

GX5733

**Digital I/O PXI Board
with 96 TTL Channels
and 32 Customizable Channels
GXPIO Software
*User's Guide***

Last Updated: June 22, 2015



Safety and Handling

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

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

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

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

Warranty

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

If You Need Help

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

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

Disclaimer

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

Copyright

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

Trademarks

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

All other trademarks are the property of their respective owners.

Table of Contents

Safety and Handling.....	1
Warranty	1
If You Need Help.....	1
Disclaimer	1
Copyright	1
Trademarks	2
Chapter 1 - Introduction	1
Manual Scope and Organization	1
Manual Scope.....	1
Manual Organization.....	1
Conventions Used in this Manual	1
Chapter 2 - Overview	3
Introduction.....	3
Features.....	3
Applications.....	3
Board Description	4
Architecture	5
Digital I/O	5
Module Strobe Signals	5
Internal Clock Source.....	6
Module Strobed Output/Input Handshaking	6
GX5701: Digital Input Latch (DIL) Module.....	6
GX5702: Digital Output Latch (DOL) Module	8
GX5704: Digital Power Output (DPO) Module.....	10
GX5709: RS422 Port I/O	12
GX5711: Bidirectional LVDS-TTL Converter Module.....	14
GX5712: Bidirectional RS422-TTL Converter Module	15
Specifications.....	17
TTL I/O Channels specifications	17
Power Requirements	17
Environmental	17
Physical	17
Chapter 3 - Installation and Connections	19
Getting Started	19
Packing List	19

Unpacking and Inspection.....	19
System Requirements.....	19
Installation of the GXPIO Software.....	20
Setup Maintenance Program	20
Overview of the GXPIO Software	21
Installation Folders	21
Configuring Your PXI System using the PXI/PCI Explorer.....	22
Board Installation.....	23
Before you Begin	23
Electric Static Discharge (ESD) Precautions	23
Installing a Board	23
Plug & Play Driver Installation.....	25
Removing a Board	25
GXPIO Driver Files Description.....	26
Driver File and Virtual Panel	26
Interface Files.....	26
GXPIO On-line Help and Manual.....	26
ReadMe File.....	26
Example Programs	27
Connectors and Jumpers	29
JP1 – Chassis Ground Jumper.....	30
JB1-JB4 – Factory Installed Jumpers.....	30
Connections	30
J1: GX5733 with GX5701 Module Port 0 Connector.....	31
J1: GX5733 with GX5702 or GX5704 Module Port 0 Connector.....	32
J1: GX5733 with GX5709 Module Port 0 Connector.....	33
J1: GX5733 with GX5711 or GX5712 Module Port 0 Connector.....	34
J2-J4 – GX5733 Ports 1-3 Digital I/O Connectors	35
Connectors and Accessories	36
Chapter 4 - Instrument Software Panel.....	37
Virtual Panel Description.....	37
Virtual Panel Initialize Dialog	38
Virtual Panel Digital I/O Page	39
Virtual Panel Advanced Page	40
Virtual Panel About Page.....	41
Chapter 5 - Programming the Board	43
The GXPIO Driver.....	43

Programming Using C/C++ Tools	43
Programming Using Visual Basic.....	43
Programming Using Pascal/Delphi	44
Programming GXPIO Boards Using ATEasy®	44
Using the GXPIO driver functions.....	44
Initialization, HW Slot Numbers and VISA Resource	44
Board Handle	45
Reset.....	46
Error Handling	46
Driver Version.....	46
Panel	46
Programming Examples.....	46
Distributing the Driver	46
Sample Programs	47
Sample Program Listing	48
Chapter 6 - Functions Reference.....	57
Introduction.....	57
GX5733 Functions	58
Gx5733GetBoardSummary.....	61
Gx5733GetInternalClock.....	62
Gx5733GetInternalClockEnable	63
Gx5733GetPort	64
Gx5733GetPortBit	65
Gx5733GetPortByte.....	66
Gx5733GetPortByteDirection.....	67
Gx5733GetPortDirection	68
Gx5733GetPortWord	69
Gx5733GetStrobeToPxiTriggerBusState	70
Gx5733GetStrobeToPxiTriggerLineState.....	71
Gx5733Initialize	72
Gx5733InitializeVisa	73
Gx5733ModuleBufferClear	74
Gx5733ModuleBufferGetMode.....	75
Gx5733ModuleBufferGetState	76
Gx5733ModuleBufferReadData	77
Gx5733ModuleBufferSetFlagMode	78
Gx5733ModuleBufferSetMode	79

Gx5733ModuleBufferTest	80
Gx5733ModuleBufferWriteData	81
Gx5733ModuleGetDirection	82
Gx5733ModuleGetDirectionSource	83
Gx5733ModuleGetExtStrobeEnablePolarity	84
Gx5733ModuleGetExtStrobeEnableState.....	85
Gx5733ModuleGetExtStrobePolarity	86
Gx5733ModuleGetExtStrobeState.....	87
Gx5733ModuleGetHandshakeOutputMode	88
Gx5733ModuleGetInternalStrobeSource.....	89
Gx5733ModuleGetOperationMode	90
Gx5733ModuleGetOutputMode	91
Gx5733ModuleGetOutputStrobeWidth	92
Gx5733ModuleGetPort.....	93
Gx5733ModuleGetPxiTriggerBus	94
Gx5733ModuleGetPxiTriggerLine	95
Gx5733ModuleGetRevision	96
Gx5733ModuleGetState	97
Gx5733ModuleGetStrobeSource	98
Gx5733ModuleGetTermination.....	99
Gx5733ModuleGetType	100
Gx5733ModuleGetVThreshold	101
Gx5733ModuleHalt	102
Gx5733ModuleRun	103
Gx5733ModuleSetDirection	104
Gx5733ModuleSetDirectionSource	105
Gx5733ModuleSetExtStrobeEnablePolarity.....	106
Gx5733ModuleSetExtStrobePolarity.....	107
Gx5733ModuleSetHandshakeOutputMode	108
Gx5733ModuleSetInternalStrobeSource	109
Gx5733ModuleSetOperationMode.....	110
Gx5733ModuleSetOutputMode.....	111
Gx5733ModuleSetOutputStrobeWidth.....	112
Gx5733ModuleSetPort	113
Gx5733ModuleSetPxiTriggerBus.....	114
Gx5733ModuleSetPxiTriggerLine.....	115
Gx5733ModuleSetStrobeSource.....	116

Gx5733ModuleSetVThreshold	117
Gx5733ModuleStep	118
Gx5733ModuleSetTermination	119
Gx5733Panel.....	120
Gx5733Reset.....	121
Gx5733ResetPort	122
Gx5733SetInternalClock.....	123
Gx5733SetInternalClockEnable.....	124
Gx5733SetPort.....	125
Gx5733SetPortBit.....	126
Gx5733SetPortByte	127
Gx5733SetPortByteDirection	128
Gx5733SetPortDirection.....	129
Gx5733SetPortWord.....	130
Gx5733SetStrobeToPxiTriggerBusState	131
Gx5733SetStrobeToPxiTriggerLineState	132
GxPioGetDriverSummary.....	133
GxPioGetErrorString	134
Index	137

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 GX5733 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 GX5733 driver (referred in this manual **GXPIO**). Therefore, good understanding of programming development tools and languages may be necessary.

Manual Organization

The GX5733 manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX5733 manual. Lists all the supported board and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX5733 features, board description, its architecture, specifications and the panel description and operation.
Chapter 3 –Installation and Connections	Provides instructions on how to install the GX5733 Board and its accompanying GXPIO software.
Chapter 4 – Instrument Software Panel	Provides instruction how to open and use the instrument front panel application in order to view and control the instrument settings.
Chapter 5 – Programming the Board	Provides a listing of GX5733 driver files, general purpose/generic driver functions, and programming methods. Discusses various supported operating systems and development tools.
Chapter 6 – Functions Reference	Contains a listing of the general GX5733 functions. Each function is described along with its syntax, parameters, and special programming comments. Samples are given for each function.

Conventions Used in this Manual

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

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

Chapter 2 - Overview

Introduction

The GX5733 is a 3U PXI instrument board that provides four ports of 32 channels parallel input and output for a total of 128 channels. The four ports are composed of three Digital I/O ports and one port that can be configured using variety of I/O Modules interfacing different UUTs.

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

The I/O Modules port expands the ability of the GX5733 by mounting different I/O Modules. Those I/O Module enable the GX5733 to acquire or transmit 32 channels vectors each by using high-speed FIFO as well as interfacing different voltage levels, compare data on the fly and output high power signals to name a few.

All installed I/O Modules can be programmed to run using internal or external strobe. When using the internal strobe the I/O Module can use any of eight available PXI signals or two internal programmable clock signals.

Features

The GX5733 is a Digital Input or Output card with:

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

Applications

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

Board Description

The GX5733 is a 3U PXI instrument board that provides four ports of 32 channels parallel input and output for a total of 128 channels. The four ports are composed of three Digital I/O ports and one port that can be configured using variety of I/O Modules interfacing different UUTs.

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

The I/O Modules port expands the ability of the GX5733 by mounting different I/O Modules. Those I/O Modules enable the GX5733 to acquire or transmit 32 channels vectors by using high-speed FIFO as well as interfacing different voltage levels, compare data on the fly and output high power signals to name a few.

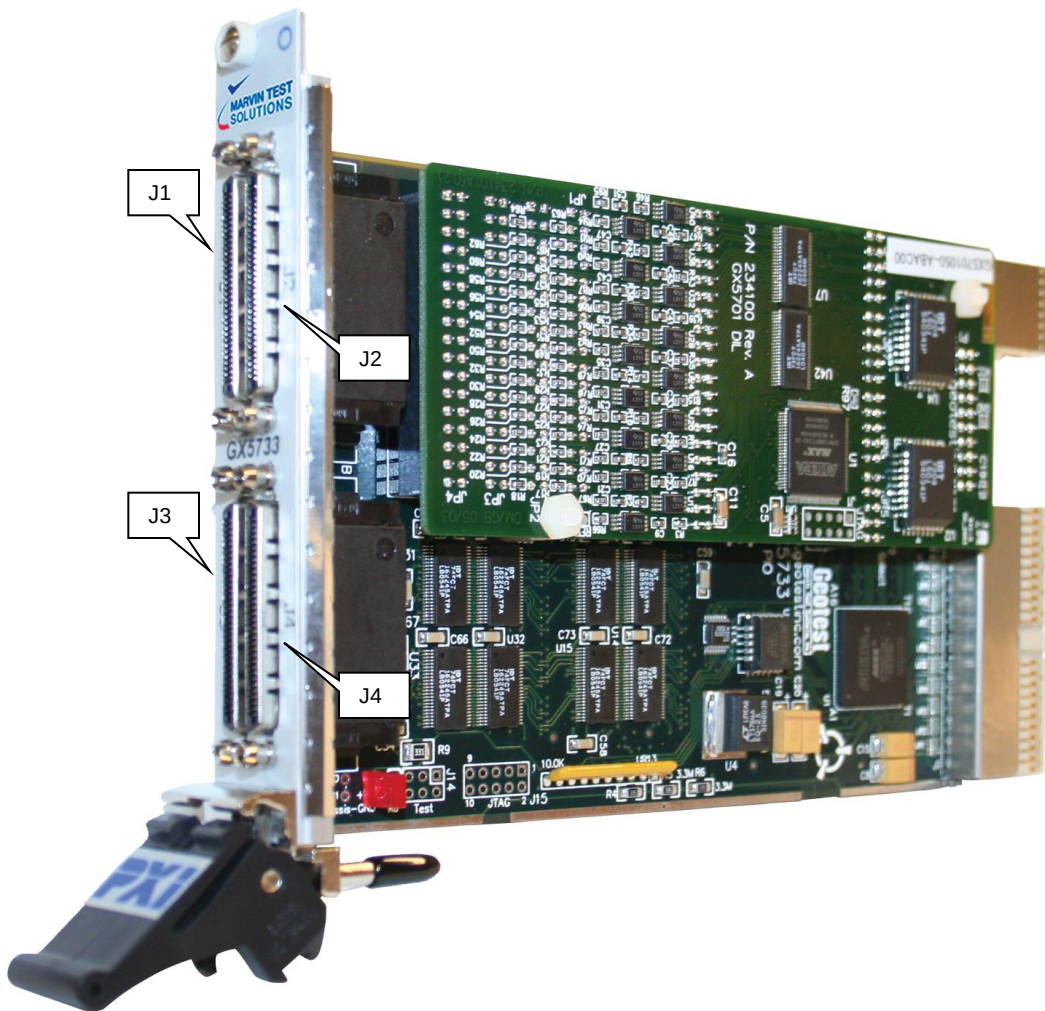


Figure 2-1: GX5733 Board Side View with no I/O Modules installed

Architecture

Digital I/O

The GX5733 provides 96 digital Inputs or Outputs. The 96 TTL channels' direction is programmable and can be set for Input or Output in groups of eight. The GX5733 has no on-board memory and it uses software driver to set or get the channels level and direction. Figure 2-2 shows a typical I/O channel block diagram:

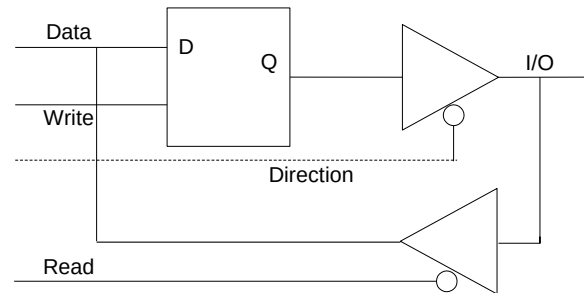


Figure 2-1: Typical I/O Channel

The GX5733 consist of three 32-bit parallel input/output ports. Each port interfaces through a 68 pins SCSI connector (J2 to J4).

Module Strobe Signals

The GX5733 can have one Module installed. Installed module has an independent set of strobe signals controlled by software and hardware. The module strobe source can be set using software to one of the following sources:

- Internal Clock0
- Internal Clock1
- External Strobe Input (J1 pin 67)
- External Strobe Enable (J1 pin 68)

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

Figure 2-2 is a block diagram for the GX5733 Module's Strobe Signals:

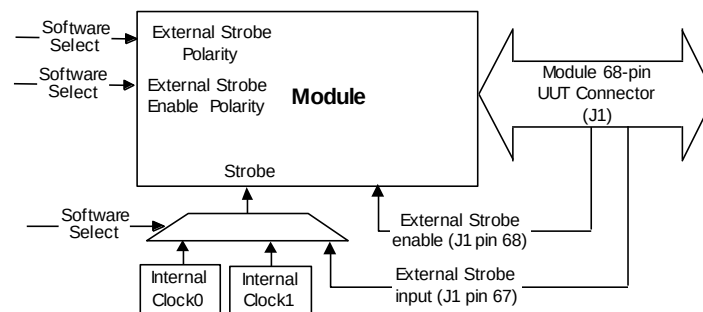


Figure 2-2: Module Strobe Signals

Internal Clock Source

The GX5733 contains two internal clock sources that can be used to drive any of the installed modules. Each of the internal clocks can be configured to use the 10MHz PXI clock or 33 MHz PCI Bus clock.

Figure 2-3: is a block diagram for the GX5733 Internal Clock Source:

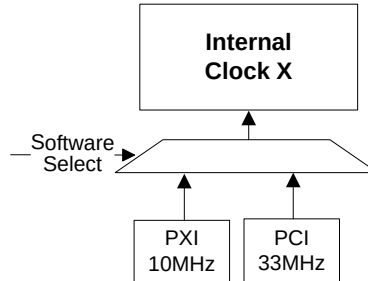


Figure 2-3: GX5733 Internal Clock Source

Module Strobed Output/Input Handshaking

The GX5733 can have one Module installed. Installed module can use Direct Output/Input or Strobed Output/Input with handshaking.

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

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

GX5701: Digital Input Latch (DIL) Module

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

Features:

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

Figure 2-4: GX5701 Strobed Input with Handshaking timing diagram demonstrates the Handshaking process. Each signal number (1 to 9) is explained in table. The description of each timing segment (numbered 1-9) explained in Table 2-1.

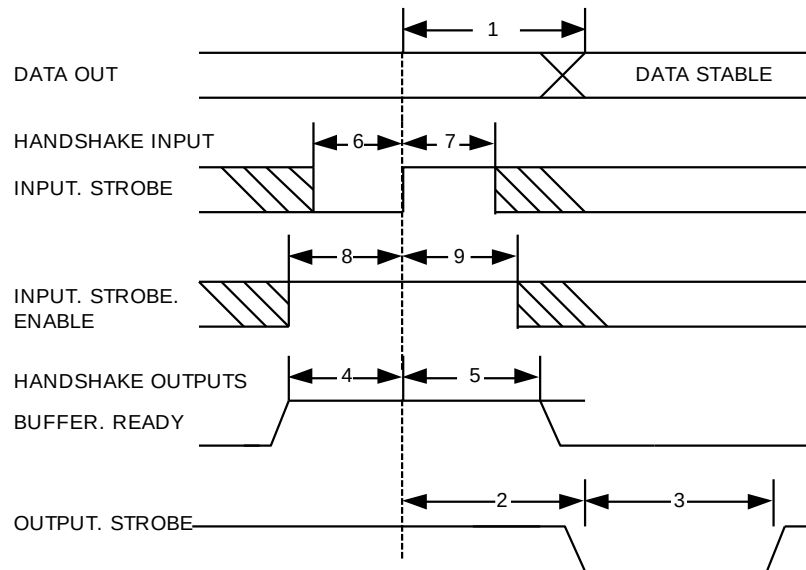


Figure 2-4: GX5701 Strobed Input with Handshaking timing diagram

The following table describes each timing segment (numbered 1-7):

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

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

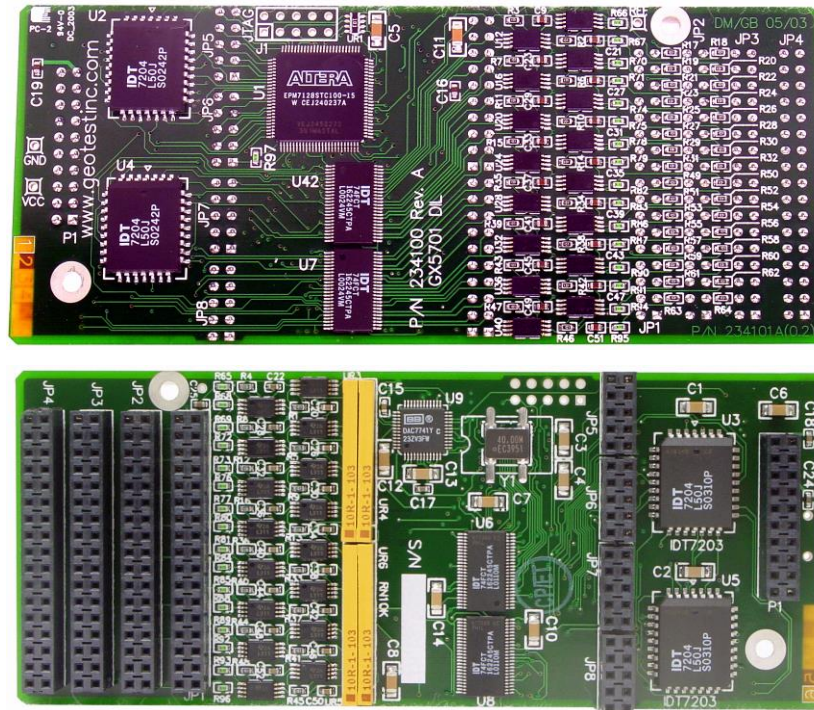


Figure 2-5: GX5701 Digital Input Latch (DIL) Module

GX5702: Digital Output Latch (DOL) Module

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

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

Features:

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

Figure 2-6: GX5702 Strobed Output with Handshaking timing diagram demonstrates the Handshaking process. The description of each timing segment (numbered 2-7) explained in Table 2-2.

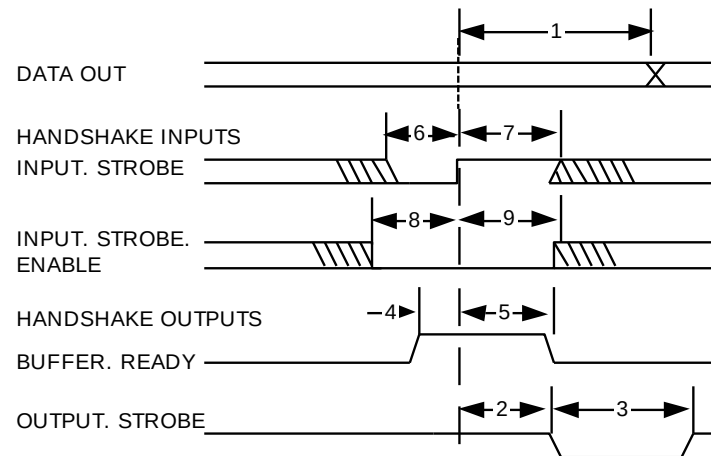


Figure 2-6: GX5702 Strobed Output with Handshaking timing diagram

The following table describes each timing segment (numbered 2-7):

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

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

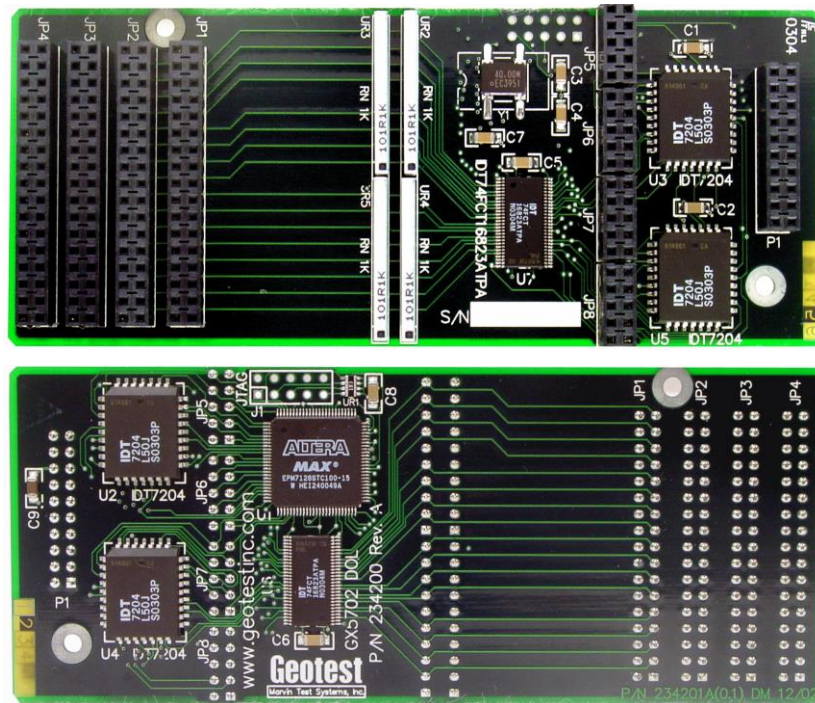


Figure 2-7: GX5702 Digital Output Latch (DOL) Module

GX5704: Digital Power Output (DPO) Module

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

The GX5704 can be triggered using internal or external strobe (software controlled). When using external strobe the module supports Strobed Input with Handshaking

Features:

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

Figure 2-8: GX5704 Strobed Output with Handshaking timing diagram demonstrates the Handshaking process. The description of each timing segment (numbered 2-7) explained in Table 2-3.

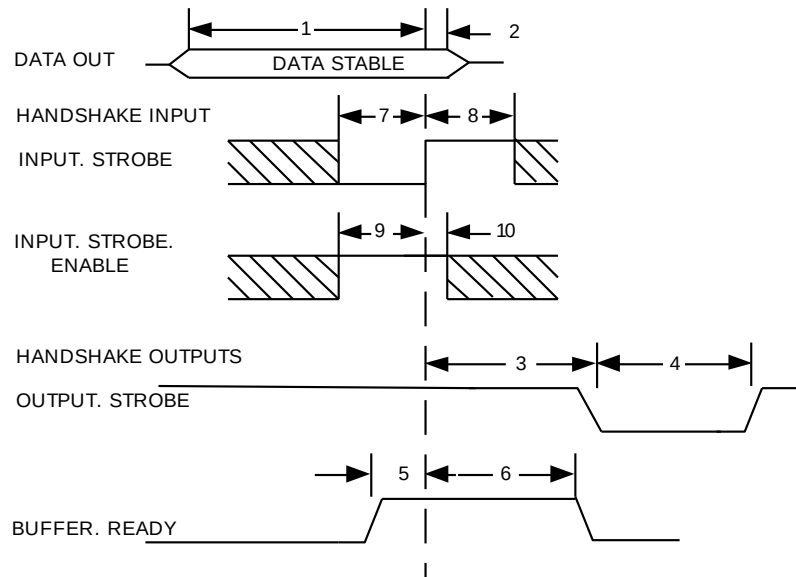


Figure 2-8: GX5704 Strobed Output with Handshaking timing diagram

The following table describes each timing segment (numbered 2-7):

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

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

GX5709: RS422 Port I/O

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

Features:

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

Architecture

The RS422 I/O channels' direction is programmable and can be set for Input or Output in groups of eight. The GX5709 Module uses software driver to set or get the channels logic level and direction. Figure 2-9 shows a typical I/O channel block diagram:

Block Diagram

Figure 2-9: Typical RS422 I/O Channel Block Diagram

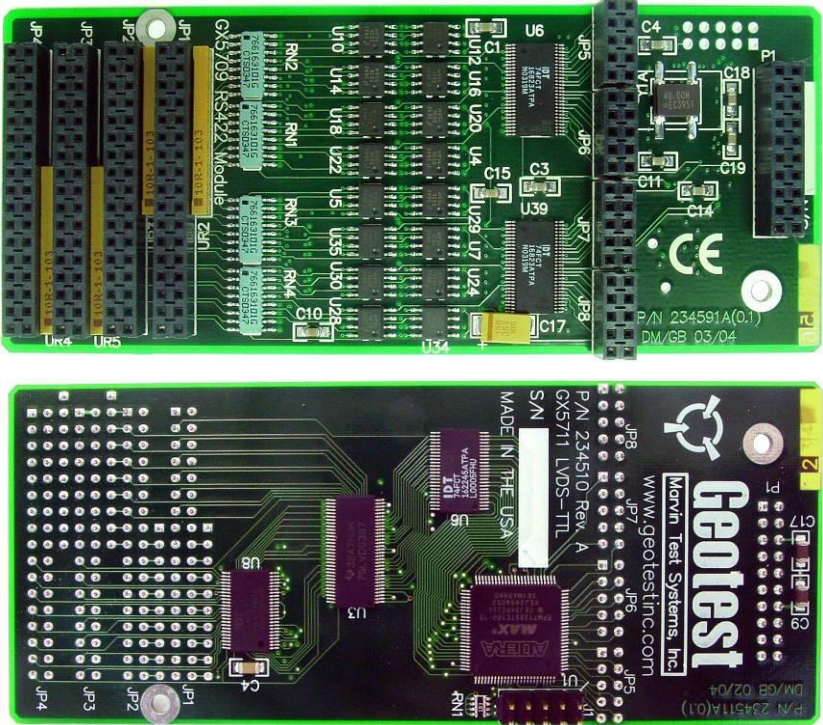


Figure 2-10: GX5709: RS422 Port I/O Module

GX5711: Bidirectional LVDS-TTL Converter Module

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

Features:

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

Block Diagram

Figure 2-11: GX5711 Typical Bidirectional LVDS-TTL channel Block Diagram

Block Diagram

Figure 2-13: GX5712 Typical Bidirectional RS422-TTL channel Block Diagram

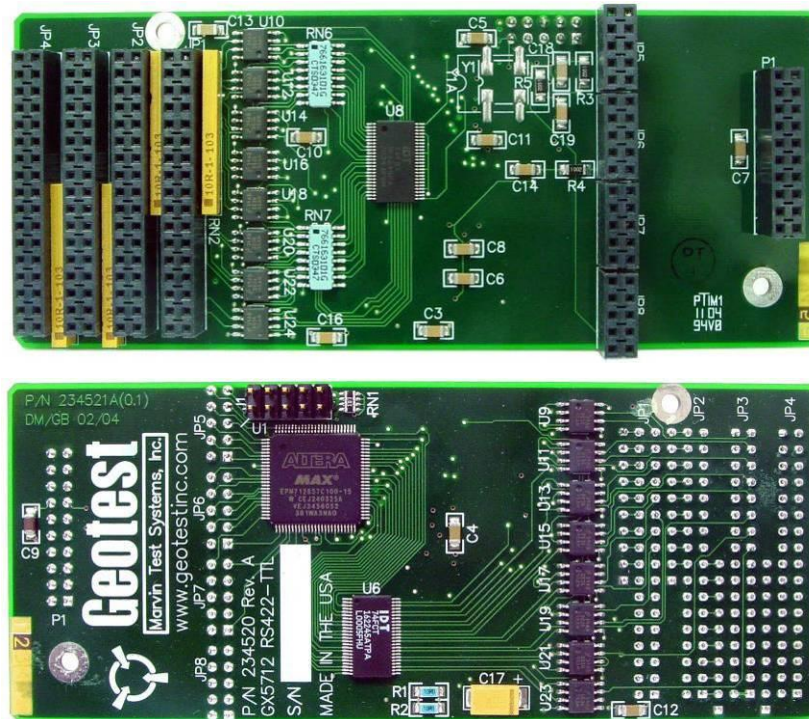


Figure 2-14: GX5712 Bidirectional RS422-TTL Converter Module

Specifications

The following table outlines the specifications of the GX5733.

TTL I/O Channels specifications

TTL I/O Levels (LVTTL)	
Input Low Voltage Level	2 V Min.
Input High Voltage Level	0.8V Max
Input Hysteresis	100mV Typical
Output Low Voltage	0.55V Max
Output High Voltage	122V Min
Source current	24mA Max
Sink current	24mA Max
Number of TTL I/O Channels	96

Power Requirements

3.3 VDC Current	0.5 A
5 VDC Current	1.3 A

Environmental

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

Physical

Size	3U PXI
Weight	18 oz.

Chapter 3 - Installation and Connections

Getting Started

This section includes general hardware installation procedures for the GX5733 board and installation instructions for the GX5733 (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
GX5733 Driver Installation	This Chapter
Programming the boards	Chapter 5
GX5733 Reference Functions	Chapter 6

Packing List

All GX5733 boards have the same basic packing list, which includes:

1. GX5733 Board
2. GX5733 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 GX5733 Instrument board is designed to run on PXI compatible computer running Windows 2000, XP and above. In addition, Microsoft Windows Explorer version 4.0 or above is required to view the online help.

The board requires one unoccupied 6U PXI bus slot.

Installation of the GXPIO Software

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

1. Insert the Marvin Test Solutions CD-ROM and locate the **GXPIO.EXE** setup program. If your computer's Auto Run is configured, when inserting the CD a browser will show several options. Select the Marvin Test Solutions Files option and then locate the setup file. If Auto Run is not configured you can open the Windows explorer and locate the setup files (usually located under \Files\Setup folder). You can also download the file from Marvin Test Solutions' web site (www.marvintest.com), 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 shown 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 GX5733 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 **Gx5733Initialize** and **Gx5733InitializeVisa** functions.
4. **Save your work.** PXI Explorer saves the configuration to the following files located in the Windows folder: PXISYS.ini, PXIeSYS.ini and GxPxiSys.ini. Click on the **Save** button to save your changes. The PXI/Explorer will prompt you to save the changes if changes were made or detected (an asterisk sign '*' in the caption indicated changes).

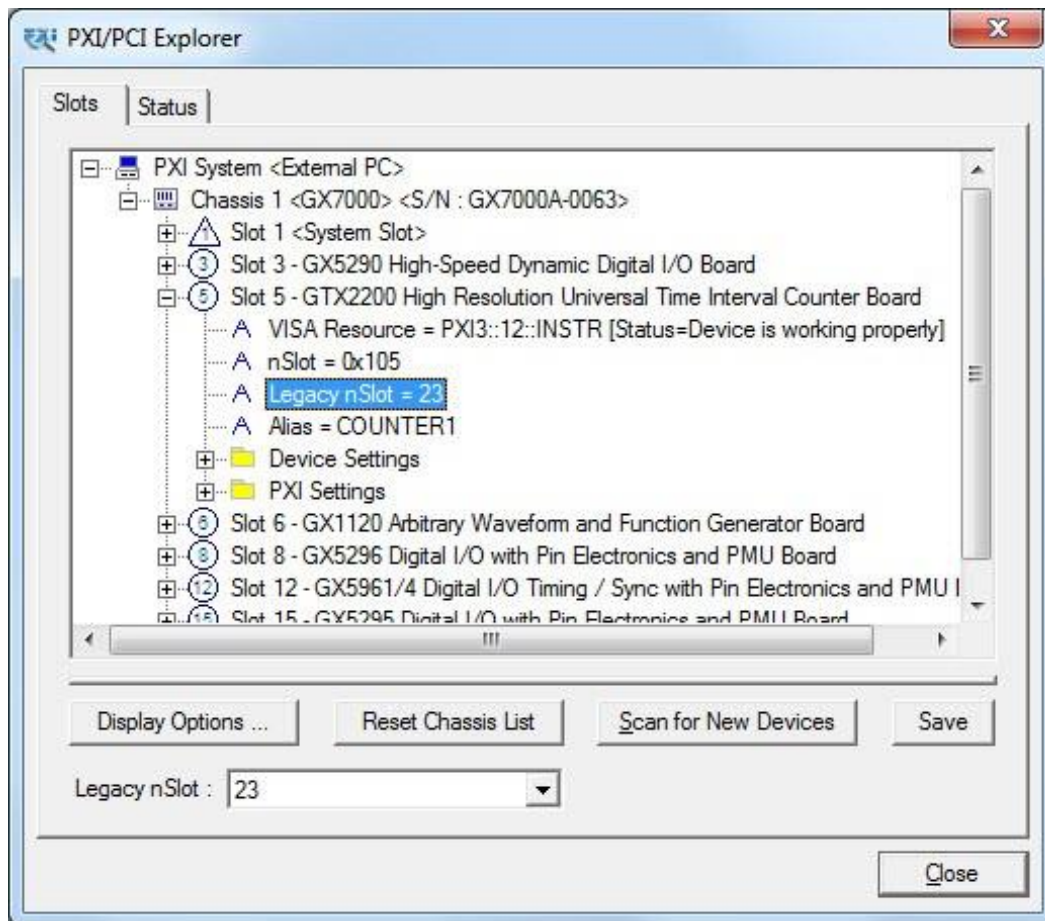


Figure 3-1: PXI/PCI Explorer

Board Installation

Before you Begin

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

Electric Static Discharge (ESD) Precautions

To reduce the risk of damage to the GX5733 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).

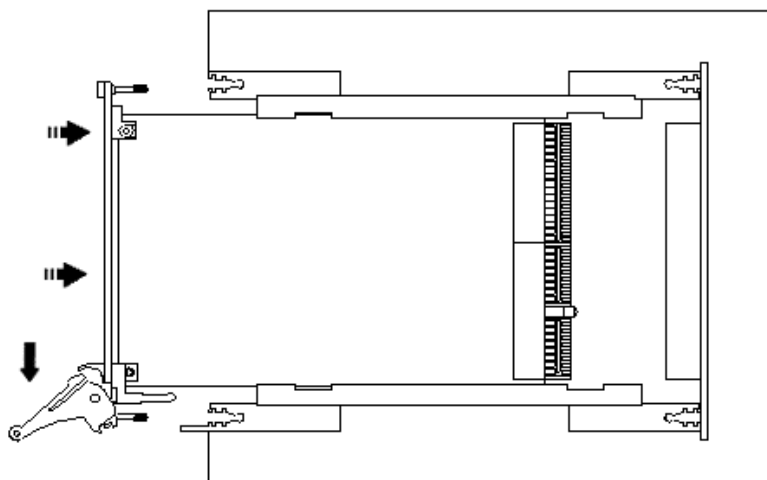


Figure 3-1: Ejector handles position during module insertion

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

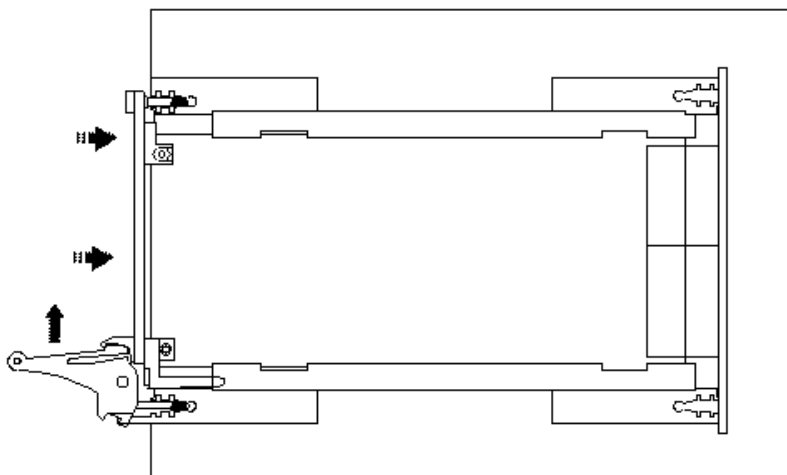


Figure 3-2: Ejector handles position after module insertion

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

Plug & Play Driver Installation

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

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

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

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

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

Removing a Board

Remove the board as follows:

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

GXPIO Driver Files Description

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

Driver File and Virtual Panel

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

Interface Files

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

- GXPIO.H - header file for accessing the DLL functions using the C/C++ programming language. The header file compatible with the following 32 bit development tools:
 - Microsoft Visual C++, Microsoft Visual C++ .NET
 - Borland C++
- GXPIO.LIB and GXPIO64.LIB - Import library for GXPIO.DLL and GXPIO64.DLL (used when linking C/C++ 32/64 bit application).
- GXPIOBC.LIB - Import library for GXPIO.DLL (used when linking Borland C/C++ application that uses GXPIO.DLL).
- GXPIO.PAS - interface file to support Borland Pascal Borland Delphi.
- GXPIO.BAS - Supports Microsoft Visual Basic 4.0, 5.0 and 6.0.
- GXPIO.VB - Supports Microsoft Visual Basic .NET.
- GX5733.DRV - ATEasy driver File for GX5733.GXPIO Virtual Panel Program

GXPIO On-line Help and Manual

GXPIO.CHM – On-line version of the GX5733 User's Guide. The help file is provided in a Windows Compiled HTML help file (.CHM). The file contains information about the GX5733 board, programming reference and panel operation.

GX5733.PDF – On line, printable version of the GX5733 User's Guide in Adobe Acrobat format. To view or print the file you must have the reader installed. If not, you can download the Adobe Acrobat reader (free) from <http://www.adobe.com/>.

ReadMe File

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

Example Programs

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

Microsoft Visual C++ .NET example files:

GXPIOExampleC.cpp - Source file

- GXPIOExampleC.ico - Icon file
- GXPIOExampleC.rc - Resource file
- GXPIOExampleC.vcproj - VC++ .NET project file
- GXPIOExampleC.exe – 32 bit example executable
- GXPIOExampleC64.exe – 64 bit example executable

Microsoft Visual C++ 6.0 example files:

GXPIOExampleC.cpp - Source file

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

Borland C++ example files:

GXPIOExampleC.cpp - Source file

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

Microsoft Visual Basic .NET example files:

GxPioExampleVB.vb - Example form.

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

Microsoft Visual Basic 6.0 example files:

GxPioExampleVB6.frm - Example form

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

ATEasy driver and examples files (ATEasy Drivers directory):

Gx5733.prj - example project

- GX5733.sys - example system
- GX5733.prg - example program

Connectors and Jumpers

Figure 3-3 shows the available GX5733 board connectors and jumpers followed by their description:

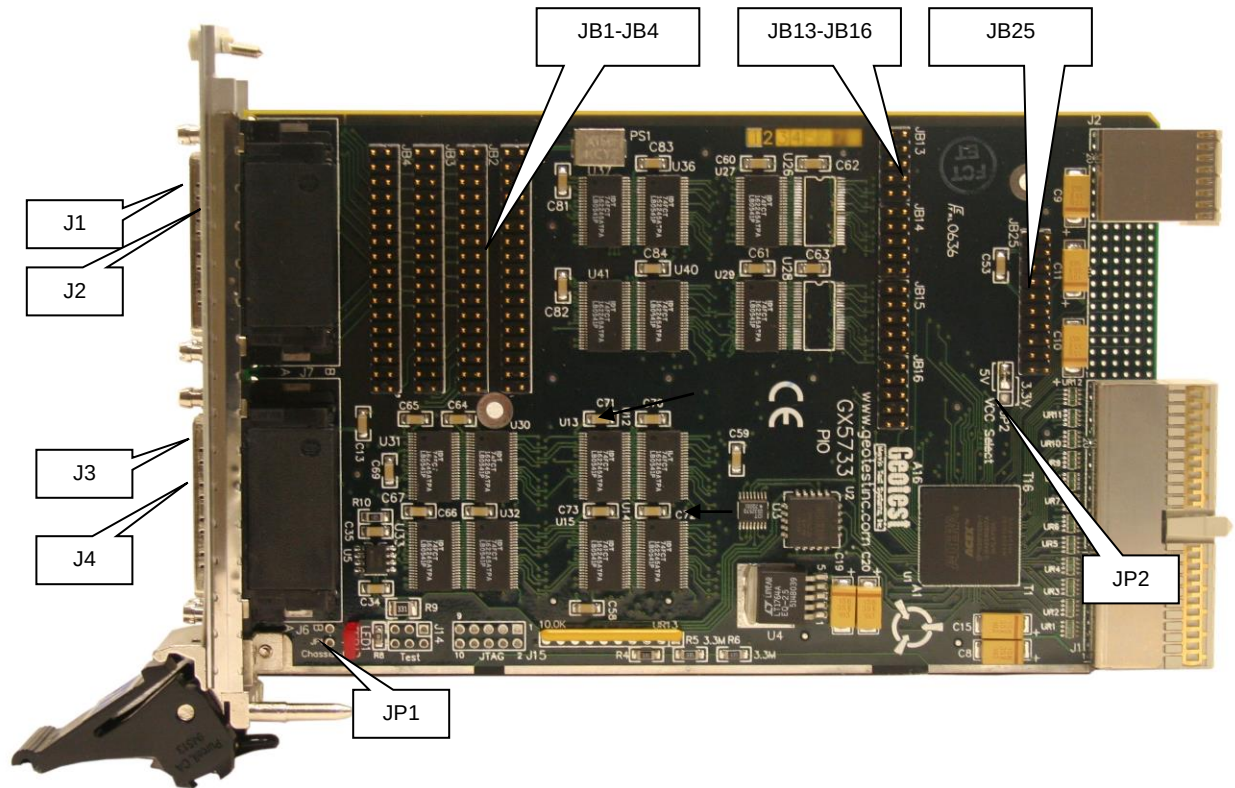


Figure 3-3: GX5733 Connectors and Jumpers

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

Connector/Jumpers	Description
J1	Port 0 digital I/O Connector
J2	Port 1 digital I/O Connector
J3	Port 2 digital I/O Connector
J4	Port 3 digital I/O Connector
JP1	Chassis ground jumper
JP2	VCC Select. Factory installed, Do Not Remove
JB - JB4, JB13 JB16, JB25	I/O Modules headers

JP1 – Chassis Ground Jumper

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

JB1-JB4 – Factory Installed Jumpers

The Jumper Blocks of JB1 through JB4 pertain to a the I/O module 0

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

Connections

Connections to the GX5733 may be made with four 68 pins SCSI male plug connector. Shielded cables with matching connectors are available from Marvin Test Solutions.

J1: GX5733 with GX5701 Module Port 0 Connector

The following are the connector signal pin assignments for GX5733 with GX5701 Module:

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

Table 3-1: J1 GX5733 with GX5701 Module Port 0 Connector

GND	Ground
IN 0 to IN 31	Input Data Lines
BufRdy	Buffer Ready Handshake line
OutStrb	Output Strobe Handshake line
XStrb	External Strobe input Handshake line
XStrbEn	External Strobe Enable Handshake line

J1: GX5733 with GX5702 or GX5704 Module Port 0 Connector

The following are the connector signal pin assignments for GX5733 with GX5702 or GX5704 Module:

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

Table 3-2: J1 GX5733 with GX5702 or GX5704 Module Port 0 Connector

GND	Ground
OUT 0 to OUT 31	Output Data Lines
BufRdy	Buffer Ready Handshake line
OutStrb	Output Strobe Handshake line
XStrb	External Strobe input Handshake line
XStrbEn	External Strobe Enable Handshake line

J1: GX5733 with GX5709 Module Port 0 Connector

The following are the connector signal pin assignments for GX5733 with GX5709 Module:

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

Table 3-3: GX5733 with GX5709 Module Port 0 Connector

I/O D0 to I/O D31 Bi-directional, Differential I/O channels data lines (0-31)

VCC +5V

GND Ground

J1: GX5733 with GX5711 or GX5712 Module Port 0 Connector

The following are the connector signal pin assignments for GX5733 with GX5711 or GX5712 Module:

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

Table 3-4: GX5733 with GX5711 or GX5712 Port 0 Connector

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

J2-J4 – GX5733 Ports 1-3 Digital I/O Connectors

The following are the connector signal pin assignments for GX5733 (Ports 1-3) Digital I/O module:

#	Signal	#	Signal	#	Signal	#	Signal
1	I/O n0	18	I/O n17	35	GND	52	GND
2	I/O n1	19	I/O n18	36	GND	53	GND
3	I/O n2	20	I/O n19	37	GND	54	GND
4	I/O n3	21	I/O n20	38	GND	55	GND
5	I/O n4	22	I/O n21	39	GND	56	GND
6	I/O n5	23	I/O n22	40	GND	57	GND
7	I/O n6	24	I/O n23	41	GND	58	GND
8	I/O n7	25	I/O n24	42	GND	59	GND
9	I/O n8	26	I/O n25	43	GND	60	GND
10	I/O n9	27	I/O n26	44	GND	61	GND
11	I/O n10	28	I/O n27	45	GND	62	GND
12	I/O n11	29	I/O n28	46	GND	63	GND
13	I/O n12	30	I/O n29	47	GND	64	GND
14	I/O n13	31	I/O n30	48	GND	65	GND
15	I/O n14	32	I/O n31	49	GND	66	GND
16	I/O n15	33	VCC	50	GND	67	VCC
17	I/O n16	34	GND	51	GND	68	GND

Table 3-5: GX5733 Ports 1-3 Digital I/O Connectors

I/O n Bi-directional, I/O Port number, n can be 1,2 or 3
VCC +5V
GND Ground

Connectors and Accessories

The following accessories are available from Marvin Test Solutions for GX5733 switching board.

Part / Model Number	Description
GX95401	Extra GX5733 User Manual
GT95014	Connector Interface for GX5xxx/GX5733, SCSI to 100 Mil Grid, Single Ended
GT95021	2' shielded cable for GX5xxx/GX5733 (68 Pin)
GT95022	3' shielded cable for GX5xxx/GX5733 (68 Pin)
GT95028	10' shielded cable for GX5xxx/GX5733 (68 Pin)
GT95031	6' shielded cable for GX5xxx/GX5733 (68 Pin)

Chapter 4 - Instrument Software Panel

Calling the **GX5733Panel** function will display the instrument front panel in a window. The panel can be used to initialize and to control the board interactively. The panel function may be used by the application to allow the user to directly interact with the board. The **Gx5733Panel** function is also used by the GXPIOPANEL.EXE or GXPIOPANEL64.EXE (with the /5733 command line argument) panel programs that is supplied with this package and provides a standalone Windows application that displays the instrument panel.

Virtual Panel Description

The GX5733 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 **GX5733 Panel** from the **Marvin Test Solutions, GXPIO** menu under the **Start** menu. The GX5733 virtual panel opens as shown here:

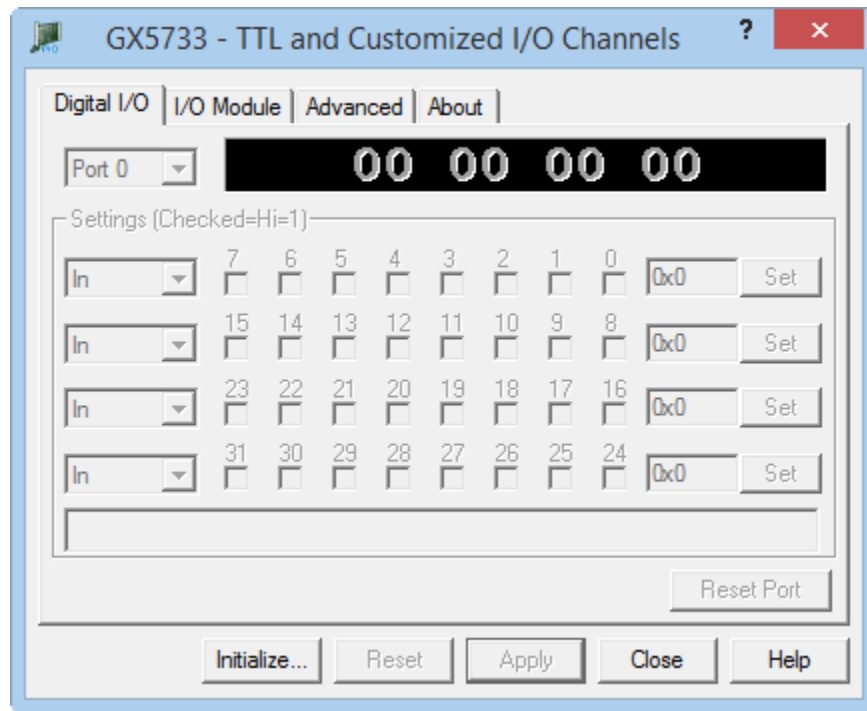


Figure 4-1: GX5733 Virtual Panel

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

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

Apply – Applies changed settings to the board.

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

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

Virtual Panel Initialize Dialog

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

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

1. **Use Marvin Test Solutions' HW** – This is the device driver installed by the setup program and is the default driver. When selected, the **Slot Number** list displays the available **GX5733** 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 **Gx5733Initialize** function which is used to initialize the driver for the specified board.

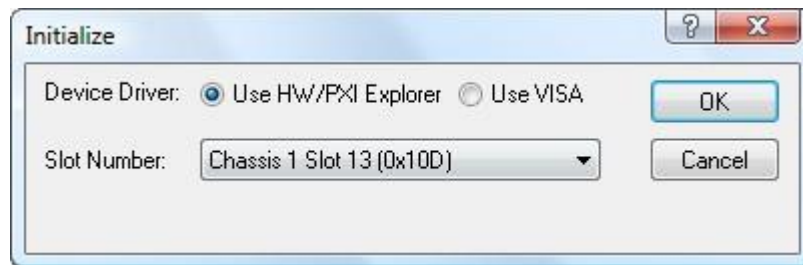


Figure 4-2: Initialize Dialog Box using Marvin Test Solutions' HW driver

2. **Use VISA** – This is a third party device driver usually provided by National Instrument (NI-VISA). When selected, the **Resource** list displays the available boards installed in the system and their VISA resource address. The chassis, slots, devices and their resources are also displayed by the VISA resource manager, **Measurement & Automation** (NI-MAX) and by Marvin Test Solutions **PXI/PCI Explorer**. The following figure shows PXI9::13::INSTR as the VISA resource (PCI bus 9 and Device 13). This is a VISA resource string argument (*szVisaResource*) which is passed by the panel when calling the driver **Gx5733InitializeVisa** function which initializes the driver for the specified board.

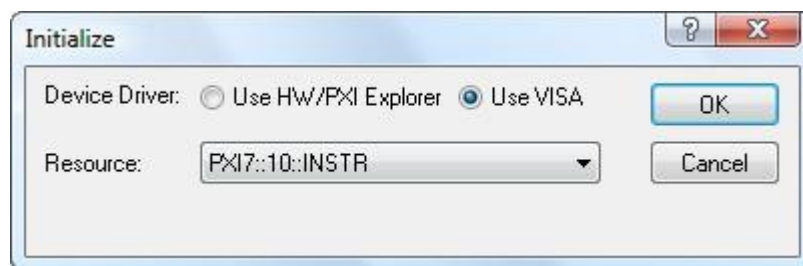


Figure 4-3: Initialize Dialog Box using VISA resources

Virtual Panel Digital I/O Page

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

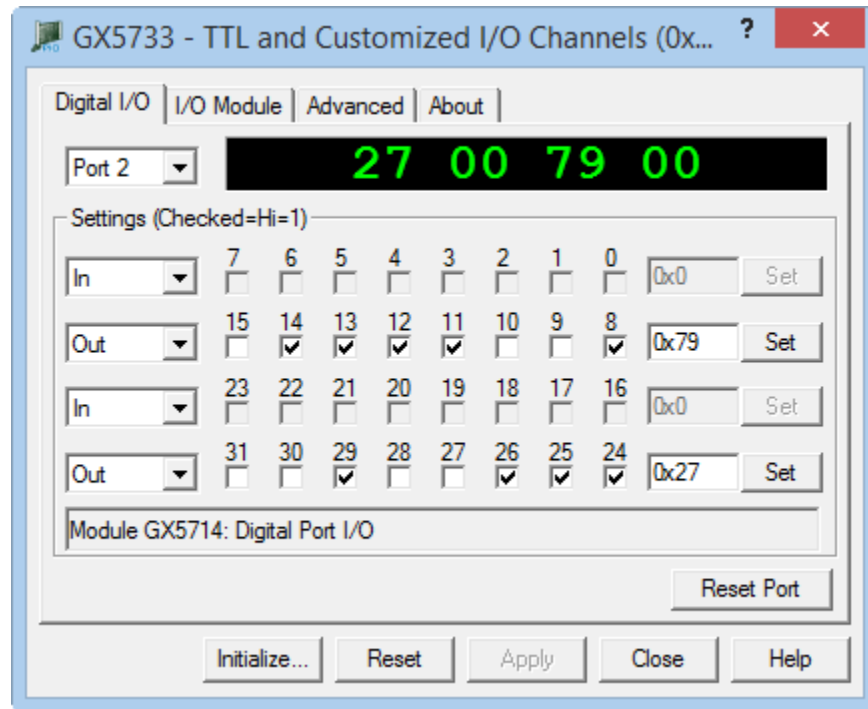


Figure 4-4: GX5733 Virtual Panel – Digital I/O page

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

Port drop-down box (Ports 0 –3): Selects a port to display its current settings in this page. All other controls in this page show the selected port settings.

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

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

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

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

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

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

Virtual Panel Advanced Page

Clicking on the **Advanced** tab will show the **Advanced page** as shown in Figure 4-5:

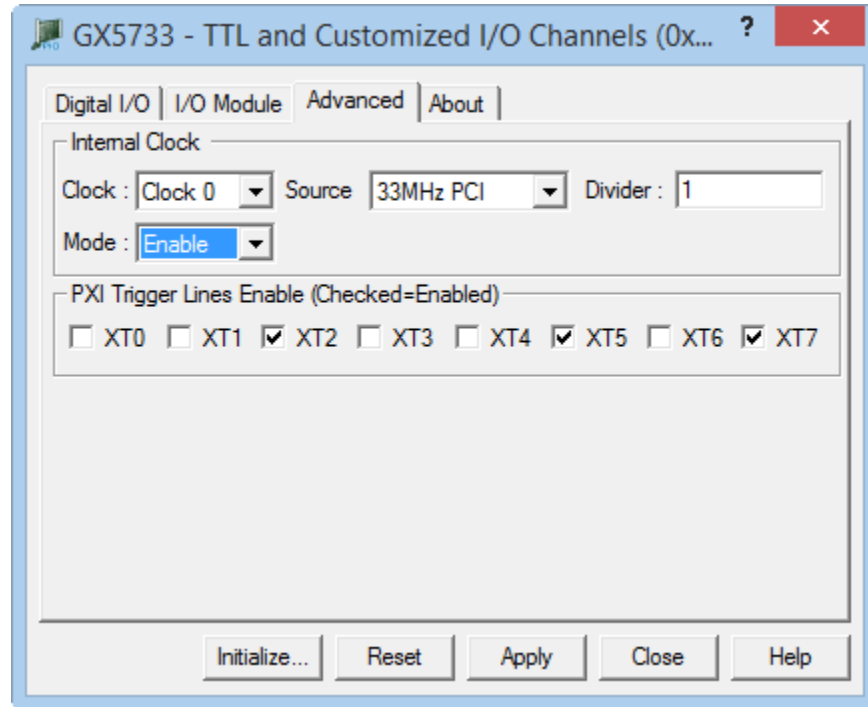


Figure 4-5: GX5733 Virtual Panel – Advanced Page

The following controls are shown in the Advanced page:

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

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

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

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

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

Virtual Panel About Page

Clicking on the **About** tab will show the **About page** as shown in Figure 4-6:

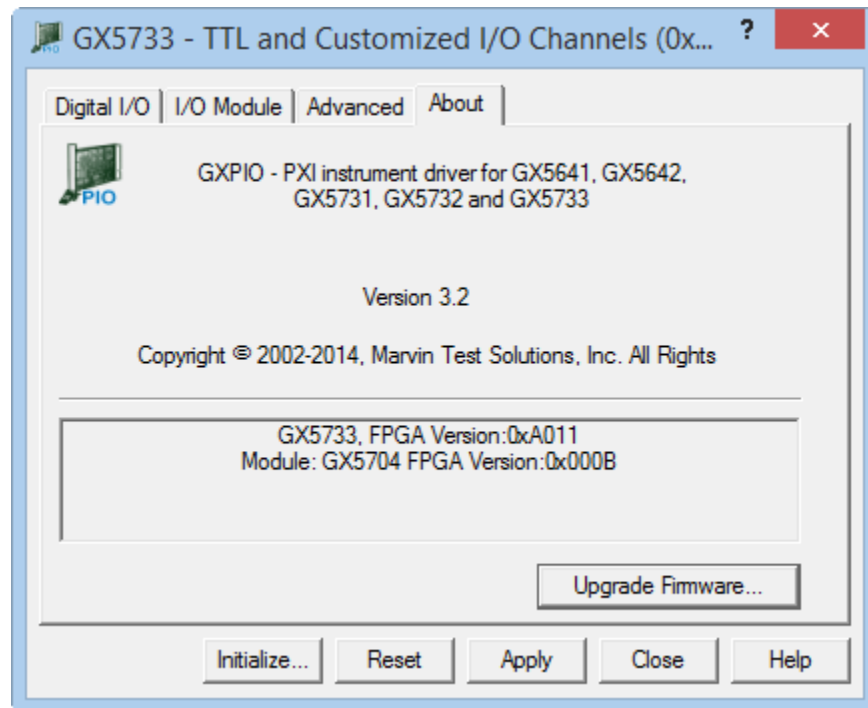


Figure 4-6: GX5733 Virtual Panel – About Page

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

Chapter 5 - Programming the Board

This chapter contains information about how to program the board using the GXPIO driver. The GXPIO driver contains functions to initialize, reset, and control the board. A brief description of the functions, as well as how and when to use them, is included in this chapter. Chapter 5 and the specific instrument User's Guide contain a complete and detailed description of the available programming functions.

The GXPIO driver supports many development tools. Using these tools with the driver is described in this chapter. In addition, the GXPIO directory contains examples written for these development tools. Refer to Chapter 3 for a list of the available examples.

An example using the DLL driver with Microsoft Visual C++ 6.0 is included at the end of this chapter. Since the driver functions and parameters are identical for all operating systems and development tools, the example can serve as an outline for other programming languages, programming tools, and other GXPIO driver types.

The GXPIO Driver

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

The DLLs can be used with various development tools such as Microsoft Visual C++, Borland C++ Builder, Microsoft Visual Basic, Borland Pascal or Delphi, ATEasy and more. The following paragraphs describe how to create an application that uses the driver with various development tools. Refer to the paragraph describing the specific development tool for more information.

Programming Using C/C++ Tools

The following steps are required to use the GX5733 driver with C/C++ development tools:

- Include the GXPIO.H header file in the C/C++ source file that uses the GX5733 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 GX5733 as required by the application.
- Build the project.
- Run, test, and debug the application.

Programming Using Visual Basic

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

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

Programming Using Pascal/Delphi

To use the driver with Borland Pascal or Delphi, the user must include the GXPIO.PAS to the project. The GXPIO.PAS file contains a **unit** with function prototypes for the DLL functions. Include the GX5733 unit in the **uses** statement before making calls to the GX5733 functions.

Programming GXPIO Boards Using ATEasy®

The GXPIO package is supplied with a separate ATEasy driver for each of board types. The **GX5733.drv** ATEasy driver uses the GXPIO.DLL to program the board. In addition, each driver is supplied with an example that contains a program and a system file pre-configured with the ATEasy driver. Use the driver shortcut property page from the System Drivers sub-module to change the PCI slot number before attempting to run the example.

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

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

- The *nHandle* parameter is omitted. The driver handles this parameter automatically. ATEasy uses driver logical names instead i.e. PIO1, PIO2 for GX5733.
- 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 GX5733 board uses the Gx5733Xxxx 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 **Gx5733Initialize** to initialize the board's driver using HW and **Gx5733InitializeVisa** to initialize using VISA. The following describes the two different methods used to initialize:

1. **Marvin Test Solutions's 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 **Gx5733Initialize(nSlot, pnHandle, pnStatus)** function. The function initializes the driver for the board at the specified PXI slot number (*nSlot*) and returns boards handle. The **PXI/PCI Explorer** applet in the Windows Control Panel displays the PXI slot assignments. You can specify the *nSlot* parameter in the following way:
 - A combination of chassis number (chassis # x 256) with the chassis slot number, e.g. 0x105 for chassis 1 and slot 5. The chassis number can be set by the **PXI/PCI Explorer** applet.

- Legacy nSlot is used by earlier versions of HW/VISA. The slot number contains no chassis number and can be changed using the **PXI/PCI Explorer** applet: 23 in this example.

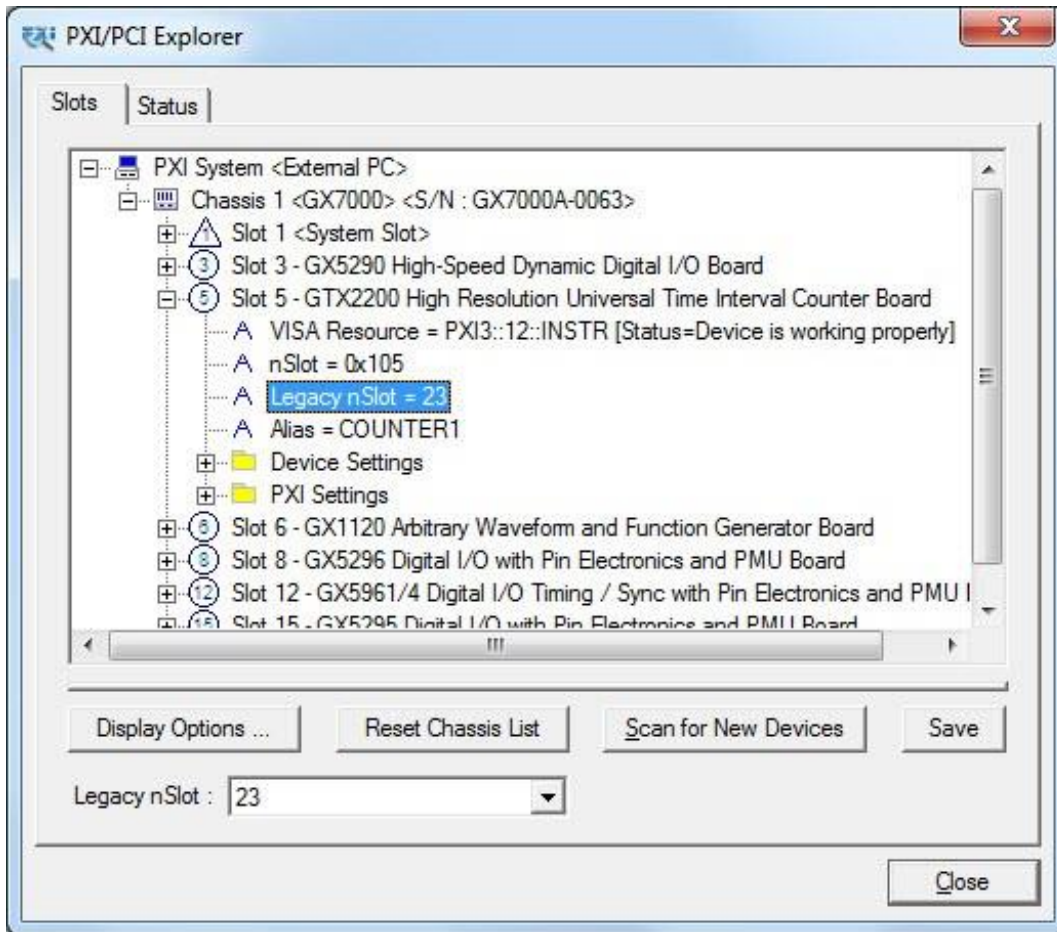


Figure 5-1: PXI/PCI Explorer

- 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 **Gx5733InitializeVisa** (*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 **Gx5733Initialize** and the **Gx5733InitializeVisa** functions return a handle that is required by other driver functions in order to program the board. This handle is usually saved in the program as a global variable for later use when calling other functions. The initialize functions do not change the state of the board or its settings.

Reset

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

Error Handling

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

Driver Version

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

Panel

Calling the **Gx5733Panel** 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 **Gx5733Panel** function is also used by the GXPIOPANEL.EXE or GXPIOPANEL64.EXE panel programs that is supplied with this package and provides a standalone Windows application that displays the instrument panel.

Programming Examples

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

Distributing the Driver

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

Sample Programs

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

To run, Enter the following command line:

GXPIOEX <PciSlot> <specific_command>

Where:

<board_type>	Board type: 5731=GX5731, 5732=GX5732, 5733=GX5733
<slot>	PCI/PXI Explorer slot number where the board resides.
<port counter numbers>	The digital i/o port number (0-6) or the counter number (0-3).
<operation>	Operation code: RD=read port direction WD=write port direction RP=read port value WP=write port value RC=read counter WC=write counter
<direction value count>	Direction for WP operation: 0-15=for the port 4 groups. Each bit controls the direction of a byte of the port. 1 for output, 0 for input. Value for WP operation: 0 - 0xFFFFFFFF for port value Count for WC operation: 0 - 255 for counter value

Sample Program Listing

```

/*****
FILE      : GxPIOExample.cpp
PURPOSE   : WIN32 sample program for GX5731, GX5732 and GX5733
            boards using the GXPIO driver.
CREATED   : March 2003
COPYRIGHT : Copyright 2007 MARVIN TEST SOLUTIONS - MTS Inc.
COMMENTS  :
To compile the WIN32 example:
1. Microsoft VC++
    Load GxPIOExampleC.dsp, .vcproj or .mak, depends on
    the VC++ version from the Project/File/Open... menu
    Select Project/Rebuild all from the menu
2. Borland C++ Builder
    Load GxPIOExampleC.bpr from the Project/Open Project... menu
    Select Project/Build all from the menu

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

// Borland C++ Builder compat. block
#ifdef __BORLANDC__
#pragma hdrstop
#include <condefs.h>
USELIB("GxPIOBC.lib");
USERC("GxPioExampleC.rc");
//-----
#endif // defined(__BORLANDC__)

/*****
//      DisplayMsg
//*****/
void DisplayMsg(PSTR lpszMsg)
{
    MessageBeep(0);
    MessageBox(0, lpszMsg, "GXPIO", MB_OK);
    return;
}

/*****
//      DisplayUsage
//*****/
void DisplayUsage(void)
{
    DisplayMsg(
        "This example shows how to use the GX5731 or GX5732\r\n"
        "Usage:\r\n"
        "GxPIOExample <board_type> <slot_number> <port|counter numbers> <operation>\r\n"
        "<direction|value|count>\r\n\r\n"
        "Where: \r\n"
        "<board_type> - Board type: 5731=GX5731, 5732=GX5732, 5733=GX5733\r\n\r\n"
        "<slot_number> - PCI/PXI slot number as shown by the PXI explorer\r\n\r\n"
        "<port|counter numbers> - depends on the operation:\r\n\r\n"
    );
}

```



```

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

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

    // Parse command line parameters
    nSlotNum=(SHORT)strtol(*(++argv), NULL, 0);
    nBoardType=(SHORT)strtol(*(++argv), NULL, 0);
    nPortOrCount=(SHORT)strtol(*(++argv), NULL, 0);
    #if _MSC_VER < 1400
        sOperation = strdup(*(++argv));
    #else
        sOperation=*(++argv);
        _strdup_s(sOperation, sizeof sOperation);
    #endif

    // Check parameters
    if (nPortOrCount<0 || nPortOrCount>7) DisplayUsage();

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

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

```

```

        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5731SetPortDirection(nHandle, nPortOrCount, SHORT(dwValue),
            &nStatus);
        CheckStatus(nStatus);
        printf("Writes port direction OK\n");
    }
    else if (!strcmp(sOperation, "RD")) // read port direction
    {
        if (nPortOrCount>6) DisplayUsage();
        Gx5731GetPortDirection(nHandle, nPortOrCount, ((PSHORT)&dwValue),
            &nStatus);
        CheckStatus(nStatus);
        printf("Reads port direction: %d\n", SHORT(dwValue));
    }
    else if (!strcmp(sOperation, "WP")) // write port data
    {
        if (nPortOrCount>6) DisplayUsage();
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5731SetPort(nHandle, nPortOrCount, dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Writes port value OK\n");
    }
    else if (!strcmp(sOperation, "RP")) // read port data
    {
        if (nPortOrCount>6) DisplayUsage();
        Gx5731GetPort(nHandle, nPortOrCount, &dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Reads port value: 0x%X\n", dwValue);
    }
    else if (!strcmp(sOperation, "WB")) // write buffer
    {
        if (argc<5 || (nPortOrCount<0 || nPortOrCount>2)) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        dwSize=1;
        Gx5731ModuleBufferWriteData(nHandle, nPortOrCount, &dwValue,
            &dwSize, &nStatus);
        CheckStatus(nStatus);
        printf("Writes data to buffer OK\n");
    }
    else if (!strcmp(sOperation, "RB")) // read buffer
    {
        if (nPortOrCount<0 || nPortOrCount>2) DisplayUsage();
        dwSize=1;
        Gx5731ModuleBufferReadData(nHandle, nPortOrCount, &dwValue,
            &dwSize, &nStatus);
        CheckStatus(nStatus);
        printf("Reads data to buffer: %d\n", dwValue);
    }
    else if (!strcmp(sOperation, "RBF")) // read buffer empty/full flag
    {
        if (argc<5 || (nPortOrCount<0 || nPortOrCount>2))
            DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5731ModuleBufferGetState(nHandle, nPortOrCount,
            ((PSHORT)&dwValue), &nStatus);
        CheckStatus(nStatus);
        printf("Module Buffer Flag state is: %d\n", dwValue);
    }
    else if (!strcmp(sOperation, "SCLK_SRC")) // set internal clock source
    {
        if (argc<5 || (nPortOrCount<0 || nPortOrCount>2)) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);

```

```

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

// Borad type is GX5732
else
{
    Gx5732Initialize(nSlotNum, &nHandle, &nStatus);
    CheckStatus(nStatus);

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

```

```

    {
        if (nPortOrCount>6) DisplayUsage();
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5732SetPort(nHandle, nPortOrCount, dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Writes port value OK\n");
    }
    else if (!strcmp(sOperation, "RP")) // read port data
    {
        if (nPortOrCount>6) DisplayUsage();
        Gx5732GetPort(nHandle, nPortOrCount, &dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Reads port value: 0x%X\n", dwValue);
    }
    else if (!strcmp(sOperation, "WC")) // write counter
    {
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5732SetCounterValue(nHandle, nPortOrCount, (BYTE)dwValue,
            &nStatus);
        CheckStatus(nStatus);
        printf("Writes counter OK\n");
    }
    else if (!strcmp(sOperation, "RC")) // read counter
    {
        Gx5732GetCounterValue(nHandle, nPortOrCount, (BYTE*)&dwValue,
            &bTerminalCountState, &nStatus);
        CheckStatus(nStatus);
        printf("Reads counter: %d, terminal count state: %d\n",
            (BYTE)dwValue, bTerminalCountState);
    }
    else
        DisplayUsage();
}
// Borad type is GX5733
if (nBoardType==5733)
{
    Gx5733Initialize(nSlotNum, &nHandle, &nStatus);
    CheckStatus(nStatus);

    if(!strcmp(sOperation, "WD")) // Write port direction
    {
        if (nPortOrCount>3) DisplayUsage();
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5733SetPortDirection(nHandle, nPortOrCount, SHORT(dwValue),
            &nStatus);
        CheckStatus(nStatus);
        printf("Writes port direction OK\n");
    }
    else if (!strcmp(sOperation, "RD")) // read port direction
    {
        if (nPortOrCount>3) DisplayUsage();
        Gx5733GetPortDirection(nHandle, nPortOrCount, ((PSHORT)&dwValue),
            &nStatus);
        CheckStatus(nStatus);
        printf("Reads port direction: %d\n", SHORT(dwValue));
    }
    else if (!strcmp(sOperation, "WP")) // write port data
    {
        if (nPortOrCount>3) DisplayUsage();
        if (argc<5) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5733SetPort(nHandle, nPortOrCount, dwValue, &nStatus);
    }
}

```

```

        CheckStatus(nStatus);
        printf("Writes port value OK\n");
    }
    else if (!strcmp(sOperation, "RP")) // read port data
    {
        if (nPortOrCount>3) DisplayUsage();
        Gx5733GetPort(nHandle, nPortOrCount, &dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Reads port value: 0x%X\n", dwValue);
    }
    else if (!strcmp(sOperation, "WB")) // write buffer
    {
        if (argc<5 || nPortOrCount!=0) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        dwSize=1;
        Gx5733ModuleBufferWriteData(nHandle, nPortOrCount, &dwValue,
            &dwSize, &nStatus);
        CheckStatus(nStatus);
        printf("Writes data to buffer OK\n");
    }
    else if (!strcmp(sOperation, "RB")) // read buffer
    {
        if (nPortOrCount!=0) DisplayUsage();
        dwSize=1;
        Gx5733ModuleBufferReadData(nHandle, nPortOrCount, &dwValue,
            &dwSize, &nStatus);
        CheckStatus(nStatus);
        printf("Reads data to buffer: %d\n", dwValue);
    }
    else if (!strcmp(sOperation, "RBF")) // read buffer empty/full flag
    {
        if (argc<5 || nPortOrCount!=0) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5733ModuleBufferGetState(nHandle, nPortOrCount,
            (PSHORT)&dwValue, &nStatus);
        CheckStatus(nStatus);
        printf("Module Buffer Flag state is: %d\n", dwValue);
    }
    else if (!strcmp(sOperation, "SCLK_SRC")) // set internal clock source
    {
        if (argc<5 || nPortOrCount!=0) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5733GetInternalClock(nHandle, nPortOrCount, &nSource,
            &dwDivider, &nStatus);
        Gx5733SetInternalClock(nHandle, nPortOrCount, SHORT(dwValue),
            dwDivider, &nStatus);
        CheckStatus(nStatus);
        printf("Set internal clock source OK\n");
    }
    else if (!strcmp(sOperation, "GCLK_SRC")) // get internal clock source
    {
        if (nPortOrCount!=0) DisplayUsage();
        Gx5733GetInternalClock(nHandle, nPortOrCount, ((PSHORT)&dwValue),
            &dwDivider, &nStatus);
        CheckStatus(nStatus);
        printf("Internal clock source is: %d\n", dwValue);
    }
    else if (!strcmp(sOperation, "SCLK_DIV")) // set internal clock divider
    {
        if (argc<5 || nPortOrCount!=0) DisplayUsage();
        dwValue=(DWORD)strtol(*(++argv), NULL, 0);
        Gx5733GetInternalClock(nHandle, nPortOrCount, &nSource,
            &dwDivider, &nStatus);
        Gx5733SetInternalClock(nHandle, nPortOrCount, nSource, dwValue,

```



```

        &nStatus);
    CheckStatus(nStatus);
    printf("Set internal clock divider OK\n");
}
else if (!strcmp(sOperation, "GCLK_DIV")) // get internal clock divider
{
    if (nPortOrCount!=0) DisplayUsage();
    Gx5733GetInternalClock(nHandle, nPortOrCount, &nSource,
        &dwDivider, &nStatus);
    CheckStatus(nStatus);
    printf("Internal clock divider is: %d\n", dwDivider);
}
else
    DisplayUsage();
}
else
    DisplayUsage();

return 0;
}

//*****
//    End Of File
//*****

```


Chapter 6 - Functions Reference

Introduction

The GX5733 driver functions reference chapter is organized in alphabetical order. Each function description contains the function name; purpose, syntax, parameters description and type followed by Comments, an Example (written in C), and a See Also sections.

All function parameters syntax follows the same rules:

- Strings are ASCIIZ (null or zero character terminated).
- Most function's first parameter is *nHandle* (16-bit integer). This parameter is required for operating the board and is returned by the **Gx5734Initialize** or the **Gx5734InitializeVisa** functions. The *nHandle* is used to identify the board when calling a function for programming and controlling the operation of that board.
- All functions return a status with the last parameter named *pnStatus*. The *pnStatus* is zero if the function was successful, or less than a zero on error. The description of the error is available using the **GxPioGetErrorString** function or by using a predefined constant, defined in the driver interface files: GXPIO.H, GXPIO.BAS, GXPIO.VB, GXPIO.PAS or GX5733.DRV.
- Parameter name are prefixed as follows:

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

Table 6-1: Parameter Name Prefixes

GX5733 Functions

The following list is a summary of functions available for the GX5733:

Driver Functions	Description
General	
Gx5733GetBoardSummary	Returns the board summary.
Gx5733Initialize	Initializes the driver for the board at the specified slot number using HW. The function returns a handle that can be used with other GX5733 functions to program the board
Gx5733InitializeVisa	Initializes the driver for the specified slot using VISA. The function returns a handle that can be used with other GX5733 functions to program the board.
Gx5733Panel	Opens a virtual panel used to interactively control the GX5733.
Gx5733Reset	Resets the GX5733 board to its default settings.
GxPioGetDriverSummary	Returns the driver name and version.
GxPioGetErrorString	Returns the error string associated with the specified error number.
Digital I/O Functions	
Gx5733GetPort	Reads the port value.
Gx5733GetPortBit	Reads a port bit value.
Gx5733GetPortByte	Reads a port byte value.
Gx5733GetPortByteDirection	Returns the direction of a port byte.
Gx5733GetPortDirection	Returns the direction of port bytes.
Gx5733GetPortWord	Reads a port word value.
Gx5733ResetPort	Reset the specified digital I/O port.
Gx5733SetPort	Writes a value to a port.
Gx5733SetPortBit	Writes a value to a port byte.
Gx5733SetPortByte	Writes a value to the port byte.
Gx5733SetPortByteDirection	Sets the direction of a port byte.
Gx5733SetPortDirection	Sets the direction for port bytes.
Gx5733SetPortWord	Writes a value to the port word.
Module Functions	
Gx5733ModuleBufferClear	Clear the Module's Buffer.
Gx5733ModuleBufferGetMode	Returns the Module's Buffer Full or Half-Full Mode.
Gx5733ModuleBufferGetState	Returns the Module's Buffer Full or Half-Full State.
Gx5733ModuleBufferReadData	Returns the Module's buffer.
Gx5733ModuleBufferSetMode	Set the Module's Buffer Full or Half-Full Mode.

Driver Functions	Description
Gx5733ModuleBufferTest	Test the specified Module buffer for reading and writing
Gx5733ModuleBufferWriteData	Write a buffer to the Module's Buffer.
Gx5733ModuleGetDirection	Returns the Module's channels direction
Gx5733ModuleGetDirectionSource	Returns the Module's channels direction source
Gx5733ModuleGetExtStrobeEnablePolarity	Returns the Module's External Strobe Enable Polarity.
Gx5733ModuleGetExtStrobeEnableState	Returns the Module's External Strobe Enable State.
Gx5733ModuleGetExtStrobePolarity	Returns the Module's External Strobe Polarity.
Gx5733ModuleGetExtStrobeState	Returns the Module's External Strobe State.
Gx5733ModuleGetHandshakeOutputMode	Returns the Module's Handshake output mode.
Gx5733ModuleGetInternalStrobeSource	Returns the Module's Internal Strobe Clock source.
Gx5733ModuleGetOperationMode	Returns the Module's Operation mode.
Gx5733ModuleGetOutputMode	Returns the Module's Output mode.
Gx5733ModuleGetOutputStrobeWidth	Returns the Module's Output Strobe Width.
Gx5733ModuleGetPort	Returns the Module's Port Data.
Gx5733ModuleGetPxiTriggerBus	Returns all specified Module Strobe to PXI Trigger lines connection mode.
Gx5733ModuleGetPxiTriggerLine	Returns the specified Module's Strobe to PXI Trigger Line connection mode.
Gx5733ModuleGetRevision	Returns the Module's Revision.
Gx5733ModuleGetState	Returns the Module's State.
Gx5733ModuleGetStrobeSource	Returns the Module's Strobe Source.
Gx5733ModuleGetTermination	Returns the Module termination mode
Gx5733ModuleGetType	Returns the Module's type.
Gx5733ModuleGetVThreshold	Returns the Module's Threshold voltage.
Gx5733ModuleHalt	Sets the specified Module to Halt state.
Gx5733ModuleRun	Sets the specified Module to Run mode.
Gx5733ModuleSetDirection	Sets the Module's channels direction
Gx5733ModuleSetDirectionSource	Sets the Module's channels direction source
Gx5733ModuleSetExtStrobeEnablePolarity	Sets the Module's External Strobe Enable Polarity.
Gx5733ModuleSetExtStrobePolarity	Sets the Module's External Strobe Polarity.
Gx5733ModuleSetHandshakeOutputMode	Sets the Module's Handshake output mode.
Gx5733ModuleSetInternalStrobeSource	Sets the Module's Internal Strobe Clock source.
Gx5733ModuleSetOperationMode	Sets the Module's Operation mode.
Gx5733ModuleSetOutputMode	Sets the Module's Output mode.
Gx5733ModuleSetOutputStrobeWidth	Sets the Module's Output Strobe Width.

Driver Functions	Description
Gx5733ModuleSetPort	Sets the Module's Port Data.
Gx5733ModuleSetPxiTriggerBus	Sets all specified Module Strobe to PXI Trigger lines connection mode.
Gx5733ModuleSetPxiTriggerLine	Sets the specified Module's Strobe to PXI Trigger Line connection mode.
Gx5733ModuleSetStrobeSource	Sets the Module's Strobe Source.
Gx5733ModuleSetVThreshold	Sets the Module's Threshold voltage.
Gx5733ModuleStep	Run the specified port Module for given number of steps.
Gx5733ModuleSetTermination	Sets the Module termination mode
Clock Functions	
Gx5733GetInternalClock	Returns the internal clock source and divider.
Gx5733GetInternalClockEnable	Returns the specified internal clock enable mode
Gx5733SetInternalClock	Sets internal clock source and divider.
Gx5733SetInternalClockEnable	Sets the specified internal clock enable mode
PXI Trigger Functions	
Gx5733GetStrobeToPxiTriggerBusState	Returns all Strobe to PXI Trigger lines Output Mode.
Gx5733GetStrobeToPxiTriggerLineState	Returns the specified Strobe to PXI Trigger Line output mode..
Gx5733SetStrobeToPxiTriggerBusState	Sets all Strobe to PXI Trigger lines Output Mode
Gx5733SetStrobeToPxiTriggerLineState	Sets the specified Strobe to PXI Trigger Line output mode

Gx5733GetBoardSummary

Purpose

Returns the board summary.

Syntax

Gx5733GetBoardSummary (*nHandle*, *szSummary*, *nSumMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>szSummary</i>	PSTR	Buffer to contain the returned board info (null terminated) string.
<i>nSumMaxLen</i>	SHORT	Size of the buffer to contain the error string.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX5733 summary string provides the following data:

- Instrument name (e.g., GX5733)
- FPGA version of main board (e.g. 0xA003)
- Each installed Module's FPGA version.

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

```
GX5733, FPGA Version:0xA000, Module: GX5701 FPGA Version:0x000A
```

Example

The following example returns the board summary:

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

Gx5733GetBoardSummary(nHandle, szSummary, sizeof(szSummary), &nStatus)
```

See Also

GxPioGetDriverSummary, **Gx5733Initialize**, **GxPioGetErrorString**

Gx5733GetInternalClock

Purpose

Returns the internal clock source and divider.

Syntax

Gx5733GetInternalClock (*nHandle*, *nClock*, *pnSource*, *pdwDivider*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nClock</i>	SHORT	Clock number: 0 GX5733_CLOCK0: Internal Clock 0 1 GX5733_CLOCK1: Internal Clock 1
<i>pnSource</i>	PSHORT	Internal Clock 0 sources are: 0. GX5733_INTERNAL_CLOCK0_NOT_CONNECTED: Clock not connected. 1. GX5733_INTERNAL_CLOCK0_T0_PCI_CLOCK_33MHZ: Internal Clock 0 connected to 33MHz PCI Clock. 2. GX5733_INTERNAL_CLOCK0_T0_PXI_CLOCK_10MHZ: Internal Clock 0 connected to 10MHz PXI Clock. 3. GX5733_INTERNAL_CLOCK0_T0_STAR_TRIGGER: Internal Clock 0 connected to the PXI Start Trigger. Internal Clock 1 sources are: 0. GX5733_INTERNAL_CLOCK1_NOT_CONNECTED: Clock not connected. 1. GX5733_INTERNAL_CLOCK1_T0_PCI_CLOCK_33MHZ: Internal Clock 1 connected to 33MHz PCI Clock. 2. GX5733_INTERNAL_CLOCK1_T0_PXI_CLOCK_10MHZ: Internal Clock 0 connected to 10MHz PXI Clock. 3. GX5733_INTERNAL_CLOCK1_T0_STAR_TRIGGER: Internal Clock 0 connected to the PXI Start Trigger.
<i>dwDivider</i>	DWORD	Divider value: 1-0x8000000
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733SetInternalClock** to setup the internal clock.

Example

The following example reads clock 0 setup:

```
SHORT nSource, nStatus;
DWORD dwDivider;

Gx5733GetInternalClock (nHandle, 0, &nSource, &dwDivider, &nStatus);
```

See Also

Gx5733SetInternalClock

Gx5733GetInternalClockEnable

Purpose

Return the specified internal clock enable mode.

Syntax

Gx5733GetInternalClockEnable (*nHandle*, *nClock*, *pnEnable*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5733 board.
<i>nClock</i>	SHORT	Clock number: 0 GX5733_CLOCK0: Internal Clock 0 1 GX5733_CLOCK1: Internal Clock 1
<i>pnEnable</i>	PSHORT	Set clock enable mode: 0 GX5733_DISABLE_INTERNAL_CLOCK: Disable the specified clock. 1 GX5733_ENABLE_INTERNAL_CLOCK: Enable the specified clock.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

```
SHORT nEnable, nStatus;
```

The following example returns the internal clock 0 enable state:

```
Gx5733GetInternalClockEnable (nHandle, GX5733_CLOCK0, &nEnable, &nStatus);
```

See Also

Gx5733SetInternalClockEnable, **Gx5733GetInternalClock**

Gx5733GetPort

Purpose

Reads a port value.

Syntax

Gx5733GetPort (*nHandle*, *nPort*, *pdwValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 1-3. • Port 0 if no I/O module installed.
<i>pdwValue</i>	PDWORD	Returned port value, 32 bit integer where each bit represents a channel. Bit 0 for channel 0 and bit 31 for channel 31.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733SetPort** to write data to the port when in Output direction.

Example

The following example reads port 1 value:

```
SHORT nStatus;
DWORD dwValue;
```

```
Gx5733GetPort (nHandle, 1, &dwValue, &nStatus);
```

See Also

Gx5733SetPort, **Gx5733GetPortBit**, **Gx5733GetPortByte**, **Gx5733GetPortWord**, **Gx5733SetPortDirection**

Gx5733GetPortBit

Purpose

Reads a port bit value.

Syntax

Gx5733GetPortBit (*nHandle*, *nPort*, *nBit*, *pucValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 1-3. • Port 0 if no I/O module installed.
<i>nBit</i>	SHORT	Port bit number: 0-31. Where 0 is used for low order bit and 31 for the high order bit.
<i>pucValue</i>	PBYTE	Returned channel value: 0 for low state and 1 for hi state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733SetPortBit** to write data to the port bit when in Output direction.

Example

The following example reads port 1, bit 30 value:

```
SHORT nStatus;
BYTE ucValue;

Gx5733GetPort (nHandle, 1, 30, &ucValue, &nStatus);
```

See Also

Gx5733SetPortBit, **Gx5733GetPortByte**, **Gx5733GetPortWord**, **Gx5733GetPort**

Gx5733GetPortByte

Purpose

Reads a port byte value.

Syntax

Gx5733GetPortByte (*nHandle*, *nPort*, *nByte*, *pucValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 1-3. • Port 0 if no I/O module installed.
<i>nByte</i>	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte.
<i>pucValue</i>	PBYTE	Returned value 0-255. Each bit represents the specific channel state bit 0 for lowest channel number.
<i>pnStatus</i>	LPSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733SetPortByte** to write data to the port byte when in Output direction.

Example

The following example reads port 1, byte 2 value:

```
SHORT nStatus;
BYTE ucValue;

Gx5733GetPortByte (nHandle, 1, 2, &ucValue, &nStatus);
```

See Also

Gx5733SetPortByte, **Gx5733GetPortBit**, **Gx5733GetPortWord**, **Gx5733GetPort**

Gx5733GetPortByteDirection

Purpose

Returns the direction of a port byte.

Syntax

Gx5733GetPortByteDirection (*nHandle*, *nPort*, *nByte*, *bInOut*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 1-3. • Port 0 if no I/O module installed.
<i>nByte</i>	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte.
<i>bInOut</i>	PBOOL	Returned byte direction: 0. (FALSE) – Input. 1. (TRUE) – Output.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
SHORT nStatus;
BOOL bout;

Gx5733GetPortByteDirection (nHandle, 1, 3, &bOut, nStatus);
```

See Also

Gx5733SetPortByteDirection, **Gx5733GetPortDirection**

Gx5733GetPortDirection

Purpose

Returns the direction of port bytes.

Syntax

Gx5733GetPortDirection (*nHandle*, *nPort*, *pnDirection*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port numbers are as follow: - Digital I/O port numbers 1-3. - Port 0 if no I/O module installed.
<i>pnDirection</i>	PSHORT	Returned port bytes directions where bit 0 corresponds to byte 0, bit 1 to byte 1, bit 2 byte 2 and bit 3 to byte 0. A bit is set to hi '1' for output and low '0' for input.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

The following example returns the direction of port 1:

```
SHORT nDirection, nStatus;
```

```
Gx5733GetPortByteDirection (nHandle, 1, &nDirection, nStatus);
```

See Also

Gx5733SetPortDirection, Gx5733GetPortByteDirection

Gx5733GetPortWord

Purpose

Reads a port word value.

Syntax

Gx5733GetPortWord (*nHandle*, *nPort*, *nWord*, *pwValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port numbers are as follow: • Digital I/O port numbers 1-3. • Port 0 if no I/O module installed.
<i>nWord</i>	SHORT	Port word number: 0. GX5733_PORT_LOW_WORD: port low order word (bytes 0 and 1). 1. GX5733_PORT_HIGH_WORD: high order word (bytes 2 and 3).
<i>pwValue</i>	PWORD	Returned port word value: 0 to 65,535 (0-0xFFFF). Where bit 0 corresponds to channel 0 and bit 15 to channel 15.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733SetPortWord** to write data to the port word when in Output direction.

Example

The following example reads port 1, word 0 value:

```
SHORT nStatus;
WORD wValue;
Gx5733GetPortByte (nHandle, 1, 0, &wValue, &nStatus);
```

See Also

Gx5733SetPortWord, **Gx5733GetPortBit**, **Gx5733GetPortByte**, **Gx5733GetPort**

Gx5733GetStrobeToPxiTriggerBusState

Purpose

Returns all Strobe to PXI Trigger lines Output Mode.

Syntax

Gx5733GetStrobeToPxiTriggerBusState (*nHandle*, *pnBusState*, *pnStatus*)

Parameters

Name	Type	Comments
<i>Nhandle</i>	SHORT	Handle to a GX5733 board.
<i>pnBusState</i>	PSHORT	Returned all eight Strobe to PXI Trigger Bus lines Output mode, 8 bits integer where each bit represents a PXI Trigger line. Bit 0 for line 0 and bit 7 for line 7. A high ('1') means the corresponding PXI Trigger line is enable, low ('0') means the corresponding PXI Trigger line is disabled.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

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

```
SHORT nBusState, nStatus;
```

```
Gx5733GetStrobeToPxiTriggerBusState (nHandle, &nBusState, &nStatus);
```

See Also

Gx5733GetStrobeToPxiTriggerBusState, **Gx5733GetStrobeToPxiTriggerLineState**, **Gx5733ModuleGetPxiTriggerBus**, **Gx5733ModuleGetPxiTriggerLine**.

Gx5733GetStrobeToPxiTriggerLineState

Purpose

Returns the specified Strobe to PXI Trigger Line output mode.

Syntax

Gx5733GetStrobeToPxiTriggerLineState (*nHandle*, *nLine*, *pbMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nLine</i>	SHORT	PXI Trigger Line numbers are: 0 GX5733_PXI_TRIGGER_LINE0: Line 0 1 GX5733_PXI_TRIGGER_LINE1: Line 1 2 GX5733_PXI_TRIGGER_LINE2: Line 2 3 GX5733_PXI_TRIGGER_LINE3: Line 3 4 GX5733_PXI_TRIGGER_LINE4: Line 4 5 GX5733_PXI_TRIGGER_LINE5: Line 5 6 GX5733_PXI_TRIGGER_LINE6: Line 6 7 GX5733_PXI_TRIGGER_LINE7: Line 7
<i>pbMode</i>	PBOOL	Returned PXI trigger line mode are: 0 GX5733_PXI_TRIGGER_LINE_DISCONNECTED: PXI trigger line is disconnected. 1 GX5733_PXI_TRIGGER_LINE_CONNECTED: PXI trigger line is Connected
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733SetStrobeToPxiTriggerBusState** function to set all PXI-Trigger lines at once.

Example

The following example reads PXI Trigger line 1mode:

```
SHORT nStatus;
BOOL bMode;

Gx5733GetStrobeToPxiTriggerLineState(nHandle, GX5733_PXI_TRIGGER_LINE1, &bMode, &nStatus);
```

See Also

Gx5733SetStrobeToPxiTriggerBusState, **Gx5733SetStrobeToPxiTriggerLineState**.

Gx5733Initialize

Purpose

Initializes the GX5733 driver for the specified board slot.

Syntax

Gx5733Initialize (*nSlot*, *pnHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nSlot</i>	SHORT	GX5733 board slot number.
<i>pnHandle</i>	PSHORT	Returned Handle for a GX5733 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, 1 on failure.

Comments

The **Gx5733Initialize** function verifies whether or not the GX5733 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 GX5733 functions.

Example

The following example initializes a GX5733 boards at PCI bus slot 1.

```
SHORT nHandle, nStatus;

Gx5733Initilize(1, &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

GxPioGetErrorString, **Gx5733Reset**

Gx5733InitializeVisa

Purpose

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

Syntax

Gx5733InitializeVisa (*szVisaResource*, *pnHandle*, *pnStatus*)

Parameters

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

Comments

The **Gx5733InitializeVisa** 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 GX5733 boards at PXI bus 5 and device 11.

```
SHORT nHandle, nStatus;
Gx5733InitializeVisa ("PXI5::11::INSTR", &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

Gx5733Initialize, **Gx5733Reset**, **GxPIOGetErrorString**

Gx5733ModuleBufferClear

Applies To

GX5701, GX5702, GX5704

Purpose

Clear the Module's Buffer.

Syntax

Gx5733ModuleBufferClear (*nHandle*, *nPort*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

The following example return the clears Module 2 Buffer:

```
SHORT nStatus;  
Gx5733ModuleBufferClear (nHandle, GX5733_PORT0, &nStatus);
```

See Also

Gx5733ModuleBufferReadData, **Gx5733ModuleBufferWriteData**

Gx5733ModuleBufferGetMode

Applies To

GX5701, GX5702, GX5704

Purpose

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

Syntax

Gx5733ModuleBufferGetFlagMode (*nHandle*, *nPort*, *pnMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnMode</i>	PSHORT	Module Buffer Flag mode are as follow: 0 GX5733_MODULE_BUFFER_MODE_FULL: The module Buffer flag is set to detect a Full or Empty state. 1 GX5733_MODULE_BUFFER_MODE_HALF_FULL: The module Buffer flag is set to detect a Half-Full or Half-Empty state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

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

```
SHORT nMode, nStatus;
Gx5733ModuleBufferGetMode (nHandle, GX5733_PORT0, &nMode, &nStatus);
```

See Also

Gx5733ModuleBufferSetMode, Gx5733ModuleBufferWriteData, Gx5733ModuleBufferReadData.

Gx5733ModuleBufferGetState

Applies To

GX5701, GX5702, GX5704

Purpose

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

Syntax

Gx5733ModuleBufferGetState (*nHandle*, *nPort*, *pnState*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnState</i>	PSHORT	Returned Buffer state can be as follow: 0 GX5733_MODULE_BUFFER_EMPTY: buffer is empty (reading from the buffer) or not full (writing to the buffer). 1 GX5733_MODULE_BUFFER_FULL: buffer is full (writing to the buffer) or buffer is empty (reading from the buffer).
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Writing to the buffer:

0. Indicates the buffer in not yet full.
1. The buffer is full. In order to write new data the buffer needs to be cleared first by calling **Gx5733ModuleBufferClear** or by reading data from the buffer using **Gx5733ModuleBufferReadData**.

Reading from the buffer:

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

Example

The following example returns Module 0 buffer flag state:

```
SHORT nState, nStatus;
Gx5733ModuleBufferGetState (nHandle, GX5733_PORT0, &nState, &nStatus);
```

See Also

Gx5733ModuleBufferWriteData, **Gx5733ModuleBufferReadData**, **Gx5733ModuleBufferSetMode**

Gx5733ModuleBufferReadData

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's buffer.

Syntax

Gx5733ModuleBufferReadData (*nHandle*, *nPort*, *pdwVector*, *pdwSize*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pdwVector</i>	PDWORD	Array of double words (32-bit)
<i>pdwSize</i>	PWORD	Number of requested of double words to read. Returns the actual number of double words read. The Array size can be between 0 and 4095.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function will continue to read data from the Module's buffer until the number of requested of double words to read is reached or if the buffer empty flag is set to TRUE. See the **Gx5733ModuleGetBufferState** for more details.

Example

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

```
SHORT nStatus;
DWORD adwVector[64];
DWORD dwSize=18;
Gx5733GetPortByte (nHandle, GX5733_PORT0, adwVector &dwSize, &nStatus);
```

See Also

Gx5733ModuleBufferWriteData.

Gx5733ModuleBufferSetFlagMode

Applies To

GX5701, GX5702, GX5704

Purpose

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

Syntax

Gx5733ModuleBufferSetFlagMode (*nHandle*, *nPort*, *nMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnMode</i>	PDWORD	Module Buffer Flag mode are as follow: 0 GX5733_MODULE_BUFFER_MODE_FULL: The module Buffer flag is set to detect a Full or Empty state. 1 GX5733_MODULE_BUFFER_MODE_HALF_FULL: The module Buffer flag is set to detect a Half-Full or Half-Empty state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets the Buffer flag mode to detect a Full or Empty state for port 0:

```
SHORT nStatus;
Gx5733ModuleBufferSetFlagMode (nHandle, GX5733_PORT0, GX5733_MODULE_BUFFER_MODE_FULL, &nStatus);
```

See Also

Gx5733ModuleBufferGetMode, Gx5733ModuleBufferWriteData, Gx5733ModuleBufferReadData.

Gx5733ModuleBufferSetMode

Applies To

GX5701, GX5702, GX5704

Purpose

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

Syntax

Gx5733ModuleBufferSetMode (*nHandle*, *nPort*, *nMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nMode</i>	SHORT	Buffer mode are as follow: 0 GX5733_MODULE_BUFFER_MODE_FULL: The module FIFO flag is set to detect a Full or Empty state. 1 GX5733_MODULE_BUFFER_MODE_HALFFULL: The module Buffer flag is set to detect a Half-Full or Half-Empty state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets Module's 0 Buffer mode to GX5733_MODULE_BUFFER_MODE_FULL:

```
SHORT nStatus;
Gx5733ModuleBufferGetMode (nHandle, GX5733_PORT0, GX5733_MODULE_BUFFER_MODE_FULL, &nStatus);
```

See Also

Gx5733ModuleBufferGetMode, **Gx5733ModuleBufferGetState**

Gx5733ModuleBufferTest

Applies To

GX5701, GX5702, GX5704

Purpose

Test the specified Module's Buffer.

Syntax

Gx5733ModuleBufferTest (*nHandle*, *nPort*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
SHORT nMode, nStatus;  
Gx5733ModuleBufferTest (nHandle, GX5733_PORT0, &nStatus);
```

See Also

Gx5733ModuleBufferWriteData, Gx5733ModuleBufferReadData.

Gx5733ModuleBufferWriteData

Applies To

GX5701, GX5702, GX5704

Purpose

Write a buffer to the Module's Buffer.

Syntax

Gx5733ModuleBufferWriteData (*nHandle*, *nPort*, *pdwVector*, *pdwSize*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pdwVector</i>	PDWORD	Array of double words (32-bit)
<i>pdwSize</i>	PWORD	Number of requested of double words to write. Returns the actual number of double words written. The Array size can be between 0 and 4095.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function will continue to write data from the Module's buffer until the number of requested of double words to write is reached or if the buffer full flag is set to TRUE. See the **Gx5733ModuleGetBufferState** for more details.

Example

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

```
SHORT nStatus;
DWORD adwVector[64];
DWORD dwSize=18;
Gx5733ModuleBufferWriteData (nHandle, GX5733_PORT0, adwVector &dwSize, &nStatus);
```

See Also

Gx5733ModuleBufferReadData

Gx5733ModuleGetDirection

Applies To

GX5711, GX5712

Purpose

Returns the Module's channels direction.

Syntax

Gx5733ModuleGetDirection (*nHandle*, *nPort*, *nGroup*, *pnDirection*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>NPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>NGroup</i>	SHORT	Specified Module's port group number: 0 GX5733_MODULE_GROUP0: Module port group 0. 1 GX5733_MODULE_GROUP1: Module port group 1.
<i>pnDirection</i>	PSHORT	Returns Module's channels direction mode: 0 GX5733_MODULE_DIRECTION_DIFFERENTIAL_TO_TTL: Module's channels direction is from differential to TTL. 1 GX5733_MODULE_DIRECTION_TTL_TO_DIFFERENTIAL: Module's channels direction is from TTL to differential.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The data that flows through the Module can be monitor by calling **Gx5733GetPort**.

Example

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

```
SHORT nDirection, nStatus;
Gx5733ModuleGetDirection (nHandle, GX5733_PORT0, GX5733_MODULE_GROUP0, &nDirection, &nStatus)
```

See Also

Gx5733ModuleSetDirection, **Gx5733ModuleSetDirectionSource**, **Gx5733GetPort**, **Gx5733ModuleSetInternalStrobeSource**

Gx5733ModuleGetDirectionSource

Applies To

GX5711, GX5712

Purpose

Returns the Module's channels direction source.

Syntax

Gx5733ModuleGetDirectionSource (*nHandle*, *nPort*, *pnSource*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnSource</i>	PSHORT	Returned Module's channels direction source: 0 GX5733_MODULE_DIRECTION_SOURCE_INTERNAL: Module's channels direction is set internally (software). 1 GX5733_MODULE_DIRECTION_SOURCE_EXTERNAL: Module's channels direction is set externally.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The data that flows through the Module can be monitor by calling **Gx5733GetPort**.

Example

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

```
SHORT nSource, nStatus;
Gx5733ModuleGetDirectionSource (nHandle, GX5733_PORT0, &nSource, &nStatus)
```

See Also

Gx5733ModuleSetDirectionSource, **Gx5733ModuleGetDirection**, **Gx5733GetPort**, **Gx5733ModuleSetInternalStrobeSource**

Gx5733ModuleGetExtStrobeEnablePolarity

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's External Strobe Enable Polarity.

Syntax

Gx5733ModuleGetExtStrobeEnablePolarity (*nHandle*, *nPort*, *pnPolarity*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnPolarity</i>	PSHORT	Returned External Strobe Enable Polarity mode: 0 GX5733_MODULE_STROBE_POSITIVE_EDGE: External Strobe will be enabled on positive edge. 1 GX5733_MODULE_STROBE_NEGATIVE_EDGE: External Strobe will be enabled on negative edge.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

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

```
SHORT nPolarity, nStatus;
Gx5733ModuleGetExtStrobeEnablePolarity (nHandle, GX5733_PORT0, &nPolarity, &nStatus);
```

See Also

Gx5733ModuleSetExtStrobeEnablePolarity, **Gx5733ModuleSetExtStrobePolarity**,
Gx5733ModuleSetStrobeSource, **Gx5733ModuleSetInternalStrobeSource**

Gx5733ModuleGetExtStrobeEnableState

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's External Strobe Enable State.

Syntax

Gx5733ModuleGetExtStrobeEnableState (*nHandle*, *nPort*, *pnState*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnState</i>	PSHORT	Returned External Strobe Enable State mode: 0 External Strobe is disabled ('0'). 1 External Strobe is enabled ('1').
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
SHORT nState, nStatus;
Gx5733ModuleGetExtStrobeEnableState (nHandle, GX5733_PORT0, &nState, &nStatus);
```

See Also

Gx5733ModuleGetExtStrobeState, **Gx5733ModuleSetExtStrobeEnablePolarity**,
Gx5733ModuleSetExtStrobePolarity, **Gx5733ModuleSetStrobeSource**,
Gx5733ModuleSetInternalStrobeSource

Gx5733ModuleGetExtStrobePolarity

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's External Strobe Polarity.

Syntax

Gx5733ModuleGetExtStrobePolarity (*nHandle*, *nPort*, *pnPolarity*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnPolarity</i>	PSHORT	Returned External Strobe Polarity mode: 0 GX5733_MODULE_STROBE_POSITIVE_EDGE: External Strobe is active on positive edge. 1 GX5733_MODULE_STROBE_NEGATIVE_EDGE: External Strobe is active on negative edge.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example returns modules port 0 external strobe polarity:

```
SHORT nPolarity, nStatus;
Gx5733ModuleGetExtStrobePolarity (nHandle, GX5733_PORT0, &nPolarity, &nStatus);
```

See Also

Gx5733ModuleSetExtStrobePolarity, **Gx5733ModuleSetExtStrobeEnablePolarity**,
Gx5733ModuleSetStrobeSource, **Gx5733ModuleSetInternalStrobeSource**

Gx5733ModuleGetExtStrobeState

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's External Strobe State.

Syntax

Gx5733ModuleGetExtStrobeState (*nHandle*, *nPort*, *pnState*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnState</i>	PSHORT	Returned External Strobe Enable State mode: 0 External Strobe state is low ('0'). 1 External Strobe state is high ('1').
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

The following example returns modules port 0 external strobe state:

```
SHORT nState, nStatus;
Gx5733ModuleGetExtStrobeState (nHandle, GX5733_PORT0, &nState, &nStatus);
```

See Also

Gx5733ModuleGetExtStrobeEnableState, **Gx5733ModuleSetExtStrobeEnablePolarity**,
Gx5733ModuleSetExtStrobePolarity, **Gx5733ModuleSetStrobeSource**,
Gx5733ModuleSetInternalStrobeSource

Gx5733ModuleGetHandshakeOutputMode

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Handshake output mode.

Syntax

Gx5733ModuleGetHandshakeOutputMode (*nHandle*, *nPort*, *pnMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnMode</i>	PSHORT	Handshake output mode: 0 GX5733_MODULE_HANDSHAKE_OUTPUT_DISABLE: Handshake output mode disabled. 1 GX5733_MODULE_HANDSHAKE_OUTPUT_ENABLE: Handshake output mode enabled.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX5733 can have one Modules installed. Each installed module can use Direct Output/Input or Strobed Output/Input with handshaking.

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

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

Example

The following example returns modules port 0 Handshake Output mode:

```
SHORT nMode nStatus;
Gx5733ModuleGetHandshakeOutputMode (nHandle, GX5733_PORT0, &nMode, &nStatus);
```

See Also

Gx5733ModuleSetHandshakeOutputMode, **Gx5733ModuleSetExtStrobeEnablePolarity**,
Gx5733ModuleSetStrobeSource, **Gx5733ModuleSetInternalStrobeSource**

Gx5733ModuleGetInternalStrobeSource

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Internal Strobe Clock source.

Syntax

Gx5733ModuleGetInternalStrobeSource (*nHandle*, *nPort*, *pnSource*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nSource</i>	SHORT	Returned source: 0. GX5731_MODULE_INTERNAL_STROBE_NOT_CONNECTED 1. GX5733_CLOCK0: Internal Clock0. 2. GX5733_CLOCK1: Internal Clock1 3. Not used 4. Not used 5. Not used 6. Not used 7. GX5733_PXI_TRIGGER_LINE0: Line 0 8. GX5733_PXI_TRIGGER_LINE1: Line 1 9. GX5733_PXI_TRIGGER_LINE2: Line 2 10. GX5733_PXI_TRIGGER_LINE3: Line 3 11. GX5733_PXI_TRIGGER_LINE4: Line 4 12. GX5733_PXI_TRIGGER_LINE5: Line 5 13. GX5733_PXI_TRIGGER_LINE6: Line 6 14. GX5733_PXI_TRIGGER_LINE7: Line 7
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns modules port 0 internal strobe source:

```
SHORT nSource, nStatus;
Gx5733ModuleGetInternalStrobeSource (nHandle, GX5733_PORT0, &nSource, &nStatus);
```

See Also

Gx5733ModuleSetInternalStrobeSource, **Gx5733SetInternalClock**, **Gx5733SetInternalClockEnable**, **Gx5733ModuleSetExtStrobeEnablePolarity**, **Gx5733ModuleSetExtStrobePolarity**, **Gx5733ModuleSetStrobeSource**

Gx5733ModuleGetOperationMode

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Operation mode.

Syntax

Gx5733ModuleGetOperationMode (*nHandle*, *nPort*, *pnMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnMode</i>	PSHORT	Returns the Module's Operation mode as follow: 0 GX5733_MODULE_OPERATION_BUFFERED: Default operation mode, in this mode the Module is using its buffer to read/write (depend on its type input or output). 1 GX5733_MODULE_OPERATION_PORT_IO: The user can use the Module's as a static port in or port out (depend on its type input or output)
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If Module type is either GX5702 or GX5704 and the Operation mode is Port I/O (GX5733_MODULE_OPERATION_PORT_IO) the outputs can be written statically.

If Module type is GX5701 and the Operation mode is Port I/O (GX5733_MODULE_OPERATION_PORT_IO) the inputs can be read statically.

Example

The following example return the module port 0 operation mode:

```
SHORT nMode, nStatus;
```

```
Gx5733ModuleGetOperationMode (nHandle, GX5733_PORT0, &nMode, &nStatus);
```

See Also

Gx5733ModuleSetOperationMode, **Gx5733ModuleSetOutputMode**, **Gx5733ModuleSetStrobeSource**

Gx5733ModuleGetOutputMode

Applies To

GX5702, GX5704

Purpose

Returns the Module's Output mode.

Syntax

Gx5733ModuleGetOutputMode (*nHandle*, *nPort*, *pnMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnMode</i>	PSHORT	Returns the Module's Output mode as follow: 0 GX5733_MODULE_OUTPUT_DISABLE: All the Module's 32 channels are in Tri-State mode. 1 GX5733_MODULE_OUTPUT_ENABLE: All the Module's 32 channels are enabled to output data..
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Outputs are set to Tri-State when in disabled.

Example

The following example return the module port 0 output mode:

```
SHORT nMode, nStatus;
```

```
Gx5733ModuleGetOutputMode (nHandle, GX5733_PORT0, &nMode, &nStatus);
```

See Also

Gx5733ModuleSetOutputMode, **Gx5733ModuleSetStrobeSource**

Gx5733ModuleGetOutputStrobeWidth

Applies To

GX5702, GX5704

Purpose

Returns the Module's Output Strobe Width.

Syntax

Gx5733ModuleGetOutputStrobeWidth (*nHandle*, *nPort*, *pnStrobeWidth*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnStrobeWidth</i>	PSHORT	Returned integer value representing the Output Strobe Width in nSec. Strobe Width value range is 50 to 5000 with resolution of 50nSec..
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example return the module port 0 strobe width:

```
SHORT nStrobeWidth, nStatus;
```

```
Gx5733ModuleGetOutputStrobeWidth (nHandle, GX5733_PORT0, &nStrobeWidth, &nStatus);
```

See Also

Gx5733ModuleSetOutputStrobeWidth, Gx5733ModuleSetStrobeSource

Gx5733ModuleGetPort

Applies To

GX5701

Purpose

Returns the Module's Port Data.

Syntax

Gx5733ModuleGetPort(*nHandle*, *nPort*, *pdwData*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pdwData</i>	PDWORD	Returned port double word value: 0 to -0xFFFFFFFF. Where bit 0 corresponds to channel 0 and bit 31 to channel 31.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows the module to be used as a port in. Prior to calling this function the module's operating mode needs to be set to Port I/O (GX5733_MODULE_OPERATION_PORT_IO). Use the

Gx5733ModuleSetOpeartionMode to set the module's operating mode.

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

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

Example

The following example reads the module port 0 data:

```
SHORT nStatus;
DWORD dwData;
Gx5733ModuleGetPort (nHandle, GX5733_PORT0, &dwData, &nStatus);
```

See Also

Gx5733ModuleSetOpeartionMode, **Gx5733ModuleGetOpeartionMode** **Gx5733ModuleSetPort**

Gx5733ModuleGetPxiTriggerBus

Purpose

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

Syntax

Gx5733ModuleGetPxiTriggerBus (*nHandle*, *nPort*, *pnPxiTrigBus*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnPxiTrigBus</i>	PSHORT	Returns the specified Module's eight output Strobe lines connections to the PXI Trigger Bus lines. The Strobe lines connections are 8 bits integer where each bit represents a PXI Trigger line. Bit 0 for line 0 and bit 7 for line 7. A high ('1') means the corresponding PXI Trigger line is connected; low ('0') means the corresponding PXI Trigger line is disconnected.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
SHORT nPxiTrigBus, nStatus;
```

```
Gx5733ModuleGetPxiTriggerBus (nHandle, &nPxiTrigBus, &nStatus);
```

See Also

Gx5733ModuleSetPxiTriggerBus, **Gx5733ModuleSetPxiTriggerLine**,
Gx5733GetStrobeToPxiTriggerBusState, **Gx5733SetStrobeToPxiTriggerLineState**.

Gx5733ModuleGetPxiTriggerLine

Purpose

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

Syntax

Gx5733ModuleGetPxiTriggerLine (*nHandle*, *nPort*, *nPxiTrigLine*, *pnMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nPxiTrigLine</i>	SHORT	PXI Trigger Line numbers are: 0 GX5733_PXI_TRIGGER_LINE0: Line 0 1 GX5733_PXI_TRIGGER_LINE1: Line 1 2 GX5733_PXI_TRIGGER_LINE2: Line 2 3 GX5733_PXI_TRIGGER_LINE3: Line 3 4 GX5733_PXI_TRIGGER_LINE4: Line 4 5 GX5733_PXI_TRIGGER_LINE5: Line 5 6 GX5733_PXI_TRIGGER_LINE6: Line 6 7 GX5733_PXI_TRIGGER_LINE7: Line 7
<i>pnMode</i>	PSHORT	PXI trigger line mode are: 0 GX5733_PXI_TRIGGER_LINE_DISCONNECTED: PXI trigger line is disconnected. 1 GX5733_PXI_TRIGGER_LINE_CONNECTED: PXI trigger line is Connected
<i>PnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733ModuleSetPxiTriggerBus** function to set all PXI-Trigger lines at once.

Example

The following example returns PXI Trigger line 1 connection mode:

```
SHORT nMode, nStatus;
```

```
Gx5733ModuleGetPxiTriggerLine (nHandle, GX5733_PXI_TRIGGER_LINE1, &nMode, &nStatus);
```

See Also

Gx5733ModuleSetPxiTriggerLine, **Gx5733ModuleSetPxiTriggerBus**,
Gx5733GetStrobeToPxiTriggerBusState, **Gx5733SetStrobeToPxiTriggerLineState**.

Gx5733ModuleGetRevision

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Revision.

Syntax

Gx5733ModuleGetRevision (*nHandle*, *nPort*, *pnRevision*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnRevision</i>	PSHORT	Return the Module's revision.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example reads module port 0 revision:

```
SHORT nRevision, nStatus;
```

```
Gx5733ModuleGetRevision (nHandle, GX5733_PORT0, &nRevision, &nStatus);
```

See Also

Gx5733GetModuleType

Gx5733ModuleGetState

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's State.

Syntax

Gx5733ModuleGetState (*nHandle*, *nPort*, *pnState*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnState</i>	PSHORT	Return the Module State as follow: 0 GX5733_MODULE_STATE_HALT: The Module is in Halt state. 1 GX5733_MODULE_STATE_RUN: The Module is in Run state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

The following example returns module port 0 state:

```
SHORT nState, nStatus;
```

```
Gx5733ModuleGetState (nHandle, GX5733_PORT0, &nState, &nStatus);
```

See Also

Gx5733ModuleRun, **Gx5733ModuleHalt**, **Gx5733ModuleSetExtStrobePolarity**,
Gx5733ModuleSetExtStrobeEnablePolarity, **Gx5733ModuleSetStrobeSource**,
Gx5733ModuleSetInternalStrobeSource

Gx5733ModuleGetStrobeSource

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's Strobe Source.

Syntax

Gx5733ModuleGetStrobeSource (*nHandle*, *nPort*, *pnSource*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnSource</i>	PSHORT	Return the Strobe Source as follow: 0 GX5733_MODULE_STROBE_INTERNAL: Strobe source is internal. 1 GX5733_MODULE_STROBE_EXTERNAL: Strobe source is external.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the Strobe Source is set to Internal see **Gx5733ModuleSetInternalStrobe** functions for more details.

If the Strobe Source is set to External see **Gx5733ModuleSetExtStrobePolarity** and **Gx5733ModuleSetExtStrobeEnablePolarity** functions for more details.

Example

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

```
SHORT nSource, nStatus;
```

```
Gx5733ModuleGetStrobeSource (nHandle, GX5733_PORT0, &nSource, &nStatus);
```

See Also

Gx5733ModuleSetStrobeSource, **Gx5733ModuleSetInternalStrobeClock**,
Gx5733ModuleGetInternalStrobeClock, **Gx5733ModuleSetExtStrobePolarity**,
Gx5733ModuleGetExtStrobePolarity, **Gx5733ModuleGetInternalStrobeSource**,
Gx5733ModuleGetExtStrobeEnablePolarity

Gx5733ModuleGetTermination

Applies To

GX5709, GX5711, GX5712

Purpose

Returns the Module termination mode.

Syntax

Gx5733ModuleGetTermination (*nHandle*, *nPort*, *pnTermination*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnTermination</i>	PSHORT	GX5709 module: Module bytes Terminations where bit 0 corresponds to byte 0, bit 1 to byte 1, bit 2 byte 2 and bit 3 to byte 3. A bit is set to hi '1' for Termination On and low '0' for Termination Off. GX5711 and GX5712 modules: Module Terminations, bit 0 is set to hi '1' for Termination On and low '0' for Termination Off.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

The following example returns Module port 0 terminators mode:

```
SHORT nTermination, nStatus;
Gx5733ModuleGetTermination (nHandle, GX5733_PORT0, &nTermination, &nStatus);
```

See Also

Gx5733ModuleSetTermination, **Gx5733ModuleGetType**

Gx5733ModuleGetType

Applies To

GX5701, GX5702, GX5704

Purpose

Returns the Module's type.

Syntax

Gx5733ModuleGetType (*nHandle*, *nPort*, *pnType*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnType</i>	PSHORT	Returned Module types, see comments for details.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

- 5700 GX5733_NO_MODULE_INSTALLED: No Module installed in the specified port.
- 5701 GX5733_DIGITAL_INPUT_LATCH: 32-channel digital input module with variable input threshold and rapid data comparison.
- 5702 GX5733_DIGITAL_OUTPUT_LATCH: 32-channel digital output module with TTL levels that can be used for digital control and propagation delay testing.
- 5704 GX5733_DIGITAL_POWER_OUTPUT_LATCH: Module capable of driving loads or driving signals over long distances.
- 5709 RS422 Port I/O Module is a differential RS422 32 I/O channels module.
- 5711 Bidirectional LVDS-TTL Converter Module is a 16 channels module converting signals from LVSD to TTL or vise versa.
- 5712 Bidirectional RS422-TTL Converter Module is a 16 channels module converting signals from RS422 to TTL or vise versa.
- 5715 GX5733_PORT_IN_ONLY: This is the most basic configuration. The port is only capable to serve as input.

When the module type is GX5733_PORT_IN_ONLY there are certain jumpers that needs to be installed.

Example

The following example reads module port 0 type:

```
SHORT nType, nStatus;
Gx5733GetPortByte (nHandle, GX5733_PORT0, &nType, &nStatus);
```

See Also

Gx5733ModuleGetRevision

Gx5733ModuleGetVThreshold

Applies To

GX5701

Purpose

Returns the Module's Threshold voltage.

Syntax

Gx5733ModuleGetVThreshold (*nHandle*, *nPort*, *pdVThreshold*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pdVThreshold</i>	PDOUBLE	Returned Module's threshold voltage. Threshold voltage range is from -30 to +30 volts with resolution of 16 bits.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

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

```
SHORT nStatus;  
DOUBLE dVThreshold;  
Gx5733ModuleSetVThreshold (nHandle, GX5733_PORT0, &dVThreshold, &nStatus);
```

See Also

Gx5733ModuleSetVThreshold, Gx5733ModuleGetType

Gx5733ModuleHalt

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the specified Module to Halt state.

Syntax

Gx5733ModuleHalt (*nHandle*, *nPort*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calling this function will take the Module from Run to Halt state. Use **Gx5733ModuleGetState** in order to monitor the module state.

Example

The following example set module port 0 to HALT mode:

```
SHORT nPolarity, nStatus;
Gx5733ModuleHalt (nHandle, GX5733_PORT0, &nStatus);
```

See Also

Gx5733ModuleGetState, **Gx5733ModuleRun**, **Gx5733ModuleSetExtStrobePolarity**,
Gx5733ModuleSetExtStrobeEnablePolarity, **Gx5733ModuleSetStrobeSource**,
Gx5733ModuleSetInternalStrobeSource

Gx5733ModuleRun

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the specified Module to Run mode.

Syntax

Gx5733ModuleRun (*nHandle*, *nPort*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Once in Run mode Module will start running depend on the strobe settings (i.e. Internal or External). Use **Gx5733ModuleGetState** in order to monitor the module state.

Example

The following example set module port 0 to RUN mode:

```
SHORT nPolarity, nStatus;
Gx5733ModuleRun (nHandle, GX5733_PORT0, &nStatus);
```

See Also

Gx5733ModuleGetState, **Gx5733ModuleHalt**, **Gx5733ModuleSetExtStrobePolarity**,
Gx5733ModuleSetExtStrobeEnablePolarity, **Gx5733ModuleSetStrobeSource**,
Gx5733ModuleSetInternalStrobeSource

Gx5733ModuleSetDirection

Applies To

GX5711, GX5712

Purpose

Sets the Module's channels direction.

Syntax

Gx5733ModuleSetDirection (*nHandle*, *nPort*, *nGroup*, *nDirection*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nGroup</i>	SHORT	Specified Module's port group number: 0 GX5733_MODULE_GROUP0: Module port group 0. 1 GX5733_MODULE_GROUP1: Module port group 1.
<i>nDirection</i>	SHORT	Module's channels direction mode: 0 GX5733_MODULE_DIRECTION_DIFFERENTIAL_TO_TTL: Module's channels direction is from differential to TTL. 1 GX5733_MODULE_DIRECTION_TTL_TO_DIFFERENTIAL: Module's channels direction is from TTL to differential.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The data that flows through the Module can be monitor by calling **Gx5733GetPort**.

Example

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

```
SHORT nStatus;
Gx5733ModuleSetDirection (nHandle, GX5733_PORT0, GX5733_MODULE_GROUP0,
    GX5733_MODULE_DIRECTION_DIFFERENTIAL_TO_TTL, &nStatus)
```

See Also

Gx5733ModuleGetDirection, **Gx5733ModuleSetDirectionSource**, **Gx5733GetPort**,
Gx5733ModuleSetInternalStrobeSource

Gx5733ModuleSetDirectionSource

Applies To

GX5711, GX5712

Purpose

Sets the Module's channels direction source.

Syntax

Gx5733ModuleSetDirectionSource (*nHandle*, *nPort*, *nSource*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nSource</i>	SHORT	Module's channels direction source: 0 GX5733_MODULE_DIRECTION_SOURCE_INTERNAL: Module's channels direction is set internally (software). 1 GX5733_MODULE_DIRECTION_SOURCE_EXTERNAL: Module's channels direction is set externally.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The data that flows through the Module can be monitor by calling **Gx5733GetPort**.

Example

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

```
SHORT nStatus;
Gx5733ModuleSetDirectionSource (nHandle, GX5733_PORT0, GX5733_MODULE_DIRECTION_SOURCE_INTERNAL,
                                &nStatus)
```

See Also

Gx5733ModuleGetDirectionSource, **Gx5733ModuleGetDirection**, **Gx5733GetPort**,
Gx5733ModuleSetInternalStrobeSource

Gx5733ModuleSetExtStrobeEnablePolarity

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's External Strobe Enable Polarity.

Syntax

Gx5733ModuleSetExtStrobeEnablePolarity (*nHandle*, *nPort*, *nPolarity*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nPolarity</i>	SHORT	External Strobe Enable Polarity mode: 0 GX5733_MODULE_STROBE_POSITIVE_EDGE: External Strobe is enabled on positive edge. 1 GX5733_MODULE_STROBE_NEGATIVE_EDGE: External Strobe is enabled on negative edge.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733SetPortWord** to write data to the port word when in Output direction.

Example

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

```
SHORT nPolarity, nStatus;
Gx5733ModuleGetExtStrobeEnPolarity (nHandle, GX5733_PORT0, &nPolarity, &nStatus);
```

See Also

Gx5733ModuleSetExtStrobePolarity, **Gx5733ModuleSetExtStrobeEnablePolarity**,
Gx5733ModuleSetStrobeSource, **Gx5733ModuleSetInternalStrobeSource**

Gx5733ModuleSetExtStrobePolarity

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's External Strobe Polarity.

Syntax

Gx5733ModuleSetExtStrobePolarity (*nHandle*, *nPort*, *nPolarity*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nPolarity</i>	SHORT	External Strobe Polarity mode: 0 GX5733_MODULE_STROBE_POSITIVE_EDGE: External Strobe active on positive edge. 1 GX5733_MODULE_STROBE_NEGATIVE_EDGE: External Strobe active on negative edge.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns modules port 0 external strobe polarity:

```
SHORT nPolarity, nStatus;
Gx5733ModuleGetExtStrobePolarity (nHandle, GX5733_PORT0, &nPolarity, &nStatus);
```

See Also

Gx5733ModuleGetExtStrobePolarity, **Gx5733ModuleSetExtStrobeEnablePolarity**,
Gx5733ModuleSetStrobeSource, **Gx5733ModuleSetInternalStrobeSource**

Gx5733ModuleSetHandshakeOutputMode

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Handshake output mode.

Syntax

Gx5733ModuleSetHandshakeOutputMode (*nHandle*, *nPort*, *nMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nMode</i>	SHORT	Handshake output mode: 0 GX5733_MODULE_HANDSHAKE_OUTPUT_DISABLE: Handshake output mode disabled. 1 GX5733_MODULE_HANDSHAKE_OUTPUT_ENABLE: Handshake output mode inabled.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX5733 can have one Modules installed. Each installed module can use Direct Output/Input or Strobed Output/Input with handshaking.

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

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

Example

The following example enable modules port 0 Handshake Output:

```
SHORT nStatus;
Gx5733ModuleSetHandshakeOutputMode (nHandle, GX5733_PORT0, GX5733_MODULE_HANDSHAKE_OUTPUT_ENABLE,
&nStatus);
```

See Also

Gx5733ModuleGetHandshakeOutputMode, **Gx5733ModuleSetExtStrobeEnablePolarity**,
Gx5733ModuleSetStrobeSource, **Gx5733ModuleSetInternalStrobeSource**

Gx5733ModuleSetInternalStrobeSource

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Internal Strobe Clock source.

Syntax

Gx5733ModuleSetInternalStrobeSource (*nHandle*, *nPort*, *nSource*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nSource</i>	SHORT	Returned source: 0. GX5733_MODULE_INTERNAL_STROBE_NOT_CONNECTED 1. GX5733_MODULE_INTERNAL_STROBE_T0_CLOCK0 2. GX5733_MODULE_INTERNAL_STROBE_T0_CLOCK1 3. Not used. 4. Not used. 5. Not used. 6. Not used. 7. GX5733_MODULE_INTERNAL_STROBE_T0_XTRIGGER0 8. GX5733_MODULE_INTERNAL_STROBE_T0_XTRIGGER1 9. GX5733_MODULE_INTERNAL_STROBE_T0_XTRIGGER2 10. GX5733_MODULE_INTERNAL_STROBE_T0_XTRIGGER3 11. GX5733_MODULE_INTERNAL_STROBE_T0_XTRIGGER4 12. GX5733_MODULE_INTERNAL_STROBE_T0_XTRIGGER5 13. GX5733_MODULE_INTERNAL_STROBE_T0_XTRIGGER6 14. GX5733_MODULE_INTERNAL_STROBE_T0_XTRIGGER7
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

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

```
SHORT nStatus;
Gx5733ModuleSetInternalStrobeSource (nHandle, GX5733_PORT0, GX5733_CLOCK0, &nStatus);
```

See Also

Gx5733ModuleSetInternalStrobeSource, **Gx5733SetInternalClock**, **Gx5733SetInternalClockEnable**, **Gx5733ModuleSetExtStrobeEnablePolarity**, **Gx5733ModuleSetExtStrobePolarity**, **Gx5733ModuleSetStrobeSource**

Gx5733ModuleSetOperationMode

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Operation mode.

Syntax

Gx5733ModuleSetOperationMode (*nHandle*, *nPort*, *nMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nMode</i>	SHORT	Module's Operation mode as follow: 0 GX5733_MODULE_OPERATION_BUFFERED: Default operation mode, in this mode the Module is using its buffer to read/write (depend on its type input or output). 1 GX5733_MODULE_OPERATION_PORT_IO: The user can use the Module's as a static port in or port out (depend on its type input or output)
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If Module type is either GX5702 or GX5704 and the Operation mode is Port I/O (GX5733_MODULE_OPERATION_PORT_IO) the outputs can be written statically.

If Module type is GX5701 and the Operation mode is Port I/O (GX5733_MODULE_OPERATION_PORT_IO) the inputs can be read statically.

Example

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

```
SHORT nStatus;
```

```
Gx5733ModuleGetOperationMode (nHandle, GX5733_PORT0, GX5733_MODULE_OPERATION_PORT_IO, &nStatus);
```

See Also

Gx5733ModuleGetOperationMode, **Gx5733ModuleSetOutputMode**, **Gx5733ModuleSetStrobeSource**

Gx5733ModuleSetOutputMode

Applies To

GX5702, GX5704

Purpose

Sets the Module's Output mode.

Syntax

Gx5733ModuleSetOutputMode (*nHandle*, *nPort*, *nMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nMode</i>	SHORT	Module's Output mode as follow: 0 GX5733_MODULE_OUTPUT_DISABLE: All the Module's 32 channels are in Tri-State mode. 1 GX5733_MODULE_OUTPUT_ENABLE: All the Module's 32 channels are enabled to output data..
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Outputs are set to Tri-State when in disabled.

Example

The following example enables the module port 0 output mode:

```
SHORT nStatus;
```

```
Gx5733ModuleGetOutputMode (nHandle, GX5733_PORT0, GX5733_MODULE_OUTPUT_ENABLE, &nStatus);
```

See Also

Gx5733ModuleSetOutputMode, **Gx5733ModuleSetStrobeSource**

Gx5733ModuleSetOutputStrobeWidth

Applies To

GX5702, GX5704

Purpose

Sets the Module's Output Strobe Width.

Syntax

Gx5733ModuleSetOutputStrobeWidth (*nHandle*, *nPort*, *nStrobeWidth*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nStrobeWidth</i>	SHORT	Integer value representing the Output Strobe Width in nSec. Strobe Width value range is 50 to 5000 with resolution of 50 nSec.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

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

```
SHORT nStatus;  
Gx5733ModuleSetOutputStrobeWidth (nHandle, GX5733_PORT0, 2, &nStatus);
```

See Also

Gx5733ModuleGetOutputStrobeWidth, **Gx5733ModuleSetStrobeSource**

Gx5733ModuleSetPort

Applies To

GX5702, GX5704

Purpose

Sets the Module's Port Data.

Syntax

Gx5733ModuleGetPort (*nHandle*, *nPort*, *dwData*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>dwData</i>	DWORD	Sets the port word value: 0-0xFFFFFFFF. Where bit 0 corresponds to channel 0 and bit 31 to channel 31.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows the module to be used as a port out. Prior to calling this function the module's operating mode needs to be set to Port I/O (GX5733_MODULE_OPERATION_PORT_IO). Use the

Gx5733ModuleSetOpeartionMode to set the module's operating mode.

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

Example

The following example sets the module port 0 data:

```
SHORT nStatus;
Gx5733ModuleSetPort (nHandle, GX5733_PORT0, 0xAAAA5555, &nStatus);
```

See Also

Gx5733ModuleSetOpeartionMode, **Gx5733ModuleGetOpeartionMode**, **Gx5733ModuleGetPort**

Gx5733ModuleSetPxiTriggerBus

Purpose

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

Syntax

Gx5733ModuleSetPxiTriggerBus (*nHandle*, *nPxiTrigBus*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nPxiTrigBus</i>	SHORT	Sets the specified Module's eight output Strobe lines connections to the PXI Trigger Bus lines. The Strobe lines connections are 8 bits integer where each bit represents a PXI Trigger line. Bit 0 for line 0 and bit 7 for line 7. A high ('1') means the corresponding PXI Trigger line is connected; low ('0') means the corresponding PXI Trigger line is disconnected.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
SHORT nStatus;

Gx5733ModuleSetPxiTriggerBus (nHandle, 0x5, &nStatus);
```

See Also

Gx5733ModuleGetPxiTriggerBus, **Gx5733ModuleSetPxiTriggerLine**,
Gx5733GetStrobeToPxiTriggerBusState, **Gx5733SetStrobeToPxiTriggerLineState**.

Gx5733ModuleSetPxiTriggerLine

Purpose

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

Syntax

Gx5733ModuleSetPxiTriggerLine (*nHandle*, *nPxiTrigLine*, *nMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nPxiTrigLine</i>	SHORT	PXI Trigger Line numbers are: 0 GX5733_PXI_TRIGGER_LINE0: Line 0 1 GX5733_PXI_TRIGGER_LINE1: Line 1 2 GX5733_PXI_TRIGGER_LINE2: Line 2 3 GX5733_PXI_TRIGGER_LINE3: Line 3 4 GX5733_PXI_TRIGGER_LINE4: Line 4 5 GX5733_PXI_TRIGGER_LINE5: Line 5 6 GX5733_PXI_TRIGGER_LINE6: Line 6 7 GX5733_PXI_TRIGGER_LINE7: Line 7
<i>nMode</i>	SHORT	PXI trigger line mode are: 0 GX5733_PXI_TRIGGER_LINE_DISCONNECTED: PXI trigger line is disconnected. 1 GX5733_PXI_TRIGGER_LINE_CONNECTED: PXI trigger line is Connected
<i>PnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733ModuleSetPxiTriggerBus** function to set all PXI-Trigger lines at once.

Example

The following example set PXI Trigger line 1 connection mode:

```
SHORT nStatus;
```

```
Gx5733ModuleSetPxiTriggerLine (nHandle, GX5733_PXI_TRIGGER_LINE1,  
                               GX5733_PXI_TRIGGER_LINE_CONNECTED, &nStatus);
```

See Also

Gx5733ModuleGetPxiTriggerLine, **Gx5733ModuleSetPxiTriggerBus**,
Gx5733GetStrobeToPxiTriggerBusState, **Gx5733SetStrobeToPxiTriggerLineState**.

Gx5733ModuleSetStrobeSource

Applies To

GX5701, GX5702, GX5704

Purpose

Sets the Module's Strobe Source.

Syntax

Gx5733ModuleSetStrobeSource (*nHandle*, *nPort*, *nSource*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>pnSource</i>	SHORT	Return the Strobe Source as follow: 0 GX5733_MODULE_STROBE_INTERNAL: Strobe source is internal. 1 GX5733_MODULE_STROBE_EXTERNAL: Strobe source is external.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the Strobe Source is set to Internal see **Gx5733ModuleSetInternalStrobe** function for more details.

If the Strobe Source is set to External see **Gx5733ModuleSetExtStrobePolarity** function for more details.

Example

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

```
SHORT nStatus;
Gx5733ModuleSetStrobeSource (nHandle, GX5733_PORT0, GX5733_MODULE_STROBE_EXTERNAL, &nStatus);
```

See Also

Gx5733ModuleGetStrobeSource, **Gx5733ModuleSetInternalStrobeClock**,
Gx5733ModuleGetInternalStrobeClock, **Gx5733ModuleSetExtStrobePolarity**,
Gx5733ModuleGetExtStrobePolarity, **Gx5733ModuleGetInternalStrobeSource**,
Gx5733ModuleGetExtStrobeEnablePolarity

Gx5733ModuleSetVThreshold

Applies To

GX5701

Purpose

Sets the Module's Threshold voltage.

Syntax

Gx5733ModuleSetVThreshold (*nHandle*, *nPort*, *dVThreshold*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>dVThreshold</i>	DOUBLE	Module's threshold voltage. Threshold voltage range is from -30 to +30 volts with resolution of 16 bits.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

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

```
SHORT nStatus;
```

```
Gx5733ModuleSetVThreshold (nHandle, GX5733_PORT0, 6.5, &nStatus);
```

See Also

Gx5733ModuleGetVThreshold, **Gx5733ModuleGetType**

Gx5733ModuleStep

Applies To

GX5701, GX5702, GX5704

Purpose

Run the specified port Module for given number of steps.

Syntax

Gx5733ModuleStep (*nHandle*, *nPort*, *dwSteps*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>dwSteps</i>	DWORD	The specified number of steps to execute.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example run port Module 0 for three steps:

```
SHORT nStatus;  
  
Gx5733ModuleStep (nHandle, GX5733_PORT0, 3, &nStatus);
```

See Also

Gx5733SetInternalClock, **Gx5733GetInternalClock**, **Gx5733ModuleGetType**

Gx5733ModuleSetTermination

Applies To

GX5709, GX5711, GX5712

Purpose

Sets the Module termination mode.

Syntax

Gx5733ModuleSetTermination (*nHandle*, *nPort*, *nTermination*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Module port 0 (GX5733_PORT0) only.
<i>nTermination</i>	SHORT	GX5709 module: Module bytes Terminations where bit 0 corresponds to byte 0, bit 1 to byte 1, bit 2 byte 2 and bit 3 to byte 3. A bit is set to hi '1' for Termination On and low '0' for Termination Off. GX5711 and GX5712 modules: Module Terminations, bit 0 is set to hi '1' for Termination On and low '0' for Termination Off.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
SHORT nStatus;
```

```
Gx5733ModuleSetTermination (nHandle, GX5733_PORT0, 1, &nStatus);
```

See Also

Gx5733ModuleGetTermination, **Gx5733ModuleGetType**

Gx5733Panel

Purpose

Opens a virtual panel used to interactively control the GX5733.

Syntax

Gx5733Panel (pnHandle, hwndParent, nMode, phwndPanel, pnStatus)

Parameters

Name	Type	Comments
<i>pnHandle</i>	PSHORT	Handle to a GX5733 board.
<i>hwndParent</i>	HWND	Panel parent window handle. A value of 0 sets the desktop as the parent window.
<i>nMode</i>	SHORT	The mode in which the panel main window is created. 0 for modeless window and 1 for modal window.
<i>phwndPanel</i>	HWND	Returned window handle for the panel.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

If the mode is set to modal dialog (*nMode*=1), the panel will disable the parent window (*hwndParent*) and the function will return only after the window was closed by the user. In that case, the *pnHandle* may return the handle created by the user using the panel Initialize dialog. This handle may be used when calling other GX5733 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;

Gx5733Panel(&nHandle, 0, 1, &dwPanel, &nStatus);
```

See Also

Gx5733Initialize, GxPioGetErrorString

Gx5733Reset

Purpose

Resets the GX5733 board to its default settings.

Syntax

Gx5733Reset (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5733 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function resets all digital I/O ports. Digital I/O ports are set to input, clock source is zero, gate is enabled, Clock are set to 33Mhz with divider set to 1.

Example

The following example initializes and resets the GX5733 board:

```
Gx5733Initialize (1, &nHandle, &nStatus);  
Gx5733Reset (nHandle, &nStatus);
```

See Also

Gx5733Initialize, **Gx5733ResetPort**.

Gx5733ResetPort

Purpose

Resets the specified digital I/O port.

Syntax

Gx5733ResetPort (*nHandle*, *nPort*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port number: 0-3.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

The following example resets port 1:

```
Gx5733ResetPort (nHandle, 1, &nStatus);
```

See Also

Gx5733Reset, **Gx5733Initialize**

Gx5733SetInternalClock

Purpose

Sets internal clock source and divider.

Syntax

Gx5733SetInternalClock (*nHandle*, *nClock*, *nSource*, *dwDivider*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5733 board.
<i>nClock</i>	SHORT	Clock number: 0. GX5733_CLOCK0: Internal Clock 0 1. GX5733_CLOCK1: Internal Clock 1
<i>nSource</i>	SHORT	Internal Clock 0 sources are: 4. GX5733_INTERNAL_CLOCK0_NOT_CONNECTED: Clock not connected. 5. GX5733_INTERNAL_CLOCK0_T0_PCI_CLOCK_33MHZ: Internal Clock 0 connected to 33MHz PCI Clock. 6. GX5733_INTERNAL_CLOCK0_T0_PXI_CLOCK_10MHZ: Internal Clock 0 connected to 10MHz PXI Clock. 7. GX5733_INTERNAL_CLOCK0_T0_STAR_TRIGGER: Internal Clock 0 connected to the PXI Start Trigger. Internal Clock 1 sources are: 4. GX5733_INTERNAL_CLOCK1_NOT_CONNECTED: Clock not connected. 5. GX5733_INTERNAL_CLOCK1_T0_PCI_CLOCK_33MHZ: Internal Clock 1 connected to 33MHz PCI Clock. 6. GX5733_INTERNAL_CLOCK1_T0_PXI_CLOCK_10MHZ: Internal Clock 0 connected to 10MHz PXI Clock. 7. GX5733_INTERNAL_CLOCK1_T0_STAR_TRIGGER: Internal Clock 0 connected to the PXI Start Trigger.
<i>dwDivider</i>	DWORD	Divider value: 1-0x8000000
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733GetInternalClock** to retrieve the internal clock settings.

Example

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

```
Gx5733SetInternalClock (nHandle, 0, 1, 20000, &nStatus);
```

See Also

Gx5733GetInternalClock

Gx5733SetInternalClockEnable

Purpose

Sets the specified internal clock enable mode.

Syntax

Gx5733SetInternalClockEnable (*nHandle*, *nClock*, *nEnable*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX5733 board.
<i>nClock</i>	SHORT	Clock number: 0. GX5733_CLOCK0: Internal Clock 0 1. GX5733_CLOCK1: Internal Clock 1
<i>nEnable</i>	SHORT	Returned clock enable mode:: 0. GX5733_DISABLE_INTERNAL_CLOCK: Disable the specified clock. 1. GX5733_ENABLE_INTERNAL_CLOCK: Enable the specified clock.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example enable the internal clock 0:

```
Gx5733SetInternalClockEnable (nHandle, GX5733_CLOCK0, GX5733_ENABLE_INTERNAL_CLOCK, &nStatus);
```

See Also

Gx5733GetInternalClockEnable, **Gx5733GetInternalClock**

Gx5733SetPort

Purpose

Write data to the specified port.

Syntax

Gx5733SetPort (*nHandle*, *nPort*, *dwValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port number: 1-3.
<i>dwValue</i>	DWORD	Port value to set: 0 to 4,294,967,295 (0-0xFFFFFFFF), where each bit represents a channel. Bit 0 for channel 0 and bit 31 for channel 31.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Use the **Gx5733GetPort** to read data from the port.

Example

The following example writes 0xFF00FF to port 1:

```
Gx5733SetPort (nHandle, 1, 0xFF00FF, &nStatus);
```

See Also

Gx5733GetPort, **Gx5733SetPortBit**, **Gx5733SetPortByte**, **Gx5733SetPortWord**, **Gx5733SetPortDirection**

Gx5733SetPortBit

Purpose

Write a specific bit of data to a port channel.

Syntax

Gx5733SetPortBit (*nHandle*, *nPort*, *nBit*, *ucValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port number: 1-3.
<i>nBit</i>	SHORT	Port channel number: 0 to 31.
<i>ucValue</i>	BYTE	Channel value: 0 for low state and 1 for hi state.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Use the **Gx5733GetPortBit** to read data from a port bit.

Example

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

```
Gx5733GetPort (nHandle, 1, 30, 1, &nStatus);
```

See Also

Gx5733GetPortBit, **Gx5733SetPortByte**, **Gx5733SetPortWord**, **Gx5733SetPort**

Gx5733SetPortByte

Purpose

Write a specific byte of data to a port byte group.

Syntax

Gx5733SetPortByte (*nHandle*, *nPort*, *nByte*, *ucValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port number: 1-3.
<i>nByte</i>	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte.
<i>ucValue</i>	BYTE	Value to set 0-255. Each bit represents the specific channel state bit 0 for lowest channel number.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Only byte set to output direction will be written.

Use the **Gx5733GetPortByte** to read data from a port byte.

Example

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

```
Gx5733GetPortByte (nHandle, 1, 2, 0x5, &nStatus);
```

See Also

Gx5733GetPortByte, **Gx5733SetPortBit**, **Gx5733SetPortWord**, **Gx5733SetPort**

Gx5733SetPortByteDirection

Purpose

Set the specific port byte direction.

Syntax

Gx5733SetPortByteDirection (*nHandle*, *nPort*, *nByte*, *bInOut*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port number: 1-3.
<i>nByte</i>	SHORT	Port byte number: 0-3, 0 for the port low order byte and 3 for the high order byte.
<i>bInOut</i>	BOOL	Byte direction: 0. (FALSE) – Input. 1. (TRUE) – Output.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
Gx5733SetPortByteDirection (nHandle, 1, 3, TRUE, nStatus);
```

See Also

Gx5733GetPortByteDirection, Gx5733SetPortDirection

Gx5733SetPortDirection

Purpose

Set the specified port direction.

Syntax

Gx5733SetPortDirection (*nHandle*, *nPort*, *nDirection*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port number: 1-3.
<i>nDirection</i>	SHORT	Port bytes directions where bit 0 corresponds to byte 0, bit 1 to byte 1, bit 2 byte 2 and bit 3 to byte 3. A bit is set to hi '1' for Input and low '0' for output.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
Gx5733GetPortByteDirection (nHandle, 1, 6, nStatus);
```

See Also

Gx5733GetPortDirection, **Gx5733SetPortByteDirection**

Gx5733SetPortWord

Purpose

Write a specific word of data to a port.

Syntax

Gx5733SetPortWord (*nHandle*, *nPort*, *nWord*, *wValue*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nPort</i>	SHORT	Digital I/O port number: 1-3.
<i>nWord</i>	SHORT	Port word number: 0. GX5733_PORT_LOW_WORD: port low order word (bytes 0 and 1). 1. GX5733_PORT_HIGH_WORD: high order word (bytes 2 and 3).
<i>wValue</i>	WORD	Port word value: 0 to 65,535 (0-0xFFFF). Where bit 0 corresponds to channel 0 and bit 15 to channel 15.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

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

Example

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

```
Gx5733GetPortByte (nHandle, 1, 2, 0x5, &nStatus);
```

See Also

Gx5733GetPortWord, **Gx5733SetPortBit**, **Gx5733SetPortByte**, **Gx5733SetPort**

Gx5733SetStrobeToPxiTriggerBusState

Purpose

Sets all Strobe to PXI Trigger lines Output Mode.

Syntax

Gx5733SetStrobeToPxiTriggerBusState (*nHandle*, *nBusState*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a GX5733 board.
<i>nBusState</i>	SHORT	Sets all eight Strobe to PXI Trigger Bus lines Output mode, 8 bits integer where each bit represents a PXI Trigger line. Bit 0 for line 0 and bit 7 for line 7. A high ('1') means the corresponding PXI Trigger line is enable, low ('0') means the corresponding PXI Trigger line is disabled.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Example

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

```
SHORT nStatus;
```

```
Gx5733SetStrobeToPxiTriggerBusState (nHandle, 0x5, &nStatus);
```

See Also

Gx5733GetStrobeToPxiTriggerBusState, **Gx5733SetStrobeToPxiTriggerLineState**.

Gx5733SetStrobeToPxiTriggerLineState

Purpose

Sets the specified Strobe to PXI Trigger Line output mode.

Syntax

Gx5733SetStrobeToPxiTriggerLineState (*nHandle*, *nLine*, *bMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>NHandle</i>	SHORT	Handle to a GX5733 board.
<i>Nline</i>	SHORT	PXI Trigger Line numbers are: 0 GX5733_PXI_TRIGGER_LINE0: Line 0 1 GX5733_PXI_TRIGGER_LINE1: Line 1 2 GX5733_PXI_TRIGGER_LINE2: Line 2 3 GX5733_PXI_TRIGGER_LINE3: Line 3 4 GX5733_PXI_TRIGGER_LINE4: Line 4 5 GX5733_PXI_TRIGGER_LINE5: Line 5 6 GX5733_PXI_TRIGGER_LINE6: Line 6 7 GX5733_PXI_TRIGGER_LINE7: Line 7
<i>bMode</i>	BOOL	PXI trigger line mode are: 0 GX5733_PXI_TRIGGER_LINE_DISCONNECTED: PXI trigger line is disconnected. 1 GX5733_PXI_TRIGGER_LINE_CONNECTED: PXI trigger line is Connected
<i>PnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use the **Gx5733SetStrobeToPxiTriggerBusState** function to set all PXI-Trigger lines at once.

Example

The following example set PXI Trigger line 1 mode:

```
SHORT nStatus;
```

```
Gx5733SetStrobeToPxiTriggerLineState(nHandle, GX5733_PXI_TRIGGER_LINE1, GX5733_PXI_TRIGGER_LINE_CONNECTED,
&nStatus);
```

See Also

Gx5733GetStrobeToPxiTriggerBusState, **Gx5733SetStrobeToPxiTriggerLineState**.

GxPioGetDriverSummary

Purpose

Returns the driver name and version.

Syntax

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

Parameters

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

Comments

The returned string is: "GXPIO Driver for GX5733 and GX5732. Version 2.0, Copyright © Marvin Test Solutions 2003.".

Example

The following example prints the driver version:

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

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

See Also

GxPioGetErrorString

GxPioGetErrorString

Purpose

Returns the error string associated with the specified error number.

Syntax

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

Parameters

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

Comments

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

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

Resource Errors

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

General Parameter Errors

-20	Invalid or unknown error number
-21	Invalid parameter
-22	Illegal slot number
-23	Illegal board handle

- 24 Illegal string length
- 25 Illegal operation mode
- 26 Parameter is out of the allowed range

Parameter Errors

- 40 Invalid port
- 41 Invalid word
- 42 Invalid byte
- 43 Invalid bit
- 44 Invalid counter
- 45 Invalid input load control
- 46 Invalid counter or all terminal counts and clocks
- 47 Invalid terminal count mode
- 48 Invalid clock source
- 49 Invalid clock internal number
- 50 Invalid clock internal source
- 51 Invalid gate source
- 52 Invalid clock divider value
- 53 Invalid internal strobe source, source can be between 0-7
- 54 Invalid strobe source, strobe source can be 0-Internal or 1-External
- 55 Invalid strobe polarity, strobe polarity can be 0-Positive Edge or 1-Negative Edge
- 56 Invalid Buffer size, Buffer size can be between 1-100
- 57 Invalid Group number
- 58 Invalid direction

Execution error

- 80 Unable to load Counter. Counter Port is not in Input
- 81 Unable to empty the specified Module's buffer
- 82 Testing the specified Module's buffer failed
- 83 Failed to write data correctly to the specified Module's buffer
- 84 Failed to read data correctly from the specified Module's buffer

Example

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

```
CHAR    sz[256];
SHORT   nStatus, nHandle;
..
Gx5733Initialize (3, &Handle, &Status);
if (nStatus<0)
{
    GxPioGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz);          // prints the error string returns
```

}

Index

A

Architecture 1, 5, 12
 ATEasy 21, 26, 27, 43, 44

B

Block Diagram 5, 6, 12
 Board Description 1, 4
 Board Handle 45
 Borland 26, 43, 44
 Borland-Delphi 44

C

C/C++ 26, 27, 43
 C++ 43
 Connectors 25, 29, 30, 31, 32, 33, 34, 35, 36
 Corrupt files 20

D

Delphi 26, 43, 44
 Directories 21
 Distributing 46
 Driver
 Directory 21
 Files 21

E

Error-Handling 46
 Example 27

F

Functions
 Boards-Supported 44

G

Gx5733GetBoardSummary 61
 Gx5733GetInternalClock 62
 Gx5733GetInternalClockEnable 63
 Gx5733GetPort 64
 Gx5733GetPortBit 65
 Gx5733GetPortByte 66
 Gx5733GetPortByteDirection 67

Gx5733GetPortDirection 68
 Gx5733GetPortWord 69
 Gx5733GetStrobeToPxiTriggerBusState 70
 Gx5733GetStrobeToPxiTriggerLineState 71
 Gx5733Initialize 38, 72
 Gx5733InitializeVisa 22, 38, 73
 Gx5733ModuleBufferClear 74
 Gx5733ModuleBufferGetMode 75
 Gx5733ModuleBufferGetState 76
 Gx5733ModuleBufferReadData 77
 Gx5733ModuleBufferSetFlagMode 78
 Gx5733ModuleBufferSetMode 79
 Gx5733ModuleBufferTest 80
 Gx5733ModuleBufferWriteData 81
 Gx5733ModuleGetDirection 82
 Gx5733ModuleGetDirectionSource 83
 Gx5733ModuleGetExtStrobeEnablePolarity 84
 Gx5733ModuleGetExtStrobeEnableState 85
 Gx5733ModuleGetExtStrobePolarity 86
 Gx5733ModuleGetExtStrobeState 87
 Gx5733ModuleGetHandshakeOutputMode 88
 Gx5733ModuleGetInternalStrobeSource 89
 Gx5733ModuleGetOperationMode 90
 Gx5733ModuleGetOutputMode 91
 Gx5733ModuleGetOutputStrobeWidth 92
 Gx5733ModuleGetPort 93
 Gx5733ModuleGetPxiTriggerBus 94
 Gx5733ModuleGetPxiTriggerLine 95
 Gx5733ModuleGetRevision 96
 Gx5733ModuleGetState 97
 Gx5733ModuleGetStrobeSource 98
 Gx5733ModuleGetTermination 99
 Gx5733ModuleGetType 100
 Gx5733ModuleGetVThreshold 101
 Gx5733ModuleHalt 102

Gx5733ModuleRun	103	GXPIO.DLL	26, 43, 44
Gx5733ModuleSetDirection	104	GXPIO.EXE	20
Gx5733ModuleSetDirectionSource	105	GXPIO.H	26, 43
Gx5733ModuleSetExtStrobeEnablePolarity	106	GXPIO.LIB	43
Gx5733ModuleSetExtStrobePolarity	107	GXPIO.PAS.....	26, 44
Gx5733ModuleSetHandshakeOutputMode.....	108	GXPIO.VB	26
Gx5733ModuleSetInternalStrobeSource	109	GXPIO64.DLL	26
Gx5733ModuleSetOperationMode.....	110	GXPIO64.LIB	43
Gx5733ModuleSetOutputMode	111	GXPIOBC.LIB	26, 43
Gx5733ModuleSetOutputStrobeWidth	112	GxPioGetDriverSummary	44, 46, 133
Gx5733ModuleSetPort	113	GxPioGetErrorString	46, 57, 134, 135
Gx5733ModuleSetPxiTriggerBus	114	GXPIO64.EXE	46
Gx5733ModuleSetPxiTriggerLine	115	GxXXXXInitialize.....	44, 45
Gx5733ModuleSetStrobeSource	116	GxXXXXInitializeVisa	44, 45
Gx5733ModuleSetTermination	119	H	
Gx5733ModuleSetVThreshold.....	117	Handle	23, 24, 44, 45
Gx5733ModuleStep.....	118	HW	21, 25, 43, 46
Gx5733Panel	46, 120	I	
Gx5733Reset	121	I/O.....	33, 58
Gx5733ResetPort.....	122	Installation Directories	21
Gx5733SetInternalClock	123	Installation:	23, 25
Gx5733SetInternalClockEnable	124	J	
Gx5733SetPort	125	J1	29, 31, 33, 34
Gx5733SetPortBit.....	126	J2	5, 29, 35
Gx5733SetPortByte	127	J3	29, 35
Gx5733SetPortByteDirection	128	J4	5, 29, 35
Gx5733SetPortDirection	129	J7	31, 32
Gx5733SetPortWord	130	J8	31, 32
Gx5733SetStrobeToPxiTriggerBusState	131	J9	31, 32
Gx5733SetStrobeToPxiTriggerLineState	132	JB1-JB12	30
GXPIO.....	1, 20	JP1	29, 30
Driver-Description	43	JP2	29
Header-file	43	Jumpers.....	29
Help-File-Description	26	L	
Panel-File-Description	26	Listing.....	48
Supported-Development-Tools.....	43	M	
GXPIO.BAS	26, 43	Module.....	58

N*nHandle*44**O**

OnError.....44

P

Panel 20, 26, 37, 39, 40, 41, 44, 46, 120

Panel About Page.....41

Panel Advanced Page40

Part / Model Number36

Pascal.....26, 43, 44

PCI.....21

Plug & Play.....25

pnStatus57

port byte.....58, 66, 67, 122, 127

Program-File-Descriptions26

Programming

Borland-Delphi44

Error-Handling.....46

Panel-Program46

Programming Examples46

PXI.....3, 4, 5, 17, 19, 23, 24, 25, 44

PXI Trigger.....60

PXI/PCI Explorer38, 44, 45, 72, 73

PXIeSYS.INI38

PXISYS.INI.....38

R

README.TXT21, 26

Readme-File26

Reset46

RS42212

S

Sample47, 48

SCSI.....5

Setup20, 21, 26

Setup Maintenance20

Setup-and-Installation.....19

Slot.....19, 23, 25, 38, 44, 47

Specifications1, 17

System

Directory21

System-Requirements19

T

TTL.....3, 5

V

Virtual Panel 20, 26, 37, 38, 39, 40, 41, 120

Initialize Dialog38

VISA.....21, 22, 38, 44, 45, 58, 72, 73

Visual Basic.....43

Visual C++26, 43

