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

5990-9495EN

Agilent Technologies

Operator's Manual

Model SMX2040 6½ Digit Digital Multimeter
Model SMX2042 6½ Digit Multi-Function Digital Multimeter
Model SMX2044 6½ Digit LCR Sourcing Digital Multimeter

Signametrics Corporation

December, 2004

CAUTION

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

TABLE OF CONTENTS

1.0 Introduction

Congratulations! You have purchased a PXI/CompactPCI Plug-in instrument with analog and systems performance that rivals the best, all-in-one box instruments. The SMX2040 series digital multimeters (DMMs) are easy to setup and use, have sophisticated analog and digital circuitry to provide very repeatable measurements, and are protected to handle any unexpected situations your measurement environment may encounter. To get years of reliable service from these DMMs, please take a few moments and review this manual before installing and using this precision instrument.

This manual describes the SMX2040 and SMX2044 DMMs. Each DMM delivers unmatched breakthrough performance in a PXI and CompactPCI plug-in instrument. With a rich repertoire of functions, the SMX2040 series out performs all other plug-in DMMs, including the trusted Signametrics SM-2020CT, and most brand named bench top units.

Note: In this manual, all references to the "SMX2040" and “DMM” apply to the SMX2040 and SMX2044. Features unique to the SMX2044 will be identified as such.

1.1 Safety Considerations

Safety Considerations

The SMX2040 series of DMMs is capable of measuring up to 300 VDC or 250 VAC across the Volt HI and LO terminals, and can also measure common mode signals that "float" the DMM above EARTH ground by up to 300 VDC or 250 VAC. When making common mode measurements, the majority of the circuits inside the DMM are at the common mode voltage. **These voltages can be lethal and can KILL! During and after installing your DMM, check to see that there are no wires or ribbon cables from your PXI/CompactPCI chassis trapped inside the DMM.**

The DMM comes installed with two shields (bottom and top) that **must not be removed for performance as well as safety reasons.** Removal of these shields and/or improper assembly of the shields can result in lethal voltages occurring within your chassis. Also make sure the chassis is 3U in size.

Warning

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

To avoid shock hazard, install the DMM only into a 3U PXI and CompactPCI chassis that has its power connector connected to a power receptacle with an earth safety ground.

When making any measurements above 50 VDC or 40 VAC, only use Safety Test Leads. Examples of these are the Signametrics Basic Test Leads and Deluxe Test Leads, offered as an optional accessory with the Signametrics DMMs. **Do not** use these units in a **6U** chassis as an **electrocution hazard will be present**.

1.2 Minimum Requirements

The SMX2040 series of system DMMs are precision plug-in modules that are compatible with 3U PXI or CompactPCI chassis. The processor type must be a Pentium or equivalent processor running MS Windows. The DMM requires a single PXI or CompactPCI slot. A mouse must be installed when controlling the DMM from the Windows Control Panel. The SMX2040 comes with a Windows' DLL, for operation with Windows' Version 95/98/Me/2000/XP and NT4.0.

1.3 Feature Set

The base unit, the SMX2040, has 6-1/2 digit performance and can be used as a general purpose DMM, giving very accurate and stable readings. The SMX2044 adds to the SMX2040 additional measurement functions not found in other DMMs such as inductance measurement and sourcing capabilities.

SMX2040, 42, 44 6½ Digit DMMs feature table:

Function	SMX2040 DMM	SMX2042 Multi-Function DMM	SMX2044 LCR Sourcing DMM
DCV 4 ranges, >10 GΩ & 10 MΩ input resistance.	√	√	√
ACV 4 ranges, 1 MΩ input	√	√	√
2-Wire Ohms, six ranges 330 Ω to 33 MΩ	√	***plus*** 33 Ω, 330 MΩ	***plus*** 33 Ω, 330 MΩ
4-Wire Ohms, four ranges 330 Ω to 330 kΩ	√	***plus*** 33 Ω	***plus*** 33 Ω range
Offset Ohms	√	√	√
DC current, four ranges 3.3 mA to 2.5 A	√	√	√
AC current, four ranges 3.3 mA to 2.5 A	√	√	√
Diode V/I characteristics at 100 ηA to 1mA	√	***plus*** 10 mA	***plus*** 10 mA
Auto range, Relative	√	√	√
Min/Max, dB and percent deviation functions	√	√	√
On board measurement buffer	√	√	√
Measurement rate: 0.2 to 1,000/sec	√	√	√
External and threshold trigger	√	√	√
Thermocouples	√	√	√
High Dynamic range; ±3,300,000 counts	√	√	√
PXI Trigger In/Out	√	√	√
Capacitance, seven ranges, 10 nF to 10 mF		√	√
Temperature (five basic RTD types)		√	√
Frequency / Period measurement		√	√
Pulse width, pos./neg., & duty cycle		√	√
Totalizer/event counter		√	√
Variable threshold DAC; all timing measure.		√	√
Peak to Peak, Crest factor, Median		√	√
Internal DMM temperature sensor		√	√
Six wire Ohms (with force/sense)			√
Inductance, six ranges 33 μH to 3.3 H			√
DCV source 0 to +/-10.0 V			√
ACV source 0 to 20 V pk-pk, 2 Hz to 75 KHz			√
DC current source, 1 nA to 12.5 mA			√
Leakage measurement			√
Synthesized resistance source			√
Extended Resistance measurements			√
In Circuit Capacitance			√

2.0 Specifications

2.1 DC Voltage Measurement

Input Characteristics

- **Input Resistance** 330 mV & 3.3 V Ranges: >10 GΩ ,
- **Input Resistance** 33 V & 330 V Ranges: 10.0 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
330 mV	330.0000 mV	100 ηV	0.003 + 4.5 μV	0.004 + 5.5 μV	0.007 + 8 μV
3.3 V	3.300000 V	1 μV	0.002 + 10 μV	0.0025 + 12 μV	0.0045 + 17 μV
33 V	33.00000 V	10 μV	0.003 + 250 μV	0.004 + 280 μV	0.007 + 330 μV
330 V	330.0000 V	100 μV	0.004 + 1 mV	0.005 + 1.2 mV	0.008 + 1.5 mV

[1] With reading rate set to 10 readings per second (rps) or slower, and within one hour of DCV zero, using Relative control.

DCV Noise Rejection Normal Mode Rejection, at 50, 60, or 400 Hz ± 0.5%, is better than 95 dB for reading rates of 10 rps or lower. Common Mode Rejection (with 1 kΩ lead imbalance) is better than 120 dB for these conditions.

2.2 DC Current Measurement

Input Characteristics

- **Burden Voltage** < 350 mV for all ranges
- **Protected** with 2.5A fuse (5x20mm, 250 V Fast)

Accuracy ± (% of reading + Amps) [1]

Range	Full Scale 5 ½ Digits	Resolution	24 hours 23°C ± 5°C	90 Days 23°C ± 5°C	One Year 23°C ± 5°C
3.3 mA	3.30000 mA	10 ηA	0.052 + 200 ηA	0.07 + 350 ηA	0.1 + 400 ηA
33 mA	33.0000 mA	100 ηA	0.04 + 1 μA	0.06 + 2 μA	0.1 + 3 μA
330 mA	330.000 mA	1 μA	0.05 + 30 μA	0.055 + 40 μA	0.075 + 60 μA
2.5 A	2.50000 A	10 μA	0.55 + 50 μA	0.6 + 200 μA	0.65 + 350 μA

[1] With reading rate set to 10 rps or slower, and within one hour of DCI zero, using Relative control.

2.3 AC Voltage Measurements

Input Characteristics

- **Input Resistance** 1.0 MΩ, shunted by < 100 pF, all ranges
- **Crest Factor** 3 at Full Scale, increasing to 7 at Lowest Specified Voltage
- **AC coupled** Specified range: 10 Hz to 100 kHz
- **Typical Settling time** < 0.5 sec to within 0.1% of final value

2.3.1 AC Voltage True RMS Measurement

Range	Full Scale 6 ½ Digits	Lowest specified Voltage	Resolution
330 mV	330.0000 mV	5 mV [1]	100 ηV
3.3 V	3.300000 V	10 mV	1 μV
33 V	33.00000 V	100 mV	10 μV
250 V [2]	250.0000 V	1 V	100 μV

[1] Between 5 mV and 10 mV, add 100 μV additional error to the accuracy table below. In many computer installations, if the DMM is not near a noisy board, usable voltage measurements of 1 mV can be obtained.

[2] 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.

		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
330 mV	10 Hz - 20 Hz	3.0 + 350 µV	3.1 + 380 µV	3.2 + 430 µV
	20 Hz - 47 Hz	0.92 + 150 µV	0.93 + 170 µV	0.95 + 200 µV
	47 Hz - 10 kHz	0.13 + 100 µV	0.14 + 110 µV	0.15 + 120 µV
	10 kHz - 50 kHz	0.55 + 160 µV	0.6 + 200 µV	0.63 + 230 µV
	50 kHz - 100 kHz	5.3 + 350 µV	5.4 + 370 µV	5.6 + 400 µV
3.3 V	10 Hz - 20 Hz	3.0 + 2 mV	3.1 + 2.2 mV	3.2 + 2.5 mV
	20 Hz - 47 Hz	0.93 + 1.3 mV	0.96 + 1.5 mV	1.0 + 1.7 mV
	47 Hz - 10 kHz	0.05 + 1 mV	0.055 + 1.1 mV	0.065 + 1.2 mV
	10 kHz - 50 kHz	0.62 + 1.2 mV	0.65 + 1.3 mV	0.70 + 1.5 mV
	50 kHz - 100 kHz	5.1 + 1.5 mV	5.2 + 1.7 mV	5.3 + 2 mV
33 V	10 Hz - 20 Hz	3.0 + 14 mV	3.1 + 16 mV	3.3 + 20 mV
	20 Hz - 47 Hz	0.93 + 12 mV	0.96 + 14 mV	1.0 + 16 mV
	47 Hz - 10 kHz	0.06 + 10 mV	0.065 + 11 mV	0.073 + 13 mV
	10 kHz - 50 kHz	0.31 + 18 mV	0.33 + 21 mV	0.35 + 25 mV
	50 kHz - 100 kHz	2.0 + 30 mV	2.2 + 35 mV	2.4 + 40 mV
250 V	10 Hz - 20 Hz	3.0 + 140 mV	3.1 + 160 mV	3.3 + 200 mV
	20 Hz - 47 Hz	0.93 + 120 mV	0.96 + 130 mV	1.0 + 150 mV
	47 Hz - 10 kHz	0.04 + 100 mV	0.045 + 110 mV	0.06 + 130 mV
	10 kHz - 50 kHz	0.32 + 150 mV	0.4 + 170 mV	0.45 + 200 mV
	50 kHz - 100 kHz	2.5 + 200 mV	2.8 + 240 mV	3.2 + 300 mV

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

2.3.2 AC Peak-to-Peak Measurement (SMX2044)

- Measures the peak-to-peak value of a repetitive waveform

ACV Range	Lowest specified input voltage (Vp-p)	Full Scale reading (Vp-p)	Resolution	Typical Accuracy 23°C ± 5°C One Year [1]
330 mV	0.1 V	1.85 V	1 mV	1.5 ±10 mV
3.3 V	1.0 V	18.5 V	10 mV	1.4 ±70 mV
33 V	10 V	185.0 V	100 mV	1.0 ±700 mV
250 V	100 V	850.0 V	1 V	1.0 ± 6 V

[1] Specified from 30Hz to 10 kHz. Input signal frequency of 30 Hz to 30 kHz.

2.3.3 AC Crest Factor Measurement (SMX2044)

- Measures the crest factor (peak / RMS) of a repetitive waveform

ACV Range	Lowest specified input voltage (Vp-p)	Highest specified input voltages (Vp-p)	Resolution	Typical Accuracy 23°C ± 5°C One Year [1]
330 mV	0.1 V	1.8 V	0.01	2.2 ±0.3
3.3 V	1.0 V	18 V	0.01	2.1 ±0.1
33 V	10 V	180 V	0.01	2.0 ±0.1
250 V	100 V	700 V	0.01	2.0 ±0.1

[1] Crest factor measurement requires signal frequency of 30 Hz to 30 kHz.

2.3.4 AC Median Value Measurement (SMX2044)

- Measures the mid-point between the positive and negative peaks of a repetitive waveform
- Used to determine the Threshold DAC setting for optimal frequency and timing measurements

ACV Range	Lowest specified input voltage (Vp-p)	Full Scale reading	Resolution	Typical Accuracy 23°C ± 5°C One Year [1]
330 mV	0.08 V	±0.95 V	1 mV	2.0% ±17 mV
3.3 V	0.80 V	±9.5 V	10 mV	3% ±160 mV
33 V	8 V	±95.0 V	100 mV	3% ±1.4 V
250 V	80 V	±350.0 V	1 V	3% ±12 V

[1] Median measurements require a repetitive signal with frequency range of 30 Hz to 30 KHz.

2.4 AC Current Measurement, True RMS

Input Characteristics

- **Burden Voltage** < 350 mV RMS all Ranges
- **Crest Factor** 3 at Full Scale, increasing to 7 at Lowest Specified Current
- **Protected** with 2.5 A fuse (5x20 mm, 250 V Fast)

Range	Full Scale 6 1/2 Digits	Lowest Specified Current	Resolution
3.3 mA	3.300000 mA	50 µA	1 nA
33 mA	33.00000 mA	500 µA	10 nA
330 mA	330.0000 mA	5 mA	100 nA
2.5 A	2.500000 A	50 mA	1 uA

Accuracy ± (% of reading + Amps)

Range	Frequency	24 hours 23°C ± 1°C	90 Days 23°C ± 10°C	One Year 23°C ± 10°C
3.3 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
33 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
330 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	0.22 + 220 µA
	1 kHz - 10 kHz	0.3 + 300 µA	0.4 + 350 µA	0.6 + 400 µA
2.5 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	0.6 + 3.8mA	0.63 + 3.8 mA	0.65 + 4 mA
	1 kHz - 10 kHz	0.6 + 4mA	0.62 + 4.5 mA	0.7 + 5 mA

Note: All AC Current ranges have typical measurement capability to 20 kHz.

2.5 Resistance Measurements

2.5.1 2-wire and 4-wire

Accuracy ± (% of reading + Ω) [1]

Range [2]	Full Scale 6 ½ Digits	Resolution	Source current	24 hours 23°C ± 1°C	90 Days 23°C ± 10°C	One Year 23°C ± 10°C
33 Ω [3]	33.00000 Ω	10 μΩ	10 mA	0.0038 + 1 mΩ	0.005 + 1.5 mΩ	0.008 + 2 mΩ
330 Ω	330.0000 Ω	100 μΩ	1 mA	0.0037 + 4.5 mΩ	0.0046 + 5 mΩ	0.007 + 6 mΩ
3.3 kΩ	3.300000 kΩ	1 mΩ	1 mA	0.0023 + 28 mΩ	0.003 + 32 mΩ	0.005 + 33 mΩ
33 kΩ	33.00000 kΩ	10 mΩ	100 μA	0.0025 + 300 mΩ	0.0033 + 330 mΩ	0.006 + 350 mΩ
330 kΩ	330.0000 kΩ	100 mΩ	10 μA	0.0055 + 3.2 Ω	0.007 + 4 Ω	0.009 + 5 Ω
3.3 MΩ	3.300000 MΩ	1 Ω	1 μA	0.018 + 40 Ω	0.03 + 50 Ω	0.04 + 70 Ω
33 MΩ	33.0000 MΩ	100 Ω	100 nA	0.12 + 400 Ω	0.13 + 500 Ω	0.2 + 600 Ω
330 MΩ [3]	330.00 MΩ	10 kΩ	10 nA	1 + 50 kΩ	1.4 + 60 kΩ	2.0 + 80 kΩ

[1] With reading rate set to 2 rps or slower, and within one hour of Ohms zero, using Relative control.
[2] 4-wire ohms is available up to the 330 kΩ range.
[3] 33 Ω and 330 MΩ ranges are only available with the SMX2042,44.

2.5.2 6-wire Guarded Resistance Measurement (SMX2044)

Typical additional error contributed by guarding

Accuracy ± (% of reading + Ω)

Range	Source current	One Year 23°C ± 5°C [1]
33 Ω	10 mA	0.3 + 4 mΩ
330 Ω	1 mA	0.003 + 20 mΩ
3.3 kΩ	1 mA	0.005 + 100 mΩ
33 kΩ	100 μA	0.03 + 1 Ω
330 kΩ	10 μA	0.35 + 10 Ω

[1] This table should be used in conjunction with the 2-wire and 4-wire table above.

2.6 Leakage Measurement (SMX2044)

Accuracy ± (% of reading + Ω) [1]

Leakage Reading	Voltage range	One Year 23°C ± 5°C [1]
1.00 ηA to 100.00 ηA	-10 V to +10 V	2 + 350 pA
100.00 ηA to 1000.00ηA	-9 V to + 9 V	1.2 + 2 ηA
1000.00 ηA to 3.3 μA	-7 V to + 7 V	1.5 + 20 ηA

[1] Error does not include external shunt resistor's tolerance.

2.7 RTD Temperature Measurement (SMX2044)

RTD Type	Ro (Ω)	Resolution	Temperature range	Temperature Accuracy 23°C ± 5°C [1] One Year
pt385, pt3911, pt3916, pt3926	100, 200 Ω	0.01°C	-150 to 650°C	±0.06°C
pt385, pt3911, pt3916, pt3926	500, 1 kΩ	0.01°C	-150 to 650°C	±0.03°C
Cu (Copper)	Less than 12 Ω	0.01°C	-100 to 200°C	±0.18°C for temperatures ≤ 20°C, ±0.05°C otherwise
Cu (Copper)	Higher than 90 Ω	0.01°C	-100 to 200°C	±0.10°C for temperatures ≤ 20°C, ±0.05°C otherwise

[1] With reading rate set to 2 rps or slower, using a 4-wire RTD. Measurement accuracy does not include RTD probe error.

2.8 Additional Component Measurements

2.8.1 Diode Characterization

- **Available DC current values** 100 ηA, 1 μA, 10 μA, 100 μA and 1 mA. SMX2042 and SMX2044 add 10 mA. SMX2044 has a variable current from 10 ηA to 12.5 mA using the DCI source.
- **Typical Current Value Uncertainty** 1%
- **Typical Voltage Value Uncertainty** 0.02%
- **Maximum diode voltage compliance** 4 V

2.8.2 Capacitance Measurement (SMX2042, 44)

Accuracy ± (% of reading + Farads) [1]

Range	Full Scale 4 ½ Digits	Resolution	One Year 23°C ± 5°C
10 ηF	11.999 ηF	1 pF	2.1 ± 5 pF
100 ηF	119.99 ηF	10 pF	1.0
1 μF	1.1999 μF	100 pF	1.0
10 μF	11.999 μF	1 ηF	1.0
100 μF	119.99 μF	10 ηF	1.0
1 mF	1.1999 mF	100 ηF	1.2
10 mF	11.999 mF	1 μF	2

[1] Within one hour of zero, using Relative control. Accuracy is specified for values higher than 5% of the selected range with the exception of the 10 ηF range, which measures down to 0 pF.

2.8.3 Inductance Measurement (SMX2044)

± (% of reading + inductance) [1]

Range	Default frequency	Full Scale 4 ½ Digits	Resolution	Accuracy 23°C ± 5°C One Year [2]
33 μH	75 kHz	33.000 μH	1 ηH	3.0% + 500 ηH
330 μH	50 kHz	330.00 μH	10 ηH	2.0% + 3 μH
3.3 mH	4 kHz	3.3000 mH	100 ηH	1.5% + 25 μH
33 mH	1.5 kHz	33.000 mH	1 μH	1.5% + 200 μH
330 mH	1 kHz	330.00 mH	10 μH	2.5 + 3 mH
3.3 H	100 Hz	3.3000 H	100 μH	3 + 35 mH

[1] Within one hour of zero, and Open Terminal Calibration.
[2] Accuracy is specified for values greater than 5% of the selected range.

2.8.4 In Circuit AC-Based Capacitance Measurements (SM2044)

Accuracy ± (% of reading + Farads) [1]

Range	Full Scale 4 ½ Digits	Resolution	One Year 23°C ± 5°C
33 ηF	32.99 ηF	10 pF	12% ± 250 pF
330 ηF	329.9 ηF	100 pF	5% ± 500 pF
3.3 μF	3.299 μF	1000 pF	4% ± 1 ηF
33 μF	32.99 μF	10 ηF	5% ± 20 ηF
330 μF	329.9 μF	100 ηF	5% ± 1 μF
3.3 mF	3.299 mF	1 μF	7% ± 50 μF

[1] Specified to 2/3 of range (ie. 22ηF on 33nF range). Within one hour from last AC-Caps Open calibration. Add an error of 50e-6*R*C (%) due to paralled resistance.

2.9 Timing Measurements (SMX2042, 44)

2.9.1 Threshold DAC

- **The Threshold DAC is used for selecting a detection threshold to give optimal frequency and timing measurements.**

± (% of setting + volts)

Selected VAC range [1]	Threshold range (DC level)	Threshold DAC resolution	Highest allowed input Vp-p	Typical one year setting uncertainty
330 mV	-1.0 V to +1.0 V	0.5 mV	1.900 V	0.2% + 4 mV
3.3 V	-10.0 V to +10.0 V	5.0 mV	19.00 V	0.2% + 40 mV
33 V	-100.0 V to 100.0 V	50 mV	190.0 V	0.2% + 0.4 V
250 V	-500 V to 500 V	500 V	850.0 V	0.2% + 4 V

[1] This table should be used in conjunction with the AC volts section above.

2.9.2 Frequency and Period Measurement

ACV Mode

- **Input Impedance** 1 MΩ with < 300 pF

Frequency Range	1 Hz - 100 Hz	100 Hz-1 kHz	1 kHz-10 kHz	10 kHz-100 kHz	100 kHz-300 kHz
Resolution	1 mHz	10 mHz	100 mHz	1 Hz	1 Hz
Uncertainty is ±0.002% of reading ± adder shown	4 mHz	20 mHz	200 mHz	2 Hz	5 Hz
Input Signal Range [1]	10% - 200% of range	10% - 200% of range	10% -200% of range	10% - 200% of range	45% -200% of range

[1] Input RMS voltage required for a valid reading. Do not exceed 250 V RMS input. For example, 10% -200% of range indicates that in the 330 mVAC range, the input voltage should be 33 mV to 660 mV RMS.

ACI Mode

- **Input Impedance** 10 Ω in the 3 mA and 30 mA ranges, 0.1 Ω in the 330 mA and 2.5 A ranges.

Frequency Range	1 Hz - 100 Hz	100 Hz-1 kHz	1 kHz-10 kHz	10 kHz-500 kHz
Resolution	1 mHz	10 mHz	100 mHz	1 Hz
Uncertainty	0.01% ±4 mHz	0.01% ±20 mHz	0.01% ±200 mHz	0.01% ±2 Hz
Input Signal Range, 3.3 mA, 330mA Ranges [1]	10% -500% of range	10% - 500% of range	10% -500% of range	10% - 500% of range
Input Signal Range, 33 mA, 2.5A ranges	50% -100% of range	50% - 100% of range	50% - 100% of range	50% - 100% of range

[1] Input current required to give a valid reading. For example, 10% -500% of range indicates that in the 3.3 mA range, the input current should be 0.33 mA to 16.5 mA.

2.9.3 Duty Cycle Measurement

Frequency Range	1 Hz to 100 Hz	100 Hz to 1 kHz	1 kHz to 10 kHz	10 kHz to 100 kHz
Resolution	0.02%	0.2%	2%	20%
Typical Uncertainty is ±0.03% of reading ± adder shown	0.03%	0.3%	3%	20%
Full scale reading	100.00 %	100.00 %	100.00 %	100.00 %

2.9.4 Pulse Width

± (% of reading + sec)

Polarity	Frequency range	Resolution	Width range	Typical Uncertainty
Positive or negative pulse widths	1 Hz to 100 kHz	2 µs	2 µs to 1 s	0.01 +/- 4 µs

2.9.5 Totalizer

- Active edge polarity: Positive or negative transition
- Maximum count: 10^9
- Allowed rate: 1 to 30,000 events per second
- Uses Threshold DAC

2.10 Trigger Functions

2.10.1 External Hardware Trigger (at DIN-7 connector)

Trigger Input voltage level range	High: +3V to +15V, Low: -15V to +0.8V
Ttrigger High current drive	Min. 1mA, Max 10mA (TTL or CMOS logic level)
Timing Characteristics	Trigger occurs within 2/Reading rate
Trigger Activation	Positive or Negative edge depending on trigger command.
Internal Reading Buffer	Up to 1,000 readings/sec into 64 locations reading buffer
Isolation of trigger input	±50 V from analog DMM inputs, and from computer chassis earth ground.

2.10.2 PXI Bus Hardware Trigger Inputs (at PXI J2)

Trigger Input	TTL or CMOS positive pulse
Trigger Pulse Width	Minimum 250µS
Internal Reading Buffer	up to 1,000 readings/sec into 64 readings buffer
Selectable lines	PXI_TRIG1,2,3,4,5,6,7 and PXI_STAR
Isolation from DMM inputs	±330 V from any of the DMM 4 main inputs terminals

2.10.3 PXI Bus Hardware Trigger Output (to PXI J2)

Trigger Output	TTL or CMOS negative pulse. Positive edge = ready
Trigger Pulse Width	Approximately 140µS
Activity	A single pulse is issued for each A/D conversion (at 10 or higher measurement rate)
Selectable lines	PXI_TRIG1,2,3,4,5,6 and PXI_STAR
Isolation from DMM inputs	±330 V from any of the DMM 4 main inputs terminals

2.10.4 Analog Threshold Trigger

- Captures up to 64 readings
- Reading rate: 10 rps or higher

2.11 Source Functions (SMX2044)

- Isolated to 300 V DC from the Chassis
- Current can be paralleled with multiple SMX2044s
- Voltage can be put in series with multiple SMX2044s

2.11.1 DC Voltage Source

Parameter	Closed Loop [1]	Open Loop
Output Voltage range	-10.000 V to +10.000 V	
Typical Current source/sink at 5V output	5 mA	5 mA
DAC resolution	18 bits	12 bits
Accuracy 23°C ± 10°C One Year	0.015% ± 350 μV	1.0% ± 35 mV
Typical settling time	3 S (rate set to 2/s)	1 mS
Typical source resistance	250 Ω	

[1] 10 rps or lower measurement rate is required for the closed loop mode.

2.11.2 AC Voltage Source

Parameter	Closed Loop [1]	Open Loop
Output Voltage, sine wave	50mV to 7.1 V RMS (0.14 to 20.0V peak-to-peak)	
DAC resolution	16 bits	12 bits
Typical Current Drive at 3.5V RMS	3.5 mA RMS	
Accuracy 18°C to 28°C One Year	ACV spec ± 2 mV	ACV spec + 0.8% ± 20 mV
Typical settling time (f-out > 40 Hz)	10 s (rate set to 2 rps)	1.5 s
Typical source resistance	250 Ω	
Frequency range / resolution	2 Hz to 75 kHz / 2 Hz	
Frequency stability	100 ppm ± 1 Hz	

[1] 5 rps or lower measurement rate is required for the closed loop mode.

2.11.3 DC Current Source

Range	Compliance Voltage	Resolution [1]	Minimum level	Accuracy 23°C ± 10°C One Year
1.25 μA	4.2 V	500 pA	1 ηA	1% + 10 ηA
12.5 μA	4.2 V	5 ηA	10 ηA	1% + 100 ηA
125 μA	4.2 V	50 ηA	100 ηA	1% + 500 ηA
1.25 mA	4.2 V	500 ηA	1 μA	1% + 5 μA
12.5 mA	1.5 V	5 μA	10 μA	1% + 50 μA

[1] Resolution without Trim DAC. The use of the Trim DAC can improve the resolution by a factor of 10, but it has to be set separately since it is not calibrated.

2.12 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, the System Calibration function must be performed once a day. System Calibration is a simple software operation that takes a few seconds. Do it by executing the **DMMCalibrate()** command, or selecting S-Cal in the control panel.

All three products are capable of continuous measurement as well as data transfer rates of up to 1,000 readings per second (rps). To achieve the 6-1/2 digit resolution, the DMM should be operated at 5 rps or slower. The maximum reading rate for 5-1/2 digits is 30 rps.

Accuracy vs. Reading Rates All of the above specifications apply to reading rates of 2 rps or lower. For higher reading rates, increase the noise floor for DCV, Resistance, and DCI by the square root of the increase in reading rate from 2 rps. For example, the noise floor for the 3.3 VDC range is 8 μV at 5 rps. At 20 readings per second, or 10x the reading rate, the noise increases by the square root of 10, or 3.16 times. The noise, then, at 20 readings per second is ± 25 μV.

The noise characteristics for the AC functions increases by the same number as the DC functions. For example, the noise floor for the 3.3 VAC, 20 rps, will have digit rattle of 8.7 mV vs. 2.75 mV at 2 rps.

Reading Rates vs. Noise Rejection The best AC (50 Hz, 60 Hz or 400 Hz) power line rejection is obtained at reading rates that are whole number divisions greater than 1 of the line frequency, as shown in the following table. For best AC line rejection you should use the reading rates checked. It is important to follow this table. Always use the lowest checked rate that is practical for the application.

Reading Rate (rps)	Power Line frequency		
	50 Hz	60 Hz	400 Hz
0.1	√	√	√
0.2	√	√	√
0.5	√	√	√
1	√	√	√
2	√	√	√
5	√	√	√
10	√	√	√
15		√	
20		√	√
25	√		√
30		√	
40			√
50	√		√
60		√	
80			√
100			√
200			√
400			√

Reading Rates vs. Digits of Resolution For reading rates of 10 readings per second (rps) and slower, the DMM has 6 ½ digits of resolution. For reading rates from 10 rps to 30 rps, the DMM has 5 ½ digits of resolution.

2.13 Other Specifications

Temperature Coefficient, All Functions	Less than 0.1 x accuracy specification per °C at 23C ± 5°C
Reading Rate (user selectable)	• 0.5 to 1,000 readings per second (rps) • Up to 10 rps, 6 ½ digits • Up to 30 rps, 5 ½ digits
Hardware Interface	Single 3U PXI or CompactPCI slot
Overload Protection (voltage inputs)	330 VDC, 250 VAC
Isolation	330 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 with a 3°C internal temperature rise. All calibration constants are stored in a text file.
Temperature Range	-10°C to 70°C, operating -65°C to +85°C, storage
Size	7" X 3.5" (Standard PXI/CompactPCI 3U format)
DMM Internal Temperature Measurement (SMX2042, 44)	±2°C
Power	+5 volts, 300 mA maximum

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

Accessories

Several accessories are available for the SMX2040 DMMs, which can be purchased directly from Signametrics, or one of its approved distributors or representatives. These are some of the accessrories avaialble:

- DMM probes SM-PRB ($15.70)
- DMM probe kit SM-PRK ($38.50)
- Delux probe kit SM-PRD ($95.00).
- Shielded SMT Tweezer Probes SM-PRSMT ($24.90).
- Multi Stacking Double Banana shielded cable 36" SM-CBL36 ($39.00).
- Multi Stacking Double Banana shielded cable 48" SM-CBL48 ($43.00).
- Mini DIN Trigger, 6-Wire Ohms connector SM2040-CON7 ($14.00).
- LabView VI's library SM204x.llb ($99.00).

3.0 Getting Started

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

The DMM is shipped with the Digital Multimeter module; three floppy disks containing the various software panels and drivers plus the calibration data specific for the unit, and this Operator's manual.

3.1 Setting the DMM

The SMX2040 series DMMs are PXI/CompactPCI Plug&Play devices and do not require any switch settings, or any other adjustments to the DMM prior to installation. The only switches are the PXI trigger input and output selection switches, capable of selecting one of the PXI triggers and PXI Start Trigger.

The **SM40CAL.DAT** file supplied with your DMM has a unique calibration record for that DMM (See "**Calibration**" at the end of this manual.) When using multiple DMMs in the same chassis, the **SM40CAL.DAT** file must have a calibration record for each DMM. Append the unique calibration records of each DMM into one **SM40CAL.DAT** file using a text editor such as Notepad. The defalut location for the **SM40CAL.DAT** file is at the root directory C:\.

3.2 Installing the DMM Module

Warning

To avoid shock hazard, install the DMM only into a chassis that has its power line connector connected to an AC receptacle with an Earth Safety ground.

After installation, check to see that no loose wires or ribbon cables infringe upon any of the internal circuits of the DMM, as this may apply measurement voltages to your chassis, causing personal injury and/or damage to your equipment!

This module is designed for 3U PXI and CompactPCI chassis. To prevent shock hazard do not plug it into other format chassis such as 6U without making shure that all sides of the DMM are covered.

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

Use extreme care when plugging the DMM module(s) into a PXI or CompactPCI chassis. If possible, choose an empty slot away from any high-speed boards (e.g. CPU or other noisy modules) or the power supply. **Please be patient during the installation process!** Due to it's shielding it is a tight fit. Watch for any interference between the module and the chassis. Gently push the DMM into the chassis, making shure the handle is correctly located. Once in, lock it in with the handle and tighten the top and bottom screws to secure it into the chassis. **Be patient!**

3.3 Installing the DMM software package

To install the DMM, turn off the PXI/CompactPCI chassis, plug in the DMM into the PXI/CompcatPCI chassis, preferably away from the CPU or any other noisy card, than turn on the power. The first time you power up your computer with the DMM installed, your Windows system will detect the new DMM and will open the "New Hardware found" wizard. It will prompt you for a driver. Insert Disk1 which contains the necessary driver.

To complete the installation, run the **'SETUP'** program provided on the Diks1. This takes care of all installation and registration requirements of the software. If you are installing the DMM on a computer that had an SMX2040 series install in it, you should first uninstall the old software. For a clean reintallation remove all INF files containing reference to the Signametrics DMM. Dependig on operating system, these files will be located at Windows\inf, Windows\inf\other or WINNT\inf. The files will be named Oemx.INF where x is 0,1,2,… and/or SIGNAMETRICSSMX2040.INF. If present, these files will prevent "Found New Hardware" wizard from detecting the new DMM. Also, make sure you backup and remove the old calibration record (C:\SM40CAL.DAT).

3.4 DMM Input Connectors

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

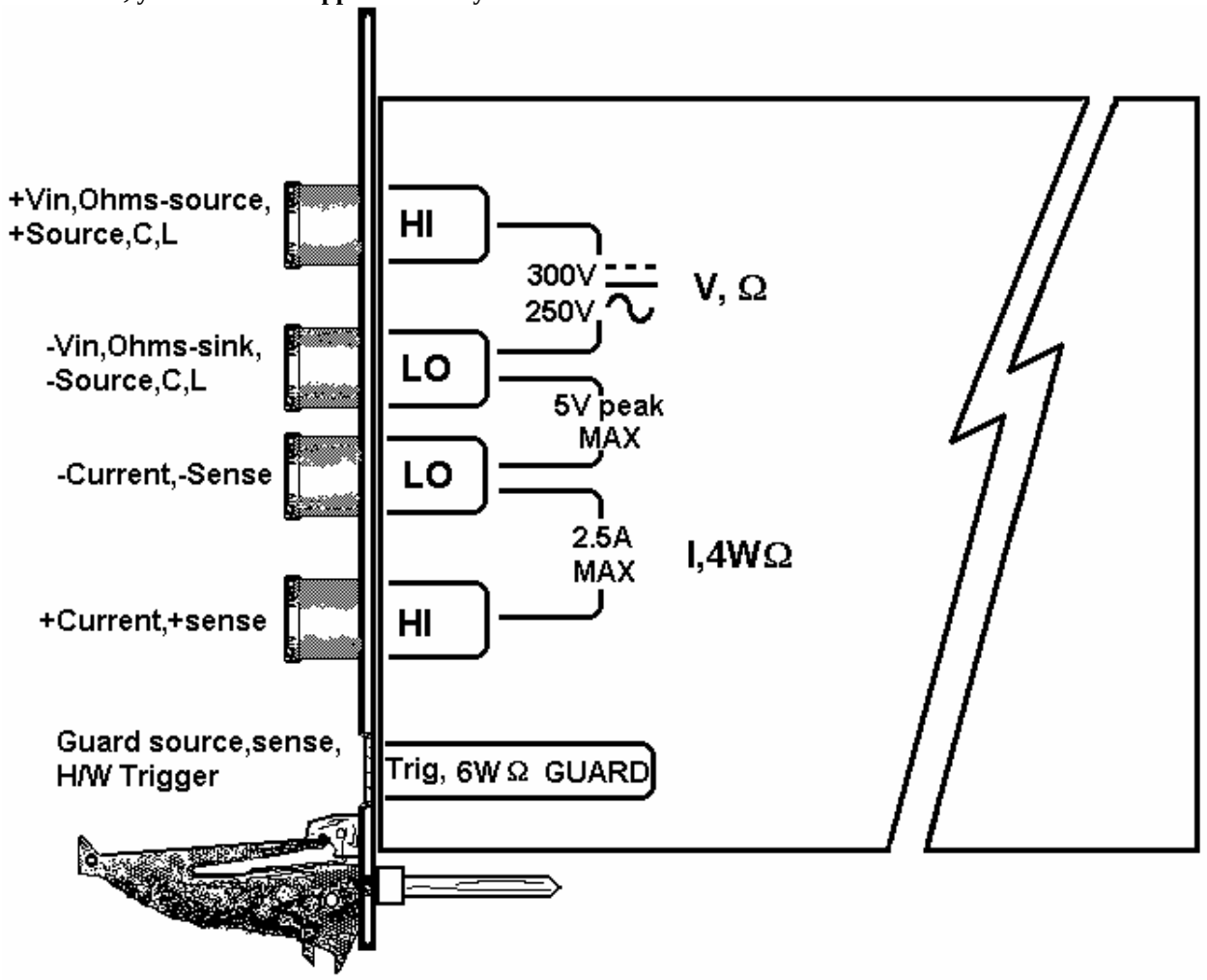


Figure 3-1. The DMM input connectors.

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

V, Ω - This is the negative terminal for all Volts, 2WΩ, capacitance diode and inductance measurements, and or sourcing of VDC, VAC and IDC. It is also the Source LO for 4WΩ. **Do not float this terminal or any other DMM terminal more than 300 VDC or 250 VAC above Earth Ground.** (Also, see **Trig, 6W Guard** below.)

I + This is the positive terminal for all Current measurements. It is also the Sense HI for 4WΩ measurements and 6WΩ guarded measurements. 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!

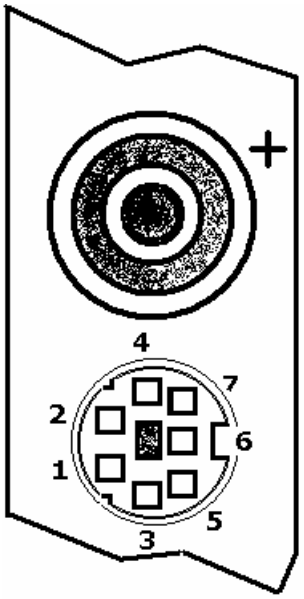
I – This is the negative terminal for all Current measurements. In the Current modes, it is protected with a **2.5 A, 250 V Fast Blow fuse** (5 x 20 mm). It is also the Sense LO for 4WΩ measurements and 6WΩ guarded measurements. **V, Ω -** and **I, 4WΩ -** should never have more than 5 V peak across them.

TRIG GUARD Both the Trigger and Guard functions use the DIN-7 connector. This group of pins include the positive and negative hardware trigger input lines and the two SMX2044 Guarded Measurement Force and Sense signals. The external trigger initiates reading(s) into the onboard buffer, and the 6W guard signals facilitate in-

circuit resistor measurements by means of isolating a loading node. The DIN-7 plug can be ordered from Signametrics and is available at many electronic hardware distributors. The connector is generically referred to as a mini DIN-7 male. The trigger signal should be in the range of 3 V to 12 V peak. The two 6W guard signals should never have more than 5 V peak across them.

Warning! The DIN connector pins are protected to a maximum of 35 V with respect to the chassis and any other DMM terminal. Do not apply any voltages greater than 35 V to the DIN connector pins. Violating this limit may result in personal injury and/or permanent damage to the DMM.

DIN-7, Pin number	Function
7	External Trigger, Positive terminal
4	External Trigger, Negative terminal
1	Guard Source (SMX2044)
6	Guard Sense (SMX2044)



DIN-7 Connector Pin Description, view from bracket side.

3.5 Starting the Control Panel

You can verify the installation and gain familiarity with the DMM by exercising its measurement functions using the Windows based Control Panel. To run the control panel, double click the "SMX2044.EXE". 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 **SM40CAL.DAT** file does not correspond to the installed DMM.

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 SMX2040 front panel powers up in DCV, 2 readings per second, 330 V range. If the DMM is operated in Autorange, with an open input, you may hear the SMX2040 relays clicking every few seconds, as a range change occurs. This is perfectly normal with ultra high impedance DMMs such as the SMX2040. This phenomenon is caused by the virtually infinite input impedance of the 330 mV and 3.3 V DCV ranges. 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 SMX2040 will change ranges, causing the relay clicking. This is normal.

3.6 Using the Control Panel

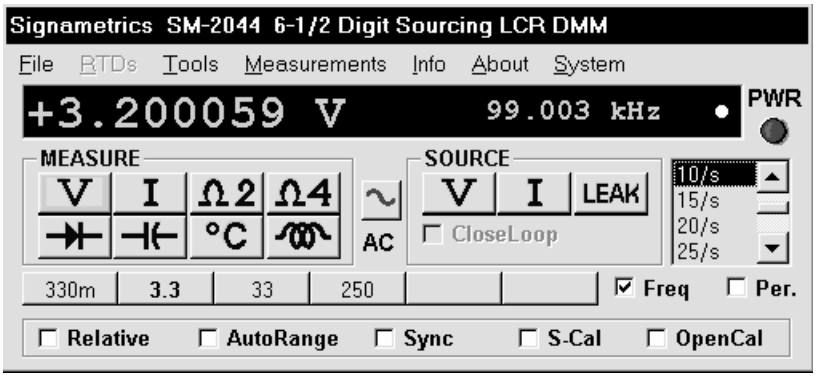
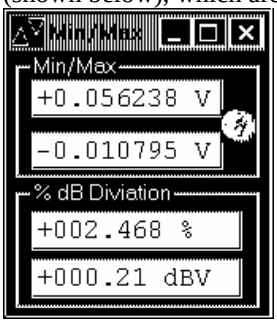


Figure 3-2**.** The Control Panel for the SMX2044. The three main groups include Measure, Source and Range buttons. The 8 Range buttons are context sensitive such that only "330m, 3.3, 33 and 250 appear when in AC Voltage Functions, "3.3m 33m 330m 2.5" appear when in Current Functions, etc.

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. Using the software command language of the SMX2040 provides powerful capabilities. Some composite functions are not included in the control panel above.

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

Relative This is the Relative function. When activated, the last reading is stored and subtracted from all subsequent readings. This is a very important function when making low level DCV measurements, or in 2WΩ. For example, when using 2WΩ, you can null out lead resistance by shorting the leads together and clicking on **Relative.** When making low level DC voltage measurements (e.g., in the μV region), first apply a copper short to the **V,Ω +** & **-** input terminals, allow the reading to stabilize for a few seconds, and click on **Relative**. This will correct for any offsets internal to the SMX2040. The **Relative** button can also be used in the Percent and dB deviation displays (shown below), which are activated using the **Tools** in the top menu.



The Min/Max box can be used to analyze variations in terms of Min, Max, Percent and dBV. This display can be activated by selecting the Min/Max/Deviation from the Tools menue. For instance, testing a circuit bandwidth with an input of 1V RMS, activate the Relative function with the frequency set to 100Hz, than sweep gradually the frequency, and monitor the percent deviation as well as the dBV error and capture any response anomalies with the Min/Max display. The left display indicates peaking of 2.468% (0.21 dBV) and maximum peaking in the response of +56.24mV and a notch of –10.79mV from the reference at 100Hz.

Rate Box Controls the SMX2040 reading rate. 0.1 rps to 1,000 rps can be set. As measurement rate increases, so does the measurement noise. For best accuracy set to the lowest rate acceptable for the application. Also consider the line frequency (50/60 Hz) of operation when setting reading rates, as certain reading rates have more noise rejection at either 50 or 60 Hz. (See "Specifications" for details.) Generally, set the measurement rate to as low a rate as practical for the application. When measuring RMS values, there is no point setting the measurement rate to a value higher than 5 rps since the RMS circuitry has a settling time that is over a second. The capacitance and inductance functions are not affected by rate setting.

Note on Measurement Rate: All three products are capable of continuous measurement as well as data transfer rates of up to 1,000 rps. To achieve the 6-1/2 digit resolution and accuracy, the DMM should be operated at 10 rps or slower. The maximum reading rate for 5-1/2 digits is 30 rps.

Range Can be set to **AutoRange** or manual by clicking on the appropriate range in the lower part of the Windows panel. Autoranging is best used for bench top application and is **not recommended** for an automated test application due to the uncertainty of the DMM range, as well as the extra time for range changes. Locking a range is highly recommended when operating in an automated test system, especially to speed up measurements. Another reason to lock a range is to control the input impedance in DCV. The 330 mV and 3.3 V ranges have virtually infinite input impedance, while the 33 V and 330 V ranges have 10 MΩ input impedance.

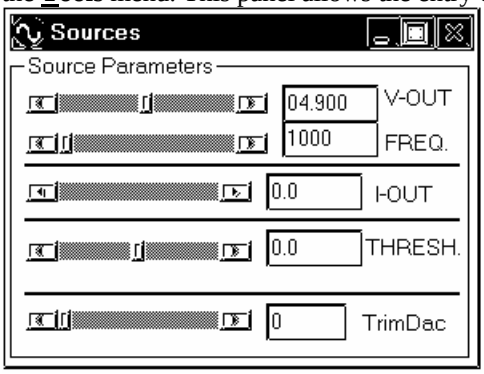
S_Cal This function is the System Calibration that corrects for internal gain, scale factor and zero errors. The DMM does this by alternatively selecting its local DC reference and a zero input. It is required at least once every day to meet the SMX2040 accuracy specifications. We recommend that you also perform this function whenever the external environment changes (e.g. the temperature in your work environment changes by more than 5°C, or the SMX2044 on board temperature sensor indicates more than a 5°C change). This function takes less than 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.

ClosedLoop This check box selection is used in conjunction with the AC and DC Voltage-Source functions of the SMX2044. When checked, the DMM monitors the output level and continuously applies corrections to the output level. When not checked, the DMM is a 12 bit source vs. 16 bits in the ClosedLoop mode.

OpenCal This check box selection is used in conjunction with inductance measurement. It is necessary to perform Open Terminal Calibration using this control, prior to measuring inductance. This function characterizes both the internal DMM circuitry as well as the probe cables. To perform OpenCal, attach the probe cables to the DMM, leaving the other end of the probe cables open circuited. Then, activate the OpenCal button.

Sync With this check box selection is active, the DMM measurements are internally synchronized, which reduces the measurement rate, but allows full scale input swings to be settled in single measurement.

Sources Panel There are three function buttons in the Source group (SMX2044 only). The **V, I, LEAK** buttons select one of three source functions, Voltage (DC and AC), IDC and Leakage. The **Sources Panel** is automatically enabled when one of the source functions is enabled. It can also be invoked using the **Sources Panel** selection under the **Tools** menu. This panel allows the entry of values for all of the source functions, including Leakage.



The **V-OUT** Scroll bar and Text box are used to set the Voltage for DC and AC Volts as well as for Leakage. When sourcing ACV, the voltage is in RMS and the **FREQ.** Scroll bar and Text box control the frequency of the source. It is also used to control inductance frequency. When sourcing DC current, use the **I-OUT** set of controls. When measuring timing or freqeuncy the **THRESH** set of controls is used for comperator threshold. All of the source controls are context sensitive and will be enabled when appropriate.

4.0 DMM Operations and Measurements

Most of the SMX2040 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 SMX2040 series DMMs, 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 300 VDC or 250 VAC. Use the **V, Ω +** and **V, Ω -** terminals, being certain to always leave the **I+, I-** and DIN-7 terminals disconnected. Use the AC/DC button on the Control Panel to switch between AC and DC.

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

4.1.1 DC Voltage Measurements

When making very low level DCV measurements (<100 μV), you should first short the DMM with a copper wire shorting plug across the **V, Ω +** and **V, Ω -** terminals and perform the **Relative** function to eliminate zero errors before making your measurements. A common source of error can come from your test leads, which can introduce several μVolts of error due to thermal voltages. To minimize thermal voltaic effects after handling the test leads, you should wait a few seconds before making measurements. Signametrics offers several high quality probes that are optimal for low level measurements.

Note: The SMX2040 front panel powers up in DCV, 2 readings per second, 330 *V range. If the DMM is operated in Autorange, with an open input, you may hear the SMX2040 relays clicking every few seconds, as a range change occurs. This is perfectly normal with ultra high impedance DMMs such as the SMX2040. This phenomenon is caused by the virtually infinite input impedance of the 330 mV and 3.3 V DCV ranges. 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 SMX2040 will change ranges, causing the relays to click. This is normal.*

4.1.2 True RMS AC Voltage Measurements

ACV is specified for signals greater than 1mV, from 10 Hz to 100 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 SMX2040 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 SMX2040 **V,Ω -** terminal. See Figure 4-1, below. This prevents any "Common Mode" problems from occurring (Common Mode refers to floating the SMX2040 **V,Ω LO** above Earth Ground.) Common Mode problems can result in noisy readings, or even cause the computer 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.

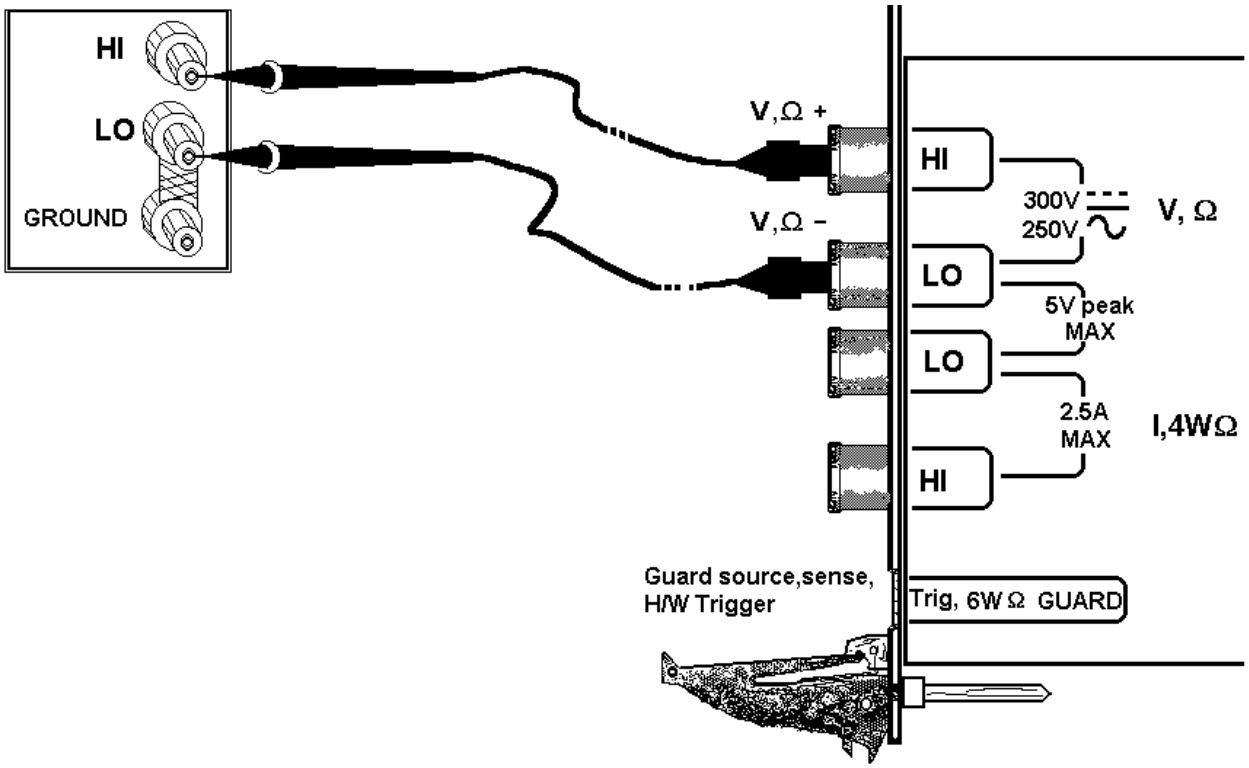


Figure 4-1**.** Make Voltage measurements with the source ground attached to the SMX2040 **V,Ω -** to minimize "Common Mode" measurement problems.

4.1.3 AC Peak-to-Peak and Crest Factor Measurement (SMX2044)

Measurement of Peak-to-Peak, Crest Factor and AC Median values requires a repetitive waveform between 30 Hz and 100 kHz. The DMM must be in AC voltage measurement mode, with the appropriate range selected. Knowing the Peak-to-Peak value of the waveform is useful for setting the Threshold DAC (described below). This latter function is a composite function, and may take over 10 seconds to perform.

4.1.4 AC Median Value Measurement (SMX2044)

To better understand the usage of this function, you should note that the DMM makes all AC voltage measurements through an internal DC blocking capacitor. The voltage is thus "AC coupled" to the DMM. The measurement of the Median value of the AC voltage is a DC measurement performed on the AC coupled input signal. This measurement returns the mid-point between the positive and negative peak of the waveform. The Median value is used for setting the comparator threshold level for best counter sensitivity and noise immunity. (It is difficult to measure the frequency of a low duty cycle, low amplitude AC signal since there is DC shift at the comparator input due to the internal AC coupling. The SMX2044 overcome this problem by allowing you to set the comparator threshold level). For further information on the usage of AC Median value and Peak-to-Peak measurements, and the Threshold DAC, see the "Frequency and Timing Measurements" section below.

This function requires a repetitive signal. The DMM must be in AC voltage measurement mode, with the appropriate range selected.

4.2 Current Measurements

The SMX2040, 42, 44 measure from 10 ηA to 2.5 A. Use the **I, 4WΩ** terminals, being certain to always leave the **V,Ω +** & **-** terminals disconnected. Use the AC/DC button to switch between AC and DC.

The Current functions are protected with a 2.5 A, 250 V fuse.

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

When making sensitive DC current measurements, be sure to use the **Relative** function to zero out any residual errors of the SMX2040. This is easily accomplished by opening all inputs to the SMX2040 and performing **Relative** in the appropriate DCI range.

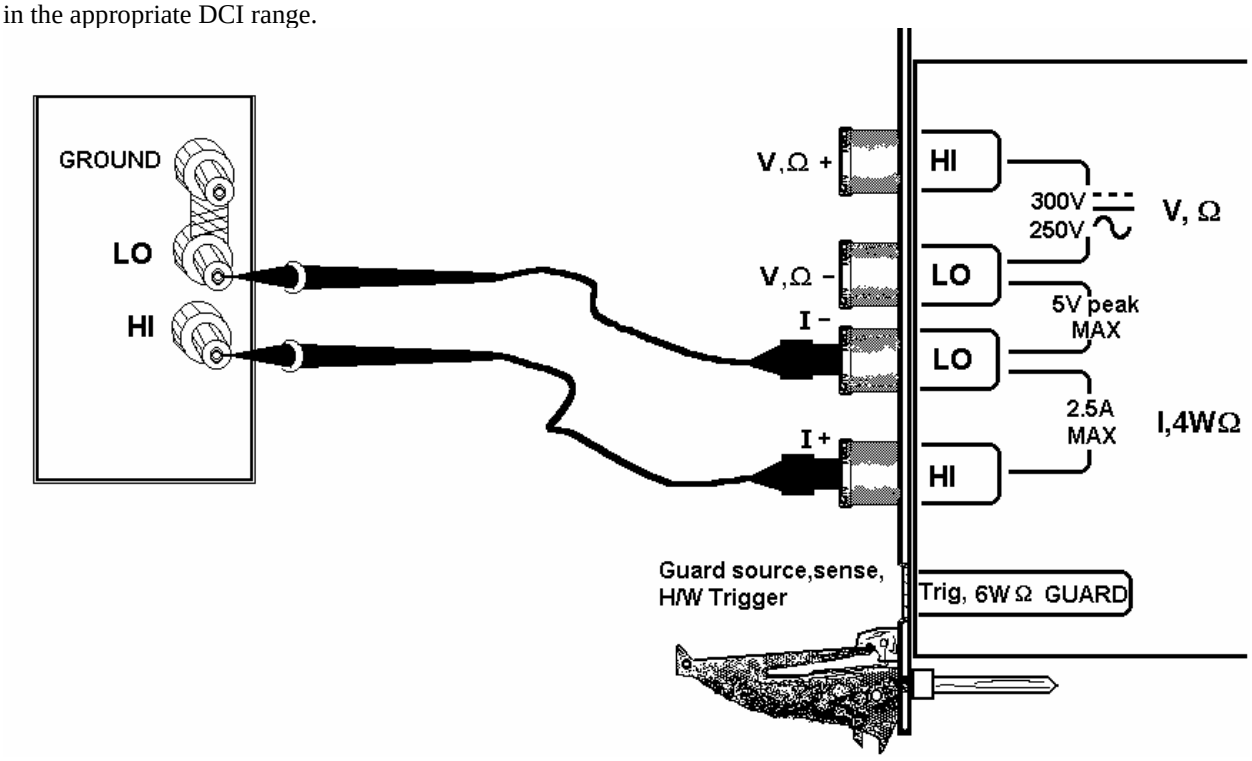


Figure 4-2. AC and DC Current measurement connection.

4.2.1 Improving Current Measurements

When making sensitive DC current measurements, be sure to use the **Relative** function to zero out any residual errors of the SMX2040. This is easily accomplished by disconnecting all terminals to the DMM and performing **Relative** in the appropriate DCI range. Using the **S-Cal** (**DMMCalibrate()**) prior to activating **Relative** will improve accuracy further. Although the SMX2040 family is designed to withstand up-to 2.5A 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. The lower two ranges of DC current may be effected by relay contamination. If the measurements seem unstable or high, while in IDC measurement, apply between 20mA and 50mA DC to the current terminals and clean the K2 relay using the **DMMCleanRelay(0, 2, 200)**. Repeat this until the measurements are stable.

4.2.2 Low Level DC Current Measurements

For low level current measurements use the **V, Ω+** and **V, Ω-** terminals. Using the 33V DCV range, the SMX2040 can measure very low currents. This hidden measurement function is facilitated by the DMM's low leakage front-end and a virtual 10.0MΩ input resistance. With a typical offset error of less than 100μV in this VDC range, it is practical to measure down to 20pA. The maximum current value that can be measured has more to do with the user's acceptable burden voltage (the voltage drop across the 10.0MΩ shunt) then the DMM limitations. Assuming a maximum burden voltage of 3.3V the maximum current level is 330ηA. This range is well within leakage measurements required in most semiconductor testing. It is also a very quite and stable. Since the DMM does not have an explicit low current function, it is necessary to calculate the current, which is equal to the measured voltage divided by 10.0MΩ.

4.2.3 Extended DC Current Measurements (SM2044)

The leakage measurement function can also be used to measure low-level currents. Using a relatively high value shunt resistor and setting the leakage test voltage to 0V allows measurements of super low level currents, without any burden voltage. Read the leakage current section of the manual for more details.

4.3 Resistance Measurements

Resistance is measured with one of eight (six in the SMX2040) precision current sources, with the DMM displaying the resistance value. Most measurements can be made in the 2-wire mode. 4-wire ohms is used to make precision low resistance measurements. All resistance measurement modes are susceptible to Thermo-Voltaic (Thermal EMF) errors. See section section 4.3.5 for details.

4.3.1 2-wire Ohm Measurements

The DMM measure using 330Ω to 33 MΩ ranges. The SMX2042 and SMX2044 add 33 Ω and 330 MΩ ranges. Use the **V,Ω+, V,Ω-** terminals, being certain to always disconnect the **I+, I-** terminals.

Most resistance measurements can be made using the simple 2-wire Ohms method. Simply connect **V,Ω+** to one end of the resistor, and the **V,Ω-** to the other end. If the resistor to be measured is less than 30 kΩ, you should null out any lead resistance errors by first touching the **V,Ω+** and **V,Ω-** test leads together and then performing a **Relative** function. If making measurements above 300 kΩ, you should use shielded or twisted leads to minimize noise pickup. This is especially true for measurements above 1 MΩ.

You may also want to control the Ohms current used in making resistance measurements. (See the Specifications section, "Resistance, 2-wire and 4-wire", for a table of resistance range vs. current level.) All of the Ohms ranges of the SMX2040 have enough current and voltage compliance to turn on diode junctions. For characterizing semiconductor part types, use the Diode measurement function. To avoid turning on a semiconductor junction, you may need to select a higher range (lower current). When checking semiconductor junctions, the DMM displays a resistance value linearly related to the voltage across the junction.

For applications requiring resistance measurements higher than 330 MΩ, the Extended Resistance Measurement method is available with the SMX2044.

4.3.2 4-wire Ohm Measurements

4-wire Ohms measurements are advantageous for making measurements below 330 kΩ, eliminating lead resistance errors. The **Voltage** (**V,Ω**) Input terminals serve as the current "Source" (i.e. they provide the current stimulus in the ohms measurement), and the **I, 4WΩ** Input terminals are the "Sense" inputs. The Source + and Sense + leads are connected to one side of the resistor, and the Source - and Sense - leads are connected to the other side. Both Sense leads should be closest to the body of the resistor. See Figure 4-3.

4-wire Ohm makes very repeatable low ohms measurements, from 100 μΩ (10 μΩ for SMX2042, 44) to 330 kΩ. It is not not recommended to us **4WΩ** when making measurements above 100 kΩ, although 4-wire ohms is allowed

up to 330 kΩ. 4-wire measurements are disabled above 330 kΩ since the extra set of leads can actually *degrade* the accuracy, due to additional leakage and noise paths.

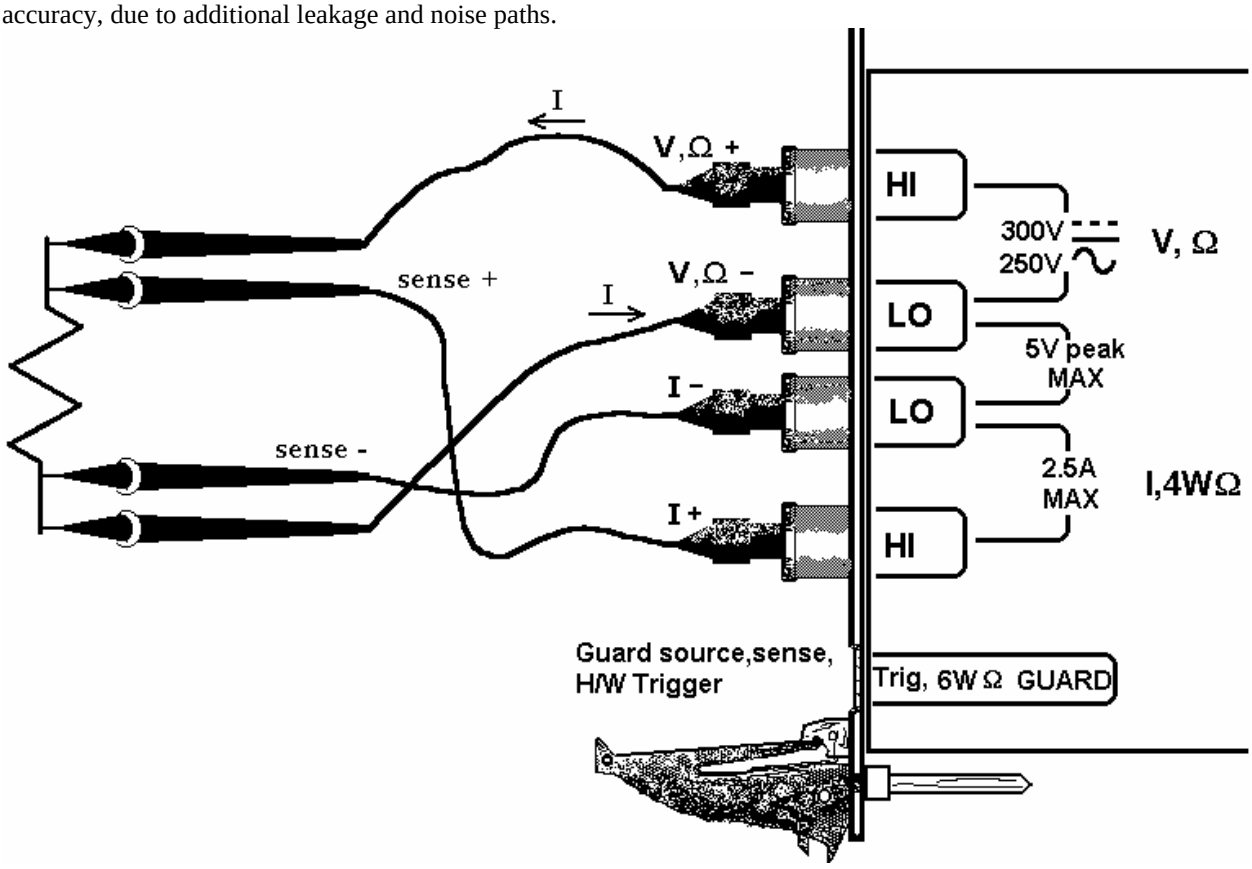


Figure 4-3. The **I- and I+** sense leads should be closest to the body of the resistor when making 4WΩ measurements.

4.3.3 Using Offset Ohms function

Inadvertent parasitic leakage currents, Thermo voltaic voltages and other voltages in series can effect resistance measurements with the measured resistance. This is common particularly when doing in-circuit measurements, and it could manifest as having a significantly different readings when changing Ohms ranges. In addition to eliminating such error, this function can also be used to measure internal resistance of low value voltage sources such as various batteries and supplies. Use the normal 2-Wire or 4-Awire Ohms connection, and set the Offset Ohms to the enabled or disabled state using the **DMMSetOffsetOhms()** function. When set TRUE, the measurement rate will be about $1/10^{th}$ of the set DMM rate. Both negative and positive polarity voltages can be handled as long as the total voltage including the Ohms source current times the measured resistance plus the parasitic voltage are less than 3.7V. To calculate this voltage consult the specification part of this manual for the specific current for each Ohms range. The default value of this function is FALSE. This function is implemented for the SMX2040, SMX2042 and SMX2044. Due to it's different hardware, the SMX2042 is different in that one of the DMM relays is being used to perform this operation, which can be heard.

4.3.4 6-wire Guarded Resistance Measurement (SMX2044)

The SMX2044 provides a guarded 6-wire resistance measurement method. It is used to make resistance measurements when the resistor-under-test has other shunting paths that can cause inaccurate readings. This method isolates the resistor-under-test by maintaining a guard voltage at a user-defined node. The guard voltage prevents the shunting of the DMM Ohms source current from the resistor-under-test to other components. The Guard Source and Guard Sense terminals are provided at pins 1 and 6 of the DIN connector respectively.

Warning! The DIN connector pins are only protected to a maximum of 35 V with respect to the chassis or any other DMM terminal. Do not apply any voltages greater than 35 V to the DIN connector pins. Violating this limit may result in personal injury and/or permanent damage to the DMM.

Example: Assume a 30 kΩ resistor is in parallel with two resistors, a 510 Ω and a 220 Ω, connected in series. In a normal resistance measurement, the 510 Ω and 220 Ω would shunt most of the DMM Ohms source current, causing

an inaccurate reading. By sensing the voltage at the top of the 30 kΩ, and then applying this same voltage to the junction of the 510 Ω and 220 Ω, there is no current flow through the shunting path. With this "guarding", the SMX2044 accurately measures the 30 kΩ resistor.

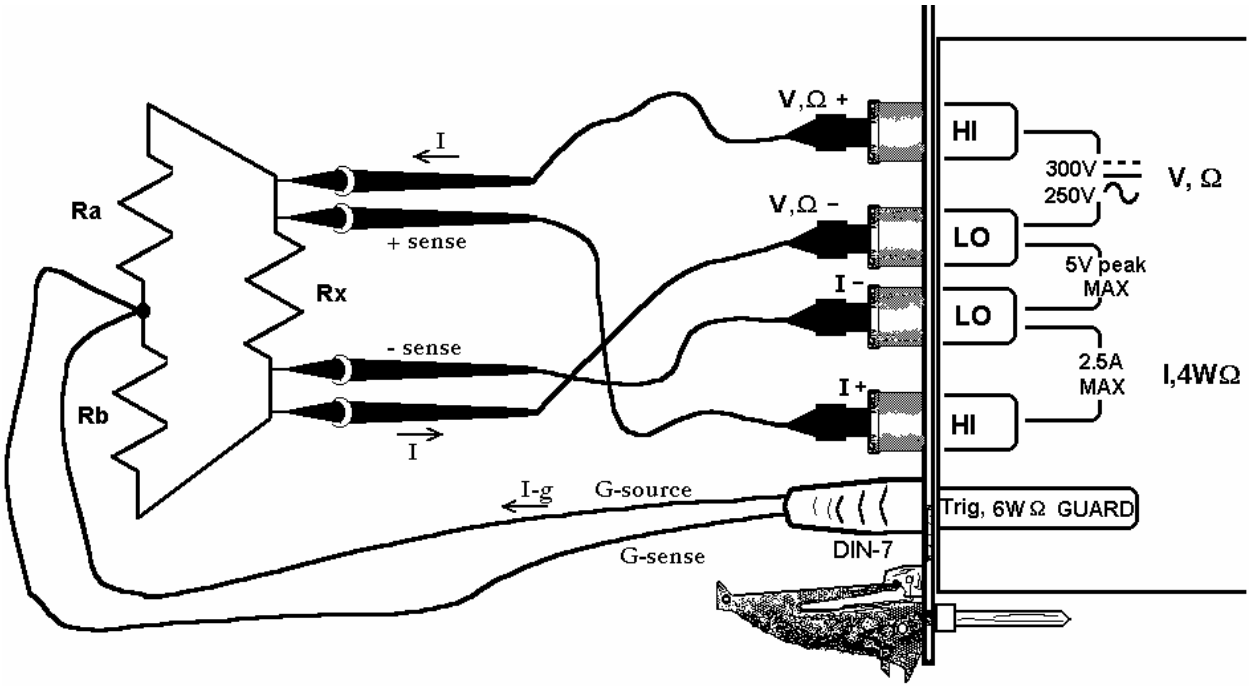


Figure 4-4. 6-wire guarded in-circuit ohms measurement configuration.

The current compliance of the Guard Force is limited to a maximum of 20 mA and is short-circuit protected. The resistor connected between the low of the 4-wire terminals and the guard point is the burden resistor, or R_b. Due to the limited guard source current, this resistor can not be lower than R_{bmin}: $R_{bmin} = I_o * R_x / 0.02$, where I_o is the ohms source current for the selected range, and R_x is the resistance being measured. For example, selecting the 330 Ω range and measuring a 300 Ω resistor imposes a limit on R_b of at least 15 Ω or greater. Since the top burden resistor, R_a, does not have this limit imposed on it, selecting the measurement polarity, R_a can become R_b and vise versa. For cases where this limit is a problem, simply set the measurement polarity such that R_a is the higher of the two burden resistors.

To measure values greater than 330 kΩ using the 6-wire guarded method, it is necessary to select the 2-wire ohms function, and maintain the 6-wire connection as in Figure 4-4 above.

4.3.5 Leakage Measurements (SMX2044)

The SMX2044 measures leakage currents by applying a DC voltage across the load and measuring the current through it. An external shunt resistor, R_S is used to sense the current. See Figures 4-5 for connection. The DC voltage at which leakage is measured is set using **DMMSetDCVSource()**, and the value of R_S is entered using **DMMSetExternalShunt()** (the default is 1 MΩ). Leakage current is read using **DMMRead(), DMMReadStr()** or **DMMReadNorm()** functions. Depending on the value of R_S and the leakage current, the test voltage may be set between –10V and +10 V. Specifically; the maximum voltage is limited to 10.2V – (R_S * I_L). The leakage current (I_L) measurement range and accuracy depends on the shunt resistor being used. The range is set as 3.3V/R_S, with resolution of 10uV/R_S. For instance, with R_S equal to 1MΩ, the maximum leakage current is 3.3uA with resolution of 10pA, and at full scale the maximum test voltage is 6.7V. The test voltage is adjusted whenever a measurement is made. Therefore, if the leakage current being measured varies significantly over time, it is best to repeatedly read the DMM allowing the DMM to make corrections to the internal source voltage, compensating for drop across the shunt. Performing open terminal calibration will imrove the accuracy of this function (use **DMMOpenTerminalCal()** with all terminals open). Refer to Figures 4-5. See section 2.6 for specifics. Setting the source to closed loop mode, **DMMSetSourceMode()**, will improve the accuracy of the test voltage.

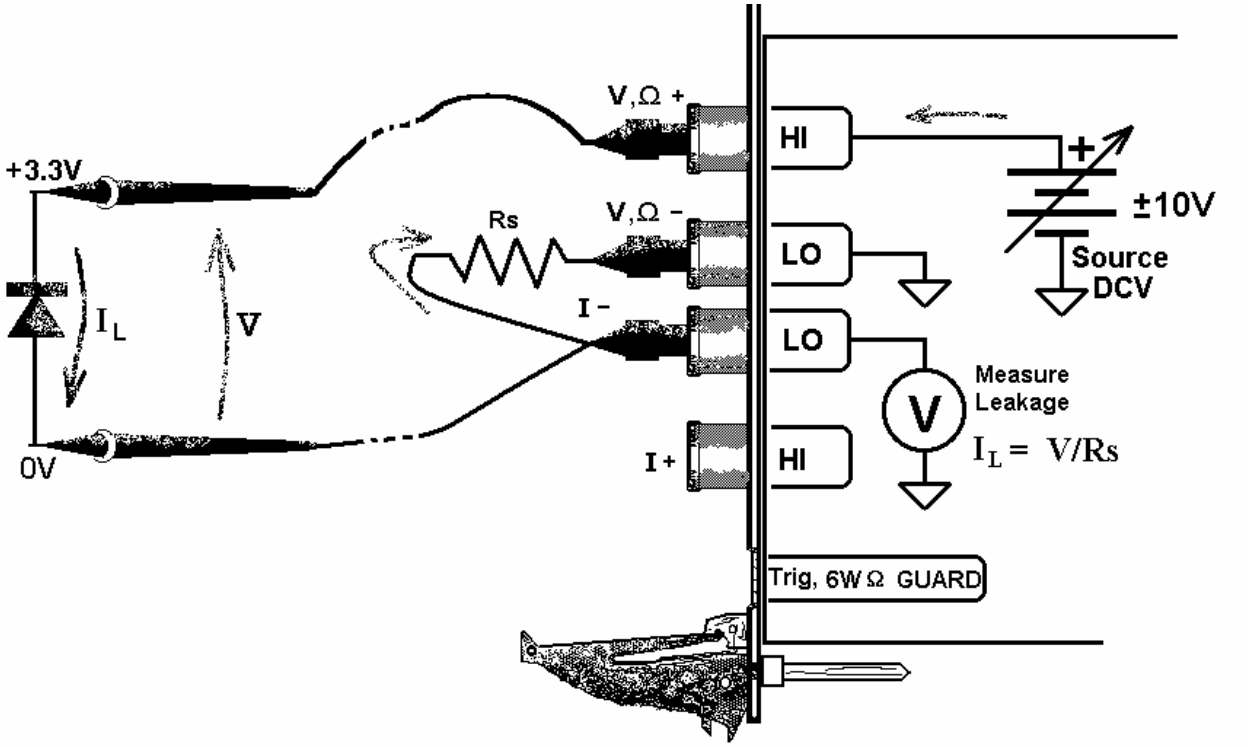


Figure 4-5. Leakage Test Configuration. Measurement of reverse diode leakage at 3.3V.

4.3.6 Extended Resistance Measurements (SMX2044)

The Extended Resistance measurement function operates as complement of the standard resistance measurement. Where the last forces a predefined current, this function forces specified voltage. Where the normal resistance measurement is limited to pre defined current sources, this function has a variable voltage, and it limits current flow by an external sense resistor. To perform this measurement use the connection diagram in Figure 4-6. Enter the value of the sense resistor R_S using **DMMSetExternalShunt()**, and set the test voltage with **DMMSetDCVSource()**. Due to the availability of a much higher test voltage than the normal resistance function, this function is fit for very high value resistance measurements. On the other hand, the ability to set to some low voltages is significant in applications where specific test voltage is required such as to prevent semiconductor's turn-on, or prevent sensitive sensors' damage due to access voltage or current. The DMM compensates for the voltage drop across the sense resistor as to provide the specified test voltage. Each time the DMM makes a measurement using **DMMRead(), DMMReadStr()** or the **DMMReadNorm,** it adjusts the test voltage. There are several limitations that should be considered while using this function. The internal voltage source of the SM2044 is limited to about +/-10.2V. Therefore the sum of the voltages across the sense resistor and the measured resistor is limited to this amount. Also the maximum voltage allowed across the sense resistor is limited to 3.3V. Exceeding either value will result in Over Range reading. The limit imposed by the shunt resistor may be expressed as $V_L/R_X * R_S < 3.3V$, where V_L is the test voltage, R_X is the resistance being measured, and R_S is the external sense resistor value. The source voltage limit may be expressed as $V_L(1+R_S/R_L) < 10.2V$. Use the **DMMSetExternalShunt()** to set the value of the external shunt being used. Use the **DMMSetDCVSource()** to set the test voltage (the voltage across the resistor being measured). For high value resistances, the shunt should be of high value and vise versa.

This function is useful for testing high value resistive elements such as cables, transformers, and other leaky objects such as printed circuit boards**,** connectors and semiconductors.

As an example, consider a 1.0MΩ shunt and a 4V test voltage. In this case, the lowest resistor one can measure will be 1.2122MΩ. The shunt voltage V_S is at its limit here since 1.0MΩ * 4V / 1.2122MΩ = 3.3V which is the limit for V_S. The source voltage limit is reached when V_L is set to 8.5V and the measured resistance falls below 5.001MΩ since 8.5V(1 + 1.0MΩ/5.001MΩ) = 10.2V.

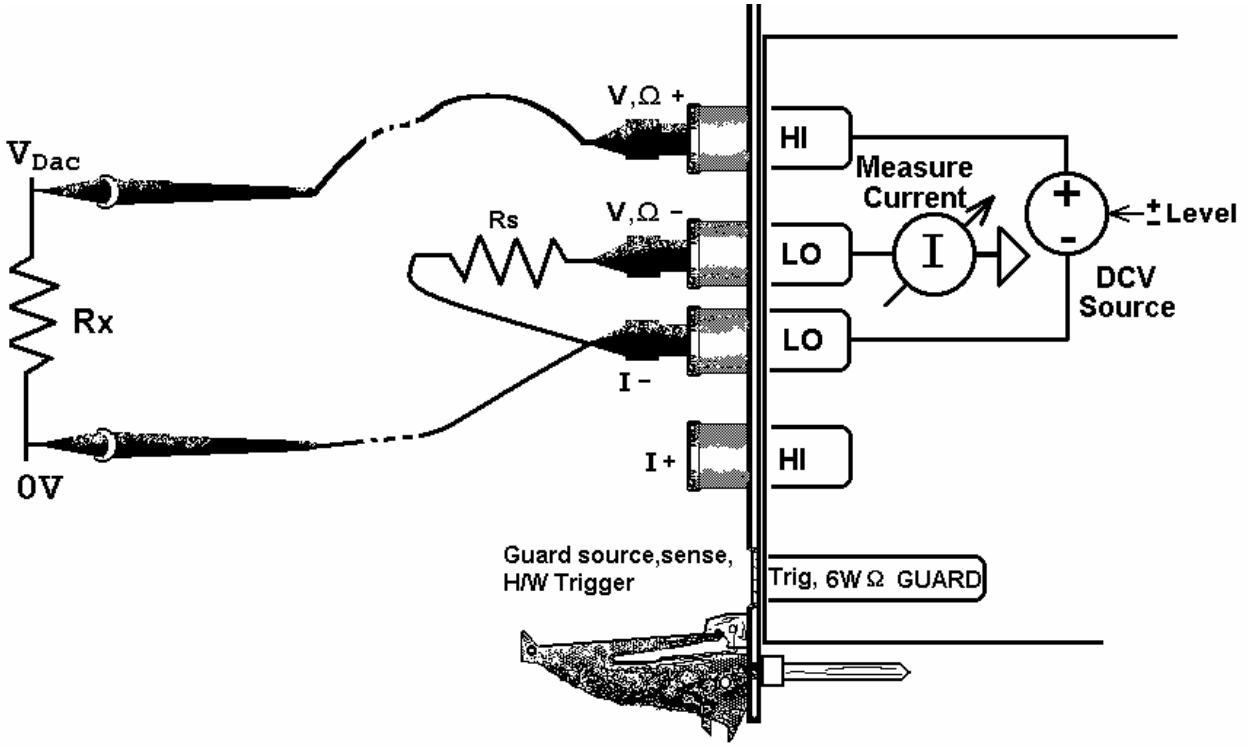


Figure 4-6. Extended Ohms range.

4.3.7 Effects of Thermo-Voltaic Offset

Resistance measurements are sensitive to Thermo-Voltaic (Thermal EMF) errors, which can be caused by poor test leads, relay contacts and other elements in the measurement path. These errors effect all measurement methods, including 2-Wire, 4-Wire, 6-Wire and 3-Wire (guarded 2-Wire ohms). 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. It is common to find relays that have more than 50 μV EMF. With several relay contacts in the path, the error is very significant. It is possible to measure it using the SMX2040 330mV 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). To calculate the worst case error, multiply this count by the Thermal EMF voltage. Use Ohms law to convert this voltage to resistance error as in the following table.

Resistance Measurement Errors due to Thermo-Voltaic offsets.

SM2042 Range	Ohms Current	DMM Resolution	Error due to 10 μV EMF	Error due to 100 μV EMF	Error due to 1 mV EMF
33 Ω	10 mA	10 μΩ	1 mΩ	10 mΩ	100 mΩ
330 Ω	1 mA	100 μΩ	10 mΩ	100 mΩ	1 Ω
3.3 kΩ	1 mA	1 mΩ	10 mΩ	100 mΩ	1 Ω
33 kΩ	100 uA	10 mΩ	100 mΩ	1 Ω	10 Ω
330 kΩ	10 uA	100 mΩ	1 Ω	10 Ω	100 Ω
3.3 MΩ	1 uA	1 Ω	10 Ω	100 Ω	10 Ω
33 MΩ	100 nA	100 Ω	100 Ω	1 kΩ	100 Ω
330 MΩ	10 nA	10 kΩ	1 kΩ	10 kΩ	100 kΩ

4.3.8 Guarding High Value Resistance Measurements (SMX2044)

Measuring high value resistors using the 2-Wire function require special attention. Due to the high impedances involved during such measurements, noise pickup and leakage could be very significant. To improve this type of measurement it is important to use good quality shielded cables with a low leakage dielectric. Even with a good dielectric, if a significant length is involved, an error would result due to leakage. Figure 4.7 exemplifies this error source. It is important to emphasize that in addition to the finite leakage associated with the distributed resistance, R_L, there must also be a voltage present between the two conductors, the shield and the center lead, for leakage current to develop. Provided there was a way to eliminate this voltage, leakage would have been eliminated.

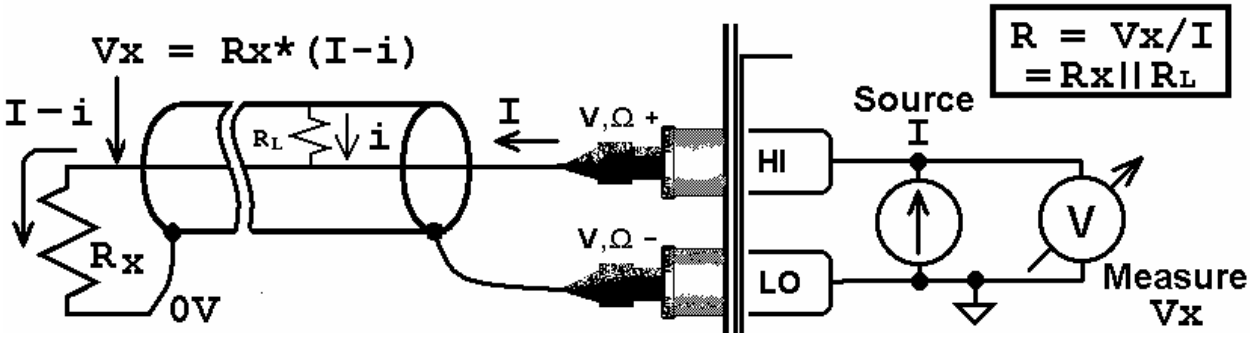


Figure 4-7. Error due to cable leakage.

The SM2044 provides an active guard signal that can be connected to the shield and prevent the leakage caused by the dielectric's finite resistance. With the shield voltage guarded with Vx, as indicated in Figure 4-8, there is 0V between the shield and the high sense wire, and therefore no current flows through R_L.

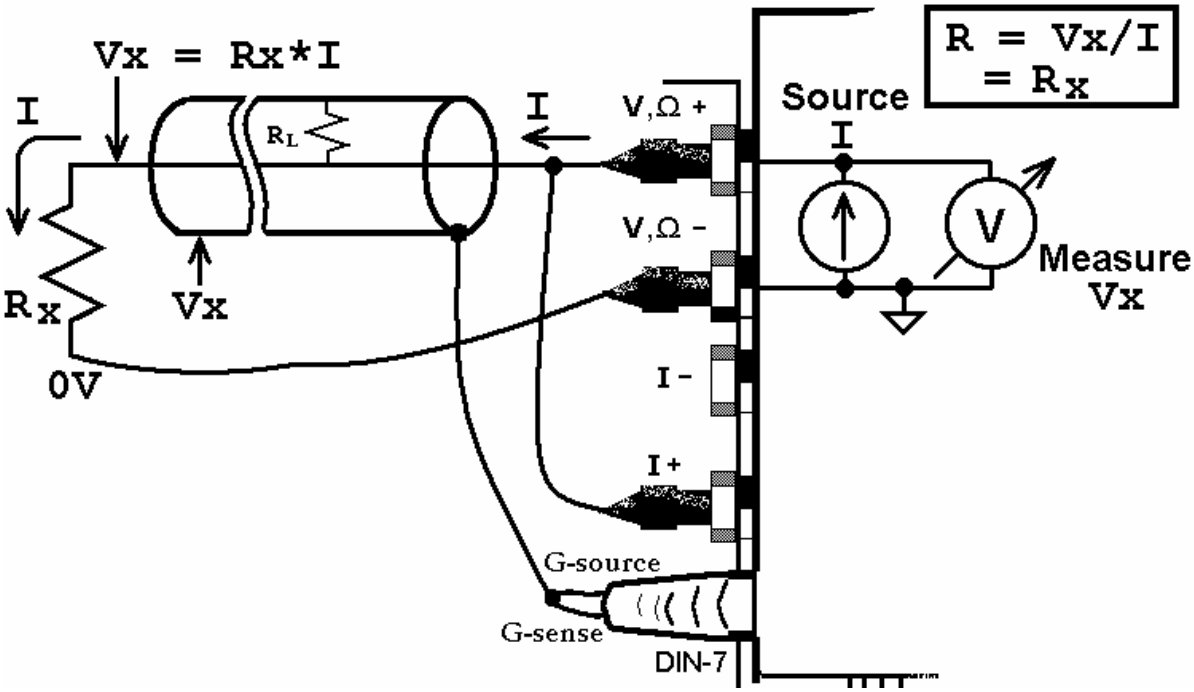


Figure 4-8. Guarding eliminates errors due to leakage associated with high resistance measurements.

4.4 RTD Temperature Measurement (SMX2044)

For temperature measurements, the SMX2044 measure and linearize RTDs. 4-wire RTD's can be used by selecting the appropriate RTD type. Any ice temperature resistance between 25 Ω and 10 kΩ can be set for the platinum type RTDs. Copper RTDs can have ice temperature resistance values of 5 Ω to 200 Ω. The highest accuracy is obtained from 4-wire devices, because the resistance of the test leads is nulled out. The connection configuration for RTDs is identical to 4-wire Ohms.

4.5 Internal Temperature (SMX2044)

A special on board temperature sensor allows monitoring of the DMM's internal temperature. This provides the means to determine when to run the self-calibration function (S-Cal) for the DMM, as well as predicting the performance of the DMM under different operating conditions. When used properly, this measurement can enhance the accuracy and stability of the DMM. It also allows monitoring of the chassis internal temperature, which is important for checking other instruments.

4.6 Diode Characterization

The Diode measurement function is used for characterizing semiconductor part types. This function is designed to display a semiconductor device's forward or reverse voltage. The DMM measures diode voltage at a selected current. The available source currents for diode I/V characterization include five DC current values; 100 ηA, 1 μA, 10 μA, 100 μA and 1 mA. The SMX2044 have an additional 10 mA range. The SMX2044 also has a variable current source which can be used concurrently with DCV measurement (see "Source Current / Measure Voltage"). This allows a variable current from 10 ηA to 12.5 mA. The maximum diode voltage compliance is approximately 4 V.

Applications include I/V characteristics of Diodes, LEDs, Low voltage Zener diodes, Band Gap devices, as well as IC testing and polarity checking. Typical current level uncertainty for diode measurements is 1%, and typical voltage uncertainty is 0.02%.

4.7 Capacitance Measurement (SMX2044)

The SMX2044 measure capacitance using a differential charge slew method, where variable currents are utilized to produce a dv/dt across the capacitor. Use short high quality shielded probe cables with no more than 500 pF. With the exception of the 10 ηF range, each of the ranges has a reading span from 5% of range to full scale. Capacitance values less than 5% of the selected range indicate zero. Since some large value electrolytic capacitors have significant inductance, as well as leakage and series resistance, the Autoranging function may not be practical. Because Capacitance measurement is sensitive to noise, you should keep the measurement leads away from noise sources such as computer monitors. For best measurement accuracy at low capacitance values, zero the DMM using the 'Relative' while in the 10 ηF range. The effect of the cable quality and its total capacitance is significant particularly on low value caps. For testing surface mount parts, use the optional Signametrics SMT Tweezer probes. See Figure 4-9 for connection.

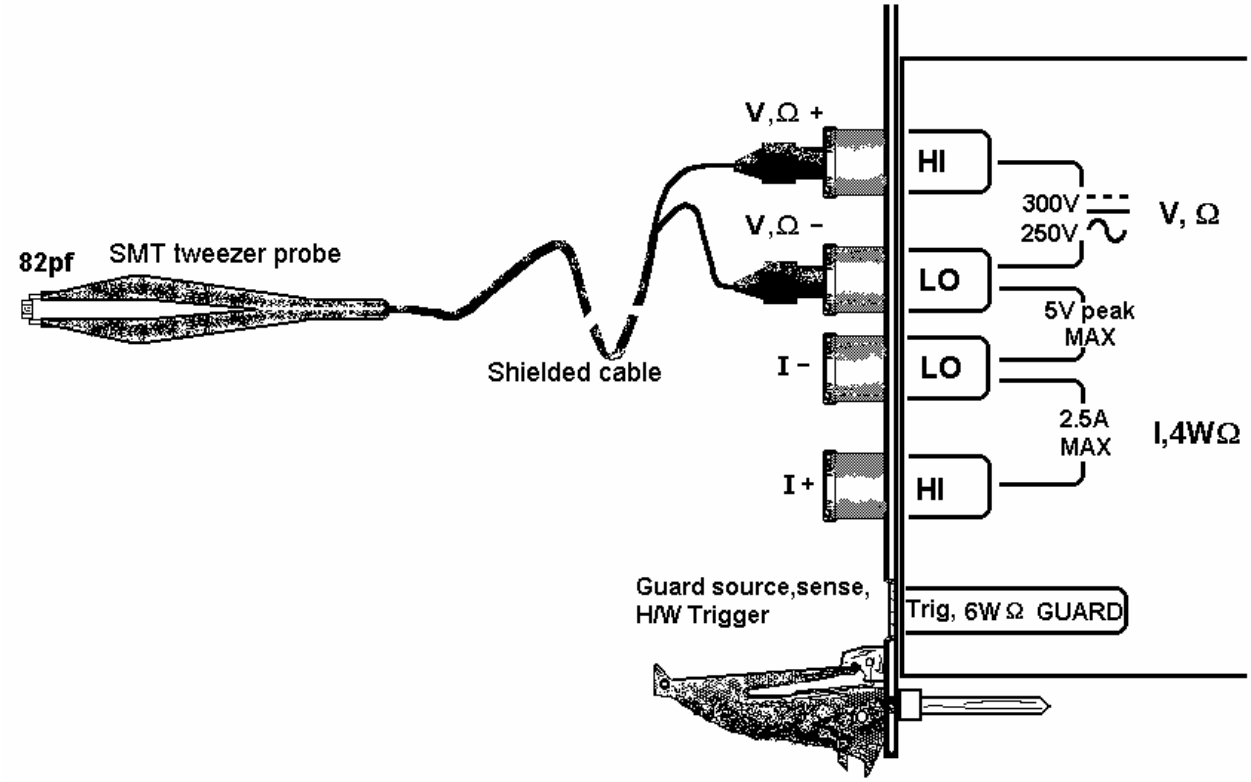


Figure 4-9. Measuring capacitors or inductors is best handled with shielded probe wires.

4.8 In-Circuit Capacitance Measurement (SMX2044)

A second method provided for measuring capacitors is the AC based method. Though not as accurate as the above function, the advantage of this method is that the default stimulus is set at 0.45V peak, which is lower than a semiconductor junction on voltage. It may also be set over a wide range of voltages. A further advantage is the ability of this function to measure capacitors that have a very low value parallel resistance, which is impossible to do using conventional methods. This test function operates by figuring the complex impedance and extracting from it both, the capacitance and resistance. The measurement is practical down to a few hundred Pico Farads, and up to several thousands micro Farads, with parallel resistances as low as 20Ω to 300Ω depending on range. Once set to this function, use **DMMRead()**, **DMMReadStr()** and **DMMReadNorm()** to measure the capacitance value. To get the resistance value use **DMMGetACCapsResist()** following a read. Each of the ranges must be calibrated with open terminals prior to making measurements. Each range must be calibrated. Do this by activating the AC-Caps function, selecting the range to be calibrated and issuing **DMMOpenCalACCaps()**. The last function normalizes the AC source signal. This open Calibration operation must be performed with the measurement cable or probes plugged into the DMM, with the other end open. See figure 4-9 above for connection. If not modified by the **DMMSetACCapsLevel()** function, when making a measurement the DMM uses a default voltage of 0.45Vpk, which means that a sine wave that has a peak-to-peak amplitude of 0.9V. This level is used during both, open calibration and measurements. Since the DMM is optimized for this value, and it is well below most semiconductors On voltage, it is recommended not to change the level from this default value. The stimulus voltage can be set from 0.1V peak to 5V peak using the **DMMSetACCapsLevel()** function. Any time the stimulus level is adjusted; open calibration must be carried out. The results of **DMMOpenCalACCaps()** are kept in memory until the DLL is unloaded. Repeating Open calibration periodically will result in improved accuracy. AC Capacitance measurement function must be used with a DMM reading rate of 10rps. Other functions of interest for this operations are **DMMSetACCapsDelay()** and **SetACCapsFreq()**.

4.9 Inductance Measurement (SMX2044)

The SMX2044 measures inductance using a precision AC source with a frequency range of 20 Hz to 75 kHz. Since inductors can vary greatly with frequency, you should choose the appropriate generator frequency. In addition to inductance, the inductor's Q factor can be measured. A shielded, high quality cable is highly recommended. For best accuracy, perform the Open Terminal Calibration function within an hour of inductance measurements. The Open Terminal Calibration function must be performed with the cables plugged into the DMM, but with the other end open circuited. This process characterizes the internal signal path inside the DMM, the open application cable, and the DMM circuitry.

For best measurement accuracy at low inductance values, zero the DMM often by using the '**Relative**' function with the leads shorted. This must be done following Open Terminal Calibration operation. This Relative action measures and removes the inductance of the DMM signal path including that of the application cable being used.

4.10 Characteristic Impedance Measurement (SMX2044)

To measure a transmission line's characteristic impedance, measure the cable's capacitance C (with the end of the cable open) and then it's inductance L (with the end of the cable shorted). The cable's impedance equals the square root of L/C. Be certain the cable is long enough such that both the capacitance and inductance are within the specified measurement range of the SMX2044.

4.11 Trigger Operation

4.11.1 External Hardware Trigger

The Trigger functions provide for a stand-alone capture of measurements. The local controller supervises the operation, and when conditions are valid, it captures data into its buffer, or sends it back to the PXI bus. The reading rate must be set to 10 rps or higher. The External Trigger's isolated high and low input lines are provided at pins 7 (+) and 4 (-), respectively, on the DIN connector located on the front panel of the instrument. You may abort the External Trigger's wait state using the Disarm command. It is important to note that the External Hardware Trigger is **Wire-Ored** with the PXI triggers. Therefore, while using the External Trigger, the PXI trigger input line selected (see section 4.11.4) must be at a low logic level (inactive). While using the PXI trigger inputs as the trigger source, the External Hardware trigger must at a low logic level, or disconnected. Failing to do this will result in a disabeled trigger. The PXI and External Hardware trigger software functions includes: **DMMArmTrigger**, which responds to

positive edge, **DMMSetBuffTrigRead,** and **DMMSetTrigRead** have a **selectable edge** parameter. Read about these functions in the Windows Command Language section (5.6) for details.

Warning! The DIN connector pins are only protected to a maximum of 30 V with respect to the chassis or any other DMM terminal. Do not apply any voltages greater than 30 V to the DIN connector pins. Violating this limit may result in personal injury and/or permanent damage to the DMM.

4.11.2 Analog Threshold Trigger

This mode triggers the DMM at a specific input level. A command to the DMM sets a threshold value and arms the DMM Analog trigger. The DMM's local controller waits for the level crossing and captures up to 64 readings, which are saved on board; at the current DMM measurement function, range and rate. The reading rate must be set to 10 rps or higher. You can abort this mode by sending the DMM a Disarm command to the Analog Trigger.

4.11.3 Software Issued Triggered Operations

There are several software trigger functions. They cause the DMM to make a single or multiple measurements, with or without setteling readings. These include **DMMSetBuffTrigRead**, **DMMSetTrigRead**, **DMMTrigger**, **DMMBurstRead,** and **DMMBurstBuffRead**. Read about these functions in the Windows Command Language section (5.6) for details.

4.11.4 Using the PXI bus Trigger Facilities

The SMX2040 series of Digital Multimeters are designed to interface to the PXI J2 Triggers. That includes the PXI_TRIG0 through PXI_TRIG6 and PXI_STAR trigger. The trigger to the DMM is a Wire-Ored function of the external trigger from the DIN-7 connector, and the PXI_TRIGn input. The data ready signal from the SMX2040 series can be selected to drive PXI_TRIG1 through PXI_TRIG6 or PXI_STAR trigger. Read section 4.10.1 to prevent Trigger input conflicts.

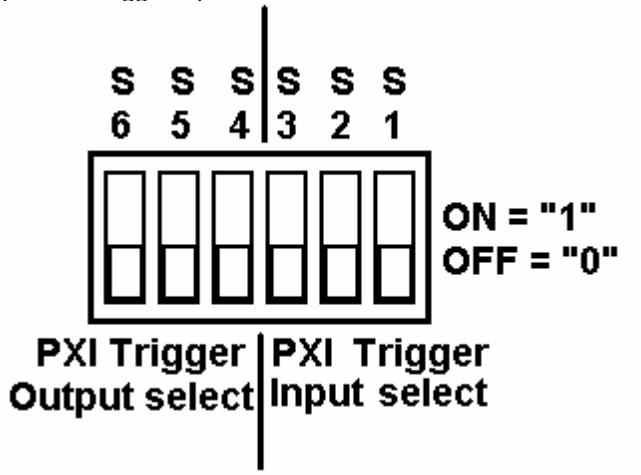


Figure 4-10. PXI Trigger input/output selection switch is located near PXI J2 connector towards the back of the DMM.

4.11.4.1 PXI Trigger Outputs

The DMM issues a Data Ready pulse each time the A/D is done making a measurement, indicating data is ready to be read. A short (about 140μs) negative pulse is issued for each measurement, with the positive edge indicating data is ready. The Trigger output is selected using S4, S5, and S6 DIP switch located near the PXI connector. The trigger pulse can be set to be output to any of the following lines.

Switch Settings S6 S5 S4	Trigger Output Routing
0 0 0	Disables trigger output
0 0 1	PXI_TRIG1
0 1 0	PXI_TRIG2
0 1 1	PXI_TRIG3
1 0 0	PXI_TRIG4
1 0 1	PXI_TRIG5
1 1 0	PXI_TRIG6
1 1 1	PXI_STAR

4.11.4.2 PXI Trigger Inputs

The trigger input to the DMM is the wired-ored signal of the trigger input from the DIN-7 connector and the PXI bus trigger. Make sure that no signal is connected to the DIN-7 trigger input while the PXI trigger is in use. When using the DIN-7 trigger input make sure the the trigger input select switches are set to Disabled position, or the selected PXI Trigger input is at a logic low level. Read about the operation of the External Hardware trigger in the above sections, since that operation pertains to both, the external and the PXI trigger input operations. The Trigger input is selected using S1, S2, and S3 DIP switch located near the J2 connector of the DMM. The DMM trigger input may be selected from any of the following lines.

Switch Settings S3 S2 S1	Trigger input Routing
0 0 0	Disables trigger input *
0 0 1	PXI_TRIG1
0 1 0	PXI_TRIG2
0 1 1	PXI_TRIG3
1 0 0	PXI_TRIG4
1 0 1	PXI_TRIG5
1 1 0	PXI_TRIG6
1 1 1	PXI_STAR

* - Rev-B hardware and above.

4.12 Frequency and Timing Measurements (SMX2042, 44)

While the maximum RMS reading is limited to the set range, you can use most of the timing functions even if the RMS voltage reading indicates Overrange. This is true as long as the input peak-to-peak value does not exceed 5.75 times the selected range (5.75 x 330 mV = 1.9 V p-p with the 330 mV range).

4.12.1 Threshold DAC

All timing measurements utilize the AC Voltage path, which is AC coupled. You need to select the appropriate ACV range prior to using the various frequency and timing measurement functions. The SMX2044 have a novel feature to accurately make these measurements for all waveforms. Unlike symmetrical waveforms such as a sine wave and square wave, non-symmetrical waves may produce a non-zero DC average at the frequency counter's comparator input. Other DMMs have the comparator hard-wired to the zero crossing. The SMX2044 include a bipolar, variable Threshold DAC for improved performance of these measurements. The Threshold DAC allows the internal timing comparator to trigger at a specific DC level. Functions affected by the Threshold DAC include frequency, period, pulse-width, duty-cycle, and the totalizer.

The Threshold DAC has 12 bits of resolution. Depending on the selected ACV range, this bipolar DAC can be set from a few mV to effectively several hundred volts (referred to the input of the DMM), positive or negative. See the Specifications sections for the limits of AC Median Value measurements and Threshold DAC settings.

The best setting of the Threshold DAC is based on the AC Median Value and Peak-to-Peak measurement described earlier. For example, a 5 V logic signal with 10% duty cycle will result in median value of 2 V, whereas a 90% duty cycle signal will have a –2 V median value. Setting the Threshold DAC to the appropriate median value will result in reliable and accurate timing measurements in each case.

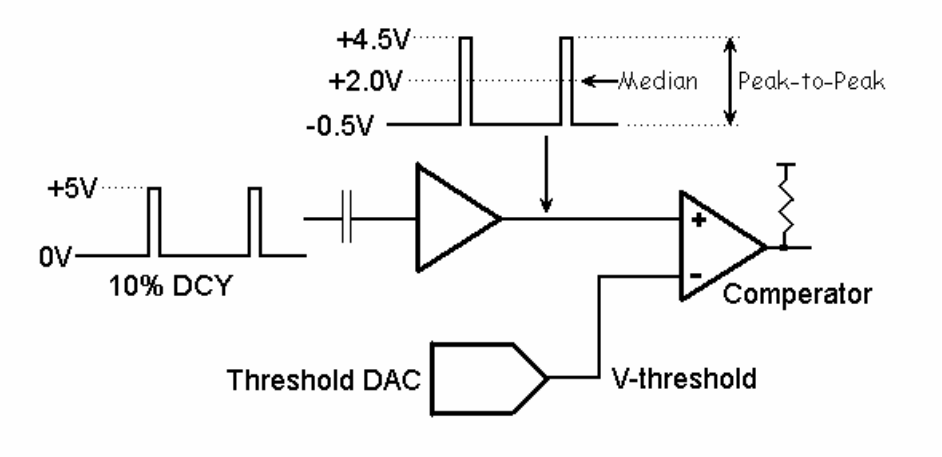


Figure 4-11. AC coupled timing measurements with Threshold DAC.

With the 3.3 ACV range selected, a 10% duty-cycle square wave with 5 V peak-to-peak value, presents a peak-to-peak signal at the internal measuring circuits of –0.5 V to + 4.5 V. The AC Median Value is +2.0 V. By setting the Threshold DAC to the Median value, the internal measuring circuits are properly biased for best performance.

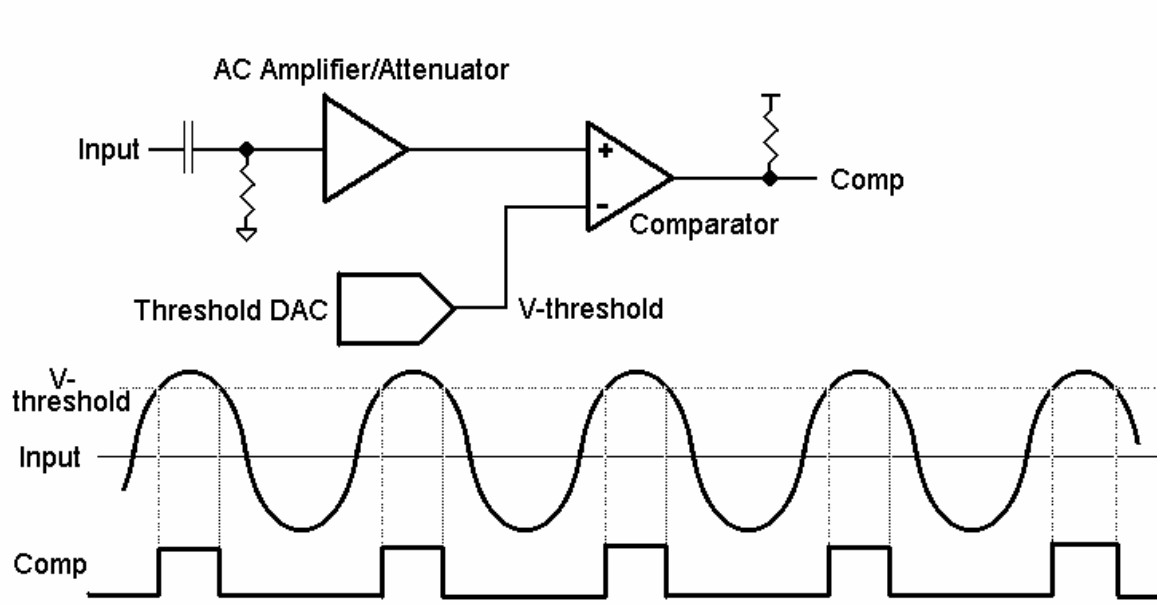


Figure 4-12. Comparator and Threshold DAC Settings

4.12.2 Frequency and Period Measurements

Both **Freq.** and **Per** check boxes are only visible when ACV or ACI functions are selected. These check boxes are used to make frequency or period measurements. **Freq.** measures from 1 Hz to 300 kHz. When activated, the control panel alternately updates the amplitude reading followed by the frequency reading. The reading rate is slower than indicated when frequency is activated. In the Windows control panel, period (**Per**) is also selectable. Once the frequency range is acquired, Frequency and Period have a maximum measurement time of about 1 second. It could take up to five measurements before the correct frequency range is auto-selected. This process is automatic. Once within range, the next frequency measurement is made at the last selected range.

Both Frequency and Period measurement performance can be improved by properly setting the Threshold DAC, a novel feature of the SMX2044. See “Threshold DAC”, “AC Median Value”, and “Peak-to-Peak” measurements for further details.

4.12.3 Duty Cycle Measurement

Duty Cycle of signals from 1 Hz to 100 kHz can be measured. The minimum positive or negative pulse width of the signal must be at least greater than 2 μs. When measuring duty cycle precisely, the voltage at which the measurement is made is important, due to finite slew rates of the signal. The Threshold voltage can be set for precise control of the level at which duty cycle is measured. For best measurement results, set the Threshold DAC to the Median value. This is particularly important for signals with low duty-cycle and small amplitude relative to the selected scale.

4.12.4 Pulse Width

User selectable positive or negative pulse widths may be measured for signal frequencies of 1 Hz to 100 kHz and a minimum pulse widths of 2 μs. The Threshold DAC feature allows measurements at a pre-defined signal level. See Threshold DAC above for more details.

To measure pulse width, the DMM must be in the AC volts range appropriate for the input voltage. Keeping the peak-to-peak amplitude of the measured signal below 5.75 times the set range will guarantee the signal is within the linear region of the AC circuitry and gives the best performance.

4.12.5 Totalizer Event Counter

The totalizer can be selected while the DMM is in the ACV mode. It is capable of counting events such as over-voltage excursions, switch closures, decaying resonance count, etc. The active edge polarity can be set for a positive or negative transition. A count of up to 10^9 can be accumulated. The maximum rate of accumulation is 30,000 events per second.

The Threshold DAC can be set for a negative or positive voltage value. See Threshold DAC above for more details.

Example One: To monitor and capture the AC line for positive spikes which exceed 10% of the nominal 120 V RMS value, first select ACV 250 V range, than set the Threshold DAC to 186.7 V. This value is the peak value of 120 V RMS plus 10% (120V + 10%) X $\sqrt{2}$). Enable the Totalizer and read it periodically to get the number of times this value was exceeded.

Example Two: Defects in coils, inductors, or transformers can be manifested as an increased decay, or greatly attenuated resonance when stimulated with a charged capacitor. The Totalizer function can be utilized to count transitions above a preset Threshold voltage as in the figure below.

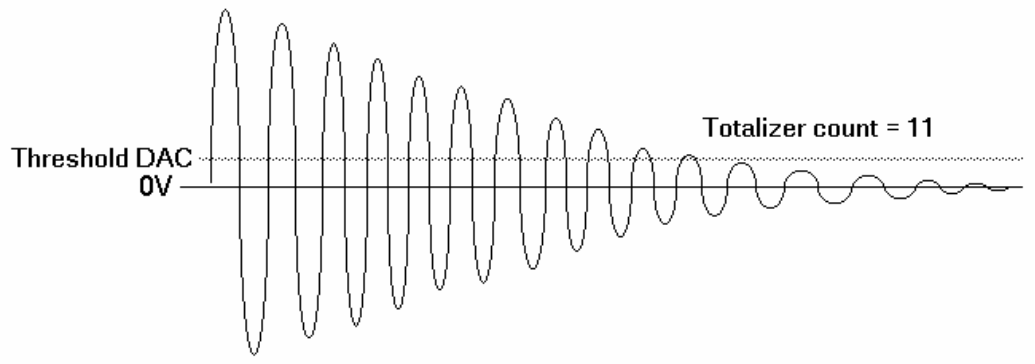


Figure 4-13. Measuring inductor Q by counting transitions of decaying resonance with preset threshold.

4.13 Sourcing Functions (SMX2044)

The SMX2044 adds a number of sourcing functions, giving greater versatility for a variety of applications. All of the available sources, VDC, VAC, IDC, are isolated (floating with respect to the chassis). This allows sourcing with a significant common mode voltage as well as the ability to connect several SMX2044 units in parallel for increased DC current, or in series for increased DC voltage.

Two digital-to-analog converters (DACs) are used for the source functions, a 12 bit DAC, and a Trim DAC. The last augments the 12 bit DAC to form a 16 bit composite DAC and adds an additional 8 bits of resolution. For functions requiring high precision, use both DACs by selecting the ClosedLoop mode, otherwise only the 12 bit DAC is utilized. DCI source is limited to the 12 bit DAC only.

All three source functions use the **V,Ω+**, and the **V,Ω-** terminals of the SMX2044.

4.13.1 DC Voltage Source

The SMX2044 has a fully isolated bipolar DC voltage source. Two modes of operation are available: fast settling or closed loop. In the Closedloop mode the DMM monitors the voltage source output, and updates it using the composite 16 bit DAC, at a rate proportional to the set measurement rate. The closedloop mode offers the best accuracy and resolution. A 10 rps or lower measurement rate is recommended for the Closedloop mode. In the fast settling mode, no adjustments are made and the 12 bit DAC is used. Up to ±10.0 V can be sourced, with 10 mA maximum drive. The output source resistance of the DCV source is approximately 250 Ω.

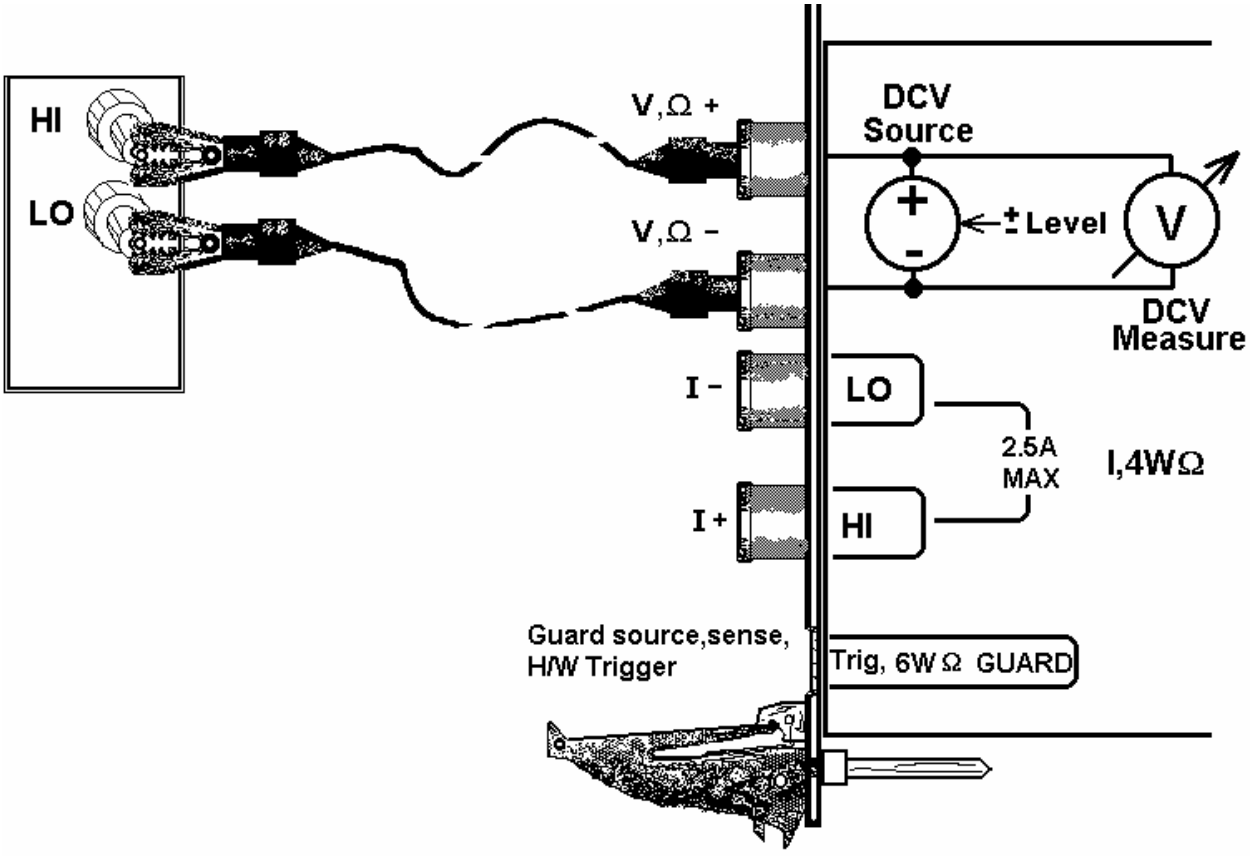


Figure 4-14. Sourcing DC voltage. The figure indicates the internal monitoring of the output in closed loop operation.

4.13.2 AC Voltage Source

The AC voltage source is fully isolated. It has two modes of operation: fast settling or closed loop. In the Closedloop mode, the source voltage is monitored, and corrections are made to the composite 16 bit DAC at a rate proportional to the set measurement rate. A 10 rps or lower reading rate is recommended for the Closedloop mode. The Closedloop mode offers the best accuracy. In the fast settling mode, the source voltage is monitored and can be displayed, but no DAC adjustments are made. Both amplitude and frequency can be set. The frequency range is 2 Hz to 75 kHz, and the amplitude is up to 20 V peak-to-peak with 10 mA maximum peak current drive. The output impedance is approximately 250 Ω.

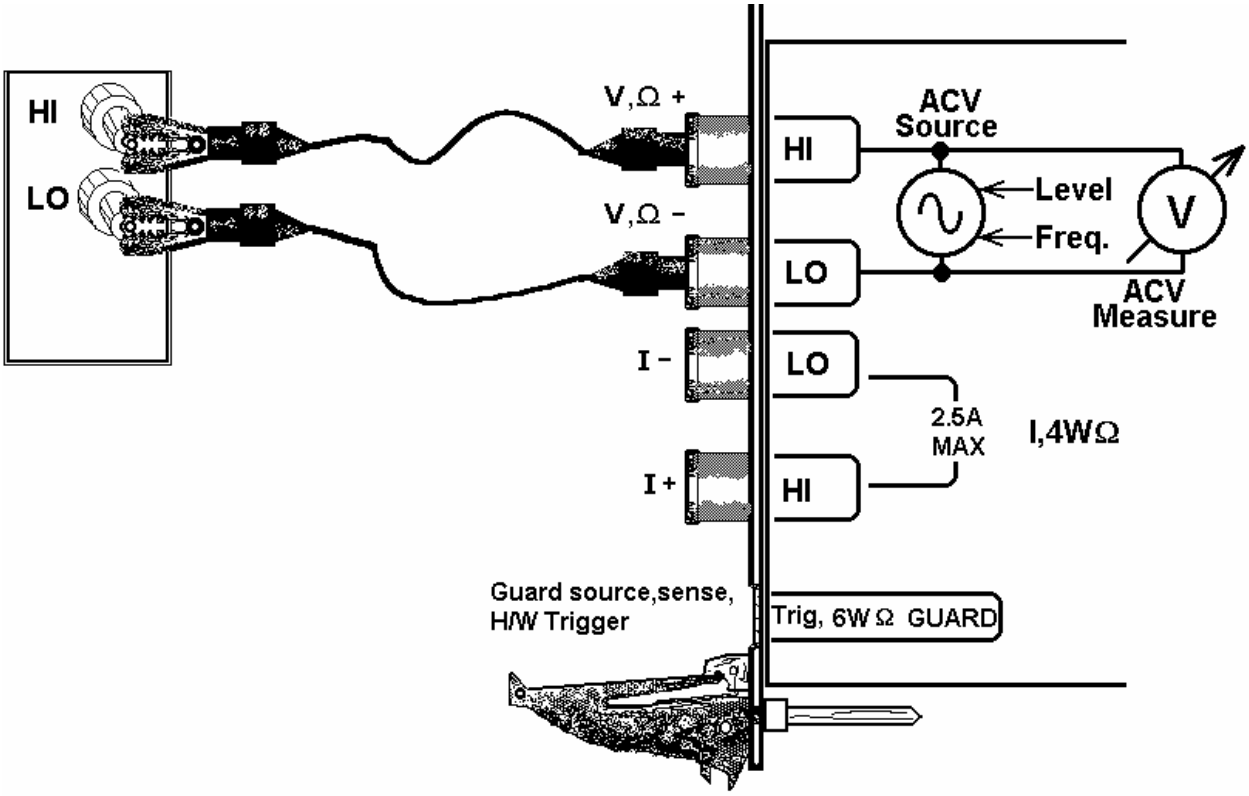


Figure 4-15. Generating AC voltage. The figure indicates the internal monitoring of the output in closed loop operation.

4.13.3 DC Current Source

The SMX2044 has a fully isolated unipolar DC current source with five ranges. It uses the 12 bit DAC to control current level. This source function is useful for parametric component measurements as well as for system verification and calibration, where a precise DC current is necessary to calibrate current sensing components.

For improved resolution of the current source, use the Trim DAC. It has to be set separately, since it is not included in the calibration record, or the control software. Use DMMSetTrimDAC() command with a parameter of 0 to 100. Further details are in Chapter 6.

4.13.4 Source Current - Measure Voltage

When sourcing current and measuring voltage, there are two connection configurations: 1) Four wire connection, where the current sourcing terminals and the voltage sense terminals are connected to the load, as in 4-wire Ohms measurement function; and 2) Two wire connection, where the current source terminals also serve as voltage sense probes as in the 2-wire Ohms measurement configuration. The first method eliminates lead resistance errors. One application is in semiconductor diode characterization discussed in Component Testing above. See Current Source Output for range details. Voltage compliance is limited to 4 V in both configurations.

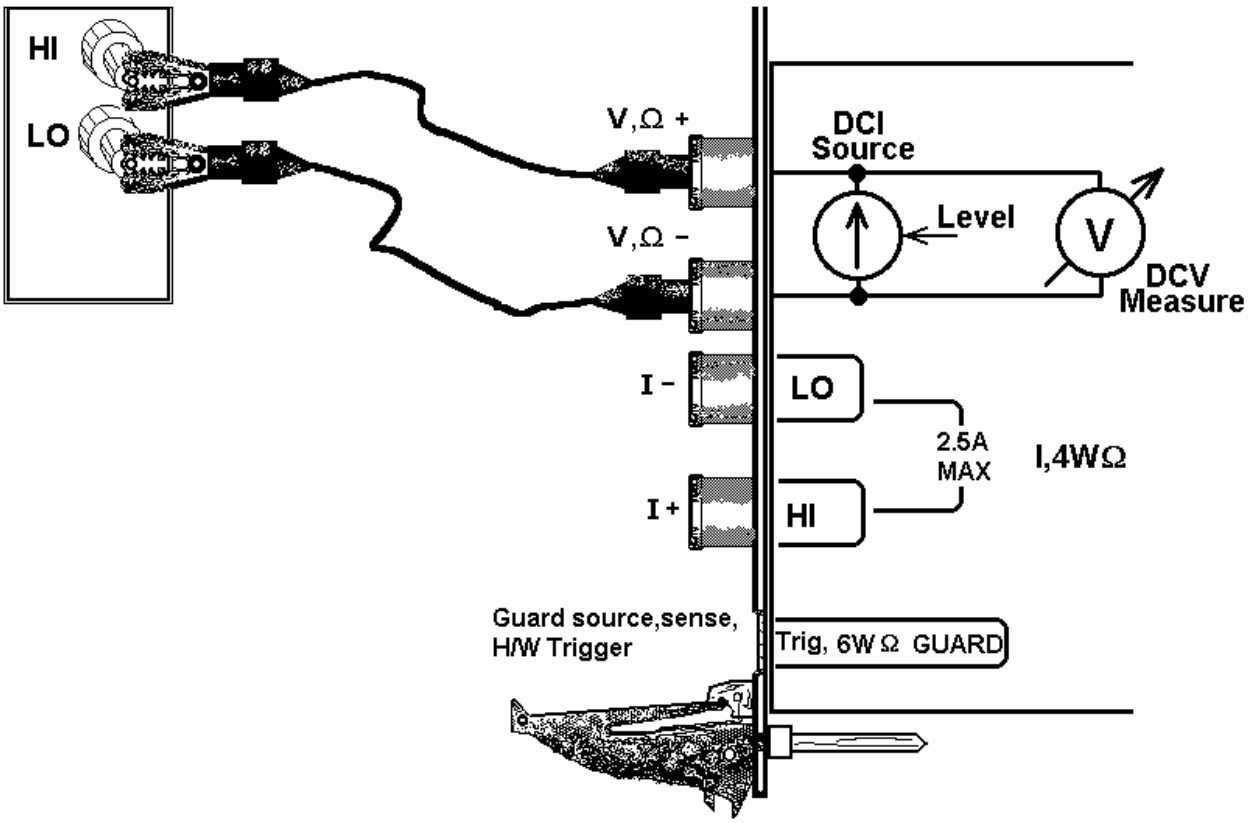


Figure 4-16. Sourcing DC current and measuring voltage in the two wire configuration. This function can be used for semiconductor parametric tests.

4.14 Synthesizing Resistance (SMX2044)

The SM2044 synthesizes resistance using an iterative method, which requires repeated reading of the DMM to make corrections to the synthesized value. While in synthesizing resistance, the readings return the measured value. There are two basic circuit types into which resistance is synthesized; current source, such as the resistance function of another Multimeter, and voltage divider type load, as in a voltage divider circuit, where one of the two resistors in the network is being synthesized as in Figure 4-16. When synthesizing a resistance into a current source, it is important to consider its compliance voltage limits. For instance, a DMM's 20k Ohms range will typically sources 100uA, and have a compliance limit of 2.5V. This means that the DMM current source does not behave as a current source when the voltage at its terminals is allowed to exceed 2.5V. Therefore synthesizing a resistance value greater than 25kΩ (2.5V/100uA) will result in the SM2044 indicating an unstable reading.

For improved accuracy, perform open terminal calibration by using the **DMMOpenTerminalCal()** function with all terminals are open.

The following are limits that must be observed while synthesizing resistances:

Limitations synthesizing into current source:

1) The maximum load current, I_S, is 10mA or $3.3/R_S$ whichever is smaller. R_S is the external resistor.
2) The maximum value that can be synthesized is $6.7/I_S – 220$ or V_C/I_S, whichever is smaller. V_C is the current source compliance voltage.
3) Resistance settings resolution equals $0.1mV/ I_S$. This means that with a source current of 1uA the resolution is 100Ω.

Limitations synthesizing into a voltage divider:

1) The maximum current the SM2044 can provide to the load is 10mA or $3.3/R_S$, or $(10 – V_x) / (R_x + R_s + 220)$, whichever is smaller. R_S is the external resistor.
2) The approximate value that can be synthesized is between $0.1 * R_s$ to about $100 * R_s$

Selecting the appropriate external resistor is very important. It sets the accuracy and range of the synthesized value.

Functions associated with the synthesis of resistance include **DMMSetExternalShunt()**, which sets the value of the external shunt, and **DMMSetResistance()**, which control the value to be synthesized. The external resistor should be 100Ω to 10MΩ. Set the measurement to 10 or higher. 20rps is optimal. The Closed loop flag does not have an effect on this function. Reselecting the function, or resetting the value has the effect of restarting the synthesis process. This will speed up the acquisition time when condition change.

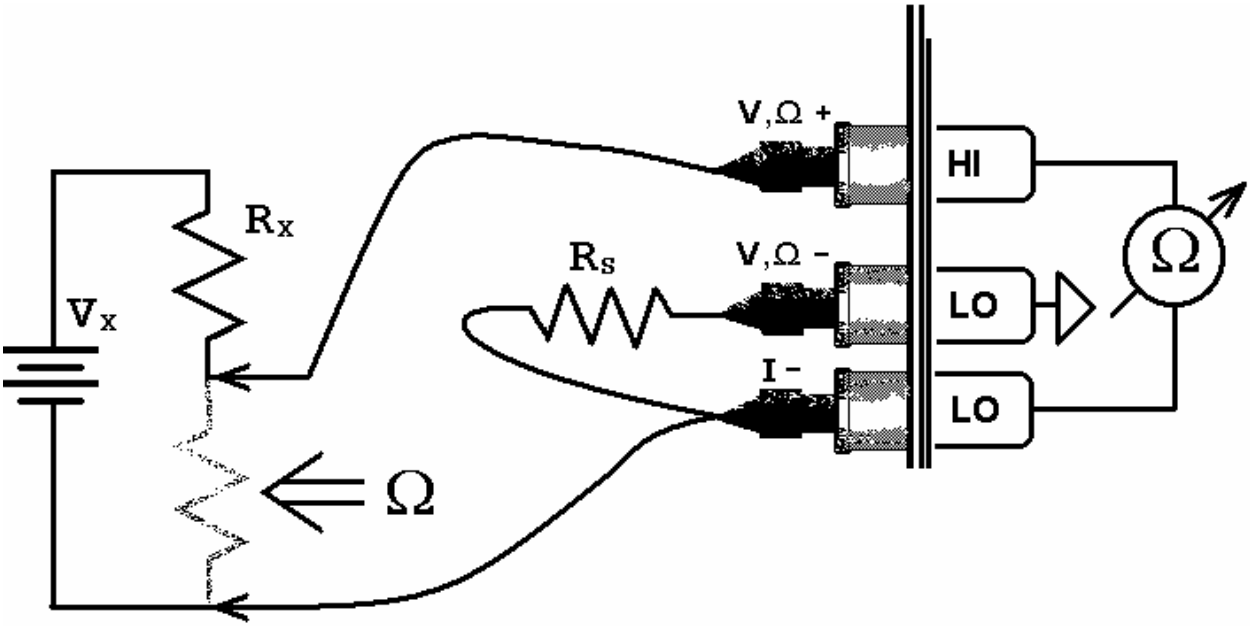


Figure 4-17. Synthesizing resistance into a voltage divider circuit.

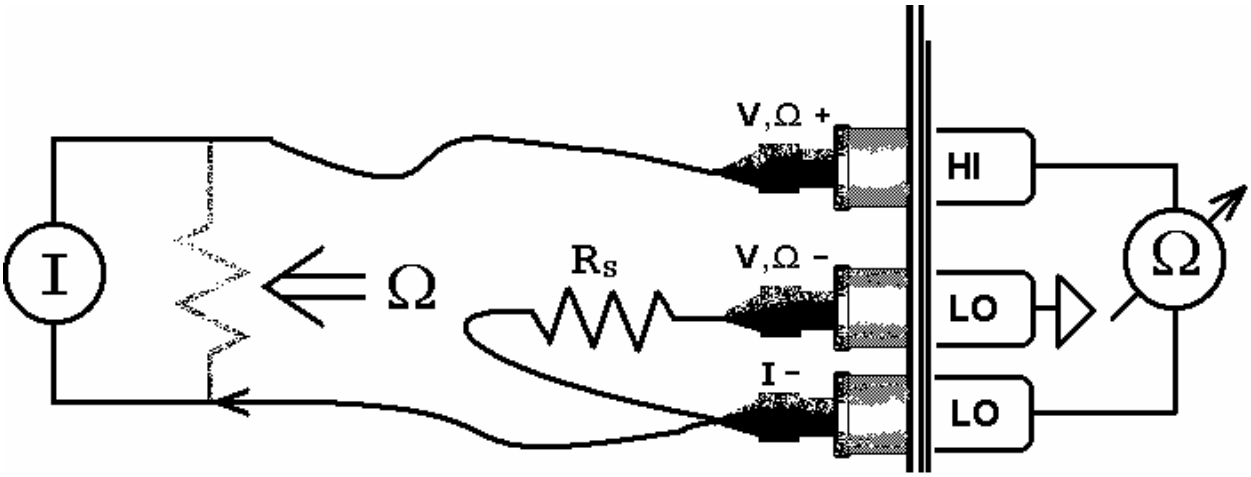


Figure 4-18. Synthesizing resistance into a current source.

4.15 Interfacing to the SMX4032 series Relay Scanners

The SMX2040 series of Digital Multimeters are designed to interface to the SM4000 series relay scanners. The following section describes both, the hardware interface and the software functions required to implement a synchronized operation.

4.15.1 Triggering the SMX2040 DMMs

The SMX2040 series can accept a hardware trigger from many sources, including the SM4000 scanners. The latter can be setup to trigger a measurement any time the scanner selects a new channel. The interface requires a single jumper between the SM4000 **Trig_com** and **Common** lines, and a connection between the SM4000 +5V and TRIG_out to the SMX2040 Trigger inputs. The various SMX4032 auto scanning operations can run independently

from the computer, whereby the Scanner selects channels from its Scan List table, and the DMM is triggered to take measurements following each channel selection.

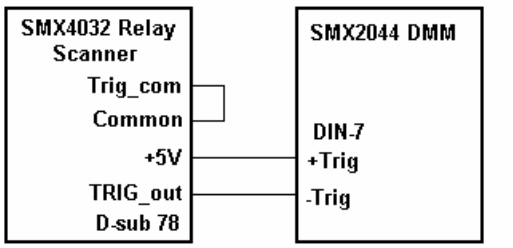


Figure 4-19. Trigger interfacing to an SMX2040 class of DMMs.

4.15.2 Multiplexing with the SMX2040 DMMs

For two wire measurements, the SMX2040 DMM must be connected to the A-Bus or the scanner, or to both, the A-Bus and C-Bus for 4-Wire measurements (assuming an SM4040 or SM4042 scanner). It is important to consider system settling time when making measurements. Time delays exist in any measurement system. These delays are contributed by various sources. These include the scanner's relay actuation times, the DMM input settling and wiring capacitance. The latter will varies with the type of measurement. For instance, when making high value Ohms measurements the DMM current source level could contribute significant delay due to the capacitance charge time. For example, with 1,000pf cable capacitance, the source current of the SMX2044 DMM using the 33MΩ range, is 0.1μA which translates to 33ms (dt = C*dV/I). It is also recommended to set the appropriate number of settling measurements for the DMM (a minimum of 4 is recommended regardless of measurement rate).

4.15.3 Interface Commands and Timing

The sequence requiring the SMX2040 DMM to make triggered measurements generated by the SMX4032 starts with the preparation of the SMX4032. Set the SMX4032 desired configuration, with Trigger Output enabled and positive polarity. Each channel selection will generate a positive pulse with duration equal to the actuation time. This could be generated by one of the scanning. The SMX2040 must be set up for triggered readings by using the **DMMSetTrigRead()** command. In the following VisualBasic® example, the SMX2040 sends readings during the scan. Since it's on board FIFO is limited to 5 readings, and the DMM must continue to send all readings during the scan, it is important to have a tight loop that reads the measurements fast enough so that no overrun error occurs. Refer to Figure 4-13 for proper trigger connection.

```
SCANTriggerOutState(nScan, Enabled, PosEdge)          // Set trigger output to Positive edge.
nReadings = 100                                       // Total number of measurements to take
DMMSetTrigRead(nDmm, 4, nReadings, NegEdge)           // Total of 100 readings and 4 settling readings
SCANAutoScan(nScan, nSteps)                //           Start auto scan
For I = 0 to nReadings -1                             // read values as they come
        while(DMMReadMeasurement(nDmm, reading) = NO     // wait for each reading and store it
Next
SCANOpenAllChannels(nScan)                            // Good idea to open all channels when done
```

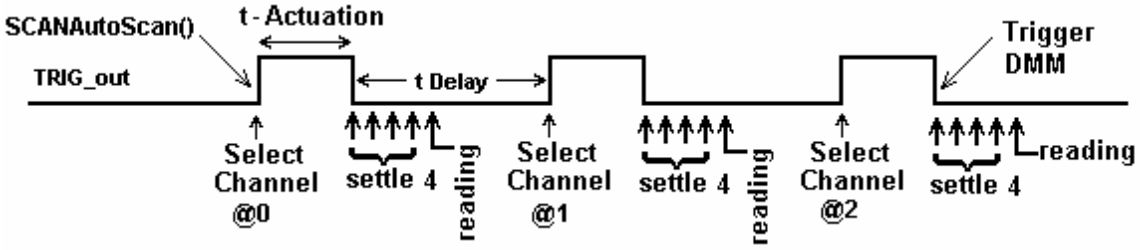


Figure 4-20. Triggered reading process and timing of SM4042 Scanner and an SMX2044 DMMs.

Unlike the previous example, **DMMSetBuffTrigRead()** is not time critical since the DMM saves all measurements to it's on-board buffer, which is read after the scan is complete. However, this function is limited to a maximum of 64 readings per scan.

```
SCANTriggerOutState(nScan, Enabled, PosEdge)          ' Set trigger output to Positive edge.
nReadings = 50                                        ' Total number of measurements to take
```

```
DMMSetBuffTrigRead(nDmm, 4, nReadings, NegEdge)    ‘ Use 4 settling readings each
SCANAutoScan(nScan, nSteps)                        ‘ Set off AutoScan
While DMMReady(nDmm) = NO                          ‘ wait for the DMM to indicate completion
Wend
For I = 0 to nReadings -1                          ‘ read values stored in the buffer
        while(DMMReadBuffer(nDmm, reading(I))      ‘ Store each reading
Next
SCANOpenAllChannels(nScan)                         ‘ Good idea to open all channels when done
While SCANReady(nScan) = NO                        ‘ Since AutoScan is a polled operation,
DoEvents                                           ' Make sure Scanner is ready
Wend
```

There are several SMX2040 family commands to considered for this operation: **DMMSetTrigRead()**, **DMMSetBuffTrigRead()**, **DMMReadMeasurement()**, **DMMReady()**, **DMMReadBuffer()** and **DMMReadBufferStr()**.

Referring to figure 4.20, the total time it takes the DMM make a reading must be set to be shorter than t-Delay, for completion of the measurements prior to the selection of the next channel.

5.0 SMX2040 DMM Windows Interface

5.1 Distribution Files

The main directory of the distribution diskette contains the Microsoft® Windows™ SMX2040 DMM software. Before installing the DMM or software, read the "Quick Install" page carefully. To install this software, enter the command "A:SETUP" in the "Run Program" menu of the Windows File Manager; or double-click on the SETUP.EXE file name from the File Explorer Tool Manager window. Most files on this diskette are compressed, and must be installed using the SETUP program.

The SMX2040 DLL is a protected-mode Microsoft® Windows™ DLL that will control the Signametrics DMM. It is provided with a sample Visual Basic™ front-panel application 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
SM40CAL.DAT	Configuration file containing calibration information for each DMM. Do not write into this file unless you are performing an external calibration! This file is normally placed at the C:\ root directory by the setup program, and should be left there. It may contain calibration records for several DMMs.
SM204032.LIB	The Windows import library. Install in a directory pointed to by your **LIB** environment variable.
SM204032.DEF	SMX2040 driver DLL module definition file.
SM204032.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**).
SM204032.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.
UserDMM.H	Header file containing all of the necessary DMM's function, range, rate definitions to be used with the various measure and source functions.
Msvbvm50.dll	Visual Basic run-time interpreter. Usually, install in your C**:\WINDOWS\SYSTEM** (or equivalent) directory. If it is not already installed, run Msvbvm50.exe for proper extraction and registration.
SM2044.vbw	Visual Basic project file
SM2044.frx	Visual Basic binary form file
SM2044.frm	Visual Basic file with main form
SM2044.vbp	Visual Basic project file
File	**Description**
2044glbl.bas	Visual Basic file with all global DMM declarations

SM2044.exe	Visual Basic DMM control panel executable
Msvcrt.dll	System file. Installs in your **C:\WINDOWS\SYSTEM** directory.
Windrvr.vxd	Win98/95 Virtual Device Driver. Installs by 'setup' in your C**:\WINDOWS\SYSTEM\VMM32** directory.
Windrvr.sys	Win NT Virtual Device Driver. Installs by 'setup' in your C**:\WINNT\SYSTEM32\DRIVERS** directory.
Install.doc	Installation instructions in MS Word

5.1.1 The SM40CAL.DAT file

The **SM40CAL.DAT** file contains calibration information for the specific DMM is it included. It determines the overall analog performance for that 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  10123    type 2044 calibration_date 06/15/1999 ; The identification should match the last digits of the SN.
ad          ; A/D compensation
72.0        20.0
vdc         ; VDC 330mV, 3.3V, 33V, 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
vac         ; VAC 1st line - DC offset. Subsequent lines: 1st entry is Offset the 2nd is gain, 3rd freq. comp
5.303       ; starting with the 330mV range, and last line is for the 250V range.
0.84                1.015461            23
0.0043              1.0256              23
0.0                 1.02205             0
0.0                 1.031386            0
idc         ; IDC 3.3mA to 2.5A ranges. 1st entry is offset, 2nd is gain parameter
-1450.0   1.00103
-176.0   1.00602
-1450.0   1.00482
-176.0   1.0
iac         ; IAC 3.3mA 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 33, 330, 3.3k,...,330Meg ranges, offset and gain
12700.0             1.002259            ;in the SMX2040, the 1st and last lines are placeholders
1256.0              1.002307
110.0               1.002665
0.0                 1.006304
0.0                 1.003066
0.0                 1.001848
0.0                 0.995664
0.0                 1.00030
…
.
.
```

The first line identifies the DMM and the calibration date. The "card-id" is stored in ROM on each DMM. During initialization the driver reads information from the DMM hardware such as serial number, DMM type, and allocated I/O space. Next it reads the corresponding calibration information from the **SM40CAL.DAT** file and makes sure there is compatibility between the hardware, the calibration recod and the software. If all is checked OK, it initialize

the DMM hardware. **DMMInit** accepts the name and location of the calibration file. A qualified technician may modify individual entries in the calibration file, then reload them using the **DMMLoadCalFile** command.

5.2 Using the SMX2040 Driver With C++ or Similar Software

Install the **SM204032.H** and **UserDMM.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 **SM204032.DEF** and **SM204032.LIB** using **ImpDef.exe** and **ImpLib.exe**, provided with the compiler.. Install **SM204032.LIB** in a directory that will be searched by the linker for import libraries. The SMX2040 software must be installed prior to running any executable code. Install the **SM204032.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 SMX2040 driver, first call **DMMInit** to read the calibration information. Call **DMMSetFunction** to set the DMM function. The DMM function constants are defined in the **UserDMM.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, which must be set to the value **nDmm**, which was returned by **DMMInit()** function. For multiple DMMs this value will be 0,1,2..n. Most functions return an error code which can be retrieved as a string using **DMMErrStr().**

Multiple Card Operations Under Windows

Single .EXE operation

Accessing multiple DMMs from a single executable is the most common way for running up to 10 DMMs using the Windows DLL. A combination of several SMX2040s and SMX2044s can be controlled, as long as the single .EXE (Thread) is used to control all of the units. Make sure that prior to issuing commands to any DMM, it is initialized using **DMMInit()**. The *nDmm* parameter is passed with each DLL command to define the DMM to be accessed. Since this configuration utilizes the DLL to service all DMMs, it must handle a single reading or control command one at a time. For example, when one DMM reads DCV, and another reads Capacitance, the DLL must finish reading the DCV before it will proceed to take a Capacitance reading. Being a relatively slow measurement, Capacitance will dictate the measurement throughput. For improved performance, one can use the following:

Multiple .EXE operation

By having several copies of **SM204032.DLL**, and renaming them, you can run multiple DMMs with separate executables. For instance, having a copy named **SM204032A.DLL** in C:\windows\system (Win98/95), and having two executable files, **MultiExe0.exe** and **MultiExe1.exe**, each of the executables will run independently, making calls to the respective DLL. This can provide an execution throughput advantage over the method mentioned above. If using VisualBasic, the **MultiExe.exe** source code should define *nDmm* = 0, and **MultiExe1.exe** should define *nDmm* = 1. In addition, the first should declare the **SM204032.DLL** and the second should declare **SM204432.DLL**:

MultiExe0.exe VB function declarations:

```
Declare Function DMMInit Lib "sm204032.dll" (ByVal calFile As String) As Long
Declare Function DMMRead Lib "sm204032.dll" (ByVal nDmm As Long, dResult As Double) As Long
NDmm = 0
```

MultiExe1.exe VB function declarations:

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

```
/*********************************************************************
 * Exmp2040.C Exmp2040.EXE
```

```
 *
 * A simple Windows .EXE example for demonstrating the SMX2040,44
 * DMMs using "C"
 * Sets Function to VDC, Range to 33V, rate to 10rps.
 * Display five measurements using a Message box.
 ************************************************************************
 * Make sure SM204032.lib is included in the libraries. For Microsoft
 * Version 4.0 C++ and above, place under 'Source Files' in the
 * Workspace, along side with Exmp2040.c
 *  PROJECT SETTINGS:
 *
 *   /nologo /ML /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE" /D "_MBCS"
 *   /FR"Release/" /Fp"Release/Exmp2040.pch" /YX /Fo"Release/" /Fd"Release/" /FD /c
 *
 *  Copy both SM204032.DLL and SM204032.LIB to the project directory.
 *
 ************************************************************************/
// #define WINAPI __stdcall
#include <windows.h>
#include <string.h>
#ifdef _Windows
  #define _WINDOWS
#endif
#include "sm204032.h"                  // functions declarations and error codes.
#include "UserDMM.h"                   // All functions, range and rate info and function declarations.
int main(void){
          int I, nDmm = 0;             // Address first DMM in the system
          char Read[16];
          char strMsg[256];
          i = DMMInit(nDmm,"C:\\sm40cal.dat");          // initialize SMX2044, and read calibration file
          if(i<0)
            MessageBox(0,"Initialization ERROR !", "Startup SM204032 DLL",MB_OK);                 // Error
          DMMSetFunction(nDmm,VDC);                     // Set to DCV function
          DMMSetRange(nDmm,_30V);                       // and to 33V  range
          DMMSetRate(nDmm,RATE_10);                     // 60 samples per sec
          strcpy(strMsg,"");                            // Clear string store
          for(i=1; i<= 5; i++){                         // take 5 readings
           DMMReadStr(nDmm,           Read);            // read
           strcat(strMsg,Read);                         // Append each reading
                    strcat(strMsg,"  ");            // insert space between readings
          }
          MessageBox(0,strMsg, "SM204032.DLL Read Resistance & VDC",MB_OK);                 // Show readings
          return 0L;
}
```

5.3 Visual Basic Front Panel Application

The Visual Basic front panel application, **SM2044.EXE,** is an interactive control panel for the SMX2040 DMM. During loading, which takes a few seconds, it initializes and self calibrates 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. It is different from the external calibration, which uses external standards and writes to the **SM40CAL.DAT** file. **S-Cal** is used to correct for any internal offset and gain drifts due to changes in operating temperature).

The **freq.** and **per** check boxes are context sensitive and appear in ACV and ACI. When **freq.** is enabled, the frequency and amplitude are shown at the same time. In this mode, the reading rate is slower than indicated. When **per** is enabled, the period is shown. The SMX2044 panel has additional capabilities, which are disabled if an SMX2040 is detected.

The source code file **GLOBAL.BAS** (in the **V_BASIC** directory of the distribution diskette) contains the function declarations and the various ranges, rates and other parameters, which 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 VisualBasic that 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 SM204032.H file.
Declare Function DMMInit Lib "sm204032.dll" (ByVal nDmm as long, ByVal calFile As String) As Long
Declare Function DMMSetRate Lib "sm204032.dll" (ByVal nDmm As Long, ByVal nRate As Long) As Long
Declare Function DMMSetFunction Lib "sm204032.dll" (ByVal nDmm As Long, ByVal nFunc As Long) As Long
Declare Function DMMSetRange Lib "sm204032.dll" (ByVal nDmm As Long, ByVal nRange As Long) As Long
Declare Function DMMRead Lib "sm204032.dll" (ByVal nDmm As Long, dResult As Double) As Long

' Definitions from UserDMM.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

'for DMMSetRate(): Reading rates
Global Const RATE_5 = 5   '5 rps
Global Const RATE_10 = 10 '10 rps
Global Const RATE_30 = 30

'Global variables
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:\sm40cal.dat")    'Initialize and load cal file
   i = DMMSetFunction(nDmm, VDCFunc)   'Set DMM to DCV function
   i = DMMSetRange(nDmm, Range2)       'Select the 33V range
   i = DMMSetRate(nDmm, RATE_10)       'Set measurement rate to 10 rps
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:

- Autoranging: Off
- Function: DC Volts
- Range: 330V
- Relative: Off
- Synchronized Mode: Off
- Measurement rate: 10 rps
- Temperature units are set to °C
- Offset Ohms: Off
- AC Caps level: 0.45V Peak.

5.5 Using the SMX2040 DLL with LabWindows/CVI®

When using the SMX2040 DLL with LabWindows/CVI, you should read the **LabWin.txt** file included with the software diskette.

An example application of SMX2040 DLL calls from LabWindows/CVI ® is shown below. It contains functions **measure_ohms()** and **measure_vdc()**, with sample calls to the SMX2040.

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 SM2040 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 **SM204032.H** header file. The **UserDMM.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).

DMMArmAnalogTrigger

SMX2040 ☑ SMX2044 ☑

Description

Arm DMM for analog level trigger operation.

#include "sm204032.h"

int DMMArmAnalogTrigger(int *nDmm,* **int** *iSamples,* **double FAR ****dThresh*)

Remarks

This function is usable for VDC, VAC, Ohms, IAC, and IDC. Setup the SMX2040 for analog level trigger operation. Following reception of this command the DMM makes measurements continuously, waiting for a value which exceeds the threshold, *dThresh.* When this occurs, a trigger is produced with identical processing as in **DMMArmTrigger**. Threshold crossing sense is determined by the first measurement following the call of **DMMArmAnalogTrigger**. If that measurement is lower than the set threshold, *dThresh*, subsequent measurements greater than *dThresh* will trigger the DMM. If the first measurement is greater than *dThresh*, subsequent measurements smaller than *dThresh* will trigger. For example, if *dThresh* is 2.00000 V and the first reading after arming the DMM is 2.500000 V, then 1.999999 V (or smaller) will trigger the DMM. On the other hand, if *dThresh* is 1.000000 V and the first reading after arming the DMM is 0.500000 V, then 1.000001 V (or greater) will trigger the DMM.

The *dThresh* value is in base units, and must be within the DMM range setting. For example, in the 330 mV range, *dThresh* must be within -0.330000 and +0.330000. In the 33 kΩ, range *dThresh* must be between 0.0 and 33.0e3.

Following an analog level trigger event, the DMM makes *iSamples* readings at the set function, range, and reading rate, and stores them in an internal buffer. Autoranging is not allowed when using **DMMAnalogTrigger**. Between the time the **DMMArmAnalogTrigger** is issued and the time the buffer is read, no other command should be sent to the DMM. One exception is the **DMMDisArmTrigger** command.

Use the **DMMReady** to monitor when the DMM is ready. When ready, you can read up-to *iSamples*, using **DMMReadBuffer** or **DMMReadBufferStr** functions. Once **DMMReady** returns **TRUE**, it should not be used again prior to reading the buffer, since it prepares the buffer for reading when it detects a ready condition.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iSamples	**int** The number of samples the DMM takes following a trigger pulse. This number must be between 1 and 64, inclusive.
dThresh	**double FAR** Analog level trigger threshold value

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

Value	Meaning
DMM_OKAY	Operation successfully terminated
Negative value	Error code.

Example

```
double Buffer[64];
DMMArmAnalogTrigger(0,64,1.5);
while( ! DMMReady(0));
for(i=0; i < 64 ; i++)
      Buffer[i] = DMMReadBuffer(0);
```

DMMArmTrigger

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Arm DMM for external trigger operation.

#include "sm204032.h"

int DMMArmTrigger(**int** *nDmm*, **int** *iSamples*)

Remarks Setup the SMX2040 for external hardware trigger operation. Following reception of this command, the DMM enters a wait state. After reception of an external trigger pulse, the DMM makes *iSamples* readings at the set function, range, and reading rate; and stores them in an internal buffer. No autoranging is allowed for external trigger operation. Between the time the **DMMArmTrigger** is issued and the time the buffer is read, no other command should be sent to the DMM. One exception is the **DMMDisarmTrigger** command. This function is usable for VDC, VAC, Ohms, IAC and IDC

Use the **DMMReady** to monitor when the DMM is ready (following trigger and the reading of *iSamples*). When ready, you can read up to *iSamples*, using **DMMReadBuffer** or **DMMReadBufferStr** functions. Once **DMMReady** returns TRUE, it should not be used again prior to reading the buffer, since it prepares the buffer for reading when it detects a ready condition.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iSamples	**int** The number of samples the DMM takes following a trigger pulse. This number must be between 1 and 64, inclusive.

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

Value	Meaning
DMM_OKAY	Operation successfully terminated
Negative Value	Error code.

Example

```
double Buffer[64];
DMMArmTrigger(0,64);
while( ! DMMReady(0));
      for(i=0; i < 64 ; i++)
      Buffer[i] = DMMReadBuffer(0);
```

DMMBurstBuffRead

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Setup the DMM for Triggered operation.

#include "sm204032.h"
#include "UserDMM.h"

int DMMBurstBuffRead(int *nDmm***, int** *iSettle***, int** *iSamples***)**

Remarks

Following reception of this command the DMM enters a burst read mode, where it takes *iSettle* + 1 readings at the set measurement function, range, and reading rate; and saves the last reading to the on-board buffer. This process repeats for *iSamples*. No other DMM command should be issued to the DMM until the it completes the operation, and the buffer is read. One exception is the **DMMDisarmTrigger** command, which terminates the process. No autoranging is allowed in this mode. This function is usable for VDC, VAC, Ohms, IAC, IDC, and RTD measurements. Measurement rate should be set to 10rps or higher. The total time it takes to complete this process is equal to *iSamples* * (*iSettle* + 1) / (measurement rate).

Use the **DMMReady** to monitor if the has completed the operation, and is ready. When ready, read up to *iSamples*, using **DMMReadBuffer** or **DMMReadBufferStr** functions. Once **DMMReady** returns TRUE, it should not be used again until the buffer is read, since it clears some flags in preparation for buffer reading when it detects a ready condition.

Parameter	Type/Description
iDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iSettle	**int** The number of settling measurements prior to read value. Must be set between 0 and 250. Recommended value is 4.
iSamples	**int** The number of samples the DMM takes following the same number of trigger pulses. This number must be between 1 and 64, inclusive.

Return Value

The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Operation successfully terminated
Negative Value	Error code.

Example

```
double Buffer[50];
```

```
DMMBurstBuffRead(0, 4, 50); // 4 settling readings for each
// measurement, and take 50 readings
while( ! DMMReady(0) );       // wait for completion
for(i=0; i < 50 ; i++)        // read 64 readings from DMM's
// on-board buffer
      Buffer[i] = DMMReadBuffer(0);
```

DMMBurstRead

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Setup the DMM for multiple readings operation, sending back measurements as they come.

#include "sm204032.h"
#include "UserDMM.h"

int DMMBurstRead(int *nDmm***, int** *iSettle***, int** *iSamples***)**

Remarks

Set the DMM to take multiple measurements, sending readings back to the computer. This function is similar to the **DMMSetTrigRead** function, with the exception that it does not wait for a hardware trigger to start making measurements. Following reception of this command the DMM enters a burst read mode, where it takes *iSettle* + 1 readings at the set measurement function, range, and reading rate; and sends the last reading to the computer. This process repeats for *iSamples*. Following the issue of this command, and until *iSampels* measurements are read, it is necessary to read the samples from the DMM using the **DMMReadMeasurement** command as fast as they become available. This will prevent an Overrun communication error, which is an indication that the rate at which measurements are read from the bus do not keep up with the DMM transmission. The DMM has five readings FIFO to lessen this problem. No autoranging is allowed in this mode. This function is usable for VDC, VAC, Ohms, IAC, IDC, and RTD measurements. Measurement rate should be set to 10rps or higher. The total time it takes to complete this process is equal to *iSamples* * (*iSettle* + 1) / (measurement rate).

Use the **DMMReadMeasurement** to monitor when reading becomes available, and to read the data. Read as many samples as *iSamples* to guarantee proper conclusion of this capture process.

Parameter	**Type/Description**
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iSettle	**int** The number of settling measurements prior to read value. Must be set between 0 and 250. Recommended value is 4.
iSamples	**int** The number of samples the DMM takes following the same number of trigger pulses. This number must be between 1 and 32,000, inclusive.

Return Value

The return value is one of the following constants.

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

Example

```
double Reading[250];
DMMBurstRead(0, 10, 250); // settle 10 reads., 250 samples
for(i=0; i < 250 ; i++) // read 250 meas. as they come
```

```
while( ! DMMReadMeasurement(0 , Reading[i]) );
```

DMMCalibrate

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Internally calibrates the DMM.

#include "sm204032.h"

int DMMCalibrate(int *nDmm*)

Remarks This function re-calibrates the DMM, and returns it to the current operating mode.

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 **SM40CAL.DAT** file.

DMMCleanRelay

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Clean specified relay.

#include "sm204032.h"

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

Remarks This function cleans *iRelay* by vibrating the contact *iCycles* . 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 numer of times the relay contac is shken. 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

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Clears the Min/Max storage.

#include "sm204032.h"

int DMMGetMin(int *nDmm***)**

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

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

Return Value Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
int status = DMMClearMinMax(0);
```

DMMClosePCI

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.h"

int DMMClosePCI(int *nDmm*)

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.

Negative Value	Error code

Example int status = DMMClosePCI(0);

DMMDelay

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Wait for a given time.

#include "sm204032.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 */

DMMDisableTrimDAC

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Terminate the operation of the Trim DAC.

#include "sm204032.h"

int DMMDisableTrimDAC(int *nDmm***)**

Remarks This function disables the Trim DAC. Since usage of the Trim DAC consumes a lot of the on-board microcontroller's resources it must be turned off with this function when not in use. See **DMMSetTrimDAC**, **DMMSetDCVSource** and **DMMSetACVSource** 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

```
DMMDisableTrimDAC(0); // Remove Trim DAC from operation
```

DMMDisArmTrigger

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Abort trigger operation.

int DMMDisArmTrigger(**int** *nDmm*)

Remarks

This function sends the DMM a trigger termination command. If the DMM is waiting for a trigger, it will exit the wait mode, and be ready for a new operation. It can be used following an external hardware or analog level trigger arm command (**DMMArmAnalogTrigger, DMMArmTrigger,** or **DMMTrigger**). It can be used with no limitation.

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

DMMDutyCycleStr

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description

Return percent duty cycle of an AC signal in string format.

#include "sm204032.h"

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

Remarks

This function is the string version of **DMMReadDutyCycle**. The measurement result is stored at the location pointed to by *lpszReading*. See **DMMReadDutyCycle** 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 16 characters long) to hold the result.

Return Value

The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Valid return.

Negative Value	Error code

Example char cBuf[17]; int status = DMMDutyCycleStr(0, cBuf);

DMMErrString

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Return the string describing the error code.

#include "sm204032.h"

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

Remarks This function returns a string containing the error description which corresponds to the *iErrorCode*. The error string is placed at *lpszError*.

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 16 characters long) to hold the error 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[48];
int length = DMMErrString( -3, cBuf, 48);
```

DMMFrequencyStr

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return the next DMM frequency reading, formatted for printing.

#include "sm204032.h"

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

Remarks This function makes frequency measurement and returns the result as a string formatted for printing. The print format is fixed to six digits plus units, e.g., 05.001 Hz. If the DMM is in autorange, be certain to take an amplitude reading before using this command. It may take several calls to **DMMFrequencyStr()** to get the measured frequency, because the DMM frequency counter uses a frequency ranging scheme which gets activated only when a frequency or period reading function is received. If the previously measured frequency was 1 Hz and the frequency being measured is 300 kHz (or vise versa), it might take as many as six calls to **DMMFrequencyStr()** or any of the other frequency measurement functions, to read the correct frequency. This function is a **Secondary** function which requires the DMM to be in either VAC or IAC function and at the appropriate range.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszReading	**LPSTR** Points to a buffer (at least 16 characters long) to hold the converted result.

Return Value

The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Operation successfully completed.
DMM_CNT_RNG	Frequency counter is over or under range.
Negative Value	Error code

Example

```
char cBuf[17];
int status;
status = DMMFrequencyStr(0, cBuf);
```

DMMGetACCapsR

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Return the resistance component of the last AC Caps measurement.

#include "sm204032.h"

int DMMGetACCapsRint *nDmm***, double ***lpdResult***)**

Remarks This function retrieves the resistance value from last reading of AC based Capacitance measurement. It 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*. Returned result is a value in ohms. Read about In-Circuit Capacitance Measurements in section 4.8 of this manual.

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

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

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code
OVER_RNG	Over range occurred, implying a very high parallel resistance value.

Example

```
double d;
int status;
status = DMMGetACCapsR(0, &d);
```

DMMGetBusInfo

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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 of 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 (0 to 15)

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
DMMGetCalDate(3, &bus, &slot); // DMM#3
```

DMMGetCalDate

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszCalDate	**LPSTR** Points to a buffer (at least 16 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[16];
int status;
status = DMMGetCalDate(0, cBuf);
```

DMMGetdB

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.h"

int DMMGetdB(int *nDmm***, double FAR *** *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 FAR *** 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 FAR dB; int status = DMMGetdB(0, &dB);

DMMGetdBStr

SMX2040 ☑ SMX2044 ☑

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

#include "sm204032.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 16 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[32]; int strLength = DMMGetdBStr(0, cBuf);
```

DMMGetCJTemp

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Retrieve the currently set cold junction temperature.

#include "sm204032.h"

int DMMGetCJTemp(int *nDmm***, double *l**p*d*Temp***)**

Remarks Get the currenly set cold juncion temperature. For more details see **DMMSetCJTemp()** function.

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

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

Value	Meaning
DMM_OKAY	Operation successfully terminated
Negative Value	Error code.

Example
```
DMMGetCJTemp(0, &temp);
```

DMMGetDeviation

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.h"

int DMMGetDeviation(int *nDmm***, double FAR *l**p*d*Dev***)**

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. 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.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdDev	**double FAR *** 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 FAR error;
int status = DMMGetDeviation(0, &error);
```

DMMGetDeviatStr

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.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 16 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[32];
int strLength = DMMGetDeviatStr(0, cBuf);
```

DMMGetFuncRange

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get DMM range code.

#include "sm204032.h"
#include "UserDMM.h"

int DMMGetFuncRange(**int** *nDmm*)

Remarks This function returns the combined DMM function/range code. See **UserDMM.h** for the complete set of codes.

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

Return Value Integer value corresponding to the currently set DMM function/range, or an error code. The following are a few examples of the returned value.

Value	Meaning

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

Example

```
if(DMMGetFnRange == VDC_300mV) printf("Lowest VDC range
selected");
```

DMMGetFunction

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get DMM function code.

#include "sm204032.h"
#include "UserDMM.h"

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

Remarks This function returns the DMM function code.

Parameter	**Type/Description**
nDmm	**int** Identifies the DMM nuber. Zero being the first.

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

Value	**Meaning**
Positive value	See UserDMM.h for function codes.
Negative Value	Error code

Example

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

DMMGetGrdVer

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get DMM firmware version.

#include "sm204032.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

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get the hardware version of the DMM.

#include "sm204032.h"

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

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

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

Return Value DMM hardware code or an error code.

Value	Meaning
Positive value	Hardware version code
Negative Value	Error code

Example

```
int HWVer = DMMGetHwVer(0);
```

DMMGetID

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get DMM ID code.

#include "sm204032.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 **SM40CAL.DAT.**

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

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get Manufacturing date stamp from the DMM hardware

#include "sm204032.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

SMX2040 ☑ SMX2044 ☑

Description Get Maximum reading history.

#include "sm204032.h"

int DMMGetMax(int *nDmm*, **double FAR ****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 is only applicable to **Primary** read functions (those that are read using **DMMRead**, **DMMReadStr** or **DMMReadNorm**). This value is updated every time one of those functions is used.

Parameter	**Type/Description**

nDmm	**int**	Identifies the DMM. DMMs are numbered starting with zero.
lpdMax	**double FAR ***	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 FAR Mx; int status = DMMGetMax(0, &Mx);

DMMGetMaxStr

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Returns the maximum as a formatted string.

#include "sm204032.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 16 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[17];
int status = DMMGetMaxStr(0, cBuf);
```

DMMGetMin

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get Minimum reading history.

#include "sm204032.h"

int DMMGetMin(int *nDmm*, **double FAR *** *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 is only applicable to **Primary** read functions (those that are read using **DMMRead**, **DMMReadStr** or **DMMReadNorm**). This value is updated every time one of those functions is used.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdMax	**double FAR *** 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 FAR Min; int status = DMMGetMin(0, &Min);
```

DMMGetMinStr

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Returns the minimum as a formatted string.

#include "sm204032.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 16 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[17];
int status = DMMGetMinStr(0, cBuf);
```

DMMGetRange

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get DMM range code.

#include "sm204032.h"
#include "UserDMM.h"

int DMMGetRange(int *nDmm*)

Remarks This function returns the DMM range code.

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

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get DMM reading rate

#include "sm204032.h"

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

Remarks This function returns a double floating rate in readings per second.

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

Return Value Integer value version code or an error code.

Value	Meaning
Negative Value	Error code

Example

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

DMMGetSourceFreq

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Get the currently set ACV Source frequency.

#include "sm204032.h"

int DMMGetSourceFreq(int *nDmm*, **double FAR ***_lpdFreq_**)**

Remarks This function returns a double floating value that is the currently set ACV source frequency of the SMX2044. It can be used to display or verify the default frequency of the stimulus for the various Inductance measurement ranges.

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

Return Value Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
double FAR f; int status = DMMGetSourceFreq(0, &f);
```

DMMGetTCType

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get the themocouple type currently selected.

#include "sm204032.h"
#include "UserDMM.h"

int DMMGetTCType(int *nDmm*)

Remarks This function returns the Themocouple type currently selected.

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
Btype to TType	Type of thermocuple as specified in UserDMM.h file
Negative Value	Error code

Example

```
int TCtype = DMMGetTCType(0);
```

DMMGetType

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get the type of the DMM.

#include "sm204032.h"

int DMMGetType(int *nDmm*)

Remarks This function returns the DMM type.

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
2040	SMX2040 is at nDmm slot
2042	SMX2042 is at nDmm slot
2044	SMX2044 is at nDmm slot
Negative Value	Error code

Example

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

DMMGetVer

SMX2040 ☑ SMX2044 ☑

Description Get DMM software driver version.

#include "sm204032.h"

int DMMGetVer(int *nDmm***, double FAR *** *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 FAR *** 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

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Initialize a DMM.

#include "sm204032.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 *lpszCal*.

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 **SM40CAL.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

```
int i = DMMInit(0,"C:\SM40CAL.dat");// Initialize 1st DMM
```

DMMIsAutoRange

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get the status of the autorange flag.

#include "sm204032.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

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get the status of the DMM.

#include "sm204032.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 and should not be addressed. 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

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Get the status of the Relative flag.

#include "sm204032.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);
```

DMMLoadCalFile

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Reload calibration record from file.

```
#include "sm204032.h"

int DMMLoadCalFile(int nDmm, LPCSTR lpszCal)
```

Remarks This function is provides the capability to reload the calibration record. This is useful in making limited calibration adjustments to the DMM. By having a copy of the original calibration file '**SM40CAL.DAT**' open with an editor, and modifying calibration entries, then reloading it using **DMMLoadCalFile**, one can instantly verify the corrections made. Make sure the '**SM40CAL.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");
```

DMMOpenPCI

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Open the PCI bus for the specified DMM. Not for user application.

#include "sm204032.h"

int DMMOpenPCI(int *nDmm*)

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
int status = DMMOpenPCI(0);
```

DMMOpenCalACCaps

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Calibrate the AC based in circuit capacitance function.

#include "sm204032.h"

int DMMOpenCalACCapsl(int *nDmm*)

Remarks This function characterizes the selected range of the AC Capacitance measurement path and source, which is required prior to making measurements. For better accuracy it should be performed frequently. The Open Terminal calibration should be performed with the test leads connected and open. This function characterizes the stimulus source at the specific frequency associated with the selected range. It takes about fifteen seconds to complete the process. Make sure to perform this operation for each range you intend to use.

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 = DMMOpenCalACCaps(0);

DMMOpenTerminalCal

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Calibrate the Inductance measurement function with open terminals.

#include "sm204032.h"

int DMMOpenTerminalCal(int *nDmm*)

Remarks This function characterizes the Inductance measurement path and source, which is required prior to making inductance measurements. It should be performed within one hour, before using the inductance measurements. For better accuracy it should be performed more frequently. The Open Terminal calibration should be performed with the test leads open. The **DMMOpenTerminalCal** sweeps the inductance stimulus source across the full bandwidth, and makes measurements at several points. It takes about twenty seconds to complete the process. For a complete characterization of the Inductance measurement system it is also necessary to perform the inductance zero operation with the inductance range and frequency selected, using the Relative function and with the probes shorted.

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 = DMMOpenterminalCal(0);
```

DMMPeriodStr

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return the next DMM period reading, formatted for printing.

#include "sm204032.h"

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

Remarks

This function makes a period measurement and returns the result as a string formatted for printing. The print format is fixed to five digits plus units, e.g., 150.01 ms. See **DMMFrequencyStr()** 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 16 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.

Value	Meaning
DMM_OKAY	Operation successfully terminated
Negative Value	Error code
DMM_CNT_RNG	Period measurement H/W is over or under range.

Example

```
char cBuf[17];
int status;
status = DMMPeriodStr(0, cBuf);
```

DMMPolledRead

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Tests the DMM for ready status, and returns the next floating-point reading.

#include "sm204032.h"

int DMMPolledRead(int *nDmm***, double FAR ***_lpdResult_**)**

Remarks **DMMPolledRead** polls the DMM for readiness. If the DMM is not ready it will return **FALSE**. If the DMM is ready with a new reading it will return **TRUE**, and the reading will be placed at the location pointed to by *lpdResult*. See **DMMPolledReadCmd** for more details. Do not use **DMMReady** to check for readiness since it will cause communication failure.

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

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

Value	Meaning
FALSE	DMM is not ready
TRUE	DMM is ready, and reading is placed at *lpdResult*
Negative Value	Error code

Example

```
double read;
if(DMMPolledRead(0, &d)) fprintf("%9.4f\n",d);
```

DMMPolledReadCmd

SMX2040 ☑ SMX2042 ☑SMX2044 ☑

Description Send DMM Polled Read command.

#include "sm204032.h"

int DMMPolledReadCmd(**int** *nDmm*)

Remarks If the DMM is not busy with a prior Polled read process, this function will trigger the DMM to execute a single read command. The DMM must be set to a specific range and one of the following functions to use the polled read command: VDC, VAC, IDC, IAC, 2-wire, 4-wire, 6-wire, or RTD function. Composite functions such as Capacitance, Inductance, Peak-to-Peak etc. are not capable of polled read operation. Measurement rate must be 10 rps or higher. If FALSE is returned, the DMM is busy processing a prior polled read. A DMM_OKAY indicates the DMM accepted the read command and entered the busy state. The DMM remains busy until it is ready with the next reading. This function is useful where it is necessary to conserve CPU time and make the DMM a polled device. Use **DMMPolledRead** or **DMMPolledReadStr** to test for readiness and read measurement. Do not use **DMMReady** to check for readiness since it will cause communication failure.

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

Return Value DMM_OKAY if command accepted, else FALSE or an error code.

Value	Meaning
FALSE	DMM is busy and can't execute a polled read command.
DMM_OKAY	Operation successful. DMM entered busy state
Negative Value	Error code

Example

```
int status = DMMPolledReadCmd(0);
```

DMMPolledReadStr

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description If DMM is ready, return the next reading from the DMM formatted for printing.

#include "sm204032.h"

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

Remarks This function is the string version of **DMMPolledRead**. See **DMMPolledRead** 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 16 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
FALSE	DMM is not ready
TRUE	DMM is ready, and reading is placed at *lpszReading*
Negative Value	Error code

Example

```
char strMsg[32];
if(DMMPolledReadStr(0, strMsg)) MessageBox(0,strMsg,
"SMX2044",MB_OK);  // display readings;
```

DMMRead

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.h"

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

Remarks **DMMRead** reads the next result from 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 **DMMSetFuncRange()**)

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdResult	**double FAR *** 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
DMM_E_RANGE	DMM over range error occurred.

Example

```
double d;
int status;
status = DMMRead(0, &d);
```

DMMReadBuffer

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Return the next double floating-point reading from the DMM internal buffer.

#include "sm204032.h"

int DMMReadBuffer(**int** *nDmm*, **double FAR ***lpdResult*)

Remarks Read the next measurement from the DMM internal buffer, pointed to by an internal buffer pointer, and increment the pointer. Store the measurement as a 64-bit double-precision floating-point number in the location pointed to by *lpdResult*. See **DMMArmTrigger()** functions for more detail.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdResult	**double FAR *** Points to the location which holds the frequency.

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error Code

Example

```
double Buffer[10];
int status;
DMMArmTrigger(0,10);    // Set up for 10 triggered samples
while( ! DMMReady(0));
for(i=0; i < 10 ; i++)
      status = DMMReadBuffer(0, &Buffer[i]);
```

DMMReadBufferStr

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Return the next reading, formatted for printing.

#include "sm204032.h"

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

Remarks

The same as **DMMReadBuffer()** except the reading is formatted as a string with units. Measurements are stored as a null terminated string at the location pointed to by *lpszReading*.

Parameter	**Type/Description**
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszReading	**LPSTR** Points to the location which holds the formatted reading string.

Return Value

The return value is one of the following constants.

Value	**Meaning**
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
char Buf[17];
DMMArmTrigger(0,1);     // take a single triggered sample
while( !DMMReady(0));
DMMReadBufferStr(0, Buf);
```

DMMReadCJTemp

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Read cold junction temperature for thermocouple measurement.

#include "sm204032.h"

int DMMReadCJTemp(int *nDmm***, double ***lp*dTemp***)**

Remarks Read the cold juncion temperature sensor for subsequent thrermocouple measurements. When measuring temperature using thermocouples it is necessary to establish a reference or cold junction temperaturem. This is the temperature at which the themocouple wires are connected to the DMM or to the switching card's cooper wires. One way to do this is by measuring the cold junction sensor using this function. **DMMReadCJTemp()** function reads the sensor output voltage (0 to +/-3.3V), and converts it to cold junction temperature using the built in equation Temp = b + (Vcjs – a)/m. The default values of a, b and m are designed specifically for the temperarute sensor of the SM40T terminal block. The value of the cold junction temperature is saved internally for subsequent thermocouple measurements as well as return at the location pointed to by lp*dTemp*.

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

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

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

Example

```
DMMReadCJTemp(0, &temp);
```

DMMReadCrestFactor

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return ACV signal's Crest Factor.

#include "sm204032.h"

int DMMReadCrestFactor(int *nDmm,* **double FAR *** *lpdResult*)

Remarks This is a **Secondary** function and the DMM must be in ACV measurement function, and a valid range must be set. A double-precision floating-point Crest Factor is stored in the location pointed to by *lpdResult*. This measurement is a composite function, utilizing several sub functions, and could take over 10 seconds to perform. See the Crest Factor measurement section of the manual for more detail.

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example
```
double CF; int status = DMMReadCrestFactor(0, &CF);
```

DMMReadDutyCycle

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return percent duty cycle of ACV signal.

#include "sm204032.h"

int DMMReadDutyCycle(int *nDmm,* **double FAR *** *lpdDcy*)

Remarks This is a **Secondary** function and the DMM must be in AC measurement function, and a valid range must be set. It returns percent duty cycle of the signal. It is stored as double-precision floating-point numbers in the location pointed to by *lpdDcy*. The measured duty cycle is effected by the setting of the Threshold DAC.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdDcy	**double FAR *** Points to the location which holds the duty cycle.

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example
```
double dcy; int state; state = DMMReadDutyCycle(0, &dcy);
```

DMMReadFrequency

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.h"

int DMMReadFrequency(int *nDmm,* **double FAR *** *lpdResult*)

Remarks If frequency counter is not engaged, select it. Make a single frequency measurement, and store the result as a 64-bit double-precision floating-point number in the location pointed to by *lpdResult*. See **DMMFrequencyStr()** for more details.

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
DMM_E_INIT	DMM is uninitialized. Must be initialize prior to using any function.
DMM_E_DMM	Invalid DMM number.
DMM_CNT_RNG	Frequency counter is over or under range.

Example

```
double d;
int status = DMMReadFrequency(0, &d);
```

DMMReadFrequencyStr

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return the next string containing frequency reading from the DMM.

#include "sm204032.h"

int DMMReadFrequencyStr(**int** *nDmm*, **LPSTR** *lpszFrequency*)

Remarks This is the string version of **DMMReadFrequency().** It returnes a formated string containing the measured frequency. For more details read about **DMMReqadFrequency()**.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszFrequency	**LPSTR** Points to the location which holds the formatted reading string. Allow minimum of 64.

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative value	Error code

Example

```
char buffer[64];
int status = DMMReadFrequencyStr(0, buffer);
```

DMMReadInductorQ

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Return inductor's Q value.

#include "sm204032.h"

int DMMReadInductorQ(int *nDmm,* **double FAR *** *lpdResult*)

Remarks This is a **Secondary** function and the DMM must be in the Inductance measurement function, and a valid inductance value must have been read prior to using this function. Resulting Q is stored as double-precision floating-point number in the location pointed to by *lpdResult*.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdResult	**double FAR *** Points to the location to hold the inductor's Q.

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
double Q;
int status = DMMReadInductorQ(0, &Q);
```

DMMReadMeasurement

SMX2040 ☑ SMX2044 ☑

Description Return a reading which is the result of **DMMSetTrigRead** operation.

#include "sm204032.h"

int DMMReadMeasurement(int *nDmm***, double FAR *** *lpdRead***)**

Remarks This measurement reading function is designed to read triggered measurements from the DMM. It is a fast reading function. It returns **FALSE** while no new reading is ready. If a reading is ready, TRUE is returned, and the result in the form of a 64-bit double-precision floating-point number is placed at the location pointed to by *lpdRead*.The returned value is in base units. That is, it will returns 0.3 for a 300mV input and 1e6 for 1.0 Mohm measurement. This function is designed to read bursting measurements form the DMM, resulting from **DMMSetTrigRead** and **DMMBurstRead** operations.

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

Return Value Integer value version code or an error code.

Value	Meaning
TRUE	Measurement was read into **lpdRead*
FALSE	No measurement is available
TIMEOUT	Communication timeout. No reading available within 9s.
OVERRUN	Communication overrun. CPU did not keep up with DMM transmission.
Other Negative Value	Error code.

Example

```
double Reading[150];
DMMBurstRead(0, 4, 150); // 4 settel., 150 samples
for(i=0; i < 150 ; i++)       // read 150 measurements
      while( DMMReadMeasurement(0 , Reading[i]) == FALSE );
// wait for all measurements to be ready, and read them.
```

DMMReadMedian

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return ACV signal's Median value.

#include "sm204032.h"

int DMMReadMedian(int *nDmm*, **double FAR *** *lpdResult*)

Remarks This is a **Secondary** function and the DMM must be in ACV measurement function, and a valid range must be set. A double-precision floating-point Median voltage result is stored in the location pointed to by *lpdResult*. This measurement is a composite function which utilizes several sub functions, and could take over 10 seconds to perform. See the Median measurement section of the manual for more detail.

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
double Median; int status = DMMReadMedian(0, &Median);
```

DMMReadNorm

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Take a reading that is in base value.

#include "sm204032.h"

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

Remarks This **Primary** read function is similar to **DMMRead()**. It returns a

double floating-point reading. The returned value is corrected for base units. That is, it returns 0.3 for a 300 mV input and 1e6 for 1.0 MOhm.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdRead	**double FAR *** 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.
DMM_E_DMM	Invalid DMM number.
DMM_OKAY	Valid return.

Example

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

DMMReadPeakToPeak

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return ACV signal's peak-to-peak value.

#include "sm204032.h"

int DMMReadPeakToPeak(int *nDmm*, **double FAR *** *lpdResult*)

Remarks This is a **Secondary** function and the DMM must be in ACV measurement function, and a valid range must be set. A double-precision floating-point peak-to-peak voltage result is stored in the location pointed to by *lpdResult*. This measurement is a composite function which utilizes several sub functions, and could take over 10 seconds to perform.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdResult	**double FAR *** Points to the location to hold the Peak-to-Peak value.

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
double ptp; int status = DMMReadPeakToPeak(0, &ptp);
```

DMMReadPeriod

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.h"

int DMMReadPeriod(**int** *nDmm*, **double FAR ***_lpdResult_)

Remarks This is a **Secondary** function and the DMM must be in ACV measurement function, and a valid range must be set It makes a single period measurement, and stores the result as a double-precision floating-point number in the location pointed to by *lpdResult*. See **DMMFrequencyStr()** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdResult	**double FAR *** Points to the location which holds the period.

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code
DMM_CNT_RNG	Period measurement hardware is over or under range.

Example

```
double d;
int status;
status = DMMReadPeriod(0, &d);
```

DMMReadStr

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.h"

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

Remarks This function is the string version of **DMMRead()**. It reads the next **Primary** 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 16 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[17]; int status = DMMReadingStr(0, cBuf);
```

DMMReadTotalizer

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Read the totalized value accumulated by the Totalizer function.

#include "sm204032.h"

long DMMReadTotalizer(int *nDmm*)

Remarks This function reads the total value accumulated by the Totalizer function. For details see **DMMStartTotalize**.

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

Return Value The return value is the totalized count, or if negative one of the following constants.

Value	**Meaning**
Negative Value	Error code

Example

```
long total = DMMReadTotalizer(0);
```

DMMReadWidth

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return the positive and negative pulse widths.

#include "sm204032.h"

int DMMReadWidth(**int** *nDmm*, **double FAR ***lpdPwid*, **double FAR ***lpdNwid*)

Remarks This is a **Secondary** function and the DMM must be in ACV measurement function, and a valid range must be set. It returns two parameters: positive and negative pulse widths. These parameters are stored as double-precision floating-point numbers in the location pointed to by *lpdPwid and lpdNwid*. The measured widths are affected by the setting of the Threshold DAC.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdPwid	**double FAR *** Points to the location which holds the positive width.
lpdNwid	**double FAR *** Points to the location which holds the negative width.

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
double pw,nw; int state; state = DMMReadWidth(0, &pw, &nw);
```

DMMReady

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Return the ready state of the DMM following trigger operation.

#include "sm204032.h"

int DMMReady(**int** *nDmm*)

Remarks Following the completion of an tirggered measurement event, be it hardware or software, the DMM indicates the completion. The **DMMReady** function checks the DMM and returns TRUE if ready, and FALSE otherwise. Once a TRUE status is returned, the **DMMReady** function should not be used again since a TRUE also indicates that some flags have been clear, which allow further operations. See **DMMArmAnalogTrigger**, **DMMArmTrigger**, **DMMTrigger**, **DMMReadBuffer** and **DMMPolledReaed** for more details on this function.

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
TRUE	DMM is done and buffer is ready to be read.
FALSE	DMM is not ready.
Negative Value	Error code

Example

```
double Buffer[10];
DMMTrigger(0,10);
while( ! DMMReady(0) );
      for(i=0; i < 10 ; i++) DMMReadBuffer(0, &Buffer[i]);
```

DMMSetACCapsDelay

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the measurement delay of AC based Capacitance.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetACCapsDelay(int *nDmm***, double** *ldDelay***)**

Remarks This **Secondary** function sets the AC based capacitance measurement delay, which is the time the measurement system settles. The DMM's default value is 2.0s. This function can set this function from 0.0 to 10.0 seconds. Since the DMM is optimized for the defalut value, it is possible that changing this value will introduce additional error.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
ldDelay	**double** The time the DMM is allowed to settle the measurement. Can be set beetween 0 and 10.0 seconds.

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetACCapsDelay(0, 0.25); // Set measurement delay to 0.25s
```

DMMSetACCapsLevel

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the DCV source output level.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetACCapsLevel(int *nDmm***, double** *ldVolts***)**

Remarks This **Secondary** function sets the AC peak voltage level for the AC based Capacitance measurement function. It actually sets an internal register to *ldVols* rather than setting the output level itself. This value is used on any of the AC Caps calibration and measurement. Following setting of this function, it is necessary to perform open calibration of the AC Capacitance ranges to be used. Since the DMM is optimized for the defalut value, it is recommended not to use this function and keep the default 0.45V peak value.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
ldVolts	**double** Peak value of AC voltage to be set. Can be 0.1V to 5.0V

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example
```
DMMSetACCapsLevel(0, 0.35); // Set source to 0.7V peak-to-peak
```

DMMSetACVSource

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the ACV source output level and frequency.

```
#include "sm204032.h"
#include "UserDMM.h"

int DMMSetACVSource(int nDmm, double FAR ldVolts, double FAR ldFreq)
```

Remarks This **Secondary** function sets the AC voltage source to RMS amplitude of *ldVolts*, and the frequency to *ldFreq*. The DMM must be in **VAC_SRC** operation for this function to execute properly. When the DMM is in **VAC_SRC** operation, and the **DMMSetACVSource** is applied, reading the DMM **(DMMRead, DMMReadStr**) will return the measurement of the output voltage. This function acts on the main 12 bit source DAC. If better accuracy is needed it can be accomplished by selecting the ClosedLoop mode (**DMMSetSourceMode**). This mode engages the Trim DAC, which augments the 12 bit DAC to produce 16 effective bits. In the ClosedLoop mode, the source level is adjusted any time the DMM is read, making small corrections until the reading is equal to *ldVolts.* However, for the ClosedLoop mode to update the source level, it is necessary to read the DMM multiple times. Update rate should not exceed 5 rps when using the Closed Loop mode. Two ACV measurement ranges are available in VAC_SRC mode, the 3.3 V and the 330 mV. If the Autorange mode is enabled, the DMM will automatically select the appropriate range.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
ldVolts	**double FAR** AC RMS voltage to be set. Range: 0.05 to 7.25 V RMS
ldFreq	**double FAR** DC voltage to be set. Range: 2 Hz to 76 kHz

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
double reading; int I;
DMMSetACVSource(0, 7.0, 1000.0); // source 7V and 1kHz
DMMSetSourceMode(0, CLOSED_LOOP); // Closed loop mode
for(I=0;I<100;I++) DMMRead(0,&reading); // update 100 times
```

DMMSetAutoRange

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Enable/Disable autorange operation of DMM

#include "sm204032.h"

int DMMSetAutoRange(int *nDmm*, **BOOL** *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	**BOOL** Determines whether or not autoranging is done. The value TRUE enables autoranging, FALSE 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 */
```

DMMSetBuffTrigRead

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Setup the DMM for Triggered operation.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetBuffTrigRead(int *nDmm*, **int** *iSettle*, **int** *iSamples*, **int** *iEdge*)

Remarks

Setup the SMX2040 for external hardware trigger operation. Following reception of this command the DMM enters a wait state. After reception of an external trigger edge of *iEdge* polarity, the DMM takes *iSettle* + 1 readings at the set measurement function, range, and reading rate; and stores the last reading in the in an internal buffer. This process is repeated for *iSamples*. This function is particularly useful in conjunction with a triggering instruments such as the SM4042 relay scanner. No autoranging is allowed in this mode. The number of trigger edges must be equal or greater than *iSamples* to properly terminate this mode. Between the time the **DMMSetBuffTrigRead** is issued and the time the buffer is read, no other command should be sent to the DMM. One exception is the **DMMDisarmTrigger** command. This function is usable for VDC, VAC, Ohms, IAC, IDC and RTD measurements.

Use the **DMMReady** to monitor when the DMM is ready (following trigger(s) and the reading of *iSamples*). When ready, you can read up to *iSamples*, using **DMMReadBuffer** or **DMMReadBufferStr** functions. Once **DMMReady** returns **TRUE**, it should not be used again prior to reading the buffer, since it prepares the buffer for reading when it detects a ready condition.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iSettle	**int** The number of setteling measurements, prior to read value. Must be set between 0 and 120. Recommanded value is 4.
iSamples	**int** The number of samples the DMM takes following the same number of trigger pulses. This number must be between 1 and 64, inclusive.
iEdge	**Int** The edge polarity of the trigger signal. 1 for Positive, or leading edge, and 0 for negative or trailing edge trigger.

Return Value

The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Operation successfully terminated
Negative Value	Error code.

Example

```
double Buffer[64];
DMMSetBuffTrigRead(0, 4, 64, 0); // Negative edge, 4
//setteling readings, and 64 samples/triggers
while( ! DMMReady(0) );       // wait for completion
for(i=0; i < 64 ; i++)        // read buffer
      DMMReadBuffer(0, &Buffer[i]);
```

DMMSetCapsAveSamp

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description

Tunes the capacitance measurement function parameters for higher measurement speed.

#include "sm204032.h"

Int DMMSetCapsAveSamp(int *nDmm*, **int** *iAverage*, **int** *iSamples*)

Remarks

This function should be used carefully since it modifies the capacitance function basic measurement parameters; the averages value, *iAverage*, and the number of points sampled, *iSamples*. This function is provided only for cases where it is necessary to

improve measurement speed. When using this function keep in mind that the accuracy specification provided for capacitance is not guaranteed. Also, modifying these values could have profound efect on the operation of the function. Any time a capacitance range is change, these values are set to the default values. For instance, values of 1 and 3 for *iAverage*, and *iSamples* will reduce measurement time on the 12nF range from 0.8s to about 50ms.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iAverage	**int** The average value, must be set between 1 and 100.
iSamples	**int** The number of samples must be set to at least 3.

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

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example
```
int status = DMMSetCapsAveSamp(0,1,3);
```

DMMSetCJTemp

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Set cold junction temperature for thermocouple measurement.

#include "sm204032.h"

int DMMSetCJTemp(int *nDmm***, double** *dTemp***)**

Remarks Set the cold junction temperature for subsequent thermocouple measurements. When measuring temperature using thermocouples it is necessary to establish a reference or cold junction temperature. This is the temperature at which the thermocouple wires are connected to the DMM or to the switching card's cooper wires. One way to do this is by simply entering this value using **DMMSetCJTemp()** function. *dTemp* must be entered using the currently set temperature units.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
dTemp	**double** The cold junction temperature. Must be set between 0°C and 50°C or the corresponding °F.

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

Value	Meaning
DMM_OKAY	Operation successfully terminated
Negative Value	Error code.

Example
```
DMMSetCJTemp(0, 22.5);
```

DMMSetCompThreshold

SMX2040 ☐ SMX2044 ☑

Description Set the Threshold DAC level.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetCompThreshold(int *nDmm***, double FAR** *ldThreshold***)**

Remarks This **Secondary** function sets the output of the Threshold DAC. To use this function, the DMM must be in AC volts. This function sets the detection threshold of the AC comparator. It is compared by the comparator to the AC coupled input voltage. This function is associated with the following functions: Totalizer, Frequency counter, Period, Pulse width and Duty Cycle measurements. *ldThreshold* range is determined by the selected ACV range. For instance, when the 250 V AC range is selected, the allowed range of *ldThreshold* is –500 V to +500 V. See the specification section for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
ldThreshold	**double FAR** DC voltage to be set. Allowed range depends on selected ACV range.

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetCompThreshold(0,28.5); // Set comp. threshold to 28.5V
```

DMMSetCounterRng

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Set the frequency counter to a specific range.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetCounterRng(int *nDmm***, int** *frange***)**

Remarks This function forces the auto-ranging frequency counter to a specific range, *frange*. Use this function if the approximate frequency to be measured is known. It will eliminate the time necessary for the counter to autorange to the appropriate range. Saves time by removing the requirement to make multiple frequency measurements in order to allow the counter to range. All ranges are defind in ***UserDMM.h*** file.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
frange	**int** The range to be set is a value between 0 and 7. See ***UserDMM.h***

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetCounterRng(0, COUNTR_320HZ); // Set counter to measure a
frequency between 65Hz to 320Hz
```

DMMSetDCISource

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the DCI source output level.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetDCISource(int *nDmm***, double FAR** *ldAmps***)**

Remarks This **Secondary** function sets the DC current source to *ldAmps*. The DMM must be in **IDC_SRC** for this function to execute properly. Further, the appropriate DCI range must be selected. When the DMM is in **IDC_SRC** operation, and the **DMMSetDCISource** is applied, reading the DMM **(DMMRead** or **DMMReadStr**) will return the output voltage measurement. This function acts on the main 12 bit source DAC. If better resolution is needed it can be accomplished by setting the Trim DAC by using the **DMMSetTrimDAC** function. There are five current source ranges. The DMM reads the output (load) voltage.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
ldAmps	**double FAR** DC current to be set. Can be 0 to 1.25 X selected range

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetRange(0, _1uA)          // Select 1uA source range
DMMSetDCISource(0, 1.1e-6);   // Set source to 1.1uA
```

DMMSetDCVSource

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the DCV source output level.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetDCVSource(int *nDmm***, double FAR** *ldVolts***)**

Remarks This **Secondary** function sets the DC voltage source to *ldVolts*. The DMM must be in **VDC_SRC** for this function to execute properly. When the DMM is in **VDC_SRC** operation, and the **DMMSetDCVSource** is applied, reading the DMM **(DMMRead** or **DMMReadStr**) will return the measurement of the output voltage. This function acts on the main 12 bit source DAC. If better accuracy is needed it can be accomplished by selecting the ClosedLoop mode (**DMMSetSourceMode**). This mode engages the Trim DAC, which augments the 12 bit DAC to produce 16 effective bits. In ClosedLoop mode, the source level is adjusted every time the DMM is read, making small corrections until the reading is equal to *ldVolts.* However, for the ClosedLoop mode to update the source level, it is necessary to read the DMM multiple times. Update rate should not exceed 10 rps when using the Closed Loop mode. The DMM reads voltages using the 33 V range.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
ldVolts	**double FAR** DC voltage to be set. Can be –10.5 to 10.5 V

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
double reading; int I;
DMMSetDCVSource(0, 1.25); // Set source to 1.25V
DMMSetSourceMode(0, CLOSED_LOOP); // Closed loop mode
for(I=0;I<100;I++) DMMRead(0,&reading); // update 100 times
```

DMMSetExternalShunt

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the value of the leakage function external shunt

#include "sm204032.h"

int DMMSetExternalShunt(int *nDmm,* **double** *ldShunt***)**

Remarks This **Secondary** function sets the value of the external shunt resistor being used. The shunt value is utilized in measurement functions such as Leakage and Extended resistance, Synthesized resistance etc. It is available with S/W versions 1.75 or higher. *ldShunt* sets the shunt value, such that 10,000.0 correxponds to a 10kΩ shunt.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
ldShunt	**double** Shunt resistance value. A value greater than zero and smaller the 200e6 (200 Mega Ohms) is allowed.

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.

Negative Value Error code

Example DMMSetExternalShunt(0, 100000.0); // Set shunt to 100kΩ

DMMSetFuncRange

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Set the DMM function and range.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetFuncRange(int *nDmm,* **int** *nFuncRnge*)

Remarks This function sets both, the function and range used by the DMM. The table of values is defined as *VDC*_330mV, *VAC*_3.3V, *IDC*_330mA, OHM_4W_330K etc. definitions in the header files.

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

Example

```
status = DMMSetFuncRange(0, VDC_3V);
```

DMMSetFunction

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Set the DMM function.

#include "sm204032.h"
#include "UserDMM.h"

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

Remarks This function sets the function used by the DMM. The table of values is defined by the *VDC*, *VAC*, *IDC*, *IAC*, *OHMS2W*, *OHMS4W* … definitions in the DLL header file. Not all functions are available for all DMM types. For instance the SMX2044 has Capacitance while the SMX2040 does not.

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, INDUCTANCE);
```

DMMSetInductFreq

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the frequency of the Inductance Source.

#include "sm204032.h"

int DMMSetInductFreq(int *nDmm***, double FAR** *lpdFreq***)**

Remarks This function sets the frequency of the Inductance measurement source. The value of the frequency should be between 20 Hz and 75 kHz. This function overrides the default frequency for each of the inductance ranges. Therefore, setting a new Inductance measurement range changes the frequency. Use this function after setting the range.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpdFreq	**double FAR** Frequency to be set.

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
int status = DMMSetInductFreq(0, 10e3); // Set source to 10kHz
```

DMMSetOffsetOhms

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Enable/Disable Offset Ohms operation

#include "sm204032.h"

int DMMSetOffsetOhms(int *nDmm*, **BOOL** *bState*)

Remarks This function enables or disables the Offset Ohms compensation function. The default value is FALSE, or no Offset Ohms compensation. When TRUE the measurement rate is about $1/10^{th}$ the set value. When enabling this function with the SM2042, a relay is used to perform it and therefore it will click while measuring.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
bState	**BOOL** Determines whether or not Offset Ohms is enabled. The value TRUE enables, FALSE 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 = DMMSetOffsetOhms(0, TRUE); // enable OffsetOhms
```

DMMSetRange

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Set the DMM range for the present function.

#include "sm204032.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 *_330mV, _3mA, etc.* definitions in the DLL header file. Not all ranges are available for all DMM types. For instance the SMX2044 has a 33 Ohms range, and the SMX2040 does 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, _330mA);
```

DMMSetRate

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Set the DMM reading rate.

#include "sm204032.h"

int DMMSetRate(int *nDmm*, **int** *nRate*)

Remarks

This function sets the reading rate used by the DMM. The table of values is defined by the RATE_ values in the header file. It is important to note that the actual range may be effected by the speed of the CPU as well as other processes running in the background, consuming CPU resources. In order to improve the DMM's measurement rate you may need to do one of the following. Stop or lower the thread priority of competing processes, and/or raise the thread priority of the DMM.

Thread Priorities can be manipulated manually, or in a programming environment. manually: In the Windows Task Manager, select the "Processes" tab, then right click on the process and select "Set Priority". In a programming envirmonment: Windows provides an API for managing the priority of different threads. For instance, in C/C++, the functions SetPriorityClass() and SetThreadPriority() can be used to manage the thread priority

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

Return Value

The return value is one of the following constants.

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code
DMM_E_RATE	Invalid DMM reading rate.

Example

```
status = DMMSetRate(0, RATE_0P1);   // Set to 0.1rps
```

DMMSetRelative

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

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

#include "sm204032.h"

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

Remarks This function selects relative or absolute reading mode for the DMM. If the *bRelative* parameter value is TRUE, the DMM will change to relative reading mode. If FALSE, the DMM will change to absolute reading mode. Caution: Do not select **DMMSetRelative** when in the autorange mode.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
bRelative	**BOOL** TRUE to enter relative mode, FALSE 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);
```

DMMSetResistance

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the resistance value to be synthesized

#include "sm204032.h"

int DMMSetResistance(int *nDmm*, **double** *ldResistance***)**

Remarks This function sets the value of the resistance to be synthesized. The DMM must be in Synthesized Resistance function for this function to be usable. The currently set external shunt resistor value effects the Synthesized Resistance operation. The *ldResistance* value must be between 10.0 to 0e6 (10MΩ). It is available with S/W versions 1.71 or higher.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
ldResistance	**double** Resistance value to be synthesized. Value from 10 to 10e6.

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetResistance(0, 10000.0); // Synthesize 10kΩ
```

DMMSetRTD

SMX2040 ☐ SMX2040 ☑ SMX2044 ☑

Description Set the RTD parameters.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetRTD(int *nDmm***, int** *iWires***, double FAR** *ldRo***)**

Remarks This **Secondary** function sets the RTD parameters. The DMM must be in **RTD** measurement function for this function to execute properly. *iWires* selects between 3-wire and 4-wire RTD (3-wire RTDs are not implemented in this version of software). *ldRo* sets the RTD R_o (Ice point resistance). This function must follow the selection of the basic RTD type, using **DMMSetRange**, since it modifies the default R_o parameter for the selected RTD.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iWires	**int** RTD's number of connecting wires RTD_4_W or RTD_3_W
ldRo	**double FAR** R_o resistance. See specs for allowed range for each RTD type.

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetFunction(0, RTD);        // RTD measurement function
DMMSetRange(0, 1 _pt385);      // Select RTD
DMMSetRTD(0, RTD_4_W, 1000.0); // Set Ro = 1k Ohms
```

DMMSetSensoreParams

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description

Set the cold junction temperature sensor equation parameters.

#include "sm204032.h"

int DMMSetSensorParams(int *nDmm***, double** *lda*, **double** *ldm*, ***double*** *ldb***)**

Remarks

Set the cold junction temperature sensor's equation parameters. Where the temperature of the cold junction equals to (Vcjs - *lda*) / *ldm* + *ldb*, where Vcjs is the cold junction sensor output. This function is used to calculate the cold junction temperature by converting the sensor voltage to temperature. For more information read about **DMMReadCJTemp().**

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lda	**double** the 'a' parameter.
ldm	**double** the 'm' parameter.
ldb	**double** the 'b' parameter.

Return Value

Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetSensorParams(0, 0.558, -0.002, 22.0);// set parameters
```

DMMSetSourceMode

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description

Set the DCV and ACV sources to ClosedLoop, or OpenLoop mode.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetSourceMode(int *nDmm***, int** *iMode***)**

Remarks This **Secondary** function sets the DC and AC voltage sources to either **OPEN_LOOP** or **CLOSED_LOOP**. In **CLOSED_LOOP** the sources use the main 12 bit source DAC. In **CLOSED_LOOP** the Trim DAC is also used, which augments the 12 bit DAC to produce 16 effective bits. Open loop updates are very quick. In ClosedLoop mode the source level is adjusted every time the DMM is read, making small corrections until the reading is equal to the set voltage. However, for the ClosedLoop mode to update the source level, it is necessary to read the DMM multiple times. See **DMMSetDCVSource** and **DMMSetACVSource** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iMode	**int** Source adjustment mode: CLOSED_LOOP or OPEN_LOOP

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetSourceMode(0, CLOSED_LOOP); // Select closed loop mode
```

DMMSetSynchronized

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Enable or disable Synchroneous operation of the DMM.

#include "sm204032.h"

int DMMSetSynchronized(int *nDmm*, **BOOL** *bSync*)

Remarks This function enables or disables the Synchronized operation of the DMM. Default operation is non-synchronized. Select the Synchronized mode when it is necessary to settle full scale input transitions from one reading to the next, and maintain the accuracy of the DMM. This is appropriate for VDC, Ohms, Leakage, DCI, Diode and Guarded Ohms. The result of the synchronized mode is a reduced measurement rate. To run synchronized, reading rate must be set to 10 rps or higher.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
bSync	**BOOL** Determines whether or not synchronized operation is enabled. TRUE enables and FALSE disables synchronization. The default is FALSE.

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

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example `int status = DMMSetSynchronized(0, FALSE); // Cancell sync.`

DMMSetTCType

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Set Thermocouple type.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetTCType(int *nDmm*, **int** *iType*)

Remarks This function selects the thermocouple type to be measured and linearized. It must be one of the following: B, E, J, K, N, R, S or T.

Parameter	Type/Description
NDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iTempUnits	**int** The thermocouple type to be selected. This value can be set from BTyppe to TType as defined in the UserDMM.H file.

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

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example `int status = DMMSetTCType(0, NType) // select N type TC`

DMMSetTempUnits

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Set temperature units to °C or °F.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetTempUnits(int *nDmm*, **int** *iTempUnits*)

Remarks This function sets the temperature units to either °C or °F. This is applicable to both the on-board temperature sensor and the RTD measurements.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iTempUnits	**int** Temperature units can be either DEG_F for °F, or DEG_C for °C. The default is °C.

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

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example

```
int status = DMMSetTempUnits(0, DEG_F) // set units to °F
```

DMMSetTrigRead

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Setup the DMM for mutiple Triggered readings operation.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetTrigRead(int *nDmm***, int** *iSettle***, int** *iSamples***, int** *iEdge***)**

Remarks Setup the SMX2040 for external hardware trigger operation. Following reception of this command the DMM enters a wait state. After reception of an external trigger edge of *iEdge* polarity, the DMM takes *iSettle* + 1 readings at the set measurement function, range, and reading rate; and sends the last reading. This process is repeated for *iSamples*. *iSamples* Trigger pulses must be issued to complete this process. This function is particularly useful in conjunction with a triggering instruments such as the SM4042 relay scanner. No autoranging is allowed in this mode. The number of trigger edges must be equal or greater than *iSamples* to properly terminate this mode. Following the issue of the **DMMSetTrigRead** command, and until the sampling process ends, it is necessary to read the samples from the DMM using the **DMMReadMeasurement** command. This will prevent an Overrun communication error, whereby the rate at which measurement are read does not keep up with the DMM transmission. The DMM has a built in 5 readings fifo to reduce this problem. This function is usable for VDC, VAC, Ohms, IAC, IDC and RTD measurements.

Use the **DMMReadMeasurement** to both, monitor when the DMM has new data, and to read this data. Read as many samples as *iSamples* to guarantee proper termination of this capture process.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iSettle	**int** The number of setteling measurements, prior to read value. Must be set between 0 and 120. Recommanded value is 4.
iSamples	**int** The number of samples the DMM takes following the same number of trigger pulses. This number must be between 1 and 250, inclusive.
iEdge	**Int** The edge polarity of the trigger signal. 1 for Positive, or leading edge, and 0 for negative or trailing edge trigger.

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

Value	Meaning
DMM_OKAY	Operation successfully terminated

Negative Value Error code.

Example

```
double Reading[150];
DMMSetTrigRead(0, 4, 150, 0); // Negative edge, 4
//setteling readings, and 150 samples/triggers
for(i=0; i < 150 ; i++)       // read buffer
      while( ! DMMReadMeasurement(0 , Reading[i]) );
```

DMMSetTrimDAC

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Set the Trim DAC level.

#include "sm204032.h"
#include "UserDMM.h"

int DMMSetTrimDAC(int *nDmm***, int** *iValue***)**

Remarks This **Secondary** function sets the Trim DAC to a value between 0 and 100. The trim DAC can be set to augment the main 12 bit DAC, whenever it is not automatically performed, such as in VDC and VAC source while **OPEN_LOOP** mode is selected. An example would be in DCI source, or when setting the Comparator Threshold. This function consumes a lot of the on-board microcontroller's resources and must be turned off when not in use. Use **DMMDisableTrimDAC** to turn off. With the Trim DAC the effective resolution of the composite DAC is increased to 16 bits. With *iValue* set to 100, the Trim DAC adds slightly over 1 LSB of the 12 bit DAC. See **DMMSetDCVSource** and **DMMSetACVSource** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iValue	**int** Amplitude can be set from 0 to 100, corresponding to 0% to 100% Trim DAC level.

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetDCVSource(0, 5.0); // Set source to 5V
DMMSetTrimDAC(0, 50);    // add about 2.5mV to output
```

DMMStartTotalizer

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Clear the totalized value and start the totalizer.

#include "UserDMM.h"
#include "sm204032.h"

int DMMStartTotalizer(**int** *nDmm*, **int** *Edge*)

Remarks

This is a **Secondary** function and the DMM must be in ACV measurement function, and a valid range must be selected. This function clears the Totalized count, sets the edge sense, and starts the Totalizer. The totalized value can be read during the accumulation period. However, it could affect the count by the interruption. If no reads are performed during accumulation, the input rate can be as high as 45 kHz. If reads are performed during the accumulation period, this rate could be as low as 20 kHz. The Threshold DAC sets the level at which signals are counted. During accumulation, no other command (except **DMMReadTotalizer**) should be used. When done, this function must be turned off using **DMMStopTotalizer**. After the Totalizer is stopped, the accumulated result can be read using **DMMReadTotalizer**. A normal procedure would be to set the DMM to the ACV function, select voltage range, set the Threshold DAC, start the totalizer, wait for the time required, stop and read the total. The total number of events is limited to 1,000,000,000.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
Edge	**int** Identifies the edge of the counter. If TRAILING (0) count negative edges, if LEADING (1) count positive edges

Return Value

Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
int status = DMMStartTotalizer(0, LEADING);
```

DMMStopTotalizer

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Terminate the accumulation process of the Totalizer.

#include "sm204032.h"

int DMMStopTotalizer(int *nDmm*)

Remarks This function stops the accumulation process. Following this function, the totalized value can be read. For details see **DMMStartTotalizer**.

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	Operation was successful.
Negative Value	Error code

Example

```
int status = DMMStopTotalizer(0);
```

DMMTerminate

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Terminate DMM operation (DLL)

#include "sm204032.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 */
```

DMMTrigger

SMX2040 ☑ SMX2042 ☑ SMX2044 ☑

Description Software Trigger the DMM. Take *iSamples*.

#include "sm204032.h"

int DMMTrigger(**int** *nDmm*, **int** *iSamples*)

Remarks Following reception of this command, the SMX2040 DMM makes *iSamples* readings at the currently set function, range and rate, and stores them in an internal buffer. Rate can be set between 10 to 1000 readings per second. No autoranging is allowed for this trigger operation. Between the time the **DMMTrigger** command is issued and the time the buffer is read, no other command should be sent to the DMM. Use the **DMMReady** function to monitor when the DMM is ready (ready implies completion of *iSamples*). When ready, you can optionally read a single reading or up to *iSamples*, using **DMMReadBuffer.**

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
iSamples	**int** The number of samples the DMM takes following a trigger pulse. This number must be between 1 and 64, inclusive.

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

Value	Meaning
DMM_OKAY	Operation successfully terminated.
DMM_E_INIT	DMM is uninitialized. Must be initialize prior to using any function.
DMM_TRIG_N	Measurement count is out of allowed range.
DMM_E_DMM	Invalid DMM number.

Example

```
double Buffer[64];
int state;
DMMTrigger(0,64);
while( ! DMMReady(0));
      for(i=0; i < 64 ; i++)
      state = DMMReadBuffer(0, &Buffer[i]);
```

DMMWidthStr

SMX2040 ☐ SMX2042 ☑ SMX2044 ☑

Description Return positive and negative pulse width in string format.

#include "sm204032.h"

int DMMWidthStr(**int** *nDmm*, **LPSTR** *lpszPos*, **LPSTR** *lpszNeg*)

Remarks This **Secondary** function is the string equivalent of **DMMReadWidth**. The measurement results are stored at the location pointed to by *lpszPos and lpszNeg*. See **DMMReadWidth** for more details.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszPos	**LPSTR** Points to a buffer (at least 16 characters long) to hold the positive width result.
lpszNeg	**LPSTR** Points to a buffer (at least 16 characters long) to hold the negative width result.

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

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example

```
char cBuf[17]; int status = DMMWidthStr(0, cBuf);
```

SetACCapsFreq

SMX2040 ☐ SMX2042 ☐ SMX2044 ☑

Description Sets the frequency of the AC capacitance source.

#include "sm204032.h"

int SetACCapsFreq(int *nDmm*, **LPSTR** *lpszPos*, **doouble** *dFrequency*)

Remarks This service function allows the modification of the AC source frequency. This is only active during in-circuit capacitance test. The frequency may be set between 10Hz and 75kHz.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
dFrequency	**doulbe** The frequency to be set (10 to 75,000).

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

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example

```
SetACCapsFreq(0, 10000.0) // Set the frequency to 10kHz
```

6.0 Maintenance

Warning

These service instructions are for use by qualified personnel only. To avoid electric shock, do not perform any procedures in this section unless you are qualified to do so.

This section presents maintenance information for the DMM.

Test equipment recommended for calibration is listed below. If the recommended equipment is not available, equipment that meets the indicated minimum specifications may be substituted. In general, the calibration equipment should be at least three times more accurate than the DMM specifications.

Table 9-1. Recommended Test Equipment

Instrument Type	Minimum Specifications	Recommended Model
Multi-Function Calibrator	DC Voltage Range: 0-300 V Voltage Accuracy: 9 ppm AC Voltage Range: 0-250 V Voltage Accuracy: 0.014% Resistance Range: 0-330 MΩ Resistance Accuracy: 22 ppm DC Current Range: 0-2.5 A Current Accuracy: 0.008% AC Current Range: 50 uA – 2.5 A Current Accuracy: 0.05% Capacitance Range: 10 ηF – 10 mF Capacitance Accuracy: 0.19%	Fluke 5520A

6.1 Performance Tests

This test compares the performance of the SMX2040/44 with the specifications given in Section 2. The test is recommended as an acceptance test when the instrument is first received, and as a verification after performing the calibration procedure. To ensure proper performance, the test must be performed with the SMX2040 installed, with the covers on. The ambient temperature must be between 18°C to 28°C. Allow the SMX2040 to warm up at least one-half hour before performing any of the tests. The default reading rate of the SMX2040 should be used in each test.

6.2 DC Voltage Test

The following procedure may be used to verify the accuracy of the DCV function:

1. If you have not done so, install the SMX2040. Ensure that the chassis has been on for at least one-half hour, with all covers on, before conducting this test.

2. Apply a high quality copper wire short to the SMX2040 **V,Ω + & -** inputs. Select the DCV function, Autorange. Allow the SMX2040 to settle for several seconds, and perform the **Relative** function.

3. Apply the following DC voltages to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SMX2040 is within the indicated range.

Table 9-2. DC Voltage Test

Step	Range	Input	Minimum Reading	Maximum Reading
1	330 mV	0V (short)	-8 μV	+8 μV
2	330 mV	190 mV	189.9787mV	190.0213 mV
3	330 mV	-190 mV	-190.0213 mV	–189.9787 mV
4	3.3 V	1.9 V	1.899898 V	1.900102 V
5	3.3 V	-1.9 V	-1.900102 V	-1.899898 V
6	33 V	19 V	18.99834 V	19.00166 V
7	33 V	-19 V	-19.00166 V	-18.99834 V
8	330 V	190 V	189.9833 V	190.0167 V
9	330 V	-190 V	-190.0167 V	-189.9833 V

6.3 Resistance Test, 2-wire

The following procedure may be used to verify the accuracy of the 2-wire function.

1. If you have not done so, install the SMX2040 and place the covers back on to the computer. Ensure that the computer has been on for at least one-half hour, with the covers on, before conducting this test.

2. Connect the SMX2040 **V,Ω** + **& -** terminals to the calibrator HI & LO Outputs. Output 0 Ω from the calibrator. Allow the SMX2040 to settle for a few seconds, and perform the **Relative** function. (This effectively nulls out the lead resistance of your cabling. If you are using a Fluke 5700A or 5520A Calibrator, the 2-wire Compensation feature will give a more accurate 2-wire ohms measurement. See the *Fluke Operator's Manual* for further instructions.)

3. Apply the following Resistance values to the **V, Ω + & -** terminals . Check to see that the displayed reading on the SMX2040 is within the indicated range.

Table 9-3 Resistance Test, 2-wire

Step	Range	Input	Minimum Reading	Maximum Reading
1	33 Ω [1]	10 Ω	9.9972 Ω	10.0028 Ω
2	330 Ω	100 Ω	99.987 Ω	100.013 Ω
3	3.3 kΩ	1 kΩ	0.999917 kΩ	1.000083 kΩ
4	33 kΩ	10 kΩ	9.99905 kΩ	10.00095 kΩ
5	330 kΩ	100 kΩ	99.986 kΩ	100.014 kΩ
6	3.3 MΩ	1 MΩ	0.99942 MΩ	1.00058 MΩ
7	33 MΩ	10 MΩ	9.973 MΩ	10.027 MΩ
8	330 MΩ [1]	100 MΩ	97.92 MΩ	102.08 MΩ

[1] SMX2044 only

6.4 Resistance Test, 4-wire

The following procedure may be used to verify the accuracy of the 4-wire function.

1. If you have not done so, install the SMX2040 and place the covers back on to the chassis. Ensure that the computer has been on for at least one-half hour, with the covers on, before conducting this test.

2. Connect the SMX2040 **V,Ω + & -** terminals to the calibrator HI & LO Output. Connect the SMX2040 **I, 4WΩ + & -** terminals to the HI & LO Sense terminals.

3. Select the 4WΩ function on the SMX2040, Autorange. Set the calibrator to 0 Ω. Be certain that the calibrator is set to external sense ("EX SNS" on the Fluke 5700A or “4-Wire Comp” on the 5520A). Allow the SMX2040 to settle for a few seconds, and perform the **Relative** function.

4. Apply the following Resistance values to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SMX2040 is within the indicated range.

Table 9-4 Resistance Test, 4-wire

Step	Range	Input	Minimum Reading	Maximum Reading
1	33 Ω [1]	0 Ω	-2 mΩ	2 mΩ
1	33 Ω [1]	10 Ω	9.9972 Ω	10.0028 Ω
1	330 Ω	0 Ω	-6 mΩ	6 mΩ
2	330 Ω	100 Ω	99.987 Ω	100.013 Ω
3	3.3 kΩ	0 Ω	-33 mΩ	33 mΩ
4	3.3 kΩ	1 kΩ	0.999917 kΩ	1.000083 kΩ
5	33 kΩ	0 Ω	-350 mΩ	350 mΩ
5	33 kΩ	10 kΩ	9.99905 kΩ	10.00095 kΩ
5	330 kΩ	0 Ω	-5 Ω	5 Ω
6	330 kΩ	100 kΩ	99.986 kΩ	100.014 kΩ

[1] SMX2044 only.

Note: The use of 4-wire Ohms for resistance values above 300 kΩ is not recommended.

6.5 AC Voltage Test

The following procedure may be used to verify the accuracy of the ACV function:

1. If you have not done so, install the SMX2040 and place the covers back on to the chassis. Ensure that the computer has been on for at least one-half hour, with the covers on, before conducting this test.

2. Apply the following AC voltages to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SMX2040 is within the indicated readings range.

Table 9-5. Mid-Frequency AC Voltage Tests
All inputs are a sine wave at **400 Hz**.

Step	Range	Input	Minimum Reading	Maximum reading
1	330 mV	10 mV	9.865 mV	10.135 mV
2	330 mV	190 mV	189.595 mV	190.405 mV
4	3.3 V	100 mV	0.098735 V	0.101265 V
5	3.3 V	1.9 V	1.897565 V	1.902435 V
6	33 V	1 V	0.98327 V	1.01673 V
7	33 V	19 V	18.97313 V	19.02687 V
8	250 V	10 V	9.864 V	10.136 V
9	250 V	190 V	189.756 V	190.244 V

Table 9-6. High-Frequency AC Voltage Tests
All inputs are at **50 kHz**.

Step	Range	Input	Minimum Reading	Maximum Reading
1	330 mV	10 mV	9.707 mV	10.293 mV
2	330 mV	190 mV	188.573 mV	191.427 mV
4	3.3 V	100 mV	0.0978 V	0.1022 V
5	3.3 V	1.9 V	1.8852 V	1.9148 V
6	33 V	1 V	0.9715 V	1.0285 V
7	33 V	19 V	18.9085 V	19.0915 V
8	250 V	10 V	9.755 V	10.245 V
9	250 V	100 V	99.35 V	100.65 V

6.6 DC Current Test

The following procedure may be used to verify the accuracy of the DCI function:

1. If you have not done so, install the SMX2040 and place the covers back on to the computer. Ensure that the computer has been on for at least one-half hour, with the covers on, before conducting this test.

2. Remove all connections from the SMX2040 inputs. Select the DCI function, Autorange. Allow the SMX2040 to settle for a few seconds, and perform the **Relative** function.

3. Apply the following DC currents to the **I,4Ω + & -** terminals. Check to see that the displayed reading on the SMX2040 is within the indicated readings range.

Table 9-7. DC Current Test

Step	Range	Input	Minimum Reading	Maximum reading
1	3.3 mA	0 mA (open)	-0.0004 mA	0.0004 mA
2	3.3 mA	1 mA	0.9986 mA	1.0014 mA
3	33 mA	0 mA (open)	-0.003 mA	0.003 mA
4	33 mA	10 mA	9.987 mA	10.013 mA
5	330 mA	0 mA (open)	-0.060 mA	0.060 mA
6	330 mA	100 mA	99.865 mA	100.135 mA
7	2.5 A	0 A	-0.00035 A	0.00035 A
8	2.5 A	1 A	0.99315 A	1.00685 A

6.7 AC Current Test

The following procedure may be used to verify the accuracy of the ACI function:

1. If you have not done so, install the SMX2040 and place the covers back on to the computer. Ensure that the computer has been on for at least one-half hour, with the covers on, before conducting this test.

2. Remove all connections from the SMX2040 inputs. Select the ACI function, Autorange.

3. Apply the following AC currents to the **I,4Ω + & -** terminals. Check to see that the displayed reading on the SMX2040 is within the indicated readings range.

Table 9-8. AC Current Test
All Inputs are at **400Hz**

Step	Range	Input	Minimum Reading	Maximum reading
1	3.3 mA	0.1 mA	0.09588 mA	0.100412 mA
2	3.3 mA	1 mA	0.9948 mA	1.0052 mA
3	33 mA	1 mA	0.9684 mA	1.0316 mA
4	33 mA	10 mA	9.954 mA	10.046 mA
5	330 mA	10 mA	9.758 mA	10.242 mA
6	330 mA	100 mA	99.56 mA	100.44 mA
7	2.5 A	100 mA	0.09535 A	0.10465 A
8	2.5 A	1 A	0.9895 A	1.0105 A

6.8 Capacitance Test (SMX2044 only)

The following procedure may be used to verify the accuracy of the Capacitance function.

1. If you have not done so, install the DMM and place the covers back on to the computer. Ensure that the computer has been on for at least one-half hour, with the covers on, before conducting this test.

2. Connect the DMM **V,Ω** + **& -** terminals to the calibrator HI & LO Outputs. Attach the test leads to the DMM, leaving the other end open circuited. Allow the DMM to settle for a few seconds, and perform the **Relative** function. (This effectively nulls out the lead capacitance of your cabling.

3. Apply the following Capacitance values to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SMX2044 is within the indicated range of readings.

Step	Range	Input	Minimum Reading	Maximum reading
1	10 ηF	10 ηF	9.785 ηF	10.215 ηF
2	100 ηF	100 ηF	99 ηF	101 ηF
3	1 μF	1 μF	0.99 μF	1.01 μF
4	10 μF	10 μF	9.9 μF	10.1 μF
5	100 μF	100 μF	99 μF	101 μF
6	1 mF	1 mF	0.988 mF	1.012 mF
7	10 mF	10 mF	9.8 mF	10.2 mF

6.9 Frequency Counter Test (SMX2044 only)

The following procedure may be used to verify the accuracy of the Frequency Counter:

1. If you have not done so, install the DMM and place the covers back on to the computer. Ensure that the computer has been on for at least one-half hour, with the covers on, before conducting this test.

2. Select the ACV function, autorange. Turn **freq** on.

3. Apply the following AC voltages to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SMX2044 is within the indicated range of readings.

Table 9-9. ACV Frequency Counter Test

Step	Range	Input	Minimum Reading	Maximum reading
1	330 mV	33 mV, 40 Hz	39.9952 Hz	40.0048 Hz
2	3.3 V	330 mV, 40 Hz	39.9952 Hz	40.0048 Hz
3	33 V	3.3 V, 40 Hz	39.9952 Hz	40.0048 Hz
4	330 V	33 V, 40 Hz	39.9952 Hz	40.0048 Hz
5	330 mV	250 mV, 100 kHz	99.996 kHz	100.004 kHz
6	33 V	25 V, 100 kHz	99.996 kHz	100.004 kHz

2. Select the ACI function, autorange. Turn **freq** on.

3. Apply the following AC currents to the **I,4Ω + & -** terminals. Check to see that the displayed reading on the SMX2040 is within the indicated range of readings.

Table 9-10. ACI Frequency Counter Test

Step	Range	Input	Counter Reading	Tolerance
1	3.3 mA	330 uA, 40 Hz		
2	33 mA	15 mA, 40 Hz		
3	330 mA	150 mA, 40 Hz		

6.10 Calibration

Each SMX2040 DMM uses its own **SM40CAL.DAT** calibration file to ensure the accuracy of its functions and ranges. The **SM40CAL.DAT** file is a text file that contains the DMM identification number, calibration date, and calibration constants for all DMM ranges. For most functions, the calibration constants are scale factor and offset terms that solve the "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 SMX2040 family, the calibration record contains a place-holder. An example **SM40CAL.DAT** is shown:

```
card_id  10123    type 2044 calibration_date 06/15/1999
ad       ; A/D compensation
72.0     20.0
vdc      ; VDC 330mV, 3.3V, 33V, 330V ranges, offset and gain parameters
-386.0   0.99961
-37.0    0.999991
-83.0    0.999795
-8.8     1.00015
vac      ; VAC 1st line - DC offset. Than offset, gain and freq each range 330mV to 250V
5.303
0.84     1.015461          23
0.0043   1.0256            23
0.0      1.02205           0
0.0      1.031386          0
idc      ; IDC 3.3mA to 2.5A ranges, offset and gain
-1450.0  1.00103
-176.0   1.00602
-1450.0  1.00482
-176.0   1.00001
iac      ; IAC 3.3mA 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 33, 330, 3.3k,33k,330k,3.3M,33M,330Meg ranges, offset and gain
1.27e+4  1.002259
1256.0   1.002307
110.0    1.002665
0.0      1.006304
0.0      1.003066
0.0      1.001848
0.0      0.995664
0.0      1.00030
```

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 33 Ω range, and "`1.002259`" is the scale factor for this range. This record must be for the SMX2044 since the SMX2040 does not have the 33 Ohms range, and therefore these values will be set to 0.0 and 1.0.

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 **SM40CAL.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 SMX2040 is best performed using calibration software available from Signametrics.

When using multiple DMMs in a single chassis, the **SM40CAL.DAT** file must have a calibration record for each DMM. You can combine the unique calibration records of each DMM into one **SM40CAL.DAT** file using any ASCII text editor.

7.0 Warranty and Service

The SMX2040 is warranted for a period of one year from date of purchase.

If your unit requires repair or calibration, contact your Signametrics representative. There are no user serviceable parts within the SMX2040. Removal of any of the three external shields will invalidate your warranty. For in-warranty repairs, you must obtain a return authorization from Signametrics prior to returning your unit.

8.0 Accessories

Several accessories are available for the SMX2040 DMMs, 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

`