

GX2065

6 ½ Digit Digital Multimeter with Digitizer GXDMM Software

User's Guide

Last updated February 11, 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> for more information about Marvin Test Solutions products, services and support options. Our web site contains sections describing support options and application notes, as well as a download area for downloading patches, example, patches and new or revised instrument drivers. To submit a support issue including suggestion, bug report or 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.

Our address (check our website for latest address): Marvin Test Solutions, 1770 Kettering, Irvine, CA 92614, USA.

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 © 2011-2015 by Marvin Test Solutions, Marvin Test Systems, 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 (prior name is Geotest - Marvin Test Systems Inc.)
C++ Builder, Delphi	Embarcadero Technologies Inc.
LabView, LabWindows tm /CVI	National Instruments
Microsoft Developer Studio, Microsoft Visual C++, Microsoft Visual Basic, .NET, Windows 95, 98, NT, ME, 2000, XP, VISTA, Windows 7 and 8	Microsoft Corporation

All other trademarks are the property of their respective owners.

Table of Contents

Safety and Handling.....	i
Warranty	i
If You Need Help.....	i
Disclaimer	i
Copyright	i
Trademarks	ii
Table of Contents	iii
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	3
High Energy Circuit Safety Precautions	5
Performance considerations	5
Warm-up	5
Line Cycle Selection and Reading Rate.....	6
Theory of Operation and Architecture	7
Voltage measurements (DCV and ACV)	8
Current measurements (DCI and ACI)	8
Resistance measurements (2 and 4 wire)	9
Frequency and period measurements	10
Virtual Panel Description.....	11
Virtual Panel Initialize Dialog	12
Virtual Panel Setup Page.....	13
Virtual Panel Advanced Page.....	15
Virtual Panel About Page	16
Chapter 3 - Installation and Connections	17
Getting Started	17
Interfaces and Accessories	17
Unpacking and Inspection.....	17

System Requirements	17
DMM Connections	18
Installation of the GXDMM Software	20
Setup Maintenance Program	20
Overview of the GXDMM 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
Chapter 4 - Programming the Board	27
The GXDMM Driver.....	27
Programming Using C/C++ Tools	27
Programming Using Visual Basic and Visual Basic .NET	27
Programming Using Pascal/Delphi	27
Programming GXDMM Boards Using ATEasy®	28
Programming Using LabView and LabView/Real Time	28
Using and Programming under Linux.....	28
Using the Signametrics SM204x and SM2060 Compatibility Layer.....	28
Using the GXDMM driver functions	30
Initialization, HW Slot Numbers and VISA Resource	30
Board Handle	31
Reset.....	31
Error Handling	31
Driver Version.....	31
Programming Examples.....	32
Distributing the Driver	32
Sample Programs	33
Sample Program Listing	34
Chapter 5 - Calibration.....	41
Introduction.....	41
Hardware Requirements.....	41
Warm up Time	41
Calibration Licensing.....	42

GXDMM Functions used for Calibration	43
Calibration Procedure	44
Initial Calibration Procedure	44
DC Voltage Calibration.....	44
AC Voltage DC Coupled Calibration.....	44
AC Voltage AC Coupled Calibration.....	45
DC Current Calibration	46
AC Current DC Coupled Calibration	46
AC Current AC Coupled Calibration	47
Resistance 2 Wire Calibration.....	47
Resistance 4 Wire Calibration.....	48
Resistance Open Calibration	49
RRef Calibration	49
Final Calibration Procedure	49
Chapter 6 - Function Reference.....	51
Introduction.....	51
GXDDMM Functions	52
General Functions	52
GxDmmAbortReading.....	54
GxDmmGetACClockDivider	55
GxDmmGetACMath.....	56
GxDmmGetACMinFrequency.....	57
GxDmmGetAperture	58
GxDmmGetAutoRange	59
GxDmmGetAutoZero	60
GxDmmGetBoardSummary	61
GxDmmGetCalibrationInfo	62
GxDmmGetCalibrationSet.....	64
GxDmmGetDriverSummary	65
GxDmmGetErrorString	66
GxDmmGetExternalTriggers.....	69
GxDmmGetFunction	71
GxDmmGetLineFrequency.....	72
GxDmmGetMinMax.....	73
GxDmmGetRange	74
GxDmmGetReadArrayStatus	77
GxDmmGetReadStatus.....	78

GxDmmGetRelative	79
GxDmmGetResolution	80
GxDmmGetTriggerMode	81
GxDmmInitialize	83
GxDmmInitializeVisa	84
GxDmmMeasure.....	85
GxDmmMeasureEx	86
GxDmmRead	87
GxDmmReadArray	88
GxDmmReadEx.....	90
GxDmmReset	91
GxDmmRestoreFactoryCalibration	92
GxDmmPanel	93
GxDmmSetACClockDivider	94
GxDmmSetACMath	95
GxDmmSetACMinFrequency	96
GxDmmSetAperture	97
GxDmmSetAutoRange	98
GxDmmSetAutoZero.....	99
GxDmmSetCalibrationMeasurements	100
GxDmmSetCalibrationSet	104
GxDmmSetExternalTriggers	105
GxDmmSetFunction	107
GxDmmSetLineFrequency	108
GxDmmSetRange	109
GxDmmSetRelative	112
GxDmmSetResolution	113
GxDmmSetTriggerMode	114
GxDmmTrig	116
GxDmmWriteCalibrationEEPROM	117
Appendix A - Specifications.....	119
Appendix B – Signametrics DMM Function Comparisons	127
Index	131

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 GX2065 instrument. This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and basic knowledge about analog instrumentation. .





This manual also provides programming information using the GX2065 driver (referred in this manual **GXDMM**). Therefore, good understanding of programming development tools and languages may be necessary.

Manual Organization

The GX2065 manual is organized in the following manner:

Chapter	Content
Chapter 1 - Introduction	Introduces the GX2065 manual. Lists all the supported board and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX2065 features, board description, its architecture, specifications and the panel description and operation.
Chapter 3 –Installation and Connections	Provides instructions on how to install a GX2065 board and the GXDMM software.
Chapter 4 – Programming the Board	Provides a list of the GXDMM software driver files, general purpose and generic driver functions, and programming methods. Discusses supported application development tools and programming examples.
Chapter 5 – Calibration	Provides information regarding calibration procedures and software.
Chapter 6 – Functions Reference	Provides a list of the GX2065 driver functions. Each function description provides syntax, parameters, and any special programming comments.
Appendix A - Specifications	Provides the GX2065 DMM specifications

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 GX2065 is a 6 ½ Digit Multimeter with 3 MS/s Digitizer. The DMM supports Volts DC, Amps DC, Two Wire Resistance, Four Wire Resistance, and Frequency measurements. The Digitizer supports Volts AC (AC/DC Coupled), Amps AC (AC/DC Coupled) measurements and waveform capture functions. Real time, data acquisition measurements are also supported by two hardware buffers. External and PXI triggering capabilities are supported.

Features

The GX2065 offers the following capabilities:

- Up to 6 ½ Digits of Resolution
- Time Stamping capability in hardware with 0.1uS resolution (based on PC Time synchronization)
- 1000 sample buffer for DMM measurements
- 8192 sample buffer for Digitizer measurements
- Continuous, Timer, and Software triggered acquisition modes
- Supports External Triggering and PXI Triggering capabilities
- Ability to generate PXI triggers and supports multi DMM synchronized operation

Applications

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

Board Description

The GX2065 is a 3U PXI instrument card. The front connectors consist of four banana type connectors and a DIN connector (see Figure 2-1). For more information about the connectors and their location on the board refer to Chapter 3 – Installation and Connections.



Warning - The GX2065's measurement inputs are rated for Measurement Category II circuits. The equipment cannot be connected to circuits with transients greater than 2.5kV peak or working voltages greater than 300 Volts. Current measurements are only rated to 250 Volts and 2 Amps DC or peak AC. Do not connect measurement inputs to Category III or IV circuits.

The GX2065 can make the following measurements:

- **DCV** — DC voltage measurements from 0.1μV to 300V
- **ACV** — AC voltage measurements from 0.1μV to 425Vp
- **DCI** — DC current measurements from 10nA to 2A
- **ACI** — AC current measurements from 3μA to 2A peak
- **2-wire ohms** — 2-wire resistance measurements from 100μ ohms to 100M ohms

- **4-wire ohms** — 4-wire resistance measurements from 100 μ ohms to 100M ohms
- **Frequency** — Frequency measurements from 3Hz to 500kHz
- **Period** — Period measurements from 333ms to 2 μ s

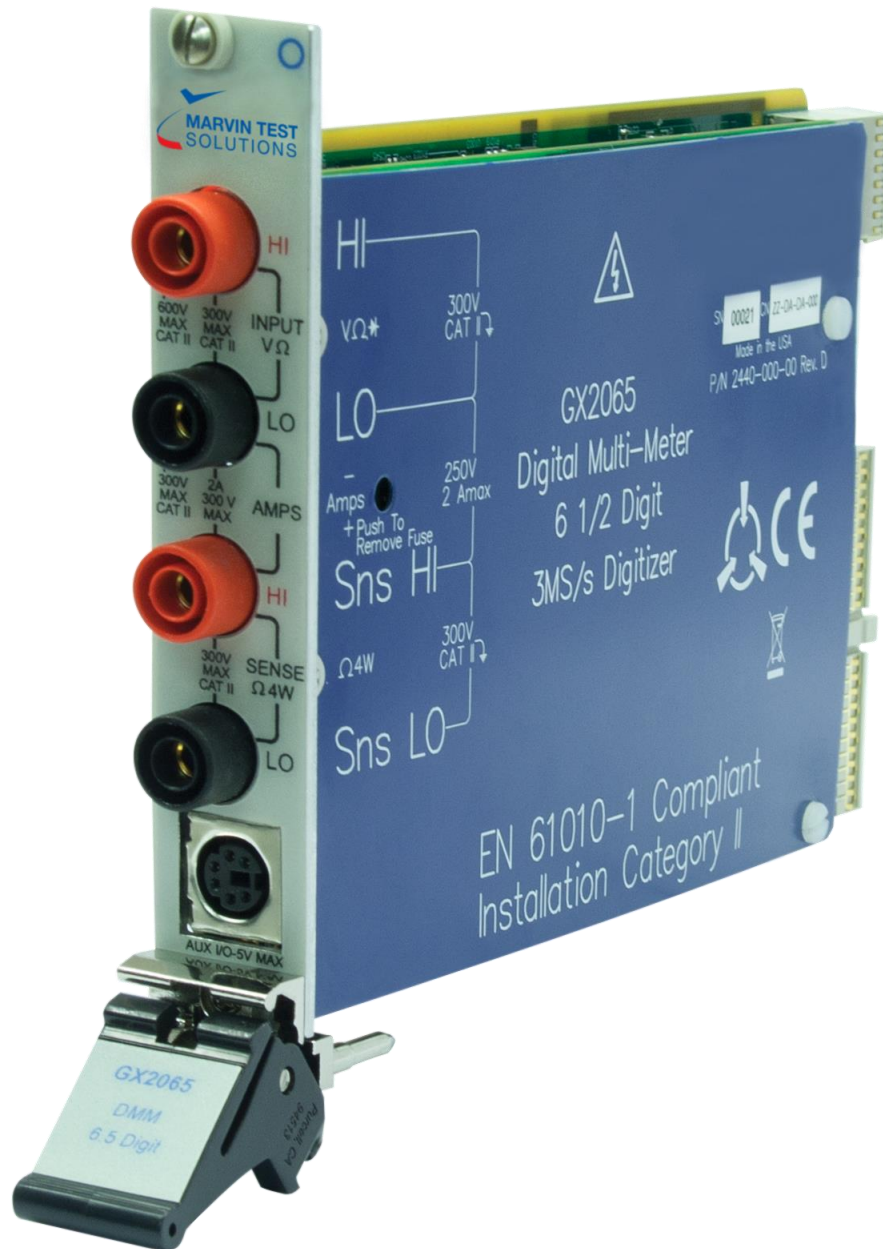


Figure 2-1: GX2065 Board

High Energy Circuit Safety Precautions

To optimize safety when measuring voltages in high-energy distribution circuits, read and use the directions in the following warning.



Warning - Dangerous arcs of an explosive nature in a high-energy circuit can cause severe personal injury or death. If the Multimeter is connected to a high-energy circuit when set to a current range or low resistance range, the circuit is virtually shorted. Dangerous arcing can result even when the Multimeter is set to a voltage range if the minimum voltage spacing is reduced in the external connections.

As described in the International Electro technical Commission (IEC) Standard IEC 61010-1:2001, the GX2065's measurement inputs are rated for Category II measurements..

When making measurements in high energy circuits, use test leads that meet the following requirements:

- Test leads should be fully insulated
- Only use test leads that can be connected to the circuit (e.g., alligator clips, spade lugs, etc.) for hands-off measurements.
- Do not use test leads that decrease voltage spacing. These diminish arc protection and create a hazardous condition.



Warning - For the front panel inputs, the maximum common-mode voltage (voltage between INPUT LO and the chassis ground) is 300VDC or rms. Exceeding this value may cause a breakdown in insulation, creating a shock hazard. Use the following sequence when testing power circuits:

1. De-energize the circuit using the regular installed connect-disconnect device, for example, by removing the device's power cord or by turning off the power switch.
2. Attach the test leads to the circuit under test. Use appropriate safety rated test leads for this application. If over 42V, uses double insulated test leads or add an additional insulation barrier for the operator.
3. Set the Multimeter to the proper function and range.
4. Energize the circuit using the installed connect-disconnect device and make measurements without disconnecting the Multimeter.
5. De-energize the circuit using the installed connect-disconnect device.
6. Disconnect the test leads from the circuit under test.

Performance considerations



Note - For maximum system performance, it is recommended that all measurement cables be limited to less than 3 meters

Warm-up

After the GX2065 is turned on, it must be allowed to warm up for at least 30 minutes to allow its internal temperature to stabilize. If the instrument has been exposed to extreme temperatures, allow extra warm-up time.

Line Cycle Selection and Reading Rate

Selecting the correct line cycle value (50 or 60 Hz) increases common mode and normal mode noise rejection. The instrument's function calls support the selection of various integration rates or PLC (power line cycles) values as well as supporting the selection of 50 Hz or 60 Hz for the line cycle rate. PLC values and resolution are detailed in the following table for various measurements.

Measurement	Measurement Conditions	Resolution (Digits)	Reading Rate @ 60Hz line (R/s)	Power Line Cycles	Aperture Time (seconds)
DC Voltage	Auto Zero Off	6.5	5	10	0.16667
	Auto Zero On	6.5	35	1	0.01667
	Auto Zero Off	6.5	50	1	0.01667
	Auto Zero Off	5.5	150	0.1	0.00167
	1K readings, Auto Zero Off	5.5	500	0.1	0.00167
	1K readings, Auto Zero Off	4.5	3000	0.006	0.00010
	1K readings, Auto Zero On	3.5	3500	0.002	0.00003
DC Amps	Auto Zero On	6.5	5	10	0.16667
	Auto Zero On	6.5	35	1	0.01667
	Auto Zero Off	6.5	50	1	0.01667
	Auto Zero Off	5.5	150	0.1	0.00167
	1K readings, Auto Zero Off	5.5	500	0.1	0.00167
	1K readings, Auto Zero Off	4.5	3000	0.006	0.00010
	1K readings, Auto Zero On	3.5	3500	0.002	0.00003
2-wire ohms, <10M	Auto Zero On	6.5	5	10	0.16667
	Auto Zero On	6.5	35	1	0.01667
	Auto Zero Off	6.5	50	1	0.01667
	Auto Zero Off	5.5	150	0.1	0.00167
	1K readings, Auto Zero Off	5.5	500	0.1	0.00167
	1K readings, Auto Zero Off	4.5	3000	0.006	0.00010
	1K readings, Auto Zero On	3.5	3500	0.002	0.00003
4-wire ohms, <10M	Auto Zero On	6.5	1.4	10	0.16667
	Auto Zero On	6.5	15	1	0.01667
	1K readings, Auto Zero Off	5.5	33	0.1	0.00167

Table 2-1: Measurement Functions, Measurement Rate and Resolution

Setting the reading rate sets the integration time (measurement speed) of the A/D converter and the period of time the input signal is measured (also known as aperture). The integration time affects the amount of reading noise, as well as the ultimate reading rate of the instrument. Note that setting the resolution will set default aperture or PLC times.

The integration time is specified in parameters based on the number of power line cycles (NPLC), where 1 PLC for 60Hz is 16.67msec (1/60) and 1 PLC for 50Hz is 20msec (1/50). The GX2065 has a parabola-like shape for its speed vs. noise characteristics as shown in Figure 2-2 and is optimized for the 1 PLC to 5 PLC reading rate. At these rates (lowest noise region in graph), the DMM will make corrections for its own internal drift and still be fast enough to settle a step response <100ms.

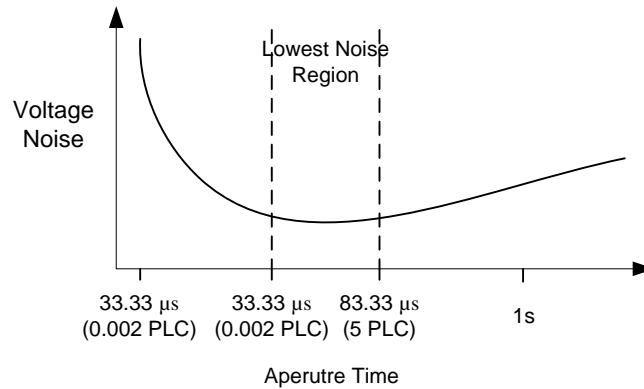


Figure 2-2: Noise Voltage vs PLC rate

Theory of Operation and Architecture

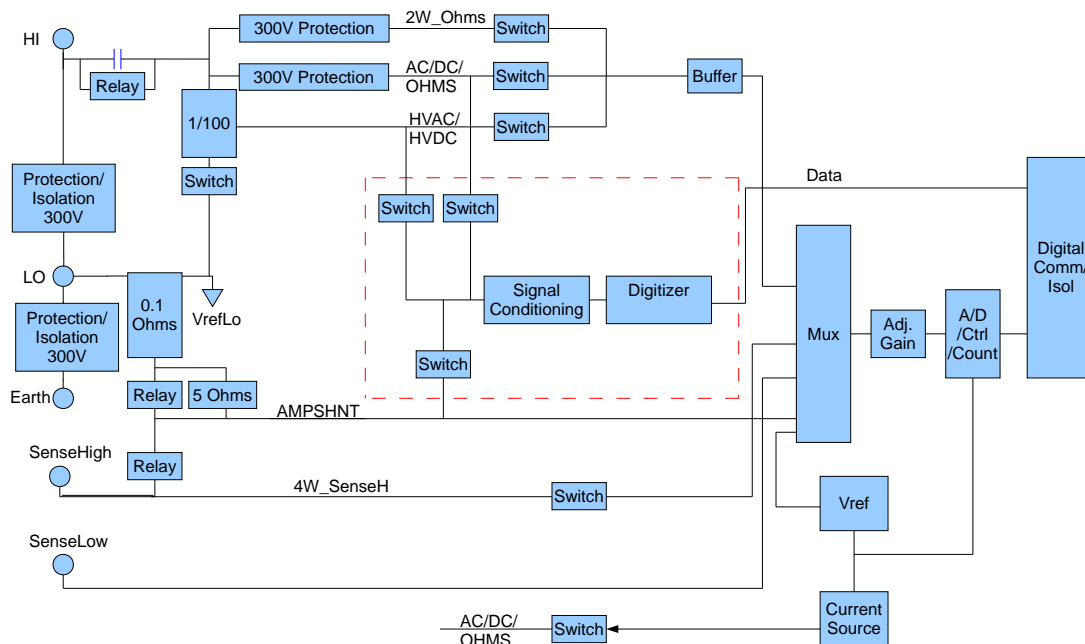


Figure 2-3: DMM Block Diagram

The GX2065 offers full-feature, 6.5 digit DMM capabilities in a compact, 3U PXI form factor. Additionally, the GX2065 includes a 16 bit, 3 MS/s digitizer which supports isolated, differential measurements for DC / AC coupled voltage and current measurements. Up to 8192 samples can be acquired by the digitizer. Figures 2-3 and 2-4 provide a block diagram of the GX2065's isolated DMM measurement and protection circuitry.

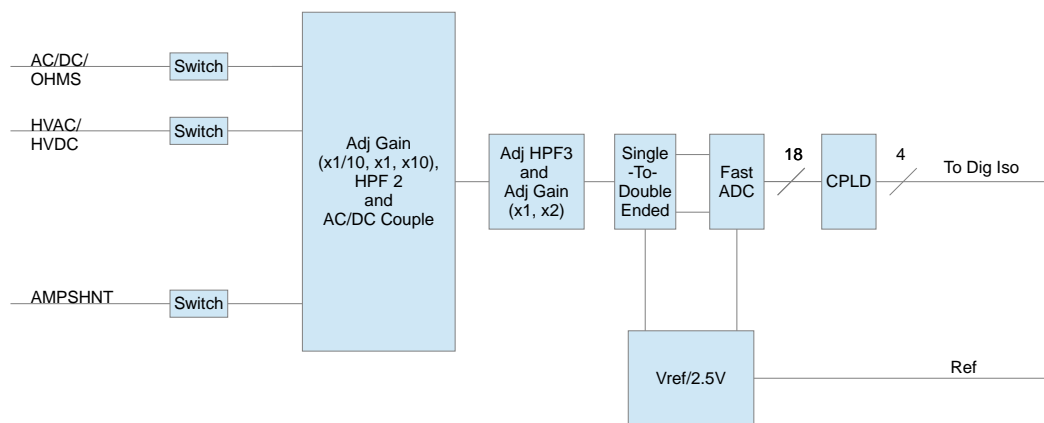


Figure 2-4: AC and Current Measurement Circuitry

All DMM measurement circuitry is galvanically isolated from the PXI bus. Protection of all four user accessible measurement terminals (High, Low, Sense High and Sense Low) is provided per the IEC and UL standards for Class II measurement equipment.

Voltage measurements (DCV and ACV)

The Hi and Lo terminals are used to make DC and AC voltage measurements. The GX2065 can make DCV measurements from 0.1 μ V to 300 V and ACV measurements from 0.1 μ V to 300V RMS, 425 V peak. The input circuitry presents the following input resistance values for DC and AC operation.

DCV input resistance:

100V and 300V ranges: 10M ohms

100mV, 1V, and 10V ranges: >10G ohms

ACV input impedance: 10M ohms

As shown in the block diagram, both the Hi and Lo input terminals are protected and isolated to 300V between the Hi and Lo terminals and the Lo and chassis ground potentials. Internal switching routes the Hi signal to the converter (DCV measurements) or to the AC circuitry for AC measurements. An internal reference provides a transfer standard for measurement traceability to an external standard. AC RMS conversion uses a high speed A/D in conjunction with on-board RMS processing algorithm.

Current measurements (DCI and ACI)

The Sense Hi terminal and the Lo terminal are used to make DCI measurements from 10nA to 2A and ACI measurements from 3 μ A to 2A peak. 5.1 and 0.1 ohm shunts are used for current measurements with the resulting voltage applied to the current measurement circuitry and converter.



Caution - The current measurement circuitry is protected by a 2 amp fuse. If the current exceeds 2 amps, the fuse will need to be replaced. Do not replace with a fuse with a higher rating than specified for the instrument.

Resistance measurements (2 and 4 wire)

The GX2065 has seven ohms ranges to measure resistance from 100μ ohms to 100M ohms. Available measurement ranges include 100 ohms, 1k ohms, 10k ohms, 100k ohm, 1M ohms, 10M ohms, and 100M ohms. The GX2065 can perform both 2 and 4 wire resistance measurements. Two wire measurements use the Hi and Lo terminals and four wire measurements use the Hi, Lo, Sense Hi, and Sense Lo terminals. For resistances >1k ohm, the 2-wire function is typically used for measurements. For resistances less than 1 K ohm, the 4-wire function is typically used which cancels the effects of test lead (and channel path) resistances.

For the 100 ohm to 1M ohm ranges, the GX2065 uses the constant-current method to measure resistance. A constant current is sourced to the unknown resistance and the voltage across the unknown resistance is measured. Resistance (R) is then calculated using the known current and measured voltage ($R = V/I$). For the 10M ohm and 100M ohm ranges, the ratiometric method is used to measure resistance which employs a test current (0.7μA source) in parallel with a 10MΩ reference resistor and the unknown resistance value. Knowing the value of the source current and the 10 M ohm reference resistor, a radiometric calculation can be made as determined by the following equation:

$$I_{\text{SOUR}} = (V_{\text{MEAS}} / R_{\text{REF}}) + (V_{\text{MEAS}} / R_{\text{DUT}})$$

Frequency and period measurements

The GX2065 can perform frequency measurements from 3Hz to 500 KHz on voltage ranges of 100mV, 1V, 10V, 100V, and 300V. Period (1 / frequency) measurements can also be performed from 2 μ s to 333ms on the same voltage ranges as the frequency.

Input impedance: 10M ohm, AC coupled.

The instrument uses the volts input to measure frequency. The signal voltage must be greater than 10% of the full-scale range.



Caution - The voltage input is subject to the 8×10^7 Volts – Hz product.

Frequency and period measurement use a zero-crossing trigger, meaning that a count is taken when the signal crosses the zero level. The instrument uses a reciprocal counting technique to measure frequency and period. This method generates constant measurement resolution for any input frequency. The Multimeter's AC voltage measurement section performs input signal conditioning.

Gate time for frequency measurement is programmable. The default gate time is 1 second. The GX2065 completes a reading when it receives its first zero-crossing after the gate time expires. In other words, the reading is completed 1/2 cycle after the gate time has expired. For example, to sample a 3Hz frequency, you may wait up to 1.33 seconds before a reading is returned.

Virtual Panel Description

The GXDMM includes a virtual panel program, which provides full access to the various configuration settings and operating 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 GXDMM Panel from the Marvin Test Solutions, GXDMM menu under the Start menu. The GXDMM virtual panel opens as shown here:

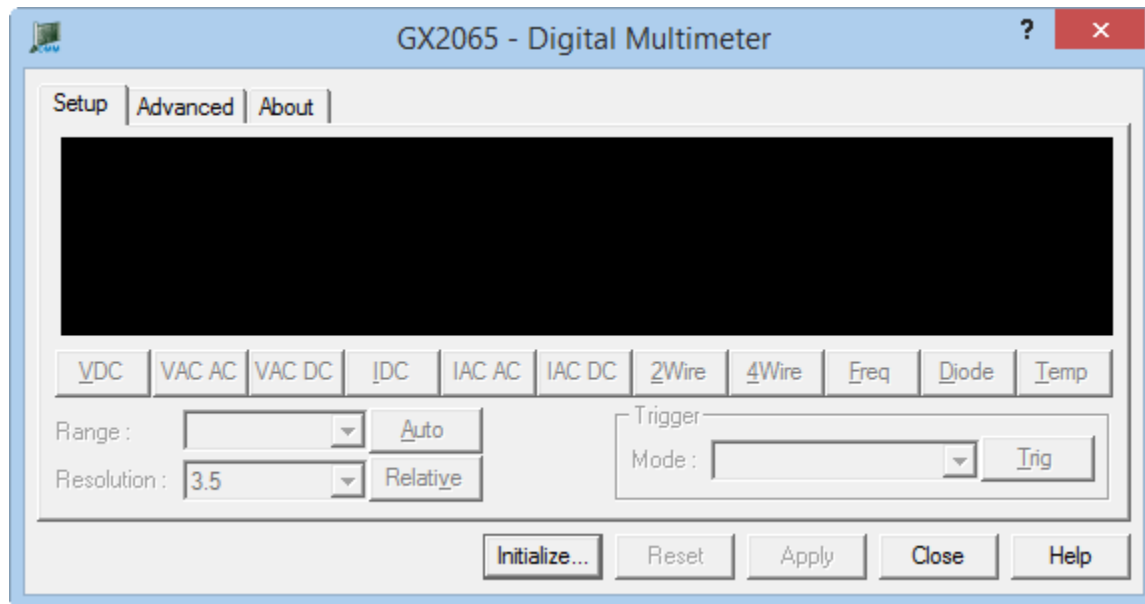


Figure 2-5: GXDMM Virtual Panel (not Initialized)

The function of the panel button controls are shown below:

Initialize Opens the Initialize Dialog (see Initialize Dialog paragraph) in order to initialize the board driver. The current settings of the selected DMM **will not change after calling initialize**. The panel will reflect the current settings of the DMM 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 DMM 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 display the What's This Help information click on the (?) button and then click on the control – a small window will display the information regarding this control.

Virtual Panel Initialize Dialog

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

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 DMM 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 0x10F (chassis 1 Slot 15). This is the slot number argument (*nSlot*) passed by the panel when calling the driver **GxDmmInitialize** function used to initialize driver with the specified board.

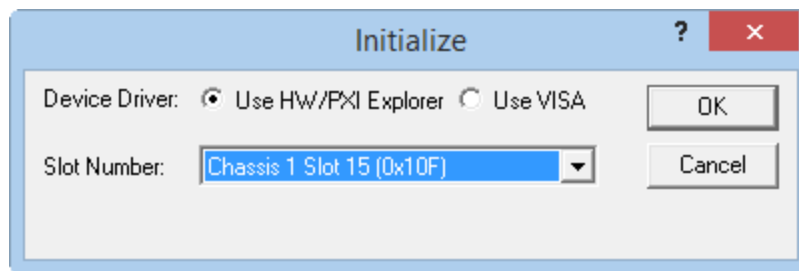


Figure 2-6: Initialize Dialog Box using Marvin Test Solutions' HW driver

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 in Marvin Test Solutions **PXI/PCI Explorer**. The following figure shows PXI8::14::INSTR as the VISA resource (PCI bus 8 and Device 14). This is VISA resource string argument (*szVisaResource*) passed by the panel when calling the driver **GxDmmInitializeVisa** function to initialize the driver with the specified board.

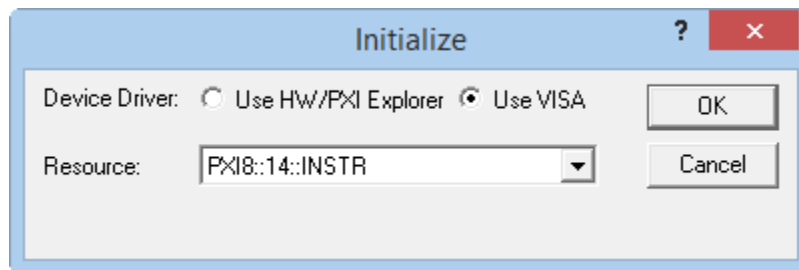


Figure 2-7: Initialize Dialog Box using VISA resources

Virtual Panel Setup Page

After the board is initialized the panel is enabled and will display the current setting of the board. The panel caption will show the board address (0x10F in this example). The following figure shows the **Setup** page settings:

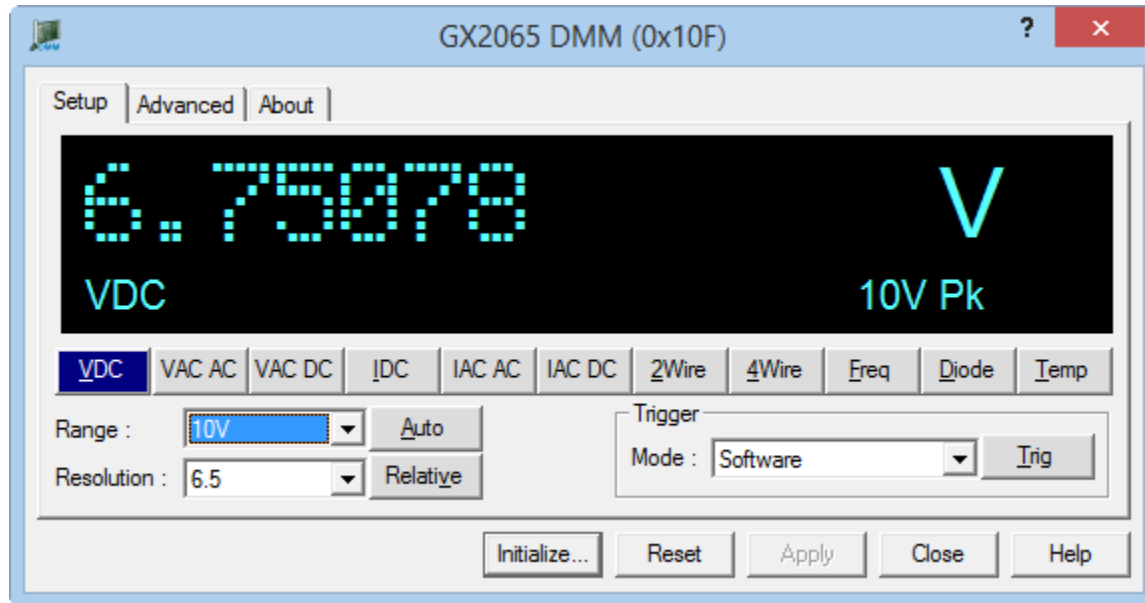


Figure 2-8: GXDMM Virtual Panel (Initialized)

The following controls are shown in the Setup page:

Measurement Display:

- Measurement value
- Units of Measurement
- Function Selected
- Auto Range State
- Relative Measurement State
- Range Selected

Function/Range Settings:

VDC Button: Sets the Measurement function to Volts DC

VAC AC Button: Sets the Measurement function to Volts AC, AC Coupled

VAC DC Button: Sets the Measurement function to Volts AC, DC Coupled

IDC Button: Sets the Measurement function to Current IDC

IAC AC Button: Sets the Measurement function to Current IAC, AC Coupled

IAC DC Button: Sets the Measurement function to Current IAC, DC Coupled

2Wire Button: Sets the Measurement function to 2 Wire Resistance

4Wire Button: Sets the Measurement function to 4 Wire Resistance

Freq Button: Sets the Measurement function to Frequency

Temp Button: Sets the Measurement function to Internal Temperature

Diode Button: Sets the Measurement function to Diode Volts

Range (dropdown list): Selects the range to set for the select function

Resolution (dropdown list): Selects the resolution (number of digits) of the measurement (3.5, 4.5, 5.5, and 6.5 digits)

Auto Button: Toggles the Auto Range state

Relative Button: Toggles the Relative Measurement state. When this button is clicked (to activate), the last measurement is stored in computer memory. Every subsequent measurement will then be displayed in relation to the stored measurement.

Trigger Settings:

Force Button: Forces a software trigger to initiate a measurement in Software Trigger mode

Trigger Mode (dropdown list): Sets/displays the Trigger mode. Selections specify how the DMM will be triggered to take measurements

- **Continuous:** Instrument continuously measures.
- **Timer:** Instrument takes a measurement every time an internal timer expires
- **External:** Instrument will trigger a reading based on an external trigger source
- **Software:** Instrument will trigger on a software command (Force Button)

Virtual Panel Advanced Page

Clicking on the **Advanced** tab will show the **Advanced** page as shown below:

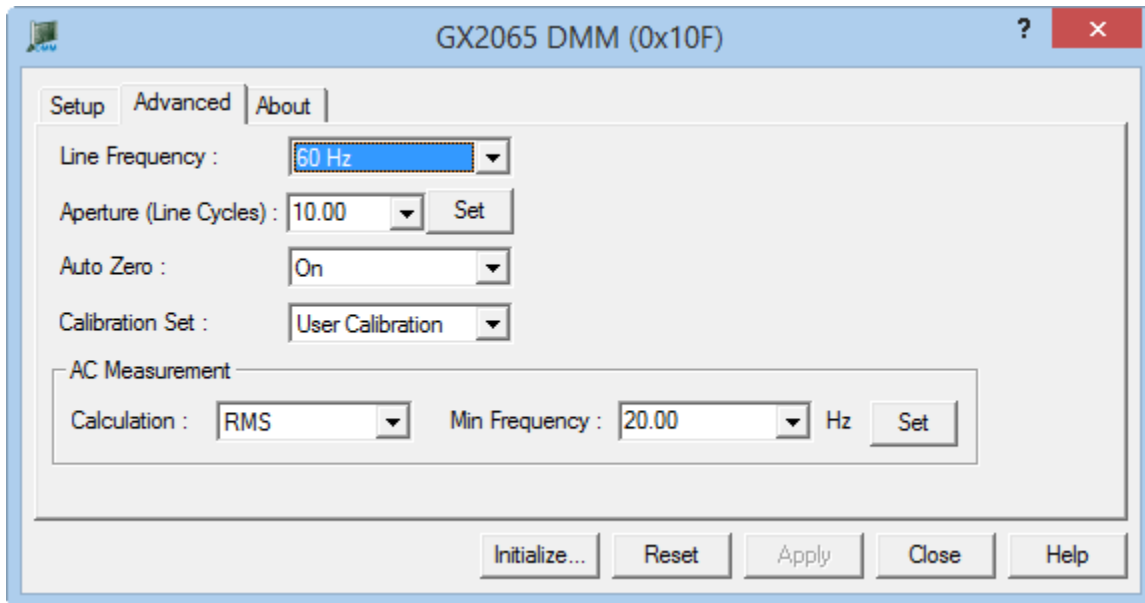


Figure 2-9: GXDMM Virtual Panel – Advanced Page

AC Measurement:

Calculation (dropdown list): Sets/displays the math type applied to the AC measurement:

1. **RMS:** Root mean average calculation
2. **Average:** Mean average calculation
3. **Peak to Peak:** Peak to Peak measurement
4. **Peak:** Peak measurement

Minimum Frequency (dropdown list): Enter the minimum frequency of the signal to be measured in AC mode. The dropdown list provides some possible settings. The user can enter a floating point value directly into the control instead of selecting from one of the choices.

Miscellanies Settings

Line Frequency (dropdown list): Sets/displays the AC line frequency (60Hz, 50Hz, and 400Hz)

Aperture (dropdown list): Enter the aperture duration in terms of line cycles (as defined in the Line Frequency settings) for DC and Resistance measurements. The dropdown list provides some possible settings. The user can enter a floating point value directly into the control instead of selecting from one of the choices.

Calibration Set (dropdown list): Sets/Displays the currently active calibration set (User or Factory sets)

Auto Zero (dropdown list): Sets/Displays the currently selected auto zero mode:

1. **Off:** Auto Zeroing is off
2. **On:** Auto Zeroing is on
3. **Now:** Auto Zeroing will occur immediately and then revert to the previous set auto zero mode (Off or On)

Virtual Panel About Page

Clicking on the **About** tab will show the **About** page as shown below:

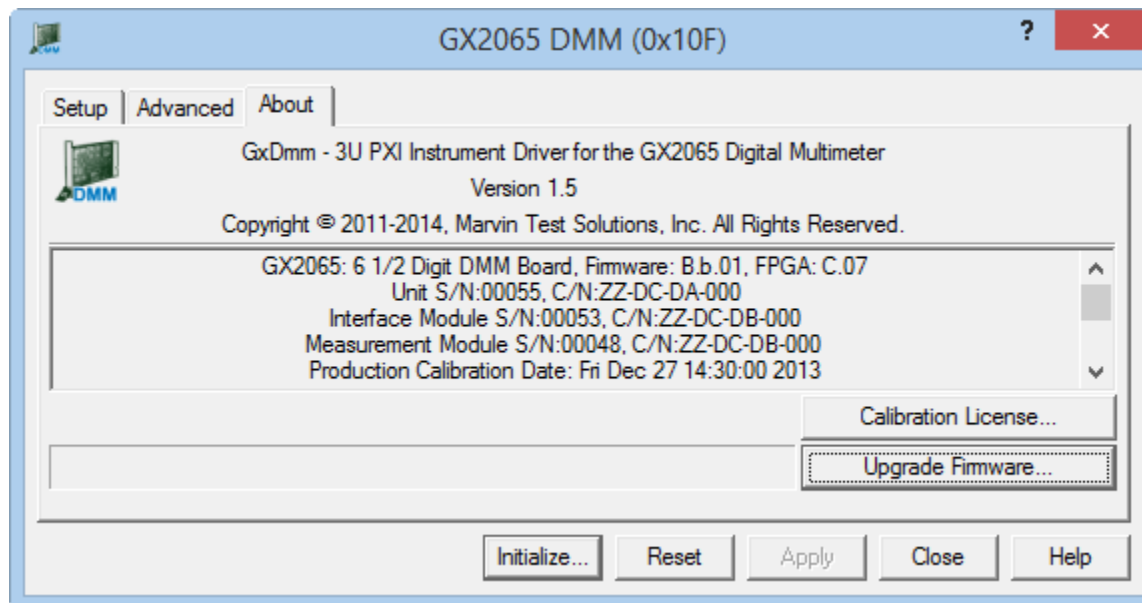


Figure 2-10: GXDMM Virtual Panel – About Page

The following controls are shown in the **About** page:

The top part of the **About** page displays version and copyright of the GXDMM driver. The bottom part displays the board summary. The board summary lists the Boards Type, e.g. GX2065, Firmware Version, FPGA Version, Unit Serial Number, Unit Control Number, Interface Module Serial Number, Interface Module Control Number, Measurement Module Serial Number, and Calibration Information.

Calibration License... button opens the **License Setup** dialog, see “Chapter 5: Calibration” for details.

The **About** page also contains a button **Upgrade Firmware...** used to upgrade the board Firmware and FPGA. This button maybe used only when the board requires upgrade as directed by Marvin Test Solutions support. The upgrade requires a firmware file (.flash) 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 3 - Installation and Connections

Getting Started

This section includes general hardware installation procedures for the GX2065 board and installation instructions for the GX2065 (GXDMM) 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
GXDMM Driver Installation	This Chapter
Programming	Chapter 3
Calibration	Chapter 4
GXDMM Function Reference	Chapter 5

Interfaces and Accessories

All GX2065 boards have the same basic packing list, which includes:

1. GX2065 Board
2. GXDMM 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 GX2065 Instrument board is designed to run on PXI compatible computer running Windows 9x, Windows Me, Windows NT, Windows 2000, XP, Vista and above. In addition, Microsoft Windows Explorer version 4.0 or above is required to view the online help.

The board requires one unoccupied 3U PXI bus slot.

DMM Connections

The following figure details the DMM connections:

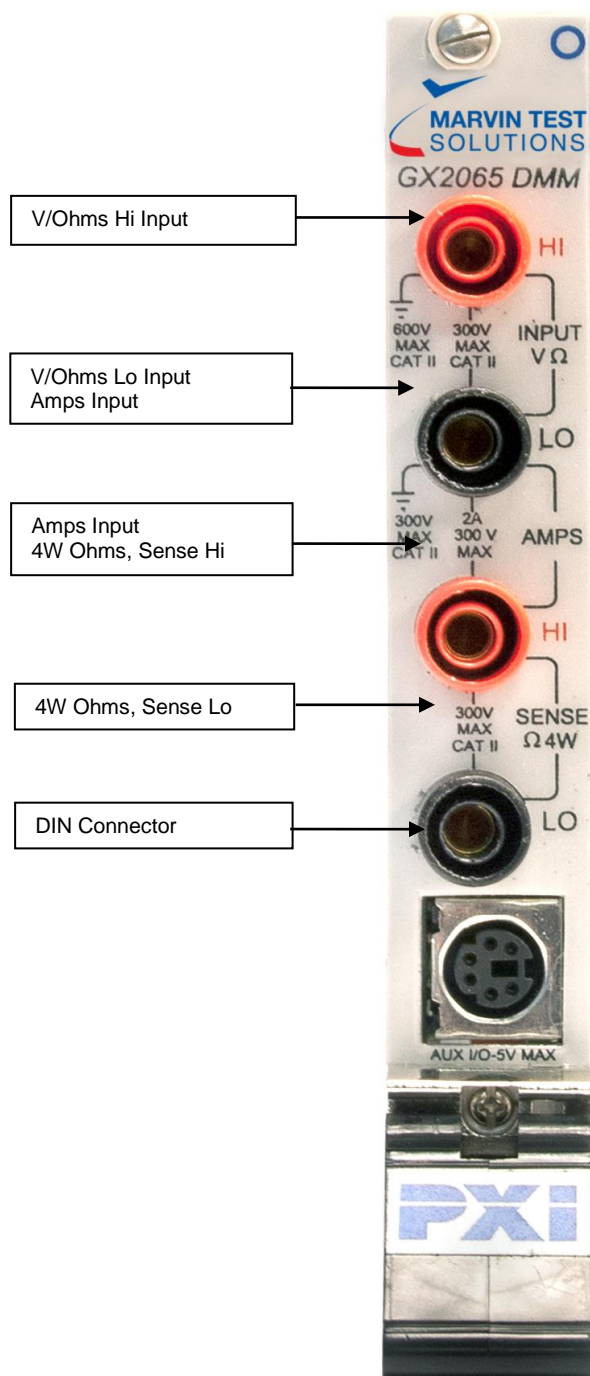


Figure 2-4: GX2065 Front Panel Connections

Connections for the 6-pin DIN connector are listed below:

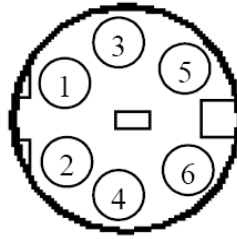


Figure 2-4: DIN Connector Pin Out, Front panel view of DMM

Pin 1: Trigger In

Pin 2 & Pin 4: Ground

Pin 3: Trigger Out

Note: These connections are referenced to PXI ground and are isolated from the DMM's measurement terminals.

Installation of the GXDMM Software

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

1. Insert the Marvin Test Solutions CD-ROM and locate the **GXDMM.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).
2. Run the GXDMM setup and follow the instruction on the Setup screen to install the GXDMM driver.

Note: When installing under Windows NT/2000/XP/VISTA, you may be required to restart the setup after logging-in as a user with Administrator privileges. This is required in-order to upgrade your system with newer Windows components and to install kernel-mode device drivers (HW.SYS and HWDEVICE.SYS) which are required by the GXDMM driver to access resources on your board.

3. The first setup screen to appear is the Welcome screen. Click **Next** to continue.
4. Enter the folder where GXDMM 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\GxDmm.
5. Select the type of Setup you wish and click **Next**. You can choose between **Typical**, **Run-Time** and **Custom** setups types. The **Typical** setup type installs all files. **Run-Time** setup type will install only the files required for controlling the board either from its driver or from its virtual panel. The **Custom** setup type lets you select from the available components.

The program will now start its installation. During the installation, Setup may upgrade some of the Windows shared components and files. The Setup may ask you to reboot after completion if some of the components it replaced were used by another application during the installation – do so before attempting to use the software.

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

Setup Maintenance Program

You can run the Setup again after GXDMM has been installed from the original disk or from the Windows Control Panel – Add Remove Programs applet. Setup will be in the Maintenance mode when running for the second time. The Maintenance window show below allows you to modify the current GXDMM installation. The following options are available in Maintenance mode:

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

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

Overview of the GXDMM Software

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

- **GXDMM Panel** – Configures and controls the GXDMM board various features via an interactive user interface.
- **GXDMM driver** - A DLL based function library (GXDMM.DLL, 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 GXDMM 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, GXDMM driver and panel.
- **HW driver and PXI/PCI Explorer applet** – HW driver allows the GXDMM 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 GXDMM driver can work with either HW or VISA to control an access the supported boards.

Installation Folders

The GX2065 driver files are installed in the default folder C:\Program Files\Marvin Test Solutions\GXDMM. You can change the default GXDMM folder to one of your choosing at the time of installation.

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

Name	Purpose / Contents
...\Marvin Test Solutions\GXDMM	The GXDMM 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. GXDMM Driver and example are copied to this directory only if ATEasy is installed to your machine.
...\Windows\System (Windows 9x/Me), or ...\\Windows\System32 when running Windows NT/2000/XP/VISTA/7	Windows System directory. Contains the GXDMM 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. "DMM1"). For more information regarding slot numbers and alias names, see the **GxDmmInitialize** and **GxDmmInitializeVisa** functions.
4. **Save your work.** PXI Explorer saves the configuration to the following files located in the Windows folder: PXISYS.ini, PXISYS.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).

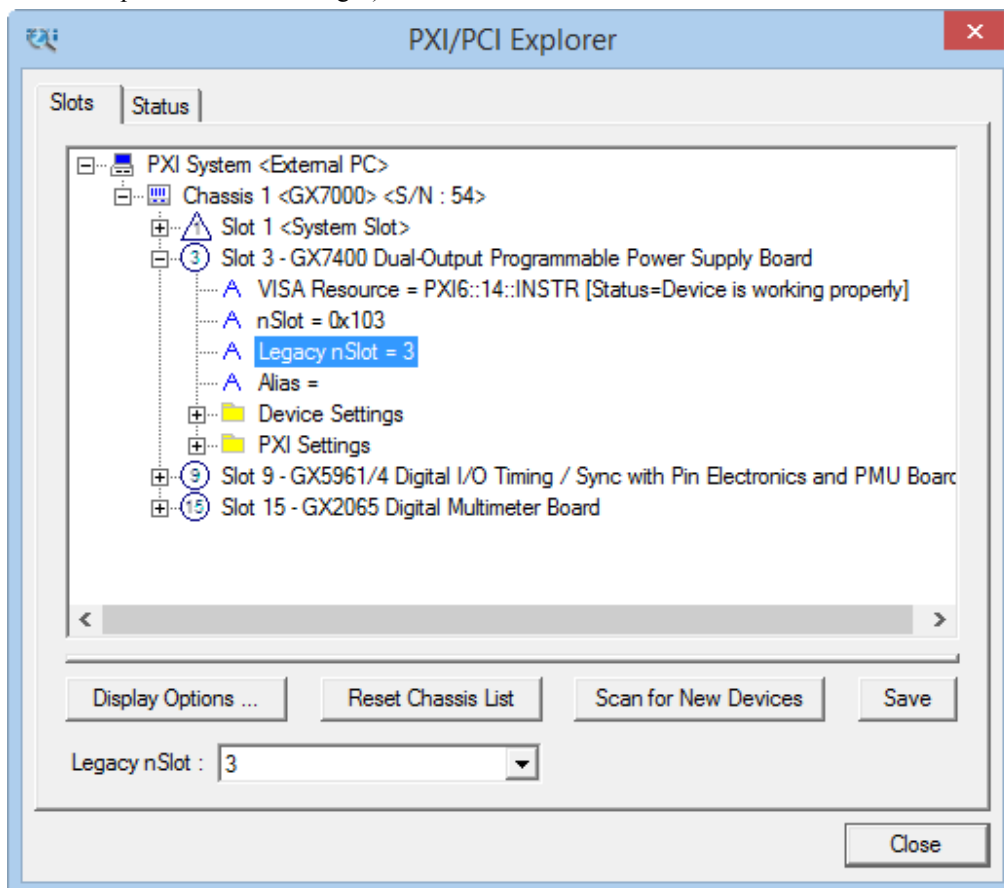


Figure 3-1: PXI/PCI Explorer

Board Installation

Before you Begin

- Install the GXDMM 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 GX2065 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 GXDMM Driver as described in the next section.
2. Turn off the PXI chassis and unplug the power cord.
3. Locate a PXI empty slot on the PXI chassis.
4. Place the module edges into the PXI chassis rails (top and bottom).
5. Carefully slide the PXI board to the rear of the chassis, make sure that the ejector handles are pushed **out** (as shown in Figure 3-2).

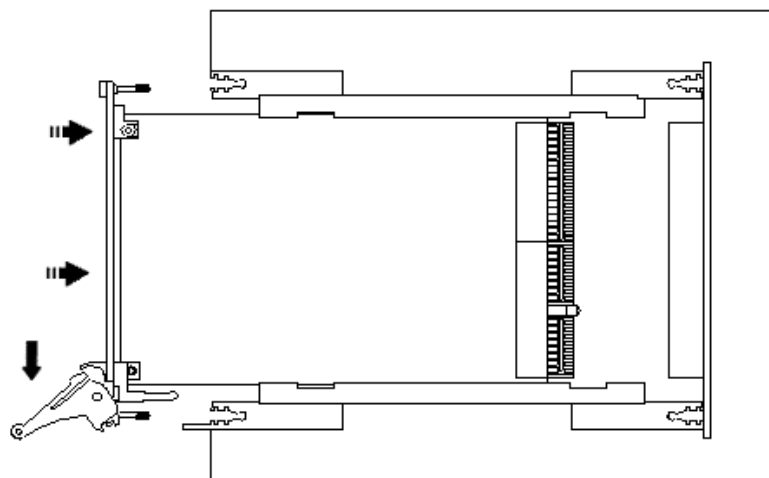


Figure 3-2: Ejector handles position during module insertion

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

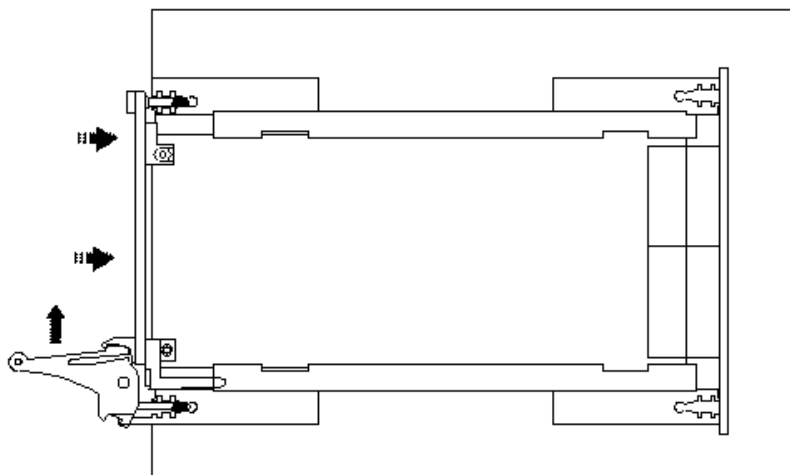


Figure 3-3: Ejector handles position after module insertion

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

Plug & Play Driver Installation

Plug & Play operating systems such as Windows 9x, Me, Windows 2000, XP or VISTA (Not Windows NT) 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 Program Files\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 GXDMM driver was not installed prior to the board installation), you may install the GXDMM driver as described in the prior section, then click on the **Have Disk** button and browse to select the HW.INF file located in C:\Program File\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 GXDMM 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 GXDMM driver.

Chapter 4 - Programming the Board

This chapter contains information about how to program the GX2065 board using the GXDMM driver.

The GXDMM driver contains functions to initialize, reset, and control the board. A brief description of the functions, as well as how and when to use them, is included in this chapter.

The GXDMM driver supports many development tools. Using these tools with the driver is described in this chapter. In addition, the GXDMM directory contains examples written for these development tools.

The GXDMM Driver

The GXDMM driver is provided with support for 32 bit Windows dynamic link library (**GxDmm.dll**) and 64 bit Windows (**GxDmm64.dll**). Additional drivers are provided for other operating systems such as Linux and LabView Real-Time; see the readme file for more information regarding these drivers. The 32 bit DLL is used with 32 bit applications running under Windows 2000/XP/VISTA/7 and the 64 runs on Windows XP/Vista/6 64 bit editions. The DLL uses device driver (HW provided by Marvin Test Solutions or VISA provided by a third party vendor) to access the board resources. The device driver HW includes HW.SYS and HW64.SYS is installed by the GXDMM setup program and is shared by other Marvin Test Solutions products (ATEasy, GTDIO, etc.).

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

Programming Using C/C++ Tools

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

- Include the **GxDmm.h** header file in the C/C++ source file that uses the GXDMM 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 **GxDmm.lib** and **GxDmm64.lib** (for 64 bit applications) for Microsoft Visual C++ and **GxDmmBc.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 GXDMM as required by the application.
- Build the project.
- Run, test, and debug the application.

Programming Using Visual Basic and Visual Basic .NET

To use the driver with Visual Basic 4.0 or above (for 32-bit applications), the user must include the **GxDmm.bas** to the project. The file can be loaded using *Add File* from the Visual Basic *File menu*. The **GxDmm.bas** contains function declarations for the DLL driver. If you are using Visual Basic .NET – use the **GxDmm.vb**. Examples for both programming languages reside in the product Examples folder.

Programming Using Pascal/Delphi

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

Programming GXDMM Boards Using ATEasy®

The GXDMM package is supplied with the **GX2065.drv** ATEasy driver. The ATEasy driver uses the **GxDmm.dll** to program the board. The driver is provided with an ATEasy example that contains a program and a system file pre-configured with the ATEasy driver. Use the driver shortcut property page from the System Drivers sub-module to change the PXI HW slot number or VISA resource string before attempting to run the example.

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

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

- The *nHandle* parameter is omitted. The driver handles this parameter automatically. ATEasy uses driver logical names instead i.e. DMM1 for GX2065.
- 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.

Programming Using LabView and LabView/Real Time

To use the driver with LabView use the provided lab view library **GxDmm.llb**. The library is located in the GXDMM folder. An example for LabView is also provided in the Examples folder. A DLL located in the LabViewRT folder can be used for deployment with LabView/Real-Time.

Using and Programming under Linux

Marvin Test Solutions provides a separate software package named **GtLinux** with Linux driver (Marvin Test Solutions Drivers Pack for Linux). The software package can be downloaded from the Marvin Test Solutions website. See the ReadMe.txt in that package for more information regarding using and programming the driver under Linux.

Using the Signametrics SM204x and SM2060 Compatibility Layer

Marvin Test Solutions provides a dynamic link library that provides a compatibility layer for users that choose to replace their Signametrics SM204x/SM2060 boards with the GX2065. Signametrics SM204x/SM2060 boards use the **sm204032.dll** or **SM2060.dll** in conjunction with the Marvin Test Solutions **sm204032.dll**. The Marvin Test Solutions **sm204032.dll** and **SM2060.dll** are installed in the GXDMM folder while the Signametrics **sm204032.dll/SM2060.dll** is installed in the Windows System32 folder on 32-bit Windows, or Windows SysWOW64 folder on 64-bit Windows. You can rename the Signametrics DLL in that folder before copying the Marvin Test Solutions DLL one. Appendix B details the supported SM204x and SM2060 functions.

In order for the Gx2065 to be fully compatible with the SM204x family the user needs to modify the calibration file provided by Signametrics as follows:

1. Locate the calibration file provided by Signametrics, the calibration file name has the following format "Sm40cal.dat".
2. Open the calibration file with any text editor (e.g. notepad).
3. Append to the end of the file "Power Line Cycles=" followed by the power line cycles value, e.g. if in the USA add the following line: "Power Line Cycles=50".
4. Save and close the file.

Using the GXDMM driver functions

The following paragraphs describe the steps required to program the boards.

Initialization, HW Slot Numbers and VISA Resource

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

1. **Marvin Test Solutions' HW** – This is the default device driver that is installed by the GXDMM driver. To initialize and control the board using the HW use the **GxDmmInitialize(*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: 3 in this example.

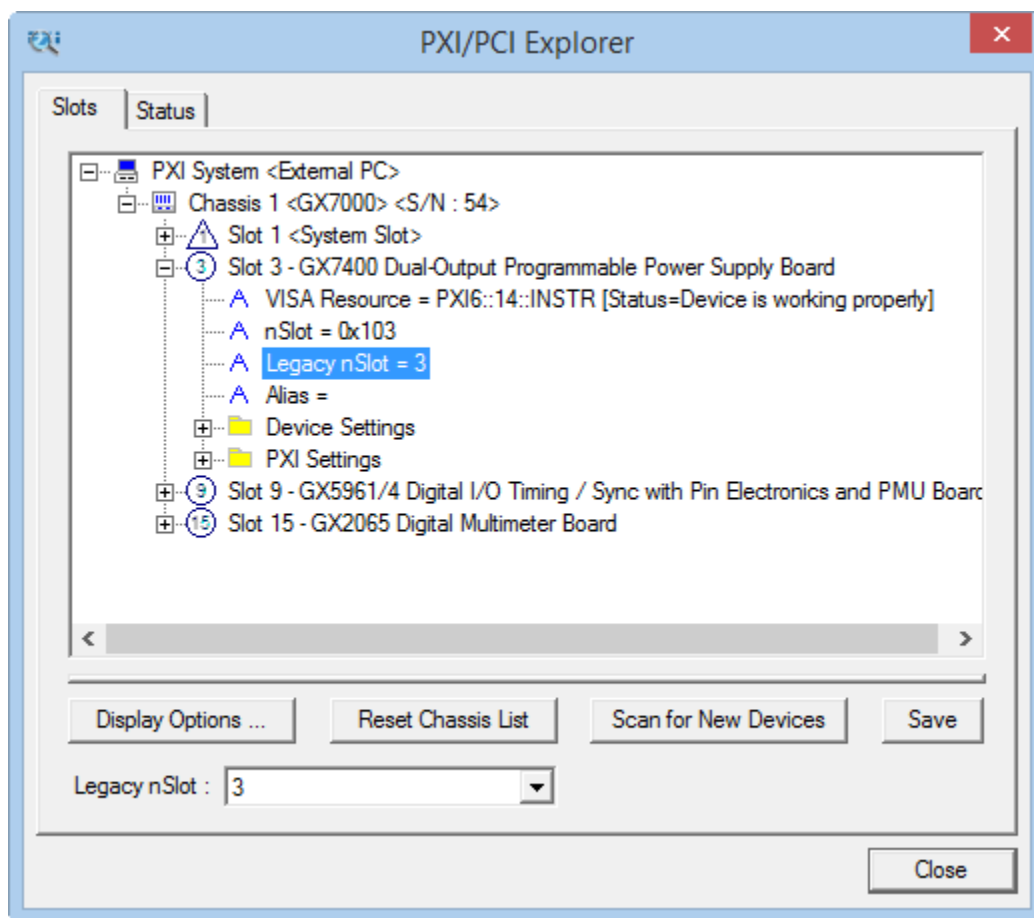


Figure 4-1: PXI/PCI Explorer

2. **VISA** – This is a third party library usually supplied by National Instruments (NI-VISA). You must ensure that the VISA installed supports PXI and PCI devices (not all VISA providers supports

PXI/PCI). GXDMM setup installs a VISA compatible driver for the GXDMM board in-order to be recognized by the VISA provider. Use the GXDMM function **GxDmmInitializeVisa** (*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 **GxDmmInitialize** and the **GxDmmInitializeVisa** 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.

The default state after reset is as follows:

- Function: VDC
- Volts Range: 10V
- Current Range: 20mA
- Resistance Range: 10KOhm
- Resolution: 6 ½ Digits
- Aperture: 10 Power Line Cycles
- Auto Range: Off
- Trigger Mode: Continuous
- AC Math: RMS
- AC Min Frequency: 10 Hz
- AC Input Line Frequency: Not affected
- User Calibration

Error Handling

All the GXDMM 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 **GxDmmGetErrorString** function to return a string representing the error. The **GxDmmGetErrorString** reference contains possible error numbers and their associated error strings.

Driver Version

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

Programming Examples

The README.txt located on the GXDMM folder contains a list of the GXDMM programming examples provided with the GXDMM 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 (GxDmm.dll, GXDMM64.dll and the HW device driver files located in the HW folder) can be shipped with the application. Typically, the GxDmm.dll should be copied to the Windows System directory. The HW device driver files should be installed using a special setup program HWSETUP.EXE that is provided with GXDMM driver files. Alternatively, you can provide the GXDMM disk to be installed along with the board.

Sample Programs

The following example demonstrates how to program the board using the C/C++ programming language under Windows. The example shows how to initialize the DMM, set it up for measurement or trigger settings and get the reading.

To run, enter the following command line:

GxDmmExampleC <Slot> <Function> <Range> <Resolution>

Where:

<Slot>	PXI Explorer slot number where the board reside.
<Function>	DMM measurement function: VDC=set function to Volts DC IDC=set function to Current DC VAC=set function to Volts AC, AC Coupled IAC=set function to Current AC AC Coupled VAC_DC_CPL=set function to Volts AC, DC Coupled IAC_DC_CPL=set function to Current AC DC Coupled TWO_WIRE=set function to 2 Wire Resistance FOUR_WIRE=set function to 4 Wire Resistance
<Range>	Measurement range: A floating point value approximating the value to be measured.
<Resolution>	One of the following resolutions: 4=set resolution to 3.5 digits 5=set resolution to 4.5 digits 6=set resolution to 5.5 digits 7=set resolution to 6.5 digits

Sample Program Listing

```

/*****

FILE           : GxDmmExampleC.cpp

PURPOSE        : WIN32/LINUX example program for GX2200 boards
                  using the GXDMM driver.

CREATED        : Mar 2011

COPYRIGHT      : Copyright 2011-2013,  MARVIN TEST SOLUTIONS, Inc.

COMMENTS      :

To compile the example:

1. Microsoft VC++
   Load GxDmmExampleC.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 GxDmmExampleC.bpr from the Project/Open
   Project... menu
   Select Project/Build all from the menu

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

*****/
#ifdef __GNUC__
#include "windows.h"
#endif
#include "GxDmm.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#ifdef __BORLANDC__
#pragma hdrstop
#include <condefs.h>
USELIB ("GxDmmBC.lib");
USERC ("GxDmmExampleC.rc");
#endif

//*****
//          DisplayMsg
//*****
void DisplayMsg(PSTR lpszMsg)
{
#ifdef __GNUC__
    MessageBeep(0);

```

```

        MessageBox(0, lpzMsg, "GxDmm example program", MB_OK);
#else
    printf("\r\nGxDmm example program: %s\r\n", lpzMsg);
#endif
    return;
}

//*****
//      __strupr
//*****
char * __strupr(char * sz)
{
    int i;

    for (i=0; sz[i]; i++)
        sz[i] = toupper(sz[i]);
    return sz;
}

//*****
//      DisplayUsage
//*****
void DisplayUsage(void)
{
    DisplayMsg(
        "This example shows how to setup and measure using the "
        "GX2065:"

        "\r\n\r\nUsage:\r\n"
        "GxDmmExampleC <slot> <function> <range> <resolution>"

        "\r\n\r\nWhere :\r\n"
        "<slot> : PCI/PXI slot number as shown by the PXI "
        "explorer\r\n"

        "<function> :\r\n"
        "\tVDC=set function to Volts DC\r\n"
        "\tIDC=set function to Current DC\r\n"
        "\tVAC=set function to Volts AC, AC Coupled\r\n"
        "\tIAC=set function to Current AC AC Coupled\r\n"
        "\tVAC_DC_CPL=set function to Volts AC, DC Coupled\r\n"
        "\tIAC_DC_CPL=set function to Current AC DC Coupled\r\n"
        "\tTWO_WIRE=set function to 2 Wire Resistance\r\n"
        "\tFOUR_WIRE=set function to 4 Wire Resistance\r\n"
        "\tSUM=displays board summary\r\n"

        "<range> : floating point value approximating the value to "
        "be measured\r\n"

        "<resolution> :\r\n"
        "\t4=set resolution to 3.5 digits\r\n"
        "\t5=set resolution to 4.5 digits\r\n"
        "\t6=set resolution to 5.5 digits\r\n"
        "\t7=set resolution to 6.5 digits\r\n"

        "\r\nThe example should be run from the Windows command "
        "prompt/Linux terminal.\r\n"
    );
}

```

```

        "Examples :\r\n"
        "\t./GxDmmExampleC 7 VDC 100 7 1\r\n"
        "\t./GxDmmExampleC 7 SUM\r\n"
        "\t./GxDmmExampleC 7 TWO_WIRE 10000 5 "
        "(under Windows, slot 7)\r\n"
        "\t./GxDmmExampleC 0x60c TWO_WIRE 10000 5 "
        "(under Linux, bus 6 device 12)\r\n"
    );
    getchar();
    exit(1);
}

//*****
//          CheckStatus
//*****
void CheckStatus(SHORT nStatus)
{
    CHAR    sz[512];

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

//*****
//          MAIN
//
// This main function receives the following parameters
//
// GX2065 GXDMM board slot number (e.g. 1)
// <function>:
//          VDC=set function to Volts DC
//          IDC=set function to Current DC
//          VAC=set function to Volts AC, AC Coupled
//          IAC=set function to Current AC AC Coupled
//          VAC_DC_CPL=set function to Volts AC, DC Coupled
//          IAC_DC_CPL=set function to Current AC DC Coupled
//          TWO_WIRE=set function to 2 Wire Resistance
//          FOUR_WIRE=set function to 4 Wire Resistance
//          SUM=display board summary
// <range>: floating point value approximating the value to be
//          measured
// <resolution>: integer value specifying the rounded up number of
//          digits of resolution
//          4=set resolution to 3.5 digits
//          5=set resolution to 4.5 digits
//          6=set resolution to 5.5 digits
//          7=set resolution to 6.5 digits
// <count>: integer value specifying the number of readings to take
//          for the buffered measurement (1-8192)
//*****

```

```

int main(int argc, char **argv)
{
    SHORT nSlotNum=0, nHandle, nStatus;
    DOUBLE      dRange, dMeasurement, adMeasurements[8192];
    CHAR  * pszFunction;
    CHAR  szMeasurement[256], szUnits[10], szSummary[1024];
    DWORD dwResolution, dwCount, dwArrayCount;
    INT    i;
    DDWORD ddwTime, addwTime[8192];
    BOOL  bArrayReadFinished;

    // Check minimum number of arguments recived
    if (argc<2) DisplayUsage();

    if (nSlotNum<0)
        DisplayUsage();

    // Parse command line parameters
    nSlotNum=(SHORT)strtol(*(++argv), NULL, 0);
    if (nSlotNum<0) DisplayUsage();

    // Initialize DMM
    GxDmmInitialize(nSlotNum, &nHandle, &nStatus);
    CheckStatus(nStatus);

    // get function
    pszFunction=__strupr(*(++argv));

    if (!strcmp(pszFunction, "SUM"))
    {
        // print board summary
        GxDmmGetBoardSummary(nHandle, szSummary, sizeof szSummary,
            &nStatus);
        CheckStatus(nStatus);
        printf("Board Summary: %s.\n", szSummary);
    }
    else
    {
        // get/set function, range, resolution, count and measure
        if (argc<5)
            DisplayUsage();
        dRange=(DOUBLE)strtod(*(++argv), NULL);
        dwResolution=(DWORD)strtol(*(++argv), NULL, 0);
        dwCount=(DWORD)strtol(*(++argv), NULL, 0);

        if (dwResolution<4 || dwResolution>7)
            DisplayUsage();

        if (dwCount<1 || dwCount>8192)
            DisplayUsage();

        GxDmmSetTriggerMode(nHandle, GXDMM_TRIGGER_SOFTWARE, 0,
            &nStatus);
        CheckStatus(nStatus);

        // Set Function
        if (!strcmp(pszFunction, "VDC"))
        {
            GxDmmSetFunction(nHandle, GXDMM_FUNCTION_VDC,

```

```

        &nStatus);
        CheckStatus(nStatus);
        printf("Function set to VDC\n");
    }
    else if (!strcmp(pszFunction, "IDC"))
    {
        GxDmmSetFunction(nHandle, GXDMM_FUNCTION_IDC,
            &nStatus);
        CheckStatus(nStatus);
        printf("Function set to IDC\n");
    }
    else if (!strcmp(pszFunction, "VAC"))
    {
        GxDmmSetFunction(nHandle, GXDMM_FUNCTION_VAC_AC_CPL,
            &nStatus);
        CheckStatus(nStatus);
        printf("Function set to VAC, AC Coupled\n");
    }
    else if (!strcmp(pszFunction, "IAC"))
    {
        GxDmmSetFunction(nHandle, GXDMM_FUNCTION_IAC_AC_CPL,
            &nStatus);
        CheckStatus(nStatus);
        printf("Function set to IAC, AC Coupled\n");
    }
    else if (!strcmp(pszFunction, "VAC_DC_CPL"))
    {
        GxDmmSetFunction(nHandle, GXDMM_FUNCTION_VAC_DC_CPL,
            &nStatus);
        CheckStatus(nStatus);
        printf("Function set to VAC, DC Coupled\n");
    }
    else if (!strcmp(pszFunction, "IAC_DC_CPL"))
    {
        GxDmmSetFunction(nHandle, GXDMM_FUNCTION_IAC_DC_CPL,
            &nStatus);
        CheckStatus(nStatus);
        printf("Function set to IAC, DC Coupled\n");
    }
    else if (!strcmp(pszFunction, "TWO_WIRE"))
    {
        GxDmmSetFunction(nHandle, GXDMM_FUNCTION_2WIRE_OHM,
            &nStatus);
        CheckStatus(nStatus);
        printf("Function set to 2 Wire Resistance\n");
    }
    else if (!strcmp(pszFunction, "FOUR_WIRE"))
    {
        GxDmmSetFunction(nHandle, GXDMM_FUNCTION_4WIRE_OHM,
            &nStatus);
        CheckStatus(nStatus);
        printf("Function set to 4 Wire Resistance\n");
    }
    else
        DisplayUsage();

    // Set Range

```

```

GxDmmSetRange(nHandle, dRange, &nStatus);
CheckStatus(nStatus);

// Set Digits Resolution
GxDmmSetResolution(nHandle, dwResolution, &nStatus);
CheckStatus(nStatus);
switch (dwResolution)
{
    case GXDMM_RESOLUTION_3_5_DIGITS:
        printf("Resolution set to 3 1/2 Digits\n");
        break;
    case GXDMM_RESOLUTION_4_5_DIGITS:
        printf("Resolution set to 4 1/2 Digits\n");
        break;
    case GXDMM_RESOLUTION_5_5_DIGITS:
        printf("Resolution set to 5 1/2 Digits\n");
        break;
    case GXDMM_RESOLUTION_6_5_DIGITS:
        printf("Resolution set to 6 1/2 Digits\n");
        break;
    default:
        //Should Never Get Here
        printf("Error setting resolution\n");
        break;
}

// Take Measurement with formatted string, units of
// measurement and time stamp in nano seconds returned
GxDmmMeasureEx(nHandle, &dMeasurement, szMeasurement,
    szUnits, &ddwTime, &nStatus);
CheckStatus(nStatus);

// Print measured value and units of measurement
printf("Single Meseasured Value: %s %s\n", szMeasurement,
    szUnits);

// Initiage buffered measurement
GxDmmSetTriggerMode(nHandle, GXDMM_TRIGGER_ARRAY, dwCount,
    &nStatus);
CheckStatus(nStatus);

printf("Array Measurement in Progress...\n");

// Check for completion of array measurement
bArrayReadFinished=0;
while (!bArrayReadFinished)
    GxDmmGetReadArrayStatus(nHandle, &bArrayReadFinished,
        &dwArrayCount, &nStatus);

// Read back measurement buffer
GxDmmReadArray(nHandle, adMeasurements, addwTime, dwCount,
    &nStatus);
CheckStatus(nStatus);

// Print buffered measurements
for (i=0; i<(INT)dwCount; i++)
    printf("Buffered Meseasured Value Index %d: %f %s\n", i,

```

```
        adMeasurements[i], szUnits);
    }

    getchar();
    return 0;
}

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


Chapter 5 - Calibration

Introduction

The GX2065 can be calibrated using the GXDMM driver API (for boards with EEPROM version 'A' only) or by using Marvin Test Solutions' CalEasy software. This chapter focuses on using the driver API for calibration. Using either method of calibration requires purchasing of a calibration license for CalEasy or for the API from Marvin Test Solutions.

Note: This chapter pertains only for boards with EEPROM version 'A', boards with EEPROM version 'B' and above need to use CalEasy. The board's EEPROM version may be retrieved by calling the GxDmmGetBoardSummary API.

The GXDMM driver API exposes a set of functions to allow the end user to create their own calibration program. The GX2065 stores all calibration data within an onboard EEPROM. This EEPROM has enough storage to include up to three User Calibration sets. The user can choose to load any of these User Calibration sets as well as the Factory Calibration data.

Hardware Requirements

In order to calibrate the GX2065 the user must have access to a high accuracy, comprehensive calibrator such as the **Fuke 5522A** and a 6 ½ digit reference **DMM** (for resistance and VDC measurements). Alternatively an 8 ½ DMM and a stable reference source may be used as a substitute the calibrator. The calibrator or DMM/Source must support the following functions: DC voltage, AC voltage, DC current, AC current, 2 Wire Resistance, and 4 Wire Resistance. The calibrator or reference source must be able to supply +/- 300 DC Volts, 300 AC Volts, 2A DC Current, and 2A AC Current. The calibrator or reference must also be able provide up to 100 M Ohm of resistance for 2Wire and support 4Wire mode.

The following table provides a guideline on the accuracy specifications required from the Calibrator:

DC Voltage	AC Voltage (1Khz)	DC Current	AC Current (1Khz)	Resistance
100mV +/- 30ppm 1V +/- 13ppm 10V +/- 14ppm 100V +/- 20ppm 300V +/- 19ppm	70mV +/- 200ppm 0.7V +/- 82ppm 7V +/- 82ppm 70V +/- 90ppm 300V +/- 85ppm	20mA +/- 60ppm 100mA +/- 70ppm 1A +/- 110mA 2A +/- 110mA	14mA +/- 200ppm 70mA +/- 190ppm 0.7A +/- 690ppm 1.4A +/- 670ppm	100Ohm +/-12ppm 1kOhm +/-12ppm 10kOhm +/- 11ppm 100kOhm +/- 13ppm 1MOhm +/- 18ppm 10MOhm +/- 22ppm 100MOhm +/- 25ppm

Table 5-1: Hardware Requirements

Warm up Time

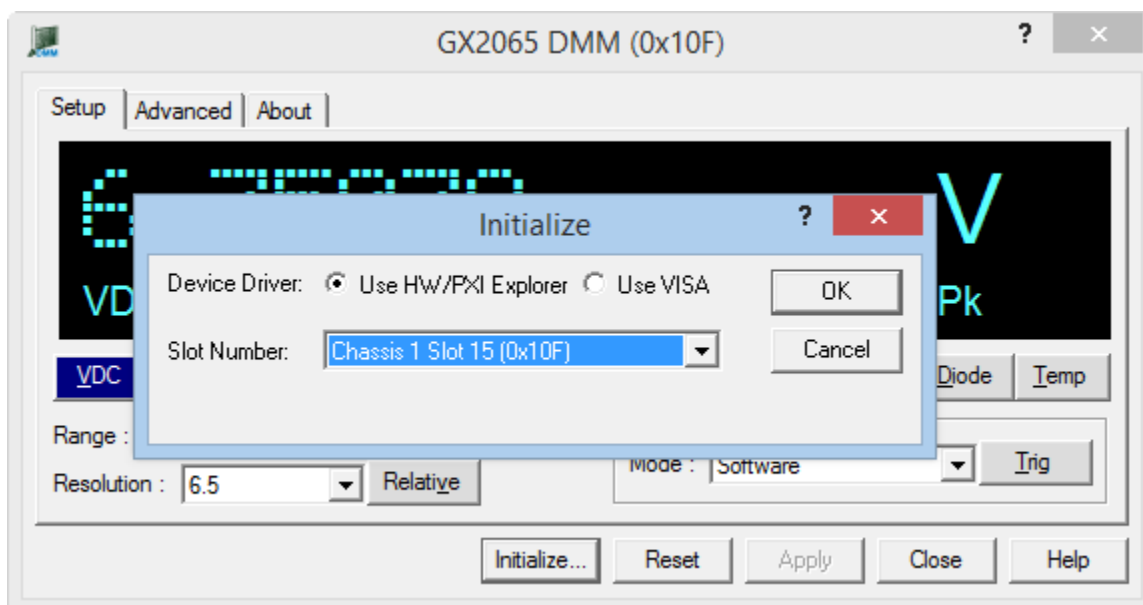
Allow the GX2065 DMM to warm up for at least 2 hours before performing calibration.

Calibration Licensing

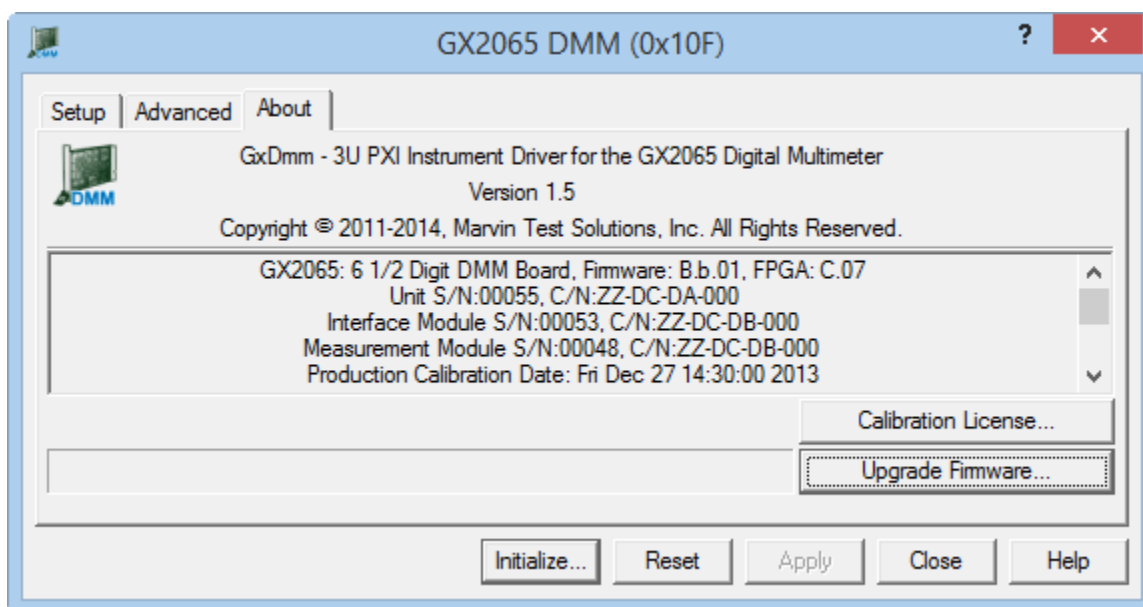
A calibration license must be obtained from Marvin Test Solutions in order to unlock the Calibration functionality.

The software front panel is used to enter a valid license string using the following procedure:

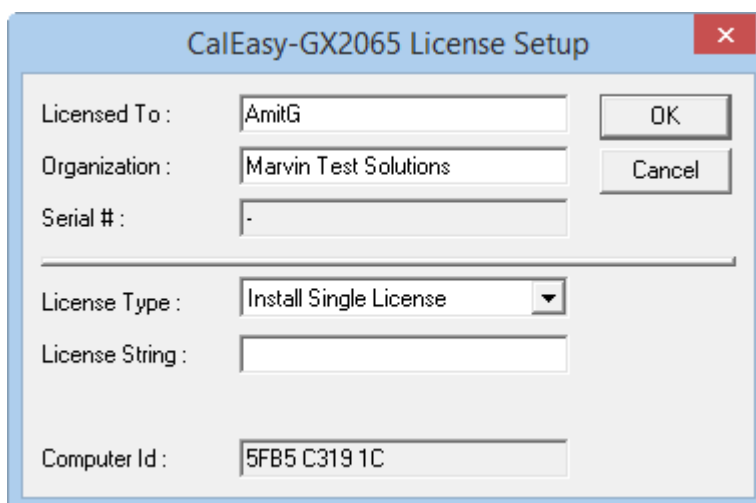
1. Initialize the software front panel:



2. Click on the About Tab:



3. Click on the Calibration License... button.
4. Make note of the Computer ID. Send this ID to Marvin Test Solutions in exchange for a License String.
5. Fill in the appropriate fields including the License String and click OK.



At this point the Calibration functionality has been unlocked on a specific computer.

GXDMM Functions used for Calibration

The functions used for calibration are:

Function	Description
GxDmmInitialize	Initializes the GX2065 Pass the hex value 0x1234 to the <i>pnStatus</i> parameter to unlock calibration functionality.
GxDmmSetCalibrationMeasurements	Set the positive, negative, and ground data points for each function/range combination for a particular Calibration Set. Optionally, the reference positive, negative, and ground values can be set if using a reference DMM, otherwise these parameters should be set to the expected output of the Calibrator in base units (Volts, Amps, or Ohms).
GxDmmWriteCalibrationEEPROM	Finalize calibration by writing Calibration Set data to the onboard EEPROM
GxDmmSetCalibrationSet	Load a Calibration Set. This is used for loading the DMM with the For Calibration Set before starting the Calibration procedure
GxDmmMeasure	Takes a measurement from the DMM This is used to take measurements from the DMM and pass them to GxDmmSetCalibrationMeasurements
GxDmmSetFunction	Set a DMM Function This is used to cycle between the different functions during Calibration
GxDmmSetRange	Set the range for the currently selected Function This is used to cycle between all the possible ranges for a given function during Calibration
GxDmmSetACMath	Set the mathematical formula used during AC measurements This is used when calibrating the AC functions.

Table 5-2: GXDMM Calibration Functions

For further information on these functions, review the **Function Reference** section of this manual.

Calibration Procedure

Use the CalEasy user guide for connection information to perform this procedure.

Initial Calibration Procedure

1. Initialize the GX2065 (**GxDmmInitialize**)
2. Reset GX2065
3. Allow **GX2065** and **Calibrator** to **Warm Up**

DC Voltage Calibration

1. Set **the GX2065** to **Function** GXDMM_FUNCTION_VDC
2. Set **the GX2065** to **Range** as listed in the table.
3. Set **Calibrator** to provide ground value (0 Volts for the VDC 10V Range etc.)
4. Take a Measurement from the **GX2065**
5. Set **Calibrator** to provide the positive extreme of the range to be calibrated (+10 Volts for VDC 10V Range etc.)
6. Take a Measurement from the **GX2065**
7. Set **the Calibrator** to provide the negative extreme of the range to be calibrated (-10 Volts for VDC 10V Range etc.)
8. Take a Measurement from the **GX2065**
9. Call **GXDMM API GxDmmSetCalibrationMeasurements** to write values from the 3 measurements (measurements from steps 4, 6, 8) to a User Calibration Set. The calibrator values used in steps 3,5, and 7 should be used to fill in the other 3 parameters of **GxDmmSetCalibrationMeasurements**
10. Repeat steps 1 to 7 for each **Range** listed in the table.

Calibration Group	Range	Positive Measurement	Negative Measurement	Ground Measurement
GXDMM_CAL_HiRes100mVDC	100mV	+100 mV	-100 mV	0 V
GXDMM_CAL_HiRes1VDC	1V	+1 mV	-1 V	0 V
GXDMM_CAL_HiRes10VDC	10V	+10 V	-10 V	0 V
GXDMM_CAL_HiRes100VDC	100V	+100 V	-100 V	0 V
GXDMM_CAL_HiRes300mVDC	300V	+300 V	-300 V	0 V

Table 5-3: DC Voltage Calibration Groups

AC Voltage DC Coupled Calibration

1. Set **the GX2065** to **Function** GXDMM_FUNCTION_VAC_DC_CPL
2. Set **the GX2065** to **Range** as listed in the table.
3. Set **the GX2065 Math Type** to GXDMM_AC_MATH_AVG
4. Set **Calibrator** to provide ground value (0 Volts for the VDC 10V Range etc.)
5. Take a Measurement from the **GX2065**
6. Set the **Calibrator** to provide the positive extreme of the range to be calibrated (+10 Volts for VDC 10V Range etc.)

7. Take a Measurement from the **GX2065**
8. Set the **Calibrator** to provide the negative extreme of the range to be calibrated (-10 Volts for VDC 10V Range etc.)
9. Take a Measurement from the **GX2065**
10. Call the **GXDMM** API to write values from the 3 measurements (measurements from steps 6, 7, 9) to a User Calibration Set.
11. Repeat steps 4 to 10 for each **Calibration Set** listed in the table.

Calibration Group	Range	Positive Measurement	Negative Measurement	Ground Measurement
GXDMM_CAL_Daq100mVDC	100mV	+100 mV	-100 mV	0 V
GXDMM_CAL_Daq1VDC	1V	+1 V	-1 V	0 V
GXDMM_CAL_Daq10VDC	10V	+10 V	-10 V	0 V
GXDMM_CAL_Daq100VDC	100V	+100 V	-100 V	0 V
GXDMM_CAL_Daq300mVDC	300V	+300 V	-300 V	0 V

Table 5-4: AC Voltage DC Coupled Calibration Groups

AC Voltage AC Coupled Calibration

1. Set the **GX2065** to **Function** GXDMM_FUNCTION_VAC_AC_CPL
2. Set the **GX2065** to **Range** as listed in the table.
3. Set the **GX2065 Math Type** to GXDMM_AC_MATH_RMS
4. Set the **Calibrator** to provide ground value (0 Volts for the VDC 10V Range etc.)
5. Take a Measurement from the **GX2065**
6. Set the **Calibrator** to provide the positive extreme of the range to be calibrated (+10 Volts for VDC 10V Range etc.)
7. Take a Measurement from the **GX2065**
8. Call the **GXDMM** API to write values from the 2 measurements (measurements from steps 5, 7) to a User Calibration Set.
9. Repeat steps 4 to 8 for each **Calibration Set** listed in the table.

Calibration Group	Range	Positive Measurement	Ground Measurement
GXDMM_CAL_Daq100mVAC	50mV	+70 mV (10KHz)	0 V
GXDMM_CAL_Daq1VAC	500 mV	+0.7 V (10KHz)	0 V
GXDMM_CAL_Daq10VAC	5V	+7 V (10KHz)	0 V
GXDMM_CAL_Daq100VAC	50V	+70 V (10KHz)	0 V
GXDMM_CAL_Daq300mVAC	300V	+140 V (10KHz)	0 V

Table 5-5: AC Voltage AC Coupled Calibration Groups

DC Current Calibration

1. Set the **GX2065** to **Function** GXDMM_FUNCTION_IDC
2. Set the **GX2065** to **Range** as listed in the table.
3. Disconnect the **GX2065** to create open circuit
4. Take a Measurement from the **GX2065**
5. Set the **Calibrator** to provide the positive extreme of the range to be calibrated (+1 Amp for IDC 1A Range etc.)
6. Take a Measurement from the **GX2065**
7. Set the **Calibrator** to provide the negative extreme of the range to be calibrated (-1 Amp for IDC 1A Range etc.)
8. Take a Measurement from the **GX2065**
9. Call the **GXDMM** API to write values from the 3 measurements (measurements from steps 4, 6, 8) to a User Calibration Set. The calibrator values from steps 5 and 7 along with a ground measurement of 0 should be used to fill in the other 3 parameters of **GxDmmSetCalibrationMeasurements**.
10. Repeat steps 1 to 7 for each **Range** listed in the table.

Calibration Group	Range	Positive Measurement	Negative Measurement	Ground Measurement
GXDMM_CAL_HiRes20mADC	20mA	+20 mA	-20 mA	Open Circuit
GXDMM_CAL_HiRes100mADC	100mA	+100 mA	-100 mA	Open Circuit
GXDMM_CAL_HiRes1ADC	1A	+1 A	-1 A	Open Circuit
GXDMM_CAL_HiRes2ADC	2A	+2 A	-2 A	Open Circuit

Table 5-6: DC Current Calibration Groups

AC Current DC Coupled Calibration

1. Set the **GX2065** to **Function** GXDMM_FUNCTION_IAC_DC_CPL
2. Set the **GX2065** to **Range** as listed in the table.
3. Set the **GX2065 Math Type** to GXDMM_AC_MATH_AVG
4. Disconnect the **GX2065** to create open circuit
5. Take a Measurement from the **GX2065**
6. Set the **Calibrator** to provide the positive extreme of the range to be calibrated (+1 Amp for IDC 1A Range etc.)
7. Take a Measurement from the **GX2065**
8. Set the **Calibrator** to provide the negative extreme of the range to be calibrated (-1 Amp for IDC 1A Range etc.)
9. Take a Measurement from the **GX2065**
10. Call the **GXDMM** API to write values from the 3 measurements (measurements from steps 5, 7, 9) to a User Calibration Set. The calibrator values from steps 6 and 8 along with a ground measurement of 0 should be used to fill in the other 3 parameters of **GxDmmSetCalibrationMeasurements**.

11. Repeat steps 2 to 10 for each **Calibration Set** listed in the table.

Calibration Group	Range	Positive Measurement	Negative Measurement	Ground Measurement
GXDMM_CAL_Daq20mADC	10mA	+14 mA	-20 mA	Open Circuit
GXDMM_CAL_Daq100mADC	50mA	+70 mA	-100 mA	Open Circuit
GXDMM_CAL_Daq1ADC	0.5A	+0.7 A	-1 A	Open Circuit
GXDMM_CAL_Daq2ADC	1.0A	+1.4 A	-2 A	Open Circuit

Table 5-7: AC Current DC Coupled Calibration Groups

AC Current AC Coupled Calibration

1. Set the **GX2065** to **Function** GXDMM_FUNCTION_IAC_AC_CPL
2. Set the **GX2065** to **Range** as listed in the table.
3. Set the **GX2065 Math Type** to GXDMM_AC_MATH_RMS
4. Disconnect the GX2065 to create open circuit
5. Take a Measurement from the **GX2065**
6. Set the **Calibrator** to provide the positive extreme of the range to be calibrated (+1 Amp for IDC 1A Range etc.)
7. Take a Measurement from the **GX2065**
8. Call the **GXDMM** API to write values from the 2 measurements (measurements from steps 5, 7) to a User Calibration Set. The calibrator value from steps 6 along with a ground measurement of 0 should be used to fill in the other 2 parameters of **GxDmmSetCalibrationMeasurements**. Set unused parameters to 0.
9. Repeat steps 4 to 8 for each **Calibration Set** listed in the table.

Calibration Group	Range	Positive Measurement	Ground Measurement
GXDMM_CAL_Daq20mAAC	14mA	+14 mA (10KHz)	Open Circuit
GXDMM_CAL_Daq100mAAC	70mA	+70 mA V (10KHz)	Open Circuit
GXDMM_CAL_Daq1AAC	0.7A	+0.7 A (10KHz)	Open Circuit
GXDMM_CAL_Daq2AAC	1.4A	+1.4 A (10KHz)	Open Circuit

Table 5-8: AC Current AC Coupled Calibration Groups

Resistance 2 Wire Calibration

1. Set the **GX2065** to **Function** GXDMM_FUNCTION_2WIRE_OHM
2. Set the **GX2065** to **Range** as listed in the table.
3. Set the Calibrator to provide a ground value (0 Ohm Resistance)
4. Take a Measurement from the **GX2065**
5. Set the **Calibrator** to provide the positive extreme of the range to be calibrated (100 Ohm for 100 Ohm Range etc.)
6. Take a Measurement from the **GX2065**
7. Call the **GXDMM** API to write values from the 2 measurements (measurements from steps 4, 6) to a User Calibration Set. The calibrator value from steps 5 along with a ground measurement of 0 should be

used to fill in the other 2 parameters of **GxDmmSetCalibrationMeasurements**. Set unused parameters to 0.

8. Repeat steps 3 to 7 for each **Calibration Set** listed in the table.

Calibration Group	Range	Positive Measurement	Ground Measurement
GXDMM_CAL_HiRes100_Ohms2W	100 Ohm	100 Ohm	0 Ohm
GXDMM_CAL_HiRes1k_Ohms2W	1 KOhm	1 KOhm	0 Ohm
GXDMM_CAL_HiRes10k_Ohms2W	10 KOhm	10 KOhm	0 Ohm
GXDMM_CAL_HiRes100k_Ohms2W	100 KOhm	100 KOhm	0 Ohm
GXDMM_CAL_HiRes1M_Ohms2W	1 MOhm	1 MOhm	0 Ohm
GXDMM_CAL_HiRes10M_Ohms2W	10 MOhm	10 MOhm	0 Ohm
GXDMM_CAL_HiRes100M_Ohms2W	100 MOhm	100 MOhm	0 Ohm

Table 5-9: Resistance 2 Wire Calibration Groups

Resistance 4 Wire Calibration

1. Set the **GX2065** to **Function** GXDMM_FUNCTION_4WIRE_OHM
2. Set the **GX2065** to **Range** as listed in the table.
3. Set the Calibrator to provide a ground value (0 Ohm Resistance)
4. Take a Measurement from the **GX2065**
5. Set the **Calibrator** to provide the positive extreme of the range to be calibrated (100 Ohm for 100 Ohm Range etc.)
6. Take a Measurement from the **GX2065**
7. Call **GXDMM** API to write values from the 2 measurements (measurements from steps 4, 6) to a User Calibration Set. The calibrator value from steps 5 along with a ground measurement of 0 should be used to fill in the other 2 parameters of **GxDmmSetCalibrationMeasurements**. Set unused parameters to 0.
8. Repeat steps 3 to 7 for each **Calibration Set** listed in the table.

Calibration Group	Range	Positive Measurement	Ground Measurement
GXDMM_CAL_HiRes100_Ohms4W	100 Ohm	100 Ohm	0 Ohm
GXDMM_CAL_HiRes1k_Ohms4W	1 KOhm	1 KOhm	0 Ohm
GXDMM_CAL_HiRes10k_Ohms4W	10 KOhm	10 KOhm	0 Ohm
GXDMM_CAL_HiRes100k_Ohms4W	100 KOhm	100 KOhm	0 Ohm
GXDMM_CAL_HiRes1M_Ohms4W	1 MOhm	1 MOhm	0 Ohm
GXDMM_CAL_HiRes10M_Ohms4W	10 MOhm	10 MOhm	0 Ohm
GXDMM_CAL_HiRes100M_Ohms4W	100 MOhm	100 MOhm	0 Ohm

Table 5-10: Resistance 4 Wire Calibration Groups

Resistance Open Calibration

1. Set **the GX2065** to **Function** GXDMM_FUNCTION_2WIRE_OHM
2. Set **the GX2065** to **Range** to 10MOhm
3. Disconnect **Calibrator**
4. Set the **Reference DMM** Function to VDC
5. Set the **Reference DMM** Range to 10 Volts
6. Take a Measurement from the **GX2065**
7. Call **GxDmmSetCalibrationMeasurements** API to write values from the measurement (from step 6) to a User Calibration Set, passing it in using the **dRefPositiveMeasurement** parameter. Set unused parameters to 0.

RRef Calibration

1. Set the **Calibrator** to 2Wire resistance and 10MOhm
2. Set the **Gx2065** to **Function** GXDMM_FUNCTION_VDC
3. Set **the GX2065** to **Range** to 10V
4. Set the **Reference DMM** to **Function** 2Wire
5. Set the **Reference DMM** to **Range** 10MOhm
6. Take a Measurement from the **Reference DMM** (PosMeasurement)
7. Set the **Reference DMM** to **Function** VDC
8. Set the **Reference DMM** to **Range** 10V
9. Set the **GX2065** to **Function** GXDMM_FUNCTION_2WIRE
10. Set the **GX2065** to **Range** 10MOhm
11. Take a Measurement from the **Reference DMM** (NegMeasurement)
12. Calculate the value of **dRefPositiveMeasurement**=

$$dOpenCalVoltage * (PosMeasurement / NegMeasurement) - PosMeasurement$$
13. Call **GxDmmSetCalibartionMeasurements** API and pass in the calculated **dRefPositiveMeasurement** from step 12. Set unused parameters to 0.

Final Calibration Procedure

1. Write the User Calibration Set to EEPROM using the **GxDmmWriteCalibrationEEPROM** API.
2. Reset **the GX2065** to load new calibration values

Chapter 6 - Function Reference

Introduction

The GXDMM's driver functions are organized in alphabetical order. Each function is presented starting with the syntax of the function, a short description of the function parameters description and type followed by a Comments, an Example (written in C), and a See Also sections.

All function parameters follow the same rules:

- Strings are ASCIIZ (null or zero character terminated).
- Most function's first parameter is *nHandle* (16-bit integer). This parameter is required for operating the board and is returned by the **GxDmmInitialize** or the **GxDmmInitializeVisa** 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 **GxDmmGetErrorString** function or by using a predefined constant, defined in the driver interface files: GxDmm.h, GxDmm.bas, GXDMMVB, GxDmm.pas or GX2065.drv.
- Parameter name are prefixed as follows:

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

Table 6-1: Parameter Prefixes

GXDMM Functions

The following list is a summary of functions available for the GX2065:

Driver Functions	Description
General Functions	
GxDmmInitialize	Initializes the driver for the board at the specified slot number using HW. The function returns a handle that can be used with other GXDMM functions to program the board
GxDmmInitializeVisa	Initializes the driver for the specified slot using VISA. The function returns a handle that can be used with other GXDMM functions to program the board.
GxDmmPanel	Launches the software front panel
GxDmmReset	Resets the instrument
GxDmmGetCalibrationInfo	Returns calibration information
GxDmmGetDriverSummary	Returns driver information
GxDmmGetErrorString	Returns an error code description
Setup and Measurements Functions	
GxDmmAbortReading	Aborts a reading that is in progress
GxDmmSetACClockDivider	Returns the AC DAQ clock divider
GxDmmGetACMath	Returns the AC function math type
GxDmmGetACMinFrequency	Returns the AC minimum frequency
GxDmmGetAperture	Returns the measurement Aperture time in Line Cycles
GxDmmGetAutoRange	Returns Auto Range status
GxDmmGetAutoZero	Returns Auto Zero status
GxDmmGetBoardSummary	Returns Board Summary
GxDmmGetCalibrationSet	Returns Current Calibration Set
GxDmmGetExternalTriggers	Returns the external triggers directions and edge sensitivities
GxDmmGetFunction	Returns the currently selected function
GxDmmGetLineFrequency	Returns AC Power Line Frequency
GxDmmGetMinMax	Returns the minimum maximum value recorded since the last function selection
GxDmmGetRange	Returns the current range
GxDmmReadArray	Reads the captured array of measurements
GxDmmGetReadArrayStatus	Returns status of buffered measure and read
GxDmmGetReadStatus	Checks if the current reading is fresh
GxDmmGetRelative	Returns the relative measurement state
GxDmmGetResolution	Returns the resolution
GxDmmGetTriggerMode	Returns the triggering mode
GxDmmMeasure	Takes measurement as floating point value

Driver Functions	Description
GxDmmMeasureEx	Takes a measurement and return the measurement as a floating point value, a normalized value as a formatted string, the units of measurement as a string, and the time stamp
GxDmmRead	Reads the last measurement taken as a floating point value
GxDmmReadEx	Reads the last measurement taken as a floating point value, a normalized value as a formatted string, the units of measurement as a string, and the time stamp
GxDmmRestoreFactoryCalibration	Restores the original Factory calibration
GxDmmSetACClockDivider	Sets the AC DAQ clock divider
GxDmmSetACMath	Sets the AC function math type
GxDmmSetACMinFrequency	Sets the AC minimum frequency
GxDmmSetAperture	Sets the measurement Aperture time in Line Cycles
GxDmmSetAutoRange	Sets Auto Range status
GxDmmSetAutoZero	Sets Auto Zero status
GxDmmSetCalibrationSet	Sets Current Calibration Set
GxDmmSetExternalTriggers	Sets the external triggers directions and edge sensitivities
GxDmmSetFunction	Sets the measurement function
GxDmmSetLineFrequency	Sets AC Power Line Frequency
GxDmmSetRange	Sets the current range
GxDmmSetRelative	Sets the relative measurement state
GxDmmSetResolution	Sets the resolution
GxDmmSetTriggerMode	Sets the triggering mode
GxDmmTrig	Generate a software trigger to a take a measurement
Calibration Functions (requires special license)	
GxDmmSetCalibrationMeasurements	Sets calibration measurements for the selected Calibration set
GxDmmWriteCalibrationEeprom	Writes calibration values to EEPROM for selected Calibration set

GxDmmAbortReading

Purpose

Aborts any currently executing measurements

Syntax

GxDmmAbortReading (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function will abort any currently executing measurement operation and result in the trigger reading being reset to software trigger mode.

Example

The following example aborts the current measurement:

```
SHORT  nHandle, nStatus;
```

```
GxDmmAbortReading (nHandle, &nStatus);
```

See Also

GxDmmSetTriggerMode, **GxDmmGetErrorString**

GxDmmGetACClockDivider

Purpose

Gets the AC DAQ clock divider

Syntax

GxDmmGetACClockDivider (*nHandle*, *pdwClockDivider*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdwClockDivider</i>	PDWORD	Gets the clock divider for the AC DAQ sampling clock. (1-255)
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the DMM is in any of the AC functions (VAC AC Coupled, VAC DC Coupled, IAC AC Coupled, or IAC DC Coupled), an internal DAQ is used to acquire up to 8192 samples of data at up to 3MS/s. The DAQ sampling clock (3 MHz) can be divided to achieve a slower sampling rate.

Example

The following example sets the AC DAQ Clock divider to 2 so that the effective sampling rate becomes 1.50MS/s :

```
SHORT  nHandle, nStatus;
DWORD  dwACClockDivider;
```

```
GxDmmGetACClockDivider (nHandle, &dwACClockDivider, &nStatus);
```

See Also

GxDmmSetACClockDivider, GxDmmSetACMath, GxDmmGetACMath, GxDmmSetACMinFrequency, GxDmmGetACMinFrequency, GxDmmGetAutoRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmSetRange, GxDmmGetRange, GxDmmGetErrorString

GxDmmGetACMath

Purpose

Gets the AC function math type

Syntax

GxDmmGetACMath (*nHandle*, *pdwMathType*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdwMathType</i>	PDWORD	Gets the math function applied the raw measurement in any of the AC measurement functions. <ol style="list-style-type: none"> 0. GXDMM_AC_MATH_RMS – Sets the AC math type to RMS (root mean square). 1. GXDMM_AC_MATH_AVG – Sets the AC Math type to Average. This math operation will take the average (arithmetic mean) of all the DAQ samples taken. 2. GXDMM_AC_MATH_PP – Sets the AC math type to Peak to Peak. The DMM will return the difference between the largest value recorded and the smallest value recorded. 3. GXDMM_AC_MATH_PKAMP – Sets the AC math type to Peak amplitude. The DMM will return the absolute maximum value recorded (positive or negative) subtracted by the arithmetic mean.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the DMM is in any of the AC functions (VAC AC Coupled, VAC DC Coupled, IAC AC Coupled, or IAC DC Coupled), an internal DAQ is used to acquire up to 8192 samples of data at up to 2MS/s. A math operation (from the types listed above) is applied to the samples to generate a final measurement result. Use the **GxDmmSetACMinFrequency** function to set the lowest expected frequency.

Example

The following example sets the AC math type to RMS:

```
SHORT  nHandle, nStatus;
DWORD  dwACMathType;

GxDmmGetACMath (nHandle, &dwACMathType, &nStatus);
```

See Also

GxDmmSetACMath, **GxDmmSetACMinFrequency**, **GxDmmGetACMinFrequency**, **GxDmmGetAutoRange**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmGetACMinFrequency

Purpose

Gets the AC minimum frequency

Syntax

GxDmmGetACMath (*nHandle*, *pdFrequency*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdFrequency</i>	PDOUBLE	Gets the minimum frequency expected to capture by the DMM for an AC measurement.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the DMM is in any of the AC functions (VAC AC Coupled, VAC DC Coupled, IAC AC Coupled, or IAC DC Coupled), an internal DAQ is used to acquire up to 8192 samples of data at up to 2MS/s. A math operation (from the types selectable using **GxDmmSetACMath**) is applied to the samples to generate a final measurement result. The user must specify the minimum AC frequency so that the timing of the DAQ can be adjusted to capture at least 4 cycles of the waveform, with the maximum number of samples per cycle. Higher frequencies can also be captured, but the number of samples per cycle is reduced.

Example

The following example sets the AC minimum frequency to 500Hz:

```
SHORT  nHandle, nStatus;
DOUBLE dACMinFrequency
```

```
GxDmmGetACMinFrequency (nHandle, &dACMinFrequency, &nStatus);
```

See Also

GxDmmGetACMinFrequency, **GxDmmGetACMath**, **GxDmmGetAutoRange**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmGetAperture

Purpose

Returns the capture aperture in terms of line cycles

Syntax

GxDmmGetAperture (*nHandle*, *pdAperture* , *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdAperture</i>	PBOOL	Returns the capture aperture in terms of line cycles (0.02-65)
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The aperture is expressed in terms of Line Cycles as set by the **GxDmmSetLineFrequency** function. The Aperture can be a fraction of a line cycle. For example, the aperture can be set to 0.1 Line Cycles which would represent approximately 0.00167 seconds if the Line Frequency was set to 60Hz.

Example

The following example gets the aperture:

```
SHORT  nHandle, nStatus;
DOUBLE dAperture;

GxDmmGetAperture (nHandle, &dAperture, &nStatus);
```

See Also

GxDmmSetAperture, **GxDmmSetLineFrequency**, **GxDmmGetLineFrequency**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmGetAutoRange

Purpose

Returns the auto range status

Syntax

GxDmmGetAutoRange (*nHandle*, *pbAutoRange* , *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pbAutoRange</i>	PBOOL	Returns the status of the Auto Range function.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The auto range automatically adjusts the range with respect to the input signal. Range changes occur on overflow and when the signal is less than 8% of the current range.

Example

The following example gets the auto range status:

```
SHORT  nHandle, nStatus;
BOOL   bAutoRange;
```

```
GxDmmGetAutoRange (nHandle, &bAutoRange, &nStatus);
```

See Also

GxDmmSetAutoRange, GxDmmSetFunction, GxDmmSetAperture, GxDmmGetFunction, GxDmmSetRange, GxDmmGetRange, GxDmmGetErrorString

GxDmmGetAutoZero

Purpose

Returns the auto zero status

Syntax

GxDmmGetAutoZero(*nHandle*, *pdwAutoZeroMode* , *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdwAutoZeroMode</i>	PDWORD	Returns the status of the Auto Zero function. 0. GXDMM_AUTO_ZERO_ON: The DMM will perform an auto zero every time a measurement is taken. 1. GXDMM_AUTO_ZERO_OFF: The DMM will not auto zero when a measurement is taken. 2. GXDMM_AUTO_ZERO_NOW: The DMM will perform an auto zero calculation as soon as this command is received.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

To help maintain stability and accuracy over time and changes in temperature, the DMM periodically measures internal voltages corresponding to offsets (zero) and amplifier gains. These measurements are used in an algorithm to calculate the reading of the input signal. This process is known as auto zeroing. When auto zero is disabled, the offset, gain, and internal temperature measurements are not performed. This increases the measurement speed (reading rate). However, the zero, gain, and temperature reference points will eventually drift resulting in less accurate readings of the input signal.

It is recommended that auto zero only be disabled for short periods of time.

An auto zero can also be forced by using the **GXDMM_AUTO_ZERO_NOW** mode.

Example

The following example gets the auto zero status:

```
SHORT  nHandle, nStatus;
DOUBLE dMeasurement;

GxDmmGetAutoZero (nHandle, &dMeasurement, &nStatus);
```

See Also

GxDmmSetAutoZero, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmGetBoardSummary

Purpose

Returns the board information

Syntax

GxDmmGetBoardSummary (*nHandle*, *pszSummary*, *nSumMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pszSummary</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 DMM summary string provides the following data from the board in the order shown:

- Instrument Name (e.g., GX2065: 6 ½ Digit DMM Board)
- Unit S/N (e.g., 00001)
- Unit C/N (e.g., '*-AA-AA-00')
- Interface Module S/N (e.g., 00001)
- Interface Module C/N (e.g., '*-AC-AA-00')
- Measurement Module S/N (e.g., 00001)
- Measurement Module C/N (e.g., '*-AA-AB-00')
- Firmware Version (A.1.9)

For example, the returned string for GX2065 looks like the following:

```
GX2065: 6 1/2 Digit DMM Board, Firmware: B.a.00, FPGA: C.06, EEPROM: A
Unit S/N:00051, C/N:ZZ-DC-DA-000
Interface Module S/N:00055, C/N:ZZ-DC-DB-000
Measurement Module S/N:00049, C/N:ZZ-DC-DB-000
Production Calibration Date: Tue Feb 19 17:07:50 2013
Calibration Date: Tue Feb 19 17:07:50 2013
Recommended Interval: 1 year
Next Calibration Date: Wed Feb 19 17:07:50 2014
Status: Expired (118 days past expiration)
```

Example

The following example gets the board summary:

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

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

See Also

GxDmmInitialize, GxDmmInitializeVisa, GxDmmGetErrorString

GxDmmGetCalibrationInfo

Purpose

Returns the calibration information

Syntax

GxDmmGetCalibrationInfo (*nHandle*, *pszCalibrationInfo*, *nInfoMaxLen*, *pnDaysUntilExpire*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle to a DMM board.
<i>pszSummary</i>	PSTR	Buffer to contain the returned board's calibration information (null terminated) string.
<i>nSumMaxLen</i>	SHORT	Size of the buffer to contain the error string.
<i>pnDaysUntilExpire</i>	PSHORT	Returns the number of days until or from expiration, if number is > 0 then calibration is current otherwise past due.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The returned board's calibration information has the following fields:

Model: model number, e.g. "GX2065"

Serial Number: serial number, e.g. 216

Control Number: Marvin Test Solutions control number, e.g. "*-CH-CB-0"

Production Calibration Date: Wed Oct 24 12:30:25 2010

Calibration Date: Wed Oct 24 12:31:58 2010

Recommended Interval: 1 year

Next Calibration Date: Fri Oct 24 12:31:58 2011

Status: calibration status can be either "Expired" followed by the number of days past expiration or "Current" followed by number of days until expire.

Calibration License: can be either "Installed" with the calibration license number or "Not Installed".

Example

The following example returns the board's calibration information string:

```
SHORT    nStatus;
char     szCalibrationInfo[1024];
BOOL     bExpired;

GxDmmGetCalibrationInfo(nHandle, szCalibrationInfo, sizeof szCalibrationInfo,
                        &bExpired, &nStatus);
```

szCalibrationInfo string printout:

```
Model: GX2065
Serial Number: 216
Control Number: *-CH-CB-0
Production Calibration Date: Wed Oct 24 12:30:25 2007
Calibration Date: Wed Oct 24 12:31:58 2007
Recommended Interval: 1 year
Next Calibration Date: Fri Oct 24 12:31:58 2008
```

Status: Expired (891 days past expiration)
Calibration License: Installed license number 3

See Also

GxDmmInitialize, GxDmmtGetBoardSummary, GxDmmGetErrorString

GxDmmGetCalibrationSet

Purpose

Returns the currently applied calibration set

Syntax

GxDmmGetCalibrationSet (*nHandle*, *plCalSet*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>plCalSet</i>	PLONG	Returns the currently applied calibration set. The following are acceptable values: 0. GXDMM_CALSET_USER_CALIBRATION: User calibration 1. GXDMM_CALSET_FACTORY_CALIBRATION: Factory calibrated values 2. GXDMM_CALSET_FOR_CALIBRATION: Special calibration set used only when calibrating the DMM
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX2065 has the ability to load different calibration sets dynamically. The EEPROM stores two full calibration sets, User and Factory. The Factory calibration is done at Marvin Test Solutions and cannot be overwritten by the end user. The User Calibration stores a copy of the Factory Calibration but can be overwritten by the User if a Calibration License is active.

The Factory calibration can be restored by calling **GxDmmRestoreFactoryCalibration**. This function will copy the contents of the Factory calibration to the User calibration.

Note that the User Calibration is always loaded by default on initial power up and reset.

Example

The following example gets the currently loaded calibration set:

```
SHORT  nHandle, nStatus;
LONG   lCalSet;

GxDmmGetCalibrationSet (nHandle, &lCalSet, &nStatus);
```

See Also

GxDmmSetCalibrationSet, **GxDmmSetCalibrationMeasurements**,
GxDmmWriteCalibrationEEPROM, **GxDmmRestoreFactoryCalibration**, **GxDmmGetErrorString**

GxDmmGetDriverSummary

Purpose

Returns the driver description string and version number

Syntax

GxDmmGetDriverSummary (*szSummary*, *nSummaryMaxLen*, *pdwVersion*, *pnStatus*)

Parameters

Name	Type	Comments
<i>pszSummary</i>	LPSTR	Buffer to receive the summary string.
<i>nSummaryMaxLen</i>	SHORT	Buffer size passed by pszSummary.
<i>pdwVersion</i>	LPDWORD	Driver version
<i>pnStatus</i>	LPSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns the driver summary:

```
SHORT nHandle, nStatus;  
DWORD dwVersion;  
CHAR szSummary[128];  
GxDmmGetDriverSummary(szSummary, 128, &dwVersion, &nStatus);
```

After the function call the parameter *szSummary* will be set to:

```
GXDMM Driver for GX2065, Version 1.00, Copyright(c) 2011 Marvin Test Solutions, Inc."
```

See Also

GxDmmGetErrorString

GxDmmGetErrorString

Purpose

Returns the error string associated with the specified error number

Syntax

GxDmmGetErrorString (*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
- 30 Unable to Load VISA32/64.DLL, make sure VISA library is installed
- 31 Unable to open default VISA resource manager, make sure VISA is properly installed
- 32 Unable to open the specified VISA resource
- 33 VISA viGetAttribute error
- 34 VISA viInXX error
- 35 VISA ViMapAddress error
- 37 Unable to create synchronization object
- 38 This function is not available under LabView/Real-Time
- 39 Unable to lock a board resource, resource is used by another process or thread
- 40 Unable to execute this function. The function requires special license to be installed

Board Specific Errors

- 98 The previous data was not read and has been overwritten
- 99 Unspecified General Warning
- 101 Unspecified General Error
- 102 One or more of the parameter values is bad
- 103 SPI is busy
- 104 Not Found
- 105 Internal Error Timer by name
- 106 Internal Error Timer by name
- 108 Device is Busy
- 109 A read operation encountered an unspecified error
- 110 The flash failed, did not come out of self timed operation
- 111 Internal Error Flash Write
- 112 The Measurement Board failed to return the serial clock, not present
- 113 The embedded controller has timed out
- 114 Exceeded end of buffer
- 120 Software Generated Timeout
- 121 Communication Error
- 122 Invalid Function Selection
- 123 Invalid Range Selection
- 124 Invalid Resolution Selection
- 125 Invalid Trigger Reading Selection
- 126 Invalid Trigger Timer Rate Selection
- 127 Invalid DAQ Buffer Pointer Selection
- 128 Invalid DAQ Buffer Count Selection
- 129 Invalid HIRES Buffer Pointer Selection

- 130 Invalid HIRES Buffer Count Selection
- 131 Invalid Aperture Selection
- 132 Invalid DAQ Math Type Selection
- 133 Invalid Calibration Selection
- 134 Invalid Calibration Group Selection
- 135 Invalid Calibration Set Selection
- 136 Invalid Minimum AC Frequency
- 137 Invalid AC Mains Line Frequency

Board Specific Warnings

- 100 Warning OverFlow has occurred
- 101 Warning Reading Aborted
- 102 Warning Reading Timeout
- 103 Warning Could not start a new reading because we are busy
- 104 Warning the DAQ was passed an invalid array
- 105 Warning the buffer pointer is past the number captured
- 106 Warning the buffer is empty
- 107 Warning the min/max has not been set yet

Example

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

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

GxDmmInitialize (3, &Handle, &Status);
if (nStatus<0)
{
    GxDmmGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz); // prints the error string returns
}
```

See Also

GxDmmInitialize, GxDmmInitializeVisa, GxDmmReset, GxDmmGetBoardSummary

GxDmmGetExternalTriggers

Purpose

Returns the external trigger configuration

Syntax

GxDmmGetExternalTriggers (*nHandle*, *pdwTriggerDirections*, *pdwTriggerEdges*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdwTriggerDirections</i>	PDWORD	Returns the direction of all external triggers
<i>pdwTriggerEdges</i>	PDWORD	Returns the trigger edge sensitivity for all external triggers
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The *dwTriggerDirections* is a 20 bit long field that determines the directionality of each external trigger. Each bit in the field represents the enable bit for a particular trigger and direction. A '1' indicates enabled and a '0' indicates disabled. By default all bits are set to '0'. The fields are described in the following table:

Bit Number	Description
0	PXI Line 0 Input Enable
1	PXI Line 1 Input Enable
2	PXI Line 2 Input Enable
3	PXI Line 3 Input Enable
4	PXI Line 4 Input Enable
5	PXI Line 5 Input Enable
6	PXI Line 6 Input Enable
7	PXI Line 7 Input Enable
8	PXI Line 0 Output Enable
9	PXI Line 1 Output Enable
10	PXI Line 2 Output Enable
11	PXI Line 3 Output Enable
12	PXI Line 4 Output Enable
13	PXI Line 5 Output Enable
14	PXI Line 6 Output Enable
15	PXI Line 7 Output Enable
16	STAR PXI Input Enable
17	STAR PXI Output Enable
18	LINK Bus Input Enable
19	LINK Bus Output Enable

Table 6-2: dwTriggerDirections Bit Field

The ***dwTriggerEdges*** is a 11 bit long field that determines the edge sensitivity of each external trigger. Each bit in the field represents the edge selected to use for triggering. A '1' indicates rising edge and a '0' indicates falling edge. By default all bits are set to '0'. The fields are described in the following table:

Bit Number	Description
0	PXI Line 0 Edge Selection
1	PXI Line 1 Edge Selection
2	PXI Line 2 Edge Selection
3	PXI Line 3 Edge Selection
4	PXI Line 4 Edge Selection
5	PXI Line 5 Edge Selection
6	PXI Line 6 Edge Selection
7	PXI Line 7 Edge Selection
8	STAR PXI Edge Selection
9	LINK Bus Edge Selection

Table 6-3: dwTriggerEdges Bit Field

Example

The following example gets all the trigger directions and all edge sensitivities to Rising Edge:

```
SHORT  nHandle, nStatus;
```

```
GxDmmGetExternalTriggers (nHandle, &dwTriggerDirections, &dwTriggerEdges, &nStatus);
```

See Also

GxDmmSetExternalTriggers, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmReadArray, GxDmmGetReadArrayStatus, GxDmmAbortReading, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmGetFunction

Purpose

Returns the currently selected function

Syntax

GxDmmGetFunction (*nHandle*, *pdwFunction*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdwFunction</i>	PDWORD	Return the currently set function: <ol style="list-style-type: none"> 0. GXDMM_FUNCTION_VDC – Voltage DC Mode 1. GXDMM_FUNCIONT_IDC – Current DC Mode 2. GXDMM_FUNCTION_VAC_AC_CPL – Voltage AC, AC Coupled Mode 3. GXDMM_FUNCTION_IAC_AC_CPL – Current AC, AC Coupled Mode 4. GXDMM_FUNCTION_VAC_DC_CPL – Voltage AC, DC Coupled Mode 5. GXDMM_FUNCTION_IAC_DC_CPL – Current AC, DC Coupled Mode 6. GXDMM_FUNCTION_2WIRE_OHM – 2Wire Resistance Mode 7. GXDMM_FUNCTION_4WIRE_OHM – 4 Wire Resistance Mode 8. GXDMM_FUNCTION_CONTINUITY – High Resolution Continuity Mode 9. GXDMM_FUNCTION_FREQUENCY – Frequency Counter Mode 10. GXDMM_FUNCTION_DIODE – Diode Voltage Mode 11. GXDMM_FUNCTION_TEMPERATURE – External Temperature Mode 12. GXDMM_FUNCTION_BOARD_TEMPERATURE – Internal, Board Temperature Mode
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX2065 can be set to several different function modes. When the function is changed, the min/max accumulator is cleared and any ongoing measurement is aborted. If the current reading trigger is not set to software trigger, changing the function will result in the reading trigger being set to software trigger, briefly, and then back to its previous setting.

Example

The following example gets the current function:

```
SHORT  nHandle, nStatus;
DWORD  dwFunction;

GxDmmGetFunction (nHandle, &dwFunction, &nStatus);
```

See Also

GxDmmSetFunction, GxDmmSetRange, GxDmmGetRange, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmGetErrorString

GxDmmGetLineFrequency

Purpose

Returns the AC Power Line Frequency setting

Syntax

GxDmmGetLineFrequency (*nHandle*, *pdwLineFrequency*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdwLineFrequency</i>	PDWORD	Returns the AC Power Line Frequency selection <ul style="list-style-type: none"> GXDMM_LINE_FREQUENCY_50HZ (50): 50Hz AC Line Frequency GXDMM_LINE_FREQUENCY_60HZ (60): 60Hz AC Line Frequency GXDMM_LINE_FREQUENCY_400HZ (400): 400Hz AC Line Frequency
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows the user to select the line frequency of the Mains Source powering the chassis. Setting this parameter correctly reduces the amount of noise in all measurements.

The last set AC line frequency is stored within the GxDmm.ini file located in Windows Common App Data folder. The input AC line frequency is loaded from the GxDmm.ini file when first initializing the DMM after power up, if the file does not exist 60Hz is used.

Example

The following example gets the line frequency setting:

```
SHORT    nHandle, nStatus;
DWORD    dwLineFrequency;

GxDmmGetLineFrequency(nHandle, &dwLineFrequency, &nStatus);
```

See Also

GxDmmInitialize, GxDmmReset, GxDmmSetLineFrequency, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmGetMinMax

Purpose

Returns the maximum value recorded since changing function

Syntax

GxDmmGetMinMax (*nHandle*, *pdMin*, *pdMax*, *bClear*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdMin</i>	PDOUBLE	Returns the minimum value recorded
<i>pdMax</i>	PDBOULE	Returns the maximum value recorded
<i>bClear</i>	BOOL	Resets the minimum/maximum accumulators
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The maximum and minimum accumulators are reset when the function is changed or when **bClear** is set to TRUE.

Example

The following example returns the minimum and maximum values without resetting the accumulators:

```
SHORT  nHandle, nStatus;
DOUBLE dMin, dMax;

GxDmmGetMinMax (nHandle, &dMin, &dMax, FALSE, &nStatus);
```

See Also

GxDmmSetFunction, **GxDmmGetFunction**, **GxDmmGetErrorString**

GxDmmGetRange

Purpose

Returns the currently selected function's range

Syntax

GxDmmGetRange (*nHandle*, *pdRange*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdRange</i>	PDOUBLE	<p>Returns the currently selected function's range.</p> <ul style="list-style-type: none"> GXDMM_RANGE_100mV (0.100): 100 millivolts Range for VDC functions GXDMM_RANGE_1V (1): 1 volt Range for VDC functions GXDMM_RANGE_10V (10): 10 volts Range for VDC functions GXDMM_RANGE_100V (100): 100 volts Range for VDC functions GXDMM_RANGE_300V (300): 300 volts Range for VDC functions GXDMM_RANGE_50mVrms (0.050): 50 millivolts Range for VAC functions GXDMM_RANGE_500mVrms (0.50): 500 millivolts Range for VAC functions GXDMM_RANGE_5Vrms (5): 5 volts Range for VAC functions GXDMM_RANGE_50Vrms (50): 50 volts Range for VAC functions GXDMM_RANGE_300Vrms (300): 300 volts Range for VAC functions GXDMM_RANGE_20mA (0.020): 20 milliamps Range for IDC functions GXDMM_RANGE_100mA (0.100): 100 milliamps Range for IDC functions GXDMM_RANGE_1A (1): 1 amp Range for IDC functions GXDMM_RANGE_2A (2): 2 amps Range for IDC functions GXDMM_RANGE_10mArms (0.010): 10 milliamps Range for IAC functions GXDMM_RANGE_50mArms (0.050): 50 milliamps Range for IAC functions GXDMM_RANGE_500mArms (0.50): 500 milliamps Range for IAC functions GXDMM_RANGE_1Arms (1.0): 1.0 amps Range for IAC functions GXDMM_RANGE_100Ohm (100): 100 ohm Range for

2Wire/4Wire functions

- GXDMM_RANGE_1KOhm (1000): 1 kilo ohm Range for 2Wire/4Wire functions
- GXDMM_RANGE_10KOhm (10,000): 10 kilo ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_100KOhm (100,000): 100 kilo ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_1MOhm (1,000,000): 1 mega ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_10MOhm (10,000,000): 10 mega ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_100MOhm (100,000,000): 100 mega ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_10uA (0.00001): 10 micro amp range for Diode voltage function
- GXDMM_RANGE_100uA (0.0001): 100 micro amp range for Diode voltage function
- GXDMM_RANGE_1mA (0.001): 1 milliamp range for Diode voltage function

<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.
-----------------	--------	--

Comments

The currently selected range will depend on the currently selected function. If auto range is on, setting the range using this function will turn the auto range off.

Example

The following example gets the current range:

```
SHORT  nHandle, nStatus;  
DOUBLE dRange;  
  
GxDmmGetRange (nHandle, &dRange, &nStatus);
```

See Also

GxDmmSetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmGetReadArrayStatus

Purpose

Returns a result from measurement that has already occurred

Syntax

GxDmmGetReadArrayStatus (*nHandle*, *pbReadFinished*, *pdwCount*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pbReadFinished</i>	PBOOL	Return the measurement
<i>pdwCount</i>	PDWORD	Returns the number of samples to be taken during an Array capture
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Use this function to check if the buffered measurement has completed after calling **GxDmmSetTriggerMode** (with the **GXDMM_TRIGGER_ARRAY** constant).

Example

The following gets the array measurement status:

```
SHORT    nHandle, nStatus;
DWORD    dwCount;
BOOL     bReadFinished;
```

```
GxDmmGetReadArrayStatus (nHandle, &bReadFinished, &dwCount, &nStatus);
```

See Also

GxDmmRead, **GxDmmReadEx**, **GxDmmMeasure**, **GxDmmMeasureEx**, **GxDmmSetTriggerMode**, **GxDmmGetTriggerMode**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmGetErrorString**

GxDmmGetReadStatus

Purpose

Returns the measurement status

Syntax

GxDmmGetReadStatus (*nHandle*, *pbValid*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pbValid</i>	PBOOL	Returns the DMM's Valid Reading flag value
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Every time a measurement is made by the instrument, the *pbValid* flag will be TRUE. After the measurement data is read by the PC using an API function call, this flag is cleared and set to FALSE. This function allows the user to determine if the measurement data stored in the DMM is fresh.

Example

The following example gets the current trigger timer rate:

```
SHORT  nHandle, nStatus;
BOOL   bValid;
```

```
GxDmmGetReadStatus (nHandle, &bValid, &nStatus);
```

See Also

GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmGetRelative

Purpose

Returns the relative measurement state

Syntax

GxDmmGetRelative (*nHandle*, *pbEnable*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pbEnable</i>	PBOOL	Returns the relative measurement state
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When in relative measurement mode, the DMM will read the last measurement taken and store it in memory. Any subsequent calls to GxDmmMeasure, GxDmmMeasureEx, GxDmmRead or GxDmmReadEx will result in a measurement that is relative to the baseline value stored when the measurement mode was set to relative.

Example

The following gets the relative measurement state:

```
SHORT  nHandle, nStatus;
BOOL   bRelative;
```

```
GxDmmSetRelative (nHandle, &bRelative, &nStatus);
```

See Also

GxDmmSetRelative, GxDmmRead, GxDmmMeasure, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmGetResolution

Purpose

Returns the measurement resolution

Syntax

GxDmmGetResolution (*nHandle*, *pdwResolution*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdwResolution</i>	PDWORD	Returns the measurement in terms of digits of resolution. 0. GXDMM_RESOLUTION_3_5_DIGITS – 3 and a ½ Digits of Resolution 1. GXDMM_RESOLUTION_4_5_DIGITS – 4 and a ½ Digits of Resolution 2. GXDMM_RESOLUTION_5_5_DIGITS – 5 and a ½ Digits of Resolution 3. GXDMM_RESOLUTION_6_5_DIGITS – 6 and a ½ Digits of Resolution
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Resolution determines the number of digits returned in a normalized reading. Larger resolutions will result in slower measurement times for a given function. Setting the resolution will result in the aperture being set to a corresponding value internally when not in a AC function.

Example

The following example gets the resolution:

```
SHORT  nHandle, nStatus;
DWORD  dwRange;

GxDmmGetResolution (nHandle, &dwResolution, &nStatus);
```

See Also

GxDmmSetResolution, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmGetTriggerMode

Purpose

Returns the trigger mode

Syntax

GxDmmGetTriggerMode (*nHandle*, *pdwTriggerMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdwTriggerMode</i>	PDWORD	Returns the measurement in terms of digits of resolution. <ol style="list-style-type: none"> 0. GXDMM_TRIGGER_CONTINUOUS – The DMM will take readings as fast as possible. Measurements can be read by calling GxDmmRead or GxDmmReadEx (get the measurements from the DMM as fast as possible without buffering). 1. GXDMM_TRIGGER_TIMER – A programmable timer is used to trigger a measurement. The timer period in seconds is passed in using the <i>dwCountOrInterval</i> parameter. 2. GXDMM_TRIGGER_PXI – The PXI Bus is used to trigger a measurement 3. GXDMM_TRIGGER_SOFTWARE – A Software Trigger will trigger a measurement 4. GXDMM_TRIGGER_ARRAY – Trigger a continuous burst of measurements and store them in the onboard buffer memory. The measurements can be retrieved at a later time by calling GxDmmReadArray. The number of measurements is specified in the <i>dwCountOrInterval</i> parameter.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Trigger Reading mode determines the trigger source used to initiate a measurement.

Note that the Timer Interval and Array Count parameters can only be set, not read back. Also, the GXDMM_TRIGGER_ARRAY mode cannot be read back because it is a temporary, one-shot mode that reverts back to Software mode as soon as it has finished with its measurements. Once the trigger mode has been set to Array, its completion can be checked by calling **GxDmmGetReadArrayStatus**. Any pending measurement operation can be cancelled by calling **GxDmmAbortReading**.

Use **GxDmmSetExternalTriggers** to configure PXI and LINK triggers when setting the trigger mode to **GXDMM_TRIGGER_EXTERNAL**

Example

The following example gets the trigger mode:

```
SHORT  nHandle, nStatus;
DWORD  dwTriggerMode;

GxDmmGetTriggerMode (nHandle, &dwTriggerMode, &nStatus);
```

See Also

**GxDmmSetTriggerMode, GxDmmSetExternalTriggers, GxDmmGetExternalTriggers,
GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction,
GxDmmGetErrorString**

GxDmmInitialize

Purpose

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

Syntax

GxDmmInitialize (*nSlot*, *pnHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nSlot</i>	SHORT	GX2065 board slot number on the PXI bus.
<i>pnHandle</i>	PSHORT	Returned handle for the board. The handle is set to zero on error and <> 0 on success.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **GxDmmInitialize** function verifies whether or not the GX2065 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 DMM 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 GX2065 functions.

The last set AC line frequency is stored within the GxDmm.ini file located in Windows Common App Data folder. The input AC line frequency is loaded from the GxDmm.ini file when first initializing the DMM after power up, if the file does not exist 60Hz is used.

Example

The following example initializes two GX2065 boards at slot 1 and 2.

```
SHORT nHandle1, nHandle2, nStatus;

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

See Also

GxDmmInitializeVisa, **GxDmmReset**, **GxDmmSetLineFrequency**, **GxDmmGetErrorString**

GxDmmInitializeVisa

Purpose

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

Syntax

GxDmmInitializeVisa (*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 **GxDmmInitializeVisa** 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: "DMM1". 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.

The last set AC line frequency is stored within the GxDmm.ini file located in Windows Common App Data folder. The input AC line frequency is loaded from the GxDmm.ini file when first initializing the DMM after power up, if the file does not exist 60Hz is used.

Example

The following example initializes a GX2065 boards at PXI bus 5 and device 11.

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

See Also

GxDmmInitialize, **GxDmmReset**, **GxDmmSetLineFrequency**, **GxDmmGetErrorString**

GxDmmMeasure

Purpose

Takes a measurement and return the result

Syntax

GxDmmMeasure (*nHandle*, *pdMeasurement*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdMeasurement</i>	PDOUBLE	Return the measurement
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function generates a software trigger and forces the DMM to take a measurement. The result is returned without normalizing in base units. For example, Voltage will be returned as Volts, Current will be returned as Amps, Resistance will be returned as Ohms, Temperature will be returned as degrees Celsius, and Frequency will be returned as Hertz.

If the DMM is in continuous trigger mode when this function is called, the DMM will switch to software trigger mode before taking a measurement.

Example

The following takes a new measurement:

```
SHORT  nHandle, nStatus;
DOUBLE dMeasurement;

GxDmmMeasure (nHandle, &dMeasurement, &nStatus);
```

See Also

GxDmmMeasureEx, GxDmmRead, GxDmmReadEx, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmMeasureEx

Purpose

Takes a measurement and return the result with additional information

Syntax

GxDmmMeasureEx (*nHandle*, *pdMeasurement*, *pszMeasurement*, *pszUnits*, *pddwTime*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdMeasurement</i>	PDOUBLE	Return the measurement
<i>pszMeausrement</i>	PSTR	Returns a string containing the formatted and normalized result.
<i>pszUnits</i>	PSTR	Returns a string containing the units of the result
<i>pddwTime</i>	PDDWORD	Returns the timestamp of the result in terms of nanoseconds. The time is synced with the PC's clock.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function generates a software trigger and forces the DMM to take a measurement. The result is returned in two parameters, *pdMeasurement*, and *pszMeasurement*. The first parameter, *pdMeasurement*, returns the result without normalizing, in base units. For example, Voltage will be returned as Volts, Current will be returned as Amps, Resistance will be returned as Ohms, Temperature will be returned as degrees Celsius, and Frequency will be returned as Hertz. The second parameter, *pszMeasurement*, returns the measurement as a formatted string containing the normalized result. The normalized result uses units that reflect the magnitude of the result as well as the resolution of the measurement. For example, 0.0312356789 Amps, measured in 100mA Range, and using 6 and a 1/2 digits of resolution, will be returned as "31.2356" Milliamps. In this case, *pszUnits* will return the units as "mA".

If the DMM is in continuous trigger mode when this function is called, the DMM will switch to software trigger mode before taking a measurement.

Example

The following takes a new measurement with additional information:

```
SHORT  nHandle, nStatus;
DOUBLE dMeasurement;
CHAR   szMeasurement[256], szUnits[10];
DDWORD ddwTime;

GxDmmMeasureEx (nHandle, &dMeasurement, szMeasurement, szUnits, &ddwTime, &nStatus);
```

See Also

GxDmmMeasure, **GxDmmRead**, **GxDmmReadEx**, **GxDmmSetTriggerMode**, **GxDmmGetTriggerMode**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmGetErrorString**

GxDmmRead

Purpose

Returns a result from measurement that has already occurred

Syntax

GxDmmRead (*nHandle*, *pdMeasurement*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdMeasurement</i>	PDOUBLE	Return the measurement
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function reads back the result from a measurement that has already occurred. The result is returned without normalizing in base units. For example, Voltage will be returned as Volts, Current will be returned as Amps, Resistance will be returned as Ohms, Temperature will be returned as degrees Celsius, and Frequency will be returned as Hertz.

Note that this function does **not** initiate a new measurement. To initiate a new single reading through software, call **GxDmmMeasure**, **GxDmmMeasureEx**, or **GxDmmTrig**.

Example

The following takes a new measurement:

```
SHORT  nHandle, nStatus;
DOUBLE dMeasurement;

GxDmmRead (nHandle, &dMeasurement, &nStatus);
```

See Also

GxDmmReadEx, **GxDmmMeasure**, **GxDmmMeasureEx**, **GxDmmSetTriggerMode**, **GxDmmGetTriggerMode**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmGetErrorString**

GxDmmReadArray

Purpose

Returns a result from measurement that has already occurred

Syntax

GxDmmReadArray (*nHandle*, *padMeasurement*, *paddwTime*, *dwCount*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>padMeasurement</i>	PDOUBLE	Return an array of measurements that were taken synchronously. The array length must be at least as large as <i>dwCount</i> .
<i>paddwTime</i>	PDDWORD	Return an array of timestamps in terms of nanoseconds, corresponding with the measurements stored in the padMeasurement array. The array length must be at least as large as <i>dwCount</i> .
<i>dwCount</i>	DWORD	The number of measurements and time stamps to return. The lengths of the padMeasurement and paddwTime arrays must be at least as large as <i>dwCount</i> . Valid values are 1-8192.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function reads back buffered measurements taken as a result of a Trigger Array operation (by calling **GxDmmTriggerMode**). The result is returned without normalizing in base units. For example, Voltage will be returned as Volts, Current will be returned as Amps, Resistance will be returned as Ohms, Temperature will be returned as degrees Celsius, and Frequency will be returned as Hertz.

Note that this function does **not** initiate a new measurement. To initiate a new single reading through software, call **GxDmmMeasure**, **GxDmmMeasureEx**, or **GxDmmTrig**. To initiate a new multiple, buffered reading, call **GxDmmTrig** and pass the **GXDMM_TRIGGER_ARRAY** constant.

Call **GxDmmGetReadArrayStatus** to make sure the Trigger Array operation has completed before calling this function.

Example

The following reads back 100 measurements after initiating a Trigger Array operation:

```

SHORT  nHandle, nStatus;
DOUBLE adMeasurement[100];
DDWORD addwTime[100];
BOOL   bStatus;

GxDmmTriggerMode (nHandle, GXDMM_TRIGGER_ARRAY, 100);

GxDmmGetReadArrayStatus (nHandle, &bStatus);

while (!bStatus)
    GxDmmGetReadArrayStatus (nHandle, &bStatus);

GxDmmReadArray (nHandle, adMeasurement, addwTime, 100, &nStatus);

```


See Also

**GxDmmGetReadArrayStatus, GxDmmReadEx, GxDmmMeasure, GxDmmMeasureEx,
GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmSetRange, GxDmmGetRange,
GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString**

GxDmmReadEx

Purpose

Returns a result from measurement that has already occurred, with additional information

Syntax

GxDmmReadEx (*nHandle*, *pdMeasurement*, *p pszMeasurement*, *pszUnits*, *pddwTime*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pdMeasurement</i>	PDOUBLE	Return the measurement
<i>pszMeausrement</i>	PSTR	Returns a string containing the formatted and normalized result.
<i>pszUnits</i>	PSTR	Returns a string containing the units of the result
<i>pddwTime</i>	PDDWORD	Returns the timestamp of the result in terms of nanoseconds. The time is synced with the PC's clock.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function reads back the result from a measurement that has already occurred. The result is returned in two parameters, *pdMeasurement*, and *pszMeasurement*. The first parameter, *pdMeasurement*, returns the result without normalizing, in base units. For example, Voltage will be returned as Volts, Current will be returned as Amps, Resistance will be returned as Ohms, Temperature will be returned as degrees Celsius, and Frequency will be returned as Hertz. The second parameter, *pszMeasurement*, returns the measurement as a formatted string containing the normalized result. The normalized result uses units that reflect the magnitude of the result as well as the resolution of the measurement. For example, 0.0312356789 Amps, measured in 100mA Range, and using 6 and a 1/2 digits of resolution, will be returned as "31.2356" Milliamps. In this case, *pszUnits* will return the units as "mA".

Note that this function does **not** initiate a new measurement. To initiate a new single reading through software, call **GxDmmMeasure**, **GxDmmMeasureEx**, or **GxDmmTrig**.

Example

The following reads a measurement that has already occurred, with additional information:

```
SHORT  nHandle, nStatus;
DOUBLE dMeasurement;
CHAR   szMeasurement[256], szUnits[10];
DWORD  ddwTime;

GxDmmReadEx (nHandle, &dMeasurement, szMeasurement, szUnits, &ddwTime, &nStatus);
```

See Also

GxDmmRead, **GxDmmMeasure**, **GxDmmMeasureEx**, **GxDmmSetTriggerMode**, **GxDmmGetTriggerMode**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmGetErrorString**

GxDmmReset

Purpose

Resets the DMM to power up state

Syntax

GxDmmReset (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function initiates a reset by causing the DMM's onboard processor to reboot and reload its program from FLASH memory. The result is that all DMM settings are reset to a default, power up condition.

- Function: VDC
- Volts Range: 10V
- Current Range: 20mA
- Resistance Range: 10KOhm
- Resolution: 6 ½ Digits
- Aperture: 10 Power Line Cycles
- Auto Range: Off
- Trigger Mode: Continuous
- AC Math: RMS
- AC Min Frequency: 10 Hz
- AC Input Line Frequency: Not affected
- User Calibration

Example

The following resets the DMM:

```
SHORT  nHandle, nStatus;
```

```
GxDmmReset (nHandle, &nStatus);
```

See Also

GxDmmInitialize, GxDmmReadEx, GxDmmMeasure, GxDmmMeasureEx, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmSetLineFrequency, GxDmmGetErrorString

GxDmmRestoreFactoryCalibration

Purpose

Restores the original Factory Calibration

Syntax

GxDmmRestoreFactoryCalibration (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The original factory calibration is saved in a special area of the EEPROM in case it must be restored by the user. The factory calibration cannot be overwritten by the user.

Example

The following example restores the factory calibration:

```
SHORT  nHandle, nStatus;
```

```
GxDmmRestoreFactoryCalibration (nHandle, &nStatus);
```

See Also

GxDmmSetCalibrationSet, **GxDmmGetErrorString**

GxDmmPanel

Purpose

Opens a virtual panel used to interactively control the GX2065.

Syntax

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

Parameters

Name	Type	Comments
<i>pnHandle</i>	PSHORT	Handle to a GX2065 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 GXDMM 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;

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

See Also

GxDmmInitialize, **GxDmmGetErrorString**

GxDmmSetACClockDivider

Purpose

Sets the AC DAQ clock divider

Syntax

GxDmmSetACClockDivider (*nHandle*, *dwClockDivider*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwClockDivider</i>	DWORD	Sets the clock divider for the AC DAQ sampling clock. (1-255)
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the DMM is in any of the AC functions (VAC AC Coupled, VAC DC Coupled, IAC AC Coupled, or IAC DC Coupled), an internal DAQ is used to acquire up to 8192 samples of data at up to 3MS/s. The DAQ sampling clock (3 MHz) can be divided to achieve a slower sampling rate.

Example

The following example sets the AC DAQ Clock divider to 2 so that the effective sampling rate becomes 1.50MS/s :

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetACClockDivider (nHandle, 2, &nStatus);
```

See Also

GxDmmGetACClockDivider, **GxDmmSetACMath**, **GxDmmGetACMath**,
GxDmmSetACMinFrequency, **GxDmmGetACMinFrequency**, **GxDmmGetAutoRange**,
GxDmmSetFunction, **GxDmmGetFunction**, **GxDmmSetRange**, **GxDmmGetRange**,
GxDmmGetErrorString

GxDmmSetACMath

Purpose

Sets the AC function math type

Syntax

GxDmmSetACMath (*nHandle*, *dwMathType*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwMathType</i>	DWORD	<p>Sets the math function applied the raw measurement in any of the AC measurement functions.</p> <ol style="list-style-type: none"> 0. GXDMM_AC_MATH_RMS – Sets the AC math type to RMS (root mean square). 1. GXDMM_AC_MATH_AVG – Sets the AC Math type to Average. This math operation will take the average (arithmetic mean) of all the DAQ samples taken. 2. GXDMM_AC_MATH_PP – Sets the AC math type to Peak to Peak. 3. GXDMM_AC_MATH_PKAMP – Sets the AC math type to Peak amplitude.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the DMM is in any of the AC functions (VAC AC Coupled, VAC DC Coupled, IAC AC Coupled, or IAC DC Coupled), an internal DAQ is used to acquire up to 8192 samples of data at up to 2MS/s. A math operation (from the types listed above) is applied to the samples to generate a final measurement result. Use the **GxDmmSetACMinFrequency** function to set the lowest expected frequency.

Example

The following example sets the AC math type to RMS:

```
SHORT  nHandle, nStatus;

GxDmmSetACMath (nHandle, GXDMM_AC_MATH_RMS, &nStatus);
```

See Also

GxDmmGetACMath, **GxDmmSetACMinFrequency**, **GxDmmGetACMinFrequency**, **GxDmmGetAutoRange**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmSetACMinFrequency

Purpose

Sets the AC minimum frequency

Syntax

GxDmmSetACMinFrequency (*nHandle*, *dFrequency*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dFrequency</i>	DOUBLE	Sets the minimum frequency expected to capture by the DMM for an AC measurement.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the DMM is in any of the AC functions (VAC AC Coupled, VAC DC Coupled, IAC AC Coupled, or IAC DC Coupled), an internal DAQ is used to acquire up to 8192 samples of data at up to 2MS/s. A math operation (from the types selectable using **GxDmmSetACMath**) is applied to the samples to generate a final measurement result. The user must specify the minimum AC frequency so that the timing of the DAQ can be adjusted to capture at least 4 cycles of the waveform, with the maximum number of samples per cycle. Higher frequencies can also be captured, but the number of samples per cycle is reduced.

Example

The following example sets the AC minimum frequency to 500Hz:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetACMinFrequency (nHandle, 500, &nStatus);
```

See Also

GxDmmGetACMinFrequency, **GxDmmGetACMath**, **GxDmmGetAutoRange**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmSetAperture

Purpose

Sets the capture aperture in terms of line cycles

Syntax

GxDmmSetAperture (*nHandle*, *dAperture* , *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dAperture</i>	PBOOL	Sets the capture aperture in terms of line cycles (0.02-65)
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The aperture is expressed in terms of Line Cycles as set by the **GxDmmSetLineFrequency** function. The Aperture can be a fraction of a line cycle. For example, the aperture can be set to 0.1 Line Cycles which would represent approximately 0.00167 seconds if the Line Frequency was set to 60Hz.

Example

The following example sets the aperture to 0.1 line cycles:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetAperture (nHandle, 0.1, &nStatus);
```

See Also

GxDmmGetAperture, **GxDmmSetLineFrequency**, **GxDmmGetLineFrequency**, **GxDmmSetFunction**, **GxDmmGetFunction**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmSetAutoRange

Purpose

Sets the auto range status

Syntax

GxDmmSetAutoRange (*nHandle*, *bAutoRange* , *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>bAutoRange</i>	PBOOL	Sets the state of the Auto Range function, TRUE (1) to ON and FALSE (0) to OFF.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The auto range automatically adjusts the range with respect to the input signal. Range changes occur on overflow and when the signal is less than 8% of the current range. Turning ON auto range will result in slower reading rate.

Example

The following example activates the auto range function:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetAutoRange (nHandle, TRUE, &nStatus);
```

See Also

GxDmmGetAutoRange, **GxDmmSetFunction**, **GxDmmSetAperture**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmSetAutoZero

Purpose

Sets the auto zero status

Syntax

GxDmmSetAutoZero(*nHandle*, *dwAutoZeroMode*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwAutoZeroMode</i>	DWORD	Sets the status of the Auto Zero function. <ol style="list-style-type: none"> 0. GXDMM_AUTO_ZERO_ON: The DMM will perform an auto zero every time a measurement is taken. 1. GXDMM_AUTO_ZERO_OFF: The DMM will not auto zero when a measurement is taken. 2. GXDMM_AUTO_ZERO_NOW: The DMM will perform an auto zero calculation as soon as this command is received.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

To help maintain stability and accuracy over time and changes in temperature, the DMM periodically measures internal voltages corresponding to offsets (zero) and amplifier gains. These measurements are used in an algorithm to calculate the reading of the input signal. This process is known as auto zeroing.

When auto zero is disabled, the offset and gain measurements are not performed. This increases the measurement speed (reading rate). However, the zero and gain reference points will eventually drift resulting in less accurate readings of the input signal.

It is recommended that auto zero only be disabled for short periods of time.

An auto zero can also be forced by using the **GXDMM_AUTO_ZERO_NOW** mode.

Example

The following example performs (forces) an auto zero before taking a measurement:

```
SHORT  nHandle, nStatus;
DOUBLE dMeasurement;

GxDmmSetAutoZero (nHandle, GXDMM_AUTO_ZERO_NOW, &nStatus);
GxDmmMeasure (nHandle, &dMeasurement, &nStatus);
```

See Also

GxDmmGetAutoZero, **GxDmmSetFunction**, **GxDmmSetAperture**, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmGetErrorString**

GxDmmSetCalibrationMeasurements

Purpose

Sets the measured values for a calibration group (range function combination)

Syntax

GxDmmSetCalibrationMeasurements (*nHandle*, *dwCalGroup*, *dRefPositiveMeasurement*, *dRefNegativeMeasurement*, *dRefGroundMeasurement*, *dDmmPositiveMeasurement*, *dDmmNegativeMeasurement*, *dDmmGroundMeasurement*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwCalGroup</i>	DWORD	Chooses the range/function combination to calibrate <ol style="list-style-type: none"> 0. GXDMM_CAL_HiRes100mVDC: Volts DC at 100mV Range 1. GXDMM_CAL_HiRes1VDC: Volts DC at 1V Range 2. GXDMM_CAL_HiRes10VDC: Volts DC at 10V Range 3. GXDMM_CAL_HiRes100VDC: Volts DC at 100V Range 4. GXDMM_CAL_HiRes300VDC: Volts DC at 300V Range 5. GXDMM_CAL_HiRes20mADC: Current DC at 20mA Range 6. GXDMM_CAL_HiRes100mADC: Current DC at 100mA Range 7. GXDMM_CAL_HiRes1ADC: Current DC at 1A Range 8. GXDMM_CAL_HiRes2ADC: Current DC at 2A Range 9. GXDMM_CAL_HiRes100_Ohms2W: 2Wire Resistance at 100 Ohm Range 10. GXDMM_CAL_HiRes1k_Ohms2W: 2Wire Resistance at 1K Ohm Range 11. GXDMM_CAL_HiRes10k_Ohms2W: 2 Wire Resistance at 10K Ohm Range 12. GXDMM_CAL_HiRes100k_Ohms2W: 2Wire Resistance at 100K Ohm Range 13. GXDMM_CAL_HiRes1M_Ohms2W: 2Wire Resistance at 1M Ohm Range 14. GXDMM_CAL_HiRes10M_Ohms2W: 2Wire Resistance at 10M Ohm Range 15. GXDMM_CAL_HiRes100M_Ohms2W: 2Wire Resistance at 100M Ohm Range 16. GXDMM_CAL_HiRes100_Ohms4W: 4Wire Resistance at 100 Ohm Range 17. GXDMM_CAL_HiRes1k_Ohms4W: 4Wire Resistance at 1K Ohm Range 18. GXDMM_CAL_HiRes10k_Ohms4W: 4Wire Resistance at

10K Ohm Range

19. GXDMM_CAL_HiRes100k_Ohms4W: 4Wire Resistance at 100K Ohm Range
20. GXDMM_CAL_HiRes1M_Ohms4W: 4Wire Resistance at 1M Ohm Range
21. GXDMM_CAL_HiRes10M_Ohms4W: 4Wire Resistance at 10M Ohm Range
22. GXDMM_CAL_HiRes100M_Ohms4W: 4Wire Resistance at 100M Ohm Range
23. GXDMM_CAL_HiRes100mVSHiDC: Not Used
24. GXDMM_CAL_HiRes1VSHiDC: Not Used
25. GXDMM_CAL_HiRes10VSHiDC: Not Used
26. GXDMM_CAL_HiResVCalOpen: 2Wire and 4Wire Open Circuit for 10M Ohm and 100M Ohm Ranges
27. GXDMM_CAL_HiRes_R_Ref:
28. GXDMM_CAL_Daq100mVDC: Volts AC, DC Coupled 100mV Range
29. GXDMM_CAL_Daq1VDC: Volts AC, DC Coupled 1V Range
30. GXDMM_CAL_Daq10VDC: Volts AC, DC Coupled 10V Range
31. GXDMM_CAL_Daq100VDC: Volts AC, DC Coupled 100V Range
32. GXDMM_CAL_Daq300VDC: Volts AC, DC Coupled 300V Range
33. GXDMM_CAL_Daq20mADC: Current AC, DC Coupled 20mA Range
34. GXDMM_CAL_Daq100mADC: Current AC, DC Coupled 100mA Range
35. GXDMM_CAL_Daq1ADC: Current AC, DC Coupled 1A Range
36. GXDMM_CAL_Daq2ADC: Current AC, DC Coupled 2A Range
37. GXDMM_CAL_Daq100mVAC: Volts AC, AC Coupled 100mV Range
38. GXDMM_CAL_Daq1VAC: Volts AC, AC Coupled 1V Range
39. GXDMM_CAL_Daq10VAC: Volts AC, AC Coupled 10V Range
40. GXDMM_CAL_Daq100VAC: Volts AC, AC Coupled 100V Range
41. GXDMM_CAL_Daq300VAC: Volts AC, AC Coupled 300V Range

42. GXDMM_CAL_Daq20mAAC: Current AC, AC Coupled 20mA Range
43. GXDMM_CAL_Daq100mAAC: Current AC, AC Coupled 100mA Range
44. GXDMM_CAL_Daq1AAC: Current AC, AC Coupled 1A Range
45. GXDMM_CAL_Daq2AAC: Current AC, AC Coupled 2A Range

<i>dRefPositiveMeasurement</i>	DOUBLE	The positive reference DMM measurement
<i>dRefNegativeMeasurement</i>	DOUBLE	The negative reference DMM measurement
<i>dRefGroundMeasurement</i>	DOUBLE	The ground reference DMM measurement
<i>dPositiveMeasurement</i>	DOUBLE	The positive GX2065 DMM measurement
<i>dNegativeMeasurement</i>	DOUBLE	The negative GX2065 DMM measurement
<i>dGroundMeasurement</i>	DOUBLE	The ground GX2065 DMM measurement
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX2065 can be calibrated in system, in field, by the end user.

Call this function to store measured values for each combination of range and function. The firmware will internally calculate the resultant gains and offset. Call **GxDmmWriteEEPROM** to write the measured values to the onboard EEPROM.

The Reference measurements, *dRefPositiveMeasurement*, *dRefNegativeMeasurement*, and *dRefGroundMeasurement*, are used to input the values being generated by the Calibrator. Normally these values should be exactly the same as the Calibrator setting. For example, when calibrating VDC 10 Volt Range, *dRefPositiveMeasurement* would be +10.0, *dRefNegativeMeasurement* would be -10.0, and *dRefGroundMeasurement* would be 0.

When using a discrete source and reference DMM instead of a Calibrator, *dRefPositiveMeasurement*, *dRefNegativeMeasurement*, and *dRefGroundMeasurement* would store measured values from the reference DMM.

Note that this function only affects the User calibration. If the User calibration is done incorrectly and the results have been finalized (by calling **GxDmmWriteCalEEPROM**), the contents can be restored by calling **GxDmmRestoreFactoryCalibration**.

Example

The following example calibrates the Volts DC function for the 10 Volt Range and stores it in the User Calibration:

```
SHORT  nHandle, nStatus;
DOUBLE dPositiveMeasurement, dNegativeMeasurement, dGroundMeasurement;

SetCalibration(VDC, 10.0, "Volts"); //Calibrator API, Set Calibrator to VDC +10.0 Volts
GxDmmMeasure(nHandle, &dPositiveMeasurement, &nStatus);
SetCalibration(VDC, -10.0, "Volts"); //Calibrator API, Set Calibrator to VDC -10.0 Volts
GxDmmMeasure(nHandle, &dNegativeMeasurement, &nStatus);
SetCalibration(VDC, 0.0, "Volts"); //Calibrator API, Set Calibrator to VDC 0.0 Volts
GxDmmMeasure(nHandle, &dGroundMeasurement, &nStatus);
GxDmmSetCalibrationMeasurements (nHandle, GXDMM_CAL_HiRes10VDC, 10.0, -10.0, 0.0,
dPositiveMeasurement, dNegativeMeasurement, dGroundMeasurement, &nStatus);
```

See Also

**GxDmmSetCalibrationSet, GxDmmGetCalibrationSet, GxDmmRestoreFactoryCalibration,
GxDmmWriteCalEEPROM, GxDmmGetErrorString**

GxDmmSetCalibrationSet

Purpose

Sets the currently applied calibration set

Syntax

GxDmmSetCalibrationSet (*nHandle*, *lCalSet*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>lCalSet</i>	LONG	<p>Sets the currently applied calibration set.</p> <p>The following are acceptable values:</p> <ol style="list-style-type: none"> 0. GXDMM_CALSET_USER_CALIBRATION: User calibration 1. GXDMM_CALSET_FACTORY_CALIBRATION: Factory calibrated values 2. GXDMM_CALSET_FOR_CALIBRATION: Special calibration set used only when calibrating the DMM
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX2065 has the ability to load different calibration sets dynamically. The EEPROM stores two full calibration sets, User and Factory. The Factory calibration is done at Marvin Test Solutions and cannot be overwritten by the end user. The User Calibration stores a copy of the Factory Calibration but can be overwritten by the User if a Calibration License is active.

The Factory calibration can be restored by calling **GxDmmRestoreFactoryCalibration**. This function will copy the contents of the Factory calibration to the User calibration.

Note that the User Calibration is always loaded by default on initial power up and reset.

Example

The following example sets the currently loaded calibration set to User Calibration:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetCalibrationSet (nHandle, GXDMM_CALSET_USER_CALIBRATION, &nStatus);
```

See Also

GxDmmGetCalibrationSet, **GxDmmSetCalibrationMeasurements**,
GxDmmWriteCalibrationEEPROM, **GxDmmRestoreFactoryCalibration**, **GetErrorString**

GxDmmSetExternalTriggers

Purpose

Sets the external trigger configuration

Syntax

GxDmmSetExternalTriggers (*nHandle*, *dwTriggerDirections*, *dwTriggerEdges*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwTriggerDirections</i>	DWORD	Sets the direction of all external triggers
<i>dwTriggerEdges</i>	DWORD	Sets the trigger edge sensitivity for all external triggers
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The *dwTriggerDirections* is a 20 bit long field that determines the directionality of each external trigger. Each bit in the field represents the enable bit for a particular trigger and direction. A '1' indicates enabled and a '0' indicates disabled. By default all bits are set to '0'. The fields are described in the following table:

Bit Number	Description
0	PXI Line 0 Input Enable
1	PXI Line 1 Input Enable
2	PXI Line 2 Input Enable
3	PXI Line 3 Input Enable
4	PXI Line 4 Input Enable
5	PXI Line 5 Input Enable
6	PXI Line 6 Input Enable
7	PXI Line 7 Input Enable
8	PXI Line 0 Output Enable
9	PXI Line 1 Output Enable
10	PXI Line 2 Output Enable
11	PXI Line 3 Output Enable
12	PXI Line 4 Output Enable
13	PXI Line 5 Output Enable
14	PXI Line 6 Output Enable
15	PXI Line 7 Output Enable
16	STAR PXI Input Enable
17	STAR PXI Output Enable
18	LINK Bus Input Enable
19	LINK Bus Output Enable

Table 6-4: dwTriggerDirections Bit Field

The *dwTriggerEdges* is a 11 bit long field that determines the edge sensitivity of each external trigger. Each bit in the field represents the edge selected to use for triggering. A '1' indicates rising edge and a '0' indicates falling edge. By default all bits are set to '0'. The fields are described in the following table:

Bit Number	Description
0	PXI Line 0 Edge Selection
1	PXI Line 1 Edge Selection
2	PXI Line 2 Edge Selection
3	PXI Line 3 Edge Selection
4	PXI Line 4 Edge Selection
5	PXI Line 5 Edge Selection
6	PXI Line 6 Edge Selection
7	PXI Line 7 Edge Selection
8	STAR PXI Edge Selection
9	LINK Bus Edge Selection

Table 6-5: dwTriggerEdges Bit Fields

Example

The following example sets the trigger direction of PXI line 0 and PXI Line 1 and all edge sensitivities to Rising Edge:

```
SHORT nHandle, nStatus;
```

```
GxDmmSetExternalTriggers (nHandle, 0x3, 0x3FF, &nStatus);
```

See Also

GxDmmGetExternalTriggers, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmReadArray, GxDmmGetReadArrayStatus, GxDmmAbortReading, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmSetFunction

Purpose

Sets the measurement function

Syntax

GxDmmSetFunction (*nHandle*, *dwFunction*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwFunction</i>	DWORD	Sets the DMM function: <ol style="list-style-type: none"> 0. GXDMM_FUNCTION_VDC – Voltage DC Mode 1. GXDMM_FUNCIONT_IDC – Current DC Mode 2. GXDMM_FUNCTION_VAC_AC_CPL – Voltage AC, AC Coupled Mode 3. GXDMM_FUNCTION_IAC_AC_CPL – Current AC, AC Coupled Mode 4. GXDMM_FUNCTION_VAC_DC_CPL – Voltage AC, DC Coupled Mode 5. GXDMM_FUNCTION_IAC_DC_CPL – Current AC, DC Coupled Mode 6. GXDMM_FUNCTION_2WIRE_OHM – 2Wire Resistance Mode 7. GXDMM_FUNCTION_4WIRE_OHM – 4 Wire Resistance Mode 8. GXDMM_FUNCTION_CONTINUITY – High Resolution Continuity Mode 9. GXDMM_FUNCTION_FREQUENCY – Frequency Counter Mode 10. GXDMM_FUNCTION_DIODE – Diode Voltage Mode 11. GXDMM_FUNCTION_TEMPERATURE – External Temperature Mode 12. GXDMM_FUNCTION_BOARD_TEMPERATURE – Internal, Board Temperature Mode
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX2065 can be set to several different function modes. When the function is changed, the min/max accumulator is cleared and any ongoing measurement is aborted. If the current reading trigger is not set to software trigger, changing the function will result in the reading trigger being set to software trigger, briefly, and then back to its previous setting.

Example

The following example sets the DMM's function to Volts DC:

```
SHORT  nHandle, nStatus;

GxDmmSetFunction (nHandle, GXDMM_FUNCTION_VDC, &nStatus);
```

See Also

GxDmmGetFunction, **GxDmmSetRange**, **GxDmmGetRange**, **GxDmmSetTriggerMode**, **GxDmmGetTriggerMode**, **GxDmmGetErrorString**

GxDmmSetLineFrequency

Purpose

Sets the AC Power Line Frequency setting

Syntax

GxDmmSetLineFrequency (*nHandle*, *dwLineFrequency*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwLineFrequency</i>	DWORD	Sets the AC Power Line Frequency selection <ul style="list-style-type: none"> GXDMM_LINE_FREQUENCY_50HZ (50): 50Hz AC Line Frequency GXDMM_LINE_FREQUENCY_60HZ (60): 60Hz AC Line Frequency GXDMM_LINE_FREQUENCY_400HZ (400): 400Hz AC Line Frequency
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows the user to select the line frequency of the Mains Source powering the chassis. Setting this parameter correctly reduces the amount of noise in all measurements.

The last set AC line frequency is stored within the GxDmm.ini file located in Windows Common App Data folder. The input AC line frequency is loaded from the GxDmm.ini file when first initializing the DMM after power up, if the file does not exist 60Hz is used. Calling this function will cause the value to be written to the file.

Example

The following example sets the line frequency to 60Hz:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetLineFrequency(nHandle, GXDMM_LINE_FREQUENCY_60HZ, &nStatus);
```

See Also

GxDmmInitialize, GxDmmReset, GxDmmSetLineFrequency, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmSetRange

Purpose

Sets the currently selected function's range

Syntax

GxDmmSetRange (*nHandle*, *dRange*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dRange</i>	DOUBLE	<p>Sets the currently selected function's range.</p> <ul style="list-style-type: none"> GXDMM_RANGE_100mV (0.100): 100 millivolts Range for VDC functions GXDMM_RANGE_1V (1): 1 volt Range for VDC functions GXDMM_RANGE_10V (10): 10 volts Range for VDC functions GXDMM_RANGE_100V (100): 100 volts Range for VDC functions GXDMM_RANGE_300V (300): 300 volts Range for VDC functions GXDMM_RANGE_50mVrms (0.050): 50 millivolts Range for VAC functions GXDMM_RANGE_500mVrms (0.50): 500 millivolts Range for VAC functions GXDMM_RANGE_5Vrms (5): 5 volts Range for VAC functions GXDMM_RANGE_50Vrms (50): 50 volts Range for VAC functions GXDMM_RANGE_300Vrms (300): 300 volts Range for VAC functions GXDMM_RANGE_20mA (0.020): 20 milliamps Range for IDC functions GXDMM_RANGE_100mA (0.100): 100 milliamps Range for IDC functions GXDMM_RANGE_1A (1): 1 amp Range for IDC functions GXDMM_RANGE_2A (2): 2 amps Range for IDC functions GXDMM_RANGE_10mArms (0.010): 10 milliamps Range for IAC functions GXDMM_RANGE_50mArms (0.050): 50 milliamps Range for IAC functions GXDMM_RANGE_500mArms (0.50): 500 milliamps Range for IAC functions GXDMM_RANGE_1Arms (1.0): 1.0 amps Range for IAC functions GXDMM_RANGE_100Ohm (100): 100 ohm Range for

2Wire/4Wire functions

- GXDMM_RANGE_1KOhm (1000): 1 kilo ohm Range for 2Wire/4Wire functions
- GXDMM_RANGE_10KOhm (10,000): 10 kilo ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_100KOhm (100,000): 100 kilo ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_1MOhm (1,000,000): 1 mega ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_10MOhm (10,000,000): 10 mega ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_100MOhm (100,000,000): 100 mega ohms Range for 2Wire/4Wire functions
- GXDMM_RANGE_10uA (0.00001): 10 micro amp range for Diode voltage function
- GXDMM_RANGE_100uA (0.0001): 100 micro amp range for Diode voltage function
- GXDMM_RANGE_1mA (0.001): 1 milliamp range for Diode voltage function

pnStatus

PSHORT

Returned status: 0 on success, negative number on failure.

Comments

The currently selected range will depend on the currently selected function. If auto range is on, setting the range using this function will turn the auto range off.

Example

The following example sets the range to 10 Volts:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetRange (nHandle, 10.0, &nStatus);
```

See Also

GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmSetRelative

Purpose

Sets the relative measurement state

Syntax

GxDmmSetRelative (*nHandle*, *bEnable*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>bEnable</i>	BOOL	Sets the relative measurement state
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When in relative measurement mode, the DMM will read the last measurement taken and store it in memory. Any subsequent calls to GxDmmMeasure, GxDmmMeasureEx, GxDmmRead or GxDmmReadEx will result in a measurement that is relative to the baseline value stored when the measurement mode was set to relative.

Example

The following sets the relative measurement state to True:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetRelative (nHandle, TRUE, &nStatus);
```

See Also

GxDmmGetRelative, GxDmmRead, GxDmmMeasure, GxDmmSetTriggerMode, GxDmmGetTriggerMode, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmSetResolution

Purpose

Sets the measurement resolution

Syntax

GxDmmSetResolution (*nHandle*, *pdwResolution*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwResolution</i>	DWORD	Sets the measurement in terms of digits of resolution. 0. GXDMM_RESOLUTION_3_5_DIGITS – 3 and a ½ Digits of Resolution 1. GXDMM_RESOLUTION_4_5_DIGITS – 4 and a ½ Digits of Resolution 2. GXDMM_RESOLUTION_5_5_DIGITS – 5 and a ½ Digits of Resolution 3. GXDMM_RESOLUTION_6_5_DIGITS – 6 and a ½ Digits of Resolution
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Resolution determines the number of digits returned in a normalized reading. Larger resolutions will result in slower measurement times for a given function. Setting the resolution will result in the aperture being set to a corresponding value internally when not in an AC function.

Example

The following example sets the resolution to 6 ½ digits:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetResolution (nHandle, GXDMM_RESOLUTION_6_5_DIGITS, &nStatus);
```

See Also

GxDmmGetResolution, GxDmmSetRange, GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString

GxDmmSetTriggerMode

Purpose

Sets the trigger mode

Syntax

GxDmmSetTriggerMode (*nHandle*, *dwTriggerMode*, *dwCountOrInterval*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>dwTriggerMode</i>	DWORD	Returns the measurement in terms of digits of resolution. 0. GXDMM_TRIGGER_CONTINUOUS – The DMM will take readings as fast as possible. Measurements can be read by calling GxDmmRead or GxDmmReadEx (get the measurements from the DMM as fast as possible without buffering). 1. GXDMM_TRIGGER_TIMER – A programmable timer is used to trigger a measurement. The timer period in seconds is passed in using the <i>dwCountOrInterval</i> parameter. 2. GXDMM_TRIGGER_PXI – The PXI Bus is used to trigger a measurement 3. GXDMM_TRIGGER_SOFTWARE – A Software Trigger will trigger a measurement 4. GXDMM_TRIGGER_ARRAY – Trigger a continuous burst of measurements and store them in the onboard buffer memory. The measurements can be retrieved at a later time by calling GxDmmReadArray . The number of measurements is specified in the <i>dwCountOrInterval</i> parameter.
<i>dwCountOrInterval</i>	DWORD	Specify the number of measurements to take when in Trigger Array mode or specify the Timer Rate in seconds if in Trigger Timer mode. If another Trigger Mode is selected, this parameter is ignored.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Trigger Reading mode determines the trigger source used to initiate a measurement.

When using the Trigger Array trigger mode, call **GxDmmReadArray** to retrieve the measurements after they have been completed. **GxDmmGetReadArrayStatus** can be used to check on the status of measurement array operation (if it has completed or not). Any pending measurement operation can be cancelled by calling **GxDmmAbortReading**.

Use **GxDmmSetExternalTriggers** to configure PXI and LINK triggers when setting the trigger mode to GXDMM_TRIGGER_EXTERNAL.

Example

The following example sets the trigger mode to continuous:

```
SHORT  nHandle, nStatus;
```

```
GxDmmSetTriggerMode (nHandle, GXDMM_TRIGGER_CONTINUOUS, 0, &nStatus);
```

See Also

**GxDmmGetTriggerMode, GxDmmSetExternalTriggers, GxDmmGetExtenralTriggers,
GxDmmReadArray, GxDmmGetReadArrayStatus, GxDmmAbortReading, GxDmmSetRange,
GxDmmGetRange, GxDmmSetFunction, GxDmmGetFunction, GxDmmGetErrorString**

GxDmmTrig

Purpose

Generates a software trigger

Syntax

GxDmmTrig (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Generating a software trigger forces the DMM to take a measurement when the trigger reading mode is set to Software.

Example

The following generates a software trigger:

```
SHORT  nHandle, nStatus;
```

```
GxDmmTrig (nHandle, &nStatus);
```

See Also

GxDmmRead, **GxDmmMeasure**, **GxDmmSetTriggerMode**, **GxDmmGetTriggerMode**,
GxDmmSetRange, **GxDmmGetRange**, **GxDmmSetFunction**, **GxDmmGetFunction**,
GxDmmGetErrorString

GxDmmWriteCalibrationEEPROM

Purpose

Writes the calibration values to the onboard EEPROM

Syntax

GxDmmWriteCalibrationEEPROM (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
<i>nHandle</i>	SHORT	Handle for a GX2065 board.
<i>pnStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This is the final function to be called during calibration. It finalizes the calibration by writing the calibrated values to the onboard EEPROM. This function can only modify the contents of the User calibration.

Example

The following writes calibration values to the EEPROM:

```
SHORT  nHandle, nStatus;
```

```
GxDmmWriteCalibrationEEPROM (nHandle, &nStatus);
```

See Also

GxDmmSetCalibrationMeasurements, **GxDmmSetCalibrationSet**, **GxDmmGetCalibrationSet**, **GxDmmRestoreFactoryCalibration**, **GxDmmGetErrorString**

Appendix A - Specifications

This section provides the GX2065 DMM specifications.

General Specifications	
Input	Hi, Lo, Hi Sense, Lo Sense; floating and isolated from ground External trigger
Input Connectors	(4) Banana, 6 pin DIN
Format	PXI, 3U single slot, hybrid slot compatible
DMM and Digitizer Measurement System Features	
Resolution	22 bits (DMM), 16 bits (digitizer / DAQ)
DMM Sampling Rate (DCV, DCI & R measurements)	Selectable PLC rate, 0.002 to 10; PLC can be set to 50, 60 or 400 Hz
DMM reading rate (DCV, DCI & 2WR (<10 Mohm))	5 readings/sec @ 6.5 digits, 60Hz (10 PLC), Auto Zero Off 50 readings/sec @ 6.5 digits, 60 Hz (1 PLC) , Auto Zero Off 500 readings/sec @ 5.5 digits, 60 Hz, (0.1 PLC) , Auto Zero Off, 1K Into Buffer 2500 readings/sec @ 4.5 digits, 60 Hz, (0.01 PLC) , Auto Zero Off, 1K Into Buffer 3500 readings/sec @ 3.5 digits, 60 Hz, (0.002 PLC) , Auto Zero Off, 1K Into Buffer
Digitizer (DAQ) Clock Rate	Programmable to 3 MHz Range: $(3\text{MS/s}) / N$, $N = 1$ to 2^{16} Accuracy: 100 PPM
DAQ Measurement Functions	AC and DC voltage and current measurements
Digitizer Memory	8192 samples
DMM Memory	1K samples
DMM Measurement Characteristics	
DCV Input range	100 nV to 300 volts
ACV Input range	3 μV to 425 V (peak); 300 VRMS, 10 Hz to 300 KHz, AC or DC coupled
Crest Factor (ACV)	No limitation as long as maximum input signal is below the maximum range value
DCV / ACV Input Impedance	>10 G ohms (0.1, 1, and 10 volt DC ranges), 400 pF shunt capacitance 10 M ohms for other AC/DC ranges, 400 pF shunt capacitance

Maximum Input (Volt – Hertz)	8 x 10e7 volt x Hz common mode input 8 x 10e7 volt x Hz (across Hi or Lo input relative to earth ground)
Input Isolation	CATII 300V
Input Overload Protection	250V for current input, 300V CATII for all other inputs
Noise Rejection	DCV: 90 dB, NMRR; 140 dB CMRR 15 readings/s, 1 PLC, 6.5 digit, 10 V range ACV: 70dB, CMRR
DCI Input range	10 nA to 2 amps
ACI Input range	3 uA to 2 amps (peak)
Crest Factor (AC current)	No limitation as long as maximum input signal is below the peak range value.
Burden Voltage (max)	< 0.2V, 20 mA range < 0.1 V, 100 mA range < 0.5 V, 1 A range < 1.0 V, 2A range
AC/DC Input Current Protection	2A, 250V, fast blow, sand filled, 1.5kA breaking
Resistance Range	0.1 m ohms to 100 M ohms
Resistance Measurement Configuration	Selectable, 2-wire or 4-wire
Internal Temperature Measuring Accuracy	+/- 2 C
Timebase Accuracy	100 ppm
Triggering	
Trigger Source	Function: Start measurement Source: PXI bus, software, continuous, external input (DIN connector), timer
Trigger Output Modes	Functions: Start of measurement, end of measurement Trigger can be routed to the PXI bus or the DIN connector
Trigger Input Voltage Range	3.3V CMOS, 5V Tolerant
Minimum Trigger Pulse Width	50ns for PXI bus, 250us for external DIN input
Trigger Input Impedance	4.75k ohms

Trigger Input Edge	Selectable, positive or negative			
AC Measurement Performance				
Reading Rate	Signal Bandwidth			
1 S/s	4 Hz to 4 kHz			
5 S/s	20 Hz to 20 kHz (Default)			
375S/s	300 Hz to 300 kHz			
Voltage, Current, Resistance, and Frequency Measurement Accuracy				
All accuracy specifications are relative to the calibration standard.				
DC Voltage				
Range	Resolution	Accuracy 24 Hours 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 90 Days 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 1 Year 23°C ±5° (% of Reading)+ (% of FS)
100mV	100nV	0.0030 + 0.0040	0.0040 + 0.0045	0.0045 + 0.0045
1V	1 μV	0.0030 + 0.0007	0.0040 + 0.0008	0.0045 + 0.0008
10V	10 μV	0.0010 + 0.0004	0.0025 + 0.0005	0.0030 + 0.0005
100V	100 μV	0.0030 + 0.0006	0.0050 + 0.0009	0.0060 + 0.0009
300V	1 mV	0.0030 + 0.0020	0.0045 + 0.0030	0.0060 + 0.0030
Notes: DC measurements @ 10 PLC or 1 PLC with digital filtering Accuracy of measurement is % of reading + % of Range				
DC Current				
Range	Resolution	Accuracy 24 Hours 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 90 Days 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 1 Year 23°C ±5° (% of Reading)+ (% of FS)
20 mA	10 nA	0.0060 + 0.0030	0.0300 + 0.0080	0.0500 + 0.0080
100 mA	100 nA	0.0100 + 0.0300	0.0300 + 0.0800	0.0500 + 0.0800
1A	1 uA	0.0200 + 0.0030	0.0500 + 0.0080	0.0800 + 0.0080

2A	10 uA		0.1000 + 0.0035	0.1200 + 0.0060	0.1200 + 0.0060
AC Volts (RMS), AC Coupled, DAQ Mode					
Range (RMS)	Resolution	Frequency	Accuracy 24 Hours 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 90 Days 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 1 Year 23°C ±5° (% of Reading)+ (% of FS)
50mV	2 uV	3 Hz - 10Hz	0.5 + 0.28	0.5 + 0.28	0.5 + 0.28
		10 Hz – 20 KHz	0.2 + 0.28	0.2 + 0.28	0.2 + 0.28
		20 KHz – 50 KHz	0.26 + 0.3	0.26 + 0.3	0.26 + 0.3
		50 KHz – 100 KHz	0.75 + 0.33	0.75 + 0.33	0.75 + 0.33
		100 KHz – 300 KHz	4.15 + 0.75	4.15 + 0.75	4.15 + 0.75
0.5 V 5 V 50 V 300 V	20 uV 200 uV 2 mV 30 mV	3 Hz - 10Hz	0.35 + 0.03	0.35 + 0.03	0.35 + 0.03
		10 Hz – 20 KHz	0.05 + 0.03	0.05 + 0.03	0.06 + 0.03
		20 KHz – 50 KHz	0.11 + 0.05	0.11 + 0.05	0.12 + 0.05
		50 KHz – 100 KHz	0.60 + 0.08	0.60 + 0.08	0.60 + 0.08
		100 KHz – 300 KHz	4.0 + 0.5	4.0 + 0.5	4.0 + 0.5

AC Current (RMS), AC Coupled, DAQ Mode					
Range(RMS)	Resolution	Frequency	Accuracy 24 Hours 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 90 Days 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 1 Year 23°C ±5° (% of Reading)+ (% of FS)
0.5 amp	40 uA	3 Hz - 10 Hz	0.30 + 0.04	0.30 + 0.04	0.30 + 0.04
		10 Hz – 3 KHz	0.10 + 0.04	0.10 + 0.04	0.10 + 0.04
		3 KHz - 5 KHz	0.14 + 0.04	0.14 + 0.04	0.14 + 0.04
1.0 amp	80 uA	3 Hz - 10 Hz	0.35 + 0.09	0.35 + 0.09	0.35 + 0.09
		10 Hz – 3 KHz	0.15 + 0.09	0.15 + 0.09	0.15 + 0.09
		3 KHz - 5 KHz	0.18 + 0.09	0.18 + 0.09	0.18 + 0.09
Resistance					
Range (ohms)	Open Circuit Voltage & Test Current	Resolution	Accuracy 24 Hours 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 90 Days 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 1 Year 23°C ±5° (% of Reading)+ (% of FS)
100	6.9V, 1 mA	100 μ ohms	0.0020 + 0.0060	0.0080 + 0.0060	0.0100 + 0.0060
1000	6.9V, 1 mA	1 m ohm	0.0025 + 0.0006	0.0085+ 0.0020	0.0105 + 0.0006
10K	6.9V, 100 uA	10 m ohm	0.0020 + 0.0006	0.0080 + 0.0020	0.0100 + 0.0006
100K	12.8V, 10 uA	100 m ohms	0.0030 + 0.0006	0.0090 + 0.0010	0.0110 + 0.0010
1 M	12.8V, 1 uA	1 ohm	0.0020 + 0.0006	0.0020 + 0.0010	0.0100 + 0.0010

10 M	7V, 0.7 uA //10 M ohm	10 ohms	0.0150 + 0.0006	0.0200 + 0.0010	0.0400 + 0.0010
100 M	7V, 0.7 uA // 10 M ohm	100 ohms	0.0800 + 0.0030	0.2000 + 0.0030	0.2000 + 0.0030

DC Voltage Measurement, DAQ Mode

Range	Resolution	Accuracy 24 Hours 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 90 Days 23°C ±5° (% of Reading)+ (% of FS)	Accuracy 1 Year 23°C ±5° (% of Reading)+ (% of FS)
100 mV	4 uV	0.06 + 0.06	0.06 + 0.06	0.06 + 0.06
1 V	40 uV	0.06 + 0.03	0.06 + 0.03	0.06 + 0.03
10 V	400 uV	0.06 + 0.03	0.06 + 0.03	0.06 + 0.03
100 V	4 mV	0.06 + 0.03	0.06 + 0.03	0.06 + 0.03
300 V	40 mV	0.06 + 0.10	0.06 + 0.10	0.06 + 0.10

Diode Voltage Measurement

Range	Voltage Range	Current Source Accuracy	Accuracy (Nominal) 24 Hours 23°C ±5° (% of Reading)+ (% of FS)	Accuracy (Nominal) 90 Days 23°C ±5° (% of Reading)+ (% of FS)	Accuracy (Nominal) 1 Year 23°C ±5° (% of Reading)+ (% of FS)
10 uA	0-12 V	5%	0.0010 + 0.0004	0.0025 + 0.0005	0.0030 + 0.0005
100 uA	0-6.5 V	5%	0.0010 + 0.0004	0.0025 + 0.0005	0.0030 + 0.0005
1 mA	0-6.5 V	5%	0.0010 + 0.0004	0.0025 + 0.0005	0.0030 + 0.0005

Frequency Measurement

Frequency Range	1 Hz to 500 KHz
Input Voltage*	20 mV to 300 V
Resolution (offset ppm)	0.33 (1 second gate time) 3.33 (100 mSec gate time) 33.3 (10 mSec gate time)
Accuracy	100 ppm of reading + offset ppm

* Input amplitude must be at least 20% of FS and input amplitude must not exceed specified volt – hertz product.

Environmental and Physical Specifications	
Safety	Complies with IEC 61010-1, CAT II 300V, pollution degree 2
EMC	Complies with EN61326-1
Calibration	Calibration is performed at the factory using NIST traceable instrumentation. All calibration constants are stored on-board in non-volatile EEROM. Calibration can be performed by any calibration laboratory with the appropriate equipment.
Temperature Range	Operating: 0 to + 50°C Extended operating range: -20°C to +70°C Storage: -20°C to +70°C storage
Relative Humidity	Operating: 80% at 40°C Storage to 95% at 40°C
Power (max)	+5VDC, 2.3 A +3.3 VDC, 255 mA +12 VDC, 16 mA -12 VDC, 25 mA
Connectors	(4) Banana jacks: Hi: Voltage, 2W ohms Lo: Voltage, Current, 2W ohms Sense Hi: Current, 4W ohms Sense Lo: 4W ohms 6-pin DIN: Trigger in, Trigger out, Trigger Gnd

Note: Specifications are subject to change.

Appendix B – Signametrics DMM Function Comparisons

This section provides a function reference comparison between the Marvin Test Solutions GXDMM Signametrics compatibility driver and the Signametrics SMX2040/SMX2060 series native driver (SM204032.dll/SM2060.dll).

The following Signametrics functions are supported by the GXDMM compatibility driver:

- DMMBurstBuffRead(int nDmm, int iSettle, int iMeasurements)
- DMMCalibrate(int nDmm)
- DMMClearMinMax(int nDmm)
- DMMDelay(double dDelay)
- DMMDisArmTrigger(int nDmm)
- DMMErrorCovert(SHORT nDmmError)
- DMMErrString(int nError, LPSTR lpcBuf, int nBufLen)
- DMMFrequencyStr(int nDmm, LPSTR lpszReading)
- DMMGetAperture(int nDmm, double *lpdRate)
- DMMGetCalDate(int nDm, LPSTR lpszDate)
- DMMGetFuncRange(int nDmm)
- DMMGetFunction(int nDmm)
- DMMGetGrdVer(int nDmm)
- DMMGetHwVer(int nDmm)
- DMMGetID(int nDmm)
- DMMGetMax(int nDmm, double FAR *lpfRes)
- DMMGetMaxStr(int nDmm, LPSTR lpszReading)
- DMMGetMin(int nDmm, double FAR *lpfRes)
- DMMGetMinStr(int nDmm, LPSTR lpszReading)
- DMMGetRange(int nDmm)
- DMMGetRate(int nDmm, double FAR *lpdRate)
- DMMGetType(int nDmm)
- DMMGetVer(int nDmm, double FAR *lpdVer)
- DMMInit(int nDmm, LPCSTR lpszCal)
- DMMIsAutoRange(int nDmm)
- DMMIsInitialized(int nDmm)
- DMMIsRelative(int nDmm)
- DMMPeriodStr(int nDmm, LPSTR lpszReading)
- DMMPolledRead(int nDmm, double FAR *lpfRes)

- DMMPolledReadCmd(int nDmm)
- DMMPolledReadStr(int nDmm, LPSTR lpszReading)
- DMMRead(int nDMM, double *lpfValue)
- DMMReadBuffer(int nDmm, double FAR *lpdRes)
- DMMReadFrequency(int nDmm, double FAR *lpdFreq)
- DMMReadMeasurement(int nDMM, double FAR *value)
- DMMReadNorm(int nDmm, double FAR *lpdRes)
- DMMReadPeakToPeak(int nDmm, double * dPtp)
- DMMReadPeriod(int nDmm, double FAR *lpdRes)
- DMMReadStr(int nDmm, LPSTR lpszReading)
- DMMReady(int nDmm)
- DMMSetAperture(int nDMM, int nAperture)
- DMMSetAutoRange(int nDmm, BOOL bAuto)
- DMMSetFuncRange(int nDmm, int nFnRange)
- DMMSetFunction(int nDmm, int nFunction)
- DMMSetPLC(int nDmm, int LineFrequency, int N)
- DMMSetRange(int nDmm, int nRange)
- DMMSetRate(int nDmm, int nRate)
- DMMSetRelative(int nDmm, BOOL bRelative)
- DMMTrigger(int nDmm, int nSamples)

The following functions exist in the compatibility driver. These functions return one of the following status codes:

- DMM_OKAY (0) – This status is returned if the function is not required by GXDMM (and the functionality exists).
- FUNC_INACTIVE (-31) – This status is returned if GXDMM does not support this feature or function.
- ClearBuffer – returns DMM_OKAY
- DMMArmAnalogTrigger – returns DMM_OKAY
- DMMArmTrigger – returns DMM_OKAY
- DMMBurstRead – returns DMM_OKAY
- DMMCleanRelay – returns DMM_OKAY
- DMMClosePCI – returns DMM_OKAY
- DMMDelayedTrigger – returns FUNC_INACTIVE
- DMMDisableTrimDAC – returns FUNC_INACTIVE
- DMMDutyCycleStr – returns FUNC_INACTIVE
- DMMGetACCapsR – returns FUNC_INACTIVE
- DMMGetBufferSize – returns FUNC_INACTIVE
- DMMGetBusInfo – returns DMM_OKAY
- DMMGetCJTemp – returns FUNC_INACTIVE

- DMMGetdB – returns FUNC_INACTIVE
- DMMGetdBStr – returns FUNC_INACTIVE
- DMMGetDeviatStr – returns FUNC_INACTIVE
- DMMGetDeviation – returns FUNC_INACTIVE
- DMMGetDiffMnMxStr – returns FUNC_INACTIVE
- DMMGetHwOption – returns DMM_OKAY
- DMMGetManDate – returns DMM_OKAY
- DMMGetReadInterval – returns FUNC_INACTIVE
- DMMGetSourceFreq – returns FUNC_INACTIVE
- DMMGetSourceMode – returns FUNC_INACTIVE
- DMMGetTCType – returns FUNC_INACTIVE
- DMMGetTriggerInfo – returns FUNC_INACTIVE
- DMMLoadCalFile – returns FUNC_INACTIVE
- DMMLongTrigger – returns FUNC_INACTIVE
- DMMLongTrigRead – returns FUNC_INACTIVE
- DMMOpenCalACCaps – returns FUNC_INACTIVE
- DMMOpenPCI – returns DMM_OKAY
- DMMOpenTerminalCal – returns DMM_OKAY
- DMMReadBufferStr – returns DMM_OKAY
- DMMReadCJTemp – returns FUNC_INACTIVE
- DMMReadCrestFactor – returns FUNC_INACTIVE
- DMMReadDutyCycle – returns FUNC_INACTIVE
- DMMReadHiSense – returns FUNC_INACTIVE
- DMMReadInductorQ – returns FUNC_INACTIVE
- DMMReadLoSense – returns FUNC_INACTIVE
- DMMReadMedian – returns FUNC_INACTIVE
- DMMReadTestV – returns FUNC_INACTIVE
- DMMReadTotalizer – returns FUNC_INACTIVE
- DMMReadWidth – returns FUNC_INACTIVE
- DMMSetACCapsDelay – returns FUNC_INACTIVE
- DMMSetACCapsLevel – returns FUNC_INACTIVE
- DMMSetACVSource – returns FUNC_INACTIVE
- DMMSetBuffTrigRead – returns DMM_OKAY
- DMMSetCapsAveSamp – returns FUNC_INACTIVE
- DMMSetCJTemp – returns FUNC_INACTIVE
- DMMSetCompThreshold – returns FUNC_INACTIVE
- DMMSetCounterRng – returns FUNC_INACTIVE
- DMMSetDCISource – returns FUNC_INACTIVE
- DMMSetDCVSource – returns FUNC_INACTIVE
- DMMSetDutyCycle – returns FUNC_INACTIVE
- DMMSetExternalShunt – returns FUNC_INACTIVE
- DMMSetFastRMS – returns FUNC_INACTIVE
- DMMSetInductFreq – returns FUNC_INACTIVE
- DMMSetOffsetOhms – returns FUNC_INACTIVE
- DMMSetPulseGen – returns FUNC_INACTIVE

- DMMSetPXITrigger – returns FUNC_INACTIVE
- DMMSetReadInterval – returns FUNC_INACTIVE
- DMMSetReference – returns FUNC_INACTIVE
- DMMSetResistance – returns FUNC_INACTIVE
- DMMSetRTD – returns FUNC_INACTIVE
- DMMSetSensorParams – returns FUNC_INACTIVE
- DMMSetSourceMode – returns FUNC_INACTIVE
- DMMSetSync – returns FUNC_INACTIVE
- DMMSetTCType – returns FUNC_INACTIVE
- DMMSetTempUnits – DMM_OKAY
- DMMSetTrigRead – DMM_OKAY
- DMMSetTrigPolarity – returns FUNC_INACTIVE
- DMMSetTrimDAC – returns FUNC_INACTIVE
- DMMStartTotalizer – returns FUNC_INACTIVE
- DMMStopTotalizer – returns FUNC_INACTIVE
- DMMTerminate – returns TRUE
- DMMTriggerBurst – returns FUNC_INACTIVE
- DMMUnlockCounter – returns FUNC_INACTIVE
- DMMWidthStr – returns FUNC_INACTIVE
- SetTotalizerDivider – returns FUNC_INACTIVE

Index

2

2Wire 13, 41, 49, 71, 75, 100, 101, 107, 110

4

4Wire 13, 41, 75, 100, 101, 110

A

AC Measurement 15, 121

Applications 3

Architecture 1

ATEasy 21, 27, 28

Auto Zero 6, 15, 52, 53, 60, 99, 119

B

Board Description 1, 3, 7

Board Handle 31

Board Installation 23

Borland 27

Borland-Delphi 27

C

C/C++ 27

CalEasy 41

Calibration 41

Calibration Licensing 42

Calibration Procedure 44

Connectors 25

Conventions 1

Copyright i

Corrupt files 20

Current measurements 8

D

Delphi 27

Diode 14, 71, 75, 107, 110, 124

Directories 21

Disclaimer i

Distributing the Driver 32

DMM API Comparisons 127

DMM Connections 18

Driver

Directory 21

Files 21

Driver Version 31

E

Error Handling 31

ESD 23

Examples 32

F

Features 3

Freq 13

Frequency . 3, 4, 10, 13, 15, 52, 53, 58, 68, 71, 72, 85, 86, 87, 88, 90, 91, 97, 107, 108, 121, 122, 123, 124

Function Reference 51

G

Getting Started 17

GxCntInitialize 12

GxCntInitializeVisa 12

GXDMM 1, 20

Driver-Description 27

Header-file 27

GXDMM Driver 27

GXDMM driver functions 30

GXDMM Functions 43, 52

GXDMM Software 21

GXDMM.bas 27

GXDMM.dll 28

GXDMM.EXE 20

GXDMM.h 27

GXDMM.lib 27

GXDMM.llb 28

GXDMM.pas 27

GXDMM.vb 27

GXDMM64.DLL 27

GXDMM64.lib 27

GxDmmAbortReading.....	52, 54	GxDmmSetAperture.....	53, 97
GXDMMBC.lib.....	27	GxDmmSetAutoRange.....	53, 98
GxDmmGetACClockDivider.....	55	GxDmmSetAutoZero.....	53, 99
GxDmmGetACMath.....	52, 56	GxDmmSetCalibrationMeasurements.....	43, 44, 53, 100, 102
GxDmmGetACMinFrequency.....	52, 57	GxDmmSetCalibrationSet.....	43, 53, 104
GxDmmGetAperture.....	52, 58	GxDmmSetCalMeasurements.....	100
GxDmmGetAutoRange.....	52, 59	GxDmmSetExternalTriggers.....	105
GxDmmGetAutoZero.....	52, 60	GxDmmSetFunction.....	43, 53, 107
GxDmmGetBoardSummary.....	52, 61	GxDmmSetLineFrequency.....	53, 108
GxDmmGetCalibrationInfo.....	52, 62	GxDmmSetRange.....	43, 53, 109
GxDmmGetCalibrationSet.....	52, 64	GxDmmSetRelative.....	112
GxDmmGetDriverSummary.....	31, 52, 65	GxDmmSetResolution.....	53, 113
GxDmmGetErrorString.....	31, 51, 52, 66, 68	GxDmmSetTriggerMode.....	52, 53, 114
GxDmmGetExternalTriggers.....	69	GxDmmTrig.....	53, 116
GxDmmGetFunction.....	52, 71	GxDmmWriteCalEeprom.....	53
GxDmmGetLineFrequency.....	52, 72	GxDmmWriteCalibrationEEPROM.....	43, 49, 117
GxDmmGetMinMax.....	52, 73	H	
GxDmmGetRange.....	52, 74	Handle.....	23, 24, 30, 31
GxDmmGetReadArrayStatus.....	52, 77	Hardware Requirements.....	41
GxDmmGetReadStatus.....	52, 78	High energy circuit safety precautions.....	4
GxDmmGetRelative.....	79	HW.....	20, 21, 25, 27, 32
GxDmmGetResolution.....	52, 80	I	
GxDmmGetTriggerMode.....	52, 81	IAC.....	13, 33, 36, 38, 46, 47, 55, 56, 57, 71, 74, 94, 95, 96, 107, 109
GxDmmInitialize.....	22, 30, 31, 43, 51, 52, 83	IDC.....	13, 33, 36, 38, 46, 47, 71, 74, 107, 109
GxDmmInitializeVisa.....	22, 30, 31, 51, 52, 84	If You Need Help.....	i
GxDmmMeasure.....	43, 52, 85	Installation.....	17, 20
GxDmmMeasureEx.....	53, 86	Installation Directories.....	21
GxDmmPanel.....	52, 93	Installation:.....	23, 25
GxDmmRead.....	53, 87	Interfaces.....	17
GxDmmReadArray.....	52, 88	Introduction.....	1, 3
GxDmmReadEx.....	53, 90	K	
GxDmmReset.....	52, 91	Keys:RATE.....	6
GxDmmRestoreFactoryCalibration.....	53	L	
GxDmmRestoreFactoryCalibration.....	92	LabView.....	28
GxDmmSetACClockDivider.....	52, 53, 94	LabView/Real Time.....	28
GxDmmSetACMath.....	43, 53, 95	Line cycle synchronization see LSYNC.....	6
GxDmmSetACMinFrequency.....	53, 96		

Line Frequency	15	Reset	31
Linux	28	Resistance measurements	9
Listing	34	Resolution... 3, 6, 14, 33, 39, 67, 71, 80, 91, 107, 113, 119, 121, 122, 123, 124	
LSYNC	6	S	
M		Safety and Handling	i
Manual Organization	1	Safety precautions:High energy circuits	5
Manual Scope	1	Sample	33, 34
N		Sample Programs	33
<i>nHandle</i>	28	Setup	20, 21
Noise:Lowest settings	7	Setup Maintenance	20
O		Setup-and-Installation	17
OnError	28	Signametrics	28, 127
Overview	3	Slot	12, 17, 23, 25, 30, 33
P		SM204x	28
Panel	15, 16, 20, 30, 93	SM2060	28
Pascal	27	SMX2040	127
PCI	21	SMX2060	127
Performance considerations	5	Specifications	1, 119
Plug & Play	25	System	
programming	27	Directory	21
Programming		System Requirements	17
Borland-Delphi	27	T	
Error-Handling	31	Temp	13
Visual	27	Trademarks	ii
Programming Examples	32	Trigger	14, 19, 67, 81, 88, 91, 114, 120, 121, 125
Programming the GX2065	27	V	
PXI	3, 17, 22, 23, 24, 25, 30	VAC 13, 33, 36, 38, 44, 45, 55, 56, 57, 71, 74, 94, 95, 96, 107, 109	
PXI System	22	VDC	13, 33, 36, 37, 38, 41, 44, 45, 49, 71, 74, 91, 102, 107, 109, 125
PXI/PCI Explorer	12, 22, 30, 31, 83, 84	Virtual Panel	12, 15, 16, 20, 93
PXIeSYS.INI	12	About Page	16
PXISYS.INI	12	Initialize Dialog	12
R		VISA	12, 21, 22, 30, 31, 52, 83, 84
Range 13, 14, 33, 38, 44, 45, 46, 47, 48, 49, 52, 53, 59, 67, 74, 75, 86, 90, 91, 98, 100, 101, 102, 109, 110, 119, 120, 121, 122, 123, 124, 125		Visual	27
RATE key	6	Visual Basic	27
README.TXT	21	Visual Basic .NET	27
Removing a Board	25		

Visual C++	27	Warm-up.....	5
Voltage measurements.....	8	Warranty	i
W			
Warm up Time.....	41		