IVI-4.2: IviDmm Class Specification

March 8, 2002 Edition
Revision 3.0

Important Information

The IviDmm Class Specification (IVI-4.2) is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at `www.ivifoundation.org`, or contact the IVI Foundation at 2515 Camino del Rio South, Suite 340, San Diego, CA 92108.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation through email at `ivilistserver@ivifoundation.org`, through the web site at `www.ivifoundation.org`, or you can write to the 2515 Camino del Rio South, Suite 340, San Diego, CA 92108.

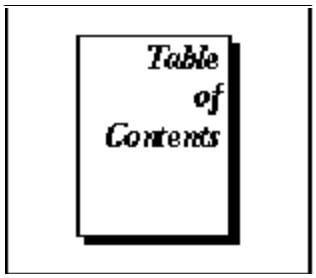
Warranty

The IVI Foundation and its member companies make no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The IVI Foundation and its member companies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Trademarks

Product and company names listed are trademarks or trade names of their respective companies.

No investigation has been made of common-law trademark rights in any work.



IviDmm Class Specification

IviDmm Revision History

This section is an overview of the revision history of the IviDmm specification.

Table 1-1 IviDmm Class Specification Revisions

Revision Number	Date of Revision	Revision Notes
Revision 0.2	April 15, 1997	Original draft.
Revision 0.3	May 15, 1997	This edition reflects the addition of the new IviDmm trigger model, the extension defaults, interchangeability checking, and guidelines for specific driver development.
Revision 0.4	July 24, 1997	This edition incorporates the channel parameter into the API as well as which attributes are channel-based.
Revision 0.5	August 12, 1997	This edition incorporates edits based on user feedback and adds introductory text.
Revision 0.6	September 24, 1997	This edition incorporates the new specification style.
Revision 0.7	June 26, 1998	This edition refines the existing documentation, and adds guidelines for specific and class drivers, attribute ID definitions, and attribute value definitions.
Revision 1.0	August 21, 1998	Technical Publications review and edit. Changes to template information.
Revision 2.0	November 22, 1999	This edition refines the organization of the specification based on feedback at the July 1999 IVI Foundation meeting. It replaces the IviDmm Miscellaneous Capabilities extension group by defining a new extension for every attribute in the group. It also defines new extension groups for AC, Frequency, and Temperature measurements.
Revision 3.0 (VC1)	July 27, 2001	Reformatted to adhere to formatting specified in *IVI – 3.4: API Style Guide*. Added COM interface.
Revision 3.0 (VC2)	November 8, 2001	Updates from Boston meeting, Steve Greer's review feedback, J. Harvey's COM feedback, and internal review.
Revision 3.0 (VC3)	December 18, 2001	Updates from Austin meeting. Some IDL changes still needed.
Revision 3.0(VC4)	January 30, 2002	Updates from review feedback. Some IDL changes still needed.
Revision 3.0 (VC5)	March 8, 2002	Update TimeOut parameter names with MaxTimeMilliseconds. Updated "IviDmmTriggerSourceSoftware" to agree with IDL and Trigger source value of

Table 1-1 IviDmm Class Specification Revisions

		"IviDmmTriggerSourceSwTrigFunc". Added updated IDL.
Revision 3.0	April 16, 2002	Accepted changes. Removed draft. Added release IDL.

1. Overview of the IviDmm Specification

1.1 Introduction

This specification defines the IVI class for digital multimeters (DMMs). The IviDmm class is designed to support the typical DMM as well as common extended functionality found in more complex instruments. This section summarizes the *IviDmm Class Specification* and contains general information that the reader might need in order to understand, interpret, and implement aspects of this specification. These aspects include the following:

? IviDmm class overview

? The definitions of terms and acronyms

? References

1.2 IviDmm Class Overview

This specification defines the IVI class for digital multimeters (DMMs). The IviDmm class is designed to support the typical DMM as well as common extended functionality found in more complex instruments. The IviDmm class conceptualizes a DMM as an instrument that can measure scalar quantities of an input signal and can be applied to a wide variety of instruments. Typically the measured quantity is a voltage (AC and DC), current, or resistance. However, the IviDmm class can support instruments that measure other quantities such as temperature and frequency etc.

The IviDmm class is divided into a base capability group and several extension groups. The base capability group is used to configure a DMM for a typical measurement (this includes setting the measurement function, desired range, desired resolution, and trigger source), initiating that measurement, and returning a measured value. The IviDmm base capability group is described in Section 4, *IviDmmBase Capability Group*.

Many DMMs support measurement types that require additional parameters to be configured, such as the minimum and maximum frequency of the input signal for AC measurements. The IviDmm class defines extension groups for each measurement type that requires these additional parameters.

The IviDmm class also defines an extension group called IviDmmMultiPoint. The IviDmmMultiPoint extension group is used to configure DMMs that can acquire multiple measurements based on multiple triggers and take multiple measurements per trigger. This type of instrument used in conjunction with a scanner is typically used to implement a scanning DMM. The IviDmmMultiPoint extensions are described in Section 11, *IviDmmMultiPoint Extension Group*.

In addition, the IviDmm class defines extension groups that configure advanced settings such as auto-zero and power line frequency, or return additional information about the current state of the instrument such as aperture time. These extension groups are defined in Sections 12 through 17.

1.3 References

Several other documents and specifications are related to this specification. These other related documents are as follows:

? IVI-3.1: Driver Architecture Specification

? IVI-3.2: Inherent Capabilities Specification

? IVI-3.3: Standard Cross Class Capabilities Specification

? IVI- 5.0: Glossary

1.4 Definitions of Terms and Acronyms

This section defines terms and acronyms that are specific to the IviDmm class.

Temperature Transducer	A device that converts thermal energy into electrical energy. Used for measuring temperature.
Reference Junction	Also known as the Cold Junction. The junction of a thermocouple that is kept at a known temperature or one for which the temperature may be measured.

Refer to *IVI–5.0: Glossary* for a description of more terms used in this specification.

2. IviDmm Class Capabilities

2.1 Introduction

The IviDmm specification divides DMM capabilities into a base capability group and multiple extension capability groups. Each capability group is discussed in a separate section. This section defines names for each capability group and gives an overview of the information presented for each capability group.

2.2 IviDmm Group Names

The capability group names for the IviDmm class are defined in the following table. The Group Name is used to represent a particular capability group and is returned as one of the possible group names from the Class Group Capabilities attribute.

Table 2-1. IviDmm Group Names

Group Name	Description
`IviDmmBase`	Base Capability Group: DMM that complies with the IviDmmBase Capability Group.
`IviDmmACMeasurement`	Extension Group: DMM with the capability to measure AC voltage, AC current, AC plus DC voltage, and AC plus DC current.
`IviDmmFrequencyMeasurement`	Extension Group: DMM with the capability to measure frequency and period.
`IviDmmTemperatureMeasurement`	Extension Group: DMM with the capability to measure temperature.
`IviDmmThermocouple`	Extension Group: DMM with the capability to measure temperature using a thermocouple.
`IviDmmResistanceTemperatureDevice`	Extension group: DMM with the capability to measure temperature using a temperature resistance device.
`IviDmmThermistor`	Extension group: DMM with the capability to measure temperature using a thermistor.
`IviDmmMultiPoint`	Extension Group: DMM with the capability to accept multiple triggers and acquire multiple samples per trigger.
`IviDmmTriggerSlope`	Extension Group: DMM with the capability to specify the trigger slope.
`IviDmmSoftwareTrigger`	Extension Group: DMM with the capability to send a software trigger.
`IviDmmDeviceInfo`	Extension Group: DMM with the capability to return extra information concerning the instrument’s state such as aperture time.
`IviDmmAutoRangeValue`	Extension Group: DMM with the capability to return the actual range when auto ranging.

Table 2-1. IviDmm Group Names

Group Name	Description
`IviDmmAutoZero`	Extension Group: DMM with the capability to take an auto-zero reading.
`IviDmmPowerLineFrequency`	Extension Group: DMM with the capability to specify the power line frequency.

2.3 Repeated Capability Names

The IviDmm Class Specification does not define repeated capabilities.

3. General Requirements

This section describes the general requirements a specific driver must meet in order to be compliant with this specification. In addition, it provides general requirements that specific drivers must meet in order to comply with a capability group, attribute, or function.

3.1 Minimum Class Compliance

To be compliant with the IviDmm Class Specification, an IVI specific driver shall conform to all of the requirements for an IVI class-compliant specific driver as specified in *IVI-3.1: Driver Architecture Specification*, implement the inherent capabilities that *IVI-3.2: Inherent IVI Capabilities Specification* defines, and implements the IviDmmBase capability group.

3.1.1 Disable

Refer to *IVI-3.2: Inherent Capabilities Specification* for the prototype of this function. The IviDmm specification does not define additional requirements on the Disable function.

3.2 Capability Group Compliance

IVI-3.1: Driver Architecture Specification defines the general rules for a specific driver to be compliant with a capability group.

4. IviDmmBase Capability Group

4.1 Overview

The IviDmmBase Capability Group supports DMMs that take a single measurement at a time. The IviDmmBase Capability Group defines attributes and their values to configure the type of measurement and how the measurement is to be performed. These attributes include the measurement function, range, resolution, and trigger source. The IviDmmBase capability group also includes functions for configuring the DMM as well as initiating and retrieving measurements.

4.2 IviDmmBase Attributes

The IviDmmBase capability group defines the following attributes:

? Function

? Range

? Resolution Absolute

? Trigger Delay

? Trigger Source

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions*.

4.2.1 Function

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Measurement

COM Property Name

Function

COM Enumeration Name

IviDmmFunctionEnum

C Constant Name

IVIDMM_ATTR_FUNCTION

Description

Specifies the measurement function.

The value of this attribute determines the units for the Range and Resolution Absolute attributes, and the measurement values that are returned by the Read, Read Multiple Point, Fetch, and Fetch Multiple Point functions.

Defined Values

Name	*Description*		
		Language	*Identifier*
DC Volts	Sets the DMM to measure DC voltage.		
		C	IVIDMM_VAL_DC_VOLTS
		COM	IviDmmFunctionDCVolts
AC Volts	Sets the DMM to measure AC voltage. Use the IviDmmACMeasurement extension group to configure additional parameters for this measurement type.		
		C	IVIDMM_VAL_AC_VOLTS
		COM	IviDmmFunctionACVolts
DC Current	Sets the DMM to measure DC current..		
		C	IVIDMM_VAL_DC_CURRENT
		COM	IviDmmFunctionDCCurrent
AC Current	Sets the DMM to measure AC current. Use the IviDmmACMeasurement extension group to configure additional parameters for this measurement type.		
		C	IVIDMM_VAL_AC_CURRENT
		COM	IviDmmFunctionACCurrent
2 Wire Resistance	Sets the DMM to measure 2-wire resistance.		
		C	IVIDMM_VAL_2_WIRE_RES
		COM	IviDmmFunction2WireRes

Name	*Description*		
		Language	*Identifier*
4 Wire Resistance	Sets the DMM to measure 4-wire resistance.		
		C	IVIDMM_VAL_4_WIRE_RES
		COM	IviDmmFunction4WireRes
AC Plus DC Volts	Sets the DMM to measure AC plus DC voltage. Use the IviDmmACMeasurement extension group to configure additional parameters for this measurement type.		
		C	IVIDMM_VAL_AC_PLUS_DC_VOLTS
		COM	IviDmmFunctionACPlusDCVolts
AC Plus DC Current	Sets the DMM to measure AC plus DC current. Use the IviDmmACMeasurement extension group to configure additional parameters for this measurement type.		
		C	IVIDMM_VAL_AC_PLUS_DC_CURRENT
		COM	IviDmmFunctionACPlusDCCurrent
Frequency	Sets the DMM to measure frequency. Use the IviDmmFrequencyMeasurement extension group to configure additional parameters for this measurement type.		
		C	IVIDMM_VAL_FREQ
		COM	IviDmmFunctionFreq
Period	Sets the DMM to measure period. Use the IviDmmFrequencyMeasurement extension group to configure additional parameters for this measurement type.		
		C	IVIDMM_VAL_PERIOD
		COM	IviDmmFunctionPeriod
Temperature	Sets the DMM to measure temperature. Use the IviDmmTemperatureMeasurement extension group to configure additional parameters for this measurement type.		
		C	IVIDMM_VAL_TEMPERATURE
		COM	IviDmmFunctionTemperature

Compliance Notes

1. If an IviDmm specific driver implements any of the defined values in the following table, it shall also implement the corresponding capability group:

Value	Required Capability Group
AC Volts	IviDmmACMeasurement
AC Current	IviDmmACMeasurement
AC Plus DC Volts	IviDmmACMeasurement
AC Plus DC Current	IviDmmACMeasurement
Frequency	IviDmmFrequencyMeasurement

Period	IviDmmFrequencyMeasurement
Temperature	IviDmmTemperatureMeasurement

2. If an IVI-C IviDmm specific driver defines additional values for this attribute, the actual values shall be greater than or equal to Function Specific Extension Base.
3. If an IVI-C IviDmm class driver defines additional values for this attribute, the actual values shall be greater than or equal to Function Class Extension Base and less than Function Specific Extension Base.
4. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Function Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Function Specific Extension Base and Function Class Extension Base.

4.2.2 Range

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	Up	Configure Measurement

COM Property Name

Range

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_RANGE

Description

Specifies the measurement range. Positive values represent the absolute value of the maximum measurement expected. The specific driver is expected to coerce this value to the appropriate range for the instrument. Negative values represent the Auto Range mode.

There is a dependency between the Range attribute and the Resolution Absolute attribute. The allowed values of Resolution Absolute attribute depend on the Range attribute.

Typically, when the value of the Range attribute changes, the instrument settings that correspond to the Resolution Absolute attribute change as well. This is true regardless of how the change of measurement range occurs.

There are two possible ways that the measurement range can change. The application program can set the value of the Range attribute. Or, the instrument changes the measurement range because Range attribute is set to Auto Range On and the input signal changes. In both cases, the instrument resolution is likely to change.

The value of the Function attribute determines the units for this attribute. The following table shows the defined values for the Function attribute and the corresponding units for the Range attribute.

Values for FUNCTION ATTRIBUTE	Units for RANGE ATTRIBUTE
DC Volts	Volts
AC Volts	Volts RMS
DC Current	Amps
AC Current	Amps
2 Wire Resistance	Ohms
4 Wire Resistance	Ohms
AC Plus DC Volts	Volts
AC Plus DC Current	Amps
Frequency	Hertz
Period	Seconds
Temperature	Degrees Celsius

Defined Values

Name	*Description*		
		Language	*Identifier*
Auto Range On	Sets the DMM to calculate the range before each measurement automatically. You can obtain the actual range the DMM is currently using by getting the value of the Auto Range Value attribute in the `IviDmmAutoRangeValue` extension group. Setting this attribute to a manual range or to Auto Range Off disables auto-ranging.		
		C	`IVIDMM_VAL_AUTO_RANGE_ON`
		COM	`IviDmmAutoRangeOn`
Auto Range Off	Disables auto-ranging. The DMM sets the range to the value it most recently calculated. Further queries of this attribute return the actual range.		
		C	`IVIDMM_VAL_AUTO_RANGE_OFF`
		COM	`IviDmmAutoRangeOff`
Auto Range Once	Sets the DMM to calculate the range before the next measurement. The DMM uses this range value for all subsequent measurements. Further queries of this attribute return the actual range.		
		C	`IVIDMM_VAL_AUTO_RANGE_ONCE`
		COM	`IviDmmAutoRangeOnce`

Compliance Notes

1. If an IVI-C IviDmm specific driver defines additional values for this attribute, the magnitude of the actual values shall be greater than or equal to Range Specific Extension Base.
2. If an IVI-C IviDmm class driver defines additional values for this attribute, the magnitude of the actual values shall be greater than or equal to Range Class Extension Base and less than Range Specific Extension Base.
3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Range Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Range Specific Extension Base and Range Class Extension Base.

4.2.3 Resolution Absolute

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	Down	Configure Measurement

COM Property Name

Resolution

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_RESOLUTION_ABSOLUTE

Description

Specifies the measurement resolution in absolute units.

The value of the Function attribute determines the units for this attribute. The following table shows the defined values for the Function attribute and the corresponding units for the Resolution Absolute attribute.

Values for FUNCTION ATTRIBUTE	Units for RESOLUTION ABSOLUTE ATTRIBUTE
DC Volts	Volts
AC Volts	Volts RMS
DC Current	Amps
AC Current	Amps
2 Wire Resistance	Ohms
4 Wire Resistance	Ohms
AC Plus DC Volts	Volts
AC Plus DC Current	Amps
Frequency	Hertz
Period	Seconds
Temperature	Degrees Celsius

4.2.4 Trigger Delay

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	Note	Configure Trigger

Note

Many DMMs have a small, non-zero value as the minimum value for this attribute. To configure the instrument to use the shortest trigger delay, the user can specify a value of zero for this attribute. Therefore, the specific driver must coerce any value between zero and the minimum value to the minimum value. No other coercion is allowed on this attribute.

COM Property Name

Trigger.Delay

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_TRIGGER_DELAY

Description

Specifies the length of time between when the DMM receives the trigger and when it takes a measurement. Use positive values to set the trigger delay in seconds. Negative values are reserved for the auto delay mode.

Defined Values

Name	***Description***		
		Language	***Identifier***
Auto Delay On	Sets the DMM to calculate the trigger delay before each measurement. Setting this attribute to a manual trigger delay or Auto Delay Off disables the auto delay mode.		
		C	IVIDMM_VAL_AUTO_DELAY_ON
		COM	IviDmmTriggerDelayAutoDelayOn
Auto Delay Off	Stops the DMM from calculating the trigger delay. Sets the trigger delay to the last trigger delay the DMM calculated. Note: After the user sets this attribute to Auto Delay Off, further queries of this attribute returns the actual delay.		
		C	IVIDMM_VAL_AUTO_DELAY_OFF
		COM	IviDmmTriggerDelayAutoDelayOff

Compliance Notes

1. If an IVI-C IviDmm specific driver defines additional values for this attribute, the magnitude of the actual values shall be greater than or equal to Trigger Delay Specific Extension Base.

2. If an IVI-C IviDmm class driver defines additional values for this attribute, the magnitude of the actual values shall be greater than or equal to Trigger Delay Class Extension Base and less than Trigger Delay Specific Extension Base.
3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Trigger Delay Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Trigger Delay Specific Extension Base and Trigger Delay Class Extension Base.

4.2.5 Trigger Source

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	ConfigureTrigger

COM Property Name

Trigger.Source

COM Enumeration Name

IviDmmTriggerSourceEnum

C Constant Name

IVIDMM_ATTR_TRIGGER_SOURCE

Description

Specifies the trigger source.

Defined Values

Name	*Description*		
		Language	*Identifier*
Immediate	The DMM exits the Wait-For-Trigger state immediately after entering. It does not wait for a trigger of any kind.		
		C	IVIDMM_VAL_IMMEDIATE
		COM	IviDmmTriggerSourceImmediate
External	The DMM exits the Wait-For-Trigger state when a trigger occurs on the external trigger input.		
		C	IVIDMM_VAL_EXTERNAL
		COM	IviDmmTriggerSourceExternal
Software Trigger	The DMM exits the Wait-For-Trigger state when the Send Software Trigger function executes. Refer to the Standardized Cross Class Capabilities specification for a complete description of this value and the Send Software Trigger function.		
		C	IVIDMM_VAL_SOFTWARE_TRIG
		COM	IviDmmTriggerSourceSwTrigFunc
TTL0	The DMM exits the Wait-For-Trigger state when it receives a trigger on TTL0.		
		C	IVIDMM_VAL_TTL0
		COM	IviDmmTriggerSourceTTL0
TTL1	The DMM exits the Wait-For-Trigger state when it receives a trigger on TTL1.		
		C	IVIDMM_VAL_TTL1
		COM	IviDmmTriggerSourceTTL1

Name	*Description*		
		Language	*Identifier*
TTL2	The DMM exits the Wait-For-Trigger state when it receives a trigger on TTL2.		
		C	`IVIDMM_VAL_TTL2`
		COM	`IviDmmTriggerSourceTTL2`
TTL3	The DMM exits the Wait-For-Trigger state when it receives a trigger on TTL3.		
		C	`IVIDMM_VAL_TTL3`
		COM	`IviDmmTriggerSourceTTL3`
TTL4	The DMM exits the Wait-For-Trigger state when it receives a trigger on TTL4.		
		C	`IVIDMM_VAL_TTL4`
		COM	`IviDmmTriggerSourceTTL4`
TTL5	The DMM exits the Wait-For-Trigger state when it receives a trigger on TTL5.		
		C	`IVIDMM_VAL_TTL5`
		COM	`IviDmmTriggerSourceTTL5`
TTL6	The DMM exits the Wait-For-Trigger state when it receives a trigger on TTL6.		
		C	`IVIDMM_VAL_TTL6`
		COM	`IviDmmTriggerSourceTTL6`
TTL7	The DMM exits the Wait-For-Trigger state when it receives a trigger on TTL7.		
		C	`IVIDMM_VAL_TTL7`
		COM	`IviDmmTriggerSourceTTL7`
ECL0	The DMM exits the Wait-For-Trigger state when it receives a trigger on ECL0.		
		C	`IVIDMM_VAL_ECL0`
		COM	`IviDmmTriggerSourceECL0`
ECL1	The DMM exits the Wait-For-Trigger state when it receives a trigger on ECL1.		
		C	`IVIDMM_VAL_ECL1`
		COM	`IviDmmTriggerSourceECL1`
PXI Star	The DMM exits the Wait-For-Trigger state when it receives a trigger on the PXI Star trigger bus.		
		C	`IVIDMM_VAL_PXI_STAR`
		COM	`IviDmmTriggerSourcePXIStar`
RTSI 0	The DMM exits the Wait-For-Trigger state when it receives a trigger on RTSI0.		
		C	`IVIDMM_VAL_RTSI_0`
		COM	`IviDmmTriggerSourceRTSI0`

Name	*Description*		
		Language	*Identifier*
RTSI 1	The DMM exits the Wait-For-Trigger state when it receives a trigger on RTSI1.		
		C	`IVIDMM_VAL_RTSI_1`
		COM	`IviDmmTriggerSourceRTSI1`
RTSI 2	The DMM exits the Wait-For-Trigger state when it receives a trigger on RTSI2.		
		C	`IVIDMM_VAL_RTSI_2`
		COM	`IviDmmTriggerSourceRTSI2`
RTSI 3	The DMM exits the Wait-For-Trigger state when it receives a trigger on RTSI3.		
		C	`IVIDMM_VAL_RTSI_3`
		COM	`IviDmmTriggerSourceRTSI3`
RTSI 4	The DMM exits the Wait-For-Trigger state when it receives a trigger on RTSI4.		
		C	`IVIDMM_VAL_RTSI_4`
		COM	`IviDmmTriggerSourceRTSI4`
RTSI 5	The DMM exits the Wait-For-Trigger state when it receives a trigger on RTSI5.		
		C	`IVIDMM_VAL_RTSI_5`
		COM	`IviDmmTriggerSourceRTSI5`
RTSI 6	The DMM exits the Wait-For-Trigger state when it receives a trigger on RTSI6.		
		C	`IVIDMM_VAL_RTSI_6`
		COM	`IviDmmTriggerSourceRTSI6`

Compliance Notes

1. If an IviDmm specific driver implements any of the defined values in the following table, it shall also implement the corresponding capability group:

Value	Required Capability Group
Software Trigger	IviDmmSoftwareTrigger

2. If an IVI-C IviDmm specific driver defines additional values for this attribute, the actual values shall be greater than or equal to Trigger Source Specific Extension Base.
3. If an IVI-C IviDmm class driver defines additional values for this attribute, the actual values shall be greater than or equal to Trigger Source Class Extension Base and less than Trigger Source Specific Extension Base.
4. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Trigger Source Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Trigger Source Specific Extension Base and Trigger Source Class Extension Base.

4.3 IviDmmBase Functions

The IviDmmBase capability group defines the following functions:

? Abort

? Configure Measurement

? Configure Trigger

? Fetch

? Initiate

? Is Over Range

? Read

This section describes the behavior and requirements of each function.

4.3.1 Abort

Description

This function aborts a previously initiated measurement and returns the DMM to the idle state.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. To check the instrument status, call the Error Query function at the conclusion of the sequence.

COM Method Prototype

```
HRESULT Measurement.Abort();
```

C Prototype

```
ViStatus IviDmm_Abort (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.2 Configure Measurement

Description

This function configures the common attributes of the DMM. These attributes include the measurement function, maximum range, and the resolution of the DMM.

If the value of the `range` parameter is Auto Range On, then the `resolution` parameter is ignored and the Resolution Absolute attribute is not set.

COM Method Prototype

```
HRESULT Configure([in] IviDmmFunctionEnum Function,
                  [in] DOUBLE Range,
                  [in] DOUBLE Resolution);
```

C Prototype

```
ViStatus IviDmm_ConfigureMeasurement (ViSession Vi,
                                      ViInt32 Function,
                                      ViReal64 Range,
                                      ViReal64 Resolution);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Function`	Specifies the measurement function. The driver uses this value to set the Function attribute. See the attribute description for more details.	`ViInt32`
`Range`	Specifies the measurement range. The driver uses this value to set the Range attribute. See the attribute description for more details.	`ViReal64`
`Resolution`	Specifies the resolution. The driver uses this value to set the Resolution Absolute attribute. See the attribute description for more details.	`ViReal64`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.3 Configure Trigger

Description

This function configures the common DMM trigger attributes. These attributes include the trigger source and the trigger delay.

COM Method Prototype

```
HRESULT Trigger.Configure([in] IviDmmTriggerSourceEnum TriggerSource,
                          [in] DOUBLE TriggerDelay);
```

C Prototype

```
ViStatus IviDmm_ConfigureTrigger (ViSession Vi,
                                  ViInt32 TriggerSource,
                                  ViReal64 TriggerDelay);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`TriggerSource`	Specifies the trigger source. The driver uses this value to set the Trigger Source attribute. See the attribute description for more details.	`ViInt32`
`TriggerDelay`	Specifies the trigger delay. The driver uses this value to set the Trigger Delay attribute. See the attribute description for more details.	`ViReal64`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.4 Fetch

Description

This function returns the measured value from a measurement that the Initiate function initiates. After this function executes, the `Reading` parameter contains an actual reading or a value indicating that an overrange condition occurred.

If an overrange condition occurs, the reading parameter contains an IEEE defined NaN (Not a Number) value and the function returns the Over Range completion code. The user may test the measurement value for overrange with the Is Over Range function.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. To check the instrument status, call the Error Query function at the conclusion of the sequence.

In most instrument classes, there is a programmatic way to determine when a measurement has completed and data is available. Therefore, a `MaxTimeMilliseconds` parameter is not needed in the Fetch function for these classes. This is not true for the majority of DMMs. The `MaxTimeMilliseconds` parameter specifies how long to wait in the Fetch operation since it is possible that no data is available or the trigger event did not occur.

The value of the Function attribute determines the units for the `Reading` parameter. The following table shows the defined values for the Function attribute and the corresponding units for the `Reading` parameter.

Values for FUNCTION ATTRIBUTE	Units for reading parameter
DC Volts	Volts
AC Volts	Volts RMS
DC Current	Amps
AC Current	Amps
2 Wire Resistance	Ohms
4 Wire Resistance	Ohms
AC Plus DC Volts	Volts
AC Plus DC Current	Amps
Freq	Hertz
Period	Seconds
Temperature	Degrees Celsius

This function is not guaranteed to return valid data if the user performs other operations on the instrument after the call to Initiate and prior to calling this function. This includes other calls to Fetch.

COM Method Prototype

```
HRESULT Measurement.Fetch([in] LONG MaxTimeMilliseconds,
                          [out, retval] DOUBLE* Reading);
```

C Prototype

```
ViStatus IviDmm_Fetch (ViSession Vi,
                       ViInt32  MaxTimeMilliseconds,
                       ViReal64 *Reading);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
MaxTimeMilliseconds	Specifies the maximum length of time allowed for the function to complete in milliseconds.	ViInt32

Outputs	Description	Base Type
Reading	Measurement value.	ViReal64

Defined Values for the MaxTimeMilliseconds Parameter

Name	*Description*		
		Language	*Identifier*
Max Time Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		C	IVIDMM_VAL_MAX_TIME_IMMEDIATE
		COM	IviDmmMaxTimeImmediate
Max Time Infinite	The function waits indefinitely for the measurement to complete.		
		C	IVIDMM_VAL_MAX_TIME_INFINITE
		COM	IviDmmMaxTimeInfinite

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Over Range	Overrange warning
Max Time Exceeded	Max time exceeded before the operation completed

Compliance Notes

An IviDmm specific driver is not required to implement the Max Time Immediate or the Max Time Infinite defined values for the MaxTimeMilliseconds parameter to be compliant with the IviDmmBase Capability group.

4.3.5 Initiate

Description

This function initiates a measurement. When this function executes, the DMM leaves the idle state and waits for a trigger.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. To check the instrument status, call the Error Query function at the conclusion of the sequence.

COM Method Prototype

```
HRESULT Measurement.Initiate();
```

C Prototype

```
ViStatus IviDmm_Initiate (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.6 Is Over Range

Description

This function takes a measurement value that you obtain from one of the Read or Fetch functions and determines if the value is a valid measurement value or a value indicating that an overrange condition occurred.

COM Method Prototype

```
HRESULT Measurement.IsOverRange([in] DOUBLE MeasurementValue,
                                [out, retval] VARIANT_BOOL* IsOverRange);
```

C Prototype

```
ViStatus IviDmm_IsOverRange (ViSession Vi,
                             ViReal64 MeasurementValue,
                             ViBoolean *IsOverRange);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`MeasurementValue`	Pass the measurement value you obtain from one of the Read or Fetch functions.	`ViReal64`

Outputs	Description	Base Type
`IsOverRange`	Returns whether the `measurementValue` is a valid measurement or a value indicating that the DMM encountered an overrange condition. Valid Return Values: True - The `measurementValue` indicates that an overrange condition occurred. False - The `measurementValue` is a valid measurement.	`ViBoolean`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.7 Read

Description

This function initiates a measurement, waits until the DMM has returned to the idle state, and returns the measured value.

After this function executes, the `Reading` parameter contains an actual reading or a value indicating that an overrange condition occurred. If an overrange condition occurs, the `Reading` parameter contains an IEEE defined NaN (Not a Number) value and the function returns Over Range. The end-user may test the measurement value for overrange with the Is Over Range function.

The value of the Function attribute determines the units for the `Reading` parameter. Refer to Section 4.3.3, *`Fetch`*, for more details.

COM Method Prototype

```
HRESULT Measurement.Read([in] LONG MaxTimeMilliseconds,
                         [out, retval] DOUBLE* Reading);
```

C Prototype

```
ViStatus IviDmm_Read (ViSession Vi,
                      ViInt32  MaxTimeMilliseconds,
                      ViReal64 *Reading);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`MaxTimeMilliseconds`	Specifies the maximum length of time allowed for the function to complete in milliseconds.	`ViInt32`

Outputs	Description	Base Type
`Reading`	Measurement value.	`ViReal64`

Defined Values for the MaxTimeMilliseconds Parameter

Name	*Description*		
		Language	***Identifier***
Max Time Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		C	`IVIDMM_VAL_MAX_TIME_IMMEDIATE`
		COM	`IviDmmMaxTimeImmediate`
Max Time Infinite	The function waits indefinitely for the measurement to complete.		
		C	`IVIDMM_VAL_MAX_TIME_INFINITE`
		COM	`IviDmmMaxTimeInfinite`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Over Range	Overrange warning
Max Time Exceeded	Max time exceeded before the operation completed

Compliance Notes

An IviDmm specific driver is not required to implement the Max Time Immediate or the Max Time Infinite defined values for the `MaxTimeMilliseconds` parameter to be compliant with the IviDmmBase extension group.

4.4 IviDmmBase Behavior Model

The following behavior model shows the relationship between the IviDmmBase capability group and DMM behavior.

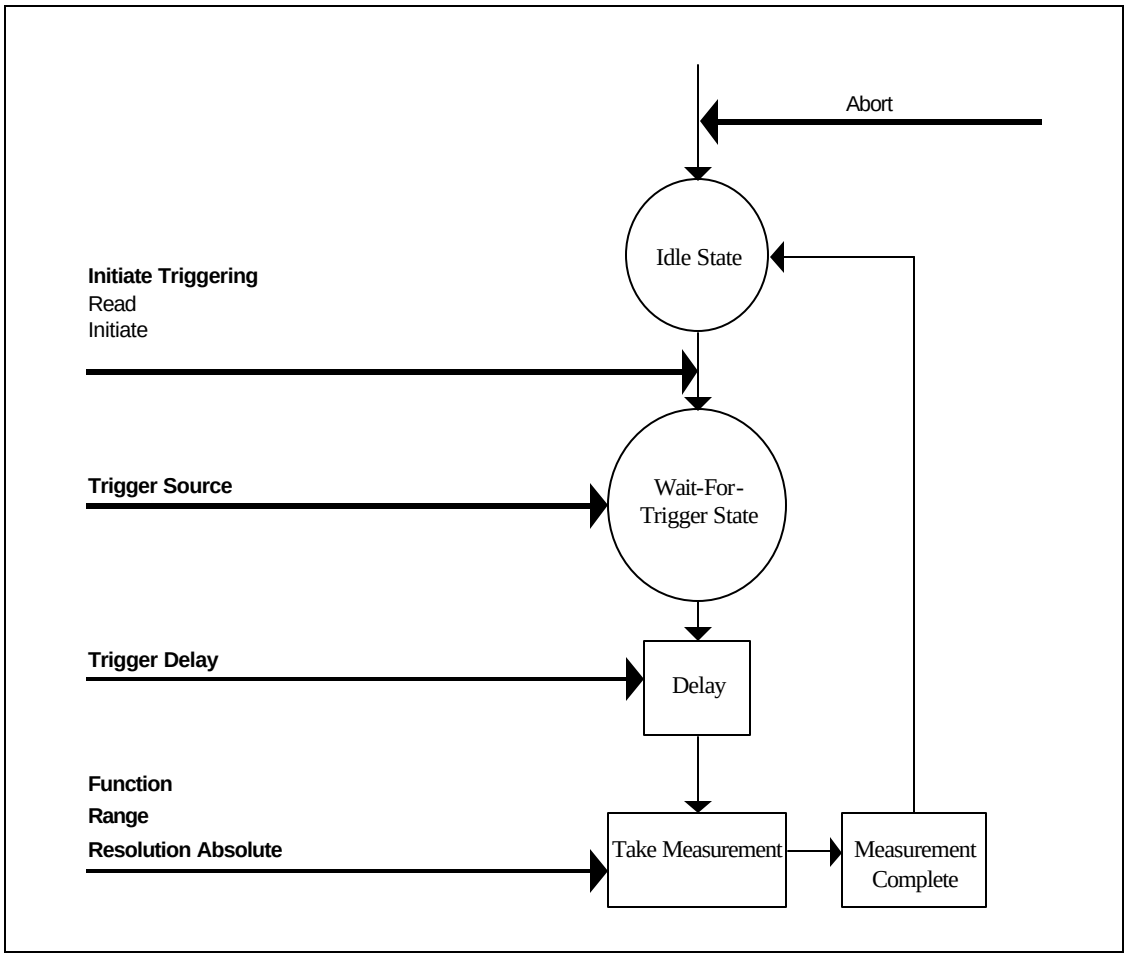


Figure 4-1. IviDmm Behavior Model

The main state in the IviDmm Class is the Idle state. The DMM enters the Idle state as the result of being "powered-on", successfully completing a measurement, or by being aborted from a previous measurement by the user with the Abort function. Typically, the user configures the DMM while it is in the Idle state. IviDmm attributes can be configured individually with the Set Attribute function or with the high-level Configure function.

The Read and Initiate functions cause the DMM to leave the Idle state and transition to the Wait-For-Trigger state. The Read function does not return until the measurement process is complete and the DMM returns to the Idle state. The Initiate function returns as soon as the DMM leaves the Idle state.

The DMM leaves the Wait-For-Trigger state when it receives a trigger event. The type of trigger event is specified by the attribute Trigger Source.

After the specified trigger event occurs, the DMM waits the amount of time specified by the attribute Trigger Delay and then takes a measurement. The type of measurement is specified by the attributes Function, Range, and Resolution Absolute.

If the Function attribute is set to a value that requires an extension capability group, the attributes of that capability group further configure the measurement.

After the measurement is taken, the DMM (if it is capable of doing so) generates the Measurement Complete signal and returns to the Idle state.

The IviDmmBase capability group does not require that a DMM be able to generate a Measurement Complete signal. The IviDmmMultiPoint capability group defines how the Measurement Complete signal is configured. The Measurement Complete signal is presented in the IviDmm behavior model diagram to define when the signal is generated as most DMMs generate this signal but may not be able to configure it.

The Fetch function is used to retrieve measurements that were initiated by the Initiate function. The measurement data returned from the Read and Fetch functions is acquired after the DMM has left the Wait-For-Trigger state.

5. IviDmmACMeasurement Extension Group

5.1 IviDmmACMeasurement Overview

The IviDmmACMeasurement extension group supports DMMs that take AC voltage or AC current measurements. It defines attributes that configure additional settings for AC measurements. These attributes are the minimum and maximum frequency components of the input signal. This extension group also defines functions that configure these attributes.

5.2 IviDmmACMeasurement Attributes

The IviDmmACMeasurement extension group defines the following attributes:

? AC Max Freq

? AC Min Freq

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions*.

5.2.1 AC Max Freq

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	Up	Configure AC Bandwidth

COM Property Name

AC.FrequencyMax

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_AC_MAX_FREQ

Description

Specifies the maximum frequency component of the input signal for AC measurements. The value of this attribute affects instrument behavior only when the Function attribute is set to an AC voltage or AC current measurement.

5.2.2 AC Min Freq

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	Down	Configure AC Bandwidth

COM Property Name

AC.FrequencyMin

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_AC_MIN_FREQ

Description

Specifies the minimum frequency component of the input signal for AC measurements. The value of this attribute affects instrument behavior only when the Function attribute is set to an AC voltage or AC current measurement.

5.3 IviDmmACMeasurement Functions

The IviDmmACMeasurement extension group defines the following function:

? Configure AC Bandwidth

This section describes the behavior and requirements of this function.

5.3.1 Configure AC Bandwidth

Description

This function configures additional parameters for DMMs that take AC voltage or AC current measurements. These attributes are the AC minimum and maximum frequency.

COM Method Prototype

```
HRESULT AC.ConfigureBandwidth ([in] DOUBLE MinFreq,
                               [in] DOUBLE MaxFreq);
```

C Prototype

```
ViStatus IviDmm_ConfigureACBandwidth (ViSession Vi,
                                      ViReal64 MinFreq,
                                      ViReal64 MaxFreq);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`MinFreq`	Specifies the AC minimum frequency. The driver uses this value to set the AC Min Freq attribute. See the attribute description for more details.	`ViReal64`
`MaxFreq`	Specifies the AC maximum frequency. The driver uses this value to set the AC Max Freq attribute. See the attribute description for more details.	`ViReal64`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

5.4 IviDmmACMeasurement Behavior Model

The IviDmmACMeasurement extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

5.5 IviDmmACMeasurement Compliance Notes

1. IviDmm specific drivers that implement this extension group shall implement at least one of the following values for the `Function` attribute in the IviDmmBase capability group:
   - ? AC Volts
   - ? AC Current
   - ? AC Plus DC Volts
   - ? AC Plus DC Volts

6. IviDmmFrequencyMeasurement Extension Group

6.1 IviDmmFrequencyMeasurement Overview

The IviDmmFrequencyMeasurement extension group supports DMMs that take frequency measurements. It defines attributes that are required to configure additional parameters needed for frequency measurements.

6.2 IviDmmFrequencyMeasurement Attributes

The IviDmmFrequencyMeasurement extension group defines the following attribute:

? Frequency Voltage Range

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

6.2.1 Frequency Voltage Range

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	Up	Configure Frequency Voltage Range

COM Property Name

Frequency.VoltageRange

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_FREQ_VOLTAGE_RANGE

Description

Specifies the expected maximum value of the input signal for frequency and period measurements. Positive values represent the manual range. Negative values represent the Auto Range mode.

The value of this attribute affects instrument behavior only when the Function attribute is set to a frequency or period measurement.

The units are specified in Volts RMS.

Defined Values

Name	*Description*		
		Language	*Identifier*
Auto Range On	Sets the DMM to calculate the frequency voltage range before each measurement automatically. Setting this attribute to a manual range or to Auto Range Off disables auto-ranging.		
		C	IVIDMM_VAL_AUTO_RANGE_ON
		COM	IviDmmFrequencyVoltageRangeAutoRangeOn
Auto Range Off	Disables auto-ranging. Further queries of this attribute return the actual frequency voltage range.		
		C	IVIDMM_VAL_AUTO_RANGE_OFF
		COM	IviDmmFrequencyVoltageRangeAutoRangeOff

Compliance Notes

1. If an IVI-C specific driver defines additional values for this attribute, the magnitude of the actual values must be greater than or equal to Frequency Volt Range Specific Extension Base.
2. If an IVI-C class driver defines additional values for this attribute, the magnitude of the actual values must be greater than or equal to Frequency Volt Range Class Extension Base and less than Frequency Volt Range Specific Extension Base.

3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Frequency Volt Range Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Frequency Volt Range Specific Extension Base and Frequency Volt Range Class Extension Base.

6.3 IviDmmFrequencyMeasurement Functions

The IviDmmFrequencyMeasurement extension group defines the following function:

? Configure Frequency Voltage Range (IVI-C only)

This section describes the behavior and requirements of this function.

6.3.1 Configure Frequency Voltage Range (IVI-C Only)

Description

This function configures the frequency voltage range of the DMM.

COM Method Prototype

```
N/A
(use the Frequency.VoltageRange property)
```

C Prototype

```
ViStatus IviDmm_ConfigureFrequencyVoltageRange (ViSession Vi,
                                                ViReal64 FrequencyVoltageRange);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`FrequencyVoltageRange`	Specifies the expected maximum value of the input signal. The driver uses this value to set the Frequency Voltage Range attribute. See the attribute description for more details.	`ViReal64`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

6.4 IviDmmFrequencyMeasurement Behavior Mode

The IviDmmFrequencyMeasurement extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

6.5 IviDmmFrequencyMeasurement Compliance Notes

1. IviDmm specific drivers that implement this extension group shall implement at least one of the following values for the Function attribute in the IviDmmBase capability group:

   ? Frequency

   ? Period

7. IviDmmTemperatureMeasurement Extension Group

7.1 IviDmmTemperatureMeasurement Overview

The IviDmmTemperatureMeasurement extension group supports DMMs that take temperature measurements with a thermocouple, an RTD, or a thermistor transducer type. This extension group selects the transducer type. Other capability groups further configure temperature settings based on the transducer type.

7.2 IviDmmTemperatureMeasurement Attributes

The IviDmmTemperatureMeasurement extension group defines the following attribute:

? Temperature Transducer Type

This section describes the behavior and requirements of this attribute. The actual value for the attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

7.2.1 Temperature Transducer Type

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Transducer Type

COM Property Name

Temperature.TransducerType

COM Enumeration Name

IviDmmTransducerTypeEnum

C Constant Name

IVIDMM_ATTR_TEMP_TRANSDUCER_TYPE

Description

Specifies the device used to measure the temperature. The value of this attribute affects instrument behavior only when the Function attribute is set to a temperature measurement.

Defined Values

Name	*Description*		
		Language	*Identifier*
Thermocouple	Sets the DMM to measure temperature using a thermocouple. Use the IviDmmThermocouple extension group to configure additional settings for this transducer type.		
		C	IVIDMM_VAL_THERMOCOUPLE
		COM	IviDmmTransducerTypeThermocouple
Thermistor	Sets the DMM to measure temperature using a thermistor. Use the IviDmmThermistor extension group to configure additional settings for this transducer type.		
		C	IVIDMM_VAL_THERMISTOR
		COM	IviDmmTransducerTypeThermistor
2 Wire RTD	Sets the DMM to measure temperature using a 2-wire temperature resistance device. Use the IviDmmResistanceTemperatureDevice extension group to configure additional settings for this transducer type.		
		C	IVIDMM_VAL_2_WIRE_RTD
		COM	IviDmmTransducerType2WireRtd
4 Wire RTD	Sets the DMM to measure temperature using a 4-wire temperature resistance device. Use the IviDmmResistanceTemperatureDevice extension group to configure additional settings for this transducer type.		
		C	IVIDMM_VAL_4_WIRE_RTD
		COM	IviDmmTransducerType4WireRtd

Compliance Notes

If an IviDmm specific driver implements any of the defined values in the following table, it shall also implement the corresponding capability group:

Value	Required Capability Group
Thermocouple	IviDmmThermocouple
Thermistor	IviDmmThermistor
2 Wire RTD	IviDmmResistanceTemperatureDevice
4 Wire RTD	IviDmmResistanceTemperatureDevice

1. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to Transducer Specific Extension Base.
2. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to Transducer Class Extension Base and less than Transducer Specific Extension Base.
3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Transducer Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Transducer Specific Extension Base and Transducer Class Extension Base.

7.3 IviDmmTemperatureMeasurement Functions

The IviDmmTemperatureMeasurement extension group defines the following function:

? Configure Transducer Type (IVI-C only)

This section describes the behavior and requirements of this function.

7.3.1 Configure Transducer Type (IVI-C Only)

Description

This function configures the DMM to take temperature measurements from a specified transducer type.

COM Method Prototype

```
N/A
(use the Temperature.TransducerType property)
```

C Prototype

```
ViStatus IviDmm_ConfigureTransducerType (ViSession Vi,
                                         ViInt32 TransducerType);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
TransducerType	Specifies the device used to measure the temperature. The driver uses this value to set the Temperature Transducer Type attribute. See the attribute description for more details	ViInt32

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

7.4 IviDmmTemperatureMeasurement Behavior Model

The IviDmmTemperatureMeasurement extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

7.5 IviDmmTemperatureMeasurement Compliance Notes

1. IviDmm specific drivers that implement this extension group shall implement the Temperature value for the Function attribute in the IviDmmBase capability group.

8. IviDmmThermocouple Extension Group

8.1 IviDmmThermocouple Extension Group Overview

The IviDmmThermocouple extension group supports DMMs that take temperature measurements using a thermocouple transducer type.

8.2 IviDmmThermocouple Attributes

The IviDmmThermocouple extension group defines the following attributes:

? Thermocouple Fixed Reference Junction

? Thermocouple Reference Junction Type

? Thermocouple Type

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions*.

8.2.1 Thermocouple Fixed Reference Junction

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	None	Configure Fixed Reference Junction

COM Property Name

Temperature.Thermocouple.FixedRefJunction

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_TEMP_TC_FIXED_REF_JUNC

Description

Specifies the external reference junction temperature when a fixed reference junction type thermocouple is used to take the temperature measurement. The temperature is specified in degrees Celsius.

This attribute may also be used to specify the thermocouple junction temperature of an instrument that does not have an internal temperature sensor.

The value of this attribute affects instrument behavior only when the Thermocouple Reference Junction Type is set to Temperature Reference Junction Fixed.

Compliance Notes

IviDmm specific drivers that implement this attribute shall implement the Temperature Reference Junction Fixed defined value for the Thermocouple Reference Junction Type attribute.

8.2.2 Thermocouple Reference Junction Type

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Thermocouple

COM Property Name

`Temperature.Thermocouple.RefJunctionType`

COM Enumeration Name

`IviDmmRefJunctionTypeEnum`

C Constant Name

`IVIDMM_ATTR_TEMP_TC_REF_JUNC_TYPE`

Description

Specifies the type of reference junction to be used in the reference junction compensation of a thermocouple measurement. The value of this attribute affects instrument behavior only when the Temperature Transducer Type is set to Thermocouple.

Defined Values

Name	*Description*		
		Language	*Identifier*
Thermocouple Reference Junction Internal	Sets the DMM to use an internal sensor at the thermocouple junction for the junction compensation.		
		C	`IVIDMM_VAL_TEMP_REF_JUNC_INTERNAL`
		COM	`IviDmmRefJunctionTypeInternal`
Thermocouple Reference Junction Fixed	Sets the DMM to use a fixed value for the thermocouple junction compensation. Use the Thermocouple Fixed Reference Junction attribute to set the fixed reference junction value.		
		C	`IVIDMM_VAL_TEMP_REF_JUNC_FIXED`
		COM	`IviDmmRefJunctionTypeFixed`

Compliance Notes

If an IviDmm specific driver implements the Temperature Reference Junction Fixed defined value, then it shall implement the Thermocouple Fixed Reference Junction attribute.

1. If an IVI-C IviDmm specific driver defines additional values for this attribute, the actual values shall be greater than or equal to Reference Junction Specific Extension Base.
2. If an IVI-C IviDmm class driver defines additional values for this attribute, the actual values shall be greater than or equal to Reference Junction Class Extension Base and less than Reference Junction Specific Extension Base.
3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Reference Junction Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Reference Junction Specific Extension Base and Reference Junction Class Extension Base.

8.2.3 Thermocouple Type

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Thermocouple

COM Property Name

Temperature.Thermocouple.Type

COM Enumeration Name

IviDmmThermocoupleTypeEnum

C Constant Name

IVIDMM_ATTR_TEMP_TC_TYPE

Description

Specifies the type of thermocouple used to measure the temperature. The value of this attribute affects instrument behavior only when the Temperature Transducer Type is set to Thermocouple.

Defined Values

Name	*Description*		
		Language	*Identifier*
Thermocouple B	Sets the DMM to measure temperature from a B-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_B
		COM	IviDmmThermocoupleTypeB
Thermocouple C	Sets the DM7M to measure temperature from a C-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_C
		COM	IviDmmThermocoupleTypeC
Thermocouple D	Sets the DMM to measure temperature from a D-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_D
		COM	IviDmmThermocoupleTypeD
Thermocouple E	Sets the DMM to measure temperature from an E-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_E
		COM	IviDmmThermocoupleTypeE
Thermocouple G	Sets the DMM to measure temperature from a G-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_G
		COM	IviDmmThermocoupleTypeG

Name	*Description*		
		Language	*Identifier*
Thermocouple J	Sets the DMM to measure temperature from a J-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_J
		COM	IviDmmThermocoupleTypeJ
Thermocouple K	Sets the DMM to measure temperature from a K-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_K
		COM	IviDmmThermocoupleTypeK
Thermocouple N	Sets the DMM to measure temperature from an N-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_N
		COM	IviDmmThermocoupleTypeN
Thermocouple R	Sets the DMM to measure temperature from an R-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_R
		COM	IviDmmThermocoupleTypeR
Thermocouple S	Sets the DMM to measure temperature from an S-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_S
		COM	IviDmmThermocoupleTypeS
Thermocouple T	Sets the DMM to measure temperature from a T-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_T
		COM	IviDmmThermocoupleTypeT
Thermocouple U	Sets the DMM to measure temperature from a U-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_U
		COM	IviDmmThermocoupleTypeU
Thermocouple V	Sets the DMM to measure temperature from a V-type thermocouple.		
		C	IVIDMM_VAL_TEMP_TC_V
		COM	IviDmmThermocoupleTypeV

Compliance Notes

1. If an IVI-C IviDmm specific driver defines additional values for this attribute, the actual values must be greater than or equal to Thermocouple Type Specific Extension Base.
2. If an IVI-C IviDmm class driver defines additional values for this attribute, the actual values must be greater than or equal to Thermocouple Type Class Extension Base and less than Thermocouple Type Specific Extension Base.

3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Thermocouple Type Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Thermocouple Type Specific Extension Base and Thermocouple Type Class Extension Base.

8.3 IviDmmThermocouple Functions

The IviDmmTemperatureMeasurement extension group defines the following functions:

- Configure Fixed Reference Junction (IVI-C only)
- Configure Thermocouple

This section describes the behavior and requirements of each function.

8.3.1 Configure Fixed Reference Junction (IVI-C Only)

Description

This function configures the fixed reference junction for a thermocouple with a fixed reference junction type.

COM Method Prototype

```
N/A
(use the Temperature.Thermocouple.FixedRefJunction property)
```

C Prototype

```
ViStatus IviDmm_ConfigureFixedRefJunction (ViSession Vi,
                                           ViReal64 FixedRefJunction);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`FixedRefJunction`	Specifies the fixed reference junction. The driver uses this value to set the Thermocouple Fixed Reference Junction attribute. See the attribute description for more details.	`ViReal64`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

Compliance Notes

The IviDmm specific driver is required to implement this function only if the Thermocouple Reference Junction Type attribute implements the Temperature Reference Junction Fixed defined value.

8.3.2 Configure Thermocouple

Description

This function configures the thermocouple type and the reference junction type of the thermocouple for DMMs that take temperature measurements using a thermocouple transducer type.

COM Method Prototype

```
HRESULT Temperature.Thermocouple.Configure(
                    [in] IviDmmThermocoupleTypeEnum Type,
                    [in] IviDmmRefJunctionTypeEnum RefJunctionType);
```

C Prototype

```
ViStatus IviDmm_ConfigureThermocouple (ViSession Vi,
                                       ViInt32 ThermocoupleType
                                       ViInt32 RefJunctionType);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`ThermocoupleType`	Specifies the type of thermocouple used to measure the temperature. The driver uses this value to set the Thermocouple Type attribute. See the attribute description for more details.	`ViInt32`
`RefJunctionType`	Specifies the type of reference junction to be used. The driver uses this value to set the Thermocouple Reference Junction Type attribute. See the attribute description for more details.	`ViInt32`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

8.4 IviDmmThermocouple Behavior Model

The IviDmmThermocouple extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

8.5 IviDmmThermocouple Compliance Notes

1. IviDmm specific drivers that implement this extension group shall implement the IviDmmTemperatureMeasurement extension group.
2. IviDmm specific drivers that implement this extension group shall implement the Thermocouple value for the Temperature Transducer Type attribute in the IviDmmTemperatureMeasurement extension group.

9. IviDmmResistanceTemperatureDevice Extension Group

9.1 IviDmmResistanceTemperatureDevice Extension Group Overview

The IviDmmResistanceTemperatureDevice extension group supports DMMs that take temperature measurements using a resistance temperature device (RTD) transducer type.

The IviDmm class assumes that you are using a Platinum Resistance Temperature Device.

9.2 IviDmmResistanceTemperatureDevice Attributes

The IviDmmResistanceTemperatureDevice extension group defines the following attributes:

? RTD Alpha

? RTD Resistance

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

9.2.1 RTD Alpha

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	None	Configure RTD

COM Property Name

Temperature.RTD.Alpha

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_TEMP_RTD_ALPHA

Description

Specifies the alpha parameter for a resistance temperature device (RTD).

The value of this attribute affects instrument behavior only when the Temperature Transducer Type is set to the 2 Wire RTD or the 4 Wire RTD defined values.

9.2.2 RTD Resistance

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	None	Configure RTD

COM Property Name

Temperature.RTD.Resistance

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_TEMP_RTD_RES

Description

Specifies the R_0 parameter (resistance) for a resistance temperature device (RTD). The RTD resistance is also known as the RTD reference value.

The value of this attribute affects instrument behavior only when the Temperature Transducer Type is set to the RTD defined value.

9.3 IviDmmResistanceTemperatureDevice Functions

The IviDmmResistanceTemperatureDevice extension group defines the following function:

? Configure RTD

This section describes the behavior and requirements of this function.

9.3.1 Configure RTD

Description

This function configures the alpha and resistance parameters for a resistance temperature device.

COM Method Prototype

```
HRESULT Temperature.RTD.Configure([in] DOUBLE Alpha,
                                  [in] DOUBLE Resistance);
```

C Prototype

```
ViStatus IviDmm_ConfigureRTD (ViSession Vi,
                              ViReal64 Alpha
                              ViReal64 Resistance);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Alpha	Specifies the alpha parameter for an RTD. The driver use this value to set the Temperature RTD Alpha attribute. See the attribute description for more details.	ViReal64
Resistance	Specifies the resistance of the RTD. The driver uses this value to set the Temperature RTD Resistance attribute. See the attribute description for more details.	ViReal64

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

9.4 IviDmmResistanceTemperatureDevice Behavior Model

The IviDmmResistanceTemperatureDevice extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

9.5 IviDmmResistanceTemperatureDevice Compliance Notes

1. IviDmm specific drivers that implement this extension group shall implement the IviDmmTemperatureMeasurement extension group.
2. IviDmm specific drivers that implement this extension group shall implement at least one of the following values for the Temperature Transducer Type attribute in the IviDmmTemperatureMeasurement extension group:
   - 2 Wire RTD
   - 4 Wire RTD

10. IviDmmThermistor Extension Group

10.1 IviDmmThermistor Extension Group Overview

The IviDmmThermistor extension group supports DMMs that take temperature measurements using a thermistor transducer type.

The IviDmm class assumes that you are using an interchangeable thermistor. Interchangeable thermistors are thermistors that exhibit similar behavior for a given resistance value.

10.2 IviDmmThermistor Attributes

The IviDmmThermistor extension group defines the following attribute:

? Thermistor Resistance

This section describes the behavior and requirements of this attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

10.2.1 Thermistor Resistance

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	None	Configure Thermistor

COM Property Name

Temperature.Thermistor.Resistance

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_TEMP_THERMISTOR_RES

Description

Specifies the resistance of the thermistor in Ohms.

The value of this attribute affects instrument behavior only when the Temperature Transducer Type attribute is set to the Thermistor defined value.

10.3 IviDmmThermistor Functions

The IviDmmThermistor extension group defines the following function:

? Configure Thermistor (IVI-C only)

This section describes the behavior and requirements of this function.

10.3.1 Configure Thermistor (IVI-C Only)

Description

This function configures the resistance for a thermistor temperature measurement device.

COM Method Prototype

```
N/A
(use the Temperature.Thermistor property)
```

C Prototype

```
ViStatus IviDmm_ConfigureThermistor (ViSession Vi,
                                     ViReal64 Resistance);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Resistance`	Specifies the resistance of the thermistor. The driver uses this value to set the Temperature Thermistor Resistance attribute. See the attribute description for more details.	`ViReal64`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

10.4 IviDmmThermistor Behavior Model

The IviDmmThermistor extension group follows the same behavior model as IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

10.5 IviDmmThermistor Compliance Notes

1. IviDmm specific drivers that implement this extension group shall implement the IviDmmTemperatureMeasurement extension group.
2. IviDmm specific drivers that implement this extension group shall implement the Thermistor value for the Temperature Transducer Type attribute in the IviDmmTemperatureMeasurement capability group.

11. IviDmmMultiPoint Extension Group

11.1 IviDmmMultiPoint Extension Group Overview

The IviDmmMultiPoint extension group defines extensions for DMMs capable of acquiring measurements based on multiple triggers, and acquiring multiple measurements for each trigger.

The IviDmmMultiPoint extension group defines additional attributes such sample count, sample trigger, trigger count, and trigger delay to control “multi-point” DMMs. The IviDmmMultiPoint extension group also adds functions for configuring the DMM as well as starting acquisitions and retrieving multiple measured values.

11.2 IviDmmMultiPoint Attributes

The IviDmmBase capability group defines the following attributes:

? Measure Complete Destination

? Sample Count

? Sample Interval

? Sample Trigger

? Trigger Count

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions*.

11.2.1 Measure Complete Destination

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Measure Complete Dest

COM Property Name

Trigger.MultiPoint.MeasurementComplete

COM Enumeration Name

IviDmmMeasCompleteDestEnum

C Constant Name

IVIDMM_ATTR_MEAS_COMPLETE_DEST

Description

After each measurement, the DMM generates a measurement-complete signal. This attribute specifies the destination of the measurement-complete signal. This signal is commonly referred to as Voltmeter Complete.

Defined Values

Name	*Description*		
		Language	*Identifier*
None	The measurement complete signal is not routed.		
		C	IVIDMM_VAL_NONE
		COM	IviDmmMeasCompleteDestNone
External	Routes the measurement complete signal to the external connector.		
		C	IVIDMM_VAL_EXTERNAL
		COM	IviDmmMeasCompleteDestExternal
TTL0	Routes the measurement complete signal to TTL0.		
		C	IVIDMM_VAL_TTL0
		COM	IviDmmMeasCompleteDestTTL0
TTL1	Routes the measurement complete signal to TTL1.		
		C	IVIDMM_VAL_TTL1
		COM	IviDmmMeasCompleteDestTTL1
TTL2	Routes the measurement complete signal to TTL2.		
		C	IVIDMM_VAL_TTL2
		COM	IviDmmMeasCompleteDestTTL2
TTL3	Routes the measurement complete signal to TTL3.		
		C	IVIDMM_VAL_TTL3
		COM	IviDmmMeasCompleteDestTTL3

Name	*Description*		
		Language	*Identifier*
TTL4	Routes the measurement complete signal to TTL4.		
		C	IVIDMM_VAL_TTL4
		COM	IviDmmMeasCompleteDestTTL4
TTL5	Routes the measurement complete signal to TTL5.		
		C	IVIDMM_VAL_TTL5
		COM	IviDmmMeasCompleteDestTTL5
TTL6	Routes the measurement complete signal to TTL6.		
		C	IVIDMM_VAL_TTL6
		COM	IviDmmMeasCompleteDestTTL6
TTL7	Routes the measurement complete signal to TTL7.		
		C	IVIDMM_VAL_TTL7
		COM	IviDmmMeasCompleteDestTTL7
ECL0	Routes the measurement complete signal to ECL0.		
		C	IVIDMM_VAL_ECL0
		COM	IviDmmMeasCompleteDestECL0
ECL1	Routes the measurement complete signal to ECL1.		
		C	IVIDMM_VAL_ECL1
		COM	IviDmmMeasCompleteDestECL1
PXI Star	Routes the measurement complete signal to the PXI Star trigger bus.		
		C	IVIDMM_VAL_PXI_STAR
		COM	IviDmmMeasCompleteDestPXIStar
RTSI 0	Routes the measurement complete signal to RTSI0.		
		C	IVIDMM_VAL_RTSI_0
		COM	IviDmmMeasCompleteDestRTSI0
RTSI 1	Routes the measurement complete signal to RTSI1.		
		C	IVIDMM_VAL_RTSI_1
		COM	IviDmmMeasCompleteDestRTSI1
RTSI 2	Routes the measurement complete signal to RTSI2.		
		C	IVIDMM_VAL_RTSI_2
		COM	IviDmmMeasCompleteDestRTSI2
RTSI 3	Routes the measurement complete signal to RTSI3.		
		C	IVIDMM_VAL_RTSI_3
		COM	IviDmmMeasCompleteDestRTSI3
RTSI 4	Routes the measurement complete signal to RTSI4.		
		C	IVIDMM_VAL_RTSI_4

Name	*Description*		
		Language	*Identifier*
		COM	`IviDmmMeasCompleteDestRTSI4`
RTSI 5	Routes the measurement complete signal to RTSI5.		
		C	`IVIDMM_VAL_RTSI_5`
		COM	`IviDmmMeasCompleteDestRTSI5`
RTSI 6	Routes the measurement complete signal to RTSI6.		
		C	`IVIDMM_VAL_RTSI_6`
		COM	`IviDmmMeasCompleteDestRTSI6`

Compliance Notes

1. If an IVI-C IviDmm specific driver defines additional values for this attribute, the actual values shall be greater than or equal to Trigger Source Specific Extension Base.
2. If an IVI-C IviDmm class driver defines additional values for this attribute, the actual values shall be greater than or equal to Trigger Source Class Extension Base and less than Trigger Source Specific Extension Base.
3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Trigger Source Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Trigger Source Specific Extension Base and Trigger Source Class Extension Base.

11.2.2 Sample Count

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Multi Point

COM Property Name

Trigger.MultiPoint.SampleCount

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_SAMPLE_COUNT

Description

Specifies the number of measurements the DMM takes each time it receives a trigger.

11.2.3 Sample Interval

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	Up	Configure Multi Point

COM Property Name

Trigger.MultiPoint.SampleInterval

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_SAMPLE_INTERVAL

Description

Specifies the interval between samples in seconds. This attribute affects instrument behavior only when Sample Count is greater than 1 and Sample Trigger is Interval.

Compliance Notes

1. IviDmm specific drivers that implement this attribute must implement the Interval value for the Sample Trigger attribute.

11.2.4 Sample Trigger

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Multi Point

COM Property Name

`Trigger.MultiPoint.SampleTrigger`

COM Enumeration Name

`IviDmmSampleTriggerEnum`

C Constant Name

`IVIDMM_ATTR_SAMPLE_TRIGGER`

Description

Specifies the sample trigger source. If the value of the Sample Count is greater than 1, the DMM enters the Wait-For-Sample-Trigger state after taking a single measurement. When the event specified by this attribute occurs, the DMM exits the Wait-For-Sample-Trigger state and takes the next measurement.

This attribute affects instrument behavior only when Sample Count is greater than 1.

Defined Values

Name	*Description*		
		Language	*Identifier*
Immediate	The DMM exits the Wait-For-Sample-Trigger state immediately after entering. It does not wait for a trigger of any kind.		
		C	`IVIDMM_VAL_IMMEDIATE`
		COM	`IviDmmSampleTriggerImmediate`
External	The DMM exits the Wait-For-Sample-Trigger state when a trigger occurs on the external trigger input.		
		C	`IVIDMM_VAL_EXTERNAL`
		COM	`IviDmmSampleTriggerExternal`
Software Trigger	The DMM exits the Wait-For-Sample-Trigger state when the Send Software Trigger function executes. Refer to the Standardized Cross Class Capabilities specification for a complete description of this value and the Send Software Trigger function.		
		C	`IVIDMM_VAL_SOFTWARE_TRIG`
		COM	`IviDmmSampleTriggerSwTrigFunc`

Name	*Description*		
		Language	*Identifier*
Interval	The DMM exits the Wait-For-Sample-Trigger state when the length of time specified by the Sample Interval attribute elapses.		
		C	IVIDMM_VAL_INTERVAL
		COM	IviDmmSampleTriggerInterval
TTL0	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on TTL0.		
		C	IVIDMM_VAL_TTL0
		COM	IviDmmSampleTriggerTTL0
TTL1	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on TTL1.		
		C	IVIDMM_VAL_TTL1
		COM	IviDmmSampleTriggerTTL1
TTL2	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on TTL2.		
		C	IVIDMM_VAL_TTL2
		COM	IviDmmSampleTriggerTTL2
TTL3	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on TTL3.		
		C	IVIDMM_VAL_TTL3
		COM	IviDmmSampleTriggerTTL3
TTL4	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on TTL4.		
		C	IVIDMM_VAL_TTL4
		COM	IviDmmSampleTriggerTTL4
TTL5	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on TTL5.		
		C	IVIDMM_VAL_TTL5
		COM	IviDmmSampleTriggerTTL5
TTL6	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on TTL6.		
		C	IVIDMM_VAL_TTL6
		COM	IviDmmSampleTriggerTTL6
TTL7	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on TTL7.		
		C	IVIDMM_VAL_TTL7
		COM	IviDmmSampleTriggerTTL7

Name	*Description*		
		Language	*Identifier*
ECL0	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on ECL0.		
		C	`IVIDMM_VAL_ECL0`
		COM	`IviDmmSampleTriggerECL0`
ECL1	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on ECL1.		
		C	`IVIDMM_VAL_ECL1`
		COM	`IviDmmSampleTriggerECL1`
PXI Star	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on the PXI Star trigger bus.		
		C	`IVIDMM_VAL_PXI_STAR`
		COM	`IviDmmSampleTriggerPXIStar`
RTSI 0	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on RTSI0.		
		C	`IVIDMM_VAL_RTSI_0`
		COM	`IviDmmSampleTriggerRTSI0`
RTSI 1	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on RTSI1.		
		C	`IVIDMM_VAL_RTSI_1`
		COM	`IviDmmSampleTriggerRTSI1`
RTSI 2	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on RTSI2.		
		C	`IVIDMM_VAL_RTSI_2`
		COM	`IviDmmSampleTriggerRTSI2`
RTSI 3	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on RTSI3.		
		C	`IVIDMM_VAL_RTSI_3`
		COM	`IviDmmSampleTriggerRTSI3`
RTSI 4	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on RTSI4.		
		C	`IVIDMM_VAL_RTSI_4`
		COM	`IviDmmSampleTriggerRTSI4`
RTSI 5	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on RTSI5.		
		C	`IVIDMM_VAL_RTSI_5`
		COM	`IviDmmSampleTriggerRTSI5`
RTSI 6	The DMM exits the Wait-For-Sample-Trigger state when it receives a trigger on RTSI6.		
		C	`IVIDMM_VAL_RTSI_6`
		COM	`IviDmmSampleTriggerRTSI6`

Compliance Notes

1. If an IviDmm specific driver implements any of the defined values in the following table, it must also implement the corresponding capability group:

Value	Required Capability Group
Software Trigger	IviDmmSoftwareTrigger

2. If the IviDmm specific driver implements the Interval defined value, then it must also implement the Sample Interval attribute.
3. If an IVI-C IviDmm specific driver defines additional values for this attribute, the actual values must be greater than or equal to Trigger Source Specific Extension Base.
4. If an IVI-C IviDmm class driver defines additional values for this attribute, the actual values must be greater than or equal to Trigger Source Class Extension Base and less than Trigger Source Specific Extension Base.
5. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Trigger Source Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Trigger Source Specific Extension Base and Trigger Source Class Extension Base.

11.2.5 Trigger Count

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Multi Point

COM Property Name

Trigger.MultiPoint.Count

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_TRIGGER_COUNT

Description

Specifies the number of triggers the DMM accepts before it returns to the idle state.

11.3 IviDmmMultiPoint Functions

The IviDmmMultiPoint extension group defines the following functions:

? Configure Measure Complete Destination (IVI-C only)

? Configure Multi Point

? Fetch Multi Point

? Read Multi Point

This section describes the behavior and requirements of each function.

11.3.1 Configure Measure Complete Destination

Description

This function configures the destination of the measurement-complete signal. This signal is commonly referred to as Voltmeter Complete.

COM Method Prototype

```
N/A
(use the Trigger.MultiPoint.MeasurementComplete property)
```

C Prototype

```
ViStatus IviDmm_ConfigureMeasCompleteDest (ViSession Vi,
                                           ViInt32 MeasCompleteDest);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`MeasCompleteDest`	Specifies the measurement complete destination. The driver uses this value to set the Measure Complete Destination attribute. See the attribute description for more details.	`ViInt32`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

11.3.2 Configure Multi Point

Description

This function configures the attributes for multi-point measurements. These attributes include the trigger count, sample count, sample trigger and sample interval.

If the value of the `SampleCount` parameter is 1, then the `SampleTrigger` and `SampleInterval` parameters are ignored and are not used to set the Sample Trigger and Sample Interval attributes.

If the value of the `SampleTrigger` parameter is not Interval, then the `SampleInterval` parameter is ignored and is not used to set the Sample Interval attribute.

COM Method Prototype

```
HRESULT Trigger.MultiPoint.Configure(
                            [in] LONG TriggerCount,
                            [in] LONG SampleCount,
                            [in] IviDmmSampleTriggerEnum SampleTrigger,
                            [in] DOUBLE SampleInterval);
```

C Prototype

```
ViStatus IviDmm_ConfigureMultiPoint (ViSession Vi,
                                     ViInt32 TriggerCount,
                                     ViInt32 SampleCount,
                                     ViInt32 SampleTrigger,
                                     ViReal64 SampleInterval);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`TriggerCount`	Specifies the trigger count. The driver uses this value to set the Trigger Count attribute. See the attribute description for more details.	`ViInt32`
`SampleCount`	Specifies the sample count. The driver uses this value to set the Sample Count attribute. See the attribute description for more details	`ViInt32`
`SampleTrigger`	Specifies the sample trigger source. The driver uses this value to set the Sample Trigger attribute. See the attribute description for more details	`ViInt32`
`SampleInterval`	Specifies the sample interval. The driver uses this value to set the Sample Interval attribute. See the attribute description for more details	`ViReal64`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

11.3.3 Fetch Multi Point

Description

This function returns an array of values from a measurement that the Initiate function initiates. After this function executes, each element in the `ReadingArray` parameter is an actual reading or a value indicating that an overrange condition occurred. The number of measurements the DMM takes is determined by the Trigger Count and Sample Count.

If an overrange condition occurs, the corresponding `ReadingArray` element contains an IEEE defined NaN (Not a Number) value and the function returns the Over Range completion code. The user may test each element in the `ReadingArray` parameter for overrange with the Is Over Range function.

In most instrument classes, there is a programmatic way to determine when a measurement has completed and data is available. Therefore, a `MaxTimeMilliseconds` parameter is not needed in the Fetch function for these classes. This is not true for the majority of DMMs. The `MaxTimeMilliseconds` parameter specifies how long to wait in the Fetch Multi Point operation since it is possible that no data is available or the trigger event did not occur.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. To check the instrument status, call the Error Query function at the conclusion of the sequence.

The value of the Function attribute determines the units for each element of the `ReadingArray` parameter. The following table shows the defined values for the Function attribute and the corresponding units for each element of the `ReadingArray` parameter.

Values for FUNCTION ATTRIBUTE	Units for readingArray parameter
DC Volts	Volts
AC Volts	Volts
DC Current	Amps
AC Current	Amps
2 Wire Resistance	Ohms
4 Wire Resistance	Ohms
AC Plus DC Volts	Volts
AC Plus DC Current	Amps
Freq	Hertz
Period	Seconds
Temperature	Degrees Celsius

This function is not guaranteed to return valid data if the user performs other operations on the instrument after the call to Initiate and prior to calling this function. This includes other calls to Fetch Multi Point.

COM Method Prototype

```
HRESULT Measurement.FetchMultiPoint(
                    [in] LONG MaxTimeMilliseconds,
                    [out, retval] SAFEARRAY (DOUBLE) *ReadingArray);
```

C Prototype

```
ViStatus IviDmm_FetchMultiPoint (ViSession Vi,
                                 ViInt32  MaxTimeMilliseconds,
                                 ViInt32 ArraySize,
                                 ViReal64 ReadingArray[],
                                 ViInt32 *ActualPts);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
MaxTimeMilliseconds	Specifies the maximum length of time allowed for the function to complete in milliseconds.	ViInt32
ArraySize	Specifies the number of elements in the readingArray.	ViInt32

Outputs	Description	Base Type
ReadingArray	A user-allocated (for IVI_C) or driver-allocated (for IVI COM) buffer into which the measured values are stored	ViReal64[]
ActualPts	The number of measured values placed in readingArray.	ViInt32

Defined Values for the MaxTimeMilliseconds Parameter

Name	*Description*		
		Language	*Identifier*
Max Time Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		C	IVIDMM_VAL_MAX_TIME_IMMEDIATE
		COM	IviDmmMaxTimeImmediate
Max Time Infinite	The function waits indefinitely for the measurement to complete.		
		C	IVIDMM_VAL_MAX_TIME_INFINITE
		COM	IviDmmMaxTimeInfinite

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Over Range	Overrange warning
Max Time Exceeded	Max time exceeded before the operation completed

11.3.4 Read Multi Point

Description

This function initiates the measurement, waits for the DMM to return to the Idle state, and returns an array of measured values. The number of measurements the DMM takes is determined by the Trigger Count and Sample Count.

After this function executes, each element in the `ReadingArray` parameter is an actual reading or a value indicating that an overrange condition occurred. If an overrange condition occurs, the corresponding `ReadingArray` element contains an IEEE defined NaN (Not a Number) value and the function returns Over Range. Test each element in the `ReadingArray` parameter for overrange with the Is Over Range function.

The value of the Function attribute determines the units for each element of the `ReadingArray` parameter. Refer to Section 11.3.3, Fetch Multi Point for more details.

COM Method Prototype

```
HRESULT Measurement.ReadMultiPoint(
                    [in] LONG MaxTimeMilliseconds,
                    [out, retval] SAFEARRAY (DOUBLE) *ReadingArray);
```

C Prototype

```
ViStatus IviDmm_ReadMultiPoint (ViSession Vi,
                                ViInt32  MaxTimeMilliseconds,
                                ViInt32 ArraySize,
                                ViReal64 ReadingArray[],
                                ViInt32 *ActualPts);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`MaxTimeMilliseconds`	Specifies the maximum length of time allowed for the function to complete in milliseconds.	`ViInt32`
`ArraySize`	Specifies the number of elements in the readingArray.	`ViInt32`

Outputs	Description	Base Type
`ReadingArray`	A user-allocated (for IVI_C) or driver-allocated (for IVI COM) buffer into which the measured values are stored	`ViReal64[]`
`ActualPts`	The number of measured values placed in readingArray.	`ViInt32`

Defined Values for the MaxTimeMilliseconds Parameter

Name	*Description*		
		Language	*Identifier*
Max Time Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		C	`IVIDMM_VAL_MAX_TIME_IMMEDIATE`
		COM	`IviDmmMaxTimeImmediate`
Max Time Infinite	The function waits indefinitely for the measurement to complete.		
		C	`IVIDMM_VAL_MAX_TIME_INFINITE`
		COM	`IviDmmMaxTimeInfinite`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Over Range	Overrange warning
Max Time Exceeded	Max time exceeded before the operation completed

11.4 IviDmmMultiPoint Behavior Model

The following behavior model shows the relationship between the IviDmmMultiPoint extension group and DMM behavior.

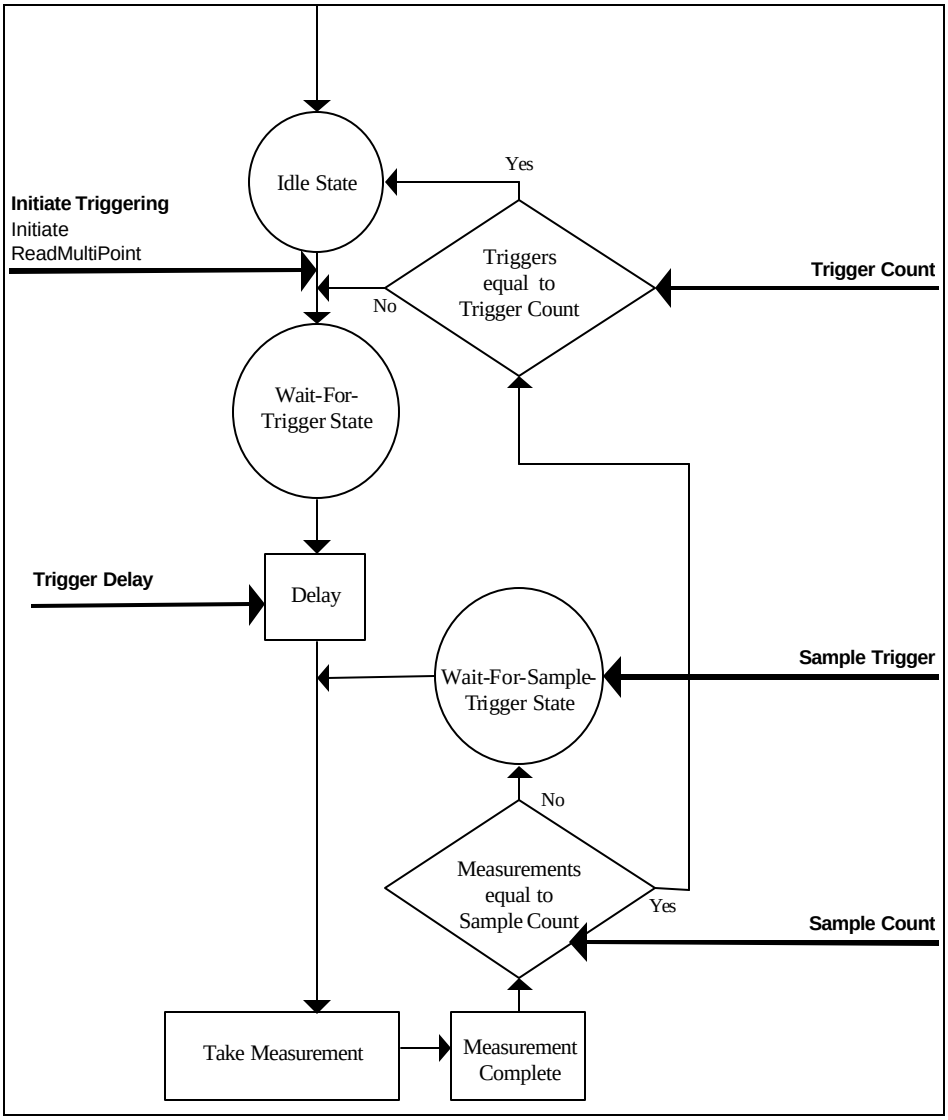


Figure 11-1. IviDmmMultiPoint Behavior Model

The IviDmmMultiPoint behavior model builds upon the fundamental IviDmm behavior model and only documents additional items introduced by the IviDmmMultiPoint extension group. The main state is the Idle state. Typically, the user configures the IviDmmMultiPoint attributes while DMM is in the Idle state. IviDmmMultiPoint attributes can be configured individually with the Set Attribute function or with the high-level Configure Multi Point function.

The Initiate and ReadMultiPoint functions cause the DMM to leave the Idle state and transition to the Wait-For-Trigger state. The ReadMultiPoint function does not return until the measurement process is complete and the DMM returns to the Idle state. The Initiate function returns as soon as the DMM leaves the Idle state.

The IviDmmMultiPoint extension group does not add additional capabilities to the Wait-For-Trigger state.

After the DMM leaves the Wait-For-Trigger state, it then executes a delay. The length of the delay is specified by the attribute Trigger Delay. After the measurement is taken, the DMM then, if it is capable of doing so, generates the Measurement Complete signal.

The DMM then compares the sample count with the number of measurements taken since the last trigger event. The sample count is specified by the attribute Sample Count. If the number measurements is not equal to the sample count the DMM moves to the Wait-For-Sample-Trigger state. The DMM remains in the Wait-For-Sample-Trigger state until the event specified by the attribute Sample Trigger occurs. Then it takes another measurement.

Once the number of measurements taken is equal to the sample count, the DMM then compares the number trigger count with the number of trigger events that have occurred since either the Initiate or Read Multi Point function was called. The trigger count is specified by the attribute Trigger Count. If the number of trigger events is not equal to the trigger count, the DMM returns to the Wait-For-Trigger state.

Once the number of trigger events is equal to the trigger count, the DMM returns to the Idle state. The Fetch Multi Point function is used to retrieve measured data from measurements initiated by the Initiate function. The measurement data returned from the Read Multi Point and Fetch Multi Point functions is acquired after the DMM has left the Wait-For-Trigger state.

12. IviDmmTriggerSlope Extension Group

12.1 IviDmmTriggerSlope Extension Group Overview

The IviDmmTriggerSlope extension group supports DMMs that can specify the polarity of the external trigger signal. It defines an attribute and a function to configure this polarity.

Special Note To Users

Typically the specific driver disables extension groups that the application program does not explicitly use or enable. The IviDmmTriggerSlope extension capability group affects the behavior of the instrument regardless of the value of the Trigger Slope attribute. It is not possible for the specific driver to disable this extension capability group.

Therefore, it is the responsibility of the user to ensure that the slope of the trigger signal is correct for their application. Most DMMs do not have a programmable trigger slope and do not implement the IviDmmTriggerSlope extension capability group. Users should avoid using the IviDmmTriggerSlope extension capability in their test program source code so that they can maximize the set of instruments that they can use interchangeably.

For instrument drivers that implement the IviDmmTriggerSlope extension, the user can set the value of the Trigger Slope attribute in the IVI configuration file. For instruments that do not implement the IviDmmTriggerSlope extension group, the user must ensure that trigger signal that their instrument receives has the correct polarity. This can be done with external circuitry.

12.2 IviDmmTriggerSlope Attributes

The IviDmmTriggerSlope extension group defines the following attribute:

? Trigger Slope

This section describes the behavior and requirements of this attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

12.2.1 Trigger Slope

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Trigger Slope

COM Property Name

Trigger.Slope

COM Enumeration Name

IviDmmTriggerSlopeEnum

C Constant Name

IVIDMM_ATTR_TRIGGER_SLOPE

Description

Specifies the polarity of the external trigger slope. The DMM triggers on either the rising or the falling edge of the external trigger source depending on the value of this attribute.

Defined Values

Name	*Description*		
		Language	*Identifier*
Positive	Sets the trigger event to occur on the rising edge of the trigger pulse.		
		C	IVIDMM_VAL_POSITIVE
		COM	IviDmmTriggerSlopePositive
Negative	Sets the trigger event to occur on the falling edge of the trigger pulse.		
		C	IVIDMM_VAL_NEGATIVE
		COM	IviDmmTriggerSlopeNegative

Compliance Notes

1. If an IVI-C IviDmm specific driver defines additional values for this attribute, the magnitude of the actual values must be greater than or equal to Trigger Slope Specific Extension Base.
2. If an IVI-C IviDmm class driver defines additional values for this attribute, the magnitude of the actual values must be greater than or equal to Trigger Slope Class Extension Base and less than Trigger Slope Specific Extension Base.
3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Trigger Slope Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Trigger Slope Specific Extension Base and Trigger Slope Class Extension Base.

12.3 IviDmmTriggerSlope Functions

The IviDmmTriggerSlope extension group defines the following function:

? Configure Trigger Slope (IVI-C only)

This section describes the behavior and requirements of this function.

12.3.1 Configure Trigger Slope (IVI-C Only)

Description

This function configures the polarity of the external trigger source of the DMM.

COM Method Prototype

```
N/A
(use the Trigger.Slope property)
```

C Prototype

```
ViStatus IviDmm_ConfigureTriggerSlope (ViSession Vi,
                                       ViInt32 Polarity);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Polarity`	Specifies the trigger slope. The driver uses this value to set the Trigger Slope attribute. See the attribute description for more details.	`ViInt32`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

12.4 IviDmmTriggerSlope Behavior Model

The IviDmmTriggerSlope extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

13. IviDmmSoftwareTrigger Extension Group

13.1 IviDmmSoftwareTrigger Extension Group Overview

The IviDmmSoftwareTrigger extension group supports DMMs that can initiate a measurement based on a software trigger signal. The user can send a software trigger to cause the DMM to initiate a measurement.

13.2 IviDmmSoftwareTrigger Functions

The IviDmmSoftwareTrigger extension group defines the following function:

? Send Software Trigger

This section describes the behavior and requirements of this function.

13.2.1 IviDmm_SendSoftwareTrigger

Refer to *IVI-3.3: Standardized Cross Class Capabilities Specification* for the prototype and complete description of this function.

13.3 IviDmmSoftwareTrigger Behavior Model

The behavior model of the IviDmmSoftwareTrigger follows the behavior model of the IviDmmBase capability group as described in Section 4.4, *IviDmmBase Behavior Model* and the IviDmmMultiPoint extension group described in Section 11.4, *IviDmmMultiPoint Behavior Model*. Furthermore, it defines an additional trigger event for the trigger source.

The DMM leaves the Wait-For-Trigger state when it receives a trigger event specified by the Trigger Source attribute. The DMM leaves the Wait-For-Sample-Trigger state when it receieves a trigger event specified by the Sample Trigger attribute When the trigger source or sample trigger is set to Software Trigger, the Send Software Trigger function is used to generate the trigger event . Calling this function causes the DMM to take a measurement.

13.4 IviDmmSoftwareTrigger Compliance Notes

1. IviDmm specific drivers that implement this extension group shall implement the Software Trigger value for the Trigger Source attribute in the IviDmmBase capability group.

14. IviDmmDeviceInfo Extension Group

14.1 IviDmmDeviceInfo Extension Group Overview

The IviDmmDeviceInfo extension group defines a set of read-only attributes for DMMs that can be queried to determine how they are presently configured. These attributes are the aperture time and the aperture time units.

14.2 IviDmmDeviceInfo Attributes

The IviDmmDeviceInfo extension group defines the following attributes:

? Aperture Time

? Aperture Time Units

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

14.2.1 Aperture Time

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	RO	N/A	None	Get Aperture Time Info

COM Property Name

Advanced.ApertureTime

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_APERTURE_TIME

Description

Returns the measurement aperture time based on the present configuration. The units for this attribute are specified by attribute Aperture Time Units.

The aperture time is also known as the integration time.

14.2.2 Aperture Time Units

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	RO	N/A	None	Get Aperture Time Info

COM Property Name

Advanced.ApertureTimeUnits

COM Enumeration Name

IviDmmApertureTimeUnitsEnum

C Constant Name

IVIDMM_ATTR_APERTURE_TIME_UNITS

Description

Returns the units for the Aperture Time attribute.

Defined Values

Name	*Description*		
		Language	*Identifier*
Seconds	Reports that the units for the value returned by Aperture Time are seconds.		
		C	IVIDMM_VAL_SECONDS
		COM	IviDmmApertureSeconds
Power Line Cycles	Reports that the units for the value returned by Aperture Time are Power Line Cycles.		
		C	IVIDMM_VAL_POWER_LINE_CYCLES
		COM	IviDmmAperturePowerLineCycles

14.3 IviDmmDeviceInfo Functions

The IviDmmDeviceInfo extension group defines the following function:

? Get Aperture Time Info (IVI-C only)

This section describes the behavior and requirements of this function.

14.3.1 Get Aperture Time Info (IVI-C Only)

Description

This function returns additional information about the state of the instrument. Specifically, it returns the aperture time and the aperture time units.

COM Method Prototype

```
N/A
(use the Advanced.ApertureTime and Advanced.ApertureTimeUnits properties)
```

C Prototype

```
ViStatus IviDmm_GetApertureTimeInfo (ViSession Vi
                                     ViReal64 *ApertureTime,
                                     ViInt32 *ApertureTimeUnits);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession

Outputs	Description	Base Type
ApertureTime	Returns the value of the Aperture Time attribute. See the attribute description for more details.	ViReal64
ApertureTimeUnits	Returns the value of the Aperture Time Units attribute. See the attribute description for more details.	ViInt32

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

14.4 IviDmmDeviceInfo Behavior Model

The IviDmmDeviceInfo extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

15. IviDmmAutoRangeValue Extension Group

15.1 IviDmmAutoRangeValue Extension Group Overview

The IviDmmAutoRangeValue extension supports DMMs with the capability to return the actual range value when auto ranging.

15.2 IviDmmAutoRangeValue Attributes

The IviDmmAutoRangeValue extension group defines the following attribute:

? Auto Range Value

This section describes the behavior and requirements of this attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

15.2.1 Auto Range Value

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	RO	N/A	None	Get Auto Range Value

COM Property Name

Advanced.ActualRange

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_AUTO_RANGE_VALUE

Description

Returns the actual range that the DMM is currently using, even while it is auto-ranging.

15.3 IviDmmAutoRangeValue Functions

The IviDmmAutoRangeValue extension group defines the following function:

? Get Auto Range Value (IVI-C only)

This section describes the behavior and requirements of this function.

15.3.1 Get Auto Range Value (IVI-C Only)

Description

This function returns the actual range the DMM is currently using, even while it is auto-ranging.

COM Method Prototype

```
N/A
(use the Advanced.ActualRange property)
```

C Prototype

```
ViStatus IviDmm_GetAutoRangeValue (ViSession Vi
                                   ViReal64 *AutoRangeValue);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession

Outputs	Description	Base Type
AutoRangeValue	Returns the value of the Range Value attribute. See the attribute description for more details.	ViReal64

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

15.4 IviDmmAutoRangeValue Behavior Model

The IviDmmAutoRangeValue extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

15.5 IviDmmAutoRangeValue Compliance Notes

1. If the IviDmm specific driver implements this extension group, then it shall also implement the Auto Range On value for the Range attribute in the IviDmmBase capability group.

16. IviDmmAutoZero Extension Group

16.1 IviDmmAutoZero Extension Group Overview

The IviDmmAutoZero extension supports DMMs with the capability to take an auto zero reading. In general, the auto-zero capability of a DMM normalizes all measurements based on a Zero Reading.

16.2 IviDmmAutoZero Attributes

The IviDmmAutoZero extension group defines the following attribute:

? Auto Zero

This section describes the behavior and requirements of this attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

16.2.1 Auto Zero

Data Type	Access	Applies To	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Auto Zero Mode

COM Property Name

Advanced.AutoZero

COM Enumeration Name

IviDmmAutoZeroEnum

C Constant Name

IVIDMM_ATTR_AUTO_ZERO

Description

Specifies the auto-zero mode. When the auto-zero mode is enabled, the DMM internally disconnects the input signal and takes a Zero Reading. The DMM then subtracts the Zero Reading from the measurement. This prevents offset voltages present in the instrument's input circuitry from affecting measurement accuracy.

Defined Values

Name	*Description*		
		Language	*Identifier*
Auto Zero Off	Disables the auto-zero feature.		
		C	IVIDMM_VAL_AUTO_ZERO_OFF
		COM	IviDmmAutoZeroOff
Auto Zero On	Configures the DMM to take a Zero Reading for each measurement. The DMM subtracts the Zero Reading from the value it measures.		
		C	IVIDMM_VAL_AUTO_ZERO_ON
		COM	IviDmmAutoZeroOn
Auto Zero Once	Configures the DMM to take a Zero Reading immediately. The DMM then subtracts this Zero Reading from all subsequent values it measures.		
		C	IVIDMM_VAL_AUTO_ZERO_ONCE
		COM	IviDmmAutoZeroOnce

Compliance Notes

1. If an IVI-C IviDmm specific driver defines additional values for this attribute, the magnitude of the actual values shall be greater than or equal to Auto Zero Specific Extension Base.
2. If an IVI-C IviDmm class driver defines additional values for this attribute, the magnitude of the actual values shall be greater than or equal to Auto Zero Class Extension Base and less than Auto Zero Specific Extension Base.

3. When an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, it is recommended that the actual values of the additional elements be greater than or equal to Auto Zero Specific Extension Base.

See Section 19, *Attribute Value Definitions*, for the definitions of Auto Zero Specific Extension Base and Auto Zero Class Extension Base.

16.3 IviDmmAutoZero Functions

The IviDmmAutoZero extension group defines the following function:

? Configure Auto Zero Mode (IVI-C only)

This section describes the behavior and requirements of this function.

16.3.1 Configure Auto Zero Mode (IVI-C Only)

Description

This function configures the auto zero mode of the DMM.

COM Method Prototype

```
N/A
(use the Advanced.AutoZero property)
```

C Prototype

```
ViStatus IviDmm_ConfigureAutoZeroMode (ViSession Vi
                                       ViInt32 AutoZeroMode);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
AutoZeroMode	Specifies the auto-zero mode The driver uses this value to set the Auto Zero attribute. See the attribute description for more details.	ViInt32

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

16.4 IviDmmAutoZero Behavior Model

The IviDmmAutoZero extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

17. IviDmmPowerLineFrequency Extension Group

17.1 IviDmmPowerLineFrequency Extension Group Overview

The IviDmmPowerLineFrequency extension supports DMMs with the capability to specify the power line frequency.

Special Note To Users

Typically the specific driver disables extension groups that the application program does not explicitly use or enable. The IviDmmPowerLineFrequency extension capability group affects the behavior of the instrument regardless of the value of the Powerline Freq attribute. It is not possible for the specific driver to disable this extension capability group.

Therefore, it is the responsibility of the user to ensure that the power line frequency is correct for their application. Most DMMs do not have a programmable power line frequency and do not implement the IviDmmPowerLineFrequency extension capability group. Users should avoid using the IviDmmPowerLineFrequency extension group in their test program source code so that they can maximize the set of instruments that they can use interchangeably.

For instrument drivers that implement the IviDmmPowerLineFrequency extension, the user can set the value of the Powerline Freq attribute in the IVI configuration file. For instruments that do not implement the IviDmmPowerLineFrequency extension group, the user must ensure that their instrument is set to use the correct power line frequency. Users can manually change the power line frequency setting on most DMMs by means of a switch on the instrument's back panel.

17.2 IviDmmPowerLineFrequency Attributes

The IviDmmPowerLineFrequency extension group defines the following attribute:

? Powerline Frequency

This section describes the behavior and requirements of this attribute. The actual value for each attribute ID is defined in Section 18, *IviDmm Attribute ID Definitions.*

17.2.1 Powerline Frequency

Data Type	Access	Applies To	Coercion	High Level Functions
ViReal64	R/W	N/A	None	Configure Power Line Frequency

COM Property Name

Advanced.PowerlineFrequency

COM Enumeration Name

N/A

C Constant Name

IVIDMM_ATTR_POWERLINE_FREQ

Description

Specifies the power line frequency in Hertz.

17.3 IviDmmPowerLineFrequency Functions

The IviDmmPowerLineFrequency extension group defines the following function:

? Configure Power Line Frequency (IVI-C only)

This section describes the behavior and requirements of this function.

17.3.1 Configure Power Line Frequency (IVI-C Only)

Description

This function configures the power line frequency of the DMM.

COM Method Prototype

```
N/A
(use the Advanced.PowerlineFrequency property)
```

C Prototype

```
ViStatus IviDmm_ConfigurePowerLineFrequency (ViSession Vi
                                             ViReal64 PowerLineFreq);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`PowerLineFreq`	Specifies the power line frequency The driver uses this value to set the Powerline Freq attribute. See the attribute description for more details.	`ViReal64`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

17.4 IviDmmPowerLineFrequency Behavior Model

The IviDmmPowerLineFrequency extension group follows the same behavior model as the IviDmmBase capability group described in Section 4.4, *IviDmmBase Behavior Model*.

18. IviDmm Attribute ID Definitions

The following table defines the ID value for all IviDmm class attributes.

Table 18-1. IviDmm Attributes ID Values

Attribute Name	ID Definition
Function	IVI_CLASS_ATTR_BASE + 1
Range	IVI_CLASS_ATTR_BASE + 2
Resolution Absolute	IVI_CLASS_ATTR_BASE + 8
Trigger Source	IVI_CLASS_ATTR_BASE + 4
Trigger Delay	IVI_CLASS_ATTR_BASE + 5
AC Min Frequency	IVI_CLASS_ATTR_BASE + 6
AC Max Frequency	IVI_CLASS_ATTR_BASE + 7
Freq Voltage Range	IVI_CLASS_ATTR_BASE + 101
Temperature Transducer Type	IVI_CLASS_ATTR_BASE + 201
Thermocouple Type	IVI_CLASS_ATTR_BASE + 231
Thermocouple Reference Junction Type	IVI_CLASS_ATTR_BASE + 232
Thermocouple Fixed Reference Junction	IVI_CLASS_ATTR_BASE + 233
RTD Alpha	IVI_CLASS_ATTR_BASE + 241
RTD Resistance	IVI_CLASS_ATTR_BASE + 242
Thermistor Resistance	IVI_CLASS_ATTR_BASE + 251
Sample Count	IVI_CLASS_ATTR_BASE + 301
Sample Trigger	IVI_CLASS_ATTR_BASE + 302
Sample Interval	IVI_CLASS_ATTR_BASE + 303
Trigger Count	IVI_CLASS_ATTR_BASE + 304
Measure Complete Destination	IVI_CLASS_ATTR_BASE + 305
Aperture Time	IVI_CLASS_ATTR_BASE + 321
Aperture Time Units	IVI_CLASS_ATTR_BASE + 322
Auto Range Value	IVI_CLASS_ATTR_BASE + 331
Auto Zero	IVI_CLASS_ATTR_BASE + 332
Powerline Freq	IVI_CLASS_ATTR_BASE + 333
Trigger Slope	IVI_CLASS_ATTR_BASE + 334

18.1 IviDmm Obsolete Attribute Names

The following attribute names are reserved by the IviDmm specification 1.0. Future versions of this specification cannot use these names:

? Resolution

18.2 IviDmm Obsolete Attribute ID Values

The following attribute ID values are reserved by the IviDmm specification 1.0. Future versions of this specification cannot use these values:

? `IVI_CLASS_ATTR_BASE + 3`

19. IviDmm Attribute Value Definitions

This section specifies the actual value for each defined attribute value.

Aperture Time Units

Value Name	*Language*	*Identifier*	*Actual Value*
Seconds	C	IVIDMM_VAL_SECONDS	0
	COM	IviDmmApertureSeconds	0
Power Line Cycles	C	IVIDMM_VAL_POWER_LINE_CYCLES	1
	COM	IviDmmAperturePowerLineCycles	1

Auto Zero

Value Name	*Language*	*Identifier*	*Actual Value*
Auto Zero Off	C	IVIDMM_VAL_AUTO_ZERO_OFF	0
	COM	IviDmmAutoZeroOff	0
Auto Zero On	C	IVIDMM_VAL_AUTO_ZERO_ON	1
	COM	IviDmmAutoZeroOn	1
Auto Zero Once	C	IVIDMM_VAL_AUTO_ZERO_ONCE	2
	COM	IviDmmAutoZeroOnce	2
	C	IVIDMM_VAL_AUTO_ZERO_CLASS_EXT_BASE	100
Auto Zero Specific Extension Base	C	IVIDMM_VAL_AUTO_ZERO_SPECIFIC_EXT_BASE	1000
	COM	N/A	1000

Frequency Voltage Range

Value Name	*Language*	*Identifier*	*Actual Value*
Auto Range On	C	IVIDMM_VAL_AUTO_RANGE_ON	-1.0
	COM	IviDmmFrequencyVoltageRangeAutoRangeOn	-1.0
Auto Range Off	C	IVIDMM_VAL_AUTO_RANGE_OFF	-2.0
	COM	IviDmmFrequencyVoltageRangeAutoRangeOff	-2.0
	C	IVIDMM_VAL_FREQ_VOLT_RANGE_CLASS_EXT_BASE	-100.0
Frequency Volt Range Specific Extension Base	C	IVIDMM_VAL_FREQ_VOLT_RANGE_SPECIFIC_EXT_BASE	-1000.0
	COM	N/A	-1000.0

Function

Value Name	*Language*	*Identifier*	*Actual Value*
DC Volts	C	IVIDMM_VAL_DC_VOLTS	1
	COM	IviDmmFunctionDCVolts	1
AC Volts	C	IVIDMM_VAL_AC_VOLTS	2
	COM	IviDmmFunctionACVolts	2
DC Current	C	IVIDMM_VAL_DC_CURRENT	3
	COM	IviDmmFunctionDCCurrent	3
AC Current	C	IVIDMM_VAL_AC_CURRENT	4
	COM	IviDmmFunctionACCurrent	4
2 Wire Resistance	C	IVIDMM_VAL_2_WIRE_RES	5
	COM	IviDmmFunction2WireRes	5
4 Wire Resistance	C	IVIDMM_VAL_4_WIRE_RES	101
	COM	IviDmmFunction4WireRes	101
AC Plus DC Volts	C	IVIDMM_VAL_AC_PLUS_DC_VOLTS	106
	COM	IviDmmFunctionACPlusDCVolts	106
AC Plus DC Current	C	IVIDMM_VAL_AC_PLUS_DC_CURRENT	107
	COM	IviDmmFunctionACPlusDCCurrent	107
Frequency	C	IVIDMM_VAL_FREQ	104
	COM	IviDmmFunctionFreq	104
Period	C	IVIDMM_VAL_PERIOD	105
	COM	IviDmmFunctionPeriod	105
Temperature	C	IVIDMM_VAL_TEMPERATURE	108
	COM	IviDmmFunctionTemperature	108
	C	IVIDMM_VAL_CLASS_EXT_BASE	500
Specific Extension Base	C	IVIDMM_VAL_SPECIFIC_EXT_BASE	1000
	COM	N/A	1000

The following values are reserved by the IviDmm specification 1.0 for the IVIDMM_ATTR_FUNCTION attribute. Future versions of this specification cannot use these values for this attribute:

? 102

? 103

? 109

? 110

? 111

Measure Complete Destination

Value Name	*Language*	*Identifier*	*Actual Value*
None	C	IVIDMM_VAL_NONE	-1
	COM	IviDmmMeasCompleteDestNone	-1
External	C	IVIDMM_VAL_EXTERNAL	2
	COM	IviDmmMeasCompleteDestExternal	2
TTL0	C	IVIDMM_VAL_TTL0	111
	COM	IviDmmMeasCompleteDestTTL0	111
TTL1	C	IVIDMM_VAL_TTL1	112
	COM	IviDmmMeasCompleteDestTTL1	112
TTL2	C	IVIDMM_VAL_TTL2	113
	COM	IviDmmMeasCompleteDestTTL2	113
TTL3	C	IVIDMM_VAL_TTL3	114
	COM	IviDmmMeasCompleteDestTTL3	114
TTL4	C	IVIDMM_VAL_TTL4	115
	COM	IviDmmMeasCompleteDestTTL4	115
TTL5	C	IVIDMM_VAL_TTL5	116
	COM	IviDmmMeasCompleteDestTTL5	116
TTL6	C	IVIDMM_VAL_TTL6	117
	COM	IviDmmMeasCompleteDestTTL6	117
TTL7	C	IVIDMM_VAL_TTL7	118
	COM	IviDmmMeasCompleteDestTTL7	118
ECL0	C	IVIDMM_VAL_ECL0	119
	COM	IviDmmMeasCompleteDestECL0	119
ECL1	C	IVIDMM_VAL_ECL1	120
	COM	IviDmmMeasCompleteDestECL1	120
PXI Star	C	IVIDMM_VAL_PXI_STAR	131
	COM	IviDmmMeasCompleteDestPXIStar	131
RTSI 0	C	IVIDMM_VAL_RTSI_0	140
	COM	IviDmmMeasCompleteDestRTSI0	140
RTSI 1	C	IVIDMM_VAL_RTSI_1	141
	COM	IviDmmMeasCompleteDestRTSI1	141
RTSI 2	C	IVIDMM_VAL_RTSI_2	142
	COM	IviDmmMeasCompleteDestRTSI2	142
RTSI 3	C	IVIDMM_VAL_RTSI_3	143
	COM	IviDmmMeasCompleteDestRTSI3	143
RTSI 4	C	IVIDMM_VAL_RTSI_4	144
	COM	IviDmmMeasCompleteDestRTSI4	144

Value Name	*Language*	*Identifier*	*Actual Value*
RTSI 5	C	IVIDMM_VAL_RTSI_5	145
	COM	IviDmmMeasCompleteDestRTSI5	145
RTSI 6	C	IVIDMM_VAL_RTSI_6	146
	COM	IviDmmMeasCompleteDestRTSI6	146
	C	IVIDMM_VAL_TRIGGER_SOURCE_CLASS_EXT_BASE	500
Trigger Source Specific Extension Base	C	IVIDMM_VAL_TRIGGER_SOURCE_SPECIFIC_EXT_BASE	1000
	COM	N/A	1000

Range

Value Name	*Language*	*Identifier*	*Actual Value*
Auto Range On	C	IVIDMM_VAL_AUTO_RANGE_ON	-1.0
	COM	IviDmmAutoRangeOn	-1.0
Auto Range Off	C	IVIDMM_VAL_AUTO_RANGE_OFF	-2.0
	COM	IviDmmAutoRangeOff	-2.0
Auto Range Once	C	IVIDMM_VAL_AUTO_RANGE_ONCE	-3.0
	COM	IviDmmAutoRangeOnce	-3.0
	C	IVIDMM_VAL_RANGE_CLASS_EXT_BASE	-100.0
Range Specific Extension Base	C	IVIDMM_VAL_RANGE_SPECIFIC_EXT_BASE	-1000.0
	COM	N/A	-1000.0

Sample Trigger

Value Name	*Language*	*Identifier*	*Actual Value*
Immediate	C	IVIDMM_VAL_IMMEDIATE	1
	COM	IviDmmSampleTriggerImmediate	1
External	C	IVIDMM_VAL_EXTERNAL	2
	COM	IviDmmSampleTriggerExternal	2
Software Trigger	C	IVIDMM_VAL_SOFTWARE_TRIG	3
	COM	IviDmmSampleTriggerSwTrigFunc	3
TTL0	C	IVIDMM_VAL_TTL0	111
	COM	IviDmmSampleTriggerTTL0	111
TTL1	C	IVIDMM_VAL_TTL1	112
	COM	IviDmmSampleTriggerTTL1	112

Value Name	*Language*	*Identifier*	*Actual Value*
TTL2	C	IVIDMM_VAL_TTL2	113
	COM	IviDmmSampleTriggerTTL2	113
TTL3	C	IVIDMM_VAL_TTL3	114
	COM	IviDmmSampleTriggerTTL3	114
TTL4	C	IVIDMM_VAL_TTL4	115
	COM	IviDmmSampleTriggerTTL4	115
TTL5	C	IVIDMM_VAL_TTL5	116
	COM	IviDmmSampleTriggerTTL5	116
TTL6	C	IVIDMM_VAL_TTL6	117
	COM	IviDmmSampleTriggerTTL6	117
TTL7	C	IVIDMM_VAL_TTL7	118
	COM	IviDmmSampleTriggerTTL7	118
ECL0	C	IVIDMM_VAL_ECL0	119
	COM	IviDmmSampleTriggerECL0	119
ECL1	C	IVIDMM_VAL_ECL1	120
	COM	IviDmmSampleTriggerECL1	120
PXI Star	C	IVIDMM_VAL_PXI_STAR	131
	COM	IviDmmSampleTriggerPXIStar	131
RTSI 0	C	IVIDMM_VAL_RTSI_0	140
	COM	IviDmmSampleTriggerRTSI0	140
RTSI 1	C	IVIDMM_VAL_RTSI_1	141
	COM	IviDmmSampleTriggerRTSI1	141
RTSI 2	C	IVIDMM_VAL_RTSI_2	142
	COM	IviDmmSampleTriggerRTSI2	142
RTSI 3	C	IVIDMM_VAL_RTSI_3	143
	COM	IviDmmSampleTriggerRTSI3	143
RTSI 4	C	IVIDMM_VAL_RTSI_4	144
	COM	IviDmmSampleTriggerRTSI4	144
RTSI 5	C	IVIDMM_VAL_RTSI_5	145
	COM	IviDmmSampleTriggerRTSI5	145
RTSI 6	C	IVIDMM_VAL_RTSI_6	146
	COM	IviDmmSampleTriggerRTSI6	146
Interval	C	IVIDMM_VAL_INTERVAL	10
	COM	IviDmmSampleTriggerInterval	10
	C	IVIDMM_VAL_TRIGGER_SOURCE_CLASS_EXT_BASE	500
Trigger Source Specific Extension Base	C	IVIDMM_VAL_TRIGGER_SOURCE_SPECIFIC_EXT_BASE	1000

Value Name	*Language*	*Identifier*	*Actual Value*
	COM	N/A	1000

The following values are reserved by the IviDmm specification 1.0 for the Sample Trigger attribute. Future versions of this specification cannot use these values:

? 101

Thermocouple Reference Junction Type

Value Name	*Language*	*Identifier*	*Actual Value*
Thermocouple Reference Junction Internal	C	IVIDMM_VAL_TEMP_REF_JUNC_INTERNAL	1
	COM	IviDmmRefJunctionTypeInternal	1
Thermocouple Reference Junction Fixed	C	IVIDMM_VAL_TEMP_REF_JUNC_FIXED	2
	COM	IviDmmRefJunctionTypeFixed	2
	C	IVIDMM_VAL_TEMP_REF_JUNC_CLASS_EXT_BASE	100
Reference Junction Specific Extension Base	C	IVIDMM_VAL_TEMP_REF_JUNC_SPECIFIC_EXT_BASE	1000
	COM	N/A	1000

Thermocouple Type

Value Name	*Language*	*Identifier*	*Actual Value*
Thermocouple B	C	IVIDMM_VAL_TEMP_TC_B	1
	COM	IviDmmThermocoupleTypeB	1
Thermocouple C	C	IVIDMM_VAL_TEMP_TC_C	2
	COM	IviDmmThermocoupleTypeC	2
Thermocouple D	C	IVIDMM_VAL_TEMP_TC_D	3
	COM	IviDmmThermocoupleTypeD	3
Thermocouple E	C	IVIDMM_VAL_TEMP_TC_E	4
	COM	IviDmmThermocoupleTypeE	4
Thermocouple G	C	IVIDMM_VAL_TEMP_TC_G	5
	COM	IviDmmThermocoupleTypeG	5
Thermocouple J	C	IVIDMM_VAL_TEMP_TC_J	6
	COM	IviDmmThermocoupleTypeJ	6
Thermocouple K	C	IVIDMM_VAL_TEMP_TC_K	7
	COM	IviDmmThermocoupleTypeK	7
Thermocouple N	C	IVIDMM_VAL_TEMP_TC_N	8
	COM	IviDmmThermocoupleTypeN	8
Thermocouple R	C	IVIDMM_VAL_TEMP_TC_R	9

Value Name	*Language*	*Identifier*	*Actual Value*
	COM	IviDmmThermocoupleTypeR	9
Thermocouple S	C	IVIDMM_VAL_TEMP_TC_S	10
	COM	IviDmmThermocoupleTypeS	10
Thermocouple T	C	IVIDMM_VAL_TEMP_TC_T	11
	COM	IviDmmThermocoupleTypeT	11
Thermocouple U	C	IVIDMM_VAL_TEMP_TC_U	12
	COM	IviDmmThermocoupleTypeU	12
Thermocouple V	C	IVIDMM_VAL_TEMP_TC_V	13
	COM	IviDmmThermocoupleTypeV	13
	C	IVIDMM_VAL_TEMP_TC_TYPE_CLASS_EXT_BASE	100
Thermocouple Type Specific Extension Base	C	IVIDMM_VAL_TEMP_TC_TYPE_SPECIFIC_EXT_BASE	1000
	COM	N/A	1000

Temperature Transducer Type

Value Name	*Language*	*Identifier*	*Actual Value*
Thermocouple	C	IVIDMM_VAL_THERMOCOUPLE	1
	COM	IviDmmTransducerTypeThermocouple	1
Thermistor	C	IVIDMM_VAL_THERMISTOR	2
	COM	IviDmmTransducerTypeThermistor	2
2 Wire RTD	C	IVIDMM_VAL_2_WIRE_RTD	3
	COM	IviDmmTransducerType2WireRtd	3
4 Wire RTD	C	IVIDMM_VAL_4_WIRE_RTD	4
	COM	IviDmmTransducerType4WireRtd	4
	C	IVIDMM_VAL_TRANSDUCER_CLASS_EXT_BASE	100
Transducer Specific Extension Base	C	IVIDMM_VAL_TRANSDUCER_SPECIFIC_EXT_BASE	1000
	COM	N/A	1000

Trigger Delay

Value Name	*Language*	*Identifier*	*Actual Value*
Auto Delay On	C	IVIDMM_VAL_AUTO_DELAY_ON	-1.0
	COM	IviDmmTriggerDelayAutoDelayOn	-1.0
Auto Delay Off	C	IVIDMM_VAL_AUTO_DELAY_OFF	-2.0
	COM	IviDmmTriggerDelayAutoDelayOff	-2.0
	C	IVIDMM_VAL_TRIGGER_DELAY_CLASS_EXT_BASE	-100.0
Trigger Delay Specific Extension Base	C	IVIDMM_VAL_TRIGGER_DELAY_SPECIFIC_EXT_BASE	-1000.0
	COM	N/A	-1000.0

Trigger Slope

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	C	IVIDMM_VAL_POSITIVE	0
	COM	IviDmmTriggerSlopePositive	0
Negative	C	IVIDMM_VAL_NEGATIVE	1
	COM	IviDmmTriggerSlopeNegative	1
	C	IVIDMM_VAL_TRIGGER_SLOPE_CLASS_EXT_BASE	100
Trigger Slope Specific Extension Base	C	IVIDMM_VAL_TRIGGER_SLOPE_SPECIFIC_EXT_BASE	1000
	COM	N/A	1000

Trigger Source

Value Name	*Language*	*Identifier*	*Actual Value*
Immediate	C	IVIDMM_VAL_IMMEDIATE	1
	COM	IviDmmTriggerSourceImmediate	1
External	C	IVIDMM_VAL_EXTERNAL	2
	COM	IviDmmTriggerSourceExternal	2
Software Trigger	C	IVIDMM_VAL_SOFTWARE_TRIG	3
	COM	IviDmmTriggerSourceSwTrigFunc	3
TTL0	C	IVIDMM_VAL_TTL0	111
	COM	IviDmmTriggerSourceTTL0	111
TTL1	C	IVIDMM_VAL_TTL1	112
	COM	IviDmmTriggerSourceTTL1	112
TTL2	C	IVIDMM_VAL_TTL2	113
	COM	IviDmmTriggerSourceTTL2	113
TTL3	C	IVIDMM_VAL_TTL3	114
	COM	IviDmmTriggerSourceTTL3	114
TTL4	C	IVIDMM_VAL_TTL4	115
	COM	IviDmmTriggerSourceTTL4	115
TTL5	C	IVIDMM_VAL_TTL5	116
	COM	IviDmmTriggerSourceTTL5	116
TTL6	C	IVIDMM_VAL_TTL6	117
	COM	IviDmmTriggerSourceTTL6	117
TTL7	C	IVIDMM_VAL_TTL7	118
	COM	IviDmmTriggerSourceTTL7	118
ECL0	C	IVIDMM_VAL_ECL0	119
	COM	IviDmmTriggerSourceECL0	119
ECL1	C	IVIDMM_VAL_ECL1	120
	COM	IviDmmTriggerSourceECL1	120
PXI Star	C	IVIDMM_VAL_PXI_STAR	131
	COM	IviDmmTriggerSourcePXIStar	131
RTSI 0	C	IVIDMM_VAL_RTSI_0	140
	COM	IviDmmTriggerSourceRTSI0	140
RTSI 1	C	IVIDMM_VAL_RTSI_1	141
	COM	IviDmmTriggerSourceRTSI1	141
RTSI 2	C	IVIDMM_VAL_RTSI_2	142
	COM	IviDmmTriggerSourceRTSI2	142
RTSI 3	C	IVIDMM_VAL_RTSI_3	143
	COM	IviDmmTriggerSourceRTSI3	143

Value Name	*Language*	*Identifier*	*Actual Value*
RTSI 4	C	IVIDMM_VAL_RTSI_4	144
	COM	IviDmmTriggerSourceRTSI4	144
RTSI 5	C	IVIDMM_VAL_RTSI_5	145
	COM	IviDmmTriggerSourceRTSI5	145
RTSI 6	C	IVIDMM_VAL_RTSI_6	146
	COM	IviDmmTriggerSourceRTSI6	146
	C	IVIDMM_VAL_TRIGGER_SOURCE_CLASS_EXT_BASE	500
Trigger Source Specific Extension Base	C	IVIDMM_VAL_TRIGGER_SOURCE_SPECIFIC_EXT_BASE	1000
	COM	N/A	1000

The following values are reserved by the IviDmm specification 1.0 for the IVIDMM_ATTR_TRIGGER_SOURCE attribute. Future versions of this specification cannot use these values:

? 101

19.1 IviDmm Obsolete Attribute Value Names

The following attribute value names are reserved by the IviDmm specification 1.0. Future versions of this specification cannot use these names:

? IVIDMM_VAL_DIODE

? IVIDMM_VAL_CONTINUITY

? IVIDMM_VAL_TEMP_C

? IVIDMM_VAL_TEMP_F

? IVIDMM_VAL_SIEMENS

? IVIDMM_VAL_COULOMBS

? IVIDMM_VAL_3_5_DIGITS

? IVIDMM_VAL_4_DIGITS

? IVIDMM_VAL_4_5_DIGITS

? IVIDMM_VAL_5_DIGITS

? IVIDMM_VAL_5_5_DIGITS

? IVIDMM_VAL_6_DIGITS

? IVIDMM_VAL_6_5_DIGITS

? IVIDMM_VAL_7_DIGITS

? IVIDMM_VAL_7_5_DIGITS

? IVIDMM_VAL_50_HERTZ

? IVIDMM_VAL_60_HERTZ

? IVIDMM_VAL_400_HERTZ

? IVIDMM_VAL_GPIB_GET

- IVIDMM_VAL_SW_TRIG_FUNC

20. IviDmm Function Parameter Value Definitions

This section specifies the actual values for each function parameter that defines values.

Read

Parameter: `MaxTimeMilliseconds`

Value Name	*Language*	*Identifier*	*Actual (hex)*
Max Time Immediate	C	`IVIDMM_VAL_MAX_TIME_IMMEDIATE`	`0`
	COM	`IviDmmMaxTimeImmediate`	`0`
Max Time Infinite	C	`IVIDMM_VAL_MAX_TIME_INFINITE`	`0xFFFFFFFF`
	COM	`IviDmmMaxTimeInfinite`	`0xFFFFFFFF`

Fetch

Parameter: `MaxTimeMilliseconds`

Same as defined for the `MaxTimeMilliseconds` parameter of the `IviDmm_Read` function.

Read Multi Point

Parameter: `MaxTimeMilliseconds`

Same as defined for the `MaxTimeMilliseconds` parameter of the `IviDmm_Read` function.

IviDmm_FetchMultiPoint

Parameter: `MaxTimeMilliseconds`

Same as defined for the `MaxTimeMilliseconds` parameter of the `IviDmm_Read` function.

21. IviDmm Error and Completion Code Value Definitions

The table below specifies the actual value for each status code that the IviDmm class specification defines.

Table 21-1. IviDmm Error and Completion Codes

Error Name	*Description*		
	Language	*Identifier*	*Value(hex)*
Max Time Exceeded	Max Time Exceeded		
	C	IVIDMM_ERROR_MAX_TIME_EXCEEDED	0xBFFA2003
	COM	E_IVIDMM_MAX_TIME_EXCEEDED	0x80042003
Trigger Not Software	The trigger source is not set to software trigger.		
	C	IVIDMM_ERROR_TRIGGER_NOT_SOFTWARE	0xBFFA1001
	COM	E_IVIDMM_TRIGGER_NOT_SOFTWARE	0x80041001
Over Range	Over Range		
	C	IVIDMM_WARN_OVER_RANGE	0x3FFA2001
	COM	S_IVIDMM_OVER_RANGE	0x00042001

Table 21-2 defines the recommended format of the message string associated with the errors. In C, these strings are returned by the Get Error function. In COM, these strings are the description contained in the ErrorInfo object.

Note: In the description string table entries listed below, **%s** is always used to represent the component name.

Table 21-2. IviDmm Error Message Strings

Name	Message String
Max Time Exceeded	"%s: Max time exceeded"
Trigger Not Software	"%s: Trigger source is not set to software trigger."
Over Range	"%s: Over range"

21.1 IviDmm Obsolete Error and Completion Code Names

The following error and completion codes names are reserved by the IviDmm specification 1.0. Future versions of this specification cannot use these names:

? Accuracy Unknown

? Accuracy Unknown While Autoranging

21.2 IviDmm Obsolete Error and Completion Code Values

The following error and completion codes values are reserved by the IviDmm specification 1.0. Future versions of this specification cannot use these values:

? `IVI_CLASS_ERROR_BASE + 1`

? `IVI_CLASS_ERROR_BASE + 2`

22. IviDmm Hierarchies

22.1 IviDmm COM Hierarchy

The full IviDmm COM Hierarchy includes the Inherent Capabilities Hierarchy as defined in Section 4.1, *COM Inherent Capabilities* of *IVI-3.2: Ineherent Capabilities Specification*. To avoid redundancy, it is omitted from Table 2-1..

Table 22-1. IviDmm COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
Configure	Configure Measurement	M
Function	Function	P
Range	Range	P
Resolution	Resolution Absolute	P
AC		
ConfigureBandwidth	Configure AC Bandwidth	M
FrequencyMax	AC Max Freq	P
FrequencyMin	AC Min Freq	P
Advanced		
ActualRange	Auto Range Value	P
ApertureTime	Aperture Time	P
ApertureTimeUnits	Aperture Time Units	P
AutoZero	Auto Zero	P
PowerlineFrequency	Powerline Freq	P
Frequency		
VoltageRange	Freq Voltage Range	P
Measurement		
Initiate	Initiate	M
Abort	Abort	M
Fetch	Fetch	M
FetchMultiPoint	Fetch Multi Point	M
Read	Read	M
ReadMultiPoint	Read Multi Point	M
SendSoftwareTrigger	Send Software Trigger	M
IsOverRange	Is Over Range	M
Temperature		
TransducerType	Temperature Transducer Type	P
RTD		
Configure	Configure RTD	M
Alpha	Temperature RTD Alpha	P
Resistance	Temperature RTD Resistance	P

Table 22-1. IviDmm COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
Thermocouple		
Configure	Configure Thermocouple	M
FixedRefJunction	Temp TC Fixed Reference Junction	P
RefJunctionType	Temp TC Reference Junction Type	P
Type	Temp TC Type	P
Thermistor		
Resistance	Temperature Thermistor Resistance	P
Trigger		
Configure	Configure Trigger	M
Slope	Trigger Slope	P
Delay	Trigger Delay	P
Source	Trigger Source	P
MultiPoint		
Configure	Configure Multi Point	M
MeasurementComplete	Measurement Complete Dest	P
Count	Trigger Count	P
SampleCount	Sample Count	P
SampleInterval	Sample Interval	P
SampleTrigger	Sample Trigger	P

22.1.1 IviDmm COM Interfaces

In addition to implementing IVI inherent capabilities interfaces, IviDmm-interfaces contain interface reference properties for accessing the following IviDmm interfaces:

- IIviDmmAC
- IIviDmmAdvanced
- IIviDmmFrequency
- IIviDmmMeasurement
- IIviDmmTemperature
- IIviDmmTrigger

The IIviDmmTemperature interface contains interface reference properties for accessing additional the following IviDmm temperature interfaces:

- IIviDmmRTD
- IIviDmmThermocouple
- IIviDmmThermistor

The IIviDmmTrigger interface contains interface reference properties for accessing additional the following IviDmm trigger interfaces:

- IIviDmmMultiPoint

Table 22-2. IviDmm lists the interfaces that this specification defines and their GUIDs.

Table 22-2. IviDmm Interface GUIDs

Interface	GUID
IIviDmm	{47ed51e8-a398-11d4-ba58-000064657374}
IIviDmmAC	{47ed51ec-a398-11d4-ba58-000064657374}
IIviDmmAdvanced	{47ed51ed-a398-11d4-ba58-000064657374}
IIviDmmFrequency	{47ed51ee-a398-11d4-ba58-000064657374}
IiviDmmMeasurement	{47ed51e9-a398-11d4-ba58-000064657374}
IIviDmmTemperature	{47ed51ef-a398-11d4-ba58-000064657374}
IIviDmmTrigger	{47ed51ea-a398-11d4-ba58-000064657374}
IIviDmmRTD	{47ed51f0-a398-11d4-ba58-000064657374}
IIviDmmThermocouple	{47ed51f1-a398-11d4-ba58-000064657374}
IIviDmmThermistor	{47ed51f2-a398-11d4-ba58-000064657374}
IIviDmmMultiPoint	{47ed51eb-a398-11d4-ba58-000064657374}

22.1.2 Interface Reference Properties

Interface reference properties are used to navigate the IviDmm COM hierarchy. This section describes the interface reference properties that the IIviDmm, IIviDmmTemperature, and IIviDmmTrigger interfaces define.

22.1.2.1 AC

Data Type	Access
IIviDmmAC*	RO

COM Property Name

AC

Description

Returns a pointer to the IIviDmmAC interface.

22.1.2.2 Advanced

Data Type	Access
IIviDmmAdvanced*	RO

COM Property Name

Advanced

Description

Returns a pointer to the IIviDmmAdvanced interface.

22.1.2.3 Frequency

Data Type	Access
IIviDmmFrequency*	RO

COM Property Name

Frequency

Description

Returns a pointer to the IIviDmmFrequency interface.

22.1.2.4 Measurement

Data Type	Access
IIviDmmMeasurement*	RO

COM Property Name

`Measurement`

Description

Returns a pointer to the IIviDmmMeasurement interface.

22.1.2.5 Temperature

Data Type	Access
`IIviDmmTemperature*`	RO

COM Property Name

`Temperature`

Description

Returns a pointer to the IIviDmmTemperature interface.

22.1.2.6 Trigger

Data Type	Access
`IIviDmmTrigger*`	RO

COM Property Name

`Trigger`

Description

Returns a pointer to the IIviDmmTrigger interface.

22.1.2.7 RTD

Data Type	Access
`IIviDmmRTD*`	RO

COM Property Name

`Temperature.RTD`

Description

Returns a pointer to the IIviDmmRTD interface.

22.1.2.8 Thermocouple

Data Type	Access
`IIviDmmThermocouple*`	RO

COM Property Name

`Temperature.Thermocouple`

Description

Returns a pointer to the IIviDmmThermocouple interface.

22.1.2.9 Thermistor

Data Type	Access
`IIviDmmThermistor*`	RO

COM Property Name

`Temperature.Thermistor`

Description

Returns a pointer to the IIviDmmThermistor interface.

22.1.2.10 Multipoint

Data Type	Access
`IIviDmmMultiPoint*`	RO

COM Property Name

`Trigger.MultiPoint`

Description

Returns a pointer to the IIviDmmMultipoint interface.

22.1.3 IviDmm COM Category

The IviDmm class COM Category shall be “IviDmm”, and the Category ID (CATID) shall be {47ed5154-a398-11d4-ba58-000064657374}.

22.2 IviDmm C Function Hierarchy

The IviDmm class function hierarchy is shown in the following table.

Table 22-1. IviDmm Function Hierarchy

Name or Class	Function Name
Configuration…	
Configure Measurement	`IviDmm_ConfigureMeasurement`
Specific Measurements…	
Configure AC Bandwidth	`IviDmm_ConfigureACBandwidth`
Configure Frequency Voltage Range	`IviDmm_ConfigureFrequencyVoltageRange`
Temperature…	
Configure Transducer Type	`IviDmm_ConfigureTransducerType`
Configure Thermocouple	`IviDmm_ConfigureThermocouple`
Configure Fixed Reference Junction	`IviDmm_ConfigureFixedRefJunction`
Configure RTD	`IviDmm_ConfigureRTD`
Configure Thermistor	`IviDmm_ConfigureThermistor`
Trigger…	
Configure Trigger	`IviDmm_ConfigureTrigger`
Configure Trigger Slope	`IviDmm_ConfigureTriggerSlope`
MultiPoint…	
Configure Multi-Point	`IviDmm_ConfigureMultiPoint`
Configure Measurement Complete Destination	`IviDmm_ConfigureMeasCompleteDest`
Measurement Operation Options…	
Configure Auto Zero Mode	`IviDmm_ConfigureAutoZeroMode`
Configure Power Line Frequency	`IviDmm_ConfigurePowerLineFrequency`
Configuration Information…	
Get Auto Range Value	`IviDmm_GetAutoRangeValue`
Get Aperture Time Info	`IviDmm_GetApertureTimeInfo`
Measurement…	
Read	`IviDmm_Read`
Read Multi-Point	`IviDmm_ReadMultiPoint`
Low Level Measurement…	
Initiate	`IviDmm_Initiate`
Send Software Trigger	`IviDmm_SendSoftwareTrigger`
Fetch	`IviDmm_Fetch`
Fetch Multi-Point	`IviDmm_FetchMultiPoint`

Name or Class	Function Name
Abort	IviDmm_Abort
Is OverRange	IviDmm_IsOverRange

22.2.1 Ivi Dmm Obsolete Function Names

The following function names are reserved by the IviDmm specification 1.0. The future versions of this specification cannot use these names:

- IviDmm_Configure
- IviDmm_CalculateAccuracy
- IviDmm_SendSWTrigger

22.3 IviDmm C Attribute Hierarchy

The IviDmm class attribute hierarchy is shown in the following table.

Table 22-3 IviDmm C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
Basic Operation	
Function	IVIDMM_ATTR_FUNCTION
Range	IVIDMM_ATTR_RANGE
Resolution	IVIDMM_ATTR_RESOLUTION_ABSOLUTE
Trigger	
Trigger Source	IVIDMM_ATTR_TRIGGER_SOURCE
Trigger Delay	IVIDMM_ATTR_TRIGGER_DELAY
Trigger Slope	IVIDMM_ATTR_TRIGGER_SLOPE
AC Measurements	
AC Minimum Frequency	IVIDMM_ATTR_AC_MIN_FREQ
AC Maximum Frequency	IVIDMM_ATTR_AC_MAX_FREQ
Frequency Measurements	
Frequency Voltage Range	IVIDMM_ATTR_FREQ_VOLTAGE_RANGE
Temperature Measurements	
Transducer Type	IVIDMM_ATTR_TEMP_TRANSDUCER_TYPE
Thermocouple	
Fixed Reference Junction	IVIDMM_ATTR_TEMP_TC_FIXED_REF_JUNC
Reference Junction Type	IVIDMM_ATTR_TEMP_TC_REF_JUNC_TYPE
Thermocouple Type	IVIDMM_ATTR_TEMP_TC_TYPE
Resistance Temperature Device	
RTD Resistance	IVIDMM_ATTR_TEMP_RTD_RES
RTD Alpha	IVIDMM_ATTR_TEMP_RTD_ALPHA
Thermistor	
Thermistor Resistance	IVIDMM_ATTR_TEMP_THERMISTOR_RES
Multi-Point Acquisition	
Trigger Count	IVIDMM_ATTR_TRIGGER_COUNT
Sample Count	IVIDMM_ATTR_SAMPLE_COUNT
Sample Trigger	IVIDMM_ATTR_SAMPLE_TRIGGER
Sample Interval	IVIDMM_ATTR_SAMPLE_INTERVAL
Meas Complete Destination	IVIDMM_ATTR_MEAS_COMPLETE_DEST
Configuration Information	

Table 22-3 IviDmm C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
Aperture Time	IVIDMM_ATTR_APERTURE_TIME
Aperture Time Units	IVIDMM_ATTR_APERTURE_TIME_UNITS
Auto Range Value	IVIDMM_ATTR_AUTO_RANGE_VALUE
Measurement Operation Options	
Auto Zero	IVIDMM_ATTR_AUTO_ZERO
Powerline Frequency	IVIDMM_ATTR_POWERLINE_FREQ

Appendix A Specific Driver Development Guidelines

A.1 Introduction

This section describes situations driver developers should be aware of when developing a specific instrument driver that complies with the IviDmm class.

A.2 Disabling Unused Extension Groups

Specific drivers are required to disable extension capability groups that an application program does not explicitly use. The specific driver can do so by setting the attributes of an extension capability group to the values that this section recommends. A specific driver can set these values for all extension capability groups when the *Prefix*_init, *Prefix*_InitWithOptions, or *Prefix*_reset functions execute. This assumes that the extension capability groups remain disabled until the application program explicitly uses them. For the large majority of instruments, this assumption is true.

Under certain conditions, a specific driver might have to implement a more complex approach. For some instruments, configuring a capability group might affect instrument settings that correspond to an unused extension capability group. If these instrument settings affect the behavior of the instrument, then this might result in an interchangeability problem. If this can occur, the specific driver must take appropriate action so that the instrument settings that correspond to the unused extension capability group do not affect the behavior of the instrument when the application program performs an operation that might be affected by those settings.

The remainder of this section recommends attribute values that effectively disable each extension capability group.

Disabling the IviDmm Measurement Extension Capability Groups

Some measurements that the user selects with the Function require an extension group to further configure the measurement. The values for the Function that require additional extension capability groups are shown in the following table.

- IviDmmACMeasurement
- IviDmmFrequencyMeasurement
- IviDmmTemperatureMeasurement
- IviDmmThermocouple
- IviDmmResistanceTemperatureDevice
- IviDmmThermistor

When the Function is set to one of these values, the corresponding extension capability group affects the behavior of the instruments. Otherwise, the extension capability group does not affect the behavior of the instrument and is effectively disabled. Therefore, this section does not recommend how to disable these extension capability groups.

Disabling the IviDmmMultiPoint Extension Group

Attribute values that effectively disable the IviDmmMultiPoint extension group are shown in the following table.

Table A-1. Values for Disabling the IviDmmMultiPoint Extension Group

Attribute	Value
Sample Count	1
Trigger Count	1

Disabling the IviDmmAutoZero Extension Group

Attribute values that effectively disable the IviDmmAutoZero extension group are shown in the following table.

Table A-2. Values for Disabling the IviDmmAutoZero Extension Group

Attribute	Value
Auto Zero	`IVIDMM_VAL_AUTO_ZERO_OFF`

Disabling the IviDmmTriggerSlope Extension Group

The purpose of disabling an extension capability group is to make instrument drivers that implement the capability group behave like instrument drivers that do not implement the capability group in cases were it is not used by the application program. The IviDmmTriggerSlope extension group affects the behavior of the instrument regardless of the value of the Trigger Slope attribute. Therefore, this section does not define any values that can disable the IviDmmTriggerSlope extension group.

Refer to Special Notes for Users in Section 12.1, IviDmmTriggerSlope Extension group Overview for further details.

Disabling the IviDmmPowerLineFrequency Extension Group

The purpose of disabling an extension capability group is to make instrument drivers that implement the capability group behave like instrument drivers that do not implement the capability group in cases were it is not used by the application program. The IviDmmPowerLineFrequency extension group affects the behavior of the instrument regardless of the value of the Power Line Freq attribute. Therefore, this section does not define any values that can disable the IviDmmPowerLineFrequency extension group.

Refer to Special Notes for Users in Section 17.1, IviDmmPowerLineFrequency Extension group Overview for further details.

A.3 Special Consideration for Query Instrument Status

Based on the value of Query Instr Status, the instrument may be queried by the specific driver to determine if it has encountered an error. In specific driver functions, the status check should not occur in the lowest-level signal generation functions `Prefix_Initiate`, `Prefix_Abort`, and `Prefix_Fetch`, `Prefix_FetchMultiPoint`, and `Prefix_SendSWTrigger`. These functions are intended to give the application developer low-level control over signal generation. When calling these functions, the application developer is responsible for checking the status of the instrument. Checking status in every function at this level would also add unnecessary overhead to the specific instrument driver.

A.4 Special Considerations for Sample Trigger

Some of the simpler DMMs on the market implement in hardware a simplified version of the IviDmmMultiPoint state model. These DMMs still have the ability to specify Trigger Count and Sample Count. However, they do not implement Sample Trigger and Sample Interval. When Sample Count is

greater than 1, these DMMs typically execute the trigger delay for each sample. Therefore, the behavior between simple and sophisticated DMMs can vary greatly when performing multipoint scanning.

If you implement the IviDmmMultiPoint extension group on instruments that do not have a Sample Trigger, you should do the following to be interchangeable with DMMs that fully support the extension:

1. Implement Sample Trigger with the only supported value Sample Interval.
2. Implement Sample Interval where the only possible value is the present value for Trigger Delay.
3. Set the Trigger Delay attribute to invalidate the Sample Interval attribute.

By following these guidelines, you will maximize interchangeable behavior between all DMMs.

A.5 Special Considerations for Auto Range Value

The purpose of the attribute Auto Range Value is to return the range that the instrument has auto-ranged to when the attribute Range is set to Auto Range On. Since the value of Auto Range is likely to change as the input signal changes, drivers that may cache attributes should never cache this attribute.

Appendix B Interchangeability Checking Rules

B.1 Introduction

IVI drivers have a feature called interchangeability checking. Interchangeability checking returns a warning when it encounters a situation were the application program might not produce the same behavior when the user attempts to use a different instrument.

B.2 When to Perform Interchangeability Checking

Interchangeability checking occurs when all of the following conditions are met:

? The Interchange Check attribute is set to True

? The user calls one of the following functions:

  ? Initiate

  ? Read

  ? Read Multi Point

B.3 Interchangeability Checking Rules

Interchangeability checking is performed on a capability group basis. When enabled, interchangeability checking is always performed on the base capability group. In addition, interchangeability checking is performed on extension capability groups for which the user has ever set any of the attributes of the group. If the user has never set any attributes of an extension capability group, interchangeability checking is not performed on that group.

In general interchangeability warnings are generated if the following conditions are encountered:

? An attribute that affects the behavior of the instrument is not in a state that the user specifies.

? The user sets a class driver defined attribute to an instrument-specific value.

? The user configures the value of an attribute that the class defines as read-only. In a few cases the class drivers define read-only attributes that specific drivers might implement as read/write.

The remainder of this section defines additional rules and exceptions for each capability group.

IviDmmBase Capability Group

If the Function attribute is set to Temperature, the Resolution Absolute attribute is not required to be in a user specified state.

IviDmmACMeasurement Extension Group

If the Function attribute is not set to AC Volts, AC Current, AC Plus DC Volts, or AC Plus DC Current, then the following attributes are not required to be in a user specified state:

? AC Min Freq

? AC Max Freq

IviDmmFrequencyMeasurement Extension Group

If the Function attribute is not set to Frequency or Period, then the Freq Voltage Range attribute is not required to be in a user specified state.

IviDmmTemperatureMeasurement Extension Group

If the Function attribute is not set to Temperature, the Temperature Transducer Type attribute is not required to be in a user specified state.

IviDmmThermocouple Extension Group

If the Temperature Transducer Type attribute is not set to Thermocouple, then the following attributes are not required to be in a user specified state:

? Thermocouple Type

? Thermocouple Reference Junction Type

? Thermocouple Fixed Reference Junction

IviDmmResistanceTemperatureDevice Extension Group

If the Temperature Transducer Type attribute is not set to 2 Wire RTD or 4 Wire RTD, then the following attributes are not required to be in a user specified state:

? RTD Alpha

? RTD Resistance

IviDmmThermistor Extension Group

If the Temperature Transducer Type attribute is not set to Thermistor, the Thermistor Resistance attribute is not required to be in a user specified state.

IviDmmMultiPoint Extension Group

1. If the Sample Count attribute is set to 1, then the following attributes are not required to be in a user specified state:

   ? Sample Trigger

   ? Sample Interval

2. If the Sample Count attribute is set 1 and the Sample Trigger attribute is set to a value other than Interval, then the Sample Interval attribute is not required to be in a user specified state.

IviDmmTriggerSlope Extension Group

No additional interchangeability rules or exceptions are defined for the IviDmmTriggerSlope extension group.

IviDmmSoftwareTrigger Extension Group

No additional interchangeability rules or exceptions are defined for the IviDmmSoftwareTrigger extension group.

IviDmmDeviceInfo Extension Group

No additional interchangeability rules or exceptions are defined for the IviDmmDeviceInfo extension group.

IviDmmAutoRangeValue Extension Group

No additional interchangeability rules or exceptions are defined for the IviDmmAutoRangeValue extension group.

IviDmmAutoZero Extension Group

No additional interchangeability rules or exceptions are defined for the IviDmmAutoZero extension group.

IviDmmPowerLineFrequency Extension Group

No additional interchangeability rules or exceptions are defined for the IviDmmPowerLineFrequency extension group.

Appendix C ANSI C Include File

The C source code below provides an example of how a class driver C interface might be defined. It provides definitions only for attributes, functions, values, and status codes that this specification defines. It does not represent a complete interface for an IviDmm compliant driver. To aid in the creation of an IviDmm-compliant specific driver, replace `IVIDMM` with the actual driver prefix using uppercase characters and replace `IviDmm` with consistent case sensitivity.

```
/****************************************************************************
 *                              I V I - D M M
 *
 * Title:       IviDmm include file
 * Purpose:     IviDmm Class declarations for Base and Extended Capabilities.
 *
****************************************************************************/


#ifndef IVIDMM_HEADER
#define IVIDMM_HEADER

#if defined(__cplusplus) || defined(__cplusplus__)
extern "C" {
#endif

/****************************************************************************
 *---------------------- IviDmm Class Attribute Defines ----------------------*
 ****************************************************************************/

    /*- IviDmmBase Attributes -*/
#define IVIDMM_ATTR_FUNCTION                (IVI_CLASS_ATTR_BASE + 1)
#define IVIDMM_ATTR_RANGE                   (IVI_CLASS_ATTR_BASE + 2)
#define IVIDMM_ATTR_RESOLUTION_ABSOLUTE     (IVI_CLASS_ATTR_BASE + 8)
#define IVIDMM_ATTR_TRIGGER_SOURCE          (IVI_CLASS_ATTR_BASE + 4)
#define IVIDMM_ATTR_TRIGGER_DELAY           (IVI_CLASS_ATTR_BASE + 5)

    /*- IviDmmAcMeasurement Attributes -*/
#define IVIDMM_ATTR_AC_MIN_FREQ             (IVI_CLASS_ATTR_BASE + 6)
#define IVIDMM_ATTR_AC_MAX_FREQ             (IVI_CLASS_ATTR_BASE + 7)

    /*- IviDmmFrequencyMeasurement Attributes -*/
#define IVIDMM_ATTR_FREQ_VOLTAGE_RANGE      (IVI_CLASS_ATTR_BASE + 101)

    /*- IviDmmTemperatureMeasurement Attributes -*/
#define IVIDMM_ATTR_TEMP_TRANSDUCER_TYPE    (IVI_CLASS_ATTR_BASE + 201)

    /*- IviDmmThermocouple Attributes -*/
#define IVIDMM_ATTR_TEMP_TC_TYPE            (IVI_CLASS_ATTR_BASE + 231)
#define IVIDMM_ATTR_TEMP_TC_REF_JUNC_TYPE   (IVI_CLASS_ATTR_BASE + 232)
#define IVIDMM_ATTR_TEMP_TC_FIXED_REF_JUNC  (IVI_CLASS_ATTR_BASE + 233)

    /*- IviDmmResistanceTemperatureDevice Attributes -*/
#define IVIDMM_ATTR_TEMP_RTD_ALPHA          (IVI_CLASS_ATTR_BASE + 241)
#define IVIDMM_ATTR_TEMP_RTD_RES            (IVI_CLASS_ATTR_BASE + 242)

    /*- IviDmmThermistor Attributes -*/
#define IVIDMM_ATTR_TEMP_THERMISTOR_RES     (IVI_CLASS_ATTR_BASE + 251)

    /*- IviDmmMultiPoint Attributes -*/
#define IVIDMM_ATTR_SAMPLE_COUNT            (IVI_CLASS_ATTR_BASE + 301)
#define IVIDMM_ATTR_SAMPLE_TRIGGER          (IVI_CLASS_ATTR_BASE + 302)
#define IVIDMM_ATTR_SAMPLE_INTERVAL         (IVI_CLASS_ATTR_BASE + 303)
#define IVIDMM_ATTR_TRIGGER_COUNT           (IVI_CLASS_ATTR_BASE + 304)
#define IVIDMM_ATTR_MEAS_COMPLETE_DEST      (IVI_CLASS_ATTR_BASE + 305)

    /*- IviDmmTriggerSlope Attributes -*/
#define IVIDMM_ATTR_TRIGGER_SLOPE           (IVI_CLASS_ATTR_BASE + 334)
```

```
    /*- IviDmmDeviceInfo Attributes -*/
#define IVIDMM_ATTR_APERTURE_TIME             (IVI_CLASS_ATTR_BASE + 321)
#define IVIDMM_ATTR_APERTURE_TIME_UNITS       (IVI_CLASS_ATTR_BASE + 322)

    /*- IviDmmAutoRangeValue Attributes -*/
#define IVIDMM_ATTR_AUTO_RANGE_VALUE          (IVI_CLASS_ATTR_BASE + 331)

    /*- IviDmmAutoZero Attributes -*/
#define IVIDMM_ATTR_AUTO_ZERO                 (IVI_CLASS_ATTR_BASE + 332)

    /*- IviDmmPowerLineFrequency Attributes -*/
#define IVIDMM_ATTR_POWERLINE_FREQ            (IVI_CLASS_ATTR_BASE + 333)
/*****************************************************************************
 *------- IviDmm Class Function Parameter and Attribute Value Defines --------*
 *****************************************************************************/

    /*- Defined values for attribute IVIDMM_ATTR_FUNCTION -*/
#define IVIDMM_VAL_DC_VOLTS                    (1)
#define IVIDMM_VAL_AC_VOLTS                    (2)
#define IVIDMM_VAL_DC_CURRENT                  (3)
#define IVIDMM_VAL_AC_CURRENT                  (4)
#define IVIDMM_VAL_2_WIRE_RES                  (5)
#define IVIDMM_VAL_4_WIRE_RES                  (101)
#define IVIDMM_VAL_AC_PLUS_DC_VOLTS            (106)
#define IVIDMM_VAL_AC_PLUS_DC_CURRENT          (107)
#define IVIDMM_VAL_FREQ                        (104)
#define IVIDMM_VAL_PERIOD                      (105)
#define IVIDMM_VAL_TEMPERATURE                 (108)

#define IVIDMM_VAL_FUNC_CLASS_EXT_BASE         (500)
#define IVIDMM_VAL_FUNC_SPECIFIC_EXT_BASE      (1000)

    /*- Defined values for attribute IVIDMM_ATTR_RANGE -*/
#define IVIDMM_VAL_AUTO_RANGE_ON               (-1.0)
#define IVIDMM_VAL_AUTO_RANGE_OFF              (-2.0)
#define IVIDMM_VAL_AUTO_RANGE_ONCE             (-3.0)

#define IVIDMM_VAL_RANGE_CLASS_EXT_BASE        (-100.0)
#define IVIDMM_VAL_RANGE_SPECIFIC_EXT_BASE     (-1000.0)

    /*- Defined values for attribute IVIDMM_ATTR_FREQ_VOLTAGE_RANGE -*/
/* #define IVIDMM_VAL_AUTO_RANGE_ON               DEFINED ABOVE */
/* #define IVIDMM_VAL_AUTO_RANGE_OFF              DEFINED ABOVE */

#define IVIDMM_VAL_FREQ_VOLT_RANGE_CLASS_EXT_BASE    (-100.0)
#define IVIDMM_VAL_FREQ_VOLT_RANGE_SPECIFIC_EXT_BASE (-1000.0)

    /*- Defined values for attribute IVIDMM_ATTR_TRIGGER_DELAY -*/
#define IVIDMM_VAL_AUTO_DELAY_ON                   (-1.0)
#define IVIDMM_VAL_AUTO_DELAY_OFF                  (-2.0)

#define IVIDMM_VAL_TRIGGER_DELAY_CLASS_EXT_BASE    (-100.0)
#define IVIDMM_VAL_TRIGGER_DELAY_SPECIFIC_EXT_BASE (-1000.0)

    /*- Defined values for attribute IVIDMM_ATTR_TRIGGER_SOURCE -*/
#define IVIDMM_VAL_IMMEDIATE                       (1)
#define IVIDMM_VAL_EXTERNAL                        (2)
#define IVIDMM_VAL_SOFTWARE_TRIG                   (3)
#define IVIDMM_VAL_TTL0                            (111)
#define IVIDMM_VAL_TTL1                            (112)
#define IVIDMM_VAL_TTL2                            (113)
#define IVIDMM_VAL_TTL3                            (114)
#define IVIDMM_VAL_TTL4                            (115)
#define IVIDMM_VAL_TTL5                            (116)
#define IVIDMM_VAL_TTL6                            (117)
#define IVIDMM_VAL_TTL7                            (118)
#define IVIDMM_VAL_ECL0                            (119)
#define IVIDMM_VAL_ECL1                            (120)
#define IVIDMM_VAL_PXI_STAR                        (131)
#define IVIDMM_VAL_RTSI_0                          (140)
#define IVIDMM_VAL_RTSI_1                          (141)
```

```
#define IVIDMM_VAL_RTSI_2                           (142)
#define IVIDMM_VAL_RTSI_3                           (143)
#define IVIDMM_VAL_RTSI_4                           (144)
#define IVIDMM_VAL_RTSI_5                           (145)
#define IVIDMM_VAL_RTSI_6                           (146)

#define IVIDMM_VAL_TRIGGER_SOURCE_CLASS_EXT_BASE    (500)
#define IVIDMM_VAL_TRIGGER_SOURCE_SPECIFIC_EXT_BASE (1000)

    /*- Defined values for attribute IVIDMM_ATTR_TEMP_TRANSDUCER_TYPE -*/
#define IVIDMM_VAL_THERMOCOUPLE                     (1)
#define IVIDMM_VAL_THERMISTOR                       (2)
#define IVIDMM_VAL_2_WIRE_RTD                       (3)
#define IVIDMM_VAL_4_WIRE_RTD                       (4)

#define IVIDMM_VAL_TRANSDUCER_CLASS_EXT_BASE        (100)
#define IVIDMM_VAL_TRANSDUCER_SPECIFIC_EXT_BASE     (1000)

    /*- Defined values for attribute IVIDMM_ATTR_TEMP_TC_REF_JUNC_TYPE -*/
#define IVIDMM_VAL_TEMP_REF_JUNC_INTERNAL           (1)
#define IVIDMM_VAL_TEMP_REF_JUNC_FIXED              (2)
#define IVIDMM_VAL_TEMP_REF_JUNC_CLASS_EXT_BASE     (100)
#define IVIDMM_VAL_TEMP_REF_JUNC_SPECIFIC_EXT_BASE  (1000)

    /*- Defined values for attribute IVIDMM_ATTR_TEMP_TC_TYPE -*/
#define IVIDMM_VAL_TEMP_TC_B                        (1)
#define IVIDMM_VAL_TEMP_TC_C                        (2)
#define IVIDMM_VAL_TEMP_TC_D                        (3)
#define IVIDMM_VAL_TEMP_TC_E                        (4)
#define IVIDMM_VAL_TEMP_TC_G                        (5)
#define IVIDMM_VAL_TEMP_TC_J                        (6)
#define IVIDMM_VAL_TEMP_TC_K                        (7)
#define IVIDMM_VAL_TEMP_TC_N                        (8)
#define IVIDMM_VAL_TEMP_TC_R                        (9)
#define IVIDMM_VAL_TEMP_TC_S                        (10)
#define IVIDMM_VAL_TEMP_TC_T                        (11)
#define IVIDMM_VAL_TEMP_TC_U                        (12)
#define IVIDMM_VAL_TEMP_TC_V                        (13)
#define IVIDMM_VAL_TEMP_TC_TYPE_CLASS_EXT_BASE      (100)
#define IVIDMM_VAL_TEMP_TC_TYPE_SPECIFIC_EXT_BASE   (1000)

    /*- Defined values for attribute IVIDMM_ATTR_MEAS_COMPLETE_DEST -*/
#define IVIDMM_VAL_NONE                             (-1)
/* #define IVIDMM_VAL_EXTERNAL                      DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL0                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL1                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL2                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL3                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL4                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL5                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL6                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL7                          DEFINED ABOVE */
/* #define IVIDMM_VAL_ECL0                          DEFINED ABOVE */
/* #define IVIDMM_VAL_ECL1                          DEFINED ABOVE */
/* #define IVIDMM_VAL_PXI_STAR                      DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_0                        DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_1                        DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_2                        DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_3                        DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_4                        DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_5                        DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_6                        DEFINED ABOVE */

    /* Defined values for attribute IVIDMM_ATTR_SAMPLE_TRIGGER -*/
/* #define IVIDMM_VAL_IMMEDIATE                     DEFINED ABOVE */
/* #define IVIDMM_VAL_EXTERNAL                      DEFINED ABOVE */
/* #define IVIDMM_VAL_SOFTWARE_TRIG                 DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL0                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL1                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL2                          DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL3                          DEFINED ABOVE */
```

```
/* #define IVIDMM_VAL_TTL4                    DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL5                    DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL6                    DEFINED ABOVE */
/* #define IVIDMM_VAL_TTL7                    DEFINED ABOVE */
/* #define IVIDMM_VAL_ECL0                    DEFINED ABOVE */
/* #define IVIDMM_VAL_ECL1                    DEFINED ABOVE */
/* #define IVIDMM_VAL_PXI_STAR                DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_0                  DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_1                  DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_2                  DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_3                  DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_4                  DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_5                  DEFINED ABOVE */
/* #define IVIDMM_VAL_RTSI_6                  DEFINED ABOVE */
#define IVIDMM_VAL_INTERVAL                   (10)

    /*- Defined values for attribute IVIDMM_ATTR_TRIGGER_SLOPE -*/
#define IVIDMM_VAL_POSITIVE                       (0)
#define IVIDMM_VAL_NEGATIVE                       (1)

#define IVIDMM_VAL_TRIGGER_SLOPE_CLASS_EXT_BASE     (100)
#define IVIDMM_VAL_TRIGGER_SLOPE_SPECIFIC_EXT_BASE  (1000)

    /*- Defined values for attribute IVIDMM_ATTR_APERTURE_TIME_UNITS -*/
#define IVIDMM_VAL_SECONDS                    (0)
#define IVIDMM_VAL_POWER_LINE_CYCLES          (1)

    /*- Defined values for extended attribute IVIDMM_ATTR_AUTO_ZER0 -*/
#define IVIDMM_VAL_AUTO_ZERO_OFF              (0)
#define IVIDMM_VAL_AUTO_ZERO_ON               (1)
#define IVIDMM_VAL_AUTO_ZERO_ONCE             (2)

#define IVIDMM_VAL_AUTO_ZERO_CLASS_EXT_BASE     (100)
#define IVIDMM_VAL_AUTO_ZERO_SPECIFIC_EXT_BASE  (1000)

    /*- Defined values for Read, Fetch, Read Multipoint & Fetch Multipoint-*/
#define IVIDMM_VAL_MAX_TIME_INFINITE    0xFFFFFFFF
#define IVIDMM_VAL_MAX_TIME_IMMEDIATE   0x0

/*****************************************************************************
 *----------- IviDmm Class Instrument Driver Function Declarations -----------*
 *****************************************************************************/


   /*- IviDmmBase Capability Group Functions -*/

ViStatus _VI_FUNC IviDmm_ConfigureMeasurement (ViSession vi,
                                               ViInt32 function,
                                               ViReal64 range,
                                               ViReal64 resolution);

ViStatus _VI_FUNC IviDmm_ConfigureTrigger (ViSession vi,
                                           ViInt32 triggerSource,
                                           ViReal64 triggerDelay);

ViStatus _VI_FUNC IviDmm_Read (ViSession vi,
                               ViInt32 maxTimeMilliseconds,
                               ViReal64 *reading);

ViStatus _VI_FUNC IviDmm_Fetch (ViSession vi,
                                ViInt32 maxTimeMilliseconds,
                                ViReal64 *reading);

ViStatus _VI_FUNC IviDmm_Abort (ViSession vi);

ViStatus _VI_FUNC IviDmm_Initiate (ViSession vi);

ViStatus _VI_FUNC IviDmm_IsOverRange (ViSession vi,
                                      ViReal64 measurementValue,
                                      ViBoolean *isOverRange);
```

```
    /*- IviDmmAcMeasurement Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureACBandwidth (ViSession vi,
                                               ViReal64 minFreq,
                                               ViReal64 maxFreq);

    /*- IviDmmFrequencyMeasurement Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureFrequencyVoltageRange (ViSession vi,
                                                         ViReal64 frequencyVoltageRange);

    /*- IviDmmTemperatureMeasurement Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureTransducerType (ViSession vi,
                                                  ViInt32 transducerType);

    /*- IviDmmThermocouple Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureFixedRefJunction (ViSession vi,
                                                    ViReal64 fixedRefJunction);
ViStatus _VI_FUNC IviDmm_ConfigureThermocouple (ViSession vi,
                                                ViInt32 thermocoupleType,
                                                ViInt32 refJunctionType);

    /*- IviDmmRTD Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureRTD (ViSession vi,
                                       ViReal64 alpha,
                                       ViReal64 resistance);

    /*- IviDmmThermistor Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureThermistor (ViSession vi,
                                              ViReal64 resistance);

    /*- IviDmmMultiPoint Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureMeasCompleteDest (ViSession vi,
                                                    ViInt32 measCompleteDest);
ViStatus _VI_FUNC IviDmm_ConfigureMultiPoint (ViSession vi,
                                              ViInt32 triggerCount,
                                              ViInt32 sampleCount,
                                              ViInt32 sampleTrigger,
                                              ViReal64 sampleInterval);
ViStatus _VI_FUNC IviDmm_ReadMultiPoint (ViSession vi,
                                         ViInt32 maxTimeMilliseconds,
                                         ViInt32 arraySize,
                                         ViReal64 readingArray[],
                                         ViInt32 *actualPts);
ViStatus _VI_FUNC IviDmm_FetchMultiPoint (ViSession vi,
                                          ViInt32 maxTimeMilliseconds,
                                          ViInt32 arraySize,
                                          ViReal64 readingArray[],
                                          ViInt32 *actualPts);

    /*- IviDmmTriggerSlope Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureTriggerSlope (ViSession vi,
                                                ViInt32 polarity);

    /*- IviDmmSoftwareTrigger Functions -*/
ViStatus _VI_FUNC IviDmm_SendSoftwareTrigger (ViSession vi);

    /*- IviDmmDeviceInfo Functions -*/
ViStatus _VI_FUNC IviDmm_GetApertureTimeInfo (ViSession vi,
                                              ViReal64 *apertureTime,
                                              ViInt32 *apertureTimeUnits);

    /*- IviDmmAutoRangeValue Functions -*/
ViStatus _VI_FUNC IviDmm_GetAutoRangeValue (ViSession vi,
                                            ViReal64 *autoRangeValue);

    /*- IviDmmAutoZero Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigureAutoZeroMode (ViSession vi,
                                                ViInt32 autoZeroMode);

    /*- IviDmmPowerLineFrequency Functions -*/
ViStatus _VI_FUNC IviDmm_ConfigurePowerLineFrequency (ViSession vi,
                                                      ViReal64 powerLineFreq);
```

```
/****************************************************************************
 *----------------- IviDmm Class Error And Completion Codes ------------------*
 ****************************************************************************/
#define IVIDMM_WARN_OVER_RANGE              (IVI_CLASS_WARN_BASE  + 1)

#define IVIDMM_ERROR_TRIGGER_NOT_SOFTWARE   (IVI_SHARED_COMPONENT_ERROR_BASE + 1)
#define IVIDMM_ERROR_MAX_TIME_EXCEEDED      (IVI_CLASS_ERROR_BASE + 3)
/****************************************************************************
 *---------------------------- End Include File ----------------------------*
 ****************************************************************************/
#if defined(__cplusplus) || defined(__cplusplus__)
}
#endif
#endif /* IVIDMM_HEADER */
```

Appendix D COM IDL File

To ease the development of a compliant an IVI-COM driver for the IviDmm class, the IVI Foundation publishes IDL (Interface Description Language) files that consolidate all the method and property definitions listed in this specification. Notice that the interface IIviDmm derives from IiviDriver. It is described in *IVI-3.2: Inherent Capabilities Specification*.

These files along with these definitions compiled into type libraries are available from the IVI Foundation web site at `http://www.ivifoundation.org/`.

D.1 IviDmmTypeLib.idl

```
#if !defined(IVI_DMM_TYPELIB_IDL_INCLUDED_)
#define IVI_DMM_TYPELIB_IDL_INCLUDED_
/*****************************************************************************
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
 *  All rights reserved.
 *
 *
 * FILENAME     : IviDmmTypeLib.idl
 *
 * STATUS       : APPROVED.
 * COMPILER     : MSVC++ 6.0, sp4 MIDL
 * CONTENT      : IVI DMM Instrument Class Standard IDL
 *                type library definition
 *
 * $Archive: /IVI/IviTypeLibraries/IviDmmTypeLib/IviDmmTypeLib.idl $
 * $Revision: 6 $
 *****************************************************************************/


import "oaidl.idl";
import "ocidl.idl";


[
   uuid(47ed5122-a398-11d4-ba58-000064657374),
   version(3.0),
   helpstring("IviDmm 3.0 Type Library"),
   helpfile("IviDmm.chm")
]
library IviDmmLib
{
   importlib("stdole32.tlb");
   importlib("stdole2.tlb");

#include "IviDmm.idl"
};


/***************************** REVISION  LOG *********************************
 * $Log: /IVI/IviTypeLibraries/IviDmmTypeLib/IviDmmTypeLib.idl $
 *
 * 6     2/19/02 4:40p Jmh00
 * JMH - Final changes prior to vote on the spec
 *
 * 5     11/06/01 8:37a Jmh00
 * JMH - Final IDL check
 *
 * 4     8/14/01 10:16p Jmh00
 * JMH Restructure IVI type libraries workspace
 *
 * 3     8/03/01 11:00a Jmh00
 * JMH Changes for review version of spec
```

```
 ***************************** END OF FILE *********************************/
#endif // !defined(IVI_DMM_TYPELIB_IDL_INCLUDED_)
```

D.2 IviDmm.idl

```
#if !defined(IVI_DMM_IDL_INCLUDED_)
#define IVI_DMM_IDL_INCLUDED_
/*****************************************************************************
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
 *  All rights reserved.
 *
 *
 * FILENAME     : IviDmm.idl
 *
 * STATUS       : APPROVED.
 * COMPILER     : MSVC++ 6.0, sp4 MIDL
 * CONTENT      : IVI DMM Instrument Class Standard IDL
 *
 * $Archive: /IVI/IviTypeLibraries/IviDmmTypeLib/IviDmm.idl $
 * $Revision: 12 $
 *****************************************************************************/


#include <winerror.h>
import "oaidl.idl";
import "ocidl.idl";

#ifdef IVI_USE_IMPORT
import"..\IviDriverTypeLib\IviDriver.idl";
#else
importlib ("..\TypeLibraries\IviDriverTypeLib.dll");
#endif


//---------------------------------------------------------------------------
//   Preprocessor Macros
//---------------------------------------------------------------------------

#define HELP_DMM(x)  helpstring(HS_DMM_ ## x ## ), helpcontext(HC_DMM_ ## x ## )


//---------------------------------------------------------------------------
//   Provides for Localization
//---------------------------------------------------------------------------

#include "IviDmmEnglish.idl"


//---------------------------------------------------------------------------
//   Interface Declarations
//---------------------------------------------------------------------------

interface IIviDmm;
interface IIviDmmMeasurement;
interface IIviDmmTrigger;
interface IIviDmmMultiPoint;
interface IIviDmmAC;
interface IIviDmmAdvanced;
interface IIviDmmFrequency;
interface IIviDmmTemperature;
interface IIviDmmRTD;
interface IIviDmmThermocouple;
interface IIviDmmThermistor;

#define UUID_IIVI_DMM              47ed51e8-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_MEASUREMENT  47ed51e9-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_TRIGGER      47ed51ea-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_MULTIPOINT   47ed51eb-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_AC           47ed51ec-a398-11d4-ba58-000064657374
```

```
#define UUID_IIVI_DMM_ADVANCED     47ed51ed-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_FREQUENCY    47ed51ee-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_TEMPERATURE  47ed51ef-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_RTD          47ed51f0-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_THERMOCOUPLE 47ed51f1-a398-11d4-ba58-000064657374
#define UUID_IIVI_DMM_THERMISTOR   47ed51f2-a398-11d4-ba58-000064657374


//---------------------------------------------------------------------------
//     TYPEDEF ENUMS
//---------------------------------------------------------------------------

[
   HELP_DMM(HRESULTS)
]
typedef enum IviDmmErrorCodesEnum
{
   E_IVIDMM_TRIGGER_NOT_SOFTWARE        = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, 0x1001),
   E_IVIDMM_MAX_TIME_EXCEEDED       = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, 0x2003),
   S_IVIDMM_OVER_RANGE              = MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF, 0x2001)
} IviDmmErrorCodesEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmApertureTimeUnitsEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(APERTURE_TIME_UNITS_ENUM)
]
enum IviDmmApertureTimeUnitsEnum
{
    IviDmmApertureSeconds                   = 0,
    IviDmmAperturePowerLineCycles            = 1
} IviDmmApertureTimeUnitsEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmAutoRangeEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(AUTO_RANGE_ENUM)
]
enum IviDmmAutoRangeEnum
{
    IviDmmAutoRangeOn                       = -1,
    IviDmmAutoRangeOff                      = -2,
    IviDmmAutoRangeOnce                      = -3
} IviDmmAutoRangeEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmAutoZeroEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(AUTO_ZERO_ENUM)
]
enum IviDmmAutoZeroEnum
{
    IviDmmAutoZeroOff                       = 0,
```

```
    IviDmmAutoZeroOn                        = 1,
    IviDmmAutoZeroOnce                      = 2
} IviDmmAutoZeroEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmFrequencyVoltageRangeEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(FREQ_VOLTAGE_RANGE_ENUM)
]
enum IviDmmFrequencyVoltageRangeEnum
{
    IviDmmFrequencyVoltageRangeAutoRangeOn    = -1,
    IviDmmFrequencyVoltageRangeAutoRangeOff   = -2
} IviDmmFrequencyVoltageRangeEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmFunctionEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(FUNCTION_ENUM)
]
enum IviDmmFunctionEnum
{
    IviDmmFunctionDCVolts                   = 1,
    IviDmmFunctionACVolts                   = 2,
    IviDmmFunctionDCCurrent                 = 3,
    IviDmmFunctionACCurrent                 = 4,
    IviDmmFunction2WireRes                  = 5,
    IviDmmFunction4WireRes                  = 101,
    IviDmmFunctionACPlusDCVolts             = 106,
    IviDmmFunctionACPlusDCCurrent           = 107,
    IviDmmFunctionFreq                      = 104,
    IviDmmFunctionPeriod                    = 105,
    IviDmmFunctionTemperature               = 108
} IviDmmFunctionEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmMaxTimeEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(MAX_TIME_ENUM)
]
enum IviDmmMaxTimeEnum
{
   IviDmmMaxTimeInfinite                    = 0xFFFFFFFFUL,
   IviDmmMaxTimeImmediate                   = 0
} IviDmmMaxTimeEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmMeasCompleteDestEnum
//---------------------------------------------------------------------------
typedef
[
   public,
```

```
   v1_enum,
   HELP_DMM(MEAS_COMPLETE_DEST_ENUM)
]
enum IviDmmMeasCompleteDestEnum
{
    IviDmmMeasCompleteDestNone               = -1,
    IviDmmMeasCompleteDestExternal           = 2,
    IviDmmMeasCompleteDestTTL0               = 111,
    IviDmmMeasCompleteDestTTL1               = 112,
    IviDmmMeasCompleteDestTTL2               = 113,
    IviDmmMeasCompleteDestTTL3               = 114,
    IviDmmMeasCompleteDestTTL4               = 115,
    IviDmmMeasCompleteDestTTL5               = 116,
    IviDmmMeasCompleteDestTTL6               = 117,
    IviDmmMeasCompleteDestTTL7               = 118,
    IviDmmMeasCompleteDestECL0               = 119,
    IviDmmMeasCompleteDestECL1               = 120,
    IviDmmMeasCompleteDestPXIStar            = 131,
    IviDmmMeasCompleteDestRTSI0              = 140,
    IviDmmMeasCompleteDestRTSI1              = 141,
    IviDmmMeasCompleteDestRTSI2              = 142,
    IviDmmMeasCompleteDestRTSI3              = 143,
    IviDmmMeasCompleteDestRTSI4              = 144,
    IviDmmMeasCompleteDestRTSI5              = 145,
    IviDmmMeasCompleteDestRTSI6              = 146
} IviDmmMeasCompleteDestEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmRefJunctionTypeEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(REF_JUNCTION_TYPE_ENUM)
]
enum IviDmmRefJunctionTypeEnum
{
    IviDmmRefJunctionTypeInternal            = 1,
    IviDmmRefJunctionTypeFixed               = 2
} IviDmmRefJunctionTypeEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmSampleTriggerEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(SAMPLE_TRIGGER_ENUM)
]
enum IviDmmSampleTriggerEnum
{
    IviDmmSampleTriggerImmediate             = 1,
    IviDmmSampleTriggerExternal              = 2,
    IviDmmSampleTriggerSwTrigFunc            = 3,
    IviDmmSampleTriggerTTL0                  = 111,
    IviDmmSampleTriggerTTL1                  = 112,
    IviDmmSampleTriggerTTL2                  = 113,
    IviDmmSampleTriggerTTL3                  = 114,
    IviDmmSampleTriggerTTL4                  = 115,
    IviDmmSampleTriggerTTL5                  = 116,
    IviDmmSampleTriggerTTL6                  = 117,
    IviDmmSampleTriggerTTL7                  = 118,
    IviDmmSampleTriggerECL0                  = 119,
    IviDmmSampleTriggerECL1                  = 120,
    IviDmmSampleTriggerPXIStar               = 131,
```

```
    IviDmmSampleTriggerRTSI0                = 140,
    IviDmmSampleTriggerRTSI1                = 141,
    IviDmmSampleTriggerRTSI2                = 142,
    IviDmmSampleTriggerRTSI3                = 143,
    IviDmmSampleTriggerRTSI4                = 144,
    IviDmmSampleTriggerRTSI5                = 145,
    IviDmmSampleTriggerRTSI6                = 146,
    IviDmmSampleTriggerInterval             = 10
} IviDmmSampleTriggerEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmThermocoupleTypeEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(THERMOCOUPLE_TYPE_ENUM)
]
enum IviDmmThermocoupleTypeEnum
{
    IviDmmThermocoupleTypeB                 = 1,
    IviDmmThermocoupleTypeC                 = 2,
    IviDmmThermocoupleTypeD                 = 3,
    IviDmmThermocoupleTypeE                 = 4,
    IviDmmThermocoupleTypeG                 = 5,
    IviDmmThermocoupleTypeJ                 = 6,
    IviDmmThermocoupleTypeK                 = 7,
    IviDmmThermocoupleTypeN                 = 8,
    IviDmmThermocoupleTypeR                 = 9,
    IviDmmThermocoupleTypeS                 = 10,
    IviDmmThermocoupleTypeT                 = 11,
    IviDmmThermocoupleTypeU                 = 12,
    IviDmmThermocoupleTypeV                 = 13
} IviDmmThermocoupleTypeEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmTransducerTypeEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(TRANSDUCER_TYPE_ENUM)
]
enum IviDmmTransducerTypeEnum
{
    IviDmmTransducerTypeThermocouple        = 1,
    IviDmmTransducerTypeThermistor          = 2,
    IviDmmTransducerType2WireRtd            = 3,
    IviDmmTransducerType4WireRtd            = 4
} IviDmmTransducerTypeEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmTriggerDelayEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(TRIGGER_DELAY_ENUM)
]
enum IviDmmTriggerDelayEnum
{
    IviDmmTriggerDelayAutoDelayOn           = -1,
```

```
    IviDmmTriggerDelayAutoDelayOff           = -2
} IviDmmTriggerDelayEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmTriggerSlopeEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(TRIGGER_SLOPE_ENUM)
]
enum IviDmmTriggerSlopeEnum
{
    IviDmmTriggerSlopePositive               = 0,
    IviDmmTriggerSlopeNegative               = 1
} IviDmmTriggerSlopeEnum;


//---------------------------------------------------------------------------
//   enum: IviDmmTriggerSourceEnum
//---------------------------------------------------------------------------

typedef
[
   public,
   v1_enum,
   HELP_DMM(TRIGGER_SOURCE_ENUM)
]
enum IviDmmTriggerSourceEnum
{
    IviDmmTriggerSourceImmediate             = 1,
    IviDmmTriggerSourceExternal              = 2,
    IviDmmTriggerSourceSwTrigFunc            = 3,
    IviDmmTriggerSourceTTL0                  = 111,
    IviDmmTriggerSourceTTL1                  = 112,
    IviDmmTriggerSourceTTL2                  = 113,
    IviDmmTriggerSourceTTL3                  = 114,
    IviDmmTriggerSourceTTL4                  = 115,
    IviDmmTriggerSourceTTL5                  = 116,
    IviDmmTriggerSourceTTL6                  = 117,
    IviDmmTriggerSourceTTL7                  = 118,
    IviDmmTriggerSourceECL0                  = 119,
    IviDmmTriggerSourceECL1                  = 120,
    IviDmmTriggerSourcePXIStar               = 131,
    IviDmmTriggerSourceRTSI0                 = 140,
    IviDmmTriggerSourceRTSI1                 = 141,
    IviDmmTriggerSourceRTSI2                 = 142,
    IviDmmTriggerSourceRTSI3                 = 143,
    IviDmmTriggerSourceRTSI4                 = 144,
    IviDmmTriggerSourceRTSI5                 = 145,
    IviDmmTriggerSourceRTSI6                 = 146
} IviDmmTriggerSourceEnum;


//---------------------------------------------------------------------------
//   IVI Dmm Driver Root Level Interface
//---------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM),
   HELP_DMM(I_IVI_DMM),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmm : IIviDriver
{
//----------------------------------------- Configure
```

```
   [HELP_DMM(CONFIGURE)]
   HRESULT Configure(
         [in] IviDmmFunctionEnum Function,
         [in] DOUBLE Range,
         [in] DOUBLE Resolution);


//---------------------------------------- Function
   [propput, HELP_DMM(FUNCTION)]
   HRESULT Function([in] IviDmmFunctionEnum newVal);

   [propget, HELP_DMM(FUNCTION)]
   HRESULT Function([out, retval] IviDmmFunctionEnum* pVal);


//---------------------------------------- Range
   [propput, HELP_DMM(RANGE)]
   HRESULT Range([in] DOUBLE newVal);

   [propget, HELP_DMM(RANGE)]
   HRESULT Range([out, retval] DOUBLE* pVal);


//---------------------------------------- Resolution
   [propput, HELP_DMM(RESOLUTION)]
   HRESULT Resolution([in] DOUBLE newVal);

   [propget, HELP_DMM(RESOLUTION)]
   HRESULT Resolution([out, retval] DOUBLE* pVal);


//---------------------------------------- AC Interface Reference
   [ propget, HELP_DMM(AC) ]
   HRESULT AC ([out, retval] IIviDmmAC **pVal);


//---------------------------------------- Advanced Interface Reference
   [ propget, HELP_DMM(ADVANCED) ]
   HRESULT Advanced ([out, retval] IIviDmmAdvanced **pVal);


//---------------------------------------- Frequency Interface Reference
   [ propget, HELP_DMM(FREQUENCY) ]
   HRESULT Frequency ([out, retval] IIviDmmFrequency **pVal);


//---------------------------------------- Measurement Interface Reference
   [ propget, HELP_DMM(MEASUREMENT) ]
   HRESULT Measurement ([out, retval] IIviDmmMeasurement **pVal);


//---------------------------------------- Temperature Interface Reference
   [ propget, HELP_DMM(TEMPERATURE) ]
   HRESULT Temperature ([out, retval] IIviDmmTemperature **pVal);


//---------------------------------------- Trigger Interface Reference
   [ propget, HELP_DMM(TRIGGER) ]
   HRESULT Trigger ([out, retval] IIviDmmTrigger **pVal);

};


//-----------------------------------------------------------------------------
//   Measurement Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM_MEASUREMENT),
   HELP_DMM(I_IVI_DMM_MEASUREMENT),
```

```
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmMeasurement : IUnknown
{
//---------------------------------------- Initiate
   [HELP_DMM(INITIATE)]
   HRESULT Initiate();


//---------------------------------------- Abort
   [HELP_DMM(ABORT)]
   HRESULT Abort();


//---------------------------------------- Fetch
   [HELP_DMM(FETCH)]
   HRESULT Fetch(
         [in] LONG MaxTimeMilliseconds,
         [out, retval] DOUBLE *Reading);


//---------------------------------------- FetchMultiPoint
   [HELP_DMM(FETCH_MULTI_POINT)]
   HRESULT FetchMultiPoint(
         [in] LONG MaxTimeMilliseconds,
         [out, retval] SAFEARRAY (DOUBLE) *ReadingArray);


//---------------------------------------- Read
   [HELP_DMM(READ)]
   HRESULT Read(
         [in] LONG MaxTimeMilliseconds,
         [out, retval] DOUBLE *Reading);


//---------------------------------------- ReadMultiPoint
   [HELP_DMM(READ_MULTI_POINT)]
   HRESULT ReadMultiPoint(
         [in] LONG MaxTimeMilliseconds,
         [out,retval] SAFEARRAY (DOUBLE) *ReadingArray);


//---------------------------------------- SendSoftwareTrigger
   [HELP_DMM(SEND_SOFTWARE_TRIGGER)]
   HRESULT SendSoftwareTrigger();


//---------------------------------------- IsOverRange
   [HELP_DMM(IS_OVER_RANGE)]
   HRESULT IsOverRange(
         [in] DOUBLE MeasurementValue,
         [out, retval] VARIANT_BOOL* IsOverRange);

};


//-----------------------------------------------------------------------------
//   Trigger Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM_TRIGGER),
   HELP_DMM(I_IVI_DMM_TRIGGER),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmTrigger : IUnknown
{
//---------------------------------------- Configure
```

```
   [HELP_DMM(CONFIGURE_TRIGGER)]
   HRESULT Configure(
         [in] IviDmmTriggerSourceEnum TriggerSource,
         [in] DOUBLE TriggerDelay);


//---------------------------------------- Slope
   [propput, HELP_DMM(SLOPE)]
   HRESULT Slope([in] IviDmmTriggerSlopeEnum newVal);

   [propget, HELP_DMM(SLOPE)]
   HRESULT Slope([out, retval] IviDmmTriggerSlopeEnum* pVal);


//---------------------------------------- Delay
   [propput, HELP_DMM(DELAY)]
   HRESULT Delay([in] DOUBLE newVal);

   [propget, HELP_DMM(DELAY)]
   HRESULT Delay([out, retval] DOUBLE* pVal);


//---------------------------------------- Source
   [propput, HELP_DMM(SOURCE)]
   HRESULT Source([in] IviDmmTriggerSourceEnum newVal);

   [propget, HELP_DMM(SOURCE)]
   HRESULT Source([out, retval] IviDmmTriggerSourceEnum* pVal);


//---------------------------------------- MultiPoint Interface Reference
   [ propget, HELP_DMM(MULTIPOINT) ]
   HRESULT MultiPoint ([out, retval] IIviDmmMultiPoint **pVal);

};


//------------------------------------------------------------------------------
//   MultiPoint Interface
//------------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM_MULTIPOINT),
   HELP_DMM(I_IVI_DMM_MULTIPOINT),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmMultiPoint : IUnknown
{
//---------------------------------------- Configure
   [HELP_DMM(CONFIGURE_MULTI_POINT)]
   HRESULT Configure(
         [in] LONG TriggerCount,
         [in] LONG SampleCount,
         [in] IviDmmSampleTriggerEnum SampleTrigger,
         [in] DOUBLE SampleInterval);


//---------------------------------------- MeasurementComplete
   [propput, HELP_DMM(MEASUREMENT_COMPLETE)]
   HRESULT MeasurementComplete([in] IviDmmMeasCompleteDestEnum newVal);

   [propget, HELP_DMM(MEASUREMENT_COMPLETE)]
   HRESULT MeasurementComplete([out, retval] IviDmmMeasCompleteDestEnum* pVal);


//---------------------------------------- Count
   [propput, HELP_DMM(COUNT)]
   HRESULT Count([in] LONG newVal);
```

```
   [propget, HELP_DMM(COUNT)]
   HRESULT Count([out, retval] LONG* pVal);


//---------------------------------------- SampleCount
   [propput, HELP_DMM(SAMPLE_COUNT)]
   HRESULT SampleCount([in] LONG newVal);

   [propget, HELP_DMM(SAMPLE_COUNT)]
   HRESULT SampleCount([out, retval] LONG* pVal);


//---------------------------------------- SampleInterval
   [propput, HELP_DMM(SAMPLE_INTERVAL)]
   HRESULT SampleInterval([in] DOUBLE newVal);

   [propget, HELP_DMM(SAMPLE_INTERVAL)]
   HRESULT SampleInterval([out, retval] DOUBLE* pVal);


//---------------------------------------- SampleTrigger
   [propput, HELP_DMM(SAMPLE_TRIGGER)]
   HRESULT SampleTrigger([in] IviDmmSampleTriggerEnum newVal);

   [propget, HELP_DMM(SAMPLE_TRIGGER)]
   HRESULT SampleTrigger([out, retval] IviDmmSampleTriggerEnum* pVal);

};


//------------------------------------------------------------------------------
//   AC Interface
//------------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM_AC),
   HELP_DMM(I_IVI_DMM_AC),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmAC : IUnknown
{
//---------------------------------------- ConfigureBandwidth
   [HELP_DMM(CONFIGURE_AC_BANDWIDTH)]
   HRESULT ConfigureBandwidth(
         [in] DOUBLE MinFreq,
         [in] DOUBLE MaxFreq);


//---------------------------------------- FrequencyMax
   [propput, HELP_DMM(AC_FREQUENCY_MAX)]
   HRESULT FrequencyMax([in] DOUBLE newVal);

   [propget, HELP_DMM(AC_FREQUENCY_MAX)]
   HRESULT FrequencyMax([out, retval] DOUBLE *pVal);


//---------------------------------------- FrequencyMin
   [propput, HELP_DMM(AC_FREQUENCY_MIN)]
   HRESULT FrequencyMin([in] DOUBLE newVal);

   [propget, HELP_DMM(AC_FREQUENCY_MIN)]
   HRESULT FrequencyMin([out, retval] DOUBLE *pVal);

};


//------------------------------------------------------------------------------
//   Advanced Interface
//------------------------------------------------------------------------------
```

```
[
   object,
   uuid(UUID_IIVI_DMM_ADVANCED),
   HELP_DMM(I_IVI_DMM_ADVANCED),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmAdvanced : IUnknown
{
//----------------------------------------- ActualRange
   [propget, HELP_DMM(ACTUAL_RANGE)]
   HRESULT ActualRange([out, retval] DOUBLE *pVal);


//----------------------------------------- ApertureTime
   [propget, HELP_DMM(APERTURE_TIME)]
   HRESULT ApertureTime([out, retval] DOUBLE *pVal);


//----------------------------------------- ApertureTimeUnits
   [propget, HELP_DMM(APERTURE_TIME_UNITS)]
   HRESULT ApertureTimeUnits([out, retval] IviDmmApertureTimeUnitsEnum* pVal);


//----------------------------------------- AutoZero
   [propput, HELP_DMM(AUTO_ZERO)]
   HRESULT AutoZero([in] IviDmmAutoZeroEnum newVal);

   [propget, HELP_DMM(AUTO_ZERO)]
   HRESULT AutoZero([out, retval] IviDmmAutoZeroEnum* pVal);


//----------------------------------------- PowerlineFrequency
   [propput, HELP_DMM(POWERLINE_FREQUENCY)]
   HRESULT PowerlineFrequency([in] DOUBLE newVal);

   [propget, HELP_DMM(POWERLINE_FREQUENCY)]
   HRESULT PowerlineFrequency([out, retval] DOUBLE* pVal);

};


//-----------------------------------------------------------------------------
//   Frequency Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM_FREQUENCY),
   HELP_DMM(I_IVI_DMM_FREQUENCY),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmFrequency : IUnknown
{
//----------------------------------------- VoltageRange
   [propput, HELP_DMM(FREQUENCY_VOLTAGE_RANGE)]
   HRESULT VoltageRange([in] DOUBLE newVal);

   [propget, HELP_DMM(FREQUENCY_VOLTAGE_RANGE)]
   HRESULT VoltageRange([out, retval] DOUBLE* pVal);

};


//-----------------------------------------------------------------------------
//   Temperature Interface
//-----------------------------------------------------------------------------

[
```

```
   object,
   uuid(UUID_IIVI_DMM_TEMPERATURE),
   HELP_DMM(I_IVI_DMM_TEMPERATURE),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmTemperature : IUnknown
{
//----------------------------------------- TransducerType
   [propput, HELP_DMM(TRANSDUCER_TYPE)]
   HRESULT TransducerType([in] IviDmmTransducerTypeEnum newVal);

   [propget, HELP_DMM(TRANSDUCER_TYPE)]
   HRESULT TransducerType([out, retval] IviDmmTransducerTypeEnum* pVal);


//----------------------------------------- RTD Interface Reference
   [ propget, HELP_DMM(RTD) ]
   HRESULT RTD ([out, retval] IIviDmmRTD **pVal);


//----------------------------------------- Thermocouple Interface Reference
   [ propget, HELP_DMM(THERMOCOUPLE) ]
   HRESULT Thermocouple ([out, retval] IIviDmmThermocouple **pVal);


//----------------------------------------- Thermistor Interface Reference
   [ propget, HELP_DMM(THERMISTOR) ]
   HRESULT Thermistor ([out, retval] IIviDmmThermistor **pVal);

};


//-----------------------------------------------------------------------------
//   RTD Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM_RTD),
   HELP_DMM(I_IVI_DMM_RTD),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmRTD : IUnknown
{
//----------------------------------------- Configure
   [HELP_DMM(CONFIGURE_RTD)]
   HRESULT Configure(
         [in] DOUBLE Alpha,
         [in] DOUBLE Resistance);


//----------------------------------------- Alpha
   [propput, HELP_DMM(RTD_ALPHA)]
   HRESULT Alpha([in] DOUBLE newVal);

   [propget, HELP_DMM(RTD_ALPHA)]
   HRESULT Alpha([out, retval] DOUBLE* pVal);


//----------------------------------------- Resistance
   [propput, HELP_DMM(RTD_RESISTANCE)]
   HRESULT Resistance([in] DOUBLE newVal);

   [propget, HELP_DMM(RTD_RESISTANCE)]
   HRESULT Resistance([out, retval] DOUBLE* pVal);

};
```

```
//----------------------------------------------------------------------------
//   Thermocouple Interface
//----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM_THERMOCOUPLE),
   HELP_DMM(I_IVI_DMM_THERMOCOUPLE),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmThermocouple : IUnknown
{
//---------------------------------------- Configure
   [HELP_DMM(CONFIGURE_THERMOCOUPLE)]
   HRESULT Configure(
         [in] IviDmmThermocoupleTypeEnum Type,
         [in] IviDmmRefJunctionTypeEnum RefJunctionType);


//---------------------------------------- FixedRefJunction
   [propput, HELP_DMM(FIXED_REF_JUNCTION)]
   HRESULT FixedRefJunction([in] DOUBLE newVal);

   [propget, HELP_DMM(FIXED_REF_JUNCTION)]
   HRESULT FixedRefJunction([out, retval] DOUBLE* pVal);


//---------------------------------------- RefJunctionType
   [propput, HELP_DMM(REF_JUNCTION_TYPE)]
   HRESULT RefJunctionType([in] IviDmmRefJunctionTypeEnum newVal);

   [propget, HELP_DMM(REF_JUNCTION_TYPE)]
   HRESULT RefJunctionType([out, retval] IviDmmRefJunctionTypeEnum* pVal);


//---------------------------------------- Type
   [propput, HELP_DMM(THERMOCOUPLE_TYPE)]
   HRESULT Type([in] IviDmmThermocoupleTypeEnum newVal);

   [propget, HELP_DMM(THERMOCOUPLE_TYPE)]
   HRESULT Type([out, retval] IviDmmThermocoupleTypeEnum* pVal);

};


//----------------------------------------------------------------------------
//   Thermistor Interface
//----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_DMM_THERMISTOR),
   HELP_DMM(I_IVI_DMM_THERMISTOR),
   oleautomation,
   pointer_default(unique)
]
interface IIviDmmThermistor : IUnknown
{
//---------------------------------------- Resistance
   [propput, HELP_DMM(THERMISTOR_RESISTANCE)]
   HRESULT Resistance([in] DOUBLE newVal);

   [propget, HELP_DMM(THERMISTOR_RESISTANCE)]
   HRESULT Resistance([out, retval] DOUBLE* pVal);

};


/*************************** REVISION  LOG *********************************
 * $Log: /IVI/IviTypeLibraries/IviDmmTypeLib/IviDmm.idl $
```

```
 *
 * 12    3/07/02 9:42p Jmh00
 * JMH change [in,out] arrays to [out,retval] arrays
 *
 * 11    2/20/02 11:55a Jmh00
 * JMH Add code to generate correct .h files
 *
 * 10    2/19/02 4:40p Jmh00
 * JMH - Final changes prior to vote on the spec
 *
 * 9     1/24/02 8:58a Jmh00
 * JMH Emergency check in due to failing disk drive
 *
 * 8     11/06/01 8:37a Jmh00
 * JMH - Final IDL check
 *
 * 7     9/06/01 1:16p Jmh00
 * JMH Fix problem with importlib IviDriverTypeLib
 *
 * 6     8/14/01 11:46p Jmh00
 * JMH Restructured type libraries project
 *
 * 5     8/03/01 11:00a Jmh00
 * JMH Changes for review version of spec
 ***************************** END OF FILE *********************************/
#endif // !defined(IVI_DMM_IDL_INCLUDED_)
```

D.3 IviDmmEnglish.idl

```
#if !defined(IVI_DMM_IDL_ENGLISH_INCLUDED_)
#define IVI_DMM_IDL_ENGLISH_INCLUDED_
/*****************************************************************************
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
 *  All rights reserved.
 *
 *
 * FILENAME     : IviDmmEnglish.idl
 *
 * STATUS       : APPROVED.
 * COMPILER     : MSVC++ 6.0, sp4 MIDL
 * CONTENT      : IVI DMM Instrument Class Standard IDL
 *                help context IDs and help strings
 *
 * $Archive: /IVI/IviTypeLibraries/IviDmmTypeLib/IviDmmEnglish.idl $
 * $Revision: 4 $
 *****************************************************************************/


#define HC_DMM_BASE  200


//---------------------------------------------------------------------------
//     TYPEDEF ENUMS
//---------------------------------------------------------------------------

#define HC_DMM_HRESULTS                                 HC_DMM_BASE + 0

#define HC_DMM_APERTURE_TIME_UNITS_ENUM                 HC_DMM_BASE + 1
#define HC_DMM_AUTO_RANGE_ENUM                          HC_DMM_BASE + 2
#define HC_DMM_AUTO_ZERO_ENUM                           HC_DMM_BASE + 3
#define HC_DMM_FREQ_VOLTAGE_RANGE_ENUM                  HC_DMM_BASE + 4
#define HC_DMM_FUNCTION_ENUM                            HC_DMM_BASE + 5
#define HC_DMM_MAX_TIME_ENUM                            HC_DMM_BASE + 6
#define HC_DMM_MEAS_COMPLETE_DEST_ENUM                  HC_DMM_BASE + 7
#define HC_DMM_REF_JUNCTION_TYPE_ENUM                   HC_DMM_BASE + 8
#define HC_DMM_SAMPLE_TRIGGER_ENUM                      HC_DMM_BASE + 9
#define HC_DMM_THERMOCOUPLE_TYPE_ENUM                   HC_DMM_BASE + 10
#define HC_DMM_TRANSDUCER_TYPE_ENUM                     HC_DMM_BASE + 11
#define HC_DMM_TRIGGER_DELAY_ENUM                       HC_DMM_BASE + 12
```

```
#define HC_DMM_TRIGGER_SLOPE_ENUM                       HC_DMM_BASE + 13
#define HC_DMM_TRIGGER_SOURCE_ENUM                      HC_DMM_BASE + 14


//---------------------------------------------------------------------------
//   Ivi Dmm Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM                                HC_DMM_BASE + 15
#define HC_DMM_CONFIGURE                                HC_DMM_BASE + 16
#define HC_DMM_FUNCTION                                 HC_DMM_BASE + 17
#define HC_DMM_RANGE                                    HC_DMM_BASE + 18
#define HC_DMM_RESOLUTION                               HC_DMM_BASE + 19
#define HC_DMM_AC                                       HC_DMM_BASE + 20
#define HC_DMM_ADVANCED                                 HC_DMM_BASE + 21
#define HC_DMM_FREQUENCY                                HC_DMM_BASE + 22
#define HC_DMM_MEASUREMENT                              HC_DMM_BASE + 23
#define HC_DMM_TEMPERATURE                              HC_DMM_BASE + 24
#define HC_DMM_TRIGGER                                  HC_DMM_BASE + 25


//---------------------------------------------------------------------------
//   Measurement Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_MEASUREMENT                    HC_DMM_BASE + 26
#define HC_DMM_INITIATE                                 HC_DMM_BASE + 27
#define HC_DMM_ABORT                                    HC_DMM_BASE + 28
#define HC_DMM_FETCH                                    HC_DMM_BASE + 29
#define HC_DMM_FETCH_MULTI_POINT                        HC_DMM_BASE + 30
#define HC_DMM_READ                                     HC_DMM_BASE + 31
#define HC_DMM_READ_MULTI_POINT                         HC_DMM_BASE + 32
#define HC_DMM_SEND_SOFTWARE_TRIGGER                    HC_DMM_BASE + 33
#define HC_DMM_IS_OVER_RANGE                            HC_DMM_BASE + 34


//---------------------------------------------------------------------------
//   Trigger Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_TRIGGER                        HC_DMM_BASE + 35
#define HC_DMM_CONFIGURE_TRIGGER                        HC_DMM_BASE + 36
#define HC_DMM_SLOPE                                    HC_DMM_BASE + 37
#define HC_DMM_DELAY                                    HC_DMM_BASE + 38
#define HC_DMM_SOURCE                                   HC_DMM_BASE + 39
#define HC_DMM_MULTIPOINT                               HC_DMM_BASE + 40


//---------------------------------------------------------------------------
//   MultiPoint Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_MULTIPOINT                     HC_DMM_BASE + 41
#define HC_DMM_CONFIGURE_MULTI_POINT                    HC_DMM_BASE + 42
#define HC_DMM_MEASUREMENT_COMPLETE                     HC_DMM_BASE + 43
#define HC_DMM_COUNT                                    HC_DMM_BASE + 44
#define HC_DMM_SAMPLE_COUNT                             HC_DMM_BASE + 45
#define HC_DMM_SAMPLE_INTERVAL                          HC_DMM_BASE + 46
#define HC_DMM_SAMPLE_TRIGGER                           HC_DMM_BASE + 47


//---------------------------------------------------------------------------
//   AC Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_AC                             HC_DMM_BASE + 48
#define HC_DMM_CONFIGURE_AC_BANDWIDTH                   HC_DMM_BASE + 49
#define HC_DMM_AC_FREQUENCY_MAX                         HC_DMM_BASE + 50
#define HC_DMM_AC_FREQUENCY_MIN                         HC_DMM_BASE + 51
```

```
//---------------------------------------------------------------------------
//   Advanced Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_ADVANCED                       HC_DMM_BASE + 52
#define HC_DMM_ACTUAL_RANGE                             HC_DMM_BASE + 53
#define HC_DMM_APERTURE_TIME                            HC_DMM_BASE + 54
#define HC_DMM_APERTURE_TIME_UNITS                      HC_DMM_BASE + 55
#define HC_DMM_AUTO_ZERO                                HC_DMM_BASE + 56
#define HC_DMM_POWERLINE_FREQUENCY                      HC_DMM_BASE + 57


//---------------------------------------------------------------------------
//   Frequency Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_FREQUENCY                      HC_DMM_BASE + 58
#define HC_DMM_FREQUENCY_VOLTAGE_RANGE                  HC_DMM_BASE + 59


//---------------------------------------------------------------------------
//   Temperature Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_TEMPERATURE                    HC_DMM_BASE + 60
#define HC_DMM_TRANSDUCER_TYPE                          HC_DMM_BASE + 61
#define HC_DMM_RTD                                      HC_DMM_BASE + 62
#define HC_DMM_THERMOCOUPLE                             HC_DMM_BASE + 63
#define HC_DMM_THERMISTOR                               HC_DMM_BASE + 64


//---------------------------------------------------------------------------
//   RTD Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_RTD                            HC_DMM_BASE + 65
#define HC_DMM_CONFIGURE_RTD                            HC_DMM_BASE + 66
#define HC_DMM_RTD_ALPHA                                HC_DMM_BASE + 67
#define HC_DMM_RTD_RESISTANCE                           HC_DMM_BASE + 68


//---------------------------------------------------------------------------
//   Thermocouple Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_THERMOCOUPLE                   HC_DMM_BASE + 69
#define HC_DMM_CONFIGURE_THERMOCOUPLE                   HC_DMM_BASE + 70
#define HC_DMM_FIXED_REF_JUNCTION                       HC_DMM_BASE + 71
#define HC_DMM_REF_JUNCTION_TYPE                        HC_DMM_BASE + 72
#define HC_DMM_THERMOCOUPLE_TYPE                        HC_DMM_BASE + 73


//---------------------------------------------------------------------------
//   Thermistor Interface
//---------------------------------------------------------------------------

#define HC_DMM_I_IVI_DMM_THERMISTOR                     HC_DMM_BASE + 74
#define HC_DMM_THERMISTOR_RESISTANCE                    HC_DMM_BASE + 75


//---------------------------------------------------------------------------
//   Help Strings
//---------------------------------------------------------------------------

//-----------------------------------------------------------------------
//      TYPEDEF ENUMS
//-----------------------------------------------------------------------

#define HS_DMM_HRESULTS \
"IVI DMM class defined HRESULTS"
```

```
#define HS_DMM_APERTURE_TIME_UNITS_ENUM \
"IVI DMM class-compliant values for ApertureTimeUnits"

#define HS_DMM_AUTO_RANGE_ENUM \
"IVI DMM class-compliant AutoRange values that can be used for Range"

#define HS_DMM_AUTO_ZERO_ENUM \
"IVI DMM class-compliant values for AutoZero"

#define HS_DMM_FREQ_VOLTAGE_RANGE_ENUM \
"IVI DMM class-compliant values for frequency VoltageRange"

#define HS_DMM_FUNCTION_ENUM \
"IVI DMM class-compliant values for Function"

#define HS_DMM_MAX_TIME_ENUM \
"IVI DMM class-compliant values for MaxTime"

#define HS_DMM_MEAS_COMPLETE_DEST_ENUM \
"IVI DMM class-compliant values for multipoint MeasurementComplete"

#define HS_DMM_REF_JUNCTION_TYPE_ENUM \
"IVI DMM class-compliant values for thermocouple RefJunctionType"

#define HS_DMM_SAMPLE_TRIGGER_ENUM \
"IVI DMM class-compliant values for multipoint SampleTrigger"

#define HS_DMM_THERMOCOUPLE_TYPE_ENUM \
"IVI DMM class-compliant values for thermocouple Type"

#define HS_DMM_TRANSDUCER_TYPE_ENUM \
"IVI DMM class-compliant values for temperature TransducerType"

#define HS_DMM_TRIGGER_DELAY_ENUM \
"IVI DMM class-compliant values for trigger Delay"

#define HS_DMM_TRIGGER_SLOPE_ENUM \
"IVI DMM class-compliant values for trigger Slope"

#define HS_DMM_TRIGGER_SOURCE_ENUM \
"IVI DMM class-compliant values for trigger Source"


//--------------------------------------------------------------------------
//   Ivi Dmm Interface
//--------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM \
"IVI DMM class-compliant root interface"

#define HS_DMM_CONFIGURE \
"Configures the Function, Range, and Resolution properties.  If the value \
of the Range parameter is Auto Range On, then the Resolution parameter is \
ignored."

#define HS_DMM_FUNCTION \
"The measurement function.  This property determines the units for Range \
and Resolution and the values returned by the Read, Read Multiple Point, \
Fetch, and Fetch Multiple Point methods."

#define HS_DMM_RANGE \
"The measurement range, coerced by the driver to the appropriate range for \
the instrument.  Positive values set the absolute value of the maximum \
measurement expected.  Negative values set Auto Range mode.  Units \
are determined by Function."

#define HS_DMM_RESOLUTION \
"The measurement resolution in absolute units.  Units are determined by \
Function."

#define HS_DMM_AC \
```

```
"Pointer to the class-compliant IIviDmmAc interface"

#define HS_DMM_ADVANCED \
"Pointer to the class-compliant IIviDmmAdvanced interface"

#define HS_DMM_FREQUENCY \
"Pointer to the class-compliant IIviDmmFrequency interface"

#define HS_DMM_MEASUREMENT \
"Pointer to the class-compliant IIviDmmMeasurement interface"

#define HS_DMM_TEMPERATURE \
"Pointer to the class-compliant IIviDmmTemperature interface"

#define HS_DMM_TRIGGER \
"Pointer to the class-compliant IIviDmmTrigger interface"


//---------------------------------------------------------------------------
//   Measurement Interface
//---------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_MEASUREMENT \
"IVI DMM class-compliant measurement interface"

#define HS_DMM_INITIATE \
"Initiates a measurement. When this method executes, the DMM leaves the \
idle state and waits for a trigger. "

#define HS_DMM_ABORT \
"Aborts a previously initiated measurement and returns the DMM to the idle \
state."

#define HS_DMM_FETCH \
"Returns the measured value from a measurement that the Initiate \
method initiates.  If an overrange condition occurs, the value contains \
IEEE NaN and the method returns an overrange error."

#define HS_DMM_FETCH_MULTI_POINT \
"Returns an array of values from a measurement that the \
Initiate method initiates.  If an overrange condition occurs, the \
corresponding element of the array contains IEEE NaN and the method \
returns an overrange error."

#define HS_DMM_READ \
"Initiates a measurement, waits for the DMM to return to the idle state, \
and returns the measured value.  If an overrange condition \
occurrs, the value contains IEEE NaN and the method returns an overrange \
error."

#define HS_DMM_READ_MULTI_POINT \
"Initiates a measurement, waits for the DMM to return to the idle state, \
and returns an array of values.  If an overrange condition occurs, the \
corresponding element of the array contains IEEE NaN and the method \
returns an overrange error."

#define HS_DMM_SEND_SOFTWARE_TRIGGER \
"Sends a software trigger, which causes the DMM to take a measurement."

#define HS_DMM_IS_OVER_RANGE \
"Takes a measurement value obtained from one of the Read or Fetch \
methods and determines if the value is a valid measurement value or a value \
indicating that an overrange condition occurred."


//---------------------------------------------------------------------------
//   Trigger Interface
//---------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_TRIGGER \
"IVI DMM class-compliant trigger interface"
```

```
#define HS_DMM_CONFIGURE_TRIGGER \
"Configures the trigger Source and Delay properties."

#define HS_DMM_SLOPE \
"The polarity of the external trigger slope, which determines whether the \
DMM triggers on either the rising or the falling edge of the external \
trigger source."

#define HS_DMM_DELAY \
"The interval between the time when the DMM receives the trigger and the \
time when it takes a measurement. Positive values set the trigger delay in \
seconds. Negative values set auto delay mode."

#define HS_DMM_SOURCE \
"The trigger source."

#define HS_DMM_MULTIPOINT \
"Pointer to the class-compliant IIviDmmMultipoint interface"


//--------------------------------------------------------------------------
//   MultiPoint Interface
//--------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_MULTIPOINT \
"IVI DMM class-compliant multipoint interface"

#define HS_DMM_CONFIGURE_MULTI_POINT \
"Configures multipoint trigger Count, SampleCount, SampleTrigger and \
SampleInterval properties."

#define HS_DMM_MEASUREMENT_COMPLETE \
"The destination of the measurement-complete signal generated after each \
measurement."

#define HS_DMM_COUNT \
"The number of triggers the DMM accepts before it returns to the idle state."

#define HS_DMM_SAMPLE_COUNT \
"The number of measurements the DMM takes each time it receives a trigger."

#define HS_DMM_SAMPLE_INTERVAL \
"The interval between samples in seconds. Applies only when Sample Count is \
greater than 1 and Sample Trigger is Interval."

#define HS_DMM_SAMPLE_TRIGGER \
"The sample trigger source.  If the value of the Sample Count is greater \
than 1, the DMM enters the Wait-For-Sample-Trigger state after taking a \
single measurement.  When a sample trigger occurs, the DMM takes the next \
measurement."


//--------------------------------------------------------------------------
//   AC Interface
//--------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_AC \
"IVI DMM class-compliant AC interface"

#define HS_DMM_CONFIGURE_AC_BANDWIDTH \
"Configures the FrequencyMax and FrequencyMin properties for DMMs that take \
AC voltage or AC current measurements."

#define HS_DMM_AC_FREQUENCY_MAX \
"The maximum frequency component of the input signal for AC measurements.  \
The value of this property affects instrument behavior only when the \
Function property is set to an AC voltage or AC current measurement."

#define HS_DMM_AC_FREQUENCY_MIN \
"The minimum frequency component of the input signal for AC measurements.  \
```

```
The value of this property affects instrument behavior only when the \
Function property is set to an AC voltage or AC current measurement."


//---------------------------------------------------------------------------
//   Advanced Interface
//---------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_ADVANCED \
"IVI DMM class-compliant advanced features interface"

#define HS_DMM_ACTUAL_RANGE \
"The actual range that the DMM is currently using, even if it is \
auto-ranging."

#define HS_DMM_APERTURE_TIME \
"The measurement aperture time (also known as integration time) based on \
the present configuration.  Units are specified by the property \
ApertureTimeUnits."

#define HS_DMM_APERTURE_TIME_UNITS \
"The units for the ApertureTime property."

#define HS_DMM_AUTO_ZERO \
"The auto-zero mode.  When the auto-zero mode is enabled, the DMM internally \
disconnects the input signal and takes a Zero Reading. The DMM then subtracts \
the Zero Reading from the measurement."

#define HS_DMM_POWERLINE_FREQUENCY \
"The power line frequency in Hertz."


//---------------------------------------------------------------------------
//   Frequency Interface
//---------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_FREQUENCY \
"IVI DMM class-compliant frequency interface"

#define HS_DMM_FREQUENCY_VOLTAGE_RANGE \
"The expected maximum voltage level of the input signal for frequency and \
period measurements. Positive values set the manual range. Negative \
values set Auto Range mode.  The units are specified in Volts RMS."


//---------------------------------------------------------------------------
//   Temperature Interface
//---------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_TEMPERATURE \
"IVI DMM class-compliant temperature interface"

#define HS_DMM_TRANSDUCER_TYPE \
"The type of device used to measure the temperature. This property affects \
instrument behavior only when Function is set to a temperature measurement."

#define HS_DMM_RTD \
"Pointer to the class-compliant IIviDmmRtd interface"

#define HS_DMM_THERMOCOUPLE \
"Pointer to the class-compliant IIviDmmThermocouple interface"

#define HS_DMM_THERMISTOR \
"Pointer to the class-compliant IIviDmmThermistor interface"


//---------------------------------------------------------------------------
//   RTD Interface
//---------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_RTD \
```

```
"IVI DMM class-compliant RTD interface"

#define HS_DMM_CONFIGURE_RTD \
"Configures the Alpha and Resistance parameters for a resistance temperature \
device."

#define HS_DMM_RTD_ALPHA \
"The alpha parameter for a resistance temperature device (RTD).  Applies only \
when the Temperature Transducer Type is set to 2 Wire RTD or 4 Wire RTD."

#define HS_DMM_RTD_RESISTANCE \
"The R0 parameter (resistance) for a resistance temperature device (RTD). \
Also known as the RTD reference value.  Applies only when the Temperature \
Transducer Type is set to 2 Wire RTD or 4 Wire RTD."


//---------------------------------------------------------------------------
//   Thermocouple Interface
//---------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_THERMOCOUPLE \
"IVI DMM class-compliant thermocouple interface"

#define HS_DMM_CONFIGURE_THERMOCOUPLE \
"Configures the thermocouple Type and RefJunctionType properties of a \
thermocouple.  Applies only when the Temperature Transducer Type is \
Thermocouple."

#define HS_DMM_FIXED_REF_JUNCTION \
"The external reference junction temperature when a fixed reference junction \
thermocouple is used to measure temperature, or the thermocouple junction \
temperature of an instrument without an internal temperature sensor, in \
degrees Celcius."

#define HS_DMM_REF_JUNCTION_TYPE \
"The type of reference junction to be used in the reference junction \
compensation of a thermocouple measurement. Applies only when the Temperature \
Transducer Type is Thermocouple."

#define HS_DMM_THERMOCOUPLE_TYPE \
"The type of thermocouple used to measure the temperature. Applies only when \
the Temperature Transducer Type is Thermocouple."


//---------------------------------------------------------------------------
//   Thermistor Interface
//---------------------------------------------------------------------------

#define HS_DMM_I_IVI_DMM_THERMISTOR \
"IVI DMM class-compliant thermistor interface"

#define HS_DMM_THERMISTOR_RESISTANCE \
"The resistance of the thermistor in Ohms.  Applies only when the \
Temperature Transducer Type property is Thermistor."


/***************************** REVISION  LOG *********************************
 * $Log: /IVI/IviTypeLibraries/IviDmmTypeLib/IviDmmEnglish.idl $
 *
 * 4     2/19/02 4:40p Jmh00
 * JMH - Final changes prior to vote on the spec
 *
 * 3     8/03/01 11:00a Jmh00
 * JMH Changes for review version of spec
 ***************************** END OF FILE *********************************/
#endif // !defined(IVI_DMM_IDL_ENGLISH_INCLUDED_)
```