GX5642

Bi-directional

Differential LVDS-TTL

I/O PXI Board

GXPIO Software

User's Guide

Last updated July 2, 2015

MARVIN TEST SOLUTIONS

Safety and Handling

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

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

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

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

Warranty

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

If You Need Help

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

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

Disclaimer

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 GX5642 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 GX5642 driver (referred in this manual **GXPIO**). Therefore, good understanding of programming development tools and languages may be necessary.

Manual Organization

The GX5642 manual is organized in the following manner:

Chapter	**Content**
Chapter 1 - Introduction	Introduces the GX5642 manual. Lists all the supported board and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX5642 features, board description, its architecture, specifications and the panel description and operation.
Chapter 3 –Installation and Connections	Provides instructions on how to install a GX5642 board and the GXPIO software.
Chapter 4 – Instrument Software Panel	Provides instruction how to open and use the instrument front panel application in order to view and control the instrument settings.
Chapter 5 – Programming the Board	Provides a list of the GXPIO software driver files, general purpose and generic driver functions, and programming methods. Discusses supported application development tools and programming examples.
Chapter 6 – Functions Reference	Provides a list of the GX5642 driver functions. Each function description provides syntax, parameters, and any special programming comments.

Conventions Used in this Manual

Symbol Convention	**Meaning**
⚡	Static Sensitive Electronic Devices. Handle Carefully.
STOP	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.
▼ TIP	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 GX5642 is a 3U PXI instrument card that consists of 64 Bi-Directional TTL-Differential I/O channels. Each channel has two ports: TTL and Differential. Each channel can be individually set to operate in one of two modes: Conversion or Static I/O. In Static I/O mode the GX5642 supports 128 individual digital inputs or outputs, 64 TTL and 64 Differential. In Conversion mode the GX5642 support 64 individual Differential channels. The GX5642 can be configured via jumper to operate in one of two modes: Independent or Software Controlled.

When the card is configured to Independent mode the PXI interface is disabled and all 64 channels operate in Conversion mode. In this mode each channel conversion operation is predefined using DIP-Switches. Each channel can be predefined to convert TTL to Differential or Differential to TTL. The channels conversion settings will be loaded automatically upon power up.

When the card is configured to Software Controlled mode each channel can be individually programmed to operate in Conversion mode or Static I/O mode. In Conversion mode each channel can be programmed to convert TTL to Differential or Differential LVDS to TTL. In Static I/O mode, each channel's port can be programmed as follows:

- TTL port input, Differential port input.
- TTL port output, Differential port input.
- TTL port input, Differential port output.
- TTL port output, Differential port output.

In Software Controlled mode each channel's TTL and Differential outputs can be individually enabled or disabled.

Features

The GX5642 is a 3U PXI instrument card with two operation modes:

Independent mode features (JP2 Installed)

- Total of 64 individual channels each capable of converting TTL to Differential or Differential to TTL.
- Each channel conversion operation is predefined using DIP-Switches. Converting TTL to Differential or Differential to TTL.
- Channels conversion settings automatically loaded on power up.
- The board is disconnected from the PXI interface and operated without any interaction with the host computer.
- J1 TTL connector has 40 TTL channels assigned to it and J2 TTL connector has 24 TTL channels. The Differential connectors, J3 and J4, have 32 channels each.
- A 100-Ohm terminator for each Differential I/O terminals.

Software Controlled mode features (JP2 Not Installed)

- Each channel can be programmed to operate in Conversion or Static I/O mode.
- In Conversion mode the board has 64 individual channels.
- In Conversion mode each channel can be programmed to convert Differential to TTL or TTL to Differential.
- In Static I/O mode the board has 64 individual channels each with TTL and Differential ports for a total of 128 individual Digital I/O ports.

- In Static I/O mode each channel's port direction can be individually set to output or input, e.g. Channel 1 Differential port as output and Channel 1 TTL port as input.
- TTL and Differential input signals can be monitored in all modes of operations.
- Channels predefined DIP-Switch conversion operation can be loaded at any time overriding current settings (predefined defaults).
- Each channel's TTL and Differential output port can be individually enabled/disabled through software control.
- On power up all 64 channels' outputs are disabled.
- A 100-Ohm terminator for each Differential I/O terminals.

Applications

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

Board Description

The GX5642 is a 3U PXI instrument card that consists of 64 bi-directional TTL I/O channels and 64 bi-directional I/O channels. The GX5642 can be configured via jumper (JP2) to operate in one of two modes: Independent or Software Controlled. The board has four 68-pin VHDCI connectors two for TTL and two for Differential channels.

Independent mode (JP2 installed)

When the card is configured to Independent mode the board is disconnected from the PXI interface, operated without any interaction with the host computer, and all 64 channels operate in Conversion mode only. Each channel conversion operation is predefined using DIP-Switches specifying conversion of TTL to Differential or Differential to TTL. All 64 channels conversion settings are automatically loaded upon power up. In this mode the J1 TTL connector has 40 TTL channels assigned to it and J2 TTL connector has 24 TTL channels. The Differential connectors, J3 and J4, have 32 channels each.

Software Controlled mode (JP2 not installed)

When the card is configured to Software Controlled mode each channel can be individually set to operate in Conversion or Static I/O mode. In Conversion mode each channel can be programmed to convert Differential to TTL or TTL to Differential. The board has 64 DIP-Switches, one for each channel to enable a predefine conversion mode per channel. The channels' predefined DIP-Switch conversion modes can be loaded at any time overriding current settings (predefined defaults).

In Static I/O mode the GX5642 support 128 individual digital inputs or outputs, 64 TTL and 64 Differential. In this mode each channel TTL port can be set as input TTL port or output TTL port. In this mode the channel Differential port can be set as input Differential port or output Differential port.

In Software Controlled mode each channel's TTL and Differential outputs can be individually enabled or disabled.

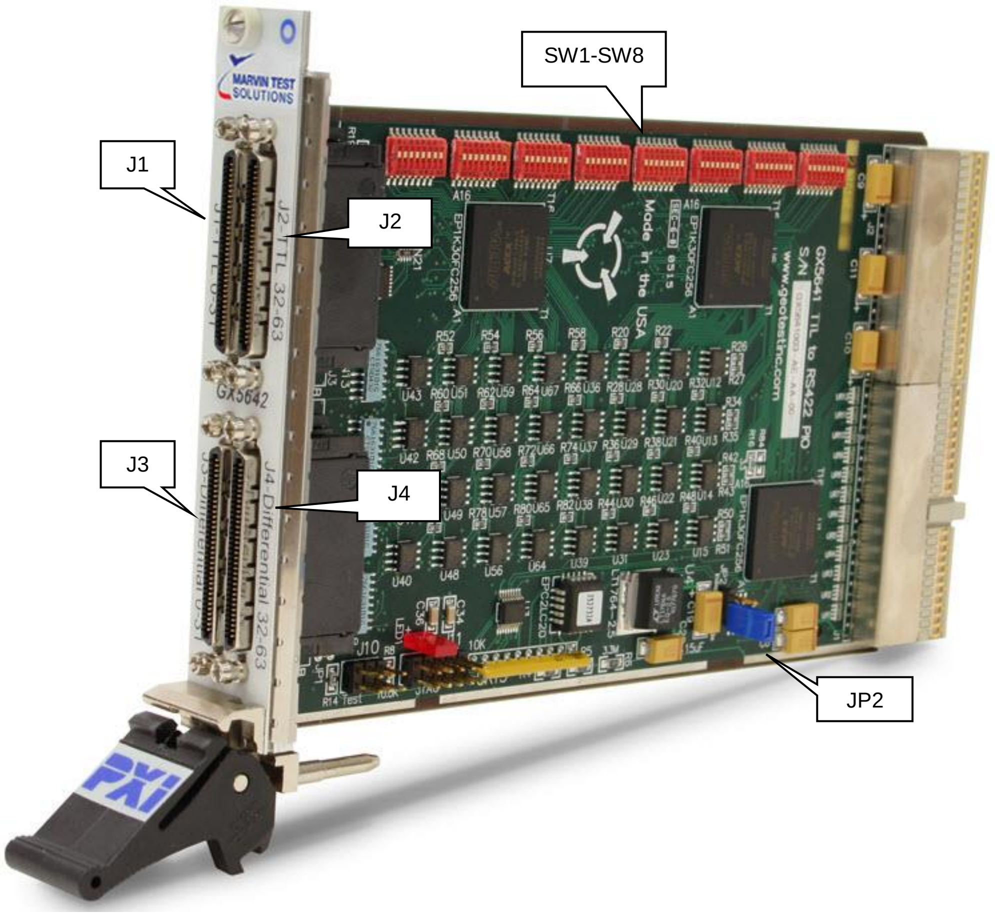


Figure 2-1: GX5642 Board Side View

J1, J2	TTL I/O Connector Channels
J3, J4	Differential I/O Connector Channels
JP2	Jumper installed sets board to Independent mode, without jumper board is in Software Controlled mode.
SW1-SW8	Switches to set conversion direction in independent mode, In software-controlled mode the switches states can be loaded by software command. Switch of SW1 sets Channel 0 direction; Switch 2 of SW1 sets Channel 1 and so on until switch 8 of SW8 who sets channel 63. Switch in ON position set the channel’s direction Differential to TTL. Switch in OFF position set the channel’s direction Differential to TTL

Architecture: Software Controlled mode (JP2 not installed)

Single I/O channel

The GX5642 provides 128 digital Inputs or Outputs with direction control. Figure 2-3 shows a typical I/O channel block diagram when the board is in Software Controlled mode:

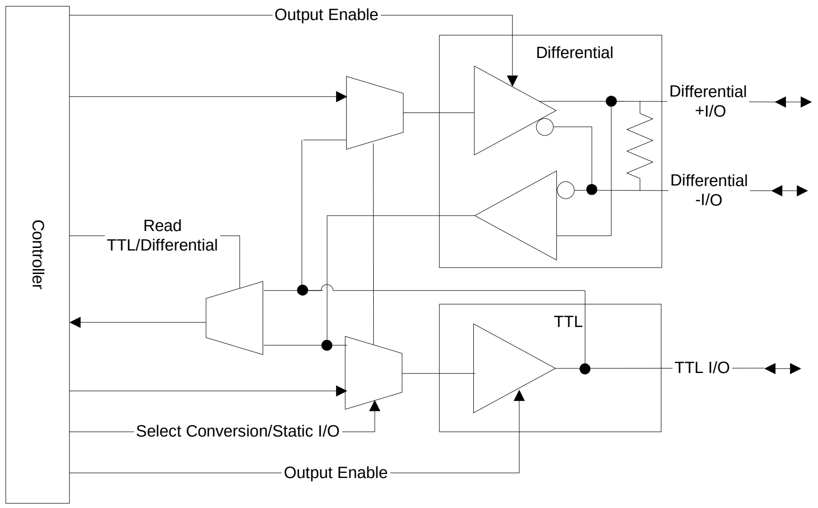


Figure 2-2: Typical I/O Channel

Under software controlled each channel can be programmed to operate in Conversion mode or Static I/O mode. All channels have two ports: TTL and Differential and each port can be programmed to be input or output depending on the channel's programmed operation mode. Each channel's output enable is controlled though software in both Conversion and Static I/O modes. Each channel's port can be read back in both Conversion and Static I/O modes.

When a channel is programmed to Conversion mode the channel's conversion direction can be programmed to be Differential to TTL (Differential port is input and TTL port is output) or TTL to Differential (TTL port is input and Differential port is output).

When a channel is programmed to Static I/O mode each of the channel's ports can be set independently from the other, i.e. the TTL port operates separately from the Differential port. In this mode the channel's TTL port can be set as input TTL or output TTL. And the channel's Differential port can be set as input Differential or output Differential. The Differential I/O terminals are terminated with a built-in 100-Ohm resistor.

Architecture: Independent mode (JP2 installed)

Single I/O channel

The GX5642 provides 128 digital Inputs or Outputs and direction. Figure 2-3 shows a typical I/O channel block diagram when the board is in Independent mode:

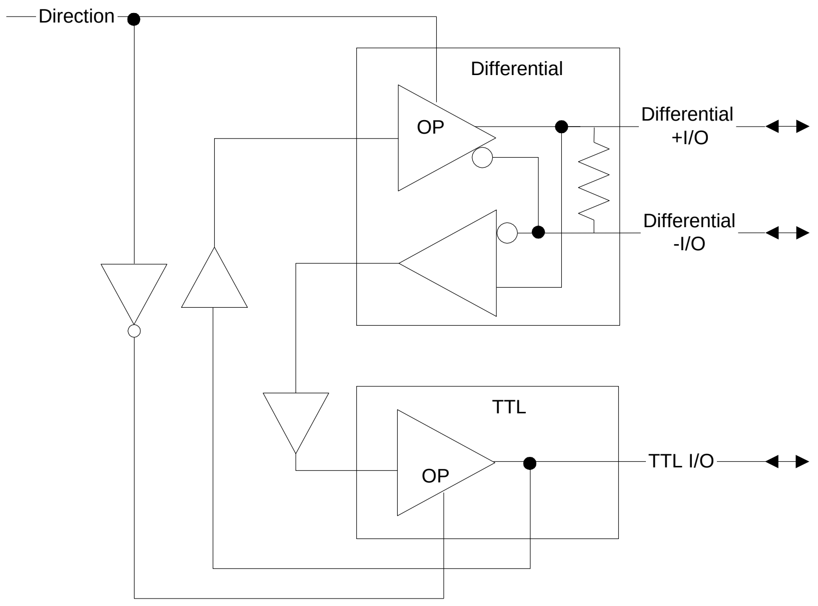


Figure 2-3: Typical Single I/O Channel

When the board is in Independent mode each channel can convert Differential to TTL (Differential port is input and TTL port is output) or TTL to Differential (TTL port is input and Differential port is output). Each channel has a dedicated DIP-Switch that sets the conversion direction for the channel upon power up. The user can preset each channel conversion direction by simply setting the specified channel’s DIP Switch to On position: Differential to TTL or Off position: TTL to Differential. The Differential I/O terminals are terminated with a built-in 100-Ohm resistor.

DIP-Switches Settings

There are eight designated surface mount DIP-Switches, SW1 through SW8, located at the top of the GX5642 board. Each DIP-Switch has eight switches for a total of 64 individual DIP-Switches. When a switch is in the ON position the related channel's direction is Differential to TTL. When a switch is in the OFF position the related channel's direction is TTL to Differential.

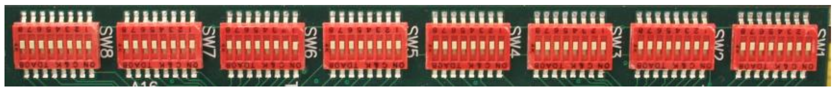


Figure 2-4: DIP-Switches SW1 through SW8

Each DIP-Switch is connected to a channel according to the following tables:

Setting DIP-Switches SW1 and SW2

DIP-Switch SW1				DIP-Switch SW2			
Ch #	**Switch #**	**Position**	**Conversion**	**Ch #**	**Switch #**	**Position**	**Conversion**
0	1	ON	DIFF to TTL	8	1	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
1	2	ON	DIFF to TTL	9	2	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
2	3	ON	DIFF to TTL	10	3	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
3	4	ON	DIFF to TTL	11	4	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
4	5	ON	DIFF to TTL	12	5	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
5	6	ON	DIFF to TTL	13	6	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
6	7	ON	DIFF to TTL	14	7	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
7	8	ON	DIFF to TTL	15	8	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF

Table 2-1: Setting DIP-Switches SW1 and SW2

Setting DIP-Switches SW3 and SW4

DIP-Switch SW3				DIP-Switch SW4			
Ch #	**Switch #**	**Position**	**Conversion**	**Ch #**	**Switch #**	**Position**	**Conversion**
16	1	ON	DIFF to TTL	24	1	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
17	2	ON	DIFF to TTL	25	2	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
18	3	ON	DIFF to TTL	26	3	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
19	4	ON	DIFF to TTL	27	4	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
20	5	ON	DIFF to TTL	28	5	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
21	6	ON	DIFF to TTL	29	6	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
22	7	ON	DIFF to TTL	30	7	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
23	8	ON	DIFF to TTL	31	8	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF

Table 2-2: Setting DIP-Switches SW3 and SW4

Setting DIP-Switches SW5 and SW6

DIP-Switch SW5				**DIP-Switch SW6**			
Ch #	**Switch #**	**Position**	**Conversion**	**Ch #**	**Switch #**	**Position**	**Conversion**
32	1	ON	DIFF to TTL	40	1	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
33	2	ON	DIFF to TTL	41	2	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
34	3	ON	DIFF to TTL	42	3	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
35	4	ON	DIFF to TTL	43	4	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
36	5	ON	DIFF to TTL	44	5	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
37	6	ON	DIFF to TTL	45	6	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
38	7	ON	DIFF to TTL	46	7	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
38	8	ON	DIFF to TTL	47	8	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF

Table 2-3: Setting DIP-Switches SW5 and SW6

Setting DIP-Switches SW7 and SW8

DIP-Switch SW7				DIP-Switch SW8			
Ch #	**Switch #**	**Position**	**Conversion**	**Ch #**	**Switch #**	**Position**	**Conversion**
48	1	ON	DIFF to TTL	56	1	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
49	2	ON	DIFF to TTL	57	2	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
50	3	ON	DIFF to TTL	58	3	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
51	4	ON	DIFF to TTL	59	4	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
52	5	ON	DIFF to TTL	60	5	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
53	6	ON	DIFF to TTL	61	6	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
54	7	ON	DIFF to TTL	62	7	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF
55	8	ON	DIFF to TTL	63	8	ON	DIFF to TTL
		OFF	TTL to DIFF			OFF	TTL to DIFF

Table 2-4: Setting DIP-Switches SW7 and SW8

Specifications

The following table outlines the specifications of the GX5642.

Channel Specifications

Data Conversion rate	150Mbps		
TTL I/O Levels			
	Min. (V)	Typ. (V)	Max. (V)
Input Low	0.0		0.8
Input High	1.8		5.0
Output Low	0.0	0.2	0.5
Output High	2.4	3.0	5.0
LVDS I/O Levels			
	Min. (V)		Max. (V)
Positive-going differential input voltage threshold	-0.2		0.2
Negative-going differential input voltage threshold		0.07	
Driver Differential VOUT	2.0	3.0	5.0
Driver Common-Mode VOUT			3.0

Power Requirements

3.3 VDC Current	0.50A Max
5 VDC Current	1.4A Typical, 2.6A Max
User 5 VDC Current (J3 pin 67 and J4 pin 67)	1A Max

Environmental

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

Physical

Size	3U PXI
Weight	18 oz.

Chapter 3 - Installation and Connections

Getting Started

This section includes general hardware installation procedures for the GX5642 board and installation instructions for the GX5642 (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
GX5642 Driver Installation	This Chapter
Programming	Chapter 5
GX5642 Function Reference	Chapter 6

Packing List

All GX5642 boards have the same basic packing list, which includes:

1. GX5642 Board
2. GX5642 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 GX5642 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 3U PXI bus slot.

Interfaces and Accessories

The following accessories are available from Marvin Test Solutions for GX5642 switching board.

Part / Model Number	**Description**
GT95015	Connector Interface SCSI to 100 Mil Grid Differential
GT95021	2' 68-Pin shielded cable
GT95022	3' 68-Pin shielded cable
GT95028	10' 68-Pin shielded cable
GT95031	6' 68-Pin shielded cable

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. 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.
4. 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 GX5642 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.

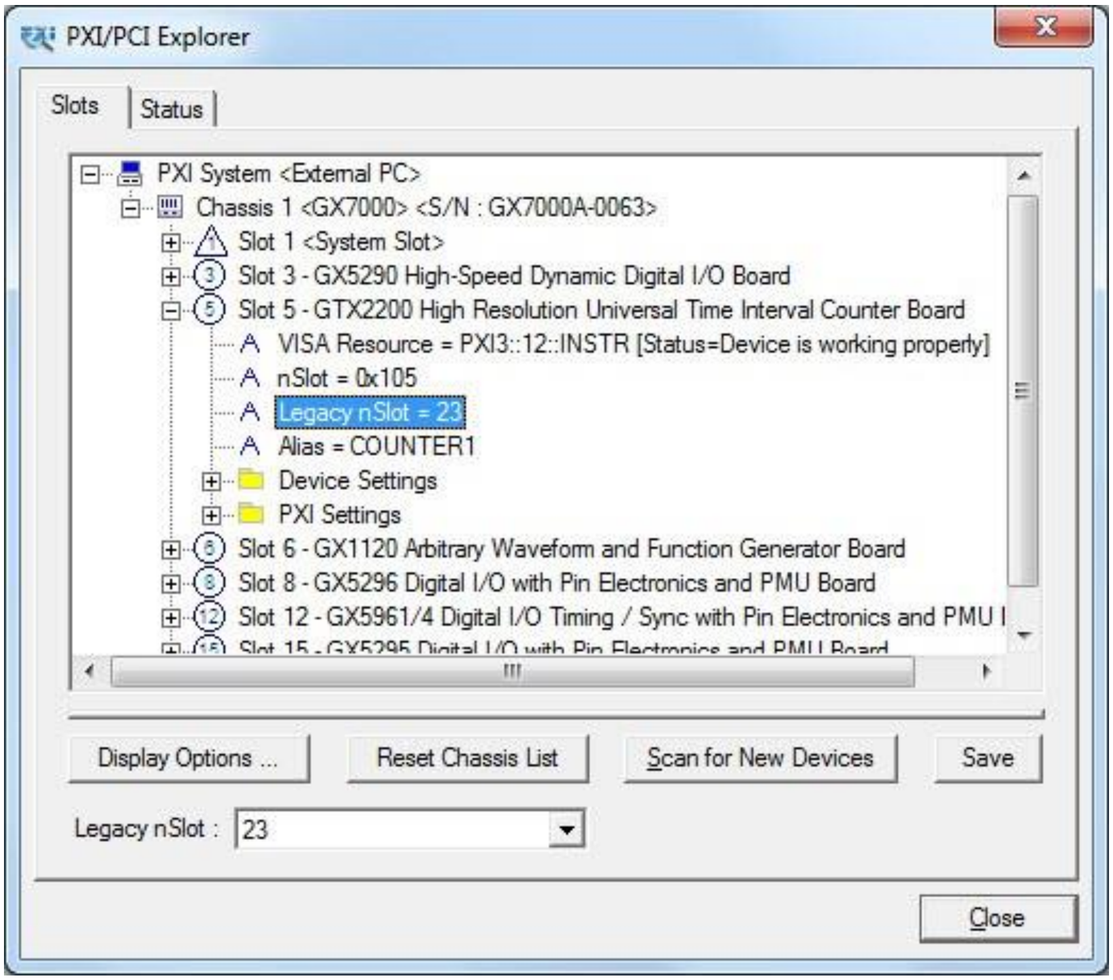
During the installation, `GXPIO` Setup creates and copies files to the following folders:

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 **Gx5642Initialize** and **Gx5642InitializeVisa** 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).



```
PXI System <External PC>
  Chassis 1 <GX7000> <S/N : GX7000A-0063>
    Slot 1 <System Slot>
    Slot 3 - GX5290 High-Speed Dynamic Digital I/O Board
    Slot 5 - GTX2200 High Resolution Universal Time Interval Counter Board
      VISA Resource = PXI3::12::INSTR [Status=Device is working properly]
      nSlot = 0x105
      Legacy nSlot = 23
      Alias = COUNTER1
      Device Settings
      PXI Settings
    Slot 6 - GX1120 Arbitrary Waveform and Function Generator Board
    Slot 8 - GX5296 Digital I/O with Pin Electronics and PMU Board
    Slot 12 - GX5961/4 Digital I/O Timing / Sync with Pin Electronics and PMU I
    Slot 15 - GX5295 Digital I/O with Pin Electronics and PMU Board
```

Legacy nSlot : 23

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 GX5642 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).

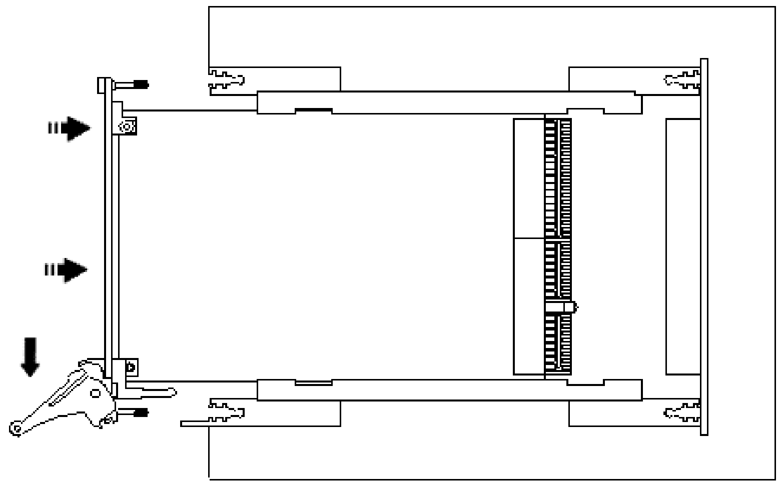


Figure 3-2: Ejector handles position during module insertion

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

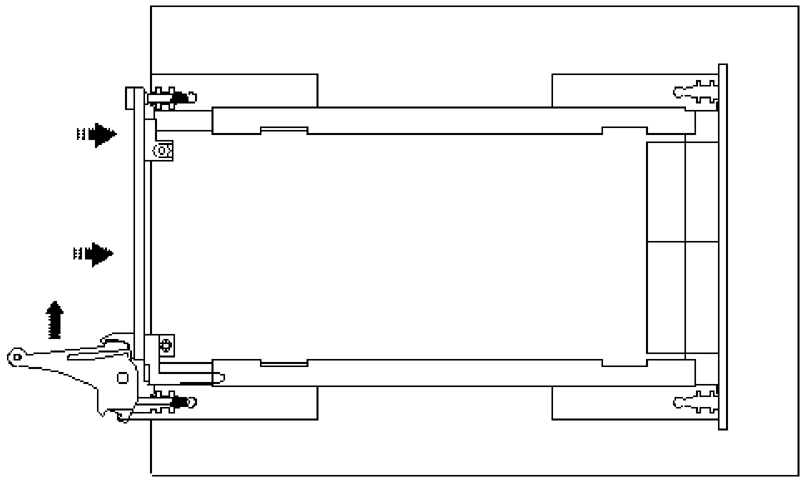


Figure 3-3: Ejector handles position after module insertion

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 C:\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 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.
- GX5642.DRV - ATEasy driver File for GX5642.GXPIO Virtual Panel Program

GXPIO On-line Help and Manual

GXPIO.CHM – On-line version of the GX5642 User's Guide. The help file is provided in a Windows Compiled HTML help file (.CHM). The file contains information about the GX5642 board, programming reference and panel operation.

GX5642.PDF – On line, printable version of the GX5642 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):

- Gx5642.prj - example project
- GX55642.sys - example system
- GX5642.prg - example program

Connectors and Jumpers

Error! Reference source not found. shows the available GX5642 board connectors and jumpers followed by their description:

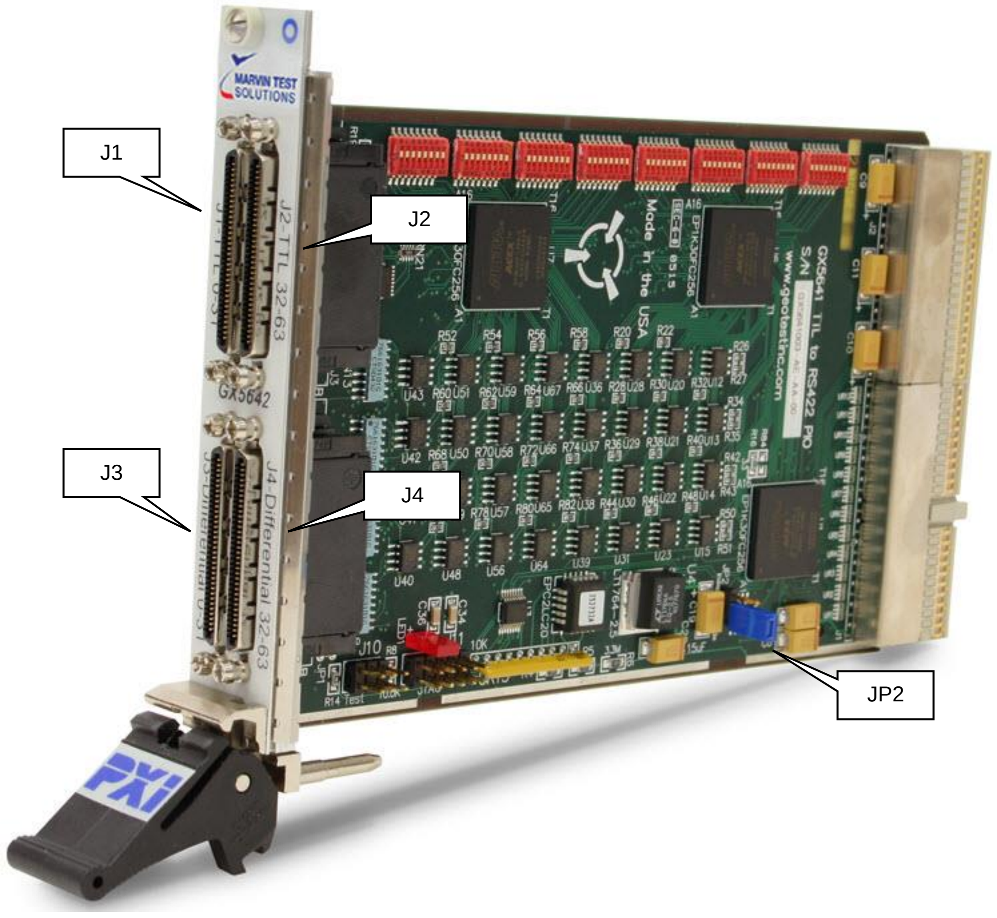


Figure 3-4: GX5642 Connectors and Jumpers

Connector/Jumpers	Description
J1, J2	TTL I/O Connector Channels
J3, J4	Differential I/O Connector Channels
JP2	Installed Jumper sets the board to Independent mode, without jumper the board is in Software Controlled mode.

JP2 – Board Operation Mode Jumper

JP2 jumper determines the Board Operation Mode. When jumper is not installed (default), the board operates in Software Controlled mode. When the jumper is installed, the board operates Independent mode.

Connectors – Default mode (J1 pins 27 and 28 Logic low)

Connections to the GX5642 may be made with 68-pin VHDCI male plug connector. Shielded cables with matching connectors are available from Marvin Test Solutions.

J1 – TTL I/O Connector

The following are connector pin assignments for J1 TTL I/O Connector with low level in Inputs pins 27 and 28:

#	Signal	#	Signal	#	Signal	#	Signal
1	GND	18	GND	35	TIO0	52	TIO17
2	GND	19	GND	36	TIO1	53	TIO18
3	GND	20	GND	37	TIO2	54	TIO19
4	GND	21	GND	38	TIO3	55	TIO20
5	GND	22	GND	39	TIO4	56	TIO21
6	GND	23	GND	40	TIO5	57	TIO22
7	GND	24	GND	41	TIO6	58	TIO23
8	GND	25	GND	42	TIO7	59	TIO24
9	GND	26	GND	43	TIO8	60	TIO25
10	GND	27	5V Sense	44	TIO9	61	TIO26
11	GND	28	5V Sense	45	TIO10	62	TIO27
12	GND	29	R	46	TIO11	63	TIO28
13	GND	30	R	47	TIO12	64	TIO29
14	GND	31	R	48	TIO13	65	TIO30
15	GND	32	R	49	TIO14	66	TIO31
16	GND	33	R	50	TIO15	67	R
17	GND	34	R	51	TIO16	68	R

Table 3-1: J1: TTL I/O Connector with pins 27 and 28 low

GND	Ground
TIO0 to TIO31	Input Output TTL Data Lines
5V Sense	Input Sense detecting if 5V (Logic High) is connected
R	Reserved

J2 – TTL I/O Connector

The following are connector pin assignments for J2 TTL I/O Connector with low level in J1 inputs pins 27 and 28:

#	Signal	#	Signal	#	Signal	#	Signal
1	GND	18	GND	35	TIO32	52	TIO49
2	GND	19	GND	36	TIO33	53	TIO50
3	GND	20	GND	37	TIO34	54	TIO51
4	GND	21	GND	38	TIO35	55	TIO52
5	GND	22	GND	39	TIO36	56	TIO53
6	GND	23	GND	40	TIO37	57	TIO54
7	GND	24	GND	41	TIO38	58	TIO55
8	GND	25	GND	42	TIO39	59	TIO56
9	GND	26	GND	43	TIO40	60	TIO57
10	GND	27	5V Sense	44	TIO41	61	TIO58
11	GND	28	5V Sense	45	TIO42	62	TIO59
12	GND	29	R	46	TIO43	63	TIO60
13	GND	30	R	47	TIO44	64	TIO61
14	GND	31	R	48	TIO45	65	TIO62
15	GND	32	R	49	TIO46	66	TIO63
16	GND	33	R	50	TIO47	67	R
17	GND	34	R	51	TIO48	68	R

Table 3-2: J2: TTL I/O Connector with J1 pins 27 and 28 low

GND	Ground
TIO32 to TIO63	Input Output TTL Data Lines
5V Sense	Input Sense detecting if 5V (Logic High) is connected
R	Reserved

J3 – Differential I/O Connector Channels 0-31

The following are connector pin assignments for J3 Differential I/O Channels 0-31 Connector:

#	Signal	#	Signal	#	Signal	#	Signal
1	DIO0+	18	DIO17+	35	DIO0-	52	DIO17-
2	DIO1+	19	DIO18+	36	DIO1-	53	DIO18-
3	DIO2+	20	DIO19+	37	DIO2-	54	DIO19-
4	DIO3+	21	DIO20+	38	DIO3-	55	DIO20-
5	DIO4+	22	DIO21+	39	DIO4-	56	DIO21-
6	DIO5+	23	DIO22+	40	DIO5-	57	DIO22-
7	DIO6+	24	DIO23+	41	DIO6-	58	DIO23-
8	DIO7+	25	DIO24+	42	DIO7-	59	DIO24-
9	DIO8+	26	DIO25+	43	DIO8-	60	DIO25-
10	DIO9+	27	DIO26+	44	DIO9-	61	DIO26-
11	DIO10+	28	DIO27+	45	DIO10-	62	DIO27-
12	DIO11+	29	DIO28+	46	DIO11-	63	DIO28-
13	DIO12+	30	DIO29+	47	DIO12-	64	DIO29-
14	DIO13+	31	DIO30+	48	DIO13-	65	DIO30-
15	DIO14+	32	DIO31+	49	DIO14-	66	DIO31-
16	DIO15+	33	5V	50	DIO15-	67	5V
17	DIO16+	34	GND	51	DIO16-	68	GND

Table 3-3: J3: Differential I/O Connector Channels 0-31

DIO0+ to DIO31+	Input/Output Differential Data Lines High
DIO0- to DIO31-	Input/Output Differential Data Lines Low
5V	5 Volt Power
GND	Ground

J4 – Differential I/O Connector Channels 32-63

The following are connector pin assignments for J4 Differential I/O Channels 32-63 Connector:

#	Signal	#	Signal	#	Signal	#	Signal
1	DIO32+	18	DIO49+	35	DIO32-	52	DIO49-
2	DIO33+	19	DIO50+	36	DIO33-	53	DIO50-
3	DIO34+	20	DIO51+	37	DIO34-	54	DIO51-
4	DIO35+	21	DIO52+	38	DIO35-	55	DIO52-
5	DIO36+	22	DIO53+	39	DIO36-	56	DIO53-
6	DIO37+	23	DIO54+	40	DIO37-	57	DIO54-
7	DIO38+	24	DIO55+	41	DIO38-	58	DIO55-
8	DIO39+	25	DIO56+	42	DIO39-	59	DIO56-
9	DIO40+	26	DIO57+	43	DIO40-	60	DIO57-
10	DIO41+	27	DIO58+	44	DIO41-	61	DIO58-
11	DIO42+	28	DIO59+	45	DIO42-	62	DIO59-
12	DIO43+	29	DIO60+	46	DIO43-	63	DIO60-
13	DIO44+	30	DIO61+	47	DIO44-	64	DIO61-
14	DIO45+	31	DIO62+	48	DIO45-	65	DIO62-
15	DIO46+	32	DIO63+	49	DIO46-	66	DIO63-
16	DIO47+	33	5V	50	DIO47-	67	5V
17	DIO48+	34	GND	51	DIO48-	68	GND

Table 3-4: J4: Differential I/O Connector Channels 32-63

DIO32+ to DIO63+	Input/Output Differential Data Lines High
DIO32- to DIO63-	Input/Output Differential Data Lines Low
5V	5 Volt Power
GND	Ground

Connectors – National Instruments Compatibility mode (J1 pins 27 and 28 Logic High)

Whenever J1 pins 27 and 28 are set to Logic High (5V Sense), J1 and J2 connectors signal assignments change. In this mode J1 connector will have an additional eight TTL I/O channels for a total of 40 channels. The additional eight channels are channels 32-39 of J2 TTL I/O connector. Those channels should not be used on J2.

J1 – TTL I/O Connector

The following are connector pin assignments for J1 TTL I/O Connector with high level in inputs pins 27 and 28:

#	Signal	#	Signal	#	Signal	#	Signal
1	GND	18	GND	35	TIO0	52	TIO17
2	GND	19	GND	36	TIO1	53	TIO18
3	GND	20	GND	37	TIO2	54	TIO19
4	GND	21	GND	38	TIO3	55	TIO20
5	GND	22	GND	39	TIO4	56	TIO21
6	GND	23	GND	40	TIO5	57	TIO22
7	GND	24	GND	41	TIO6	58	TIO23
8	GND	25	GND	42	TIO7	59	TIO24
9	GND	26	GND	43	TIO8	60	TIO25
10	GND	27	5V Sense	44	TIO9	61	TIO26
11	GND	28	5V Sense	45	TIO10	62	TIO27
12	GND	29	TIO28	46	TIO11	63	TIO29
13	GND	30	TIO30	47	TIO12	64	TIO31
14	GND	31	TIO32	48	TIO13	65	TIO33
15	GND	32	TIO34	49	TIO14	66	TIO35
16	GND	33	TIO36	50	TIO15	67	TIO37
17	GND	34	TIO38	51	TIO16	68	TIO39

Table 3-5: J1: TTL I/O Connector with pins 27 and 28 high

GND	Ground
TIO0 to TIO39	Input Output TTL Data Lines
5V Sense	Input Sense detecting if 5V (Logic High) is connected

J2 – TTL I/O Connector

The following are connector pin assignments for J2 TTL I/O Connector with high level in J1 inputs pins 27 and 28:

#	Signal	#	Signal	#	Signal	#	Signal
1	GND	18	GND	35	R	52	TIO49
2	GND	19	GND	36	R	53	TIO50
3	GND	20	GND	37	R	54	TIO51
4	GND	21	GND	38	R	55	TIO52
5	GND	22	GND	39	R	56	TIO53
6	GND	23	GND	40	R	57	TIO54
7	GND	24	GND	41	R	58	TIO55
8	GND	25	GND	42	R	59	TIO56
9	GND	26	GND	43	TIO40	60	TIO57
10	GND	27	R	44	TIO41	61	TIO58
11	GND	28	R	45	TIO42	62	TIO59
12	GND	29	R	46	TIO43	63	TIO60
13	GND	30	R	47	TIO44	64	TIO61
14	GND	31	R	48	TIO45	65	TIO62
15	GND	32	R	49	TIO46	66	TIO63
16	GND	33	R	50	TIO47	67	R
17	GND	34	R	51	TIO48	68	R

Table 3-6: J2: TTL I/O Connector with J1 pins 27 and 28 high

GND	Ground
TIO40 to TIO63	Input Output TTL Data Lines
R	Reserved

J3 – Differential I/O Connector Channels 0-31

The following are connector pin assignments for J3 Differential I/O Channels 0-31 Connector with high level in J1 inputs pins 27 and 28:

#	Signal	#	Signal	#	Signal	#	Signal
1	DIO0+	18	DIO17+	35	DIO0-	52	DIO17-
2	DIO1+	19	DIO18+	36	DIO1-	53	DIO18-
3	DIO2+	20	DIO19+	37	DIO2-	54	DIO19-
4	DIO3+	21	DIO20+	38	DIO3-	55	DIO20-
5	DIO4+	22	DIO21+	39	DIO4-	56	DIO21-
6	DIO5+	23	DIO22+	40	DIO5-	57	DIO22-
7	DIO6+	24	DIO23+	41	DIO6-	58	DIO23-
8	DIO7+	25	DIO24+	42	DIO7-	59	DIO24-
9	DIO8+	26	DIO25+	43	DIO8-	60	DIO25-
10	DIO9+	27	DIO26+	44	DIO9-	61	DIO26-
11	DIO10+	28	DIO27+	45	DIO10-	62	DIO27-
12	DIO11+	29	DIO28+	46	DIO11-	63	DIO28-
13	DIO12+	30	DIO29+	47	DIO12-	64	DIO29-
14	DIO13+	31	DIO30+	48	DIO13-	65	DIO30-
15	DIO14+	32	DIO31+	49	DIO14-	66	DIO31-
16	DIO15+	33	5V	50	DIO15-	67	5V
17	DIO16+	34	GND	51	DIO16-	68	GND

Table 3-7: J3: Differential I/O Connector Channels 0-31

DIO0+ to DIO31+	Input/Output Differential Data Lines High
DIO0- to DIO31-	Input/Output Differential Data Lines Low
5V	5 Volt Power
GND	Ground

J4 – Differential I/O Connector Channels 32-63

The following are connector pin assignments for J4 Differential I/O Channels 32-63 Connector with high level in J1 inputs pins 27 and 28:

#	Signal	#	Signal	#	Signal	#	Signal
1	DIO32+	18	DIO49+	35	DIO32-	52	DIO49-
2	DIO33+	19	DIO50+	36	DIO33-	53	DIO50-
3	DIO34+	20	DIO51+	37	DIO34-	54	DIO51-
4	DIO35+	21	DIO52+	38	DIO35-	55	DIO52-
5	DIO36+	22	DIO53+	39	DIO36-	56	DIO53-
6	DIO37+	23	DIO54+	40	DIO37-	57	DIO54-
7	DIO38+	24	DIO55+	41	DIO38-	58	DIO55-
8	DIO39+	25	DIO56+	42	DIO39-	59	DIO56-
9	DIO40+	26	DIO57+	43	DIO40-	60	DIO57-
10	DIO41+	27	DIO58+	44	DIO41-	61	DIO58-
11	DIO42+	28	DIO59+	45	DIO42-	62	DIO59-
12	DIO43+	29	DIO60+	46	DIO43-	63	DIO60-
13	DIO44+	30	DIO61+	47	DIO44-	64	DIO61-
14	DIO45+	31	DIO62+	48	DIO45-	65	DIO62-
15	DIO46+	32	DIO63+	49	DIO46-	66	DIO63-
16	DIO47+	33	5V	50	DIO47-	67	5V
17	DIO48+	34	GND	51	DIO48-	68	GND

Table 3-8: J4: Differential I/O Connector Channels 32-63

DIO32+ to DIO63+	Input/Output Differential Data Lines High
DIO32- to DIO63-	Input/Output Differential Data Lines Low
5V	5 Volt Power
GND	Ground

Connectors – Independent mode (JP2 installed)

Whenever JP2 jumper is installed the board is working in independent mode. When the card is configured to Independent mode the PXI interface is disabled and all 64 channels operate in Conversion mode. In this mode each channel conversion operation is predefined using DIP-Switches. Each channel can be predefined to convert TTL to Differential or Differential to TTL. The channels conversion settings will be loaded automatically upon power up.

NOTE: In this mode J1 pins 27 and 28 are set to output high.

J1 – TTL I/O Connector

The following are connector pin assignments for J1 TTL I/O Connector with JP2 installed:

#	Signal	#	Signal	#	Signal	#	Signal
1	GND	18	GND	35	TIO0	52	TIO17
2	GND	19	GND	36	TIO1	53	TIO18
3	GND	20	GND	37	TIO2	54	TIO19
4	GND	21	GND	38	TIO3	55	TIO20
5	GND	22	GND	39	TIO4	56	TIO21
6	GND	23	GND	40	TIO5	57	TIO22
7	GND	24	GND	41	TIO6	58	TIO23
8	GND	25	GND	42	TIO7	59	TIO24
9	GND	26	GND	43	TIO8	60	TIO25
10	GND	27	High	44	TIO9	61	TIO26
11	GND	28	High	45	TIO10	62	TIO27
12	GND	29	TIO28	46	TIO11	63	TIO29
13	GND	30	TIO30	47	TIO12	64	TIO31
14	GND	31	TIO32	48	TIO13	65	TIO33
15	GND	32	TIO34	49	TIO14	66	TIO35
16	GND	33	TIO36	50	TIO15	67	TIO37
17	GND	34	TIO38	51	TIO16	68	TIO39

Table 3-9: J1: TTL I/O Connector with JP2 installed

GND	Ground
TIO0 to TIO39	Input Output TTL Data Lines
5V Sense	Input Sense detecting if 5V (Logic High) is connected

J2 – TTL I/O Connector

The following are connector pin assignments for J2 TTL I/O Connector with JP2 installed:

#	Signal	#	Signal	#	Signal	#	Signal
1	GND	18	GND	35	R	52	TIO49
2	GND	19	GND	36	R	53	TIO50
3	GND	20	GND	37	R	54	TIO51
4	GND	21	GND	38	R	55	TIO52
5	GND	22	GND	39	R	56	TIO53
6	GND	23	GND	40	R	57	TIO54
7	GND	24	GND	41	R	58	TIO55
8	GND	25	GND	42	R	59	TIO56
9	GND	26	GND	43	TIO40	60	TIO57
10	GND	27	R	44	TIO41	61	TIO58
11	GND	28	R	45	TIO42	62	TIO59
12	GND	29	R	46	TIO43	63	TIO60
13	GND	30	R	47	TIO44	64	TIO61
14	GND	31	R	48	TIO45	65	TIO62
15	GND	32	R	49	TIO46	66	TIO63
16	GND	33	R	50	TIO47	67	R
17	GND	34	R	51	TIO48	68	R

Table 3-10: J2: TTL I/O Connector connected with JP2 installed

GND	Ground
TIO40 to TIO63	Input Output TTL Data Lines
R	Reserved

J3 – Differential I/O Connector Channels 0-31

The following are connector pin assignments for J3 Differential I/O Channels 0-31 Connector with JP2 installed::

#	Signal	#	Signal	#	Signal	#	Signal
1	DIO0+	18	DIO17+	35	DIO0-	52	DIO17-
2	DIO1+	19	DIO18+	36	DIO1-	53	DIO18-
3	DIO2+	20	DIO19+	37	DIO2-	54	DIO19-
4	DIO3+	21	DIO20+	38	DIO3-	55	DIO20-
5	DIO4+	22	DIO21+	39	DIO4-	56	DIO21-
6	DIO5+	23	DIO22+	40	DIO5-	57	DIO22-
7	DIO6+	24	DIO23+	41	DIO6-	58	DIO23-
8	DIO7+	25	DIO24+	42	DIO7-	59	DIO24-
9	DIO8+	26	DIO25+	43	DIO8-	60	DIO25-
10	DIO9+	27	DIO26+	44	DIO9-	61	DIO26-
11	DIO10+	28	DIO27+	45	DIO10-	62	DIO27-
12	DIO11+	29	DIO28+	46	DIO11-	63	DIO28-
13	DIO12+	30	DIO29+	47	DIO12-	64	DIO29-
14	DIO13+	31	DIO30+	48	DIO13-	65	DIO30-
15	DIO14+	32	DIO31+	49	DIO14-	66	DIO31-
16	DIO15+	33	5V	50	DIO15-	67	5V
17	DIO16+	34	GND	51	DIO16-	68	GND

Table 3-11: J3: Differential I/O Connector Channels 0-31

DIO0+ to DIO31+	Input/Output Differential Data Lines High
DIO0- to DIO31-	Input/Output Differential Data Lines Low
5V	5 Volt Power
GND	Ground

J4 – Differential I/O Connector Channels 32-63

The following are connector pin assignments for J4 Differential I/O Channels 32-63 Connector with JP2 installed:

#	Signal	#	Signal	#	Signal	#	Signal
1	DIO32+	18	DIO49+	35	DIO32-	52	DIO49-
2	DIO33+	19	DIO50+	36	DIO33-	53	DIO50-
3	DIO34+	20	DIO51+	37	DIO34-	54	DIO51-
4	DIO35+	21	DIO52+	38	DIO35-	55	DIO52-
5	DIO36+	22	DIO53+	39	DIO36-	56	DIO53-
6	DIO37+	23	DIO54+	40	DIO37-	57	DIO54-
7	DIO38+	24	DIO55+	41	DIO38-	58	DIO55-
8	DIO39+	25	DIO56+	42	DIO39-	59	DIO56-
9	DIO40+	26	DIO57+	43	DIO40-	60	DIO57-
10	DIO41+	27	DIO58+	44	DIO41-	61	DIO58-
11	DIO42+	28	DIO59+	45	DIO42-	62	DIO59-
12	DIO43+	29	DIO60+	46	DIO43-	63	DIO60-
13	DIO44+	30	DIO61+	47	DIO44-	64	DIO61-
14	DIO45+	31	DIO62+	48	DIO45-	65	DIO62-
15	DIO46+	32	DIO63+	49	DIO46-	66	DIO63-
16	DIO47+	33	5V	50	DIO47-	67	5V
17	DIO48+	34	GND	51	DIO48-	68	GND

Table 3-12: J4: Differential I/O Connector Channels 32-63

DIO32+ to DIO63+	Input/Output Differential Data Lines High
DIO32- to DIO63-	Input/Output Differential Data Lines Low
5V	5 Volt Power
GND	Ground

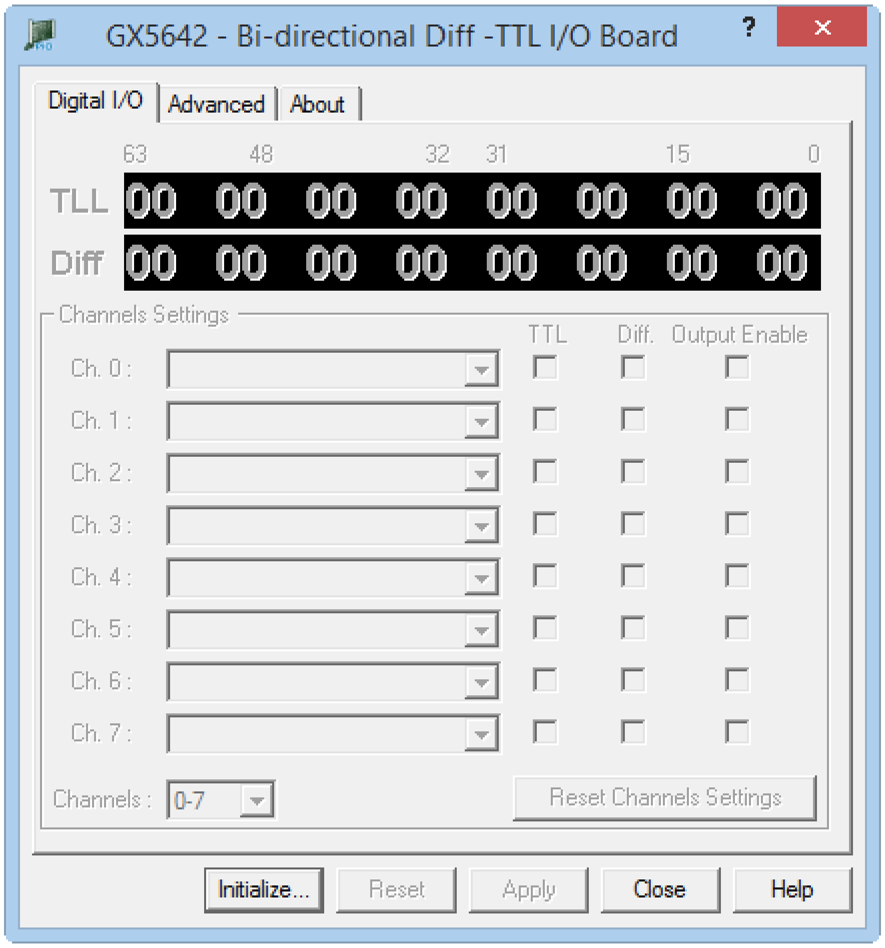
Chapter 4 - Instrument Software Panel

Calling the **Gx5642Panel** 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 **Gx5642Panel** function is also used by the GXPIOPANEL.EXE /5731 or GXPIOPANEL64.EXE /**5642** panel programs that is supplied with this package and provides a standalone Windows application that displays the instrument panel.

Virtual Panel Description

The GX5642 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 **GX5642 Panel** from the **Marvin Test Solutions**, **GXPIO** menu under the **Start** menu. The GX5642 virtual panel opens as shown here:



TLL: 00 00 00 00 00 00 00 00

Diff: 00 00 00 00 00 00 00 00

Channels : 0-7

Figure 4-1: GX5642 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.

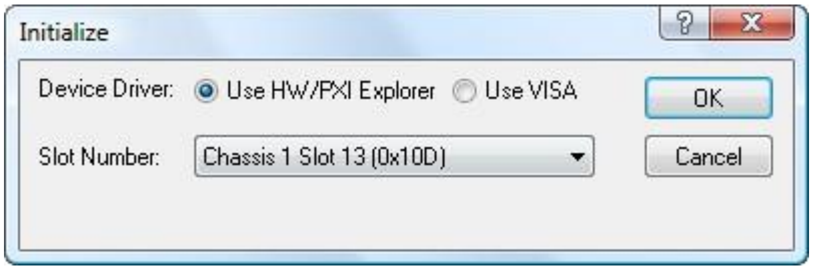
Help – Opens the on-line help window. In addition to the help menu, the caption shows a **What's This Help** button (?) button. This button can be used to obtain help on any control that is displayed in the panel window. To displays the What's This Help information click on the (?) button and then click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

4. **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 **GX5642** 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 **Gx5642Initialize** function which is used to initialize the driver for the specified board.

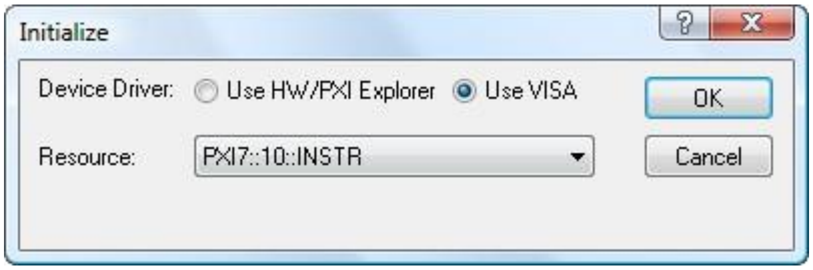

Initialize

Device Driver: [✓ Use HW/PXI Explorer] Use VISA

Slot Number: Chassis 1 Slot 13 (0x10D)

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

5. **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 **Gx5642InitializeVisa** function which initializes the driver for the specified board.


Initialize

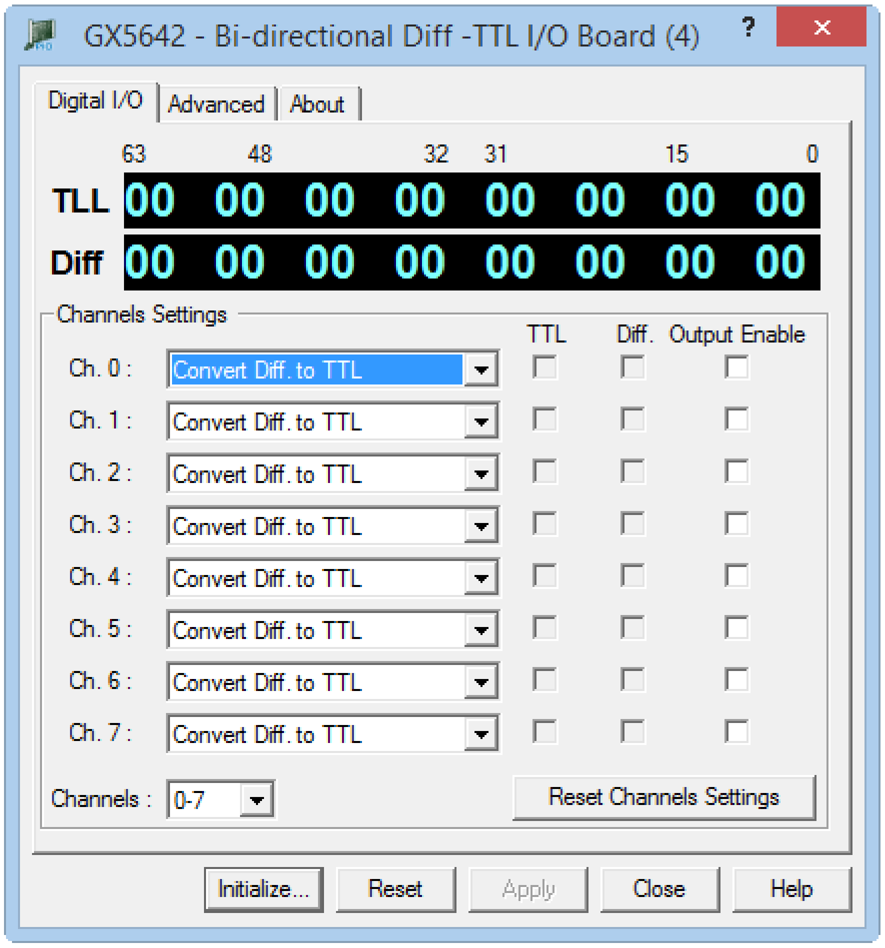
Device Driver: Use HW/PXI Explorer [✓ Use VISA]

Resource: PXI7::10::INSTR

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:



TLL	00	00	00	00	00	00	00	00
Diff	00	00	00	00	00	00	00	00

		TTL	Diff.	Output Enable
Ch. 0 :	Convert Diff. to TTL	☐	☐	☐
Ch. 1 :	Convert Diff. to TTL	☐	☐	☐
Ch. 2 :	Convert Diff. to TTL	☐	☐	☐
Ch. 3 :	Convert Diff. to TTL	☐	☐	☐
Ch. 4 :	Convert Diff. to TTL	☐	☐	☐
Ch. 5 :	Convert Diff. to TTL	☐	☐	☐
Ch. 6 :	Convert Diff. to TTL	☐	☐	☐
Ch. 7 :	Convert Diff. to TTL	☐	☐	☐

Channels : 0-7

Figure 4-4: GX5642 Virtual Panel – Digital I/O page

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

Channel Settings drop-down box (Channels 0+8n – 7+8n): Selects the mode settings, conversion or static I/O, and data port direction, input or output, for each channel.

Port Value display area: Displays data values of each channel for the TTL and Differential ports. The value is displayed in hexadecimal. The right two digits correspond to byte 0 and the left two digits to byte 7. Bytes 0-3 represent Group 1 and bytes 4-7 represent Group 2.

Channels drop-down box (Channels 0+8n – 7+8n): Selects the group of 8 channels that is currently being shown on the Digital I/O page. All changes MADE to the settings will apply to this range of channels.

Channel Settings drop-down box (Channels 0+8n – 7+8n): Selects the mode settings, conversion or static I/O, and data port direction, input or output, for each channel.

TTL Data checkbox (Channels 0+8n – 7+8n): A checked box represents a logical 1 and an unchecked box represents a logical 0 for the respective TTL I/O channel. The checkbox will only be active if the channel's output is enabled and the TTL port is set to output. Otherwise it provides a read back of the channel's logic state.

Differential RS-422 Data checkbox (Channels 0+8n – 7+8n): A checked box represents a logical 1 and an unchecked box represents a logical 0 for the respective Differential I/O channel. The checkbox will only be active if the channel's output is enabled and the Differential port is set to output. Otherwise it provides a read back of the channel's logic state.

Output Enable checkbox (Channels 0+8n – 7+8n): This checkbox represents the current output state of the I/O channel. A checked box indicates that the channel's output is enabled and an unchecked box indicates that the channel's output is disabled.

Reset Channels Settings: Resets the currently selected I/O Channels mode to Conversion Mode, direction to input, data to 0, and Output to Disabled.

Virtual Panel Advanced Page

Clicking on the **Advanced** tab will show the **Advanced page** as shown in Figure 2-8

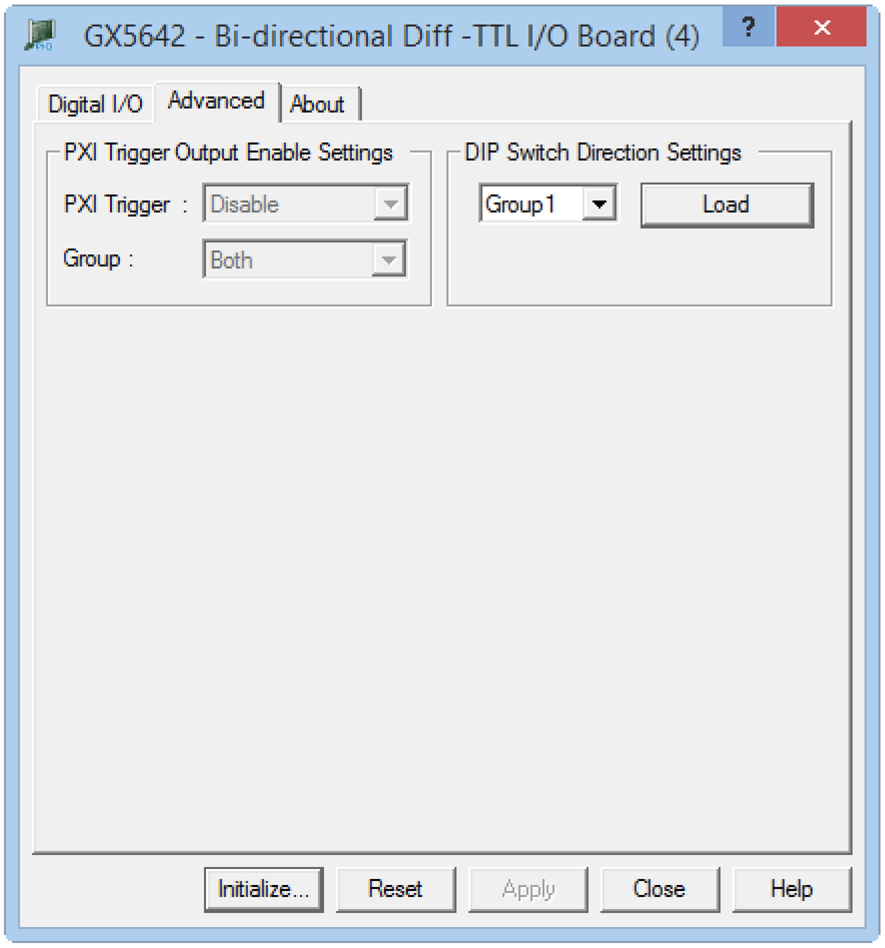

GX5642 - Bi-directional Diff -TTL I/O Board (4)
Digital I/O
Advanced
About
PXI Trigger Output Enable Settings
PXI Trigger :
Disable
Group :
Both
DIP Switch Direction Settings
Group1
Load
Initialize...
Reset
Apply
Close
Help

Figure 4-5: GX5642 Virtual Panel – Advanced Settings page

The following controls are shown in the Advanced page:

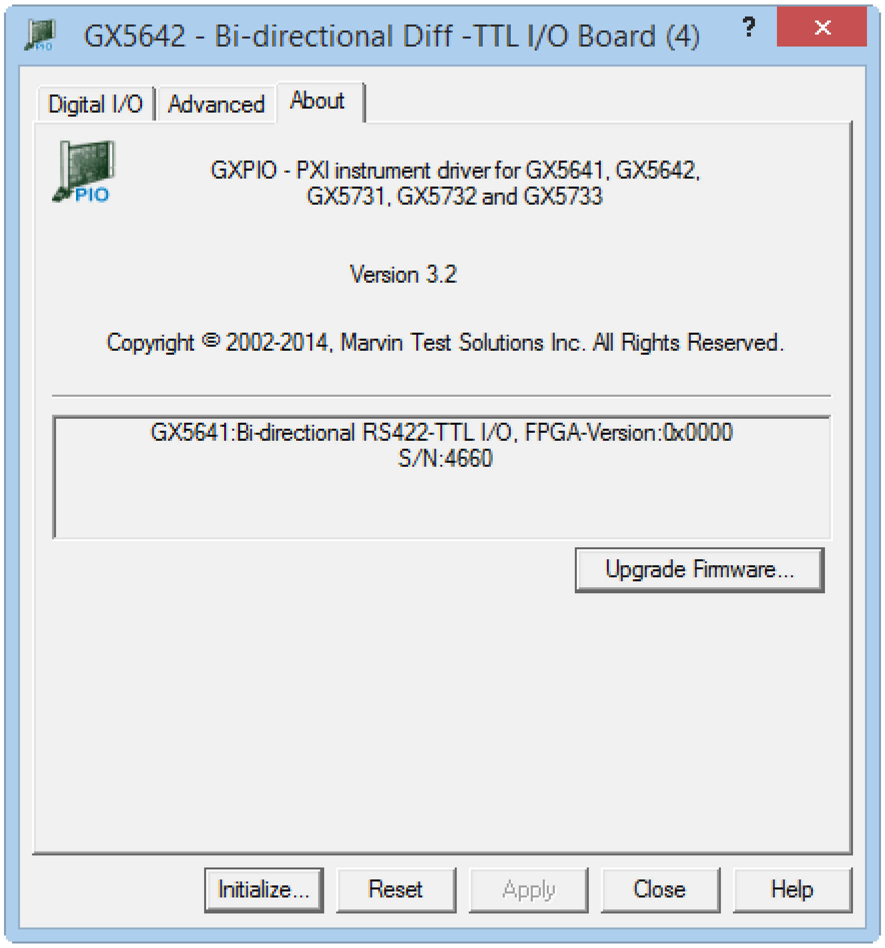
PX Trigger Output Enable Settings drop-down box: Selects the PXI Trigger line (0-7) to use for controlling all the I/O channels' output states. When the respective PXI Trigger line is raised to a logic 1, all the channels' output states will be enabled. When Disable is selected, the PXI Trigger bus is not used to control the I/O channels' output states.

DIP Switch Direction Settings drop-down box: Selects the group of DIP switches to load the I/O direction settings from.

Load: Loads the DIP Switch I/O Direction settings into all the channels of the group currently selected.

Virtual Panel About Page

Clicking on the **About** tab will show the **About page** as shown in Figure 2-9



GXPIO - PXI instrument driver for GX5641, GX5642, GX5731, GX5732 and GX5733

Version 3.2

Copyright © 2002-2014, Marvin Test Solutions Inc. All Rights Reserved.

GX5641:Bi-directional RS422-TTL I/O, FPGA-Version:0x0000
S/N:4660

Figure 4-6: GX5642 Virtual Panel – About Page

The top part of the **About** page displays version and copyright of the GX5642 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 GX5642 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.

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.

The GXPIO Driver

The GXPIO driver is a 32-bit Windows DLL file GXPIO.DLL and a 64-bit DLL GXPIO64.DLL. The DLLs are 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 GXPIO driver with C/C++ development tools:

- Include the GXPIO.H header file in the C/C++ source file that uses the GXPIO 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 GXPIO 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 or above (for 32-bit applications), the user must include the GXPIO.BAS to the project. The file can be loaded using *Add File* from the Visual Basic *File menu*. The GXPIO.BAS 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.BAS to the project. The GxPio.pas file contains a **unit** with function prototypes for the DLL functions. Include the GXPIO unit in the **uses** statement before making calls to the GXPIO functions.

Programming GXPIO Boards Using ATEasy®

The GXPIO package is supplied with a separate ATEasy driver for each board types. The GX5642.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 PXI HW slot number or VISA resource string 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 exceptions that allow 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 for GX5642.
- 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

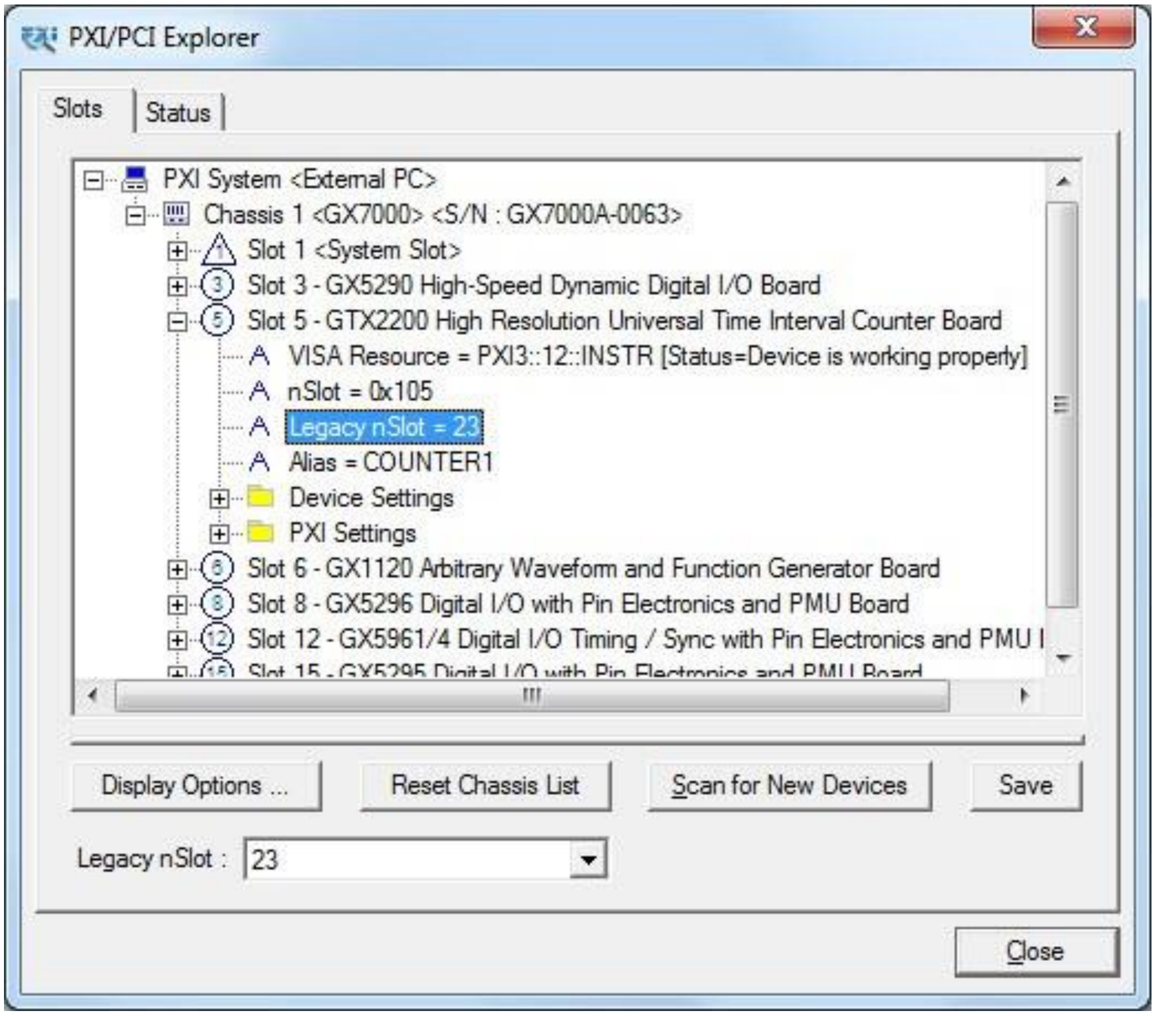
The GXPIO driver contains a set of functions for each of the supported instrument boards. The function name also starts with the board type. For example the GX5642 board uses the Gx6542Xxxx functions. Other functions are available as general purpose functions that apply to all boards (i.e GxPioGetDriverSummary).

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 **Gx5642Initialize** to initialize the board's driver using HW and **Gx5642InitializeVisa** 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 **GX5642Initialize**(*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.


PXI/PCI Explorer

Slots | Status

PXI System <External PC>
Chassis 1 <GX7000> <S/N : GX7000A-0063>
Slot 1 <System Slot>
Slot 3 - GX5290 High-Speed Dynamic Digital I/O Board
Slot 5 - GTX2200 High Resolution Universal Time Interval Counter Board
VISA Resource = PXI3::12::INSTR [Status=Device is working properly]
nSlot = 0x105
Legacy nSlot = 23
Alias = COUNTER1
Device Settings
PXI Settings
Slot 6 - GX1120 Arbitrary Waveform and Function Generator Board
Slot 8 - GX5296 Digital I/O with Pin Electronics and PMU Board
Slot 12 - GX5961/4 Digital I/O Timing / Sync with Pin Electronics and PMU I
Slot 15 - GX5295 Digital I/O with Pin Electronics and PMU Board

Legacy nSlot : 23

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 **GX5642InitializeVisa** (*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 **Gx5642Initialize** and the **Gx5642InitializeVisa** 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.

Once the board is initialized the handle can be used with other functions calls to program the board.

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 **Gx5642Panel** 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 **Gx5642Panel** 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 and the HW device driver files located in the HW folder) can be shipped with the application. Typically, the GXPIO.DLL should be copied to the Windows System directory. The HW device driver files should be installed using a special setup program HWSETUP.EXE that is provided with GXPIO driver files. Alternatively, you can provide the GXPIO disk to be installed along with the board.

Chapter 6 - Functions Reference

Introduction

The GX5642 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 **Gx5642Initialize** or the f **Gx5642InitializeVisa** 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 GX5642.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	Double word (unsigned 32-bit)	dwTimeout
l	Long (signed 32-bit)	lBits
p	Pointer. Usually used to return a value. Prefix this before the simple type.	pnStatus
sz	Null (zero value character) terminated string	szMsg
w	Unsigned short (unsigned 16-bit)	wParam
hwnd	Window handle (32-bit integer).	hwndPanel

Table 6-1: Parameter Prefixes

GX5642 Functions

The following list is a summary of functions available for the GX5642:

Driver Functions	Description
General	
Gx5642Initialize	Initializes the GX5642 driver for the specified PXI slot.
Gx5642InitalizeVisa	Initializes the driver for the specified slot using VISA. The function returns a handle that can be used with other GX5642 functions to program the board.
Gx5642Panel	Opens a virtual panel used to interactively control the GX5642.
Gx5642Reset	Opens all the board relays.
Gx5642GetBoardSummary	Returns the board summary.
GxPioGetDriverSummary	Returns the driver name and version.
GxPioGetErrorString	Returns the error string associated with the specified error number.
Channel settings	
Gx5642GetChannelConversionMode	Returns the specified channel Conversion Mode.
Gx5642GetChannelDifferentialPort	Returns the specified channel Differential Port value.
Gx5642GetChannelDifferentialPortDirection	Returns the specified channel Differential Port Direction.
Gx5642GetChannelMode	Returns the specified channel operating mode.
Gx5642GetChannelOutputState	Returns the specified channel Output State.
Gx5642GetChannelTTLPort	Returns the specified channel TTL Port value.
Gx5642GetChannelTTLPortDirection	Returns the specified channel Port Direction.
Gx5642SetChannelConversionMode	Sets the specified channel Conversion Mode.
Gx5642SetChannelDifferentialPort	Sets the specified channel Differential Port value.
Gx5642SetChannelDifferentialPortDirection	Sets the specified channel Differential Port Direction.
Gx5642SetChannelMode	Sets the specified channel operating mode.
Gx5642SetChannelOutputState	Sets the specified channel Output State.
Gx5642SetChannelTTLPort	Sets the specified channel TTL Port value.
Gx5642SetChannelTTLPortDirection	Sets the specified channel TTL Port Direction.
Group settings	
Gx5642GetGroupConversionMode	Returns the specified group conversion mode.
Gx5642GetGroupDifferentialPort	Returns the specified group differential port value.
Gx5642GetGroupDifferentialPortDirection	Returns the specified group differential Port Direction.
Gx5642GetGroupMode	Returns the specified group mode.
Gx5642GetGroupOutputState	Returns the specified group output state.

Driver Functions	**Description**
Gx5642GetGroupTTLPort	Returns the specified group TTL Port value.
Gx5642GetGroupTTLPortDirection	Returns the specified group TTL Port Direction.
Gx5642GetPxiTriggerBusToGroupOutputState	Returns the specified PXI trigger bus line controlling the specified group outputs states.
Gx5642LoadGroupDirectionFromDIPSwitch	Load and set the specified group Direction according to the on-board DIP Switch settings.
Gx5642ResetGroup	Resets the specified group its default settings.
Gx5642SetGroupConversionMode	Sets the specified group conversion mode.
Gx5642SetGroupDifferentialPort	Sets the specified group differential Port value.
Gx5642SetGroupDifferentialPortDirection	Sets the specified group differential Port Direction.
Gx5642SetGroupMode	Sets the specified group mode.
Gx5642SetGroupOutputState	Sets the specified group output state.
Gx5642SetGroupTTLPort	Sets the specified group TTL Port value.
Gx5642SetGroupTTLPortDirection	Sets the specified group TTL Port Direction.
Gx5642SetPxiTriggerBusToGroupOutputState	Sets the specified PXI trigger bus line to control the specified group outputs states.

Gx5642GetBoardSummary

Purpose

Returns the board summary.

Syntax

Gx5642GetBoardSummary (*nHandle*, *szSummary*, *nSumMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a GX5642 board.
szSummary	PSTR	Buffer to contain the returned board info (null terminated) string.
nSumMaxLen	SHORT	Size of the buffer to contain the board info string.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX5642 summary string provides the following data from in the order shown:

- Instrument Name (e.g., GX5642)
- FPGA version (e.g. 0xA002)
- Serial Number (e.g. 56420210)

For example, the returned string looks like the following:

```
"GX5642, FPGA-Version:0xA002, S/N 56420210"
```

See Also

GxPioetDriverSummary, GxPioGetErrorString

Gx5642GetChannelConversionMode

Purpose

Returns the specified channel Conversion Mode.

Syntax

Gx5642GetChannelConversionMode (*nHandle, nChannel, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pnMode	PSHORT	Conversion Mode can be as follows: 0. GX5642_CHANNEL_CONVERT_DIFFERENTIAL_TO_TTL – convert differential signal to TTL level. 1. GX5642_CHANNEL_CONVERT_TTL_TO_DIFFERENTIAL – convert TTL level to differential.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the specified channel conversion mode: differential signal to TTL level or convert TTL level to differential.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

<u>Note</u>: The channel should be set to conversion mode prior calling this function by calling **Gx5642SetChannelMode** otherwise **t**he function returns an error.

Example

The following example sets channel 0 to conversion mode, sets the conversion to be TTL level to differential, enables the channel and returns the channels conversion mode:

```
SHORT nMode;

Gx5642SetChannelMode (nHandle, 0, GX5642_CHANNEL_MODE_CONVERSION, &nStatus);
Gx5642SetChannelConversionMode (nHandle, 0, GX5642_CHANNEL_CONVERT_TTL_TO_DIFFERENTIAL,
                                 &nStatus);
Gx5642SetChannelOutputState(nHandle, 0, GX5642_CHANNEL_OUTPUT_ENABLE, &nStatus);
Gx5642GetChannelConversionMode (nHandle, 0, &nMode, &nStatus);
```

See Also

Gx5642SetChannelMode, Gx5642SetChannelConversionMode, Gx5642SetChannelOutputState, GxPioGetErrorString

Gx5642GetChannelDifferentialPort

Purpose

Returns the specified channel Differential Port value.

Syntax

Gx5642GetChannelDifferentialPort (*nHandle, nChannel, pbData, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pbData	PBOOL	Differential Port value: 0. Logic low 1. Logic high
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

The following example returns channel 0 Differential Port value:

```
BOOL bData;
Gx5642GetChannelDifferentialPort (nHandle, 0, &bData, &nStatus);
```

See Also

Gx5642SetChannelDifferentialPort, **GxPioGetErrorString**

Gx5642GetChannelDifferentialPortDirection

Purpose

Returns the specified channel Differential Port Direction.

Syntax

Gx5642GetChannelDifferentialPortDirection (*nHandle*, *nChannel*, *pnDirection*, *pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pnDirection	PSHORT	The channel Differential Port Direction can be as follows: 0. GX5642_CHANNEL_PORT_INPUT – channel port is set as input. 1. GX5642_CHANNEL_PORT_OUTPUT – channel port is set as output.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Note: The channel should be set to Static I/O mode prior calling this function by calling **Gx5642SetChannelMode** otherwise **t**he function returns an error.

Example

The following example returns channel 0 Differential Port Direction:

```
SHORT nDirection;
Gx5642GetChannelDifferentialPortDirection (nHandle, 0, &nDirection, &nStatus);
```

See Also

Gx5642SetChannelDifferentialPortDirection, **Gx5642SetChannelDifferentialPort**, **Gx5642SetChannelMode, GxPioGetErrorString**

Gx5642GetChannelMode

Purpose

Returns the specified channel operating mode.

Syntax

Gx5642GetChannelMode (*nHandle*, *nChannel*, *pnMode*, *pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pnMode	PSHORT	Channel operating modes are as follows: 0. GX5642_CHANNEL_MODE_CONVERSION – channels is set for conversion mode. 1. GX5642_CHANNEL_MODE_STATIC_IO – channels is set for static I/O mode.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

The following example returns channel 0 mode:

```
Gx5642GetChannelMode (nHandle, 0, &nMode, &nStatus);
```

See Also

Gx5642SetChannelMode, **Gx5642GetChannelDifferentialPortDirection**, **Gx5642SetChannelDifferentialPort**, **GxPioGetErrorString**

Gx5642GetChannelOutputState

Purpose

Returns the specified channel Output State.

Syntax

Gx5642GetChannelOutputState (*nHandle, nChannel, pnState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pnState	PSHORT	Channel output states are as follows: 0. GX5642_CHANNEL_OUTPUT_DISABLE - channel output is disabled. 1. GX5642_CHANNEL_OUTPUT_ENABLE - channel output is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

The following returns channel 0 output state:

```
SHORT nState;
Gx5642GetChannelOutputState (nHandle, 0, &nState, &nStatus);
```

See Also

Gx5642SetChannelOutputState, **Gx5642GetChannelMode**, **Gx5642GetChannelDifferentialPortDirection**, **Gx5642SetChannelDifferentialPort**, **GxPioGetErrorString**

Gx5642GetChannelTTLPort

Purpose

Returns the specified channel TTL Port value.

Syntax

Gx5642GetChannelTTLPort (*nHandle*, *nChannel*, *pbData*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pbData	PBOOL	TTL Port value: 0. Logic low 1. Logic high
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

The following example returns channel 0 TTL Port value:

```
BOOL bData;
Gx5642GetChannelTTLPort (nHandle, 0, &bData, &nStatus);
```

See Also

Gx5642SetChannelTTLPort, GxPioGetErrorString

Gx5642GetChannelTTLPortDirection

Purpose

Returns the specified channel Port Direction.

Syntax

Gx5642GetChannelTTLPortDirection (*nHandle*, *nChannel*, *pnDirection*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pnDirection	PSHORT	The channel TTL Port Direction can be as follows: 0. GX5642_CHANNEL_PORT_INPUT – channel port is set as input. 1. GX5642_CHANNEL_PORT_OUTPUT – channel port is set as output.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Note: The channel should be set to Static I/O mode prior calling this function by calling **Gx5642SetChannelMode** otherwise **t**he function returns an error.

Example

The following example returns channel 0 TTL Port Direction:

```
SHORT nDirection;
Gx5642GetChannelTTLPortDirection (nHandle, 0, &nDirection, &nStatus);
```

See Also

Gx5642SetChannelTTLPortDirection, **Gx5642SetChannelTTLPort**, **Gx5642SetChannelMode, GxPioGetErrorString**

Gx5642GetGroupConversionMode

Purpose

Returns the specified group channels conversion mode.

Syntax

Gx5642GetGroupConversionMode (*nHandle, nGroup, pdwMode, pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pdwMode	PDWORD	Each of the 32 bits represents a channel in the group, bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - GX5642_CHANNEL_CONVERT_DIFFERENTIAL_TO_TTL – convert differential signal to TTL level. Bit high - GX5642_CHANNEL_CONVERT_TTL_TO_DIFFERENTIAL – convert TTL level to differential.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the conversion mode for all the channels in the specified group.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Note: The group should be set to conversion mode prior calling this function by calling **Gx5642SetGroupMode**.

Example

```
DWORD dwMode;

// Sets all the channels in group 0 to conversion mode.
Gx5642SetGroupMode (nHandle, GX5642_GROUP0, 0, &nStatus);
// sets channels 1, 5, 8 TTL to differential conversion mode, rest of the channels are set to
// convert differential to TTL level.
Gx5642SetGroupConversionMode (nHandle, GX5642_GROUP0, 0x00000121, &nStatus);
// Enables all channels outputs.
Gx5642SetGroupOutputState (nHandle, GX5642_GROUP0, 0xFFFFFFFF, &nStatus);
// Returns all the channels in group 0 conversion mode
Gx5642GetCGroupConversionMode (nHandle, GX5642_GROUP0, &dwMode, &nStatus);
```

See Also

Gx5642SetGroupMode, Gx5642SetGroupConversionMode, Gx5642SetGroupOutputState, GxPioGetErrorString

Gx5642GetGroupDifferentialPort

Purpose

Returns the specified group differential ports values.

Syntax

Gx5642GetGroupDifferentialPort (*nHandle, nGroup, pdwPortData, pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pdwPortData	PDWORD	Group's differential ports values. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - differential channel port is logic low. Bit high - differential channel port is logic high.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the differential ports values for all the channels in the specified group.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

The following example returns group 0 Differential Ports values:

```
DWORD dwData;
Gx5642GetGroupDifferentialPort (nHandle, GX5642_GROUP0, &dwData, &nStatus);
```

See Also

Gx5642SetGroupDifferentialPort, **GxPioGetErrorString**

Gx5642GetGroupDifferentialPortDirection

Purpose

Returns the specified group differential Port Direction.

Syntax

Gx5642GetGroupDifferentialPortDirection (*nHandle, nGroup, pdwDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pdwDirection	PDWORD	Group's differential Port Direction. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel differential Port Direction is input. Bit high - channel differential Port Direction is output.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the differential Port Direction for all the channels in the specified group.

Note: Only channels in the group that were set to Static I/O mode prior calling this function by calling **Gx5642SetGroupMode** will be set.

Example

The following example returns group 0 Differential Port Direction:

```
DWORD dwDirection;
Gx5642GetGroupDifferentialPortDirection (nHandle, GX5642_GROUP0, &dwDirection, &nStatus);
```

See Also

Gx5642SetGroupDifferentialPortDirection, **Gx5642SetGroupDifferentialPort**, **Gx5642SetGroupMode, GxPioGetErrorString**

Gx5642GetGroupMode

Purpose

Returns the specified group mode.

Syntax

Gx5642GetGroupMode (*nHandle, nGroup, pdwMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pdwMode	PDWORD	Group's Mode. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel set for conversion mode. Bit high - channel set for static I/O mode.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the operating mode for all the channels in the specified group.

For protection, by default all channel's connections (TTL and Deferential) are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetGroupOutputState**.

Example

The following returns group 0 operating mode:

```
Gx5642GetGroupMode (nHandle, GX5642_GROUP0, &dwMode, &nStatus);
```

See Also

Gx5642SetGroupMode, Gx5642SetGroupConversionMode, Gx5642SetGroupOutputState, GxPioGetErrorString

Gx5642GetGroupOutputState

Purpose

Returns the specified group output state.

Syntax

Gx5642GetGroupOutputState (*nHandle, nGroup, pdwStates, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pdwStates	PDWORD	Group's output state. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel output is disabled. Bit high - channel output is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channel's connections (TTL and Deferential) are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

Example

```
DWORD dwMode;

// Sets all the channels in group 0 to conversion mode.
Gx5642SetGroupMode (nHandle, GX5642_GROUP0, 0, &nStatus);
// sets channels 1, 5, 8 TTL to differential conversion mode, rest of the channels are set to
// convert differential to TTL level.
Gx5642SetGroupConversionMode (nHandle, GX5642_GROUP0, 0x00000121, &nStatus);
// Enables all channels outputs.
Gx5642SetGroupOutputState (nHandle, GX5642_GROUP0, 0xFFFFFFFF, &nStatus);
// Returns all the channels in group 0 conversion mode
Gx5642GetCGroupConversionMode (nHandle, GX5642_GROUP0, &dwMode, &nStatus);
```

See Also

Gx5642SetGroupOutputState, Gx5642SetGroupMode, Gx5642SetGroupConversionMode, GxPioGetErrorString

Gx5642GetGroupTTLPort

Purpose

Returns the specified group TTL Ports values.

Syntax

Gx5642GetGroupTTLPort (*nHandle*, *pdwPortData*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pdwPortData	PDWORD	Group’s differential ports values. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - differential channel port is logic low. Bit high - differential channel port is logic high.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the TTL Ports values for all the channels in the specified group.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

The following example returns group 0 TTL Ports values:

```
DWORD dwData;
Gx5642GetGroupTTLPort (nHandle, GX5642_GROUP0, &dwData, &nStatus);
```

See Also

Gx5642SetGroupTTLPort, **GxPioGetErrorString**

Gx5642GetGroupTTLPortDirection

Purpose

Returns the specified group TTL Port Direction.

Syntax

Gx5642GetGroupTTLPortDirection (*nHandle, nGroup, pdwDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pdwDirection	PDWORD	Group's TTL Port Direction. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel differential Port TTL is input. Bit high - channel differential Port TTL is output.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the TTL Port Direction for all the channels in the specified group.

Note: Only channels in the group that were set to Static I/O mode prior calling this function by calling **Gx5642SetGroupMode** will be set.

Example

The following example returns group 0 TTL Port Direction:

```
DWORD dwDirection;
Gx5642GetGroupTTLPortDirection (nHandle, GX5642_GROUP0, &dwDirection, &nStatus);
```

See Also

Gx5642SetGroupTTLPortDirection, **Gx5642SetGroupTTLPort**, **Gx5642SetGroupMode, GxPioGetErrorString**

Gx5642GetPxiTriggerBusToGroupOutputState

Purpose

Returns the specified PXI trigger bus line controlling the specified group outputs states.

Syntax

Gx5642GetPxiTriggerBusToGroupOutputState (*nHandle, nGroup, pnLine, pbEnable, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pnline	PSHORT	Returned PXI Trigger Line number: 0. GX5642_XTRIGGER_LINE0: Line 0 1. GX5642_XTRIGGER_LINE1: Line 1 2. GX5642_XTRIGGER_LINE2: Line 2 3. GX5642_XTRIGGER_LINE3: Line 3 4. GX5642_XTRIGGER_LINE4: Line 4 5. GX5642_XTRIGGER_LINE5: Line 5 6. GX5642_XTRIGGER_LINE6: Line 6 7. GX5642_XTRIGGER_LINE7: Line 7
pbEnable	PBOOL	Specified PXI trigger line state 0. FALSE: the specified PXI trigger line does not control the specified group outputs. 1. TRUE: the specified PXI trigger line controls the specified group outputs. Whenever the specified PXI trigger line is high the specified group outputs are enabled, whenever the specified PXI trigger line is low the specified group outputs are disabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function allows the user to control all 32 channels of the specified group to either be enabled or disabled by controlling a single PXI trigger bus line.

Example

The following example returns the associated PXI trigger line with group 0 and its state:

```
BOOL    bEnable;
SHORT   nLine;
Gx5642GetPxiTriggerBusToGroupOutputState (nHandle, GX5642_GROUP0, &nLine, &bEnable, &nStatus);
```

See Also

Gx5642SetPxiTriggerBusToGroupOutputState, Gx5642SetGroupTTLPortDirection, **Gx5642SetGroupTTLPort**, **Gx5642SetGroupMode, GxPioGetErrorString**

Gx5642Initialize

Purpose

Initializes the driver for the board at the specified slot number. The function returns a handle that can be used with other GX5642 functions to program the board.

Syntax

Gx5642Initialize (*nSlot, pnHandle, pnStatus*)

Parameters

Name	Type	Comments
nSlot	Short	GX5642 board slot number on the PXI bus.
pnHandle	PSHORT	Returned handle for the board. The handle is set to zero on error and <> 0 on success.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **Gx5642Initialize** function verifies whether or not the GX5642 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 GX5642 functions.

Example

The following example initializes two GX5642 boards at slot 1 and 2.

```
SHORT nHandle1, nHandle2, nStatus;
Gx5642Initilize (1, &nHandle1, &nStatus);
Gx5642Initilize (2, &nHandle2, &nStatus);
if (nHandle1==0 || nHandle2==0)
    {   printf("Unable to Initialize the board")
        return;
}
```

See Also

Gx5642Reset, **GxPioGetErrorString**

Gx5642InitializeVisa

Purpose

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

Syntax

Gx5642InitializeVisa (*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 **Gx5642InitializeVisa** 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 GX5642 boards at PXI bus 5 and device 11.

```
SHORT nHandle, nStatus;
Gx5642InitializeVisa (“PXI5::11::INSTR”, &nHandle, &nStatus);
if (nHandle==0)
{
   printf("Unable to Initialize the board")
   return;
}
```

See Also

Gx5642Initialize, Gx5642Reset, **GxPIOGetErrorString**

Gx5642LoadGroupDirectionFromDIPSwitch

Purpose

Load and set the specified group Direction according to the on-board DIP Switch settings.

Syntax

Gx5642LoadGroupDirectionFromDIPSwitch (*nHandle*, *nGroup*, *pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The group's channels conversion settings will be loaded and set according to the on-board conversion direction DIP-Switches.

Note: All channels in the specified group need to be set to static I/O mode prior calling this function by calling **Gx5642SetGroupMode**.

Example

The following example load and set the group 0 Direction according to the on-board DIP Switch settings:

```
Gx5642LoadGroupDirectionFromDIPSwitch (nHandle, GX5642_GROUP0, &nStatus);
```

See Also

Gx5642SetGroupConversionMode,.Gx5642SetGroupMode, Gx5642SetGroupOutputState, GxPioGetErrorString

Gx5642Panel

Purpose

Opens a virtual panel used to interactively control the GX5642.

Syntax

Gx5642Panel *(pnHandle, hwndParent, nMode, phwndPanel, pnStatus)*

Parameters

Name	Type	Comments
pnHandle	PSHORT	Handle to a GX5642 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 GX5642 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;

Gx5642Panel(&nHandle, 0, 1, &dwPanel, &nStatus);
```

See Also

Gx5642Initialize, GxPioGetErrorString

Gx5642Reset

Purpose

Resets the GX5642 board to its default settings.

Syntax

Gx5642Reset (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

After calling this function all the channels will be disabled and in conversion mode of differential signal to TTL level.

Example

The following example initializes and resets the GX5642 board:

```
Gx5642Initialize (1, &nHandle, &nStatus);
Gx5642Reset (nHandle, &nStatus);
```

See Also

Gx5642Initialize, **GxPioGetErrorString**

Gx5642ResetGroup

Purpose

Resets the specified group its default settings.

Syntax

Gx5642ResetGroup (*nHandle*, *nGroup*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

After calling this function the specified group's channels will be disabled and in conversion mode of differential signal to TTL level.

Example

The following example rests group 0:

```
Gx5642ResetGroup (nHandle, GX5642_GROUP0, &nStatus);
```

See Also

Gx5642GetChannelMode, **Gx5642GetGroupMode**, **Gx5642GetChannelOutputState**, **Gx5642GetGroupOutputState**, **Gx5642Initialize**, **GxPioGetErrorString**

Gx5642SetChannelConversionMode

Purpose

Sets the specified channel Conversion Mode.

Syntax

Gx5642SetChannelConversionMode (*nHandle*, *nChannel*, *nMode*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
nMode	SHORT	Conversion Mode can be as follows: 0. GX5642_CHANNEL_CONVERT_DIFFERENTIAL_TO_TTL – convert differential signal to TTL level. 1. GX5642_CHANNEL_CONVERT_TTL_TO_DIFFERENTIAL – convert TTL level to differential.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function sets the specified channel to convert differential signal to TTL level or convert TTL level to differential.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Note: The channel should be set to conversion mode prior calling this function by calling **Gx5642SetChannelMode** otherwise **t**he function will return an error.

Example

The following example sets channel 0 to conversion mode, sets the conversion to be TTL level to differential, enables the channel and returns the channels conversion mode:

```
SHORT nMode;

Gx5642SetChannelMode (nHandle, 0, GX5642_CHANNEL_MODE_CONVERSION, &nStatus);
Gx5642SetChannelConversionMode (nHandle, 0, GX5642_CHANNEL_CONVERT_TTL_TO_DIFFERENTIAL,
                                 &nStatus);
Gx5642SetChannelOutputState(nHandle, 0, GX5642_CHANNEL_OUTPUT_ENABLE, &nStatus);
Gx5642GetChannelConversionMode (nHandle, 0, &nMode, &nStatus);
```

See Also

Gx5642SetChannelMode, Gx5642GetChannelConversionMode, Gx5642SetChannelOutputState, GxPioGetErrorString

Gx5642SetChannelDifferentialPort

Purpose

Sets the specified channel Differential Port value.

Syntax

Gx5642SetChannelDifferentialPort (*nHandle*, *nChannel*, *bData*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pbData	PBOOL	Differential Port value: 0. Logic low 1. Logic high
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function sets the Differential Port value. If the Differential Port direction was set to output then the function returns the current channel’s port settings, if the port direction was set to input was then the function returns the Differential input value.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

<u>**Note**</u>: The channel should be set to Static I/O mode prior calling this function by calling **Gx5642SetChannelMode** otherwise **t**he function returns an error.

Example

The following example sets channel 0 Differential Port value to 1:

```
Gx5642SetChannelDifferentialPort (nHandle, 0, 1, &nStatus);
```

See Also

Gx5642SetChannelDifferentialPort, **Gx5642SetChannelMode, GxPioGetErrorString**

Gx5642SetChannelDifferentialPortDirection

Purpose

Sets the specified channel Differential Port Direction.

Syntax

Gx5642SetChannelDifferentialPortDirection (*nHandle, nChannel, nDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
nDirection	SHORT	The channel Differential Port Direction can be as follows: 0. GX5642_CHANNEL_PORT_INPUT – channel port is set as input. 1. GX5642_CHANNEL_PORT_OUTPUT – channel port is set as output.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Note: The channel should be set to Static I/O mode prior calling this function by calling **Gx5642SetChannelMode** otherwise **t**he function returns an error.

Example

The following example sets channel 0 Differential Port Direction to output:

```
Gx5642SetChannelDifferentialPortDirection (nHandle, 0, GX5642_CHANNEL_PORT_OUTPUT, &nStatus);
```

See Also

Gx5642GetChannelDifferentialPortDirection, **Gx5642SetChannelDifferentialPort**, **Gx5642SetChannelMode, GxPioGetErrorString**

Gx5642SetChannelMode

Purpose

Sets the specified channel operating mode.

Syntax

Gx5642SetChannelMode (*nHandle, nChannel, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
nMode	SHORT	Channel operating modes are as follows: 0. GX5642_CHANNEL_MODE_CONVERSION – channels is set for conversion mode. 1. GX5642_CHANNEL_MODE_STATIC_IO - – channels is set for static I/O mode.
PnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

The following example sets channel 0 operating mode to static I/O:

```
Gx5642GetChannelMode (nHandle, 0, GX5642_CHANNEL_MODE_STATIC_IO, &nStatus);
```

See Also

Gx5642GetChannelMode, **Gx5642GetChannelDifferentialPortDirection**, **Gx5642SetChannelDifferentialPort**, **GxPioGetErrorString**

Gx5642SetChannelOutputState

Purpose

Sets the specified channel Output State.

Syntax

Gx5642SetChannelOutputState (*nHandle*, *nChannel*, *nState*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
pnState	PSHORT	Channel output states are as follows: 0. GX5642_CHANNEL_OUTPUT_DISABLE - channel output is disabled. 1. GX5642_CHANNEL_OUTPUT_ENABLE - channel output is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Example

The following enables channel 0 output state:

```
SHORT nState;
Gx5642SetChannelOutputState (nHandle, 0, GX5642_CHANNEL_OUTPUT_ENABLE, &nStatus);
```

See Also

Gx5642GetChannelOutputState, **Gx5642GetChannelMode**, **Gx5642GetChannelDifferentialPortDirection**, **Gx5642SetChannelDifferentialPort**, **GxPioGetErrorString**

Gx5642SetChannelTTLPort

Purpose

Sets the specified channel TTL Port value.

Syntax

Gx5642SetChannelTTLPort (*nHandle, nChannel, bData, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
bData	BOOL	TTL Port value: 0. Logic low 1. Logic high
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function sets the TTL Port value.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Note: The channel should be set to Static I/O mode prior calling this function by calling **Gx5642SetChannelMode** otherwise **t**he function returns an error.

Example

The following example sets channel 0 TTL Port value to 1:

```
Gx5642SetChannelTTLPort (nHandle, 0, 1, &nStatus);
```

See Also

Gx5642GetChannelTTLPort, **Gx5642SetChannelMode, GxPioGetErrorString**

Gx5642SetChannelTTLPortDirection

Purpose

Sets the specified channel TTL Port Direction.

Syntax

Gx5642SetChannelTTLPortDirection (*nHandle*, *nChannel*, *nDirection*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nChannel	SHORT	Channel range is: GX5642_FIRST_CHANNEL (0) to GX5642_LAST_CHANNEL (63).
nDirection	SHORT	The channel TTL Port Direction can be as follows: 0. GX5642_CHANNEL_PORT_INPUT – channel port is set as input. 1. GX5642_CHANNEL_PORT_OUTPUT – channel port is set as output.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Note: The channel should be set to Static I/O mode prior calling this function by calling **Gx5642SetChannelMode** otherwise **t**he function returns an error.

Example

The following example sets channel 0 TTL Port Direction to output:

```
Gx5642SetChannelTTLPortDirection (nHandle, 0, GX5642_CHANNEL_PORT_OUTPUT, &nStatus);
```

See Also

Gx5642GetChannelTTLPortDirection, **Gx5642SetChannelTTLPort**, **Gx5642SetChannelMode, GxPioGetErrorString**

Gx5642SetGroupConversionMode

Purpose

Sets the specified group channels conversion mode.

Syntax

Gx5642SetGroupConversionMode (*nHandle, nGroup, dwMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
dwMode	DWORD	Each of the 32 bits represents a channel in the group, bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - GX5642_CHANNEL_CONVERT_DIFFERENTIAL_TO_TTL – convert differential signal to TTL level. Bit high - GX5642_CHANNEL_CONVERT_TTL_TO_DIFFERENTIAL – convert TTL level to differential.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the conversion mode for all the channels in the specified group.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

<u>**Note**</u>: The group should be set to conversion mode prior calling this function by calling **Gx5642SetGroupMode**.

Example

Example

```
DWORD dwMode;

// Sets all the channels in group 0 to conversion mode.
Gx5642SetGroupMode (nHandle, GX5642_GROUP0, 0, &nStatus);
// sets channels 1, 5, 8 TTL to differential conversion mode, rest of the channels are set to
// convert differential to TTL level.
Gx5642SetGroupConversionMode (nHandle, GX5642_GROUP0, 0x00000121, &nStatus);
// Enables all channels outputs.
Gx5642SetGroupOutputState (nHandle, GX5642_GROUP0, 0xFFFFFFFF, &nStatus);
// Returns all the channels in group 0 conversion mode
Gx5642GetCGroupConversionMode (nHandle, GX5642_GROUP0, &dwMode, &nStatus);
```

See Also

Gx5642GetGroupMode, Gx5642SetGroupConversionMode, Gx5642SetGroupOutputState, GxPioGetErrorString

Gx5642SetGroupDifferentialPort

Purpose

Sets the specified group differential Port value.

Syntax

Gx5642SetGroupDifferentialPort (*nHandle, nGroup, dwPortData, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
dwPortData	DWORD	Group's differential ports values. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - differential channel port is logic low. Bit high - differential channel port is logic high.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function sets the differential ports values for all the channels in the specified group.

For protection, by default all channels are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetChannelOutputState**.

Note: Only channels in the group that were set to Static I/O mode prior calling this function by calling **Gx5642SetGroupMode** will be set.

Example

The following example sets group 0 differential Port value:

```
Gx5642SetGroupDifferentialPort (nHandle, GX5642_GROUP0, 0x55AA1234&nStatus);
```

See Also

Gx5642GetChannelDifferentialPort, **Gx5642GetChannelDifferentialPortDirection**, **GxPioGetErrorString**

Gx5642SetGroupDifferentialPortDirection

Purpose

Sets the specified group differential Port Direction.

Syntax

Gx5642SetGroupDifferentialPortDirection (*nHandle, nGroup, dwDirection, pnStatus*)

Parameters

Name	**Type**	**Comments**
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
dwDirection	DWORD	Group's output state. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel port is set as input. Bit high - channel port is set as output.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channel's connections (TTL and Deferential) are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetGroupOutputState**.

Note: Only channels in the group that were set to Static I/O mode prior calling this function by calling **Gx5642SetGroupMode** will be set.

Example

The following example sets group 0 Differential Port channels 0, 4, 8 direction to output:

```
Gx5642SetGroupDifferentialPortDirection (nHandle, GX5642_GROUP0, 0x00000111, &nStatus);
```

See Also

Gx5642GetGroupDifferentialPortDirection, Gx5642SetGroupDifferentialPort, Gx5642SetGroupMode, GxPioGetErrorString

Gx5642SetGroupMode

Purpose

Sets the specified group mode.

Syntax

Gx5642SetGroupMode (*nHandle*, *nGroup*, *dwMode*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
dwMode	DWORD	Group's Mode. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel set for conversion mode. Bit high - channel set for static I/O mode.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function sets the operating mode for all the channels in the specified group.

For protection, by default all channel's connections (TTL and Deferential) are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetGroupOutputState**.

Example

The following example sets the channels 1, 5, and 8 to static I/O mode:

```
Gx5642SetGroupMode (nHandle, GX5642_GROUP0, 0x00000121, &nStatus);
```

See Also

Gx5642GetGroupMode, Gx5642SetGroupConversionMode, Gx5642SetGroupOutputState, GxPioGetErrorString

Gx5642SetGroupOutputState

Purpose

Sets the specified group output state.

Syntax

Gx5642SetGroupOutputState (*nHandle, nGroup, dwStates, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
dwStates	DWORD	Group's output state. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel output is disabled. Bit high - channel output is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channel's connections (TTL and Deferential) are disabled after power-up and reset.

Example

```
DWORD dwMode;

// Sets all the channels in group 0 to conversion mode.
Gx5642SetGroupMode (nHandle, GX5642_GROUP0, 0, &nStatus);
// sets channels 1, 5, 8 TTL to differential conversion mode, rest of the channels are set to
// convert differential to TTL level.
Gx5642SetGroupConversionMode (nHandle, GX5642_GROUP0, 0x00000121, &nStatus);
// Enables all channels outputs.
Gx5642SetGroupOutputState (nHandle, GX5642_GROUP0, 0xFFFFFFFF, &nStatus);
// Returns all the channels in group 0 conversion mode
Gx5642GetCGroupConversionMode (nHandle, GX5642_GROUP0, &dwMode, &nStatus);
```

See Also

Gx5642GetGroupOutputState, Gx5642SetGroupMode, Gx5642SetGroupConversionMode, GxPioGetErrorString

Gx5642SetGroupTTLPort

Purpose

Sets the specified group TTL Port value.

Syntax

Gx5642SetGroupTTLPort (*nHandle*, *nGroup*, *dwPortData*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
dwPortData	DWORD	Group's output state. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel logic low. Bit high - channel logic high.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channel's connections (TTL and Deferential) are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetGroupOutputState**.

Note: Only channels in the group that were set to Static I/O mode prior calling this function by calling **Gx5642SetGroupMode** will be set.

Example

The following example sets group 0 TTL Port channels 0, 4, 8 value to 1

```
Gx5642SetGroupTTLPort (nHandle, GX5642_GROUP0, 0x00000111, &nStatus);
```

See Also

Gx5642GetGroupTTLPort, **Gx5642SetGroupMode**, **GxPioGetErrorString**

Gx5642SetGroupTTLPortDirection

Purpose

Sets the specified group TTL Port Direction.

Syntax

Gx5642SetGroupTTLPortDirection (*nHandle, nGroup, dwDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
dwDirection	DWORD	Group's output state. Each of the 32 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 31 is the last channel in the group. Bit low - channel port is set as input. Bit high - channel port is set as output.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For protection, by default all channel's connections (TTL and Deferential) are disabled after power-up and reset. The channels can be enabled and or disabled by calling **Gx5642SetGroupOutputState**.

Note: Only channels in the group that were set to Static I/O mode prior calling this function by calling **Gx5642SetGroupMode** will be set.

Example

The following example sets group 0 TTL Port channels 0, 4 and 8 direction to output:

```
Gx5642SetGroupTTLPortDirection (nHandle, GX5642_GROUP0, 0x00000111, &nStatus);
```

See Also

Gx5642GetGroupTTLPortDirection, **Gx5642SetGroupTTLPort**, **Gx5642SetGroupMode, GxPioGetErrorString**

Gx5642SetPxiTriggerBusToGroupOutputState

Purpose

Sets the specified PXI trigger bus line controlling the specified group outputs states.

Syntax

Gx5642SetPxiTriggerBusToGroupOutputState (*nHandle*, *nGroup*, *nLine*, *bEnable*, *pnStatus*)

Parameters

Name	Type	Comments
NHandle	SHORT	Handle for a GX5642 board.
nGroup	SHORT	Group value is as follows: 0. GX5642_GROUP0 (channels 0 to 31) 1. GX5642_GROUP1(channels 32 to 63)
nline	SHORT	PXI Trigger Line number: 0. GX5642_XTRIGGER_LINE0: Line 0 1. GX5642_XTRIGGER_LINE1: Line 1 2. GX5642_XTRIGGER_LINE2: Line 2 3. GX5642_XTRIGGER_LINE3: Line 3 4. GX5642_XTRIGGER_LINE4: Line 4 5. GX5642_XTRIGGER_LINE5: Line 5 6. GX5642_XTRIGGER_LINE6: Line 6 7. GX5642_XTRIGGER_LINE7: Line 7
bEnable	BOOL	Specified PXI trigger line state 0. FALSE: the specified PXI trigger line does not control the specified group outputs. 1. TRUE: the specified PXI trigger line controls the specified group outputs. Whenever the specified PXI trigger line is high the specified group outputs are enabled, whenever the specified PXI trigger line is low the specified group outputs are disabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function allows the user to control all 32 channels of the specified group to either be enabled or disabled by controlling a single PXI trigger bus line.

Example

The following example sets the associated PXI trigger line with group 0 and its state:

```
BOOL    bEnable;
SHORT   nLine;
Gx5642SetPxiTriggerBusToGroupOutputState (nHandle, GX5642_GROUP0, GX5642_XTRIGGER_LINE1, TRUE,
&nStatus);
```

See Also

Gx5642GetPxiTriggerBusToGroupOutputState, Gx5642SetGroupTTLPortDirection, **Gx5642SetGroupTTLPort**, **Gx5642SetGroupMode, GxPioGetErrorString**

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 GX5642. Version 2.10, Copyright © Marvin Test Solutions 2006.".

Example

The following example prints the driver version:

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

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

See Also

GxPioGetErrorString

GxPioGetErrorString

Purpose

Returns the error string associated with the specified error number.

Syntax

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

Parameters

Name	Type	Comments
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

Example

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

```
CHAR    sz[256];
SHORT   nStatus, nHandle;
..
Gx5642Initialize (3, &Handle, &Status);
if (nStatus<0)
{   GxPioGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz);// prints the error string returns

}
```

Index