

**OM4106D and OM4006D  
Coherent Lightwave Signal Analyzer  
User Manual**



071-3160-01

**Tektronix**



**OM4106D and OM4006D  
Coherent Lightwave Signal Analyzer  
User Manual**

Copyright © Tektronix. All rights reserved. Licensed software products are owned by Tektronix or its subsidiaries or suppliers, and are protected by national copyright laws and international treaty provisions.

Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specifications and price change privileges reserved.

TEKTRONIX and TEK are registered trademarks of Tektronix, Inc.

MATLAB is a registered trademark of The MathWorks, Inc.

LabVIEW is a trademark of National Instruments, Inc.

Intel and Pentium are registered trademarks of the Intel Corporation.

Other product and company names listed are trademarks and trade names of their respective companies.

### **Contacting Tektronix**

Tektronix, Inc.  
14150 SW Karl Braun Drive  
P.O. Box 500  
Beaverton, OR 97077  
USA

For product information, sales, service, and technical support:

- In North America, call 1-800-833-9200.
- Worldwide, visit [www.tektronix.com](http://www.tektronix.com) to find contacts in your area.

## Warranty

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product. Parts, modules and replacement products used by Tektronix for warranty work may be new or reconditioned to like new performance. All replaced parts, modules and products become the property of Tektronix.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. Customer shall be responsible for packaging and shipping the defective product to the service center designated by Tektronix, with shipping charges prepaid. Tektronix shall pay for the return of the product to Customer if the shipment is to a location within the country in which the Tektronix service center is located. Customer shall be responsible for paying all shipping charges, duties, taxes, and any other charges for products returned to any other locations.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Tektronix shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair or service the product; b) to repair damage resulting from improper use or connection to incompatible equipment; c) to repair any damage or malfunction caused by the use of non-Tektronix supplies; or d) to service a product that has been modified or integrated with other products when the effect of such modification or integration increases the time or difficulty of servicing the product.

THIS WARRANTY IS GIVEN BY TEKTRONIX WITH RESPECT TO THE PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

[W2 – 15AUG04]



---

# Table of Contents

Important safety information .....	vi
General safety summary .....	vi
Service safety summary .....	viii
Terms in this manual .....	ix
Symbols and terms on the product .....	ix
Compliance information .....	xii
EMC compliance .....	xii
Safety compliance .....	xiii
Environmental considerations .....	xv
Preface .....	xvii
Supported products .....	xvii
Getting started .....	1
Product description .....	1
Accessories .....	2
Options .....	3
Initial product inspection .....	4
Environmental operating requirements .....	4
Power requirements .....	5
PC requirements .....	6
Software installation .....	6
Set the instrument IP address .....	8
Equipment setup .....	13
Operating basics .....	17
OM4000 controls and connectors .....	17
Software overview .....	18
The Laser Receiver Control Panel (LRCP) user interface .....	19
The OM4000 user interface (OUI) .....	23
Configuring the OM4000 user interface (OUI) .....	57
VISA connections .....	57
Non-VISA oscilloscope connections (Scope Service Utility) .....	59
Two-oscilloscope configuration .....	61
MATLAB .....	62
Taking measurements .....	63
Setting up your measurement .....	63
MATLAB Engine file .....	64
Taking measurements .....	65
Multicarrier support (MCS) option .....	66
Detailed configuration of experiments .....	77
References .....	77

Core processing software guide.....	79
Interaction with OUI .....	79
MATLAB variables.....	80
MATLAB functions .....	81
Signal processing steps in CoreProcessing.....	82
Block processing.....	87
Alerts management .....	88
Core Processing function reference.....	91
AlignTribes .....	91
ApplyPhase.....	94
ClockRetime.....	94
DiffDetection.....	95
EstimateClock.....	97
EstimatePhase .....	99
EstimateSOP .....	100
MaskCount .....	101
GenPattern.....	102
Jones2Stokes .....	103
JonesOrth .....	103
LaserSpectrum .....	104
QDecTh.....	104
zSpectrum.....	106
Appendix A: MATLAB variables used by core processing.....	107
Appendix B: Alerts.....	109
Appendix C: Calibration and adjustment (RT oscilloscope).....	111
Calibration and adjustment (RT) .....	111
Hybrid calibration (RT) .....	115
Absolute power calibration .....	119
Laser linewidth factor .....	120
Receiver equalization.....	120
Appendix D: Automatic receiver deskew.....	123
Appendix E: Equivalent-Time (ET) oscilloscope measurements .....	125
Configuring hardware (ET).....	125
Configuring the software (ET) .....	128
OM4000 User Interface (OUI) (ET).....	130
Calibration and adjustment (ET) .....	132
Setting up an ET Oscilloscope .....	142
Matlab Engine file (ET) .....	143
Taking measurements (ET) .....	144
OUI overview (ET).....	144
OUI Controls panel (ET).....	144



---

Analysis Parameters window (ET).....	145
Appendix F: Configuring two Tektronix 70000 series oscilloscopes .....	147
Oscilloscope settings .....	149
OUI settings for 2-oscilloscope operation .....	152
Appendix G: The automated test equipment (ATE) interface .....	155
The LRCP ATE interface .....	155
The OUI4000 ATE interface.....	161
ATE functionality in MATLAB .....	167
Building an OM4006 ATE client in VB.NET .....	169
Appendix H: Cleaning and maintenance.....	177
Cleaning .....	177
Maintenance .....	177
Index	

## List of Figures

Figure 1: Real-time (RT) oscilloscope setup diagram .....	13
Figure 2: Equivalent-time (ET) oscilloscope setup diagram .....	14
Figure 3: Color grade constellation- fine traces .....	44
Figure 4: Color Key constellation .....	45
Figure 5: Multicarrier Setup button (Home ribbon) .....	66
Figure 6: Multicarrier setup window .....	67
Figure 7: Multicarrier spectrum context menu .....	69
Figure 8: Multicarrier spectrum plot .....	70
Figure 9: Multicarrier spectrum plot details .....	72
Figure 10: Multicarrier constellation plots .....	73
Figure 11: Multicarrier Eye diagrams plot .....	74
Figure 12: EVM vs. Channel plot .....	74
Figure 13: Q vs. Channel plot .....	75
Figure 14: Meas vs. Channel table .....	75
Figure 15: When adjusting the middle slider, watch the Y-Eye and Y-Const to minimize the signal in the Y-polarization .....	114
Figure 16: Final channel delay values provide only noise in Y polarization .....	114
Figure 17: Typical ET oscilloscope setup diagram .....	126
Figure 18: ChDelay(2) off by 2 ps causes curvature on constellation and signal on Q-Eye for 28 Gbps BPSK .....	134
Figure 19: When adjusting the middle slider, watch the Y-Eye and Y-Const to minimize the signal in the Y-polarization .....	136
Figure 20: Final channel delay values provide only noise in Y polarization .....	137
Figure 21: Equivalent-time (ET) oscilloscope setup diagram .....	142

## List of Tables

Table 1: Standard and optional accessories.....	2
Table 2: OM4000 options .....	3
Table 3: Software options .....	3
Table 4: OM4000 environmental requirements .....	4
Table 5: AC line power requirements .....	5
Table 6: Software install sequence: controller PC (PC or oscilloscope) .....	7
Table 7: Software install: oscilloscope.....	8
Table 8: OUI plots and measurements summary (real-time oscilloscopes).....	25
Table 9: Controls panel elements .....	30
Table 10: Record length and block size interaction behavior.....	32
Table 11: OUI: Analysis Parameters window .....	33
Table 12: Oscilloscope connectivity (VISA vs. Scope Service Utility).....	57
Table 13: Multicarrier spectrum menu choices (right-click).....	69
Table 14: Multicarrier spectrum controls .....	70
Table 15: Alert code descriptions.....	109

## Important safety information

This manual contains information and warnings that must be followed by the user for safe operation and to keep the product in a safe condition.

To safely perform service on this product, additional information is provided at the end of this section. (See page viii, *Service safety summary*.)

### General safety summary

Use the product only as specified. Review the following safety precautions to avoid injury and prevent damage to this product or any products connected to it. Carefully read all instructions. Retain these instructions for future reference.

Comply with local and national safety codes.

For correct and safe operation of the product, it is essential that you follow generally accepted safety procedures in addition to the safety precautions specified in this manual.

The product is designed to be used by trained personnel only.

Only qualified personnel who are aware of the hazards involved should remove the cover for repair, maintenance, or adjustment.

Before use, always check the product with a known source to be sure it is operating correctly.

This product is not intended for detection of hazardous voltages.

Use personal protective equipment to prevent shock and arc blast injury where hazardous live conductors are exposed.

When incorporating this equipment into a system, the safety of that system is the responsibility of the assembler of the system.

#### To avoid fire or personal injury

**Use proper power cord.** Use only the power cord specified for this product and certified for the country of use.

Do not use the provided power cord for other products.

**Ground the product.** This product is grounded through the grounding conductor of the power cord. To avoid electric shock, the grounding conductor must be connected to earth ground. Before making connections to the input or output terminals of the product, make sure that the product is properly grounded.

Do not disable the power cord grounding connection.

**Power disconnect.** The power cord disconnects the product from the power source. See instructions for the location. Do not position the equipment so that it

is difficult to disconnect the power cord; it must remain accessible to the user at all times to allow for quick disconnection if needed.

**Observe all terminal ratings.** To avoid fire or shock hazard, observe all ratings and markings on the product. Consult the product manual for further ratings information before making connections to the product.

Do not apply a potential to any terminal, including the common terminal, that exceeds the maximum rating of that terminal.

Do not float the common terminal above the rated voltage for that terminal.

The measuring terminals on this product are not rated for connection to mains or Category II, III, or IV circuits.

**Do not operate without covers.** Do not operate this product with covers or panels removed, or with the case open. Hazardous voltage exposure is possible.

**Avoid exposed circuitry.** Do not touch exposed connections and components when power is present.

**Do not operate with suspected failures.** If you suspect that there is damage to this product, have it inspected by qualified service personnel.

Disable the product if it is damaged. Do not use the product if it is damaged or operates incorrectly. If in doubt about safety of the product, turn it off and disconnect the power cord. Clearly mark the product to prevent its further operation.

Examine the exterior of the product before you use it. Look for cracks or missing pieces.

Use only specified replacement parts.

**Replace batteries properly.** Replace batteries only with the specified type and rating.

**Use proper fuse.** Use only the fuse type and rating specified for this product.

**Wear eye protection.** Wear eye protection if exposure to high-intensity rays or laser radiation exists.

**Do not operate in wet/damp conditions.** Be aware that condensation may occur if a unit is moved from a cold to a warm environment.

**Do not operate in an explosive atmosphere.**

**Keep product surfaces clean and dry.** Remove the input signals before you clean the product.

**Provide proper ventilation.** Refer to the installation instructions in the manual for details on installing the product so it has proper ventilation.

Slots and openings are provided for ventilation and should never be covered or otherwise obstructed. Do not push objects into any of the openings.

**Provide a safe working environment.** Always place the product in a location convenient for viewing the display and indicators.

Avoid improper or prolonged use of keyboards, pointers, and button pads. Improper or prolonged keyboard or pointer use may result in serious injury.

Be sure your work area meets applicable ergonomic standards. Consult with an ergonomics professional to avoid stress injuries.

Use care when lifting and carrying the product.

**Warning- Use correct controls and procedures.** Use of controls, adjustments, or procedures other than those listed in this document may result in hazardous radiation exposure.

**Do not directly view laser output.** Under no circumstances should you use any optical instruments to view the laser output directly.

## Service safety summary

The *Service safety summary* section contains additional information required to safely perform service on the product. Only qualified personnel should perform service procedures. Read this *Service safety summary* and the *General safety summary* before performing any service procedures.

**To avoid electric shock.** Do not touch exposed connections.

**Do not service alone.** Do not perform internal service or adjustments of this product unless another person capable of rendering first aid and resuscitation is present.

**Disconnect power.** To avoid electric shock, switch off the product power and disconnect the power cord from the mains power before removing any covers or panels, or opening the case for servicing.

**Use care when servicing with power on.** Dangerous voltages or currents may exist in this product. Disconnect power, remove battery (if applicable), and disconnect test leads before removing protective panels, soldering, or replacing components.

**Verify safety after repair.** Always recheck ground continuity and mains dielectric strength after performing a repair.

## Terms in this manual

These terms may appear in this manual:



**WARNING.** *Warning statements identify conditions or practices that could result in injury or loss of life.*



**CAUTION.** *Caution statements identify conditions or practices that could result in damage to this product or other property.*

## Symbols and terms on the product

These terms may appear on the product:

- DANGER indicates an injury hazard immediately accessible as you read the marking.
- WARNING indicates an injury hazard not immediately accessible as you read the marking.
- CAUTION indicates a hazard to property including the product.



When this symbol is marked on the product, be sure to consult the manual to find out the nature of the potential hazards and any actions which have to be taken to avoid them. (This symbol may also be used to refer the user to ratings in the manual.)

The following symbol(s) may appear on the product:



CAUTION  
Refer to Manual



Protective Ground  
(Earth) Terminal



Mains Disconnected  
OFF (Power)

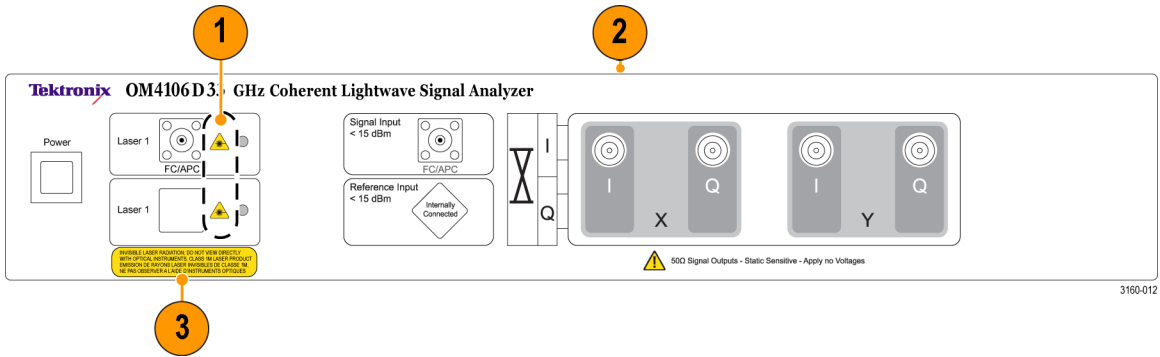




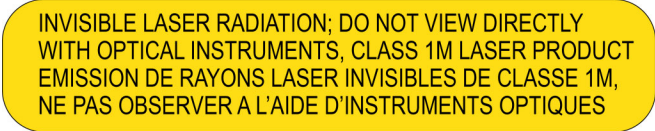
Mains Connected  
ON (Power)



Invisible Laser  
Radiation

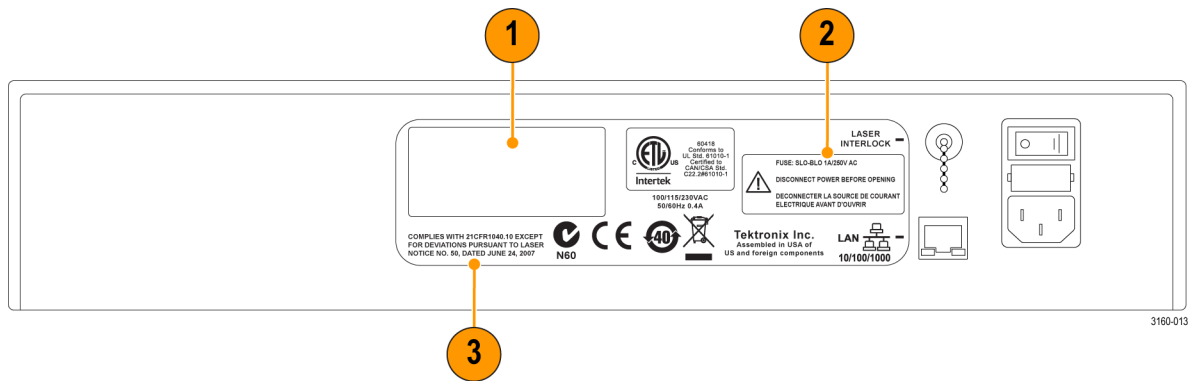
### Front panel labels



Item	Description
1	 Indicates the location of laser apertures
2	On inside cover of the instrument 
3	



Rear panel labels



Item	Description
1	Instrument model and serial number label
2	Fuse safety information
3	<b>COMPLIES WITH 21CFR1040.10 EXCEPT FOR DEVIATIONS PURSUANT TO LASER NOTICE NO. 50, DATED JUNE 24, 2007</b>

## Compliance information

This section lists the EMC (electromagnetic compliance), safety, and environmental standards with which the instrument complies.

### EMC compliance

#### EC Declaration of Conformity – EMC

Meets intent of Directive 2004/108/EC for Electromagnetic Compatibility. Compliance was demonstrated to the following specifications as listed in the Official Journal of the European Communities:

**EN 61326-1 2006.** EMC requirements for electrical equipment for measurement, control, and laboratory use. <sup>1 2 3</sup>

- CISPR 11:2003. Radiated and conducted emissions, Group 1, Class A
- IEC 61000-4-2:2001. Electrostatic discharge immunity
- IEC 61000-4-3:2002. RF electromagnetic field immunity
- IEC 61000-4-4:2004. Electrical fast transient / burst immunity
- IEC 61000-4-5:2001. Power line surge immunity
- IEC 61000-4-6:2003. Conducted RF immunity
- IEC 61000-4-11:2004. Voltage dips and interruptions immunity

**EN 61000-3-2:2006.** AC power line harmonic emissions

**EN 61000-3-3:1995.** Voltage changes, fluctuations, and flicker

#### European contact.

Tektronix UK, Ltd.  
Western Peninsula  
Western Road  
Bracknell, RG12 1RF  
United Kingdom

<sup>1</sup> This product is intended for use in nonresidential areas only. Use in residential areas may cause electromagnetic interference.

<sup>2</sup> Emissions which exceed the levels required by this standard may occur when this equipment is connected to a test object.

<sup>3</sup> For compliance with the EMC standards listed here, high quality shielded interface cables should be used.

**Australia / New Zealand  
Declaration of  
Conformity – EMC**

Complies with the EMC provision of the Radiocommunications Act per the following standard, in accordance with ACMA:

- CISPR 11:2003. Radiated and Conducted Emissions, Group 1, Class A, in accordance with EN 61326-1:2006.

**Australia / New Zealand contact.**

Baker & McKenzie  
Level 27, AMP Centre  
50 Bridge Street  
Sydney NSW 2000, Australia

## Safety compliance

This section lists the safety standards with which the product complies and other safety compliance information.

**EU declaration of  
conformity – low voltage**

Compliance was demonstrated to the following specification as listed in the Official Journal of the European Union:

Low Voltage Directive 2006/95/EC.

- EN 61010-1. Safety Requirements for Electrical Equipment for Measurement, Control, and Laboratory Use – Part 1: General Requirements.
- EN 60825-1. Safety of Laser Products - Part 1: Equipment classification and requirements.

**U.S. nationally recognized  
testing laboratory listing**

- UL 61010-1. Safety Requirements for Electrical Equipment for Measurement, Control, and Laboratory Use – Part 1: General Requirements.

**Canadian certification**

- CAN/CSA-C22.2 No. 61010-1. Safety Requirements for Electrical Equipment for Measurement, Control, and Laboratory Use – Part 1: General Requirements.

**Additional compliances**

- IEC 61010-1. Safety Requirements for Electrical Equipment for Measurement, Control, and Laboratory Use – Part 1: General Requirements.
- IEC 60825-1. Safety of Laser Products - Part 1: Equipment classification and requirements.
- This laser product complies with 21CFR1040.10 except for deviations pursuant to Laser Notice No. 50, dated June 24, 2007.

**Equipment type**

Test and measuring equipment.

<b>Safety class</b>	Class 1 – grounded product.
<b>Pollution degree descriptions</b>	<p>A measure of the contaminants that could occur in the environment around and within a product. Typically the internal environment inside a product is considered to be the same as the external. Products should be used only in the environment for which they are rated.</p> <ul style="list-style-type: none"> <li>■ Pollution degree 1. No pollution or only dry, nonconductive pollution occurs. Products in this category are generally encapsulated, hermetically sealed, or located in clean rooms.</li> <li>■ Pollution degree 2. Normally only dry, nonconductive pollution occurs. Occasionally a temporary conductivity that is caused by condensation must be expected. This location is a typical office/home environment. Temporary condensation occurs only when the product is out of service.</li> <li>■ Pollution degree 3. Conductive pollution, or dry, nonconductive pollution that becomes conductive due to condensation. These are sheltered locations where neither temperature nor humidity is controlled. The area is protected from direct sunshine, rain, or direct wind.</li> <li>■ Pollution degree 4. Pollution that generates persistent conductivity through conductive dust, rain, or snow. Typical outdoor locations.</li> </ul>
<b>Pollution degree rating</b>	Pollution degree 2 (as defined in IEC 61010-1). Rated for indoor, dry location use only.
<b>IP rating</b>	IP20 (as defined in IEC 60529).
<b>Measurement and overvoltage category descriptions</b>	<p>Measurement terminals on this product may be rated for measuring mains voltages from one or more of the following categories (see specific ratings marked on the product and in the manual).</p> <ul style="list-style-type: none"> <li>■ Category II. Circuits directly connected to the building wiring at utilization points (socket outlets and similar points).</li> <li>■ Category III. In the building wiring and distribution system.</li> <li>■ Category IV. At the source of the electrical supply to the building.</li> </ul> <hr/> <p><b>NOTE.</b> Only mains power supply circuits have an overvoltage category rating. Only measurement circuits have a measurement category rating. Other circuits within the product do not have either rating.</p> <hr/>
<b>Mains overvoltage category rating</b>	Overvoltage category II (as defined in IEC 61010-1).

## Environmental considerations

This section provides information about the environmental impact of the product.

### Product end-of-life handling

Observe the following guidelines when recycling an instrument or component:

**Equipment recycling.** Production of this equipment required the extraction and use of natural resources. The equipment may contain substances that could be harmful to the environment or human health if improperly handled at the product's end of life. To avoid release of such substances into the environment and to reduce the use of natural resources, we encourage you to recycle this product in an appropriate system that will ensure that most of the materials are reused or recycled appropriately.



This symbol indicates that this product complies with the applicable European Union requirements according to Directives 2002/96/EC and 2006/66/EC on waste electrical and electronic equipment (WEEE) and batteries. For information about recycling options, check the Support/Service section of the Tektronix Web site ([www.tektronix.com](http://www.tektronix.com)).

**Perchlorate materials.** This product contains one or more type CR lithium batteries. According to the state of California, CR lithium batteries are classified as perchlorate materials and require special handling. See [www.dtsc.ca.gov/hazardouswaste/perchlorate](http://www.dtsc.ca.gov/hazardouswaste/perchlorate) for additional information.

### Restriction of hazardous substances

This product is classified as an industrial monitoring and control instrument, and is not required to comply with the substance restrictions of the recast RoHS Directive 2011/65/EU until July 22, 2017.



# Preface

This manual describes how to install and operate the OM4000 instrument Coherent Lightwave Signal Analyzers.

## Supported products

The information in this manual applies to the following Tektronix products:

- OM4006D Coherent Lightwave Signal Analyzer
- OM4106D Coherent Lightwave Signal Analyzer
- OM1106 Coherent Lightwave Signal Analyzer stand-alone software (OUI) (included with OM4000 Series)





---

# Getting started

This section contains the following information to get you started using the instrument:

- Product description
- List of instrument accessories and options
- Initial product inspection
- Operating requirements (environmental, power)
- Software, network, and hardware setup

## Product description

The OM4106D and OM4006D Coherent Lightwave Signal Analyzer is a 1550 nm (C- and L-band) fiber-optic test system for visualization and measurement of complex modulated signals, offering a complete solution to test both coherent and direct-detected transmission systems. The OM4106D/OM4006D (referred to as the OM4000 in the rest of the manual) consists of a polarization- and phase-diverse receiver and analysis software, enabling simultaneous measurement of modulation formats important to advanced fiber communications, including polarization-multiplexed (PM-) QPSK.

The OM4000 User Interface (OUI) software performs all calibration and processing functions to enable real-time burst-mode constellation diagram display, eye-diagram display, Poincaré sphere, and bit-error detection.

A remote interlock for the laser, located on the rear of the unit, allows for remote locking of laser output.

You can use the OM4000 instrument along with a Tektronix OM2012 and OM2210, as well as supported real-time and equivalent-time oscilloscopes, for a complete optical calibration and testing system.

### Key features

- Coherent lightwave signal analyzer architecture is compatible with both real-time and equivalent-time oscilloscopes
- Complete coherent signal analysis system for polarization-multiplexed QPSK, offset QPSK, QAM, differential BPSK/QPSK, and other advanced modulation formats
- Displays constellation diagrams, phase eye diagrams, Q-factor, Q-plot, spectral plots, Poincaré sphere, signal vs. time, laser phase characteristics, BER, with additional plots and analyses available through the MATLAB interface
- Measures polarization mode dispersion (PMD) of arbitrary order with most polarization multiplexed signals

## Accessories

The following table lists the standard and optional accessories provided with the OM4000 instrument.

**Table 1: Standard and optional accessories**

<b>Accessory</b>	<b>Std.</b>	<b>Opt.</b>	<b>Tektronix part number</b>
OM4106D or OM4006D Coherent Lightwave Signal Analyzer	•		Varies by option
OM4006D and OM4106D Coherent Lightwave Signal Analyzer User Manual (this manual)	•		071-3160-xx
HRC and LRCP software USB flashdrive	•		650-5643-xx
HASP USB key	•		650-5642-xx
Ethernet cable, 7 ft.	•		174-6230-xx
RF Cable, 2.92 mm, 6 in. (4 cables)	•		174-6229-xx
Shorting cap for BNC interlock connector	•		131-8925-xx
Power cord (See page 3, <i>International power cord options.</i> )	•		Varies by option
Reply card	•		Not orderable
Cleaning swab	•		Not orderable
USB flashdrive case	•		016-2067-xx
China ROHS sheet	•		Not orderable
Patch Cord, Fiber, APC/APC, 8 in. (Opt EXT)		•	174-6231-xx

## Options

The following table lists some of the options that can be ordered with the OM4000. See the *Coherent Lightwave Signal Analyzer OM4000 Series Datasheet* (Tektronix part number 52W-27474-x) for a complete listing of options and recommended configurations.

**Table 2: OM4000 options**

Model	Option	Description
OM4006D 23 GHz	CC	Two C-band lasers
	CL	One C-band and one L-band laser
	LL	Two L-band lasers
OM4106D 33 GHz	CC	Two C-band lasers
	CL	One C-band and one L-band laser
	LL	Two L-band lasers

**Table 3: Software options**

Option	Description
QAM	Adds QAM and other software demodulators
MCS	Adds multicarrier superchannel support

**NOTE.** Option MCS requires that the oscilloscope or PC running the OM software have an nVidia graphics card installed

### International power cord options

All of the available power cord options listed below include a lock mechanism except as otherwise noted.

- Opt. A0 – North America power (standard)
- Opt. A1 – Universal EURO power
- Opt. A2 – United Kingdom power
- Opt. A3 – Australia power
- Opt. A4 – North America power (240 V)
- Opt. A5 – Switzerland power
- Opt. A6 – Japan power
- Opt. A10 – China power
- Opt. A11 – India power (no locking cable)
- Opt. A12 – Brazil power (no locking cable)

## Initial product inspection

Do the following when you receive your instrument:

1. Inspect the shipping carton for external damage, which may indicate damage to the instrument.
2. Remove the OM4000 instrument from the shipping carton and check that the instrument has not been damaged in transit. Prior to shipment the instrument is thoroughly inspected for mechanical defects. The exterior should not have any scratches or impact marks.

**NOTE.** *Save the shipping carton and packaging materials for instrument repackaging in case shipment becomes necessary.*

3. Verify that the shipping carton contains the basic instrument, the standard accessories and any optional accessories that you ordered. (See Table 1.)

Contact your local Tektronix Field Office or representative if there is a problem with your instrument or if your shipment is incomplete.

## Environmental operating requirements

Check that the location of your installation has the proper operating environment. (See Table 4.)



**CAUTION.** *Damage to the instrument can occur if this instrument is powered on at temperatures outside the specified ambient temperature range.*

**Table 4: OM4000 environmental requirements**


Parameter		Description
Temperature	Operating	+10 °C to +35 °C
	Nonoperating	-20 °C to +70 °C
Relative Humidity	Operating	15% to 80% (No condensation)
Altitude	Operating	To 2,000 m (6,560 feet) Maximum operating temperature decreases 1 °C each 300 m above 1.5 km.
	Nonoperating	To 15,000 m (49,212 feet)

Do not obstruct the fan so that there is an adequate flow of cooling air to the electronics compartment whenever the unit is operating. Leave space for cooling by providing at least 2 inches (5.1 cm) at rear of instrument for benchtop use. Also, provide sufficient rear clearance (approximately 2 inches) so that any cables are not damaged by sharp bends.


## Power requirements


**Table 5: AC line power requirements**

Parameter	Description
Line voltage range	100/115 VAC single phase 230 VAC single phase
Line frequency	50/60 Hz
Maximum current	0.4 A

 **WARNING.** To reduce the risk of fire and shock, ensure that the AC supply voltage fluctuations do not exceed 10% of the operating voltage range.

To avoid the possibility of electrical shock, do not connect your OM4000 to a power source if there are any signs of damage to the instrument enclosure.

 **WARNING.** Always connect the unit directly to a grounded power outlet. Operating the OM instrument without connection to a grounded power source could result in serious electrical shock.

 **CAUTION.** Protective features of the OM4000 instrument may be impaired if the unit is used in a manner not specified by Tektronix.

**Connecting the power cable.** Connect the power cable to the instrument first, and then connect the power cable to the AC power source. Install or position the OM4000 instrument so that you can easily access the rear-panel power switch.

Power on the instrument (rear-panel power switch and front-panel standby power switch). After powering on, make sure that the fan on the rear panel is working. If the fan is not working, turn off the power by disconnecting the power cable from the AC power source, and then contact your local Tektronix Field Office or representative.

## PC requirements

The equipment and DUT used with the OM4000 determine the controller PC requirements. Following are the requirements to use the OM4000 instruments or OM2210 Coherent Receiver Calibration Source:

Item	Description
Operating system	U.S.A. Microsoft Windows 7 64-bit U.S.A. Microsoft Windows XP Service Pack 3 32-bit (.NET 4.0 required)
Processor	Intel i7, i5 or equivalent; min clock speed 2 GHz Minimum: Intel Pentium 4 or equivalent
RAM	Minimum: 4 GB 64-bit releases benefit from as much memory as is available
Hard Drive Space	Minimum: 20 GB >300 GB recommended for large data sets
Video Card	nVidia dedicated graphics board w/ 512+ MB minimum. graphics memory <b>NOTE.</b> <i>The OUI color grade display feature, the MCS option, and the equivalent-time (ET) oscilloscope mode require that the oscilloscope or PC running the OM software have an nVidia graphics card installed</i>
Networking	Gigabit Ethernet (1 Gb/s) or Fast Ethernet (100 Mb/s)
Display	20" minimum, flat screen recommended for displaying multiple graph types when using with the OM4000 Software
Other Hardware	2 USB 2.0 ports
MATLAB Software	For Windows 7 (64-bit): MATLAB version 2011b (64-bit) For Windows XP (32-bit): MATLAB version 2009a (32-bit), .NET version 4 or later.
Adobe Reader	Adobe reader required for viewing PDF format files (release notes, installation instructions, user manuals).

## Software installation

The OM4000 requires several programs and drivers to be installed on your controlling PC (separate PC or oscilloscope) for proper operation. Install the programs listed in the following table, in the order indicated (starting from the top of table). All programs are on the OM4000 software USB flashdrive unless otherwise noted.

**NOTE.** *Read the installation notes or instructions that are in each application installation folder before installing each item of software. Only install the software that is appropriate for your OM instrument, PC, and oscilloscope configuration.*

## Install software on the controller PC

**Table 6: Software install sequence: controller PC (PC or oscilloscope)**

Program	Description	Path (from root directory of USB drive)
TekVISA	Instrument USB and Ethernet connectivity software.	OUI\ISSetupPrerequisites\TekVISA_v3.3.8\TekVISA\setup.exe <i><b>NOTE.</b> TekVISA is only required when using MSO/DSO70000 or 70000B series oscilloscopes, or when using the HRC software.</i>
LRCP	Laser Receiver Control Panel. Detects OM instruments on a network, controls laser and other hardware settings.	LRCP\LRCPSetup x.x.x.x.msi
MATLAB	Required for OUI.	Not part of the OM4000 software distribution. Contact The MathWorks, Inc. to obtain the MATLAB software. See PC requirements to determine the appropriate version of MATLAB for your PC.
OUI	OM4000 User Interface (OUI). The interface for using the OM4000 instrument to take and display measurements.	OUI\SetupOUI_x.x.x.x 32-bit OS.exe (Windows 7 32-bit or XP 32-bit; assumes Matlab 2009a installed) OUI\SetupOUI_x.x.x.x.exe (Windows 7 64-bit; either Matlab choice)
Power meter	Software and drivers to communicate with the instrument optical power meter. Install in the order listed. Only required for use with the HRC software.	HRC\ISSetupPrerequisites\ThorPowerMeter\setup.exe HRC\ISSetupPrerequisites\ThorPowerMeterDriver\setup.exe
HRC	Hybrid-Receiver Calibration software. Uses SQL to keep track of calibration configurations. SQL software installed automatically if not present on the PC.	HRC\Setup Optametra HRC_x.x.x.x.exe

## Install software on the oscilloscope

The Scope Service Utility (SSU) is required for MSO/DSO70000C and 70000D series real-time (RT) oscilloscopes, and for the DSA8300 and DSA8200 equivalent-time (ET) oscilloscopes.

Copy the appropriate Scope Service Utility software installation file (RT or ET) from the software USB flashdrive to the oscilloscope and double-click on the program to install it.

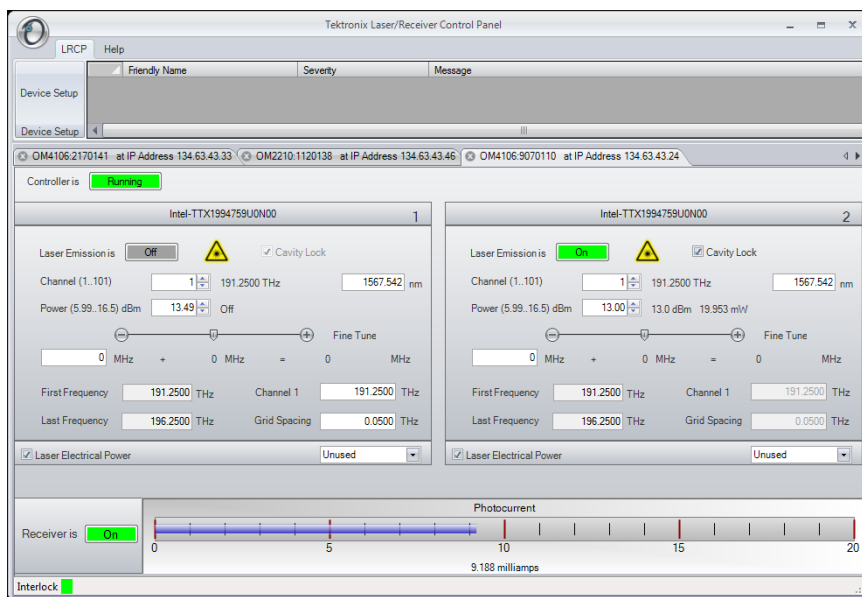
***NOTE.** Read the installation notes or instructions that are in each application installation folder before installing each item of software. Only install the software that is appropriate for your instrument, PC, and oscilloscope configuration.*

**Table 7: Software install: oscilloscope**

Program	Description	Path
Scope Service Utility	Enables collecting and analyzing coherent optical signals at 100Gs/s on four channels.  There is a separate program for real-time (RT) and equivalent-time (ET) oscilloscopes.	OUI\Tektronix Scope Service Utilityx.x.x.x.exe (install on RT oscilloscope only)  OUI\Tektronix Scope Service For ET Utilityx.x.x.x.exe (install on ET oscilloscope only)

## Set the instrument IP address

Use the Laser Receiver Control Panel (LRCP) application to verify and/or set the IP address of OM instruments (OM4106D, OM4006D, OM2210, OM2012) if required for your network test setup. All OM instruments must be set to the same network subnet (DHCP-enabled networks do this automatically) to communicate with each other using the LRCP and OM4000 User Interface (OUI) software.



Before using LRCP, you must make sure that IP addresses of the OM series instruments are set correctly to communicate with LRCP on your network. The following sections describe how to set the OM instrument IP addresses for use on DHCP and non-DHCP networks.

### Set IP address for DHCP-enabled network

The OM instruments are set with automatic IP assignment (DHCP) enabled by default. Therefore you do not need to specifically set the instrument IP address, as the DHCP server automatically assigns an IP address during instrument power-on.



The following procedure describes how to use LRCP software to verify connectivity of an OM instrument to a DHCP-enabled network.

Prerequisite: OM instrument, and the controller PC (with LRCP installed), both connected to the same DHCP-enabled network.

1. Connect the OM instrument to the DHCP-enabled network.
2. Power on the OM instrument with the rear power switch (set to **1**). The instrument queries the DHCP server to obtain an IP address. Wait until the front panel Power button light turns off indicating it is ready. Press the front panel Power button to enable the network connection (button light turns On).
3. On a PC connected to the same network as the OM instrument, start the **LRCP** program. (See page 19, *The Laser Receiver Control Panel (LRCP) user interface*.)
4. Enter password **1234** when requested.
5. When running LRCP for the first time after installation, click on the **Configuration/Device Setup** link on the application screen to open the Device Setup window. Otherwise click the **Device Setup** button (upper left of application window).
6. In the **Device Setup** dialog box, click the **Auto Configure** button. LRCP searches the network and lists any OM instruments that it detects. If no devices are detected, work with you IT resource to resolve the connection problem.
7. (optional) Use the **Friendly Name** field to create a custom label for each instrument. There is no limit to the size of the name you enter.
8. Click **OK** to close the configuration dialog box and return to the LRCP main window. The main LRCP window displays a tab for each instrument detected. Click on a tab to display the laser controls for that instrument. Refer to the LRCP documentation for help on using the software.

### Set IP address for a non-DHCP network

To connect the OM series instrument to a non-DHCP network, you must reset the default IP address and related settings on the OM instrument to match those of your non-DHCP network. All devices on this network (OM instruments, PCs and other remotely accessed instruments such as oscilloscopes) need the same subnet values (first three number groups of the IP address) to communicate, and a unique instrument identifier (the fourth number group of the IP address) to identify each instrument.

Work with your network administrator to obtain a unique IP address for each device. Your network administrator may need the MAC addresses of the computer, oscilloscope, and OM instrument. The MAC address is located on the OM instrument rear panel label.

---

**NOTE.** Make sure to record the IP addresses used for each OM instrument, or attach a label with the new IP address to the instrument.

---

If you are setting up a new isolated network just for controlling OM and associated instruments, Tektronix recommends using the OM instrument default IP subnet address of **172.17.200.XXX**, where XXX is any number between 0 and 255. Use the operating systems of the oscilloscope and computer to set their IP addresses.

---

**NOTE.** *If you need to change the default IP address of more than one OM instrument, you must connect each instrument separately to change the IP address.*

---

There are two ways to change the IP address of an OM instrument:

- Use LRCP on a PC connected to a DHCP-enabled network (easiest)
- Use LRCP on a PC set to the same IP address as the OM instrument to change the OM instrument IP address

**Use DHCP network to change instrument IP address.** To use a DHCP network to change the IP address of an OM series instrument:

1. Do steps 1 through 6 of the *Set network access (DHCP network)* procedure.
2. Click the **Set IP** button in the instrument list.
3. Use the dialog box to enter the new IP address and related parameters for the OM instrument.
4. Click **OK** to close the dialog box and set the IP address.
5. Exit the LRCP program.
6. Power off the OM instrument and connect it to the non-DHCP network.
7. Run LRCP and use the **Auto Config** button in the Device Setup dialog box to verify that the instrument is listed.

**Use direct PC connection to change instrument IP address.** To use a direct PC connection to change the default IP address of an OM series instrument, you need to:

- Install LRPC on the PC
- Use the Windows Network tools to set the IP address of the PC to match that of the current subnet setting of the OM series instrument whose IP address you need to change
- Connect the OM instrument directly to the PC, or through a hub or switch (not over a network)
- Use LRCP to change the OM instrument IP address

Do the following steps to use a direct PC connection to change the IP address of an OM series instrument:

---

**NOTE.** *The following instructions are for Windows 7.*

---

---

**NOTE.** *If you need to change the default IP address of more than one OM instrument using this procedure, you must connect each instrument separately to change the IP address.*

---

1. On the PC with LRCP installed, click **Start > Control Panel**.
2. Open the **Network and Sharing Center** link.
3. Click the **Manage Network Connections** link to list connections for your PC
4. Right-click the **Local Area Connection** entry for the Ethernet connection and select **Properties** to open the Properties dialog box.
5. Select **Internet Protocol Version 4** and click **Properties**.
6. Enter a new IP address for your PC, using the same first three numbers as used by the OM instrument. For example, **172.17.200.200**. This sets your PC to the same subnet (first three number groups) as the default IP address setting for the OM series instruments.
7. Click **OK** to set the new IP address.
8. Click **OK** to exit the Local Area Connection dialog box.
9. Exit the **Control Panel** window.
10. Connect the OM instrument to the PC.
11. Power on the OM instrument with the rear power switch (set to **1**). Wait until front panel Power button light turns off.
12. On the PC, start the **LRCP** program. (See page 19, *The Laser Receiver Control Panel (LRCP) user interface*.)
13. Enter password **1234** when requested.
14. Select **Configuration > Device Setup** from the menu to open the Device Setup window.
15. Click the **Auto Configure** button. LRCP lists the OM-series instrument connected to the PC. If LRCP does not list the connected instrument, verify that you entered a correct IP address into the PC and your Ethernet cable is good. If the IP address was entered correctly, you may need to connect the OM instrument to a DHCP network to determine if the IP address you used to set the computer was correct.

16. (optional) Use the **Friendly Name** field to create a custom label for each instrument. There is no limit to the size of the name you enter. Friendly Names are retained and are associated with the MAC address of each instrument.
17. Click the **IP address** button in the list.
18. Click **OK** to continue to the **Set IP Address** dialog box.
19. Set the address to an IP Address that is compatible with your network. For example, **172.17.200.040**
20. Edit the Gateway and Net Mask (obtain this information from your network support).
21. Click **Set IP** to exit the IP address dialog and return to the LRCP main window.

---

**NOTE.** *If you change the instrument to an IP address that is different than the Subnet of the PC, and click Set IP, the OM series instrument is no longer detectable or viewable to that PC and LRCP.*

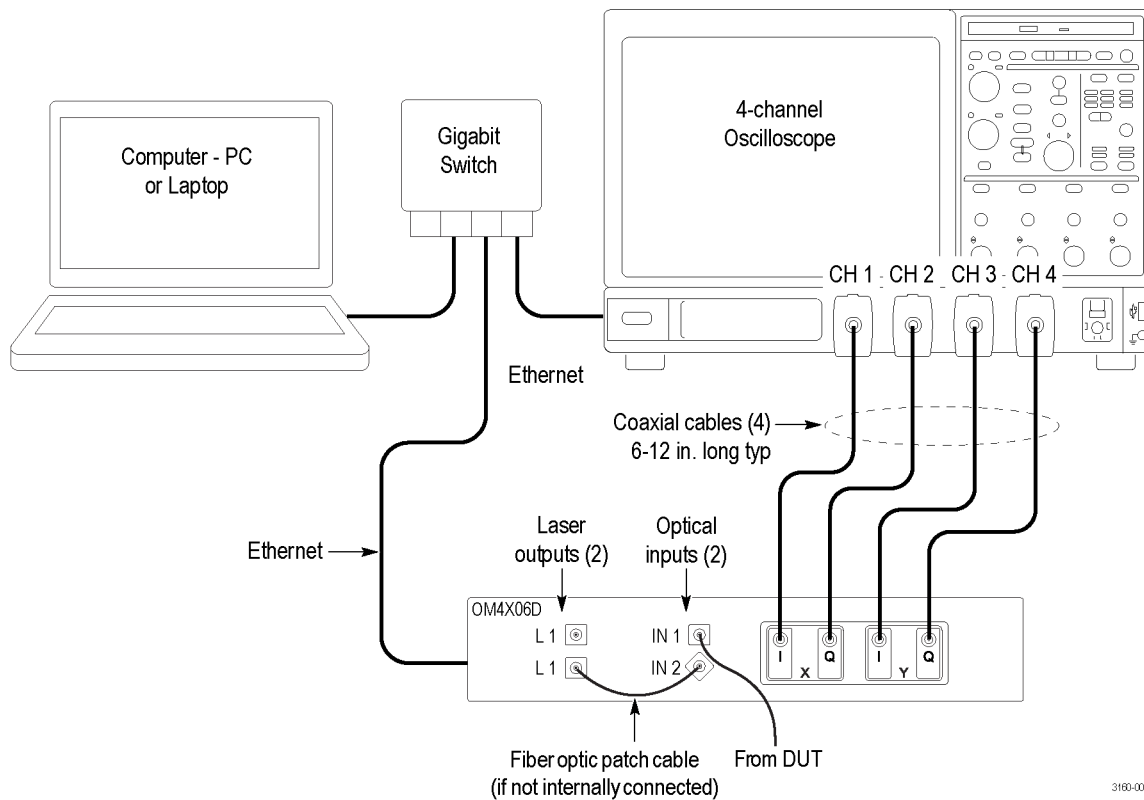
---

22. Exit the LRCP program.
23. Unplug the network cable from between the PC and the OM instrument.
24. Connect the OM instrument to the target network switch/router.
25. Run the LRCP software on the PC connected to the same network as the OM instrument.
26. Click **Device Setup**. Click **Auto Config** and verify that the instrument is detected and listed on the display.

## Equipment setup

### Real-time (RT) oscilloscope setup

See the following figure for how to connect the OM4000 instrument to take measurements with real-time oscilloscopes (Tektronix MSO/DSO70000 series). Appendix F contains information for how to use two real-time oscilloscopes to take measurements. (See page 147, *Configuring two Tektronix 70000 series oscilloscopes.*)

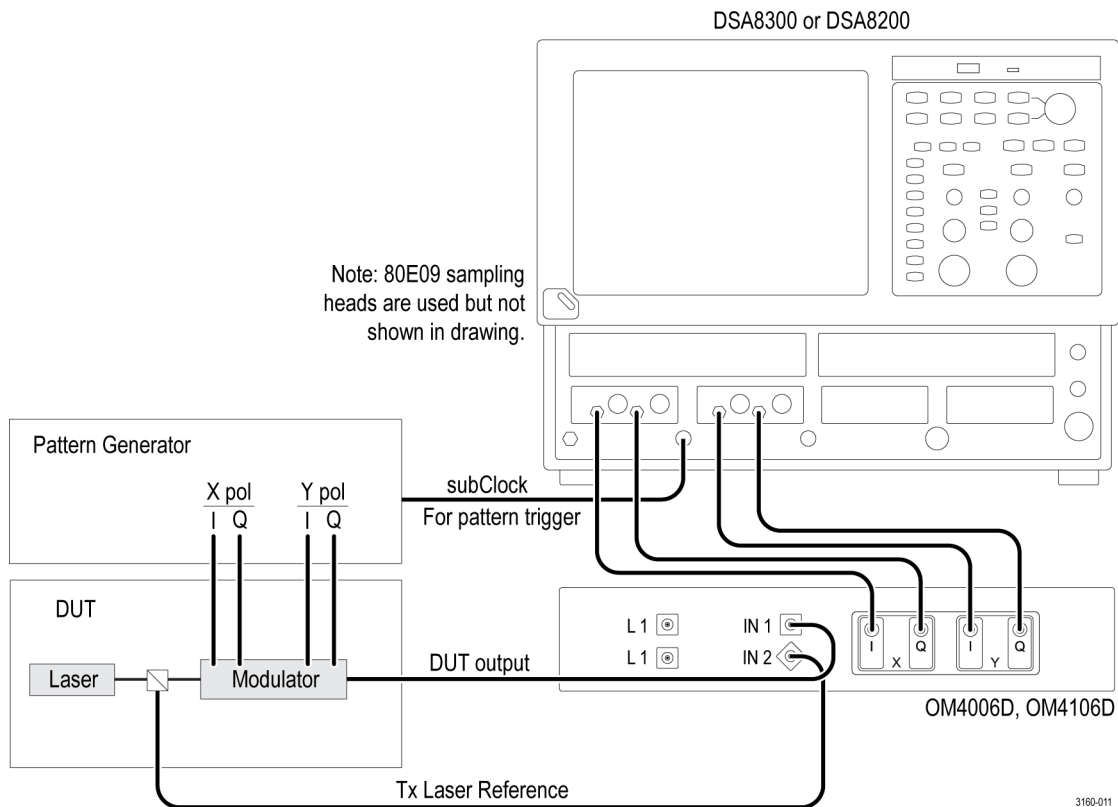


3180-001

Figure 1: Real-time (RT) oscilloscope setup diagram

**Equivalent-time (ET) oscilloscope setup**

See the following figure for how to connect the OM4000 instrument to take measurements with equivalent-time (ET) oscilloscopes (Tektronix DSA8300 or DSA8200 Digital Serial Analyzer sampling oscilloscopes). See Appendix E for information on using an ET oscilloscope to take measurements. (See page 125, *Equivalent-Time (ET) oscilloscope measurements.*)



**Figure 2: Equivalent-time (ET) oscilloscope setup diagram**

**Key ET setup and measurement difference.** The most important difference between the real-time (RT) and equivalent-time (ET) oscilloscope measurements is the need for a coherent reference signal for ET oscilloscopes. The TX reference signal is picked off before the modulator, using a PM fiber cable, with a total path length equal to the path from the splitting point to the Signal Input on the OM4000 Receiver. Use a SMF fiber cable to connect the DUT to the Signal Input connection on the OM4000.

Since the laser phase noise is a real-time quantity, it must be sufficiently suppressed so that it can be tracked in the available bandwidth of the ET scope.

As an example, consider a laser with frequency noise given by

$$f(t) = f_0 + f_D \sin 2\pi f_n t$$

If this laser signal is split and then input to the Signal Input and Reference Input of the OM4000, the resulting beat frequency will be

$$f_m(t) \approx (2\pi f_D f_n \Delta t) \sin 2\pi f_n t$$

so that while the modulating frequency,  $f_n$  is still the same, the frequency deviation,  $f_D$ , has been reduced by  $2\pi f_n \Delta t$  where  $\Delta t$  is the time difference for the two paths.

Some lasers can have frequency deviations in the 200 MHz range over 1 ms. To minimize the FM bandwidth after detection, reduce the frequency deviation to ~ 1 kHz. This is accomplished with  $\Delta t = 0.8$  ns or a path difference of 16 cm or less.

Generally speaking the ET performance will be best with a path difference less than 10 cm when possible. For lower noise lasers, path lengths differences up to 2 m can be tolerated.

**Connections** Make connections in the following order:

1. Ethernet connections and other computer connections
2. Power cord from the OM4000 instrument to the mains AC connector or to the instrument rack (if used)
3. Power cord from rack (if used) to mains (keeping main front panel switch off)
4. RF connections (the four coaxial cables from OM4000 instrument to the oscilloscope)
5. Fiber optic PM patch cable connection from Laser 2 to Reference (if needed)
6. Fiber optic Signal input connection

---

**NOTE.** Turn off laser optical outputs before attaching cables.

---

Store all dust covers and coaxial connector caps for future use. Keep dust and coaxial connectors installed on all unused instrument connections.

Power on the equipment and start applications:

1. Controlling PC
2. Oscilloscope
3. Scope Service Utility (SSU) application after the oscilloscope completes its power-on cycle
4. OM4000 instrument

When powered on, the OM4000 front-panel power button will light briefly after main power is applied, indicating it is searching for a DHCP server, and then turn off. Press the power button one time to enable the unit. The steady power button light indicates the instrument is ready for use and that lasers can be activated using the appropriate controller software.

The power light turns off and the unit is disabled any time AC power is removed or the IP address is changed. Press the power button to re-enable. This feature prevents a remote user from activating the lasers when the local user may not be ready.

---

**NOTE.** *Ethernet only allows devices on the same subnet to communicate. You should now have three devices on a localized Ethernet network: computer, oscilloscope, and OM4000 instrument. This little network may be connected to your corporate network or router or you may choose to leave it isolated.*

---

---

**NOTE.** *For setup purposes, to ease communication between the LRCP and the controller PC, be sure the controller PC (such as a laptop) has only one Ethernet connection (either wireless or wired) activated.*

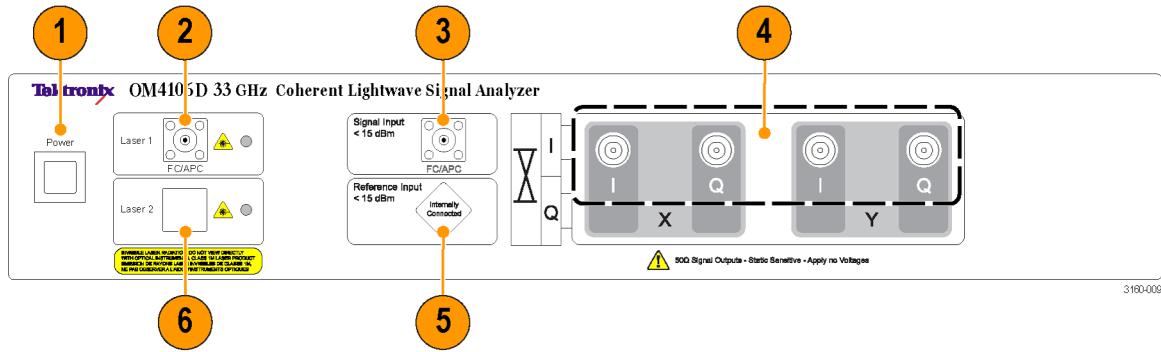
---



# Operating basics

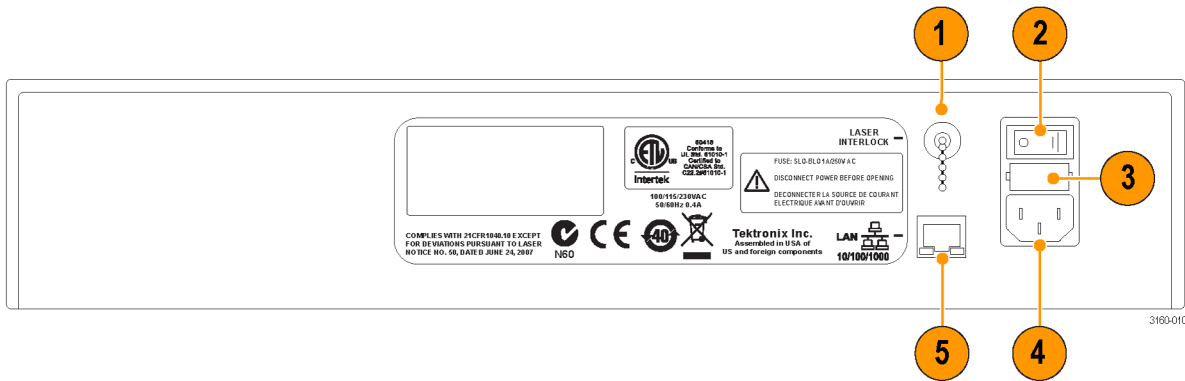
## OM4000 controls and connectors

### Front panel



1. On/Off standby switch
2. Laser 1 output
3. Optical Input (Signal input)
4. X, Y I/Q outputs (RF connectors, to connect to the oscilloscope)
5. Reference Input
6. Laser 2 output (may be internally connected at the factory)

## Rear panel



1. BNC connector for optional laser remote interlock
2. Power switch
3. Fuse holder
4. Power cable connector
5. 10/100/1000 Ethernet port

## Software overview

The OM4000 instrument uses two primary software programs, the OM4000 User Interface (OUI) and the Laser Receiver Control Panel (LRCP).

The OUI:

- Sets up measurement parameters for the OM4000
- Takes input from the OM4000, oscilloscope, and LRCP
- Processes data to display a wide assortment of plots

---

**NOTE.** *The OUI requires the LRCP software to take measurements.*

---

The LRCP:

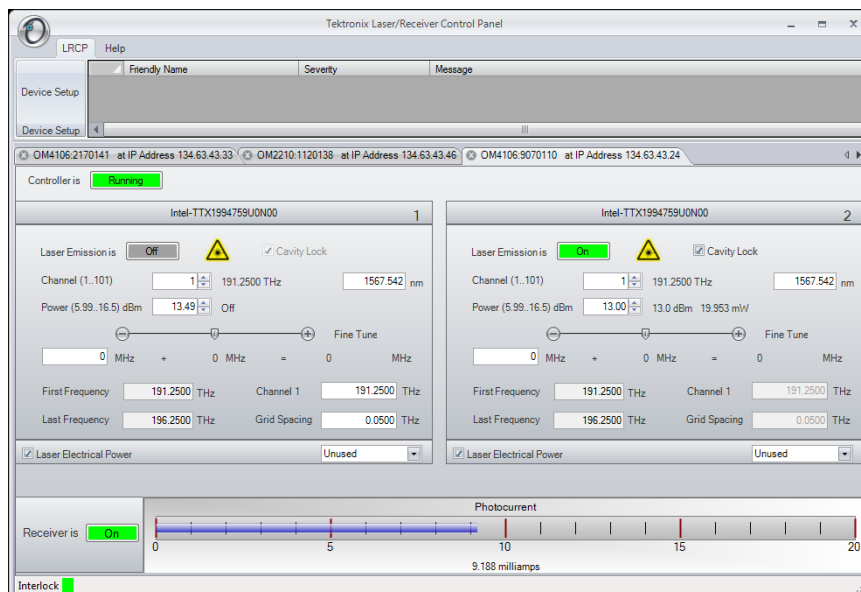
- Detects and provides communication between all detected OM instruments and the OUI.
- Sets the OM instrument default IP address
- Sets OM instrument laser parameters

More information on the LRCP and the OUI are located in the following sections.

The OM4000 also makes use of a third party program, MATLAB by MathWorks, which must be installed on the same PC as the other two applications. The OUI automatically launches the MATLAB application and then interfaces with MATLAB using engine mode. The user does not have to interact with MATLAB for basic operation of the OM4000.

## The Laser Receiver Control Panel (LRCP) user interface

The Laser-Receiver Control Panel application (LRCP) is used to control a variety of Integratable Tunable Laser Assembly (ITLA) lasers. The LRCP interface simplifies the control of the lasers, eliminating the need to use low level ITLA commands. The interface automates locating and configuring all OM devices that are present on the local network. It also provides a Windows Communication Foundation (WCF) service interface, allowing Automated Test Equipment (ATE) to interact directly with the controllers and lasers while LRCP is running.



The main components of the LRCP user interface are:

- **Menu tabs:** Lists available application actions.
- **Controller tabs:** Each tab represents one physical Laser Control device (for example, an OM4000 or an OM2210) on the network. The tab shows the controls for the one or more lasers that are associated with the device.
- **Status bar:** provides important information about the overall state of the communications with the controllers. Each controller has a unique status bar.
- **Receiver gauge:** This gauge displays the total photocurrent output from an instrument. This readout is only functional on devices like the OM4000 instrument that have the appropriate hardware installed.

### Device setup and auto configure

Click the Device Setup button to open the Device Setup dialog box. Use this dialog box on initial setup of the controllers and anytime network configuration changes and devices are moved to a new IP address. Click the Auto Configure button to have LRCP search for and list detected OM devices.

An important setting on the Device Setup screen that users will want to adjust is the Friendly Name. Setting this value for each device will aid in the identification of the physical location of the controllers as Friendly Names are retained and are tied to the corresponding MAC Address. Make sure to exit the form by clicking the OK button to save changes.

The Set IP button is used to modify the addressing as described in the next section. It is not necessary to use the Set IP button to change the Friendly Name.

Each device must be assigned an IP address in order to communicate with the device. How you manage IP addresses in your network, namely with or without DHCP, will determine the method in which you connect to the devices on your network.

### Connecting to your OM instruments

Once configured and detected, devices are listed as tabs on the main LRCP screen. They are listed with the friendly name and IP address to allow for easy identification. Lasers are numbered and once the controller is brought online the laser panels will populate with the laser manufacturer and model number.

Once the user presses the button that reads Offline the button will change colors as the control panel attaches to the OM4000 instrument. First, the button will turn yellow and read “Connecting...” indicating that a physical network connection is being established over a socket. Second, the button will turn teal and read “Connected...”. This indicates that a session is established between the device and Control Panel. Commands will be sent to initialize the communications with the laser and identify their capabilities. Finally, the button will turn bright green when the controller and lasers are ready for action.

---

**NOTE.** *The button color scheme of bright green meaning running or active, grey meaning off line or inactive and red indicating a warning or error state is consistent throughout the application.*

---

Once the controller tab is active and the laser panels have populated with the corresponding laser information, you can change the laser settings and/or turn the lasers on. When the controller is first turned on the current state of the hardware is read and displayed in the laser panel. Any time you exit the application, the current state of the lasers is preserved, including the emission state.

If the lasers are used in conjunction with the OM4000 instrument and OUI, the laser usage type needs to be set using the dialog on the lower right corner of each laser panel. The OUI uses the setting to determine from which laser frequency information is retrieved. A usage type can only be selected once between all of the tabs but you can have one usage type on one tab and another usage type on a second tab.

---

**NOTE.** *Only the Reference laser selection is important to OUI operation. The other selections are to help identify which laser is which.*

---

Once laser emission is On, the channel 1 and grid spacing settings become read-only and cavity lock becomes editable. Also the power goes from “off” to the actual power being read from the laser. Readings are taken from the laser once per second.

The receiver gauge (shown at the bottom of the LRCP window) is only active for equipment that have the appropriate hardware present (such as the OM4000 instruments). The receiver gauge, when active, displays the total photocurrent.

## Setting laser parameters

---

**NOTE.** *For all text field entries it is necessary to click away from the field, or press the Tab key, for the value entered to be accepted by the application.*

---



**CAUTION.** *The LRCP saves all laser parameter settings, including the emission output value, when you exit the application, including the emission state. Make sure to verify laser emission parameters before running tests on a new test setup.*

---

- **Channel:** Type a number or use the up/down arrows to choose a channel. The range of channels available will depend on the type of laser, the First Frequency, and the Grid. The finer the Grid, the more channels are available for a given laser. The channel range is indicated next to the word Channel. The laser channel can also be set by entering a wavelength in the text box to the right of the channel entry. The laser will tune to the nearest grid frequency.
- **Cavity Lock:** The Intel/Emcore ITLA laser that is included in the OM4000 instrument has the ability to toggle its channel lock function. Ordinarily, Cavity Lock should be checked so that the laser is able to tune and lock on to its frequency reference. However, once tuning is complete and the laser has stabilized, this box can be unchecked to turn off the frequency dither needed for locking the laser to its reference. The laser can hold its frequency for days without the benefit of the frequency dither. The OM4000 software will work equally well with the Cavity Lock dither on or off.

- **Power:** Sets the laser power level. Type or use the up/down arrows to choose the desired laser power level. The allowed power range is shown next to the control.
- **Fine Tune:** The Intel/Emcore lasers can be tuned off grid up to 12 GHz. This can be done by typing a number in the text box or by dragging the slider. The sum of the text box and slider values will be sent to the laser. Once the laser has accepted the new value it will be displayed after the '=' sign.
- **First Frequency:** Not settable. This is the lowest frequency that can be reached by the laser.
- **Last Frequency:** Not settable. This is the highest frequency that can be reached by the laser.
- **Channel 1:** Settable when emission is off. This is the definition of Channel 1.
- **Grid Spacing:** Settable (with 100 MHz resolution) when emission is off. 0.1, 0.05 or 0.01THz are typical choices. Use 0.01 THz if tuning to arbitrary (non-ITU-grid) frequencies. Using this grid plus Fine Tune, any frequency in the laser band is accessible.
- **Laser Electrical Power:** This should normally be checked. Unchecking this box turns off electrical power to the laser module. This should only be needed to reset the laser to its power-on state, or to save electrical power if a particular laser is never used.
- **Emission:** Click to turn on or off front panel laser emission.

Channel setting within the ITLA grid gives the corresponding frequency (in THz) and wavelength (in nm). Power is set within the range allowed by the laser. It is best to set the Signal and Reference lasers to within 1 GHz of each other. This is simple if using the internal OM4000 instrument lasers: just type in the same channel number for each.

If using an external transmitter laser, you can type in its wavelength and the controller selects the nearest channel. If this is not close enough, try choosing a finer WDM grid or use the fine tuning feature. If available, fine tuning of the laser is done with the Fine Tune slider bar, and typically works over a range of  $\pm 10$  GHz from the center frequency of the channel selected.

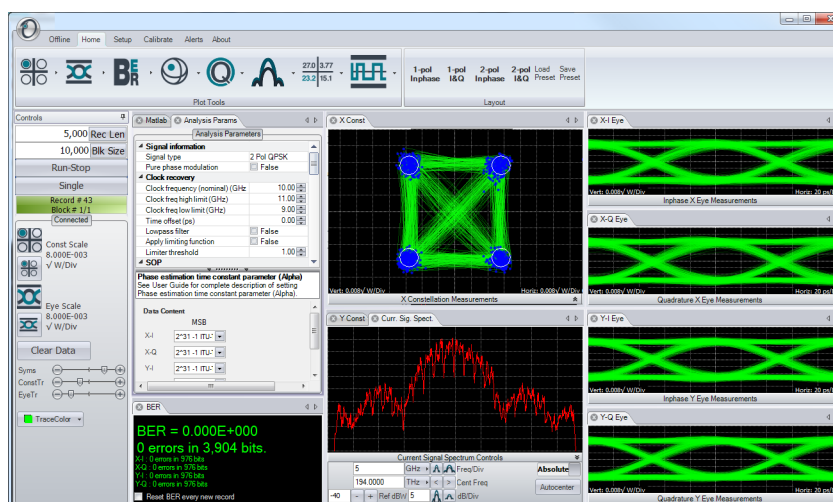
Certain laser models have a cavity lock feature that increases their frequency accuracy at the expense of dithering the frequency; this feature can be toggled with the Cavity Lock button. Cavity Lock is necessary to tune the laser, but can be unchecked to suppress the dither.



Once the channel and power for each laser is set, turn on laser emission for each laser by clicking on its Laser Emission button; the emission status is indicated both by the orange background of the button and by the corresponding green LED on the OM4000 instrument front panel.

## The OM4000 user interface (OUI)

Double-click the OUI desktop icon to open the OM4000 user interface (OUI).

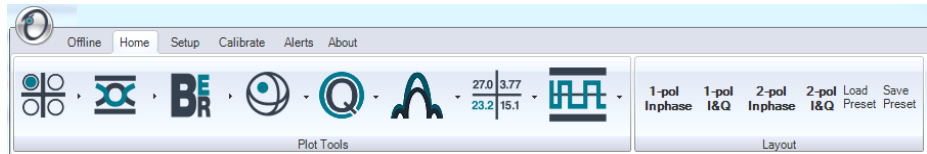


The OUI is a flexible panel-based application. You can move panels within the OUI application or drag and position panels from the OUI onto the Windows Desktop. Clicking and dragging a panel title tab opens a positioning guide. Hold down the left mouse button to position the window onto the positioning guide, then release to organize the plots.


You can rescale Constellation and Eye plots by clicking on the relevant Plot icons in the Controls panel (located by default on the left side of the OUI application). The scale units are  $\sqrt{W}/div$ .

Use the Home tab to set up the plots to display. Click on a Display Format icon in the Plot Tools bar and select a plot type from the menu to display plots relevant to the selected icon. Or select a predefined plot layout from the Layout bar to populate the OUI with parameter, control, and plot panels for the selected measurement.

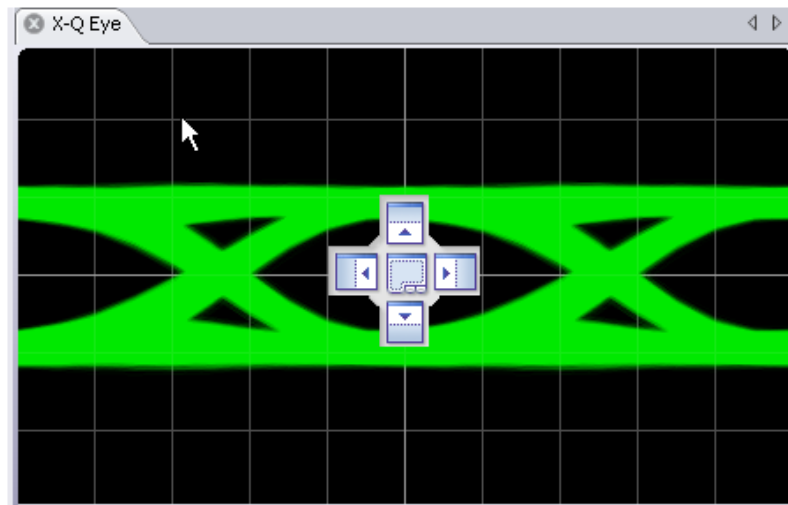
The OUI is designed to allow you maximum control of the graphical presentation. There are three types of displays in the OUI: ribbons, fly-out panels, and windows. The Home ribbon, shown below, normally displayed, provides fast access to key tasks. To get more room for readouts or plots, you can hide the ribbon by double clicking in the tab area. Bring it back by double clicking again on one of the tabs.



Click on an icon to see the available menu items from which to select.

Flyout panels are used for information that is needed less often. Click on the double arrow on a tab  displays or hides the contents of that tab.

The graphics windows can be docked or free floating. To move a graphics window, click and hold over the tab then drag. As you drag the window, different docking targets will appear as shown below. Moving the pointer to the center of the target will cause the window being dragged to be displayed on top of the existing window. Dragging it to one of the four squares surrounding the center of the docking target will split the window so that both the new and old windows are visible. You may also drag the window to another monitor or leave it free floating in front of the OUI main window.





## OUI plots and measurements summary

The following table is an overview of available OUI plots and measurements

Table 8: OUI plots and measurements summary (real-time oscilloscopes)

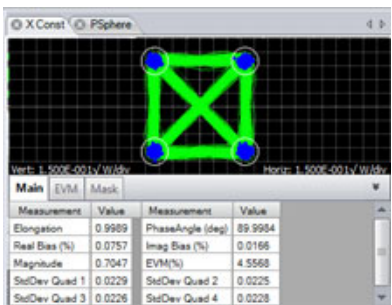
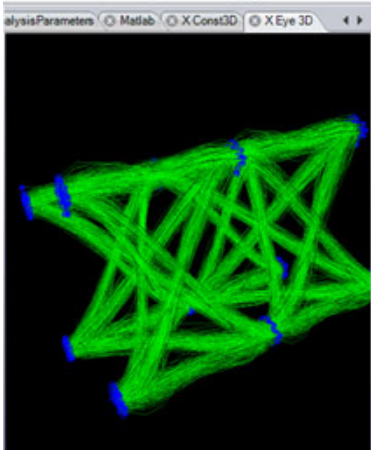
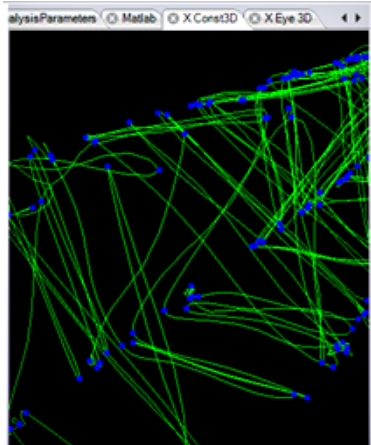
Item	Description																								
 <table border="1" data-bbox="159 604 548 739"> <thead> <tr> <th>Measurement</th> <th>Value</th> <th>Measurement</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Elongation</td> <td>0.9989</td> <td>PhaseAngle (deg)</td> <td>89.9994</td> </tr> <tr> <td>Real Bias (%)</td> <td>0.0757</td> <td>Imag Bias (%)</td> <td>0.0166</td> </tr> <tr> <td>Magnitude</td> <td>0.7047</td> <td>EVM(%)</td> <td>4.5568</td> </tr> <tr> <td>StdDev Quad 1</td> <td>0.0229</td> <td>StdDev Quad 2</td> <td>0.0225</td> </tr> <tr> <td>StdDev Quad 3</td> <td>0.0226</td> <td>StdDev Quad 4</td> <td>0.0228</td> </tr> </tbody> </table>	Measurement	Value	Measurement	Value	Elongation	0.9989	PhaseAngle (deg)	89.9994	Real Bias (%)	0.0757	Imag Bias (%)	0.0166	Magnitude	0.7047	EVM(%)	4.5568	StdDev Quad 1	0.0229	StdDev Quad 2	0.0225	StdDev Quad 3	0.0226	StdDev Quad 4	0.0228	<p>Constellation diagram for X or Y signal polarization with numerical readout bottom tabs. Right-click to see graphics options Symbol-center values are shown in blue Symbol errors are shown in red Right-click for other color options</p>
Measurement	Value	Measurement	Value																						
Elongation	0.9989	PhaseAngle (deg)	89.9994																						
Real Bias (%)	0.0757	Imag Bias (%)	0.0166																						
Magnitude	0.7047	EVM(%)	4.5568																						
StdDev Quad 1	0.0229	StdDev Quad 2	0.0225																						
StdDev Quad 3	0.0226	StdDev Quad 4	0.0228																						
	<p>3d Eye for X or Y signal polarization. This plot can be scaled and rotated to view on a 2d or 3d monitor. It shows the Constellation Diagram with a time axis modulo two bit periods.</p>																								
	<p>3d Constellation for X or Y signal polarization. This plot can be scaled and rotated to view on a 2d or 3d monitor. It shows the Constellation Diagram with a time axis.</p>																								

Table 8: OUI plots and measurements summary (real-time oscilloscopes) (cont.)

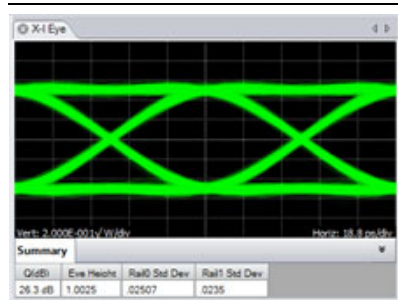
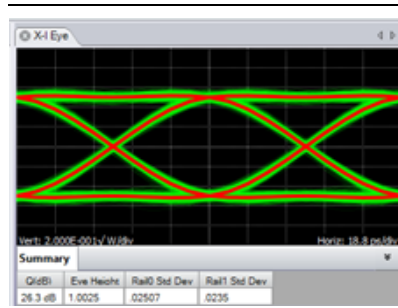
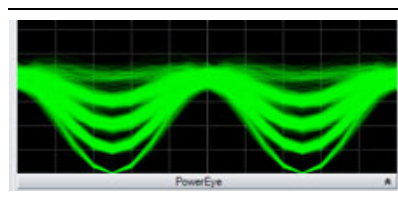
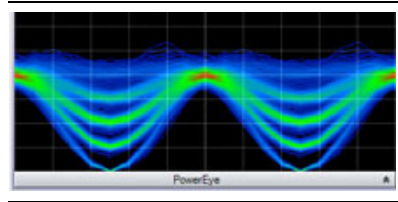
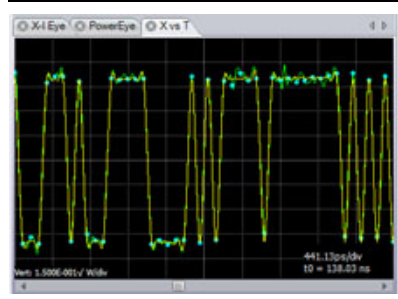

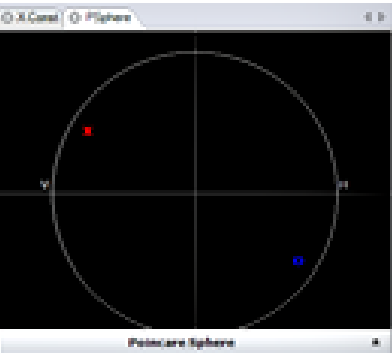
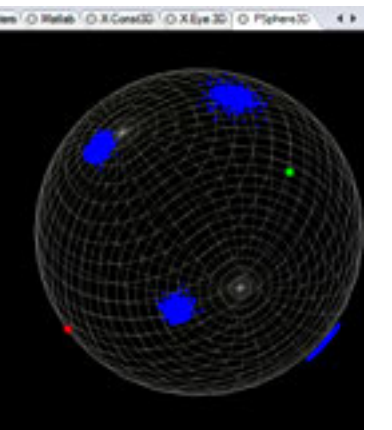
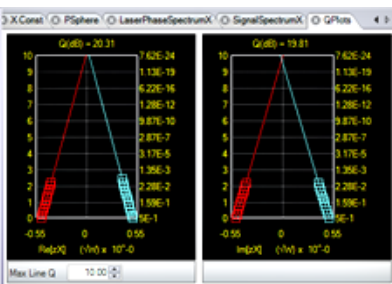
Item	Description
	<p>The coherent eye diagram for X or Y signal polarization shows the In-Phase or Quadrature components vs. time modulo two bit periods. The Q-factor results are provided in a tab below accessed by clicking on the arrows in the lower left corner.</p>
	<p>Right-click on the coherent eye diagram to get options including transition and eye averaging. The transition average shown in red is an average of each logical transition. The calculation is enabled in the Analysis Parameters tab and is used for calculating transition measurements.</p>
	<p>The Power eye shows the computed power per polarization vs time modulo 2 bit periods. This is a calculation of the eye diagram typically obtained with a photodiode-input oscilloscope.</p>
	<p>Most plots can be viewed in colorgrade by right-clicking on the plot.  <b>NOTE.</b> Colorgrade viewing requires an nVidia graphics card installed on the PC or oscilloscope running the OUI software.</p>
	<p>Right-click on the X vs T plot to display field, averaged-field, and symbol quantities. Zoom in or out or scroll through the record. Error symbols are shown in red.</p>

Table 8: OUI plots and measurements summary (real-time oscilloscopes) (cont.)

Item	Description
	<p>BER is shown by physical tributary and in total. Color changes on synch loss.</p>
	<p>2d Poincaré shows the position of the data signal polarizations relative to the receiver's H (1, 2) and V (3, 4).</p>
	<p>3d Poincaré shows polarization of each symbol-center value. Click and drag to rotate the sphere.</p>
	<p>The Decision-Threshold Q-Factor is an ideal signal quality measurement based on measured BER values. The horizontal axis corresponds to the vertical axis on the corresponding coherent eye plot. Linear Q is on the left and BER on the right of the plot. Measured values are indicated by squares: blue for 1's red for 0's.</p>

**Table 8: OUI plots and measurements summary (real-time oscilloscopes) (cont.)**

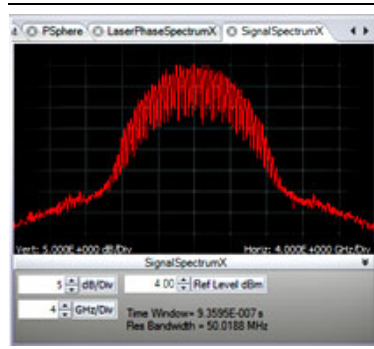
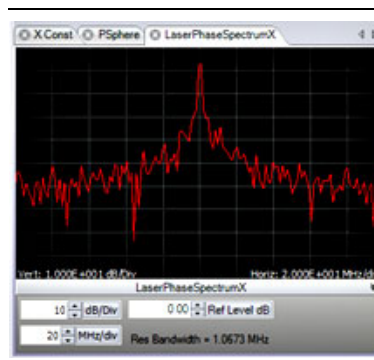
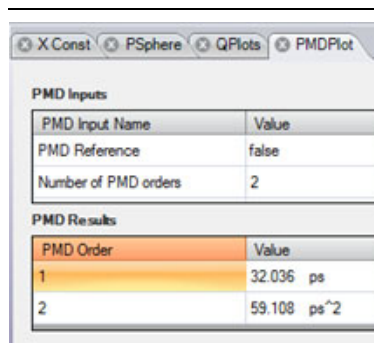

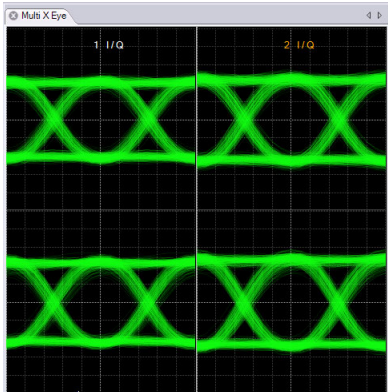
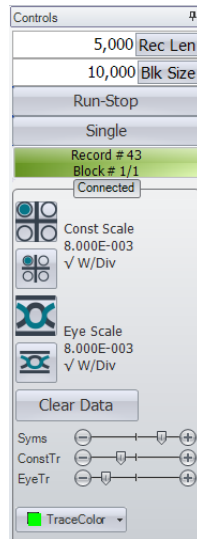
Item	Description																
	<p>The frequency spectrum of the signal field is calculated using an FFT after polarization separation to obtain the spectrum of each signal polarization.</p>																
	<p>The laser phase noise spectrum is obtained by taking an FFT of the <math>e^{i\theta}</math>, where <math>\theta</math> is the recovered laser phase vs. time.</p>																
 <table border="1" data-bbox="129 1092 487 1218"> <thead> <tr> <th colspan="2">PMD Inputs</th> </tr> <tr> <th>PMD Input Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>PMD Reference</td> <td>false</td> </tr> <tr> <td>Number of PMD orders</td> <td>2</td> </tr> </tbody> </table> <table border="1" data-bbox="129 1239 487 1344"> <thead> <tr> <th colspan="2">PMD Results</th> </tr> <tr> <th>PMD Order</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>32.036 ps</td> </tr> <tr> <td>2</td> <td>59.108 ps<sup>2</sup></td> </tr> </tbody> </table>	PMD Inputs		PMD Input Name	Value	PMD Reference	false	Number of PMD orders	2	PMD Results		PMD Order	Value	1	32.036 ps	2	59.108 ps <sup>2</sup>	<p>The PMD plot provides the results of the PMD Calculation</p>
PMD Inputs																	
PMD Input Name	Value																
PMD Reference	false																
Number of PMD orders	2																
PMD Results																	
PMD Order	Value																
1	32.036 ps																
2	59.108 ps <sup>2</sup>																

Table 8: OUI plots and measurements summary (real-time oscilloscopes) (cont.)

Item	Description
	<p>The Measurements Tab provides a convenient place to find almost all of the numerical outputs provided by the OUI with statistics on each value.</p>
	<p>Multicarrier measurements. (See page 66, <i>Multicarrier support (MCS) option</i>.)</p>

**OUI Controls panel**

The **Controls** panel is typically pinned to the left side for easy access to signal acquisition and plot scale controls.



**Table 9: Controls panel elements**

Control	Description
Rec Len	Record Length. Determines the oscilloscope record length for the next acquisition. The record length in turn determines the horizontal time scale given a fixed sampling rate. Since different oscilloscopes allow different record lengths, the OUI replaces your entry with the closest available larger record length for the connected oscilloscope after you click Single or Run-Stop to start the acquisition.
Blk Size	Block Size. Sets the number of points that are processed at one time. For record lengths up to 10,000 or even 50,000 points, it makes sense to process everything at once. This will happen if Blk Size is greater than or equal to Rec Len. However, for record sizes above 50,000, there can be a delay of many seconds waiting for processing. In this case, breaking the processing up into blocks gives you more frequent screen updates. Select Blk Size < Rec Len.  The progress bar shows the task completion status. Block Processing is further explained later in the document  Block Processing is most important when the size of the record is so large that it begins to tax the memory limits of the computer. This can begin to happen at 200,000 points but is more likely a problem at 1,000,000 points and above.

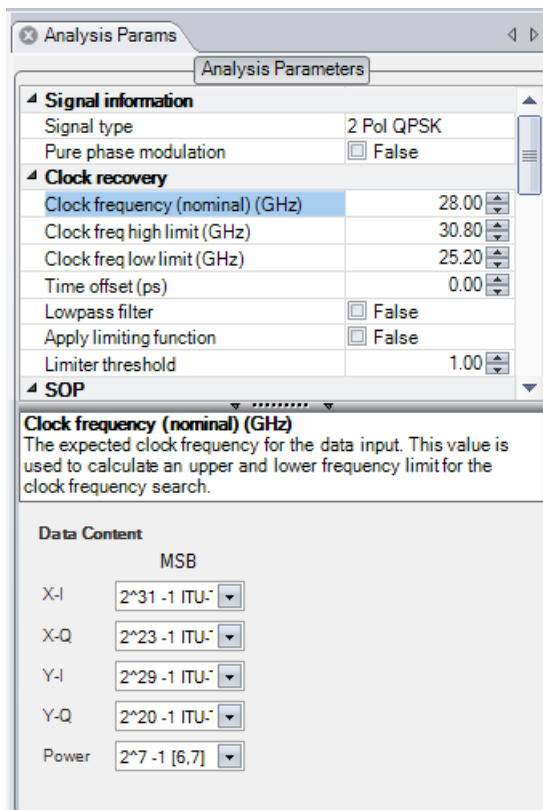
**Table 9: Controls panel elements (cont.)**

Control	Description
	For record sizes between 1,000,000 and the oscilloscope memory limit (usually many tens or hundreds of megapoints), it is essential to break processing into blocks to avoid running out of processor memory. In addition, since neither the entire waveform, nor the entire processed variables will fit in computer memory at one time, it is necessary to make some decisions as to what information will be retained as each block is processed. By default, raw data, electric field values, and other time series data are not aggregated over all blocks in a record greater than 1,000,000 samples. (See page 53, <i>Managing data sets with record length &gt; 1,000,000.</i> )
Run-Stop	The Run button repeatedly takes Single acquisitions until stopped.
Single	The Single button takes a single waveform acquisition for processing and data display.
Scale controls	Provides convenient access to change the plot scale setting. Click on the icons to increment or decrement the setting.
Other controls	Other controls will display depending on the selected plots or measurements.

**Table 10: Record length and block size interaction behavior**

Record length	Block size	Behavior
<1,000,000	≥Rec Len	All data processed in one block. Aggregated variables such as constellation and eye diagrams available for plotting.
<1,000,000	<Rec Len	Data broken up into blocks for processing. Aggregated variables such as constellation and eye diagrams available for plotting after each block has completed.
>1,000,000	<1,000,000	Data broken up into blocks for processing. Only BER and other summary measurements are aggregated block to block. Raw data and time series variables are erased when next block is processed. Need to save workspace of intermediate blocks of interest for later viewing if this data is needed. Run/Stop mode is disabled.
Any	= 1,000,000	The maximum allowable entry in the blk size field is 1,000,000.

**Analysis Parameters window**



Use the Analysis Parameters window to set parameters relevant to the system and its measurements. Click on a parameter to show help on that item in the area at the bottom of the parameter table.



The following controls are relevant to both equivalent-time and real-time oscilloscopes except where noted.

**Table 11: OUI: Analysis Parameters window**

Parameter	Description
Signal type	Sets the type of signal to be analyzed and the algorithms to be applied corresponding to that type.
Pure phase modulation	Sets the clock recovery for when there is no amplitude modulation.
Clock frequency (Nominal, Low, High)	The nominal frequency of the clock recovered by CoreProcessing bounded by a low (Low) and a high frequency (High), provided the clock signal power is sufficient.
Time offset	Applies an offset in time (horizontal movement on the eye diagram) to the signal. If a signal has structure, such as ringing, then the clock recovery process may give a result away from the symbol center. The Time offset adjustment can move it back.
Lowpass filter	If the edges of a signal are steep or if there is some ringing then the clock recovery process may give an eye diagram displaced from the symbol center. Switching on the lowpass filter may cause the eye diagram to become centered. The lowpass filter is applied in the clock recovery path; it does not affect the signals seen in the OUI plots.
Apply limiting function	Some signal distortions may cause the clock component of the signal to be weak, so that the eye diagram is shifted in time or the clock recovery fails (wrong frequency reported). Switching in the limiting function may correct these issues. The limiting function is applied in the clock recovery path, and does not affect the signals seen in the OUI.
Limiter threshold	Threshold level for clock recovery limiting function. A low threshold applies steeper limiting, but may require a longer record length.
Assume Orthogonal Polarizations	Checking this box forces Core Processing to assume that the polarization multiplexing is done in such a way that the two data signals have perfectly orthogonal polarization. Making this assumption speeds processing since only one polarization must be found while the other is assumed to be orthogonal. In this case, the resulting SOPs will be a best effort fit if the signals are not in fact perfectly orthogonal. Unchecking the box forces the code to search for the SOP of both data signals.
Reset SOP Each Block (RT oscilloscopes only)	Checking this causes the SOP to be recalculated for each Block of the computation. By adjusting the Block Size (see Blk Size) you can track a changing polarization. When false, the SOP is assumed constant for the entire Record (see Rec Len).

**Table 11: OUI: Analysis Parameters window (cont.)**

Parameter	Description
2nd Phase Estimate	<p>Checking this box forces Core Processing to do a second estimate of the laser phase after the data is recovered. This second estimate can catch cycle slips, that is, an error in phase recovery that results in the entire constellation rotating by a multiple of 90 degrees. Once the desired data pattern is synchronized with the incoming data stream, these slips can be removed using the known data sequence.</p>
Homodyne (RT oscilloscopes only)	<p>The first step in phase estimation is to remove the residual IF frequency that is the difference between the LO and Signal laser frequencies. The function EstimatePhase will fail if there is no difference frequency. This case occurs when the Signal laser is split to drive both the modulator and the Reference Input of the receiver (ie. only one laser). Checking the Homodyne box will prevent EstimatePhase from failing by adding an artificial frequency shift, which is removed by EstimatePhase.</p>
Phase estimation time constant parameter (Alpha)	<p>After removing the optical modulation from the measured optical field information, what remains is the instantaneous laser phase fluctuations plus additive noise. Filtering the sample values improves the accuracy of the laser phase estimation by averaging the additive noise.</p> <p>The optimum digital filter has been shown to be of the form <math>1/(1+\alpha z^{-1})</math></p> <p>where <math>\alpha</math> is related to the time constant, <math>\tau</math>, of the filter by the relation</p> $\tau = -T/\ln(\alpha)$ <p>where T is the time between symbols.</p> <p>So, an <math>\alpha = 0.8</math> when the baud rate is 10 Gbaud gives a time constant, <math>\tau = 450</math> ps, or a low-pass filter bandwidth of 350 MHz.</p> <p>The value of <math>\alpha</math> also gives an indication of how many samples are needed to provide a good implementation of the filter since the filter delay is approximately equal to the time constant. Continuing with the above example, approximately 5 samples (<math>\sim\tau/T</math>) are needed for the filter delay. This of course is not a problem, but an <math>\alpha=0.999</math> would require 1000 samples and put a practical lower limit on the record length and block size chosen for the acquisition. As a simple rule, the record or block size should be <math>\geq 10/(1-\alpha)</math>.</p>

Table 11: OUI: Analysis Parameters window (cont.)

Parameter	Description
	<p>The selection of the optimum value of Alpha is discussed later in the CoreProcessing guide. (See page 99, <i>EstimatePhase</i>.) This optimum value depends on the laser linewidth and level of additive noise moving from a value near 1 when the additive noise is vastly greater than the phase noise to a value near zero when phase noise is the only consideration (no filtering needed). In practice, a value of 0.8 is fine for most lasers.</p> <p>If Alpha is too small for a given laser there will be insufficient filtering which is evidenced by an elliptical constellation group with its long axis pointed toward the origin (along the symbol vector). When Alpha is too large then there is excessive filtering for the given laser linewidth. Excessive phase filtering is evidenced by the constellation group stretching out perpendicular to the symbol vector and may also lead to non-ideal rotation of the entire constellation.</p> <p>As is often the case, when laser frequency wander is greater than the linewidth, very long record lengths will lead to larger variance in laser phase. This means that an Alpha that worked well with 5000 sample points might not work well with 500,000 points. Longer record lengths will not be a problem if you choose a block size small enough that peak-to-peak frequency wander is on the order of the laser linewidth.</p> <p>For the lasers supplied with the OM4000 instruments, a block size of 50,000 points is a good choice. Refer to Block Processing for more information. (See page 30, <i>OUI Controls panel</i>.)</p>
Signal center freq	Sets the approximate center frequency of the signal. If the signal optical frequency is significantly different from the local oscillator frequency, then this control tells Core Processing where it is. If entered incorrectly then frequency aliasing occurs, and the constellation appears to rotate from one symbol to the next.
Balanced Differential Detection (BDD) (RT oscilloscopes only)	Sets the differential-detection emulator to emulate balanced instead of single-ended detection.
Continuous Traces	Enables drawing fine trace lines that connect the constellation points. If unchecked, the traces will be suppressed for calculation speed if the calculations are not needed for other plots such as eye diagrams.
Mask Threshold	Sets the ratio of radius to symbol spacing used for the circular constellation masks.
Symbol Center Width (ET oscilloscopes only)	Sets the fraction of the eye-center that should be considered "symbol center" for the purpose of certain calculations such as which symbols to color blue. The sample closest to symbol center is used to represent that symbol for calculations such as BER and Q-factor.

**Table 11: OUI: Analysis Parameters window (cont.)**

<b>Parameter</b>	<b>Description</b>
Apply Gray coding for QAM	If checked then the bit error rate reported with a QAM signal is the BER after applying Gray decoding. The Gray coded BER is typically less than the base BER.
Continuous trace points per symbol	Sets the number of samples per symbol for the clock retiming that is done to create the fine traces in the phase and eye diagrams.
Tributaries contributing to average	Sets which possible crosstalk contributions are included in the calculation of impulse response. The average waveforms are based on finding the symbol impulse response and convolving with the data pattern.
Number of symbols in impulse response	Sets the number of values calculated for the impulse response. More values should provide a more accurate average but takes longer to calculate.
Calculate linear average eye	Controls computation of the average eye. Refresh rate is faster when disabled, but must be enabled for the linear average to be displayed in an eye diagram.
Calculate linear average vs. time	Controls computation of the average signal vs. time. Refresh rate is faster when disabled, but must be enabled for the linear average to be displayed in the X vs. T diagram.
Calculate subsequence average	Controls computation of the subsequence averaging. Refresh rate is faster when disabled, but must be enabled for the subsequence average to be displayed in the spectrum plots.
Subsequence average length	Sets the number of symbols in each subsequence.
Calculate transition average	Controls computation of the transition average. Refresh rate is faster when disabled. However, this must be checked to enable calculations based on transition average such as risetime.
Filter type	Sets the type of front end filter, out of Bessel, Butterworth, square root raised cosine, raised cosine, user-defined filter, matched filter or Nyquist filter.
Filter order	Sets the order of the Bessel and Butterworth filter types.
Filter roll-off factor	Sets the roll-off factor of the square root raised cosine and raised cosine filter types.
Cutoff frequency	Sets the cutoff point of the filter. The cutoff frequency refers to the lowpass filter cutoff point. It is equal to half the width of the filter as an optical (bandpass) filter. The cutoff frequency is the 3 dB point of a Bessel, Butterworth or square root raised cosine filter, and the 6 dB point of a raised cosine filter.
Auto-center filter on signal	Sets whether to exactly center the filter on the signal, or to apply it at the nominal signal center frequency.

Table 11: OUI: Analysis Parameters window (cont.)

Parameter	Description
Chromatic Dispersion	The value of Dpsnm used by the Compensate CD function in ps/nm. The sign of Dpsnm should be the same as that of the dispersion compensating fiber that it replaces. In other words, Compensate CD is a dispersion compensator with dispersion of Dpsnm.
Compensate CD	Applies a mathematical model to remove Chromatic Dispersion (CD). The mathematical model used for the filter is: $H(\omega) = e^{i\omega\beta_2/2},$ where $\beta_2 = D_{psnm} \times \frac{10^{-12}}{10^{-9}} \frac{\lambda_0^2}{2\pi c}$
PMD	Polarization Mode Dispersion (PMD) measurement. (See page 52, <i>PMD measurement</i> .)
Data Content	For error counting, constellation orientation, and two-stage phase estimation, the data pattern of each tributary must be specified. Omitting the data specification or providing incorrect information about your data pattern will not stop the constellation or eye displays except that there will be no consistent identification of each tributary since the identification of I and Q and X and Y is arbitrary in the case where the data is not known. Identify your data patterns for each tributary by choosing a standard PRBS from the drop-down menu, or by assigning the pattern variable directly. Select a user pattern from the drop-down menu before assigning the variable directly.

**Front end filtering.** The signal may be filtered according to the settings of the Front end filter group. The filter is a bandpass filter in the optical domain, which is equivalent to a lowpass filter acting on the electrical input signals to the oscilloscope (assuming that the center frequency is zero). The cutoff frequencies specified are those corresponding to a lowpass filter. The width of the bandpass (optical domain) filter is twice the specified lowpass cutoff frequency.

When Auto-center is checked, the filter is tuned to the exact center frequency of the signal. Otherwise the filter is centered at the frequency specified in the Phase group under Analysis Parameters.

There are several filter types available, which fall into three filter categories:

- Fixed filters: Bessel (also known as Bessel-Thomson), Butterworth, square root raised cosine, raised cosine
- User-specified filter
- Adaptive filters: Matched filter, Nyquist filter

The fixed filter types have their cutoff frequency (either 3 dB point for Bessel, Butterworth and square root raised cosine; or 6 dB point for raised cosine) specified by the relevant control. The steepness of the filter is set by the order in the case of Bessel and Butterworth, and by the roll-off factor in the case of the square root raised cosine and raised cosine filters.

When the User-specified filter is selected as the filter type, core processing applies an FIR filter defined in a variable `UserFilter`. If the variable does not exist, or if it is not valid, then core processing continues without applying a filter, and an Alert is issued in the Alerts window to that effect.

`UserFilter` should have three fields: `.t0`, `.dt` and `.Values`. The `.Values` field should be a row vector of complex numbers, corresponding to the FIR coefficients. The time grid (specified by `UserFilter.dt`) does not have to be the same as the oscilloscope sample time interval, or be synchronous with the symbol rate. Core processing resamples the `UserFilter` time grid to the input signal time grid before it is applied. Core processing also tunes the `UserFilter` in frequency to the center frequency specified in the Phase group of Analysis Parameters, and tunes it to the exact center frequency of the signal if Auto-center is checked. Therefore, the FIR coefficients in `UserFilter` should be defined so that it is centered at zero frequency.

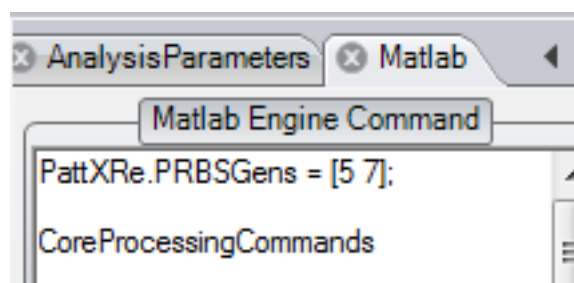
The matched and Nyquist filter types are not fixed, but are defined based on the signal. The matched filter type, as its name implies, is the matched filter having FIR coefficients equal to the time inversion of the signal's impulse response. The matched filter is the best possible filter in terms of the height of an isolated pulse compared to the noise standard deviation. The matched filter may suffer from intersymbol interference (ISI). In general, a Nyquist filter is a filter chosen for a specific signal to have the property that there is no intersymbol interference.

When the Nyquist filter type option is selected a filter is inserted such that the combination of the signal's impulse response with the filter's impulse response is a Nyquist function, having zero ISI. In principle, there are many possible Nyquist functions. The Nyquist function that has been implemented is a raised cosine function, and the steepness (roll-off factor) of the raised cosine is matched to the steepness of the signal spectrum. With the Nyquist filter type, the ISI seen in the eye diagrams should be minimal, but the filter may not suppress noise as well as the matched filter.

The matched and Nyquist filters are available in the MATLAB workspace in variables `FIR` (actual filter) and `FIRCent` (centered version). The filter can be used later as a user-specified filter by assigning `FIRCent` to `UserFilter`. For example, the Nyquist filter may be calculated accurately using a long record, and then recalled later to be applied to short records.

### Direct assignment of pattern variables.

When the transmitter is sending a PRBS pattern that is not one of the standard patterns provided in the drop-down list, you may assign the PRBS polynomial directly in the MATLAB Engine Command Window in the OUI. The acceptable PRBS polynomials are of the form  $X^A + X^B + 1$ , where  $A > B$ . For example, in the polynomial  $X^7 + X^5 + 1$ ,  $A = 7$  and  $B = 5$ . This can be assigned to the Real tributary of the X-polarization as `PattXRe.PRBSGens = [5 7]`; shown in the following figure. Select any standard PRBS in the Analysis Parameters tab. That value will be overridden by the statement in the Matlab Engine Window.



**Direct assignment of pattern variables when not using a PRBS.** When the transmitter is sending something other than a PRBS, even if it is just a DQPSK pre-code, the analyzer needs to know what data is being sent in order to calculate the BER. In this case, it is necessary to load your pattern into MATLAB and assign it to the pattern variable. You must also select User Pattern for the data content in the Analysis Parameters tab.

```
PattXRe.Values = Seq1;
PattXRe.SyncFrameEnd = 100;
PattXIm.Values = Seq2;
PattXIm.SyncFrameEnd = 100;
PattYRe.Values = Seq3;
PattYRe.SyncFrameEnd = 100;
PattYIm.Values = Seq4;
PattYIm.SyncFrameEnd = 100;
```

The code assigns the user's pattern variables Seq1, Seq2, Seq3, and Seq4 to the four tributaries. These variables must be loaded into the separate MATLAB Command Window as shown in the following figure.

```

MATLAB Command Window
>> load('values.mat', 'PattXRe')
>> Seq = PattXRe.Values;
>> size(Seq)

ans =
         1    32767

>> Seq(1,1:10)

ans =
         1         1         1         1         0         0         0         1         1
    
```

In the case shown, a previously saved .mat file is loaded and the Seq variable is created using the PattXRe.Values content. The figure also shows the resulting size of the Seq variable as well as the first 10 values. The pattern for each tributary may have any length, but must be a row vector containing logical values.

Synchronizing a long pattern can take a long time. The easiest way to keep calculations fast when using non-PRBS patterns longer than  $2^{15}$ , and if using record lengths long enough to capture at least as many bits as in the pattern, is to simply use the .SyncFrameEnd field as shown above. Otherwise contact customer support for help in optimizing the synchronization.

**Example capturing unknown pattern.** Another way to load the pattern variable when using a pattern that is not one of the PRBS selections is to use the OUI to capture the pattern and store it in a variable. Do the following:

1. Connect the optical signal with the desired modulation pattern to the OUI.
2. Set up the Analysis Parameters properly with the exception of the data pattern which is not yet known.
3. Choose **Unknown** as the data pattern (do not choose “User Pattern” yet). Optimize the signal to achieve open eye-diagrams and low EVM so that no errors are expected.
4. Set the record length long enough to capture the entire data pattern. For example, you need 32,767 bits to capture a  $2^{15-1}$  pattern. So if this is at 28 Gbaud and the scope has a sampling rate of 50 Gs/s, then you need at least  $32,767 * 50 / 28 = 58,513$  points in the record. Stop acquisition after successfully displaying a good constellation with sufficient record length. All the data you need is now in the MATLAB workspace. It just needs to be put in the proper format for use in the pattern variable.
5. In the separate MATLAB Command Window, add the following commands:

For QPSK:

```
PattXReM = real(zXSymUI.Values) > 0;
```



```
PattXImM = imag(zXSymUI.Values) > 0;
```

For dual-pol QPSK add these commands: (in addition to above)

```
PattYReM = real(zYSymUI.Values) > 0;
```

```
PattYImM = imag(zYSymUI.Values) > 0;
```

6. To get a single full pattern, delete the extra data as follows (in this case for 32,767 bits):

For QPSK:

```
PattXReM = PattXReM(1: 32767);
```

```
PattXImM = PattXImM(1: 32767);
```

For dual-pol QPSK add these commands: (in addition to above)

```
PattYReM = PattYReM(1: 32767);
```

```
PattYImM = PattYImM(1: 32767);
```

7. In the CLSA MATLAB Engine Command Window, add the following lines before the CoreProcessing statement:

For QPSK:

```
PattXRe.Values = PattXReM;
```

```
PattXIm.Values = PattXImM;
```

For dual-pol QPSK add these commands: (in addition to above)

```
PattYRe.Values = PattYReM;
```

```
PattYIm.Values = PattYImM;
```

8. Select User Pattern for any of the tributaries where you assigned a user pattern in the above steps. You should now be able to measure BER using your new patterns.
9. To save the patterns for later use, type the following in the separate MATLAB Command Window:

```
save('mypatterns.mat', 'PattXReM', 'PattXImM', 'PattYReM',  
'PattYImM')
```

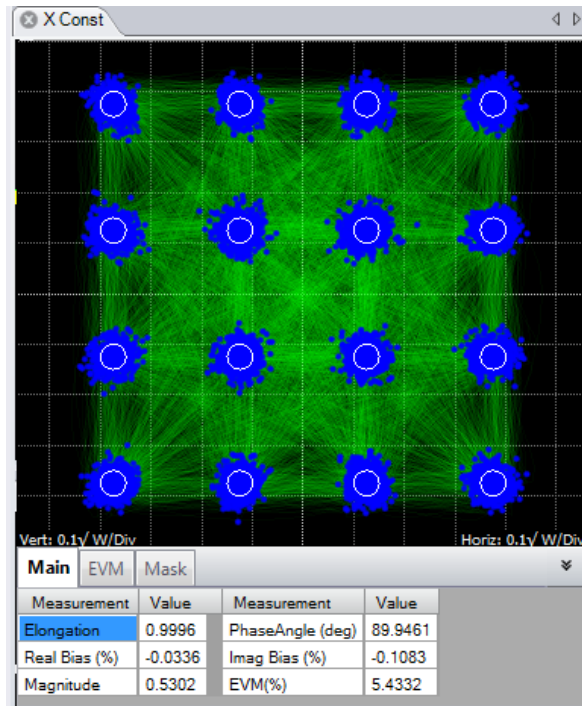
## Constellation diagrams

Many types of constellation diagrams can be chosen by clicking on the constellation icon. Once the laser phase and frequency fluctuations are removed, the resulting electric field can be plotted in the complex plane.

When only the values at the symbol centers are plotted, this is called a Constellation Diagram. When continuous traces are also shown in the complex plane, this is often called by the more generic term of IQ Diagram. Since the continuous traces can be turned on or off in the OUI, we refer to both as the Constellation Diagram.

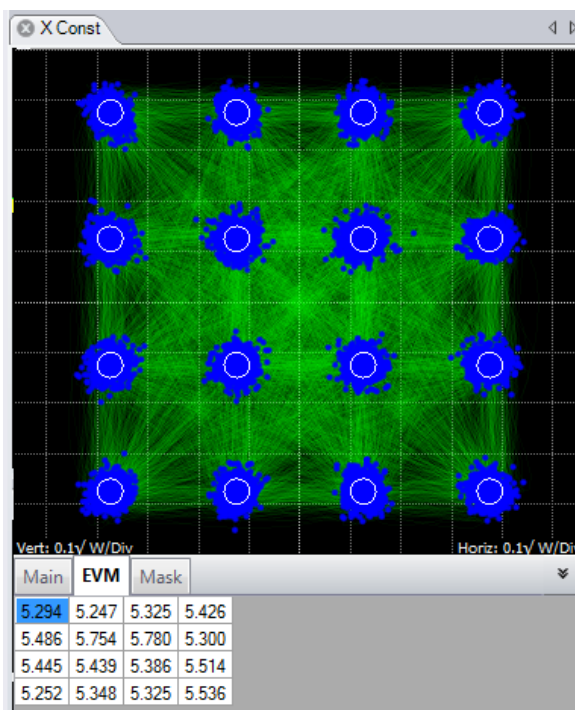
The scatter of the symbol points indicates how close the modulation is to ideal. The symbol points spread out due to additive noise, transmitter eye closure, or fiber impairments. The scatter can be measured by symbol standard deviation, error vector magnitude, or mask violations.

**Constellation measurements.** Measurements made on constellation diagrams are the most comprehensive in the OUI. Numerical measurements are available on the flyout panel associated with each graphic window. The measurements available for constellations are described in the following text.



- Elongation: The mean inter-symbol spacing of the quadrature signals divided by the mean inter-symbol spacing of the in-phase signals. “Tall” constellations have Elongation > 1.
- Real Bias: The real part of the mean value of all symbols divided by the magnitude; expressed as a percent. A positive value means the constellation is shifted right.
- Imag Bias: The imaginary part of the mean value of all symbols divided by the magnitude; expressed as a percent. A positive value means the constellation is shifted up.
- Magnitude: The mean value of the magnitude of all symbols with units given on the plot.
- Phase Angle: The phase angle between the two tributaries.

- **StdDev by Quadrant:** The standard deviation of symbol point distance from the mean symbol in units given on the plot. This is displayed for BPSK and QPSK.

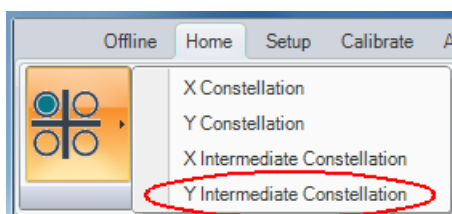


- **EVM (%):** The rms distance of each symbol point from the ideal symbol point divided by the magnitude of the ideal symbol expressed as a percent.
- **EVM Tab:** The separate EVM tab shown in the right figure provides the EVM% by constellation group. The numbers are arranged to correspond to the symbol arrangement.
- **Mask Tab:** The separate Mask tab shown in the right figure provides the number of Mask violations by constellation group. The numbers are arranged to correspond to the symbol arrangement.

The Q calculation can cause alerts if it can't calculate a Q factor for the outer transitions. For example, in 32-QAM. 32-QAM is a subset of 64-QAM, where the outer constellation points are never used. It is not possible to calculate a Q factor for those outer slices, hence the alert. The subconstellation identification feature notices the unused constellation points, and removes them from the relevant constellation parameters (zXSym.Mean, zXSym.ConstPtMean, etc.), but that happens in EngineCommandBlock, after the Q calculation has occurred. QDecTh does not know that the outer constellation points never occur, and so it generates the appropriate alert, but it does continue processing. (See page 104, *QDecTh*.)

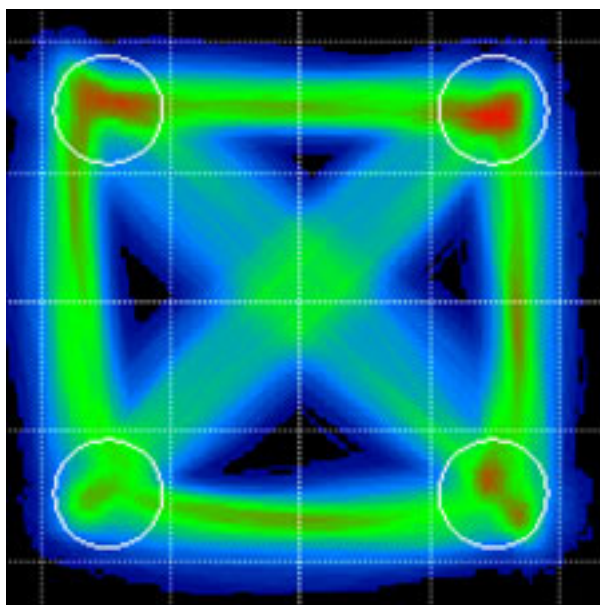
**Offset modulation formats.** Both polarization and quadrature offset formats are available. To properly display polarization offset formats, select **Home** >

**Constellation button > Y Intermediate Constellation** as the display, as shown in the following figure.



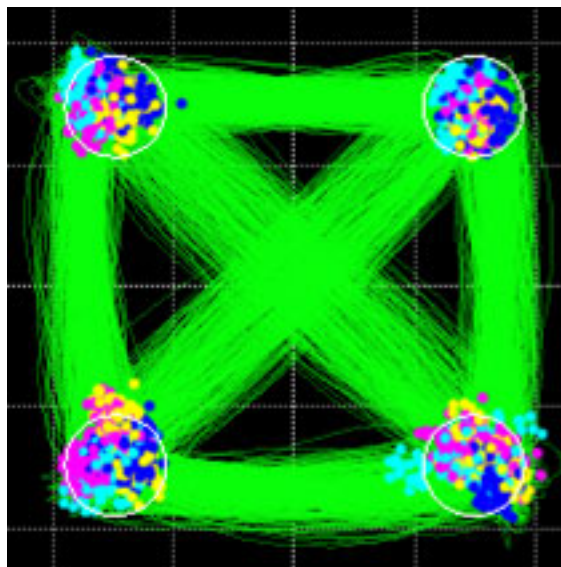
Note that the Y polarization is a half-symbol offset from the X polarization; the standard “Y const” display will be empty, and the offset (or intermediate) constellation display is selectable from the constellation pull-down menu as shown in the upper left-hand corner.

**Color features.** A new feature (dependent on your PC video adapter capabilities), beginning with Version 1.2.0, is the ability to Right-Click on any constellation window and get a list options including Color Grade, Display Traces in Color Grade, and Color Key Constellation Points.



**Figure 3: Color grade constellation- fine traces**

The Color Grade option provides an infinite persistence plot where the frequency of occurrence of a point on the plot is indicated by its color. This mode helps reveal patterns not readily apparent in monochrome. Persistence can be cleared or set from the Right-Click menu as well.



**Figure 4: Color Key constellation**

Color Key Constellation Points is a special feature that works when not in Color Grade mode. The value of the previous symbol determines the symbol color. This helps reveal pattern dependence.

The Color Key colors:

- If the prior symbol was in Quadrant 1 (upper right) then the current symbol is colored Yellow
- If the prior symbol was in Quadrant 2 (upper left) then the current symbol is colored Magenta
- If the prior symbol was in Quadrant 3 (lower left) then the current symbol is colored light blue (Cyan)
- If the prior symbol was in Quadrant 4 (lower right) then the current symbol is colored solid Blue

### Eye diagrams

Eye diagram plots can be selected for appropriate modulation formats by clicking on the eye-diagram icon and selecting an eye format. Supported eye formats include field Eye (which is simply the real part of the phase trace in the complex plane), Power Eye (which simulates the Eye displayed with a conventional oscilloscope optical input), and Diff-Eye (which simulates the Eye generated by using a 1-bit delay-line interferometer). As with the Constellation Plot you can right-click to choose color options as well.

The field Eye diagram provides the following measurements:

- Q (dB): Computed from  $20 \cdot \text{Log}_{10}$  of the linear decision threshold Q-factor of the eye
- Eye Height: The distance from the mean one level to the mean zero level (units of plot)
- Rail0 Std Dev: The standard deviation of the 0-Level as determined from the decision threshold Q-factor measurement
- Rail1 Std Dev: The standard deviation of the 1-Level as determined from the decision threshold Q-factor measurement

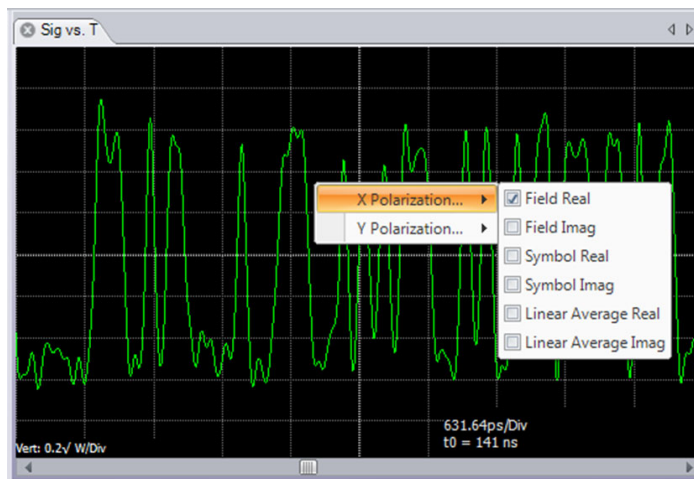
In the case of multi-level signals, the above measurements will be listed in the order of the corresponding eye openings in the plot. The top row values correspond to the top-most eye opening.

The above functions involving Q factor use the decision threshold method described in the paper by Bergano <sup>1</sup>. When the number of bit errors in the measurement interval is small, as is often the case, the Q-factor derived from the bit error rate may not be an accurate measure of the signal quality. However, the decision threshold Q-factor is accurate because it is based on all the signal values, not just those that cross a defined boundary.

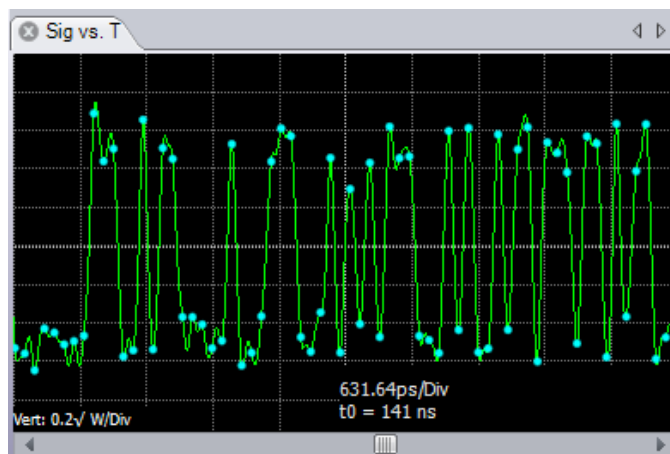
<sup>1</sup> N.S. Bergano, F.W. Kerfoot, C.R. Davidson, "Margin measurements in optical amplifier systems," IEEE Phot. Tech. Lett., 5, no. 3, pp. 304-306 (1993).

## Signal vs. Time

Several plots of field components as a function of time are available by selecting **Signal vs. Time** after clicking the waveform icon under the Home tab of the main ribbon. Sig vs T is different from other plots in that it allows many different variables to be displayed, and the user chooses which variables. The plot displays only the inphase X polarization field component when created. Right clicking the plot opens a context menu with X and Y polarization menus, and hovering the mouse shows a list of available selections.



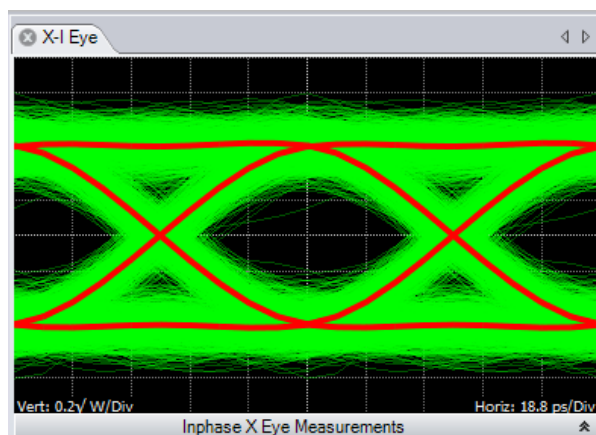
The field options are the as-measured electric field components, plotted as green lines. The symbol options draw blue dots at the symbol center times. The linear average is discussed in a later section, and is plotted as yellow lines. (See page 47, *Waveform averaging*.)



Clicking the mouse scroll wheel zooms the Sig vs T plot in time, and the scroll bar along the bottom shows how much of the record is being displayed. Slide the horizontal scroll bar to offset the plot in time. t0 is the plot center relative to the trigger time. Errored symbols are shown in red.

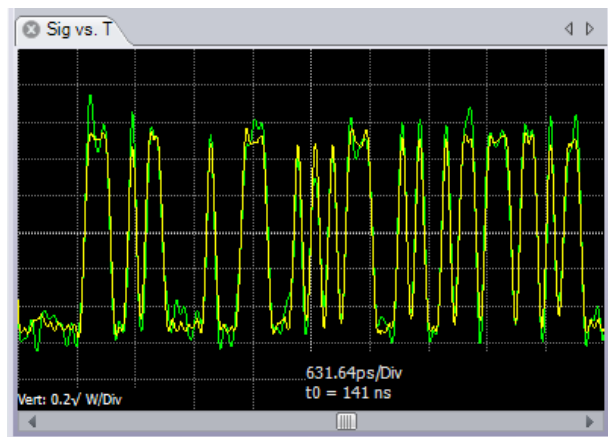
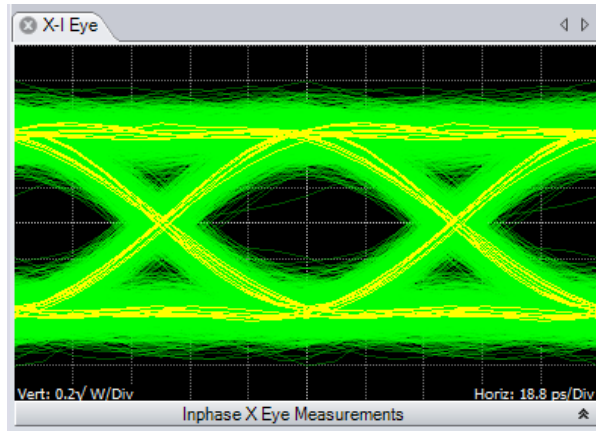
### Waveform averaging

Two types of averaged display of the eye diagram and signal vs. time are available. These show a cleaner version of the signal, having a reduced level of additive noise. The transition average is available by checking Averaging: Show Transition Average under Analysis Parameters and selecting Show Transition Average from the right click menu of the eye diagram where the average is to be displayed. The red trace shows the average of the different transitions between levels: 0-0, 1-1, 0-1 and 1-0.



The transition parameters listed in the X-Trans, Y-Trans and Pow-Trans sections of the Measurements table are derived from the transition average curves. Transition average is available for the field component eye diagrams, and if the modulation format is an OOK type, for the power eye diagram.

The linear average is made visible by checking Averaging: Show Linear Average Eye or Show Linear Average vs. Time. The average is displayed as yellow traces in any field eye diagram or Signal vs. Time plot where the linear average is selected from the right click menu.



The linear average is obtained using a two-step process:

- The impulse response associated with the signal is calculated by a deconvolution process
- That impulse response is applied to the known data content of the signal to produce a linear average.

The linear average assumes that the signal has a linear dependence on the data bits. If there is nonlinearity, for example if the crossing point is higher than 50%, then the linear average is a poor fit to the actual waveform.



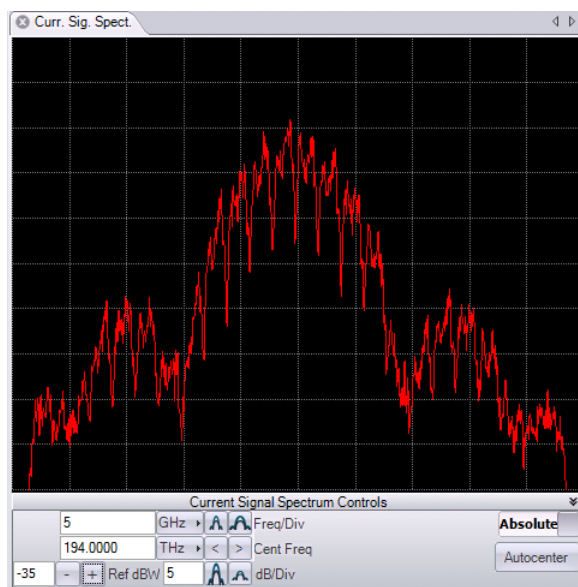
The linear average can provide useful information about the nature of the signal. The length of the impulse response is set by Averaging: Number of Symbols in Impulse Response. Typically the average eye diagram appears noisier as the impulse response length is increased, because the number of traces in the eye diagram increases. However if the true impulse response of the signal has a long duration, for example if there is a reflection from a length of RF cable inside the transmitter, then the linear average eye diagram will clean up once the impulse response length is made long enough to capture that reflection event.

There are several options of which tributaries to take into account when calculating the impulse response, selected from Averaging: Tributaries contributing to average. The basic option is Same trib only, which typically gives the cleanest result. It is possible to include other tributaries and exclude the same tributary, for example Other SOP tribs. This setting computes the impulse response only by taking into account the signal on the other state of polarization. For an ideal signal the linear average computed this way should be a flat line. If there is structure on the linear average waveform then that suggests there is a crosstalk mechanism between the states of polarization.

The impulse response variables are available in the MATLAB workspace in variable Imp, which has fields .XRe, .XIm, .YRe and .YIm.

## Current Signal Spectrum plot

The Current Signal Spectrum plot is accessed by clicking on the spectrum icon button on the Home tab. Right-click on the plot to select what to display. The OUI displays the input signal spectrum by default.



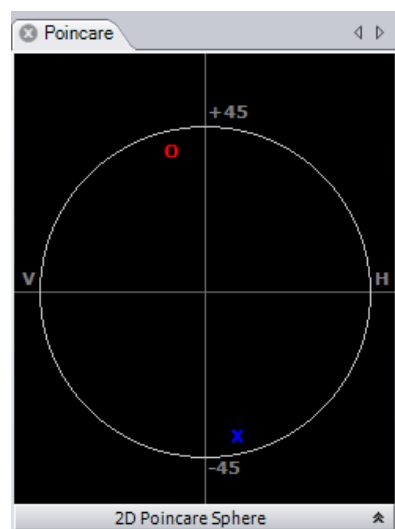
## Measurements Statistics table

The Measurement Statistics table is displayed by the Tabular-Data menu (click on the Tabular Data icon). The Measurements Statistics table contains essentially every measurement made by the OUI with statistics. In cases such as EVM or Q-factor for QAM, where there may be too many numbers to list in the table, the table shows an average for each tributary. The detailed values by constellation group are found on the constellation, eye, or Q plots, and are also available in the MATLAB workspace.

The table shows the following measurements:

- **X-Eye (Y-Eye):** These are the measurements related to the decision-based Q-factor method. Sweeping the decision threshold value while computing the resulting BER, provides a measure of the Eye Height, and standard deviations of each rail.
- **X-Const (Y-Const):** These measurements are made on the constellation groups calculated for the constellation diagram display.
- **X-Trans (Y-Trans):** The transition parameter measurement is based on the Transition Average. (See page 47, *Waveform averaging*.) The values listed are measured on the averaged transition.
- **Pow-Trans:** Is the transition parameters for power signal. These values are only calculated for the power signaling types such as OOK and ODB.
- **XY Measurements:** Sig Freq Offset is the calculated difference by between Signal and Reference Lasers. Signal Baud Rate is the recovered Clock Frequency. PER is the polarization extinction ratio of the transmitter calculated when Assume Orthogonal SOPs is no checked. PDL is the relative size of the X and Y constellations (PDL of a PM modulator).
- **PMD:** (See page 52, *PMD measurement*.)

## 2D Poincaré sphere



The Core Processing software locks on to each polarization signal. Depending on how the signals were multiplexed, the polarizations of the two signals may or may not be orthogonal. The polarization states of the two signals are displayed on a circular plot representing one face of the Poincaré sphere. States on the back side are indicated by coloring the marker blue.

The degree of orthogonality can be visualized by inverting the rear face so that orthogonal signals always appear in the same location with different color. Thus Blue means back side (negative value for that component of the Stokes vector), X means X-tributary, O means Y-tributary, and the Stokes vector is plotted so that left, down, blue are all negative on the sphere.

**InvertedRearFace:** checking this box inverts the rear face of the Poincaré sphere display so that two orthogonal polarizations will always be on top of each other.

CoreProcessing reports pXSt and pYst organized Q, U, V in the terminology shown below. These values are plotted as X,Y pairs (Q,U) with V determining the color (blue negative). The plot is from the perspective of the “North Pole.”

$$I = |E_x|^2 + |E_y|^2$$

$$Q = |E_x|^2 - |E_y|^2$$

$$U = 2 \operatorname{Re}(E_x E_y^*)$$

$$V = 2 \operatorname{Im}(E_x E_y^*)$$

### Bit-Error-Rate reporting

Bit error rates are determined by examination of the data payload. You may choose BER or Differential BER. Differential BER compares the output of a simulated delay-line interferometer to a differential form of the data pattern specified in the Analysis Parameters. If you choose to pre-code your data signal prior to the modulator as in a typical differential transmitter, you will need to enter the patterns seen at the I and Q modulators into the respective pattern variables, (for example, PattXRe.Values and PattXIm.Values). If no pre-coding is used, then you may use the drop-down menus to specify standard PRBS codes. (See page 37, *Front end filtering*.)

For multilevel signal types such as QAM, the Gray code BER or the direct BER may be reported. The check box BER: Apply gray coding for QAM under Analysis Parameters selects which BER type is reported. More information is given in a detailed application note on automated BER measurements at the end of this manual.

## PMD measurement

Polarization mode dispersion (PMD) is an effect associated with propagation through long distances of optical fiber that degrades the signal through inter-symbol interference. It is described by several coefficients. Often the first order and second order coefficients are all that is needed. The OUI can estimate the amount of PMD that a signal has experienced from the structure of the waveform. The method used is described in the research paper by Taylor<sup>1</sup>. The PMD measurement works with dual polarization signals. Two kinds of measurement are possible, reference based and non-reference based, according to the check box in PMD: Use PMD Reference under Analysis Parameters. If the non-reference based measurement is chosen then the OUI estimates the PMD directly from the signal. The first and second order PMD values are reported in the Measurements window.

The reference based measurement is sometimes more accurate than the non-reference based measurement. If the signal itself has an offset in time between the X and Y polarizations then with the non-reference measurement that offset is effectively added to the reported PMD values. With the reference based measurement, that offset between polarizations is taken into account, by first acquiring a measurement (the reference) direct from the transmitter.

It is necessary to tell the OUI when the reference is being acquired, and that is done by checking the PMD: Acquire PMD Reference check box. When the reference measurement is complete this check box must be unchecked. The reference-based measurement uses a linear impulse response, and the settings under Averaging (See page 47, *Waveform averaging.*), also apply to the reference-based PMD measurement. The PMD: Measure PMD check box must be checked for the PMD values to be reported in the Measurements window.

<sup>1</sup> M.G. Taylor & R.M. Sova, "Accurate PMD Measurement by Observation of Data-Bearing Signals," IEEE Photonics Conf. 2012, paper ThS4, Burlingame CA, 2012.

## Recording and playback

You can record the workspace as a sequence of .mat files using the Record button in the Offline ribbon. These are recorded to C:\Users\\Documents\TekApplications\OUI\MAT Files.

You can play back the workspace from a sequence of .mat files by first using the Load button in the Offline Commands section of the Home ribbon. Load a sequence by marking the files you want to load using the Ctrl key and marking the filenames with the mouse. You can also load a contiguous series using the Shift key and marking the first and last filenames in the series with the mouse.

Use the Run button in the Offline Commands section of the Home ribbon to cycle through the .mat files you recorded. OUI uses the filter settings stored in the .mat files.

## Alerts

See the Appendix for alert messages information. (See page 109, *Alerts.*)

## Managing data sets with record length > 1,000,000

It is important to break up records larger than 1,000,000 points into blocks that can fit into the computer memory. This is done by setting the Blk Size to something between 10,000 and 200,000. Typically 50,000 is a good balance between speed of progress updates and overall processing time. When operating in this mode, only the number of errors and other numerical measurements are maintained from block to block. Time series information such as electric field values and raw data are discarded to conserve memory. This means that if you do nothing else, the large acquisition will end with only the field and symbol values for the last block available along with the BER, EVM, and measurements which are collected over the entire record. It is important in the large-acquisition case to learn how to save intermediate data sets if the time-series data is needed.

**Saving intermediate data sets: examples.** The simplest way to save intermediate data is to record every record. (See page 52, *PMD measurement*.) However, this may generate a very large set of files that will then need to be analyzed later. If you only want to save the workspace on a particular event, you can use the save command after the CoreProcessing call.

Once you have saved the data sets, you can view them later by using the Load command described earlier. You can load them one at a time or as a group to see a replay. Just be sure the correct analysis parameters are being used. If you save the entire workspace by omitting the variable names in the save statement, then you can also open the .mat files later in MATLAB and use MATLAB plots to examine the variables.

There are two parts to setting this up. First you need a unique file name that can be created automatically, second you need to design an if-statement to trigger on the proper event.

**Examples of save statements to unique file names.** The following command saves data to files with the name test*n*.mat, where the *n* is replaced with whichever block is being processed at the time.

This is simple but has the drawback that the files will be overwritten by future acquisitions that happen to save on the same test name and block numbers.

```
%
save(['test', num2str(BlockNum), '.mat'], 'vblock')
%
```

The following example saves the entire workspace to time-stamped filenames of the form Test11\_4\_2009\_12\_24\_53.mat., with the first string (Test) followed by parts of the Clk string with the month (2), day (3), year (1), hour (4), minute (5), and second (6) the save was executed.

```
%
clk = clock;
```

```

save(['Test', num2str(Clk(2)), '_ ', num2str(Clk(3)), '_ ',
num2str(Clk(1)), '_ ', ...

num2str(Clk(4)), '_ ', num2str(Clk(5)), '_ ',
num2str(round(Clk(6))), '.mat'])

%

```

**Examples of if-statements and alerts to trigger a save.** The following example (when placed after the CoreProcessing call in the MATLAB window) saves the Vblock variables every time there is a bit error on the XRe tributary. The Vblock variables are really all that are necessary for later analysis, but saving the whole workspace can help be sure that the original processing information, such as patterns and signal type, are not lost.

In the following example, using BER.NumErrs, instead of NumErrs.XRe, has the effect of triggering on any error in any tributary, rather than just XRe.

```

%
if Errs.XRe.NumErrs>0
Clk = clock;

save(['HybridCal', num2str(Clk(2)), '_ ', num2str(Clk(3)), '_ ',
num2str(Clk(1)), '_ ', ...

num2str(Clk(4)), '_ ', num2str(Clk(5)), '_ ', num2str(round(Clk(6))),
'.mat'], vblock)

end

%

```

The following example triggers on an alert, using the Alert variable existence or the type of alert as a trigger

```

%
if(isfield(Alerts, 'Active'))
Clk = clock;

save(['HybridCal', num2str(Clk(2)), '_ ', num2str(Clk(3)), '_ ', num2str(Clk(1)),
...

num2str(Clk(4)), '_ ', num2str(Clk(5)), '_ ', num2str(round(Clk(6))), '.mat'],
vblock)

end

%

```

## Using the OUI with other receivers

The OUI software may be used to process data taken from any receiver that has the following properties:

- The signal is linear. This means that AGC amplifiers (if used) must be put in fixed gain mode.
- Stable over the measurement time. The OUI software uses calibration routines to correct for receiver optical properties such as gain, skew, and phase angle error. If the receiver properties change significantly between the time of calibration and the time of using the calibration data, then you will not get optimal results. It is still possible to assume ideal receiver properties.

To use a receiver other than the OM-Series, follow the same setup and calibration procedure as for the OM4000 instrument with the following exceptions:

1. Disable the automatic hybrid calibration. This can be done by placing the following statement in the Engine Command Window in the OUI prior to the CoreProcessing statement:

```
pHyb = [1 i 0 0; 0 0 1 i];
```

2. Disable the factory frequency response equalization for the OM4000. This can be done by placing the following statement in the Engine Command Window in the OUI prior to the CoreProcessing statement: `EqFilt(1).dt = 1;` This can go before or after the pHyb statement.
3. Follow the other calibration procedures (See page 111, *Calibration and adjustment (RT oscilloscope)*). In the case of Hybrid Calibration (See page 115, *Hybrid calibration (RT)*), put the following statement in the Engine Command Window in the OUI before the DispCalEllipses statement:

```
CorrectPhase = true;
```

The resulting pHyb statement replaces the one in step 1. Hybrid Calibration is optional if the receiver gain and phase accuracy is good enough to be approximated by the ideal hybrid model in step 1.

After connecting and calibrating the receiver, it is ready for use with the OUI software.

To emulate the behavior of a deployed receiver:

- The OM4000 product is an analyzer for understanding the physical performance of transmitters and optical communication links. It uses fixed calibration parameters to remove uncertainties. Deployed receivers will run algorithms to correct in real time for skew, gain, and phase errors of the receiver, being concerned more about data integrity rather than physical measurement accuracy. To emulate this function of the receiver, it is important to follow the calibration steps 1- 3, and to repeat the calibration when the receiver properties change.
- A deployed receiver will also run an adaptive algorithm that optimizes the frequency response of the receiver. To emulate this function, use the adaptive

filter algorithm provided (you may have to ask for this as it is distributed separately.)



# Configuring the OM4000 user interface (OUI)

Start the OUI with the icon on your desktop or in the programs menu.

---

**NOTE.** Be sure that MATLAB is available and properly licensed, since the OUI attempts to launch a matlab command window, and will appear to stall if MATLAB is not available.

---

Connecting to the oscilloscope upon OUI startup is done with the Connect button in the Scope Setup section of the Setup ribbon. Notice that there are two choices for making an oscilloscope connection: VISA and non-VISA. VISA is only necessary when working with older real-time oscilloscopes. (See Table 12.)

**Table 12: Oscilloscope connectivity (VISA vs. Scope Service Utility)**

OUI Capability	VISA	Scope Service Utility (non-VISA )
Segmented readout for unlimited record size	Yes	Yes
Ability to collect data from two networked oscilloscopes running the Scope Service	No	Yes
Software required on oscilloscope	LAN server	Scope Service Utility or ET Scope Service Utility
Real-time oscilloscope compatibility	Any real-time Tektronix oscilloscope supported by the IVI driver	C and D-model 70000 series oscilloscopes with firmware v6.4 or later
Equivalent-time oscilloscope compatibility	No	DSA8300 or 8200 with ET Scope Service Utility

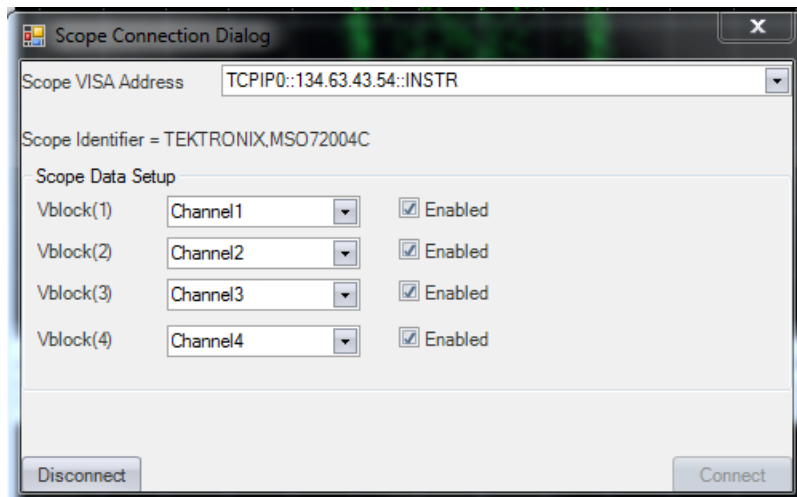
## VISA connections

The VISA address of the oscilloscope contains its IP address, which is retained from the previous session, so it should not normally need to be changed, unless the network or the oscilloscope has changed. The VISA address string should be TCPIP0::IPADDRESS::INSTR where IPADDRESS is replaced by the scope IP address, such as 172.17.200.138 in the example below.

---

**NOTE.** To quickly determine the oscilloscope IP address, open a command window (“DOS box”) on the oscilloscope and enter **ipconfig /all** to display the instrument IP address.

---



After clicking Connect, the drop down boxes will be populated for channel configuration. Choose the oscilloscope channel name which corresponds to each receiver output and MATLAB variable name. These are:

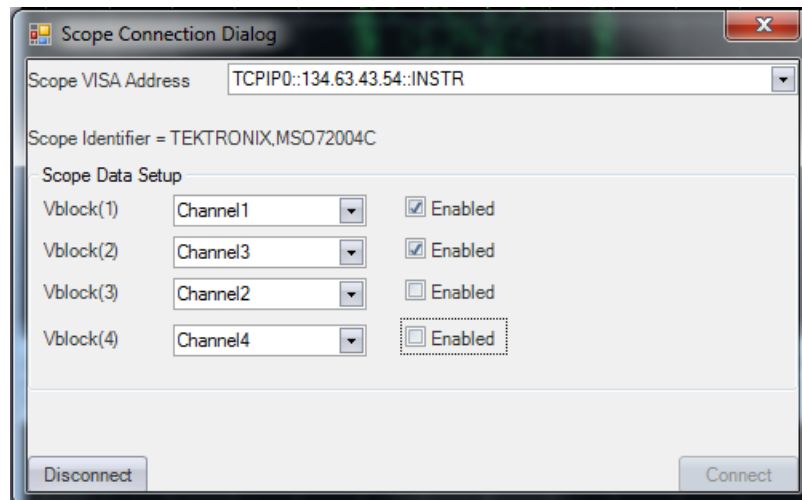
- Vblock(1) – X-polarization, In-Phase
- Vblock(2) – X-polarization, Quadrature
- Vblock(3) – Y-polarization, In-Phase
- Vblock(4) – Y-polarization, Quadrature

In the case below we disable two channels and set the other two to Channel 1 and Channel 3 since these can be active channels in 100Gs/s mode. The disabled channels must still have some sort of valid drop-down box choice. Do not leave the choice blank.

---

**NOTE.** *It is important to have the oscilloscope in single-acquisition mode (not Run mode). If you put the oscilloscope into Run mode to make some adjustment, please remember to press Single on the oscilloscope before connecting from the OUI.*

---



## Non-VISA oscilloscope connections (Scope Service Utility)

As mentioned in the previous section, the other method for connecting to the oscilloscope and collecting data is the Scope Service Utility (SSU). The SSU is a program that runs on each oscilloscope to be connected to the OUI.

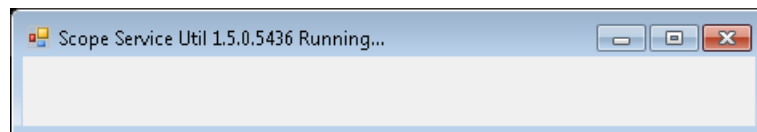
---

**NOTE.** *The Scope Service Utility runs on the target oscilloscope. Be sure to install the proper version of SSU for either real-time or equivalent-time (ET) oscilloscopes. See installation guide.*

---

Once the SSU is installed on the oscilloscope, start the “Socket Server” and the TekScope oscilloscope application before starting the SSU using the desktop icon.

The Scope Service has a small User Interface shown below.




---

**NOTE.** *It is best to have the oscilloscope in single-acquisition mode (not Run mode). The Scope Service Utility takes data directly from the oscilloscope memory and serves it up over a WCF interface to the OUI.*

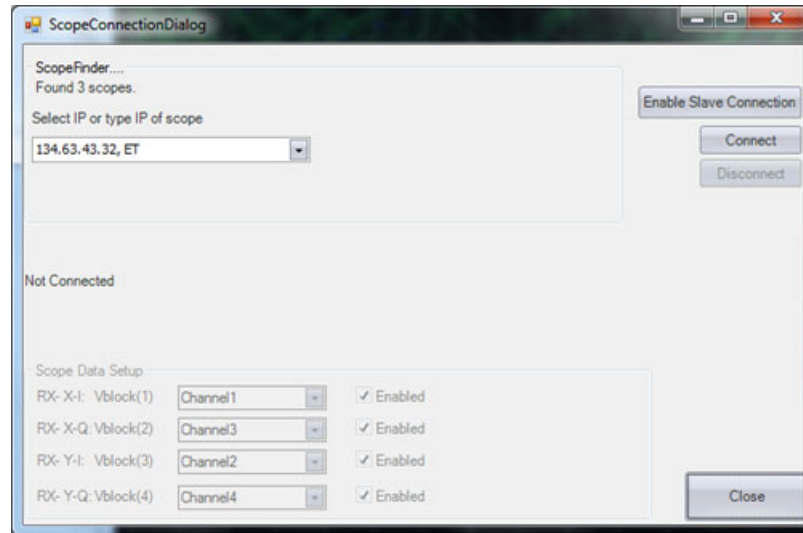
---

When connecting from the OUI, you will see a check box for VISA. Do not check the box unless you require a VISA connection.

---

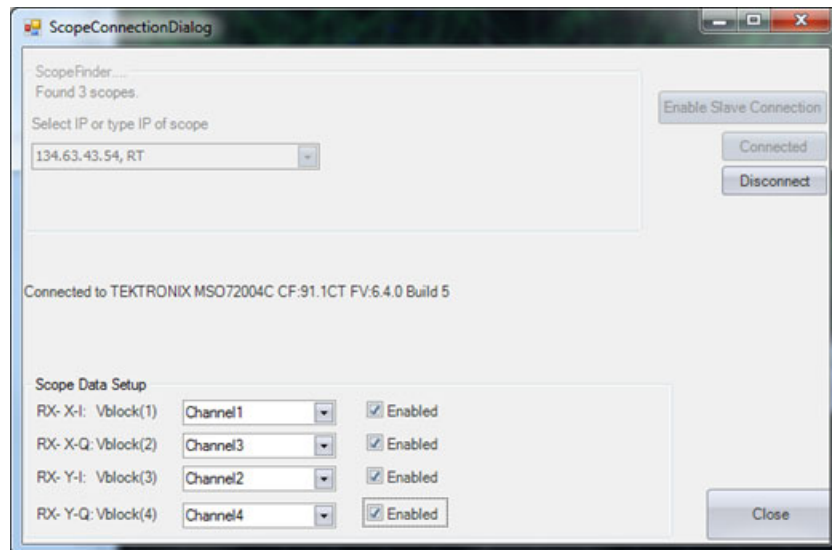
**NOTE.** Clicking *Connect* on the *OUI Setup Tab* opens the *Scope Connection dialog box* for connecting to the *Scope Service Utility*.

---



The green bar at the top indicates that the software is searching for oscilloscopes on the same subnet that are running the Scope Service Utility. As they are found they are added to the drop-down menu. If the OUI Scope Connection Dialog box reports 0 Scopes Found, you will have to type in the IP address manually. This happens when connecting over a VPN or when network policies prevent the IP broadcast. When typing the address in manually, do not include “, ET” or “, RT” on the end. Click **Connect**.

After connection, map the channels to the physical receiver channels and corresponding MATLAB variables as shown. This means that data from the selected channel will be moved into the indicated Vblock variable. Vblock(1) is X-Inphase, Vblock(2) is X-Quadrature, Vblock(3) is Y-Inphase, Vblock(4) is Y-Quadrature. The mapping you choose will depend on the cable connections made to the receiver.



Once connected and configured, close the connect dialog box. The OUI is ready to use.

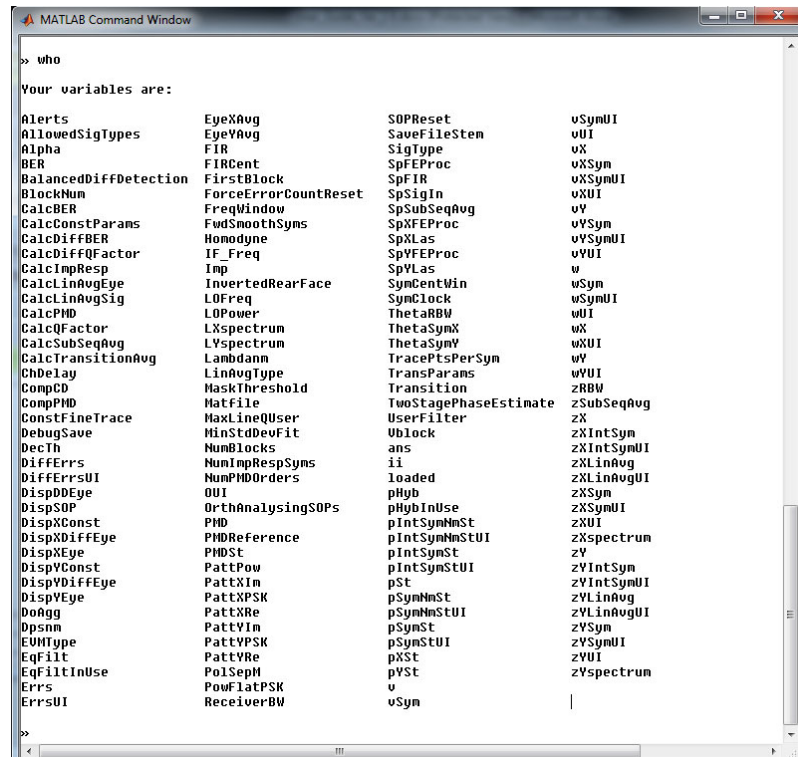
## Two-oscilloscope configuration

OUI Version 1.5 supports a configuration where two Tektronix MSO/DSO70000C- or D-Series oscilloscopes are both connected to an OM4000. (See page 147, *Configuring two Tektronix 70000 series oscilloscopes.*)

# MATLAB

Launching the OUI also launches the installed MATLAB application.

The MATLAB default working directory is the installation directory. Use the `cd` command to change to another directory if desired. Any files saved will go to the working directory. Once the OUI is running, the MATLAB Command Window is populated with the variables and functions used in coherent signal processing:



```

MATLAB Command Window
>> who
Your variables are:
Alerts                EyeXAvg              SOPReset             vSynUI
AllowedSigTypes      EyeYAvg              SaveFileStem        vUI
Alpha                 FIR                  SigType             vX
BER                   FIRCent              SpFProc              vXSyn
BalancedDiffDetection FirstBlock            SpFIR                vXSynUI
BlockNum              ForceErrorCountReset SpSigIn              vXUI
CalcBER                FreqWindow           SpSubSeqAvg          vY
CalcConstParams       FwdSmoothSyms        SpXFProc             vYSyn
CalcDiffBER            Homodyne              SpXLas               vYSynUI
CalcDiffQFactor        IF_Freq               SpYFProc             vYUI
CalcImpResp            Imp                    SpYLas               w
CalcLinAvgEye          InvertedRearFace      SynCentWin           wSyn
CalcLinAvgSig          LOFreq                SynClock             wSynUI
CalcPMD                LOPower              ThetaRBW             wUI
CalcQFactor            LXSpectrum           ThetaSynX            wX
CalcSubSeqAvg          LYSpectrum           ThetaSynY            wXUI
CalcTransitionAvg      Lambdamm              TracePtsPerSym       wY
ChDelay               LinAvgType            TransParams           wYUI
CompCD                 MaskThreshold         Transition            zRBW
CompPMD                MatFile               TwoStagePhaseEstimate zSubSeqAvg
ConstFineTrace        MaxLineQUser          UserFilter            zX
DebugSave              MinStdDevFit          Unblock              zXIntSyn
Dech                   NumBlocks             ans                   zXIntSynUI
DiffErrs               NumImpRespSyms        ii                    zXLinAvg
DiffErrsUI             NumPMDOrders          loaded                zXLinAvgUI
DispDDEye              OUI                   pHyb                 zXSyn
DispSOP                OrthAnalysingSOPs     pHybInUse            zXSynUI
DispXConst             PHD                   pIntSynNmSt          zXUI
DispXDiffEye           PHDReference          pIntSynNmStUI        zXSpectrum
DispXEye               PHDSt                pIntSynSt            zY
DispYConst             PattPow               pIntSynStUI          zYIntSyn
DispYDiffEye           PattXIm               pSt                  zYIntSynUI
DispYEye               PattXPSK              pSynNmSt             zYLinAvg
DoAvg                  PattXRe               pSynNmStUI           zYLinAvgUI
Dpsnm                  PattYIm               pSynSt               zYSyn
EUNType                PattVPSK              pSynStUI             zYSynUI
EqFilt                 PattVRe               pXSt                 zYUI
EqFiltInUse            PolSepM               pYSt                 zYSpectrum
Errs                    PowFlatPSK            v                     v
ErrsUI                 ReceiverBW            vSyn                  |
  
```

---

# Taking measurements

## Setting up your measurement

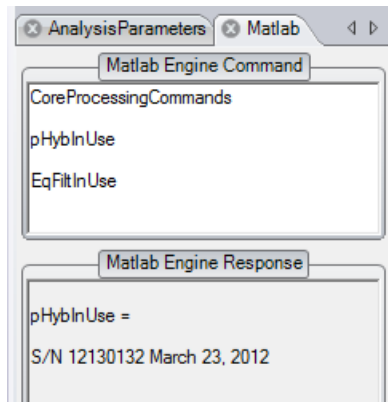
Since the OM4000 is a reconfigurable (complex, dual-polarized) reference receiver, it requires a modulated signal on the input fiber. Depending on the options configured in the receiver, this modulation can be single- or dual-polarized, with several formats available, including OOK (on-off keying), BPSK (binary phase-shift keying), and QPSK (quadrature phase-shift keying), and either coherent or differential QAM and other formats are also available.

The OM4000 includes two network-tunable sources (C- or L-band); you can use these or your own lasers for signal and reference inputs. Furthermore, each polarization can be independently driven by a distinct source, though all sources must be tuned to the same ITLA channel, or at least to the same wavelength. While no phase locking of the sources is necessary, the beatnote between any signal laser and the reference should be at a low enough frequency that the bandwidth of the real-time oscilloscope can accommodate the modulation bandwidth plus the beatnote frequency.

Before testing a modulated source, ensure that there is no modulation of the lasers (i.e. that your data source [pattern generator] is not enabled). With both lasers emitting and tuned to the same channel, use the fine-tune feature of the Laser/Receiver Control Panel to obtain a 100-500 MHz beat note on the oscilloscope. Then enable modulation by activating your data source or pattern generator.

## MATLAB Engine file

You can configure MATLAB to perform a wide range of mathematical operations on the raw or processed data using the Engine window. Normally the only call is to `CoreProcessingCommands`, the set of routines performing phase and clock recovery.



---

**NOTE.** To view a complete list of variables, open the MATLAB Command Window and enter *who*.

---

---

**NOTE.** *CoreProcessingCommands* will provide either ET or RT processing depending on which mode the OUI is in. To ensure you only get ET processing you can use *CoreProcessingET* in the window instead of *CoreProcessingCommands*. Similarly you can use *CoreProcessing* in the Engine Window if you want to be sure you only get real-time processed data.

---

As with all other settings, the last engine file used is recalled; you can locate or create another appropriate engine file and paste it into the OUI MATLAB window. Subsequent chapters explain in detail the operations of Core Processing. In addition to any valid MATLAB operations you may wish to use, there are some special variables that can be set or read from this window to control processing for a few special cases:

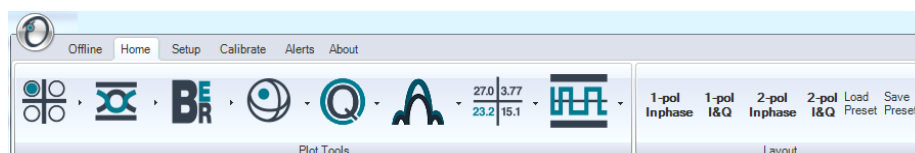


- EqFiltInUse – a string which contains the properties of the equalization filter in use
- pHybInUse – a string which contains the properties of the optical calibration in use
- DebugSave – logical variable that controls saving of detailed .mat files for analysis:
  - DebugSave = 1 in the MATLAB Engine Command window results in two files saved per block plus one final save.
  - DebugSave = 0 or empty suppresses .mat file saves.

## Taking measurements

Click **Single** and observe that the oscilloscope takes a burst of data; this confirms the connection. Use a short record length (for example, 2000 points) to speed up the display, click on Run-Stop to show continuously-updated measurements.

Using the Home tab, set up the plots you want, either using the stored Layout button or by clicking on the particular display format icon in the Plot Tools bar.



Displays can be rearranged within the UI window or dragged and positioned randomly on the Windows Desktop. Clicking and dragging a window using its title tab brings up a positioning guide. Hold down the left mouse button to position the window onto the positioning guide, then release to organize the plots.

Constellation and Eye plots can be rescaled by clicking on the relevant scaling icons in the Controls tab of the left panel of the UI. The scale in  $\sqrt{W}/\text{div}$  is indicated.

Options for each plot are accessed by right-clicking on the plot. Trace and symbol contrast can be set globally using the sliders on the left bar of the OUI. Set trace and symbol properties for a particular plot using the right-click on the plot.

## Multicarrier support (MCS) option

As network operators seek to increase the capacity of their fiber-optic transmission systems, moving wavelength division multiplexing (WDM) signals closer together is an attractive option. The densely packed signals are more readily separated using digital filters after coherent detection rather than more coarse WDM filters. This also simplifies routing since more is under digital control.

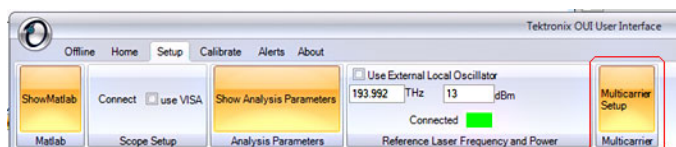
A group of tightly packed WDM channels is sometimes called a superchannel because of the many channels. It is also known as a multicarrier signal since the various channels come from separately modulated carriers. In this document the term multicarrier refers to the group, where channel refers to an individual modulated carrier.

The optional MCS function displays the results of multiple channels within a multicarrier signal at the same time. The MCS option can scan automatically between channels. Alternatively, the scan may be performed manually, and the OUI displays the results with the appropriate channel label by recognizing the channel based on the frequency entered by the user.

### The Multicarrier Setup window

The Multicarrier Setup window allows the user to define the carrier channel plan, to start and stop an automatic scan, and to define which channels will be included in separate axis plots Multicarrier Eye and Multicarrier Constellation. The window is divided into two sections: Multicarrier channel list and Multicarrier display layout. The absolute channel list is shown on the right, the relative version on the left.

When the MCS feature is enabled in the USB HASP key, the OUI displays the Multicarrier Setup button on the Setup ribbon.



**Figure 5: Multicarrier Setup button (Home ribbon)**

Click the Multicarrier Setup button to open the Multi Setup window.

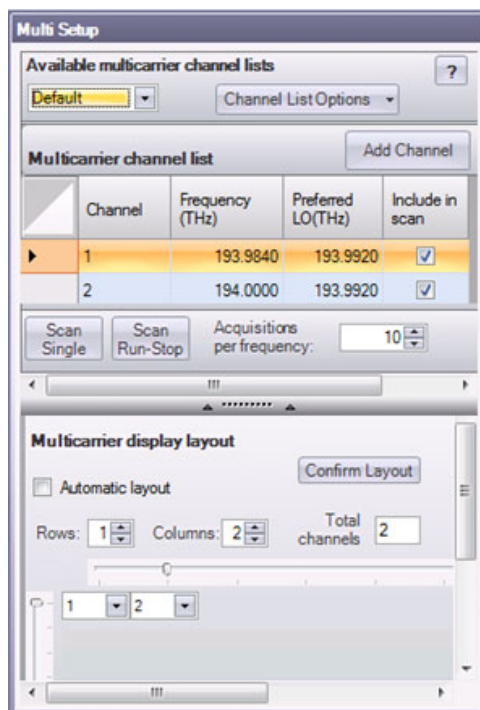


Figure 6: Multicarrier setup window

**Multicarrier channel list.** The main part of this section is the channel definition table. There are two types of channel definition table: absolute and relative. The default table type is absolute. A relative table may be entered by selecting Channel List Options: Add channel list: Add a new relative channel list. With an absolute channel definition table the channel frequencies specified are the actual optical frequencies, in the region of 195 THz. If the table is relative then the frequencies refer to the difference between the channel frequency and the current local oscillator frequency. Relative frequencies are specified in gigahertz.

The absolute channel definition table has four columns:

- The Channel column contains an integer identifying the channel. The values in this column do not have to be consecutive. The Frequency column contains the absolute channel frequency.
- The Preferred LO is the frequency that the local oscillator (also called the Reference Laser) is tuned to during an automatic scan to observe that channel. If the bandwidth of the multicarrier channels are so large that only one channel can be observed at a time given the bandwidth of the oscilloscope, then the Preferred LO is typically set to the same value as Frequency. If the channels have smaller bandwidth, then several table rows may be set to the same Preferred LO (even though the Frequency is different) so that all those channels are captured at the same local oscillator setting. This can save time, because tuning the local oscillator may be slow.

- The OUI identifies the channel by the difference between its absolute frequency (in the Frequency column) and the LO frequency.
- The final column decides whether a channel will be included in the automatic scan. The relative channel definition table has only three columns. The second column, called Offset Frequency, contains the difference frequency between the channel and current local oscillator frequency.

---

**NOTE.** *The terms “Reference Laser” and “Local Oscillator” (LO) are used interchangeably in the OUI and LRCP. The term LO is used here and in the channel list because it is more compact.*

---

When the Add Channel button is pressed a new table row is added. The new entry has a Channel number one higher than the previous entry. The frequency columns contain values that are incremented from the previous row by an amount equal to the difference between the previous row and the row preceding that. The user is free to edit all the new row's values. A table entry is removed by clicking on that row and pressing delete on the keyboard.

The user may enter several channel definition tables, and choose which one applies from the dropdown menu at the top left. A table may be deleted from the Channel List Options button.

The Scan Single button and Scan Run-Stop buttons start single and continuous automatic scans respectively. The OUI may take many acquisitions at each LO setting during the automatic scan, according to the Acquisitions per frequency control.

**Multicarrier display layout.** This section allows the user to specify which plots will appear in the separate axis plots, and how the subplots will be arranged. The Automatic layout checkbox (on by default) leaves the OUI to decide how to arrange the subplots. When Automatic layout is not checked, the region below becomes active. The user may choose the number of columns and rows of subplots either from the named controls, or by moving the horizontal and vertical sliders. Then the channel number to be plotted in each subplot is chosen from the dropdown menu.

### Multicarrier Plots

A number of new plots have been added to the OUI for the MCS option. These plots are only available when the MCS feature is found on the USB HASP key. The common feature of the multicarrier plots and tables is that they store information from prior data acquisition.

Since the display is organized to show the performance of an entire channel group while analyzing one at a time, only one portion of the plot or table is updating while the rest retains its last data. Use the Clear Data button in the left-pane Controls group to clear all stored data.

**Multicarrier spectrum.** The Multicarrier Spectrum plot is accessed by clicking on the spectrum icon button on the Home tab. Right-click on the plot to select what to display. By default the Input Signal spectrum is displayed. Each spectrum is labeled with its channel number appearing at its center frequency.



**Figure 7: Multicarrier spectrum context menu**

**Table 13: Multicarrier spectrum menu choices (right-click)**

Item	Description
Input signal	Displays the power spectrum of the input signal in dBW.
After front-end proc	Displays the power spectrum after digital filter is applied as specified in Analysis Parameters.
Front end filter	Displays the digital filter transfer function specified in Analysis Parameters. A control is provided to adjust where the plot is placed since the units are relative, not dBW.
Subsequence average	Displays the power spectrum of the averaged signal as calculated by the subsequence averaging function which is controlled by settings in Analysis Parameters.
After FE proc, X Poln	Displays the power spectrum of the transmitter's X-polarization signal after front-end processing.
After FE proc, Y Poln	Displays the power spectrum of the transmitter's Y-polarization signal after front-end processing.
Multicarrier channels	Choose which of the multicarrier channels to display.
Show new channels	Automatically display any new channels which are added to the Multicarrier channel list.
Save graphics to PNG file	Saves the current plot to a PNG file.

**Table 14: Multicarrier spectrum controls**

Item	Description
Freq/Div	Click the narrow spectrum icon (narrower spectrum, more GHz/Div) or wide spectrum icon (wider spectrum, less GHz/Div) to change the horizontal frequency axis scale. Units are selectable via the drop-down menu but also change automatically when a change is made to the Absolute/Relative/Autocenter choices.
Cent Freq	Click the left arrow (spectrum left, higher center frequency) or right arrow (spectrum right, lower center frequency) to change the center frequency value one full division at a time or type in the value directly. Units are selectable via the drop-down menu but also change automatically when a change is made to the Absolute/Relative/Autocenter choices.
Ref dBW	Use the +/- buttons to increase or decrease the power in dBW corresponding to the top of the plot.
dB/Div	Click the tall spectrum icon (taller spectrum, less dB/Div) or short spectrum icon (shorter spectrum, more dB/Div) to change the vertical axis scale.
Absolute/ Relative	Click to toggle between Absolute and Relative (relative to current Reference Laser frequency) frequency axis modes. A 194 THz signal with a 193.999 THz Reference Laser frequency will be displayed at 194THz in Absolute mode and at 1 GHz in Relative mode.
Autocenter	Click to turn Autocenter on or off. With Autocenter on, all channels will be shifted to zero Hz center frequency to facilitate visual comparison.

**Understanding the multicarrier spectrum plot.**

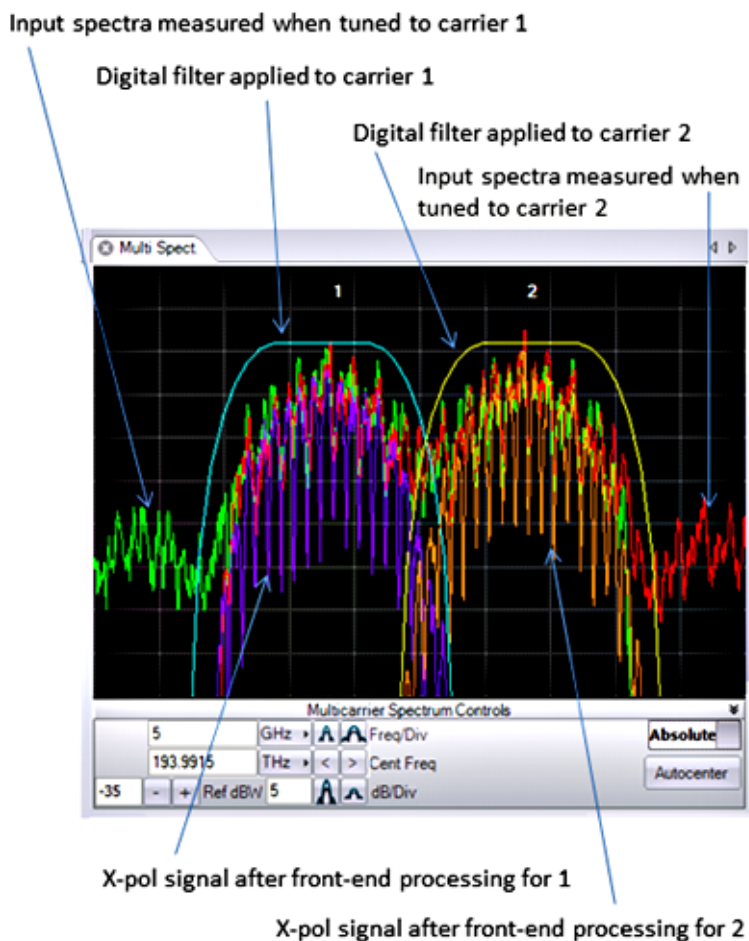


**Figure 8: Multicarrier spectrum plot**

Like the other multicarrier plots, the Multicarrier Spectrum plot saves data from each channel analyzed to form a composite view of the channel group. In the example shown above, this means that the Input Signal spectrum calculated while analyzing channel 1 is plotted in green together with that calculated while analyzing channel 2 in red. This scan was done with an absolute channel definition table.

Because a different LO frequency was specified for each channel, the red and green plots cover different spectral regions. When the LO was tuned to channel number 1, the green spectrum was found. When it was tuned to channel 2 it was the red spectrum. Since the channels are close together relative to the optical bandwidth of the system, there is substantial overlap between red and green. The red spectrum cuts off sharply on the left side corresponding to the lower limit of the optical system bandwidth whereas the green spectrum cuts off sharply on the right side corresponding to the upper edge of the optical system bandwidth for that LO tuning.

Ignoring the sharp cut-offs, the union of the red and green curves gives the true optical spectrum which is wider than what the 23 GHz receiver used here could achieve in a single LO tuning.



**Figure 9: Multicarrier spectrum plot details**

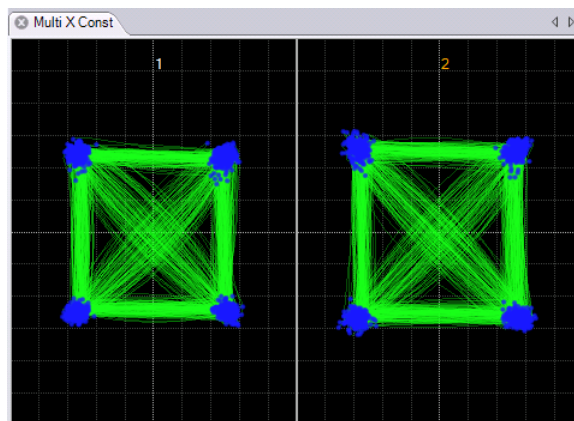
The figure above shows some of the other features of the Multicarrier Spectrum plot. Here the X-pol signal after front-end processing is shown together with the digital filter used. Notice that the X-pol signal (purple or orange) show much deeper nulls than the total power spectrum (red or green). The nulls are the result of using a split and delay technique to generate the electrical I and Q modulator input signals from one data stream. There is a null at integer multiples of one over the difference in cable delays.

Since different cables are used on the X and Y inputs of the modulator, the nulls do not perfectly align and so wash out when looking at the total power spectrum. The nulls are clearly visible once the transmitter polarization signals have been separated.

**Multicarrier Constellation Plots.** The Multicarrier Constellation plots are accessed by clicking on the constellation icon button on the Home tab. These plots behave in a similar fashion to the existing constellation plots except that there are regions reserved for each channel. The layout is controlled by the Multicarrier Setup window described above.



As each channel is analyzed, only that portion of the plot will be updated while the most recent data displayed will continue to be shown in the other regions so that an aggregate view of the multicarrier group can be displayed. Use the Clear Data button to discard prior data.



**Figure 10: Multicarrier constellation plots**

**Multicarrier Eye diagrams.** The Multicarrier Eye Diagrams are accessed by clicking on the Eye icon button on the Home tab. These plots behave in a similar fashion to the existing eye plots except that there are regions reserved for each channel. The layout is controlled by the Multicarrier Setup window described above.

As each channel is analyzed, only that portion of the plot will be updated while the most recent data displayed will continue to be shown in the other regions so that an aggregate view of the multicarrier group can be displayed. Use the Clear Data button to discard prior data.

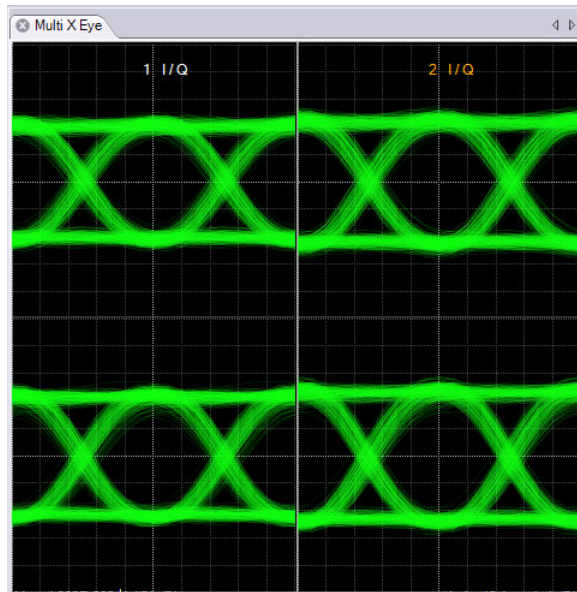


Figure 11: Multicarrier Eye diagrams plot

**EVM vs. Channel and Q vs. Channel.** The EVM vs. Channel and Q vs. Channel plots are accessed by clicking on the Q icon button on the Home tab. These plots display the most recently measured EVM or Q factor for each channel. Only the current channel will be updated while the most recent data displayed will continue to be shown for the other channels so that an aggregate plot of the multicarrier group can be displayed. Use the Clear Data button to discard prior data.

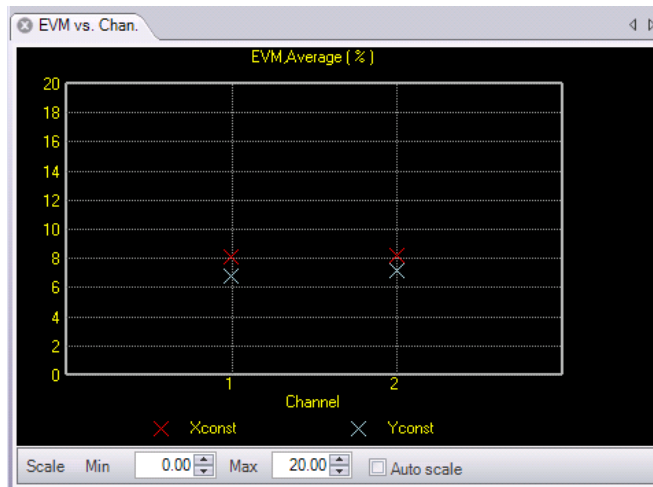


Figure 12: EVM vs. Channel plot

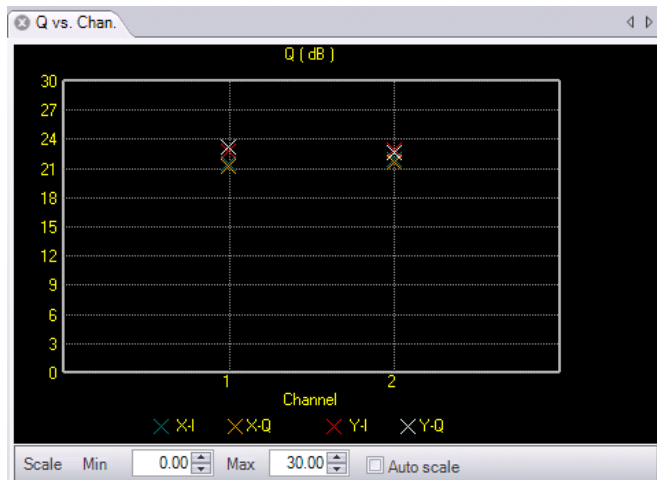


Figure 13: Q vs. Channel plot

**Measurement vs. Channel.** The Measurement vs. Channel table is accessed by clicking on the tabular data icon on the Home tab. This table is similar to the Measurement Statistics table except that only the most recent value is shown so that data from every channel can be displayed in one plot. As with the Measurement Statistics plot, it is necessary to make sure the desired measurement is enabled in the Analysis Parameters since some measurements are not calculated unless specifically enabled.

Measurement	Unit	Ch-All	Ch-1	Ch-2
<b>X - Eye</b>				
X-Q Q-Factor	dB	21.607	21.283	21.607
X-Q Eye Height	$\sqrt{mW}$	1.519	1.419	1.519
X-Q Rail 0 Std Dev	$\sqrt{mW}$	0.048	0.047	0.048
X-Q Rail 1 Std Dev	$\sqrt{mW}$	0.078	0.076	0.078
<b>X-I Eye</b>				
X-I Q-Factor	dB	21.901	21.565	21.901
X-I Eye Height	$\sqrt{mW}$	1.429	1.333	1.429
X-I Rail 0 Std Dev	$\sqrt{mW}$	0.050	0.051	0.050
X-I Rail 1 Std Dev	$\sqrt{mW}$	0.064	0.060	0.064
<b>Y - Eye</b>				
<b>X - Const</b>				
<b>Y - Const</b>				
<b>X - Trans</b>				
<b>Y - Trans</b>				
<b>Pow - Trans</b>				
<b>XY Measurements</b>				
<b>PMD</b>				

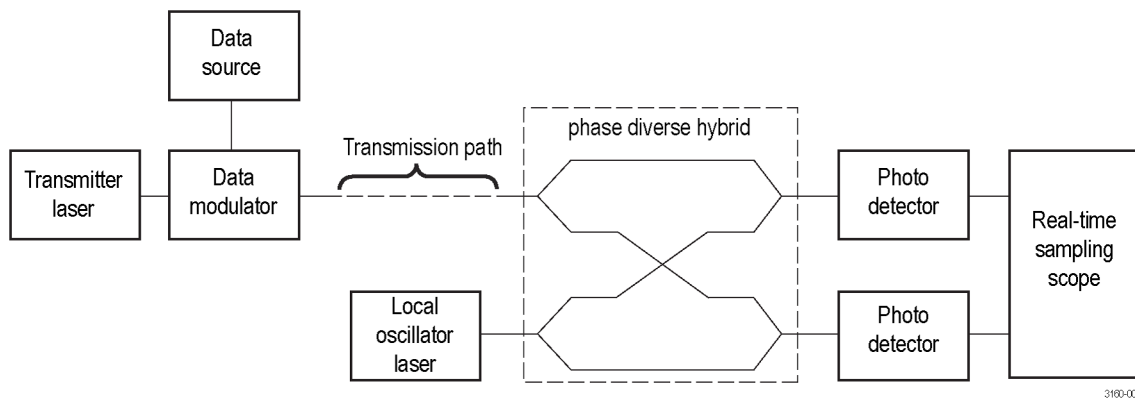
Figure 14: Meas vs. Channel table



## Detailed configuration of experiments

Coherent detection has been recognized for some time as offering superior performance to direct detection. Until recently coherent receivers were prohibitively expensive, and that is why all commercial optical communications receivers use direct detection. However, a coherent receiver which uses real-time digital signal processing technology can meet both the performance and cost needs of future optical communications networks.

Most researchers have not built an actual digital processor that operates at the relevant data rates, 10 Gb/s and higher, because of the large amount of resources that task would consume. Instead they have employed a real-time sampling oscilloscope, and then processed a short measurement burst offline on a convenient computing platform. The references listed below all describe this kind of measurement. Although the approach has been used to date to demonstrate optical communications systems, it can also be used to record the electric field of an optical signal, to study signal distortions, or to characterize active and passive optical components.



## References

M.G. Taylor, "Coherent detection method using DSP for demodulation of signal and subsequent equalization of propagation impairments," *IEEE Phot. Tech. Lett.*, vol. 16, no. 2, p. 674-676, 2004.

M.G. Taylor, "Measurement of phase diagrams of optical communication signals using sampled coherent detection," *NIST Tech. Dig.: Symp. Optical Fiber Measurement*, Boulder, CO, vol. 1024, p. 163-166, Sep. 2004.

D.-S. Ly-Gagnon, K. Katoh, K. Kikuchi, "Unrepeated optical transmission of 20 Gbit/s quadrature phase-shift keying signals over 210km using homodyne phase-diversity receiver and digital signal processing," *IEE Electron. Lett.*, vol. 41, no. 4, p. 59-60, 2005.

S. Tsukamoto, D.-S. Ly-Gagnon, K. Katoh, K. Kikuchi, "Coherent Demodulation of 40-Gbit/s Polarization-Multiplexed QPSK Signals with 16-GHz Spacing after 200-km Transmission," OFC 2005 conference, Anaheim, US, PDP29, 2005.

M.G. Taylor, "Accurate Digital Phase Estimation Process for Coherent Detection Using a Parallel Digital Processor," ECOC 2005 conference, Glasgow, UK, paper Tu4.2.6, Sep. 2005.

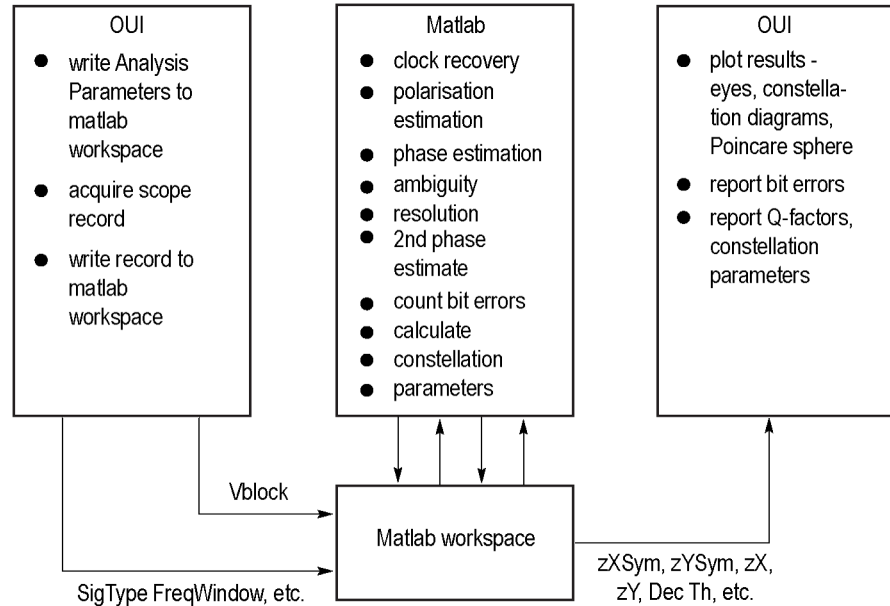
D.-S. Ly-Gagnon, S. Tsukamoto, K. Katoh, K. Kikuchi, "Coherent detection of optical quadrature phase-shift keying signals with carrier phase estimation," IEEE J. Lightwave Technol., vol. 24, no. 1, p. 12-21, 2006.

K. Kikuchi, "Phase-Diversity Homodyne Receiver for Coherent Optical Communications," COTA 2006 conference, Whistler, Canada, paper CThB3, 2006.

S.J. Savory, A.D. Stewart, S. Wood, G. Gavioli, M.G. Taylor, R.I. Killey, P. Bayvel, "Digital Equalization of 40Gbit/s per Wavelength Transmission over 2480km of Standard Fibre without Optical Dispersion Compensation," ECOC 2006 conference, Cannes, France, paper Th2.5.5, Sep. 2006.

# Core processing software guide

## Interaction with OUI



3160-007

The core processing software performs all the computations to obtain the quantities displayed in the OM4000 User Interface (OUI), starting from the raw data records acquired by the oscilloscope. Core processing is written in MATLAB. The code is executed on an instance of MATLAB launched and controlled in engine mode by the OUI. The core processing code and the OUI exchange data via the MATLAB workspace.

The order of processing for each acquisition is as follows:

1. OUI launches MATLAB engine
2. OUI writes variables to MATLAB workspace corresponding to settings in the OUI Analysis Parameters window
3. OUI fetches a record from oscilloscope, and writes to MATLAB workspace (in variable `Vblock`)
4. OUI executes commands listed in MATLAB Engine Commands window

By default, the MATLAB Engine Commands window contains one instruction, `CoreProcessing`, so the m-file `CoreProcessing.m` is executed. This file computes the various output variables which then reside in the MATLAB workspace.

5. OUI retrieves output variables from MATLAB workspace
6. OUI displays output variables as eye diagrams, constellation diagrams, numerical values, etc.

In fact when the record size is larger than the block size multiples calls to CoreProcessing are executed for each oscilloscope record. This mode of processing is known as block processing. (See page 87, *Block processing*.)

To customize the signal processing of the OM4000, the user must modify the commands in the OUI's MATLAB Commands Window, or modify CoreProcessing.m, or both. Commands can be added to the MATLAB Commands Window following the call to CoreProcessing if additional processing is desired after CoreProcessing has completed. If deeper changes to the signal processing are needed then it is necessary to modify CoreProcessing.m.

This Section will discuss the code in CoreProcessing.m, and how it derives the output variables from the scope record Vblock.

## MATLAB variables

MATLAB is a loosely typed language. All numerical variables are created as reals by default, and arrays and structured variables are sized as they are created and manipulated. The core processing software follows some rules of its own with regard to variable structure and naming. A variable representing a quantity that varies with time is a structured variable having (at least) three fields:

- .t0 – time of first element in seconds
- .dt – time separation between elements
- .Values – actual values of elements

The time dimension of the .Values field extends from left to right, dimension 2 in MATLAB terms. The number of rows of the .Values field depends on the nature of the variable it represents. A voltage has one row of real numbers. A phase factor has one row of complex numbers. The electric field of an optical signal has two rows of complex numbers, for the two states of polarization, and can be thought of as a row of Jones vectors. Similarly, a Stokes vector as a function of time has three rows of reals.

Variables representing a Jones vector vs. time typically begin with “p”. Variables representing a phasor factor, having inphase and quadrature components, begin with “z”. Variables having sample times at the center of the symbols in a digital comms signal contain the letters “Sym”. Variables representing the X or Y polarizations contain the letter “X” or “Y”. Variables representing the inphase part of a signal contain “Re”, and the quadrature part contain “Im”.

For example, the X polarization of a signal at the symbol centers is stored in variable zXSym. The data sequence (for example, PRBS sequence) encoded on the inphase part of the X polarization of a signal is defined in variable PattXRe.



Some of the important input variables to CoreProcessing are listed in the *Core Processing function reference* section. (See page 91.)

## MATLAB functions

The core processing code calls several MATLAB functions, that are key to processing the signal. A function typically acts on many variables as input, and produces many variables as an output. A function call has the form:

```
[OutParam1,OutParam2 ... BoundVals,Alerts] =  
FunctionName(InParam1,InParam2 ... BoundVals,Alerts)
```

Where:

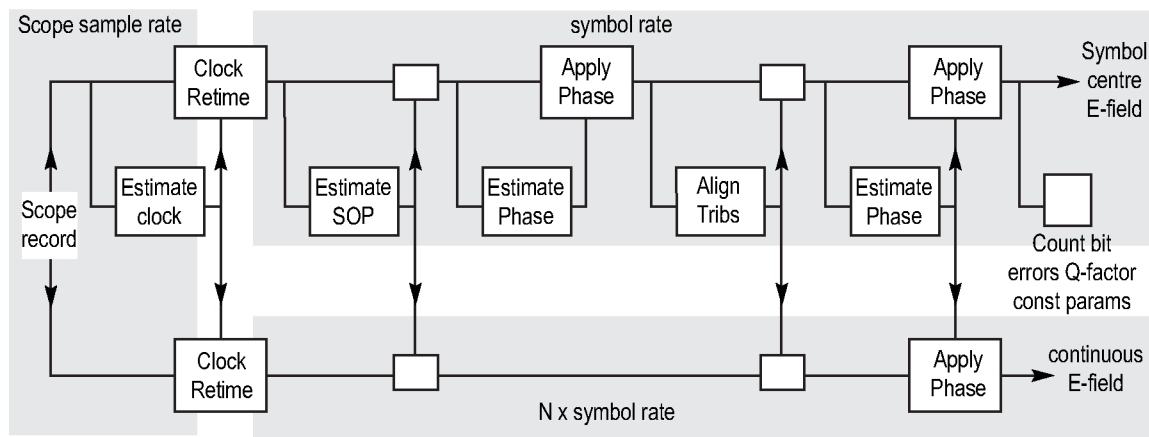
- BoundVals is used in block processing mode (See page 87, *Block processing*.) It contains information about the tail end of the previous block, and is used to ensure continuity with the next block.
- Alerts is a variable containing status information, typically warnings and error messages, accumulated from all the functions called so far. (See page 88, *Alerts management*.)
- The other variables contain the actual inputs and outputs of the function.

While the code of CoreProcessing.m is visible to the user, the code for many of the functions is not. Each function includes extensive parameter checking to make sure that the inputs passed to it are reasonable. An error message will be produced if an input variable is out of range or has missing fields, and the message says what is wrong. However it is possible to cause an error that cannot be trapped by having the input variables in the wrong order. Each function has a detailed description. (See page 91, *Core Processing function reference*.) Help text is available by typing **help FunctionName** at the MATLAB prompt.

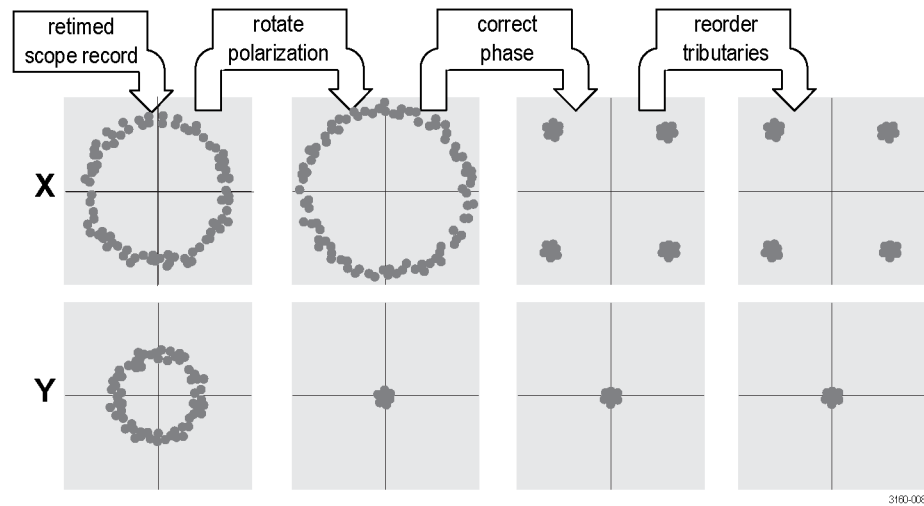
## Signal processing steps in CoreProcessing

CoreProcessing.m is a long file because it includes all the signal processing options for the many possible modulation formats. There are several “switch SigType” statements, where one case includes some repetition of code from an earlier case. If a user is interested in only one modulation format it is expedient to delete all the case statements that are not of interest (after making a copy of the original CoreProcessing.m). The resulting file will be shorter and easier to navigate.

A file CoreProcessing1chQPSK .m, located in the same folder as CoreProcessing.m, has been prepared in this way to process single polarization QPSK signals. It can be called by replacing the line in the MATLAB Commands Window “CoreProcessing” with “CoreProcessing1chQPSK”. The processing stages in CoreProcessing1chQPSK will be studied now, as an example of how the full CoreProcessing works. The headings of the sections that follow are the same as the headings (comment lines) within CoreProcessing1chQPSK. The following diagram shows how the processing is applied to signals at three different sample rates.



The appearance of the single channel QPSK signal is shown below, as the processing progresses. The signal has X and Y polarization components, each of which is shown as a plot on the complex plane.



## Clock recovery

The variable `Vblock` is a 1x4 array of data structures, and contains the four oscilloscope channel voltages vs. time. `Vblock(1)` is a time series (having `.t0`, `.dt` and `.Values` fields) of scope channel 1 voltage, and similarly for `Vblock(2)` ... `Vblock(4)`.

The first step of clock recovery is to find the symbol clock frequency and phase, via a call to the function `EstimateClock`. (See page 97, *EstimateClock*.) In addition to `Vblock`, `EstimateClock` takes as inputs `ChDelay` and `pHyb`. `ChDelay` contains the relative delays between the four scope channels, due to cable length mismatches for example. It is obtained from the delay calibration routine. (See page 133, *Delay adjustment (system deskew) (ET)*.) (See page 111, *Delay adjustment (system deskew) RT*.)

`pHyb` is a 2x8 array of complex values that describes any non-idealities in the optical hybrid within the OM4000. If the phase angle deviates from 90° or if the polarizations are not perfectly orthogonal, that is included in `pHyb` and corrected within `CoreProcessing`.

The main output of `EstimateClock` is `SymClock`, which has fields `.t0` and `.dt`. `SymClock` contains the recovered symbol clock and phase.

Next, `ClockRetime` uses the newly recovered symbol clock to retime the raw scope records `Vblock` to a time series of Jones vectors, output `pSym`. Each element of `pSym` is timed at the center of the symbol. `ClockRetime` also uses calibration parameters `ChDelay` and `pHyb`, so that it corrects for non-idealities in the receiver.

## Initial polarization estimate

The optical signal typically has a random state of polarization compared to the axes of the hybrid within the OM4000. This means that each polarization of `pSym` (each of the two rows of `pSym.Values`) contains some of the signal content. The next step is to apply a polarization rotation so that one polarization (the X polarization) contains the signal, and the other has only a low level of noise.

This worked example is for a single polarization QPSK signal. If a dual polarization signal were used, then before the polarization rotation each polarization of pSym would contain a mixture of the two polarization tributaries. After polarization rotation the top row of pSym.Values would contain one polarization tributary, and the bottom row the other.

EstimateSOP has pSym as an input, and produces a Jones matrix RotM as an output. RotM is a type of matrix called an orthogonal matrix, which represents a rotation. Then pSym is rotated by the inverse of RotM to separate the X and Y tributaries. zXSym is assigned to the X polarization part, and zYSym the Y polarization part. zXSym and zYSym are time series of complex numbers, not Jones vectors like pSym.

### Initial phase estimate

zXSym contains the signal, but it does not appear as four separate constellation points yet, because the phase of the signal is continually changing. Instead zXSym appears as a ring of points. The phase is calculated by EstimatePhase and assigned to variable ThetaSymX. The phase has two components. ThetaSymX.CentFreq is the heterodyne frequency, the difference frequency between the center of the signal spectrum and the local oscillator. The constellation effectively spins at this difference frequency before phase correction. ThetaSymX.Values contains the random phase component, in radians, which arises from phase noise in the signal and local oscillator lasers.

zXSym is corrected for ThetaSymX by the function ApplyPhase. At this point zXSym appears as a conventional QPSK signal, four clusters of constellation points on the corners of a square.

### Align signal tributaries with data content

Although the signal appears to be a good QPSK constellation, a further adjustment in phase is typically required. The phase estimation algorithm in EstimatePhase randomly produces a phase offset which is a multiple of 90° away from the true phase. The constellation may be 0, 1, 2 or 3 quarter turns from the true constellation. The function AlignTlibs compares the actual data content of the two tributaries with the known data content specified in PattXRe and PattXIm.

These two variables are loaded by the OUI with the data patterns specified in the Analysis Parameters window. (See page 32, *Analysis Parameters window*.) If the inphase component of zXSym contains the data pattern of the quadrature component, and vice versa, then AlignTlibs produces an output DRotM which is equivalent to a quarter turn to correct the phase of zXSym. AlignTlibs decides the phase correction. (See page 91, *AlignTlibs*.) The function uses several criteria, including tie-breaker criteria if both tributaries have the same data pattern.

When a dual polarization signal is used AlignTlibs may exchange the SOPs as well as adjusting the phase. If AlignTlibs were not applied then the tributaries would be randomly mapped to different components in the OUI display each acquisition. If one tributary had different features from the others, for example a bias offset, then that feature would appear to randomly jump between the different eye diagrams each acquisition, instead of staying in the same place.

AlignTrib returns the pattern variables, PattXRe etc., as outputs in addition to DRotM. The pattern variable outputs have additional fields compared to the pattern variable inputs, to indicate the synchronization location within the pattern. For example, if PattXRe specifies a pseudorandom bit sequence (PRBS), the output PattXRe has a field .Seed which contains the contents of the PRBS shift register at time PattXRe.t0. Downstream functions in CoreProcessing.m are able to generate the data sequence on the tributary from PattXRe.

## Second phase estimate

At this point the signal in zXSym appears ready to make a decision and count the bit error rate. The constellation has the correct phase, comprising four tight clusters, and the true data content on each tributary is known. However, in cases where there is considerable phase noise (where the laser linewidths are not narrow) the estimated phase may contain cycle slips. The phase is correct for the initial portion of the record, and then after a cycle slip it becomes in error by a quarter of a turn. This means the bit error rate is low or zero for the early portion, and then suddenly rises to 0.5.

In principle the cycle slips can be avoided, given that the true data content of the signal is known, and the purpose of the second phase estimate is to calculate the phase without cycle slips. The starting point is the variable pSym, containing the Jones vector of the signal resampled from the oscilloscope record. The correct polarization rotation is applied to pSym, and the X component assigned to zXSym. At this point zXSym can be thought of as the four-state QPSK constellation rotating at the phase difference between signal and LO.

Next zXSym is multiplied by the converse of the (known) true data modulation on the signal. This operation has the effect of removing the data modulation, so the signal is equivalent to a single constellation state rotating at the phase difference between signal and LO. EstimatePhase is called, to calculate the phase. This time the SigType parameter passed to EstimatePhase is set to 0. With EstimatePhase, SigType = 0 tells the function it is an unmodulated signal. The phase estimate algorithm does not raise to the 4th power, with the inherent four-way ambiguity in phase following the subsequent 4th root operation. The resulting ThetaSymX does not contain cycle slips, no matter how high the level of phase noise.

With a dual polarization signal type, there is also a second polarization estimate. The second SOP estimate is typically more accurate than the original SOP estimate. The second SOP estimate proceeds in the same way as the second phase estimate. The signal is multiplied by the converse of the known data modulation, and EstimateSOP is applied with SigType = 0, as if it were an unmodulated signal.

## Apply polarization & phase estimates

The new cycle slip-free phase estimate ThetaSymX is applied to the resampled oscilloscope record pSym via ApplyPhase, to produce a new zXSym variable. This new zXSym inherently does not contain cycle slips.

**Count bit errors**

A decision is made on each component of the QPSK signal by testing whether  $zXSym > 0$ . The decided values are compared with the known true data values, via the xor function, to locate bit errors. The number of bit errors is stored in variable Errs.

The decision threshold Q-factor of each component is calculated using the QDecTh function. The outputs of QDecTh are the Q-factor and also the points of inverse error function vs. decision threshold that are seen in the Q plot in the OUI.

The mean and standard deviation of the constellation points are calculated later in EngineCommandBlock. (See page 87, *Block processing*.) These quantities are based on the sum and sum-of-squares of the constellation points, which are calculated as fields of zXSym.

**Calculate signal vs. time  
on fine time grid**

All of the manipulations of the signal so far have been on a representation of the signal at one sample per symbol, timed at the center of the symbol. The various eye diagrams and constellation diagrams in the OUI contain traces that appear as smooth lines over time. The final task in CoreProcessing is to calculate these fine traces.

The first call to function ClockRetime (See page 83, *Clock recovery*.) used SymClock as the input variable defining the clock frequency and phase. ClockRetime is called again, this time with TraceClock as the clock input variable. TraceClock has field .dt which is TracePtsPerSym time smaller than SymClock.dt. The output of this call to ClockRetime is assigned to variable p.

The polarization and phase adjustment operations are repeated with fine trace variable p. It is multiplied by the polarization rotation Jones matrix, and the X and Y polarizations are separated (into variables zX and zY). The phase adjustment is applied to zX and zY via the ApplyPhase function. zX and zY contain the fine traces that are displayed as eye diagrams. For single polarization QPSK format signals, zX contains a modulated waveform, while zY contains only a low level of noise.

## Block processing

The OM4000 core processing software is able to cope with very large record sizes, for example 250M samples, by breaking down the record into smaller pieces, or blocks, and processing them in sequence. All functions are designed so that the output is near-identical if block processing is used compared to performing the processing in a single block. Each function collects information about the signal and any relevant internal variables at the end of the time period of one block, and then effectively tags that information onto the start of the next block. The result is a seamless transition from one block to the next.

Block processing is an advanced feature. It applies only if the record size is larger than the block size set in the OUI, and typical computing hardware can cope with block sizes larger than 100 k samples. A user wishing to customize the core processing software does not have to be concerned about block processing unless large record sizes have to be processed.

The complete steps of interaction between the OUI and CoreProcessing, taking block processing into account, are listed below. Italics are used to highlight a change compared to the simpler description in *Interaction with OUI*. (See page 79.)

1. OUI launches MATLAB engine
2. OUI writes variables to MATLAB workspace corresponding to settings in the OUI Analysis Parameters window
3. OUI fetches one block of a record from oscilloscope, and writes to MATLAB workspace
4. OUI executes EngineCommandInit
5. Loop over blocks until scope record is exhausted
  - a. OUI executes commands listed in MATLAB Engine Commands window
  - b. OUI executes EngineCommandBlock
6. End of loop over blocks
7. OUI executes EngineCommandPost
8. OUI retrieves output variables from MATLAB workspace
9. OUI displays output variables as eye diagrams, constellation diagrams, numerical values, etc.

The way the information is passed from one block to the next is via a boundary values variable (which includes “BoundVals” in its name). The boundary values variables are declared as persistent variables in CoreProcessing.m. They are listed after the keyword “persistent” at the head of CoreProcessing. The BoundVals output variable returned by a function contains information from the latter part of the block. The same variable is passed as the input to the function when it is called during the next block. For the first block the boundary values variables are initialized to empty variables. The variable FirstBlock is set by the OUI, and is 1 only for the first block. The initialization statements appear at the start of CoreProcessing.m, bracketed by “if FirstBlock”.

When a function is passed an empty variable as a boundary values input, it knows it is dealing with the first block. The results of the different blocks must be stitched together to form contiguous output variables. This is done in EngineCommandBlock.m.

Many variables have two versions: a per-block version, and an aggregated version having the same name suffixed by “UI”. For example, the X component of the field fine trace is stored in zX when CoreProcessing (one block) has finished executing, while variable zXUI contains that parameter for the whole oscilloscope record.

The function Aggregate is called many times in EngineCommandBlock. It is a multipurpose function that aggregates the latest block result of a variable into the UI version of that variable. For example, the following line of code appears in EngineCommandBlock `zXUI = Aggregate(zXUI,zX)`; The results displayed in the OUI, as plots or text, are the aggregated UI versions of the variable. They refer to all of the oscilloscope record processed so far, and not just the most recent block.

## Alerts management

In addition to their numerical variable outputs, the functions in CoreProcessing can report the occurrence of specific events via the Alerts variable. Alerts is passed as an input to each function, and returned as an output. If an event occurs a message is appended to the Alerts structure variable. The Alerts section of the main ribbon in the OUI then displays a list of all the alert messages that apply. (See page 52, *Alerts*.)

The Alerts variable has a field .Active which is a vector of structures containing the active alert messages. Each element of Alerts.Active has the following fields:

- .Zone – a string saying where the functions was called from (more details below)
- .FunctionName – name of function where alert was activated
- .Code – an integer value unique to that alert message
- .Msg – a string stating the nature of the alert
- .TimesAsserted – an integer equal to the number of times the alert has been activated (automatically assigned by AssertAlert)



These fields correspond to the columns of the Alerts table in the OUI.

The purpose of the `.Zone` field is to identify not just which function triggered the alert, but where in the code it was triggered. The value of the `.Zone` field is assigned to the value of `Alerts.CurrZone` when the function triggering the alert is called. Referring to `CoreProcessing1chQPSK.m`, at the start of each new section of the file `Alerts.CurrZone` is assigned to a new string, identifying the stage of processing. As an example, `EstimatePhase` is called twice, in the sections for first phase estimate and second phase estimate. When an alert occurs in `EstimatePhase` the location is reported in the `Zone` column of the Alerts table.

The `.Code` field, a unique integer, is available to conditionally execute code depending on whether an alert has occurred.

### Writing a function that uses the Alerts variable

The Alerts variable is by convention declared as the final parameter at both input and output. MATLAB sometimes requires it to have a different name at input and output in the function declaration, depending on how the function is called, so in core processing functions it is named `AlertsIn` and `AlertsOut`. All the alert messages are defined before the main part of the function, using code of the following form:

```
persistent ZonesAlreadyCalledFrom
AllAlerts = [];
AllAlerts = [AllAlerts,struct('Code',<code number 1>,'Msg',
<first alert message>)];
AllAlerts = [AllAlerts,struct('Code',<code number 2>,'Msg',
<second alert message>)];
... <more alert definitions> ...
FunctionName = <name of function>;
[AllAlerts.FunctionName] = deal(FunctionName);
AlertsOut = AlertsIn;
if ~isfield(AlertsOut,'CurrZone')||
    isempty(AlertsOut.CurrZone)
    CurrZone = '';
else
    CurrZone = AlertsOut.CurrZone
end
if isempty(ZonesAlreadyCalledFrom)
    AlertsOut = RegisterAlerts(AllAlerts,AlertsOut);
    ZonesAlreadyCalledFrom = {[CurrZone,'x']};
elseif ~any(strcmp(ZonesAlreadyCalledFrom,[CurrZone,'x']))
    AlertsOut = RegisterAlerts(AllAlerts,AlertsOut);
    ZonesAlreadyCalledFrom = [ZonesAlreadyCalledFrom;
    [CurrZone,'x']];
end
```

The integer Code values <code number 1> and <code number 2> must be different from one another and from any other alert code values using in core processing. To save the user from searching through all the alert codes in use, the first time core processing is executed (or after entering “clear all” at the MATLAB prompt) an error is generated if two calls to RegisterAlerts anywhere in the core processing software have the same alert code but different function names or different alert messages.

The following code is used in the body of the function to trigger the alert.

```
if <alert condition>
    AlertsOut = AssertAlert(<code
        number>,AllAlerts,AlertsOut);
end
```

---

# Core Processing function reference

## AlignTribes

[Rot,PattXReOut,PattXImOut,PattYReOut,PattYImOut,BoundValsOut,AlertsOut]  
= AlignTribes(pSym,SigType,PattXReIn,PattXImIn,PattYReIn,PattYImIn,  
BoundValsIn,AlertsIn)

- pSym – single or dual polarization parameter vs. time:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – 1xN or 2xN array of complex values
- SigType – integer value indicating signal modulation format
- PattXReIn – data pattern of real part of X polarization
- PattXImIn – data pattern of imag part of X polarization
- PattYReIn – data pattern of real part of Y polarization
- PattYImIn – data pattern of imag part of Y polarization
- BoundValsIn – structure of boundary values from previous block
- AlertsIn – structure of alerts accumulated before executing function
- PattXReOut – data pattern of real part of X polarization, including synchronization parameters
- PattXImOut – data pattern of imag part of X polarization, including synchronization parameters
- PattYReOut – data pattern of real part of Y polarization, including synchronization parameters
- PattYImOut – data pattern of imag part of Y polarization, including synchronization parameters
- BoundValsOut – structure of boundary values to be passed to next block
- AlertsOut – structure of alerts including any alerts raised during execution of function

AlignTribes performs ambiguity resolution. The function acts on variable pSym which is already corrected for phase and state of polarization, but for which the tributaries have not been ordered. AlignTribes uses the data content of the tributaries to distinguish between them.

pSym may be a dual polarization of single polarization parameter. The result of aligning the tributaries is reported in RotM. When input parameter pSym is single polarization RotM is a complex number, and when pSym is dual polarization RotM is a 2x2 Jones matrix. RotM is either a phase factor (single polarization) or an orthogonal (rotation) matrix (dual polarization). Unlike the output of EstimateSOP, it can only take on a discrete set of values. RotM represents a whole number of quarter turns, in polarization space and/or phase space, as appropriate to the modulation format. For example, if the modulation format is QPSK, RotM can take on values 1,i,-1,-i; for BPSK it takes on values 1,-1. pSym is transformed to align correctly with the given data patterns by multiplying by the inverse of RotM.

The four data pattern input parameters are structure-type variables. Each may define a pseudo-random bit sequence (PRBS) or a specific data pattern, given as a sequence of 0s and 1s. In the case of a PRBS the pattern variable has field .PRBSGens. This field is a row vector of positive integers indicating the coefficients in the generator polynomial. For a specified data sequence the pattern variable has field .Values, which is a row vector of logical values, 0s and 1s. The other acceptable form for the pattern variable is an empty variable, which is used if the data sequence is not known. AlignTrib then does not attempt to synchronize to the data pattern.

The default behavior with a specified data pattern is that the whole of the pattern is used to synchronize with the input data pattern (a component of pSym). When the specified data pattern is large, i.e. the .Values field is long, and the input data pattern (pSym) is also large, synchronization may be a slow process. The pattern variable may have an optional field .SyncFrameEnd which causes only a portion of the specified data pattern to be used for synchronization, elements .Values(1:SyncFrameEnd). Typically 100 elements is sufficient to avoid false synchronization, and may considerably speed up execution.

The shortened synchronization frame is only used when the number of bits in the input pattern (the length of pSym) is greater than the length of the specified data pattern. The .SyncFrameEnd field has no effect with a PRBS pattern, since data synchronization with a PRBS is always fast.

AlignTrib processes the data patterns in order according to the modulation format, starting with PattXReIn. For each pattern it tries to match the given data pattern with the available tributaries of pSym. For example, consider the polarization multiplexed QPSK format (SigType = 4):

- The four tributaries are compared with PattXReIn:
  - If one tributary is found to match, then assignment of PattXReIn is complete.
  - If more than one tributary matches, the matched tributaries are compared in time, and the earliest (shortest delay) is assigned to PattXReIn.
  - If no tributaries are found to synchronize with PattXReIn, then that pattern is left unassigned.
- Next PattXImIn is considered. If PattXReIn was synchronized successfully, then only one tributary is tested for PattXImIn, the tributary having the same SOP as the one assigned to PattXReIn; otherwise all four tributaries are tested.
- After PattXImIn has been matched (or found not to match), PattYReIn and then PattYImIn are similarly tested.

The use of delay as a secondary condition to distinguish between tributaries means that AlignTribS will work with transmission experiments where a single data pattern generator is used, and is split several ways with different delays. The delay search is performed only over a limited range of 1000 bits in the case of PRBS patterns, so this method of distinguishing tributaries will not usually work when separate data pattern generators are used programmed with the same PRBS.

The results of synchronization on the different tributaries are reported in the output parameters PattXReOut, PattXImOut, PattYReOut and PattYImOut. The .PRBSGens or .Values field of the input pattern variable is copied to each corresponding output pattern variable. Additional fields are set to the output variable giving the results of the synchronization attempt. A field .Synchronised is set to 1 or 0 indicating whether synchronization was successful. If a PRBS was synchronized a field .Seed gives the PRBS seed, a row vector of logical values whose length is the PRBS generator polynomial length.

If a specified data pattern was synchronized then a field .Start is set to the integer indicating the position in the .Values sequence corresponding to the first element in pSym. The .t0 and .dt fields of pSym are also copied to the output pattern variable. A field .DataPolarity is a logical scalar indicating the polarity of the match. If an input data pattern is not synchronized with any tributaries then the .Seed or .Start fields are set to random values.

To synchronize one data sequence DataIn to one pattern PattIn, call AlignTribS in the following way:

```
[RotM, PattOut, Junk, Junk, Junk, BoundValsOut, AlertsOut] =
  AlignTribS(DataIn, 5, PattIn, [], [], [], BoundValsIn, AlertsIn)
```

In this mode the data polarity should be calculated from both PattOut.DataPolarity and RotM actual data polarity = xor(PattOut.DataPolarity, ~sign(RotM))

### Block processing

AlignTribS attempts to synchronize tributaries only on the first block. For subsequent blocks the output parameters are copies of the first block.

## ApplyPhase

[Y, BoundValsOut, AlertsOut] = ApplyPhase(X, Theta, BoundValsIn, AlertsIn)

- X – single or dual polarization parameter vs. time:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – 1xN or 2xN array of complex values
- Theta – phase to be applied:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – row vector of phase values (radians)
  - .CentFreq – frequency offset between local oscillator and optical signal
- BoundValsIn – structure of boundary values from previous block
- AlertsIn – structure of alerts accumulated before executing function
- Y – result of applying phase adjustment to X
- BoundValsOut – structure of boundary values to be passed to next block
- AlertsOut – structure of alerts including any alerts raised during execution of function

This function multiplies X.Values by a phase factor to give Y.Values. X may be a single or dual polarization representation.

There is no requirement that Theta has the same time grid as X. If they have different time grids then the values of Theta are interpolated to align with the time grid of X. This means that Theta may be derived for symbol center times, for example, and then applied to a waveform having a finer grid.

The actual values of phase that are used take into account the heterodyne frequency CentFreq, that is,  $2 \cdot \pi \cdot \text{Theta} \cdot \text{CentFreq} \cdot (0:\text{number of symbols}-1) + \text{Theta} \cdot \text{Values}$

## ClockRetime

[p, BoundValsOut, AlertsOut] =  
ClockRetime(V, ChDelay, pHyb, Clock, BoundValsIn, AlertsIn)

- V – 1x4 array of structures, each containing waveform (voltage values) from scope
- ChDelay – 4-element vector of time delays between scope channels, usually obtained by separate calibration

- pHyb – 2x4 array (4 column vectors) containing characteristic Jones vectors of optical hybrid ports, usually obtained by separate calibration
- Clock – structure having fields (and may have more fields) defining the output time grid:
  - .t0 – time of first symbol
  - .dt – symbol duration BoundValsIn – structure of boundary values from previous block
- AlertsIn – structure of alerts accumulated before executing function
- p – dual polarization representation of optical signal constructed from scope records V, retimed to align with Clock
  - .t0 – time of first symbol
  - .dt – symbol duration
- .Values – 2xN array (row of Jones vectors) of symbol center signal values
- BoundValsOut – structure of boundary values to be passed to next block
- AlertsOut – structure of alerts including any alerts raised during execution of function

ClockRetime forms an output parameter p, representing a dual-polarization signal vs. time, from four oscilloscope waveforms V. The output p is retimed to be aligned with the timing grid specified by Clock. The function is executed in two steps:

1. The scope waveforms are adjusted for the known relative delays ChDelay between the four scope channels and retimed to be aligned with Clock.
2. The dual polarization signal (a Jones vector vs. time) is computed, given the known relative phase and polarizations of the optical hybrid pHyb.

## DiffDetection

[v,BoundValsOut,AlertsOut] =  
DiffDetection(p,SigType,Delay,CentFreq,BalancedDiffDetection,BoundValsIn,  
AlertsIn)

- p – single or dual polarization parameter vs. time:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – 1xN or 2xN array of complex values
- SigType – integer value indicating signal modulation format
- Delay – positive real value, interferometer delay

- CentFreq – real value, optical frequency of signal compared to multiple of  $1/\text{Delay}$  BalancedDiffDetection – 0 = single-ended detection; 1 = balanced detection
- BoundValsIn – structure of boundary values from previous block
- AlertsIn – structure of alerts accumulated before executing function v – structure giving optical power at output of delay discriminator:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values –  $1 \times N$  or  $2 \times N$  array of values (real or complex)
- BoundValsOut – structure of boundary values to be passed to next block
- AlertsOut – structure of alerts including any alerts raised during execution of function

DiffDetection simulates the insertion of an optical passive delay discriminator into the path of the signal. This mode of optical detection does not use coherent detection, but is a form of self-homodyne detection. It is described in [1] for BPSK modulated signals and [2] for QPSK.

Output parameter v records the optical power at the output of the delay discriminator. For BPSK modulation formats (SigType = 1,3) the delay discriminator has the form of [1], while for QPSK it is as described in [2]. For QPSK formats the result is given as complex numbers, where each value refers to power in one arm + i \* power in other (quadrature) arm

The optical power in one output arm (single-ended detection) is reported if BalancedDiffDetection = 0, or the difference in powers between two balanced outputs when BalancedDiffDetection = 1. Input parameter p may be a single-polarization or dual-polarization representation. Output v is always a single-polarization variable. The delay within the delay discriminator is of duration Delay. The standard delay discriminator configuration uses one symbol of delay.

CentFreq sets the offset between the center of the signal spectrum and a comb of frequencies at multiples of  $1/\text{Delay}$ . For a real implementation of a delay discriminator to work well the signal optical frequency should be a multiple of  $1/\text{Delay}$ . This is achieved either by tuning the signal laser or adjusting the delay over a small range. The variable CentFreq allows a non-optimal delay discriminator configuration to be simulated. Use CentFreq = 0 to obtain the standard operating condition.

## References

K. Yonenaga, S. Aisawa, N. Takachio, K. Iwashita, "Reduction of four-wave mixing induced penalty in unequally spaced WDM transmission system by using optical DPSK," IEE Electron. Lett., vol. 32, no. 23, p. 1218-1219, 1996.



R.A. Griffin, A.C. Carter, "Optical differential quadrature phase-shift key (oDQPSK) for high capacity optical transmission," OFC 2002 conference, Anaheim, US, paper WX6, 2002.

## EstimateClock

```
function [Clock,BoundValsOut,AlertsOut] =
EstimateClock(V,ChDelay,pHyb,FreqWindow,NonlinFunc,BoundValsIn,AlertsIn)
```

V – 1x4 array of structures, each containing waveform (voltage values) from scope

ChDelay – 4-element vector of time delays between scope channels, usually obtained by separate calibration

pHyb – 2x4 array (4 column vectors) containing characteristic Jones vectors of optical hybrid ports, usually obtained by separate calibration

FreqWindow – range of frequency in which symbol clock frequency may be located

.Low – low frequency limit

.High – high frequency limit

NonlinFunc – string containing instruction for nonlinear function used in clock recovery

BoundValsIn – structure of boundary values from previous block AlertsIn – structure of alerts accumulated before executing function

Clock – structure containing estimated symbol clock:

.t0 – time of first symbol center after  $t = 0$

.dt – symbol period

BoundValsOut – structure of boundary values to be passed to next block

AlertsOut – structure of alerts including any alerts raised during execution of function

EstimateClock determines the symbol clock frequency of a digital data-carrying optical signal based on oscilloscope waveform records. The scope sampling rate may be arbitrary (having no integer relationship) compared to the symbol rate.

The clock recovery process has three steps:

1. The oscilloscope waveforms are adjusted for the known relative delays ChDelay between the four scope channels. The delay-adjusted values are integrated into a dual polarization representation (a Jones vector) vs. time, given the known relative phase and polarizations of the optical hybrid pHyb. Often there is no content at the symbol clock frequency in this dual polarization signal.
2. A nonlinear function is applied to the signal to generate some clock content. The nonlinear function is defined in the input variable NonlinFunc.
3. The application sweeps the frequency within the given window, FreqWindow, to search for a spike at the clock frequency.

**Nonlinear function**

The nonlinear function is set by the user via the input parameter NonlinFunc. This parameter is a string which should contain a MATLAB function evaluating variable Y in terms of variable X. If NonlinFunc is an empty string then the function defaults to

$$Y = \text{abs}(X(1,:)).^2 + \text{abs}(X(2,:)).^2$$

The same function is used if NonlinFunc cannot be evaluated, or if it does not produce a valid Y, and an alert is generated.

The default function works in most cases with a non-return to zero (NRZ) signal. If the signal is a return-to-zero (RZ) signal then it may be better to use

$$Y = \text{sqrt}(\text{abs}(X(1,:)).^2 + \text{abs}(X(2,:)).^2)$$

If the edges of the signal have excessive ringing then that may cause the clock phase reported by EstimateClock to be offset compared to the true center of the symbol. In that case a limiting function can be applied to reduce the impact of the ringing, such as this one

$$Y = \min(\text{abs}(X(1,:)).^2 + \text{abs}(X(2,:)).^2, 0.2 * \text{mean}(\text{abs}(X(1,:)).^2 + \text{abs}(X(2,:)).^2))$$

**Frequency window**

The symbol clock frequency output lies between FreqWindow.Low and FreqWindow.High. If FreqWindow.Low = FreqWindow.High then the frequency search step is omitted, and the clock phase only is calculated at the given frequency. Also the function takes less time to execute.

**Block processing**

The three step process outlined above is followed for the first block to be processed. For subsequent blocks the clock phase is calculated using a fixed clock frequency, which is the same clock frequency as determined by the previous block. The phase of the new block is used together with the phases of earlier blocks to calculate an average clock frequency and corresponding average clock phase, which are reported in output parameter Clock. The reported clock frequency is forced to lie within FreqWindow.Low...FreqWindow.High. Thus, the clock frequency and phase reported by the final block are averages of the whole measurement record, and are more accurate than those reported by earlier blocks.

## EstimatePhase

[Theta, BoundValsOut, AlertsOut] =  
EstimatePhase(zSym, SigType, Alpha, BoundValsIn, AlertsIn)

- zSym – complex electric field values (single polarization) vs. time:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – 1xN array (row vector of complex values), symbol center signal values
- SigType – integer value indicating signal modulation format
- Alpha – real number in interval 0..1 determining averaging time of phase estimate
- BoundValsIn – structure of boundary values from previous block
- AlertsIn – structure of alerts accumulated before executing function
- Theta – unwrapped estimated phase of input optical signal; extends from  $\infty$  to  $+\infty$ :
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – row vector of phase values (radians) at symbol centers
  - .CentFreq – frequency offset between local oscillator and optical signal
- BoundValsOut – structure of boundary values to be passed to next block
- AlertsOut – structure of alerts including any alerts raised during execution of function

The EstimatePhase function estimates the phase of the optical signal. The algorithm used is known to be close to the optimal estimate of the phase [1]. The algorithm first determines the heterodyne frequency offset and then estimates the phase. The phase reported in the .Values field is after the frequency offset has been subtracted. Thus the actual phase of the signal with respect to LO is:

$$2*\pi*Theta.CentFreq*(0: \text{number of symbols}-1)*Theta.dt + Theta.Values$$

To calculate the phase difference of two different Theta structures the center frequency must be taken into account.

### Block processing

The second and subsequent blocks report the value of .CentFreq calculated by the first block. The reported .CentFreq value is not averaged by block processing. To calculate a better average of the heterodyne frequency after many blocks use:

$\text{Theta.CentFreq} + (\text{Theta.Values}(\text{end}) - \text{Theta.Values}(1)) / 2 / \pi / \text{number of symbols} / \text{Theta.dt}$

Although it reports the same value for .CentFreq every block, the function internally estimates a new heterodyne offset frequency for each block. This means that it is able to track a slowly varying frequency change. The block size must be smaller than the time taken for the frequency to shift in order to track it accurately. The phase calculate internally is adjusted for the reported .CentFreq (calculated in the first block), so that the reported Theta.CentFreq and Theta.Values are consistent. The user can differentiate Theta.Values to see the changing frequency, perhaps with a large step size to effectively average the result.

- References** M.G. Taylor, "Phase Estimation Methods for Optical Coherent Detection Using Digital Signal Processing," IEEE J. Lightwave Technol., vol. 27, no. 7, p. 901-914, 2009.

## EstimateSOP

[RotM, BoundValsOut, AlertsOut] = EstimateSOP(pSym, SigType, BoundValsIn, AlertsIn)

- pSym – dual-polarization description of optical signal, at symbol center times:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – 2xN array (row of Jones vectors), symbol center signal values
- SigType – integer value indicating signal modulation format
- BoundValsIn – structure of boundary values from previous block
- AlertsIn – structure of alerts accumulated before executing function
- RotM – 2x2 matrix (Jones matrix) containing axes of polarization states of signal tributaries BoundValsOut – structure of boundary values to be passed to next block
- AlertsOut – structure of alerts including any alerts raised during execution of function

EstimateSOP reports the state of polarization (SOP) of the tributaries in the optical signal. The result is provided in the form of an orthogonal (rotation) matrix RotM. For a polarization multiplexed signal the first column of RotM is the SOP of the first tributary, and the second column the SOP of the second tributary. For a single tributary signal, the first column is the SOP of the tributary, and the second column is orthogonal to it. The signal is transformed into its basis set (the tributaries horizontal vertical polarizations) by multiplying by the inverse of RotM.

The function employs a different algorithm for each modulation format. The standard SigType values are supported (1 to 5), as well as 0. With SigType = 0 the function finds an unmodulated tributary from a mix of other bipolar-modulated channels and noise. This mode of the function is useful if the modulated data on a tributary is known. Multiplying the optical signal by the conjugate of the known modulation converts that tributary into an unmodulated tributary, and EstimatePhase can be applied with SigType = 0 to obtain the SOP of that tributary alone.

### Block processing

The block processing part of the algorithm assumes that the tributary states of polarization are unchanged from block to block. The SOP estimates are improved by more averaging from one block to the next. To commence a new estimate with a new block, for example to track a slowly varying SOP, pass an empty variable as BoundValsIn for each block.

## MaskCount

[MaskViolations] =

EVMcalc(zSym,SeqRe,SeqIm,PatReSyn,PatImSyn,MaskThreshold,SigType)

- zSym – complex electric field values (single polarization) vs. time with fields:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – (1xN) array (row vector of complex values), symbol center signal values
- SeqRe – bit sequence corresponding to in-phase signal
- SeqIm – bit sequence corresponding to quadrature signal
- PatReSyn – binary (0 or 1) indicating if the in-phase signal is synchronized
- PatImSyn – binary (0 or 1) indicating if the quadrature signal is synchronized
- MaskThreshold – threshold defining the radius of the mask; is used to count mask violations
- SigType – integer indicating the signal modulation format

MaskCount calculates the instantaneous error vector magnitude and counts the number of measured mask violations. The instantaneous EVM is defined as the magnitude of the difference between the ideal symbol location (calculated on a block by block basis) and the measured symbol location (zXSym.Values) in root watts. The ideal symbol location is calculated by projecting the measured symbol values onto the corresponding I/Q radials defining their ideal locations. The average magnitude of this projection defines the value of ConstPtMag. This value scales the unit magnitude ideal constellation to calculate the ideal symbol locations.

The function returns the number of mask violations corresponding to each symbol. A mask violation occurs when the instantaneous EVM exceeds a set threshold. The function returns MaskViolations as a vector with one entry for each constellation point. The value of the threshold used to count the number of mask violations can be set from the Engine Command Window in OUI as (using 60% as an example):

MaskThreshold = 0.6;

The number of mask violations is reported in the variables zXSymUI.MaskViolations and zYSymUI.MaskViolations

## GenPattern

[Seq,BoundValsOut,AlertsOut] =  
GenPattern(Patt,NumBitsVar,BoundValsIn,AlertsIn)

- Patt – data pattern
- NumBitsVar – parameter indicating number of bits to generate
- BoundValsIn – structure of boundary values from previous block
- AlertsIn – structure of alerts accumulated before executing function Seq – sequence of logical values
- BoundValsOut – structure of boundary values to be passed to next block
- AlertsOut – structure of alerts including any alerts raised during execution of function

GenPattern generates a sequence of logical values, 0s and 1s, given an exact data pattern. The exact pattern specifies not only the form of the sequence but also the place it starts and the data polarity. The data pattern specified by Patt may be a pseudo-random bit sequence (PRBS) or a specified sequence. In the case of PRBS Patt has these fields:

- .PRBSGens – row vector of positive integers; indicates generator polynomial coefficients
- .Seed – seed of PRBS
- .DataPolarity – 0 = inverted; 1= true

In the case of a specified data pattern Patt has these fields:

- .Values – row vector of logical values
- .Start – position in .Values of first element of output sequence
- .DataPolarity – 0 = inverted; 1= true

Patt may also have fields .t0 and .dt.

NumBitsVar takes on one of two forms. If it is a positive integer value then that is used as the number of bits to generate in output parameter Seq. The time field Patt.t0 is ignored (if it exists). The output sequence starts with Patt.Seed in the case of a PRBS, or Seq starts at position Patt.Start in the case of a specified data pattern.

Alternatively, NumBitsVar may have fields' .t0 and .dt, in which cases the difference between NumBitsVar.t0 and Patt.t0 is used, together with Patt.dt, to determine the start point of Seq. The offset in time may be positive or negative. It is acceptable for NumBitsVar to have other fields, for example a .Values field, because they are ignored.

Note that Seq is not a structure. It is a variable containing a row vector of logical values, and does not have .t0 and .dt fields.

## Jones2Stokes

$Y = \text{Jones2Stokes}(X)$

- X – 2xN array, row of Jones vectors
- Y – 3xN array, row of Stokes vectors

This function converts a row of Jones vectors into a row of Stokes vectors.

## JonesOrth

$Y = \text{JonesOrth}(X)$

- X – 2xN array, row of Jones vectors
- Y – 2xN array, row of Jones vectors, each orthogonal to the corresponding Jones vector in X

This function converts a row of Jones vectors into a row of Jones vectors representing the orthogonal state of polarization.

In general there are many Jones vectors that are orthogonal to a given Jones vector. The many vectors can have different absolute values and also they can be related to one another by a phase factor. The output Y of JonesOrth is chosen to have the same absolute value as input X, and the first element of Y is real.

## LaserSpectrum

[Lspectrum] = LaserSpectrum(ThetaSym, RBW)

- ThetaSym – symbol rate phase estimates:
  - .t0 – time of first symbol
  - .dt – symbol period
  - .Values – (1xN) array (row vector of complex values) of signal values
- RBW – desired resolution bandwidth

The function LaserSpectrum estimates the power spectral density of the combined laser waveform in units of dBc. The function LaserSpectrum takes ThetaSym, the estimated phase of the signal at the sampling rate as input, and, defines the frequency centered laser waveform as 'LaserWaveForm = exp(j\*ThetaSym.Values)'. This waveform is then scaled by a hamming window, and the power spectral density of the waveform is estimated as discrete Fourier transform of this signal. Note - the output produced by LaserSpectrum does not represent the behavior of the transmit laser alone; instead, it displays the convolved power spectral density of the local oscillator and the transmit laser.

The resolution bandwidth can be set from the Engine Command window with the following command (using 20 MHz as an example): ThetaRBW = 20e6;

## QDecTh

[QDecTh,Rail0,Rail1,BoundValsOut,AlertsOut] =  
QDecTh(S,Seq,MinStdDevFit,BoundValsIn,AlertsIn)

- S – symbol center values of signal of one tributary:
  - .t0 – time of first symbol
  - .dt – symbol duration
  - .Values – 1xN row vector of real values
- Seq – row vector of logical values, actual data sequence for symbols of S
- MinStdDevFit – smallest number of standard deviations of noise to include in straight line fit
- BoundValsIn – structure of boundary values from previous block
- AlertsIn – structure of alerts accumulated before executing function
- QDecTh – Q-factor of signal (linear units)



- Rail0 – structure describing straight line fit to 0 rail:
  - .S – row vector of signal values used for fit, at which decision threshold was set
  - .Q – inverse error function of bit error rate at decision threshold values S
  - .Mean – mean signal of 0 rail, based on straight line fit
  - .Sigma – standard deviation of noise on 0 rail, based on straight line fit
- Rail1 – structure describing straight line fit to 1 rail:
  - .S – row vector of signal values used for fit, at which decision threshold was set
  - .Q – inverse error function of bit error rate at decision threshold values S
  - .Mean – mean signal of 1 rail, based on straight line fit
  - .Sigma – standard deviation of noise on 1 rail, based on straight line fit
- BoundValsOut – structure of boundary values to be passed to next block
- AlertsOut – structure of alerts including any alerts raised during execution of function

This function uses the decision threshold method [1] to estimate the Q-factor of a component of the optical signal. The method is useful because it quickly gives an accurate estimate of Q-factor (the output signal-to-noise ratio) even if there are no bit errors, or if it would take a long time to wait for a sufficient number of bit errors.

The actual signal vs. time, including noise and distortions, is provided as input S, together with the true data sequence Seq it corresponds to. The function varies the decision threshold and counts the number of bit errors at each decision threshold, then fits a straight line to the points of (decision threshold, inverse error function of bit error rate). The maximum decision threshold used (farthest away from the middle of the rail) is just before the number of errors counted is too small to be statistically significant.

The minimum decision threshold used in the straight line fit (closest to the middle of the rail) is given by MinStdDevFit. Negative values of MinStdDevFit are acceptable. The value for MinStdDevFit is typically chosen by plotting the rails from the middle of the rail first (MinStdDevFit = 0), and then clipping to the region where there is no curvature in the (.S,.Q) points.

## References

N.S. Bergano, F.W. Kerfoot, C.R. Davidson, "Margin measurements in optical amplifier systems," IEEE Phot. Tech. Lett., vol. 5, no. 3, p. 304-306, 1993.

## zSpectrum

[zSpectrum] = zSpectrum(z, RBW, vdt)

- z – complex oversampled signal (usually zX or zY) with fields:
  - .t0 – time of first element
  - .dt – sampling period
  - .Values – (1xN) array (row vector of complex values) of signal values
- RBW – desired resolution bandwidth
- vdt – scope sampling rate, usually Vblock(1).dt.

The function zSpectrum.m takes z, a signal, and a desired resolution bandwidth, RBW, to calculate the empirical power spectral density (PSD) in units of dBm per resolution bandwidth. The function calculates the spectrum of the signal as follows:

1. The signal is downsampled by an integer rate to the slowest rate faster than the sampling rate of the digital oscilloscope to avoid unnecessary processing.
2. The resolution bandwidth is used to calculate the length of discrete Fourier transform, and the signal is divided into blocks of this length.
3. The discrete Fourier transform of each block is calculated, and the squared magnitude is averaged across all blocks (this results in smoothing of the PSD).
4. The values are returned for plotting.

The resolution bandwidth can be set from the Engine Command window with the following command (using 20 MHz as an example): zRBW = 20e6;

---

## Appendix A: MATLAB variables used by core processing

- Vblock – voltage vs. time on four scope channels
- SigType – integer value indicating the modulation format of the signal
- 1 – single polarization binary phase shift keying (BPSK)
- 2 – single polarization quadrature phase shift keying (QPSK)
- 3 – dual polarization BPSK
- 4 – dual polarization QPSK
- 5 – on-off keying (OOK)
- 6 – three state OOK, where the on state can take on a field of +1 or -1
- 7 – single polarization 16-state quadrature amplitude modulation (16-QAM)
- 8 – dual polarization 16-QAM
- 9 – single polarization 64-QAM
- 10 – dual polarization 64-QAM
- 11 – single polarization offset QPSK
- 12 – dual polarization offset QPSK
- 13 – offset polarization QPSK
- 14 – single polarization 8-ary phase shift keying (8-PSK)
- 15 – dual polarization 8-PSK
- 16 – single polarization 8-QAM
- 17 – dual polarization 8-QAM
- PattXRe, PattXIm, PattYRe, PattYIm – define data patterns on tributaries, used to identify bit errors

The variables that are calculated by CoreProcessing include:

- zXSym, zYSym – complex electric field at symbol center times on X and Y polarizations; displayed as blue dots on constellation diagram in OUI
- zX, zY – complex electric field, continuous with time; displayed as eye diagrams
- wSym – optical power (electric field squared) at symbol centers
- w – optical power, continuous with time
- NumErrs – bit error count on each tributary, together with total number of bits and sync loss status



## Appendix B: Alerts

Alerts may appear in the Alerts section of the main ribbon, accompanied by a change in the “Alerts” text as notification.

**Table 15: Alert code descriptions**

Code	Calling function	Description
1	EngineCommandPre	Local oscillator (LO) frequency not set. Set the LO frequency automatically by opening the Laser Receiver Control Panel, or manually under the Setup tab in OUI.
2	EngineCommandPre	Local oscillator (LO) power not set. Set the LO power automatically by opening the Laser Receiver Control Panel, or manually under the Setup tab in OUI.
3	EngineCommandPre	Channel delays are not specified. Set the channel delays under the Calibrate tab in OUI.
4	GetpHybCalib	Coherent receiver calibration not set. Using default pHyb. Set the receiver calibration by placing the supplied calibration file, pHybCalib.mat, in the 'ExecFiles' directory.
5	CoreProcessing	Equalization not applied. Equalization filter coefficient file not found or scope sampling rate unsupported. Define the equalization filter coefficients by placing the supplied file, EqFiltCoef.mat, in the ExecFiles folder.
10	CoreProcessing	DC Calibration may be needed. Click DC Calibration on Calibration tab.
20	CoreProcessing	Cannot execute 2nd SOP estimate because one or more tribs is not synchronised.
21	CoreProcessing	Cannot execute 2nd phase estimate because one or more tribs is not synchronised.
22	CoreProcessing	2nd phase estimate is not recommended when Alpha < 0.75.
30	CoreProcessing	PMD cannot be measured using reference because no reference is stored. Applying non-reference method instead.
201	EstimatePhase	Cannot evaluate NonlinFunc or does not give usable Y; used $Y = \text{abs}(X).^2$ instead.
202	EstimatePhase	Cannot evaluate NonlinFunc or does not give usable Y; used $Y = \text{abs}(X(1,:)).^2 + \text{abs}(X(2,:)).^2$ instead.
203	EstimatePhase	Power in clock component is low. Clock frequency may be incorrect.
204	EstimatePhase	Excessive clock jitter or clock frequency lies outside given window.
205	EstimatePhase	Size of block or record too small to produce sufficient number of symbols using estimated clock frequency. Returning higher clock frequency that is incorrect.

Table 15: Alert code descriptions (cont.)

Code	Calling function	Description
206	EstimatePhase	Size of block or record too small to produce sufficient number of symbols given FreqWindow.High. Returning clock frequency outside given window.
207	EstimatePhase	Clock frequency may be incorrect because of aliasing. Specify narrower frequency window.
300	EstimatePhase	Alpha has changed from previous block. Original value being used.
301	EstimatePhase	zSym does not resemble a QAM signal. Cannot estimate phase.
302	EstimatePhase	Rise time of phase smoothing filter longer than block time. Estimated phase may not be accurate.
303	EstimatePhase	zSym does not resemble an offset QPSK signal. Cannot estimate phase.
310	EstimateSOP	Cannot calculate Jones matrix because pSym does not resemble a dual polarisation signal.
400	AlignTribes	Unable to sync trib to pattern. Returning random data pattern for %s.
410	GenPattern	Clock frequency for Patt is different from NumBitsVar. Using Patt clock frequency.
411	GenPattern	Generating random data values because length(NumBitsVar.Values) less than PRBS length.
412	GenPattern	Generating random data values because NumBitsVar less than PRBS length.
413	GenPattern	Patt.t0 has a different clock phase from BoundValsIn.Patt.t0. Using BoundValsIn.Patt.t0.
414	GenPattern	Patt.t0 has a different clock phase from NumBitsVar.t0. Rounding number of symbols to nearest whole number.
420	AlignTribes	Data pattern synchronization may be incorrect because %s.SyncFrameEnd <50.
421	AlignTribes	Number of bits too small to recover PRBS. Use longer block, or shorter PRBS in %s.
422	AlignTribes	Cannot recover PRBS because number of bits smaller than length of PRBS in %s.
430	DiffDetection	p.Values too short to produce sufficient number of output values given Delay. Using smaller Delay = %d instead.
440	QDecTh	Seq must contain at least ten 0s and ten 1s.
441	QDecTh	Not enough points available to fit valid straight line to 0 rail.
442	QDecTh	Not enough points available to fit valid straight line to 1 rail.
902	EngineCommandPos	One or more required parameters were not calculated by CoreProcessing. Variables needed to calculate summary parameters were not calculated. CoreProcessing may be commented out of the MATLAB Engine window.

# Appendix C: Calibration and adjustment (RT oscilloscope)

## Calibration and adjustment (RT)

The OM4000 receiver requires the following calibration before taking measurements:

- DC calibration (to compensate for any offsets in the photodiode outputs)
- Delay adjustment (channel to channel skew in the scope connections)
- Hybrid calibration (correction for cross-talk and phase error in the hybrid)
- Laser linewidth factor (choosing the correct filter for phase recovery) (See page 34.)
- Receiver equalization

### DC calibration (RT)

Although the OM4000D Series units use balanced detection, there is usually some small offset voltage present at the signal outputs. This offset voltage depends on the total optical power input to the system and so can change. The offset is small enough that only large changes to the reference or optical input power substantially affect the computation.

DC calibration determines the DC levels in the receiver's photodiodes, and subtracts this off within the analysis. This can be done as often as needed or desired. If there is uncompensated dc offset in the system, this will be evident by a smearing out of the constellation point groups. If the offset is large enough, the point groups will begin to look like donuts. Perform a dc calibration whenever there is any question.

To perform a DC calibration, click the DC Calibration button in the Calibrate tab.

### Delay adjustment (system deskew) RT

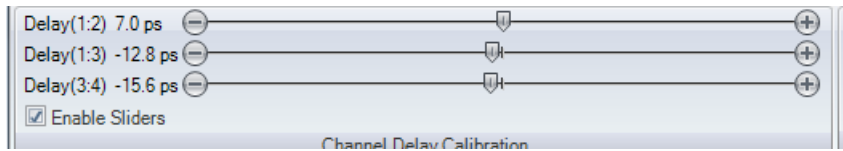
---

**NOTE.** *Initial delay adjustment should be done by trained personnel. This section is provided for experienced users. Delay adjustment should be done during installation and should not require attention unless the scope is removed and/or reconnected.*

---

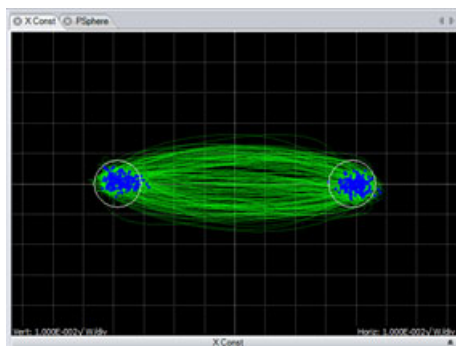
Delay adjustment among the four electrical channels of the system, including the OM4000 receiver and oscilloscope, is done through the Channel Delay Calibration section of the Calibrate ribbon. Once this is accomplished for a specific oscilloscope installation, it should not need further attention, and should be left untouched. If the receiver/oscilloscope interface is altered (for example, installing new cables or shifting of the instrument), delay calibration may need to be performed again.

Access to the delay sliders with the **Enable Slider** control in the Calibrate tab.

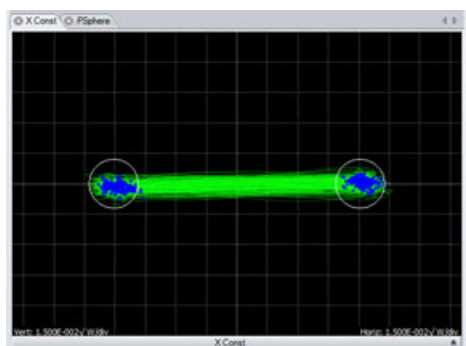


Each slider indicates the relative delay for the given channel pair. The corresponding waveforms will be shifted to account for this skew. Delay (1:2) means the delay of RX channel 2 (the X-Q channel) relative to RX channel 1 (the X-I channel).

Use the sliders to get the best possible eye diagram and constellation diagram. Improper skew will cause horizontal eye closure and filling in of the constellation diagram. If the I and Q channels have unequal delay, there will be a phase offset proportional to the difference frequency between the reference and signal laser oscillation frequencies. This phase offset will turn a straight line on the phase diagram into a circle or a portion of a circle.



So, for verification, it is best to use a known Mach-Zehnder modulator to generate a BPSK optical input. As the input signal polarization is adjusted, the I-Q diagram should always be a straight line.



Programmers note: The ChDelay variable in the MATLAB workspace is defined as follows:

- ChDelay(1): zero.
- ChDelay(2): delay of OM4000 RX channel X-Q relative to X-I



- ChDelay(3): delay of OM4000 RX channel Y-I relative to X-I
- ChDelay(4): delay of OM4000 RX channel Y-Q relative to X-I

This will also include connecting cable and digitizer delays if the digitizer/cable combination has not been de-skewed separately.

**Manual ChDelay determination with a 1-pol BPSK input signal.** Once you have used the sliders to set the delays at least close to their actual values, you can use the shape of the phase-diagram curves to fine tune as described here. It is best to do this one polarization at a time using the following steps:

1. Delete everything from the MATLAB Engine Window except for CoreProcessingCommands or CoreProcessing.
2. Perform a DC calibration (See page 111, *DC calibration (RT)*).
3. Adjust the input polarization so the signal is mostly on Vblock(1) and Vblock(2) which correspond to receiver channels X-I and X-Q. Depending on your receiver connections, this may correspond to different oscilloscope channels. This can be done using a polarization adjustor, or mathematically by putting the following statements before CoreProcessing:

```
Vblock(3).Values = Vblock(3).Values*0.001;
Vblock(4).Values = Vblock(4).Values*0.001;
```

In either case it is good to adjust the fiber somewhat to maximize the signals on RX channels X-I and X-Q.

Now only the top slider will make any difference.

4. Click **Run** to get a repeating constellation and eye-diagram update.
5. Click the + and – to make 0.1-ps adjustments to the top slider until the BPSK constellation as perfectly straight lines connecting the two groups of constellation points as shown in the figure above.
6. Displaying the X-Q eye will show the BPSK signal going into the “wrong” quadrature. When the delay is set properly the X-Q signal shown should only be noise.
7. Shut down RX channels X-I and X-Q by moving the polarization state of the signal or by zeroing it by deleting the lines from step 3 and replacing them with:

```
Vblock(1).Values = Vblock(1).Values*0.001;
Vblock(2).Values = Vblock(2).Values*0.001;
```

now only the bottom slider will matter.

8. Move the last slider using the + and – buttons to get 0.1-ps changes until the constellation looks as it did with channels 1 and 2. Once again use the X-Q eye as a measure of residual error as well as constellation quality.

9. Once this is done get approximately equal signal on both polarizations by deleting everything from the Engine Command Window except for CoreProcessingCommands and adjusting the input polarization as needed.
10. Set the middle slider to achieve minimum signal in the Y-polarization. Since the input is a single-pol signal at this point, nothing should be in the Y constellation or eyes except for noise.
11. Display the Y-I or Y-Q eye diagrams and Y-Constellation to see the signal going into the wrong polarization. This should just be noise when the middle slider is set properly.
12. The delay calibration is done if there is only noise in eye plots except the X-I and only noise in the Y-constellation.
13. Click the check box to hide the sliders to avoid accidental change.
14. Save the MATLAB workspace as Delay.mat for later recovery of the delay data if needed. The delays are now stored in the ChDelay variable.

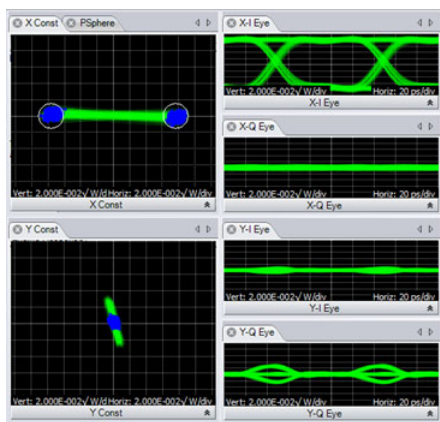


Figure 15: When adjusting the middle slider, watch the Y-Eye and Y-Const to minimize the signal in the Y-polarization

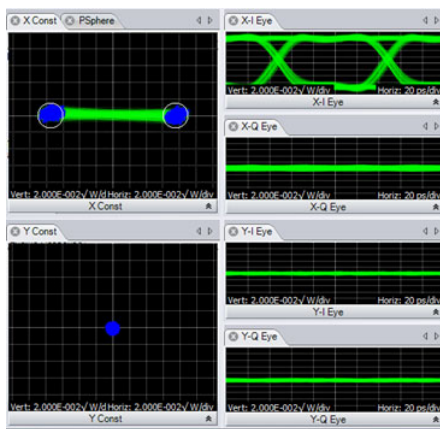
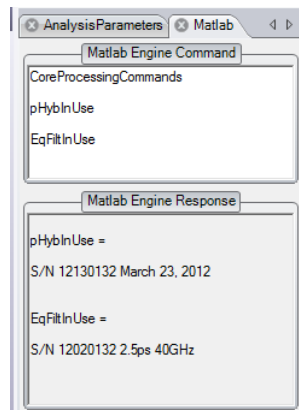


Figure 16: Final channel delay values provide only noise in Y polarization

## Hybrid calibration (RT)

Hybrid calibration is a factory calibration. Imperfections in the OM4000 instrument receiver are corrected using a factory-supplied calibration table. Check the Setup tab for a green indicator to be sure the OUI is successfully retrieving the Reference laser (Local Oscillator) frequency and power which are needed to choose the correction factors from the calibration table. The table is the pHybCalib.mat file in the ExecFiles directory. You can verify you have a valid pHybCalib file by connecting to an oscilloscope, typing pHybInUse in the MATLAB Engine Window, and clicking Single. Similarly, EqFiltInUse shows the equalization filter in use (if any).



The information statement contains the serial number, date of calibration, and other notes about the calibration. If the serial number is correct then you have the proper file.

The following procedure can be used to verify and correct the calibration at a single wavelength using a minimum of external hardware. This procedure assumes that “Laser 1” and Laser 2” can be tuned to the same wavelength. It is easiest if you have a polarization controller before the signal input, but the instructions are written assuming you are moving the fiber around to change the input polarization.

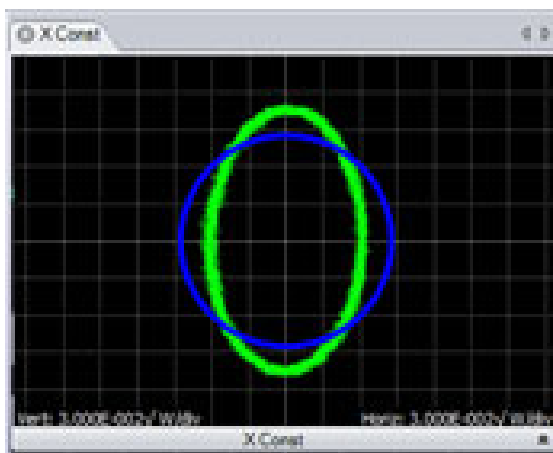
### Hybrid calibration prerequisites (RT)

- System should be deskewed following the procedure above.
- Connect the Reference as usual from Laser 2 to the Reference input with short PM/APC jumper.
- Use a standard SM/APC (not PM) fiber to connect the Laser 1 to the Signal input. Use the LRCF to set the Signal 1 power level to about 10 dBm to start.

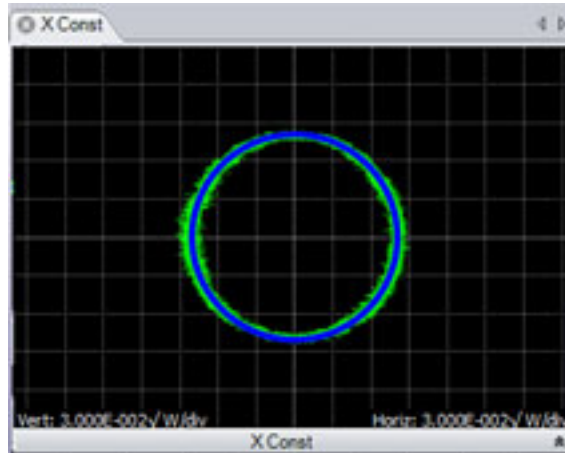
- In the LRCP tool:
  - Turn on both lasers and tune them to the same channel where you will be working.
  - Set the Laser 1 power to get about 100 mV pp.
  - Tune Laser 1 off grid by 100 to 500 MHz
- Click **Run** on the oscilloscope to get a rapidly updating display. You should see 100 to 500 MHz sine waves.
- Move the SM fiber around until you get most signal on RX channels X-I and X-Q (at least 3:1 ratio between X-I and Y-I. This is most easily done with a polarization controller).
- Tape the fiber down so that you continue to get most signal on X-I and X-Q.
- Click **Single** on the oscilloscope.

**View and correct the X-polarization calibration (RT)**

1. Use the Optametra OUI to connect to the oscilloscope. Record length of ~20,000 points recommended in single block (BlockSize > 20,000).
2. Display the MATLAB Engine Window, the X-Constellation Window and the Y-Constellation Window. Close other windows.
3. Put DispCalEllipses in the MATLAB Engine Window of the OUI. Delete everything else in the MATLAB Engine Window.
4. Click **Run** on the Optametra OUI.
5. Observe that the ellipses are displayed on the Constellation plots. Right now only the X-constellation has a signal. The green trace should line up with the blue circle in the X-constellation plot.



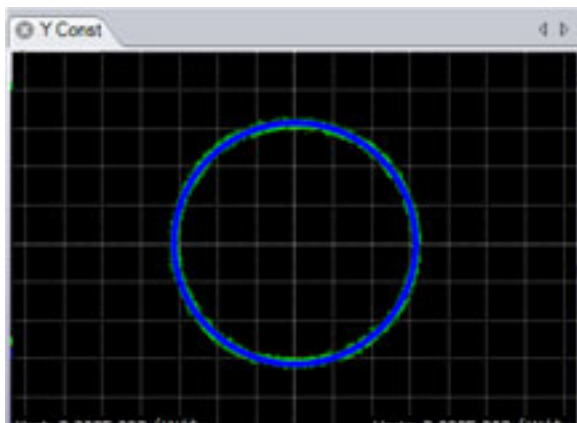
6. If the green trace in X-Constellation is elliptical:
  - Click **Stop**.
  - Enter CorrectX in the separate MATLAB Application Window.
  - Copy and paste the resulting pHyb statement before DispCalEllipses in the MATLAB Engine window in the OUI as shown in the following image.



7. Click **Run**. The green trace should now be circular.

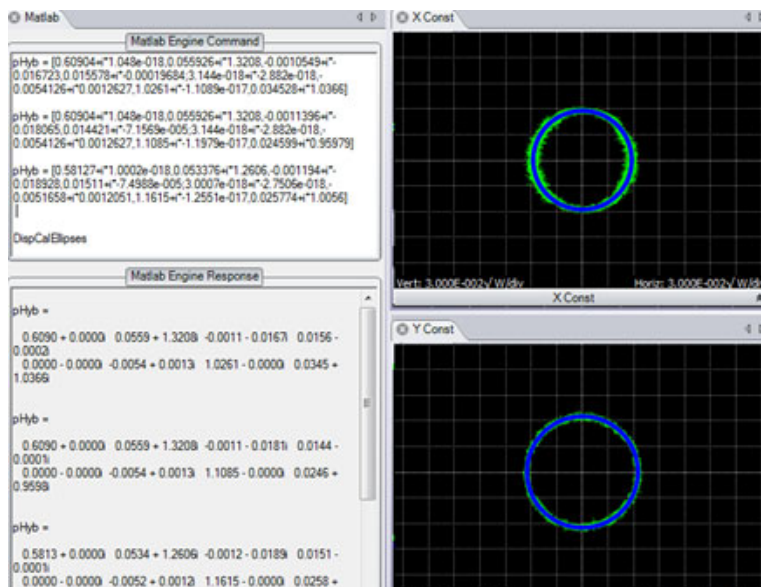
### View and correct the Y-polarization calibration (RT)

1. Move the input fiber to get most of the signal on RX channels Y-I and Y-Q. Tape it down.
2. Click **Single** on the oscilloscope.
3. Click **Run** on the OUI.
4. Observe that the ellipses are displayed on the Constellation plots. Right now only the Y-constellation has signal. The green trace should line up with the blue circle in the Y-constellation plot.
5. If the green trace in Y-Constellation is elliptical:
  - a. Click **Stop**.
  - b. Enter CorrectY in the separate MATLAB Engine Window.
  - c. Copy and paste the resulting pHyb statement before DispCalEllipses, replacing any other pHyb statement.
6. Click **Run**. The green trace should now be circular.



**Correct the relative X-Y gain (RT)**

1. You must complete all of the above steps first including CorrectX and CorrectY.
2. Type CorrectXY in the separate MATLAB Application Window.
3. Copy and paste the resulting pHyb statement before DispCalEllipses, replacing any other pHyb statement.
4. Move the input fiber until there is signal on all four channels.
5. Click Run. The green trace should now be circular in both constellations.



6. The pHyb statement in step 3 is the final output that is corrected for RX X-polarization I-Q gain and phase, RX Y-polarization I-Q gain and phase, and X-Y relative gain.

**Return instrument to normal operation (RT)** Replace the DispCalEllipses statement with CoreProcessing for normal operation. Keep the pHyb statement as it is correcting the calibration.

## Absolute power calibration

As of OM4000 version 1.2.0, OUI can plot signal data on an absolute scale independent of the LO signal strength. This requires absolute scaling of the pHybCalib.mat file, which was not available on all earlier versions of OUI. The following power calibration should be accurate to 15%:

1. Check the absolute scale by connecting a CW signal of known power (no modulation) with sufficient power to fill most of the oscilloscope display.
2. Be sure the OUI and LRCP are connected by checking for the green square on the SetUp Tab.
3. Do a DC calibration. The OUI should display one group of symbols in the X constellation. The distance from the center of that group to the origin is the signal strength in root-Watts. Square this value and compare to the known value in Watts.
4. To use the built in Magnitude measurement, choose BPSK signal type and a clock frequency equal to twice the offset between the signal and LO frequencies. This will display two clusters of points and the Magnitude measurement will provide the average signal strength.

## Laser linewidth factor

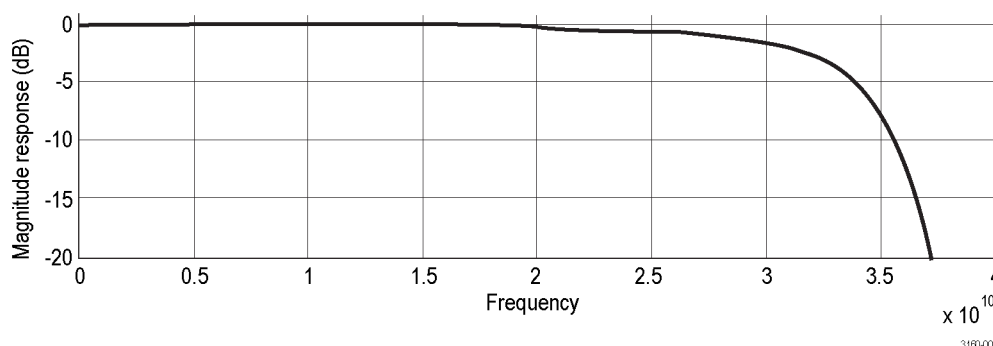
Refer to discussion on modifying the default value of Alpha. (See page 34.)

## Receiver equalization

Receiver Equalization is a factory calibration. Digital equalization is applied to the four channels of the OM4000 instrument to account for the non-ideal frequency dependent response introduced by the coherent optical receiver and the receiver radio frequency front end. Depending on the sampling rate of the oscilloscope and the bandwidth of the OM4000 instrument, digital equalization is applied so that the combined effect of the receiver and the digital equalization filter meet a specified reference response:

Oscilloscope sampling rate	Reference magnitude response	reference phase
$\leq 2x$ the bandwidth of the OM4000 (BW)	Flat	Linear
$> 2x$ the bandwidth of the OM4000 (BW)	4th order digital (bilinear) Bessel filter with 3 dB cutoff at BW + 2 GHz	Linear

Equalization is specific to the sampling rate of the oscilloscope. As an example, if the bandwidth of the OM4000 is 30 GHz and the sampling rate of the oscilloscope is 80 Gs/s, the combined effect of the OM4000 receiver front end response and the digital equalization filter will be that of a 4th order digital Bessel filter with 3 dB cutoff at 32 GHz, as shown in the following graph.



The digital filter is applied directly to the MATLAB workspace variables *Vblock(1)*. *Values* through *Vblock(4)*. *Values* using a 100 tap FIR filter.



Equalization provides a challenge when working in ET mode. Equivalent-Time captured signals can be equalized to see the corrected waveshape when using small enough time resolution or time step to emulate a real-time captured waveform. A number of filters are provided with the most commonly used time steps which are valid for correction. The OUI searches through the Program Files\Optametra\OUI4006\ExecFiles directory and applies the filter it finds with the proper time step. If the proper filter is not found, then none is applied and a warning is printed in the Engine Command Response Window. When the SNR is  $< 20$  dB in ET mode, it is best to turn off the correction filter using `EqFilt = []`; in the Engine Command Window.

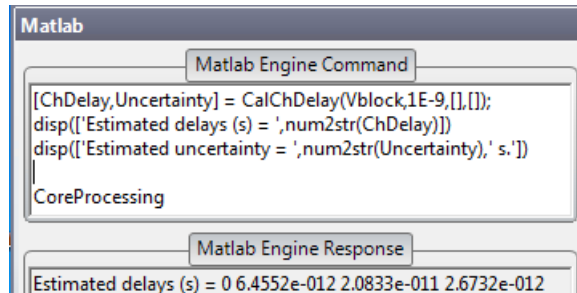
For more information, or for equalization support of a different scope sampling rate, contact customer support.



## Appendix D: Automatic receiver deskew

When setting up for the first time or whenever the channel delays are completely unknown, it is best to use the utility CalChDelay as shown in the figure at step 6. To use this utility, do the following steps:

1. Complete the initial setup.
2. Launch the Laser Receiver Control Panel (LRCP) and connect to the OM4000 instrument and use the drop-down menu to select the reference laser. (See page 19, *The Laser Receiver Control Panel (LRCP) user interface*.)
3. Connect a high-baud rate BPSK signal to the Input of the OM4000 instrument.
4. Tune the reference laser to the same WDM channel as the BPSK signal. Use the oscilloscope controls to verify that the BPSK source is on and at proper level. It does not have to be perfectly biased. Adjust the vertical gain on the oscilloscope so that the signal is filling up at least 50% of the oscilloscope screen. Adjust the signal polarization as needed to get good signal level on all four oscilloscope inputs.
5. Launch the OUI software and connect to the oscilloscope using the Setup tab. Ensure that the proper LO Frequency is displayed on the Setup tab as well. On the display tab select Single-pol BPSK and also select 1 Pol BPSK on the Analysis Parameters tab. Enter the Clock Frequency of the BPSK signal.
6. Enter the CalChDelay function in the MATLAB Engine Command window.



```
Matlab
Matlab Engine Command
[ChDelay,Uncertainty] = CalChDelay(Vblock,1E-9,[],[]);
disp(['Estimated delays (s) = ',num2str(ChDelay)])
disp(['Estimated uncertainty = ',num2str(Uncertainty),' s.'])
CoreProcessing
Matlab Engine Response
Estimated delays (s) = 0 6.4552e-012 2.0833e-011 2.6732e-012
```

7. Click **Single** to take an acquisition; be sure no errors are reported in the MATLAB Engine Response.
8. Use the sliders in the Calibrate tab to set the displayed ChDelay for future use (only the last 3 numbers are set since the first is always zero).



---

# Appendix E: Equivalent-Time (ET) oscilloscope measurements

## Configuring hardware (ET)

Ensure that the required power sources for the OM4000 Series (100, 115 or 230 VAC, 50–60 Hz, 0.4 A), the associated oscilloscope, and the external computer (if used) are available. The OM4000 Series Coherent Modulation Receiver, along with proprietary software comprises the OM4000 Series Coherent Lightwave Signal Analyzer (CLSA). This system is used in laboratory or industrial facilities to analyze next-generation complex-modulation fiber-optic data signals.

The signal to be analyzed is connected to the “Signal” optical input. Four coaxial cables connect the OM4000 Series to a high-speed sampling oscilloscope. An Ethernet connector will connect the receiver, via a router, to a computer and to the oscilloscope. An IEC power cord is connected to a rack or wall outlet. The OM4000 Series User Interface running on the computer controls the OM4000 Series and the oscilloscope.

---

**NOTE.** *A password is required to turn on the lasers through the Laser / Receiver Control Panel. The default is ‘1234’*

---

Associated cabling includes the power cord, an Ethernet cable, four coaxial cables, a polarization-maintaining fiber cable to connect the Reference Input and a standard single-mode fiber to connect the user’s signal to the Signal Input. (See page 142, *Setting up an ET Oscilloscope.*)

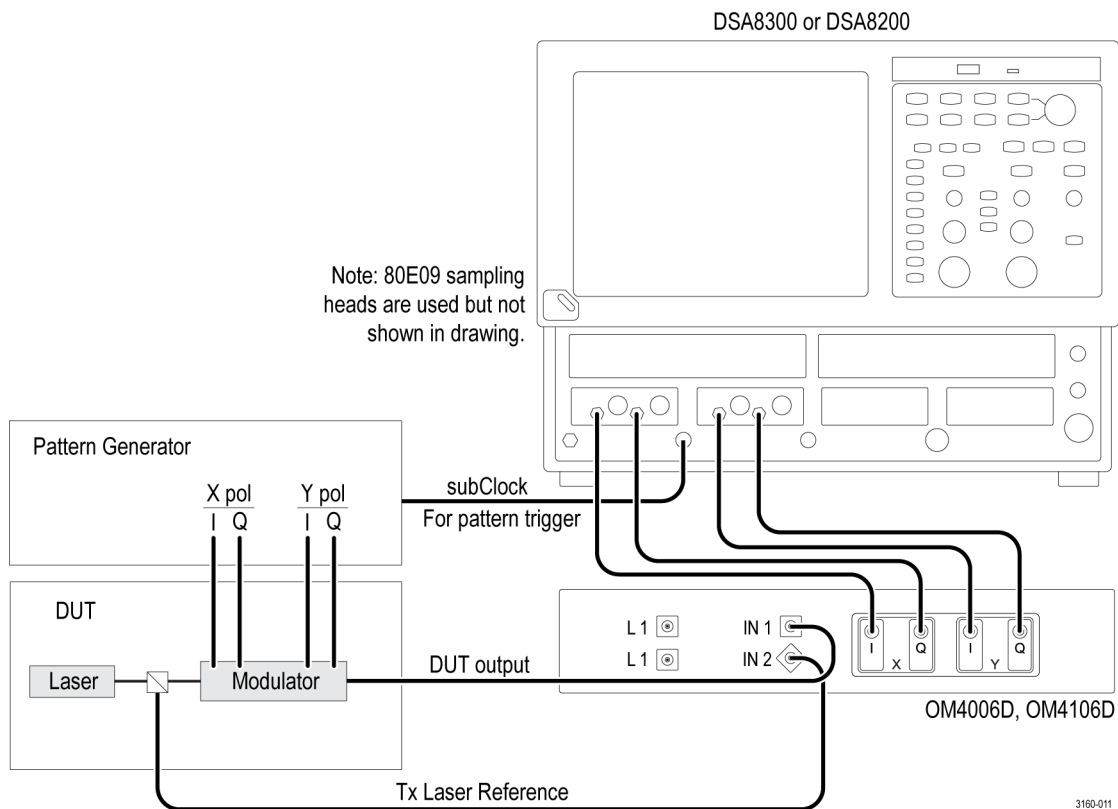


Figure 17: Typical ET oscilloscope setup diagram

The Tx Laser Reference should be connected with Polarization-Maintaining (PM) fiber. Ethernet connections are not shown. All instruments need to be connected to the same network.

**Connections list (ET)**

OM4000 Series Complex Modulation Receiver:

- Power cable
- Ethernet cable
- BNC shorting cap for interlock
- (4) Dust covers for optical inputs not in use
- (4) SMA caps to protect electrical outputs
- Short PM patch cable to connect Laser 2 to Reference input

Short coaxial cables shaped to connect OM4000 Series outputs to oscilloscope inputs Additional items needed, not part of Receiver:

- Supported oscilloscope (1 of the following):
  - Equivalent-Time Tektronix oscilloscope DSA8300 or DSA8200 with supported sampling heads. See data sheet for supported samplers
- Power cable
- Ethernet cable
- Mouse and keyboard (unless touch-screen controlled)
- System controller PC (running Windows 7 or XP, MATLAB software, the OM4000 Series User Interface (OUI) software, and the Laser Receiver Control Panel (LRCP) software):
  - Monitor plus cable
  - Mouse and keyboard
  - Power cables
  - Ethernet cable
- An Ethernet switch or hub plus a router running DHCP, and associated cabling (not shown)
- Equipment for calibration as needed (see Calibration section)
- OMRACK, 19" rack, or other method of ensuring OM4000 Series and ET oscilloscope are securely stacked



**WARNING.** *To reduce the risk of fire and shock, ensure that the AC supply voltage fluctuations do not exceed 10% of the operating voltage range.*

*To avoid the possibility of electrical shock, do not connect your OM4000D to a power source if there are any signs of damage to the instrument enclosure.*

---

### Electrical power requirements

The OM4000 Series can operate from any AC power source that provides 100, 115, or 230 VAC, at a frequency of 60 Hz or 50 Hz respectively with a 0.4 A rating. (The US rack-mount system has a power connector that requires the special 20 A outlet configuration.) The OM4000 Series must be connected directly to a grounded power outlet only.



**WARNING.** *Always connect the unit directly to a grounded power outlet. Operating the OM4000D without connection to a grounded power source could result in serious electrical shock.*

---



**CAUTION.** *Protective features of the OM4000D series instrument may be impaired if the unit is used in a manner not specified by Tektronix.*

---

**Location, positioning, and operating environment**

If the OM4000 Series is to be used in an installation other than a standard 19" rack, be sure to position the unit so that the power switch at the rear of the unit can be easily accessed. Do not obstruct the fan so that there is an adequate flow of cooling air to the electronics compartment whenever the unit is operating.

Make sure that you operate the OM4000 instrument in the appropriate environment. (See page 4, *Environmental operating requirements*.)

**Computer and software requirements**

Make sure that your controller computer (PC or oscilloscope) meets minimum requirements. (See page 6, *PC requirements*.)

Make sure that you have installed all necessary software. (See page 6, *Software installation*.)

## Configuring the software (ET)

Before you can connect to a DSA8300 or DSA8200 Equivalent-Time Oscilloscope, you will need to install the ET Scope Service Utility on the target ET oscilloscope by following the software installation instructions provided with the software distribution. Once the Utility is installed, please start the "Socket Server" and the Oscilloscope Application before starting the Utility using the desktop icon.

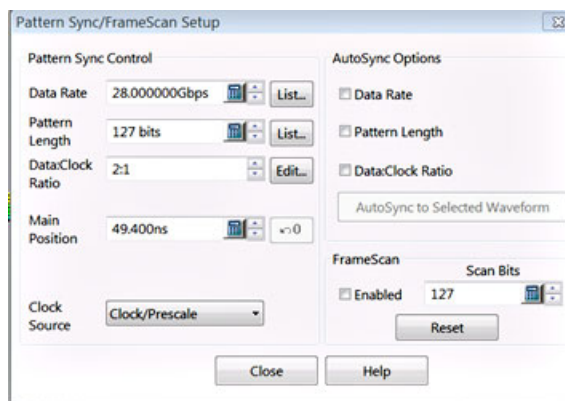
If you have trouble launching the ET Scope Service Utility, check that the Socket Server is running and that you have a valid default.stp file. At a minimum, the default.stp file should have 4 channels enabled. Restart the oscilloscope after making corrections.

On the first launch the Utility prompts you to set up the DSA8000 oscilloscope the way you want it, after which it will save that setup to My Documents\TekScope\UI\default.stp. You can change the default setup at any time by over-writing the default.stp file in the Tektronix Scope Utility for ET folder.

Make sure to set the following in the **DSA8300** instrument:



- **Setup > Mode/Trigger** tab:
  - Select the Trigger Source as either **Clock** or **Direct**, based on which trigger input is connected.
  - Click the **Pattern Sync/FrameScan Setup** button and enter the appropriate parameters into the **Pattern Sync/FrameScan Setup** dialog box, based on settings from the data generator.



**For the DSA8200:** select Pattern Sync or External Direct based on which trigger input is connected. When using the pattern trigger module, be sure to connect the module's Trigger Output to the Direct Trigger input and choose Pattern Sync.

- **Setup >Vert** tab:
  - Use Channels 1-4 or Channels 5-8. Also set the channels used in the ET Scope Service Utility (SSU) application.
  - Be sure that Deskew values are **zero**. Deskew is managed by the Scope Service Utility (SSU).
- **Setup > Acq** tab:
  - Set the Acquisition mode to **Sample**
  - Set the Stop After mode to **Condition, Number of Acquisitions, 1**.

For the DSA8200:

- **Trig** tab: Select **Pattern Sync** or **External Direct** based on which trigger input is connected.
- When using the pattern trigger module, be sure to connect the module's Trigger Output to the Direct Trigger input and select Pattern Sync

When the oscilloscope is correctly configured for your signals, click OK to allow the SSU to finish its initial launch. This configuration will be recalled any time you launch the SSU. This is done so that the oscilloscope may be used for other tasks and easily made ready for use with the OUI again. If you make changes to the setup, be sure to over write the default.stp file so that your settings will be recalled next time.

## OM4000 User Interface (OUI) (ET)

Start the OUI in Vertex Processing mode with the icon on your desktop or in the Programs menu.

---

**NOTE.** Be sure that Matlab is available and properly licensed, since the OUI will attempt to launch a Matlab Command Window, and will appear to stall if Matlab is not available.

---

Connecting to the oscilloscope upon OUI startup is done with the Connect button in the Scope Setup section of the Setup ribbon.

---

**NOTE.** The Scope Service Utility runs on the target oscilloscope. Be sure to install the proper version for equivalent-time (ET) oscilloscopes. See software installation information.

---

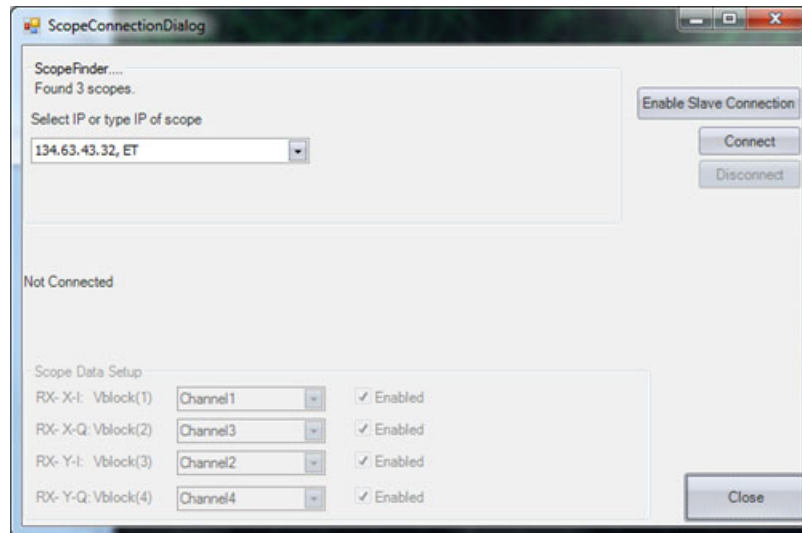
When connecting from the OUI, you will see a check box for VISA. Do not check the “use Visa” box when connecting to an ET oscilloscope.



---

**NOTE.** Clicking Connect on the OUI Setup Tab brings up the Scope Connection Dialog box for connecting to the Scope Service Utility.

---

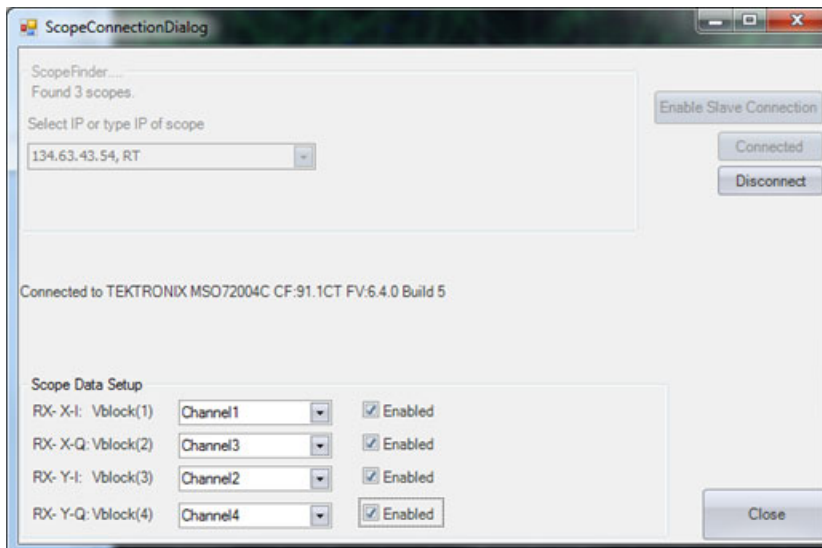


The green bar at the top indicates that the software is searching for oscilloscopes on the same subnet that are running the Scope Service Utility. As they are found they are added to the drop-down menu. If the OUI Scope Connection Dialog box reports 0 Scopes Found, you will have to type in the IP address manually. This happens when connecting over a VPN or when network policies prevent the IP broadcast. When typing the address in manually, do not include “, ET” or “, RT” on the end. Click **Connect**.

After connection, map the channels to the physical receiver channels and corresponding MATLAB variables as shown. This means that data from the selected channel will be moved into the indicated Vblock variable:

- Vblock(1) is X-Inphase
- Vblock(2) is X-Quadrature
- Vblock(3) is Y-Inphase
- Vblock(4) is Y-Quadrature

The mapping you choose will depend on the cable connections made to the OM4000.



Once connected and configured, close the connect dialog box. The OUI is ready to use.

## Calibration and adjustment (ET)

The OM4000 receiver requires the following calibration before taking measurements:

- DC calibration (to compensate for any offsets in the photodiode outputs)
- Delay adjustment (channel to channel skew in the scope connections)
- Hybrid calibration (correction for cross-talk and phase error in the hybrid)
- Laser linewidth factor (choosing the correct filter for phase recovery) (See page 34.)
- Receiver equalization

### DC calibration (ET)

Although the OM4000 Series units use balanced detection, there is usually some small offset voltage present at the signal outputs. Since the Common-Mode Rejection Ratio (CMRR) is optimized for the signal input, the dc offset is almost entirely due to the Reference (LO) input power. The presence of a dc offset causes the recovered signal to be unstable.

To correct for dc offset when connected to an equivalent time oscilloscope, follow this procedure:

1. Set up all of the connections for your anticipated measurement. Connect to the receiver using the LRCP and turn on the sources you require. Verify that the oscilloscope is set up correctly.
2. Disconnect the Signal Input to the receiver, so that only the Reference (LO) is present.
3. Click the **DC calibration** button on the Setup Ribbon. This initiates a process which determines the DC levels in the receiver's photodiodes, and subtracts this value during analysis.
4. Reconnect the Signal Input.

This can be done as often as needed or desired. If there is uncompensated dc offset in the system, this will be evident by a smearing out of the constellation point groups. If the offset is large enough, the point groups will begin to look like donuts. Perform a dc calibration whenever there is any question.

Homodyne detection, which is assumed when using the software in equivalent-time mode, creates special difficulty for dc offset measurement since a homodyne signal can have an important dc component. This is why the Input Signal is disconnected in the procedure. However, if the Input Signal is comparable in magnitude to the Reference Input, the above procedure will not remove all of the dc-offset

### Delay adjustment (system deskew) (ET)

---

**NOTE.** *Initial delay adjustment should be done by trained personnel. This section is provided for experienced users. Delay adjustment should be done during installation and should not require attention unless the scope is removed and/or reconnected.*

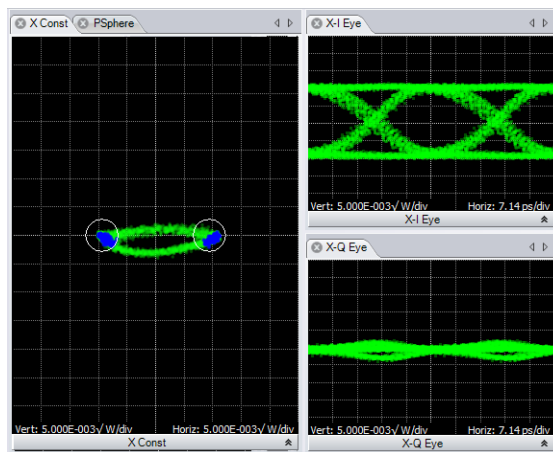
---

Delay adjustment among the four electrical channels of the oscilloscope is done through the sliders on the ET Scope Service Utility (SSU) running on the ET oscilloscope.



Click the button corresponding to the mainframe channels for the sampler-to-receiver connections. Channels 5:8 are the default selection because these are closer to the OM4000 inputs.

Use the sliders to get the best possible eye diagram and constellation diagram. Improper skew will cause horizontal eye closure and filling in of the constellation diagram. If the I and Q channels have unequal delay, there will be a phase offset proportional to the difference frequency between the reference and signal laser oscillation frequencies. This phase offset will turn a straight line on the phase diagram into a circle or a portion of a circle. So, for verification, it is best to use a known Mach-Zehnder modulator to generate a BPSK optical input. As the input signal polarization is adjusted, the I-Q diagram should always be a straight line.



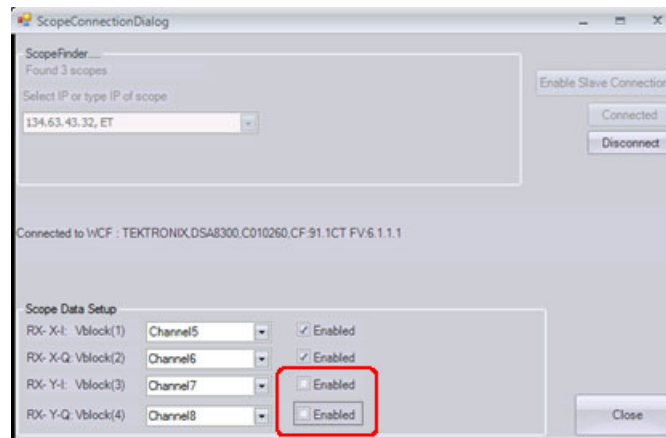
**Figure 18: ChDelay(2) off by 2 ps causes curvature on constellation and signal on Q-Eye for 28 Gbps BPSK**

Note that the sliders control the 1:2 delay, the 1:3 delay and the 3:4 delay (or 5:6, 5:7, 7:8 if right-hand mainframe slots are chosen). These are the most convenient groups:

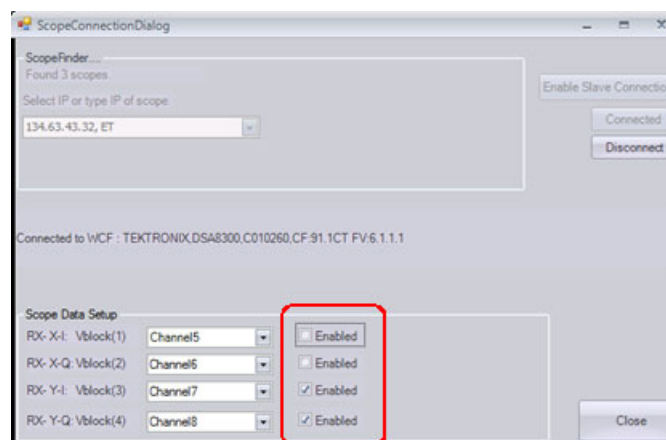
- The 1:2 delay error causes I-Q crosstalk on channels 1 and 2
- The 3:4 delay error causes I-Q crosstalk on channels 3 and 4
- The 1:3 delay error causes X-Y crosstalk

The proper slider values are found by zeroing out these crosstalks one at a time. First by looking only at channels 1 and 2, then only 3 and 4, then finally all channels on to see the X-Y crosstalk. Do the following:

1. Delete everything from the MATLAB Engine Window except for CoreProcessingCommands
2. Perform a DC calibration (See page 111, *DC calibration (RT)*).
3. Adjust the input polarization so the signal is mostly on oscilloscope RX X-polarization channels. This can be done mathematically by disabling the RX Y-polarization channels:

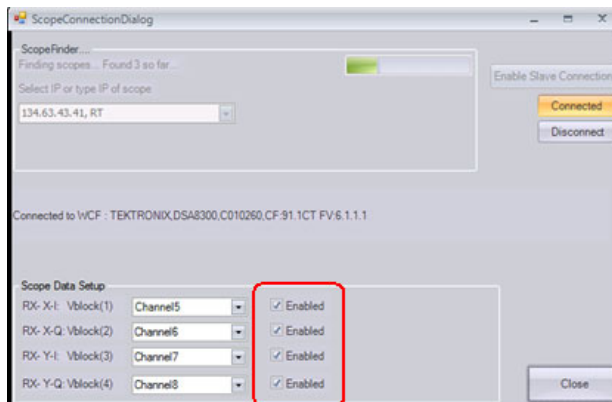


4. Now only the top slider will make any difference. Click **Run** to get a repeating constellation and eye-diagram update. Click the + and – to make 0.1-ps adjustments to the top slider until the BPSK constellation shows as perfectly straight lines connecting the two groups of constellation points as shown above. (See Figure 18.)
5. Displaying the X-Q eye will show the BPSK signal going into the “wrong” quadrature. When the delay is set properly the X-Q signal shown should only be noise.
6. Shut down the RX X-polarization by again moving the polarization state of the signal and disabling channels the RX X-polarization channels. Now only the bottom slider will matter.



7. Move the last slider using the + and – buttons to get 0.1-ps changes until the constellation looks as it did with the RX X-polarization channels. Once again use the X-Q eye as a measure of residual error as well as constellation quality. Note that the data is displayed in the X-I diagram because X and Y on the OUI plots refer to transmitter polarization signals, not receiver outputs. When only one transmitter polarization signal is present it is called “X.”

- Once this is done get equal signal on both polarizations by re-enabling all four channels:



- The last step is to set the middle slider to achieve minimum signal in the Y-polarization. Since the input is a single-pol signal at this point, nothing should be in the Y constellation or eyes except for noise.
- Display the Y-I or Y-Q eye diagrams and Y-Constellation to see the signal going into the wrong polarization. This should just be noise when the middle slider is set properly.
- The delay calibration is done if there is only noise in eye plots the X-I and only noise in the Y-constellation for a single-pol BPSK signal.
- Save the setup on the oscilloscope using the **File > Save Setup As** command to overwrite the default.stp file in My Documents\TekScope\UI\default.stp.

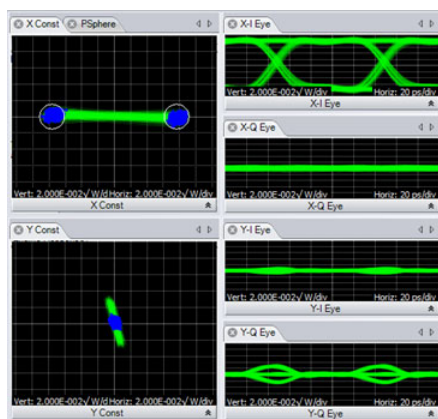


Figure 19: When adjusting the middle slider, watch the Y-Eye and Y-Const to minimize the signal in the Y-polarization



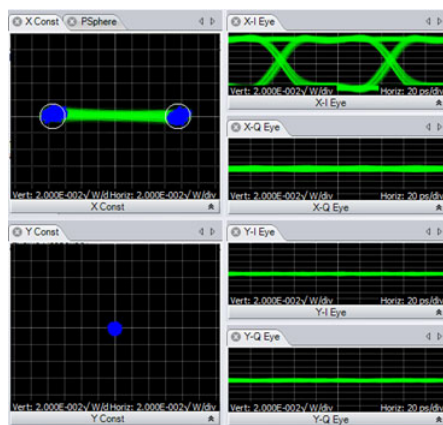
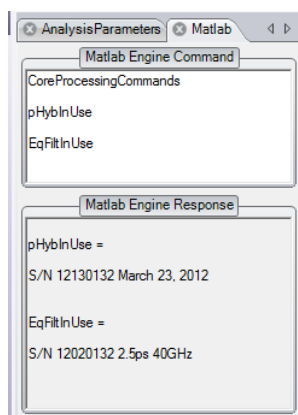


Figure 20: Final channel delay values provide only noise in Y polarization

### Hybrid calibration (ET)

Hybrid calibration is a factory calibration. Imperfections in the OM4000 instrument receiver are corrected using a factory-supplied calibration table. Check the Setup tab for a green indicator to be sure the OUI is successfully retrieving the Reference laser (Local Oscillator) frequency and power which are needed to choose the correction factors from the calibration table. The table is the pHybCalib.mat file in the ExecFiles directory.

You can verify you have a valid pHybCalib file by connecting to an oscilloscope, typing pHybInUse in the MATLAB Engine Window, and clicking Single. Similarly, EqFiltInUse shows the equalization filter in use (if any).



The information statement contains the serial number, date of calibration, and other notes about the calibration. If the serial number is correct then you have the proper file.

The following procedure can be used to verify and correct the calibration at a single wavelength using a minimum of external hardware. This procedure assumes that “Laser 1” and Laser 2” can be tuned to the same wavelength. It is easiest if you have a polarization controller before the signal input, but the instructions are written assuming you are moving the fiber around to change the input polarization.

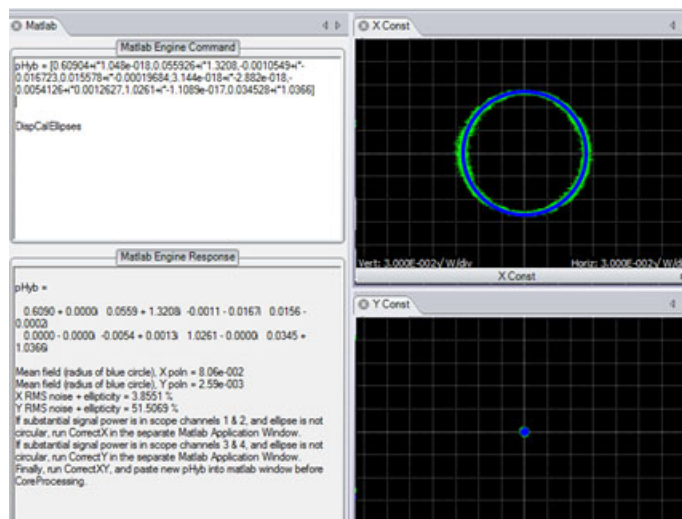
**Prerequisites.**

1. System should be set up and de-skewed following the procedure above.
2. Connect the Reference from Laser 2 to the Reference input with short PM/APC jumper.
3. Use a standard SM/APC (not PM) fiber to connect the Laser 1 to the Signal input.
4. Use the LRCP to set the Signal 1 power level to about 10 dBm to start.
5. Use the LRCP to turn on both lasers and tune them to the same channel where you will be working. Set the Laser 1 power to get about 100mV pp.
6. Use the LRCP to tune Laser 1 off grid by 100 to 500 MHz.
7. Click **Run** on the oscilloscope; the data should look like randomly sampled sine waves.
8. Move the SM fiber around until you get most signal on RX channels X-I and X-Q (at least 3:1 ratio between X-I and Y-I. This is most easily done with a polarization controller).
9. Tape the fiber down so that you continue to get most signal on X-I and X-Q.
10. Click **Single** on the oscilloscope.

**Procedure to inspect and correct the X-polarization calibration.**

1. Use the Optametra OUI to connect to the scope. Record length of ~8000 points recommended.
2. Display the Matlab Engine Window, the X-Constellation Window and the Y-Constellation Window. Close other windows.
3. Be sure the Constellation Continuous traces is checked (selected) to show the green curves.
4. Put DispCalEllipsesET in the Matlab Engine Window of the OUI. Delete everything else in the Matlab Engine Window.
5. Click **Run** on the OUI.
6. Observe that the ellipses are displayed on the Constellation plots. Right now only the X-constellation has signal. The green trace should line up with the blue circle in the X-constellation plot.

7. If the green trace in X-Constellation is elliptical:
  - Click **Stop**.
  - Type **CorrectX** in the separate Matlab Application Window.
  - Copy and paste the resulting pHyb statement before DispCalEllipses in the Matlab Engine Window in the OUI as shown below.
8. Click **Run**. The green trace should now be circular. If it is not, repeat the procedure one time.

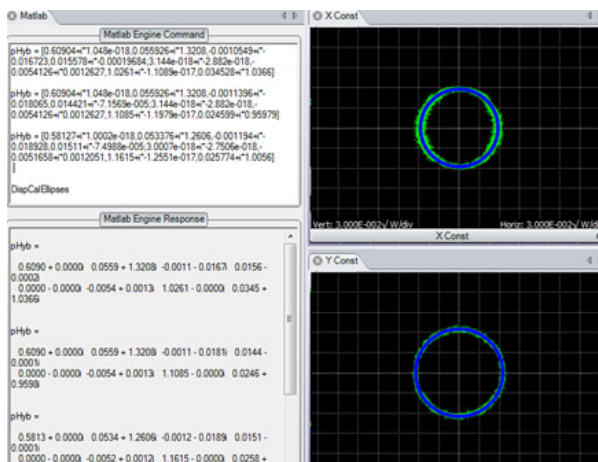


### Procedure to inspect and correct the Y-polarization calibration (ET).

1. Move the input fiber to get most of the signal on RX channels Y-I and Y-Q. Tape it down.
2. Click **Single** on the Oscilloscope.
3. Click **Run** on the OUI.
4. Observe that the ellipses are displayed on the Constellation plots. Right now only the Y-constellation has signal. The green trace should line up with the blue circle in the Y-constellation plot.
5. If the green trace in Y-Constellation is elliptical:
  - Click **Stop**.
  - Type **CorrectY** in the separate Matlab Engine window.
  - Copy and paste the resulting pHyb statement before DispCalEllipses, replacing any other pHyb statement.
6. Click **Run**. The green trace should now be circular. If it is not, repeat this procedure one time.

**Procedure to Correct the relative X-Y gain (ET).**

1. You must complete all of the above steps first including CorrectX and CorrectY.
2. Type **CorrectXY** in the separate Matlab Application Window.
3. Copy and paste the resulting pHyb statement before DispCalEllipses, replacing any other pHyb statement.
4. Move the input fiber until there is signal on all four channels.
5. Click **Run**. The green trace should now be circular in both Constellations.



**Finish the hybrid calibration (ET).**

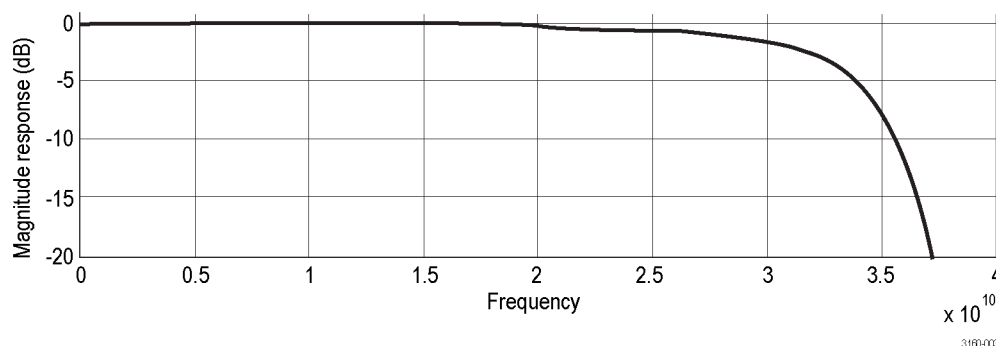
1. The pHyb statement in 3 is the final output that is corrected for RX X-polarization, I-Q gain and phase, RX Y-polarization, I-Q gain and phase, and X-Y relative gain.
2. Replace the DispCalEllipses statement with CoreProcessingCommands for normal operation. Keep the pHyb statement as it is correcting the calibration.

**Laser linewidth factor.** Refer to information on modifying the default value of Alpha. (See page 34.)

**Receiver Equalization.** Receiver Equalization is a factory calibration. Digital equalization is applied to the four channels of the OM4000 instrument to account for the non-ideal frequency dependent response introduced by the coherent optical receiver and the receiver radio frequency front end. Depending on the sampling rate of the oscilloscope and the bandwidth of the OM4000 instrument, digital equalization is applied so that the combined effect of the receiver and the digital equalization filter meet a specified reference response:

Oscilloscope sampling rate	Reference magnitude response	reference phase
$\leq 2x$ the bandwidth of the OM4000 (BW)	Flat	Linear
$> 2x$ the bandwidth of the OM4000 (BW)	4th order digital (bilinear) Bessel filter with 3 dB cutoff at BW + 2 GHz	Linear

Equalization is specific to the sampling rate of the oscilloscope. As an example, if the bandwidth of the OM4000 is 30 GHz and the sampling rate of the oscilloscope is 80 Gs/s, the combined effect of the OM4000 receiver front end response and the digital equalization filter will be that of a 4th order digital Bessel filter with 3 dB cutoff at 32 GHz, as shown in the following graph.



The digital filter is applied directly to the MATLAB workspace variables *Vblock(1).Values* through *Vblock(4).Values* using a 100 tap FIR filter.

Equalization provides a challenge when working in ET mode. Equivalent-Time captured signals can be equalized to see the corrected waveshape when using small enough time resolution or time step to emulate a real-time captured waveform. A number of filters are provided with the most commonly used time steps which are valid for correction.

The OUI searches through the Program Files\Optametra\OUI4006\ExecFiles directory and applies the filter it finds with the proper time step. If the proper filter is not found, then none is applied and a warning is printed in the Engine Command Response Window. When the SNR is  $< 20$  dB in ET mode, it is best to turn off the correction filter using `EqFilt = []`; in the Engine Command Window.

For more information, or for equalization support of a different scope sampling rate, contact customer support.

## Setting up an ET Oscilloscope

### Equivalent-time oscilloscope setup

See the following figure for how to connect the OM4000 instrument to take measurements with equivalent-time (ET) oscilloscopes (Tektronix DSA8300 or DSA8200 sampling oscilloscopes).

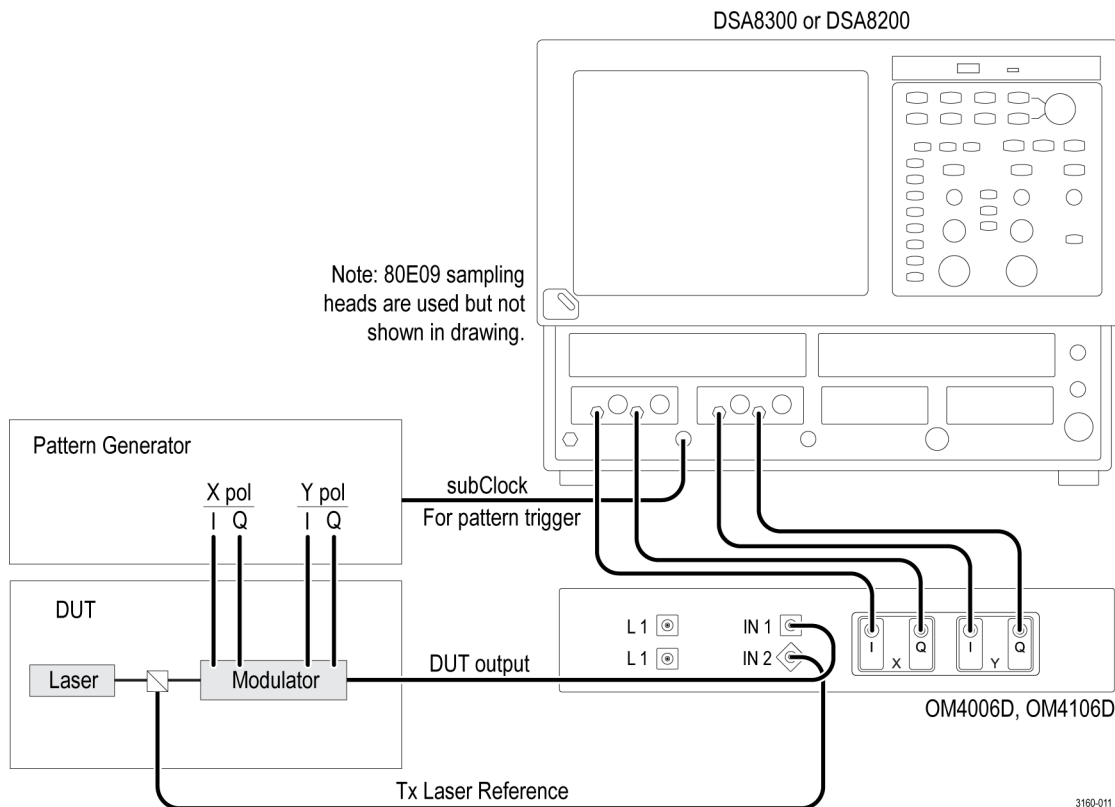


Figure 21: Equivalent-time (ET) oscilloscope setup diagram

The most important difference between the real-time (RT) and equivalent-time (ET) oscilloscope measurements is the need for a coherent reference signal for ET oscilloscopes. The TX Reference signal is picked off before the modulator, using a PM fiber cable, with a total path length equal to the path from the splitting point to the Signal Input on the OM4000 Receiver. Use a SMF fiber cable from the DUT to the Signal Input connection on the OM4000.

Since the laser phase noise is a real-time quantity, it must be sufficiently suppressed so that it can be tracked in the available bandwidth of the ET scope.

As an example, consider a laser with frequency noise given by

$$f(t) = f_0 + f_D \sin 2\pi f_n t$$

If this laser signal is split and then input to the Signal Input and Reference Input of the OM4000, the resulting beat frequency will be

$$f_m(t) \approx (2\pi f_D f_n \Delta t) \sin 2\pi f_n t$$

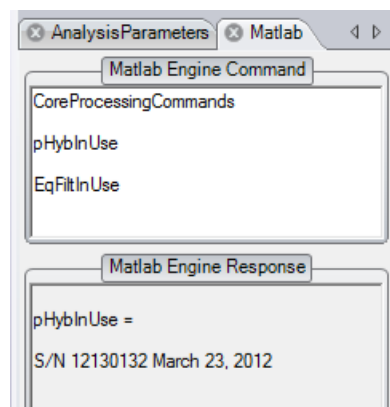
While the modulating frequency,  $f_n$  is still the same, the frequency deviation,  $f_D$ , has been reduced by  $2\pi f_n \Delta t$  where  $\Delta t$  is the time difference for the two paths.

Some lasers can have frequency deviations in the 200 MHz range over 1 ms. To minimize the FM bandwidth after detection, reduce the frequency deviation to  $\sim 1$  kHz. This is accomplished with  $\Delta t = 0.8$  ns or a path difference of 16 cm or less.

Generally speaking, the ET measurement performance is best with a path difference less than 10 cm when possible. For lower noise lasers, path lengths differences up to 2 m can be tolerated.

## Matlab Engine file (ET)

You can configure MATLAB to perform a wide range of mathematical operations on the raw or processed data using the Engine window. Normally the only call is to `CoreProcessingCommands`, the set of routines performing phase and clock recovery.




---

**NOTE.** To view a complete list of variables, open the MATLAB Command Window and enter *who*.

---



---

**NOTE.** `CoreProcessingCommands` will provide either ET or RT processing depending on which mode the OUI is in. To ensure you only get ET processing you can use `CoreProcessingET` in the window instead of `CoreProcessingCommands`. Similarly you can use `CoreProcessing` in the Engine Window if you want to be sure you only get real-time processed data.

---

As with all other settings, the last engine file used is recalled; you can locate or create another appropriate engine file and paste it into the OUI Matlab Command window. Subsequent chapters explain in detail the operations of Core Processing. In addition to any valid MATLAB operations you use, there are some special variables that can be set or read from this window to control processing for a few special cases:

- EqFiltInUse – a string which contains the properties of the equalization filter in use
- pHybInUse – a string which contains the properties of the optical calibration in use
- DebugSave – logical variable that controls saving of detailed .mat files for analysis:
  - DebugSave = 1 in the MATLAB Engine Command window results in two files saved per block plus one final save.
  - DebugSave = 0 or empty suppresses .mat file saves.

## Taking measurements (ET)

Click **Single** in the OUI and observe that the oscilloscope takes a burst of data; this confirms the connection. Using a short record length (such as 2000 points) to speed up the display, click on **Run-Stop** to show continuously-updated measurements.

Use controls in the Home tab to set up the plots and measurements to take, either using the stored Layout button or by clicking on the particular display format icon in the Plot Tools bar. (See page 23, *The OM4000 user interface (OUI)*.)

## OUI overview (ET)

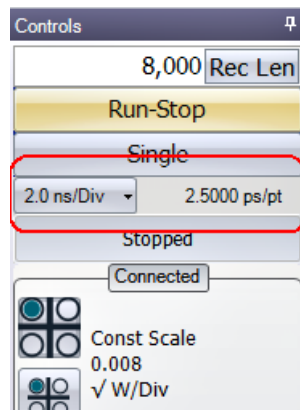
(See page 23, *The OM4000 user interface (OUI)*.)

## OUI Controls panel (ET)

(See page 30, *OUI Controls panel*.)

The difference between the standard (RT) control panel and the ET control panel is that the ET OUI adds a **Time/Div** control. The Time/Div control is located below the Run and Single buttons.





This control reduces the need to use the oscilloscope front-panel controls while using the OUI. The resolution readout is provided for convenience.

## Analysis Parameters window (ET)

(See page 32, *Analysis Parameters window*.)

The following controls are relevant to Equivalent Time processing:

- **Signal Type:** Chooses the type of signal to be analyzed and so also the algorithms to be applied corresponding to that type.
- **Pure Phase Modulation:** Sets the clock recovery for when there is no amplitude modulation.
- **Clock Frequency:** Is the nominal frequency of the data clock bounded by a low (Low) and a high frequency (High) provided the clock signal power is sufficient.
- **Assume Orthogonal Polarizations:** Is not yet implemented for ET mode
- **2nd Phase Estimate:** Is not yet implemented for ET mode
- **Phase estimation time constant parameter:** The parameter used in phase estimation which depends on laser linewidth and noise. The default of 0.8 is usually fine.
- **Balanced Differential Detection:** Not yet implemented for ET mode.
- **Continuous traces:** Checking this box ensures that the fine traces connecting the constellation points will be drawn. If unchecked, the traces will be suppressed for calculation speed if the calculations are not needed for other plots such as eye diagrams.
- **Mask Threshold:** The ratio of radius to symbol spacing used for the circular constellation masks.
- **Symbol Center Width:** The fraction of the eye-center that should be considered “symbol center” for the purpose of certain calculations such as

which symbols to color blue. At present, the sample closest to symbol center is used to represent that symbol for calculations such as BER and Q-factor.

- **Continuous trace points per symbol:** The number of samples per symbol for the clock retiming that is done to create the fitted curves such as Eye and Signal Average and Transition Average.
- **Tributaries contribution to average:** The average waveforms are based on finding the symbol impulse response and convolving with the data pattern. This setting switches which possible crosstalk contributions are included in the calculation of impulse response.
- **Number of symbols in impulse response:** The number of values calculated for the impulse response. More values should provide a more accurate average but take longer to calculate.
- **Calculate linear average eye:** Controls computation of the average eye. Refresh rate is faster when disabled.
- **Calculate linear average vs. time:** Controls computation of the average signal vs. time. Refresh rate is faster when disabled.
- **Calculate transition average:** Controls computation of the transition average. Refresh rate is faster when disabled. However, this must be checked to enable calculations based on transition average such as risetime.
- **Data Patterns:** For error counting, constellation orientation, and two-stage phase estimation, the data pattern of each tributary must be specified. Omitting the data specification or providing incorrect information about your data pattern will not impair the constellation or eye displays except that there will be no consistent identification of each tributary since the identification of I and Q and X and Y is arbitrary in the case where the data is not known. Identify your data patterns for each tributary by choosing a standard PRBS from the drop-down menu, or by assigning the pattern variable directly. When assigning the variable directly be sure to select user pattern from the drop-down menu first.



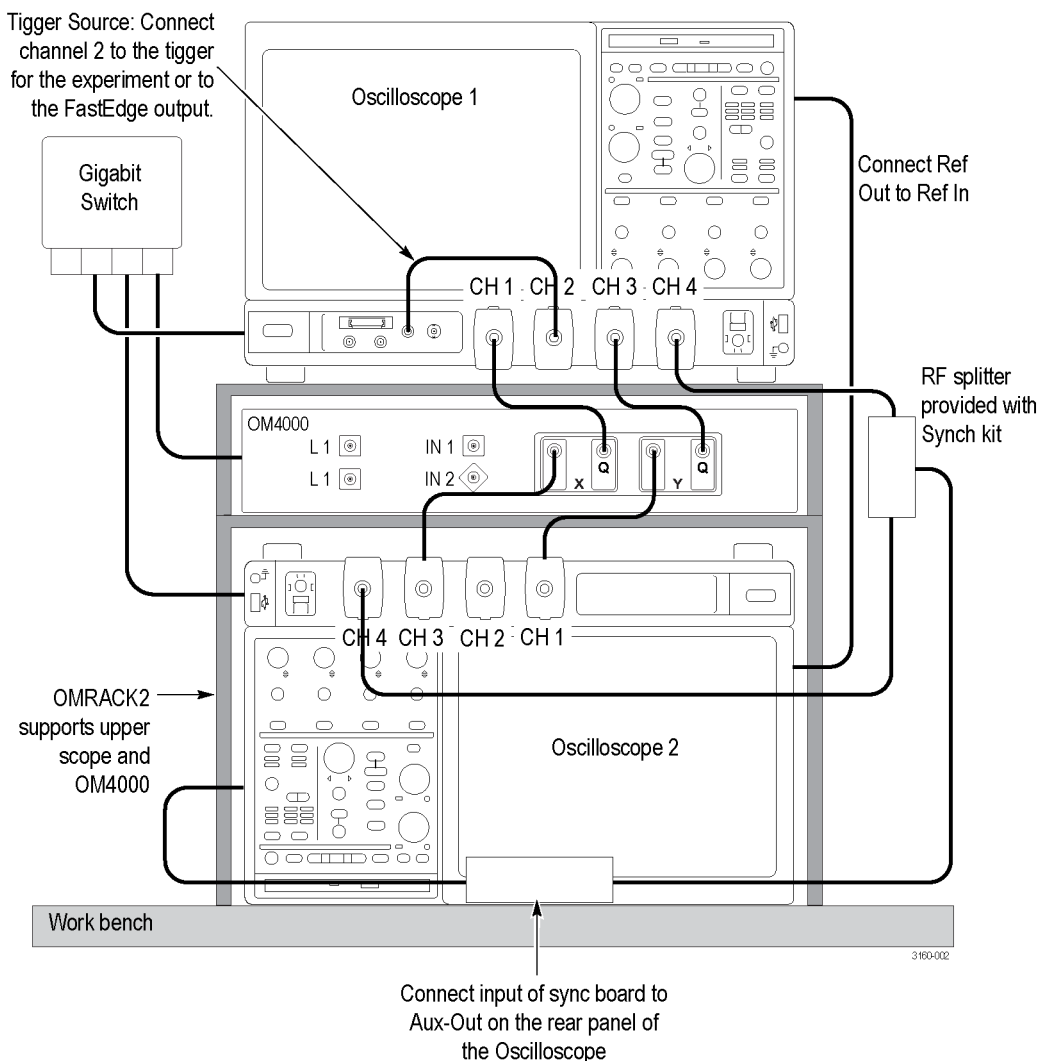
---

## Appendix F: Configuring two Tektronix 70000 series oscilloscopes

OUI Version 1.5 supports a new configuration where two Tektronix 70000-series C- or D-model oscilloscopes are both connected to an OM4000. In this case, the Scope Service Utility (SSU) is installed and run on both oscilloscopes. The OUI connects independently to the two oscilloscopes using the Scope Service Utility running on each oscilloscope.

You can connect one receiver polarization to each oscilloscope. However, with the recommended hardware configuration shown below, it is possible to use the SkewControl function to remove inter-scope jitter. Turning the bottom oscilloscope upside down will shorten the cable length and reduce corresponding cable loss between oscilloscope two and the OM4000.

When connected as shown below, both oscilloscopes will trigger on Channel 4 from the sharpened Fast Edge provided by the Sync Board.



**Ethernet.** Connect each instrument to the GigE switch. If you are using an external PC connect this too. See instructions provided above for configuring the IP addresses.

**USB cable.** Connect the standard USB cable between one of the ports on the scopes and the Sync Board. This cable is simply used to power the board.

**RF cabling.**

- Connect rear-panel BNC connections Ref Out on the master to Ref In on the slave oscilloscope.
- For self-triggering (vs. using an external trigger source): Using an SMA cable, connect the Fast Edge of the Master (usually top scope) to Ch 2 of the Master.

- If using an external trigger source instead of self-triggering, drive Ch 2 of the Master with the external source instead of using the Fast Edge of the Master. Configure the master Ch 2 input as needed to trigger off of your trigger source.
- Using a BNC cable, connect AUX OUT of Master to input of Sync Board. Ensure that there is a DC block on input of Sync Board or (for newer Sync Boards) that there is no DC bias applied to the Sync Board's bias input.
- Using identical-length SMA cables, connect the + output of the Sync Board to Ch 4 of each scope; this Sync Board output should be taken from one of its two + output ports and split with a >15 GHz passive splitter (see picture below) to achieve absolute minimum jitter. Using both of the Sync Board's + outputs is also acceptable. In every case, unused Sync Board outputs must be terminated in 50 ohms.
- Connect the IQ signal inputs:
  - Connect X-I on the OM4000 to Ch3 on Oscilloscope 2 (lower oscilloscope)
  - Connect X-Q on the OM4000 to Ch1 on Oscilloscope 1 (upper oscilloscope)
  - Connect Y-I on the OM4000 to Ch1 on Oscilloscope 2 (lower oscilloscope)
  - Connect Y-Q on the OM4000 to Ch3 on Oscilloscope 1 (upper oscilloscope)
  - You can use other IQ-Channel mappings, just be sure to set the OUI Connect dialog channel parameters accordingly.

**Optical.**

- Connect DUT signal to Signal In on the OM4000
- Connect Laser 2 Output to Reference Input with a short PM fiber if not internally connected

## Oscilloscope settings

Be sure the Socket Server is on and run the Scope Service Utility on both oscilloscopes.

---

**NOTE.** *The following screens may look different depending on the version of oscilloscope firmware.*

---

### Master oscilloscope trigger settings

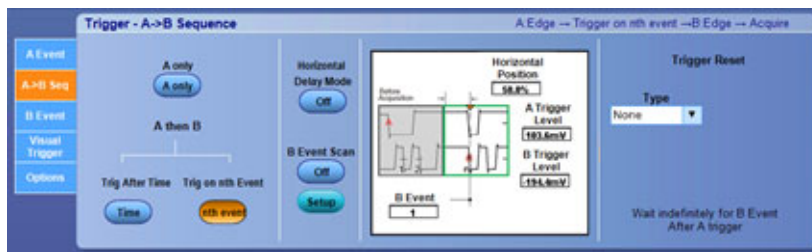
- Set up the Aux-Out on the master scope to provide a positive edge after the A event.
- Set the master Reference to Internal



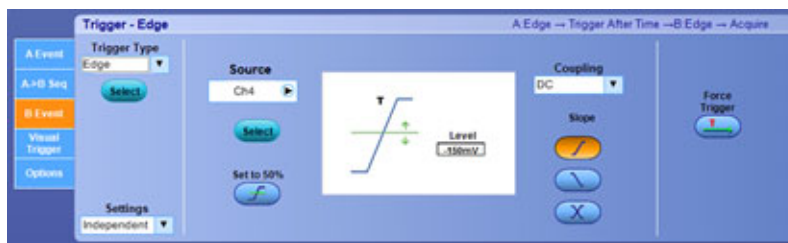
- Set the A Event to Ch 2 with appropriate settings for the Ch 2 input. The settings shown are for the Fast Edge oscilloscope output. This is the primary system trigger.



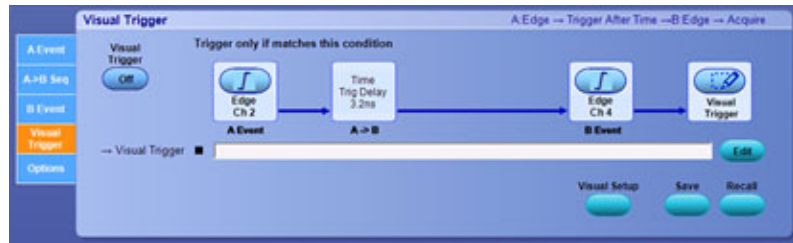
- The master scope gets two triggers: the primary “A” trigger which arms the system, and the “B” trigger from the Sync Board which provides the low jitter trigger relative to the slave scope. Both master and slave need to react immediately to the “B” trigger so the Trigger on nth Event is the best choice with “B Event” set to 1.



- The “B” event levels are set appropriate for the sync board output



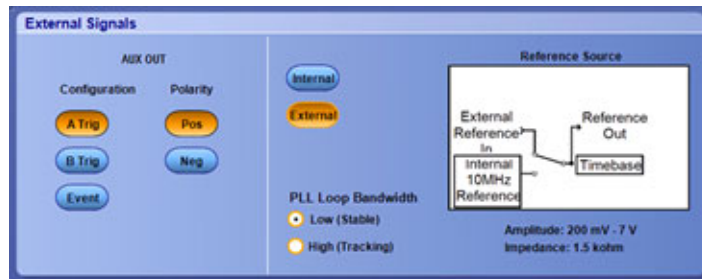
- The visual trigger is not used because Ch 2 and 4 must be turned off to enter 100 Gs/s mode



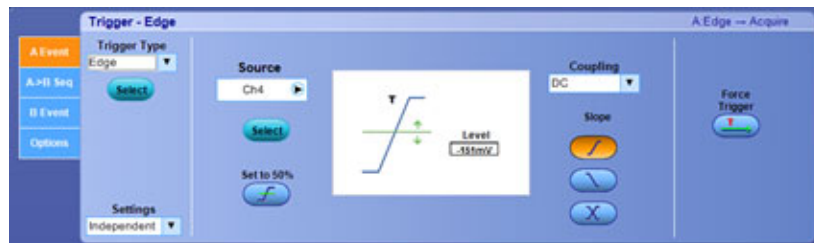
- The trigger holdoff is set to a fixed time which is managed by the OUI. Longer holdoffs are required for longer records.
- In the Options tab, set the Master Scope Trigger Holdoff as required. Make sure that the Slave Scope Trigger Holdoff is set to minimum time.

### Slave oscilloscope trigger settings

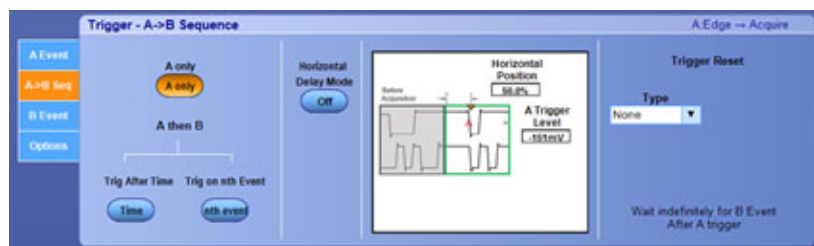
- The slave uses the Reference signal from the master via the rear-panel BNC

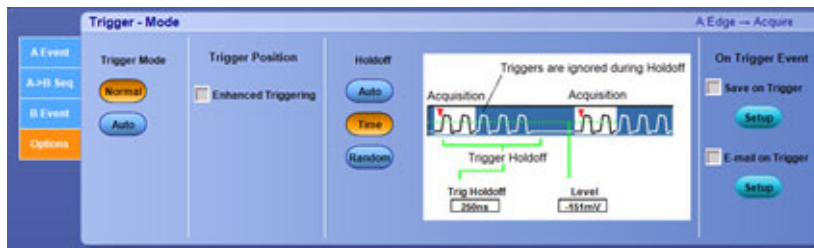


- The Ch 4 trigger settings should be identical to the master so that both take data simultaneously. Residual skew will be handled by the OUI.



- The slave scope has only one trigger. Select the default or minimum holdoff.

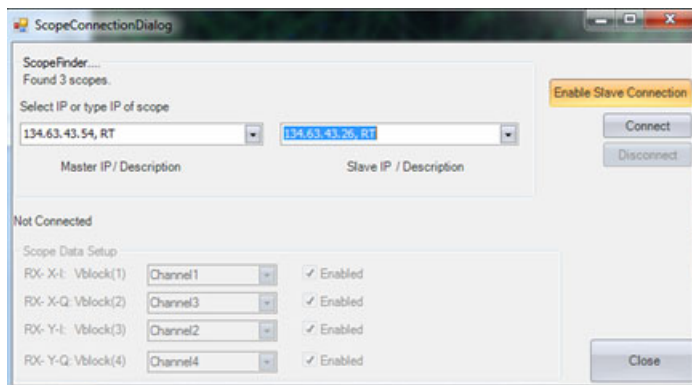




- Turn off Channels 2 and 4 when trigger is setup.
- Set the oscilloscope for 100 Gs/s operation on channels 1 and 3.
- Select the maximum corrected bandwidth (not the HW setting).

## OUI settings for 2-oscilloscope operation

1. On the OUI, click Connect on the Setup Tab.

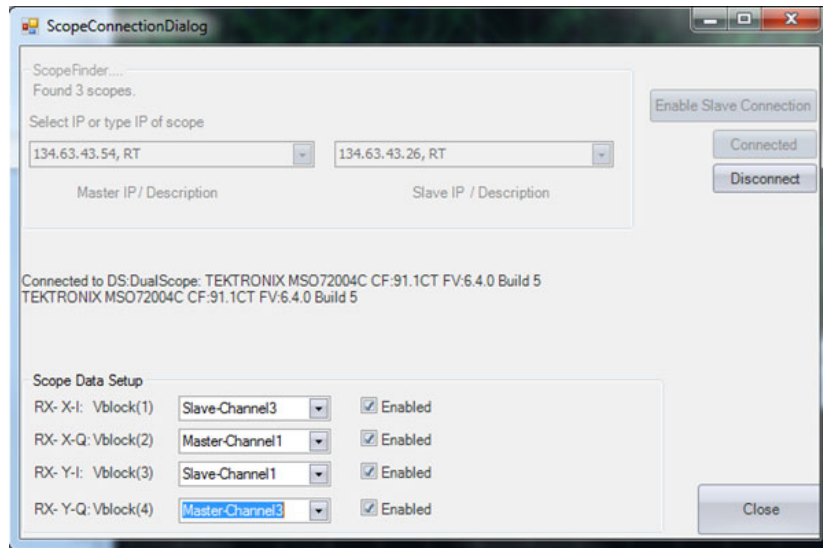


2. Click **Enable Slave Connection** to enable the selection of a second IP address. Find the two oscilloscopes and decide which one is the Master. The Master oscilloscope is the one receiving the external trigger. The Slave scope is the one triggering on the sync board output only.

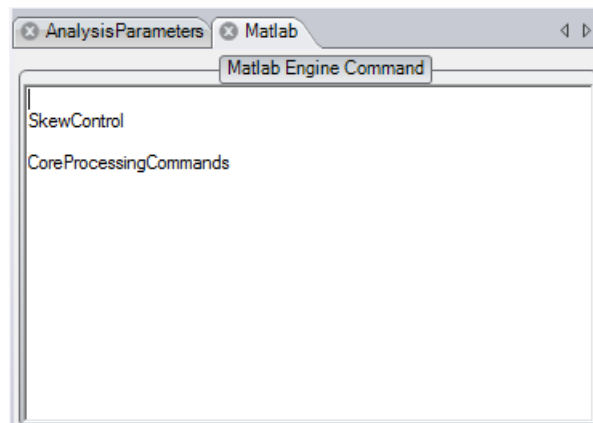
If the OUI Scope Connection Dialog box reports “0 Scopes Found,” you will have to type in the IP address manually. This happens when running over a VPN or when network policies prevent the IP broadcast.

After connection, the drop-down boxes can be populated. You can select channels from both the Master and Slave instruments.





3. To remove inter-scope jitter, add the SkewControl command to the Engine Window as shown in the following figure. SkewControl removes scope to scope jitter by aligning transitions found in the data. Its utility depends on the quality of the data; highly impaired signals may not benefit from SkewControl.





---

# Appendix G: The automated test equipment (ATE) interface

Both the OUI and the LRCP have two types of WCF interfaces to allow control from a user application. Both types are provided to achieve full functionality and compatibility with simple interfaces such as MATLAB and via a client application program.

## The LRCP ATE interface

The Automated Test Equipment (ATE) interface exposes the LRCP functionality through a Windows Communication Foundation (WCF) service. As the LRCP is used with all OM4000 instruments, its interface exposes more commands than those used by the OM4x06 CLSA.

### Basic/Advanced WCF Service Interface for the LRCP

The WCF services (basic and advanced) are available on port 9000 in the machine that is running the LRCP. The service (basic and advanced) interface was developed for incorporation into an ATE client application that can be developed in your choice of .NET language, typically C# or VB.NET. Both services expose most of the functionality that is available through the LRCP's user interface.

The basic service, implemented using a `wsBasicHTTPBinding`, exposes the same subset of commands as the advanced service. It was implemented using a simpler binding for compatibility with applications like MATLAB (See page 167, *ATE functionality in MATLAB.*) or Labview that only support the `wsBasicHTTPBinding`. The basic service is referenced at the following URL: [http://localhost:9000/LaserReceiverControlPanel/Laser\\_ReceiverServiceBasic/](http://localhost:9000/LaserReceiverControlPanel/Laser_ReceiverServiceBasic/)

The advanced service, implemented using a `wsHTTPBinding`, (and which is not available in MATLAB) was developed for use with an ATE client application (See page 169, *Building an OM4006 ATE client in VB.NET.*) and uses events to provide a time-efficient interface.

The advanced service is referenced at the following URL: [http://localhost:9000/LaserReceiverControlPanel/Laser\\_ReceiverService/](http://localhost:9000/LaserReceiverControlPanel/Laser_ReceiverService/)

---

**NOTE.** *For safety reasons, neither the basic or advanced services support the activation of a laser; this must be done on the user interface.*

---

### LRCP service interface function list

The following are the available commands in both the basic and advanced service interfaces and demonstrate their functionality using the MATLAB syntax. *Setting laser parameters* has more information on the LRCP functionality of these exposed functions. (See page 21.)

- **int AvailableLasers(classname);**  
 Description: Returns the count of available lasers on the active controller.  
 Controller Types: All  
 Example: `AvailableLasers(Obj);`  
 Returns: `ans = 2`
- **bool Connect(classname);**  
 Description: Connects to the active controller, starts controller running.  
 Controller Types: All  
 Example: `Connect(Obj);`  
 Returns: `ans = true`
- **bool Disconnect(classname);**  
 Description: Disconnects from the currently active controller, takes offline.  
 Controller Types: All  
 Example: `Connect(Obj);`  
 Returns: `ans = true`
- **bool GetActualCavityLock(classname);**  
 Description: Returns the actual cavity lock state for the active controller/laser.  
 Locked = True.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `GetActualCavity(Obj);`  
 Returns: `ans = true`
- **double GetActualChannel(classname);**  
 Description: Returns the actual channel number for the active controller/laser.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `GetActualChannel(Obj);`  
 Returns: `ans = 1`
- **double GetActualChannel1(classname);**  
 Description: Returns the actual channel 1 frequency (in THz) for the active laser.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `GetActualChannel1(Obj);`  
 Returns: `ans = 191.5`
- **bool GetActualEmitting(classname);**  
 Description: Returns the emission status of the active laser is emitting.  
 Emitting = True.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `GetActualEmitting(Obj);`  
 Returns: `ans = true`
- **short GetActualFineTuneFrequency(classname);**  
 Description: Returns the actual fine tune frequency (in MHz) of the active laser.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `GetActualFineTuneFrequency(Obj);`  
 Returns: `ans = 0`

- **double GetActualGridSpacing(classname);**  
Description: Returns the actual grid spacing (in THz) of the active laser.  
Controller Types: 4006, 4106, 2210, 2012  
Example: `GetActualFineTuneFrequency(Obj);`  
Returns: `ans = 0.05`  
Should see in LRCP screen->
- **double GetActualPower(classname);**  
Description: Returns the actual power (in dBm) of the active laser.  
Controller Types: 4006, 4106, 2210, 2012  
Example: `GetActualPower(Obj);`  
Returns: `ans = 14.5`
- **double GetCalculatedFrequency(classname, laserusagetype);**  
Description: Searches all of the connected controllers for the first laser of the specified laser usage type and returns the calculated frequency (in THz).  
Valid usage types are: unused, signalx, signaly, signalxy, reference.  
Controller Types: 4006, 4106, 2210, 2012, 5110  
Example: `GetCalculatedFrequency(Obj, reference);`  
Returns: `ans = 191.5`
- **string[] GetControllers(classname);**  
Description: Returns a list (array) of controller devices (strings) that are being controlled by the serving application.  
Example: `Controllers = GetControllers(Obj);`  
Returns:  
'OM2210:Prototype1'  
'OM2210:8180123'  
'OM4006:6300121'
- **double GetFirstFrequency(classname);**  
Description: Returns the first frequency (in THz) of the active laser.  
Controller Types: 4006, 4106, 2210, 2012  
Example: `GetFirstFrequency(Obj);`  
Returns: `ans = 191.5`
- **bool GetInterlock(classname);**  
Description: Returns the current interlock state of the active controller. The normal, working state is TRUE. If the interlock is disconnected from the back of the instrument or if the instrument is powered off, this function returns FALSE.  
Controller Types: All  
Example: `GetInterlock(Obj);`  
Returns: `ans = true`
- **string GetIP(classname);**  
Description: Returns the IP Address (as a string) for the active controller.  
Controller Types: All  
Example: `Address = GetIP(Obj);`  
Returns: `Address = '172.17.200.114'`

- **double GetLastFrequency(classname);**  
 Description: Returns the last frequency (in THz) of the active laser.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `GetLastFrequency(Obj);`  
 Returns: `ans = 196.25`
- **float GetPhotoCurrent(classname);**  
 Description: Returns the photocurrent (in mA) of the receiver in the active controller.  
 Controller Types: 4006, 4106  
 Example: `GetPhotoCurrent(Obj);`  
 Returns: `ans = 11.034`
- **polarization GetPolarization(classname);**  
 Description: Returns the polarization state.  
 Valid Polarization states: filter1, filter2, unknown, hardwarefailed.  
 Controller Types: 2210  
 Example: `GetPolarization(Obj);`  
 Returns: `ans = filter2`
- **bool InitializePolarization(classname);**  
 Description: Initializes the polarization state. Returning True = Successful.  
 Controller Types: 2210  
 Example: `InitializePolarization(Obj);`  
 Returns: `ans = true`
- **bool SetActiveControllerByIPAddress(classname, string\_ipAddress);**  
 Description: Sets the active controller by IP Address. True = Successful.  
 Controller Types: All  
 Example: `SetActiveControllerByIPAddress(Obj, '172.17.200.112');`  
 Returns: `ans = true`  
 Returns (after `GetIP(Obj)`): `ans = '172.17.200.112'`
- **bool SetActiveControllerByName(classname, string\_activeController);**  
 Description: Sets the active controller by name. True = Successful.  
 Controller Types: All  
 Example: `SetActiveControllerByName(Obj, 'OM4106:6300121');`  
 Returns: `ans = true`
- **bool SetActiveLaser(classname, byte\_activeLaser);**  
 Description: Sets the active laser. Returning True = Successful.  
 Controller Types: 4106, 4006, 2210, 2012  
 Example: `SetActiveLaser(Obj, 2);`  
 Returns: `ans = true`
- **bool SetControllerAndLaserByUsageType(laserusagetype laserUsageType);**  
 Description: Searches the “running” controllers for a laser matching the requested usage type (usually reference) and selects that controller and laser.  
 Controller Types: 4006, 4106, 2210, 2012

Returns: ans = True - First laser for the specified usage type was selected  
 False - No lasers found on running controllers for the specified usage type

- **bool SetDesiredCavityLock(classname, bool\_desiredCavityLock);**  
 Description: Sets the desired cavity lock state for the active laser. 1 (True) = Locked. Returning True = Successful.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `SetDesiredCavityLock(Obj, 1);`  
 Returns: ans = true
- **bool SetDesiredChannel(classname, int\_desiredChannel);**  
 Description: Sets the channel number for the active laser. Returning True = Successful.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `SetDesiredChannel(Obj, 45);`  
 Returns: ans = true
- **bool SetDesiredChannel1(classname, double\_desiredChannel1);**  
 Description: Sets the channel 1 frequency (in THz) for the active laser. Can only be set if the active laser is NOT emitting. Returning True = Successful.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `SetDesiredChannel1(Obj, 192.5);`  
 Returns: ans = true
- **bool SetDesiredEmittingOff(classname);**  
 Description: Sets the Active Laser to Off (not emitting). Returning True = Successful.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `SetDesiredEmittingOff(Obj);`  
 Returns: ans = true
- **bool SetDesiredEmittingOn(classname);**  
 Description: Sets the Active Laser to emitting. Returning True = Successful.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `SetDesiredEmittingOn(Obj);`  
 Returns: ans = true
- **bool SetDesiredFineTuneFrequency(classname, short\_desiredFineTuneFrequency);**  
 Description: Sets the desired fine tune frequency (in MHz) of the active laser.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `SetDesiredFineTuneFrequency(Obj, 300);`  
 Returns: ans = true
- **bool SetDesiredGridSpacing(classname, double\_desiredGridSpacing);**  
 Description: Sets the desired grid spacing (in THz) of the active laser. Can only be set if the active laser is NOT emitting. Returning true = Successful.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `SetDesiredGridSpacing(Obj, 0.05,0);`  
 Returns: ans = true

- **bool SetDesiredPower(classname, double\_desiredPower, bool\_waitUntilFinished);**  
 Description: Sets the desired power (in dBm) of the active laser. For WaitUntilFinished, 0 (False) = don't wait. Returning True = Successful.  
 Controller Types: 4006, 4106, 2210, 2012  
 Example: `SetDesiredPower(Obj, 13, 0);`  
 Returns: ans = true  
 Should see in LRCP screen->
- **bool SetPolarizationIn(classname);**  
 Description: Puts both polarization filters in. Returning True = Successful.  
 Controller Types: 2210  
 Example: `SetPolarizationIn(Obj);`  
 Returns: ans = true
- **bool SetPolarizationOut(classname);**  
 Description: Puts both polarization filters out. Returning True = Successful.  
 Controller Types: 2210  
 Example: `SetPolarizationOut(Obj);`  
 Returns: ans = true
- **bool SetReceiverOff(classname);**  
 Description: Turns the receiver off in the active controller. Returning True = Successful.




---

**CAUTION.** *Ensure laser power is reduced to zero before running this command, otherwise the photoreceiver could be damaged.*

---



---

**NOTE.** *To turn off lasers, click the LRCP software Laser Emission button to **Off**, or press the OM2012 front-panel **Power** button to power off the instrument.*

---

Controller Types: 4006, 4106  
 Example: `SetReceiverOff(Obj);`  
 Returns: ans = true

- **bool SetReceiverOn(classname);**  
 Description: Turns the receiver on in the active controller. Returning True = Successful.  
 Controller Types: 4006, 4106  
 Example: `SetReceiverOn(Obj);`  
 Returns: ans = true
- **void SetDesiredFrequency(double\_desiredFrequency);**  
 Description: Sets the channel and fine tune frequency of the currently selected laser to the specified frequency.  
 Controller Types: 2210, 2012
- **void TogglePolarization(classname);**



Description: Toggles the polarization state by moving both filters to the opposite position.

Controller Types: 2210

Example: `TogglePolarization(Obj);`

## The OUI4000 ATE interface

The Automated Test Equipment (ATE) interface exposes the OUI4006 user interface functionality through one of two types of Windows Communication Foundation (WCF) services. The WCF services (referred to as advanced and basic) are available on port 9200 in the machine that is running the OUI4006.

---

**NOTE.** *Because of an optimization in the OUI4006 application, when you attempt to read variables from MATLAB through this interface you must have the related plots displayed in the OUI4006 application or the values will not be calculated.*

---

### OUI setup for ATE

There is some setup that is done in the OUI4006 application in preparation for an ATE application. It is necessary to add function calls that perform additional calculations. Variables used in these function calls and shared with the ATE application should be declared as “global” in the MATLAB Engine Command window as shown below with the variable “ChOffset”:

The standard variables produced by CoreProcessing and any additional declared variables will be repopulated each time a single is executed. After the data record is acquired from the scope, the commands that are in the MATLAB Engine Command window will be executed using that data. The variables used in these executed commands will only be available until the next single acquisition occurs. The ATE application should save the global variables to local storage for processing before executing the next single acquisition. This process is synchronized via a callback method or .NET event that is sent to the ATE application when the processing of an acquisition is completed and the variables are ready for storage and processing.

Thus the Data/Variable process works like this:

1. A Single is executed
2. Record/Block data is retrieved from the scope and saved to vBlock
3. All commands in the MATLAB Engine Command window are executed against the acquired data record and all variables, both standard and global, are populated
4. End of Block is triggered for each multiple of block size less than record size
5. End of Record is triggered

The ATE application should implement a callback handler for each event, block and/or record, it is interested in. In general, ATE applications will probably have the block and record size equal so the block and record events will be one for one. There are some additional summary variables that are populated at the end of the record so it is best practice to attach to the Record Event when block size is  $\geq$  record size. The example of this implementation is described in *Basic Method for getting MATLAB variable values* (See page 169, *Building an OM4006 ATE client in VB.NET*.)

**Basic and advanced WCF service interfaces for the OUI4006**

The basic service, implemented using a wsBasicHTTPBinding, exposes a smaller subset of commands. It was implemented using a simpler binding for compatibility with applications like MATLAB or LabView that only support the wsBasicHTTPBinding. (See page 167, *ATE functionality in MATLAB*.)

The basic service can be referenced at the following URL:  
<http://localhost:9200/Optametra/OM4006/WCFServiceOM4006Basic/>

The advanced service, implemented using a wsHTTPBinding and not available in MATLAB, uses events to provide a time-efficient interface. This service, which is only visible when the OIU4006 application is run as administrator, has two components which reside at the following URLs:

<http://localhost:9200/Optametra/OM4006/WCFServiceOM4006/>

<http://localhost:9200/Optametra/OM4006/WCFServiceOM4006Bulk/>

A wrapper DLL (OM4006ATEClient.DLL) has been supplied to simplify the interface to these services. It is highly recommended that you use this DLL which is designed to deal with the handshaking that is associated with working with events instead of interfacing directly to the service. Appendix F explains how to reference this DLL in your client ATE Application. (See page 169, *Building an OM4006 ATE client in VB.NET*.)

---

**NOTE.** *For safety reasons neither the basic or advanced services support the activation of a laser; this must be done on the user interface.*

---

**OUI basic and advanced service interface function list**

Refer to *The OM4000 user interface OUI* (See page 23.) for more information on the OUI functionality of these functions both as exposed by the simple http binding and the Client DLL. The following are the available commands in both the basic and advanced service interfaces. These are the only OUI functions that are available through the MATLAB interface.

---

**NOTE.** *All functions can generate an exception if there is an error. This fact should be used, especially with functions that return void, to verify the correct operation of your ATE program or script. For example, use the try-catch statement in MATLAB to define how certain errors are handled.*

---

- **uint GetBlockSize(classname);**  
Description: Returns the current block size from the OUI as an unsigned integer.  
Example: `GetBlockSize(Obj);`  
Returns: `ans = 50000`
- **uint GetRecordLength(classname);**  
Description: Returns the current record length from the OUI as an unsigned integer.  
Example: `GetRecordLength(Obj);`  
Returns: `ans = 1000`
- **void SetBlockSize(classname, uint newBlockSize);**  
Description: Sets the desired block size in the OUI as an unsigned integer for the next acquisition.  
Example: `SetBlockSize(Obj, 2000);`
- **void SetRecordLength(classname, uint newRecordLength);**  
Description: Sets the desired record length in the OUI as an unsigned integer for the next acquisition.  
Example: `SetRecordLength(Obj, 10000);`
- **uint GetNumberOfAcquisitions(classname);**  
Description: Returns the number of acquisition records taken by the OUI as an unsigned integer.  
Example: `GetNumberOfAcquisitions(Obj);`  
Returns: `ans = 3578`
- **uint GetNumberOfProcessedAcquisitions(classname);**  
Description: Returns the number of processed acquisition records taken by the OUI as an unsigned integer.  
Example: `GetNumberOfProcessedAcquisitions(Obj);`  
Returns: `ans = 1357`
- **bool IsRunning(classname);**  
Description: Returns a boolean flag of the scope interface state; True = Acquisition is running  
Example: `IsRunning(Obj);`  
Returns: `ans = true`
- **bool IsProcessing(classname);**  
Description: Returns a boolean flag of the scope data state; True = Data is being processed  
Example: `IsProcessing(Obj);`  
Returns: `ans = true`
- **bool IsAcquiring(classname);**  
Description: Returns a boolean flag of the scope interface state; True = Data is being acquired from the scope  
Example: `IsAcquiring(Obj);`  
Returns: `ans = true`

- **bool IsCancelling(classname);**  
Description: Returns a boolean flag of the scope interface state; True = Acquisition is being cancelled  
Example: `IsCancelling(Obj);`  
Returns: `ans = true`
- **bool IsReadyForSingle(classname);**  
Description: Returns a boolean flag of the application state; True = Application is ready for another single command  
Example: `IsReadyForSingle(Obj);`  
Returns: `ans = true`
- **bool IsScopeConnected(classname);**  
Description: Returns a boolean flag of the scope interface state; True = Application is connected to a scope  
Example: `IsScopeConnected(Obj);`  
Returns: `ans = true`
- **string GetScopeIDN(classname);**  
Description: Returns a string containing the scope identifier  
Example: `GetScopeIDN(Obj);`  
Returns: `ans = 'TESTER'`
- **string GetScopeAddress(classname);**  
Description: Returns a string containing the scope IP address  
Example: `GetScopeAddress(Obj);`  
Returns: `ans = '192.168.117.0'`
- **double GetLOFreq(classname);**  
Description: Returns a double containing the LO frequency  
Example: `GetLOFreq(Obj);`  
Returns: `ans = 191.5`
- **void SetLOFreq(classname, double loFreq);**  
Description: Sets the desired LO Frequency as an double in the OUI for the next acquisition.

---

**NOTE.** *This value may be overridden if the OUI retrieves the value from the LRCP.*

---

Example: `SetLOFreq(Obj, 193.5);`

- **void Connect(classname, string Address);**  
Description: Connects the OUI to the specified scope having the valid VISA address.  
Example: `Connect(Obj, );`  
Returns: `ans = 'TESTER'`
- **void Disconnect(classname);**  
Description: Disconnects the application from the currently connected scope.

Example: `Disconnect(Obj);`

- **void single(classname, int timeoutInMilliseconds);**  
Description: Performs a single acquisition within the timeout period.  
Example: `Single(Obj, 3000);`
- **void DCCalib(classname);**  
Description: Performs a basic DC calibration of the scope channels.  
Example: `DCCalib(Obj);`

### OUI advanced service interface function list

Refer to *Analysis parameters* for more information on the OUI functionality of these functions as exposed by the Client DLL. (See page 32, *Analysis Parameters window*.) Refer to *Building an OM4006ATE client in VB.NET* for examples of using these functions in an ATE client application. (See page 169, *Building an OM4006 ATE client in VB.NET*.)

The following commands are available only in the advanced service interface:

- **public AnalysisParameters GetAnalysisParameters()**  
Description: Returns a analysis parameters object containing the OUI variables.
- **void SetAnalysisParameters(AnalysisParameters analysisParameters)**  
Description: Sets the analysis parameters to new values.
- **double GetDouble(string vname)**  
Description: Returns the value of the passed variable as a double.
- **bool GetBoolean(string vname)**  
Description: Returns the value of the passed variable as boolean.
- **bool[] GetArrayOfBoolean(string vname)**  
Description: Returns the value of the passed variable as an array of boolean.
- **double[] GetArrayOfDoubles(string vname)**  
Description: Returns the value of the passed variable as an array of doubles.
- **double GetComplexImaginary(string vname)**  
Description: Returns the imaginary part of the specified complex number.
- **double GetComplexReal(string vname)**  
Description: Returns the real part of the specified complex number.
- **double[] GetArrayOfComplexImaginary(string vname)**  
Description: Returns an array of imaginary double values of the specified array of complex numbers.
- **void RegisterForBlockUpdate(MATLABVariable variable)**  
Description: Registers a MATLAB variable of interest for updating by MATLAB at the end of block processing.
- **void RegisterForRecordUpdate(MATLABVariable variable)**

Description: Registers a MATLAB variable of interest for updating by MATLAB at the end of record processing.

- **void RegisterForBlockUpdate(List<MATLABVariable> variables)**  
Description: Registers a list of MATLAB variables of interest for updating by MATLAB at the end of block processing.
- **void RegisterForRecordUpdate(List<MATLABVariable> variable)**  
Description: Registers a list of MATLAB variables of interest for updating by MATLAB at the end of record processing.
- **void RegisterForBlockUpdate(string variableName)**  
Description: Registers a variable for the block update by variable name.
- **void RegisterForRecordUpdate(string variableName)**  
Description: Registers a variable for the record update by variable name.
- **string SendMATLABCommand(string MATLABCommand)**  
Description: This command is used to execute a MATLAB statement.

---

**NOTE.** *this can only be used when the normal CoreProcessing is disabled or there will be contention with the OUI MATLAB engine. See MATLAB section for usage.*

---

- **void ExecuteMATLABCommands(string commandsToexecute)**  
Description: This command is used to execute a block of MATLAB statements.

---

**NOTE.** *this can only be used when the normal CoreProcessing is disabled or there will be contention with the OUI MATLAB engine. See MATLAB section for usage.*

---

- **void SetBlockCommands(string blockCommands)**  
Description: Sets a string that has one or more MATLAB commands that get run before block processing occurs.
- **void SetRecordCommands(string recordCommands)**  
Description: Sets a string that has one or more MATLAB commands that get run before record processing occurs.
- **void BlockProcessed(BlockProcessedEventArgs eventArgs)**  
Description: Event Trigger that is executed when the OUI4006 application completes processing of a block. You attach your event handler to this delegate and it is executed when the event occurs.
- **void RecordProcessed(RecordProcessedEventArgs eventArgs)**  
Description: Event Trigger that is executed when the OUI4006 application completes processing of a record. You attach your event handler to this delegate and it is executed when the event occurs.

- **string GetScopeState()**  
Description: Returns a string containing a text description of the scope's state. Valid states are: Unknown, Disconnected, Connected, Acquiring  
laserusagetype exposelaserusage() Exposes the laser usage type to the OUI interface.
- **bool ScreenShot(string filename)**  
Description: Takes a snapshot of what the OUI has on the display, exactly like doing a PrintScreen, and saves it the specified file.

## ATE functionality in MATLAB

MATLAB supports a limited subset of the OM4000 Series services, namely the Basic service. This section describes how to create and address the functions from MATLAB.

### LRCP control

The Laser/Receiver Control Panel communicates with other programs using port 9000 on the computer running the Control Panel software. MATLAB 2009a has a built-in capability that makes control from MATLAB easy if you are running the February 2010 or later release of the Laser/Receiver Control Panel.

---

**NOTE.** *Ensure that the lrcp is running before using this interface.*

---

Initialize the interface in the MATLAB desktop command window with the following commands:

```
url =
'http://localhost:9000/LaserReceiverControlPanel/Laser_ReceiverServiceBasic.svc?wsdl';
createClassFromWsd1(url);
obj = Laser_ReceiverServiceBasic;
```

Where:

The first specifies the URL or path to a WSDL application programming interface (API) that defines the web service methods, arguments, and transactions for the LRCP.

The second creates the new class based upon that API and builds a series of M-Files for accessing the Laser/Receiver Control Panel service.

The third instantiates the object class name and opens a connection to the service.

These commands only need to be run anytime the service interface (available methods) changes.

To get an up-to-date listing of methods for the service, type the following:

`methods(obj)`

Matlab should return the same functions (See page 168, *OUI control in MATLAB.*) and any new functions that have been added. These functions are self-documented when they are generated. By enabling the MATLAB help window, you can find out the function's parameters by typing the function name followed by a "(" and waiting for the help to display.

## OUI control in MATLAB

The OM4000 OUI software communicates with other programs via port 9200 on the computer running the OUI software.

---

**NOTE.** *Ensure that the OUI is running before using this interface.*

---

To get an up-to-date listing of methods for the OUI basic service type the following: `methods(obj)` MATLAB should return the same functions (See page 168, *OUI control in MATLAB.*) and any new functions that have been added. These functions are self-documented when they are generated.

By enabling the MATLAB help window, you can find out the function's parameters by typing the function name followed by a "(" and waiting for the help to pop up.

Initialize the interface in the MATLAB desktop command window with the following commands:

```
url =  
'http://localhost:9200/Optametra/OM4006/WCFServiceOM4006Basic/?wsdl';  
createClassFromWsd1(url);  
obj = WCFServiceOM4006Basic;
```

Where:

The first specifies the URL or path to a WSDL application programming interface (API) that defines the web service methods, arguments, and transactions for the OUI web service.

The second creates the new class based upon that API and builds a series of M-Files for accessing the OUI basic service.

The third instantiates the object class name and opens a connection to the OUI basic service.

These commands only need to be run anytime the service interface (available methods) changes.

To get an up-to-date listing of methods for the service, type the following:

`methods(obj)`



Matlab should return the same functions (See page 168, *OUI control in MATLAB.*) and any new functions that have been added. These functions are self-documented when they are generated. By enabling the MATLAB help window, you can find out the function's parameters by typing the function name followed by a “(“ and waiting for the help to display.

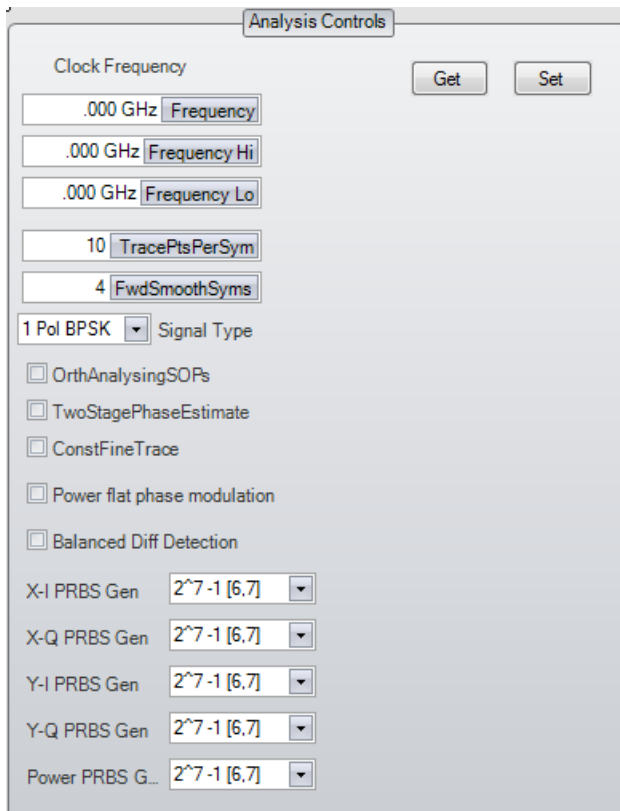
## Building an OM4006 ATE client in VB.NET

The following document explains how to build an OM4006 ATE Client application in VB.NET using the OM4006ATEClient .NET Assembly provided with the OUI4006 ATE Toolkit. The ATE interface to OUI4006.EXE is implemented using several WCF Services. There are actually two ways that a ATE client application can interface to the WCF Services. One is to create a service reference in a VB Project that talks directly to the WCF Service. The other is to add a reference to the OM4006ATEClient .NET assembly. The preferred method is to use the OM4006ATEClient .NET assembly and this document will explain how to implement that method.

### WCF service background

The OUI4006 application exposes functionality using several WCF services. The OM4006ATEClient assembly uses two of those WCF services (See page 162, *Basic and advanced WCF service interfaces for the OUI4006.*)to convey detailed processing information and expose a variety of control to the ATE Client application. WCFServiceOM4006 exposes most of the functionality, including asynchronous events. WCFServiceOM4006Bulk exposes bulk data methods to retrieve large arrays of data. Aside from the information about the location of the services (added to the APP.Config file, see below) the client that uses the OM4006Client.NET assembly really doesn't need to worry about any of the details of the services.

The following user control (ucAnalysisParameters) is provided in OM4006ATEClient for use in ATE client applications.



**Add service references in APP.CONFIG file of the ATE client application**

The OM4006Client assembly will need to know where to look for the WCF Services. Those services can reside on the local machine or on a remote computer. The information is used by the OM4006ATEClient assembly to establish a connection to the host OUI4006 application.

Please note the “localhost” below. If the computer is not local then replace ”localhost” with the name of the remote machine, for example “DavesAsus”. Adding the XML below to your APP.CONFIG file will define two service references running on the local machine.

```
<system.serviceModel>
<bindings>
  <basicHttpBinding>
    <binding name="BasicHttpBinding_IWCFServiceOM4006Bulk"
      closeTimeout="00:10:00" openTimeout="00:10:00"
      receiveTimeout="00:10:00" sendTimeout="0010:00"
      maxBufferSize="1000000000"
      maxBufferPoolSize="1000000000"
      maxReceivedMessageSize="1000000000"
      transferMode="Streamed"
      useDefaultWebProxy="true">
      <readerQuotas maxDepth="1000000000"
```

```

        maxStringContentLength="1000000000"
        maxArrayLength="1000000000"
        maxBytesPerRead="1000000000"
        maxNameTableCharCount="1000000000" />
    </binding>
</basicHttpBinding>
<wsDualHttpBinding>
  <binding name="WSDualHttpBinding_IWCFServiceOM4006"
    closeTimeout="00:10:00" openTimeout="00:01:00"
    receiveTimeout="00:10:00" sendTimeout="00:10:00"
    bypassProxyOnLocal="false" transactionFlow="false"
    hostnameComparisonMode="StrongWildcard"
    maxBufferSize="524288"
    maxReceivedMessageSize="65536" messageEncoding="Text"
    textEncoding="utf-8" useDefaultWebProxy="true">
    <readerQuotas maxDepth="32"
      maxStringContentLength="8192"
      maxArrayLength="16384" maxBytesPerRead="4096"
      maxNameTableCharCount="16384" />
    <reliableSession ordered="true"
      inactivityTimeout="00:10:00" />
    <security mode="Message">
      <message clientCredentialType="windows"
        negotiateServiceCredential="true"
        algorithmSuite="Default" />
    </security>
  </binding>
</wsDualHttpBinding>
</bindings>
<client>
  <endpoint
    address="http://localhost:9200/Optametra/OM4006/
    WCFServiceOM4006/"
    binding="wsDualHttpBinding"
    bindingConfiguration="WSDualHttpBinding_IWCFService
    OM4006"
    contract="WCFServiceOM4006.IWCFServiceOM4006"
    name="WSDualHttpBinding_IWCFServiceOM4006">
    <identity>
      <dns value="localhost" />
    </identity>
  </endpoint>
  <endpoint
    address="http://localhost:9200/Optametra/OM4006/
    WCFServiceOM4006Bulk/"
    binding="basicHttpBinding"
    bindingConfiguration="BasicHttpBinding_IWCFService

```

```

        OM4006Bulk"
        contract="WCFServiceOM4006Bulk.IWCFServiceOM4006Bulk"
        name="BasicHttpBinding_IWCFServiceOM4006Bulk">
        <identity>
        <dns value="localhost" />
        </identity>
    </endpoint>
</client>
</system.serviceModel>

```

### OM4006ATEClientNET assembly

This assembly includes a basic interface for retrieving MATLAB variable values and an object oriented interface that includes specialized .NET classes for all of the OM4000 instrument specific variables. This interface allows a client application to read and set all analysis parameters such as Block Size and Record Length. It also has the ability to connect or disconnect to/from a scope based on IP address as well as trigger single acquisitions. Additionally, the application can register for events that occur at the end of blocks and records.

To get started add the following (highlighted in yellow) reference to your ATE Client application. The file can be found in the folder where OM4000 OUI is installed.

Once the reference is added it can be browsed to understand the available functionality. Double click on the reference to bring up the Object browser.

Scroll down from Optametra.OM4006.OM4006ATEClient and double-click on OM4006ATEClient:

The right pane on the object browser should look like the following image:

### Basic method for getting MATLAB variable values

The following is an example code block that references the OM4006 Client, allocates a new object, registers the BER variable for the block update, connects to a scope, acquires a single block and saves it to a local variable. This method is straight forward but requires more code to extract values to local variables. It allows for simpler implementation of code to extract customer defined variables. It requires that case sensitive strings be passed to the Register and Get methods as shown in the following code.

```

Imports Optametra.OM4006.OM4006ATEClient
Imports System.IO
Imports System.Xml.Serialization
Imports System.Collections.Generic
Public Class SampleATEApplication
    DIM BER as double
    Dim WithEvents MyOM4006ATEClient
        AS OM4006ATEClient = New OM4006ATEClient()
    Dim totalBits As Double
    Dim totalBitsUI as double

```

```

Dim processingDone As ManualResetEvent = New
ManualResetEvent(false)
' this event handler will register the "BER" variable
' for update at the end of a block, connect to a scope
' and do a single acquisition

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs)
Handles MyBase.Load
    My4006ATEClient.RegisterForBlockUpdate("BER")
    My4006ATEClient.Connect(TCPIP0::172.17.200.111::
inst0::INSTR")
    My4006ATEClient.Single()
    processingDone.WaitOne(30000)
    ' wait for the event to trigger
    ' put code here to make use of the variables
    ' that are populated in the event handler
    ' before issuing more singles.
End sub
' callback that executes at the end of each block.
' process all variables that have been registered
' in this method.
' values returned from the Get methods outside of
' this method will potentially be corrupt.

Private Sub EndOfBlockEvent(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyOM4006ATEClient.BlockVariablesPopulated
    totalBits =
    MyOM4006ATEClient.GetDouble("BER.TotalBits")
    totalBitsUI =
    MyOM4006ATEClient.GetDouble("BER.TotalBitsUI")
    processingDone.Set()
' signals main thread; event handled
End Sub
End Class

```

### Object oriented method for getting MATLAB variable values

The OM4006ATEClient comes with specialized classes for accessing all of the MATLAB variables that are created by OM4000 instrument software. They are developed from a base set of MATLABVariable\* classes. The following is an example code block that references the OM4006 Client, allocates a new object, registers the BER variable for the block update, connects to a scope, acquires a single block and saves it to a local variable.

Use of this method requires a bit more knowledge of object oriented development and the use of classes. It requires less code to implement and it deals with the string case sensitivity within the specialized variable classes so it is less prone to error.

```
Imports Optametra.OM4006.OM4006ATEClient
Imports System.IO
Imports System.Xml.Serialization
Imports System.Collections.Generic
Public Class SampleATEApplication
    Dim myBER As BER = New BER()
    Dim totalBits As Double
    Dim processingDone As ManualResetEvent = New
    ManualResetEvent(false)
    Dim WithEvents MyOM4006ATEClient
    As OM4006ATEClient = New OM4006ATEClient()

    ' this event handler will register the "BER" variable
    ' for update at the end of a block, connect to a scope
    ' and do a single acquisition.

    Private Sub Form1_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load
        My4006ATEClient.RegisterForBlockUpdate(myBER)
        My4006ATEClient.Connect(TCPIP0::172.17.200.111::
        inst0::INSTR")
        processingDone.Reset()
        My4006ATEClient.Single()
        processingDone.WaitOne(30000)

        ' wait for the event to trigger
        ' put code here to make use of the variables that are
        ' populated in the event handler before
        ' issuing more singles
    End Sub

    ' callback that executes at the end of each block
    ' process all variables that have been registered
    ' in this method.
    ' Values returned from the Get methods outside
    ' of this method will potentially be corrupt.

    Private Sub EndOfBlockEvent(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles MyOM4006ATEClient.BlockVariablesPopulated

    'At this point all the variables that are registered
    ' for have been completely populated and they can just
```

```

' be referenced like any other .NET class

totalBits = myBER.TotalBits
processingDone.Set()
' signals the main thread that the event has been handled

End Sub
End Class

```

If there is a desire by the customer to access their custom variables using the second method they can develop their own classes by inheriting from the appropriate `MATLABVariable*`. The following shows a customer variable structure that has two fields for counting good and bad blocks. This variable will be automatically populated by the `OM4006Client` assembly if it is registered as shown above.

```

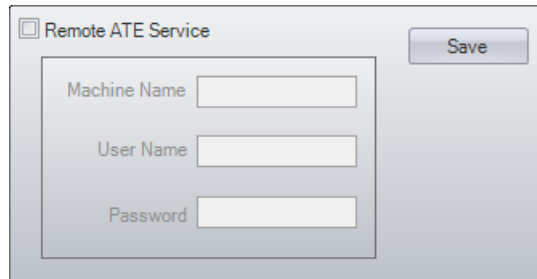
using System;
using System.Collections.Generic;
using System.Text;
using Optametra.OM4006;
namespace Optametra.OM4006.OM4006ATEClient
{
    [Serializable]
    public class CustomerVariable : MATLABVariableStructure
    {
        public BER()
        {
            this.MATLABVariableName = "MyCustomStructure";
            BadBlocks = new MATLABVariableDouble("BadBlocks");
            GoodBlocks = new MATLABVariableDouble("GoodBlocks");
            this.Variables.Add(BadBlocks);
            this.Variables.Add(GoodBlocks);
        }
        public MATLABVariableDouble BadBlocks { get; set;}
        public MATLABVariableDouble GoodBlocks {get; set;}
    }
}

```

### Local vs. remote hosting

The `OM4006ATEClient` can reside on the same machine as the server application (`OUI4006.exe`) or on a remote machine. The `APP.CONFIG` (explained above) comes configured for local hosting. “`LocalHost`” can be changed to a machine name or an IP address depending on your network's DNS support. If the ATE client application is not going to run on a machine other than the one the server application is on then the user must specify the logon information of a user on the remote machine.

The following user control (ucATESecurity) is provided in the OM4006Client. A password is required and is saved in an encrypted form in the App.Config.AppSettings along with the unscrambled user name and machine name. The information is accessed any time the application is started and “Remote ATE Service” is selected.



The image shows a user control window titled "Remote ATE Service". At the top left, there is a checkbox labeled "Remote ATE Service". To the right of the checkbox is a "Save" button. Below the checkbox, there is a rectangular container with three input fields: "Machine Name", "User Name", and "Password". Each field is represented by a text label followed by a rectangular input box.



---

# Appendix H: Cleaning and maintenance

## Cleaning

- To clean the outside of the OM4000 enclosure, use a dry, soft cotton cloth. Do not use any liquid cleaning agents or chemicals that could possibly infiltrate the enclosure, or that could damage markings or labels.
- If the dust filter on the underside of the unit becomes clogged, use a small vacuum or brush to clean the filter.
- From time to time it will be necessary to clean the optical input and output connectors on the front of the unit. Use square-ended swabs made for this purpose to clean each connector.
- Do not attempt to clean the inside of the instrument; cleaning of internal parts is not necessary.

## Maintenance

- There are no user-serviceable components or subsystems within the OM4000. Attempting any internal repairs will void your warranty. Never remove the external lid on the unit.



**WARNING.** *Removing the external lid and the internal cover on the optics package while the unit is operating will result in exposure to invisible laser radiation. Never view directly with optical instruments.*

---

- If it becomes necessary to replace the fuse in the power input module in the rear of the unit, use a 5X20mm “slo-blo” fuse rated at 1A, 250 VAC. Use a small screwdriver to gently pry open the fuse drawer.



**WARNING.** *Disconnect the unit from the power source when changing the fuse to ensure that line voltage is not present during the replacement.*

---



# Index

## Symbols and Numbers

2d Poincaré, 27  
2D Poincaré sphere, 50  
2nd Phase Estimate, 34  
3d constellation plot, 25  
3d Eye plot, 25  
3d Poincaré, 27

## A

Absolute power calibration, 119  
AC line voltage requirements, 5  
Accessories  
    optional, 2  
    standard, 2  
Add service references, 170  
advanced service interface  
    function list, 165  
Alert code descriptions, 109  
Alerts, 52  
Alerts management, 88  
Align signal tributaries with data  
    content, 84  
AlignTrib function, 91  
Analysis Parameters window  
    (ET), 145  
Apply Gray coding for QAM, 36  
Apply limiting function, 33  
Apply polarization & phase  
    estimates, 85  
ApplyPhase function, 94  
Assume Orthogonal  
    Polarizations, 33  
ATE  
    MATLAB functionality, 167  
    OUI advanced service  
        interface function  
        list, 165  
    OUI basic, advanced interface  
        services list, 162  
    OUI basic, advanced WCF  
        services, 162  
    OUI interface setup, 161

ATE functionality in  
    MATLAB, 167  
ATE interface, 155  
    add service references, 170  
    building an ATE client in  
        VB.NET, 169  
    service interface function  
        list, 155  
    WCF service, 155  
Auto-center filter on signal, 36  
Automatic receiver deskew, 123

## B

Balanced Differential Detection  
    (BDD), 35  
BER, 27  
Bit-Error-Rate reporting, 51  
Blk Size, 30  
Block processing, 87  
Block size and record length  
    interaction, 32  
Building an ATE client in  
    VB.NET, 169

## C

Calculate linear average eye, 36  
Calculate linear average vs.  
    time, 36  
Calculate signal vs. time on fine  
    time grid, 86  
Calculate subsequence  
    average, 36  
Calculate transition average, 36  
Calibration and adjustment  
    (ET), 132  
Calibration and adjustment  
    (RT), 111  
Cavity Lock (LRCP), 22  
Chromatic Dispersion, 37  
Cleaning and maintenance, 177  
Clock frequency (Nominal, Low,  
    High), 33  
Clock recovery, 83

ClockRetime function, 94  
Code descriptions, Alert  
    messages, 109  
Coherent detection, 77  
Coherent eye diagram, 26  
Color features, 44  
Colorgrade, 26  
Compensate CD, 37  
Configuring hardware (ET), 125  
Configuring OUI, 57  
Configuring the software  
    (ET), 128  
Configuring two  
    oscilloscopes, 147  
Connections list (ET), 126  
Constellation  
    color features, 44  
    diagrams, 41  
    measurements, 42  
    offset modulation formats, 43  
Constellation plot, 25  
Continuous trace points per  
    symbol, 36  
Continuous Traces, 35  
Controls and connectors  
    front panel, 17  
    rear panel, 18  
Controls panel  
    Blk Size, 30  
    Rec Len, 30  
    Run-Stop, 31  
    Scale controls, 31  
    Single, 31  
Core Processing function  
    reference, 91  
Core processing software  
    guide, 79  
Count bit errors, 86  
Current Signal Spectrum plot, 49  
Cutoff frequency, 36

## D

Data Content, 37

DC calibration (ET), 132  
DC calibration (RT), 111  
Decision-Threshold Q-Factor, 27  
Delay adjustment (system  
deskew), 133  
Delay adjustment (system deskew)  
(RT), 111  
Description  
  product overview, 1  
Deskew, 133  
Deskew (RT), 111  
Detailed configuration of  
  experiments, 77  
DiffDetection function, 95  
Direct assignment of pattern  
  variables, 39  
DSA8200, 127, 128  
DSA8300, 127, 128

## E

Environmental operating  
  requirements, 4  
Equipment setup  
  connections, 15  
  equivalent-time (ET)  
  oscilloscope, 14  
  equivalent-time (ET)  
  oscilloscopes, 142  
  real-time (RT)  
  oscilloscope, 13

Equivalent-time (ET) oscilloscope  
  Analysis Parameters  
  window, 145  
  calibration and  
  adjustment, 132  
  configure software, 128  
  configuring hardware, 125  
  connections, 126  
  DC calibration, 132  
  delay adjustment (system  
  deskew), 133  
  deskew, 133  
  equipment setup, 142  
  hybrid calibration, 137  
  MATLAB Engine file, 143  
  OUI, 130  
  OUI Controls panel, 144  
  receiver equalization, 140  
  setup diagram, 125  
  system deskew, 133  
  taking measurements, 144  
Equivalent-time (ET) oscilloscope  
  setup, 14  
EstimateClock function, 97  
EstimatePhase function, 99  
EstimateSOP function, 100  
Example  
  capturing unknown  
  pattern, 40  
  function that uses Alerts  
  variable, 89  
  save to unique file names, 53  
  saving intermediate data  
  sets, 53  
  trigger a save, 54  
Eye diagrams, 45

## F

Features, 1  
Filter order, 36  
Filter roll-off factor, 36  
Filter type, 36  
Frequency spectrum, 28  
Front end filtering, 37  
Front panel controls and  
  connectors, 17  
Front panel labels, x

## G

General safety summary, vi  
GenPattern function, 102

## H

Homodyne (RT only), 34  
Hosting (local vs. remote), 175  
Hybrid calibration, 137  
Hybrid calibration (RT), 115

## I

Important safety information, vi  
Incoming inspection, 4  
Initial phase estimate, 84  
Initial polarization estimate, 83  
Initial product inspection, 4  
Interaction with OUI, 79  
IP address setup, 8  
  DHCP-enabled network, 8  
  non-DHCP network, 9

## J

Jones2Stokes function, 103  
JonesOrth function, 103

## K

Key features, 1

## L

Large record length data,  
  managing, 53  
Laser phase noise spectrum, 28  
Laser Receiver Control Panel  
  (LRCP) user interface, 19  
Laser safety labels, x, xi  
LaserSpectrum function, 104  
Limiter threshold, 33  
Local vs. remote hosting, 175  
Lowpass filter, 33

**L**

- LRCP
  - ATE interface, 155
  - cavity lock, 22
  - connecting to OM instruments, 20
  - device setup and auto configure, 20
  - overview, 19
  - service interface function list, 155
  - setting laser parameters, 21
  - WCF service interface, 155

**M**

- Managing large record length data, 53
- Mask Threshold, 35
- MaskCount function, 101
- MATLAB, 62
  - ATE functionality, 167
  - Engine file, 64
  - Engine file (ET), 143
  - functions, 81
  - object oriented method for getting MATLAB variable values, 173
  - retrieving MATLAB variable values, 172
  - variables, 80
  - variables used by core processing, 107

**MCS**

- about the multicarrier spectrum plot, 70
- channel list, 67
- constellation plot, 72
- display layout, 68
- EVM vs. Channel, 74
- eye diagrams, 73
- Measurement vs. Channel, 75
- multicarrier spectrum plot, 70
- overview, 66
- plots, 68
- Q vs. Channel, 74
- setup window, 66
- spectrum plot, 69
- spectrum plot controls, 70
- spectrum plot explanation, 72
- spectrum plot menu, 69
- MCS measurements, 29
- Measurements Statistics table, 50
- Multicarrier
  - about the multicarrier spectrum plot, 70
  - constellation plot, 72
  - display layout, 68
  - EVM vs. Channel, 74
  - eye diagrams, 73
  - Measurement vs. Channel, 75
  - multicarrier channel list, 67
  - overview, 66
  - plots, 68
  - Q vs. Channel, 74
  - setup window, 66
  - spectrum plot, 69
  - spectrum plot controls, 70
  - spectrum plot explanation, 72
  - spectrum plot menu, 69
- Multicarrier channel list, 67
- Multicarrier support (MCS) option, 66

**N**

- Non-VISA oscilloscope connections (Scope Service Utility), 59
- Number of symbols in impulse response, 36

**O**

- Object oriented method for getting MATLAB variable values, 173
- Offset modulation formats, 43
- OM4000 user interface (OUI)
  - 2D Poincaré sphere, 50
  - Analysis parameters window, 32
  - assignment of pattern variables, 39
  - assignment of pattern variables (not using PRBS), 39
  - constellation diagrams, 41
  - constellation measurements, 42
  - Controls panel, 30
  - example: capturing unknown pattern, 40
  - eye diagrams, 45
  - front end filtering, 37
  - Home ribbon, 24
  - offset modulation formats, 43
  - overview, 23
  - plots and measurements summary, 25
  - Signal vs. Time, 46
  - waveform averaging, 47
- OM4006ATEClientNET assembly, 172
- Operating requirements, 4
- Optional accessories, 2
- Options
  - CC (two C-band lasers), 3
  - CL (one C-band, one L-band laser), 3
  - instrument, 3
  - international power cords, 3
  - LL (two L-band lasers), 3
  - software, 3
- OUI
  - 2d Poincaré plot, 27
  - 2D Poincaré sphere, 50
  - 3d constellation plot, 25
  - 3d eye plot, 25
  - 3d Poincaré plot, 27

- Analysis parameters
  - window, 32
- assignment of pattern variables, 39
- assignment of pattern variables (not using PRBS), 39
- ATE advanced service interface function list, 165
- ATE basic, advanced interface services list, 162
- ATE basic, advanced WCF services, 162
- ATE interface, 161
- ATE interface setup, 161
- BER, 27
- Bit-Error-Rate reporting, 51
  - capturing unknown pattern, 40
- coherent eye diagram, 26
- Colorgrade function, 26
- configuring, 57
- constellation diagrams, 41
- constellation plot, 25
- Controls panel, 30
- Decision-Threshold Q-Factor plot, 27
- eye diagrams, 45
- frequency spectrum plot, 28
- front end filtering, 37
- Home ribbon, 24
- MATLAB Engine file, 64
- MCS measurements, 29
- MCS Setup window, 66
- multicarrier support (MCS) option, 66
- offset modulation formats, 43
- overview, 23
- PDM plot, 28
- playback, 52
- plot, 28
- plots and measurements summary, 25
- PMD measurement, 52
- power eye plot, 26
- record, 52

- save to unique file names, 53
- saving intermediate data sets, 53
- Setting up your measurement, 63
- Signal vs. Time, 46
- taking measurements, 65
- trigger a save, 54
- using OUI with other receivers, 55
- waveform averaging, 47
- OUI Controls panel (ET), 144

## P

- PC requirements, 6
- Phase estimation time constant parameter (Alpha), 34
- Playback (OUI), 52
- PMD, 37
- PMD measurement, 52
- PMD plot, 28
- Polarization mode dispersion (PMD), 52
- Power cord option, 3
- Power eye, 26
- Power requirements, 5
- Product description, 1
- Pure phase modulation, 33

## Q

- QDecTh function, 104

## R

- Real-time (RT) oscilloscope
  - Absolute power calibration, 119
  - calibration and adjustment, 111
  - configuring two oscilloscopes, 147
  - DC calibration, 111
  - delay adjustment (system deskew), 111
  - deskew, 111
  - hybrid calibration, 115
  - laser linewidth factor, 120
  - receiver equalization, 120
  - setup diagram, 147
  - system deskew, 111
- Real-time (RT) oscilloscopes setup, 13
- Rear panel controls and connectors, 18
- Rear panel labels, xi
- Rec Len, 30
- Receiver equalization, 120
- Receiver equalization (ET), 140
- Record (OUI), 52
- Record length and block size interaction, 32
- Record length data, managing, 53
- Remote hosting, 175
- Requirements
  - AC line voltage, 5
  - environmental, 4
  - operating, 4
  - PC, 6
  - Power, 5
- Reset SOP Each Block (RT only), 33
- Retrieving MATLAB variable values, 172
- Run-Stop, 31

## S

- Scale controls, 31
- Scope Service Utility (SSU), 59
- Scope Service Utility (SSU) installation, 7

- Second phase estimate, 85
  - Service safety summary, viii
  - Set instrument IP address
    - DHCP-enabled network, 8
    - non-DHCP network, 9
    - overview, 8
  - Setting up your measurement, 63
  - Signal center freq, 35
  - Signal processing steps in
    - CoreProcessing, 82
  - Signal type, 33
  - Signal vs. Time, 46
  - Single, 31
  - Software installation
    - controller PC, 7
    - HRC, 7
    - LRCP, 7
    - oscilloscope, 7
    - OUI, 7
    - overview, 6
    - Scope Service Utility (SSU), 7
    - sequence, 7
    - SSU (for ET oscilloscope), 7
    - SSU (for RT oscilloscope), 7
    - TekVISA, 7
  - Software overview, 18
  - SSU (Scope Service Utility), 59
  - SSU (Scope Service Utility) installation, 7
  - Standard accessories, 2
  - Subsequence average length, 36
  - Symbol Center Width, 35
  - Symbols and terms on the product, ix
  - System deskew, 133
  - System deskew (RT), 111
- T**
- Taking measurements, 63
  - Taking measurements (ET), 144
  - Terms in this manual, ix
  - Time offset, 33
  - Tributaries contributing to average, 36
  - Two oscilloscope measurements (RT)
    - Master oscilloscope trigger settings, 149
    - OUI settings, 152
    - settings, 149
    - slave oscilloscope trigger settings, 151
- U**
- Using OUI with other receivers, 55
- V**
- VB.NET, 169
  - VISA connections, 57
- W**
- Waveform averaging, 47
  - WCF service background, 169
- Z**
- zSpectrum function, 106
- Two-oscilloscope configuration, 61