Agilent ParBERT 81250 Measurement Software

Eye Opening Measurement Programming Reference

Agilent Technologies

Important Notice

Authors: t3 medien GmbH

Contents

About this Reference

This document describes the functions, properties and methods for controlling the Bit Error Rate measurement from a remote application.

NOTE Basic knowledge on handling the *Agilent ParBERT 81250 Measurement Software* is assumed. For further information, refer to the *Framework User Guide* and the *Eye Opening Measurement User Guide*.

For general information on remote programming, refer to the *Measurement Software Programming Guide*.

Programming Reference

The following sections describe the methods and properties from a Visual Basic, VEE or LabView user perspective. Some differences in the syntax will exist for the Visual C (VC) user. The VC syntax is denoted in each of the properties and methods.

NOTE

- Some of the methods, events and properties are not available to the wrapper dll user.
- The methods `InitMeasurement` and `CloseMeasurement` are only available to the wrapper dll user.

The functions are sorted according to the following categories:

- *"Measurement Basics" on page 10* shows the functions used to handle measurements, to establish the connection to the firmware server and to work with the settings.
- *"Measurement Setup" on page 28* shows the functions used to work with ports and terminals, and to set the parameters for the measurement.
- *"Running the Measurement" on page 66* shows the functions used to run and stop the measurement.
- *"Handling Events and Callbacks" on page 69* shows the functions used to handle events and callbacks for the measurement.
- *"Error Handling" on page 73* shows the functions used to analyze errors.
- *"Handling Measurement Results" on page 75* shows the functions used to get, calculate, and modify measurement results.
- *"Pass/Fail Functionality" on page 87* shows the functions used to set and evaluate pass/fail decisions.
- *"Copy/Paste Functions" on page 94* shows the functions used to edit the data display.
- *"Persistence" on page 101* shows the functions used to load/save measurements and to export data from the measurements.

- *"Functions for General Purposes" on page 114* shows the functions used to access the online help, for example.

Measurement Basics

The following section shows the functions used to handle measurements, to establish the firmware connection and to work with systems and system settings.

Functions to Get/Delete Measurement Handles

The following table gives an overview on the methods, events and properties available to handle measurements:

Purpose	**Refer to...**
To close a specific measurement.	*"CloseMeasurement" on page 13*
To initialize a measurement.	*"InitMeasurement" on page 22*

Functions to Establish the Firmware Connection

The following table gives an overview on the methods, events and properties available to control measurements:

Purpose	**Refer to...**
To set the connection to the firmware server.	*"Server" on page 23*
To set the port the firmware server is connected to.	*"ServerPort" on page 24*
To specify the analyzer system.	*"AnalyzerSystem" on page 12*
To specify the generator system.	*"GeneratorSystem" on page 17*
To specify the name of a system to be delayed.	*"DelayStartSystem" on page 16*
To specify a start delay between two systems.	*"StartDelay" on page 25*
To create a new measurement.	*"CreateMeasurement" on page 15*
To prepare the measurement execution.	*"CreateMeasEx" on page 14*
To check whether the connection to the firmware server is valid.	*"IsFWSConnected" on page 22*

Working with Settings

The following table gives an overview on the methods, events and properties available to handle systems and system settings:

Purpose	**Refer to...**
To get the number of system settings defined for the analyzer.	*"GetAnalyzerSettingsCount" on page 19*
To get the names of the system settings defined for the analyzer.	*"GetAnalyzerSettingsName" on page 20*
To get the number of system settings defined for the generator.	*"GetGeneratorSettingsCount" on page 21*
To get the names of the system settings defined for the generator.	*"GetGeneratorSettingsName" on page 21*
To specify the system setting for the analyzer.	*"AnalyzerSystemSetting" on page 13*
To specify the system setting for the generator.	*"GeneratorSystemSetting" on page 18*
To send the analyzer's system setting to the firmware server.	*"UseAnalyzerSettings" on page 26*
To send the generator's system setting to the firmware server.	*"UseGeneratorSettings" on page 27*

AnalyzerSystem

ActiveX syntax

```
Object.AnalyzerSystem = [sSystem]
```

For Visual C:

```
Object.SetAnalyzerSystem(sSystem)
sSystem = Object.GetAnalyzerSystem()
```

Wrapper dll syntax

```
EYESetAnalyzerSystem(hMeasurement,
                     sSystem)
EYEGetAnalyzerSystem(hMeasurement,
                     bufferSize,
                     *sSystem)
```

Description

Sets/returns the name of the analyzer system. The analyzer system specifies the analyzer for the measurement together with the related system setting.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

sSystem Specifies the name of the analyzer system (data type: `string`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the analyzer system "DSRA":

```
m_EyeOpeningCTRL.AnalyzerSystem = "DSRA"
```

Related functions and methods

“AnalyzerSystemSetting” on page 13
“GeneratorSystem” on page 17
“CreateMeasurement” on page 15

AnalyzerSystemSetting

ActiveX syntax

```
Object.AnalyzerSystemSetting = [sSetting]
```

For Visual C:

```
Object.SetAnalyzerSystemSetting(sSetting)
sSetting = Object.GetAnalyzerSystemSetting()
```

Wrapper dll syntax

```
EYESetAnalyzerSystemSetting(hMeasurement,
                            sSetting)
EYEGetAnalyzerSystemSetting(hMeasurement,
                            bufferSize,
                            *sSetting)
```

Description

Sets/returns the name of the analyzer system setting specified in the *Agilent 81250 User Software*. The analyzer system specifies together with the related system setting the analyzer for the measurement.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

sSetting Specifies the name of the analyzer system setting (data type: `string`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the analyzer system setting "TEST_APP":

```
m_EyeOpeningCTRL.AnalyzerSystemSetting = "TEST_APP"
```

Related functions and methods

"AnalyzerSystem" on page 12

CloseMeasurement

Wrapper dll syntax

```
EYECloseMeasurement(hMeasurement)
```

NOTE This method is only available when using the wrapper dll.

Description

Closes the measurement. The handle to the measurement will be deleted. Any subsequent property or method using the handle will return an error.

Input parameter

hMeasurement Handle to the measurement (data type: `ViSession`) returned by `CreateMeasurement`

Related functions and methods

"CreateMeasurement" on page 15

CreateMeasEx

ActiveX syntax `Object.CreateMeasEx()`

Wrapper dll syntax `EYECreateMeasEx(hMeasurement)`

Description Creates the connection to the server, sets the generator and analyzer systems, delay system and the delay. It assumes that the server name, port number, analyzer and generator system, delay system and delay value have been set by using the appropriate objects properties.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Remarks You have to set the following properties before using this method: `Server`, `ServerPort`, `AnalyzerSystem`, `GeneratorSystem`, `DelayStartSystem`, `StartDelay`

Related functions and methods

“Server” on page 23
“ServerPort” on page 24
“AnalyzerSystem” on page 12
“GeneratorSystem” on page 17
“DelayStartSystem” on page 16
“StartDelay” on page 25

CreateMeasurement

ActiveX syntax

```
Object.CreateMeasurement(sServer,
                         sPort,
                         sAnalyzer,
                         sGenerator,
                         sDelaySystem,
                         dDelay)
```

Wrapper dll syntax

```
EYECreateMeasurement(hMeasurement,
                     sServer,
                     sPort,
                     sAnalyzer,
                     sGenerator,
                     sDelaySystem,
                     dDelay)
```

Description Creates the connection to the server, sets the generator and analyzer systems, the delay system and the delay.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement (data type: `ViSession`).

sServer IP address of the server or LOCALHOST, for example, "10.3.6.14" (data type: `String`).

sPort Port number, set 2203 for the default port (data type: `String`).

sAnalyzer Name of the analyzer system, for example, "DSRA" (data type: `String`).

sGenerator Name of the generator system, for example, "DSRB" (data type: `string`).

sDelaySystem Name of the delayed system. It should either be the same name as the analyzer system or the generator system. If the analyzer and generator systems are the same, set this to an empty string (data type: `string`).

dDelay Delay in seconds that the sDelaySystem is delayed from the other system (data type: `Double`).

Example To connect to the firmware server under IP address **`10.3.6.14`**, port number **`2203`**, using the system **`DSRA`** as analyzer and generator, with no start delay:

```
m_EyeOpeningCTRL.CreateMeasurement("10.3.6.14","2203","DSRA",
                                   "DSRA","",0)
```

Related functions and methods

"AnalyzerSystem" on page 12
"AnalyzerSystemSetting" on page 13
"GeneratorSystem" on page 17
"GeneratorSystemSetting" on page 18
"DelayStartSystem" on page 16
"StartDelay" on page 25

DelayStartSystem

ActiveX syntax

```
Object.DelayStartSystem = [sDelaySystem]
```

For Visual C:

```
Object.SetDelayStartSystem(sDelaySystem)
sDelaySystem = Object.GetDelayStartSystem()
```

Wrapper dll syntax

```
EYESetDelayStartSystem(hMeasurement,
                       sDelaySystem)
EYEGetDelayStartSystem(hMeasurement,
                       bufferSize,
                       *sDelaySystem)
```

Description Sets the name of the system that is delayed.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

sDelaySystem Name (data type: `String`) of the delayed system. It should be either the name of the analyzer or generator system set by `AnalyzerSystem` and `GeneratorSystem`. The delay system can be a null string, resulting in no delay. For the wrapper dll GET function, this parameter is a pointer.

Example To define a delay for the system **`DSRA`**:

```
m_EyeOpeningCTRL.DelayStartSystem = "DSRA"
```

Related functions and methods

"AnalyzerSystem" on page 12
"GeneratorSystem" on page 17

GeneratorSystem

ActiveX syntax

```
Object.GeneratorSystem = [sSystem]
```

For Visual C:

```
Object.SetGeneratorSystem(sSystem)
sSystem = Object.GetGeneratorSystem()
```

Wrapper dll syntax

```
EYESetGeneratorSystem(hMeasurement,
                      sSystem)
EYEGetGeneratorSystem(hMeasurement,
                      bufferSize,
                      *sSystem)
```

Description

Sets/gets the name of the generator system. The generator system specifies together with the related system setting the generator for the measurement.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

sSystem Specifies the name of the generator system (data type: `string`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the generator system "DSRA":

```
m_EyeOpeningCTRL.GeneratorSystem = "DSRA"
```

Related functions and methods

“GeneratorSystemSetting” on page 18
“AnalyzerSystem” on page 12

GeneratorSystemSetting

ActiveX syntax

```
Object.GeneratorSystemSetting = [sSetting]
```

For Visual C:

```
Object.GetGeneratorSystemSetting(sSetting)
sSetting = Object.SetGeneratorSystemSetting()
```

Wrapper dll syntax

```
EYESetGeneratorSystemSetting(hMeasurement,
                             sSetting)
EYEGetGeneratorSystemSetting(hMeasurement,
                             bufferSize,
                             *sSetting)
```

Description

Sets/returns the name of the generator system setting specified in the *Agilent 81250 User Software*. The generator system specifies the generator for the measurement together with the related system setting.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

sSetting Specifies the name of the generator system setting (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the generator system setting "TEST_APP":

```
m_EyeOpeningCTRL.GeneratorSystemSetting = "TEST_APP"
```

Related functions and methods

"GeneratorSystem" on page 17

GetAnalyzerSettingsCount

ActiveX syntax

```
nItems = Object.GetAnalyzerSettingsCount()
```

Wrapper dll syntax

```
EYEGetAnalyzerSettingsCount(hMeasurement,
                            *nItems)
```

Description

Returns the number of settings stored in firmware for the analyzer system.

Output parameter

nItems Number of settings for the analyzer system (data type: `Integer`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example

To get the number of settings for the analyzer system:

```
Dim nItems as Integer
nItems = m_EyeOpeningCTRL.GetAnalyzerSettingsCount
```

Related functions and methods

“GetAnalyzerSettingsName” on page 20

GetAnalyzerSettingsName

ActiveX syntax

```
sSettingName = Object.GetAnalyzerSettingsName(nIndex)
```

Wrapper dll syntax

```
EYEGetAnalyzerSettingsName(hMeasurement,
                           nIndex,
                           bufferSize,
                           *sSettingName)
```

Description Returns the setting name of the analyzer for a designated index.

Output parameter **sSettingName** Name of the analyzer setting (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nIndex Unique identifier for the setting. This is an index beginning at 0 (data type: `Integer`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

Example To get the first setting name for the analyzer:

```
Dim sSettingName as String
sSettingName = m_EyeOpeningCTRL.GetAnalyzerSettingsName(0)
```

Related functions and methods *"GetAnalyzerSettingsCount" on page 19*

GetGeneratorSettingsCount

ActiveX syntax

```
nItems = Object.GetGeneratorSettingsCount()
```

Wrapper dll syntax

```
EYEGetGeneratorSettingsCount(hMeasurement,
                             *nItems)
```

Description Returns the number of settings stored in firmware for the generator system.

Output parameter **nItems** Number of settings (data type: `Integer`) for the generator system. For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example To get the number of system settings defined for the generator system:

```
Dim nItems as Integer
nItems = m_EyeOpeningCTRL.GetGeneratorSettingsCount()
```

GetGeneratorSettingsName

ActiveX syntax

```
sSettingName = Object.GetGeneratorSettingsName(nIndex)
```

Wrapper dll syntax

```
EYEGetGeneratorSettingsName(hMeasurement,
                            nIndex,
                            bufferSize,
                            *sSettingName)
```

Description Returns the setting name of the generator for a designated index.

Output parameter **sSettingName** Name of the generator setting (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nIndex Unique identifier (data type: `Integer`) for the setting, an index starting at 0.

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

Example To get the first setting name for the generator:

```
Dim sSettingName as String
sSettingName = m_EyeOpeningCTRL.GetGeneratorSettingsName(0)
```

InitMeasurement

Wrapper dll syntax

```
ViSession hMeasurement = VI_NULL;
InitMeasurement(&hMeasurement)
```

NOTE This method is only available when using the wrapper dll.

Description Initializes the measurement and the handle to the measurement is returned. Any subsequent property or method calls should use the handle value that is returned.

Output parameter **hMeasurement** Handle to the measurement (data type: `ViSession`).

Related functions and methods

"SamplingDelay" on page 52
"TimingUnits" on page 56

IsFWSConnected

ActiveX syntax

```
bConnect = Object.IsFWSConnected()
```

Wrapper dll syntax

```
EYEIsFWSConnected(hMeasurement,
                  *bConnect)
```

Description Returns whether there is a connection to the firmware server. To run the measurement, a connection to the firmware server must be established.

Output parameter **bConnect** The following constants (data type: `Boolean`) are returned:

Constant	Description
`True`	There is a connection to the firmware server.
`False`	There is no connection to the firmware server.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example To check the connection to the firmware server:

```
Dim bConnect as Boolean
bConnect = m_EyeOpeningCTRL.IsFWSConnected()
```

Related functions and methods

"Server" on page 23
"ServerPort" on page 24

Server

ActiveX syntax

```
Object.Server = [sServer]
```

For Visual C:

```
Object.SetServer(sServer)
sServer = Object.GetServer()
```

Wrapper dll syntax

```
EYESetServer(hMeasurement,
             sServer)
EYEGetServer(hMeasurement,
             bufferSize,
             *sServer)
```

Description Sets/returns the server name for the *81200 Firmware Server.*

Parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

sServer Address of the server (data type: `String`). If the *81200 Firmware Server* is located on the local machine, any empty string "" is used. For the wrapper dll GET function, this parameter is a pointer.

Example To establish a connection to the server address "10.3.6.14":

```
m_EyeOpeningCTRL.Server = "10.3.6.14"
```

Related functions and methods *"ServerPort" on page 24*

ServerPort

ActiveX syntax

```
Object.ServerPort = [sPort]
```

For Visual C:

```
Object.SetServerPort(sPort)
SPort = Object.GetServerPort()
```

Wrapper dll syntax

```
EYESetServerPort(hMeasurement,
                 sPort)
EYEGetServerPort(hMeasurement,
                 bufferSize,
                 *sPort
```

Description Sets/returns the port ID for the firmware server connection.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

sPort Port number for the firmware server connection (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Example To establish a connection to the server located at port "2203":

```
m_EyeOpeningCTRL.ServerPort = "2203"
```

Related functions and methods *“Server” on page 23*

StartDelay

ActiveX syntax

```
Object.StartDelay = [dDelay]
```

For Visual C:

```
Object.SetStartDelay(dDelay)
dDelay = Object.GetStartDelay()
```

Wrapper dll syntax

```
EYEGetStartDelay(hMeasurement,
                 *dDelay)
EYESetStartDelay(hMeasurement,
                 dDelay)
```

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

dDelay Start delay in seconds between the two systems (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the start delay to 40 ms:

```
m_EyeOpeningCTRL.StartDelay = 0.04
```

Related functions and methods

“DelayStartSystem” on page 16

UseAnalyzerSettings

ActiveX syntax

```
Object.UseAnalyzerSettings = [boolean]
```

For Visual C:

```
Object.SetUseAnalyzerSettings(boolean)
boolean = Object.GetUseAnalyzerSettings()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns whether the analyzer settings will be sent to the *81200 Firmware Server*.

Parameter **boolean** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Send the analyzer settings to the firmware server.
`False`	Do not send the analyzer settings to the firmware server (default setting).

Remarks If this parameter is set to `TRUE`, a setting must be selected for the analyzer.

Example To send the selected system setting for the analyzer to the firmware server:

```
m_EyeOpeningCTRL.UseAnalyzerSettings = True
```

UseGeneratorSettings

ActiveX syntax

```
Object.UseGeneratorSettings = [boolean]
```

For Visual C:

```
Object.SetUseGeneratorSettings(boolean)
Boolean = Object.GetUseGeneratorSettings()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns whether the generator settings will be sent to the *81200 Firmware Server*.

Parameter **boolean** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Send the selected generator setting to the firmware server.
`False`	Do not send the generator setting to the firmware server (default setting).

Remarks If this parameter is set to true, a setting must be selected for the generator.

Example To send the system setting specified for the generator to the firmware server:

```
m_EyeOpeningCTRL.UseGeneratorSettings = True
```

Measurement Setup

The following section shows the functions used to handle the parameters for the measurement.

Working with Ports and Terminals

The following table gives an overview on the methods, events and properties available to handle ports and terminals involved into the measurement:

Purpose	**Refer to...**
To get the number of analyzer ports configured in the system.	*"GetAnalyzerPortCount" on page 34*
To get the names of analyzer ports configured in the system.	*"GetAnalyzerPortName" on page 35*
To get the number of analyzer terminals configured in the system.	*"GetAnalyzerTermCount" on page 36*
To get the name of one analyzer terminal.	*"GetAnalyzerTermName" on page 37*
To get a port ID for a given index.	*"GetPortId" on page 38*
To get a terminal ID for a given index and port.	*"GetTerminalId" on page 39*
To set/return if a given port is part of the measurement.	*"PortInvolved" on page 49*
To set/return if a given terminal is part of the measurement.	*"TermInvolved" on page 54*

Setting Eye Opening Parameters

The following table gives an overview on the methods, events and properties available to handle the values of the *Parameters* page:

Purpose	**Refer to...**
To set the number of bits to be compared before the measurement moves to the next sample point.	*"MaxComparedBits" on page 47*
To activate the property MaxError.	*"UseMaxError" on page 59*
To set the number of errors to be compared before the measurement moves to the next sample point.	*"MaxError" on page 48*
To get/set the timing resolution for the sample points to be taken.	*"TimingResolution" on page 55*
To activate edge resolution optimization.	*"UseEdgeResOptimization" on page 57*
To set the voltage resolution for the sample points to be taken.	*"VoltageResolution" on page 65*
To get/set the lower threshold of the sample voltage.	*"SampleLowLevelVoltage" on page 51*
To get/set the upper threshold of the sample voltage.	*"SampleHighLevelVoltage" on page 51*

Setting the Eye Opening Pass/Fail Parameters

The following table gives an overview on the methods, events and properties available to handle the values of the *Pass/Fail* page

The following table gives an overview on the methods, events and properties available to get/set pass/fail criteria:

Purpose	**Refer to...**
To activate the pass/fail checks for the measurement.	*"UseEyeOpeningPassFail" on page 58*
To activate the pass/fail check for the parameter *Time Eye Opening*.	*"UseMinTimeEyeOpening" on page 60*
To set the pass/fail value for the parameter *Time Eye Opening*.	*"MinTimeEyeOpening" on page 45*
To activate the pass/fail check for the parameter *Voltage Eye Opening*.	*"UseMinVoltsEyeOpening" on page 61*
To set the pass/fail value for the parameter *Voltage Eye Opening*.	*"MinVoltsEyeOpening" on page 46*
To activate the pass/fail check for the parameter *Optimal Sample Delay*.	*"UseSampleDelay" on page 62*
To get/set the lower pass/fail threshold for the parameter *Optimal Sample Delay*.	*"MinSampleDelay" on page 43*
To get/set the upper pass/fail threshold for the parameter *Optimal Sample Delay*.	*"MaxSampleDelay" on page 41*
To activate the pass/fail check for the parameter *Optimal Sample Voltage*.	*"UseSampleVoltage" on page 63*
To set the lower pass/fail threshold for the parameter *Optimal Sample Voltage*.	*"MinSampleVoltage" on page 44*
To get/set the upper pass/fail threshold for the parameter *Optimal Sample Voltage*.	*"MaxSampleVoltage" on page 42*

Setting Eye Opening View Parameters

The following table gives an overview on the methods, events and properties available to handle the values of the *View* page:

Purpose	**Refer to...**
To specify the BER threshold.	*"BERThreshold" on page 31*
To check if results can be displayed in absolute mode.	*"DisplayedAbs" on page 32*
To display the measured points in the graphical view.	*"DisplayPoints" on page 33*
To set the number of decimal places to be displayed in the numerical view.	*"GridPrecision" on page 40*
To update the graphical view while running the measurement.	*"RedrawingEnabled" on page 50*
To set the result display to absolute or relative mode.	*"SamplingDelay" on page 52*
To show markers in the graphical display.	*"ShowMarkers" on page 53*
To set the timing units to be used.	*"TimingUnits" on page 56*
To specify if the bathtub curve or the jitter graph will be displayed.	*"ViewType" on page 64*

Specify the Properties Dialog

The following table gives an overview on the methods, events and properties available to display the *Properties* dialog:

Purpose	**Refer to...**
To set the title of the *Properties* dialog.	*"PropertiesTitle" on page 50*
To display the *Properties* dialog.	*"ShowProperties" on page 53*

BERThreshold

ActiveX syntax

```
Object.BERThreshold = [dBERThreshold]
```

For Visual C:

```
Object.SetBERThreshold(dBERThreshold)
dBERThreshold = Object.GetBERThreshold()
```

Wrapper dll syntax

```
EYEGetBERThreshold(hMeasurement,
                   *dBERThreshold)
EYESetBERThreshold(hMeasurement,
                   dBERThreshold)
```

Description Sets/returns the BER threshold value that is used in the analysis of the calculated parameters:

Constant	Equivalent to parameter...
`EyeOpeningTimeSpan`	Time Eye Opening
`EyeOpeningVoltageSpan`	Voltage Eye Opening
`SampleDelay`	Optimal Sample Delay
`SampleVoltage`	Optimal Sample Voltage

For parameter definitions, refer to *Definitions* in the *Eye Opening Measurement User Guide*.

Parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

dBERThreshold Specifies the BER threshold used to calculate the parameters *Time Eye Opening*, *Voltage Eye Opening*, *Optimal Sample Point Delay*, and *Optimal Sample Point Voltage* for the numerical view (data type: `double`). For the wrapper dll GET function, this parameter is a pointer.

Example To set the BER threshold to $4 \cdot 10^{-3}$:

```
m_EyeOpeningCTRL.BERThreshold = 4E-3
```

Related functions and methods *"CalcMeasParams" on page 77*

DisplayedAbs

ActiveX syntax

```
boolean = Object.DisplayedAbs
```

For Visual C:

```
boolean = Object.GetDisplayedAbs()
```

Wrapper dll syntax

```
EYEGetDisplayedAbs(hMeasurement,
                   *boolean)
```

Description

Checks if data can be displayed in *absolute* mode. For the definitions of the display modes, refer to *Timing Unit Definitions* in the *Eye Opening Measurement User Guide*.

Output parameter

boolean The following values (data type: `Boolean`) are available:

Constant	Description
`True`	The data can be displayed in *absolute* mode.
`False`	The data can not be displayed in *absolute* mode.

This parameter is read-only. For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example

To check if results can be displayed in absolute mode:

```
Dim bDisplay as Boolean
bDisplay = m_EyeOpeningCTRL.DisplayedAbs
```

DisplayPoints

ActiveX syntax

```
Object.DisplayPoints = [bDisplayPoints]
```

For Visual C:

```
Object.SetDisplayPoints(bDisplayPoints)
bDisplayPoints = Object.GetDisplayPoints()
```

NOTE This property is not available for the wrapper dll.

Description Specifies if the data points will be displayed on the graph.

Input parameter **bDisplayPoints** The following values (data type: `Boolean`) are defined:

Constant	Description
`True`	Turns on the display of data points on the graph.
`False`	Turns off the display of data points on the graph.

Example To display the measured points in the graphical view:

```
m_EyeOpeningCTRL.DisplayPoints = True
```

GetAnalyzerPortCount

ActiveX syntax

```
nPorts = Object.GetAnalyzerPortCount()
```

Wrapper dll syntax

```
EYEGetAnalyzerPortCount(hMeasurement,
                        *nPorts)
```

Description Returns the number of analyzer ports configured on the 81200 hardware.

Output parameter **nPorts** Number of data ports (data type: `Integer`) configured on the 81200 hardware. For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example To get the number of analyzer ports:

```
Dim nPorts as Integer
nPorts = m_EyeOpeningCTRL.GetAnalyzerPortCount
```

Related functions and methods *“GetAnalyzerPortName” on page 35*

GetAnalyzerPortName

ActiveX syntax

```
sName = Object.GetAnalyzerPortName(nPortID)
```

Wrapper dll syntax

```
EYEGetAnalyzerPortName(hMeasurement,
                       nPortID,
                       bufferSize,
                       *sName)
```

Description Returns the analyzer port name for a designated port ID.

Output parameter **sName** Name of the port as configured on the GUI (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

bufferSize Only for the wrapper dll: Specifies the size of the data buffer for the returned data (data type: `ViInt32`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

Example To get the name of port 1:

```
Dim sName as String
sName = m_EyeOpeningCTRL.GetAnalyzerPortName(1)
```

Related functions and methods *"GetAnalyzerPortCount" on page 34*

GetAnalyzerTermCount

ActiveX syntax

```
nTermCount = Object.GetAnalyzerTermCount(nPortID)
```

Wrapper dll syntax

```
EYEGetAnalyzerTermCount(hMeasurement,
                        nPortID,
                        *nTermCount)
```

Description

Returns the number of terminals for the specified port as configured on the 81200 hardware.

Output parameter

nTermCount Number of terminals for the specified port (data type: `Integer`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

Example

To get the number of terminals configured for port 1:

```
Dim nTermCount as Integer
nTermCount = m_EyeOpeningCTRL.GetAnalyzerTermCount(1)
```

Related functions and methods

"GetAnalyzerTermName" on page 37

GetAnalyzerTermName

ActiveX syntax

```
sName = Object.GetAnalyzerTermName(nPortID, nTerminalID)
```

Wrapper dll syntax

```
EYEGetAnalyzerTermName(hMeasurement,
                       nPortID,
                       nTerminalID,
                       *sName)
```

Description

Returns the analyzer terminal name for a designated terminal ID.

Output parameter

sName Name of the port as configured on the GUI (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

nTerminalID A terminal is addressed by the terminal number (data type: `Integer`). This is an index starting at 1 for each port.

Example

To get the name of terminal 1:

```
Dim sName as String
sName = m_EyeOpeningCTRL.GetAnalyzerTermName(1)
```

Related functions and methods

"GetAnalyzerTermCount" on page 36

GetPortId

ActiveX syntax

```
nPortID = Object.GetPortId(nIndex)
```

Wrapper dll syntax

```
EYEGetPortId(hMeasurement,
             nIndex,
             *nPortID)
```

Description

Returns the port ID associated with an index. In some configurations, the ports are not sequential and may not begin with port ID = 1. This method allows you to uniquely identify ports.

Output parameter

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1. For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nIndex Unique identifier (data type: `Integer`) for the port, an index starting at 0.

Example

To get the port ID for the first index:

```
Dim nPortID as Integer
nPortID = m_EyeOpeningCTRL.GetPortId(0)
```

GetTerminalId

ActiveX syntax

```
nTermID = Object.GetTerminalId(nIndex,
                               nPortID)
```

Wrapper dll syntax

```
EYEGetTerminalId(hMeasurement,
                 nIndex,
                 nPortID,
                 *nTermID)
```

Description

Returns the terminal ID associated with an index and a port. In some configurations, the terminals are not sequential and may not begin with terminal ID = 1. This method allows you to uniquely identify terminals.

Output parameter

nTermID Returned value: A terminal is addressed by the terminal number (data type: `Integer`). For the wrapper dll GET function, this parameter is a pointer.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nIndex Index starting at 0 (data type: `Integer`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

Example

To get the terminal Id for the first index of port 1:

```
Dim nTermID as Integer
nTermID = m_EyeOpeningCTRL.GetTerminalId(0, 1)
```

GridPrecision

ActiveX syntax

```
Object.GridPrecision = [ePrecision]
```

For Visual C:

```
Object.SetGridPrecision(ePrecision)
ePrecision = Object.GetGridPrecision()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns the number of decimal places shown in the numerical view.

Parameter **ePrecision** The following values (data type: `PRECISION`) are defined:

Constant	Description
`Zero`	No significant digits after the decimal place are shown.
`One`	One significant digit after the decimal place is shown.
`Two`	Two significant digits after the decimal place are shown.
`Three`	Three significant digits after the decimal place are shown.

Example To set the number of decimal places to be displayed to two:

```
m_EyeOpeningCTRL.GridPrecision = Two
```

MaxSampleDelay

ActiveX syntax

```
Object.GetMaxSampleDelay(dValue,
                         eUnits)
```

```
Object.SetMaxSampleDelay(dValue,
                         eUnits)
```

Wrapper dll syntax

```
EYEGetMaxSampleDelay(hMeasurement,
                     *dValue,
                     *eUnits)
```

```
EYESetMaxSampleDelay(hMeasurement,
                     dValue,
                     eUnits)
```

Description

Sets/returns the pass/fail criterion for the measurement parameter *Optimal Sample Delay.* If the calculated measurement parameter returned by `GetPortCalculatedValue` or `GetTermCalculatedValue` is less than the pass/fail criterion, the measurement parameter passes.

For parameter definitions, refer to *Definitions* in the *Eye Opening Measurement User Guide.*

Output parameters

dValue The pass/fail criterion (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

eUnits The units for `dValue`. The following constants are defined (data type: `TIMINGUNITS`):

Constant	Description
`UnitInterval`	To use unit intervals as time base.
`Seconds`	To use seconds as time base.

For the wrapper dll GET function, this parameter is a pointer.

Remarks

The pass/fail value is entered in the units according to the `SamplingDelay` property. In addition, the pass/fail criterion is applied according to the `SamplingDelay` property.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example

To get the pass/fail criterion:

```
Dim dValue as Double
Dim eUnits as TimingUnitsEnums
m_EyeOpeningCTRL.GetMaxSampleDelay(dValue, eUnits)
```

Related functions and methods

"GetPortCalculatedValue" on page 82
"GetTermCalculatedValue" on page 84
"MinSampleDelay" on page 43

MaxSampleVoltage

ActiveX syntax

```
Object.MaxSampleVoltage = [dValue]
```

Wrapper dll syntax

```
EYEGetMaxSampleVoltage(hMeasurement,
                       *dValue)
```

```
EYESetMaxSampleVoltage(hMeasurement,
                       dValue)
```

Description

Sets/returns the pass/fail criterion for the measurement parameter *Optimal Sample Voltage*. If the calculated measurement parameter returned by `GetPortCalculatedValue` or `GetTermCalculatedValue` is less than the pass/fail criterion, the measurement parameter passes.

For parameter definitions, refer to *Definitions* in the *Eye Opening Measurement User Guide*.

Output parameter

dValue The pass/fail criterion (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example

To get the pass/fail criterion:

```
Dim dValue as Double
m_EyeOpeningCTRL.GetMaxSampleVoltage(dValue)
```

Related functions and methods

"GetPortCalculatedValue" on page 82
"GetTermCalculatedValue" on page 84
"MinSampleVoltage" on page 44
"MaxSampleVoltage" on page 42

MinSampleDelay

ActiveX syntax

```
Object.GetMinSampleDelay(dValue,
                         tUnits)
```

```
Object.SetMinSampleDelay(dValue,
                         tUnits)
```

Wrapper dll syntax

```
EYEGetMinSampleDelay(hMeasurement,
                     *dValue,
                     *tUnits)
```

```
EYESetMinSampleDelay(hMeasurement,
                     dValue,
                     tUnits)
```

Description

Gets/sets the pass/fail criterion for the measurement parameter *Optimal Sample Delay*. If the calculated measurement parameter returned by `GetPortCalculatedValue` or `GetTermCalculatedValue` is higher than the pass/fail criterion, the measurement parameter passes.

For parameter definitions, refer to *Definitions* in the *Eye Opening Measurement User Guide*.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

dValue The pass/fail criterion (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

tUnits The units for `dValue` (data type: `TIMINGUNITS`). The following constants are defined:

Constant	Description
`UnitInterval`	To use unit intervals as time base.
`Seconds`	To use seconds as time base.

For the wrapper dll GET function, this parameter is a pointer.

Remarks

The pass/fail value is entered in the units according to the `SamplingDelay` property. In addition, the pass/fail criterion is applied according to the `SamplingDelay` property.

Example To get the pass/fail criterion:

```
Dim dValue as Double
Dim eUnits as TimingUnitsEnums
m_EyeOpeningCTRL.GetMinSampleDelay(dValue, eUnits)
```

Related functions and methods

"SamplingDelay" on page 52
"GetPortCalculatedValue" on page 82
"GetTermCalculatedValue" on page 84

MinSampleVoltage

ActiveX syntax

```
Object.MinSampleVoltage = [dValue]
```

Wrapper dll syntax

```
EYEGetMinSampleVoltage(hMeasurement,
                       *dValue)
```

```
EYESetMinSampleVoltage(hMeasurement,
                       dValue)
```

Description Sets/returns the pass/fail criterion for the measurement parameter *Optimal Sample Voltage*. If the calculated measurement parameter returned by `GetPortCalculatedValue` or `GetTermCalculatedValue` is higher than the pass/fail criterion, the measurement parameter passes.

For parameter definitions, refer to *Definitions* in the *Eye Opening Measurement User Guide*.

Output parameter **dValue** The pass/fail criterion (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example To get the pass/fail criterion:

```
Dim dValue as Double
m_EyeOpeningCTRL.MinSampleVoltage = [dValue]
```

Related functions and methods

"GetPortCalculatedValue" on page 82
"GetTermCalculatedValue" on page 84

MinTimeEyeOpening

ActiveX syntax

```
Object.GetMinTimeEyeOpening(dValue,
                            tUnits)
```

```
Object.SetMinTimeEyeOpening(dValue,
                            tUnits)
```

Wrapper dll syntax

```
EYEGetMinTimeEyeOpening(hMeasurement,
                        *dValue,
                        *tUnits)
```

```
EYESetMinTimeEyeOpening(hMeasurement,
                        dValue,
                        tUnits)
```

Description

Sets/returns the pass/fail criterion for the measurement parameter *Time Eye Opening*. If the calculated measurement parameter returned by `GetPortCalculatedValue` or `GetTermCalculatedValue` is higher than the pass/fail criterion, the measurement parameter passes.

For parameter definitions, refer to *Definitions* in the *Eye Opening Measurement User Guide*.

Output parameters

dValue The pass/fail criterion (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

tUnits The units for `dValue` (data type: `TIMINGUNITS`). The following constants are defined:

Constant	Description
`UnitInterval`	To use unit intervals as time base.
`Seconds`	To use seconds as time base.

For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example

To get the pass/fail criterion:

```
Dim dValue as Double
m_EyeOpeningCTRL.GetMinTimeEyeOpening(dValue)
```

To set the pass/fail criterion:

```
Dim dValue as Double
m_EyeOpeningCTRL.SetMinTimeEyeOpening(dValue, tUnits)
```

Related functions and methods

“GetPortCalculatedValue” on page 82
“GetTermCalculatedValue” on page 84

MinVoltsEyeOpening

ActiveX syntax

```
Object.MinVoltsEyeOpening = [dValue]
```

Wrapper dll syntax

```
EYEGetMinVoltsEyeOpening(hMeasurement,
                         *dValue)

EYESetMinVoltsEyeOpening(hMeasurement,
                         dValue)
```

Description

Sets/returns the pass/fail criterion for the measurement parameter *Voltage Eye Opening*. If the calculated measurement parameter returned by `GetPortCalculatedValue` or `GetTermCalculatedValue` is higher than the pass/fail criterion, the measurement parameter passes.

For parameter definitions, refer to *Definitions* in the *Eye Opening Measurement User Guide*.

Output parameter

dValue The pass/fail criterion (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example

To get the pass/fail criterion:

```
Dim dValue as Double
m_EyeOpeningCTRL.MinVoltsEyeOpening = [dValue]
```

Related functions and methods

"GetPortCalculatedValue" on page 82
"GetTermCalculatedValue" on page 84

MaxComparedBits

ActiveX syntax

```
Object.MaxComparedBits = [dMComparedBits]
```

For Visual C:

```
Object.SetMaxCompareBits(dMCompareBits)
dMCompareBits = Object.GetMaxCompareBits()
```

Wrapper dll syntax

```
EYEGetMaxComparedBits(hMeasurement,
                      *dMComparedBits)
EYESetMaxComparedBits(hMeasurement,
                      dMComparedBits)
```

Description Sets/returns the maximum number of compared bits. This value is used as a stopping criterion for each data point of the measurement.

Parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

dMComparedBits Number of compared bits that must be reached before the firmware server moves on to the next sample point delay (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Example To allow a maximum of 100000 bits for each sample point:

```
m_EyeOpeningCTRL.MaxComparedBits = 100000
```

Related functions and methods *"MaxError" on page 48*

MaxError

ActiveX syntax

```
Object.MaxError = [dErrors]
```

For Visual C:

```
Object.SetMaxError(dErrors)
dErrors = Object.GetMaxError()
```

Wrapper dll syntax

```
EYEGetMaxError(hMeasurement,
               *dErrors)
EYESetMaxError(hMeasurement,
               dErrors)
```

Description Sets/returns the maximum number of errors. This value is used as a stopping criteria for each data point.

Parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

dErrors Maximum number of errors that must be reached before the *81200 Firmware Server* moves on to the next sample point delay (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Example To allow a maximum of 1000 errors for each sample point:

```
m_EyeOpeningCTRL.MaxError = 1000
```

Related functions and methods *“MaxComparedBits” on page 47*

PortInvolved

ActiveX syntax

```
Object.SetPortInvolved(nPortID,
                       boolean)
```

```
boolean = Object.GetPortInvolved(nPortID)
```

Wrapper dll syntax

```
EYESetPortInvolved(hMeasurement,
                   nPortID,
                   boolean)
```

```
EYEGetPortInvolved(hMeasurement,
                   nPortID,
                   *boolean)
```

Description Sets/returns whether a port is involved in the measurement.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

boolean Indicates if the port is involved in the measurement. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Port is involved in the measurement.
`False`	Port is not involved in the measurement.

For the wrapper dll GET function, this parameter is a pointer.

Example To set port 1 involved in the measurement:

```
m_EyeOpeningCTRL.SetPortInvolved(1, True)
```

To get whether port 1 is involved in the measurement:

```
Dim bIsInvolved as Boolean
bIsInvolved = m_EyeOpeningCTRL.GetPortInvolved(1)
```

PropertiesTitle

ActiveX syntax

```
Object.PropertiesTitle = [sTitle]
```

For Visual C:

```
Object.SetPropertiesTitle(sTitle)
sTitle = Object.GetPropertiesTitle()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns the title of the *Properties* dialog.

Parameter **sTitle** Specifies the title of the *Properties* dialog (data type: `String`).

Example To set the title of the *Properties* dialog to "`Eye Opening`":

```
m_EyeOpeningCTRL.PropertiesTitle = "Eye Opening"
```

RedrawingEnabled

ActiveX syntax

```
Object.RedrawingEnabled = [boolean]
```

NOTE This property is not available for the wrapper dll.

Description Allows to update the graphical view while running the measurement.

Parameter **boolean** Indicates if the port is involved in the measurement. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	The graphical view will be updated (default setting).
`False`	The graphical view will not be updated.

SampleHighLevelVoltage

ActiveX syntax

```
Object.SampleHighLevelVoltage = [dValue]
```

Wrapper dll syntax

```
EYEGetSampleHighLevelVoltage(hMeasurement,
                             *dValue)
```

```
EYESetSampleHighLevelVoltage(hMeasurement,
                             dValue)
```

Description Gets/sets the upper threshold of the sample voltage.

Output parameter **dValue** The voltage in V (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example To set the upper threshold of the sample voltage to 0.5 V.

```
m_EyeOpeningCTRL.SampleHighLevelVoltage = 0.5
```

Related functions and methods *“SampleLowLevelVoltage” on page 51*

SampleLowLevelVoltage

ActiveX syntax

```
Object.SampleLowLevelVoltage = [dValue]
```

Wrapper dll syntax

```
EYEGetSampleLowLevelVoltage(hMeasurement,
                            *dValue)
```

```
EYESetSampleLowLevelVoltage(hMeasurement,
                            dValue)
```

Description Gets/sets the lower threshold of the sample voltage.

Output parameter **dValue** The voltage in V (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example To set the lower threshold of the sample voltage to 0.5 V.

```
m_EyeOpeningCTRL.SampleLowLevelVoltage = 0.5
```

Related functions and methods *“SampleHighLevelVoltage” on page 51*

SamplingDelay

ActiveX syntax

```
Object.SamplingDelay = [eSamplingDelay]
```

Wrapper dll syntax

```
EYESetSamplingDelay(hMeasurement,
                    eSamplingDelay)
EYEGetSamplingDelay(hMeasurement,
                    *eSamplingDelay)
```

Description

Sets/returns the sampling delay which effects the display of information on the grid and graph. This property controls how the delay value is reported in the `GetMeasData` and `GetEYEDataPoint` functions. This property also controls the units of the analysis parameters reported in `GetPortCalculatedValue` and `GetTermCalculatedValue`. Also, this property is used in the application of the pass/fail criteria for the calculated parameters. If `SamplingDelay` is set to `Relative`, the values entered for the pass/fail decision are relative values and the pass/fail criterion is applied to the relative calculated values.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

eSamplingDelay The following constants (data type: `SAMPLINGDELAY`) are defined:

Constant	Description
`Relative`	The terminal data will be displayed in the grid and graph relative to its port's optimal sample point delay.
`Absolute`	The terminal data will be displayed in the grid and graph in absolute time.

For the wrapper dll GET function, this parameter is a pointer.

Example

To set the display type to "absolute":

```
m_EyeOpeningCTRL.SamplingDelay = Absolute
```

Related functions and methods

"GetPortCalculatedValue" on page 82
"GetTermCalculatedValue" on page 84
"GetMeasData" on page 81
"GetEYEDataPoint" on page 79

ShowMarkers

ActiveX syntax

```
Object.ShowMarkers = [boolean]
```

For Visual C:

```
Object.SetShowMarkers(boolean)
boolean = Object.GetShowMarkers()
```

NOTE This property is not available for the wrapper dll.

Description Sets the display of markers on or off.

Input parameter **boolean** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Turn on the display of markers.
`False`	Turn off the display of markers (default setting).

Example To display markers in the graphical view:

```
m_EyeOpeningCTRL.ShowMarkers = True
```

ShowProperties

ActiveX syntax

```
Object.ShowProperties()
```

NOTE This method is not available for the wrapper dll.

Description Displays the *Properties* dialog.

Parameter None

TermInvolved

ActiveX syntax

```
boolean = Object.GetTermInvolved(nPortID,
                                 nTermID)
```

```
Object.SetTermInvolved(nPortID,
                       nTermID,
                       boolean)
```

Wrapper dll syntax

```
EYEGetTermInvolved(hMeasurement,
                   nPortID,
                   nTermID,
                   *boolean)
```

```
EYESetTermInvolved(hMeasurement,
                   nPortID,
                   nTermID,
                   boolean)
```

Description Sets whether a port is involved in the measurement.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

nTermID A terminal is addressed by the terminal number (data type: `Integer`). This is an index starting at 1 for each port.

boolean Indicates if the port is involved in the measurement. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Terminal is involved in the measurement.
`False`	Terminal is not involved in the measurement.

For the wrapper dll GET function, this parameter is a pointer.

Example To set terminal 1 of port 1 not involved in the measurement:

```
m_EyeOpeningCTRL.SetTermInvolved(1, 1, False)
```

Related functions and methods *"PortInvolved" on page 49*

TimingResolution

ActiveX syntax

```
Object.GetTimingResolution(dValue,
                           eUnits)
```

```
Object.SetTimingResolution(dValue,
                           eUnits)
```

Wrapper dll syntax

```
EYEGetTimingResolution(hMeasurement,
                       *dValue,
                       *eUnits)
```

```
EYESetTimingResolution(hMeasurement,
                       dValue,
                       eUnits)
```

Description

Sets/gets the measurement's timing resolution.

For parameter definitions, refer to *Timing Unit Definitions* in the *Eye Opening Measurement User Guide.*

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

dValue Value of the timing resolution (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

eUnits The units for `dValue` (data type: `TIMINGUNITS`). The following constants are defined:

Constant	Description
`UnitInterval`	To use unit intervals as time base.
`Seconds`	To use seconds as time base.

For the wrapper dll GET function, this parameter is a pointer.

Example

To get the timing resolution:

```
Dim dValue as Double
Dim eUnits as TimingUnitEnums
m_EyeOpeningCTRL.GetTimingResolution(dValue, eUnits)
```

To set the distance between two sample points to 0.01 unit intervals:

```
m_EyeOpeningCTRL.SetTimingResolution(0.01, UnitInterval)
```

TimingUnits

ActiveX syntax

```
Object.TimingUnits = eTimingUnits
```

For Visual C:

```
Object.SetTimingUnits(eTimingUnits)
eTimingUnits = Object.GetTimingUnits()
```

Wrapper dll syntax

```
EYESetTimingUnits(hMeasurement,
                  eTimingUnits)
EYEGetTimingUnits(hMeasurement,
                  *eTimingUnits)
```

Description

Determines how the data on the graph and the grid will be displayed, either in Unit Interval or in seconds. This property controls how the delay value is reported in the `GetMeasData` and `GetEYEDataPoint` functions. This property also controls what units the analysis parameters are reported in (refer to `GetPortCalculatedValue` and `GetTermCalculatedValue`).

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

eTimingUnits Specifies the basic unit for the display. The following constants are defined (data type: `TimingUnitsEnum`):

Constant	Description
`UnitInterval`	Data on the graph and the calculated measurement values in the grid will be shown in the unit interval (default setting).
`Seconds`	Data on the graph and the calculated measurement values in the grid will be shown in seconds using the appropriate engineering unit (for example, ms).

For the wrapper dll GET function, this parameter is a pointer.

Example

To display the results in seconds:

```
m_EyeOpeningCTRL.TimingUnits = Seconds
```

Related functions and methods

"Run" on page 68

UseEdgeResOptimization

ActiveX syntax

```
Object.UseEdgeResOptimization = [boolean]
```

For Visual C:

```
Object.GetUseEdgeResOptimization(boolean)
boolean = SetUseEdgeResOptimization()
```

Wrapper dll syntax

```
EYEGetUseEdgeResOptimization(hMeasurement,
                             *boolean)
EYESetUseEdgeResOptimization(hMeasurement,
                             boolean)
```

Description

Turns on and off the edge resolution algorithm in the *81200 Firmware Server*. If `UseEdgeResOptimization` is `False`, equidistant delay steps are used to generate the eye diagram.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The following constants (data type: `Boolean`) are defined:

Constant	Description
`False`	Equidistant delay steps are used to generate the bathtub curve (default setting).
`True`	An optimization algorithm is used and more data points are taken at the transition from all errors to no errors.

For the wrapper dll GET function, this parameter is a pointer.

Example

To switch edge resolution optimization off:

```
m_EyeOpeningCTRL.UseEdgeResOptimization = False
```

UseEyeOpeningPassFail

ActiveX syntax

```
Object.UseEyeOpeningPassFail = [boolean]
```

For Visual C:

```
Object.SetUseEyeOpeningPassFail(boolean)
boolean = Object.GetUseEyeOpeningPassFail()
```

Wrapper dll syntax

```
EYEGetUseEyeOpeningPassFail(hMeasurement,
                            *boolean)
EYESetUseEyeOpeningPassFail(hMeasurement,
                            boolean)
```

Description Sets/returns the turning on and off of all of the pass/fail checks for the measurement parameters.

Parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The following constants (data type: `Boolean`) are defined:

Constant	Description
True	Turns on the pass/fail checking for all eye opening parameters.
False	Turns off the pass/fail checking for all eye opening parameters (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Example To enable all pass/fail criteria:

```
m_EyeOpeningCTRL.UseEyeOpeningPassFail = True
```

UseMaxError

ActiveX syntax

```
Object.UseMaxError = [boolean]
```

For Visual C:

```
Object.SetUseMaxError(boolean)
boolean = Object.GetUseMaxError()
```

Wrapper dll syntax

```
EYEGetUseMaxError(hMeasurement,
                  *boolean)
EYESetUseMaxError(hMeasurement,
                  boolean)
```

Description

Sets/returns whether the firmware server will use the property `MaxError` as a criteria for moving to the next measurement point.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Turn on the checking of max errors for each data point.
`False`	Turn off the checking of max errors for each data point (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Example

To disable the `MaxError` option:

```
m_EyeOpeningCTRL.UseMaxError = False
```

Related functions and methods

"MaxError" on page 48

UseMinTimeEyeOpening

ActiveX syntax

```
Object.UseMinTimeEyeOpening = [boolean]
```

For Visual C:

```
Object.SetUseMinTimeEyeOpening(boolean)
boolean = Object.GetUseMinTimeEyeOpening()
```

Wrapper dll syntax

```
EYESetUseMinTimeEyeOpening(hMeasurement,
                           boolean)
EYEGetUseMinTimeEyeOpening(hMeasurement,
                           *boolean)
```

Description Sets/returns the turning on and off of all of the pass/fail checks for the parameter *Time Eye Opening*.

Remarks You must set `UseEyeOpeningPassFail` to `True` to enable the individual eye opening pass/fail parameters.

Parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The following constants (data type: `Boolean`) are defined:

Constant	Description
True	Turns on the pass/fail checking for *Time Eye Opening*.
False	Turns off the pass/fail checking for *Time Eye Opening* (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Example To enable the pass/fail criterion:

```
m_EyeOpeningCTRL.UseMinTimeEyeOpening = True
```

Related functions and methods *"MinTimeEyeOpening" on page 45*

UseMinVoltsEyeOpening

ActiveX syntax

```
Object.UseMinVoltsEyeOpening = [boolean]
```

For Visual C:

```
Object.SetUseMinVoltsEyeOpening(boolean)
boolean = Object.GetUseMinVoltsEyeOpening()
```

Wrapper dll syntax

```
EYESetUseMinVoltsEyeOpening(hMeasurement,
                            boolean)
EYEGetUseMinVoltsEyeOpening(hMeasurement,
                            *boolean)
```

Description Sets/returns the turning on and off of all of the pass/fail check for the parameter *Voltage Eye Opening*.

Remarks You must set `UseEyeOpeningPassFail` to `True` to enable the individual eye opening pass/fail parameters.

Parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The following constants (data type: `Boolean`) are defined:

Constant	Description
True	Turns on the pass/fail checking for *Voltage Eye Opening*.
False	Turns on the pass/fail checking for *Voltage Eye Opening* (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Example To enable the pass/fail criterion:

```
m_EyeOpeningCTRL.UseMinVoltsEyeOpening = True
```

Related functions and methods *“MinVoltsEyeOpening” on page 46*

UseSampleDelay

ActiveX syntax

```
Object.UseSampleDelay = [boolean]
```

For Visual C:

```
Object.SetUseSampleDelay(boolean)
boolean = Object.GetUseSampleDelay()
```

Wrapper dll syntax

```
EYEGetUseSampleDelay(hMeasurement,
                     *boolean)
EYESetUseSampleDelay(hMeasurement,
                     boolean)
```

Description

Turns the pass/fail checks for the parameter *Optimal Sample Delay* on or off.

Remarks

You must set `UseEyeOpeningPassFail` to `True` to enable the individual eye opening pass/fail parameters.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Turns on the pass/fail checking for this parameter.
`False`	Turns off the pass/fail checking for this parameter (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Example

To enable the pass/fail criterion:

```
m_EyeOpeningCTRL.UseEyeOpeningPassFail = True
m_EyeOpeningCTRL.UseSampleDelay = True
```

Related functions and methods

“MaxSampleDelay” on page 41
“MinSampleDelay” on page 43
“UseEyeOpeningPassFail” on page 58
“UseSampleVoltage” on page 63

UseSampleVoltage

ActiveX syntax

```
Object.UseSampleVoltage = [boolean]
```

For Visual C:

```
Object.SetUseSampleVoltage(boolean)
boolean = Object.GetUseSampleVoltage()
```

Wrapper dll syntax

```
EYESetUseSampleVoltage(hMeasurement,
                       boolean)
EYEGetUseSampleVoltage(hMeasurement,
                       *boolean)
```

Description Turns the pass/fail checks for the parameter *Optimal Sample Voltage* on or off.

Remarks You must set `UseEyeOpeningPassFail` to `True` to enable the individual eye opening pass/fail parameters.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Turns on the pass/fail checking for this parameter.
`False`	Turns off the pass/fail checking for this parameter (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Example To enable the pass/fail criterion:

```
m_EyeOpeningCTRL.UseEyeOpeningPassFail = True
m_EyeOpeningCTRL.UseSampleVoltage = True
```

Related functions and methods

“MaxSampleVoltage” on page 42
“MinSampleVoltage” on page 44
“UseEyeOpeningPassFail” on page 58
“UseSampleDelay” on page 62

Related functions and methods

“GetMeasPassValue” on page 87
“GetPortPassValue” on page 89

ViewType

ActiveX syntax

```
Object.ViewType = [eViewType]
```

For Visual C:

```
Object.SetViewType(eViewType)
eViewType = Object.GetViewType()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns the view type for the eye opening graph.

For the definitions of the view types, refer to *Display Modes* in the *Eye Opening Measurement User Guide*.

Parameter **eViewType** The following constants (data type: `VIEW`) are defined:

Constant	Description
`ContourPlot`	To display several curves for equal bit error rates.
`PseudoColorPlot`	To use different colors for the regions between the lines of equal bit error rate (BER).
`EqualBERatThreshold`	To display only one curve for the bit error threshold.

Example To display the eye diagram in the contour plot:

```
m_EyeOpeningCTRL.ViewType = ContourPlot
```

VoltageResolution

ActiveX syntax

```
Object.VoltageResolution = [dValue]
```

Wrapper dll syntax

```
EYEGetVoltageResolution(hMeasurement,
                        *dValue)
```

```
EYESetVoltageResolution(hMeasurement,
                        dValue)
```

Description

Sets/returns the measurement’s voltage resolution.

For parameter definitions, refer to *Timing Unit Definitions* in the *Eye Opening Measurement User Guide*.

Output parameter

dValue Value of the voltage resolution (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example

To set the distance between two sample points to 0.1 V:

```
m_EyeOpeningCTRL.VoltageResolution = 0.1
```

Running the Measurement

The following table gives an overview on the methods, events and properties available to run the measurement:

Purpose	Refer to...
To load the measurement settings to the firmware server.	*"Download" on page 66*
To start a measurement run.	*"Run" on page 68*
To get the status of a measurement.	*"MeasState" on page 67*
To ensure a synchronous run on several systems.	*"SynchronousRun" on page 68*
To stop a measurement run.	*"Stop" on page 68*

Download

ActiveX syntax	`Object.Download()`
Wrapper dll syntax	`EYEMeasureDownload(hMeasurement)`
Description	Ensures that all sequences are downloaded to the firmware server, so that a subsequent run has repeatable runtime behavior.
Input parameter	**hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).
Related functions and methods	*"Run" on page 68*

MeasState

ActiveX syntax

```
sState = Object.MeasState()
```

For Visual C:

```
sState = Object.GetMeasState()
```

Wrapper dll syntax

```
EYEMeasureState(hMeasurement,
                *sState)
```

NOTE The wrapper dll function is called `MeasureState`.

Description Returns the status of the measurement object. This property is read-only.

Output parameter **sState** The following measurement states (data type: `String`) are defined:

Constant	Description
`RUNN`	Measurement is running.
`SYNC`	Measurement is synchronizing.
`PROG`	Measurement is ready.
`HALT`	Measurement is prepared to run but the clocks are not ready and therefore, the measurement is waiting for an external trigger to start.

For the wrapper dll GET function, this parameter is a pointer.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example To check the state of the measurement:

```
Dim sState as String
sState = m_EyeOpeningCTRL.MeasState
```

Related functions and methods *“MeasHelpPath” on page 118*

Run

ActiveX syntax	`Object.Run()`
Wrapper dll syntax	`EYEMeasureRun(hMeasurement)`
Description	Starts the measurement and ensures that the systems are started in the proper order as specified. Necessary sequence downloads are performed before, so that the time between starting system 1 and system 2 is always the specified delay.
Input parameter	**hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Stop

ActiveX syntax	`Object.Stop()`
Wrapper dll syntax	`EYEMeasureStop(hMeasurement)`
Description	Stops the measurement.
Input parameter	**hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).
Related functions and methods	*"Run" on page 68*

SynchronousRun

ActiveX syntax	`Object.SynchronousRun()`
Wrapper dll syntax	`MeasureRun(hMeasurement)`
Description	Starts the measurement but it does not return until the measurement has been completed. This method also ensures that the systems are started in the proper order as specified. Necessary sequence downloads are performed before, so that the time between starting system 1 and system 2 is always the specified delay.
Input parameter	**hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Handling Events and Callbacks

The following table gives an overview on the methods, events and properties available to handle events and callbacks

Purpose	**Refer to...**
To indicate if result data is available.	*“OnMeasDataAvailable” on page 70*
To indicate if the measurement run is complete.	*“OnMeasurementComplete” on page 70*
To indicate that a transition in the measurement status occured.	*“OnMeasurementState” on page 71*
To register a callback function for certain events.	*“SetMeasEventsCallback” on page 72*

OnMeasDataAvailable

ActiveX syntax `Private Sub Object_OnMeasDataAvailable(lItemCount as Long)`

NOTE This event is not available for the wrapper dll.

Description Returns the number of data items that are currently available in the measurement object. The returned number can be used to allocate the required buffer size for requesting measured data or to check whether data is available or not.

Input parameter **lItemCount** A long value that indicates the number of data points available.

Example To get the number of data points available and display the value in a message box:

```
Private Sub MeasureEyeOpening1_OnMeasDataAvailable(lItemCount as Long)
   Dim msg as string
   msg = "Number of items:"+ lItemCount
   Msgbox msg
End Sub
```

OnMeasurementComplete

ActiveX syntax `Private Sub Object_OnMeasurementComplete(ByVal lStatus As Long)`

NOTE This event is not available for the wrapper dll.

Description Returns the measurement status upon completion of the measurement.

Input parameter **lStatus** A long value that indicates that the measurement run has been completed.

Example To check whether the measurement run is finished and display the fact in a message box:

```
Private Sub MeasureEyeOpening1_OnMeasurementComplete(
                                          ByVal lStatus As Long)
   MsgBox "Measurement is complete"
End Sub
```

OnMeasurementState

ActiveX syntax

```
Private Sub Object_OnMeasurementState(eMeasState As
                                        MeasureStateEnums)
```

NOTE This event is not available for the wrapper dll.

Description Returns the measurement status when a transition in the measurement status occurs.

Input parameter **eMeasState** Indicates the measurement status. Possible values (data type: `MEASURESTATE`) are:

Constant	Description
`StateAbort`	The measurement run has been aborted.
`StateComplete`	The measurement run has been completed.
`StateError`	An error occurred.
`StateHalt`	The system is in halted state, waiting for external start.
`StateProg`	Measurement data is transferred.
`StateRunning`	The measurement is running.
`StateRunPulse`	Heartbeat state.
`StateSync`	The system is synchronizing.

For further information on the states, refer to *Handle Events and Callbacks* in the *Measurement Software Programming Guide*.

Example To check the measurement status and display the fact in a message box:

```
Private Sub MeasureEyeOpening1_OnMeasurementState(ByVal lStatus As Long)
   MsgBox "The measurement switched to state: " + eMeasState
End Sub
```

SetMeasEventsCallback

NOTE This function is only available when using the wrapper dll.

Wrapper dll syntax

```
EYESetMeasEventsCallback(hMeasurement,
                         lpMeasEventsFunc,
                         lParamUserCBData)
```

Description To register a callback function which is called for certain events.
For further information, refer to *Handle Events and Callbacks* in the *Measurement Software Programming Guide*.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

lpMeasEventsFunc Callback function, for example:

```
ViStatus CALLBACK MeasEventsProc(ViSession hMeasurement, LPARAM
                                 lUserData, LONG lEventId, LPARAM
                                 lParam)
```

lParamUserCBData Measurement information structure.

Related functions and methods

"CopyToClipboard" on page 95
"CutToClipboard" on page 96

Error Handling

The following table gives an overview on the methods, events and properties available to handle errors:

Purpose	**Refer to...**
To get the most recent error number from the firmware server.	*"GetLastMeasError" on page 73*
To specify if errors are to be reported when running a measurement.	*"SilentMode" on page 74*

GetLastMeasError

ActiveX syntax

```
sError = Object.GetLastMeasError(lError)
```

Wrapper dll syntax

```
EYEGetLastMeasError(hMeasurement,
                    *lError,
                    sError)
```

Description Returns the last measurement error description from the firmware server.

Output parameters **sError** Error description (data type: `String`).

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

lError Error number (data type: `Long`). For the wrapper dll GET function, this parameter is a pointer.

SilentMode

ActiveX syntax

```
Object.SilentMode = [boolean]
```

For Visual C:

```
Object.SetSilentMode(boolean)
boolean = Object.GetSilentMode()
```

NOTE This property is not available for the wrapper dll.

Description Turns the display of error messages on and off.

Input parameter **boolean** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Turn on the display of error messages.
`False`	Turn off the display of error messages (default setting).

Example To turn error messages on:

```
m_EyeOpeningCTRL.SilentMode = True
```

Handling Measurement Results

- The following table gives an overview on the methods, events and properties available to get and calculate measurement results:

Purpose	Refer to...
To specify the error type to be displayed.	*"AnalyzeErrors" on page 76*
To specify whether the measurement parameters will be calculated for the BER threshold or for "0".	*"CalcMeasParams" on page 77*
To get the number of data points available for a terminal.	*"DataAvailable" on page 78*
To get one single data point (result).	*"GetEYEDataPoint" on page 79*
To get the raw measurement results.	*"GetMeasData" on page 81*
To get the calculated results for one port.	*"GetPortCalculatedValue" on page 82*
To get the calculated results for one terminal.	*"GetTermCalculatedValue" on page 84*
To get the measurement period.	*"MeasPeriod" on page 86*

AnalyzeErrors

ActiveX syntax

```
Object.AnalyzeErrors = [eAnalyzeErrors]
```

For Visual C:

```
Object.SetAnalyzeErrors(eAnalyzeErrors)
eAnalyzeErrors = Object.GetAnalyzeErrors()
```

Wrapper dll syntax

```
EYEGetAnalyzeErrors(hMeasurement,
                    *eAnalyzeErrors)
EYESetAnalyzeErrors(hMeasurement,
                    eAnalyzeErrors)
```

Description

Indicates which data will be shown in the graphical display and which data will be used to calculate the measurement parameters.

For parameter definitions, refer to *How to Work with a Measurement* in the *Framework User Guide*.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

eAnalyzeErrors Defines the error types to be displayed. The following constants (data type: `ANALYZE_ERRORS`) are defined:

Constant	Description
`AllErrors`	Data will be shown for all errors (default setting).
`ErrorsIf1s`	To display the errors if “1” is expected, but “0” received.
`ErrorsIf0s`	To display the errors if “0” is expected, but “1” received.

For the wrapper dll GET function, this parameter is a pointer.

Example

To display the results for all errors:

```
m_EyeOpeningCTRL.AnalyzeErrors = AllErrors
```

CalcMeasParams

ActiveX syntax

```
Object.CalcMeasParams = [eMeasure]
```

For Visual C:

```
Object.SetCalcMeasParams(eMeasure)
eMeasure = Object.GetCalcMeasParams()
```

Wrapper dll syntax

```
EYEGetCalcMeasParams(hMeasurement,
                     *eMeasure)
EYESetCalcMeasParams(hMeasurement,
                     eMeasure)
```

Description Sets whether the measurement parameters will be calculated at the BER threshold or at "0".

Parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

eMeasure Determines if the measurement parameters will be calculated at the BER threshold or at "0". The following constants (data type: `CALCMEAS`) are defined:

Constant	Description
`Errors`	Measurement parameters will be calculated at a BER threshold = 0.
`BERThreshold`	Measurement parameters will be calculated at the BER threshold set by `BERThreshold`.

For the wrapper dll GET function, this parameter is a pointer.

Example To calculate the measurement parameters for the BER threshold:

```
m_EyeOpeningCTRL.CalcMeasParams = BERThreshold
```

Related functions and methods *"BERThreshold" on page 31*

DataAvailable

ActiveX syntax

```
Object.DataAvailable(nPortId,
                     nTerminalID,
                     lNumPoints)
```

Wrapper dll syntax

```
EYEDataAvailable( hMeasurement,
                  nPortId,
                  nTerminalID,
                  *lNumPoints)
```

Description Returns the number of data points that are available for a specified port and terminal.

Output parameter **lNumPoints** The returned value (data type: `long`) specifies the number of points that are available in the measurement. For the wrapper dll GET function, this parameter is a pointer.

Input parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortId A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

nTerminalID A terminal is addressed by the terminal number (data type: `Integer`). This is an index starting at 1 for each port.

Example To get the number of data points available for terminal 1 of port 1:

```
Dim nItems as Long
m_EyeOpeningCTRL.DataAvailable(1, 1, nItems)
```

Related functions and methods *"GetEYEDataPoint" on page 79*

GetEYEDataPoint

ActiveX syntax

```
Object.GetEYEDataPoint(nPortID,
                       nTermID,
                       lDataIndex,
                       dDelay,
                       dThreshold,
                       dComparedBits,
                       dErroneousBits,
                       dErroneousZeros,
                       dErroneousOnes,
                       bExtrapolatedFlag)
```

Wrapper dll syntax

```
EYEGetEYEDataPoint(hMeasurement,
                   nPortID,
                   nTermID,
                   lDataIndex,
                   *dDelay,
                   *dComparedBits,
                   *dErroneousBits,
                   *dErroneousZeros,
                   *dErroneousOnes,
                   *bExtrapolatedFlag)
```

Description

Gets a single data point for a particular port and terminal ID. Use the `DataAvailable` method to get the number of data points.

Output parameters

dComparedBits Number of compared bits (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

dErroneousBits Number of errors (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

dErroneousZeros Number of errors from zeros (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

dErroneousOnes Number of errors from ones (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

bExtrapolatedFlag Flag (data type: `Boolean`) indicating whether the data was calculated or extrapolated by the firmware server. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Data point was extrapolated by firmware server.
`False`	Data point was not extrapolated by firmware server.

For the wrapper dll GET function, this parameter is a pointer.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

nTermID A terminal is addressed by the terminal number (data type: `Integer`). This is an index starting at 1 for each port.

lDataIndex Data index (data type: `Long`) starting at 0. Use the `DataAvailable` method to return the number of data points.

dDelay Delay (data type: `Double`) reported according to the `TimingUnits` and `SamplingDelay` properties. For the wrapper dll GET function, this parameter is a pointer.

dThreshold BER threshold used to calculate the parameters (data type: `Double`).

Remarks

Values returned are according to the `TimingUnits` and `SamplingDelay` properties. If you wish to have the delay value in relative time, set the `SamplingDelay` to relative. If you want the delay value reported in seconds then set the `TimingUnits` to seconds.

For the definitions of the display modes, refer to *Timing Unit Definitions* in the *Eye Opening Measurement User Guide*.

Example

To get the first data point for terminal 1 of port 1:

```
Dim dDelay, dThreshold, dComparedBits, dErroneousBits,
dErroneousZeros, dErroneousOnes as Double;
Dim bExtrapolatedFlag as Boolean;

m_EyeOpeningCTRL.GetEYEDataPoint(1, 1, 0, dDelay, dThreshold,
               dComparedBits, dErroneousBits, dErroneousZeros,
               dErroneousOnes, bExtrapolatedFlag);
```

Related functions and methods

“DataAvailable” on page 78
“TimingUnits” on page 56
“SamplingDelay” on page 52

GetMeasData

ActiveX syntax

```
outEyeData = Object.GetMeasData(sPortID,
                                sTermID,
                                lItemCount,
                                lItemsRet)
```

Wrapper dll syntax

```
EYEGetMeasData(hMeasurement,
               sPortID,
               sTermID,
               lItemCount,
               *lItemsRet,
               *outEyeData)
```

Description

Returns a variant which contains the *raw* data from the measurement for a designated port and terminal. The method allows you to enter a start index and the number of data points to be returned. It also returns the number of data points found.

Output parameters

lItemsRet Number of data points (data type: `Long`) returned to the variant. You can request more data points then may be available. For the wrapper dll GET function, this parameter is a pointer.

outEyeData Array (data type: `Variant`) of data that contains the delay (data type: `Double`), the voltage (data type: `Double`), number of compared bits (data type: `Double`), errors (data type: `Double`), errors from zeros (data type: `Double`), errors from ones (data type: `Double`) and the extrapolated flag (data type: `Boolean`) of the data. The index starts at 0. For the wrapper dll GET function, this parameter is a pointer.

- `dDelay = vData(index, 0)`
- `voltage = vData(index, 1)`
- `comparedBits = vData(index, 2)`
- `nErrors = vData(index, 3)`
- `nErrors0s = vData(index, 4)`
- `nErrors1s = vData(index, 5)`
- `bExtrapolatedFlag = vData(index, 6)`

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

sPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

sTermID A terminal is addressed by the terminal number (data type: `Integer`). This is an index starting at 1 for each port.

lStartItem Start index of data points to be returned (data type: `Long`). This is an index starting at 0.

lItemCount Number of data points to be returned (data type: `Long`).

Remarks Values returned are according to the `TimingUnits` and `SamplingDelay` properties. If the user wishes to have the delay value in relative time, set the `SamplingDelay` to relative. If the user wants the delay value reported in seconds then set the TimingUnits to seconds.

Example To return a variant that contains 5 sets of data if `lItemRet` returns 5:

```
Dim vData as Variant
Dim lItemRet as Long
VData = m_EyeOpeningCTRL.GetMeasData(1, 1, 0, 5, lItemRet)
```

GetPortCalculatedValue

ActiveX syntax

```
dValue = Object.GetPortCalculatedValue(nPortID,
                                       nAnalysisTerm,
                                       plsValid)
```

Wrapper dll syntax

```
EYEGetPortCalculatedValue(hMeasurement,
                          nPortID,
                          nAnalysisTerm,
                          *plsValid,
                          *dValue)
```

Description Returns the measurement parameter for the designated port and analysis term. The parameter is calculated at the BER threshold value.

Output parameter **dValue** Returned value (data type: `Double`): Calculated value at the BER threshold for the designated analysis term. For the wrapper dll GET function, this parameter is a pointer.

Input parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

nAnalysisTerm The following constants (data type: `ANALYSIS_TERM`) are defined:

Parameter	Description
`EyeOpeningTimeSpan`	The *Time Eye Opening* for a terminal is the maximum extension of the BER threshold contour line in sample delay direction.
`EyeOpeningVoltageSpan`	The *Voltage Eye Opening* for a terminal is the maximum extension of the BER threshold contour line in sample voltage direction.
`SampleDelay`	The *Optimal Sample Delay* for a terminal is the sample delay coordinate of the center of a bounding box around the BER threshold contour line.
`SampleVoltage`	The *Optimal Sample Voltage* for a terminal is the sample voltage coordinate of the center of a bounding box around the BER threshold contour line.

pIsValid Returns whether the value returned is valid for the measurement (data type: `Boolean`). The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Value is valid for measurement.
`False`	Value is not valid for measurement.

For the wrapper dll GET function, this parameter is a pointer.

Remarks You have to check if `bValid = True` before reporting or using the returned value. Values returned are according to the `TimingUnits` and `SamplingDelay` properties. If the user wishes to have the analysis parameter in relative time, set the SamplingDelay to relative. If the user wants the analysis value reported in seconds then set the TimingUnits to seconds.

Example To get the calculated parameter `EyeOpeningTimeSpan` for port 1:

```
Dim dValue as Double
Dim bValid as boolean
dValue = m_EyeOpeningCTRL.GetPortCalculatedValue(1,
                                        EyeOpeningTimeSpan, bValid)
```

Related functions and methods *"SamplingDelay" on page 52*
"TimingUnits" on page 56

GetTermCalculatedValue

ActiveX syntax

```
dValue = Object.GetTermCalculatedValue(nPortID,
                                        nTermID,
                                        nAnalysisTerm,
                                        plsValid)
```

Wrapper dll syntax

```
EYEGetTermCalculatedValue(hMeasurement,
                          nPortID,
                          nTermID,
                          nAnalysisTerm,
                          *plsValid,
                          *dValue)
```

Description

Returns the measurement parameter for the designated port, terminal and analysis term. The parameter is calculated at the BER threshold value.

Output parameter

dValue Returned value (data type: `Double`): Calculated value at the BER threshold for the designated analysis term. For the wrapper dll GET function, this parameter is a pointer.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

nTermID A terminal is addressed by the terminal number (data type: `Integer`). This is an index starting at 1 for each port.

nAnalysisTerm The following constants (data type: `ANALYSIS_TERM`) are defined:

Constant	Equivalent to parameter...
`EyeOpeningTimeSpan`	Time Eye Opening
`EyeOpeningVoltageSpan`	Voltage Eye Opening
`SampleDelay`	Optimal Sample Delay
`SampleVoltage`	Optimal Sample Voltage

plsValid Returns whether the value returned is valid for the measurement. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Value is valid for measurement.
`False`	Value is not valid for measurement.

For the wrapper dll GET function, this parameter is a pointer.

Remarks You have to check if `bValid` = `True` before reporting or using the returned value. Values returned are according to the `TimingUnits` and `SamplingDelay` properties. If you wish to have the analysis parameter in relative time, set the `SamplingDelay` to `Relative`. If you want the analysis value reported in seconds then set the `TimingUnits` to `Seconds`.

Example To get the calculated parameter `EyeOpeningTimeSpan` for terminal 3 of port 1:

```
Dim dValue as Double
Dim bValid as boolean
dValue = m_EyeOpeningCTRL.GetTermCalculatedValue(1, 3,
                                 EyeOpeningTimeSpan, bValid)
```

Related functions and methods *"SamplingDelay" on page 52*
"TimingUnits" on page 56

MeasPeriod

ActiveX syntax

```
dPeriod = Object.MeasPeriod
```

For Visual C:

```
dPeriod = Object.GetMeasPeriod()
```

Wrapper dll syntax

```
EYEGetMeasPeriod(hMeasurement,
                 *dPeriod)
```

Description

Returns the measurement period for the unit interval. This is needed to convert from unit interval to seconds and vice versa. This property is read-only.

Output parameter

dPeriod Measurement period in seconds (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

Input parameter

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

Example

To get the measurement period:

```
Dim dPeriod as Double
dPeriod = m_EyeOpeningCTRL.MeasPeriod
```

Pass/Fail Functionality

The following sections show the functions used to set and evaluate pass/fail decisions.

The following table gives an overview on the methods, events and properties available to check whether a port or terminal has passed or failed:

Purpose	Refer to...
To check if a parameter has passed for the complete measurement.	*"GetMeasPassValue" on page 87*
To check if a parameter has passed for a given port.	*"GetPortPassValue" on page 89*
To check if a parameter has passed for a given terminal.	*"GetTermPassValue" on page 91*

GetMeasPassValue

NOTE This function is only available when using the wrapper dll.

Wrapper dll syntax

```
EYEGetMeasPassValue(hMeasurement,
                    nAnalysisTerm,
                    *pIsPass)
```

Description Returns whether a measurement parameter has passed or failed the pass/fail criterion. The criteria are set by separate pass/fail methods.

Output parameter **pIsPass** Returns whether the designated measurement parameter passed or failed. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Measurement passed.
`False`	Measurement failed.

For the wrapper dll GET function, this parameter is a pointer.

Input parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nAnalysisTerm The following constants (data type: `ANALYSIS_TERM`) are defined:

Parameter	Description
`EyeOpeningTimeSpan`	The *Time Eye Opening* for a terminal is the maximum extension of the BER threshold contour line in sample delay direction.
`EyeOpeningVoltageSpan`	The *Voltage Eye Opening* for a terminal is the maximum extension of the BER threshold contour line in sample voltage direction.
`SampleDelay`	The *Optimal Sample Delay* for a terminal is the sample delay coordinate of the center of a bounding box around the BER threshold contour line.
`SampleVoltage`	The *Optimal Sample Voltage* for a terminal is the sample voltage coordinate of the center of a bounding box around the BER threshold contour line.
`AllAnalysis`	All the defined values.

Remarks The criteria will always pass if the pass/fail criterion has been turned off using the properties `UseMinTimeEyeOpening`, `UseMinVoltsEyeOpening`, `UseSampleDelay`, and `UseSampleVoltage`.

For example, if `UseMinTimeEyeOpening = False`, `GetPortPassValue` will always return `True` regardless of the criteria.

Related functions and methods

"GetPortPassValue" on page 89
"GetTermPassValue" on page 91
"UseMinTimeEyeOpening" on page 60
"UseMinVoltsEyeOpening" on page 61
"UseSampleDelay" on page 62
"UseSampleVoltage" on page 63

GetPortPassValue

ActiveX syntax

```
bPass = Object.GetPortPassValue(nPortID,
                                nAnalysisTerm,
                                plsValid)
```

Wrapper dll syntax

```
EYEGetPortPassValue(hMeasurement,
                    nPortID,
                    nAnalysisTerm,
                    *plsValid,
                    *bPass)
```

Description Returns whether a measurement parameters has passed or failed the pass/fail criterion. The criteria are set by separate pass/fail methods.

Output parameter **bPass** Returns whether the designated measurement parameter passed or failed. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Port measurement value passed.
`False`	Port measurement value failed.

For the wrapper dll GET function, this parameter is a pointer.

Input parameters **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1.

nAnalysisTerm The following constants (data type: `ANALYSIS_TERM`) are defined:

Parameter	Description
`EyeOpeningTimeSpan`	The *Time Eye Opening* for a terminal is the maximum extension of the BER threshold contour line in sample delay direction.
`EyeOpeningVoltageSpan`	The *Voltage Eye Opening* for a terminal is the maximum extension of the BER threshold contour line in sample voltage direction.
`SampleDelay`	The *Optimal Sample Delay* for a terminal is the sample delay coordinate of the center of a bounding box around the BER threshold contour line.
`SampleVoltage`	The *Optimal Sample Voltage* for a terminal is the sample voltage coordinate of the center of a bounding box around the BER threshold contour line.
`AllAnalysis`	All the defined values.

plsValid Returns whether the value returned is valid for the measurement. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Data is valid for the measurement.
`False`	Data is not valid for the measurement.

For the wrapper dll GET function, this parameter is a pointer.

Remarks The criteria will always pass if the pass/fail criterion has been turned off using the properties `UseMinTimeEyeOpening`, `UseMinVoltsEyeOpening`, `UseSampleDelay`, and `UseSampleVoltage`.
For example, if `UseMinTimeEyeOpening = False`, `GetPortPassValue` will always return `True` regardless of the criteria.

You have to check if `bValid = True` before reporting or using the returned value.

Example To check the pass/fail criterion *EyeOpeningTimeSpan* for port 1:

```
Dim bPass, bValid as Boolean
bPass = m_EyeOpeningCTRL.GetPortPassValue(1, EyeOpeningTimeSpan,
bValid)
```

Related functions and methods
"GetMeasPassValue" on page 87
"GetTermPassValue" on page 91
"UseMinTimeEyeOpening" on page 60
"UseMinVoltsEyeOpening" on page 61
"UseSampleDelay" on page 62
"UseSampleVoltage" on page 63

GetTermPassValue

ActiveX syntax

```
bPass =  Object.GetTermPassValue(nPortID,
                                 nTermID,
                                 nAnalysisTerm,
                                 bValid)
```

Wrapper dll syntax

```
EYEGetTermPassValue(hMeasurement,
                    nPortID,
                    nTermID,
                    nAnalysisTerm,
                    *bValid,
                    *bPass)
```

Description

Returns whether the measured value of the object passed or failed its pass/fail criterion. The criteria are set by separate pass/fail methods.

For more information on how to set pass/fail criterion, refer to *How to Set Pass/Fail Criteria* in the *Eye Opening Measurement User Guide*.

NOTE Before `GetTermPassValue` can return a valid result, the following methods have to be called:

- `UseEyeOpeningPassFail(TRUE);`

  Turns on all the pass/fail checks.

- `UseSampleDelay(TRUE);`

  Turns on the pass/fail check for the measurement parameter `Optimal Sample Delay`.

- `SetMaxSampleDelay(dValue, eUnits);`

  Sets the pass/fail criterion for the measurement parameter *Optimal Sample Delay.*

If these values are not correctly set, `GetTermPassValue` always returns `TRUE`.

Output parameters

bPass Returns if the designated measurement parameter passed or failed. The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Terminal measurement value passed.
`False`	Terminal measurement value failed.

For the wrapper dll GET function, this parameter is a pointer.

bValid Returns (data type: `Boolean`) whether the value returned is valid for the measurement. There are several cases where the value may not be valid: For example, `ClockToMin` and `ClockToMax` are not valid when there is no clock signal defined. For the wrapper dll GET function, this parameter is a pointer.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

nPortID A port is addressed by the port number (data type: `Integer`). This is an index starting at 1 for each clock group.

nTermID A terminal is addressed by the terminal number (data type: `Integer`). This is an index starting at 1 for each port.

nAnalysisTerm Defines the parameter to be checked. The following constants (data type: `ANALYSIS_TERM`) are defined:

Constant	Equivalent to parameter...
`EyeOpeningTimeSpan`	Time Eye Opening
`EyeOpeningVoltageSpan`	Voltage Eye Opening
`SampleDelay`	Optimal Sample Delay
`SampleVoltage`	Optimal Sample Voltage
`AllAnalysis`	All the defined values.

For more information on the parameters, refer to *Result Parameter Definitions* in the *Eye Opening Measurement User Guide*.

Remarks

The parameter will always pass if its pass/fail criterion has been turned off using the properties `UseMinTimeEyeOpening`, `UseMinVoltsEyeOpening`, `UseSampleDelay`, or `UseSampleVoltage`.
For example, if `UseSampleVoltage = False`, `GetTermPassValue` will always return `True` regardless of the specified criteria.

You have to check if `bValid` = `True` before reporting or using the returned value.

Example

To check the pass/fail criterion *SampleDelay* for terminal 3 of port 1:

```
m_EyeOpeningCTRL.SetUseEyeOpeningPassFail(TRUE);
m_EyeOpeningCTRL.SetUseSampleDelay(TRUE);
dVal=m_EyeOpeningCTRL.GetTermCalculatedValue(1,3,SampleDelay,&bBool);
m_EyeOpeningCTRL.SetMaxSampleDelay(1e-15,Seconds);
bPass1=m_EyeOpeningCTRL.GetTermPassValue(1,3,SampleDelay, &bBool);
m_EyeOpeningCTRL.SetMaxSampleDelay(1e-6,Seconds);
bPass2=m_EyeOpeningCTRL.GetTermPassValue(1,3,SampleDelay, &bBool);
```

Related functions and methods	*"GetMeasPassValue" on page 87*
	"GetPortPassValue" on page 89
	"MinTimeEyeOpening" on page 45
	"MinVoltsEyeOpening" on page 46
	"MinSampleDelay" on page 43
	"MaxSampleVoltage" on page 42
	"MinSampleVoltage" on page 44
	"MaxSampleDelay" on page 41

Copy/Paste Functions

The following table gives an overview on the methods, events and properties available to cut, copy and delete measurement results:

Purpose	**Refer to...**
To copy data to the clipboard.	*"CopyToClipboard" on page 95*
To cut data from the measurement and copy the data to the clipboard.	*"CutToClipboard" on page 96*
To delete data from the numerical view.	*"EditDelete" on page 97*
To determine if a copy function call is possible.	*"IsCopyAvailable" on page 97*
To determine if a cut function call is possible.	*"IsCutAvailable" on page 98*
To determine if a delete function call is possible.	*"IsEditDeleteAvailable" on page 98*
To determine if a paste function call is possible.	*"IsPasteAvailable" on page 99*
To insert data previously copied to the clipboard.	*"PasteFromClipboard" on page 100*

CopyToClipboard

ActiveX syntax `Object.CopyToClipboard(boolean)`

NOTE This method is not available for the wrapper dll.

Description Copies the measurement window and the data of the numerical view to the clipboard.

To get the information from the clipboard, use *Paste Special* and then select *Enhanced Metafile* to get the graphics, *Unformatted Text* to get the data stored in the grid as comma delimited ASCII text, *HTML Format* to get the grid in HTML format. *Paste* returns the data displayed in the grid in HTML format.

In addition, data is stored in the clipboard in a MUI proprietary format that can be returned using the `PasteFromClipboard` method.

Use the `IsCopyAvailable` method to determine if a copy operation can be performed before calling `CopyToClipboard`.

Input parameter **boolean** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	The clipboard is cleared and the measurement window is copied to the clipboard. In addition, the measurement data in the numerical view is copied to the clipboard.
`False`	The clipboard is not cleared. However, the measurement window and measurement data are copied to the clipboard. Prior to this call, if there is other data formats in the clipboard, they will remain in the clipboard.

Example To clear the clipboard and copy measurement data to it:

```
m_EyeOpeningCTRL.CopyToClipBoard(True)
```

Related functions and methods *"CutToClipboard" on page 96*
"PasteFromClipboard" on page 100
"IsCopyAvailable" on page 97

CutToClipboard

ActiveX syntax `Object.CutToClipboard(boolean)`

NOTE This method is not available for the wrapper dll.

Description Cuts the measurement data from the dialog and copies the data to the clipboard. To get the information from the clipboard, use *Paste Special* and then select *Enhanced Metafile* to return the graphics, *Unformatted Text* to get the data stored in the grid as comma delimited ASCII text, *HTML Format* to get the grid in HTML format. *Paste* returns the data displayed in the grid as HTML.

In addition, data is stored in the clipboard in a proprietary format that can be returned using the `PasteFromClipboard` method.

Use the `IsCutAvailable` method to determine if a cut operation can be performed before calling `CutToClipboard`.

Input parameter **boolean** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	The clipboard is cleared out and the measurement window is copied to the clipboard. In addition the measurement data in the grid is copied into the clipboard.
`False`	The clipboard is not cleared out, however the measurement window and measurement data are copied into the clipboard. Prior to this call, if there is other data formats in the clipboard, they will remain in the clipboard.

Example To clear the clipboard, remove data from the measurement and copy the data to the clipboard:

```
m_EyeOpeningCTRL.CutToClipBoard(True)
```

EditDelete

ActiveX syntax `Object.EditDelete(boolean)`

NOTE This method is not available for the wrapper dll.

Description Deletes copied data in the grid. What is deleted depends on where the focus is. If there is no focus then nothing is deleted. If there is focus on the copied data line then this data will be deleted.

Use the `IsEditDeleteAvailable` method to determine if deleting is possible before calling `EditDelete`.

Input parameter **boolean** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Deletes copied data.
`False`	Reserved for future use.

Example To delete copied data from the numerical view of the measurement:

```
m_EyeOpeningCTRL.EditDelete(True)
```

Related functions and methods *“IsEditDeleteAvailable” on page 98*

IsCopyAvailable

ActiveX syntax `bCopy = Object.IsCopyAvailable()`

NOTE This method is not available for the wrapper dll.

Description Returns whether a copy function can be called.

Output parameter **bCopy** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Copy is a valid operation to call.
`False`	Copy is not a valid operation to call.

Example To check whether a copy operation is possible:

```
Dim bCopy as Boolean
bCopy = m_EyeOpeningCTRL.IsCopyAvailable()
```

Related functions and methods *“IsCutAvailable” on page 98*

IsCutAvailable

ActiveX syntax `bCut = Object.IsCutAvailable()`

NOTE This method is not available for the wrapper dll.

Description Returns whether a cut function can be called.

Output parameter **bCut** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Cut operation can be called.
`False`	Cut operation can not be called.

Example To check whether a cut operation is possible:

```
Dim bCut as Boolean
bCut = m_EyeOpeningCTRL.IsCutAvailable()
```

Related functions and methods *“IsCopyAvailable” on page 97*

IsEditDeleteAvailable

ActiveX syntax `bDelete = Object.IsEditDeleteAvailable()`

NOTE This method is not available for the wrapper dll.

Description Returns whether a delete function can be called.

Output parameter **bDelete** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Delete is a valid operation to call.
`False`	Delete is not a valid operation to call.

Example To check whether a delete operation is possible:

```
Dim bDelete as Boolean
bDelete = m_EyeOpeningCTRL.IsEditDeleteAvailable()
```

IsPasteAvailable

ActiveX syntax

```
bPaste = Object.IsPasteAvailable()
```

NOTE This method is not available for the wrapper dll.

Description Returns whether a paste function can be called. If there is anything on the clipboard, this function returns `True`.

Output parameter **bPaste** The following constants (data type: `Boolean`) will be returned:

Constant	Description
`True`	Paste operation can be called.
`False`	Paste operation can not be called.

Example To check whether a paste operation is possible:

```
Dim bPaste as Boolean
bPaste = m_EyeOpeningCTRL.IsPasteAvailable()
```

PasteFromClipboard

ActiveX syntax `Object.PasteFromClipboard(boolean)`

NOTE This method is not available for the wrapper dll.

Description Pastes information into the grid. What is pasted depends on what was selected during the copy operation:

- If there is no focus during the copy operation, the paste will copy all of the rows into the grid.
- If the focus was on the measurement line, the paste will copy all of the rows into the grid.
- If the focus was on an individual port, only that port will be pasted.
- If the focus was on an individual terminal, only the results of the terminal will be pasted.

Before calling `PasteFromClipboard`, use `IsPasteAvailable` to determine if a paste operation can be performed.

Input parameter **boolean** The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	Copies the information from the clipboard and then clears the clipboard.
`False`	Copies the information from the clipboard. The information remains in the clipboard.

Related functions and methods *"CopyToClipboard" on page 95*
"CutToClipboard" on page 96

Persistence

The following section shows the functions used to load/save measurements and to export data from the measurements.

Functions to Load/Save Measurements

The following table gives an overview on the methods, events and properties available to get and calculate measurement results:

Purpose	**Refer to...**
To load a stored measurement.	*"LoadMeasurement" on page 102*
To save a measurement.	*"SaveMeasurement" on page 113*

Export Data from Measurements

The following table gives an overview on the methods, events and properties available to export measurement results:

Purpose	**Refer to...**
To export data to the clipboard or to a file.	*"ExecuteExport" on page 103*
To specify if the calculated results or the raw data will be exported.	*"ExportDataType" on page 104*
To set the delimiter for the data export.	*"ExportDelimiter" on page 105*
To set the file name the data will be exported to.	*"ExportFileName" on page 106*
To set the date format for export.	*"ExportLocale" on page 107*
To specify if the results will be exported to file or clipboard.	*"ExportToClipboard" on page 108*
To export only results for expected 0s.	*"ExportUse0s" on page 109*
To export only results for expected 1s.	*"ExportUse1s" on page 110*
To export results for expected 0s and expected 1s.	*"ExportUseAll1s0s" on page 111*
To indicate if the data point was extrapolated.	*"ExportUseExtrapolatedFlag" on page 112*

LoadMeasurement

ActiveX syntax

```
Object.LoadMeasurement(sFileName)
```

Wrapper dll syntax

```
EYELoadMeasurement(hMeasurement,
                   sFileName)
```

Description Loads the measurement stored in the designated file.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

sFileName Full path and name of the stored measurement file (data type: `String`). The MUI application stores these files with the extension `.mcp`.

Example To load the measurement `EyeOpening.mcp`:

```
m_EyeOpeningCTRL.LoadMeasurement("C:\\Temp\\EyeOpening.mcp")
```

ExecuteExport

ActiveX syntax `Object.ExecuteExport()`

Wrapper dll syntax `EYEExecuteExport(hMeasurement)`

Description Exports the measurement data to the clipboard or to a file. The format of the export is determined by the Export properties: `ExportDataType`, `ExportLocale`, `ExportDelimiter`, `ExportFileName`, `ExportToClipboard`, `ExportUse0s`, `ExportUse1s`, `ExportUseAll1s0s`, and `ExportUseExtrapolatedFlag`.

Input parameter **hMeasurement** Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

NOTE The exported data is exported as raw data, meaning that the time value is reported in absolute seconds. The TimingUnits and SamplingDelay properties are **ignored**.

Related functions and methods

“ExportDataType” on page 104
“ExportLocale” on page 107
“ExportDelimiter” on page 105
“ExportFileName” on page 106
“ExportToClipboard” on page 108
“ExportUse0s” on page 109
“ExportUse1s” on page 110
“ExportUseAll1s0s” on page 111
“ExportUseExtrapolatedFlag” on page 112
“TimingUnits” on page 56
“SamplingDelay” on page 52

ExportDataType

ActiveX syntax

```
Object.ExportDataType = eExportData
```

For Visual C:

```
Object.SetExportDataType(eExportData)
eExportData = Object.GetExportDataType()
```

Wrapper dll syntax

```
EYEGetExportDataType(hMeasurement,
                          *eExportData)
EYESetExportDataType(hMeasurement,
                          eExportData)
```

Description

Determines what data will be exported to the clipboard or file.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

eExportData The following constants (data type: `EXPORT_DATA`) are defined:

Constant	Description
`BER`	Exports only the BER data.
`ALL`	Exports the BER, compared bits and errors data.

For the wrapper dll GET function, this parameter is a pointer.

Related functions and methods

“ExecuteExport” on page 103
“ExportFileName” on page 106
“ExportLocale” on page 107
“ExportToClipboard” on page 108
“ExportUse0s” on page 109
“ExportUse1s” on page 110
“ExportUseAll1s0s” on page 111
“ExportUseExtrapolatedFlag” on page 112

ExportDelimiter

ActiveX syntax

```
Object.ExportDelimiter = [sDelimiter]
```

For Visual C:

```
Object.SetExportDelimiter(sDelimiter)
sDelimiter = Object.GetExportDelimiter()
```

Wrapper dll syntax

```
EYEGetExportDelimiter(hMeasurement,
                      bufferSize,
                      *sDelimiter)
EYESetExportDelimiter(hMeasurement,
                      sDelimiter)
```

Description

Sets the delimiter for export.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

sDelimiter The delimiter (data type: `String`) that will be used between data elements in the export. The delimiter can be a space, tab or any writable character, for example, "," or "~". For the wrapper dll GET function, this parameter is a pointer.

Example

To set the delimiter between the data to "^":

```
m_EyeOpeningCTRL.ExportDelimiter = "^"
```

Related functions and methods

"ExecuteExport" on page 103
"ExportDataType" on page 104
"ExportFileName" on page 106
"ExportLocale" on page 107
"ExportToClipboard" on page 108
"ExportUse0s" on page 109
"ExportUse1s" on page 110
"ExportUseAll1s0s" on page 111
"ExportUseExtrapolatedFlag" on page 112

ExportFileName

ActiveX syntax

```
Object.ExportFileName = [sFileName]
```

For Visual C:

```
Object.SetExportFileName(sFileName)
sFileName = Object.GetExportFileName()
```

Wrapper dll syntax

```
EYEGetExportFileName(hMeasurement,
                     *sFileName)
EYESetExportFileName(hMeasurement,
                     sFileName)
```

Description Sets the file name and the directory data will be exported to.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

sFileName Directory and name of the file that data will be exported to (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Example To export data to the file "`C:\Temp\export.txt`":

```
m_EyeOpeningCTRL.ExportFileName = "C:\\Temp\\export.txt"
```

Related functions and methods

"ExecuteExport" on page 103
"ExportDataType" on page 104
"ExportDelimiter" on page 105
"ExportLocale" on page 107
"ExportToClipboard" on page 108
"ExportUse0s" on page 109
"ExportUse1s" on page 110
"ExportUseAll1s0s" on page 111
"ExportUseExtrapolatedFlag" on page 112

ExportLocale

ActiveX syntax

```
Object.ExportLocale = [sLocale]
```

For Visual C:

```
Object.SetExportLocale(sLocale)
sLocale = Object.GetExportLocale()
```

Wrapper dll syntax

```
EYEGetExportLocale(hMeasurement,
                   *sLocale)
EYESetExportLocale(hMeasurement,
                   sLocale)
```

Description

Sets the locale for the export which will define the date format and the default delimiter.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

sLocale Name of the default language of the export format. This will define the date format and the default delimiter (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Related functions and methods

"ExecuteExport" on page 103
"ExportDataType" on page 104
"ExportDelimiter" on page 105
"ExportFileName" on page 106
"ExportToClipboard" on page 108
"ExportUse0s" on page 109
"ExportUse1s" on page 110
"ExportUseAll1s0s" on page 111
"ExportUseExtrapolatedFlag" on page 112

ExportToClipboard

ActiveX syntax

```
Object.ExportToClipboard = [boolean]
```

For Visual C:

```
Object.SetExportToClipboard(boolean)
boolean = Object.GetExportToClipboard()
```

Wrapper dll syntax

```
EYEGetExportToClipboard(hMeasurement,
                        *boolean)
EYESetExportToClipboard(hMeasurement,
                        boolean)
```

Description

Sets whether the export will go to the clipboard or not.

Parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The settings for boolean (data type: `Boolean`) are:

Constant	Description
`True`	Turns on the export to the clipboard.
`False`	Turns off the export to the clipboard (default setting). Data will be exported to file.

For the wrapper dll GET function, this parameter is a pointer.

Example

To export data to the clipboard:

```
m_EyeOpeningCTRL.ExportToClipboard = False
```

Related functions and methods

“ExecuteExport” on page 103
“ExportDataType” on page 104
“ExportDelimiter” on page 105
“ExportFileName” on page 106
“ExportUse0s” on page 109
“ExportUse1s” on page 110
“ExportUseAll1s0s” on page 111
“ExportUseExtrapolatedFlag” on page 112

ExportUse0s

ActiveX syntax

```
Object.ExportUse0s = [boolean]
```

For Visual C:

```
Object.SetExportUse0s(boolean)
boolean = Object.GetExportUse0s()
```

Wrapper dll syntax

```
EYEGetExportUse0s(hMeasurement,
                  *boolean)
EYESetExportUse0s(hMeasurement,
                  boolean)
```

Description

Sets whether the export will contain data elements for errors from expected zeros.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The settings for boolean (data type: `Boolean`) are:

Constant	Description
`True`	BER (calculated from errors from zeros) and the errors from expected zeros will be included in the export.
`False`	BER (calculated from errors from zeros) and errors from zeros will *not* be included in the export (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Related functions and methods

“ExecuteExport” on page 103
“ExportDataType” on page 104
“ExportDelimiter” on page 105
“ExportFileName” on page 106
“ExportToClipboard” on page 108
“ExportUse1s” on page 110
“ExportUseAll1s0s” on page 111
“ExportUseExtrapolatedFlag” on page 112

ExportUse1s

ActiveX syntax

```
Object.ExportUse1s = [boolean]
```

For Visual C:

```
Object.SetExportUse1s(boolean)
boolean = Object.GetExportUse1s()
```

Wrapper dll syntax

```
EYEGetExportUse1s(hMeasurement,
                  *boolean)
EYESetExportUse1s(hMeasurement,
                  boolean)
```

Description Sets whether the export will contain data elements for errors from ones.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The settings for boolean (data type: `Boolean`) are:

Constant	Description
`True`	The BER (calculated from errors from ones) and errors from ones will be included in the export.
`False`	The BER (calculated from errors from ones) and errors from ones will not be included in the export (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Related functions and methods

"ExecuteExport" on page 103
"ExportDataType" on page 104
"ExportDelimiter" on page 105
"ExportFileName" on page 106
"ExportUse0s" on page 109
"ExportToClipboard" on page 108
"ExportUseAll1s0s" on page 111
"ExportUseExtrapolatedFlag" on page 112

ExportUseAll1s0s

ActiveX syntax

```
Object.ExportUseAll1s0s = [boolean]
```

For Visual C:

```
Object.SetExportUseAll1s0s(boolean)
boolean = Object.GetExportUseAll1s0s()
```

Wrapper dll syntax

```
EYEGetExportUseAll1s0s(hMeasurement,
                       *boolean)
EYESetExportUseAll1s0s(hMeasurement,
                       boolean)
```

Description Sets whether the export will contain data elements for all errors.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The settings for boolean (data type: `Boolean`) are:

Constant	Description
`True`	Total BER and all errors will be included in the export (default setting).
`False`	Total BER and all errors will not be included in the export.

For the wrapper dll GET function, this parameter is a pointer.

Related functions and methods

"ExecuteExport" on page 103
"ExportDataType" on page 104
"ExportDelimiter" on page 105
"ExportFileName" on page 106
"ExportUse0s" on page 109
"ExportToClipboard" on page 108
"ExportUse1s" on page 110
"ExportUseExtrapolatedFlag" on page 112

ExportUseExtrapolatedFlag

ActiveX syntax

```
Object.ExportUseExtrapolatedFlag = [boolean]
```

For Visual C:

```
Object.SetExportUseExtrapolatedFlag(boolean)
boolean = Object.GetExportUseExtrapolatedFlag()
```

Wrapper dll syntax

```
EYEGetExportUseExtrapolatedFlag(hMeasurement,
                                *boolean)
EYESetExportUseExtrapolatedFlag(hMeasurement,
                                boolean)
```

Description

Sets whether the export will contain the extrapolated flag indicating that the data point was not calculated but extrapolated by the *81200 Firmware Server.*

NOTE

The *Extrapolated Flag* is for future use. For the moment, it is ignored (always 0).

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

boolean The following constants (data type: `Boolean`) are defined:

Constant	Description
`True`	The extrapolated flag will be included in the export.
`False`	The extrapolated flag will not be included in the export (default setting).

For the wrapper dll GET function, this parameter is a pointer.

Related functions and methods

"ExecuteExport" on page 103
"ExportDataType" on page 104
"ExportDelimiter" on page 105
"ExportFileName" on page 106
"ExportUse0s" on page 109
"ExportToClipboard" on page 108
"ExportUseAll1s0s" on page 111
"ExportUse1s" on page 110

SaveMeasurement

ActiveX syntax

```
Object.SaveMeasurement(sFileName)
```

Wrapper dll syntax

```
EYESaveMeasurement(hMeasurement,
                   sFileName)
```

Description

Saves the measurement into a designated file.

Input parameters

hMeasurement Only for the wrapper dll: Handle to the measurement created by `InitMeasurement` (data type: `ViSession`).

sFileName Full path and name of the stored measurement file (data type: `String`). The MUI application stores these files with the extension `.mcp`.

Example

To save the measurement `EyeOpening.mcp`:

```
m_EyeOpeningCTRL.SaveMeasurement("C:\\Temp\\EyeOpening.mcp")
```

Functions for General Purposes

The following sections show the functions used to access the online help, for example.

The following table gives an overview on the methods, events and properties available:

Purpose	**Refer to...**
To set the background color of the graphical view.	*"BackColor" on page 115*
To set the color of the BER threshold marker.	*"BERMarkerColor" on page 116*
To set the text color of the graphical display.	*"ForeColor" on page 117*
To set/return the name and the path of the help file.	*"MeasHelpPath" on page 118*
To set the help ID for the measurement window.	*"MeasureWinHelp" on page 118*

BackColor

ActiveX syntax

```
Object.BackColor = [nColor]
```

For Visual C:

```
Object.SetBackColor(nColor)
nColor = Object.GetBackColor()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns the background color of the graphical view.

Parameter **nColor** Sets the background color (data type: `integer`). The following table lists typical color values:

Color	RGB Values	nColor Value
White	255, 255, 255	16777215
Black	0, 0, 0	0
Gray	192, 192, 192	12632256
Dark gray	128, 128, 128	8421504
Red	255, 0, 0	255
Dark red	128, 0, 0	128
Yellow	255, 255, 0	65535
Dark yellow	128, 128, 0	32896
Green	0, 255, 0	65280
Dark green	0, 128, 0	32768
Cyan	0, 255, 255	16776960
Dark cyan	0, 128, 128	8421376
Blue	0, 0, 255	16711680
Dark blue	0, 0, 128	8388608
Magenta	255, 0, 255	16711935
Dark magenta	128, 0, 128	8388736

Example To set the background color of the graphical view to "Blue":

```
m_EyeOpeningCTRL.BackColor = 16711680
```

Related functions and methods *"ForeColor" on page 117*

BERMarkerColor

ActiveX syntax

```
Object.BERMarkerColor = [nColor]
```

For Visual C:

```
Object.SetBERMarkerColor(nColor)
nColor = Object.GetBERMarkerColor()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns the color of the BER threshold marker.

Parameter **nColor** Sets the BER threshold marker color (data type: `integer`). The following table lists typical color values:

Color	RGB Values	nColor Value
White	255, 255, 255	16777215
Black	0, 0, 0	0
Gray	192, 192, 192	12632256
Dark gray	128, 128, 128	8421504
Red	255, 0, 0	255
Dark red	128, 0, 0	128
Yellow	255, 255, 0	65535
Dark yellow	128, 128, 0	32896
Green	0, 255, 0	65280
Dark green	0, 128, 0	32768
Cyan	0, 255, 255	16776960
Dark cyan	0, 128, 128	8421376
Blue	0, 0, 255	16711680
Dark blue	0, 0, 128	8388608
Magenta	255, 0, 255	16711935
Dark magenta	128, 0, 128	8388736

Example To set the color of the BER threshold marker to "Red":

```
m_EyeOpeningCTRL.BERMarkerColor = 255
```

Related functions and methods *“BERThreshold” on page 31*

ForeColor

ActiveX syntax

```
Object.ForeColor = [nColor]
```

For Visual C:

```
Object.SetForeColor(nColor)
nColor = Object.GetForeColor()
```

NOTE This property is not available for the wrapper dll.

Description Sets the foreground color (text) of the graph.

Parameter **nColor** Sets the foreground color (data type: `integer`). The following table lists typical color values:

Color	RGB Values	nColor Value
White	255, 255, 255	16777215
Black	0, 0, 0	0
Gray	192, 192, 192	12632256
Dark gray	128, 128, 128	8421504
Red	255, 0, 0	255
Dark red	128, 0, 0	128
Yellow	255, 255, 0	65535
Dark yellow	128, 128, 0	32896
Green	0, 255, 0	65280
Dark green	0, 128, 0	32768
Cyan	0, 255, 255	16776960
Dark cyan	0, 128, 128	8421376
Blue	0, 0, 255	16711680
Dark blue	0, 0, 128	8388608
Magenta	255, 0, 255	16711935
Dark magenta	128, 0, 128	8388736

Example To set the text color of the graphical view to "Black":

```
m_EyeOpeningCTRL.ForeColor = 0
```

Related functions and methods *“BackColor” on page 115*

MeasHelpPath

ActiveX syntax

```
Object.MeasHelpPath = [sHelpPath]
```

For Visual C:

```
Object.SetMeasHelpPath(sHelpPath)
sHelpPath = Object.GetMeasHelpPath()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns the default help file name.

Parameter **sHelpPath** Full path and name of help file (data type: `String`).

Example To set the path to the help file to "`C:\Temp\tmEyeOpeningM.chm`":

```
m_EyeOpeningCTRL.MeasHelpPath = "C:\\Temp\\tmEyeOpeningM.chm"
```

MeasureWinHelp

ActiveX syntax

```
Object.MeasureWinHelp(lWndHandle,
                      sHelpPath,
                      lCommand,
                      lData)
```

NOTE This method is not available for the wrapper dll.

Input parameters **lWndHandle** Window handle for the parent window (data type: `Long`).

sHelpPath Path and name of the controls help file (data type: `String`). The file must have a ".chm" extension. Can be obtained using the `MeasHelpPath` property.

lCommand Value should be set to 1 for context help support (data type: `Long`). Not supporting any other help at this time.

lData Default control help ID, 131172 (data type: `Long`). Can be obtained using the `DefaultHelpID` property.

Example To call the online help with the default help ID:

```
Dim helpstr as String
Dim helpID as Long
Dim frmMain as Form
helpstr = m_EyeOpeningCTRL.MeasHelpPath
helpID = m_EyeOpeningCTRL.DefaultHelpID
m_EyeOpeningCTRL.MeasureWinHelp(frmMain.hWnd, helpstr, 1, helpID)
```

Index