OM1106
Optical Modulation Analysis Software
Version 2.3.x

User Manual

Register now!
Click the following link to protect your product.
► **www.tek.com/register**

www.tek.com

077-1093-03

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.tek.com to find contacts in your area.

Table of Contents

List of Figures

List of Tables

Preface

This document describes how to install, configure, and operate the OM1106 Optical Modulation Analyzer (OMA) software version 2.3.x.

Related documents

Document	Tektronix part number
OM4245, OM4225 Optical Modulation Analyzer Installation and Safety Instructions	071-3414-xx
OM2210 Coherent Receiver Calibration Source Installation Safety Instructions	071-3050-xx
OM2012 nLaser Tunable Laser Source Installation and Safety Instructions	071-3154-xx
OM5110 46 GBaud Multi-Format Optical Transmitter Installation and Safety Instructions	071-3203-xx
Tektronix OM5000, OM4000, OM2000 Series Optical Modulation Instruments Declassification and Security Instructions	077-0992-xx
Avertissements - Mises en garde (manuels OM4245, OM4225, OM4106D, OM4006D, OM2210, OM2012 et OM5110)	071-3184-xx

Install software

The OM1106 Optical Modulation Analyzer software (referred to as OMA in this document) provides an ideal platform for research and testing of coherent optical systems. It offers a complete software package for acquiring, demodulating, analyzing, and visualizing complex modulated systems with an easy-to-use user interface. The software performs all calibration and processing functions to enable real-time burst-mode constellation diagram display, eye diagram display, Poincaré sphere display, and BER evaluation.

This section describes how to install and configure the OMA software and OM series instruments to correctly communicate with each other.

PC hardware and software requirements

The following are the PC requirements needed to install and run the OMA software to control the OM4000 and OM2000 series instruments. The term PC applies to a supported oscilloscope, PC, or laptop on which the OMA and other required software is installed, and that is connected to OM instruments over a local network.

Item	Description
Operating system	U.S.A. Microsoft Windows 7 or 10 (64-bit), with latest updates and service packs installed
Windows .NET version	4.51 or later (64-bit) ***NOTE.*** *The OMA install process updates the .NET software if 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 with 512+ MB minimum graphics memory ***NOTE.*** *OMA will run with video cards other than nVidia. However, color gradient display (for plots that support that feature) and some advanced plot features are only available when running OMA on an oscilloscope or PC that has an nVidia graphics card installed.* Download and install the latest drivers available from the video card manufacturer. There may be newer drivers available even if Windows says the drivers are up to date.
Networking	Gigabit Ethernet (1 Gb/s) or Fast Ethernet (100 Mb/s)
Display	20” minimum flat screen recommended for displaying multiple graph types

Item	Description
Other Hardware	2 USB 2.0 ports
Adobe Reader	Adobe reader used for viewing PDF format files

MATLAB® software requirements

The OMA software requires the appropriate version of the MathWorks, Inc. MATLAB® software for performing analysis. MATLAB is not included on the product software media. Please contact The MathWorks, Inc. to obtain the correct MATLAB version for your PC (www.mathworks.com).

OMA requires the following versions of MATLAB:

- For Microsoft Windows 7 or 10 (64-bit), U.S.A. version:
  MATLAB version 2011b or 2014a (Preferred version) or 2016a (64-bit)

NOTE. *OMA uses the most recently installed MATLAB software. If you need to revert to a previously installed Matlab, please see further instructions below.*

Set Windows 7 or 10 user account setting before installing software

Default Windows 7 or 10 user account settings interfere with OMA IVI/Visa operation. To fix this, do the following before installing any software:

- Click **Start > Control Panel > User Accounts**.
- Click **Change User Account Control settings** and set the notify control to **Never notify**.

Required software to install

Software required on the controller PC

Install the following software on the controller PC in the order listed.

NOTE. *Install all software as an Administrator.*

CAUTION. *Do not insert the product USB HASP key when installing the following software. Only install the HASP key after all software is installed and you are ready to run the OMA software.*

1. MATLAB: Required for OMA, must be installed and activated before installing OMA. (See page 3.)
2. TekVISA: Required when using Tektronix MSO/DSO70000 or 70000B series oscilloscopes.

NOTE. *Do not load TekVISA for MSO/DSO70000C/D/DX/SX series real-time (RT) oscilloscopes. These oscilloscopes use the Scope Service Utility (SSU).*

3. Power meter software: Required to run OMA Receiver Test functions (RXTest) in conjunction with the OM2210.
4. OM1106 (OMA) software.

Software required on the oscilloscope

If you are using a Tektronix MSO/DSO70000C/D/DX/SX series real-time (RT) oscilloscope, then you need to install the Scope Service Utility (SSU) on the oscilloscope.

To install MATLAB software

The OMA software requires the appropriate version of the MathWorks, Inc. MATLAB® software to perform analysis on the acquired data. MATLAB is not included on the product USB software media. Please contact The MathWorks, Inc. to obtain the correct MATLAB version for your PC (www.mathworks.com).

OMA requires the following versions of MATLAB:

- For Microsoft Windows 7 or 10 (64-bit), U.S.A. version:
  MATLAB version 2011b or 2014a (Preferred version) or 2016a (64-bit)

Follow the instructions provided with MATLAB to install, activate, and verify the software opens and runs.

NOTE. *MATLAB must be installed and activated before installing OMA. Do not install the rest of the software until you have confirmed that MATLAB runs and is activated.*

To install TekVISA software

***NOTE.** Only install TekVISA when you are using Tektronix MSO/DSO70000B, 70000, or earlier series oscilloscopes.*

TekVISA is required when using the OMA software with Tektronix MSO/DSO70000B, 70000, or earlier series oscilloscopes. If used, make sure to install TekVISA on the controller PC before installing the OMA software.

TekVISA is not included on the product USB flashdrive software media.

To download and install the correct version of TekVISA:

1. Go to **www.tek.com/downloads**.
2. Enter **tekvisa** in the search field, select **Software** in the download type field, and click **GO**.
3. Click **TEKVISA CONNECTIVITY SOFTWARE, V4.0.4**. Follow on-screen instructions to download the software file.
4. Copy the downloaded file to the controller PC (where the OMA software will be installed).
5. Double-click the TekVISA install file. Follow any on-screen instructions.

To install power meter software (for RXTest and OM2210)

The power meter software and drivers enable communication with the instrument optical power meter. This software is only required to run the OMA Receiver Test (RXTest) measurements with an OM2210 instrument.

***NOTE.** The power meter software only runs on Windows 7 or 10 (64-bit) installations. Windows 7 (32-bit) installations are not supported at this time.*

To install the power meter software:

1. Insert the product software media USB drive into a USB port on the controller PC (where the OMA software will be installed).
2. Navigate to the following file (from the USB root drive): **ThorLabsSoftware\PM100x_Instrument_Driver_64bit_V3.1.0.zip**
3. Double-click the file to display the compressed file contents.

***NOTE.** Windows 7 comes with a .zip-file-compatible compression-decompression program. If you cannot access the contents of the .zip file, check that the .zip file type is associated with the decompression program.*

4. Double-click **setup.exe** to install the power meter software. Follow any on-screen instructions.

To install OMA software

***NOTE.** The OMA software is labeled as OUI4006 in the screens and menu paths displayed by the installer.*

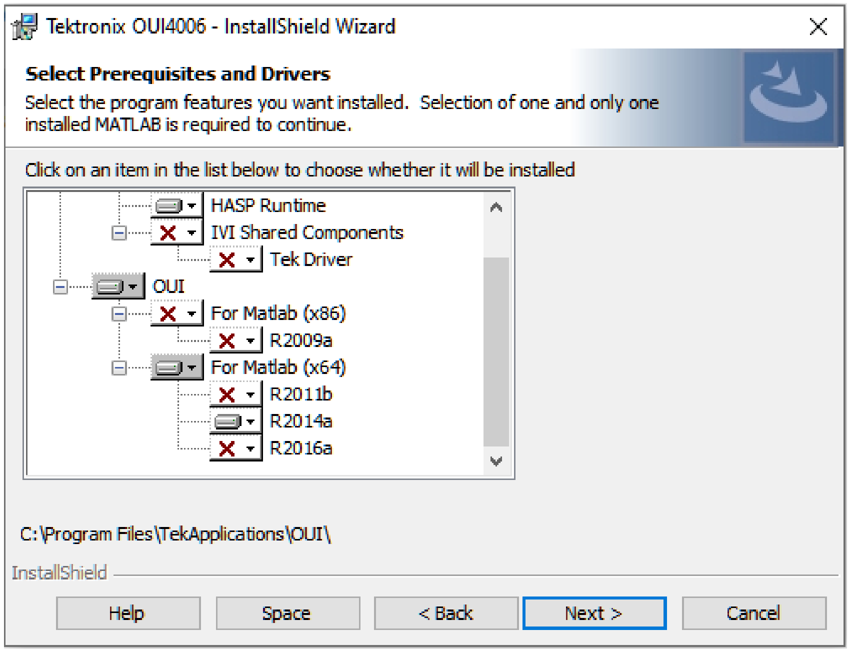
***NOTE.** Do **not** plug the OMA USB HASP key into the controller PC before installing OMA. The required HASP and related drivers must be loaded as part of the OMA install before you can use the USB HASP key.*

To install the OMA software:

1. Insert the OMA software media USB drive into a USB port on the controller PC.
2. For Windows 7 or 10 (64-bit) installations, navigate to the OMA software installation file (from the USB root drive):
   - **SetupOUI_x.x.x.x.exe**

   ***NOTE.** A 64-bit version of MATLAB 2011b, 2014a or 2016a must be installed and activated before installing the 64-bit version of the OMA software.*

3. Double-click the appropriate program file to begin the install and open the InstallShield Wizard.
4. Accept the license agreement and select the default options until you get to the choice for Complete or Custom. Select **Custom** to open the Select Prerequisites and Drivers dialog box.


Tektronix OUI4006 - InstallShield Wizard

Select Prerequisites and Drivers

Select the program features you want installed. Selection of one and only one installed MATLAB is required to continue.

Click on an item in the list below to choose whether it will be installed

- HASP Runtime
- ✗ IVI Shared Components
  - ✗ Tek Driver
- OUI
  - ✗ For Matlab (x86)
    - ✗ R2009a
  - For Matlab (x64)
    - ✗ R2011b
    - R2014a
    - ✗ R2016a

C:\Program Files\TekApplications\OUI\

5. Examine the Select Prerequisites and Drivers installation list to verify that the installer has properly identified the correct MATLAB version to use with the OMA. In the above example, the installer has identified that MATLAB R2014a is installed and will be used with the OMA.

   If there are multiple Matlab versions, use this list to disable all versions of MATLAB except the one that is required for your Windows OS.

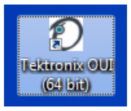
NOTE. *For future partial updates or re-installs, turn off all of the Subordinate Installation items except for the ones you are updating.*

---

6. Click **Next** to begin installation. The Install Wizard launches individual installers as required, such as for the HASP and IVI software.
7. Select **Finish** when the installer completes successfully.

The OMA desktop icons. The install program adds two OMA application icons to the desktop, which start different versions of OMA:

-

  Tektronix OUI (xx-bit) - Vertex Processing: This version uses your Graphics Processing Unit (GPU) to add features and enhance performance of the User Interface. The required minimum OpenGL version is 2.1.0 which most recent PCs support. If the graphics driver is out of date, a prompt to install latest driver may appear. If the GPU present is the recommended nVidia type, color-grade features will be enabled.
- 

  Tektronix OUI (xx-bit): This version is for older computers that lack support for Open GL 2.1.0 and for which no driver is available. This version disables some features including 3-D plots, Signal-vs-Time, and color grade options.

To determine if color shading and 3-d plots work on your controller PC, see the verify installation procedure. (See page 9, *Verify software installation*.)

To install Scope Service Utility (SSU) software

The Scope Service Utility (SSU) is required for OMA to communicate with Tektronix MSO/DSO70000C/D/DX/SX series real-time (RT) oscilloscopes. The SSU is installed and runs on the target oscilloscope to collect and send data to the OMA.

To install the SSU:

1. If SSU is installed on the oscilloscope, uninstall the current version before installing the new version.
2. Insert the OMA software media USB drive into a USB port on the **oscilloscope**.
3. Navigate to the SSU software installation file (from the USB root directory) on the MSO/DSO70000C/D/DX/SX RT oscilloscope:
   - **OUI\Tektronix Scope Service Utilityx.x.x.x.exe**
4. Double-click the appropriate program file to install. Follow any on-screen instructions. The installer adds SSU icons on the oscilloscope desktop.

Using the SSU. Double-click the **Tektronix Scope Service Utility** icon to start the SSU software on the oscilloscope before using the OMA to acquire data and analyze results. You can also drag the SSU icon to the Startup folder on real-time oscilloscopes so that the SSU starts automatically when you power-on or reboot the instrument. The *Non-VISA oscilloscope connections (Scope Service Utility)* section has more information on SSU. (See page 35, *Non-VISA oscilloscope connections (Scope Service Utility)*.)

Updating existing OMA (OUI) installations

Do the following before replacing an earlier version of OMA (referred to as OUI in earlier releases):

- Back up any critical data files; in particular, back up pHybCalib.mat and EqFiltCoef.mat from the C:\Program Files\TekApplications\OUI\ folder, and replace them in the new ExecFiles folder after the upgrade is complete.
- Back up any files you may have edited in the \Program Files\Optametra folder or \Program Files\TekApplications sub folders.
- Record your channel delay values on the Calibration tab. If upgrading from an older OUI version, the format of the Channel Delay sliders may be different. Starting with OUI V1.6, the top slider is the Channel XI to XQ delay, the middle slider is Channel XI to YI and the bottom is now Channel YI to YQ.

  If you have the XI to YQ value but not the YI to YQ value, simply subtract the XI to YI value from the XI to YQ value to get the YI to YQ value.

NOTE. *Please call Tektronix for assistance if upgrading from OUI V1.4 or V1.5 that is installed on an oscilloscope.*

- Do the following to upgrade from OUI4006 Version 1.3 or earlier to the current software, when installed on a PC (not an oscilloscope). Using the Remove Program tool in the system Control Panel to:
  - Uninstall the Scope specific IVI driver (for example, TekScope IVI Driver 2.7).
  - Uninstall the IVI Shared Components (do NOT uninstall the VISA Shared Components).
  - Uninstall OUI4006 (version 1.3 or earlier).
  - Uninstall LRCP (if present) by using the remove program feature in the system Control Panel.
  - Install this version of OMA and required software.
- If the OUI installer asks to uninstall an old HASP driver and install a new one, click **YES**. This prevents future problems with Windows Updates.

Verify software installation

Do the following to verify that OMA and MATLAB are installed correctly:

1. Insert the product USB HASP key in a USB port on the controller PC.

2. Double-click the desktop OMA icon **Tektronix OUI (xx bit) Vertex Processing**. Wait for the OMA application to open. If the OMA application opens but seems to be locked up, see the troubleshooting section. (See page 10, *Troubleshooting OMA/MATLAB installation*.)

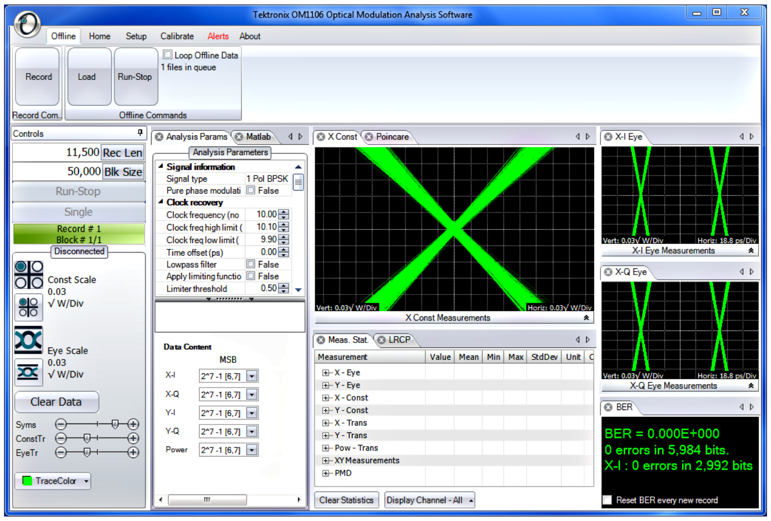
3. Click the **Home** menu tab, and then click the **1-pol I&Q** button (in the Layout section of the menu bar). The OMA screen populates to show the plot, readout, and control tabs for the 1-pol I&Q measurement.

4. Right-click in the **X-I Eye** plot to open the X-I Eye Options context menu:
   - If the Color Grade menu item is selectable, then the graphics card on your controller PC supports color grade and 3-d plots.
   - If the Color Grade menu item is grayed out, then the graphics card on your controller PC does not support the Vertex Processing version of OMA. Close OMA and then restart OMA using the **Tektronix OUI (xx bit)** desktop icon. Then repeat this procedure and continue to the next step.

5. Click **Offline** in the menu bar, click **Load**, and navigate to **My Documents\TekApplications\OUI\Mat Files\Simulated Data Files**.

6. Select file **QPSK1chA.mat**. and click **Open**.

7. Click **Run-Stop** (in the Offline menu bar). The OMA screen should look similar to the following image:



1 files in queue

11,500 Rec Len
50,000 Blk Size

Record # 1
Block # 1/1

Const Scale 0.03 √ W/Div
Eye Scale 0.03 √ W/Div

Signal information	
Signal type	1 Pol BPSK
Pure phase modulati	False
Clock recovery	
Clock frequency (no	10.00
Clock freq high limit (	10.10
Clock freq low limit (	9.90
Time offset (ps)	0.00
Lowpass filter	False
Apply limiting functio	False
Limiter threshold	0.50

Data Content

	MSB
X-I	2^7 -1 [6,7]
X-Q	2^7 -1 [6,7]
Y-I	2^7 -1 [6,7]
Y-Q	2^7 -1 [6,7]
Power	2^7 -1 [6,7]

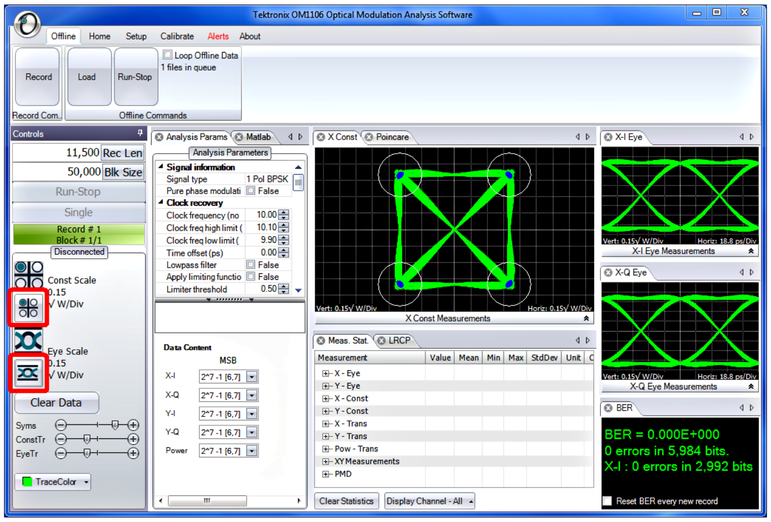
Vert: 0.03√ W/Div Horiz: 0.03√ W/Div

Vert: 0.03√ W/Div Horiz: 18.8 ps/Div

Measurement	Value	Mean	Min	Max	StdDev	Unit	C
X - Eye							
Y - Eye							
X - Const							
Y - Const							
X - Trans							
Y - Trans							
Pow - Trans							
XY Measurements							
PMD							

BER = 0.000E+000
0 errors in 5,984 bits.
X-I : 0 errors in 2,992 bits

8. Click the **Cont Scale** and **Eye Scale** buttons (marked in red on the following image) to reduce the plot scales to show the entire plot.



11,500	Rec Len
50,000	Blk Size

Const Scale 0.15 V W/Div

Eye Scale 0.15 V W/Div

Signal information

Signal type	1 Pol BPSK
Pure phase modulati	False

Clock recovery

Clock frequency (no	10.00
Clock freq high limit (	10.10
Clock freq low limit (	9.90
Time offset (ps)	0.00
Lowpass filter	False
Apply limiting functio	False
Limiter threshold	0.50

Data Content

	MSB
X-I	2^7 -1 [6,7]
X-Q	2^7 -1 [6,7]
Y-I	2^7 -1 [6,7]
Y-Q	2^7 -1 [6,7]
Power	2^7 -1 [6,7]

Vert: 0.15V W/Div Horiz: 0.15V W/Div

Vert: 0.15V W/Div Horiz: 18.8 ps/Div

Measurement	Value	Mean	Min	Max	StdDev	Unit
X - Eye						
Y - Eye						
X - Const						
Y - Const						
X - Trans						
Y - Trans						
Pow - Trans						
XY Measurements						
PMD						

BER = 0.000E+000
0 errors in 5,984 bits.
X-I : 0 errors in 2,992 bits

9. If you see the above screen, then OMA and MATLAB installed correctly.

   Go to the *Verify or set OM series instrument IP address* section to set OM instrument connections. (See page 12.)

   Go to the *Scope Connect* section to learn how to connect to an oscilloscope. (See page 31, *Scope Connect button*.)

Further information

Troubleshooting OMA/MATLAB installation

If the OMA window opens, but is not functional (appears to lock up), the most likely cause is a MATLAB version or activation problem.

- Open the MATLAB software interface and check that the correct version was installed. If the wrong version of MATLAB was installed, uninstall both MATLAB and OMA, and then reinstall the software following the installation order and instructions in this manual.

NOTE. *If more than one MATLAB version is installed on the controller PC, OMA may be 'connected' to the wrong version. (See page 11, Using other MATLAB installations.)*

- If the version is correct, confirm that the MATLAB software was successfully activated after the install (see the MATLAB install instructions for how to activate the program).
- If the above items do not fix the problem, please contact Tektronix Customer Support for assistance.

Using other MATLAB installations

Reverting to other MATLAB versions or post installation steps for a system with multiple MATLAB installation: To use a version of MATLAB that is not the most recent one installed on your computer, you need to register the older version as the COM server. This is done as follows:

- Once MATLAB installation completes, find the shortcut for the MATLAB version you want to use and double click it to start the MATLAB Desktop for that version.
- Run the following two lines in the MATLAB Command window to establish the running Matlab.exe as the correct Com-Server:
  ```
  cd(fullfile(matlabroot,'bin',computer('arch')))
  !matlab /regserver
  ```

Unexpected results

If you get unexpected results, note anything reported in the MATLAB Engine Response Window, or in the Alerts tab. Also open the MATLAB Command Window from the MATLAB desktop application, enter **CoreProcessingCommands**, and note any response. Provide this information when contacting Tektronix Customer Service.

Verify HASP protection

You can verify the HASP protection by opening the SafeNet Sentinel Admin Control Center:

1. Open a web browser on the controller PC and enter **http://localhost:1947** in the address bar to open the Sentinel Admin Control Center Web page.
2. Click the **Sentinel Keys** link in the left panel.
3. Verify that your key is listed as a local HASP HL Pro key.

Verify or set OM series instrument IP address

Before you can use the OMA software to take measurements, you must make sure that IP addresses of the connected OM series instruments are set correctly for your network, to enable communications between the instruments and the OMA software. The following sections describe how to connect OM instruments to DHCP and non-DHCP networks.

All OM instruments must be set to the same network subnet (DHCP-enabled networks do this automatically) to communicate with each other.

Verify OM instrument connectivity on DHCP-enabled network

OM series instruments are set by default to use DHCP to automatically assign IP addresses. If you are connecting the OM instruments and PC controller over a DHCP network, you do not need to specifically set the OM instrument IP address, as the DHCP server automatically assigns an IP address during each instrument's power-on process.

The following procedure describes how to verify that the OMA software can detect and connect to OM instruments that are set to use DHCP-generated IP addresses on a DHCP-enabled network:

Prerequisites:

- Required software is installed on the controller PC (OMA, MATLAB, and so on).
- OM instrument(s) and the controller PC are connected to the same DHCP-enabled local network.
- OM instruments are set to use DHCP (default configuration).

Verify OMA can detect OM instruments on DHCP network

To verify that the OMA software can detect and connect to OM instruments located on a DHCP-enabled local network:

1. Connect the OM instrument(s) to the DHCP-enabled local network.
2. Power on each OM instrument. The instrument queries the DHCP server to obtain an IP address. Wait until the front panel Enable/Standby button light turns off, indicating it has obtained an address. Push the front panel Enable/Standby button to enable the network connection (button light turns On).
3. Double-click the **Tektronix OUI** software desktop icon to start the OMA software.
4. Click **Setup > Optical Connect** to open the Device Setup dialog box.
5. Click **Auto Configure** to search the network and list all detected OM instruments. If all connected instruments are listed, then correct IP addresses were automatically assigned.

   If the Device Setup dialog does not list all connected instruments:

   - Verify that instruments are connected to the correct DHCP-enabled network.
   - Verify that instruments are powered on and in the correct power-on state for network access (front panel power button light is on).
   - Work with your IT resource to resolve the connection problem.
6. Click **OK** to close the Device Setup dialog box and return to the OMA main screen. The OMA software can now access and control the OM instruments.

Set OM instrument IP address for use on non-DHCP network

To connect the OM series instrument to a non-DHCP network, you must set the IP address and related settings on the OM instrument to match those of your non-DHCP network. All devices on non-DHCP network (OM instruments, PCs running OM software, 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. If your network administrator needs the MAC address of the OM instrument, the MAC address is located on the 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.

NOTE. *Use the system configuration tools on 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:

- Connect the OM instrument(s) to a DHCP-enabled network and use the LRCP tab in OMA to change the IP address (easiest way).
- Connect the OM instrument directly to a PC (with OMA installed) that is already set to the same IP address subnet as the OM instrument, and use the LRCP tab controls in OMA to change the IP address.

Change OM instrument IP address using DHCP network

To use a DHCP network to change the IP address of an OM instrument:

1. Connect the OM instrument(s) to the DHCP-enabled network.
2. Power on the OM instrument. The instrument queries the DHCP server to obtain an IP address. Wait until the front panel Enable/Standby button light turns off, indicating it has obtained an address. Push the front panel Enable/Standby button to enable the network connection (button light turns On).

3. Access the connection setup controls:
   - **OM1106**:
     a. Double-click the **OM1106** software desktop icon.
     b. Click **Setup** > **Optical Connect** to open the Device Setup dialog box.
   - **LRCP**:
     - Double-click the **LRCP** desktop icon. Enter password **1234** if requested.
     - Click **Device Setup** to open the Device Setup dialog box.
4. Click **Auto Configure** to search the network and list all detected OM instruments.

   If the Device Setup dialog does not list all connected instruments:
   - Verify that instruments are connected to the correct DHCP-enabled network
   - Verify that instruments are powered on
   - Work with your IT resource to resolve the connection problem
5. Double-click in the **IP Address** field of the instrument to change and enter the new IP address for that OM instrument.
6. Click the corresponding **Set IP** button. A warning dialog box appears indicating that the IP address will be changed and that you must record the new IP address. Losing the IP address will require connecting the instrument to a DHCP router.
7. Click **Yes** to set the new IP address.
8. Edit the Gateway and Net Mask (obtain this information from your network support).
9. Click **OK**.
10. Repeat steps 5 through 9 to change any other OM instrument IP addresses.
11. Exit the OM program.
12. Power off the OM instrument(s) and connect it to the non-DHCP network.
13. Run LRCP or OM1106 on the non-DHCP network and use the **Auto Config** button in the Device Setup dialog box to verify that the instrument is listed with the new IP address.

Change OM instrument IP address using direct PC connection

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

- Install OM1106 or LRCP 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 OM1106 or 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.*

Set PC IP address to match OM instrument.

1. On the PC with OM1106 or 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.

Run OMA on direct-connected PC to change OM instrument IP address.

1. Connect the OM instrument to the PC (directly, or through a hub or switch connected to the PC). Do not connect over a network.
2. Power on the OM instrument. Wait until the front panel Enable/Standby button light turns Off.
3. Push the **Enable/Standby** button again to enable the network connection (button light turns On).
4. Double-click the **OM1106** software desktop icon.
5. Click **Setup > Optical Connect** to open the Device Setup dialog box.
6. Click **Auto Configure** to search the network and list all detected OM instruments.

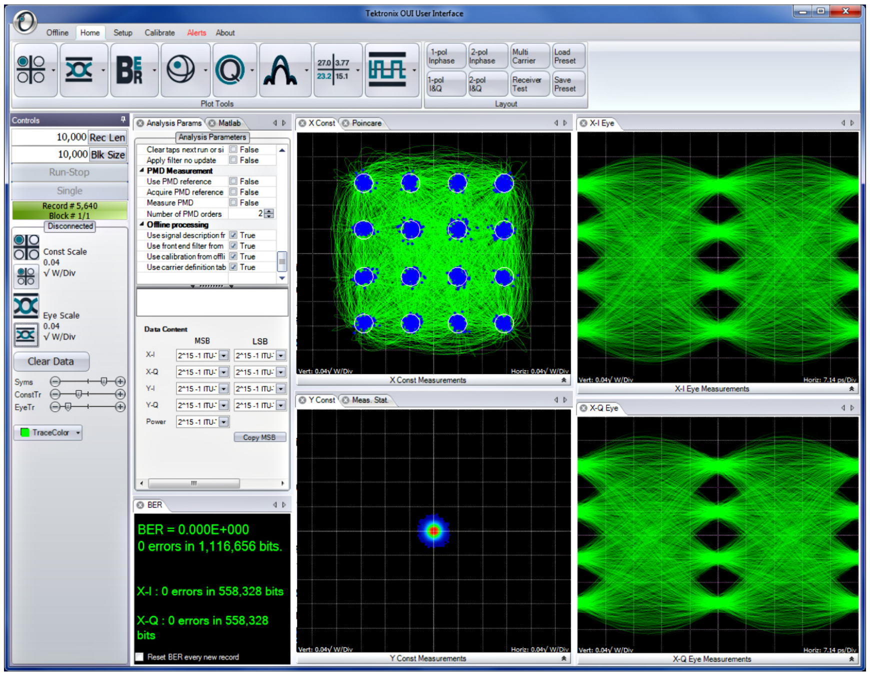
   If the Device Setup dialog does not list all connected instruments:

   - Verify that instruments are connected to the correct DHCP-enabled network
   - Verify that instruments are powered on
   - Work with your IT resource to resolve the connection problem
7. Double-click in the **IP Address** field of the instrument to change and enter the new IP address for that OM instrument.
8. Click the corresponding **Set IP** button. A warning dialog box appears indicating that the IP address will be changed and that you must record the new IP address. Losing the IP address will require connecting the instrument to a DHCP router.
9. Click **Yes** to set the new IP address.
10. Edit the Gateway and Net Mask (obtain this information from your network support).
11. Click **OK**.
12. Exit the OM program.
13. Power off the OM instrument.
14. Disconnect the OM instrument from the PC.
15. Connect the OM instrument to the target network.
16. Run the OM1106 or LRCP software on the PC connected to the same network as the OM instrument to verify that the OM software detects the OM instrument.

OM1106 Optical Modulation Analysis (OMA) user interface

The OM1106 Optical Modulation Analysis Software user interface (referred to as the OMA or OMA software throughout this document) is a powerful and flexible panel-based user interface for optical signal analysis. The main functions of OMA are:

- Detect OM instruments on the local network.
- Set parameters on the OM instruments (laser, modulator, and so on)
- Provide communication between all detected OM instruments, oscilloscopes, and the OMA software
- Acquire and process signals from OM instruments and the oscilloscope
- Display plots and measurements



10,000 Rec Len
10,000 Blk Size
Run-Stop
Single
Record # 5,640
Block # 1/1
Disconnected
Const Scale 0.04 √ W/Div
Eye Scale 0.04 √ W/Div
Clear Data
Syms
ConstTr
EyeTr
TraceColor

Analysis Parameters

Clear taps next run or si	False
Apply filter no update	False
PMD Measurement	
Use PMD reference	False
Acquire PMD reference	False
Measure PMD	False
Number of PMD orders	2
Offline processing	
Use signal description fr	✓ True
Use front end filter from	✓ True
Use calibration from offli	✓ True
Use carrier definition tab	✓ True

Data Content

	MSB	LSB
X-I	2^15 -1 ITU-	2^15 -1 ITU-
X-Q	2^15 -1 ITU-	2^15 -1 ITU-
Y-I	2^15 -1 ITU-	2^15 -1 ITU-
Y-Q	2^15 -1 ITU-	2^15 -1 ITU-
Power	2^15 -1 ITU-	

Copy MSB

BER = 0.000E+000
0 errors in 1,116,656 bits.

X-I : 0 errors in 558,328 bits

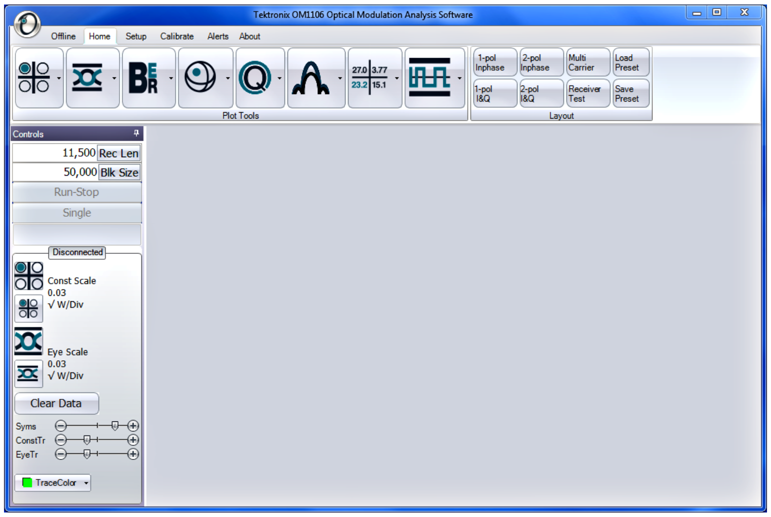
X-Q : 0 errors in 558,328 bits

Reset BER every new record

Figure 1: The OMA screen with plots and measurements

To start the OMA software, double-click the desktop icon to open the default screen.

***NOTE.** Insert the application HASP key into a USB port on the PC before starting the OMA.*

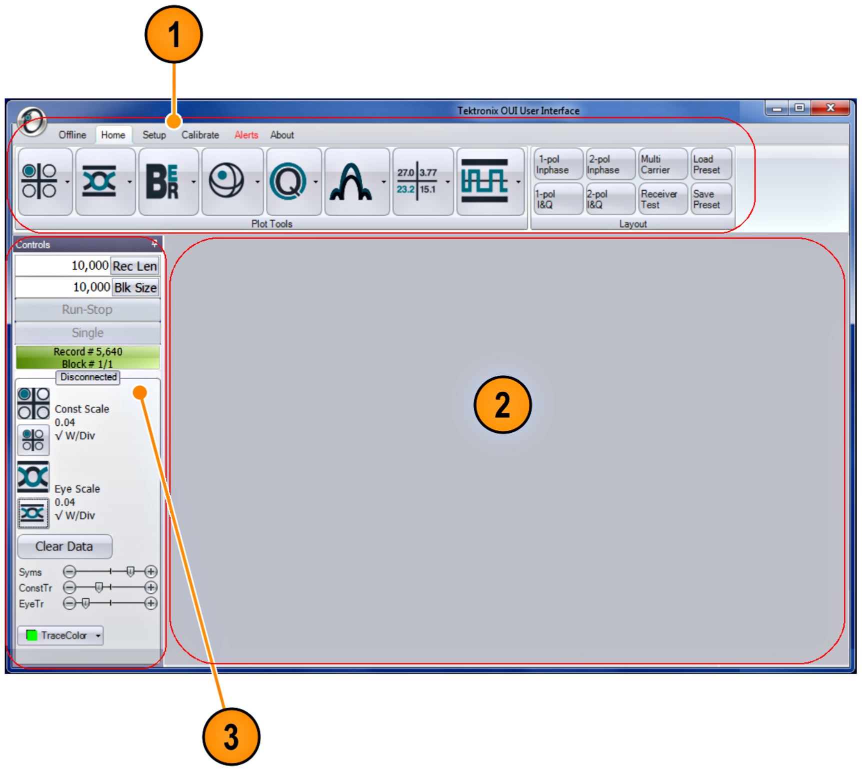


The OMA requires a third party program, MATLAB by MathWorks, which must be installed on the same PC as the OMA sofware. The OMA automatically launches and interfaces with the MATLAB application using engine mode. No direct user interaction with MATLAB is needed for most operations while using the OMA.

***NOTE.** MATLAB must be installed and activated before running the OMA software.*

***NOTE.** Typing **desktop** in the separate Matlab application window opens the full Matlab UI.*

OMA user interface elements



10,000 Rec Len

10,000 Blk Size

Record # 5,640
Block # 1/1

Const Scale 0.04 √ W/Div

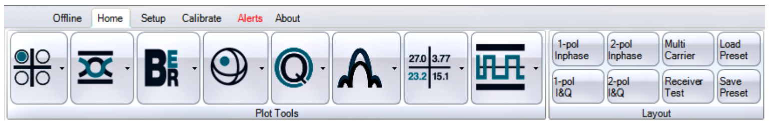
Eye Scale 0.04 √ W/Div

Figure 2: Default OMA startup screen

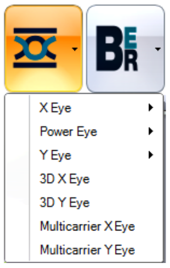
Item	Description
1	The Menu ribbon provides fast access to key tasks.
2	The Plots panel is a flexible panel-based area that displays the plots, measurement, and parameter control panels. (See page 23.)
3	The global Controls panel sets record length and block size, runs and stops live data acquisitions, sets plot display scales, and shows other controls as needed for particular plots. (See page 29.)

The Menu ribbon

The Menu ribbon provides fast access to key tasks. Click a main menu tab to show the associated task buttons.



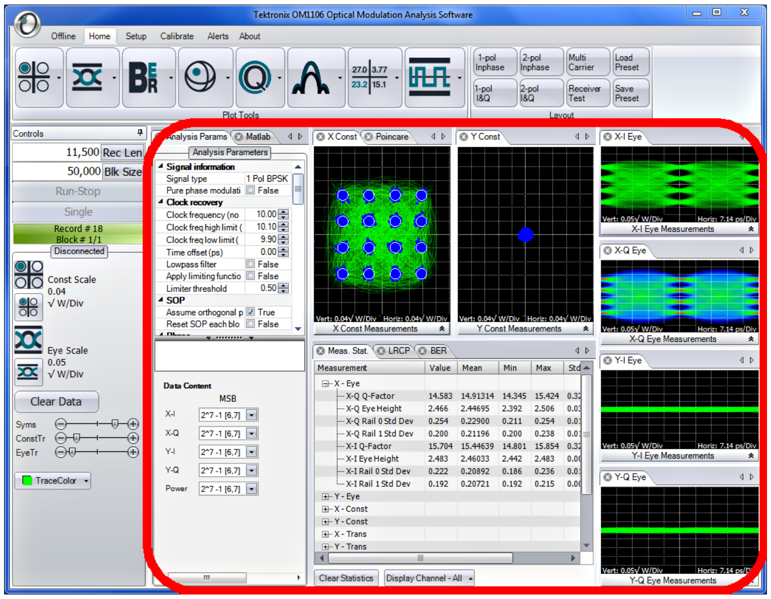
Click an icon to show the menu of available functions and sub-functions, open a dialog box, or run that task.



The menu ribbon buttons can be hidden to provide more room for plots and control tabs. To hide the menu bar, double click any main menu tab other than About. Unhide by double clicking again on a tab.

The Plots panel

The Plots panel is a flexible panel-based area that displays the plots, measurement, and parameter control panels. The Plots panel is empty by default when you start OMA. Select a predefined plot layout from the **Home > Layout** buttons to quickly populate the plot panel with parameter, control, and plots for the selected measurement. The following image shows the Plots panel populated by selecting the 2-pol I&Q layout button.



Analysis Parameters

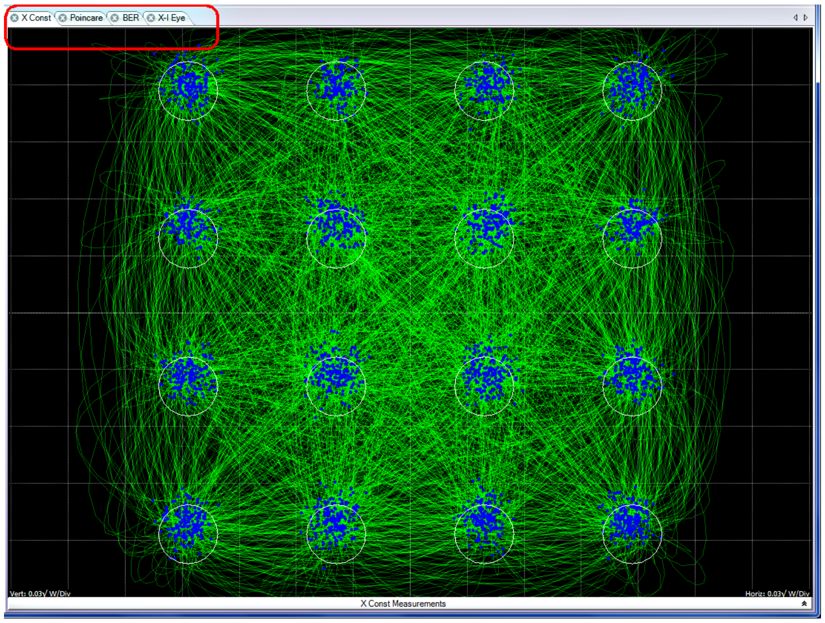
Signal information	
Signal type	1 Pol BPSK
Pure phase modulati	False
Clock recovery	
Clock frequency (no	10.00
Clock freq high limit (	10.10
Clock freq low limit (	9.90
Time offset (ps)	0.00
Lowpass filter	False
Apply limiting functio	False
Limiter threshold	0.50
SOP	
Assume orthogonal p	☑ True
Reset SOP each blo	False

Data Content

	MSB
X-I	2^7 -1 [6,7]
X-Q	2^7 -1 [6,7]
Y-I	2^7 -1 [6,7]
Y-Q	2^7 -1 [6,7]
Power	2^7 -1 [6,7]

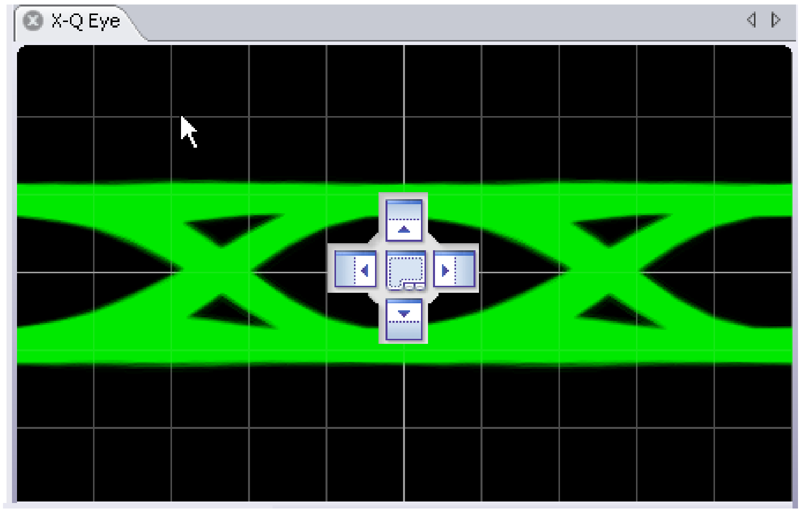
Measurement	Value	Mean	Min	Max	Std
X - Eye					
X-Q Q-Factor	14.583	14.91314	14.345	15.424	0.32
X-Q Eye Height	2.466	2.44695	2.392	2.506	0.03
X-Q Rail 0 Std Dev	0.254	0.22900	0.211	0.254	0.01
X-Q Rail 1 Std Dev	0.200	0.21196	0.200	0.238	0.01
X-I Q-Factor	15.704	15.44639	14.801	15.854	0.32
X-I Eye Height	2.483	2.46033	2.442	2.483	0.00
X-I Rail 0 Std Dev	0.222	0.20892	0.186	0.236	0.01
X-I Rail 1 Std Dev	0.192	0.20721	0.192	0.215	0.00
Y - Eye					
X - Const					
Y - Const					
X - Trans					
Y - Trans					

Select individual plots and measurements from the Home menu items to add to the Plots panel. New individually added plots open as full-screen plot on blank screens, or are added to an existing area when inserted in an existing layout, with each plot, readout, or control on its own tab. You can move plot, control, and readout tabs (referred to as plots throughout this manual) within the Plots panel area or drag them to the PC desktop.

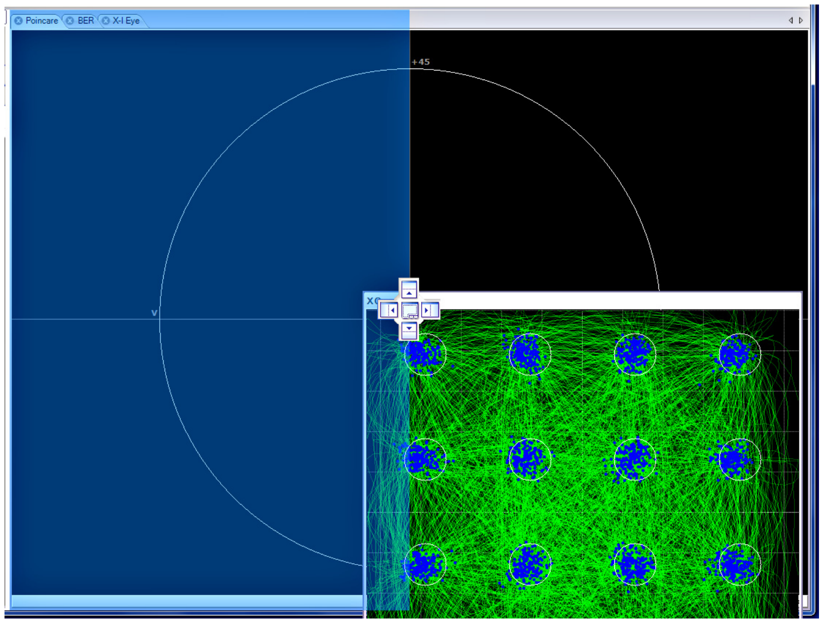


To change plot tab order. To change the plot tab order (within the same group of plot tabs), click and hold on the tab and drag it horizontally in the same tab group.

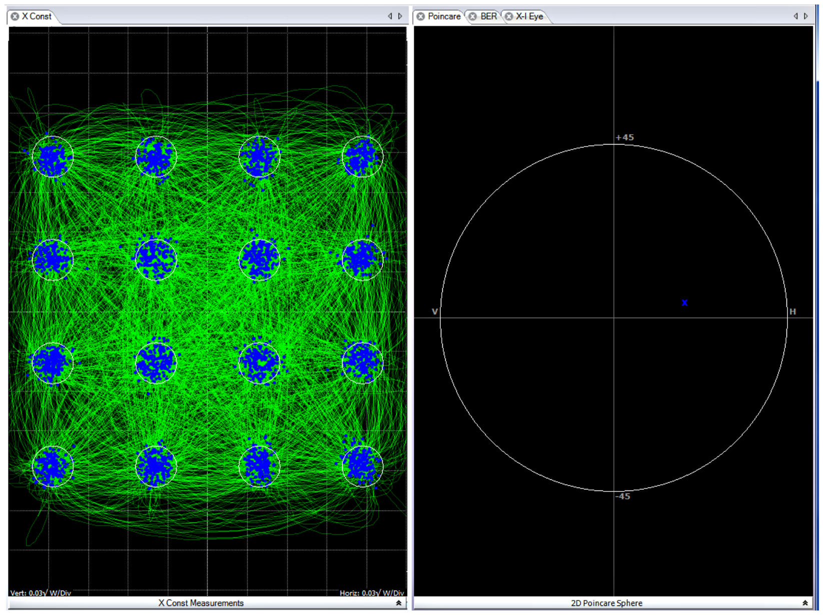
To move/arrange plots in the Plots panel. To move and arrange plots within the Plots panel, click and drag a plot tab into any open plot. OMA shows a plot position icon. Continue holding the mouse button and drag the cursor on to the positioning guide.



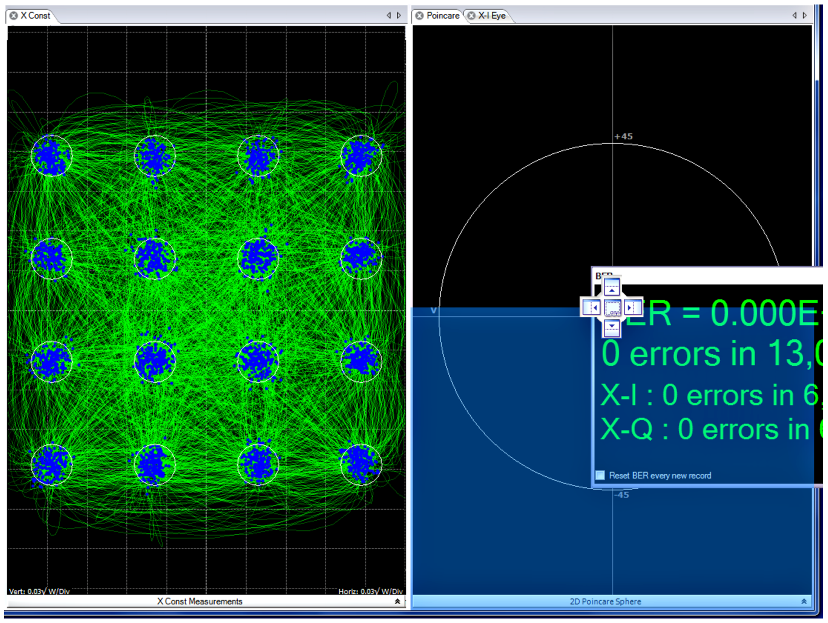
As you drag the cursor on the different areas of the plot position icon, the screen is shaded light blue to indicate where the plot will be positioned (above, below, left, or right of the current plot).



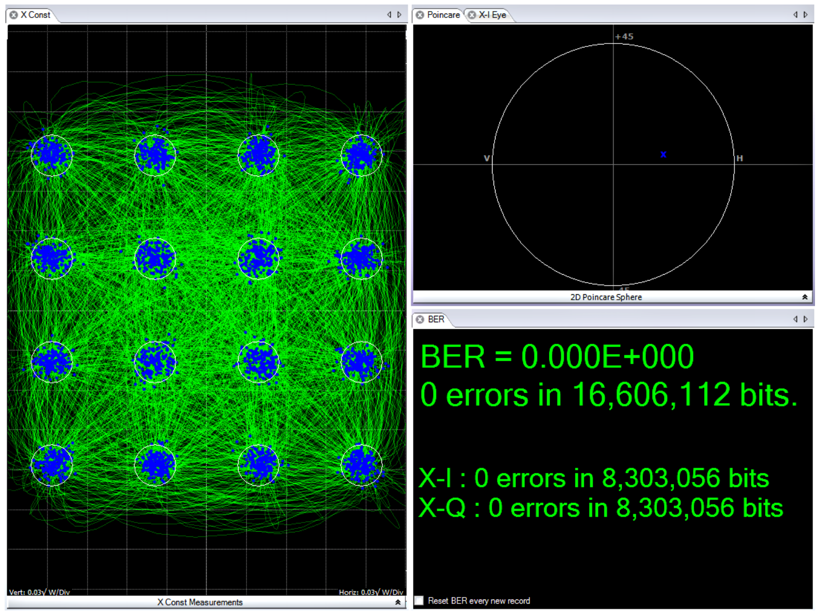
Release the mouse button and the OMA places the plot into a new panel in the selected area.



The following images show moving the BER plot to display below the Poincaré plot, and the X-I Eye plot to display below the constellation plot.

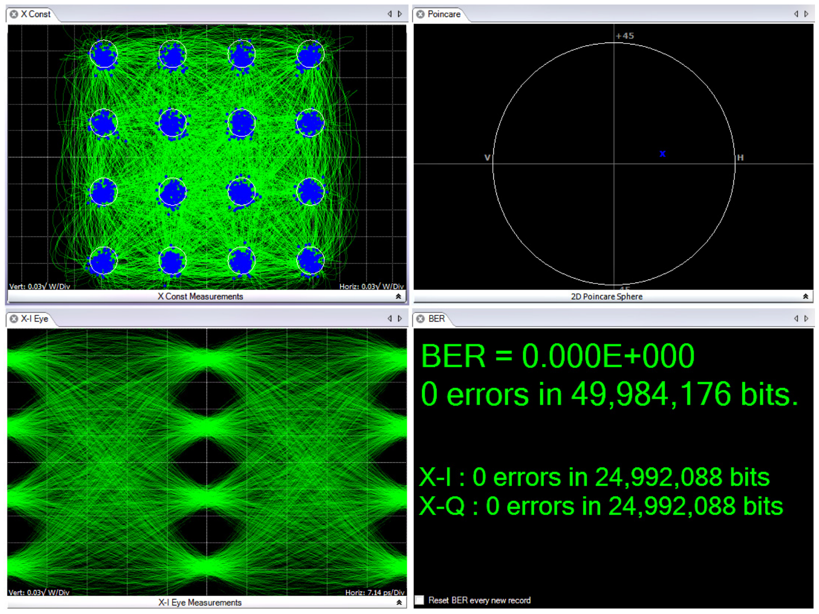


ER = 0.000E
0 errors in 13,
X-I : 0 errors in 6
X-Q : 0 errors in



BER = 0.000E+000
0 errors in 16,606,112 bits.

X-I : 0 errors in 8,303,056 bits
X-Q : 0 errors in 8,303,056 bits



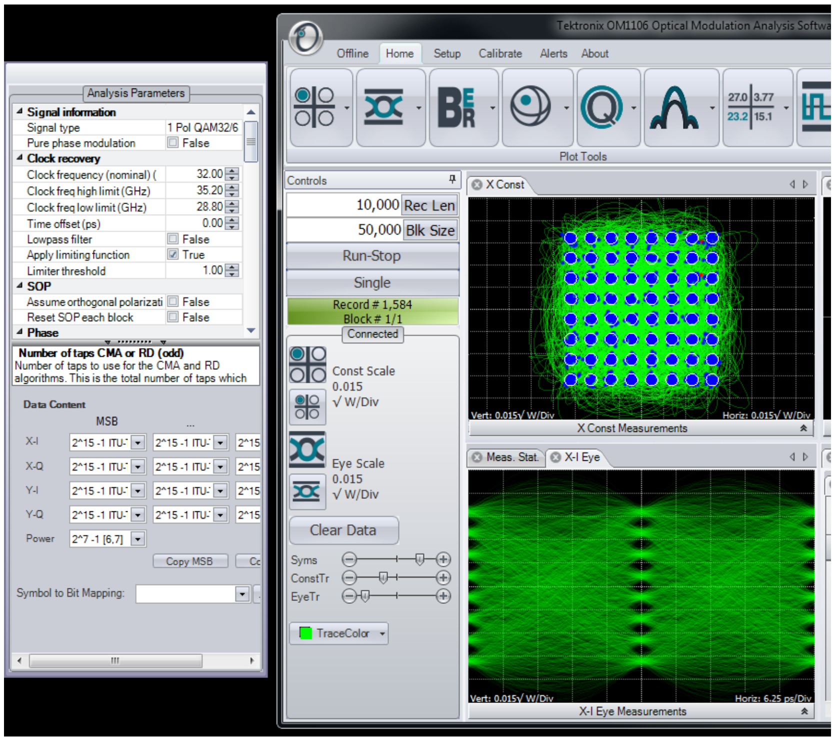
BER = 0.000E+000
0 errors in 49,984,176 bits.

X-I : 0 errors in 24,992,088 bits
X-Q : 0 errors in 24,992,088 bits

Dragging a plot to the center of the positioning icon places the moved plot as a tab in the plot pane to which it was dragged.

To move a plot to the desktop. To move a plot to a separate window on the desktop, click and hold on the plot tab and drag it to the desktop. To return the plot to the OMA application, double click in the title bar of the desktop plot window.

NOTE. *The plot may not move back to the same position in the Plots panel from where your dragged it.*



Analysis Parameters

Signal information	
Signal type	1 Pol QAM32/6
Pure phase modulation	☐ False
Clock recovery	
Clock frequency (nominal) (	32.00
Clock freq high limit (GHz)	35.20
Clock freq low limit (GHz)	28.80
Time offset (ps)	0.00
Lowpass filter	☐ False
Apply limiting function	✓ True
Limiter threshold	1.00
SOP	
Assume orthogonal polarizati	☐ False
Reset SOP each block	☐ False
Phase	

Number of taps CMA or RD (odd)
Number of taps to use for the CMA and RD algorithms. This is the total number of taps which

Data Content

	MSB	...	
X-I	2^15 -1 ITU-	2^15 -1 ITU-	2^15
X-Q	2^15 -1 ITU-	2^15 -1 ITU-	2^15
Y-I	2^15 -1 ITU-	2^15 -1 ITU-	2^15
Y-Q	2^15 -1 ITU-	2^15 -1 ITU-	2^15
Power	2^7 -1 [6,7]		

Copy MSB

Symbol to Bit Mapping:

Controls

10,000	Rec Len
50,000	Blk Size

Run-Stop

Single

Record # 1,584
Block # 1/1

Connected

Const Scale 0.015 √ W/Div

Eye Scale 0.015 √ W/Div

Clear Data

Syms
ConstTr
EyeTr

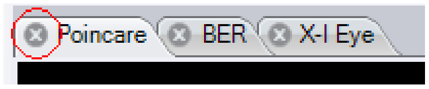
🟩 TraceColor

To save plot layouts. You can save and recall (load) custom plot layouts by using the Load Preset and Save Preset functions in the Home ribbon menu. (See page 110, *Layout controls*.)

To show additional plot controls, information. Plots may have dropdown panels to show extra information or controls associated with that plot. Click the double arrow on the bottom border of a tab ⮝ to display or hide the extra information for that tab.

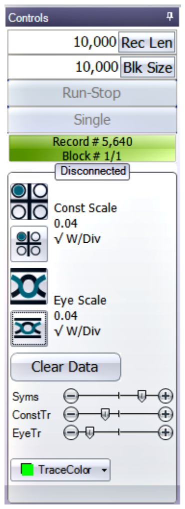
To resize plot panels. To resize a plot panel, position the cursor in the gray divider area between plots; the cursor changes shape to a bar with arrows. Click and hold the right mouse button and drag the panel divider to change the panel size.

To delete a plot. To delete a plot, click the X icon (circled in red in the following image).



To show additional plot display settings. Plots may have additional display settings (markers, save to, color grade, trace color, and so on). Right-click in a plot to see available display or other options.

The global Controls panel



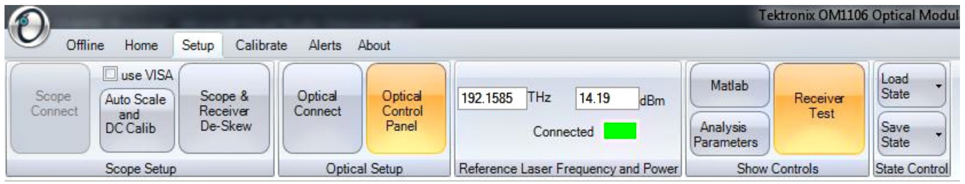
The global Controls panel sets record length and block size, runs and stops live data acquisitions, sets plot display scales, and shows other controls as needed for particular plots. This control panel is pinned to the left side of the application by default for easy access. You can click the pushpin icon in the upper right to minimize the controls to allow for maximum Plot panel area.

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 application). The scale units are √W/div.

There are also controls to adjust the intensity of symbol points and the intensity and color of constellation and eye traces. You can also adjust trace color and intensity levels may using the right-click menu on each individual plot. (See page 119, *The global Controls panel*.)

The Setup tab controls

Use the Setup tab controls to detect, connect, and configure connected oscilloscopes (real-time and equivalent time) and OM series instruments to perform measurements with the OMA software. The following sections describe the controls in detail, grouped by the control categories (Scope Setup, Optical Setup, Reference Laser, Show Controls, and State Control).

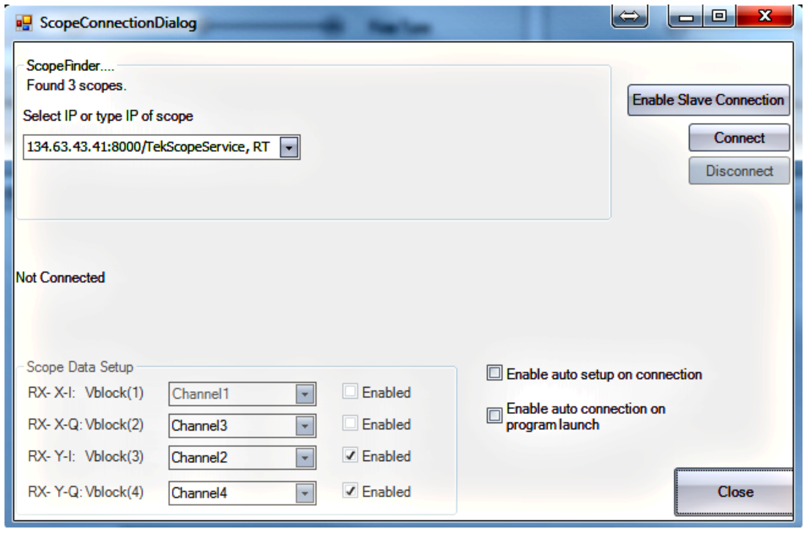


Scope Setup controls

Scope Connect button

The Scope Connect functions search for and connect to network-connected oscilloscopes, and assign oscilloscope channels to the OM instrument inputs. Click **Scope Connect** on the Setup tab to open the ScopeConnectionDialog box.

NOTE. *The Scope Connect functions require that the Scope Service Utility (SSU) be installed and running on connected oscilloscopes.*



ScopeConnectionDialog

ScopeFinder....
Found 3 scopes.

Select IP or type IP of scope

134.63.43.41:8000/TekScopeService, RT

Not Connected

Scope Data Setup

RX- X-I: Vblock(1)	Channel1	Enabled
RX- X-Q: Vblock(2)	Channel3	Enabled
RX- Y-I: Vblock(3)	Channel2	✓ Enabled
RX- Y-Q: Vblock(4)	Channel4	✓ Enabled

Enable auto setup on connection

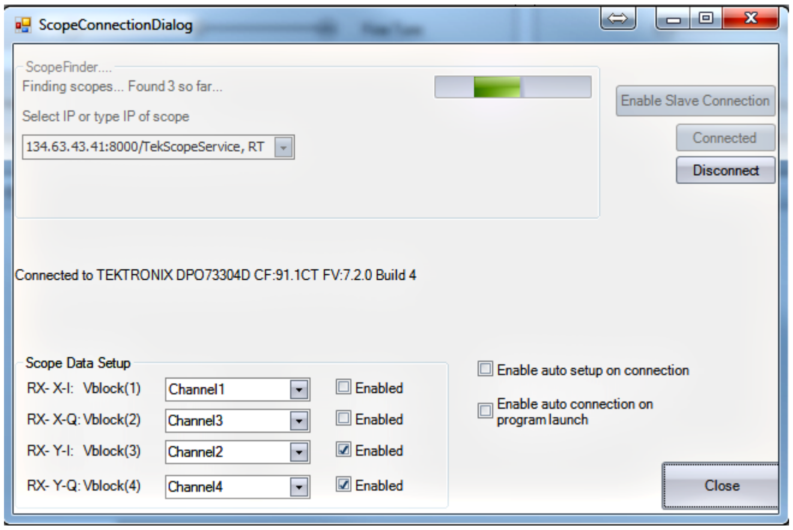
Enable auto connection on program launch

A green progress bar at the top indicates that the software is searching for oscilloscopes on the same subnet that are running the Scope Service Utility (SSU). As they are found they are added to the drop-down menu. Click the IP address field and select the IP of the oscilloscope to which to connect, then click Connect.

If the OMA Scope Connection Dialog box reports 0 Scopes Found, you will have to manually enter the IP address. This happens when connecting over a VPN or when network policies prevent the IP broadcast. When typing the address in manually, do not include “, RT” on the end; just enter the IP address. Click **Connect**.

After connection, use the Scope Data Setup fields to map the oscilloscope channels to the OM instrument receiver channels and corresponding MATLAB variable fields. The MATLAB variable storing the oscilloscope data is named Vblock. Data from the selected channel is moved into the indicated Vblock variable.

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



ScopeFinder....
Finding scopes... Found 3 so far...
Select IP or type IP of scope
134.63.43.41:8000/TekScopeService, RT

Connected to TEKTRONIX DPO73304D CF:91.1CT FV:7.2.0 Build 4

Scope Data Setup

RX- X-I: Vblock(1)	Channel1	☐ Enabled
RX- X-Q: Vblock(2)	Channel3	☐ Enabled
RX- Y-I: Vblock(3)	Channel2	☑ Enabled
RX- Y-Q: Vblock(4)	Channel4	☑ Enabled

☐ Enable auto setup on connection
☐ Enable auto connection on program launch

There are two features which may be enabled on this dialog box. "Enable auto setup on connection" means that any time you connect to an oscilloscope, the oscilloscope will be requested to restore the state saved the last time the OUI was closed.

"Enable auto connection on program launch" means the oscilloscope connection will be made automatically when the OUI software is launched. This assumes that the oscilloscope is present at the last IP address where it was found and that the Scope Service Utility is running on the oscilloscope. It is a good idea to use fixed or reserved IP addresses when using any autoconnect features.

Once connected and configured, close the connect dialog box.

Click **Enable Slave Connection** to enable selecting 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 oscilloscope is the one triggering on the sync board output only. (See page 151, *Configuring two Tektronix 70000 series oscilloscopes*.)

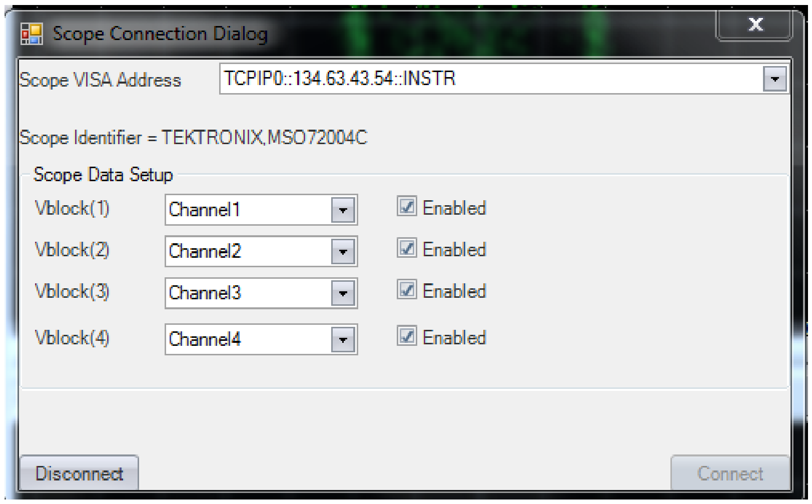
NOTE. *The DPO/DPS70000SX series oscilloscopes behave as a single oscilloscope when connected in the UltraSync configuration, and do not require the second IP address.*

Use VISA control (Scope Setup)

Click (select) the **Use VISA** box in the Scope Setup area of the Setup menu to use TekVISA to communicate with a Tektronix MSO/DSO70000 or 70000B series oscilloscope.

NOTE. *Click Use VISA before clicking Scope Connect.*

Click the Connect button to open the VISA-specific Scope Connection Dialog box.


Scope Connection Dialog

Scope VISA Address: TCPIP0::134.63.43.54::INSTR

Scope Identifier = TEKTRONIX,MSO72004C

Scope Data Setup

Vblock(1)	Channel1	✓ Enabled
Vblock(2)	Channel2	✓ Enabled
Vblock(3)	Channel3	✓ Enabled
Vblock(4)	Channel4	✓ Enabled

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 oscilloscope IP address, for example 172.17.200.138 in the example below.

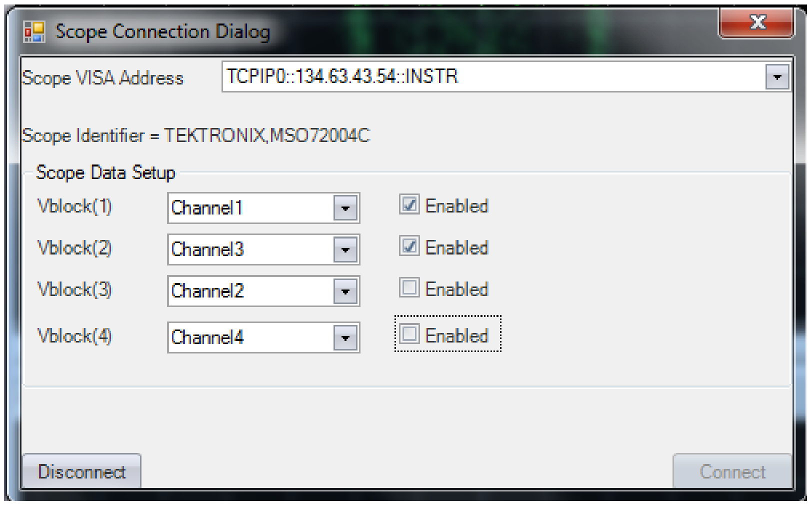
NOTE. *To quickly determine the oscilloscope IP address, open a command window ("DOS box") on the oscilloscope and enter* ***ipconfig /all*** *to display the instrument IP address.*

After clicking Connect, the drop down boxes are 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

The following example disables two channels and sets the other two channels 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 click Single on the oscilloscope before connecting from the OMA.*



Scope VISA Address: TCPIP0::134.63.43.54::INSTR

Scope Identifier = TEKTRONIX,MSO72004C

Scope Data Setup

Vblock(1)	Channel1	☑ Enabled
Vblock(2)	Channel3	☑ Enabled
Vblock(3)	Channel2	☐ Enabled
Vblock(4)	Channel4	☐ Enabled

Table 1: Oscilloscope connectivity capabilities (TekVISA vs. Scope Service Utility)

OMA Capability	TekVISA	Scope Service Utility (non-TekVISA)
Segmented readout for unlimited record size	Yes	Yes
Ability to collect data from two networked oscilloscopes running the Scope Service	No	Yes
Scope config, Auto connect, Auto scale, and Deskew	No	Yes
Software required on oscilloscope	LAN server	Scope Service Utility
DPO/DPS 70000 SX series compatibility	No	Yes
Real-time oscilloscope compatibility	Any real-time Tektronix oscilloscope supported by the IVI driver	MSO/DSO70000C, D, DX, SX series oscilloscopes with firmware v6.4 or later

Non-VISA oscilloscope connections (Scope Service Utility)

As mentioned above, the other choice for connecting to the oscilloscope and collecting data is through the Scope Service Utility (SSU). The SSU is a program that runs on each oscilloscope connected to the OMA controller PC.

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

Once the SSU is installed on the oscilloscope, start the “Socket Server” and the TekScope oscilloscope application, then double-click the SSU desktop icon to start the SSU application. Minimize the application before connecting to the oscilloscope from OMA.

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

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

NOTE. *Clicking Connect on the OMA Setup Tab opens the Scope Connection dialog box for connecting to the SSU. (See page 31, Scope Connect button.)*

Once the oscilloscope connection is configured, close the Connect dialog box. Then use the Configure Scope, Auto Scale, and Scope & Receiver Deskew buttons (in that order) to finish setting up oscilloscope.

Auto Scale and DC Calib button

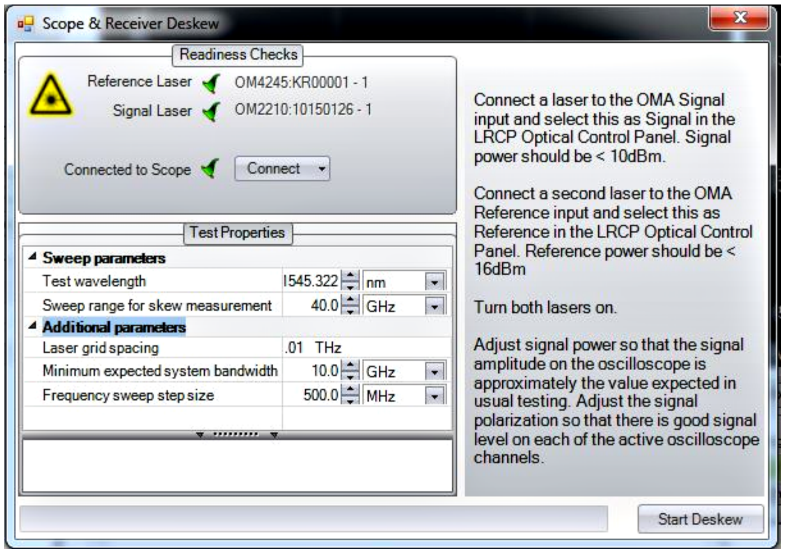
Click the **Auto Scale and DC Calib** button once you have connected to an oscilloscope and have prepared your signal and Reference laser as required for your testing. Click Auto Scale any time the signal level from the oscilloscope may have changed.

DC Calibration (measuring and removing static dc offsets) will be performed anytime the an Auto Scale is requested. To run DC Calibration without an Auto Scale, use the button on the Calibration tab.

Auto Scale senses the size of the signal on each oscilloscope input to determine the proper scale and offset settings, and then sets the oscilloscope with the settings.

Scope & Receiver Deskew button

Click the **Scope & Receiver Deskew** button to open the tool dialog box.



Reference Laser OM4245:KR00001 - 1
Signal Laser OM2210:10150126 - 1
Connected to Scope

Sweep parameters		
Test wavelength	1545.322	nm
Sweep range for skew measurement	40.0	GHz
Additional parameters		
Laser grid spacing	.01	THz
Minimum expected system bandwidth	10.0	GHz
Frequency sweep step size	500.0	MHz

Connect a laser to the OMA Signal input and select this as Signal in the LRCP Optical Control Panel. Signal power should be < 10dBm.

Connect a second laser to the OMA Reference input and select this as Reference in the LRCP Optical Control Panel. Reference power should be < 16dBm

Turn both lasers on.

Adjust signal power so that the signal amplitude on the oscilloscope is approximately the value expected in usual testing. Adjust the signal polarization so that there is good signal level on each of the active oscilloscope channels.

The Scope & Receiver Deskew dialog box facilitates the deskew process for a new OMA setup or one that has had any changes to the X, Y, I, or Q path lengths or cabling. The dialog box includes instructions on how to set up for deskewing.

Select the Skew measurement range. The range should be between 25% and 50% of the combined OMA/oscilloscope system bandwidth. The default value of 10 GHz works in most cases.

The laser grid is changed for this measurement because it may be necessary to tune continuously across grid points. If you are already using a 10 or 12.5 GHz grid, choose that value for the grid to be used for the deskew. If you are using a different grid, the grid is set back to its original value after the deskew is complete. A step size of 500 MHz is recommended but can be increased to save time or decreased to collect more data.

Select the desired test wavelength for the deskew. The lasers are set back to their original wavelength after the deskew. Skew is a not a significant function of wavelength, so the test wavelength can be chosen to be any convenient value inside of the tuning range of the Signal and Reference lasers.

Set the minimum expected system bandwidth to be equal to the lesser of the OMA and oscilloscope bandwidths. The deskew utility will report an error if no signal is found at any frequency below this value. The default value works in most cases.

Once setup is complete and the readiness checks are passed, click **Start Deskew** to perform the deskew process.

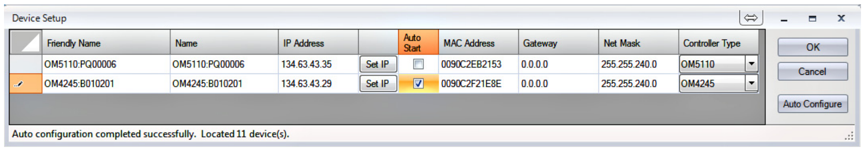
The signal and reference lasers are fine tuned over the specified skew measurement range to measure the average phase slope and calculate the relative path delays (skews).

When the deskew process is complete, the values in the Calibration tab "Sliders" area are updated to those measured.

It is a good idea to save the OMA state after completing a deskew to save the values. (see Setup: Save State)

Optical Connect

The Optical Connect button (Setup > Optical Connect) opens the Device Setup dialog box, which detects and connects to all OM instruments that it detects on the local network.



Friendly Name	Name	IP Address		Auto Start	MAC Address	Gateway	Net Mask	Controller Type
OM5110:PQ00006	OM5110:PQ00006	134.63.43.35	Set IP	☐	0090C2EB2153	0.0.0.0	255.255.240.0	OM5110
OM4245:B010201	OM4245:B010201	134.63.43.29	Set IP	✓	0090C2F21E8E	0.0.0.0	255.255.240.0	OM4245

Auto configuration completed successfully. Located 11 device(s).

Run the **Setup > Optical Connect** task when you run the OMA software for the first time, and any time that you add or remove instruments from the network.

To have OMA detect network-connected instruments:

1. Start OMA.
2. Click **SETUP > Optical Connect** to open the Device Setup dialog box.
3. Click **Auto Configure** to search the network and list all detected OM instruments. This search can take a few minutes.

   If the Device Setup dialog does not list all connected instruments:

   - Verify that OM instruments are connected to the correct network
   - Verify that instruments are powered on and their network connection is enabled (the On/ Standby button on the front panel is on)

   NOTE. *If it is necessary to reset the instrument network connection, press and hold the front-panel On/Standby button until the light changes color to reboot the instrument.*

   - If the above items do not help, work with your IT resource to resolve the connection problem
4. Use the Friendly Name field to attach custom labels to OM instruments that help you identify the type and/or location of the instruments. Friendly Names are retained in the LRCP software and are tied to the corresponding instrument MAC address.
5. (Optional) You can use the **Set IP** button to manually set the instrument IP address. This is only necessary in a network environment that is not using DHCP to automatically assign IP Addresses. (See page 12, *Verify or set OM series instrument IP address*.) The Set IP button only changes the IP address and does not save other modified fields like Friendly Name.

6. (Optional) Select Auto Start to enable auto connection and configuration of this hardware when the OUI/LRCP is launched. The Auto Start hardware is configured at OUI/LRCP launch to match the state when the OUI/LRCP was last closed. The hardware must be present at the last known IP address for the automatic connection to work.
7. Click **OK** to exit the dialog and save any changes (such as Friendly Name). OMA lists the detected OM devices as tabs on the main screen, using the friendly name and IP address to allow for easy identification.

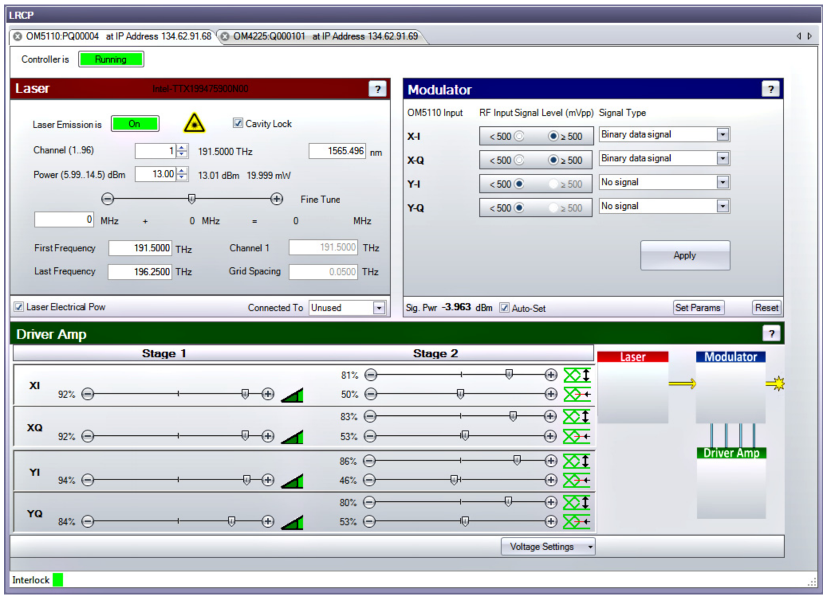
NOTE. *If you do not click OK, the listed instruments are not connected to LRCP or saved in the software.*

NOTE. *OMA does not automatically update the connected devices list on startup. Disconnected or powered-off instruments will still be shown in the list and be shown as offline. You should run the Auto Configure task in on a regular basis after starting OMA or if OMA has been on for a long period of time, to update the connected device list.*

Optical Control Panel (LRCP)

Click the **Optical Setup** button to open a LRCP (Laser/Receiver Control Panel) tab in the Plots panel. The LRCP lets you control connected instrument lasers and modulator parameters.

LRCP creates a tab for each detected instrument, labeled with the device name and IP address. Clicking an instrument tab displays the available controls for that device. The contents of a control pane depend on the OM instrument associated with that tab.



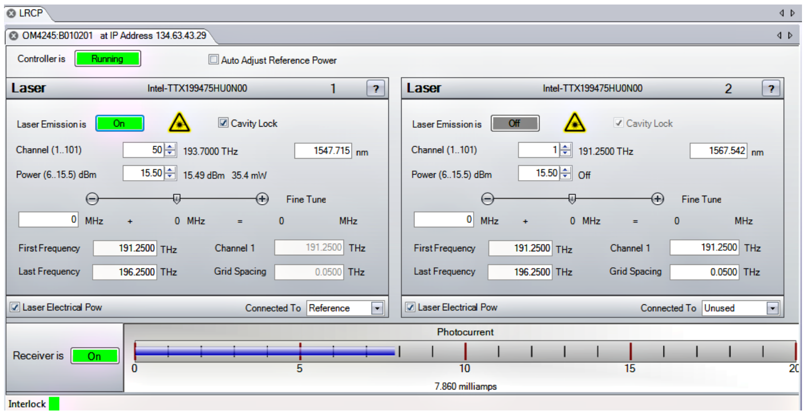
Controller is Running

Laser: Laser Emission is On; Cavity Lock ✓
Channel (1..96) 1, 191.5000 THz, 1565.496 nm
Power (5.99..14.5) dBm 13.00, 13.01 dBm 19.999 mW
Fine Tune: 0 MHz + 0 MHz = 0 MHz
First Frequency 191.5000 THz; Channel 1 191.5000 THz
Last Frequency 196.2500 THz; Grid Spacing 0.0500 THz
Laser Electrical Pow ✓; Connected To Unused

Modulator: OM5110 Input; RF Input Signal Level (mVpp); Signal Type
X-I: ≥ 500; Binary data signal
X-Q: ≥ 500; Binary data signal
Y-I: < 500; No signal
Y-Q: < 500; No signal
Sig. Pwr -3.963 dBm; Auto-Set ✓

Driver Amp:
XI: Stage 1 92%; Stage 2 81%, 50%
XQ: Stage 1 92%; Stage 2 83%, 53%
YI: Stage 1 94%; Stage 2 86%, 46%
YQ: Stage 1 84%; Stage 2 80%, 53%

Figure 3: LRCP tab showing OM5110 controls.



Controller is Running; Auto Adjust Reference Power ☐

Laser Intel-TTX199475HU0N00 1: Laser Emission is On; Cavity Lock ✓
Channel (1..101) 50, 193.7000 THz, 1547.715 nm
Power (6..15.5) dBm 15.50, 15.49 dBm 35.4 mW
Fine Tune: 0 MHz + 0 MHz = 0 MHz
First Frequency 191.2500 THz; Channel 1 191.2500 THz
Last Frequency 196.2500 THz; Grid Spacing 0.0500 THz
Laser Electrical Pow ✓; Connected To Reference

Laser Intel-TTX199475HU0N00 2: Laser Emission is Off; Cavity Lock ✓
Channel (1..101) 1, 191.2500 THz, 1567.542 nm
Power (6..15.5) dBm 15.50, Off
Fine Tune: 0 MHz + 0 MHz = 0 MHz
First Frequency 191.2500 THz; Channel 1 191.2500 THz
Last Frequency 196.2500 THz; Grid Spacing 0.0500 THz
Laser Electrical Pow ✓; Connected To Unused

Receiver is On; Photocurrent 7.860 milliamps

Figure 4: LRCP tab showing OM4200 controls.

Each **Instrument tab** in the LRCP represents one physical Laser Control device (for example, an OM4245 or an OM2210) on the local network. Each instrument tab has one or more of the following control types:

- Laser controls show available laser control functions for connected instruments with laser output capability. (See page 42, *The Laser controls*.)
- Modulator controls set the optical modulator bias of an OM instrument. (See page 45, *The Modulator controls*.)
- Driver Amp controls set the behavior of the optical modulator RF Input electrical amplifier. (See page 49, *The Driver Amp controls*.)
- The **Receiver gauge** displays the total photocurrent output from an instrument. This readout is only functional on devices like the OM4000-series instruments that have the appropriate hardware installed.
- The **Status bar** provides important information about the overall state of the communications with the instrument controllers. Each controller has a unique status bar.

Connect to an OM instrument

You need to connect to an instrument before you can make changes to settings. To connect to an instrument from the LRCP tab:

1. Click an instrument tab.
2. Click the **Offline** button. The button changes colors to show the connection status:
   - **a.** The button turns yellow and reads "Connecting…" to show that a physical network connection is being established over a socket.
   - **b.** The button turns teal and reads "Connected…" to show that a session is established between the device and Control Panel. Commands are sent to initialize the communications with the laser and identify their capabilities.
   - **c.** The button turns bright green when the controller and lasers are ready to operate from the software.

   NOTE. *The button color scheme (bright green = running or active; gray = off line or inactive; red = warning or error state) is consistent throughout the application.*

3. Once the instrument is connected, the tab populates with controls and fields relevant to the connected OM device (instrument name, laser manufacturer and model, available settings, an so on). You can now change settings and turn the laser(s) on or off.

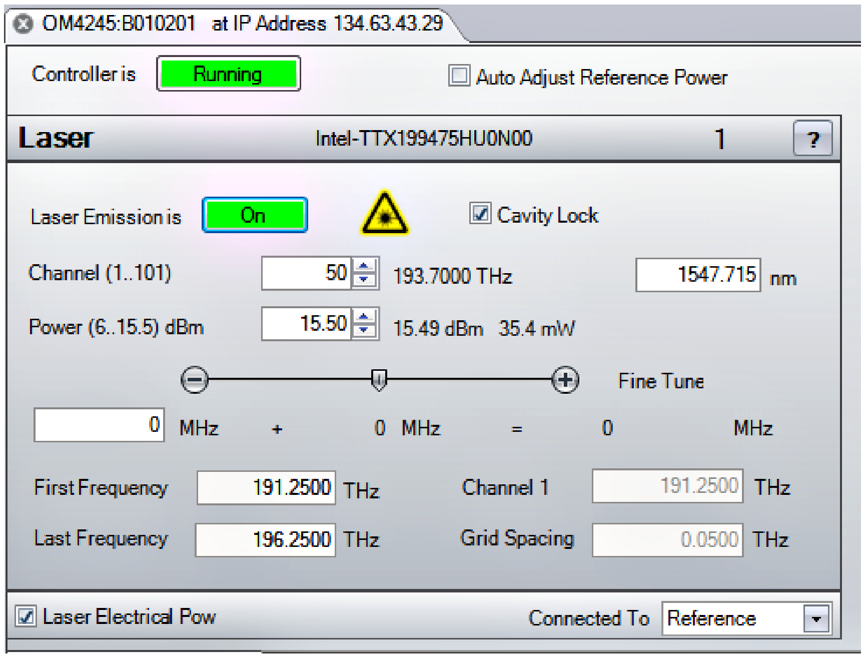
Each instrument preserves its settings (including the emission state) when you exit the application. If the OM device is powered down, it will return to its default power-on state when it is switched back on.

The very first time the LRCP connects to an OM5110, there is a delay while the LRCP calculates the initial modulator parameters so that they may be stored away in the LRCP Program Files directory. The modulator parameters, including null voltages and Vpi voltages for the various modulator sections, are needed to obtain proper optical bias for the modulator. The LRCP saves the current state of each OM5110 on first connection so that you can restore the parameters if needed.

More information on manually setting the modulator parameters is listed in the Modulator Controls section. (See page 46, *Manual modulator settings view (Auto-Set check box cleared)*.)

The Laser controls

The Laser control area displays available laser control functions for connected instruments with laser output capability.



OM4245:B010201 at IP Address 134.63.43.29

Controller is Running

☐ Auto Adjust Reference Power

Laser Intel-TTX199475HU0N00 1

Laser Emission is On

☑ Cavity Lock

Channel (1..101) 50 193.7000 THz 1547.715 nm

Power (6..15.5) dBm 15.50 15.49 dBm 35.4 mW

Fine Tune

0 MHz + 0 MHz = 0 MHz

First Frequency 191.2500 THz Channel 1 191.2500 THz

Last Frequency 196.2500 THz Grid Spacing 0.0500 THz

☑ Laser Electrical Pow Connected To Reference

Table 2: Laser controls (LRCP)

Control	Description
Auto Adjust Reference Power	Enables the automatic control of the power setting of the laser identified as the Reference laser. The automatic control loop will set the laser power near maximum unless the Signal input power is so large that the total photocurrent is above the recommended range. If the total photocurrent is too high, the Reference laser power setting is reduced to bring the photocurrent into the recommended range.
Laser Emission is	Enables or disables laser emission output from the front panel connectors. The emission status is indicated both by the green color of the button and by the green LED on the instrument front panel.
Cavity Lock	Enables or disables the ITLA laser cavity lock. Certain laser models have a cavity lock feature that increases their frequency accuracy at the expense of dithering the frequency. Cavity Lock is necessary to tune the laser, but can be unchecked to suppress the dither. Ordinarily, Cavity Lock should be enabled (selected) so that the laser is able to tune, change power level, and lock on to its frequency reference. However, once tuning is complete and the laser has stabilized, you can disable Cavity Lock 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. This feature is helpful where the lowest phase noise is required.
Channel	Sets the laser channel. Type a number or use the up/down arrows to choose a channel. The range of channels available depends 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.
Power	Sets the laser power level. Type or use the up/down arrows to select the laser power level. The allowed power range is shown next to the control.
Fine Tune	Enables tuning the laser off grid up to 12 GHz. Change this value by typing a number in the text box or by dragging the slider. The sum of the text box and slider values is sent to the laser. Once the laser accepts the new value, that value is displayed after the '=' sign.
First Frequency	Shows the lowest frequency to which you can tune the laser. Readout only.
Last Frequency	Shows the highest frequency to which you can tune the laser. Readout only.
Channel 1	Settable when emission is off. This is the definition of Channel 1.
Grid Spacing	Sets the laser 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.

Table 2: Laser controls (LRCP) (cont.)

Control	Description
Laser Electrical Power	Turns on or off electrical power to the laser module. This should normally be selected (checked). Unchecking this box turns off electrical power to the laser module. Only turn off electrical power to reset the laser to its power-on state, or to preserve laser lifetime if a particular laser is never used.
Connected To	Sets where this laser is connected. The control software must know if this laser is being used as the Reference for a coherent receiver. Select Reference if this laser is connected to the Reference (LO) input of a coherent receiver.

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-series instrument lasers: just type in the same channel number for each laser.

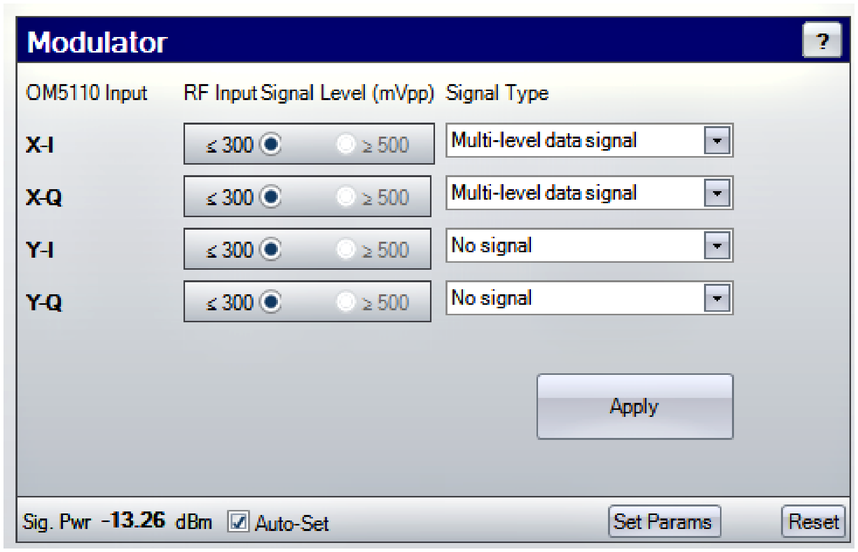
If using an external transmitter laser, you can type in its wavelength and the controller selects the nearest channel. If this is not close enough, try choosing a finer WDM grid or use the fine tuning feature. Use the Fine tuning slider bar (when available) to fine tune the laser. Fine tuning typically works over a range of ±10 GHz from the center frequency of the channel selected.

The Modulator controls

The Modulator controls set the optical modulator bias of an OM5110 or other supported instrument.

Auto modulator settings view (Auto-Set check box selected)

The Auto-Set check box, at the bottom of the control area, enables or disables the modulator automatic optical bias function settings screen. When the check box is selected, settings are controlled automatically based on the specified signal level and type. When Auto-Set is cleared, you can manually enter modulator settings.



OM5110 Input	RF Input Signal Level (mVpp)	Signal Type
X-I	[✓ ≤ 300] ≥ 500	Multi-level data signal
X-Q	[✓ ≤ 300] ≥ 500	Multi-level data signal
Y-I	[✓ ≤ 300] ≥ 500	No signal
Y-Q	[✓ ≤ 300] ≥ 500	No signal

Sig. Pwr -13.26 dBm [✓] Auto-Set

Table 3: OM5110 Modulator controls (Auto-Set mode) (LRCP)

Control	Description
RF Input Signal Level (mVpp)	Set whether the input signal to each OM5110 input (X-I, X-Q, Y-I, and Y-Q) is less than or greater than the listed value. ***NOTE.*** *Signal level should be less than 300 mV$_{pp}$ or greater than 500 mV$_{pp}$. Values between 300 mV$_{pp}$ and 500 mV$_{pp}$ require reducing the electrical amplifier gain or use of external attenuators to obtain a signal level between 100 mV$_{pp}$ and 300 mV$_{pp}$.*
Signal Type	Sets the input signal type. Valid types are No Signal, Binary data signal, and Multi-level data signal.
Apply	Send the settings to the OM5110. When the wait circle disappears, your settings have been applied. The OM5110 retains these settings until they are changed. No settings are sent or retained by the OM5110 until you click the Apply button.

Table 3: OM5110 Modulator controls (Auto-Set mode) (LRCP) (cont.)

Control	Description
Sig. Pwr	(Readout only) The Modulated Output Signal power (abbreviated Sig. Pwr.) readout at the bottom of the Modulator control area. If the output is too high or too low, it may temporarily affect the controller circuits of the OM5110. In this case the power readout changes color and mouse-over text is available to indicate that optical bias and power readout may not be precise. There is no harm operating like this if the input optical power is within the specified range.
Set Params	Opens the Set Modulator Parameters dialog to set the Optimum Bias Voltage and Vpi Voltage parameters. ***NOTE.*** *It is particularly important to have a good estimate for the XP and YP quadrature phase settings. See the calibration section for details.*
Reset	Sets the optical bias control voltages to the default values. This is helpful whenever a major change is made to the system such as turning on the laser or input signals. Clicking Reset generally helps the system reach steady-state operation the fastest.

Manual modulator settings view (Auto-Set check box cleared)

The Manual Settings View provides the greatest degree of control flexibility, but is more complex than Automatic Settings View. Since each setting may take five seconds to be stored in an instrument, and possibly several minutes to reach steady state, it is best to use the Automatic Settings View where all the settings are established at once. The Manual Settings View is helpful when it is necessary to make fine adjustments to optimize a signal, or when it is desirable to impair the signal.

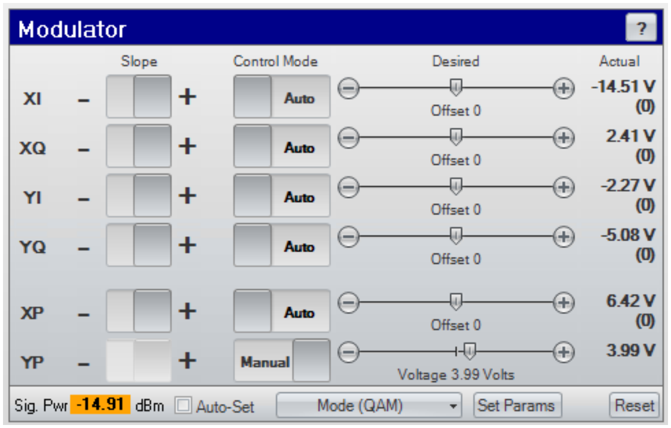


Table 4: OM5110 Modulator controls (manual mode) (LRCP)

Control	Description
Slope	Usually **-** for > 500 mV_{pp} inputs and **+** for < 300 mV_{pp} inputs.
Control Mode	Auto to use automatic optical bias control based on feedback from the output optical signal. Manual to set the optical modulator bias voltage to a particular value.
Voltage/Offset	The slider control is used to set the desired voltage when in Manual mode or to set the Offset when in Auto mode. Offset is the amount to offset the bias from where it would normally be in Auto mode. The units are arbitrary and vary based on Optical Input power. The Offset must be tuned while observing the Modulated Optical Output signal on an appropriate optical signal analyzer to obtain the desired signal behavior.
Actual	This column shows the voltages at the optical modulator bias inputs. The value in parentheses is the actual Offset value.
Signal Mode	The optical bias controller behaves differently depending on the type of electrical signal input. Binary signals require 2-pol QPSK mode. QAM signals generally require QAM mode. Again it is best to use the Automatic Settings View which chooses the most appropriate Signal Mode automatically.
Set Modulator Parameters	The 6 modulator sections of the OM5110 modulator (X-I, X-Q, Y-I, YQ, XP, and YP) each have particular null voltages, where that section outputs minimum optical power, and Vpi voltages, which is the voltage difference between null and peak transmission. This type of information is needed by the OM5110 optical bias controller to properly control the modulator sections. The OM5110 is preprogrammed at the factory with the optimum bias and Vpi voltages. Optimum bias voltages are stored rather than null voltages to make them easier to set. The optimum bias voltages do change with time, and are different for different RF drive levels. It is not important for these values to be very precise. You should update your modulator parameters only if the OM5110 fails to obtain proper optical bias within a few minutes. Providing a better set of optimum bias voltages speeds the time to proper optical bias. The Vpi voltages do not change appreciably with time or temperature and may be left at their factory-set values.

To determine the optimal bias voltage values:

1. Connect the OM5110 to an analyzer, such as the OM4245, that will report the signal quality of the OM5110. Connect the necessary signal inputs and turn on the laser source.
2. Use the **Modulator Auto-Set** view to set up the OM5110 for the required signal types and drive levels. Click **Apply**. Wait for this step to complete.
3. Deselect the **Auto-Set** box to see the Manual Settings view. Wait for the analyzer to report that the optical bias is correct.
4. If the optical bias does not meet your requirements, use the Manual Control Mode or the Offset function to correct the optical bias. This is easiest if the OM5110 is connected for single polarization IQ operation. That is, there should be proper drive signals connected to either XI and XQ or to YI and YQ. The X parameters are determined with XI and XQ driven, the Y parameters are determined with YI and YQ driven. It is important to drive both I and Q or the phase (XP or YP) will not be known.

   After connecting the XI and XQ signals, use Auto Control Mode for XI, XQ, YI, YQ, and manual control for XP and YP. Try several values for XP leaving YP alone, waiting each time for XI, XQ, YI, and YQ to auto bias. Once proper X constellation bias is achieved, record these values and then move the drive signals to YI and YQ and repeat the process.

   If the autobias does not work for several different XP voltage settings, verify that the signal levels are < 300 mVpp or > 500 mVpp and that the Auto Set panel was correspondingly configured and Applied.
5. Record the voltages shown on the Manual Settings view once the optical bias value meets your requirements.
6. Click **Set Params**. Enter the voltages shown in the Manual Settings view (step 5) as the Null Voltages in the Set Parameters dialog box.

NOTE. *If using Set Params results in worse values, click **Restore Initial Values** to reload the settings originally detected by the LRCP at first connection to the OM5110.*

---

7. Click **OK**.
8. To verify the Null Voltage values, change every segment to **Manual Control Mode** and click **Reset**. The voltages shown should match those found in step 5) to within 0.01 V.
9. Return to the Auto-Set view and click **Apply** to return to automatic control.

The Driver Amp controls

The Driver Amp controls set the behavior of the optical modulator RF Input electrical amplifier. This two-stage amplifier can work in both linear and nonlinear modes to enable both linear electrical-to-optical conversion and binary optical signal generation which is insensitive to the electrical input signal level.

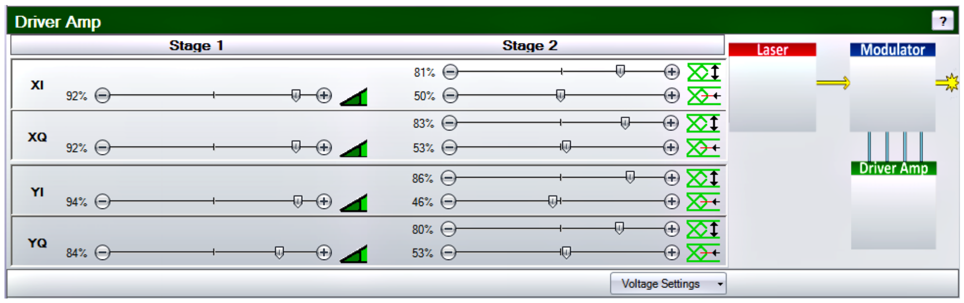


Table 5: OM5110 Driver Amp controls (LRCP)

Control	Description
Stage 1	First stage of electrical amplification. You can adjust the gain of each Stage 1 amplifier. This should not be needed for most applications, but is helpful to balance the amplitude of I-Q signals when operating in the linear range (< 300 mV$_{pp}$ electrical input).
Stage 2	Second stage of electrical amplification. When operating with >500 mVpp electrical input, you can adjust the crossing point and amplitude of the signal driving the optical modulator. These controls are not effective in the linear range (< 300 mV$_{pp}$) and can be left at their default values.
Voltage Settings	**Save current voltages as power-on defaults**, which stores all of the current Driver Amp settings in the OM5110 as the new defaults. **Restore to factory defaults**, which loads the factory default values for the Driver Amp, overriding the current values. When the OM5110 is turned on and off by the rear-panel Primary power switch, or when it loses mains power, only the “power-on default settings,” and “factory defaults” are retained.

Each of the adjustments for linear gain, nonlinear crossing point, and nonlinear amplitude are indicated by a value in percent. This value is provided to help documentation of the amplifier settings. The control is not strictly proportional to this value, so these settings must be determined experimentally using the appropriate optical signal analyzer.

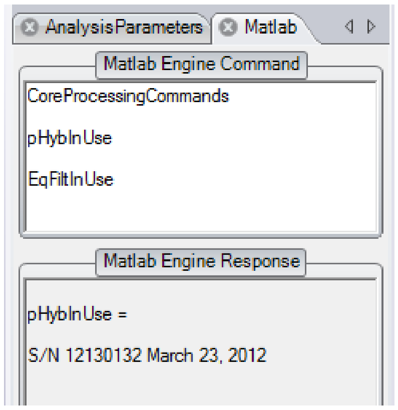
Reference Laser Frequency and Power

Use these fields to enter the frequency and power level of the reference laser connected to the OM instrument. A green readout indicates that the software recognizes the connected reference laser signal as valid.

MATLAB Command/Response tab

Click the Matlab button to open the **Matlab** command and response tab. The upper window is an interface to the MATLAB command processor. The lower window displays the response to commands entered into the upper window.

You can configure MATLAB to perform a wide range of mathematical operations on the raw or processed data using the Matlab window. Normally the only call is to CoreProcessingCommands, the set of routines that perform signal processing and analysis for real-time oscilloscopes.



```
CoreProcessingCommands

pHybInUse

EqFiltInUse
```

```
pHybInUse =

S/N 12130132 March 23, 2012
```

***NOTE.** The command 'CoreProcessingCommands' provides signal processing and analysis for real-time oscilloscopes.*

***NOTE.** To view a complete list of variables, open the MATLAB application window on the controller PC desktop and enter **who**.*

As with other OMA settings, the last MATLAB Engine Command file used is recalled when you run OMA. You can locate or create another appropriate engine file and paste it into the OMA MATLAB Command window. You can also use the Save/Load State command in the Setup tab to save the Software Settings which include the Matlab Engine Window contents.

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
- TXPulseType – When the RDLMS adaptive filter is enabled, skew measurements are disabled by default since the adaptive filter compensates for XY skew. The skew measurements may be re-enabled, by providing information about your signal so that the software can estimate the skew based on the RDLMS filter tap weights.

  Use the following settings depending on your TX signal type:

  TXPulseType: this variable sets if the pulse type after receive-side filtering is raised-cosine, using a setting of 0, or root-raised-cosine, using a setting of 1. For square pulse types, use raised cosine with a TXPulseRollOffFactor of 1.

  TXPulseRollOffFactor: this variable sets the rolloff factor for either the root-raised-cosine or raised-cosine pulse types. For square pulses use a value of 1 to best approximate the pulse shape.

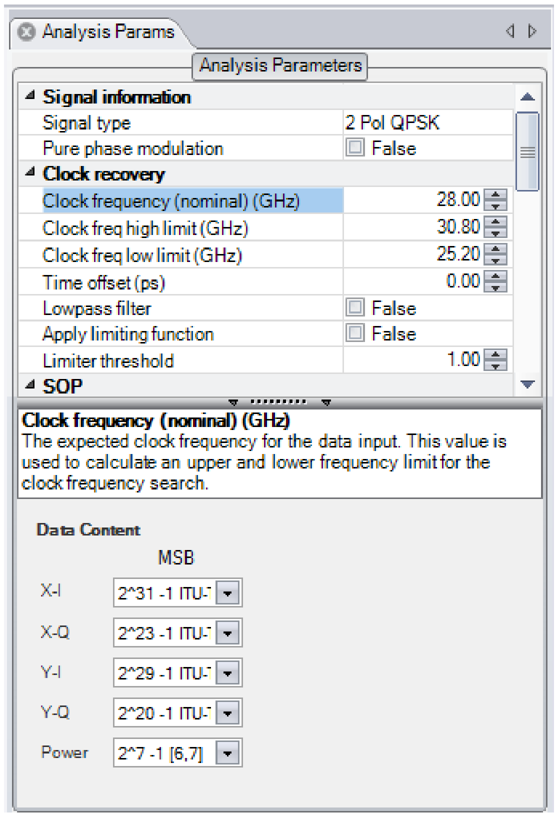
  Example for raised cosine pulse type with roll off factor of 0.2:

  TXPulseType = 0; TXPulseRollOffFactor = 0.2;
- EVMType – If not specified or set to 1, the Error Vector Magnitude (EVM) is calculated using the largest ideal constellation point magnitude as a reference for presenting the EVM as a percentage. This has been the most popular definition for optical signals. EVMType = 3 uses the average rms magnitude of the ideal symbols weighted by their frequency of occurrence in the data set as the reference. This type is more popular for rf signals and is growing in usage for optical signals.
- 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.

See the MATLAB-related appendices for more information on MATLAB and OMA operation using the ATE interface. (See page 173, *The ATE (automated test equipment) interface*.) (See page 207, *MATLAB CoreProcessing function reference*.) (See page 225, *MATLAB variables used by CoreProcessing*.) (See page 161, *MATLAB CoreProcessing software guide*.)

Analysis Parameters control and configuration tab

Click the Analysis Parameters button to open the **Analysis Params** control and configuration tab. This tab lets you set measurement analysis parameters, including signal information, clock recovery, SOP, phase, eye, and so on. This is the main parameter configuration control to use while running OMA.



Signal information

Signal type	2 Pol QPSK
Pure phase modulation	False

Clock recovery

Clock frequency (nominal) (GHz)	28.00
Clock freq high limit (GHz)	30.80
Clock freq low limit (GHz)	25.20
Time offset (ps)	0.00
Lowpass filter	False
Apply limiting function	False
Limiter threshold	1.00

SOP

Clock frequency (nominal) (GHz)
The expected clock frequency for the data input. This value is used to calculate an upper and lower frequency limit for the clock frequency search.

Data Content

	MSB
X-I	2^31 -1 ITU-T
X-Q	2^23 -1 ITU-T
Y-I	2^29 -1 ITU-T
Y-Q	2^20 -1 ITU-T
Power	2^7 -1 [6,7]

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

The controls listed in Table 7 are relevant to both equivalent-time and real-time oscilloscopes except where noted.

Table 6: Analysis Parameters fields

Parameter	Description
Signal information	
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.

Table 6: Analysis Parameters fields (cont.)

Parameter	Description
Clock recovery	
Clock frequency (nominal) (GHz)	The expected clock frequency for the data input. This value is used to calculate an upper and lower frequency limiter for the clock frequency search.
Clock freq high limit (GHz)	The highest expected clock frequency. A smaller low to high limit range improves the clock recovery function. Try to exclude the frequency that is equal to the sampling rate divided by two.
Clock freq low limit (GHz)	The lowest expected clock frequency. A smaller low to high limit range improves clock recovery function. Try to exclude the frequency that is equal to the sampling rate divided by two.
Time offset	Offset in time applied after clock recovery. Applies an offset in time (horizontal movement on the eye diagram) to the signal. If a signal has structure, for example ringing, then the clock recovery process may give a result displaced from the symbol center. The Time offset adjustment can move it back.
Lowpass filter	Inserts a lowpass filter to just the clock recovery path; it does not affect the signals seen in the OMA plots. 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. Enabling the lowpass filter can center the eye diagram.
Apply limiting function	Inserts a limiting function to just the clock recovery path; it does not affect the signals seen in the OMA plots. Some signal distortions such as ringing can 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). Enabling the limiting function can center the eye diagram.
Limiter threshold	Sets the clock recovery limiter threshold level, relative to mean. Start with a threshold value of 1. Increase or decrease the value to achieve best eye-timing stability.
SOP	
Assume Orthogonal Polarizations	Checking this box forces Core Processing to assume that the two polarization multiplexing 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 are 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 6: Analysis Parameters fields (cont.)

Parameter	Description
Phase	
2nd Phase Estimate	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.
Homodyne (RT oscilloscopes only)	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 it 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.
Phase estimation time constant parameter (Alpha)	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. The best digital filter is of the form: $1/(1+\alpha z^{-1})$ where α is related to the time constant, τ, of the filter by the relation $\tau = -T/\ln(\alpha)$ where T is the time between symbols. So, an $\alpha = 0.8$ when the baud rate is 10 Gbaud gives a time constant, $\tau = 450$ ps, or a low-pass filter bandwidth of 350 MHz. The value of α 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 ($\sim\tau/T$) are needed for the filter delay. This of course is not a problem, but an $\alpha=0.999$ 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 $\geq 10/(1-\alpha)$.

Table 6: Analysis Parameters fields (cont.)

Parameter	Description
Phase estimation time constant parameter (Alpha) (cont.)	The selection of the best value of Alpha is discussed later in the *EstimatePhase* section. (See page 216, *EstimatePhase*.) This best 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. An Alpha that is too small for a given laser means there is 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. 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 such that peak-to-peak frequency wander is on the order of the laser linewidth. For the lasers supplied with the OM4000 instruments, a block size of 50,000 points is a good choice.
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 rotates from one symbol to the next.
Eye	
Balanced Differential Detection (BDD)	Sets the differential-detection emulator to emulate balanced instead of single-ended detection.
Constellation	
Continuous traces	Enables drawing fine trace lines that connect the constellation points. If unchecked, the traces are 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.

Table 6: Analysis Parameters fields (cont.)

Parameter	Description
BER	
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.
Display	
Continuous trace points per symbol	Sets the number of samples per symbol for the clock retiming used to create the fine traces in the phase and eye diagrams.
Averaging	
Calculate transition average	Enables 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 rise time.
Calculate subsequence average	Enables computation of the subsequence averaging. Refresh rate is faster when disabled, but must be enabled to display the subsequence average in the spectrum plots.
Subsequence average length	Sets the number of symbols in each subsequence.
System impulse response	
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. This control is also used by the adaptive filter choices (such as Nyquist), which uses the impulse response to calculate the needed filter.
Tributaries contributing to impulse response	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.
Calculate expected eye	Controls computation of the expected eye based on the system impulse response. Refresh rate is faster when disabled, but must be enabled to display the expected eye in an eye diagram.
Calculate expected waveform vs. time	Controls computation of the expected waveform vs. time based on the system impulse response. Refresh rate is faster when disabled, but must be enabled to display the expected waveform in the X vs. T diagram.
Calculate response correction filter	Enables or disables calculating the response correction filter value.
Apply response correction filter	Sets the type of response correction filters. Valid values are None, matched, and Nyquist.
Front end filter	
Filter type	Sets the type of front end filter, out of Bessel, Butterworth, square root raised cosine, raised cosine, or user-defined 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.

Table 6: Analysis Parameters fields (cont.)

Parameter	Description
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. The display refresh rate is considerably faster when this feature is disabled.
CD	
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: $H(\omega) = e^{i\beta_2 \omega^2/2}$ where $\beta_2 = D_{psnm} \times \frac{10^{12}}{10^9} \frac{\lambda_0^2}{2\pi c}$
Adaptive LMS Filter Controls	
Constant modulus	Enables/disables constant modulus filtering mode. The filter coefficients are calculated to minimize the sum of the squared deviations between the signal moduli and a constant. This method is particularly effective for modulation formats such as QPSK or N-ary PSK that have constellations with a constant modulus. Constant modulus has also been found to work well with QAM signals that have significant impairments. Setting Constant modulus to True automatically forces Radius directed to False.
Radius directed	Enables/disables the radius directed filtering mode. The filter coefficients are calculated to minimize the sum of the squared deviations between an observed symbol modulus and the radius in the constellation that is its closest match. Radius-Directed filter may provide improved performance relative to the Constant modulus for constellations such as QAM8 or QAM16 that have multiple radii. Setting Radius directed to True automatically forces Constant modulus to False.
Number of taps CMA or RD (odd)	Sets the total number of taps to use for the CMA or RD algorithms. ***NOTE.*** *The total number of taps must be an odd number.*

Table 6: Analysis Parameters fields (cont.)

Parameter	Description
Symbol directed	Enables/disables the symbol- directed LMS filtering. The filter coefficients are chosen to minimize the sum of the squared deviations between an observed symbol and its closest match in the constellation. The Symbol-Directed filter can correct a variety of impairments including residual PMD or chromatic dispersion, error from imperfect polarization demultiplexing, or non-ideal transmitter or receiver frequency responses. This filter always reduces the signal EVM.
Number of taps (SD) odd	Sets the total number of taps to be used for the symbol-directed LMS filter, ***NOTE.*** *The total number of taps must be an odd number.*
Use true symbols	Optimize the SD filters based on the known actual symbol locations rather than the nearest symbol locations. This is only possible if the data content is known and synchronization is possible.
PMD Measurement	
PMD	Polarization Mode Dispersion (PMD) measurement. (See page 106, *PMD measurement*.)
Acquire PMD reference	Enables/disables acquisition of the 'back-to'back' waveform used for PMD measurement.
Measure PMD	Enables/disables the PMD measurement.
Number of PMD orders	Sets the number of PMD orders to use when calculating the PMD measurement.
Offline processing	
Use signal description from offline file	When selected, applies signal description parameters (signal type, clock frequency, data content) taken from offline file. When not selected, applies parameters from Analysis Parameters. This control has no effect when processing live data.
Use front end filter from offline file	When selected, applies front end filter parameters taken from offline file. When not selected, applies parameters from Analysis Parameters. This control has no effect when processing live data.
Use calibration from offline file	When selected, applies calibration data (hybrid, equalization calibration) taken from offline file. When not selected, applies calibration data loaded from disk at OMA startup. This control has no effect when processing live data.
Use carrier definition table from offline file	When selected, applies multicarrier carrier definition table taken from offline file. When not selected, applies current carrier definition table in Multicarrier Setup window. This control has no effect when processing live data.
Data Content	

Table 6: Analysis Parameters fields (cont.)

Parameter	Description
Data Content (below parameters listing)	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. If the data pattern is unknown, choose the Unknown entry so that the software does not waste time trying to synchronize to the wrong pattern.
Symbol to Bit Mapping	Select a previously defined mapping from symbol location to bit value for the constellation by choosing it from the drop-down menu. To create a new mapping, click on the box to the right of the drop-down menu to open the UI for creating or importing new symbol to bit mapping files. If no Symbol to Bit mapping file is specified, it is assumed that increasingly positive In-Phase values correspond to increasing In-Phase bit combinations, and increasingly positive Quadrature values correspond to increasing Quadrature bit combinations. For example, for 16QAM, the upper-right symbol location corresponds to the bit combination 1 1 1 1, while the lower-left default would be 0 0 0 0..

Front end filtering

The signal may be filtered according to the settings of the Front end filter group. Adaptive filters are controlled by the System impulse response group for response-correcting filters, or by the Adaptive LMS filter controls group for LMS type filters. 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.

The available filter categories are:

- **Fixed filters**: Bessel (also known as Bessel-Thomson), Butterworth, square root raised cosine, and raised cosine.
- **User-specified** filters.
- **Measured response-correcting filters**: Matched filter, Nyquist filter. These are calculated based on the computed system impulse response. Their controls are located in the System impulse response group.
- **Adaptive LMS Filters**: Constant modulus, Radius directed, or Symbol directed. These controls are located in the Adaptive LMS Filter Controls area. (See page 61, *LMS filtering*.)

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: .Values, .dt and .t0. 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 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 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.

LMS filtering

OMA can perform adaptive filtering of the received signal. This filtering can do a variety of functions, including polarization demultiplexing and the correction of signal impairment resulting from PMD or from the variation in the frequency response of the transmitter or receiver electronics. The adaptive filtering performed by MATLAB CoreProcessing is controlled by setting control variables in the Analysis parameters tab of the OMA software.

The adaptive filtering performed by the OMA software has the general form:

$$x_j = \sum_{k=-M}^{M} \left[H_{XX}^{(k)} X_{j-k} + H_{XY}^{(k)} Y_{j-k} \right]$$

$$y_j = \sum_{k=-M}^{M} \left[H_{YX}^{(k)} X_{j-k} + H_{YY}^{(k)} Y_{j-k} \right]$$

Where:

M = (Number of Taps – 1)/2

$\left\{ H_{XX}^{(k)}, H_{XY}^{(k)}, H_{YX}^{(k)}, H_{YY}^{(k)} \right\}$ are the tap weights.

(X_j, Y_j) are the components of the signal field at the center of the jth symbol slot.

(x_j, y_j) are the components of the filtered result.

OMA determines the tap weights through the adaptive minimization of an objective function represented as a sum of squared deviations.

This filtering is referred to as adaptive Least-Mean-Square (LMS) filtering. The widely-used Constant Modulus Algorithm (CMA) filtering is a type of adaptive LMS filtering.

OMA can perform three types of adaptive LMS filtering: **Constant Modulus**, **Radius-Directed**, and **Symbol-Directed**. The first two filters are for polarization demultiplexing and PMD correction, and some impairment correction. These filters, when enabled, replace the default polarization demultiplexing performed by OMA. (See page 166, *Initial polarization estimate*.) (See page 218, *EstimateSOP*.)This is essential when the received signal has significant PMD, as the default demultiplexing employed by OMA cannot correct for this impairment.

Constant Modulus LMS filtering

The **Constant Modulus** (CMA) method calculates the tap weights to minimize the sum of the squared deviations between the signal moduli and a constant. This method is particularly effective for modulation formats (like QPSK or N-ary PSK) that have constellations with a constant modulus.

To enable the Constant Modulus filter, set the “Constant modulus” field in Adaptive LMS Filter Controls to True. Then set the number of taps in the filter.

NOTE. *The number of taps must be an odd number*

Turning on the CMA filter turns off skew computation unless the pulse shape is specified so that the skew values may be estimated from the filter tap weights. See the Matlab tab description for information on how to specify the pulse shape.

NOTE. *Because OMA run-time increases as the number of taps increases, use the smallest number of taps necessary to obtain a desired performance level. You can begin with one or 3 taps, and then gradually increase the number of taps while monitoring demodulation performance metrics (such as the EVM or BER) for improvement.*

Radius-directed LMS filtering

The second type of filtering is **Radius-directed** adaptive LMS (RDLMS) filtering . The moduli of symbols in a given constellation generally have a discrete set of values (radii). This filtering method calculates the tap weights to minimize the sum of the squared deviations between an observed symbol modulus and the radius in the constellation that is its closest match. Radius-directed filtering is a blind equalization algorithm: the true symbol radii are not known. The Radius-directed filter can improve performance relative to the Constant modulus filter for constellations such as QAM8 or QAM16 that have multiple radii.

To enable the Radius-directed filter, set the “Radius directed” field in Adaptive LMS Filter Controls to True.

Turning on the Radius-directed filter turns off skew computation unless the pulse shape is specified so that the skew values may be estimated from the filter tap weights. See the Matlab tab description for information on how to specify the pulse shape.

NOTE. *The Constant modulus and Radius-directed methods are mutually exclusive: setting one choice to True automatically forces the other choice to False. You are not required to use either filter. Setting both choices to False causes OMA to use the default method for polarization demultiplexing. (See page 166, Initial polarization estimate.) (See page 218, EstimateSOP.)*

Symbol-directed LMS filtering

Symbol-Directed LMS (SDLMS) filtering calculates the tap weights to minimize the sum of the squared deviations between an observed symbol and its closest match in the constellation. This is also a blind equalization algorithm, as the true symbol value is not known. Alternatively, for best performance, the algorithm can use the known data (assuming data synchronization was successful) if the “Use true symbols” field is checked.

OMA evaluates the symbol directed tap weights by solving the normal equations associated with minimizing the sum of squared deviations between the observed symbols and their "true" values.

The Symbol-Directed filter can correct a variety of impairments including residual PMD or chromatic dispersion, error from imperfect polarization demultiplexing, or non-ideal transmitter or receiver frequency responses.

NOTE. *The Symbol-Directed filter always results in a reduction in the EVM of the signal. This reduction may or may not be significant, depending on the degree of impairment in the signal.*

To enable this filter, set the “Symbol directed” selection to True. This filter is independent from the constant modulus or Radius-Directed filters and can be used in combination with them or on its own.

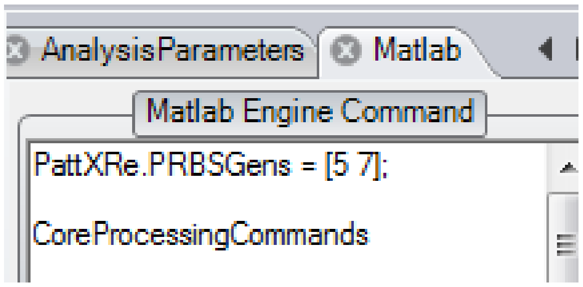
Keep in mind that the measurement panel displays results after all processing is complete. The values shown are after the application of the SDLMS filter.

NOTE. *OMA run-time increases with the number of taps, so select the smallest number of taps needed to correct the impairments present.*

NOTE. *If you save files using the Record function, the tap weights from that session/run are saved in the files.*

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 can assign the PRBS polynomial directly in the MATLAB Engine Command Window in the OMA. 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 is overridden by the statement in the MATLAB Engine Window. Any PRBS polynomial can be specified in this manner, enabling the use of different sequences having the same length. For example, [5 9] and [4 6 7 9] are both valid 2^{9-1} PRBS sequences.



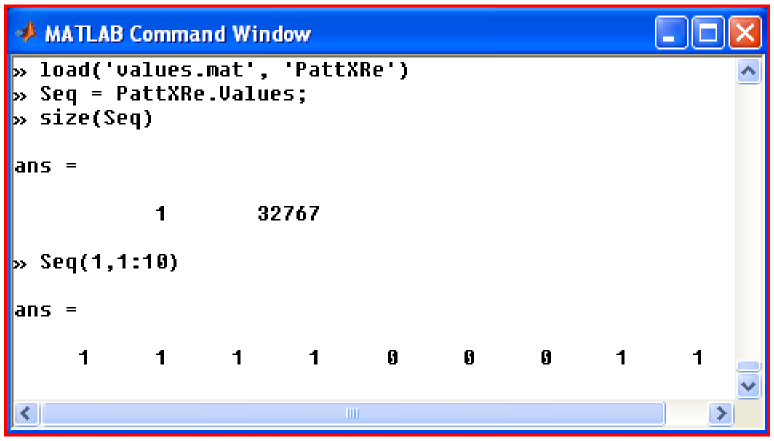
```
PattXRe.PRBSGens = [5 7];

CoreProcessingCommands
```

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 precode, the analyzer must know what data is being sent 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;

PattXlm.Values = Seq2;

PattXlm.SyncFrameEnd = 100;

PattYRe.Values = Seq3;

PattYRe.SyncFrameEnd = 100;

PattYlm.Values = Seq4;

PattYlm.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.



```
>> 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 and 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 OMA 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 OMA.
2. Set up the Analysis Parameters properly except for the data pattern which is not yet known.
3. Choose **Unknown** as the data pattern (do not choose "User Pattern" yet). Optimize the signal for 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 oscilloscope 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 an adequate 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 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’)
   ```

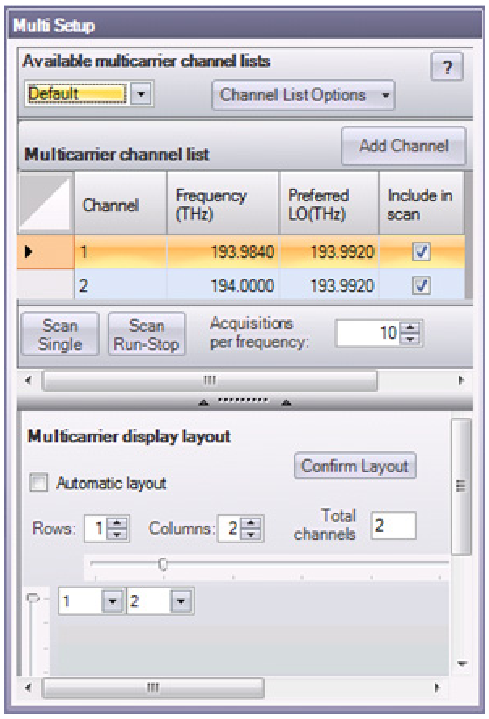
The Multicarrier Setup window

The Multicarrier Setup tab defines the carrier channel plan, to start and stop an automatic scan, and to define which channels to include in the separate Multicarrier Eye and Multicarrier Constellation axis plots. 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 OMA displays the Multicarrier Setup button on the Setup ribbon.

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

NOTE. *Click the Multi Carrier setup button in the Layout region of the Home tab to quickly arrange the screen for multicarrier measurements.*


Multi Setup
Available multicarrier channel lists
?
Channel List Options
Multicarrier channel list
Add Channel
Scan Single
Scan Run-Stop
Multicarrier display layout
Confirm Layout

Default

Channel	Frequency (THz)	Preferred LO(THz)	Include in scan
1	193.9840	193.9920	✓
2	194.0000	193.9920	✓

Acquisitions per frequency: 10

Automatic layout

Rows: 1 Columns: 2 Total channels 2

1 2

Figure 5: 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 OMA 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 is 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 OMA 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 increment 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.

You can enter several channel definition tables, and use the drop-down menu at the top left to select which one to apply. You can delete tables from the Channel List Options button.

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

Multicarrier display layout

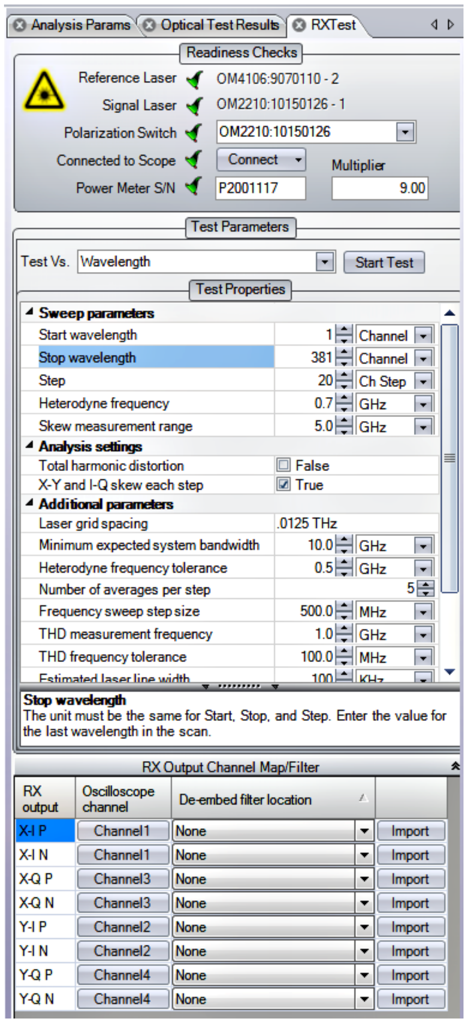
This section sets which plots to add in the separate axis plots, and how to arrange the subplots. The Automatic layout check box (enabled by default) lets OMA decide how to arrange the subplots. When Automatic layout is not checked, the region below becomes active. You can choose the number of columns and rows of subplots either from the named controls, or by moving the horizontal and vertical sliders. Use the drop-down menu to select the channel number to be plotted in each subplot.

The Receiver Test configuration tab

Click the **Receiver Test** button to open the **RXTest** configuration tab. This tab is for setting up the OMA for receiver tests, and includes checks to verify that the receiver test is ready to run, and receiver-related test parameters settings. Click a parameter to display a brief explanation of that item in the lower help text pane.

NOTE. *Use the Receiver Test button on the Home Ribbon to display the correct plot and control tabs for Receiver Test operation.*

NOTE. *Receiver tests require a Tektronix OM2210.*



Reference Laser: OM4106:9070110 - 2
Signal Laser: OM2210:10150126 - 1
Polarization Switch: OM2210:10150126
Connected to Scope
Power Meter S/N: P2001117
Multiplier: 9.00

Test Vs.: Wavelength

Sweep parameters		
Start wavelength	1	Channel
Stop wavelength	381	Channel
Step	20	Ch Step
Heterodyne frequency	0.7	GHz
Skew measurement range	5.0	GHz
Analysis settings		
Total harmonic distortion	☐ False	
X-Y and I-Q skew each step	☑ True	
Additional parameters		
Laser grid spacing	.0125 THz	
Minimum expected system bandwidth	10.0	GHz
Heterodyne frequency tolerance	0.5	GHz
Number of averages per step	5	
Frequency sweep step size	500.0	MHz
THD measurement frequency	1.0	GHz
THD frequency tolerance	100.0	MHz
Estimated laser line width	100	KHz

Stop wavelength
The unit must be the same for Start, Stop, and Step. Enter the value for the last wavelength in the scan.

RX output	Oscilloscope channel	De-embed filter location	
X-I P	Channel1	None	Import
X-I N	Channel1	None	Import
X-Q P	Channel3	None	Import
X-Q N	Channel3	None	Import
Y-I P	Channel2	None	Import
Y-I N	Channel2	None	Import
Y-Q P	Channel4	None	Import
Y-Q N	Channel4	None	Import

Table 7: RXTest parameters

Parameter	Description
Start wavelength	Choose units and enter the starting value for the wavelength scan. If using channel numbers, they are based on the grid chosen above.
Stop wavelength	The unit must be the same for Start, Stop, and Step. Enter the value for the last wavelength in the scan.
Step	The unit must be the same for Start, Stop, and Step. Enter the value of the increment between steps in the scan.
Heterodyne Frequency	Target heterodyne difference frequency between Signal and Reference lasers at each wavelength step.
Skew measurement range	Skew is measured first to enable measurement of phase angles with skew removed. The skew is calculated by measuring phase over the frequency range indicated and extracting the slope.
Total harmonic distortion	Check the box to include THD measurement. Additional laser tuning is necessary for the heterodyne frequency target if included. See settings for heterodyne target in Additional Parameters.
X-Y and I-Q skew each Step	Check the box to measure the skew values at each wavelength step. Measuring skew values adds substantial test time due to the additional frequency sweeps.
Laser grid spacing	The laser grid is set to the specified value to allow for continuous tuning between grid frequencies. The grid is set back to the original value when the test is complete.
Minimum expected system bandwidth	The measurement stops if a low signal is detected within the minimum expected system bandwidth of the combined coherent receiver and oscilloscope system.
Heterodyne frequency tolerance	The laser frequencies are adjusted to obtain the target heterodyne frequency to within this tolerance before taking data. Keeping a consistent heterodyne frequency removes frequency domain effects from the wavelength plots. Tolerance below 200 MHz will increase test time.
Number of averages per step	The number of measurements to average when computing each wavelength entry for the calibration table.
Frequency sweep step size	Frequency step size for the skew measurement frequency sweep.
THD measurement frequency	The heterodyne frequency target for the THD test.
THD frequency tolerance	The laser frequencies are adjusted to obtain the target heterodyne frequency to within this tolerance before taking data. Keeping a consistent heterodyne frequency results in greater reproducibility. Tolerance below 200 MHz will increase test time.

Table 7: RXTest parameters (cont.)

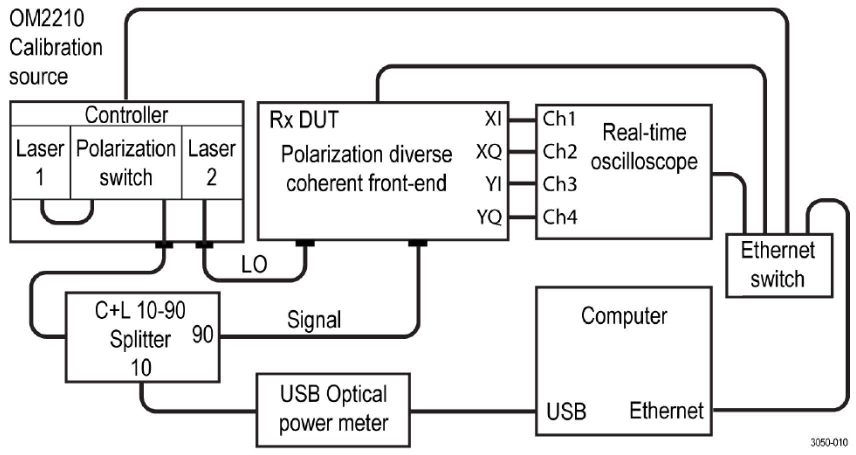
Parameter	Description
Estimated laser line width	This value is only used for the calculation of the entries for the calibration table output file, pHybCalib.mat. The laser linewidth is used to help in recovering the phase of the heterodyne signals.
Oscilloscope record length	The number of points to be acquired by the oscilloscope for each measurement. 200,000 is recommended when producing a pHybCalib.mat file during a wavelength sweep. Shorter record length down to 20,000 is faster for other measurements
TIA gain control mode	If the receiver has a Transimpedance Amplifier, it must be set to constant gain mode to provide meaningful results for some measurements.
Use default folder location	The MATLAB results of the RxTest run are stored in the default directory if this is checked. Otherwise, the user is prompted for a location and file information. A dialog box will indicate the destination folder at the end of the run.
Sweep Range for Frequency Response	The highest frequency to include in the frequency response measurement. Measurement will start at this modulation frequency and stop at 500 MHz.
Include P and N measurements	If your receiver has differential outputs, you may chose to include the P, N, or both P and N outputs in the measurement. If including both, you are prompted to reconnect cables twice during the measurement unless the Tri Mode probe is selected. P and N measurements are only allowed for the test versus Modulation Frequency. The Wavelength Sweep test assumes P outputs only.
Find low frequency cutoff	If checked, the oscilloscope settings are changed to measure very low frequencies to calculate a low cut off frequency. Because of laser frequency drift, many data acquisitions are required to obtain low frequency data. As a result, this test can take a significant amount of time (> 30 minutes).
Use Tri Mode probe	If this is checked and P/N measurements are included, a command is sent to the oscilloscope to switch the Tri Mode probe between inputs A and B to measure the P and N outputs of the differential receiver.

To perform a receiver test

NOTE. *Receiver tests require a Tektronix OM2210 Laser Calibration Source and the appropriate power meter software loaded.*

NOTE. *Only real-time DPO/MSO70000 series oscilloscopes are supported for the Receiver Test application at this time.*

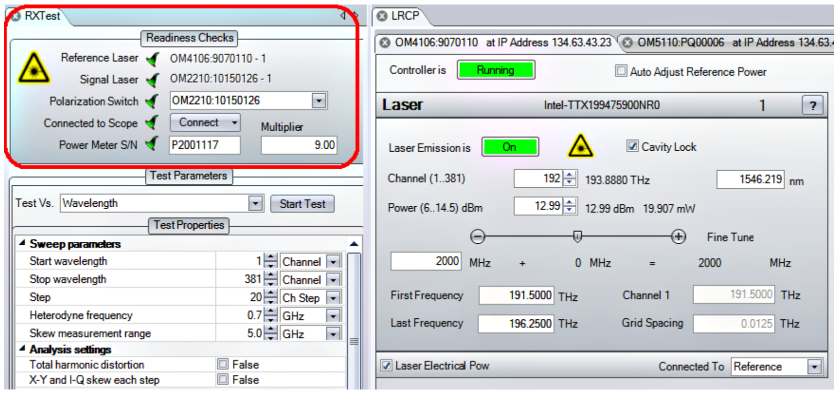
If the receiver DUT has differential P and N outputs, connect the P outputs first. P and N measurements are supported for all frequency response measurements. Wavelength sweep measurements assume that only P outputs are connected.



Hardware readiness checks

To initiate a wavelength sweep to measure parameters such as quadrature phase angle vs. wavelength, it is necessary to configure the OMA for the hardware set up and the measurement parameters such as start and stop wavelength. Click **Home > Receiver Test** to lay out the screen and open the RxTest tab that has all of the configuration settings.

Complete the Readiness Checks in the upper part of the RxTest tab by identifying the required hardware:



RXTest

Readiness Checks

Reference Laser ✓ OM4106:9070110 - 1
Signal Laser ✓ OM2210:10150126 - 1
Polarization Switch ✓ OM2210:10150126
Connected to Scope ✓ Connect
Power Meter S/N ✓ P2001117
Multiplier 9.00

Test Parameters

Test Vs. Wavelength
Start Test

Test Properties

Sweep parameters		
Start wavelength	1	Channel
Stop wavelength	381	Channel
Step	20	Ch Step
Heterodyne frequency	0.7	GHz
Skew measurement range	5.0	GHz
Analysis settings		
Total harmonic distortion	False	
X-Y and I-Q skew each step	False	

LRCP

OM4106:9070110 at IP Address 134.63.43.23
OM5110:PQ00006 at IP Address 134.63.

Controller is Running
Auto Adjust Reference Power

Laser Intel-TTX199475900NR0 1

Laser Emission is On
Cavity Lock ✓
Channel (1..381) 192 193.8880 THz 1546.219 nm
Power (6..14.5) dBm 12.99 12.99 dBm 19.907 mW
Fine Tune
2000 MHz + 0 MHz = 2000 MHz
First Frequency 191.5000 THz Channel 1 191.5000 THz
Last Frequency 196.2500 THz Grid Spacing 0.0125 THz

Laser Electrical Pow ✓ Connected To Reference

1. Click the ‘Click to Select Reference Laser’ link in the Readiness Checks area. Use the LRCP controls to select and connect to the laser to be used as the Reference Laser (also called the Local Oscillator or LO). Select it as the Reference (using the drop down menu in the LRCP tab). The Reference Laser readiness status changes to a green check mark when the connection to the instrument is confirmed.

2. Click the ‘Click to Select Signal Laser’ link in the Readiness Checks area. Use the LRCP controls to select and connect to the OM instrument to use as the Signal Laser/Polarization switch inputs. Select it as the Signal Laser (using the drop down menu in the LRCP tab). The Signal Laser readiness status changes to a green check mark when the connection to the instrument is confirmed.

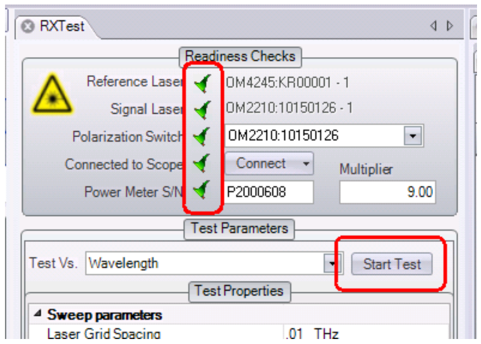
3. Click the **Scope Connect** button. If you are using a DPO720004C or later model real-time scope, makes sure that the Scope Service Utility on the oscilloscope is installed and running before connecting OMA to the oscilloscope. For older oscilloscopes, use the TekVISA connection method. (See page 4, *To install TekVISA software*.)

4. When connecting to the oscilloscope using the Connect button, set which receiver outputs are connected to which oscilloscope inputs, and enable all of the channels to be measured. Wavelength sweeps require four channels, and the Modulation Frequency test allows two or four channels.

5. Once connected, OMA populates the Rx Output Channel Map/ Filter area. Verify that the receiver DUT is connected to the specified scope channels. Each receiver output may have a filter designated. The filter may be constructed using SDLA or other methods to correct for fixture frequency response and skew.

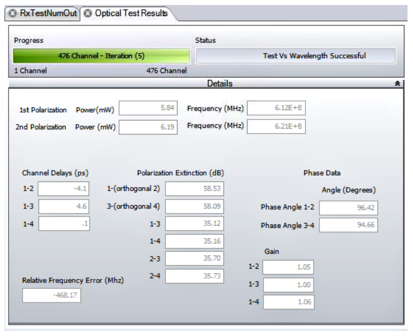
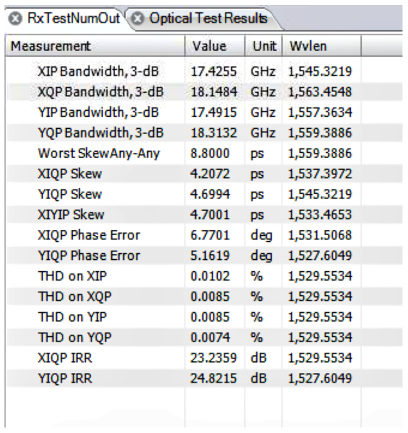
6. Enter the serial number and multiplier value of the USB Power Meter in the **Power Meter S/N** and **Multiplier** fields to satisfy the final readiness check. If the green check mark does not appear, it usually means there is a connection or driver problem with the power meter. The multiplier is the power ratio between the DUT signal input fiber and the Power meter fiber. Normally, a 90/10 splitter is used with 90 percent of the power going to the DUT, so the default value is 9. However, for sensitive receivers, you may swap the connections to get lower power at the DUT input. In this case the Multiplier would be 0.111.

7. Once all of the hardware is configured and green check marks are showing for all of the Readiness Checks, OMA enables the Start Test button.



Receiver test readouts

Table 8: Receiver test readout tabs

Readout	Description
Optical Test Results 	The Optical Test Results panel shows the progress and status of the test. The details panel below the progress bars can be opened to view intermediate results or hidden using the arrow button to the far right of the “Details” title.
RxTestNumOut 	The Rx Test Num Out panel shows the worst case values versus wavelength for each of the measurements listed. The value and the wavelength where that value was measured are displayed in each row.

RxTestNumOut | Optical Test Results

Progress: 476 Channel - Iteration (5)
1 Channel … 476 Channel
Status: Test Vs Wavelength Successful

Details

1st Polarization	Power(mW)	5.84	Frequency (MHz)	6.12E+8
2nd Polarization	Power (mW)	6.19	Frequency (MHz)	6.21E+8

Channel Delays (ps)

1-2	-4.1
1-3	4.6
1-4	.1

Polarization Extinction (dB)

1-(orthogonal 2)	58.53
3-(orthogonal 4)	58.09
1-3	35.12
1-4	35.16
2-3	35.70
2-4	35.73

Phase Data

	Angle (Degrees)
Phase Angle 1-2	96.42
Phase Angle 3-4	94.66

Gain

1-2	1.05
1-3	1.00
1-4	1.06

Relative Frequency Error (Mhz): -468.17

RxTestNumOut | Optical Test Results

Measurement	Value	Unit	Wvlen
XIP Bandwidth, 3-dB	17.4255	GHz	1,545.3219
XQP Bandwidth, 3-dB	18.1484	GHz	1,563.4548
YIP Bandwidth, 3-dB	17.4915	GHz	1,557.3634
YQP Bandwidth, 3-dB	18.3132	GHz	1,559.3886
Worst Skew Any-Any	8.8000	ps	1,559.3886
XIQP Skew	4.2072	ps	1,537.3972
YIQP Skew	4.6994	ps	1,545.3219
XIYIP Skew	4.7001	ps	1,533.4653
XIQP Phase Error	6.7701	deg	1,531.5068
YIQP Phase Error	5.1619	deg	1,527.6049
THD on XIP	0.0102	%	1,529.5534
THD on XQP	0.0085	%	1,529.5534
THD on YIP	0.0085	%	1,529.5534
THD on YQP	0.0074	%	1,529.5534
XIQP IRR	23.2359	dB	1,529.5534
YIQP IRR	24.8215	dB	1,527.6049

Wavelength Sweep measurements

1. Select Test Vs. Wavelength (RXTest tab, Test Parameters area) to perform the wavelength sweep.
2. Set "X-Y and I-Q Skew each step" to True. This substantially increases test time, but provides complete frequency response data over the "Skew measurement range" in addition to wavelength data for each wavelength tested.

NOTE. *For receivers with automatic gain control, it is important to set the gain control to fixed gain. If the automatic gain control loop is operating, it will confuse calculations such as receiver gain. Some tests such as Skew will still function with automatic gain control, but it is best to switch it off.*

3. Click **Start Test** to begin the wavelength sweep.

The test time depends on the selected parameters and settings. If they are selected, the following steps are performed:

1. Set laser grid. The laser tuning grid must be set to a value low enough to permit continuous tuning between grid points. 10 GHz and 12.5 GHz grid spacings are available. The grid is set back to the original setting at the end of the test.
2. Find frequency error. The lasers may have a fixed frequency error (an offset between the laser frequency setting and the actual laser frequency). This is measured by the oscilloscope after the lasers are tuned to a particular frequency offset.
3. Measure skew. To extract the receiver phase angles, it is necessary to first deskew the system. If the oscilloscope and cables have already been deskewed, then the skew measured will be that of the DUT. This is only measured once at the starting wavelength unless the "X-Y and I-Q Skew each step" is selected. The skew values are determined by performing two laser heterodyne frequency sweeps, one for each polarization.
4. Measure phase and amplitude data at frequency target. To remove the effect of frequency response on a wavelength sweep, the lasers are tuned to the same heterodyne frequency at each wavelength step as determined by the Heterodyne Frequency target. The measurements are repeated five times by default unless changed in the Additional Parameters section. Values outside the specified tolerance are omitted.
5. Measure THD at the THD frequency target. The lasers are re-tuned to the heterodyne frequency specified by the THD frequency target. THD is measured. Values outside the specified THD frequency tolerance are omitted.

NOTE. *OIF-DPC-Rx 01.2 specifies a frequency target of 1 GHz and tolerance of 0.1 GHz for this measurement*

The test status is shown in the Optical Test Results tab. At the end of the test, the data is shown in the plots. The RxTestNumOut tab displays the worst-case measurements that were the found during the wavelength sweep.

Optical hybrid calibration. The Wavelength Sweep may also be used to produce a calibration file called pHybCalib.mat which is used by the OMA. This file is used to correct for coherent receiver impairments when the receiver is being used as part of an Optical Modulation Analysis (OMA) system. You may indicate where this file should be stored using the "Use default folder location" check box in the Additional parameters section. If this is not checked, you will be prompted for a storage location and info about the device to be stored in the file.

If you have previously used our Hybrid Receiver Calibration (HRC) software, you may wish to see the plots in this format for comparison against past results. These plots are still available for backward compatibility. To see the HRC plots, type 'ShowHRCplots = true;' in the separate Matlab application window (do not enter the quote marks).

Wavelength sweep plots. All wavelength sweep plots assume the DUT has single-ended outputs or that only the P-outputs are connected if the DUT has differential outputs.

You can measure wavelength properties of the N-outputs if these ports are connected to the oscilloscope. The results are plotted and reported as P-outputs.

Table 9: RxTest: Wavelength sweep plots

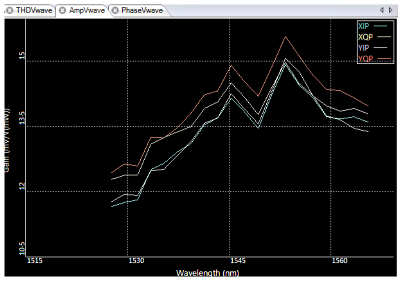
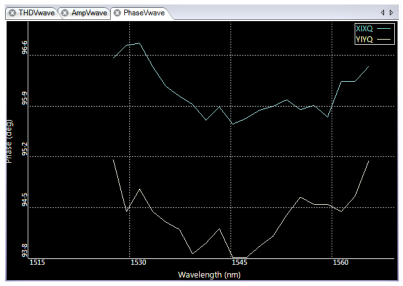
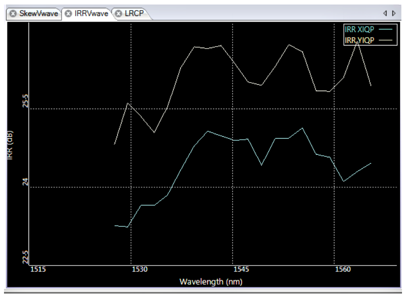
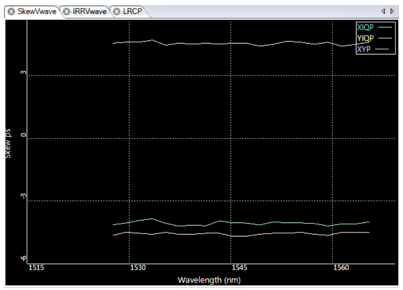
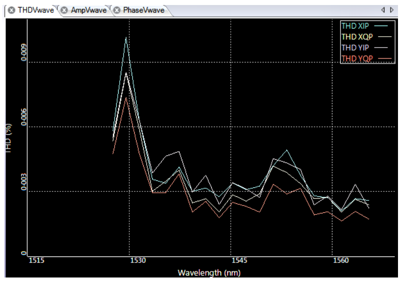
Plot	Description
Amplitude Vs. Wavelength 	Gain for each channel is computed based on the measured input power and output voltages when the DUT is excited by two orthogonal signal input states. Right-click options: ■ Gain in *mV* per √*mW* of signal power ■ Relative gain from channel to channel ■ Turn traces and markers on and off
Phase Vs. Wavelength 	The effect of skew is removed to provide the phase angle near zero modulation frequency. The signal input polarization state which provides the highest SNR is chosen for the phase measurement. Right-click options: Turn traces and markers on and off.

Table 9: RxTest: Wavelength sweep plots (cont.)

Plot	Description
IRR Vs. Wavelength 	The image rejection ratio is found by taking 10 times the log^{10} of the power ratio at the signal and image frequencies at the target heterodyne frequency. The effect of skew is not removed from this measurement. Deskew the oscilloscope and test fixture using the oscilloscope UI or by providing the appropriate De-embed filter to get the best result for the DUT. Right-click options: Turn traces and markers on and off.
Skew Vs. Wavelength 	Check the RxTest control "X-Y and I-Q skew each step" to get valid Skew Vs. Wavelength data for each wavelength measured. Unchecking this box will yield valid data only at the first wavelength. The effect of skew is not removed from this measurement. Deskew the oscilloscope and test fixture using the oscilloscope UI or by providing the appropriate De-embed filter to get the best result for the DUT. Right-click options: Turn traces and markers on and off.
THD Vs. Wavelength 	The Total Harmonic Distortion (THD) is measured at the target heterodyne frequency with the tolerance specified in the RxTest Additional Parameters area. ***NOTE.*** *OIF-DPC-Rx 01.2 specifies a frequency target of 1 GHz and tolerance of 0.1 GHz for this measurement*

Modulation frequency sweep measurements

The modulation frequency sweep measurements perform a heterodyne frequency sweep to find the frequency response of the receiver at a particular wavelength. The term “Modulation Frequency” is used in contrast to measurements vs. optical wavelength. The modulation used is the heterodyne beat frequency between the Signal and Reference laser frequencies.

Many of the modulation frequency tests can be performed during a wavelength sweep, so that both modulation and optical properties can be measured simultaneously. The separate Test Vs. Modulation Frequency provides a more in-depth analysis at a particular wavelength.

“Include P and N measurements:” Unlike the Test vs. Wavelength, the Test vs. Modulation Frequency enables testing of P and N response. Check “Use TriMode probe” if you will use a TriMode probe to connect to the P and N outputs of the DUT. The A input should be connected to the DUT “P” output. If a TriMode probe is not available for each channel, the P and N responses can still be measured by moving the cable connections as prompted by the OMA. The OMA prompts the user to change the cable connections two times. This permits calculation of P vs N measurements such as P-N skew.

NOTE. *For receivers with automatic gain control, it is important to set the gain control to fixed gain. If the automatic gain control loop is operating, it will confuse calculations such as receiver gain. Some tests such as Skew will still function with automatic gain control, but it is best to switch it off.*

Select the modulation test to perform, set any parameters, and click Start Test to run the modulation frequency sweep measurements.

The test time depends on the options selected. If they are selected, the following steps are performed:

1. Set laser grid. The laser tuning grid must be set to a value low enough to permit continuous tuning between grid points. 10 GHz and 12.5 GHz grid spacings are available. The grid is set back to the original setting at the end of the test.
2. Find frequency error. The lasers may have a fixed frequency error (an offset between the laser frequency setting and the actual laser frequency). This is measured by the oscilloscope after the lasers are tuned to a particular frequency offset.
3. Measure frequency response over indicated sweep range with indicated step size. A subset of the “Sweep range for frequency response” may be specified as the “Sweep range for skew measurement” if desired.

4. Toggle TriMode probe or prompt user to move cable connections if P and N are to be measured. Repeat step 3 two times.

5. Measure low frequency cutoff. If this feature is selected, the lasers tune to the same frequency and data is collected at a very low sample rate to find the low frequency corner of the DUT. Because of laser frequency drift, many data acquisitions are required to obtain low frequency data. As a result, this test can require a significant amount of time (> 30 minutes). Given the long test time, this test is omitted from the P/N testing loop. Repeat the test as needed to measure P and N low frequency cutoff.

The test status is shown in the Optical Test Results tab. At the end of the test, the data is shown in the plots. The RxTestNumOut tab displays the worst-case measurements found during the testing.

EqFilt calibration. The Frequency Sweep is used to produce a calibration file called EqFilt.Mat, which is used by the Optical Modulation Analysis (OMA). The file corrects the coherent receiver impairments when the receiver is being used as part of an OMA system. The CreateCalibrationCoef function uses the s12 results of the modulation versus frequency test and generates the EqFilt file at specified sampling rates.

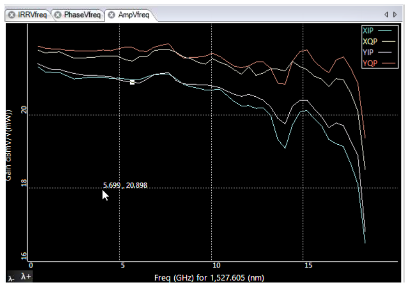
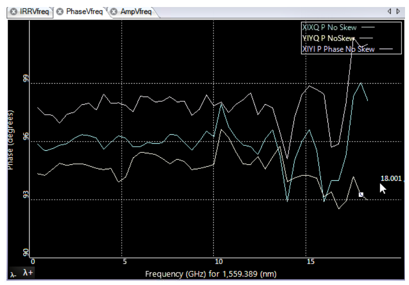
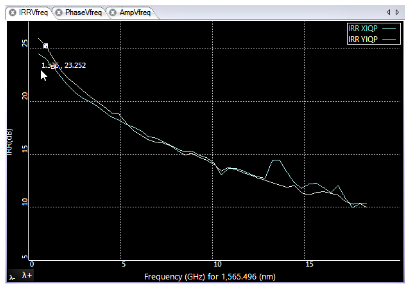
Syntax: CreateCalibrationCoef (FreqStart, FreqStop, Freq3dB, dts, ResultsLoc, RxTestRslts)

Example: CreateCalibrationCoef(606e6,10e9,8e9,[0.625, 6.25, 5.0, 3.125, 2.5, 2.0, 1.25]*1e-12,'c:\',RxTestRslts)

The CreateCalibrationCoef function is executed in the MATLAB runtime command window.

Modulation frequency sweep plots. All of the frequency sweep plots have right-click options to display P, N, or both P and N measurements (if available). Set P, N, or P+N data collection in the RxTest control.

Table 10: RxTest: Modulation Frequency sweep plots

Plot	Description
Amplitude Vs. Frequency 	Gain for each channel is computed based on the measured input power and output voltages when the DUT is excited by two orthogonal signal input states. Right-click options: ■ Gain in *mV* per √*mW* of signal power ■ Relative gain from channel to channel ■ Amplitude and relative amplitude ■ Turn traces and markers on and off
Phase Vs. Frequency 	The effect of skew can be removed to provide the phase angle near zero modulation frequency, or included to see the effect and measurement of skew. The signal input polarization state which provides the highest SNR is chosen for the phase measurement. Right-click options: ■ Phase (degrees) ■ Phase with skew effect removed ■ Phase Error with skew effect removed ■ Turn traces and markers on and off
IRR Vs. Frequency 	The image rejection ratio is found by taking the power ratio at the signal and image frequencies at the target heterodyne frequency. The effect of skew is not removed from this measurement. Deskew the oscilloscope and test fixture using the oscilloscope UI or by providing the appropriate De-embed filter to get the best result for the DUT. Right-click options: ■ Turn traces and markers on and off

State controls

The State controls save or load the current OMA system hardware and/or software state parameters to an .xml file. When hardware state information is loaded, an IP connection will be made to the target hardware so that it can be configured. This means that the hardware must be at the same IP address as it was when the state was saved or the IP address must be updated using the Optical Connect > Auto Configure feature. Using reserved or fixed IP addresses will make it easier to reliably restore hardware configuration.

The OMA software state information consists of the following:

- All Analysis Parameters Settings
- Record Length and Block Size
- Auto Connect State and Auto Configure State from scope connect dialog box
- Matlab window contents
- OUI global display scales
- Reference laser frequency

The OMA hardware state information consists of the following:

- All LRCP settings are considered hardware settings. The Reference laser frequency is considered a software setting as well so will be loaded if either a hardware or software state is loaded
- Information needed to connect to the scope such as IP address
- All oscilloscope settings are part of the hardware state. These are saved on the target oscilloscope using a setup file with the same name as the hardware state xml file
- OMA system deskew values from the Calibration tab
- Oscilloscope channel mapping and which channels are enabled

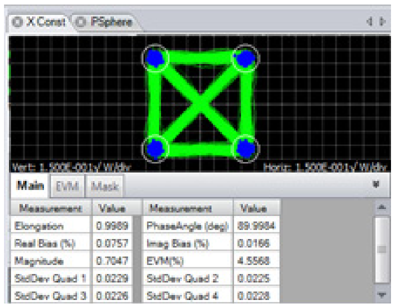
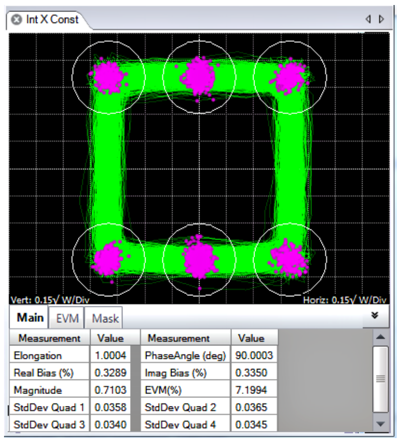
The Home tab

The Home tab lets you select the plots to display in the Plots panel.

Constellation plots

Click the **Constellation** plot button (in the Home controls) to display the list of available plots.

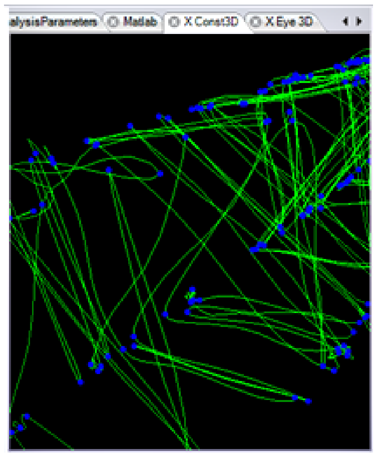
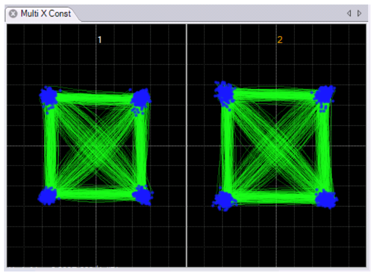
Table 11: Constellation plots

Plot type	Description
X, Y Constellation 	Displays the constellation diagram for X or Y signal polarization with numerical readout bottom tabs. Symbol-center values are shown in blue. Symbol errors are shown in red. Right-click for other color options. Right-click to see plot display options.
Intermediate X, Y Constellation 	Displays offset (intermediate) polarization plots. Both polarization and quadrature offset formats are available. The Y polarization is a half-symbol offset from the X polarization; the standard “Y const” display is empty.

Measurement	Value	Measurement	Value
Elongation	0.9989	PhaseAngle (deg)	89.9984
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

Measurement	Value	Measurement	Value
Elongation	1.0004	PhaseAngle (deg)	90.0003
Real Bias (%)	0.3289	Imag Bias (%)	0.3350
Magnitude	0.7103	EVM(%)	7.1994
StdDev Quad 1	0.0358	StdDev Quad 2	0.0365
StdDev Quad 3	0.0340	StdDev Quad 4	0.0345

Table 11: Constellation plots (cont.)

Plot type	Description
3D X Constellation, 3D Y Constellation 	Displays 3D Constellation for X or Y signal polarization. Displays the constellation diagram with a time axis. You can scale and rotate to view on a 2D or 3D monitor.
Multicarrier X Constellation, Multicarrier Y Constellation (Requires Option MCS) 	Displays the constellation diagram for each multicarrier channel. This feature requires Option MCS. As each channel is analyzed, only that part of the plot is 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 is displayed. Use the Clear Data button to discard prior data. The layout is controlled by the Multicarrier Setup window. (See page 67.)

About constellation diagrams

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 OMA, 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. You can measure the scatter by symbol standard deviation, error vector magnitude, or mask violations.

Constellation measurements

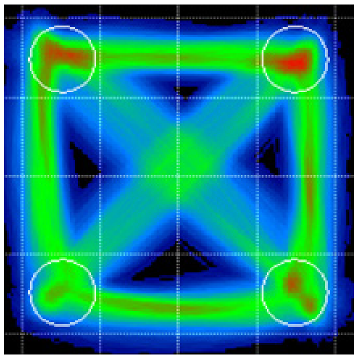
Measurements made on constellation diagrams are the most comprehensive in the OMA. Numerical measurements are available on the flyout panel associated with each graphic window and also summarized in the Measurements Panel. The available constellation measurements are:

- **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. This is related to IQ Gain Imbalance, which is 1/Elongation expressed in dB. IQ Gain Imbalance is reported in the Measurement Statistics plot.
- **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. Another measure of constellation bias is IQ Offset which is reported in the Measurement Statistics plot.
- **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. The deviation from 90 degrees is reported in the Measurement Statistics plot as Quadrature Error.
- **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 average distance of each symbol point from the ideal symbol point divided by the magnitude of the largest ideal symbol expressed as a percent.
- **EVM Tab**: The separate EVM tab 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.

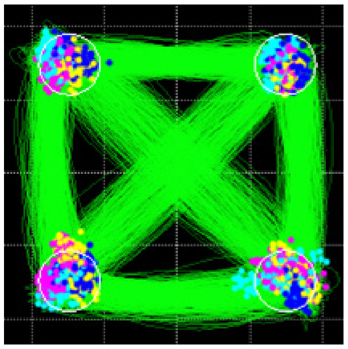
The Q calculation can cause alerts if it can't calculate a Q factor for the outer transitions. For example, in 32-QAM. 32-QAM is a subset of 64-QAM, where the outer constellation points are never used. It is not possible to calculate a Q factor for those outer slices, hence the alert. The subconstellation identification feature notices the unused constellation points, and removes them from the relevant constellation parameters (zXSym.Mean, zXSym.ConstPtMean, and so on), 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 127, QDecTh.)

Color features in constellation plots

Right-click any constellation plot to show a list of display options including Color Grade, Display Traces in Color Grade, and Color Key Constellation Points.



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. Use the right-click context menu in each plot to clear or set the color grade mode (requires nVidia graphic card on the PC).



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**

Multicarrier constellation plots (Requires Option MCS)

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.

This feature requires Option MCS.

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

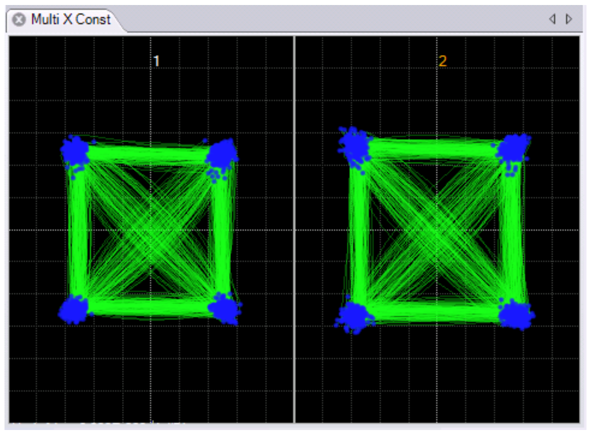


Figure 6: Multicarrier constellation plots

Eye plots

Click the **Eye** plot button (in the Home controls) to display the list of available plots.

Table 12: Eye plots

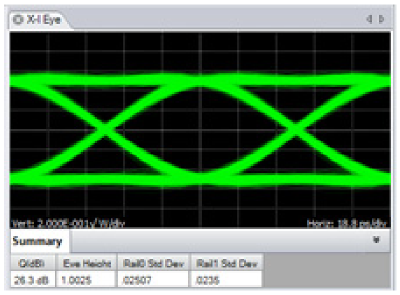
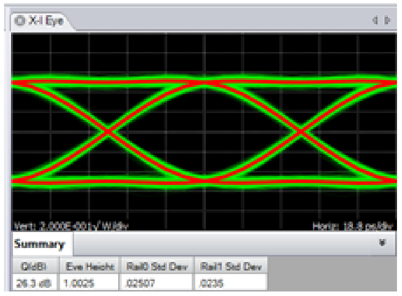
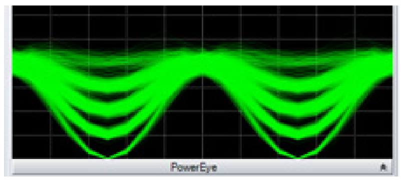
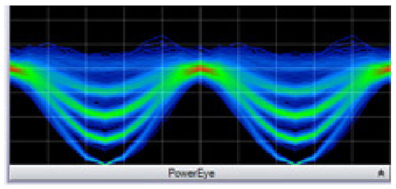
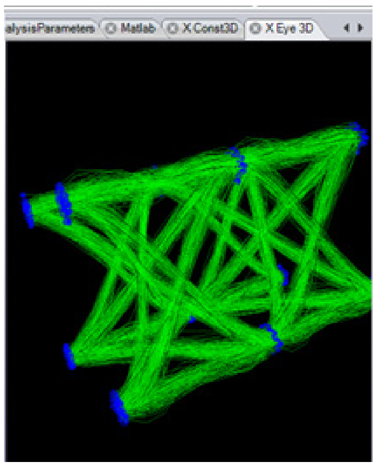
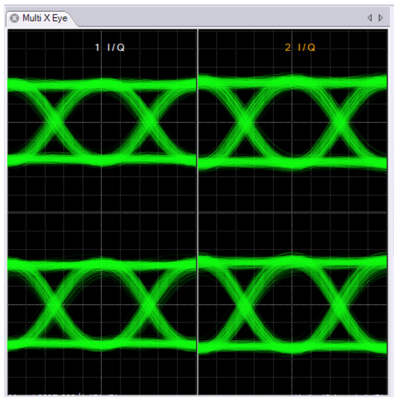
Plot type	Description
X Eye, Y Eye  X-I Eye Vert: 2.000E-001V W/div Horiz: 18.8 ps/div Summary Q(dB) Eye Height Rail0 Std Dev Rail1 Std Dev 26.3 dB 1.0025 .02507 .0235	The coherent eye diagram for X or Y signal polarization shows the In-Phase or Quadrature components versus time modulo two bit periods. Click the Measurements bar at the bottom of the menu to display associated measurements. Available X and Y Eye plots are: ■ Inphase Eye ■ Quadrature Eye ■ Differential Inphase Eye ■ Differential Quadrature Eye
 X-I Eye Vert: 2.000E-001V W/div Horiz: 18.8 ps/div Summary Q(dB) Eye Height Rail0 Std Dev Rail1 Std Dev 26.3 dB 1.0025 .02507 .0235	Right-click the plot to list 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.
Power Eye  	Displays 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. Available Power Eye plots are: ■ Power Eye ■ X Power Eye ■ Y Power Eye Right-click the plot to list options including color grade.

Table 12: Eye plots (cont.)

Plot type	Description
3D X Eye, 3D Y Eye 	Displays the 3D representation constellation diagram with a time axis modulo 2 bit periods, for X or Y signal polarization. You can scale and rotate the plot to view on a 2D or 3D monitor.
Multicarrier X, Multicarrier Y Eye 	Displays eye plots for each channel. This feature requires Option MCS. As each channel is analyzed, only that part of the plot is updated while the most recent data displayed continues to be shown in the other regions to display an aggregate view of the multicarrier group. Use the Clear Data button to discard prior data. The layout is controlled by the Multicarrier Setup window. (See page 67.)

About eye diagrams

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). Right-click to select display color options.

The field Eye diagram provides the following measurements:

- Q (dB): Computed from 20*Log10 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

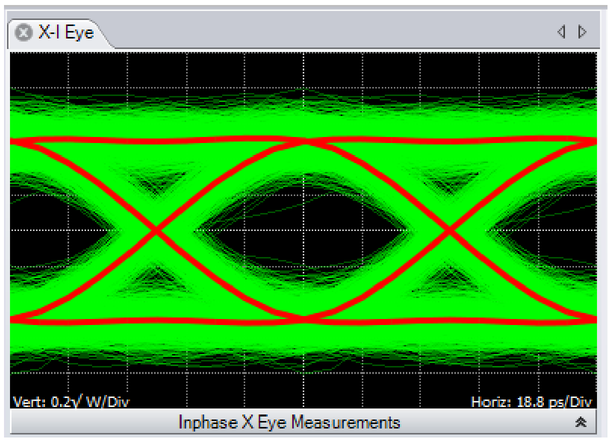
In the case of multi-level signals, the above measurements are listed in the order of the corresponding eye openings in the plot. The top row values correspond to the top-most eye opening.

The above functions involving Q factor use the decision threshold method described in the paper by Bergano [1]. 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.

[1] 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).

Eye waveform averaging

Two types of averaged signal display are available for eye diagrams and signal vs. time. 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.

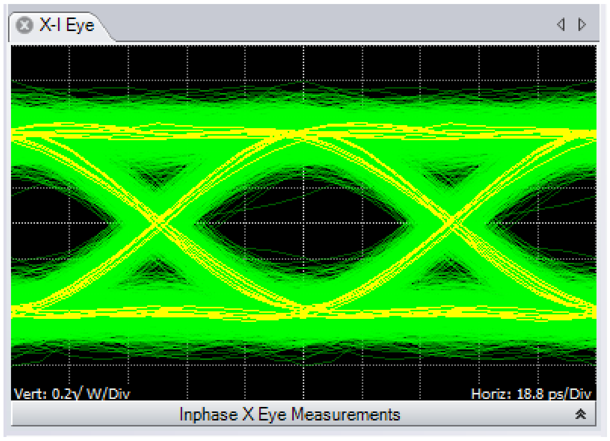


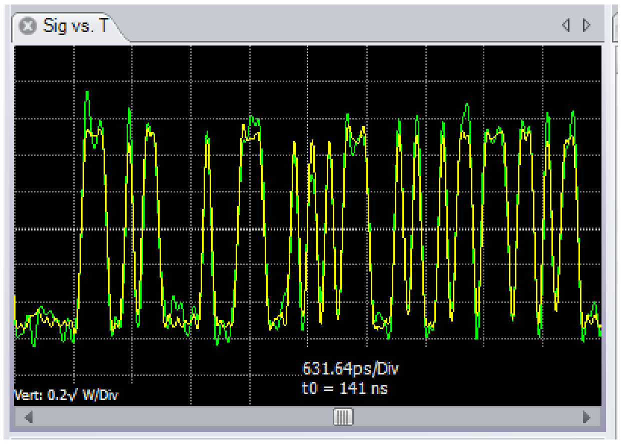
The transition parameters listed in the X-Trans, Y-Trans and Pow-Trans sections of the Measurements table are derived from the transition average curves. Transition average is available for the field component eye diagrams, and if the modulation format is an OOK type, for the power eye diagram.

It is also possible to show the waveform that would be expected in the absence of random noise (referred to as the Expected Eye or Expected Waveform) by enabling the system impulse response calculation in the Analysis Parameters tab. To display Expected Eye or Expected Waveform plots:

1. Click the **Analysis Parameters** tab.
2. Scroll to the **System impulse response** fields
3. Set Calculate expected eye and/or Calculate expected waveform to **True**.
4. Right-click the plot.
5. Select **Show Expected Eye** or **Show Expected Waveform**.

The expected eye or waveform is shown as yellow traces in any field eye diagram or Signal vs. Time plot.





The expected waveform or eye plot data is calculated using a two-step process:

- A deconvolution process calculates the signal impulse response.
- The impulse response is applied to the known data content of the signal to display the expected waveform or eye.

The calculation 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 expected waveform is a poor fit to the actual waveform.

The length of the impulse response is set by the Number of Symbols in impulse field in the System impulse response settings of the Analysis Parameters tab.

The expected waveform can provide useful information about the nature of the signal. Typically the computed eye diagram appears noisier as the impulse response length increases, 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 expected eye diagram calculation cleans up once the impulse response length parameter is set to a value long enough to capture that reflection event.

There are several options of which tributaries to take into account when calculating the impulse response. To select the tributaries contributing to the impulse response:

1. Click the **Analysis Parameters** tab.
2. Scroll to the **System impulse response** fields
3. Open the menu associated with **Tributaries contributing to impulse response** field.

The default option is Same trib only, which typically gives the cleanest result. You can include other tributaries and exclude the same tributary, for example Other SOP tribs.

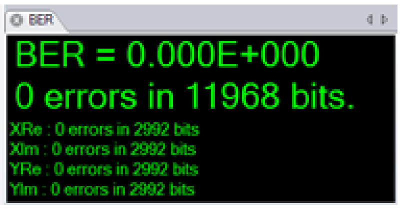
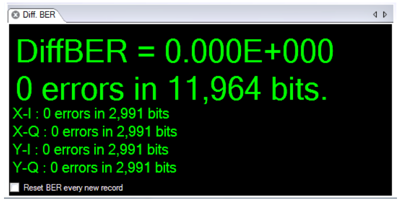
This setting computes the impulse response only by taking into account the signal on the other state of polarization. For an ideal signal the expected waveform data computed this way should be a flat line. If there is structure on the expected waveform then that suggests there is a crosstalk mechanism 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.

BER plots

Click the **BER** plot button (in the Home controls) to display the list of available plots.

Table 13: BER plots

Plot type	Description
Bit Error Ratio (BER)  BER = 0.000E+000 0 errors in 11968 bits. XRe : 0 errors in 2992 bits XIm : 0 errors in 2992 bits YRe : 0 errors in 2992 bits YIm : 0 errors in 2992 bits	Displays the BER by physical tributary and in total. The display color changes on sync loss.
Differential BER  DiffBER = 0.000E+000 0 errors in 11,964 bits. X-I : 0 errors in 2,991 bits X-Q : 0 errors in 2,991 bits Y-I : 0 errors in 2,991 bits Y-Q : 0 errors in 2,991 bits Reset BER every new record	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 precode your data signal before the modulator as in a typical differential transmitter, you must enter the patterns seen at the I and Q modulators into the respective pattern variables, (for example, PattXRe.Values and PattXIm.Values). If no precoding is used, then you may use the drop-down menus to specify standard PRBS codes. (See page 59, *Front end filtering*.)

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. (See page 59, *Front end filtering*.)

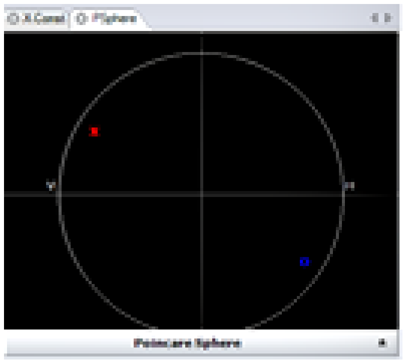
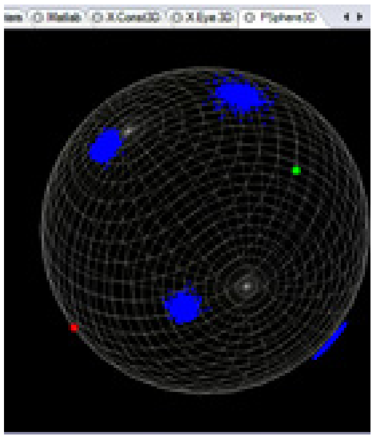
If you choose to precode your data signal before the modulator, as in a typical differential transmitter, you need to enter the patterns seen at the I and Q modulators into the respective pattern variables, (for example, PattXRe.Values and PattXIm.Values). If no precoding is used, then use the drop-down menus to specify standard PRBS codes.

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 (in Analysis Parameters) selects which BER type to report.

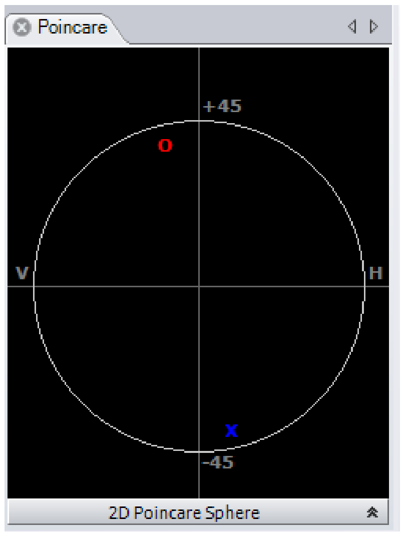
Poincaré plots

Click the **Poincaré** plot button (in the Home controls) to display the list of available plots.

Table 14: Poincaré plots

Plot type	Description
2D Poincaré Sphere 	Shows the position of the data signal polarizations relative to the receiver's H (X-I, X-Q) and V (Y-I, Y-Q).
3D Poincaré Sphere 	3D Poincaré shows the polarization of each symbol-center value. Click and drag to rotate the sphere.

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.

You can visualize the degree of orthogonality 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.”

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

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

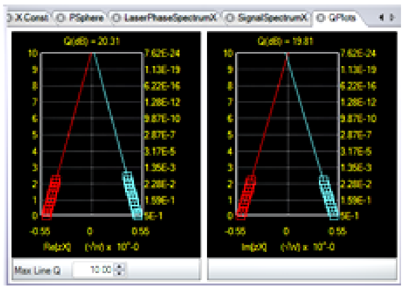
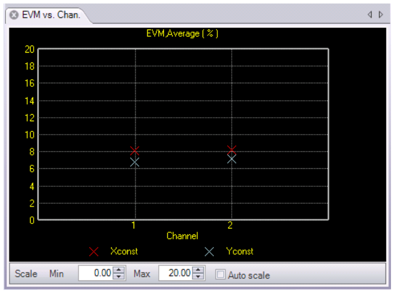
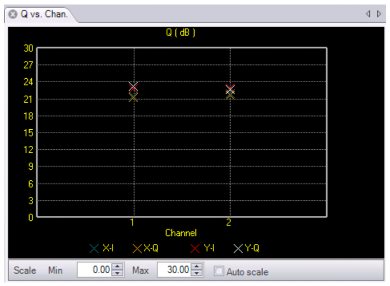
$U=2 \text{ Re}(E_x E_y^*)$

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

Q factor plots

Click the **Q factor** plot button (in the Home controls) to display the list of available plots.

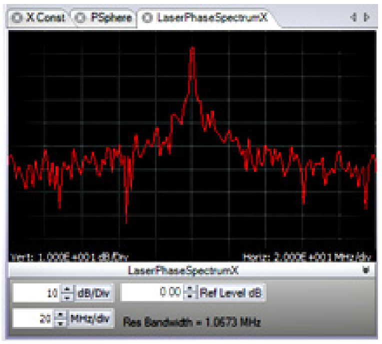
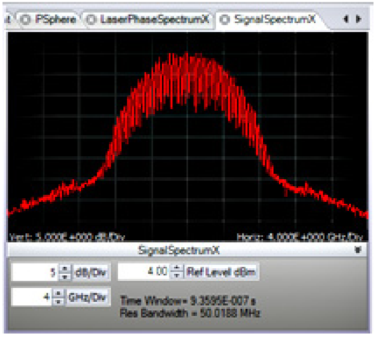
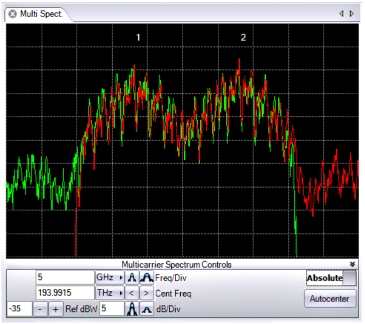
Table 15: Q factor plots

Plot type	Description
Decision Threshold Q-Factor 	Displays 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, and red for 0's.
EVM vs Channel (requires Option MCS) 	Displays the most recently measured EVM factor for each channel. This function requires Option MCS. Only the current channel is updated to display an aggregate plot of the multicarrier group, while the most recent data displayed for the other channels continues to be shown. Click the Clear Data button to discard prior data.
Q-Factor vs Channel (requires Option MCS) 	Displays the most recently measured Q factor for each channel. This function requires Option MCS. Only the current channel is updated to display an aggregate plot of the multicarrier group, while the most recent data displayed for the other channels continues to be shown. Click the Clear Data button to discard prior data.

Spectrum plots

Click the **Spectrum** plot button (in the Home controls) to display the list of available plots.

Table 16: Spectrum plots

Plot type	Description
Laser X Spectrum, Laser Y Spectrum  Vert: 1.000E+001 dB/Div Horiz: 2.000E+001 MHz/div LaserPhaseSpectrumX 10 dB/Div 0.00 Ref Level dB 20 MHz/div Res Bandwidth = 1.0673 MHz	The laser phase noise spectrum is obtained by taking an FFT of the $e^{i\theta}$, where θ is the recovered laser phase versus time. Increase the record length (Rec Len) to improve the frequency resolution. Right-click the plot to select what to display.
Current Signal Spectrum  Vert: 5.000E+000 dB/Div Horiz: 4.000E+000 GHz/Div SignalSpectrumX 5 dB/Div 4.00 Ref Level dBm 4 GHz/Div Time Window= 9.3595E-007 s Res Bandwidth = 50.0188 MHz	The frequency spectrum of the signal field is calculated using an FFT. The right-click menu options include spectrum calculations before and after front end processing, as well as after polarization separation to obtain the spectrum of each signal polarization. The plot can also show the spectrum of any Front end filter being applied. Use the available controls to set the vertical position of the front end filter plot
Multicarrier Spectrum (Requires Option MCS)  Multicarrier Spectrum Controls 5 GHz Freq/Div 193.9915 THz Cent Freq -35 Ref dBW 5 dB/Div	Displays a composite view of the channel group spectrum. This feature requires Option MCS. By default the Input Signal spectrum is displayed. Each spectrum is labeled with its channel number appearing at its center frequency. Right-click the plot to select display options similar to those for Current Signal Spectrum.

Multicarrier support (MCS) (Option MCS)

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. 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 OMA displays the results with the appropriate channel label by recognizing the channel based on the frequency entered by the user.

Multicarrier spectrum. The Multicarrier Spectrum plot is accessed by clicking on the spectrum icon button on the Home tab. Right-click 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.

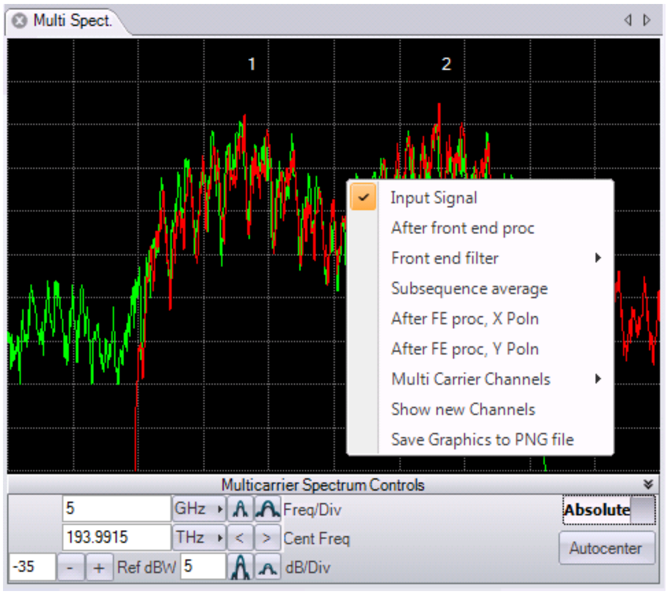


Figure 7: Multicarrier spectrum context menu

Table 17: 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 18: 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. Use drop-down menu to select units. Units 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. Use drop-down menu to select units. Units also change automatically when a change is made to the Absolute/Relative/Autocenter choices.
Ref dBW	Use the + and - 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 is 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 are shifted to zero Hz center frequency for visual comparison.

Understanding the multicarrier spectrum plot.

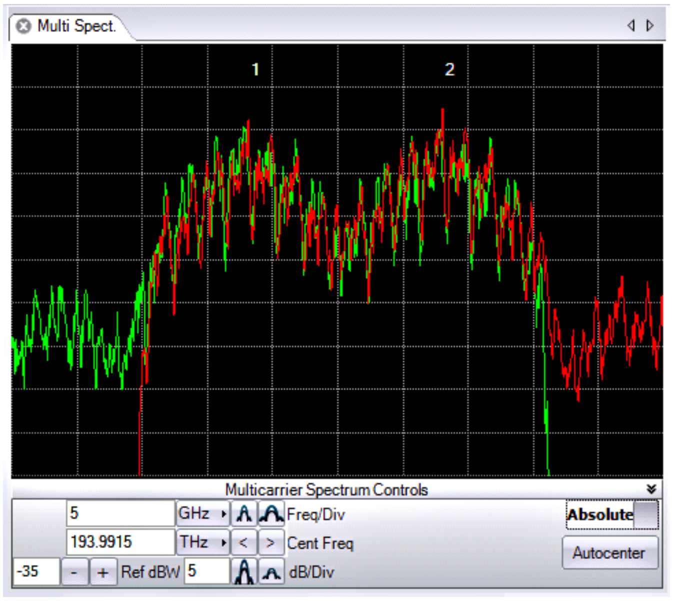


Figure 8: Multicarrier spectrum plot

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.

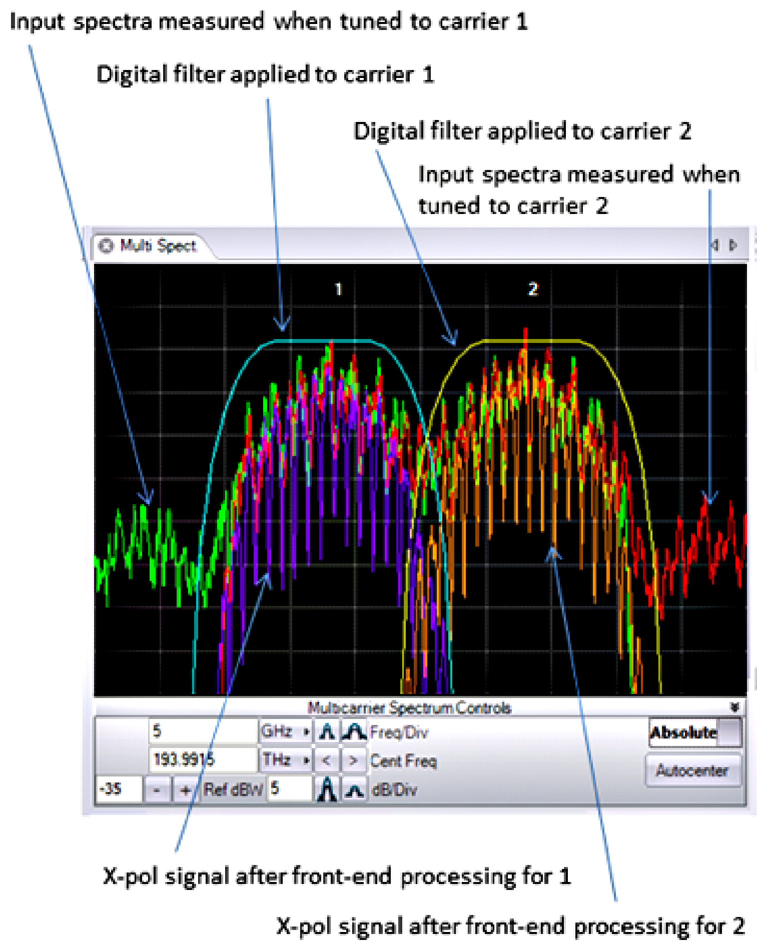

Input spectra measured when tuned to carrier 1
Digital filter applied to carrier 1
Digital filter applied to carrier 2
Input spectra measured when tuned to carrier 2
Multi Spect
1
2
Multicarrier Spectrum Controls
5
GHz
Freq/Div
193.9915
THz
Cent Freq
-35
Ref dBW
5
dB/Div
Absolute
Autocenter
X-pol signal after front-end processing for 1
X-pol signal after front-end processing for 2

Figure 9: Multicarrier spectrum plot details

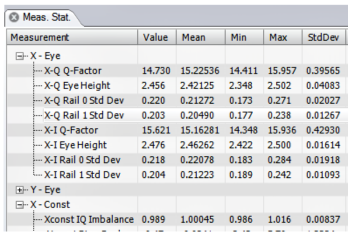
The preceding figure shows some of the other features of the Multicarrier Spectrum plot. Here the X-pol signal after front-end processing is shown together with the digital filter used. Notice that the X-pol signal (purple or orange) show much deeper nulls than the total power spectrum (red or green). 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.

Measurement plots

Click the **Measurements** button (in the Home controls) to display the list of available measurement results.

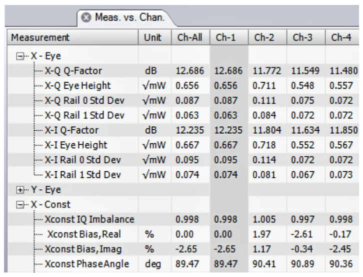
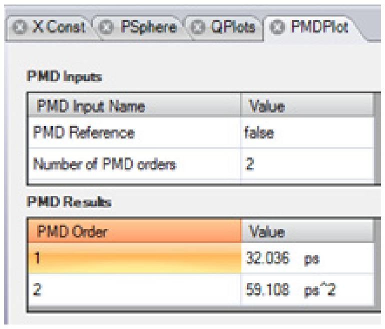
Table 19: Measurement plots

Plot type	Description
Measurement Statistics 	Displays every measurement made by the OMA 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. The following constellation measurements are shown only in the measurement plots: ■ IQ Quadrature Error: The IQ Quadrature Error, expressed in degrees, is the deviation of the measured phase angle between the I and Q components of the received signal from its target of 90 degrees. ■ IQ Offset: The IQ Offset is a measure of the magnitude of the shift of the center of the measured constellation from (0,0). It is a measure of carrier feedthrough in the optical modulator. It is evaluated as the magnitude of the average measured symbol, normalized by the square root of the average power in the measured constellation. The IQ Offset expresses this ratio in units of dB. Ideally, this ratio should be small, so the IQ Offset is typically a large negative number. ■ IQ Gain Imbalance: The IQ Gain Imbalance, expressed in dB, is a measure of the ratio between the sizes of the real parts of the measured constellation points and the sizes of the imaginary parts. Ideally, this ratio is unity, so the IQ Gain Imbalance should be near zero. When, on average, the sizes of the real parts of the measured constellation points exceed the sizes of the imaginary parts, the constellation appears “wider and flatter” than expected and the IQ Gain Imbalance > 0. Conversely, when the sizes of the imaginary parts exceed the sizes of the real parts, the constellation appears “taller and thinner” than expected and the IQ Gain Imbalance < 0. **X-Trans (Y-Trans)**: The transition parameter measurement is based on the Transition Average. (See page 108, *Signal vs time 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.

Meas. Stat.

Measurement	Value	Mean	Min	Max	StdDev
X - Eye					
X-Q Q-Factor	14.730	15.22536	14.411	15.957	0.39565
X-Q Eye Height	2.456	2.42125	2.348	2.502	0.04083
X-Q Rail 0 Std Dev	0.220	0.21272	0.173	0.271	0.02027
X-Q Rail 1 Std Dev	0.203	0.20490	0.177	0.238	0.01267
X-I Q-Factor	15.621	15.16281	14.348	15.936	0.42930
X-I Eye Height	2.476	2.46262	2.422	2.500	0.01614
X-I Rail 0 Std Dev	0.218	0.22078	0.183	0.284	0.01918
X-I Rail 1 Std Dev	0.204	0.21223	0.189	0.242	0.01093
Y - Eye					
X - Const					
Xconst IQ Imbalance	0.989	1.00045	0.986	1.016	0.00837

Table 19: Measurement plots (cont.)

Plot type	Description
	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 not checked. PDL is the relative size of the X and Y constellations (PDL of a PM modulator). **PMD**: Polarization mode dispersion. ***NOTE.*** *Make sure the desired measurement is enabled in the Analysis Parameters since some measurements are not calculated unless specifically enabled.*
Measurement vs. Channel (Requires Option MCS) 	Displays the most recent value to display the data from every channel in one plot. These are the same measurements described in the Measurement Statistics tab, but here the columns provide the most recent value for each channel rather than the statistics. Requires Option MCS. ***NOTE.*** *Make sure the desired measurement is enabled in the Analysis Parameters since some measurements are not calculated unless specifically enabled.*
PMD 	Displays the results of the polarization mode dispersion (PMD) calculation. 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. (See page 106, *PMD measurement.*)

Measurement	Unit	Ch-All	Ch-1	Ch-2	Ch-3	Ch-4
X - Eye						
X-Q Q-Factor	dB	12.686	12.686	11.772	11.549	11.480
X-Q Eye Height	√mW	0.656	0.656	0.711	0.548	0.557
X-Q Rail 0 Std Dev	√mW	0.087	0.087	0.111	0.075	0.072
X-Q Rail 1 Std Dev	√mW	0.063	0.063	0.084	0.072	0.072
X-I Q-Factor	dB	12.235	12.235	11.804	11.634	11.850
X-I Eye Height	√mW	0.667	0.667	0.718	0.552	0.567
X-I Rail 0 Std Dev	√mW	0.095	0.095	0.114	0.072	0.072
X-I Rail 1 Std Dev	√mW	0.074	0.074	0.081	0.067	0.073
Y - Eye						
X - Const						
Xconst IQ Imbalance		0.998	0.998	1.005	0.997	0.998
Xconst Bias,Real	%	0.00	0.00	1.97	-2.61	-0.17
Xconst Bias,Imag	%	-2.65	-2.65	1.17	-0.34	-2.45
Xconst PhaseAngle	deg	89.47	89.47	90.41	90.89	90.36

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^2

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 OMA can estimate the amount of PMD that a signal has experienced from the structure of the waveform. The method used by OMA is described in the research paper by Taylor[1]. 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 OMA 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.

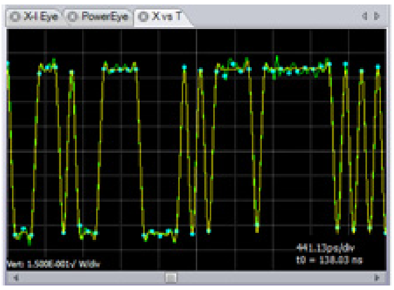
It is necessary to tell the OMA when the reference is being acquired, and that occurs by checking the PMD: Acquire PMD Reference check box. When the reference measurement is complete this check box must be unchecked. The reference-based measurement uses a linear impulse response, and the settings under Averaging (See page 108, *Signal vs time 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.

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

Signal vs. Time plot

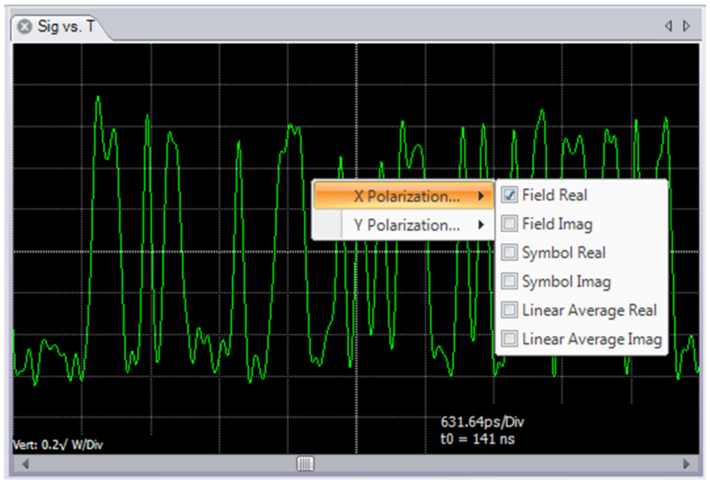
Click the **Signal vs. Time** plot button (in the Home controls) plot button (in the Home controls) to display the list of available plots.

Table 20: Signal vs. Time plot

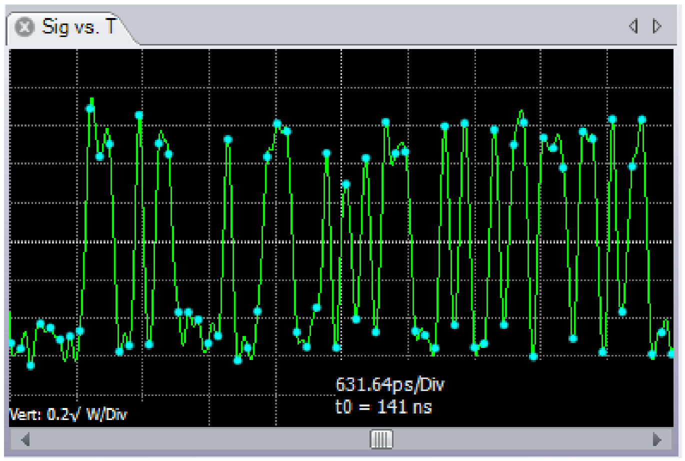
Plot type	Description
Signal vs. Time  X-I Eye PowerEye X vs T 441.13ps/div t0 = 138.03 ns Vert: 1.500E-001√ W/div	Displays field components as a function of time. t0 is the plot center relative to the trigger time. Errored symbols are shown in red. Signal vs Time is different from other plots in that 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 click the plot to open a context menu with X and Y polarization plot options. 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 Expected waveform vs. Time is discussed in the following section, and is plotted as yellow lines. Use the mouse wheel to zoom in or out. Use the scroll slider at the bottom of the plot to scroll left and right.

About 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 selections draw blue dots at the symbol center times. Expected waveform vs. Time is plotted as yellow lines. (See page 108, *Signal vs time waveform averaging*.)



Signal vs time waveform averaging

When the computation of the Expected waveform vs. time based on the system impulse response is enabled in Analysis Parameters, you can display a cleaner version of the signal with a reduced level of additive noise.

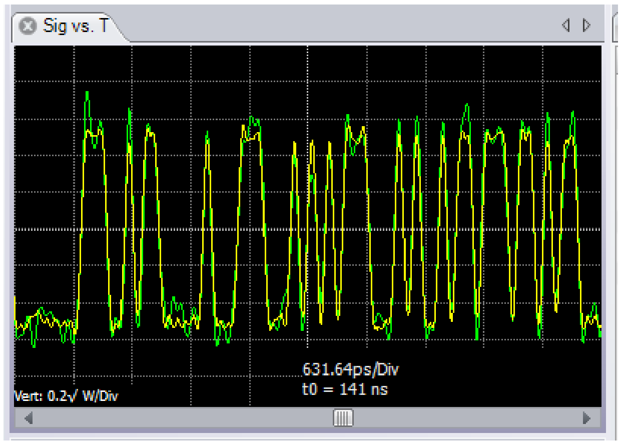
To enable displaying the waveform that would be expected in the absence of random noise (referred to as the Expected Eye or Expected Waveform)

1. Click the **Analysis Parameters** tab.
2. Scroll to the **System impulse response** fields
3. Set Calculate expected eye and/or Calculate expected waveform to **True**.

To show the expected waveform data on the plot:

1. Right-click the plot.
2. Select **X-Polarization | Y-Polarization > Show Expected Eye** or **Show Expected Waveform**.

The expected waveform is shown as yellow traces in the Signal vs. Time plot.



The expected waveform vs. time is calculated using a two-step process:

- A deconvolution process calculates the signal impulse response.
- The impulse response is applied to the measured signal data content to display the expected waveform or eye.

The calculation 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 expected waveform is a poor fit to the actual waveform.

The length of the impulse response is set by the Number of Symbols in impulse field in the System impulse response settings of the Analysis Parameters tab.

The expected waveform can provide useful information about the nature of the signal. Typically the computed eye diagram appears noisier as the impulse response length increases, 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 expected eye diagram calculation cleans up once the impulse response length parameter is set to a value long enough to capture that reflection event.

There are several options of which tributaries to take into account when calculating the impulse response. To select the tributaries contributing to the impulse response:

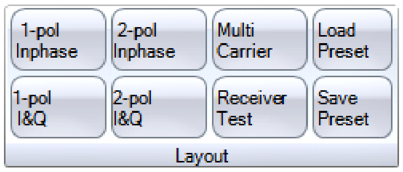
1. Click the **Analysis Parameters** tab.
2. Scroll to the **System impulse response** fields
3. Open the menu associated with **Tributaries contributing to impulse response** field.

The default option is Same trib only, which typically gives the cleanest result. You can 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 expected waveform data computed this way should be a flat line. If there is structure on the expected 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.

Layout controls



Layout controls provide a fast way to set up the software to display multiple plot and measurement tabs for a measurement:

Click a defined layout button (1-pol Inphase, 2-pol I&Q, Receiver Test, and so on) to load and configure the plot screens for the selected measurement type.

Click **Load Preset** to open a Windows Navigation dialog box. Select a layout file (.xml) to load. Loading a layout deletes the current plot tabs.

Click **Save Preset** to open a Windows Navigation dialog box. Enter a name for the layout file and click Save to save the current screen plot layout to the file. You may not overwrite the default layout files that correspond to the other layout buttons.

The Calibrate tab

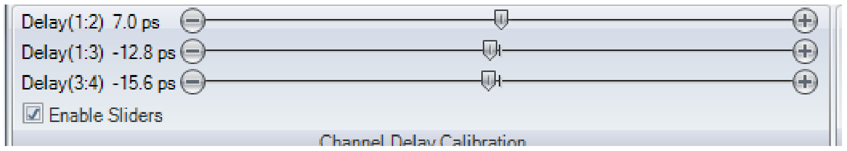
The OM4000 Series receiver requires the following calibrations to be completed before taking measurements:

- Hybrid receiver calibration (correction for cross-talk and phase error in the hybrid). This is handled by the Get Calibration Files function.
- Laser linewidth factor (choosing the correct filter for phase recovery) (See page 54.)
- Receiver equalization. This is handled by the Get Calibration Files function which places equalizers for use by OMA.
- Signal delay fine adjustment (channel to channel skew in the oscilloscope connections)
- DC calibration (to compensate for any offsets in the photodiode outputs)

Signal delay adjustment is performed by the Scope & Receiver De-Skew function (Setup > Scope Setup controls). (See page 36, *Scope & Receiver Deskew button*.) The signal fine delay and DC calibration functions are accessed from the Calibrate tab and described in the following sections.

Enable Sliders (signal delay fine adjusts) (RT)

Once the Scope and Receiver Deskew utility has set the channel delay values, you can use the calibration sliders to make fine adjustments to the delays. Click the **Enable Slider** control in the Calibrate tab, then click the + and - buttons for each setting (or drag the setting marker with the mouse).


Delay(1:2) 7.0 ps
Delay(1:3) -12.8 ps
Delay(3:4) -15.6 ps
☑ Enable Sliders
Channel Delay Calibration

- Delay(1:2) refers to the XI to XQ skew
- Delay(1:3) refers to the XI to YI skew
- Delay (3:4) refers to the YI to YQ skew.

Disable the sliders and save the OMA state when complete to avoid inadvertent changes.

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

DC Calibration (real-time oscilloscopes)

NOTE. *Clicking the Auto Scale / DC Calib button (Setup tab) automatically runs the DC Calibration. The DC Calibration button verifies DC calibration without changing the oscilloscope scale settings.*

Although the OM4000 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 receiver photodiode DC levels and subtracts this from the analysis. Uncompensated DC offset in the system is indicated by a smearing out of the constellation point groups. If the offset is large enough, the point groups will begin to look like donuts.

To perform a DC calibration, click the DC Calibration button in the Calibrate tab. The DC calibration can be done as often as needed to compensate for potential DC offset.

Get Calibration Files

The factory instrument calibrations for Hybrid receiver calibration (correction for cross-talk and phase error in the hybrid) and Receiver equalization are stored on the instrument USB flash memory. You must copy these calibration files to the PC so that OMA can access their values.

Click **Get Calibration Files** to copy the calibration files from the flash drive to the proper location on the PC. The OMA software detects and uses the calibration files to compensate for the measured optical and electrical properties of the connected OM4000 hardware.

For verification and correction of the factory calibration, see the sections on Hybrid Calibration. (See page 121, *Hybrid calibration (RT)*.)

The Offline tab

The Offline tab controls let you record a session, load saved acquisition, and run the loaded files for analysis.

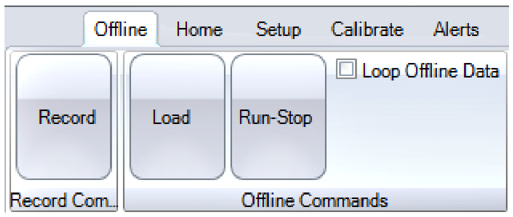


Table 21: Offline controls

Control	Description
Record	Saves the oscilloscope data and all signal description parameters as a sequence of MATLAB (.mat) files at location C:\Users\<user>\Documents\TekApplications\OUI\MAT Files. You can then load the saved files for later analysis. The program saves one .mat file for each oscilloscope block processed. To record files, click Offline > Record to set the program to save each analysis to a separate .mat file, then click Run-Stop on the Global control panel to start a live acquisition run and save analysis results.
Load	Loads saved .mat files. Click Load to open a standard Windows file dialog and navigate to the location of the .mat files. The default location is **My Documents/TekApplications/OUI/MAT Files** Use standard Windows operations to select single, multiple, or all files in a location. Click OK to load the files.
Run-Stop	Runs loaded .mat files. OMA uses the filter settings stored in the .mat files. Use the Offline Processing controls (at the bottom of the Analysis Parameters window) to override some .mat file settings. This makes it possible to reprocess the saved data using different filters or other settings, rather than those settings stored in the .mat file. (See page 58.)

The Alerts tab

The Alerts tab of the menu ribbon displays a list of all the alert messages that occur during OMA processing. The Alerts section has the following fields:

- Zone: A string saying from where the function was called
- Code: The code number associated with that alert message
- Function Name: The name of the function where the alert was activated
- Message: The alert message
- Times Asserted: The number of times the alert was activated

The purpose of the Zone field is to identify not just which function triggered the alert, but where in the code it was triggered.

See the *Alert codes* appendix for alert code information. (See page 159, *Alert codes*.)

The About tab

Opens a dialog box that displays the application software version number.

The global Controls panel

The global **Controls** panel is where you set record length and block size, run and stop data acquisitions, set display scale, trace intensity and color, and other controls as available for particular plots. The controls affect almost all plots. This control panel is pinned to the left side of the application by default for easy access.

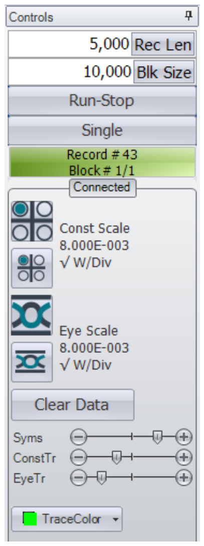


Table 22: Controls panel elements

Control	Description
Rec Len	Record Length. Sets 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 OMA 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 22: 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. Also, 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 is 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 227, *Managing data sets with record length > 1,000,000*.)
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 the icons to increment or decrement the setting.
Other controls	Other controls will display depending on the selected plots or measurements.

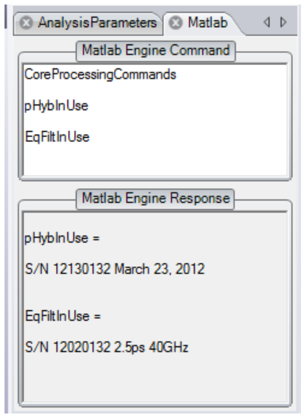
Table 23: Record length and block interaction behavior

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.

Appendix A: Hybrid calibration (RT)

This 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 OMA 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).



```
CoreProcessingCommands
pHybInUse
EqFiltInUse
```

```
pHybInUse =
S/N 12130132 March 23, 2012

EqFiltInUse =
S/N 12020132 2.5ps 40GHz
```

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 verifies and corrects the calibration at a single wavelength using a minimum of external hardware. This procedure assumes that you can tune “Laser 1” and Laser 2” 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 before doing this procedure. (See page 36, *Scope & Receiver Deskew button*.)
2. Connect the Reference from a OM4000 laser output (1 or 2) to the **Reference input** connector with short PM/APC jumper.

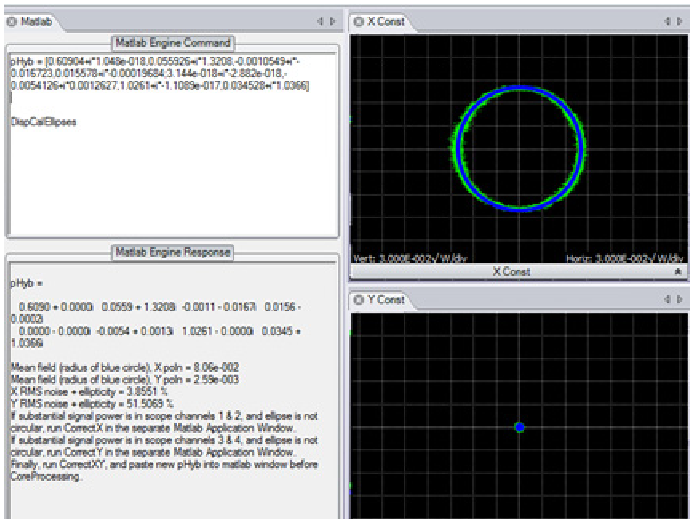
NOTE. *You can use either laser output on the OM4000 instrument to connect to the Reference input or Signal Input connectors.*

3. Use a standard SM/APC (not PM) fiber to connect from the remaining OM4000 laser output (1 or 2) to the **Signal input** connector.
4. Use the LRCP tab to set the Signal Laser power level to about 10 dBm to start.
5. Use the LRCP tab to turn on both lasers and tune them to the same channel where you will be working. Set the Signal Laser power to get about 100 mV pp.
6. Use the LRCP tab to fine tune Laser 1 off grid by +1000 MHz.
7. Click **Run** on the oscilloscope; the data should look like 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 (RT).

1. Use the OMA to connect to the oscilloscope. A record length of ~10,000 points is 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. EnterDispCalEllipses in the OMA MATLAB Engine Window. Delete everything else in the MATLAB Engine Window.
5. Click **Run** on the OMA.
6. Observe that the Constellation plots show ellipses. Only the X-constellation has a 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 (desktop) MATLAB Application Window.
   - Copy and paste the resulting pHyb statement before DispCalEllipses in the MATLAB Engine Window in the OMA as shown below.
8. Click **Run**. The green trace should now be circular. If it is not, repeat the procedure one time.



```
pHyb = [0.60904+i*1.048e-018,0.055926+i*1.3208,-0.0010549+i*-
0.016723,0.015578+i*-0.00019684;3.144e-018+i*-2.882e-018,-
0.0054126+i*0.0012627,1.0261+i*-1.1089e-017,0.034528+i*1.0366]

DispCalEllipses
```

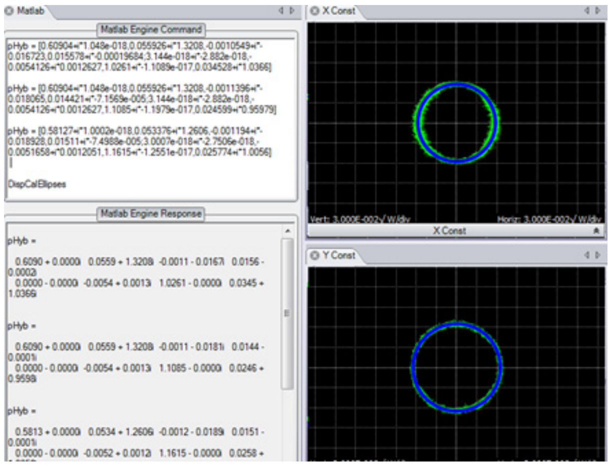
```
pHyb =

  0.6090 + 0.0000i  0.0559 + 1.3208i  -0.0011 - 0.0167i  0.0156 -
0.0002i
  0.0000 - 0.0000i  -0.0054 + 0.0013i  1.0261 - 0.0000i  0.0345 +
1.0366i

Mean field (radius of blue circle), X poln = 8.06e-002
Mean field (radius of blue circle), Y poln = 2.59e-003
X RMS noise + ellipticity = 3.8551 %
Y RMS noise + ellipticity = 51.5069 %
If substantial signal power is in scope channels 1 & 2, and ellipse is not
circular, run CorrectX in the separate Matlab Application Window.
If substantial signal power is in scope channels 3 & 4, and ellipse is not
circular, run CorrectY in the separate Matlab Application Window.
Finally, run CorrectXY, and paste new pHyb into matlab window before
CoreProcessing.
```

Procedure to inspect 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 OMA.
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 (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.



```
pHyb = [0.60904+i*1.048e-018,0.055926+i*1.3208,-0.0010549+i*-
0.016723,0.015578+i*-0.00019684;3.144e-018+i*-2.882e-018,-
0.0054126+i*0.0012627,1.0261+i*-1.1089e-017,0.034528+i*1.0366]

pHyb = [0.60904+i*1.048e-018,0.055926+i*1.3208,-0.0011396+i*-
0.018065,0.014421+i*-7.1569e-005;3.144e-018+i*-2.882e-018,-
0.0054126+i*0.0012627,1.1085+i*-1.1979e-017,0.024599+i*0.95979]

pHyb = [0.58127+i*1.0002e-018,0.053376+i*1.2606,-0.001194+i*-
0.018928,0.01511+i*-7.4988e-005;3.0007e-018+i*-2.7506e-018,-
0.0051658+i*0.0012051,1.1615+i*-1.2551e-017,0.025774+i*1.0056]

DispCalEllipses
```

```
pHyb =

  0.6090 + 0.0000i  0.0559 + 1.3208i  -0.0011 - 0.0167i  0.0156 -
0.0002i
  0.0000 - 0.0000i  -0.0054 + 0.0013i  1.0261 - 0.0000i  0.0345 +
1.0366i


pHyb =

  0.6090 + 0.0000i  0.0559 + 1.3208i  -0.0011 - 0.0181i  0.0144 -
0.0001i
  0.0000 - 0.0000i  -0.0054 + 0.0013i  1.1085 - 0.0000i  0.0246 +
0.9598i


pHyb =

  0.5813 + 0.0000i  0.0534 + 1.2606i  -0.0012 - 0.0189i  0.0151 -
0.0001i
  0.0000 - 0.0000i  -0.0052 + 0.0012i  1.1615 - 0.0000i  0.0258 +
```

Finish the hybrid calibration (RT).

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.

Appendix B: OM5110 information

Theory of operation

The OM5110 contains a dual-polarization IQ optical modulator capable of producing optical modulation at up to 46 Gbaud for binary modulation and 34 Gbaud for small-signal modulation. The optical modulator translates the RF input signals to the frequency of the Optical Input using dual nested Mach-Zehnder modulators to provide RF IQ modulation on two orthogonal polarizations at the Optical Output.

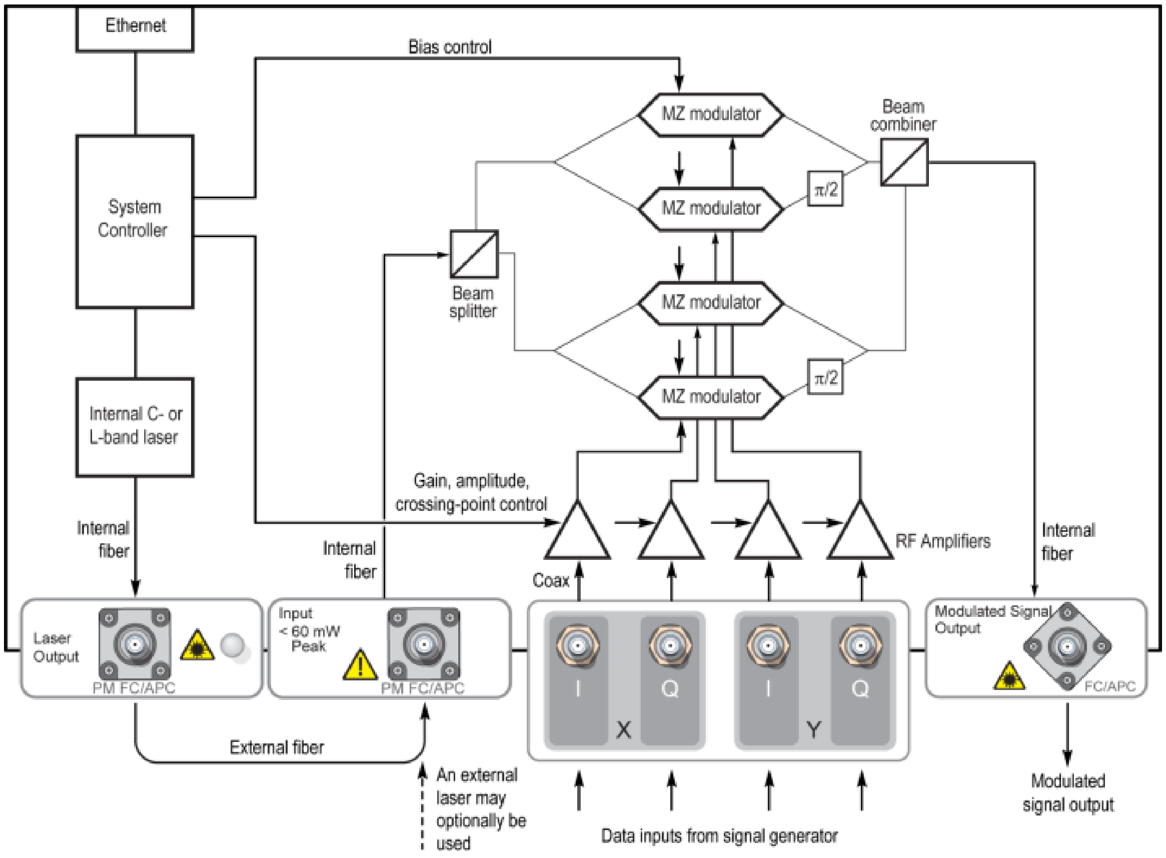


Figure 10: OM5110 block diagram

The linear two-stage amplifiers increase the RF input signals by 20 dB before going to the modulator. The amplifier gain reduces the input-referred Vpi voltages of the modulators to approximately 350 mV. Full-amplitude binary-phase-shift-keyed modulation (BPSK) is achieved with signals greater than 700 mVpp but no more than 1 Vpp is required for complete saturation.

For 2-polarization IQ modulation, 4 RF input signals are required. If the OM5110 is used for other modulation types, only the necessary number of inputs need be connected, for example, a single RF data signal is required for 1-pol BPSK. The LRCP software is used to configure the optical modulator bias controller for the modulation type. The optical bias controller makes sure that each of the data modulators and IQ-phase control modulators are biased at the proper points for IQ modulation. Each IQ RF-input pair should have minimum correlation to obtain the best result with automatic optical bias control. For example, putting exactly the same signal on I and Q will not necessarily result in the expected IQ modulation unless manual optical bias control is used.

There are two primary modes of operation for the OM5110: large-signal and small-signal. Large-signal modulation occurs from around 500 mVpp to where the amplifier is completely saturated at 1 Vpp. In this range both the amplifier and modulator have a nonlinear characteristic that provide higher quality modulation for binary signals. As the amplifier becomes saturated, use the LRCP software to adjust the amplitude and duty-cycle distortion (crossing-point) provided by the amplifier's second stage of gain. Factory settings provide 2Vpi modulation amplitude and 50% crossing point, but you can adjust these to accommodate different signal sources.

The small signal mode of operation is most often used for multi-level inputs used to create QAM signals. The amplifier remains linear up to 300 mVpp so the amplitude and duty cycle adjustments will have little effect. In this case, you adjust the first amplifier gain stage to equalize the small signal gains of the different channels. The factory settings typically provide 2% amplitude matching between input channels. Use this mode of operation to convey baseband multi-level IQ signals such as 16-QAM to optical frequencies.

The OM5110 is ideally suited for use with the AWG70001A and RFxpress application software for creating arbitrary signals that are precompensated for the response of the AWG and the OM5110 to provide the desired modulated optical waveform. See the Application Note provided with the OM5110 AWG file library for further information on using the OM5110 with the AWG70001A and RFxpress.

Appendix C: OM2210 information

Product description

The OM2210 Coherent Receiver Calibration Source and associated software are used in laboratory or industrial facilities to measure the properties of polarization-diverse, phase-diverse coherent fiber optic receivers that are linear or can be set in a linear mode.

The OM2210 may be used for measuring coherent receiver heterodyne frequency response, measure optical properties vs wavelength, or both. When measuring optical properties vs. wavelength, the end result is a transfer matrix, measured at a particular heterodyne frequency over a specified wavelength range, that describes the optical-to-electrical transfer function of the analog part of the receiver. The transfer matrix is then used to compute critical receiver specifications such as quadrature phase angle, polarization cross-talk, and path gains. This information can be used for quantitative evaluation of the receiver, or to provide calibrated optical field measurements when used with the OM4000 signal analysis software.

Optical communications signals are the combination of a continuous laser field (cw) and a modulation field that carries the data. In the case of polarization multiplexed signals, there may be two independent lasers and modulators that are polarization-multiplexed together.

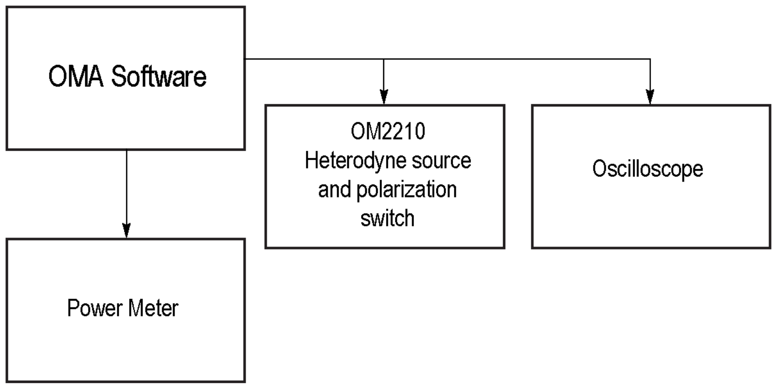
The OM4000 signal processing software is designed to extract the cw and modulation for each polarization from which data and signal-quality metrics may be extracted. Since the software extracts fields, not just data, it is necessary to have independent measurements of the receiver properties so that the effect of the receiver may be removed. This process consists of a low-frequency measurement of the linear OE transfer matrix relating the input field vector, E, to the output voltages, V, and a separate frequency response measurement. The OM2210 can measure the linear OE transfer matrix, H, at frequencies well within the oscilloscope bandwidth.

$V=[H]E$

This process can be applied both to linear receivers like the OM4000 Series and receivers with automatic gain control (AGC) such as commercial coherent receivers. The AGC must be turned off during the measurement so that the software can measure the linear part of the transfer function. The matrix can then be normalized to simulate the AGC operation.

The calibration software extracts the hybrid matrix, H, by applying a heterodyne signal with different polarizations and then measuring the resulting sine wave outputs on the oscilloscope while separately monitoring the power. Proprietary calculations are performed to extract the hybrid matrix in the presence of receiver impairments, heterodyne phase fluctuations, and channel skews.

This process is directed by the RxTest function of the Optical Modulation Analyzer software. The Rx Test function controls the heterodyne source and collects data from the oscilloscope.



The final wavelength scan calibration results are provided in .mat, .csv, and .xls file formats. The .mat format is used by the OMA software hybrid calibration function. The other formats are provided for viewing the results. See the RxTest function for further details. (See page 70, *The Receiver Test configuration tab*.)

Appendix D: Connection diagrams

OM4000-series equipment setup

Real-time (RT) oscilloscopes

The following figures are examples of how to connect the OM4000 series instrument to take measurements with real-time oscilloscopes.

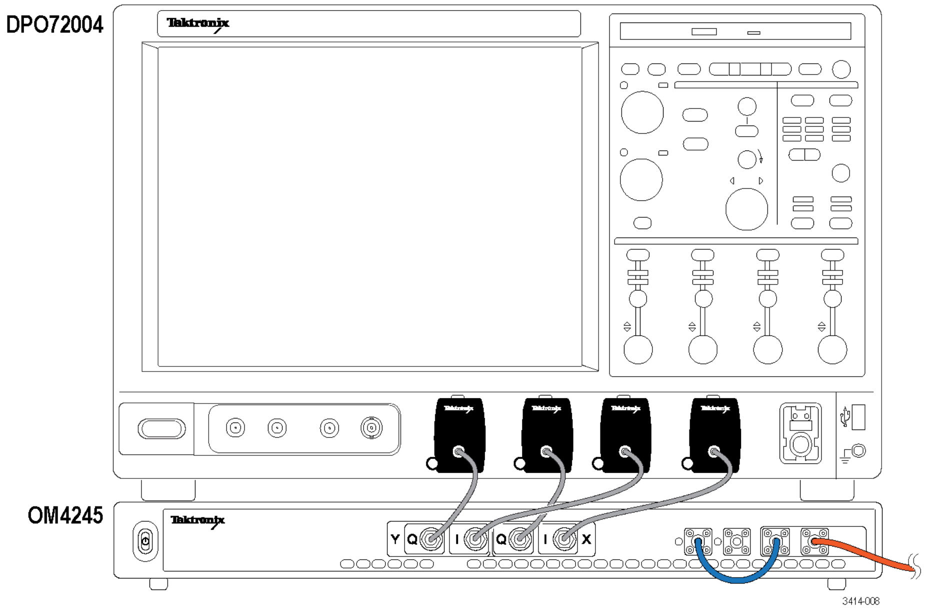


Figure 11: Real-time (RT) oscilloscope setup diagram: Tektronix DSO/MSO70000C/D/DX series and OM4245 Optical Modulation Analyzer

---

NOTE. *The above setup example provides the flexibility to test all X and Y polarization simultaneously at 50 GS/s, or test either the X or Y polarization at 100 GS/s, using the oscilloscope feature to enable 50 GS/s on four channels or 100 Gs/s on two channels (either channel 1+3 or 2+4), without having to change cables.*

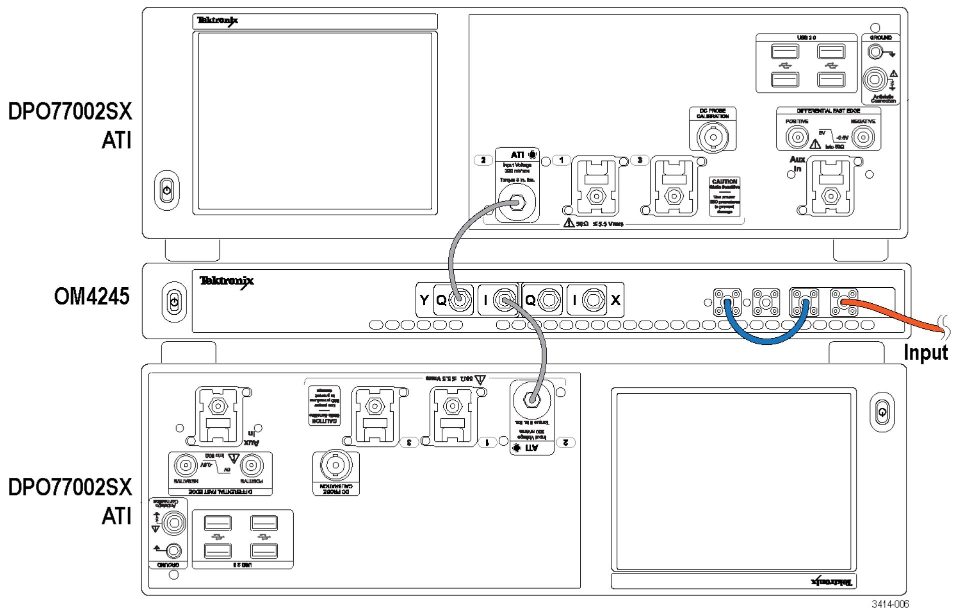


Figure 12: Real-time (RT) oscilloscope setup diagram: <80 GBaud single polarization testing using DPO77002SX ATI oscilloscopes and OM4245 Optical Modulation Analyzer

NOTE. *A pair of two DPO77002SX oscilloscopes is called a DPS77004SX system, which includes the necessary UltraSync cable for automatic synchronization of the oscilloscopes.*

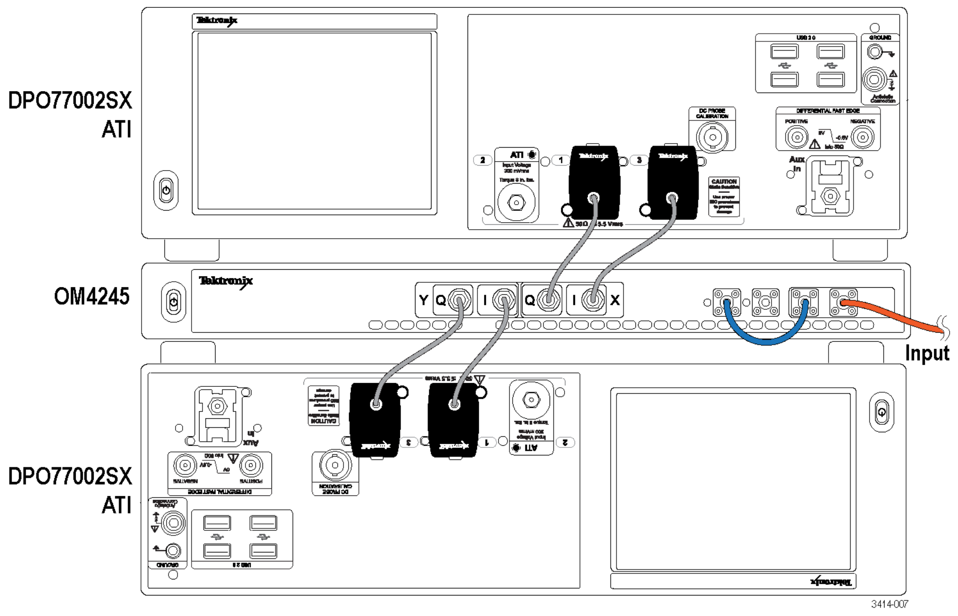


Figure 13: Real-time (RT) oscilloscope setup diagram: <60 GBaud dual polarization testing using DPO77002SX ATI oscilloscopes and OM4245 Optical Modulation Analyzer

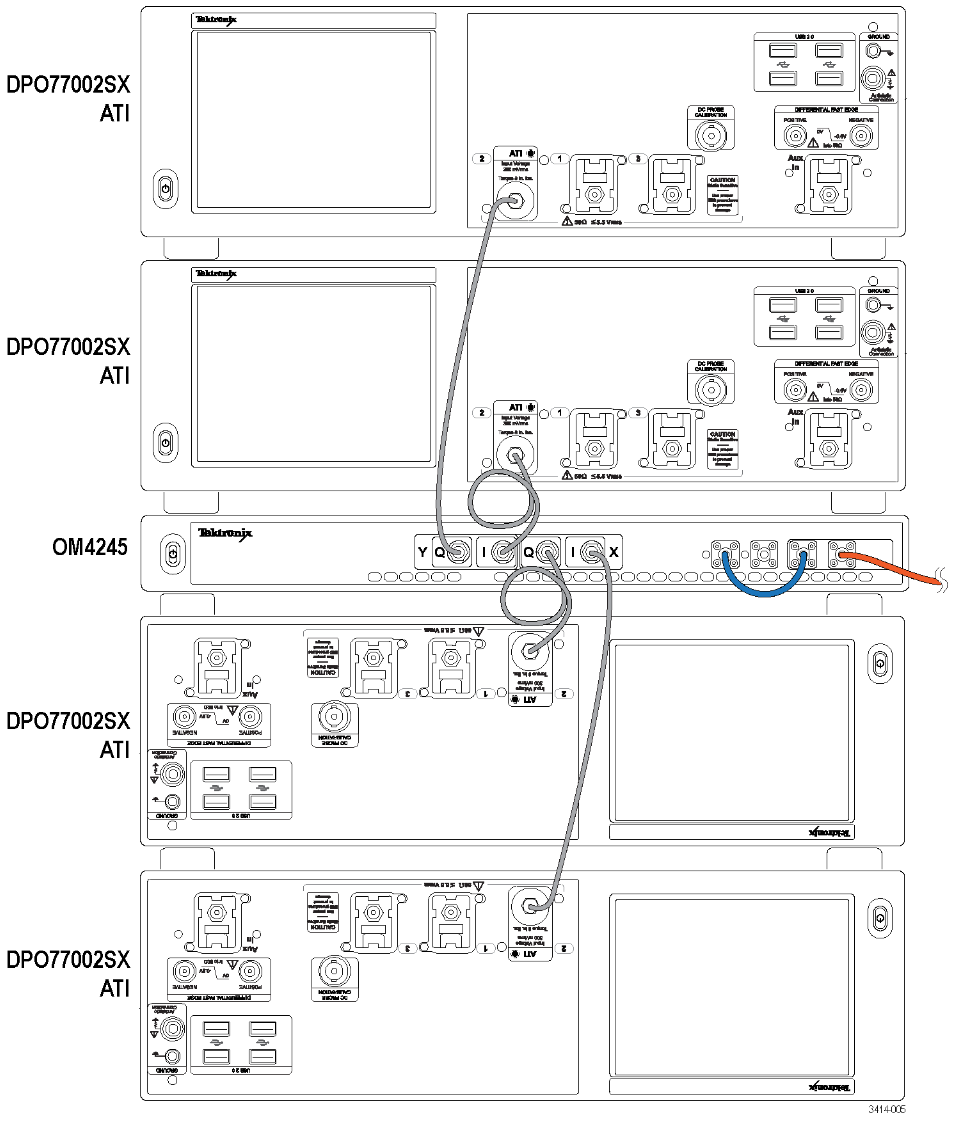


Figure 14: Real-time (RT) oscilloscope setup diagram: 400G (<80 GBaud) dual-polarization signal testing using DPO77002SX ATI oscilloscopes and OM4245 Optical Modulation Analyzer

NOTE. *The instruments on the bottom of the stack are upside-down to minimize cable lengths. The instruments can also be stacked normally if cable length is not an issue.*

Oscilloscope/OM4000 series connection order (generic)

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 orange. Push the power button one time to enable the unit. The steady green power button light indicates the instrument is ready for use and that lasers can be activated using the appropriate controller software.

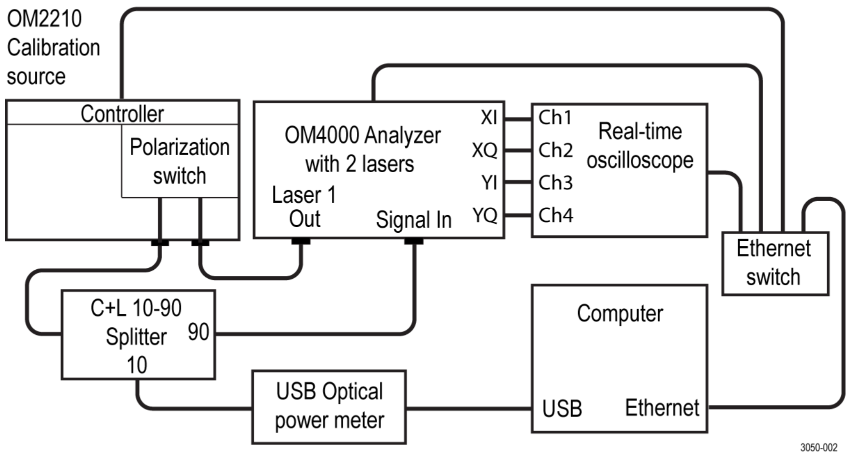
The power light turns orange and the unit is disabled any time AC power is removed or the OM4000 is rebooted by pressing and holding the front-panel power button for 10 seconds. Push the power button to re-enable. This feature prevents a remote user from activating the lasers when the local user may 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, be sure the controller PC (such as a laptop) has only one Ethernet connection (either wireless or wired) activated.*

OM2210 equipment setup

The following figure shows how to set up the OM2210 (Option NL) to calibrate the OM4000 Series Optical Modulation Analyzer.



Connect the Polarization Switch Output to the input fiber of the optical splitter (provided).

Connect the 90% output fiber of the splitter to the OM4000 series Signal Input.

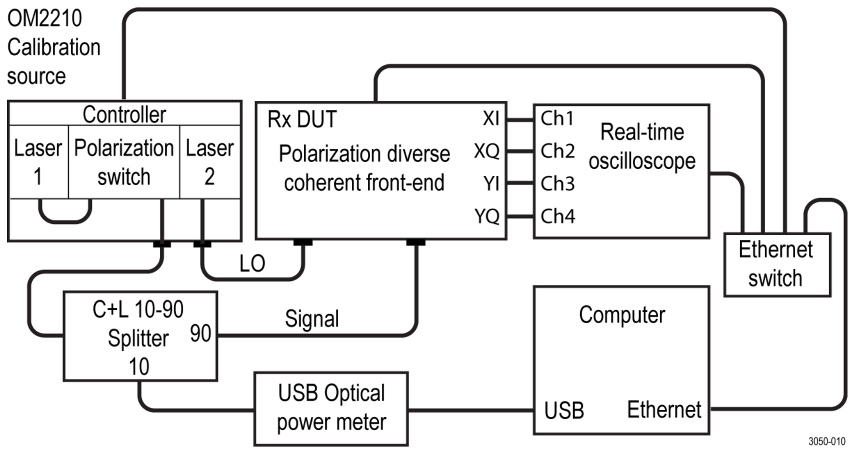
Connect the 10% output fiber of the splitter to the optical power meter (provided).

Connect the power meter to the computer used to run the calibration.

Connect the OM4000 X and Y I/Q outputs to the oscilloscope using high-quality SMA cables of equal electrical length.

Connect the OM2210, OM4000 Series Analyzer, oscilloscope, and computer to an Ethernet router switch or over a network.

The following is the setup configuration for receiver testing (OM2210 option CC or LL):



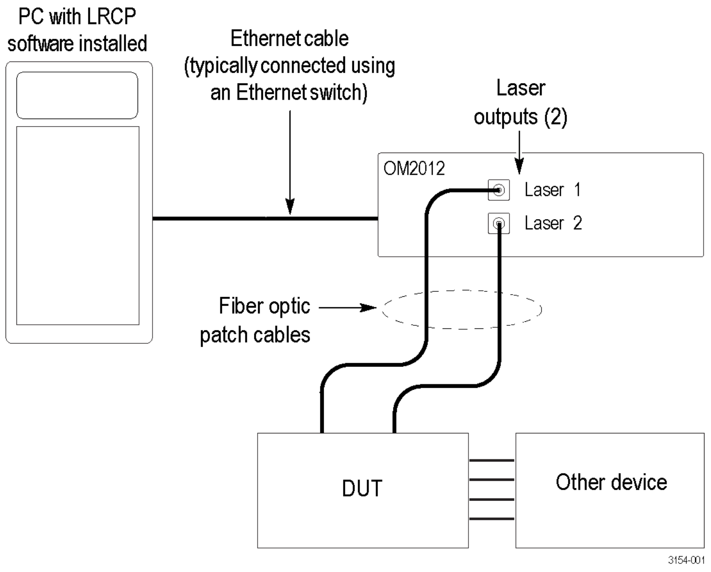
Once the equipment is connected, turn on the computer, the oscilloscope, the DUT, and the main power switch on the back of the OM2210. The OM2210 front-panel power button will light briefly after main power is applied indicating it is searching for a DHCP server. When an IP address is assigned or when the search fails in the case of an isolated network, the power light will go off.

Push the power button one time to enable the unit. The steady power button light indicates the OM2210 is ready for use and that lasers may be activated at any time a user connects from the Ethernet. The light will go out and the unit is disabled any time main power is removed or the IP address is changed. Push 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 four devices on an Ethernet network: computer, oscilloscope, DUT, and OM2210. You can keep this setup isolated, or connect it to your corporate network or router.

OM2012 equipment setup


PC with LRCP software installed
Ethernet cable (typically connected using an Ethernet switch)
Laser outputs (2)
OM2012
Laser 1
Laser 2
Fiber optic patch cables
DUT
Other device
3154-001

Appendix E: Receiver Test

Receiver Test overview

NOTE. *Receiver Test requires a Tektronix OM2210.*

The Receiver Test software is bundled with the OMA software provided with the OM1106, OM2210, and OM4000 products. It is installed separately and maintains signal processing files, application data files, and user files that are separate from the OMA software, but it has many similarities with the OMA software on which it is based. For example, the Receiver Test and the OMA software use the same hardware setup utility. Receiver Test has the same dependence on Matlab and supports the same versions of Matlab as the OMA software.

The Receiver Test software automates heterodyne coherent receiver frequency response and harmonic-distortion test. It uses the OM2210 Coherent Receiver Calibration Source to provide the Signal and LO excitation, as well as a polarization switch to provide two orthogonal states of polarization to the Device Under Test (DUT).

Optionally, the software can be provided with the Tektronix, Inc. SX70000 coherent receiver test system. This system uses the Keithley S46T Microwave Switch System that can be equipped with Radiall 50 GHz SPDT terminated microwave switches or other compatible switches. These switches select which of the DUT outputs are connected to the four DPO70000SX oscilloscope channels enabling measurements on receivers with differential outputs. The entire system is controlled by the Receiver Test software running on the rack system computer.

Differential coherent receivers may also be tested using the Tektronix TriMode probe in place of the coaxial switch for applications up to 33 GHz. In this case the software is used to make a single-ended measurement, but the TriMode probe may be set to connect either the positive (P), negative (N), or provide P-N to the oscilloscope.

Heterodyne coherent receiver test system

The receiver test software controls the OM2210 lasers to provide heterodyne signals at two orthogonal polarizations to the DUT. The DUT outputs are measured in turn by the oscilloscope. Using the DPS77004SX enables measurements up to 70 GHz. Other configurations are supported for lower bandwidth or single-ended receivers.

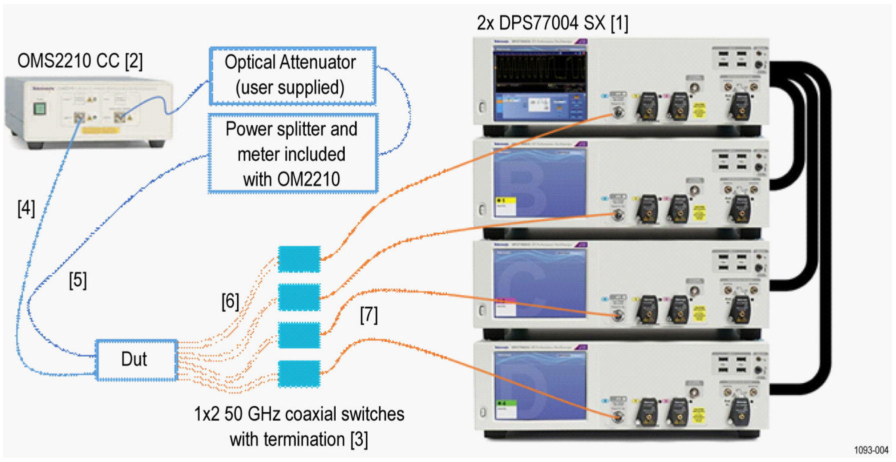

2x DPS77004 SX [1]
OMS2210 CC [2]
Optical Attenuator (user supplied)
Power splitter and meter included with OM2210
[4]
[5]
[6]
[7]
Dut
1x2 50 GHz coaxial switches with termination [3]
1093-004

Figure 15: Heterodyne coherent receiver test system diagram

The oscilloscope [1] may be a number of different options ranging from the 23-GHz DP072304SX to the 70 GHz DPS77004SX. The OM2210 CC is connected via polarization maintaining fiber [4] to the DUT local-oscillator input and single mode fiber [5] to the DUT signal input via the attenuator and splitter. DUTs with differential outputs require either a coaxial switch [3] or a TriMode probe

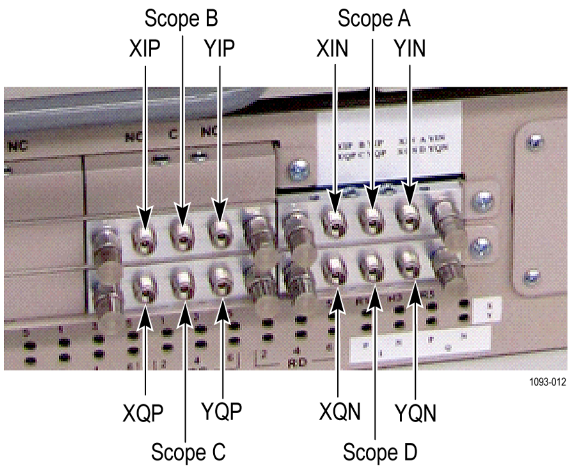


Figure 16: Relay connections on the S46T when used with the SX70000

Receiver test setup

1. If connecting a DHCP router to the system, be sure the router is on and ready before powering on the OM2210. The OM2210 only picks up an IP address when powering up.
2. Launch the Scope Service Utility on the Master Scope (Scope A). Wait for it to finish launching. It should display "running" and have the Trigger Group Master box checked.
3. Oscilloscope settings:
   a. Recall the oscilloscope state if there have been any changes since last use.
   b. It is important to Trigger on Line. It is not possible to trigger on any DUT waveforms because all of the waveforms will go to zero at the highest frequencies.
4. Launch the Receiver Test software. Note that when the Receiver Test software is launched, it will put the oscilloscope in Run mode and turn off the waveforms. The waveforms are off to speed processing.

NOTE. *During the Receiver Test, do not run any other software on the Master Scope that could slow down data processing.*

5. Use the LRCP control panel to set the laser powers as desired but less than or equal to 15.5 dBm. The insertion loss through the polarization switch should be less than 10 dB. Additional attenuation may be needed to achieve the desired Signal input power.
6. The oscilloscope vertical scale and offset is set automatically by the "autoscale" feature during the test unless this feature is disabled in the property grid. The autoscale feature can also be tuned using the scale factor in the property grid. The default value is .23; larger values will result in a larger number of volts per division for the same signal amplitude.
7. Signal adjustments:
   a. After setting the LO and Signal power levels appropriately, and setting the DUT in the desired state, tune the lasers to the same wavelength so that there is a low frequency sine-wave on the four channels.
   b. Because the Signal excitation is single polarization, move the Signal fiber or use a polarization adjuster to approximately maximize the displayed waveforms for both polarizations. Ideally, there should be approximately equal signal on both polarizations, but good results are possible with ratios of up to 3:1 between the displayed signal levels.

c. If not using the autoscale feature, center the four waveforms so they all have the ability to fill the entire display. (set all offsets and positions to zero)

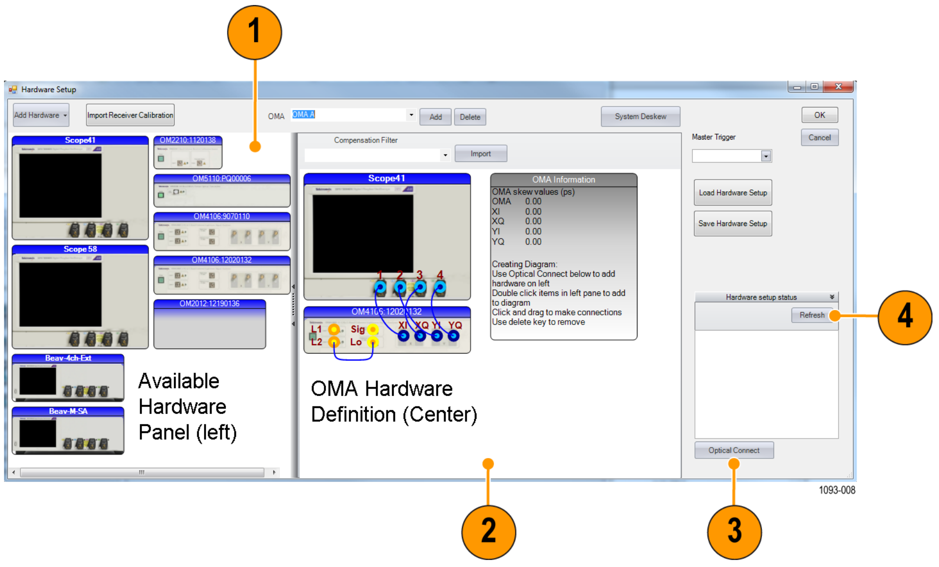
d. If not using the autoscale feature, set the vertical scale on the oscilloscope so that the waveforms fill 80% of the screen. If the DUT has significant peaking in the frequency response, it may be necessary to reduce the initial scale, but cable losses will compensate for typical response peaking so that the amplitude at low frequencies should be representative of the maximum amplitude over the frequency sweep.

***NOTE.** Do not launch the ThorLabs software unless using it as a diagnostic. The ThorLabs software and the Receiver Test software compete for the same resource and should not be run at the same time.*

Software operation

Hardware setup utility - Connectivity for Receiver Test

Indicate how the DUT is connected by adding equipment to the central OMA Definition Panel and adding connections, as shown below. Add equipment by double clicking on items in the Available Hardware Panel.



Creating Diagram:
Use Optical Connect below to add hardware on left
Double click items in left pane to add to diagram
Click and drag to make connections
Use delete key to remove

1. Double-click on an item in the **Available Hardware panel (left)** to add it to the **OMA Definition Panel (center)**.

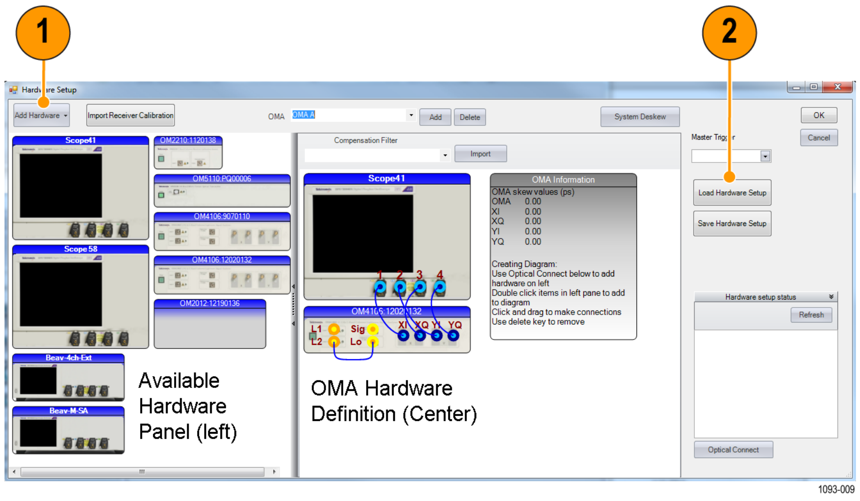
2. Click and drag to make OMA connections matching the actual configuration in the **OMA Definition Panel (center)**.

3. Click **Optical Connect** to auto-configure a list of available OM4K, OM5K, and OM2K devices and IP addresses.

4. Click **Refresh** for a list of available oscilloscopes (will appear in the left panel), and reload the OMA definition (center).

NOTE. *This Hardware Setup utility does not adequately support the Receiver Test of a DUT with differential outputs since only four oscilloscope connections can be shown per OMA. So, for the case of the differential-output DUT, the physical connections will not match the diagram. In this case, it is important to make the channel assignments as shown in the figure: XI -> Channel 1; XQ -> Channel 2; YI -> Channel 3; YQ- Channel 4.*

The Receiver Test software will use the S46T switch matrix to make the proper connections automatically since only four oscilloscope channels are available at one time. When the S46T switch is not being used, The Receiver Test software will prompt the user to manually switch the cable connections.



OMA skew values (ps)

```
OMA    0.00
XI     0.00
XQ     0.00
YI     0.00
YQ     0.00
```

Creating Diagram:
Use Optical Connect below to add hardware on left
Double click items in left pane to add to diagram
Click and drag to make connections
Use delete key to remove

1. Click **Add Hardware** to include generic DUT or add by IP address if hardware does not appear in the left panel after an Optical Connect and Refresh.

2. Click to **Save** or **Load** an OMA definition file.

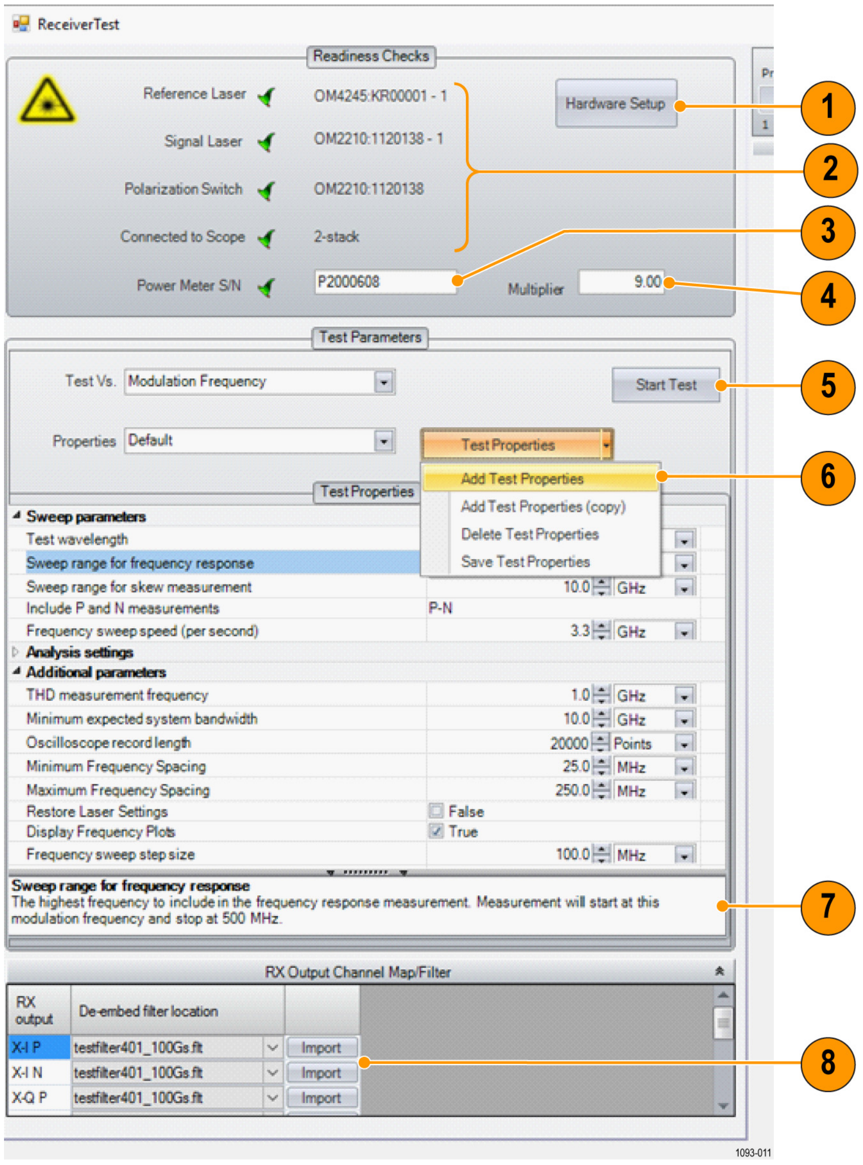
NOTE. *Only the most recent definition is automatically saved on exit.*

The following table lists the supported Receiver Test configuration and their oscilloscope channel mappings:

Table 24: Receiver Test configurations and channel mappings

Dual-polarization coherent DUT Type	Direct scope connection	Tri-Mode Probe connection to scope	S46T switch connection
Single-ended outputs	Any	N/A	N/A
Differential outputs testing P or N only	Any	Any; TriMode probe manually set to desired input	N/A
Differential outputs testing P and N separately	Any, but follow prompts to move cables for P and N	N/A	Fixed mapping
Differential outputs testing P-N	Fixed mapping , but follow prompts to move cables for X and Y	Any; use P only processing and manually set probe to P-N	Fixed mapping
Any: Any channel mapping that matches hardware setup utility OMA definition diagram			
N/A: Not applicable or not supported at this time			
Fixed mapping: Hardware Setup Utility Wiring diagram from DUT to oscilloscope must have XI -> Channel 1; XQ -> Channel 2; YI -> Channel 3; YQ- Channel 4. Actual wiring from DUT to S46T switch or scope should follow the diagram provided with the switch and/or software prompt.			

Test vs modulation frequency



Reference Laser ✓	OM4245:KR00001 - 1
Signal Laser ✓	OM2210:1120138 - 1
Polarization Switch ✓	OM2210:1120138
Connected to Scope ✓	2-stack
Power Meter S/N ✓	P2000608
Multiplier	9.00

Test Vs. Modulation Frequency

Properties Default

Sweep parameters		
Test wavelength		
Sweep range for frequency response		
Sweep range for skew measurement	10.0	GHz
Include P and N measurements	P-N	
Frequency sweep speed (per second)	3.3	GHz
Analysis settings		
Additional parameters		
THD measurement frequency	1.0	GHz
Minimum expected system bandwidth	10.0	GHz
Oscilloscope record length	20000	Points
Minimum Frequency Spacing	25.0	MHz
Maximum Frequency Spacing	250.0	MHz
Restore Laser Settings	False	
Display Frequency Plots	✓ True	
Frequency sweep step size	100.0	MHz

Sweep range for frequency response
The highest frequency to include in the frequency response measurement. Measurement will start at this modulation frequency and stop at 500 MHz.

RX Output Channel Map/Filter

RX output	De-embed filter location	
X-I P	testfilter401_100Gs.flt	Import
X-I N	testfilter401_100Gs.flt	Import
X-Q P	testfilter401_100Gs.flt	Import

Intro para...

1. Click to launch Hardware Setup Utility.
2. Set by Hardware Setup Utility.
3. ThorLabs meter serial number.
4. Split ratio for meter connection 90%/10% splitter = 9.
5. Start or Stop the test.

6. Create new sets of properties to choose in drop-down menu. These can also be chosen via programmatic interface, but are created manually.
7. Help text for selected property.
8. Click to browse to location of correction filter and add to the drop-down menu. Filters must be the same length for each channel and match scope time spacing if used.

   The filter format is that used by the SDLA application. The correction filter chosen should compensate for the entire path including test-system skew.

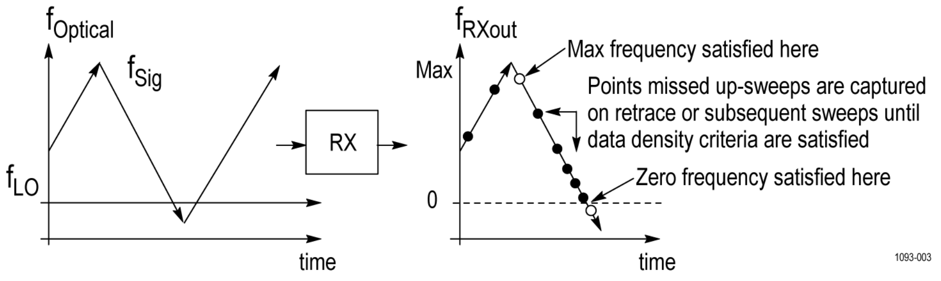
System calibration

The receiver calibration is based on the following:

1. Factory frequency response and gain calibration to the oscilloscope input.
2. UltraSync timing stability between channels.
3. Deskew to the end of the test cables using the TekScope application Vertical\Deskew menu.
4. Eight compensation filters created by SDLA compensate for the frequency response of the interconnecting coaxial cables, S46T-switch if used, and DUT test fixture.
5. The eight compensation filters also compensate for any additional skew from the test cables to the DUT. This can be done via the tap weights in the filter, or a DELAY statement at the top of the filter file. The filter files are in SDLA format.

It is a critical point that all system skews must be corrected either by the TekScope deskew values or by the compensation filters or a combination of the two. Any remaining skew will be attributed to the DUT.

Principles of operation


$f_{Optical}$
f_{Sig}
f_{LO}
time
RX
f_{RXout}
Max
0
Max frequency satisfied here
Points missed up-sweeps are captured on retrace or subsequent sweeps until data density criteria are satisfied
Zero frequency satisfied here
time
1093-003

The laser connected to the signal input of the RX is put into sweep mode at the beginning of the test. The laser connected to the LO input is held at a constant frequency. The receiver will output an rf signal at the heterodyne frequency (the difference between the Signal and LO laser frequencies). The software sets the Signal-laser frequency scan range to ensure that the frequency difference both reaches the user's specified "Sweep range for frequency response," and crosses zero as shown in the figure above (right panel). The software can detect the zero crossing as long as the receiver IQ phase error at low frequencies is less than 20 degrees.

As the laser scans over the user-selected range, the software collects data from the oscilloscope as fast as possible to achieve the smallest possible frequency spacing between data acquisitions. After a sweep is completed, the precise frequency, amplitude, and phase of each signal waveform are determined and used to calculate the results such as frequency response and relative-phase response.

The following frequency scan specifications can be set by the user in the "Test Properties" panel:

Table 25: Test properties

Type	Definition
Sweep range for frequency response	The highest frequency to include in the frequency response measurement. Because of laser tuning error (where the actual laser frequency differs from the frequency setting), a guard-band is added to the sweep range. Therefore, frequencies slightly larger than the sweep range may be acquired.
Frequency sweep step size	The mean value of the spacing between acquired frequencies should be less than or equal to this value. This will also be the frequency step used by the plots. The actual frequency spacing between adjacent frequencies will vary due to fluctuations in data collection speed.
Maximum frequency spacing	The spacing between acquired frequencies should be less than this value. Use this parameter to prevent missing spectral features of a certain width.
Minimum frequency spacing	The spacing between frequencies collected should be greater than this value. Set this higher to reduce the size of data sets. Set it lower for best overall performance. Default is 20 MHz. This parameter should always be ≥ Scope sampling rate/Scope Record length.

Table 25: Test properties (cont.)

Type	Definition
Frequency sweep speed	This is the rate at which the Signal Laser will tune. Faster tuning speed will result in a faster test unless the speed is so fast that extra data collection is needed to satisfy the other frequency sweep specifications (see below).
Minimum expected system bandwidth	If the DUT does not have sufficient bandwidth to permit measurement of highest frequencies in the designated frequency response sweep range, the test will terminate normally if frequencies are at least collected up to this value.

The system will continue to scan over frequency until the following conditions are met, or until the maximum number of data acquisitions are acquired:

- A frequency ≥ Sweep range has been acquired
- A frequency below 0 has been acquired
- The mean spacing between acquired frequencies is ≤ the Frequency sweep step size
- All frequency spacings are ≤ Maximum frequency spacing

Test vs modulation frequency results

Once the test vs. modulation frequency is complete, the results are available in various forms:

1. If ‘Show frequency plots’ is enabled in the Test Properties panel, the results will be plotted in a separate Matlab window. These plots may be edited, copied, or saved in the Matlab user interface.
2. Selected numerical results are printed to two formatted string variables in the Matlab work space and are also available via the automated-test interface.
   a. **OutputStr** contains the frequency response of each port of the DUT as well as any error or warning messages.
   b. **WvlenStr** contains the low-frequency parameters that are typically collected during a wavelength sweep such as phase angle, skew, and THD. This string also includes error or warning messages.
3. All numerical results will be available in the Matlab work space.
   a. All frequency response data is in the **RxTestRslts** variable.
   b. The magnitude of worst-case values will be found in the **RxTestNR** variable.

Test vs Wavelength

A Test Vs. Wavelength is essentially just a Modulation Frequency test repeated at a number of different wavelengths. To speed the results, normally a smaller sweep range is used for the wavelength test since the focus is on how the DUT properties at a low heterodyne frequency vary with wavelength. However, it is possible to test over the same frequency range as in the Modulation Frequency test and so obtain a frequency response vs. wavelength (including the 3-dB point of the response vs. wavelength).

The Test Properties are essentially the same as for the Test Vs. Modulation Frequency except that it is necessary to specify the start and stop wavelengths as well as the wavelength step. If the wavelength step does not divide evenly into the wavelength range, the test will stop at the last step before the stop wavelength. The test always proceeds from lowest optical frequency to highest, so a wavelength scan will run from longest wavelength to shortest. It is possible to choose various units for the start and stop values. If "Channel" is chosen as the unit, then the values will round to the nearest channel on default grid. To hit a wavelength or frequency exactly, be sure to use wavelength or frequency units. The default channel grid has channel 1 at the first available laser channel and a grid determined by the reference laser grid at the time the Test Parameter list is created. These values are saved with the properties so that the test is reproducible.

Test vs wavelength results

Once the test vs. modulation frequency is complete, the results are available in various forms:

1. If '**Show wavelength plots**' is enabled in the Test Properties, the results vs. wavelength will be plotted in a separate Matlab window. These plots may be edited, copied, or saved in the Matlab user interface.
2. If '**Show frequency plots**' is enabled in the Test Properties, the results will be plotted in a separate Matlab window for each wavelength. These plots may be edited, copied, or saved in the Matlab user interface.
3. Selected numerical results are printed to the **WvlenStr** variable in the Matlab workspace and are also available via the automated-test interface.
4. All numerical results will be available in the Matlab workspace.
   a. All frequency response data is in the **RxTestRslts** variable which is indexed by wavelength step.
   b. The worst case values over the measurement range are found in the **RxTestNR** variable.

Receiver Test ATE Commands

The Receiver Test software has its own service that can be invoked to control the software from an external program.

Constructors

```
RXTestATEClient() :  this("localhost", 9500)
```

```
RXTestATEClient(string IPAddress, int port) :
this(IPAddress, port, 60, 60, 60, 1000000, 1000000,
1000000)
```

```
RXTestATEClient(int openTimeout_s,
   int sendTimeout_s,
   int receiveTimeout_s,
   int maxReceivedMessageSize,
   int MaxBufferSize,
   int MaxBufferPoolSize) :  this("localhost", 9500,
      openTimeout_s,
      sendTimeout_s,
      receiveTimeout_s,
      maxReceivedMessageSize,
      MaxBufferSize,
      MaxBufferPoolSize)
```

```
public RXTestATEClient(string IPAddress, int port,
   int openTimeout_s,
   int sendTimeout_s,
   int receiveTimeout_s,
   int maxReceivedMessageSize,
   int maxBufferSize,
   int maxBufferPoolSize)
```

Methods

```
void SignalSwitchChanged()

void StartTest()

void StopTest()

void StartATEControl()

bool ScreenShot(string fileName)

bool LoadTestProperties(RXTestService.testtype testType,
string testName)

double[] PerformScopeAutoscale(RXTestService.pn pn,
RXTestService.pnprocessingmethod pnProcessingMethod, double
autoscaleNumber = .23)

void
TestFinished(RXTestAPI.RXTestATEClient.RXTestService.TestEventArgs
eventArgs)
```

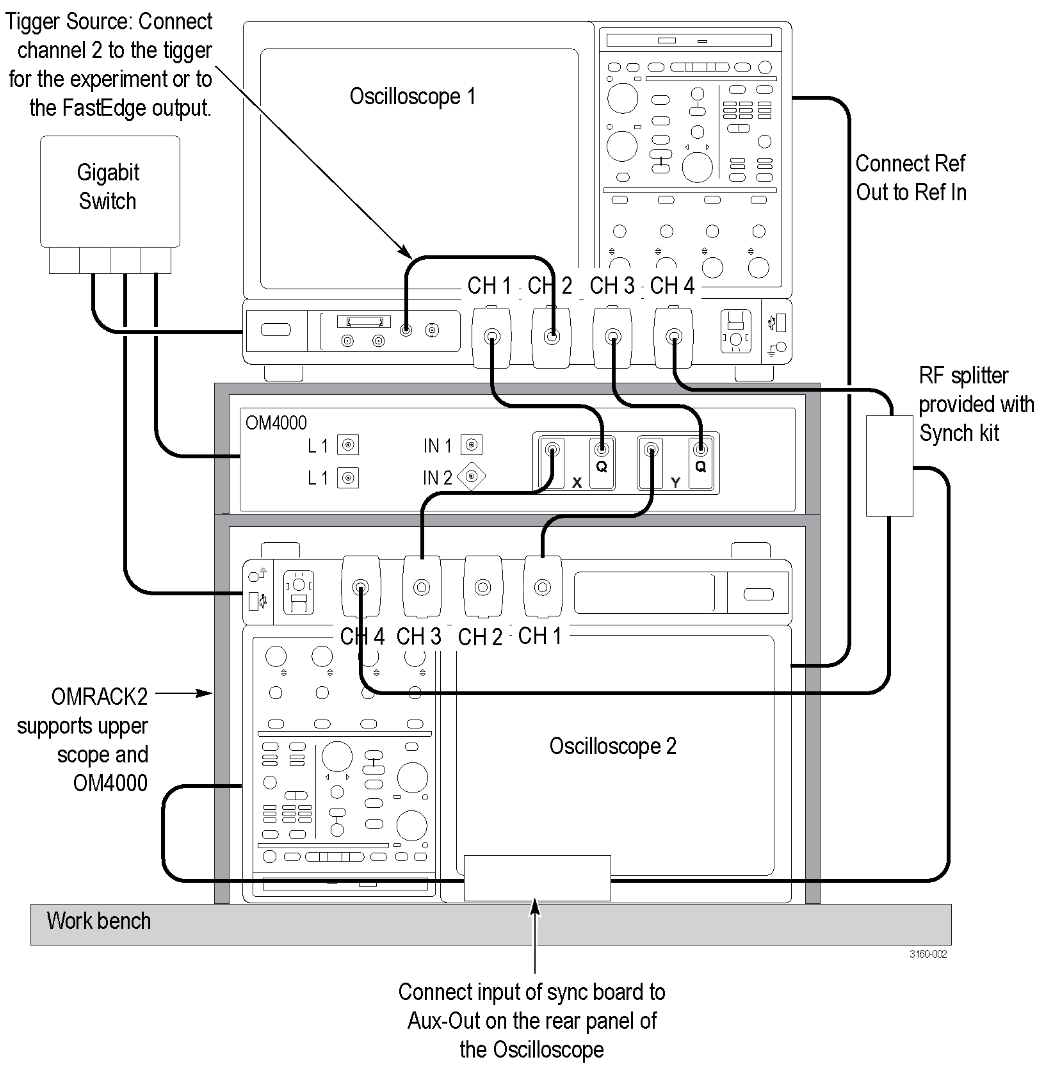
Appendix F: Configuring two Tektronix 70000 series oscilloscopes

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

NOTE. *This method is not needed for the Tektronix 70000SX models, which can function together as a single oscilloscope using the UltraSync feature.*

You can connect one receiver polarization to each oscilloscope. However, with the recommended hardware configuration shown below, you can use the SkewControl function to remove inter-oscilloscope 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.


Tigger Source: Connect channel 2 to the tigger for the experiment or to the FastEdge output.
Gigabit Switch
Oscilloscope 1
CH 1 CH 2 CH 3 CH 4
Connect Ref Out to Ref In
RF splitter provided with Synch kit
OM4000
L 1
L 1
IN 1
IN 2
X
Q
Y
Q
CH 4 CH 3 CH 2 CH 1
OMRACK2 supports upper scope and OM4000
Oscilloscope 2
Work bench
3160-002
Connect input of sync board to Aux-Out on the rear panel of the Oscilloscope

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 oscilloscope) 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. Make sure 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 oscilloscope; 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 OMA 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

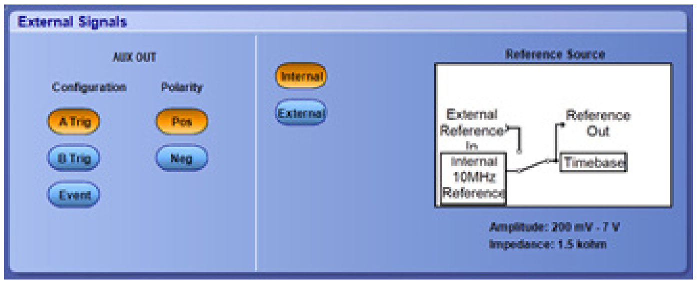
Oscilloscope settings

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

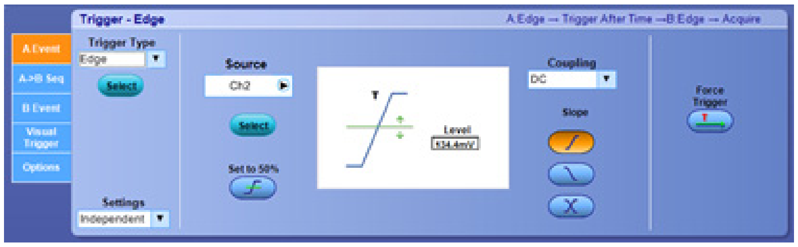
NOTE. *The following screens may look different depending on the version of oscilloscope firmware.*

Master oscilloscope trigger settings

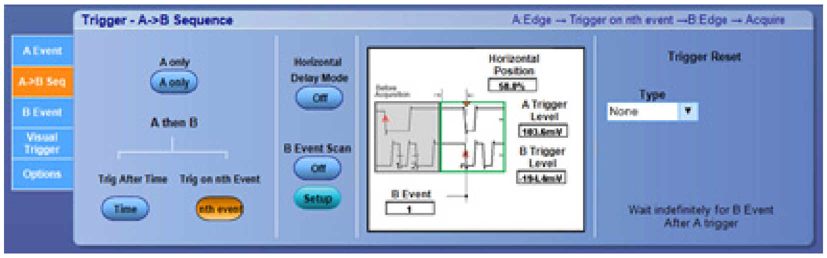
- Set up the Aux-Out on the master oscilloscope to provide a positive edge after the A event.
- Set the master Reference to Internal



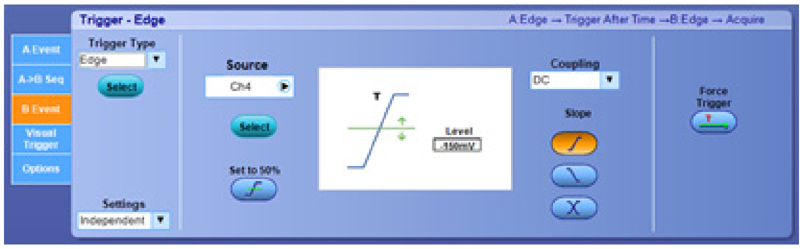
- Set the A Event to Ch 2 with appropriate settings for the Ch 2 input. The settings shown are for the Fast Edge oscilloscope output. This is the primary system trigger.



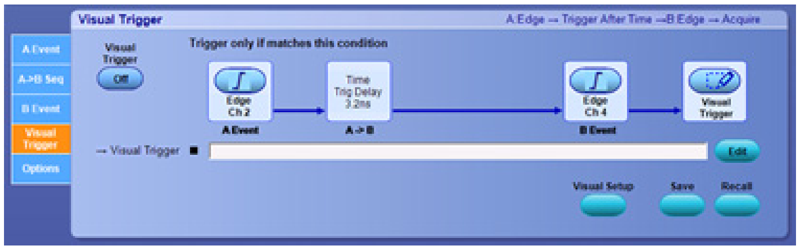
- The master oscilloscopes gets two triggers: the primary “A” trigger which arms the system, and the “B” trigger from the Sync Board which provides the low jitter trigger relative to the slave oscilloscope. Both master and slave need to react immediately to the “B” trigger so the Trigger on nth Event is the best choice with “B Event” set to 1.



- The “B” event levels are set appropriate for the sync board output



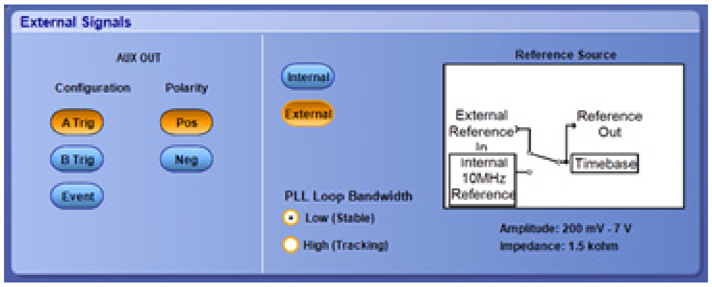
- The visual trigger is not used because Ch 2 and 4 must be turned off to enter 100 Gs/s mode



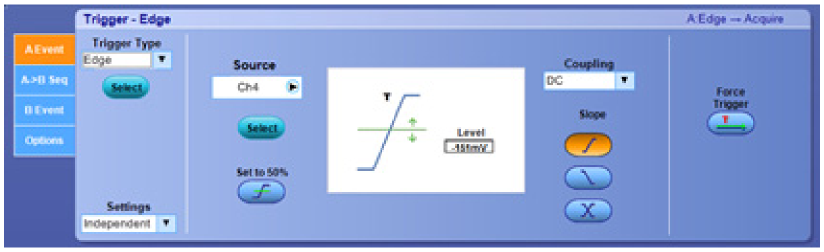
- The trigger holdoff is set to a fixed time which is managed by the OMA. Longer holdoffs are required for longer records.
- In the Options tab, set the Master Scope Trigger Holdoff as required. Make sure that the Slave Scope Trigger Holdoff is set to minimum time.

Slave oscilloscope trigger settings

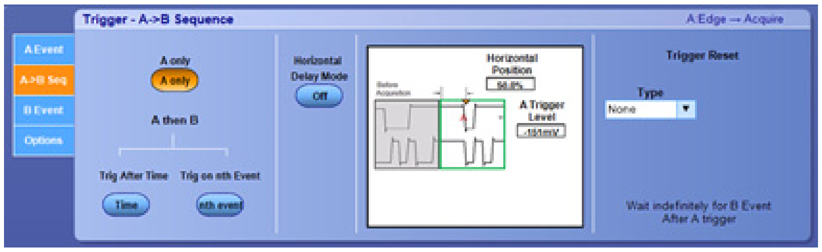
- The slave uses the Reference signal from the master via the rear-panel BNC

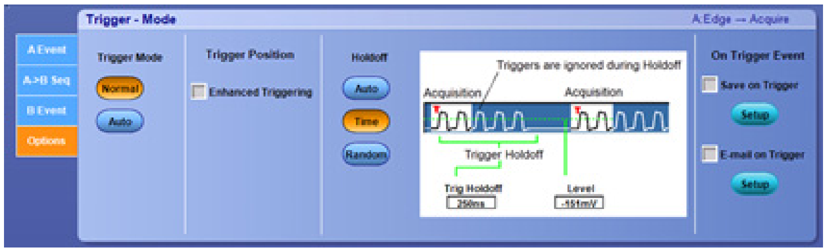


- The Ch 4 trigger settings should be identical to the master so that both take data simultaneously. Residual skew will be handled by the OMA.



- The slave oscilloscope has only one trigger. Select the default or minimum holdoff.



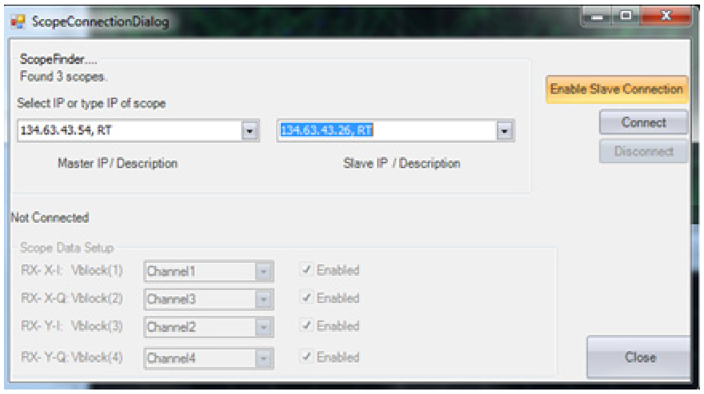


- Turn off Channels 2 and 4 when trigger is setup.
- Set the oscilloscope for 100 Gs/s operation on channels 1 and 3.
- Select the maximum corrected bandwidth (not the HW setting).

OMA settings for two-oscilloscope operation

NOTE. *This section applies only when connected to two Tektronix model 70000C/D/DX oscilloscopes. When using two or more DPO70000SX oscilloscopes, the instrument's UltraSync multi-unit synchronization bus sets one oscilloscope as a Master while one or more additional units function as Extensions. OMA only needs to connect to and control the Master oscilloscope when using this configuration.*

1. On the OMA, click Connect on the Setup Tab.



ScopeFinder....
Found 3 scopes.
Select IP or type IP of scope

Master IP / Description: 134.63.43.54, RT
Slave IP / Description: 134.63.43.26, RT

Not Connected

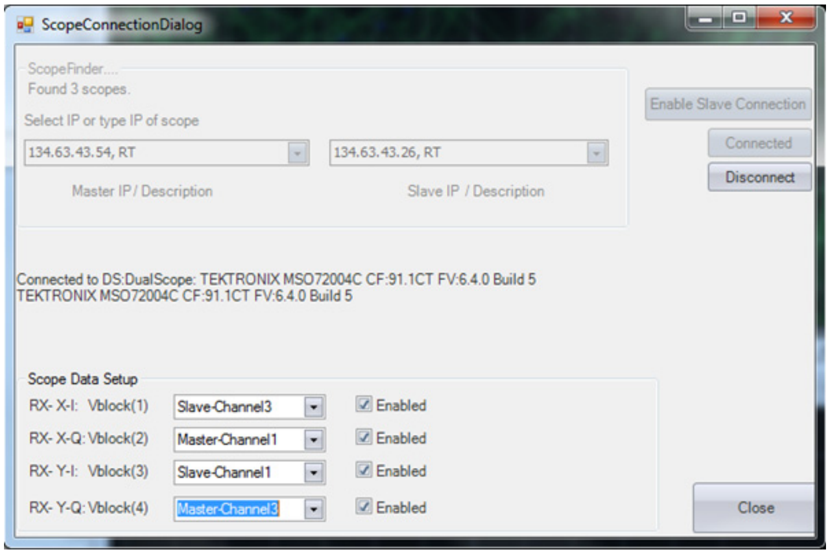
Scope Data Setup

RX- X-I: Vblock(1)	Channel1	✓ Enabled
RX- X-Q: Vblock(2)	Channel3	✓ Enabled
RX- Y-I: Vblock(3)	Channel2	✓ Enabled
RX- Y-Q: Vblock(4)	Channel4	✓ Enabled

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 oscilloscope is the one triggering on the sync board output only.

If the OMA 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, use the drop-down menus to enter required values. You can select channels from both the Master and Slave instruments.



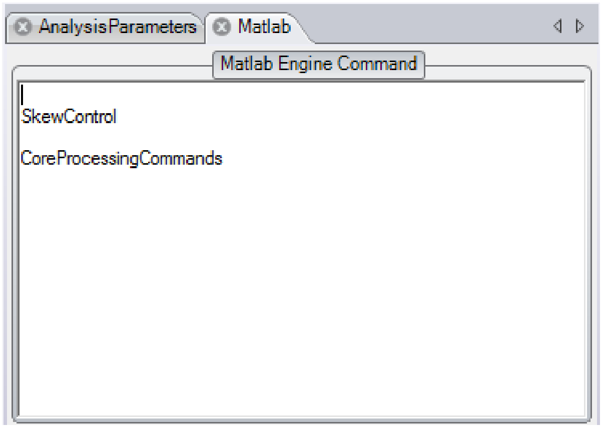
Found 3 scopes.
Select IP or type IP of scope
134.63.43.54, RT
134.63.43.26, RT
Master IP / Description
Slave IP / Description

Connected to DS:DualScope: TEKTRONIX MSO72004C CF:91.1CT FV:6.4.0 Build 5
TEKTRONIX MSO72004C CF:91.1CT FV:6.4.0 Build 5

Scope Data Setup

RX- X-I: Vblock(1)	Slave-Channel3	✓ Enabled
RX- X-Q: Vblock(2)	Master-Channel1	✓ Enabled
RX- Y-I: Vblock(3)	Slave-Channel1	✓ Enabled
RX- Y-Q: Vblock(4)	Master-Channel3	✓ Enabled

3. To remove inter-oscilloscope jitter, add the SkewControl command to the Engine Window as shown in the following figure. SkewControl removes oscilloscope to oscilloscope 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.



```
SkewControl

CoreProcessingCommands
```

Appendix G: Alert codes

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

Table 26: Alert code descriptions

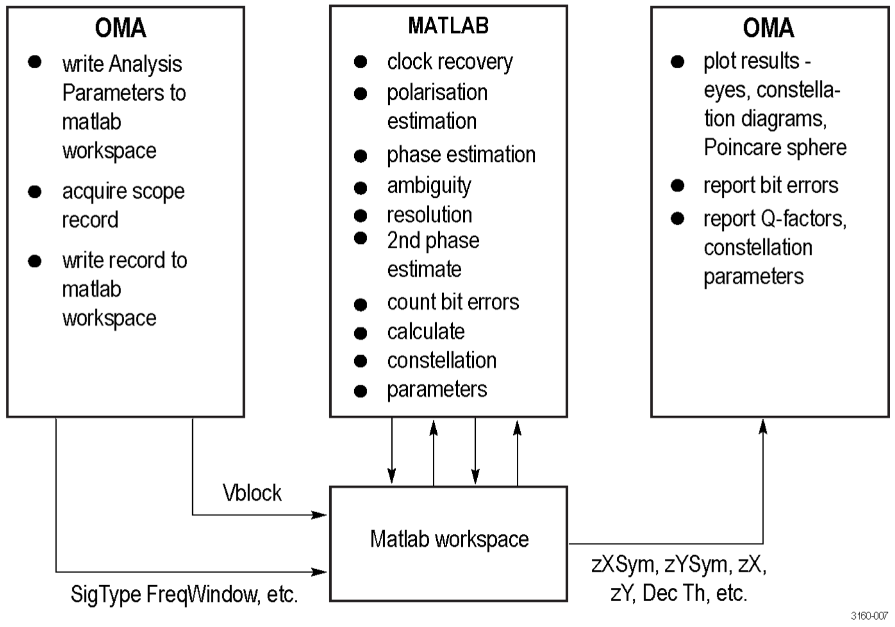
Code	Calling function	Description
1	EngineCommandPre	Local oscillator (LO) frequency not set. Set the LO frequency automatically by opening the Laser Receiver Control Panel (LRCP) tab, or manually under the Setup tab.
2	EngineCommandPre	Local oscillator (LO) power not set. Set the LO power automatically by opening the Laser Receiver Control Panel (LRCP) tab, or manually under the Setup tab in OMA.
3	EngineCommandPre	Channel delays are not specified. Set the channel delays under the Calibrate tab in OMA.
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 oscilloscope 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 = abs(X).^2 instead.
202	EstimatePhase	Cannot evaluate NonlinFunc or does not give usable Y; used Y = abs(X(1,:)).^2+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 26: 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 polarization signal.
400	AlignTribs	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	AlignTribs	Data pattern synchronization may be incorrect because %s.SyncFrameEnd <50.
421	AlignTribs	Number of bits too small to recover PRBS. Use longer block, or shorter PRBS in %s.
422	AlignTribs	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	EngineCommandPost	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 H: MATLAB CoreProcessing software guide

MATLAB interaction with OMA



The core processing software performs all the computations to obtain the quantities displayed in the OMA software, 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 OMA. The core processing code and the OMA exchange data via the MATLAB workspace.

The order of processing for each acquisition is as follows:

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

   By default, the MATLAB Engine Commands window contains one instruction, CoreProcessing, or CoreProcessingCommands which calls either the real-time or equivalent-time versions of Core Processing (CoreProcessing.m or CoreProcessingET.m). This file computes the various output variables which then reside in the MATLAB workspace.

5. OMA retrieves output variables from MATLAB workspace
6. OMA displays output variables as eye diagrams, constellation diagrams, numerical values, and so on.

In fact, when the record size is larger than the block size, multiple calls to CoreProcessing are executed for each oscilloscope record. This mode of processing is known as block processing. (See page 169, *MATLAB block processing*.) When in block processing mode for a record length of less than one million points, most signal metrics, such as Q factor, are calculated after all blocks are complete. This post processing is performed by the file EngineCommandPost.m.

To customize the signal processing of the OM4000, the user must modify the commands in the OMA MATLAB Commands tab, 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.

NOTE. *The user's Matlab directory is earlier in the path than the directory with the OMA functions. Any .m files in the user's Matlab directory with the same name as those used by the OMA will be executed instead of the OMA versions. Keeping modified .m files in the user's Matlab directory is preferred because those in the OMA install directory are removed if the OMA software is uninstalled.*

This Section will discuss the code in CoreProcessing.m, and how it derives the output variables from the oscilloscope 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 *MATLAB Core Processing function reference* section. (See page 207.)

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:

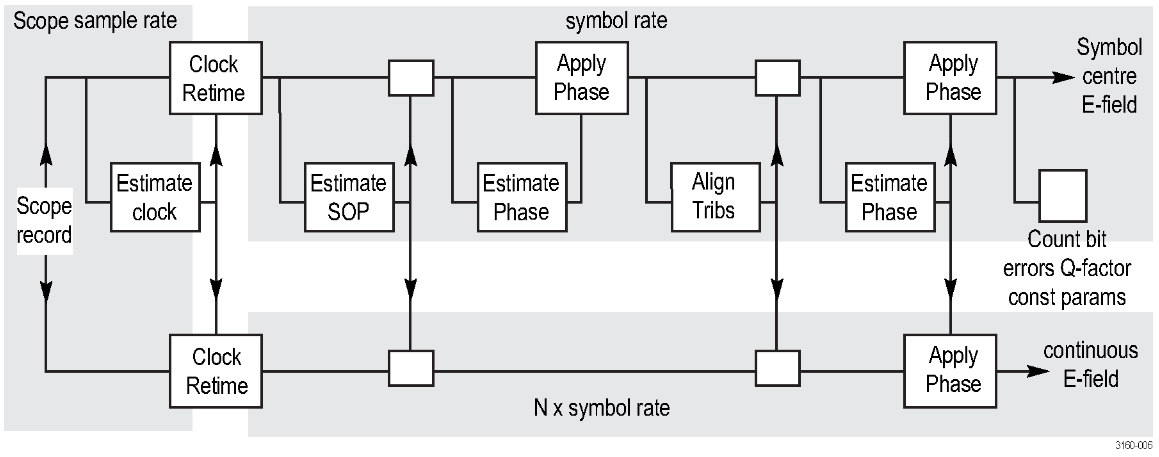
- BoundVals is used in block processing mode. (See page 169, *MATLAB block processing*.) It contains information about the tail end of the previous block, and is used to ensure continuity with the next block.
- Alerts is a variable containing status information, typically warnings and error messages, accumulated from all the functions called so far. (See page 171, *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 you can cause an error that cannot be trapped by having the input variables in the wrong order. Each function has a detailed description. (See page 207, *MATLAB CoreProcessing function reference*.) Help text is available by typing **help FunctionName** at the MATLAB prompt.

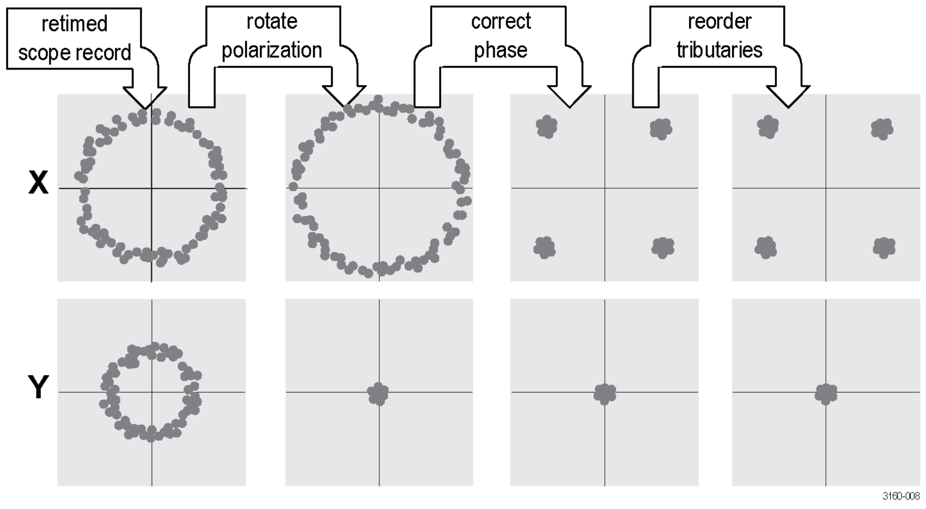
Signal processing steps in MATLAB CoreProcessing

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 oscilloscope 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 214, *EstimateClock.*) In addition to Vblock, EstimateClock takes as inputs ChDelay and pHyb. ChDelay contains the relative delays between the four oscilloscope channels, due to cable length mismatches for example. It is obtained from the delay calibration routine.

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 oscilloscope 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 in the OM4000 instrument. 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 looks like 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 AlignTribs 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 OMA with the data patterns specified in the Analysis Parameters tab. (See page 52, *Analysis Parameters control and configuration tab*.) If the inphase component of zXSym contains the data pattern of the quadrature component, and vice versa, then AlignTribs produces an output DRotM which is equivalent to a quarter turn to correct the phase of zXSym. AlignTribs decides the phase correction. (See page 207, *AlignTribs*.) The function uses several criteria, including tie-breaker criteria if both tributaries have the same data pattern.

When a dual polarization signal is used AlignTribs may exchange the SOPs and adjust the phase. If AlignTribs were not applied then the tributaries would be randomly mapped to different components in the OMA 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.

AlignTribs returns the pattern variables, PattXRe, and others, 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 part 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 part, 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 similarly 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 OMA.

The mean and standard deviation of the constellation points are calculated later in EngineCommandBlock. (See page 169, *MATLAB 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 OMA 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 165, *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.

MATLAB block processing

The OMA software handles large record sizes, for example 250M samples, by breaking down the record into smaller pieces, or blocks, and then processes each block 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 to 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 OMA, 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 block processing interaction between the OMA and CoreProcessing are as follows: (See page 161.)

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

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 OMA, 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 OMA, 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.

NOTE. *The larger variables, such as fine traces and symbols, are not aggregated when the record size is greater than 1 million points. Constellation metrics are always aggregated.*

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 OMA then displays a list of all the alert messages that apply.

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 OMA.

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.

Alert codes and descriptions are provided in the appendices. (See page 159, *Alert codes*.)

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
```

Appendix I: The ATE (automated test equipment) interface

The Automated Test Equipment (ATE) interface exposes the OMA user interface functionality through Windows Communication Foundation (WCF) services to enable control from a user application. The services (basic and advanced) are compatible with simple interfaces such as MATLAB and client application programs.

There are two services available; basic and advanced. The basic service is implemented using a simple binding to be compatible with applications like MATLAB or Labview. The advanced service, implemented using a netTcpBinding (which is not available in MATLAB), was developed to use events to provide a time-efficient interface. Both services provide most of the functionality that is available through the OMA user interface.

The ATE functionality is also grouped by instrument interface (referred to as LRCP in the following sections) and OMA interface.

You can use your choice of .NET language (such as C# or VB.NET) to develop your ATE application.

The LRCP ATE interface

LRCP basic service interface

- The basic service for LRCP is available on port 9000.
- The basic service uses wsBasicHTTPBinding to be compatible with applications like MATLAB or Labview that only support the simpler binding.
- Exposes a subset of the advanced service commands.
- The basic service is referenced at the following URL:

  http://localhost:9000/LaserReceiverControlPanel/Laser_ReceiverServiceBasic/

LRCP advanced service interface

- The LRCP advanced service is available on port 9300.
- The LRCP advanced service uses a netTcpBinding (which is not available in MATLAB) and uses events to provide a time-efficient interface.
- The advanced service is referenced at the following URL:

  net.tcp://localhost:9300/LaserReceiverControlPanel/Laser_ReceiverService/
- Wrapper client DLLs (LRCPATEClient.DLL) have been installed into your documents folder under .\TekApplications\ATE Support Files. Copy the appropriate DLL to your ATE application's project folder and add a reference to the DLL to your project.

NOTE. *In previous releases it was required that the user edit their App.Config file for the client applications to supply URL information for accessing the advanced service. This is no longer necessary if user adds the reference to LRCPATEClient.DLL to their client ATE application. The detail of the WCF Service is wrapped in the DLL.*

- A new constructor has been added to the LRCPATEClient class to allow the user to change the service's URL and/or port. This is useful for accessing the service remotely or for dealing with port conflicts.

  Syntax: LRCPATEClient(string url, int port)

  Coding example: (C#):

  LRCPATEClient myLRCPService = new LRCPATEClient("155.90.55.23", 9300);

NOTE. *The advanced service binding in earlier versions of OMA software was wsHTTPBinding, referenced at URL http://localhost:9000/LaserReceiverControlPanel/Laser_ReceiverService/. These are not available in the new OMA software. Make sure to use the new binding and URL when you update your ATE code to run with the new OMA software.*

***NOTE.** MATLAB returns strings, not numeric values. To convert returned string values to numeric values, use the following: VarName = CmndName(obj); x = str2num(VarName). For example: LoFreq = GetLOFreq(obj); x = str2num(LoFreq).*

***WARNING.** WCF services can turn lasers on and off. Verify that no one is physically working with lasers or fibre while running ATE applications. Power off the OM instruments to disable lasers.*

LRCP ATE service function list

The following are the available instrument commands (OM4000 series, OM2000 series, OM5110) for both the basic and advanced service interfaces, and show their functionality using the MATLAB syntax. Each command lists the instruments with which that command operates.

- **int AvailableLasers(classname);**
  Description: Returns the count of available lasers on the active controller.
  Controller Types: All
  Example: `AvailableLasers(obj);`
  Returns: ans = 2

- **bool Calibrate(classname);** (OM5110 only)
  Description: Performs an automatic modulator calibration to determine the optimal modulator parameters. These are the same parameters that are manually set using the user interface Set Params button, or the SetManualCalibration() function.

  Calibrate() is an automatic calibration that requires the modulator control mode (see GetActualModulatorMode()) be set to 2-pol QPSK and that >500 mV_{pp} binary signals are applied to all four inputs. Longer patterns are better (2^{31} PRBS is optimal). Each of the four input patterns should be different in some way: a different pattern, the same pattern delayed, or a different seed.

  The modulator must receive adequate input power levels to produce a signal power that is high enough to avoid a power level warning. If these conditions are not met the resulting calibration can result in unstable optical bias. See the section on Set Paramaters to restore these values to initial (factory) values.
  Controller Types: OM5110
  Example: `Calibrate(obj);`
  Returns: ans = true/false

- **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 active controller, takes offline.
  Controller Types: All
  Example: `Disconnect(obj);`
  Returns: ans = true

- **bool GetActualCavityLock(classname);**
  Description: Returns the actual cavity lock state for the active controller/laser. Locked = True.
  Controller Types: All
  Example: `GetActualCavityLock(obj);`
  Returns: ans = true

- **double GetActualChannel(classname);**
  Description: Returns the actual channel number for the active controller/laser.
  Controller Types: All
  Example: `GetActualChannel(obj);`
  Returns: ans = 1

- **double GetActualChannel1(classname);**
  Description: Returns the actual channel 1 frequency (in THz) for the active laser.
  Controller Types: All
  Example: `GetActualChannel1(obj);`
  Returns: ans = 191.5

- **controlmode GetActualControlMode(classname, enum_modulator);** (OM5110 only)
  Description: Returns the control mode for the modulator that is passed in the 'modulator' parameter.
  Controller Types: OM5110
  Example: `GetActualControlMode(obj, modulator.YQ);`
  Returns: ans = one of the following:
  `controlmode.automatic`
  `controlmode.manual`
  `controlmode.notset`

- **float GetActualCurrent(classname, string_voltageName);** (OM5110 only)
  Description: Returns current (mA) associated with the specified input voltage.
  Controller Types: OM5110
  Example: `GetActualCurrent(obj, 'YQ G1');`
  Returns: ans = 30

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **bool GetActualEmitting(classname);**
  Description: Returns the emission status of the active laser. Emitting = True.
  Controller Types: All
  Example: `GetActualEmitting(obj);`
  Returns: ans = true

- **byte GetActualFactoryDefault(classname, string_voltageName);**
  (OM5110 only)
  Description: Returns the factory default value for the specified voltage, in the range of 0 to 255.
  Controller Types: OM5110
  Example: `GetActualFactoryDefault(obj, 'XQ D2');`
  Returns: ans = 0

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **short GetActualFineTuneFrequency(classname);**
  Description: Returns the actual fine tune frequency (in MHz) of the active laser.
  Controller Types: All
  Example: `GetActualFineTuneFrequency(obj);`
  Returns: ans = 0

- **double GetActualGridSpacing(classname);**
  Description: Returns the actual grid spacing (in THz) of the active laser.
  Controller Types: All
  Example: `GetActualGridSpacing(obj);`
  Returns: ans = 0.05

- **byte GetActualMaxPotSetting(classname, string_voltageName);**
  (OM5110 only)
  Description: Returns the maximum allowed step setting value for the specified voltage, in the range of 0 to 255
  Controller Types: OM5110
  Example: `GetActualMaxPotSetting(obj, 'XI G1');`
  Returns: ans = 100

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **float GetActualMaxVoltage(classname, string_voltageName);** (OM5110 only)
  Description: Returns the maximum voltage value for the specified voltage, in Volts.
  Controller Types: OM5110
  Example: `GetActualMaxVoltage(obj, 'YI D2');`
  Returns: ans = 10.2

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **byte GetActualMinPot(classname, string_voltageName);** (OM5110 only)
  Description: Returns the minimum allowed step setting value for the specified voltage, in the range of 0 to 255
  Controller Types: OM5110
  Example: `GetActualMinPot(obj, 'XI G2');`
  Returns: ans = 100

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **float GetActualMinVoltage(classname, string_voltageName);** (OM5110 only)
  Description: Returns the minimum voltage value for the specified voltage, in Volts.
  Controller Types: OM5110
  Example: `GetActualMinVoltage(obj, 'YQ G1');`
  Returns: ans = 5.5

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **modulatormode GetActualModulatorMode(classname);** (OM5110 only)
  Description: Returns the modulator mode (notset, dpqpsk, qam, or arbirary)
  Controller Types: OM5110
  Example: `GetActualModulatorMode(obj);`
  Returns: ans = one of the following:
  modulatormode.notset
  modulatormode.dpqpsk
  modulatormode.qam
  modulatormode.arbitrary

- **double GetActualOffset(classname, enum_modulator);** (OM5110 only)
  Description: Returns the offset voltage adjustment value for the modulator that is in Automatic mode.
  Controller Types: OM5110
  Example: `GetActualOffset(obj);`
  Returns: ans = offset value

- **double GetActualPower(classname);**
  Description: Returns the actual power (in dBm) of the active laser.
  Controller Types: All
  Example: `GetActualPower(obj);`
  Returns: ans = 14.5

- **byte GetActualPowerOnDefault(classname, string_voltageName);** (OM5110 only)
  Description: Returns the power-on default setting for the specified voltage, in the range of 0 to 255.
  Controller Types: OM5110
  Example: `GetActualPowerOnDefault(obj, 'YQ G1');`
  Returns: ans = 150

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **slope GetActualSlope(classname, enum_modulator);** (OM5110 only)
  Description: Returns the slope setting (positive, negative, or notset) of the specified modulator (XI, XQ, YI, YQ).
  Controller Types: OM5110
  Example: `GetActualSlope(obj, modulator.XQ);`
  Returns: ans = one of the following:
  slope.notset
  slope.positive
  slope.negative

- **byte GetActualStep(classname, string_voltageName);** (OM5110 only)
  Description: Returns the step setting for the specified voltage, in the range of 0 to 255.
  Controller Types: OM5110
  Example: `GetActualStep(obj, 'YQ G1');`
  Returns: ans = 10

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **double GetActualVoltage(classname, enum_modulator);** (OM5110 only)
  Description: Returns the voltage setting of the specified modulator (XI, XQ, YI, YQ), in Volts.
  Controller Types: OM5110
  Example: `GetActualVoltage(obj, modulator.YQ);`
  Returns: ans = setting in Volts

- **double GetCalculatedFrequency(classname, enum_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 laserUsageType values are: unused, signalx, signaly, signalxy, reference.
  Controller Types: All
  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.
  Controller Types: All
  Example: `Controllers = GetControllers(obj);`
  Returns:
  'OM5110:Prototype1'
  'OM2210:8180123'
  'OM4006:6300121'

- **double GetFirstFrequency(classname);**
  Description: Returns the first frequency (in THz) of the active laser.
  Controller Types: All
  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: All
  Example: `GetLastFrequency(obj);`
  Returns: ans = 196.25

- **double GetOpticalPower(classname);** (OM5110 only)
  Description: Returns the optical power setting, in dBm.
  Controller Types: OM5110
  Example: `GetOpticalPower(obj);`
  Returns: ans = 12.4

- **double GetOpticalPowerAdjustment(classname);** (OM5110 only)
  Description: Returns the optical power adjustment setting, in dBm.
  Controller Types: OM5110
  Example: `GetOpticalPowerAdjustment(obj);`
  Returns: ans = 10.4

- **float GetPhotoCurrent(classname);**
  Description: Returns the photocurrent (in mA) of the receiver in the active controller.
  Controller Types: OM4245, OM4225, OM4106D, OM4006D
  Example: `GetPhotoCurrent(obj);`
  Returns: ans = 11.034

- **Dictionary<string, float> GetPhotoDiodeCurrents(classname);**
  Description: Returns the photo diode currents (mA) of the individual channels of the active controller.
  Controller Types: OM4245, OM4225
  Example: `GetPhotoDiodeCurrents(obj);`
  Returns: The format depends on the environment. In a .NET application the dictionary contains the keys XQP, XQN, XIP, XIN, YQP, YQN, YIP, and YIN. Access the current values as follows: < return_variable_name> ["RXIP"]

- **enum_polarization GetPolarization(classname);** (OM2210 only)
  Description: Returns the polarization state.
  Valid Polarization states: filter1, filter2, unknown, hardwarefailed.
  Controller Types: OM2210
  Example: `GetPolarization(obj);`
  Returns: ans = filter2

- **double GetPowerByLaserUsageType(classname, enum_laserusagetype);** (OM2210 only)
  Description: Returns the current power reading for the specified laser.
  Valid laser usage types: unused, signalx, signaly, signalxy, reference.
  Controller Types: All
  Example: `GetPowerByLaserUsageType(obj, laserusagetype.signalx)`
  Returns: ans = 13.5

- **bool InitializePolarization(classname);** (OM2210 only)
  Description: Sets the initial polarization state. Returning True = Successful.
  Controller Types: OM2210
  Example: `InitializePolarization(obj);`
  Returns: ans = true

- **bool IsActiveControllerChosen5110Y(classname);** (OM5110 only)
  Description: Returns true if the active controller is an OM5110; returns false if it is not an OM5110.
  Controller Types: OM5110
  Example: `IsActiveControllerChosen5110Y(obj);`
  Returns: ans = true/false

  The ATE methods for the modulator use the following enumerations:

  enum_modulator (YQ, YI, YP, XQ, XI, XP)
  modulatormode (noset, dpqpsk, qam, arbitrary)
  slope (notset, negative, positive)
  controlmode (notset, automatic, manual)

- **bool Reset(classname);** (OM5110 only)
  Description: Resets the modulator.
  Controller Types: OM5110
  Example: `Reset(obj);`
  Returns: ans = true/false

- **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: All
  Example: `SetActiveLaser(obj, 2);`
  Returns: ans = true

- **bool SetControllerAndLaserByUsageType(classname, enum_laserUsageType);**
  Description: Searches the “running” controllers for a laser matching the requested usage type (usually reference) and selects that controller and laser.
  Controller Types: OM4245, OM4225, OM4006D, OM4106D, OM2210, OM2012
  Valid laserUsageType values are: unused, signalx, signaly, signalxy, reference.
  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: All
  Example: `SetDesiredCavityLock(obj, 1);`
  Returns: ans = true

- **bool SetDesiredChannel(classname, int_desiredChannel, bool_waitUntilComplete);**
  Description: Sets the channel number for the active laser. Returning True = Successful.
  Controller Types: All
  Example: `SetDesiredChannel(obj, 45, true);`
  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: All
  Example: `SetDesiredChannel1(obj, 192.5);`
  Returns: ans = true

- **bool SetDesiredControlMode(classname, enum_modulator, enum_controlmode);** (OM5110 only)
  Description: Sets the control mode of the specified modulator. Returns true if control mode was set, or false if mode was not set.
  Controller Types: OM5110
  Example: `SetDesiredControlMode(obj, modulator.XI, controlmode.automatic);`
  Returns: ans = true/false

- **bool SetDesiredEmittingOff(classname);**
  Description: Sets the Active Laser to Off (not emitting). Returning True = Successful.
  Controller Types: All
  Example: `SetDesiredEmittingOff(obj);`
  Returns: ans = true

- **bool SetDesiredEmittingOn(classname);**
  Description: Sets the Active Laser to emitting. Returning True = Successful.
  Controller Types: All
  Example: `SetDesiredEmittingOn(obj);`
  Returns: ans = true

- **bool SetDesiredFineTuneFrequency(classname, short_desiredFineTuneFrequency, bool_waitUntilComplete);**
  Description: Sets the desired fine tune frequency (in MHz) of the active laser.
  Controller Types: All
  Example: `SetDesiredFineTuneFrequency(obj, 300, true);`
  Returns: ans = true

- **void SetDesiredFrequency(double_desiredFrequency);**
  Description: Sets the channel and fine tune frequency of the selected laser to the specified frequency.
  Controller Types: OM2210, OM2012

- **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: All
  Example: `SetDesiredGridSpacing(obj, 0.05,0);`
  Returns: ans = true

- **bool SetDesiredModulatorMode(classname, enum_modulatorMode);**
  (OM5110 only)
  Description: Sets the specified modulator mode. Returns true if mode was set, or false if mode was not set.
  Controller Types: OM5110
  Example: `SetDesiredModulatorMode (obj, modulatormode.qam)`
  Returns: ans = true/false

- **bool SetDesiredOffset(classname, enum_modulator, double_offset);**
  (OM5110 only)
  Description: Sets the offset for the specified modulator. Returns true if offset was set, or false if offset was not set.

  Valid offset value range depends on the modulator that is being set:
  Q and I modulators range from -2000 to 2000.
  P modulators range from -6000 to 6000.
  Controller Types: OM5110
  Example: `SetDesiredOffset(obj, modulator.XI, 2000);`
  Returns: ans = true/false

- **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: All
  Example: `SetDesiredPower(obj, 13, 0);`
  Returns: ans = true

- **bool SetDesiredPowerOnDefault(classname, string_voltageName, byte_step);** (OM5110 only)
  Description: Sets the default power-on value for the specified voltage. Returns true if value was set, or false if value was not set.
  Controller Types: OM5110
  Example: `SetDesiredPowerOnDefault(obj, 'YI G1', 125);`
  Returns: ans = true/false

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **bool SetDesiredSlope(classname, enum_modulator, enum_slope);** (OM5110 only)
  Description: Sets the slope for the specified modulator to positive or negative. Returns true if slope was set, or false if slope was not set.
  Controller Types: OM5110
  Example: `SetDesiredSlope(obj, modulator.XI, slope.positive);`
  Returns: ans = true/false

- **bool SetDesiredVoltage(classname, enum_modulator, double_voltage);** (OM5110 only)
  Description: Sets the voltage of the specified modulator. Returns true if voltage was set, or false if voltage was not set.
  Controller Types: OM5110
  Example: `SetDesiredVoltage(obj, modulator.XI, 10.2);`
  Returns: ans = true/false

- **bool SetDesireStep(classname, string_voltageName, byte_step);** (OM5110 only)
  Description: Sets the step for the specified voltage. Returns true if value was set, or false if value was not set.
  Controller Types: OM5110
  Example: `SetDesireStep(obj, 'YQ G1', 135);`
  Returns: ans = true/false

  The following are valid voltageName string values:

  'YQ G1', 'YI G1', 'YQ G2', 'YI G2', 'YQ D2', 'YI D2', 'XI D2', 'XQ D2', 'XI G2', 'XQ G2', 'XI G1', 'XQ G1'

- **bool SetManualCalibration(classname);** (OM5110 only)
  Description: This is the same as the SetParams button in the user interface. It sends a new set of null voltages and Vpi voltages to the OM5110 for use by the optical bias controller. Returns true if the calibration was successfully started, and false if the calibration was not started.
  Controller Types: OM5110
  Example: Calibrate(obj, float, float, float, float, float, float, float, float, float, float, float, float);
  Returns: ans = true/false

- **bool SetOpticalPowerAdjustment(classname, double_power);** (OM5110 only)
  Description: Sets the optical power adjustment, in dBm. Returns true if power adjustment value was set, false if the value was not set.
  Controller Types: OM5110
  Example: `SetOpticalPowerAdjustment(obj, 10.1);`
  Returns: ans = true/false

- **bool SetPolarizationIn(classname);** (OM2210 only)
  Description: Puts both polarization filters in. Returning True = Successful.
  Controller Types: OM2210
  Example: `SetPolarizationIn(obj);`
  Returns: ans = true

- **bool SetPolarizationOut(classname);** (OM2210 only)
  Description: Puts both polarization filters out. Returning True = Successful.
  Controller Types: OM2210
  Example: `SetPolarizationOut(obj);`
  Returns: ans = true

- **bool SetReceiverOff(classname);**
  Description: Turns the receiver off in the active controller. Returning True = Successful.

CAUTION. *Make sure that laser power is off 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 push the front-panel **Power** button to power off the instrument.*

Controller Types: OM4245, OM4225, OM4106D, OM4006D
Example: `SetReceiverOff(obj);`
Returns: ans = true

- **bool SetReceiverOn(classname);**
  Description: Turns the receiver on in the active controller. Returning True = Successful.
  Controller Types: OM4245, OM4225, OM4106D, OM4006D
  Example: `SetReceiverOn(obj);`
  Returns: ans = true
- **void TogglePolarization(classname);** (OM2210 only)
  Description: Toggles the polarization state by moving both filters to the opposite position.
  Controller Types: OM2210
  Example: `TogglePolarization(obj);`

LRCP control in MATLAB

The Laser/Receiver Control Panel tab communicates with other programs using port 9000 on the computer running the OMA software.

NOTE. *Make sure that the LRCP tab is open before using this interface.*

To initialize the interface, open a MATLAB application on the desktop before starting the OMA software. Then enter the following commands in the MATLAB desktop command window:

```
url = 'http://localhost:9000/LaserReceiverControlPanel/
  Laser_ReceiverServiceBasic/?wsdl';
```

```
createClassFromWsdl(url);
```

```
obj = Laser_ReceiverServiceBasic;
```

Where:

- ‘url= ...‘ specifies the URL or path to a WSDL application programming interface (API) that defines the web service methods, arguments, and transactions for the Laser/Receiver Control Panel.
- ‘createClassFrom...’ creates the new class based on the specified API and builds a series of M-Files for accessing the Laser/Receiver Control Panel service.
- ‘obj=Laser_Receiver...’ instantiates the object class name and opens a connection to the service.

These commands only need to be run whenever the service interface (available methods) changes.

To display an up-to-date listing of methods for the initialized service, enter the following in the MATLAB desktop command window:

```
methods(obj);
```

Matlab returns a list of available functions for the initialized service. (See page 199, *OMA control in MATLAB*.) The list may also show new functions that have been added (and may not be documented in this manual). To show information on a specific function in the list:

1. Click on a function name in the list to position the cursor in that function.
2. Right-click and select **Help on Selection**. MATLAB displays the function information.

The OMA ATE interface

***NOTE.** Because of an optimization in the OMA application, you must have the related plots displayed in the OMA application before you attempt to read variables for those measurements from MATLAB through this interface, or the values will not be calculated.*

OMA basic service interface

- The basic service for OMA is available on port 9200.
- The basic service uses wsBasicHTTPBinding to be compatible with applications like MATLAB or Labview that only support the simpler binding.
- Exposes a subset of the advanced service commands.
- The basic service is referenced at the following URL:

  http://localhost:9200/Optametra/OM4006/WCFServiceOM4006Basic/

OMA advanced service interface

- The OMA advanced service is available on port 9300.
- The OMA advanced service uses a netTcpBinding (which is not available in MATLAB) and uses events to provide a time-efficient interface.
- The advanced service is referenced at the following URLs:

  net.tcp://localhost:9300/WCFServiceOM4006/
  net.tcp://localhost:9300/WCFServiceOM4006Bulk/

- Wrapper client DLLs (OM4006ATEClient.dll) have been installed into your documents folder under .\TekApplications\ATE Support Files. Copy the appropriate DLL to your application's project folder and add a reference to the DLL to your project.

NOTE. *In previous releases it was required that the user edit their App.Config file for the client applications to supply URL information for accessing the advanced service. This is no longer necessary if the user adds the reference to OM4006ATEClient.DLL to their client ATE Application. The detail of the WCF service is wrapped in the DLL.*

- A new constructor has been added to the OM4006ATEClient class to allow you to change the service's URL and/or port. This is useful for accessing the service remotely or for dealing with port conflicts.

  Syntax: OM4006ATEClient(string url, int port)

  Coding example:

  OM4006ATEClient myOM4006 = new OM4006ATEClient("144.23.45.100", 9300)

NOTE. *The advanced service binding in earlier versions of OMA software was wsHTTPBinding, referenced at URLs http://localhost:9200/Optametra/OM4006/WCFServiceOM4006/ and http://localhost:9200/Optametra/OM4006/WCFServiceOM4006Bulk/. These are not available in the new OMA software. Make sure to use the new binding and URL when you update your ATE code to run with the new OMA software.*

NOTE. *A wrapper DLL (OM4006ATEClient.DLL) is provided to simplify the interface to the OMA advanced service. It is highly recommended that you use this DLL to deal with the handshaking that is associated with working with events, instead of interfacing directly to the service. See Appendix I for how to reference this DLL in your client ATE Application. (See page 200, Building an OMA ATE client in VB.NET.)*

⚠ ***WARNING.*** *WCF services can turn lasers on and off. Verify that no one is physically working with lasers or fibre while running ATE applications. Power off the OM instruments to disable lasers.*

OMA setup for ATE

There is some setup to do in the OMA 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 with the variable "ChOffset":

The standard variables produced by CoreProcessing and any additional declared variables are repopulated each time a single acquisition occurs. After OMA acquires the data record from the oscilloscope, the MATLAB Engine Command window commands execute using that data.

The variables used in the executed commands are only available until the next single acquisition occurs. Make sure that your ATE application saves the global variables to local storage for processing before executing the next single acquisition. This process is synchronized using 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.

The Data/Variable process works as follows:

1. A Single is executed.
2. Record/Block data is retrieved from the oscilloscope 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 ≥ record size.

The example of this implementation is described in *Basic Method for getting MATLAB variable values.* (See page 202.)

OMA basic ATE service function list

The following are the available OMA basic ATE service functions. These functions are available in both the basic and advanced services. These are the only OMA 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.*

NOTE. *MATLAB returns strings, not numeric values. To convert returned string values to numeric values, use the following: VarName = CmndName(obj); x = str2num(VarName). For example: LoFreq = GetLOFreq(obj); x = str2num(LoFreq).*

- **void AutoVerticalScaleScope(classname);**
  Description: Performs an autoscale on the oscilloscope.
  Example: `AutoVerticalScaleScope(obj);`
- **void ConfigureScope(classname);**
  Description: Configures settings on the oscilloscope.
  Example: `ConfigureScope(obj);`
- **void ClearStatisticsMeasurementForChannel(classname, string_channel);**
  Description: Clears all measurement statistics of the specified channel.
  Example: `ClearStatisticsMeasurementForChannel(obj, 'Channel 1');`
- **void Connect(classname, string_Address);**
  Description: Connects OMA to the specified (valid) oscilloscope address.
  Example: `Connect(obj, 'net.tcp://10.49.206.142:8000/TekScopeService');`
- **void DCCalib(classname);**
  Description: Performs a basic DC calibration of the oscilloscope channels.
  Example: `DCCalib(obj);`
- **void Disconnect(classname);**
  Description: Disconnects the application from the connected oscilloscope.
  Example: `Disconnect(obj);`
- **uint GetBlockSize(classname);**
  Description: Returns the current block size from the OMA as an unsigned integer.
  Example: `GetBlockSize(Obj);`
  Returns: ans = 50000

- **double GetLOFreq(classname);**
  Description: Returns a double containing the LO frequency
  Example: `GetLOFreq(obj);`
  Returns: ans = 191.5
- **double GetMeasurementForChannel(classname, string_channel, string_measName);**
  Description: Returns a double containing information for the specified channel and measurement.
  Example: `GetMeasurementForChannel(obj, 'Channel 1', 'Pow Crossing Point');`
  Returns: ans = **25.036**
  The value for channel is either '' for non-multi carrier measurements, or matches the name chosen in the Multi Carrier setup screen if the measurement is for multi-carrier. Values for channel name are case sensitive.
  The value for name is the measurement name. Values for measurement names are case sensitive. Valid measurement names are:

***NOTE.** Semicolons used in the list are not part of the names; they just separate the name values.*

PDL; PER; PMD first order; PMD second order; Pow Crossing Point; Pow Falltime; Pow Overshoot; Pow Risetime; Pow Skew; Pow Undershoot; Sig Freq Offset; Signal Baud Rate; Xconst Bias, Imag; Xconst Bias, Real; Xconst Elongation; Xconst EVM,Average; Xconst IQ Imbalance; Xconst IQ Offset; Xconst Magnitude; Xconst Mask Violations; Xconst Phase Angle; Xconst Quadrature Error; Xconst Symbol Std. Dev.; Xconst Symbols Displayed; X-I Chirp; X-I Crossing Point; X-I Eye Height; X-I Falltime; X-I Overshoot; X-I Q-Factor; X-I Rail 0 Std Dev; X-I Rail 1 Std Dev; X-I Risetime; X-I Skew; X-I Undershoot; X-Q Chirp; X-Q Crossing Point; X-Q Eye Height; X-Q Falltime; X-Q Overshoot; X-Q Q-Factor; X-Q Rail 0 Std Dev; X-Q Rail 1 Std Dev; X-Q Risetime; X-Q Skew; X-Q Undershoot; Yconst Bias, Imag; Yconst Bias, Real; Yconst Elongation; Yconst EVM,Average; Yconst IQ Imbalance; Yconst IQ Offset; Yconst Magnitude; Yconst Mask Violations; Yconst Phase Angle; Yconst Quadrature Error; Yconst Symbol Std. Dev.; Yconst Symbols Displayed; Y-I Chirp; Y-I Crossing Point; Y-I Eye Height; Y-I Falltime; Y-I Overshoot; Y-I Q-Factor; Y-I Rail 0 Std Dev; Y-I Rail 1 Std Dev; Y-I Risetime; Y-I Skew; Y-I Undershoot; Y-Q Chirp; Y-Q Crossing Point; Y-Q Eye Height; Y-Q Falltime; Y-Q Overshoot; Y-Q Q-Factor; Y-Q Rail 0 Std Dev; Y-Q Rail 1 Std Dev; Y-Q Risetime; Y-Q Skew; Y-Q Undershoot

- **double GetMeasurementForChannelCount(classname, string_channel, string_measName);**
  Description: Returns a double of the number of measurements used to calculate the statistics measurement.

  See GetMeasurementForChannel for a list of measurement names.
  Example: `GetMeasurementForChannelCount(obj, 'Channel 1', 'Pow Crossing Point');`
  Returns: ans = 10

- **double GetMeasurementForChannelMax(classname, string_channel, string_measName);**
  Description: Returns a double of the maximum value of the specified channel measurement.

  See GetMeasurementForChannel for a list of measurement names.
  Example: `GetMeasurementForChannelMax(obj, 'Channel 1', 'Pow Crossing Point');`
  Returns: ans =25.036

- **double GetMeasurementForChannelMean(classname, string_channel, string_measName);**
  Description: Returns a double of the Mean of the specified channel measurement.

  See GetMeasurementForChannel for a list of measurement names.
  Example: `GetMeasurementForChannelMean(obj, 'Channel 1', 'Pow Crossing Point');`
  Returns: ans = 25.036

- **double GetMeasurementForChannelMin(classname, string_channel, string_measName);**
  Description: Returns a double of the minimum value of the specified channel measurement.

  See GetMeasurementForChannel for a list of measurement names.
  Example: `GetMeasurementForChannelMin(obj, 'Channel 1', 'Pow Crossing Point');`
  Returns: ans = 35.036

- **double GetMeasurementForChannelStdDev(classname, string_channel, string_measName);**
  Description: Returns a double of the standard deviation of the specified channel measurement.

  See GetMeasurementForChannel for a list of measurement names.
  Example: `GetMeasurementForChannelStdDev(obj, 'Channel 1', 'Pow Crossing Point');`
  Returns: ans = 15

- **uint GetNumberOfAcquisitions(classname);**
  Description: Returns the number of acquistion records taken by the OMA as an unsigned integer.
  Example: `GetNumberOfAcquisitions(obj);`
  Returns: ans = 3578

- **uint GetNumberOfProcessedAcquisitions(classname);**
  Description: Returns the number of processed acquistion records taken by the OMA as an unsigned integer.
  Example: `GetNumberOfProcessedAcquisitions(Obj);`
  Returns: ans = 1357

- **uint GetRecordLength(classname);**
  Description: Returns the current record length from the OMA as an unsigned integer.
  Example: `GetRecordLength(Obj);`
  Returns: ans = 1000

- **string GetScopeAddress(classname);**
  Description: Returns a string containing the oscilloscope IP address
  Example: `GetScopeAddress(Obj);`
  Returns: ans = '192.168.117.0'

- **string GetScopeIDN(classname);**
  Description: Returns a string containing the oscilloscope identifier
  Example: `GetScopeIDN(obj);`
  Returns: ans = "TEKTRONIX,DPO73304D,Q200004,CF:91.1CT FV:7.2.0 Build 4", which is a comma-delimited string containing the instrument model number and firmware version.

- **bool IsAcquiring(classname);**
  Description: Returns a boolean flag of the oscilloscope interface state; True = Data is being acquired from the oscilloscope
  Example: `IsAcquiring(obj);`
  Returns: ans = true

- **bool IsCancelling(classname);**
  Description: Returns a boolean flag of the oscilloscope interface state; True = Acquisition is being canceled
  Example: `IsCancelling(obj);`
  Returns: ans = true

- **bool IsProcessing(classname);**
  Description: Returns a boolean flag of the oscilloscope data state; True = Data is being processed
  Example: `IsProcessing(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 IsRunning(classname);**
  Description: Returns a boolean flag of the oscilloscope interface state; True = Acquisition is running.
  Example: `IsRunning(obj);`
  Returns: ans = true

- **bool IsScopeConnected(classname);**
  Description: Returns a boolean flag of the oscilloscope interface state; True = Application is connected to an oscilloscope.
  Example: `IsScopeConnected(obj);`
  Returns: ans = true

- **void SetBlockSize(classname, uint_newBlockSize);**
  Description: Sets the desired block size in the OMA as an unsigned integer for the next acquisition.
  Example: `SetBlockSize(obj, 2000);`

- **void SetLOFreq(classname, double_loFreq);**
  Description: Sets the desired LO Frequency as a double in the OMA for the next acquisition.

NOTE. *This value may be overridden if the OMA retrieves the value from the LRCP.*

  Example: `SetLOFreq(obj, 193.5);`

- **void SetMultiCarrierChannelList(classname, string_name);**
  Description: Sets the active carrier channel list.
  Example: `SetMultiCarrierChannelList(obj, 'Carrier List Name');`

- **void SetRecordLength(classname, uint_newRecordLength);**
  Description: Sets the desired record length in the OMA as an unsigned integer for the next acquisition.
  Example: `SetRecordLength(obj, 10000);`

- **void Single(classname, int_timeoutInMilliseconds);**
  Description: Performs a single acquisition within the timeout period.
  Example: `Single(obj,3000);`

- **void StartMulticarrierScanSingle(classname);**
  Description: Starts a single multi carrier scan.
  Example: `StartMulticarrierScanSingle(Obj);`
- **void StopMulticarrierScanSingle(classname);**
  Description: Stops a single multi carrier scan.
  Example: `StopMulticarrierScanSingle(Obj);`

OMA advanced ATE service function list

See *Analysis parameters* for more information on the OMA functionality of these functions as exposed by the Client DLL. (See page 52.)

See *Building an OMA ATE client in VB.NET* for examples of using these functions in an ATE client application. (See page 200.)

The following functions are available only in the advanced service interface (all OMA basic service functions are available in advanced services):

- **AnalysisParameters GetAnalysisParameters()**
  Description: Returns an analysis parameters object containing the OMA variables.
- **void BlockProcessed(BlockProcessedEventArgs eventArgs)**
  Description: Event Trigger that is executed when the OMA application completes processing of a block. Attach your event handler to this delegate and it executes when the event occurs.
- **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 OMA MATLAB engine. See MATLAB section for usage.*

- **bool[] GetArrayOfBoolean(string_vname)**
  Description: Returns the value of the passed variable as an array of boolean.
- **double[] GetArrayOfComplexImaginary(string_vname)**
  Description: Returns an array of imaginary double values of the specified array of complex numbers.

- **double[] GetArrayOfDoubles(string_vname)**
  Description: Returns the value of the passed variable as an array of doubles.
- **bool GetBoolean(string_vname)**
  Description: Returns the value of the passed variable as boolean.
- **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 GetDouble(string_vname)**
  Description: Returns the value of the passed variable as a double.
- **string GetScopeState()**
  Description: Returns a string containing a text description of the oscilloscope's state. Valid states are: Unknown, Disconnected, Connected, Acquiring laserusagetype exposelaserusage() Exposes the laser usage type to the OMA interface.
- **void LoadState(string_stateName, bool_loadSoftware, bool_loadHardware)**
  Description: Loads the current state of the system which includes software and/or hardware as specified by the flags. The state information is loaded from an XML file located in folder documents\tekapplications\OUI.
- **void RecordProcessed(RecordProcessedEventArgs eventArgs)**
  Description: Event Trigger that is executed when the OMA application completes processing of a record. Attach your event handler to this delegate and it executes when the event occurs.
- **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 RegisterForBlockUpdate(MATLABVariable variable)**
  Description: Registers a MATLAB variable of interest for updating by MATLAB at the end of block processing.
- **void RegisterForBlockUpdate(string_vname)**
  Description: Registers a variable for the block update by variable name.
- **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 RegisterForRecordUpdate(MATLABVariable variable)**
  Description: Registers a MATLAB variable of interest for updating by MATLAB at the end of record processing.
- **void RegisterForRecordUpdate(string_vname)**
  Description: Registers a variable for the record update by variable name.

- **void ScopeReceiverDeskew()**
  Description: Starts the oscilloscope/receiver deskew operation.
- **void SaveState(string_stateName, bool_saveSoftware, bool_saveHardware)**
  Description: Saves the current state of the system which includes software and/or hardware as specified by the flags. The state information is saved as the file <statename>.xml, in the folder documents\tekapplications\OUI.
- **void ScopeReceiverDeskewFinished(ScopeReceiverDeskewEventArgs eventArgs)**
  Description: This event occurs when the oscilloscope receiver deskew finishes with a success or failure. EventArgs contains the status and resulting channel delays.
- **bool ScreenShot(string_filename)**
  Description: Takes a snapshot of the OMA display (like doing a PrintScreen) and saves the image file to the specified file.
- **string SendMATLABCommand(string_MATLABCommand)**
  Description: This command is used to execute a MATLAB statement.

NOTE. *This command can only be used when the normal CoreProcessing is disabled; otherwise there is contention with the OMA MATLAB engine. See MATLAB section for usage.*

- **void SetAnalysisParameters(AnalysisParameters analysisParameters)**
  Description: Sets the analysis parameters to new values.
- **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.

OMA control in MATLAB

The OMA basic service communicates with other programs using port 9200 on the computer running the OMA software.

To initialize the OMA service interface, open a MATLAB application on the desktop before starting the OMA software. Start the OMA software. Then enter the following commands in the MATLAB desktop command window:

```
url = 'http://localhost:9200/Optametra/OM4006/
 WCFServiceOM4006Basic/?wsdl';
```

```
createClassFromWsdl(url);
```

```
obj = WCFServiceOM4006Basic;
```

Where:

- ‘url= ...‘specifies the URL or path to a WSDL application programming interface (API) that defines the web service methods, arguments, and transactions for the OMA software.
- ‘createClassFrom...’ creates the new class based on the specified API and builds a series of M-Files for accessing the OMA service.
- ‘obj=WCFService...’ instantiates the object class name and opens a connection to the service.

These commands only need to be run whenever the service interface (available methods) changes.

To display an up-to-date listing of functions for the initialized service, enter the following in the MATLAB desktop command window:

```
methods(obj);
```

Matlab returns a list of available functions for the initialized service. (See page 199, *OMA control in MATLAB*.) The list may also show new functions that have been added (and may not be documented in this manual). To show information on a specific function in the list:

1. Click on a function name in the list to position the cursor in that function.
2. Right-click and select **Help on Selection**. MATLAB displays the function information.

Building an OMA ATE client in VB.NET

The following document explains how to build an OMA 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 two ways that a ATE client application can interface to the WCF Services:

- Create a service reference in a VB Project that talks directly to the WCF Service.
- Add a reference to the OM4006ATEClient .NET assembly.

The preferred method is to use the OM4006ATEClient .NET assembly and the following sections explain how to implement that method.

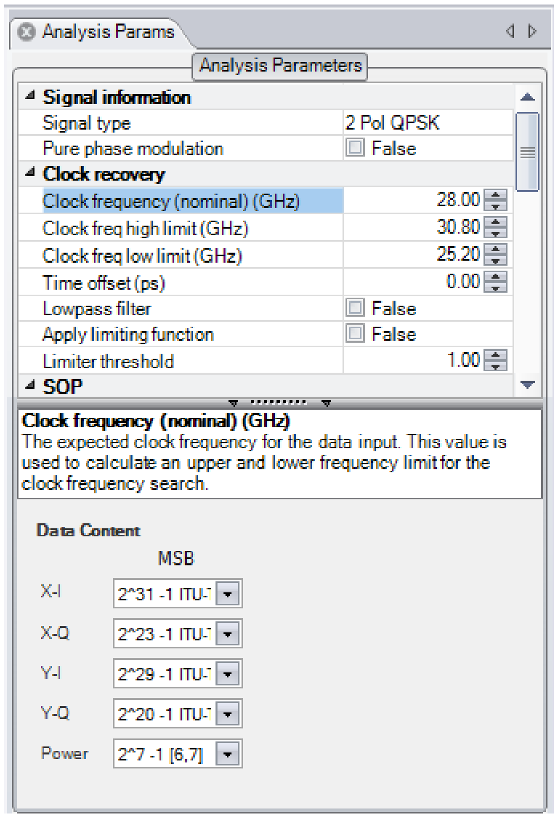
WCF service background

The OMA application exposes functionality using several WCF services. The OM4006ATEClient assembly uses two of those WCF services 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 doesn’t need to worry about the details of the services.

The following user control (ucAnalysisParameters) is provided in OM4006ATEClient for use in ATE client applications.



Signal information	
Signal type	2 Pol QPSK
Pure phase modulation	☐ False
Clock recovery	
Clock frequency (nominal) (GHz)	28.00
Clock freq high limit (GHz)	30.80
Clock freq low limit (GHz)	25.20
Time offset (ps)	0.00
Lowpass filter	☐ False
Apply limiting function	☐ False
Limiter threshold	1.00
SOP	

Clock frequency (nominal) (GHz)
The expected clock frequency for the data input. This value is used to calculate an upper and lower frequency limit for the clock frequency search.

Data Content

	MSB
X-I	2^31 -1 ITU-T
X-Q	2^23 -1 ITU-T
Y-I	2^29 -1 ITU-T
Y-Q	2^20 -1 ITU-T
Power	2^7 -1 [6,7]

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 can also connect or disconnect to/from an oscilloscope 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 OMA 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 to view the file tree.

Basic method for getting MATLAB variable values

The following is an example code block that references the OMA 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.RegisterForBlockUpate(“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 OMA 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.RegisterForBlockUpate(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 is 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 applicatoin (OUI4006.exe) or on a remote machine. You can access a service remotely using the OM4006ATEclient constructor that includes the URL and Port.

Appendix J: MATLAB CoreProcessing function reference

AlignTribs

[Rot,PattXReOut,PattXImOut,PattYReOut,PattYImOut,BoundValsOut,AlertsOut] = AlignTribs(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

AlignTribs 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. AlignTribs 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. AlignTribs 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, that is 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 part of the specified data pattern to be used for synchronization, elements .Values(1:SyncFrameEnd). Typically 100 elements is enough 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.

AlignTribs 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))

AlignTribs 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*pi*Theta.CentFreq*(0:number of symbols-1)+Theta.Values

ClockRetime

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

- V – 1x4 array of structures, each containing waveform (voltage values) from oscilloscope
- ChDelay – 4-element vector of time delays between oscilloscope 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 oscilloscope 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 oscilloscope waveforms are adjusted for the known relative delays ChDelay between the four oscilloscope 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/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 – 1xN or 2xN 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 Reference [1] for BPSK modulated signals and [2] for QPSK. (See page 213.)

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/Delay. For a real implementation of a delay discriminator to work well the signal optical frequency should be a multiple of 1/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

[1]. 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.

[2]. 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 oscilloscope

ChDelay – 4-element vector of time delays between oscilloscope 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 oscilloscope 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 oscilloscope 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. Teh application sweeps the frequency within the given window, FreqWindow, to search for a spike at the clock frequency.

EstimateClock 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 = abs(X(1,:)).^2+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 = sqrt(abs(X(1,:)).^2+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(abs(X(1,:)).^2+abs(X(2,:)).^2, 0.2*mean(abs(X(1,:)).^2+abs(X(2,:)).^2))

EstimateClock 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.

EstimateClock 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 ∞ to +∞:
  - .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: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.

EstimatePhase 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:

Theta.CentFreq + (Theta.Values(end)-Theta.Values(1))/2/pi/number of symbols/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, 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.

EstimateSOP 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 start 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,PattReSyn,PattImSyn,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
- PattReSyn – binary (0 or 1) indicating if the in-phase signal is synchronized
- PattImSyn – 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 on 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 OMA 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. *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 = 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 = 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 (See page 223, *References (QDecTh)*.) 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 (QDecTh)

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 – oscilloscope 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.

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

Appendix K: MATLAB variables used by CoreProcessing

MATLAB input variables

- Vblock – voltage vs. time on four oscilloscope channels
- PattXRe, PattXIm, PattYRe, PattYIm – define data patterns on tributaries, used to identify bit errors
- 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
  - 18 - Single polarization 4 Level
  - 19 - Dual polarization 4 Level
  - 19 - Single polarization PAM 4
  - 21 - Dual polarization PAM 4

MATLAB calculated variables

The variables that are calculated by CoreProcessing include:

NOTE. *Variables of the same name that have UI appended are created after post processing on each full record of data. These values are not available until after processing is completed and so can not be accessed in the OMA software Matlab Engine Window.*

- zXSym, zYSym – complex electric field at symbol center times on X and Y polarizations; displayed as blue dots on constellation diagram in OMA
- 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 L: 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 a value 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 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. 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 for unique file name

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 trigging 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

%
```

Index