Keysight N437x Series Lightwave Component Analyzer

KEYSIGHT
TECHNOLOGIES

Programmer's Guide

Notices

Keysight Technologies, Deutschland GmbH
Herrenberger Straße 130
71034 Böblingen, Germany.

Manual Part Number

437xB-90A01

Edition

Edition 2.0, February 2015

Subject Matter

The material in this document is subject to change without notice.

Keysight Technologies *makes no warranty of any kind with regard to this printed material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.*

Keysight Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Printing History

New editions are complete revisions of the guide reflecting alterations in the functionality of the instrument. Updates are occasionally made to the guide between editions. The date on the title page changes when an updated guide is published. To find out the current revision of the guide, or to purchase an updated guide, contact your Keysight Technologies representative.

Warranty

This Keysight Technologies instrument product is warranted against defects in material and workmanship for a period of one year from date of shipment. During the warranty period, Keysight will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Keysight. Buyer shall prepay shipping charges to Keysight and Keysight shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Keysight from another country.

Keysight warrants that its software and firmware designated by Keysight for use with an instrument will execute its programming instructions when properly installed on that instrument. Keysight does not warrant that the operation of the instrument, software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

No other warranty is expressed or implied. Keysight Technologies specifically disclaims the implied warranties of Merchantability and Fitness for a Particular Purpose.

Exclusive Remedies

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

Assistance

Product maintenance agreements and other customer assistance agreements are available for Keysight Technologies products. For any assistance contact your nearest Keysight Technologies Sales and Service Office.

Certification

Keysight Technologies certifies that this product met its published specifications at the time of shipment from the factory.

Keysight Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, NIST, to the extent allowed by the Institutes’s calibration facility, and to the calibration facilities of other International Standards Organization members.

ISO 9001 Certification

Produced to ISO 9001 international quality system standard as part of our objective of continually increasing customer satisfaction through improved process control.

Remote Operation

1

Remote Operation

Overview

This programming guide supports LCA models beginning with the B generation. These now include: N4373D, N7373C, N4373B, N4374B, N4375D, N4375B, N4376D and N4376B.

This chapter will help you control an LCA from your own computer. The chapter covers how to write your own applications. The next chapter explains examples based on Keysight VEE and VBA/Excel in more detail. Note that applications for remote control can also be run on the LCA itself, which is useful for automated measurement procedures.

The LCA is a remoting enabled, Microsoft .NET instrument that can be controlled across any LAN that can relay an http web page. The provided remote control client has an Active X interface and a .NET interface, so you can program the LCA from many established programming environments such as Visual Basic 6.0 and VBA, as well as from .NET enabled programming environments such as C#.

Beginning with the LCA software version 3.00.03 for Windows XP systems or 3.01.00 with Window 7, an SCPI interface is also available, which may be more comfortable for other environments like Labview. The SCPI interface can be used over either a LAN or USB port.

The LCA uses .NET remoting as the foundation for its external communications. Remoting is the process of programs or distributed components interacting across different processes or machines.

In .NET remoting, the server program publishes an object on a network channel and the client program subscribes to that channel when loading or connecting to that object. In the case of the LCA, a RemoteObject object is published to an http channel and the subscribing client program is the LCA RemoteClient. A Remoting server is embedded in the LCA Server application. The LCA RemoteClient is a layer of abstraction, which provides an easy to use interface with methods to control the LCA. The LCA Remote Client layer consists of 3 files, named "RemoteClient.dll", "RemoteObjects.dll" and "RemoteClient.tlb". These files are installed as part of the the LCA Remote Client installation package, together with a number of programming examples.

Since the LCA interface does not provide any methods to set network analyzer related parameters or to retrieve measurement data from the network analyzer, most applications also need to program the network analyzer. The network analyzer's native functions can be controlled either using SCPI or COM. We recommend using the COM interface. This is reflected in the programming examples.

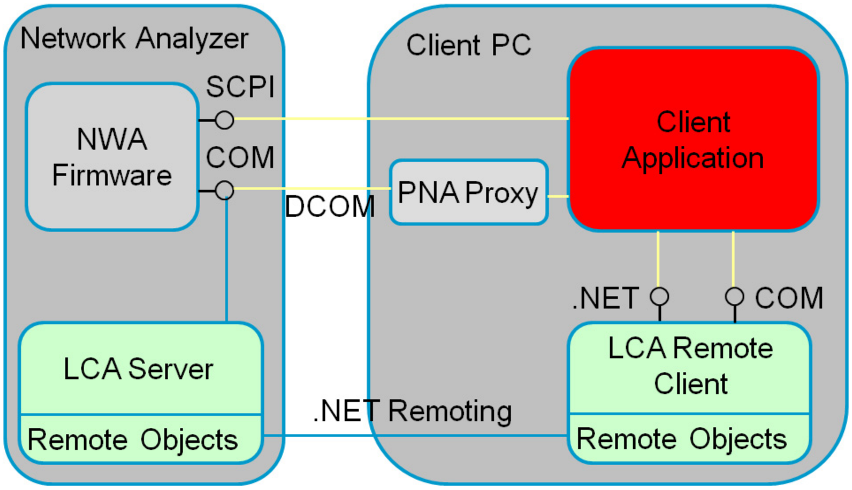


Figure 1 LCA Remoting Architecture

While this chapter assumes you are familiar with your programming environment, it does not assume familiarity with controlling remote objects from within that environment. Examples are provided for VB.NET, C#, VB 6.0, VBA and Keysight VEE, which can be extrapolated to most environments for controlling the LCA. After installing the LCA Remote Client on your computer, you can find these examples in the folder:

C:\Program Files (x86)\Agilent\Agilent LCA Remote Client\Examples

The location on your computer depends on the folder in which you installed the LCA Remote Client.

The Excel-VBA example pulls data directly from the LCA into Excel. This is very useful if you are setting up measurements manually, but want to analyze the results on your own computer.

Transferring code from the 8703A/B to the Keysight N437x Series Lightwave Component Analyzer

Tools are available to migrate code from the 8720 network analyzer to the new PNA network analyzer platform at

www.keysight.com/find/nadisco

The 8703A/B Lightwave Component Analyzers are based on the 8720 network analyzers, so you can use these code conversion tools to migrate existing code to the N437x Series LCA based on the PNA platform.

Most of the code in a typical application for the 8703 LCA controls the functionality of the network analyzer. This part of the application can be migrated with these tools.
The code related to LCA specific functionality has to be migrated by hand.

LCA System Configuration

How to configure the LCA for networking

Remote programming of the LCA with the .NET interface is only possible if the LCA is connected to a local area network (LAN) via the built-in LAN connector. When the LCA is connected to a network, it is also possible to connect it to network printers and remote servers, with access to shared folders and files.

NOTE Using the SCPI interface, the LCA can also be controlled from a USB port.

How to connect the LCA to your network

The LCA comes configured for DHCP networking, and has a default machine name. In many cases, connecting the LCA to your LAN is simply a case of registering the machine name with your IT department.

NOTE Do not connect the LCA to a network that is configured to automatically install software on network devices. Installing or overwriting files on the LCA computer system may impact the operation of the instrument. Please contact your network administrator or IT department to find out if you have this type of network.

NOTE The LCA LAN connector supports 10 Base-T and 100 Base-T Ethernet networks using TCP/IP and other Microsoft supported networking protocols. The LCA uses Microsoft® Windows 7 or XP.

How to change network settings

You can change the LCA network settings as needed so that it connects properly to your specific network.

NOTE Because your network settings are unique to your IT infrastructure, Keysight Technologies will not be able to assist you with connecting your instrument to your network. Please contact your network administrator or IT department for assistance. For more information, refer to the MS Windows resource kit (available from Microsoft) that is appropriate for your computer system. You can also refer to the online Help for Windows (Start > Help).

NOTE

By default, as the instrument starts up, you are logged on as an administrator with the logon name PNA-Admin. The login password, which is usually not needed, is "pna" for systems with Windows 7 or "agilent" for systems using Windows XP.
Keysight only recommends using the LCA application while you are logged on as an administrator.

You can change network settings by using the standard Microsoft® Windows functions.

To view or change the computer machine name

1 On the Task bar, click Start, point to Settings, and then click Control Panel.

2 Double-click the System icon and click on the Computer Name tab. From here you can view or change the machine name.

3 When you have finished making changes, restart the instrument.

To configure TCP/IP to use DNS or WINS

NOTE

If using a protocol other than TCP/IP, please contact your IT department for assistance.

NOTE

Editing your instrument’s protocols and file access permissions can result in unwanted behaviors that are difficult to reverse. Ensure that your changes are valid!

NOTE

Please consult with your network administrator concerning advanced TCP/IP and multi-protocol configuration settings to support your network.

NOTE

Please contact your network administrator or IT department if you have any problems connecting the LCA to your network.

4 On the Task bar, click Start, point to Settings, and then click Network and Dial-up Connections.

5 Then click Local Area ConnectionProperties.

6 On the General tab (for a local area connection) or the Networking tab (all other connections), click Internet Protocol (TCP/IP), and then click Properties. From here, you can make all desired changes.

7 When you have finished making changes, restart the instrument.

NOTE

For more information, click Start > Help > Index, and search for “DNS” or “WINS” or “static” or “dynamic.”

To configure TCP/IP for static or dynamic addressing

- To get started, follow the same steps listed above.

Install the LCA Remote Client

The LCA Remote Client is described in “Overview” on page 6.

NOTE

This installation is not for the LCA itself. (Applications using the remote programming commands can be run on the LCA itself without installing the remote client package.)

1 If not already installed, install the .NET Framework Version 2.0 from Microsoft. Go to www.microsoft.com and search for ‘How to get the Microsoft .NET framework’. Be sure to get the framework and all the service packs. Make sure that you get the framework, not the SDK (software development kit.)

2 The LCA CD shipped with the LCA contains the Remote Client Installation Package to install the LCA specific DLLs and the programming examples. The most recent version of the LCA Remote Client Installation Package is available from the Keysight web site (www.keysight.com/find/lca).

- Insert the CD into the CD drive, use Windows Explorer to find LCA Remote Client Installer Folder, or
- Start the downloaded installer.

3 If you want to program the network analyzer via its COM interface you need to install the PNAProxy. The installation executable “PNAProxy.exe” can be found on the network analyzer in the folder:

```
C:\Program Files\Agilent\Network Analyzer\Automation
```

Install the PNA Proxy by running the installation program “PNAProxy.exe” on your client machine.

When asked to type in the host name or IP address of the remote network analyzer during installation, you do not need to type in anything.
You can specify the host name or IP address during program development or execution.

How to use the LCA Remote Client

Here you can see the basic steps required to write an LCA client application.

The code sequences presented here are in VB.NET syntax. For sequences in other languages like C#, VB 6.0, VBA or C++ refer to the different programming examples. You can find these examples in the “Examples” folder, in the “Keysight LCA Remote Client” installation folder.

Since most client applications will also control the network analyzer for setting measurement parameters like start- and stop-frequency and for reading out the measurement data, we also show the basic steps required to control the network analyzer using its COM interface over LAN (DCOM).
The network analyzer can also be programmed using its SCPI interface, but this is not covered here. For details about programming the network analyzer, please refer to the relevant network analyzer documentation.

Adding references to your project

In VB.NET, C#, VB 6.0 or VBA projects, you have to add references to the LCA Remote Client Library and to the PNAProxy type library (the network analyzer proxy, assuming you also want to program the network analyzer).

The LCA Remtote Client implements two different interface technologies.

- In environments which support .NET assemblies, we recommend using the LCA Remote Client .NET assembly directly.
- If your programming environment does not support .NET assemblies, use the LCA Remote Client over its COM interface.

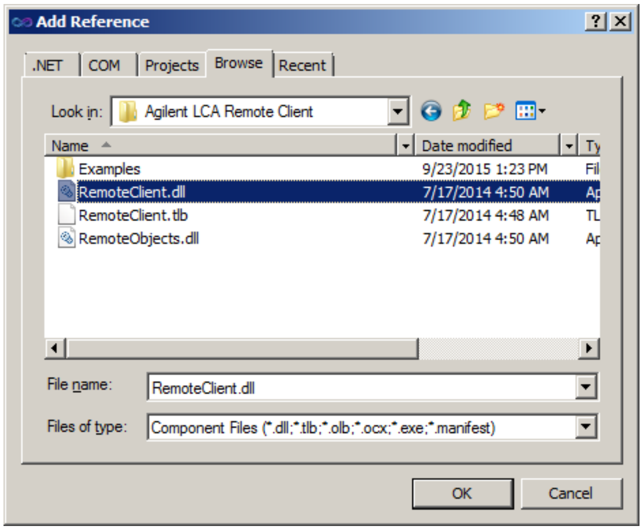
Here we show how this is done in Microsoft Visual Studio 2005 using the LCA Remote Client .NET assembly directly. When using the COM interface, the basic structure is the same.

For the differences, please check the VBA, VB 6.0 and C++ example projects, installed with the LCA Remote Client.

1 From the “Project” menu, select “Add Reference”.

2 Switch to the “Browse” tab.

3 Browse to your LCA Remote Client installation folder.



Look in: Agilent LCA Remote Client

Name	Date modified	Ty
Examples	9/23/2015 1:23 PM	Fil
RemoteClient.dll	7/17/2014 4:50 AM	Ap
RemoteClient.tlb	7/17/2014 4:48 AM	TL
RemoteObjects.dll	7/17/2014 4:50 AM	Ap

File name: RemoteClient.dll

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

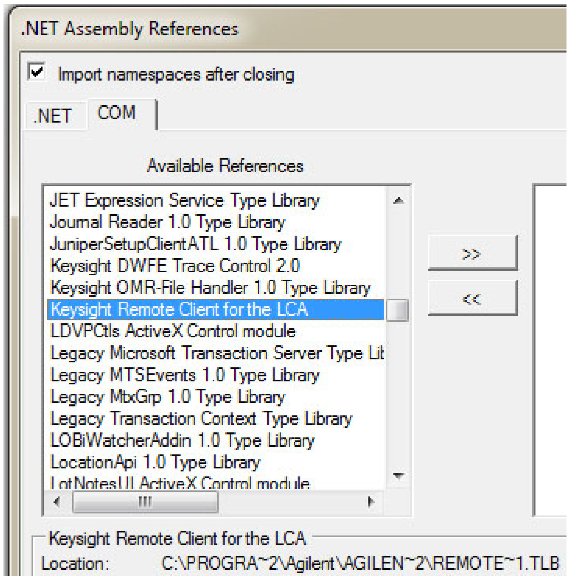
4 Select “RemoteClient.dll” and press OK.

5 If you also want to use the network analyzer COM interface, please refer to the network analyzer documentation, including: http://na.support.keysight.com/pna/programming/

Because environments like VB 6.0 or VBA cannot work directly with .NET assemblies, you have to use the COM interface of the LCA Remote Client in such cases.

1 In VB 6.0, from the “Project” menu, select “References”.
In VBA, open the “Tools” menu and select “References”.

In both programming environments you will see a dialog like the following:



JET Expression Service Type Library
Journal Reader 1.0 Type Library
JuniperSetupClientATL 1.0 Type Library
Keysight DWFE Trace Control 2.0
Keysight OMR-File Handler 1.0 Type Library
Keysight Remote Client for the LCA
LDVPCtls ActiveX Control module
Legacy Microsoft Transaction Server Type Lit
Legacy MTSEvents 1.0 Type Library
Legacy MtxGrp 1.0 Type Library
Legacy Transaction Context Type Library
LOBiWatcherAddin 1.0 Type Library
LocationApi 1.0 Type Library
LotNotesUI ActiveX Control module

Keysight Remote Client for the LCA
Location: C:\PROGRA~2\Agilent\AGILEN~2\REMOTE~1.TLB

2 Select“Keysight Remote Client for the LCA”.

Declare and create the required objects

The LCA Remote Client defines

- three interfaces ILCARemoteClient, ILCAMeasParams, ILCAProperties and
- three classes, LCARemoteClient, LCAMeasParams and LCAProperties.

Each of these classes implements the corresponding interface. To be able to use the LCA Remote Client, you have to create objects from these classes.

```
‘ Declare the objects
Private lcaClient As Agilent.LCA.RemoteClient.LCARemoteClient
Private lcaMeasParams As Agilent.LCA.RemoteClient.LCAMeasParams
Private lcaProperties As Agilent.LCA.RemoteClient.LCAProperties
....

‘Create the objects
lcaClient = New Agilent.LCA.RemoteClient.LCARemoteClient()
lcaMeasParams = New Agilent.LCA.RemoteClient.LCAMeasParams()
lcaProperties = New Agilent.LCA.RemoteClient.LCAProperties()
```

If you also want to use the network analyzer, you have to declare and create a network analyzer application object.
This is quite different to the LCA. When working with the LCA you are creating a local LCA Remote Client object. The connection to the remote LCA server is done with the “Connect” command on the LCA Remote Client interface.
When using the network analyzer over its COM interface, you are using DCOM and have to remotely activate the network analyzer interface. For examples on how this is done in different programming environments, see the programming examples installed with the LCA Remote Client.
Here we show how this is done in VB.NET:

```
‘ Declare the object

Private pnaClient As AgilentPNA835x.Application

 ...

Public Sub Open(ByVal serverName As String)

    ‘ the class-id of the AgilentPNA835x.Application class

    Dim clsID As System.Guid = New Guid(
                     "16D3C697-5F97- 11D2-BC1F-0060B0B52EA7")

    Dim srvtype As System.Type =
                                      System.Type.GetTypeFromCLSID(
                                              clsID, serverName, True)

    ‘ now we connect to the remote PNA

    pnaClient =
        CType(System.Activator.CreateInstance(srvtype),
                                      AgilentPNA835x.IApplication9)

End Sub
```

For further details on programming the network analyzer, please refer to the relevant network analyzer documentation.

Basic structure of an LCA client application

When programming the LCA you have to follow this basic structure:

1 **(optional)** Set a time-out value

```
lcaClient.SetTimeout(timout_ms)
```

2 Connect to the LCA server.

```
lcaClient.Connect(serverName)
```

now you could call commands which do not require an open session. In the case of the LCA client, this is the GetLCAProperties command.

```
lcaClient.GetLCAProperties(lcaProperties)
```

3 Open a session on the LCA, and check the return value of the Open() command. A return value False indicates that the Open() command has failed.

```
lcaClient.Open()
```

4 All commands that change the state of the LCA require an active session opened on the LCA. All these commands have to be enclosed by Open() and Close() commands. Commands which do not change the state of the LCA, like reading properties, only require a passive session on the LCA.

5 When finished with working on the LCA, close the session

```
lcaClient.Close()
```

6 Before leaving the application, make sure to call the Disconnect() command. This prevents unnecessary processing overhead on the LCA, needed to monitor and close inactive sessions.

```
lcaClient.Disconnect()
```

Synchronous vs. Asynchronous Method Calls

A traditional remote control application consists of a list of actions that you send to the instrument, expecting it to execute them in that order and to tell you when it is done. This makes programming easy - you can do your whole measurement in a single function or sub-routine.

In this approach you send the actions to the instrument in synchronous mode. This means that an action you send to the instrument blocks the program flow of the calling thread until it finishes. The advantage is that your program structure is very simple. The drawback is that you have to wait for the instrument to finish the action. For example this could lead to an unresponsive user interface.

This can be solved using multithreading . Run the measurement sequence in a new thread while the main thread handles other things like running the user interface.

A third possibility is to call potentially time consuming actions asynchronously. The LCA Remote Client lets you call some commands in asynchronous mode. This means that the call returns immediately, even before the action on the instrument has finished execution.

In such cases you need an additional method to determine, when an action finishes. The LCA Remote Client offers two different methods to accomplish this.

- The first is the property OperationComplete().
  This property value is True, when the last asynchronously called operation on the LCA has finished execution. Otherwise the property value is False.
- The other method is named WaitForOPC().
  This method blocks program execution on the calling thread until the operation on the instrument finishes.

Here are two short examples in VB.NET syntax, showing the usage of asynchronous calls:

Using the OperationComplete() Property in a loop:

```
oLCAClient.Init_OO(params, False)
Do
    ‘ let the application handle events
    Application.DoEvents()
    System.Threading.Thread.Sleep(200)
While oLCAClient.OperationComplete = False
```

Using the WaitForOPC() command:

```
oLCAClient.Init_OO(params, False)
DoMyActionsAfterCallingInit() ‘ doing some other stuff
‘ When we are done with our own stuff,
‘ we need to wait for Init_OO to finish
oLCAClient.WaitForOPC()
```

Troubleshooting

During application development you may encounter situations where the Open() call fails.
This happens when a session on the LCA is already open.
If there are no other applications using the LCA, the most likely reason is that an application finished without closing its session, for example when running an application in the debugger and you terminate it by stopping the debugger.

The LCA and the LCA Remote Client have a heartbeat mechanism to detect abandoned sessions. The LCA checks for 60 seconds of inactivity. If nothing happens in this time, the LCA assumes the session has been abandoned and it closes this session, so that other clients are able to open a session.

You may want to workaround this behavior during application development. There are two cases here.

- If your client application halts on a breakpoint, the heartbeat is suspended, so if your application is suspended for more than 60 seconds, the server closes the session. When you try to continue execution, you get an error telling you that no session is open.
  To keep sessions open, start the LCA server on the network analyzer with the command-line parameter “NOAUTOCLOSE”.
- If you are running into problems restarting your application because aborted sessions are still open, call CloseAll() before the Open() call.

NOTE

We recommend you only use these workarounds during development.

Only use CloseAll() in environments where you are sure no other client could have a session opened. CloseAll() will close sessions from all the LCA clients.

LCA Remote Programming

The LCA remote programming interface uses Microsoft.NET Remoting technology. It is controlled by manipulating the properties and methods exposed by the server object. The list of properties and methods in this section describe the interface that is available to a programmer wanting to program the LCA system in other applications.

LCA remote control DLLs

The LCA RemoteClient DLL provides a communication link with the LCA server. The DLLs are comprised of a set of properties, and methods that together provide a basic set of remote LCA capabilities. The two DLLs of interest are: RemoteClient.dll and RemoteObjects.dll. By default these two DLLs are installed to: C:\Program Files\Agilent\Agilent LCA Remote Control\.

Specific Commands

Interface structure

There are three classes to control the LCA: the LCAMeasParams, the LCAProperties and the LCARemoteClient.

- The class LCAMeasParams summarizes all possible parameters of your measurement.
- The class LCAProperties provides read-only properties, which give you some information about the network analyzer and the LCA.
- The class LCARemoteClient provides the methods to connect to the LCA, perform measurements and change hardware settings.

Enumeration

This is the list of enumeration names, with their possible values.

Enumeration	Description	Possible values
ELaserState	Enumerates the possible laser states, on or off.	NotSet LaserOff LaserOn
ELaserWvl	Enumerates the possible laser wavelengths.	NotSet Wvl_850nm Wvl_1310nm Wvl_1550nm
EMeasMode	Specify if you are doing single ended or differential measurements. Note: differential measurements require a 4-port network analyzer.	NotSet SingleEnded Differential
EMeasType	Enumerates the different LCA measurement types	NotSet EE EO OE OO
EModBiasOpt	Specify how often a modulator bias voltage optimization has to be performed. Once: only once when the laser is switched on. EverySweep: prior to each measurement started by the LCA. Continuous: the optimization loop runs continuously.	NotSet Once EverySweep Continuous
EOpticalInput	Enumerates the optical inputs on the optical test head's front panel. High power input is comparable to input 2 and standard to input 1.	NotSet Standard HighPower
ERFSwitch	Enumerates the RF switches in a switched LCA system	NotSet Source Receiver
ERFSwitchState	Enumerates the possible settings of the RF switches	UnKnown Thru Intern

Class LCAMeasParams

These are common properties of the LCA measurement parameters

Property	Description	Type	Default value
Wavelength_nm	Specify with which laser wavelength the LCA will measure.	Enum ELaserWvl	NotSet
OpticalPower_dBm	Specify the optical output power of the LCA in dBm	Double	0.0
HighPower_Input	If you are using the high power optical input you have to set the HighPower_Input property to true.	Boolean	False
MeasMode	Specify if you want to do single ended or differential measurements	Enum EMeasMode	SingleEnded
ModBiasOptimization	Specify how often a modulator bias voltage optimization has to be performed	Enum EModBiasOpt	EverySweep

Property	Description	Type	Default value
Advanced	Enable the possibility to overwrite some of the default behavior of the LCA. In advanced mode you can force the LCA to switch the laser on or off independently of the measurement type. You also have additional Optical- and RF-path deembedding possibilities, or can apply additional deembedding on the receiver and the source side, independent of the measurement type.	Boolean – if true, advanced features are active	False
Laser_On	Switch the intern laser on or off. Note: The value of this property is only evaluated in advanced mode. In default mode the laser is switched on or off according to the measurement type.	Boolean – if true, the laser is on	True

The following properties control additional optical path deembedding.

Property	Description	Type	Default value
UseOpticalConnData	With this property you could switch the whole optical path deembedding on or off.	Boolean	False
SrcAttOpt_dB	Specify the optical attenuation on the source path in dB. In default mode only evaluated for O/E and O/O measurements.	Double	0.0
RcvAttOpt_dB	Specify the optical attenuation on the receiver path in dB In default mode only evaluated for E/O and O/O measurements.	Double	0.0

Property	Description	Type	Default value
SrcRefIdx	Specify the refractive index of the source path in dB. In default mode only evaluated for O/E and O/O measurements.	Double	0.0
RcvRefIdx	Specify the refractive index of the receiver path in dB. In default mode only evaluated for E/O and O/O measurements.	Double	0.0
SrcLengthOpt_m	Specify the geometrical length of the source path in m. In default mode only evaluated for O/E and O/O measurements.	Double	0.0
RcvLengthOpt_m	Specify the geometrical length of the receiver path in m. In default mode only evaluated for E/O and O/O measurements.	Double	0.0
UseOpticalS2PFile	Specify if you want to describe the optical paths by the parameters above or by transmission data stored in a s2p file. Only the S21 transmission data is used.	Boolean	False
OptRcvFile	The name of the s2p file to use for additional adaptor deembedding on the receiver side In default mode only evaluated for E/O and O/O measurements.	String	Empty string
OptSrcFile	The name of the s2p file to use for additional adaptor deembedding on the source side. In default mode only evaluated for O/E and O/O measurements.	String	Empty string

These properties are for controlling the additional electrical path deembedding.

Property	Description	Type	Default value
UseElAdaptor	With this property you could switch the whole electrical path deembedding on or off.	Boolean	False
ElRcv1File	The name of the s2p file to use for electrical adaptor deembedding. This property has to be used for receiver side deembedding in single ended measurements or for the receiver port with the lower number in differential measurements.	String	Empty string
ElRcv2File	The name of the s2p file to use for electrical adaptor deembedding. This property has to be used only for the receiver port with the higher number in differential measurements.	String	Empty string

Property	Description	Type	Default value
ElSrc1File	The name of the s2p file to use for electrical adaptor deembedding. This property has to be used for source side deembedding in single ended measurements or for the source port with the lower number in differential measurements.	String	Empty string
ElSrc2File	The name of the s2p file to use for electrical adaptor deembedding. This property has to be used only for the source port with the higher number in differential measurements.	String	Empty string
CalSetUserCal	Name a Calset on the network analyzer which has to be used for the user calibration measurement. If an empty string is passed, the current calset is used. If “NONE” is passed, no calset is applied for the user calibration measurement.	String	Empty string

Class LCAProperties

NOTE These properties are all read-only

Property	**Description**	**Type**	**Default value**
NWAModel	The model number of the network analyzer	String	
NumNWAPorts	The number of ports of the network analyzer	Integer	
NumOpticalInputs	The number of optical inputs of the LCA test head	Integer	
ProductNumber	The product number of the LCA system	String	
SerialNumber	The serial number of the LCA system	String	
SwitchedArchitecture	True: LCA test head has a switched architecture, False: non switched architecture	Boolean	
SoftwareVersion	The version of the LCA server software	String	
SourceWvl	An array showing all available wavelengths of the LCA test head	array ELaserWvl	
MaxPower_dBm	An array holding the maximum optical output power values in dB. These values are correlated to the wavelength values in “SourceWvl” at the same position.	array double	
MinPower_dBm	An array holding the minimum optical output power values in dB. These values are correlated to the wavelength values in “SourceWvl” at the same position.	array double	

Interface ILCARemoteClient

General commands

Sub Connect (ByVal server As String)

Create a connection to an LCA server application.
An LCA client application can only have one open connection to an LCA server at any time.
The LCA server could handle several open connections concurrently.

Parameters: ByVal **server** As String
Host name or IP address of the network analyzer where the LCA server is running.

Return value: No return value.

Sub Disconnect ()

Closes the connection to the LCA server application

Parameters: No parameters.

Return value: No return value.

Function IsConnected() As Boolean

Checks if a connection to an LCA server already exists.

Parameters: No parameters.

Return value: Boolean
True: a connection to an LCA server exists
False: no connection exists

Function Open () As Boolean

Opens an active session on the LCA.

All commands that change the state of the LCA require an active session.
The LCA server allows only one active session at any time.

All actions allowed in a passive session are also allowed in an active session.

Parameters: No parameters.

Return value: Boolean
True: A session has been opened
False: Opening a session failed

Function OpenPassive () As Boolean

Opens an passive session on the LCA.

All commands that just read settings from the LCA require at least an open passive session.

Several passive sessions could be opened concurrently.

Parameters: No parameters.

Return value: Boolean
True: A session has been opened
False: Opening a session failed

Sub Close ()

Closes active session on the LCA.

Parameters: No parameters.

Return value: No return value.

Sub ClosePassive ()

Closes passive session on the LCA.

Parameters: No parameters.

Return value: No return value.

Sub CloseAll ()

Closes the active sessions on the LCA. Any measurements that are currently running are aborted.

This can be useful if an abandoned, open session prevents a successful Open() command. However, be careful not to disturb any other connected client applications.
The LCA automatically closes abandoned sessions after some time (>60s) of inactivity.

Parameters: No parameters.

Return value: No return value.

Sub ResetLCASystem ()

Restarts the LCA server. Open sessions are closed and running measurements are aborted.

A restart is necessary, when the network analyzer application has been restarted or when the LCA testhead has been switched off while the LCA server was running.

Parameters: No parameters.

Return value: No return value.

Sub GetLCAProperties (ByVal properties As RemoteClient.ILCAProperties)

Read out the properties of the LCA system.

Parameters: ByVal **properties** As RemoteClient.ILCAProperties
The properties are written to this LCAProperties object

Return value: No return value.

Sub SetTimeout (ByVal timeout_ms As Integer)

Set the timeout value for the .NET remoting.

A value of 0 or -1 indicates an infinite timeout period, which is also the default value.

The timeout value is set in the .NET remoting layer during execution of the “Connect” command. If you want to set a timeout value, you have to do this before calling the “Connect” command.

If you are using the LCA Remote Client .NET assembly directly, you can also specify the timeout value in the LCARemoteClient constructor.
When using the COM interface you could only use the default constructor, so you have to use this command to specify a non-default timeout value.

Parameters: ByVal **timeout_ms** As Integer
An integer that specifies the number of milliseconds to wait before a .NET remoting request times out

Return value: No return value.

Measurement commands

Sub Init_EE (ByVal parameters As RemoteClient.ILCAMeasParams, ByVal sync As Boolean)

Initializes the LCA for a EE measurement.

Parameters: ByVal **parameters** As RemoteClient.ILCAMeasParams
The measurement parameters for initialization

Optional ByVal **sync** As Boolean
True (default): the call is blocked until initialization is complete
False: the call returns immediately.

For synchronization use the synchronization methods WaitForOPC or OperationComplete

Return value: No return value.

Sub Init_EO (ByVal parameters As RemoteClient.ILCAMeasParams, ByVal sync As Boolean)

Initializes the LCA for an EO measurement.

Parameters: ByVal **parameters** As RemoteClient.ILCAMeasParams
The measurement parameters for initialization

Optional ByVal **sync** As Boolean
True (default): the call is blocked until initialization is complete
False: the call returns immediately. For synchronization use the synchronization methods WaitForOPC or OperationComplete

Return value: No return value.

Sub Init_OE (ByVal parameters As RemoteClient.ILCAMeasParams, ByVal sync As Boolean)

Initializes the LCA for an OE measurement.

Parameters: ByVal **parameters** As RemoteClient.ILCAMeasParams
The measurement parameters for initialization

Optional ByVal **sync** As Boolean
True (default): the call is blocked until initialization is complete
False: the call returns immediately.

For synchronization use the synchronization methods WaitForOPC or OperationComplete

Return value: No return value.

Sub Init_OO (ByVal parameters As RemoteClient.ILCAMeasParams, ByVal sync As Boolean)

Initializes the LCA for an OO measurement.

Parameters: ByVal **parameters** As
RemoteClient.ILCAMeasParams
The measurement parameters for initialization

Optional ByVal **sync** As Boolean
True (default): the call is blocked until initialization is complete
False: the call returns immediately.

For synchronization use the synchronization methods WaitForOPC or OperationComplete

Return value: No return value.

Sub LoadOOTxCalData (ByVal parameters As RemoteClient.ILCAMeasParams, ByVal filename As String, ByVal sync As Boolean)

Use this command instead of Init_OE if you want the LCA to load and use previously saved user calibration data.

The loaded user calibration data will be used by the LCA until the next initialization command is called.

See also: SaveUserCalData

Parameters:	ByVal **parameters** As RemoteClient.ILCAMeasParams The measurement parameters for initialization
	ByVal **filename** As String The name of the file containing the user calibration data
	Optional ByVal **sync** As Boolean True (default): the call is blocked until initialization is complete False: the call returns immediately.
	For synchronization use the synchronization methods WaitForOPC or OperationComplete
Return value:	No return value.

Sub LoadOETxCalData (ByVal parameters As RemoteClient.ILCAMeasParams, ByVal filename As String, ByVal sync As Boolean)

Use this command instead of Init_OE if you want the LCA to load and use previously saved user calibration data.

The loaded user calibration data will be used by the LCA until the next initialization command is called.
See also: SaveUserCalData

Parameters: ByVal **parameters** As RemoteClient.ILCAMeasParams
The measurement parameters for initialization

ByVal **filename** As String
The name of the file containing the user calibration data

Optional ByVal **sync** As Boolean
True (default): the call is blocked until initialization is complete
False: the call returns immediately. For synchronization use the synchronization methods WaitForOPC or OperationComplete

Return value: No return value.

Sub Measure (ByVal continuous As Boolean, ByVal sync As Boolean)

NOTE

Be careful when calling a continuous measurement in synchronous mode. Since the synchronous call blocks the program execution of the calling thread, you can't stop this measurement from the calling thread. It can only be stopped from another thread.

Triggers a measurement on the LCA.
If you call a continuous measurement while another measurement is running, the original measurement is stopped without starting a new measurement.
If you call a single measurement while another measurement is running, this measurement is stopped and a new single measurement is started.

It requires that one of the initialization routines above has been called. If no measurement type has been initialized, an “InvalidOperationException” is thrown. The type of the measurement is the one initialized by the last “Init_XX” or “LoadXXTxCalData” call.

You should trigger your DUT measurements with this routine, as it takes care of optical DC power dependent deembedding and modulator bias voltage optimization.

For synchronization use the synchronization methods WaitForOPC or OperationComplete.

Parameters: ByVal **continuous** As Boolean
True: measurements are done continuously
False (default): a single measurement is triggered

Optional ByVal **sync** As Boolean
True (default): the call is blocked until initialization is complete
False: the call returns immediately.

Return value: No return value.

Sub SaveUserCalData (ByVal filename As String)

Save the measured user calibration data into a s2p-file.

If no user calibration data has been measured during last OE or OO initialization, default values are stored.

This command is only allowed when OE or OO measurement mode is inititalized.

Parameters: ByVal **filename** As String
The filename, where the data should be stored.

Return value: No return value.

Sub Abort ()

Aborts a currently running measurement or initialization.

Parameters: No parameters.

Return value: No return value.

Sub WaitForOPC ()

Waits until the last asynchronously called command has finished execution. Exceptions thrown during execution of an asynchronously called command could be caught when calling WaitForOPC() or OperationComplete().

Se also property: OperationComplete()

Parameters: No parameters.

Return value: No return value.

Properties

Reading these properties requires only a passive session, while setting these properties requires an active session.

LaserWvl_nm As RemoteClient.ELaserWvl

Get or set the current wavelength of the LCA optical output in nanometers.

Parameters: No parameters.

LaserPower_dBm As Double

Get or set the current power of the LCA optical output in dBm

Parameters: No parameters.

LaserState As RemoteClient.ELaserState

Get or set the current state of the LCA optical output

Parameters: No parameters.

OpticalInput As RemoteClient.EOpticalInput

Get or set the current optical input of the LCA testhead

Parameters: No parameters.

RFSwitchState (ByVal RFSwitch As RemoteClient.ERFSwitch)

Setting the RF switches in the LCA testhead. With a non switched LCA system, setting this property has no effect. Trying to set this property to UnKnown, is ignored. Reading this property from a non switched system will always return UnKnown.

Parameters: ByVak **RFSwich** As RemoteClient.ERFSwitch
The switch you want to read from or you want to set.

RFPowerFwd_dBm As Double

Gets or sets the RF power on the network analyzer ports for forward measurements. To set this property back to the factory defined default value, set it to Double.NaN or a value < -200dBm.

Parameters: No parameters.

RFPowerRev_dBm As Double

Gets or sets the RF power on the network analyzer ports for reverse measurements. To set this property back to the factory defined default value, set it to Double.NaN or a value < -200dBm.

Parameters: No parameters.

ReadOnly OpticalDCPower_dBm As Double

Get the actual optical DC power, measured by the optical powermeter built into the LCA testhead

Parameters: No parameters.

ReadOnly LCAProperties As RemoteClient.ILCAProperties

See the command GetLCAProperties

Parameters: No parameters.

ReadOnly CurrentMeasType As RemoteClient.EMeasType

Get the measurement type which has been initialized by the last call to one of the Init_XX commands or by one of the LoadXXTxCalData commands.

Parameters: No parameters.

ReadOnly OperationComplete As Boolean

Get the operation status of the last asynchronously called command. Exceptions thrown during execution of an asynchronously called command could be caught when calling WaitForOPC() or OperationComplete().

Parameters: No parameters.

The LCA SCPI Interface

Overview

The LCA instrument is a combined instrument. It is a network analyzer with additional hardware and software to become the LCA. The network analyzer already offers a SCPI interface on different ports. Now the new LCA SCPI interface extends the existing LCA application. It is implemented with the Keysight Translator Framework and the LCA Remote Server. Each SCPI command is intercepted and linked to an LCA Remote Interface method. The LCA SCPI interface is not completely IEEE compliant. It only implements the most necessary common commands besides the application specific commands.

Port Types

The LCA SCPI interface is available either on a network socket or on the device USB port. Other ports like GPIB are not supported. You may select and configure one of the available types. Using both ports in parallel is not supported.

Socket Port
The LCA SCPI talker/listener runs on port 5026. The network analyzer SCPI interface runs on port 5025. You may run both SCPI interfaces for the
LCA and the network analyzer application in parallel, since they take different socket ports.

USB Port
The LCA system is an integrated system. The system has only one USB device port which can be used to control the application from a remote PC. Therefore you can use the USB port to control either the network analyzer via SCPI or the LCA application via SCPI. You can’t control both applications over the USB port at the same time.

You always have to run the network analyzer application to get the LCA functionality. Therefore if you only run the network analyzer and NOT the LCA SCPI interface, the USB device port is taken by the network analyzer SCPI talker/listener. When you first connect your PC with a USB cable to the LCA (combined instrument), you get the Network Analyzer identification string if you send the *IDN? query.

If you start the LCA SCPI interface and configure it to run on the USB device port, you will get theidentification string for the LCA instrument when
you send the *IDN? SCPI query.

However, if you run the LCA SCPI interface on the socket port and have connected your PC via USB with the LCA instrument, the network analyzer
identification string will still be returned.

After a system reboot, the USB device port is always taken by the network analyzer SCPI interface by default. The LCA SCPI interface has to be started manually. If you run the LCA SCPI interface on the USB device port and want to switch to the socket port, you have to stop the LCA SCPI interface first, then change the configuration to socket port and save it. This action will restart the network analyzer application automatically, to reclaim the USB device port for the network analyzer. Now the LCA SCPI interface can be restarted with the new configuration.

GPIB port

The LCA SCPI interface doesn’t support the GPIB port. However you may control the network analyzer application through SCPI over the GPIB port.
This gives you the possibility to control the instrument independent of LAN by controlling the LCA application through SCPI over USB and the
network analyzer application through SCPI over GPIB.

Configuration

Select the communication port for your LCA SCPI interface, either the LAN socket port 5026 or the USB device port. Run the Agilent.LCA.SCPI.Config.exe program or click on the LCA SCPI Configuration shortcut on the network analyzer macro list to select the preferred port. The LCA SCPI talker/listener runs on the socket port 5026 by default. The port is not selectable to avoid conflicts with the network analyzer SCPI interface, which runs on port 5025.

For support purposes, you may turn the logging on or off. The logging stores all program outputs into a file. Note: it may fill up your hard disk if you
run the SCPI interface in logging mode for a long time.

When done with configuration, click the “Save Config” button to store all settings. After saving the settings, the LCA SCPI module will adopt the modified configuration when you click on the “Start SCPI” button.

Start/Stop the LCA SCPI Module

All LCA modules require the network analyzer application. It should always start after a system reboot automatically. If the network analyzer is not running, please start it manually. Like all other LCA modules, the LCA SCPI module does not start automatically. You have to start it manually. Use the Agilent.LCA.SCPI.Conf.exe program to start or stop the LCA SCPI interface. To launch this program, you may either use the LCA SCPI link in the network analyzer GUI macro list under utilities, or the shortcut LCA SCPI Interface on the desktop or in the program menu.

The LCA SCPI interface is implemented on the LCA Remote Interface methods and the Agilent Translator Framework. Therefore the LCA Server starts automatically when you start the LCA SCPI interface. When you click on the “Start SCPI” button on the SCPI configuration form, the Agilent Translator Framework starts and loads the Agilent.LCA.SCPI.Module. The LCA Server cannot handle more than one session. Therefor you can run either the LCA Measurement Setup application or the LCA SCPI interface, but not both in parallel. This is the same for the LCA Remote Client. It also connects to the LCA Server and therefore the SCPI interface cannot run at the same time.

LCA SCPI Commands

Overview

The LCA SCPI commands do not fulfill the IEEE standard. They just offer a simple way to control the LCA application on a LAN dependent socket port or on a USB port.
Except for the *IDN? and :SYST:ERR? Commands, there is always a direct relation between a SCPI command and a method or property of the LCA.Net Remote Interface.

Command Tree

```
*CLS
[:LCA]:PNUMber? -> <string>
[:LCA]:SNUMber? -> <string>
[:LCA]:SOFTware:VERSion? -> <string>
:LOAD:OO:CALibration:NAME noquery “<string>”
:LOAD:OE:CALibration:NAME noquery “<string>”
:MEASurement:ABORt
:MEASurement:CALData:SAVE noquery “<string>”
:MEASurement:CURRent:TYPE? -> <string>
:MEASurement:INITialize:EE
:MEASurement:INITialize:EO
:MEASurement:INITialize:OE
:MEASurement:INITialize:OO
:MEASurement:STARt <SINGle|CONTinuous>
```

:NWA:MODel? -> <string>
:NWA:PORT:NUMBer? -> <integer>

*OPC? -> <0|1> as string
:PARameter:ADVAnced:MODE /? -> <0|1> as string
:PARameter:ELECtrical:PATH:DEEMbedding /? -> <0|1> as string
:PARameter:ELECtrical:RECeiver:S2PFile[n] /? <string> (index n = 1|2)
:PARameter:ELECtrical:SOURce:S2PFile[n] /? -> <string > (index n = 1|2)
:PARameter:MEASurement:MODE /? -> <SINGel|DIFFerential >
:PARameter:MODUlator:BIAS:MODE /? <CONTinuous|EVERysweep|ONCE>
:PARameter:OPTical:INPut:POWer:HIGH /? -> <0|1> as string
:PARameter:OPTical:OUTput:POWer /? -> <double>
:PARameter:OPTical:PATH:DEEMbedding /? <0|1> as string
:PARameter:OPTical:RECeiver:S2PFile /? -> <string >
:PARameter:OPTical:S2PFile:USE /? -> <0|1> as string
:PARameter:OPTical:SOURce:S2PFile /? -> <string >
:PARameter:RECeiver:ATTenuation /? -> <double>
:PARameter:RECeiver:PATH:LENGth /? -> <double>
:PARameter:RECeiver:REFR:INDex /? -> <double>
:PARameter:SOURce:ATTenuation /? -> <double>
:PARameter:SOURce:POWer:STATe /? <0|1off|on >
:PARameter:SOURce:PATH:LENGth /? -> <double>
:PARameter:SOURce:REFR:INDex /? -> <double>
:PARameter:USER:CALIbration:CALSet /? -> <string>
:PARameter:WAVelength /? -> <string>

:RF:POWer:FWD /? -> <double> unit is dBm
:RF:POWer:REVerse /? -> <double> unit is dBm
:RF:SWITch:STATe /? <RECeiver|SOURce >,<INTern|THRu >
:SOURce{n}:MAXPower? qonly -> <string>, n = index of array
:SOURce{n}:MINPower? qonly -> <string>, n = index of array
:SOURce:POWer /? -> <double> {dBm}
:SOURce:STATe /? <ON|OFF|0|1>
:SOURce:WAVelength /? -> <1310|1550> as string
:SOURce:WAVelength:ALL? qonly -> <string>
:THEAd:INPut:MODe /? ->< STD|HIGH>
:THEAd:INPut:POWer? qonly <double> unit is dBm
:THEAd:INPut:NUMBers? qonly -> <integer>
:THEAd:SWITched:ARCHitecture? qonly -> <0|1>

Command Details

Command:	***CLS**
syntax:	*CLS
description:	Clears the system error queue.
parameters:	none
response:	none
example:	*cls

Command:	**[:LCA]:PNUMber?**
syntax:	[:LCA]:PNUMber?
description:	The product number of the LCA system
parameters:	none
response:	string
C#:	(property) ProductNumber
example:	:PNUM? -> N4373B

Command:	**[:LCA]:SNUMber?**
syntax:	[:LCA]:SNUMber?
description:	The serial number of the LCA system
parameters:	none
response:	string
C	
C#:	(property) SerialNumber
example:	:SNUMber? ->,MY49151038

Command:	**[:LCA]:SOFTware:VERSion?**
syntax:	[:LCA]:SOFTware:VERSion?
description:	The version of the LCA server software
parameters:	none
response	string
C#:	(property)
example	:SOFT:VERS? -> 2.3.10.2

Command:	**:LOAD:OO:CALibration:NAME**
syntax:	:LOAD:OO:CALibration:NAME<wsp>"<path string>"
description:	Use this command instead of Init_OO if you want the LCA to load and use previously saved user calibration data
parameters:	"<string>" path and filename enclosed in double quotes
response:	none
C#:	(method) LoadOOTxCalData
example:	:LOAD:OO:CAL:NAME "c:\temp\test.s2p"

Command:	**:LOAD:OE:CALibration:NAME**
syntax:	:LOAD:OE:CALibration:NAME<wsp>"<path string>"
description:	Use this command instead of Init_OE if you want the LCA to load and use previously saved user calibration data
parameters:	"<string>" path and file name enclosed in double quotes
response:	none
C#:	(method) LoadOETxCalData
example:	:LOAD:OE:CAL:NAME "c:\temp\test.snp"

Command: **:MEASurement:ABORt**
syntax: :MEASurement:ABORt
description: Aborts a currently running measurement or initialization.
parameters: none
response: none
C#: (method) Abort()
example: :MEAS:ABOR

Command: **:MEASurement:CALData:SAVE**
syntax: :MEASurement:CALData:SAVE<wsp>”<path string>”
description: Save the measured user calibration data into a s2p-file.
parameters: “<string>” path and file name enclosed in double quotes
response: none
C#: (method) SaveUserCalData()
example: :MEAS:CALD:SAVE “c:\temp\test.s2p”

Command: **:MEASurement:CURRent:TYPE?**
syntax: :MEASurement:CURRent:TYPE?
description: Get the measurement type that has been initialized
by the last call to one of the :MEAS:INIT XX commands
or by one of the.:LOAD:XX: commands.
parameters: none
response: <string> NotSet | EE | EO | OE | OO
C#: (method) CurrentMeasType()
example: :MEAS:CURR:TYPE? -> OO

Command: **:MEASurement:INITialize:EE**
syntax: :MEASurement:INITialize:EE
description: Initializes the LCA for an EE measurement
parameters: none
response: none
C#: (method) Init_EE()
example: :MEAS:INIT:EE

command: **:MEASurement:INITialize:EO**
syntax: :MEASurement:INITialize:EO
description: Initializes the LCA for an EO measurement.
parameters: none
response: none
C#: (method) Init_EO()
example: :MEAS:INIT:EO

command: **:MEASurement:INITialize:OE**
syntax: :MEASurement:INITialize:OE
description: Initializes the LCA for an OE measurement
parameters: none
response: none
C#: (method) Init_OE
example: :MEAS:INIT:OE

command: **:MEASurement:INITialize:OO**
syntax: :MEASurement:INITialize:OO
description: Initializes the LCA for an OO measurement
parameters: none
response: none
C#: (method) Init_OO()
example: :MEAS:INIT:OO

command: **:MEASurement:STARt**
syntax: MEASurement:STARt<wsp>[SINGle|CONTinuous]

description: Triggers a measurement on the LCA. If you call a continuous measurement while another measurement is running, the original measurement is stopped without starting a new measurement
parameters: <string> SINGle | CONTinuous
response: none
C#: (method) Measure()
example: :MEAS:STAR CONT

command: **:NWA:MODel?**
syntax: :NWA:MODel?
description: The model number of the network analyzer
parameters: none
response: string
C#: (property) NWAModel
example: :NWA:MOD? -> N5245A

command: **:NWA:PORT:NUMBer?**
syntax: :NWA:PORT:NUMBer?
description: The number of network analyzer ports
parameters: none
response: integer
C#: (property) NumNWAPorts
example: :NWA:PORT:NUMBer? -> 4

command: ***OPC?**
syntax: *OPC?
description: Retrieves the operation complete state
parameters: none
response: <string> 0 | 1
C#: (method) OperationComplete()
example: *OPC? -> 1

command: **:PARameter:ADVAnced:MODE?**
syntax: :PARameter:ADVAnced:MODE?
description: Returns 1 if advanced mode is enabled. In advanced mode you can force the LCA to switch the laser on or off, independent of the measurement type. You also have additional optical- and RF-path de-embedding possibilities, or can apply additional de-embedding on the receiver and the source side, independent of the measurement type.
parameters: none
response: <string> 0 | 1
C#: (property) Advanced
example: :PAR:ADVA:MODE? -> 0

command: **:PARameter:ADVAnced:MODE**
syntax: :PARameter:ADVAnced:MODE<wsp>ON | OFF | 1 | 0
description: Enables or disables advance mode, which allows changing some default settings. In advanced mode you can force the LCA to switch the laser on or off, independent of the measurement type. You also have additional optical- and RF-path de-embedding possibilities, or can apply additional de-embedding on the receiver and the source side, independent of the measurement type.
parameters: <string> ON | OFF | 1 | 0
response: none
C#: (property) Advanced
example: :PAR:ADVA:MODE ON

command: **:PARameter:ELECtrical:PATH:DEEMbedding?**

syntax: :PARameter:ELECtrical:PATH:DEEMbedding?

description: Retrieves the property which shows whether the whole electrical path de-embedding is switched on or off.

parameters: none

response: <string> 1 | 0

C#: (property)UseElAdaptor

example: :PAR:ELEC:PATH:DEEM? -> 0

command: **:PARameter:ELECtrical:PATH:DEEMbedding**

syntax: :PARameter:ELECtrical:PATH:DEEMbedding<wsp>ON|OFF|1|0

description: Sets the property which enables or disables the whole electrical path de-embedding .

parameters: <string> ON | OFF | 1 | 0

response: none

C#: (property)UseElAdaptor

example: :PAR:ELEC:PATH:DEEM OFF

command: **:PARameter:ELECtrical:RECeiver:S2PFile[1 - 2]?**

syntax: :PARameter:ELECtrical:RECeiver:S2P:FILE[1 - 2]:NAME?

description: Gets the name of the s2p file to use for electrical adaptor de-embedding. File index 1 has to be used for receiver side de-embedding in single-ended measurements or for the receiver port with the lower number in differential measurements. Index 2 has to be used only for the receiver port with the higher number in differential measurements

parameters: none

response: <string>

C#: (property) ElRcv1File / ElRcv2File

example: :PAR:ELEC:REC:S2PFile? -> c:\temp\test.s2p

command: **:PARameter:ELECtrical:RECeiver:S2PFile[1 - 2]**

syntax: :PARameter:ELECtrical:RECeiver:S2P:FILE[1 - 2]:NAME<wsp>"<path string>"

description: Sets the name of the s2p file to use for electrical adaptor de-embedding. File index 1 has to be used for receiver side de-embedding in single-ended measurements or for the receiver port with the lower number in differential measurements. Index 2 has to be used only for the receiver port with the higher number in differential measurements

parameters: "<string>" path and file name

response: none

C#: (property) ElRcv1File / ElRcv2File

example: :PAR:ELEC:REC:S2PFile"c:\temp\test.s2p"

command: **:PARameter:ELECtrical:SOURce:S2PFile[1 - 2]?**
syntax: :PARameter:ELECtrical:SOURce:S2PFile[[1- 2]:NAME?
description: Gets the name of the s2p file to use for electrical adaptor de-embedding. This property has to be used with file index 1 for source side de-embedding in single-ended measurements or for the source port with the lower number in differential measurements. Index 2 is the file for the source port with the higher number in differential measurements.
parameters: none
response: <string> path and file name
C#: (property) ElSrc1File / ElSrc2File
example: :PARameter:ELECtrical:SOURce:S2PFile1? -> c:\temp\test.s2p

command: **:PARameter:ELECtrical:SOURce:S2PFile[1 - 2]**
syntax: :PARameter:ELECtrical:SOURce:S2PFile[[1- 2]<wsp>"<path string>"
description: Sets the name of the s2p file to use for electrical adaptor de-embedding. This property has to be used with file index 1 for source side de-embedding in single-ended measurements or for the source port with the lower number in differential measurements. Index 2 is the file for the source port with the higher number in differential measurements
parameters: "<string>" path and file name
response: none
C#: (property) ElSrc1File / ElSrc2File
example: :PARameter:ELECtrical:SOURce:S2PFile1 "c:\temp\test.s2p"

command: **:PARameter:MEASurement:MODE?**
syntax: :PARameter:MEASurement:MODE?
description: Returns setting for selecting single-ended or differential measurements
parameters: none
response: <string> DIFFerential|SINGleended | NOTSet
C#: (property) MeasMode
example: :PAR:MEAS:MODE? -> NotSet

command: **:PARameter:MEASurement:MODE**
syntax: :PARameter:MEASurement:MODE<wsp>DIFFerential|SINGleended|NOTSet
description: Specify single ended or differential measurements
parameters: <string> DIFFerential|SINGleended | NOTSet
response: none
C#: (property) MeasMode
example: :PAR:MEAS:MODE SING

command: **:PARameter:MODUlator:BIAS:MODE?**
syntax: :PARameter:MODUlator:BIAS:MODE?
description: Returns how often a modulator bias voltage optimization will be performed
parameters: none
response: <string> Continuous|EverySweep|Once
C#: (property) ModBiasOptimization
example: :PAR:MODU:BIAS:MODE? -> EverySweep

command: **:PARameter:MODUlator:BIAS:MODE**
syntax: :PARameter:MODUlator:BIAS:MODE<wsp>CONT|EVER|ONCE
description: Specify how often a modulator bias voltage optimization will be performed
parameters: <string> CONTinuous|EVERysweep|ONCE
response: none
C#: (property) ModBiasOptimization
example: :PAR:MODU:BIAS:MODE EVER

command: **:PARameter:OPTical:INPut:POWer:HIGH?**
synstax: :PARameter:OPTical:INPut:POWer:HIGH?
description: Returns the state of the high power input property.
parameters: none
response: <string> 1 | 0, input power high true = 1, false = 0
C#: (property) HighPower_Input
example: :PAR:OPT:INP:POWer:HIGH? -> 0

command: **:PARameter:OPTical:INPut:POWer:HIGH**
syntax: :PARameter:OPTical:INPut:POWer:HIGH<wsp>ON|OFF|1|0
description: Gets the state of the high power input property.
parameters: <string> ON | 1 enables high power input, OFF | 0 disables high power input
response: none
C#: (property) HighPower_Input
example: :PAR:OPT:INP:POWer:HIGH ON

command: **:PARameter:OPTical:OUTput:POWer?**
synstax: :PARameter:OPTical:OUTput:POWer?
description: Returns the optical output power of the LCA in dBm.
parameters: none
response: <double> power value, the default unit is dBm.
C#: (property) OpticalPower_dBm
example: :PAR:OPT:OUT:POWer -> -1

command: **:PARameter:OPTical:OUTput:POWer**
synstax: :PARameter:OPTical:OUTput:POWer<ws><*power*>
description: Specify the optical output power of the LCA in dBm.
parameters: *power* <double>, power value in dBm
response: none.
C#: (property) OpticalPower_dBm
example: :PAR:OPT:OUT:POWer -1

command: **:PARameter:OPTical:PATH:DEEMbedding?**
synstax: :PARameter:OPTical:PATH:DEEMbedding?
description: Returns whether the whole optical path de-embedding is set on or off.
parameters: none
response: <string> 1 = optical path de-embedding is enabled, 0 = disabled
C#: (property) UseOpticalConnData
example: :PAR:OPT:PATH:DEEM? -> 1

command: **:PARameter:OPTical:PATH:DEEMbedding**
syntax: :PARameter:OPTical:PATH:DEEMbedding<wsp>ON|1|OFF|0
description: Switches the whole optical path de-embedding on or off.
parameters: <string> ON | 1 = enabled optical path de-embedding, OFF | 0 = disable
response: none
C#: (property) UseOpticalConnData
example: :PAR:OPT:PATH:DEEM? -> 1

command:	**:PARameter:OPTical:RECeiver:S2PFile?**
synstax:	:PARameter:OPTical:RECeiver:S2PFile?
description:	Returns the name of the s2p file to use for additional adaptor de-embedding on the receiver side. In default mode, only evaluated for E/O and O/O measurements.
parameters:	none
response:	<string> path and file name
C#:	(property) OptRcvFile
example:	:PAR:OPT:REC:S2PF? -> c:\temp\test1.s2p

command:	**:PARameter:OPTical:RECeiver:S2PFile**
syntax:	:PARameter:OPTical:RECeiver:S2PFile<wsp>"<path string>"
description:	Sets the name of the s2p file which is used for additional adaptor de-embedding on the receiver side. In default mode only evaluated for E/O and O/O measurements.
parameters:	"<string>" file name and path enclosed in double quotes
response:	none
C#:	(property) OptRcvFile
example:	:PAR:OPT:REC:S2PF "c:\temp\test1.s2p"

command:	**:PARameter:OPTical:S2PFile:USE?**
syntax:	:PARameter:OPTical:S2PFile:USE?
description:	Returns whether the optical paths are described by transmission data stored in an s2p file. Only the S21 transmission data is used
parameters:	none
response:	<string> 1 = s2p file use enabled, 0 = disabled
C#:	(property) UseIOpticalS2PFile
example:	:PAR:OPT:S2PF:USE? -> 0

command:	**:PARameter:OPTical:S2PFile:USE**
syntax:	:PARameter:OPTical:S2PFile:USE<wsp>ON|1|OFF|0
description:	enables or disables description of the optical paths by transmission data stored in an s2p file. Only the S21 transmission data is used
parameters:	<string> ON | 1 = s2p file use enabled, OFF | 0 = disabled
response:	none
C#:	(property) UseIOpticalS2PFile
example:	:PAR:OPT:S2PF:USE ON

command:	**:PARameter:OPTical:SOURce:S2PFile?**
syntax:	:PARameter:OPTical:SOURce:S2PFile?
description:	Retrieves the name of the s2p file to use for additional adaptor de-embedding on the source side. In default mode only evaluated for O/E and O/O measurements.
parameters:	none
response:	<string> file name and path of the s2p file on the LCA system.
C#:	(property) OptSrcFile
example:	:PAR:OPT:SOUR:S2PF? -> c:\temp\test1.s2p

command: **:PARameter:OPTical:SOURce:S2PFile**
syntax: :PARameter:OPTical:SOURce:S2PFile<wsp>"<path string>"
description: Specifies the name of the s2p file to use for additional adaptor de-embedding on the source side. In default mode only evaluated for O/E and O/O measurements.
parameters: "<string>" the file name and path enclosed in double quotes.
response: <string> file name and path of the s2p file on the LCA system.
C#: (property) OptSrcFile
example: :PAR:OPT:SOUR:S2PF "c:\temp\test1.s2p"

command: **:PARameter:RECeiver:ATTenuation?**
syntax: :PARameter:RECeiver:ATTenuation?
description: Retrieves the optical attenuation on the receiver path. In default mode only evaluated for E/O and O/O measurements
parameters: none
response: <double> attenuation value, default unit is dB
C#: (property) RcvAttOpt_dB
example: :PAR:REC:ATT? -> 3

command: **:PARameter:RECeiver:ATTenuation**
syntax: :PARameter:RECeiver:ATTenuation <wsp><*attenuation*>
description: Specifies the optical attenuation on the receiver path. In default mode only evaluated for E/O and O/O measurements
parameters: *attenuation* <double> attenuation value, default unit is dB
response: none
C#: (property) RcvAttOpt_dB
example: :PAR:REC:ATT 2

command: **:PARameter:RECeiver:PATH:LENGth?**
syntax: :PARameter:RECeiver:PATH:LENGth?
description: Retrieves the geometrical length of the receiver path in m. In default mode only evaluated for E/O and O/O measurements
parameters: none
response: <double> The path length value, default unit is meter.
C#: (property) RcvLengthOpt_m
example: :PAR:REC:PATH:LENG? -> 0.3

command: **:PARameter:RECeiver:PATH:LENGth**
syntax: :PARameter:RECeiver:PATH:LENGth<wsp> <*length*>
description: Specifies the geometrical length of the receiver path in m. In default mode only evaluated for E/O and O/O measurements
parameters: *length* <double> path length value, default unit is meter.
response: none.
C#: (property) RcvLengthOpt_m
example: :PAR:REC:PATH:LENG 0.45

command: **:PARameter:RECeiver:REFR:INDex?**
syntax: :PARameter:RECeiver:REFR:INDex ?
description: Retrieves the refractive index of the receiver path in dB. In default mode only evaluated for E/O and O/O measurements.
parameters: none
response: <double> the refractive index value, unit is dB.
(property) RcvRefIdx
example: :PAR:REC:REFR:IND? -> 0

command: **:PARameter:RECeiver:REFR:INDex**
syntax: :PARameter:RECeiver:REFR:INDex<wsp><*index*>
description: Specifies the refractive index of the receiver path in dB. In default mode only evaluated for E/O and O/O measurements.
parameters: *index* <double> the receiver refractive value, unit id dB.
response: none
C#: (property) RcvRefIdx
example: :PAR:REC:REFR:IND 1.3

command: **:PARameter:SOURce:ATTenuation?**
syntax: :PARameter:SOURce:ATTenuation?
description: Retrieves the optical attenuation on the source path. In default mode only evaluated for O/E and O/O measurements.
parameters: none
response: <double> attenuation value in dB
C#: (property) SrcAttOpt_dB
example: :PAR:SOUR:ATT? -> 0

command: **:PARameter:SOURce:ATTenuation**
syntax: :PARameter:SOURce:ATTenuation<wsp><*attenuation*>
description: Specifies the optical attenuation on the source path. In default mode only evaluated for O/E and O/O measurements.
parameters: *attenuation* <double> attenuation value in dB
response: none
C#: (property) SrcAttOpt_dB
example: :PAR:SOUR:ATT 0.4

command: **:PARameter:SOURce:POWer:STATe?**
syntax: :PARameter:SOURce:POWer:STATe?
description: Retrieves the internal laser state, on or off. Note: the value of this property is only evaluated in advanced mode. In default mode the laser is switched on or off according to the measurement type.
parameters: none
response: <string> 1 = internal laser is on, 0 = internal laser is off
C#: (property) Laser_On
example: :PAR:SOUR:POW:STAT? -> 1

command: **:PARameter:SOURce:POWer:STATe**
syntax: :PARameter:SOURce:POWer:STATe<wsp>ON|1|OFF|0
description: Switches the internal laser on or off. Note: The value of this property is only evaluated in advanced mode. In default mode the laser is switched on or off according to the measurement type
parameters: <string> ON | 1 to switch the laser on, OFF | 0 to switch the laser off
response: none
C#: (property) Laser_On
example: :PAR:SOUR:POW:STAT ON

command: **:PARameter:SOURce:PATH:LENGth?**
syntax: :PARameter:SOURce:PATH:LENGth?
description: Retrieves the geometrical length of the source path in m . In default mode only evaluated for O/E and O/O measurements
parameters: .none
response: <double> the path length in meter
C#: SrcLengthOpt_m
example: :PAR:SOUR:PATH:LENG? -> 0.27

command: **:PARameter:SOURce:PATH:LENGth**
synstax: :PARameter:SOURce:PATH:LENGth<wsp><*length*>
description: Specifies the geometrical length of the source path in m. In default mode only evaluated for O/E and O/O measurements
parameters: *length* <double> the path length value, default unit is meter.
response: none
C#: SrcLengthOpt_m
example: :PAR:SOUR:PATH:LENG 0.42

command: **:PARameter:SOURce:REFR:INDex?**
syntax: :PARameter:SOURce:REFR:INDex?
description: Retrieves the refractive index of the source path in dB. In default mode only evaluated for O/E and O/O measurements.
parameters: none
response: <double> the refractive index
C#: (property) SrcRefIdx
example: :PAR:SOUR:REFR:IND? -> 0

command: **:PARameter:SOURce:REFR:INDex**
syntax: :PARameter:SOURce:REFR:INDex<wsp><*index*>
description: Specifies the refractive index of the source path in dB. In default mode only evaluated for O/E and O/O measurements.
parameters: *index* <double> the refractive index value in dB
response: none
C#: (property) SrcRefIdx
example: :PAR:SOUR:REFR:IND 0.13

command: **:PARameter:USER:CALIbration:CALSet?**
syntax: :PARameter:USER:CALIbration:CALSet?
description: Retrieves the name of a Calset on the network analyzer to be used for the user calibration measurement. If an empty string is returned, the current Calset is used. If “NONE” is returned, no Calset is applied for the user calibration measurement.
parameters: none
response: <string> NONE | path and file name
C#: (property) CalSetUserCal
example: :PAR:USER:CAL:CALS? -> c:\temp\calset1.s2p

command: **:PARameter:USER:CALIbration:CALSet**
syntax: :PARameter:USER:CALIbration:CALSet<wsp>[NONE | “<path string>”]
description: Specifies the name of a Calset on the network analyzer to be used for the user calibration measurement. If an empty string is passed, the current Calset is used. If “NONE” is passed, no Calset is applied for the user calibration measurement.
parameters: NONE| <string> |No argument, None or the path and file name surrounded by double quotes.
response: none
C#: (property) CalSetUserCal
example: :PAR:USER:CAL:CALS “c:\temp\calset1.s2p”

command: **:PARameter:WAVelength?**
syntax: :PARameter:WAVelength?
description: Returns the laser wavelength set on the LCA.
parameters: none
response: <string> The wavelength and unit as a string.
C#: (property) Wavelength_nm
example: :PAR:WAV? -> Wvl_1550nm

command: **:PARameter:WAVelength**
syntax: :PARameter:WAVelength<wsp>850|1310|1550
description: Specifies with which laser wavelength the LCA will measure
parameters: <string> 850 | 1310 | 1550
response: none
C#: (property) Wavelength_nm
example: :PAR:WAV 1550

command: **:RF:POWer:FWD?**
syntax: :RF:POWer:FWD?
description: Gets the RF power on the network analyzer ports for forward measurements
parameters: none
response: <double> forward power value in dBm
C#: (property) RFPowerFwd_dBm
example: :RF:POWer:FWD? -> -8

command: **:RF:POWer:FWD**
syntax: :RF:POWer:FWD<wsp><*power*>
description: Sets the RF power on the network analyzer ports for forward measurements
parameters: *power* <double> forward power value in dBm
response: none
C#: (property) RFPowerFwd_dBm
example: :RF:POWer:FWD -1

command: **:RF:POWer:REVerse?**
syntax: :RF:POWer:REVerse?
description: Gets the RF power on the network analyzer ports for reverse measurements.
parameters: none
response: <double> RF reverse power value in dBm.
C#: (property) RFPowerRev_dBm
example: :RF:POW:REV? -> -8

command: **:RF:POWer:REVerse**
syntax: :RF:POWer:REVerse<wsp><*power*>
description: Sets the RF power on the network analyzer ports for reverse measurements. To set this property back to the factory defined default value, set it to Double.NaN or a value < -200dBm.
parameters: *power* <double> RF reverse power value in dBm.
response: none
C#: (property) RFPowerRev_dBm
example: :RF:POW:REV -4

command: **:RF:SWITch:STATe?**
syntax: :RF:SWITch:STATe?
description: Retrieves the RF switch settings in the LCA test-head. With a non-switched LCA system, setting this property has no effect.. Reading this property from a non-switched system will always return Unknown.
parameters: none
response: <string> NotSet | Receiver | Source , Intern | Thru | Unknown
C#: (property) RFSwitchState
example: :RF:SWIT:STAT? -> NotSet, Unknown

command: **:RF:SWITch:STATe**
syntax: :RF:SWITch:STATe<wsp>REC|SOUR,INT|THRU
description: Setting the RF switches in the LCA testhead. With a non switched LCA system, setting this property has no effect. Trying to set this property to UnKnown, is ignored. Setting this property for a non switched system will stay UnKnown.
parameters: <string> RECeiver | SOURce , INTern, THRU
response: none
C#: (property) RFSwitchState
example: :RF:SWIT:STAT REC,INT

command: **:SOURce[1 – n]:MAXPower?**

syntax: :SOURce[1 – n]:MAXPower?

description: Retrieves the maximum optical output power values in dB. The maximum power for an index n corresponds to the wavelength value from :SOUR:WAV:ALL? at position n.

parameters: none

response: <double> The maximum power value in dBm. For an invalid index it returns -200 and there is an entry in the error queue. See :SYST:ERR?.

C#: (property) MaxPower_dBm

example: :SOUR:MAXP? -> 6

command: **:SOURce[1 – n]:MINPower**?

syntax: :SOURce[1 – n]:MINPower?

description: Retrieves the minimum optical output power value in dBm. The minimum power for an index n corresponds to the wavelength value from :SOUR:WAV:ALL at position n.

parameters: none

response: <double> The minimum power value in dBm. For an invalid index it returns -200 and there is an entry in the error queue. See :SYST:ERR?.

C#: (property) MimPower_dBm

example: :SOUR:MINP? -> -1

command: **:SOURce:POWer?**

syntax: :SOURce:POWer?

description: Gets the current power of the LCA optical output in dBm

parameters: none

response: <double> Laser power value in dBm.

C#: (property) LaserPower_dBm

example: :SOUR:POW? -> 5.00375

command: **:SOURce:POWer**

syntax: :SOURce:POWer<wsp><*power*>

description: Sets the current power of the LCA optical output in dBm

parameters: *power* <double> Laser power value in dBm

response: .none

C#: (property) LaserPower_dBm

example: :SOUR:POW 2.45

command: **:SOURce:STATe?**

syntax: :SOURce:STATe?

description: Gets the current state of the LCA optical output.

parameters: none

response: <string> LaserOn | LaserOff

C#: (property) LaserState

example: :SOUR:STAT? -> LaserOn

command: **:SOURce:STATe**

syntax: :SOURce:STATe<wsp>ON|OFF|1|0

description: Sets the current state of the LCA optical output.

parameters: <string> ON | 1 | OFF | 0

response: none

C#: (property) LaserState

example: :SOUR:STAT ON

command: **:SOURce:WAVelength?**

syntax: :SOURce:WAVelength?

description: Gets the current wavelength of the LCA optical output.

parameters: none

response: <string> The wavelength as string together with the unit.

C#: (property) LaserWavelength_nm

example: :SOUR:WAV? -> <Wvl_1550nm

command:	**:SOURce:WAVelength**
syntax:	:SOURce:WAVelength<wsp><*wavelength*>
description:	Sets the current wavelength of the LCA optical output. The available wavelengths can be retrieved with :SOUR:WAV:ALL?
parameters:	*wavelength* <string> the wavelength value as string, unit is nm e. g. 1550.
response:	none
C#:	(property) LaserWavelength_nm
example:	:SOUR:WAV 1550

command:	**:SOURce:WAVelength:ALL?**
syntax:	:SOURce:WAVelength:ALL?
description:	Retrieves a list showing all available wavelengths of the LCA test head.
parameters:	none
response:	<string> comma separated list of wavelengths units.
C#:	(property) SourceWvl
example: :	SOUR:WAV:ALL? -> Wvl_1310nm, Wvl_1550nm

command:	**:THEAd:INPut:MODe?**
syntax:	:THEAd:INPut:MODe?
description:	Gets the current optical input of the LCA test-head.
parameters:	none
response:	<string> Standard | HighPower
C#:	(property) OpticalInput
example:	THEA:INP:MODE? -> Standard

command:	**:THEAd:INPut:MODe**
syntax:	:THEAd:INPut:MODe<wsp>STAN|HIGH
description:	Sets the current optical input of the LCA test-head.
parameters:	<string> HIGH | STANdard
response:	none
C#:	(property) OpticalInput
example:	THEA:INP:MODE HIGH

command:	**:THEAd:INPut:POWer?**
syntax:	:THEAd:INPut:POWer?
description:	Gets the actual optical DC power, measured by the optical power meter built into the LCA test-head.
parameters:	none
response:	<double> the power value in dBm.
C#:	(property) OpticalDCPower_dBm
example:	:THEA:INP:POW? -> -40.3798

command:	**:THEAd:INPut:NUMBers?**
syntax:	:THEAd:INPut:NUMBers?
description:	Gets the number of optical inputs of the LCA test-head
parameters:	none
response:	<integer> number of optical inputs.
C#:	(property) NumOpticalInputs
example:	:THEA:INP:NUMB? -> 2

command:	**:THEAd:SWITched:ARCHitecture?**
syntax:	:THEAd:SWITched:ARCHitecture?
description:	Gets the LCA test-head architecture. True: LCA test head has switched, False: non-switched architecture.
parameters:	none
response:	<string> 0 = false or 1 = true
C#:	(property) SwtichedArchitecture
example:	:THEA:SWIT:ARCH? -> 0

2

Programming Examples

Further programming examples are installed with the LCA Remote Client in the folder

C:\Program Files (x86)\Agilent\Agilent LCA Remote Client\Examples

The location on your computer depends on the folder in which you installed the LCA Remote Client.

VEE Programming Example

Keysight VEE is a Visual Engineering Environment that allows you to program by creating intuitive “block diagrams.” You select and edit objects from pull-down menus and connect them to each other by wires to specify the program’s flow, mimicking the order of tasks you want to perform.

This makes it easy to get useful results in a short time and in only a few steps.

Getting started

Starting a complete measurement means interacting

- over the .NET interface with the LCA and
- over the COM interface with the network analyzer.

Working with the LCA

If your version of VEE version can use .NET assemblies, we recommend you reference the LCA Remote Client .NET assembly directly, as described here.
If you have an older version of VEE which cannot use .NET assemblies, you need to reference the Active X interface of the LCA Remote Client. This is not described here, but is similar to referencing the PNAProxy, which is described later in this example.

1 After having opened VEE, in the “Device” menu, select “.NET Assembly References”.

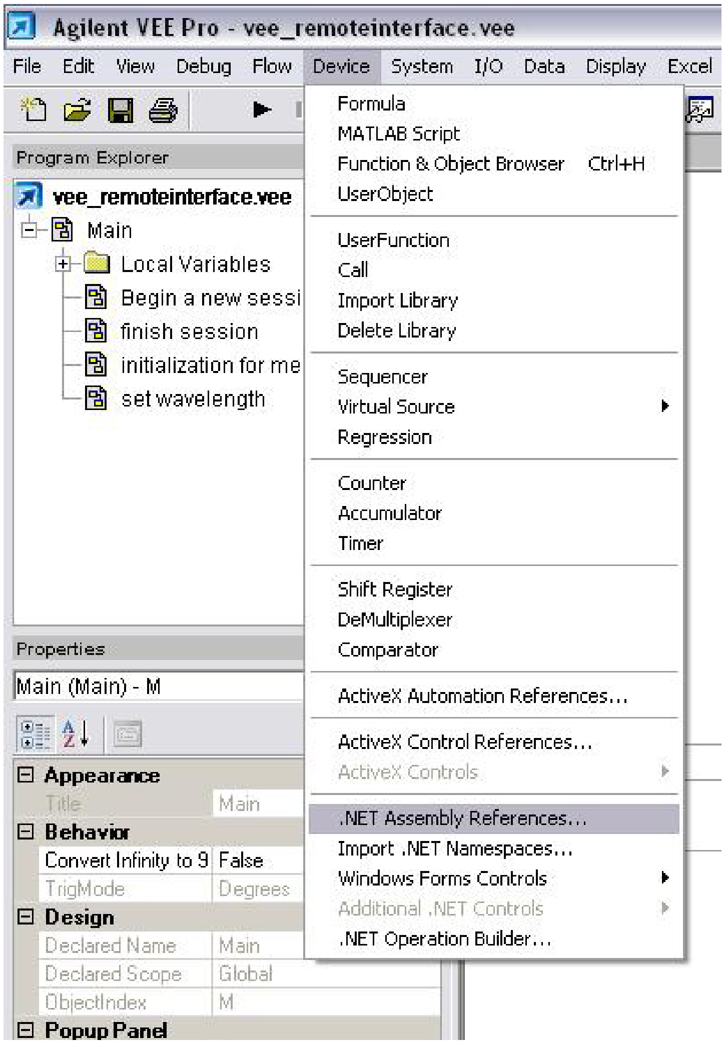
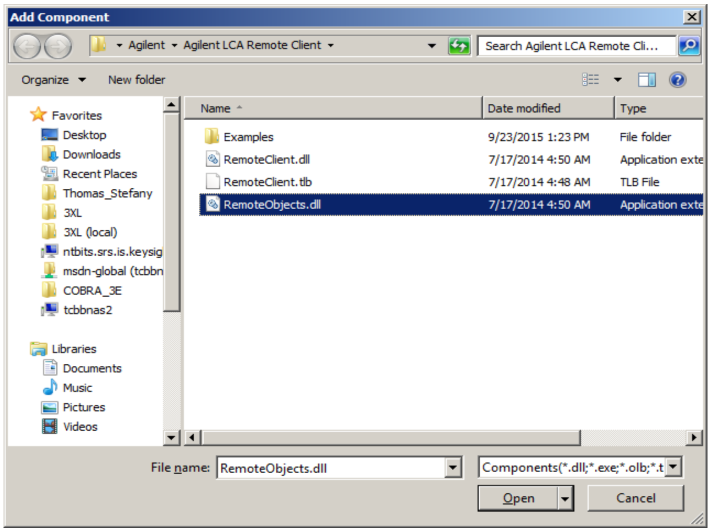


Figure 2 Calling the .NET Assembly references

2 Using the "Browse" button, find the references “RemoteClient.dll” and “RemoteObjects.dll”
These are in the folder:

C:\Program Files (x86)\Agilent\Agilent LCA Remote Client



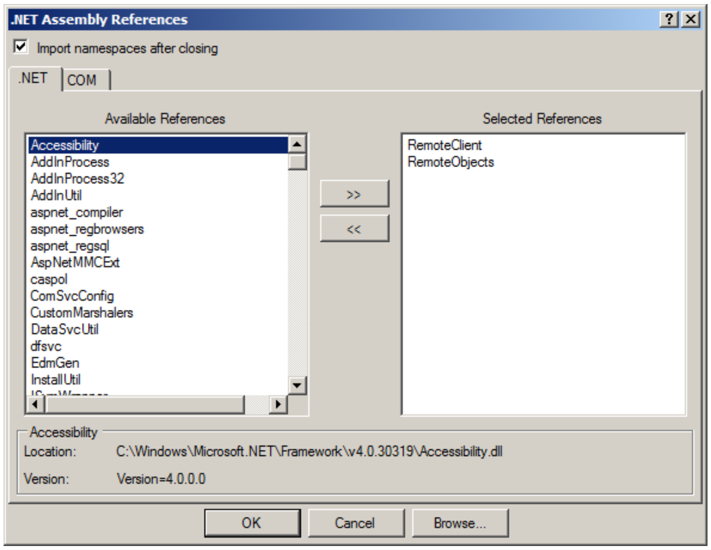
Name	Date modified	Type
Examples	9/23/2015 1:23 PM	File folder
RemoteClient.dll	7/17/2014 4:50 AM	Application exte
RemoteClient.tlb	7/17/2014 4:48 AM	TLB File
RemoteObjects.dll	7/17/2014 4:50 AM	Application exte

File name: RemoteObjects.dll

Components(*.dll;*.exe;*.olb;*.t

3 Enable the flag “Import namespaces after closing”.

4 Ensure that “RemoteClient” and “RemoteObjects” are selected.



☑ Import namespaces after closing

Available References:
Accessibility
AddInProcess
AddInProcess32
AddInUtil
aspnet_compiler
aspnet_regbrowsers
aspnet_regsql
AspNetMMCExt
caspol
ComSvcConfig
CustomMarshalers
DataSvcUtil
dfsvc
EdmGen
InstallUtil

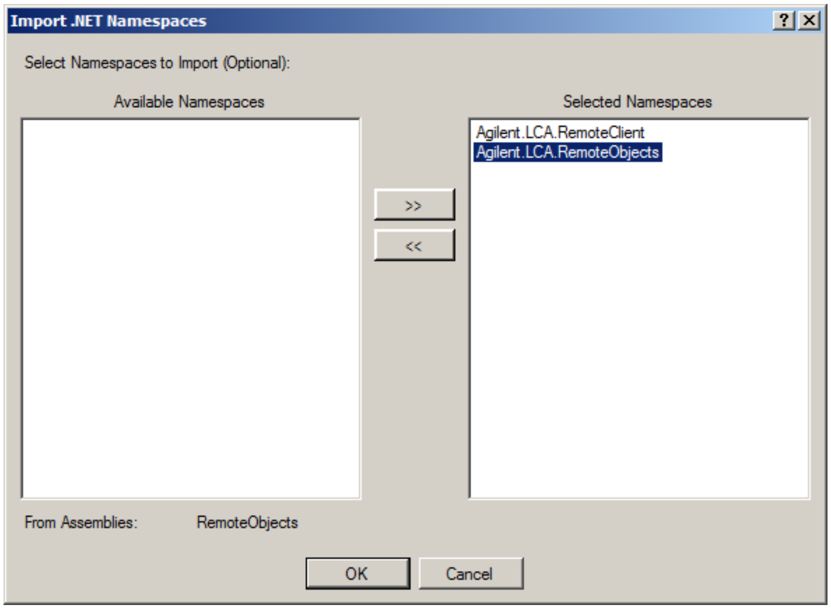
Selected References:
RemoteClient
RemoteObjects

Accessibility
Location: C:\Windows\Microsoft.NET\Framework\v4.0.30319\Accessibility.dll
Version: Version=4.0.0.0

Figure 3 Selecting the required references

5. Set the flag to import the namespaces

“Agilent.LCA.RemoteClient” and click OK



Agilent.LCA.RemoteClient
Agilent.LCA.RemoteObjects

From Assemblies: RemoteObjects

Figure 4 Importing namespaces

You now find the required functions in the “Function & Object Browser”.
You will find this in the “Device” menu.

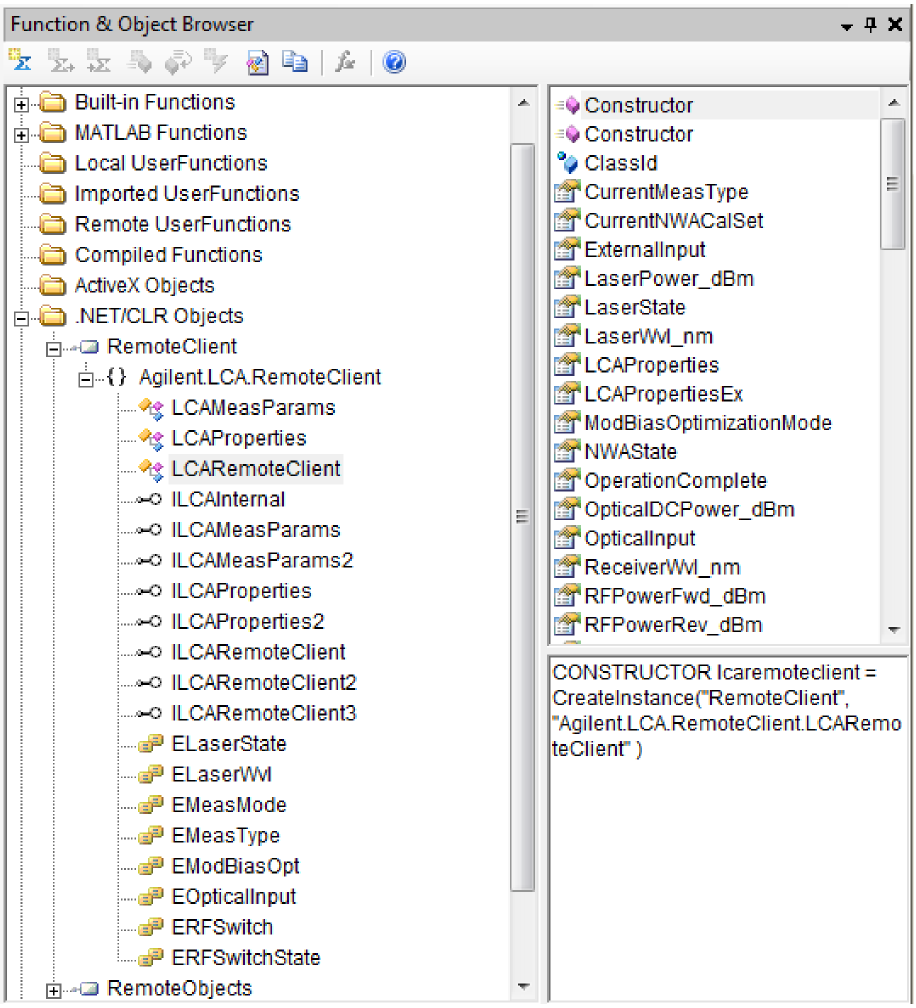
Select

- type: .NET/CLR Objects
- assembly: RemoteClient
- namespace: Agilent.LCA.RemoteClient

to choose the function you want.

NOTE

Before using one of the functions or properties, you have to create an instance of the constructor.



```
CONSTRUCTOR lcaremoteclient =
CreateInstance("RemoteClient",
"Agilent.LCA.RemoteClient.LCARemo
teClient" )
```

Figure 5 Function & Object Browser

Working with the network analyzer

To get your measurement data from the network analyzer, you have to use the COM-interface. To be able to communicate with the network analyzer over its COM interface, you have to install the PNAProxy. For information on installing the PNAProxy see “Install the LCA Remote Client” on page 12.

You can communicate with it using ActiveX references.

1 In the “Device” menu, select “ActiveX Automation References”.

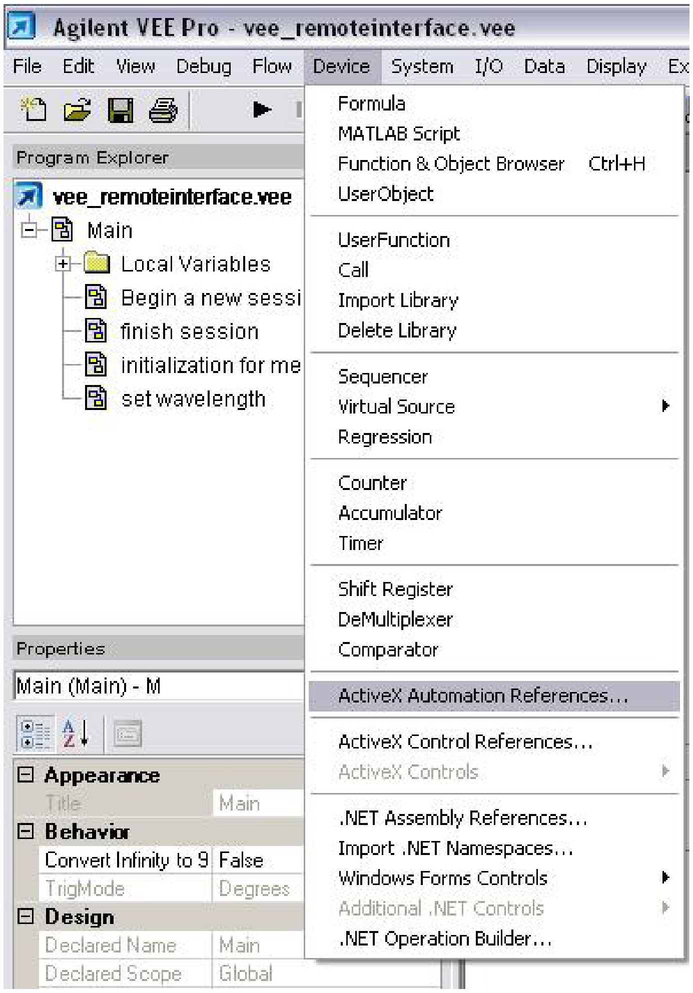
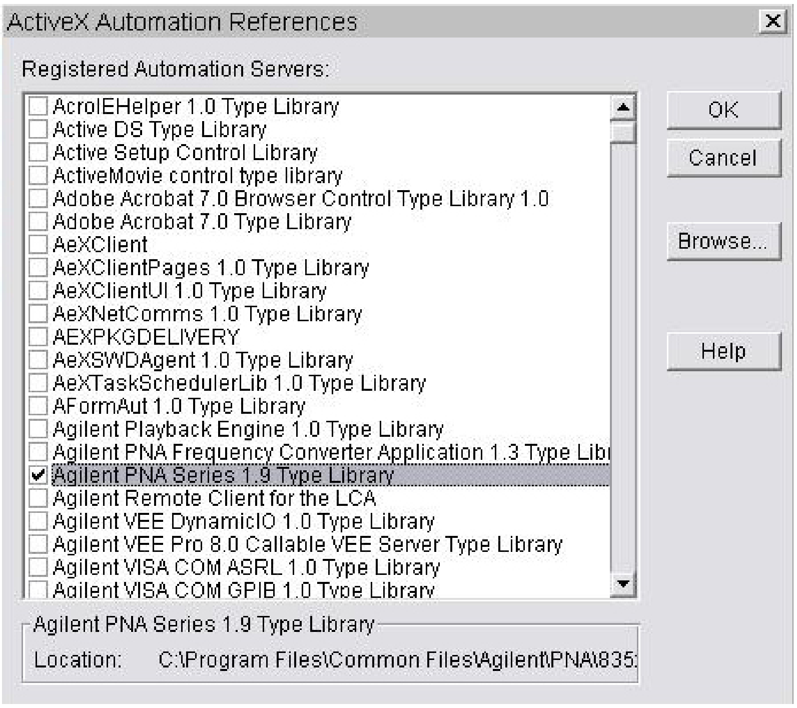


Figure 5 Calling the ActiveX automation references

2 In the dialog, which appears, enable the “Agilent PNA Series 1.9 Type Library” and click OK.



ActiveX Automation References

Registered Automation Servers:

- ☐ AcroIEHelper 1.0 Type Library
- ☐ Active DS Type Library
- ☐ Active Setup Control Library
- ☐ ActiveMovie control type library
- ☐ Adobe Acrobat 7.0 Browser Control Type Library 1.0
- ☐ Adobe Acrobat 7.0 Type Library
- ☐ AeXClient
- ☐ AeXClientPages 1.0 Type Library
- ☐ AeXClientUI 1.0 Type Library
- ☐ AeXNetComms 1.0 Type Library
- ☐ AEXPKGDELIVERY
- ☐ AeXSWDAgent 1.0 Type Library
- ☐ AeXTaskSchedulerLib 1.0 Type Library
- ☐ AFormAut 1.0 Type Library
- ☐ Agilent Playback Engine 1.0 Type Library
- ☐ Agilent PNA Frequency Converter Application 1.3 Type Libr
- ☑ Agilent PNA Series 1.9 Type Library
- ☐ Agilent Remote Client for the LCA
- ☐ Agilent VEE DynamicIO 1.0 Type Library
- ☐ Agilent VEE Pro 8.0 Callable VEE Server Type Library
- ☐ Agilent VISA COM ASRL 1.0 Type Library
- ☐ Agilent VISA COM GPIB 1.0 Type Library

Agilent PNA Series 1.9 Type Library

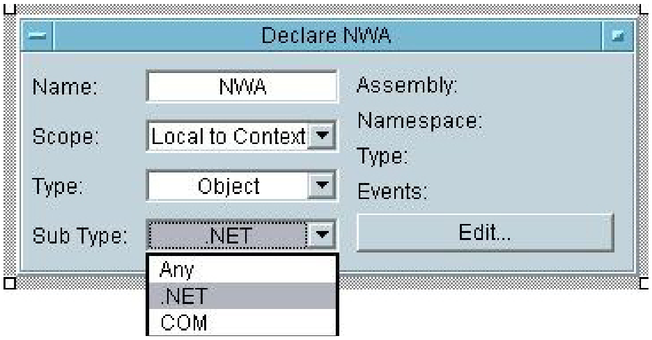
Location: C:\Program Files\Common Files\Agilent\PNA\835

Figure 7 Selecting the required references

Before you can start, you need to declare a variable as object, so Keysight VEE knows you want to create an object.

3 In the “Data” menu, select “Variable”, then “Declare Variable”.

4 As type, select “Object”, and as subtype, select “COM”, which is available in the advanced dialog.



Declare NWA

Name: NWA

Scope: Local to Context

Type: Object

Sub Type: .NET

Assembly:

Namespace:

Type:

Events:

Figure 8 Declaration

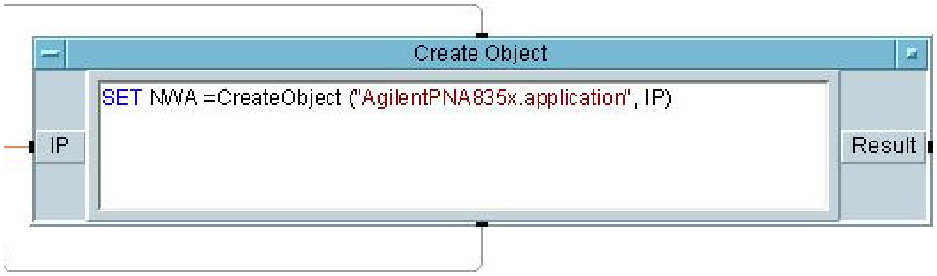
You can set the declared variable in a new formula.

5 In the “Device” menu, select “Formula”.

6 Type the following command into the formula

SET *name* =CreateObject (“AgilentPNA835x.application”, *IP-Address)*

where *name* is the name of the variable you declared, and *IP-address* by the IP address of the LCA.



```
SET NWA =CreateObject ("AgilentPNA835x.application", IP)
```

Figure 9 Setting the variable

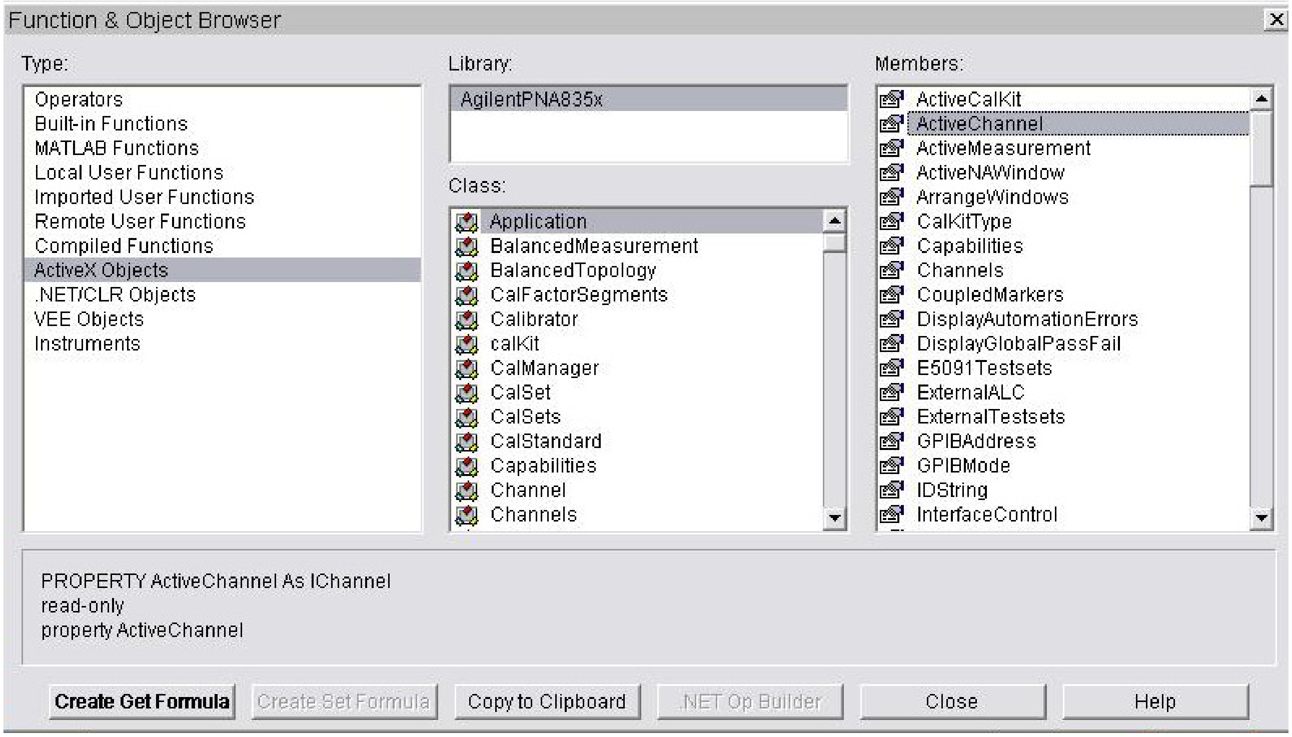
This creates the identifier for object calls “AgilentPNA835x.Application”.

You now find the required functions in the “Function & Object Browser”.
You will find this in the “Device” menu, under Function & Object Browser.

Select

- type: ActiveX Objects
- namespace: AgilentPNA835x to

choose the function you want.



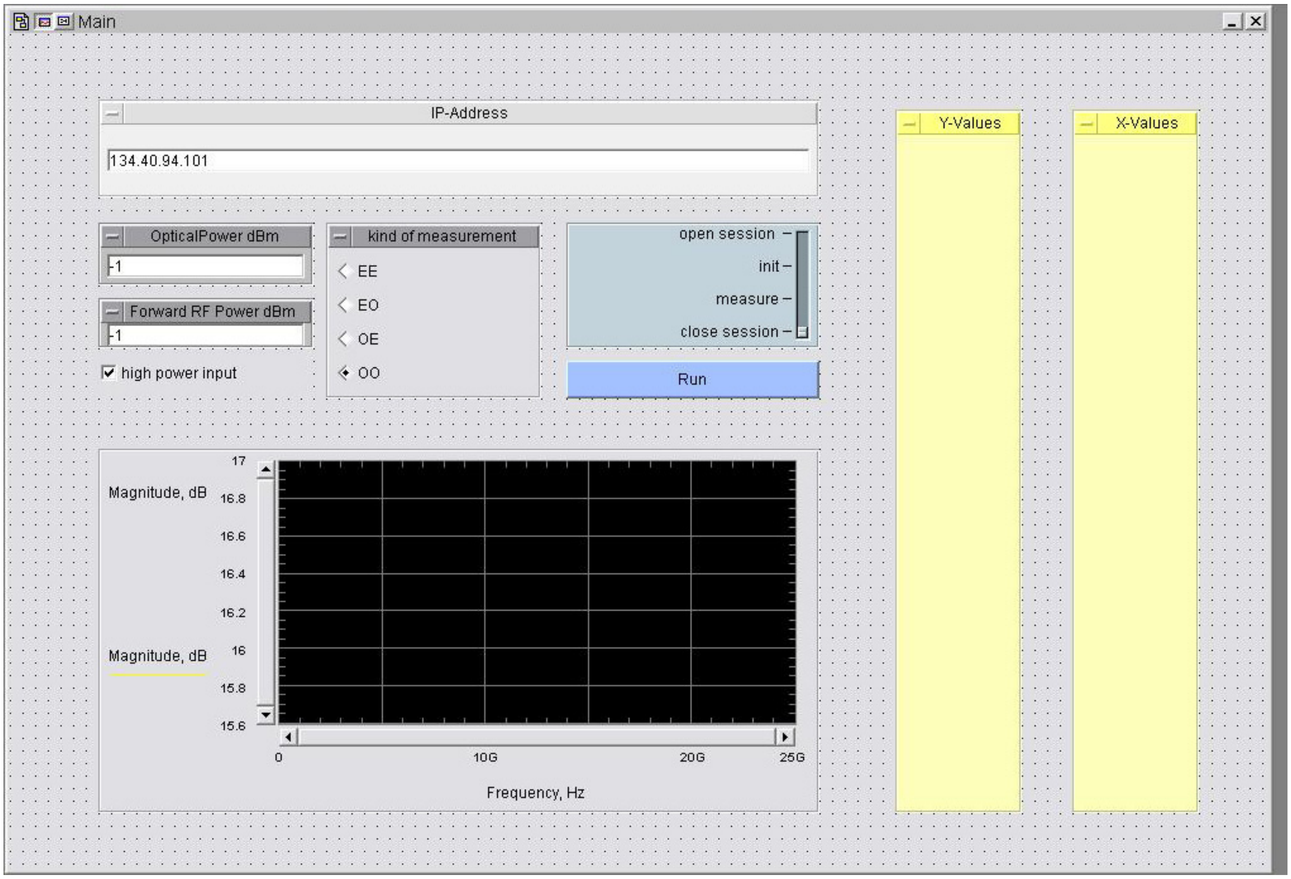
PROPERTY ActiveChannel As IChannel
read-only
property ActiveChannel

Figure 10 Using the Function & Object Browser

Description of the VEE-example

You can find the examples in the “Examples” folder in the “Agilent LCA Remote Client” installation folder. In the VEE-example you can switch between the panel and the detail view.

The panel view lets you set LCA parameters and control the LCA.



IP-Address: 134.40.94.101

OpticalPower dBm: -1

Forward RF Power dBm: -1

☑ high power input

kind of measurement:
○ EE
○ EO
○ OE
● OO

Figure 11 Panel view

In the detail view you can trace the relationships between the different parts of the program:

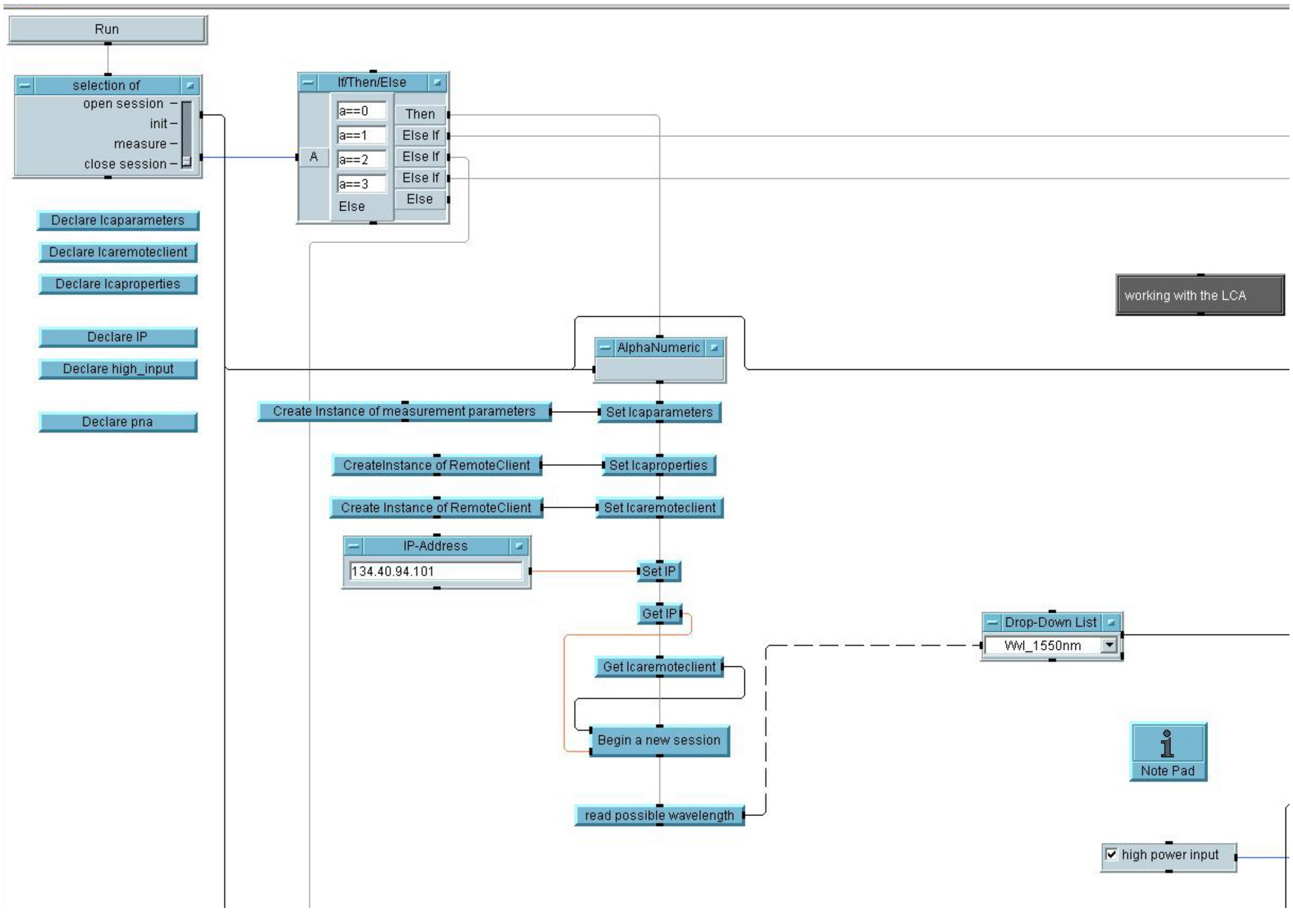


Figure 12 Detail view

NOTE

Starting a measurement always needs an open session. That’s why you have to begin with the point open session of the slider list.

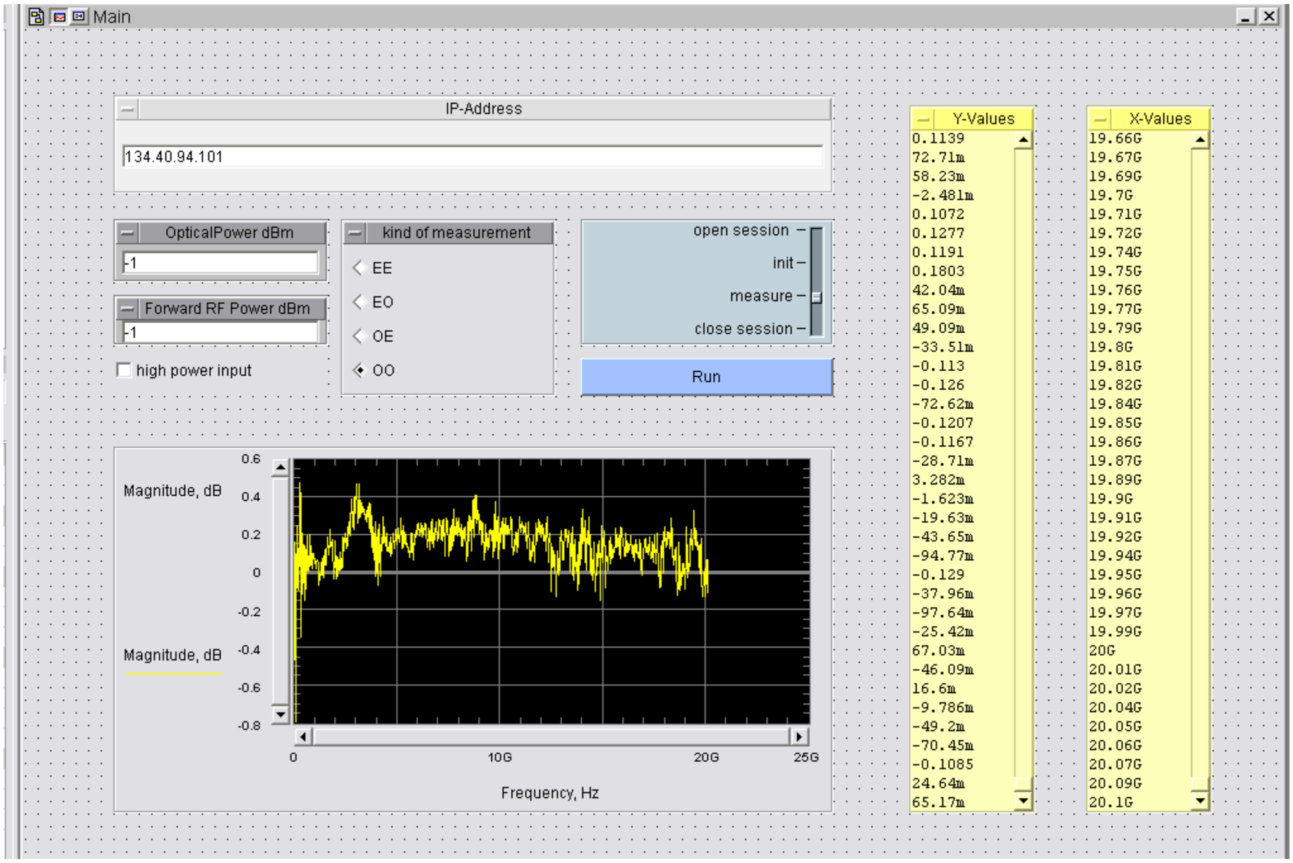
Beginning a new session connects to the server. Before opening a new session, make sure no other session is open that could block the LCA. Then you have the right to control the LCA.

When you select “open a session”, the program sets the instances of LCARemoteClient, LCAMeasParams and LCAProperties. The program then sets the IP address variable and makes it available to the other parts of the program.

The “init”-part reads out the parameters set by the user, and saves them in the properties of the LCAMeasParams. The LCA calls the correct initialization routine for the measurement selected.

The “measure”-part makes a DUT measurement. When the measurement is finished, the program retrieves the results from the network analyzer andplots them as an X-Y plot.

You can read out our data in the panel view too.



IP-Address: 134.40.94.101

OpticalPower dBm: -1

Forward RF Power dBm: -1

□ high power input

kind of measurement:
○ EE
○ EO
○ OE
● OO

Y-Values	X-Values
0.1139	19.66G
72.71m	19.67G
58.23m	19.69G
-2.481m	19.7G
0.1072	19.71G
0.1277	19.72G
0.1191	19.74G
0.1803	19.75G
42.04m	19.76G
65.09m	19.77G
49.09m	19.79G
-33.51m	19.8G
-0.113	19.81G
-0.126	19.82G
-72.62m	19.84G
-0.1207	19.85G
-0.1167	19.86G
-28.71m	19.87G
3.282m	19.89G
-1.623m	19.9G
-19.63m	19.91G
-43.65m	19.92G
-94.77m	19.94G
-0.129	19.95G
-37.96m	19.96G
-97.64m	19.97G
-25.42m	19.99G
67.03m	20G
-46.09m	20.01G
16.6m	20.02G
-9.786m	20.04G
-49.2m	20.05G
-70.45m	20.06G
-0.1085	20.07G
24.64m	20.09G
65.17m	20.1G

Figure 13 Measured data

When you have finished working with the LCA, close the session and disconnect from the LCA. Do this by selecting “close session” and pressing “Run”.

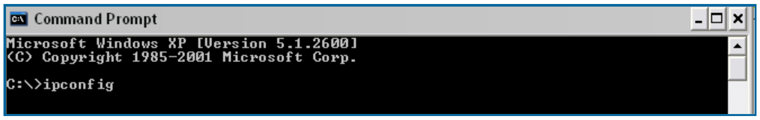
VBA/Excel Programming Example

You can find this programming example in the folder: Examples\Excel VBA\ in your LCA Remote Client installation directory.

Run the VBA/Excel example:

1 To connect to the LCA you need its IP-address

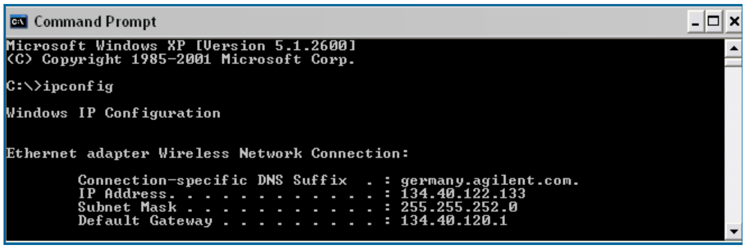
  a On the LCA, from the “Start” menu, select “Run”, then enter “cmd.exe”.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>ipconfig
```

  b Enter “ipconfig” and press Enter.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>ipconfig

Windows IP Configuration


Ethernet adapter Wireless Network Connection:

        Connection-specific DNS Suffix  . : germany.agilent.com.
        IP Address. . . . . . . . . . . . : 134.40.122.133
        Subnet Mask . . . . . . . . . . . : 255.255.252.0
        Default Gateway . . . . . . . . . : 134.40.120.1
```

  Now you can read off the IP address of the LCA.

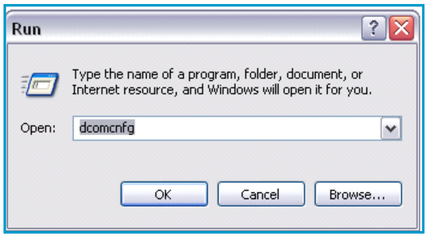
2 Configure the security settings for DCOM.

CAUTION

This procedure sets DCOM security is set to the lowest level. This ensures the application runs.

For information on how to apply a more specific DCOM security setting for different environments we recommend your read the related network analyzer documentation. (Chapter: Configure for COM-DCOM Programming in the network analyzer help file) or the document “dcom.rtf” on the network analyzer support CD.

a From the “Start” menu select “Run”.

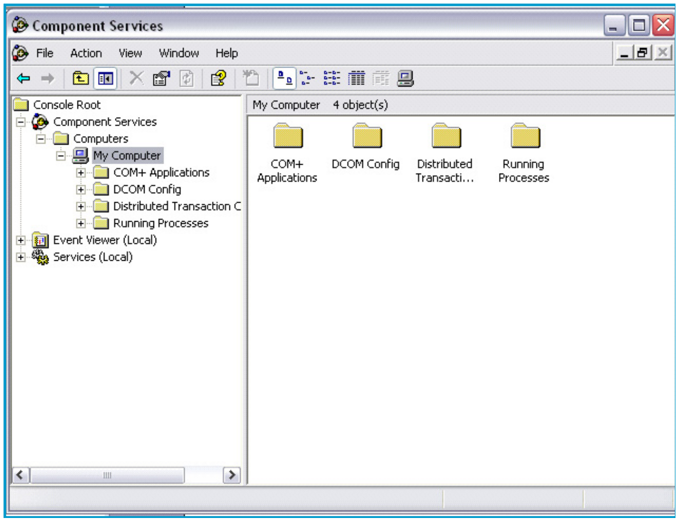


Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: dcomcnfg

b Enter “dcomcnfg” and press ENTER.

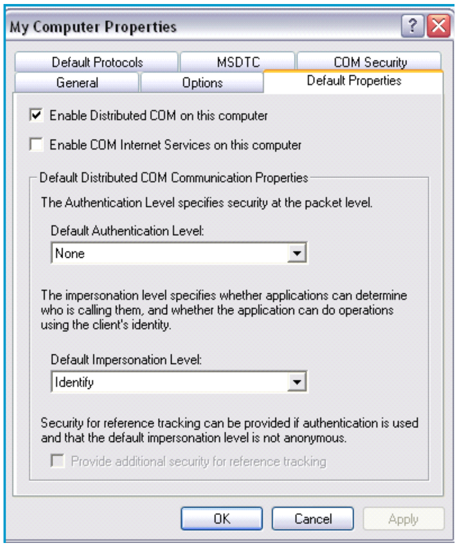
c Expand “Component Services” and “Computers”, and select “My Computer”.



d Right click on “My Computer”, and select “Properties”.

e Select the “Default Properties” tab.

f For the “Default Authentication Level”, choose None.



☑ Enable Distributed COM on this computer

□ Enable COM Internet Services on this computer

Default Distributed COM Communication Properties

The Authentication Level specifies security at the packet level.

Default Authentication Level: None

The impersonation level specifies whether applications can determine who is calling them, and whether the application can do operations using the client's identity.

Default Impersonation Level: Identify

Security for reference tracking can be provided if authentication is used and that the default impersonation level is not anonymous.

□ Provide additional security for reference tracking

g Close with OK.

3 If it is not already installed, install the PNA-Proxy.
For help, see the document “dcom.rtf” on the network analyzer support CD.

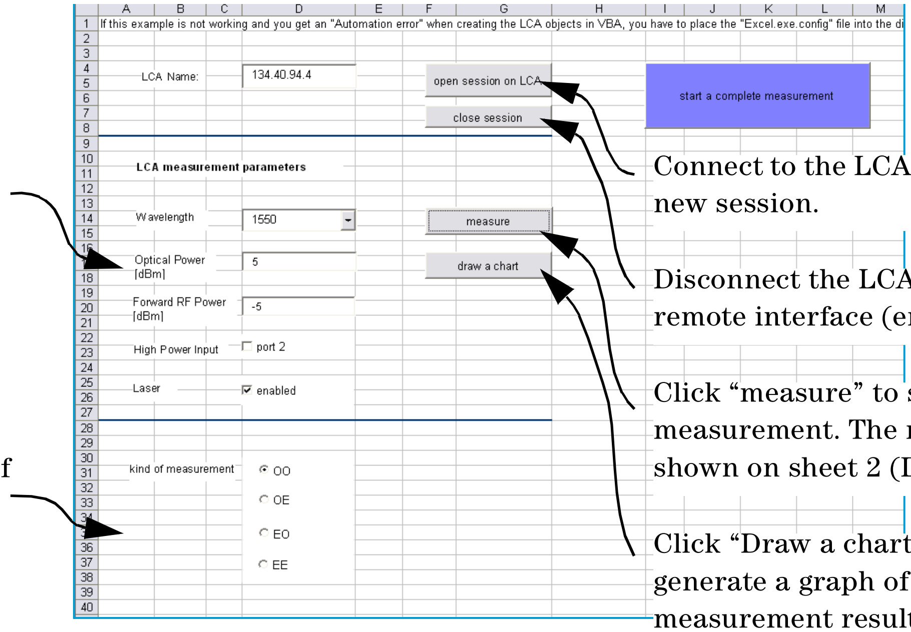
4 Open the Excel workbook “example.xls”.

NOTE

If you encounter COM automation errors when running this example, copy the file “Ecel.exe.config” into the folder where the Excel executable is located. This forces the runtime environment to bind Excel with .NET 2.0 at startup.
If you have an older version of Excel it might be bound to an older version of the .NET framework by default. Since an application can only be bound to one version of the .NET framework you will see errors in this case, because the LCA Remote Client requires .NET 2.0.

This .config file can be found in the same folder as the file “example.xls”.

In the field “LCA Name”, enter the host name or the IP address of the LCA.



If this example is not working and you get an "Automation error" when creating the LCA objects in VBA, you have to place the "Excel.exe.config" file into the di

LCA Name: 134.40.94.4

LCA measurement parameters

Wavelength: 1550

Optical Power [dBm]: 5

Forward RF Power [dBm]: -5

High Power Input: □ port 2

Laser: ☑ enabled

kind of measurement: ● OO ○ OE ○ EO ○ EE

Change the LCA parameters.

Select the type of measurement

Connect to the LCA and start a new session.

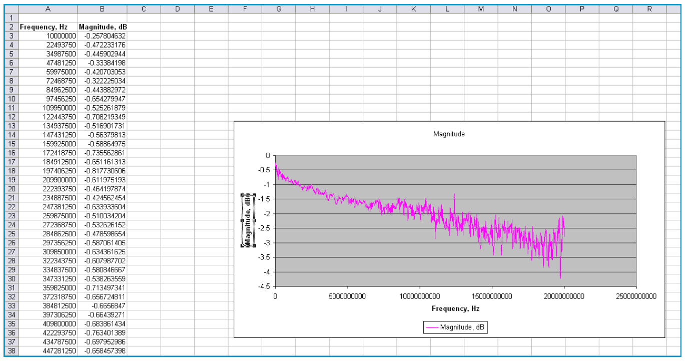
Disconnect the LCA from the remote interface (end session).

Click “measure” to start a measurement. The results are shown on sheet 2 (Data).

Click “Draw a chart” to generate a graph of your measurement results (on sheet 2).

The blue button combines the grey buttons, and runs a full measurement.

Change to sheet 2 to see your measurement results.



Frequency, Hz	Magnitude, dB
10000000	-0.257804632
22493750	-0.472233176
34987500	-0.445902944
47481250	-0.33384198
59975000	-0.420703053
72468750	-0.322225034
84962500	-0.443882972
97456250	-0.654279947
109950000	-0.525261879
122443750	-0.708219349
134937500	-0.516901731
147431250	-0.56379813
159925000	-0.58864975
172418750	-0.735562861
184912500	-0.651161313
197406250	-0.817730606
209900000	-0.611975193
222393750	-0.464197874
234887500	-0.424562454
247381250	-0.633933604
259875000	-0.510034204
272368750	-0.532626152
284862500	-0.478598654
297356250	-0.587061405
309850000	-0.634361625
322343750	-0.607987702
334837500	-0.580846667
347331250	-0.538263559
359825000	-0.713497341
372318750	-0.656724811
384812500	-0.6656847
397306250	-0.66439271
409800000	-0.683861434
422293750	-0.763401389
434787500	-0.697952986
447281250	-0.658457398

3
Warranty Information

Warranty

All system warranties and support agreements are dependent upon the integrity of the Lightwave Component Analyzer. Any modification of the system software or hardware will terminate any obligation that Keysight Technologies may have to the purchaser. Please contact your local Keysight field engineer before embarking in any changes to the system.

System

In addition to the standard warranty, extended warranty periods, on-site troubleshooting, reduced response times and increased coverage hours can be negotiated under a separate support agreement and will be charged at an extra cost.

Remove all doubt

Keysight offers a wide range of additional expert test and measurement services for your equipment, including initial start-up assistance onsite education and training, as well as design, system integration, and project management.

Our repair and calibration services will get your equipment back to you, performing like new, when promised. You will get full value out of your Keysight equipment throughout its lifetime. Your equipment will be serviced by Keysight-trained technicians using the latest factory calibration procedures, automated repair diagnostics and genuine parts. You will always have the utmost confidence in your measurements.

For more information on repair and calibration services, go to

www.keysight.com/find/removealldoubt

Keysight E-mail Updates

Get the latest information on the products and applications you select.

www.keysight.com/find/emailupdates

MyKeysight

Quickly choose and use your test equipment solutions with confidence.

www.keysight.com/find/mykeysight

Keysight Open

Keysight Open simplifies the process of connecting and programming test systems to help engineers design, validate and manufacture electronic products. Keysight offers open connectivity for a broad range of systemready instruments, open industry software, PC-standard I/O and global support, which are combined to more easily integrate test system development.

www.keysight.com/find/open

Phone or Fax

United States: (tel) 800 829 4444

(fax) 800 829 4433

Canada: (tel) 877 894 4414

(fax) 800 746 4866

China: (tel) 800 810 0189

(fax) 800 820 2816

Europe: (tel) 31 20 547 2111

Japan: (tel) (81) 426 56 7832

(fax) (81) 426 56 7840

Korea: (tel) (080) 769 0800

(fax) (080) 769 0900

Latin America: (tel) (305) 269 7500

Taiwan: (tel) 0800 047 866

(fax) 0800 286 331

Other Asia Pacific Countries: (tel) (65) 6375 8100

(fax) (65) 6755 0042

Email: tm_ap@keysight.com

Keysight online information:

Optical test instruments

www.keysight.com/find/oct

Lightwave Component Analyzers

www.keysight.com/find/lca

Polarization solutions

www.keysight.com/find/pol

Spectral analysis products

www.keysight.com/comms/octspectral

Electro-optical converters

www.keysight.com/find/ref

Optical test instruments accessories

www.keysight.com/comms/oct-accessories

Firmware and driver download

www.keysight.com/comms/octfirmware

Keysight photonic discussion forum

www.keysight.com/find/photonic_forum

For Network analyzer related literature, please visit:

Keysight Network Analyzers: www.keysight.com/find/na

Mechanical and Electronic Calibration Kits:
www.keysight.com/find/ecal

RF Test Accessories, Cabinets, Cables:
www.keysight.com/find/accessories

KEYSIGHT
TECHNOLOGIES

This information is subject to change without notice.

Edition 2.0, February 2015

437xB-90A01

www.keysight.com