Support Information Only

NOTICE: Signametrics was acquired by Agilent Technologies in October 2010. This document, published prior to that date, is provided as a courtesy and may contain references to products or services no longer supported by Agilent. For the latest information on Agilent's modular test and measurement products go to: **www.agilent.com/find/modular**

Or in the US, call Agilent Technologies at 1-800-829-4444 (8am-8pm EST)
For other Countries: www.agilent.com/find/contactus

Agilent Technologies

Signametrics

Operator's Manual

Model SMX2055   5-½ Digit Digital PXI Multimeter

January 2010 Driver compatibility: Rev 1.60
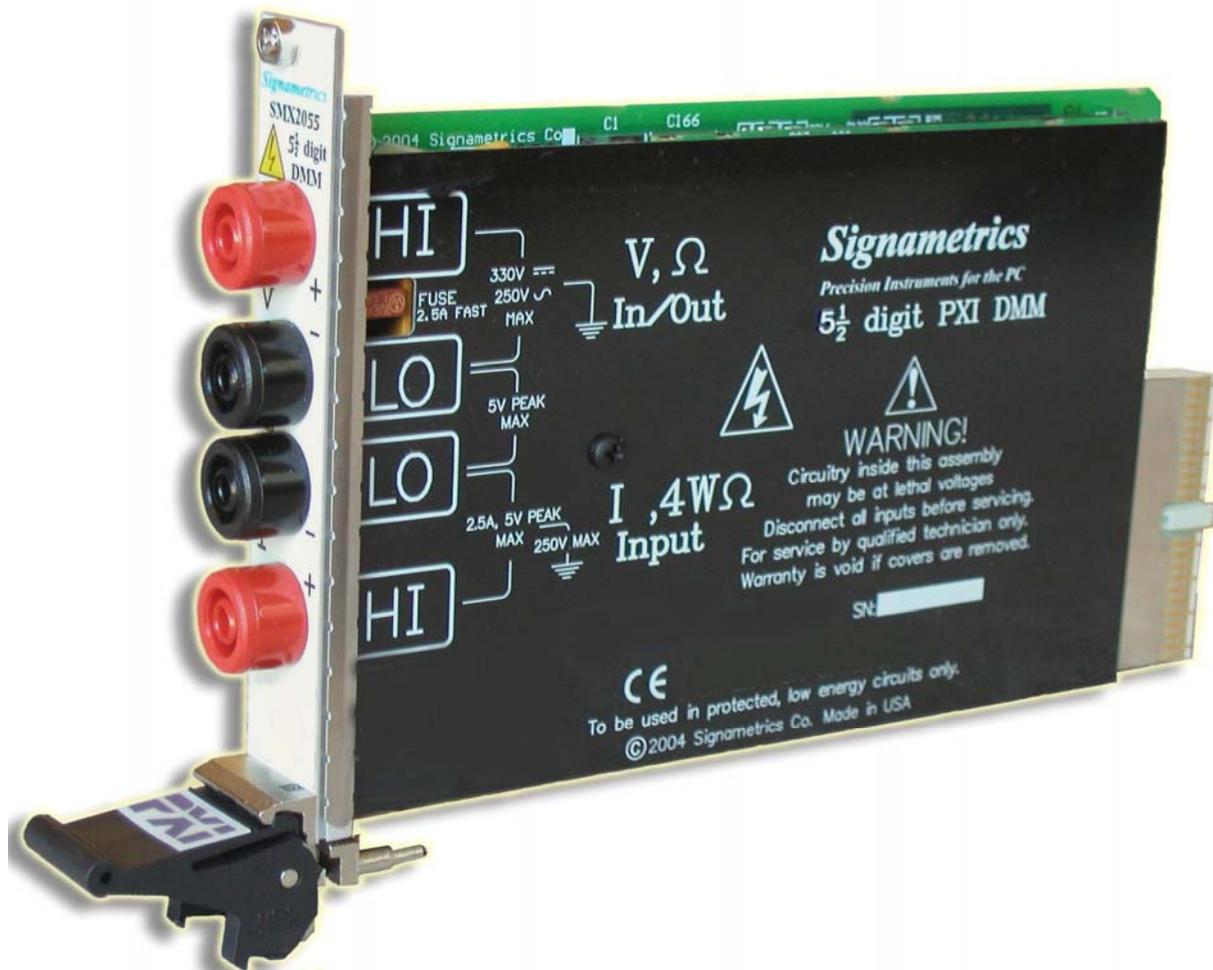
CAUTION

In no event shall Signametrics or its Representatives are liable for any consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other loss) arising out of the use of or inability to use Signametrics products, even if Signametrics has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential damages, the above limitations may not apply to you.

TABLE OF CONTENTS

1.0 Introduction

Congratulations! You have purchased a PXI/CompactPCI Plug-in Digital Multimeter. The SMX2055 Digital Multimeters (DMM's) are easy to setup and use, have sophisticated analog and digital circuitry to provide very repeatable measurements, and are protected to handle any unexpected situations your measurement environment may encounter. To get years of reliable service from these DMM's, please take a few moments and review this manual before installing and using this precision instrument.

This manual describes the SMX2055 PXI/cPCI module.

1.1 Safety Considerations

<div style="border:1px solid">

Safety Considerations

The SMX2055 DMM is capable of measuring up to 240 VDC or 240 VAC across the Volt HI and LO terminals, and can also measure common mode signals that "float" the DMM above EARTH ground by up to 300 VDC or 250 VAC. When making common mode measurements, the majority of the circuits inside the DMM are at the common mode voltage. **These voltages can be lethal and can KILL! During and after installing your DMM, check to see that there are no wires or ribbon cables from your PC trapped inside the DMM.**

The DMM comes installed with four shields (bottom, top and two edge strips) that **must not be removed for performance as well as safety reasons.**   Removal of these shields and/or improper assembly of the shields can result in lethal voltages occurring within your PC.   Be sure to check your installation before closing the cover on your personal computer.

Warning

Check to see that no loose wires or ribbon cables infringe upon any of the internal circuits of the DMM, as this may apply measurement voltages to your computer, causing electrocution and/or damage to your computer!

To avoid shock hazard, install the DMM only into a computer that has its power connector connected to a power receptacle with an earth safety ground.

When making any measurements above 50 VDC or 40 VAC, only use Safety Test Leads. Examples of these are the Signametrics Basic Test Leads and Deluxe Test Leads, offered as an accessory with the Signametrics DMM's.

</div>

1.2 Minimum Requirements
The SMX2055 DMM's are precision plug-in modules that are compatible with IBM type personal computers (PCs), PXI and cPCI chassis. It requires as a minimum a Pentiums computer. They require a half-length expansion slot on the PCI bus or 3U PXI slot. A mouse must be installed when controlling the DMM from the Windows Control Panel. The SMX2055 comes with a Windows' DLL, for operation with Windows' Version 95/98/Me/2000/XP and NT4.0.

1.3 Feature Set
The SMX2055 is a traditional 5-1/2 digit DMM and it can be used as a general purpose DMM. The High Workload Multi Function SM2064 adds timing, capacitance, inductance, sourcing and a lot more speed. With its specialized measurements, it can replace some costly instruments, shrinking the size and cost of a test system.

SMX2055 basic features:

Volts DC; for ranges, 240mV to 240V
Volts AC; Tru RMS; four ranges, 240mV to 240V
2-Wire Ohms, six ranges 240 Ω to 24 MΩ
4-Wire Ohms, five ranges 240 Ω to 2.4 MΩ
DC current, four ranges 2.4 mA to 2.4 A
AC current, four ranges 2.4 mA to 2.4 A
Diode V/I characteristics at 100 ηA to 1mA
Auto range, Relative, Min, Max, dB, %deviation operations
High Dynamic range; ±240,000 counts (240.000V)
Selectable measurement rate: 1 to 100 readings/sec

2.0 Specifications

The following specifications are based on both, verification of large number of units as well as mathematical evaluation. They should be considered under the environment specified.

It is important to note that a DMM specified range is expressed as a numeric value indicating the highest absolute measurement that can be displayed using the range. The lowest value that can be detected is expressed by the corresponding resolution for the range.

2.1 DC Voltage Measurement

Input Characteristics
- **Input Resistance 240 mV, 2.4 V Ranges:** >10 GΩ, with typical leakage of 50pA
- **Input Resistance 24 V, 240 V Ranges:** 10.00 MΩ

Range	Full Scale 5-½ Digits	Resolution	Accuracy ± (% of reading + Volts) [1] One Year 23°C ± 10°C
240 mV	240.000 mV	1 μV	0.015 + 7 μV
2.4 V	2.40000 V	10 μV	0.014 + 30 μV
24 V	24.0000 V	100 μV	0.02 + 750 μV
240 V	240.000 V	1 mV	0.02 + 3 mV

[1] With measurement rate set to 2rps or lower rate, within one hour from Zero (Relative control).

For resolution at higher measurement rates, see the following table. Use this table for DC Volts, DC current and Resistance measurements.

Maximum reading rate	Resolution	
2 / second	5-1/2 digits	19 bits
8 / second	5 digits	18 bits
50 / second	4-1/2 digits	17 bits
100 / second	4 digits	16 bits

DCV Noise Rejection   Normal Mode Rejection, at 50, 60, or 400 Hz ± 0.5%, is better than 95 dB for Rates of 6rps and lower. Common Mode Rejection (with 1 kΩ lead imbalance) is better than 120 dB for these conditions.

2.2 DC Current Measurement

Input Characteristics

- **Number of built-in shunts** Two
- **Currents greater than 2.4A** require external shunt
- **Protected** with 2.5A Fast blow fuse

Accuracy ± (% of reading + Amps) [1]

Range	Full Scale 5-½ Digits	Resolution	Max Burden Voltage	One Year 23°C ± 10°C
2.4 mA	2.40000 mA	10 ηA	25mV	0.07 + 7 μA
24 mA	24.0000 mA	100 ηA	250mV	0.08 + 9 μA
240 mA	240.000 mA	1 μA	55mV	0.07 + 60 μA
2.4 A	2.40000 A	10 μA	520mV	0.2 + 160 μA

[1] With measurement rate set to 2rps or lower rate, within one hour from Zero (Relative control).

2.3 Resistance Measurements

2.3.1 2-wire

Accuracy ± (% of reading + Ω) [1]

Range [2]	Full Scale 5-½ Digits	Resolution	Test current	One Year 23°C ± 10°C
240 Ω	240.000 Ω	1 mΩ	1 mA	0.02 + 100 mΩ
2.4 kΩ	2.40000 kΩ	10 mΩ	1 mA	0.02 + 200 mΩ
24 kΩ	24.0000 kΩ	100 mΩ	100 μA	0.02 + 1 Ω
240 kΩ	240.000 kΩ	1 Ω	10 μA	0.06 + 20 Ω
2.4 MΩ	2.40000 MΩ	10 Ω	1 μA	0.06 + 200 Ω
24 MΩ	24.0000 MΩ	100 Ω	100 nA	0.2 + 25 kΩ

[1] With measurement rate set to 2rps or lower rate, within one hour from Zero (Relative control).
[2] Test voltages are 2.4V max with the exception of the 240 Ω ranges 240 mV.

2.3.2 4-wire

Accuracy ± (% of reading + Ω) [1]

Range [2]	Full Scale 5-½ Digits	Resolution	Source current	One Year 23°C ± 10°C
240 Ω	240.000 Ω	1 mΩ	1 mA	0.02 + 50 mΩ
2.4 kΩ	2.40000 kΩ	10 mΩ	1 mA	0.02 + 100 mΩ
24 kΩ	24.0000 kΩ	100 mΩ	100 μA	0.02 + 500 mΩ
240 kΩ	240.000 kΩ	1 Ω	10 μA	0.06 + 10 Ω
2.4 MΩ	2.40000 MΩ	10 Ω	1 μA	0.06 + 200 Ω

[1] With measurement rate set to 2rps or lower rate, within one hour from Zero (Relative control).
[2] Test voltages are 2.4V max with the exception of the 240 Ω ranges 240 mV.

2.4 AC Voltage Measurements

Input Characteristics

- **Input Resistance** 1 MΩ, shunted by < 300 pF, all ranges
- **Max. Crest Factor** 4 at Full Scale, increasing to 7 at Lowest Specified Voltage
- **AC coupled** Specified range: 10 Hz to 100 kHz
- **Typical Settling time** < 0.5 sec to within 0.1% of final value
- **Typical Settling time Fast RMS** < 0.05 sec to within 0.1% of final value

2.4.1 AC Voltage True RMS Measurement

Accuracy ± (% of reading + Volts) [1]

Range	Full Scale 5-½ Digits [3]	Resolution	Lowest specified Input Voltage	One Year [2] 23°C ± 10°C
240 mV	240.000 mV	1 μV	5 mV	0.15 + 150μV
2.4 V	2.40000 V	10 μV	20 mV	0.25 + 10mV
24 V	24.0000 V	100 μV	200 mV	0.15 + 100mV
240 V	240.000 V	1 mV	2 V	0.25 + 400mV

[1] With measurement rate set to 2rps or lower rate
[2] Input frequency 47Hz to 10kHz. For other frequencies add error in tabel below
[3] Signal is limited to 8×10^6 Volt Hz Product

2.4.2 Additional Errors Due To Signal Frequency

Add the following error to the above values for signal frequencies lower than 47Hz or greater than 10kHz

Range	Signal Frequency	% of reading + Volts [1]
240 mV	20 Hz - 47 Hz	0.8 + 50μV
	10 kHz - 50 kHz	0.48 + 80μV
2.4 V	20 Hz - 47 Hz	0.75 + 1mV
	10 kHz - 50 kHz	0.45 + 2mV
24V	20 Hz - 47 Hz	0.85 + 20mV
	10 kHz - 50 kHz	0.2 + 15mV
240V	20 Hz - 47 Hz	0.85 + 200mV
	10 kHz - 50 kHz	0.15 + 100mV

[1] Select measurement rates that are lower than 1/10th of the signal frequency.

ACV Noise Rejection   Common Mode rejection, for 50 Hz or 60 Hz with 1 kΩ imbalance in either lead, is better than 80 dB.

2.5 AC Current Measurement, True RMS

Input Characteristics
- **Crest Factor** 4 at Full Scale
- **Number of built-in shunts** Two
- **Currents greater 2.4A** require external shunt
- **Protected** with 2.5A Fast blow fuse

2.5.1 AC Current True RMS Measurement

Accuracy ± (% of reading + Amps) [1]

Range	Full Scale 5-½ Digits	Resolution	Lowest Specified Current	Max Burden Voltage	One Year 23°C ± 10°C [2]
2.4 mA	2.40000 mA	10 ηA	60 μA	25mV	0.3 + 20 μA
24 mA	24.0000 mA	100 ηA	300 μA	250mV	0.2 + 100 μA
240 mA	240.000 mA	1 μA	3 mA	55mV	0.17 + 1 mA
2.4 A	2.40000 A	10 μA	30 mA	520mV	0.31 + 10 mA

[1] With measurement rate set to 2rps or lower rate
[2] Input frequency 47Hz to 1kHz. For other frequencies, see tabel below

2.5.2 Additional Errors Due To Current Frequency

Range	Signal Frequency [1]	% of reading
2.4 mA	20 Hz - 47 Hz	0.88
	1 kHz - 10 kHz	0.12
24 mA	20 Hz - 47 Hz	0.84
	1 kHz - 10 kHz	0.24
240 mA	20 Hz - 47 Hz	0.8
	1 kHz - 10 kHz	0.2
2.4 A	20 Hz - 47 Hz	0.55
	1 kHz - 10 kHz	0.2

[1] All AC Current ranges have typical measurement capability of at least 20 kHz

2.6 Diode Test Function

- **Test Currents** Five

- **Current sources voltage compliance** 4 V

Accuracy $\pm$ (% of reading + Volts) [1]

Range	Full Scale 5-½ Digits	Resolution	One Year 23°C ± 10°C
0.1 µA	2.40000 V	10 µV	0.022 + 15 µV
1 µA			0.018 + 12 µV
10 µA			0.015 + 10 µV
100 µA			0.014 + 8 µV
1 mA			0.014 + 8 µV

[1] With measurement rate set to 2rps or lower rate

2.7 Measurement Rates and power line rejection

- **Use DMMSetRate() using the following codes.**

Rate (Readings/sec)	Symbol	Code	Power line Rejection		
			50Hz	60Hz	400Hz
1	RATE_1	1	√	√	√
2	RATE_2	2	√	√	√
3	RATE_3	3	√	√	
7	RATE_7	7	√	√	
14	RATE_14	14	√	√	√
27	RATE_27	27			√
55	RATE_55	55		√	√
100	RATE_100	100	√		√

2.8 Accuracy Notes

Important: all accuracy specifications for DCV, Resistance, DCI, ACV, and ACI apply for the time periods shown in the respective specification tables. To meet these specifications, Self Calibration must be performed once a day or as indicated in the specification table. This is a simple software operation that takes a few seconds. It can be performed by calling Windows command DMMCal(), or selecting S-Cal in the control panel.

These products are capable of continuous measurement as well as data transfer rates of up to 100 readings per second (rps). In general, to achieve 5-1/2 Digits of resolution, the rate should be set to 2rps or lower.

2.9 Other Specifications

Temperature Coefficient over 0°C to 50°C Range
- Less than 0.1 x accuracy specification per °C At 23C ± 10°C

Hardware Interface	Single PXI/cPCI 3U slot
Overload Protection (voltage inputs)	300 VDC, 250 VAC
Isolation	300 VDC, 250 VAC from Earth Ground
Maximum Input (Volt x Hertz)	8×10^6 Volt x Hz normal mode input (across Voltage HI & LO). 1×10^6 Volt x Hz Common Mode input (from Voltage HI or LO relative to Earth Ground).
Safety	Designed to IEC 1010-1, Installation Category II.
Calibration	Calibrations are performed by *Signametrics* in a computer at 23°C internal temperature rise. All calibration constants are stored in a text file.
Temperature Range Operating	**-**10°C to 65°C
Temperature Range Storage	-40°C to 85°C
Size	Single 3U PXI or CompactPCI slot
Power	+5 volts, 200 mA maximum

Note: Signametrics reserves the right to make changes in materials, specifications, product functionality, or accessories without notice.

Accessories

Several accessories are available for the SM2060 series DMM's, which can be purchased directly from Signametrics, or one of its approved distributors or representatives. These are some of the accessories available:

- DMM probes SM-PRB ($15.70)
- DMM probe kit SM-PRK ($38.50)
- Deluxe probe kit SM-PRD ($95.00).
- Shielded SMT Tweezers Probes SM-PRSMT ($24.90).
- Multi Stacking Double Banana shielded cable 36" SM-CBL36 ($39.00).

- Multi Stacking Double Banana shielded cable 48" SM-CBL48 ($43.00).

- Mini DIN Trigger, 6-Wire Ohms connector SM2060-CON7 ($14.00).

- Lab View VI's library SM204x.llb (free).

- Extended 3 Year warrantee (does not include calibration) $120.00 for SM2055.

3.0 Getting Started

After unpacking the DMM, please inspect for any shipping damage that may have occurred, and report any claims to your transportation carrier.

The DMM is shipped with the Digital Multimeter module; Installation CD and a floppy disk that contain the calibration and verification files. Also included is the Certificate of Calibration.

3.1 Setting the DMM

The DMM is provided with plug-and-play installation software, and does not require any switch settings, or other adjustments prior to installation.

The **SM60CAL.DAT** file supplied with your DMM has a unique calibration record for that DMM (See "**Calibration**" at the end of this manual.)   When using multiple DMM's in the same chassis, the **SM60CAL.DAT** file must have a calibration record for each DMM. Append the unique calibration records of each DMM into one **SM60CAL.DAT** file using a text editor such as Notepad. The default location for the **SM60CAL.DAT** file is at the root directory C:\.

3.2 Installing the DMM Module

<div align="center">Warning</div>

> **To avoid shock hazard, install the DMM only into a personal computer that has its power line connector connected to an AC receptacle with an Earth Safety ground.**
>
> **After installation, check to see that no loose wires or ribbon cables infringe upon any of the internal circuits of the DMM, as this may apply measurement voltages to your computer, causing personal injury and/or damage to your computer!**

Caution: Only install the DMM module with the power turned OFF to the PC!

Use extreme care when plugging the DMM module(s) into a PCI bus slot. If possible, choose an empty slot away from any high-speed boards (e.g. video cards) or the power supply. **Please be patient during the installation process!** The DMM comes with 4 safety-input jacks. Because of their necessary size, they are a tight fit in many PC chassis. Insert the bracket end of the DMM into your PC first, watching for any interference between the safety input jacks and your PC chassis. "Sliding" the bracket end of the DMM into the chassis may be helpful. **Be patient! You should only have to install it once!**

3.3 Installing the Software

It is recommended that you first plug in the DMM into the PC chassis, than turn on the computer power. The first time you power up your computer with the DMM installed, your computer will detect it as new hardware and prompt you for a driver. The driver your computer requires is located on the installation CD (SM2060.INF).

Following the above driver installation, run the **'SETUP'** program provided on the CD. This takes care of all installation and registration requirements of the software. If you are installing the DMM on a computer that had an SM2060 series install in it, you should first uninstall the old software. Also make sure you backup and remove the old calibration record (SM60CAL.DAT). For a clean reinstallation remove all INF files containing reference to the Signametrics DMM. Depending on operating system, these files will be located at Windows\inf, Windows\inf\other or WINNT\inf. The files will be named Oemx.INF where x is 0,1,2,… and/or SIGNAMETRICS.INF or SM2060.INF. If present, these files will prevent "Found New Hardware" wizard from detecting the new DMM.

3.4 DMM Input Connectors

Before using the DMM, please take a few moments and review this section to understand where the voltage, current, or resistance and other inputs and outputs should be applied. **This section contains important information concerning voltage and current limits. Do not exceed these limits, as personal injury or damage to the instrument, your computer or application may result.**
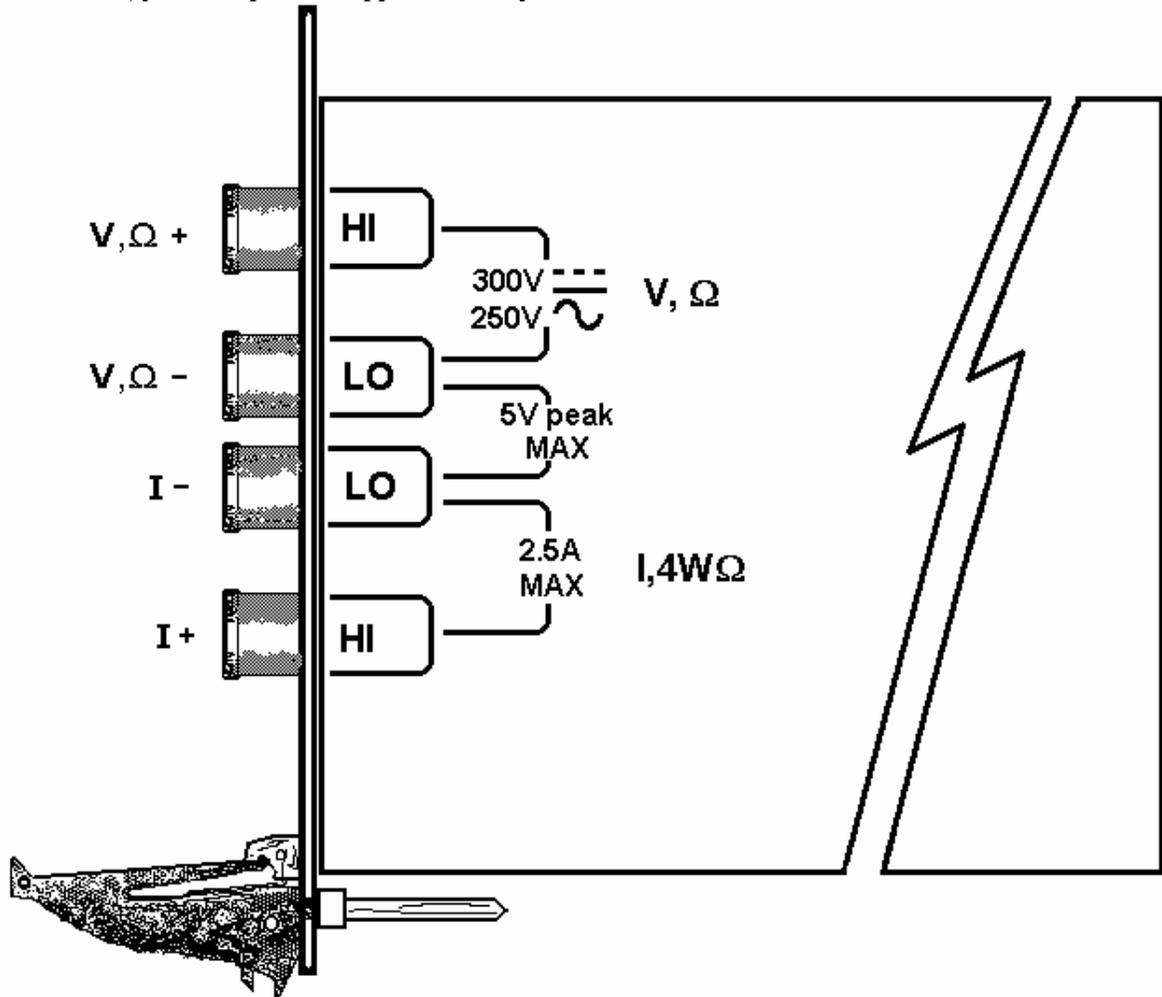
Figure 3-1. The DMM input connectors.

V, Ω + **This** is the positive terminal for all Volts, 2WΩ and diode test. It is also the Source high for 4WΩ measurements. The maximum input across **V, Ω +** and **V, Ω -** is 300 VDC or 250 VAC when in the measuring mode**. When in the 2-Wire or 4-Wire resistance mode, the maximum input allowed before damage occurs is 100 volts.**

V, Ω - This is the negative terminal for all Volts, 2WΩ and diode test. It is also the Source low for 4WΩ measurements. **Do not float this terminal or any other DMM terminal more than 300 VDC or 250 VAC above Earth Ground.**

I + This is the positive terminal for all Current measurements. It is also the Sense high for 4WΩ measurements. The maximum input across **I, 4WΩ +** and **I, 4WΩ -** is **2.5 A**. Do not apply more than 5 V peak across the I+ and I-terminals.

I – This is the negative terminal for all Current measurements. In the Current modes, it is protected with a **2.5 A, 250 V Fast Blow fuse** (5 x 20 mm). It is also the Sense low for 4WΩ measurements. **V, Ω -** and **I, 4WΩ -** should never have more than 5 V peak across them.

3.5 Starting the Control Panel

You can verify the installation and gain familiarity with the DMM by exercising its measurement functions using the Windows based Control Panel. To run the control panel, double click the "SM2064.EXE" icon. If you do not hear the relays click, it is most likely due to an installation error. Another possible source for an error is that the **SM60CAL.DAT** file does not correspond to the installed DMM.

When the DMM is started the first time, using the provided control panel (SM2064.EXE), it takes a few extra seconds to extract its calibration data from the on-board store, and write it to a file C:\SM60CAL.DAT

The Control Panel is operated with a mouse. All functions are accessed using the left mouse button. When the DMM is operated at very slow reading rates, you may have to hold down the left mouse button longer than usual for the program to acknowledge the mouse click.

Note: The SM2060 front panel powers up in DCV, 8 readings per second (rps) and 240 V range. If the DMM is operated in Autorange, with an open input, it will switch between the 2.4V and 24V ranges every few seconds, as a range change occurs. This is perfectly normal with ultra high impedance DMM's such as the SM2055. This phenomenon is caused by the virtually infinite input impedance of the 2.4V DC range. On these ranges, an open input will read whatever charge is associated with the signal conditioning of the DMM. As this electrical charge changes, the SM2055 will change ranges, causing the range switching. This is normal.

3.5 Using the Control Panel

Figure 3-2. The Control Panel. The three main groups include Measure, Source and Range buttons. The Range buttons are context sensitive such that only "240m, 2.4, 24 and 240 appear when in AC Voltage Function is selected, and 2.4m, 24m, 240m and 2.4 appear when AC Current functions is selected, etc.

Note: All of the controls described below correspond to their respective software function, which can be invoked within your control software or as objects in a visual programming environment. The software command language provides a powerful set of capabilities. Some of the functions are not included in the control panel, but are in the software.

DC/AC   This function switches between DC and AC. This is applicable for the following DMM functions: Voltage, Current, and Voltage-Source. If Voltage-Source is the function presently in use, the Source control under the Tools menu can be used to set frequency and amplitude in ACV, and amplitude only in DCV and DCI.

Relative   This is the Relative function. When activated, the last reading is stored and subtracted from all subsequent readings. This is a very important function when making low-level DCV measurements, or in 2WΩ. For example, when using 2WΩ, you can null out lead resistance by shorting the leads together and clicking on **Relative.** When making low level DC voltage measurements (e.g., in the μV region), first apply a copper short to the **V,Ω** + & **-** input terminals, allow the reading to stabilize for a few seconds, and click on **Relative**. This will correct for any internal offsets.   The **Relative** button can also be used in the Percent and dB deviation displays (shown below), which are activated using the **Tools** in the top menu.
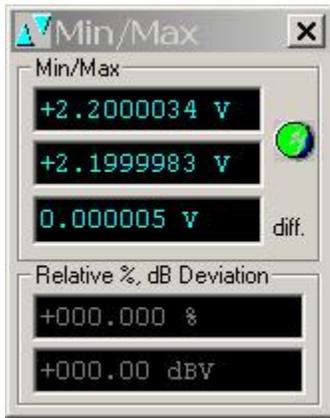
The Min/Max box can be used to analyze variations in terms of Min, Max, Percent and dBV. This display can be activated by selecting the Min/Max/Deviation from the Tools menue. For instance, testing a circuit bandwidth with an input of 1V RMS, activate the Relative function with the frequency set to 100Hz, than sweep gradually the frequency, and monitor the percent deviation as well as the dBV error and capture any response anomalies with the Min/Max display. The left display indicates peaking of 2.468% (0.21 dBV) and maximum peaking in the response of +56.24mV and a notch of –10.79mV from the reference at 100Hz.

Rate Box: Controls the DMM Measurement rate. As rate decreases, the measurement noise decreases. Also consider the power line frequency (50/60 Hz) of operation when setting it, as certain rates have better noise rejection at either 50 or 60 Hz. (See "Specifications" for details.). When measuring RMS values, there is no point setting the Rate to a value greater than 2rps.

 Range: Can be set to **Autorange** or manual by clicking on the appropriate range in the lower part of the Windows panel. Autoranging is best used for bench top application and is **not recommended** for an automated test application due to the uncertainty of the DMM range, as well as the extra time for range changes. Locking a range is highly recommended when operating in an automated test system, especially to speed up measurements. Another reason to lock a range is to control the input impedance in DCV. The 240 mV and 2.4 V ranges have virtually infinite input impedance, while the 24 V and 240 V ranges have 10 MΩ input impedance.

S_Cal: This function is the System Calibration that corrects for internal gain, scale factor and zero errors. The DMM does this by alternatively selecting its local DC reference and a zero input. It is required at least once every day to meet the accuracy specifications. It is recommended that you also perform this function whenever the external environment changes (e.g. the temperature in your work environment changes by more than 5°C. This function takes a few seconds to perform. Disconnect all leads to the DMM before doing this operation. Keep in mind that this is not a substitute for periodic calibration, which must be performed with external standards.

4.0 DMM Operation and Measurement Tutorial

Most of the measurement functions are accessible from the Windows Control Panel (Figure above). All of the functions are included in the Windows DLL driver library. To gain familiarity with the DMM, run the Windows 'SETUP.EXE' to install the software, then run the DMM, as described in the previous section. This section describes in detail the DMM's operation and measurement practices for best performance.

4.1 Voltage Measurement

Measures from 0.1 µV to 240 VDC or VAC. Use the **V, Ω +** and **V, Ω -** terminals, being certain to always leave the **I+, I-** and DIN-7 terminals disconnected. Use the AC/DC button on the Control Panel to switch between AC and DC.

Making Voltage Measurements is straightforward. The following tips will allow you to make the most accurate voltage measurements.

4.1.1 DC Voltage Measurements

When making very low-level DCV measurements (<1 mV), you should first place a copper wire shorting plug across the **V, Ω +** and **V, Ω -** terminals and perform **Relative** function to eliminate zero errors before making your measurements.   A common source of error can come from your test leads, which can introduce several µVolts of error due to thermal voltages. To minimize thermal voltaic effects, after handling the test leads; you should wait a few seconds before making measurements. Signametrics offers several high quality probes that are optimal for low-level measurements.

Note: The front panel powers up in 8rps, DCV, 240 V range. If the DMM is operated in Autorange, with an open input, The DMM will keep changing ranges. This is perfectly normal with ultra high impedance DMM'. The virtually infinite input impedance of the 240 mV and 2.4 V DCV ranges causes this phenomenon. On these ranges, an open input will read whatever charge is associated with the signal conditioning of the DMM. As this electrical charge accumulates, the DMM will change ranges.

4.1.2 True RMS AC Voltage Measurements

ACV is specified for signals greater than 1mV, from 10 Hz to 50 kHz. The ACV function is AC coupled, and measures the true RMS value of the waveform.

ACV measurements, if possible, should have the NEUTRAL or GROUND attached to the **V,Ω -** terminal. See Figure 4-1, below. This prevents any "Common Mode" problems from occurring (Common Mode refers to floating the SM2060 **V,Ω LO** above Earth Ground.)   Common Mode problems can result in noisy readings, or even cause the PC to hang-up under high V X Hz input conditions. In many systems, grounding the source to be measured at Earth Ground (being certain to avoid any ground loops) can give better results.

The settling time and low end bandwidth of the RMS function are effected by the status of the Fast RMS control circuit. When fast RMS is selected, the RMS settling time is about 10 times faster, but the low end frequency is significantly increased.
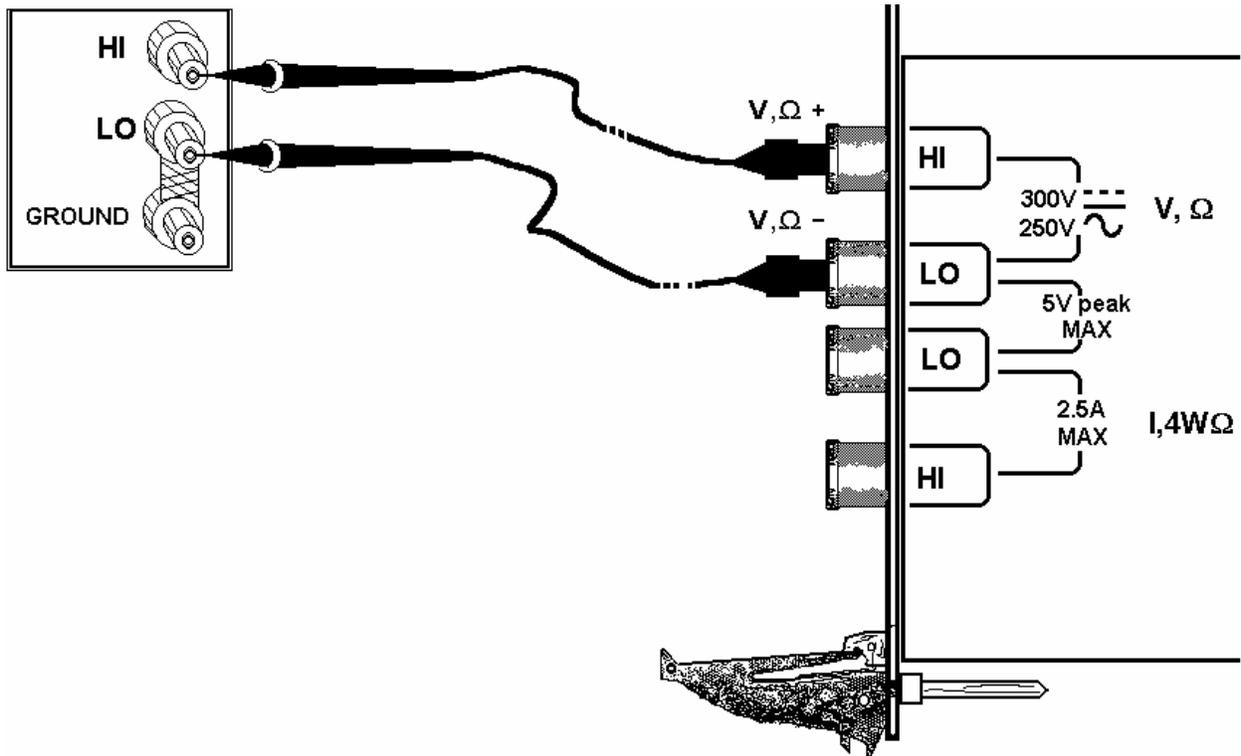
Figure 4-1. Make Voltage ACV measurements with the source ground attached to the **V,Ω -** to minimize "Common Mode" measurement problems.

4.2 Current Measurements

The DMM measures AC and DC currents between 100 ηA and 2.5 A. Use the $\pm$**I, 4WΩ** terminals, being certain to always leave the **V,Ω** + & **-** terminals disconnected. Use the AC/DC button to switch between AC and DC. The AC current is an AC coupled True RMS measurement function. See figure 4-2 for connection.

The Current functions are protected with a 2.5 A, 250 V fuse. The 2.4mA and 24mA ranges utilize a 10Ω shunt, while the 240mA and 2.4A ranges use a 0.1Ω shunt. In addition to the shunt resistors, there is some additional parasitic resistance in the current measurement path associated with the fuse and the internal wiring. The result is a low burden voltage.

Warning! Applying voltages greater than 35 V to the I+, I- terminals can cause personal injury and/or damage to your DMM and computer! Think before applying any inputs to these terminals!
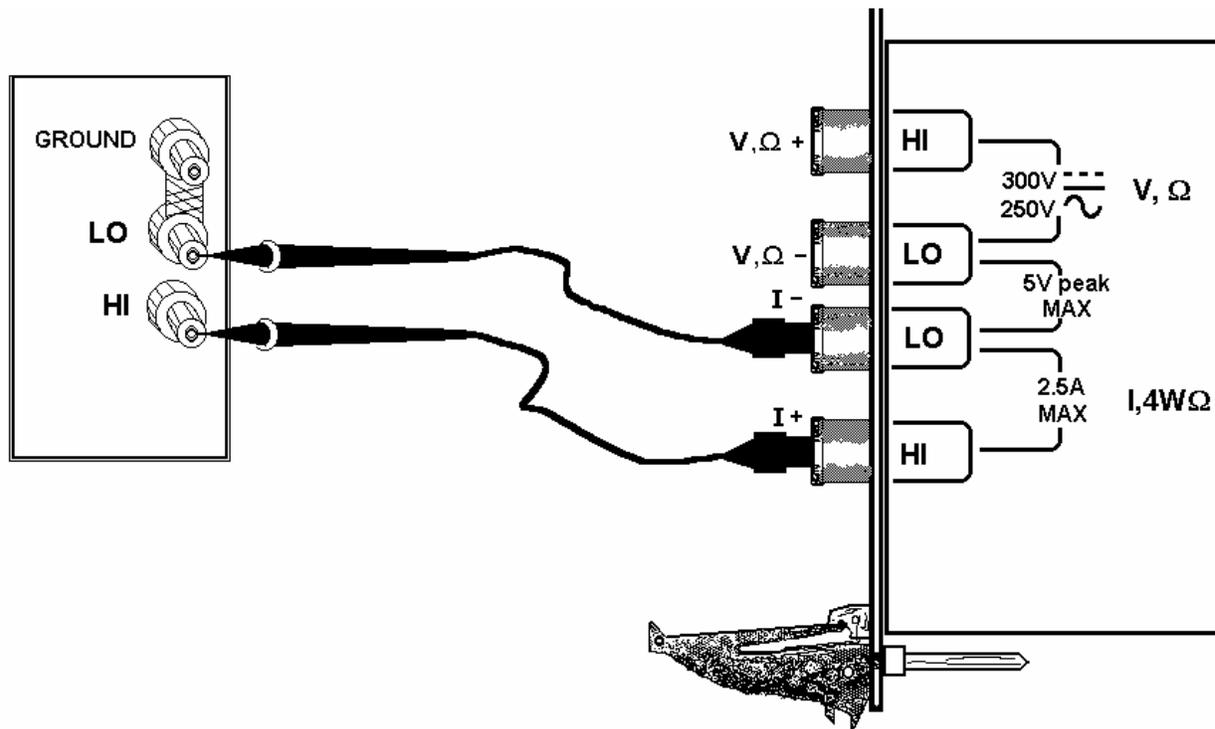
Figure 4-2. AC and DC Current measurement connection.

4.2.2 Improving DC Current Measurements

When making sensitive DC current measurements disconnect all terminals not associated with the measurement. User the **Relative** function while in the desired DC current range to zero out any residual error. Using the **S-Cal** (**DMMCalibrate ()**) prior to activating **Relative** will improve accuracy further. Although the DMM is designed to withstand up-to 2.4A indefinitely, be aware that excessive heat may be generated when measuring higher AC or DC currents. If allowed to rise this heat may adversely effect subsequent measurements. In consideration with this effect, it is recommended that whenever practical, higher current measurements be limited to short time intervals. The lower two ranges of DC current may be effected by relay contamination.

4.3 Resistance Measurements

The key for stable and accurate Resistance measurements, with low test voltage, is in the number of current sources used. This DMM uses six. The DMM measures resistance by forcing a current, and measuring a voltage, which the DMM converts and displays as a resistance value. Most measurements can be made in the 2-wire mode. The 4-wire ohms is used to make low value resistance measurements. All resistance measurement modes are susceptible to Thermo-Voltaic (Thermal EMF) errors. See section 4.3.5 for details.

4.3.1 2-Wire Ohm Measurements

The DMM measure using 240Ω to 24 MΩ ranges. Use the **V,Ω+, V,Ω-** terminals for this function. Be certain to disconnect the **I+, I-** terminals in order to reduce leakage, noise and for better safety.

Most resistance measurements can be made using the simple 2-wire Ohms method. Simply connect **V,Ω+** to one end of the resistor, and the **V,Ω-** to the other end. If the resistor to be measured is less than 24 kΩ, you should null out any lead resistance errors by first touching the **V,Ω+** and **V,Ω-** test leads together and then performing a **Relative** function. If making measurements above 200 kΩ, you should use shielded or twisted leads to minimize noise pickup. This is especially true for measurements above 1 MΩ.

You may also want to control the Ohms current used in making resistance measurements. (See the Specifications section, "Resistance, 2-wire and 4-wire", for a table of resistance range vs. current level.) All of the Ohms ranges of the DMM have enough current and voltage compliance to turn on diode junctions. For characterizing semiconductor part types, use the Diode measurement function. To avoid turning on a semiconductor junction, you may need to select a higher range (lower current). When checking semiconductor junctions, the DMM displays a resistance value linearly related to the voltage across the junction.

For applications requiring voltage and current controlled resistance measurements, use the SMX2064, which has Extended Resistance Measurement function as well as active guarding.

4.3.2 4-Wire Ohm Measurements

4-wire Ohms measurements are advantageous for making measurements below 200 kΩ, eliminating lead resistance errors. The **Voltage (V,Ω)** Input terminals serve as a current source to stimulus the resistance, and the **I, 4WΩ** Input terminals are the sense inputs. The Source + and Sense + leads are connected to one side of the resistor, and the Source - and Sense - leads are connected to the other side. Both Sense leads should be closest to the body of the resistor. See Figure 4-3.

4-wire Ohm makes very repeatable low ohms measurements, from 10 mΩ to 200 kΩ. It is not recommended to use **4WΩ** when making measurements above 200 kΩ.
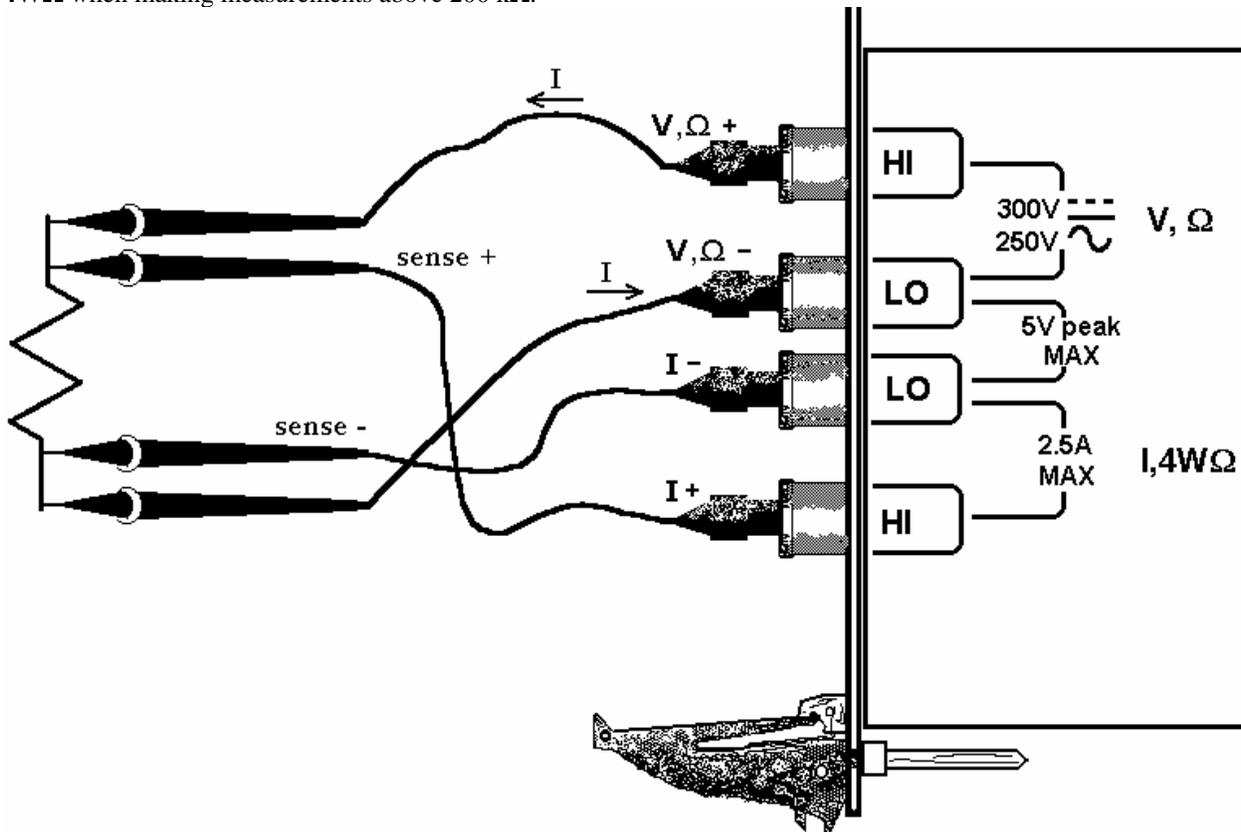
Figure 4-3. The **I- and I+** sense leads should be closest to the body of the resistor when making 4WΩ measurements.

4.3.6 Effects of Thermo-Voltaic Offset

Resistance measurements are sensitive to Thermo-Voltaic (Thermal EMF) errors. These error voltages can be caused by poor test leads, relay contacts and other elements in the measurement path. They affect all measurement methods, including 2-Wire and 4-Wire. To quantify this error, consider a system in which signals are routed to the DMM via a relay multiplexing system. Many vendors of switching products do not provide Thermal EMF specification, and it is not uncommon to find relays that have more than 50 μV. With several relay contacts in the path, the error can be significant. This error can be measured using the DMM's 240mV DC range. To do this, close a single relay that is not connected to any load, wait for a short time (about 2 minutes), than measure the voltage across the shorted relay contacts. Make sure to short the DMM leads and set 'relative' to clear the DMM offset prior to the measurement. To calculate worst-case error, count all relay contacts, which are in series with the measurement (**V, Ω+, V, Ω-** terminals in 2-Wire, and **I+, I-** terminals in 4-Wire mode). Multiply this count by the Thermal EMF voltage. Use Ohms law to convert this voltage to resistance error as in the following table.

Resistance Measurement Errors due to Thermo-Voltaic offsets.

Range	Ohms	DMM	Error due to	Error due to	Error due to

	Current	Resolution	10 µV EMF	100 µV EMF	1mV EMF
240 Ω	1 mA	1m µΩ	10 mΩ	100 mΩ	1 Ω
2.4 kΩ	1 mA	10 mΩ	10 mΩ	100 mΩ	1 Ω
24 kΩ	100 uA	100 mΩ	100 mΩ	1 Ω	10 Ω
240 kΩ	10 uA	1 Ω	1 Ω	10 Ω	100 Ω
2.4 MΩ	1 uA	10 Ω	10 Ω	100 Ω	10 Ω
24 MΩ	100 nA	100 Ω	100 Ω	1 kΩ	100 Ω

4.8 Diode Characterization

The Diode measurement function is used for characterizing semiconductor part types. This function is designed to display a semiconductor device's forward or reverse voltage. The DMM measures diode voltage at a selected current. The available source currents for diode I/V characterization include five DC current values, 100 ηA, 1 µA, 10 µA, 100 µA and 1 mA. The SMX2064 have an additional 10 mA range. It also has a variable current source that can be used concurrently with DCV measurement (see "Source Current / Measure Voltage"). This allows a variable current from 10 ηA to 12.5 mA. The maximum diode voltage compliance is approximately4 V.

Applications include I/V characteristics of Diodes, LEDs, Low voltage Zener diodes, Band Gap devices, as well as IC testing and polarity checking.

5.0 Windows Interface

The SM2060 Windows interface package provided, contains all required componenets for the following products: SMX2055, SM2060, SMX2060, SM2064, SMX2064. is a 32bit DLL based modules, which includes both, a DLL and a windows Kernel driver. This package is sufficient for most windows based software applications.

5.1 Distribution Files

The distribution CD contains all the necessary components to install and run the DMM on computers running any of the Microsoft® Windows™ operating systems. It also provides means for various software packages to control the DMM. Before installing the DMM or software, read the "Readme.txt" file. To install this software "Run Program" menu select 'autorun.exe' from the provided CD by double-click. Most files on this CD are compressed, and are automatically installed by running 'autorun', which in turn executes the setup.exe file located on the CD in the respective product directory.

The DLL is a protected-mode Microsoft® Windows™ DLL that is capable of handling up to ten Signametrics DMM's. Also provided are samples Visual Basic™ front-panel application and a C++ sample, to demonstrate the DMM and the interface to the DLL. Check the README.TXT file for more information about the files contained on the diskette. Some important files to note are:

File	Description
SM60CAL.DAT	Configuration file containing calibration information for each DMM. Do not write into this file unless you are performing an external calibration! This file is normally placed at the C:\ root directory by the setup program, and should be left there. It may contain calibration records for several DMM's.
SM2060.LIB	The Windows import library. Install in a directory pointed to by your **LIB** environment variable.
SMX2060.DEF	SM2060 driver DLL module definition file.
SM2060.DLL	The 32-bit driver DLL. This should be installed either in your working directory, in the Windows system directory, or in a directory on your **PATH**. The installation program installs this file in your Windows system directory (usually **C:\WINDOWS\SYSTEM for Win98/95 or at C:\WINNT\SYSTEM32 for Windows NT**).
SMX2060.H	Driver header file. Contains the definitions of all the DMM's function prototypes for the DLL, constant definitions, and error codes. Install in a directory pointed to by your **INCLUDE** environment variable.
DMMUser.H	Header file containing all of the necessary DMM's function, range, rate definitions to be used with the various measure and source functions.
Msvbvm50.dll	Visual Basic run-time interpreter. Usually already installed in your **C:\WINDOWS\SYSTEM** (or equivalent) directory. If it is not already installed, you will be prompted to install it by running Msvbvm50.exe for proper extraction and registration.
SM2064.vbw	Visual Basic project file
SM2064.frm	Visual Basic file with main form
SM2064.vbp	Visual Basic project file
2060glbl.bas	Visual Basic file with all global DMM declarations

File	Description

Signametrics

SM2064.exe	Visual Basic DMM control panel executable
Msvcrt.dll	System file. Installs in your **C:\WINDOWS\SYSTEM** directory.
Windrvr.vxd	Win98/95/Me Virtual Device Driver. Installs by 'setup' in your **C:\WINDOWS\SYSTEM\VMM32** directory.
Windrvr.sys	Win NT Virtual Device Driver. Installs by 'setup' in your **C:\WINNT\SYSTEM32\DRIVERS** directory.
Install.doc	Installation instructions in MS Word

Important Notes about the SM60CAL.DAT file:

The file **SM60CAL.DAT** contains calibration information for each DMM, and determines the overall analog performance for that DMM. The first time you startup your DMM, this file will be created from the stored calibration data on-board the DMM. You must not alter this file unless you are performing an external calibration of the DMM. This file may contain multiple records for more than one DMM. Each record starts with a header line, followed by calibration data.

```
card_id  8123     type 2055 calibration_date  06/15/2005
ad                 ;              A/D compensation. Set during manufacturing.
72.0    20.0 1.0
vdc      ; VDC 240mV, 2.4V, 24V,… 330V ranges. 1st entry is Offset the 2nd is gain parameters
-386.0 0.99961
-37.0 .999991
-83.0 0.999795
-8.8  1.00015
0       1.0               ;Place holder on SMX2055
vac      ; VAC 1st line - DC offset. Subsequent lines: 1st entry is Offset the 2nd is gain, 3rd freq. comp
0.0
0.84            1.015461           23
0.0043          1.0256             22
0.1             1.02205            2
0.4             1.031386           1
0               1.0                0        ;Place holder on SMX2055
idc      ; IDC 240nA to 2.5A ranges. 1st entry is offset, 2nd is gain parameter
0  1.0                   ;Place holder on SMX2055
0  1.0                   ;Place holder on SMX2055
0  1.0                   ;Place holder on SMX2055
0  1.0                   ;Place holder on SMX2055
-10.0  1.00083           ; 2.4mA range
-16.0  1.00222
-50.0  1.0034
-176.0  1.0              ; 2.4A range
iac       ; IAC 2.4mA to 2.4A ranges, offset and gain
1.6  1.02402
0.0  1.03357
1.69  1.00513
0.0  1.0142
2w-ohm  ; Ohms 24, 240, 2.4k,...,240Meg ranges, offset and gain
0               1.00               ; placeholders
1256.0          1.002307           ;240 Ohms range
110.0           1.002665
0.0             1.006304
0.0             1.003066
0.0             1.001848
0.0             0.995664                 ;24Meg range
0.0             1.0                ;      placeholders
…
```

The first line identifies the DMM and the calibration date. The "card-id" is stored on each DMM. During initialization the driver reads it from the DMM and matches it to that in the calibration record.

During initialization (**DMMInit()**), the driver reads various parameters such as DMM type (SM2060/44), and serial number, and then reads the corresponding calibration information from the **SM60CAL.DAT** file.

The **DMMInit()** function reads the information from these files to initialize the DMM. **DMMInit** accepts parameters that are the names of these files. A qualified technician may modify individual entries in the calibration file, then reload them using the **DMMLoadCalFile** command.

5.2 Using the SM2060 Driver Set With C++ or Similar Software

Install the **SM2060.H** and **DMMUser.H** header file in a directory that will be searched by your C/C++ compiler for header files. This header file is known to work with Microsoft Visual C++™. To compile using Borland, you will need to convert the **SM2060.DEF** and **SM2060.LIB** using **ImpDef.exe** and **ImpLib.exe**, provided with the compiler. Install **SM2060.LIB** in a directory that will be searched by the linker for import libraries. The SM2060 software must be installed prior to running any executable code. Install the **SM2060.DLL** in a location where either your program will do a **LoadLibrary** call to load it, or on the **PATH** so that Windows will load the DLL automatically.

In using the SM2060 driver, first call **DMMInit** which read the calibration information, performs self test and auto-calibration. Call **DMMSetFunction** to set the DMM to a measurement function. The DMM function constants are defined in the **DMMUser.H** header file, and have names that clearly indicate the function they invoke. Use **DMMSetRate** to set the reading rate defined in the header file.

Two functions are provided to return DMM readings. **DMMRead** returns the next reading as a scaled double-precision (`double`) result, and **DMMReadStr** returns the next reading as a formatted string ready to be displayed.

All functions accept a DMM-number parameter. This value, **nDmm**, is used to identify the DMM number in a multiple DMM system. This value will be 0,1,2.. n. Most functions return an error or warning code, which can be retrieved as a string using **DMMErrStr()**.

5.2.1 Multiple Card Operations under Windows

Single .EXE operation

Accessing multiple DMM's from a single executable is the most common way for running up to 10 DMM's using the Windows DLL. A combination of several SM2060s and SM2064s can be controlled, as long as the single .EXE (Thread) is used to control all of the units. Make sure that prior to issuing commands to any DMM, it is initialized using **DMMInit()**. The *nDmm* parameter is passed with each DLL command to define the DMM to be accessed. Since this configuration utilizes the DLL to service all DMM's, it must handle a single reading or control command one at a time. For example, when one DMM reads DCV, and another reads Capacitance, the DLL must finish reading the DCV before it will proceed to take a Capacitance reading. Being a relatively slow measurement, Capacitance will dictate the measurement throughput. For improved performance, one can use the following:

Multiple .EXE operation

By having several copies of **SM2060.DLL**, and renaming them, you can run multiple DMM's with separate executables. For instance, having a copy named **SM2060A.DLL** in C:\windows\system (Win98/95), and having two executable files, **MultiExe0.exe** and **MultiExe1.exe**, each of the executables will run independently, making calls to the respective DLL. This can provide an execution throughput advantage over the method mentioned above. If using Visual Basic, the **MultiExe.exe** source code should define *nDmm* = 0, and **MultiExe1.exe** should define *nDmm* = 1. Also the first should declare the **SM2060.DLL** and the second should declare **SM2064.DLL**:

MultiExe0.exe VB function declarations:
```
    Declare Function DMMInit Lib "SM2060.dll" (ByVal calFile As String) As Long
    Declare Function DMMRead Lib "SM2060.dll" (ByVal nDmm As Long, dResult As Double) As Long
    NDmm = 0
```

        MultiExe1.exe VB function declarations:
```
    Declare Function DMMInit Lib "sm20432A.dll" (ByVal calFile As String) As Long
    Declare Function DMMRead Lib "sm20432A.dll" (ByVal nDmm As Long, dResult As Double) As Long
    NDmm = 1
```

```c
/************************************************************************
 * Exmp2040.C Exmp2040.EXE
 *
 * A simple Windows .EXE example for demonstrating the SM2060,64
 * DMM's using "C"
 * Sets Function to VDC, Range to 24V, Rate set to 6rps.
 * Display five measurements using a Message box.
 ************************************************************************
 * Make sure SM2060.lib is included in the libraries. For Microsoft
 * Version 4.0 C++ and above, place under 'Source Files' in the
 * Workspace, along side with Exmp2060.c
 *  PROJECT SETTINGS:
 *
 *   /nologo /ML /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE" /D "_MBCS"
 *   /FR"Release/" /Fp"Release/Exmp2060.pch" /YX /Fo"Release/" /Fd"Release/" /FD /c
 *
 *  Copy both SM2060.DLL and SM2060.LIB to the project directory.
 *
 ************************************************************************/
// #define WINAPI __stdcall
#include <windows.h>
#include <string.h>
#ifdef _Windows
  #define _WINDOWS
#endif
#include "SMX2060.h"              // functions declarations and error codes.
#include "DMMUser.h"              // All functions, range and rate info and function declarations.
int main(void){
        int I, nDmm = 0;          // Address first DMM in the system
        char Read[16];
        char strMsg[256];
        i = DMMInit(nDmm,"C:\\SM60CAL.dat");      // initialize SM2055, and read calibration file
        if(i<0)
           MessageBox(0,"Initialization ERROR !", "Startup SM2060 DLL",MB_OK);      // Error
        DMMSetFunction(nDmm,VDC);              // Set to DCV function
        DMMSetRange(nDmm,_24V);                // and to 33V  range
        DMMSetRate(nDmm, RATE_6R50);           // meas. Rate = 6
        strcpy(strMsg,"");           //              Clear string store
        for(i=1; i<= 5; i++){                   // take 5 readings
         DMMReadStr(nDmm,        Read);      // read
         strcat(strMsg,Read)        ;           // Append each reading
         strcat(strMsg,"          ");       // insert space between readings
        }
        MessageBox(0,strMsg, "SM2060.DLL Read Resistance & VDC",MB_OK);      // Show readings
        return 0L;
}
```

5.3 Visual Basic Front Panel Application

The Visual Basic front panel application, **SM2064.EXE,** is an interactive control panel for the SM2060 family of DMMs. When it loads it will take a few seconds to initialize and self calibrate the hardware before the front panel is displayed.

The push buttons labeled **V, I**, etc. control the DMM function. As you push a function, the front panel application will switch the DMM to the selected range and function. Autorange mode is selected by pushing the **AutoRange** check box. The **S-Cal** box recalibrates the DMM, leaving the DMM in the same state prior to operation. (This is an

internal calibration only, and is different from the external calibration, which writes to the **SM60CAL.DAT** file. **S-Cal** is used to correct for any internal offset and gain drifts due to changes in operating temperature).

The source code file **2060Glbl.BAS** (in the **V_BASIC** directory of the distribution diskette) contains the function declarations and the various ranges, rates and other parameters that are required. These definitions are the duplicates of the "C" header files required to write Visual Basic applications which interact with the driver DLL, along with some global variables required for this particular front-panel application.

5.3.1 Visual Basic Simple Application

The following is a simple panel application for Visual Basic, which includes two files, Global.Bas and SimplePanel.frm. It has a panel that contains two objects, a **Text Box** to display the DMM readings, and a **Command Button** that acts as a reading trigger.

Global.bas module file contents:

```
Option Explicit
' Declare all functions we are going to be using: From SM2060.H file.
Declare Function DMMInit Lib "SM2060.dll" (ByVal nDmm as long, ByVal calFile As String) As Long
Declare Function DMMSetRate Lib "SM2060.dll" (ByVal nDmm As Long, ByVal nRate As Long) _
As Long
Declare Function DMMSetFunction Lib "SM2060.dll" (ByVal nDmm As Long, ByVal nFunc As Long) As Long
Declare Function DMMSetRange Lib "SM2060.dll" (ByVal nDmm As Long, ByVal nRange As Long) As Long
Declare Function DMMRead Lib "SM2060.dll" (ByVal nDmm As Long, dResult As Double) As Long

' Definitions from DMMUser.H
' for DMMSetFunction()
Global Const VDCFunc = 0
Global Const VACFunc = 4
Global Const Ohm2Func = 21
Global nDmm as Long

' for DMMSetRange()
Global Const Range0 = 0
Global Const Range1 = 1
Global Const Range2 = 2
Global Const Range3 = 3

'Measurement Rate for use with DMMSetRate()
Global Const RATE_1R60= 4        '1rps with 60Hz line rejection
Global Const RATE_1R50 = 5       '1rps with 50Hz line rejection
Global Const RATE_2R60= 6        '2rps
Global Const RATE_4R60 = 8       '4rps
Global nDmm As Long              ' Global store for the DMM number
```

SimplePanel.frm Form file contents:

```
Private Sub Form_Load()                          'Fomr_Load allways gets executed first.
    Dim i As Long
    nDmm = 0                     'Set to first DMM in the system
    i = DMMInit(nDmm,"C:\SM60CAL.dat")    'Initialize and load cal file
    i = DMMSetFunction(nDmm, VDCFunc)    'Set DMM to DCV function
    i = DMMSetRange(nDmm, Range2)       'Select the 24V range
    i = DMMSetRate(nDmm, RATE_2R60)   'Set measurement rate
End Sub

Private Sub ReadBotton_Click()          'Read Botton Click action.
    Dim i As Long                       'Any time this botton is pressed
    Dim dReading As Double              'the DMM takes a reading and displays it.
    i = DMMRead(nDmm, dReading)               'Take a reading
    TextReading.Text = dReading               'display it in a Text box.
End Sub
```

5.4 Windows DLL Default Modes and Parameters

After initialization, the Windows DLL default modes and parameters on your DMM are set up as follows:

- Auto ranging: Off
- Function: DC Volts
- Range: 240V
- Relative: Off
- Measurement Rate: 2rps

5.5 Using the SM2060 DLL with LabWindows/CVI®

When using the SM2060 DLL with LabWindows/CVI, you should read the **LabWin.txt** file included with the software diskette.

An example application of SM2060 DLL calls from LabWindows/CVI ® is shown below. It contains functions **measure_ohms()** and **measure_vdc()**, with sample calls to the SM2060.

NOTE: Although these measurement functions use LabWindows/CVI® and the LabWindows/CVI(R) Test Executive, they are not necessarily coded to LabWindows® instrument driver standards.

```
/* function: measure_ohms, purpose: measure 2-wire ohms */
int measure_ohms(double OHMreading) {
        short ret, i;
        DMMSetFunctions (0, OHMS2W);
        DMMSetAutoRange (0, TRUE);
        /* to settle auto-range and function changes ignore three readings */
        for( i = 0 ; i < 4 ; i++ )  ret = DMMReadNorm (0, & OHMreading);
        return ret;
}
/* function: measure_vdc, purpose: measure DC Volts */
int measure_vdc(double Vreading) {
        short ret, i;
        DMMSetFunctions (0, VDC);
        DMMSetAutoRange (0, TRUE);
        /* to settle auto-range and function changes ignore three readings */
        for( i = 0 ; i < 4 ; i++ ) ret = DMMReadNorm (0, &Vreading);
        return ret; }
```

5.6 Windows Command Language

The following section contains detailed descriptions of each function of the Windows command language. Those commands that pertain to only the SM2060 are indicated. Most functions return an error code. The code can either be retrieved as a string using **DMMErrString** function, or looked up in the **SM2060.H** header file. The **DMMUser.H** file contains all the pertinent definitions for the DMM ranges functions etc. The following description for the various functions is based on "C" function declarations. Keep in mind that the Windows DLL containing these functions assumes all **int** values to be windows 32bit integers (corresponds to VisualBasic **long** type). TRUE is 1 and FALSE is 0 (which is also different from VisualBasic where True is –1 and False is 0).

Grayed out functions are either, untested or unimplemented.

DMMCalibrate

Description   Internally calibrate the DMM.

                        #include "SM2060.h"

                        int DMMCalibrate(**int** *nDmm*)

Remarks   This function performs self calibration of the various components of the DMM, as well as an extensive self test. At the end of this operation it returns the DMM to the current operating mode. Using this function periodically, or when the DMM internal temperature varies, will enhance the accuracy of the DMM. Using this function does not remove the requirement to perform periodic external calibration.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	DMM is OK.
Negative Value	Error

Example   `status = DMMCalibrate(0); /* a quick internal cal.*/`

Comments   This performs an internal DMM calibration and is the same as the **S-Cal** command in the VB Control Panel. It is not related to the external calibration represented in the **SM60CAL.DAT** file.

DMMCleanRelay

Description   Service function that cleans specified relay.

                        #include "SM2060.h"

                        int DMMCleanRelay(int *nDmm,* **int** *iRelay,* **int** *iCycles*)

Remarks   This function cleans *iRelay* by vibrating the contact *iCycles* times. This function is useful for removing oxides and other deposits from the relay contacts. DC Current measurements are particularly sensitive to K2 contact resistance and therefore should be cleaned periodically. It is also useful for making sound in computer without a speaker.

Parameter	Type/Description
iRelay	**int** The relay to clean. 1 for K2, 2 for K2 and 3 for K3.
iCycles	**int** The number of times the relay contact is shaken. 1 to 1000.
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example   `int status = DMMCleanRelay(0, 2, 100); // Shake K2 1000`

DMMClearMinMax

Description   Clears the Min/Max storage.

        #include "SM2060.h"

        int DMMClearMinMax(int *nDmm***)**

Remarks   This function clears the Min/Max values, and initiates a new Min/Max detection. See **DMMGetMin** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example   `int status = DMMClearMinMax(0);`

DMMClosePCI

Description   Close the PCI bus for the specified DMM. Not for user applications.

        #include "SM2060.h"

        int DMMClosePCI(int *nDmm*)

Remarks   This function is limited for servicing the DMM. It has no use in normal DMM operation. See also **DMMOpenPCI()** function.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.

	Negative Value	Error code

Example   `int status = DMMClosePCI(0);`

DMMDelay

Description   Wait for a given time.

#include "SM2060.h"

int DMMDelay(**double** *dTime*)

Remarks   Delay of *dTime* seconds. *dTime* must be a positive double number between 0.0 and 100.0 seconds.

Parameter	**Type/Description**
dTime	**double** Delay time in seconds.

Return Value   The return value is one of the following constants.

Value	**Meaning**
DMM_OKAY	Operation successfully terminated
Negative Value	Error code

Example   `DMMDelay(1.2); /* wait for 1.2 Sec */`

DMMErrString

Description   Return the string describing the warning or error code.

#include "SM2060.h"

int DMMErrString(**int** *iErrorCode,* **LPSTR** *lpszError,* **int** *iBuffLength*)

Remarks   This function returns a string containing the error or warning description which corresponds to the *iErrorCode*. The string is placed at *lpszError*. Error codes are negative numbers, while warning codes are positive numbers.

Parameter	Type/Description
iErrorCode	**int** Error code.
iBuffLength	**int** The maximum available length of the string buffer
lpszError	**LPSTR** Points to a buffer (at least 64 characters long) to hold the error/warning string.

Return Value   The return value is the length of the error string or one of the following constants.

Value	Meaning
Negative Value	Error code

Example
```
char cBuf[64];
int length = DMMErrString( -3, cBuf, 48);
```

DMMGetBusInfo

Description   Returns the PCI Bus and Slot numbers for the selected DMM.

int DMMGetBusInfo(**int** *nDmm,* **int** **bus,* **int** **slot*)

Remarks   This function reads the PCI *bus* and *slot* numbers for the selected DMM. . It provides means to relate the physical card location to the *nDmm* value by detecting the location of a DMM in the PCI system tree. This function actually scans the hardware rather then look up the information in the registry.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
bus	**int *** a pointer to integer at which the bus number is stored (0 to 255)
slot	**int *** A pointer to an integer where the slot number is stored

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Operation was successful.
Negative number	Error code

Example
```
int bus, slot; // Find on which bus, and slot the DMM is at
DMMGetBusInfo(3, &bus, &slot); // DMM#3
```

DMMGetCalDate

Description   Return the calibration date string from the DMM.

int DMMGetCalDate(**int** *nDmm,* **LPSTR** *lpszCalDate*)

Remarks   This function reads the calibration date string from the structure. This is the date the DMM was calibrated last.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszCalDate	**LPSTR** Points to a buffer (at least 64 characters long) to hold the cal date string.

Return Value   The return value is one of the following constants.

Value	Meaning
any positive number	Length of the date string
Negative number	Error code

Example
```
char cBuf[64];
int status;
status = DMMGetCalDate(0, cBuf);
```

DMMGetdB

Description   Get dB deviation from the reading at the time relative was activated.

#include "SM2060.h"

int DMMGetdB(int *nDmm***, double ****lpdDev***)**

Remarks   This function returns a double floating value that is the dB deviation relative to the reading made just before the relative function was activated. This function is useful in determining measurement errors in dB. It can be used for bandwidth measurements or DC evaluation.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdDev	**double *** Pointer where the dB value is to be saved.

Return Value   Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example
```
double dB; int status = DMMGetdB(0, &dB);
```

DMMGetdBStr

Description   Get dB deviation from the reading at the time relative was activated.

#include "SM2060.h"

int DMMGetdBStr(int *nDmm***, LPCSTR** *lpszDB***)**

Remarks   This function is the same as the **DMMGetdB()**, with the exception that it returns a string. See **DMMGetdB()** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszDB	**LPCSTR** Points to a buffer (at least 64 characters long) to hold the result. The return value will consist of a leading sign a floating-point, and a 'dB' units specifier

Return Value   Integer string length if successful, or an error code..

Value	Meaning
Negative Value	Error code

Example   `char cBuf[64]; int strLength = DMMGetdBStr(0, cBuf);`

DMMGetDeviation

Description   Get percent deviation from the reading at the time relative was activated.

#include "SM2060.h"

int DMMGetDeviation(int *nDmm***, double ****lpdDev***)**

Remarks   This function returns a double floating value that is the percent deviation relative to the reading made just before the relative function was activated (**DMMSetRelative**). This function is useful in quantifying measurement errors. It can be used for bandwidth measurements or DC evaluation, or percent variation of a device under test over temperature. The absolute value of *lpdDev* can be used as a pass/fail window for production. Another function effecting **DMMGetDeviation** is **DMMSetReference**. Unlike **DMMSetRelative**, which uses the current measurement as a reference, **DMMSetReference** provides the facility to set this reference.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdDev	**double *** Pointer where the deviation value is to be saved.

Return Value   Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example   `double  error;`
                     `int status = DMMGetDeviation(0, &error);`

DMMGetDeviatStr

Description   Get percent deviation from the reading at the time relative was activated.

#include "SM2060.h"

int DMMGetDeviatStr(int *nDmm***, LPCSTR** *lpszDev***)**

Remarks   This function is the same as the **DMMGetDeviation()**, with the exception that it returns a string. See **DMMGetDeviation()** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszDev	**LPCSTR** Points to a buffer (at least 64 characters long) to hold the result. The return value will consist of a leading sign a floating-point, and a % units specifier

Return Value   Integer string length if successful, or an error code.

Value	Meaning
Negative Value	Error code

Example
```
char cBuf[64];
int strLength = DMMGetDeviatStr(0, cBuf);
```

DMMGetDiffMnMxStr

Description   Returns the difference between the max and min values as string.

#include "SM2060.h"

int DMMGetDiffMnMxStr (**int** *nDmm,* **LPSTR** *lpszReading*)

Remarks   This function return the difference between the current Max. and Min values, which is the peak-to-peak range of recent readings. It returns the result as a string formatted for printing. The print format is determined by the range and function.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszReading	**LPSTR** Points to a buffer (at least 64 characters long) to hold the result.

Return Value   The return value is one of the following constants, or the string length is OK.

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example
```
char cBuf[64];
int status = DMMGetDiffMnMxStr(0, cBuf);
```

DMMGetFunction

Description   Get DMM function code.

                           #include "SM2060.h"
                           #include "DMMUser.h"

                           int DMMGetFunction(**int** *nDmm*)

Remarks   This function returns the DMM function code. The codes are defined in the DMMUser.h file.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer value corresponding to the current function, or an error code.

Value	Meaning
Positive value	See DMMUser.h for function/range codes.
Negative Value	Error code

Example   `if(DMMGetFunction == VDC) printf("VDC Function selected");`

DMMGetGrdVer

Description   Get DMM firmware version.

                           #include "SM2060.h"

                           int DMMGetGrdVer(**int** *nDmm*)

Remarks   This function returns the DMM firmware version of the on-board controller.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer value. The return value is the version value or an error code.

Value	Meaning
Positive Value	Version
Negative Value	Error code

Example   `firmwarever = DMMGetGrdVer(0);`

DMMGetHwVer

Description   Get the hardware version of the DMM.

#include "SM2060.h"

int DMMGetHwVer(**int** *nDmm*)

Remarks   This function returns the hardware version. A returned value of 0 corresponds to Rev_, 1 corresponds to Rev_A, 2 to Rev_B etc.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   DMM hardware code or an error code.

Value	Meaning
Positive value	Hardware version code
Negative Value	Error code

Example   `int HWVer = DMMGetHwVer(0);`

DMMGetID

Description   Get DMM ID code.

#include "SM2060.h"

int DMMGetID(**int** *nDmm*)

Remarks   This function returns the DMM identification code. Each DMM has a unique ID code that must match the calibration file **card_ID** field in **SM60CAL.DAT**. This code must reflect the last digits of the DMM serial number.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer value card ID code (serial number) or an error code.

Value	Meaning
DMM_E_DMM	Invalid DMM number.

Example   `int id = DMMGetID(0);`

DMMGetManDate

Description   Get Manufacturing date stamp from the DMM hardware

#include "SM2060.h"

int DMMGetManDate(**int** *nDmm,* **int ****month,* **int ****day,* **int ****year*)

Remarks   This function returns the DMM manufacturing date which is read from the hardware. The month, day and year are returned as integers. This is used to track the DMM to a specific manufacturing date.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
month	**int *** A pointer to an integer where the month is stored
day	**int *** A pointer to an integer where the day is stored
year	**int *** A pointer to an integer where the year is stored

Return Value   Integer error code or.

Value	Meaning
DMM_OKAY	Operation was successful.
DMM_E_DMM	Invalid DMM number.

Example
```
int month, day, year, status
status = DMMGetManDate(0, &month, &day, &year);
```

DMMGetMax

Description   Get Maximum reading history.

#include "SM2060.h"

int DMMGetMax(int *nDmm***, double ****lpdMax***)**

Remarks   This function returns a double floating value that is the maximum (of the Min/Max function) value since either a function change, range change or call to the **DMMClearMinMax** function was made. This value is updated every time a measurement is performed using **DMMRead**, **DMMReadStr** or **DMMReadNorm**.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdMax	**double *** Pointer where the Max value is to be saved.

Return Value   Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example   `double  Mx; int status = DMMGetMax(0, &Mx);`

DMMGetMaxStr

Description   Returns the maximum as a formatted string.

#include "SM2060.h"

int DMMGetMaxStr(**int** *nDmm,* **LPSTR** *lpszReading*)

Remarks   This function is the string version of **DMMGetMax**. It returns the result as a string formatted for printing. The print format is determined by the range and function. See **DMMGetMax** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszReading	**LPSTR** Points to a buffer (at least 64 characters long) to hold the result.

Return Value   The return value is one of the following constants, or the string length is OK.

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example   `char cBuf[64];`
`int status = DMMGetMaxStr(0, cBuf);`

DMMGetMin

Description   Get Minimum reading history.

#include "SM2060.h"

int DMMGetMin(int *nDmm***, double** **lpdMax*)

Remarks   This function returns a double floating value that is the minimum (of the Min/Max function) value since either a function change, range change or a call to the **DMMClearMinMax**() function was made. This value is updated every time a measurement is performed using **DMMRead**, **DMMReadStr** or **DMMReadNorm**.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdMax	**double *** Pointer where the Min value is to be saved.

Return Value   Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example   `double  Min; int status = DMMGetMin(0, &Min);`

DMMGetMinStr

Description   Returns the minimum as a formatted string.

#include "SM2060.h"

int DMMGetMinStr(**int** *nDmm,* **LPSTR** *lpszReading*)

Remarks   This function is the string version of **DMMGetMin**. It returns the result as a string formatted for printing. The print format is determined by the range and function. See **DMMGetMin** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszReading	**LPSTR** Points to a buffer (at least 64 characters long) to hold the result.

Return Value   The return value is one of the following constants, or the string length is OK.

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example
```
char cBuf[64];
int status = DMMGetMinStr(0, cBuf);
```

DMMGetRange

Description   Get DMM range code.

#include "SM2060.h"
#include "DMMUser.h"

 int DMMGetRange(**int** *nDmm*)

Remarks   This function returns the DMM range code. The range codes are in the sequence of 0, 1, 2, 3, … where 0 is the lowest range.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer value corresponding to the currently set DMM range, or an error code.

Value	Meaning
Zero or positive value	Range; zero being the lowest
Negative Value	Error code

Example
```
int id;
if(DMMGetRange == 0) printf("Lowest range selected");
```

DMMGetRate

Description   Get the currently set reading rate

#include "SM2060.h"

int DMMGetRate(int *nDmm*, double *lpdRate)

Remarks   This function returns a double floating rate in readings per second. See **DMMSetRate** for details

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdRate	**double *** Pointer where the rate is saved to.

Return Value   Integer value version code or an error code.

Value	Meaning
Negative Value	Error code

Example
```
int status; double  rate;
status = DMMGetRate(0, & rate);
```

DMMGetType

Description   Get the type of the DMM.

#include "SM2060.h"

int DMMGetType(**int** *nDmm*)

Remarks   This function returns a value identifying the DMM model.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   DMM type Integer or an error code.

Value	Meaning
2055	SM2055 is at nDmm slot
2060	SM2060 is at nDmm slot
2064	SM2064 is at nDmm slot
Negative Value	Error code

Example   `int DMMtype = DMMGetType(0);`

DMMGetVer

Description   Get DMM software driver version.

#include "SM2060.h"

int DMMGetVer(**int** *nDmm***, double** ***lpfResult*)

Remarks   This function returns the DMM software driver version, which is a double floating value.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpfResult	**double *** Pointer to the location which holds the version.

Return Value   Integer error code.

Value	Meaning
Negative Value	Error code

Example   `int status; double ver;`
        `status = DMMGetVer(0, &ver);`

DMMInit

Description   Initialize a DMM.

#include "SM2060.h"

int DMMInit(**int** *nDmm,* **LPCSTR** *lpszCal*)

Remarks   This function must be the first function to be executed. It opens the driver for the
specified DMM. The first DMM being 0, the second 1, etc.. It also initializes the DMM
hardware and does extensive self test to the DMM hardware. It then initializes the
software and reads the appropriate calibration record for the respective DMM from the
file specified by *lpszCa,* followed by self calibration. If the calibration record is outdated,
it opens a warning window. If an error is detected, an error code is returned.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszCal	**LPCSTR** Points to the name of the file containing the calibration constants for the DMM. Calibration information is normally read from the file named **SM60CAL.DAT** located in the current directory.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code

Example
```
/* initialize DMM */
int i = DMMInit(0,"C:\SM60CAL.dat");// Initialize the first DMM
```

DMMIsAutoRange

Description   Get the status of the autorange flag.

#include "SM2060.h"

int DMMIsAutoRange(**int** *nDmm*)

Remarks   This function returns the DMM autorange flag state.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Value	Meaning
TRUE	Autoranging mode is selected.
FALSE	Autoranging mode is not selected.
DMM_E_DMM	Invalid DMM number.

Return Value   TRUE, FALSE or an error code.

Example   `int autorange = DMMIsAutoRange(0);`

DMMIsInitialized

Description   Get the status of the DMM.

#include "SM2060.h"

int DMMIsInitialized(**int** *nDmm*)

Remarks   This function returns the status of the DMM. If TRUE, the DMM has been initialized and is active. If FALSE the DMM is not initialized. To use the DMM, it must be initialized using **DMMInit** or **DMMQuickInit** functions. This function is used for maintenance and is not needed under normal operation.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   TRUE, FALSE or an error code.

Value	Meaning
TRUE	DMM is initialized and active.
FALSE	DMM is not initialized.
DMM_E_DMM	Invalid DMM number.

Example   `int active = DMMIsInitialzied(0);`

DMMIsRelative

Description   Get the status of the Relative flag.

#include "SM2060.h"

int DMMIsRelative(**int** *nDmm*)

Remarks   This function returns the DMM Relative flag state.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer TRUE, FALSE or an error code.

Value	Meaning
TRUE	Relative mode is selected.
FALSE	Relative mode is not selected.
Negative Value	Error code

Example   `int rel = DMMIsRelative(0);`

DMMOpenPCI

Description   A service function which open the PCI bus for the specified DMM. Not for user application.

#include "SM2060.h"

int DMMOpenPCI(int *nDmm*)

Remarks   This function is limited for servicing the DMM. It has no use in normal DMM operation.. See also **DMMClosePCI()** function.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example   `int status = DMMOpenPCI(0);`

DMMRead

Description   Return the next floating-point reading from the DMM.

#include "SM2060.h"

int DMMRead(int *nDmm***, double** *lpdResult*)

Remarks

Executing the **DMMRead** function causes the DMM to perform a single conversion and retrieve the result. The DMM, performs all scaling and conversion required, and returns the result as a 64-bit double-precision floating-point number in the location pointed to by *lpdResult*. It can read all the **Primary** functions (those that can be selected using **DMMSetFunction()** and **DMMSetRange()**). Returned result is a scaled value which is normilized to the selected range. That is . That is, it returns 200 for 200mV input in the 240 mV range, and 100 for 100 kΩ input in the 330k Ω range. Alternatively use the **DMMReadNorm()** function for base units read function, or **DMMReadStr()** to return the results as formated string of the **DMMRead()**.Very large values are indication of over range condition.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdResult	**double *** Points to the location to hold the next reading.

Return Value

The return value is one of the following constants.

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code
Positive Value	Warning code, including over range.

Example

```
double dResults[100];
int status;
For(i=0; I < 100; i++) DMMRead(0, &dResults[i]);// Read to a buffer
```

DMMReadNorm

Description

Take a reading that is in base value.

#include "SM2060.h"

int DMMReadNorm(int *nDmm***, double ****lpdRead***)**

Remarks

This function returns a double floating-point reading. Unlike **DMMRead()** the returned value is in base units. That is, it returns 0.2 for a 200 mV input and 1e6 for a 1.0 MΩ. Very large values are indication of over range condition.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdRead	**double *** Pointer to a location where the reading is saved.

Return Value

Integer value version code or an error code.

Value	Meaning
DMM_E_RANGE	Over/Under range error.
Negative Value	Error code
DMM_OKAY	Valid return.

Example   `double reading; int status = DMMReadNorm(0, &reading);`

DMMReadStr

SM2060 ☑ SM2064 ☑

Description   Return the next reading from the DMM formatted for printing.

#include "SM2060.h"

int DMMReadStr(**int** *nDmm,* **LPSTR** *lpszReading*)

Remarks   This function is the string version of **DMMRead()**. It reads the next measurement result, performs all scaling and conversion required, and returns the result as a string formatted for printing. The print format is determined by the range and function. See **DMMRead()** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszReading	**LPSTR** Points to a buffer (at least 64 characters long) to hold the converted result. The return value will consist of a leading sign, a floating-point value in exponential notation, and a units specifier.

Return Value   The return value is one of the following constants, or the string length is OK.

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code
DMM_E_RANGE	DMM over range error occurred.

Example   `char cBuf[64]; int status = DMMReadingStr(0, cBuf);`

DMMSetAutoRange

Description   Enable/Disable autorange operation of DMM

#include "SM2060.h"

int DMMSetAutoRange(**int** *nDmm,* **int** *bAuto*)

Remarks   This function enables or disables autorange operation of the DMM.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
bAuto	**int** Determines whether or not autoranging is done. The value TRUE (1) enables autoranging, FALSE (0) disables it.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example   `status = DMMSetAutoRange(0, TRUE); /* enable autoranging */`

DMMSetFunction

Description   Set the DMM function.

#include "SM2060.h"
#include "DMMUser.h"

int DMMSetFunction(**int** *nDmm,* **int** *nFunc*)

Remarks   This function selects the DMM's measurement function. The DMMUser.h file contains a table of values defined as *VDC*, *VAC*, *IAC*, *IDC*, *OHMS4W*, *OHMS2W etc...* Not all functions are available for all DMM types.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
nFunc	**int** A pre-defined constant corresponding to the desired function.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code
DMM_E_FUNC	Invalid DMM function.

Example   `status = DMMSetFunction(0, IDC); // Set for DC current`

DMMSetRange

Description   Set the DMM range for the present function.

#include "SM2060.h"

int DMMSetRange(**int** *nDmm,* **int** *nRange*)

Remarks   This function sets the range used by the DMM for the present function. The table of values is defined by the *_240mV, _2400uA, etc.* In general, the lowest range is 0, next is 1 etc. Each function has a pre defined number of ranges as specified in the specification section of this manual. Not all ranges are available for all DMM types. For instance the SM2064 has a 24 Ohms and 240Meg range, while the SM2060 and SMX2055 do not.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
nRange	**int** A pre-defined constant corresponding to the desired range.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code
DMM_E_RANGE	Invalid DMM range value.

Example   `status = DMMSetRange(0, _24mA);`

DMMSetRate

Description   Set the measurement rate.

#include "SMX2060.h"
#include "DMMUser.h"

int DMMSetRate(**int** *nDmm,* **int** *iRate*)

Remarks   This function sets the rate at which the DMM makes measurements. The allowed values are defined in the DMMUser.h file. The rate (*iRate*) can be set from 1rps (RATE_1) to 100rps (RATE_100). Some of the rates have specific power line rejection as indicated in the specification part of this manual.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iRate	**int** A pre-defined constant corresponding to the desired rate.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code
ERR_APERTURE	Invalid measurement rate value entered.

Example   `status = DMMSetRate(0, RATE_2);      // Set to 2rps`

DMMSetRelative

Description   Set the DMM relative reading mode for the present function.

               #include "SM2060.h"

               int DMMSetRelative(int *nDmm,* **int** *bRelative*)

Remarks   This function selects relative or absolute reading mode for the DMM. If the *bRelative* parameter value is TRUE (1), the DMM will change to relative reading mode. If FALSE, the DMM will change to absolute reading mode.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
bRelative	**int** TRUE (1) to enter relative mode, FALSE (0) to clear mode.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	DMM mode changed successfully.
Negative Value	Error code

Example   `status = DMMSetRelative(0, TRUE);`

DMMTerminate

Description   Terminate DMM operation (DLL)

               #include "SM2060.h"

               int DMMTerminate(**int** *nDmm*)

Remarks   Removes DMM number *nDmm*. This routine is used only where it is needed to terminate one DMM and start a new one at the same *nDmm* location. Otherwise, it is not recommended to use this function.

Parameter	Type/Description
nDmm	**int** Identifies the DMM to be suspended.

Return Value   The return value is one of the following constants.

Value	Meaning
TRUE	DMM Terminated
FALSE	DMM was not initialized, termination is redundant.

Example   `DMMTerminate(0); /* Terminate DMM # 0 */`

5.7 Calibration Service Commands

AC_zero

Description   Disable AC measurement zero funciton.

> **#include "SM2060.h"**
> **#include "UseroDMM.h"**

> **int AC_zero(int** *nDdmm,* **int** *bACZero* **)**

Remarks   ith bACZero FALSE, the AC zero function is disabled. If TRUE it is enabled. The default value is TRUE. Diabeling the AC Zero funciton allows the derivation of the value to be set as offset parameter for the selected ACV range. This function is used during calibration.

Parameter	Type/Description
iDmm	Identifies the DMM. DMMs are numbered starting with zero.
bACZero	Forces the AC zero to be active or inactive. Allowed values are TRUE of FALSE.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example
```
int err;
Err = AC_zero(0, FALSE); // disable AC Zero.
```

DMMLoadCalFile

Description   Reload calibration record from file.

> **#include "SM2060.h"**

> **int DMMLoadCalFile**(**int** *nDmm*, **LPCSTR** *lpszCal*)

Remarks   This function provides the capability to reload the calibration record. This is useful in making limited calibration adjustments, and verifying them. By having a copy of the original calibration file '**SM60CAL.DAT**' open with an editor, modifying calibration parameters and then reloading using **DMMLoadCalFile**, one can instantly verify the corrections made. Make sure the '**SM60CAL.DAT**' file itself is not altered since that will void the calibration.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

lpszCal		**LPCSTR** Points to the name of the file containing the calibration constants for the DMM.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Cal record loaded successfully.
Negative Value	Error code

Example
```
/* Load a modified copy of the original calibration file to
verify correction made to a specific entry */
int i = DMMLoadCalFile(0, "C:\CAL_A.dat");
```

GetGain

Description   Retrieve currently set gain.

#include "SM2060.h"
#include "UseroDMM.h"

int GetGain(int *nDmm,* **doulbe *** *lpdGain***)**

Remarks   This function returns the currently set gain,. This is the gain associated with the currently selected function and range. The value should be the same as that set in the calibration record for this function and range. The gain is returned as a 64-bit double-precision floating-point number in the location pointed to by *lpdGaint*. This function is useful while performaing calibration. Set **SetGain()** function for additional details.

Parameter	Type/Description
iDmm	Identifies the DMM. DMMs are numbered starting with zero.

Return Value   The return value is one of the following constants.

Value	Meaning
lpdGain	**double *** Points to the location to hold the gain.
DMM_OKAY	Valid return.
Negative Value	Error code

Example
```
double gain;
GetGain(0, &gain); // read gain
```

GetOffset

Description   Retrieve currently set gain.

#include "SM2060.h"
#include "UseroDMM.h"

int GetOffset(int *nDmm,* **doulbe *** *lpdOffset***)**

Remarks   This function returns the currently set offset,. This is the offset associated with the currently selected function and range. The value should be the same as that set in the calibration record for this function and range. The offset is returned as a 64-bit double-precision floating-point number in the location pointed to by *lpdOffsett*. This function is useful while performaing calibration. Set **SetOffset()** function for additional details.

Parameter	Type/Description
iDmm	Identifies the DMM. DMMs are numbered starting with zero.

Return Value   The return value is one of the following constants.

Value	Meaning
lpdOffset	**double *** Points to the location to hold the offset.
DMM_OKAY	Valid return.
Negative Value	Error code

Example
```
double offst;
GetOffset(0, &offst); // read gain
```

SetFcomp

Description   Set the ACV Frequency compensation factor
#include "SM2060.h"

int SetFcomp(int *nDmm,* **int** *iFcomp***)**

Remarks   This function sets the value of the ACV frequency compensation DAC. It is used for calibration the ACV bandwidth..

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iFcomp	**int** Freqeuncy Compnensation DAC value to be set. Allowed value is between 0 and 31.

Return Value   Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

| Example | `SetFcomp(0, 12); // set the frequency compensation` |

SetOffset

Description   Set the the offset correction factor

#include "SM2060.h"

int SetOffset(int *nDmm,* **double** *dOffset***)**

Remarks   This function sets the value of the offset correction factor for the currently set function and range..

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
dOffset	**double** Offset value to be set.

Return Value   Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

| Example | `SetOffset(0, 11212.0); // Assert the offset factor` |

Linearize_AD

Description   Activate/Deactivate A/D linearization correction.

#include "SM2060.h"
#include "UseroDMM.h"

int Lineaize_AD(int *nDdmm,* **int** *bLinerize* **)**

Remarks   If *bLinerize* is set to FALSE disables the A/D Linearization correction. The default value is TRUE. Diabeling allows for the derivation of the parameters for calibration purposes. This function is used during calibration.

Parameter	Type/Description
iDmm	Identifies the DMM. DMMs are numbered starting with zero.
bACZero	Forces the AC zero to be active or inactive. Allowed values are TRUE of FALSE.

Return Value   The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

| Example | `int err;` |

```
Err = Linearize_AD(0, FALSE); // disable AC Zero.
```

Read_ADcounts

Description   Read A/D offset counts.
                        #include "SM2060.h"

                        int Read_ADcounts(int *nDmm***)**

Remarks   This function returnes the A/D raw counts. It is useful for retrieving the offset parameter
                        for various functions, including VDC, 2-W and 4-W ohms and DC current. It is limited
                        for service use.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.

Return Value   Integer error code.

Value	Meaning
Any value	**int** Offset reading.

Example   int i = Read_ADcounts(0); // read offset parameter

5.8 Maintanance Commands

GrdXingTest

Description

Perform the specified test
#include "SM2060.h"

int GrdXingTgest(int *nDmm,* **int** *iNumber,* **int** *iTest***)**

Remarks

Perform the specified test as indicated by *iTest*. Repeat it for *iNumber* times. This function is used to perform basic H/W tests.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iTest	**int** Test type. 0: Basic Read/Write. 1: Toggle Reset line iNumber times. 2: High Speed Guard Crossing stimulation. 3: Guarded controller communication test. 4: Guard Crossing loopback test. 5: High Speed Guard Crossing test (SM2064).
iNumber	**int** Number of tests to be repeated.

Return Value

Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

int i = GrdXingTest(0, 1, 3); // Test Guarded controller

5.9 Error Codes

Operation of the DMM may be impaired, should be aborted or is not possible following an Error. Use the **DMMErrString()** function, to retrieve the string describing the error.

DMM_OKAY	0	// no error
DMM_E_CAL	-1	// cannot open the calibration file at the specified location.
DMM_E_INIT	-3	// DMM must be initialized in order to execute the operation.
DMM_E_IO	-4	// I/O Error, DMM not responding
NOT_FOUND	-5	// DMM was not detected PCI bus
DMM_E_CAL_R	-6	// Not found a valid calibration record in specified file.
ERR_AD_HW	-7	// H/W Error, the A/D does not respond.
ERR_HW_INIT	-8	// H/W error, can't access H/W to initialize it. May be due to bad address
NO_CAL_RECORD	-9	// can't find a cal record for for this DMM in the specified cal file.
ID_HW_ERR	-10	// Can't read ID from DMM
GUARD_COM	-12	// Communication error with DMM uP
TIMEOUT	-13	// process timed out Error
GUARD_XING	-14	// Guard crossing is broken
CONTROLR_COM	-15	// Microcontroller communication error
OVERRUN	-16	// Communication Overrun error
FRAME	-17	// Communication Frame error
RCV_FIFO	-18	// Com receive Fifo error

```
PARITY              -19    // Com parity error
WRONG_TYPE          -20    // Wrong Cal record for DMM type
WRONG_GRD_VER       -21    // MCU Firmwhare does not support operation
CANT_OPEN_PCI       -22    // Can't open PCI device. Already open ?
PCI_ITEMS           -23    // Card does not have all PCI items.
GENERAL_ERR         -24    // General Error
CAL_STORE           -26    // Error reading Cal record from local storage
CREAT_CAL_FILE      -27    // Can't create named cal file to write cal record to
OPEN_CAL_FILE       -28    // Can't open cal file for reading cal record
CREAT_CAL_RCRD      -29    // Can't create on-board Cal Record
```

5.10 Warning Codes

Following a warning, the DMM will continue to run normally with the exception of the fault indicated by the warning code. Use the **DMMErrString()** function, to retrieve the string describing the warning. This string may be used to notify the user. Based on it, an action may be taken to correct the source of the warning. Several of the warning codes are part of a normal operation. Such are DMM_CNT_RNG, which indicates that the counter requires more iterations, or the POS_FS and NEG_FS are indication that the signal level is too high for the selected range, which is normal.

```
DMM_E_FUNC          102    // Invalid function value used
DMM_E_RNG           103    // Invalid range value used
DMM_E_IS_INIT       105    // Dmm already initialized: in use
ERR_FUNC            107    // Illegal function selection
ERR_PARAMETER       110    // wrong parameter value
UN_CALIBRATED       111    // Expired Calibration. Needs service
MC_STOP             115    // Microcontroller was stopped/interruped during an operation
POS_FS              116    // Positive Over Range
NEG_FS              117    // Negative Over Range
BUSY                118    // DMM is busy, wait for ready
FUNC_INACTIVE       119    // Function can not be selected, or not available for this type DMM.
```

5.11 Parameter List

The following definitions are from the DMMUser.H file.

5.11.1 Measurement and Source Functions

The following list contains values that set the DMM functions. Use the **DMMSetFuncction()** function to set these values. Use **DMMGetFunction()** to retrieve the value of the currently set function

```
#define  VDC             0       DC Volts
#define  VAC             5       AC Volts
#define  IAC             10      AC Current
#define  IDC             14      DC Current
#define  OHMS4W          22      2-Wire resistance
#define  OHMS2W          29      4-Wire resistance
#define  DIODE           37      Diode test
```

5.11.2 Range Values

The following list contains the allowed values for range setting with **DMMSetRange()** function. Use the **DMMGetRange()** function to retrieve the currently set range

```
// AC and DC Volts
#define  _240mV          0               // five DCV ranges
#define  _2400mV         1
```

```
#define  _24V                    2
#define  _240V                   3
// AC Current
#define  _2400uAAC               0            // 2.4mA
#define  _24mAAC                 1            // 24mA
#define  _240mAAC                2
#define  _2400mAAC               3            // 2.4A
// DC Current
#define  _2400uA                 4            // 2.4mA
#define  _24mA                   5            // 24mA
#define  _240mA                  6            // 240mA
#define  _2400mA                 7            // 2.4A
// 2 Wire and 4 Wire Ohms
#define  _240                    1
#define  _2400                   2
#define  _24k                    3
#define  _240k                   4
#define  _2400k                  5            // Two Meg range
#define  _24MEG                  6            // 2-Wire
// Diode test
#define  _D100n                  0            //Test current = 100nA
#define  _D1u                    1            // 1uA
#define  _D10u                   2            // 10uA
#define  _D100u                  3            // 100uA
#define  _D1m                    4            // 1mA
```

5.11.3 Measurement Rate parameters

The following list contains the definitions for the available Apertures. Use **DMMSetRate()** and **DMMGetRate()** to set and retrieve the apertures.

```
#define RATE_1              1            // 1rps with 60Hz line rejection
#define RATE_2              2            // 2rps with 50Hz line rejection
#define RATE_3              3            // 3rps with 60Hz line rejection
#define RATE_7              7            // 7rps with 50Hz line rejection
#define RATE_14             14           // 14rps with 60Hz line rejection
#define RATE_27             27           // 27rps with 50Hz rejection
#define RATE_55             55           // 54rps with 60Hz rejection
#define RATE_100     100          // 100rps with 50Hz rejection
```

6 Calibration

Each SM/SMX2055/60/64 DMM uses its own **SM60CAL.DAT** calibration record to ensure the accuracy of its functions and ranges. The **SM60CAL.DAT** file is a text file that contains the DMM identification number, calibration date, and calibration constants for all DMM ranges. When the DMM is installed this file is generated from an internally stored record. Once extracted, the DMM reads it from a file rather than from its on-board record, since it is faster to read from a file. For most functions, the calibration constants are scale factor and offset terms that solve an "y = mx + b" equation for each range. An input "x" is corrected using a scale factor term "m" and an offset term "b"; this gives the desired DMM reading, "y". Keep in mind that for ranges and functions that are unavailable for a particular product in the SM2060 family. The following calibration record is for the SMX2055 and it contains some placeholders for ranges that are not available with the product. An example **SM60CAL.DAT** follows:

```
card_id  8123     type 2055 calibration_date 06/15/2005
ad       #A/D compensation
2.0      10    0.99995
vdc      #VDC 240mV, 2.4V,24V, 240V, 330V ranges, offset and gain parameters
-386.0   0.99961
-37.0    0.999991
-83.0    0.999795
-8.8     1.00015
0        1.0               ;Place holder
vac      #VAC 1st line - DC offset. Than offset, gain and freq each range240mV to 330V
0                    ;Place                    holder
0.84     1.015461         23
0.0043   1.0256           23
0.0      1.02205          0
0.0      1.031386         0
0        1.0              0         ;Place holder
idc      # IDC 240nA to 2.5A, 8 ranges, offset and gain
0        1 ;Place        holder
0        1 ;Place        holder
0        1 ;Place        holder
0        1 ;Place        holder
-1450.0  1.00103 ;2.4mA range
-176.0   1.00602
-1450.0  1.00482
-176.0   1.00001          ;2.4A range
iac      # IAC 2.4mA to 2.5A ranges, offset and gain
1.6      1.02402
0.0      1.03357
1.69     1.00513
0.0      1.0142
2w-ohm #Ohms 24, 240, 2.4k,24k,240k,2.4M,24M,240Meg ranges, offset and gain
0        1               ;Place holder
1256.0   1.002307        ;240 Ohms
110.0    1.002665
0.0      1.006304
0.0      1.003066
0.0      1.001848
0.0      0.995664         ;24 MOhms
0        1               ;Place holder
...
```

The first column under any function, e.g.,"vdc", is the offset term "b", expressed as a value proportional to analog-to-digital (a/d) counts. The second column is the scale factor term "m". Within each function, the "b" and "m" terms are listed with the lowest range at the beginning. For example, under "2w-ohm" above, "1.27e+4 1.002259" represents the offset term for the 24 Ω range, and "1.002259" is the scale factor for this range.

For the ACV function, the first line in the calibration record is the DC offset value. The rest of the lines contain the RMS offset, gain correction factor, and a third column that represents a digital code from 0 to 31 that controls the high frequency performance of each AC function. A large value, e.g., 31, implies high attenuation.

The **SM60CAL.DAT** file is created by performing external calibration. The general calibration algorithm consists of applying a zero value to the DMM followed by a value of $2/3^{rd}$ of the top of each range. Calibration of your SM/SMX2055/60/64 is best performed using calibration software available from Signametrics.

When using multiple DMMs in a single chassis, the **SM60CAL.DAT** file must have a calibration record for each DMM. You can combine the unique calibration records of each DMM into one **SM60CAL.DAT** file using any ASCII text editor such as "notepad.exe".

7.0 Warranty and Service

The SM2060, SM2064, SMX2055, SMX2060 and SMX2064 are warranted against defects in manufacturing and materials for a period of one year from date of purchase. Removal of any of the three external shields or any attempt to repair the unit by other than unauthorized Signametrics service personnel will invalidate your warranty. Operating the Signametrics products outside their specified limits will void the warranty. For in-warranty repairs, you must obtain a return materials authorization (RMA) from Signametrics prior to returning your unit. Customer ships products at customer's expense. Within the USA Signametrics will ship serviced or replaced unit at Signametrics' expense.

Warranty extensions are available at the time of purchase for terms up to 36 months, in increments of 12 months.

If your unit requires repair or calibration, contact your Signametrics representative. There are no user serviceable parts within these products.

8.0 Accessories

Several accessories are available for the SM2055 DMM, which can be purchased directly from Signametrics, or one of its distributors or representatives. These include:

- Basic DMM probes

- DMM probe kit

- Deluxe DMM probe set

- Shielded SMT Tweezer Probes

- Multi Stacking Double Banana shielded cable 36"

- Multi Stacking Double Banana shielded cable 48"

- Mini DIN-7 Trigger, 6-Wire Ohms connector

- 4-Wire Kelvin probes