731GetPort.**

Example

The following example sets module's port 0 group 0 direction from Differential to TTL:

```
SHORT nStatus;
Gx5731ModuleSetDirection (nHandle, GX5731_PORT0, GX5731_MODULE_GROUP0,
                    GX5731_MODULE_DIRECTION_DIFFERENTIAL_TO_TTL, &nStatus)
```

See Also

Gx5731ModuleGetDirection, Gx5731ModuleSetDirectionSource, Gx5731GetPort, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleSetDirectionSource

Applies To

GX5711, GX5712

Purpose

Sets the Module's channels direction source.

Syntax

Gx5731ModuleSetDirectionSource (*nHandle*, *nPort*, *nS*ource, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nSource	SHORT	Module's channels direction source: 0 GX5731_MODULE_DIRECTION_SOURCE_INTERNAL: Module's channels direction is set internally (software). 1 GX5731_MODULE_DIRECTION_SOURCE_EXTERNAL: Module's channels direction is set externally.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The data that flows through the Module can be monitor by calling **Gx5731GetPort.**

Example

The following example sets module's port 0 direction source to External:

```
SHORT nStatus;
Gx5731ModuleSetDirectionSource (nHandle, GX5731_PORT0, GX5731_MODULE_DIRECTION_SOURCE_INTERNAL,
                    &nStatus)
```

See Also

Gx5731ModuleGetDirectionSource, Gx5731ModuleGetDirection, Gx5731GetPort, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleSetExtStrobeEnablePolarity

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's External Strobe Enable Polarity.

Syntax

Gx5731ModuleSetExtStrobeEnablePolarity (*nHandle*, *nPort*, *nPolarity*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nPolarity	SHORT	External Strobe Enable Polarity mode: 0 GX5731_MODULE_STROBE_POSITIVE_EDGE: External Strobe is enabled on positive edge. 1 GX5731_MODULE_STROBE_NEGATIVE_EDGE: External Strobe is enabled on negative edge.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetPortWord** to write data to the port word when in Output direction.

Example

The following example returns modules port 0 external enable strobe polarity:

```
SHORT nPolarity, nStatus;
Gx5731ModuleGetExtStrobeEnPolarity (nHandle, GX5731_PORT0, &nPolarity, &nStatus);
```

See Also

Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleSetExtStrobePolarity

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's External Strobe Polarity.

Syntax

Gx5731ModuleSetExtStrobePolarity (*nHandle*, *nPort*, *nPolarity*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nPolarity	SHORT	External Strobe Polarity mode: 0 GX5731_MODULE_STROBE_POSITIVE_EDGE: External Strobe active on positive edge. 1 GX5731_MODULE_STROBE_NEGATIVE_EDGE: External Strobe active on negative edge.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns modules port 0 external strobe polarity:

```
SHORT nPolarity, nStatus;
Gx5731ModuleGetExtStrobePolarity (nHandle, GX5731_PORT0, &nPolarity, &nStatus);
```

See Also

Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleSetHandshakeOutputMode

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Handshake output mode.

Syntax

Gx5731ModuleSetHandshakeOutputMode (*nHandle*, *nPort*, *nMode*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nMode	SHORT	Handshake output mode: 0 GX5731_MODULE_HANDSHAKE_OUTPUT_DISABLE: Handshake output mode disabled. 1 GX5731_MODULE_HANDSHAKE_OUTPUT_ENABLE: Handshake output mode inabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX5731 can have up to three Modules installed. Each installed module can use Direct Output/Input or Strobed Output/Input with handshaking.

When using Direct Output/Input on each strobe signal transition (software controlled positive edge or negative edge).

When using Strobed Output/Input with handshaking the hardware controls the Output/Input sequences. The handshaking mechanism is implemented using input and output control and strobe signals. See **Chapter 2: Module Strobed Output/Input Handshaking** section for typical handshaking events in order to output a single double word of data.

Example

The following example enable modules port 0 Handshake Output:

```
SHORT nStatus;
Gx5731ModuleSetHandshakeOutputMode (nHandle, GX5731_PORT0, GX5731_MODULE_HANDSHAKE_OUTPUT_ENABLE,
                                    &nStatus);
```

See Also

Gx5731ModuleGetHandshakeOutputMode, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetStrobeSource, Gx5731ModuleSetInternalStrobeSource

Gx5731ModuleSetInternalStrobeSource

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Internal Strobe Clock source.

Syntax

Gx5731ModuleSetInternalStrobeSource (*nHandle, nPort, nSource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nSource	SHORT	Strobe Source: 0 GX5731_CLOCK0: Internal Clock0. 1 GX5731_CLOCK1: Internal Clock1 2 GX5731_PXI_CLOCK_USER_LINE2: User External input line2. 3 GX5731_PXI_CLOCK_USER_LINE3: User External input line3. 4 GX5731_PXI_CLOCK_USER_LINE4: User External input line4.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets modules port 0 internal strobe source to Internal Clock0:

```
SHORT nStatus;
Gx5731ModuleSetInternalStrobeSource (nHandle, GX5731_PORT0, GX5731_CLOCK0, &nStatus);
```

See Also

Gx5731ModuleSetInternalStrobeSource, Gx5731SetInternalClock, Gx5731SetInternalClockEnable, Gx5731ModuleSetExtStrobeEnablePolarity, Gx5731ModuleSetExtStrobePolarity, Gx5731ModuleSetStrobeSource

Gx5731ModuleSetOperationMode

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Operation mode.

Syntax

Gx5731ModuleSetOperationMode (*nHandle, nPort, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nMode	SHORT	Module's Operation mode as follow: 0 GX5731_MODULE_OPERATION_BUFFERED: Default operation mode, in this mode the Module is using its buffer to read/write (depend on its type input or output). 1 GX5731_MODULE_OPERATION_PORT_IO: The user can use the Module's as a static port in or port out (depend on its type input or output)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If Module type is either GX5702 or GX5704 and the Operation mode is Port I/O (GX5731_MODULE_OPERATION_PORT_IO) the outputs can be written statically.

If Module type is GX5701 and the Operation mode is Port I/O (GX5731_MODULE_OPERATION_PORT_IO) the inputs can be read statically.

Example

The following example sets the module port 0 operation mode to Port I/O:

```
SHORT nStatus;

Gx5731ModuleGetOperationMode (nHandle, GX5731_PORT0, GX5731_MODULE_OPERATION_PORT_IO, &nStatus);
```

See Also

Gx5731ModuleGetOperationMode, **Gx5731ModuleSetPort, Gx5731ModuleGetPort, Gx5731moduleSetOutputMode, Gx5731moduleSetStrobeSource**

Gx5731ModuleSetOutputMode

Applies To

GX5702, GX5704

Purpose

Sets the Module's Output mode.

Syntax

Gx5731ModuleSetOutputMode (*nHandle, nPort, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nMode	SHORT	Module's Output mode as follow: 0 GX5731_MODULE_OUTPUT_DISABLE:All the Module's 32 channels are in Tri-State mode. 1 GX5731_MODULE_OUTPUT_ENABLE: All the Module's 32 channels are enabled to output data..
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Outputs are set to Tri-State when in disabled.

Example

The following example enables the module port 0 output mode:

```
SHORT nStatus;

Gx5731ModuleGetOutputMode (nHandle, GX5731_PORT0, GX5731_MODULE_OUTPUT_ENABLE, &nStatus);
```

See Also

Gx5731moduleSetOutputMode, Gx5731moduleSetStrobeSource

Gx5731ModuleSetOutputStrobeWidth

Applies To

GX5702, GX5704

Purpose

Sets the Module's Output Strobe Width.

Syntax

Gx5731ModuleSetOutputStrobeWidth (*nHandle, nPort, nStrobeWidth, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nStrobeWidth	SHORT	Integer value representing the Output Strobe Width in nSec. Strobe Width value range is 50 to 5000 with resolution of 50 nSec.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example set module port 0 strobe width to 100nSec:

```
SHORT nStatus;
Gx5731ModuleSetOutputStrobeWidth (nHandle, GX5731_PORT0, 2, &nStatus);
```

See Also

Gx5731moduleGetOutputStrobeWidth, Gx5731moduleSetStrobeSource

Gx5731ModuleSetPort

Applies To

GX5702, GX5704

Purpose

Sets the Module's Port Data.

Syntax

Gx5731ModuleGetPort (*nHandle, nPort, dwData, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
dwData	DWORD	Sets the port word value: 0-0xFFFFFFFF. Where bit 0 corresponds to channel 0 and bit 31 to channel 31.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows the module to be used as a port out. Prior to calling this function the module's operating mode needs to be set to Port I/O (GX5731_MODULE_OPERATION_PORT_IO). Use the **Gx5731ModuleSetOpeartionMode** to set the module's operating mode.

Note: Calling this function will clear the Module's buffer.

Example

The following example sets the module port 0 data:

```
SHORT nStatus;
Gx5731ModuleSetPort (nHandle, GX5731_PORT0, 0xAAAA5555, &nStatus);
```

See Also

Gx5731ModuleSetOpeartionMode, Gx5731ModuleGetOpeartionMode, Gx5731ModuleGetPort

Gx5731ModuleSetPxiTriggerBus

Purpose

Sets all specified Module Strobe to Pxi Trigger lines connection mode.

Syntax

Gx5731ModuleSetPxiTriggerBus (*nHandle, nPxiTrigBus, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nPxiTrigBus	SHORT	Sets the specified Module's eight output Strobe lines connections to the Pxi Trigger Bus lines. The Strobe lines connections are 8 bits integer where each bit represents a Pxi Trigger line. Bit 0 for line 0 and bit 7 for line 7. A high ('1') means the corresponding Pxi Trigger line is connected; low ('0') means the corresponding Pxi Trigger line is disconnected.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Connections to the Pxi Trigger Bus lines from the Module eight output Strobe lines can be done ion parallel, i.e. from any of currently installed Modules.

Example

The following example sets all eight Strobe to Pxi Trigger Bus lines to 0x5:

```
SHORT nStatus;

Gx5731ModuleSetPxiTriggerBus (nHandle, 0x5, &nStatus);
```

See Also

Gx5731ModuleGetPxiTriggerBus, Gx5731ModuleSetPxiTriggerLine, Gx5731GetStrobeToPxiTriggerBusState, Gx5731SetStrobeToPxiTriggerLineState, Gx5731GetExtTriggerToPxiTriggerLine,

Gx5731ModuleSetPxiTriggerLine

Purpose

Sets the specified Module's Strobe to Pxi Trigger Line connection mode.

Syntax

Gx5731ModuleSetPxiTriggerLine (*nHandle, nPxiTrigLine, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nPxiTrigLine	SHORT	Pxi Trigger Line numbers are: 0 GX5731_XTRIGGER_LINE0: Line 0 1 GX5731_XTRIGGER_LINE1: Line 1 2 GX5731_XTRIGGER_LINE2: Line 2 3 GX5731_XTRIGGER_LINE3: Line 3 4 GX5731_XTRIGGER_LINE4: Line 4 5 GX5731_XTRIGGER_LINE5: Line 5 6 GX5731_XTRIGGER_LINE6: Line 6 7 GX5731_XTRIGGER_LINE7: Line 7
nMode	SHORT	Pxi trigger line mode are: 0 GX5731_XTRIGGER_LINE_DISCONNECTED: Pxi trigger line is disconnected. 1 GX5731_XTRIGGER_LINE_CONNECTED: Pxi trigger line is Connected
PnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731ModuleSetPxiTriggerBus** function to set all Pxi-Trigger lines at once.

Example

The following example set Pxi Trigger line 1 connection mode:

```
SHORT nStatus;

Gx5731ModuleSetPxiTriggerLine (nHandle, GX5731_XTRIGGER_LINE1, GX5731_XTRIGGER_LINE_CONNECTED,
                               &nStatus);
```

See Also

Gx5731ModuleGetPxiTriggerLine, Gx5731ModuleSetPxiTriggerBus, Gx5731GetStrobeToPxiTriggerBusState, Gx5731SetStrobeToPxiTriggerLineState, Gx5731GetExtTriggerToPxiTriggerLine,

Gx5731ModuleSetStrobeSource

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Strobe Source.

Syntax

Gx5731ModuleSetStrobeSource (*nHandle, nPort, nSource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nSource	SHORT	Sets the Strobe Source as follow: 0 GX5731_MODULE_STROBE_INTERNAL: Strobe source is internal. 1 GX5731_MODULE_STROBE_EXTERNAL: Strobe source is external.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the Strobe Source is set to Internal see **Gx5731ModuleSetStrobeSource** functions for more details.

If the Strobe Source is set to External see **Gx5731ModuleSetExtStrobePolarity** and **Gx5731SetExtStrobeEnablePolarity** functions for more details.

Example

The following example set module port 0 Strobe Source to external:

```
SHORT nStatus;
Gx5731ModuleSetStrobeSource (nHandle, GX5731_PORT0, GX5731_MODULE_STROBE_EXTERNAL, &nStatus);
```

See Also

Gx5731ModuleGetStrobeSource, Gx5731ModuleSetExtStrobePolarity, Gx5731GetExtStrobePolarity, Gx5731SetExtStrobeEnablePolarity, Gx5731GetExtStrobeEnablePolarity

Gx5731ModuleSetVThreshold

Applies To

GX5701

Purpose

Sets the Module's Threshold voltage.

Syntax

Gx5731ModuleSetVThreshold (*nHandle, nPort, dVThreshold, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
dVThreshold	DOUBLE	Returned Module's threshold voltage. Threshold voltage range is from –30 to +30 volts with resolution of 16-bits.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets module port 0 the threshold voltage to 6.5 volts:

```
SHORT nStatus;

Gx5731ModuleSetVThreshold (nHandle, GX5731_PORT0, 6.5, &nStatus);
```

See Also

Gx5731ModuleGetVThreshold, **Gx5731ModuleGetType**

Gx5731ModuleStep

Applies To

GX5701, GX5702, GX5704

Purpose

Run the specified port Module for given number of steps.

Syntax

Gx5731ModuleStep (*nHandle, nPort, dwSteps, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
dwSteps	DWORD	The number of steps to run.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example run port Module 0 for three steps:

```
SHORT nStatus;

Gx5731ModuleStep (nHandle, GX5731_PORT0, 3, &nStatus);
```

See Also

Gx5731SetInternalClock, **Gx5731GetInternalClock**, **Gx5731ModuleGetType**

Gx5731ModuleSetTermination

Applies To

GX5709, GX5711, GX5712

Purpose

Sets the Module termination mode.

Syntax

Gx5731ModuleSetTermination (*nHandle, nPort, nTermination, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Specified Module's port number: 0 GX5731_PORT0: Module port 0 1 GX5731_PORT1: Module port 1 2 GX5731_PORT2: Module port 2
nTermination	SHORT	GX5709 module: Module bytes Terminations where bit 0 corresponds to byte 0, bit 1 to byte 1, bit 2 byte 2 and bit 3 to byte 3. A bit is set to hi '1' for Termination On and low '0' for Termination Off. GX5711 and GX5712 modules: Module Terminations, bit 0 is set to hi '1' for Termination On and low '0' for Termination Off.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Settings the terminators to be On will only be active only when outputting data through the differential lines.

Example

The following example sets the Module GX5712 terminators to be On:

```
SHORT nStatus;

Gx5731ModuleSetTermination (nHandle, GX5731_PORT0, 1, &nStatus);
```

See Also

Gx5731ModuleGetTermination, **Gx5731ModuleGetType**

Gx5731Panel

Purpose

Opens a virtual panel used to interactively control the GX5731.

Syntax

Gx5731Panel (pnHandle, hwndParent, nMode, phwndPanel, pnStatus)

Parameters

Name	Type	Comments
pnHandle	PSHORT	Handle to a GX5731 board.
hwndParent	HWND	Panel parent window handle. A value of 0 sets the desktop as the parent window.
nMode	SHORT	The mode in which the panel main window is created. 0 for modeless window and 1 for modal window.
phwndPanel	HWND	Returned window handle for the panel.
pnStatus	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 GX5731 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;

Gx5731Panel(&nHandle, 0, 1, &dwPanel, &nStatus);
```

See Also

Gx5731Initialize, GxPioGetErrorString

Gx5731Reset

Purpose

Resets the GX5731 board to its default settings.

Syntax

Gx5731Reset (*nHandle, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5731 board.
pnStatus	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 GX5731 board:

```
Gx5731Initialize (1, &nHandle, &nStatus);
Gx5731Reset (nHandle, &nStatus);
```

See Also

Gx5731Initialize, Gx5731ResetPort

Gx5731ResetPort

Purpose

Resets the specified digital I/O port.

Syntax

Gx5731ResetPort (*nHandle, nPort, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5731 board.
nPort	SHORT	Digital I/O port number: 0-6.
pnStatus	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:

```
Gx5731ResetPort (nHandle, 1, &nStatus);
```

See Also

Gx5731Reset, Gx5731Initialize

Gx5731SetExtTriggerToPxiTriggerBus

Purpose

Sets the connections for all External Trigger lines to their corresponding Pxi Triggers lines number.

Syntax

Gx5731SetExtTriggerToPxiTriggerBus (*nHandle, nExtTrigToXTrig, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nExtTrigToXTrig	SHORT	Sets the connections for all External Trigger lines to their corresponding Pxi Triggers lines number. Each of bits 0-7 represents a External Trigger lines, bit 0 for line 0 and bit 7 for line 7.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

External Trigger lines can only be connected to their corresponding Pxi Triggers lines number. I.e. External Trigger line 0 can only be connected to Pxi Triggers line 0 and so forth.

Use the **Gx5731SetExtTriggerToPxiTriggerLine**to set a specified External Trigger line to Pxi Trigger Line.

Example

The following example connects External Trigger lines 0, 2,5 to their corresponding Pxi Triggers lines and disconnects all other lines:

```
SHORT nXTrig, nStatus;

Gx5731SetExtTriggerToPxiTriggerBus (nHandle, 0x25, &nStatus);
```

See Also

Gx5731GetExtTriggerToPxiTriggerBus, **Gx5731SetExtTriggerToPxiTriggerLine**, **Gx5731SetStrobeToPxiTriggerBusState**

Gx5731SetExtTriggerToPxiTriggerLine

Purpose

Sets the connection for the specified External Trigger to Pxi Trigger line.

Syntax

Gx5731SetExtTriggerToPxiTriggerLine (*nHandle, nExtTrigToXTrigLine, bMode, pnStatus*)

Parameters

Name	Type	Comments
NHandle	SHORT	Handle to a GX5731 board.
NExtTrigToXTrigLine	SHORT	External Trigger to Pxi Trigger Line numbers are: 0 GX5731_EXT_TRIGGER_LINE0: External Trigger 0. 1 GX5731_EXT_TRIGGER_LINE1: External Trigger 1. 2 GX5731_EXT_TRIGGER_LINE2: External Trigger 2. 3 GX5731_EXT_TRIGGER_LINE3: External Trigger 3. 4 GX5731_EXT_TRIGGER_LINE4: External Trigger 4. 5 GX5731_EXT_TRIGGER_LINE5: External Trigger 5. 6 GX5731_EXT_TRIGGER_LINE6: External Trigger 6. 7 GX5731_EXT_TRIGGER_LINE7: External Trigger 7.
bMode	BOOL	Pxi trigger line mode are: 0 GX5731_XTRIGGER_LINE_DISCONNECTED: Pxi trigger line is disconnected. 1 GX5731_XTRIGGER_LINE_CONNECTED: Pxi trigger line is Connected
PnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetExtTriggerToPxiTriggerBus** to set External Trigger lines can only be connected to their corresponding Pxi Triggers lines number.

Example

The following example connects External Trigger lines 2 to Pxi Triggers line 2:

```
SHORT nStatus;

Gx5731SetExtTriggerToPxiTriggerLine (nHandle, GX5731_EXT_TRIGGER_LINE2,
                                     GX5731_XTRIGGER_LINE_CONNECTED, &nStatus);
```

See Also

Gx5731GetExtTriggerToPxiTriggerLine, **Gx5731SetExtTriggerToPxiTriggerBus**, **Gx5731GetExtTriggerToPxiTriggerLine, Gx5731SetStrobeToPxiTriggerBusState**

Gx5731SetInternalClock

Purpose

Sets internal clock source and divider.

Syntax

Gx5731SetInternalClock (*nHandle, nClock, nSource, dwDivider, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5731 board.
nClock	SHORT	Clock number: 0. GX5731_CLOCK0: Internal Clock 0 1. GX5731_CLOCK1: Internal Clock 1
nSource	SHORT	Internal Clock 0 and Internal Clock 1 sources are listed in comment.
dwDivider	DWORD	Divider value: 1-0x8000000
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Internal Clock 0 sources are:

0 GX5731_INTERNAL_CLOCK0_T0_CLOCK_IN0: Internal Clock 0 connected to Clock Input 1.

1 GX5731_INTERNAL_CLOCK0_T0_PCI_CLOCK_33MHZ: Internal Clock 0 connected to 33MHz PCI Clock.

2 GX5731_INTERNAL_CLOCK0_T0_PXI_CLOCK_10MHZ: Internal Clock 0 connected to 10MHz PXI Clock.

3 GX5731_INTERNAL_CLOCK0_T0_STAR_TRIGGER: Internal Clock 0 connected to the PXI Start Trigger.

4 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER0: Internal Clock 0 connected to the PXI External Trigger Line 0.

5 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER1: Internal Clock 0 connected to the PXI External Trigger Line 1.

6 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER2 Internal Clock 0 connected to the PXI External Trigger Line 2

7 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER3 Internal Clock 0 connected to the PXI External Trigger Line 3.

Internal Clock 1 sources are:

0 GX5731_INTERNAL_CLOCK0_T0_CLOCK_IN1: Internal Clock 1 connected to Clock Input 0.

1 GX5731_INTERNAL_CLOCK0_T0_PCI_CLOCK_33MHZ: Internal Clock 1 connected to 33MHz PCI Clock.

2 GX5731_INTERNAL_CLOCK0_T0_PXI_CLOCK_10MHZ: Internal Clock 1 connected to 10MHz PXI Clock.

3 GX5731_INTERNAL_CLOCK0_T0_STAR_TRIGGER: Internal Clock 1 connected to the PXI Start Trigger.

4 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER4: Internal Clock 1 connected to the PXI External Trigger Line 4.

5 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER5: Internal Clock 1 connected to the PXI External Trigger Line 5.

6 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER6 Internal Clock 1 connected to the PXI External Trigger Line 6.

7 GX5731_INTERNAL_CLOCK0_T0_EXT_TRIGGER7 Internal Clock 1 connected to the PXI External Trigger Line 7.

Use the **Gx5731GetInternalClock** to retrieve the internal clock settings.

Example

The following example sets the internal clock 1 to 10Mhz and divider to 20000:

```
Gx5731SetInternalClock (nHandle, 0, 1, 20000, &nStatus);
```

See Also

Gx5731GetInternalClock

Gx5731SetInternalClockEnable

Purpose

Sets the specified internal clock enable mode.

Syntax

Gx5731SetInternalClockEnable (*nHandle, nClock, nEnable, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5731 board.
nClock	SHORT	Clock number: 0. GX5731_CLOCK0: Internal Clock 0 1. GX5731_CLOCK1: Internal Clock 1
nEnable	SHORT	Returned clock enable mode:: 0. GX5731_DISABLE_INTERNAL_CLOCK: Disable the specified clock. 1. GX5731_ENABLE_INTERNAL_CLOCK: Enable the specified clock.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example enable the internal clock 0:

```
Gx5731SetInternalClockEnable (nHandle, GX5731_CLOCK0, GX5731_ENABLE_INTERNAL_CLOCK, &nStatus);
```

See Also

Gx5731GetInternalClockEnable, Gx5731GetInternalClock

Gx5731SetPort

Purpose

Writes whole data to a port.

Syntax

Gx5731SetPort (*nHandle, nPort, dwValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port number: 0-6.
dwValue	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.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Use the **Gx5731GetPort** to read data from the port.

Example

The following example writes 0xFF00FF to port 1:

```
Gx5731SetPort (nHandle, 1, 0xFF00FF, &nStatus);
```

See Also

Gx5731GetPort, **Gx5731SetPortBit**, **Gx5731SetPortByte**, **Gx5731SetPortWord, Gx5731SetPortDirection**

Gx5731SetPortBit

Purpose

Writes a specific bit of data to a port channel.

Syntax

Gx5731SetPortBit (*nHandle, nPort, nBit, ucValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port number: 0-6.
nBit	SHORT	Port channel number: 0 to 31.
ucValue	BYTE	Channel value: 0 for low state and 1 for hi state.
pnStatus	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 **Gx5731GetPortBit** to read data from a port bit.

Example

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

```
Gx5731GetPort (nHandle, 1, 30, 1, &nStatus);
```

See Also

Gx5731GetPortBit, **Gx5731SetPortByte**, **Gx5731SetPortWord**, **Gx5731SetPort**

Gx5731SetPortByte

Purpose

Writes a specific byte of data to a port group.

Syntax

Gx5731SetPortByte (*nHandle, nPort, nByte, ucValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port number: 0-6.
nByte	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte.
ucValue	BYTE	Value to set 0-255. Each bit represents the specific channel state bit 0 for lowest channel number.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Only byte set to output direction will be written.

Use the **Gx5731GetPortByte** 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:

```
Gx5731GetPortByte (nHandle, 1, 2, 0x5, &nStatus);
```

See Also

Gx5731GetPortByte, Gx5731SetPortBit, **Gx5731SetPortWord**, **Gx5731SetPort**

Gx5731SetPortByteDirection

Purpose

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

Syntax

Gx5731SetPortByteDirection (*nHandle, nPort, nByte, bInOut, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port number: 0-6.
nByte	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte.
bInOut	BOOL	Byte direction: 0. (FALSE) – Input. 1. (TRUE) – Output.
pnStatus	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:

```
Gx5731SetPortByteDirection (nHandle, 1, 3, TRUE, nStatus);
```

See Also

Gx5731GetPortByteDirection, Gx5731SetPortDirection

Gx5731SetPortDirection

Purpose

Sets a direction for a port groups.

Syntax

Gx5731SetPortDirection (*nHandle*, *nPort*, *nDirection*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port number: 0-6.
nDirection	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 3. A bit is set to hi '1' for Input and low '0' for output.
pnStatus	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:

```
Gx5731GetPortByteDirection (nHandle, 1, 6, nStatus);
```

See Also

Gx5731GetPortDirection, Gx5731SetPortByteDirection

Gx5731SetPortWord

Purpose

Writes a specific word of data to a port.

Syntax

Gx5731SetPortWord (*nHandle, nPort, nWord, wValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5731 board.
nPort	SHORT	Digital I/O port number: 0-6.
nWord	SHORT	Port word number: 0-1. Where 0 is for the port low order word (bytes 0 and 1) and 1 for the high order word (bytes 2 and 3).
wValue	WORD	Port word value: 0 to 65,535 (0-0xFFFF). Where bit 0 corresponds to channel 0 and bit 15 to channel 15.
pnStatus	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 0x5 (bit 16 and 19 hi and bit 17, 18, 20-23 to low state) to port 1, byte 2 value:

```
Gx5731GetPortByte (nHandle, 1, 2, 0x5, &nStatus);
```

See Also

Gx5731GetPortWord, **Gx5731SetPortBit**, **Gx5731SetPortByte**, **Gx5731SetPort**

Gx5731SetStrobeToPxiTriggerBusState

Purpose

Sets all Strobe to Pxi Trigger lines Output Mode.

Syntax

Gx5731SetStrobeToPxiTriggerBusState (*nHandle, nBusState, pnStatus*)

Parameters

Name	Type	Comments
Nhandle	SHORT	Handle to a GX5731 board.
nBusState	SHORT	Sets all eight Strobe to Pxi Trigger Bus lines Output mode, 8 bits integer where each bit represents a Pxi Trigger line. Bit 0 for line 0 and bit 7 for line 7. A high ('1') means the corresponding Pxi Trigger line is enable, low ('0') means the corresponding Pxi Trigger line is disabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetStrobeToPxiTriggerBusState** to set a specified Pxi Trigger Line output.

Example

The following example sts all eight Strobe to Pxi Trigger Bus lines output mode to 0x5:

```
SHORT nStatus;

Gx5731SetStrobeToPxiTriggerBusState (nHandle, 0x5, &nStatus);
```

See Also

Gx5731GetStrobeToPxiTriggerBusState, Gx5731SetStrobeToPxiTriggerLineState, Gx5731GetExtTriggerToPxiTriggerLine,

Gx5731SetStrobeToPxiTriggerLineState

Purpose

Sets the specified Strobe to Pxi Trigger Line output mode.

Syntax

Gx5731SetStrobeToPxiTriggerLineState (*nHandle, nLine, bMode, pnStatus*)

Parameters

Name	Type	Comments
NHandle	SHORT	Handle to a GX5731 board.
Nline	SHORT	Pxi Trigger Line numbers are: 0 GX5731_XTRIGGER_LINE0: Line 0 1 GX5731_XTRIGGER_LINE1: Line 1 2 GX5731_XTRIGGER_LINE2: Line 2 3 GX5731_XTRIGGER_LINE3: Line 3 4 GX5731_XTRIGGER_LINE4: Line 4 5 GX5731_XTRIGGER_LINE5: Line 5 6 GX5731_XTRIGGER_LINE6: Line 6 7 GX5731_XTRIGGER_LINE7: Line 7
bMode	BOOL	Pxi trigger line mode are: 0 GX5731_XTRIGGER_LINE_DISCONNECTED: Pxi trigger line is disconnected. 1 GX5731_XTRIGGER_LINE_CONNECTED: Pxi trigger line is Connected
PnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5731SetStrobeToPxiTriggerBusState** function to set all Pxi-Trigger lines at once.

Example

The following example set Pxi Trigger line 1 mode:

```
SHORT nStatus;

Gx5731SetStrobeToPxiTriggerLineState(nHandle, GX5731_XTRIGGER_LINE1, GX5731_XTRIGGER_LINE_CONNECTED,
                                     &nStatus);
```

See Also

Gx5731GetStrobeToPxiTriggerBusState, Gx5731SetStrobeToPxiTriggerLineState, Gx5731GetExtTriggerToPxiTriggerLine, Gx5731GetExtTriggerToPxiTriggerBus

GxPioGetDriverSummary

Purpose

Returns the driver name and version.

Syntax

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

Parameters

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

Comments

The returned string is: "GXPIO Driver for GX5731 and GX5732. Version 2.0, Copyright © Marvin Test Solutions 2003.".

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

Comments

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

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

-53 Invalid internal strobe source, source can be between 0-7

-54 Invalid strobe source, strobe source can be 0-Internal or 1-External

-55 Invalid strobe polarity, strobe polarity can be 0-Positive Edge or 1-Negative Edge

-56 Invalid Buffer size, Buffer size can be between 1-100

-57 Invalid Group number

-58 Invalid direction

Execution error

-80 Unable to load Counter. Counter Port is not in Input

-81 Unable to empty the specified Module's buffer

-82 Testing the specified Module's buffer failed

-83 Failed to write data correctly to the specified Module's buffer

-84 Failed to read data correctly from the specified Module's buffer

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