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

5990-9492EN

Agilent Technologies

Operator's Manual

Model SMU2055 6½ Digit USB Digital Multimeter

Signametrics Corporation

April, 2010
Compatible with Rev 1.62 Hardware

CAUTION

In no event shall Signametrics or its Representatives be held 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 from of the use of or inability to use Signametrics products, even if Signametrics Corporation 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 be applicable to you.

TABLE OF CONTENTS

1.0 Introduction

Congratulations! You have purchased a USB Digital Multimeter. The SMU2055 Digital Multimeter (DMM) has sophisticated analog and digital circuitry to provide repeatable measurements, and its protected to handle any unexpected situations your measurement environment may encounter. Your new DMM is easy to setup and use. To get years of reliable service from your DMM, please take a few moments to review this manual before installing and using this precision instrument.

This manual describes the SMU2055 USB module.

1.1 Safety Considerations

Safety Considerations

The SMU2055 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 set to the common mode voltage. **These voltages can be lethal and may KILL!**

The DMM comes installed in a plastic chassis with 4 components (bottom, top, front and back) that **must not be removed due to performance as well as safety reasons.** Removal of these shields and/or improper assembly of the shields can expose lethal voltages. The plastic screws are **required** for safety reasons.

Warning

Check to see that all measurement wires are separated from the USB communication cable. If measurement wires come into contact with the USB cable, this may apply measurement voltages to your computer, causing personal injury and/or damage to your computer!

When making any measurements above 50 VDC or 40 VAC, use only approved Safety Test Leads such as Signametrics Basic Test Leads or Deluxe Test Leads, offered as optional accessories with the Signametrics DMM's.

1.2 Minimum Requirements

The SMU2055 DMMs are precision modules that are compatible with any USB 2.0 or higher computer port. A mouse must be installed when controlling the DMM from the Windows Control Panel. The SMU2055 comes with a Windows' DLL, for operation with Windows Version 95/98/Me/2000/XP, Vista and LINUX.

1.3 Feature Set

The SMU2055 is a traditional 6-1/2 digit DMM and can be used as a general purpose DMM. Signametrics also offers the high precision 7-1/2 digit USB DMM, the SMU2060, which adds capacitance measurements, triggered operation and a high resolution frequency counter. If your requirements call for a High Workload, then the Multi Function SMU2064 will probably fit the ticket with its timing measurements, inductance, sourcing and higher speed. The SMU2064 specialized measurements can replace costly instruments and thus shrinking the size and cost of a test system.

SMU2055 basic features:

Function	SMU2055 DMM
Volts DC; four ranges, 240mV to 240V	√
Volts AC; TRMS; four ranges, 240mV to 240V	√
2-Wire Ohms, six ranges 240 Ω to 24 MΩ	√
4-Wire Ohms, six ranges 240 Ω to 24 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 Five ranges, 100 ηA to 1mA	√
Auto range, Relative	√
Min/Max, dB and percent deviation functions	√
High Dynamic range; ±2,400,000 counts	√
Measurement rate selectable: 0.5 to250 readings/s	√

2.0 Specifications

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

It is important to note that a specified range for a DMM is expressed as a numeric value indicating the highest absolute measurement that can be displayed using that 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

Accuracy ± (% of reading + Volts) [1]

Range	Full Scale 6-½ Digits	Resolution	24 hours 23°C ± 1°C	90 Days 23°C ± 5°C	One Year 23°C ± 5°C
240 mV	240.0000 mV	100 ηV	0.003 + 2 μV	0.005 + 3 μV	0.007 + 3.5 μV
2.4 V	2.400000 V	1 μV	0.003 + 3 μV	0.004 + 4 μV	0.005 + 10 μV
24 V	24.00000 V	10 μV	0.004 + 200 μV	0.005 + 250 μV	0.007 + 300 μV
240 V	240.0000 V	100 μV	0.004 + 600 μV	0.005 + 800 μV	0.007 + 1 mV

[1] With reading rate set to ≤ 2/sec, and within one hour from Self Calibration (S-Cal).

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

Maximum reading rate	Digits of Resolution	
1 / second	6-1/2	22 bits
27/second	6	20 bits
250 / second	5-1/2	18 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 Reading	Resolution	Max Burden Voltage	24 hours 23°C ± 5°C	90 Days 23°C ± 5°C	One Year 23°C ± 5°C
2.4 mA	2.40000 mA	10 ηA	25mV	0.05 + 300 ηA	0.06 + 700 ηA	0.07 + 1 μA
24 mA	24.0000 mA	100 ηA	250mV	0.05 + 350 ηA	0.065 + 800 ηA	0.08 + 1 μA
240 mA	240.000 mA	1 μA	55mV	0.05 + 50 μA	0.055 + 60 μA	0.065 + 80 μA
2.4 A	2.40000 A	10 μA	520mV	0.3 + 60 μA	0.4 + 70 μA	0.45 + 90 μA

[1] With reading rate set to ≤ 2/sec, and within one hour from Zero (Relative control).

2.3 Resistance Measurements

Characteristics

- **Number of current sources** Five
- **Maximum Test Voltage** 2.4V

2.3.1 2-wire

Accuracy ± (% of reading + Ω) [1]

Range	Full Scale 6 ½ Digits	Resolution	Source current	24 hours 23°C ± 1°C	90 Days 23°C ± 5°C	One Year 23°C ± 5°C
240 Ω	240.0000 Ω	100 μΩ	1 mA	0.004 + 4.5 mΩ [2]	0.005 + 5 mΩ [2]	0.008 + 6 mΩ [2]
2.4 kΩ	2.400000 kΩ	1 mΩ	1 mA	0.0025 + 28 mΩ	0.004 + 32 mΩ	0.007 + 33 mΩ
24 kΩ	24.00000 kΩ	10 mΩ	100 μA	0.003 + 300 mΩ	0.004 + 330 mΩ	0.007 + 350 mΩ
240 kΩ	240.0000 kΩ	100 mΩ	10 μA	0.008 + 3.2 Ω	0.012 + 4 Ω	0.02 + 5 Ω
2.4 MΩ	2.400000 MΩ	1 Ω	1 μA	0.02 + 40 Ω	0.03 + 50 Ω	0.04 + 70 Ω
24 MΩ	24.0000 MΩ	100 Ω	100 nA	0.2 + 400 Ω	0. 3 + 500 Ω	0.4 + 600 Ω

[1] With reading rate set to ≤ 2/sec, and within one hour from Self Calibration (S-Cal).
[2] Use of S-Cal and Relative to improve measurement floor.

2.3.2 4-wire

Accuracy ± (% of reading + Ω) [1]

Range [3]	Full Scale 6 ½ Digits	Resolution	Source current	24 hours 23°C ± 1°C	90 Days 23°C ± 5°C	One Year 23°C ± 5°C
240 Ω	240.0000 Ω	100 μΩ	1 mA	0.004 + 3 mΩ [2]	0.005 + 4 mΩ [2]	0.008 + 5 mΩ [2]
2.4 kΩ	2.400000 kΩ	1 mΩ	1 mA	0.0025 + 28 mΩ	0.004+ 32 mΩ	0.007 + 33 mΩ
24 kΩ	24.00000 kΩ	10 mΩ	100 μA	0.003 + 300 mΩ	0.004 + 330 mΩ	0.007 + 350 mΩ
240 kΩ	240.0000 kΩ	100 mΩ	10 μA	0.008 + 3.2 Ω	0.012 + 4 Ω	0.02 + 5 Ω

[1] With reading rate set to ≤ 2/sec, and within one hour from Self Calibration (S-Cal).
[2] Use of Relative to facilitate indicated floor.
[3] The 2.4MΩ and 24MΩ ranges are functional but not specified for 4-Wire operations.

2.4 AC Voltage Measurements

Input Characteristics

- **Input Resistance** 1 MΩ, shunted by < 300 pF
- **Max. Crest Factor** 4 at Full Scale, increasing to 7 at Lowest Specified Voltage
- **AC coupled** Specified range: 10 Hz to 200 kHz
- **Typical Settling time** < 0.5 sec to within 0.1% of final value
- **Common Mode rejection:** at 50Hz to 60Hz, 1 kΩ imbalance in either lead, is better than 60 dB.

2.4.1 AC Voltage True RMS Measurement

Range [1]	Full Scale 6-½ Digits	Lowest specified Voltage	Resolution
240 mV	240.0000 mV	5 mV	100 ηV
2.4 V	2.400000 V	20 mV	1 µV
24 V	24.00000 V	200 mV	10 µV
240 V	240.0000 V	2 V	100 µV

[1] Signal is limited to $8x10^6$ Volt Hz Product. For example, the largest frequency input at 250 V is 32 kHz, or $8x10^6$ Volt x Hz.

2.4.2 AC Volts Accuracy

Settles to rated accuracy within 0.5 seconds for signals >50% of scale.
May take up to 2s to settle to rated accuracy for signals < 5% of scale.

Accuracy ± (% of reading + Volts) [1]

Range	Frequency	24 hours 23°C ± 1°C	90 Days 23°C ± 5°C	One Year 23°C ± 5°C
240 mV	10 Hz - 20 Hz	3 + 300 µV	3.4 + 500 µV	3.6 + 0.6 mV
	20 Hz - 47 Hz	0.45 + 150 µV	0.47 + 170 µV	0.5 + 0.35 mV
	47 Hz – 10 kHz	0.13 + 100 µV	0.14 + 110 µV	0.16 + 0.25 mV
	10 kHz – 50 kHz	0.3 + 180 µV	0.35 + 220 µV	0.4 + 0.35 mV
	50 kHz - 100 kHz	2 + 0.6 mV	2.1 + 0.8 mV	2.2+ 1 mV
2.4 V	10 Hz - 20 Hz	3 + 2 mV	3.3 + 2.2 mV	3.5 + 2.5 mV
	20 Hz - 47 Hz	0.45 + 1.3 mV	0.47 + 1.5 mV	0.5 + 1.7 mV
	47 Hz – 10 kHz	0.07 + 1 mV	0.075 + 1.1 mV	0.08 + 1.2 mV
	10 kHz – 50 kHz	0.35 + 2 mV	0.37 + 2.3 mV	0.4 + 2.5 mV
	50 kHz - 100 kHz	2+ 2.5 mV	2.1 + 2.8 mV	2.2 + 5 mV
24 V	10 Hz - 20 Hz	4 + 25 mV	4.3 + 16 mV	4.5 + 35 mV
	20 Hz - 47 Hz	0.50 + 20 mV	0.55 + 25 mV	0.65 + 30 mV
	47 Hz – 10 kHz	0.07 + 15 mV	0.075 + 20 mV	0.085 + 23 mV
	10 kHz – 50 kHz	0.2 + 25 mV	0.25 + 30 mV	0.3 + 35 mV
	50 kHz - 100 kHz	1.5 + 35 mV	1.6 + 28 mV	1.7 + 50 mV
240 V	10 Hz - 20 Hz	3 + 250 mV	3.1 + 350 mV	3.3 + 400 mV
	20 Hz - 47 Hz	0.45 + 230 mV	0.+ 300 mV	0.+ 350 mV
	47 Hz – 10 kHz	0.07 + 180 mV	0.075 + 220 mV	0.08 + 250 mV
	10 kHz – 50 kHz	0.28 + 250 mV	0.29 + 300 mV	0.55 + 350 mV
	50 kHz - 100 kHz	1.5 + 350 mV	1.6 + 400 mV	1.8 + 500 mV

[1] With reading rate set to ≤ 2/sec

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

Specs in **red** indicate change from last manual release.

2.5 AC Current Measurement, True RMS

Input Characteristics

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

2.5.1 AC Current True RMS Measurement

Range	Full Scale 6 1/2 Digits	Lowest Specified Current	Maximum Burden Voltage (RMS)	Resolution
2.4 mA	2.400000 mA	60 µA	25mV	1 nA
24 mA	24.00000 mA	300 µA	250mV	10 nA
240 mA	240.0000 mA	3 mA	55mV	100 nA
2.4 A	2.400000 A	30 mA	520mV	1 uA

Accuracy ± (% of reading + Amps)

Range	Frequency [1]	24 hours 23°C ± 1°C	90 Days 23°C ± 10°C	One Year 23°C ± 10°C
2.4 mA	10 Hz - 20 Hz	3.8 + 4 µA	2.7 + 4 µA	2.9 + 4 µA
	20 Hz - 47 Hz	0.9 + 4 µA	0.9 + 4 µA	1.0 + 4 µA
	47 Hz - 1 kHz	0.04 + 1.5 µA	0.08 + 3 µA	0.12 + 4 µA
	1 kHz - 10 kHz	0.12 + 4 µA	0.14 + 4 µA	0.22 + 4 µA
24 mA	10 Hz - 20 Hz	1.8 + 30 µA	2.6 + 30 µA	2.8 + 30 µA
	20 Hz - 47 Hz	0.6 + 30 µA	0.9 + 30 µA	1.0 + 30 µA
	47 Hz - 1 kHz	0.07 + 10 µA	0.15 + 20 µA	0.16 + 30 µA
	1 kHz - 10 kHz	0.21 + 30 µA	0.3 + 40 µA	0.4 + 40 µA
240 mA	10 Hz - 20 Hz	1.8 + 400 µA	2.7 + 400 µA	2.8 + 400 µA
	20 Hz - 47 Hz	0.6 + 400 µA	0.9 + 400 µA	1.0 + 400 µA
	47 Hz - 1 kHz	0.1 + 100 µA	0.17 + 180 µA	0.2 + 220 µA
	1 kHz - 10 kHz	0.3 + 300 µA	0.35 + 350 µA	0.4 + 400 µA
2.4 A	10 Hz - 20 Hz	1.8 + 4 mA	2.5 + 4.5 mA	2.7 + 5 mA
	20 Hz - 47 Hz	0.66 + 4 mA	0.8 + 6 mA	0.9 + 6 mA
	47 Hz - 1 kHz	0.3 + 3.8mA	0.33 + 3.8 mA	0.35 + 4 mA
	1 kHz - 10 kHz	0.4 + 4mA	0.45 + 4.5 mA	0.5 + 5 mA

2.6 Diode Test Function

- **Test Currents** Five
- **Current sources voltage compliance** 4 V

Accuracy ± (% 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 Control

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

Rate (Readings/sec)	Symbol	Code	Power line Rejection		
			50Hz	60Hz	400Hz
0.5	RATE_p5	0	√	√	√
1	RATE_1	1	√	√	√
2	RATE_2	2	√	√	√
3	RATE_3	3	√	√	√
7	RATE_7	7	√	√	√
14	RATE_14	14			√
27	RATE_27	27			√
50	RATE_50	50			√
90	RATE_90	90			
170	RATE_170	170			
250	RATE_250	250			

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 250 readings per second (rps). In general, to achieve 6-1/2 Digits of resolution, the rate should be set to 2rps or lower.

Since the DMM is powered via a USB calbe (AM/BM 6’ cable), it is important to make sure it gets the required 5V supply. Using the right USB cable is very important. Make sure this cable has a 24 AWG wires for power supply, indicated by marking on the cable such as 28/1P + 24/2C. Be aware that there are a lot of cables which are marked 28/1P + 28/2C. These have high resistance, and will not be adequate. Another issue can be with powered USB hubs. Some of the lower quality units can have upwords of 8V rather than the required 5V +/-5%. On initialization (DMMInint) the DMM measures its internal supply voltage and returnes an error or warning code if the power is inadequate. See DMMGetSupplyV function.

2.9 Other Specifications

Temperature Coefficient	Less than 0.1 x accuracy specification below 18°C and over 28°C
Hardware Interface	Single USB/cPCI 3U slot
Overload Protection (voltage inputs)	250 VDC, 250 VAC
Isolation	300 VDC, 250 VAC from Earth Ground
Maximum Input (Volt x Hertz)	$8x10^6$ Volt x Hz normal mode input (across Voltage HI & LO). $1x10^6$ 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 an onboard memory and in a text file.
Temperature Range Operating	-10°C to 65°C
Temperature Range Storage	-40°C to 85°C
Size	Single 3U USB 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.

2.10 Accessories

Several accessories are available for the SM2060 series DMMs, which can be purchased directly from Signametrics Corp, or from one of its approved distributors or representatives. Some of the accessories available:

- DMM Probes SM-PRB ($15.70)
- 6 ft. USB 2.0 AM/BM cable SMU-CBL6ft
- 3 ft. USB 2.0 AM/BM cable SMU-CBL3ft
- 10 ft. USB 2.0 AM/BM cable SMU-CBL10ft
- 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).
- Lab View VI’s library SM204x.llb (free).
- Linux Driver kit (free).
- Extended 3 Year warrantee (calibration not included) $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 package includes the Digital Multimeter; Installation CD, a Floppy Disk containing the calibration and verification records, 6' USB cable and 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.

3.2 Installing the Software.

Before connecting the DMM Hardware, it is necessary to install the DMM software. Insert the Signametrics Product Installation CD into your CD drive. A menu will appear automatically on most computers. Otherwise, double-click on the autorun.exe file in the root directory of the Installation CD.

A menu will appear, allowing you to choose which Signametrics product to install. Select the product you would like to install, "SMU2055/2060/2064 USB DMMs". A Software Setup Wizard will begin. Follow the installation process, selecting which components you would like installed, and where they should be installed. The Hardware Driver and the Front Panel are required components to run and test the product. On the last page of the wizard, click Install.

After the software has been installed, The Signametrics USB Driver Wizard will appear. Click "next". A windows message may appear asking if you are sure you wish to install this driver. Continue the installation. Afterwards, you should now see a screen that indicates the drivers have been sucsessfully installed on this computer.

3.3 Installing the DMM Module

Warning

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.

Use extreme care when plugging the DMM module(s) into a USB port on your computer. Make sure no cable is connected to the front panel of the DMM while you plug it into the USB port.

Connect the SMU2055/2060/2064 to one of the USB ports on your computer. On Windows 2000, XP, or Vista a "Found New Hardware" Wizard dialog box should appear. On Windows 7, the drivers may automatically be detected and installed without a Found New Hardware Wizard Appearing.

The Wizard asks "Can Windows connect to Windows Update to search for software?" Select "No, not this time" and click on "Next". Select "Install the software automatically" and click on "Next". Windows should be able to find the drivers automatically since they were copied to the system (section 3.2). Windows may double check whether you want to install the software. If this is the case, click "Continue Anyways". The Wizard should say "The wizard has finished installing software for: [multimeter product name]". Click "Finish" to complete the installation.

3.4 Calibration File

The **SM60CAL.DAT** file supplied with your DMM has a unique calibration record for that DMM (See "**Calibration**" at the end of this manual.). In most cases, the installation of the calibration file is handled automatically by the DMM software.

A copy of the calibration file resides on an EEProm on the DMM and is copied to your computer the first time you use the instrument. A backup copy of the calibration file is included on a diskette that comes with the DMM.

The default location of the Calibration File is "C:\SM60CAL.DAT". If your system uses multiple DMMs, the software will append the Calibration Records of each DMM into a single SM60CAL.DAT file. The SM60CAL.DAT file is a text file, and can be opened using a text editor such as Notepad, should it be necessary.

3.5 DMM Input Terminals

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 consist of four Banana jacks.

V, 2Ω + This is the positive terminal for all Volts, 2-Wire Resistance and diode test. When in 4-Wire resistance measurement mode, it serves as the positive terminal of the current source. The maximum input across **V, 2Ω +** and **V, 2Ω -** is 240 VDC or 240 VAC**.**

V, 2Ω - This is the negative terminal for all Volts, 2-Wire Resistance and diode test. When in 4-Wire resistance measurement mode, it serves as the negative terminal of the current source. **Do not float this terminal or any other DMM terminal more than 240 VDC or 240 VAC above Earth Ground.**

I , 4Ω+ This is the positive terminal for all Current measurements. While in 4-Wire resistance measurement mode it is the high sense. The maximum input across **I, 4Ω +** and **I, 4Ω -** is **2.5 A**. Do not apply more than 5 V peak across the I,4**Ω**+ and I,4**Ω**- terminals.

I,4Ω - This is the negative terminal for Current measurements. While in 4-Wire resistance measurement mode it serves as the low sense. The maximum input across **I, 4WΩ +** and **I, 4WΩ -** is **2.5 A**. Do not apply more than 5 V peak across these two terminals.

The I,4Ω - Current function is protected by a 2.5 A, 250 V Fast Blow fuse (PCT type).

3.6 DMM Rear Panel

The rear panel includes various compliance and warning text and graphics, the unit serial number, its modle number and the installed options. The USB connector provides for both, compueter interface and power to run the DMM..

Figure 3-2. The Rear panel has the USB BF type connector. Compatible with BM cable.

3.7 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, Start→SMU2060 Series Multimeters→SMU2064 Multimeter. 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 for the first time, it takes a few extra seconds to extract its calibration data from the on-board memory, and write it to the calibration 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 SMU2055 front panelstarts up in DCV, 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 wth high end 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.

3.8 Using the Control Panel

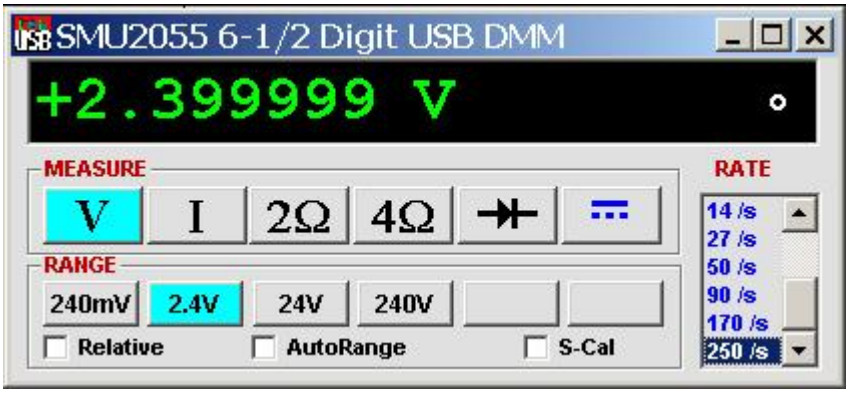

Figure 3-2**.** The Control Panel. The two main groups include Measurement Function and Range buttons. The Range buttons are context sensitive such that only "240m, 2.4, 24 and 240 appear when in a Voltage Measurement Function is selected, and 2.4m, 24m, 240m and 2.4 appear when a Current Measurement Function 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 Voltage and Current DMM functions.

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.

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 VAC 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. **Autorange is not recommended for automated test applications** 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, 2Ω+** and **V, 2Ω-** terminals, being certain to always leave the **I,4Ω+ and I,4Ω-** 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, 2Ω +** and **V, 2Ω -** terminals and perform **Relative** operation to eliminate zero errors. 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 2rps, 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. As with virtually all true-RMS measuring meters, the DMM may not read a perfect zero with a shorted input. This is normal.

ACV measurements, if possible, should have the NEUTRAL or GROUND attached to the **V,2Ω -** terminal. This prevents any "Common Mode" problems from occurring (Common Mode refers to floating the DMM **V,2Ω-** 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 frequency limits of the RMS functions (AC Voltage and current) are effected by both, the value of the input as a percentage of range, and the frequency of the signal.

Take into account the amount of time it takes for the RMS circuit to settle. With a setteling time of 0.5s, you may use a measurement rate of 2 readings/s and take two readings. The first will provide the 0.5s to settele, and the second will make a stable reading. Do not average since the first reading will introduce an error.

For improved accuracy and stability while measuring low frequency signals, the measurement time (1/rate) should be at least ten (10) times the period of the measured signal.

4.2 Current Measurements

The DMM measures AC and DC currents between 100 ηA and 2.5 A. Use the <u>+</u>**I, 4WΩ** terminals, being certain to always leave all other 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 internal to the DMM. 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 maximum burden voltage is about 250mV.

Warning! Applying voltages greater than 35 V to the I+, I- terminals can cause personal injury and/or damage to your DMM! Think before applying any inputs to these terminals!

4.2.1 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.

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 five. The **V, 2Ω +** provides the positive terminal and the **V, 2Ω-** negative terminal of the current source. 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

In the 2-Wire resistance measurement the DMM sources current and measure resuting voltage. The DMM measure Resistance using six ranges; 240Ω to 24 MΩ. Use the **V,2Ω+, V,2Ω-** terminals for this function. Disconnect the **I,2Ω+** and **I,2Ω-** terminals in order to reduce error due to leakage and noise, as well as better safety.

If the resistor to be measured is less than 24 kΩ, you may null out any lead resistance errors by first shorting the ends of the **V,2Ω+** and **V,2Ω-** test leads together and performing a **Relative** operation (**DMMSetRelative** under program control). Making measurements above 200 kΩ, you should consider shielded or twisted leads to minimize noise pickup. Further improvement can be achieved using guarding (section 4.3.5).

It is a good idea to be aware of the test voltages, particularly when measuring a circuite that includes semiconductors. To reduce this voltage, select a higher resistance range (lower current). For instance, measuring 10k resistor using the 24k range (100uA), results in 1V test voltage, which will turn on semiconductor junctions, resulting in lower resistance reading. To avoid this error, select the 240k range (10uA), which will result in 100mV and will read the 10k a lot more accurately (see section 2.3 for resistance ranges vs. current). For characterizing semiconductor part types, use the Diode measurement function.

For applications requiring voltage and current controlled resistance measurements, use the SMU2064, 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 **V,2Ω+** and **V,2Ω-** terminals apply a current source stimulus to the resistance, and the **I,4Ω+** and **I,4Ω-** 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.

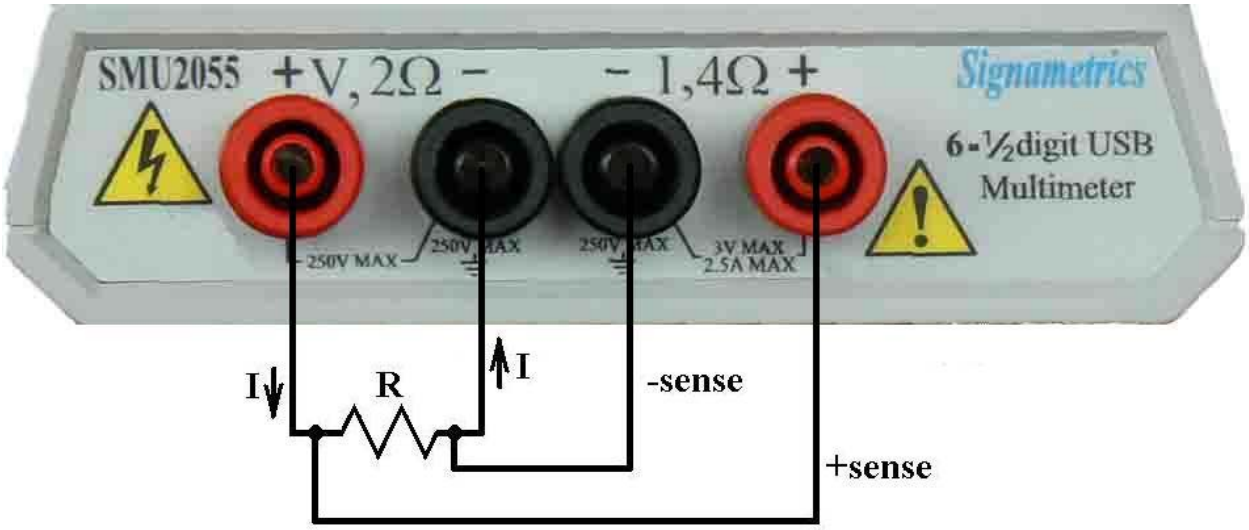

Figure 4-1. The **I,4Ω+** and **I,4Ω-** sense leads should be closest to the body of the resistor when making 4-Wire resistance measurements.

4.3.3 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.

Range	Ohms Current	DMM Resolution	Error due to 10 μV EMF	Error due to 100 μV EMF	Error due to 1mV EMF
240 Ω	1 mA	0.1mΩ	10 mΩ	100 mΩ	1 Ω
2.4 kΩ	1 mA	1 mΩ	10 mΩ	100 mΩ	1 Ω
24 kΩ	100 uA	10 mΩ	100 mΩ	1 Ω	10 Ω
240 kΩ	10 uA	0.1 Ω	1 Ω	10 Ω	100 Ω
2.4 MΩ	1 uA	1 Ω	10 Ω	100 Ω	10 Ω
24 MΩ	100 nA	10 Ω	100 Ω	1 kΩ	100 Ω

Figure 4-2. Resistance measurement errors contributed by Thermo-Voltaic offset.

4.4 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 forces a current and measures voltage drop. 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 maximum diode voltage that can be measured with this function is 2.4V. For higher voltages see the leakage test function.

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 SMU2060 Windows interface package provided, contains all required componenets for the following products: SMU2055, SMU2060 and SMU2064. It is a 32bit DLL based modules, which includes 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	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 the first time the DMM is used. It may contain calibration records for several DMM's.
SMU2060.LIB	The Windows import library. Install in a directory pointed to by your **LIB** environment variable.
SMU2060.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**).
SMU2060.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.
USBDMMUser.H	Header file containing all of the necessary DMM's function, range, rate definitions to be used with the various measure and source functions.

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 SMU2055
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 SMU2055
idc         ; IDC 240nA to 2.5A ranges. 1st entry is offset, 2nd is gain parameter
0   1.0                         ;Place holder on SMU2055
0   1.0                         ;Place holder on SMU2055
0   1.0                         ;Place holder on SMU2055
0   1.0                         ;Place holder on SMU2055
-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 (SM2060/55/64), 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 SMU2060 Driver Set With C++ or Similar Software

The SMU2055 uses the SMU2060 driver package. Install the **SMU2060.H** and **USBMMUser.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 **SMU2060.DEF** and **SMU2060.LIB** using **ImpDef.exe** and **ImpLib.exe**, provided with the compiler. Install **SMU2060.LIB** in a directory that will be searched by the linker for import libraries. The SMU2060 software must be installed prior to running any executable code. Install the **SMU2060.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 SMU2060 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().**

5.3 Visual Basic Front Panel Application

The Visual Basic front panel application, **SMU2064.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 SMU2060.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 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 **SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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);
```

DMMCloseUSB

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

#include "SMU2060.H"

int DMMCloseUSB(int *nDmm*)

Remarks This function is limited for servicing the DMM. It has no use in normal DMM operation. See also **DMMOpenUSB()** 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 = DMMCloseUSB(0);
```

DMMDelay

Description

Wait for a given time.

#include "SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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);
```

DMMGetDevLocation

Description Get a string containing the location of the DMM in the USB structure.

#include "SMU2060.h"

int DMMGetDevLocation(int *nDmm***, LPCSTR** *lpszLoc***)**

Remarks This service function retrieves the location of the USB DMM specified by *nDmm* in the USB bus. A zero terminated string of length 8 is returned in *lpszLoc*. This function must be used prior to opening the DMM. That is, prior to initializing or opening it by **DMMInit()** or **DMMOpenUSB()**.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszLoc	**LPCSTR** Points to a buffer (at least 8 characters long) to hold the result.

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

Value	Meaning
Negative Value	Error code
Positive Value < 100	The length of the returned string
Postive Value ≥ 100	Warning code

Example

```
char cBuf[8];
int i = DMMGetDevLocation(0, cBuf);
```

DMMGetDeviation

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

#include "SMU2060.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.

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 "SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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);
```

DMMGetHwOption

Description Get the hardware option installed in the DMM.

#include "SMU2060.h"

int DMMGetHwOption(int *nDmm*)

Remarks This function returns the hardware options installed. It returns a single character value corresponding to the option. For instance, if option ‘R’ in installed, 0X12, the ASCII equivalent or the letter ‘R’ is returned.

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 HWOption = DMMGetHwOption(0);
```

DMMGetID

Description Get DMM ID code.

#include "SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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);
```

DMMGetNumDevices

Description Get the number of USB DMM devices connected to the USB structure.

#include "SMU2060.h"

int DMMGetNumDevices(int * *nDevices***)**

Remarks This function retrieves the number of USB DMM devices connected to the USB bus. The number of devices is saved at a location pointed to by *nDevices*. This function must be used prior to opening the DMM. That is, prior to initialized or opening it by **DMMInit()** or **DMMOpenUSB()**. See also **DMMGetDevLocation()**.

Parameter	Type/Description
nDevices	*** int** Points to a location at which the number of devices is saved.

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

Value	Meaning
Negative Value	Error code
Positive Value < 100	The length of the returned string
Postive Value ≥ 100	Warning code

Example

```
int I; int number;
I = DMMGetNumDevices(& number);
```

DMMGetRange

Description Get DMM range code.

#include "SMU2060.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 "SMU2060.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);
```

DMMGetSupplyV

Description Returns the one of the DMM supplies voltages.

#include "SMU2060.H"

int DMMGetSupplyV(int *nDmm,* **, double** ***lpdVoltage**)

Remarks This function makes a measurement of one of the DMM power supplies voltages as an indication of the USB supply voltage level. The nominal value is -12V. The USB interconnect and some off the shelve hubs can make this voltage higher or lower than is required. The acceptable value should be -10.5 to -13.5V. Voltages higher than -13.5V may damage the SMU2055 and voltages below -9.0 are inadequate for proper operation, and is usually indicative of poor USB cable. The value of this voltage is stored at a double precision location pointed to by *lpdVoltage*.

Parameter	**Type/Description**
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdVp;tage	**LPSTD** Points to a double 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
Positive Value	Warning code

Example

```
double v;
int status = DMMGetSupplyV(0, @v);
```

DMMGetStoredReading

Description Get a single stored reding.

#include "SMU2060.h"

int DMMGetStoredReading(int *nDmm***, int** *iIndex***, double** ***lpdRdng***)**

Remarks User this function to retrieve readings previously captured by **DMMReadNsamples.** Return a double precision reading number *iIndex* by placing it at a location pointed to by lpdRdng. iIndex can have a value between 0 and the total number of measuremens taken by **DMMReadNsamples** minus 1 (iN – 1).

Parameter	**Type/Description**

nDmm	**int**	Identifies the DMM. DMMs are numbered starting with zero.
iIndex	**int**	Index to the stored reading, can be 0 to the total of number of readings taken mins 1.
lpdRdmg	**double ***	Pointer where the reading value is to be saved.

Return Value Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
double  v;

int status = DMMGetStoredReading(0, 0, &v); // get the 1st
reading
```

DMMGetType

Description Get the type of the DMM.

#include "SMU2060.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
Other positive value	Warning code
Negative Value	Error code

Example

```
int DMMtype = DMMGetType(0);
```

DMMGetVer

Description Get DMM software driver version.

#include "SMU2060.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 "SMU2060.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 "SMU2060.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.

Return Value TRUE, FALSE or an error code.

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

Example

```
int autorange = DMMIsAutoRange(0);
```

DMMIsInitialized

Description Get the status of the DMM.

#include "SMU2060.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** function. 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 "SMU2060.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);
```

DMMOpenUSB

Description A service function which open the USB bus for the SMU2055. Not for user application.

#include "SMU2060.H"

int DMMOpenUSB(int *nDmm*)

Remarks This function is limited for servicing the DMM. It has no use in normal DMM operation.. See also **DMMCloseUSB()** 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 = DMMOpenUSB(0);
```

DMMRead

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

#include "SMU2060.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 "SMU2060.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);`

DMMReadNsamples

Description Take a reading that is in base value.

#include "SMU2060.h"

int DMMReadNsamples(int *nDmm*, **int** *iN***)**

Remarks In response to this command the DMM take *iN* measurements, and sends them back to the USB bus. In order not to loose any, and cause overrun, use **DMMGetStoredReading()** in a tight loop. Measurements are made using the currently selected function, range and aperture.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iN	**Int** The number of measurements to be taken. This value must be between 2 and 10,000.

Return Value Integer value version code or an error code.

Value	Meaning
POS_FS or **NEG_FS**	Positive or Negative Full Scale, or overrange
Negative Value	Error code
DMM_OKAY	No error

Example `int status = DMMReadNsamples(0, 100);`

DMMReadStr

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

#include "SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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 SMU2055 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 "SMU2060.h"
#include "USBDMMUser.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 0.5rps (RATE_p5) to 250rps (RATE_250). Some of the rates have specific power line rejection as indicated in the specification part of this manual. See DMM

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_PARAMETER	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 "SMU2060.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 "SMU2060.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 "SMU2060.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 "SMU2060.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");
```

SetGain

Description Set currently set gain during service.

#include "SMU2060.H"
#include "UseroDMM.h"

int SetGain(int *nDmm*, **doulbe** *Gain***)**

Remarks This function sets the currently set gain,. Sets the gain of the the currently selected function and range. The gain is returned as double-precision floating-point number *Gaint*. This function is useful while performaing calibration. Set **GetGain()** 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** the gain.
DMM_OKAY	Valid return.
Negative Value	Error code

Example

```
SetGain(0, 1.00023); // set gain
```

GetGain

Description Retrieve currently set gain.

#include "SMU2060.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 "SMU2060.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 during service.
#include "SMU2060.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 "SMU2060.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 during service.

#include "SMU2060.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 only.

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 during calibration.
#include "SMU2060.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 and calibration 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 "SMU2060.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
```

WrCalFileToStore

Description Transfer the contents of a cal file to the on-board cal store.
#include "SMU2060.h"

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

Remarks This function copies the specified calibration file, pointed to by lpszCal, to the on-board none volatile store of the DMM. This is appropriate following calibration operation. The currently stored on-board record is replaced with the contents of the speified file. Make sure that the calibration file only contains only one record, for the specified DMM.

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 C:\ root directory.

Return Value Integer error code.

Value	Meaning
Any value not 0	**int** Eror or warning code

Example int i = **WrCalFileToStore** (0, `"C:\\SM60CAL.dat");`

WrCalStoreToFile

Description Transfer the contents of the on-board cal store to a file.
#include "SMU2060.h"

int WrCalStoreToFile (int *nDmm*,**LPCSTR** *lpszCal*, **int** *mode***)**

Remarks This function copies the calibration record stored in the on-board none volatile memory of the DMM to the specified calibration file, pointed to by lpszCal. If *mode* is 'a' and a file exists, the record is appended to the end of this file. If *mode* is 'w', a new file is created, wiping out the old if it exists.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
mode	**int** Sets the file creation mode.
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 C:\ root directory.

Return Value Integer error code.

Value	Meaning
Any value not 0	**int** Eror or warning code

Example int i = **WrCalStoreToFile** (0, `"C:\\SM60CAL.dat", 'a');`

EraseCalStore

Description Service function that wipes the Calibration record off the internal memory.
#include "SMU2060.h"

int EraseCalStore(int *nDmm***)**

Remarks This function reformats the none volatile calibration store on-board the DMM, preparing it for storing a calibration record. This function will remove the currently stored calibration record and is not necessary during calibration.

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

Return Value Integer error code.

Value	Meaning
Any value	**int** Eror or warning code.

Example int i = EraseCalStore(0); `// Erase/Format cal store EEProm`

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_INIT            -1    // DMM is not initialized. Use DMMInit() prior to using
DMM_E_CAL_R           -2    // cannot find valid calibration record
ERR_AD_HW             -3    // A/D does not respond. H/W error
NO_CAL_RECORD         -4    // can't find cal record for DMM
TRIG_ERR              -5    // Trigger circuit error
GUARD_COM             -6    // Microcontroller communication error
TIMEOUT               -7    // process timed out Error
GUARD_XING                  -8      // Guard crossing is broken
WRONG_TYPE            -9    // Wrong Cal record for DMM type
UNKNOWN_ERROR         -10   // Undefined Error
CANT_OPEN_USB         -11   // can't open USB device. Already open or an SMU403X
GENERAL_ERR           -12   // General Error
CAL_STORE             -13   // Error reading Cal record from local storage
CREAT_CAL_FILE        -14   // can't create named cal file to write to
OPEN_CAL_FILE         -15    // can't open cal file for reading cal record
CREAT_CAL_RCRD        -16    // can't create on-board Cal Record
ERROR_EEPROM_DTYPE    -17    // Invalid dmm type in EEProm
ERROR_READBYTES       -18     // unexpected number of bytes read
ERROR_WRITEBYTES      -19    // unexpected number of bytes written
ERROR_DTYPE           -20   // invalid input, bad DMM Type parameter
ERROR_READ_EEPROM     -21    // invalid data on the EEPROM
ERROR_USB_IO          -22   // I/O error from USB bus
ERROR_USB_PWR         -23   // USB 5V supply is too low
MCU_COM_ERROR         -24   // Microcontroller communication error
```

5.10 Warning Codes

Following a warning, the DMM will continue to run normally with the exception of the fault indicated by 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 additional iterations, or the POS_FS and NEG_FS are indication that the signal level is too high for the selected range, which is normal.

```
APERTR_TOO_HIGH 101   // Aperture value is too high for operation
DMM_E_FUNC      102   // Invalid function value used
DMM_E_RNG       103   // Invalid range value used
DMM_CNT_RNG     104   // DMM counter out of range
DMM_E_IS_INIT   105   // Dmm already initialized: in use
CAP_RATE_ERR    106   // Can't change rate in Cap mode.
ERR_FUNC        107   // Illegal function selection
ERR_APERTURE    108   // Wrong Aperture selected, see rate definition
```

```
TRIG_SAMPL_ERR   109   // Wrong number of Trigger samples
ERR_PARAMETER    110   // wrong parameter value
UN_CALIBRATED    111   // Expired Calibration. Needs service
TOO_COLD         112   // Temperature too low
TOO_HOT          113   // Temperature too high
BAD_TC_TYPE      114   // Wrong TC type
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.
READ_INTERVL     120   // Read Interval value incompatible with Aperture,
FAIL_OPEN_CAL    121    // Failed to perform Open-Cal operation
CAL_2usOffset    122   // Failed to Cal offset in 2.5uS Aperture
CAL_2usGain      123   // Failed to Cal gain in 2.5uS Aperture
USB_LOW_POWER    124   // USB supply is too low for this opereation
 USB_HIGH_POWER  125   // USB supply is too high
WRONG_GRD_VER    126   // MCU Firmwhare does not support operation
```

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     // four AC and DC voltage 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 Rates. Use **DMMSetRate()** and **DMMGetRate()** to set and retrieve the currently set measurement rate..

```
#define RATE_p5            0          // 0.5rps with 50/60Hz rejection
#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_50            50         // 50rps with 60Hz rejection
#define RATE_90            90         // 90rps with 50Hz rejection
#define RATE_170           170        // 170rps
#define RATE_250           250        // 250rps
```

6 Calibration

Each SMU2055 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 SMU2055 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 06/15/2008
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 2/3rd of the top of each range. Calibration of your SM/SMU2055/60/64 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 SMU2060, SMU2064 and SMU2055 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