ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Keysight / Signametrics SM2010* at our website: ***Click HERE***

Operator's Manual

Model SM-2010
4½ Digit Digital Multimeter

Model SM-2010CT
4½ Digit Digital Multimeter With Frequency Counter

Signametrics Corporation
Precision Instruments for the PC

6073 50th Avenue N.E.
Seattle, WA 98115
FAX (206) 525-8578
Phone (206) 524-4074

June 1998

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.

© 1997 Signametrics Corp. Printed in the USA. All rights reserved. Contents of this publication must not be reproduced in any form without the permission of Signametrics Corporation.

TABLE OF CONTENTS

1.0 INTRODUCTION 5

1.1 SAFETY CONSIDERATIONS 5
1.2 MINIMUM REQUIREMENTS 5

2.0 SPECIFICATIONS, SM-2010 AND SM-2010CT 6

2.1 DC VOLTAGE 6
2.2 RESISTANCE 6
2.3 AC VOLTAGE, TRUE RMS 7
2.4 DC CURRENT 7
2.5 AC CURRENT, TRUE AC+DC RMS 8
2.6 ACCURACY NOTES 9
2.7 OTHER SPECIFICATIONS 10
2.8 SM-2010CT SPECIFICATIONS 11

3.0 GETTING STARTED 12

3.1 SETTING THE SM-2010 ADDRESS 12
3.2 INSTALLING THE SM-2010 BOARD 14
3.3 INSTALLING THE SOFTWARE 14
3.4 STARTING THE VISUAL BASIC CONTROL PANEL 14

4.0 SM-2010 INPUT CONNECTORS 15

5.0 MEASUREMENT FUNCTIONS 16

6.0 MAKING MEASUREMENTS 18

6.1 VOLTAGE MEASUREMENTS 18
6.2 DC AND AC CURRENT MEASUREMENTS 19
6.3 RESISTANCE MEASUREMENTS 19

7.0 SM-2010 DOS LIBRARY AND INTERFACE 21

7.1 DISTRIBUTION FILES 21
7.2 ADDRESSING THE DMM 22
7.3 USING THE SM-2010 LIBRARY 22
7.4 DOS DEFAULT MODES AND PARAMETERS 22
7.5 DOS COMMAND LANGUAGE 22
CalibrateDmm 23
ClearRelative 23
GetCalDate 24
GetCalLab 24
GetDmmFuncRange 24
GetDmmFunction 25
GetDmmRange 25
GetDmmRate 25
GetFreq (SM-2010CT only) 26
GetFreqStr (SM-2010CT only) 26
GetHwVer 26
GetPeriod (SM-2010CT only) 27
GetPeriodStr (SM-2010CT only) 27
GetPrintFormat 27
GetSerialNumber 28
GetUnits 28
GetVer 28
HasCount (SM-2010CT only) 29
InitDmm 29
IsAutoRange 30

IsRelative 30
ReadDmm, ReadDmmFloat 31
ReadDmmNorm 32
ReadDmmStrg 32
SetAutoRange, ClearAutoRange 33
SetDmmFuncRange 33
SetDmmRate 34
SetRelative 34
ShutDownDmm 34

8.0 SM-2010 DMM WINDOWS INTERFACE 35

8.1 DISTRIBUTION FILES 35
8.2 CONFIGURATION FILE 36
8.3 USING THE SM-2010 WINDOWS DRIVER 37
8.4 VISUAL BASIC CONTROL PANEL APPLICATION 37
8.5 WINDOWS DLL DEFAULT MODES AND PARAMETERS 38
8.6 USING THE SM2010 DLL WITH LABWINDOWS/CVI® VERSION 3.1 38
8.7 WINDOWS COMMAND LANGUAGE 41
DMMCalibrate 41
DMMErrString 41
DMMGetCalDate 42
DMMGetCalLab 42
DMMGetFnRange 43
DMMGetFunction 44
DMMGetHwVer 44
DMMGetRange 45
DMMGetRate 45
DMMGetSerialNumber 46
DMMGetVer 46
DMMHasCountOption 47
DMMInit 47
DMMIsAutoRange 48
DMMIsRelative 48
DMMRead, DMMReadDbl 49
DMMReadingStr 49
DMMReadFrequency (SM-2010CT only) 50
DMMReadFrequencyStr (SM-2010CT only) 50
DMMReadNorm 51
DMMReadPeriod (SM-2010CT only) 51
DMMReadPeriodStr (SM-2010CT only) 52
DMMSetAutoRange 52
DMMSetFnRange 53
DMMSetRate 53
DMMSetRelative 54
DMMShutDown 54

9.0 MAINTENANCE 55

9.1 PERFORMANCE TESTS 56
9.2 DC VOLTAGE TEST 56
9.3 RESISTANCE TEST 57
9.4 AC VOLTAGE TEST 58
9.5 DC CURRENT TEST 59
9.6 AC CURRENT TEST 59
9.7 FREQUENCY COUNTER TEST (SM-2010CT ONLY) 60
9.8 CALIBRATION 61

10.0 WARRANTY AND SERVICE 61

1.0 Introduction

Congratulations! You have purchased a Personal Computer (PC) Plug-in instrument with analog and systems performance that rivals the best, all-in-one box, instruments. The SM-2010 series DMMs are easy to setup and use, have sophisticated analog and digital circuitry to give 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.

Note: This manual describes both the SM-2010 and SM-2010CT DMMs. The SM-2010CT is identical to the SM-2010 except for the addition of a Frequency Counter. All references to the "SM-2010" also apply to the SM-2010CT. Features unique to the SM-2010CT will be identified as so.

1.1 Safety Considerations

Safety Considerations

The SM-2010 is capable of measuring up to 250VDC or 250VAC across the Volt HI and LO terminals, and can also measure common mode signals that "float" the SM-2010 above EARTH ground by up to 250VDC or 250VAC. When making common mode measurements, the majority of the circuits inside the SM-2010 are at the common mode voltage. **These voltages can be lethal and can KILL! After installing your SM-2010, check to see that there are no wires or ribbon cables from your PC trapped inside the SM-2010.**

The SM-2010 comes installed with a fully enclosed, four-sided shield that must **not be removed for performance as well as safety reasons.** Removal of these shields and/or improper assembly of the shields can result in lethal voltages occurring within your PC. Be sure to check your installation before closing the cover on your personal computer.

Warning

Check to see that no loose wires or ribbon cables infringe upon any of the internal circuits of the SM-2010, as this may apply measurement voltages to your computer!

To avoid shock hazard, install the SM-2010 only into a computer that has its power connector connected to a power receptacle with an earth safety ground.

When making any measurements above 50 VDC or 40 VAC, only use Safety Test Leads. An example of these are the ITT Pomona model 5901 Bench Test Kit, or the ITT Pomona model 5567A DMM Test Lead Kit. ITT Pomona Electronics can be reached at (909) 469-2900. (These test leads are also available from many electronic distributors. They are often referred to as "Fluke Safety DMM leads".)

1.2 Minimum Requirements

The SM-2010 series of system DMMs are precision plug-in boards that are compatible with IBM type personal computers (PCs), from the 286 AT class to the Pentium Pro. They require a one half length expansion slot on the 16 bit ISA bus. You will need a minimum of 512k memory and Windows 3.1 or higher. There are no special graphics requirements.

2.0 Specifications, SM-2010 and SM-2010CT

2.1 DC Voltage

- **Input Resistance:** > 1000 MΩ on 250 mV and 2.5V ranges. 10 MΩ on 25V and 250V ranges.
- **Settling Time:** 1 second to 2 counts of final reading on range
- **DCV Noise Rejection:** Normal Mode Rejection, at 50, 60, or 400 Hz ± 0.5%, is better than 80 dB for readings of 10 or slower. Common Mode Rejection (with 1 kΩ lead imbalance) is better than 120 dB for these same conditions.

DC Voltage ± (% of reading + Number of Counts)

Range	= Full Scale 4 1/2 Digits	Resolution	Accuracy 18°C to 28°C, One Year
250 mV	250.00 mV	10 µV	0.03 + 2
2.5 V	2.5000 V	100 µV	0.03 + 1
25 V	25.000 V	1 mV	0.03 + 1
250 V	250.00 V	10 mV	0.03 + 1

2.2 Resistance

- **Maximum Allowable Input:** 250V DC or rms AC.
- **Open Circuit Voltage:** < 5 V on all ranges.
- **Settling Time:** 2 seconds to within 1 count of final reading on range.

Resistance = (% of reading + Number of Counts)

Range	Full Scale 4 1/2 Digits	Resolution	Accuracy 18°C to 28°C, One Year
250 Ω	250.00 Ω	10 mΩ	0.03 + 2 [1]
2.5 kΩ	2.5000 kΩ	100 mΩ	0.03 + 2 [1]
25 kΩ	25.000 kΩ	1 Ω	0.03 + 2
250 kΩ	250.00 kΩ	10 Ω	0.05 + 2
2.5 MΩ	2.5000 MΩ	100 Ω	0.2 + 2
25 MΩ	25.000 MΩ	1 kΩ	1.0 + 2

[1] Using Relative control.

The SM-2010 measures resistance by sourcing a constant current, and then measuring the resultant voltage across the resistor. The value of this constant current is shown in the table below.

Resistance Range	Current Level
250 Ω	1 mA
2.5 kΩ	100 uA
25 kΩ	100 uA
250 kΩ	1 uA
2.5 MΩ	1 uA
25 MΩ	100 nA

2.3 AC Voltage, True RMS

- **Input Impedance:** 1 MΩ in parallel with < 75 pF.
- **Crest Factor:** 1.6 at Full Scale, increasing to 7 at Lowest Specified Voltage
- **Settling Time:** 2 second to 20 counts of final reading on range.
- **ACV Noise Rejection:** Common Mode rejection, for 50 Hz or 60 Hz with 1 kΩ imbalance in either lead is better than 60 dB.

Range	**Full Scale 4 1/2 Digits**	**Lowest Specified Voltage**	**Resolution**
250 mV	250.00 mV	20.00 mV	10 uV
2.5V	2.5000 V	100 mV	100 uV
25V	25.000 V	1.000 V	1 mV
250V [2]	250.00 V	10.00 V	10 mV

[2] Limited to $8x10^6$ Volt x Hz.

AC Voltage, True RMS ± (% of reading + Number of Counts)

Range	Full Scale 4 1/2 Digits	Frequency	**Accuracy 18°C to 28°C, One Year [3]**
250 mV	250.00 mV	10 - 45 Hz	3.0 – 40
		45 - 5 kHz	0.3 – 35
		5 kHz - 20 kHz	0.8 – 35
2.5V - 250V [4]	2.5000 V, 25.000 V, 250.00 V	20 - 45 Hz	3.0 – 30
		45 - 5 kHz	0.3 – 20
		5 kHz - 20 kHz	0.7 – 30

[3] For sinewave inputs. When using the SM-2010CT, accuracies apply when the Frequency Counter is not active.
[4] Limited to $8x10^6$ Volt x Hz. For example, the largest frequency input at 250V is 32 kHz, or $8x10^6$.

2.4 DC Current

- **Burden Voltage** < 0.5 V for the 2.5 mA and 25 mA Range, < 1.2 V for the 250 mA and 2.5A Range
- **Protected** with 2.5A fuse (5x20mm. 250V Medium Time Delay -- Buss® GMC 2.5A or equivalent)

DC Current ± (% of reading + Number of Counts)

Range	**± Full Scale 4 1/2 Digits**	**Resolution**	**Accuracy 18°C to 28°C, One Year**
2.5 mA	2.5000 mA	100 nA	0.4 + 9 [5]
25 mA	25.000 mA	1 uA	0.3 + 2
250 mA	250.00 mA	10 uA	0.3 + 9 [5]
2.5A	2.5000 A	100 uA	0.5 + 2

[5] Within ten minutes of DC Current zero, using Relative control.

2.5 AC Current, True AC+DC RMS

Input Characteristics

- **Burden Voltage** < 0.5 V RMS for the 2.5 mA and 25 mA Range, <1.2 V RMS for the 250 mA and 2.5A Range
- **Crest Factor** 1.6 at Full Scale, increasing to 7 at Lowest Specified Current
- **Protected** with 2.5A fuse (5x20mm, 250V Medium Time Delay – Buss® GMC 2.5A or equivalent)
- **DC coupled** in all ranges, giving the true AC+DC rms value

Range	Full Scale 4 1/2 Digits	Lowest Specified Current	Resolution
2.5 mA	2.5000 mA	.1000 mA	100 nA
25 mA	25.000 mA	1.000 mA	1 uA
250 mA	250.00 mA	10.00 mA	10 uA
2.5A	2.5000A	.1000 mA	100 uA

AC Current, True DC+AC RMS + (% of reading + Number of Counts)

Range	Full Scale 4 1/2 Digits	Frequency	Accuracy 18°C to 28°C, One Year [6]
2.5 mA - 2.5 A	2.5000 mA, 25.000 mA, 250.00 mA, 2.5000 A	10 - 45 Hz	1.0 + 30
		45 - 1 kHz	0.9 + 20
		1 kHz - 5 kHz [7]	1.2 + 30

[6] For sinewave inputs. When using the SM-2010CT, accuracies apply when the Frequency Counter is not active.
[7] Typically to 20 kHz

2.6 Accuracy Notes

System Calibration To meet the accuracies stated, the **AutoCal** function must be performed at least once per month, whenever the DMM is powered OFF/ON, or if the external operating temperature changes by more than + 5°C. To perform the **AutoCal** function, allow the DMM to warm up for one-half hour. Remove all input connections to the DMM, and simply press the **AutoCal** button in the Visual Basic Control Panel, or call the **CalibrateDMM** or **DMMCalibrate** function using either DOS or Windows commands. The system calibration takes less than one second to perform.

Number of Counts To convert "Number of counts" to volts, multiply the "Resolution" by the "Number of counts". For example, the 250 mV Range has 3 counts of error, or 30 uV. The 250 mV Range accuracy for one year is .05% + 30 uV.

Accuracy vs. Reading Rates All of the above specifications apply to reading rates of less than 10 readings per second. For higher reading rates, increase the noise floor or "counts" by two times.

Reading Rates vs. Noise Rejection The best AC (50 Hz or 60 Hz) 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.

Reading Rate (RPS)	Set Constant (as defined in DMM_USER.H)	Best Rejection for Line Frequency
1	DMMRate1	60 Hz
1	DMMRate1p0	50 Hz
2	DMMRate2	60 Hz
2.5	DMMRate2p5	50 Hz
5	DMMRate5	60 Hz
5	DMMRate5p0	50 Hz
10	DMMRate10	50/60 Hz
12.5	DMMRate12p5	50 Hz
20	DMMRate20	50/60 Hz
25	DMMRate25	50 Hz
50	DMMRate50	50 Hz
100	DMMRate100	None
200	DMMRate200	None

2.7 Other Specifications

Temperature Coefficient, All Functions	Less than 0.1 x accuracy specification per ^{O}C from 0^{O}C to 18^{O}C and 28^{O}C to 50^{O}C.
Reading Rate	1 to 60 readings/sec
Overload Protection (voltage inputs)	250VDC or VAC across Voltage HI & LO terminals
Isolation	250VDC or 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 guidelines, for IEC 664 Installation Category I. All DMM inputs are to be used in protected, low energy circuits only.
Calibration	External calibrations are performed by *Signametrics* in a computer with a 10°C internal temperature rise. All calibration constants are stored within the SM-2010 DMM.
Temperature Range	0^{O}C to 50^{O}C, operating
Physical Size	One half length ISA card.
Power	+5 volts, 100 mA maximum. Can be turned ON/OFF by software control

2.8 SM-2010CT Specifications

The SM-2010CT is identical to the SM-2010 with the addition of a Frequency Counter. Specifications for this feature are shown below.

Note: There are two ways the frequency counter can be utilized: (1) It can make alternating frequency and amplitude measurements, or (2) It can make a frequency only measurement. The latter will provide the best frequency measurement sensitivity and speed, but the alternating measurement mode has the convenience of a dual measurement display. However, this results in degradation in both amplitude accuracy and frequency sensitivity. In the alternating mode (as in the control panel provided), measurements are made to the amplitude and frequency circuitry alternately. For best frequency counter operation, utilize the frequency only measurement function.

Frequency Measurement, for AC Voltage Input
Input Impedance: 1 MΩ in parallel with < 75 pF.
Sensitivity: Input Voltage > 250 mV up to 50 kHz, > 2V up to 100 kHz.

Frequency Range	5 Hz - 100 Hz	100 Hz-1 kHz	1 kHz-50 kHz	50 kHz-100 kHz
Resolution	1 mHz	10 mHz	100 mHz	1 Hz
Uncertainty is ±0.01% of reading ± adder shown	2 mHz	20 mHz	200 mHz	2 Hz
Input Signal Range [1]	10% - 110% of range	10% - 110% of range	10% - 110% of range	80% - 110% of range

[1] Input voltage required to give a valid reading in ACV ranges 2.5V, 25V, and 250V. For example, 10% -110% of range indicates that in the 2.5 VAC range, the input voltage should be 250 mV to 2.75 V.

Frequency Measurement, for AC Current Input
Input Impedance: 10Ω in the 2.5 mA and 25 mA range, 0.1Ω in the 250 mA and 2.5A ranges.
Sensitivity: Input Current > 10% of range up to 20 kHz.

Frequency Range	5 Hz - 100 Hz	100 Hz-1 kHz	1 kHz-10 kHz	10 kHz-20 kHz [2]
Resolution	1 mHz	10 mHz	100 mHz	1 Hz
Uncertainty is ±0.01% of reading ± adder shown	4 mHz	20 mHz	200 mHz	2 Hz
Input Signal Range [3]	10% -110% of range	10% - 110% of range	10% -110% of range	10% - 110% of range

[2] Performance from 10 kHz to 20 kHz is typical.
[3] Input current required to give a valid reading. For example, 10% - 110% of range indicates that in the 2.5 mA range, the input current should be 0.25 mA to 2.75 mA. The maximum current into the DMM is 2.5A.

All other specifications (DC Voltage, Resistance, AC Voltage, DC Current, and AC Current) are the same as the SM-2010.

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

3.0 Getting Started

After unpacking the SM-2010, please inspect for any shipping damage that may have occurred, and report any claims to your transportation carrier.

The SM-2010 is shipped with the SM-2010 PC System Digital Multimeter; a 3 1/2" floppy disk which contains a Windows Control Panel, "C" drivers, Windows drivers; and this Operator's manual.

3.1 Setting the SM-2010 Address

Before installing the SM-2010, you must first set the base address to one that does not conflict with any other PC cards. (The SM-2010 occupies four I/O locations, starting at the base address). The factory default Base Address is 0x200, (HEX 200). The address is determined by the set of switches located near the bottom of the DMM. Bit "A9" is the most significant bit, and is at the very bottom of the board, with **the two least significant bits (A1 and A0) assumed** to be 0.

The base address consists of three HEX digits. The most significant digit can be HEX 0 to 3, as shown on the top cover (see Figure 3-1 below). The second HEX digit can be 0-F; and the least significant digit can be either 0, 4, 8, or C. The hardware address must be called out whenever addressing the SM-2010 for the first time, using either the **DMMInit()** function in Windows; or the **InitDmm()** function in DOS. The following is an example of initializing a DMM at base address 200 (HEX):

```
/* initialize DMM */
DMMInit(0, 0X200);
```

If you are using the Visual Basic Control Panels provided, on initialization, the panel reads the base address from the configuration file **DMM.CFG**. The file **DMM.CFG** contains the instrument base addresses of the DMM(s) contained in your PC. This file must reside at the same directory as the Control Panel executable file (**SM2010.EXE**, or **HANDHELD.EXE**). An appropriate **DMM.CFG** file is created during the Windows **SETUP** of the DMM software. You can easily modify the **DMM.CFG** file using any ASCII text editor. A total of four consecutive addresses are used by either the SM-2010 or the SM-2010CT. The base address in the **DMM.CFG** file can be expressed in either HEX or decimal. The following **DMM.CFG** contents indicates a DMM at HEX address 0x200 (512 decimal):

```
512
```

Examples:

The HEX base address of 0x300 has a switch pattern of "11000000"; HEX 0x200 has a pattern of "10000000". HEX 0x304 has a pattern of "11000001". If you intend to use the Visual Basic Control Panels, the **DMM.CFG** file must be changed to have the same address as indicated by the switch pattern. For example, if you change the pattern to "11000001", the contents of **DMM.CFG** should be changed to 0x304 (or 772). Use any ASCII text editor to change the **DMM.CFG** file.

Tip: If you need assistance converting a HEX number to BINARY, **Windows 3.1 Accessories Group** has a **Calculator** that has this capability. Be sure to set the **View** to **Scientific** in the Calculator.

The software can support up to four DMMs in one chassis. To do so, the **DMM.CFG** file must contain the base address of each DMM as well as the DMM type. Each base address is on a separate line. No empty lines are allowed between each record. The following **DMM.CFG** contents indicates DMM number 0 (the first DMM) at HEX address 0x200 (512 decimal), and DMM number 1 is at 0x204 (516 decimal):

```
512
516
```

Note: The SM-2010 does not use any interrupts from the PC bus. There is no need to set one.

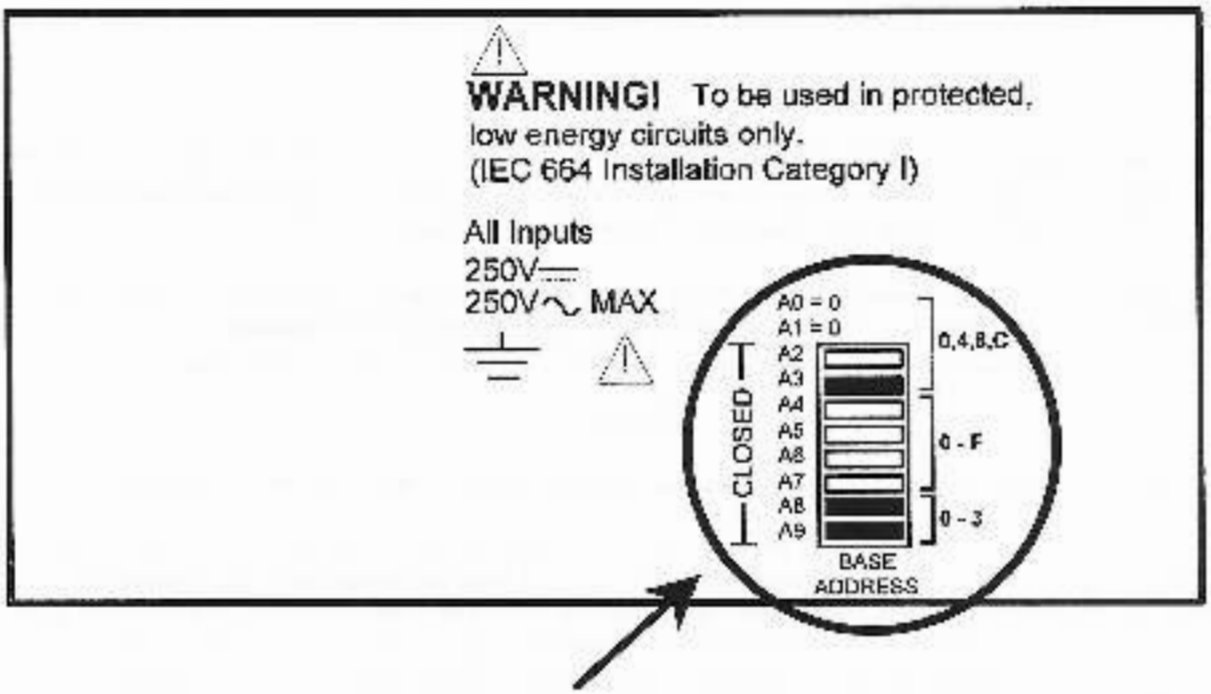


Figure 3-1. Base address set to HEX 308. Dark switch position corresponds to "Closed" or "1". There are three HEX digits that need to be set. A9 and A8 control the most significant digit (3 in this example), and can have a value of 0-3. The second HEX digit (0) is controlled by A7-A4, and can have a value of 0-F. The least significant digit (8 in this example) is controlled by A2 and A3, and can have a value of 0, 4, 8, or C. Note that bits A1 and A0 cannot be set, and are assumed to be 0. When using the Visual Basic Control Panel, the DMMLCFG file should contain "308".

3.2 Installing the SM-2010 Board

Warning

To avoid shock hazard, install the SM-2010 or SM-2010CT only into a personal computer that has its power line connector connected to an AC receptacle with an Earth Safety ground.

After installation, check to see that no loose wires or ribbon cables infringe upon any of the internal circuits of the SM-2010 or SM-2010CT, as this may apply measurement voltages to your computer!

Caution

Never install or remove the DMM from the computer unless the power to the PC has been turned off!

After setting the address, carefully plug the SM-2010 into a 16 bit ISA slot. If possible, choose an empty slot away from any high speed boards (e.g. video cards) or the power supply. **Please be patient during the installation process!** The SM-2010 comes with 3 safety input jacks. Because of their necessary size, they are a tight fit in many PC chassis. (This is especially true for IBM and Gateway computers.) Slide the bracket end of the SM-2010 into your PC first, watching for any interference between the safety input jacks and your PC chassis. **Be patient! You should only have to install it once!**

3.3 Installing the Software

For Windows 3.x installation, use the Windows 3.1 distribution disk provided. For Windows95 installation use the Windows95 distribution disk. Insert the SM-2010 floppy disk into your drive A: From the File Manager, choose the A: drive and run "**SETUP**".

3.4 Starting the Visual Basic Control Panel

You can gain familiarity with the SM-2010 by exercising its measurement functions using the main Visual Basic Control Panel. (Remember, your computer must have a mouse installed.) Go into the SM-2010 Program Group and double click on the SM-2010 icon. After a few seconds the DMM's relays should click, and the Control Panel (shown in the next section) should appear. If you do not hear the relays click, it is most likely due to a conflict of the SM-2010 address and another card. Change the base address of the SM-2010, and try again. You need to also change the file **DMM.CFG** to reflect the new base address. (See "Setting the SM-2010 Address" above.) The file **DMM.CFG** should reside in the same directory as **SM2010.EXE**.

Note: The SM-2010 front panel powers up in DCV, 5 readings per second, 250V range. If you place the DMM in autorange and leave the input terminals unconnected, you may periodically hear some relays click. This is perfectly normal. The input impedance of the 250 mV and 2.5V DC ranges are virtually infinite, so an open input will not read 0 volts. It will instead read whatever charge is associated with the signal conditioning of the DMM. As this electrical charge changes, the SM-2010 will automatically go to the appropriate range resulting in the relay clicking sound.

Note: There is also a simple handheld DMM Control Panel that you may use to exercise the SM-2010. The following sections describe the more complete Control Panel which controls more functions of the DMM.

4.0 SM-2010 Input Connectors

Before using the SM-2010, please take a few moments and review this section to understand where the voltage, current, or resistance inputs 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 may result.**

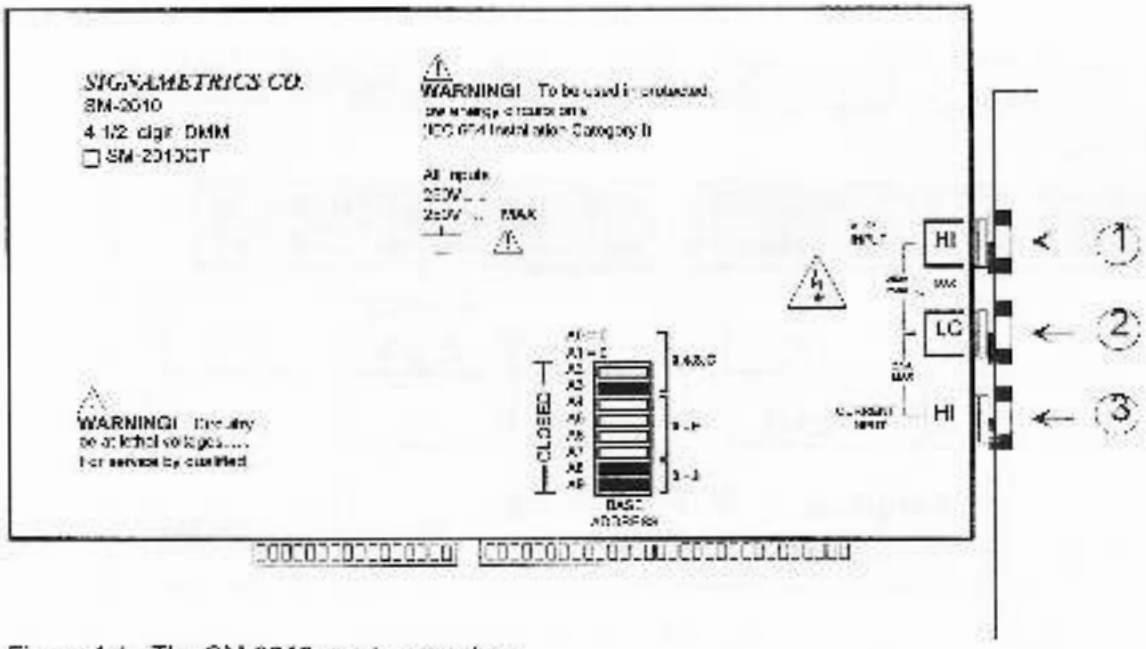


Figure 4-1. The SM-2010 input connectors.

(1) **V, Ω Input HI** This is the positive terminal for all Volts (including VAC Frequency), and Resistance measurements. The maximum input across **V, Ω HI** and **LO** is 250VDC or VAC.

(2) **LO** This is the negative terminal for ALL measurements. **Do not float this terminal or any other SM-2010 terminal more than 250VDC or VAC above Earth Ground.**

(3) **Current Input HI** This is the positive terminal for all Current (including IAC Frequency) measurements. The maximum input across **Current Input HI and LO** is 2.5A. Do not apply more than 40 V peak across these two terminals!

5.0 Measurement Functions

All of the SM-2010 measurement functions are accessible from Windows Control Panel (Figure 5-1, or from the Windows DLL. (To gain familiarity with the SM-2010, run the Windows Control Panel, as described in the previous section.) This section describes the SM-2010's capabilities, using the Windows control panel to guide you through each function. The features of the SM-2010CT are described as well.

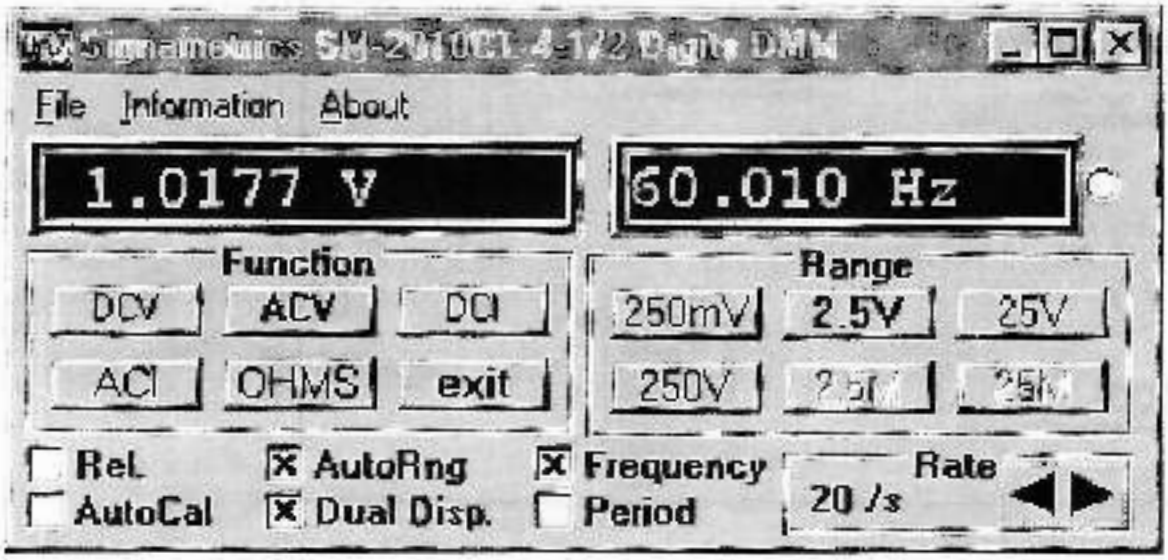


Figure 5-1. The Windows Control Panel for the SM-2010CT. The SM-2010 panel is very similar. The latter does not have the "Period", "Freq." buttons, or dual output display.

DCV Measures from 10 µV to 250V, at 1 reading per second (rps) to 60 rps. As you change the reading rate, the resolution automatically changes to give you optimum results. Use the Volts, Ω (V,Ω) HI & LO terminals, being certain to always leave the **Current Input HI** terminal unconnected.

DCI Measures from 100 nA to 2.5A, at 1 rps to 60 rps. Use the **Current Input HI and LO** terminals, being certain to always leave the V,Ω HI terminal unconnected.

ACV Measures from 10 mV (with 10 µV resolution) to 250V, from 5 Hz to 50 kHz. The ACV function is AC coupled (the DC signal component is not measured), and measures the true RMS value of the wave form. As with virtually all true RMS measuring meters, the SM-2010 may not read a perfect zero with a shorted input. This is normal. The reading rates are from 1 rps to 60 rps. Use the V,Ω HI & LO terminals, being certain to always disconnect the **Current Input HI** terminal.

ACI Measures from 100 µA (with 100 nA resolution) to 2.5A, at 1 rps to 60 rps. Use the **Current Input HI and LO** terminals, being certain to always disconnect the V,Ω HI terminal.

Resistance Measures from 10 mΩ to 25 MΩ, at 1 rps to 60 rps. Use the VΩ HI & LO terminals, being certain to always disconnect the **Current Input** HI terminal. When measuring values below 25 kΩ, you should use the Relative function below to compensate for test lead resistance.

Frequency (SM-2010CT only) Frequency counter, which measures from 5 Hz to 100 kHz. When activated, the control panel alternately updates the amplitude reading followed by the frequency reading, for amplitudes > 250 mVAC. In this dual display mode, the amplitude accuracy is *typical and for indication* only. To determine the actual amplitude, you should disable the Frequency Counter. For best results when using the Frequency Counter, you should select an amplitude range that puts the input signal near the full scale point of that range. The inverse of Frequency, **Period**, is also selectable.

Rel 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 Resistance. For example, when using Resistance, you can null out lead resistances by shorting the leads together and clicking on **Rel**. When making low level DC voltage measurements (in the µV region), first apply a copper short to the VΩ HI & LO input terminals, allow the reading to stabilize for 10 seconds, and click on **Rel**. This will correct for any offsets internal to the SM-2010.

Rate Controls the SM-2010 reading rate. Rates of 1, 2, 5, 10, 15, 20, 25, 30, 40, 50, and 60 readings per second are available. For most applications, you should choose a reading rate that gives the best power line rejection. See Section 2.6, Accuracy Notes.

Range Can be set to **AutoRange** or manual by clicking on the Range Control Buttons. In **AutoRange**, the SM-2010 automatically chooses the optimal range.

250 mV, 2.5V, 25V, 250V, etc. These rows of buttons contains the ranges of the SM-2010. Clicking on any Range Button will change the SM-2010 range, provided the **AutoRange** button is not selected, and will lock the DMM in that range. The labeling and the number of range buttons change as you change the SM-2010 functions.

Locking to a range, rather than **AutoRange**, is useful in many applications, especially to speed up measurements in a system. Locking a range avoids any intermediate readings that can occur during the autoranging process. Another reason to lock a range is to control the input impedance in DCV. The 250 mV and 2.5V ranges have virtually infinite input impedance, while the 25V and 250V ranges have 10 MΩ input impedance.

You may also want to control the Ohms current used in making resistance measurements. (See Section 2.2 for a table of resistance range vs. current level.) All of the Ohms ranges of the SM-2010 have enough current and voltage compliance to turn on diode junctions. For best performance when checking diodes, use the 25kΩ range. For characterizing semiconductor part types, performing diode tests in ranges corresponding to desired excitation currents may be helpful.

6.0 Making Measurements

In general, you should choose the measurement rate appropriate for the local power line frequency. See Section 2.6, Accuracy Notes, for a list of reading rates that give excellent noise rejection. For best noise rejection and stable readings, use the slowest measurement rate whenever possible.

6.1 Voltage Measurements

Making Voltage Measurements is straightforward. There are a few tips you should follow when making voltage measurements:

1) ACV measurements, if possible, should have the NEUTRAL or GROUND attached to the SM-2010 **LO** terminal. See Figure 6-1, below. This prevents any "Common Mode" problems from occurring ("Common Mode" refers to floating the SM-2010 above Earth Ground.) Common Mode problems can result in noisy readings, or even cause the PC to hang-up under high V x Hz input conditions. In many systems, grounding the source to be measured at Earth Ground (being certain to avoid any ground loops) can yield better results.

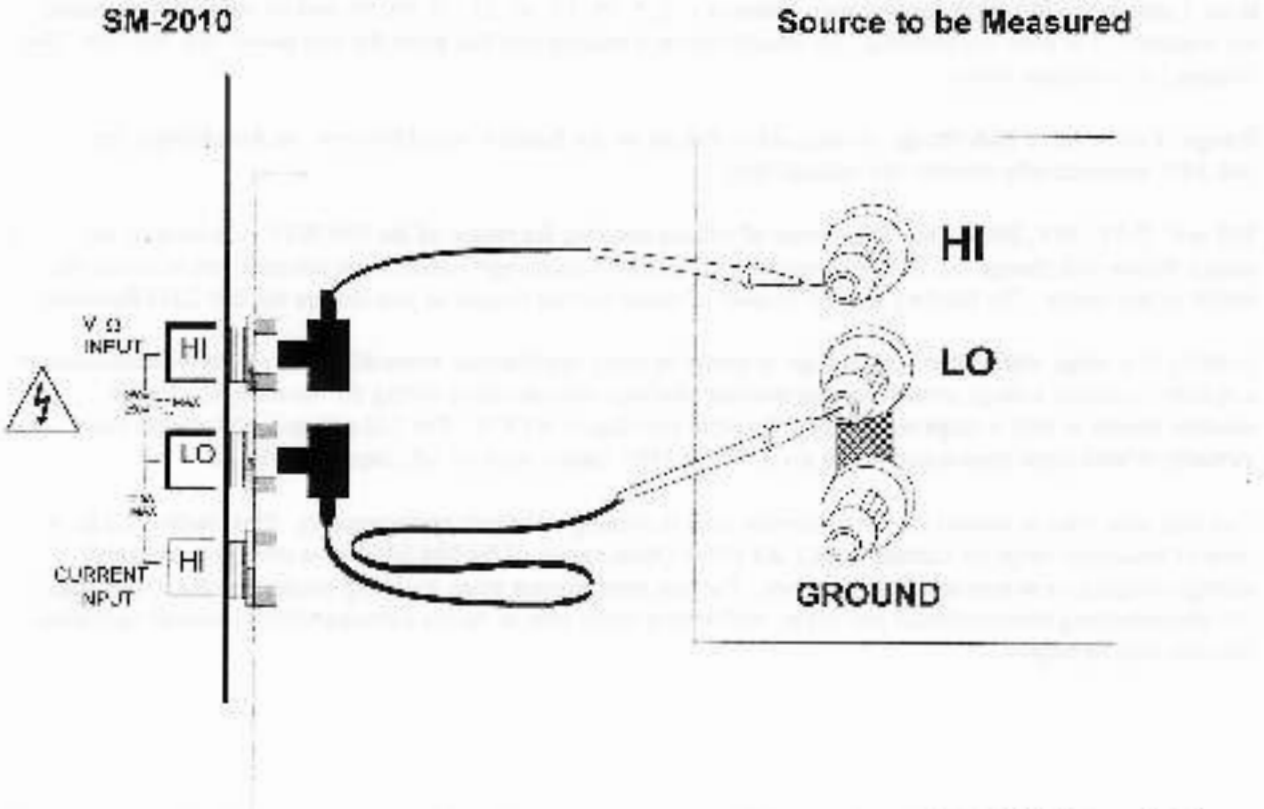


Figure 6-1. Make ACV measurements with the source ground attached to the SM-2010 **LO** to minimize "Common Mode" measurement problems.

2) When making very low level DCV measurements (<100 uV), you should first short the SM-2010 with a copper wire shorting plug and perform **Relative** to eliminate zero errors before making your measurements. Also be aware that your test leads can introduce several µVolts of error, if they have high thermal voltages relative to copper. To minimize thermal voltaic effects after handling the test leads, you should wait a few seconds before making measurements.

6.2 DC and AC Current Measurements

Always remember that the Current functions are protected with a 2.5A, 250V fuse. Applying high voltages to the **Current Input HI** & **LO** inputs can be dangerous! 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 SM-2010. This is easily accomplished by opening all inputs to the SM-2010 and performing **Relative** in the appropriate DCI range.

6.3 Resistance Measurements

Simply connect the **V,Ω HI** to one end of the resistor, and the **LO** to the other end. If the resistor to be measured is less than 25 kΩ, you should null our any lead resistance errors by firmly touching the **V,Ω HI** and **LO** test leads together and then performing a **Relative** function. See Figure 6-2 below. If making measurements above 250 kΩ, you should use shielded leads to minimize noise pickup.

Knowing the SM-2010 Source current in Ohms measurements may be helpful to ensure repeatability of your measurements. (See Section 2.2, Resistance, for a table of resistance range vs. current level.)

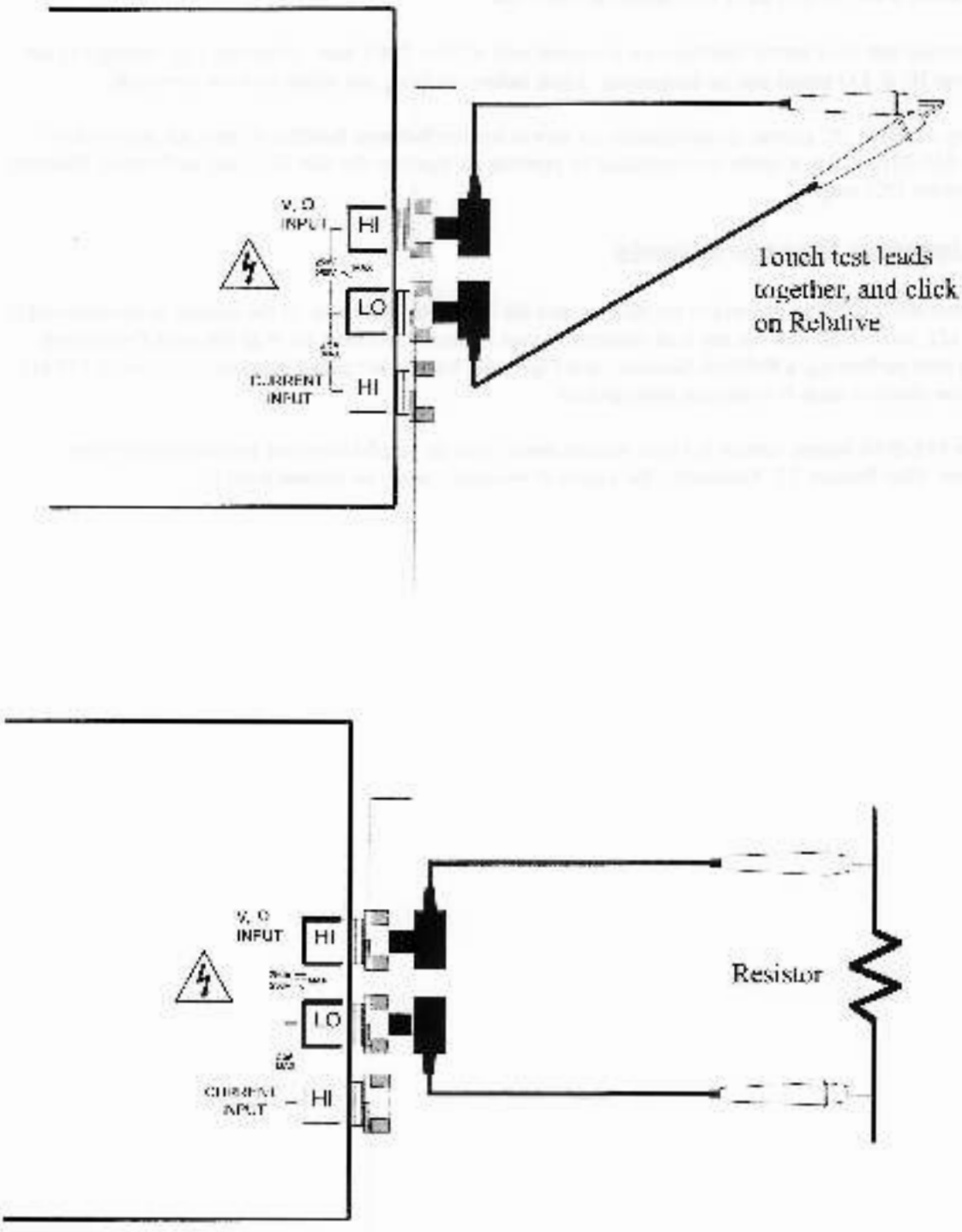


Figure 6-2. When making low resistance measurements below 25 kΩ, you should first null out the lead resistance by firmly touching the lead tips together, and then activate the Relative function on the SM-2010. Now, you can make more accurate resistance measurements.

7.0 SM-2010 DOS Library and Interface

7.1 Distribution Files

Before installing, make sure you copy and save the distribution diskette. This chapter describes the DOS functions and tools found after installing the software. DOS libraries are provided for Microsoft C++ and Borland C++. Both the Borland C++ DOS library (**2010BLIB.LIB**) and the Microsoft Visual C++ DOS library (**2010VCLB.LIB**) are compiled for the Large Memory Model. To install the software, run **SETUP** from the distribution diskette. **SETUP** installs the SM-2010 DOS software in the directory C:\SM2010 (default install directory).

The DOS part of the distribution diskette contains the following files:

File	Description
README.TXT	Release notes, latest information at the time of release. Important!
SM2010VC.C	Sample code which demonstrates simple application written in Microsoft C for the 2010. It uses the DOS library **2010VCLB.LIB**.
SM2010VC.EXE	The executable for the above code (**SM2010VC.C**).
SM2010VC.MAK	Make file for Microsoft C compiler which is used to compile **SM2010VC.C** with the **2010VCLB.LIB** and the headers.
2010VCLB.LIB	Microsoft C DOS library for Large memory model, used by the above sample program. See this manual for available commands. Also requires **DMM_USER.H** and **PROT2010.H**
SM2010BC.C	Sample code which demonstrates simple application written in Borland C for the 2010. It uses the DOS library **2010BLIB.LIB**
2010BORS.DSW, 2010BORS.IDE	Borland sample code (**SM2010BC.C**) make files.
2010BLIB.LIB	Borland C DOS library for Large memory model, used by the above sample program. See this manual for available commands. Also requires **DMM_USER.H** and **PROT2010.H**
BOR_READ.TXT	Readme file with a list of the Borland files.
DMM_USER.H	Driver header file, constant and error codes and definitions.
PROT2010.H	Function prototypes for use with the library files.

7.2 Addressing the DMM

Up to four SM2010 DMMs (and four SM-2020 DMMs) can be installed in your PC. When using DOS, the SM-2010 DMM is addressed using the function **InitDmm()**. The first parameter of **InitDmm()** refers to the DMM number: DMMs are numbered starting from 0 for the first, to 3 for the fourth. The second parameter in **InitDmm()** is the base address of the DMM. The base address **must correspond** to the selected base address of the switches on the DMM. Please see "Setting the SM-2010 Address" for details.

7.3 Using the SM-2010 Library

Copy the header files in a directory that will be searched by your C compiler for header files. For Borland C++ use the **2010BLIB.LIB** for import libraries. Also provided are make files and a sample source code for each compiler. Both Borland and Microsoft DOS libraries contain the same command set, documented in this manual. Both also use the same header files.

In using the SM-2010 driver, you must first call **InitDmm()** to set the DMM base address. Call **SetDmmFuncRange()** next to set the DMM function and range. The DMM function/range constants are defined in the **dmm_user.h** header file, and have names that clearly indicate the function and range they invoke. Notice that the codes are inter-mixed. Do not hard-code the numeric values, since they may change. Use **SetAutoRange()** to control autoranging. Use **SetDmmRate()** to set the reading rate using the **DMMRate*** defined in the **DMM_USER.H** header file

Four functions are provided to return DMM readings. **ReadDmmFloat()** returns the next reading as a scaled single-precision (float) result, **ReadDmm()** returns the next reading as a scaled double-precision (double) result, **ReadDmmStrg()** returns the next reading as a formatted string ready to be displayed, and **ReadDmmNorm()** returns a double-precision (double) normalized value (not scaled for range).

7.4 DOS Default Modes and Parameters

The DOS default modes and parameters on your SM-2010 and SM-2010CT are set up as follows:

- Function is set to VDC.
- Measurement rate is set to RATE_10, or 10 readings per second.
- Autoranging is disabled, range is set to 25 V.
- Relative mode is disabled.

7.5 DOS Command Language

The following sections contain detailed descriptions of each function of the DOS command language. Those commands that pertain to only the SM-2010CT are indicated. SM-2010 DOS software can support up to four DMMs. The card number is a value from 0 for the first DMM, to 3 for the fourth DMM. The card number is an integer *nDMM* and is the first parameter of each command (except for the **GetVer** function).

Note: Error codes are defined in **dmm_user.h** file. All error codes are defined as negative integers.

CalibrateDmm

Description

Internally calibrate the DMM's analog-to-digital converter. Performs the same function as the **Auto**Cal function on the Visual Basic Control Panel.

int CalibrateDmm(int *nDMM*)

Remarks

Recalibrate the internal analog-to-digital converter of the DMM specified by *nDMM* (DMMs are numbered from 0 to 3), preserving its current state after the calibration process is complete.

Return Value

The return value is an integer error code.

Example

```
if(CalibrateDmm(0) != DMM_OKAY)
printf("H/W error");
```

Comments

This performs an internal calibration of the DMM's analog-to-digital converter. This calibration must be performed at least once a month, with all inputs to the DMM disconnected. It is not related to the external calibration required on a yearly basis.

ClearRelative

Description

Terminate (clear) the DMM relative measurement operation mode.

void **ClearRelative(int** *nDMM*)

Remarks

Clear the relative mode of operation on the DMM specified by *nDMM* (DMMs are numbered from 0 to 3). The offset is set to zero. Function changes automatically clears relative mode.

Return Value

None

Example

```
ClearRelative(0);
```

GetCalDate

Description Return the calibration date string.

#include "dmm_user.h"

int GetCalDate(int *nDMM*, **char far** **pstrBuf)*

Remarks This function reads the date on which the DMM was last calibrated. The string containing this date is written to the buffer pointed to by *pstrBuf*. The DMM is specified by *nDMM*. Make sure the buffer pointed to by *pstrBuf* is 16 or more characters long.

Return Value The return value is an integer error code.

Example

```
char buffer[16]; GetCalDate(0,buffer);
printf("Cal Date is %s", buffer); /* print a cal date */
```

GetCalLab

Description Return the calibration laboratory string (i.e. where the DMM was calibrated).

#include "dmm_user.h"

int GetCalLab(int *nDMM*, **char far** *^pstrBuf)*

Remarks This function reads the Lab at which the DMM was last calibrated. The string containing this location is written to the buffer pointed to by *pstrBuf*. The DMM is specified by *nDMM*. Make sure the buffer pointed to by *pstrBuf* is 16 or more characters long.

Return Value Integer error code.

GetDmmFuncRange

Description Get the DMM composite function and a range.

#include "dmm_user.h"

int GetDmmFuncRange(int *nDMM*)

Remarks Gets the DMM specified by *nDMM* (DMMs are numbered from 0 to 3) composite function and range code. Composite codes are defined in **dmm_user.h**.

Return Value int composite function/range code

Example

```
int func_range;
func_range = GetDmmFuncRange(0); /* get the combo code */
if( func_range == VDC_250 ) printf("high DCV level");
```

GetDmmFunction

Description Get the DMM function.

#include "dmm_user.h"

int GetDmmFunction(int *nDMM***)**

Remarks Gets the function of the DMM specified by *nDMM* (DMMs are numbered from 0 to 3) as defined in dmm_user.h. Possible return values are: **VDC**, **VAC**, **IDC**, **IAC**, and **OHMS**.

Return Value Integer function code

Example

```
if( GetDmmFunction(0) == VDC) printf("VDC selected");
```

GetDmmRange

Description Get DMM range.

#include "dmm_user.h"

int GetDmmRange(int *nDMM***)**

Remarks Gets the range value of the DMM specified by *nDMM* (DMMs are numbered from 0 to 3). There are at most six ranges, zero to five, depending on selected function. Zero is the first and lowest range.

Return Value int range value

Example

```
if( GetDmmRange(0) == 0 printf("Lowest range.");
```

GetDmmRate

Description Get DMM reading rate.

#include "dmm_user.h"

double GetDmmRate(int *nDMM***)**

Remarks Returns reading rate of the DMM specified by *nDMM* (DMMs are numbered from 0 to 3) in double float format.

Return Value double format rate value.

Example

```
double rate; rate = GetDmmRate(0);  /* read actual
measurement rate */
```

GetFreq (SM-2010CT only)

Description Take a single frequency measurement.

#include "dmm_user.h"

double GetFreq(int *nDMM***)**

Remarks Take a single frequency reading. Do a counter autorange to optimize the counter for the next frequency reading. If the Counter is not installed, or the DMM is not in ACV or ACI, a zero frequency value is returned. If the DMM is in autorange, be certain to take an amplitude reading before using this command.

Return Value Double Floating Frequency in Hertz.

Example

```
printf("My wave Frequency is %7.3 Hz\n",GetFreq(0) );
```

GetFreqStr (SM-2010CT only)

Description Take a single frequency measurement.

#include "dmm_user.h"

int GetFreqStr(int *nDMM*, **char far** **lpszReading***)*

Remarks Takes a single frequency reading. Do a single counter autorange to optimize the counter for the next frequency reading. If the Counter is not installed, or the DMM is not in ACV or ACI, a zero frequency value is returned. If the DMM is in autorange, be certain to take an amplitude reading before using this command.

lpszReading is a pointer to a buffer of at least 16 characters to hold the converted result. The return value is a formatted string containing the frequency in a five digit format with units.

Return Value Integer error code

Example

```
char cBuff[17]; int status;
status = GetFreqStr(0, cBuff);
printf("Frequency is %s\n",cBuff );
```

GetHwrVer

Description Get hardware version of the DMM

int GetHwrVer(int *nDmm***)**

Remarks This function provides the hardware version of the DMM specified by *nDMM*.

Return Value Integer version number. 0 for version A, 1 for B etc..

Example

```
hardware_ver = GetHwrVer(0);
```

Comments It is important to provide this value in any service calls to Signametrics.

GetPeriod (SM-2010CT only)

Description Make a single period measurement.

#include "dmm_user.h"

double GetPeriod(int *nDMM*)

Remarks Takes a single period reading. Do a counter autorange to optimize the counter range for the next period reading. If the Counter is not installed, or the DMM is not in ACV or ACI, a zero period value is returned. If the DMM is in autorange, be certain to take an amplitude reading before using this command, to guarantee the minimum amplitude requirement is met.

Return Value Double Floating Point period (in seconds).

Example

```
printf("My wave Period is %7.3 S\n",GetPeriod(0) );
```

GetPeriodStr (SM-2010CT only)

Description Take a single period measurement.

#include "dmm_user.h"

int GetPeriodStr(int *nDMM*, **char far** **lpszReading*)

Remarks Takes a single period reading. Do a counter autorange to optimize the counter range for the next period reading. If the Counter is not installed, or the DMM is not in ACV or ACI, a zero period value is returned. If the DMM is in autorange, be certain to take an amplitude reading before using this command, to guarantee the minimum amplitude requirement is met.

lpszReading is a pointer to a buffer of at least 16 characters to hold the converted result. The return value is a formatted string containing the period in a five digit format with units.

Return Value Integer error code

Example

```
char cBuff[16]; int status;
status = GetPeriodStr(0, cBuff);
printf('Period is %s\n",cBuff );
```

GetPrintFormat

Description Get the current print format of the DMM.

#include "dmm_user.h"

char * GetPrintFormat**(int** *nDMM*)

Remarks Gets the current print format of the DMM specified by *nDMM* (DMMs are numbered from 0 to 3).

Return Value Pointer to a string which has the printing format (e.g., "%6.3f")

Example

```
printf(GetPrintFormat(0), lpfRes);
```

GetSerialNumber

Description Get serial number of the DMM.

int GetSerialNumber(int *nDmm*)

Remarks This function provides the serial number of the DMM specified by *nDMM*.

Return Value Integer serial number

Example
```
serial_number = GetSerialNumber(0);
```

Comments It is important to provide this value in any service calls to Signametrics.

GetUnits

Description Get the appropriate measurement units string.

char * GetUnits(int *nDMM*)

Remarks Gets the units appropriate for the current measurement range and function, for the DMM specified by *nDMM*.

Return Value Pointer to a string which has the units (e.g., "mV")

Example
```
printf(GetPrintFormat(0), lpfRdg);    // Print reading
printf(" %s\n",GetUnits(0));          // print units
```

GetVer

Description Get main driver library software version number

double GetVer(void)

Remarks This function provides the driver software version.

Return Value Double floating version number.

Example
```
software_ver = GetVer();
```

Comments It is important to provide this value in any service calls to Signametrics.

HasCount (SM-2010CT only)

Description Test if the Counter is available.

int HasCount(int *nDMM*)

Remarks This function detects whether or not the Counter is installed in the DMM specified by *nDMM* (DMMs are numbered from 0 to 3).

Return Value Integer status code

Value	Meaning
TRUE	Counter is available.
FALSE	Counter is not available.

Example

```
int status;
if(HasCount(0)) status = GetPeriodStr(0, cBuff);
```

InitDmm

Description Initialize the DMM

#include "dmm_user.h"

int InitDmm(int *nDMM*, **int** *base_add*)

Remarks Initialize the DMM specified by *nDMM* (DMMs are numbered from 0 to 3) at base address specified by *base_add* (in HEX). The hardware is set to an initial state which is VDC, 25V range, 10 readings per second.

Return Value Integer Error code.

Example

```
InitDmm(0, 0x200);
```

IsAutoRange

Description Returns the state of autorange.

#include "dmm_user.h"

int IsAutoRange(int *nDMM*)

Remarks See **SetAutoRange()** and **ClearAutoRange()**.

Return Value Return **TRUE** if autorange is selected. FALSE if it is not. TRUE and FALSE are defined in **dmm_user.h**.

Example

```
if( IsAutoRange(0) ) printf("DMM is in Autorange");
```

IsRelative

Description Returns the Relative state of the DMM.

#include "dmm_user.h"

int IsRelative(int *nDMM*)

Remarks Returns the state of the Relative mode in the DMM specified by *nDMM* (DMMs are numbered from 0 to 3).

Return Value Return TRUE if DMM is set to relative mode, FALSE if it is not. TRUE and FALSE are defined in **dmm_user.h**.

Example

```
if( IsRelative(0) ) printf("DMM is in relative mode");
```

ReadDmm, ReadDmmFloat

Description

Return the current floating-point reading from the DMM specified by *nDMM* (DMMs are numbered from 0 to 3).

#include "dmm_user.h"

int ReadDmmFloat(int *nDMM*, float **fResult*)

int ReadDmm(**int** *nDMM*, **double** **dResult*)

Remarks

ReadDmmFloat reads the current result from the DMM, performs all scaling and conversion required, and returns the result as a 32-bit single-precision floating-point number. **ReadDmm** is the same as **ReadDmmFloat** but it returns a 64-bit double-precision floating-point number. The DMM continuously makes measurement at the set rate without any external software intervention. As a new measurement becomes available, it replaces the old measurement. Therefore, if the rate at which **ReadDmm** or **ReadDmmFloat** is executed does not keep up with the current DMM rate, old measurements are ignored and current measurements are returned. *dResult* is a pointer to a double and *fResult* is a pointer to a floating number at which these functions place the measurement result. The returned value is scaled to the currently set range as the examples in the following table:

DMM Range	Input	Returned value
250mV	100mV	100.0
2.5V	1.0V	1.0
25V	10V	10
250V	100V	100.0
2.5mA	1mA	1.0
25mA	10mA	10.0
250mA	100mA	100.0
2.5A	1 A	1.0
250 Ohm	100	100
2.5k	1k	1.0
25k	10k	10.0
250k	100k	100.0
2.5Meg	1Meg	1.0
25Meg	10Meg	10.0

Return Value

The return value is an integer error code.

Example 1

```
float f;
ReadDmmFloat(0, &f);
```

Example 2

```
double d;
ReadDmm(0, &d);
```

ReadDmmNorm

Description

Return the current floating-point reading from the DMM specified by *nDMM* (DMMs are numbered from 0 to 3) in base units.

#include "dmm_user.h"

int ReadDmmNorm(int *nDMM*, **double ****dResult***)**

Remarks

ReadDmmNorm reads the current result from the DMM, performs all scaling and conversion required, and returns the result as a 64-bit double-precision floating-point number. *dResult* is a pointer to a double precision floating number at which the function stores the measurement result. It is different from **ReadDmm** and **ReadDmmFlout** in its returned value, which is not scaled to the set range. It is instead in the base value as in the following table:

DMM Range	Input	Returned value
250mV	100mV	0.1
2.5V	1.0V	1.0
25V	10V	10.0
250V	100V	100.0
2.5mA	1mA	1.0e-3
25mA	10mA	10.0e-3
250mA	100mA	100.0e-3
2.5A	1 A	1.0
250 Ohm	100	100.0
2.5k	1k	1.0e3
25k	10k	10.0e3
250k	100k	100.0e3
2.5Meg	1Meg	1.0e6
25Meg	10Meg	10.0e6

Return Value

Integer error code.

Example

```
double d; int i;
i = ReadDmmNorm(0, &d);
```

ReadDmmStrg

Description

Return the next reading from the DMM specified by *nDMM* (DMMs are numbered from 0 to 3), formatted for printing.

#include "dmm_user.h"

char *ReadDmmStrg(int *nDMM*)

Remarks

This function reads the next result from the DMM, performs all scaling and conversion required, and returns the result as a string formatted for printing. The print format is determined by the set range and function. It is scaled as **ReadDmm()**.

Return Value

The return value is a pointer to the measurement string.

Example

```
printf("%s",ReadDmmStrg(0)); /* print a measurement */
```

SetAutoRange, ClearAutoRange

Description

Select autoranging operation. ClearAutoRange operation for the DMM specified by *nDMM* (DMMs are numbered from 0 to 3).

#include "dmm_user.h"

void SetAutoRange(int *nDMM*)

void ClearAutoRange(int *nDMM*)

Remarks

Sets the DMM to autoranging operation mode. The DMM will automatically select the appropriate range for the input. It will remain in this mode until a range is changed or a **ClearAutoRange()** command is issued. It is compatible with function changes. When in autorange mode, the DMM will up range when the reading is over 100% of the current range (e.g. 25,000 counts). Down ranging will occur when the reading falls below 9% of the current range (e.g. 2,250 counts).

Return Value

None

Example

```
SetAutoRange(0); /* begin autoranging operation */
```

SetDmmFuncRange

Description

Set the DMM specified by *nDMM* (DMMs are numbered from 0 to 3) to a function and range at the same time.

#include "dmm_user.h"

void SetDmmFuncRange(int *nDMM*, int *funcrange*)

Remarks

Sets the DMM function and range using a single command. When having to change both function and range, it is faster to use **SetDmmFuncRange** than to use the SetDmmFunction followed by **SetDmmRange.** Composite *funcrange* values are defined in dmm_user.h.

Return Value

None

Example

```
SetDmmFuncRange(0, VDC_25V);
```

SetDmmRate

Description Set DMM reading rate.

#include "dmm_user.h"

int SetDmmRate(int *nDMM*, int *rate*)

Remarks Sets reading rate of the DMM specified by *nDMM* (DMMs are numbered from 0 to 3) to a pre-set value. The legal rate values are defined in **dmm_user.h.** This function tests *rate* for validity and returns an error code. The default rate on initialization is five readings per second (DMMRate5).

Return Value Integer error code.

Example
```
SetDmmRate(0, DMMRate10);  /* Select 10 readings/Sec. */
```

SetRelative

Description Set the DMM to relative measurement operation.

void SetRelative(int *nDMM*)

Remarks Sets the DMM specified by *nDMM* (DMMs are numbered from 0 to 3) to relative operation mode. The current reading is subtracted from all subsequent readings, until the function is changed or **ClearRelative()** function is issued. It is compatible with DMM range changes.

Return Value None

Example
```
SetRelative(0);
```

ShutDownDmm

Description Turns the power off to the measurement section of the DMM.

#include "dmm_user.h"

int ShutDownDmm(int *nDMM*)

Remarks This function turns off the power to the measurement section of the DMM specified by *nDMM* (DMMs are numbered from 0 to 3). Power is turned on when executing the **InitDmm()** function call.

Return Value DMM OKAY

Example
```
ShutDownDmm(0);   /* Shut off power to DMM #0 */
```

8.0 SM-2010 DMM Windows Interface

8.1 Distribution Files

In addition to the DOS files, the distribution diskette contains the Microsoft® Windows™ SM-2010 DMM software. 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 Manager window. Most files on this diskette are compressed, and must be installed using the SETUP program.

The SM-2010 DMM DLL is a protected-mode Microsoft® Windows™ DLL that controls the *Signametrics'* DMM. It is provided with two samples of Visual Basic™ front-panel application to demonstrate the DMM and the interface method 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
SM2010.DLL	The 16 bit Dynamically Loaded Library (DLL) for the SM-2010 and SM-2010CT. See this manual for available commands. See the Visual Basic sources for declarations needed for the DLL functions. This file is installed in the Windows System directory. It could also be used in your working directory.
SM2010.DEF	SM-2010 driver DLL module definition file.
SM2010.LIB	DLL Library file, for use mainly with C programming in Windows.
DMM_USER.H	Header file needed for both Windows DLL and DOS libraries.
PROT2010.H	Header file containing all DOS functions declarations.
SM2010.H	Windows DLL function declaration file.
	Visual Basic 3.0 MAIN CONTROL PANEL RELATED FILES
\SM2010FP	Subdirectory containing the Visual Basic Control Panel, **SM2010.EXE**, its source code, and associated files. This program was generated with Visual Basic version 3.0.
VBRUN300.DLL	Visual Basic run-time interpreter. Install in your **C:\WINDOWS\SYSTEM** (or equivalent) directory.
THREED.VBX	Visual Basic extension DLL. Install in your **C:\WINDOWS\SYSTEM** (or equivalent) directory.
SPIN.VBX	Visual Basic extension DLL. Install in your **C:\WINDOWS\SYSTEM** (or equivalent) directory.
GLOBAL.BAS	Main Control Panel global declarations source file.
DMM.CFG	Configuration file which has the I/O base address for the SM-2010. This file is read by the control panels.
SM2010.EXE	The main Control Panel VB3.0 executable file, which reads **DMM.CFG** for the I/O base address.

SM2010.FRM	Main control panel "form" source code.
SM2010.MAK	Main control panel VB3.0 make file. Open it as a project from VB3.0 to make changes or view the source code enclosed.

Visual Basic 3.0 HANDHELD DMM CONTROL PANEL RELATED FILES

GLOB.BAS	Simple handheld Control Panel global declarations source file.
HANDHELD.EXE	The simple panel VB3.0 executable file, which reads DMM.CFG for the I/O base address.
DMMFORM.FRM	Simple handheld panel "form" source code.
HANDHELD.MAK	Handheld panel VB3.0 make file. Open it as a project from VB3.0 to make changes or view the source code enclosed.

To install the files from the distribution diskette, run the program **A:\SETUP.EXE** from the Program Manager's "File|Run..." dialog, or double-click the file name **A:\SETUP.EXE** from the File Manager. The files contained on the distribution diskette are compressed, and must be installed using SETUP.

8.2 Configuration File

The SM-2010 DMM is configured in one of two ways. If writing software in Visual Basic, C++ or other programs, the base address is passed to the DLL on the first call, using the **DMMInit()** function (or **InitDmm()** for DOS applications). The base address is the setting of the base I/O address switches on the card.

If using the Visual Basic Control Panel, on initialization, the panel reads the base address from a configuration file, **DMM.CFG**. The file DMM.CFG contains the instrument base addresses of the DMMs contained in your PC. A total of four consecutive addresses are used by either the SM-2010 or the SM-2010CT. The following **DMM.CFG** contents indicates a DMM at HEX address 0x200 (512 decimal):

```
512
```

Up to four DMMs can be installed in your PC. DMMs are numbered starting from 0 for the first, to 3 for the fourth. **DMM.CFG** must have a line indicating the base address of each DMM. No empty lines are allowed. Use any ASCII text editor to modify **DMM.CFG**, to reflect the DMM type and the base address. The following **DMM.CFG** contents indicates DMM number 0 is at Hex address 0x200 (512 decimal), and DMM number 1 is at 0x204 (516 decimal):

```
512
516
```

8.3 Using the SM-2010 Windows Driver

Install the **SM2010.H** header file in a directory that will be searched by your C/C++ compiler for header files. Install **SM2010.LIB** in a directory that will be searched by the linker for import libraries. During Setup, the DLL file **SM2010.DLL** is installed in the C:\WINDOWS\SYSTEM directory, which is automatically searched when a Windows application issues a **LoadLibrary** command. If desired, copy the **SM2010.DLL** in a location where either your program will do a **LoadLibrary** call, or in a **PATH** such that Windows will load the DLL automatically.

In using the SM-2010 driver, first call **DMMInit** to read the DMM configuration and calibration information, and to identify the I/O base address. Call **DMMSetFnRange** to set the DMM measurement function and range. The DMM function/range constants are defined in the **dmm_user.h** header file, and have names that clearly indicate the function they invoke. Notice that the codes are inter-mixed. Don't hard-code the numeric values, since they may change. Use **DMMSetRate** to set the reading rate using the **DMMRate*** defines in the header file.

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

All functions accept a DMM-number parameter. Each function returns an error code.

8.4 Visual Basic Control Panel Application

The Visual Basic Control Panel, **SM2010.EXE** (in the **SM2010FP** subdirectory) is a simple interactive control panel for the SM-2010 DMM. When it loads it will take a few seconds to initialize the hardware before the front panel is displayed.

The push-buttons labeled "DCV", "ACV", etc., control the DMM function. As you push one, the front panel application will switch the DMM to the selected range and function. Please see Section 5.0, Measurement Functions, for more details.

The source code file **GLOBAL.BAS** contains the interface and constant definitions required to write Visual Basic applications which interact with the driver DLL, along with some global variables required for this particular front-panel application. See the **README.TXT** file in **V_BASIC** for more information.

8.5 Windows DLL Default Modes and Parameters

The Windows DLL default modes and parameters on your SM-2010 and SM-2010CT are set up as follows:

- Function is set to DCV.
- Measurement rate is set to RATE_10, or 10 readings per second.
- Autoranging is disabled, range is set to 25V.
- Relative mode is disabled.

8.6 Using the SM2010 DLL with LabWindows/CVI® Version 3.1

If using LabWindows/CVI® Version 4.01 or higher, you must use the *Signametrics* Windows95 Release SOFTWARE. When using the SM2010DLL with LabWindows/CVI, you should make these changes to the **SM2010.H** file:

1. Include a LabWindows/CVI type definitions header file (for instance '**winevi.h**'), which is needed to call your other DLLs including the SM2010.DLL. See below

   #include "winevi.h"

2. Delete '__export' from all functions declarations.

3. Replace type 'int' with type 'short'

4. Replace type 'unsigned' with type 'unsigned short' for instance:

   change: int __export WINAPI DMMSetRate(int *nUnit*, int *nRate*);

   to: short WINAPI DMMSetRate(short *nUnit*, short *nRate*);

The header file (for example: **wincvi.h**) is needed to compensate for the National Instruments DLL interface limitation. The **wincvi.h** file is shown below:

```
/* filename: WINCVI.H

        This file contains the required definitions of Window's data types used to call driver DLL's such as
        Signametrics' SM2010.DLL. The definitions are consistent with Watcom's <windows.h>, as required by
        National Instruments LabWindows/CVI (R).

*/

#ifndef WINCVI__H

        #define WINCVI__H 1
        #define FALSE 0
        #define TRUE    1
        #define OFF     0
        #define ON      1
        /*** common definitions and typedefs ***/
        #define VOID    void
        #define FAR     __far
        #define near    __near
        #define pascal  __pascal
        #define CDECL __cdecl
        #define WINAPI          __far __pascal
        #define CALLBACK        __far __pascal
        /*** simple types & common helper macros ***/
        typedef short   BOOL;
        typedef unsigned char   BYTE;
        typedef unsigned short  WORD;
        typedef unsigned long   DWORD;
        typedef unsigned int    UINT;
        #ifdef STRICT
                typedef signed long     LONG;
        #else
                #define LONG  long
        #endif
        /*** common pointer types ***/
        #ifndef NULL
                #define NULL  0
        #endif
        typedef DWORD NEAR*         PDWORD;
        typedef DWORD FAR*  LPDWORD;
        typedef void FAR*       LPVOID;
        typedef short           SHORT;
        typedef short _far *    LPSHORT;
        typedef char NEAR*      PSTR;
        typedef char NEAR*      NPSTR;
        typedef char FAR*       LPSTR;
        typedef const char FAR* LPCSTR;
        typedef BYTE NEAR*      PBYTE;
        typedef BYTE FAR*       LPBYTE;
        typedef int NEAR*       PINT;
        typedef int FAR*        LPINT;
        typedef WORD  NEAR* PWORD;
        typedef WORD FAR*       LPWORD;
        typedef long NEAR*      PLONG;
        typedef long FAR*       LPLONG;
#endif
```

An example application of SM2010 DLL calls from LabWindows/CVI ® is shown below. It contains functions **measure_ohms()** and **measure_vdc()**, with sample calls to the SM-2010. 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 ohms */
int measure_ohms(double OHMreading) {
        short ret, i;
        DMMSetFnRange(0, OHMS_250);
        DMMSetAutoRange (0, TRUE);
        /* to settle auto-range and function changes ignore ten readings */
        for( i = 0 ; i < 11 ; i++ ) ret = DMMReadNorm (0, & OHMreading);
        return ret;
}
/* function: measure_vdc, purpose: measure DC Volts */
int measure_vdc(double Vreading) {
        short ret, i;
        DMMSetFnRange(0, VDC_25);
        DMMSetAutoRange (0, TRUE);
        /* to settle auto-range and function changes ignore ten readings */
        for( i = 0 ; i < 11 ; i++ ) ret = DMMReadNorm (0, &Vreading);
        return ret;
}
```

For LabWindows/CVI Version 4.01 and above, use the *Signametrics* Windows95 software release.

8.7 Windows Command Language

The following sections contain detailed descriptions of each function of the Windows command language. Those commands that pertain to only the SM-2010CT are indicated.

Note: All error codes are defined in the **dmm_user.h** file as negative integers. There are two methods to look at returned error codes. Looking them up in the **dmm_user.h** file is the most straight forward way. The second method requires calling the DMMErrString() function, which return a string describing the error code.

DMMCalibrate

Description Calibrate the DMM's analog-to-digital converter. Performs the same function as the AutoCal function on the Visual Basic Control Panel.

#include "sm2010.h"

int DMMCalibrate(int *nDmm*)

Remarks This function recalibrates 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 Integer error code. The return value is one of the following constants.

Example

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

Comments This performs an internal DMM calibration of the analog-to-digital converter. It must be performed at least once per month, with all input connections to the DMM removed. It is not related to the once per year, external calibration which stores the calibration constants in the SM-2010's internal, non-volatile memory.

DMMErrString

Description Return a pointer to a string describing a DMM error code.

#include "sm2010.h"

int DMMErrString(int *nError*, **LPSTR** *lpcBuf*, **int** *nBufLen*)

Remarks

Parameter	Type/Description
nError	int error code returned from a call to the DMM
lpcBuf	**LPSTR** Points to a buffer (at least 32 characters long) to hold the error string.
nBufLen	**int** The length of the buffer in characters

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

Value	Meaning
Integer	Length of the returned string in *lpcBuf*

Example

```
int I; char buf[48];
I = DMMErrString(nError, errtext, 48);
printf("Error: %s\n",errtext);
```

DMMGetCalDate

Description Return the calibration date string from the DMM.

```
#include "sm2010.h"

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
Any negative number	Error code

Example

```
char cBuf[32];
int string_length;
string_length = DMMGetCalDate(0, cBuf);
```

DMMGetCalLab

Description Return a pointer to a string describing the calibration laboratory that calibrated the DMM.

```
#include "sm2010.h"

int DMMGetCalLab(int nDmm, LPSTR lpszLab)
```

Remarks This function returns the name of the calibration laboratory that calibrated the DMM.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszLab	**LPTSR** Points to a buffer (at least 22 characters long) where the cal lab string is written to.

Return Value Integer length of cal lab string or an error code.

Example

```
int I; char buff[32];

I = DMMGetCalLab(0, buff);
```

DMMGetFnRange

Description

Get DMM range code

```
#include "sm2010.h"
#include "dmm_user.h"

int DMMGetFuRange(int nDmm)
```

Remarks

This function returns the combined DMM function/range code. See **dmm_user.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 the positive returned values.

Value	Meaning
VDC_250mV	First and lowest VDC range (250 mV)
VDC_2500mV	Second VDC range.
VDC_25V	Third VDC range.
VDC_250V	Highest VDC range.
VAC_250mV	First and lowest VAC range (250 mV)
VAC_2500mV	Second VAC range.
VAC_25V	Third VAC range.
VAC_250V	Highest VAC range.
IAC_2500uA	First and lowest IAC range (2.5 mA)
IAC_25mA	Second IAC range
IAC_250mA	Third IAC range
IAC_2500mA	Highest IAC range.
IDC_2500uA	First and lowest IDC range is 2.5 mA.
IDC_25mA	Second IDC range.
IDC_250mA	Third IDC range.
IDC_2500mA	Highest IDC range.
OHM_250	First Ohms range is 250 Ohms.
OHM_2500	Second Ohms range.
OHM_25K	Third Ohms range.
OHM_250K	Fourth Ohms range.
OHM_2500K	Fifth Ohms range.
OHM_25M	The highest Ohms range is 25 MOhms.

Example

```
int id;if(DMMGetFnRange == VDC_250mV)
print("Lowest VDC range selected");
```

DMMGetFunction

Description Get DMM function code

```
#include "sm2010.h"
#include "dmm_user.h"

int DMMGetFunction(int nDmm)
```

Remarks This function returns the DMM function code.

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

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

Value	Meaning
VDC	Volts DC
VAC	Volts AC
IDC	DC current
IAC	AC current
OHMS	Ohms

Example

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

DMMGetHwrVer

Description Get DMM hardware version

```
#include "sm2010.h"

int DMMGetHwrVer(int nDmm)
```

Remarks This function returns the DMM hardware version.

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

Return Value Integer value. The return value is the hardware version.

Example

```
hardwarever = DMMGetHwrVer(0);
```

DMMGetRange

Description Get DMM present range code

```
#include "sm2010.h"
#include "dmm_user.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
0	First and lowest range (250mV, 250Ohms, 2.5mA)
1	Second range (2.5V, 2.5kOhm, 25mA)
2	Third range (25V, 25kOhm, 250mA)
3	Fourth range (250V, 250kOhm, 2.5A)
4	Fifth range (2.5MOhm)
5	Sixth range (25MOhm)

Example

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

DMMGetRate

Description Get DMM reading rate

```
#include "sm2010.h"

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

Remarks This function returns a double float 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 error code.

Example

```
double FAR rate;
DMMGetRate(0, &rate);
```

DMMGetSerialNumber

Description Get DMM serial number

#include "sm2010.h"

int DMMGetSerialNumber(int *nDmm***)**

Remarks This function returns the DMM serial number. Each DMM has a unique serial number. The returned value should match the least significant digits of the serial number on the label attached to the DMM.

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

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

Example

```
int id; id = DMMGetSerialNumber(0);
```

DMMGetVer

Description Get DMM software version

#include "sm2010.h"

int DMMGetVer(double FAR **lpdVer* **)**

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

Parameter	Type/Description
lpdVer	**double FAR** pointer to the location to which the version is written.

Return Value Integer value DMM_OKAY

Example

```
double d; DMMGetVer(&d);
```

DMMHasCountOption

Description Test for the presence of the SM-2010CT Counter.

#include "sm2010.h"

BOOL DMMHasCountOption(int *nDmm*)

Remarks Test if the installed DMM is a SM-2010CT. This determination is made by detecting the presence of configuration information stored in the DMM hardware.

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, or a negative error code.

Value	Meaning
TRUE	Counter is present (It's a SM-2010CT)
FALSE	Counter is not present. (It's a SM-2010)

DMMInit

Description Initialize a DMM

#include "sm2010.h"
#include "dmm_user.h"
int DMMInit(int *nDmm*, int *base_addrs*)

Remarks Turn on the power and initialize the DMM numbered *nDmm* (where the first DMM is number 0) at the base address *base_addrs*. This function is always the first one to be issued before using the DMM. This function sets the *base_addrs* for all subsequent operations. The sequence of events upon initialization includes turning on the power to the DMM, reading the calibration data record, AutoCalibration; and setting the default measurement function (VDC), range (25V), and reading rate (10rps).

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

Return Value The return value could be one of several, as specified in "DMM_USER.H".

Example

```
/* Initialize DMM at I/O address 0x200 (512 decimal) */
I = DMMInit(0, 0x200);
```

DMMIsAutoRange

Description Get the status of the autorange flag.

#include "sm2010.h"

BOOL 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

Value	Meaning
TRUE	Autoranging mode is selected.
FALSE	Autoranging mode is not selected.

Example

```
if( DMMIsAutoRange(0)) Auto = TRUE;
```

DMMIsRelative

Description Get the status of the Relative flag.

#include "sm2010.h"

BOOL 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.

Example

```
int relstate;
relstate = DMMIsRelative(0);
```

DMMRead, DMMReadDbl

Description

Return the next floating-point reading from the DMM.

#include "sm2010.h"

int DMMRead(int *nDmm*, **float FAR *** *lpfValue*)
int DMMReadDbl(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 32-bit single-precision floating-point number in the location pointed to by *lpfResult*. **DMMReadDbl** 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*.

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

Return Value

Integer error code.

Example 1

```
float f;
int status;
status = DMMRead(0, &f);
```

Example 2

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

DMMReadingStr

Description

Return the next reading from the DMM formatted for printing.

#include "sm2010.h"

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

Remarks

This function reads the next result from the DMM, 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.

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

Integer error code.

Example

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

DMMReadFrequency (SM-2010CT only)

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

#include "sm2010.h"

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

Remarks Make a single frequency measurement, and store the result as a 64-bit double-precision floating-point number in the location pointed to by *lpdFreq*. If the DMM is in autorange, be certain to take an amplitude reading before using this command.

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

Return Value Integer error code..

Example

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

DMMReadFrequencyStr (SM-2010CT only)

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

#include "sm2010.h"

int DMMReadFrequencyStr(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 five 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.

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 Integer error code.

Example

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

DMMReadNorm

Description Take a reading that is in base value.

#include "sm2010.h"
int DMMReadNorm(int *nDmm*, **double FAR** **lpdRes*)

Remarks This function returns a double floating-point number. The returned value is corrected for base units. That is, it returns 0.20 for a 200 mV input and 1e6 for 1.0 MOhm.

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

Return Value Integer error code.

Example

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

DMMReadPeriod (SM-2010CT only)

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

#include "sm2010.h"

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

Remarks Make a single period measurement, and store the result as a 64-bit double-precision floating-point number in the location pointed to by *lpdPeriod*. If the DMM is in autorange, be certain to take an amplitude reading before using this command.

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

Return Value Integer error code.

Example

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

DMMReadPeriodStr (SM-2010CT only)

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

```
#include "sm2010.h"

int DMMReadPeriodStr(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. If the DMM is in autorange, be certain to take an amplitude reading before using this command.

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 Integer error code.

Example

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

DMMSetAutoRange

Description Set autorange status for DMM

```
#include "sm2010.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 Integer error code.

Example

```
status = DMMSetAutoRange(0, TRUE); /* enable */
status = DMMSetAutoRange(0, FALSE); /* disable */
```

DMMSetFnRange

Description Set the DMM function and range.

```
#include "sm2010.h"
#include "dmm_user.h"

int DMMSetFnRange(int nDmm, int nFnRange)
```

Remarks This function sets the function and range used by the DMM. The table of values is defined by the *VDC_**, *VAC_**, *IDC_**, *LAC_**, and *OHM_** definitions in the header file.

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

Return Value Integer error code.

Example

```
status = DMMSetFnRange(0, IDC_25mA);
```

DMMSetRate

Description Set the DMM reading rate.

```
#include "sm2010.h"
#include "dmm_user.h"

int DMMSetRate(int nDmm, int nRate)
```

Remarks This function sets the reading rate used by the DMM (expressed in readings per second). The table of values is defined by the definitions in the DLL header file.

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

Return Value Integer error code.

Example

```
status = DMMSetRate(0, DMMRate1);
```

DMMSetRelative

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

#include "sm2010.h"

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

Remarks This function selects relative or absolute reading mode for the DMM. If the *bRel* parameter value is TRUE the DMM will change to relative reading mode. 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.
bRel	**BOOL** TRUE to enter relative mode, FALSE to enter absolute mode.

Return Value Integer error code.

Example

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

DMMShutDown

Description Turn the power off to the measurement section of the DMM.

#include "sm2010.h"

int DMMShutDown(int *nDmm*)

Remarks This function turns the power off to the measurement section of the DMM, and is used whenever re-setting the DMM is necessary.

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

Return Value Integer error code.

Example

```
status = DMMShutDown(0);
```

9.0 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 SM-2010. It includes a performance test and describes the calibration procedure.

Test equipment recommended for the performance test and 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 four times more accurate than the SM-2010 specifications.

Table 9-1. Recommended Test Equipment

Instrument Type	Minimum Specifications	Recommended Model
Multi-Function Calibrator	DC Voltage Range: 0-250V Voltage Accuracy: 25 ppm AC Voltage Range: 0-250V Voltage Accuracy: 0.025% Resistance Range: 0-10 MΩ Resistance Accuracy: 70 ppm DC Current Range: 0-2.2A Current Accuracy: 0.018% AC Current Range: 50 uA - 2.2A Current Accuracy: 0.18%	Fluke 5500A Multi-Produc Calibrator

9.1 Performance Tests

This test compares the performance of the SM-2010 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 SM-2010 installed in a personal computer, with the covers on. The ambient temperature must be between 18°C to 28°C. Allow the SM-2010 to warm up one-half hour before performing any of the tests. After warm up, perform the **Auto**Cal function with all input connections removed from the DMM. (The AutoCal function should be performed at least once per month, or if power to the DMM is cycled OFF/ON.) The default reading rate of the SM-2010 should be used in each test.

9.2 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 SM-2010 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. Apply the following DC voltages to the V, Ω HI & LO terminals. Check to see that the displayed reading on the SM-2010 is within the tolerances listed.

Table 9-1. DC Voltage Test

Step	Range	Input	Minimum Reading	Maximum Reading
1	250 mV	0V (short)	-0.03 mV	0.03 mV
2	250 mV	190 mV	189.88 mV	190.12 mV
3	250 mV	-190 mV	-189.89 mV	-190.11 mV
4	2.5V	1.9 V	1.8992 V	1.9008 V
5	2.5V	-1.9 V	-1.8992 V	-1.9008 V
6	25V	19 V	18.989 V	19.011 V
7	25V	-19 V	-18.989 V	-19.011 V
8	250V	190 V	189.90 V	190.10 V
9	250V	-190 V	-189.90 V	-190.10 V

9.3 Resistance Test

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

1. If you have not done so, install the SM-2010 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 SM-2010 **V,Ω HI & LO** terminals to the calibrator HI & LO Outputs. Output 0Ω from the calibrator. Allow the SM-2010 to settle for a few seconds, short the DMM test leads together, and perform the **Relative** function. (This effectively nulls out the lead resistance of your cabling. If you are using a Fluke 5500A Calibrator, the 2-Wire Compensation feature will give a more accurate 2-wire ohms measurement. See the *Fluke 5500A Operator's Manual* for further instructions.)

3. Apply the following Resistance values to the **V, Ω HI & LO** terminals. Check to see that the displayed reading on the SM-2010 is within the tolerances listed.

Table 9-2 Resistance Test

Step	Range	Input	Minimum Reading	Maximum Reading
1	250 Ω	100 Ω	99.91 Ω	100.09 Ω
2	2.5 kΩ	1 kΩ	0.9991 kΩ	1.0009 kΩ
3	25 kΩ	10 kΩ	9.992 kΩ	10.008 kΩ
4	250 kΩ	100 kΩ	99.90 kΩ	100.10 kΩ
5	2.5 MΩ	1 MΩ	0.9978 MΩ	1.0022 MΩ
6	25 MΩ	10 MΩ	9.898 MΩ	10.102 MΩ

9.4 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 SM-2010 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. Apply the following AC voltages to the **V, Ω HI & LO** terminals. Check to see that the displayed reading on the SM-2010 is within the tolerances listed.

Table 9-3. Mid-Frequency AC Voltage Tests
All inputs are at 400 **Hz**.

Step	Range	Input	Minimum Reading	Maximum Reading
1	250 mV	10 mV	9.60 mV	10.40 mV
2	250 mV	190 mV	188.70 mV	191.30 mV
4	2.5V	0.1 V	0.0977 V	0.1023 V
5	2.5V	1.9 V	1.8870 V	1.9130 V
6	25V	1 V	0.977 V	1.023 V
7	25V	19 V	18.870 V	19.130 V
8	250V	10 V	9.77 V	10.23 V
9	250V	190 V	188.70 V	191.30 V

Table 9-4. High-Frequency AC Voltage Tests
All inputs are at 20 **kHz**.

Step	Range	Input	Minimum Reading	Maximum Reading
1	250 mV	10 mV	9.57 mV	10.43 mV
2	250 mV	190 mV	188.13 mV	191.87 mV
4	2.5V	0.1 V	0.0963 V	0.1037 V
5	2.5V	1.9 V	1.8837 V	1.9163 V
6	25V	1 V	0.963 V	1.037 V
7	25V	19 V	18.837 V	19.163 V
8	250V	10 V	9.63 V	10.37 V
9	250V	190 V	188.37 V	191.63 V

9.5 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 SM-2010 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 SM-2010 inputs. Select the DCI function. Allow the SM-2010 to settle for a few seconds, and perform the **Relative** function with no external connections to the DMM.

3. Apply the following DC currents to the **Current Input** HI & LO terminals. Check to see that the displayed reading on the SM-2010 is within the tolerances listed.

Table 9-5. DC Current Test

Step	Range	Input	Minimum Reading	Maximum Reading
1	2.5 mA	0 mA (open)	-0.0009 mA	0.0009 mA
2	2.5 mA	1 mA	0.9951 mA	1.0049 mA
3	25 mA	0 mA (open)	-0.002 mA	0.002 mA
4	25 mA	10 mA	9.968 mA	10.032 mA
5	250 mA	0 mA (open)	-0.09 mA	0.09 mA
6	250 mA	100 mA	99.61 mA	100.39 mA
7	2.5A	0 mA (open)	- 0.0002 A	0.0002 A
8	2.5A	1A	0.9948 A	1.0052 A

9.6 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 SM-2010 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 SM-2010 inputs. Select the ACI function, Autorange.

3. Apply the following AC currents to the **Current Input** HI & LO terminals. Check to see that the displayed reading on the SM-2010 is within the tolerances listed.

Table 9-6. AC Current Test
All Inputs are at 1 **kHz**

Step	Range	Input	Minimum Reading	Maximum Reading
1	2.5 mA	0.1 mA	0.0971 mA	0.1029 mA
2	2.5 mA	1 mA	0.9890 mA	1.0110 mA
3	25 mA	1 mA	0.971 mA	1.0029 mA
4	25 mA	10 mA	9.890 mA	10.110 mA
5	250 mA	10 mA	9.71 mA	10.29 mA
6	250 mA	100 mA	98.90 mA	101.10 mA
7	2.5A	100 mA	0.0971 A	0.1029 A
8	2.5A	1A	0.9890 A	1.0110 A

9.7 Frequency Counter Test (SM-2010CT only)

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

1. If you have not done so, install the SM-2010 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 HI & LO** terminals. Check to see that the displayed reading on the SM-2010 is within the tolerances listed. The signal source should be a distortion free sine wave or preferably square wave.

Table 9-7. ACV Frequency Counter Test

Step	Range	Input	Minimum Counter Reading	Maximum Counter Reading
1	250 mV	300 mV, 40 Hz	39.988 Hz	40.012 Hz
2	2.5V	1V, 40 Hz	39.992 Hz	40.008 Hz
3	25V	10V, 40 Hz	39.992 Hz	40.008 Hz
4	250V	100V, 40 Hz	39.992 Hz	40.008 Hz
5	2.50 V	2.5V, 100 kHz	99.988 kHz	100.01 kHz
6	25V	25V, 100 kHz	99.988 kHz	100.01 kHz

4. Remove the inputs from the V, Ω HI & LO terminals.

5. Select the ACI function, autorange. Turn **Freq** on.

6. Apply the following AC currents to the **Current Input HI & LO** terminals. Check to see that the displayed reading on the SM-2010 is within the tolerances listed.

Table 9-8. ACI Frequency Counter Test

Step	Range	Input	Minimum Counter Reading	Maximum Counter Reading
1	2.5 mA	330 uA, 40 Hz	39.992 Hz	40.008 Hz
2	25 mA	15 mA, 40 Hz	39.992 Hz	40.008 Hz
3	250 mA	150 mA, 40 Hz	39.992 Hz	40.008 Hz

9.8 Calibration

The SM-2010 and SM-2010CT do not have any mechanical adjustments. All calibration is done "closed case" with the covers on, utilizing calibration constants rather than adjustments. There are no serviceable parts in the unit. Under no circumstance should the covers be removed. This is very important, for both safety and proper operation of the unit.

Each SM-2010 DMM uses its own internal calibration record to ensure the accuracy of its functions and ranges. The internal calibration record includes 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".

For the AC functions, there is a third parameter that represents a digital code from 0 to 15 that controls the high frequency performance of each AC function. A large value, e.g., 15, causes the frequency response to go up.

The internal calibration record is created by performing an external calibration. The general calibration algorithm consists of applying a zero input value to the DMM followed by an input value near the top of each range. Calibration of your SM-2010 is best performed using calibration software available from Signametrics. Calibration service is also available from Signametrics for a nominal charge.

Note: The function **AutoCal** performs a System Calibration of the DMM's analog-to-digital converter, and does not write into the internal calibration record. **AutoCal** must be performed at least once per month, or if power is cycled OFF/ON to the DMM, with all input connections removed from the DMM. It does not require any external stimulus and takes less than a second to perform.

10.0 Warranty and Service

The SM-2010 is warranted for a period of one year from date of customer purchase.

If your unit requires repair or calibration, contact your Signametrics representative. There are no user serviceable parts within the SM-2010. Removal of any of the four external shields will invalidate your warranty. For in-warranty repairs, you must obtain a return materials authorization (RMA) from Signametrics prior to returning your unit.

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP