5.6 Windows Command Language

The following section contains detailed descriptions of each function of the Windows command language. Those commands that pertain to only the SM2040 are indicated. Most functions return an error code. The code can either be retrieved as a string using **DMMErrString** function, or looked up in the **SM204032.H** header file. The **UserDMM.H** file contains all the pertinent definitions for the DMM ranges functions etc. The following description for the various functions is based on "C" function declarations. Keep in mind that the Windows DLL containing these functions assumes all **int** values to be windows 32bit integers (corresponds to VisualBasic **long** type). TRUE is 1 and FALSE is 0 (which is also different from VisualBasic where True is –1 and False is 0).

DMMArmAnalogTrigger

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description

Arm DMM for analog level trigger operation.

#include "sm204032.h"

int DMMArmAnalogTrigger(**int** *nDmm*, **int** *iSamples*, **double ***_dThresh_)

Remarks

This function is usable for VDC, VAC, Ohms, IAC and IDC. Setup the SM2040 for analog level trigger operation. Following reception of this command the DMM makes measurements continuously, waiting for a value, which exceeds the threshold, *dThresh.* When this occurs, a trigger is produced with identical processing as in **DMMArmTrigger**. Threshold crossing sense is determined by the first measurement following the call of **DMMArmAnalogTrigger**. If that measurement is lower than the set threshold, *dThresh*, subsequent measurements greater than *dThresh* will trigger the DMM. If the first measurement is greater than *dThresh*, subsequent measurements smaller than *dThresh* will trigger. For example, if *dThresh* is 2.00000 V and the first reading after arming the DMM is 2.500000 V, then 1.999999 V (or smaller) will trigger the DMM. On the other hand, if *dThresh* is 1.000000 V and the first reading after arming the DMM is 0.500000 V, then 1.000001 V (or greater) will trigger the DMM.

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

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

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

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

Return Value

The return value is one of the following constants.

Value	Meaning

DMMGetMax

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get Maximum reading history.

#include "sm204032.h"

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

Remarks This function returns a double floating value that is the maximum (of the Min/Max function) value since either a function change, range change or call to the **DMMClearMinMax** function was made. This is only applicable to **Primary** read functions (those that are read using **DMMRead**, **DMMReadStr** or **DMMReadNorm**). This value is updated every time one of those functions is used.

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

Return Value Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMGetMaxStr

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Returns the maximum as a formatted string.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example

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

DMMGetMin

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get Minimum reading history.

#include "sm204032.h"

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

Remarks This function returns a double floating value that is the minimum (of the Min/Max function) value since either a function change, range change or a call to the **DMMClearMinMax**() function was made. This is only applicable to **Primary** read functions (those that are read using **DMMRead**, **DMMReadStr** or **DMMReadNorm**). This value is updated every time one of those functions is used.

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

Return Value Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMGetMinStr

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Returns the minimum as a formatted string.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example

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

DMMGetRange

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get DMM range code.

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

int DMMGetRange(int *nDmm*)

Remarks This function returns the DMM range code.

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

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

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

Example

```
int id;
```

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

DMMGetRate

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get DMM reading rate

#include "sm204032.h"

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

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

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

Return Value Integer value version code or an error code.

Value	Meaning
Negative Value	Error code

Example

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

DMMGetSourceFreq

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Get the currently set ACV Source frequency.

#include "sm204032.h"

int DMMGetSourceFreq(int *nDmm***, double *** *lpdFreq***)**

Remarks This function returns a double floating value that is the currently set ACV source frequency of the SM2044. It can be used to display or verify the default frequency of the stimulus for the various Inductance measurement ranges.

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

Return Value Integer error code..

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

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

DMMGetTCType

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get the themocouple type currently selected.

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

int DMMGetTCType(int *nDmm*)

Remarks This function returns the Themocouple type currently selected.

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

Return Value DMM type Integer or an error code.

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

Example `int TCtype = DMMGetTCType(0);`

DMMGetType

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get the type of the DMM.

#include "sm204032.h"

int DMMGetType(int *nDmm*)

Remarks This function returns the DMM type.

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

Return Value DMM type Integer or an error code.

Value	Meaning
2040	SM2040 is at nDmm slot
2042	SM2042 is at nDmm slot
2044	SM2044 is at nDmm slot
Negative Value	Error code

Example

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

DMMGetVer

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get DMM software driver version.

#include "sm204032.h"

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

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

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

Return Value Integer error code.

Value	Meaning
Negative Value	Error code

Example

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

DMMInit

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Initialize a DMM.

#include "sm204032.h"

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

Remarks This function must be the first function to be executed. It opens the driver for the specified DMM. The first DMM being 0, the second 1, etc... It also initializes the DMM hardware and does extensive self test to the DMM hardware. It then initializes the software and reads the appropriate calibration record for the respective DMM from the file specified by *lpszCal*.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszCal	**LPCSTR** Points to the name of the file containing the calibration constants for the DMM. Calibration information is normally read from the file named **SM40CAL.DAT** located in the current directory.

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

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code

Example

```
/* initialize DMM */
int i = DMMInit(0,"C:\SM40CAL.dat");      // Initialize the first DMM
```

DMMIsAutoRange

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get the status of the autorange flag.

#include "sm204032.h"

int DMMIsAutoRange(int *nDmm*)

Remarks This function returns the DMM autorange flag state.

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

Return Value TRUE, FALSE or an error code.

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

Example

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

DMMIsInitialized

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get the status of the DMM.

#include "sm204032.h"

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

Remarks This function returns the status of the DMM. If TRUE, the DMM has been initialized and is active. If FALSE the DMM is not initialized and should not be addressed. This function is used for maintenance and is not needed under normal operation.

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

Return Value TRUE, FALSE or an error code.

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

Example

```
int active = DMMIsInitialzied(0);
```

DMMIsRelative

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Get the status of the Relative flag.

#include "sm204032.h"

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

Remarks This function returns the DMM Relative flag state.

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

Return Value Integer TRUE, FALSE or an error code.

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

Example

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

DMMLoadCalFile

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Reload calibration record from file.

#include "sm204032.h"

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

Remarks This function is provides the capability to reload the calibration record. This is useful in making limited calibration adjustments to the DMM. By having a copy of the original calibration file '**SM40CAL.DAT**' open with an editor, and modifying calibration entries, then reloading it using **DMMLoadCalFile**, one can instantly verify the corrections made. Make sure the '**SM40CAL.DAT**' file itself is not altered since that will void the calibration.

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
lpszCal	**LPCSTR** Points to the name of the file containing the calibration constants for the DMM.

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

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

Example

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

DMMOpenPCI

SM2040 ☑ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

int DMMOpenPCI(**int** *nDmm*)

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMOpenCalACCaps

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Calibrate the AC based in circuit capacitance function.

#include "sm204032.h"

int DMMOpenCalACCapsl(int *nDmm*)

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMOpenTerminalCal

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Calibrate the Inductance measurement function with open terminals.

#include "sm204032.h"

int DMMOpenTerminalCal(int *nDmm*)

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMPeriodStr

SM2040 ☐ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

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

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

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

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

Example

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

DMMPolledRead

SM2040 ☑ SM2040 ☑ SM2044 ☑

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

#include "sm204032.h"

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

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

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

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

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

Example

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

DMMPolledReadCmd

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Send DMM Polled Read command.

#include "sm204032.h"

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

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

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

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

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

Example

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

DMMPolledReadStr

SM2040 ☑ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

Remarks This function is the string version of **DMMPolledRead**. See **DMMPolledRead** for more details.

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

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

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

Example

```
char strMsg[64];
if(DMMPolledReadStr(0, strMsg)) MessageBox(0,strMsg,
"SM2044",MB_OK);  // display readings;
```

DMMRead

SM2040 ☑ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

Remarks **DMMRead** reads the next result from the DMM, performs all scaling and conversion required, and returns the result as a 64-bit double-precision floating-point number in the location pointed to by *lpdResult*. It can read all the **Primary** functions (those that can be selected using **DMMSetFunction()** and **DMMSetRange()**). Returned result is a scaled value which is normilized to the selected range. That is . That is, it returns 300 for 300mV input in the 330 mV range, and 100 for 100 kΩ input in the 330k Ω range. Use the **DMMReadNorm()** function for base units read function.

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

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

Value	Meaning
DMM_OKAY	DMM initialized successfully.
Negative Value	Error code
DMM_E_RANGE	DMM over range error occurred.

Example

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

DMMReadBuffer

SM2040 ☑ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error Code

Example

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

DMMReadBufferStr

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Return the next reading, formatted for printing.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
char Buf[64];
DMMArmTrigger(0,1);     // take a single triggered sample
while( !DMMReady(0));
DMMReadBufferStr(0, Buf);
```

DMMReadCJTemp

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Read cold junction temperature for thermocouple measurement.

#include "sm204032.h"

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

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

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

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

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

Example

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

DMMReadCrestFactor

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Return ACV signal's Crest Factor.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMReadDutyCycle

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Return percent duty cycle of ACV signal.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMReadFrequency

SM2040 ☐ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
DMM_E_INIT	DMM is uninitialized. Must be initialize prior to using any function.
DMM_E_DMM	Invalid DMM number.

Example

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

DMMReadInductorQ

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Return inductor's Q value.

#include "sm204032.h"

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

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

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

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

Value	Meaning

DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMReadMeasurement

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description

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

#include "sm204032.h"

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

Remarks

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

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

Return Value Integer value version code or an error code.

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

Example

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

DMMReadMedian

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Return ACV signal's Median value.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMReadNorm

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Take a reading that is in base value.

#include "sm204032.h"

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

Remarks This **Primary** read function is similar to **DMMRead()**. It returns a double floating-point reading. The returned value is corrected for base units. That is, it returns 0.3 for a 300 mV input and 1e6 for 1.0 MOhm.

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

Return Value Integer value version code or an error code.

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

Example

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

DMMReadPeakToPeak

SM2040 ☐ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMReadPeriod

SM2040 ☐ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMReadStr

SM2040 ☑ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

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

Parameter	Type/Description

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

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

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

Example

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

DMMReadTotalizer

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Read the totalized value accumulated by the Totalizer function.

#include "sm204032.h"

long DMMReadTotalizer(int *nDmm*)

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

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

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

Value	Meaning
Negative Value	Error code

Example

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

DMMReadWidth

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Return the positive and negative pulse widths.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMReady

SM2040 ☑ SM2042 ☑ SM2044 ☑

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

#include "sm204032.h"

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

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

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

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

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

Example

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

DMMSetACCapsDelay

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Set the measurement delay of AC based Capacitance.

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

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetACCapsLevel

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Set the DCV source output level.

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

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

```
DMMSetACCapsLevel(0, 0.35); // Set stimulus to 0.35V peak
```

DMMSetACVSource

SM2040 ☐ SM2042 ☐ SM2042 ☑

Description Set the ACV source output level and frequency.

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

int DMMSetACVSource(int *nDmm***, double** *ldVolts***, double** *ldFreq***)**

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetAutoRange

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Enable/Disable autorange operation of DMM

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example

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

DMMSetBuffTrigRead

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Setup the DMM for Triggered operation.

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

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

Remarks

Setup the SM2040 for external hardware trigger operation. Following reception of this command the DMM enters a wait state. After reception of an external trigger edge of *iEdge* polarity, the DMM takes *iSettle* + 1 readings at the set measurement function, range, and reading rate; and stores the last reading in the in an internal buffer. This process is repeated for *iSamples*. This function is particularly useful in conjunction with a triggering instruments such as the SM4042 relay scanner. No autoranging, function or ranges changes allowed while the DMM is waiting for triggers. The number of trigger edges must be equal or greater than *iSamples* to properly terminate this mode. Any trigger received following iSamples is ignored. Between the time this command is issued and the time the buffer is read, no other command should be sent to the DMM with the exception of **DMMDisarmTrigger** command, which terminates this mode, and **DMMReady** which monitors readyness. This function is usable for VDC, VAC, Ohms, IAC, IDC and RTD measurements.

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

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

Return Value

The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Operation successfully terminated
Negative Value	Error code.

Example

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

DMMSetCapsAveSamp

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description

Tunes the capacitance measurement function parameters for higher measurement speed.

#include "sm204032.h"

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

Remarks This function should be used carefully since it modifies the capacitance function basic measurement parameters; the averages value, *iAverage*, and the number of points sampled, *iSamples*. This function is provided only for cases where it is necessary to improve measurement speed. When using this function keep in mind that the accuracy specification provided for capacitance is not guaranteed. Also, modifying these values could have profound efect on the operation of the function. Any time a capacitance range is change, these values are set to the default values. For instance, values of 1 and 3 for *iAverage*, and *iSamples* will reduce measurement time on the 12nF range from 0.8s to about 50ms.

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

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

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

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

DMMSetCJTemp

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Set cold junction temperature for thermocouple measurement.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Operation successfully terminated

Negative Value Error code.

Example

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

DMMSetCompThreshold

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Set the Threshold DAC level.

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

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetCounterRng

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Set the frequency counter to a specific range.

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

int DMMSetCounterRng(int *nDmm***, int** *fRange***)**

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetDCISource

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Set the DCI source output level.

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

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetDCVSource

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Set the DCV source output level.

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

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetExternalShunt

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Set the value of the leakage function external shunt

#include "sm204032.h"

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

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

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

	ldShunt	**double** Shunt resistance value. A value greater than zero and smaller the 200e6 (200 Mega Ohms) is allowed.

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetFuncRange

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Set the DMM function and range.

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

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

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

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

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

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

Example

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

DMMSetFunction

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Set the DMM function.

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

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

Remarks This function sets the function used by the DMM. The table of values is defined by the *VDC*, *VAC*, *IDC*, *IAC*, *OHMS2W*, *OHMS4W* … definitions in the DLL header file. Not all functions are available for all DMM types. For instance the SM2042 has Capacitance while the SM2040 does not.

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

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

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

Example

```
status = DMMSetFunction(0, INDUCTANCE);
```

DMMSetInductFreq

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Set the frequency of the Inductance Source.

#include "sm204032.h"

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetOffsetOhms

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description

Enable/Disable Offset Ohms operation

#include "sm204032.h"

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

Remarks

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

Parameter	Type/Description
nDmm	**int** Identifies the DMM. DMMs are numbered starting with zero.
bState	**BOOL** Determines whether or not Offset Ohms is enabled. The value TRUE enables, FALSE disables it.

Return Value

The return value is one of the following constants.

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example

```
status = DMMSetOffsetOhms(0, TRUE); /* enable OffsetOhms */
```

DMMSetRange

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description

Set the DMM range for the present function.

#include "sm204032.h"

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

Remarks

This function sets the range used by the DMM for the present function. The table of values is defined by the *_330mV, _3mA, etc.* definitions in the DLL header file. Not all ranges are available for all DMM types. For instance the SM2042 has a 33 Ohms range, and the SM2040 does not.

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

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

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

Example

```
status = DMMSetRange(0, _330mA);
```

DMMSetRate

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Set the DMM range for the present function.

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

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

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

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

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

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

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

Example

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

DMMSetRelative

SM2040 ☑ SM2042 ☑ SM2044

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

#include "sm204032.h"

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

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

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

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

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

Example

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

DMMSetResistance

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Set the resistance value to be synthesized

#include "sm204032.h"

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetRTD

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Set the RTD parameters.

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

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.

Negative Value Error code

Example

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

DMMSetSensorParams

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Set the cold junction temperature sensor equation parameters.

#include "sm204032.h"

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

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

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

Return Value Integer error code.

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

Example

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

DMMSetSourceMode

SM2040 ☐ SM2042 ☐ SM2044 ☑

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

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

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

Remarks This **Secondary** function sets the DC and AC voltage sources to either **OPEN_LOOP** or **CLOSED_LOOP**. In **CLOSED_LOOP** the sources use the main 12 bit source DAC. In **CLOSED_LOOP** the Trim DAC is also used, which augments the 12 bit DAC to produce 16 effective bits. Open loop updates are very quick. In ClosedLoop mode the source level is adjusted every time the DMM is read, making small corrections until the

reading is equal to the set voltage. However, for the ClosedLoop mode to update the source level, it is necessary to read the DMM multiple times. See **DMMSetDCVSource** and **DMMSetACVSource** for more details.

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMSetSynchronized

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Enable or disable Synchroneous operation of the DMM.

#include "sm204032.h"

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

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

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

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

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example

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

DMMSetTCType

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Set Thermocouple type.

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

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

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

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

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

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example

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

DMMSetTempUnits

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Set temperature units to °C or °F.

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

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

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

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

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

Value	Meaning
DMM_OKAY	Function succeeded.
Negative Value	Error code

Example

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

DMMSetTrigRead

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description

Setup the DMM for mutiple Triggered readings operation.

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

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

Remarks

Setup the SM2040 for external hardware trigger operation. Following reception of this command the DMM enters a wait state. After reception of an external trigger edge of *iEdge* polarity, the DMM takes *iSettle* + 1 readings at the set measurement function, range, and reading rate; and sends the last reading to the PC. This process is repeated for *iSamples*. *iSamples* Trigger pulses must be issued to complete this process. This function is particularly useful in conjunction with triggering instruments such as the SM4042 relay scanner. No autoranging is allowed in this mode. The number of trigger edges must be equal or greater than *iSamples* to properly terminate this mode. Following the issue of the **DMMSetTrigRead** command, and until the sampling process ends, it is necessary to read the samples from the DMM using the **DMMReadMeasurement** command. This will prevent an Overrun communication error. In other words, the rate at which measurement are read must keep up with the DMM transmission of readings. The DMM has a built in 5 readings FIFO to help with this problem. This function is usable for VDC, VAC, Ohms, IAC, IDC and RTD measurements. Use the **DMMReadMeasurement** to monitor for data availability, and to read this data.

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

Return Value

The return value is one of the following constants.

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

Example

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

DMMSetTrimDAC

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description Set the Trim DAC level.

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

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

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

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMStartTotalizer

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Clear the totalized value and start the totalizer.

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

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

Remarks This is a **Secondary** function and the DMM must be in ACV measurement function, and a valid range must be selected. This function clears the Totalized count, sets the edge sense, and starts the Totalizer. The totalized value can be read during the accumulation period. However, it could affect the count by the interruption. If no reads are performed during accumulation, the input rate can be as high as 45 kHz. If reads are performed during the accumulation period, this rate could be as low as 20 kHz. The Threshold DAC sets the levels at which signals are counted. During accumulation, no other command (except **DMMReadTotalizer**) should be used. When done, this function must be turned off using **DMMStopTotalizer**. After the Totalizer is stopped, the accumulated result can

be read using **DMMReadTotalizer**. A normal procedure would be to set the DMM to the ACV function, select voltage range, set the Threshold DAC, start the totalizer, wait for the time required, stop and read the total. The total number of events is limited to 1,000,000,000. The SM2044S product allows up to 90 kHz input, but reduces the resolution of the count.

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

Return Value Integer error code.

Value	Meaning
DMM_OKAY	Operation successfully completed.
Negative Value	Error code

Example

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

DMMStopTotalizer

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description Terminate the accumulation process of the Totalizer.

#include "sm204032.h"

int DMMStopTotalizer(int *nDmm*)

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

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

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

Value	Meaning
DMM_OKAY	Operation was successful.
Negative Value	Error code

Example

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

DMMTerminate

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Terminate DMM operation (DLL)

#include "sm204032.h"

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

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

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

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

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

Example

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

DMMTrigger

SM2040 ☑ SM2042 ☑ SM2044 ☑

Description Software Trigger the DMM. Take *iSamples*.

#include "sm204032.h"

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

Remarks Following reception of this command, the SM2040 DMM makes *iSamples* readings at the currently set function, range and rate, and stores them in an internal buffer. Rate can be set between 10 to 1000 readings per second. No autoranging is allowed for this trigger operation. Between the times the **DMMTrigger** command is issued and the time the buffer is read, no other command should be sent to the DMM. Use the **DMMReady** function to monitor when the DMM is ready (ready implies completion of *iSamples*). When ready, you can optionally read a single reading or up to *iSamples*, using **DMMReadBuffer.**

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

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

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

DMM_E_DMM	Invalid DMM number.

Example

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

DMMWidthStr

SM2040 ☐ SM2042 ☑ SM2044 ☑

Description

Return positive and negative pulse width in string format.

#include "sm204032.h"

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

Remarks

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

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

Return Value

The return value is one of the following constants.

Value	**Meaning**
DMM_OKAY	Valid return.
Negative Value	Error code

Example

```
char P[64],N[64]; int status = DMMWidthStr(0,P,N);
```

SetACCapsFreq

SM2040 ☐ SM2042 ☐ SM2044 ☑

Description

Sets the frequency of the AC capacitance source.

#include "sm204032.h"

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

Remarks

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

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

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

Value	Meaning
DMM_OKAY	Valid return.
Negative Value	Error code

Example

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

6.0 Maintenance

Warning

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

This section presents maintenance information for the DMM.

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

Table 9-1. Recommended Test Equipment

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

6.1 Performance Tests

This test compares the performance of the SM2040 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 SM2040 installed in a personal computer, with the covers on. The ambient temperature must be between 18°C to 28°C. Allow the SM2040 to warm up at least one-half hour before performing any of the tests. The default reading rate of the SM2040 should be used in each test.

6.2 DC Voltage Test

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

1. If you have not done so, install the SM2040 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 a high quality copper wire short to the SM2040 **V,Ω + & -** inputs. Select the DCV function, Autorange. Allow the SM2040 to settle for several seconds, and perform the **Relative** function.

3. Apply the following DC voltages to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SM2040 is within the indicated range.

Table 9-2. DC Voltage Test

Step	Range	Input	Minimum Reading	Maximum Reading
1	330 mV	0V (short)	-8 μV	+8 μV
2	330 mV	190 mV	189.9787 mV	190.0213 mV
3	330 mV	-190 mV	-190.0213 mV	–189.9787 mV
4	3.3 V	1.9 V	1.899898 V	1.900103 V
5	3.3 V	-1.9 V	-1.900103 V	-1.899898 V
6	33 V	19 V	18.99834 V	19.00166 V
7	33 V	-19 V	-19.00166 V	-18.99834 V
8	330 V	190 V	189.9833 V	190.0167 V
9	330 V	-190 V	-190.0167 V	-189.9833 V

6.3 Resistance Test, 2-wire

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

1. If you have not done so, install the SM2040 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 SM2040 **V,Ω** + **& -** terminals to the calibrator HI & LO Outputs. Output 0 Ω from the calibrator. Allow the SM2040 to settle for a few seconds, and perform the **Relative** function. (This effectively nulls out the lead resistance of your cabling. If you are using a Fluke 5700A or 5520A Calibrator, the 2-wire Compensation feature will give a more accurate 2-wire ohms measurement. See the *Fluke Operator's Manual* for further instructions.)

3. Apply the following Resistance values to the **V, Ω + & -** terminals . Check to see that the displayed reading on the SM2040 is within the indicated range.

Table 9-3 Resistance Test, 2-wire

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

[1] SM2044 only

6.4 Resistance Test, 4-wire

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

1. If you have not done so, install the SM2040 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 SM2040 **V,Ω + & -** terminals to the calibrator HI & LO Output. Connect the SM2040 **I, 4WΩ + & -** terminals to the HI & LO Sense terminals.

3. Select the 4WΩ function on the SM2040, Autorange. Set the calibrator to 0 Ω. Be certain that the calibrator is set to external sense ("EX SNS" on the Fluke 5700A or "4-Wire Comp" on the 5520A). Allow the SM2040 to settle for a few seconds, and perform the **Relative** function.

4. Apply the following Resistance values to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SM2040 is within the indicated range.

Table 9-4 Resistance Test, 4-wire

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

[1] SM2044 only.
Note: The use of 4-wire Ohms for resistance values above 300 kΩ is not recommended.

6.5 AC Voltage Test

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

1. If you have not done so, install the SM2040 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 + & -** terminals. Check to see that the displayed reading on the SMX2040 is within the indicated readings range.

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

Step	Range	Input	Minimum Reading	Maximum reading
1	330 mV	10 mV	9.8650 mV	10.1350 mV
2	330 mV	190 mV	189.5950 mV	190.4050 mV
4	3.3 V	100 mV	0.098735 V	0.101265 V
5	3.3 V	1.9 V	1.897565 V	1.902435 V
6	33 V	1 V	0.98327 V	1.01673 V
7	33 V	19 V	18.97313 V	19.02687 V
8	250 V	10 V	9.864 V	10.136 V
9	250 V	190 V	189.756 V	190.244 V

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

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

6.6 DC Current Test

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

1. If you have not done so, install the SM2040 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 SM2040 inputs. Select the DCI function, Autorange. Allow the SM2040 to settle for a few seconds, and perform the **Relative** function.

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

Table 9-7. DC Current Test

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

6.7 AC Current Test

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

1. If you have not done so, install the SM2040 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 SM2040 inputs. Select the ACI function, Autorange.

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

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

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

6.8 Capacitance Test (SM2042, SM2044 only)

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

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

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

3. Apply the following Capacitance values to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SM2042/44 is within the indicated range of readings.

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

6.9 Frequency Counter Test (SM2042, SM2044 only)

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

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

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

3. Apply the following AC voltages to the **V, Ω + & -** terminals. Check to see that the displayed reading on the SM2042/44 is within the indicated range of readings.

Table 9-9. ACV Frequency Counter Test

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

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

3. Apply the following AC currents to the **I,4Ω + & -** terminals. Check to see that the displayed reading on the SM2040 is within the tolerance appropriate for your application (e.g. 90 day or 1 year accuracy).

Table 9-10. ACI Frequency Counter Test

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

6.10 Calibration

Each SM2040 DMM uses its own **SM40CAL.DAT** calibration file to ensure the accuracy of its functions and ranges. The **SM40CAL.DAT** file is a text file that contains the DMM identification number, calibration date, and calibration constants for all DMM ranges. For most functions, the calibration constants are scale factor and offset terms that solve the "y = mx + b" equation for each range An input "x" is corrected using a scale factor term "m" and an offset term "b"; this gives the desired DMM reading, "y". Keep in mind that for ranges and functions that are unavailable for a particular product in the SM2040 family, the calibration record contains a place-holder. An example **SM40CAL.DAT** is shown:

```
card_id  10123    type 2044 calibration_date 06/15/1999
ad       ; A/D compensation
72.0     20.0
vdc      ; VDC 330mV, 3.3V, 33V, 330V ranges, offset and gain parameters
-386.0   0.99961
-37.0    0.999991
-83.0    0.999795
-8.8     1.00015
vac      ; VAC 1st line - DC offset. Than offset, gain and freq each range 330mV to 250V
5.303
0.84     1.015461          23
0.0043   1.0256            23
0.0      1.02205           0
0.0      1.031386          0
idc      ; IDC 3.3mA to 2.5A ranges, offset and gain
-1450.0  1.00103
-176.0   1.00602
-1450.0  1.00482
-176.0   1.00001
```

```
iac       ; IAC 3.3mA to 2.5A ranges, offset and gain
1.6       1.02402
0.0       1.03357
1.69      1.00513
0.0       1.0142
2w-ohm    ; Ohms 33, 330, 3.3k,33k,330k,3.3M,33M,330Meg ranges, offset and gain
1.27e+4   1.002259
1256.0    1.002307
110.0     1.002665
0.0       1.006304
0.0       1.003066
0.0       1.001848
0.0       0.995664
0.0       1.00030
```

The first column under any function, e.g.,"`vdc`", is the offset term "b", expressed as a value proportional to analog-to-digital (a/d) counts. The second column is the scale factor term "m". Within each function, the "b" and "m" terms are listed with the lowest range at the beginning. For example, under "`2w-ohm`" above, "`1.27e+4 1.002259`" represents the offset term for the 33 Ω range, and "`1.002259`" is the scale factor for this range. This record must be for the SM2042 or SM2044 since the SM2040 does not have the 33 Ohms range, and therefore these values will be set to 0.0 and 1.0.

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

The **SM40CAL.DAT** file is created by performing external calibration. The general calibration algorithm consists of applying a zero value to the DMM followed by a value of 2/3rd of the top of each range. Calibration of your SM2040 is best performed using calibration software available from Signametrics.

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

7.0 Warranty and Service

The SM2040 is warranted for a period of one year from date of purchase. The warrantee does not include calibration.

If your unit requires repair or calibration, contact your Signametrics representative. There are no user serviceable parts within the SM2040. Removal of any of the three external shields will invalidate your warranty. For in-warranty repairs, you must obtain a return authorization from Signametrics prior to returning your unit.

8.0 Accessories

Several accessories are available for the SM2040 DMMs, which can be purchased directly from Signametrics, or one of its distributors or representatives. These include:

- Basic DMM probes
- DMM probe kit
- Deluxe DMM probe set
- Shielded SMT Tweezer Probes
- Multi Stacking Double Banana shielded cable 36”
- Multi Stacking Double Banana shielded cable 48”
- Mini DIN-7 Trigger, 6-Wire Ohms connector
- 4-Wire Kelvin probes

`