HP BenchLink XL 54600

Software for the HP 54600-Series Oscilloscopes

ActiveX™ Control and Automation Server Programmer's Reference

Warranty

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local HP Sales and Service Office.

Exclusive Remedies

The remedies provided herein are the Buyer's sole and exclusive remedies. HP shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Notice

The information contained in this document is subject to change without notice. **Hewlett-Packard makes 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.** Hewlett-Packard 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. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Hewlett-Packard Company.

Restricted Rights

The Software and Documentation have been developed entirely at private expense. They are delivered and licensed as "commercial computer software" as defined in DFARS 252.227-7013 (Oct 1988), DFARS 252.211-7015 (May 1991), or DFARS 252.227-7014 (Jun 1995), as a "commercial item" as defined in FAR 2.101(a), or as "restricted computer software" as defined in FAR 52.227-19 (Jun 1987) (or any equivalent agency regulation or contract clause), whichever is applicable. You have only those rights provided for such Software and Documentation by the applicable FAR or DFARS clause or the HP standard software agreement for the product involved.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, may contain additional information and replacement pages which you merge into the manual. The dates on this page change only when a new edition is published.

Edition 1 (Part Number 54600-97031) July 1999
Printed in U.S.A.

HP BenchLink XL 54600 Automation

There are two objects in HP BenchLink XL 54600 for automation. The two objects have almost identical capability and functionality. You will use one of the following two objects, depending upon your application.

HP54600Scope Control

Use this Control whenever quick and simple programming is required. This control provides a Graphical User Interface to set I/O parameters and use oscilloscope setups. Recommended for:

- Visual Basic
- VBA in Excel and Word
- HP Vee, LabVIEW

HP54600EZ Automation Server

Use this Automation Server for a smaller file size and faster performance. Recommended for:

- Visual Basic
- Visual C++

Technical Support

Hewlett-Packard (HP) provides programming samples for illustration purposes only, without warranty either expressed or implied, including, but not limited to, the implied warranties of merchantability and/or fitness for a particular purpose.

This help file assumes that you are familiar with the programming language being demonstrated and the tools used to create and debug procedures. HP support engineers can help answer questions relating to the functionality of the software components provided by HP, but they will not modify these samples to provide added functionality or construct procedures to meet your specific needs.

To contact HP for technical assistance, refer to the support numbers listed in the README.TXT file located in the directory where you installed HP BenchLink XL. By default, HP BenchLink XL is installed in the following directory:

```
...\Program Files\Hewlett-Packard\HP BenchLink XL\HP54600
```

If you have limited programming experience, please contact the manufacturer of your development language for further information and assistance.

HP54600Scope Object

The **HP54600Scope Control** allows you to communicate with an HP 54600-series oscilloscope using Visual Basic.

With Visual Basic and the **HP54600Scope Control**, you can:

- control the instrument.
- download waveform data.
- download a bitmap of the screen image.
- return single value measurements such as frequency, voltage and others.

A special feature of the **HP54600Scope Control** allows you to:

- read the current oscilloscope setup (configuration).
- save named oscilloscope setups (configuration) on disk.
- send named oscilloscope setups to the instrument (to return the oscilloscope to a known state).

Use this control for:

- Visual Basic
- Visual Basic for Applications
- Where ActiveX custom controls are supported

Name: HP 54600 Scope Control
Library: HP54600ControlLib
File Name: HP54600.OCX

Syntax

HP54600Scope

Properties

Acquisition Property**, Address** Property, **AnalogChannels** Property, **CheckIDOnInitialize** Property, **ConnectionName** Property, **CountryCode** Property, **LogicChannel** Property, **Measure** Property, **ResetOnInitialize** Property, **Setups** Property, **Timebase** Property, **Trace** Property, **Trigger** Property, **Utilities** Property.

Methods

AboutBox Method, **AutoScale** Method, **CloseConnection** Method, **Connect** Method, **Digitize** Method, **Enter** Method, **GetAllWaveformData** Method, **GetLogicData** Method, **GetScreenImage** Method, **GetWaveformData** Method, **Initialize** Method, **Output** Method, **ReadBytes** Method, **Run** Method, **SaveScreenImage** Method, **ShowPropertyPages** Method, **SingleTrigger** Method, **StopTrigger** Method, **WriteBytes** Method.

Events

SupportMessage Event.

Remarks

- The **HP54600Scope Object** contains Property Pages to make programming the oscilloscope setup easier.
- The **HP54600Scope Control** supports both GPIB and RS-232 interfaces.
- Before you can use the **HP54600Scope** object in your application, you must add the **HP54600.OCX** file to your project. If you use the object in most of your Visual Basic projects, you may want to add it to Visual Basic's Autoload file.
- To distribute applications you create with the **HP54600Scope** object, you must install and register it on the user's computer.
- To see the full object hierarchy in the object browser, you must also reference the HP54600 Automation Server.

HP54600Scope Object Hierarchy

- **HP54600Scope**
  - **Acquisition (AcquisitionHP54600)**
  - **AnalogChannels (AnalogChannelHP54600)**
    - **AnalogChannel (AnalogChannelHP54600)**
  - **LogicChannel (LogicChannelHP54600)**
  - **Measure (MeasureHP54600)**
  - **Setups (SetupsHP54600Scope)**
  - **Timebase (TimebaseHP54600)**
  - **Trace (TraceHP54600)**
  - **Trigger (TriggerHP54600)**
  - **Utilities (UtilitiesHP54600)**
    - **Initialize (IInitializeDevice)**
      - **InitializeI/O (IInitialize)**

HP54600EZ Object

The **HP54600EZ Automation Server** allows you to communicate with a HP 54600-Series oscilloscope using Visual C++ and Visual Basic.

With the **HP54600EZ Automation Server**, you can:

- control the instrument.
- download waveform data.
- download a bitmap of the screen image.
- return single value measurements (such as frequency or voltage).

Recommended for:

- Visual Basic
- Visual C++

Name: HP54600 Automation Server
Description: HP54600ServerLib
File Name: HP54600X.DLL

Syntax

HP54600EZ

Properties

Acquisition Property, **AnalogChannels** Property, **ConnectionName** Property, **LogicChannel** Property, **Measure** Property, **Timebase** Property, **Trace** Property, **Trigger** Property, **Utilities** Property.

Methods

AutoScale Method, **Close** Method, **Connect** Method, **Digitize** Method, **Enter** Method, **GetAllWaveformData** Method, **GetLogicData** Method, **GetScreenImage** Method, **GetWaveformData** Method, **Output** Method, **ReadBytes** Method, **Run** Method, **SaveScreenImage** Method, **SingleTrigger** Method, **StopTrigger** Method, **WriteBytes** Method.

Events

WriteLog Event.

Remarks

- Before you can use **HP54600EZ** object in your application, you must reference the HP 54600 Automation Server.
- To distribute applications you create with the **HP54600EZ** object, you must install and register it on the user's computer.

HP54600EZ Object Hierarchy

- **HP54600EZ**
  - **Acquisition (AcquisitionHP54600)**
  - **AnalogChannels (AnalogChannelHP54600)**
    - **AnalogChannel (AnalogChannelHP54600)**
  - **LogicChannel (LogicChannelHP54600)**
  - **Measure (MeasureHP54600)**
  - **Timebase (TimebaseHP54600)**
  - **Trace (TraceHP54600)**
  - **Trigger (TriggerHP54600)**
  - **Utilities (UtilitiesHP54600)**
    - **Initialize (IInitializeDevice)**
      - **InitializeI/O (IInitialize)**

AboutBox Method

Applies to: HP54600Scope

Shows the About Box.

Syntax

*object.***AboutBox**

Acquisition Property

Applies to: HP54600Scope and HP54600EZ

Returns an AcquisitionHP54600 object that provides a configuration interface for the oscilloscope's acquisition subsystem.

Syntax

*object.***Acquisition**

Data Type

AcquisitionHP54600

Address Property

Applies to: HP54600Scope

Sets/Gets the port and instrument address for I/O communication.

Syntax

object.**Address** [= *instrumentaddress*]

Data Type

String

Settings

- *instrumentaddress* As String defines the port and address.
- Default = GPIB0::7::INSTR

Remarks

GPIB

- Use a string in this form:
- GPIB*m*::*n*
- where *m* is the board number, and *n* is the instrument GPIB address (for example, "GPIB0::22").
- Alternately use a VISA address ("GPIB::22::INSTR"). The I/O operations do not require that VISA be installed on the PC.

RS-232

- Use a string in this form:
- COM*m*::*parametername* = *nn*
- where *m* is the RS-232 port and *parametername* is one of the parameters described below. A comma separates multiple parameter names. Any, all, or no parameters may be used. If a parameter is missing, the default value is used.

Baud=*nnnn*	where *nnnn* are the digits of the baud rate. Default = 9600
Handshake=*s*	where *s* is none, xon_xoff, or dtr_dsr. Default = xon_xoff

Examples

```
"COM1::Baud=9600"
"COM2::Baud=2400,Handshake=xon_xoff"
```

AnalogChannels Property

Applies to: HP54600Scope and HP54600EZ

Returns a reference to an AnalogChannelsHP54600 Collection. The collection provides selection and a configuration interface for a specified analog channel of the oscilloscope. Read-Only.

Syntax

*object.***AnalogChannels**

AutoScale Method

Applies to: HP54600Scope and HP54600EZ

Performs the 'AutoScale' function of the oscilloscope's front panel.

Syntax

*object.***AutoScale**

CheckIDOnInitialize Property

Applies to: HP54600Scope

Sets/gets the Initialize method to check if the instrument is working and is compatible with this control.

Syntax

*object.***CheckIDOnInitialize** = { TRUE | FALSE}

Data Type

- Boolean
- Default = False

Close Method

Applies to: HP54600EZ

Closes the communication connection with the instrument.

Syntax

Object.**Close**

Remarks

- If the session is not open or has already been closed, the Close method does nothing.

CloseConnection Method

Applies to: HP54600Scope

Closes the connection to the automation server.

Syntax

object.**CloseConnection**

Connect Method

Applies to: HP54600EZ

Connects the automation server to an instrument at the specified address or symbolic name.

Syntax

object.**Connect** (*connectionname*, [*IOProgID*])

Settings

- *connectionname* As String is the symbolic name of the connection.
- *IOProgID* As String is an optional I/O address of the connection.

Data Type

- String

Remarks

GPIB

Use a string in this form:

GPIB*m*::*n*

where *m* is the board number, and *n* is the instrument GPIB address (for example, "GPIB0::22").

Alternately use a VISA address ("GPIB::22::INSTR"). The I/O operations do not require that VISA be installed on the PC.

RS-232

Use a string in this form:

COM*m*::*parametername=nn*

where *m* is the RS-232 port and *parametername* is one of the parameters described below. A comma separates multiple parameter names. Any, all, or no parameters may be used. If a parameter is missing, the default value is used.

Baud=*nnnn*	where *nnnn* are the digits of the baud rate. Default = 9600
Handshake=*s*	where *s* is none, xon_xoff, or dtr_dsr. Default = xon_xoff

Examples

```
"COM1::Baud=9600"
"COM2::Baud=2400,Handshake=xon_xoff"
```

ConnectionName Property

Applies to: HP54600Scope and HP54600EZ

Get the instrument's symbolic connection address/name. Read-Only.

Syntax

object.**ConnectionName**

Data Type

- String

CountryCode Property

Applies to: HP54600Scope

Gets/sets the language of the property pages and the help files.

Syntax

object.**CountryCode** [= *language*]

Data Type

- HP54600_CountryCode
- Default = HP54600_CountryCode_English

Settings

- *language* As HP54600_CountryCode sets the language for the property pages and help files.

Remarks

- Changing the country code will cause the help files, HP54600.hlp and HP54600.cnt in the directory of the control to change. When the country code is changed, the corresponding help files in the \help directory are moved to the control directory and renamed to HP54600.hlp and HP54600.cnt.

Digitize Method

Applies to: HP54600Scope and HP54600EZ

Causes an acquisition to take place. Acquisition data is placed in the oscilloscope's channel buffer.

Syntax

object.**Digitize**

Enter Method

Applies to: HP54600Scope and HP54600EZ

Reads data from the instrument as a string or number.

Syntax

object.**Enter** (*result* [, *format*])

Remarks

- Use this method to enter instrument data. Use the Output Method to send instrument commands.
- For most applications that return a string or a number, no format argument is needed. These examples show the most common usage. *scope* is an object in the Applies to list.
- **Returning a string**

```
Dim reply As String
scope.Enter reply
```

- **Returning a number**

```
Dim reading As Double
scope.Enter reading
```

- For higher throughput with large amounts of data, see the Read and Write Methods returned by the IO Property. Reference the "IIO Manager and Utilities" to see these methods in the Object Browser.

Settings

- *result* returns the data. The optional Format parameter determines how the data is returned.
- *format* As String determines the format of the returned data.
- Default = K (Freefield entry)
- *format* is optional, or can contain one or two format identifiers separated by commas. The order of the format identifiers is ignored.
- The format characters are:
- "K" for Freefield entry,
- "#" for don’t flush buffer entry, and
- IEEE 488.2 block formatted data. (See description below.)
- Do not use the Freefield character and the IEEE 488.2 block characters in the same format string.

Format Identifiers

"K" -- Freefield

The data is interpreted based upon the data type of the result parameter. Use the data type best suited for the data returned. K is the default if no Format string is given

result data type	Description
String	Characters are placed in the string. Carriage-return not immediately followed by line-feed is entered into the string. Entry to a string terminates on CR/LF, LF, or a character received with EOI true.
String()	Same as string, but parses the received characters at any comma. The entry terminates as in String or when the array is full.
Numeric	Returns the first number of the ASCII data returned from the instrument. Leading non-numeric characters are ignored. All blanks are ignored. Trailing non-numeric characters and characters sent with EOI true are delimiters. Numeric characters include digits, decimal point, +, -, e, and E when their order is meaningful. Valid data types are Byte, Long, Integer, Double, and Single.
Numeric()	Same as Numeric, but parses the ASCII string from the instrument and fills the array. The entry terminates when the array is full or at the end of data.
Variant	Same as string.
Variant ()	Same as string except that array is filled until end on CR/LF, LF, or a character is received with EOI true.

"#" -- Don't flush buffer

Saves the remaining data in the buffer after completion of Enter method. When the instrument returns several numbers as one ASCII string, you can retain any remaining data in the buffer by using this format character when reading with an Enter method. In the following example, the instrument returns two data points. The first line reads the first data point, and the second line reads the second data point after which the data in the buffer is discarded. The variables *reading1* and *reading2* are declared as double.

```
scope.Enter reading1, "K,#"
scope.Enter reading2, "K"
```

IEEE 488 block data

Using the Enter command with a format statement you can read IEEE 488.2 block data. This is a standard format used by some instruments to return large amounts of data in a binary form.

Setting	Description
I1	Integer, 1 byte
I2BE	Integer, 2 bytes, Big Endian
I2LE	Integer, 2 bytes, Little Endian
I4BE	Integer, 4 bytes, Big Endian
I4LE	Integer, 4 bytes, Little Endian
R4BE	Real, 4 bytes, Big Endian
R4LE	Real, 4 bytes, Little Endian
R8BE	Real, 8 bytes, Big Endian
R8LE	Real, 8 bytes, Little Endian

GetAllWaveformData Method

Applies to: HP54600Scope and HP54600EZ

Returns the waveform data (previously acquired) for all the enabled channels of the oscilloscope.

Syntax

object.**GetAllWaveformData** (*points*, *data*)

Settings

- *points* As HP54600_WaveformPoints indicates the number of waveform data points contained in *Data*.
- *data* As array of Variants contains the returned waveform data.

Remarks

- The number of points depends upon the oscilloscope model used.
- This method is for use in VB and VBA only and is not supported in LabVIEW or HP VEE.

GetLogicData Method

Applies to: HP54600Scope and HP54600EZ

Returns the logic state data (previously acquired) of the logic analysis subsystem.

Syntax

object.**GetLogicData** (*Points*, *Time*, *Data*)

Settings

- *Points* As HP54600_WaveformPoints
- *Time* As Variant is the time corresponding the data.
- *Data* As Variant is the returned data.

Remarks

- This is only supported for the HP 54645D.
- The number of points depends upon the oscilloscope model used.
- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.

GetScreenImage Method

Applies to: HP54600Scope and HP54600EZ

Returns a bitmap image of what is currently displayed on the oscilloscope's display screen.

Syntax

object.**GetScreenImage** [(*format*)]

Data Type

- Picture

Settings

- *format* As HP54600_ImageFormat defines the format for the image. The format can be set to BMP or HiResBMP.

Remarks

- You can use the GetScreenImage method to insert the screen image in a control (such as Image or PictureBox) or you can save the screen image in a variable declared as a Picture.
- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.
- This code inserts the screen image in an image control.

```
Set Image1.Picture = HP54600Scope1.GetScreenImage
```

- This code will set the variable screenShot to the screen image and then insert it in an Image control.

```
Dim screenShot As Picture
Set screenShot = HP54600Scope1.GetScreenImage
Set Image1.Picture = screenShot
```

GetWaveformData Method

Applies to: HP54600Scope and HP54600EZ

Returns the waveform data (previously acquired) for the specified analog channel of the oscilloscope.

Syntax

object.**GetWaveformData** (*channel*, *points*, *time*, *data*)

Settings

- *channel* As HP54600_AnalogChannels sets the analog channel. You may also use a Long ranging from 1 to 4.
- *points* As HP54600_WaveformPoints sets the number of points contained in *Data*.
- *time* As Variant returns the time, in seconds, for the x-axis of the waveform data.
- *data* As Variant contains the waveform data.

Remarks

- The number of points depends upon the oscilloscope model used.
- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.

Initialize Method (HP54600Scope)

Applies to: HP54600Scope

Initializes the device and I/O automation servers and then opens the communication connection with the instrument.

Syntax

object.**Initialize**

Remarks

- If a connection to the instrument cannot be established, this method returns an error (exception).
- If ResetOnInitialize (HP54600Scope Control only) is true, this method sends a *RST to the instrument.
- If CheckIDOnInitialize (HP54600Scope Control only) is true, this method sends a *IDN? Query to the instrument and verifies the instrument response. If an incorrect instrument is found, the Initialize method returns False.

LogicChannel Property

Applies to: HP54600Scope and HP54600EZ

Returns a LogicChannelHP54600 object that provides a configuration interface for the oscilloscope's logic analysis subsystem. Read-Only.

Syntax

object.**LogicChannel**

Data Type

- LogicChannelHP54600

Remarks

- This property is only supported for the HP 54620A/C and HP 54645D.

Measure Property

Applies to: HP54600Scope and HP54600EZ

Returns a MeasureHP54600 object that provides methods for making measurements on the (previously acquired) waveform data of the specified analog channel. Read-Only.

Syntax

object.**Measure**

Output Method

Applies to: HP54600Scope and HP54600EZ

Sends a string to the instrument.

Syntax

object.**Output** (*string*)

Settings

- *string* As String.

Remarks

- Use this command to send instrument commands. Use the Enter method to read the reply from the instrument.

- For example, this code requests a peak-to-peak voltage measurement from an HP 54600B. *scope* is an object in the Applies to list.

```
scope.Output "Measure:VPP?"
```

ReadBytes Method

Applies to: HP54600Scope and HP54600EZ

Reads a response in binary (byte) format from the instrument into the specified buffer.

Syntax

object.**ReadBytes** (*count*, *buffer()*)

Data Type

- *count* As Long returns the number of bytes in *Buffer()*.
- *buffer()* As Byte contains the data.

ResetOnInitialize Property

Applies to: HP54600Scope

Sets/gets the Initialize method to send a reset command to the instrument as part of its execution.

Syntax

object.**ResetOnInitialize**

Data Type

- Boolean
- Default = False

Run Method

Applies to: HP54600Scope and HP54600EZ

Performs the 'Run' waveform acquisition of the oscilloscope's front panel.

Syntax

object.**Run**

SaveScreenImage Method

Applies to: HP54600Scope and HP54600EZ

Saves a bitmap image of what is currently displayed on the oscilloscope's display screen to the specified file.

Syntax

object.**SaveScreenImage** (*filename*, [*format*])

Settings

- *filename* As String contains a valid filename and path.
- *format* As HP54600_ImageFormat sets the type of file to save (BMP or HiResBMP).

Remarks

- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.

Setups Property

Applies to: HP54600Scope

Returns a SetupsHP54600Scope collection. Read-Only.

Syntax

object.**Setups**

Data Type

- SetupsHP54600Scope

Remarks

- This code sends an oscilloscope setup named "MySetup" to the oscilloscope.

```
Dim scopeSetup as object
Set scopeSetup = HP54600Scope1.Setups("MySetup")
scopeSetup.SendToInstrument
```

- You can create and name an oscilloscope setup using the property pages. This named setup can then be called programmatically.
- You can save an oscilloscope setup as a file using the property pages. You can then use this file to send the setup to the oscilloscope.

ShowPropertyPages Method

Applies to: HP54600Scope

Displays the selected property page.

Syntax

object.**ShowPropertyPages** (*pages*)

Settings

- *pages* As HP54600ShowPropertyPage indicates the property page to display.
- Default = HP54600_ShowPropertyPage_Default

Remarks

- Using the property pages during run time to change the I/O parameters creates a temporary change. To make permanent changes, use the Visual Basic property page (right-click the properties menu or press F4).

SingleTrigger Method

Applies to: HP54600Scope and HP54600EZ

Performs the 'Single' waveform acquisition function of the oscilloscope's front panel.

Syntax

object.**SingleTrigger**

SupportMessage Event

Applies to: HP54600Scope

Provides information for product support. Not for general use.

Syntax

object.**SupportMessage** (*message*)

Settings

- *message* As String.

StopTrigger Method

Applies to: HP54600Scope and HP54600EZ

Performs the 'Stop' waveform acquisition of the oscilloscope's front panel.

Syntax

object.**StopTrigger**

Timebase Property

Applies to: HP54600Scope and HP54600EZ

Returns a TimebaseHP54600 object that provides a programming interface for the oscilloscope's horizontal or timebase subsystem. Read-Only.

Syntax

object.**Timebase**

Trace Property

Applies to: HP54600Scope and HP54600EZ

Returns a TraceHP54600 object that provides a programming interface to the oscilloscope's trace storage subsystem. Read-Only.

Syntax

object.**Trace**

Trigger Property

Applies to: HP54600Scope and HP54600EZ

Returns a TriggerHP54600 object that provides a programming interface for the oscilloscope's triggering subsystem. Read-Only.

Syntax

object.**Trigger**

Utilities Property

Applies to: HP54600Scope and HP54600EZ

Returns a UtilitiesHP54600 object that provides an interface to a set of instrument utility functions. Read-Only.

Syntax

object.**Utilities**

WriteBytes Method

Applies to: HP54600Scope and HP54600EZ

Writes data in binary (byte) format to the instrument from the specified buffer.

Syntax

object.**WriteBytes** (*count*, *buffer*)

Data Type

- *count* As Long is the number of bytes in *buffer()*.
- *buffer()* As Byte contains the data.

WriteLog Event

Applies to: HP54600EZ

This event is sent from the ILog interface. The ILog interface sends diagnostic trace messages for every method of the automation server.

Syntax

object.**WriteLog** (*source*, *logging*, *logmessage*)

Settings

- *source* As String is the source of the event. *Source* should be either the InstanceName if not null or the class name parsed from the ComponentProgID.
- *logging* As LogType sets the type of logging as Error, Trace, Warning, or information event.
- *logmessage* As String contains the message to be logged.

Remarks

- The I/O and device servers let the client handle actually writing to a log or trace file, and clients are free to handle the messages however they see fit.

AcquisitionHP54600 Object

Applies to: HP54600Scope and HP54600EZ

Use the AcquisitionHP54600 Object to:

- Get or set the time for the oscilloscope's acquisition.
- Get or set the number of averages to use during acquisition.
- Turn on or off an anti-aliasing function.
- Set the type of acquisition to use.

Syntax

AcquisitionHP54600

Properties

CompletionMinimum Property, **Count** Property, **DitherEnabled** Property, **Type** Property.

CompletionMinimum Property

Applies to: AcquisitionHP54600 Object

Gets/sets the minimum completion criteria for an acquisition. This parameter determines what percentage (0 to 100) of the time buckets need to be full before an acquisition is considered complete.

Syntax

object.**CompletionMinimum** [= *value*]

Data Type

- Long

Settings

- *value* As Long sets the percentage. Ranges from 0 to 100.

Count Property

Applies to: AcquisitionHP54600 Object

Gets/sets the number of values to be averaged for each time bucket before the acquisition is considered to be complete for that time bucket.

Syntax

object.**Count** [= *value*]

Data Type

- HP54600_AcquisitionCount (Enumeration)

Settings

- *value* As HP54600_AcquisitionCount.

Remarks

- This property is not supported by the HP 54620A/C Oscilloscope.
- HP54600_AcquisitionCount can have values of 8, 64, or 256 for the following oscilloscopes: HP 54600B, HP 54602B, HP 54603B, HP 54610B, HP 54615B, and HP 54616B/C.
  HP54600_AcquisitionCount can have values of 1, 4, 8, 16, 32, 64, 128, or 256 for the HP 54645A/D Oscilloscope.

DitherEnabled Property

Applies to: AcquisitionHP54600 Object

Enables/disables an anti-aliasing function for slow sweep speeds. When enabled, the function that adds a small random time offset to the sampling clock.

Syntax

object.**DitherEnabled** [= {True | False}]

Data Type

- Boolean

Remarks

- This is only supported for the HP 54645A/D.

Type Property

Applies to: AcquisitionHP54600 Object

Gets/sets the type of acquisition that is to take place.

Syntax

object.**Type** [(*value*)]

Data Type

- HP54600_AcquisitionType (Enumeration)

Settings

- *value* As HP54600_AcquisitionType sets the acquisition.

Remarks

- The oscilloscope model determines valid acquisition types.
- When set to 'average', the number of averages is set by the Count property.

AnalogChannelsHP54600 Collection

Applies to: HP54600Scope and HP54600EZ

A group of Analog channels. Use this to retrieve a specific AnalogChannelHP54600 and its properties.

Syntax

object.**AnalogChannelsHP54600** (*index*)

Settings

- *index* is a Long used to identify a specific analog channel.

Properties

Count Property

Methods

Item Method

Count Property (Collections)

Applies to: AnalogChannelsHP54600 Collection

Returns the total number of items in the collection. Read-Only.

Syntax

object.**Count**

Data Type

- Long

Item Method

Applies to: AnalogChannelsHP54600 Collection

Returns a specific analog channel.

Syntax

object.**Item** (*channel*)

Settings

- *channel* As Long is an expression that specifies the position of a member of the collection. *channel* must be a number from 1 to the value of the collection's Count property.

Remarks

- If the value provided as *channel* doesn't match any existing member of the collection, an error occurs.
- The Item method is the default method for a collection.

AnalogChannelHP54600 Object

Applies to: HP54600Scope and HP54600EZ

Use this object to set and query the Analog Channel properties.

Syntax

AnalogChannelHP54600

Properties

BandWidthLimit Property, **Enabled** Property, **ProbeAttenuation** Property, **ProbeCoupling** Property, **ProbeMode** Property, **VerticalOffset** Property, **VerticalRange** Property.

BandWidthLimit Property

Applies to: AnalogChannelsHP54600 Object

Enables/disables an internal low-pass filter. When the filter is enabled, the bandwidth of the specified channel is limited to approximately 20 MHz.

Syntax

object.**BandWidthLimit** [= *onoff*]

Settings

- *onoff* As HP54600_OnOffState

Enabled Property

Applies to: AnalogChannelsHP54600 Object

Enable/disables the state of the analog channel.

Syntax

object.**Enabled** [= {True | False}]

Data Type

- Boolean

Remarks

- The state of this property can be set or changed manually using the front panel of the oscilloscope.

ProbeAttenuation Property

Applies to: AnalogChannelsHP54600 Object

Gets/sets the probe attenuation factor for the selected channel. The probe attenuation factor may be 1, 10, 20, or 100.

Syntax

object.**ProbeAttenuation** [= *attenuation*]

Settings

- *attenuation* As HP54600_ProbeAttenuation sets attenuation to 1, 10, 20, or 100.

ProbeCoupling Property

Applies to: AnalogChannelsHP54600 Object

Gets/sets the input coupling for the specified analog channel.

Syntax

object.**ProbeCoupling** [= *coupling*]

Settings

- *coupling* As HP54600_ProbeCoupling sets AC, DC, or GND.

ProbeMode Property

Applies to: AnalogChannelsHP54600 Object

Get/sets the probe sense mode.

Syntax

object.**ProbeMode** [= *mode*]

Settings

- *mode* As HP54600_ProbeMode sets either Auto or Manual.

Remarks

- This property is only supported for the following oscilloscopes:
  HP 54610B
  HP 54615B
  HP 54616B/C
  HP 54645A/D

VerticalOffset Property

Applies to: AnalogChannelsHP54600 Object

Gets/sets the voltage that is represented at center screen for the specified analog channel.

Syntax

object.**VerticalOffset** [= *value*]

Data Type

- Double

VerticalRange Property

Applies to: AnalogChannelsHP54600 Object

Get/sets the full-scale voltage of the specified analog channel.

Syntax

object.**VerticalRange** [= *value*]

Data Type

- Double

Remarks

- This property sets the full-scale voltage; the oscilloscope front panel settings set the voltage per division.

IInitializeDevice Object

Applies to: UtilitiesHP54600

Use this object to initialize the device type.

See additional information about the IInitializeDevice Object.

Properties

InitializeIO Property, **IOProgID** Property.

Methods

Close Method, **InitializationProperties** Method, **Initialize (IInitializeDevice)** Method, **LoadIO** Method, **LoadPropertyBag** Method, **PutOption** Method, **SavePropertyBag** Method.

Close Method

Applies to: IInitializeDevice Object

Closes the communication connection with the instrument.

Syntax

Object.**Close**

Remarks

- If the session is not open or has already been closed, the Close method does nothing.

InitializationProperties Method

Applies to: IInitializeDevice Object

A variant containing a two-dimensional array of initialization properties and values. Property names are stored in the first column, and the corresponding property values in the second column.

Syntax

object.**InitializationProperties** (*pVal*)

Settings

- *pVal* is a variant containing a two-dimensional array which contains property names in the first column and property values in the second column.

Remarks

- The initialization properties and values can be initialized with the LoadPropertyBag method.

Initialize Method (IInitializeDevice)

Applies to: IInitializeDevice Object

Open the communication session with the instrument, using the current values of the initialization properties.

Syntax

object.**Initialize**

Remarks

- To use the Initialize method, you must have either:
  1 Made a prior call to the LoadPropertyBag method, or
  2 Manually loaded and set the I/O server initialization and connection parameters.
- If a connection to the instrument cannot be established, this method returns an error (exception).

InitializeIO Property

Applies to: IInitializeDevice Object

Returns a pointer to the I/O component's Initialize interface. Read-Only.

Syntax

object.**InitializeIO**

Data Type

IInitialize

Remarks

- If an I/O component has been loaded, InitializeIO returns a reference to the I/O component's IInitialize object. If an I/O component has not been loaded, InitializeIO returns a null reference.
- See additional information about the InitializeIO Property.

IOProgID Property

Applies to: IInitializeDevice Object

Gets/sets the Program ID of the loaded I/O object.

Syntax

object.**IOProgID** [= *value*]

Data Type

- String

Settings

- *value* As String sets the Program ID.

Remarks

- The Progream ID is set when an I/O component is loaded. If no I/O component is loaded, this property returns a null string.

LoadIO Method

Applies to: IInitializeDevice Object

Loads the specified I/O component into memory and returns a reference to it. It also does the preliminary connection to the instrument.

Syntax

Object.**LoadIO**

Data Type

- Unknown

Remarks

- LoadIO is used in cases where the initialization is done manually. See the Initialize method and LoadPropertyBag method.
- This method performs the loading of the underlying I/O automation server based on the ConnectionName and IOProgID properties.

LoadPropertyBag Method

Applies to: IInitializeDevice Object

Reads initialization properties from the supplied property bag and sets the properties to the supplied values.

Syntax

*object.***LoadPropertyBag** (*propbag*)

Settings

- *propbag* As IPropertyBag is a valid property bag reference.

Remarks

- After using the LoadPropertyBag method, the Initialize method may be called.
- The property bag must contain enough information to initialize and identify the instrument.
- The property bag does not contain properties used to set the instrument state. Use ReadStateData and WriteStateData to set the device state.

PutOption Method

Applies to: IInitializeDevice Object

Add an option to the option list.

Syntax

*object.***PutOption** (*name*)

Settings

- *name* As String contains the option to add.

Remarks

- The PutOption method is used to specify options that the device component cannot detect programmatically.

SavePropertyBag Method

Applies to: IInitializeDevice Object

Save initialization properties to the supplied property bag.

Syntax

*object.***SavePropertyBag** (*propbag*)

Settings

- *propbag* As IPropertyBag is a valid property bag reference.

IInitialize Object

Applies To: Initialize Property

Returned by the InitializeIO property.

This object is the IInitialize interface of either an I/O server or an instrument server.

Use this object when you are working with multiple servers.

See additional information about the IInitialize Object.

Methods

Close Method, **InitializationProperties** Method, **Initialize** Method(IInitializeDevice), **LoadPropertyBag** Method, **SavePropertyBag** Methods.

Close Method

Applies to: IInitialize Object

Closes the communication connection with the instrument.

Syntax

Object.**Close**

Remarks

- If the session is not open or has already been closed, the Close method does nothing.

InitializationProperties Method

Applies to: IInitialize Object

A variant containing a two-dimensional array of initialization properties and values. Property names are stored in the first column, and the corresponding property values in the second column.

Syntax

object.**InitializationProperties** (*pVal*)

Settings

- *pVal* is a variant containing a two-dimensional array which contains property names in the first column and property values in the second column.

Remarks

- The initialization properties and values can be initialized with the LoadPropertyBag method.

Initialize Method (IInitializeDevice)

Applies to: IInitialize Object

Open the communication session with the instrument, using the current values of the initialialization properties.

Syntax

object.**Initialize**

Remarks

- To use the Initialize method, you must have either:
  1 Made a prior call to the LoadPropertyBag method, or
  2 Manually loaded and set the I/O server initialization and connection parameters.
- If a connection to the instrument cannot be established, this method returns an error (exception).

LoadPropertyBag Method

Applies to: IInitialize Object

Reads initialization properties from the supplied property bag and sets the properties to the supplied values.

Syntax

object.**LoadPropertyBag** (*propbag*)

Settings

- *propbag* As IPropertyBag is a valid property bag reference.

Remarks

- After using the LoadPropertyBag method, the Initialize method may be called.
- The property bag must contain enough information to initialize and identify the instrument.
- The property bag does not contain properties used to set the instrument state. Use ReadStateData and WriteStateData to set the device state.

SavePropertyBag Method

Applies to: IInitialize Object

Save initialization properties to the supplied property bag.

Syntax

object.**SavePropertyBag** (*propbag*)

Settings

- *propbag* As IPropertyBag is a valid property bag reference.

LogicChannelHP54600 Object

Applies to: HP54600Scope and HP54600EZ

With the LogicChannelHP54600 object you can read, write, and control logic pods.

Properties

BitEnabled Property, **EnableStatus** Property, **PodEnabled** Property.

BitEnabled Property

Applies to: LogicChannelHP54600 Object

Enables/disables the individual logic channels (bits) D0 through D15.

Syntax

object.**BitEnabled** (*Bit*) [= {True | False}]

Data Type

- Boolean

Settings

- Bit As HP54600_LogicBits specifies which bit to set or check. Ranges from 0 to 15.

Remarks

- This is only supported for the HP 54620A/C and HP 54645D.
- The state of this property can be set or changed manually using the front panel of the oscilloscope.

EnableStatus Property

Applies to: LogicChannelHP54600 Object

Returns the status (enable/disable state) for all logic channels (bits) as a binary weighted sum. Bit 0 = D0, Bit 15 = D15.

Syntax

object.**EnableStatus**

Data Type

- Integer

Remarks

- This is only supported for the HP 54620A/C and HP 54645D.

PodEnabled Property

Applies to: LogicChannelHP54600 Object

Enables/disables the logic channels in pods. POD1 = D0-D7. POD2 = D8-D15.

Syntax

object.**PodEnabled** (*Pod*) [= {True | False}]

Data Type

- Boolean

Settings

- *Pod* As HP54600_LogicPods selects either pod 1 or pod 2.
- True enables the logic pod, False disables the logic pod.

Remarks

- This is only supported for the HP 54620A/C and HP 54645D.
- The state of this property can be set or changed manually using the front panel of the oscilloscope.

MeasureHP54600 Object

Applies to: HP54600Scope and HP54600EZ

Provides methods to return a single measurement from the oscilloscope.

For example, this code returns the frequency in Hertz from channel 1.

```
Dim MyMeasurement As Double
MyMeasurement = HP54600Scope1.Measure.Frequency(1)
```

Methods

AverageVoltage Method, **DutyCycle** Method, **FallTime** Method, **Frequency** Method, **MaxVoltage** Method, **MinVoltage** Method, **NegativePulseWidth** Method, **PeakToPeakVolts** Method, **Period** Method, **PositivePulseWidth** Method, **RiseTime** Method, **RMSVoltage** Method.

Remarks

- To make a measurement, the portion of the waveform required for that measurement must be displayed.
- To make a frequency measurement, be sure that at least one full waveform cycle is displayed.
- For a pulse width measurement, the entire pulse must be displayed.
- For a fall time (rise time) measurement, the trailing (leading) edge of the waveform must be displayed.
- Except for AverageVoltage and RMSVoltage, which use only one cycle, voltage measurements are made using the entire display. If you want to make a measurement on a particular cycle, display only that cycle on the screen.
- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

AverageVoltage Method

Applies to: MeasureHP54600 Object

Returns the average voltage measurement. Read-Only.

Syntax

object.**AverageVoltage** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AnalogChannels is the Analog channel for the oscilloscope. The analog channel ranges from 1 to 4.

Remarks

- The returned value is scaled in volts.
- The AverageVoltage measurement is performed using one cycle of the waveform in the display.
- For example, this code returns the average voltage of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.AverageVoltage(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

DutyCycle Method

Applies to: MeasureHP54600 Object

Returns the duty cycle as a ratio of the positive pulse width to the period of the waveform. Read-Only.

Syntax

object.**DutyCycle** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AllChannels is the channel for the oscilloscope.

Remarks

- The returned value is scaled as a percent.
- At least one complete waveform must be displayed to make the Duty Cycle measurement.
- For example, this code returns the duty cycle of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.DutyCycle(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

FallTime Method

Applies to: MeasureHP54600 Object

Returns the measured fall-time (in seconds) of the waveform. Read-Only.

Syntax

object.**FallTime** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AnalogChannels is the Analog channel for the oscilloscope. Ranges from 1 to 4.

Remarks

- The returned value is scaled in seconds.
- For a fall time measurement, the leading edge of the waveform must be displayed
- For example, this code returns the fall time of the displayed waveform edge on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.FallTime(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

Frequency Method

Applies to: MeasureHP54600 Object

Returns the frequency (in hertz) of the waveform. Read-Only.

Syntax

object.**Frequency** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AllChannels is the channel for the oscilloscope.

Remarks

- The returned value is scaled in Hertz.
- For a frequency measurement, one complete waveform must be displayed
- For example, this code returns the frequency of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.Frequency(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

MaxVoltage Method

Applies to: MeasureHP54600 Object

Returns the maximum measured voltage of the waveform. Read-Only.

Syntax

object.**MaxVoltage** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AnalogChannels is the Analog channel for the oscilloscope. Ranges from 1 to 4.

Remarks

- The returned value is scaled in Volts.
- Voltage measurements are made using the entire display. If you want to make a measurement on a particular cycle, display only that cycle on the screen.
- For example, this code returns the maximum voltage of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.MaxVoltage(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

MinVoltage Method

Applies to: MeasureHP54600 Object

Returns the minimum measured voltage (in volts) of the waveform. Read-Only.

Syntax

object.**MinVoltage** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AnalogChannels is the Analog channel for the oscilloscope. Ranges from 1 to 4.

Remarks

- The returned value is scaled in Volts.
- Voltage measurements are made using the entire display. If you want to make a measurement on a particular cycle, display only that cycle on the screen.
- For example, this code returns the minimum voltage of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.MinVoltage(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

NegativePulseWidth Method

Applies to: MeasureHP54600 Object

Returns the measured pulse width (in seconds) of the negative portion of the waveform. Read-Only.

Syntax

object.NegativePulseWidth (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AllChannels is the channel for the oscilloscope.

Remarks

- The returned value is scaled in seconds.
- For a pulse width measurement, the entire pulse must be displayed.
- For example, this code returns the negative pulse width of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.NegativePulseWidth(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

PeakToPeakVolts Method

Applies to: MeasureHP54600 Object

Returns the measured peak-to-peak voltage of the waveform. Read-Only.

Syntax

object.**PeakToPeakVolts** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AnalogChannelsis the Analog channel for the oscilloscope. Ranges from 1 to 4.

Remarks

- The returned value is scaled in Volts.
- Voltage measurements are made using the entire display. If you want to make a measurement on a particular cycle, display only that cycle on the screen.
- For example, this code returns the peak-to-peak voltage of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.PeakToPeakVolts(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

Period Method

Applies to: MeasureHP54600 Object

Returns the period (in seconds) of the waveform. Read-Only.

Syntax

object.**Period**(*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AllChannels is the channel for the oscilloscope.

Remarks

- The returned value is scaled in seconds.
- For a period measurement, one complete waveform must be displayed
- For example, this code returns the period of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.Period(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

PositivePulseWidth Method

Applies to: MeasureHP54600 Object

Returns the measured pulse width (in seconds) of the positive portion of the waveform. Read-Only.

Syntax

object.**PositivePulseWidth** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AllChannels is the channel for the oscilloscope.

Remarks

- The returned value is scaled in seconds.
- For a pulse width measurement, the entire pulse must be displayed.
- For example, this code returns the positive pulse width of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.PositivePulseWidth(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

RiseTime Method

Applies to: MeasureHP54600 Object

Returns the measured rise-time (in seconds) of the waveform. Read-Only.

Syntax

object.**RiseTime** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AnalogChannels is the Analog channel for the oscilloscope. Ranges from 1 to 4.

Remarks

- The returned value is scaled in seconds.
- For a rise time measurement, the trailing edge of the waveform must be displayed.
- For example, this code returns the rise time of the displayed waveform edge on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.RiseTime(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

RMSVoltage Method

Applies to: MeasureHP54600 Object

Returns the root-mean-square voltage measurement (in volts) of the waveform. Read-Only.

Syntax

object.**RMSVoltage** (*channel*)

Data Type

- Double

Settings

- *channel* As HP54600_AnalogChannels is the Analog channel for the oscilloscope. Ranges from 1 to 4.

Remarks

- The returned value is scaled in volts.
- The RMS Voltage measurement is performed using one cycle of the waveform in the display.
- For example, this code returns the RMS voltage of the displayed waveform on channel 1.

```
Dim MyMeasurement as Double
MyMeasurement = HP54600Scope1.Measure.RMSVoltage(1)
```

- If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned.

SetupsHP54600Scope Collection

Applies to: HP54600Scope

A collection of oscilloscope setups. Use this collection to retrieve a specific SetupHP54600Scope object and its properties.

Syntax

object.**SetupsHP54600Scope** (*name*)

Settings

- *name* As String is used to identify a specific setup.

Properties

Count Property, **Item** Property.

Methods

Add Method, **Remove** Method.

Remarks

- You can create and test a collection of oscilloscope setups (collection of SetupHP54600Scope objects) during design time using the Setups property pages.
- Each SetupHP54600Scope object contains information about the oscilloscope settings, user comments, version, and model number.

Add Method

Applies to: SetupsHP54600 Collection

Use the Add method to create a collection of oscilloscope configurations. See the SetupsHP54600Scope collection.

Syntax

Object.**Add** (*name*)

Data Type

- SetupHP54600Scope

Settings

- *name* As String is an arbitrary string used to identify the setup being added.

Count Property (Collections)

Applies to: SetupsHP54600 Collection

Returns the total number of items in the collection. Read-Only.

Syntax

object.**Count**

Data Type

- Long

Item Property

Applies to: SetupsHP54600 Collection

Returns a SetupHP54600Scope object. Read-Only.

Syntax

object.**Item** (*name*)

Data Type

- SetupHP54600Scope

Settings

- *name* As Variant identifies the key or index of the object to return.

Remove Method

Applies to: SetupsHP54600 Collection

Removes an object from the SetupsHP54600ScopeCollection.

Syntax

object.**Remove** (*name*)

Settings

- *name* As String specifies the object to remove from the collection.

SetupHP54600Scope Object

Applies to: SetupsHP54600 Collection

Use the SetupHP54600Scope object to:

- Get oscilloscope information such as settings, related user comments, version, and model number.
- Get settings from the oscilloscope.
- Send settings to the oscilloscope.
- Save and retrieve settings to a file.

SetupHP54600Scope is returned by the SetupsHP54600Scope collection.

Object Name: SetupHP54600Scope
File Name: HP54600.OCX

Syntax

SetupHP54600Scope

Properties

Comments Property, **Model** Property, **Name** Property, **Version** Property.

Methods

GetFromInstrument Method, **IsLearnStringCompatible** Method, **ReadFromFile** Method, **SaveToFile** Method, **SendToInstrument** Method.

Remarks

- An error is generated if the setup being used by the SetupHP54600Scope object does not agree with the version and model of the oscilloscope.
- Use the Setups property page to save oscilloscope settings to a file during design time. Use the SaveToFile method to save oscilloscope settings during run time.

Using the collection to save oscilloscope settings.

You can save oscilloscope settings as a collection by either:

- Using the Add method of the SetupsHP54600Scope collection during run time. Settings saved as a collection during run time will be lost when the code terminates.
- Using the property pages of the HP54600Scope Control during design time. When using the property pages, the oscilloscope settings are saved as part of the project.

Comments Property

Applies to: SetupHP54600 Object

Sends or retrieves user comments to the oscilloscope setup in the SetupHP54600Scope object.

Syntax

object.**Comments** [(*comments*)]

Data Type

- String
- Default = ""

Settings

- *comments* As String contains an arbitrary string.

Remarks

- This property can be used to time-stamp the oscilloscope setup.
- These comments can also be set or read using the Setups property page.
- The "|" character is not allowed in *comments*.

GetFromInstrument Method

Applies to: SetupHP54600 Object

Retrieves an oscilloscope setup from the instrument and places it in the SetupHP54600Scope object.

Syntax

object.**GetFromInstrument**

IsLearnStringCompatible Method

Applies to: SetupHP54600 Object

Returns a Boolean to indicate the compatibility with a specified instrument.

Syntax

object.**IsLearnStringCompatible** (*model*, *version*)

Data Type

- Boolean

Settings

- *model* As String contains the instrument model of interest.
- *version* As String contains the instrument version of interest.

Remarks

- Returns True if the specified instrument is compatible, returns False otherwise.

Model Property

Applies to: SetupHP54600 Object

Returns the instrument model number. Read-Only.

Syntax

object.**Model**

Data Type

- String
- Default = ""

Remarks

- The Model is updated when the GetFromInstrument method is executed.
- This property returns the same string as the InstrumentModel property. For example: "54602B".
- The Model property and Version property of a setup must be the same as the target instrument's model number and version.

Name Property

Applies to: SetupHP54600 Object

Returns the name of the object in the SetupsHP54600Scope collection. Read-Only.

Syntax

object.**Name**

Data Type

- String
- Default = ""

ReadFromFile Method

Applies to: SetupHP54600 Object

Retrieves an oscilloscope setup from a file and places the setup in the SetupHP54600Scope object.

Syntax

object.**ReadFromFile** (*filename*)

Settings

- *filename* As String contains a valid path and filename.

Remarks

- This method does not send the setup to the instrument.

SaveToFile Method

Applies to: SetupHP54600 Object

Saves an oscilloscope setup to a specified file and path name.

Syntax

object.**SaveToFile** (*filename*)

Settings

- *filename* As String contains a path and file name.

SendToInstrument Method

Applies to: SetupHP54600 Object

Sends an oscilloscope setup from the SetupHP54600Scope object to the instrument.

Syntax

object.**SendToInstrument**

Remarks

- This method verifies the model and version of the instrument before sending the setup to the instrument.

Version Property

Applies to: SetupHP54600 Object

Returns the instrument version. Read-Only.

Syntax

object.**Version**

Data Type

- String
- Default = ""

Remarks

- The Version is updated when the GetFromInstrument method is executed.
- The Model property and Version property of a setup must be the same as the target instrument's model number and version.

TimebaseHP54600 Object

Applies to: HP54600Scope and HP54600EZ

Use the TimebaseHP54600 Object to:

- Get or set the horizontal timebase of the oscilloscope.
- Get or set the sweep mode of the oscilloscope.
- Get or set the sweep reference point.

Properties

HorizontalDelay Property, **HorizontalRange** Property, **Mode** Property, **Reference** Property.

HorizontalDelay Property

Applies to: TimebaseHP54600 Object

Gets/sets the timebase delay. This delay is the oscilloscope's internal time between the trigger event and the on-screen delay reference point in seconds. The delay reference point is set with the Reference property.

Syntax

object.**HorizontalDelay** [= *value*]

Data Type

- Double

Settings

- *value* As Double is the desired delay in seconds.

HorizontalRange Property

Applies to: TimebaseHP54600 Object

Gets/sets the full-scale horizontal time of the display screen in seconds.

Syntax

object.**HorizontalRange** [= *value*]

Data Type

- Double

Settings

- *value* As Double is the desired horizontal time in seconds.

Remarks

- This property sets the full-scale horizontal range; the oscilloscope front panel settings set the horizontal range per division.

Mode Property

Applies to: TimebaseHP54600 Object

Gets/sets the sweep timebase mode of the oscilloscope.

Syntax

object.**Mode** [= *sweep*]

Data Type

- HP54600_TimeMode (Enumeration)

Settings

- *sweep* As HP54600_TimeMode sets the sweep to normal, delayed, XY, or roll.

Reference Property

Applies to: TimebaseHP54600 Object

Gets/sets the display reference. The reference can be set to one division from the left (or right) side of screen, or to the center of the screen.

Syntax

object.**Reference** [= *timeref*]

Data Type

- HP54600_TimeReference (Enumeration)

Settings

- *timeref* As HP54600_TimeReference sets the reference to left, center, or right.

Remarks

- HP54600_TimeReference_Right is only valid when the Mode Property is set to HP54600_TimeMode_Roll.

TraceHP54600 Object

Applies to: HP54600Scope and HP54600EZ

Use the TraceHP54600 object to:

- Display trace memory on the oscilloscope.
- Get and save trace memories.
- Work with oscilloscope trace memories.

Properties

Enabled Property, **TotalMemories** Property.

Methods

Clear Method, **GetData** Method, **PutData** Method, **Save** Method.

Clear Method

Applies to: TraceHP54600 Object

Clears the contents of the specified trace memory.

Syntax

object.**Clear** (*tracenumber*)

Settings

- *tracenumber* As Long is the trace to clear. *TraceNumber* can range from 1 to the maximum number of available trace memories.

Remarks

- Obtain the number of trace memories using the TotalMemories property.

Enabled Property

Applies to: TraceHP54600 Object

Enables/disables the displaying of the specified trace memory on the oscilloscope's display screen.

Syntax

object.**Enabled** (*tracenumber*) [= {True | False}]

Data Type

- Boolean

Settings

- *tracenumber* As Long is the trace to clear. *tracenumber* can range from 1 to the maximum number of available trace memories.

Remarks

- Obtain the number of trace memories using the TotalMemories property.
- The state of this property can be set or changed manually using the front panel of the oscilloscope.

GetData Method

Applies to: TraceHP54600 Object

Gets the trace data from the specified trace memory of the instrument and returns it in the specified buffer.

Syntax

object.**GetData** (*tracenumber*, *tracedata*)

Settings

- *tracenumber* As Long specifies which trace to get.
- *tracedata* As Variant contains the returned data.

Remarks

- Obtain the number of active trace memories using the TotalMemories property.
- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.

PutData Method

Applies to: TraceHP54600 Object

Sends the specified trace data to instrument and saves it to the specified trace memory.

Syntax

object.**PutData** (*tracenumber*, *tracedata*)

Settings

- *tracenumber* As Long is the trace number to write to.
- *tracedata* As Variant contains the data to put in the trace memory.

Remarks

- Obtain the number of active trace memories using the TotalMemories property.
- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.

Save Method

Applies to: TraceHP54600 Object

Saves the currently displayed waveform to the specified trace memory.

Syntax

object.**Save** (*tracenumber*)

Settings

- *tracenumber* As Long specifies the trace memory to save.

Remarks

- Obtain the number of active trace memories using the TotalMemories property.

TotalMemories Property

Applies to: TraceHP54600 Object

Returns the total number of trace memories for the instrument and/or storage module.

Syntax

object.**TotalMemories**

Data Type

- Long

TriggerHP54600 Object

Applies to: HP54600Scope and HP54600EZ

Use the TriggerHP54600 Object to work with the oscilloscope's trigger system.

Properties

Coupling Property, **Mode** Property, **Slope** Property, **Source** Property, **TriggerLevel** Property.

Coupling Property

Applies to: TriggerHP54600 Object

Gets/sets the input coupling for the selected external trigger source.

Syntax

object.**Coupling** [= *acdc*]

Settings

- *acdc* As HP54600_TriggerCoupling to either AC or DC.

Remarks

- The trigger source must be set to "external" using the Source property.

Mode Property

Applies to: TriggerHP54600 Object

Gets/sets the triggering mode.

Syntax

object.**Mode** [= *trigger*]

Settings

- *trigger* As HP54600_TriggerMode sets AutoLevel, Auto, or Normal triggering.

Slope Property

Applies to: TriggerHP54600 Object

Gets/sets the slope of the edge for the trigger.

Syntax

object.**Slope** [= *edge*]

Settings

- *edg*e As HP54600_TriggerSlope sets either positive or negative trigger slopes.

Source Property

Applies to: TriggerHP54600 Object

Gets/sets the channel used for the trigger.

Syntax

object.**Source** [= *trigsource*]

Settings

- *trigsource* As HP54600_TriggerSource sets a source to be a channel, external, line, or a logic bit.

TriggerLevel Property

Applies to: TriggerHP54600 Object

Gets/sets the trigger level voltage for the active trigger.

Syntax

object.**TriggerLevel** [= *value*]

Settings

- *value* As Long is the value, in volts, of the triger level.

Data Type

- Double

UtilitiesHP54600 Object

Applies to: HP54600Scope and HP54600EZ

Use the UtilitiesHP54600 Object to:

- Work with the oscilloscope connection to the PC.
- Work with the oscilloscope descriptions and identity.
- Initialize a session with the oscilloscope.
- Work with the automation server from the **HP54600Scope Control**.

Properties

ComponentDescription Property, **ComponentManufacturer** Property, **ComponentProgID** Property, **ComponentVersion** Property, **DetectDeviceErrors** Property, **Initialize** Property, **InitializeIO** Property, **InstanceName** Property, **InstrumentFirmwareVersion** Property, **InstrumentManufacturer** Property, **InstrumentModel** Property, **InstrumentSerialNumber** Property, **IO** Property, **LogInterface** Property, **PanelLock** Property, **RangeChecking** Property, **Timeout** Property.

Methods

ClearDevice Method, **ClearStatus** Method, **DisplayMessage** Method, **GetOption** Method, **Options** Method, **Preset** Method, **QueryInstrumentError** Method, **ReadStateData** Method, **RecallState** Method, **Reset** Method, **SaveState** Method, **SelfTest** Method, **StatusBits** Method, **VerifyDevice** Method, **WriteStateData** Method .

ClearDevice Method

Applies to: UtilitiesHP54600 Object

Performs a device clear of the instrument.

Syntax

object.**ClearDevice**

ClearStatus Method

Applies to: UtilitiesHP54600 Object

Clears the instrument's status registers.

Syntax

object.**ClearStatus**

ComponentDescription Property

Applies to: UtilitiesHP54600 Object

Returns a description of the automation server. Read-Only.

Syntax

object.**ComponentDescription**

Data Type

- String

ComponentManufacturer Property

Applies to: UtilitiesHP54600 Object

Returns the Component Manufacturer/Developer name. Read-Only.

Syntax

object.**ComponentManufacturer**

Data Type

- String

ComponentProgID Property

Applies to: UtilitiesHP54600 Object

Returns the component Program ID of the automation server. Read-Only.

Syntax

object.**ComponentProgID**

Data Type

String

Remarks

- The program ID is placed in the system registry as a cross-reference to the GUID.

ComponentVersion Property

Applies to: UtilitiesHP54600 Object

Returns the version of the automation server. Read-Only.

Syntax

object.**ComponentVersion**

Data Type

- String

DetectDeviceErrors Property

Applies to: UtilitiesHP54600 Object

Enables/disables the device errors debugging function.

Syntax

object.**DetectDeviceErrors** [= {True | False}]

Data Type

Boolean

Settings

- {True | False}
- Default = FALSE

Remarks

- When DeviceDetectErrors is set to TRUE, the driver polls the instrument after any property or method that uses I/O to see if an error occurred. If an error is detected, the property or method will return an appropriate error.
- When DeviceDetectErrrors is FALSE, the driver will not check for instrument errors. If an I/O operation results in an instrument error, it will go undetected until either:
- DeviceDetectErrors is set to TRUE, or
- The QueryInstrumentError method is used.

DisplayMessage Method

Applies to: UtilitiesHP54600 Object

Writes the specified message string to the instrument's display.

Syntax

object.**DisplayMessage** (*message*)

Settings

- *message* As String is an arbitrary string.

GetOption Method

Applies to: UtilitiesHP54600 Object

Verifies if a specific option is installed.

Syntax

object.**GetOption** (*name*, *autodetected*)

Data Type

- Boolean

Settings

- *name* As String is the option name to verify.
- *autodetected* As Boolean indicates if the option to verify is detected automatically or was placed in the options list (typically by the PutOption method).

About the Initialize and InitializeIO Properties

Utilities (UtilitiesHP54600)

Initialize (IInitializeDevice)

InitializeI/O (IInitialize)

The Initialize Property and InitializeIO Property and their referenced objects (IInitializeDevice and IInitialize) are designed for use in complex test systems with multiple instruments and automation servers. For single instrument systems, the Connect Method is easier and simpler to use.

Initialize Property

Applies to: UtilitiesHP54600 Object

Returns an IInitializeDevice interface of the automation server. Read-Only.

Syntax

object.**Initialize**

Data Type

IInitializeDevice

Remarks

- See additional information about the Initialize Property.

InitializeIO Property

Applies to: UtilitiesHP54600 Object

Returns a pointer to the I/O component's Initialize interface. Read-Only.

Syntax

object.**InitializeIO**

Data Type

IInitialize

Remarks

- If an I/O component has been loaded, InitializeIO returns a reference to the I/O component's IInitialize object. If an I/O component has not been loaded, InitializeIO returns a null reference.
- See additional information about the InitializeIO Property.

InstanceName Property

Applies to: UtilitiesHP54600 Object

Gets/sets the name of this instance of the automation server.

Syntax

object.**InstanceName** [= *value*]

Data Type

- String

Settings

- *value* As String may be set to any arbitrary string.

Remarks

- The instance name is used for logging and other features that require object identification.

InstrumentFirmwareVersion Property

Applies to: UtilitiesHP54600 Object

Returns the instrument's version. Read-Only.

Syntax

object.**InstrumentFirmwareVersion**

Data Type

- String

Remarks

- The instrument version is returned as a part of the instrument's response to the IEEE488.2 *IDN? Query.

InstrumentManufacturer Property

Applies to: UtilitiesHP54600 Object

Returns the name of the instrument's manufacturer. Read-Only.

Syntax

object.**InstrumentManufacturer**

Data Type

- String

InstrumentModel Property

Applies to: UtilitiesHP54600 Object

Returns the instrument's model number. Read-Only

Syntax

object.**InstrumentModel**

Data Type

- String

Remarks

- This property returns the same string as the Model property. For example: "54602B".

InstrumentSerialNumber Property

Applies to: UtilitiesHP54600 Object

Returns the instrument's serial number. This may not be supported by all instruments. Read-Only.

Syntax

object.**InstrumentSerialNumber**

Data Type

- String

IO Property

Applies to: UtilitiesHP54600 Object

Returns an object that provides an interface to the underlying I/O automation server. Read-Only.

Syntax

object.**IO**

Remarks

- Returns the IIO object from the 'IIO Manager and Utilities' automation server.

LogInterface Property

Applies to: UtilitiesHP54600 Object

Enables/disables the error logging facility.

Syntax

object.**LogInterface** [= {True | False}]

Data Type

- Boolean

Settings

- True enables error logging, False disables error logging. Default = False.

Remarks

- When enabled, the WriteLog Events are sent to clients registered In Visual Basic (using the WithEvents option).

Options Method

Applies to: UtilitiesHP54600 Object

Returns a variant containing a list of the options installed in the currently connected instrument.

Syntax

object.**Options**

Remarks

- Each variant in the returned array contains a string describing the option. Possible options for the HP 54600-Series of oscilloscopes include:
  "Basic Interface Module (Option 0)"
  "Test Automation (Option 1)"
  "Measurement/Storage Module (Option 2)"
  "Enhanced Video Trigger (Option 5)"
  "RS-232 Communication"
  "HP-IB Communication"
  "Basic Interface Module"

PanelLock Property

Applies to: UtilitiesHP54600 Object

Enables/disables the locking out of the instrument's front panel.

Syntax

object.**PanelLock** [= {True | False}]

Data Type

- Boolean

Settings

- True enables the panel lock, False disables the panel lock. Default = False.

Preset Method

Applies to: UtilitiesHP54600 Object

Performs a preset of the instrument.

Syntax

object.**Preset**

QueryInstrumentError Method

Applies to: UtilitiesHP54600 Object

Returns a number and a string that report the most recent error in the instrument's error queue.

Syntax

object.**QueryInstrumentError** (*errornumber*, *errordescription*)

Data Type

- *errornumber* As Long returns the instrument error number.
- *errordescription* As String returns the instrument error string.

Remarks

- The instrument errors are reported as a signed number and, on some instruments, a descriptive string. Refer to the instrument's documentation for a complete list of error codes.
- Instruments with no errors in the error queue return 0,"No Error".
- Errors are reported on a first-in, first-out basis.
- You can empty the error queue without reading all the errors using the ClearStatus method.

RangeChecking Property

Applies to: UtilitiesHP54600 Object

Enables/disables input value range checking.

Syntax

object.**RangeChecking** [= {True | False }]

Data Type

- Boolean

Settings

- True enables range checking, False disables range checking.
- Default = True

Remarks

- Use this property for program development and debugging.
- When enabled, values are checked for applicability to the oscilloscope model. Invalid ranges are reported before the I/O operation executes.
- The range checking operation slows program execution. For fastest program execution set RangeChecking to False, DetectDeviceErrors to False and LogInterface to False.

ReadStateData Method

Applies to: UtilitiesHP54600 Object

Reads the current system setup data from the instrument. The system setup data is encoded in the binary IEEE 488.2 definite block format. The data within the definite block is encoded in the instrument's internal format.

Syntax

object.**ReadStateData**

Data Type

- Variant

Remarks

- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.

RecallState Method

Applies to: UtilitiesHP54600 Object

Recalls the instrument state from its specified internal register and updates the current instrument state.

Syntax

object.**RecallState** (*statenumber*)

Settings

- *statenumber* As Long is the state register to recall.

Reset Method

Applies to: UtilitiesHP54600 Object

Performs a reset of the instrument.

Syntax

object.**Reset**

SaveState Method

Applies to: UtilitiesHP54600 Object

Saves the current instrument state to its specified internal register.

Syntax

object.**SaveState** (*statenumber*)

Settings

- *statenumber* As Long is the internal register number to use.

SelfTest Method

Applies to: UtilitiesHP54600 Object

Performs a self-test on the connected instrument and returns results of the test.

Syntax

object.**SelfTest** (*testresult*, *resultmessage*)

Data Types

- *testresult* As Long returns a number indicating the self-test status.
- *resultmessage* As String returns a string indicating the self-test result.

Remarks

- The instrument returns a "0" and an empty string if the self-test passes.
- If the self-test has a failure, the instrument returns a value and a string indicating the nature of the failing test. Some instruments return only the *testresult* value and *resultmessage* is set to null. Refer to the instrument documentation for results returned by the *TST command (IEEE 488.2 common command).
- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.

StatusBits Method

Applies to: UtilitiesHP54600 Object

Returns the contents of the 'Status Events' and 'Service Request' registers of the instrument. This is a destructive read operation. Status Event register is returned in lower word. Service Request register is returned in upper word.

Syntax

object.**StatusBits**

Data Type

- Long

Timeout Property

Applies to: UtilitiesHP54600 Object

Gets/sets the automation server's I/O timeout value in milliseconds.

Syntax

object.**Timeout** [= *value*]

Data Type

- Long

Settings

- *value* As Long sets the timeout in milliseconds. For example, setting *value* to 1000 sets a 1-second timeout value. Default = 5000.

Remarks

- For example, use the following statement to set a 5 second timeout.

```
HP54600Scope1.Timeout = 5000
```

VerifyDevice Method

Applies to: UtilitiesHP54600 Object

Verifies, by returning a Boolean operator, that the connected instrument is compatible with the specified parameters.

Syntax

object.**VerifyDevice** (*model*, [*version*], [*serialnumber*])

Data Type

- Boolean

Settings

- *model* As String is the only required parameter.
- *version* As String is an optional parameter.
- *serialnumber* As String is an optional parameter.

WriteStateData Method

Applies to: UtilitiesHP54600 Object

Writes system setup data in the specified buffer to the instrument and updates current system setup. See the ReadStateData method.

Syntax

object.**WriteStateData** (*statedata*)

Settings

- *statedata* As Variant is a buffer containing the state data.

Remarks

- This method extends the timeout value in effect to allow for execution and then restores the timeout value to the one set by the Timeout property.

Constants

Oscilloscope Models and the Number of Points

The number of data points retrieved by the GetAllWaveformData, GetWaveformData, and GetLogicData methods varies by the oscilloscope model number.

Points setting	HP 54600B, HP 54602B, HP 54603B, HP 54610B	HP 54615B, HP 54616B/C	HP 54620A/C	HP 54645A/D (analog channels)	HP 54645D (digital channel)
100	100		128	100	100
200	200				
250	250	250	256	250	250
500	500		512	500	500
1000	1000	1000	1024	1000	
2000	2000		2048	2000	
All	Up to 2000	1000	Up to 8192	Up to 1M	Up to 2M

Remarks

For the HP 54645D, select either analog or digitial channels when using GetAllWaveform-Data to retrieve all data points.

HP54600AnalogChannels Constants

Used with: AverageVoltage Method, FallTime Method, Frequency Method, MaxVoltage Method , MinVoltage Method , NegativePulseWidth Method, PeakToPeakVolts Method, Period Method, PostivePulseWidth Method, RiseTime Method, and RMSVoltage Method

Value	Constant
1	HP54600_Channel_1
2	HP54600_Channel_2
3	HP54600_Channel_3
4	HP54600_Channel_4

HP54600_AnalogChannels Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
1	✓	✓	✓	✓	✓	✓	×	✓
2	✓	✓	✓	✓	✓	✓	×	✓
3	×	✓	×	×	×	×	×	×
4	×	✓	×	×	×	×	×	×

× = value not supported by this model.

✓ = value is supported by this model.

HP54600AcquisitionCount Constants

Used with: Count Property

Value	Constant
1	HP54600_AcquisitionCount_1
4	HP54600_AcquisitionCount_4
8	HP54600_AcquisitionCount_8
16	HP54600_AcquisitionCount_16
32	HP54600_AcquisitionCount_32
64	HP54600_AcquisitionCount_64
128	HP54600_AcquisitionCount_128
256	HP54600_AcquisitionCount_256

HP54600_AcquisitionCount Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
1	✖	✖	✖	✖	✖	✖	✖	✔
4	✖	✖	✖	✖	✖	✖	✖	✓
8	✓	✓	✓	✓	✔	✔	✖	✓
16	✖	✖	✖	✖	✖	✖	✖	✓
32	✖	✖	✖	✖	✖	✖	✖	✓
64	✓	✔	✔	✓	✔	✔	✓	✓
128	✖	✖	✖	✖	✖	✖	✖	✓
256	✓	✓	✓	✓	✓	✓	✖	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600AcquisitionType Constants

Used with: Type property

Value	Constant
1	HP54600_AcquisitionType_Average
2	HP54600_AcquisitionType_Peak
3	HP54600_AcquisitionType_Glitch
4	HP54600_AcquisitionType_Real
5	HP54600_AcquisitionType_Auto
6	HP54600_AcquisitionType_Normal

HP54600_AcquisitionType Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
Normal	✓	✓	✓	✓	✓	✓	✓	✓
Average	✓	✓	✓	✓	✓	✓	×	✓
Peak	✓	✓	✓	✓	✓	✓	×	✓
Glitch	×	×	×	×	×	×	✓	×
Real	×	×	×	×	×	×	×	✓
Auto	×	×	×	×	×	×	✓	×

× = value not supported by this model.

✓ = value is supported by this model.

HP54600AllChannels Constants

Used with: DutyCycle Method, Frequency Method, NegativePulseWidth Method, Period Method, and PostivePulseWidth Method

Value	Constant
1	HP54600_Channel_A1
2	HP54600_Channel_A2
3	HP54600_Channel_A3
4	HP54600_Channel_A4
10	HP54600_Channel_D0
11	HP54600_Channel_D1
12	HP54600_Channel_D2
13	HP54600_Channel_D3
14	HP54600_Channel_D4
15	HP54600_Channel_D5
16	HP54600_Channel_D6
17	HP54600_Channel_D7
18	HP54600_Channel_D8
19	HP54600_Channel_D9
20	HP54600_Channel_D10
21	HP54600_Channel_D11
22	HP54600_Channel_D12
23	HP54600_Channel_D13
24	HP54600_Channel_D14
25	HP54600_Channel_D15

HP54600_AllChannels Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A	54645D
A1	✓	✓	✓	✓	✓	✓	✖	✓	✓
A2	✓	✓	✓	✓	✓	✓	✖	✓	✓
A3	✖	✓	✖	✖	✖	✖	✖	✖	✖
A4	✖	✓	✖	✖	✖	✖	✖	✖	✖
D0	✖	✖	✖	✖	✖	✖	✓	✖	✓
D1	✖	✖	✖	✖	✖	✖	✓	✖	✓
D2	✖	✖	✖	✖	✖	✖	✓	✖	✓
D3	✖	✖	✖	✖	✖	✖	✓	✖	✓
D4	✖	✖	✖	✖	✖	✖	✓	✖	✓
D5	✖	✖	✖	✖	✖	✖	✓	✖	✓
D6	✖	✖	✖	✖	✖	✖	✓	✖	✓
D7	✖	✖	✖	✖	✖	✖	✓	✖	✓
D8	✖	✖	✖	✖	✖	✖	✓	✖	✓
D9	✖	✖	✖	✖	✖	✖	✓	✖	✓
D10	✖	✖	✖	✖	✖	✖	✓	✖	✓
D11	✖	✖	✖	✖	✖	✖	✓	✖	✓
D12	✖	✖	✖	✖	✖	✖	✓	✖	✓
D13	✖	✖	✖	✖	✖	✖	✓	✖	✓
D14	✖	✖	✖	✖	✖	✖	✓	✖	✓
D15	✖	✖	✖	✖	✖	✖	✓	✖	✓

✖ = value not supported by this model.

✓ = value is supported by this model.

HP54600CountryCode Constants

Used with: CountryCode Property

Value	Constant
1	HP54600_Language_English
33	HP54600_Language_French
34	HP54600_Language_Spanish
39	HP54600_Language_Italian
49	HP54600_Language_German
82	HP54600_Language_Korean
886	HP54600_Language_Traditional_Chinese

HP54600ImageFormat Constants

Used with: GetScreenImage Method and SaveScreenImage Method

Value	Constant
0	HP54600_ImageFormat_BMP
2	HP54600_ImageFormat_HiResBMP

HP54600LogicBits Constants

Used with: BitEnabled Property

Value	Constant
0	HP54600_Logic_D0
1	HP54600_Logic_D1
2	HP54600_Logic_D2
3	HP54600_Logic_D3
4	HP54600_Logic_D4
5	HP54600_Logic_D5
6	HP54600_Logic_D6
7	HP54600_Logic_D7
8	HP54600_Logic_D8
9	HP54600_Logic_D9
10	HP54600_Logic_D10
11	HP54600_Logic_D11
12	HP54600_Logic_D12
13	HP54600_Logic_D13
14	HP54600_Logic_D14
15	HP54600_Logic_D15

HP54600_LogicBits Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A	54645D
D0	✖	✖	✖	✖	✖	✖	✓	✖	✓
D1	✖	✖	✖	✖	✖	✖	✓	✖	✓
D2	✖	✖	✖	✖	✖	✖	✓	✖	✓
D3	✖	✖	✖	✖	✖	✖	✓	✖	✓
D4	✖	✖	✖	✖	✖	✖	✓	✖	✓
D5	✖	✖	✖	✖	✖	✖	✓	✖	✓
D6	✖	✖	✖	✖	✖	✖	✓	✖	✓
D7	✖	✖	✖	✖	✖	✖	✓	✖	✓
D8	✖	✖	✖	✖	✖	✖	✓	✖	✓
D9	✖	✖	✖	✖	✖	✖	✓	✖	✓
D10	✖	✖	✖	✖	✖	✖	✓	✖	✓
D11	✖	✖	✖	✖	✖	✖	✓	✖	✓
D12	✖	✖	✖	✖	✖	✖	✓	✖	✓
D13	✖	✖	✖	✖	✖	✖	✓	✖	✓
D14	✖	✖	✖	✖	✖	✖	✓	✖	✓
D15	✖	✖	✖	✖	✖	✖	✓	✖	✓

✖ = value not supported by this model.

✓ = value is supported by this model.

HP54600LogicPods Constants

Used with: PodEnable Property

Value	Constant
1	HP54600_Logic_POD1
2	HP54600_Logic_POD2

HP54600_LogicPods Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A	54645D
POD1	✖	✖	✖	✖	✖	✖	✓	✖	✓
POD2	✖	✖	✖	✖	✖	✖	✓	✖	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600OnOffState Constants

Used with: BandWidthLimit Property

Value	Constant
0	HP54600_Off
1	HP54600_On

HP54600ProbeAttenuation Constants

Used with: ProbeAttenuation Property

Value	Constant
1	HP54600_ProbeAttenuation_°1
10	HP54600_ProbeAttenuation_°10
20	HP54600_ProbeAttenuation_°20
100	HP54600_ProbeAttenuation_°100

HP54600_ProbeAttenuation Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
°1	✓	✓	✓	✓	✓	✓	✖	✓
°10	✓	✓	✓	✓	✓	✓	✖	✓
°20	✖	✖	✖	✓	✓	✓	✖	✓
°100	✓	✓	✓	✓	✓	✓	✖	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600ProbeCoupling Constants

Used with: ProbeCoupling Property

Value	Constant
1	HP54600_ProbeCoupling_DC
2	HP54600_ProbeCoupling_GND
3	HP54600_ProbeCoupling_AC

HP54600_ProbeCoupling Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
AC	✓	✓	✓	✓	✓	✓	✖	✓
DC	✓	✓	✓	✓	✓	✓	✖	✓
GND	✓	✓	✓	✓	✓	✓	✖	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600ProbeMode Constants

Used with: ProbeMode Property

Value	Constant
1	HP54600_ProbeMode_Manual
2	HP54600_ProbeMode_Auto

HP54600_ProbeMode Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
Auto	✖	✖	✖	✓	✓	✓	✖	✓
Manual	✖	✖	✖	✓	✓	✓	✖	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600ShowPropertyPage Constants

Used with: ShowPropertyPages Method

Value	Constant
0	HP54600_ShowPropertyPage_Default
1	HP54600_ShowPropertyPage_Search
2	HP54600_ShowPropertyPage_SetIO
3	HP54600_ShowPropertyPage_Setups

HP54600TimeMode Constants

Used with: Mode Property

Value	Constant
1	HP54600_TimeMode_Delayed
2	HP54600_TimeMode_°Y
3	HP54600_TimeMode_Roll
4	HP54600_TimeMode_Normal

HP54600_TimeMode Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
Normal	✓	✓	✓	✓	✓	✓	✓	✓
Delayed	✓	✓	✓	✓	✓	✓	✓	✓
°Y	✓	✓	✓	✓	✓	✓	✖	✓
Roll	✓	✓	✓	✓	✓	✓	✖	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600TimeReference Constants

Used with: Reference Property

Value	Constant
1	HP54600_TimeReference_Center
2	HP54600_TimeReference_Right
3	HP54600_TimeReference_Left

HP54600_TimeReference Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
Left	✓	✓	✓	✓	✓	✓	✓	✓
Center	✓	✓	✓	✓	✓	✓	✓	✓
Right	✓	✓	✓	✓	✓	✓	✓	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600TriggerCoupling Constants

Used with: Coupling Property

Value	Constant
1	HP54600_TriggerCoupling_DC
2	HP54600_TriggerCoupling_AC

HP54600_TriggerCoupling Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
AC	✓	✓	✓	✓	✓	✓	✖	✓
DC	✓	✓	✓	✓	✓	✓	✖	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600TriggerMode Constants

Used with: Mode Property

Value	Constant
1	HP54600_TriggerMode_Auto
2	HP54600_TriggerMode_Normal
3	HP54600_TriggerMode_AutoLevel

HP54600_TriggerMode Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
AutoLevel	✓	✓	✓	✓	✓	✓	✓	✓
Auto	✓	✓	✓	✓	✓	✓	✓	✓
Normal	✓	✓	✓	✓	✓	✓	✓	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600TriggerSlope Constants

Used with: Slope Property

Value	Constant
1	HP54600_TriggerSlope_Positive
2	HP54600_TriggerSlope_Negative

HP54600_TriggerSlope Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A/D
Negative	✓	✓	✓	✓	✓	✓	✖	✓
Positive	✓	✓	✓	✓	✓	✓	✖	✓

✖ = value not supported by this model.
✓ = value is supported by this model.

HP54600TriggerSource Constants

Used with: Source Property

Value	Constant
1	HP54600_TriggerSource_Channel1
2	HP54600_TriggerSource_Channel2
3	HP54600_TriggerSource_Channel3
4	HP54600_TriggerSource_Channel4
5	HP54600_TriggerSource_E°ternal
6	HP54600_TriggerSource_Line
10	HP54600_TriggerSource_Logic_D0
11	HP54600_TriggerSource_Logic_D1
12	HP54600_TriggerSource_Logic_D2
13	HP54600_TriggerSource_Logic_D3
14	HP54600_TriggerSource_Logic_D4
15	HP54600_TriggerSource_Logic_D5
16	HP54600_TriggerSource_Logic_D6
17	HP54600_TriggerSource_Logic_D7
18	HP54600_TriggerSource_Logic_D8
19	HP54600_TriggerSource_Logic_D9
20	HP54600_TriggerSource_Logic_D10
21	HP54600_TriggerSource_Logic_D11
22	HP54600_TriggerSource_Logic_D12
23	HP54600_TriggerSource_Logic_D13
24	HP54600_TriggerSource_Logic_D14
25	HP54600_TriggerSource_Logic_D15

HP54600_TriggerSource Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A	54645D
A1	✓	✓	✓	✓	✓	✓	×	✓	✓
A2	✓	✓	✓	✓	✓	✓	×	✓	✓
A3	×	✓	×	×	×	×	×	×	×
A4	×	✓	×	×	×	×	×	×	×
Exter-nal	✓	×	✓	✓	✓	✓	×	✓	×
Line	✓	×	✓	✓	✓	✓	×	✓	✓
D0	×	×	×	×	×	×	×	×	✓
D1	×	×	×	×	×	×	×	×	✓
D2	×	×	×	×	×	×	×	×	✓
D3	×	×	×	×	×	×	×	×	✓
D4	×	×	×	×	×	×	×	×	✓
D5	×	×	×	×	×	×	×	×	✓
D6	×	×	×	×	×	×	×	×	✓
D7	×	×	×	×	×	×	×	×	✓
D8	×	×	×	×	×	×	×	×	✓
D9	×	×	×	×	×	×	×	×	✓
D10	×	×	×	×	×	×	×	×	✓
D11	×	×	×	×	×	×	×	×	✓
D12	×	×	×	×	×	×	×	×	✓
D13	×	×	×	×	×	×	×	×	✓
D14	×	×	×	×	×	×	×	×	✓
D15	×	×	×	×	×	×	×	×	✓

× = value not supported by this model.

✓ = value is supported by this model.

HP54600WaveformPoints Constants

Used with: GetAllWaveformData Method, GetLogicData Method, and GetWaveformData Method

Value	Constant
0	HP54600_DataPoints_ALL
100	HP54600_DataPoints_100
250	HP54600_DataPoints_250
500	HP54600_DataPoints_500
1000	HP54600_DataPoints_1000
2000	HP54600_DataPoints_2000

HP54600_WaveformPoints Supported Models

	54600B	54602B	54603B	54610B	54615B	54616B/C	54620A/C	54645A	54645D
100	✓	✓	✓	✓	✖	✖	✓	✓	✓
250	✓	✓	✓	✓	✓	✓	✓	✓	✓
500	✓	✓	✓	✓	✖	✖	✓	✓	✓
1000	✓	✓	✓	✓	✓	✓	✓	✓	✓✉
2000	✓	✓	✓	✓	✖	✖	✓	✓	✓✉
All	✓	✓	✓	✓	✓	✓	✓	✓	✓

✖ = value not supported by this model.

✓ = value is supported by this model.

* Only valid for analog channels (do not use for digital channels).

LogType Constants

Used with: WriteLog Event

Value	Constant
0	LogType_Error
1	LogType_Trace
2	LogType_Warning

This constant defines the types of logging events sent by the ILog interface with the Write-Log event.

Property Pages

Search Instrument Property Page

Applies to: HP54600Scope

Use the 'Search Instruments' property page to locate instruments connected to your PC over the GPIB ports or RS-232 (COM) ports. The search will also identify, by model number and manufacturer name, those instruments that are IEEE 488.2 compliant.

Prerequisites

Connecting to the Oscilloscope To communicate with the oscilloscope from a PC, you must have either a GPIB or RS-232 interface module installed on the oscilloscope. You must also have the proper interface cable connected between the oscilloscope and your PC (for RS-232 communications, be sure to use the cable provided with the interface module -- HP Part Number 5182-4794). For communications over GPIB, you must have a GPIB card installed in your PC. For communications over RS-232, use the standard COM ports on your PC.

Verifying the Oscilloscope's Interface Settings Before attempting to make a connection, make sure that the oscilloscope is properly configured to communicate over the desired interface. From the front panel of the oscilloscope, press Print/Utility to display 'HP-IB Menu' or 'RS-232 Menu' and verify that the oscilloscope is configured for communication with a computer (the default is 'Printer' on some instruments).

Verifying the RS-232 Settings For proper operation, the oscilloscope's RS-232 settings (baud rate and handshake) must match the RS-232 settings on the 'Search Criteria' property page. From the front panel of the oscilloscope, press Print/Utility to display 'RS-232 Menu' and view the current settings. Note that the handshake mode on the oscilloscope MUST be set to 'DTR'.

List of Instruments Found

Click **Search** to find all instruments connected to your PC on the GPIB ports and RS-232 ports. All instruments found are added to the list. Supported HP 54600-Series oscilloscopes are shown in the list with an 'X' next to the model number. The active (connected) instrument is indicated with the icon.

Note: If the oscilloscope is configured to communicate with a printer, the search will find the oscilloscope but the model number and manufacturer will be listed as "Unknown".

The first time that you use the 'Search Instruments' property page, the following GPIB address is used:

```
"GPIB0::7"
```

From the list of instruments found, highlight the desired instrument and click **Apply** or **OK** to establish a connection. Click **Test** to verify the connection.

- [Search]
- Searches the GPIB ports and RS-232 ports for installed instruments. Supported HP 54600-Series oscilloscopes are shown in the list with an 'X' next to the model number. The active (connected) instrument is indicated with the icon.
- [Test]
- Tests the connection to the instrument currently selected from the list. The instrument is queried for its model number and manufacturer name. The response from the instrument and the result from the test are displayed.
- **[Apply]** and **[OK]**
- Click **Apply** to connect to the instrument currently highlighted (the property page remains open). Click **OK** to connect to the instrument currently highlighted and close the property page.

Note: When you select one of the other property page tabs ('Search Criteria', 'Set I/O', etc.), an implicit **Apply** is executed and a connection is made to the highlighted instrument. Also, if you go to the 'Set I/O' property page, make changes, and then go back to the 'Search Instruments' property page, the list of instruments will be cleared. Press **Search** from the 'Search Instruments' property page to perform a new search.

Search Criteria Property Page

Applies to: HP54600Scope

Use the 'Search Criteria' property page to select which GPIB ports and RS-232 ports will be included in the search and to set the RS-232 parameters (baud rate, parity, and handshake).

[Find Ports]

Searches the PC for registered GPIB ports and RS-232 ports. Click **Find Ports** if there are no ports currently shown in the 'Exclude' or 'Include' lists. Ports shown in the 'Include' list will be included in the search. To speed the search or to avoid errors in the search, you can move specific ports (such as a COM port connected to a modem) to the 'Exclude' list.

Note: If multiple instruments are found at the same GPIB address, you must resolve the address conflict and click **Find Ports** again.

[Add] and [Remove]

To move a port from the 'Include' list to the 'Exclude' list, select the desired port and click **Remove**. Similarly, to move a port from the 'Exclude' list to the 'Include' list, select the desired port and click **Add**.

- 'Exclude' List
- Ports in this list were found in the PC but will NOT be included in the search performed from the 'Search Instruments' property page.
- 'Included' List
- Ports in this list were found in the PC and will be included in the search performed from the 'Search Instruments' property page.

Excluding a Specific GPIB Instrument Address

In addition to excluding a specific GPIB port from the search, you can also choose to exclude specific instrument addresses. First, select the desired GPIB port from the 'Include' or 'Exclude' list. Then, from the address list at the bottom of the property page, select the addresses to be included in the search. Click on the check box to add the address or remove the check to exclude the address.

Setting COM (RS-232) Parameters

To set the RS-232 parameters (baud rate, parity, and handshake), first select the desired COM port from the 'Include' or 'Exclude' list. The parameters for the selected port will appear at the bottom of the property page.

For proper operation, the oscilloscope's RS-232 settings (baud rate and handshake) must match the RS-232 settings on the 'Search Criteria' property page. From the front panel of the oscilloscope, press the Print/Utility button to display 'RS-232 Menu' and view the current settings. Note that the handshake mode on the oscilloscope MUST be set to 'DTR'. Set the desired parameters and then click Apply or OK.

Set I/O Property Page

Applies to: HP54600Scope

Use the 'Set I/O' property page to set the I/O configuration manually. First, select the desired port from the list and then set the GPIB address or RS-232 setup parameters. Click **Apply** or **OK** to set the address properties.

I/O Port List

Lists all GPIB ports and RS-232 ports found in the PC.

Address

Displays the address for the selected I/O port.

[Test]

Tests the connection to the instrument currently selected from the list. The instrument is queried for its model number and manufacturer name. The response from the instrument and the result from the test are displayed.

GPIB and COM (RS-232) Setup

Use the drop-down lists at the bottom of this dialog box to set the specific interface parameters. For GPIB operation, the address selection is shown. For RS-232 operation, the baud rate, parity, and handshake selections are shown.

For proper operation, the oscilloscope's RS-232 settings must match the RS-232 settings on the 'Set I/O' property page. From the front panel of the oscilloscope, press the Print/Utility button to display 'RS-232 Menu' and view the current settings. Set the desired parameters and then click Apply or OK.

Note: If you make changes on the 'Set I/O' property page, the list of instruments will be cleared from the 'Search Instruments' property page unless you press **Test** to make the connection. Press **Search** from the 'Search Instruments' property page to perform a new search.

Setups Property Page

Applies to: HP54600Scope

Use the 'Setups' property page to retrieve settings from the oscilloscope, save the settings in memory, and send the settings back to the oscilloscope. You can store or retrieve the settings during design time or during run time (note that settings stored during design time are available to send during run time).

Alternately, you can store the settings to a file on your PC for later retrieval. The stored settings can also be used during run time to set the oscilloscope configuration. Settings stored from this property page are available to the SetupsHP54600Scope collection.

Note: An error may be generated if you attempt to download to a different oscilloscope than was used to create the original settings file.

Setups List

The setups list contains the names of the stored setups, with associated comments if used. The active setup is indicated with the icon.

[Get]

Retrieves the current settings from the oscilloscope and stores them in the active setup name (indicated with the icon).

[Send]

Downloads the selected setup to the oscilloscope.

[Open]

Retrieves a stored settings file and stores them in the active setup name (indicated with the icon). Note that you can also open a file stored from HP BenchLink Scope (.stp file extension).

[Save As]

Stores the selected settings to a file on your PC. The settings are stored in a binary format (.scp file extension).

[Add]

Adds a new setup to the setups list. Specify a setup name and add comments if desired. By default, 'Get Setup from Instrument' is checked on the 'Add New Setup' dialog box. To reserve a setup name without storing the settings from the oscilloscope, clear the check from before 'Get Setup from Instrument'.

[Remove]

Removes the selected setup.

Changing the Setup Properties

To rename a stored setup or to add additional comments, double-click the desired setup name from the list. Note that you can also copy the setup name to the clipboard for programming.

Getting Oscilloscope Settings

From the Setups Property page:

1 Click ‘Add’. The ‘Add New Setup’ dialog will appear.
2. Change the name and the comments as desired. The name is the name of the oscilloscope setup and is the same when called from code.
3. Click ‘OK’. If 'get setup from instrument' is checked, the instrument's settings are loaded into memory. The setup name is added to the list (representing the SetupsHP54600Scope collection). An icon next to name indicates that this is the setup that last agreed with the oscilloscope settings.

Sending a Saved Setup to the Oscilloscope

From the Setups Property Page:

1 Select a named setting from the list.
2. Click ‘Send’ to send the named settings to the oscilloscope.

Saving an Oscilloscope Setting to a File

From the Setups Property Page:

1 Select a named setting from the list.
2. Click ‘Save As’. A file dialog box will appear.
3. Select a location and a file name and click ‘Save’ to save the oscilloscope settings.

Loading an Oscilloscope Setup From a File

You can load an oscilloscope setup from a file to the Setups Property Page. From here you can set the oscilloscope or use it in the Setups collection. To load the file;

1 Click ‘Open’. A file dialog box will appear.
2. Select the file and click ‘Open’. The setup will be saved as a new setup in the list. If the name of the file is the same as an existing name on the list, you can choose to overwrite the existing setup or cancel the operation.
3. To send the setup to the oscilloscope, follow instructions under ‘Sending a saved setup to the oscilloscope’.

HP BenchLink XL and Microsoft Excel

Using the HP54600Scope Control on a Microsoft Excel User Form

The following example illustrates how to create a User Form that contains a **HP54600Scope Control** and a CommandButton control. The User Form displays a message box with the instrument model number. To illustrate how the **HP54600Scope Control** works in Microsoft Excel, follow these steps:

1 In a new workbook, click on any cell, and press ALT+F11 to activate the Visual Basic Editor.
2. On the **Insert menu**, click 'UserForm'. This step inserts UserForm1 into your project. If the Toolbox is not displayed, click 'Toolbox' on the **View menu**.
3. In the Toolbox, right-click on an empty space to get the 'Additional Controls' dialog box (or on the 'Tools' menu, click 'Additional Controls'). Select the **HP54600Scope Control** check box, and then click OK. The **HP54600Scope Control** will appear in the toolbox.
4. Connect a HP 54600-series oscilloscope to the PC.
5. Draw the **HP54600Scope Control** on UserForm1. With the **HP54600Scope Control** selected, press F4 to display the Properties window. Click on the (Custom) property, and then click on the ellipsis button.
6. The Property pages of the control will appear. You can use the 'Search Instruments' property page to search for the instrument, or the 'Set I/O' property page to manually set the I/O address. When using RS-232 or GPIB, be sure that the correct cable is used, and that the settings in the property page match the settings of the instrument. Use the 'Print/Utility' button on the front-panel of the HP 54600 oscilloscope to check the settings of the instrument. Use the 'Test' button to confirm the connection to the instrument.
7. In the Toolbox dialog box, click 'CommandButton' and draw the CommandButton on UserForm1.
8. With the CommandButton selected, press F4 to display the Properties window. Change the Caption property of the control to OK.
9. Right-click CommandButton and click View Code.
10. Type the following code for the Click event of the CommandButton:

```
Sub CommandButton1_Click()
   MsgBox "model number is; " & _
   HP54600Scope1.Utilities.InstrumentModel
End Sub
```

11. Press F5 to run the UserForm.

The UserForm is displayed. Click the OK button. A message box with the model number of the instrument will appear.

You can run the user form from a sheet. Go to the Excel main program and from the **View menu** select 'Toolbars' and click on 'Toolbox' . Fill in a macro name and select a letter (for example 'M') for the shortcut key. Click OK.

12. Click on the 'Command Button' in the 'Control Toolbox' and draw the button on the sheet.
13. To add macro code to the control, right-click the control, and then click 'View Code' on the shortcut menu. Add the following to the subroutine.

```
Private Sub CommandButton1_Click()
   UserForm1.Show
End Sub
```

14. To return to Microsoft Excel from the Visual Basic Editor, click 'Close and Return to Microsoft Excel' on the **File menu** or press ALT+Q.
15. To exit design mode and enable the ActiveX control, click 'Exit Design Mode' on the 'Control Toolbox'.
16. Now click on the command button. The UserForm will be displayed on the sheet.

Remarks

To access the complete help file, view the properties window and select (F1) a property page of the **HP546000Scope** control.

Using the HP54600Scope Control on a Microsoft Excel Worksheet

The simplest way of establishing communication with the instrument is to use the property pages of the control. This section shows how to get a measurement from the oscilloscope and put it on a worksheet.

1 Open the worksheet you want to use with the **HP 54600Scope Control.**
2. If the Control Toolbox is not displayed, point to 'Toolbars' on the **View menu**, and then click 'Control Toolbox'.
3. Click 'More Controls' -- usually on the bottom or far right of the 'Control Toolbox' toolbar.
4. Select the HP 54600Scope Control and then drag the mouse on the worksheet.

The HP 54600Scope Control will be placed on the worksheet. The control is normally not visible. To see the properties of the control be sure the 'Design Mode' button is enabled on the 'Control Toolbox' and then select properties on the 'Control Toolbox'. From the 'Properties' window select HP54600Scope1. Note the name in the properties window. This is the name used in code to call the control.

5. From the 'Properties' window, click on the (Custom) property, and then click on the ellipsis button.

The Property pages of the control will appear. You can use the 'Search Instruments' property page to search for the instrument, or the 'Set I/O' property page to manually set the I/O address. When using RS-232 or GPIB, be sure that the correct cable is used, and that the settings in the property page match the settings of the instrument. Use the 'Print/Utility' button on the front panel of oscilloscope to check the settings of the instrument.

6. Select the instrument connection if using the 'Search Instruments' page. Use the 'Test' button to confirm the connection to the instrument and click **OK** to complete the I/O setting.
7. Click on the 'Command Button' in the 'Control Toolbox' and draw the button on the sheet.
8. To add macro code to the button control, right-click the button control, and then click 'View Code' on the shortcut menu. Add the following to the subroutine.

```
Private Sub CommandButton1_Click()
   Cells(1, 6) = "Average Voltage"
   Cells(2, 6) = HP54600Scope1.Measure.AverageVoltage(1)
End Sub
```

9. To return to Microsoft Excel from the Visual Basic Editor, click 'Close and Return to Microsoft Excel' on the **File menu** or press ALT+Q.
10. To exit design mode and enable the ActiveX control, click 'Exit Design Mode' on the 'Control Toolbox'.
11. Now click on the command button. The measurement will be displayed in cell F2 on the worksheet.

Plot Waveform Data Using GetAllWaveformData Method (Excel)

The simplest way of plotting waveform data is to use an array of Variants returned by GetAllWaveformData, and then set the Range of the worksheet to the array. Once on the spreadsheet, use the Excel graph to plot the data. The following example gets 250 points from the oscilloscope and puts the data into the active worksheet:

```
Dim Data () As Variant
Dim cellrange as range

' Create the array of variants to send to spreadsheet
HP54600Scope1.GetAllWaveformData 250, Data

numbrows = UBound(data, 1)
numbcol = UBound(data, 2)

' set up the sheet range and put data on sheet
Set cellrange = Range(Cells(1, 1), Cells(1, 1).Offset(numbrows, numbcol))
Cellrange = data
```

Saving and Displaying a Screen Image (Excel)

The **HP54600Scope Control** returns a bitmap to a file. The file can be used to place a screen image on the Microsoft Excel worksheet. Use this code in the Excel worksheet where the **HP54600Scope Control** is sited.

```
HP54600Scope1.SaveScreenImage "C:\temp.bmp"
Shapes.AddPicture "c:\temp.bmp", False, True, 50, 50, 256, 140
Kill "c:\temp.bmp"
```

HP BenchLink XL and Microsoft Visual Basic

Using the HP54600Scope Control with Visual Basic

The simplest way to communicate with the instrument is to use the property pages of the control.

1 From the Visual Basic **File menu**, select 'New Project', 'Standard EXE'.
2. From the Components Control list dialog box (Ctrl T), select the HP 54600 Scope Control.
3. Click OK, the icon for the HP 54600 Scope control will appear in the toolbox.
4. Select the HP 54600 Scope Control and place on the form.
5. Connect the desired oscilloscope on either GPIB or RS-232.
6. Right click on the HP 54600 Scope control icon. The property pages will come up and attempt to find an instrument at the default address. If the default address is valid, the model number and address will appear in the 'Search' dialog. If the default address is not valid, select 'Search' or set the address manually from the 'Set I/O' property page. Alternately you can set the 'Address' in the property window.
7. Use the 'Test' button to confirm the connection to the instrument.
8. Now place a command button on the form and copy the following code to the form.

```
MsgBox "model number is; " & HP54600Scope1.Utilities.InstrumentModel
```

9. Run the project. A message box with the model number of the instrument will appear.

AnalogChannels Property Examples (Visual Basic)

This example checks to see if channel 1 is on. If the channel is on, the parameters for that channel are set.

```
With scope.AnalogChannels(1)
   If .Enabled Then
         .BandWidthLimit = HP54600_On
         .ProbeAttenuation = HP54600_ProbeAttenuation_X10
         .VerticalOffset = 1#
         .VerticalRange = 5#
         .ProbeCoupling = HP54600_ProbeCoupling_AC
   End If
End With
```

This example checks to see if channel 2 is on. If the channel is on, the rise time is returned to the variable reading.

```
Dim channelOn As Boolean
Dim reading As Double
channelOn = scope.AnalogChannels(2).Enabled
If channelOn Then
   reading = scope.Measure.RiseTime(2)
End If
```

Connect Method Examples (Visual Basic)

This example makes a connection to the oscilloscope using a GPIB address. Once connection is made, you can use the **scope object** to send commands, and receive data.

```
Dim reading As Double
Dim scope As New HP54600ServerLib.HP54600EZ
scope.Connect ("GPIB0::7")
reading = scope.Measure.RMSVoltage(1)
```

This example makes a connection to the oscilloscope using an RS-232 address. Once connection is made, you can use the **scope object** to send commands, and receive data.

```
Dim reading As Double
Dim scope As New HP54600ServerLib.HP54600EZ
scope.Connect ("COM1::Handshake=DTR_DSR")
reading = scope.Measure.RMSVoltage(1)
```

Enter and Output Methods Examples (Visual Basic)

To try these examples, place a command button on a form, double click on the button to show the code window and place the following code in the button subroutine. On the HP54600Scope control property page, set the address property to the instrument address.

The Enter Method provides several ways to read the data from an instrument. It has an internal number generator. The Enter Method can:

- Read data as a string
- Read data as a number (real)
- Parse a string into a numeric array
- Put IEEE 488.2 block data into an array

Reading a String

This code reads the instrument response as a string:

```
Dim reply As String
scope.Output "*idn?"
scope.Enter reply
```

Reading a String Array

If you want to parse a returned string, use a string array with the required or larger dimension.

```
Dim reply(4) As String
scope.Output "*idn?"
scope.Enter reply
```

Variant Array for Strings

If you do not know the array size needed, use a variant array. This code will size the array for the number of fields parsed.

```
Dim reply() As Variant
scope.Output "*idn?"
scope.Enter reply
```

Returning a Number

To return a number, such as a voltage measurement, declare the variable as a double. To return an array, declare the variable as an array. The operation is similar to the 'ENTER' functionality of HP Basic (Rocky Mountain Basic).

To make a voltage measurement using an HP 54602B:

```
Dim reading As Double
scope.Output "Measure:VRMS?"
scope.Enter reading
```

Returning a Numeric Array

To parse a list of numbers from a comma-separated string returned by the instrument, declare the variable as an array. This code reads the preamble of a HP 54602B oscilloscope.

```
Dim Preamble(10) As Double
scope.Output "Waveform:Preamble?"
scope.Enter Preamble()
```

Returning IEEE 488.2 Block Data

To read back an IEEE 488.2 block of data, declare the variable as a variant or, if the size of the data is known, you can declare it of the data type expected.

This code reads the waveform data from a HP 54602B oscilloscope.

```
Dim ydata(99) as Integer
scope.Output "Waveform:Format Word"
scope.Output "Waveform:data?"
scope.Enter ydata, "I2BE"
```

Alternately, if the data size is not known, you can declare the variable as a variant.

```
Dim ydata as variant
scope.Output "Waveform:Format byte"
scope.Output "Waveform:data?"
scope.Enter ydata, "I1"
```

Get and Plot Waveform Data Example (Visual Basic)

To plot waveform data, use the array of Variants returned by GetAllWaveformData, and then set the ChartData property to the array.

The following example gets 250 points from the oscilloscope and graphs the data using the Visual Basic Chart object:

```
HP54600Scope1.Initialize
Dim  Data  () as  Variant
'  Get  the  array  of  variants  to  send  to  the  graph  routine
HP54600Scope1.GetAllWaveformData  250,  Data

With  Chart1
   .Plot.SeriesCollection(1).Position.Excluded  =  True
   .Plot.Axis(VtChAxisIdX).CategoryScale.Auto  =  False
   .Plot.Axis(VtChAxisIdX).CategoryScale.DivisionsPerTick  =  25
   .Plot.Axis(VtChAxisIdX).CategoryScale.DivisionsPerLabel  =  26
   .legend.Location.Visible  =  True
   .chartType  =  VtChChartType2dLine
   .ChartData  =  Data
End  With
```

Note: the Visual Basic Chart object cannot handle the large number of data points returned by the HP 54620A/C and HP5 54645A/D oscilloscopes when using HP54600_DataPoints_ALL enumerator.

Get and Save a Screen Image Example (Visual Basic)

The **HP54600Scope Control** returns a Picture type object that can be used to set an Image or PictureBox control. To show a screen image on a form, place an Image object on a Visual Basic form and use GetScreenImage to obtain the oscilloscope image. For example:

```
Set Image1.Picture = HP54600Scope1.GetScreenImage
```

To save the screen image to a file, use SaveScreeImage. For example:

```
HP54600Scope1.SaveScreenImage "C:\Mydirectory\screen1.bmp"
```

GetWaveformData Example (Visual Basic)

This example gets 250 points of time and voltage data on channel 1 and returns the data as a Variant array.

```
Dim xtime As Variant
Dim yVolts As Variant
scope.GetWaveformData 1, 250, xtime, yVolts
```

InitializationProperties Example (Visual Basic)

This example displays all the I/O component's property settings, one per message box.

```
With scope.Utilities.Initialize.InitializeIO
   Dim properties As Variant
   properties = .InitializationProperties
   Dim idx As Integer
   Dim listofprops As String
   Dim prop As String
   listofprops = ""
   For idx = LBound(properties, 1) To UBound(properties, 1)
         prop = properties(idx, 0) & " = " & properties(idx, 1) & _
         vbCrLf
         listofprops = listofprops & prop
   Next
   MsgBox listofprops,,"Initialization Properties"
End With
```

Saving a Configuration to a Collection Example (Visual Basic)

This example first gets an oscilloscope setup, names it 'mySetup' and then saves it in the collection. The second code segment sends the same setup in the collection back to the oscilloscope. You can create a collection of oscilloscope setups using the Setups property page during design time and then use the SendToInstrument method to send a setup from the collection to the oscilloscope.

```
HP54600Scope1.Setups.Add("mySetup").GetFromInstrument

HP54600Scope1.Setups("mySetup").SendToInstrument
```

Saving and Using Setup Files Example (Visual Basic)

An oscilloscope configuration can be retrieved from the oscilloscope, saved to a file, and later sent back to the oscilloscope. This example first gets a configuration, names it 'temp' and then saves it to a file. The second code segment sends the setup from the file back to the oscilloscope. The name is then removed so it can be used again.

```
With HP54600Scope1.Setups.Add("temp")
   .GetFromInstrument
   .SaveToFile "C:\mydirectory\mySetup.scp"
End With
HP54600Scope1.Setups.Remove "temp"

With HP54600Scope1.Setups.Add ("temp")
   .ReadFromFile "C:\mydirectory\mySetup.scp"
   .SendToInstrument
End With
HP54600Scope1.Setups.Remove "temp"
```

ReadStateData and WriteStateData Examples (Visual Basic)

You can manually configure the oscilloscope and then use the ReadStateData Method to retrieve a setup string from the oscilloscope and place the setup in a variant where it can be stored to a file.

```
Dim g_scope As HP54600EZ
Dim g_statedata As Variant
g_statedata = g_scope.Utilities.ReadStateData
```

Once an oscilloscope setup has been saved, it can be sent to the oscilloscope with the WriteStateData method.

```
g_scope.Utilities.WriteStateData (g_statedata)
```

Returning a Single Measurement of the Waveform (Visual Basic)

The simplest way of making a measurement such as frequency is to let the oscilloscope make the measurement for you. Use this code to return the RMS voltage of channel 1. See the Measure Object for other measurements available.

```
Dim reading as double
reading = HP54600Scope1.Measure.RMSVoltage(1)
```

Saving and Sending an Oscilloscope Setup (Visual Basic)

The simplest way to setup the oscilloscope is to use the 'Setups' property page. The property page can be used to get an oscilloscope setup during design time. Once named, you can send the setup to the oscilloscope in code.

When you add a setup using the 'Setups' property page in the design mode, it becomes part of the worksheet, form, and project that contains the control. Saving or compiling the project will save all of the setups listed on the property page. You can also save individual setups to a separate file. This can be done at design time through the 'Open' and 'Save As' command buttons on the 'Setups' property page or when the program is run through your own code.

This example sends the setup named "mySetup" to the oscilloscope. The setup was named and retrieved from the oscilloscope with the Setups property page during design mode:

```
HP54600Scope1.Setups("mySetup").SendToInstrument
```

A setup can be retrieved from the oscilloscope with code. This example first gets an oscilloscope setup, names it 'mySetup' and then saves it to a file. The second code segment sends the setup in the file back to the oscilloscope.

```
With HP54600Scope1.Setups.Add("mySetup")
   .GetFromInstrument
   .SaveToFile "C:\mydirectory\mySetup.scp"
End With

With HP54600Scope1.Setups.Add ("temp")
   .ReadFromFile "C:\mydirectory\mySetup.scp"
   .SendToInstrument
End With
HP54600Scope1.Setups.Remove "temp"
```

Working with Oscilloscope Setups (Configuration) Example (Visual Basic)

To set up the oscilloscope during run time, use the 'Setups' property page. The Setups property page can also be used during design time to retrieve a setup. A retrieved setup can be named and saved and later sent to the oscilloscope.

When you add a setup using the 'Setups' property page in the design mode it becomes part of the document, form, and project that contains the control. Saving or compiling the project will save all of the setups listed on the property page. You can also save individual setups to a separate file. This can be done at design time through the 'Open' and 'Save As' command buttons on the 'Setups' property page or when the program is run through your own code.

This example sends the setup named 'mySetup' to the oscilloscope:

```
HP54600Scope1.Setups("mySetup").SendToInstrument
```

A setup can also be retrieved from the oscilloscope. This example first gets an oscilloscope setup, names it 'mySetup' and then saves it to a file. The second code segment sends the setup in the file back to the oscilloscope.

```
With HP54600Scope1.Setups.Add("mySetup")
   .GetFromInstrument
   .SaveToFile "C:\mydirectory\mySetup.scp"
End With

With HP54600Scope1.Setups.Add ("temp")
   .ReadFromFile "C:\mydirectory\mySetup.scp"
   .SendToInstrument
End With
HP54600Scope1.Setups.Remove "temp"
```

HP BenchLink XL and Visual C++

Using the Automation Server with Visual C++

To get started with Visual C++, first you will need to bring up a skeleton console project in Visual C++ and then modify it to get a reading from the oscilloscope. The project uses the **HP54600 Automation Server** (HP54600x.dll) located in the default directory:

```
C:\Program Files\Hewlett-Packard\HP BenchLink XL\HP54600
```

This sample assumes familiarity with the Visual C++ environment and some experience with the C++ language and the use of COM.

From the

```
BenchLink XL\HP54600\Samples\VC\GettingStartedInCPP
```

directory, copy the files to the directory of your choice. Now double click on the file named RoadRunner.dsw. Alternately, go to the menu 'File | Open Workspace' and from the Open Workspace dialog select RoadRunner.dsw in the 'GettingStartedInCPP' directory.

In the file tab of the RoadRunner Workspace, double click on the Header file 'StdAfx.h' and note the following is for the **HP54600 Automation Server**:

```
#define _WIN32_WINNT 0x0400
#define _ATL_APARTMENT_THREADED
#include <atlbase.h>
extern CComModule _Module;
#include <atlcom.h>
```

External dependencies are:

```
hp54600x.dll and HP54600x_i.c
```

Both are in the default directory

```
...\HP BenchLink XL\HP54600
```

The entry point for the console application is RoadRunner.cpp. Double click on the 'Source' file 'RoadRunner.cpp'.

You must include the stdafx.h file. The **HP54600 Automation Server** is added with an 'import' statement. The directory shown is the default install directory. Similarly for the hp54600x_i.c file. Both of these files are installed in the same directory by default.

```
#include "stdafx.h"
#import "C:\Program Files\Hewlett-Packard\HP BenchLink
XL\HP54600\hp54600x.dll"
#include "C:\Program Files\Hewlett-Packard\HP BenchLink
XL\HP54600\hp54600x_i.c"
```

The following line sets the GPIB address to 7 on card 0. Change this to match the settings of the oscilloscope. The second line is the message sent to the oscilloscope screen.

```
#define ADDRESS L"GPIB0::7::INSTR"
#define MSG L"Getting Started in C/C++ With HP54600 Automation
Server"
```

Now add this code under 'Do something useful here'

```
pScope->raw_Connect(ADDRESS,NULL);
pScope->raw_AutoScale();
pScope->Utilities->raw_DisplayMessage(MSG);
double result;
pScope->Measure-
>raw_DutyCycle((HP54600ServerLib::HP54600_AllChannels)1,&result);
printf("Result = %f\n",result);
```

The first line will establish the connection at the specified GPIB address.

The second line auto-scales the oscilloscope.

The third line sends a message to the oscilloscope display.

The fifth and sixth lines will take a duty cycle measurement and display it on the PC screen.

For additional examples, see the \samples\vcpp_50 directory.

To run the project, first from the 'Build' menu select 'Build RoadRunner.exe' (F7), and then from the 'Build' menu select 'Execute RoadRunner.exe' (Ctrl F5).

GetAllWaveformData Example (Visual C++)

This example gets the time and vertical axis data for all the channels of the oscilloscope that are on. It includes text headers for each column.

```
SAFEARRAY* psa;
HRESULT hr = g_pScope->raw_GetAllWaveformData _
(HP54600ServerLib::HP54600_DataPoints_250,&psa);
```

GetWaveformData Example (Visual C++)

This example will get the time and vertical axis data for the channel specified.

```
VARIANT varTimeAxis; VariantInit(varTimeAxis);
VARIANT varVertAxis; VariantInit(varVertAxis);
HRESULT hr = pScope->raw_GetWaveformData _
(HP54600ServerLib::HP54600_Channel_1, _
HP54600ServerLib::HP54600_DataPoints_250,&varTimeAxis,&varVertAxis);
```

ReadStateData and WriteStateData Examples (Visual C++)

You can manually configure the oscilloscope and then use the ReadStateData Method to retrieve a setup string from the oscilloscope and place the setup in a variant.

This example saves the oscilloscope settings to a file specified by SZ_FILE_SETUPDATA. The code assumes a device server has been instantiated and connected to the oscilloscope.

In the example, pScope is of the type HP54600ServerLib::IHP54600EZPtr.

```
#define SZ_FILE_SETUPDATA "c:\\temp\\instrumentsetup.data"

HP54600ServerLib::IUtilitiesHP54600Ptr pUtilities;
HRESULT hr = pScope->get_Utilities(&pUtilities);

VARIANT varSetupData;
VariantInit(&varSetupData);
 hr = pUtilities->raw_ReadStateData(&varSetupData);

// Figure out how much data to write
long lb, ub;
hr = SafeArrayGetLBound(varSetupData.parray,1,&lb);
hr = SafeArrayGetUBound(varSetupData.parray,1,&ub);
long FileSize = ub - lb + 1;

// Get a hold of the data in the safearray.
BYTE HUGEP* pdata;
hr = SafeArrayAccessData(varSetupData.parray, _
(void HUGEP* FAR*)&pdata);

// Open the file and put in the data...
FILE* fp = fopen(SZ_FILE_SETUPDATA,"wb");
long WriteCount = fwrite(pdata,1,FileSize,fp);

// Close the file and check for a successful write
fclose(fp);

// Let go of the data in the safearray
hr = SafeArrayUnaccessData(varSetupData.parray);if FAILED(hr) throw;
```

Once an oscilloscope setup has been saved, it can be sent to the oscilloscope with the WriteStateData method.

```
HRESULT hr;
HP54600ServerLib::IUtilitiesHP54600Ptr pUtilities;
hr = g_pScope->get_Utilities(&pUtilities);
hr = pUtilities->raw_WriteStateData(varSetupData);
```

SaveScreenImage Example (Visual C++)

This example saves the oscilloscope screen image to the file specified by LSZ_FILE_SCREENIMAGE

```
#define LSZ_FILE_SCREENIMAGE L"c:\\temp\\screenimage.bmp"
HRESULT hr = pScope->raw_SaveScreenImage _
(LSZ_FILE_SCREENIMAGE,HP54600ServerLib::HP54600_ImageFormat_BMP);
```

Index

H (continued)

I

L

M

N

O

P

Q

R

S

S (continued)

T

U

V

W