

MultiScope MATLAB Time Synchronization Software

Notices

© Keysight Technologies, Inc. 2014

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies, Inc. as governed by United States and international copyright laws.

Manual Part Number

Version 05.46.0000

Edition

September 2014

Available in electronic format only

Published by:
Keysight Technologies, Inc.
1900 Garden of the Gods Road
Colorado Springs, CO 80907 USA

Warranty

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

Technology Licenses

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

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014 (June 1995), or as a “commercial item” as defined in FAR

2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Keysight Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

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

WARNING

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

MultiScope MATLAB Time Synchronization Software—At a Glance

The MultiScope MATLAB time synchronization software is a collection of MATLAB scripts and functions that run on a host (controller) computer to capture time-synchronized acquisitions from MultiScope systems (up to 10 oscilloscopes) with very low oscilloscope-to-oscilloscope jitter and drift. Because MATLAB scripts and functions are used, a moderate understanding and proficiency using MATLAB is required.

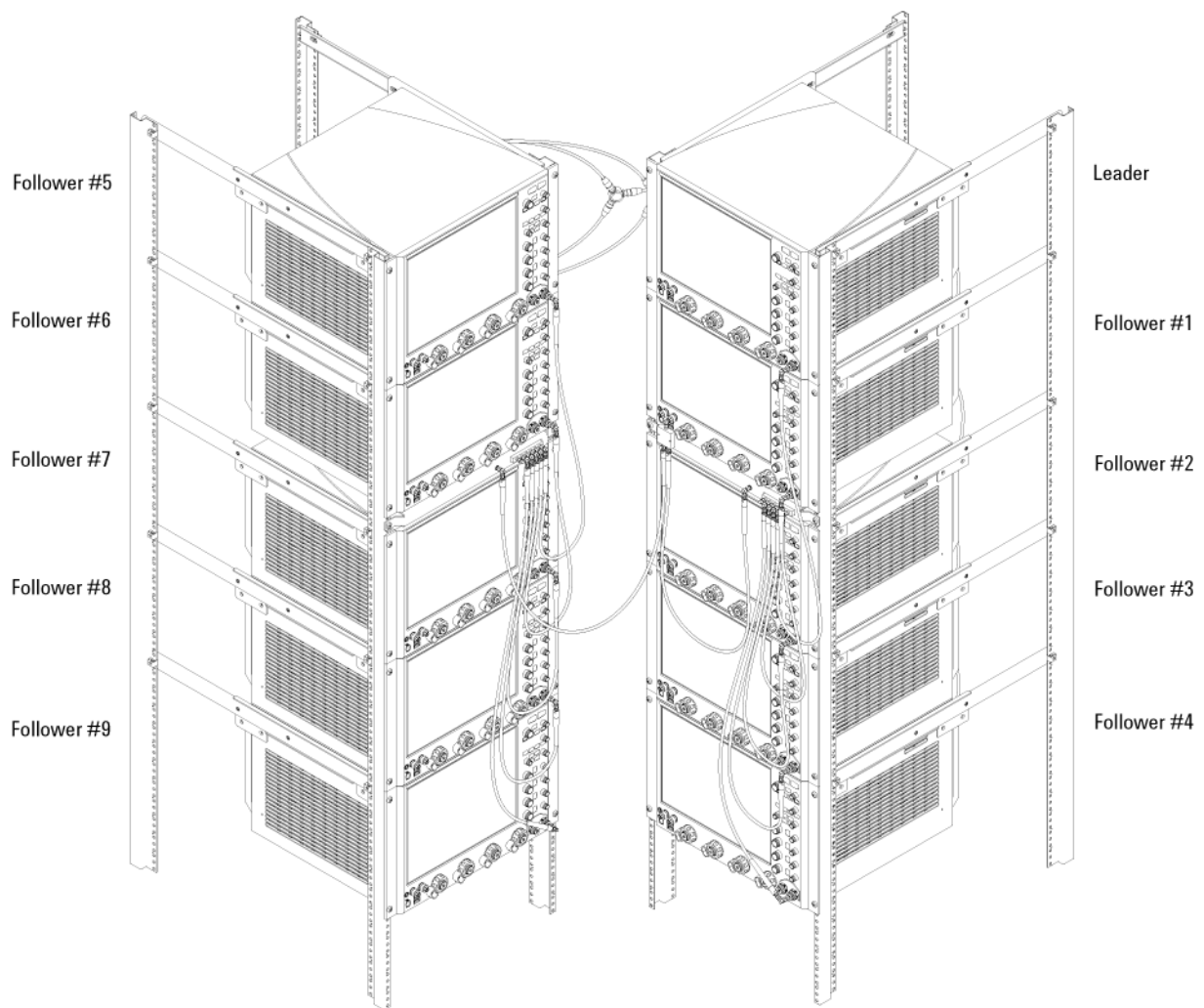


Figure 1 40-Channel, 33 GHz MultiScope System

The MultiScope MATLAB software is designed to help you configure the Infiniium real-time oscilloscopes, perform acquisitions, and then save the digitized time-synchronized waveform data to disk. From there, you can analyze the waveform data using your own MATLAB functions or other analysis tools.

You can customize the provided MATLAB scripts and functions however you like. Small customizations are necessary for your particular MultiScope system configuration, but you can also make large customizations like, for example, incorporating Keysight's MultiScope functions into your own proprietary software. While most of the provided files are not intended to be modified, you can still do so with sufficient care.

For information on setting up and connecting oscilloscopes into a MultiScope system, see the *Keysight MultiScope Hardware Configuration Guide* ("www.keysight.com/find/MultiScope-hw-config").

In This Guide This guide shows you how to use the example scripts and functions included with the N8822A MATLAB time synchronization software:

- **Chapter 1**, "Setting Up," starting on page 9
- **Chapter 2**, "Getting Started," starting on page 23
- **Chapter 3**, "MultiScope MATLAB Software in More Detail," starting on page 53
- **Chapter 4**, "Customizing Functions/Scripts," starting on page 63
- **Chapter 5**, "Troubleshooting," starting on page 67
- **Chapter 6**, "Theory of Operation," starting on page 71
- **Chapter 7**, "Infiniium 90000 X-Series and 90000L Series Oscilloscopes," starting on page 93

Any or all of the provided MATLAB functions can be compiled and called from C++. For more information, refer to the MATLAB documentation.

Contents

MultiScope MATLAB Time Synchronization Software—At a Glance / 3

1 Setting Up

Setting Up Oscilloscopes in the MultiScope System / 10

Selecting a Host Computer / 11

Installing Prerequisite Software on the Host Computer / 12

MATLAB software / 12

Installing the MultiScope MATLAB Software / 13

Keysight IO Libraries Suite / 13

Making Remote Connections to the Oscilloscopes / 14

Using the LAN interface / 14

Using the USB 3.0 interface / 14

Verifying oscilloscope remote connections / 15

2 Getting Started

Viewing the Oscilloscope User Interfaces / 25

MATLAB Basics / 26

Starting the MultiScope MATLAB Environment / 27

Keysight_startup script / 28

User_startup script / 28

Files in the Examples Folder / 29

Starting the MultiScope MATLAB Graphical User Interface (GUI) / 30

Specifying the Oscilloscopes in Your MultiScope System / 32

In the GUI Frames Tab / 32

In the SysConfig script / 33

Finding VISA Addresses / 33

Initialize the Oscilloscopes (& Calibrate Reference Clock Skew) / 35

InitNewConfig script / 35

InitPowerOn script / 36

Setting Up the Oscilloscopes / 37	
Make System-Common Settings on the Leader Oscilloscope / 37	
Make Channel-Specific Settings on All Oscilloscopes / 38	
Ways to Make MultiScope Settings / 38	
SaveSetup script / 40	
LoadSetup script / 40	
Deskewing Measured Waveforms / 41	
DeskewChans script / 43	
DeskewFrames script / 43	
DeskewSignals script / 43	
ZeroskewSignals script / 44	
Digitizing Input Signals / 45	
DigitizeWfms script / 45	
SetScopeAndDigitize script / 46	
LoadAndPlotWfms script / 46	
Measuring Jitter (Optional) / 47	
MeasJitter script / 47	
LoadAndPlotJitter script / 48	
Measuring Acquisition Times (Optional) / 49	
MeasAcqTime script / 49	
LoadAndPlotAcqTime script / 50	
Using Advanced GUI Settings / 51	
Saving and Opening GUI Projects / 52	

3 MultiScope MATLAB Software in More Detail

Setting MultiScope Configuration Parameters / 54	
Setup File Parameters / 54	
System-Common Parameters / 55	
Channel-Specific Parameters / 55	
Operation Parameters / 56	
Waveform Plot Parameters / 57	
Other Parameters and Defaults / 58	
Organization of MultiScope MATLAB Functions / 59	
AcquireMultiscope function / 59	
ConfigMultiscope function / 59	
Multiscope function / 60	

4 Customizing Functions/Scripts

- Software Coding Conventions / 64
- Parameter Passing / 65
- Save Changes in the User Folder / 66

5 Troubleshooting

- Errors Opening a Device Interface That Is Already Open / 68
- Remote Command Errors / 69
- "Data out of range" Errors / 70

6 Theory of Operation

- Oscilloscope Architecture / 72
- Hardware Configuration / 73
- Synchronizing Multiple Oscilloscopes / 74
- Trigger Out to Trigger In Synchronization / 76
- Frame Lag / 77
- Skew / 78
- External Skew / 80
- Internal Skew / 81
- Sample Alignment / 83
- Channel Delay Skew / 84
- Timebase Delay Reference Position / 85
- Jitter Correction / 86
 - Leader.XOrg / 87
 - Follower.XOrg / 87
 - LeaderXOrgDelay / 87
 - FollowerXOrgDelay / 87
 - SliceDelay / 88
 - TotalLeaderSkew, TotalFollowerSkew / 88
 - RefClkSkew / 88
 - JitterFreeSkew / 89
- Drift Correction / 90
- Post-Acquisition Skew / 91
- Setup/Recall / 92

7 Infiniium 90000 X-Series and 90000L Series Oscilloscopes

90000 X-Series and 90000L Series Oscilloscope Requirements / 94

Connections for Synchronization / 95

Connections for Reference Clock Skew Calibration / Drift Correction / 96

Sample Rate Selection and Available Channels / 97

Index

1 Setting Up

Setting Up Oscilloscopes in the MultiScope System / 10

Selecting a Host Computer / 11

Installing Prerequisite Software on the Host Computer / 12

Making Remote Connections to the Oscilloscopes / 14

Installing the MultiScope MATLAB Software / 13

Setting Up Oscilloscopes in the MultiScope System

To set up the physical configuration of oscilloscopes in a MultiScope system, see the *Keysight MultiScope Hardware Configuration Guide* ("www.keysight.com/find/MultiScope-hw-config").

Oscilloscope hardware requirements are described in the configuration guide.

Required Software Version and License

Each Infiniium oscilloscope in a MultiScope system must have software version 4.60.0013 or later.

The MultiScope MATLAB time synchronization software also requires each oscilloscope in a MultiScope system to have the N8822A license. When this license is installed, you see **MultiScope Control** listed as an installed option in the About Infiniium dialog box (**Help > About Infiniium...**).

The N8822A license enables (undocumented) remote commands in the oscilloscope that are used by the MultiScope MATLAB time synchronization software.

Selecting a Host Computer

In order to run the MultiScope application, you need:

- An oscilloscope designated as the Leader.
- A computer that runs the MultiScope MATLAB software. This is the host computer.

The host computer can be a separate PC or any of the Infiniium oscilloscopes in the MultiScope system.

The MultiScope MATLAB software does not need to run on the Leader oscilloscope, and in fact, it is recommended to put the software on a separate PC (preferred), or on one of the Follower oscilloscopes.

When using one of the Infiniium oscilloscopes as the host computer, it is best to choose one of the Follower oscilloscopes. This is because you frequently need work with both the Leader's oscilloscope application window and the MultiScope MATLAB window simultaneously, and they would have to share the same display if they are both running on the Leader. You don't typically work with the Follower's oscilloscope application windows as often as you do the Leader's.

- Host Computer Requirements** The host computer requirements are whatever is required by the MATLAB software; however, for best performance, Keysight recommends:
- A 64-bit operating system with at least 8 GBytes of RAM.
 - The 64-bit version of MATLAB.

Installing Prerequisite Software on the Host Computer

Along with the MultiScope MATLAB time synchronization software, the host computer must also have MATLAB software.

MATLAB software

Before you can install the MultiScope MATLAB time synchronization software on the host computer, a fully-licensed version of MATLAB must already be installed.

For best performance, the 64-bit version of MATLAB is recommended (along with a 64-bit operating system and at least 8 GBytes of RAM).

MultiScope MATLAB software will probably work with any version of MATLAB from 2008 or later, but it has been tested extensively and known to work with 2011a.

MultiScope MATLAB software also requires the instrument control toolbox.

Installing the MultiScope MATLAB Software

The MultiScope installer package creates and installs all necessary files into a program folder on the host controller. The default location of this program folder on a Windows 7 PC is:

`\Program Files (x86)\Keysight\Infiniium\Apps\MultiScope`

Included in this folder is a start-up script file called `Keysight_startup.m` (see **"Starting the MultiScope MATLAB Environment"** on page 27).

Keysight IO Libraries Suite

The MultiScope MATLAB time synchronization software communicates with oscilloscopes using their remote interfaces. Therefore, the Keysight IO Libraries Suite software must also be installed on the host computer.

The MultiScope installer package will install the IO Libraries Suite software if it is needed.

Making Remote Connections to the Oscilloscopes

Oscilloscopes in a MultiScope system have two interfaces that can be used for remote communication: 1G-BaseT LAN or USB 3.0.

Characteristic	LAN Interface	USB 3.0 Interface
Speed	Good	Good
VISA address format	Use format "TCPIP0::<ip_address>::inst0::INSTR" or get VISA address from Keysight Connection Expert.	Get VISA address from Keysight Connection Expert (see "In the GUI Frames Tab" on page 32).

Using the LAN interface

NOTE

HiSLIP is not supported for the LAN interface.

To connect from the host computer to an oscilloscope using its local area network (LAN) interface:

- 1 If the host controller computer is not already connected to the LAN, do that first.
- 2 Contact your network administrator about adding the oscilloscope to the network.

Setting up an Infiniium oscilloscope on a network is the same as setting up any other computer with the Windows operating system.

- 3 Connect the oscilloscope to the LAN by inserting a LAN cable into the LAN port on the oscilloscope.

You can increase the LAN interface performance by placing the oscilloscopes and the host controller on an isolated network (by using a switch).

Using the USB 3.0 interface

To connect from the host computer to the oscilloscopes using their USB 3.0 interfaces:

- 1 Connect a USB 3.0 cable from one of the controller PC's USB 3.0 host ports to the device port on a USB 3.0 hub.
- 2 Connect USB 3.0 cables from the hub's USB 3.0 host ports to the oscilloscopes' USB 3.0 device ports (on the back I/O connector panel).

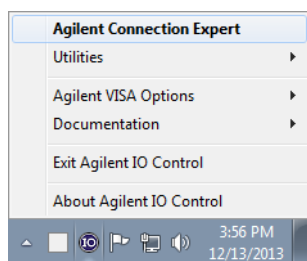
A USB 3.0 hub is necessary only if there are not enough USB ports on the host computer to connect directly to all of the oscilloscopes in the MultiScope system.

Verifying oscilloscope remote connections

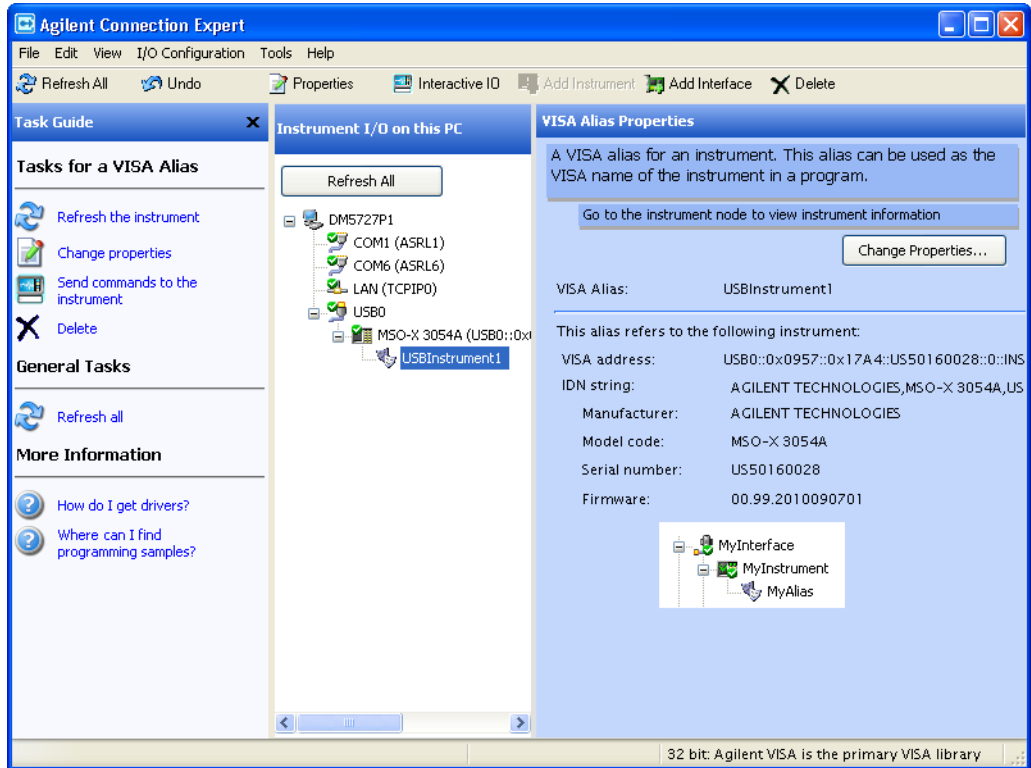
NOTE

Make sure the Keysight Infiniium software is running on the oscilloscopes. It must be running before you can add an instrument or verify its connection using the Keysight Connection Expert.

- 1 On the controller PC, click the Keysight IO Control icon in the taskbar and choose **Keysight Connection Expert** from the pop-up menu.



- 2 In the Keysight Connection Expert application, add instruments if necessary.

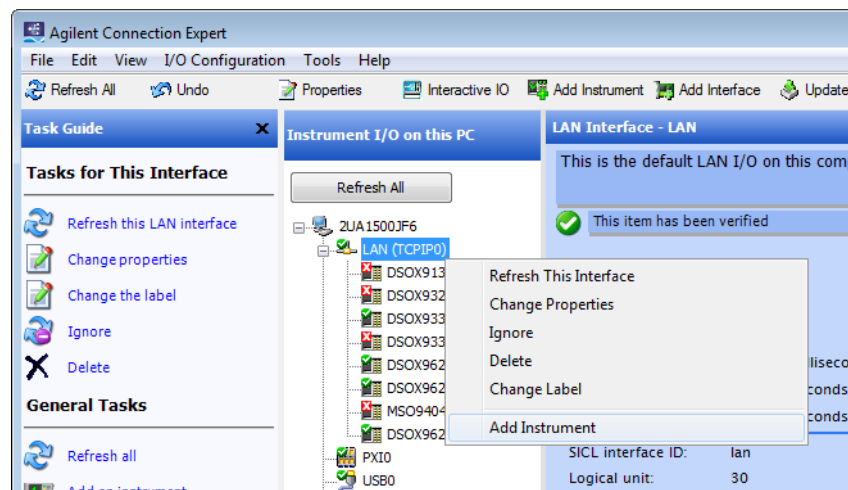


Instruments connected to the controller's USB interfaces should automatically appear. (You can click **Refresh All** to update the list of instruments on these interfaces.)

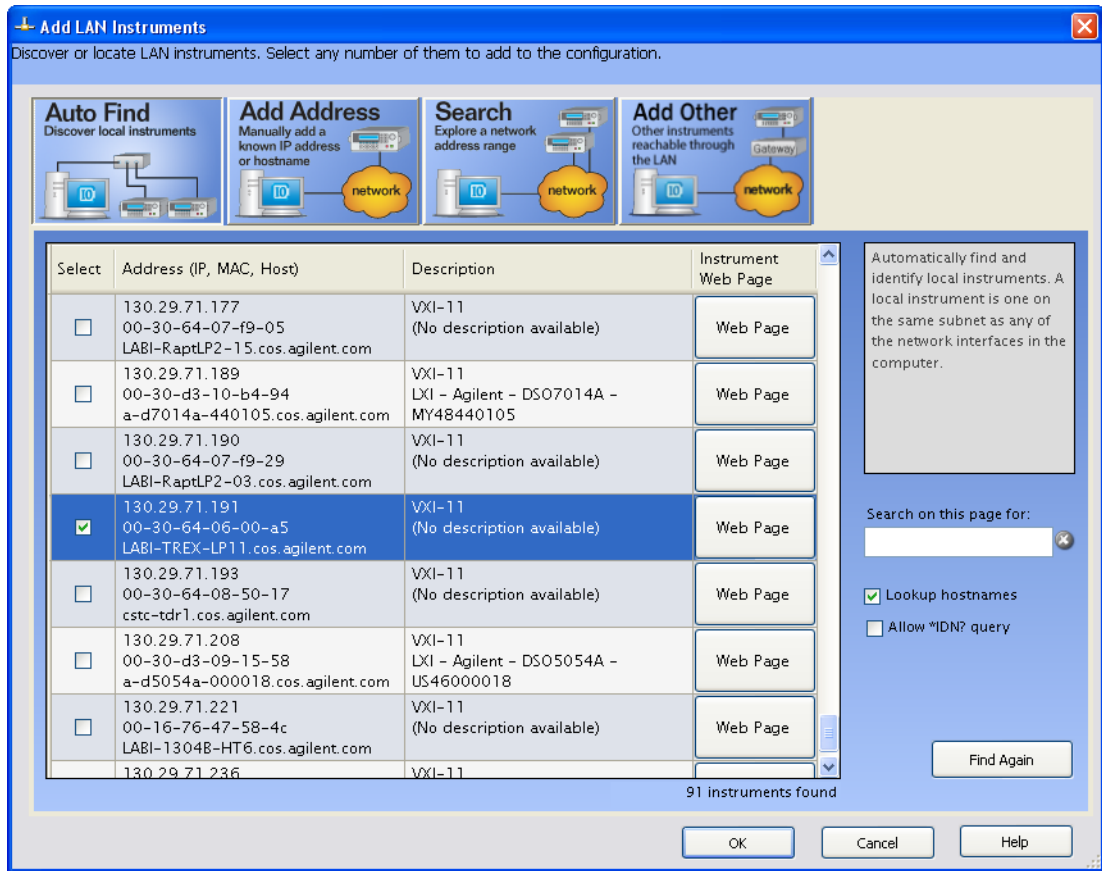
If your oscilloscopes are not listed under **Instrument I/O on this PC**, they will need to be added.

You must manually add instruments on LAN interfaces:

- a Right-click on the LAN interface and choose **Add Instrument** from the pop-up menu.

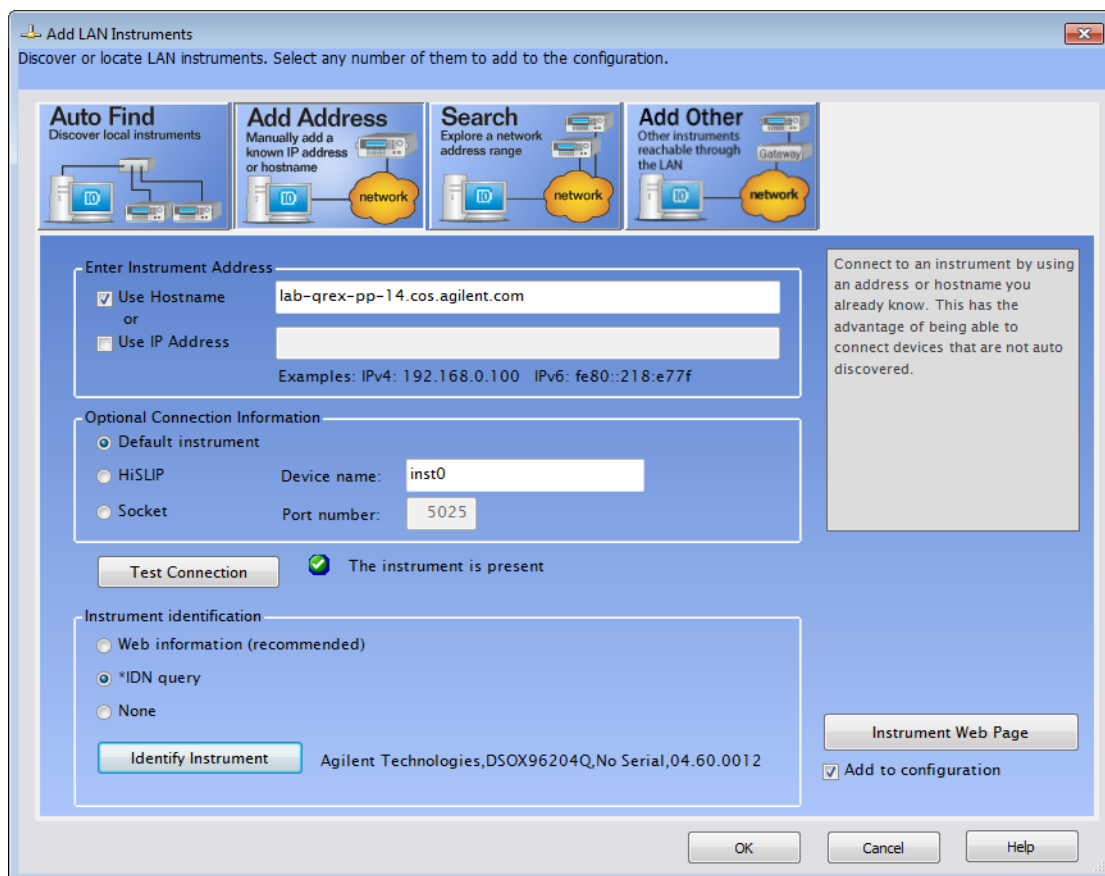


- b If the oscilloscope is on the same subnet, select it, and click **OK**.



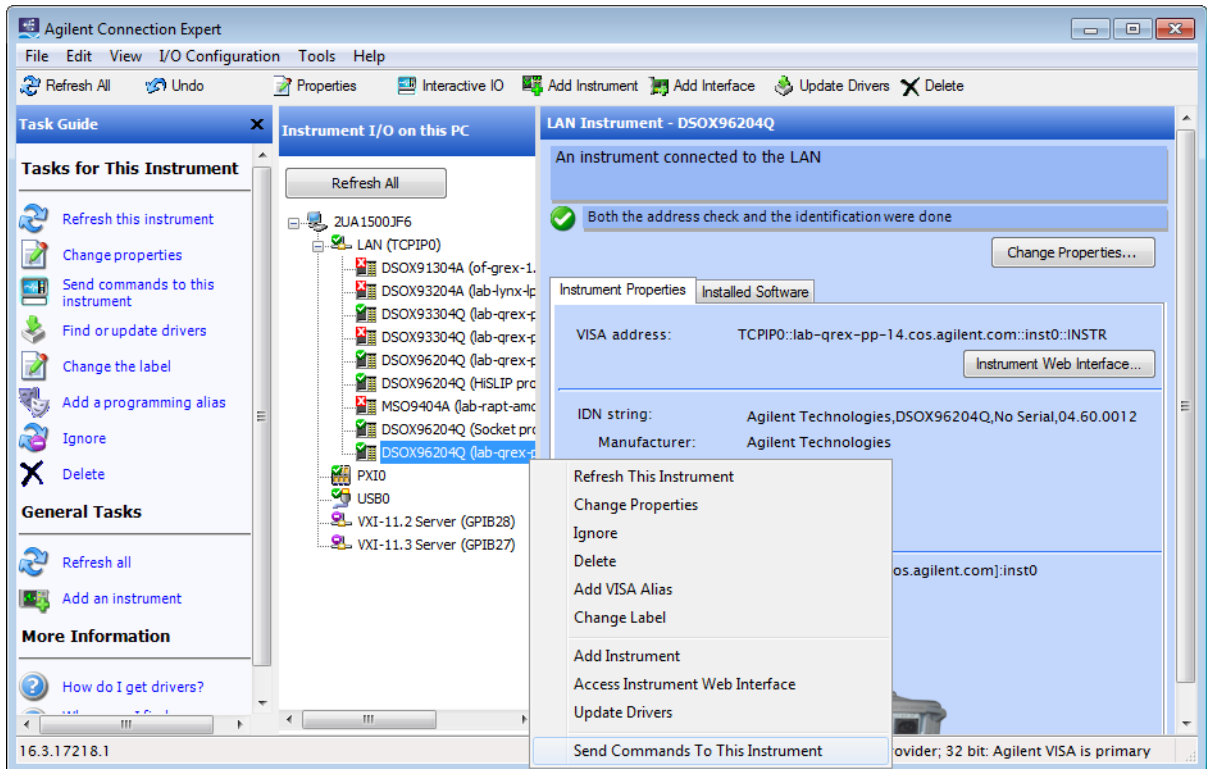
Otherwise, if the instrument is not on the same subnet, click **Add Address**.

- i In the next dialog box, select either **Hostname** or **IP address**, and enter the oscilloscope's hostname or IP address.
- ii Click **Test Connection**.

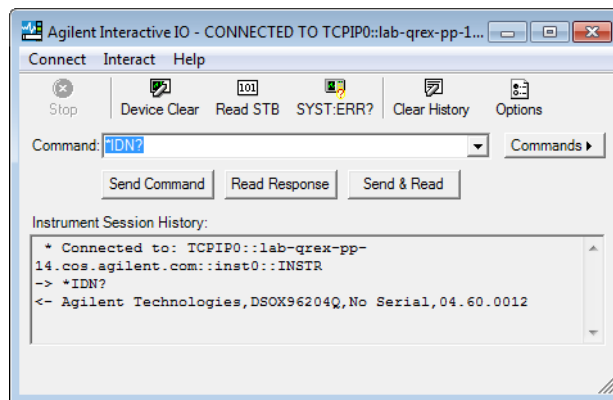


- iii If the instrument is successfully opened, click **OK** to close the dialog box. If the instrument is not opened successfully, go back and verify the LAN connections and the oscilloscope setup.

- 3 Test some commands on the instrument:
 - a Right-click on the instrument and choose **Send Commands To This Instrument** from the pop-up menu.



- b In the Keysight Interactive IO application, enter commands in the **Command** field and press **Send Command**, **Read Response**, or **Send & Read**.



- c Choose **Connect > Exit** from the menu to exit the Keysight Interactive IO application.

- 4 In the Keysight Connection Expert application, choose **File > Exit** from the menu to exit the application.

1 Setting Up

2 Getting Started

Viewing the Oscilloscope User Interfaces / 25
MATLAB Basics / 26
Starting the MultiScope MATLAB Environment / 27
Files in the Examples Folder / 29
Starting the MultiScope MATLAB Graphical User Interface (GUI) / 30
Specifying the Oscilloscopes in Your MultiScope System / 32
Setting Up the Oscilloscopes / 37
Deskewing Measured Waveforms / 41
Digitizing Input Signals / 45
Measuring Jitter (Optional) / 47
Measuring Acquisition Times (Optional) / 49
Using Advanced GUI Settings / 51
Saving and Opening GUI Projects / 52

The basic steps you take when first using the MultiScope MATLAB software are:

- 1 Set up the MultiScope system (see **Chapter 1**, "Setting Up," starting on page 9).
- 2 Power on all oscilloscopes in the system and start the Infiniium oscilloscope application on each oscilloscope.

(To view each oscilloscope's user interface on the host computer, see "**Viewing the Oscilloscope User Interfaces**" on page 25.)

- 3 On the host computer, start the MultiScope MATLAB environment.

This will navigate to the appropriate working directory and run a startup script to set the search paths (see "**Starting the MultiScope MATLAB Environment**" on page 27).

- 4 Specify:
 - a The number of oscilloscope frames in the system.
 - b The oscilloscopes' remote interface (VISA) addresses.

(See "**Specifying the Oscilloscopes in Your MultiScope System**" on page 32.)

- 5 Configure the system and create the calibration file, MultiscopeCalData.mat (see **"Initialize the Oscilloscopes (& Calibrate Reference Clock Skew)"** on page 35).
- 6 Connect the signals to be measured to the oscilloscope input channels.
- 7 Configure the oscilloscopes manually or load their setups from a file (see **"Setting Up the Oscilloscopes"** on page 37).
- 8 Deskew the input signals by running one of the deskew scripts (see **"Deskewing Measured Waveforms"** on page 41).
- 9 Digitize the input signals (see **"Digitizing Input Signals"** on page 45).

The rest of this chapter describes these steps in more detail.

Viewing the Oscilloscope User Interfaces

It can be helpful to view each oscilloscope's user interface on the host computer while running the provided MATLAB scripts and functions. To do this, you can use:

- Remote Desktop Connection software that comes with the Windows operating system.
- UltraVNC Viewer client software that can be downloaded and installed from the internet for free.

The VNC server software comes pre-installed on Infiniium oscilloscopes. Refer to the oscilloscope's online help for instructions on configuring the VNC server.

MATLAB Basics

If you are new to the MATLAB software, here are some basic things that are good to know.

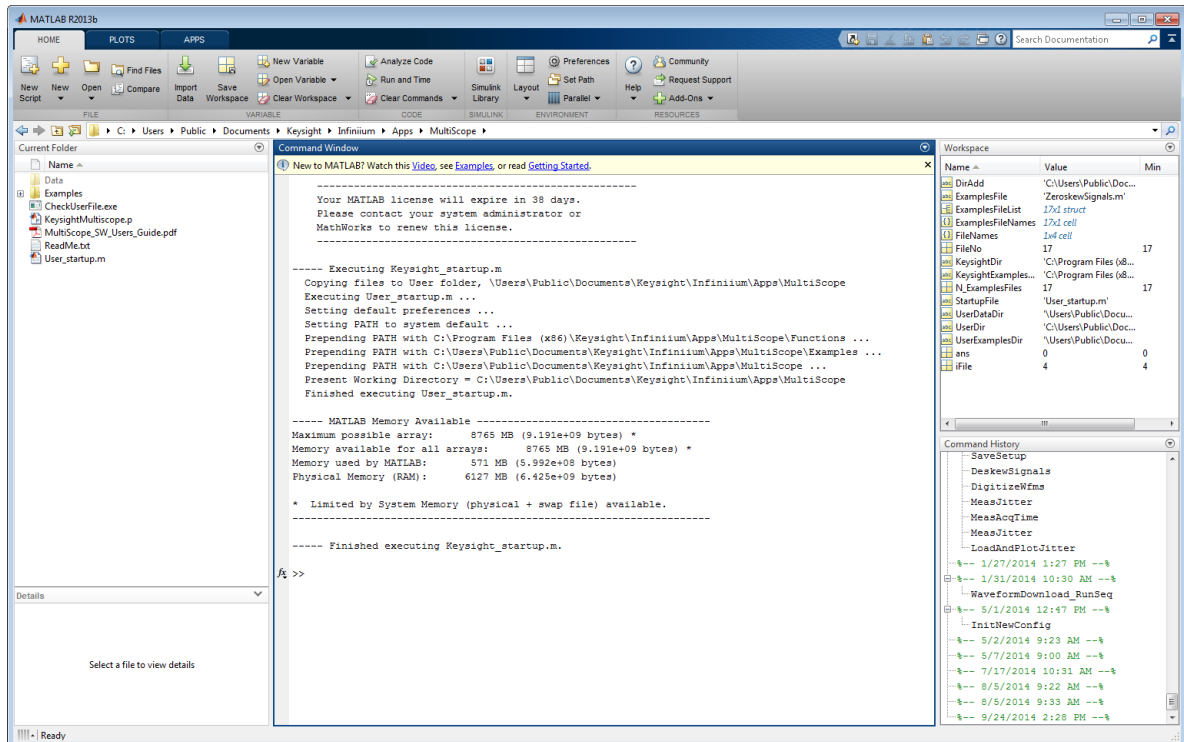
- Scripts vs. Functions** Functions create a new programming environment (that is, workspace or program scope); scripts simply execute commands in the current programming environment.
- Both scripts and functions contain MATLAB code, and both are stored in text files named with the script/function name and a .m extension.
- Editor Tips**
- In the MATLAB editor, you can right-click a script or function name to open its file.
- The MATLAB search path is used to locate the file.
- Use Ctrl+R and Ctrl+T to comment/uncomment highlighted blocks.
 - You can click just to the right of line numbers to set/clear breakpoints. When a breakpoint is hit during execution, press F10 to step or F5 to continue execution.
- Command Window Tips**
- In the Command Window, use the up arrow and down arrow keys to recall commands. Start typing to recall commands that began with the same characters.
- Running Scripts**
- In the Command Window, type in the script name and press Enter.
 - In the Current Folder hierarchy, select the script and press F9 or right-click and choose **Run**.
 - When editing scripts, you can click the Run icons in the Editor window.

Starting the MultiScope MATLAB Environment

After the MultiScope hardware has been configured, software has been installed on the host computer, and the remote interfaces between the host computer and the oscilloscopes have been set up, you can start the MultiScope MATLAB environment by double-clicking the **Keysight MultiScope** icon (on the host computer's desktop). This starts the MATLAB software and runs the `Keysight_startup` script (see "[Keysight_startup script](#)" on page 28).

The `Keysight_startup` script copies files to the default folder if necessary, runs the `User_startup` script to set defaults and the MATLAB search path (see "[User_startup script](#)" on page 28), and displays MATLAB memory statistics.

When all this completes, you will see a MATLAB window that looks similar to:



Note that a user-customized MATLAB start-up need only execute the `User_startup` file and not the `Keysight_startup` file.

Keysight_startup script

The `keysight_startup` script:

- 1 Copies files to the default user folder if necessary.

The `keysight_startup` script checks for the existence of the default user folder. The location of this default user folder on a Windows 7 PC is:

```
\Users\Public\Documents\Keysight\Infiniium\Apps\MultiScope
```

In the absence of this folder, `keysight_startup` creates the folder and copies a set of example files and a `User_startup.m` file to the folder.

A working data folder named "Data" is also created in the default user folder.

When you are installing a new version of the MultiScope MATLAB software, if the default folder exists, its contents will be copied to a backup folder at the same location before the latest versions of the files are installed.

- 2 Runs the `user_startup` script (see "**User_startup script**" on page 28).
- 3 Displays MATLAB memory statistics.

User_startup script

The `user_startup` script:

- 1 Sets default MATLAB preferences, for example, the default text interpreter or the command window format.
- 2 Sets the MATLAB search path.

The search path is used to locate the various scripts and functions used by MultiScope. Careful management of the search path determines which version of a file gets executed when multiple versions of the file exist.

When experimenting with script changes, you can copy a file to a directory that appears earlier in the search path and edit the file there.

Files in the Examples Folder

The Examples folder contains files that help you use the MultiScope system, including scripts and the Multiscope.m function file.

```

DeskewChans.m
DeskewFrames.m
DeskewSignals.m
DigitizeWfms.m
InitNewConfig.m
InitPowerOn.m
LoadAndPlotAcqTime.m
LoadAndPlotJitter.m
LoadAndPlotWfms.m
LoadSetup.m
MeasAcqTime.m
MeasJitter.m
Multiscope.m
SaveSetup.m
SetScopeAndDigitize.m
SysConfig.m
ZeroskewSignals.m

```

Scripts The scripts in the Examples folder either call the `Multiscope` function with appropriate Task (and other) arguments or load and plot recently captured data or measurements.

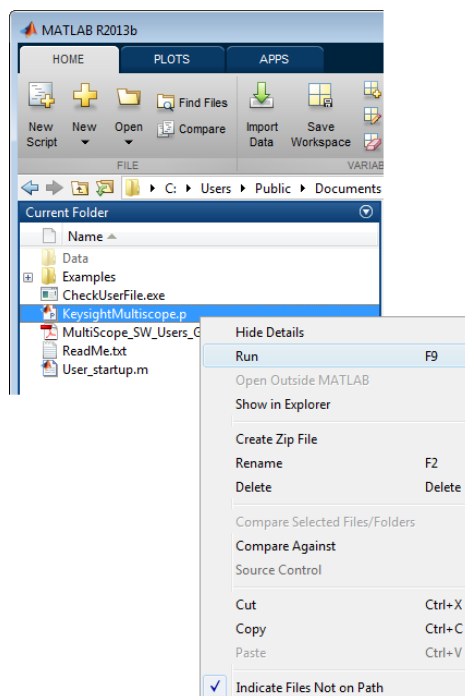
Multiscope Function The `Multiscope` function is a custom wrapper used to perform all MultiScope operations. For detailed information, see "[Multiscope function](#)" on page 60.

Starting the MultiScope MATLAB Graphical User Interface (GUI)

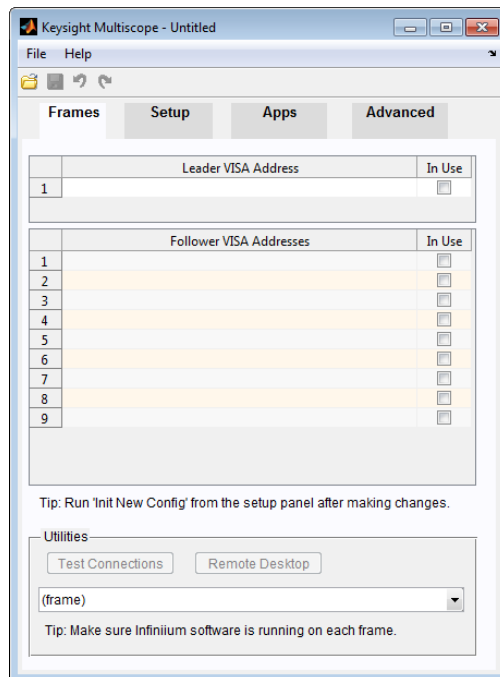
The MultiScope MATLAB graphical user interface provides an easier, more guided way to configure the MultiScope software and run the scripts in the Examples folder.

To start the MultiScope MATLAB graphical user interface:

- 1 In the MATLAB environment's Current Folder list, right-click KeysightMultiscope.p.
- 2 Choose **Run** from the right-click menu.



The graphical user interface has tabs (Frames, Setup, Apps, and Advanced) that guide you through the stages of using the MultiScope MATLAB software.

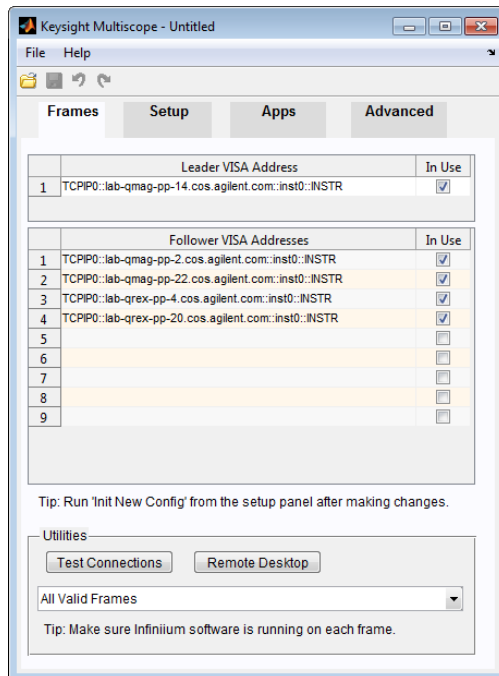


Specifying the Oscilloscopes in Your MultiScope System

You can specify the oscilloscopes in your MultiScope system either in the Frames tab of the MultiScope MATLAB graphical user interface or in the SysConfig script in the Examples folder.

In the GUI Frames Tab

In the MultiScope MATLAB graphical user interface, you can enter the VISA addresses of the oscilloscopes in your MultiScope system (see ["Finding VISA Addresses"](#) on page 33).



Once oscilloscope VISA addresses are entered, you can use the frames drop-down menu and these buttons:

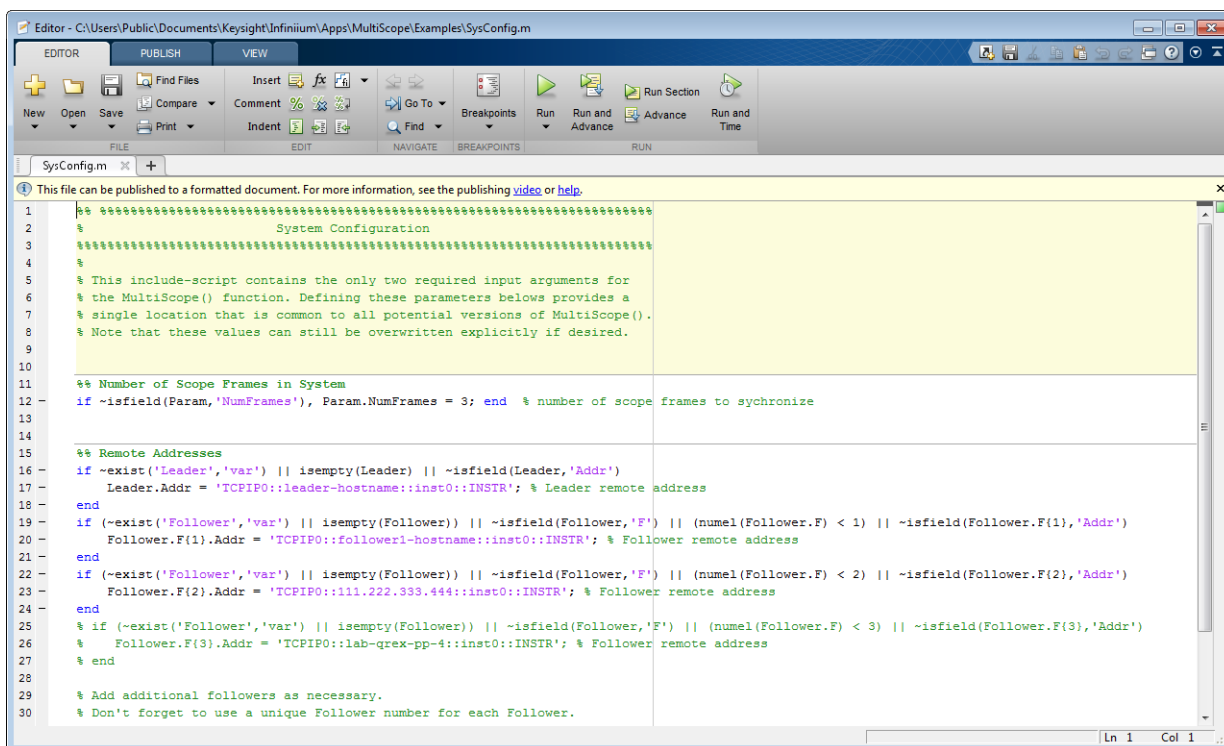
- **Test Connections** – sends a remote *IDN? query to the selected oscilloscope frames to verify connections (similar to the "testing commands" step in ["Verifying oscilloscope remote connections"](#) on page 15).
- **Remote Desktop** – launches a Windows Remote Desktop Connection to the oscilloscope frame for viewing the oscilloscope's user interface (as described in ["Viewing the Oscilloscope User Interfaces"](#) on page 25).

In the SysConfig script

You can edit the SysConfig.m script to specify the number of oscilloscopes in the MultiScope system and their remote interface VISA addresses.

While you can edit the original copy of `sysconfig.m` located in the Examples folder, it may be advantageous to create your own custom copy of `sysconfig.m` in the User folder. The precedence of the MATLAB search path should then locate and execute your custom version of the file instead of the original version.

When editing the SysConfig.m script file:



```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                               System Configuration
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %
5  % This include-script contains the only two required input arguments for
6  % the MultiScope() function. Defining these parameters below provides a
7  % single location that is common to all potential versions of MultiScope().
8  % Note that these values can still be overwritten explicitly if desired.
9
10
11 %%% Number of Scope Frames in System
12 if ~isfield(Param,'NumFrames'), Param.NumFrames = 3; end % number of scope frames to synchronize
13
14
15 %%% Remote Addresses
16 if ~exist('Leader','var') || isempty(Leader) || ~isfield(Leader,'Addr')
17     Leader.Addr = 'TCPIP0::leader-hostname::inst0::INSTR'; % Leader remote address
18 end
19 if (~exist('Follower','var') || isempty(Follower) || ~isfield(Follower,'F') || (numel(Follower.F) < 1) || ~isfield(Follower.F{1},'Addr'))
20     Follower.F{1}.Addr = 'TCPIP0::follower1-hostname::inst0::INSTR'; % Follower remote address
21 end
22 if (~exist('Follower','var') || isempty(Follower) || ~isfield(Follower,'F') || (numel(Follower.F) < 2) || ~isfield(Follower.F{2},'Addr'))
23     Follower.F{2}.Addr = 'TCPIP0::111.222.333.444::inst0::INSTR'; % Follower remote address
24 end
25 % if (~exist('Follower','var') || isempty(Follower) || ~isfield(Follower,'F') || (numel(Follower.F) < 3) || ~isfield(Follower.F{3},'Addr'))
26 %     Follower.F{3}.Addr = 'TCPIP0::lab-grex-pp-4::inst0::INSTR'; % Follower remote address
27 % end
28
29 % Add additional followers as necessary.
30 % Don't forget to use a unique Follower number for each Follower.

```

- 1 Set the `NumFrames` parameter to the number of oscilloscopes in the MultiScope system.

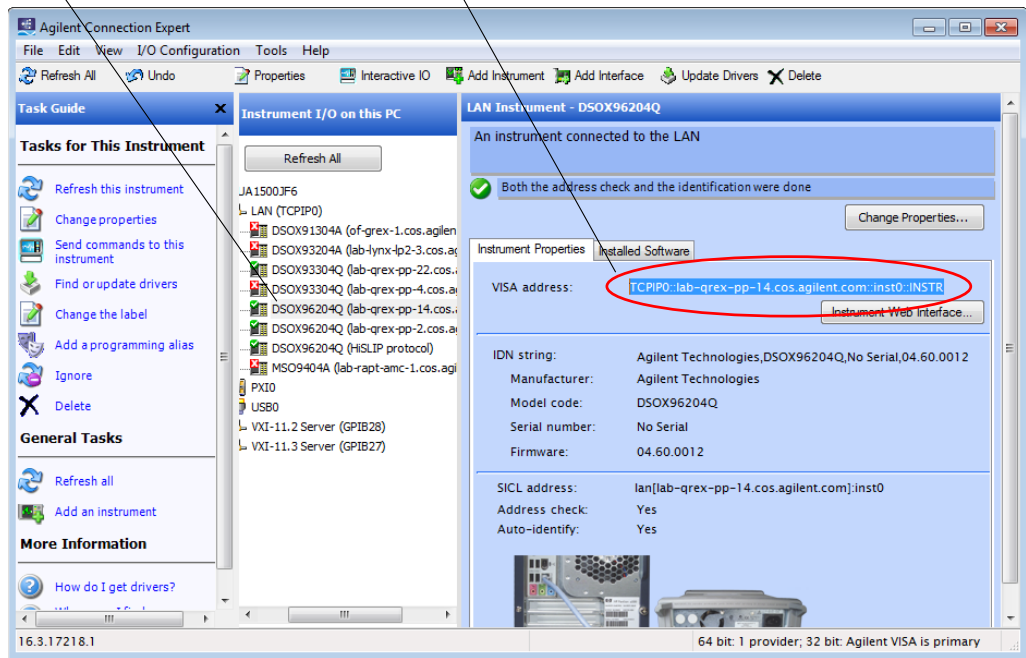
While it is uncommon, the `Param.NumFrames` can be set to 1 for the MultiScope software to configure and digitize waveforms on a single oscilloscope.

- 2 Set the `Leader.Addr` and `Follower.F{n}.Addr` variables to the VISA addresses of the leader and follower oscilloscopes (see **"Finding VISA Addresses"** on page 33).

Finding VISA Addresses

You can find oscilloscope VISA addresses using Keysight Connection Expert.

1. Select the oscilloscope
2. Cut-and-paste the VISA address



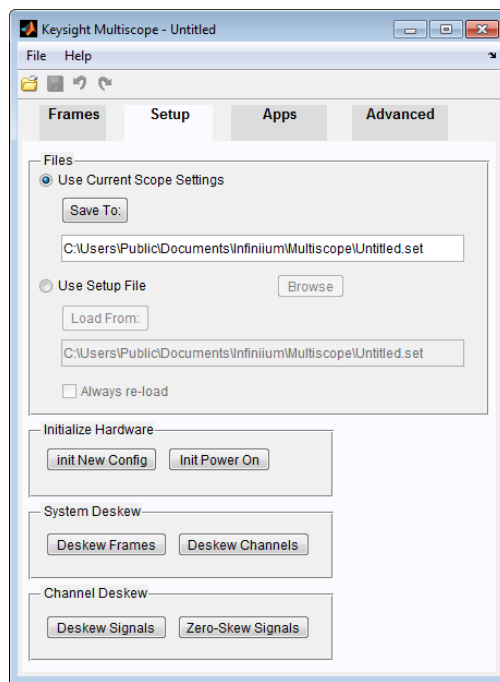
For instructions on opening the Keysight Connection Expert, see "[Verifying oscilloscope remote connections](#)" on page 15.

Initialize the Oscilloscopes (& Calibrate Reference Clock Skew)

Any time the MultiScope system is powered on, or when any part of the configuration changes, the oscilloscopes must be initialized.

You can initialize hardware in the Setup tab of the MultiScope MATLAB graphical user interface, by clicking these buttons:

- **Init New Config** – for complete initialization. This is the same as running the `InitNewConfig` Examples script (see "**InitNewConfig script**" on page 35).
- **Init Power On** – for a quicker, reference clock skew only initialization. This is the same as running the `InitPowerOn` Examples script (see "**InitPowerOn script**" on page 36).



A reference clock skew calibration needs to be performed each time the system is re-started, regardless of whether the connections used for synchronization are changed.

Once the reference clock skew calibration factors have been determined, you can begin digitizing input signals.

InitNewConfig script

The `InitNewConfig` script is a complete initialization that must be re-run each time the following hardware or acquisition configurations change:

- External system cables or adapters connected to the external timebase reference (rear panel), external trigger (rear panel), or drift correction calibration signal (front panel).
- The number of oscilloscope frames in the system.
- The oscilloscope frame serial number.
- The RealEdge mode.

InitPowerOn script

The `InitPowerOn` script is a simplified (and much faster) initialization script that re-establishes the phase-lock between oscilloscope timebase clocks and runs only the reference clock skew calibration.

The reference clock skew value needs to be re-computed each time the system is powered-on, regardless of a any configuration change.

It is not necessary to run `InitPowerOn` after running `InitNewConfig`, unless at least one of the oscilloscopes in the system was restarted.

Setting Up the Oscilloscopes

After initializing the MultiScope system, you can begin digitizing signals; however, before you do that, you most likely want to set up the oscilloscopes so that you digitize the waveforms you're interested in.

MultiScope system configuration setups are made up of system-common settings and channel-specific settings.

To set up the MultiScope system:

- 1 Make the system-common settings on the Leader oscilloscope (see "[Make System-Common Settings on the Leader Oscilloscope](#)" on page 37).
- 2 Make the channel-specific settings on all oscilloscopes (see "[Make Channel-Specific Settings on All Oscilloscopes](#)" on page 38).
- 3 Save the MultiScope setup (optional, see "[SaveSetup script](#)" on page 40).

CAUTION

After you have a desired setup, it is recommended to save it because subsequent processing made by the MultiScope MATLAB software will temporarily change oscilloscope configuration settings. If the processing is interrupted, the oscilloscopes will likely be left with different configurations.

Make System-Common Settings on the Leader Oscilloscope

The system-common settings include:

- sample rate
- acquisition bandwidth
- interpolation ratio
- horizontal scale
- horizontal position
- horizontal reference
- various trigger settings

Once stored, system-common settings are read from the Leader oscilloscope and copied to the Follower oscilloscopes.

Trigger Settings When specifying MultiScope setups, remember that the trigger source always originates from the Leader and is set up there.

Sample Rate Settings All frames in the MultiScope system share the same sample rate and acquisition bandwidth.

The automatic sample rate setting is not allowed. It will be interpreted as max sample rate. The first time the MultiScope MATLAB software is run on an oscilloscope that has a max sample rate set, it is converted to manual sample rate with the value set to max.

Bandwidth and sample rate have complicated interaction. If you specify both, sample rate is set first, then bandwidth second.

See Also ["Ways to Make MultiScope Settings"](#) on page 38

Make Channel-Specific Settings on All Oscilloscopes

The channel-specific settings include:

- vertical scale
- vertical offset
- channel display (on or off)
- channel skew – this refers to the **Skew** control in the Channel Setup dialog box (which can also be accessed remotely with the `:CHANnel<N>:PROBe:SKEW` command and query).

These are the very same oscilloscope settings contained within the Channel Setup dialog boxes of each oscilloscope.

Channel-specific settings are read from each oscilloscope channel in the system, including those on the Follower oscilloscopes (not copied just from the Leader oscilloscope).

See also ["Ways to Make MultiScope Settings"](#) on page 38.

Ways to Make MultiScope Settings

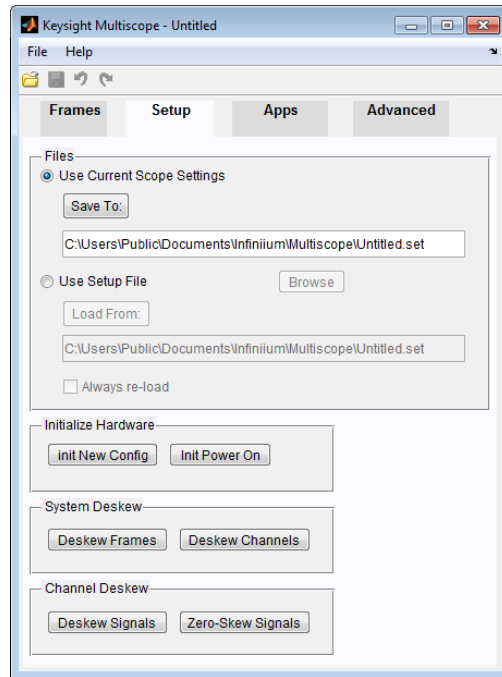
In the MultiScope MATLAB software, all configuration settings exist as valid MATLAB input parameter values. If MATLAB parameter values are not set, they are read from the oscilloscopes (and copied from the Leader oscilloscope to the Follower oscilloscopes in the case of system-common settings). Knowing this, you can make MultiScope system settings in these ways:

- From the oscilloscopes' front panel user interfaces (including using the **File > Load** menu to load setup files intended for each individual oscilloscope).

This is a common and familiar way to make MultiScope settings when a MultiScope setup file does not yet exist. Just be aware that system-common settings are read from the Leader oscilloscope (and copied to all Follower oscilloscopes) and that channel-specific settings are read from all oscilloscopes.

To access an oscilloscope's front panel user interface remotely from the host computer, see ["Viewing the Oscilloscope User Interfaces"](#) on page 25.

- In the Setup tab of the MultiScope MATLAB graphical user interface, by selecting **Use Setup File**. Click **Load From:** to load a previously saved MultiScope setup file. This is similar to running the `LoadSetup` script (see "**LoadSetup script**" on page 40).



This will set all MultiScope system configuration settings (except `NumFrames` and `<scope>.Addr`, which are specified in the Frames tab).

MultiScope setup files are saved by selecting **Use Current Scope Settings** and clicking **Save To:**. This is similar to running the `saveSetup` script (see "**SaveSetup script**" on page 40).

- By loading previously saved MultiScope setup files with the `LoadSetup` script (see "**LoadSetup script**" on page 40).

This will set all MultiScope system configuration settings (except `NumFrames` and `<scope>.Addr`, which are specified in `sysConfig.m`).

MultiScope setup files are saved with the `saveSetup` script (see "**SaveSetup script**" on page 40).

- By hard-coding parameter values in the MultiScope MATLAB scripts or functions.

Settings made this way override front panel settings and settings previously loaded from MultiScope setup files. See "**Setting MultiScope Configuration Parameters**" on page 54.

SaveSetup script

The `saveSetup` script saves a MultiScope setup to a `.set` file on the Leader oscilloscope. The MultiScope setup file has the same file format as conventional single-oscilloscope setup files, but includes additional channel information for the Follower oscilloscopes.

To specify a unique setup file name, edit the `saveSetup` script to uncomment the `SetupFileName` parameter, and specify the file name. Otherwise, the default "Untitled.set" file name as specified in the `MultiScope` function is used. Alternatively, you can modify the default name within the `MultiScope` function so that all scripts will inherit the new unique file name.

LoadSetup script

The `LoadSetup` script recalls a previously saved MultiScope setup from a `.set` file on the Leader oscilloscope.

To specify the exact file name, edit the `LoadSetup` script to uncomment the `SetupFileName` parameter, and specify the file name. Otherwise, the default "Untitled.set" file name as specified in the `MultiScope` function is used (or modify the default name within the `MultiScope` function).

When you load a single-oscilloscope `.set` file (saved from the front panel) using the `LoadSetup` script, the Leader's channels settings are copied to those of the same channels on all of the Followers.

If you load a MultiScope `.set` file using the Leader's front panel, all of the single-oscilloscope settings contained in the file are loaded into the Leader, and the follower settings are simply ignored.

The `LoadSetup` script does not change the Follower's other system-common settings (like sample rate, for example) because anything else you do with the `MultiScope` function will set them appropriately anyway.

NOTE

Channel skew is NOT affected by setup save/recall. This matches the long-time behavior of the Infiniium software on standalone oscilloscopes.

Deskewing Measured Waveforms

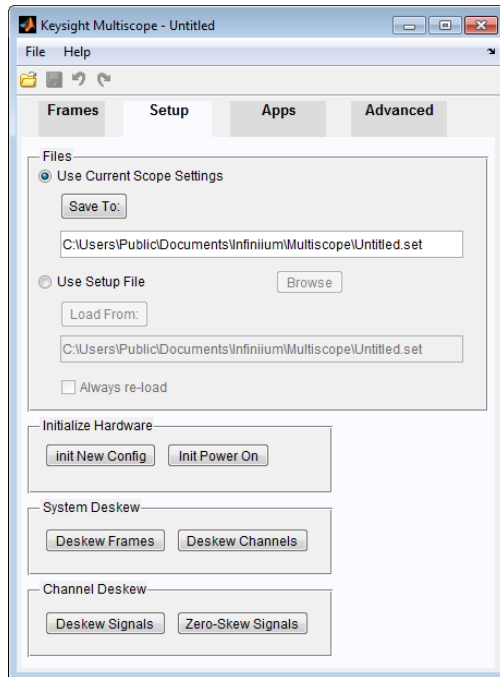
When digitizing multiple signals simultaneously, the relative time skew between the signals is often important to the measurement and must be calibrated. The reference to which you calibrate and the method used to apply the skew, however, depends on the application.

MultiScope has two different skew values that you can calibrate: system skew and channel skew.

- System skew is an internal MultiScope calibration factor that deskews all of the input channels to a specific physical reference plane (that is, the front-panel connectors). When the system skew is calibrated to the front panel connectors, for example, input signal waveform events (voltage pulses) that arrive simultaneously to all of the input connectors will appear as simultaneous events on the output digitized waveforms, provided that the channel skew values of all the input channels are set to zero. The total time skew applied is the sum of channel skew and system skew.
- Users do not have direct control over the system skew value, so channel skew is provided to enable users to deskew other channel-to-channel delays that are external to the oscilloscope measurement system (for example, cables or test fixtures). Channel skew in MultiScope is identical to channel skew on a single oscilloscope and is controlled from the GUI's Channel Setup dialog boxes, as are vertical scale and vertical offset.

System Deskew MultiScope provides two ways to calibrate the system skew:

- Deskewing frames – click the **Deskew Frames** button in the Setup tab of the MultiScope MATLAB graphical user interface or run the `DeskewFrames` script (see "[DeskewFrames script](#)" on page 43).
- Deskewing channels – click the **Deskew Channels** button in the Setup tab of the MultiScope MATLAB graphical user interface or run the `DeskewChans` script (see "[DeskewChans script](#)" on page 43).



These deskewing procedures prompt you through a series of steps connecting a common calibration signal to each input.

Deskewing the system skew is recommended each time you run the `InitNewConfig` script. Performing a system deskew after running `InitNewConfig` is recommended, but not strictly necessary. Doing so will reduce the calibrated skew from about 20 ps to less than 1 ps.

Channel Deskew MultiScope also provides these ways to set the channel deskew:

- Deskewing signals – click the **Deskew Signals** button in the Setup tab of the MultiScope MATLAB graphical user interface or run the `DeskewSignals` script (see "[DeskewSignals script](#)" on page 43).

Deskewing signals is primarily used for demonstration purposes because the optimum channel deskew values are normally signal and application dependent.

- Resetting channel skew values to zero – click the **Zero-Skew Signals** button in the Setup tab of the MultiScope MATLAB graphical user interface or run the `ZeroskewSignals` script (see "[ZeroskewSignals script](#)" on page 44).

The channel skew values are more commonly determined by your custom post-processing software and then passed into the Multiscope function as input parameters. See "[Channel-Specific Parameters](#)" on page 55.

It is rarely necessary to perform both a system deskew and a channel deskew because the channel skew is determined and then applied in addition to the system skew. If you do wish to deskew them both, you need to deskew the channel skew last.

DeskewChans script

The `DeskewChans` script deskews all channels in the system to a physical reference plane. The calibration process prompts the user to connect the Leader oscilloscope's Cal Out signal to each of the system's input channels in turn.

Note that the measurement reference plane need not be the oscilloscope's front-panel input connectors. You could also connect the common calibration signal to input connectors of a test fixture or collection of input cables.

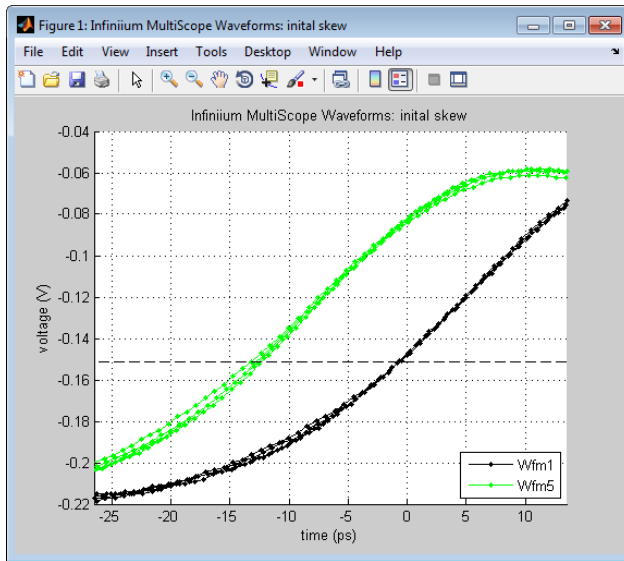
DeskewFrames script

The `DeskewFrames` script is an accelerated version of the `DeskewChans` script. It is identical to `DeskewChans`, except that it measures only one input channel from each oscilloscope frame. It assumes that the skew of all channels within a single oscilloscope frame is the same since they were deskewed in production.

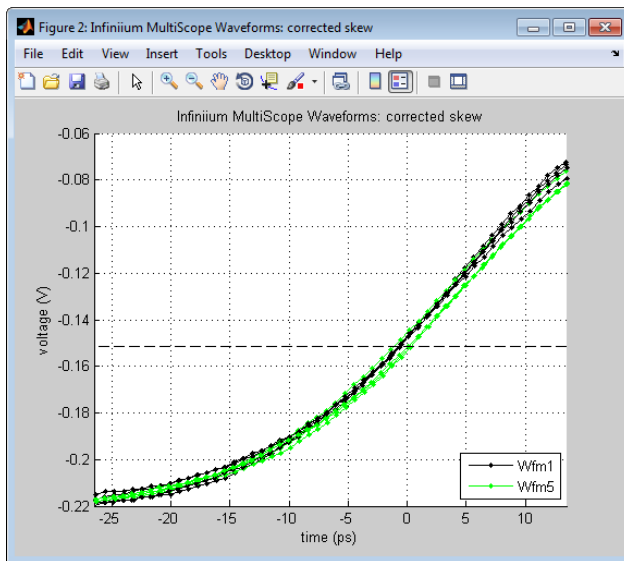
DeskewSignals script

The `DeskewSignals` script aligns the horizontal positions of the closest rising edges of all input signals together. This is typically used for demonstration or quick verification purposes, but can also be a quick alternative to the system deskew process if your SUT (signals under test) are already connected and have the necessary rising edges.

After the `DeskewSignals` script starts, it will plot initial skew values:



When the `DeskewSignals` script completes, it will plot the corrected skew results:



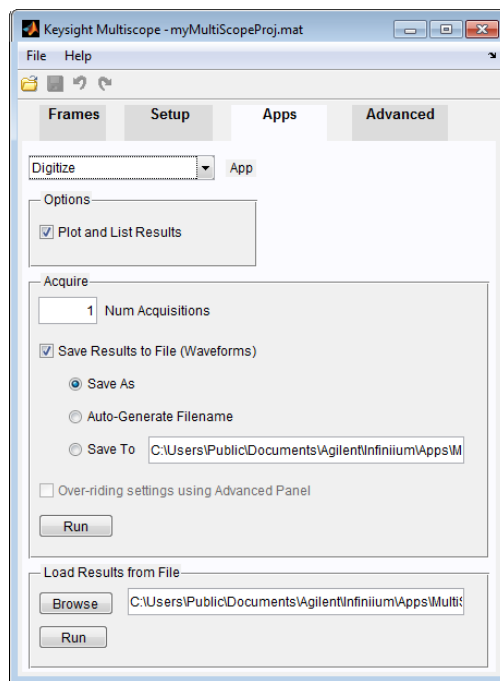
ZeroskewSignals script

The `zeroskewSignals` script simply resets all channel skew value settings to zero. This is provided as a convenience because the skew values are distributed across all channels of all oscilloscope frames and are not changed by setup recall or default setup. Factory default setup does set all skew values to zero, but must be performed on all oscilloscope frames.

Digitizing Input Signals

When **Digitize** is selected in the Apps tab of the MultiScope MATLAB graphical user interface, you can:

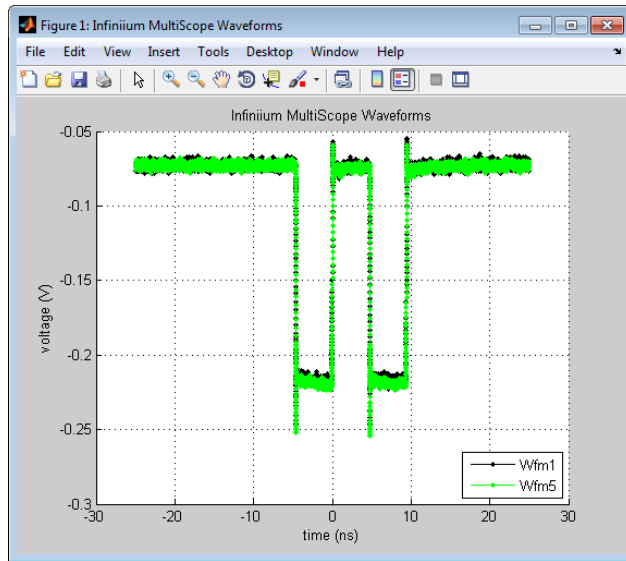
- Acquire (digitize) input signals, save, and plot waveform data. This is similar to the `DigitizeWfms` script (see "**DigitizeWfms script**" on page 45).
- Select to override oscilloscope settings during the acquisition using settings specified in the Advanced tab. This is similar to the `SetScopeAndDigitize` script (see "**SetScopeAndDigitize script**" on page 46).
- Load waveform data from a file and plot it. This is similar to the `LoadAndPlotWfms` script (see "**LoadAndPlotWfms script**" on page 46).



In the graphical user interface, click **Run** to perform the specified action.

DigitizeWfms script

The `DigitizeWfms` script digitizes the input signals (through the `Multiscope` function), saves the waveforms to a file, and then plots them.



After the MultiScope captures the input signals, it returns the digitized waveform data within a structured cell array variable, called `waveforms{}`.

See **"Output Argument"** on page 61 for more detail about the format of the waveform data.

CAUTION

To run fast, the `Digitize` function does not verify that the current calibration file data applies to the current oscilloscope settings and may result in unexpected error messages or measurement errors. So always ensure that the system has been properly initialized prior to digitizing waveforms.

SetScopeAndDigitize script

The `setScopeAndDigitize` script is the same as the `DigitizeWfms` script except that it allows you to specify hard-coded settings before digitizing the input signals.

See **"Setting MultiScope Configuration Parameters"** on page 54 for more details about explicitly specifying oscilloscope settings.

LoadAndPlotWfms script

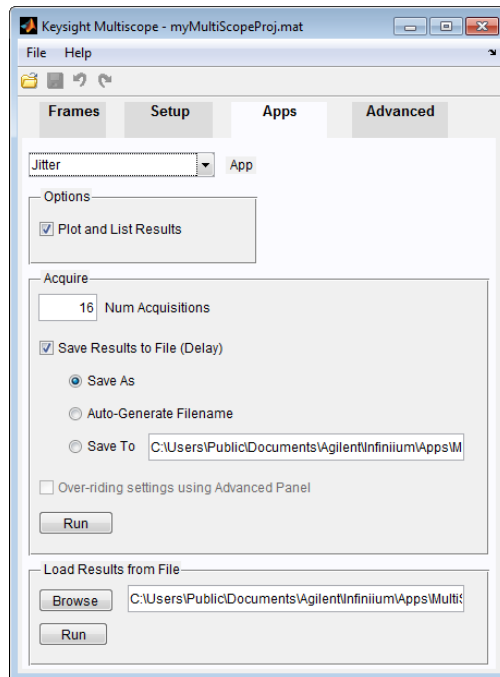
The `LoadAndPlotWfms` script loads and then plots saved waveform data.

You can run this script, for example, to load and plot waveform data that was previously digitized and saved to file. By default, the waveform data file "Test1_Wfm" will be loaded from the "Data" folder, but you can modify the script to select a different waveform file.

Measuring Jitter (Optional)

When **Jitter** is selected in the Apps tab of the MultiScope MATLAB graphical user interface, you can:

- Acquire (measure) and plot the MultiScope system's jitter (delay trend). This is similar to the `MeasJitter` script (see "**MeasJitter script**" on page 47).
- Load jitter results from a file and plot it. This is similar to the `LoadAndPlotJitter` script (see "**LoadAndPlotJitter script**" on page 48).

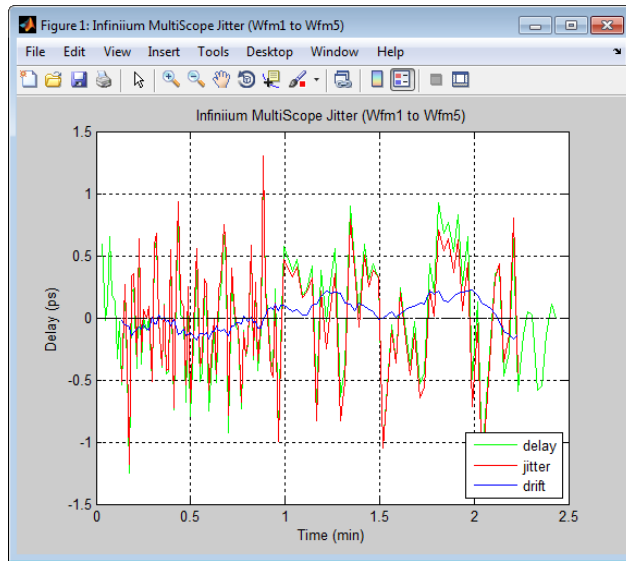


In the graphical user interface, click **Run** to perform the specified action.

MeasJitter script

The `MeasJitter` script calls Multiscope's `Digitize` for a specified number of acquisitions (`NumAcqs`) and calculates and plots the jitter and drift between `waveform{1}` and each of the other waveforms. It also plots the delay trend between the channels as well.

If you set the number of acquisitions larger than the `DriftSpan` value, this script will also break down the total time error into short-term jitter and long-term drift.



```

Delay (128 acqs) :
  Wfm1 to Wfm1 = 0.063 ps rms, 0.262 ps pp
  Wfm1 to Wfm5 = 0.477 ps rms, 2.544 ps pp

Jitter (112 acqs) :
  Wfm1 to Wfm1 = 0.064 ps rms, 0.271 ps pp
  Wfm1 to Wfm5 = 0.485 ps rms, 2.579 ps pp

Drift (112 acqs) :
  Wfm1 to Wfm1 = 0.012 ps rms, 0.051 ps pp
  Wfm1 to Wfm5 = 0.110 ps rms, 0.402 ps pp

```

LoadAndPlotJitter script

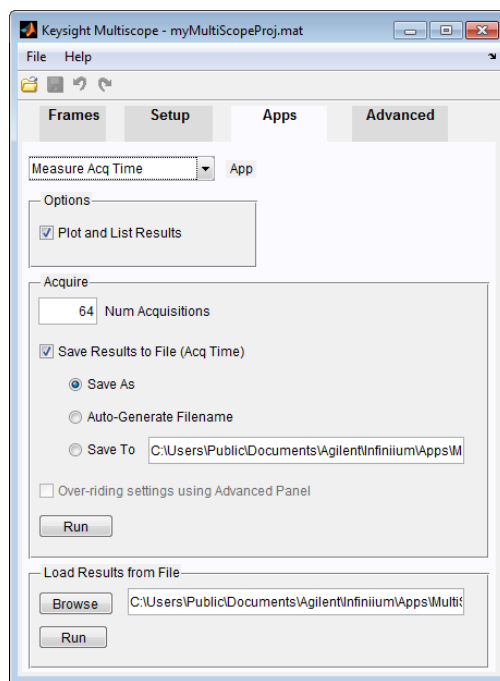
The `LoadAndPlotJitter` script loads and then plots saved jitter (delay trend) and drift data.

You can run this script, for example, to load and plot jitter and drift data that was previously saved to a file. By default, the jitter and drift data file "Test1_Delay" will be loaded from the "Data" folder, but you can modify the script to select a different jitter and drift data file.

Measuring Acquisition Times (Optional)

When **Measure Acq Time** is selected in the Apps tab of the MultiScope MATLAB graphical user interface, you can:

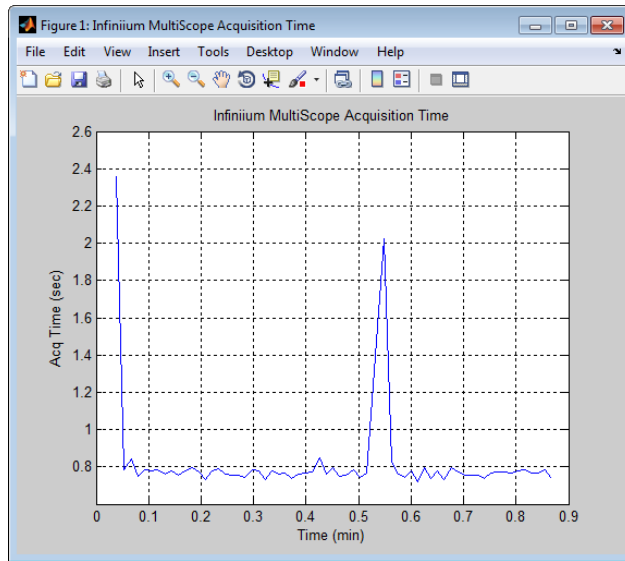
- Acquire (measure) and plot the MultiScope system's acquisition update rate. This is similar to the `MeasAcqTime` script (see "**MeasAcqTime script**" on page 49).
- Load acquisition update rate results from a file and plot it. This is similar to the `LoadAndPlotAcqTime` script (see "**LoadAndPlotAcqTime script**" on page 50).



In the graphical user interface, click **Run** to perform the specified action.

MeasAcqTime script

The `MeasAcqTime` script calls Multiscope's `Digitize` for a specified number of acquisitions (`NumAcqs`) and calculates and plots the acquisition update rate.



```
Acquisition Time = 0.72 sec/acq (min)
                  0.81 sec/acq (avg)
                  2.36 sec/acq (max)
```

The spikes that appear in the acquisition update rate plot are due to the extra time required to run drift correction. Drift correction is not performed after every acquisition. The frequency at which drift correction is performed can be changed using the `Param.DriftMeas.Rate` parameter.

LoadAndPlotAcqTime script

The `LoadAndPlotAcqTime` script loads and then plots saved acquisition time data.

You can run this script, for example, to load and plot acquisition time data that was previously saved to a file. By default, the acquisition time data file "Test1_Time" will be loaded from the "Data" folder, but you can modify the script to select a different acquisition time data file.

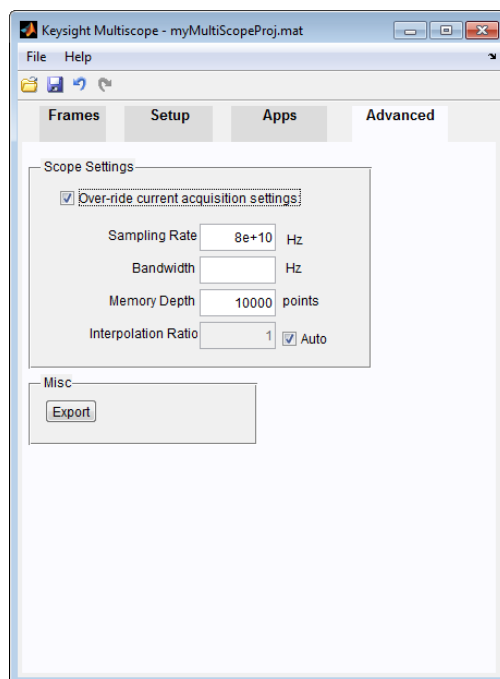
Using Advanced GUI Settings

In the Advanced tab of the MultiScope MATLAB graphical user interface, you can:

- Override current oscilloscope acquisition settings, including: sampling rate, bandwidth, memory depth, and interpolation ratio.

In some cases, such as changing the bandwidth, the hardware needs to be re-initialized (tool-tips provide information in these cases).

- Export all of the setup information to the MATLAB workspace, where you can run scripts or perform operations from the Matlab command line.



Saving and Opening GUI Projects

In the MultiScope MATLAB graphical user interface, you can choose **File > Save Project** or **File > Save Project As** to save MultiScope oscilloscope frame, setup file, data file, and other information to a project file.

You can choose **File > Open Project** to open a previously saved project file.

3 MultiScope MATLAB Software in More Detail

Setting MultiScope Configuration Parameters / 54

Organization of MultiScope MATLAB Functions / 59

Multiscope function / 60

You can use the scripts in the Examples folder (as described in [Chapter 2](#), “Getting Started,” starting on page 23) to perform all the basic tasks supported by the MultiScope MATLAB software.

This chapter describes options you can choose by setting configuration parameters, describes the `Multiscope` function in more detail, and attempts to answer questions you may have about using the MultiScope MATLAB software.

Setting MultiScope Configuration Parameters

You can hard-code parameter values within the scripts in the Examples folder or within the `MultiScope` function. Because `MultiScope` is called by the other scripts, it is a good central location for setting configuration parameter values.

MultiScope configuration parameters are used for setup file names, oscilloscope settings, operation settings:

- **"Setup File Parameters"** on page 54
- **"System-Common Parameters"** on page 55
- **"Channel-Specific Parameters"** on page 55
- **"Operation Parameters"** on page 56
- **"Waveform Plot Parameters"** on page 57
- **"Other Parameters and Defaults"** on page 58

Setup File Parameters

Table 1 Setup File Parameters

Parameter	Value	Description
SetupFileName	string	Name of the file on the Leader oscilloscope used for saving and loading and saving MultiScope setups. (Default = 'C:\Users\Public\Documents\Infiniium\Multiscope\Untitled.set')
AutoLoadSetup	boolean	When true, specifies that the the setup file identified by <code>SetupFileName</code> be reloaded prior to most tasks. This can be useful when debugging. (Default = false)

System-Common Parameters

Table 2 System-Common Parameters

Parameter	Value	Description
SRate	float	Sample rate (valid values constrained by specific oscilloscope hardware). RealEdge mode is not enabled/disabled explicitly with a dedicated parameter field. You enable RealEdge channels indirectly by setting the sample rate value to 160 Gs/s or something less than 160 Gs/s.
Points	integer	Memory depth (un-interpolated points).
Int	'auto', 'on', 1, 'off', 0, 'int1', 'int2', 'int4', 'int8', or 'int16'	Sets the interpolation ratio (for example, 'int16' = 16 interpolated points to 1 raw sample point). Use 'int16' (the maximum) for best-quality skew measurements.
BW	float	The -3 dB band width of all measured signals (limited to specific values by oscilloscope software).

System-common settings can be specified by passing their parameter values into `Multiscope` or setting within the `Multiscope` function. They can be set as a `Leader` field or a `Param` field. For example:

```
Leader.Int = 'int8';
```

Or:

```
Param.Int = 'int8';
```

Channel-Specific Parameters

Table 3 Channel-Specific Parameters

Parameter	Value	Description
Display(n)	1 or 0	This array contains channel display values indexed by waveform numbers.
Scale(n)	float	This array contains channel vertical scale values indexed by waveform numbers.
Offset(n)	float	This array contains channel vertical offset values indexed by waveform numbers.
Skew(n)	float	This array contains channel skew setting values indexed by waveform numbers.

Channel-specific settings can be specified by passing their parameter values into `MultiScope` or by setting them within the `MultiScope` function. Leader values are set as `Leader` fields. For example:

```
Leader.Chan1.Scale = 100e-3;
Leader.Chan3.Scale = 100e-3;
```

Follower values are set for each specific follower oscilloscope. For example:

```
Follower.F{1}.Chan1.Offset = 0; % first follower
Follower.F{1}.Chan3.Offset = 0; % first follower
Follower.F{2}.Chan1.Offset = 0; % second follower
Follower.F{2}.Chan3.Offset = 0; % second follower
Follower.F{3}.Chan1.Offset = 0; % third follower
Follower.F{3}.Chan3.Offset = 0; % third follower
```

Alternatively, values can be set for common channels of all followers. For example:

```
Follower.Chan1.Offset = 0;
Follower.Chan3.Offset = 0;
```

Although channel skew values can be set in the same way, it is sometimes more convenient to reference them by their MultiScope channel number instead of their single oscilloscope channel number. Alternatively, you can reference the setting values as follows:

```
Param.Skew(1) = 10e-12; % leader, chan 1
Param.Skew(3) = 20e-12; % leader, chan 3
Param.Skew(5) = 30e-12; % follower 1, chan 1
Param.Skew(7) = 40e-12; % follower 1, chan 3
Param.Skew(9) = 10e-12; % follower 2, chan 1
Param.Skew(11) = 20e-12; % follower 2, chan 3

Param.Scale(11) = 100e-3; % follower 2, chan 3
Param.Offset(11) = 0e-3; % follower 2, chan 3
Param.Display(11) = 1; % follower 2, chan 3
```

The channel display setting even provides a third method, as row vector of MultiScope channel numbers.

```
Param.WfmList = [1,2,5,9,13];
```

Operation Parameters

These parameters are passed in as fields to the `Param` structured array variable.

Table 4 Operation Parameters

Parameter	Value	Description
CalFileName	string	Configuration and reference clock deskew calibration data file path and name.
Debug	1 or 0	Specifies whether to run with debug messages or not. (Default = 1)
JitterCorr	boolean	Enable jitter correction. Enabling jitter correction phase-locks all of the oscilloscope timebase references together and provides the lowest inter-scope (between oscilloscope frames) jitter measurements. Disabling increases the inter-scope jitter considerably, but provides the fastest acquisition rate. (Default = true)
DriftCorr	boolean	Enable drift correction. Drift correction corrects for slowly varying time drift between oscilloscope frames, primarily due to temperature variations. Drift correction provides the best quality measurements, but has the slowest acquisition rate. Drift correction also requires jitter correction to be enabled. (Default = true)
AlignSamples	boolean	Aligns waveform samples of all waveform records. The output waveforms are always properly time-aligned relative to one another; however, their individual sample points may not occur at the sample points in time. When enabled, all output waveform records have the same number of samples and the same X-origin values. Enabling AlignSamples does slow down the acquisition rate because of the additional post-processing used to shift the waveform sample points. (Default = true)
DriftMeas.Rate	float	The rate that drift correction measurements are performed in times per second. This lets you increase the effective acquisition rate, for example, by only performing the drift correction once every few minutes. (Default = 1/30).

Waveform Plot Parameters

The waveform plot parameters are used as input arguments to the `PlotWaveforms` function.

Table 5 Waveform Plot Parameters

Parameter	Value	Description
Title	string	Window title to be used for plotted waveform.
IFig	int	Figure number to be used for plotted waveform.
XUnits	string	Units to be used for the plotted waveform X axis.
XMult	float	The multiplier that corresponds to XUnits used when plotting waveforms. For example, for XUnits = 'ps', XMult = 1e-12.

Other Parameters and Defaults

Not all MultiScope configuration parameters were described in the previous sections. You can learn about other parameters and their default settings in the `DefaultParams` function.

Organization of MultiScope MATLAB Functions

The MultiScope MATLAB software is designed to be efficiently incorporated into a larger acquisition and analysis program. The high-level Multiscope function was written later to form a "wrapper" around a collection of low-level functions. This wrapper simplifies the application's usability for simple waveform digitizing and also provides convenient examples of using the lower-level functions. The top-level scripts (such as `InitNewConfig`, `DeskewChans`, and `MeasJitter`) are additional examples of how to use the wrapper function, `MultiScope`.

Within `MultiScope`, there are two primary functions, `ConfigMultiscope` and `AcquireMultiscope`, which do most of the work.

AcquireMultiscope function

As the name implies, `AcquireMultiscope` digitizes the input signals and returns the digitized waveforms in the form of a structured cell array. But just acquiring the waveforms involves several steps that include: reading the existing oscilloscope settings, changing the oscilloscope settings prior to the acquisition, triggering all the oscilloscopes in sequence, reconfiguring the oscilloscopes for the optional drift correction acquisitions, correcting the waveform's inter-oscilloscope jitter and drift, and then returning the oscilloscope's settings to their original values.

`AcquireMultiscope` assumes that all of the necessary configuration parameters and calibration factors it uses are already defined in the MATLAB workspace and are passed into it through the `Leader`, `Follower`, and `Param` structured array variables. It also assumes that the oscilloscope's timebases are already phase-locked together, if that level of synchronization is specified.

ConfigMultiscope function

`ConfigMultiscope` performs a variety of configuration and calibration functions depending on the values of the various `Param.<field_name>` configuration values. One of the things it does, for example, is phase-lock the oscilloscope's timebases together if necessary.

Although `ConfigMultiscope` can be called prior to every acquisition, it saves the necessary calibration factors in a calibration data file, so it does not really need to be called again until some calibration factor has changed.

Multiscope function

It may be helpful to understand the `multiscope` function in more detail, especially when making changes to the provided files.

The `multiscope` function is the main function called by the scripts in the Examples directory. It is a wrapper function to perform all basic MultiScope operations.

Input Arguments `Task, Param, Leader, Follower`

The Task argument can be one of the strings described in the following table.

Task Argument String	Description
<code>InitNewConfig</code>	<p>This task:</p> <ol style="list-style-type: none"> 1 Loads setup if <code>AutoLoadSetup</code> is true. 2 Sets any uninitialized input arguments required by <code>AcquireMultiscope</code> to default values and performs a reference clock skew calibration (calls <code>ConfigMultiscope</code> with <code>Param.ConfigChanged = true</code>). 3 Saves the current time as the <code>CalTime</code>.
<code>InitPowerOn</code>	<p>This task:</p> <ol style="list-style-type: none"> 1 Loads setup if <code>AutoLoadSetup</code> is true. 2 Performs a reference clock skew calibration (calls <code>ConfigMultiscope</code>). 3 Saves the current time as the <code>CalTime</code>.
<code>DeskewChans</code>	<p>This task:</p> <ol style="list-style-type: none"> 1 Loads setup if <code>AutoLoadSetup</code> is true. 2 Sets any uninitialized input arguments required by <code>AcquireMultiscope</code> to default values and performs a reference clock skew calibration (calls <code>ConfigMultiscope</code> with <code>Param.DeskewChans = true</code>). 3 Displays deskew results.
<code>DeskewFrames</code>	<p>This task:</p> <ol style="list-style-type: none"> 1 Loads setup if <code>AutoLoadSetup</code> is true. 2 Sets any uninitialized input arguments required by <code>AcquireMultiscope</code> to default values and performs a reference clock skew calibration (calls <code>ConfigMultiscope</code> with <code>Param.DeskewFrames = true</code>). 3 Displays deskew results.

Task Argument String	Description
DeskewSignals	<p>This task:</p> <ol style="list-style-type: none"> 1 Loads setup if <code>AutoLoadSetup</code> is true. 2 Tests whether the oscilloscope has been powered-off since last reference clock skew values have been calculated (<code>InitPowerOn</code> or <code>InitNewConfig</code> may be required). 3 Measures initial signals (<code>AcquireMultiscope</code>) and calculates delay trend. 4 Plots initial signals. 5 Calculates deskew. 6 Measures and plots corrected signals. 7 Lists deskew results.
ZeroskewSignals	<p>This task:</p> <ol style="list-style-type: none"> 1 Sets up default <code>Param</code> values for parameters that are not defined (<code>DefaultParams</code>). 2 Sets skew on all active channels to zero.
SaveSetup	<p>This task:</p> <ol style="list-style-type: none"> 1 Saves Leader and Follower setups to <code>SetupFileName</code>. 2 Clears Follower setups from Leader in case single frame setups are saved from the front panel.
LoadSetup	<p>This task:</p> <ol style="list-style-type: none"> 1 Loads Leader and Follower setups from <code>SetupFileName</code>. 2 If Follower settings are not found in the setup file, Leader settings are copied to all Followers. 3 Clears Follower setups from Leader in case single frame setups are saved from the front panel.
Digitize	<p>This task:</p> <ol style="list-style-type: none"> 1 Loads setup if <code>AutoLoadSetup</code> is true. 2 Loads parameter values from calibration data file. 3 Configures the oscilloscopes, performs a synchronized acquisition, optionally corrects for jitter and drift between waveforms, and returns the digitized waveforms.

Notice the similarity between the supported Task string names and the other provided scripts in the Examples directory.

Output Argument `Waveforms`

The `Multiscope` function returns waveforms in a structured cell array variable, called `Waveforms{}`. Each element within `Waveforms` represents a specific channel within the entire MultiScope system. Each single-frame channel number maps to the MultiScope channel number by the following formula:

$$\text{WfmNum} = 4 * (\text{FrameNum} - 1) + \text{ChanNum}$$

Where:

- The Leader is FrameNum=1
- The first Follower, Follower.F{1} is FameNum=2, etc.

When RealEdge channels are used, the numbering is the same, but there are no even-numbered MultiScope channels.

The contents of each Waveform element are a structured array with fields:

Field	Value	Min	Max
ScopeVendor	'Agilent Technologies'		
ScopeModel	'DSOX96204Q'		
ScopeSerNum	'No Serial'		
ScopeOptions	'04.60.9010P'		
ScopeChan	'Chan2'		
XOrg	-6.2500e-08	-6.2500e-08	-6.2500e-08
XInc	7.8125e-13	7.8125e-13	7.8125e-13
XRef	0	0	0
YOrg	0.0025	0.0025	0.0025
YInc	0.0017	0.0017	0.0017
YRef	0	0	0
Points	160000	160000	160000
Data	<160000x1 double>	-94	88
Scale	0.0500	0.0500	0.0500
Offset	0	0	0

waveform{ } elements for those MultiScope channels that are not digitized return empty cells.

Examination of the LoadAndPlotWfms script, and the sub-functions it calls, provides examples of how to interpret and post-process the waveform data within Waveforms.

4 Customizing Functions/Scripts

Software Coding Conventions / 64

Parameter Passing / 65

Save Changes in the User Folder / 66

When first customizing the functions or scripts in the MultiScope MATLAB software, you need to understand things like coding conventions used, how parameters are passed, and strategies for saving and testing changes.

Software Coding Conventions

Although not strictly necessary for functionality, the following coding conventions are used in the MultiScope MATLAB software to make it a little easier to read.

Custom function names, variable names, and file names begin with a capital letter and can be constructed as compound words separated by capitalization. This distinguishes them from built-in MATLAB names, which all begin with a lowercase letter.

Single index variable names begin with either `i`, `n`, or `I`, where:

- `i` is used in iterated loops (for example, `for iChan = 1:4`).
- `n` is used as an arrayed index variable (for example, `nPoint = (1:Points).'`).
- `I` is used to select a specific element of an array (for example, `wfm{IWfm}.XOrg = SharedXOrg`).

Parameter Passing

All input parameters used in the MultiScope MATLAB software are optional (except `<scope>.Addr` and `Param.NumFrames`) and will default to some predictable value if not explicitly specified.

Note that while the MultiScope MATLAB software's use of structured arrays for parameter passing simplifies function customization, it does expose you to a certain risk. If you misspell a particular input parameter (for example, "Leader.SampRate" instead of "Leader.sRate"), that particular input parameter will be ignored and will default to current oscilloscope setting without producing an error message.

Also, some parameters can be specified using different variables, with a particular precedence of interpretation. For example, you can explicitly specify which waveforms are to be acquired and returned by defining `<scope>. (ChanStr) .Display`. If `<scope>. (ChanStr) .Display` is not defined, then `Param.Display(<multiscope_channel_number>)` is used. If `Param.Display(<multiscope_channel_number>)` is not defined, then `Param.WfmList` is used. If `Param.WfmList` is not defined either, then only those channels that are currently being displayed will be acquired.

In the case of the Follower's parameters, you can specify each individual Follower's settings using `Follower.F{iF} .<field_name>` (where `iF` specifies the Follower oscilloscope number) or you can specify all of the Followers simultaneously using `Follower.<field_name>`.

Save Changes in the User Folder

When making changes to functions/scripts, it is best to save the modified copy to the default user folder (that is the parent of the Examples folder) because this folder appears earlier in the MATLAB search path and will be read before the functions/scripts in the Examples folder.

Because the MATLAB editor also uses the search path, the next time you open a customized script or function file by right-clicking its name in another file, you will be editing the version saved in the user folder.

5 Troubleshooting

Errors Opening a Device Interface That Is Already Open / 68

Remote Command Errors / 69

"Data out of range" Errors / 70

Successful use of MultiScope MATLAB software requires a working knowledge of MATLAB, including its debugging features (such as setting break points and single-step execution). This chapter provides additional information to aid in troubleshooting MultiScope MATLAB software problems.

Errors Opening a Device Interface That Is Already Open

MATLAB will report an error if a program tries to open a device interface that is already open. The error message will look something like the following:

```

??? Error using ==> icinterface.fopen at 83
The specified configuration: TCPIP0::lab-qrex-pp-2::inst0::INSTR is not
available.
Use INSTRHWINFO for a list of available configurations. Use INSTRFIND to
determine if other instrument objects are connected to the requested ins
trument.

Error in ==> OpenDeviceInterface at 31
    fopen(Device.ID);

Error in ==> CleansingBreath at 7
Leader = OpenDeviceInterface(Leader);

Error in ==> ConfigMultiscope at 17
[Leader,Follower,Param] = CleansingBreath(Leader,Follower,Param);

Error in ==> Multiscope_probecomp at 169
    [Leader,Follower,Param] = ConfigMultiscope(Leader,Follower,Param
);

```

This can happen if the MultiScope MATLAB software exits unexpectedly without closing the device interfaces.

In this case, you can close all open device interfaces by running the `CloseInterfaces` script by executing the following command in the Command Window:

```
CloseInterfaces;
```

The scripts provided in the Examples folder all include the `CloseInterfaces` script to prevent these errors.

Remote Command Errors

A function called `CheckDeviceStatus` is distributed throughout the various MultiScope MATLAB software functions to check for remote command problems. `CheckDeviceStatus` checks the oscilloscope's Device Status Register to see if a remote command has produced an error. If an error occurs, MultiScope MATLAB software will report an error message similar to the following.

```
??? Error using ==> ConfigMultiscope at 169
Follower error in ConfigMultiscope: -113,"Undefined header"

Error in ==> Multiscope_probecomp at 169
    [Leader,Follower,Param] = ConfigMultiscope(Leader,Follower,Param
);
```

The above message indicates that the Follower has received an unrecognized remote command. Other issues will also cause errors, such as trying to set a scale value beyond its allowable range.

Because `CheckDeviceStatus` is not called after every remote command is sent, the error message only tells us that the error occurred before the line in which it was detected and after the previous call to `CheckDeviceStatus`. The following process is recommended for troubleshooting these errors:

- 1 Identify which oscilloscope experienced the error.
- 2 Identify the location in the MultiScope MATLAB software where the error was detected and reported.
- 3 Identify the location in the MultiScope MATLAB software where `CheckDeviceStatus` was previously called.
- 4 Set a break point in the software immediately after the previous call to `CheckDeviceStatus`.
- 5 Re-run MultiScope MATLAB software until it stops at the break point.
- 6 Single-step program execution while monitoring the display of the oscilloscope that experienced the error.

An error or warning message will appear on the oscilloscope display when the problem remote command is executed.

Note also that you can call `CheckDeviceStatus` from the Command Window while stopped in debug mode. Executing:

```
Leader = CheckDeviceStatus(Leader)
```

for example (without a semicolon on the end), will display the `Leader.Status` and `Leader.Message` values.

Calling `CheckDeviceStatus` resets the Device Status Register after it is read.

"Data out of range" Errors

The '-222, "Data out of range"' error is a common, but uninformative error message. It is actually an oscilloscope remote command error message that is simply passed through the MultiScope MATLAB software.

A couple ways this error can occur are:

- By not running `InitNewConfig` after a hardware change.

In this case, simply re-run `InitNewConfig`.

- When performing a new measurement after the previous one exited unexpectedly, leaving one or more of the oscilloscopes in an invalid configuration.

In this case, restore the oscilloscopes to a valid configuration (see **"Setting Up the Oscilloscopes"** on page 37) and perform the new measurement again.

If this error occurs for some other reason, try the procedure in **"Remote Command Errors"** on page 69.

6 Theory of Operation

Oscilloscope Architecture /	72
Hardware Configuration /	73
Synchronizing Multiple Oscilloscopes /	74
Trigger Out to Trigger In Synchronization /	76
Frame Lag /	77
Skew /	78
External Skew /	80
Internal Skew /	81
Sample Alignment /	83
Channel Delay Skew /	84
Jitter Correction /	86
Drift Correction /	90
Post-Acquisition Skew /	91
Setup/Recall /	92

It is not necessary to understand oscilloscope architecture or how the MultiScope MATLAB software works in order to use it, but it may be helpful when customizing the provided files or porting them to another programming language.

Oscilloscope Architecture

Infiniium real-time oscilloscopes use a random-repetitive/real-time sampling architecture. The timing of their ADC's sampling aperture is synchronous to the oscilloscope's fundamental timebase reference and asynchronous from input signals and the system trigger event. This timebase reference originates from either an internal 10 MHz crystal oscillator or an external reference signal. This timebase reference is multiplied up in frequency to generate a 10 GHz sample clock. Multiple phases of the 10 GHz sample clock form the ultimate effective sample rate. The 10 GHz sample clock is divided down to generate a 250 MHz memory controller clock. All of these clocks are synchronous to one another, but asynchronous from the actual trigger event.

Prior to each acquisition (system trigger event), the ADC begins digitizing the enabled channels and saving them to memory. The memory is effectively circular. It will eventually start overwriting itself, but never fills up. After enough pre-trigger data has been collected, the trigger is enabled. After the trigger event occurs, the ADC is allowed to continue saving data to memory until the required post-trigger data has been collected. Because the memory controller clock is 250 MHz, the beginning and ending of the acquired waveforms always fall on 4 ns boundaries. This 4 ns boundary characteristic is not true of displayed or extracted waveforms, but it is true of the internally captured waveforms.

The X-origin (time position of the first point in the waveform record) of each waveform defines the waveform's horizontal position relative to the trigger event. The memory controller always knows the X-origin within 4 ns because it controls the acquisition window. However, a trigger interpolator circuit must be used in order to determine exactly where within the 4 ns the trigger event occurred to a finer resolution. The trigger interpolator measures the time between the trigger event and the end of the current memory controller clock cycle.

Hardware Configuration

The MultiScope MATLAB software works with a variety of oscilloscope models having different hardware capabilities. However, you do not have to specifically identify the model numbers of each oscilloscope in the MultiScope system. Instead, the MultiScope MATLAB software interrogates the hardware capabilities of each oscilloscope in the system by testing how each of them responds to various remote queries. For example, if an oscilloscope complains that it cannot be set to 160 GS/s, for example, then it must not have RealEdge capability.

Synchronizing Multiple Oscilloscopes

When synchronizing multiple oscilloscopes to increase the effective number of channels, the MultiScope MATLAB software has three selectable levels of inter-oscilloscope time synchronization:

- trigger out
- jitter correction
- drift correction

Trigger out synchronization must always be present, but you can disable or re-enable jitter or drift correction using the parameters `JitterCorr` and `DriftCorr`.

These three levels build on one another, such that all three methods use the trigger output process. Jitter correction then adds some additional processing to it. Drift correction uses both the trigger output process and the jitter correction process, and then adds the drift correction process to those. Because of this, the trigger output method has the fastest acquisition update rate, but also has the highest jitter and drift. Jitter correction runs a little slower, but has the lowest jitter. Drift correction has the slowest update rate of the three methods, but also has the lowest temperature drift of all three methods.

Trig Out to Trig In Synchronization

The simplest method of synchronizing multiple oscilloscopes is to connect Trig Out (the oscilloscope's system trigger output) from one oscilloscope, call it the Leader, to the Aux Trig (the auxiliary trigger input) of another oscilloscope, call it the Follower. Then, when the Leader triggers and digitizes its input signals, the Follower will also trigger and digitize its input waveforms some fixed time delay later. This does work and does not require the use of MultiScope MATLAB software, but it suffers from a relatively large amount of inter-oscilloscope jitter and drift between the channels of the different oscilloscopes. See "[Trigger Out to Trig In Synchronization](#)" on page 76.

Ref Clk Out to Ref Clk In Synchronization

The next level of synchronization phase-locks the timebase reference of the Follower oscilloscopes to that of the Leader oscilloscope and it does require the use of MultiScope MATLAB software. This method works by calculating the X-origin of the Follower oscilloscopes' waveforms from the X-origin of the Leader's waveforms. Because the oscilloscopes sample clocks are phase-locked together, their waveform samples must be as well. Note that this requires the use of some undocumented remote commands that access the X-origins of the oscilloscope's internal acquisition waveforms that always begin and end at those 4 ns boundaries discussed earlier.

Reference Clock Skew Calibration

Note also that, although the oscilloscopes' 250 MHz memory controller clocks are effectively phase-locked together so that the relative phase between the oscilloscopes 4 ns boundaries are fixed, that phase can change every time either oscilloscope timebase is restarted. This start-up phase uncertainty requires you to perform a reference clock skew calibration cycle each time the system is re-started or re-configured before you can use MultiScope MATLAB software. This

MultiScope MATLAB software reference clock skew calibration requires you to connect the Follower's calibrator output through a power splitter to a synchronization input on each oscilloscope.

Jitter and Drift Correction

The highest level of synchronization corrects for jitter and long-term time drift between the oscilloscopes. Drift correction is done by measuring the inter-oscilloscope time drift on every signal acquisition and correcting for it. Drift correction requires the same calibrator output connection used for the reference clock skew calibration to be present all the time. See:

- **"Jitter Correction"** on page 86
- **"Drift Correction"** on page 90

Trigger Out to Trigger In Synchronization

The trigger out process provides the most basic level of inter-oscilloscope time synchronization. This process connects the trigger output signal (Trig Out) from the Leader oscilloscope through each Follower oscilloscope (Aux Trig) in a daisy-chain configuration. The Followers are each armed in descending order from furthest from Leader ($F\{N\}$) to nearest from Leader ($F\{1\}$). The Leader's trigger is armed last. Once the Leader triggers, then each Follower will also trigger in turn, down through the chain.

Using only the trigger out synchronization, all waveforms throughout the system are read into the MultiScope MATLAB software across the remote interface. The time position of each waveform is not altered by the MultiScope software, but passed on directly as it was determined by the oscilloscope that digitized it. In the absence of sample alignment (explained later), none of the waveform's X-origins are modified by the MultiScope software's post-processing.

Note that in this mode of operation, the Followers are unaware of any jitter or drift between the Leader's trigger event and their own. So the Trig Out + Aux Trig jitter and drift between each pair of oscilloscopes appears as an error in the Follower's `wfm{i}wfm}.Xorg` values.

Frame Lag

Note that in the previous Trigger Out description, the system trigger signal experiences a physical propagation delay as it travels down the trigger chain. The length of this propagation delay depends on the physical length of the cables used to connect the trigger signals between oscilloscopes, and is approximately 30 ns between each oscilloscope when using the recommended 2 ft BNC cables. This too would add to the error in each Follower's `wfm{iWfm}.xor_g` value without some corrective action.

During the hardware initialization process (that is, running `InitNewConfig`), the actual trigger delay between oscilloscopes is measured and then calibrated out using the variable `Follower.F{iF}.FrameLag` which represents the nominal time-lag of each Follower's system trigger event relative to the Leader's system trigger event.

Each Follower's `FrameLag` value is corrected by `MultiScope`, even when using the basic trigger out level of synchronization.

Skew

While "**Synchronizing Multiple Oscilloscopes**" on page 74 discusses the random time error variation between waveforms, this section discusses the constant or time-invariant component of time error between waveforms. To understand how waveforms are deskewed in a MultiScope system, it is helpful to understand how they are deskewed within a single oscilloscope.

Consider the single-frame block diagram of **Figure 2**. In this simplified system, the ADCs sample the input signals synchronously to the timebase clock, but asynchronously to the input signals themselves and the system trigger event. The memory controller begins storing pre-trigger waveform samples in memory prior to enabling the trigger. Once the trigger is enabled and a trigger event occurs, the memory controller stores the appropriate number of post-trigger samples into memory and the acquisition is complete.

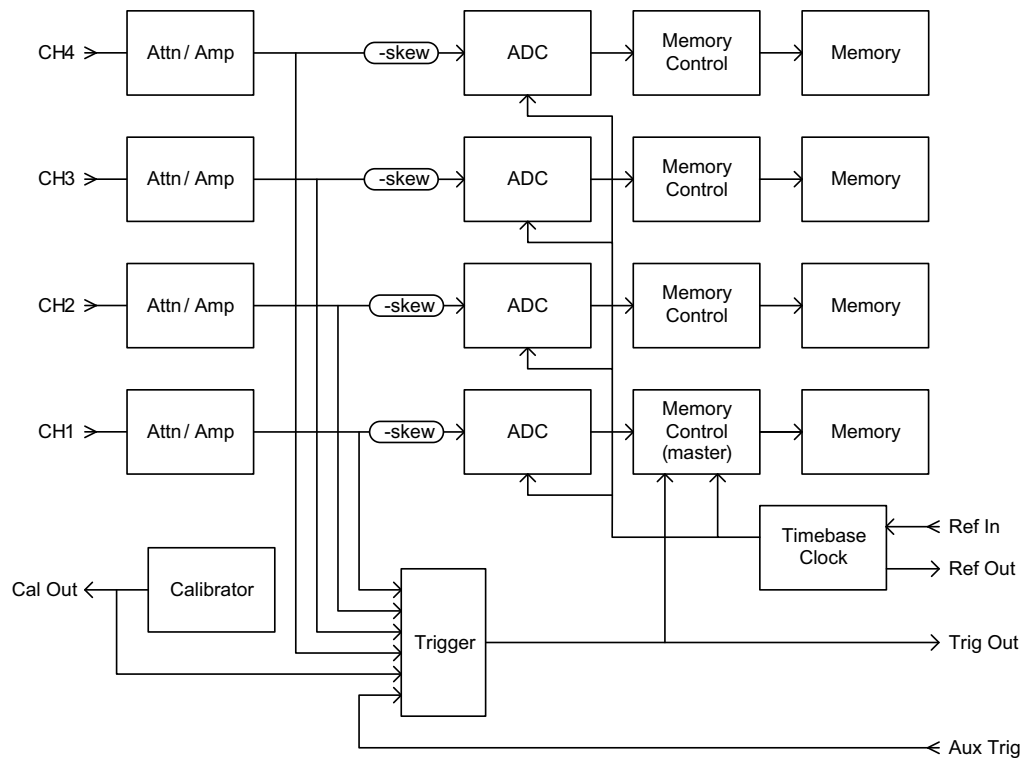


Figure 2 Simplified block diagram of a single oscilloscope frame

Differing propagation delays between the various physical electrical paths in this system (such as signal paths, clock paths, and trigger paths) cause time skews to occur between the digitized waveforms. However, all of these skews are corrected for by skewing (adjusting the $x0_{\pm g}$ value of) the four digitized channel waveforms.

The skew adjustment value applied to each channel is comprised of all the various calibrated deskew components of the system plus your specified channel skew value.

At this point, we should attempt to explain the non-intuitive polarity of the skew value. Notice the delay element used to represent skew in **Figure 2** has a minus sign in it. This is because the channel skew (that is, probe skew) value historically referred to translating the physical probe point or measurement reference plane. For a voltage wave, $v(t)$ traveling in the positive direction, x with a velocity, c , shifting the waveform backwards in time is equivalent to skewing it forward in location. $v(x,t) = v(x-ct)$. Hence, increasing the skew value decreases the channel's ADC input path delay, which shifts the signal to the left on screen. So, correcting a channel waveform's time position by adding a positive skew value actually decreases its `xOrig` value.

Figure 2 can also help explain the polarity of the skew value found in the Channel Setup dialog boxes. Notice that increasing the skew delay value of a particular channel decreases the delay of that channel's input signal, causing it to arrive at its ADC earlier than the other channels. This makes events on the skewed channel appear as if they occurred earlier in time, or shifted to the left on screen. So, correcting a channel waveform's time position by adding a positive skew value actually decreases its `xOrig` value.

Note that the applied skew corrections can actually change during an acquisition cycle. An acquisition starts with an initial skew value that is determined by the configuration of the oscilloscope and its calibrated correction factors. But after the trigger event occurs and the waveform has been digitized, they are sometimes adjusted to apply horizontal dither and/or JitterFree trigger correction.

Also note that the memory controller uses the pre-acquisition skew value to optimize which waveform samples for each channel are stored in memory prior to each acquisition cycle. For example, assume you digitize channels 1 and 2 over the same 1 ns window of time by setting their channel skew values to zero and performing a single trigger. Next, you change the channel skew value of channel 2 to 5 ns to correct for some external skew that you know to exist between the sources of the two signals. Although the two signals may be properly aligned relative to simultaneous events between the signals, the two sampled waveform records will no longer overlap on screen. That is because the skew was applied after the signals were captured. Perform another signal acquisition. Now the two signals are properly aligned relative to simultaneous events between the signals and the two waveform records will again overlap on screen.

External Skew

The MultiScope MATLAB software uses an undocumented channel skew control (`:chan<n>:ext:skew <value>`) to apply a time skew to each individual channel that corrects for MultiScope-specific system skew errors. This external skew acts exactly like the normal channel skew (`:chan<n>:probe:skew <value>`) control, whereby the total applied skew is the sum of the channel skew and the external skew values. This lets you apply your own deskew independently from the calibrated system deskews applied by the MultiScope software.

In the case of correcting for frame lag, the previously discussed `FrameLag` values are used to deskew each Follower's horizontal position by contributing to the composite `Follower.F{iF}.(ChanStr).PreAcqSkew` value of each Follower. The `PreAcqSkew` values are then applied to the Follower's external skew controls prior to digitizing the waveforms.

Internal Skew

One complication that arises when synchronizing multiple oscilloscopes together is that the Leader automatically corrects its input channels for trigger path delay changes when its trigger configuration changes, but the Followers don't automatically get these corrections. For the Followers' input channels to apply the same calibrated trigger path delay corrections, the MultiScope MATLAB software must first read those calibration factors out of the Leader.

The trigger path delay values are read out of the Leader using the internal skew query (`:chan<n>:int:skew?`); however, it is not as easy as reading out the specific values you need. **Table 6** explains how to get the trigger path delay values we need from the total system values returned by the internal skew query. The query returns the current total system skew value used by the specified channel. So, in order to determine the trigger path delay applied to channel 2 when triggering on channel 4 with no hysteresis, you need to set the trigger source to channel 4 and set its sensitivity to "normal" prior to querying the internal skew of channel 2.

As you can see from the table, the internal skew query returns all components of the pre-acquisition skew, including the analog path delay. Analog path delay is an oscilloscope calibration factor that deskews each channel relative to one another (as opposed to trigger path delay that applies to all channels simultaneously). If you examine the actual query return values, you will see that there is also a very large (~100 ns) constant skew term that is applied to all channels under all conditions. While difficult to explain, the constant skew term does not hurt anything as long as it is accounted for. The MultiScope MATLAB software ultimately uses only the relative trigger path delay differences to adjust the Followers' `PreAcqSkew` values.

Table 6 Values returned by the internal skew query

		Displayed Channel			
		Chan1	Chan2	Chan3	Chan4
Trigger Channel	Chan1	TPS(1)	TPS(1) + APS(2) + PS(2) + ES(2) - (APS(1) + PS(1) + ES(1))	TPS(1) + APS(3) + PS(3) + ES(3) - (APS(1) + PS(1) + ES(1))	TPS(1) + APS(4) + PS(4) + ES(4) - (APS(1) + PS(1) + ES(1))
	Chan2	TPS(2) + APS(1) + PS(1) + ES(1) - (APS(2) + PS(2) + ES(2))	TPS(2)	TPS(2) + APS(3) + PS(3) + ES(3) - (APS(2) + PS(2) + ES(2))	TPS(2) + APS(4) + PS(4) + ES(4) - (APS(2) + PS(2) + ES(2))
	Chan3	TPS(3) + APS(1) + PS(1) + ES(1) - (APS(3) + PS(3) + ES(3))	TPS(3) + APS(2) + PS(2) + ES(2) - (APS(3) + PS(3) + ES(3))	TPS(3)	TPS(3) + APS(4) + PS(4) + ES(4) - (APS(3) + PS(3) + ES(3))
	Chan4	TPS(4) + APS(1) + PS(1) + ES(1) - (APS(4) + PS(4) + ES(4))	TPS(4) + APS(2) + PS(2) + ES(2) - (APS(4) + PS(4) + ES(4))	TPS(4) + APS(3) + PS(3) + ES(3) - (APS(4) + PS(4) + ES(4))	TPS(4)
	Aux Trig	TPS(A) + APS(1) + PS(1) + ES(1)	TPS(A) + APS(2) + PS(2) + ES(2)	TPS(A) + APS(3) + PS(3) + ES(3)	TPS(A) + APS(4) + PS(4) + ES(4)
	Cal (Internal)	TPS(C) + APS(1) + PS(1) + ES(1)	TPS(C) + APS(2) + PS(2) + ES(2)	TPS(C) + APS(3) + PS(3) + ES(3)	TPS(C) + APS(4) + PS(4) + ES(4)
<p>Where:</p> <ul style="list-style-type: none"> ▪ TPS() = trigger path delay ▪ APS() = analog path delay ▪ PS() = probe skew (that is, channel skew) ▪ ES() = external skew 					

The MultiScope MATLAB software finds the trigger path delay by subtracting out the other known delays.

Sample Alignment

Throughout this discussion of the the MultiScope MATLAB software implementation, we describe time-shifting waveforms for a variety of reasons. Whether for correcting constant skew errors or variable jitter errors, the magnitudes of the time-shift values can be arbitrary in size. In the absence of further processing, the relative time position of the corrected output waveforms will be correctly represented by their `xOrg` and `xInc` values, but the time location of the sample points across the waveforms will rarely be aligned with one another. In fact, the number of points within each waveform record will likely differ across waveforms as well.

Therefore, the MultiScope MATLAB software provides an additional (optional, using the `AlignSamples` parameter) post-processing step that applies a unique time-shifting FIR filter to each output waveform and then truncates them so that all output waveforms have the same `xOrg` and `Points` values.

Channel Delay Skew

From your perspective, there is a channel skew value that works just like it does in a single oscilloscope application. When the channel skew is set to zero on all channels, you would expect that all of the input channels would be deskewed to their input connectors in a calibrated system. But we have not achieved this yet with what we have discussed so far. This is true for several reasons, but the most significant reason is that all of the calibrations performed in the initialization process except `DeskewFrames` and `DeskewChans` are performed using the dedicated synchronization channels, and not your SUT (signal under test) input channels.

Another skew value called `Follower.F{iF}.ChanDelay.(ChanStr)` applies calibrated channel-to-channel skews between all of the MultiScope waveforms. Like `FrameLag`, `ChanDelay` also gets incorporated into the `PreAcqSkew` value as part of the pre-acquisition oscilloscope configuration. The value is calibrated by running `ConfigMultiScope` with either the `DeskewFrames` or `DeskewChans` flag enabled.

Note that this MultiScope system calibration must be performed after all other initialization deskews have been performed. That means any time you re-run `InitNewConfig`, you have to re-run `DeskewFrames` or `DeskewChans`. Running `InitPowerOn` with jitter correction enabled does not require a new MultiScope deskew calibration, as discussed later.

Timebase Delay Reference Position

The oscilloscope's timebase delay reference position setting, or simply delay reference, refers to the horizontal location on the waveform display grid that corresponds to the oscilloscope's horizontal delay position setting. In all Infiniium software versions prior to 5.0, the delay reference value was limited to three discrete values; left, center, and right. Beginning with version 5.0, the delay reference value was enhanced to allow any integer percentage of the grid from 0% (left) to 100% (right).

Regardless of the user's present delay reference setting, MultiScope always configures its acquisition settings to a delay reference setting of 50% (center). If the user's delay reference setting is something other than 50%, the delay position setting is also modified accordingly, such that after appropriate data record de-skewing and trimming, the resultant acquired waveforms appear as if they were acquired using the user's desired delay reference and delay position settings.

The original revisions of MultiScope used the parameter `Leader.Ref` to represent the system's delay reference value, with valid values of 'LEFT', 'CENT', and 'RIGH'. Newer revisions of MultiScope, that support Infiniium 5.0 and later use two parameters to represent the delay reference value; `Leader.RefStr` and `Leader.Ref`:

- `Leader.RefStr` is a string variable with valid values of 'LEFT', 'CENT', 'RIGH' and 'PERC'.
- `Leader.Ref` becomes an integer value from 0 through 100.

For backward-compatibility, the function input arguments, `Param.Ref` and `Leader.Ref` can be passed into `MultiScope` with string values of 'LEFT', 'CENT', and 'RIGH', or they can be passed in as integer values of 0 through 100.

Jitter Correction

Now comes the difficult part. As mentioned in "**Trigger Out to Trigger In Synchronization**" on page 76, the basic trigger out synchronization method results in a large amount of jitter and drift between the Followers and the Leader. This excessive jitter and drift can be reduced considerably by enabling the jitter correction synchronization process. To be fair, this jitter correction is not really a correction of jitter error, but more an alternate method of determining the X-origins of the Follower waveforms from knowledge of the delay between the oscilloscope's phase-locked sample clocks.

The idea is this. If you phase-lock the sample clocks of all of the oscilloscopes in the system together, then the jitter between the samples of any two waveforms should be very small (at least much smaller than the Trig Out + Aux Trig jitter). The trick though, is to figure out the time delay between the sample points in each waveform relative to those in the other waveforms. Ultimately, this boils down to calculating the X-origin of each Follower waveform from the X-origin of the Leader's waveforms.

One characteristic of the Infiniium oscilloscope architecture that makes this possible is that the oscilloscope's memory controller keeps track of which samples in the waveform are associated with slice-0 of each ADC. The ADC is comprised of a fixed number of sample points that are distributed uniformly across a periodic 4 ns interval. Each of these repeating samples is referred to as an ADC slice. Slice-0 is the first of these slices. Another necessary architecture characteristic is that once the oscilloscope timebases are locked together, the physical time delay between slice-0 on each oscilloscope frame does not change.

So in the simplest terms, if you know which of the sample points within two waveforms are associated with slice-0, and you know the physical delay between slice-0 on the two waveforms, then you can calculate the relative time delay between the two waveforms. Of course, nothing is ever simple.

To be more specific, let's define `LeaderXOrgDelay`, `FollowerXOrgDelay`, `SliceDelay`, and `RefClkSkew`, such that:

$$\text{Follower.XOrg} = \text{Leader.XOrg} - \text{LeaderXOrgDelay} + \text{SliceDelay} + \text{RefClkSkew} + \text{FollowerXOrgDelay}$$

Where:

<code>Follower.XOrg</code>	is the X-origin (time of first sample point in the waveform record) of the Follower waveform being corrected.
<code>Leader.XOrg</code>	is the X-origin (time of first sample point in the waveform record) of the Leader waveform being used as a time reference.
<code>LeaderXOrgDelay</code>	is the time between the reference waveform's X-origin and its first slice-0 sample point.

<code>FollowerXOrgDelay</code>	is the time between the Follower waveform's X-origin and its first slice-0 sample point.
<code>RefClkSkew</code>	is the fractional portion of 250 MHz reference clock periods (4 ns) between the reference waveform's first slice-0 sample point and the Follower's first slice-0 sample point (in seconds).
<code>SliceDelay</code>	is the integer portion of 250 MHz reference clock periods (4 ns) between the reference waveform's first slice-0 sample point and the Follower's first slice-0 sample point (in seconds).

Now, let us dive into each one of these values even deeper.

Leader.XOrg

Although this document refers to the reference waveform's X-origin as `Leader.XOrg`, it is implemented using `wfm{RefWfm}.XOrg` within the actual `CorrectJitter` function.

The MultiScope MATLAB software always uses channel 1 of the Leader as the reference waveform for the SUT calculations. This forces the Leader's channel 1 to always be acquired even if it is not being displayed or returned by the MultiScope MATLAB software.

The MultiScope MATLAB software uses the synchronization channel of the Leader as the reference waveform for the calibration acquisition.

Follower.XOrg

Although this document refers to the Follower waveform's X-origin as `Follower.XOrg`, it is implemented using `wfm{IWfm}.XOrg` within the actual `CorrectJitter` function.

LeaderXOrgDelay

The MultiScope MATLAB software uses the undocumented query `" :wav :s0x? cre1"` to determine the time of the reference waveform's first slice-0 sample point.

Note that this time value usually falls well-before the X-origin of the digitized waveform because a large portion of the final waveform was removed by FIR correction filter truncation.

FollowerXOrgDelay

The MultiScope MATLAB software uses the undocumented query `" :wav :s0x? cre1"` to determine the time of each Follower waveform's first slice-0 sample point.

Note that this time value usually falls well-before the X-origin of the digitized waveform because a large portion of the final waveform was removed by FIR correction filter truncation.

SliceDelay

The MultiScope MATLAB software breaks the delay between slice-0 on the reference waveform and slice-0 on each Follower waveform into an integer component (`sliceDelay`) and a fractional component (`RefClkSkew`). That is because the fractional component is fixed by the phase-locking of the oscilloscope timebases, but the integer part can vary between oscilloscope settings or even between different acquisitions using the same settings. The integer portion changes because the slice-0 samples can fall arbitrarily within the system's target acquisition time window. Note that the `sliceDelay` values are not actually integers, but integer multiples of 4 ns.

TotalLeaderSkew, TotalFollowerSkew

Note that in that the actual `CorrectJitter` function's implementation, there are two additional values, `TotalLeaderSkew` and `TotalFollowerSkew`, used in the calculation of `sliceDelay`. The MultiScope MATLAB software normalizes-out the channel skew and external skew values of each channel from its jitter correction calculations. Doing so makes the jitter correction and its calibration independent of the current channel skew and external skew settings.

Note that `TotalLeaderSkew` and `TotalFollowerSkew` are also used in the `RefClkSkew` calculations to normalize-out the channel skew and external skew settings.

RefClkSkew

The `RefClkSkew` component of the inter-oscilloscope slice-0 delay is also broken into an integer component (`RefClkCoarse`) and a fractional component (`RefClkFine`), but for a different reason than that of `sliceDelay`. Once each oscilloscope is powered on and its Infiniium application software has started, the `RefClkSkew` delay value becomes fixed, and can then be calibrated and reused for subsequent acquisitions. However, every time one of the oscilloscopes in the system is power-cycled or the oscilloscope application is restarted, the `RefClkSkew` value can change by a random integer multiple of 100 ps.

Separating out the `RefClkCoarse` component allows the MultiScope MATLAB software to adjust for system power cycles and application restarts without having to perform a full system re-calibration. Note that `InitNewConfig` performs a full system re-calibration, including the re-calculation of `RefClkFine`, while `InitPowerOn` recalculates only the new `RefClkCoarse` value.

The direct calculation of `RefClkCoarse` simply uses the `xOrg` and `s0x` values without digitizing any calibration signals. MultiScope calculates a unique `RefClkCoarse` value for each Follower channel even though there really only needs to be one for the whole follower. Over worst-case conditions, it is possible for the Trig Out / Aux Trig trigger drift to cause errors in this calculation. So, MultiScope performs a correction to this calculation at the end of `ConfigMultiscope` using the drift correction calibration signals. This correction determines proper frame-wide `RefClkCoarse` change since the previous calibration and applies it to all channels of that Follower. The proper change is determined by verifying looking for any unexpected 100 ps jumps in the drift skew value.

NOTE

The previous section describes using the query `:wav:s0x? cre1` to determine the time between the waveform X-origin and the slice-0 X-origin. The `"cre1"` option is used in that query because it returns the `s0x` value after `JitterFree` and/or horizontal dither have been applied. However, when determining any of the values related to the inter-oscilloscope slice-0 delays, the `"cabs"` option must be used. The `"cabs"` option returns the `s0x` value before `JitterFree` and/or horizontal dither have been applied.

JitterFreeSkew

`JitterFreeSkew` is yet another delay component used in the calculation of the Follower waveform's X-origin from the reference waveform's X-origin. `JitterFreeSkew` results from the use of the uncorrected Trigger Out X-origin values for the `RefClkSkew` calibration.

Drift Correction

Even with jitter correction enabled, the horizontal position of channels within different oscilloscope frames can drift around with temperature (inter-oscilloscope drift). The drift correction synchronization process corrects for this drift by monitoring the inter-oscilloscope drift on a separate dedicated synchronization channel. The idea is that any "time corrections" that keep the synchronization channels aligned will also keep the SUT signals aligned.

The basic correction method is simple. During the initial configuration/calibration, the time delay between the synchronization signals on all oscilloscopes is measured. Then, after each jitter-corrected SUT acquisition, the synchronization signals on all oscilloscopes are measured again. A drift correction value is determined that will return the synchronization signals on each Follower back to their original relative time positions. This same drift correction value is then also applied to all SUT signals within the same Follower frame.

The only aspect of the drift correction process that is mildly complicated is how it measures the time delay between frames. To maximize the sensitivity of the time delay measurement, drift correction specifically measures the phase of the synchronization signal's fundamental spectral frequency component. This is done using the oscilloscope application's built-in FFTPhase waveform math function.

Post-Acquisition Skew

We've already talked about the external skew value set for each oscilloscope channel prior to each acquisition cycle. This was known as the `PreAcqSkew` value. Well, it turns out that the MultiScope MATLAB software sometimes also updates the external skew value in the oscilloscope after each acquisition. This is known as the `PostAcqSkew` value. The MultiScope MATLAB software uses post-acquisition skew adjustment to apply jitter and drift corrections back to each displayed channel waveform in the system so that you can manually adjust each waveform's channel skew on the oscilloscope's displays.

Note that post-acquisition skew adjustments cannot be applied after a drift correction has been performed because the drift correction acquisition destroys the uncorrected SUT waveform data displayed on the oscilloscope. Nor is it even needed for the basic Trigger Out level of synchronization, because the trigger out process does not apply any post-acquisition timing corrections.

Setup/Recall

There are a few important things to know about how setup save and recall is implemented in MultiScope. First of all, there is only one .set file for the entire system configuration and it is always saved and loaded by the Leader. The MultiScope setup file uses the same XML text format as the standard single-oscilloscope setup file, except that it includes additional optional configuration strings for each of the Follower channel settings. All of the system-common settings, like sample rate and interpolation ratio, are already stored in the Leader's setup file.

The MultiScope MATLAB software uses the undocumented remote command (`:mscope:mchannel<n> <config_string>`) to write each Follower's channel settings to the Leader's memory. Then, when the Leader saves its setup, the setup file will contain the current values of the Follower's channel settings. The act of loading setup files into the Leader does not change the Follower's settings. It only updates the Follower channel configuration strings within the Leader. The MultiScope MATLAB software must then read these configuration strings from the Leader and use that information to change the Follower's settings.

The Follower configuration strings are actually a concatenation of MATLAB variable definition commands, such that each configuration string may simply be evaluated by the MultiScope MATLAB software using the `eval(<config_string>)` command. The Follower configuration strings are saved by MultiScope channel number, so `mchannel11`, for example, would refer to Channel 3 on Follower 2.

It is important to clear the Follower configuration strings from the Leader's memory immediately after saving the setup file so that the configuration information does not become stale after subsequent Follower setting changes.

7 Infiniium 90000 X-Series and 90000L Series Oscilloscopes

90000 X-Series and 90000L Series Oscilloscope Requirements / 94

Connections for Synchronization / 95

Connections for Reference Clock Skew Calibration / Drift Correction / 96

Sample Rate Selection and Available Channels / 97

In addition to the oscilloscopes described in the *Keysight MultiScope Hardware Configuration Guide* ("www.keysight.com/find/MultiScope-hw-config"), the MultiScope MATLAB time synchronization software also supports up to two 90000 X-Series or 90000L Series oscilloscopes.

90000 X-Series and 90000L Series Oscilloscope Requirements

The version of Infiniium application software required for the MultiScope MATLAB software runs only on the Windows 7 operating system; therefore, 90000 X-Series or 90000L Series oscilloscopes used in a MultiScope system must have the Windows 7 operating system.

A maximum of two 90000 X-Series or 90000L Series oscilloscopes in a MultiScope system are supported.

Connections for Synchronization

The following two connections are required for synchronization.

- 1 Connect Trig Out from the Leader to Aux Trig on the Follower through a BNC cable.
- 2 Connect Ref Clk Out from the Leader to Ref Clk In on the Follower:
 - Connect the 10 MHz Out from the Leader to 10 MHz In on the Follower though a BNC cable.

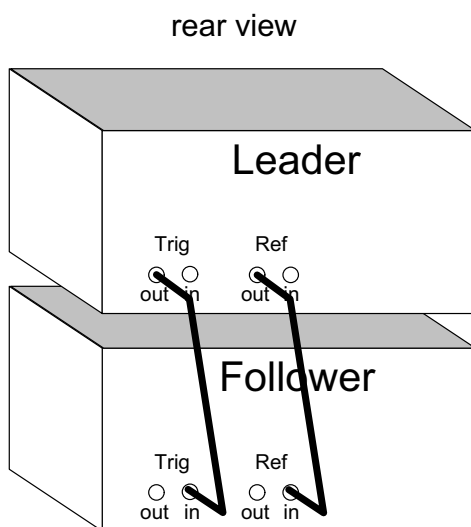


Figure 3 Connections for synchronization

Connections for Reference Clock Skew Calibration / Drift Correction

The following connections are required for MultiScope MATLAB software's reference clock skew calibration. These same connections are also required for the optional drift correction.

- 1 Connect the Follower's Cal Out signal to the Sync In inputs on both oscilloscopes using high-quality SMA cables and a passive power splitter or divider. On 90000 X-Series or 90000L Series oscilloscopes, use channel 4 as the Sync In input.

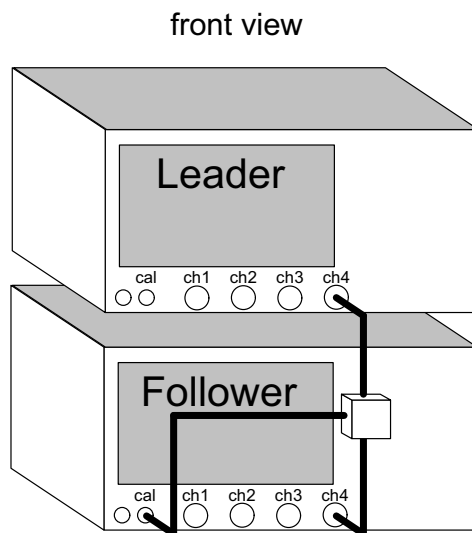


Figure 4 Connections for reference clock skew calibration and drift correction

Sample Rate Selection and Available Channels

When using MultiScope with 90000 X-Series or 90000L Series oscilloscopes, the channel configuration has a higher precedence than the sample rate and will limit the sample rate as you turn on more channels (full channel mode limits the sample rate to 40 GSa/s).

The maximum sample rate of 90000 X-Series or 90000L Series oscilloscopes is available only when no more than one channel of each pair is on (channels 1 and 2 are one pair, channels 3 and 4 are the other pair). When channel 4 is used as the synchronization input (for drift correction), channels 2 and 4 can never be used at the maximum sample rate.

Because the maximum sample rate of the 90000 X-Series or 90000L Series oscilloscopes depends on the number of acquired channels, the `DeskewFrames` and `DeskewChans` scripts use the current maximum sample rate that is available when the deskew operation is initiated.

Index

Symbols

:chan<n>:ext:skew <value> command, 80
:chan<n>:int:skew? query, 81
:chan<n>:probe:skew <value>
command, 80
:mscope:mchannel<n> <config_string>
command, 92
:wav:s0x? cabs query, 89
:wav:s0x? crel query, 87, 89

Numerics

-222,Data out of range errors,
troubleshooting, 70
90000 X-Series oscilloscope support, 93
90000L Series oscilloscope support, 93

A

AcquireMultiscope function, 59
acquisition band width, 37
alignment, sample, 83
AlignSamples parameter, 57, 83
analog path delay, 82
AutoLoadSetup parameter, 54

B

band width setting, override, 51
BW parameter, 55

C

CalFileName parameter, 57
channel delay skew, 84
channel display, 38
channel skew, 82
channel skew, channel-specific setting, 38
channels available, 90000 X-Series, 97
channels available, 90000L Series, 97
channel-specific settings, 38
CloseInterfaces script, 68
coding conventions, software, 64
ConfigMultiscope function, 59
configuration, things that change the, 35
connections for ref clk skew calibration,
90000 X-Series oscilloscope, 96

connections for ref clk skew calibration,
90000L Series oscilloscope, 96
connections for trig out/in and ref clk out/in,
90000 X-Series oscilloscope, 95
connections for trig out/in and ref clk out/in,
90000L Series oscilloscope, 95
customizing functions/scripts, 63

D

Data out of range errors,
troubleshooting, 70
Debug parameter, 57
default user folder, 28
DefaultParams function, 58
DeskewChans script, 43
DeskewChans task, 60
DeskewFrames script, 43
DeskewFrames task, 60
deskewing measured waveforms, 41
DeskewSignals script, 43
DeskewSignals task, 61
device open errors, troubleshooting, 68
Digitize task, 61
DigitizeWfms script, 45
digitizing input signals, 45
Display(n) parameter array, 55
drift correction, 75, 90
drift correction and acquisition update
rate, 50
DriftCorr parameter, 57
DriftMeas.Rate parameter, 50, 57

E

Examples folder, files in, 29
export setup to MATLAB workspace, 51
external skew, 80, 82

F

Follower.XOrg, 86, 87
FollowerXOrgDelay, 87
FrameLag, 77
Frames tab, MultiScope MATLAB GUI, 32
front panel set up, 38
functions, organization of, 59
functions/scripts, customizing, 63

G

getting started, 23

H

HiSLIP LAN connection, 14
horizontal position, 37
horizontal reference, 37
horizontal scale, 37
host (controller) computer
requirements, 11
host (controller) computer software, 12

I

IFig parameter, 58
InitNewConfig script, 35
InitNewConfig task, 60
InitPowerOn task, 60
InitPowerOn.m script, 36
install location, 13
Int parameter (interpolation points), 55
internal skew, 81
interpolation ratio, 37
interpolation ratio setting, override, 51
interrogation of oscilloscope hardware, 73
isolated network, 14

J

jitter correction, 86
JitterCorr parameter, 57
JitterFreeSkew, 89

K

Keysight Connection Expert, 15
Keysight Interactive IO application, 20
Keysight IO Control icon, 15
Keysight IO Libraries Suite software, 13
Keysight_startup script, 28

L

LAN interface, 17
LAN interface for remote connections, 14

Leader.XOrg, 86, 87
 LeaderXOrgDelay, 86, 87
 license required, 10
 LoadAndPlotAcqTime script, 50
 LoadAndPlotJitter script, 48
 LoadAndPlotWfms script, 46
 LoadSetup script, 40
 LoadSetup task, 61

M

MATLAB basics, 26
 MATLAB search path, 28
 MATLAB software, 12
 MeasAcqTime script, 49
 MeasJitter script, 47
 memory depth setting, override, 51
 Multiscope function, 60
 Multiscope function in Examples folder, 29
 MultiScope MATLAB environment, starting, 27
 MultiScope MATLAB GUI, advanced settings, 51
 MultiScope MATLAB GUI, saving/opening projects, 52
 MultiScope MATLAB GUI, starting, 30
 MultiScope MATLAB software installation, 13
 MultiScope MATLAB time synchronization software, 12
 MultiScope MATLAB time synchronization software, product overview, 3

N

N8822A license, 10
 notices, 3
 number of oscilloscopes, specifying, 33
 NumFrames parameter, 33, 39, 65

O

Offset(n) parameter array, 55
 oscilloscope architecture, 72
 oscilloscope requirements, 10
 oscilloscope software version required, 10
 oscilloscopes, initialization, 35
 oscilloscopes, set up, 37
 override oscilloscope acquisition settings, 51

P

parameter passing, 65
 parameters, channel-specific, 55
 parameters, default, 58
 parameters, MultiScope configuration, 54

parameters, operation, 56
 parameters, setup file, 54
 parameters, system-common, 55
 parameters, waveform plot, 57
 Points parameter (memory depth), 55
 PostAcqSkew, 91
 post-acquisition skew, 91
 PreAcqSkew, 91
 probe skew, 82

R

RealEdge waveform numbering, 62
 recall, MultiScope, 92
 Ref Clk Out to Ref Clk In synchronization, 74
 RefClkSkew, 87, 88
 reference clock skew calibration, 35, 36, 74
 remote command errors, troubleshooting, 69
 remote connections between host computer and oscilloscopes, 14
 remote connections, verifying, 15
 Remote Desktop Connection, 25
 requirements, 90000 X-Series oscilloscopes, 94
 requirements, 90000L Series oscilloscopes, 94

S

sample alignment, 83
 sample rate, 37
 sample rate for MultiScope system, 37
 sample rate selection, 90000 X-Series, 97
 sample rate selection, 90000L Series, 97
 sampling rate setting, override, 51
 SaveSetup script, 40
 SaveSetup task, 61
 Scale(n) parameter array, 55
 scripts in Examples folder, 29
 scripts/functions, customizing, 63
 search path, MATLAB, 28
 set up (MultiScope), ways to, 38
 SetScopeAndDigitize script, 46
 setup files, MultiScope, 39
 setup, MultiScope, 92
 SetupFileName parameter, 40, 54
 skew, 78
 Skew(n) parameter array, 55
 SliceDelay, 87, 88
 software coding conventions, 64
 software version, oscilloscope, required, 10
 SRate parameter (sample rate), 55
 synchronizing multiple oscilloscopes, 74
 SysConfig script, 33
 system-common settings, 37

T

theory of operation, 71
 timebase delay reference position, 85
 Title parameter, 58
 TotalFollowerSkew, 88
 TotalLeaderSkew, 88
 Trigger Out synchronization, 74, 76
 trigger path delay, 82
 trigger settings, 37
 troubleshooting, 67

U

UltraVNC Viewer, 25
 USB 3.0 interface for remote connections, 14
 User_startup.m script, 28

V

vertical offset, 38
 vertical scale, 38
 VISA addresses of oscilloscopes, specifying, 33
 VISA addresses, finding, 33

W

waveform numbering, 61

X

XMult parameter, 58
 XUnits parameter, 58

Z

ZeroskewSignals script, 44
 ZeroskewSignals task, 61