Keysight N7700 Photonic Application Suite

Lambda Scan Measurement Engine

User's Guide

KEYSIGHT TECHNOLOGIES

Notices

Manual Part Number

N7700-91200

Edition

Edition 6.0, August 2021

Printed in Germany

Keysight Technologies Deutschland GmbH
Herrenberger Strasse 130,
71034 Böblingen, Germany

Technology Licenses

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

U.S. Government Rights

The Software is “commercial computer software,” as defined by Federal Acquisition Regulation (“FAR”) 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement

(“DFARS”) 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at http://www.keysight.com/find/sweula. The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.

Safety Notices

CAUTION

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

WARNING

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

Compliance and Environmental Information

Table 1 Compliance and Environmental Information

Safety Symbol	Description
	This product complies with WEEE Directive (2002/96/EC) marking requirements. The affixed label indicates that you must not discard this electrical/electronic product in domestic household waste. Product Category: With reference to the equipment types in WEEE Directive Annex I, this product is classed as a “Monitoring and Control instrumentation” product. Do not dispose in domestic household waste. To return unwanted products, contact your local Keysight office, or see http://about.keysight.com/en/companyinfo/environment/takeback.shtml for more information.

Contents

Keysight N7700 Photonic Application Suite

LS Engine User's Guide

1 Quick Start Information

This section illustrates how to connect your instruments optically and electrically for typical applications. It is intended as a quick start guide, although it is strongly recommended that you read through Chapter 2, "Getting Started" and Chapter 4, "Automation"
of this User's Guide and through the user's guides of all instruments prior to operating the instruments.

NOTE You need to be logged in as an administrator to install the *Photonic Application Software Suite*.

KEYSIGHT TECHNOLOGIES

Any necessary instrument drivers should be installed on the PC before connecting the instruments, especially via USB. This can be ensured by first installing the indicated packages of the Photonic Application Suite. If you have connected the instrument(s) prior to software installation, you may have to delete the instrument(s) manually from the Windows Device Manager. This does not apply to instruments that are connected via GPIB.

After installation of the Photonic Application Suite, connect the instruments to the local area network or the USB or GPIB. You may use USB hubs to increase the number of USB ports on your PC. Several instruments can also be accessed through your local network. For the connected instruments to be found by Photonic Application Suite engines, they need to be configured in VISA, for example with the Connection Expert which is part of the IO Libraries Suite, if Keysight VISA is primary.

NOTE

M9601A/M9614A/M9615A SMUs are PXIe instruments. For points to be considered when connecting these instruments, refer to Receiver Characterization using M9601A/M9614A/M9615A SMU instruments **on page 29.**

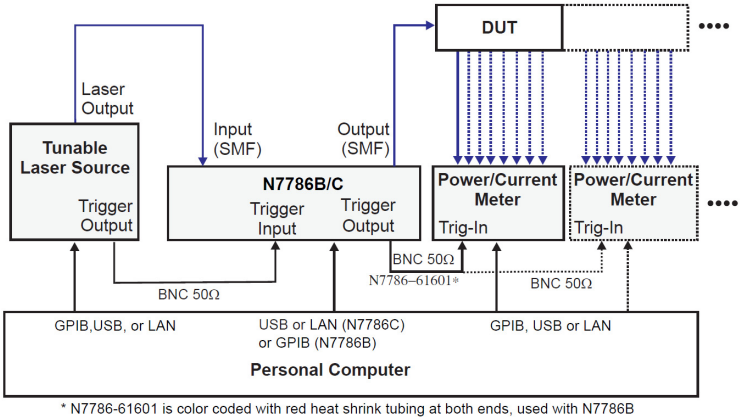
After the instruments have been connected and recognized, start the *Launch Pad* located in the Start menu. Then click on “Lambda Scan”. On first start, the *Configuration Wizard* comes up and searches for all suitable instruments. Follow the instructions given by the *Configuration Wizard*. After finishing the wizard, the application comes up with all configured instruments. If you change your hardware configuration, run the wizard again by clicking on *File* > *Run Configuration Wizard*. For more information, see Running the LS Engine Configuration Wizard on page 38.

Fast Multichannel Lambda Scan Setup

The Lambda Scan (LS) engine can be used in a variety of different hardware configurations. Shown here is a rather common, basic setup, followed by an extended configuration with three tunable laser sources (TLS), an optical switch and return loss module. Similar setups can also be used without a polarization synthesizer. Next, the hardware setup for making measurements using N7788C Optical Component Analyzer in PMD mode is shown. After that, important details on the trigger setup and receiver characterization are explained.

Basic Measurement Setup

The general setup is illustrated in the following diagram:



NOTE **Make sure that you perform the software installation steps described in Software Installation on page 37 before connecting the instruments.**

- Connect the N7786C to the PC either using the USB or the LAN interface. If using N7786B, connect the N7786B to the PC either using the GPIB or the USB interface.
- Connect any power meter or SMU that you want to use with this application to the PC.

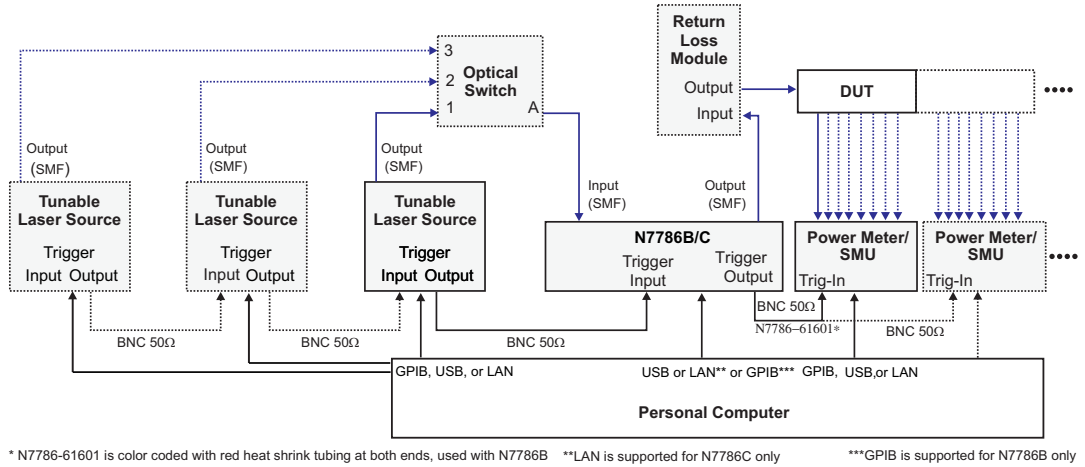
- Put any Lightwave Measurement System (LMS) modules, such as power meters, interface modules, return loss modules, or optical switches that you want to use in the TLS mainframe.
- Connect the TLS mainframe to the PC using the GPIB or LAN interface.
- Connect the Trigger Output of the TLS with the Trigger Input of the N7786B/C (using a 50 Ohm BNC cable).
- Connect the Trigger Output port of the N7786C and the Trigger Input of one of the power meters / SMUs using a 50 Ohm BNC cable.
  If using N7786B, connect the N7786-61601 Trigger Cable to the Expansion port of the N7786B and to the Trigger Input of one of the power / current meters. The N7786-61601 Trigger Cable is marked by red heat-shrink tubing at both ends.
- When using power meters, trigger signals at the trigger input will be routed internally to the trigger output, so daisy-chaining is possible. (using 50 Ohms BNC cables).
- When using the supported B2900-series Precision Source / Measure Units or 81636B Power Sensors, coaxial tee connectors are required to route the trigger signal to all power meters / SMUs. (see Trigger Connections on page 21)
- Connect the optical output of the TLS with the optical input of the N7786B/C using a patchcord with appropriate connectors (angled or straight).
- Connect the optical output of the N7786B/C with the optical input of the device under test (DUT) using a patchcord with appropriate connectors (angled / straight).
- Connect any optical outputs of the DUT to the optical inputs of the power meters/81636B. Connect any electrical outputs of the DUT to the electrical inputs of the N7745A-E01/-E02 or SMU instruments.
- For the reference measurement, the fiber or fibers that will be applied to the DUT input later are connected to optical power meter ports directly first.

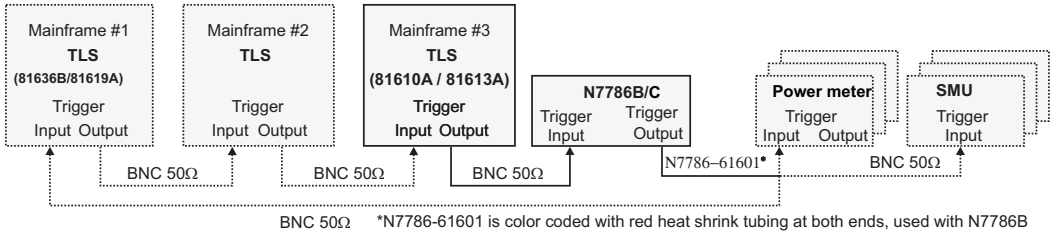
NOTE The fiber connecting TLS and N7786B/C should be carefully positioned and secured for stability without movement or vibrations.

NOTE Optimum performance is obtained when power meters / SMUs (and return loss module, if used) have been zeroed appropriately (see Power Meter / SMU Zeroing on page 138).

Extended Measurement Setup

The extended setup and trigger details are illustrated in the following figure:

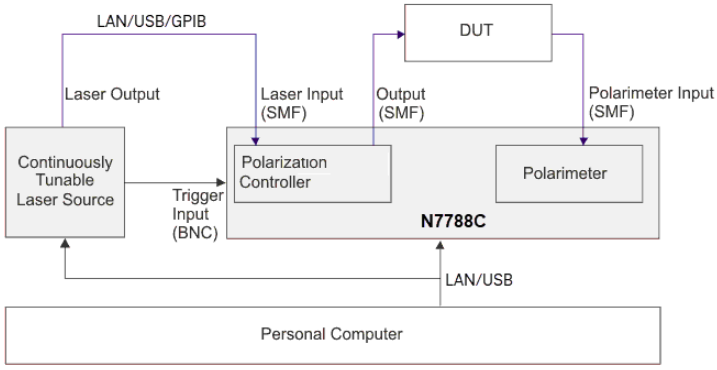




The LS engine may be used with up to three tunable laser sources, covering multiple wavelength bands. If more than one TLS is used, an optical switch is required as well, such as the N7731A or the 81595B module in one of the mainframes. In addition to supporting multiple power meters and Source Measure Units, an optical return loss module 81610A or 81613A can be used.

Measurement Setup for PMD Mode

The following figure shows the hardware setup required to perform the PMD measurements.



The N7788C Optical Component Analyzer combines polarization control functions and polarimetric measurement functions within a single instrument. Therefore, optical connections are reduced to a minimum. A tunable laser source (TLS) is connected to the input and the device under test (DUT) is connected to the corresponding optical ports.

Other than the N7786C, the N7788C comprises an internal optical switch. In addition to every reference and device measurement sweep, this switch is employed to run an internal reference sweep, which makes measurements more robust against environmental changes between the time of taking the reference and performing a measurement. Also, it improves the result when reference measurement has been obtained with different settings than those selected for the device measurement.

The engine also supports combining up to three lasers with an optical switch for PMD mode. However, a return loss module is not supported. Only single-port measurements can be performed, as the N7788C is the only data acquisition instrument capable of measuring the complete state of polarization. That is also the reason why neither power meters nor SMUs are supported in PMD mode.

Trigger Connections

The following points must be kept in mind while configuring the triggering connections for setups with multiple instruments.

- The engine automatically configures all mainframes and laser instruments that are included in the current configuration, assuming they are daisy-chained as shown in the above illustration. If an instrument in the physical trigger path is not included in the current configuration, it will not be configured by the engine and thus, most likely, interrupt the daisy chain.
  When including a Lightwave Measurement System (LMS) mainframe in the configuration that does not contain any module used by the current configuration, it will be configured to pass through triggers from input to output.
- Any instrument or mainframe that is included in a configuration will be preset upon engine activation. In case of LMS modules, the hosting mainframe will be preset, which causes all modules to be preset as well. So, when including a mainframe solely to get the trigger path set to passthrough mode, be aware that the mainframe will be preset too. However, there is one exception to this behavior. In case the only LMS module used from a mainframe is an optical switch, then the mainframe does not need to be added to the configuration automatically and will not be preset, in case it is not included explicitly.
- When other instruments not used by the software are connected to the trigger paths, care should be taken to avoid unexpected behavior.
- Standard RG-58 (50 ohm) BNC cables must be used.
- Use the N7786-61601 trigger cable to connect the N7786B expansion port to the first power/current meter instrument.
- Connect each trigger output of the TLS instruments to the trigger input of the next TLS and the last trigger output to the trigger input of the N7786B/C in case of setups for polarization-resolved measurements or to the power meter(s) and/or SMU(s) otherwise.
- If an 81610A or an 81613A return loss module is used in the setup, it must be placed in the last mainframe of the TLS/mainframe daisy chain, i.e., the one that is connected to the trigger input of the N7786B/C (in case of setups for polarization-resolved measurements or to the power meter(s) and/or SMU(s) otherwise) so that it can receive triggers from any of the TLS. (This is automatically fulfilled if only a single TLS is used in the setup). Alternatively, it can be placed in a separate mainframe with its input trigger port connected parallel to the power meters and SMUs (use BNC-Tee(s)).

- If one or more power meters/optical head interface modules are used in a mainframe with one of the TLS, they must be placed in the first mainframe of the TLS/mainframe daisy chain (This is automatically fulfilled if only a single mainframe is used in the setup). The input trigger of this mainframe should be connected to the trigger output of the N7786B/C (in case of setups for polarization-resolved measurements or to the power meter(s) and/or SMU(s) otherwise). Alternatively, it can be placed in a separate mainframe with its input trigger port connected parallel to the power meters and SMUs (use BNC-Tee(s)).
- All data acquisition instruments should receive input triggers from the N7786B/C (in case of setups for polarization-resolved measurements). Power meter instruments that support this functionality (N7744A/C, N7745A/C, N7747A/C, N7748A/C, N7742C, N7743C, or N7749C) will be configured by the software to pass triggers from the input to the output, so they can be series-connected in a daisy chain, like the TLS instruments.
- If more than one power meter/SMU is used in the setup that is not an N7744A/C, N7745A/C, N7747A/C, N7748A/C, N7742C, N7743C, or N7749C use BNC Tee connectors to split the trigger signal accordingly.
- For measuring photocurrent with the supported B2900-series source/measure units, use the N1294A-031 trigger adapter to connect the trigger signal (i.e., the output of the BNC 50 ohms trigger cable or a respective tee connector) to the Digital I/O port of each SMU. (DIO9). There is currently no trigger passthrough configuration for B2900-series instruments.

Figure 1 N1294A-031 GPIO - BNC Trigger Adapter

- For the M9601A/M9614A/M9615A SMUs, the following accessories are recommended for triggering purposes:

- One M9601-87002 Connector-terminal block 2.5 mm 6-terminal for each M9601A (comes with the M9601A instrument)
  Or
  One M9615-87001 Connector-terminal block 2.5 mm 5-terminal for each M9614A/M615A (comes with the M9614A/M9615A instrument)
- One PX0101A-001 or PX0101A-002 BNC-Ferrule Terminal Cable

One such set is required per SMU and the terminal block is plugged into the front panel of each SMU module. Connect the two wires of the ferrule cable to the top two slots of the terminal block, thus enabling EXT1 trigger, which is used by this software. The Configuration Wizard enables using EXT2 input as well, which uses different slots on the connector block (see SMU User's Guide or front panel indications).

Using the above accessories, each SMU module is triggered directly, and completely independent of the PXIe chassis and whether an embedded controller PC is used or not.

In case there are multiple SMU modules to be used in a chassis, PXI backplane triggering can be used. Both PXI chassis and embedded controller PCs might offer trigger input ports (usually SMB connector type, so either PX0108A-001 or PX0108A-002 SMB to BNC adapter cables are recommended). Refer to PXIe chassis and embedded controller documentation for details on how to map the respective input triggers to the PXI backplane lines. Note that there might be limitations on which chassis slots the backplane lines can be routed from/to simultaneously.

The Configuration Wizard enables using any of the backplane trigger lines (PXI0-PXI7), in addition to the two module trigger inputs (EXT1 and EXT2).

General Details

NOTE

When using 81636B/81619A and/or supported B2900-series instruments, the maximum number of data points will be fewer than when using the N77xxA/C power meters, M9601A, or M9614/5A instruments, hence limiting the wavelength range at high resolutions. As long as such instruments are configured for the current setup, this limitation applies to any instruments that could support a larger number of data points as well. If necessary, the engine will limit the sweep range and inform the user accordingly. Alternatively, the sweep rate can be increased or averaging time setting can be increased. Both will reduce the number of samples acquired, at the cost of a reduced wavelength resolution, but enabling wider sweep ranges.

NOTE

An 81619A can be used with one or two optical power meter heads connected. However, the physical instrument configuration has to be performed before running the Configuration Wizard; when using a single head only, that head must be connected to the top connector interface. Adding or removing heads such that the configuration no longer matches a previously-saved agconfig file is not supported and may result in unexpected behavior. Run Configuration Wizard again, to reflect such a change.

NOTE

If using any of the following, an 81595B optical switch, the return loss module and 81636B power meter or 81619A optical head interface, modules can be placed in the mainframe of one of the tunable laser sources. However, special restrictions regarding the trigger configuration might apply (see Trigger Connections on page 21).

NOTE In case of multiple TLS, connect the optical TLS outputs to ports 1 to 4 of the optical switch. You don't have to connect specific lasers to specific ports of the optical switch. The engine will perform a port auto-detection upon activation. Connect switch port A to the optical input port of the N7786B.

NOTE Refer to Basic Measurement Setup on page 17 for more information on the measurement setup.

Receiver Characterization

For measuring a component with integrated photodetectors, like a receiver optical subassembly (ROSA), the output photocurrent can be detected to determine the dependence of responsivity on the input wavelength and polarization.

Measurements are usually done by connecting the photodiode bias pins of the receiver device to the current measurement instruments (N7745A-E01/-E02/supported B2900-series instruments/M9601A/M9614A/M9615A).

For characterizing receivers with a significant capacitance (e.g. 100nF) in parallel to the bias contacts, as is commonly the case, use an appropriately large averaging time (see Instrument Setup Parameters on page 58). This uses slower sampling to accommodate the lower analog bandwidth of the bias circuit. This is especially important when performing polarization-resolved measurements, as DUT PDL will be underestimated when switching the state of polarization faster than what the response time of the DUT allows.

NOTE When doing receiver characterization, SMU ranges will be shown as dBm-equivalent values. 10/0/-10/-20dBm correspond to 10/1/0.1/0.01mA range.

Receiver Characterization using B2900-series SMU instruments

Figure 2 B29xxA Front View

The following points should be kept in mind when performing receiver characterization using B2900-series SMU instruments:

NOTE **For the list of supported B2900-series SMU instruments, see System Requirements on page 35.**

- Use the center connector pair (Force High (red) and Force Low (black)) of each port of the B2900-series instruments for receiver measurements.
- Positive bias voltages set in the engine configuration will cause a bias voltage with the high potential at the red connector interface.
- Care should be taken regarding electromagnetic interference. Wire loops should be avoided. Best results are achieved with shielded cables. Twisted wires and wires within a single flat ribbon cables usually work fine as well.
- B2902A/B and B2912A/B instruments have a second measurement port at the rear panel.
- The measurement ports of B2900-series instruments are set to floating, i.e., they don't have an internal connection to case ground / protective earth. This allows for maximum flexibility regarding DUT configurations.
- B2900-series instruments are configured to limit measured currents to 7mA.

- At bias voltages with a magnitude of 2V or more, B2900-series instruments will operate the source in a different voltage range and thus add some noise to the measurement results. Best results will be achieved for bias voltages below 2V.

  Using the High Capacitance Mode of the measurement engine yields better results in case of noisy signals.
- Please note that autoranging with the B2900-series SMU will select ranges of 0dBm and above only. More sensitive ranges can be selected manually, but significantly high averaging time settings should be used then, to avoid inherent bandwidth limitations. (see Instrument/DUT Bandwidth and Measurement Trigger Timing on page 47)

Receiver Characterization using M9601A/M9614A/M9615A SMU instruments

Figure 3 M9601A SMU instrument

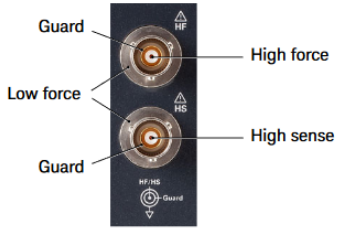
The following points should be kept in mind when performing receiver characterization using M9601A/M9614A/M9615A SMU instrument:

- This software does not support multi-chassis configurations.
- When selecting M9601A/M9614A/M9615A in the Configuration Wizard, ensure that the M9601A/M9614A/M9615A Soft Front Panel (SFP) is not running; otherwise, you won't be able to connect to the module. Similarly, you will not be able to open the M9601A/M9614A/M9615A module's Soft Front Panel while an LS Engine configuration comprising that module is active.
- Since M9601A/M9614A/M9615A is a module in a PXIe chassis, there are only two ways to connect the PXIe mainframe (and, thus, M9601A/M9614A/M9615A module) to the Photonic Application Software software:
  - Using an embedded controller PC module, such as the M9036A/37A, in the mainframe and run Photonic Application Suite and LS engine on that PC

- Using a cable interface module, such as the M9021A to M9024A, in the PXI mainframe and a PCIe card, such as the M9048A/B or M9049A in any given controller PC, then connect the PXIe mainframe to that PC using a PCIe cable.

You might also want to refer to https://www.keysight.com/in/en/assets/7018-04770/brochures/5992-0600.pdf

- There can never be two PCs connected to one mainframe.
- Care should be taken regarding electromagnetic interference. Wire loops should be avoided. Best results are achieved with shielded cables. Twisted wires and wires within a single flat ribbon cables usually work fine as well.
- Use the High-Force connector of the M9601A/M9614A/M9615A for receiver measurements.
- Never connect the Guard terminal (inner shield) to any output, including the frame/chassis ground or any other guard terminal. Doing so will damage the M9601A/M9614A/M9615A.



- Positive bias voltage set in the engine configuration will cause a bias voltage with the high potential at the core terminal of the triax connector interface.
- To perform floating measurements with M9601A/M9614A/M9615A instruments, the "short bar" has to be removed from the M9601A/M9614A/M9615A front panel. Refer to the M9601A/M9614A/M9615A documentation for further details.
- M9601A/M9614A/M615A drivers must be installed before the modules can be used. Download drivers from https://www.keysight.com/main/software.jspx?ckey=3085523&lc=eng&cc=US&nid=-33786.1291408&id=3085523
- Once drivers are installed ensure that the instruments are correctly identified by Connection Expert. Refer to M9601A/M9614A/M9615A documentation in case of any connection/instrument detection issues.

Receiver Characterization using N7745A instruments

A special customization of the N7745A, identified by option E02 replaces 4 of the optical power meter ports with photocurrent input ports through triaxial front-panel connectors.

For connecting the DUT, insulated triax-to-coax adapters (1250-2650) are recommended. The (negative) bias voltage (-4 to 0V) will then be connected to the coax shielding. While the instrument is powered, the center contact will be at N7745A case ground potential.

N7745A instruments measure photocurrent in a single polarity only. The photodiode cathode must be connected to the triax/coax center contact. Only negative biasing is available.

A single N7745A port's bias voltage source can be used for multiple measurement ports. This is required for DUTs with multiple photodiodes with common anodes.

N7745A does not have a controlled current limitation. However, there is an absolute limitation, since their bias voltage sources can only provide a maximum current of 20mA.

Care should be taken regarding induced distortions. Best results are achieved with shielded cables. Twisted wires and wires within a single flat ribbon cables usually work fine as well.

Software User Interface

Figure 4 The *Launch Pad* gives you access to the installed components. Click on "Lambda Scan" to start the LS Engine

NOTE

The LS Engine consists of a *Client Software* which handles the graphical input/output and a *Server Software* which handles the communication with the hardware. For remote control or automation, the controlling program communicates directly with the *Server Software*. The server will be automatically started when you click on "Lambda Scan". The presence of the server is indicated in the task bar:

2 Getting Started

This chapter provides information for setting up your instruments.

About this manual

This manual describes how to use the Photonic Application Suite Lambda Scan application for fast swept-wavelength measurements with a variety of hardware configurations for obtaining different results.

- Configurations using one to three tunable lasers and a number of optical power meter or SMU instruments, and an optional return loss module are enabled with the N7700102C FLS or N7700100C PLS license.
- For polarization-dependent loss measurements, the N7786B/C can be added to these configurations with the N7700100C PLS license.
- For polarization-dependent measurements including differential group delay (DGD) / polarization mode dispersion (PMD), the N7788C can be configured with one to three tunable lasers and the N7700103C PMD license. Please note that these configurations do not support Return Loss measurements.

You might also refer to the *N7700 Keysight Photonic Application Suite User's Guide* to get information on the core Photonic Application Suite functionality including the File Viewer, Plugins, and the general COM API.

System Requirements

- One to three Keysight continuous-sweep tunable laser source (N7776C, N7778C, 81960A, 81940A, 81980A or any TLS models for 8164B Slot 0, e.g., 81600B, 81602A, 81606A, 81607A, or 81608A.)
- One Optical Switch (81595B, N7731A (left switch only), or N7734A) in case more than one tunable laser source is used
- One N7786B/C Polarization Synthesizer (optional)
- One N7788C Optical Component Analyzer (optional)
- One or more power meter/SMU instruments:
  - N7742C
  - N7743C
  - N7744A /N7745A
  - N7744C / N7745C
  - N7745A-E01/-E02
  - N7747A/N7748A
  - N7747C/N7748C
  - 8162xC (81623C, 81624C, 81626C, or 81628C) Optical power meter head connected to N7749C
  - B2901A/B/BL, B2902A/B, B2910BL, B2911A/B, B2912A/B
  - M9601A/M9614A/M9615A
  - 81636B
  - 8162xB Optical power meter head connected to 81619A, N7749C
- Up to one 81610A / 81613A return loss module
- One or more 8163B or 8164B mainframes (in case any Lightwave Measurement System modules are included in the configuration)
- The following patchcords are required:
  - one patchcord per TLS (single-mode or polarization maintaining)
  - one additional (single mode) patchcord in case of a multi-TLS setup
  - one additional (single mode) patchcord in case a return loss module is included in the configuration
  - one additional (single mode) patchcord in case an N7786B/N7786C/N7788C is included in the configuration
  - additional (single mode) patchcords per power meter/photodetector input port that are supposed to measured simultaneously in a single sweep
- One N7786-61601 Trigger Cable (required when using N7786B)

- One N1294A-031 trigger adapter for each B29xxA
- One M9601-87002 Connector-terminal block 2.5 mm 6-terminal for each M9601A (comes with the M9601A instrument)
- One M9615-87001 Connector-terminal block 2.5 mm 5-terminal for each M9614A/M9615A (comes with the M9614A/M9615A instrument)
- One PX0101A-001 or PX0101A-002 BNC-Ferrule Terminal Cable for each M9601A/M9614A/M9615A
- One BNC trigger cable (RG58, 50 Ohms) per instrument (except N7786B) / Lightwave Measurement System mainframe
- LAN, GPIB or USB cables
- Personal Computer:
  - Operating System: Microsoft Windows 10 (64 bit)
  - Depending on the instruments, PC interfaces to LAN, USB, PCIe, and GPIB can be used.
- One or more licenses for the desired measurements: N7700100C, N7700102C, or N7700103C

Software Installation

The LS engine communicates with the instruments on a VISA software layer that should be installed before connecting the instruments. Keysight VISA is provided by installing the IO Libraries Suite, which also provides drivers that may be required by some devices, like GPIB interface adapters and instruments connected via USB. Consult the instrument documentation for establishing connections with VISA. In order to use PXIe modules, their respective drivers need to be downloaded and installed manually. See Receiver Characterization using M9601A/M9614A/M9615A SMU instruments on page 29.

All the other drivers and supplements necessary for using instruments with the LS engine are included in the Photonic Application Suite installation packages. The Package Manager will indicate if any required packages are missing and still need to be installed.

Refer to the *N7700 Photonic Application User's Guide (N7700A-91001)* for information on how to install this software.

Instruments connected via GPIB, PCIe, and often USB will usually be recognized automatically for VISA, for example, by Connection Expert when using Keysight VISA. Instruments connected via LAN might need to be added manually. Note that the N77xxC family of instrument will also be identified as LAN instruments in VISA, when connected via USB, and do not require extra USB drivers. For use with the LS engine, instruments should be connected with the primary VISA if more than one VISA installation is used.

NOTE You need to be logged in as an administrator to install the *Photonic Application Software Suite*.

Running the LS Engine Configuration Wizard

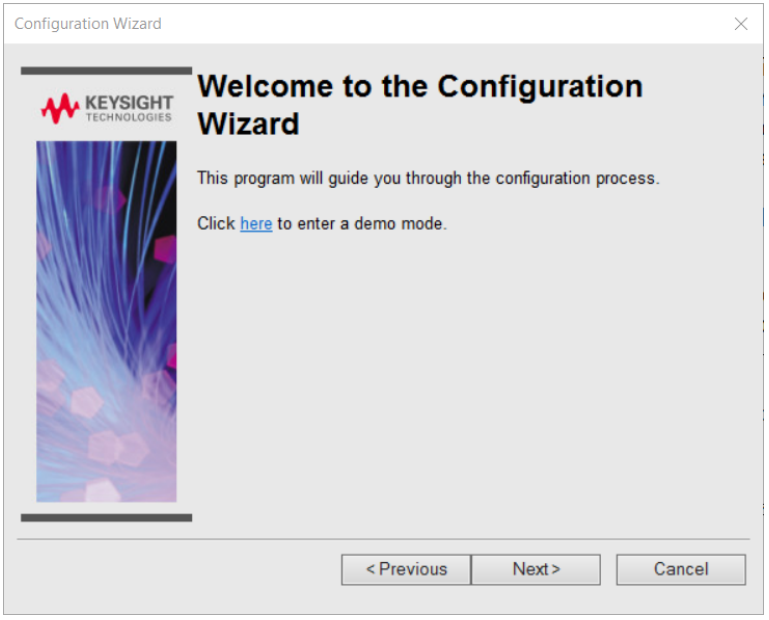
NOTE

If Keysight VISA is used as primary VISA, all instruments need to be identified using the Keysight Connection Expert before the Photonic Application Suite Configuration Wizard is able to add them to the list of available devices. The Connection Expert is part of the Keysight IO Libraries. Start the Connection Expert and click the Refresh All button. When all connected instruments have been identified proceed with the steps below.

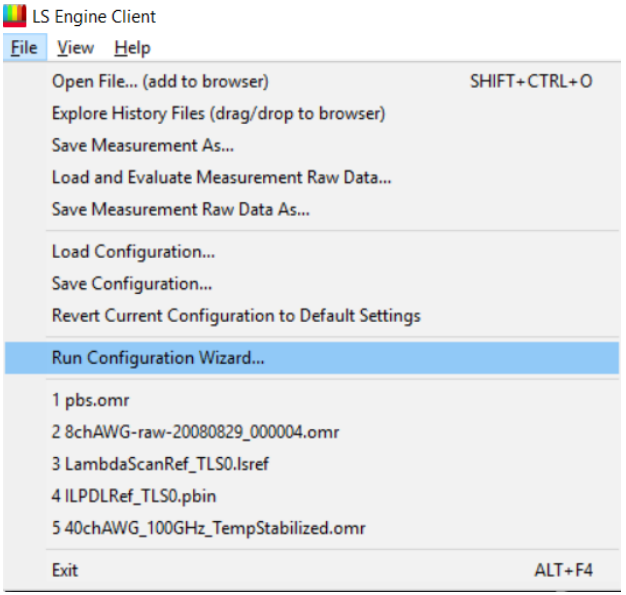
This procedure is only required if a certain supported instrument is connected to the PC for the first time.

Now that the software is installed and the instrument is connected and turned on, you can start the **Lambda Scan Engine** and run the Configuration Wizard.

On first startup the Configuration Wizard will come up automatically.

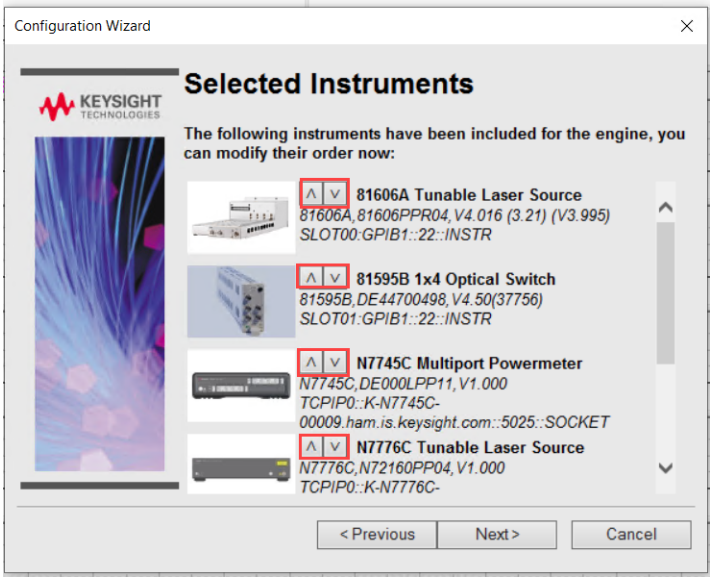


If you want to rerun the Configuration Wizard, for example to use a different set of instruments later on, you can select Run Configuration Wizard from the File menu.



Please follow the instructions shown by the Configuration Wizard. The Configuration Wizard enables you to select interfaces to scan for connected instruments. Usually it is recommended to scan all instruments, but scan time can be improved and potential interruption of other users (in case of remotely shared USB or GPIB instruments) can be avoided by excluding interfaces that are not involved in the target configuration. Please note that N77xxC instruments connected via USB require the LAN or TCP/IP interface to be scanned instead of the USB interface.

After you have selected the instruments, you will see a list of selected instruments, to double-check your choices, which can be helpful in systems with lots of detected instruments. Furthermore, you can reorder the selected instruments using the up and down arrows in front of the instrument names. This can be a very helpful feature if you prefer a certain sequence of instruments in the engine, e.g., lasers in the order of their covered wavelength bands (O, E, SCL), to match the sequence in which they are connected to the optical switch ports or in case you want to keep a configuration file consistent with former versions, although the instrument is now connected to a different interface or has been replaced (different model code / serial number, all of which might lead to a different instrument sequence, as identified by the Configuration Wizard.)



As described in Lambda Scan Application on page 41, the LS engine supports many different configurations. Based on these configurations, the engine generates some measurement results (such as PDL or DGD) only when certain instruments are included. The LS Engine user interface might look slightly different as well, especially when including an N7788C optical component analyzer. Also, the required license depends upon the selected instruments, as described in System Requirements on page 35.

The Configuration Wizard will list all the found instruments and modules, then enable you to select the instruments that you want to include. The Configuration Wizard applies certain checks, such as, not allowing more than three lasers or one return loss module, complaining about a missing optical switch in case more than one laser is selected or when more than one polarization instrument (N7786B / N7786C / N7788C) is selected.

For certain instruments, the Configuration Wizard might show additional drop-down controls, to enable you to configure instrument-specific settings, e.g., TLS output port for certain TLS or the trigger input to be used for M96xxA SMUs.

Once the selection makes a valid configuration, the required license is displayed on the last screen, based on the instruments/modules included in the configuration. If you have the required license installed, you can click on Finish and the LS engine will start up, loading the new configuration. Only at that time, a license check will be performed.

Keysight N7700 Photonic Application Suite

LS Engine User's Guide

3 Lambda Scan Application

KEYSIGHT TECHNOLOGIES

Starting Lambda Scan Engine

You can start the Lambda Scan Engine using ANY of the following steps:

- From the Photonic Application Suite Launch Pad.
  The Launch Pad can be accessed either by clicking its icon on the Desktop or by locating it from the Start menu. (**Start > Keysight N7700 > Launch Pad**)
  The Launch Pad gives you access to the installed components. Click on "Lambda Scan" to start the Lambda Scan Engine.

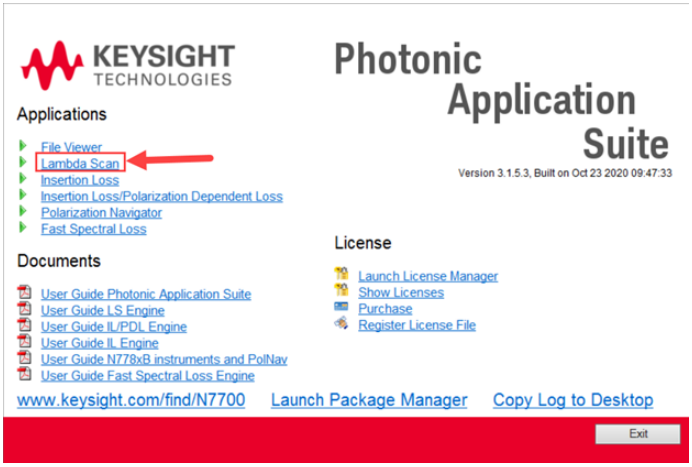


Figure 5 Accessing Lambda Scan Engine from the Launch Pad

- From the Start menu. Click **Start > Keysight N7700 > LS Client**.
- From the Windows Explorer. Navigate to C:\Program Files\Keysight\Photonic Application Suite\bin and double-click **KtPasEngineLS.exe**.
- From the Windows Explorer. Navigate to a previously saved agconfig (LS Engine configuration) file and double-click it.

Application: Lambda Scan

Use the Lambda Scan application for fast wavelength-resolved multi-channel measurements. It uses a continuous sweep of the tunable laser source (TLS) to determine the following parameters versus wavelength, depending on the hardware configuration and software license:

- Polarization Dependent Loss
- Insertion loss (polarization-averaged or at fixed polarization)
- Mueller matrix data
- Polarization-averaged return loss
- Photodiode responsivity (average and max/min)
- Loss or responsivity for TE/TM polarizations
- Photodiode current
- Common mode rejection ratio (DC)
- Polarization extinction ratio (fixed wavelengths)
- Photodiode dark current (TLS output turned off)
- DGD, second-order PMD (denoted PMD2nd)

NOTE **For information comparing the Lambda Scan engine to the N7700 IL/PDL engine, refer to the *Lambda Scan Migration Guide*.**

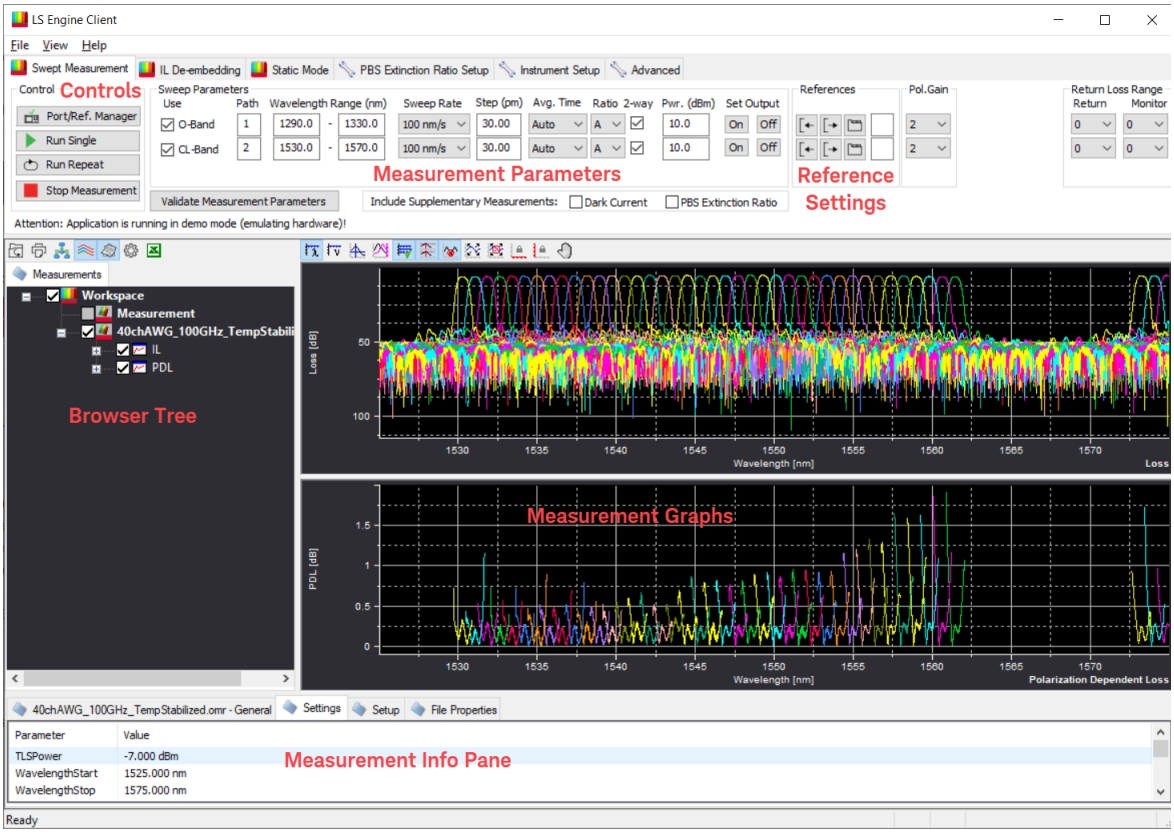
See Basic Measurement Setup on page 17 for details on how to connect the DUT and the trigger cables.

When working with a more complex measurement setup, including optical switches and splitter, for instance, these can be characterized once, then de-embedded from sweep measurement results later on. See IL De-embedding on page 104 for details.

When working with highly polarization-dependent devices, it might be desired to perform additional custom measurement operations, once input polarization has been aligned with either of the DUT's main axes. See Performing Static Mode Measurements on page 113 for details on how to do this.

The PMD Mode measurements are performed using N7788C Optical Component Analyzer instead of N7786B/C Polarization Synthesizer. For more information about the PMD Mode, refer to PMD Mode on page 121.

Before starting a measurement you should make sure that the measurement parameters suit your measurement task.



- **Controls, Measurement Parameters, and Reference Settings**: The measurement control buttons are displayed in the top left portion of the application window, the measurement parameters are shown to the right of these and the reference properties are shown even further to the right. In case the reference description does not fit on your screen, you can see the full line in the tooltip by hovering the mouse pointer over the reference description control. Details of the measurement parameters are given in the tables in the next section.

- **Browser Tree, Measurement Graphs, and Measurement Info Panes:** The browser tree to the left of the application window lists the current measurement and all open files and allows configuring which measurements, which traces (e.g. IL, PDL, TE/TM) and which channels are displayed in the measurement view to the right. The measurement information pane can be toggled by clicking the Show/Hide Info Pane icon above the browser tree.

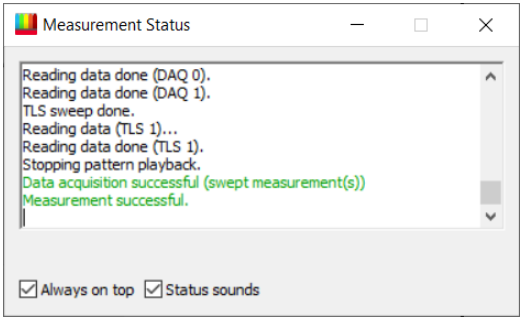
For information on how to configure the Measurement File Viewer area, i.e. the browser tree and the display of the measurement traces, refer to the *Photonic Application Suite User's Guide*.

Measurement Status Window

Information on the current operation progress is displayed in the *Measurement Status* window. Sometimes user inputs are required, e.g. choosing whether to continue or to abort a certain operation. These user input requests are shown in the *Measurement Status* window. There will then be different response buttons at the bottom of the status window.

The Measurement Status window can be resized for optimum screen space and the amount of status information being shown.

Enable the *Status sounds* check box to receive auditory notifications (beeps) when the status of the engine operations change, for example, when a measurement completes or when an error is encountered.



NOTE

If the *Measurement Status* window is closed, it will open automatically, once any user input is required. It can be opened manually by checking *Measurement Status* in the *View* menu.

If the Measurement Status window is not visible at all, it may have been moved to a screen region that is currently not visible. By choosing *Reset Status Window Position* from the *View* menu, the Measurement Status window will be put to the top left corner of the application window.

Instrument/DUT Bandwidth and Measurement Trigger Timing

The LS engine performs a continuously swept wavelength measurement in which the data acquisition instruments sample data based on the input trigger connected to those instruments. The TLS makes a continuous wavelength sweep and generates output trigger pulses at periodic intervals, based on the measurement parameters. The term “actual TLS step size” is used below to indicate the nominal wavelength interval swept between wavelength samples measured by the TLS and making a trigger pulse. Additional intermediate triggers may be configured if the laser supports the oversampling function described below.

As a rule of thumb:

- larger sweep rates at the same target wavelength step cause higher trigger rates
- shorter target wavelength steps at the same sweep rate cause higher trigger rates
- increasing the trigger oversampling ratio setting causes higher trigger rates

Before version 3.2, the LS engine used two distinct timing configurations:

- “Regular” mode: a trigger rate of 20 kHz and 25 µs averaging time
- “High-Capacitance” mode: a trigger rate of 2 kHz and 200 µs averaging time

With this fixed trigger period and the user-defined TLS sweep rate, the resulting wavelength step was determined and could only be modified by changing the TLS sweep rate.

In general, there is a trade-off between a high wavelength resolution (small TLS steps) and low-noise measurements (large TLS steps with longer measurement averaging between trigger pulses) with the sweep rate as an additional parameter, influencing both step size and measurement duration.

Starting with version 3.2, the LS engine allows choosing the desired target wavelength step to allow this with a much finer granularity than the two timing modes provided by earlier engine versions.

The engine will automatically find the required trigger rate to get close to the desired wavelength step and then apply interpolation to get measurement results at exactly the desired target step size. This automatic process takes into account several limitations of any involved instrument, such as maximum TLS trigger rate, minimum TLS step size, or instrument memory size and whether polarization-resolution is turned on or off. The engine will show the relevant resulting settings in the measurement status output.

By default, an appropriate averaging time is chosen automatically that fits within the calculated trigger period unless the averaging time is set manually (see the description of Avg. Time parameter in Application Setup Parameters on page 54). Note that the averaging time is selected from a predefined list of values that is supported by a majority of the supported power meter and SMU instruments.

Alternatively, the averaging time can be selected manually. If the chosen averaging time is smaller than the available period that results from above computations, there is usually no benefit, as an additional portion of the period is left unused. Setting the averaging time to a value that exceeds the period computed from desired step size and sweep rate, to further reduce measurement noise, will automatically increase the actual TLS step size and, thus, reduce the trigger rate.

In both cases, measurement results will be interpolated to the chosen target wavelength step.

With this enhanced flexibility regarding measurement timing, sampling rates might sometimes exceed power meter or SMU bandwidths. The instruments might be limited by analog or digital filter bandwidths, thus, being unable to follow quick signal transitions and in consequence, might not reflect DUT characteristics correctly. This is especially relevant in case of polarization-resolved measurements and DUTs with significant polarization-dependent characteristics, as the polarization instrument is changing the state of polarization with every sample, thus, causing potentially strong power variations from sample to sample. In case of PDL/PER measurements, this effect usually leads to underestimated PDL/PER results.

The bandwidth limitations might depend both on the selected instrument power/current measurement range as well as on the selected averaging time.

The LS engine will show warnings whenever the current sampling rate is too close to the instrument's current estimated bandwidth. It is highly recommended to choose settings resulting in slower sampling rates then (increased step size, reduced sweep rate, increased averaging time).

LS engine features, such as autoranging and high dynamic range measurements might change the instruments' power/current ranges during the measurement, so it is important to check the measurement status output for any bandwidth-related warnings at the end of the measurement. The actual bandwidth-related warning will be shown during the measurement, but there will be an additional warning at the very end of the measurement, in case at least one instrument port was affected by this during any of the measurement sweeps.

When characterizing photodetectors or receiver sub-assemblies, the DUT can lead to a further reduction of overall measurement bandwidth in case there is a significant capacitance in parallel to the photodetector. This is something the engine does not take into account, so a correspondingly slow sampling rate should be chosen (i.e., slower sweep rate, larger step size or longer averaging time).

For photodetector or receiver characterization, using 200 us or larger averaging times is recommended. This can either be achieved by choosing appropriate sweep rate and step size settings, or by explicitly selecting the corresponding averaging time instead of just using the Auto setting. When running supported B2900-series SMU measurements in ranges below 0 dBm, significantly larger averaging times might be required to avoid bandwidth warnings. To avoid accidental end up with such settings, autoranging will not reduce range settings below 0 dBm for supported B2900-series instruments. By turning off autoranging, lower ranges might still be set manually.

When in doubt, it is usually a good practice to start with measurements with rather large step sizes, then reduce the step size iteratively. As long as this only affects noise characteristics, the faster sampling settings can be considered fine. If the average PDL level or loss characteristics begin to change deterministically when using faster sampling settings, the measurement is obviously adversely affected by instrument or DUT bandwidth limitations and slower sampling settings must be used instead.

For PMD/DGD measurements, a good practice is actually the opposite approach, at least if the DGD of the DUT is completely unknown. The polarization-rotation effect that allows measuring the DGD, scales with both the DGD itself and the actual TLS step size used. It becomes ambiguous if either the wavelength step or DUT DGD is too large. Therefore, initial measurements should be done with a reasonably small step size, which can then be increased, if desired. To avoid long and

repeated measurements, a measurement over a short wavelength range, with a small step size can be performed to get an estimation of the actual DGD (assuming that the DGD is reasonably flat over the full wavelength range of interest), then a measurement over the full wavelength range can be done with the intended parameters and the initial short-range/high-resolution measurement can be referred to for validation purposes.

Once the measurement is started, the LS engine will show an estimate of the maximum DUT DGD that can be unambiguously characterized with the current measurement parameter.

In addition to the aforementioned parameters (sweep rate, step size, averaging time), there is also the trigger oversampling ratio (supported by 81960A and sweep-capable 8160xA and N777xC TLS).

Usually the trigger oversampling factor is set to 1, which means there is one trigger generated per TLS wavelength step. Since lasers in the 8164B mainframes are more limited in wavelength sample memory than the N777xC TLS, if the sweep settings (start/stop/step wavelength and sweep rate) would cause the sample memory to be exceeded, the LS engine will automatically configure capable TLS models to generate one or more intermediate trigger pulses between actually measured TLS wavelength steps and linearly interpolate the wavelength values, if the wavelength sample count would otherwise exceed the instrument's sample memory.

The oversampling ratio can also be used to reduce the amount of data to be transferred from the TLS, which can have a relevant impact on measurement duration, mostly when using non-N777xC TLS. To achieve this, manually setting to values larger than 1 to have the TLS generate additional output trigger pulses will cause the engine to enlarge the actual TLS wavelength steps by the same factor to keep the total number of powermeter samples constant while reducing the amount of measured TLS wavelength data (with correspondingly reduced wavelength accuracy).

NOTE

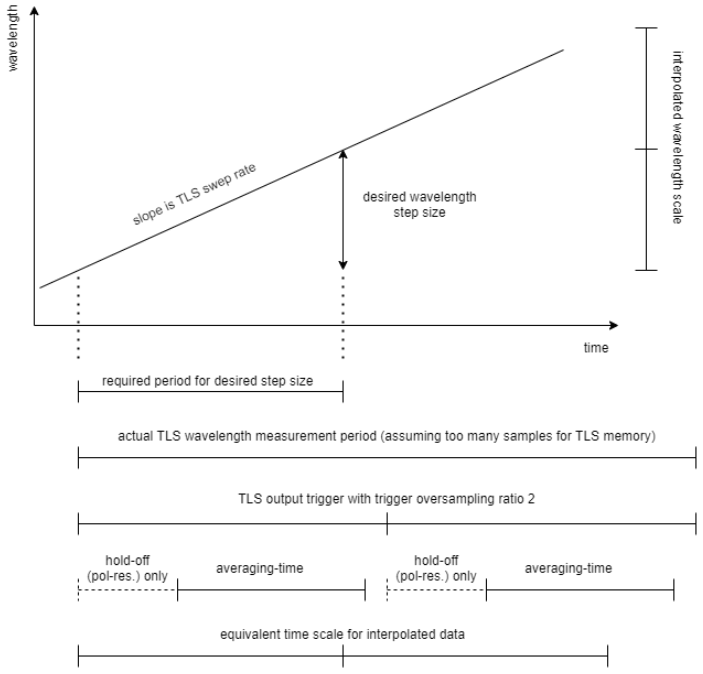
While earlier releases of the LS engine had a strict ratio between selected sweep rate and resulting wavelength step size, this is now decoupled. When keeping the step size value but changing the sweep rate, the measurement results will be returned on the same wavelength grid. Thus, keeping a given step size, then increasing the sweep rate, e.g., by a factor of 10 will keep the number of measurement result samples unchanged, while the same sweep rate setting change would have reduced the sample count by a factor of 10 in the previous engine versions.

With the new settings flexibility, parameters can be chosen for which the desired wavelength step does not match well with the actual TLS step size. The LS engine will show a warning if the actual TLS wavelength step size differs too much from the desired step size chosen in the measurement parameters.

NOTE

For users upgrading from the IL/PDL measurement engine or for the users that find the earlier coupling of sweep rates and step sizes more convenient, the Step Size Compatibility Mode check box under the Advanced tab offers the ability to revert to the older or IL/PDL Engine-compatible settings. In this case, the wavelength step size is defined by sweep rate and averaging time, instead of being configured independently. For more information, see Advanced Parameters on page 61.

The following diagram indicates the relevant timing relations:



Receiver Measurements

The LS engine allows for characterizing wavelength dependent properties of passive optical components as well as O/E receiver devices. For measuring receiver devices, an appropriate instrument for measuring photocurrent is required. Such instruments are Keysight N7745A-E01 or N7745A-E02 specials, as well as supported Keysight B2900-series/M9601A/M9614A/M9615A Source / Measure Units.

Certain aspects have to be considered when using the LS engine for receiver measurements.

Bias Voltages

The LS engine performs measurements at the bias voltage contacts of the receiver device. The most important consideration is the bias voltage polarity. While N7745A-E01 and N7745A-E02 instruments only allow for a negative bias voltage, supported B2900-series/M9601A/M9614A/M9615A

SMU instruments can be used for either bias voltage polarity. For single port devices, this should not mean any limitation if the device can be isolated from electrical ground and as long as one pays attention to the orientation of the DUT. For multiport devices that share a common contact, such as common anode or common cathode configuration, this may require special considerations and using the SMU is probably better. Refer to section on page 19 for further details on configuring the setup and section Instrument Setup Parameters on page 58 for details on bias settings.

High Capacitance Mode

Earlier versions of the LS engine supported the “High Capacitance” mode, primarily intended for characterizing electrical devices. The “High Capacitance” and “Regular” modes have been replaced by a much more flexible measurement timing scheme, as described in Instrument/DUT Bandwidth and Measurement Trigger Timing on page 47.

Since the polarization dependence is measured with 6 samples for each chosen step interval, a sweep rate and desired step size combination that gives 3 ms/step, like 10 nm/s rate and 30 pm step, is equivalent to High Capacitance mode. The averaging time setting can then also be set to the “high capacitance” setting.

Reference Measurements

Receiver measurements require an optical reference measurement, so at least one of the supported optical power sensor instruments is required for reference measurements to measure the optical power that will be applied to the device input during the sweep.

NOTE

The LS engine will use the measured responsivity data to compute the Common Mode Rejection Ratio (DC) by evaluating every pair of neighboring ports of the measurement results.

These don't necessarily have to be physically neighboring ports of the SMU instruments. Refer to Performing a Reference Measurement on page 70 for details on how to obtain and handle reference measurements. After taking a reference measurement on an optical port, you have to copy the reference data to all electrical ports to be used. It is recommended to uncheck the reference data on the optical port then, since the optical port is not connected during the receiver measurement and would show only noise.

For the CMRR results to be valid, ensure that the DUT's balanced photodiode pairs are connected to port pairs according to the reference obtained before.

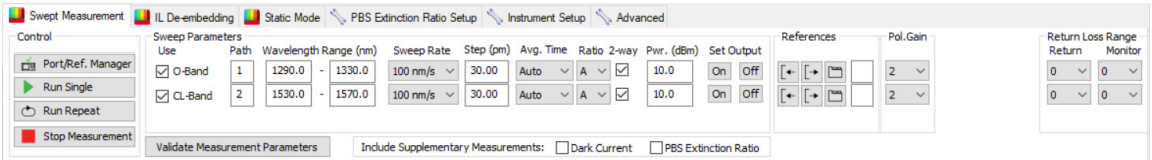
Zeroing

Whether characterizing passive optical components or receiver devices, best performance is acquired if zeroing is performed for the power meter / SMU instruments, once they have settled to operating temperature. For optical measurement ports, zeroing is performed after the detector has been covered. For electrical measurement ports, DUT and connecting cables must be removed prior to starting the zeroing operation.

Application Setup Parameters

The application setup parameters are grouped in separate tabs.

The Measurement Setup parameters are located on the Swept Measurement tab:



The parameters and buttons are explained in the table below.

NOTE

The parameters on the Swept Measurement tab will be different in the PMD Mode. For more information, see Setting the PMD Mode parameters on page 122

Table 2 Application Lambda Scan Measurement Parameters (per TLS)

Parameter	Description
Dark Current	If checked, a dark current measurement will be performed before the actual measurement. This is intended for use with current measurement instruments such as N7745A-E01/-E02 and N7745C-E01/-E02 specials, M9601A, M9614A, M9615A, or supported B2900-series instruments.
PBS Extinction Ratio	If checked, PER measurements will be performed at fixed wavelengths before the actual sweep measurement. Refer to sections “Polarization Extinction Ratio Parameters” on page 64 and “Performing Polarization Extinction Ratio Measurement” on page 98 for further details.
Use	Defines whether the corresponding TLS should be included in subsequent measurement operations.
Switch Path	Enter the number of the switch port that this TLS is connected to (refer to the front panel labels of the optical switch). By default, Switch Path is set to 0 for all TLS. When starting a measurement, the engine will then automatically detect the corresponding switch port for each laser that has path 0 assigned. Alternatively, you can click the **Detect Switch Ports** button on the Instrument Setup tab to automatically detect switch paths for the selected lasers. (see “Instrument Setup Parameters” on page 58)
Start wavelength (nm)	Defines the start of the wavelength sweep.
Stop wavelength (nm)	Defines the stop of the wavelength sweep.

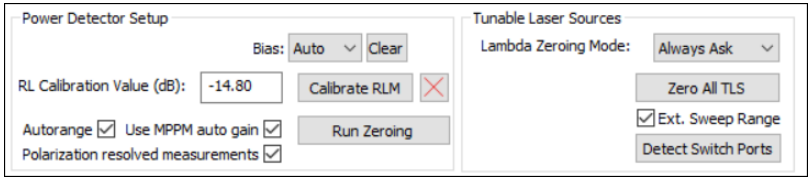
Parameter	Description
Sweep Rate	Defines the sweep rate. See “Instrument/DUT Bandwidth and Measurement Trigger Timing” on page 47 for details on measurement timing.
Step (pm)	Defines the desired step size of the measurement result trace. The entered value also serves as a target for the actual TLS step size. However, the actually used TLS step size might differ largely from the desired step size if required by the given sweep configuration parameters, which will be indicated by a corresponding warning message. See “Instrument/DUT Bandwidth and Measurement Trigger Timing” on page 47 for details on measurement timing.
Avg. Time	Sets the averaging time of the power meter sampling. Higher averaging time will reduce noise on the signals, but might impact features in the transfer characteristics, such as steep filter slopes. The minimum averaging time is 1us (not available for all instruments). If the default option Auto is selected, the measurement engine will choose the largest possible averaging time for the given sweep configuration automatically. See “Instrument/DUT Bandwidth and Measurement Trigger Timing” on page 47 for details on measurement timing.
Ratio	Ratio between the number of triggers generated and number of internally logged wavelength values for upload from TLS. This will be increased automatically in case the sweep settings would lead to a number of samples that exceeds the laser's sampling memory. In addition, it can be used to reduce the amount of wavelength data transferred from the TLS and, thus, increase measurement speed, assuming that the resulting reduction in wavelength accuracy is acceptable (The software will linearly interpolate the wavelength for intermediate triggers). See “Instrument/DUT Bandwidth and Measurement Trigger Timing” on page 47 for details on measurement timing. Requires N777xC, 8160xA, or 81960A laser.

Parameter	Description
Twoway	If checked, measurement data will be obtained during backward scan of the TLS as well, thus increasing the measurement repetition rate. This is only supported with N777xC, 8160xA, or 81960A lasers. Also, configurations involving the N7786B polarization synthesizer do not support bidirectional TLS sweep mode.
TLS Power (dBm)	Defines the optical output power of the laser source. If the specified power is not supported by the TLS, the TLS power will be adjusted to the maximum/minimum power that is supported. Note that this check will be made and the adjustment applied once an actual measurement operation is started, not during any initial measurement parameter checks.
On and Off buttons	These buttons are used to switch on or off the laser output. Clicking “On” for a laser that is already switched on will have no effect on the state of laser output. Similarly, clicking Off for a laser that is already switched off will have no effect.
Pol. Gain	Select the gain setting for the N7786B/C or N7788C polarimeter. When using a non-PMD configuration, sets the N7786B/C polarimeter gain for the specified TLS (device under test and reference measurements). When using a PMD configuration, sets the N7788C polarimeter gain for the internal path measurement for the specified TLS (device under test and reference measurements).
Return Loss Range - Return (dBm)	Sets up the power range of the return loss module (if used). The value defines the maximum allowed power returned to the detector. Larger powers will result in clipped results. In case of low power at the return loss module (e.g. DUTs with high return loss) the return loss module range should be reduced to utilize maximum dynamic range.

Parameter	Description
Return Loss Range - Monitor (dBm)	Sets up the power range of the monitor path in the return loss module (if used). The value defines the maximum allowed power. Larger powers will result in clipped results. In case of low power at the input of the return loss module the range should be reduced to utilize maximum dynamic range.
References	This section contains buttons for reference handling (refer to "Managing References" on page 93 for details). It also shows a comprehensive description of the reference currently selected for the TLS.
Validate Measurement Parameters	Checks measurement parameters for validity and makes adjustments, if required, e.g. wavelength range exceeding the sweep range of the TLS used.

Instrument Setup Parameters

The Instrument Setup parameters are located on the Instrument Setup tab:



For details on Bias, RLM Calibration and Zeroing, see sections Bias Settings on page 144, Return Loss Calibration on page 139, Power Meter / SMU Zeroing on page 138 and TLS Lambda Zeroing on page 139.

A brief description of the instrument setup parameters is shown in the table below.

NOTE The parameters on the Instrument Setup tab will be different in the PMD Mode. For more information, see Setting the PMD Mode parameters on page 122.

Table 3 Application Fast Lambda Scan Instrument Setup Parameters

Parameter	Description
Autorange	If turned on, the power meter / SMU, polarimeter, and return loss module ranges (during regular and reference measurements) are adjusted automatically for each sweep. The range does not change during the sweep.
Ext. Sweep Range	If checked, power meter sampling will start sooner after the TLS sweep start for polarization dependent measurements. This increases the available sweep range, especially when using high sweep speeds, or operating close to the minimum allowed TLS wavelength. Measurement data in the wavelength region extended this way may show slightly reduced performance. The default setting is on.
RL Calibration Value (dB)	This is the return loss value of the reference reflection artifact used in the return loss calibration procedure (refer to “Return Loss Calibration” on page 139 for details).
Lambda Zeroing Mode	There are a number of reasons that may require a lambda zeroing operation. Such an operation takes several ten seconds to complete. To minimize the impact on running measurements, but keeping up the wavelength accuracy, choose one of three settings: **Always Ask** (Default): Whenever a zeroing operation is required, the user will be prompted whether to perform the zeroing operation or not. Once this situation occurs, each subsequent measurement will cause such prompt until either the user allows the zeroing to be performed, or performs the zeroing operation manually from the GUI or changes the **Lambda Zeroing Mode** to either of the other two settings. **Automatically**: Whenever a zeroing operation is required, it is performed automatically right before the next measurement. This is the recommended setting for optimum lambda accuracy. **Manual Only**: Although a zeroing operation may be required, it is neither performed automatically, nor will there be a prompt, requiring a user response. Instead a notification will be shown in the measurement status box. The user will have to take care of the lambda zeroing.

Parameter	Description
Use MPPM auto gain	If checked, the measurement engine will configure any MPPM to use its auto gain feature, which gives improved dynamic range performance. Refer to the *N77xx User's Guide N7744-90B01*, the *N77xx Programming Guide N7744-90C01* and the Application Note "*Transient Optical Power Measurements with the N7744A and N7745A*" The default setting is on. Measurements of devices with high polarization dependence as well as devices with steep transmission characteristics, such as gas cell notches, especially >10dB, can be disrupted by the auto gain switching. A typical adverse effect is that autoranging measurements might not be able to identify an optimum gain and that artifacts like spikes may be caused on filter transmission slopes. Then it should be disabled. **Note:** This setting is effective only in configurations involving N7742/43/44/45 instruments, although this check box is still displayed irrespective of any power/current meter configured in the (non-PMD) setup.
Polarization resolved measurements	This control is available when the current configuration includes an N7786C instrument. Use this check box to toggle between polarization resolved and non-polarization resolved measurements. Switching between the two modes influences the available measurement result trace types and the nominal step size for a given sweep rate. Note that references obtained with the polarization-resolved measurements setting turned off cannot be used to run referenced, polarization-resolved measurements. In the reverse case, references may be used, but will contain some periodic artifacts in the insertion loss traces, so for highest accuracy, matching settings for reference and subsequent device measurement should be used.

Parameter	Description
Detect Switch Ports	Triggers the switch port auto-detection, which requires all the TLS to be connected to the polarization instrument (if any) or to any port of one of the configured power meter / SMU devices (if there is no polarization instrument used). It will only search for those TLS that have their switch path setting set to 0, which is the default value. Manually setting the switch path values for one or more of the TLS will skip the automatic detection for those TLS. This is not a mandatory step, as switch port auto-detection will automatically be performed when starting a measurement and in case any TLS is still set to switch path 0 at that time. Ensure that there is no additional, active light source connected to any switch port that is not used by any of the currently configured TLS. Otherwise, the engine might end up with an incorrect port assignment in case that "parasitic" light source has a higher power output signal than used by the engine for the configured TLS during switch port detection.

Advanced Parameters

The Advanced Measurement and Post-Processing settings are located on the Advanced tab:

NOTE

The parameters on the Advanced Parameters tab will be different in the PMD Mode. For more information, see Setting the PMD Mode parameters on page 122.

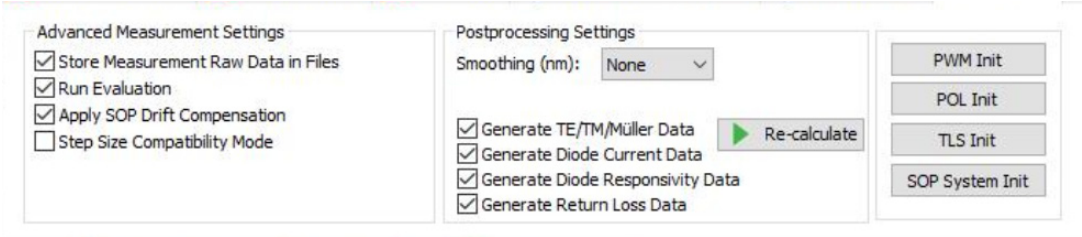


Table 4 Application Lambda Scan Advanced Measurement Settings

Parameter	Description
Store Measurement Raw Data in Files	If checked, measurement raw data is stored in saved files, in addition to evaluated data. File size will be increased. Note that this is just for visual analysis of raw data traces. In order to perform the data processing at a later time, either to change the resolution/smoothing parameter, or in order to separate measurement and data processing from one another, use the File menu item or COM API method for saving measurement raw data.
Run Evaluation	If checked (default), measurement data is evaluated immediately after data acquisition. Disable to run a quick measurement that can aid in verifying setup validity by looking at raw data only or for faster measurement repetitions by saving only the raw data for later data processing.
Apply SOP Drift Compensation	If checked (default for N7786C setups), the data processing for polarization-resolved measurements can considerably reduce the degradation of reference data over time, mostly caused by temperature changes and, thus, can significantly increase the accuracy of measurement results. A reference measurement is required for this functionality. N7786C setups will only benefit from this additional data processing, at the cost of some small overhead in processing time; hence, this check box is enabled, by default. N7786B setups will similarly benefit from the improved lifetime of reference measurement data; however, there is a trade-off due to the lower dynamic range of the N7786B polarimeter, which will cause some additional noise on the processed measurement results. Hence, by default, sop drift compensation will be disabled for N7786B setups.
Step Size Compatibility Mode	If checked, the wavelength step size is defined by sweep rate and averaging time. In this case, the sweep rates and the step sizes are coupled together under the Swept Measurement tab. Convenient for users upgrading from the IL/PDL measurement engine or for the users that prefer the earlier coupling of sweep rates and step sizes over the independent Sweep Rate and Step controls under the Swept Measurement tab.

Table 5 Application Lambda Scan Postprocessing Settings

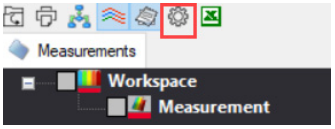
Parameter	Description
Smoothing	Defines the width of the averaging window. Use smaller values for a higher wavelength resolution.
Generate TE/TM/Müller Data	If checked, TE/TM traces and the first row of the corresponding Mueller matrix will be calculated during evaluation and included in the results.
Generate Diode Current Data	If checked, photodiode current traces will be calculated during evaluation. Option is only available if a power meter with electrical inputs, an M9601A, M9614A, M9615A, or a supported B2900-series Source / Measure Unit is used. **Note**: The calculated Diode Current is displayed only when the Detail Level is set to 1 in the Options dialog/Property Editor. (The Options dialog/Property Editor can be accessed by clicking the Options (or the gear) icon on top of the Browser tree on the left.) 
Generate Diode Responsivity Data	If checked, photodiode responsivity traces will be calculated during evaluation. Option is only available if a power meter with electrical inputs, M9601A, M9614A, M9615A, or a supported B2900-series Source / Measure Unit is used.
Generate Return Loss Data	If checked, return loss traces will be calculated during evaluation. Option is only available if a return loss module is included in the current configuration.
Re-calculate	Click this button to re-evaluate the most recently obtained measurement data after changing any postprocessing settings, such as the smoothing parameter. **Important Note**: After loading a raw data file into the engine GUI, clicking this button will re-calculate from the raw data of the loaded file and replace the top item in the OMR measurement tree to the left, i.e., the most recent live measurement results get replaced by those from the loaded raw data file. (For information on loading and recalculating raw data, see “Recalculating Raw Data” on page 126

Table 6 Application Lambda Scan Postprocessing Settings (contd.)

Parameter	Description
PWM Init	When using any direct instrument control to the power meter/SMU instrument(s) in between any two Lambda Scan engine measurements, click this button once before starting the next LS measurement to make sure the instruments are initialized correctly.
POL Init	When using any direct instrument control to the N7786B/C instrument in between any two Lambda Scan engine measurements, click this button once before starting the next LS measurement to make sure the instrument is initialized correctly.
TLS Init	When using any direct instrument control to the laser instrument(s) in between any two Lambda Scan engine measurements, click this button once before starting the next LS measurement to make sure the instruments are initialized correctly.
SOP System Init	Polarization-resolved measurements require specific optimized states of polarization (SOPs) to be generated by the polarization controller. This optimization depends on the wavelength sweep range, on mechanical movement of the fibers up to the polarization instrument, and can depend on temperature changes as well. When taking a new reference, either at first start, after clearing a previous reference, or by replacing all ports in an existing reference, a new optimization is triggered automatically. When performing measurements without a reference, SOPs are optimized on first measurement after activating an engine. By clicking this button, the software will redo the SOP optimization at the beginning of the next measurements. This can be helpful in case the fiber placement or the room temperature has changed significantly.

Polarization Extinction Ratio Parameters

The Polarization Extinction Ratio (PER) settings are located on the PBS Extinction Ratio Setup tab.

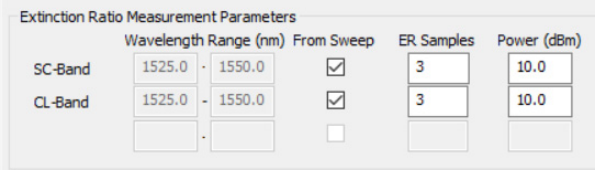


Table 7 Lambda Scan Extinction Ratio Parameters (per TLS)

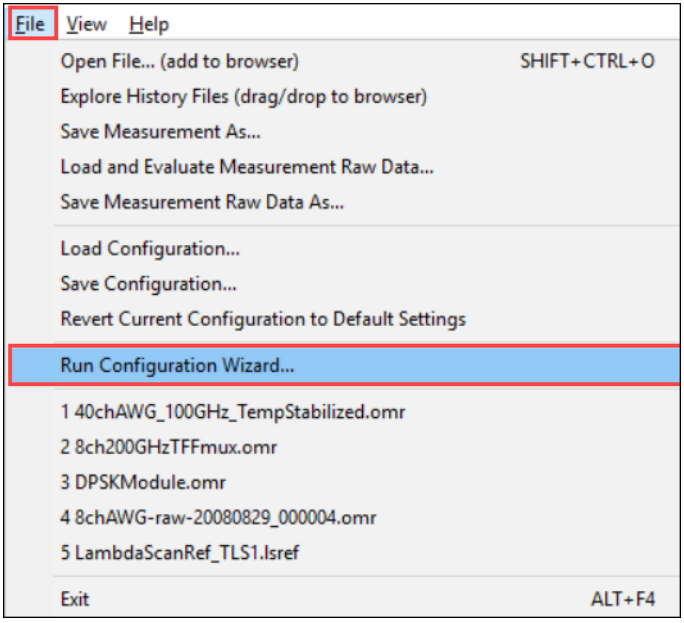
Parameter	Description
Wavelength Range (nm)	Start and Stop Wavelength for (stepped) Polarization Extinction Ratio measurements. While **From Sweep** is checked, these settings cannot be changed.
From Sweep	If checked, start and stop wavelength are taken from the main (sweep) measurement tab. Otherwise the values from the controls on this tab are used.
ER Samples	Number of Extinction Ratio Samples to be obtained. When set to three, one PER measurement will be performed at the start wavelength, one at the stop wavelength and one at the center wavelength. To perform the measurement at just a single wavelength, e.g., the center wavelength, set this field to one. (It is to be noted that even in this case two very closely placed samples will be plotted in the resultant graph and consequently in the resulting OMR data as the LS Engine/File Viewer does not visualize single data samples.)
Power (dBm)	Defines the TLS power to be used for PER measurements.

NOTE Since the Extinction Ratio measurement is not available in the PMD mode, the PBS Extinction Ratio measurement parameters are disabled in the PMD mode.

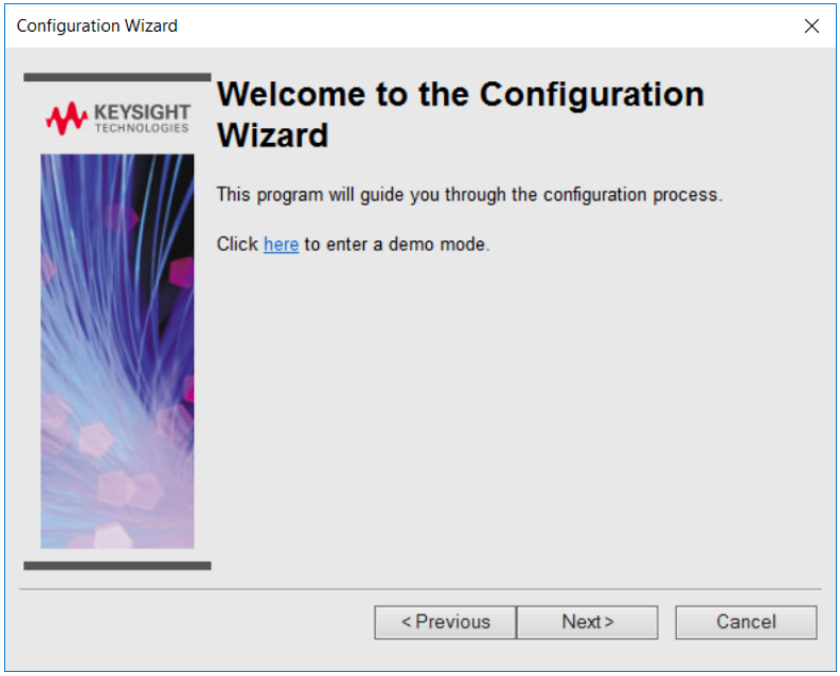
Configuring Instruments Using Configuration Wizard

When you start the Lambda Scan Engine for the first time, the Configuration Wizard is displayed. Otherwise, perform the following steps:

1 Click **File > Run Configuration Wizard**.

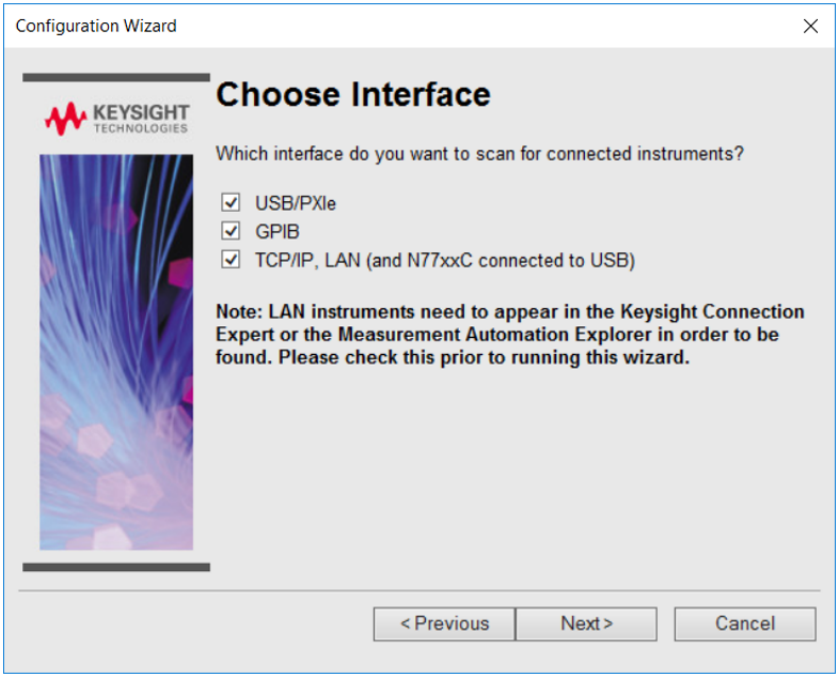


The Configuration Wizard is displayed.



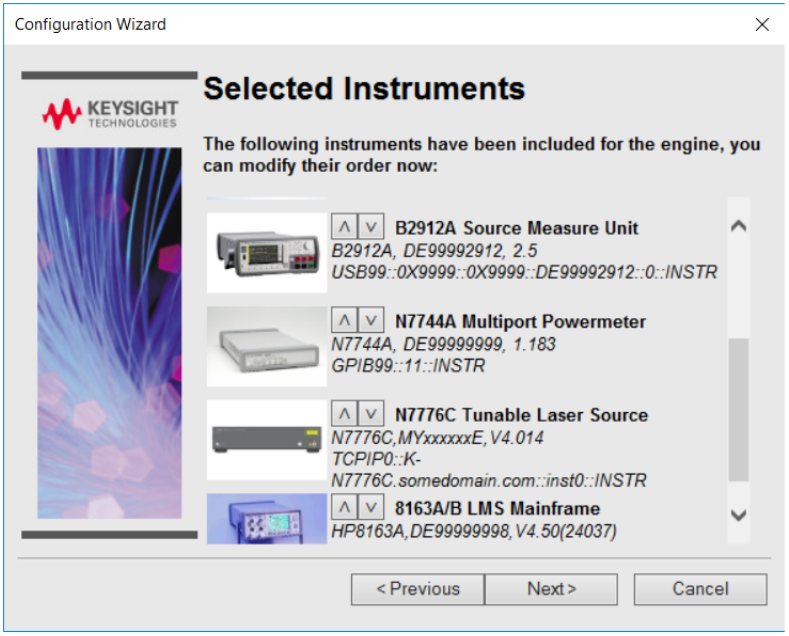
2 To work with the LS Engine in demo mode (without connecting to any instruments), click the **here** link. Otherwise, click **Next**.

3 Select the interface(s) to be scanned for locating and selecting the connected instruments. The available options include USB/PXIe, GPIB, and TCP/IP, LAN.

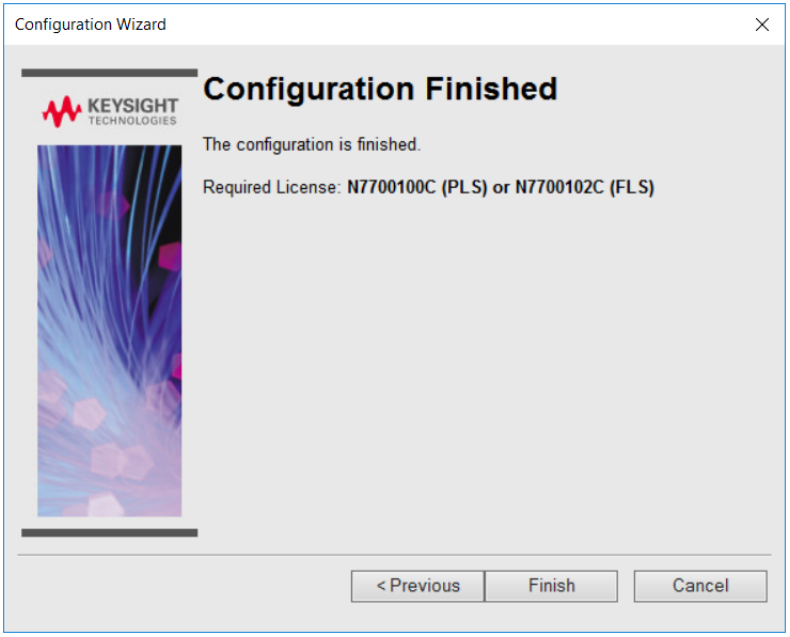


4 Select the instruments required for performing the measurement.

5 Once you have selected the instruments, use the up and down arrow buttons for the instruments to reorder them. You might want to reorder the detector instruments to order the ports in the results or reorder the lasers according to their wavelength ranges.



6 Clicking **Next** displays the Configuration Finished screen with information about the required licenses based on the selected instruments.



7 Click **Finish**.

Performing a Reference Measurement

NOTE

This section explains a typical procedure for a common configuration that comprises a single TLS, an N7786B/C polarization synthesizer and any number of power meter / SMU instruments.

Please note that for setups without polarization synthesizer, with multiple TLS and the corresponding optical switch, or when including a return loss module in the setup, some optical connections described here may involve different instruments instead. Please refer to the setup diagrams in Fast Multichannel Lambda Scan Setup on page 17 to check the optical connections required between instruments and DUT in those cases.

NOTE The information in this section applies to measurements performed in non-PMD mode only.

A reference measurement is performed to make adjustments to the actual swept measurement owing to the environmental conditions or any characteristics inherent in the system or instruments. In the simplest and most common case, a reference measurement measures the signal that will be applied to the input of the device under test.

Before performing a single measurement or a series of measurements with similar settings, you usually take a reference. This is used to determine the optical power that will be applied to the input of the DUT during the TLS sweep, i.e., as a function of the TLS wavelength and the input polarization, as generated by the N7786B/C in case of polarization-resolved measurements.

In the most simple case, the output from the N7786B/C is applied directly to one or more ports of the optical power meters. Select the desired measurement parameters first (see Application Setup Parameters on page 54). Some parameters are adjusted automatically, when starting the actual measurement, e.g. if the chosen wavelength range exceeds the available sweep range of the TLS.

The reference measurement must cover the wavelength range that you are going to use when measuring the device under test. Polarization-resolved measurements require a polarization-resolved reference measurement, while non-polarization-resolved measurements can be performed even with polarization-resolved references, but this might cause measurement artifacts. Maximum accuracy is achieved if the TLS power and sweep rate / nominal step size during reference and DUT sweep are identical.

NOTE Performing a reference measurement is highly recommended before starting the actual DUT measurement or measurement series.

NOTE

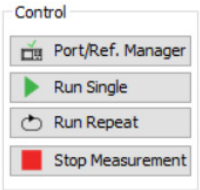
In case of polarization-resolved measurements or when the setup involves a polarization instrument, after performing a reference, the patchcord connecting the TLS with the N7786B/C must not be moved if it is an SMF patchcord. When loading other references files, make sure that the patchcord connecting TLS and N7786B/C has not been moved since taking that other reference.

NOTE

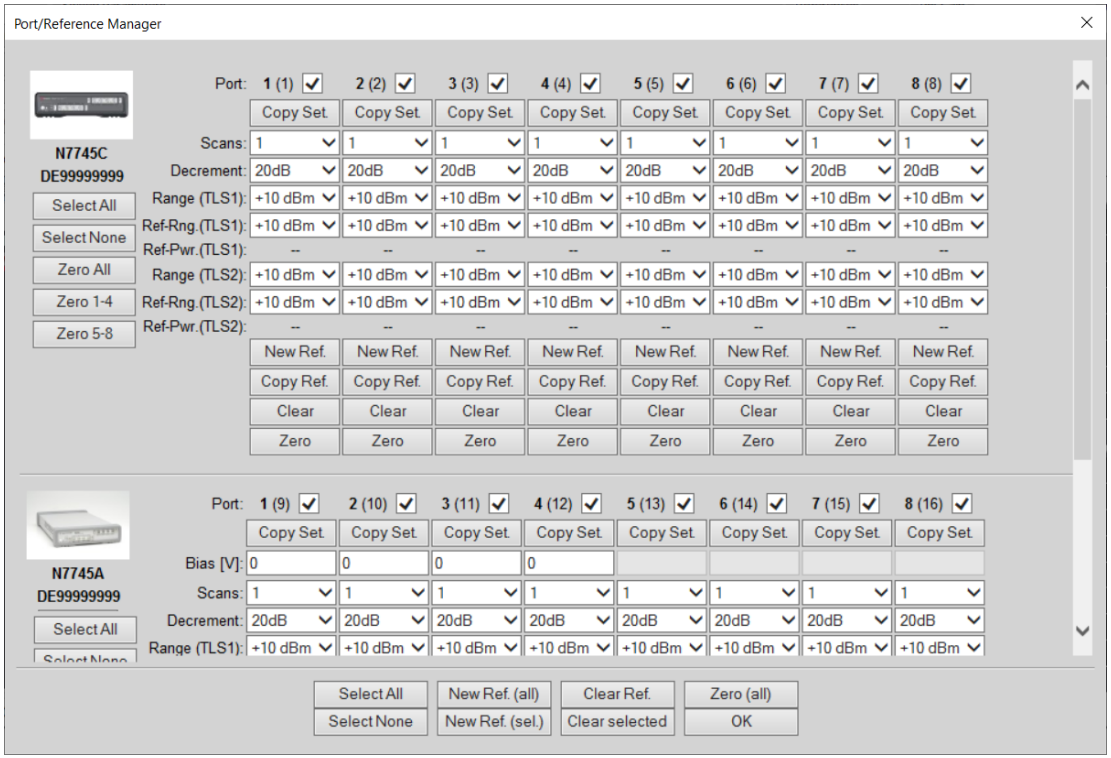
The information in this note applies only when using an N7786B/C and using the polarization-resolved measurement setting (when using N7786C).

When performing a reference measurement after clearing the previously loaded/measured reference or in case the new reference will be replacing all ports of the previous one, the software starts an optimization process for the N7786B/C polarization pattern. This optimization leads to a different pattern each time. Therefore, it is recommended to run a regular measurement right after obtaining that reference as a verification, which is expected to show reasonably flat IL and PDL traces. When adding reference data for additional ports to an existing reference, no such optimization is performed. So usually, in these cases it is sufficient to verify the average power for each port as displayed in the Port/Reference Manager (refer the following figure).

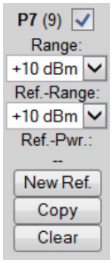
To perform reference operations such as obtaining new reference traces for individual ports, copy reference data from one port to another or clear reference data from certain ports, click Port/Ref. Manager to open the Port/Reference Manager.

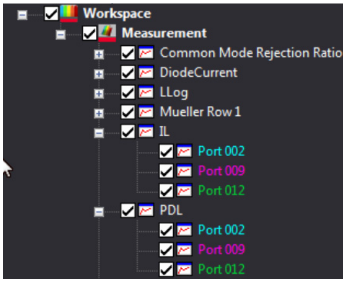


This dialog also displays helpful information about the reference data as described below. When you are finished checking or modifying the reference data, click the OK button on the bottom right to close the Port/Reference Manager.



In the Port/Reference Manager, there will be a list of all ports for all power meter / SMU instruments configured for the application. To the left there is the instrument model code and serial number. To the right are all ports of the powermeter with their port numbers and check boxes (described below). Next to the physical port number of each power meter / SMU port there is a number in brackets. This is the virtual port number of each port, i.e., as if there was just one large array of power meter / SMU ports, no matter which instrument they are physically located in. This is the number shown as port number for measurements in the browser tree:





Beneath the port selection check box of each power meter / SMU port, the bias voltage corresponding to the port is displayed in case of a SMU or a power meter that has electric ports.

Under the bias voltage box for each port, the Power Range and Ref. Range for the port can be viewed and modified. The values define the maximum allowed power to the power meter. Optical powers exceeding the selected range value will result in clipped results and cause the measurements to fail. In case of lower optical power at the power meter (e.g. intermediate instrumentation with significant loss) the range should be reduced accordingly.

NOTE When using SMUs, +10/0/-10/-20dBm correspond to 10/1/0.1/0.01mA ranges for those measurements.

NOTE See Instrument/DUT Bandwidth and Measurement Trigger Timing on page 47 for additional details on range settings and measurement timing.

NOTE

The wavelength sweep settings must be identical for all measurement traces in a reference. If you change the sweep settings after obtaining reference data on some ports, the next reference sweep will display an error, indicating that settings have been adjusted (if possible) or are inconsistent in some way.

In the former case, measurement execution will continue, in the latter, measurement must be aborted by clicking Cancel, then parameters need to be resolved accordingly before retrying.

NOTE

With an N7786B/C instrument and polarization-resolved measurements active, when there is no reference data available (e.g. the engine has just been installed, its wavelength sweep settings have changed or its reference has been cleared), the N7786B/C instrument settings will be optimized with respect to the input state of polarization. As mentioned before, the fiber(s) connecting the TLS and the N7786B/C must not be moved after taking a reference. So if the fiber has been moved, or has been subject to significant environmental changes, the N7786B/C setting optimization may become inaccurate.

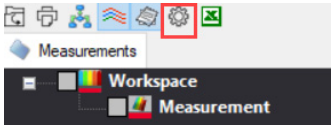
Taking a new reference on only some of the ports will not necessarily cause the N7786B/C to be optimized again. Optimization will be performed only if either the reference was cleared before (**Clear** button) or the reference is obtained using **New Ref. (all)**. Optimization will also be performed if the next reference sweep will replace all currently existing reference traces simultaneously. Additionally, you can click the SOP System Init button on the Advanced tab to force an optimization operation at the start of the next sweep.

Whenever reference traces for at least one port remain unchanged by the next reference sweep, no optimization will be performed. (The intention is that the references on all ports use the same polarization states.)

As an indication you can check the measurement status box during the reference operation. If it is optimizing the N7786B/C settings, it will show “Optimizing Input SOPs” right after “Starting Reference Measurement”.

NOTE

The displayed average power serves as a verification for the reference validity. Depending upon the TLS power selected in the GUI and the insertion loss of the reference connection, you should compare this value to the expected average power level.
After making a reference measurement, the raw data associated with the measurement can also be viewed for verification. To enable raw data trace visualization, enable the **Store Measurement Raw Data in Files** setting on the Advanced tab. Additionally, set the Detail Level to 1 in the Options dialog/Property Editor. (The Options dialog/Property Editor can be accessed by clicking the Options (or the gear) icon on top of the Browser tree on the left.)



NOTE

When using lasers capable of bidirectional sweeping, taking a reference will automatically perform a forward and a backward sweep, to automatically apply the matching reference data to future device measurements.

NOTE

Using reference data on some ports and no reference data on other ports is not supported. Neither is it supported to use polarization-resolved reference data on some ports and non-polarization-resolved reference data on other ports. Whenever you try to do either operation, the engine shows a corresponding error message.

NOTE When taking a new reference (single port, selected ports, all ports), the laser selection from the Swept Measurement tab is taken into consideration. In this way references for single lasers can be updated/replaced individually.

Procedure Summary for Obtaining a Reference

Ensure that the measurement is started in stable temperature environment and that the instruments should have been powered up for 20-30 minutes prior to the measurement.

1 In the Configuration Wizard, select the instruments required for performing the reference measurement. You require:

a One or more tunable laser sources

b Optional: A polarization synthesizer

c One or more power meters

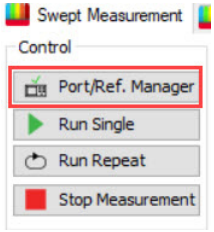
Refer to the Configuring Instruments Using Configuration Wizard on page 66 for instructions on selecting instruments using the Configuration Wizard.

2 Under the Swept Measurement tab, make appropriate settings, esp., the following:

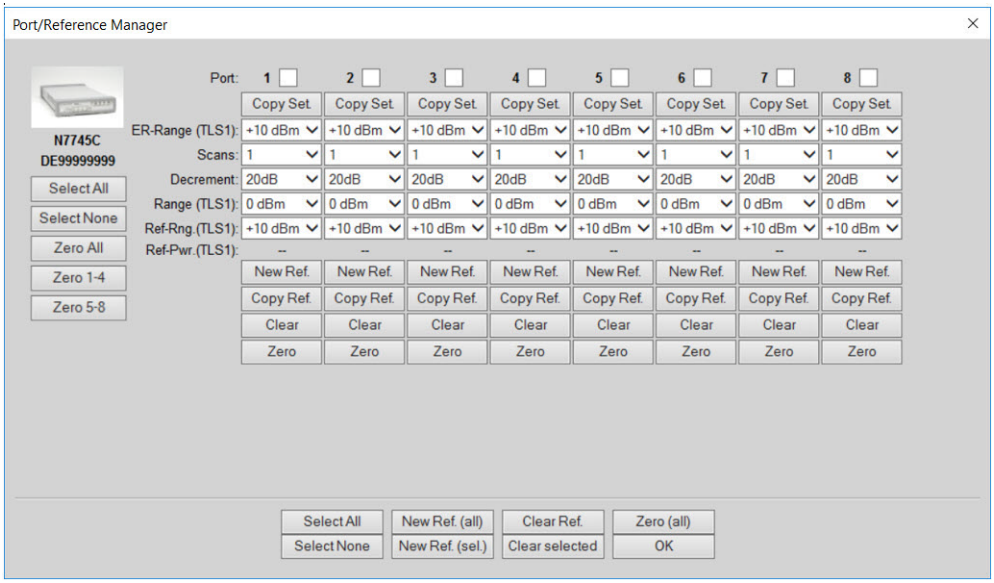
a Sweep Rate

b Pwr.

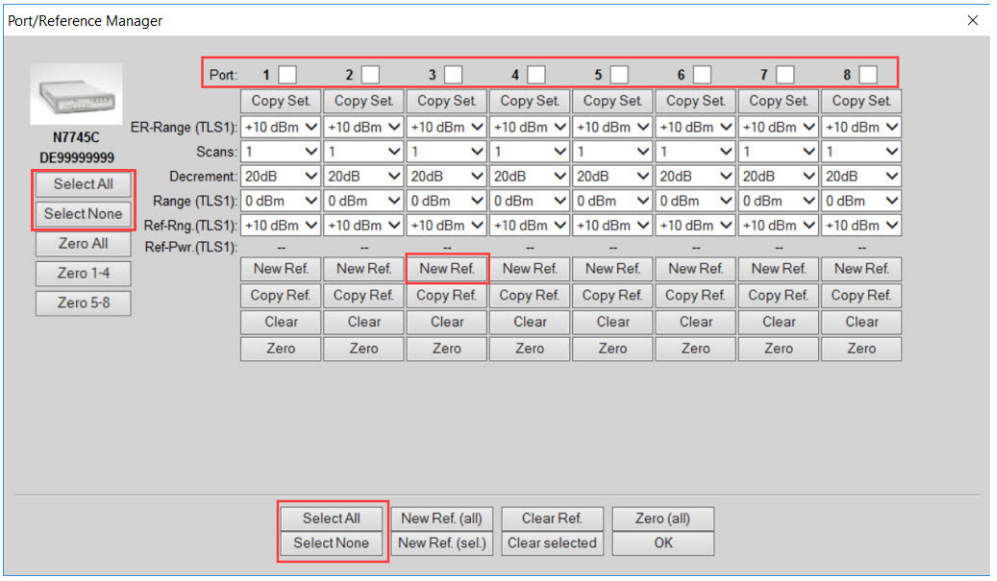
3 Click the Port/Ref. Manager... button.



4 In the Port/Reference Manager, clear any existing reference data that might not be required. See Procedure Summary for Clearing Reference Data from a Port on page 83.



5 Select the ports for which you want to perform the reference measurements. You can perform a single measurement sweep on multiple ports simultaneously by using the check boxes above the individual ports. You can also select or deselect all the ports corresponding to an instrument by clicking the Select All or Select None buttons just under the instrument descriptor.



If there are multiple instruments displayed in the Port/Reference Manager, use the Select All and Select None buttons at the bottom of the dialog to select or deselect all the ports for all the instruments.

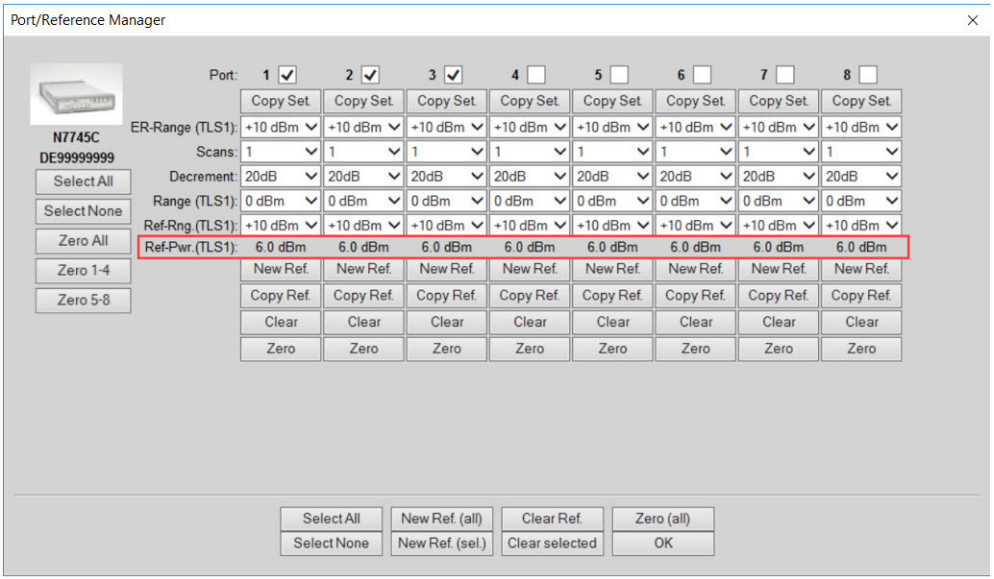
6 Select the New Ref. button corresponding to a port to perform a reference measurement for a single port.

7 Click the New Ref. (all) or New Ref. (sel.) button to perform a reference measurement on all or the selected ports, respectively. All checked ports will be included in the reference sweep, once you click the New Ref. (sel.) button. To obtain a reference on all ports simultaneously, click the New ref. (all) button. In this case, it does not matter which ports have been checked or unchecked.

8 The engine might prompt you in case there has been a considerable time lapse since the last TLS zeroing operation. Click Yes to instruct the LS engine to perform TLS zeroing now. You might also want to refer to TLS Lambda Zeroing on page 139.

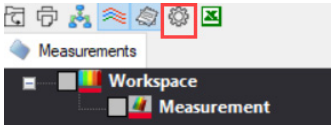
An overview of the reference data is displayed in the Port/Reference Manager. Each channel that has been processed in this way will display the average power of the reference sweep on that port.
In case of a Multi-TLS configuration, the information will be shown for

each TLS individually.



NOTE

After making a reference measurement, the raw data associated with the measurement can also be viewed for verification. To enable raw data trace visualization, enable the **Store Measurement Raw Data in Files** setting on the Advanced tab. Additionally, set the Detail Level to 1 in the Options dialog/Property Editor. (The Options dialog/Property Editor can be accessed by clicking the Options (or the gear) icon on top of the Browser tree on the left.)

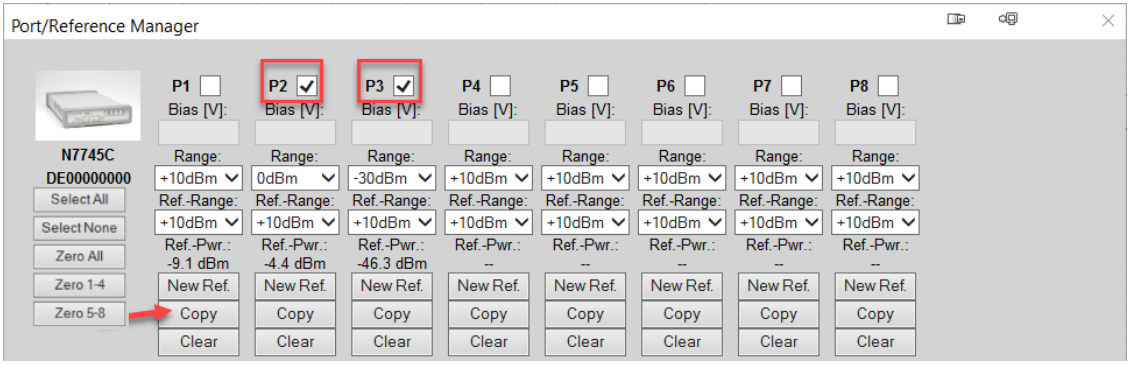


Related COM API

- method GetReferenceAvailable
- method GetReferenceDescription

- method GetReferenceAveragePower
- method StartCustomReference/ StartCustomReferenceSinglePort

You can obtain reference data on one or more ports, then copy those traces to be used as reference for other ports. To do so, obtain the actual reference sweep, then check all destination ports (make sure other ports are unchecked) and click the Copy button at the source port:



All destination ports will show the average power of the source port afterwards. See Procedure Summary for Copying Reference Data from One Port to Another on page 87.

NOTE

Especially when using N7745A specials with electrical input ports, M9601A, M9614A, M9615A, or supported B2900-series Precision Source / Measure Units, the Copy operation is required. A reference measurement is performed on one (or more) optical power meter ports, then the reference data needs to be copied to the electrical ports.

You can clear data for individual ports by clicking the Clear button of the respective port. By checking several ports and clicking the Clear selected button, reference data for those ports is cleared. See Procedure Summary for Clearing Reference Data from a Port on page 83

By clicking the Clear Ref. button, the complete reference data is cleared from memory. Also upon restarting the engine, no reference data will then be automatically loaded, until a new reference has been obtained.

Actually, the engine might still load the default reference files during engine activation, but these do not contain any reference data. You might check the Port/Reference Manager to check whether any actual port reference data exists.

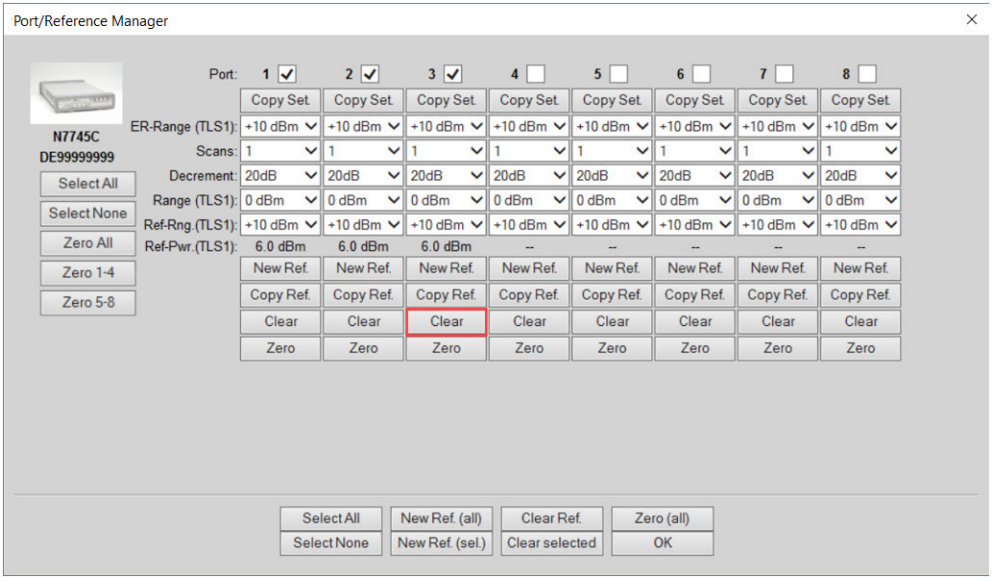
NOTE **Ports with a not-a-number indicator as power level in the reference details (or no average power indication at all) will not be evaluated during subsequent measurements.**

You can perform a zeroing operation on the ports of the power meters /SMUs or the return loss module by selecting the port check box and clicking the corresponding Zero button. To zero all the ports, click Zero (all) button. Depending on the power meter module, you can also zero ports in groups. This may make it more convenient to block the light at only the selected ports.

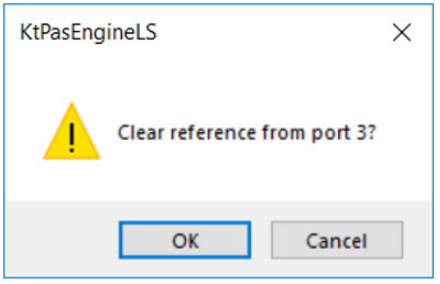
You can load, save and delete reference files from the main GUI. See Managing References on page 93 for details.

Procedure Summary for Clearing Reference Data from a Port

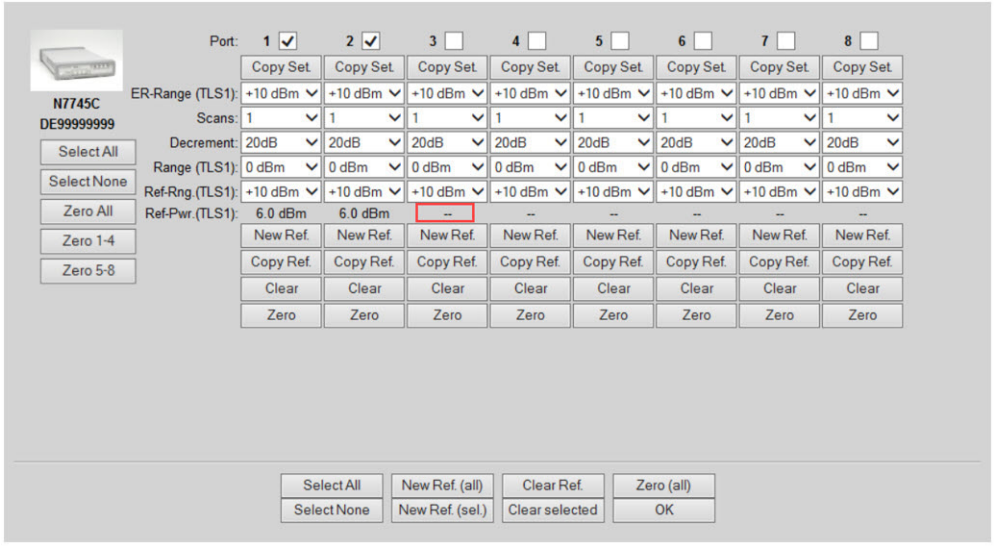
1 Click Clear corresponding to the port from which you want to clear the reference data.



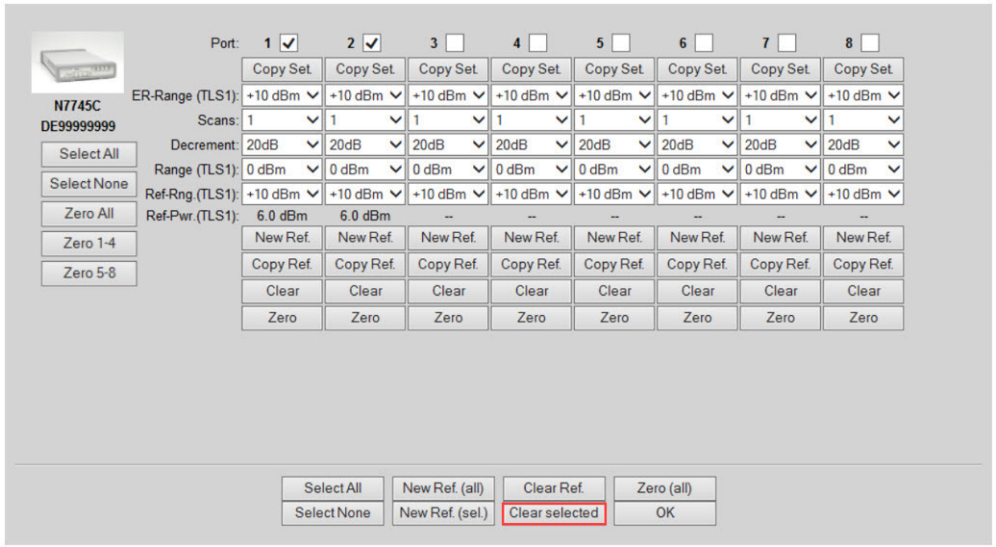
2 Click OK to prompt that is displayed.

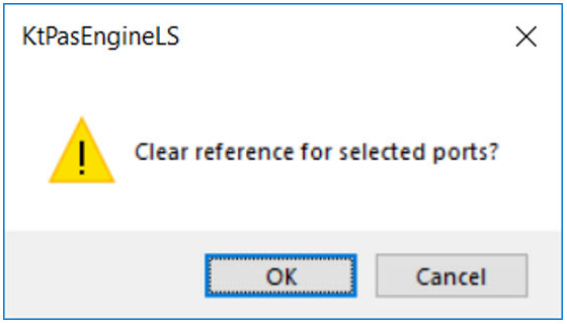


The reference data is deleted from that port.

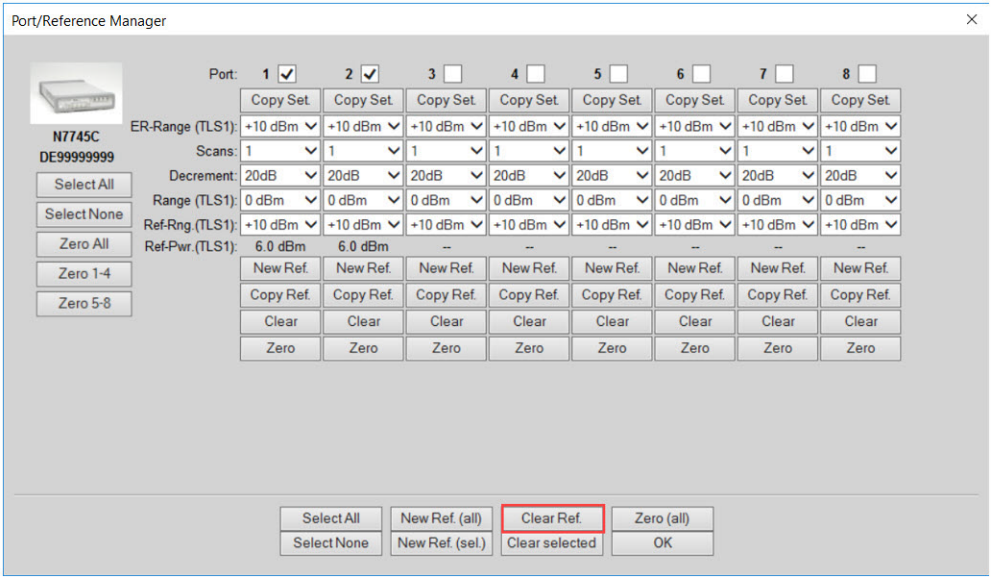


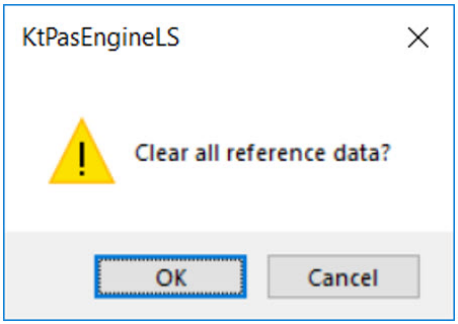
3 Click Clear selected to delete reference data from the selected ports. Click OK to the prompt that is displayed.



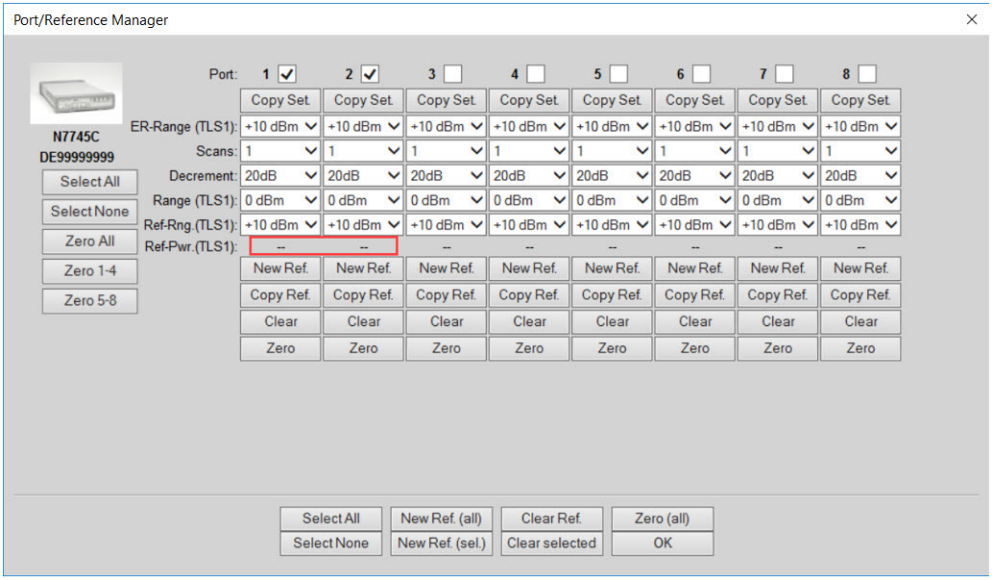


4 Click Clear Ref. to delete reference data from all the ports. Click OK to the prompt that is displayed.





The reference data gets deleted from the selected or all ports, as indicated in the Port/Reference Manager.



Related COM API

- method ClearReferenceChannels / ClearReferenceChannelsSinglePort
- method ClearReferences

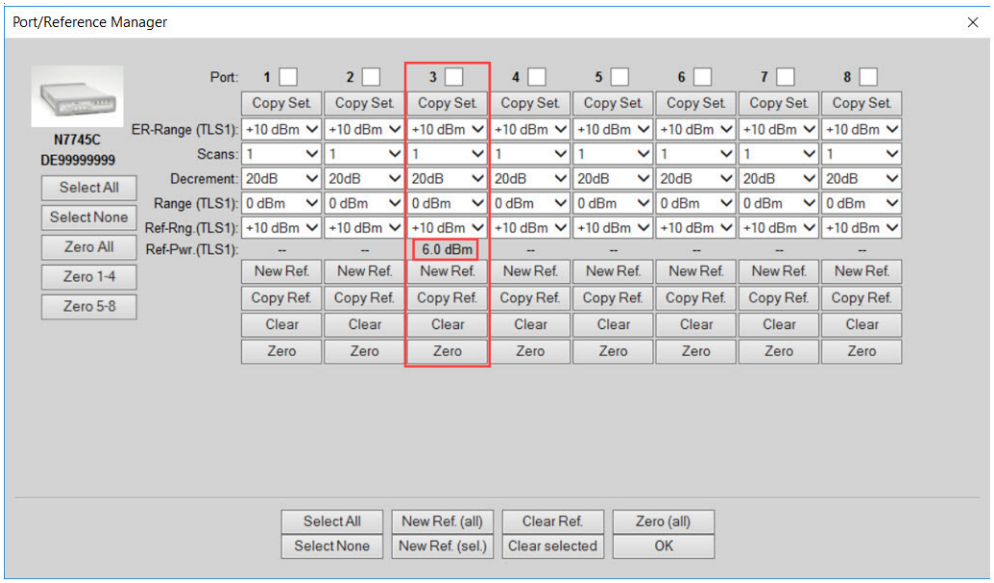
Procedure Summary for Copying Reference Data from One Port to Another

Rather than performing a reference measurement on each port of your instrument, you can take a reference measurement on one port and copy the reference data from that port to the other ports on which the swept measurement is intended to be performed. This saves time when you are confident that the conditions on the ports are sufficiently similar.

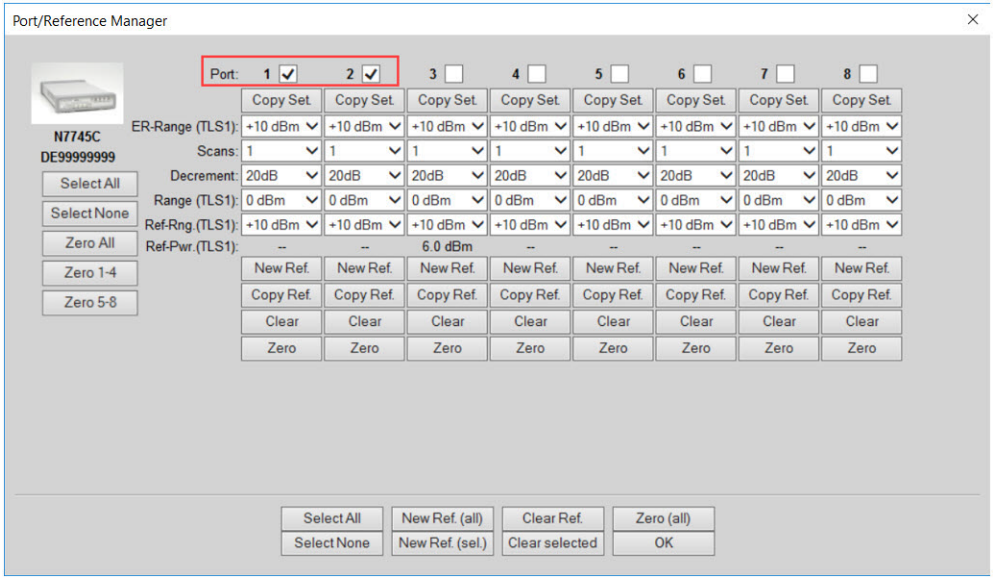
Another use case for copying reference data is when performing a measurement using an electric meter or a power meter with electrical ports.

Perform the following steps to copy reference data from one port to another:

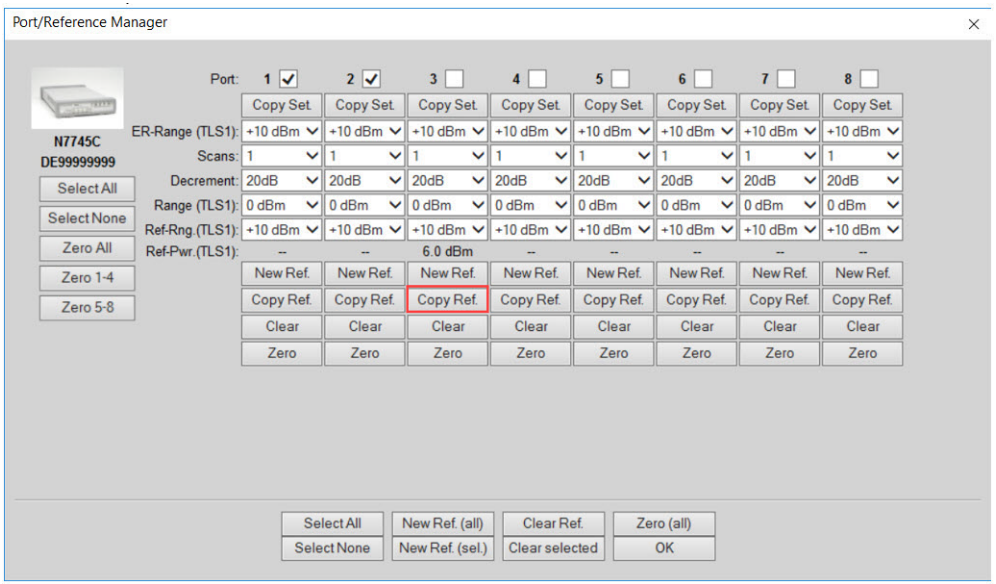
1 Ensure there is valid reference data on a port of the power meter. Perform a reference measurement, if required.



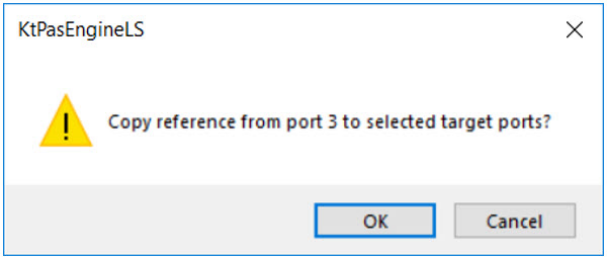
2 Select the check boxes corresponding to the port numbers on which you want to copy the reference data.



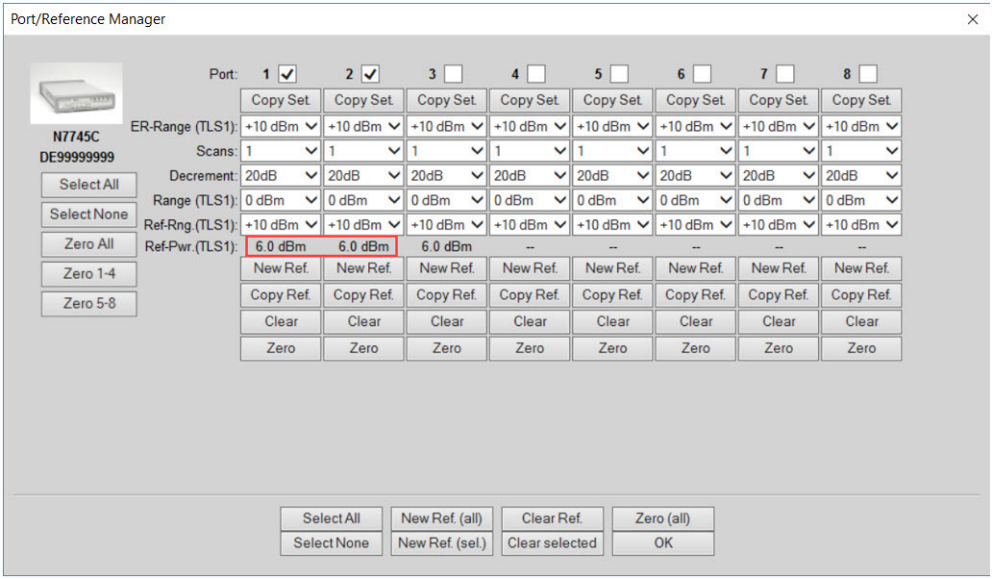
3 Click Copy Ref. corresponding to the port from which you want to copy the reference data to the selected ports.



4 Click OK to the following prompt.

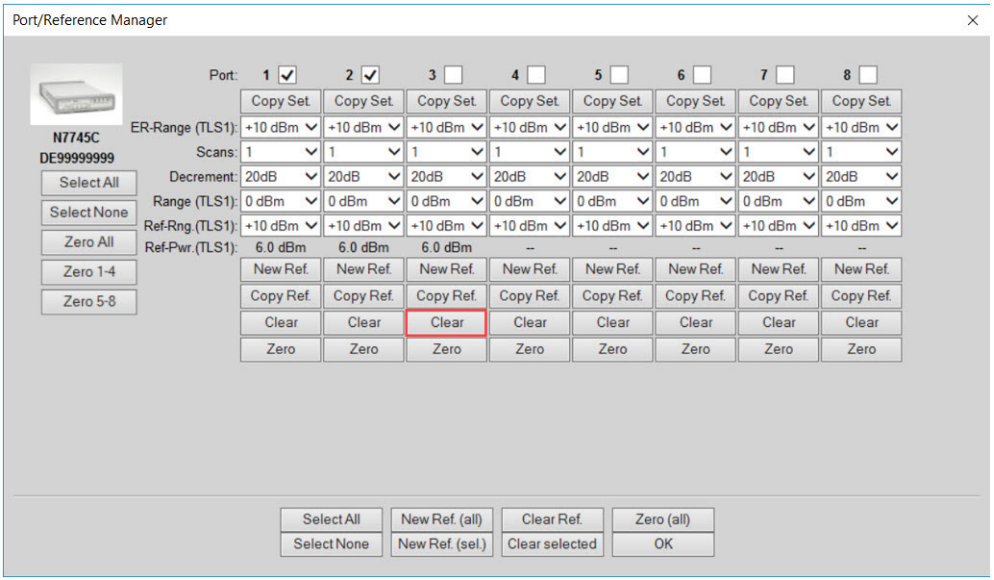


The reference data is copied to the selected ports.

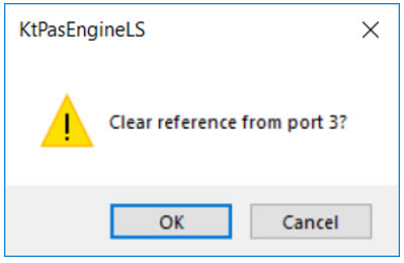


5 Optionally, clear the reference from the original port on which the actual reference measurement was performed, if not required. Click the

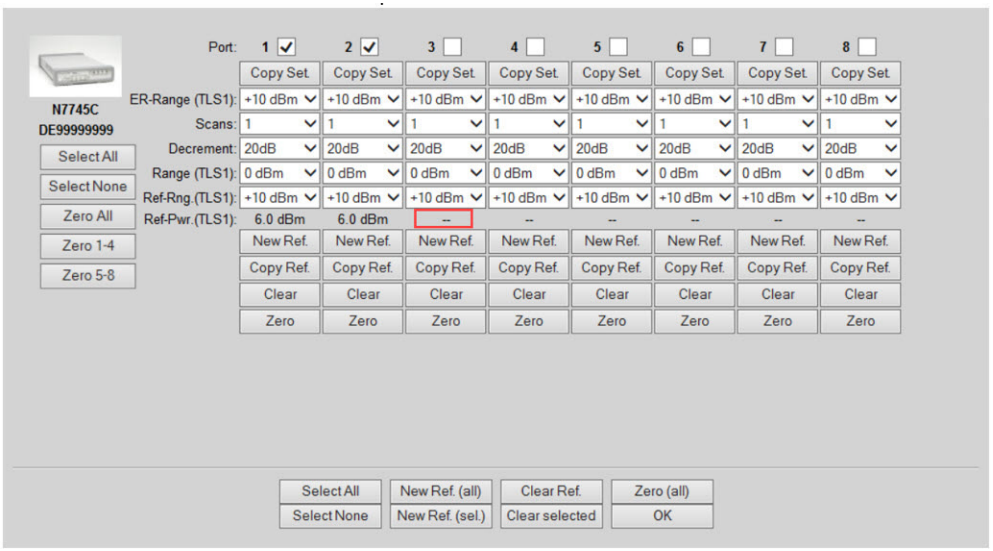
corresponding Clear button.



6 Click OK to prompt that is displayed.



The reference data is deleted from that port.



Related COM API

- method CopyReferenceChannel / CopyReferenceChannelSinglePort

NOTE

Since the reference data is obtained and required by the engine server, the files will be stored in a folder on the server PC, so no different folder may be selected. This folder should be:

C:\ProgramData\Keysight\Photonic Application Suite\LS Application\References\

When uninstalling and upgrading the PAS main package, you will be prompted whether to uninstall or to keep application data for this engine, such as reference files.

NOTE When you start working with a certain setup and referencing mode, it is useful to perform a verification measurement after taking the first reference. Do this by either leaving the reference patchcord (or the splitter output pigtails) connected to the ports used during the reference measurement or by connecting them to other ports that are supposed then to have valid references. Click **Run Single** and check the measurement results.

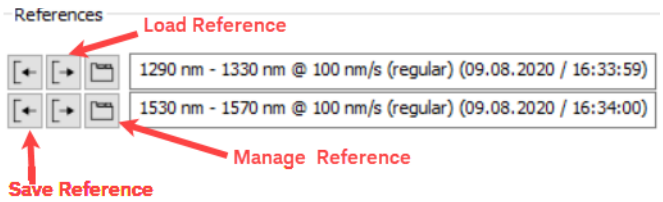
NOTE When working with IL de-embedding, it is not necessary to modify the IL de-embedding file settings in Port/Reference Manager. Reference measurements will always ignore any IL de-embedding settings.

Managing References

When you run the Lambda Scan engine for the first time and perform a reference measurement, the references will be saved automatically, using a default filename (one file per TLS). However, when exiting the Lambda Scan engine after subsequent reference measurement(s) have been performed, this automatically generated file will be overwritten.

Saving, Loading, and Deleting Reference Files

You can save a reference file by clicking the Save Reference button and load a reference file by clicking the Load Reference button. Clicking these buttons will bring up corresponding file selection dialogs, showing details on the reference files.



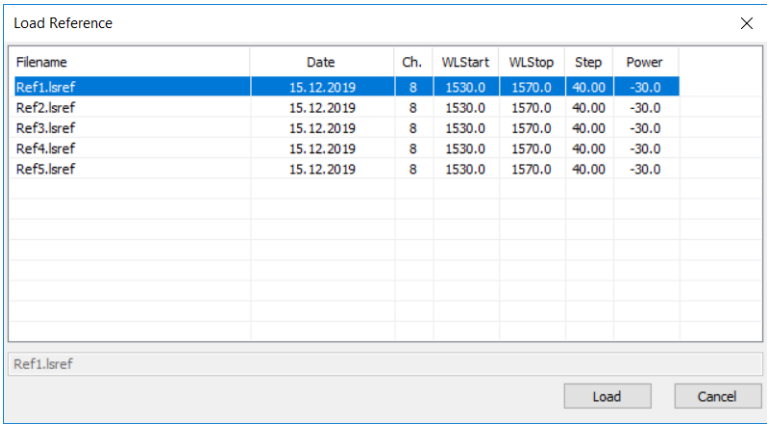
For best results, a reference file should be made with exactly the same instruments and sweep settings as the device measurement. In certain cases, it can be enough to use a reference that has been obtained with a similar instrument and similar settings. So, before loading a reference file

for a laser, check the sweep parameters and continue if they match your desired measurement settings, or if the requirements are less demanding, e.g., to perform a preview measurement only.

During load, the engine will check if the reference data has been obtained with exactly the same laser instrument (model code, serial number, firmware version). A warning will be shown if the laser that this reference is being loaded for differs in any of these aspects. Reference data will be loaded, but for optimum results, a new reference should be obtained, using exactly the same laser that will be used for subsequent measurements.

The engine will not load a reference file that does not have an overlapping region with the TLS that the engine is loaded for.

The engine does not check which power meter instruments were used for obtaining the reference, as all the relevant instrument-specific information is included in the reference file. The software allows using a reference file obtained with one power meter for a subsequent measurement on a different power meter.



By clicking the Manage References button you can delete references. You can select multiple references at the same time by holding the shift or ctrl key while selecting the files.

After every reference operation (new, copy, clear), the default reference file will be updated or deleted automatically. This ensures that upon engine restart, the same reference information is restored, even in case it has not been explicitly saved. The default reference file names areLambdaScanRef_TLS0/1/2.lsdef in the default reference save folder.

You may load the lsdef references, but not overwrite them by an explicit save reference operation, nor delete them. They get updated only automatically, by performing reference operations.

When loading or saving a reference file, the current configuration in memory is updated to use that reference file for the respective laser. Thus, you can save agconfig files that load specific reference files during activation, instead of the default reference files.

To revert an agconfig to using a default reference file, load the matching file (LambdaScanRef_TLS0.lsdef for the first TLS, etc.) once, then save the agconfig file.

In the save reference dialog, the prompted save filename is either a default name (same as automatic default files, but with the regular reference extension ".lsref", instead of the ".lsdef" extension used for automatically saved reference files) or the name of the load / saved reference file that was associated with that laser either in saved agconfig, or as updated by latest load/save operation.

When saving to a custom reference file, the reference data is stored as per that instant in time. After performing any additional reference operations, another save operation is required to have the reference file updated accordingly. This is different from using the automatic reference files only, which get updated automatically and immediately. The difference in operation is intended to avoid unintended modification or deletion of actively saved reference files.

Configuring Dynamic Range Related Settings (Ranges and Averaging Time)

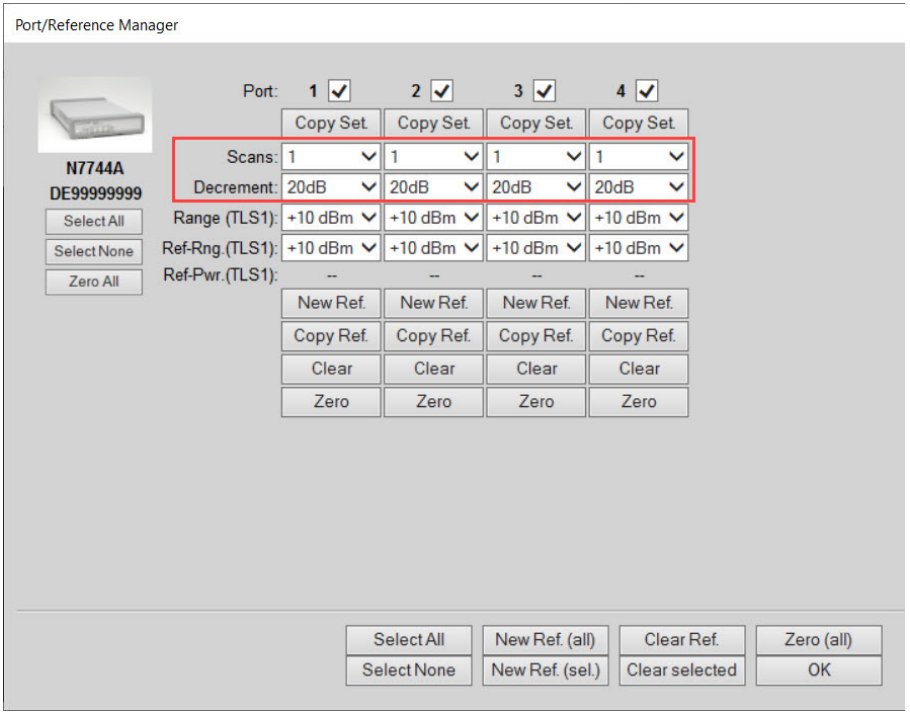
The LS engine allows for performing swept wavelength measurements on devices with very high dynamic ranges.

By default, the engine is configured to automatically choose for each port the most sensitive power meter range, such that the input power does not exceed the maximum allowed at that range, thus avoiding data clipping. By default, it is also set to make a single dynamic range scan. That way, each time a measurement is performed, it will yield data subject to the power meter's dynamic range, depending on the type of power meter used and the averaging time selected (refer to the power meter's specifications for further details).

The dynamic range of the measurement can also be optimized by choosing the laser power so that the maximum power at the power meters is close to the maximum allowed at the power meter range setting. This is normally 3 dB higher than the nominal range value. For example, the power meter can measure up to +3 dBm when using the 0 dBm power range setting.

To extend the dynamic range of the measurement, the engine can be set to perform multiple sweeps at different ranges, then combine the results. To use this feature, choose a number of dynamic range scans and a decrement value that determines the amount by which the power meter range should be changed between sweeps. For the N7744 or the N7745, a typical range decrement would be 20-40dB, while for N7747A/N7748A instruments it would be 20-30dB, based on the single-sweep dynamic range of the models. Depending on the combination of power meter types involved and the initial ranges of each port, it is possible that the minimum range of a power meter is reached before the lowest range value defined by dynamic range scans and range decrement is reached. In that case, the power meter's minimum range is used instead.

You can use the Port/Reference Manager to define power range / dynamic range scan settings on a per-port basis.

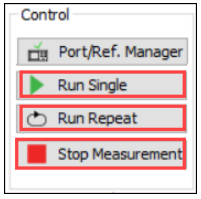


NOTE Using the Autoranging feature might also lead to different Initial Range settings for different power meter ports and, in turn, to a smaller number of scans actually being performed if the minimum instrument range is already covered within fewer scans.

Best dynamic range is achieved for high averaging times, while highest wavelength resolution may require low averaging times unless low sweep rates are used. If the Auto option for the Avg. Time control is selected, the engine chooses the appropriate averaging time based on the selected wavelength step size. User higher step sizes for reduced noise and increased dynamic range. Refer to Instrument/DUT Bandwidth and Measurement Trigger Timing on page 47 for further details.

Performing a Measurement

After successfully completing a reference measurement or loading a reference file from disk, connect the signal fiber to the input port of the DUT (usually using the reference patchcord or another measurement patchcord, to avoid excessive reconnection at the instrument). Then connect any output ports of the DUT to the power meter / SMU ports. Click Run Single or Run Repeat on the Swept Measurement tab.



NOTE Though it is possible to run referenced measurements with sweep settings that differ from reference sweep settings, the best results are achieved if the two sets of settings are identical.

During a measurement you can abort the current measurement sweep and stop the repeat operation by clicking Stop Measurement. The measurement will be stopped as soon as possible, but usually not immediately, because some hardware operations have to be completed

first. Usually the next measurement is completed before stopping, so this feature should be used primarily for stopping the Repeat mode operation and to interrupt undesired/unintended autoranging repetitions.

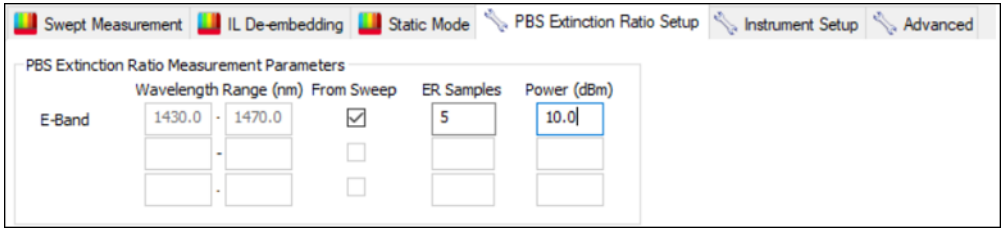
NOTE

Measurement data will be returned for all ports that have their check box set in Port/Reference Manager (see Performing a Reference Measurement on page 70). Note that either all the selected ports must have reference data assigned, or no selected port may have reference data assigned. A mix of selected ports with and without reference data is not supported. Unselected ports may or may not have reference data assigned. For measurements without reference data, the calculation will use the TLS power setting as the input power level to the DUT.

Performing Polarization Extinction Ratio Measurement

Polarization Extinction Ratio measurements are considered to be supplementary measurements and are performed prior to the actual sweep measurement if selected (see Application Setup Parameters on page 54 or Reference: Interface “IKtPasServerEngine” on page 169 for details on how to include those measurements).

This measurement is made at a set of equally distributed distinct wavelength samples within a chosen wavelength range, which, by default, is the same as that for the swept measurement. The TLS power can also be defined for the extinction ratio measurement to optimize the dynamic range as mentioned above. The measurements are made at the chosen sample points. This information can be specified under the PBS Extinction Ratio Setup tab.



To perform the measurement at just a single wavelength, e.g., the center wavelength, set this field to one. (It is to be noted that even in this case two very closely placed samples will be plotted in the resultant graph and consequently in the resulting OMR data as the LS Engine/File Viewer does not visualize single data samples.)

Polarization extinction ratio measurements are done by creating a pseudo-continuous SOP pattern of a fixed length / duration, then evaluating maximum and minimum of the obtained power trace. This allows measurement of higher extinction ratio than can usually be obtained from the matrix analysis used for the PDL results because some of the power samples will be very close to the SOP for minimum transmission.

The noise level or sensitivity limit of the detector will contribute uncertainty to the measurement that is higher for high PER and IL or low responsivity of the DUT, which all reduce the minimum signal level. To judge the impact of noise on the measurement, the measurement results also contain an uncertainty estimation, shown in an additional measurement info pane at the bottom of the GUI. The measurement data taken during the “dark” measurement, before the actual PER measurements, are evaluated for double standard deviation (assuming gaussian noise). The impact is given in the measurement results as the DUT PER level for which the present noise would result in an uncertainty of +/-1dB. PER results higher than this level would have greater uncertainty.

As Polarization Extinction Ratio measurements are a supplement to regular sweep measurements, they use a very similar measurement timing (hold-off and averaging time (see Instrument/DUT Bandwidth and Measurement Trigger Timing on page 47)) as swept measurements do, even though the TLS is not sweeping in this case.

It is to be noted that Polarization Extinction Ratio measurements obtain 10,000 samples per wavelength. Using settings that cause large sampling periods (slow sweep rate, large target step size, large averaging time), the extinction ratio measurement part might take rather long (roughly 10,000 times twice the averaging time, i.e., 0.5s for 25us averaging time, 4s for 200us averaging time and 400s for 20ms averaging time.)

What devices can be characterized

Polarization Extinction Ratio measurements performed with the LS engine are mostly intended to verify a certain minimum polarization extinction ratio of integrated receiver devices, typically around 20 dB, or polarizer / polarization beam splitter devices with typical polarization extinction ratio values of 30 dB or more.

It is not intended for measuring very low PER/PDL values, since N7786B/C instrument PDL and measurement setup PDL are not taken into account. Parasitic PDL of the measurement setup depends on the components used in the setup. Fabry-Perot resonances (can occur when using multiple straight connectors) will have a similar impact on the results and should be avoided.

Required instruments

The Polarization Extinction Ratio measurements use a setup similar to the regular IL/PDL measurements.

- One or more tunable laser sources
- One Polarization Synthesizer
- One or more power meter or SMU

Prerequisite

- Zeroing has been performed for the TLS and power meter(s)/SMU(s)

Procedure

Ensure that the measurement is started in stable temperature environment and that the instruments should have been powered up for 20-30 minutes prior to the measurement.

NOTE **Polarization Extinction Ratio measurements do not use reference data. All the selected ports in the Port/Reference Manager are used for these measurements. There is no requirement to have reference data on the selected ports.**

NOTE **Polarization Extinction Ratio measurements are independent of any IL-de-embedding setting.**

1 Open the Configuration Wizard to include the requirement instruments in the measurement setup. Refer to the Configuring Instruments Using Configuration Wizard on page 66 for instructions on selecting instruments using the Configuration Wizard.

2 Connect the DUT to the polarization synthesizer on one end and the power meter on the other. (If a return loss module is part of the current

configuration, then connect the output of that module to the DUT input, instead of the polarization synthesizer output.)

3 Select the PBS Extinction Ratio check box under the Swept Measurement tab.

4 Under the PBS Extinction Ratio Setup tab, specify values for the fields as described in Polarization Extinction Ratio Parameters on page 64.

5 Click the Run Single button on the Swept Measurement tab.

6 The engine might prompt you in case there has been a considerable time lapse since the last TLS zeroing operation. Click Yes to instruct the LS engine to perform TLS zeroing now. You might also want to refer to TLS Lambda Zeroing on page 139 for more information.

NOTE If the noise / disturbance cannot be assumed gaussian, the PER uncertainty estimation will be incorrect. Therefore it is very important to avoid induced distortions, especially in case of small signals (high loss / low effective responsivity).

NOTE Due to the logarithmic scale of the PER and the absolute level of noise and disturbances, measurements of PER values significantly above the value given from the uncertainty analysis may differ greatly between subsequent measurements.

NOTE In case of very high actual DUT PER, due to the finite length of the SOP pattern used, the measured PER value may not be as high as the actual PER value.

NOTE In case of high PER / high loss DUTs, better PER results may be achieved by using larger averaging time settings, even if not required for the swept measurement.

Results

Polarization Extinction Ratio measurements will add a graph named “Extinction Ratio” (with a TLS ID suffix) plotted against the chosen ER samples. The data for the chosen plots is shown individually. in addition they will add an additional ERUncertaintyDueToNoise measurement info pane that shows the power levels port-wise for the chosen samples.

Related COM API

- property ERIncludeMeasurement
- method SetERPWMSensitivity
- method SetERPWMSensitivityAllPorts
- method GetERPWMSensitivity
- method SetERTLSPower
- method GetERTLSPower
- method SetERWavelengths
- method GetERWavelengths
- method SetERUseSweepRange
- method GetERUseSweepRange

Measuring Dark Current

Background

This measurement can be used to characterize the dark current from the photodiodes. The dark current is generally offset by zeroing before the required opto-electric measurements are performed if the photodiode is connected during zeroing. So if measurement of the photodiode dark current is required, the DUT must be disconnected from the SMU(s) during the zeroing operation.

Required instruments

- One or more tunable laser sources (not required for dark current measurement, but for the sweep measurement that needs to be performed afterwards).
- Optional: One Polarization Synthesizer
- Optional: One or more power meters (for performing an optical reference measurement used in subsequent swept measurement)
- One or more SMUs

Procedure

Dark Current measurement is performed prior to the actual sweep measurement if set to be included (see Application Setup Parameters on page 54 or Reference: Interface "IKtPasServerEngine" on page 169 for details on how to include this measurement).

Ensure that the measurement is started in stable temperature environment and that the instruments should have been powered up for 20-30 minutes prior to the measurement.

NOTE Dark Current measurements do not use reference data. All the selected ports in the Port/Reference Manager are used for these measurements. There is no requirement to have reference data on the selected ports and nor does the existence of reference data for a port guarantees that it will be included in the dark current measurement.

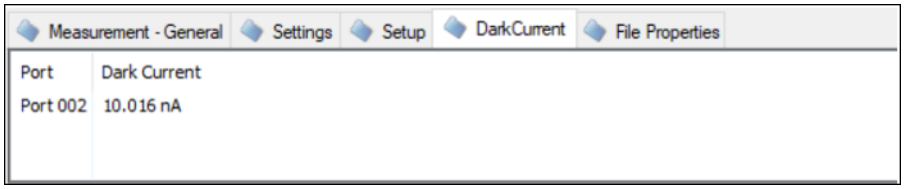
NOTE Dark Current measurements are independent of any IL-de-embedding setting.

1. Open the Configuration Wizard to include the requirement instruments in the measurement setup. Refer to the Configuring Instruments Using Configuration Wizard on page 66 for instructions on selecting instruments using the Configuration Wizard.
2. Connect the optical output fiber (depending on setup configuration this can be the TLS output, the optical switch output, the polarization synthesizer output, or the return loss module output) to the optical input of the DUT.
   Connect the DUT electrically (usually via the bias voltage connectors) to the SMU input.
3. In the Include Supplementary Measurements section under the Swept Measurement tab, enable the Dark Current check box.
4. Click the Run Single button to perform the swept measurement.

Results

Dark current measurements do not add graphs, since they return single scalar values for each measurement port. Dark current results are listed in an additional measurement info pane.

A Dark Current measurement info pane that shows the dark current port-wise for the ports having reference data is displayed under the graphs area.



Related COM API

- property DarkIncludeMeasurement

IL De-embedding

In optical and opto-electrical measurements using the Lambda Scan engine, you generally have a requirement of performing frequent reference measurements for more accurate results. However, for insertion loss arising out of the devices placed in the path of the DUT, such as switches or splitters, you can characterize those once and use the data in subsequent measurements provided the setup and environmental conditions remain considerably constant.

Using the IL De-embedding functionality of the Lambda Scan engine, polarization-averaged insertion loss of components in the measurement setup, such as optical switches or splitters, as well as differences in the absolute responsivity of individual power meter ports can be characterized once and corrected for in subsequent IL/PDL sweep measurements. Insertion loss characterization data of individual components, such as wafer probes, can also be applied without prior characterization measurements, when available in OMR or CSV file format. The de-embedding data are generally used to remove IL differences in the measurement paths used for the reference measurement and DUT measurements. Multiple data files can be concatenated for the measurement to represent paths with multiple individually measured components.

The IL De-embedding measurements are performed independently of the swept measurements. The necessary controls for regulating these measurements can be found under the IL De-embedding tab and Port/Reference Manager (after IL De-embedding is enabled from the IL De-embedding tab).

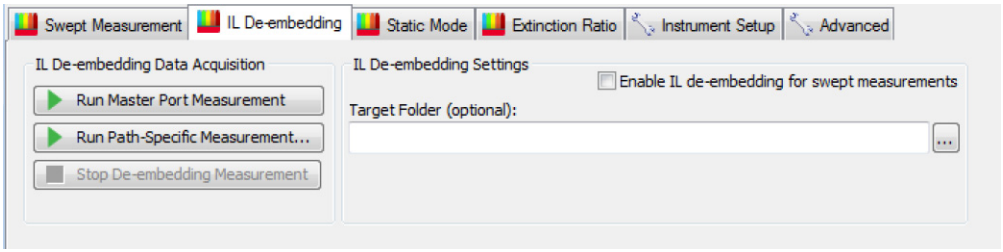
For simple corrections, such as quick comparisons and sanity checks, it is also possible to provide one or more scalar values per port instead of complete measurement files for de-embedding.

NOTE

You might also want to refer to the Application Note, Calibrating Optical Paths in Spectral Test Station Using N7700A IL/PDL SW Engine, to know more about how N7700 software uses IL De-embedding to simplify system calibration in test solutions with multiple optical paths, to support use of optical switches, splitters and especially optical probes for measuring wafers and chips.

Characterizing Components / Obtaining IL De-embedding Data

For obtaining component characterization data, there is a corresponding tab, the **IL De-embedding** tab, in the main engine GUI.



When obtaining IL de-embedding data, a master port measurement must be performed as a first step, then any number of specific path measurements may be performed.

Use the **Enable IL de-embedding for swept measurements** check box to enable/disable application of IL de-embedding data without having to clear all port-specific de-embedding file list controls in the Port/Reference Manager.

Performing Master Port Measurements

A master port measurement is required as a reference for subsequent specific-path characterization measurements. To perform a master port measurement, connect the output of the N7786B/C Polarization Synthesizer directly to one of the power meter ports in the setup and click the **Run Master Port Measurement** button. (If future reference measurements will be made over a path including other components, like

a switch, then this same path can be used for the master port measurement too, but this can make the concatenation of multiple elements in a path more complicated.)

NOTE An optical signal should only be applied to a single port, which will then be identified automatically.

NOTE IL de-embedding measurements use the same settings as regular swept measurements or reference measurements do. In order for a subsequent measurement to work with IL de-Embedding data, ensure that the IL de-Embedding data covers the full range that the subsequent measurement is supposed to cover. Ideally, IL de-Embedding measurements should be obtained over the whole available wavelength range, for maximum flexibility, later on. There is a trade-off regarding sweep rate for IL de-Embedding measurements. Larger sweep rates result in faster measurements and smaller files, but at the same time reduce the measurement resolution and the available sweep range. For typical IL de-Embedding applications (switches, couplers, etc.), a low resolution is not an issue, but a reduced sweep range (e.g. start wavelength) might be.

NOTE Any master port measurement remains available during the current session. A new one needs to be obtained after restarting the LS engine, in case further setup-component characterization measurements are to be performed.

Performing Specific Path Measurements

For specific path measurements, connect the output of the N7786B/C Polarization Synthesizer to the input of the instrument, component or path to be characterized, and the output of said device to the power meter port used for the master port measurement. Then click the **Run Path-Specific Measurement** button. You will be prompted to specify a name for the file to be saved. The default folder for the file save dialog will be the one defined by the **Target Folder** control, unless the control is empty or describes a non-existent folder.

NOTE Target file names must not contain any semicolon and must not start with plus (+) or minus (-) characters, as these characters are used for operation control, as described in De-embedding Data for Setup Components on page 107.

NOTE Although not strictly necessary, using the same power meter port as for the master port measurement provides measurement data characterizing the path difference alone, without a contribution due to differences between power meter ports. Measurements for correcting differences in absolute responsivities of different ports (details described in De-embedding Data for Cross-Port Responsivity on page 109) can be made and applied separately for flexibility in configuring and concatenating IL de-embedding data for the optical paths used later.

NOTE Since using the same power meter port during IL de-embedding master port and IL de-embedding specific path measurements provides de-embedding data independent of the port used, any subsequent reference measurements may be performed on any port as long as de-embedding data for all ports is considered correctly (see Applying IL De-embedding Data on page 107 for details). When creating additional files for correction of any differences between power meter port responsivities, the IL de-embedding master port should best be the one that is used for subsequent reference measurements (see De-embedding Data for Cross-Port Responsivity on page 109 for details), to simplify the required concatenations.

Applying IL De-embedding Data

De-embedding Data for Setup Components

Once IL de-embedding data for setup components is available, the Port/Reference Manager is used to configure which de-embedding files are to be applied to each power meter port's data.

NOTE Based on the state of the **Enable IL de-embedding for swept measurements** check box on the IL De-embedding tab of the main GUI, the controls described in this section might be hidden to indicate that no IL De-embedding will be performed unless the said check box state is toggled before any further measurement/recalculation.

For each power meter port, there is a configuration section for IL de-embedding settings.

The text control may be edited directly, using copy/paste or using the buttons below the text control.

In general, the text control defines a list of IL de-embedding files to be applied to the corresponding power meter port.

Alternatively, one or more scalar values, representing loss or gain in dB can be provided for very simple and wavelength-independent corrections. This can, e.g., be useful in case a live measurement is to be compared with a saved measurement that shows similar characteristics but at a significantly different power level. It is possible to use both scalar correction values and file names in any desired sequence, separated by semi-colons. For example, at Port 1 the IL Deemb field could use “+Sw1”, at Port 2 “+Sw1;+PM2”, at Port 3 “+Sw1;6;+PM3” and so forth on the remaining ports.

Filenames may consist of full path, file name and file extension, but for the sake of compact visualization and efficient editing, path and extension may be omitted.

File names without full path are expected within the folder defined by the **Target Folder** control on the IL De-embedding tab of the main GUI. It is also allowed to use sub-folders (plus filenames) here, which are then concatenated with the Target Folder settings.

For file names without extension, the LS engine will check whether a file of the respective name exists with either OMR or CSV file extension. If both exist, the OMR file will be used.

Each file in the list must be preceded by either a plus (+) or a minus (-) character, indicating the de-embedding operation to be performed. The corresponding operation is defined as follows:

Plus (+)	The device characterized by this file is present in the current path to this power meter port.
Minus (-)	The device characterized by this file was present in the path to the power meter port used for obtaining the currently active reference measurement used for this port.

As explained in Performing a Reference Measurement on page 70, reference measurement data is stored without applying any IL de-embedding data that might be configured in the Port/Reference Manager. Therefore, any devices present in the optical path during a reference measurement, must be corrected for in subsequent swept measurements by using the corresponding de-embedding files, preceded by a minus character.

Clicking the buttons with the plus or minus sign opens a file selection dialog and, unless canceled, adds the selected file using the corresponding prefix and a trailing semicolon to the corresponding port's de-embedding file list. Clicking the **Clr** button clears the corresponding port's de-embedding file list.

NOTE

A semicolon is expected between any two file names, so file names must not contain semicolon characters. A semicolon at the end of the file list is allowed, but not necessary. Multiple, adjacent semicolons are treated as a single one.

The **Copy IL-d** button can be used to copy the IL de-embedding settings from one port to another. This button also has another use of clearing data from all the ports in one go. In that case, clear the data from the text field of one port, click **Select All** to select the remaining ports, and click **Copy IL-d.**

De-embedding Data for Cross-Port Responsivity

While wavelength-dependent responsivities of the power meter ports are stored as part of the instrument calibration data and automatically corrected for during swept IL/PDL measurements, IL de-embedding may be used to compensate for responsivity differences between individual ports.

To do so, obtain an IL de-embedding master port measurement, as described above, then remove the fiber connector from the master port, connect it to another power meter port and run a specific path measurement. Repeat this for all power meter ports.

In the Port/Reference Manager, for each active port in your setup, add the corresponding cross-port IL de-embedding file to the IL de-embedding text control of the corresponding port.

NOTE Keep in mind that for appropriate cross-port de-embedding, reference measurements must be obtained on the same port where the IL de-embedding master port measurement was obtained.

NOTE If the Specific Path measurements above were made using power meter ports other than the one used for the master port measurement, then the Cross-Port Responsivity contribution is already included in those data and should not be added again in the IL de-embedding settings. But then that port is bound to that path so reconfiguration of the setup is more complicated and less flexible.

File Formats

While the IL de-embedding measurement of the LS engine can be used to characterize setup components, it is also possible, to use existing OMR or CSV files for IL de-embedding operation.

In case of OMR files, it is important that there is an “Average IL (TLSx)” trace present, as is the case for LS engine measurements, or an “IL” trace, as is the case for IL engine or FSIL engine measurements. When accessing OMR data through the OMR File Handler COM interface, the corresponding trace name is RXTXAvgIL or TLSx_RXTXAvgIL.

The IL de-embedding data wavelength range must exceed or at least match the wavelength range of the current measurement.

NOTE In case of multi-TLS IL/PDL measurement files, there is a single IL trace for each TLS involved and the appropriate one is used automatically.

When using CSV files, highest evaluation speed is achieved if both the PC running the IL/PDL evaluation and the CSV file use the same decimal and list separators characters. The IL/PDL evaluation should be able to cope with mismatching settings between operating system locale and CSV file used, but might require some additional time for data conversion.

CSV files for IL de-embedding must consist of exactly two data columns, namely the wavelength in meters and the insertion loss (IL) in dB.

NOTE

When viewing OMR measurement files in File Viewer (or in measurement engine GUIs), the y-axis is mirrored, i.e., negative numbers above the origin and positive numbers below. This is to visualize more intuitively that larger insertion loss means weaker transmission / less received power. So, the IL data in OMR files is in fact loss data, not transmission data and, correspondingly, CSV files used for IL de-embedding must contain loss data (positive values for attenuation, negative values for amplification).

NOTE

CSV files must be provided in plain ASCII format, i.e., without, for example, a UTF-8 header. When saving CSV file from Excel, please be aware that Excel may provide multiple CSV file formats (including CSV UTF8, depending on the Excel version). Be sure to use “CSV (Comma delimited)”, if available. Please note that the UTF-8 header (as well as other encoding headers) are usually not shown, when viewing the file. In case a CSV file causes trouble that might be caused by an invalid encoding, open Notepad.exe, load the offending file and save it using the “ANSII” option.

IL De-embedding Data Cache

For maximum evaluation speed, the LS engine stores internal files, containing IL de-embedding data from one or more source files, after merging and interpolation to target wavelength grid. Whenever the same list of source files is used later, for the same wavelength settings, the stored data is used instead of repeating merge and interpolation operations. The location on the disk drive that contains these internal files is called the IL De-embedding Data Cache.

Files in the IL De-embedding Data Cache are not deleted automatically. You can manually delete obsolete files by navigating to

C:\Users\All Users\Agilent\Photonic Application Suite\LS Application\ILDeembeddingDataCache

and delete files there.

NOTE File names are automatically generated and cannot be used to identify contributing IL de-embedding files or wavelength range settings. In case you are dealing with many different and large IL de-embedding files, consider a recurring cleanup operation of just deleting all the files in the IL De-embedding Data Cache. The LS engine will then create new files, whenever new configurations are being used.

NOTE Since the IL De-embedding measurements are not available in the PMD mode, the controls on the IL De-embedding tab are disabled in the PMD mode.

NOTE The fiber connections from the laser, through the polarization synthesizer to the switch should be stable before making the de-embedding calibration measurements on the switch. The stability of the connection from the laser to the polarization synthesizer is particularly important as a change in the polarization input to the polarization synthesizer between calibration steps can influence the results.

Related COM API

- property EnableILDeembedding
- property ILDeembeddingTargetFolder
- method StartILDeembeddingMasterPortMeasurement
- method StartILDeembeddingSpecificPathMeasurement
- method GetILDeembeddingFileList
- method SetILDeembeddingFileList

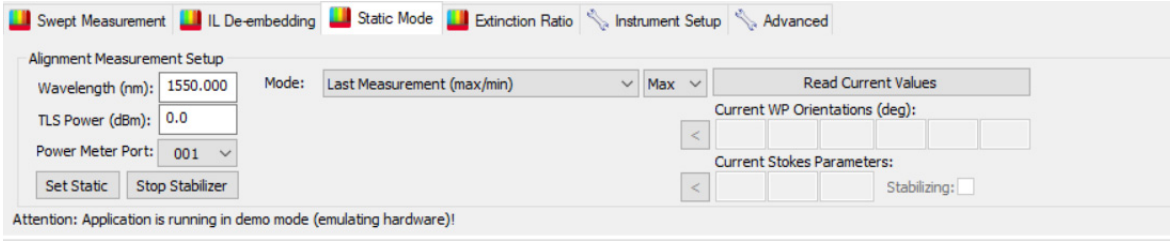
Performing Static Mode Measurements

While swept IL/PDL measurements are a quick and robust method for characterizing polarization-dependent and polarization-averaged transmission properties of passive optical devices and receiver sub-assemblies, sometimes specific custom measurements might also be required for certain devices.

To support this kind of scenario, the LS engine provides the Static Mode, which combines a variety of ways for setting a specific state of polarization at the DUT.

Using static mode, we extract polarization information right after performing a swept measurement or from previously saved measurement files.

The Static Mode is configured on a separate tab, the **Static Mode** tab, in the main GUI.



Static Mode operation is configured largely independent of any other LS engine measurement mode, except for selecting the TLS output port (in case there is more than one), which must be done on the Swept Measurement tab of the GUI.

The Static Mode parameters and buttons are explained in the table below.

Parameter	Description
Wavelength (nm)	Defines the wavelength to be set once the **Set Static** button is pressed. For the **Last Measurement** and **OMR File min/max** modes, it also defines the wavelength at which the measurement data is to be evaluated for min/max transmission or PSPs.
TLS Power (dBm)	Defines the optical output power of the laser source.
Power meter port	Defines the power meter port to be evaluated in OMR measurement data when using **Last Measurement** and **OMR File** modes. The drop-down menu lists power meter serial number and power meter port number for reference. When using **OMR File** mode, make sure to only use files that have been obtained using the same power meter configuration that is currently active.
Set Static	Activates Static Mode using the current settings. Unless Mode is set to **Waveplate Orientations**, **Set Static** starts the polarization stabilizer operation of the N7786B/C, with a target State of Polarization (SOP) based on the Static Mode configuration. With Mode set to **Waveplate Orientations**, stabilizer operation is neither required nor supported.
Stop Stabilizer	Clicking this button will make sure that the stabilizer operation of the N7786B/C is stopped, in case it is active. If clicked even when the stabilizer was not active, it has no effect, so it may always be used to make sure the stabilizer is stopped. Stopping the stabilizer after it has found the desired target SOP might be desired, to avoid mild SOP variation due to stabilizer dithering. But the SOP can then drift or change if the polarization at the N7786B/C input changes or due to temperature variation. It is not necessary to stop the stabilizer manually, before starting a swept IL/PDL measurement.
Mode	Defines the mode of operation for Static Mode. The different behavior of these modes and the corresponding GUI changes are described below this table.
Read Current Values	Clicking this button reads current SOP and current waveplate orientations, as well as the current stabilizer state. Details are described below.

Static Mode Modes

Consider the following aspects when using Static Mode in general:

- It is useful, although not mandatory, to use a PMF between TLS and N7786B/C and it is highly recommended to use an SMF between N7786B/C and DUT, except for PMF that is part of the DUT.

Consider the following aspects when using any Mode other than Waveplate Orientations:

- Once **Set Static** is clicked, the polarization optimization is started. The engine / GUI remains busy until the target SOP has been reached or an unexpected event has occurred, such as an instrument timeout or an error caused by a too low input power level. After the SOP has been found, the stabilizer operation continues, until manually stopped or until another kind of measurement is performed from within the LS engine.
  This can be used both for performing additional, and then non-polarization resolved, swept wavelength measurement with the LS engine or for running custom measurements and operations that benefit from the input polarization to the DUT being aligned for maximum or minimum transmission.
  Keep in mind, though, that the stabilizer operation employs polarization dithering, so manually stopping the stabilizer might be helpful in certain situations.

NOTE Once Set Static operation has been performed, any subsequent swept measurement requires some re-initialization and, thus, may take some additional time.

- Stokes parameters are defined relative to the reference system at the polarimeter inside the N7786B/C, not relative to any DUT main axes or geometries.
- Polarization stabilization works based on SOP measurements within the N7786B/C. Therefore, polarization changes in front of the N7786B/C can be compensated for, but polarization changes between N7786B/C and DUT cannot. Accordingly, connecting fibers should be fixed or taped in place and should not be subject to movement/vibrations or temperature variation.
- When using **PDLPSP** mode (as described below), assignment to PSP1 or PSP2 is arbitrary to some degree. In case of stable DUT and setup and when using the same reference, association should be persistent between measurements, but in case any of these requirements do not apply, the assignment may swap between measurements. Assignment will never toggle within a single measurement sweep, but PSP orientation may gradually transform over wavelength, in case there is strong birefringence in the measurement Setup or DUT (with main axes other than the PDL main axes).

Consider the following aspects when using Waveplate Orientations Mode:

- Using **Waveplate Orientations** Mode, there is no stabilizer operation at all. This is a pure feed-forward operation. For example, if the SOP input to the N7786B/C changes, the output SOP will also change.

The different available modes are described in the following sections.

Last measurement (min/max) or (PDLPSP)

When Mode is set to **Last Measurement**, a swept IL/PDL measurement must first have been performed since starting the LS engine.

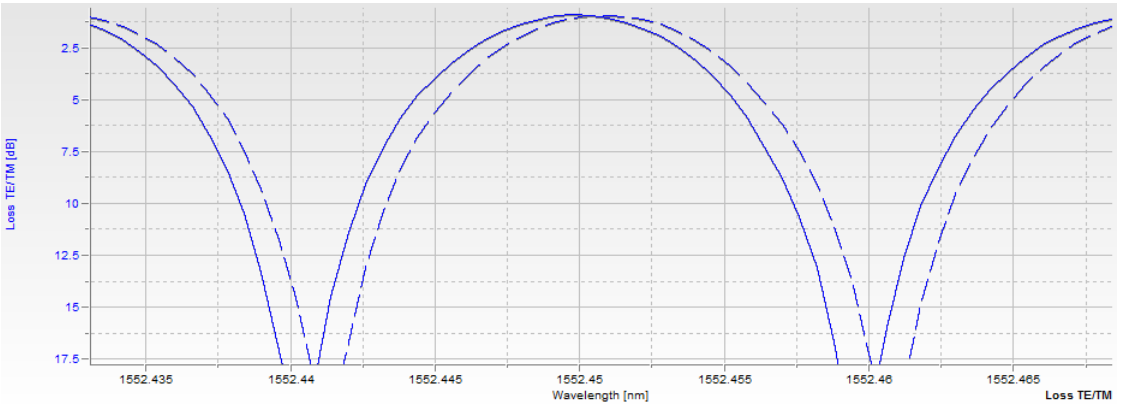
Once **Set Static** is clicked, the current measurement data, i.e., data from the most recent measurement sweep, is analyzed on the selected port and the resulting SOP is set as target SOP and polarization stabilizer operation of the N7786B/C is started.

In **min/max** mode, based on the min/max selector, the corresponding SOP is extracted, to provide minimum/maximum transmission at the selected wavelength.

In **PDLPSP** mode, the complete measurement range is analyzed in the same manner that leads to TE/TM traces in a regular IL/PDL measurement; an appropriate wavelength region of high DUT PDL is selected automatically, then the corresponding principal state SOPs are extracted and applied based on the PSP1/PSP2 selector.

NOTE Measurement traces are named TE/TM for distinction. They do not necessarily refer to physical definition of TE/TM, but might be swapped instead.

The difference between TE/TM and min/max traces is most prominent in certain filter devices, where there can be a wavelength-shifted filter response for two orthogonal polarizations. This filter shift is, obviously, discernible when sequentially sweeping the wavelength after aligning the input polarization to the DUT's principal axes. It is also obtainable from a single-sweep IL/PDL measurement (TE/TM trace), as shown in the graph below. As the graph indicates, using PDLPSP mode will use SOPs matching the DUT's main axes and, thus, return maximum or minimum transmission, based on the chosen wavelength. Using min/max mode, on the other hand, would align to either of the main axes, when far from the filter's passband, but might align to some average SOP close to its passband center, where the response curves cross.



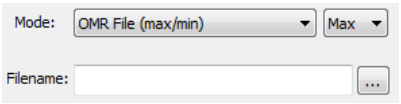
NOTE Last Measurement Mode requires Mueller data to be present in the measurement data.

NOTE It is possible to run a non-polarization resolved measurement after using the Last Measurement Mode to set the SOP and perform a swept measurement at this SOP. However, this will replace the last measurement in memory, with a non-polarization-resolved measurement, causing the Last Measurement Mode to be inapplicable. In such a scenario, it is recommended to run a polarization-resolved measurement, save it to an OMR file. then use the OMR File Mode instead and select the previously saved file.

OMR File (min/max) or (PDLPSP)

This Mode is basically identical to Last measurement (min/max) or (PDLPSP) on page 116, except that it does not extract the min/max/PSP information from the current measurement, but rather from a saved file.

When this Mode is selected, there will be additional controls for either typing or copy/pasting a filename (including full path and file extension) of an OMR file, or opening a file selection dialog to choose such an OMR file.

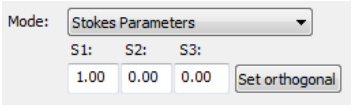


NOTE Make sure to only use files that have been obtained using the same power meter configuration that is currently active.

NOTE **OMR File** Mode requires Mueller data to be present in the measurement data.

Stokes Parameters

In **Stokes Parameters** Mode, the target SOP is provided using its normalized Stokes parameters.

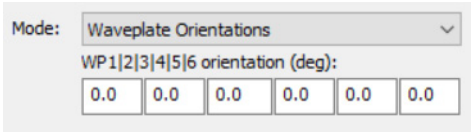


These parameters can be obtained, e.g., from polarimetric measurements using the N7786B/C web-based graphical user interface or SCPI automation or using the Get Current Values operation described in Current Value Operations on page 119.

Use the **Set Orthogonal** button to switch to the orthogonal SOP, for example, after having manually optimized for one of the DUT's main axes before.

Waveplate Orientations

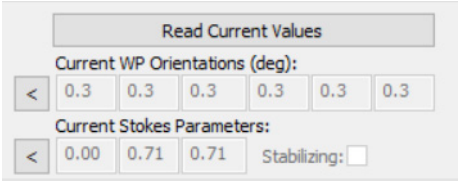
In **Waveplate Orientations** Mode, the target orientations for the six equivalent waveplates of the N7786C instrument (five equivalent plates in case of the N7786B instrument), respectively, are provided in degrees.



NOTE There is no polarization stabilization involved when using **Waveplate Orientations** Mode.

Current Value Operations

To check the current SOP or waveplate settings, as well as the current polarization stabilizer state, click the **Read Current Values** button.



Current values may result either from a previous Set Static operation or an operation performed independent of the LS engine, for example, by using the N7786B/C web-based graphical user interface or SCPI automation. After reading the current values, the buttons to the left of the respective waveplate or Stokes values can be used to make these values the new target values and switch the Mode to **Waveplate Orientations** or **Stokes Parameters**, respectively.

NOTE Since the Static Mode measurements are not available in the PMD mode, the controls on the Static Mode tab are disabled in the PMD mode.

Procedure Summary

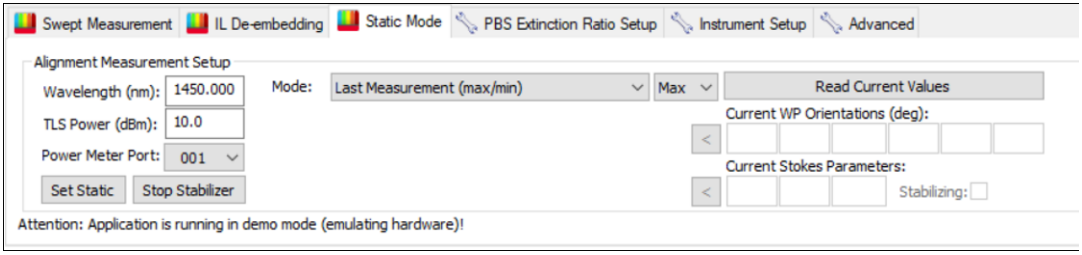
Ensure that the measurement is started in stable temperature environment and that the instruments should have been powered up for 20-30 minutes prior to the measurement.

1 Open the Configuration Wizard to create a measurement configuration including the required instruments or load an existing agconfig file. Refer to the Configuring Instruments Using Configuration Wizard on page 66 for instructions on selecting instruments using the Configuration Wizard.

2 Connect the DUT to the polarization synthesizer on one end and the power meter on the other.

3 In order to use Last Measurement mode below, at this point a measurement needs to be performed.

4 Under the Static Mode tab, specify values for the following fields:



1 TLS Wavelength

2 TLS Power

3 Power Meter Port on which the measurement is to be performed

4 Mode – Select one of the following:

- Last Measurement (max/min)
- Last Measurement (PDLPSP)
- OMR File (max/min)
- OMR File (PDLPSP)
- Stokes Parameters
- Waveplate Orientations

5 Click the Set Static button on the Static Mode tab.

6 The engine might prompt you in case there has been a considerable time lapse since the last TLS zeroing operation. Click Yes to instruct the LS engine to perform TLS zeroing now. You might also want to refer to TLS Lambda Zeroing on page 139 for more information.

7 Click the Stop Stabilizer button to stop the measurement, if required.

8 Perform the desired custom (non-LS-Engine) measurements / operations.

NOTE

After using Static Mode and before performing the next polarization-resolved swept-wavelength measurement, it is recommended to either set all the waveplate orientations to zero (using Set Static in Waveplate Orientations mode), or to repeat the swept measurement once. This is intended to avoid any potential measurement artifacts that might arise from an arbitrary polarization state at measurement start.

Related COM API

- method SetStatic
- method StopStabilizer
- property StaticWavelength
- property StaticTLSPower
- property StaticPWMPort
- property StaticPolarizationMode
- property StaticPolarizationMaxMinMode
- property StaticPolarizationPSPMode
- property StaticFilename
- property StaticStokesParameters
- property StaticWaveplateOrientations
- method ReadCurrentStokesParameters
- property ReadCurrentWaveplateOrientations
- method ReadCurrentStabilizerState

PMD Mode

The Lambda Scan engine supports PMD measurements using N7788C Optical Component Analyzer. You must have the following license for using the PMD measurements.

- N7700103C PMD

NOTE Unlike Polarization Navigator, the LS Engine does not support N7788B/N7788BD component analyzers for the PMD measurements.

In the swept, polarization-resolved measurements, a continuous sweep of the tunable laser source (TLS) is used to measure the following parameters versus wavelength:

- Polarization Dependent Loss
- Insertion loss (polarization-averaged or at fixed polarization)
- Mueller data
- DGD, second-order PMD (denoted PMD2nd)

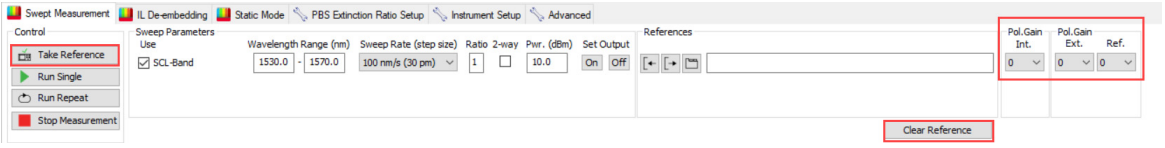
Measurement setup

See Measurement Setup for PMD Mode on page 20.

Setting the PMD Mode parameters

Before performing the measurements, you must set up the application parameters appropriately on the Swept Measurement tab, Instrument Setup tab, and Advanced tab.

The following image displays the Swept Measurement tab when you have configured your hardware setup for PMD measurements,



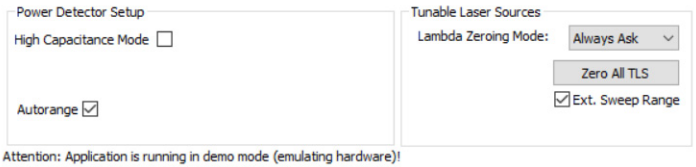
For PMD measurements, the parameters on the **Swept Measurement** tab are similar to those for non-PMD measurements, except for the following differences:

- There are no check boxes for enabling the following supplementary measurements:
  - Dark current
  - PBS Extinction Ratio

- Additional **Pol. Gain Int.** and **Pol. Gain Ref.** controls are available to configure gain setting for polarimeter on internal path and during reference measurement, respectively. The reason for these parameters to be configured independently is to avoid autoranging steps, if required. The internal path is used when performing the internal reference sweep with the optical switch inside the N7788C. For the other switch path ("external"), the engine distinguishes between the actual device measurement (Pol Gain Ext.) and the reference measurement (Pol Gain Ref.), in which the polarimeter usually is subject to a higher optical power level than during device measurements.
- The **Port/Ref. Manager** button is replaced with **Take Reference** button as PMD mode supports single port operations only.
- **Clear Reference** button for clearing the current reference.
- The Return Loss Range controls are not available in the PMD measurements mode.

For information on the Swept Measurement parameters for the non-PMD measurements, refer to Application Setup Parameters on page 54.

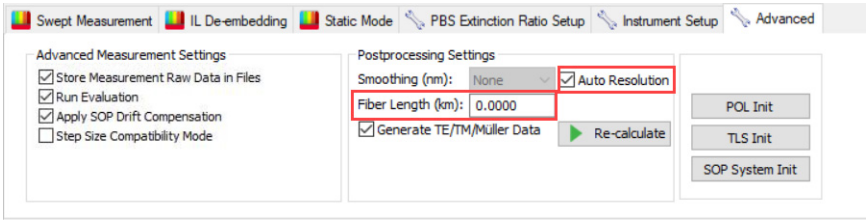
For PMD measurements, the parameters on the **Instrument Setup** tab are similar to those for non-PMD measurements, except for the following differences:



- As neither of the related instruments is supported in the PMD mode, the following controls are not available:
  - Bias settings
  - Return Loss calibration controls
  - Use MPPM auto gain check box
  - Run Zeroing button

For information on the Instrument Setup parameters for the non-PMD measurements, refer to Instrument Setup Parameters on page 58.

For PMD measurements, the parameters on the **Advanced** tab are similar to those for non-PMD measurements except for the following differences.



- Additional **Fiber Length** parameter is available. This is used for computing the PMD coefficient in ps/sqrt(km) from a DGD-over-wavelength measurement.
- The **Smoothing** parameter, in addition to its function of smoothing the measurement data (as in case of non-PMD mode), also determines the data processing step size in PMD mode, which should be chosen relative to the amount of DGD/PMD present. Hence, it is strongly recommended to use the Auto Resolution setting instead.
- Additional **Auto Resolution** check box is available. When enabled, the data-processing resolution (smoothing) parameter will automatically be chosen based on the coarsely estimated DGD/PM of the device under test.
- Only Generate TE/TM/Muller Data check box is available. The following features/check boxes are not available (as none of the related instruments is supported in PMD mode):
  - Generate Diode Current Data
  - Generate Diode Responsivity Data
  - Generate Return Loss Data
- The PWM Init button is not available.

For information on the Advanced parameters for the non-PMD measurements, refer to Advanced Parameters on page 61.

NOTE

Since the Extinction Ratio, Dark Current, Static Mode, and IL De-embedding measurements are not available in PMD mode, all controls on the following tabs are disabled in PMD mode:

- IL De-embedding
- Static Mode
- PBS Extinction Ratio Setup

Performing Measurements

Connect a patchcord to the DUT ports, enter the desired settings under the Swept Measurement, Instrument Setup, and Advanced tabs, and perform a reference measurement by clicking **Take Reference**.

Then connect the DUT and click **Run Single** or **Run Repeat**. A referenced measurement requires the reference sweep to cover the whole range of the DUT sweep, therefore the measurement parameters should be entered before taking the reference, or the reference sweep chosen should be large enough to be able to vary the DUT sweep wavelengths later on.

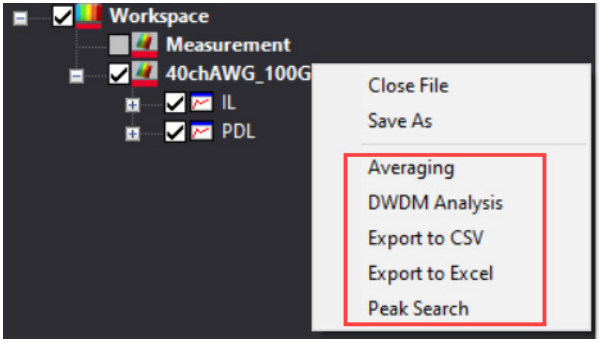
NOTE Though it is possible to run referenced measurements with sweep settings that differ from reference sweep settings, the best results can usually be achieved if the two sets of settings are identical.

You can save measurement data using the **Save as** option from the File menu and providing a filename. Saved files can be accessed through the **Open...** dialog from the File menu, by clicking Open or by opening the corresponding file using the Windows Explorer.

Measurement data can be exported to different file types (e.g. CSV).

Applying Plugins

You can apply evaluation plugins to the current measurement, e.g. averaging, peak-search or DWDM channel analysis functions. The plugins are listed in the measurement context menu in the browser tree. Which plugins are available depends on the license(s) that are installed on your system. For more information about the available feature packages and licenses, refer to Keysight N7700 Photonic Application Suite - Brochure. Some plugins directly modify the traces of the measurement, some add more traces, and some add information to the table on the bottom of the user interface.



You might also consult *The Plugin System* chapter in the main *Photonic Application Suite User Guide* to get more information about the plugins.

Recalculating Raw Data

Prerequisites

The LS engine must be running to perform the recalculation on raw data. No license is required for performing recalculation. Ideally, an agconfig file that matches the configuration with which the raw data file was initially obtained, is used, but in case no adjustments to data processing parameters are intended, it should be enough to just start the Configuration Wizard and start the demo mode to have an Engine running for recalculation operations.

Handling agconfig Files

An agconfig file can changed to simulation mode (e.g. to be used on a different computer) by replacing the string `<Item Name="Emulation" Type="bool" Value="false"/>` to `<Item Name="Emulation" Type="bool" Value="true"/>`, using a text editor, such as notepad.exe.

NOTE When modifying the default agconfig file (C:\Users\Public\ Documents\ Photonic Application Suite\KtPasEngineLS.agconfig), make sure that the engine is deactivated/closed before editing the file, as it would otherwise get overwritten.

Loading/activating this modified configuration file will now look almost identical to the original configuration, but will not connect to any instrument.

When modifying agconfig files for use on the same computer that is also running the live measurements, ensure that neither an LS Engine GUI, nor the corresponding KtPasServer executable is running, when modifying the files.

In case you have both an OMR and an lsraw file, and you intend to do some recalculation with adjusted settings, you can extract the agconfig from the omr by following the steps described in Creating .agconfig Files from Measurement Files on page 131, then change that agconfig to simulation mode as described above.

In case you only have an lsraw file, the above process can still be applied by recalculating the file once (using the settings stored in the lsraw file), then saving the resulting omr. Now the agconfig can be extracted as explained above.

NOTE

While many operations are simulated for an agconfig file that has been modified as described above, running simulated measurements like this is not a supported mode of operation, as there may be unexpected behavior for certain instruments/settings/operations. This approach is only recommended to run offline data processing of lsraw files.

NOTE

Whenever switching between simulated and live operation on your own agconfig files, make sure to stop both LS Engine GUI and KtPasServer executable before switching from one to the other, as the server would still assume the instruments to be either simulated or live, based on whatever agconfig comprising a certain instrument was activated first. The demo mode agconfig files, provided with the LS engine installation are not affected by this, as their virtual instruments are simulated with USB, GPIB, and TCP descriptors that should not occur in any live setup.

Performing the Recalculation on the Raw Data

Recalculating using the File Menu Option

The saved raw data file can be reloaded from the File menu. Click File > Load and Evaluate Measurement Raw Data and select a measurement raw data file. This will cause immediate recalculation (or first time data processing, in case no .omr file was generated / saved during the measurement) and add it to the OMR measurement tree on the left.

All settings relevant for data processing are taken from the saved file. Especially in case of IL De-embedding, this might cause recalculation to fail, as the respective IL De-embedding OMR files might not be present at all or at least not within the expected folder.

Recalculating using the Re-calculate Button

Clicking the Re-calculate button under the Advanced tab will then perform another data processing operation on the most recently loaded raw data file and replace the top item in the omr measurement tree (where there is usually the live measurement).

In this operation, the engine will apply as many settings from the current GUI as possible. This includes IL De-embedding settings from IL De-embedding tab and Port/Ref Manager, data processing options from the Advanced tab, and Wavelength start/stop/step settings from the Swept Measurement tab.

Many current GUI settings will not have any effect, though, as parameters, such as TLS power or power meter ranges and averaging times, only have an impact on instrument configuration, not on processing the measurement raw data.

The Wavelength Start, Stop, and Step parameters assume a special role here, as these are used in instrument configuration, and also for defining the desired wavelength target grid that results are being interpolated to. So, as long as the new start and stop wavelength lie within the wavelength range covered by the original measurement, these can be changed subsequently.

Recalculating on Raw Data from Multi-TLS Configurations

Especially when performing recalculation with changed settings on raw data files obtained in multi-TLS configurations, special care is required.

- First, the current configuration must comprise at least as many TLS as were present while obtaining the original raw data file, as otherwise, sweep range adjustments cannot be applied for some TLS.

- Second, if there were, e.g., three lasers in the original configuration, with only the first and the third selected, then recalculation will use sweep settings from current configuration's first and second TLS only and ignore settings for the third TLS. As, usually, the lasers span completely different wavelength ranges, it might be necessary, e.g., to enter L-band wavelength settings in the O-band TLS controls, for the recalculation to work as expected. This is perfectly fine for recalculation case, but would, definitely, cause issues when switching back to live measurements when there are now L-band wavelengths configured for O-band TLS.

Similarly, but more straightforward, the configuration used for recalculation should contain as many (or more) power meter / SMU ports as the original measurement configuration, in case modifications to IL De-embedding settings are intended.

NOTE

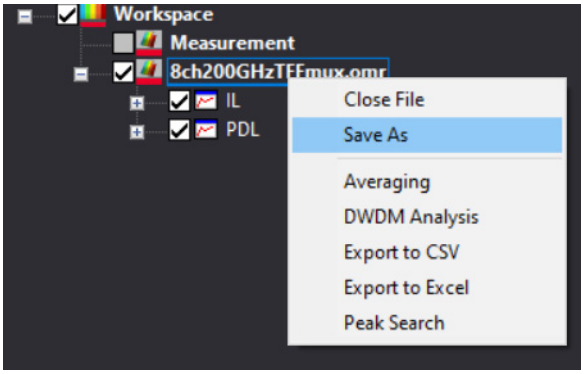
While it is not strictly necessary to use a matching configuration (same number of TLS instruments, power meter / SMU ports, presence of return loss module), it is recommended to use the same configuration. This has already been motivated above, but it is also relevant, e.g., when the lsraw contains return loss data, but the current engine configuration used for recalculation does not include a return loss module, as then the recalculation will not generate the return loss trace.

NOTE

For demonstration purposes or basic API programming work, it is recommended to start the Configuration Wizard and click the "Demo mode" text on the initial dialog page. This will start the engine with a certain (pre-tested) instrument configuration.

Saving Measurements

You can save measurement data using the Save as option from the measurement context menu in the browser tree and providing a filename. Alternatively, select **File > Save Measurement As...** The measurements are saved as .omr files, which can be opened with the same N7700 engine, with the N7700 File Viewer, or with user programs using the N7700 OMR File Handler library.

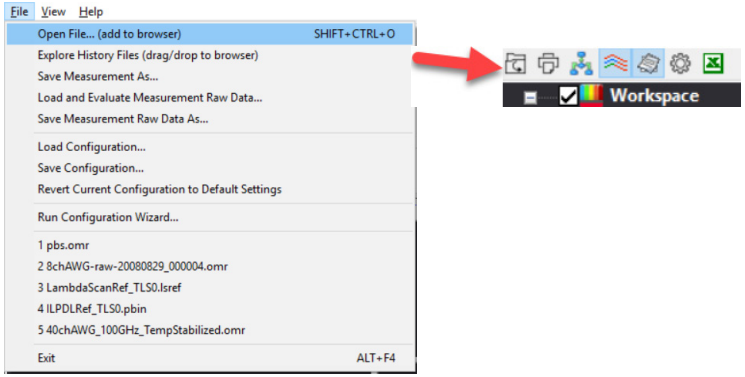


NOTE

The **File > Save Measurement Raw Data As...** menu option enables you to save raw data associated with measurements as *.lsraw* files, which can be used for troubleshooting by Keysight support.

Loading Measurements

Saved files can be accessed through the Open File dialog from the File menu or by clicking the Open icon above the browser tree. Open files will be listed in the browser tree to the left of the user interface.



Creating .agconfig Files from Measurement Files

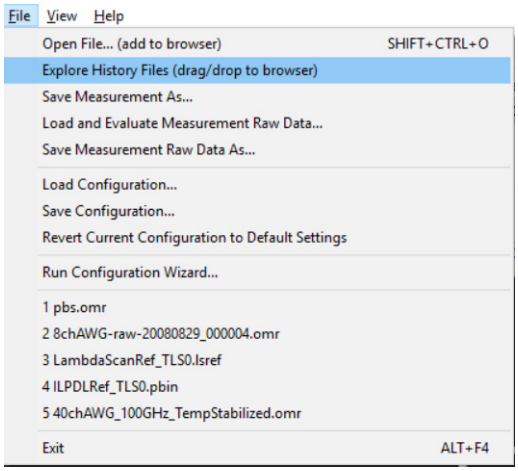
You might require the .agconfig file for a particular measurement that you have performed in the past. In case you did not explicitly save the .agconfig file, you can extract it from a measurement (.omr) file by using ExtractAgconfigFromOmr, a PowerShell utility, available in the Examples folder (C:\Program Files\Keysight\Photonic Application Suite\Examples\DemoPowerShell). This utility takes an OMR file as an input and creates an agconfig file that reflects the settings with which the specified measurement had been performed. The extracted .agconfig file has the same name as the specified OMR file and is created in the same directory as the OMR file. For easy access and execution of this PowerShell utility, use the batch file ExtractAgconfigFromOmr.bat specifying the OMR file as a parameter (either specify the full path of the OMR file or place it in the current working directory).

This utility can also be used for troubleshooting purposes.

Automatic History Saves

By default, the measurement engine will automatically save a number of recent measurements (.omr) files to a specific folder. You can configure automatic saving of N number of raw data (.lsraw) files also. The capability to save .omr and .lsraw files is a very helpful feature, since one often performs a number of measurements that are not intended for saving, then realizes, after a certain, subsequent measurement, that comparison with one of the previous measurements would indeed reveal certain insights.

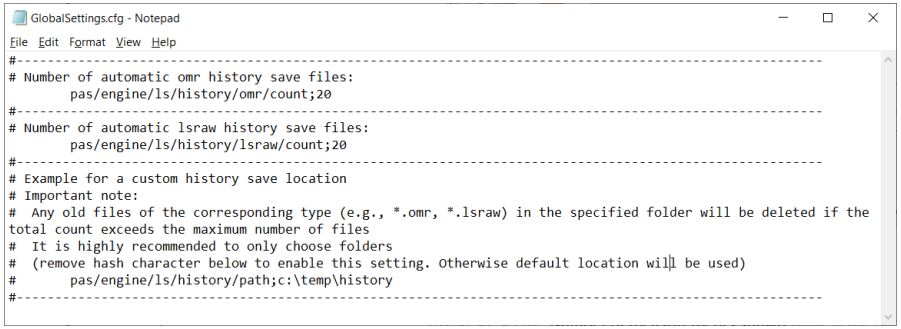
Selecting **Explore History Files** from the File menu will open this engine's history file save folder. The files are named using a time stamp of the measurement. Use drag & drop to add any file(s) to the engine's browser tree or double-click any file(s) for display in the File Viewer application.



Especially when working with large amounts of measurement data (many ports, broad range, high resolution), you may want to reduce the number of automatically stored files or even turn off the automatic saving of history files. On the other hand you may want to increase the number for obtaining many measurement/raw data files without having to perform manual saves, or having to use measurement automation.

Automatic history saves (OMR files) and automatic lsraw file saves are performed by the engine server, and not by the engine GUI. The earlier Number of measurements kept and Number of lsraw files kept controls have been removed from the Property Editor or the Settings dialog. Instead, use the Keysight N7700 Global Settings configuration file available from the Start menu. When clicked, it will get opened in Windows' Notepad for editing. You can edit this file in Notepad.exe or any other text editor to set the number of automatic .omr history save files, the number of automatic .lsraw history save files and the target folder that both of these will be stored in.

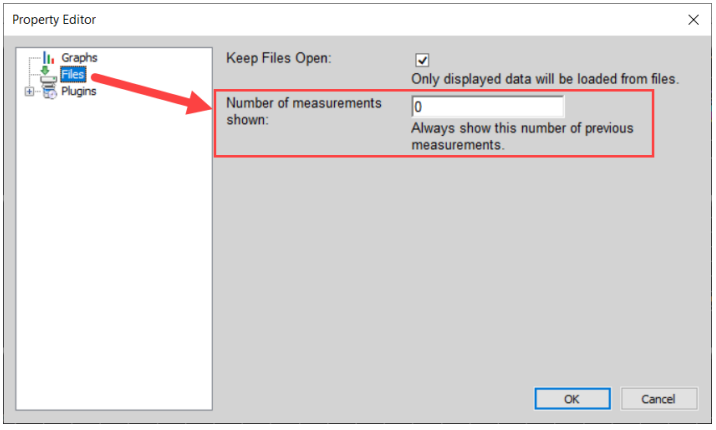
Set the **Number of automatic omr history save files** value and **Number of automatic lsraw history save files** to 0 to turn off automatic saves.



When tuning a device or a component, or monitoring stability, it might be desirable to automatically have a certain number of most recent measurements displayed simultaneously.

To do so, set the **Number of measurements shown** value in the options to the desired value or to 0 to disable this feature.

You can do so by modifying the corresponding value in the options:



NOTE

Number of automatic omr history save files must be larger than or equal to **Number of measurements shown**, for this feature to work as intended.

NOTE

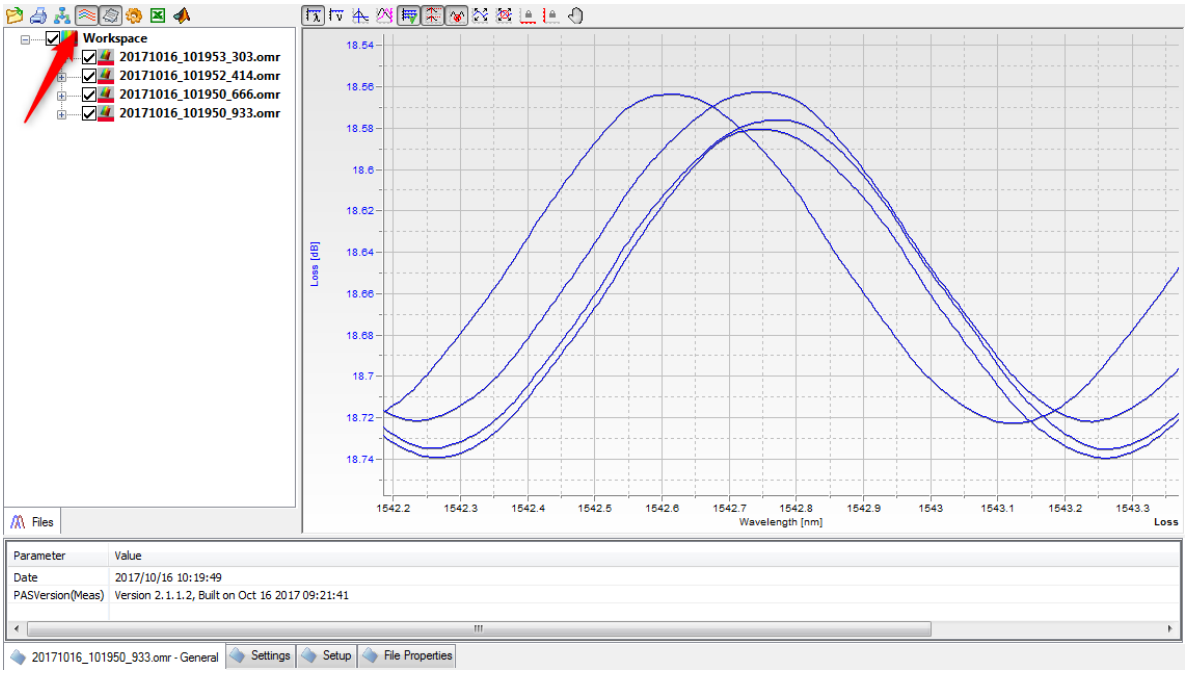
Number of measurements shown is a global setting used for all Photonic Application Suite engines.
Number of automatic omr history save files settings in the GlobalSettings.cfg file, as described above, is applicable to the LS engine only. For the other, legacy measurement engines, the functionality of this setting can be realized by using the **Number of measurements kept** setting in the Property Editor.

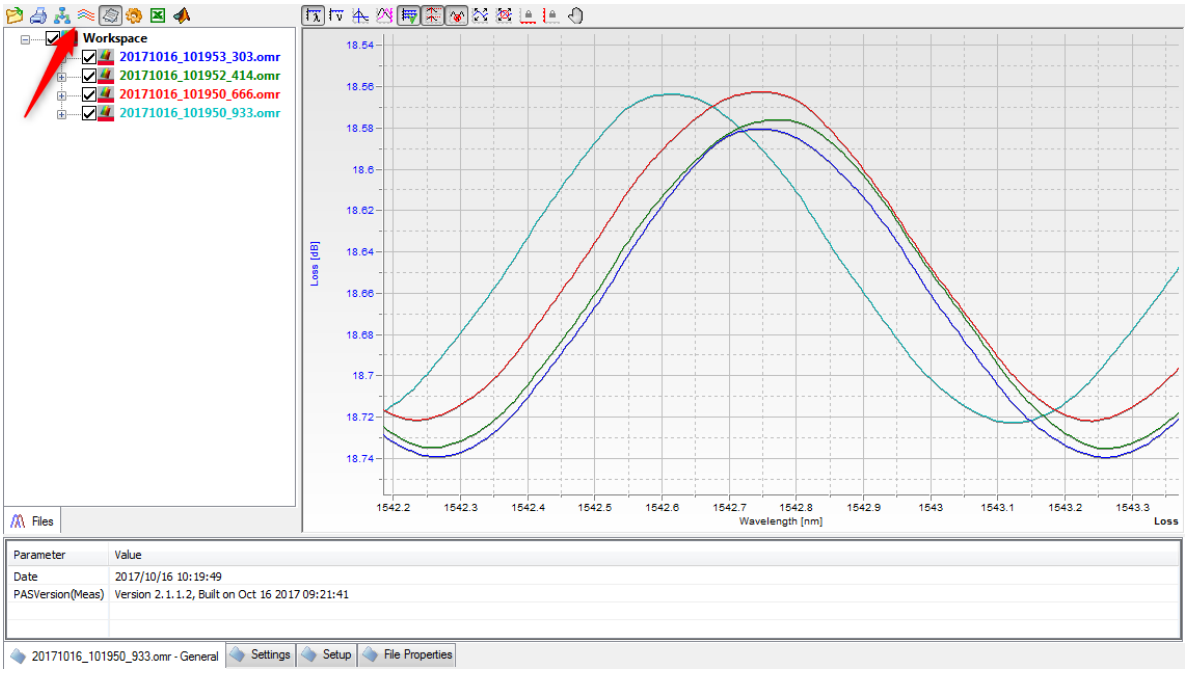
NOTE

The **Number of measurements shown** setting affects measurements performed after starting the engine, after changing this setting, or after closing all the open files. It does not open any older files, so after any of the mentioned operations, the number of automatically shown history files will grow with each new measurement until the total number reaches the **Number of measurements shown** value. For each new measurement after that, the oldest history file will be closed automatically.

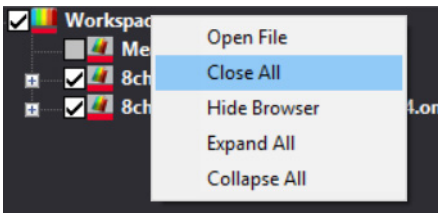
By default, trace colors indicate different ports, but by toggling the **Color by Channel/File** icon, this can be changed to indicate different measurements instead. This is usually desired, when using automatic history file display with single-port devices, or devices, where individual ports are clearly distinguishable.

See examples for both modes below.


Workspace
20171016_101953_303.omr
20171016_101952_414.omr
20171016_101950_666.omr
20171016_101950_933.omr
Files
Loss [dB]
18.54
18.56
18.58
18.6
18.62
18.64
18.66
18.68
18.7
18.72
18.74
1542.2
1542.3
1542.4
1542.5
1542.6
1542.7
1542.8
1542.9
1543
1543.1
1543.2
1543.3
Wavelength [nm]
Loss
Parameter
Value
Date
2017/10/16 10:19:49
PASVersion(Meas)
Version 2.1.1.2, Built on Oct 16 2017 09:21:41
20171016_101950_933.omr - General
Settings
Setup
File Properties



When either replacing the DUT, introducing a significant change to DUT tuning parameters, or running a specific path de-embedding measurement, corresponding measurement results may vary significantly from previous history files. To remove all the old measurements from the Workspace at once, click **Close All** from the context menu of the Workspace node in the tree on the left of the GUI.



NOTE The location for saving the history files can be specified in the GlobalSettings.cfg file available from the Start menu.

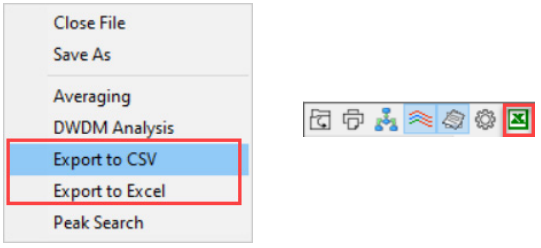
Exporting Measurement Data

Measurement data can be exported to different applications or file types (e.g. Excel, CSV).

You can export the current measurement by:

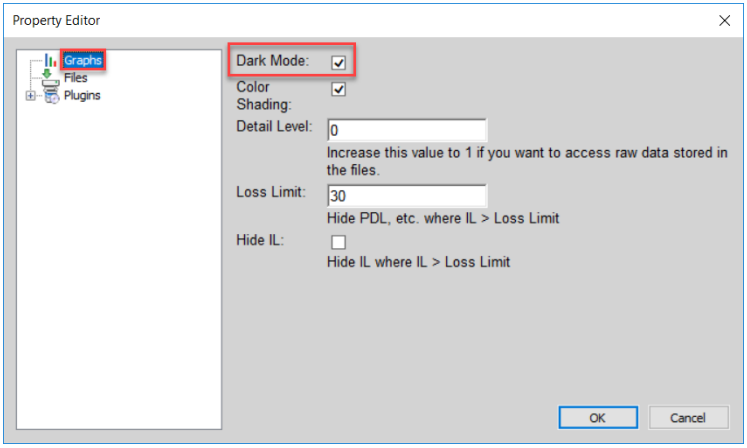
- Selecting the corresponding entry of the measurement context menu in the browser tree.
- Selecting the corresponding icon above the browser tree.

Measurement raw data will also be exported to CSV even if DetailLevel is set to 0 in the GUI.



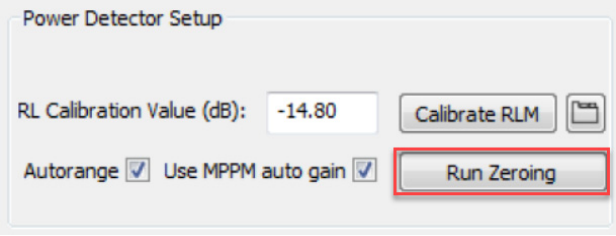
Changing the Color Theme

You can switch to a lighter color theme for the File Viewer within the user interface of LS Engine. Click the Settings icon and disable the Dark Mode check box in the Graphs section.



Power Meter / SMU Zeroing

Use the Run Zeroing button on the Instrument Setup tab to zero all ports of all connected power meter / SMU instruments, as well as an 81610A or 81613A return loss module, if used. You will be prompted to cover all detectors before proceeding. The TLS output is turned off automatically during zeroing. Optimum dynamic range and sensitivity is achieved, after power meters / SMUs have been zeroed appropriately. The instruments should have reached stable operating temperature conditions for best results.



You can perform the zeroing operation for one or more ports of the power meter / SMU using the Port/Reference Manager. See “Performing a Reference Measurement” on page 70.

TLS Lambda Zeroing

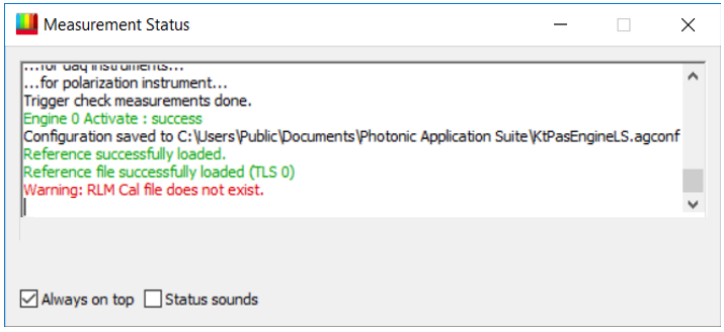
In order to ensure optimum wavelength sweeps through varying ambient conditions, a TLS lambda zeroing function is available.

As a practice, TLS zeroing or lambda zeroing must be performed at least once a day. The TLS zeroing operation might take several ten seconds to complete. To minimize the impact on running measurements, but keeping up the wavelength accuracy, choose one of the three settings for the Lambda Zeroing Mode under Instrument Setup tab - Always Ask, Automatically, and Manual Only. See Instrument Setup Parameters on page 58.

NOTE **For maximum wavelength accuracy, a TLS lambda zeroing operation should be performed, once the instrument reaches stable operating temperature conditions.**

Return Loss Calibration

Besides characterizing a device for IL/PDL, you might need to characterize the device for return loss as well. You can use a supported return loss module for that. However, you must perform a return loss calibration for the return loss module before proceeding to the return loss measurement. Or, a valid return loss calibration file must exist before performing the return loss measurement. Otherwise, you get the following message when trying to perform a return loss measurement:



By default, the return loss module will be operated using some default calibration parameters (wavelength-independent), not yielding optimum accuracy and can only serve as a rough estimate.

For accurate return loss measurements, it is mandatory to perform calibration steps for the return loss module. This can be done using a calibrated or known reference reflector such as the 81610CC listed in the RL module user's guide.

To perform a return loss measurement/calibration, you must include a return loss module in the Configuration Wizard besides the tunable laser source power meter.

Required instruments

- One return loss module
- One or more tunable laser sources
- Optional: polarization synthesizer
- One or more power meters and / or SMUs

Prerequisite

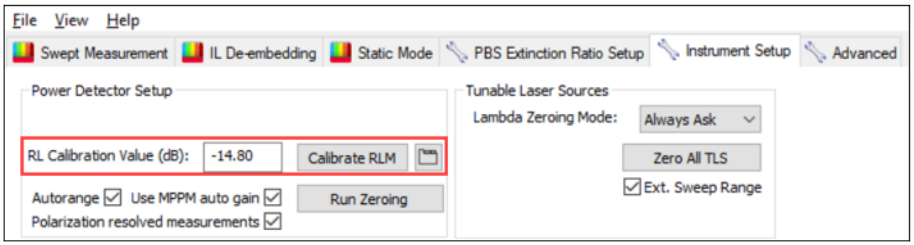
- Zeroing has been performed for TLS and return loss module. (Return loss module zeroing is performed by clicking the Run Zeroing button in the Power Detector Setup section on the Instrument Setup tab. See Power Meter / SMU Zeroing on page 138)
- Calibration artifact (Reference Reflector) is available.

Procedure

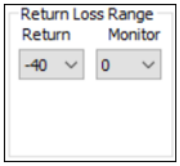
Ensure that the measurement is started in stable temperature environment and that the instruments should have been powered up for 20-30 minutes prior to the measurement.

1 Open the Configuration Wizard to include the requirement instruments in the measurement setup or load an existing agconfig file. Refer to the Configuring Instruments Using Configuration Wizard on page 66 for instructions on selecting instruments using the Configuration Wizard. Note that the inclusion of a return loss module in the configuration

wizard makes the following controls visible under the Instrument Setup tab.



Also, the Swept Measurement tab shows the Return Loss Range parameters as shown below:



2 Under the Instrument Setup tab, set the RL Calibration Value (dB) to match your reference reflection artifact. This value can be had from the label on the 81610CC reference reflector.

3 Click Calibrate RLM. The return loss calibration starts.

4 The engine might prompt you in case there has been a considerable time lapse since the last TLS zeroing operation. Click Yes to instruct the LS engine to perform TLS zeroing now. You might also want to refer to TLS Lambda Zeroing on page 139 for more information.

5 Connect the calibration artifact to the output of the return loss module. If the calibration artifact reflection value is not what you specified in Step 2, cancel the calibration and change value. And, start the calibration again. Otherwise, click OK in the Measurement Status window.

6 After you get a message for the completion of the return loss calibration artifact measurement, terminate the fiber after return loss module output, when prompted.

7 Click OK. Upon successful completion, you should get the following message:

Return loss termination measurement done.

Calibration will be performed for all the configured TLS no matter whether they are checked in the GUI or not. The calibration range will cover the selected sweep range of the laser sources, so it should be performed using a reasonably large range in order to be able to use the calibration file for various portions of the total wavelength range.

The calibration data will be saved automatically and applied to all subsequent measurements until a new calibration is performed or the calibration data is deleted. The engine will show a message in the measurement status window, telling the user whether a calibration file has been used or not.

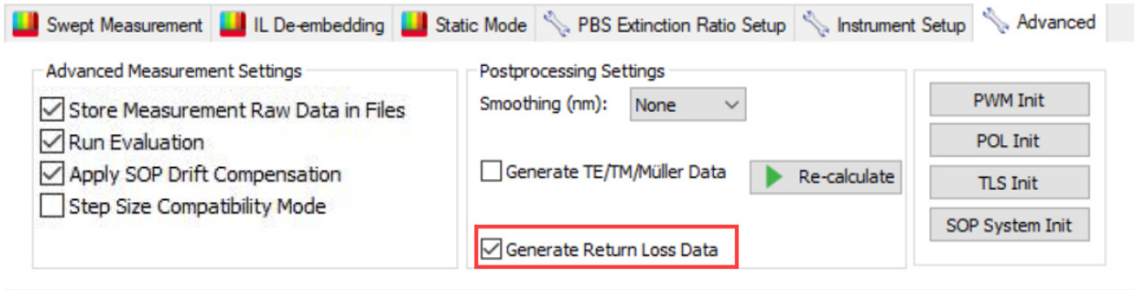
You can also delete the return loss module calibration file using the Delete/Clear Calibration File button. This will return the configuration to an uncalibrated state with respect to return loss.

Performing a Return Loss Measurement

1 Perform a reference measurement. See Performing a Reference Measurement on page 70.
Alternatively, load a reference file.

2 Connect the DUT to the return loss module, on one end, and the power meter on the other.

3 Ensure that under the Advanced tab, the Generate Return Loss Data check box is selected.

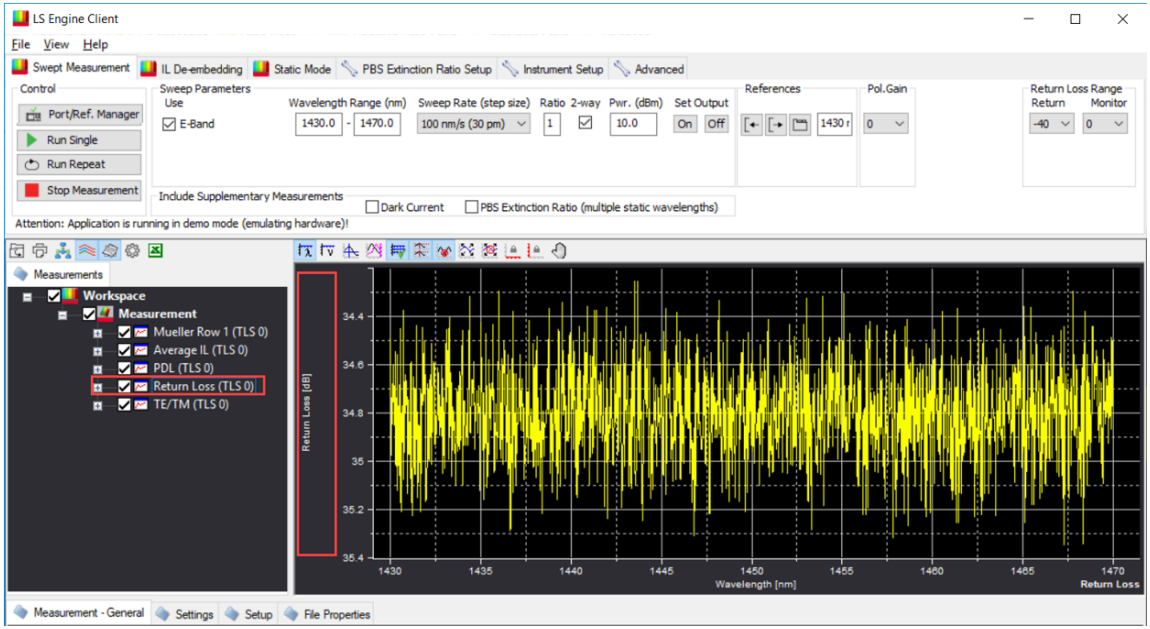


4 Click the Run Single button on the Swept Measurement tab.

5 The engine might prompt you in case there has been a considerable time lapse since the last TLS zeroing operation. Click Yes to instruct the LS engine to perform TLS zeroing now. You might also want to refer to TLS Lambda Zeroing on page 139 for more information.

Results

- A return loss trace in the IL or IL/PDL measurement results.

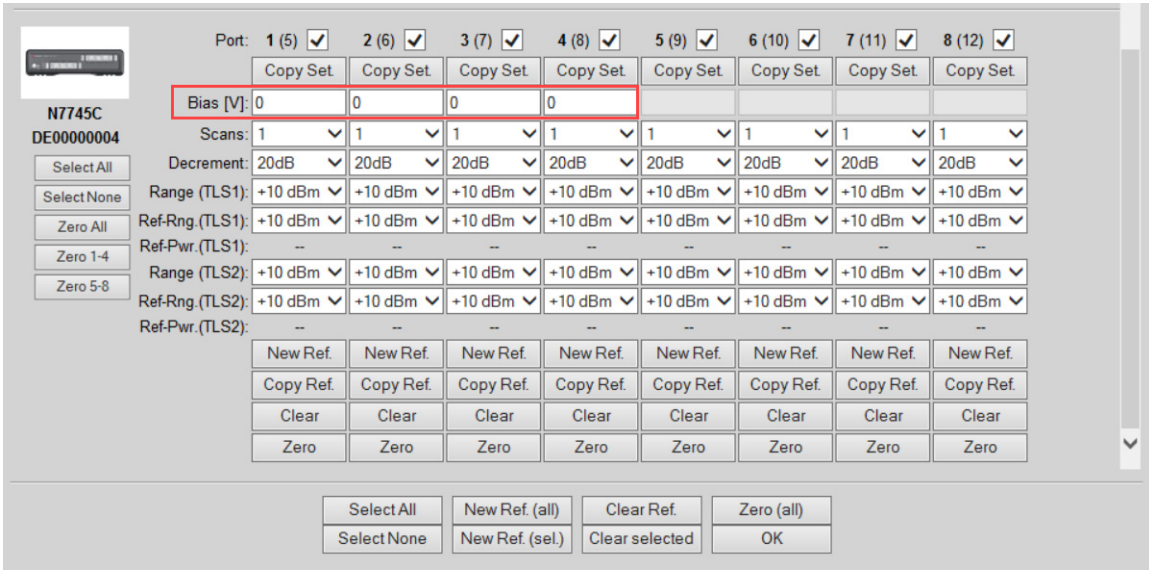


Related COM API

- property ReturnLossModulePresent
- method CalibrateRLM
- property RLMCalReflectorValue
- method SetRLMSensitivity
- method GetRLMSensitivity
- method SetRLMSensitivityMon
- method GetRLMSensitivityMon
- property RLMCalibrationFileList
- method DeleteRLMCalibrationFile

Bias Settings

If the setup comprises at least one SMU or power meter with electrical inputs for measuring photo currents, the Port/Reference Manager will show enabled fields for setting the bias values.

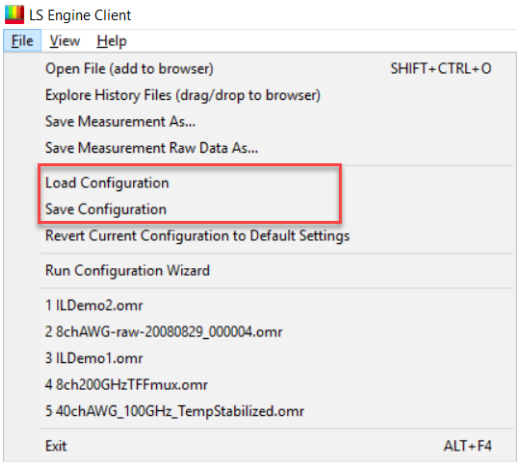


The Bias drop-down list determines the bias mode as described in the following table.

Auto (Default)	Actual bias voltages are kept at 0V until the measurement is started. During the measurement they are set to the defined values automatically. After the measurement has finished, actual bias voltages are reverted to 0V.
Off	Actual bias voltages are set to 0V, regardless of the defined bias voltage settings.
On	Actual bias voltages are set to the defined bias voltage settings right away and kept throughout, as well as after the measurement.

Configuration Handling

The current configuration, i.e. hardware setup as well as current parameters like wavelength range, can be saved and loaded from the File menu by selecting Save Configuration and Load Configuration.



A saved configuration file (*.agconfig) can also be started from the Windows Explorer and will launch the LS engine if it is not running. The configuration wizard can be launched from the File menu as well.

NOTE

If an agconfig file with identical hardware settings is loaded from within an activated engine, only the changed measurement parameters will be applied to the current engine. You can use this to quickly switch between different parameter sets.

If an agconfig file with different hardware settings is loaded instead, the current engine will be deactivated and the chosen agconfig file will be used to activate a new engine.

4 Automation

KEYSIGHT TECHNOLOGIES

COM Components

Automation is implemented using a mechanism called “COM”. COM has been introduced by Microsoft on the Windows platform to allow a unified way to communicate between different software components. Today, almost every programming language such as C#, C++, LabVIEW, Keysight Vee as well as MATLAB offers ways to use these so-called COM components. Once familiar with handling COM components, programming can be done using function browsers or auto-completion that show you the available properties and functions/methods including their parameter names. It is thus often possible to use the available functions without consulting the documentation.

Since the syntax of calling COM components is a little different in every programming language, we focus in our examples on MATLAB code. MATLAB has a very generic syntax and it should be easy to adapt the code to any other language. Further examples are included in the software distribution. These examples get installed with the Photonic Application Suite and can usually be found at the following location:

C:\Program Files\Keysight\Photonic Application Suite\Examples

Here is a simple example which starts a measurement (similar to clicking the button “Run Single”):

```
% Connect to Engine Manager
EngineMgr=actxserver('KtPasServer.EngineMgr');

% List all Engines currently running
EngineIDs=EngineMgr.EngineIDs;

% Always connect to first engine
Engine=EngineMgr.OpenEngine(EngineIDs(1));

% Start measurement
Engine.StartMeasurement;

% Release the engine and the engine manager
Engine.release;
EngineMgr.release;
```

This example assumes that you have already started the Client Software and thus a server is already running. Therefore, this example connects to an existing engine.

The next section explains how to create a new instance of the engine, in case the server has not yet been started.

Creating a New Engine

When writing your own software, it is very useful to start the Client Software first. Starting the Client Software will start the Server Software and creates an instance of the “Engine” COM component which handles the hardware communication. If your self-written software connects to this engine, you can see changes immediately in the client user interface. This is extremely useful in the debugging phase.

Finalizing your software, you probably don’t need the user interface anymore. In that case, you can start the COM server yourself and create an instance of the engine:

```
% Connect to Engine Manager
EngineMgr=actxserver('KtPasServer.EngineMgr');

% Create a new engine
Engine=EngineMgr.NewEngine;

% Load configuration file
Engine.LoadConfiguration('c:\\test.agconfig');

% Activate engine
Engine.Activate;

% Start measurement
Engine.StartMeasurement;

% Deactivate engine
Engine.DeActivate;

% Release the engine and the engine manager
Engine.release;
EngineMgr.release;
```

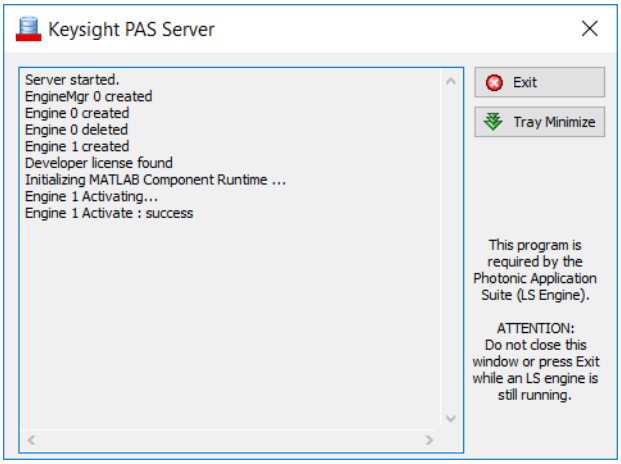
Note that you have to activate/deactivate the engine explicitly. If connecting to an existing engine, this is done by the client. When activating the engine, communication with the hardware is started.

We recommend to configure your engine initially using the Client Software and store the configuration using the “Save Configuration” menu. This example uses the “LoadConfiguration” command to load an .agconfig-file which has been stored before by the client.

NOTE

An icon appears in the Windows taskbar (on the bottom of your screen) when the server has started (i.e. the engine manager has been invoked):

Left-click on the icon to bring the server window to the front. The server will display a protocol line when an engine has been created or deleted:



Connecting to an Existing Engine

Use the following code to connect to an existing engine:

```
% Connect to Engine Manager
EngineMgr=actxserver('KtPasServer.EngineMgr');

% List all Engines currently running
EngineIDs=EngineMgr.EngineIDs;

% Always connect to first engine
Engine=EngineMgr.OpenEngine(EngineIDs(1));

% Start measurement
Engine.StartMeasurement;

% Release the engine and the engine manager
Engine.release;
EngineMgr.release;
```

NOTE In most of the following MATLAB examples, we assume that the engine has been created and activated before.

Performing a Measurement

A full MATLAB demo named “LS_Remote_Demo.m“ is included in the *DemoMATLAB* subfolder of the *Examples* folder available at the following location:

C:\Program Files\Keysight\Photonic Application Suite

Please take a look at this demo file even if you use a different programming language. The demo file is easy to understand even if you are not familiar with MATLAB. It shows you how to start a measurement, wait for the measurement to be finished and read the result.

The following code starts a measurement and waits for the measurement to be finished. The result is stored as an OMR file.

```
% Start
Engine.StartMeasurement;

% Wait for measurement to be finished*
while Engine.Busy;
pause(1);
LastProtocolMsg=UpdateProtocolText(Engine,LastProtocolMsg);
LastProtocolMsg=HandleUserInputs(Engine,LastProtocolMsg);
end;

% Get result object
MeasurementResult = Engine.MeasurementResult;

% Save as OMR file
MeasurementResult.Write('c:\test.omr');

% Release measurement object
MeasurementResult.release;

% Release engine
Engine.release;


% Release Engine Manager
EngineMgr.release;
```

* Here, the function call LastProtocolMsg=UpdateProtocolText(Engine,LastProtocolMsg) uses the Engine.ProtocolMin,Engine.ProtocolMax, and GetProtocolTextAt methods to display the most recent set of messages from the LS Engine server console.

The function call LastProtocolMsg=HandleUserInputs(Engine,LastProtocolMsg) uses the Engine.UserInputWaiting, Engine.UserInputPrompt, Engine.UserInputChoice, Engine.UserInputResponse(Response) methods to systematically receive the user input.

Accessing the Measurement Result

The measurement result is returned as reference to an .omr file object. This file COM object represents an OMR file and wraps all methods to read and write these files (Interface: IOMRFile).

NOTE Refer to the *N7700 Photonic Application Suite User's Guide* for a reference of the common interfaces, e.g. IOMRFile, IOMRGraph and IOMRProperty.

Once the measurement has finished, you can request a reference to this object and manipulate it. Typically you want to store the file somewhere on your hard disk or you want to access the graphs.

Saving the file can be done using the Write method:

```
% Save as OMR file
MeasurementResult.Write('c:\test.omr');
```

Accessing the graph data of the IL measurement is done like this:

```
% Access IL-graph
Graph = MeasurementResult.Graph('TLS0_RXTXAvgIL');
noChannels      = Graph.noChannels;
dataPerCurve    = Graph.dataPerCurve;
YData = reshape(Graph.YData,dataPerCurve,noChannels);
xStart = Graph.xStart;
xStep = Graph.xStep;
```

The graphs in the file have names. The name "TLS0_RXTXAvgIL" corresponds to the IL measurement performed with the first (or only) TLS in the setup. Please refer to the "Reference" section in the *Photonic Application Suite User Guide* for a list of available names.

The data are returned as a one-dimensional array, even if it contains more than one channel. The data for each channel are concatenated. The array starts with the first channel and continues with the next channels. The MATLAB command "reshape" converts this to a 2-dimensional array, a matrix. You can use the properties "noChannels" to get the total number of contained channels and the property "dataPerCurve" to find out how many values belong to a single channel. Graphs, such as TE/TM, have multiple curves per channel (as given by "noCurves" property).

NOTE

Using automation commands it is possible to start a new measurement engine as well as connect to an existing one, for example, one started from the Keysight Client GUI.

Certain automation operations, such as the Write method of the OMRFile interface or the FileSave method of the IKtPasServerEngine interface, lock the measurement object while accessing it. This might cause temporary failures in automation operations that are triggered from other instances connected to the same N7700 engine server.

This might especially be the case when using engine automation in addition to running the Keysight Client, which locks the OMRFile object in memory while updating measurement data in the GUI right after a measurement and when performing any automatic history save (see Automatic History Saves on page 131 to learn how to disable automatic saves).

To deal with this behavior, either implement retries or delays on affected operations or avoid connecting multiple instances to the same measurement engine server.

NOTE

Only the set of properties and methods described in the following sections should be used to build custom applications using the Lambda Scan engine.

Automation Using LabVIEW

This section contains information on how to set up LabVIEW VIs for automating N7700 Photonic Application Suite LS engine. It describes the required steps to use the LS engine's COM API from within LabVIEW.

A Note on Supported LabVIEW Versions

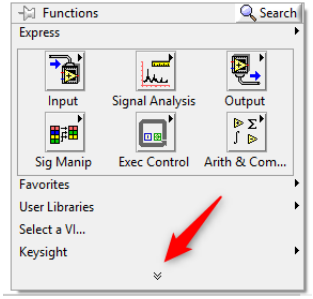
As VIs saved in a particular LabVIEW version cannot be opened in an older version of the LabVIEW software, but can be opened in newer versions, the LabVIEW examples available after the installation of the N7700 Photonic Application Suite engines have been saved in rather older LabVIEW versions. Similarly, this document describes the steps for LabVIEW 2009. These should work in newer LabVIEW versions right away, or display some messages about the required changes, if there are any.

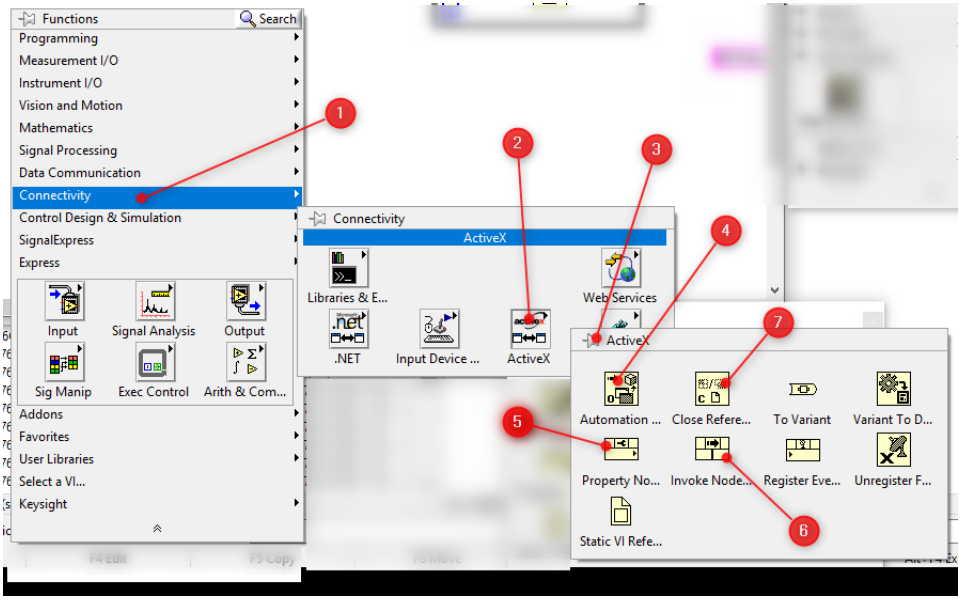
LS Engine Automation from NI LabVIEW

The Functions Palette

Perform the following steps:

1. Either start LabVIEW and create a blank VI or load a VI that you want to extend for accessing the LS engine.
2. Switch to the Block Diagram view and right-click on some blank region.
3. Expand the palette:

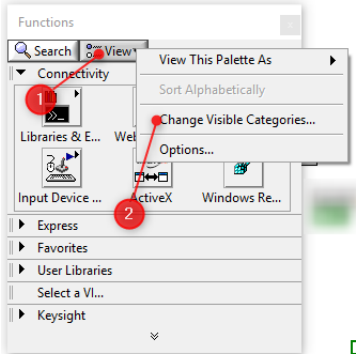


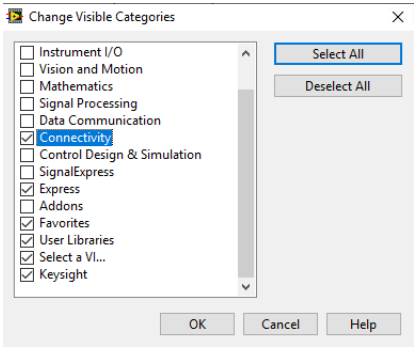


4 Click Connectivity (1), then ActiveX (2). You may want to pin the ActiveX palette for later access (3).

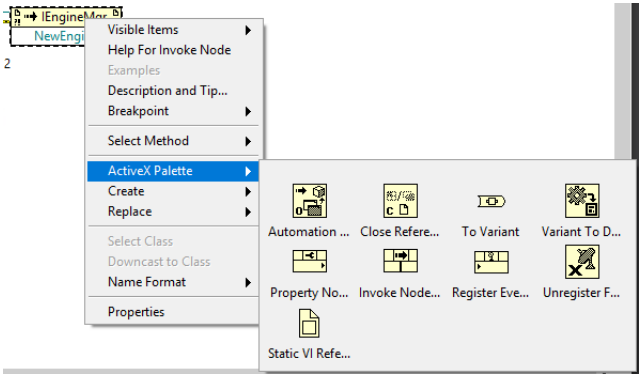
The most important items for COM API automation are Automation Open (4), Property Node (5), Invoke Node (6) and Close Reference (7).

You can also modify the items listed, by default, from the menu (View > Functions Palette, then clicking View and Change Visible Categories...)



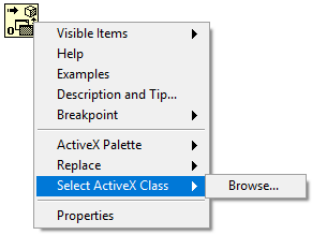


In addition, once any ActiveX (COM) function is placed, it can be right-clicked to access the ActiveX palette items from there.

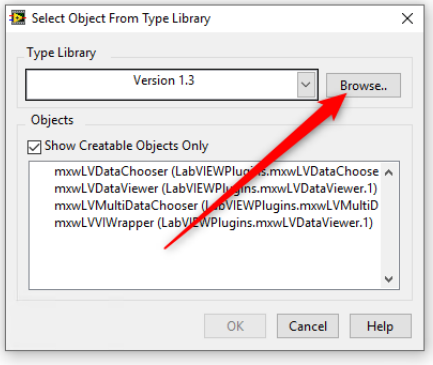


Accessing EngineMgr and Engine objects

From the ActiveX palette, place an Automation Open function in your block diagram, then right-click and select Select ActiveX Class > Browse...



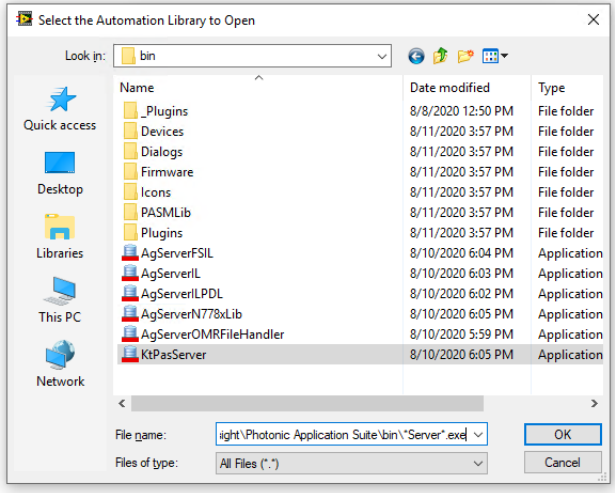
In a 64-bit LabVIEW, the respective PAS Engine COM classes should be listed in the Type Library drop-down. In case they are not (which is the case in 32-bit LabVIEW versions), click the Browse... button in the Select Object From Type Library dialog.



Navigate to C:\Program Files\Keysight\Photonic Application Suite\bin and select *.* as file type.

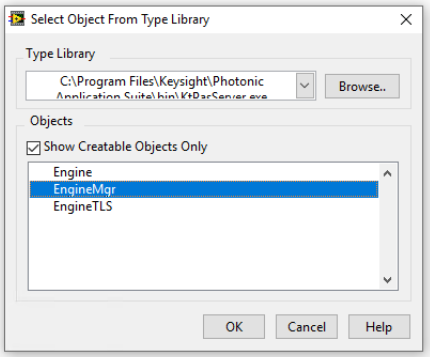
You can paste the following to the dialog's file name control and click OK / press Enter to narrow down the display as required:

C:\Program Files\Keysight\Photonic Application Suite\bin*Server*.exe



Choose the KtPasServer.exe server executable.

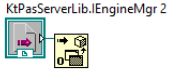
Select EngineMgr Object:



NOTE

The EngineTLS object type is only intended for the sake of compatibility with the IL/PDL Engine. Unless you are trying to enable an IL/PDL automation VI with the LS Engine, please refrain from using any EngineTLS objects.

This will automatically create a terminal for the EngineMgr object:

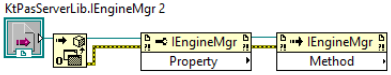


The Automation Open function provides two outputs:

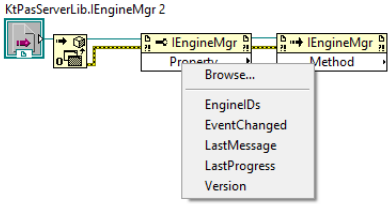
- A reference to the COM object
- An error output

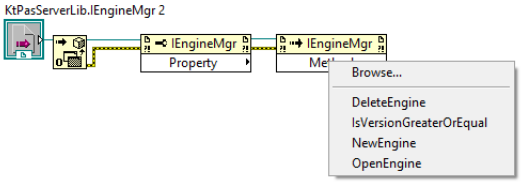
Both can be daisy-chained between COM automation functions.

From the ActiveX palette, select the Property Node and the Invoke Node, place them to the right of the Automation Open function, then connect engine reference and error out to the corresponding inputs of the next function:

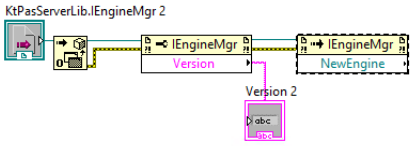


You can click the Property and Invoke nodes to see a list of the properties and methods provided by the COM object:

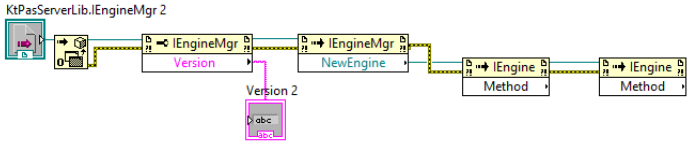




With the Property Node we could get a list of active engines, but for this example, we'll just display the EngineMgr version, then create a new engine:

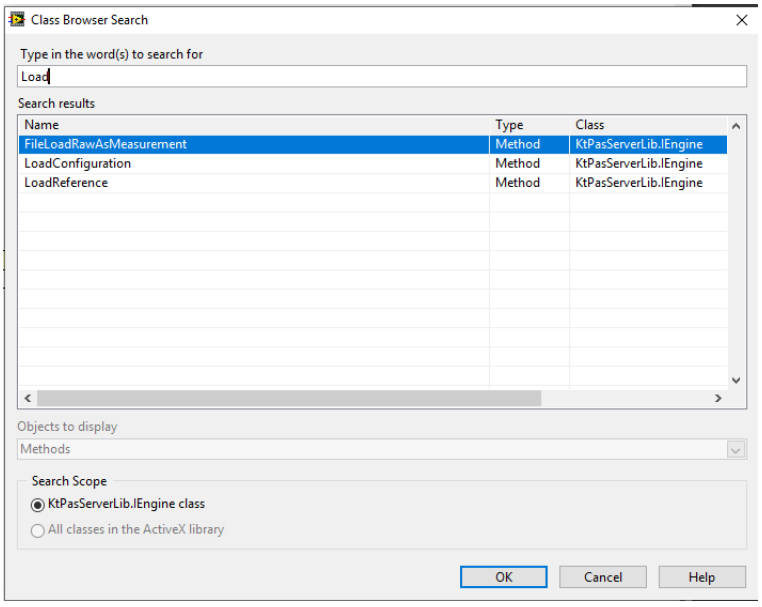


Note that the output of the NewEngine is a COM object (an Engine object) itself, so we will wire that to the reference input of the next Invoke Node:

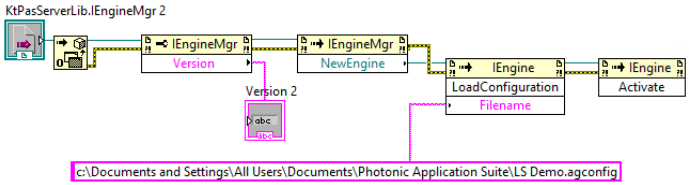


The Engine object provides numerous properties and methods, so the right-click menu will not fit on the screen. To navigate the list you can:

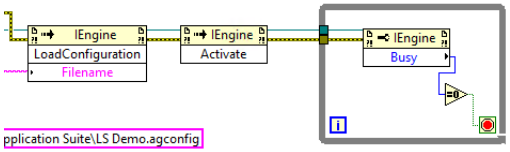
- Type a single letter, which will take you to the first item starting with that letter
- Hover the mouse cursor over the down-pointing triangle icon at the bottom of the list to scroll (ideally after typing the first letter)
- Click the Browse... item at the top of the list, then click the magnifying glass icon to open the Class Browser Search. In this dialog, you can type a portion of the desired property / method to narrow down the list of items.



Let us load a configuration file and Activate the engine:

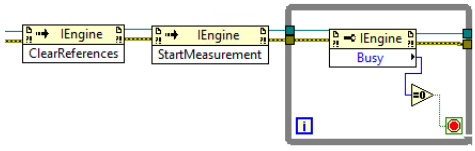


For the sake of simplicity, let us wait until the engine is no longer busy. In general, on any operation, such as engine activation or measurement operations, we must expect user interactions (in case of errors, or inconsistent settings). For details on how to deal with those, refer to the LS engine LabVIEW automation example.



Now we can automate any operation in the LS Engine that is also available in the GUI provided with the engine.

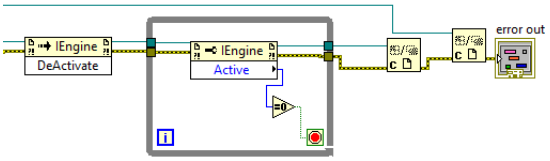
In this example, we will clear any existing reference, then run a measurement with the settings as defined in the loaded agconfig file:



As mentioned above, the measurement operation should not only check the Busy flag, but also the UserInputWaiting flag, then either parse (or prompt to user) UserInputPrompt and UserInputChoice. The selected response must then be set using UserInputResponse() and UserInputWaiting has to be set to 0 afterwards.

This is implemented in LSGetStatusMessageAndHandleUserInputs.vi and PASDialogBox.vi in C:\Program Files\Keysight\Photonic Application Suite\Examples\DemoLabView.

After performing the desired automation operations, the engine must be deactivated, and both the Engine and the EngineMgr object references should be closed. At the end of the daisy chain there should be an Indicator, such that any COM related error state can be read from the front panel:



Closing the references will happen automatically, if the VI execution is stopped, but in case of a more complex VI, it might be beneficial to close any references, as COM objects will be kept alive as long at least one reference to them still exists.

Refer to the LS_Demo_NewEngine.vi example and its sub-VIs for examples of how to use local variables to COM object references and how to use such references as “arguments” to sub-VIs.

LS Engine Example VI

An LS Engine automation example including some helper VIs can be found at the following location:

C:\Program Files\Keysight\Photonic Application Suite\Examples\DemoLabView\LS_Demo_NewEngine.vi

This example requires at least LabVIEW 2009 but has also been tested successfully with LabVIEW 2020.

To use the demo VI, perform the following steps:

1 Choose an LS Engine configuration (agconfig) file.

2 Start the VI. This will open KtPasServer (unless, it is already running) and create a new instance of the engine.

3 Watch the status messages during engine activation (and any subsequent operations).

4 Change the tunable laser parameters or data acquisition instrument ports to be used.

5 Run a measurement.

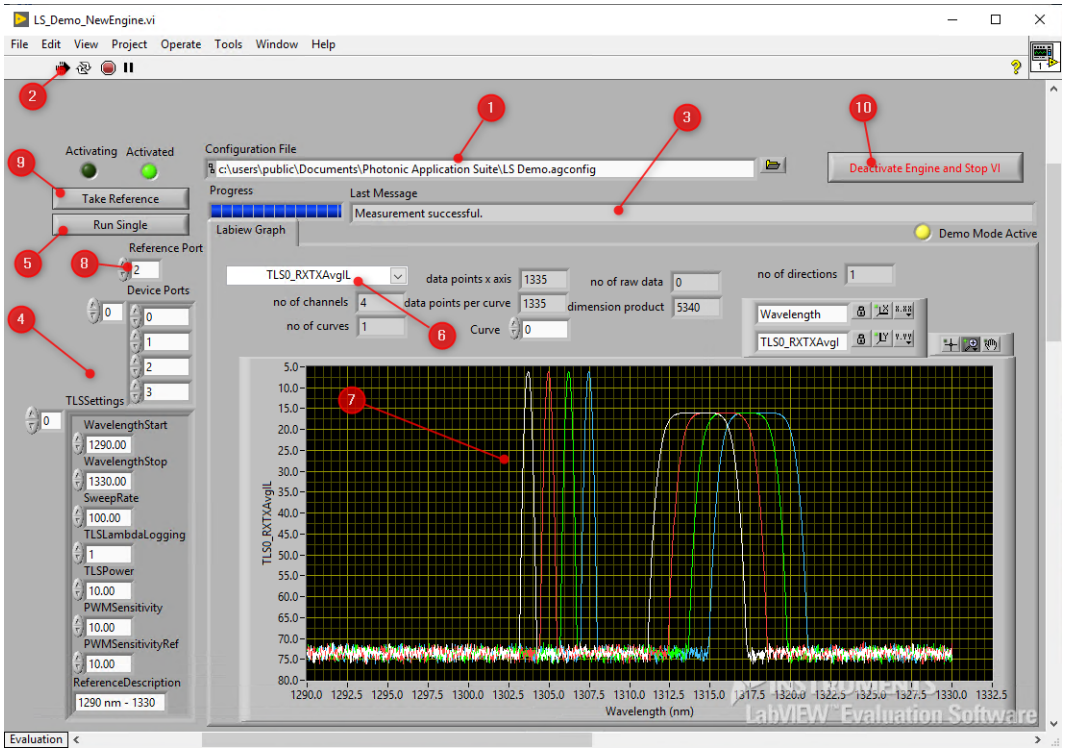
6 Choose a trace type to show.

7 Analyze the trace data.

8 Select a power meter instrument port to take a reference measurement with.

9 Take a reference and repeat measurement (steps 4-5).

10 Stop the automation session. This will release the engine and allow KtPasServer to close (unless started manually before step 2).



Reference: Interface “IKtPasServerEngineMgr”

The IKtPasServerEngineMgr interface is invoked using the following PROGID to identify the COM server:

```
KtPasServer.EngineMgr
```

MATLAB:

```
EngineMgr=actxserver('KtPasServer.EngineMgr');
```

property Version

Type-Library:

```
get: HRESULT Version([out, retval] BSTR* pVal)
```

MATLAB:

```
Version=EngineMgr.Version;
```

Shows information about revision and build date of the engine.

method IsVersionGreaterOrEqual

Type-Library:

```
HRESULT IsVersionGreaterOrEqual([in] BSTR
VersionNumber,[out,retval] BOOL* pVal);
```

MATLAB:

```
Engine=EngineMgr.IsVersionGreaterOrEqual('1.1.0.1');
```

Performs a check, whether the engine revision is greater than or equal to the revision number provided as a parameter.

method NewEngine

Type-Library:

```
HRESULT NewEngine([out,retval] IKtPasServerEngine**
aEngine);
```

MATLAB:

```
Engine=EngineMgr.NewEngine;
```

Tells the engine manager to create a new engine and returns the corresponding handle.

property EngineIDs

Type-Library:

```
get: HRESULT EngineIDs([out, retval] SAFEARRAY(LONG)* pVal)
```

MATLAB:

```
EngineIDs=EngineMgr.EngineIDs;
```

Returns the IDs of all created engines.

method OpenEngine

Type-Library:

```
HRESULT OpenEngine([in] LONG EngineID,[out,retval]
IKtPasServerEngine** aEngine);
```

MATLAB:

```
Engine=EngineMgr.OpenEngine(EngineID);
```

Returns the handle to an existing engine. The engine ID is a unique identifier which can be obtained by requesting the property “EngineIDs”.

method DeleteEngine

Type-Library:

```
HRESULT DeleteEngine([in] IKtPasServerEngine* aEngine);
```

MATLAB:

```
EngineMgr.DeleteEngine(Engine);
```

To ensure that all memory cleanup operations are executed properly, it is recommended to call the EngineMgr's DeleteEngine method with the current engine object as an argument after running the engine's DeActivate method and before releasing the engine object. After that, either create a new engine or release the EngineMgr object.

Reference: Interface “IKtPasServerEngine”

The IKtPasServerEngine interface is invoked either using the method “NewEngine” or “OpenEngine” of the IKtPasServerEngineMgr interface:

MATLAB:

```
Engine=EngineMgr.NewEngine;
```

NOTE

Status and user interaction messages refer to the instruments and ports using their labels (1-based), in conformance with the GUI representation, while API operations usually require the instrument index (0-based) as arguments.

In the commands documented below, all the tls and port indices are zero-based, i.e., the first laser and the first port are indexed using "0".

NOTE

The **Supported Configurations** section in each of the API descriptions lists one of more of the following three applicable flags:

PLS: Indicates applicability to LS engine configurations for IL and PDL measurements (comprising an N7786B/C plus one or more power meters or SMUs). Corresponds to setups using N7700100C PLS license.

FLS: Indicates applicability to LS engine configurations for IL measurements (comprising neither N7786B/C nor N7788C but one or more power meters or SMUs). Corresponds to setups using N7700102C FLS license or to setups using N7700100C PLS license, with polarization synthesizer not being part of the currently active configuration.

PMD: Indicates applicability to LS engine configurations for PMD measurements (comprising an N7788C and neither power meters nor SMUs). Corresponds to setups using N7700103C PMD license.

property EngineConstructionDone

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT EngineConstructionDone([out, retval] BOOL*
pVal);
```

MATLAB:

```
b=Engine.EngineConstructionDone;
```

Performs a check whether the engine has been created or not. This can be helpful to avoid proceeding with engine operations in case license checks take unexpectedly longer time on certain systems.

property Version

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT Version([out, retval] BSTR* pVal)
```

MATLAB:

```
Version=Engine.Version;
```

Shows information about revision and build date of the engine.

method IsVersionGreaterOrEqual

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT IsVersionGreaterOrEqual([in] BSTR
VersionNumber,[out,retval] BOOL* pVal);
```

MATLAB:

```
Engine=Engine.IsVersionGreaterOrEqual('1.1.0.1');
```

Performs a check, whether the engine revision is greater than or equal to the revision number provided as a parameter.

method Activate

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT Activate(void);
```

MATLAB:

```
Engine.Activate;
```

Activates the engine. Usually a configuration file, created using the client software, should be loaded prior to activating the engine. This is done using the “LoadConfiguration” method. Activating an engine will cause the server to communicate with the instruments.

After running this method, you must ensure that the Active property returns 1, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method DeActivate

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT DeActivate(void);
```

MATLAB:

```
Engine.DeActivate;
```

Deactivates the engine. You should deactivate the engine before loading a different configuration.

After running this method, you must ensure that the Active property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

NOTE **After deactivating an engine and before releasing the engine object and its corresponding EngineMgr object, the EngineMgr's DeleteEngine method should be called. Refer to method DeleteEngine on page 168 for further details.**

property Active

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT Active([out, retval] BOOL* pVal);
```

MATLAB:

```
x = Engine.Active;
```

Returns 0/1 if the engine is inactive/active.

property Busy

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT Busy([out, retval] BOOL* pVal);
```

MATLAB:

```
x = Engine.Busy;
```

Returns 0/1 if the engine is not busy/busy. The engine might be busy when transferring large amounts of data to or from the instruments, when being activated, or when it is waiting for a measurement operation to complete. This flag is intended for operations that run asynchronously, such as those mentioned above. Please note that the flag might not reflect the engine's actual busy state until an engine activation has been performed. Especially after creating an engine using an EngineMgr object and after loading a configuration file, but before executing the Activate method, the Busy flag will return 1 (busy).

After running some of the methods described in this section, an explicit value check for the Busy property is required to ensure that the engine instance is free before running any subsequent operations. This has been mentioned in the description of these methods.

property EmulationMode

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT EmulationMode([out, retval] BOOL* pVal);
```

```
put: HRESULT EmulationMode([in] BOOL Val);
```

MATLAB:

```
b = Engine.EmulationMode;
```

```
Engine.EmulationMode = b;
```

Sets or gets the emulation (demo) mode setting. If operated in demo mode, no actual communication with instruments will be performed and simulated measurement results will be shown instead.

method RefreshClients

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT RefreshClients(void);
```

MATLAB:

```
Engine.RefreshClients();
```

Refreshes the settings in the LS Engine GUI after these settings have been changed programmatically or using remote calls.

method LoadConfiguration

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT LoadConfiguration([in] BSTR Filename);
```

MATLAB:

```
Engine.LoadConfiguration('c:\\test.agconfig');
```

Loads a configuration file (extension: .agconfig). The engine should be inactive while loading a new config file, i.e., either before calling Activate for the first time after the engine has been created or after calling Deactivate.

method CheckForValidConfigFile

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT CheckForValidConfigFile([in] BSTR Filename, [out,
retval] BOOL * pVal);
```

MATLAB:

```
b=Engine.CheckForValidConfigFile('c:\\test.agconfig');
```

Returns true if the configuration specified by the input .agconfig file is valid for the current engine version.

method CompareHardwareConfiguration

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT CompareHardwareConfiguration([in] BSTR Filename,
[out, retval] BOOL * pVal);
```

MATLAB:

```
b=Engine.CompareHardwareConfiguration('c:\\test.agconfig');
```

Returns true if the currently active engine configuration is same as the one specified by the input .agconfig file.

property Configuration

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT Configuration([out, retval] BSTR* ConfigXML);
put: HRESULT Configuration([in] BSTR ConfigXML);
```

MATLAB:

```
xml = Engine.Configuration;
Engine.Configuration = xml;
```

Transfers the contents of an .agconfig-file to or from the engine. An .agconfig-file contains XML formatted text.

method SetTLSInclude

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetTLSInclude([in] LONG nTLS, [in] BOOL newVal);
```

MATLAB:

```
Engine.SetTLSInclude(2,1);
```

Includes or excludes the specified TLS from subsequent operations (Taking references, running measurements, performing return loss calibrations, etc).

method GetTLSInclude

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetTLSInclude([in] LONG nTLS, [ out, retval ] BOOL *
pVal);
```

MATLAB:

```
b = Engine.GetTLSInclude(1);
```

Gets whether the specified TLS is included or excluded from subsequent operations (Taking references, running measurements, performing return loss calibrations, etc).

method GetTLSModelCode

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetTLSModelCode([in] LONG nTLS, [ out, retval ] BSTR
* pVal);
```

MATLAB:

```
s = Engine.GetTLSModelCode(1);
```

Gets the model code of the specified TLS.

method GetTLSSerialNumber

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetTLSSerialNumber([in] LONG nTLS, [ out, retval ]
BSTR * pVal);
```

MATLAB:

```
s = Engine.GetTLSSerialNumber(1);
```

Gets the serial number of the specified TLS.

method SetWavelengthStart

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetWavelengthStart([in] LONG nTLS, [in] DOUBLE
newVal);
```

MATLAB:

```
Engine.SetWavelengthStart(1,1530);
```

Sets the start wavelength of the specified TLS in nm.

method GetWavelengthStart

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetWavelengthStart([in] LONG nTLS, [ out, retval ]
DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetWavelengthStart(1);
```

Gets the start wavelength of the specified TLS in nm.

method SetWavelengthStop

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetWavelengthStop([in] LONG nTLS, [in] DOUBLE
newVal);
```

MATLAB:

```
Engine.SetWavelengthStop(1,1570);
```

Sets the stop wavelength of the specified TLS in nm.

method GetWavelengthStop

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetWavelengthStop([in] LONG nTLS, [ out, retval ]
DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetWavelengthStop(1);
```

Gets the stop wavelength of the specified TLS in nm.

method SetSweepRate

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetSweepRate([in] LONG nTLS, [in] DOUBLE newVal);
```

MATLAB:

```
Engine.SetSweepRate(1,20);
```

Sets the sweep rate of the specified TLS in nm/s.

method GetSweepRate

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetSweepRate([in] LONG nTLS, [ out, retval ] DOUBLE
* pVal);
```

MATLAB:

```
d = Engine.GetSweepRate(1);
```

Gets the sweep rate for the specified TLS in nm/s.

method SetWavelengthStep

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetWavelengthStep([in] LONG nTLS, [in] DOUBLE
newVal);
```

MATLAB:

```
Engine.SetWavelengthStep(1,30);
```

Sets the wavelength step size for the specified TLS in pm.

method GetWavelengthStep

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetWavelengthStep([in] LONG nTLS, [ out, retval ]
DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetWavelengthStep(1);
```

Gets the wavelength step size for the specified TLS in pm.

method SetTriggerOverSamplingRatio

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetTriggerOverSamplingRatio([in] LONG nTLS, [in]
LONG newVal);
```

MATLAB:

```
Engine.SetTriggerOverSamplingRatio(1,1);
```

Sets the ratio between the number of triggers generated and number of actual wavelength samples for the specified TLS.

method GetTriggerOverSamplingRatio

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetTriggerOverSamplingRatio([in] LONG nTLS, [ out,
retval ] LONG * pVal);
```

MATLAB:

```
i = Engine.GetTriggerOverSamplingRatio(1);
```

Gets the ratio between the number of triggers generated and number of actual wavelength samples for the specified TLS.

method SetUseTwoWay

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetUseTwoWay([in] LONG nTLS, [in] BOOL newVal);
```

MATLAB:

```
Engine.SetUseTwoWay(1,1);
```

If true, configures the specified TLS to use bidirectional sweeps. (increasing/decreasing wavelength)

method GetUseTwoWay

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetUseTwoWay([in] LONG nTLS, [ out, retval ] BOOL *
pVal);
```

MATLAB:

```
b = Engine.GetUseTwoWay(1);
```

Specifies whether the specified TLS is configured to use bidirectional sweeps (increasing/decreasing wavelength).

method SetTLSPower

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetTLSPower([in] LONG nTLS, [in] DOUBLE newVal);
```

MATLAB:

```
Engine.SetTLSPower(1,0);
```

Sets the laser power for the specified TLS in mW.

method GetTLSPower

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetTLSPower([in] LONG nTLS, [ out, retval ] DOUBLE *
pVal);
```

MATLAB:

```
d = Engine.GetTLSPower(1);
```

Gets the laser power for the specified TLS in mW.

method SetOutputActive

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetOutputActive([in] LONG nTLS, [in] BOOL newVal);
```

MATLAB:

```
Engine.SetOutputActive(1,0);
```

Enable/Disable the optical output for the specified TLS. Setting this property will immediately turn on or off the optical output, while the other settings will only take effect upon the next sweep operation, e.g. running a single measurement.

method GetOutputActive

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetOutputActive([in] LONG nTLS, [ out, retval ] BOOL
* pVal);
```

MATLAB:

```
b = Engine.GetOutputActive(1);
```

Returns whether the optical output for the specified TLS is enabled or disabled.

method SetTlsOpticalSwitchPort

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetTlsOpticalSwitchPort([in] LONG nTLS, [in] LONG
newVal);
```

MATLAB:

```
Engine.SetTlsOpticalSwitchPort(1,1);
```

Sets the optical switch port to be used for a measurement for the specified TLS. Use switch port as indicated on instrument front panel, for example, 1 for the first switch port.

method GetTlsOpticalSwitchPort

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetTlsOpticalSwitchPort([in] LONG nTLS, [ out,
retval ] LONG * pVal);
```

MATLAB:

```
i = Engine.GetTlsOpticalSwitchPort(1);
```

Gets the optical switch port to be used for a measurement for the specified TLS.

method GetSweepRates

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetSweepRates([in] LONG nTLS, [ out, retval ]
SAFEARRAY(DOUBLE) * pVal);
```

MATLAB:

```
d = Engine.GetSweepRates(1);
```

Returns a double array containing the sweep rates in nm/s that the specified TLS can sweep at. This method is not supported with PMD (N7788C) configurations.

method GetBands

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetBands([in] LONG nTLS, [ out, retval ] BSTR *
pVal);
```

MATLAB:

```
s = Engine.GetBands(1);
```

Gets the wavelength bands covered by the specified TLS. The returned string is a concatenation of the band characters, e.g. “CL”.

method GetReferenceAvailable

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetReferenceAvailable([in] LONG nTLS, [in] LONG
nPort, [ out, retval ] BOOL * pVal);
```

MATLAB:

```
b = Engine.GetReferenceAvailable(1,1);
```

Returns TRUE if reference data is available for the specified port and the specified TLS or FALSE otherwise.

method GetReferenceFileName

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetReferenceFileName([in] LONG nTLS, [out, retval ]
BSTR * pVal);
```

MATLAB:

```
s = Engine.GetReferenceFileName(1);
```

Returns the file name that reference data has most recently been saved to / loaded from corresponding to the specified TLS.

method GetReferenceDescription

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetReferenceDescription([in] LONG nTLS, [ out,
retval ] BSTR * pVal);
```

MATLAB:

```
s = Engine.GetReferenceDescription(1);
```

Gets a string containing the most important parameters of the current reference for the specified TLS.

method GetReferenceAveragePower

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetReferenceAveragePower([in] LONG nTLS, [in] LONG
nPort, [out, retval] DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetReferenceAveragePower(1,1);
```

Use this method to get the average power of a reference trace for a specific port and specified TLS. Provide the port number to be checked, starting at 0 for the first port of the first power meter / SMU and counting continuously across multiple power meters / SMU.

The average power is computed from the first, the last and the center sample of each trace. Will return NaN if no reference trace exists for the queried port.

property TlsSwitchPresent

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT TlsSwitchPresent([out, retval] BOOL* pVal);
```

MATLAB:

```
b = Engine.TlsSwitchPresent;
```

Returns whether a TLS switch is present in the setup or not.

method StartCustomReference/ StartCustomReferenceSinglePort

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT StartCustomReference([in] SAFEARRAY(LONG) pVal);
HRESULT StartCustomReferenceSinglePort([in] LONG Port);
```

MATLAB:

```
Engine.StartCustomReference(int32([3;6]));
Engine.StartCustomReferenceSinglePort(3);
```

(For non-PMD configurations only) Starts a reference measurement. As a parameter, an integer array of port numbers or a single port number, respectively (starting at 0 for port 1), must be provided. A reference sweep will be performed and reference data for all provided ports will be stored.

NOTE

When using MATLAB (and potentially other automation environments), it might not be possible to match the argument type (integer array) when trying to perform this operation on a single port. In such a situation, use Engine.StartCustomReferenceSinglePort(nPort) instead.

After running these methods, you must ensure that the Busy property returns 0, before you run any subsequent methods or get/set any other properties. (except for those related to user input handling)

method StartReference

Supported Configurations: PMD

Type-Library:

```
HRESULT StartReference(void);
```

MATLAB:

```
Engine.StartReferences();
```

(For PMD configurations only) Starts a reference measurement.

method ClearReferences

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT ClearReferences(void);
```

MATLAB:

```
Engine.ClearReferences();
```

Clears reference data for all the ports and all the TLS.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method ClearReferenceChannels / ClearReferenceChannelsSinglePort

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT ClearReferenceChannels([in] SAFEARRAY(LONG) Ports);
HRESULT ClearReferenceChannelsSinglePort([in] LONG Port);
```

MATLAB:

```
Engine.ClearReferenceChannels(int32([3;6]));
Engine.ClearReferenceChannelsSinglePort(3);
```

Clears reference data from the currently loaded reference file. The saved file will not be changed by this operation. To update the saved reference file, the modified reference needs to be saved with the same filename. As a parameter, an integer array of port numbers or a single port number, respectively (starting at 0 for port 1), must be provided.

NOTE When using MATLAB (and potentially other automation environments), it might not be possible to match the argument type (integer array) when trying to perform this operation on a single port. In such a situation, use Engine.ClearReferenceChannelsSinglePort(nPort) instead.

After running these methods, you must ensure that the Busy property returns 0, before you run any subsequent methods or get/set any other properties. (except for those related to user input handling)

method CopyReferenceChannel / CopyReferenceChannelSinglePort

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT CopyReferenceChannel([in] LONG SourcePort, [in]
SAFEARRAY(LONG) TargetPorts);
HRESULT CopyReferenceChannelSinglePort([in] SourcePort, [in]
LONG TargetPort);
```

MATLAB:

```
Engine.CopyReferenceChannel(5, int32([3;6]));
Engine.CopyReferenceChannelSinglePort(5, 3);
```

Copies reference data from the source port to all target ports in the currently loaded reference file. The saved file will not be changed by this operation. To update the saved reference file, the modified reference

needs to be saved with the same filename. As parameters, the source port of the copy operation and an integer array of the target port numbers or a single port number, respectively (parameters starting at 0 for port 1), must be provided.

NOTE **When using MATLAB (and potentially other automation environments), it might not be possible to match the argument type (integer array) when trying to perform this operation on a single port. In such a situation, use Engine.CopyReferenceChannelSinglePort(nPort) instead.**

After running these methods, you must ensure that the Busy property returns 0, before you run any subsequent methods or get/set any other properties. (except for those related to user input handling)

property ReferenceList

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT ReferenceList([out, retval] BSTR*
ReferenceList);
```

MATLAB:

```
s = Engine.ReferenceList;
```

Returns an xml string containing the filenames and important parameters for all reference files available on the computer running the engine server.

method LoadReference

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT LoadReference([in] LONG tls, [in] BSTR Filename);
```

MATLAB:

```
Engine.LoadReference(1,'Reference.lsref');
```

Loads a reference file in .lsref format for the current TLS.

Use `Engine.ReferenceList` to get a list of reference files present on the computer running the server.

method SaveReference

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SaveReference([in] LONG tls, [in] BSTR Filename,
[in] BOOL Overwrite);
```

MATLAB:

```
Engine.SaveReference(1, 'Reference.lsref', 1);
```

Saves the current reference of the specified TLS to a .lsref file. Will not overwrite an existing file unless Overwrite parameter is set to 1.

Note that the file will be stored on the computer running the engine server.

method DeleteReferenceFile

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT DeleteReferenceFile([in] BSTR Filename);
```

MATLAB:

```
Engine.DeleteReferenceFile('Reference.pbin');
```

Deletes a reference file on the engine server. Use `Engine.ReferenceList` to get a list of reference files present on the server.

method ValidateSettings

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT ValidateSettings(void);
```

MATLAB:

```
Engine.ValidateSettings;
```

Checks measurement parameters for validity and prompts for adjustments, if required, e.g. wavelength range exceeding the sweep range of the TLS used. Will also check whether the parameters match the currently selected reference.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method ValidateSettingsNoRefCheck

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT ValidateSettingsnoRefCheck(void);
```

MATLAB:

```
Engine.ValidateSettingsnoRefCheck;
```

Checks measurement parameters for validity and prompts for adjustments, if required, e.g. wavelength range exceeding the sweep range of the TLS used. Will not check whether the parameters match the current reference. Can be used before taking a new reference with changed parameters.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

property DarkIncludeMeasurement

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT DarkIncludeMeasurement([out, retval] BOOL*
pVal);
```

```
put: HRESULT DarkIncludeMeasurement([in] BOOL Val);
```

MATLAB:

```
b = Engine.DarkIncludeMeasurement;
```

```
Engine.DarkIncludeMeasurement = b;
```

Defines whether dark current measurements should be performed in addition to the swept measurement.

property ERIncludeMeasurement

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT ERIncludeMeasurement([out, retval] BOOL* pVal);
```

```
put: HRESULT ERIncludeMeasurement([in] BOOL Val);
```

MATLAB:

```
b = Engine.ERIncludeMeasurement;
```

```
Engine.ERIncludeMeasurement = b;
```

Defines whether stepped Polarization Extinction Ratio measurements should be performed in addition to the swept measurement.

method StartMeasurement

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT StartMeasurement(void);
```

MATLAB:

```
Engine.StartMeasurement;
```

Starts a measurement.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method StartMeasurementRepeat

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT StartMeasurementRepeat(void);
```

MATLAB:

```
Engine.StartMeasurementRepeat;
```

Starts a series of measurements. Will stop after StopMeasurement method has been called. Will not save any measurement data automatically, except for any automatic history saves that might have been configured in an engine client (LS Engine GUI), connected to the same engine server.

method StopMeasurement

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT StopMeasurement(void);
```

MATLAB:

```
Engine.StopMeasurement;
```

During a measurement you can abort the current measurement sweep and stop the repeat operation by using this method. The measurement will be stopped as soon as possible, but usually not immediately, because some hardware operations have to be completed first. Usually the next measurement is completed before stopping, so this method should be used primarily for stopping the Repeat mode operation.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method FileSave

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT FileSave([in] BSTR Filename);
```

MATLAB:

```
Engine.FileSave('test.omr');
```

Saves the measurement result to an OMR file at the location provided as parameter.

See the last Note in Accessing the Measurement Result on page 155.

property MeasurementResult

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT MeasurementResult([out, retval] IOMRFile** pVal);
```

MATLAB:

```
Result = Engine.MeasurementResult;
```

Returns a reference to the underlying IOMRFile object which contains the measurement result.

NOTE

Refer to the *N7700 Photonic Application Suite User's Guide* for a reference of the common interfaces, e.g. IOMRFile, IOMRGraph and IOMRProperty.

See the last Note in Accessing the Measurement Result on page 155.

property ResolutionValues

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT ResolutionValues([out, retval]
SAFEARRAY(double)* pVal);
```

MATLAB:

```
DoubleArray = Engine.ResolutionValues;
```

Returns a double array containing the suggested window widths for the moving-average smoothing in nm.

property Resolution

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT Resolution([out, retval] DOUBLE* pVal);
```

```
put: HRESULT Resolution([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.Resolution;
```

```
Engine.Resolution = d;
```

Sets or gets the window widths for the moving-average smoothing in nm.

method ReCalculate

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT ReCalculate(void);
```

MATLAB:

```
Engine.ReCalculate;
```

Triggers a recalculation. Call this method if you have changed evaluation parameters such as resolution or, e.g., GenerateTETMData.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method GetPWMModelCode

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMModelCode([in] LONG nPWM, [ out, retval ] BSTR
* pVal);
```

MATLAB:

```
s = Engine.GetPWMModelCode(1);
```

Gets the model code of the specified power meter /SMU.

method GetPWMSerialNumber

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMSerialNumber([in] LONG nPWM, [ out, retval ]
BSTR * pVal);
```

MATLAB:

```
s = Engine.GetPWMSerialNumber(1);
```

Gets the serial number of the specified power meter / SMU.

property PWMCount

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT PWMCount([out, retval] LONG* pVal);
```

MATLAB:

```
NumberOfPWMsAttached = Engine.PWMCount;
```

Returns the number of power meters / SMUs configured in the current setup. Will only return a positive, i.e., valid, value after engine activation has been started.

method GetPWMChannelCount

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMChannelCount([in] LONG nPWM,[out,retval] LONG*
pVal);
```

MATLAB:

```
PWMChannelCount = Engine.GetPWMChannelCount(nPWM);
```

Returns the number of ports for a given power meter/SMU in the setup. You can obtain the number of power meters/SMUs attached and configured by using Engine.GetPWMChannelCount.

property TotalPortCount

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT TotalPortCount([ out, retval ] LONG * pVal);
```

MATLAB:

```
NumberOfPorts = Engine.TotalPortCount;
```

Returns the total number of ports for all the available power meters/SMUs in the setup.

property TLSCount

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT TLSCount([out, retval] LONG* pVal);
```

MATLAB:

```
NumberOfTLSAttached = Engine.TLSCount;
```

Returns the number of TLS configured in the current setup. Will only return a non-zero, i.e., valid value after engine activation has been started.

property KeepRawData

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT KeepRawData([out, retval] BOOL* pVal);
put: HRESULT KeepRawData([in] BOOL Val);
```

MATLAB:

```
b = Engine.KeepRawData;
Engine.KeepRawData = b;
```

Defines whether stored measurement files will contain measurement raw data.

property PWMAutoRanging

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT PWMAutoRanging([out, retval] BOOL* pVal);
```

```
put: HRESULT PWMAutoRanging([in] BOOL Val);
```

MATLAB:

```
b = Engine.PWMAutoRanging;
```

```
Engine.PWMAutoRanging = b;
```

Defines whether the power ranges of any configured power meters / SMUs / polarization synthesizer / component analyzer / return loss modules are adjusted automatically to avoid overrange and underrange measurements.

method SetNumberOfScans

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetNumberOfScans([in] LONG nTLS, [in] LONG nPort,
[in] LONG newVal);
```

MATLAB:

```
Engine.SetNumberOfScans(1,1,2);
```

Sets the number of dynamic range scans to be performed corresponding to the specified TLS and port. Use the method SetPWMRangeDecrement on page 194 to define the step size by which the power meter range setting will be changed in each dynamic range scan.

method GetNumberOfScans

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetNumberOfScans([in] LONG nTLS, [in] LONG nPort, [
out, retval ] LONG * pVal);
```

MATLAB:

```
d = Engine.GetNumberOfScans(1,1);
```

Gets the number of dynamic range scans to be performed corresponding to the specified TLS and port.

method SetPWMRangeDecrement

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetPWMRangeDecrement([in] LONG nTLS, [in] LONG
nPort, [in] DOUBLE newVal);
```

MATLAB:

```
Engine.SetNumberOfScans(1,1,2);
```

Sets the step size by which the power meter range setting will be changed in each dynamic range scan. Provided value must be a multiple of 10dB. Use the method SetNumberOfScans on page 193 to define the number of such scans to be performed in each measurement.

method GetPWMRangeDecrement

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMRangeDecrement([in] LONG nTLS, [in] LONG
nPort, [ out, retval ] DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetPWMRangeDecrement(1,1);
```

Gets the step size by which the power meter range setting will be changed in each dynamic range scan.

method SetAveragingTime

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetAveragingTime([in] LONG nTLS, [in] DOUBLE
newVal);
```

MATLAB:

```
Engine.SetAveragingTime(1,5);
```

Sets the power meter averaging time in us corresponding to the specified TLS.

The engine accepts the following averaging time values (in us), but not every power meter or SMU supports all of these. The engine will show an error when trying to set an averaging time that is unsupported.

0/1/2/5/10/20/25/50/100/200/500/1000/2000/5000/10000/20000/50000

When set to 0, the engine will choose an appropriate averaging time, based on the selected sweep rate and step size (see Instrument/DUT Bandwidth and Measurement Trigger Timing on page 47 for details).

method GetAveragingTime

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetAveragingTime([in] LONG nTLS, [ out, retval ]
DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetAveragingTime(1);
```

Gets the power meter averaging time in us corresponding to the specified TLS. When set to 0, the engine will choose an appropriate averaging time, based on the selected sweep rate and step size (see Instrument/DUT Bandwidth and Measurement Trigger Timing on page 47 for details).

property MPPMAutoGain

Supported Configurations: PLS, FLS

Type- Library:

```
get: HRESULT MPPMAutoGain([out, retval] BOOL* pVal);

put: HRESULT MPPMAutoGain([in] BOOL Val);
```

MATLAB:

```
b= Engine.MPPMAutoGain;

Engine.MPPMAutoGain= b;
```

If set to true, the measurement engine will configure any MPPM to use its auto gain feature, which gives improved dynamic range performance. Refer to the Instrument Setup Parameters on page 58 for further details.

property TLSExtendedSweepRange

Supported Configurations: PLS, FLS, PMD

Type- Library:

```
get: HRESULT TLSExtendedSweepRange([out, retval] BOOL*
pVal);
```

```
put: HRESULT TLSExtendedSweepRange([in] BOOL Val);
```

MATLAB:

```
b= Engine.TLSExtendedSweepRange;
```

```
Engine.TLSExtendedSweepRange= b;
```

If set to true, power meter sampling will start sooner after the TLS sweep start.

Refer to the Instrument Setup Parameters on page 58 for further details.

property GenerateTETMData

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT GenerateTETMData([out, retval] BOOL* pVal);
```

```
put: HRESULT GenerateTETMData([in] BOOL Val);
```

MATLAB:

```
b = Engine.GenerateTETMData;
```

```
Engine.GenerateTETMData = b;
```

Defines whether TE/TM and Mueller data is computed during measurement evaluation.

property GenerateReturnLossData

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT GenerateReturnLossData([out, retval] BOOL*
pVal);
```

```
put: HRESULT GenerateReturnLossData([in] BOOL Val);
```

MATLAB:

```
b = Engine.GenerateReturnLossData;
```

```
Engine.GenerateReturnLossData = b;
```

Defines whether return loss data is computed during measurement evaluation (if a return loss module is connected and configured for the current setup).

property GenerateResponsivityData

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT GenerateResponsivityData([out, retval] BOOL*
pVal);
```

```
put: HRESULT GenerateResponsivityData([in] BOOL Val);
```

MATLAB:

```
b = Engine.GenerateResponsivityData;
```

```
Engine.GenerateResponsivityData = b;
```

Defines whether photodiode responsivity data is computed during measurement evaluation (if a power meter / SMU instrument with electrical inputs is connected and configured for the current setup).

property GenerateDiodeCurrentData

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT GenerateDiodeCurrentData([out, retval] BOOL*
pVal);
```

```
put: HRESULT GenerateDiodeCurrentData([in] BOOL Val);
```

MATLAB:

```
b = Engine.GenerateDiodeCurrentData;
```

```
Engine.GenerateDiodeCurrentData = b;
```

Defines whether photodiode current data is computed during measurement evaluation (if a power meter / SMU instrument with electrical inputs is connected and configured for the current setup).

method FileSaveRaw

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT FileSaveRaw([in] BSTR Filename);
```

MATLAB:

```
Engine.FileSaveRaw('C:\test.lsraw');
```

Saves the raw data associated with a measurement to the specified .lsraw file.

method FileLoadRawAsMeasurement

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT FileLoadRawAsMeasurement([in] BSTR Filename);
```

MATLAB:

```
Engine.FileLoadRawAsMeasurement('C:\test.lsraw');
```

Loads a measurement raw data file that has been previously saved from the LS engine. The raw data will be immediately processed and the measurement results can be accessed using the `MeasurementResult` property. Please note that the immediate evaluation will use the parameters (Resolution/Smoothing, GenerateTE/TM, etc) as saved with the raw data. By executing `ReCalculate` method later, the current engine's settings can be applied instead.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

property ReturnLossModulePresent

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT ReturnLossModulePresent([out, retval] BOOL*
pVal);
```

MATLAB:

```
x = Engine.ReturnLossModulePresent;
```

Returns 1 if a return loss module is used in the current setup or 0 if not.

property ElectricalInputsPresent

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT ElectricalInputsPresent([out, retval] BOOL*
pVal);
```

MATLAB:

```
x = Engine.ElectricalInputsPresent;
```

Returns 1 if a power meter / SMU with electrical inputs is used in the current setup or 0 if not.

property BiasEnabledInputsPresent

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT BiasEnabledInputsPresent([out, retval] BOOL*
pVal);
```

MATLAB:

```
b = Engine.BiasEnabledInputsPresent;
```

Returns 1 if a power meter / SMU with bias enabled inputs is used in the current setup or 0 if not.

method GetBiasEnabled

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetBiasEnabled([in] LONG nDaq, [in] LONG nPort, [
out, retval ] BOOL * pVal);
```

MATLAB:

```
b = Engine.GetBiasEnabled(1,2);
```

Checks whether a given port of a given power meter / SMU has an electrical input that allows for bias adjustment.

Refer to property property PWMCount on page 191 and method GetPWMChannelCount on page 191 for input parameters nDaq and nPort.

method SetBiasVoltage

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetBiasVoltage([in] LONG nDaq, [in] LONG nPort, [in]
DOUBLE newVal);
```

MATLAB:

```
Engine.SetBiasVoltage(1,2,-1.15);
```

Sets the bias setting for a given channel of a given power meter / SMU. Use method GetBiasEnabled on page 199 to check whether that port supports bias adjustment.

Refer to property property PWMCount on page 191 and method GetPWMChannelCount on page 191 for input parameters nDaq and nPort.

method GetBiasVoltage

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetBiasVoltage([in] LONG nDaq, [in] LONG nPort, [
out, retval ] DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetBiasVoltage(1,2);
```

Returns the current bias setting for a given channel of a given power meter / SMU. Use method GetBiasEnabled on page 199 to check whether that port supports bias adjustment.

Refer to property property PWMCount on page 191 and method GetPWMChannelCount on page 191 for input parameters nDaq and nPort.

property BiasMode

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT BiasMode([out, retval] LONG* pVal);
```

```
put: HRESULT BiasMode([in] LONG Val);
```

MATLAB:

```
i= Engine.BiasMode;
```

```
Engine.BiasMode = i;
```

Sets or gets the behavior for bias voltage application. Set to 0/1/2 for Off/On/Auto. Refer to section Bias Settings on page 144 for further details.

method ZeroAllMPPMs

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT ZeroAllMPPMs(void);
```

MATLAB:

```
Engine.ZeroAllMPPMs;
```

Starts the zeroing procedure on all configured power meters / SMUs (and return loss modules, if present) in the current setup.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

property LambdaZeroingMode

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT LambdaZeroingMode([out, retval] LONG* pVal);
```

```
put: HRESULT LambdaZeroingMode([in] LONG Val);
```

MATLAB:

```
i= Engine.LambdaZeroingMode;
```

```
Engine.LambdaZeroingMode = i;
```

Sets or gets the behavior when a lambda zeroing operation is suggested by the software. Set to 0/1/2 for Automatically/Always Ask/Manual Only. Refer to Instrument Setup Parameters on page 43 for further details.

method ZeroAllTLS

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT ZeroAllTLS(void);
```

MATLAB:

```
Engine.ZeroAllTLS;
```

Performs a lambda zeroing operation on all tunable laser sources used in the current configuration.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method CalibrateRLM

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT CalibrateRLM(void);
```

MATLAB:

```
Engine.CalibrateRLM;
```

Starts the return loss module calibration procedure (if a return loss module is present in the current setup). Use `Engine.RLMCalReflectorValue` to enter the correct value for your reference reflection artifact.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method CalibrateRLMWithAcknowledgement

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT CalibrateRLMWithAcknowledgement(void);
```

MATLAB:

```
Engine.CalibrateRLMWithAcknowledgement();
```

Starts the return loss module calibration procedure (if a return loss module is present in the current setup) but waits for an additional acknowledgment for the current waveform range settings. Use `Engine.RLMCalReflectorValue` to enter the correct value for your reference reflection artifact.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

property RLMCalReflectorValue

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT RLMCalReflectorValue([out, retval] DOUBLE*
pVal);
```

```
put: HRESULT RLMCalReflectorValue([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.RLMCalReflectorValue;
```

```
Engine.RLMCalReflectorValue = d;
```

Sets or gets the return loss of the reference reflection artifact used in return loss calibration in dB.

method SetRLMSensitivity

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetRLMSensitivity([in] LONG nTLS, [in] DOUBLE
newVal);
```

MATLAB:

```
Engine.SetRLMSensitivity(1,1);
```

Sets the sensitivity of the return loss module (return path) in dBm corresponding to the specified TLS. If property PWMAutoRanging on page 193 is set to 1, this property will be adjusted automatically. This property has to be set in multiples of 10dBm (-60dBm to +10dBm).

Requires return loss module to be part of the optical setup.

method GetRLMSensitivity

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetRLMSensitivity([in] LONG nTLS, [ out, retval ]
DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetRLMSensitivity(1);
```

Gets the sensitivity of the return loss module (return path) in dBm corresponding to the specified TLS.

method SetRLMSensitivityMon

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetRLMSensitivityMon([in] LONG nTLS, [in] DOUBLE
newVal);
```

MATLAB:

```
Engine.SetRLMSensitivityMon(1,1);
```

Sets the sensitivity of the return loss module (monitor path) in dBm corresponding to the specified TLS. If property PWMAutoRanging on page 193 is set to 1, this property will be adjusted automatically. This property has to be set in multiples of 10dBm (-60dBm to +10dBm).

Requires a return loss module to be part of the optical setup.

method GetRLMSensitivityMon

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetRLMSensitivityMon([in] LONG nTLS, [ out, retval ]
DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetRLMSensitivityMon(1);
```

Gets the sensitivity of the return loss module (monitor path) in dBm corresponding to the specified TLS.

property RLMCalibrationFileList

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT RLMCalibrationFileList([out, retval] BSTR*
ReferenceList);
```

MATLAB:

```
s = Engine.RLMCalibrationFileList;
```

Returns a list of return loss module calibration files.

method DeleteRLMCalibrationFile

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT DeleteRLMCalibrationFile([in] BSTR Filename);
```

MATLAB:

```
Engine.DeleteRLMCalibrationFile('test.omr');
```

Deletes the specified return loss module calibration file.

property EventPropertiesChanged

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT EventPropertiesChanged([out, retval] LONG*
pVal);
```

MATLAB:

```
nEvent = Engine.EventPropertiesChanged;
```

This property is a 32 bit integer value. The value will be increased whenever the state of the engine has changed. By polling this value and observing the changes, a client can be notified of any change of the engine state.

property EventDataAcquisitionFinished

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT EventDataAcquisitionFinished([out, retval]
LONG* pVal);
```

MATLAB:

```
i = Engine.EventDataAcquisitionFinished;
```

Indicates the moment at which data acquisition is complete and data evaluation starts. In very time-critical scenarios, this event can be employed to start manual instrument control without interfering with engine-initiated instrument control.

property EventMeasurementFinished

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT EventMeasurementFinished([out, retval] LONG*
pVal);
```

MATLAB:

```
nEvent = Engine.EventMeasurementFinished;
```

This property is a 32 bit integer value. The value will be increased whenever the engine has finished a measurement. By polling this value and observing the changes, a client can be notified of any change of the engine state.

property EventReferenceOperationFinished

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT EventReferenceOperationFinished([out, retval]
LONG* pVal);
```

MATLAB:

```
nEvent = Engine.EventReferenceOperationFinished;
```

This property is a 32 bit integer value. The value will be increased whenever the engine has finished a reference operation, e.g. performing a reference sweep, deleting the reference data for certain ports, etc.. By polling this value and observing the changes, a client can be notified of any change of the engine state.

property ProtocolText

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT ProtocolText([out, retval] BSTR* pVal)
```

MATLAB:

```
ProtocolText = Engine.ProtocolText;
```

Returns a string containing all engine messages separated by a '|' character. If the number of messages exceeds 1000, the oldest message is deleted from the list for each new message.

property ProtocolMin

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT ProtocolMin([out, retval] LONG* pVal)
```

MATLAB:

```
First = Engine.ProtocolMin;
```

Returns the index to the first entry in the message list.

property ProtocolMax

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT ProtocolMax([out, retval] LONG* pVal)
```

MATLAB:

```
Last = Engine.ProtocolMax;
```

Returns the index to the most recent entry in the message list.

method GetProtocolTextAt

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetProtocolTextAt([in] LONG nMin,[in] LONG nMax,
[out,retval] BSTR* pVal);
```

MATLAB:

```
s = Engine.GetProtocolTextAt(Engine.ProtocolMin,
Engine.ProtocolMax);
```

Returns a portion of the message history defined by nMin und nMax.

method WriteCustomStatusMessage

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT WriteCustomStatusMessage([in] BSTR newVal);
```

MATLAB:

```
Engine.WriteCustomStatusMessage('yourmessage');
```

Enables you to write a custom message to the log file. This can help to monitor/troubleshoot custom applications.

property UserInputWaiting

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT UserInputWaiting([out, retval] BOOL* pVal);
```

```
put: HRESULT UserInputWaiting([in] BOOL Val);
```

MATLAB:

```
b = Engine.UserInputWaiting;
```

```
Engine.UserInputWaiting = b;
```

Returns 1 if the engine is waiting for a user input, e.g. to choose between different options or to acknowledge an error message. Returns 0 otherwise. Set to 0 if you have handled the user input (see method UserInputResponse on page 208).

property UserInputPrompt

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT UserInputPrompt([out, retval] BSTR* pVal)
```

MATLAB:

```
Prompt = Engine.UserInputPrompt;
```

Returns a string containing the prompt of the current user input request, if any.

property UserInputChoice

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT UserInputChoice([out, retval] BSTR* pVal)
```

MATLAB:

```
Choices = Engine.UserInputChoice;
```

Returns a string containing all possible responses to the user input requests separated by a '|' character. Each response contains a corresponding number to be used by `Engine.UserInputResponse` and the description of the response, e.g. for labeling a button. Number and description are separated by a ',' character.

method UserInputResponse

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT UserInputResponse([in] LONG nResponse);
```

MATLAB:

```
Engine.UserInputResponse(1);
```

Set the response to a user input request. Must be a number from the list obtained by `Engine.UserInputChoice`. Set `Engine.UserInputWaiting` to 0 after setting `Engine.UserInputResponse` for the engine to continue operation.

method SaveConfiguration

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SaveConfiguration([in] BSTR Filename, [in] BOOL
Overwrite);
```

MATLAB:

```
Engine.SaveConfiguration('C:\test.agconfig',1);
```

Saves the configuration corresponding to the current test setup as the specified .agconfig file. The boolean parameter specifies whether an already existing .agconfig file at the same location with the same name as the name specified in the command needs to be overwritten.

method SetDefaultSettings

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT SetDefaultSettings();
```

MATLAB:

```
Engine.SetDefaultSettings();
```

Reverts the current test settings to the default software configuration.

property ImmediateEvaluation

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT ImmediateEvaluation ([out, retval] BOOL* pVal);
put: HRESULT ImmediateEvaluation ([in] BOOL Val);
```

MATLAB:

```
b = Engine.ImmediateEvaluation;
Engine.ImmediateEvaluation= b;
```

If set to true, measurement data will be evaluated right after data acquisition.

Refer to the Advanced Parameters on page 61 for further details.

property ReferenceSopDriftCompensation

Supported Configurations: PLS

Type- Library:

```
get: HRESULT ReferenceSopDriftCompensation([out, retval]
BOOL* pVal);
```

```
put: HRESULT ReferenceSopDriftCompensation([in] BOOL Val);
```

MATLAB:

```
b= Engine.ReferenceSopDriftCompensation;
```

```
Engine.ReferenceSopDriftCompensation= b;
```

If set to true, the data-processing for polarization-resolved measurements can considerably reduce the degradation of reference data over time, mostly caused by temperature changes and, thus, can significantly increase the accuracy of measurement results. A reference measurement is required for this functionality.

Refer to the Advanced Parameters on page 61 for further details.

property WavelengthStepCompatibilityMode

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
get: HRESULT WavelengthStepCompatibilityMode([out, retval]
BOOL* pVal);
```

```
put: HRESULT WavelengthStepCompatibilityMode([in] BOOL
newVal);
```

MATLAB:

```
b= Engine.WavelengthStepCompatibilityMode;
```

```
Engine.WavelengthStepCompatibilityMode = b;
```

When set to true, the wavelength step size is defined by sweep rate and averaging time. Any step size defined by SetWavelengthStep() method will be ignored then.

Refer to the Advanced Parameters on page 61 for further details.

property PortSelectionCompatibilityMode

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT PortSelectionCompatibilityMode([out, retval]
BOOL* pVal);
```

```
put: HRESULT PortSelectionCompatibilityMode([in] BOOL
newVal);
```

MATLAB:

```
b= Engine.PortSelectionCompatibilityMode;
```

```
Engine.PortSelectionCompatibilityMode = b;
```

Power meter and SMU ports to be included in a measurement can be selected individually in the LS engine. Setting the property PortSelectionCompatibilityMode to 1 alters this behavior such that upon starting a regular (i.e., non-reference, non-il-de-embedding) measurement, the port selection will automatically be adjusted to the availability of reference data at that time. This will reflect both previously performed Take Reference, Copy, and Clear Reference operations as well as a different set of referenced ports loaded with a reference file.

property MinimumAutoRangeCompatibilityMode

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT MinimumAutoRangeCompatibilityMode([out, retval]
BOOL* pVal);
```

```
put: HRESULT MinimumAutoRangeCompatibilityMode([in] BOOL
newVal);
```

MATLAB:

```
b= Engine.MinimumAutoRangeCompatibilityMode;
```

```
Engine.MinimumAutoRangeCompatibilityMode = b;
```

When set to 1, the LS Engine does not use any ranges lower than -20dBm in power meter autoranging operations and, thus, behaves as the IL/PDL Engine did.

method ForcePWMInitialization

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT ForcePWMInitialization (void);
```

MATLAB:

```
Engine.ForcePWMInitialization;
```

Ensures that all power meter / SMU initialization steps are performed before the next measurement.

Refer to the Advanced Parameters on page 61 for further details.

method ForceAllInstrumentsInitialization

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT ForceAllInstrumentsInitialization (void);
```

MATLAB:

```
Engine.ForceAllInstrumentsInitialization;
```

Ensures that initialization steps for all instruments are performed before the next measurement.

Refer to the Advanced Parameters on page 61 for further details.

method ForcePOLInitialization

Supported Configurations: PLS, PMD

Type-Library:

```
HRESULT ForcePOLInitialization (void);
```

MATLAB:

```
Engine.ForcePOLInitialization;
```

Ensures that all the initialization steps for the polarization synthesizer are performed before the next measurement.

Refer to the Advanced Parameters on page 61 for further details.

method ForceTLSInitialization

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT ForceTLSInitialization (void);
```

MATLAB:

```
Engine.ForceTLSInitialization;
```

Ensures that all TLS initialization steps will be performed before the next measurement.

Refer to the Advanced Parameters on page 61 for further details.

method ForceSopSystemReCalculate

Supported Configurations: PLS, PMD

Type-Library:

```
HRESULT ForceSopSystemReCalculate(void);
```

MATLAB:

```
Engine.ForceSopSystemReCalculate;
```

Forces SOP system optimization before the next measurement.

method SetPWMSensitivity

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetPWMSensitivity([in] LONG tls, LONG port, DOUBLE
newVal);
```

MATLAB:

```
Engine.SetPWMSensitivity(1,2,10);
```

Sets the power meter / SMU sensitivity for the DUT sweep in dBm. If property PWMAutoRanging on page 193 is set to 1, this property will be adjusted automatically. This property has to be set in multiples of 10dBm.

method GetPWMSensitivity

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMSensitivity([in] LONG tls, LONG port, [ out,
retval ] DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetPWMSensitivity(1,2);
```

Gets the power meter / SMU sensitivity for the DUT sweep in dBm for the specified TLS and power meter / SMU port.

method SetPWMInclude

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetPWMInclude([in] LONG nPort, [in] BOOL newVal);
```

MATLAB:

```
Engine.SetPWMInclude(2,1);
```

Includes or excludes the specified port from subsequent operations (Taking references, running measurements, performing return loss calibrations, etc).

method GetPWMInclude

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMInclude([in] LONG nPort, [out, retval] BOOL *
pVal);
```

MATLAB:

```
b = Engine.GetPWMInclude(2);
```

Gets whether the specified port is included or excluded from subsequent operations (Taking references, running measurements, performing return loss calibrations, etc).

method GetPWMMinimumRange

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMMinimumRange([in] LONG nPort, [ out, retval ]
DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetPWMMinimumRange(3);
```

Gets the minimum power range of the specified PWM. The maximum is always +10dBm.

method GetRanges

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetRanges([in] LONG nPort, [ out, retval ]
SAFEARRAY(DOUBLE) * pVal);
```

MATLAB:

```
d_array = Engine.GetRanges(2);
```

Returns a list of supported power ranges for the specified port.

method SetPWMSensitivityRef

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetPWMSensitivityRef([in] LONG nTLS, [in] LONG
nPort, [in] DOUBLE newVal);
```

MATLAB:

```
Engine.SetPWMSensitivityRef(1,2,2);
```

Sets the power meter / SMU sensitivity for the reference sweep in dBm corresponding to the specified TLS and port. If property PWMAutoRanging on page 193 is set to 1, this property will be adjusted automatically. This property has to be set in multiples of 10dBm.

method GetPWMSensitivityRef

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMSensitivityRef([in] LONG nTLS, [in] LONG
nPort, [ out, retval ] DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetPWMSensitivityRef(1,2);
```

Gets the power meter / SMU sensitivity for the reference sweep in dBm corresponding to the specified TLS and port.

method GetPWMMaximumRange

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetPWMMaximumRange([in] LONG nPort, [ out, retval ]
DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetPWMMaximumRange(1);
```

Gets the maximum power range corresponding to the specified port of the power meter.

method SetPWMSensitivityAllPorts

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetPWMSensitivityAllPorts([in] LONG TLS, [in] DOUBLE
newVal);
```

MATLAB:

```
Engine.SetPWMSensitivityAllPorts(1,1);
```

Sets the power meter / SMU sensitivity for all the ports for the DUT sweep in dBm corresponding to the specified TLS.

If property PWMAutoRanging on page 193 is set to 1, this property will be adjusted automatically. This property has to be set in multiples of 10dBm.

method SetPWMSensitivityRefAllPorts

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetPWMSensitivityRefAllPorts([in] LONG TLS, [in]
DOUBLE newVal);
```

MATLAB:

```
Engine.SetPWMSensitivityRefAllPorts(1,1);
```

Sets the power meter / SMU sensitivity for all the ports for the reference measurement in dBm corresponding to the specified TLS.

method SetERPWMSensitivityAllPorts

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetERPWMSensitivityAllPorts([in] LONG TLS, [in]
DOUBLE newVal);
```

MATLAB:

```
Engine.SetERPWMSensitivityAllPorts(1,1);
```

Sets the power meter / SMU sensitivity for all the ports for the extinction ratio measurement in dBm corresponding to the specified TLS.

method ZeroPWMChannels

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT ZeroPWMChannels([in] SAFEARRAY(LONG) pVal);
```

MATLAB:

```
Engine.ZeroPWMChannels(int32[3;2]);
```

Performs a zeroing operation on the selected power meter ports. As a parameter, an integer array of port numbers (starting at 0 for port 1) must be provided. There will be a subsequent user input request to make sure that the corresponding ports are covered.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method StartILDeembeddingMasterPortMeasurement

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT StartILDeembeddingMasterPortMeasurement(void);
```

MATLAB:

```
Engine.StartILDeembeddingMasterPortMeasurement;
```

Starts an IL de-embedding master port measurement.
Refer to Performing Master Port Measurements on page 105 for details.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method StartILDeembeddingMasterPortMeasurementWithAcknowledgement

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT
StartILDeembeddingMasterPortMeasurementWithAcknowledgement(v
oid);
```

MATLAB:

```
Engine.StartILDeembeddingMasterPortMeasurementWithAcknowledg
ement();
```

Starts an IL de-embedding master port measurement but waits for an acknowledgment for the current wavelength range settings.
Refer to Performing Master Port Measurements on page 105 for details.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method StartILDeembeddingSpecificPathMeasurement

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT StartILDeembeddingSpecificPathMeasurement([in] BSTR
Filename, BOOL Overwrite);
```

MATLAB:

```
Engine.StartILDeembeddingSpecificPathMeasurement(Filename,
1);
```

Starts an IL de-embedding specific path measurement and saves resulting OMR file using the provided file name. The `Overwrite` argument is used to specify whether an existing file is overwritten automatically. Otherwise, there will be an error message and a user input prompt that must be handled accordingly, in case the target file exists. The `Filename` argument must contain extension (.omr), since this is not added automatically if omitted.

Refer to Performing Specific Path Measurements on page 106 for details.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method GetILDeembeddingFileList

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetILDeembeddingFileList([in] LONG
nPort,[out,retval] BSTR*pVal);
```

MATLAB:

```
ILDeembeddingFileList =
Engine.GetILDeembeddingFileList(nPort);
```

Returns a string containing all files to be used for IL de-embedding on the port defined by `nPort` (starting at 0 for port 1). File / path names are considered relative to value defined by the "property ILDeembeddingTargetFolder".

Refer to Applying IL De-embedding Data on page 107 for details.

method SetILDeembeddingFileList

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetILDeembeddingFileList([in] LONG nPort, BSTR
ILDeembeddingFileList);
```

MATLAB:

```
Engine.SetILDeembeddingFileList(nPort,
ILDeembeddingFileList);
```

Sets a string defining all the files to be used for IL de-embedding on the port defined by `nPort` (starting at 0 for port 1). File / path names are considered relative to value defined by "property ILDeembeddingTargetFolder".

Refer to Applying IL De-embedding Data on page 107 for further details, especially on how to combine multiple files that are to be applied on a single port and whether each one should be added to the overall correction data in an additive or subtractive manner.

property ILDeembeddingTargetFolder

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT ILDeembeddingTargetFolder([out, retval] BSTR*
pVal);
```

```
put: HRESULT ILDeembeddingTargetFolder([in] BSTR newVal);
```

MATLAB:

```
s = Engine.ILDeembeddingTargetFolder;
```

```
Engine.ILDeembeddingTargetFolder = s;
```

Defines the default folder for IL de-embedding data files.

Refer to Applying IL De-embedding Data on page 107 for details.

property EnableILDeembedding

Supported Configurations: PLS, FLS

Type-Library:

get: `HRESULT EnableILDeembedding([out, retval] BOOL*pVal);`

put: `HRESULT EnableILDeembedding([in] BOOL Val);`

MATLAB:

```
b = Engine.EnableILDeembedding;
Engine.EnableILDeembedding= b;
```

Defines whether IL de-embedding data will be applied to subsequent measurements (or recalculations).

Refer to Characterizing Components / Obtaining IL De-embedding Data on page 105 for details.

method SetStatic

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetStatic(void);
```

MATLAB:

```
Engine.SetStatic;
```

Performs a Static Mode operation, based on the "property StaticWavelength", "property StaticTLSPower", "property StaticPolarizationMode" and corresponding other properties. For all the modes that use Stokes parameters as target values, a polarization stabilizer operation is started on the N7786B/C. Use "method ReadCurrentStabilizerState" to check the current stabilizer state and "method StopStabilizer" to stop stabilizer operation.

Refer to Performing Static Mode Measurements on page 113 for details.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

method StopStabilizer

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT StopStabilizer(void);
```

MATLAB:

```
Engine.StopStabilizer;
```

Stops the N7786B/C stabilizing operation, in case it is currently active. Use "method ReadCurrentStabilizerState" to check the current stabilizer state.

Refer to Performing Static Mode Measurements on page 113 for details.

After running this method, you must ensure that the Busy property returns 0, before you run any subsequent method or get/set any other properties. (except for those related to user input handling)

property StaticWavelength

Supported Configurations: PLS, FLS

Type-Library:

get: `HRESULT StaticWavelength([out, retval] DOUBLE* pVal);`

put: `HRESULT StaticWavelength([in] DOUBLE Val);`

MATLAB:

```
d = Engine.StaticWavelength;
Engine.StaticWavelength = d;
```

Sets or gets the static mode wavelength in nm.

Refer to Performing Static Mode Measurements on page 113 for details.

property StaticTLSPower

Supported Configurations: PLS, FLS

Type-Library:

get: `HRESULT StaticTLSPower([out, retval] DOUBLE* pVal);`

put: `HRESULT StaticTLSPower([in] DOUBLE Val);`

MATLAB:

```
d = Engine.StaticTLSPower;
```

```
Engine.StaticTLSPower = d;
```

Sets or gets the static mode laser output power in dBm.

Refer to Performing Static Mode Measurements on page 113 for details.

property StaticPWMPort

Supported Configurations: PLS, FLS

Type-Library:

get: `HRESULT StaticPWMPort([out, retval] LONG* pVal);`

put: `HRESULT StaticPWMPort([in] LONG Val);`

MATLAB:

```
i = Engine.StaticPWMPort;
```

```
Engine.StaticPWMPort = i;
```

Sets or gets the power meter port to be considered for certain Static Mode modes, such as Last Measurement or OMR File. Refer to Performing Static Mode Measurements on page 113 for details.

As noted earlier, the port numbering starts from 0.

property StaticPolarizationMode

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT StaticPolarizationMode([out, retval] enum
eILPDLStatPolMode_t* pVal);
```

```
put: HRESULT StaticPolarizationMode([in] enum
eILPDLStatPolMode_t Val);
```

MATLAB:

```
mode = Engine.StaticPolarizationMode;
```

```
Engine.StaticPolarizationMode = mode;
```

```
Engine.StaticPolarizationMode =
'eILPDLStatPolModeStokesParameters';
```

```
Engine.StaticPolarizationMode = 4;
```

Enumerator names and values for `eILPDLStatPolMode_t`:

eILPDLStatPolModeMeasurementMaxMin	0
eILPDLStatPolModeMeasurementPSP	1
eILPDLStatPolModeFileMaxMin	2
eILPDLStatPolModeFilePSP	3
eILPDLStatPolModeFilePSP	4
eILPDLStatPolModeWaveplateOrientations	5

Sets or gets the Static Mode operation to be performed, once "method SetStatic" is executed.

Refer to Static Mode Modes on page 114 for details.

property StaticPolarizationMaxMinMode

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT StaticPolarizationMaxMinMode([out, retval] enum
eILPDLStatPolMaxMinMode_t* pVal);
```

```
put: HRESULT StaticPolarizationMaxMinMode([in] enum
eILPDLStatPolMaxMinMode_t Val);
```

MATLAB:

```
mode = Engine.StaticPolarizationMaxMinMode;
```

```
Engine.StaticPolarizationMaxMinMode = mode;
```

```
Engine.StaticPolarizationMaxMinMode = 'eILPDLStatPolMax';
```

```
Engine.StaticPolarizationMaxMinMode = 0;
```

Enumerator names and values for `eILPDLStatPolMaxMinMode_t`:

eILPDLStatPolMax	0
eILPDLStatPolMin	1

Defines whether to optimize for maximum or minimum transmission when executing "method SetStatic" with "property StaticPolarizationMode" set to `eILPDLStatPolModeMeasurementMaxMin` or `eILPDLStatPolModeFileMaxMin`.

Refer to Static Mode Modes on page 114 for details.

property StaticPolarizationPSPMode

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT StaticPolarizationPSPMode([out, retval] enum
eILPDLStatPolPSPMode_t* pVal);
```

```
put: HRESULT StaticPolarizationPSPMode([in] enum
eILPDLStatPolPSPMode_t Val);
```

MATLAB:

```
mode = Engine.StaticPolarizationPSPMode;
```

```
Engine.StaticPolarizationPSPMode = mode;
```

```
Engine.StaticPolarizationPSPMode = ' eILPDLStatPolPSP1;
```

```
Engine.StaticPolarizationPSPMode = 0;
```

Enumerator names and values for `eILPDLStatPolPSPMode_t`:

`eILPDLStatPolPSP1`	0
`eILPDLStatPolPSP2`	1

Defines to which principal state of polarization (regarding PDL) the polarization is aligned when executing "method SetStatic" with "property StaticPolarizationMode" set to `eILPDLStatPolModeMeasurementPSP` or `eILPDLStatPolModeFilePSP`.

Refer to Static Mode Modes on page 114 for details.

property StaticFilename

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT StaticFilename([out, retval] BSTR* pVal);
```

```
put: HRESULT StaticFilename([in] BSTR TargetFolder);
```

MATLAB:

```
s = Engine.StaticFilename;
```

```
Engine.StaticFilename = s;
```

Defines the file to be used when executing "method SetStatic" with "property StaticPolarizationMode" set to `eILPDLStatPolModeFileMaxMin` or `eILPDLStatPolModeFilePSP`.

Refer to Performing Static Mode Measurements on page 113 for details.

property StaticStokesParameters

Supported Configurations: PLS, FLS

Type-Library:

get: `HRESULT StaticStokesParameters([out, retval] SAFEARRAY(double)* pVal)`

put: `HRESULT StaticStokesParameters([in] SAFEARRAY(double) newVal)`

MATLAB:

```
d_array = Engine.StaticStokesParameters;
Engine.StaticStokesParameters = d_array;
```

Sets or gets the target normalized Stokes parameters (3-element double array) to be used when executing "method SetStatic" with "property StaticPolarizationMode" set to `eILPDLStatPolModeStokesParameters`.

Refer to Static Mode Modes on page 114 for details.

property StaticWaveplateOrientations

Supported Configurations: PLS, FLS

Type-Library:

get: `HRESULT StaticWaveplateOrientations([out, retval] SAFEARRAY(double)* pVal)`

put: `HRESULT StaticWaveplateOrientations([in] SAFEARRAY(double) newVal)`

MATLAB:

```
d_array = Engine.StaticWaveplateOrientations;
Engine.StaticWaveplateOrientations = d_array;
```

Sets or gets the target waveplate orientations in degrees (5-element double array) to be used when executing "method SetStatic" with "property StaticPolarizationMode" set to `eILPDLStatPolModeWaveplateOrientations`.

Refer to Static Mode Modes on page 114 for details.

method ReadCurrentStokesParameters

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT ReadCurrentStokesParameters([out, retval]
SAFEARRAY(double)* pVal)
```

MATLAB:

```
d_array = Engine.ReadCurrentStokesParameters();
```

Gets the current Stokes parameters (3-element double array), as measured by the N7786B/C polarimeter at the time of executing this method.

Refer to Current Value Operations on page 119 for details.

property ReadCurrentWaveplateOrientations

Supported Configurations: PLS, FLS

Type-Library:

```
get: HRESULT ReadCurrentWaveplateOrientations([out, retval]
SAFEARRAY(double)* pVal)
```

MATLAB:

```
d_array = Engine.ReadCurrentWaveplateOrientations;
```

Gets the current waveplate orientations in degrees (5 or 6-element double array), as currently set at the N7786B/C polarization-controller at the time of executing this method.

Refer to Current Value Operations on page 119 for details.

method ReadCurrentStabilizerState

Supported Configurations: PLS, FLS

Type-Library:

get: `HRESULT ReadCurrentStabilizerState([out, retval] BOOL* pVal)`

MATLAB:

```
b = Engine.ReadCurrentStabilizerState();
```

Gets the N7786B/C polarization stabilizer state at the time of executing this method.

Refer to Current Value Operations on page 119 for details.

method SetERPWMSensitivity

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetERPWMSensitivity([in] LONG nTLS, [in] LONG nPort,
[in] DOUBLE newVal);
```

MATLAB:

```
Engine.SetERPWMSensitivity(1,1,1);
```

Sets the power meter / SMU sensitivity for the Polarization Extinction Ratio measurement in dBm corresponding to the specified TLS.

method GetERPWMSensitivity

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetERPWMSensitivity([in] LONG nTLS, [in] LONG nPort,
[ out, retval ] DOUBLE * pVal);
```

MATLAB:

```
d = Engine.GetERPWMSensitivity(1,1);
```

Gets the power meter / SMU sensitivity for the Polarization Extinction Ratio measurement in dBm corresponding to the specified TLS and specified port.

method SetERTLSPower

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetERTLSPower([in] LONG nTLS, [in] DOUBLE newVal);
```

MATLAB:

```
Engine.SetERTLSPower(1,1);
```

Sets the laser power in mW for use in Polarization Extinction Ratio measurements corresponding to the specified TLS.

method GetERTLSPower

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetERTLSPower([in] LONG nTLS, [ out, retval ] DOUBLE
* pVal);
```

MATLAB:

```
d = Engine.GetERTLSPower(1);
```

Gets the laser power in mW for use in Polarization Extinction Ratio measurements corresponding to the specified TLS.

method SetERWavelengths

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetERWavelengths([in] LONG nTLS, [in]
SAFEARRAY(DOUBLE) newVal);
```

MATLAB:

```
Engine.SetERWavelengths(1,int32([1,2]));
```

Sets a list of wavelengths to be used for the polarization extinction ratio measurement for the specified TLS.

method GetERWavelengths

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetERWavelengths([in] LONG nTLS, [ out, retval ]
SAFEARRAY(DOUBLE) * pVal);
```

MATLAB:

```
d = Engine.GetERWavelengths(1);
```

Gets the list of wavelengths used for the polarization extinction ratio measurement for the specified TLS.

method SetERUseSweepRange

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT SetERUseSweepRange([in] LONG tls, BOOL newVal);
```

MATLAB:

```
Engine.SetERUseSweepRange;
```

If set to true, Polarization Extinction Ratio measurements will use the Start and Stop, parameters of the sweep measurement instead of the corresponding Engine.ERWavelengthStart / Stop parameters.

method GetERUseSweepRange

Supported Configurations: PLS, FLS

Type-Library:

```
HRESULT GetERUseSweepRange([in] LONG tls, [ out, retval ]
BOOL * pVal);
```

MATLAB:

```
b = Engine.GetERUseSweepRange(1);
```

Returns whether Polarization Extinction Ratio measurements will use the Start and Stop parameters of the sweep measurement instead of the corresponding Engine.ERWavelengthStart / Stop parameters.

property UsePolarizationResolvedMeasurement

Supported Configurations: PLS

Type-Library:

```
get: HRESULT UsePolarizationResolvedMeasurement([out,
retval] BOOL* pVal);
```

```
put: HRESULT UsePolarizationResolvedMeasurement([in] BOOL
newVal);
```

MATLAB:

```
b = Engine.UsePolarizationResolvedMeasurement;
```

```
Engine.UsePolarizationResolvedMeasurement = b;
```

Specifies whether measurements should be polarization-resolved or not, in case an N7786C instrument is part of the current configuration. Enabling this will cause generation of traces such as PDL and TE/TM. Disabling it will reduce measurement data file size and can speed up evaluation (when using similar step sizes).

method GetPolarizationInstrumentPresent

Supported Configurations: PLS, FLS, PMD

Type-Library:

```
HRESULT GetPolarizationInstrumentPresent([ out, retval ]
BOOL * pVal);
```

MATLAB:

```
b = Engine.GetPolarizationInstrumentPresent();
```

Returns whether an N7786B/C instrument is present in the current configuration.

method SetPolarimeterGain

Supported Configurations: PLS, PMD

Type-Library:

```
HRESULT SetPolarimeterGain([in] LONG nTLS, LONG newVal);
```

MATLAB:

```
Engine.SetPolarimeterGain(1,5);
```

When using a non-PMD configuration, sets the N7786B/C polarimeter gain for the specified TLS (device under test and reference measurements).

When using a PMD configuration, sets the N7788C polarimeter gain for the internal path measurement for the specified TLS (device under test and reference measurements).

method GetPolarimeterGain

Supported Configurations: PLS, PMD

Type-Library:

```
HRESULT GetPolarimeterGain([in] LONG nTLS, [ out, retval ]
LONG * pVal);
```

MATLAB:

```
d = Engine.GetPolarimeterGain(1);
```

When using a non-PMD configuration, gets the N7786B/C polarimeter gain for the specified TLS (device under test and reference measurements).

When using a PMD configuration, gets the N7788C polarimeter gain for the internal path measurement for the specified TLS (device under test and reference measurements).

method SetPolarimeterGainExternalPath

Supported Configurations: PMD

Type-Library:

```
HRESULT SetPolarimeterGainExternalPath([in] LONG nTLS, LONG
newVal);
```

MATLAB:

```
Engine.SetPolarimeterGainExternalPath(1,5);
```

(For PMD configurations only) Sets the gain value for the N7788C polarimeter for the device under test measurements for the specified TLS.

method GetPolarimeterGainExternalPath

Supported Configurations: PMD

Type-Library:

```
HRESULT GetPolarimeterGainExternalPath([in] LONG nTLS, [
out, retval ] LONG * pVal);
```

MATLAB:

```
d = Engine.GetPolarimeterGainExternalPath(1);
```

(For PMD configurations only) Gets the gain value for the N7788C polarimeter for the device under test measurements for the specified TLS.

method SetPolarimeterGainRef

Supported Configurations: PMD

Type-Library:

```
HRESULT SetPolarimeterGainRef([in] LONG nTLS, LONG newVal);
```

MATLAB:

```
Engine.SetPolarimeterGainRef(1,5);
```

(For PMD configurations only) Sets the gain value for the N7788C polarimeter for the reference measurements for the specified TLS.

method GetPolarimeterGainRef

Supported Configurations: PMD

Type-Library:

```
HRESULT GetPolarimeterGainRef([in] LONG nTLS, [ out, retval
] LONG * pVal);
```

MATLAB:

```
d = Engine.GetPolarimeterGainRef(1);
```

(For PMD configurations only) Gets the gain value for the N7788C polarimeter for the reference measurements for the specified TLS.

property FiberLength

Supported Configurations: PMD

Type-Library:

```
get: HRESULT FiberLength([out, retval] DOUBLE* pVal);
```

```
put: HRESULT FiberLength([in] DOUBLE newVal);
```

MATLAB:

```
d = Engine.FiberLength;
```

```
Engine.FiberLength = d;
```

Specifies the length of the DUT in km. This is used for computing the PMD coefficient in ps/sqrt(km) from a DGD-over-wavelength measurement.

property AutoResolution

Supported Configurations: PMD

Type-Library:

```
get: HRESULT AutoResolution([out, retval] BOOL* pVal);
```

```
put: HRESULT AutoResolution([in] BOOL newVal);
```

MATLAB:

```
b= Engine.AutoResolution;
```

```
Engine.AutoResolution = b;
```

When set to true, the data-processing resolution (smoothing) parameter will automatically be chosen based on the coarsely estimated DGD/PM of the device under test.

Alphabetical Automation Index

V

W

Z

Keysight N7700 Photonic Application Suite

LS Engine User's Guide

5 Troubleshooting

KEYSIGHT TECHNOLOGIES

Symptoms and Solutions

No measurement data shown

Possible Reason	Solution
An error occurred during the measurement, preventing the software from fully evaluating the data.	Check the output in the measurement status box for errors.
The trace contains invalid data only.	Under certain circumstances (e.g. certain over-range conditions, or bad/outdated power meter / SMU or TLS zeroing data, certain regions may contain NaN values instead of reasonable data. NaN values are not displayed in the engine GUIs / FileViewer.
There is no reference data available for the specific port(s).	Check Port /Reference Manager.
The specific graph type is not shown in the graph area of the engine GUI / File Viewer.	Especially when switching between different PAS engines or using the File Viewer for viewing data from different Photonic Application Suite (PAS) engines, often different graph types are selected to be shown in the graph area. This is done by expanding the measurement in the browser tree, right-clicking the corresponding data type (e.g. Insertion Loss or PDL), then selecting **View Graph**. Usually unused graphs are removed from the graph area then, by right-clicking into one of the graphs and clicking **Remove Graph**. This view configuration is shared by the File Viewer and the engine GUIs, so it can happen that the graph types used by the current engine have been removed from the graph area from within another engine. See the N7700 User's Guide for further details.

Possible Reason	Solution
Reference data is invalid (The Port/Reference Manager may contain NaN strings as average port power)	Clear the reference or take new reference.
IL de-embedding data is invalid (In case of omr files, this can be checked in FileViewer)	Repeat the measurement with IL de-embedding control set to off. If that helps, then replace with valid IL de-embedding data files and turn IL de-embedding back on.
In multi-TLS setup, the TLS/switch port mapping might be incorrect	Change switch port settings in engine configuration or run the Configuration Wizard and change laser sequence accordingly.

Measurement Status Box not visible

Possible Reason	Solution
Window minimized.	If you minimized the status window, it will stay that way until a user input is required. You can get it back to front anytime by clicking its entry in the task bar.
Window in background.	Unless “always on top” is selected, the measurement status box may get behind other windows. You can get it back to front by clicking its entry in the task bar.
Window closed.	If you closed the status window, it will stay that way until a user input is required. You can get it back by selecting **Measurement Status** from the **View** menu.
Window position stored in screen region that is currently invisible.	This could happen when working with multiple screens or varying screen resolutions. By selecting **Reset Status Window Position** from the **View** menu, the measurement status box position will be reset to the current top left corner of the engine window.

The list of values for a drop-down control appears invisible or displaced

Possible Reason	Solution
Different scaling factors for the different screens in some multi-screen setups.	Ensure that the scaling factor for all the screens is consistent.
	Consider using a single screen setup, define another screen as the default screen, or start the engine from an Explorer window on the screen that the engine GUI is intended to show on.
The engine window is maximized.	To make the invisible list of values appear on the screen, restore the window to its normal size ensuring that it is not too close to the top left edge of the screen. Then set the required value in the drop-down control. Note that the list of values will still appear displaced.
	If the list of values is visible (although displaced), select the drop-down list and use the cursor (arrow) and Enter keys to navigate to and select the required value.

When using N774xA/N774xC instruments:

- **Autoranging for an apparently fine port/trace keeps toggling the range over and over**
- **Glitches in steep transitions, e.g., when measuring filter devices**
- **Regions of missing trace data (containing NaNs), although the power range is chosen appropriately**

Possible Reason	Solution
	Disable the *Use MPPM auto gain* setting on the Instrument Setup tab.

COM API registration fails

Possible Reason	Solution
MATLAB Compiler Runtime (MCR) has not been installed prior to measurement engine installation.	Register the COM API after later MCR installation or re-register for any other reasons, by running the **Re-register all N7700 COM APIs** program from the Start menu.

Instrument not found in Configuration Wizard

Possible Reason	Solution
PXIe instrument not found	Make sure that no other software, such as the soft front panel is accessing the instrument.
TCP/IP instrument not found	For certain instruments there can be a maximum number of concurrent TCP/IP sessions, so make sure to close any unnecessary connections, such as Interactive IO windows. As TCP/IP instruments can be accessed by any network user, restarting the instrument should help resetting all the connections.

Instrument timeout errors occur after a certain idle period of the LS Engine

Possible Reason	Solution
When running the LS Engine using instruments in certain remote scenarios and leaving the engine idle for an extended period of time (about an hour and longer), subsequent engine operations might result in timeout errors during instrument communication.	Such timeouts are recognized by the engine and upon retry, the instrument will communicate correctly once again (provided there is not an actual connection issue or the instrument has not been powered off). Please note that only one instrument connection is reset at a time, so depending on the number of the instruments involved, repeated retries (one per instrument) might be required.

Measurement fails or measurement results look strange (especially after interacting with the instruments in between LS Engine measurements)

Possible Reason	Solution
	Have the engine re-configure the instruments by clicking the **PWM Init**, **POL Init**, **TLS Init**, and **SOP System Init** buttons under the Advanced tab. Certain error messages might indicate which instrument group requires reconfiguration. When in doubt, just click all the buttons. When using the automation programs, use any of the following COM methods: - ForcePWMInitialization - ForcePOLInitialization - ForceTLSInitialization - ForceAllInstrumentsInitialization - ForceSopSystemReCalculate

NOTE A troubleshooting utility called **Get N7700 System Information** has been introduced to help Keysight technical support gather some vital details required to resolve problems with the software at the user end, engine startup problems, for instance. This utility is accessible from the Start menu and should be run to capture information that can be shared with Keysight technical support.

NOTE Consider using the ExtractAgconfigFromOmr utility for extracting measurement settings for a measurement for which you did not explicitly save the .agconfig file. For more information, see Creating .agconfig Files from Measurement Files on page 131.

Edition 6.0, August 2021

KEYSIGHT
TECHNOLOGIES

www.keysight.com