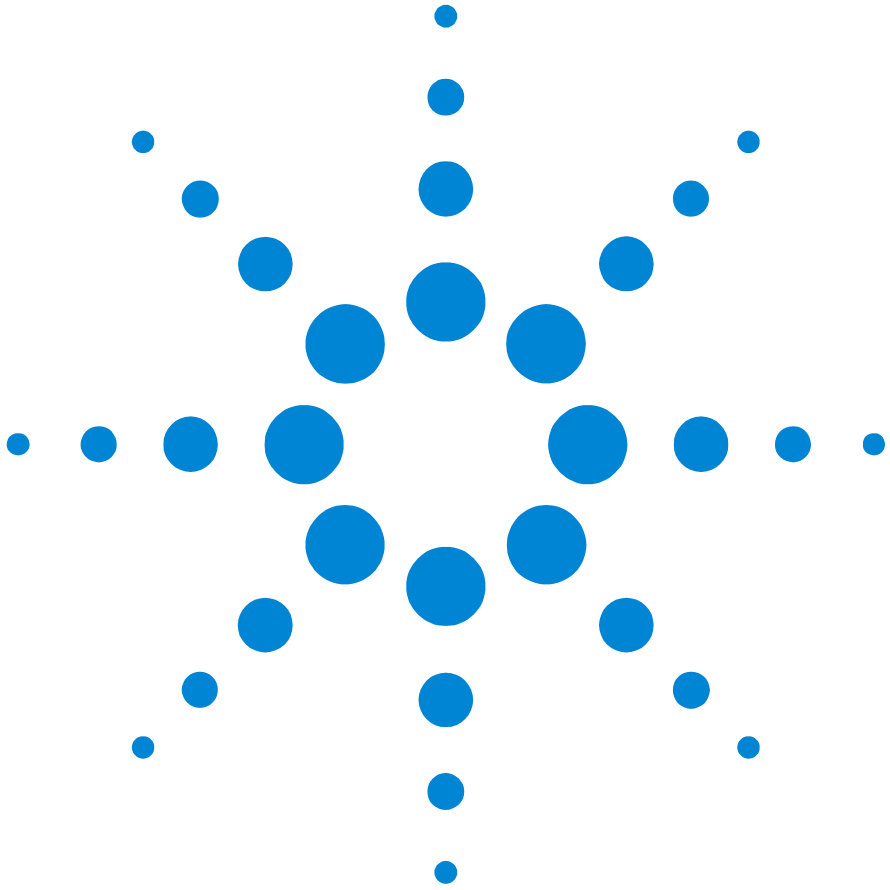

Agilent 81250 ParBERT Measurement Software

Spectral Jitter Measurement Programming Reference

Agilent Technologies

Important Notice

Manual Part Number

5988-9645EN

Revision

Revision May 2003

Printed in Germany

Agilent Technologies
Herrenberger Straße 130
D-71034 Böblingen
Germany

Authors: t3 medien GmbH

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Trademarks

Windows NT ® and MS Windows ® are U.S. registered trademarks of Microsoft Corporation.

Contents

Contents

About this Reference

This document describes the functions, properties and methods for controlling the Spectral Jitter 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 *Spectral Jitter 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 33* shows the functions used to work with ports and terminals, and to set the parameters for the measurement.
- *"Running the Measurement" on page 68* shows the functions used to run and stop the measurement.
- *"Handling Events and Callbacks" on page 74* shows the functions used to handle events and callbacks for the measurement.
- *"Error Handling" on page 80* shows the functions used to analyze errors.
- *"Handling Measurement Results" on page 83* shows the functions used to get, calculate, and modify measurement results.
- *"Pass/Fail Functions" on page 93* shows the functions used to set and evaluate pass/fail decisions.
- *"Copy/Paste Functions" on page 100* shows the functions used to edit the data display.

- *"Persistence" on page 109* shows the functions used to load/save measurements and to export data from the measurements.
- *"Functions for General Purposes" on page 117* 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 14*
To initialize a measurement.	*"InitMeasurement" on page 25*

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 28*
To set the port the firmware server is connected to.	*"ServerPort" on page 29*
To specify the analyzer system.	*"AnalyzerSystem" on page 12*
To specify the generator system.	*"GeneratorSystem" on page 19*
To specify the name of a system to be delayed.	*"DelayStartSystem" on page 18*
To specify a start delay between two systems.	*"StartDelay" on page 30*
To create a new measurement.	*"CreateMeasurementEx" on page 16*
To prepare the measurement execution.	*"CreateMeasEx" on page 15*
To check whether the connection to the firmware server is valid.	*"IsFWSConnected" on page 26*

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 21*
To get the names of the system settings defined for the analyzer.	*"GetAnalyzerSettingsName" on page 22*
To get the number of system settings defined for the generator.	*"GetGeneratorSettingsCount" on page 23*
To get the names of the system settings defined for the generator.	*"GetGeneratorSettingsName" on page 24*
To specify the system setting for the analyzer.	*"AnalyzerSystemSetting" on page 13*
To specify the system setting for the generator.	*"GeneratorSystemSetting" on page 20*
To send the analyzer's system setting to the firmware server.	*"UseAnalyzerSettings" on page 31*
To send the generator's system setting to the firmware server.	*"UseGeneratorSettings" on page 32*

AnalyzerSystem

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITSetAnalyzerSystem(hMeasurement,
                         sSystem)
SPECJITGetAnalyzerSystem(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_SpectralJitterCTRL.AnalyzerSystem = "DSRA"
```

Related functions and methods

“AnalyzerSystemSetting” on page 13
“GeneratorSystem” on page 19
“CreateMeasurementEx” on page 16

AnalyzerSystemSetting

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITSetAnalyzerSystemSetting(hMeasurement,
                                sSetting)
SPECJITGetAnalyzerSystemSetting(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_SpectralJitterCTRL.AnalyzerSystemSetting = "TEST_APP"
```

Related functions and methods

"AnalyzerSystem" on page 12

CloseMeasurement

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

Wrapper dll syntax

```
SPECJITCloseMeasurement(hMeasurement)
```

Description Closes the measurement. The handle to the measurement is 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 `InitMeasurement`.

Related functions and methods

"CreateMeasurementEx" on page 16
"InitMeasurement" on page 25

CreateMeasEx

ActiveX syntax `Object.CreateMeasEx()`

Wrapper dll syntax `SPECJITCreateMeasEx(hMeasurement)`

Description Creates the connection to the server, sets the generator and analyzer systems, delay system, the delay, and the measurement type. It assumes that the server name, port number, analyzer and generator system, delay system, delay value, and measurement type 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`, `MeasurementType`

Related functions and methods

"Server" on page 28
"ServerPort" on page 29
"AnalyzerSystem" on page 12
"GeneratorSystem" on page 19
"DelayStartSystem" on page 18
"StartDelay" on page 30
"MeasurementType" on page 27

CreateMeasurementEx

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITCreateMeasurementEx(hMeasurement,
                           sServer,
                           sPort,
                           sAnalyzer,
                           sGenerator,
                           sDelaySystem,
                           dDelay,
                           measType)
```

Description

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

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`).

measType Measurement type: can be either `MeasTypeElectrical`, `MeasTypeOptical`, or `MeasTypeAll` (data type: `enum`).

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, for a system with only electrical connections:

```
m_SpectralJitterCTRL.CreateMeasurementEx("10.3.6.14","2203","DSRA",
                                         "DSRA","",0,MeasTypeElectrical)
```

Related functions and methods

"AnalyzerSystem" on page 12
"AnalyzerSystemSetting" on page 13
"DelayStartSystem" on page 18
"GeneratorSystem" on page 19
"GeneratorSystemSetting" on page 20
"MeasurementType" on page 27
"StartDelay" on page 30

DelayStartSystem

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITSetDelayStartSystem(hMeasurement,
                           sDelaySystem)
SPECJITGetDelayStartSystem(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_SpectralJitterCTRL.DelayStartSystem = "DSRA"
```

Related functions and methods

"AnalyzerSystem" on page 12
"GeneratorSystem" on page 19

GeneratorSystem

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITSetGeneratorSystem(hMeasurement,
                          sSystem)
SPECJITGetGeneratorSystem(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.

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 select the generator system "DSRA":

```
m_SpectralJitterCTRL.GeneratorSystem = "DSRA"
```

Related functions and methods

"AnalyzerSystem" on page 12
"GeneratorSystemSetting" on page 20

GeneratorSystemSetting

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITSetGeneratorSystemSetting(hMeasurement,
                                 sSetting)
SPECJITGetGeneratorSystemSetting(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_SpectralJitterCTRL.GeneratorSystemSetting = "TEST_APP"
```

Related functions and methods

“GeneratorSystem” on page 19

GetAnalyzerSettingsCount

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetAnalyzerSettingsCount(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 defined for the analyzer system:

```
Dim nItems as Integer
nItems = m_SpectralJitterCTRL.GetAnalyzerSettingsCount()
```

Related functions and methods

"GetAnalyzerSettingsName" on page 22

GetAnalyzerSettingsName

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetAnalyzerSettingsName(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 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_SpectralJitterCTRL.GetAnalyzerSettingsName(0)
```

Related functions and methods

"GetAnalyzerSettingsCount" on page 21

GetGeneratorSettingsCount

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetGeneratorSettingsCount(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 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_SpectralJitterCTRL.GetGeneratorSettingsCount()
```

GetGeneratorSettingsName

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetGeneratorSettingsName(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_SpectralJitterCTRL.GetGeneratorSettingsName(0)
```

InitMeasurement

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

Wrapper dll syntax

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

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`)

IsFWSConnected

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITIsFWSConnected(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.

For the wrapper dll 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 connection to the firmware server:

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

Related functions and methods

"Server" on page 28
"ServerPort" on page 29

MeasurementType

ActiveX syntax

```
Object.MeasurementType = [measType]
[measType] = Object.MeasurementType
```

For Visual C:

```
Object.SetMeasurementType(measType)
measType = Object.GetMeasurementType()
```

Wrapper dll syntax

```
SPECJITSetMeasurementType(hMeasurement,
                          measType)
SPECJITGetMeasurementType(hMeasurement,
                          *measType)
```

Description

Sets/returns the measurement's *type* (whether the measurement is electrical only, optical only, or a combination of both).

Parameters

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

measType Type of the measurement (data type: `enum`). The value can be either `MeasTypeElectrical`, `MeasTypeOptical`, or `MeasTypeAll` (both types in one system). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the measurement type for electrical only:

```
m_SpectralJitterCTRL.MeasurementType = MeasTypeElectrical
```

Related functions and methods

"CreateMeasurementEx" on page 16

Server

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITSetServer(hMeasurement,
                 sServer)
SPECJITGetServer(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_SpectralJitterCTRL.Server = "10.3.6.14"
```

Related functions and methods *"ServerPort" on page 29*

ServerPort

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITSetServerPort(hMeasurement,
                     bufferSize,
                     sPort)
SPECJITGetServerPort(hMeasurement,
                     *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_SpectralJitterCTRL.ServerPort = "2203"
```

Related functions and methods *"Server" on page 28*

StartDelay

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITGetStartDelay(hMeasurement,
                     *dDelay)
SPECJITSetStartDelay(hMeasurement,
                     dDelay)
```

Description

Sets the start delay between the two systems. The `DelayStartSystem` property defines which system is delayed, the generator or analyzer.

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 delay time to 0.5 seconds:

```
m_SpectralJitterCTRL.StartDelay = 0.5
```

Related functions and methods

"DelayStartSystem" on page 18

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 firmware server.

Parameters

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_SpectralJitterCTRL.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 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_SpectralJitterCTRL.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 43*
To get the names of analyzer ports configured in the system.	*"GetAnalyzerPortName" on page 44*
To get the number of analyzer terminals configured in the system.	*"GetAnalyzerTermCount" on page 45*
To get the name of one analyzer terminal.	*"GetAnalyzerTermName" on page 46*
To get a port id for a given index.	*"GetPortId" on page 47*
To get a terminal id for a given index and port.	*"GetTerminalId" on page 48*
To set/return if a given port is part of the measurement.	*"PortInvolved" on page 56*
To set/return if a given terminal is part of the measurement.	*"TermInvolved" on page 65*

Setting Spectral Jitter 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 and captured.	*"AcquisitionDepth" on page 35*
To set the analyzer sampling point offset.	*"SamplePointOffset" on page 59*
To obtain the list of available FFT windows.	*"FFTWindowList" on page 38*
To enable an FFT window.	*"FFTWindow" on page 37*

Setting the Spectral Jitter 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:

Purpose	Refer to...
To set / get which pass / fail parameters are considered when measuring Spectral Jitter.	*"PassFailUse" on page 51*
To set / get pass / fail values for the bit error rate.	*"PassFailValue" on page 53*
To set / get pass / fail maximum power limits.	*"PassFailValuePower" on page 54*

Setting Spectral Jitter 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 if absolute spectral power, true relative, or relative power is displayed.	*"ViewType" on page 67*
To set / get the reference frequency for a relative power measurement.	*"RelativeReferenceFrequency" on page 58*
To set / get the noise threshold.	*"NoiseThreshold" on page 50*
To set / get the number of top frequency / power pairs.	*"TopFrequencyCount" on page 66*
To define a frequency range for analysis.	*"FreqRange" on page 40*
To enable / disable a frequency range.	*"FreqRangeEnable" on page 42*
To set / get the frequency scale of the graph to logarithmic or linear.	*"Scale" on page 60*
To toggle the display of power values between linear and dB.	*"DisplayPowerUnits" on page 36*
To set min / max values for the frequency zoom function.	*"SetFreqAxisRange" on page 61*
To enable / disable the frequency zoom function.	*"FreqAxisRangeType" on page 39*
To display / hide markers in the graphical display.	*"ShowMarkers" on page 63*
To show or hide the grid of the graph.	*"ShowGrid" on page 62*
To set / get the number of decimal places to be displayed in the numerical view.	*"GridPrecision" on page 49*

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 57*
To display the *Properties* dialog.	*"ShowProperties" on page 64*

AcquisitionDepth

ActiveX syntax

```
Object.AcquisitionDepth = [eAcqDepth]
[eAcqDepth] = Object.AcquisitionDepth
```

For Visual C:

```
Object.SetAcquisitionDepth(eAcqDepth)
eAcqDepth = Object.GetAcquisitionDepth()
```

Wrapper dll syntax

```
SPECJITSetAcquisitionDepth(hMeasurement,
                    eAcqDepth)
SPECJITGetAcquisitionDepth(hMeasurement,
                    *eAcqDepth)
```

Description

Sets/returns the acquisition depth. This is the number of bits to be captured and compared and hence the length of the time record for the FFT. For details see the *Spectral Jitter Measurement User Guide*.

Parameters

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

eAcqDepth Specifies the acquisition depth (data type: `AcquisitonDepthEnums`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the acquisition depth to 512 Kbit:

```
m_SpectralJitterCTRL.AcquisitionDepth = AD_512_Kbit
```

Related functions and methods

"FFTWindowList" on page 38
"FFTWindow" on page 37

DisplayPowerUnits

ActiveX syntax

```
Object.DisplayPowerUnits = [ePowUnit]
```

For Visual C:

```
Object.SetDisplayPowerUnits(ePowUnit)
ePowUnit P= Object.GetDisplayPowerUnits()
```

NOTE This property is not available for the wrapper dll.

Description Sets the power unit to linear or dB.

Input parameter **ePowUnit** The following constants (data type: `PowerUnitsEnums`) are defined:

Constant	Description
`PowerLinear`	Displays unitless power values (linear).
`PowerDB`	Displays power values in dB (logarithmic).

Example To display power values in dB:

```
m_SpectralJitterCTRL.DisplayPowerUnits = PowerDB
```

Related functions and methods *"Scale" on page 60*

FFTWindow

ActiveX syntax

```
Object.FFTWindow = [sWindow]
[sWindow] = Object.FFTWindow
```

For Visual C:

```
Object.SetFFTWindow(sWindow)
sWindow = Object.GetFFTWindow()
```

Wrapper dll syntax

```
SPECJITSetFFTWindow(hMeasurement,
                    sWindow)
SPECJITGetFFTWindow(hMeasurement,
                    bufferSize,
                    *sWindow)
```

Description

Sets/returns the window used for the FFT. For details see the *Spectral Jitter Measurement User Guide*.

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`).

sWindow The name of the window (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Example

To enable the Hanning window:

```
m_SpectralJitterCTRL.FFTWindow = "Hanning"
```

Related functions and methods

"FFTWindowList" on page 38

FFTWindowList

ActiveX syntax

```
[sWindowlist] = Object.FFTWindowList
```

For Visual C:

```
sWindowlist = Object.FFTWindowlist()
```

Wrapper dll syntax

```
SPECJITFFTWindowList(hMeasurement,
                     bufferSize,
                     *sWindowlist)
```

Description

Returns the list of available FFT windows.

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`).

sWindowlist The list of available windows (data type: `String`). For the wrapper dll GET function, this parameter is a pointer.

Example

To get the list of available FFT windows:

```
Dim sWindows as String
sWindows = m_SpectralJitterCTRL.FFTWindowList()
```

The string may return: "Uniform, Hanning, Hamming, Blackman"

Related functions and methods

"FFTWindow" on page 37

FreqAxisRangeType

Syntax

```
Object.FreqAxisRangeType = [eAxisType]
```

For Visual C:

```
Object.SetFreqAxisRangeType(eAxisType)
eAxisType = Object.GetFreqAxisRangeType()
```

NOTE This property is not available for the wrapper dll.

Description Sets/gets the type of the graph's horizontal frequency axis. The frequency axis can show either the whole frequency range of the measurement or a section.

Parameter **eAxisType** The following constants (data type: `AxisFreqRangeTypeEnums`) are defined:

Constant	Description
`AFT_EntireRange`	The graph shows the whole frequency range.
`AFT_ZoomMinMax`	The graph zooms a range defined by a min and max frequency.

Example To set the graph's FreqAxisRangeType to show the whole frequency range:

```
m_SpectralJitterCTRL.FreqAxisRangeType = AFT_EntireRange
```

Related functions and methods

"SetFreqAxisRange" on page 61
"Scale" on page 60

FreqRange

ActiveX syntax

```
Object.SetFreqRange(sIndex,
                    eFreqRangeType,
                    eFreqRangePower,
                    dVal1,
                    dVal2)

boolean = Object.GetFreqRange(sIndex,
                    *eFreqRangeType,
                    *eFreqRangePower,
                    *dVal1,
                    *dVal2)
```

Wrapper dll syntax

```
SPECJITSetFreqRange(hMeasurement,
                    sIndex,
                    eFreqRangeType,
                    eFreqRangePower,
                    dVal1,
                    dVal2)
SPECJITGetFreqRange(hMeasurement,
                    sIndex,
                    *eFreqRangeType,
                    *eFreqRangePower,
                    *dVal1,
                    *dVal2)
```

Description Sets/returns the characteristics of a contiguous frequency range. For details see the Spectral Jitter Measurement User Guide.

Parameters

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

sIndex The identifier of the frequency range: 0 to 7 (data type: `VInt16`).

eFreqRangeType 0 = defined by middle and bandwidth, 1 = defined by min and max frequency (data type: `FrequencyRangeTypeEnums`).

eFreqRangePower 0 = total power, 1 = peak power (data type: `FrequencyRangePowerEnums`).

dVal1 First frequency range value – middle or minimum frequency, depending on `eFreqRangeType` (data type: `ViInt64`).

dVal2 Second frequency range value – bandwidth or maximum frequency, depending on `eFreqRangeType` (data type: `ViInt64`).

Example To define frequency range #1 with min/max (2.55 MHz to 2.65 MHz) and total power calculation:

```
m_SpectralJitterCTRL.FreqRange = 0,
                                 AD_512_Kbit,
                                 FFT_MinMax,
                                 FFT_TotalPower,
                                 2550000,
                                 2650000
```

Related functions and methods *"FreqRangeEnable" on page 42*

FreqRangeEnable

ActiveX syntax

```
Object.SetFreqRangeEnable (sIndex, bEnable)
boolean = Object.GetFreqRangeEnable (sIndex, bEnable)
```

Wrapper dll syntax

```
SPECJITSetFreqRangeEnable(hMeasurement,
                          sIndex,
                          bEnable)
SPECJITGetFreqRangeEnable(hMeasurement,
                          sIndex,
                          *bEnable)
```

Description

The Set command enables/disables a frequency range. The Get command returns the status of the frequency range.

Parameters

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

sIndex The identifier of the frequency range: 0 to 7 (data type: `ViInt16`).

bEnable True = enable, false = disable (data type: `Boolean`).

Example

To enable frequency range #1:

```
m_SpectralJitterCTRL.FreqRangeEnable = 0, true
```

Related functions and methods

"FreqRange" on page 40

GetAnalyzerPortCount

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetAnalyzerPortCount(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 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_SpectralJitterCTRL.GetAnalyzerPortCount()
```

Related functions and methods

"GetAnalyzerPortName" on page 44

GetAnalyzerPortName

ActiveX syntax

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

Wrapper dll syntax

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

Description

Returns the analyzer port name for a designated port id.

Output parameters

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

sName Name of the port as configured on the GUI (data type: `String`).

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

Example

To get the name of port 1:

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

Related functions and methods

"GetAnalyzerPortCount" on page 43

GetAnalyzerTermCount

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetAnalyzerTermCount(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`).

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

Example To get the number of terminals configured for port 1:

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

Related functions and methods *"GetAnalyzerTermName" on page 46*

GetAnalyzerTermName

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetAnalyzerTermName(hMeasurement,
                           nPortID,
                           nTerminalID,
                           bufferSize,
                           *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 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.

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

Example

To get the name of terminal 1:

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

Related functions and methods

"GetAnalyzerTermCount" on page 45

GetPortId

ActiveX syntax

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

Wrapper dll syntax

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

Example

To get the ID of the first port:

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

GetTerminalId

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetTerminalId(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 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_SpectralJitterCTRL.GetTerminalId(1, 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 possible values (data type: `PrecisionEnums`) used are the integers 0 – 10, written out (`Zero`, `One`, ... `Ten`).

Example To set the number of decimal places to be displayed to 2:

```
m_SpectralJitterCTRL.GridPrecision = Two
```

NoiseThreshold

ActiveX syntax

```
Object.SetNoiseThreshold(dValue, eUnit)
boolean = Object.GetNoiseThreshold(dValue, eUnit)
```

Wrapper dll syntax

```
SPECJITSetNoiseThreshold(hMeasurement,
                         dValue,
                         eUnit)
SPECJITGetNoiseThreshold(hMeasurement,
                         *dValue,
                         *eUnit)
```

Description

Sets/returns the noise threshold. This is the threshold used to separate between total power and noise power.

Parameters

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

dValue The threshold value (data type: `VIReal64`).

eUnit The power unit: Linear or dB (data type: `PowerUnitsEnums`).

Example

To set the noise threshold to –45 dB:

```
m_SpectralJitterCTRL.NoiseThreshold = -45, PowerDB
```

PassFailUse

ActiveX syntax

```
Object.SetPassFailUse(eTerm,
                      bUsed)

bUsed = Object.GetPassFailUse(eTerm)
```

Wrapper dll syntax

```
SPECJITSetPassFailUse(hMeasurement,
                      eTerm,
                      bUsed)

SPECJITGetPassFailUse(hMeasurement,
                      eTerm,
                      *bUsed)
```

Description This method allows you to set/get which pass/fail parameters are considered for the test.

NOTE This method only allows you to *define* whether a parameters is considered during the test. To *set* the pass/fail values, use the PassFailValue and PassFailValuePower methods.

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

eTerm Defines one Pass/Fail criterion for testing the DUT (data type: `PassFailUseEnums`). It can have one of the following values:

Constant	Description
`PFE_FreqRange1`	Test passes if the measured power is below limit.
`PFE_FreqRange2`	Test passes if the measured power is below limit.
`PFE_FreqRange3`	Test passes if the measured power is below limit.
`PFE_FreqRange4`	Test passes if the measured power is below limit.
`PFE_FreqRange5`	Test passes if the measured power is below limit.
`PFE_FreqRange6`	Test passes if the measured power is below limit.
`PFE_FreqRange7`	Test passes if the measured power is below limit.
`PFE_FreqRange8`	Test passes if the measured power is below limit.
`PFE_BitErrorRate`	Test passes if the BER stays within the limits.
`PFE_TotalPower`	Test passes if the total power is below limit.
`PFE_NoisePower`	Test passes if the noise power is below limit.

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

Constant	Description
`True`	Parameter is considered during the test.
`False`	Parameter is not considered during the test.

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

Example

To set up a test so that the total jitter power is evaluated:

```
With m_SpectralJitterCTRL
   .SetPassFailUse PFE_TotalPower, True
End With
```

Related functions and methods

"PassFailValue" on page 53
"PassFailValuePower" on page 54

PassFailValue

ActiveX syntax

```
Object.SetPassFailValue(eTerm,
                        dValue)

dValue = Object.GetPassFailValue(eTerm)
```

Wrapper dll syntax

```
SPECJITSetPassFailValue(hMeasurement,
                        eTerm,
                        dValue)

SPECJITGetPassFailValue(hMeasurement,
                        eTerm,
                        *dValue)
```

Description

Sets the pass/fail values for the bit error rate (BER).

Input parameters

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

eTerm Defines the parameter (data type: `PassFailValueEnums`). For the wrapper dll GET function, this parameter is a pointer. The following table indicates which parameters you can set with this command.

Constant	Description
`PFV_BitErrorRateMin`	Sets/gets the minimum value.
`PFV_BitErrorRateMax`	Sets/gets the maximum value.

dValue The pass/fail value used (data type: `VIReal64`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the maximum allowed BER to 0.27:

```
With m_SpectralJitterCTRL
   .SetPassFailUse PFE_BitErrorRate, True
   .SetPassFailValue PFV_BitErrorRateMax, 0.27
End With
```

Related functions and methods

"PassFailUse" on page 51
"PassFailValuePower" on page 54

PassFailValuePower

ActiveX syntax

```
Object.SetPassFailValuePower(eTerm,
                             eUnits,
                             dValue)

dValue = Object.GetPassFailValuePower(eTerm,
                             eUnits)
```

Wrapper dll syntax

```
SPECJITSetPassFailValuePower(hMeasurement,
                             eTerm,
                             eUnits,
                             dValue)

SPECJITGetPassFailValuePower(hMeasurement,
                             eTerm,
                             *eUnits,
                             *dValue)
```

Description

Sets the pass/fail power value for Total Power, Noise Power, or a frequency range.

Input parameters

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

eTerm Defines one Pass/Fail power limit (data type: `PassFailValueEnums`). The following table indicates which parameters you can set with this command.

Constant	Description
PFV_FreqRange1	Sets/gets the maximum power value.
PFV_FreqRange2	Sets/gets the maximum power value.
PFV_FreqRange3	Sets/gets the maximum power value.
PFV_FreqRange4	Sets/gets the maximum power value.
PFV_FreqRange5	Sets/gets the maximum power value.
PFV_FreqRange6	Sets/gets the maximum power value.
PFV_FreqRange7	Sets/gets the maximum power value.
PFV_FreqRange8	Sets/gets the maximum power value.
PFV_TotalPowerMax	Sets/gets the maximum power value.
PFV_NoisePowerMax	Sets/gets the maximum power value.

eUnits Unit used for the value (data type: `PowerUnitsEnums`). The unit can be `PowerLinear` or `PowerDB`. For the wrapper dll GET function, this parameter is a pointer.

dValue The value used for the parameter (data type: `ViReal64`). For the wrapper dll GET function, this parameter is a pointer.

Example To set the upper limit for frequency range #2 to –36 dB:

```
With m_SpectralJitterCTRL
   .SetPassFailUse PFE_FreqRange2, TRUE
   .SetPassFailValuePower PFV_FreqRange2, PowerDB, -36
End With
```

Related functions and methods

"PassFailUse" on page 51
"PassFailValue" on page 53

PortInvolved

ActiveX syntax

```
Object.SetPortInvolved(nPortID,
                       boolean)

boolean = Object.GetPortInvolved(nPortID)
```

Wrapper dll syntax

```
SPECJITSetPortInvolved(hMeasurement,
                       nPortID,
                       boolean)

SPECJITGetPortInvolved(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_SpectralJitterCTRL.SetPortInvolved(1, True)
```

To get whether port 1 is involved in the measurement:

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

Related functions and methods *"TermInvolved" on page 65*

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 `"Spectral Jitter"`:

```
m_SpectralJitterCTRL.PropertiesTitle = "Spectral Jitter"
```

RelativeReferenceFrequency

ActiveX syntax

```
Object.RelativeReferenceFrequency = [dValue]
[dValue] = Object.RelativeReferenceFrequency
```

For Visual C:

```
Object.SetRelativeReferenceFrequency(dValue)
dValue = Object.GetRelativeReferenceFrequency()
```

Wrapper dll syntax

```
SPECJITSetRelativeReferenceFrequency(hMeasurement,
                                     dValue)
SPECJITGetRelativeReferenceFrequency(hMeasurement,
                                     *dValue)
```

Description

Sets/returns the reference frequency for relative measurements. This takes effect if you set the *View* to "Relative". The measured power at this frequency is considered 0 dB, and all other power values are calibrated to that power. For details see the *Spectral Jitter Measurement User Guide*.

Parameters

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

dValue Specifies the reference frequency (data type: `ViReal64`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the reference frequency to 625 MHz:

```
m_SpectralJitterCTRL.RelativeReferenceFrequency = 625000000
```

Related functions and methods

"ViewType" on page 67

SamplePointOffset

ActiveX syntax

```
Object.SamplePointOffset = [dValue]
[dValue] = Object.SamplePointOffset
```

For Visual C:

```
Object.SetSamplePointOffset(dValue)
dValue = Object.GetSamplePointOffset()
```

Wrapper dll syntax

```
SPECJITSetSamplePointOffset(hMeasurement,
                            dValue)
SPECJITGetSamplePointOffset(hMeasurement,
                            *dValue)
```

Description

Sets/returns the offset of the sampling point for the measurement from the current sampling point in Unit Intervals (UI). For details see the *Spectral Jitter Measurement User Guide*.

Parameters

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

dValue Specifies the offset in UI (data type: `ViReal64`). For the wrapper dll GET function, this parameter is a pointer.

Example

To set the offset to –0.49 UI:

```
m_SpectralJitterCTRL.SamplePointOffset = -0.49
```

Scale

Syntax

```
Object.Scale = [eScale]
```

For Visual C:

```
Object.SetScale(eScale)
eScale = Object.GetScale()
```

NOTE This property is not available for the wrapper dll.

Description Sets/gets the graph's horizontal frequency scale.

Parameter **eScale** The following constants (data type: SCALE) are defined:

Constant	Description
Logarithmic	The graph is displayed in logarithmic mode.
Linear	The graph is displayed in linear mode.

Example To set the graph's scale to logarithmic:

```
m_SpectralJitterCTRL.Scale = Logarithmic
```

Related functions and methods *"DisplayPowerUnits" on page 36*

SetFreqAxisRange

Syntax `Object.SetFreqAxisRange(dMin, dMax)`

NOTE This property is not available for the wrapper dll.

Description Sets the range for the frequency zoom function. The zoom function can be enabled/disabled with `FreqAxisRangeType`.

Parameters **dMin** Specifies the start frequency (data type: `Double`).

dMax Specifies the end frequency (data type: `Double`).

Example To define a range for the frequency zoom from 1 MHz to 2.5 MHz:

```
m_SpectralJitterCTRL.SetFreqAxisRange = 1e6, 2.5e6
```

Related functions and methods *"FreqAxisRangeType" on page 39*

ShowGrid

ActiveX syntax

```
Object.ShowGrid = [boolean]
```

For Visual C:

```
Object.SetShowGrid(boolean)
boolean = Object.GetShowGrid()
```

NOTE This property is not available for the wrapper dll.

Description Switches the display of the graphical grid on or off.

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

Constant	Description
`True`	Turn on the display of the grid.
`False`	Turn off the display of the grid.

Example To display the grid in the graphical view:

```
m_SpectralJitterCTRL.ShowGrid = True
```

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_SpectralJitterCTRL.ShowMarkers = True
```

ShowProperties

ActiveX syntax `Object.ShowProperties()`

NOTE This method is not available for the wrapper dll.

Description Displays the *Properties* dialog for the control.

Parameter None

TermInvolved

ActiveX syntax

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

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

Wrapper dll syntax

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

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

Description

Sets/returns whether a terminal 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.

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 terminal 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 port 1, terminal 1 not involved in the measurement:

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

To get whether terminal 1 of port 1 is involved in the measurement:

```
Dim bIsInvolved as Boolean
bIsInvolved = m_SpectralJitterCTRL.GetTermInvolved(1, 1)
```

Related functions and methods

"PortInvolved" on page 56

TopFrequencyCount

ActiveX syntax

```
Object.TopFrequencyCount = [nValue]
[nValue] = Object.TopFrequencyCount
```

For Visual C:

```
Object.SetTopFrequencyCount(nValue)
nValue = Object.GetTopFrequencyCount()
```

Wrapper dll syntax

```
SPECJITSetTopFrequencyCount(hMeasurement,
                            nValue)
SPECJITGetTopFrequencyCount(hMeasurement,
                            *nValue)
```

Description Sets/returns the number of top frequency/power pairs to be displayed.

Parameters

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

nValue Specifies the number of top frequency/power pairs: 1 to 16 (data type: `ViInt16`). For the wrapper dll GET function, this parameter is a pointer.

Example To display six top frequency/power pairs:

```
m_SpectralJitterCTRL.TopFrequencyCount = 6
```

Related functions and methods *"RelativeReferenceFrequency" on page 58*

ViewType

ActiveX syntax

```
Object.ViewType = [eViewType]

[eViewType] = Object.ViewType
```

For Visual C:

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

Wrapper dll syntax

```
SPECJITSetViewType(hMeasurement,
                   eViewType)
SPECJITGetViewType(hMeasurement,
                   *eViewType)
```

Description

Sets/returns the view type. The view type defines whether power values are calculated as absolute, true relative, or relative-to-reference-frequency values. True relative values are normalized to the total spectral power. Relative values are normalized to the spectral power measured at a reference frequency.

Parameters

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

eViewType The following constants (data type: `ViewTypeEnums`) are defined:

Constant	Description
`AbsPowerVsFreq`	Displays absolute power ratios.
`TrueRelPowerVsFreq`	Displays power values normalized to the total spectral power.
`RelRefPowerVsFreq`	Displays power values normalized to the spectral power measured at the reference frequency.

Example

To display the true relative power values:

```
m_SpectralJitterCTRL.ViewType = TrueRelPowerVsFreq
```

Related functions and methods

"RelativeReferenceFrequency" on page 58

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 69*
To start a measurement run.	*"Run" on page 71*
To get the status of a measurement.	*"MeasState" on page 70*
To ensure a synchronous run on several systems.	*"SynchronousRun" on page 73*
To stop a measurement run.	*"Stop" on page 72*

Download

ActiveX syntax `Object.Download()`

Wrapper dll syntax `SPECJITMeasureDownload(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 71*

MeasState

ActiveX syntax

```
sState = Object.MeasState
```

For Visual C:

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

Wrapper dll syntax

```
SPECJITMeasureState(hMeasurement,
                    bufferSize,
                    sState)
```

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.

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`).

Example

To check the state of the measurement:

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

Run

ActiveX syntax `Object.Run()`

Wrapper dll syntax `SPECJITMeasureRun(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`).

Related functions and methods *"SynchronousRun" on page 73*

Stop

ActiveX syntax `Object.Stop()`

Wrapper dll syntax `SPECJITMeasureStop(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 71*

SynchronousRun

ActiveX syntax

```
Object.SynchronousRun()
```

NOTE This method is not available for the wrapper dll.

The wrapper dll function is called Run.

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.

Related functions and methods *"Run" on page 71*

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 75*
To indicate if the measurement run is complete.	*"OnMeasurementComplete" on page 76*
To indicate that a transition in the measurement status occured.	*"OnMeasurementState" on page 77*
To register a callback function for certain events.	*"SetMeasEventsCallback" on page 79*

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 MeasureSpectralJitter1_OnMeasDataAvailable(lItemCount as
Long)
   Dim msg as string
   msg = "Number of items:"+ lItemCount
   Msgbox msg
End Sub
```

OnMeasurementComplete

ActiveX syntax `Private Sub Object_OnMeasurementComplete(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 MeasureSpectralJitter1_OnMeasurementComplete(
                                                ByVal lStatus As Long)
   MsgBox "Measurement is complete"
End Sub
```

OnMeasurementState

ActiveX syntax `Private Sub Object_OnMeasurementState(eMeasState As Long)`

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: `MeasureStateEnums`) are:

Constant	Value	Description
`StateProg`	1	Measurement data is transferred.
`StateRunning`	2	The measurement is running.
`StateComplete`	3	The measurement run has been completed.
`StateAbort`	4	The measurement run has been aborted.
`StateError`	5	An error occurred.
`StateSync`	6	The system is synchronizing.
`StateHalt`	7	The system is in halted state, waiting for external start.
`StateRunPulse`	8	Heartbeat state.

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 a message box when the measurement is complete:

```
Private Sub MeasureSpectralJitter1_OnMeasurementState
                              (ByVal lStatus as Long)
' This event is called when the measurement state changes
Select Case lStatus

   Case 1
   ' Data transferred

   Case 2
   ' Measurement running

   Case 3
   ' Measurement complete
      MsgBox "The measurement is complete."

   Case 4
   ' Measurement aborted

   Case 5
   ' Error occurred
```

```
   Case 6
   ' System synchronizing

   Case 7
   ' System waiting for start

   Case 8
   ' Heartbeat state

End Select
End Sub
```

SetMeasEventsCallback

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

Wrapper dll syntax

```
SPECJITSetMeasEventsCallback(hMeasurement,
                             lpMeasEventsFunc,
                             lParamUserCBData)
```

Description To register a callback function that 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.

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 81*
To specify if errors are to be reported when running a measurement.	*"SilentMode" on page 82*

GetLastMeasError

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetLastMeasError(hMeasurement,
                        *lError,
                        sError
                        bufferSize)
```

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`).

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

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

Example

```
Dim sError as String
sError = m_SpectralJitterCTRL.GetLastMeasError()
```

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_SpectralJitterCTRL.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 get the number of captured and compared data points for a terminal.	*"DataAvailable" on page 84*
To get the number of power values for a terminal.	*"GetPowerDataPointCount" on page 89*
To get the power values for a terminal.	*"GetPowerDataArray" on page 87*
To get the frequency and power values for one data point.	*"GetPowerDataPoint" on page 88*
To get the measured values of a terminal individually.	*"GetTermCalculatedValue" on page 90*
To get the measured values of a top frequency/power pair.	*"GetTopFreqPowerValue" on page 92*
To get the frequency resolution of the measurement.	*"GetMeasFreqPrecision" on page 85*
To get the power precision of the measurement.	*"GetMeasPowerPrecision" on page 86*

DataAvailable

ActiveX syntax

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

Wrapper dll syntax

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

Description

Returns the number of data points that have been captured and compared 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. 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_SpectralJitterCTRL.DataAvailable(1, 1, nItems)
```

GetMeasFreqPrecision

ActiveX syntax

```
dValue = Object.GetMeasFreqPrecision()
```

Wrapper dll syntax

```
SPECJITGetMeasFreqPrecision(hMeasurement,
                            *dValue)
```

Description

Returns the frequency resolution of the measurement. This is calculated as the analyzer data rate divided by the acquisition depth.

Output parameter

dValue The returned value (data type: `ViReal64`) specifies the frequency distance between adjacent power values.

Input parameters

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

Example

To get the frequency precision of the measurement:

```
Dim dValue as Double
dValue = m_SpectralJitterCTRL.GetMeasFreqPrecision()
```

GetMeasPowerPrecision

ActiveX syntax

```
dValue = Object.GetMeasPowerPrecision(eUnit)
```

Wrapper dll syntax

```
SPECJITGetMeasPowerPrecision(hMeasurement,
                             eUnit,
                             *dValue)
```

Description

Returns the power resolution of the measurement.

Output parameter

dValue The returned value (data type: `ViReal64`) specifies the power precision.

Input parameters

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

eUnit The power unit of dValue (data type: `PowerUnitsEnums`). Accepted values are `PowerLinear` and `PowerDB`.

Example

To get the power precision of the measurement:

```
Dim dValue as Double
dValue = m_SpectralJitterCTRL.GetMeasPowerPrecision(PowerDB)
```

GetPowerDataArray

ActiveX syntax

```
Object.GetPowerDataArray(nPortId,
                         nTerminalID,
                         lStartItem,
                         lItemCount,
                         *lItemsRet,
                         PowerData)
```

Wrapper dll syntax

```
SPECJITGetPowerDataArray(hMeasurement,
                         nPortId,
                         nTerminalID,
                         lStartItem,
                         lItemCount,
                         *lItemsRet,
                         PowerData)
```

Description

Returns for a specified port and terminal a selectable number of consecutive power values.

Output parameter

lItemsRet The number of returned power values (data type: `ViInt32`).

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.

lStartItem The start index (data type: `ViInt32`).

lItemCount The number of requested power values (data type: `ViInt32`).

Power Data The pointer to the array that receives the power values (data type: `ViReal64`).

Related functions and methods

"GetPowerDataPointCount" on page 89
"GetPowerDataPoint" on page 88

GetPowerDataPoint

ActiveX syntax

```
Object.GetPowerDataPoint(nPortId,
                         nTerminalID,
                           lDataIndex,
                           *dFrequency,
                           *dPower)
```

Wrapper dll syntax

```
SPECJITGetPowerDataPoint(hMeasurement,
                         nPortId,
                         nTerminalID,
                         lDataIndex,
                         *dFrequency,
                         *dPower)
```

Description

Returns for a specified port and terminal the frequency and power of a single data point.

Output parameters

dFrequency The returned value (data type: `ViReal64`) specifies the frequency of the data point.

dPower The returned value (data type: `ViReal64`) specifies the power of the data point.

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.

lDataIndex The index number of the data point (data type: `ViInt32`).

Example

To get the frequency and power values of data point `DPnum` for terminal 1 of port 1:

```
Dim DPnum as Long
Dim dFreq, dPow as Double
m_SpectralJitterCTRL.GetPowerDataPoint(1, 1, DPnum, dFreq, dPow)
```

Related functions and methods

"GetPowerDataPointCount" on page 89
"GetPowerDataArray" on page 87

GetPowerDataPointCount

ActiveX syntax

```
Object.GetPowerDataPointCount(nPortId,
                              nTerminalID,
                              *lNumPoints)
```

Wrapper dll syntax

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

Description

Returns the number of power values that are available for a specified port and terminal.

Output parameter

lNumPoints The returned value (data type: `long`) specifies the number of power values that are available in the measurement. 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 power values available for terminal 1 of port 1:

```
Dim nValues as Long
m_SpectralJitterCTRL.GetPowerDataPointCount(1, 1, nValues)
```

Related functions and methods

"GetPowerDataPoint" on page 88
"GetPowerDataArray" on page 87

GetTermCalculatedValue

ActiveX syntax

```
dValue = Object.GetTermCalculatedValue(nPortID,
                                       nTermID,
                                       eAnalysisTerm,
                                       *bValid)
```

Wrapper dll syntax

```
SPECJITGetTermCalculatedValue(hMeasurement,
                              nPortID,
                              nTermID,
                              eAnalysisTerm,
                              *bValid,
                              *dValue)
```

Description

Returns the value of the requested analysis term measured at the designated terminal.

Output parameters

dValue Returned measured or calculated value (data type: `Double`). For the wrapper dll GET function, this parameter is a pointer.

bValid Returns whether the value returned is valid for the measurement. There are several cases where the value may not be valid. The following constants (data type: `Boolean`) are defined:

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

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.

eAnalysisTerm The following constants (data type: `AnalysisTermEnums`) are defined:

Constant	Corresponds to:
`AT_FreqRange1`	Power of frequency range #1
`AT_FreqRange2`	Power of frequency range #2
`AT_FreqRange3`	Power of frequency range #3

Constant	Corresponds to:
AT_FreqRange4	Power of frequency range #4
AT_FreqRange5	Power of frequency range #5
AT_FreqRange6	Power of frequency range #6
AT_FreqRange7	Power of frequency range #7
AT_FreqRange8	Power of frequency range #8
AT_BitErrorRate	Bit error rate
AT_TotalPower	Total power
AT_NoisePower	Noise power

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

Example To get the calculated power of frequency range #2 for terminal 3 of port 1:

```
Dim dValue as Double
Dim bValid as boolean
dValue = m_SpectralJitterCTRL.GetTermCalculatedValue(1,
                                                     3,
                                                     AT_FreqRange2,
                                                     bValid)
```

GetTopFreqPowerValue

ActiveX syntax

```
boolean = Object.GetTopFreqPowerValue(nPortID,
                                      nTermID,
                                      nDataIndex,
                                      *dFrequency,
                                      *dPower)
```

Wrapper dll syntax

```
SPECJITGetTopFreqPowerValue(hMeasurement,
                                    nPortID,
                                    nTermID,
                                    nDataIndex,
                                    *dFrequency)
                                    *dPower)
```

Description

Returns the frequency and power for the specified pair of top frequency/power values.

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: `ViInt16`). This is an index starting at 1.

nTermID Terminal ID number within the port (data type: `ViInt16`).

nDataIndex The number of the pair of frequency/power values:1 to 16 (data type: `ViInt32`).

Output parameters

dFrequency The center frequency of the frequency bin (data type: `ViReal64`).

dPower The power of the frequency bin (data type: `ViReal64`).

Related functions and methods

"TopFrequencyCount" on page 66
"GetTerminalId" on page 48

Pass / Fail Functions

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 if a measurement, port or terminal has passed or failed:

Purpose	Refer to...
To check if a parameter has passed for the complete measurement.	*"GetMeasPassValue" on page 94*
To check if a parameter has passed for a given port.	*"GetPortPassValue" on page 96*
To check if a parameter has passed for a given terminal.	*"GetTermPassValue" on page 98*

GetMeasPassValue

ActiveX syntax

```
bIsPass = Object.GetMeasPassValue(eAnalysisTerm)
```

Wrapper dll syntax

```
SPECJITGetMeasPassValue(hMeasurement,
                        eAnalysisTerm,
                        *bIsPass)
```

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

bIsPass 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`).

eAnalysisTerm The following constants (data type: `AnalysisTermEnums`) are defined:

Constant	Corresponds to:
`AT_FreqRange1`	Power of frequency range #1
`AT_FreqRange2`	Power of frequency range #2
`AT_FreqRange3`	Power of frequency range #3
`AT_FreqRange4`	Power of frequency range #4
`AT_FreqRange5`	Power of frequency range #5
`AT_FreqRange6`	Power of frequency range #6
`AT_FreqRange7`	Power of frequency range #7
`AT_FreqRange8`	Power of frequency range #8
`AT_BitErrorRate`	Bit error rate
`AT_TotalPower`	Total power
`AT_NoisePower`	Noise power
`AT_AllAnalysis`	Values for all conditions

Remarks The criteria will always pass if the pass/fail criterion has been turned off using the `PassFailUse`.

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

Example To see if the measurement has passed the BER limits:

```
Dim bBERPassed as Boolean
bBERPassed = m_SpectralJitterCTRL.GetMeasPassValue(AT_BitErrorRate)
```

Related functions and methods

"GetPortPassValue" on page 96
"GetTermPassValue" on page 98
"PassFailUse" on page 51

GetPortPassValue

ActiveX syntax

```
bPass = Object.GetPortPassValue(nPortID,
                                eAnalysisTerm,
                                *bValid)
```

Wrapper dll syntax

```
SPECJITGetPortPassValue(hMeasurement,
                        nPortID,
                        eAnalysisTerm,
                        *bValid,
                        *bPass)
```

Description

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

Output parameters

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 function, this parameter is a pointer.

bValid 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.

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.

eAnalysisTerm The following constants (data type: `AnalysisTermEnums`) are defined:

Constant	Corresponds to:
`AT_FreqRange1`	Power of frequency range #1
`AT_FreqRange2`	Power of frequency range #2
`AT_FreqRange3`	Power of frequency range #3
`AT_FreqRange4`	Power of frequency range #4
`AT_FreqRange5`	Power of frequency range #5
`AT_FreqRange6`	Power of frequency range #6
`AT_FreqRange7`	Power of frequency range #7
`AT_FreqRange8`	Power of frequency range #8
`AT_BitErrorRate`	Bit error rate
`AT_TotalPower`	Total power
`AT_NoisePower`	Noise power
`AT_AllAnalysis`	Values for all conditions

Remarks The criteria will always pass if the pass/fail criterion has been turned off using the PassFailUse.

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

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

```
Dim bPass, bValid as Boolean
bPass = m_SpectralJitterCTRL.GetPortPassValue(1, AT_TotalPower,
bValid)
```

Related functions and methods

"GetMeasPassValue" on page 94
"GetTermPassValue" on page 98
"PassFailUse" on page 51

GetTermPassValue

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITGetTermPassValue(hMeasurement,
                        nPortID,
                        nTermID,
                        eAnalysisTerm,
                        bValid,
                        bPass)
```

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

NOTE Before `GetTermPassValue` can return a valid result, `PassFailUse` must have first been called for all parameters of interest.

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

Output parameters

bPass Returns whether 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 ActiveX function, this parameter is a pointer.

bValid 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.

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.

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

Constant	Corresponds to:
`AT_FreqRange1`	Power of frequency range #1
`AT_FreqRange2`	Power of frequency range #2
`AT_FreqRange3`	Power of frequency range #3
`AT_FreqRange4`	Power of frequency range #4
`AT_FreqRange5`	Power of frequency range #5
`AT_FreqRange6`	Power of frequency range #6
`AT_FreqRange7`	Power of frequency range #7
`AT_FreqRange8`	Power of frequency range #8
`AT_BitErrorRate`	Bit error rate
`AT_TotalPower`	Total power
`AT_NoisePower`	Noise power
`AT_AllAnalysis`	Values for all conditions

Remarks

The parameter will always pass if its pass/fail criterion have been turned off using PassFailUse.

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

Example

To check whether terminal 3 of port 1 has passed the test:

```
Dim bTPass, bValid as Boolean
bTPass=m_SpectralJitterCTRL.GetTermPassValue(1,
                                             3,
                                             AT_AllAnalysis,
                                             &bBool);
```

Related functions and methods

"GetMeasPassValue" on page 94
"GetTermPassValue" on page 98
"PassFailUse" on page 51

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 101*
To cut data from the measurement and copy the data to the clipboard.	*"CutToClipboard" on page 102*
To delete data from the numerical view.	*"EditDelete" on page 103*
To determine whether a copy function call is possible.	*"IsCopyAvailable" on page 104*
To determine whether a cut function call is possible.	*"IsCutAvailable" on page 105*
To determine whether a delete function call is possible.	*"IsEditDeleteAvailable" on page 106*
To determine whether a paste function call is possible.	*"IsPasteAvailable" on page 107*
To insert data previously copied to the clipboard.	*"PasteFromClipboard" on page 108*

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* gets the data displayed in the grid in HTML format.

In addition, data is stored in the clipboard in a format specific to the measurement software 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_SpectralJitterCTRL.CopyToClipBoard(True)
```

Related functions and methods

"CutToClipboard" on page 102
"PasteFromClipboard" on page 108
"IsCopyAvailable" on page 104

CutToClipboard

ActiveX syntax `Object.CutToClipboard(boolean)`

NOTE This method is not available for the wrapper dll.

Description Cuts data from the measurement and copies the data into 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 as HTML.

In addition, data is stored in the clipboard in a format specific to the measurement software 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_SpectralJitterCTRL.CutToClipBoard(True)
```

Related functions and methods

"CopyToClipboard" on page 101
"IsCutAvailable" on page 105

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_SpectralJitterCTRL.EditDelete(True)
```

Related functions and methods *"IsEditDeleteAvailable" on page 106*

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_SpectralJitterCTRL.IsCopyAvailable()
```

Related functions and methods *"IsCutAvailable" on page 105*

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_SpectralJitterCTRL.IsCutAvailable()
```

Related functions and methods *"IsCopyAvailable" on page 104*

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_SpectralJitterCTRL.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_SpectralJitterCTRL.IsPasteAvailable()
```

Related functions and methods

"IsCutAvailable" on page 105
"IsCopyAvailable" on page 104

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 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 101*
"CutToClipboard" on page 102

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 111*
To save a measurement.	*"SaveMeasurement" on page 110*

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 112*
To set the delimiter for the data export.	*"ExportDelimiter" on page 113*
To set the file name the data will be exported to.	*"ExportFileName" on page 114*
To set the date format for export.	*"ExportLocale" on page 115*
To specify if the results will be exported to file or clipboard.	*"ExportToClipboard" on page 116*

SaveMeasurement

ActiveX syntax

```
Object.SaveMeasurement(sFileName)
```

Wrapper dll syntax

```
SPECJITSaveMeasurement(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 an extension of `.mcp`.

Example

To save the measurement `DUTSpectralJitter.mcp`:

```
m_SpectralJitterCTRL.SaveMeasurement
("C:\\Temp\\DUTSpectralJitter.mcp")
```

LoadMeasurement

ActiveX syntax

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

Wrapper dll syntax

```
SPECJITLoadMeasurement(hMeasurement,
                       sFileName)
```

Description

Loads the measurement stored in the designated file into the control.

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 `DUTSpectralJitter.mcp`:

```
m_SpectralJitterCTRL.LoadMeasurement
("C:\\Temp\\DUTSpectralJitter.mcp")
```

ExecuteExport

ActiveX syntax `Object.ExecuteExport()`

Wrapper dll syntax `SPECJITExecuteExport(hMeasurement)`

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

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

Example
```
m_SpectralJitterCTRL.ExecuteExport
```

Related functions and methods
"ExportLocale" on page 115
"ExportDelimiter" on page 113
"ExportFileName" on page 114
"ExportToClipboard" on page 116

ExportDelimiter

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITGetExportDelimiter(hMeasurement,
                          bufferSize,
                          *sDelimiter)
SPECJITSetExportDelimiter(hMeasurement,
                          sDelimiter)
```

Description

Sets/returns the delimiter for export.

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`).

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_SpectralJitterCTRL.ExportDelimiter = "^"
```

Related functions and methods

"ExecuteExport" on page 112
"ExportFileName" on page 114
"ExportLocale" on page 115
"ExportToClipboard" on page 116

ExportFileName

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITGetExportFileName(hMeasurement,
                         bufferSize,
                         *sFileName)
SPECJITSetExportFileName(hMeasurement,
                         sFileName)
```

Description

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

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`).

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

Related functions and methods

"ExecuteExport" on page 112
"ExportDelimiter" on page 113
"ExportLocale" on page 115
"ExportToClipboard" on page 116

ExportLocale

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITGetExportLocale(hMeasurement,
                       bufferSize,
                       *sLocale)
SPECJITSetExportLocale(hMeasurement,
                       sLocale)
```

Description

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

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`).

sLocale Name of the language that the export format will default to. 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 112
"ExportDelimiter" on page 113
"ExportFileName" on page 114
"ExportToClipboard" on page 116

ExportToClipboard

ActiveX syntax

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

For Visual C:

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

Wrapper dll syntax

```
SPECJITGetExportToClipboard(hMeasurement,
                            *boolean)
SPECJITSetExportToClipboard(hMeasurement,
                            boolean)
```

Description

Sets whether the export will go to the clipboard or not.

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`	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_SpectralJitterCTRL.ExportToClipboard = True
```

Related functions and methods

"ExecuteExport" on page 112
"ExportDelimiter" on page 113
"ExportFileName" on page 114

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 118*
To set the color of the noise threshold marker.	*“PowerMarkerColor” on page 123*
To set the color of frequency ranges.	*“FrequencyRangesColor” on page 120*
To set the text color of the graphical display.	*“ForeColor” on page 119*
To set/return the name and the path of the help file.	*“MeasHelpPath” on page 121*
To set the help id for the measurement window.	*“MeasureWinHelp” on page 122*

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 color of the background 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_SpectralJitterCTRL.BackColor = 16711680
```

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 color of the foreground (text) on the graph.

Input 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_SpectralJitterCTRL.ForeColor = 0
```

Related functions and methods *"BackColor" on page 118*

FrequencyRangesColor

ActiveX syntax

```
Object.FrequencyRangesColor = [nColor]
```

For Visual C:

```
Object.SetFrequencyRangesColor(nColor)
nColor = Object.GetFrequencyRangesColor()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns the color of enabled frequency ranges.

Parameter **nColor** Sets the color frequency ranges (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 frequency ranges to "Green":

```
m_SpectralJitterCTRL.FrequencyRangesColor = 65280
```

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\tmSpectralJitterM.chm"`:

```
m_SpectralJitterCTRL.MeasHelpPath =
"C:\\Temp\\tmSpectralJitterM.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).

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_SpectralJitterCTRL.MeasHelpPath
helpID = m_SpectralJitterCTRL.DefaultHelpID
m_SpectralJitterCTRL.MeasureWinHelp(frmMain.hWnd, helpstr, 1,
helpID)
```

Related functions and methods

"MeasHelpPath" on page 121

PowerMarkerColor

ActiveX syntax

```
Object.PowerMarkerColor = [nColor]
```

For Visual C:

```
Object.SetPowerMarkerColor(nColor)
nColor = Object.GetPowerMarkerColor()
```

NOTE This property is not available for the wrapper dll.

Description Sets/returns the color of the noise threshold marker.

Parameter **nColor** Sets the noise power 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 noise threshold marker to "Green":

```
m_SpectralJitterCTRL.PowerMarkerColor = 65280
```

Index

Printed in Germany May 2003

5988-9645EN

Agilent Technologies