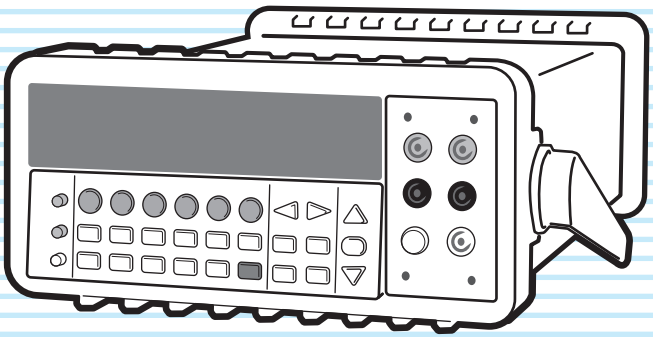
KIKUSUI

Part No. IB022462
Nov. 2012

Remote Interface Manual

6-1/2 Digit Multimeter

DME1600



Thank you for purchasing the DME1600 6-1/2 digit multimeter.

About the operation manuals

There are two operation manuals for the DME1600: the User's Manual and the Remote Interface Manual (this manual).

The manuals are intended for users of the DME1600 and their instructors. These manuals assume that the reader has knowledge about electrical aspects of measuring instruments.

- User's Manual
  This manual is intended for first-time users of the DME1600. It gives an overview of the DME1600, connecting procedures, safety precautions, etc. Please read through and understand this manual before operating the product.
- Remote Interface Manual (this manual)
  This manual explains how to control the DME1600 remotely using SCPI commands.
  The interface manual is written for readers with sufficient basic knowledge of how to control measuring instruments using a PC.

Every effort has been made to ensure the accuracy of this manual. However, if you have any questions or find any errors or omissions, please contact your Kikusui agent or distributor.

After reading, always keep the manual nearby so that you may refer to it as needed. When moving the product to another location, be sure to bring the manual as well.

You can download the most recent version of the manuals from the Kikusui Electronics Corporation website (http://www.kikusui.co.jp/en/download/).

The product that this manual covers

This manual is for the DME1600.

When contacting us about the product, please provide us with:
The model (written on the front panel)
The serial number (written on the rear panel)

Before reading this manual

First read the User's Manual, which includes information on the product's hardware, to avoid connecting or operating the product incorrectly.

Trademarks

Microsoft, Windows, and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other company and product names used in this manual are trademarks or registered trademarks of their respective owners.

Copyrights

Notations used in this manual

- In this manual, the DME1600 6-1/2 Digit Multimeter is also referred to as the "DME1600."
- The term "PC" is used to refer generally to both personal computers and workstations.
- The following markings are used in the explanations in the text.

⚠CAUTION

Indicates a potentially hazardous situation which, if ignored, may result in damage to the product or other property.

NOTE

Indicates information that you should know.

See

Indicates a reference to detailed information.

Contents

This page left blank intentionally.

1

Setup

This chapter explains the settings that you need to configure to use the interfaces.

Selecting the Remote Interface

The DME1600 has USB(USBTMC) and GPIB remote interfaces. You can use either interface but not both at the same time.

Select the remote interface you want to use from the front panel.

USB interface

1. **Press MENU.**
2. **Press PREV or NEXT to select INTERFACE.**
3. **Press ENTER.**
4. **Press PREV or NEXT to select USB.**
5. **Press ENTER.**
6. **Press PREV or NEXT to select ENABLE or DISABLE.**
7. **Press ENTER.**

GPIB interface

Only DME1600s that are equipped with the GPIB interface card, which is a factory option, have support for GPIB.

1. **Press MENU.**
2. **Press PREV or NEXT to select INTERFACE.**
3. **Press ENTER.**
4. **Press PREV or NEXT to select GPIB.**
5. **Press ENTER.**
6. **Press PREV or NEXT to move between digits and the up and down keys to set the GPIB address.**

   The factory default GPIB address is 22. You can set the address to a number from 0 to 31.
7. **Press ENTER to enter the GPIB address.**

Pass/Fail Output from the USB Connector

⚠CAUTION	Doing the following may damage the connected device: Do not connect a normal USB interface device to the DME1600's USB connector when pass/fail signal output is enabled.

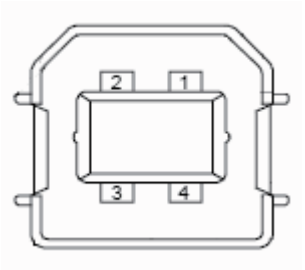
NOTE	When pass/fail signal output is enabled, you cannot use USB for remote control. Use GPIB instead.(Only for models with GPIB cards installed.)

You can output the results of upper and lower limit testing from the USB connector on the rear panel.

When the USB interface is disabled, the internal pass and fail TTL output signals (upper and lower limit testing) are connected to the USB connector.

The pass/fail signal uses negative logic and indicates whether the measured values that will be output next passed or failed. For each acquired measured value, the signal is active (low level) for approximately 2 ms (±100 µs).

The DME1600 uses a B type USB connector.



USB connector pinout

Contact	Signal	Normal wire color	Description
1	VBUS	Red	Floating
2	D-	White	Upper and lower limit test pass
3	D+	Green	Upper and lower limit test fail
4	GND	Black	GND

Configuring pass/fail signal output

You can configure pass/fail signal output using one of the following two methods.

■ Using the MENU key

1. **Press MENU.**
2. **Press PREV or NEXT to select INTERFACE.**
3. **Press ENTER.**
4. **Press PREV or NEXT to select USB.**
5. **Press ENTER.**
6. **Press PREV or NEXT to select ENABLE or DISABLE.**
   Pass/fail signal output is enabled when you disable the USB interface.
7. **Press ENTER.**

Pass/fail output from the USB connector (continued)

■ Using the CONFIG key

1. **Press CONFIG.**
2. **Press LIMITS (SHIFT+RATIO).**
3. **Press PREV or NEXT to select OUTPUT.**
4. **Press ENTER.**
5. **Press PREV or NEXT to select ENABLE or DISABLE.**
6. **Press ENTER.**

2

Message Reference

This chapter explains the SCPI commands.

Message Overview

The information that is transferred between the controller (PC) and the DME1600 is referred to as “messages.”

The DME1600 uses the SCPI language for these messages.

The messages that the PC sends to the DME1600 are commands. The messages that the DME1600 sends to the PC are responses.

Commands are used to execute functions or change settings on the DME1600 or to query the DME1600’s settings or status. Responses are used to return the DME1600’s settings or status.

NOTE You need to set an appropriate wait time for sending and receiving messages. Communication errors may occur if a wait time has not been set.

Command hierarchy

SCPI is an ASCII-based command language that was designed for test and measuring equipment. The command structure is composed of the common roots and nodes that are the building blocks of the SCPI subsystem. A command consists of a program header, parameters, and punctuation marks.

The following table uses the SENSe subsystem as an example to explain the hierarchy.

Program Header	Parameters	Node Level
[SENSe:]		Root node
VOLTage		2nd level
:DC		3rd level
:RANGe	{<range>|MINimum|MAXimum}	4th level
:RANGe?	[MINimum|MAXimum]	4th level
FREQuency:		2nd level
:VOLTage		3rd level
:RANGe	{<range>|MINimum|MAXimum}	4th level
:RANGe?	[MINimum|MAXimum]	4th level

A colon (:) separates a higher node from a lower node.

SCPI command syntax

Command syntax

In this manual, SCPI commands are expressed in the following format.

Example:

VOLTage:DC:RANGe {<range>|MINimum|MAXimum}

- SCPI commands can be written in long form (with all the characters) or in short form (omitting the lowercase characters).

  SCPI commands can be transmitted in either long form or short form.

- SCPI commands are not case sensitive. VOLT, Volt, and volt are all received as the short form of the VOLTage command.

  VOLTAGE, Voltage, and voltage are all received as the long form of the VOLTage command.

- A space separates a program header and its parameters.
- Multiple parameters are separated by commas.
- Compound commands can be created through the concatenation of two commands with a semicolon.

  Example

  TRIG:DELAY 1; COUNT 10

  This compound command sends the same commands as the two following commands.

  TRIG:DELAY 1

  TRIG:COUNT 10

- Program headers are separated by colons.
- By using colons and semicolons, you can concatenate commands of different subsystems.

  Example

  SAMP:COUN 10;:TRIG:SOUR EXT

Special symbols and characters

The special symbols and characters that are used in this manual for the SCPI command syntax are explained below.

Symbol or character	Definition
< >	Character strings inside the < and > symbols indicate program data. Do not include the < and > symbols in the actual program.
{ }	Characters and numbers delimited by "|" inside the { and } symbols indicate that one of the delimited items is to be selected. Do not include the { and } symbols in the actual program.
[ ]	Character strings inside [and] indicate optional data. When optional data is not sent with the program, the default value is sent. Do not include the [and] symbols in the actual program.

SCPI command syntax (continued)

Queries

You can query the settings and status of the DME1600.

To make a query, append a question mark to the end of the program header section.

NOTE If you want to send two queries on separate lines, send the second query after you have received the response to the first one. If you send query commands on two lines at the same time, you may receive an incomplete response.

Terminating character strings

The command strings sent to the DME1600 are terminated with <new line> characters. The IEEE-488 EOI (end or identify) message is treated the same as <new line> characters, so you can use the EOI message instead of <new line> characters as the end of a command string. You can also use <carriage return> + <new line>.

When a command string is terminated, the current SCPI command path is reset to the root level.

Parameters

The SCPI language defines various data formats for use in messages.

■ Numeric parameters

Commands that require numeric parameters can contain all widely used decimal expressions, including optional signs, decimal points, and scientific notation. They can also contain special numeric parameter values, such as MINimum, MAXimum, and DEFault. You can also attach an engineering unit suffix (such as M, K, or u) to a numerical parameter when you send it. When the DME1600 can only accept specific values, it automatically rounds the input parameter values. The following is a command that contains a numeric parameter.

```
VOLTage:DC:RANGe {<range>|MINimum|MAXimum}
```

■ Discrete parameters

Discrete parameters are used for settings that have a limited number of values (such as BUS, IMMediate, and EXTernal). Like command keywords, discrete parameters can be transmitted in short form or long form, and they are not case-sensitive.

Query responses are always given in all caps in short form. The following is a command that contains a discrete parameter.

```
TRIGger:SOURce {BUS|IMMediate|EXTernal}
```

■ Boolean parameters

Boolean parameters can be either true or false. The DME1600 recognizes parameter values of “OFF” and “0” as indicating false Boolean values. The DME1600 recognizes parameter values of “ON” and “1” as indicating true Boolean values. The DME1600 only responds to Boolean parameter queries with “0” or “1.” The following is a command that contains a Boolean parameter.

```
INPut:IMPedance:AUTO {OFF|ON}
```

■ Character string parameters

You can enter any combination of ASCII characters for a character string parameter. Character strings are enclosed by single or double quotation marks. The opening and closing quotation marks must match (you cannot mix single and double quotation marks). If you want to include a quotation mark as part of the string, enter two consecutive quotation marks (with no characters between them). The following is a command that contains a character string parameter.

```
DISPlay:TEXT <quoted string>
```

Using the MIN and MAX parameters

For most commands, you can enter MINimum or MAXimum as the parameter. For example, instead of selecting a voltage range, you can use MIN to specify the minimum voltage range or MAX to specify the maximum voltage range.

Example

```
VOLTage:DC:RANGe {<range>|MINimum|MAXimum}
```

MEASure Commands

The MEASure? commands lack flexibility, but using them is the simplest way to program DME1600 measurements. After you select the measurement function, range, and resolution, the DME1600 automatically sets the other parameters, makes measurements, and sends the results to the output buffer.

MEAS:VOLT:DC

Starts DC voltage measurement with the specified range and resolution settings. Measured values are sent to the output buffer.

Command `MEASure:VOLTage:DC?`
`{<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:VOLT:DC:RAT

Starts DC:DC ratio measurement with the specified range and resolution settings. Measured values are sent to the output buffer. In ratio measurement, the specified range is applied to the input signal, but the auto range is determined by the reference signal.

Command `MEASure:VOLTage:DC:RATio?`
`{<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:VOLT:AC

Starts AC voltage measurement with the specified range and resolution settings. Measured values are sent to the output buffer. In AC measurement, the default resolution is 5 1/2 digits. Therefore, only the front panel display is affected by the specified resolution parameter.

Command `MEASure:VOLTage:AC?`
`{<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:CURR:DC

Starts DC current measurement with the specified range and resolution settings. Measured values are sent to the output buffer.

Command `MEASure:CURRent:DC?`
`{<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:CURR:AC

Starts AC current measurement with the specified range and resolution settings. Measured values are sent to the output buffer. In AC measurement, the default resolution is 5 1/2 digits. Therefore, only the front panel display is affected by the specified resolution parameter.

Command `MEASure:CURRent:AC?`
`{<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:RES

Starts 2-wire resistance measurement with the specified range and resolution settings. Measured values are sent to the output buffer.

Command `MEASure:RESistance?`
`{<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:FRES

Starts 4-wire resistance measurement with the specified range and resolution settings. Measured values are sent to the output buffer.

Command `MEASure:FRESistance? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:FREQ

Starts frequency measurement with the specified range and resolution settings. Measured values are sent to the output buffer. In frequency measurement, the DME1600 uses a single range for all input and output signals from 3 Hz to 300 kHz. If no signal is applied, the measured frequency is zero.

Command `MEASure:FREQuency? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:PER

Starts period measurement with the specified range and resolution settings. Measured values are sent to the output buffer. In period measurement, the DME1600 uses a single range for all input and output signals from 0.33 s to 3.3 µs. If no signal is applied, the measured period is zero.

Command `MEASure:PERiod? {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

MEAS:CONT

Starts continuity testing with fixed settings. Measured values are sent to the output buffer. The range is fixed at 1 kΩ, and the resolution is fixed at 5 1/2 digits.

Command `MEASure:CONTinuity?`

MEAS:DIOD

Starts diode testing with fixed settings. Measured values are sent to the output buffer. The range is fixed at 1 Vdc with a 1 mA current source, and the resolution is fixed at 5 1/2 digits.

Command `MEASure:DIODe?`

MEAS:TEMP

Starts RTD temperature measurement with fixed settings. Measured values are sent to the output buffer.

Command `MEASure:TEMPerature?`

MEAS:TCO

Starts thermocouple temperature measurement with fixed settings. Measured values are sent to the output buffer.

Command `MEASure:TCOuple?`

CONFigure Commands

The CONFigure commands are slightly more flexible than the MEASure? commands. These commands set the parameters, range, and resolution for the specified feature, but they do not start measurement. To start measurement, use the INITiate or READ? command.

CONF:VOLT:DC

Sets the measurement function to DC voltage measurement and sets the range and resolution. This command does not start measurement.

Command `CONFigure:VOLTage:DC {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:VOLT:DC:RAT

Sets the measurement function to DC:DC ratio measurement and sets the range and resolution. This command does not start measurement. In ratio measurement, the specified range is applied to the input signal, but the auto range is determined by the reference signal.

Command `CONFigure:VOLTage:DC:RATio {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:VOLT:AC

Sets the measurement function to AC voltage measurement and sets the range and resolution. This command does not start measurement. In AC measurement, the default resolution is 5 1/2 digits. Therefore, only the front panel display is affected by the specified resolution parameter.

Command `CONFigure:VOLTage:AC {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:CURR:DC

Sets the measurement function to DC current measurement and sets the range and resolution. This command does not start measurement.

Command `CONFigure:CURRent:DC {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:CURR:AC

Sets the measurement function to AC current measurement and sets the range and resolution. This command does not start measurement. In AC measurement, the default resolution is 5 1/2 digits. Therefore, only the front panel display is affected by the specified resolution parameter.

Command `CONFigure:CURRent:AC {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:RES

Sets the measurement function to 2-wire resistance measurement and sets the range and resolution. This command does not start measurement.

Command `CONFigure:RESistance {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:FRES

Sets the measurement function to 4-wire resistance measurement and sets the range and resolution. This command does not start measurement.

Command `CONFigure:FRESistance {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:FREQ

Sets the measurement function to frequency measurement and sets the range and resolution. This command does not start measurement. In frequency measurement, the DME1600 uses a single range for all input and output signals from 3 Hz to 300 kHz. If no signal is applied, the measured frequency is zero.

Command `CONFigure:FREQuency {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:PER

Sets the measurement function to period measurement and sets the range and resolution. This command does not start measurement. In period measurement, the DME1600 uses a single range for all input and output signals from 0.33 s to 3.3 μs. If no signal is applied, the measured period is zero.

Command `CONFigure:PERiod {<range>|MIN|MAX|DEF},{<resolution>|MIN|MAX|DEF}`

CONF:CONT

Sets the measurement function to continuity measurement. This command does not start measurement. The range is fixed at 1 kΩ, and the resolution is fixed at 5 1/2 digits.

Command `CONFigure:CONTinuity`

CONF:DIOD

Sets the measurement function to diode testing. This command does not start measurement. The range is fixed at 1 Vdc with a 1 mA current source, and the resolution is fixed at 5 1/2 digits.

Command `CONFigure:DIODe`

CONF:TEMP

Sets the measurement function to RTD temperature measurement. This command does not start measurement. The resolution is fixed at 6 1/2 digits.

Command `CONFigure:TEMPerature`

CONF:TCO

Sets the measurement function to thermocouple temperature measurement. This command does not start measurement. The resolution is fixed at 6 1/2 digits.

Command `CONFigure:TCOuple`

CONF

Queries the current measurement function settings. The DME1600 returns a character string enclosed in quotation marks.

Command `CONFigure?`

Measurement Configuration Commands

FUNC

Sets the measurement function. Enclose the character string indicating the function name (<function>) in quotation marks (for example, FUNC “VOLT:DC”). Use one of the character strings listed below.

VOLTage:DC	VOLTage:AC	VOLTage:DC:RATio
CURRent:DC	CURRent:AC	RESistance (2-wire resistance)
FRESistance (4-wire resistance)	FREQuency	PERiod
CONTinuity	DIODe	TCOuple
TEMPerature		

Command `[SENSe:]FUNCtion “<function>”`
`[SENSe:]FUNCtion?`

Response In response to FUNC?, the DME1600 returns the measurement function that is currently being executed.

<function>:RANG

Returns the range of the selected measurement function. For frequency and period measurements, the range applies to the signal’s input voltage, not its period (use FREQuency:VOLTage or PERiod:VOLTage). Use MIN to select the minimum range of the selected measurement function; use MAX to select the maximum range.

Use one of the following character strings to indicate the function name (<function>).

VOLTage:DC	VOLTage:AC	CURRent:DC
CURRent:AC	RESistance (2-wire resistance)	FRESistance (4-wire resistance)
FREQuency:VOLtage	PERiod:VOLtage	

Command `[SENSe:]<function>:RANGe {<range>|MINimum|MAXimum}`
`[SENSe:]<function>:RANGe? [MINimum|MAXimum]`

Response In response to <function>:RANG?, the DME1600 returns the range of the selected measurement function.

<function>:RANGe:AUTO

Turns the auto range feature for the selected measurement function on or off. The default setting is “ON.” For frequency or period measurement, use FREQuency:VOLTage or PERiod:VOLTage. When the auto range feature is enabled, the next lower range is selected when the signal falls below 10 % of the current range, and the next higher range is selected when the input signal exceeds 120 % of the current range.

Use one of the following character strings to indicate the function name (<function>).

VOLTage:DC	VOLTage:AC	CURRent:DC
CURRent:AC	RESistance (2-wire resistance)	FRESistance (4-wire resistance)
FREQuency:VOLtage	PERiod:VOLtage	

Command
```
[SENSe:]<function>:RANGe:AUTO {OFF|ON}
[SENSe:]<function>:RANGe:AUTO?
```

Response In response to <function>:RANG:AUTO?, the DME1600 returns “1” (ON) or “0” (OFF).

<function>:RES

Sets the resolution of the selected measurement function (this command does not apply to the frequency, period, or ratio measurement functions). Specify the resolution using the same unit as the measurement function, not using the number of digits. MIN is the minimum allowable value (the maximum value for the resolution). MAX is the maximum allowable value (the minimum value for the resolution).

Use one of the following character strings to indicate the function name (<function>).

VOLTage:DC	VOLTage:AC	CURRent:DC
CURRent:AC	RESistance (2-wire resistance)	FRESistance (4-wire resistance)

Command
```
[SENSe:]<function>:RESolution {<resolution>|MINimum|MAXimum}
[SENSe:]<function>:RESolution? [MINimum|MAXimum]
```

Response In response to <function>:RES?, the DME1600 returns the resolution of the selected measurement function.

UNIT

Sets the unit used for temperature measurement. Cel indicates Celsius, Far indicates Fahrenheit, and K indicates Kelvin. The default value is “Cel.”

Command
```
[SENSe:]UNIT {Cel|Far|K}
[SENSe:]UNIT?
```

Response In response to UNIT?, the DME1600 returns the unit used for temperature measurement.

TCO:TYPE

Sets the thermocouple type. The default setting is “K.”

Command
```
[SENSe:]TCOuple:TYPE {E|J|K|N|R|S|T}
[SENSe:]TCOuple:TYPE?
```

Response In response to TCO:TYPE?, the DME1600 returns the thermocouple type.

TCO:RJUN:RSEL

Sets the reference junction type.

Command `[SENSe:]TCOuple:RJUNction:RSELect {REAL|SIMulated}`
`[SENSe:]TCOuple:RJUNction:RSELect?`

Response In response to TCO:RJUN:RSEL?, the DME1600 returns the reference junction type.

TCO:RJUN:SIM

Sets the default temperature of the simulated reference junction.

Command `[SENSe:]TCOuple:RJUNction:SIMulated {<value>|MINimum|MAXimum}`
`[SENSe:]TCOuple:RJUNction:SIMulated?`

Response In response to TCO:RJUN:SIM?, the DME1600 returns the default temperature of the simulated reference junction.

TCO:RJUN:REAL:OFFS

Sets the offset voltage of the real reference junction.

Command `[SENSe:]TCOuple:RJUNction:REAL:OFFSet`
`{<value>|MINimum|MAXimum}`
`[SENSe:]TCOuple:RJUNction:REAL:OFFSet? [MINimum|MAXimum]`

Response In response to TCO:RJUN:REAL:OFFS?, the DME1600 returns the offset voltage of the real reference junction.

TEMP:RTD:TYPE

Sets the RTD type used for RTD temperature measurement. The default setting is “PT100.”

Command `[SENSe:]TEMPerature:RTD:TYPE`
`{PT100|D100|F100|PT385|PT3916|USER|SPRTD|NTCT}`
`[SENSe:]TEMPerature:RTD:TYPE?`

Response In response to TEMP:RTD:TYPE?, the DME1600 returns the RTD type used for RTD temperature measurement.

TEMP:RTD:RZER

Set the R-Zero coefficient for the RTD type.

Command `[SENSe:]TEMPerature:RTD:RZERo {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:RTD:RZERo? [MINimum|MAXimum]`

Response In response to TEMP:RTD:RZER?, the DME1600 returns the R-Zero coefficient for the RTD type.

TEMP:RTD:ALPH

Sets the alpha coefficient for the specified RTD type.

Command `[SENSe:]TEMPerature:RTD:ALPHa {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:RTD:ALPHa? [MINimum|MAXimum]`

Response In response to TEMP:RTD:ALPH?, the DME1600 returns the alpha coefficient for the RTD type.

TEMP:RTD:BETA

Sets the beta coefficient for the specified RTD type.

Command `[SENSe:]TEMPerature:RTD:BETA {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:RTD:BETA? [MINimum|MAXimum]`

Response In response to TEMP:RTD:BETA?, the DME1600 returns the beta coefficient for the RTD type.

TEMP:RTD:DELT

Sets the delta coefficient for the specified RTD type.

Command `[SENSe:]TEMPerature:RTD:DELTa {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:RTD:DELTa? [MINimum|MAXimum]`

Response In response to TEMP:RTD:DELT?, the DME1600 returns the delta coefficient for the RTD type.

TEMP:SPRTD:RZER

Sets the sensor R value at 0 degrees Celsius.

Command `[SENSe:]TEMPerature:SPRTD:RZERo {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:SPRTD:RZERo? [MINimum|MAXimum]`

Response In response to TEMP:SPRTD:RZER?, the DME1600 returns the sensor R value at 0 degrees Celsius.

TEMP:SPRTD:A4

Sets the A4 coefficient.

Command `[SENSe:]TEMPerature:SPRTD:A4 {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:SPRTD:A4? [MINimum|MAXimum]`

Response In response to TEMP:SPRTD:A4?, the DME1600 returns the A4 coefficient.

TEMP:SPRTD:B4

Sets the B4 coefficient.

Command `[SENSe:]TEMPerature:SPRTD:B4 {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:SPRTD:B4? [MINimum|MAXimum]`

Response In response to TEMP:SPRTD:B4?, the DME1600 returns the B4 coefficient.

TEMP:SPRTD:AX

Sets the A coefficient.

Command `[SENSe:]TEMPerature:SPRTD:AX {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:SPRTD:AX? [MINimum|MAXimum]`

Response In response to TEMP:SPRTD:AX?, the DME1600 returns the A coefficient.

TEMP:SPRTD:BX

Sets the B coefficient.

Command `[SENSe:]TEMPerature:SPRTD:BX {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:SPRTD:BX? [MINimum|MAXimum]`

Response In response to TEMP:SPRTD:BX?, the DME1600 returns the B coefficient.

TEMP:SPRTD:CX

Sets the C coefficient.

Command `[SENSe:]TEMPerature:SPRTD:CX {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:SPRTD:CX? [MINimum|MAXimum]`

Response In response to TEMP:SPRTD:CX?, the DME1600 returns the C coefficient.

TEMP:SPRTD:DX

Sets the D coefficient.

Command `[SENSe:]TEMPerature:SPRTD:DX {<value>|MINimum|MAXimum}`
`[SENSe:]TEMPerature:SPRTD:DX? [MINimum|MAXimum]`

Response In response to TEMP:SPRTD:DX?, the DME1600 returns the D coefficient.

TEMP:TRAN FRTD

Sets RTD measurement to 4-wire mode.

Command `[SENSe:]TEMPerature:TRANsducer FRTD`

TEMP:TRAN RTD

Sets RTD measurement to 2-wire mode.

Command `[SENSe:]TEMPerature:TRANsducer RTD`

<function>:NPLC

Sets the integration time for the selected measurement function in power line cycles (PLC). The default setting is “1.” This command only applies to DCV, DCI, 2-wire resistance, and 4-wire resistance measurements.

Use one of the following character strings to indicate the function name (<function>).

VOLTage:DC　　CURRent:DC　　RESistance (2-wire resistance)
FRESistance (4-wire resistance)

Command `[SENSe:]<function>:NPLCycles {0.02|0.1|1|10|MINimum|MAXimum}`
`[SENSe:]<function>:NPLCycles? [MINimum|MAXimum]`

Response In response to <function>:NPLC?, the DME1600 returns the integration time for the selected measurement function.

FREQ:APER

Sets the gate time (or aperture time) for frequency measurement to 10 ms (4 1/2 digits), 100 ms (default setting; 5 1/2 digits), or 1 s (6 1/2 digits). The default setting is “0.1.”

Command `[SENSe:]FREQuency:APERture {0.01|0.1|1|MINimum|MAXimum}`
`[SENSe:]FREQuency:APERture? [MINimum|MAXimum]`

Response In response to FREQ:APER?, the DME1600 returns the gate time (or aperture time) for frequency measurement.

PER:APER

Sets the gate time (or aperture time) for period measurement to 10 ms (4 1/2 digits), 100 ms (default setting; 5 1/2 digits), or 1 s (6 1/2 digits). The default setting is “0.1.”

Command `[SENSe:]PERiod:APERture{0.01|0.1|1|MINimum|MAXimum}`
`[SENSe:]PERiod:APERture? [MINimum|MAXimum]`

Response In response to FER:APER?, the DME1600 returns the gate time (or aperture time) for period measurement.

DET:BAND

Sets the minimum input signal frequency. The default setting is “20.” A slow, medium, or fast AC filter is selected according to the specified frequency.

Command `[SENSe:]DETector:BANDwidth {3|20|200|MINimum|MAXimum}`
`[SENSe:]DETector:BANDwidth? [MINimum|MAXimum]`

Response In response to DET:BAND?, the DME1600 returns the bandwidth.

AVER:TCON

Sets the digital filter to moving average mode (MOVing) or repeating average mode (REPeat). The default setting is “MOV.”

Command `[SENSe:]AVERage:TCONtrol {MOVing|REPeat}`
`[SENSe:]AVERage:TCONtrol?`

Response In response to AVER:TCON?, the DME1600 returns the digital filter mode.

AVER:COUN

Sets the number of digital filter stack entries (2 to 100).

Command `[SENSe:]AVERage:COUNt {<value>|MINimum|MAXimum}`
`[SENSe:]AVERage:COUNt? [MINimum|MAXimum]`

Response In response to AVER:COUN?, the DME1600 returns the number of digital filter stack entries.

AVER:STAT

Turns the digital filter on or off. The default setting is “ON.”

Command `[SENSe:]AVERage:STATe {OFF|ON}`
`[SENSe:]AVERage:STATe?`

Response In response to AVER:STAT?, the DME1600 returns “0” (OFF) or “1” (ON).

ZERO:AUTO

Turns auto zero adjustment on or off. The default setting is “ON.” The results for OFF and ONCE are similar. When auto zero adjustment is set to OFF, the DME1600 does not perform a new offset measurement until it is in a trigger-wait state. When auto zero adjustment is set to ONCE, an offset measurement is performed immediately.

Command `[SENSe:]ZERO:AUTO {OFF|ONCE|ON}`
`[SENSe:]ZERO:AUTO?`

Response In response to ZERO:AUTO?, the DME1600 returns “1” (ON) or “0” (OFF or ONCE).

GAIN:AUTO

Turns auto gain on or off. The default setting is “ON.” The results for OFF and ONCE are similar. When auto gain is set to OFF, the DME1600 does not perform a new offset measurement until it is in a trigger-wait state. When auto zero adjustment is set to ONCE, an offset measurement is performed immediately.

Command `[SENSe:]GAIN:AUTO {OFF|ONCE|ON}`
`[SENSe:]GAIN:AUTO?`

Response In response to GAIN:AUTO?, the DME1600 returns “1” (ON) or “0” (OFF or ONCE).

INP:IMP:AUTO

Turns auto input resistance mode for DC voltage measurement on or off. The default setting is “OFF.” When auto input resistance mode is set to ON, the input resistance for the 100 mV, 1 V, and 10 V ranges is set to > 10 GΩ. When auto input resistance mode is set to OFF, the input resistance for all ranges is fixed at 10 MΩ.

Command `INPut:IMPedance:AUTO {OFF|ON}`
`INPut:IMPedance:AUTO?`

Response In response to INP:IMP:AUTO?, the DME1600 returns “1” (ON) or “0” (OFF).

ROUT:TERM

Queries whether the front panel or rear panel input terminals are selected.

Command `ROUTe:TERMinals?`

Response In response to ROUT:TERM?, the DME1600 returns “FRON” or “REAR.”

The following commands are for models that are equipped with a scanner card.

ROUT:CLOS

Sets the channels to close (the channel range is from 1 to 10).

Command
```
ROUTe:CLOSe <channel>
ROUTe:CLOSe?
```

Response In response to ROUT:CLOS?, the DME1600 returns the closed channels.

ROUT:OPEN

Opens all channels.

Command `ROUTe:OPEN`

ROUT:STAT

Returns the scanner card state after scanning.

Command `ROUTe:STATe?`

Response In response to ROUT:STAT?, the DME1600 returns 1 (the scanner card is inserted) or 0 (the scanner card is not inserted).

ROUT:SCAN:FUNC

Sets the measurement function for a channel on the scanner card to VAC, VDC, frequency, 2-wire resistance, or 4-wire resistance measurement or disables the channel.

Command
```
ROUTe:SCAN:FUNCtion <channel>,{<function>|"VOLT:DC"|"VOLT:AC"
   |"FREQuency"|"RESistance"|"FRESistance"|"NONE"}
ROUTe:SCAN:FUNC? <channel>
```

Response In response to ROUT:SCAN:FUNC? <channel>, the DME1600 returns the measurement function for a channel on the scanner card.

The following commands are for models that are equipped with a scanner card.

ROUT:SCAN:TIMER

Sets the scan interval (in units of seconds).

Command `ROUTe:SCAN:TIMER <value>`
`ROUTe:SCAN:TIMER?`

Response In response to ROUT:SCAN:TIMER?, the DME1600 returns the scan interval.

ROUT:SCAN:COUNT

Sets the scan count.

Command `ROUTe:SCAN:COUNT <value>`
`ROUTe:SCAN:COUNT?`

Response In response to ROUT:SCAN:COUNT?, the DME1600 returns the scan count.

ROUT:SCAN:STAT

Queries the number of scanned channels.

Command `ROUTe:SCAN:STATe?`

ROUT:SCAN:SCAN

Switches to scan mode.

Command `ROUTe:SCAN:SCAN`

ROUTe:SCAN:STEP

Switches to step mode.

Command `ROUTe:SCAN:STEP`

Math Function Commands

There are eight math functions, but only one function can be enabled at a time. Each math function either stores data for later use or performs mathematical operations on the measured values. These eight math functions are available to all measurement functions except for continuity and diode testing.

The math functions use one or more internal registers. You can set the values in some of the registers. The results of the math operations are stored in the registers.

CALC:FUNC

Sets the math function. Only one math function can be enabled at a time. The default setting is for the percentage function to be enabled.

Command `CALCulate:FUNCtion {PERCent|AVERage|NULL|LIMit|MXB|DB|DBM}`
`CALCulate:FUNCtion?`

Response In response to CALC:FUNC?, the DME1600 returns PERC, AVER, NULL, LIM, MXB, DB, or DBM.

CALC:STAT

Turns the selected math function on or off. The default setting is “OFF.”

Command `CALCulate:STATe {OFF|ON}`
`CALCulate:STATe?`

Response In response to CALC:STAT?, the DME1600 returns “0” (OFF) or “1” (ON).

CALC:PERC:TARG

Sets the target value for the percentage math function. This value is cleared when you turn the maximum and minimum math function off, turn off the power, or reset the remote interface settings.

Command `CALCulate:PERCent:TARGet {<value>|MINimum|MAXimum}`
`CALCulate:PERCent:TARGet? [MINimum|MAXimum]`

Response In response to CALC:PERC:TARG?, the DME1600 returns the target value for the percentage math function.

CALC:AVER:MIN

Returns the minimum value detected by the maximum and minimum math function. This value is cleared when you turn the maximum and minimum math function off, turn off the power, or reset the remote interface settings.

Command `CALCulate:AVERage:MINimum?`

CALC:AVER:MAX

Returns the maximum value detected by the maximum and minimum math function. This value is cleared when you turn the maximum and minimum math function off, turn off the power, or reset the remote interface settings.

Command `CALCulate:AVERage:MAXimum?`

CALC:AVER:AVER

Returns the average of all the measured values that have been acquired since the maximum and minimum math function was enabled. This value is cleared when you turn the maximum and minimum math function off, turn off the power, or reset the remote interface settings.

Command `CALCulate:AVERage:AVERage?`

CALC:AVER:COUN

Returns the number of measured values that have been acquired since the maximum and minimum math function was enabled. This value is cleared when you turn the maximum and minimum math function off, turn off the power, or reset the remote interface settings.

Command `CALCulate:AVERage:COUNt?`

CALC:NULL:OFFS

Saves a null value to the DME1600's null register. Before you save a value to the math register, you must turn the math function on. The null value can be set to a value from 0 to ±120 % of the maximum range.

Command `CALCulate:NULL:OFFSet {<value>|MINimum|MAXimum}`
`CALCulate:NULL:OFFSet?`

Response In response to CALC:NULL:OFFS?, the DME1600 returns the null value.

CALC:LIM:LOW

Sets the lower limit for upper and lower limit testing. In relation to the percentage function, the lower limit can be set to a value from 0 to ±120 % of the maximum range.

Command `CALCulate:LIMit:LOWer {<value>|MINimum|MAXimum}`
`CALCulate:LIMit:LOWer?`

Response In response to CALC:LIM:LOW?, the DME1600 returns the lower limit for upper and lower limit testing.

CALC:LIM:UPP

Sets the upper limit for upper and lower limit testing. In relation to the percentage function, the upper limit can be set to a value from 0 to ±120 % of the maximum range.

Command `CALCulate:LIMit:UPPer {<value>|MINimum|MAXimum}`
`CALCulate:LIMit:UPPer?`

Response In response to CALC:LIM:UPP?, the DME1600 returns the upper limit for upper and lower limit testing.

CALC:MXB:MMF

Sets the value of M for the MX+B math function.

Command `CALCulate:MXB:MMFactor {<value>|MINimum|MAXimum}`
`CALCulate:MXB:MMFactor? [MINimum|MAXimum]`

Response In response to CALC:MXB:MMF?, the DME1600 returns the M value.

CALC:MXB:MBF

Sets the value of B for the MX+B math function.

Command `CALCulate:MXB:MBFactor {<value>|MINimum|MAXimum}`
`CALCulate:MXB:MBFactor? [MINimum|MAXimum]`

Response In response to CALC:MXB:MMF?, the DME1600 returns the B value.

CALC:DB:REF

Stores a relative value in the dB relative register. Before you save a value to the math register, you must turn the math function on. The relative value can be a value from 0 dBm to ± 200 dBm.

Command `CALCulate:DB:REFerence {<value>|MINimum|MAXimum}`
`CALCulate:DB:REFerence? [MINimum|MAXimum]`

In response to CALC:DB:REF?, the DME1600 returns the relative dB value.

CALC:DBM:REF

Sets the dBm reference value to a value from 50 Ω to 8000 Ω.

Command `CALCulate:DBM:REFerence {<value>|MINimum|MAXimum}`
`CALCulate:DBM:REFerence? [MINimum|MAXimium]`

Response In response to CALC:DBM:REF?, the DME1600 returns the dBm reference value.

DATA:FEED RDG_STORE

Sets whether values measured with the INITiate command are stored to the DME1600's internal memory (default setting). The default setting (DATA:FEED RDG_STORE,"CALC") is for up to 2000 measured values to be stored to the memory when the INITiate command is executed. For the CONFigure and MEASure? commands, "CALC" is automatically selected. When the memory is disabled (DATA:FEED RDG_STORE,""), values measured using the INITiate command are not stored. This setting is convenient when used with the maximum and minimum math function, because it enables you to determine the average measured value without storing individual measured values. If you use the FETCh? command to send the measured values to the output buffer, an error is generated.

Command `DATA:FEED RDG_STORE,{"CALCulate"|""}`

DATA:FEED

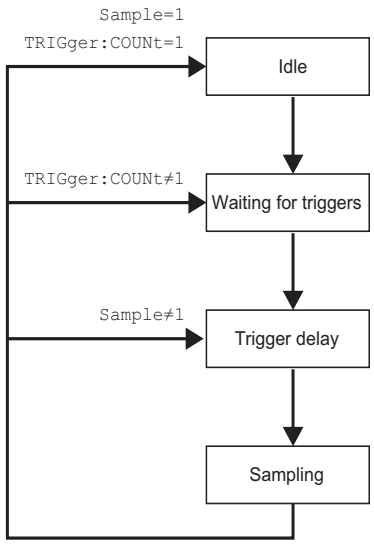
Queries the state of the measurement memory.

Command `DATA:FEED?`

Response In response to DATA:FEED?, the DME1600 returns "CALC" or "".

Trigger Commands

The DME1600 offers a variety of different trigger operations. For each individual measurement function, you can set the trigger mode, trigger source, and trigger settings. A flow chart of the triggering system is shown on the right.



Using the remote interface for triggering is a multi-step process. After configuring the DME1600 by setting the desired function, range, and resolution, set the trigger source that the DME1600 will trigger from. The DME1600 can receive instant internal triggers, software triggers from the remote interface, or external triggers from the rear panel. Next, check whether the DME1600 is ready to receive triggers (trigger-wait state).

The DME1600 detects triggers that it receives when it is in the trigger-wait state. After you finish configuring the DME1600 and selecting the trigger source, you need to set the DME1600 to the trigger-wait state so that it will start measuring when it receives a trigger. The INITiate, READ?, or MEASure? command will set the DME1600 to the trigger-wait state.

INIT

Changes the triggering system state from the “idle state” to the “trigger-wait state.” After the DME1600 receives the INITiate command and the necessary trigger conditions are met, it will start measuring. The measured values are stored in the memory until they are retrieved. Use the FETCh? command to retrieve the measured values.

The INITiate and FETCh? commands are the minimum controls necessary to trigger measurement and retrieve measured values, but they also provide the most flexibility.

Command `INITiate`

READ

Changes the triggering system state from the “idle state” to the “trigger-wait state.” After the DME1600 receives the READ? command and the necessary trigger conditions are met, it will start measuring. Measured values are sent to the output buffer immediately. Enter the measured values into the bus controller. If you do not enter the values into the bus controller, the output buffer will fill, and measurement will stop. When you use the READ? command, the measured values are not stored in the DME1600’s internal memory.

Using the READ? command is the same as using the INITiate command and then the FETCh? command except that the measured values are not stored in the internal buffer.

Command `READ?`

FETC

Sends the measured values that have been stored in the memory by the INITiate command to the output buffer and then reads them using the bus controller.

The INITiate and FETCh? commands are the minimum controls necessary to trigger measurement and retrieve measured values, but they also provide the most flexibility.

Command `FETCh?`

TRIG:SOUR

Sets the trigger source. The DME1600 can accept a software (BUS) trigger, an immediate internal trigger, or a hardware trigger applied to the EXT TRIG input terminal on the rear panel. The default setting is for the DME1600 to use an immediate trigger.

Command
```
TRIGger:SOURce {BUS|IMMediate|EXTernal}
TRIGger:SOURce?
```

Response In response to TRIG:SOUR?, the DME1600 returns the trigger source.

TRIG:DEL

Sets the trigger delay time in seconds. The trigger delay is the time between a trigger signal and the measurements (samples) that follow it. You can set the delay to a value between 0 and 3600 seconds.

Command
```
TRIGger:DELay {<seconds>|MINimum|MAXimum}
TRIGger:DELay?
```

Response In response to TRIG:DEL?, the DME1600 returns the trigger delay.

TRIG:DEL:AUTO

Turns the auto trigger delay on or off. The delay is determined according to the measurement function, range, integration time, and AC filter settings. If you specify the delay time, the auto trigger delay is automatically disabled.

Command
```
TRIGger:DELay:AUTO {OFF|ON}
TRIGger:DELay:AUTO?
```

Response In response to TRIG:DEL:AUTO?, the DME1600 returns “0” (OFF) or “1” (ON).

SAMP:COUN

Sets the number of measurements (samples) per trigger. You can set the number of measurements per trigger to a number from 1 to 50000.

Command
```
SAMPle:COUNt {<value>|MINimum|MAXimum}
SAMPle:COUNt? [MINimum|MAXimum]
```

Response In response to SAMP:COUN?, the DME1600 returns the number of measured values per trigger.

TRIG:COUN

Sets the number of triggers that the DME1600 receives before it returns to the idle state to a number between 1 and 50000. You can specify the INFinite parameter to make the DME1600 receive triggers continuously. The trigger count is ignored in local mode.

Command
```
TRIGger:COUNt {<value>|MINimum|MAXimum|INFinite}
TRIGger:COUNt? [MINimum|MAXimum|INFinite]
```

Response In response to TRIG:COUN, the DME1600 returns the trigger count. If INFinite is selected, the DME1600 returns “9.90000000E+37.”

System Commands

The system commands are not directly related to measurement but are nevertheless important for making measurements.

DISP

Turns the display on or off. The default setting is “ON.”

Command `DISPlay {OFF|ON}`
`DISPlay?`

Response In response to DISP?, the DME1600 returns “0” (OFF) or “1” (ON).

DISP:TEXT

Displays a message on the front panel display. A message of up to 16 characters can be displayed on the bottom line of the display. Additional characters are discarded.

Command `DISPlay:TEXT <quoted string>`
`DISPlay:TEXT?`

Response In response to DISP:TEXT?, the DME1600 returns the message that is sent to the front panel display.

DISP:TEXT:CLE

Clears the message shown on the front panel display.

Command `DISPlay:TEXT:CLEar`

SYST:BEEP

Makes the DME1600 beep once.

Command `SYSTem:BEEPer`

SYST:BEEP:STAT

Turns the beeper on or off. The default setting is “ON.”

Command `SYSTem:BEEPer:STATe {OFF|ON}`
`SYSTem:BEEPer:STATe?`

Response In response to SYST:BEEP:STAT?, the DME1600 returns “0” (OFF) or “1” (ON).

SYST:ERR

Queries the DME1600 error queue. Up to 20 errors can be stored in the queue. Errors are retrieved in first in, first out (FIFO) order. Each error line can be up to 80 characters long.

Command `SYSTem:ERRor?`

SYST:VERS

Queries the current SCPI version.

Command `SYSTem:VERSion?`

L0

Sets the DME1600's language to DEFAULT (the factory default setting).

Command `L0`

L1

Sets the DME1600's language to COMPATIBLE (Agilent A34401 compatible).

Command `L1`

SYSTEM:IDNSTR

Changes the character string that is used to identify the DME1600 from remote applications (the character string can contain up to 39 characters). You can put information such as the maker and product names into the character string.

Command `SYSTEM:IDNSTR <quoted string>`

DATA:POIN

Queries the number of measured values stored in the DME1600's internal memory.

Command `DATA:POINts?`

*RST

Resets the DME1600 to its power-on settings. This command clears the error queue.

Command `*RST`

*IDN

Retrieves the character string (of 35 characters or more in length) used to identify the DME1600.

Command `*IDN?`

SYST:LOC

Switches to local mode. All the keys on the front panel can be used.

Command `SYSTem:LOCal`

SYST:REM

Switches to remote mode. None of the keys on the front panel can be used except for the LOCAL key.

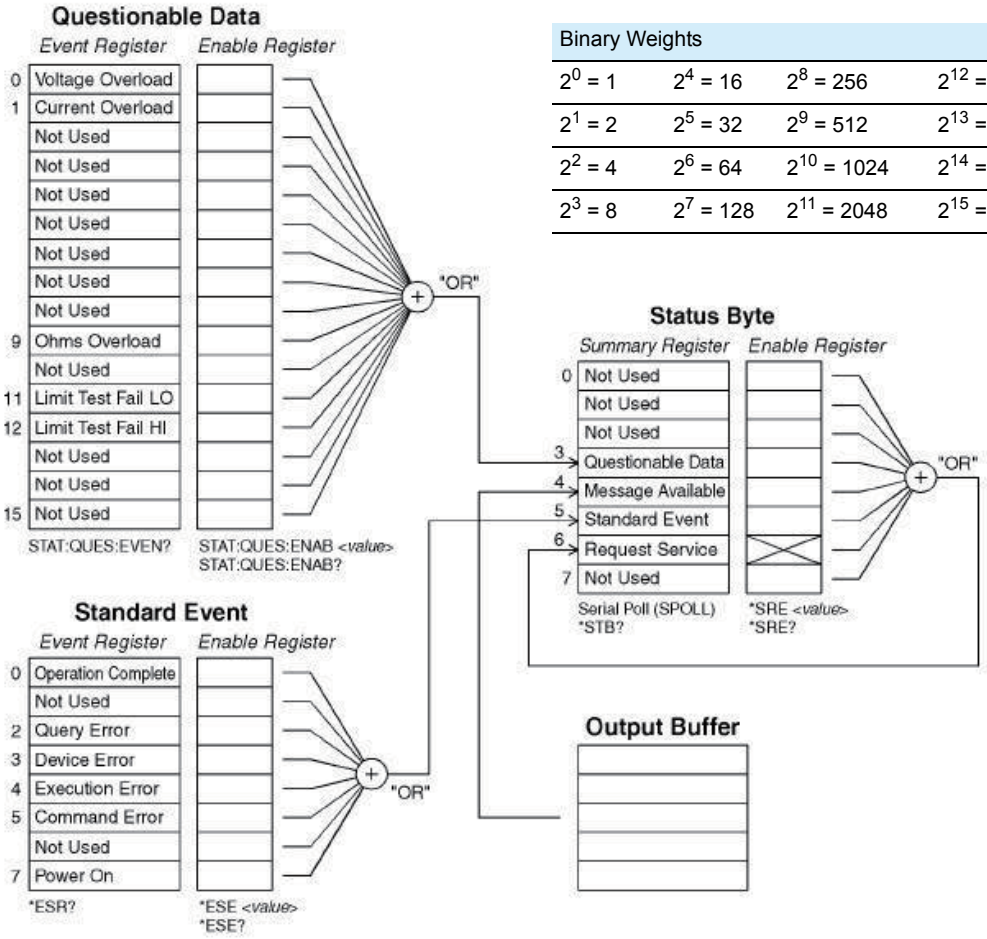
Command `SYSTem:REMote`

SCPI Status System

Status registers are defined in the same way by all SCPI equipment. The status system stores three register groups, which have various equipment conditions. They are the status byte register, the standard event register, and the questionable data register.

The status byte register collects and stores the high-level summary information that is reported in other register groups. The figure below illustrates the SCPI status system. The standard event register and the questionable data register are event registers. They are read-only registers for reporting the states defined by the DME1600. Bits are latched in the event registers. As long as an event bit is set, subsequent state changes will be ignored. The bits in an event register are cleared automatically by commands (such as *ESR?, STAT:QUES:EVEN?, and *CLS). The reset command (*RST) and the device clear command do not clear the bits in the event registers. When you query an event register, the DME1600 returns a decimal value corresponding to the binary-weighted sum of all the bits set in the register.

An enable register is readable and writable and can define which bits in the corresponding event register are OR'ed together to generate a single summary bit. Querying an enable register does not clear it. The *CLS command does not clear the enable registers, but it can clear the bits in the event registers. The STATus:PRESet command clears the questionable data register. To configure the bits in an enable register, you have to write a decimal value corresponding to the binary-weighted sum of the bit configuration you want to set in the register.



Binary Weights			
$2^0 = 1$	$2^4 = 16$	$2^8 = 256$	$2^{12} = 4096$
$2^1 = 2$	$2^5 = 32$	$2^9 = 512$	$2^{13} = 8192$
$2^2 = 4$	$2^6 = 64$	$2^{10} = 1024$	$2^{14} = 16384$
$2^3 = 8$	$2^7 = 128$	$2^{11} = 2048$	$2^{15} = 32768$

SCPI status system

Status byte register

The status byte summary register reports the states of other status registers. The query data in the DME1600's output buffer is reported immediately through the "Message Available" bit (bit 4). Bits are not latched in the summary registers. Clearing an event register clears the corresponding bit in the status byte summary register. After all the messages in the output buffer, including all pending queries, are retrieved, the message available bit is cleared.

Status byte register bit definitions

Bit		Decimal Value	Definition
0	Not used	1	Set to 0
1	Not used	2	Set to 0
2	Not used	4	Set to 0
3	Questionable Data	8	One or more bits (that are enabled in the enable register) are set in the questionable data register.
4	Message Available	16	There is data in the DME1600 output buffer.
5	Standard Event	32	One or more bits (that are enabled in the enable register) are set in the standard event register.
6	Request Service	64	The DME1600 is requesting service (serial poll).
7	Not used	128	Set to 0

The status byte summary register is cleared in the following situations.

- When the *CLS (clear status) command is executed.
- When the standard event register or questionable data register is queried, only the corresponding bit in the summary register is cleared.

The enable register is cleared in the following situations.

- When you turn on the power after configuring the DME1600 using the *PSC 1 command
- When you execute an *SRE 0 command

NOTE The status byte enable register is not cleared when you turn on the power after configuring the DME1600 using the *PSC 0 command.

Status byte register (continued)

Using SRQs (service requests) and serial polling

To use this feature, you have to set your bus controller to respond to the IEEE-488 SRQ and interrupt signals. Use the status byte enable register (SRE) to select the low-level IEEE-488 SRQ signal set by the summary bits. When status byte bit 6 is set, an IEEE-488 SRQ interrupt message will be sent automatically to the bus controller, which will poll the instruments on the bus to identify which one requested service. Retrieving the status byte using an IEEE-488 serial poll or retrieving the event registerwhose summary bit is causing the service request will clear the “Request Service” bit.

To retrieve the status byte summary register, send an IEEE-488 serial poll message. When you query the summary register, the DME1600 returns a decimal value corresponding to the binary-weighted sum of the bits set in the register.

The serial poll clears the “Request Service” bit (bit 6) in the status byte summary register. Other bits are unaffected by the poll. Serial polling does not affect the DME1600’s throughput.

NOTE

IEEE-488.2 does not guarantee synchronization between the bus controller and the device. You can view commands that have already been sent to the DME1600 using the *OPC? command. If you perform serial polling before an *RST, *CLS, or other command is completed, you may receive a report of the previous status.

Using the *STB? command to retrieve the status byte

Using the *STB? command is similar to serial polling except that the *STB? command is processed in the same manner as other device commands. This command returns the same results as serial polling, but it does not clear the “Request Service” bit (bit 6), which is cleared when serial polling is performed. The IEEE-488 bus interface hardware cannot automatically process an *STB? command. This command is executed after the execution of the previous command finishes. You cannot use the *STB? command for polling. Even if you use this command, the status byte summary register is not cleared.

Using an SRQ to interrupt the bus controller

1. **Send a bus device clear message.**
2. **Use the *CLS command to clear the event register.**
3. **Use the *ESE and *SRE commands to enable masks.**
4. **Send the *OPC? command, and enter the result to enable synchronization.**
5. **Enable the bus controller’s IEEE-488 SRQ interrupt.**

Determining when a command sequence has been completed

1. **Clear the DME1600's output buffer by sending a device clear message.**
2. **Use the *CLS command to clear the event register.**
3. **Enable the "Operation Complete" bit by using the *ESE 1 command.**
4. **Send the *OPC? command, and enter the result to enable synchronization.**
5. **When bit 5 is set in the status byte summary register, use a serial poll to check.**

   You can set the DME1600 for an SRQ interrupt by using the *SRE 32 command.

Using the message available bit (MAV)

You can use the status bytes "Message Available" bit (bit 4) to determine when data can be retrieved by the bus controller. The DME1600 automatically enables bit 4 when the first trigger caused by the TRIGger:SOURce:IMMediate command occurs. The DME1600 clears bit 4 after all the messages have been retrieved from the output buffer.

The MAV bit only indicates when the first retrieval caused by the READ? command occurred. The MAV bit is useful for users who do not know when a trigger event such as a BUS or EXTernal event occurred. After the INITiate and FETCH? commands are executed and all specified measurements are completed, the MAV bit is set to 1.

The INITiate command stores the measured values in the DME1600's internal memory. The FETCh? command transfers the measured values to the DME1600's output buffer.

Using *OPC to send a signal when there is data in the output buffer

Normally, you can use the "Operation Complete" bit (bit 0) in the standard event register to generate a signal indicating that the command sequence has completed. After you execute the *OPC command, this bit is set to 1.

If you send the *OPC command after a command to load a messages from the DME1600's output buffer, you can use the "Operation Complete" bit to determine when the message is available. However, if too many messages are generated before the *OPC command is executed, the output buffer will become full, and the DME1600 will stop acquiring measured values.

Standard event register

The standard event register reports the following types of device events.

Power-on detected, command syntax errors, command execution errors, self-test (calibration errors), query errors, and the execution of the *OPC command

Through the enable register, all states are reported in the standard event summary bit. To configure the enable register mask, you need to use an *ESE command to write a decimal value.

NOTE

- Unless the SYSTem:ERRor? command is used to retrieve errors from the error queue, the errors in the DME1600's error queue are recorded as error states (bits 2, 3, 4, and 5 in the standard event register).
- Reading overload states are always reported in the standard event register (bit 3) and the questionable data event register (bits 0, 1, and 9). However, no error message is recorded in the DME1600's error queue.

Standard event register bit definitions

Bit		Decimal Value	Definition
0	Operation Complete	1	All commands prior to and including the *OPC command have been executed.
1	Not used	2	Set to 0
2	Query Error	4	The DME1600 tried to retrieve data from the output buffer, but it was empty. Or a new command was received before the previous query was read. Or the input and output buffers are full.
3	Device Error	8	There was a self-test error, calibration error, or reading overload.
4	Execution Error	16	An execution error occurred.
5	Command Error	32	A command syntax error occurred.
6	Not used	64	Set to 0
7	Power On	128	The power has been turned off and on since the last time the event register was read or cleared.

The standard event register is cleared in the following situations.

- When you execute a *CLS command
- When you query the event register using the *ESR? command.

The enable register is cleared in the following situations.

- When you turn on the power after configuring the DME1600 using the *PSC 1 command
- When you execute an *ERE 0 command

NOTE

The standard event enable register is not cleared when you turn on the power after configuring the DME1600 using the *PSC 0 command.

Questionable data register

The questionable data register reports information about the quality of the DME1600's measured results, such as overload states and the results of upper and lower limit testing.

Through the enable register, all states are reported in the questionable data summary bit.

To configure the bits in the enable register, you need to use a STATus:QUESTionable:ENABle command to write a decimal value.

Questionable data register bit definitions

Bit		Decimal Value	Definition
0	Voltage Overload	1	The measurement range of the DC/AC voltage, frequency, period, diode, or ratio function has been exceeded.
1	Current Overload	2	The DC/AC current has exceeded the measurement range.
2	Not used	4	Set to 0
3	Not used	8	Set to 0
4	Not used	16	Set to 0
5	Not used	32	Set to 0
6	Not used	64	Set to 0
7	Not used	128	Set to 0
8	Not used	256	Set to 0
9	Ohms Overload	512	The measurement range of the 2-wire or 4-wire resistance function has been exceeded.
10	Not used	1024	Set to 0
11	Limit Failed at LO	2048	The measured value is lower than the lower limit.
12	Limit Failed at HI	4096	The measured value is higher than the upper limit.
13	Not used	8192	Set to 0
14	Not used	16384	Set to 0
15	Not used	32768	Set to 0

The questionable data register is cleared in the following situations.

- When you execute a *CLS command
- When you query the event register using the *STATus:QUEStionable:EVENt? command.

The questionable data enable register is cleared in the following situations.

- When you turn on the power without using the *PSC command.
- When the STATus:PRESet command is executed.
- When the STATus:QUEStionable:ENABle 0 command is executed.

Status Reporting Commands

SYST:ERR

Queries the DME1600 error queue. Up to 20 errors can be stored in the queue. Errors are retrieved in first in, first out (FIFO) order. Each error line can be up to 80 characters long.

Command `SYSTem:ERRor?`

STAT:QUES:ENAB

Enables bits in the questionable data enable register. The selected bits are reported to the status byte.

Command `STATus:QUEStionable:ENABle <enable value>`
`STATus:QUEStionable:ENABle?`

Response In response to STAT:QUES:ENAB?, the DME1600 returns a binary-weighted decimal representing the bits set in the enable register.

STAT:QUES:EVEN

Queries the questionable data event register.

Command `STATus:QUEStionable:EVENt?`

Response In response to STAT:QUES:EVEN?, the DME1600 returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

STAT:PRES

Clears all the bits in the questionable data enable register.

Command `STATus:PRESet`

*CLS

Clears the status byte summary register and all event registers.

Command `*CLS`

*ESE

Enables bits in the standard event enable register. The selected bits are reported to the status byte.

Command `*ESE <enable value>`
`*ESE?`

Response In response to *ESE?, the DME1600 returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

*ESR

Queries the standard event register.

Command `*ESR?`

Response In response to *ESR?, the DME1600 returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

*OPC

After the command is executed, the “Operation Complete” bit (bit 0) in the standard event register is set.

Command `*OPC`
`*OPC?`

Response In response to *OPC?, the DME1600 returns “1” to the output buffer after the command is executed.

*PSC

Sets the power-on status. The default setting is “1.”

Command `*PSC {0|1}`
`*PSC?`

Parameter

Value:		
	0	When the power is turned on, the status byte and the standard event register enable mask are not cleared (they are stored in non-volatile memory).
	1	When the power is turned on, the status byte and the standard event register enable mask are cleared.

Response In response to *PSC?, the DME1600 returns “0” (*PSC 0) or “1” (*PSC 1).

*SRE

Enables bits in the status byte enable register.

Command `*SRE <enable value>`
`*SRE?`

Parameter In response to *SRE?, the DME1600 returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

*STB

Queries the status byte summary register.*STB? This command is similar to serial polling except that this command is processed in the same manner as other device commands.*STB? This command returns the same results as serial polling but it does not clear the “Request Service” bit (bit 6), which is cleared when serial polling is performed.

Command `*STB?`

Common Commands

There are commands that are common to the IEEE-488.2 and SCPI standards for functions such as resetting devices and performing self-diagnoses. These common commands start with an asterisk ("*"). These commands may have one or multiple parameters.

*CLS

See p.40

See ***CLS** under "Status Reporting Commands."

*ESE

See p.40

See ***ESE** under "Status Reporting Commands."

*ESR

See p.41

See ***ESR** under "Status Reporting Commands."

*IDN

See p.33

See ***IDN** under "Status Reporting Commands."

*OPC

See p.41

See ***OPC** under "Status Reporting Commands."

*PSC

See p.41

See ***PSC** under "Status Reporting Commands."

*RST

See p.33

See ***RST** under "Status Reporting Commands."

*SRE

See p.41

See ***SRE** under "Status Reporting Commands."

*STB

See p.41

See ***STB** under "Status Reporting Commands."

*TRG

The trigger command.
This is a substitute command for the IEEE 488.1 get message (Group Execute Trigger).

Command `*TRG`

Appendix

A. Error Messages
B. Sample Programs

A Error Messages

Errors are detected and saved to the error queue in first in, first out (FIFO) order. The error that was detected first is displayed first. When all the messages in the error queue are retrieved, the ERROR indicator turns off. The DME1600 beeps once each time an error occurs.

When more than 20 errors have been detected, the last error stored in the error queue (the most recent error) changes to “-350, Too many error.” As long as no errors are deleted from the queue, no additional errors can be stored, and the DME1600 will show “+0, No error.”

The error queue is cleared when the power is turned off or a *CLS (clear status) command is executed. The *RST (reset) command does not clear the error queue.

Error code	Description
-101 Invalid character	An invalid character was found in the command string.
-102 Syntax error	Invalid syntax was found in the command string.
-103 Invalid separator	An invalid separator was found in the command string.
-104 Data type error	A parameter type error was found in the command string.
-105 GET not allowed	A Group Execute Trigger (GET) is not allowed in the command string.
-108 Parameter not allowed	More parameters were found than needed for the command.
-109 Missing parameter	Not enough parameters were received for the command.
-112 Program mnemonic too long	A command header with too many characters was received.
-113 Undefined header	An invalid command was received.
-121 Invalid character in number	An invalid character was found in the number specified for a parameter value.
-123 Numeric overflow	A numeric parameter with exponent larger than 32000 was found.
-124 Too many digits	A numeric parameter was found whose mantissa contained more than 255 digits, excluding leading zeros.
-131 Invalid suffix	A suffix was incorrectly specified for a numeric parameter.
-138 Suffix not allowed	A suffix was received following a numeric parameter which does not accept a suffix.
-148 Character not allowed	A discrete parameter was received but a character string or a numeric parameter was expected.
-151 Invalid string data	An invalid character string was received.
-158 String data not allowed	A character string was received but not allowed for the command.
-160 to -168 Block data errors	Block data is not acceptable.
-170 to -178 Expression errors	The meter does not accept mathematical expression.
-211 Trigger ignored	A Group Execute Trigger (GET) or *TRG was received but the trigger was ignored.
-213 Trigger deadlock	A trigger deadlock occurs when the trigger source is BUS and a READ? Command is received.
-214 Init Ignored	An INITiate command was received but could not be executed because a measurement was already in progress. Send a device clear to halt a measurement in progress and place the meter in the “idle” state.
-221 Settings conflict	This error can be generated in one of the following situations: Situation 1: You sent a CONFigure or MEASure command with autorange enabled and with a fixed resolution. Situation 2: Situation 2: You turned math on and then changed to a math operation that was not valid with the present measurement function.
-222 Data out of range	A numeric parameter value is out of range.

Error code	Description
-223 Too much data	A character string was too long.
-224 Illegal parameter value	A discrete parameter was received which was not a valid choice for the command.
-230 Data Stale	A FETCh? Command was received but the memory was empty.
-350 Too many errors	The error queue is full.
-410 Query INTERRUPTED	A command was received which sends data to the output buffer, but the output buffer contained data from a previous command.
-420 Query UNTERMINATED	The multimeter was addressed to talk (i.e., to send data over the interface) but a command has not been received which send data to the output buffer.
-430 Query DEADLOCKED	A command was received which generates too much data to fit in the output buffer and input buffer is also full. Command execution continues but all data is lost.
-440 Query UNTERMINATED after indefinite response	The *IDN? Command must be the last query command within a command string.
521 Input buffer overflow	
522 Output buffer overflow	
531 Insufficient memory	There is not enough memory to store the requested number of readings in internal memory using the INITiate command. The product of the sample count (SAMPle:COUNt) and the trigger count (TRIGger:COUNt) must not exceed 512 readings.
532 Cannot achieve requested resolution	The multimeter cannot achieve the requested measurement resolution. You may have specified an invalid resolution in the CONFigure or MEASure command.
540 Cannot use overload as math reference	The multimeter cannot store an overload reading (9.90000000E+37) as the math reference for null or dB measurements. The math state is turned off as a result of this condition.
550 Command not allowed in local	The multimeter received a READ? Command while in the local mode.

B Sample Programs

This appendix contains three sample programs.

Program 1: Making a single measurement by using a MEASure? command

In the following example, a MEASure? command is used to measure the AC current once. This is the simplest method for programming DME1600 measurements, but it is also the least flexible. This sample program is written in Visual Basic.

An initialization code must be added to the Sub Main function.

```
Sub Main()

      Dim stat  As ViStatus
      Dim dfltRM As ViSession
      Dim sesn As ViSession
      Dim fList As ViFindList
      Dim desc As String * VI_FIND_BUFLEN
      Dim nList As Long
      Dim ret As Long
      Dim readin As String * 64

      stat = viOpenDefaultRM(dfltRM)
      If (stat < VI_SUCCESS) Then
            'Rem Error initializing VISA ... exiting
            MsgBox "USBTMC resource not found.", vbExclamation, "DME1600
multimeter device test"
            Exit Sub
      End If

      Rem Find all DME1600 USBTMC instruments in the system
      stat = viFindRsrc(dfltRM, "USB[0-9]*::0x164E::0x0DAD::?*INSTR", fList,
nList, desc)
      If (stat < VI_SUCCESS) Then
            'Rem Error finding resources ... exiting
            MsgBox "DME1600 device not found.", vbExclamation, "DME1600
multimeter device test"
            viClose (dfltRM)
            Exit Sub
      End If

      Rem Open a session to each and determine if it matches
      stat = viOpen(dfltRM, desc, VI_NULL, VI_NULL, sesn)
      If (stat < VI_SUCCESS) Then
            MsgBox "Open device failed.", vbExclamation, "DME1600 multimeter
device test"
            stat = viClose(fList)
            Exit Sub
      End If
      Rem send reset command '*RST' -- reset DME1600
      stat = viWrite(sesn, "*RST", 4, ret)
      If (stat < VI_SUCCESS) Then
```

```
            MsgBox "System command error. (*RST)", vbExclamation, "DME1600
multimeter device test"
            stat = viClose(fList)
            Exit Sub
      End If
      Rem send Clear command '*CLS'-- Clear DME1600 status register
      stat = viWrite(sesn, "*CLS", 4, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error. (*CLS)", vbExclamation, "DME1600
multimeter device test"
            stat = viClose(fList)
            Exit Sub
      End If

      Rem send measure command -- Set to 0.1 volt dc range
      stat = viWrite(sesn, "meas:volt:DC? 0.1,0.01", 22, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error. (meas:volt:dc? ...)", vbExclamation,
"DME1600 multimeter device test"
            stat = viClose(fList)
            Exit Sub
      End If

      Rem fetch the measure data
      stat = viRead(sesn, readin, 64, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "Read in data error.", vbExclamation, "DME1600 multimeter
device test"
            stat = viClose(fList)
            Exit Sub
      End If

      Debug.Print "Rdg = "; reading

Rem set to local mode
      stat = viWrite(sesn, "system:local", 12, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error. (system:local)", vbExclamation,
"DME1600 multimeter device test"
            stat = viClose(fList)
            Exit Sub
      End If

      stat = viClose(sesn)
      stat = viClose(fList)
      stat = viClose(dfltRM)

      MsgBox "End of Job."

End Sub
```

Program 2: Using CONFigure commands for math functions

In the following example, CONFigure commands are used for dBm calculation.

The CONFigure commands are slightly more flexible than the MEASure? commands. You can use these commands to gradually change the DME1600's settings. This sample program is written in Visual Basic.

```
PPublic Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Sub main()

      Rem
########################################
      Rem
      Rem    Using NI-VISA library visa32.dll
      Rem
      Rem    Set sample count 5 configuration and
      Rem    read the trigger
      Rem
      Rem
########################################

      Dim stat  As ViStatus
      Dim dfltRM As ViSession
      Dim sesn As ViSession
      Dim fList As ViFindList
      Dim desc As String * VI_FIND_BUFLEN
      Dim nList As Long
      Dim ret As Long
      Dim readin As String * 128
      Dim i As Integer      ' Array index

      stat = viOpenDefaultRM(dfltRM)
      If (stat < VI_SUCCESS) Then
            'Rem Error initializing VISA ... exiting
            MsgBox "USBTMC resource not found.", vbExclamation, "DME1600
multimeter device test"
            Exit Sub
      End If

      Rem Find all DME1600 USBTMC instruments in the system
      stat = viFindRsrc(dfltRM, "USB[0-9]*::0x164E::0x0DAD::?*INSTR", fList,
nList, desc)
      If (stat < VI_SUCCESS) Then
            'Rem Error finding resources ... exiting
            MsgBox "DME1600 device not found.", vbExclamation, "DME1600
multimeter device test"
            viClose (dfltRM)
            Exit Sub
      End If

      Rem Open a session to each and determine if it matches
      stat = viOpen(dfltRM, desc, VI_NULL, VI_NULL, sesn)
      If (stat < VI_SUCCESS) Then
              MsgBox "Open device failed.", vbExclamation, "DME1600 multimeter
device test"
              stat = viClose(fList)
              Exit Sub
      End If
```

```
      Rem send reset command '*RST' -- reset DME1600
      stat = viWrite(sesn, "*RST", 4, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error. (*RST)", vbExclamation, "DME1600
multimeter device test"
            stat = viClose(fList)
            Exit Sub
      End If
      Rem send Clear command '*CLS'-- Clear DME1600 status register
      stat = viWrite(sesn, "*CLS", 4, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error. (*CLS)", vbExclamation, "DME1600
multimeter device test"
            stat = viClose(fList)
            Exit Sub
      End If

      Rem send command -- 50 ohm reference resistance
      stat = viWrite(sesn, "CALC:DBM:REF 50", 15, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error.", vbExclamation, "DME1600 multimeter
device test"
            stat = viClose(fList)
            Exit Sub
      End If
      Rem send command -- Set DME1600 to 1 amp ac range
      stat = viWrite(sesn, "CONF:VOLT:AC 1,0.001", 20, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error.", vbExclamation, "DME1600 multimeter
device test"
            stat = viClose(fList)
            Exit Sub
      End If
      Rem send command -- Select 200 Hz (fast) ac filter
      stat = viWrite(sesn, "DET:BAND 200", 12, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error.", vbExclamation, "DME1600 multimeter
device test"
            stat = viClose(fList)
            Exit Sub
      End If
      Rem send command -- DME1600 will accept 5 triggers
      stat = viWrite(sesn, "SAMP:COUN 5", 11, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error.", vbExclamation, "DME1600 multimeter
device test"
            stat = viClose(fList)
            Exit Sub
      End If
      Rem send command -- Trigger source is IMMediate
      stat = viWrite(sesn, "TRIG:SOUR IMM", 13, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error.", vbExclamation, "DME1600 multimeter
device test"
            stat = viClose(fList)
            Exit Sub
      End If
      Rem send command -- Select dBm function
      stat = viWrite(sesn, "CALC:FUNC DBM", 13, ret)
      If (stat < VI_SUCCESS) Then
            MsgBox "System command error.", vbExclamation, "DME1600 multimeter
device test"
            stat = viClose(fList)
            Exit Sub
```

```
      End If

      Rem send command -- Enable math
      stat = viWrite(sesn, "CALC:STAT ON", 12, ret)
      If (stat < VI_SUCCESS) Then
           MsgBox "System command error.", vbExclamation, "DME1600 multimeter
device test"
           stat = viClose(fList)
           Exit Sub
      End If

      Rem send command -- Take readings
      stat = viWrite(sesn, "READ?" & vbLf, 6, ret)
      If (stat < VI_SUCCESS) Then
           MsgBox "System command error.", vbExclamation, "DME1600 multimeter
device test"
           stat = viClose(fList)
           Exit Sub
      End If

      Sleep (3000)  ' wait for math processing

      Rem fetch the measure data
      stat = viRead(sesn, readin, 128, ret)
      If (stat < VI_SUCCESS) Then
           MsgBox "Read in data error.", vbExclamation, "DME1600 multimeter
device test"
           stat = viClose(fList)
           Exit Sub
      End If

      Rem set to local mode
      stat = viWrite(sesn, "system:local", 12, ret)
      If (stat < VI_SUCCESS) Then
           MsgBox "System command error. (system:local)", vbExclamation,
"DME1600 multimeter device test"
           stat = viClose(fList)
           Exit Sub
      End If

      stat = viClose(sesn)
      stat = viClose(fList)
      stat = viClose(dfltRM)

      For i = 0 To (5 - 1) ' print out the 5 times samples reading
           Debug.Print "Rdgs = "; Mid(readin, i * 16 + 1, 15)
      Next

      MsgBox "End of Job."

End Sub
```

Program 3: DEVQUERY function

The following sample program is written in Visual C++. It is a Win32 console application. A Win32 console application is a Win32 application in which input and output are performed through text entry rather than through a graphic interface. You can quickly create a Win32 application through simple input and output functions.

```
// devquery.cpp : Defines the entry point for the console application.
//
//  Call the NI-VISA library visa32.dll
//
//

#include "stdafx.h"
#include "visa.h"

//standard include for a Microsoft Visual C++ project
#include "stdio.h"
#include "windows.h"

void main(int argc, char* argv[])
{
      // TODO: Add your control notification handler code here

      HINSTANCE hUSBTMCLIB;              // for USBTMC HANDLE
      unsigned long m_defaultRM_usbtmc, m_instr_usbtmc;
      unsigned long m_findList_usbtmc;
      unsigned long m_nCount;
      ViStatus  status;
      int       m_Timeout = 7000;
      char      *pStrout;                // Write out data buffer
      BYTE      pStrin[64];              // Read in data buffer
      int       len;
      ULONG     nWritten;
      ULONG     nRead = 0;
      char      buffer[256];
      char      instrDescriptor[256];

      // Load the NI-VISA library for USBTMC device
      hUSBTMCLIB = LoadLibrary ("visa32.dll");

      if (!hUSBTMCLIB)
      {
      MessageBox(NULL, "NIVISA for USBTMC library not found.", "DME1600
multimeter device test", MB_OK);
      return;
      }

      // Link the libraries
      signed long (__stdcall  *PviOpenDefaultRM_usb)(unsigned long *vi);
      signed long (__stdcall  *PviFindRsrc_usb)(unsigned long sesn, char *expr,
unsigned long *vi, unsigned long *retCnt, char far desc[]);
      signed long (__stdcall  *PviOpen_usb)(unsigned long sesn, char *name,
unsigned long mode, unsigned long timeout, unsigned long *vi);
      signed long (__stdcall  *PviClose_usb)(unsigned long vi);
      signed long (__stdcall  *PviWrite_usb)(unsigned long vi, unsigned char
*name, unsigned long len, unsigned long *retval);
      signed long (__stdcall  *PviRead_usb)(unsigned long vi, unsigned char
*name, unsigned long len, unsigned long *retval);
      signed long (__stdcall  *PviSetAttribute_usb)(unsigned long vi, unsigned
long viAttr, unsigned long attrstat);
```

```
      PviOpenDefaultRM_usb  = (signed long (__stdcall*)(unsigned
long*))GetProcAddress(hUSBTMCLIB, (LPCSTR)"viOpenDefaultRM");
      PviFindRsrc_usb       = (signed long (__stdcall*)(unsigned long, char*,
unsigned long*, unsigned long*, char[]))GetProcAddress(hUSBTMCLIB,
(LPCSTR)"viFindRsrc");
      PviClose_usb          = (signed long (__stdcall*)(unsigned
long))GetProcAddress(hUSBTMCLIB, (LPCSTR)"viClose");
      PviOpen_usb           = (signed long (__stdcall*)(unsigned long, char*,
unsigned long, unsigned long, unsigned long*))GetProcAddress(hUSBTMCLIB,
(LPCSTR)"viOpen");
      PviWrite_usb          = (signed long (__stdcall*)(unsigned long, unsigned
char*, unsigned long, unsigned long*))GetProcAddress(hUSBTMCLIB,
(LPCSTR)"viWrite");
      PviRead_usb           = (signed long (__stdcall*)(unsigned long, unsigned
char*, unsigned long, unsigned long*))GetProcAddress(hUSBTMCLIB,
(LPCSTR)"viRead");
      PviSetAttribute_usb  = (signed long (__stdcall*)(unsigned long, unsigned
long, unsigned long))GetProcAddress(hUSBTMCLIB, (LPCSTR)"viSetAttribute");

      if (PviOpenDefaultRM_usb == NULL ||
          PviFindRsrc_usb      == NULL ||
          PviClose_usb         == NULL ||
          PviOpen_usb          == NULL ||
          PviWrite_usb         == NULL ||
          PviRead_usb          == NULL ||
          PviSetAttribute_usb  == NULL
      )
      {
      FreeLibrary (hUSBTMCLIB);
      hUSBTMCLIB = NULL;
      MessageBox(NULL, "NIVISA for USBTMC library not ready.", "DME1600
multimeter device test", MB_OK);
      return;
      }

      printf("\n ###### Start C++ Example program. ######\n");
      printf(" We check the DME1600 multimeter on USB port and\n");
      printf(" identify the first connected DME1600 device.\n\n");

      // Open Device -- Resource Manager
      status = PviOpenDefaultRM_usb(&m_defaultRM_usbtmc);
      if (status < 0L)
      {
      PviClose_usb(m_defaultRM_usbtmc);
      hUSBTMCLIB = NULL;
      m_defaultRM_usbtmc = 0;
MessageBox(NULL, "USBTMC resource not found.", "DME1600 multimeter device
test", MB_OK);
            return;
      }
      else
      {
      // Find the USBTMC device USB[0-9]*::0x164E::0x0DAD::?*INSTR ( Hex )
            status = PviFindRsrc_usb (m_defaultRM_usbtmc, "USB[0-
9]*::0x164E::0x0DAD::?*INSTR", &m_findList_usbtmc, &m_nCount, instrDescriptor);
      if (status < 0L)
      {
      // Find the USBTMC device USB[0-9]*::0x164E::0x0DAD::?*INSTR ( Dec )
            status = PviFindRsrc_usb (m_defaultRM_usbtmc, "USB[0-
9]*::5710::3501::?*INSTR", &m_findList_usbtmc, &m_nCount, instrDescriptor);
      if (status < 0L)
      {
            PviClose_usb(m_defaultRM_usbtmc);
            hUSBTMCLIB = NULL;
```

```
            m_defaultRM_usbtmc = 0;
            }
            else
            {
            PviOpen_usb(m_defaultRM_usbtmc, instrDescriptor, 0, 0,
&m_instr_usbtmc);
            status = PviSetAttribute_usb(m_instr_usbtmc, VI_ATTR_TMO_VALUE,
m_Timeout);
            }
            }
            else
            {
            PviOpen_usb(m_defaultRM_usbtmc, instrDescriptor, 0, 0,
&m_instr_usbtmc);
            status = PviSetAttribute_usb(m_instr_usbtmc, VI_ATTR_TMO_VALUE,
m_Timeout);
            }
      }

      if (!hUSBTMCLIB)
      {
            printf("DME1600 device connect failed.\n");
            return;
      }

      // Write command "*IDN?" and read the DME1600 identification string
      len = 64;
      pStrout = new char[len];
      ZeroMemory(pStrout, len);
      strcpy(pStrout, "*idn?");
      status = PviWrite_usb(m_instr_usbtmc, (unsigned char *)pStrout, 6,
&nWritten);
      Sleep(30);
      if (status != VI_SUCCESS)
      {
            MessageBox(NULL, "Write to device error.", "DME1600 multimeter
device test", MB_OK);
            PviClose_usb(m_defaultRM_usbtmc);
            hUSBTMCLIB = NULL;
            m_defaultRM_usbtmc = 0;
            return;
      }
      else
      {
            printf(" output : *IDN?\n");
      }
      Sleep(1000);
      // Read data from device
      len = 64;
      if (hUSBTMCLIB)
      {
            status = PviRead_usb(m_instr_usbtmc, pStrin, len, &nRead);
            if (nRead > 0)
            {
            for (len=0; len < (long) nRead; len++)
            {
            buffer[len] = pStrin[len];
            }
            }
            buffer[nRead] = '\0';
            printf(" input : %s\n\n",buffer);
      }

      // Set sample count to 1
```

```
      strcpy(pStrout, "SAMP:COUN 1");
      status = PviWrite_usb(m_instr_usbtmc, (unsigned char *)pStrout, 12,
&nWritten);
      Sleep(30);

      // Set configure Voltage AC, range 0.1A
      strcpy(pStrout, "CONF:VOLT:AC 0.1,0.01");
      status = PviWrite_usb(m_instr_usbtmc, (unsigned char *)pStrout, 22,
&nWritten);
      Sleep(3000);

      // Set configure frequency, range Auto
      strcpy(pStrout, "CONF:FREQ");
      status = PviWrite_usb(m_instr_usbtmc, (unsigned char *)pStrout, 10,
&nWritten);
      Sleep(3000);

      // Set configure Current DC, range 0.1A
      strcpy(pStrout, "CONF:CURR:DC 1,0.01");
      status = PviWrite_usb(m_instr_usbtmc, (unsigned char *)pStrout, 20,
&nWritten);
      Sleep(3000);

      // Fetch the DME1600 measure value ( screen value )
      // Set Voltage DC measure
      strcpy(pStrout, "CONF:VOLT:DC 0.1,0.1");
      status = PviWrite_usb(m_instr_usbtmc, (unsigned char *)pStrout, 21,
&nWritten);
      Sleep(1000);

      // Send read command
      strcpy(pStrout, "READ?");
      status = PviWrite_usb(m_instr_usbtmc, (unsigned char *)pStrout, 6,
&nWritten);
      Sleep(30);
      printf(" output : READ?\n");

      status = PviRead_usb(m_instr_usbtmc, pStrin, 64, &nRead);
      if (nRead > 0)
      {
            for (len=0; len < (long) nRead; len++)
            {
            buffer[len] = pStrin[len];
            }
      }
      buffer[nRead] = '\0';
      printf(" input : %s\n\n", buffer);

      // Set device to local mode
      strcpy(pStrout, "system:local");
      status = PviWrite_usb(m_instr_usbtmc, (unsigned char *)pStrout, 13,
&nWritten);
free(pStrout);

      // Close device
      if (!hUSBTMCLIB)
      return;
      m_nCount = 0;
      m_defaultRM_usbtmc = 0;
      FreeLibrary (hUSBTMCLIB);
      hUSBTMCLIB = NULL;

      return;
}
```

KIKUSUI ELECTRONICS CORP.

1-1-3 Higashiyamata, Tsuzuki-ku, Yokohama, 224-0023, Japan
Tel: +81-45-593-7570 Fax: +81-45-593-7571

Website http://www.kikusui.co.jp