

# ***CalEasy***

## **Calibration and Verification Executive**

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

Last updated January 31, 2017



## Safety and Handling

---

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

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

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

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

## Warranty

---

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

## If You Need Help

---

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

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

## Disclaimer

---

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

## Copyright

---

Copyright © 2011-2017 by Marvin Test Solutions, Inc. All rights reserved. No part of this document can be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Marvin Test Solutions.

## Trademarks

---

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

All other trademarks are the property of their respective owners.

# Table of Contents

Safety and Handling.....	i
Warranty .....	i
If You Need Help.....	i
Disclaimer .....	i
Copyright .....	i
Trademarks .....	ii
<b>Chapter 1 - Introduction .....</b>	<b>1</b>
Manual Scope and Organization .....	1
Manual Scope.....	1
Manual Organization.....	1
Conventions Used in this Manual .....	1
<b>Chapter 2 - Overview .....</b>	<b>3</b>
Introduction.....	3
Features.....	3
Architecture .....	4
Instruments, Standards and Capabilities .....	5
Instrument Interchangeability .....	5
Calibration and Verification Programs .....	5
Calibration Run Mode .....	5
Record Keeping and Certificate/Log Numbering .....	6
<b>Chapter 3 - Setup and Installation .....</b>	<b>7</b>
Program Requirements .....	7
Installing CalEasy .....	7
Setup Maintenance Program .....	8
Installation Directories.....	8
CalEasy Files Description.....	9
CalEasy Files .....	9
Instrument Standards DLL Files .....	9
COUNTER Standard: .....	9
DMM Standard: .....	9
MATRIX Standard:.....	11
PWRMETER Standard: .....	12
PS Standard:.....	12
REF Standard: .....	12
SCOPE Standard: .....	12

<b>Chapter 4 - Using CalEasy .....</b>	<b>15</b>
Overview.....	15
Starting CalEasy .....	15
License Installation .....	15
CalEasy Main Window .....	17
CalEasy Menu and Toolbar Commands .....	18
Working with CalEasy .....	19
Test Log .....	20
Calibration Certificates .....	22
Device Information Window .....	23
Standards / Instruments Window .....	25
Setting CalEasy Options .....	26
General Options .....	26
Log Options .....	27
Certificate Options .....	28
<b>Chapter 5 - Device Calibration .....</b>	<b>29</b>
Overview.....	29
GX1110 Calibration.....	29
Overview.....	29
Calibration Harness Connections .....	30
Connections instructions .....	30
Calibration Procedure .....	30
Approximate Test Time .....	30
GX1120 Calibration.....	31
Overview.....	31
Calibration Harness Connections .....	32
Connections instructions .....	32
Calibration Procedure .....	32
Approximate Test Time .....	32
GX1649 Calibration.....	33
Overview.....	33
Calibration Harness Connections .....	34
Calibration Procedure .....	34
Approximate Test Time .....	34
GX1838 Calibration.....	35
Overview.....	35
Connections instructions .....	35

Calibration Procedure .....	35
Approximate Test Time .....	36
GX2065 Series Calibration .....	37
Overview .....	37
Calibration Harness Connections .....	37
Connections instructions .....	38
Calibration Procedure .....	39
Approximate Test Time .....	39
GX3348/GX3364 Calibration .....	40
Overview .....	40
Calibration Harness Connections .....	41
Calibration Procedure .....	41
Approximate Test Time .....	41
GX3788 Calibration .....	42
Overview .....	42
Calibration Harness Connections .....	43
Calibration Procedure .....	43
Approximate Test Time .....	43
GX5055 Calibration .....	44
Overview .....	44
Calibration Harness Connections .....	45
Connections instructions .....	45
Calibration Procedure .....	46
Approximate Test Time .....	46
GX5295 Calibration .....	47
Overview .....	47
Calibration Harness Connections .....	48
Connections instructions .....	48
Calibration Procedure .....	48
Approximate Test Time .....	48
GX5960 Series Calibration .....	49
Overview .....	49
Calibration Harness Connections .....	50
Connections instructions .....	50
Calibration Procedure .....	51
Approximate Test Time .....	51
<b>Chapter 6 - Standard Drivers .....</b>	<b>53</b>

Overview.....	53
Writing a Standard Driver.....	53
Function Reference Summary.....	57
COUNTER Standard Functions .....	57
DMM (Digital Multimeter) Standard Functions .....	57
MATRIX (Switch Matrix) Standard Functions.....	57
PS (Power Supply) Standard Functions .....	57
PWRMETER (Power Meter) Standard Functions .....	58
SCOPE (Oscilloscope) Standard Functions .....	58
REF (Calibrator/Reference Source) Standard Functions .....	59
COUNTER Standard Driver .....	60
CalCounterGetError.....	60
CalCounterGetInfo.....	61
CalCounterInitialize.....	62
CalCounterMeasure .....	63
CalCounterReset .....	64
CalCounterSetAcquisitionMode .....	65
CalCounterSetDigitalFilter .....	66
CalCounterSetFunctionPulseWidth .....	67
CalCounterSetGateTime .....	68
CalCounterSetImpedance .....	69
CalCounterSetSlope.....	70
CalCounterSetTriggerLevel.....	71
DMM Standard Driver .....	72
CalDmmGetError.....	72
CalDmmGetFunction.....	73
CalDmmGetInfo .....	74
CalDmmInitialize.....	75
CalDmmMeasure .....	76
CalDmmReset.....	77
CalDmmSetFunction .....	78
CalDmmSetRange .....	79
CalDmmSetReadingRate .....	80
CalDmmSetResolution .....	81
MATRIX Standard Driver .....	82
CaMatrixClose .....	82
CalMatrixGetError.....	83



CalMatrixGetInfo.....	84
CalMatrixInitialize.....	85
CaMatrixOpen .....	86
CaMatrixReset .....	87
PS Standard Driver .....	88
CalPsGetChannelState .....	88
CalPsGetCurrent .....	89
CalPsGetCurrentLimit .....	90
CalPsGetError.....	91
CalPsGetInfo.....	92
CalPsGetVoltage.....	93
CalPsInitialize .....	94
CaPsReset .....	95
CalPsSetChannelState.....	96
CalPsSetCurrentLimit .....	97
CalPsSetVoltage .....	98
PWRMETER Standard Driver.....	99
CalPwrMeterClose.....	99
CalPwrMeterGetError.....	100
CalPwrMeterGetInfo .....	101
CalPwrMeterInitialize.....	102
CalPwrMeterMeasure .....	103
CaPwrMeterReset .....	104
CalPwrMeterSetupMeasurement .....	105
CalPwrMeterZero .....	106
SCOPE Standard Driver .....	107
CalScopeGetError.....	107
CalScopeGetInfo.....	108
CalScopeGetInputImpedance .....	109
CalScopeInitialize .....	110
CalScopeMeasure .....	111
CalScopeReset .....	112
CalScopeSetupAcquisition .....	113
CalScopeSetupAuto .....	114
CalScopeSetupChannel.....	115
CalScopeSetupChannelState.....	116
CalScopeSetupHorizontal .....	117

CalScopeSetupInputImpedance .....	118
REF Standard Driver .....	119
CalRefGetCalibrationStatus.....	119
CalRefGetError.....	120
CalRefGetInfo.....	121
CalRefGetRequiredCalibration.....	122
CalRefInitialize.....	123
CaRefReset .....	124
CaRefSet2Wire .....	125
CaRefSet4Wire .....	126
CaRefSetACCurrent .....	127
CaRefSetACVolts .....	128
CaRefSetDCCurrent .....	129
CaRefSetDCVolts .....	130
CaRefSetState .....	131
<b>Index .....</b>	<b>133</b>

# Chapter 1 - Introduction

## Manual Scope and Organization

### Manual Scope





This manual provides all the information necessary for installation, operation, and maintenance of CalEasy - a software package designed to calibrate and verify Marvin Test Solutions' devices and instruments. This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and a general knowledge of modular test equipment.

### Manual Organization

The CalEasy manual is organized in the following manner:

Chapter	Content
Chapter 1 – Introduction	Introduces CalEasy manual and shows warning conventions used in the manual.
Chapter 2 – Overview	Provides an overview of CalEasy's features and architecture.
Chapter 3 –Setup and Installation	Provides instructions about how to install CalEasy, system requirements and provides an overview of CalEasy's folders and files.
Chapter 4 –CalEasy	Provides instructions about how to use the CalEasy software to verify and calibrate instruments.
Chapter 5 –Device Calibration	Provides a list of Marvin Test Solutions' devices/instruments that can be calibrated using CalEasy. Each required standards and capabilities such as hardware and cabling are described in order to support the calibration/verification procedure for the supported Marvin Test Solutions instruments.
Chapter 6 – Standard Drivers	Provides a list of the standard drivers used by the calibration programs and provides a function reference to each standard. The chapter also explains how to create a standard DLL to control an instrument that is not supported by CalEasy.

## Conventions Used in this Manual

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

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

## Chapter 2 - Overview

### Introduction

---

CalEasy is a calibration executive software package that is designed to verify and calibrate Marvin Test Solutions instruments. The software is intended for use by users that prefer to perform in-house calibration and verification of their Marvin Test Solutions' instruments. By calibrating or verifying instrument functionality in-house, the time and cost associated with returning an instrument to Marvin Test Solutions for calibration service can be eliminated or reduced. CalEasy is also used internally by Marvin Test Solutions' production to perform initial calibration and to recalibrate the instrument when received for service. This ensures that user and factory based calibration performance is similar.

### Features

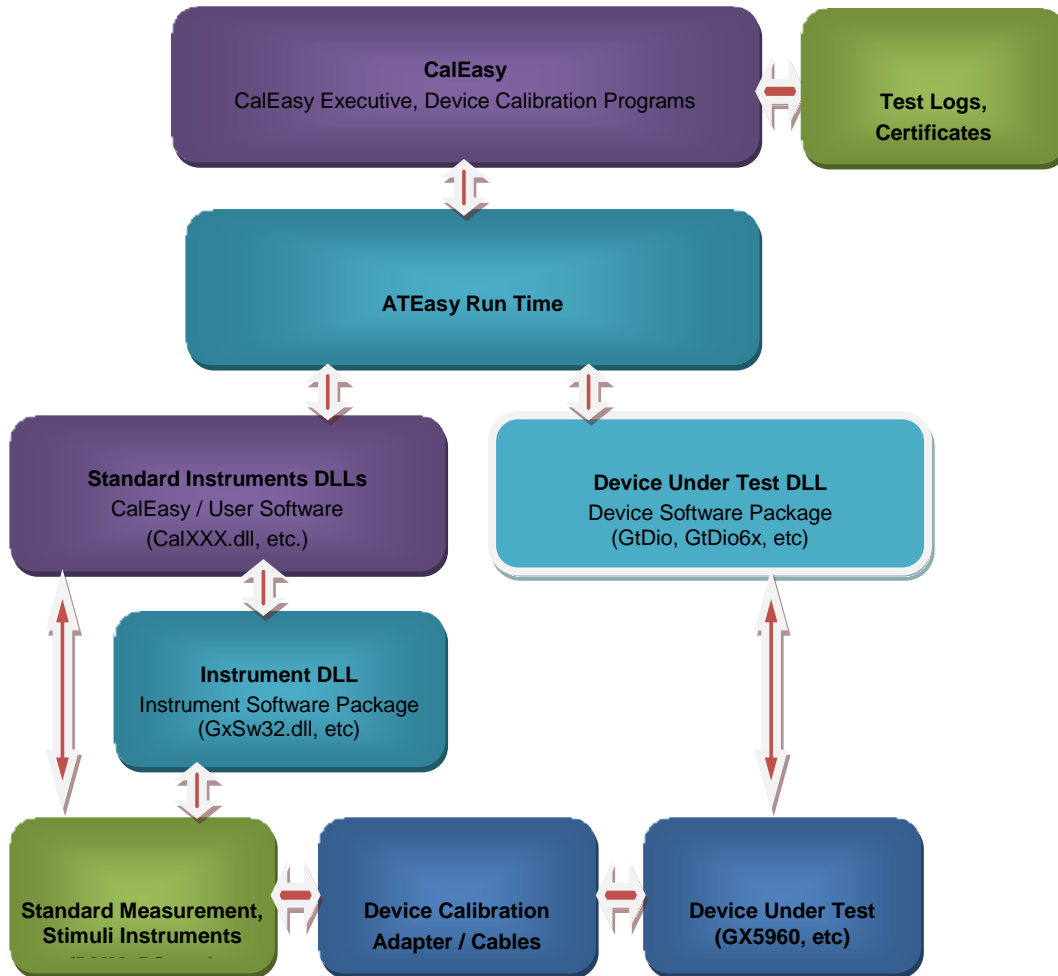
---

CalEasy has the following capabilities and features:

- Verifies instrument calibration and operation
- Calibrates the instrument
- Supports automated or manual operation mode when using a non-programmable instrument standard
- Supports configuration / control of the standards instruments used by the calibration software
- Supports interchangeability of the standards instruments using a modular architecture
- Provides a way to use any user standards-compliant instrument (requires some programming)
- Provides the option to immediately abort verification or calibration upon a single failure
- Generates a calibration certificate
- Generates detailed calibration or verification log
- Supports calibration certificate templates for certificate customization
- Supports Auto Saving and Printing of calibration logs and certificates to user defined folders

## Architecture

The CalEasy architecture is detailed in the following block diagram:



**Figure 2-1: CalEasy Block Diagram**

CalEasy uses DLLs to communicate with the device that is being verified / calibrated and the standards instruments which are used by the device calibration program. Every device under test (DUT), that is a Marvin Test Solutions instrument and that is being calibrated, is connected to the standards instruments via calibration adapters or cables. These cables or adapters are specifically designed for the calibration program that is specific to the DUT. See the section in this manual that describes your device for specific information about required cables and adapters.

## Instruments, Standards and Capabilities

---

CalEasy uses **instruments** to perform calibration and calibration verification. CalEasy's instrument drivers are called **Standards** and can be interchanged to accommodate different instruments from different manufacturers. To provide interchangeability, CalEasy uses a DLL with a predefined instrument interface (functions/API) for each driver type. Each calibration program requires certain functions and capabilities to perform calibration and for each function, the user must supply a DLL that supports the required functional interface.

For example: the GX5960 series of digital I/O PXI instrument requires three standards / instruments:

- DMM - digital multi-meter
- MATRIX - switch matrix
- PS - power supply Note: If calibrating the GX5960 in a GX7005, GX7005 or GX7017 chassis this power supply is not required.

When configuring the calibration setup, the user must have a standards DLLs for each one of the standards / instruments.

In addition to the standards, the user must ensure that the selected instruments will meet all the required capabilities as specified by the specific device that is being calibrated. For example: In order to calibrate the GX5960 the DMM must have DC, AC, 2 Wire measurement and must be accurate to 6 ½ digits, etc.

## Instrument Interchangeability

---

CalEasy supports exchanging the standards instrument drivers supplied with CalEasy with a user supplied instrument driver(s). This provides a way to use your own instrument(s) for calibration instead of the drivers supplied with CalEasy. If you choose to use your own instrument, you must create a DLL driver for your standards instrument and configure CalEasy to use your driver. New drivers can be written using the ATEasy software development environment or with any other development environment including C/C++, Visual Basic, LabVIEW or any other language that supports the creation of DLLs. CalEasy is supplied with samples for several instrument drivers created in ATEasy. See the README.txt for a list of supplied drivers and examples.

## Calibration and Verification Programs

---

CalEasy is supplied with a program for each MTS instrument that can be calibrated using CalEasy. Each program is divided into two sections or Tasks: Verification and Calibration. Each Task is comprised of several tests that perform the calibration / verification of your instrument. When running the program, the user is required to select the DUT's or instrument's address, the standards drivers used for calibration /verification and the calibration run mode.

## Calibration Run Mode

---

CalEasy supports the following calibration run modes:

- **Verify** – in this mode no calibration is performed. The instrument is only verified to its original specification and the result is printed to the test log.
- **Verify and if Passed, Update Calibration Date**—in this mode verification is performed and if the verification task passes then the instrument calibration date is updated. The non-volatile memory is modified with a new calibration date and a calibration certification is generated. If the verification task fails nothing further is done and the program exits.
- **Verify and Calibrate If Required** – in this mode verification is performed and if required (the verification task fails) the instrument is calibrated. If calibration is successful, the instrument's non-volatile memory is modified with the new calibration values and a calibration certificate is generated. If verification is OK, the

instrument's non-volatile memory is modified with the new calibration date (no new calibration values are written to the instrument non-volatile memory) and a certificate is generated.

- **Calibrate Always** – in this mode no initial verification is performed. The calibration task is performed and the certificate is generated upon successful calibration.

## Record Keeping and Certificate/Log Numbering

---

CalEasy performs record keeping for calibration or verification. Every time the calibration or verification program is executed, a test log is created and saved (option). A calibration certificate is always saved upon successful calibration, or if **Verify and if passed, Update Calibration Date** or **Verify and Calibrate If Required** mode is selected and the verification is successful.

Test logs and calibration certificates are saved to their designated folder(s) as set by the program options. The user can set options that defines when to automatically save these files. By default, the files are saved as a HTML file format. Test Logs can also be saved in a text file format.

File names for the test logs and the certificate are saved according to the following file name specifications:

*Program~Serial Number~Year-Month-Day-Hour-Minute~Log/Certificate.htm*

Where:

- *Program* – calibration program name
- *Serial Number* – the calibrated device serial number
- *Year* – 4 digits' year (2011, etc.)
- *Month* – two numeric digits for the month (01-12)
- *Day* – two numeric digits for the day in the month (01-31)
- *Hour* – two numeric digits for the hour of the day (00-23)
- *Minute* – two numeric digits for the minute of the hour

The time is the calibration time as stored in the card's non-volatile memory or the time the test log is created.

For example, the following files were saved after running the GX5055 program to calibrate serial number 123456 on November 21, 2011 at 5:29 PM:

GX5055~123456~2011-11-21-17-29~Certificate.htm

GX5055~123456~2011-11-21-17-29~Log.htm



## Chapter 3 - Setup and Installation

### Program Requirements

---

For proper operation of CalEasy your computer must meet the following minimum requirements

- Windows 32/64-bit desktop or PXI master chassis (with CPU boards), Windows XP (SP3)-Windows 10 are supported (check the readme file if you have a newer operating system)
- ATEasy run-time (v9 build 152c or newer) installed, download from <http://www.marvintest.com/ATEasy/>
- CalEasy, installed, download from <http://www.marvintest.com/Product.aspx?model=CalEasy>
- CalEasy License – obtained from Marvin Test Solutions support <http://www.marvintest.com/magic/>, provide your product serial number and the computer ID as displayed when you first launch CalEasy
- Additional instruments are required to perform the calibration, for more information check the section that describes the instrument you are calibrating

Detailed installation procedures for the required software are provided in the next sections.

### Installing CalEasy

---

To install the CalEasy software follow the instructions described below:

1. **Installing ATEasy run-time files.** This step is required if ATEasy or ATEasy run-time is not installed. Insert the Marvin Test Solutions CD-ROM and locate the ATEasy setup program. If your computer's Auto Run is configured, when inserting the CD, a browser will show several options. Select the **Run AExplorer.exe** option, and then locate the ATEasy product page and run the setup (ATEasy9-xxx.xe). If Auto Run is not configured, you can open the Windows explorer and locate the AExplorer.exe and run it. You can also download the latest ATEasy setup from Marvin Test Solutions' web site (<http://www.marvintest.com/ATEasy/>). After starting the setup, you can choose between Full installation and Run-Time only. Use the Full installation option if you intend to look or view the drivers' sources provided as standards DLLs. ATEasy full installation is provided with a free for 30 days evaluation license for development and a free run-time. CalEasy requires ATEasy v9, build 152c or newer. Installing the latest version of ATEasy will always work. Check the CalEasy README.txt file for the required ATEasy run-time version for CalEasy.

---

**Note:** When installing ATEasy under Windows 2000 or newer, you may be required to restart the setup after logging-in as a user with local 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 ATEasy to access resources on your machine.

---

2. Install the software package required to operate your instrument and standards. If you are calibrating the GX5961/4 you will need to install the **GtDio6x** software, for GX5055 install the **GtDio** software, etc.
3. Insert the Marvin Test Solutions CD-ROM and locate the CalEasy product page and run **CalEasy.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 latest version from Marvin Test Solutions' web site (<http://www.marvintest.com/Downloads/>) and search for CalEasy software package.
4. The first setup screen to appear is the Welcome screen. Click **Next** to continue.
5. Enter the folder where CalEasy 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\CalEasy\.

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.

## Setup Maintenance Program

---

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

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

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

## Installation Directories

---

The CalEasy files are installed in the default directory C:\Program Files [(x86)]\Marvin Test Solutions\CalEasy. You can change the default CalEasy directory to one of your choosing at the time of installation.

During the installation, CalEasy Setup creates and copies files to the following directories:

Name	Purpose / Contents
...\Marvin Test Solutions\CalEasy	The CalEasy main folder. Contains CalEasy program, documentation, certificate template
...\Marvin Test Solutions\CalEasy\Log	Default test log and certificate files folder
...\Marvin Test Solutions\CalEasy\Standards	Standard Instrument DLLs and sources
...\ATEasy	ATEasy run-time or development (if installed full version) files
...\Marvin Test Solutions\HW	HW device driver. Provides 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
...\Windows	CalEasy.ini – settings and defaults used by CalEasy
...\Windows\System32	Windows System directory. Contains shared components such as ATEasy run-time, device and instruments DLLs and some upgraded system components, such as the HTML help viewer, etc.

## CalEasy Files Description

---

The Setup program installs the CalEasy software files. The following is a brief description of each installed file:

### CalEasy Files

- CalEasy.exe – Windows application used to calibrate and verify Marvin Test Solutions’ instruments.
- CalEasyCertificateTemplate.htm – Default template used to generate the calibration certificate
- CalEasy.pdf – This user manual in acrobat file format
- ReadMe.txt – Last minute information, version information, change list and other technical notes

### Instrument Standards DLL Files

Each instrument standards DLL used by CalEasy such as PS, DMM, MATRIX, etc. is stored in a subfolder of the **CalEasy\Standards** folder. Each instrument that corresponds to a standard has its own sub folder. For example, Keithley 200x Digital Multimeter is stored in the “CalEasy\Standards\DMM\Keithley 200x folder”; this folder has the associated DLL driver CalDmmKi200x.dll and optionally the DLL sources (usually written in ATEasy). The generic interface DLL and header file resides in the DMM folder CalDmm.h and CalDmm.dll. Each standards driver requires the installation of the manufacturer’s driver and needs to be configured such that it will recognize the instrument. For GPIB/VXI instruments you may also need to setup the ATEasy GPIB/VXI interface driver to match with the current GPIB/VXI interface board you have (see ATEasy Getting Started manual). The following sections describe the standards files installed and the additional software required in order to use these standards instruments.

The following list details each supported instrument driver with its name, manufacturer, form factor (GPIB, 3U PXI, etc.), supported model and primary specifications:

### COUNTER Standard:

- CalCounter.drv - ATEasy COUNTER driver template
- CalCounter.h – C/C++ header for COUNTER driver template

### Ag531xx – Keysight Technologies/Agilent/HP, GPIB/COM, Ag53131A/53132A, 12 digits

- CalCounterAg531xx.dll – DLL driver
- CalCounterAg531xx.prj – ATEasy project
- CalCounterAg531xx.sys – ATEasy system

### GX2200 – Marvin Test Solutions, 3U PXI, GTX2210/GTX2220/GTX2230, 225MHz/1.3GHz/2.0 GHz:

- CalCounterGX2200.dll – DLL driver
- CalCounterGX2200.prj – ATEasy project
- CalCounterGX2200.sys – ATEasy system
- Requires Marvin Test Solutions GXCNT software

### DMM Standard:

- CalDmm.drv - ATEasy DMM driver template
- CalDmm.h – C/C++ header for DMM driver template

### Ag3458 – Keysight Technologies/Agilent/HP, GPIB, 3458A, 8.5 digits:

- CalDmmAg3458.dll – DLL driver
- CalDmmAg3458.prj – ATEasy project

- CalDmmAg3458.sys – ATEasy system
- Requires ATEasy GPIB Interface setup

**GX2065 - Marvin Test Solutions, 3U PXI, GX2065, 6.5 digits:**

- CalDmmGX2065.dll – DLL driver
- CalDmmGX2065.prj – ATEasy project
- CalDmmGX2065.sys – ATEasy system

**Ki200x – Keithley, GPIB, 2001/2002, 7.5/8.5 digits:**

- CalDmmKi200x.dll – DLL driver
- CalDmmKi200x.prj – ATEasy project
- CalDmmKi200x.sys – ATEasy system
- Requires ATEasy GPIB Interface setup

**SMX204x – Signametrics, 3U PXI, SM2040, 6.5 digits:**

- CalDmmSm204x.dll – DLL driver
- CalDmmSm204x.prj – ATEasy project
- CalDmmSm204x.sys – ATEasy system
- Requires Signametrics software

**SMX206x – Signametrics, 3U PXI, SM2060, 7.5 digits:**

- CalDmmSm206x.dll – DLL driver
- CalDmmSm206x.prj – ATEasy project
- CalDmmSm206x.sys – ATEasy system
- Requires Signametrics software

**MATRIX Standard:**

- CalMatrix.drv – ATEasy MATRIX driver template
- CalMatrix.h – C/C++ header for MATRIX driver template

**GX6377 - Marvin Test Solutions, 3U PXI, GX6377, Switch Matrix:**

- CalMatrixGX6377.dll – DLL driver
- CalMatrixGX6377.prj – ATEasy project
- CalMatrixGX6377.sys – ATEasy system
- Requires Marvin Test Solutions GXSW software

**GX6384 - Marvin Test Solutions, 3U PXI, GX6384, Switch Matrix:**

- CalMatrixGX6384.dll – DLL driver
- CalMatrixGX6384.prj – ATEasy project
- CalMatrixGX6384.sys – ATEasy system
- Requires Marvin Test Solutions GXSW software

**GX6616 - Marvin Test Solutions, 6U PXI, GX6616, Switch Matrix:**

- **CalMatrixGX6616.dll – DLL driver**
- CalMatrixGX6616.prj – ATEasy project
- CalMatrixGX6616.sys – ATEasy system
- Requires Marvin Test Solutions GXSW software

**PWRMETER Standard:**

- CalPwrMeter.drv – ATEasy PWRMETER (Power Meter) driver template
- CalPwrMeter.h – C/C++ header for PWRMETER driver template

**KtRFPowerMeter - Keysight Technologies, USB, U8481A, RF Power Meter:**

- **CalPwrMeterU8481A.dll – DLL driver**
- CalPwrMeterU8481A.prj – ATEasy project
- CalPwrMeterU8481A.sys – ATEasy system
- Requires VISA software

**PS Standard:**

- CalPs.drv – ATEasy PS (Power Supply) driver template
- CalPs.h – C/C++ header for PS driver template

**GX70x5/GX70x7 - Marvin Test Solutions, 6U PXI, GX7005/GX7015/GX7007/GX7017, High Power Chassis:**

- CalPsGX70x5.dll – DLL driver
- CalPsGX70x5.prj – ATEasy project
- CalPsGX70x5.sys – ATEasy system
- Requires Marvin Test Solutions GXPS software

**GX7400, Marvin Test Solutions, GX7400, 6U PXI, Power Supply:**

- CalPsGX7400.dll – DLL driver
- CalPsGX7400.prj – ATEasy project
- CalPsGX7400.sys – ATEasy system
- Requires Marvin Test Solutions GXPIXI software

**REF Standard:**

- CalRef.drv – ATEasy REF (Reference source, voltage source, current source, etc.) driver template
- CalRef.h – C/C++ header for REF driver template

**FI5522A, Fluke, GPIB, 5522A, Calibrator:**

- CalRef5522.dll – DLL driver
- CalRef5522.prj – ATEasy project
- CalRef5522.sys – ATEasy system
- Requires ATEasy GPIB Interface setup

**SCOPE Standard:**

- CalScope.drv – ATEasy SCOPE driver template

- CalScope.h – C/C++ header for SCOPE driver template

**ZT4441- ZTEC, 3U PXI, ZT4441, Scope/Digitizer:**

- CalScopeZT4441.dll – DLL driver
- CalScopeZT4441.prj – ATEasy project
- CalScopeZT4441.sys – ATEasy system
- Requires ZTec ZScopeM software





## Chapter 4 - Using CalEasy

### Overview

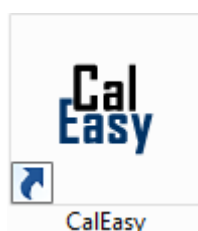
---

This chapter provides an overview of how to use CalEasy to verify and calibrate your instrument. For a list of instrument calibrations supported by CalEasy, refer to chapter 5. Each instrument requires standards instruments in order to perform calibration and verification. Chapter 5 also lists the requirements and capabilities required for each standard instrument.

### Starting CalEasy

---

To start CalEasy, click on the **CalEasy** icon on your desktop or select **Marvin Test Solutions, CalEasy, and CalEasy** from the Windows Start menu.

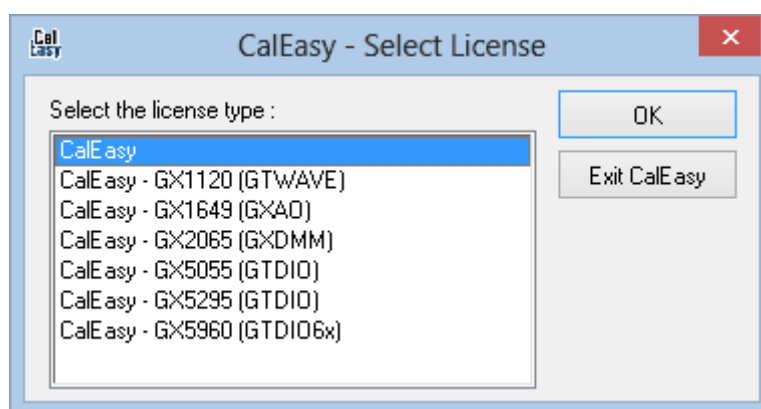


### License Installation

---

When you first start CalEasy you will need to enter a license string. To obtain the license string, create a Marvin Test Solutions support incident from [www.marvintest.com/magic/](http://www.marvintest.com/magic/). You will need to supply the product Serial Number (shown in your packing list) and the Computer ID that is displayed at the bottom of the CalEasy License Setup dialog. Once you have received the license string, you can install it by entering the string when prompted.

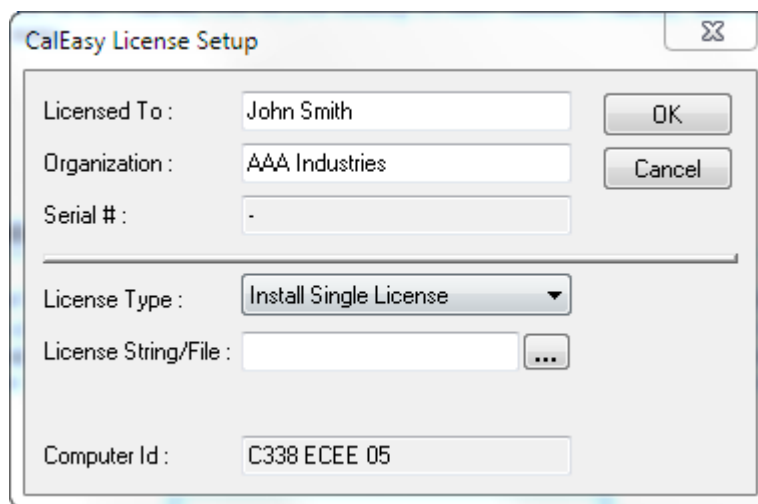
The following dialog will be displayed in no license was found:



**Figure 4-1: CalEasy Select License dialog**

**Note:** Other licenses may be displayed depends on the version of CalEasy installed.

Select the license you purchased and click **OK**. Next, the CalEasy License dialog is displayed showing the Computer ID you will need to provide. Once the license is received, enter the license string and click OK to complete the license installation.



The image shows a Windows-style dialog box titled "CalEasy License Setup". It contains several input fields and buttons. The "Licensed To:" field is filled with "John Smith". The "Organization:" field is filled with "AAA Industries". The "Serial #:" field is filled with "-". The "License Type:" dropdown menu is set to "Install Single License". The "License String/File:" field is empty, with a browse button ("...") to its right. The "Computer Id:" field is filled with "C338 ECEE 05". There are "OK" and "Cancel" buttons on the right side of the dialog.

Licensed To :	John Smith	OK
Organization :	AAA Industries	Cancel
Serial # :	-	
License Type :	Install Single License	
License String/File :		...
Computer Id :	C338 ECEE 05	

**Figure 4-2: CalEasy License Setup dialog**

Following a successful license installation, the Main CalEasy window is displayed as explained in the next section.

## CalEasy Main Window

Once CalEasy is started the main window is displayed as shown below:

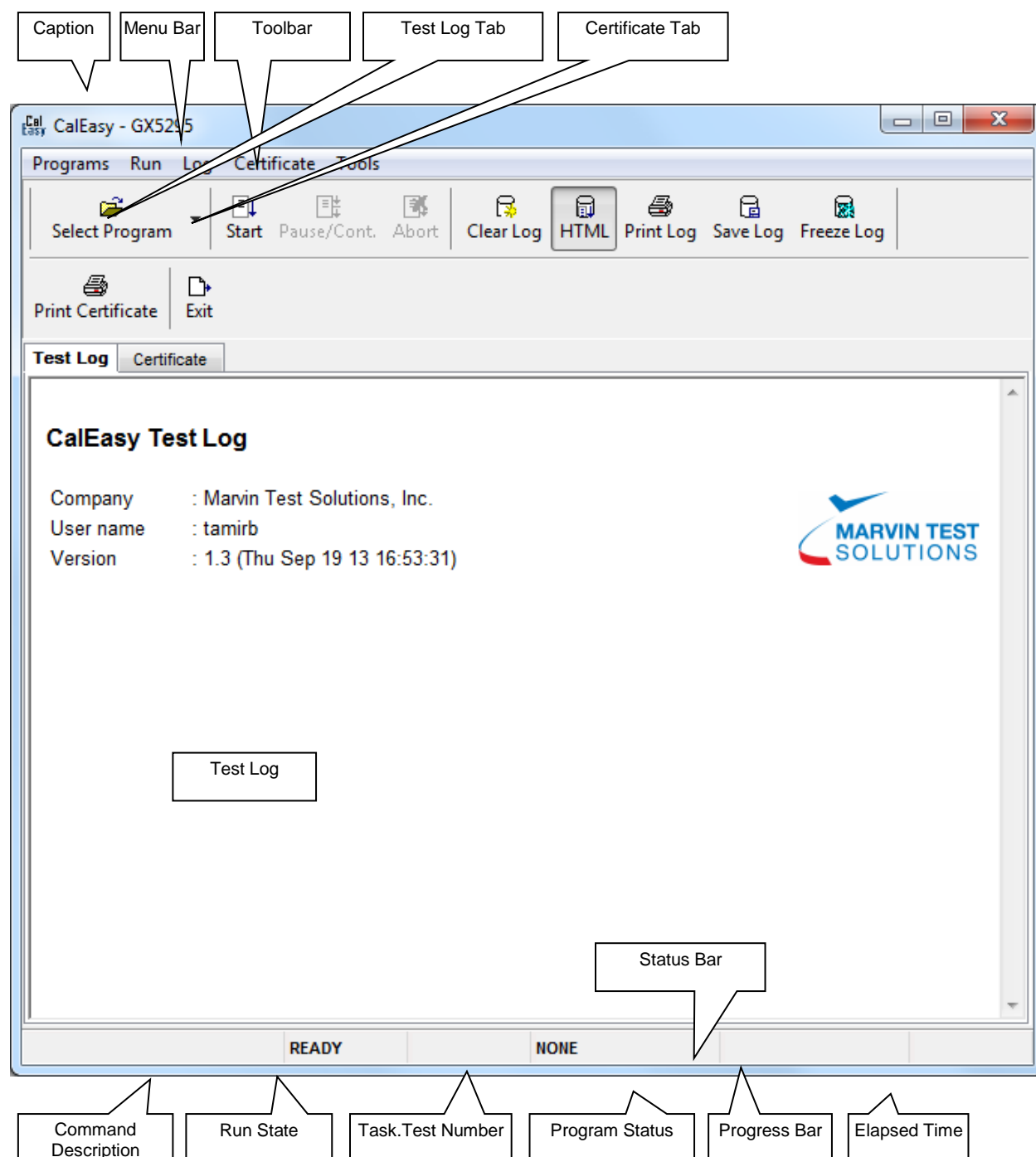


Figure 4-3: CalEasy main window

The following items are displayed:

- **Caption** – Displays the selected calibration program name followed by device serial number and location (Usually PXI Slot).
- **Menu Bar** – The application menu bar.
- **Toolbar** – Displays the toolbar which is used for quick access to invoke commands that are also available from the menu.
- **Test Log Tab** – The Test Log tab displays the verification and calibration program test results in the Test Log page. Results can be displayed in Text or HTML format. The test log is usually saved to the log folder automatically (See Options).
- **Certificate Tab** – The Certificate Tab displays the calibration certificate once the calibration program successfully managed to calibrate the device. The certificate is usually saved to the certificates folder automatically (See Options).
- **Status Bar** – The Status Bar displayed at the bottom of the windows and have the following panes:
  - **Command Description** – Displays the highlighted menu or toolbar item
  - **Run State** – Displays whether the program is running (RUN), Aborted (ABORT) or is idle ready for a new run (READY)
  - **Task.Test Number** – When running it displays the current task/test number being executed
  - **Program Status** – Displays the current program status (NONE, PASS, FAIL, and ERROR). When running verification task, it shows green if all tests are PASS or red if one of the tests fails. When running calibration task, it starts as green and changes to red if one of the tests fails.
  - **Progress Bar** – Displays progress bar of the current running program
  - **Elapsed Time** – Displays the elapsed since the program started to run

## CalEasy Menu and Toolbar Commands

---

The following section describes the CalEasy commands as they appear in the menu and toolbar of the main window.

- **Programs**
  - **Program names** – Sets the current program for execution. Once the program is selected the CalEasy caption will show the selected program.
  - **Exit** – Exits CalEasy. The command is disabled while running the program.
- **Run**
  - **Start** – Starts the current selected program. This command is disabled while running a program.
  - **Pause / Continue** – Suspends / Resumes execution.
  - **Abort** – Aborts the current running program. This command is disabled when the program is not running.
  - **Abort Verification On Failure** – The command can be checked or unchecked. By default, the command is checked. When checked, the verification Task is aborted and will not complete on the first failure. Normally there is no need to complete verification if verification failed in one of the tests. Continuing to the calibration task saves some execution time. CalEasy remembers this option next time it starts.

- **Abort Calibration On Failure** – The command can be checked or unchecked. By default, the command is checked. When checked, the calibration Task is aborted and will not complete on the first failure. Normally there is no need to complete calibration if calibration failed in one of the tests. CalEasy remember this option next time it starts. Uncheck this option to perform all test no matter if failed to check what tests fails and to provide more complete report about the instrument state.
- **Log**
  - **Clear Log** – Clears the current Test Log tab content.
  - **Print Log ...** – Prints the current Test Log tab content.
  - **Save Log ...** – Saves the current Test Log tab content to a file.
  - **Freeze Log...** – Suspends automatic scrolling when a test result is appended to the log.
  - **Clear Log On Start** – This command can be checked or unchecked. By default, it is checked. When checked, CalEasy will clear the log upon start running of a program. Uncheck this command to accumulate multiple runs of the test log to the same file.
  - **HTML Log** – This command can be checked or unchecked. By default, it is checked. When checked, CalEasy will generate HTML test logs. When unchecked, text file format logs will be generated.
- **Certificate**
  - **Clear Certificate** – Clears the current Certificate tab content.
  - **Print Certificate ...** – Prints the current Certificate tab content.
  - **Save Certificate ...** – Saves the current Certificate tab content to a file.
- **Tools**
  - **Standards / Instruments ...** – Displays and modifies the standard instrument calibration information and drivers.
  - **Options ...** – Displays and modifies the program options.
  - **Help.....** – Displays CalEasy user's Guide.
  - **About ...** – Displays program information, version and license.

## Working with CalEasy

---

The following steps are required to perform verification or calibration for your instrument:

1. Start CalEasy
2. Select the required Program for your instrument from your Program menu. CalEasy should display the program name in its caption bar.
3. Run your program. From the Run menu, select Start. CalEasy will start running your program and will prompt you with **the Device Information window**. This window is used to select the device/instrument you are about to calibrate or verify and the program's execution mode. Each program is divided into two portions, **Verification task and Calibration task**. CalEasy will run these tasks as selected by the Mode field in the window. Enter the required information as described in the next section and click OK.
4. The Program will start execution and perform verification or calibration as required. Upon successful calibration the Certificate tab will show. CalEasy normally automatically saves your certificate and the test log to the folder specified in the options form. You can also print the test log or certificate from the Log or Certificate menu.

5. Optionally, you can perform Verification after Calibration is done by running your program again and selecting Verify only from the Device Information dialog (run step 3– 4 again).

## **Test Log**

---

Test logs are created while the program is running. Each test results in a single line that contains the test name, channel and status. These files can be generated in TEXT or HTML file format. You can set the default format from the Options window. The following image shows a sample test log:

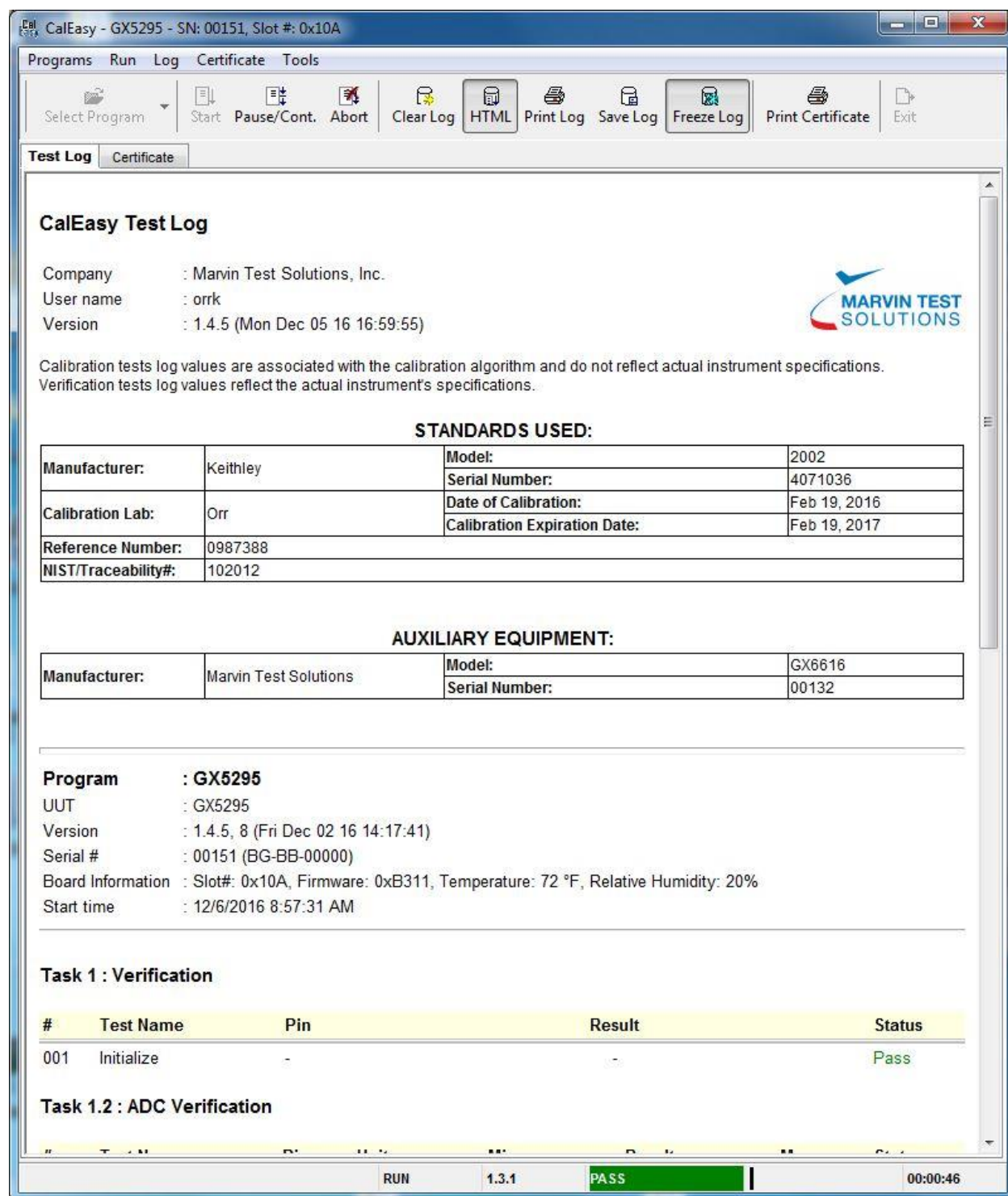


Figure 4-4: Program Test Log Results window

## Calibration Certificates

The calibration certificate is created using the calibration template HTML file (CalEasyCertificateTemplate.htm). This file can be modified to create your own certificate template. The following image shows the calibration certificate created from the default template:

 <p>1770 Kettering Irvine, CA 92614 (949) 263-2222</p>		<h1>Certificate of Calibration</h1> <p><b>Certificate #: Gx5295~00151~2016-12-06-10-50</b></p>	
<p>Marvin Test Solutions, Inc. certifies that the original manufacture product defined on this certificate has been calibrated and meets the manufacturer's specifications. The calibration instruments used are traceable to the National Institute of Standards and Technology (NIST).</p>			
<b>UNIT CALIBRATED:</b>			
<b>Manufacturer:</b>	Marvin Test Solutions, Inc.		
<b>Model:</b>	Gx5295		
<b>Serial Number:</b>	00151	<b>Control Number:</b>	
<input type="checkbox"/> Initial Calibration <input checked="" type="checkbox"/> Recalibration	<b>Received:</b> <input type="checkbox"/> In Tolerance <input type="checkbox"/> Out of Tolerance <input checked="" type="checkbox"/> N/A	<b>Temperature:</b>	72 °F
		<b>Relative Humidity:</b>	20 %
<b>Calibration Date:</b>	Dec 06, 2016	<b>Calibration Time:</b>	10:50:12 AM
<b>Next Calibration Due:</b>	Dec 05, 2017		
<b>STANDARDS USED:</b>			
<b>Manufacturer:</b>	Keithley	<b>Model:</b>	2002
		<b>Serial Number:</b>	4071036
<b>Calibration Lab:</b>	Orr	<b>Date of Calibration:</b>	Feb 19, 2016
		<b>Calibration Expiration Date:</b>	Feb 19, 2017
<b>Reference Number:</b>	0987388		
<b>NIST/Traceability#:</b>	102012		
<b>AUXILIARY EQUIPMENT:</b>			
<b>Manufacturer:</b>	Marvin Test Solutions	<b>Model:</b>	GX6616
		<b>Serial Number:</b>	00132
<b>CALIBRATION PERFORMED BY:</b>			
<b>Name:</b>	orrk		
<b>Signature:</b>			

Figure 4-5: Calibration Certificate



## Device Information Window

The Device Information window is displayed after starting the selected program. The form is used to select the device for calibration or verification as well as select the standard instruments used by the program.

The following screen capture shows the Device Information form:

CalEasy Device Information

Device :  Address :  Mode : ☐ Production Initial Calibration  
 ☐ Multiboard calibration

Serial Number :

Control Number :

Model:GX5055, S/N: 12345-AA-BB-01266, Calibration: 5/10/2009

Standard	Manufacturer	Model	Serial #	Interface	Address	Valid
DMM	Keithley	2002	4071036	GPIB	1:15	OK
MATRIX	Marvin Test ...	GX6616	3333	PXI	0x102	OK

Select :

Interface :  Address :

Temperature :   Humidity (%) :  User Name :

**Figure 4-6: Device Information window**

Enter the required fields as described here:

1. Select the **Mode** as follows:
  - a. Select **Verify** if you wish to verify calibration for the selected device. Verify will not change the device calibration information and will not update the device's non-volatile memory. No certificate will be generated in this mode and only the Verification portion of the program will run.
  - b. Select **Verify and if Passed, Update Calibration Date** if you wish to verify the calibration for the selected device and update the calibration date if the verification passes. This mode will update the non-volatile memory of the instrument and will generate a certificate if the verification passes.
  - c. Select **Verify and Calibrate if Required**. In this mode, CalEasy will first attempt to verify calibration by running the program Verification task. If the device requires calibration, the

program will write and update the device's non-volatile memory with the new date and will issue a new certificate. If the verification task fails, CalEasy will continue and run the Calibration task. Upon successful calibration, the program will write the new calibration information and date to the device's non-volatile memory.

- d. Select **Calibrate Always** to Skip the verification and calibrate the instrument.
2. Select the device's address. Select the device by highlighting a device from the **Address** drop list. The selected device will also cause the device's Serial Number and Summary to be displayed.
3. Select the standard instruments required to perform calibration. For Each instrument displayed in the Standards/Instruments list perform the following
  - a. Highlight the instrument row (DMM in this example)
  - b. Select the instrument from the **Select** drop list and click on the **Set** button. If the **Select** drop list is empty, click on **Other...** to define a standard instrument as described in the next section: **Standards / Instruments Window**. Upon return, select the instrument and click on Set.
  - c. Repeat steps a – b until all of the required instruments are selected. Once all the instruments are selected the Device Information OK button will be enabled.
4. Enter the **Temperature** in C (Celsius) or F (Fahrenheit), type in the **Humidity** and enter your user name. These fields are used sometimes during the calibration or verification process and are also recorded in the calibration certificate and test log.
5. Click **OK** to start running the program and perform verification and/or calibration as selected.

## Standards / Instruments Window

Use this window to define the standards instruments as required by the program to perform calibration or verification. Each entered instrument is recorded in the CalEasy.ini file so it can later be used from the Device Information window. CalEasy uses a fixed DLL functions set for each standard instrument type (i.e. DMM, PS, etc.). Chapter 6 describes the DLL API requirements when implementing an instrument DLL driver that is required by CalEasy as part of an instrument calibration / verification operation.

The following screen capture shows the Standards / Instruments window:

Manufacturer	Model	Serial #	Interface	Address	Valid
Keysight	3458	2823A14363	GPIB	1:15	OK
Keithley	2002	4071036	GPIB	1:15	OK
Marvin Test Soluti...	Gx2065	00036	PXI	0x10F	OK
Marvin Test Soluti...	Gx2065	00033	PXI	0x10C	OK

**Figure 4-7: Standards / Instruments window**

To add an instrument:

1. Select the **Standard** from the Standard list.
2. Enter the **Manufacturer, Model and Serial #** for the instrument.
3. Select the **DLL path**.
4. Enter the instrument **Address**. This could be a string representing a number. For PXI instruments you can enter the PXI slot # (in this example PXI slot #1). For GPIB instrument you can enter 0xBBPP00, where BB is the board number and PP is the primary address, for example 0x10A00 is GPIB board #1, Primary address 10. See the instrument interface Initialize function for more information regarding the instrument address.
5. If the Instrument requires calibration, check the **Required** checkbox and enter the calibration **By**, date (**On**) and Expiration date (**Expired On**). Also, enter the **Reference** and **Trace #**.

6. Click **Add** to add the instrument to the instruments list.

To change an instrument:

1. Select the standard type where the instrument is defined.
2. Select the instrument that you wish to change.
3. Modify the instrument information in the fields (manufacturer, model, etc. and the calibration group).
4. Click **Change** to update the instrument with the new data.

To delete an instrument:

1. Select the standard type where the instrument is defined.
2. Select the instrument that you wish to change.
3. Click **Delete** to remove the instrument.

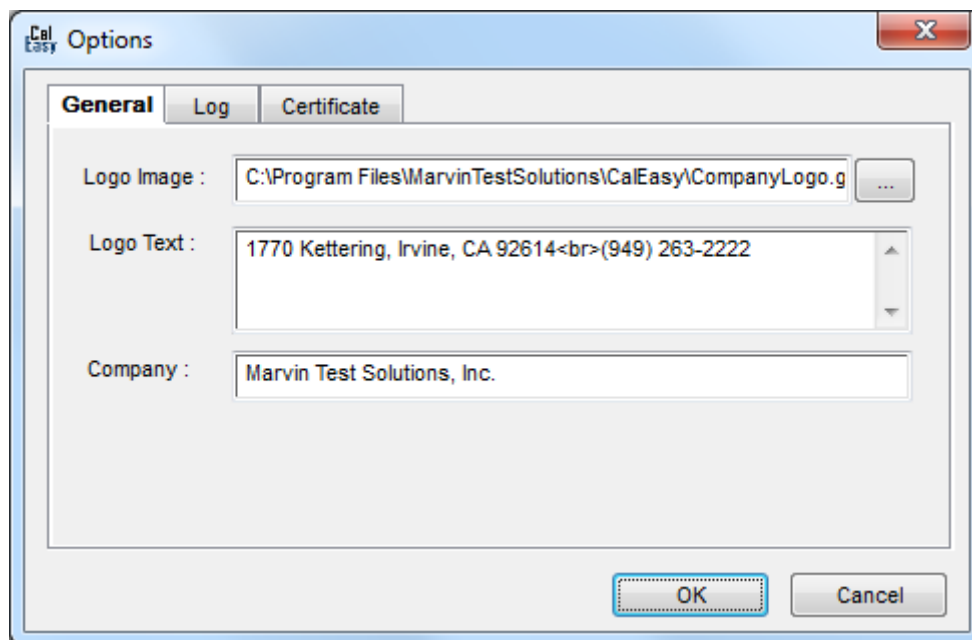
## Setting CalEasy Options

---

Use the CalEasy Options form to set and customize the CalEasy environment. The Options form can be opened from the Tools menu. The following options tabs are available:

### General Options

The following image shows the General options form:

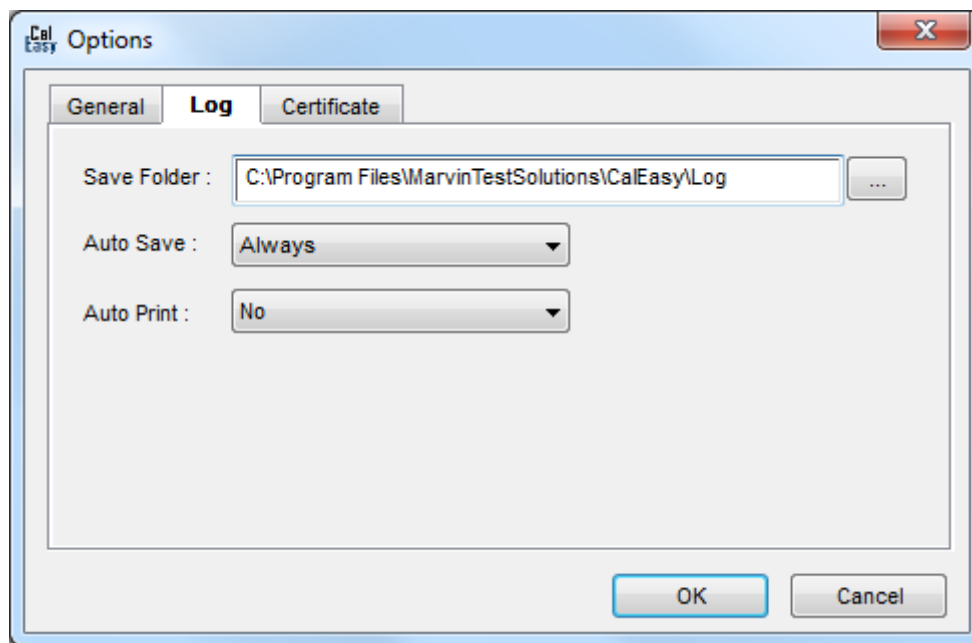


**Figure 4-8: Options General page window**

The General options allow you to set the Company Logo image. The image is displayed in the test log and in the calibration certificate. You can also set the Text displayed next to the Logo and the company name.

## Log Options

The following image shows the Log options form:

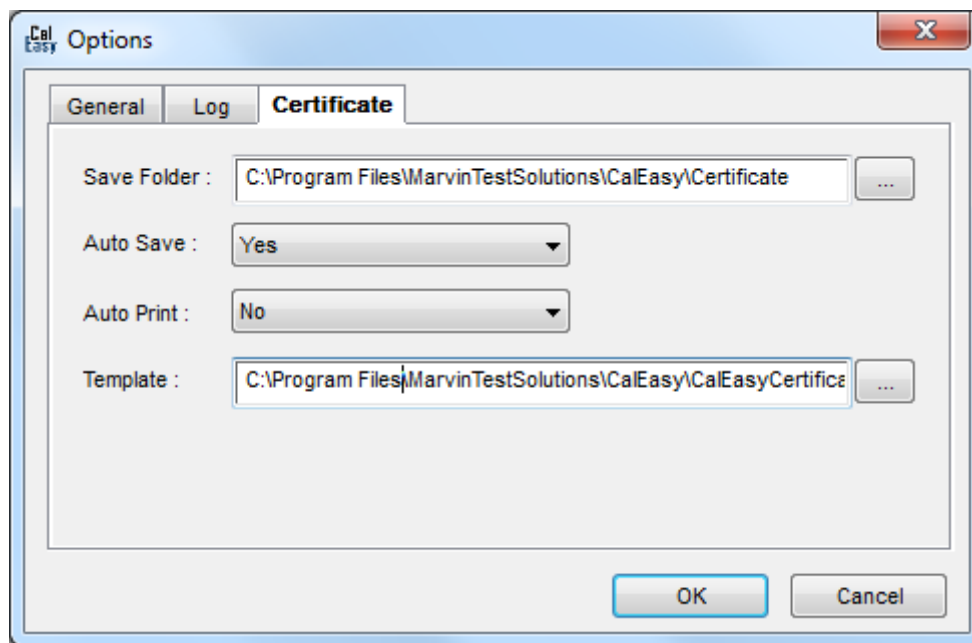


**Figure 4-9: Options Log page window**

The Log options allow you to select the folder where the test log files are saved. You can also choose to automatically save the log always, when the program status passes or failed or not save the log automatically. Similar options are available to automatically print at the end of the program's execution. By default, Auto Save is set to **Always** and Auto Print to **No**.

## Certificate Options

The following image shows the Certificate options form:



**Figure 4-10: Options Certificate page window**

The Certificate options allow you to select the folder where the certificate files are saved. You can also choose to automatically save the certificate always, when the program status passes or fails or not save the certificate automatically. Similar options are available to automatically print the certificate at the end of the program run. By default, Auto Save is set to **Always** and Auto Print to **No**.

You can also specify the certificate template file that is used as a template for generating certificates. By default, the **CalEasyCertificateTemplate.htm** is selected (located in CalEasy folder). You can modify this file to customize the certificate formatting, using an HTML editor to open the file modify and save to your own template.

## Chapter 5 - Device Calibration

### Overview

---

This chapter describes the Marvin Test Solutions instruments that CalEasy supports for Verification and Calibration. Each Marvin Test Solutions instrument is calibrated or verified using a specific verification and calibration program and procedures, standards instrument(s) and additional hardware or cable(s).

### GX1110 Calibration

---

This section describes the requirements for calibrating Marvin Test Solutions GX1110 Arbitrary Waveform Function Generator PXI card.

#### Overview

**Supported Devices:** GX1110

**Program:** GX1110

#### Required Hardware:

1. GX1110
2. 50 Ohm 3 Watt (at minimum) BNC Male Load, Up To 1 GHz BW., +/- 2/5% accuracy
3. BNC T connector with 1 male and two female connections to connect with up to three co-axial cables.
4. BNC Female to Double Stacking Banana Plug Adapter.
5. 1' BNC male-to-male cable.

#### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GtWave v1.3.2 or newer
3. CalEasy v1.4.1 or newer
4. Standards driver manufacturer as specified below

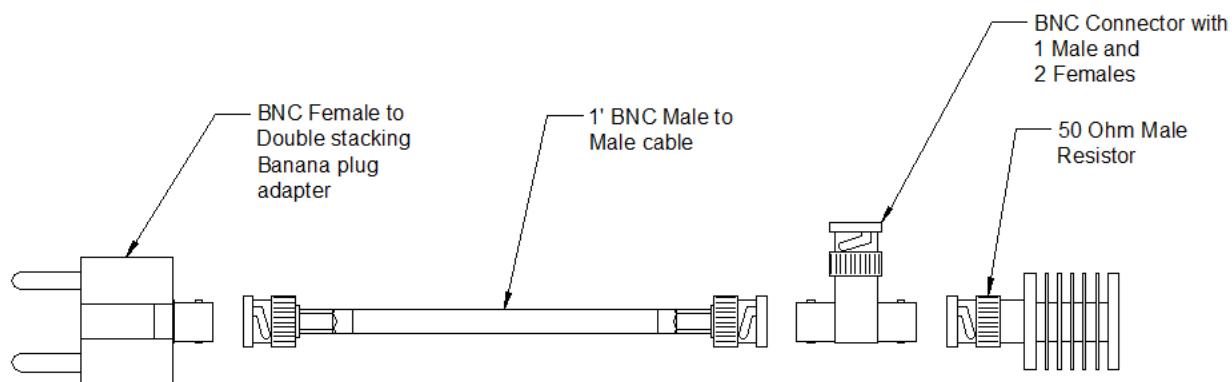
#### Required Standards and Capabilities:

1. **DMM** – 6 ½ DMM with VDC and IDC measurement functions.  
**Provided instrument drivers:** Agilent 3458, Keithley 200x (2001, 2002), Marvin Test Solutions GX2065, Signametrics SMX204x (SM2040, SM2042, SM2044), Signametrics SMX206x.
2. **COUNTER** – Universal Counter with at least 10 digits of measurement accuracy  
**Provided instruments drivers:** Marvin Test Solutions GX2200, Keysight Technologies Ag53131A/Ag53132A.

**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

## Calibration Harness Connections

The following diagram shows the calibration harness connections in order to calibrate the GX1110 board.



### Connections instructions

1. Connect the GX1110 channel 1 to the SMA Plug to BNC Female Adapter.
2. Connect the BNC T connector to the 50-ohm terminator.
3. Connect the BNC T connector to the BNC cable.
4. Connect the BNC cable to the BNC Female to Double Stacking Banana Plug Adapter.

### Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for DMM.
3. Click Start.
4. Select the verification and calibration program mode: **Verify**, **Verify and if Passed, Update Calibration Date**, **Verify and Calibrate If Required** or **Calibrate**.
5. Run Calibration Program.
6. Follow the prompted harness connection messages.

### Approximate Test Time

- Verification – About 4 minutes
- Calibration – About 6 minutes.



## GX1120 Calibration

---

This section describes the requirements for calibrating Marvin Test Solutions GX1120 Arbitrary Waveform Function Generator PXI card.

### Overview

**Supported Devices:** GX1120

**Program:** GX1120

### Required Hardware:

1. GX1120
2. 50 Ohm 3 Watts (at minimum) BNC Male Load Up To 1 GHz, +/- 2.5% accuracy.
3. SMB Plug to BNC Female Adapter.
4. BNC T connector with 1 male and two female connections to connect with up to three co-axial cables.
5. BNC Female to Double Stacking Banana Plug Adapter.
6. 1' BNC male-to-male cable.

### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GtWave v1.3.2 or newer
3. CalEasy v1.4.1 and above
4. Standards driver manufacturer as specified below

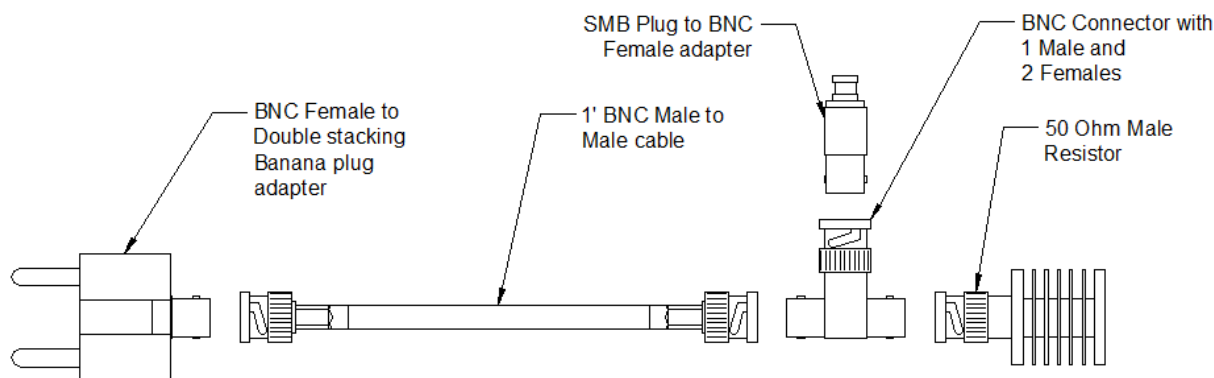
### Required Standards and Capabilities:

1. **COUNTER** – Universal Counter with at least 10 digits of measurement accuracy  
**Provided instruments drivers:** Marvin Test Solutions GX2200, Keysight Technologies Ag53131A/Ag53132A.
2. **DMM** – 6 ½ DMM with VDC and IDC measurement functions.  
**Provided instrument drivers:** Agilent 3458x, Keithley 200x (2001, 2002), Marvin Test Solutions GX2065, Signametrics SMX204x (SM2040, SM2042, SM2044), Signametrics SMX206x.
3. **POWER METER** – power meter with high measurement accuracy  
**Provided instruments drivers:** Keysight U8481A.

**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

## Calibration Harness Connections

The following diagram shows the calibration harness connections in order to calibrate the GX\*1120 board.



### Connections instructions

1. Connect the GX1120 channel 1 to the SMB Plug to BNC Female Adapter.
2. Connect the SMB Plug to BNC Female Adapter to the BNC T connector.
3. Connect the BNC T connector to the 50 Ohm terminator.
4. Connect the BNC T connector to the BNC cable.
5. Connect the BNC cable to the BNC Female to Double Stacking Banana Plug Adapter.

### Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for Counter, DMM, and Power Meter.
3. Click Start.
4. Select the verification and calibration program mode: **Verify, Verify and if Passed, Update Calibration Date, Verify and Calibrate If Required** or **Calibrate**.
5. Run Calibration Program.
6. Follow the prompted harness connection messages.

### Approximate Test Time

- Verification – About 35 minutes
- Calibration – About 35 minutes.

## GX1649 Calibration

---

This section describes the requirements for calibrating Marvin Test Solutions GX1649 Arbitrary Waveform Generator and DC Source PXI card.

### Overview

**Supported Devices:** GX1649

**Program:** GX1649

### Required Hardware:

1. GX1649
2. GX6384 or GX6616 Switch Matrix
3. 7 ½ or 8 ½ digit DMM
4. Calibration Cable, GT96109

### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GxAo v1.3 or newer
3. CalEasy v1.4.1 or newer
4. Standards driver manufacturer software as specified below

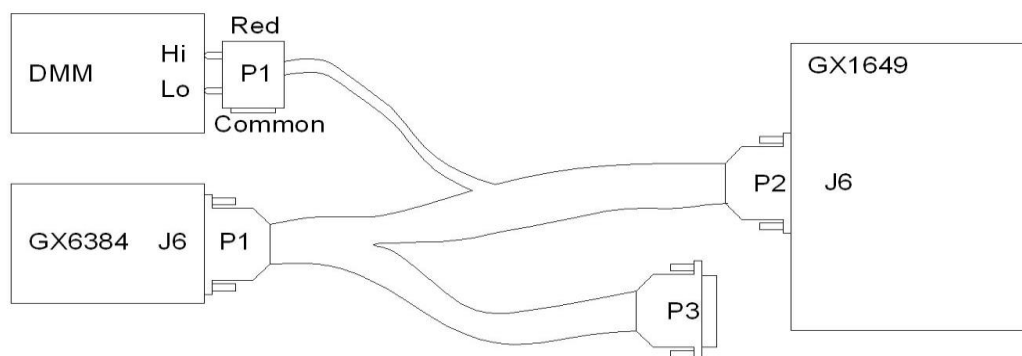
### Required Standards and Capabilities:

1. **DMM** – 8½ DMM with VDC and IDC measurement functions.  
**Provided instrument drivers:** Agilent 3458, Keithley 200x (2002).
2. **MATRIX** – Switch matrix with at least 64 x 2 connections  
**Provided instruments drivers:** Marvin Test Solutions GX6384, GX6616.

**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

## Calibration Harness Connections

The following diagram shows the calibration harness connections in order to calibrate the GX1649 board.



## Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for Matrix, and DMM.
3. Click Start.
4. Select the verification and calibration program mode: **Verify**, **Verify and if Passed, Update Calibration Date**, **Verify and Calibrate If Required** or **Calibrate**.
5. Run Calibration Program.
6. Follow the prompted harness connection messages.

## Approximate Test Time

- Verification – About 2.5 minutes
- Calibration – About 2.5 minutes

## GX1838 Calibration

This section describes the requirements for calibrating Marvin Test Solutions GX1838 Precision Multi-Channel DC Source PXI 3U card.

### Overview

**Supported Devices:** GX1838(-10V to 32 V), GX1838-20 (-20V to 20V)

**Program:** GX1838

### Required Hardware:

1. GX1838 or GX1838-20
2. 6 ½ digit DMM (minimum requirement)
3. GX1838 Calibration cable with load, 1K ohm, +/- 1%; GT96111

### Required Software:

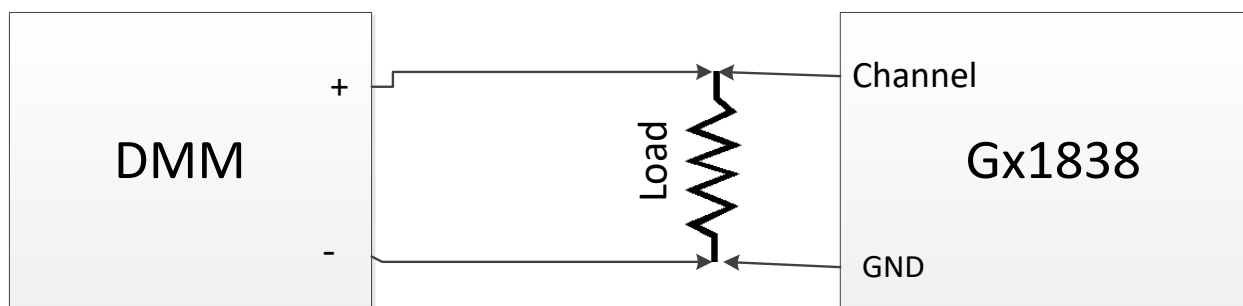
1. ATEasy v9 (build 152c or newer)
2. GxPdo v3.2 or newer
3. CalEasy v1.4.1 or newer
4. Standards driver manufacturer software as specified below

### Required Standards and Capabilities:

1. **DMM** – 6½ DMM (minimum requirement) with VDC and IDC measurement functions.  
**Provided instrument drivers:** Agilent 3458, Keithley 200x (2001, 2002), Marvin Test Solutions GX2065, Signametrics SMX204x (SM2040, SM2042, SM2044) or SM206x.

**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

### Connections instructions



### Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for DMM.
3. Click Start.
4. Select the verification and calibration program mode: **Verify**, **Verify and if Passed, Update Calibration Date**, **Verify and Calibrate If Required** or **Calibrate**.
5. Run Calibration Program.
6. When prompted, connect the DMM and the GX1838 using the GT96111 cable as instructed by the pop up boxes.

### **Approximate Test Time**

- Verification – About 2 minutes.
- Calibration – About 3 minutes.

## GX2065 Series Calibration

---

This section describes the requirements for calibrating the Marvin Test Solutions GX2065 Digital Multi-Meter with CalEasy.

### Overview

**Supported Devices:** GX2065

**Program:** GX2065

### Required Hardware:

1. GX2065 DMM
2. 2 Shielded Double Banana plug cables (Pomona RG58C)
3. 2 High current 30A test leads
4. Fluke 5522A or Equivalent Calibrator

### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GxDmm v2.3 or newer
3. CalEasy v1.4.1 or newer
4. Standards driver manufacturer software as specified below

### Required Standards and Capabilities:

1. **REF** – Calibrator with specifications that are equivalent to the Fluke 5522A specifications  
**Provided instrument driver:** Fluke 5522A.

**NOTE:** See **Standards DLLs Files** section for more information about the Standards Driver software requirements and setup.

### Calibration Harness Connections

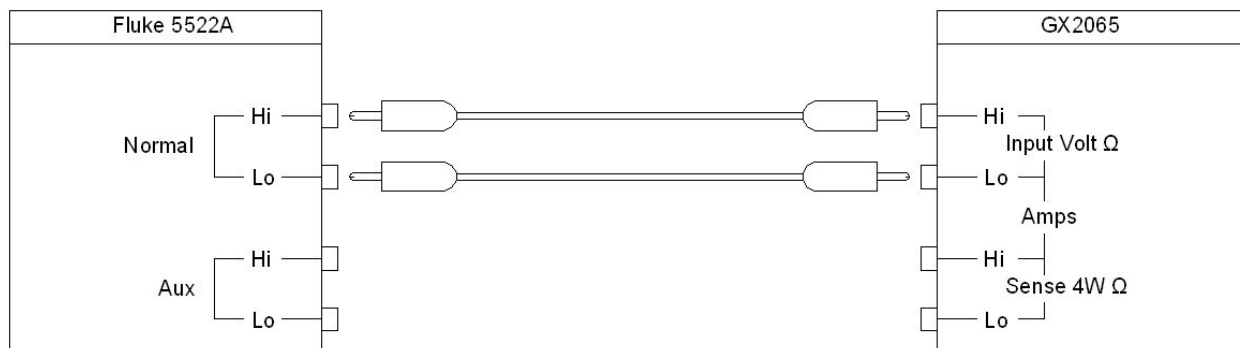
In order to calibrate the GX2065 the following cables are required:

Part number	Description
RG58C/U 24	24 inch double Banana Jack plug cable
5520A-525A/LEADS	Calibration Lead Kit (includes 2x High current, 30A, test lead), available from Fluke Corporation

## Connections instructions

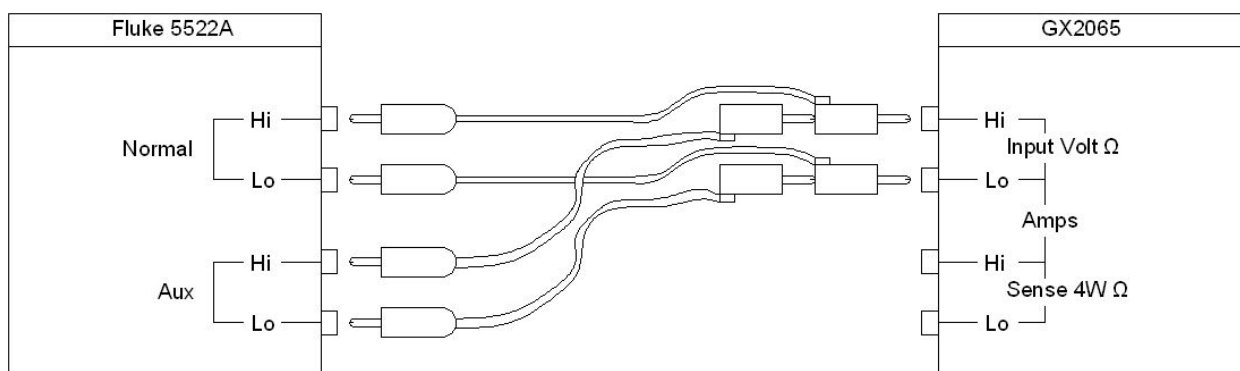
### VDC and VAC connection:

1. Connect Banana Plug from Calibrator Normal Hi and Lo to Gx2065 Input Hi and Lo



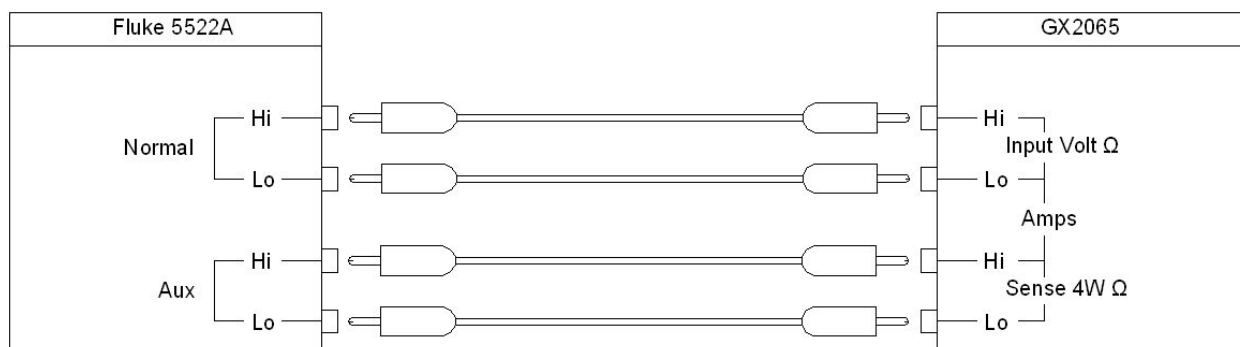
### 2Wire Connection:

1. Compensated: Connect Banana Plug from Calibrator Normal Hi and Lo to Gx2065 Input Hi and Lo
2. Connect another Banana Plug from Calibrator Auxiliary Hi and Lo to Gx2065 Input Hi and Lo



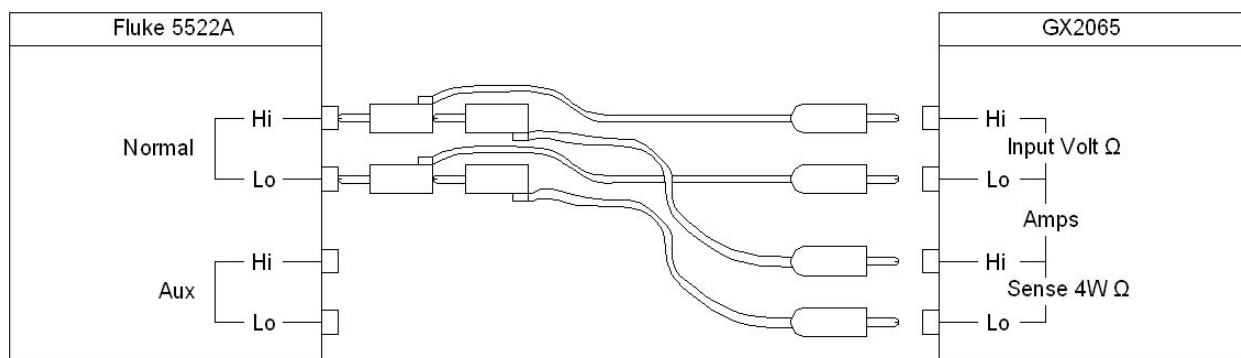
### 4Wire Connection:

1. Compensated: Connect Banana Plug from Calibrator Normal Hi and Lo to Gx2065 Input Hi and Lo
2. Connect another Banana Plug from Calibrator Auxiliary Hi and Lo to Gx2065 Sense Hi and Lo



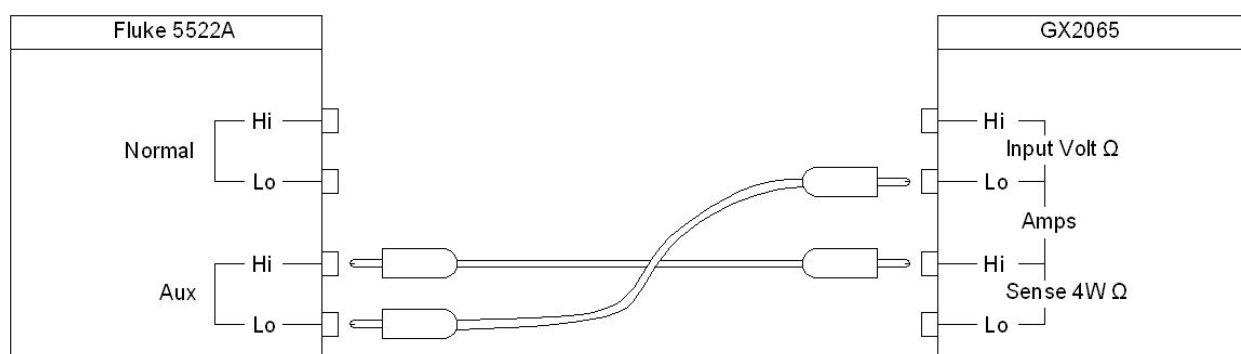
3. Not Compensated: Connect Banana Plug from Calibrator Normal Hi and Lo to Gx2065 Sense Hi and Lo
4. Connect another Banana Plug from Calibrator Normal Hi and Lo to Gx2065 Input Hi and Lo





#### IDC and IAC Connection:

1. Connect low resistance high current cable from Calibrator Auxiliary Hi to Gx2065 Sense Hi. Connect another low resistance high current cable from Calibrator Auxiliary Lo to Gx2065 Lo Voltage Input



#### Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for REF.
3. Click Start.
4. Select the verification and calibration program mode: **Verify**, **Verify and if Passed, Update Calibration Date**, **Verify and Calibrate If Required** or **Calibrate**.
5. Run Calibration Program.
6. When prompted, connect and disconnect the Voltage and Current cables as instructed by pop up boxes.

#### Approximate Test Time

- Verification – About 55 minutes.
- Calibration – About 120 minutes.

## GX3348/GX3364 Calibration

---

This section describes the requirements for calibrating Marvin Test Solutions GX3348 and GX3364 Multi-Channel Analog I/O PXI 6U cards.

### Overview

**Supported Devices:** GX3348 (48 channels), GX3364 (64 channels)

**Program:** GX3348

### Required Hardware:

1. GX3348 or GX3364
2. GX6384 or GX6616 Switch Matrix
3. 6 ½ digit DMM (minimum requirement)
4. Calibration Cable, GT96109

### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GxPdo v3.2 or newer
3. CalEasy v1.4.1 or newer
4. Standards driver manufacturer software as specified below

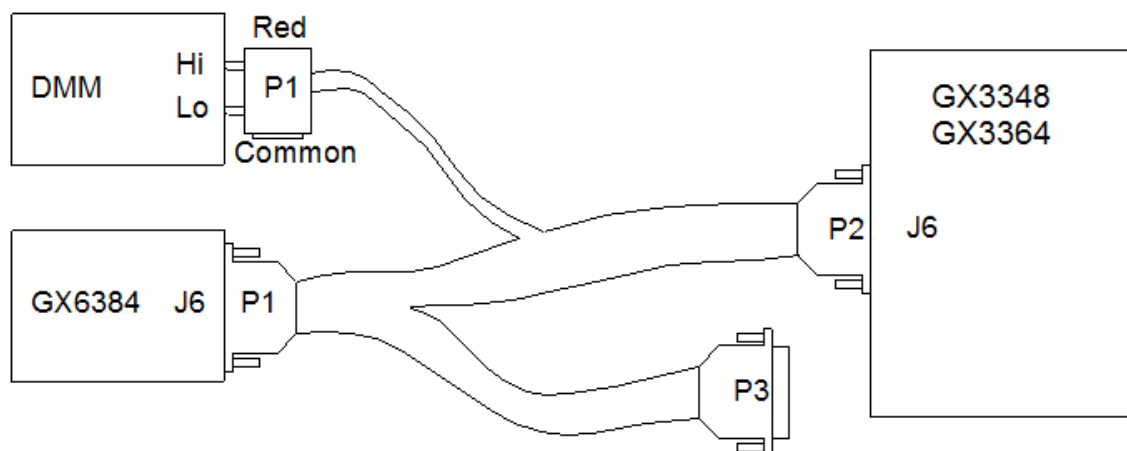
### Required Standards and Capabilities:

1. **DMM** – 6½ DMM (minimum requirement) with VDC and IDC measurement functions.  
**Provided instrument drivers:** Agilent 3458, Keithley 200x (2001, 2002), Marvin Test Solutions GX2065, Signametrics SMX204x (SM2040, SM2042, SM2044) or SM206x.
2. **MATRIX** – Switch matrix with at least 64 x 2 connections  
**Provided instruments drivers:** Marvin Test Solutions GX6384, GX6616.

**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

## Calibration Harness Connections

The following diagram shows the calibration harness connections in order to calibrate the GX3348 board.



## Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for Matrix, and DMM.
3. Click Start.
4. Select the verification and calibration program mode: **Verify, Verify and if Passed, Update Calibration Date, Verify and Calibrate If Required or Calibrate.**
5. Run Calibration Program.
6. Follow the prompted harness connection messages.

## Approximate Test Time

- Verification – About 3.5 minutes
- Calibration – About 3.5 minutes

## GX3788 Calibration

---

This section describes the requirements for calibrating Marvin Test Solutions' GX3788 FPGA Multi-Function PXI 3U card.

### Overview

**Supported Devices:** GX3788

**Program:** GX3788

### Required Hardware:

1. GX3788
2. GX6377 or GX6616 Switch Matrix
3. 6 ½ digit DMM (minimum requirement)
4. Calibration cable kit, GX3788-CALKIT, consisting of the GX95962 and a GT950xx SCSI cable assembly

### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GxFpga v2.3.4 or newer
3. CalEasy v1.4.4 or newer
4. Standards driver manufacturer software as specified below

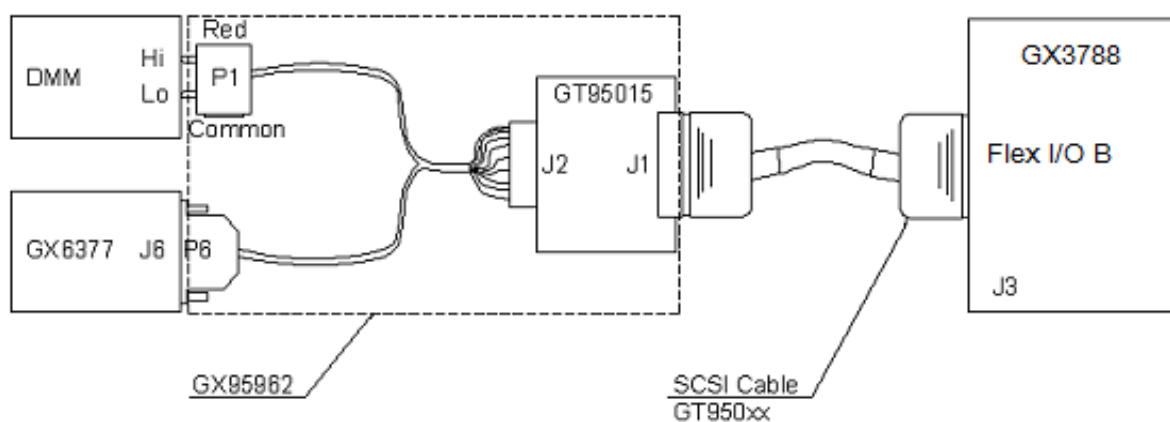
### Required Standards and Capabilities:

1. **DMM** – 6½ DMM (minimum requirement) with VDC and IDC measurement functions.  
**Provided instrument drivers:** Agilent 3458, Keithley 200x (2001, 2002), Marvin Test Solutions GX2065, Signametrics SMX204x (SM2040, SM2042, SM2044) or SM206x.
2. **MATRIX** – Switch matrix with at least 32x2 rows/columns. The channel I/O must support the pin-out of the GX95962 Calibration Harness.  
**Provided instruments drivers:** Marvin Test Solutions GX6377 or GX6616.

**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

## Calibration Harness Connections

The following diagram shows the calibration harness connections in order to calibrate the Gx3788 board.



## Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for Matrix, and DMM.
3. Click Start.
4. Select the required mode the verification and calibration program should run: **Verify, Verify and if Passed, Update Calibration Date, Verify and Calibrate If Required** or **Calibrate**.
5. Run Calibration Program.
6. Follow the prompted harness connection messages.

## Approximate Test Time

- Verification – About 9 minutes
- Calibration – About 15 minutes

## GX5055 Calibration

---

This section describes the requirements for calibrating Marvin Test Solutions' GX5055 Dynamically Controlled, High Voltage Digital I/O PXI Card with Pin Electronics cards using CalEasy.

### Overview

**Supported Devices:** GX5055

**Program:** GX5055

### Required Hardware:

1. GX5055
2. GX6377 or GX6616 Switch Matrix
3. GX95962 – GX5055 Calibration Harness.
4. GT950xx SCSI cable assembly
5. GX95963 – GX5055 Power Cable. Required only when calibrating the module using a standalone power supply (GX7400 or other), do not use if calibrating the module in a GX70x5/GX70x7 chassis. The cable connects the GX5055 to a VCC and VEE power supply source and is wired for direct connection to Marvin Test Solutions' GX7400 PXI power supply module.

Note: Items 3, 4 and 5 are included in the GX5055-5960-CALKIT which includes the GT95032-12, 12 inch SCSI cable assembly.

### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GtDio v5.1 build 70 or newer
3. CalEasy v1.4.1 or newer
4. Standards driver manufacturer software as specified below

### Required Standards and Capabilities:

1. **DMM** – 6 ½ DMM with VDC and IDC measurement functions.  
**Provided instrument drivers:** Agilent 3458, Keithley 200x (2001, 2002), Marvin Test Solutions GX2065, Signametrics SMX204x (SM2040, SM2042, SM2044) or SM206x.
2. **SWITCH MATRIX** – Switch matrix with at least 32x2 rows/columns, channel must support pin-out of the GX95962 Calibration Harness.  
**Provided instruments drivers:** Marvin Test Solutions GX6377 or GX6616.
3. **PS** – Power Supply with two independently controlled rails that can supply +16V/-16V and +18V/-14V. Each rail must be able to supply a maximum of 4A of current.  
**Provided instruments drivers:** Marvin Test Solutions GX70x5 (GX7005, GX7015), GX70x7 (GX7007, GX7017), Marvin Test Solutions GX7400.
4. **NOTE:** The GX7400 power supply is not required if the GX5055 is installed in a GX70x5/GX70x7 chassis.

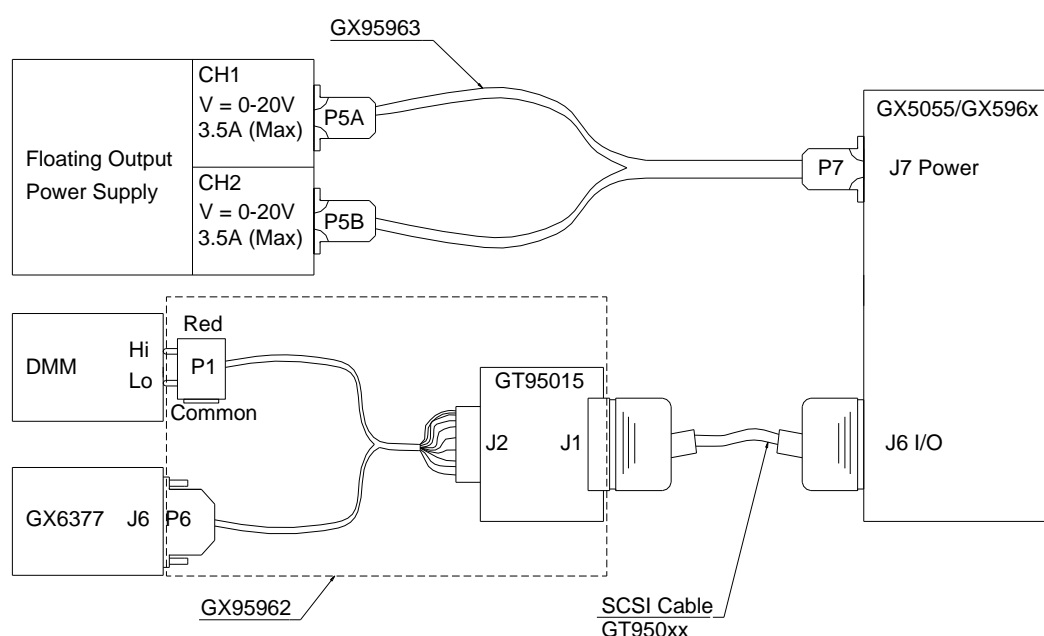
**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

## Calibration Harness Connections

In order to calibrate the Gx5055 the following cables are required:

Part number	Description
GX95962	GX5055 Calibration Harness. The harness connects the Gx5055 I/O connector (J6) to the switch matrix and the DMM using the 68-Pin SCSI shielded cable that was shipped with the Gx5055. The 68-Pin SCSI shielded cable can be any of the following: GT95021 2' shielded cable GT95022 3' shielded cable GT95032 6' shielded cable
GX95963	GX5055 Power Cable. The cable is used if the Gx5055 board is not installed in the Marvin Test Solutions' high-power GX70x5/GX70x7 chassis.

The following diagram shows the calibration harness connections in order to calibrate the Gx5055 board.



## Connections instructions

1. Connect one side of the 68-Pin SCSI shielded cable that was shipped with the GX5055 to the GX5055 J6 I/O connector and the other side to the GT95015 (Connector Interface board) which is part of the GX95962 Calibration Harness.
2. Connect GX95962 P6 to the Switch Matrix.
3. Connect GX95962 P1 to the DMM (for voltage measurement).

Power supply connections (if not using Marvin Test Solutions' high-power GX70x5/GX70x7 chassis):

1. Connect GX95963 P7A or P7B to GX5055 J7 connector.
2. Connect GX95963 P5A to Gx7400 Ch1 Out
3. Connect GX95963 P5B to Gx7400 Ch2 Out

## Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for DMM, Matrix, and Power Supply.
3. Click Start.
4. Select the required mode the verification and calibration program should run: **Verify and if Passed, Update Calibration Date, Verify, Verify and Calibrate If Required or Calibrate.**
5. Run Calibration Program.
6. When prompted, disconnect Calibration Harness SCSI cable from J8 I/O connector on DIO (For Slew Rate calibration).
7. When prompted, reconnect Calibration Harness SCSI cable from J8 I/O connector on DIO.
8. When prompted, connect Calibration Harness P1 to Current measure on DMM (For current calibration).

## Approximate Test Time

- Verification – About 12 minutes (GX7400/Keithley 2001), 8 minutes (GX7400/Sigametrics 2060).
- Calibration – About 45 minutes (GX7400/Keithley 2001), 25 minutes (GX7400/Sigametrics 2060).



## GX5295 Calibration

---

This section describes the requirements for calibrating Marvin Test Solutions GX5295 Dynamically Controlled, Digital I/O PXI Card with PMU per pin using CalEasy.

### Overview

**Supported Devices:** GX5295

**Program:** GX5295

### Required Hardware:

1. GX5295
2. GX6377 or GX6616 Switch Matrix
3. GT950xx SCSI cable assembly
4. GX95962-1 – GX5295 / GX5296 Calibration Harness.

Note: Items 3 and 4 are included in the GX5295-96-CALKIT which includes the GT95032-12, 12-inch SCSI cable assembly.

### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GtDio v4.1 build 56 or newer
3. CalEasy v1.2 or newer
4. Standards driver manufacturer software as specified below

### Required Standards and Capabilities:

1. **DMM** – 7 ½ DMM with VDC and IDC measurement functions that has the following current accuracy measure capabilities (e.g. Keithley (2002):

Force Current Range	Maximum measurement reading error
± 32 mA	± 64 uA
± 8 mA	+/- 16 uA
+/- 2 mA	+/- 4 uA
± 512 uA	+/- 1.024 uA
± 128 uA	+/- 256 nA
± 32 uA	+/- 64 nA
± 8 uA	+/- 16 nA
± 2 uA	+/- 4 nA

**Provided instrument drivers:** Agilent 3458, Keithley 200x (2001/2002).

2. **SWITCH MATRIX** – Switch matrix with at least 32x2 rows/columns, the channel I/O must support the pin-out of the GX95962-1 Calibration Harness.

**Provided instruments drivers:** Marvin Test Solutions GX6377 or GX6616.

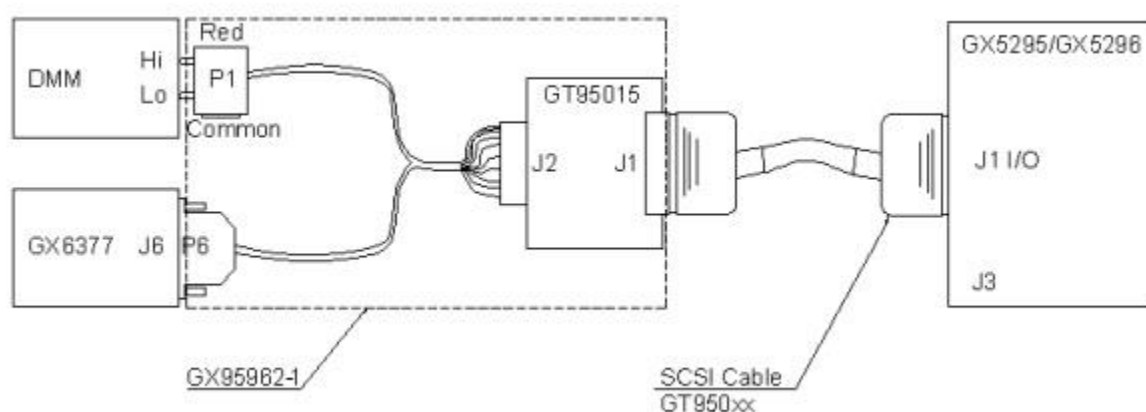
**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

## Calibration Harness Connections

In order to calibrate the Gx5295 the following cables are required:

Part number	Description
GX95962-1	<p>GX5295 Calibration Harness. The harness connects the Gx5295 I/O connector (J1) to the switch matrix and the DMM using the 68-Pin SCSI shielded cable that was shipped with the Gx5295.</p> <p>The 68-Pin SCSI shielded cable can be any of the following:</p> <p>GT95021 2' shielded cable</p> <p>GT95022 3' shielded cable</p> <p>GT95032 6" shielded cable</p>

The following diagram shows the calibration harness connections in order to calibrate the Gx5295 board.



## Connections instructions

1. Connect one side of the 68-Pin SCSI shielded cable that was shipped with the GX5295 to the GX5295 J1 I/O connector and the other side to the GT95015 (Connector Interface board) which is part of the GX95962-1 Calibration Harness.
2. Connect GX95962-1 P6 to the Switch Matrix.
3. Connect GX95962-1 P1 to the DMM (for voltage measurement).

## Calibration Procedure

1. Start CalEasy.exe executable.
2. Select standards instruments for DMM, and Matrix.
3. Click Start.
4. Select the verification and calibration program mode: **Verify**, **Verify and if Passed, Update Calibration Date**, **Verify and Calibrate If Required** or **Calibrate**.
5. Run Calibration Program.
6. When prompted connect Calibration Harness SCSI cable from J1 I/O connector on DIO to J3 connector on DIO.

## Approximate Test Time

- Verification – About 12 minutes (Keithley 2002).
- Calibration – About 45 minutes (Keithley 2002).

## GX5960 Series Calibration

---

This section describes the requirements for calibrating the Marvin Test Solutions GX5961 and GX5964 digital I/O cards with CalEasy.

### Overview

**Supported Devices:** GX5961, GX5964

**Program:** GX5960

### Required Hardware:

1. GX5961 or GX5964
2. GX95962 – GX5960 Calibration Harness.
3. GX6377 or GX6616 Switch Matrix
4. GT950xx SCSI cable assembly
5. GX95963 – GX5960 Power Cable. Required only when calibrating the module using a standalone power supply (GX7400 or other), Do not use if calibrating the module in a GX70x5/GX70x7 chassis. The cable connects the GX5961/64 to a VCC and VEE power supply source and is wired for direct connection to Marvin Test Solutions' GX7400 PXI power supply module.

Note: Items 2, 4 and 5 are included in the GX5055-5960-CALKIT which includes the GT95032-12, 12-inch SCSI cable assembly.

If calibrating the GX5964A module, the GX5960A-CALKIT is available which includes the GX95962 calibration harness assembly, the GX95916 IDC adapter module, a GT95032-12, inch SCSI3 cable assembly and a GX95963 external power cable assembly for powering a GX5960 series modules with GX7400 power supply if the board is not installed in a high power GX70x5 / GX7017 chassis. GX95917-8 and GX95917-6, 8 inch and 6-inch calibration flat cables are also supplied with the kit.

### Required Software:

1. ATEasy v9 (build 152c or newer)
2. GtDio6x v2.0.4 or newer
3. CalEasy v1.4.1 or newer
4. Standards driver manufacturer software as specified below

### Required Standards and Capabilities:

1. **DMM** – 6 ½ DMM with VDC and IDC measurement functions.  
**Provided instrument drivers:** Agilent 3458, Keithley 200x (2001, 2002), Marvin Test Solutions GX2065, Signametrics SMX204x (SM2040, SM2042, SM2044) or SM206x.
2. **MATRIX** – Switch matrix with at least 32x2 rows/columns. The channel I/O must support the pin-out of the GX95962 Calibration Harness.  
**Provided instruments drivers:** Marvin Test Solutions GX6377 or GX6616.
3. **PS** – Power Supply with two independently controlled rails that can supply +17V/-17V and +19V/-15V. Each rail must be able to supply a maximum of 4A of current.  
**Provided instruments drivers:** Marvin Test Solutions GX70x5 (GX7005, GX7015), GX70x7 (GX7007, GX7017) Marvin Test Solutions GX7400.

**NOTE:** The GX7400 power supply is not required if the GX5960 is installed in a GX70x5/GX70x7 chassis.

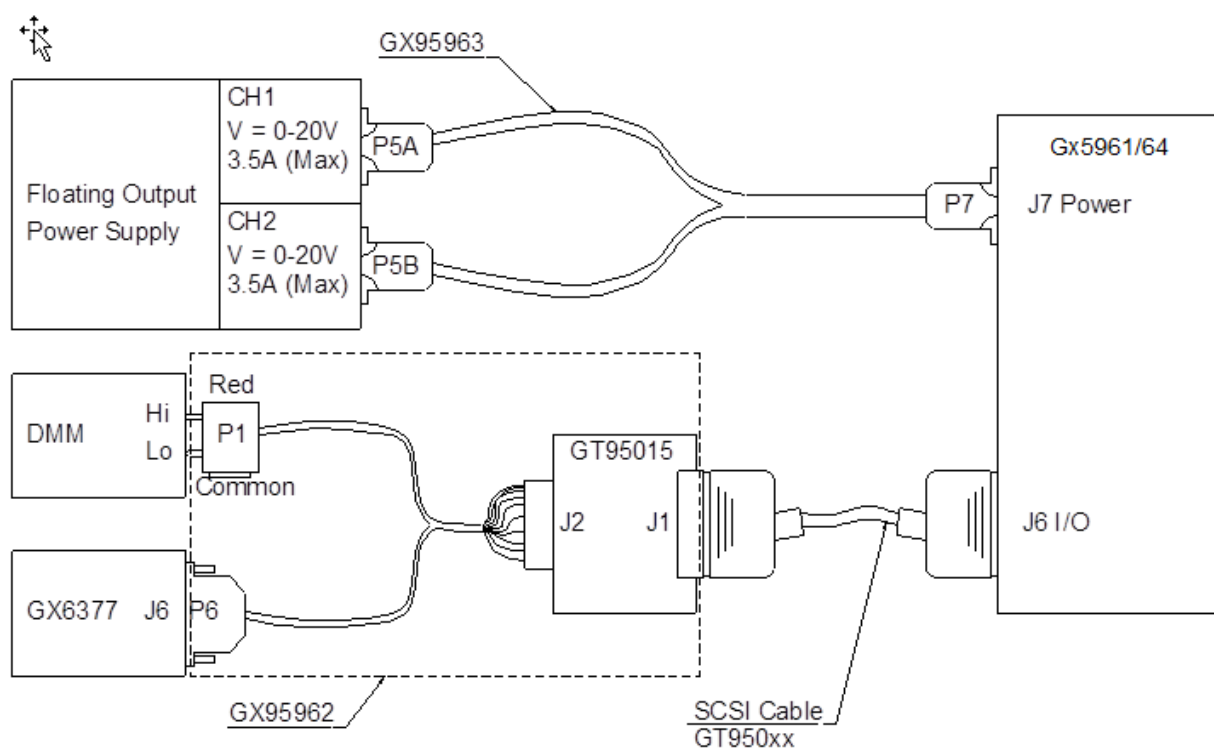
**NOTE:** See **Standards DLLs Files** section for more information about the Standard Driver software requirements and setup.

## Calibration Harness Connections

In order to calibrate the GX5961/GX5964 the following cables are required:

Part number	Description
GX95962	Calibration Harness. The harness connects the GX5960 I/O connector (J6) to the switch matrix and the DMM using the 68-Pin SCSI shielded cable that was shipped with the GX5960. The 68-Pin SCSI shielded cable can be any of the following: GT95021 2' shielded cable GT95022 3' shielded cable GT95032 6' shielded cable
GX95963	Power Cable. The cable is used if the GX5960 board is not installed in the Marvin Test Solutions' high-power GX70x5 or GX70x7 chassis.

The following diagram shows the calibration harness connections in order to calibrate the Gx5961/GX596464 board.



### Connections instructions

1. Connect one side of the 68-Pin SCSI shielded cable that was shipped with the GX5960 to the GX5960 J6 I/O connector and the other side to the GT95015 (Connector Interface board) which is part of the GX95962 Calibration Harness.
2. Connect GX95962 P6 to the Switch Matrix.
3. Connect GX95962 P1 to the DMM (for voltage measurement).

Power supply connections (if not using Marvin Test Solutions' high-power GX70x5//GX70x7 chassis):

1. Connect GX95963 P7A or P7B to GX5960 J7 connector.
2. Connect GX95963 P5A to Gx7400 Ch1 Out

3. Connect GX95963 P5B to Gx7400 Ch2 Out

**Calibration Procedure**

1. Start CalEasy.exe executable.
2. Select standards instruments for DMM, Matrix, and Power Supply.
3. Click Start.
4. Select the verification and calibration program mode: **Verify, Verify and if Passed, Update Calibration Date, Verify and Calibrate If Required** or **Calibrate**.
5. Run Calibration Program.
6. When prompted, connect Calibration Harness P1 to Current measure on DMM (For current calibration).

**Approximate Test Time**

- Verification – About 12 minutes (GX7400/Keithley 2001), 8 minutes (GX7400/Signametrics 2060).
- Calibration – About 35 minutes (GX7400/Keithley 2001), 25 minutes (GX7400/Signametrics 2060).



## Chapter 6 - Standard Drivers

### Overview

This chapter describes the various standard instrument drivers as required by CalEasy calibration programs. Each instrument standard driver type has its own software interface. The software interface is comprised of a DLL with exported functions as defined in this chapter. CalEasy is provided with several standard instrument types including DMM, MATRIX, PS, SCOPE, REF etc. Instruments drivers are installed in the CalEasy\Standards folder. Several examples are also installed in the Standard folders showing how to create such instruments. The provided examples are written in ATEasy; however similar driver/ DLLs can be written with any programming language that can create a 32-bit DLL.

### Writing a Standard Driver

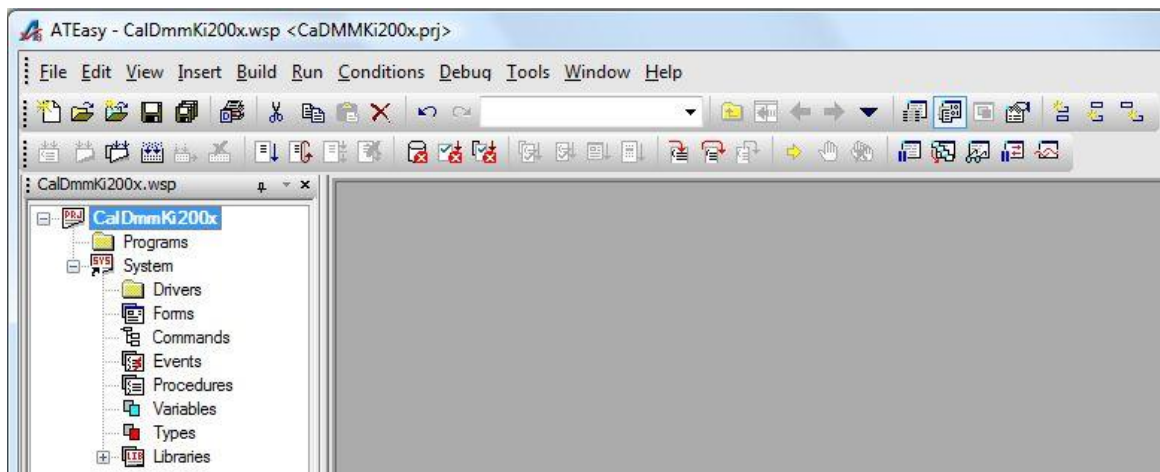
Each standard driver must comply with the standard function interface outlined in the function description within this user guide. The purpose of the standard function interface is to abstract the calibration program's implementation from the actual instrument driver used in the calibration system to a more general implementation.

The provided examples located under CalEasy\Standards folder, are written in ATEasy, however similar driver/ DLLs can be written with any programming language that can create a 32-bit DLL. A C/C++ header file is provided for each standard; this file can be used when writing a driver using C/C++.

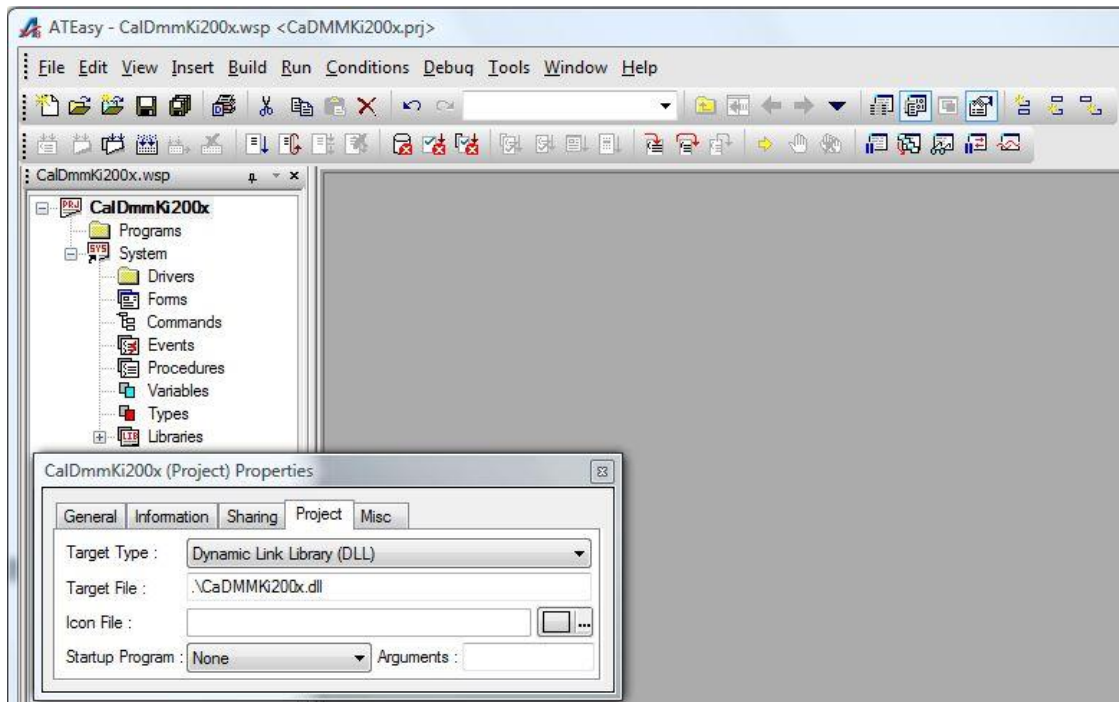
For example, the DMM standard (CalDMM) provides a generic interface when porting a specific DMM (hardware) to work with CalEasy. When the user wants to implement this interface for a certain instrument, they must do so by creating an ATEasy DLL Project. The ATEasy DLL Project will then expose the standard interface and implement each function according to specific software driver of the physical instrument.

Follow these steps to create a Standard Driver implementation:

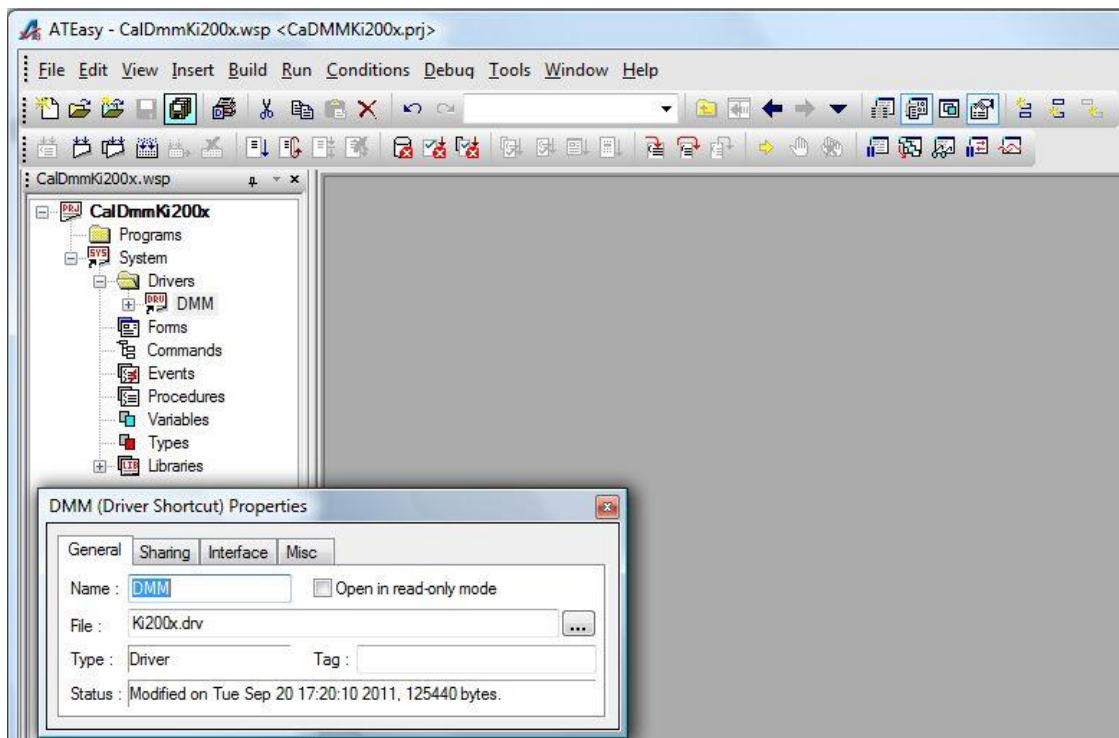
1. Start ATEasy 9.0 and choose to create a new project. Use the naming convention Cal[InstrumentClass][Manufacturer][Model Series] (ex. CalDMMKi200x) for all the project files. Add a new System in the project with the same naming convention.



2. Select the project type to be a DLL in the project properties.

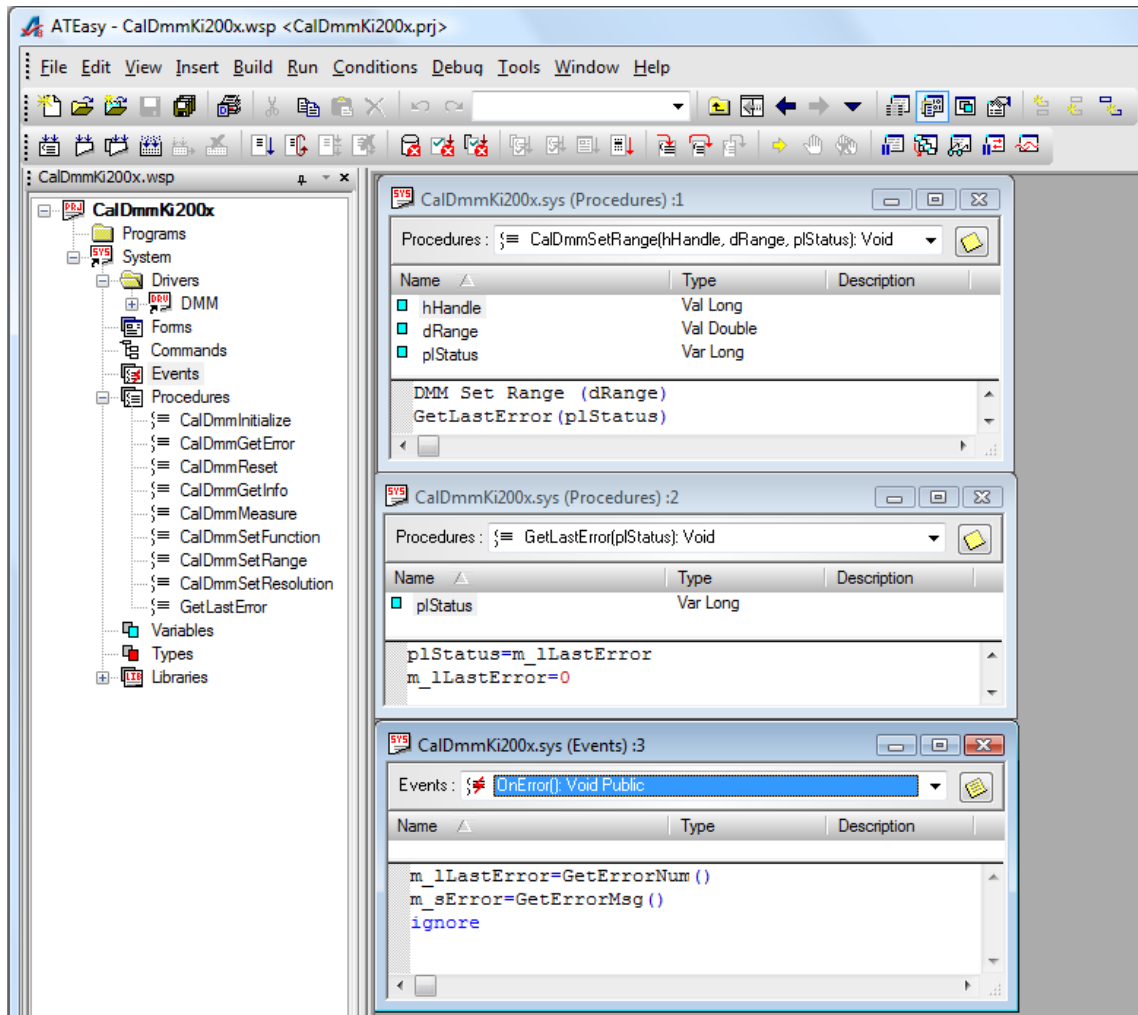


3. Insert the ATEasy driver that controls the specific instrument that is being implemented, if required. The standard interface functions that implemented will use this driver for communication with the hardware.





4. Implement all the standard interface functions in the System's procedure section. Use the ATEasy driver that was previously inserted in the Driver section to communicate with the instrument. Make sure to include the appropriate code for the System's **OnError** event, and the **GetLastError** procedure. The System procedures are exported



5. The examples drivers contain **Initialize** procedure that is not exported by the driver but can be copied to your driver to parse the address passed to your driver in order to initialize the instrument. The *sAddress* parameter passed to your driver for initialization can be one in the format of the following examples:
  - **PXI:0x105** (PXI Chassis 1 Slot 5) or **PXI:5**(PXI legacy slot 5)
  - **GPIB:1:15** (GPIB Board 1, Primary Address 15)
  - **WinSock:192.168.1.5:2045** (Winsock IP Address 192.168.1.5 remote port 2045)
  - **USB:3025:6132:00032** (USB Vendor ID 0x3025, Product ID 0x6132, Serial Number 00032)
  - **COM:4** (Com port 4)
  - **VISA:PXI0::3::12** (VISA resource string PXI0::3::12)

6. The procedures created in the System will be exported from a DLL when the project is built. Click on the **Build** File Menu and select **Build** to generate a DLL with the exported interface functions. This DLL can now be selected in CalEasy when setting up Standards/Instruments.

CalEasy Standards / Instruments

Standard : DMM

Instruments :

Manufacturer	Model	Serial #	Interface	Address	Valid
Keysight	3458	2823A14363	GPIB	1:15	OK
Keithley	2002	4071036	GPIB	1:15	OK
Marvin Test Soluti...	Gx2065	00036	PXI	0x10F	OK
Marvin Test Soluti...	Gx2065	00033	PXI	0x10C	OK

Manufacturer : Keysight Model : 3458 Serial # : 2823A14363

Interface Type : GPIB Address : 1:15 Set...

DLL Path : C:\Program Files (x86)\Marvin Test Solutions\CalEasy\Standa Set...

Calibration : ☒ Required

By : Keysight Reference # : 1234 Trace # : 1234

On : 21 Apr, 2016 00:00 Expired On : 04 Nov, 2016 00:00

OK Help Add Change Delete

## Function Reference Summary

Driver Functions	Description
<b>COUNTER Standard Functions</b>	
CalCounterInitialize	Initializes the COUNTER
CalCounterGetError	Returns error information
CalCounterGetInfo	Returns information on the model and manufacturer of the COUNTER
CalCounterReset	Resets the COUNTER to power up state
CalCounterMeasure	Reads a single measurement
CalCounterSetAcquisitionMode	Sets the board acquisition mode.
CalCounterSetDigitalFilter	Sets the board digital filter settings.
CalCounterSetFunctionPulseWidth	Sets the function mode to Pulse Width.
CalCounterSetGateTime	Sets the Gate time
CalCounterSetImpedance	Sets the specified channel input Impedance.
CalCounterSetSlope	Sets the specified channel input slope.
CalCounterSetTriggerLevel	Sets the specified channel input trigger level.
<b>DMM (Digital Multimeter) Standard Functions</b>	
CalDmmGetFunction	Returns the function that the DMM is currently set to
CalDmmGetError	Returns error information
CalDmmGetInfo	Returns information on the model and manufacturer of the DMM
CalDmmInitialize	Initializes the DMM
CalDmmMeasure	Takes a measurement
CalDmmReset	Resets the DMM to power up state
CalDmmSetFunction	Sets the function of the DMM
CalDmmSetRange	Sets the range of the DMM
CalDmmSetReadingRate	Sets the reading rate of the DMM
CalDmmSetResolution	Sets the digits of resolution of the DMM
<b>MATRIX (Switch Matrix) Standard Functions</b>	
CalMatrixClose	Closes a row/column in the Matrix
CalMatrixGetError	Returns error information
CalMatrixGetInfo	Returns information on the model and manufacturer of the Matrix
CalMatrixInitialize	Initializes the Matrix
CalMatrixOpen	Opens a row/column in the Matrix
CalMatrixReset	Resets the Matrix to power up state
<b>PS (Power Supply) Standard Functions</b>	
CalPsGetChannelState	Returns the on/off state a channel

Driver Functions	Description
CalPsGetCurrent	Returns the current being supplied by a channel
CalPsGetCurrentLimit	Returns the current limit of a channel
CalPsGetError	Returns error information
CalPsGetInfo	Returns information on the model and manufacturer of the Power Supply
CalPsGetVoltage	Returns the voltage being supplied by a channel
CalPsInitialize	Initializes the Power Supply
CalPsReset	Resets the Power Supply to power up state
CalPsSetChannelState	Sets the on/off state of a channel
CalPsSetCurrentLimit	Sets the current limit of a channel
CalPsSetVoltage	Sets the voltage generated by a channel
<b>PWRMETER (Power Meter) Standard Functions</b>	
CalPwrMeterClose	Closes connection to Power Meter
CalPwrMeterGetError	Returns error information
CalPwrMeterGetInfo	Returns information on the model and manufacturer of the Power Meter
CalPwrMeterInitialize	Initializes the Power Meter
CalPwrMeterMeasure	Returns a single measurement based on the specified measurement type.
CalPwrMeterReset	Resets the Power Meter to power up state
CalPwrMeterSetupMeasurement	Sets the board measurement settings.
CalPwrMeterZero	Zeroes the Power Meter.
<b>SCOPE (Oscilloscope) Standard Functions</b>	
CalScopeGetError	Returns error information
CalScopeGetInfo	Returns information on the model and manufacturer of the Oscilloscope
CalScopeGetInputImpedance	Returns the channel's input impedance.
CalScopeInitialize	Initializes the Scope
CalScopeMeasure	Returns a single measurement based on the specified measurement type.
CalScopeReset	Resets the Oscilloscope to power up state
CalScopeSetupAcquisition	Sets the board acquisition settings.
CalScopeSetupAuto	Automatically sets the range, offset, coupling, and sample rate based on the input signal
CalScopeSetupChannel	Configures the selected channel settings.
CalScopeSetupChannelState	Sets the selected channel state.
CalScopeSetupHorizontal	Configures the horizontal and time base settings.
CalScopeSetupInputImpedance	Setup the channel's input impedance.

Driver Functions	Description
<b>REF (Calibrator/Reference Source) Standard Functions</b>	
CalRefGetError	Returns error information
CalRefGetInfo	Returns information on the model and manufacturer of the Reference
CalRefInitialize	Initializes the Reference
CalRefReset	Resets the Reference to power up state
CalRefSet2Wire	Sets the 2Wire Reference
CalRefSet4Wire	Sets the 4Wire Reference
CalRefSetACCurrent	Sets the AC Current Reference
CalRefSetACVolts	Sets the AC Voltage Reference
CalRefSetDCCurrent	Sets the DC Current Reference
CalRefSetDCVolts	Sets the DC Voltage Reference
CalRefSetState	Sets the operational state of the Reference.

## COUNTER Standard Driver

---

### CalCounterGetError

---

#### Purpose

Returns a string containing information about a status code

#### Syntax

**CalCounterGetError**(*nHandle*, *lStatus*, *psError*, *pnStatus*)

#### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter
<i>lStatus</i>	LONG	Error status to decode
<i>psError</i>	PSTR	Returns the error message associated with lStatus
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

#### Comments

#### Example

The following example gets the error information for **LErrorStatus**:

```
LONG    lHandle, lErrorStatus, lFunction, lStatus;

CalCounterGetError (lHandle, lErrorStatus, psError, &lStatus);
Printf("Error Status code %d is: %s", lErrorStatus, psError);
```

#### See Also

## CalCounterGetInfo

---

### Purpose

Returns the manufacturer and model of the Counter

### Syntax

**CalCounterGetInfo**(*nHandle*, *psManufacturer*, *psModel*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter
<i>psManufacturer</i>	PSTR	Returns a string with the manufacturer name
<i>psModel</i>	PSTR	Returns a string with the model name
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the Matrix information:

```
LONG    lHandle, lStatus;  
CHAR    psManufacturer[256];  
CHAR    psModel[256];  
  
CalCounterGetInfo (lHandle, sManufacturer, sModel, &lStatus);  
Printf("The Manufacturer is: %s and the Model is: %s", psManufacturer, psModel);
```

### See Also

#### CalCounterGetError

## CalCounterInitialize

---

### Purpose

Initializes the driver for the board at the specified address of VISA resource. The function returns a handle that can be used with other CalCounterXXXX functions to program the counter.

### Syntax

**CalCounterInitialize** (*sAddress*, *pnHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>sAddress</i>	STR	Address of Counter.
<i>phHandle</i>	PLONG	Returned handle for Counter access
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function verifies and establishes communication with the instrument at the specified address information. The address information can be PXI, GPIB, USB, WinSock or VISA resource string. This address string format is starting by the address type followed by a column and the address: **PXI / GPIB / WinSock / USB / Com / VISA : Address** . The specifics of the **Address** string characteristics can be found in **Writing a Standard Driver** topic in step 5. The function does not change any of the board settings.

The returned handle *phHandle* is used to identify the specified instrument with other driver functions.

### Example

The following example initializes two counters at address 2 and 0x103.

```
SHORT nHandle1, nHandle2, nStatus;

CalCounterInitilize ("PXI:2", &nHandle1 &nStatus);
CalCounterInitilize ("PXI:0x103", &nHandle2, &nStatus);
```

### See Also

**CalCounterReset**, **CalCounterGetError**



## CalCounterMeasure

---

### Purpose

Reads a single measurement.

### Syntax

**CalCounterMeasure** (*hHandle*, *nChannel*, *pdMeasurement*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>pdMeasurement</i>	PDOUBLE	Returned measurement.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function is for reading measurement results out of the board. If a result is available it is returned immediately, otherwise the function will wait for a measurement to become available.

### Example

The following example read a single measurement from channel 0:

```
LONG    lHandle, lStatus;
DOUBLE dMeasurement;

CalCounterMeasure (lHandle, 0, &dMeasurement, &lStatus);
```

### See Also

**CalCounterSetFunctionPulseWidth**, **CalCounterSetAcquisitionMode**, **CalCounterSetGateTime**, **CalCounterSetTriggerLevel**, **CalCounterGetError**

## CalCounterReset

---

### Purpose

Resets the Counter

### Syntax

**CalCounterReset** (*hHandle*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function resets the Counter to power up state.

### Example

The following example resets the Counter

```
LONG    lHandle, lStatus;
```

```
CalCounterReset (lHandle, &lStatus);
```

### See Also

**CalCounterInitialize**, **CalCounterGetError**

## CalCounterSetAcquisitionMode

---

### Purpose

Sets the board acquisition mode.

### Syntax

**CalCounterSetAcquisitionMode** (*hHandle*, *nChannel*, *lAcquisitionMode*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>lAcquisitionMode</i>	LONG	Acquisition mode: <ul style="list-style-type: none"> <li>0 = <code>aCounterAcquisitionContinuous</code>: The counter continuously makes measurements (default after reset).</li> <li>1 = <code>aCounterAcquisitionSingle</code>: Instrument makes a single measurement. Each call to trigger initiates a new measurement.</li> </ul>
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets a channel's acquisition mode.

### Example

The following example sets channel 0 acquisition mode to continuous:

```
LONG    lHandle, lStatus;
```

```
CalCounterSetAcquisitionMode (lHandle, 0, aCounterAcquisitionContinuous, &lStatus);
```

### See Also

**CalCounterMeasure**, **CalCounterSetFunctionPulseWidth**, **CalCounterSetGateTime**, **CalCounterSetTriggerLevel**, **CalCounterGetError**

## CalCounterSetDigitalFilter

---

### Purpose

Sets the specified channel filter value and mode.

### Syntax

**CalCounterSetDigitalFilter** (*hHandle*, *nChannel*, *lFilterMode*, *lFilterValue*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>lFilterMode</i>	LONG	Digital Filter mode: <ul style="list-style-type: none"> <li>0 = Filter is disabled.</li> <li>1 = Filter is enabled and the driver analyzes the measured signal and determines the best filter value each time a measument is done.</li> <li>2 = Filter is enabled and the user sets the filter value, see <i>lFilterValue</i> paramterfor details.</li> </ul>
<i>lFilterValue</i>	LONG	Filter value is an integer representing filter value in uSec, e.g. value of 50 represents 50uSec. Filter value range from 5 to 6400, (5uSec to 6400uSec).
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The Filter should be used when measuring low frequencies (less than 20KHz) superimposed with a high-frequency noise. The Filter value adds a delay equal to *lFilterValue* in microseconds. After first trigger is detected the counter will disregard any additional trigger events for *lFilterValue* microseconds.

Note: The delay needs to be less than half of the expected frequency signal duty cycle. Filter value can be set when the filter mode is set to fixed value.

The filter value should be set according to the following equation:

$$\text{Filter Value (uS)} \leq \frac{1.0\text{E} + 6}{\text{ExpectedFrequency} * 4}$$

E.g.: if the expected value is about 10KHz then filter value should be around 25uSec.

### Example

The following example sets channel 0 filter value to 55uSec:

```
LONG    lHandle, lStatus;

CalCounterSetDigitalFilter (lHandle, 0, 2, 55, &lStatus);
```

### See Also

**CalCounterMeasure**, **CalCounterSetFunctionPulseWidth** ,**CalCounterSetGateTime**,  
**CalCounterSetTriggerLevel**, **CalCounterGetError**

## CalCounterSetFunctionPulseWidth

---

### Purpose

Sets the function mode to Pulse Width.

### Syntax

**CalCounterSetFunctionPulseWidth** (*hHandle*, *nChannel*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

In this mode the counter measures pulse width on the specified channel. The pulse polarity (i.e. positive or negative) is set by **CalCounterSetSlopefunction**. Threshold levels can be set using **CalCounterSetTriggerLevel**.

### Example

The following example sets channel 0 function mode to Pulse Width:

```
LONG    lHandle, lStatus;

CalCounterSetFunctionPulseWidth (lHandle, 0, &lStatus);
```

### See Also

**CalCounterMeasure**, **CalCounterSetSlopefunction**, **CalCounterSetGateTime**, **CalCounterSetTriggerLevel**, **CalCounterGetError**

## CalCounterSetGateTime

---

### Purpose

Sets the Gate time.

### Syntax

**CalCounterSetGateTime** (*hHandle*, *nChannel*, *dSeconds* *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>dSeconds</i>	DOUBLE	Gate time in seconds.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The gate time sets the minimum measurement interval.

### Example

The following example sets channel 0 gate time to 0.3 Sec:

```
LONG    lHandle, lStatus;
```

```
CalCounterSetGateTime (lHandle, 0, 0.3, &lStatus);
```

### See Also

**CalCounterMeasure**, **CalCounterSetFunctionPulseWidth**, **CalCounterSetGateTime**,  
**CalCounterSetTriggerLevel**, **CalCounterGetError**

## CalCounterSetImpedance

---

### Purpose

Sets the specified channel input impedance.

### Syntax

**CalCounterSetImpedance** (*hHandle*, *nChannel*, *lInputImpedance*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>lInputImpedance</i>	LONG	Channel's input impedance: <ul style="list-style-type: none"> <li>0 = 1MOhm impedance.</li> <li>1 = 50 Ohms impedance.</li> </ul>
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example sets channel 0 impedance to 1MOhm:

```
LONG    lHandle, lStatus;
```

```
CalCounterSetImpedance (lHandle, 0, 0, &lStatus);
```

### See Also

**CalCounterMeasure**, **CalCounterSetFunctionPulseWidth**, **CalCounterSetGateTime**, **CalCounterSetTriggerLevel**, **CalCounterGetError**

## CalCounterSetSlope

---

### Purpose

Sets the specified channel input slope.

### Syntax

**CalCounterSetSlope** (*hHandle*, *nChannel*, *lSlope*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>lSlope</i>	SHORT	Channel's slope: <ul style="list-style-type: none"> <li>0 = aCounterPositiveSlope: Rising (<b>Positive</b>) edge.</li> <li>1 = aCounterNegativeSlope: Falling (Negative) edge.</li> </ul>
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The channel slope sets the input signal edge that utilized for the measurement. The slope can specify either rising (**Positive**) edge or falling (**Negative**) edge.

### Example

The following example sets channel 0 input slope:

```
LONG    lHandle, lStatus;

CalCounterSetSlope (lHandle, 0, 0, &lStatus);
```

### See Also

**CalCounterMeasure**, **CalCounterSetFunctionPulseWidth**, **CalCounterSetGateTime**, **CalCounterSetTriggerLevel**, **CalCounterGetError**



## CalCounterSetTriggerLevel

---

### Purpose

Sets the specified channel input trigger level.

### Syntax

**CalCounterSetTriggerLevel** (*hHandle*, *nChannel*, *dVoltage*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>dVoltage</i>	DOUBLE	Channel's input trigger level voltage.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The Trigger Level setting defines the input signal voltage that creates a logical event from the input comparator circuit (trigger level is also commonly known as “threshold” voltage).

### Example

The following example sets channel 0 input trigger level to 1.2V:

```
LONG    lHandle, lStatus;  
  
CalCounterSetSlope (lHandle, 0, 1.2, &lStatus);
```

### See Also

**CalCounterMeasure**, **CalCounterSetFunctionPulseWidth**, **CalCounterSetGateTime**, **CalCounterGetError**

## DMM Standard Driver

---

### CalDmmGetError

---

#### Purpose

Returns a string containing information about a status code

#### Syntax

**CalDmmGetError**(*nHandle*, *lStatus*, *psError*, *pnStatus*)

#### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM
<i>lStatus</i>	LONG	Error status to decode
<i>psError</i>	PSTR	Returns the error message associated with lStatus
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

#### Comments

#### Example

The following example gets the error information for **LErrorStatus**:

```
LONG    lHandle, lErrorStatus, lFunction, lStatus;  
  
CalDmmGetError (lHandle, lErrorStatus, psError, &lStatus);  
Printf("Error Status code %d is: %s", lErrorStatus, psError);
```

#### See Also

**CalDmmGetError**

## CalDmmGetFunction

---

### Purpose

Returns the currently selected function

### Syntax

**CalDmmGetFunction** (*nHandle*, *penFunction*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM
<i>penFunction</i>	enumCalDmmFunction *	Return the currently set function: aDmmFunctionVDC – Voltage DC Mode aDmmFunctionIDc – Current DC Mode aDmmFunctionVAcAcCpl – Voltage AC, AC Coupled Mode aDmmFunctionIAcAcCpl – Current AC, AC Coupled Mode aDmmFunctionVAcDcCpl – Voltage AC, DC Coupled Mode aDmmFunctionIAcDcCpl – Current AC, DC Coupled Mode aDmmFunction2Wire – 2Wire Resistance Mode aDmmFunction4Wire – 4 Wire Resistance Mode aDmmFunctionContinuity – High Resolution Continuity Mode aDmmFunctionFrequency – Frequency Counter Mode aDmmFunctionDiode – Test Diode aDmmFunctionTemperature – External Temperature Mode
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the currently set function:

```
LONG                lHandle, lStatus;
enumCalDmmFunction enFunction;

GxDmmGetFunction (lHandle, &enFunction, &lStatus);
```

### See Also

**CalDmmSetFunction, CalDmmSetRange, CalDmmGetRange, CalDmmGetError**

## CalDmmGetInfo

---

### Purpose

Returns the manufacturer and model of the DMM

### Syntax

**CalDmmGetInfo**(*nHandle*, *psManufacturer*, *psModel*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM
<i>psManufacturer</i>	PSTR	Returns a string with the manufacturer name
<i>psModel</i>	PSTR	Returns a string with the model name
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the DMM information:

```
LONG    lHandle, lStatus;
CHAR    psManufacturer[256];
CHAR    psModel[256];

CalDmmGetInfo (lHandle, sManufacturer, sModel, &lStatus);
Printf("The Manufacturer is: %s and the Model is: %s", psManufacturer, psModel);
```

### See Also

#### CalDmmGetError

## CalDmmInitialize

---

### Purpose

Initializes the driver for the board at the specified address, which includes support for VISA resource. The function returns a handle that can be used with other CALDMM functions to program the DMM.

### Syntax

**CalDmmInitialize** (*sAddress*, *pnHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>sAddress</i>	STR	Address of DMM.
<i>phHandle</i>	PLONG	Returned handle for DMM access
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function verifies and establishes communication with the instrument at the specified address information. The address information can be PXI, GPIB, USB, WinSock or VISA resource string. This address string format is starting by the address type followed by a column and the address: **PXI / GPIB / WinSock / USB / Com / VISA : Address** . The specifics of the **Address** string characteristics can be found in **Writing a Standard Driver** topic in step 5. The function does not change any of the board settings.

The returned handle *phHandle* is used to identify the specified instrument with other driver functions.

### Example

The following example initializes two DMMs at PXI address 0x1105 and GPIB board 1, address 15.

```
SHORT nHandle1, nHandle2, nStatus;

CalDmmInitilize ("PXI:0x105", &nHandle1 &nStatus);
CalDmmInitilize ("GPIB:1:15", &nHandle2, &nStatus);
```

### See Also

**CalDmmGetError**

## CalDmmMeasure

---

### Purpose

Takes a measurement and return the result

### Syntax

**CalDmmMeasure** (*lHandle*, *pdMeasurement*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM.
<i>pdMeasurement</i>	PDOUBLE	Return the measurement
<i>plStatus</i>	PLONG	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.

### Example

The following takes a new measurement:

```
LONG    lHandle, lStatus;
DOUBLE  dMeasurement;

CalDmmMeasure (lHandle, &dMeasurement, &lStatus);
```

### See Also

**CalDmmSetRange, CalDmmGetRange, CalDmmSetFunction, CalDmmGetFunction, CalDmmGetError**

## CalDmmReset

---

### Purpose

Resets the DMM to power up state

### Syntax

**CalDmmReset** (*hHandle*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function initiates a reset.

### Example

The following resets the DMM:

```
LONG    lHandle, lStatus;  
  
CalDmmReset (lHandle, &lStatus);
```

### See Also

**CalDmmInitialize, CalDmmMeasure, CalDmmSetRange, CalDmmGetRange, CalDmmSetFunction, CalDmmGetFunction, CalDmmGetError**

## CalDmmSetFunction

---

### Purpose

Sets the currently selected function

### Syntax

**CalDmmSetFunction** (*hHandle*, *enFunction*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM
<i>enFunction</i>	enumCalDmmFunction	Return the currently set function: aDmmFunctionVDc – Voltage DC Mode aDmmFunctionIDc – Current DC Mode aDmmFunctionVAcAcCpl – Voltage AC, AC Coupled Mode aDmmFunctionIAcAcCpl – Current AC, AC Coupled Mode aDmmFunctionVAcDcCpl – Voltage AC, DC Coupled Mode aDmmFunctionIAcDcCpl – Current AC, DC Coupled Mode aDmmFunction2Wire – 2Wire Resistance Mode aDmmFunction4Wire – 4 Wire Resistance Mode aDmmFunctionContinuity – High Resolution Continuity Mode aDmmFunctionFrequency – Frequency Counter Mode aDmmFunctionDiode – Test Diode aDmmFunctionTemperature – External Temperature Mode
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example sets the currently set function to Volts DC:

```
LONG    lHandle, lStatus;

GxDmmSetFunction (lHandle, aDmmFunctionVDc, &lStatus);
```

### See Also

**CalDmmGetFunction, CalDmmSetRange, CalDmmGetRange, CalDmmGetError**



## CalDmmSetRange

---

### Purpose

Sets the currently selected function's range.

### Syntax

**CalDmmGetRange** (*hHandle*, *dRange*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM.
<i>dRange</i>	DOUBLE	Sets the currently selected function's range.
<i>plStatus</i>	PSHORT	Returned status: 0 on success, negative number on failure.

### Comments

The currently selected range will depend on the currently selected function.

### Example

The following example sets the function to Volts DC and the range to 10 Volts:

```
LONG    lHandle, lStatus;
DOUBLE dRange;

CalDmmSetFunction(lHandle, CALDMM_FUNCTION_VDC, &lStatus);
CalDmmSetRange (lHandle, 10, &lStatus);
```

### See Also

**CalDmmGetRange**, **CalDmmSetFunction**, **CalDmmGetFunction**, **CalDmmGetError**

## CalDmmSetReadingRate

---

### Purpose

Sets the DMM reading rate in terms of Number of Power Line Cycles.

### Syntax

**CalDmmSetReadingRate** (*hHandle*, *dReadingRate*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM.
<i>dReadingRate</i>	DOUBLE	DMM reading rate in terms of Number of Power Line Cycles (NPLC).
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The reading rate is expressed in terms of Number of Power Line Cycles (NPLC). The Number of Power Line Cycles (NPLC) indicates how long a signal is integrated to obtain a single measurement. Noise introduced from the power line tends to be periodic. If the A/D converter integrates for an amount of time equal to one cycle of the power line noise, then the signal components from the periodic noise can be canceled. Generally speaking, the longer a signal is integrated by the A/D converter, the more accurate the reading results.

For 60Hz power, a DMM operating at 1 NPLC can report a new reading every 16.67 mSec. For 50Hz power and 1 NPLC, a new value can be reported every 20 mSec.

The reading rate can be a fraction of a power line cycle, e.g. reading rate can be set to 0.1 Line Cycles which would represent approximately 0.00167 seconds per reading if the Power Line Frequency is 60Hz.

### Example

The following sets the DMM reading rate to 0.5 Power Line Cycles:

```
LONG    lHandle, lStatus;
```

```
CalDmmSetReadingRate (lHandle, 0.5, &lStatus);
```

### See Also

**CalDmmInitialize**, **CalDmmMeasure**, **CalDmmSetRange**, **CalDmmGetRange**, **CalDmmSetFunction**, **CalDmmGetFunction**, **CalDmmGetError**

## CalDmmSetResolution

---

### Purpose

Sets the measurement resolution.

### Syntax

**CalDmmSetResolution** (*hHandle*, *enResolution*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a DMM.
<i>enResolution</i>	enumCalDmmResolution	<p>Sets the measurement in terms of digits of resolution.</p> <ol style="list-style-type: none"> <li>4. aDmmResolution3_5_Digits – 3 and a ½ Digits of Resolution.</li> <li>5. aDmmResolution4_5_Digits – 4 and a ½ Digits of Resolution.</li> <li>6. aDmmResolution5_5_Digits – 5 and a ½ Digits of Resolution</li> <li>7. aDmmResolution6_5_Digits – 6 and a ½ Digits of Resolution</li> <li>8. aDmmResolution7_5_Digits – 7 and a ½ Digits of Resolution</li> <li>9. aDmmResolution8_5_Digits – 8 and a ½ Digits of Resolution</li> </ol>
<i>plStatus</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.

### Example

The following example sets the resolution to 5 ½ digits:

```
LONG    lHandle, lStatus;
```

```
CalDmmGetResolution (lHandle, 6. aDmmResolution5_5_Digits, &lStatus);
```

### See Also

**CalDmmGetResolution**, **CalDmmSetRange**, **CalDmmGetRange**, **CalDmmSetFunction**, **CalDmmGetFunction**, **CalDmmGetError**

## MATRIX Standard Driver

---

### CaMatrixClose

---

#### Purpose

Closes a relay in the matrix

#### Syntax

**CaMatrixClose** (*hHandle, lGroup, lRow, lCol, plStatus*)

#### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Matrix.
<i>lGroup</i>	LONG	Set the matrix group to set
<i>lRow</i>	LONG	The Row coordinate of the relay in the matrix to close
<i>lCol</i>	LONG	The Column coordinate of the relay in the matrix to close
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

#### Comments

This function closes a relay in the matrix to make a connection.

#### Example

The following example closes the relay at row 5, column 2:

```
LONG    lHandle, lStatus;
```

```
CaMatrixClose (lHandle, 5, 2, &lStatus);
```

#### See Also

**CaMatrixOpen, CaMatrixGetError**

## CalMatrixGetError

---

### Purpose

Returns a string containing information about a status code

### Syntax

**CalMatrixGetError**(*nHandle, lStatus, psError, pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Matrix
<i>lStatus</i>	LONG	Error status to decode
<i>psError</i>	PSTR	Returns the error message associated with lStatus
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the error information for **LErrorStatus**:

```
LONG    lHandle, lErrorStatus, lFunction, lStatus;

CalMatrixetGetError (lHandle, lErrorStatus, psError, &lStatus);
Printf("Error Status code %d is: %s", lErrorStatus, psError);
```

### See Also

## CalMatrixGetInfo

---

### Purpose

Returns the manufacturer and model of the Matrix

### Syntax

**CalMatrixGetInfo**(*nHandle*, *psManufacturer*, *psModel*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Matrix
<i>psManufacturer</i>	PSTR	Returns a string with the manufacturer name
<i>psModel</i>	PSTR	Returns a string with the model name
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the Matrix information:

```
LONG    lHandle, lStatus;
CHAR    psManufacturer[256];
CHAR    psModel[256];

CalMatrixGetInfo (lHandle, sManufacturer, sModel, &lStatus);
Printf("The Manufacturer is: %s and the Model is: %s", psManufacturer, psModel);
```

### See Also

**CalMatrixGetError**

## CalMatrixInitialize

---

### Purpose

Initializes the driver for the board at the specified address, which includes support for VISA resource. The function returns a handle that can be used with other CALMATRIX functions to program the Matrix.

### Syntax

**CalMatrixInitialize** (*sAddress*, *pnHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>sAddress</i>	STR	Address of Matrix.
<i>phHandle</i>	PLONG	Returned handle for Matrix access
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function verifies and establishes communication with the instrument at the specified address information. The address information can be PXI, GPIB, USB, WinSock or VISA resource string. This address string format is starting by the address type followed by a column and the address: **PXI / GPIB / WinSock / USB / Com / VISA : Address** . The specifics of the **Address** string characteristics can be found in **Writing a Standard Driver** topic in step 5. The function does not change any of the board settings.

The returned handle *phHandle* is used to identify the specified instrument with other driver functions.

### Example

The following example initializes two Matrices at PXI addresses 0x105 and 6.

```
SHORT nHandle1, nHandle2, nStatus;

CalMatrixInitilize ("PXI:0x105", &nHandle1 &nStatus);
CalMatrixInitilize ("PXI:6", &nHandle2, &nStatus);
```

### See Also

**CalMatrixGetError**

## CaMatrixOpen

---

### Purpose

Opens a relay in the matrix

### Syntax

**CalDmmGetRange** (*hHandle, lGroup, lRow, lCol, plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Matrix.
<i>lGroup</i>	LONG	Set the matrix group to set
<i>lRow</i>	LONG	The Row coordinate of the relay in the matrix to open
<i>lCol</i>	LONG	The Column coordinate of the relay in the matrix to open
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function opens a relay in the matrix to break a connection.

### Example

The following example opens the relay at row 5, column 2:

```
LONG    lHandle, lStatus;
```

```
CaMatrixOpen (lHandle, 5, 2, &lStatus);
```

### See Also

**CalMatrixClose, CalMatrixGetError**



## CalMatrixReset

---

### Purpose

Resets the matrix

### Syntax

**CalMatrixReset** (*hHandle*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Matrix.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function resets the matrix to power up state where all relays are open.

### Example

The following example resets the matrix

```
LONG    lHandle, lStatus;  
  
CalMatrixReset (lHandle, &lStatus);
```

### See Also

**CalMatrixClose**, **CalMatrixOpen**, **CalMatrixInitialize**, **CalMatrixGetError**

## PS Standard Driver

---

### CalPsGetChannelState

---

#### Purpose

Returns the on/off state of a channel

#### Syntax

**CalPsGetChannelState** (*hHandle*, *nChannel*, *pnChannelState*, *plStatus*)

#### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply.
<i>nChannel</i>	SHORT	Select which channel to retrieve (0-based)
<i>pnChannelState</i>	LONG	The Row coordinate of the relay in the matrix to close 0. CALPS_CHANNEL_STATE_ON – Turn the channel On 1. CALPS_CHANNEL_STATE_OFF – Turn the channel Off
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

#### Comments

This function gets the state of a power supply channel. The channel must be turned on in order to generate voltage and current.

#### Example

The following example gets the channel 0 state:

```
LONG    lHandle, lStatus;
SHORT   nChannelState;

CalPsGetChannelState (lHandle, 0, nChannelState, &lStatus);
```

#### See Also

**CalPsSetChannelState, CalPsGetError**

## CalPsGetCurrent

---

### Purpose

Returns the current that is being generated by a channel

### Syntax

**CalPsGetCurrent** (*hHandle*, *nChannel*, *pdCurrent*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply.
<i>nChannel</i>	SHORT	Select which channel to retrieve (0-based)
<i>pdCurrent</i>	PDOUBLE	Returns the real-time current read back for a channel in units of Amps
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function gets a channel's real-time current read back in Amps.

### Example

The following example gets the channel 0 real-time current read back:

```
LONG    lHandle, lStatus;
DOUBLE  dCurrent;

CalPsGetCurrent (lHandle, 0, &dCurrent, &lStatus);
```

### See Also

**CalPsGetCurrentLimit, CalPsSetCurrentLimit, CalPsGetError**

## CalPsGetCurrentLimit

---

### Purpose

Returns the current limit that has been set for a channel

### Syntax

**CalPsGetCurrentLimit** (*hHandle*, *nChannel*, *pdCurrentLimit*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply.
<i>nChannel</i>	SHORT	Select which channel to retrieve (0-based)
<i>pdCurrentLimit</i>	PDOUBLE	Returns the channel's current limit level in units of Amps
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function gets a channel's current limit. This setting limits the current that can be generated for a given voltage. The power supply will not exceed this current limit.

### Example

The following example gets the channel 0 current limit:

```
LONG    lHandle, lStatus;
DOUBLE  dCurrentLimit;

CalPsGetCurrentLimit (lHandle, 0, &dCurrentLimit, &lStatus);
```

### See Also

**CalPsSetCurrentLimit, CalPsGetCurrent, CalPsGetError**

## CalPsGetError

---

### Purpose

Returns a string containing information about a status code

### Syntax

**CalPsGetError**(*nHandle*, *lStatus*, *psError*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply
<i>lStatus</i>	LONG	Error status to decode
<i>psError</i>	PSTR	Returns the error message associated with lStatus
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the error information for **lErrorStatus**:

```
LONG    lHandle, lErrorStatus, lFunction, lStatus;

CalPsGetError (lHandle, lErrorStatus, psError, &lStatus);
Printf("Error Status code %d is: %s", lErrorStatus, psError);
```

### See Also

## CalPsGetInfo

---

### Purpose

Returns the manufacturer and model of the Power Supply

### Syntax

**CalPsGetInfo**(*nHandle*, *psManufacturer*, *psModel*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply
<i>psManufacturer</i>	PSTR	Returns a string with the manufacturer name
<i>psModel</i>	PSTR	Returns a string with the model name
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the Matrix information:

```
LONG    lHandle, lStatus;  
CHAR    psManufacturer[256];  
CHAR    psModel[256];  
  
CalPsGetInfo (lHandle, sManufacturer, sModel, &lStatus);  
Printf("The Manufacturer is: %s and the Model is: %s", psManufacturer, psModel);
```

### See Also

#### CalPsGetError

## CalPsGetVoltage

---

### Purpose

Returns the voltage that is being generated by a channel

### Syntax

**CalPsGetVoltage** (*hHandle*, *nChannel*, *pdVoltage*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply.
<i>nChannel</i>	SHORT	Select which channel to retrieve (0-based)
<i>pdVoltage</i>	PDOUBLE	Returns the real-time voltage read back for a channel in units of Volts
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function gets a channel's real-time voltage read back in Volts.

### Example

The following example gets the channel 0 real-time voltage read back:

```
LONG    lHandle, lStatus;
DOUBLE  dVoltage;

CalPsGetVoltage (lHandle, 0, &dVoltage, &lStatus);
```

### See Also

**CalPsSetVoltage, CalPsGetCurrent, CalPsGetCurrentLimit, CalPsSetCurrentLimit, CalPsGetError**

## CalPsInitialize

---

### Purpose

Initializes the driver for the board at the specified address, which includes support for VISA resource. The function returns a handle that can be used with other CALPS functions to program the Power Supply.

### Syntax

**CalPsInitialize** (*sAddress*, *pnHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>sAddress</i>	STR	Address of Power Supply.
<i>phHandle</i>	PLONG	Returned handle for Power Supply access
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function verifies and establishes communication with the instrument at the specified address information. The address information can be PXI, GPIB, USB, WinSock or VISA resource string. This address string format is starting by the address type followed by a column and the address: **PXI / GPIB / WinSock / USB / Com / VISA : Address** . The specifics of the **Address** string characteristics can be found in **Writing a Standard Driver** topic in step 5. The function does not change any of the board settings.

The returned handle *phHandle* is used to identify the specified instrument with other driver functions.

### Example

The following example initializes two Power Supplies at address 0x105 and 7.

```
SHORT nHandle1, nHandle2, nStatus;

CalPsInitilize ("PXI:0x105", &nHandle1 &nStatus);
CalPsInitilize ("PXI:7", &nHandle2, &nStatus);
```

### See Also

**CalPsReset**, **CalPsGetError**



## CaPsReset

---

### Purpose

Resets the Power Supply

### Syntax

**CalPsReset** (*hHandle*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function resets the power supply to power up state where all relays are off and all Current Limits and Voltages are set to 0.

### Example

The following example resets the power supply

```
LONG    lHandle, lStatus;  
  
CalPsReset (lHandle, &lStatus);
```

### See Also

**CalPsInitialize**, **CalPsGetError**

## CalPsSetChannelState

---

### Purpose

Sets the on/off state of a channel

### Syntax

**CalPsSetChannelState** (*hHandle*, *nChannel*, *nChannelState*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply.
<i>nChannel</i>	SHORT	Select which channel to set (0-based)
<i>nChannelState</i>	LONG	The Channel state of change 0. CALPS_CHANNEL_STATE_ON – Turn the channel On 1. CALPS_CHANNEL_STATE_OFF – Turn the channel Off
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets the state of a power supply channel. The channel must be turned on in order to generate voltage and current.

### Example

The following example sets the channel 0 state to On:

```
LONG    lHandle, lStatus;
```

```
CalPsSetChannelState (lHandle, 0, CALPS_CHANNEL_STATE_ON, &lStatus);
```

### See Also

**CalPsGetChannelState**, **CalPsGetError**

## CalPsSetCurrentLimit

---

### Purpose

Sets the current limit that has been set for a channel

### Syntax

**CalPsSetCurrentLimit** (*hHandle*, *nChannel*, *dCurrentLimit*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply.
<i>nChannel</i>	SHORT	Select which channel to retrieve (0-based)
<i>dCurrentLimit</i>	DOUBLE	Sets the channel's current limit level in units of Amps
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets a channel's current limit. This setting limits the current that can be generated for a given voltage. The power supply will not exceed this current limit.

### Example

The following example sets the channel 0 current limit to 5 Amps:

```
LONG    lHandle, lStatus;
```

```
CalPsGetCurrentLimit (lHandle, 0, 5.0, &lStatus);
```

### See Also

**CalPsGetCurrentLimit**, **CalPsGetCurrent**, **CalPsGetError**

## CalPsSetVoltage

---

### Purpose

Sets the voltage that is being generated by a channel

### Syntax

**CalPsSetVoltage** (*hHandle*, *nChannel*, *dVoltage*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Power Supply.
<i>nChannel</i>	SHORT	Select which channel to retrieve (0-based)
<i>dVoltage</i>	DOUBLE	Sets the voltage setting in units of Volts
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets a channel's voltage in units of Volts.

### Example

The following example sets the channel 0 voltage to 18 Volts:

```
LONG    lHandle, lStatus;
```

```
CalPsGetVoltage (lHandle, 0, 18.0, &lStatus);
```

### See Also

**CalPsGetVoltage, CalPsGetCurrent, CalPsGetCurrentLimit, CalPsSetCurrentLimit, CalPsGetError**

## PWRMETER Standard Driver

---

### CalPwrMeterClose

---

#### Purpose

Closes the connection to the power meter

#### Syntax

**CalPwrMeterClose** ()

#### Comments

This function gets the state of a power supply channel. The channel must be turned on in order to generate voltage and current.

#### Example

The following example closes the session to the power meter:

```
CalPwrMeterClose (lHandle, 0, nChannelState, &lStatus);
```

#### See Also

**CalPwrMeterInitialize**

## CalPwrMeterGetError

---

### Purpose

Returns the error message from power meter.

### Syntax

**CalPwrMeterGetError** (*lStatus*,*psError*,*plStatus*)

### Parameters

Name	Type	Comments
<i>lStatus</i>	LONG	Status of the power meter.
<i>psError</i>	PSTR	Error String returned from the power meter.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function gets current error string corresponding to the error status.

### Example

The following example gets the channel 0 state:

```
LONG    lError, lStatus;  
CHAR    sError[256];  
  
CalPwrMeterGetError (lError, &sError, &lStatus);
```

### See Also

**CalPwrMeterInitialize**

## CalPwrMeterGetInfo

---

### Purpose

Returns the manufacturer and model of the Power Meter

### Syntax

**CalPwrMeterGetInfo**(*psManufacturer*, *psModel*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>psManufacturer</i>	PSTR	Returns a string with the manufacturer name
<i>psModel</i>	PSTR	Returns a string with the model name
<i>pnStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the Matrix information:

```
LONG    lStatus;  
CHAR    sManufacturer[256];  
CHAR    sModel[256];  
  
CalPwrMeterGetInfo (sManufacturer, sModel, &lStatus);  
Printf("The Manufacturer is: %s and the Model is: %s", sManufacturer, sModel);
```

### See Also

### CalPwrMeterGetError

## CalPwrMeterInitialize

---

### Purpose

Initializes the driver for the instrument at the specified address or VISA resource.

### Syntax

**CalPwrMeterInitialize** (*lAddress*, *sVisaResource*, *plStatus*)

### Parameters

Name	Type	Comments
<i>lAddress</i>	LONG	Address of Power Supply.
<i>sVisaResource</i>	STR	VISA resource of Power Supply (optional)
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function verifies and establishes communication with the instrument at the specified address information. The address information can be PXI, GPIB, USB, WinSock or VISA resource string. This address string format is starting by the address type followed by a column and the address: **PXI / GPIB / WinSock / USB / Com / VISA : Address** . The specifics of the **Address** string characteristics can be found in **Writing a Standard Driver** topic in step 5. The function does not change any of the board settings.

The returned handle *phHandle* is used to identify the specified instrument with other driver functions.

### Example

The following example initializes two Power Meters with VISA resource strings:

USB0::0x2A8D::0x8018::MY55160016::INSTR and USB0::0x2A8D::0x8018::MY55160018::INSTR.

```
SHORT lStatus;
```

```
CalPwrMeterInitilize ("VISA: USB0::0x2A8D::0x8018::MY55160016::INSTR", , &lStatus);
```

```
CalPwrMeterInitilize ("VISA:USB0::0x2A8D::0x8018::MY55160018::INSTR", , &nHandle2, &lStatus);
```

### See Also

**CalPwrMeterReset**, **CalPwrMeterGetError**



## CalPwrMeterMeasure

---

### Purpose

Returns a single measurement based on the specified measurement type.

### Syntax

**CalPwrMeterMeasure** (*nChannel*, *lTimeout*, *pdResult*, *plStatus*)

### Parameters

Name	Type	Comments
<i>nChannel</i>	SHORT	Select channel (1-based)
<i>lTimeout</i>	LONG	Measurement timeout.
<i>pdResult</i>	PDOUBLE	Returned measurement value.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function captures waveforms and returns a single measurement based on the specified measurement type.

### Example

The following example read a single AC RMS measurement from channel 0, while reapplying measurement setup to the input signal:

```
LONG    lHandle, lStatus;
DOUBLE dMeasurement;

CalPwrMeterMeasure (1, 1000, &dMeasurement, &lStatus);
```

### See Also

**CalPwrMeterSetupMeasurement**, **CalPwrMeterGetInfo**, **CalPwrMeterReset**, **CalPwrMeterZero**, **CalPwrMeterGetError**

## CaPwrMeterReset

---

### Purpose

Resets the Power Meter

### Syntax

**CalPwrMeterReset** (*plStatus*)

### Parameters

Name	Type	Comments
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function resets the power meter to power up state.

### Example

The following example resets the power meter

```
LONG    lHandle, lStatus;  
  
CalPwrMeterReset (&lStatus);
```

### See Also

**CalPwrMeterInitialize**, **CalPwrMeterGetError**

## CalPwrMeterSetupMeasurement

---

### Purpose

Sets the power meter measurement settings.

### Syntax

**CalPwrMeterSetupMeasurement** (*nChannel*, *dFrequency*, *enPwrMeterMeasureType*, *lAverageCount*, *plStatus*)

### Parameters

Name	Type	Comments
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>enPwrMeterMeasureType</i>	LONG	The Row coordinate of the relay in the matrix to close <ol style="list-style-type: none"> <li>0. CALPWRMETER_MEASURE_AUTO – Set Measure to Auto.</li> <li>1. CALPWRMETER_MEASURE_AVERAGE – Set Measre to Average.</li> <li>2. CALPWRMETER_MEASURE_IMMEDIATE – Set Measre to Immediate.</li> </ol>
<i>lAverageCount</i>	LONG	Specify the average count for repetitive measurement modes. In Average mode, this specifies the number of measurements to be averaged before the measurement is complete.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets a channel's measurement mode.

### Example

The following example sets channel 1 measurement settings with average count of 256:

```
LONG    lStatus;
```

```
CalPwrMeterSetupMeasurement (1, CALPWRMETER_MEASURE_AVERAGE, 256, &lStatus);
```

### See Also

**CalPwrMeterGetInfo**, **CalPwrMeterMeasure**, **CalPwrMeterReset**, **CalPwrMeterGetError**

## CalPwrMeterZero

---

### Purpose

Zeroes the instrument at a specific channel.

### Syntax

**CalPwrMeterMeasure** (*nChannel*, *plStatus*)

### Parameters

Name	Type	Comments
<i>nChannel</i>	SHORT	Select channel (1-based)
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function zeroes the instrument at a specific channel. The channel should be disconnected before zeroeing..

### Example

The following example zeroes channel 1 of the instrument.

```
LONG    lStatus;
```

```
CalPwrMeterZero (1 &lStatus);
```

### See Also

**CalPwrMeterGetInfo, CalPwrMeterReset, CalPwrMeterGetError**

## SCOPE Standard Driver

---

### CalScopeGetError

---

#### Purpose

Returns a string containing information about a status code

#### Syntax

**CalScopeGetError**(*nHandle*, *lStatus*, *psError*, *pnStatus*)

#### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Counter
<i>lStatus</i>	LONG	Error status to decode
<i>psError</i>	PSTR	Returns the error message associated with lStatus
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

#### Comments

#### Example

The following example gets the error information for **LErrorStatus**:

```
LONG    lHandle, lErrorStatus, lFunction, lStatus;  
  
CalScopeGetError (lHandle, lErrorStatus, psError, &lStatus);  
Printf("Error Status code %d is: %s", lErrorStatus, psError);
```

#### See Also

## CalScopeGetInfo

---

### Purpose

Returns the manufacturer and model of the Oscilloscope

### Syntax

**CalScopeGetInfo**(*nHandle*, *psManufacturer*, *psModel*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope
<i>psManufacturer</i>	PSTR	Returns a string with the manufaturer name
<i>psModel</i>	PSTR	Returns a string with the model name
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the Matrix information:

```
LONG    lHandle, lStatus;
CHAR    psManufacturer[256];
CHAR    psModel[256];

CalScopeGetInfo (lHandle, sManufacturer, sModel, &lStatus);
Printf("The Manufacturer is: %s and the Model is: %s", psManufacturer, psModel);
```

### See Also

#### CalScopeGetError

## CalScopeGetInputImpedance

---

### Purpose

Returns the specified channel's input impedance.

### Syntax

**CalScopeSetupInputImpedance** (*hHandle*, *nChannel*, *plInputImpedance*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>plInputImpedance</i>	PLONG	Specified channel's input impedance: 0= Input Impedance is 1MOhms 1= Input Impedance is 50Ohms
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

Returns the specified channel's input impedance.

### Example

The following example returns channel 0 input impedance:

```
LONG    lHandle, lStatus, lInputImpedance;
```

```
CalScopeGetInputImpedance (lHandle, 0, &lInputImpedance, &lStatus);
```

### See Also

**CalScopeSetupAuto, CalScopeGetInfo, CalScopeMeasure, CalScopeReset, CalScopeSetupAcquisitio  
CalScopeSetupChannel, CalScopeGetError**

## CalScopeInitialize

---

### Purpose

Initializes the driver for the board at the specified address, which includes support for VISA resource. The function returns a handle that can be used with other CalScopeXXXX functions to program the scope.

### Syntax

**CalScopeInitialize** (*sAddress*, *sVisaResource*, *pnHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>sAddress</i>	STR	Address of Oscilloscope.
<i>phHandle</i>	PLONG	Returned handle for Counter access
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function verifies and establishes communication with the instrument at the specified address information. The address information can be PXI, GPIB, USB, WinSock or VISA resource string. This address string format is starting by the address type followed by a column and the address: **PXI / GPIB / WinSock / USB / Com / VISA : Address** . The specifics of the **Address** string characteristics can be found in **Writing a Standard Driver** topic in step 5. The function does not change any of the board settings.

The returned handle *phHandle* is used to identify the specified instrument with other driver functions.

### Example

The following example initializes two scopes at PXI address 0x105 and GPIB board 1, address 15 .

```
SHORT nHandle1, nHandle2, nStatus;
```

```
CalScopeInitilize ("PXI:0x105", &nHandle1 &nStatus);
CalScopeInitilize ("GPIB:1:15", &nHandle2, &nStatus);
```

### See Also

**CalScopeReset**, **CalScopeGetError**



## CalScopeMeasure

---

### Purpose

Returns a single measurement based on the specified measurement type.

### Syntax

**CalScopeMeasure** (*hHandle*, *nChannel*, *iType*, *bApplySetup*, *dTimeout*, *pdMeasurement*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>iType</i>	LONG	Set the type of measurement: 0=aScopeMeasureAcRms: Returns the captured waveform AC RMS. 1=aScopeMeasurePeakToPeak: Returns the captured waveform Peak-To-Peak. 2=aScopeMeasureAmplitude: Returns the captured waveform Amplitude
<i>bApplySetup</i>	LONG	0= The function will use current measutmrtns setup. 1, The function will apply all necessary measurement setup to the current captured waveform for optimal measurement.
<i>dTimeout</i>	DOUBLE	Measurement timeout.
<i>pdMeasurement</i>	PDOUBLE	Returned measurement value.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function captures waveforms and returns a single measurement based on the specified measurement type.

### Example

The following example read a single AC RMS measurement from channel 0, while reapplying measurement setup to the input signal:

```
LONG    lHandle, lStatus;
DOUBLE dMeasurement;

CalScopeMeasure (lHandle, 0, 0, 1, 30, &dMeasurement, &lStatus);
```

### See Also

**CalScopeSetupAuto**, **CalScopeGetInfo**, **CalScopeReset**, **CalScopeSetupAcquisitio** **CalScopeSetupChannel**, **CalScopeSetupHorizontal**, **CalScopeGetError**

## CalScopeReset

---

### Purpose

Performs a hardware reset and sets all scope settings to default parameters.

### Syntax

**CalScopeReset** (*hHandle*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function resets the Oscilloscope to power up state.

### Example

The following example resets the Oscilloscope

```
LONG    lHandle, lStatus;  
  
CalScopeReset (lHandle, &lStatus);
```

### See Also

**CalScopeSetupAuto, CalScopeGetInfo, CalScopeMeasure, CalScopeSetupAcquisitio CalScopeSetupChannel, CalScopeSetupHorizontal, CalScopeGetError**

## CalScopeSetupAcquisition

---

### Purpose

Sets the board acquisition settings.

### Syntax

**CalScopeSetupAcquisition** (*hHandle*, *nChannel*, *lAcquireType*, *lAcquireCount*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>lAcquireType</i>	LONG	Select the type of acquisition.
<i>lAcquireCount</i>	LONG	Specify the acquisition count for repetitive acquisition modes. In Average mode, this specifies the number of waveforms to be averaged before the acquisition is complete.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets a channel's acquisition mode.

### Example

The following example sets channel 0 acquisition settings with count of 256:

```
LONG    lHandle, lStatus;

CalScopeSetupAcquisition (lHandle, 0, 0, 256, &lStatus);
```

### See Also

**CalScopeSetupAuto**, **CalScopeGetInfo**, **CalScopeMeasure**, **CalScopeReset**, **CalScopeSetupChannel**, **CalScopeSetupHorizontal**, **CalScopeGetError**

## CalScopeSetupAuto

---

### Purpose

Automatically sets the range, offset, coupling, and sample rate based on the input signal.

### Syntax

**CalScopeSetupAuto** (*hHandle*, *nChannel*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example automatically sets the range, offset, coupling, and sample rate based on the input signal:

```
LONG    lHandle, lStatus;  
  
CalScopeSetupAuto (lHandle, 0, &lStatus);
```

### See Also

**CalScopeGetInfo**, **CalScopeMeasure**, **CalScopeReset**, **CalScopeSetupAcquisitio** **CalScopeSetupChannel**, **CalScopeSetupHorizontal**, **CalScopeGetError**

## CalScopeSetupChannel

---

### Purpose

Configures the selected channel settings.

### Syntax

**CalScopeSetupChannel** (*hHandle*, *nChannel*, *dAmplitude*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>dAmplitude</i>	DOUBLE	Set the full scale acquisition range in volts for the specified input channel.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function configures the selected channel settings including the input amplitude.

### Example

The following example sets channel 0 with 10V peak-to-peak:

```
LONG    lHandle, lStatus;
```

```
CalScopeSetupChannel (lHandle, 0, 10, &lStatus);
```

### See Also

**CalScopeSetupAuto**, **CalScopeGetInfo**, **CalScopeMeasure**, **CalScopeReset**, **CalScopeSetupAcquisitio**  
**CalScopeSetupHorizontal**, **CalScopeGetError**

## CalScopeSetupChannelState

---

### Purpose

Sets the selected channel state.

### Syntax

**CalScopeSetupChannelState** (*hHandle*, *nChannel*, *lState*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>lState</i>	LONG	Set the channel's state: 0= Off 1=On
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example sets channel 0 state to ON:

```
LONG    lHandle, lStatus;
```

```
CalScopeSetupChannelState (lHandle, 0, 1, &lStatus);
```

### See Also

**CalScopeSetupAuto, CalScopeGetInfo, CalScopeMeasure, CalScopeReset, CalScopeSetupAcquisitio  
CalScopeSetupHorizontal, CalScopeGetError**

## CalScopeSetupHorizontal

---

### Purpose

Configure the horizontal and time base settings.

### Syntax

**CalScopeSetupHorizontal** (*hHandle*, *nChannel*, *dFrequency* *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>dFrequency</i>	DOUBLE	Expected waveform's frequency.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function configures the horizontal and time base settings based of the expected waveform's frequency.

### Example

The following example sets channel 0 horizontal and time base settings for a 1MHz waveform:

```
LONG    lHandle, lStatus;

CalScopeSetupHorizontal (lHandle, 0, 1e6, &lStatus);
```

### See Also

**CalScopeSetupAuto**, **CalScopeGetInfo**, **CalScopeMeasure**, **CalScopeReset**, **CalScopeSetupAcquisitio**  
**CalScopeSetupChannel**, **CalScopeGetError**

## CalScopeSetupInputImpedance

---

### Purpose

Sets the specified channel's input impedance.

### Syntax

**CalScopeSetupInputImpedance** (*hHandle*, *nChannel*, *lInputImpedance*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Oscilloscope.
<i>nChannel</i>	SHORT	Select channel (0-based)
<i>lInputImpedance</i>	LONG	Specified channel's input impedance: 0= Input Impedance is 1MOhms 1= Input Impedance is 50Ohms
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

T Sets the specified channel's input impedance.

### Example

The following example sets channel 0 input impedance to 50Ohms:

```
LONG    lHandle, lStatus;
```

```
CalScopeSetupInputImpedance (lHandle, 0, 1, &lStatus);
```

### See Also

**CalScopeSetupAuto, CalScopeGetInfo, CalScopeMeasure, CalScopeReset, CalScopeSetupAcquisitio  
CalScopeSetupChannel, CalScopeGetError**



## REF Standard Driver

---

### CalRefGetCalibrationStatus

---

#### Purpose

Returns status of the calibrator's calibration at the specified calibration program duration.

#### Syntax

**CalRefGetCalibrationStatus** (*nHandle*, *dwCalibrationProgramDuration*, *pbValidCalibration*, *pnStatus*)

#### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference
<i>dwCalibrationProgramDuration</i>	DWORD	Calibration Program Duration in seconds.
<i>pbValidCalibration</i>	PBOOL	Returns True if the calibrator is calibrated.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

#### Comments

The function returns TRUE if while running the calibration program (duration specified by the *dwCalibrationProgramDuration* parameter) the Calibrator's calibration is valid.

If the function returns FALSE, the user needs to run the Calibrator's self-calibration routines from its front panel.

#### Example

The following example returns the status of the calibrator's calibration in the next 2 hours:

```
LONG    lHandle, lStatus;
BOOL    bValidCalibration;

CalRefGetCalibrationStatus (lHandle, 7200, &bValidCalibration &lStatus);
```

#### See Also

**CalRefGetRequiredCalibration**

## CalRefGetError

---

### Purpose

Returns a string containing information about a status code

### Syntax

**CalRefGetError**(*nHandle*, *lStatus*, *psError*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference
<i>lStatus</i>	LONG	Error status to decode
<i>psError</i>	PSTR	Returns the error message associated with lStatus
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the error information for **lErrorStatus**:

```
LONG    lHandle, lErrorStatus, lFunction, lStatus;

CalRefGetError (lHandle, lErrorStatus, psError, &lStatus);
Printf("Error Status code %d is: %s", lErrorStatus, psError);
```

### See Also

## CalRefGetInfo

---

### Purpose

Returns the manufacturer and model of the Reference

### Syntax

**CalRefGetInfo**(*nHandle*, *psManufacturer*, *psModel*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference
<i>psManufacturer</i>	PSTR	Returns a string with the manufacturer name
<i>psModel</i>	PSTR	Returns a string with the model name
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

### Example

The following example gets the Reference information:

```
LONG    lHandle, lStatus;
CHAR    psManufacturer[256];
CHAR    psModel[256];

CalRefGetInfo (lHandle, sManufacturer, sModel, &lStatus);
Printf("The Manufacturer is: %s and the Model is: %s", psManufacturer, psModel);
```

### See Also

**CalRefGetError**

## CalRefGetRequiredCalibration

---

### Purpose

Returns the specified required calibration status.

### Syntax

**CalRefGetRequiredCalibration** (*nHandle*, *enCalRefRequiredCalibration*, *pbValidCalibration*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference
<i>enCalRefRequiredCalibration</i>	enumCalRefRequiredCalibration	Required Calibration constants, see comments for details aRefRequiredCalibrationInstrument aRefRequiredCalibrationZero aRefRequiredCalibrationZeroOhms aRefRequiredCalibrationScope
<i>pbValidCalibration</i>	PBOOL	Returns True if the calibrator is calibrated.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

Required Calibration constants:

aRefRequiredCalibrationInstrument – query if the calibrator is due for its annuly calibration.

aRefRequiredCalibrationZero - query if the calibrator is due for its internal zero calibration, this calibration is done through the calibrator's front panel.

aRefRequiredCalibrationZeroOhms - query if the calibrator is due for its internal zero ohms calibration, this calibration is done through the calibrator's front panel.

aRefRequiredCalibrationScope - query if the calibrator is due for its internal scope calibration, this calibration is done through the calibrator's front panel.

### Example

The following example returns if calibrator is due for zero ohms calibration:

```
LONG    lHandle, lStatus;
BOOL    bValidCalibration;
CalRefGetRequiredCalibration (lHandle, aRefRequiredCalibrationZeroOhms, &bValidCalibration,
&lStatus);
```

### See Also

**CalRefGetCalibrationStatus**

## CalRefInitialize

---

### Purpose

Initializes the driver for the board at the specified address, which includes support for VISA resource. The function returns a handle that can be used with other CALREF functions to program the Power Supply.

### Syntax

**CalRefInitialize** (*sAddress*, *pnHandle*, *pnStatus*)

### Parameters

Name	Type	Comments
<i>sAddress</i>	STR	Address of Reference.
<i>phHandle</i>	PLONG	Returned handle for Matrix access
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

The function verifies and establishes communication with the instrument at the specified address information. The address information can be PXI, GPIB, USB, WinSock or VISA resource string. This address string format is starting by the address type followed by a column and the address: **PXI / GPIB / WinSock / USB / Com / VISA : Address** . The specifics of the **Address** string characteristics can be found in **Writing a Standard Driver** topic in step 5. The function does not change any of the board settings.

The returned handle *phHandle* is used to identify the specified instrument with other driver functions.

### Example

The following example initializes a Reference at GPIB board 1 address 15 and VISA resource USB0::1234::5678.

```
SHORT nHandle1, nHandle2, nStatus;
```

```
CalRefInititalize ("GPIB:1:15", &nHandle1 &nStatus);
```

```
CalRefInititalize ("VISA:USB0::1234::5678", &nHandle1 &nStatus);
```

### See Also

**CalRefReset**, **CalRefGetError**

## CaRefReset

---

### Purpose

Resets the Reference

### Syntax

**CalRefReset** (*hHandle*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference.
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function resets the reference to power up state where output is on standby.

### Example

The following example resets the reference

```
LONG    lHandle, lStatus;
```

```
CalRefReset (lHandle, &lStatus);
```

### See Also

**CalRefInitialize**, **CalRefGetError**

## CalRefSet2Wire

---

### Purpose

Sets the output to 2Wire Resistance Mode

### Syntax

**CalRefSet2Wire** (*hHandle*, *dwResistance*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference.
<i>dwResistance</i>	DWORD	Set the Ohms of Resistance
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets the Reference output function to 2Wire and sets the corresponding amount of resistance.

### Example

The following example set the Reference to output 1000 Ohms in 2Wire mode:

```
LONG    lHandle, lStatus;
```

```
CalRefSet2Wire (lHandle, 1000, &lStatus);
```

### See Also

**CalRefInitialize**, **CalRefSet4Wire** **CalRefGetError**

## CaRefSet4Wire

---

### Purpose

Resets the output to 4Wire Resistance Mode

### Syntax

**CalRefSet2Wire** (*hHandle*, *dwResistance*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference.
<i>dwResistance</i>	DWORD	Set the Ohms of Resistance
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets the Reference output function to 4Wire and sets the corresponding amount of resistance.

### Example

The following example set the Reference to output 1000 Ohms in 4Wire mode:

```
LONG    lHandle, lStatus;  
  
CalRefSet4Wire (lHandle, 1000, &lStatus);
```

### See Also

**CalRefInitialize, CalRefSet2Wire CalRefGetError**



## CalRefSetACCurrent

---

### Purpose

Sets the output to AC Current Mode

### Syntax

**CalRefSetACCurrent** (*hHandle*, *dCurrent*, *dFrequency*., *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference.
<i>dCurrent</i>	DOUBLE	Set the current in Amps
<i>dFrequency</i>	DOUBLE	Sets the current frequency in Hertz
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets the Reference output function to AC Current and sets the corresponding amount of current and frequency.

### Example

The following example set the Reference output function to AC Current and sets the corresponding amount of current to 0.5 Amps at 400 Hertz.

```
LONG    lHandle, lStatus;
```

```
CalRefSetACCurrent (lHandle, 0.5, 400, &lStatus);
```

### See Also

**CalRefInitialize**, **CalRefSetACVoltage**, **CalRefGetError**

## CalRefSetACVolts

---

### Purpose

Sets the output to AC Voltage Mode

### Syntax

**CalRefSetACVolts** (*hHandle*, *dVoltage*, *dFrequency*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference.
<i>dVoltage</i>	DOUBLE	Set the voltage in Volts
<i>dFrequency</i>	DOUBLE	Sets the voltage frequency in Hertz
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets the Reference output function to AC Voltage and sets the corresponding amount of voltage and frequency.

### Example

The following example set the Reference output function to AC Voltage and sets the corresponding amount of voltage to 15 Volts at 2000 Hertz.

```
LONG    lHandle, lStatus;
```

```
CalRefSetACCurrent (lHandle, 15, 2000, &lStatus);
```

### See Also

**CalRefInitialize, CalRefSetACCurrent, CalRefGetError**

## CalRefSetDCCurrent

---

### Purpose

Sets the output to DC Current Mode

### Syntax

**CalRefSetDCCurrent** (*hHandle*, *dCurrent*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference.
<i>dCurrent</i>	DOUBLE	Set the current in Amps
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets the Reference output function to DC Current and sets the corresponding amount of current.

### Example

The following example set the Reference output function to DC Current and sets the corresponding amount of current to 0.5 Amps.

```
LONG    lHandle, lStatus;
```

```
CalRefSetDCCurrent (lHandle, 0.5, &lStatus);
```

### See Also

**CalRefInitialize, CalRefSetDCVoltage, CalRefGetError**

## CalRefSetDCVolts

---

### Purpose

Sets the output to DC Voltage Mode

### Syntax

**CalRefSetDCVolts** (*hHandle*, *dVoltage*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference.
<i>dVoltage</i>	DOUBLE	Set the voltage in Volts
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets the Reference output function to DC Voltage and sets the corresponding amount of voltage.

### Example

The following example set the Reference output function to DC Voltage and sets the corresponding amount of voltage to 15 Volts.

```
LONG    lHandle, lStatus;  
  
CalRefSetDCCurrent (lHandle, 15, &lStatus);
```

### See Also

**CalRefInitialize**, **CalRefSetDCCurrent**, **CalRefGetError**

## CalRefSetState

---

### Purpose

Sets the output state

### Syntax

**CalRefSetState** (*hHandle*, *bOperate*, *plStatus*)

### Parameters

Name	Type	Comments
<i>hHandle</i>	LONG	Handle for a Reference.
<i>bOperate</i>	BOOL	Set operational state of the Reference
<i>plStatus</i>	PLONG	Returned status: 0 on success, negative number on failure.

### Comments

This function sets the Reference operation state. When this function is called with *bOperate* set to FALSE, the Reference is set to standby mode. If *bOperate* is set to TRUE, the Reference is operating.

### Example

The following example set the Reference operate state Standby:

```
LONG    lHandle, lStatus;
```

```
CalRefSetState (lHandle, False, &lStatus);
```

### See Also

**CalRefInitialize**, **CalRefGetError**



# Index

## A

Address Specification .....	55
Applications.....	5
Architecture .....	1, 4
ATEasy5, 7, 8, 9, 29, 31, 33, 35, 37, 40, 42, 44, 47, 49	
ATEasy.EXE .....	7
Auto Print .....	27, 28
Auto Save .....	27, 28

## B

Block Diagram.....	4
Board Description.....	1

## C

CalCounterGetError.....	60
CalCounterGetInfo .....	61
CalCounterInitialize.....	62
CalCounterMeasure .....	63
CalCounterReset.....	64
CalCounterSetAcquisitionMode.....	65
CalCounterSetDigitalFilter .....	66
CalCounterSetFunctionPulseWidth.....	67
CalCounterSetGateTime.....	68
CalCounterSetImpedance .....	69
CalCounterSetSlope.....	70
CalCounterSetTriggerLevel.....	71
CalDmmGetError .....	72
CalDmmGetFunction.....	73
CalDmmGetInfo .....	74
CalDmmInitialize .....	75
CalDmmMeasure.....	76
CalDmmReset.....	77
CalDmmSetFunction .....	78
CalDmmSetRange .....	79
CalDmmSetReadingRate .....	80
CalDmmSetResolution .....	81

CalEasy Files .....	9
CalEasy.exe .....	7
Calibrate Always .....	6
Calibration Certificates.....	22
Calibration Harness Connections30, 32, 34, 37, 41, 43, 45, 48, 50	
Calibration Procedure30, 32, 34, 35, 39, 41, 43, 46, 48, 51	
Calibration Run Mode .....	5
CalMatrixClose.....	82
CalMatrixGetError.....	83
CalMatrixGetInfo .....	84
CalMatrixInitialize.....	85
CalMatrixOpen .....	86
CalMatrixReset .....	87
CalPsGetChannelState .....	88
CalPsGetCurrent.....	89
CalPsGetCurrentLimit .....	90
CalPsGetError.....	91
CalPsGetInfo .....	92
CalPsGetVoltage .....	93
CalPsInitialize.....	94
CalPsReset .....	95
CalPsSetChannelState .....	96
CalPsSetCurrentLimit.....	97
CalPsSetVoltage .....	98
CalPwrMeterClose.....	99
CalPwrMeterGetError .....	100
CalPwrMeterGetInfo .....	101
CalPwrMeterInitialize .....	102
CalPwrMeterMeasure .....	103
CalPwrMeterReset.....	104
CalPwrMeterSetupMeasurement .....	105
CalPwrMeterZero .....	106
CalRefGetCalibrationStatus .....	119
CalRefGetError.....	120

CalRefGetInfo .....	121	DMM Standard .....	9
CalRefGetRequiredCalibration.....	122	DMM Standard Driver.....	72
CalRefInitialize.....	123	<b>F</b>	
CalRefReset .....	124	Features.....	3
CalRefSet2Wire.....	125	Files Description.....	9
CalRefSet4Wire.....	126	Function Reference Summary .....	57
CalRefSetACCurrent .....	127	<b>G</b>	
CalRefSetACVolts.....	128	General Options.....	26
CalRefSetDCCurrent .....	129	GX1110 .....	29
CalRefSetDCVolts.....	130	GX1110 Calibration .....	29
CalRefSetState.....	131	GX1120 .....	31
CalScopeGetError.....	107	GX1120 Calibration .....	31
CalScopeGetInfo .....	108	GX1649 Calibration .....	33
CalScopeGetInputImpedance .....	109	GX1838 Calibration .....	35
CalScopeInitialize.....	110	GX2065 Series Calibration.....	37
CalScopeMeasure .....	111	GX3348 Calibration .....	40
CalScopeReset .....	112	GX3788 Calibration .....	42
CalScopeSetupAcquisition .....	113	GX5055 .....	44
CalScopeSetupAuto.....	114	GX5055 Calibration .....	44
CalScopeSetupChannel .....	115	GX5295 .....	47
CalScopeSetupChannelState.....	116	GX5295 Calibration .....	47
CalScopeSetupHorizontal.....	117	GX5954 .....	37
CalScopeSetupInputImpedance .....	118	GX5960 Series Calibration.....	49
Certificate .....	6	GX5961 .....	49
Certificate Options.....	28	GX5964 .....	49
certificate template .....	28	GX95962 .....	37, 44, 45, 49, 50
Company Logo .....	26	GX95962-1 .....	47, 48
Copyright.....	i	GX95963 .....	37, 44, 45, 49, 50, 51
Corrupt files.....	8	<b>H</b>	
COUNTER .....	29, 31	Help .....	i
COUNTER Standard .....	9	HW .....	7, 8
COUNTER Standard Driver.....	60	<b>I</b>	
<b>D</b>		Installation Directories .....	8
Device Information Window .....	23	Installing.....	7
Directories .....	8	Interchangeability .....	5
Disclaimer.....	i	<b>L</b>	
DMM .. 5, 29, 31, 33, 35, 40, 42, 44, 47, 49		License.....	15, 16



Log Options .....	27	SCOPE.....	33, 40
<b>M</b>		SCOPE Standard.....	12
Main Window .....	17	SCOPE Standard Driver .....	107
MATRIX .....	5, 42, 49	Setup .....	7, 8, 9
MATRIX Standard .....	11	Setup Maintenance .....	8
MATRIX Standard Driver .....	82	Standard Drivers .....	53
Menu and Toolbar Commands .....	18	Standards .....	5
<b>O</b>		Standards / Instruments Window .....	25
Options .....	26	Standards DLLs Files .....	9
Overview .....	3	Starting CalEasy .....	15
<b>P</b>		SWITCH MATRIX .....	44, 47
PCI.....	8	System	
POWER METER.....	31	Directory .....	8
Program-File-Descriptions .....	9	<b>T</b>	
Programs.....	5	Technical support .....	i
PS.....	5, 44, 49	Test Log.....	20
PS Standard .....	12	Trademarks .....	ii
PS Standard Driver .....	88	<b>U</b>	
PWRMETER Standard.....	12	Using CalEasy .....	15
PWRMETER Standard Driver .....	99	<b>V</b>	
<b>R</b>		Verify.....	5
README.TXT .....	8, 9	Verify and Calibrate If Required .....	5
Records .....	6	Verify and if Passed, Update Calibration Date .....	5
<b>REF</b> .....	37	<b>W</b>	
REF Standard.....	12	Warranty .....	i
<b>S</b>		Working with CalEasy.....	19
Safety and Handling .....	i	Writing a Standard Driver .....	53

