

NI-DMM Function Reference Help

June 2008, 372116C-01

Expand this topic to view the C/CVI/VB functions and attributes included with NI-DMM that you can use to configure and operate your digital multimeter.

To navigate this help file, use the **Contents**, **Index**, and **Search** tabs to the left of this window.

To comment on this documentation, refer to the [National Instruments Web site](#).

© 2004–2008 National Instruments Corporation. All rights reserved.

niDMM_init

ViStatus = niDMM_init(ViString Resource_Name, ViBoolean ID_Query,
ViBoolean Reset_Device, ViSession* Instrument_Handle)

Purpose

This function completes the following tasks:

- Creates a new IVI instrument driver session.
- Opens a session to the device you specify for the **Resource_Name** parameter.
- If the **ID_Query** parameter is set to VI_TRUE (1), this function queries the instrument ID and checks that it is valid for this instrument driver.
- If the **Reset_Device** parameter is set to VI_TRUE (1), this function resets the instrument to a known state. Sends initialization commands to set the instrument to the state necessary for the operation of the instrument driver.
- Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Parameters

Input

| Name | Type | Description |
|----------------------|----------|---|
| Resource_Name | ViString | Contains the Resource_Name of the device to initialize. The Resource_Name is assigned in Measurement & Automation Explorer (MAX). Refer to Related Documentation for the <i>NI Digital Multimeters Getting Started Guide</i> for more information about configuring and testing the DMM in MAX. |

Valid Syntax:

- NI-DAQmx name
- DAQ::NI-DAQmx name[::INSTR]
- DAQ::Traditional NI-DAQ device number[::INSTR]
- IVI logical name



Caution All IVI names for the **Resource_Name**, such as logical names or virtual names, are case-sensitive. If you use logical names, driver session names, or virtual names in your program, you must make sure that the name you use matches the name in the IVI Configuration Store file exactly, without any variations in the case of the characters in the name.

| | | |
|-----------------|-----------|--|
| ID_Query | ViBoolean | Verifies that the device you initialize is one that the driver supports. NI-DMM automatically performs this query, so setting this parameter is not necessary. |
|-----------------|-----------|--|

Defined Values:

| | | |
|-------------------|---|------------------|
| VI_TRUE (default) | 1 | Perform ID Query |
| VI_FALSE | 0 | Skip ID Query |

| | | |
|---------------------|-----------|--|
| Reset_Device | ViBoolean | Specifies whether to reset the instrument during the initialization procedure. |
|---------------------|-----------|--|

Defined Values:

| | | |
|-------------------|---|--------------|
| VI_TRUE (default) | 1 | Reset Device |
| VI_FALSE | 0 | Don't Reset |

Output

| Name | Type | Description |
|--------------------------|-------------|--|
| Instrument_Handle | ViSession* | Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_InitWithOptions

```
ViStatus = niDMM_InitWithOptions(ViString Resource_Name, ViBoolean  
    ID_Query, ViBoolean Reset_Device, ViString Option_String,  
    ViSession* Instrument_Handle)
```

Purpose

This function completes the following tasks:

- Creates a new IVI instrument driver session and, optionally, sets the initial state of the following session attributes: [RangeCheck](#), [QueryInstrStatus](#), [Cache](#), [Simulate](#), [Recordcoercions](#).
- Opens a session to the device you specify for the **Resource_Name** parameter. If the **ID_Query** parameter is set to VI_TRUE, this function queries the instrument ID and checks that it is valid for this instrument driver.
- If the **Reset_Device** parameter is set to VI_TRUE, this function resets the instrument to a known state. Sends initialization commands to set the instrument to the state necessary for the operation of the instrument driver.
- Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Parameters

Input

| Name | Type | Description |
|----------------------|----------|--|
| Resource_Name | ViString | Contains the Resource_Name of the device to initialize. The Resource Measurement & Automation Explorer (MAX). Refer to Related Documents <i>Multimeters Getting Started Guide</i> for more information about configuring |

Valid Syntax:

- NI-DAQmx name
- DAQ::NI-DAQmx name[::INSTR]
- DAQ::Traditional NI-DAQ device number[::INSTR]
- IVI logical name



Caution All IVI names for the **Resource_Name** names or virtual names, are case-sensitive names, driver session names, or virtual names you must make sure that the name you use the IVI Configuration Store file exactly, with case of the characters in the name.

| | | |
|-----------------|-----------|--|
| ID_Query | ViBoolean | Verifies that the device you initialize is one that the driver supports. NI-query, so setting this parameter is not necessary. |
|-----------------|-----------|--|

Defined Values:

| | | |
|-------------------|---|------------------|
| VI_TRUE (default) | 1 | Perform ID Query |
| VI_FALSE | 0 | Skip ID Query |

| | | |
|---------------------|-----------|---|
| Reset_Device | ViBoolean | Specifies whether to reset the instrument during the initialization process |
|---------------------|-----------|---|

Defined Values:

| | | |
|-------------------|---|--------------|
| VI_TRUE (default) | 1 | Reset Device |
| VI_FALSE | 0 | Don't Reset |

| | | |
|----------------------|----------|---|
| Option_String | ViString | Sets the initial value of certain attributes for the session. The following attribute constant, and default value for each attribute that you can use |
|----------------------|----------|---|

| | |
|------------------|-----------------------------|
| Check | NIDMM_ATTR_RANGE_CHECK |
| QueryInstrStatus | NIDMM_ATTR_QUERY_INSTRUMENT |
| Cache | NIDMM_ATTR_CACHE |
| Simulate | NIDMM_ATTR_SIMULATE |

| | |
|-----------------|------------------------|
| RecordCoercions | NIDMM_ATTR_RECORD_COE |
| DriverSetup | NIDMM_ATTR_DRIVER_SETU |

The format of this string is, "AttributeName=Value." To set multiple attrib with a comma.

If you pass NULL or an empty string for this parameter, the session use attributes. You can override the default values by assigning a value ex parameter. You do not have to specify all of the attributes and may lea use the default value).

Refer to [Simulating NI Digital Multimeters](#) for more information.

Output

| Name | Type | Description |
|--------------------------|------------|---|
| Instrument_Handle | ViSession* | Returns a ViSession handle that you use to identify the instrument in a function calls. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_close

ViStatus = niDMM_close(ViSession Instrument_Handle)

Purpose

Closes the specified session and deallocates resources that it reserved.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureMeasurementDigits

```
ViStatus = niDMM_ConfigureMeasurementDigits(ViSession  
Instrument_Handle, ViInt32 Measurement_Function, ViReal64  
Range, ViReal64 Resolution_Digits)
```


Purpose

Configures the common attributes of the measurement. These attributes include NIDMM_ATTR_FUNCTION, NIDMM_ATTR_RANGE, and NIDMM_ATTR_RESOLUTION_DIGITS.

Parameters

Input

| Name | Type | Description |
|-----------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session parameter from niDMM_init or niDMM |
| Measurement_Function | ViInt32 | Specifies the Measurement_Function driver sets NIDMM_ATTR_FUNCTION |
| Range | ViReal64 | Specifies the range for the function parameter. When frequency is specified you must supply the minimum frequency. For example, you must type in 100 Hz if |

For all other functions, you must supply a range for measuring. For example, you must supply a range for voltage. Values are coerced up to the closest valid range. The driver sets a list of valid ranges. The driver sets a default range. The default is 0.02 V.



Note The NI 4050, NI 4060, and NI 4070 trigger and sample trigger are

| |
|--------------------------|
| NIDMM_VAL_AUTO_RANGE_ON |
| NIDMM_VAL_AUTO_RANGE_OFF |
| NIDMM_VAL_AUTO_RANGE_ONC |

Resolution_Digits

ViReal64 Specifies the resolution of the meas
[Overview](#) for a list of valid ranges. T
[NIDMM_ATTR_RESOLUTION_DIGI](#)
ignored when the **Range** parameter
(-1.0) or NIDMM_VAL_AUTO_RAN



Note NI-DMM ignores this pa
measurements on the NI 4072
measurements, use the [NIDM](#)
attribute.

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureMeasurementAbsolute

ViStatus = niDMM_ConfigureMeasurementAbsolute(ViSession
Instrument_Handle, ViInt32 Measurement_Function, ViReal64
Range, ViReal64 Resolution_Absolute)

Purpose

Configures the common attributes of the measurement. These attributes include NIDMM_ATTR_FUNCTION, NIDMM_ATTR_RANGE, and NIDMM_ATTR_RESOLUTION_ABSOLUTE.

Parameters

Input

| Name | Type | Description |
|-----------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session parameter from niDMM_init or niDM |
| Measurement_Function | ViInt32 | Specifies the Measurement_Function driver sets NIDMM_ATTR_FUNCTION |
| Range | ViReal64 | Specifies the Range for the function parameter. When frequency is specified you must supply the minimum frequency. For example, you must type in 100 Hz if |

For all other functions, you must supply the range you are measuring. For example, you must supply the **Range** values are coerced up to the maximum value. [Overview](#) for a list of valid ranges. The default value. The default is 0.02 V.



Note The NI 4050, NI 4060, and NI 4070 trigger and sample trigger are

NIDMM_VAL_AUTO_RANGE_ON

NIDMM_VAL_AUTO_RANGE_OFF

NIDMM_VAL_AUTO_RANGE_ONC

Resolution_Absolute

ViReal64

Specifies the absolute resolution for [NIDMM_ATTR_RESOLUTION_ABS](#) when the **Range** parameter is set to `NIDMM_VAL_AUTO_RANGE_ONC`



Note NI-DMM ignores this parameter for measurements on the NI 4072. For other measurements, use the [NIDMM_ATTR_RESOLUTION_ABS](#) attribute.

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureMultiPoint

```
ViStatus = niDMM_ConfigureMultiPoint(ViSession Instrument_Handle, ViInt32  
    Trigger_Count, ViInt32 Sample_Count, ViInt32 Sample_Trigger,  
    ViReal64 Sample_Interval)
```


Purpose

Configures the attributes for multipoint measurements. These attributes include [NIDMM_ATTR_TRIGGER_COUNT](#), [NIDMM_ATTR_SAMPLE_COUNT](#), [NIDMM_ATTR_SAMPLE_TRIGGER](#), and [NIDMM_ATTR_SAMPLE_TRIGGER](#).

For continuous acquisitions, set [NIDMM_ATTR_TRIGGER_COUNT](#) or [NIDMM_ATTR_SAMPLE_COUNT](#) to zero. For more information, refer to [Multiple Point Acquisitions](#), [Triggering](#), and [Using Switches](#).

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Trigger_Count | ViInt32 | Sets the number of triggers you want the DMM to receive before returning to the Idle state. The driver sets NIDMM_ATTR_TRIGGER_COUNT to this value. The default value is 1. |
| Sample_Count | ViInt32 | Sets the number of measurements the DMM makes in each measurement sequence initiated by a trigger. The driver sets NIDMM_ATTR_SAMPLE_COUNT to this value. The default value is 1. |
| Sample_Trigger | ViInt32 | Specifies the Sample_Trigger source you want to use. The driver sets NIDMM_ATTR_SAMPLE_TRIGGER to this value. The default is Immediate. |
| | |  Note To determine which values are supported by each device, refer to the LabWindows/CVI Trigger Routing section. |
| Sample_Interval | ViReal64 | Sets the amount of time in seconds the DMM waits between measurements. The driver sets NIDMM_ATTR_SAMPLE_INTERVAL to this value. Specify a sample interval to add settling time between measurements or to decrease the |

measurement rate. **Sample_Interval** only applies when the **Sample_Trigger** is set to INTERVAL.

On the NI 4060, the **Sample_Interval** value is used as the settling time. When sample interval is set to 0, the DMM does not settle between measurements. The NI 4065 and NI 4070/4071/4072 use the value specified in **Sample_Interval** as additional delay. The default value (-1) ensures that the DMM settles for a recommended time. This is the same as using an Immediate trigger.



Note This attribute is not used on the NI 4050.

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureWaveformAcquisition

ViStatus = niDMM_ConfigureWaveformAcquisition(ViSession
Instrument_Handle, ViInt32 Measurement_Function, ViReal64
Range, ViReal64 Rate, ViInt32 Waveform_Points)

Purpose

Configures the NI 4070/4071/4072 for waveform acquisitions.

Parameters

Input

| Name | Type | Description |
|-----------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument ses Instrument_Handle parameter from niDMM_InitWithOptions . The default |
| Measurement_Function | ViInt32 | Specifies the Measurement_Funct waveform acquisition. The driver se NIDMM_ATTR_FUNCTION to this v <div style="border: 1px solid black; padding: 5px;"> <p>NIDMM_VAL_WAVEFORM_VOLTAGE (default)</p> <p>NIDMM_VAL_WAVEFORM_CURR</p> </div> |
| Range | ViReal64 | Specifies the expected maximum ar signal and sets the Range for the Measurement_Function . NI-DMM NIDMM_ATTR_RANGE to this value coerced up to the closest input Ran 10.0. For valid ranges refer to the topics i Auto-ranging is not supported during acquisitions. |
| Rate | ViReal64 | Specifies the Rate of the acquisition second. NI-DMM sets NIDMM_ATTR_WAVEFORM_RATE The valid Range is 10.0–1,800,000 coerced to the closest integer divisio default value is 1,800,000. |
| Waveform_Points | ViInt32 | Specifies the number of points to ac waveform acquisition completes. NI NIDMM_ATTR_WAVEFORM_POINTS |

To calculate the maximum and minimum waveform points that you can acquire refer to the [Waveform Acquisition M](#)

The default value is 500.

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureCableCompType

```
ViStatus = niDMM_ConfigureCableCompType(ViSession Instrument_Handle,  
ViInt32 Cable_Comp_Type)
```

Purpose

For the NI 4072 only, sets the [NIDMM_ATTR_CABLE_COMPENSATION_TYPE](#) attribute for the current capacitance/inductance mode range.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Cable_Comp_Type | ViInt32 | Specifies the type of cable compensation that is used for the current range. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureOpenCableCompValues

```
ViStatus = niDMM_ConfigureOpenCableCompValues(ViSession  
Instrument_Handle, ViReal64 Conductance, ViReal64 Susceptance)
```


Purpose

For the NI 4072 only, configures the NIDMM_ATTR_OPEN_CABLE_COMP_CONDUCTANCE and NIDMM_ATTR_OPEN_CABLE_COMP_SUSCEPTANCE attributes.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Conductance | ViReal64 | Specifies the open cable compensation Conductance . |
| Susceptance | ViReal64 | Specifies the open cable compensation Susceptance . |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureShortCableCompValues

ViStatus = niDMM_ConfigureShortCableCompValues(ViSession
Instrument_Handle, ViReal64 Resistance, ViReal64 Reactance)

Purpose

For the NI 4072 only, configures the NIDMM_ATTR_SHORT_CABLE_COMP_RESISTANCE and NIDMM_ATTR_SHORT_CABLE_COMP_REACTANCE attributes.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Resistance | ViReal64 | Specifies the short cable compensation Resistance . |
| Reactance | ViReal64 | Specifies the short cable compensation Reactance . |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_PerformOpenCableComp

```
ViStatus = niDMM_PerformOpenCableComp(ViSession Instrument_Handle,  
                                       ViReal64 *Conductance, ViReal64 *Susceptance)
```


Purpose

For the NI 4072 only, performs the open cable compensation measurements for the current capacitance/inductance range, and returns open cable compensation **Conductance** and **Susceptance** values. You can use the return values of this function as inputs to [niDMM_ConfigureOpenCableCompValues](#).

This function returns an error if the value of the [NIDMM_ATTR_FUNCTION](#) attribute is not set to NIDMM_VAL_CAPACITANCE (1005) or NIDMM_VAL_INDUCTANCE (1006).

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|--------------------|-----------|--|
| Conductance | ViReal64* | Conductance is the measured value of open cable compensation Conductance . |
| Susceptance | ViReal64* | Susceptance is the measured value of open cable compensation Susceptance . |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_PerformShortCableComp

```
ViStatus = niDMM_PerformShortCableComp(ViSession Instrument_Handle,  
ViReal64 *Resistance, ViReal64 *Reactance)
```

Purpose

Performs the short cable compensation measurements for the current capacitance/inductance range, and returns short cable compensation **Resistance** and **Reactance** values. You can use the return values of this function as inputs to [niDMM_ConfigureShortCableCompValues](#).

This function returns an error if the value of the [NIDMM_ATTR_FUNCTION](#) attribute is not set to NIDMM_VAL_CAPACITANCE (1005) or NIDMM_VAL_INDUCTANCE (1006).

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|-------------------|-----------|---|
| Resistance | ViReal64* | Resistance is the measured value of short cable compensation Resistance . |
| Reactance | ViReal64* | Reactance is the measured value of short cable compensation Reactance . |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

Purpose

Configures the transducer type.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Transducer_Type | ViInt32 | Specifies the type of device used to measure the temperature. NI-DMM uses this value to set the Transducer Type property. The default is NIDMM_VAL_THERMOCOUPLE. |

| | |
|------------------------|--------------|
| NIDMM_VAL_2_WIRE_RTD | 2-wire RTD |
| NIDMM_VAL_4_WIRE_RTD | 4-wire RTD |
| NIDMM_VAL_THERMISTOR | Thermistor |
| NIDMM_VAL_THERMOCOUPLE | Thermocouple |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureThermocouple

```
ViStatus = niDMM_ConfigureThermocouple(ViSession Instrument_Handle,  
ViInt32 Thermocouple_Type, ViInt32 Reference_Junction_Type)
```

Purpose

Configures the thermocouple type and reference junction type for a chosen thermocouple.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Thermocouple_Type | ViInt32 | Specifies the type of thermocouple used to measure the temperature. NI-DMM uses this value to set the Thermocouple Type property. The default is NIDMM_VAL_TEMP_TC_J. |

| | |
|---------------------|---------------------|
| NIDMM_VAL_TEMP_TC_B | Thermocouple type B |
| NIDMM_VAL_TEMP_TC_E | Thermocouple type E |
| NIDMM_VAL_TEMP_TC_J | Thermocouple type J |
| NIDMM_VAL_TEMP_TC_K | Thermocouple type K |
| NIDMM_VAL_TEMP_TC_N | Thermocouple type N |
| NIDMM_VAL_TEMP_TC_R | Thermocouple type R |
| NIDMM_VAL_TEMP_TC_S | Thermocouple type S |
| NIDMM_VAL_TEMP_TC_T | Thermocouple type T |

| | | |
|--------------------------------|---------|--|
| Reference_Junction_Type | ViInt32 | Specifies the type of reference junction to be used in the reference junction compensation of a thermocouple measurement. NI-DMM uses this value to set the Reference Junction Type property. The only supported value is NIDMM_VAL_TEMP_REF_JUNC_FIXED. |
|--------------------------------|---------|--|

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureFixedRefJunction

```
ViStatus = niDMM_ConfigureFixedRefJunction(ViSession Instrument_Handle,  
ViReal64 Fixed_Reference_Junction)
```


Purpose

Configures the fixed reference junction temperature for a thermocouple with a fixed reference junction type.

Parameters

Input

| Name | Type | Description |
|---------------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Fixed_Reference_Junction | ViReal64 | Specifies the reference junction temperature when a fixed reference junction is used to take a thermocouple measurement. The units are degrees Celsius. NI-DMM uses this value to set the Fixed Reference Junction property. The default is 25.00 (°C). |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureRTDType

```
ViStatus = niDMM_ConfigureRTDType(ViSession Instrument_Handle, ViInt32  
    RTD_Type, ViReal64 RTD_Resistance)
```

Purpose

Configures the RTD Type and RTD Resistance parameters for an RTD.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle is None. |
| RTD_Type | ViInt32 | Specifies the type of RTD used to measure the temperature resistance. NIDMM_VAL_TEMP_RTD_PT3851. |

| Enum | Standards | M |
|---------------------------|---|---|
| NIDMM_VAL_TEMP_RTD_PT3851 | IEC-751 DIN 43760 BS 1904 ASTM-E1137 EN-60751 | F |
| NIDMM_VAL_TEMP_RTD_PT3750 | Low-cost vendor compliant RTD* | F |
| NIDMM_VAL_TEMP_RTD_PT3916 | JISC 1604 | F |
| NIDMM_VAL_TEMP_RTD_PT3920 | US Industrial Standard D-100 American | F |

| | | |
|------------------------------|--|---|
| NIDMM_VAL_TEMP_RTD_PT3911 | US Industrial Standard American | F |
| NIDMM_VAL_TEMP_RTD_PT3928 | ITS-90 | F |
| *No standard. Check the TCR. | | |

RTD_Resistance ViReal64 Specifies the RTD resistance in ohms at 0 °C. NI-DMM uses this value

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureRTDCustom

```
ViStatus = niDMM_ConfigureRTDCustom(ViSession Instrument_Handle,  
ViReal64 RTD_A, ViReal64 RTD_B, ViReal64 RTD_C)
```

Purpose

Configures the A, B, and C parameters for a custom RTD.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| RTD_A | ViReal64 | Specifies the Callendar-Van Dusen A coefficient for RTD scaling when RTD Type parameter is set to Custom in the niDMM_ConfigureRTDType function. The default is 3.9083e-3 (Pt3851) |
| RTD_B | ViReal64 | Specifies the Callendar-Van Dusen B coefficient for RTD scaling when RTD Type parameter is set to Custom in the niDMM_ConfigureRTDType function. The default is -5.775e-7 (Pt3851). |
| RTD_C | ViReal64 | Specifies the Callendar-Van Dusen C coefficient for RTD scaling when RTD Type parameter is set to Custom in the niDMM_ConfigureRTDType function. The default is -4.183e-12 (Pt3851). |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

Purpose

Configures the thermistor type.

Parameters

Input

Name

Type

Description

Instrument_Handle

ViSession

Identifies a particular instrument session. You obtain the **Instrument_Handle** parameter from [niDMM_init](#) or [niDMM_InitWithOptions](#). The default is None.

| Defined Values | Thermistor Type | Value | 25 °C Resistance |
|----------------------------------|-----------------|-------|------------------|
| NIDMM_VAL_TEMP_THERMISTOR_CUSTOM | Custom | 0 | — |
| NIDMM_VAL_TEMP_THERMISTOR_44004 | 44004 | 1 | 2.25 kΩ |
| NIDMM_VAL_TEMP_THERMISTOR_44006 | 44006 | 2 | 10 kΩ |
| NIDMM_VAL_TEMP_THERMISTOR_44007 | 44007 | 3 | 5 kΩ |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureThermistorCustom

```
ViStatus = niDMM_ConfigureThermistorCustom(ViSession Instrument_Handle,  
ViReal64 Thermistor_A, ViReal64 Thermistor_B, ViReal64  
Thermistor_C)
```

Purpose

Configures the A, B, and C parameters for a custom thermistor.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Thermistor_A | ViReal64 | Specifies the Steinhart-Hart A coefficient for thermistor scaling when Thermistor Type is set to Custom in the niDMM_ConfigureThermistorType function. The default is 1.0295e-3 (44006). |
| Thermistor_B | ViReal64 | Specifies the Steinhart-Hart B coefficient for thermistor scaling when Thermistor Type is set to Custom in the niDMM_ConfigureThermistorType function. The default is 2.391e-4 (44006). |
| Thermistor_C | ViReal64 | Specifies the Steinhart-Hart C coefficient for thermistor scaling when Thermistor Type is set to Custom in the niDMM_ConfigureThermistorType function. The default is 1.568e-7 (44006). |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureAutoZeroMode

```
ViStatus = niDMM_ConfigureAutoZeroMode(ViSession Instrument_Handle,  
    ViInt32 Auto_Zero_Mode)
```

Purpose

Configures the DMM for **Auto_Zero_Mode**. When **Auto_Zero_Mode** is ON, the DMM internally disconnects the input signal and takes a zero reading. It then subtracts the zero reading from the measurement. This prevents offset voltages present on the input circuitry of the DMM from affecting measurement accuracy. When **Auto_Zero_Mode** is OFF, the DMM does not compensate for zero reading offset.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Auto_Zero_Mode | ViInt32 | Specifies the Auto_Zero_Mode . NI-DMM sets the NIDMM_ATTR_AUTO_ZERO_MODE value. |

ON enables **Auto_Zero_Mode** for each measurement. ONCE enables the next measurement. The **Auto_Zero_Mode** value is stored and used for subsequent measurements until the device is reconfigured.

OFF disables **Auto_Zero_Mode**. If you set this parameter to AUTO, NI-DMM enables Auto Zero based on the measurement function that you configure. For NI 4070/4071/4072 for a 6½-digit and greater resolution DC measurement, NI-DMM sets **Auto_Zero_Mode** to ON.

For all other DC measurement configurations on the NI 4070/4071/4072, NI-DMM sets **Auto_Zero_Mode** to ONCE. For all AC measurements or waveform acquisition on NI 4070/4071/4072, NI-DMM sets **Auto_Zero_Mode** to OFF. For NI 4070/4071/4072, NI-DMM sets **Auto_Zero_Mode** to OFF when AUTO is selected.

For NI 4065 devices, **Auto_Zero_Mode** is always ON. **Auto_Zero_Mode** adds no extra time to the overall measurement phase and adds no extra time to the overall measurement phase.

| | | |
|------------------------------------|----|---|
| NIDMM_VAL_AUTO_ZERO_AUTO (default) | -1 | Auto Zero Mode is ON for all measurements. |
| NIDMM_VAL_AUTO_ZERO_OFF | 0 | Auto Zero Mode is OFF for all measurements. |
| NIDMM_VAL_AUTO_ZERO_ON | 1 | Auto Zero Mode is ON for all measurements. |

| | | |
|--------------------------|---|------------------|
| | | |
| NIDMM_VAL_AUTO_ZERO_ONCE | 2 | ir (& |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

Purpose

For NI 4070/4071/4072 only, allows the DMM to compensate for voltage offsets in resistance measurements. When **Offset_Comp_Ohms** is enabled, the DMM measures the resistance twice (once with the current source on and again with it turned off). Any voltage offset present in both measurements is cancelled out. **Offset_Comp_Ohms** is useful when measuring resistance values less than 10 K Ω .

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_ default is None. |
| Offset_Comp_Ohms | ViInt32 | Enables or disables Offset_Comp_Ohms . The driver sets NIDMM_A |

| Name |
|------------------------------------|
| NIDMM_VAL_OFFSET_COMP_OHMS_OFF (de |
| NIDMM_VAL_OFFSET_COMP_OHMS_ON |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

Purpose

Configures the [NIDMM_ATTR_AC_MIN_FREQ](#) and [NIDMM_ATTR_AC_MAX_FREQ](#) attributes, which the DMM uses for AC measurements.

Parameters

Input

| Name | Type | Description |
|--------------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| AC_Minimum_Frequency_Hz | ViReal64 | <p>Specifies the minimum expected frequency component of the input signal in hertz. This parameter affects the DMM only when you set the NIDMM_ATTR_FUNCTION attribute to AC measurements. NI-DMM uses this parameter to calculate the proper aperture for the measurement.</p> <p>The driver sets the NIDMM_ATTR_FUNCTION attribute to this value. The valid range is 1 Hz–300 kHz for the NI 4070/4071/4072, 10 Hz–100 Hz for the NI 4065, and 20 Hz–25 kHz for the NI 4050 and NI 4060.</p> |
| AC_Maximum_Frequency_Hz | ViReal64 | <p>Specifies the maximum expected frequency component of the input signal in hertz within the device limits. This parameter is used only for error checking and verifies that the value of this parameter is less than the maximum frequency of the device.</p> <p>This parameter affects the DMM only when you set the NIDMM_ATTR_FUNCTION attribute to AC measurements. The driver sets the NIDMM_ATTR_AC_MAX_FREQ attribute to this value. The valid range is 1 Hz–300 kHz for the NI 4070/4071/4072, 10 Hz–100 Hz for the NI 4065, and 20 Hz–25 kHz for the NI 4050 and NI 4060.</p> |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureFrequencyVoltageRange

```
ViStatus = niDMM_ConfigureFrequencyVoltageRange(ViSession  
Instrument_Handle, ViReal64 Voltage_Range)
```

Purpose

For the NI 4070/4071/4072 only, specifies the expected maximum amplitude of the input signal for frequency and period measurements.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_H niDMM_init or niDMM_InitWithOptions . The default is None. |
| Voltage_Range | ViReal64 | Sets the expected maximum amplitude of the input signal. Refer to the NI 4072 sections for a list of valid values. NI-DMM sets NIDMM_ATTR_FREQ_VOLTAGE_RANGE to this value. The minimum p amplitude that can be detected is 10% of the specified Voltage_Range |

| Name | Value |
|-----------------------------------|-------|
| NIDMM_VAL_AUTO_RANGE_ON (default) | -1.0 |
| NIDMM_VAL_AUTO_RANGE_OFF | -2.0 |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigurePowerLineFrequency

```
ViStatus = niDMM_ConfigurePowerLineFrequency(ViSession  
Instrument_Handle, ViReal64 Power_Line_Frequency_Hz)
```

Purpose

Specifies the powerline frequency.

Parameters

Input

| Name | Type | Description |
|--------------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Power Line Frequency Hz | ViReal64* | Powerline Frequency specifies the powerline frequency in hertz. NI-DMM sets the Powerline Frequency property to this value. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureCurrentSource

```
ViStatus = niDMM_ConfigureCurrentSource(ViSession Instrument_Handle,  
ViReal64 Current_Source)
```

Purpose

The NI 4050 and NI 4060 are not supported. Configures the **Current_Source** for diode measurements.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_H parameter from niDMM_init or niDMM_InitWithOptions . The default is N |
| Current_Source | ViReal64 | Specifies the Current_Source provided during diode measurements. For valid ranges on the NI 4065, NI 4070, 4071, or 4072, refer to the NI 4065 , NI 4070 , NI 4071 , or NI 4072 topics. The driver sets NIDMM_ATTR_CURRENT_SOURCE to this value. |

| | | |
|-----------------------------------|-----------|----------------------------------|
| NIDMM_VAL_1_MICROAMP | 1 μA | NI 4070/4071/4 only |
| NIDMM_VAL_10_MICROAMP | 10 μA | NI 4070/4071/4 only |
| NIDMM_VAL_100_MICROAMP | 100 μA | NI 4070/4071/4 and NI 4065 |
| NIDMM_VAL_1_MILLIAMP (default) | 1 mA | NI 4070/4071/4 and NI 4065 |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureWaveformCoupling

```
ViStatus = niDMM_ConfigureWaveformCoupling(ViSession  
Instrument_Handle, ViInt32 Waveform_Coupling)
```

Purpose

For the NI 4070/4071/4072 only, configures instrument coupling for voltage waveforms.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_ niDMM_init or niDMM_InitWithOptions . The default is None. |
| Waveform_Coupling | ViInt32 | Selects DC or AC coupling. The driver sets NIDMM_ATTR_WAVEFOF value. |

| Name | Value |
|---|-------|
| NIDMM_VAL_WAVEFORM_COUPLING_AC | 0 |
| NIDMM_VAL_WAVEFORM_COUPLING_DC (default) | 1 |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureTrigger



```
ViStatus = niDMM_ConfigureTrigger(ViSession Instrument_Handle, ViInt32  
    Trigger_Source, ViReal64 Trigger_Delay)
```

Purpose

Configures the DMM **Trigger_Source** and **Trigger_Delay**. Refer to [Triggering](#) and [Using Switches](#) for more information.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Trigger_Source | ViInt32 | Specifies the Trigger_Source that initiates the acquisition. The driver sets NIDMM_ATTR_TRIGGER_SOURCE to this value. Software configures the DMM to wait until niDMM_SendSoftwareTrigger is called before triggering the DMM.  Note To determine which values are supported by each device, refer to the LabWindows/CVI Trigger Routing section. |
| Trigger_Delay | ViReal64 | Specifies the time that the DMM waits after it has received a trigger before taking a measurement. The driver sets the NIDMM_ATTR_TRIGGER_DELAY attribute to this value. By default, Trigger_Delay is <code>NIDMM_VAL_AUTO_DELAY</code> (-1), which means the DMM waits an appropriate settling time before taking the measurement. On the NI 4060, if you set Trigger_Delay to 0, the DMM does not settle before taking the measurement. The NI 4065 and NI 4070/4071/4072 use the value specified in Trigger_Delay as additional settling time.  Note When using the NI 4050, Trigger_Delay must be set to <code>NIDMM_VAL_AUTO_DELAY</code> (-1). |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_SendSoftwareTrigger

ViStatus = niDMM_SendSoftwareTrigger(ViSession Instrument_Handle)

Purpose

Sends a command to trigger the DMM. Call this function if you have configured either the [NIDMM_ATTR_TRIGGER_SOURCE](#) or [NIDMM_ATTR_SAMPLE_TRIGGER](#) attributes. If the [NIDMM_ATTR_TRIGGER_SOURCE](#) and/or [NIDMM_ATTR_SAMPLE_TRIGGER](#) attributes are set to NIDMM_VAL_EXTERNAL or NIDMM_VAL_TTL n , you can use this function to override the trigger source that you configured and trigger the NI 4065 and NI 4070/4071/4072.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureMeasCompleteDest

```
ViStatus = niDMM_ConfigureMeasCompleteDest(ViSession  
Instrument_Handle, ViInt32 Meas_Complete_Destination)
```

Purpose

Specifies the destination of the DMM Measurement Complete (MC) signal. Refer to [Triggering](#) for more information.

Parameters

Input

| Name | Type | Description |
|----------------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Meas_Complete_Destination | ViInt32 | Specifies the destination of the Measurement Complete signal. This signal is issued when the DMM completes a single measurement. The driver sets the NIDMM_ATTR_MEAS_COMPLETE_DEST attribute to this value. This signal is commonly referred to as Voltmeter Complete. |



Note To determine which values are supported by each device, refer to the [LabWindows/CVI Trigger Routing](#) section.

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

Purpose

Sets the NIDMM_ATTR_TRIGGER_SLOPE attribute to either rising edge (positive) or falling edge (negative) polarity.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Trigger_Slope | ViInt32 | Specifies the polarity of the trigger signal on which the measurement is triggered for values of either NIDMM_VAL_POSITIVE or NIDMM_VAL_NEGATIVE. The driver sets the NIDMM_ATTR_TRIGGER_SLOPE attribute to this value. |

| | | |
|------------------------------|---|--|
| NIDMM_VAL_POSITIVE | 0 | The driver triggers on the rising edge of the trigger signal. |
| NIDMM_VAL_NEGATIVE (default) | 1 | The driver triggers on the falling edge of the trigger signal. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureSampleTriggerSlope

```
ViStatus = niDMM_ConfigureSampleTriggerSlope(ViSession  
Instrument_Handle, ViInt32 Sample_Trigger_Slope)
```

Purpose

Sets the NIDMM_ATTR_SAMPLE_TRIGGER_SLOPE to either rising edge (positive) or falling edge (negative) polarity.

Parameters

Input

| Name | Type | Description |
|-----------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Sample_Trigger_Slope | ViInt32 | Specifies the polarity of the Trigger signal on which the measurement is triggered for values of either NIDMM_VAL_POSITIVE or NIDMM_VAL_NEGATIVE. The driver sets NIDMM_ATTR_SAMPLE_TRIGGER_SLOPE to this value. |

| | | | |
|------------------------|---|--------------------|--|
| Rising Edge | 0 | NIDMM_VAL_POSITIVE | The driver triggers on the rising edge of the trigger signal. |
| Falling Edge (default) | 1 | NIDMM_VAL_NEGATIVE | The driver triggers on the falling edge of the trigger signal. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ConfigureMeasCompleteSlope

```
ViStatus = niDMM_ConfigureMeasCompleteSlope(ViSession  
Instrument_Handle, ViInt32 Meas_Complete_Slope)
```

Purpose

Sets the Measurement Complete signal to either rising edge (positive) or falling edge (negative) polarity.

Parameters

Input

| Name | Type | Description |
|----------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Meas_Complete_Slope | ViInt32 | Specifies the polarity of the signal that is generated. The driver sets NIDMM_ATTR_MEAS_DEST_SLOPE to this value. |

| | | | |
|------------------------|---|--------------------|--|
| Rising Edge | 0 | NIDMM_VAL_POSITIVE | The driver triggers on the rising edge of the trigger signal. |
| Falling Edge (default) | 1 | NIDMM_VAL_NEGATIVE | The driver triggers on the falling edge of the trigger signal. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetAutoRangeValue

```
ViStatus = niDMM_GetAutoRangeValue(ViSession Instrument_Handle,  
    ViReal64 *Actual_Range)
```

Purpose

Returns the **Actual_Range** that the DMM is using, even when Auto Range is off.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|---------------------|-----------|---|
| Actual_Range | ViReal64* | Indicates the Actual_Range the DMM is using. Returns the value of the NIDMM_ATTR_AUTO_RANGE_VALUE attribute. The units of the returned value depend on the function. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetApertureTimeInfo

```
ViStatus = niDMM_GetApertureTimeInfo(ViSession Instrument_Handle,  
                                       ViReal64 *Aperture_Time, ViInt32 *Aperture_Time_Units)
```

Purpose

Returns the DMM **Aperture_Time** and **Aperture_Time_Units**.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithC . The default is None. |

Output

| Name | Type | Description |
|----------------------|-----------|--|
| Aperture_Time | ViReal64* | Specifies the amount of time the DMM digitizes the input signal for a measurement. This parameter does not include settling time. Returns value of the NIDMM_ATTR_APERTURE_TIME attribute. The units of attribute depend on the value of the NIDMM_ATTR_APERTURE_TIME_UNITS attribute. |

On the NI 4070/4071/4072, the minimum aperture time is 8.89 μ s, an maximum aperture time is 149 s. Any number of powerline cycles (PLC) within the minimum and maximum ranges is allowed on the NI 4070/4071/4072.

On the NI 4065 the minimum aperture time is 333 μ s, and the maximum aperture time is 78.2 s. If setting the number of averages directly, the measurement time is aperture time X the number of averages, which be less than 72.8 s. The aperture times allowed are 333 μ s, 667 μ s, or multiples of 1.11 ms—for example 1.11 ms, 2.22 ms, 3.33 ms, and so on. If you set an aperture time other than 333 μ s, 667 μ s, or multiples of 1.11 ms, the value will be coerced up to the next supported aperture time.

On the NI 4060, when the powerline frequency is 60, the PLCs allowed are 1 PLC, 6 PLC, 12 PLC, and 120 PLC. When the powerline frequency is 50, the PLCs allowed are 1 PLC, 5 PLC, 10 PLC, and 100 PLC.

| | | |
|----------------------------|----------|--|
| Aperture_Time_Units | ViInt32* | Indicates the units of aperture time as powerline cycles (PLCs) or seconds. Returns the value of the NIDMM_ATTR_APERTURE_TIME_UNITS attribute. |
|----------------------------|----------|--|

| | | |
|-----------------------------|---|-------------|
| NIDMM_VAL_SECONDS | 0 | Seconds |
| NIDMM_VAL_POWER_LINE_CYCLES | 1 | Power Cycle |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

Purpose

Returns the measurement **Period**, which is the amount of time it takes to complete one measurement with the current configuration. Use this function right before you begin acquiring data—after you have completely configured the measurement and after all configuration functions have been called.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|---------------|-----------|---|
| Period | ViReal64* | Returns the number of seconds it takes to make one measurement. The first measurement in a multipoint acquisition requires additional settling time. This function does not include this additional time or any NIDMM_ATTR_TRIGGER_DELAY associated with the first measurement. Time required for internal measurements, such as NIDMM_ATTR_AUTO_ZERO , is included. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_SetAttributeViBoolean

```
ViStatus = niDMM_SetAttributeViBoolean(ViSession Instrument_Handle,  
    ViConstString Channel_Name, ViAttr Attribute_ID, ViBoolean  
    Attribute_Value)
```

Purpose

This function sets the value of a ViBoolean attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes.

In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViBoolean | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_SetAttributeViInt32

```
ViStatus = niDMM_SetAttributeViInt32(ViSession Instrument_Handle,  
ViConstString Channel_Name, ViAttr Attribute_ID, ViInt32  
Attribute_Value)
```

Purpose

This function sets the value of a `ViInt32` attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes.

In contrast, when you set multiple attributes using the `SetAttribute` functions, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViInt32 | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_SetAttributeViReal64

ViStatus = niDMM_SetAttributeViReal64(ViSession Instrument_Handle,
ViConstString Channel_Name, ViAttr Attribute_ID, ViReal64
Attribute_Value)

Purpose

This function sets the value of a ViReal64 attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes.

In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViReal64 | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_SetAttributeViSession

ViStatus = niDMM_SetAttributeViSession(ViSession Instrument_Handle,
ViConstString Channel_Name, ViAttr Attribute_ID, ViSession
Attribute_Value)

Purpose

This function sets the value of a ViSession attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViSession | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

Purpose

This function sets the value of a ViString attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes.

In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViString | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetAttributeViBoolean

ViStatus = niDMM_GetAttributeViBoolean(ViSession Instrument_Handle,
ViConstString Channel_Name, ViAttr Attribute_ID, ViBoolean
*Attribute_Value)

Purpose

Queries the value of a ViBoolean attribute. You can use this function to get the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |

Output

| Name | Type | Description |
|------------------------|-------------|---|
| Attribute_Value | ViBoolean* | Returns the current value of the attribute. Pass the address of a ViBoolean variable. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetAttributeViInt32

```
ViStatus = niDMM_GetAttributeViInt32(ViSession Instrument_Handle,  
ViConstString Channel_Name, ViAttr Attribute_ID, ViInt32  
*Attribute_Value)
```

Purpose

Queries the value of a ViInt32 attribute. You can use this function to get the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |

Output

| Name | Type | Description |
|------------------------|-------------|---|
| Attribute_Value | ViInt32* | Returns the current value of the attribute. Pass the address of a ViInt32 variable. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetAttributeViReal64

```
ViStatus = niDMM_GetAttributeViReal64(ViSession Instrument_Handle,  
    ViConstString Channel_Name, ViAttr Attribute_ID, ViReal64  
    *Attribute_Value)
```

Purpose

Queries the value of a ViReal64 attribute. You can use this function to get the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |

Output

| Name | Type | Description |
|------------------------|-------------|--|
| Attribute_Value | ViReal64* | Returns the current value of the attribute. Pass the address of a ViReal64 variable. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetAttributeViSession

```
ViStatus = niDMM_GetAttributeViSession(ViSession Instrument_Handle,  
    ViConstString Channel_Name, ViAttr Attribute_ID, ViSession  
    *Attribute_Value)
```

Purpose

Queries the value of a ViSession attribute. You can use this function to get the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |

Output

| Name | Type | Description |
|------------------------|-------------|---|
| Attribute_Value | ViSession* | Returns the current value of the attribute. Pass the address of a ViSession variable. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetAttributeViString

ViStatus = niDMM_GetAttributeViString(ViSession Instrument_Handle,
ViConstString Channel_Name, ViAttr Attribute_ID, ViInt32
Buffer_Size, ViChar Attribute_Value[])

Purpose

Queries the value of a ViString attribute. You can use this function to get the values of instrument-specific attributes and inherent IVI attributes.

If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid. You must provide a ViChar array to serve as a buffer for the value. You pass the number of bytes in the buffer as the Array Size parameter.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Buffer_Size | ViInt32 | Pass the number of bytes in the ViChar array you specify for the Attribute_Value parameter. If the current value of the attribute, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies Buffer_Size —1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer_Size is 4, the function places "123" into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value. If you pass 0, you can pass VI_NULL for the Attribute_Value buffer parameter. |

Output

| Name | Type | Description |
|------------------------|-----------|---|
| Attribute_Value | ViChar[] | The buffer in which the function returns the current value of the attribute. The buffer must be of type ViChar and have at least as many bytes as indicated in the Buffer_Size parameter. If you specify 0 for the Buffer_Size parameter, you can pass VI_NULL for this parameter. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CheckAttributeViBoolean

```
ViStatus = niDMM_CheckAttributeViBoolean(ViSession Instrument_Handle,  
ViConstString Channel_Name, ViAttr Attribute_ID, ViBoolean  
Attribute_Value)
```

Purpose

This function checks the validity of a value you specify for a ViBoolean attribute.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViBoolean | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CheckAttributeViInt32

```
ViStatus = niDMM_CheckAttributeViInt32(ViSession Instrument_Handle,  
    ViConstString Channel_Name, ViAttr Attribute_ID, ViInt32  
    Attribute_Value)
```

Purpose

This function checks the validity of a value you specify for a `ViInt32` attribute.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViInt32 | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CheckAttributeViReal64

```
ViStatus = niDMM_CheckAttributeViReal64(ViSession Instrument_Handle,  
    ViConstString Channel_Name, ViAttr Attribute_ID, ViReal64  
    Attribute_Value)
```

Purpose

This function checks the validity of a value you specify for a ViReal64 attribute.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViReal64 | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CheckAttributeViSession

```
ViStatus = niDMM_CheckAttributeViSession(ViSession Instrument_Handle,  
ViConstString Channel_Name, ViAttr Attribute_ID, ViSession  
Attribute_Value)
```

Purpose

This function checks the validity of a value you specify for a ViSession attribute.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViSession | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CheckAttributeViString

```
ViStatus = niDMM_CheckAttributeViString(ViSession Instrument_Handle,  
    ViConstString Channel_Name, ViAttr Attribute_ID, ViChar  
    Attribute_Value[])
```

Purpose

This function checks the validity of a value you specify for a ViString attribute.

Parameters

Input

| Name | Type | Description |
|--------------------------|---------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Channel_Name | ViConstString | This parameter is ignored. National Instruments DMMs do not support channel names since they only have a single channel. This parameter is included in order to support interchangeability and upgradability to multiple channel DMMs. The default value is " " (an empty string). |
| Attribute_ID | ViAttr | Pass the ID of an attribute. |
| Attribute_Value | ViChar[] | Pass the value that you want to set the attribute to. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_Read

ViStatus = niDMM_Read(ViSession Instrument_Handle, ViInt32
Maximum_Time, ViReal64 *Reading)

Purpose

Acquires a single measurement and returns the measured value.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Maximum_Time | ViInt32 | Specifies the Maximum_Time allowed for this function to complete in milliseconds. If the function does not complete within this time interval, the function returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete. The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically. |

Output

| Name | Type | Description |
|----------------|-------------|---|
| Reading | ViReal64* | The measured value returned from the DMM. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ReadMultiPoint

ViStatus = niDMM_ReadMultiPoint(ViSession Instrument_Handle, ViInt32
Maximum_Time, ViInt32 Array_Size, ViReal64 Reading_Array[],
ViInt32 *Actual_Number_Of_Points)

Purpose


Acquires multiple measurements and returns an array of measured values. The number of measurements the DMM makes is determined by the values you specify for the **Trigger_Count** and **Sample_Count** parameters in [niDMM_ConfigureMultiPoint](#).

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Maximum_Time | ViInt32 | <p>Specifies the Maximum_Time allowed for this function to complete in milliseconds. If the function does not complete within this time interval, the function returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.</p> <p>The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically.</p> |
| Array_Size | ViInt32 | <p>Specifies the number of measurements to acquire. The maximum number of measurements for a finite acquisition is the (Trigger Count x Sample Count) parameters in niDMM_ConfigureMultiPoint.</p> <p>For continuous acquisitions, up to 100,000 points can be returned at once. The number of measurements can be a subset. The valid range is any positive ViInt32. The default value is 1.</p> |

Output

| Name | Type | Description |
|--------------------------------|------------|--|
| Reading_Array | ViReal64[] | <p>An array of measurement values.</p> <p> Note The size of the Reading_Array must be at least the size that you specify for the Array_Size parameter.</p> |
| Actual_Number_Of_Points | ViInt32* | Indicates the number of measured values actually retrieved from the DMM. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ReadWaveform

```
ViStatus = niDMM_ReadWaveform(ViSession Instrument_Handle, ViInt32  
    Maximum_Time, ViInt32 Array_Size, ViReal64 Waveform_Array[],  
    ViInt32 *Actual_Number_Of_Points)
```

Purpose


For the NI 4070/4071/4072 only, acquires a waveform and returns data as an array of values or as a waveform data type. The number of elements in the **Waveform_Array** is determined by the values you specify for the **Waveform_Points** parameter in [niDMM_ConfigureWaveformAcquisition](#).

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Maximum_Time | ViInt32 | Specifies the Maximum_Time allowed for this function to complete in milliseconds. If the function does not complete within this time interval, the function returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete. The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically. |
| Array_Size | ViInt32 | Specifies the number of waveform points to return. You specify the total number of points that the DMM acquires in the Waveform Points parameter of niDMM_ConfigureWaveformAcquisition . The default value is 1. |

Output

| Name | Type | Description |
|--------------------------------|------------|--|
| Waveform_Array | ViReal64[] | An array of measurement values.  Note The size of the Waveform_Array must be at least the size that you specify for the Array_Size parameter. |
| Actual_Number_Of_Points | ViInt32* | Indicates the number of measured values actually retrieved from the DMM. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_IsOverRange


```
ViStatus = niDMM_IsOverRange(ViSession Instrument_Handle, ViReal64  
    Measurement_Value, ViBoolean *Is_Over_Range)
```

Purpose

Takes a **Measurement_Value** and determines if the value is a valid measurement or a value indicating that an overrange condition occurred.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Measurement_Value | ViReal64 | The measured value returned from the DMM.  Note If an overrange condition occurs, the Measurement_Value contains an IEEE-defined NaN (Not a Number) value. |

Output

| Name | Type | Description |
|----------------------|------------|---|
| Is_Over_Range | ViBoolean* | Returns whether the measurement value is a valid measurement or an overrange condition. |

| | | |
|----------|---|---|
| VI_TRUE | 1 | The value indicates that an overrange condition occurred. |
| VI_FALSE | 0 | The value is a valid measurement. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_IsUnderRange


```
ViStatus = niDMM_IsUnderRange(ViSession Instrument_Handle, ViReal64  
    Measurement_Value, ViBoolean *Is_Under_Range)
```

Purpose

Takes a **Measurement_Value** and determines if the value is a valid measurement or a value indicating that an underrange condition occurred.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Measurement_Value | ViReal64 | The measured value returned from the DMM.  Note If an overrange condition occurs, the Measurement_Value contains an IEEE-defined NaN (Not a Number) value. |

Output

| Name | Type | Description |
|-----------------------|------------|---|
| Is_Under_Range | ViBoolean* | Returns whether the Measurement_Value is a valid measurement or an underrange condition. |

| | | |
|----------|---|--|
| VI_TRUE | 1 | The value indicates that an underrange condition occurred. |
| VI_FALSE | 0 | The value is a valid measurement. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_Initiate

ViStatus = niDMM_Initiate(ViSession Instrument_Handle)

Purpose

Initiates an acquisition. After you call this function, the DMM leaves the Idle state and enters the Wait-for-Trigger state. If trigger is set to Immediate mode, the DMM begins acquiring measurement data. Use [niDMM_Fetch](#), [niDMM_FetchMultiPoint](#), or [niDMM_FetchWaveform](#) to retrieve the measurement data.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_Fetch

niDMM_Fetch(ViSession Instrument_Handle, ViInt32 Maximum_Time,
ViReal64 *Reading)

Purpose

Returns the value from a previously initiated measurement. You must call [niDMM_Initiate](#) before calling this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Maximum_Time | ViInt32 | Specifies the Maximum_Time allowed for this function to complete in milliseconds. If the function does not complete within this time interval, the function returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete. The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically. |

Output

| Name | Type | Description |
|----------------|-------------|---|
| Reading | ViReal64* | The measured value returned from the DMM. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_FetchMultiPoint

```
ViStatus = niDMM_FetchMultiPoint(ViSession Instrument_Handle, ViInt32  
    Maximum_Time, ViInt32 Array_Size, ViReal64 Reading_Array[],  
    ViInt32 *Actual_Number_Of_Points)
```

Purpose


Returns an array of values from a previously initiated multipoint measurement. The number of measurements the DMM makes is determined by the values you specify for the **Trigger_Count** and **Sample_Count** parameters of [niDMM_ConfigureMultiPoint](#). You must first call [niDMM_Initiate](#) to initiate a measurement before calling this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Maximum_Time | ViInt32 | <p>Specifies the Maximum_Time allowed for this function to complete in milliseconds. If the function does not complete within this time interval, the function returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.</p> <p>The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically.</p> |
| Array_Size | ViInt32 | <p>Specifies the number of measurements to acquire. The maximum number of measurements for a finite acquisition is the (Trigger Count x Sample Count) parameters in niDMM_ConfigureMultiPoint.</p> <p>For continuous acquisitions, up to 100,000 points can be returned at once. The number of measurements can be a subset. The valid range is any positive ViInt32. The default value is 1.</p> |

Output

| Name | Type | Description |
|--------------------------------|------------|--|
| Reading_Array | ViReal64[] | <p>An array of measurement values.</p> <p> Note The size of the Reading_Array must be at least the size that you specify for the Array_Size parameter.</p> |
| Actual_Number_Of_Points | ViInt32* | Indicates the number of measured values actually retrieved from the DMM. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_FetchWaveform

```
ViStatus = niDMM_FetchWaveform(ViSession Instrument_Handle, ViInt32  
    Maximum_Time, ViInt32 Array_Size, ViReal64 Waveform_Array[],  
    ViInt32 *Actual_Number_Of_Points)
```

Purpose

For the NI 4070/4071/4072 only, returns an array of values from a previously initiated waveform acquisition. You must call [niDMM_Initiate](#) before calling this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Maximum_Time | ViInt32 | Specifies the Maximum_Time allowed for this function to complete in milliseconds. If the function does not complete within this time interval, the function returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete. The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically. |
| Array_Size | ViInt32 | Specifies the number of waveform points to return. You specify the total number of points that the DMM acquires in the Waveform Points parameter of niDMM_ConfigureWaveformAcquisition . The default value is 1. |

Output

| Name | Type | Description |
|--------------------------------|----------------|---|
| Waveform_Array | ViReal64 [] | Waveform Array is an array of measurement values stored in waveform data type. |
| Actual_Number_Of_Points | ViInt32* | Indicates the number of measured values actually retrieved from the DMM. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_Abort

ViStatus = niDMM_Abort(ViSession Instrument_Handle)

Purpose

Aborts a previously initiated measurement and returns the DMM to the Idle state.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ReadStatus

ViStatus = niDMM_ReadStatus(ViSession Instrument_Handle, ViInt32
*Acquisition_Backlog, ViInt16 *Acquisition_Status)

Purpose

Returns measurement backlog and acquisition status. Use this function to determine how many measurements are available before calling [niDMM_Fetch](#), [niDMM_FetchMultipoint](#), or [niDMM_FetchWaveform](#).



Note The NI 4050 is not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|----------------------------|----------|---|
| Acquisition_Backlog | ViInt32* | The number of measurements available to be read. If the backlog continues to increase, data is eventually overwritten, resulting in an error. |



Note On the NI 4060, the **Backlog** does not increase when autoranging. On the NI 4065, the **Backlog** does not increase when Range is set to AUTO RANGE ON (-1), or before the first point is fetched when Range is set to AUTO RANGE ONCE (-3). These behaviors are due to the autorange model of the devices.

| | | |
|---------------------------|----------|--|
| Acquisition_Status | ViInt16* | Indicates status of the acquisition. The following table shows the acquisition states: |
|---------------------------|----------|--|

| | |
|---|----------------------------|
| 0 | Running |
| 1 | Finished with backlog |
| 2 | Finished with no backlog |
| 3 | Paused |
| 4 | No acquisition in progress |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_Control

```
ViStatus = niDMM_Control(ViSession Instrument_Handle, ViInt32  
Control_Action)
```

Purpose

Controls the DMM. Use this function if you want a parameter change to be immediately reflected in the hardware. Use this function before calling [niDMM_Initiate](#) to make the initiate call as quickly as possible.



Notes The NI 4050 and NI 4060 are not supported.

Calling this function while the DMM is taking measurements results in an error. After the DMM is finished taking measurements, calling this function will make any unfetched data points unavailable.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Control Action | ViInt32 | The action you want the driver to perform. Only NIDMM_VAL_CONTROL_COMMIT (0) is supported, which commits to hardware all of the configured attributes associated with the session. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_InitExtCal

```
ViStatus = niDMM_InitExtCal(ViString Resource_Name, ViChar  
    Calibration_Password[], ViSession *Instrument_Handle)
```

Purpose

The following operations are performed if the **Calibration_Password** is valid:

- Creates a new session for external calibration to the device you specify for the **Resource_Name** parameter.
- Resets the device and prepares the EEPROM for new calibration coefficients.
- Returns a ViSession handle that you use to identify the instrument in all calibration adjustments and post-adjustment verification steps.



Notes The NI 4050 and NI 4060 are not supported.

Refer to the *NI 4065 6½ Digit DMM Calibration Procedure*, the *NI 4070/4072 6½ Digit FlexDMM Calibration Procedure*, or the *NI 4071 7½-Digit FlexDMM Calibration Procedure* before using this function.

This function creates a new session the first time you invoke it for a device. If you call this function on the same resource, an error is returned. You should use [niDMM_CloseExtCal](#) to close a session obtained using this function.

After opening a calibration session, the device cannot take valid measurements using this session until the device has been properly adjusted. Once the adjustment phase is complete, you can use this session to verify the new calibration constants. After verification, you have the option of saving the new calibration constants or reverting to the previous calibration constants by specifying the **Action** parameter in [niDMM_CloseExtCal](#).

If you encounter a fatal error such as a power failure or system crash while performing an external calibration, you can call [niDMM_RestoreLastExtCalConstants](#) to return the device to a usable state.

Parameters

Input

| Name | Type | Description |
|----------------------|----------|---|
| Resource_Name | ViString | Contains the Resource_Name of the device to initialize. The Resource_Name is assigned in Measurement & Automation Explorer (MAX). Refer to Related Documentation for the <i>NI Digital Multimeters Getting Started Guide</i> for more information about configuring and testing the DMM in MAX. |

Valid Syntax:

- NI-DAQmx name
- DAQ::NI-DAQmx name[::INSTR]
- DAQ::Traditional NI-DAQ device number[::INSTR]
- IVI logical name



Caution All IVI names for the **Resource_Name**, such as logical names or virtual names, are case-sensitive. If you use logical names, driver session names, or virtual names in your program, you must make sure that the name you use matches the name in the IVI Configuration Store file exactly, without any variations in the case of the characters in the name.

| | | |
|-----------------------------|----------|---|
| Calibration_Password | ViChar[] | Specifies the password required to enable external calibration functionality. The maximum password string length is eight characters, excluding the termination character. "NI" is the factory-default password. |
|-----------------------------|----------|---|

Output

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | The session handle that you obtain from niDMM_InitExtCal. The handle identifies a particular instrument calibration session. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CalAdjustLinearization

```
ViStatus = niDMM_CalAdjustLinearization(ViSession Instrument_Handle,  
    ViInt32 Mode, ViReal64 Range, ViReal64 Input_Resistance,  
    ViReal64 Expected_Value)
```

Purpose

For the NI 4065 only, compensates for any non-linearities.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | The session handle that you obtain from niDMM_InitExtCal . The handle identifies a particular instrument calibration session. |
| Function | ViInt32 | Specifies the calibration Function used to acquire the measurement. For valid modes, refer to the <i>NI 4065 6½ Digit DMM Calibration Procedure</i> . |
| Range | ViReal64 | Specifies the Range to calibrate. Range values are coerced up to the closest Range . For valid ranges, refer to the <i>NI 4065 6½ Digit DMM Calibration Procedure</i> . Auto-ranging is not supported for calibration operations. |
| Input_Resistance | ViReal64 | Specifies the Input_Resistance that the device should use. Input_Resistance values are coerced up to the closest Input_Resistance . |
| Expected_Value | ViReal64 | Specifies the Expected_Value of the measurement. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CalAdjustGain

ViStatus = niDMM_CalAdjustGain(ViSession Instrument_Handle, ViInt32 Mode, ViReal64 Range, ViReal64 Input_Resistance, ViReal64 Expected_Value)

Purpose

Calibrates the gain coefficient for the supplied **Mode**, **Range**, and **Input_Resistance**.



Notes The NI 4050 and NI 4060 are not supported.

Refer to the *NI 4065 6½ Digit DMM Calibration Procedure*, the *NI 4070/4072 6½ Digit FlexDMM Calibration Procedure*, or the *NI 4071 7½-Digit FlexDMM Calibration Procedure* before using this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|--|
| Instrument_Handle | ViSession | The session handle that you obtain from niDMM_InitExtCal . The handle identifies a particular instrument calibration session. |
| Mode | ViInt32 | Specifies the calibration Mode used to acquire the measurement. For valid modes, refer to the <i>NI 4065 6½ Digit DMM Calibration Procedure</i> , the <i>NI 4070/4072 6½ Digit FlexDMM Calibration Procedure</i> , or the <i>NI 4071 7½-Digit FlexDMM Calibration Procedure</i> . |
| Range | ViReal64 | Specifies the Range to calibrate. For valid ranges, refer to the <i>NI 4065 6½ Digit DMM Calibration Procedure</i> , the <i>NI 4070/4072 6½ Digit FlexDMM Calibration Procedure</i> , or the <i>NI 4071 7½-Digit FlexDMM Calibration Procedure</i> . Auto-ranging is not supported for calibration operations. |
| Input_Resistance | ViReal64 | Specifies the Input_Resistance that the device should use. Input_Resistance values are coerced up to the closest Input_Resistance . |
| Expected_Value | ViReal64 | Specifies the Expected_Value of the measurement. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CalAdjustOffset

ViStatus = niDMM_CalAdjustOffset(ViSession Instrument_Handle, ViInt32 Mode, ViReal64 Range, ViReal64 Input_Resistance)

Purpose

Calibrates the offset and Auto Zero offset for the supplied **Mode**, **Range**, and **Input_Resistance**.



Note The NI 4050 and NI 4060 are not supported.

Refer to the *NI 4065 6½ Digit DMM Calibration Procedure*, the *NI 4070/4072 6½ Digit FlexDMM Calibration Procedure*, or the *NI 4071 7½-Digit FlexDMM Calibration Procedure* before using this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|--|
| Instrument_Handle | ViSession | The session handle that you obtain from niDMM_InitExtCal . The handle identifies a particular instrument calibration session. |
| Mode | ViInt32 | Specifies the calibration Mode used to acquire the measurement. For valid modes, refer to the <i>NI 4065 6½ Digit DMM Calibration Procedure</i> , the <i>NI 4070/4072 6½ Digit FlexDMM Calibration Procedure</i> , or the <i>NI 4071 7½–Digit FlexDMM Calibration Procedure</i> . |
| Range | ViReal64 | Specifies the Range to calibrate. For valid ranges, refer to the <i>NI 4065 6½ Digit DMM Calibration Procedure</i> , the <i>NI 4070/4072 6½ Digit FlexDMM Calibration Procedure</i> , or the <i>NI 4071 7½–Digit FlexDMM Calibration Procedure</i> . Auto-ranging is not supported for calibration operations. |
| Input_Resistance | ViReal64 | Specifies the Input_Resistance that the device should use. Input_Resistance values are coerced up to the closest Input_Resistance . |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CalAdjustMisc

ViStatus = niDMM_CalAdjustMisc(ViSession Instrument_Handle, ViInt32
Type)

Purpose

Performs a specialized calibration step depending on the specified **Type**.



Note The NI 4050 and NI 4060 are not supported.

Refer to the *NI 4065 6½ Digit DMM Calibration Procedure*, the *NI 4070/4072 6½ Digit FlexDMM Calibration Procedure*, or the *NI 4071 7½-Digit FlexDMM Calibration Procedure* before using this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | The session handle that you obtain from niDMM_InitExtCal . The handle particular instrument calibration session. |
| Type | ViInt32 | Specifies which of the miscellaneous calibration steps to perform. |

| |
|-------------------------------------|
| NIDMM_EXTCAL_MISCCAL_VREF (default) |
| NIDMM_EXTCAL_MISCCAL_RREF |
| NIDMM_EXTCAL_MISCCAL_ZINT |
| NIDMM_EXTCAL_MISCCAL_2WIRELEAKAGE |
| NIDMM_EXTCAL_MISCCAL_4WIRELEAKAGE |
| NIDMM_EXTCAL_MISCCAL_SECTION |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CalAdjustLC

ViStatus = niDMM_CalAdjustLC(ViSession Instrument_Handle, ViInt32 Type)

Purpose

For the NI 4072 only, performs a specialized LC calibration step depending on the specified **Type**.



Note Refer to the *NI 4070/4072 6½ Digit FlexDMM Calibration Procedure* before using this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | The session handle that you obtain from niDMM_InitExtCal . The handle identifies a particular instrument calibration session. |
| Type | ViInt32 | Specifies which of the LC calibration steps to perform. |

| | |
|----------------------|---|
| L & C Open (default) | Calibrates the open compensation. |
| L & C Short | Calibrates the short compensation. |
| L & C 25 Ω | Calibrates the 25 Ω resistance. |
| L & C 1 k Ω | Calibrates the 1 k Ω resistance. |
| L & C 5 k Ω | Calibrates the 5 k Ω resistance. |
| L & C 100 k Ω | Calibrates the 100 k Ω resistance. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CalAdjustACFilter

ViStatus = niDMM_CalAdjustACFilter(ViSession Instrument_Handle, ViInt32 Mode, ViReal64 Range, ViReal64 Frequency, ViReal64 Expected_Value)

Purpose

For the NI 4070/4071/4072 only, calibrates the filter coefficients used for AC measurements of the supplied **Mode** and **Range**.



Note Refer to the *NI 4070/4072 6½ Digit FlexDMM Calibration Procedure* or the *NI 4071 7½–Digit FlexDMM Calibration Procedure* before using this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | The session handle that you obtain from niDMM_InitExtCal . The handle identifies a particular instrument calibration session. |
| Mode | ViInt32 | Specifies the calibration Mode used to acquire the measurement. For valid modes, refer to the <i>NI 4065 6½ Digit DMM Calibration Procedure</i> , the <i>NI 4070/4072 6½ Digit FlexDMM Calibration Procedure</i> , or the <i>NI 4071 7½-Digit FlexDMM Calibration Procedure</i> . |
| Range | ViReal64 | Specifies the Range to calibrate. For valid ranges, refer to the <i>NI 4070/4072 6½ Digit FlexDMM Calibration Procedure</i> or the <i>NI 4071 7½-Digit FlexDMM Calibration Procedure</i> . Auto-ranging is not supported for calibration operations. |
| Frequency | ViReal64 | Specifies the Frequency of the input signal. |
| Expected_Value | ViReal64 | Specifies the Expected_Value of the measurement. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_CloseExtCal

ViStatus = niDMM_CloseExtCal(ViSession Instrument_Handle, ViInt32 Action)

Purpose

Performs the specified **Action**, closes the specified external calibration session obtained from [niDMM_InitExtCal](#), and deallocates resources that it reserved.



Note The NI 4050 and NI 4060 are not supported.

Refer to the *NI 4065 6½ Digit DMM Calibration Procedure*, the *NI 4070/4072 6½ Digit FlexDMM Calibration Procedure*, or the *NI 4071 7½-Digit FlexDMM Calibration Procedure* before using this function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | The session handle that you obtain from niDMM_InitExtCal . The handle particular instrument calibration session. |
| Action | ViInt32 | Specifies whether the driver saves the updated calibration constants. |

| | |
|-------------------------------------|---|
| NIDMM_EXTCAL_ACTION_ABORT (default) | Re: the cal cor to v the bef sta the ext cal pro |
| NIDMM_EXTCAL_ACTION_SAVE | Sa nev cal cor to t dev |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetCalCount

```
ViStatus = niDMM_GetCalCount(ViSession Instrument_Handle, ViInt32  
    Cal_Type, ViInt32 *Count)
```

Purpose

Returns the calibration **Count** for the specified type of calibration.




Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_H parameter from niDMM_init or niDMM_InitWithOptions . The default is N |
| Cal_Type | ViInt32 | Specifies the type of calibration performed (external or self-calibration). |

| | | |
|--|---|------------|
| NIDMM_VAL_INTERNAL_AREA (default) | 0 | Self-Calik |
|  Note The NI 4065 does not support self-calibration. | | |
| NIDMM_VAL_EXTERNAL_AREA | 1 | Exte Calik |

Output

| Name | Type | Description |
|--------------|----------|---|
| Count | ViInt32* | The number of times calibration has been performed. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetCalDateAndTime

```
ViStatus = niDMM_GetCalDateAndTime(ViSession Instrument_Handle,  
    ViInt32 Cal_Type, ViInt32 *Month, ViInt32 *Day, ViInt32 *Year,  
    ViInt32 *Hour, ViInt32 *Minute)
```

Purpose

Returns the date and time of the last calibration performed.




Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_H parameter from niDMM_init or niDMM_InitWithOptions . The default is N |
| Cal_Type | ViInt32 | Specifies the type of calibration performed (external or self-calibration). |

| | | |
|--|---|------------|
| NIDMM_VAL_INTERNAL_AREA (default) | 0 | Self-Calik |
|  Note The NI 4065 does not support self-calibration. | | |
| NIDMM_VAL_EXTERNAL_AREA | 1 | Exte Calik |

Output

| Name | Type | Description |
|---------------|----------|--|
| Month | ViInt32* | Indicates the Month of the last calibration. |
| Day | ViInt32* | Indicates the Day of the last calibration. |
| Year | ViInt32* | Indicates the Year of the last calibration. |
| Hour | ViInt32* | Indicates the Hour of the last calibration. |
| Minute | ViInt32* | Indicates the Minute of the last calibration. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetDevTemp

ViStatus = niDMM_GetDevTemp(ViSession Instrument_Handle, ViString
Options, ViReal64 *Temperature)

Purpose

Returns the current **Temperature** of the device.



Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Options | ViString | Reserved. |

Output

| Name | Type | Description |
|--------------------|-------------|---|
| Temperature | ViReal64* | Returns the current Temperature of the device. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetLastCalTemp

```
ViStatus = niDMM_GetLastCalTemp(ViSession Instrument_Handle, ViInt32  
    Cal_Type, ViReal64 *Temperature)
```

Purpose

Returns the **Temperature** during the last calibration procedure.




Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_H parameter from niDMM_init or niDMM_InitWithOptions . The default is N |
| Cal_Type | ViInt32 | Specifies the type of calibration performed (external or self-calibration). |

| | | |
|--|---|------------|
| NIDMM_VAL_INTERNAL_AREA (default) | 0 | Self-Calik |
|  Note The NI 4065 does not support self-calibration. | | |
| NIDMM_VAL_EXTERNAL_AREA | 1 | Exte Calik |

Output

| Name | Type | Description |
|--------------------|-----------|---|
| Temperature | ViReal64* | Returns the Temperature during the last calibration. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetCalUserDefinedInfoMaxSize

```
ViStatus = niDMM_GetCalUserDefinedInfoMaxSize(ViSession  
Instrument_Handle, ViInt32 *Info_Size)
```

Purpose

Returns the maximum string length that can be stored in the EEPROM. Use [niDMM_SetCalUserDefinedInfo](#) to store user-defined information.



Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|------------------|-------------|---|
| Info_Size | ViInt32* | Returns the value of maximum string length that can be stored in the EEPROM using niDMM_SetCalUserDefinedInfo . The Info_Size value is given in characters, but it does not include the termination character. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

Purpose

Returns the user-defined calibration information stored in the EEPROM.



Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Buffer_Size | ViInt32 | <p>Passes the number of bytes in the ViString you specify for the Info parameter.</p> <p>If zero is passed for this parameter, the Buffer_Size needed to store the information is returned. If the Info parameter, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies Buffer_Size - 1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the Buffer_Size you must pass to get the entire value.</p> <p>For example, if the value is "123456" and the Buffer_Size is 4, the function places "123" into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.</p> |

Output

| Name | Type | Description |
|-------------|----------|---|
| Info | ViChar[] | Returns the user-defined calibration information stored in the EEPROM. If this is NULL, the Buffer_Size needed to store the information is returned. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_SetCalUserDefinedInfo

```
ViStatus = niDMM_SetCalUserDefinedInfo(ViSession Instrument_Handle,  
ViChar Info[])
```

Purpose

Stores the user-defined information in the EEPROM. Use [niDMM_GetCalUserDefinedInfoMaxSize](#) to learn the maximum string size that is allowed. If the **Info** string size is larger than the maximum string size, NI-DMM stores as much of the information as possible, truncates the remainder, and returns a warning.



Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Info | ViChar[] | Specifies the user-defined information to be stored in the EEPROM such as the operator who performed the calibration operation or system information. Use niDMM_GetCalUserDefinedInfoMaxSize to learn the maximum string size that is allowed. If the Info string size is larger than the maximum string size, NI-DMM stores as much of the information as possible, truncates the remainder, and return a warning. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetExtCalRecommendedInterval

ViStatus = niDMM_GetExtCalRecommendedInterval(ViSession
Instrument_Handle, ViInt32 *Months)

Purpose

Returns the recommended interval between external recalibration in **Months**.



Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|---------------|-------------|--|
| Months | ViInt32* | Returns the recommended number of Months between external calibrations. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetSelfCalSupported

ViStatus = niDMM_GetSelfCalSupported(ViSession Instrument_Handle,
ViBoolean *Self_Cal_Supported)

Purpose

Returns a Boolean value that expresses whether or not the DMM that you are using can perform self-calibration.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|---------------------------|------------|--|
| Self_Cal_Supported | ViBoolean* | Returns whether Self Cal is supported for the device specified by the given session. |

| | | |
|----------|---|---|
| VI_TRUE | 1 | The DMM that you are using can perform self-calibration. |
| VI_FALSE | 0 | The DMM that you are using cannot perform self-calibration. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_RestoreLastExtCalConstants

ViStatus = niDMM_RestoreLastExtCalConstants(ViSession Instrument_Handle)

Purpose

Reverts the device to the calibration constants from the last complete external calibration. This function recovers the hardware if a fatal system error should occur during an external or self-calibration procedure.

For the NI 4070/4071/4072 only, after calling this function, you should call [niDMM_SelfCal](#) before taking measurements with the device to adjust the device for any temperature drifts since the last external calibration.



Note The NI 4050 and NI 4060 are not supported.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_SetCalPassword

```
ViStatus = niDMM_SetCalPassword(ViSession Instrument_Handle, ViChar  
    Old_Password[], ViChar New_Password[])
```

Purpose

Changes the password required to enable external calibration functionality for the specified instrument. The maximum password string length is eight characters, excluding the termination character. "NI" is the default password.



Notes The NI 4050 and NI 4060 are not supported.

A password is required for external calibration. Be sure to record the password in a secure location.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Old_Password | ViChar[] | Specifies the current password required to enable external calibration functionality. The maximum password string length is eight characters, excluding the termination character. |
| New_Password | ViChar[] | Specifies the New_Password required to enable external calibration functionality. The maximum password string length is eight characters, excluding the termination character. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_reset

ViStatus = niDMM_reset(ViSession Instrument_Handle)

Purpose

Resets the instrument to a known state and sends initialization commands to the instrument. The initialization commands set instrument settings to the state necessary for the operation of the instrument driver.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ResetWithDefaults

ViStatus = niDMM_ResetWithDefaults(ViSession Instrument_Handle)

Purpose

Resets the instrument to a known state and sends initialization commands to the DMM. The initialization commands set the DMM settings to the state necessary for the operation of NI-DMM. All user-defined default values associated with a logical name are applied after setting the DMM.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_self_test

```
ViStatus = niDMM_self_test(ViSession Instrument_Handle, ViInt16  
    *Self_Test_Result, ViChar Self_-_Test_Message[])
```

Purpose

Performs a self-test on the DMM to ensure that the DMM is functioning properly. Self-test does not calibrate the DMM.



Note This function calls [niDMM_reset](#), and any configurations previous to the call will be lost. All attributes will be set to their default values after the call returns.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|-------------------------|----------|---|
| Self_Test_Result | ViInt16* | Contains the value returned from the instrument self-test. Zero indicates success. On the NI 4070/4072, the error code 1013 indicates that you should check the fuse and replace it, if necessary. |



Note Self-test does not check the fuse on the NI 4065 and NI 4071. Hence, even if the fuse is blown on the NI 4065 or NI 4071, self-test does not return error code 1013.

| | | |
|---------------------------------|-----------|--|
| Self - _Test_Message | ViChar[] | This parameter contains the string returned from the instrument self-test. The array must contain at least 256 elements. For the NI 4050 and NI 4060, the error codes returned for self-test failures include the following: <ul style="list-style-type: none">• NIDMM_ERROR_AC_TEST_FAILURE• NIDMM_ERROR_DC_TEST_FAILURE• NIDMM_ERROR_RESISTANCE_TEST_FAILURE These error codes indicate that the DMM should be repaired. For the NI 4070/4071/4072, the error code returned for a self-test failure is NIDMM_ERROR_SELF_TEST_FAILURE. This error code indicates that the DMM should be repaired. |
|---------------------------------|-----------|--|

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_revision_query

```
ViStatus = niDMM_revision_query(ViSession Instrument_Handle, ViChar  
    Instrument_Driver_Revision[], ViChar Firmware_Revision[])
```

Purpose

Returns the revision numbers of the instrument driver and instrument firmware.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|-----------------------------------|----------|--|
| Instrument_Driver_Revision | ViChar[] | Returns a string containing the instrument driver software revision numbers. |



Note The array must contain at least 256 elements ViChar[256].

| | | |
|--------------------------|----------|--|
| Firmware_Revision | ViChar[] | Returns a string containing the instrument Firmware_Revision numbers. |
|--------------------------|----------|--|



Note The array must contain at least 256 elements ViChar[256].

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_FormatMeasAbsolute

ViStatus = niDMM_FormatMeasAbsolute(ViInt32 Measurement_Function,
ViReal64 Range, ViReal64 Resolution, ViReal64 Measurement,
ViChar Mode_String[], ViChar Range_String[], ViChar
Data_String[])

Purpose

Formats the **Measurement** to the proper number of displayed digits according to the **Measurement_Function**, **Range**, and **Resolution**. Returns the formatted data, range, and mode strings.

Parameters

Input

| Name | Type | Description |
|-----------------------------|-------------|--|
| Measurement_Function | ViInt32 | Specifies the Measurement_Function used to acquire the measurement. The driver sets <u>NIDMM_ATTR_FUNCTION</u> to this value. |
| Range | ViReal64 | Specifies the <u>NIDMM_ATTR_RANGE</u> used to acquire the Measurement . |
| Resolution | ViReal64 | Specifies the <u>NIDMM_ATTR_RESOLUTION_ABSOLUTE</u> of the Measurement . |
| Measurement | ViReal64 | Specifies the measured value returned from the DMM. |

Output

| Name | Type | Description |
|---------------------|-------------|--|
| Mode_String | ViChar[] | Returns a string containing the units of the Measurement mode. |
| Range_String | ViChar[] | Returns the <u>NIDMM_ATTR_RANGE</u> of the Measurement , formatted into a string with the correct number of display digits. |
| Data_String | ViChar[] | Returns the Measurement , formatted according to the <u>NIDMM_ATTR_FUNCTION</u> , <u>NIDMM_ATTR_RANGE</u> , and <u>NIDMM_ATTR_RESOLUTION_ABSOLUTE</u> . |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_Disable

ViStatus = niDMM_Disable(ViSession Instrument_Handle)

Purpose

Places the instrument in a quiescent state where it has minimal or no impact on the system to which it is connected. If a measurement is in progress when this function is called, the measurement is aborted.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetChannelName

ViStatus = niDMM_GetChannelName(ViSession Instrument_Handle, ViInt32
Index, ViInt32 Buffer_Size, ViChar Channel_String[])

Purpose

Returns the **Channel_String** that is in the channel table at an **Index** you specify. Not applicable to National Instruments DMMs. Included for compliance with the *IviDmm Class Specification*.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Index | ViInt32 | A 1-based Index into the channel table. |
| Buffer_Size | ViInt32 | Passes the number of bytes in the ViChar array you specify for the Channel_String parameter. If the next Channel_String , including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies Buffer_Size -1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer_Size is 4, the function places "123" into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value. If you pass 0, you can pass VI_NULL for the Channel_String buffer parameter. The default value is None. |

Output

| Name | Type | Description |
|-----------------------|-----------|---|
| Channel_String | ViChar[] | Returns the Channel_String that is in the channel table at the Index you specify. Do not modify the contents of the Channel_String . |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_error_message

```
ViStatus = niDMM_error_message(ViSession Instrument_Handle, ViStatus  
    Error_Code, ViChar Error_Message[])
```

Purpose

Takes the **Error_Code** returned by the instrument driver functions, interprets it, and returns it as a user-readable string.

Parameters

Input

| Name | Type | Description |
|--------------------------|-------------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Error_Code | ViStatus | The Error_Code returned from the instrument. The default is 0, indicating VI_SUCCESS. |

Output

| Name | Type | Description |
|----------------------|-------------|--|
| Error_Message | ViChar[] | The error information formatted into a string. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ClearError

ViStatus = niDMM_ClearError(ViSession Instrument_Handle)

Purpose

Clears the error information for the current execution thread and the IVI session you specify. If you pass VI_NULL for the **Instrument_Handle** parameter, this function clears the error information only for the current execution thread.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetError

ViStatus = niDMM_GetError(ViSession Instrument_Handle, ViStatus *Error
Code, ViInt32 Buffer_Size, ViChar Description[])

Purpose

Returns the error information associated with the **Instrument_Handle**. This function retrieves and then clears the error information for the session. If you leave the **Instrument_Handle** unwired, this function retrieves and then clears the error information for the process.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Buffer_Size | ViInt32 | <p>Passes the number of bytes in the ViChar array you specify for the Description parameter. If the error description, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies Buffer_Size -1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the Buffer_Size you must pass to get the entire value.</p> <p>For example, if the value is "123456" and the Buffer_Size is 4, the function places "123" into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value. If you pass 0, you can pass VI_NULL for the Description buffer parameter. The default value is None.</p> |

Output

| Name | Type | Description |
|--------------------|-----------|---|
| Error_Code | ViStatus* | Returns the Error_Code for the session or execution thread. If you pass 0 for the Buffer_Size , you can pass VI_NULL for this parameter. |
| Description | ViChar[] | Returns the error Description for the IVI session or execution thread. If there is no Description , the function returns an empty string. The buffer must contain at least as many elements as the value you specify with the Buffer_Size parameter. If you pass 0 for the Buffer_Size , you can pass VI_NULL for this parameter. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_error_query

```
ViStatus = niDMM_error_query(ViSession Instrument_Handle, ViStatus  
    *Error_Code, ViChar Error_Message[])
```

Purpose


Reads an **Error_Code** and message from the DMM error queue. National Instruments DMMs do not contain an error queue. Errors are reported as they occur. Therefore, this function does not detect errors; it is included for compliance with the *IviDmm Class Specification*.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|----------------------|-----------|---|
| Error_Code | ViStatus* | The Error_Code returned from the instrument. The default value is VI_SUCCESS (0). |
| Error_Message | ViChar[] | Formats the Error_Code into a user-readable message string.  Note The array must contain at least 256 elements ViChar[256]. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_LockSession

ViStatus = niDMM_LockSession(ViSession Instrument_Handle, ViBoolean
*Caller_Has_Lock)

Purpose

This function obtains a multithread lock on the instrument session. Before it does so, it waits until all other execution threads have released their locks on the instrument session.

Other threads might have obtained a lock on this session in the following ways:

- The user application called this function.
- A call to the instrument driver locked the session.
- A call to the IVI Library locked the session.

After your call to this function returns successfully, no other threads can access the instrument session until you call [niDMM_UnlockSession](#).

Use this function and [niDMM_UnlockSession](#) around a sequence of calls to instrument driver functions if you require that the instrument retain its settings through the end of the sequence. You can safely make nested calls to this function within the same thread.

To completely unlock the session, you must balance each call to this function with a call to [niDMM_UnlockSession](#). If, however, you use the **Caller_Has_Lock** parameter in all calls to this function and [niDMM_UnlockSession](#) within a function, the IVI Library locks the session only once within the function regardless of the number of calls you make to this function. This feature allows you to call [niDMM_UnlockSession](#) just once at the end of the function.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|------------------------|------------|---|
| Caller_Has_Lock | ViBoolean* | This parameter serves as a convenience. If you do not want to use this parameter, pass VI_NULL. Use this parameter in complex functions to keep track of whether you obtain a lock and, therefore, need to unlock the session. To use this parameter, complete the following steps: <ol style="list-style-type: none">1. Pass the address of a local ViBoolean variable.2. In the declaration of the local variable, initialize it to VI_FALSE (0).3. Pass the address of the same local variable to any other calls you make to this function or niDMM_UnlockSession in the same function. |

The parameter is an input/output parameter. This function and [niDMM_UnlockSession](#) each inspect the current value and take the following actions:

If the value is VI_TRUE (1), this function does not lock the session again. If the value is VI_FALSE, this function obtains the lock and sets the value of the parameter to VI_TRUE.

If the value is VI_FALSE, [niDMM_UnlockSession](#) does not attempt to unlock the session. If the value is VI_TRUE, [niDMM_UnlockSession](#) releases the lock and sets the value of the parameter to VI_FALSE. Thus, you can, call [niDMM_UnlockSession](#) at the end of your function without worrying about whether you actually have the lock.

Example

ViStatus TestFunc (ViSession vi, ViInt32 flags)

```
{
    ViStatus error = VI_SUCCESS;
    ViBoolean haveLock = VI_FALSE;
    if (flags & BIT_1)
    {
        viCheckErr( NIDMM_LockSession(vi, &haveLock));
        viCheckErr( TakeAction1(vi));
        if (flags & BIT_2)
        {
            viCheckErr( NIDMM_UnlockSession(vi, &haveLock));
            viCheckErr( TakeAction2(vi));
            viCheckErr( NIDMM_LockSession(vi, &haveLock));
        }
    }
}
```

```
if (flags & BIT_3)
viCheckErr( TakeAction3(vi));
}
```

Error:

```
/*
    At this point, you cannot really be sure that you have the lock.
    Fortunately, the haveLock variable takes care of that for you.
*/
niDMM_UnlockSession(vi, &haveLock);
return error;
}
```

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_UnlockSession

ViStatus = niDMM_UnlockSession(ViSession Instrument_Handle, ViBoolean
*Caller_Has_Lock)

Purpose

This function releases a lock that you acquired on an instrument session using `niDMM_LockSession`. Refer to [niDMM_LockSession](#) for additional information on session locks.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Output

| Name | Type | Description |
|------------------------|------------|---|
| Caller_Has_Lock | ViBoolean* | This parameter serves as a convenience. If you do not want to use this parameter, pass VI_NULL. |

Use this parameter in complex functions to keep track of whether you obtain a lock and, therefore, need to unlock the session.

To use this parameter, complete the following steps:

1. Pass the address of a local ViBoolean variable.
2. In the declaration of the local variable, initialize it to VI_FALSE (0).
3. Pass the address of the same local variable to any other calls you make to [niDMM_LockSession](#) or this function in the same function.

The parameter is an input/output parameter. [niDMM_LockSession](#) and this function each inspect the current value and take the following actions:

If the value is VI_TRUE (1), [niDMM_LockSession](#) does not lock the session again. If the value is VI_FALSE, [niDMM_LockSession](#) obtains the lock and sets the value of the parameter to VI_TRUE.

If the value is VI_FALSE, this function does not attempt to unlock the session. If the value is VI_TRUE, this function releases the lock and sets the value of the parameter to VI_FALSE. Thus, you can, call this function at the end of your function without worrying about whether you actually have the lock.

Example

```
ViStatus TestFunc (ViSession vi, ViInt32 flags)
```

```
{
    ViStatus error = VI_SUCCESS;
    ViBoolean haveLock = VI_FALSE;
    if (flags & BIT_1)
    {
        viCheckErr( NIDMM_LockSession(vi, &haveLock));
        viCheckErr( TakeAction1(vi));
        if (flags & BIT_2)
        {
            viCheckErr( NIDMM_UnlockSession(vi, &haveLock));
            viCheckErr( TakeAction2(vi));
        }
    }
}
```

```
viCheckErr( NIDMM_LockSession(vi, &haveLock);  
}  
if (flags & BIT_3)  
viCheckErr( TakeAction3(vi));  
}
```

Error:

```
/*  
    At this point, you cannot really be sure that you have the lock.  
    Fortunately, the haveLock variable takes care of that for you.  
*/  
niDMM_UnlockSession(vi, &haveLock);  
return error;  
}
```

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetNextInterchangeWarning

```
ViStatus = niDMM_GetNextInterchangeWarning(ViSession Instrument_Handle,  
      ViInt32 Buffer_Size, ViChar Interchange_Warning[])
```

Purpose

This function returns the interchangeability warnings associated with the IVI session. It retrieves and clears the oldest instance in which the class driver recorded an interchangeability warning. Interchangeability warnings indicate that using your application with a different instrument might cause different behavior.

The driver performs interchangeability checking when `NIDMM_ATTR_INTERCHANGE_CHECK` is set to `VI_TRUE` (1). The function returns an empty string in the **Interchange_Warning** parameter if no interchangeability warnings remain for the session. In general, the instrument driver generates interchangeability warnings when an attribute that affects the behavior of the instrument is in a state that you did not specify.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Buffer_Size | ViInt32 | <p>Passes the number of bytes in the ViChar array you specify for the Interchange_Warning parameter. If the next interchangeability warning string, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies Buffer_Size - 1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value.</p> <p>For example, if the value is "123456" and the Buffer_Size is 4, the function places "123" into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value. If you pass 0, you can pass VI_NULL for the Interchange_Warning buffer parameter. The default value is None.</p> |

Output

| Name | Type | Description |
|----------------------------|----------|---|
| Interchange_Warning | ViChar[] | Returns the next interchange warning for the IVI session. If there are no interchange warnings, the function returns an empty string. The buffer must contain at least as many elements as the value you specify with the Buffer_Size parameter. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ClearInterchangeWarnings

ViStatus = niDMM_ClearInterchangeWarnings(ViSession Instrument_Handle)

Purpose

Clears the list of current interchange warnings.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_ResetInterchangeCheck

ViStatus = niDMM_ResetInterchangeCheck(ViSession Instrument_Handle)

Purpose

When developing a complex test system that consists of multiple test modules, it is generally a good idea to design the test modules so that they can run in any order. To do so requires ensuring that each test module completely configures the state of each instrument it uses.

If a particular test module does not completely configure the state of an instrument, the state of the instrument depends on the configuration from a previously executed test module. If you execute the test modules in a different order, the behavior of the instrument and therefore the entire test module is likely to change. This change in behavior is generally instrument specific and represents an interchangeability problem. You can use this function to test for such cases. After you call this function, the interchangeability checking algorithms in NI-DMM ignore all previous configuration operations. By calling this function at the beginning of a test module, you can determine whether the test module has dependencies on the operation of previously executed test modules.

This function does not clear the interchangeability warnings from the list of previously recorded interchangeability warnings. If you want to guarantee that [niDMM_GetNextInterchangeWarning](#) only returns those interchangeability warnings that are generated after calling this function, you must clear the list of interchangeability warnings. You can clear the interchangeability warnings list by repeatedly calling [niDMM_GetNextInterchangeWarning](#) until no more interchangeability warnings are returned. If you are not interested in the content of those warnings, you can call [niDMM_ClearInterchangeWarnings](#).

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

niDMM_GetNextCoercionRecord

```
ViStatus = niDMM_GetNextCoercionRecord(ViSession Instrument_Handle,  
                                       ViInt32 Buffer_Size, ViChar Coercion_Record[])
```


Purpose

This function returns the coercion information associated with the IVI session, and it retrieves and clears the oldest instance in which NI-DMM coerced a value you specified to another value.

If you set [NIDMM_ATTR_RECORD_COERCIONS](#) to VI_TRUE (1), NI-DMM keeps a list of all coercions it makes on ViInt32 or ViReal64 values that you pass to NI-DMM functions. Use this function to retrieve information from that list.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . The default is None. |
| Buffer_Size | ViInt32 | <p>Passes the number of bytes in the ViChar array you specify for the Coercion_Record parameter. If the next coercion record string, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies Buffer_Size – 1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value.</p> <p>For example, if the value is "123456" and the Buffer_Size is 4, the function places "123" into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.</p> <p>If you pass 0, you can pass VI_NULL for the Coercion_Record buffer parameter.</p> <p>The default value is None.</p> |

Output

| Name | Type | Description |
|------------------------|----------|---|
| Coercion_Record | ViChar[] | <p>Returns the next Coercion_Record for the IVI session.</p> <p>If there are no coercions records, the function returns an empty string. The buffer must contain at least as many elements as the value you specify with the Buffer_Size parameter.</p> |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|

NIDMM_ATTR_RESOLUTION_ABSOLUTE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViReal64 | R/W | None | niDMM_ConfigureMeasurementDigits niDMM_ConfigureMeasurementAbsolute |

Description

Specifies the measurement resolution in absolute units. Setting this attribute to higher values increases the measurement accuracy. Setting this attribute to lower values increases the measurement speed.



Note NI-DMM ignores this attribute for capacitance and inductance measurements on the NI 4072. To achieve better resolution for such measurements, use the [NIDMM_ATTR_LC_NUMBER_MEAS_TO_AVERAGE](#) attribute.

NIDMM_ATTR_APERTURE_TIME

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViReal64 | R/W | Yes | niDMM_GetApertureTimeInfo |

Description

Specifies the measurement aperture time for the current configuration. Aperture time is specified in units set by [NIDMM_ATTR_APERTURE_TIME_UNITS](#). To override the default aperture, set this attribute to the desired aperture time after calling [niDMM_ConfigureMeasurementDigits](#). To return to the default, set this attribute to NIDMM_VAL_APERTURE_TIME_AUTO (-1).

On the NI 4070/4071/4072, the minimum aperture time is 8.89 μ s, and the maximum aperture time is 149 s. Any number of powerline cycles (PLCs) within the minimum and maximum ranges is allowed on the NI 4070/4071/4072.

On the NI 4065 the minimum aperture time is 333 μ s, and the maximum aperture time is 78.2 s. If setting the number of averages directly, the total measurement time is aperture time X the number of averages, which must be less than 72.8 s. The aperture times allowed are 333 μ s, 667 μ s, or multiples of 1.11 ms—for example 1.11 ms, 2.22 ms, 3.33 ms, and so on. If you set an aperture time other than 333 μ s, 667 μ s, or multiples of 1.11 ms, the value will be coerced up to the next supported aperture time.

On the NI 4060, when the powerline frequency is 60 Hz, the PLCs allowed are 1 PLC, 6 PLC, 12 PLC, and 120 PLC. When the powerline frequency is 50 Hz, the PLCs allowed are 1 PLC, 5 PLC, 10 PLC, and 100 PLC.

NIDMM_ATTR_APERTURE_TIME_UNITS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | niDMM_GetApertureTimeInfo |

ViInt32

Description

Specifies the units of aperture time for the current configuration.



Note The NI 4060 does *not* support an aperture time set in seconds.

Defined Values

| | | |
|-----------------------------|---|------------------|
| NIDMM_VAL_SECONDS | 0 | Seconds |
| NIDMM_VAL_POWER_LINE_CYCLES | 1 | Powerline Cycles |

NIDMM_ATTR_NUMBER_OF_AVERAGES

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

Specifies the number of averages to perform in a measurement. For the NI 4070/4071/4072, applies only when the aperture time is not set to AUTO and Auto Zero is ON. The default is 1.



Note The NI 4050 and NI 4060 are not supported.

NIDMM_ATTR_OPERATION_MODE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

Specifies how the NI 4065 and NI 4070/4071/4072 acquire data.

When you call [niDMM_ConfigureMeasurementDigits](#), NI-DMM sets this attribute to NIDMM_VAL_IVIDMM_MODE. When you call [niDMM_ConfigureWaveformAcquisition](#), NI-DMM sets this attribute to NIDMM_VAL_WAVEFORM_MODE. If you are programming attributes directly, you must set this attribute before setting other configuration attributes.

Defined Values

| | |
|-------------------------|---|
| NIDMM_VAL_IVIDMM_MODE | Single or multipoint measurements—when the NIDMM_ATTR_TRIGGER_COUNT and NIDMM_ATTR_SAMPLE_COUNT attributes are both set to 1, the NI 4065 and NI 4070/4071/4072 take a single-point measurement; otherwise, NI-DMM takes multipoint measurements. |
| NIDMM_VAL_WAVEFORM_MODE | Waveform Mode configures the NI 4070/4071/4072 to take waveform measurements. |

NIDMM_ATTR_SETTLE_TIME

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViReal64 | R/W | None | None |

Description

Specifies the settling time in seconds. To override the default settling time, set this attribute. To return to the default, set this attribute to NIDMM_VAL_SETTLE_TIME_AUTO (-1).



Note The NI 4050 and NI 4060 are not supported.

NIDMM_ATTR_AUTO_RANGE_VALUE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViReal64 | RO | None | niDMM_GetAutoRangeValue |

Description

Specifies the value of the range. If auto ranging is enabled, shows the actual value of the active range. The value of this attribute is set during a read operation.

NIDMM_ATTR_RESOLUTION_DIGITS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViReal64 | R/W | None | niDMM_ConfigureMeasurementDigits niDMM_ConfigureMeasurementAbsolute |

Description

Specifies the measurement resolution in digits. Setting this attribute to higher values increases the measurement accuracy. Setting this attribute to lower values increases the measurement speed.



Note NI-DMM ignores this attribute for capacitance and inductance measurements on the NI 4072. To achieve better resolution for such measurements, use the [NIDMM_ATTR_LC_NUMBER_MEAS_TO_AVERAGE](#) attribute.

NIDMM_ATTR_FUNCTION

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViInt32 | R/W | None | niDMM_ConfigureMeasurementDigits niDMM_ConfigureMeasurementAbsolute niDMM_ConfigureWaveformAcquisition |

Description

Specifies the measurement function.



Note If you are setting this attribute directly, you must also set the [NIDMM_ATTR_OPERATION_MODE](#) attribute, which controls whether the DMM takes standard single or multipoint measurements, or acquires a waveform. If you are programming attributes directly, you must set the [NIDMM_ATTR_OPERATION_MODE](#) attribute before setting other configuration attributes. If the [NIDMM_ATTR_OPERATION_MODE](#) attribute is set to NIDMM_VAL_WAVEFORM_MODE, the only valid function types are NIDMM_VAL_WAVEFORM_VOLTAGE and NIDMM_VAL_WAVEFORM_CURRENT. Set the [NIDMM_ATTR_OPERATION_MODE](#) attribute to NIDMM_VAL_IVIDMM_MODE to set all other function values.

Defined Values

| Name | Default Value | Description | Devices supported |
|-------------------------------|---------------|-----------------------------|---|
| NIDMM_VAL_DC_VOLTS | 1 | DC Voltage | All |
| NIDMM_VAL_AC_VOLTS | 2 | AC Voltage with AC Coupling | All |
| NIDMM_VAL_DC_CURRENT | 3 | DC Current | All |
| NIDMM_VAL_AC_CURRENT | 4 | AC Current | All |
| NIDMM_VAL_2_WIRE_RES | 5 | 2-Wire Resistance | All |
| NIDMM_VAL_4_WIRE_RES | 101 | 4-Wire Resistance | NI 4060, NI 4065, and NI 4070/4071/4072 |
| NIDMM_VAL_FREQ | 104 | Frequency | NI 4070/4071/4072 |
| NIDMM_VAL_PERIOD | 105 | Period | NI 4070/4071/4072 |
| NIDMM_VAL_AC_VOLTS_DC_COUPLED | 1001 | AC Voltage with DC Coupling | NI 4070/4071/4072 |
| NIDMM_VAL_DIODE | 1002 | Diode | All |
| NIDMM_VAL_WAVEFORM_VOLTAGE | 1003 | Waveform Voltage | NI 4070/4071/4072 |
| NIDMM_VAL_WAVEFORM_CURRENT | 1004 | Waveform Current | NI 4070/4071/4072 |
| NIDMM_VAL_CAPACITANCE | 1005 | Capacitance | NI 4072 |
| NIDMM_VAL_INDUCTANCE | 1006 | Inductance | NI 4072 |

NIDMM_ATTR_ADC_CALIBRATION

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | niDMM_ConfigureADCCalibration |

Description

For the NI 4070/4071/4072 only, specifies the ADC calibration mode.

Defined Values

| | | |
|--|------|--|
| NIDMM_VAL_ADC_CALIBRATION_AUTO (default) | -1.0 | The DMM enables or disables ADC calibration based on the configured function and resolution. |
| NIDMM_VAL_ADC_CALIBRATION_OFF | 0 | The DMM does not compensate for changes to the gain. |
| NIDMM_VAL_ADC_CALIBRATION_ON | 1 | The DMM measures an internal reference to calculate the correct gain for the measurement. |

NIDMM_ATTR_AUTO_ZERO




Specific Attribute

Data type **Access** **Coercion** **High-Level Function**
ViInt32 R/W None [niDMM_ConfigureAutoZeroMode](#)

Description

Specifies the Auto Zero mode.

Defined Values

| | | |
|------------------------------------|----|--|
| NIDMM_VAL_AUTO_ZERO_AUTO (default) | -1 | NI-DMM chooses the Auto Zero setting based on the configured function and resolution. |
| NIDMM_VAL_AUTO_ZERO_OFF | 0 | Disables Auto Zero.  Note The NI 4065 does <i>not</i> support this setting. |
| NIDMM_VAL_AUTO_ZERO_ON | 1 | The DMM internally disconnects the input signal following each measurement and takes a zero reading. It then subtracts the zero reading from the preceding reading.  Note For NI 4065 devices, Auto Zero is always ON. Auto Zero is an integral part of the signal measurement phase and adds no extra time to the overall measurement. |
| NIDMM_VAL_AUTO_ZERO_ONCE | 2 | The DMM internally disconnects the input signal following each measurement and takes a zero reading. It then subtracts the zero reading from the preceding reading.  Note The NI 4060/4065 does <i>not</i> support this setting. |

NIDMM_ATTR_DC_BIAS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

For the NI 4072 only, controls the available DC bias for capacitance measurements.

Defined Values

| | | |
|---------------|---|---|
| NIDMM_VAL_OFF | 0 | NI-DMM programs the device <i>not</i> to use the DC bias. |
| NIDMM_VAL_ON | 1 | NI-DMM programs the device to use the DC bias. |

NIDMM_ATTR_LC_CALCULATION_MODEL

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

For the NI 4072 only, specifies the type of algorithm that the measurement processing uses for capacitance and inductance measurements.

Defined Values

| | | |
|--------------------|----|--|
| NIDMM_VAL_AUTO | -1 | NI-DMM chooses the algorithm based on function and range. |
| NIDMM_VAL_SERIES | 0 | NI-DMM uses the series impedance model to calculate capacitance and inductance. |
| NIDMM_VAL_PARALLEL | 1 | NI-DMM uses the parallel admittance model to calculate capacitance and inductance. |

NIDMM_ATTR_CABLE_COMP_TYPE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViInt32 | R/W | None | niDMM_ConfigureCableCompType |

Description

For the NI 4072 only, specifies the type of cable compensation that is applied to the current capacitance or inductance measurement for the current range.



Note Changing the function or the range through this attribute or through [niDMM_ConfigureMeasurementDigits](#) resets the value of this attribute to the default value.

Defined Values

| | | |
|-------------------------------------|---|-----------------------------------|
| NIDMM_VAL_CABLE_COMP_NONE (default) | 0 | No Cable Compensation |
| NIDMM_VAL_CABLE_COMP_OPEN | 1 | Open Cable Compensation |
| NIDMM_VAL_CABLE_COMP_SHORT | 2 | Short Cable Compensation |
| NIDMM_VAL_CABLE_COMP_OPEN_AND_SHORT | 3 | Open and Short Cable Compensation |

NIDMM_ATTR_OPEN_CABLE_COMP_CONDUCT/

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViReal64 | R/W | None | niDMM_ConfigureOpenCableCompValues |

Description

For the NI 4072 only, specifies the active part (conductance) of the open cable compensation. The valid range is any real number >0. The default value (-1.0) indicates that compensation has not taken place.



Note Changing the function or the range through this attribute or through [niDMM_ConfigureMeasurementDigits](#) resets the value of this attribute to the default value.

NIDMM_ATTR_OPEN_CABLE_COMP_SUSCEPTA

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViInt32 | R/W | None | niDMM_ConfigureOpenCableCompValues |

Description

For the NI 4072 only, specifies the reactive part (susceptance) of the open cable compensation. The valid range is any real number >0. The default value (-1.0) indicates that compensation has not taken place.



Note Changing the function or the range through this attribute or through [niDMM_ConfigureMeasurementDigits](#) resets the value of this attribute to the default value.

NIDMM_ATTR_LC_NUMBER_MEAS_TO_AVERAG

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

For the NI 4072 only, specifies the number of LC measurements that are averaged to produce one reading.

NIDMM_ATTR_SHORT_CABLE_COMP_REACTAN

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | niDMM_ConfigureShortCableCompValues |

Description

For the NI 4072 only, represents the reactive part (reactance) of the short cable compensation. The valid range is any real number >0. The default value (-1) indicates that compensation has not taken place.



Note Changing the function or the range through this attribute or through [niDMM_ConfigureMeasurementDigits](#) resets the value of this attribute to the default value.

NIDMM_ATTR_SHORT_CABLE_COMP_RESISTANCE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | ConfigureShortCableCompValues |

Description

For the NI 4072 only, represents the active part (resistance) of the short cable compensation. The valid range is any real number >0. The default value (-1) indicates that compensation has not taken place.



Note Changing the function or the range through this attribute or through [niDMM ConfigureMeasurementDigits](#) resets the value of this attribute to the default value.

NIDMM_ATTR_CURRENT_SOURCE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViReal64 | R/W | None | niDMM_ConfigureCurrentSource |

Description

Specifies the current source provided during diode measurements.



Note The NI 4050 and NI 4060 are not supported.

Defined Values

| | | |
|--------------------------------|-------------|-------------------------------|
| NIDMM_VAL_1_MICROAMP | 1 μ A | NI 4070/4071/4072 only |
| NIDMM_VAL_10_MICROAMP | 10 μ A | NI 4070/4071/4072 only |
| NIDMM_VAL_100_MICROAMP | 100 μ A | NI 4070/4071/4072 and NI 4065 |
| NIDMM_VAL_1_MILLIAMP (default) | 1 mA | NI 4070/4071/4072 and NI 4065 |

NIDMM_ATTR_DC_NOISE_REJECTION

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |


Description

Specifies the DC noise rejection mode.



Note The NI 4050 and NI 4060 are not supported.

Defined Values

| | | |
|-----------------------------|----|---|
| NIDMM_VAL_DCNR_AUTO | -1 | The driver chooses the DC noise rejection setting based on the configured function and resolution. |
| NIDMM_VAL_DCNR_NORMAL | 0 | NI-DMM weighs all samples equally. |
| NIDMM_VAL_DCNR_SECOND_ORDER | 1 | NI-DMM weighs the samples taken in the middle of the aperture time more than samples taken at the beginning and the end of the measurement using a triangular weighing function. |
| NIDMM_VAL_DCNR_HIGH_ORDER | 2 | NI-DMM weighs the samples taken in the middle of the aperture time more than samples taken at the beginning and the end of the measurement using a bell-curve weighing function.  Note The NI 4065 does <i>not</i> support this setting. |

NIDMM_ATTR_FREQ_VOLTAGE_AUTO_RANGE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViReal64 | RO | None | None |

Description

For the NI 4070/4071/4072 only, specifies the value of the frequency voltage range. If Auto Ranging, shows the actual value of the active frequency voltage range. If not Auto Ranging, the value is the same as that of [NIDMM_ATTR_FREQ_VOLTAGE_RANGE](#).

NIDMM_ATTR_FREQ_VOLTAGE_RANGE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViReal64 | R/W | Yes | niDMM_ConfigureFrequencyVoltageRange |

Description

For the NI 4070/4071/4072 only, specifies the maximum amplitude of the input signal for frequency measurements.

| | | |
|-----------------------------------|------|--|
| NIDMM_VAL_AUTO_RANGE_ON (default) | -1.0 | Configures the DMM to take an Auto Range measurement to calculate the voltage range before each frequency or period measurement. |
| NIDMM_VAL_AUTO_RANGE_OFF | -2.0 | Disables Auto Ranging. NI-DMM sets the voltage range to the last calculated voltage range. |

NIDMM_ATTR_INPUT_RESISTANCE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViReal64 | R/W | Yes | None |

Description

Specifies the input resistance of the instrument.



Note The NI 4050 and NI 4060 are not supported.

Defined Values

| | | |
|-----------------------------------|----------------|----------------|
| NIDMM_VAL_1_MEGAOHM | 1000000.00 | 1 M Ω |
| NIDMM_VAL_10_MEGAOHM | 10000000.00 | 10 M Ω |
| NIDMM_VAL_GREATER_THAN_10_GIGAOHM | 10000000000.00 | >10 G Ω |

NIDMM_ATTR_AC_MAX_FREQ

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViReal64 | R/W | Yes | niDMM_ConfigureACBandwidth |

Description

Specifies the maximum frequency component of the input signal for AC measurements. This attribute is used only for error checking and verifies that the value of this parameter is less than the maximum frequency of the device. This attribute affects the DMM only when you set the [NIDMM_ATTR_FUNCTION](#) attribute to AC measurements. The valid range is 1 Hz–300 kHz for the NI 4070/4071/4072, 10 Hz–100 kHz for the NI 4065, and 20 Hz–25 kHz for the NI 4050 and NI 4060.

NIDMM_ATTR_AC_MIN_FREQ

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViReal64 | R/W | None | niDMM_ConfigureACBandwidth |

Description

Specifies the minimum frequency component of the input signal for AC measurements. This attribute affects the DMM only when you set the [NIDMM_ATTR_FUNCTION](#) attribute to AC measurements. The valid range is 1 Hz–300 kHz for the NI 4070/4071/4072, 10 Hz–100 kHz for the NI 4065, and 20 Hz–25 kHz for the NI 4050 and NI 4060.

NIDMM_ATTR_OFFSET_COMP_OHMS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | niDMM_ConfigureOffsetCompOhms |

Description

For the NI 4070/4071/4072 only, enables or disables offset compensated ohms.

Defined Values

| | | |
|--|---|---------------------------------------|
| NIDMM_VAL_OFFSET_COMP_OHMS_OFF (default) | 0 | Off disables Offset Compensated Ohms. |
| NIDMM_VAL_OFFSET_COMP_OHMS_ON | 1 | On enables Offset Compensated Ohms. |

NIDMM_ATTR_POWERLINE_FREQ

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViReal64 | R/W | None | niDMM_ConfigurePowerlineFrequency |

Description

Specifies the powerline frequency. The NI 4050 and NI 4060 use this value to select an aperture time to reject powerline noise by selecting the appropriate internal sample clock and filter. The NI 4065 and NI 4070/4071/4072 use this value to select a timebase for setting the [NIDMM_ATTR_APERTURE_TIME](#) attribute in powerline cycles (PLCs).

After configuring powerline frequency, set the [NIDMM_ATTR_APERTURE_TIME_UNITS](#) attribute to PLCs. When setting the [NIDMM_ATTR_APERTURE_TIME](#) attribute, select the number of PLCs for the powerline frequency. For example, if powerline frequency = 50 Hz (or 20ms) and aperture time in PLCs = 5, then aperture time in Seconds = 20ms * 5 PLCs = 100 ms. Similarly, if powerline frequency = 60 Hz (or 16.667 ms) and aperture time in PLCs = 6, then aperture time in Seconds = 16.667 ms * 6 PLCs = 100 ms.

Defined Values

| | |
|--------------------|-------|
| NIDMM_VAL_50_HERTZ | 50 Hz |
| NIDMM_VAL_60_HERTZ | 60 Hz |



Note For 400 Hz powerline frequency, use the 50 Hz setting.

NIDMM_ATTR_SHUNT_VALUE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViReal64 | R/W | None | None |

Description

For the NI 4050 only, specifies the shunt resistance value.



Note The NI 4050 requires an external shunt resistor for current measurements. This attribute should be set to the value of shunt resistor.

NIDMM_ATTR_TEMP_RTD_A

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
|-----------|--------|----------|---------------------|

| | | | |
|----------|-----|------|------|
| ViReal64 | R/W | None | None |
|----------|-----|------|------|

Description

Specifies the Callendar-Van Dusen A coefficient for RTD scaling when the RTD Type property is set to Custom. The default value is 3.9083e-3 (Pt3851).

NIDMM_ATTR_TEMP_RTD_B

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
|-----------|--------|----------|---------------------|

| | | | |
|----------|-----|------|------|
| ViReal64 | R/W | None | None |
|----------|-----|------|------|

Description

Specifies the Callendar-Van Dusen B coefficient for RTD scaling when the RTD Type property is set to Custom. The default value is $-5.775e-7$ (Pt3851).

NIDMM_ATTR_TEMP_RTD_C

Specific Attribute

Data type **Access** **Coercion** **High-Level Function**

ViReal64 R/W None None

Description

Specifies the Callendar-Van Dusen C coefficient for RTD scaling when the RTD Type property is set to Custom. The default value is $-4.183e-12$ (Pt3851).

NIDMM_ATTR_TEMP_RTD_RES

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
|-----------|--------|----------|---------------------|

| | | | |
|----------|-----|------|------|
| ViReal64 | R/W | None | None |
|----------|-----|------|------|

Description

Specifies the RTD resistance at 0 degrees Celsius. This applies to all supported RTDs, including custom RTDs. The default value is 100 (Ω).

NIDMM_ATTR_TEMP_RTD_TYPE

Specific Attribute

Data type Access Coercion High-Level Function

ViInt32 R/W None None

Description

Specifies the type of RTD used to measure temperature. The default value is NIDMM_VAL_TEMP_RTD_PT3851.

| Enum | Standards | Material | TCR | Typical R_0 (Ω) | Callendar-Van Dusen Coefficient | Notes |
|---------------------------|---|----------|------|-------------------------------|--|----------------------------|
| NIDMM_VAL_TEMP_RTD_PT3851 | IEC-751 DIN 43760 BS 1904 ASTM-E1137 EN-60751 | Platinum | 3851 | 100 Ω 1000 Ω | A = 3.9083 $\times 10^{-3}$ B = - 5.775 $\times 10^{-7}$ C = - 4.183 $\times 10^{-12}$ | Most common RTDs |
| NIDMM_VAL_TEMP_RTD_PT3750 | Low-cost vendor compliant RTD* | Platinum | 3750 | 1000 Ω | A = 3.81 $\times 10^{-3}$ B = - 6.02 $\times 10^{-7}$ C = - 6.0 $\times 10^{-12}$ | Low-cost RTD |
| NIDMM_VAL_TEMP_RTD_PT3916 | JISC 1604 | Platinum | 3916 | 100 Ω | A = 3.9739 $\times 10^{-3}$ B = - 5.870 $\times 10^{-7}$ C = -4.4 $\times 10^{-12}$ | Used in primarily in Japan |
| NIDMM_VAL_TEMP_RTD_PT3920 | US Industrial Standard D-100 American | Platinum | 3920 | 100 Ω | A = 3.9787 $\times 10^{-3}$ B = - 5.8686 $\times 10^{-7}$ C = -4.167 $\times 10^{-12}$ | Low-cost RTD |
| NIDMM_VAL_TEMP_RTD_PT3911 | US Industrial Standard American | Platinum | 3911 | 100 Ω | A = 3.9692 $\times 10^{-3}$ B = - 5.8495 $\times 10^{-7}$ C = -4.233 $\times 10^{-12}$ | Low-cost RTD |
| NIDMM_VAL_TEMP_RTD_PT3928 | ITS-90 | Platinum | 3928 | 100 Ω | A = 3.9888 | The |

| | | | | | |
|------------------------------|--|--|--|---|---------------------------|
| | | | | $\times 10^{-3}$ $B = -$ 5.915×10^{-7} $C = -3.85$ $\times 10^{-12}$ | definition of temperature |
| *No standard. Check the TCR. | | | | | |

NIDMM_ATTR_TEMP_TC_FIXED_REF_JUNC

Specific Attribute

Data type **Access** **Coercion** **High-Level Function**

ViReal64 R/W None None

Description

Specifies the reference junction temperature when a fixed reference junction is used to take a thermocouple measurement. The default value is 25.0 (°C).

NIDMM_ATTR_TEMP_TC_REF_JUNC_TYPE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
|-----------|--------|----------|---------------------|

| | | | |
|--------------|-----|------|------|
| Vilnt32 enum | R/W | None | None |
|--------------|-----|------|------|

Description

Specifies the type of reference junction to be used in the reference junction compensation of a thermocouple. The only supported value, NIDMM_VAL_TEMP_REF_JUNC_FIXED, is fixed.

NIDMM_ATTR_TEMP_TC_TYPE

Specific Attribute

Data type **Access** **Coercion** **High-Level Function**

ViInt32 enum R/W None None

Description

Specifies the type of thermocouple used to measure the temperature. The default value is NIDMM_VAL_TEMP_TC_J.

Defined Values

| | |
|---------------------|---------------------|
| NIDMM_VAL_TEMP_TC_B | Thermocouple type B |
| NIDMM_VAL_TEMP_TC_E | Thermocouple type E |
| NIDMM_VAL_TEMP_TC_J | Thermocouple type J |
| NIDMM_VAL_TEMP_TC_K | Thermocouple type K |
| NIDMM_VAL_TEMP_TC_N | Thermocouple type N |
| NIDMM_VAL_TEMP_TC_R | Thermocouple type R |
| NIDMM_VAL_TEMP_TC_S | Thermocouple type S |
| NIDMM_VAL_TEMP_TC_T | Thermocouple type T |

NIDMM_ATTR_TEMP_THERMISTOR_A

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
|-----------|--------|----------|---------------------|

| | | | |
|----------|-----|------|------|
| ViReal64 | R/W | None | None |
|----------|-----|------|------|

Description

Specifies the Steinhart-Hart A coefficient for thermistor scaling when the Thermistor Type property is set to Custom. The default value is 0.0010295 (44006).

NIDMM_ATTR_TEMP_THERMISTOR_B

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
|-----------|--------|----------|---------------------|

| | | | |
|----------|-----|------|------|
| ViReal64 | R/W | None | None |
|----------|-----|------|------|

Description

Specifies the Steinhart-Hart B coefficient for thermistor scaling when the Thermistor Type property is set to Custom. The default value is 0.0002391 (44006).

NIDMM_ATTR_TEMP_THERMISTOR_C

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
|-----------|--------|----------|---------------------|

| | | | |
|----------|-----|------|------|
| ViReal64 | R/W | None | None |
|----------|-----|------|------|

Description

Specifies the Steinhart-Hart C coefficient for thermistor scaling when the Thermistor Type property is set to Custom. The default value is $1.568e-7$ (44006).

NIDMM_ATTR_TEMP_THERMISTOR_TYPE

Specific Attribute

Data type Access Coercion High-Level Function

ViInt32 R/W None None

Description

Specifies the type of thermistor used to measure the temperature. The default value is NIDMM_VAL_TEMP_THERMISTOR_44006.

| Defined Values | Thermistor Type | Value | 25 °C |
|----------------------------------|-----------------|-------|--------|
| NIDMM_VAL_TEMP_THERMISTOR_CUSTOM | Custom | 0 | — |
| NIDMM_VAL_TEMP_THERMISTOR_44004 | 44004 | 1 | 2.25 k |
| NIDMM_VAL_TEMP_THERMISTOR_44006 | 44006 | 2 | 10 k |
| NIDMM_VAL_TEMP_THERMISTOR_44007 | 44007 | 3 | 5 k |

NIDMM_ATTR_TEMP_TRANSDUCER_TYPE

Specific Attribute

Data type **Access** **Coercion** **High-Level Function**

ViInt32 R/W None None

Description

Specifies the type of device used to measure the temperature. The default value is NIDMM_VAL_4_THERMOCOUPLE.

Defined Values

| | |
|--------------------------|--------------|
| NIDMM_VAL_2_WIRE_RTD | 2-wire RTD |
| NIDMM_VAL_4_WIRE_RTD | 4-wire RTD |
| NIDMM_VAL_4_THERMISTOR | Thermistor |
| NIDMM_VAL_4_THERMOCOUPLE | Thermocouple |

NIDMM_ATTR_RANGE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViInt32 | R/W | Yes | niDMM_ConfigureMeasurementDigits niDMM_ConfigureMeasurementAbsolute |

Description

Specifies the measurement range. Use positive values to represent the absolute value of the maximum expected measurement. The value is in units appropriate for the current value of the [NIDMM_ATTR_FUNCTION](#) attribute. For example, if [NIDMM_ATTR_FUNCTION](#) is set to NIDMM_VAL_DC_VOLTS, the units are volts.



Note The NI 4050, NI 4060, and NI 4065 only support Auto Range when the trigger and sample trigger are set to IMMEDIATE.

Defined Values

| | | |
|--------|---------------------------|---|
| (-1.0) | NIDMM_VAL_AUTO_RANGE_ON | NI-DMM performs an Auto Range before acquiring the measurement. |
| (-2.0) | NIDMM_VAL_AUTO_RANGE_OFF | NI-DMM sets the Range to the current NIDMM_ATTR_AUTO_RANGE_VALUE and uses this range for all subsequent measurements until the measurement configuration is changed. |
| (-3.0) | NIDMM_VAL_AUTO_RANGE_ONCE | NI-DMM performs an Auto Range before acquiring the next measurement. The NIDMM_ATTR_AUTO_RANGE_VALUE is stored and used for all subsequent measurements until the measurement configuration is changed. |

NIDMM_ATTR_RESOURCE_DESCRIPTOR

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the resource descriptor of the instrument.

NIDMM_ATTR_LOGICAL_NAME

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the logical name of the instrument.

NIDMM_ATTR_CHANNEL_COUNT

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | RO | None | None |

Description

Indicates the number of channels that the specific instrument driver supports. For each attribute for which the `IVI_VAL_MULTI_CHANNEL` flag attribute is set, the IVI engine maintains a separate cache value for each channel.

NIDMM_ATTR_SPECIFIC_DRIVER_PREFIX

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

The prefix for the specific instrument driver. The name of each user-callable function in this driver starts with this prefix. The prefix can be up to a maximum of eight characters.

NIDMM_ATTR_INSTRUMENT_FIRMWARE_REVIS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--------------------------------------|
| ViString | RO | None | niDMM_revision_query |

Description

A string containing the instrument firmware revision number.

NIDMM_ATTR_INSTRUMENT_MANUFACTURER

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the manufacturer of the instrument.

NIDMM_ATTR_INSTRUMENT_MODEL

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the instrument model.

NIDMM_ATTR_INSTR_SERIAL_NUMBER

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the serial number of the instrument. This attribute corresponds to the serial number label that is attached to most products.

NIDMM_ATTR_GROUP_CAPABILITIES

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the capabilities and extension groups supported by the specific driver.

NIDMM_ATTR_SUPPORTED_INSTRUMENT_MOD

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the instrument models supported by the specific driver.

NIDMM_ATTR_SPECIFIC_DRIVER_CLASS_SPEC

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| Vilnt32 | RO | None | None |

Description

The major version number of the class specification for the specific driver.

NIDMM_ATTR_SPECIFIC_DRIVER_CLASS_SPEC

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| Vilnt32 | RO | None | None |

Description

The minor version number of the class specification for the specific driver.

NIDMM_ATTR_SPECIFIC_DRIVER_DESCRIPTION

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|------------------|---------------|-----------------|----------------------------|
| ViString | RO | None | None |

Description

A string containing a description of the specific driver.

NIDMM_ATTR_SPECIFIC_DRIVER_VENDOR

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the vendor of the specific driver.

NIDMM_ATTR_CACHE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------------------|
| ViBoolean | R/W | None | InitWithOptions |

ViBoolean

Description

Specifies whether to cache the value of attributes. When caching is enabled, the instrument driver keeps track of the current instrument settings and avoids sending redundant commands to the instrument. Thus, it significantly increases execution speed. The instrument driver can choose to always cache or to never cache particular attributes regardless of the setting of this attribute. The default value is VI_TRUE (1). Use the [niDMM_InitWithOptions](#) function to override this setting.

NIDMM_ATTR_DRIVER_SETUP

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------------------------|
| ViString | RO | None | niDMM_InitWithOptions |

Description

This attribute indicates the Driver Setup string that the user specified when initializing the driver. Some cases exist where the end-user must specify instrument driver options at initialization time. An example of this is specifying a particular instrument model from among a family of instruments that the driver supports. This is useful when using simulation. The end-user can specify driver-specific options through the DriverSetup keyword in the **Option_String** parameter in [niDMM_InitWithOptions](#). If the user does not specify a Driver Setup string, this attribute returns an empty string.

NIDMM_ATTR_INTERCHANGE_CHECK

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViBoolean | R/W | None | niDMM_GetNextInterchangeWarning niDMM_ClearInterchangeWarnings |

Description

Specifies whether to perform interchangeability checking and log interchangeability warnings when you call niDMM functions. Interchangeability warnings indicate that using your application with a different instrument might cause different behavior. Use [niDMM_GetNextInterchangeWarning](#) to extract interchange warnings. Use [niDMM_ClearInterchangeWarnings](#) to clear the list of interchangeability warnings without reading them. Interchangeability checking examines the attributes in a capability group only if you specify a value for at least one attribute within that group. Interchangeability warnings can occur when an attribute affects the behavior of the instrument and you have not set that attribute, or the attribute has been invalidated since you set it.

Defined Values

| | | |
|--------------------|---|-------|
| VI_FALSE (default) | 0 | False |
| VI_TRUE | 1 | True |

NIDMM_ATTR_QUERY_INSTRUMENT_STATUS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------------------------|
| ViBoolean | R/W | None | niDMM_InitWithOptions |

Description

Specifies whether the instrument driver queries the instrument status after each operation. Querying the instrument status is very useful for debugging. After the user program is validated, this attribute can be set to VI_FALSE (0) to disable status checking and maximize performance. The instrument driver can choose to ignore status checking for particular attributes regardless of the setting of this attribute. The default value is VI_TRUE (1). Use [niDMM_InitWithOptions](#) to override this setting.

NIDMM_ATTR_RANGE_CHECK

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------------------------|
| ViBoolean | R/W | None | niDMM_InitWithOptions |

Description

Specifies whether to validate attribute values and function parameters. If enabled, the instrument driver validates the parameter values passed to driver functions. Range checking parameters is very useful for debugging. After the user program is validated, this attribute can be set to VI_FALSE (0) to disable range checking and maximize performance. The default value is VI_TRUE (1). Use the [niDMM_InitWithOptions](#) function to override this setting.

NIDMM_ATTR_RECORD_COERCIONS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViBoolean | R/W | None | None |

Description

Specifies whether the IVI engine keeps a list of the value coercions it makes for ViInt32 and ViReal64 attributes. Call [niDMM_GetNextCoercionRecord](#) to extract and delete the oldest coercion record from the list. The default value is VI_FALSE (0). Use the [niDMM_InitWithOptions](#) function to override this setting.

NIDMM_ATTR_SIMULATE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------------------|
| ViBoolean | R/W | None | InitWithOptions |

Description

Specifies whether or not to simulate instrument driver I/O operations. If simulation is enabled, instrument driver functions perform range checking and call IVI Get and Set functions, but they do not perform instrument I/O. For output parameters that represent instrument data, the instrument driver functions return calculated values. The default value is VI_FALSE (0). Use the [InitWithOptions](#) function to override this setting.



Note Simulate can only be set within the [InitWithOptions](#) function. The attribute value cannot be changed outside of the function.

NIDMM_ATTR_SPECIFIC_DRIVER_MAJOR_VERS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | RO | None | None |

Description

Returns the major version number of this instrument driver.

NIDMM_ATTR_SPECIFIC_DRIVER_MINOR_VERS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | RO | None | None |

Description

The minor version number of this instrument driver.

NIDMM_ATTR_SPECIFIC_DRIVER_REVISION

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string that contains additional version information about this instrument driver.

NIDMM_ATTR_ID_QUERY_RESPONSE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViString | RO | None | None |

Description

A string containing the type of instrument used in the current session.

NIDMM_ATTR_BUFFER_SIZE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

Size in samples of the internal data buffer. Maximum is 134,217,727 (0X7FFFFFFF) samples. When set to NIDMM_VAL_BUFFER_SIZE_AUTO(-1), NI-DMM chooses the buffer size.

NIDMM_ATTR_LATENCY

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

Specifies the number of measurements transferred at a time from the instrument to an internal buffer. When set to NIDMM_VAL_LATENCY_AUTO (-1), NI-DMM chooses the transfer size.

NIDMM_ATTR_SAMPLE_COUNT

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| Vilnt32 | R/W | None | niDMM_ConfigureMultiPoint |

Description

Specifies the number of measurements the DMM takes each time it receives a trigger in a [multiple point acquisition](#).

NIDMM_ATTR_SAMPLE_DELAY_MODE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

For the NI 4060 only, specifies a delay interval after an external sample trigger.

Defined Values

| | | |
|----------------|-------------------|--|
| 0 (default) | IVI compliant | NIDMM_ATTR_SAMPLE_INTERVAL is only used when the Sample Trigger is set to INTERVAL. |
| 1 | Not IVI compliant | NIDMM_ATTR_SAMPLE_INTERVAL is used as a delay after <i>any</i> type of Sample Trigger. |

NIDMM_ATTR_SAMPLE_INTERVAL

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViReal64 | R/W | None | niDMM_ConfigureMultiPoint |

Description

Specifies the amount of time in seconds the DMM waits between measurements. This attribute only applies when the [NIDMM_ATTR_SAMPLE_TRIGGER](#) attribute is set to INTERVAL. The default value (–1) ensures that the DMM settles for a recommended time, which is the same as using an immediate trigger.

The NI 4065 and NI 4070/4071/4072 use the value specified in this attribute as additional delay. On the NI 4065 and NI 4070/4071/4072, the onboard timing resolution is 34.72 ns and the valid range is 0–149 s.

On the NI 4060, the value for this attribute is used as the settling time. When this attribute is set to 0, the NI 4060 does not settle between measurements. The onboard timing resolution is 1 μ s on the NI 4060.

Only positive values are valid when setting the sample interval.



Note The NI 4050 is not supported.

NIDMM_ATTR_SAMPLE_TRIGGER_SLOPE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| VilInt32 | R/W | None | None |

Description

Specifies the edge of the signal from the specified sample trigger source on which the DMM is triggered.

Defined Values

| | | | |
|------------------------|---|--------------------|--|
| Rising Edge | 0 | NIDMM_VAL_POSITIVE | The driver triggers on the rising edge of the trigger signal. |
| Falling Edge (default) | 1 | NIDMM_VAL_NEGATIVE | The driver triggers on the falling edge of the trigger signal. |

NIDMM_ATTR_SAMPLE_TRIGGER

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | niDMM_ConfigureMultiPoint |

Description

Specifies the sample trigger source.



Note To determine which values are supported by each device, refer to the [LabWindows/CVI Trigger Routing](#) section.

NIDMM_ATTR_TRIGGER_COUNT

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | niDMM_ConfigureMultiPoint |

Description

Specifies the number of triggers the DMM receives before returning to the Idle state. This attribute can be set to any positive ViInt32 value for the NI 4065 and NI 4070/4071/4072.

The NI 4050 and NI 4060 only support this attribute being set to 1.

Refer to the [Multiple Point Acquisitions](#) section for more information.

NIDMM_ATTR_MEAS_COMPLETE_DEST

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | niDMM_ConfigureMeasCompleteDest |

Description

Specifies the destination of the measurement complete (MC) signal.



Notes The NI 4050 is not supported.

To determine which values are supported by each device, refer to the [LabWindows/CVI Trigger Routing](#) section.

NIDMM_ATTR_MEAS_DEST_SLOPE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViInt32 | R/W | None | None |

Description

Specifies the polarity of the generated measurement complete signal.

Defined Values

| | | | |
|------------------------|---|--------------------|--|
| Rising Edge | 0 | NIDMM_VAL_POSITIVE | The driver triggers on the rising edge of the trigger signal. |
| Falling Edge (default) | 1 | NIDMM_VAL_NEGATIVE | The driver triggers on the falling edge of the trigger signal. |

NIDMM_ATTR_TRIGGER_DELAY

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---------------------|
| ViReal64 | R/W | None | None |

Description

Specifies the time (in seconds) that the DMM waits after it has received a trigger before taking a measurement. The default value is AUTO DELAY (-1), which means that the DMM waits an appropriate settling time before taking the measurement. (-1) signifies that AUTO DELAY is on, and (-2) signifies that AUTO DELAY is off.

The NI 4065 and NI 4070/4071/4072 use the value specified in this attribute as additional settling time. For the The NI 4065 and NI 4070/4071/4072, the valid range for Trigger Delay is AUTO DELAY (-1) or 0.0-149.0 seconds and the onboard timing resolution is 34.72 ns.

On the NI 4060, if this attribute is set to 0, the DMM does not settle before taking the measurement. On the NI 4060, the valid range for AUTO DELAY (-1) is 0.0-12.0 seconds and the onboard timing resolution is 100 ms.

When using the NI 4050, this attribute must be set to AUTO DELAY (-1).

Use positive values to set the trigger delay in seconds.

NIDMM_ATTR_TRIGGER_SLOPE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| Vilnt32 | R/W | None | niDMM_ConfigureTriggerSlope niDMM_ConfigureSampleTriggerSlope |

Description

Specifies the edge of the signal from the specified trigger source on which the DMM is triggered.

Defined Values

| | | | |
|------------------------------|---|----------|--|
| NIDMM_VAL_POSITIVE | 0 | POSITIVE | The driver triggers on the rising edge of the trigger signal. |
| NIDMM_VAL_NEGATIVE (default) | 1 | NEGATIVE | The driver triggers on the falling edge of the trigger signal. |

NIDMM_ATTR_TRIGGER_SOURCE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViInt32 | R/W | None | niDMM_ConfigureTrigger |

Description

Specifies the trigger source. When [niDMM_Initiate](#) is called, the DMM waits for the trigger specified with this attribute. After it receives the trigger, the DMM waits the length of time specified with the [NIDMM_ATTR_TRIGGER_DELAY](#) attribute. The DMM then takes a measurement.



Note To determine which values are supported by each device, refer to the [LabWindows/CVI Trigger Routing](#) section.

NIDMM_ATTR_WAVEFORM_COUPLING

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|---|
| ViInt32 | R/W | None | niDMM_ConfigureWaveformCoupling |

Description

For the NI 4070/4071/4072 only, specifies the coupling during a waveform acquisition.

Defined Values

| | | |
|--|---|-------------|
| NIDMM_VAL_WAVEFORM_COUPLING_AC | 0 | AC coupling |
| NIDMM_VAL_WAVEFORM_COUPLING_DC (default) | 1 | DC coupling |

NIDMM_ATTR_WAVEFORM_POINTS

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViInt32 | R/W | None | niDMM_ConfigureWaveformAcquisition |

Description

For the NI 4070/4071/4072 only, specifies the number of points to acquire in a waveform acquisition.

NIDMM_ATTR_WAVEFORM_RATE

Specific Attribute

| Data type | Access | Coercion | High-Level Function |
|-----------|--------|----------|--|
| ViReal64 | R/W | Yes | niDMM_ConfigureWaveformAcquisition |

Description

For the NI 4070/4071/4072 only, specifies the rate of the waveform acquisition in Samples per second (S/s). The valid range is 10.0–1,800,000 S/s. Values are coerced to the closest integer divisor of 1,800,000. The default value is 1,800,000.

niDMM_GetDigitsOfPrecision (Obsolete)

Usage

```
ViStatus = niDMM_GetDigitsOfPrecision(ViSession Instrument_Handle,  
ViReal64 Digits)
```

Purpose

Returns the digits of precision calculated from the range and resolution information specified in [niDMM_ConfigureMeasurementDigits](#).

Parameters

| Name | Type | Description |
|--------------------------|-----------|--|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle parameter from niDMM_init or niDMM_InitWithOptions . |
| Digits | ViReal64 | This indicator shows the digits of precision that correspond to the range/resolution selection made in niDMM_ConfigureMeasurementDigits . Values include 3½, 4½, 5½, 6½, and 7½. 6½ is available for the NI 4070/4071/4072 only, and 7½ is available for the resistance and DCV modes of the NI 4071 only. |

Return Value

Reports the status of this operation. To obtain a text description of the status code, call [niDMM_Error_Message](#). To obtain additional information concerning the error condition, use [niDMM_GetError](#).

Purpose

For the NI 4070/4071/4072 only, allows the DMM to compensate for gain drift since the last external calibration or self-calibration. When **ADC_Calibration** is ON, the DMM measures an internal reference to calculate the correct gain for the measurement. When **ADC_Calibration** is OFF, the DMM does not compensate for changes to the gain.

Parameters

Input

| Name | Type | Description |
|--------------------------|-----------|---|
| Instrument_Handle | ViSession | Identifies a particular instrument session. You obtain the Instrument_Handle from niDMM_InitWithOptions . The default is None. |
| ADC_Calibration | ViInt32 | Specifies the ADC_Calibration setting. The driver sets NIDMM_ATTR_NIDMM_VAL_ADC_CALIBRATION_ON enables ADC_Calibration . NI-4070/4071/4072 driver disables ADC_Calibration . If you set the value to NIDMM_VAL_ADC_CALIBRATION_OFF , the driver determines whether to enable ADC_Calibration based on the measurement configuration. If you configure the NI 4070/4071/4072 for a 6½-digit and the driver enables ADC Calibration. For all other measurement configurations, the driver disables ADC_Calibration . |

| Name |
|--|
| NIDMM_VAL_ADC_CALIBRATION_AUTO (default) |
| NIDMM_VAL_ADC_CALIBRATION_OFF |
| NIDMM_VAL_ADC_CALIBRATION_ON |

Return Value

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------------|----------|--|
| Status | ViStatus | Reports the Status of this operation. To obtain a text description of the status code, call niDMM_error_message . To obtain additional information concerning the error condition, use niDMM_GetError . |
|---------------|----------|--|