User Defined Function

User's and Programmer's Guide

KEYSIGHT TECHNOLOGIES

Notices

Revision

Version 11.30.00203

Edition

July 2022

Available in electronic format only

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

Trademarks

Infineon is a registered trademark of Infineon Technologies AG. The Infiniium mark is used with the kind approval of Infineon Technologies, AG.

MIPI® service marks and logo marks are owned by MIPI Alliance, Inc. and any use of such marks by Keysight Technologies is under license. Other service marks and trade names are those of their respective owners.

PCI-SIG®, PCIe®, and the PCI Express® are US registered trademarks and/or service marks of PCI-SIG.

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 License

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

U.S. Government Rights

The Software is "commercial computer software," as defined by Federal Acquisition Regulation ("FAR") 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement ("DFARS") 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at www.keysight.com/find/sweula. The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

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.

User Defined Function—At a Glance

The User Defined Function lets you perform your own mathematical transformations on the captured waveform data using MATLAB software. It adds a menu item entry into the Operator pull-down menu of the Infiniium Math dialog box.

In This Book

This manual describes the procedures required by the User Defined Function in more detail.

- Chapter 1, “Installation,” starting on page 7 shows how to install and license the User Defined Function application software (if it was purchased separately) and information on system requirements.
- Chapter 2, “Overview,” starting on page 11 shows how to install your User Defined Function and gives a brief overview of how it is used.
- Chapter 3, “MATLAB Script File,” starting on page 15 contains information on how to create a MATLAB script.
- Chapter 4, “XML File,” starting on page 29 contains information on how to create an XML file.
- Chapter 5, “GPIB Function Commands,” starting on page 35 contains information on the GPIB remote commands used to program your user defined function.
- Chapter 6, “Deep User Defined Functions,” starting on page 41 contains information on the ability to perform file based IO for user defined functions.

Contents

1 Installation

Requirements

Before you can use your own mathematical transformations, you need:

- Infiniium version 5.50 or later oscilloscope software for the 8000 or 80000B Series oscilloscopes. Infiniium version 1.20 or later oscilloscope software for the 90000A Series, 90000 X-Series, or 9000A Series oscilloscopes.
  (90000 Q-Series, Z-Series, V-Series, and S-Series oscilloscopes were released with an Infiniium version later than 1.20.)
- MATLAB software.

Infiniium Oscilloscope Software

The User Defined Function option requires Infiniium Oscilloscope software version 5.50 or later for the 8000 or 80000B Series oscilloscopes and software version 1.20 or later for the 90000A Series, 90000 X-Series, or 9000A Series oscilloscopes. (90000 Q-Series, Z-Series, V-Series, and S-Series oscilloscopes were released with an Infiniium version later than 1.20.)

You can find out which version of software that the Infiniium Oscilloscope is running by looking in the **Help > About Infiniium...** dialog box. If you do not have the correct version of Infiniium Oscilloscope software, go to www.keysight.com and search for Infiniium oscilloscope software to download the current version.

MATLAB Software

The User Defined Function requires MATLAB to be installed before you can use your mathematical transformation. You will have to purchase this software yourself.

NOTE Two of the example MATLAB script files that come with the User Defined Function software, Butterworth and LFE, also require the Signal Processing Toolkit.

2 Overview

User Defined Function

The User Defined Function provides you with a method of adding your own mathematical transform as a math operator entry in the Infiniium Math dialog box. The captured waveform data is passed to the MATLAB script that you create to process the waveform data. The processed waveform data is passed back to the oscilloscope to be displayed as a function in the waveform viewing area. You will also create an XML file that describes to Infiniium the interface used for your mathematical transformation.

Installing Your User Defined Function

To install your user defined function on the oscilloscope::

1. Create an XML file (.xml file) for your mathematical transformation (see "XML File Format" on page 30).
2. Create a MATLAB script file (.m file) implementing your mathematical transformation (see "Example MATLAB Script" on page 24).
3. Put both files in the following directory on the oscilloscope:
   - For Windows XP: C:\Documents and Settings\All Users\Documents\Infiniium\User-Defined Functions
   - For Windows 7: C:\Users\Public\Public Documents\Infiniium\User-Defined Functions
4. If the XML file is new or changed, then restart the oscilloscope.

3 MATLAB Script File

MATLAB Interface

The mathematical transformation script, that you write, performs the operations on the waveform data. Several variables can be passed to your script from the oscilloscope. These variables are determined by the field set by the <FunctionType> tag in your XML file. The fields are:

<FunctionType> tag field:	For the variables that can be used when this type is selected, see:
1 Source	"One-Source Operator Variables" on page 16
2 Source	"Two-Source Operator Variables" on page 17
Clock Data	"Clock Data Operator Variables" on page 18
Sampled Clock Data	"Sampled Clock Data Operator Variable" on page 19

One-Source Operator Variables

For a 1 Source operator, the input variables available are:

- SrcXInc
- SrcXFirst
- SrcYScale
- SrcYMiddle
- SrcData, or SrcDataFile when <FileIO> is 'true'.
- SrcXDispScale
- SrcXDispFirst
- SrcYDispScale
- SrcYDispMiddle
- SrcBandwidth
- SrcSampleRate
- Control1
- Control2
- Control3
- Control4
- Control5
- Control6
- Reset
- FnNumber

For Segmented acquisitions:

- NumSegments
- SegmentIndex
- SegmentTime

Before exiting the script, the following output variables may be set:

- FnXInc
- FnXFirst
- FnYScale
- FnYMiddle
- FnData (this is the only output variable that must be set before exiting the script), or FnDataFile when <FileIO> is 'true'.
- FnAutoXDispScale
- FnAutoXDispFirst
- FnAutoYDispScale
- FnAutoYDispMiddle

For definitions of these variables, see "Variable Definitions" on page 21.

Two-Source Operator Variables

For a 2 Source operator, the input variables available are:

- Src1YScale
- Src1YMiddle
- Src1Data, or Src1DataFile when <FileIO> is 'true'.
- Src2YScale
- Src2YMiddle
- Src2Data, or Src2DataFile when <FileIO> is 'true'.
- Src1YDispScale
- Src1YDispMiddle
- Src2YDispScale
- Src2YDispMiddle
- Control1
- Control2
- Control3
- Control4
- Control5
- Control6

- Reset
- FnNumber
- Src1XInc
- Src1SampleRate
- Src1Bandwidth

For Segmented acquisitions:

- NumSegments
- SegmentIndex
- SegmentTime

Before exiting the script, the following output variables may be set:

- FnYScale
- FnYMiddle
- FnData (this is the only output variable that must be set before exiting the script), or FnDataFile when <FileIO> is 'true'.
- FnAutoYDispScale
- FnAutoYDispMiddle
- FnXInc
- FnXOrg

For definitions of these variables, see "Variable Definitions" on page 21.

Clock Data Operator Variables

For a Clock Data operator, the input variables available are:

- SrcXInc
- SrcXFirst
- SrcYScale
- SrcYMiddle
- SrcData, or SrcDataFile when <FileIO> is 'true'.
- ClockData
- SrcXDispScale
- SrcXDispFirst
- SrcYDispScale
- SrcYDispMiddle
- Control1
- Control2

- Control3
- Control4
- Control5
- Control6
- Reset
- FnNumber

Before exiting the script, the following output variables may be set:

- FnXInc
- FnXFirst
- FnYScale
- FnYMiddle
- FnData (this is the only output variable that must be set before exiting the script), or FnDataFile when <FileIO> is 'true'.
- FnAutoXDispScale
- FnAutoXDispFirst
- FnAutoYDispScale
- FnAutoYDispMiddle

For definitions of these variables, see "Variable Definitions" on page 21.

Sampled Clock Data Operator Variable

For a Sampled Clock Data operator, the input variables available are:

- SrcYScale
- SrcYMiddle
- SrcXDispScale
- SrcXDispFirst
- SrcYDispScale
- SrcYDispMiddle
- SrcData, or SrcDataFile when <FileIO> is 'true'.
- ClockData
- Control1
- Control2
- Control3
- Control4
- Control5
- Control6

- Reset
- FnNumber

Before exiting the script, the following output variables may be set:

- FnYScale
- FnYMiddle
- FnAutoXDispScale
- FnAutoXDispFirst
- FnAutoYDispScale
- FnAutoYDispFirst
- FnData (this is the only output variable that must be set before exiting the script), or FnDataFile when <FileIO> is 'true'.

For definitions of these variables, see "Variable Definitions" on page 21.

Variable Definitions

The following table shows the list of variables and their definitions.

Variable	Definition
Annotation1 Annotation2 Annotation3	If the Annotation1, Annotation2, or Annotation3 variables are defined, the text contained in the variable is shown as an annotation on the oscilloscope screen. Annotations work just like normal screen annotations. The annotation must be displayed in the grid area but can be placed manually in the user interface wherever you want. These annotations can also be customized in the user interface to change their color, size, etc. Annotations contain text strings, but you can convert MATLAB floating point values to strings (with an appropriate number of digits, units, etc.) using MATLAB functions. Although MATLAB supports wide-characters, Infiniium converts the string returned to a standard 8 bit character array. See "Example: Annotations" on page 26.
FnXInc	This would typically be the time between samples returned by your function. Of course if the units are not time, it would be in whatever units you set up in the XML file. If your MATLAB function does not change the time between points, set FnXInc = SrcXInc.
FnXFirst	This would typically be the time of the first sample returned. Of course if the units are not time, it would be in whatever units you set up in the XML file. If your MATLAB function does not shift the input data, you may set FnXFirst = SrcXFirst.
FnYScale	This would typically be the allowable voltage range of your samples. Of course if the units are not voltage, it would be in whatever units you set up in the XML file. Infiniium stores waveforms internally as 16 bit arrays. FnData values that are returned will be quantized into a 16 bit integer. The FnYScale indicates the range of allowed data values. Any data values returned outside of this scale will be indicated as clipped. The automatic vertical scale of the function in the user interface will be set to this scale divided by the number of vertical divisions (8). If your MATLAB function does not change the gain or vertical scale of the input data, you can set FnYScale =SrcYScale.
FnYMiddle	This would typically be the voltage at the center of the FnYScale. Of course if the units are not voltage, it would be in whatever units you set up in the XML file. The automatic vertical offset of the function in the user interface will be set to this value. If your MATLAB function does not change the gain or vertical scale of the input data, you can set FnYMiddle = SrcYMiddle.
FnData	This is an array of values typically in voltage. FnData values will be quantized into 16 bit integer values according to the FnYScale and FnYMiddle values. Any returned values outside of these limits will be indicated as clipped. The number of points in the function waveform will correspond to the number of points in the FnData array. If a Not A Number (NaN) value is returned, then a hole will be represented in the function. If a value outside of the FnYScale range is returned then a clip high or low value will be represented in the function. If your MATLAB script does not need to return any data, set 'FnData = 0' in your script.
FnDataFile	Used when the XML file's <FileIO> is 'true'. This variable is a string that is the filename and path to the files that contain function output data. The function data files are binary files of type 'double'. See Chapter 6, “Deep User Defined Functions,” starting on page 41.
FnAutoXDispScale	This specifies the automatic horizontal scale of the function.

Variable	Definition
FnAutoXDispFirst	This specifies the automatic horizontal position (left side of screen) of the function.
FnAutoYDispScale	This specifies the automatic vertical scale of the function.
FnAutoYDispMiddle	This specifies the automatic vertical position of the function.
FnNumber	This specifies the value (1-4) indicating the function number that is being computed. This may be useful if data is preserved between calls to your MATLAB script and you want to allow multiple versions of your function to run simultaneously. You may want to create an array of values indexed by FnNumber.
SrcXDispScale	This specifies the horizontal display of the source.
SrcXDispFirst	This specifies the horizontal position (left of screen) of the source.
SrcYDispScale	This specifies the vertical display scale of the source.
SrcYDispMiddle	This specifies the vertical middle of the display of the souce.
SrcXInc	This would typically be the time between samples of the source waveform. Of course the source waveform may not be in units of time.
Src1YDispScale	This specifies the vertical display scale of source 1.
Src1YDispMiddle	This specifies the vertical middle of the display of source 1.
Src2YDispScale	This specifies the vertical display scale of source 2.
Src2YDispMiddle	This specifies the vertical middle of the display of course 2.
SrcXFirst	This would typically be the time of the first sample of the source waveform. Of course the source waveform may not be in units of time.
SrcYScale Src1YScale Src2YScale	This would typically be the full scale voltage range of the source waveform. The source waveform may not achieve the full scale voltage range, particularly if it is not across the full screen. Of course the source waveform may not be in units of voltage.
SrcYMiddle Src1Middle Src2Middle	This would typically be the voltage at the center of the SrcYScale and the offset of the source channel. Of course the source waveform may not be in units of voltage.
SrcData Src1Data Src2Data	This is a double array of values typically in voltage corresponding to the input waveform. Use length(SrcData) to get the length of the SrcData array. If a value is clipped high, the value 1.7976931348623158e+308 (DBL_MAX) will be passed. If a value is clipped low, the value -1.7976931348623158e+308 (-(DBL_MAX)) will be passed. If there is a hole in the data a Not A Number (NaN) value will be passed. Generally it is not necessary to check for these values unless operating on functions which can generate them or equivalent time data.
SrcDataFile Src1DataFile Src2DataFile	Used when the XML file's <FileIO> is 'true'. These variables are strings that are the filename and path to the files that contain source data. The source data files are binary files of type 'double'. See Chapter 6, “Deep User Defined Functions,” starting on page 41.
SrcBandwidth	This specifies the bandwidth.
Src1Bandwidth	This specifies the bandwidth. (Used with a two-source UDF.)

Variable	Definition
SrcSampleRate	This specifies the sample rate.
Src1SampleRate	This specifies the sample rate. (Used with a two-source UDF.)
ClockData	This is a double array of clock times. The clock recovery algorithms should be set up from the scope user interface. There are fewer clock times than SrcData points.
Control1	This is the user value of the first user interface control. If no controls were set up in the XML file, then this value is undefined. For Double, Integer and Enumeration type controls, this is a double variable representing the value entered. For Enumeration type controls, it is the 1 based index into the selection list as ordered by the XML file. For example, if the 5th selection was selected, the value 5 will be passed. For String type controls, this is a string array containing the string that the user typed into the control. If the string contains MATLAB variable expressions, eval(Control1) will parse them.
Control2	This is the user value of the second user interface control. If the second control was not set up in the XML file, then this value is undefined. See Control1 for more information.
Control3	This is the user value of the third user interface control. If the third control was not set up in the XML file, then this value is undefined. See Control1 for more information.
Control4	This is the user value of the fourth user interface control. If the fourth control was not set up in the XML file, then this value is undefined. See Control1 for more information.
Control5	This is the user value of the fifth user interface control. If the fifth control was not set up in the XML file, then this value is undefined. See Control1 for more information.
Control6	This is the user value of the sixth user interface control. If the sixth control was not set up in the XML file, then this value is undefined. See Control1 for more information.

matlababort	This is the user abort variable that can be check to determine if a user has chosen to abort the function. The abort dialog box is created by including the <CheckAbort> tag in your xml file.
	The matlababort variable should be checked in any of the areas of the MATLAB script, such as loops, that may take a long time to compute. An example entry is:
	`If matlababort == 1`
	`  return`
	`end`

Reset	This is a double variable indicating that the source waveform's scale has changed. The passed values are 0 or 1. The value 1 indicates a change. This variable is helpful if accumulation is occurring in MATLAB and needs to reset when the source scale changes.
NumSegments	Total number of segments acquired.
SegmentIndex	Index (0 based) of currently processed segment.
SegmentTime	Time tag of the current segment.

Example MATLAB Script

```
%=======================================================================
% Keysight Technologies
% MATLAB filter for Infiniium
% ----------------------------------------------------------------------
% The Infiniium/MATLAB interaction happens in three steps:
% 1) Infiniium pushes a number of variables into the MATLAB workspace,
% including the source waveform.
% 2) Infiniium executes this script.
% 3) Infiniium pulls a number of variables out of the MATLAB workspace,
% including the output waveform.
% ----------------------------------------------------------------------
% The input variables available for a single source function are:
% SrcXInc: The spacing of the values of SrcData in time.
% SrcXFirst: The time of the first value of SrcData.
% SrcYScale: The potential full scale voltage range of the source wavefo
rm.
% SrcYMiddle: The voltage at the center of the voltage range for the sou
rce
% waveform.
% SrcData[]: The input waveform. Values are typically in volts.
% Control1: The current value of the user defined control 1 in the user
% interface.
% Control2: The current value of the user defined control 2 in the user
% interface.
% Reset: This variable will have the value 1 when the source waveform's
% scale has changed or if clear display has been pressed.
% ----------------------------------------------------------------------
% The output variables are:
% FnXInc: The spacing of the values of FnData in time.
% FnXFirst: The time of the first value of FnData.
% FnYScale: The potential full scale voltage range of the function wavef
orm.
% FnYMiddle: The voltage at the center of the voltage range for the func
tion
% waveform.
% FnData[]: The output waveform.
%=======================================================================
%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
% This example script will be overwritten when the Infiniium Software is
% upgraded. If you wish to modify it, make a copy first.
%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
%=======================================================================
% This script applies a butterworth filter to the % input waveform.
%=======================================================================
% Filter order:
N = 2;
% Cutoff frequency:
BitRate = Control1;
Fc = BitRate/2;
% Constant output variables
Gain = 1.0;
FilterDelay = N * 0.12 / Fc;
FilterWidth = N / Fc;
```

```
% Create the filter
Wc = 2 * SrcXInc * Fc;
[B,A] = butter(N,Wc);
% Perform the filtering
FnData = filter(B,A,SrcData);
% Remove group delay
FnXFirst = SrcXFirst - FilterDelay;
FnXInc = SrcXInc;
FnYScale = SrcYScale * Gain;
FnYMiddle = SrcYMiddle;
```

Example: Annotations

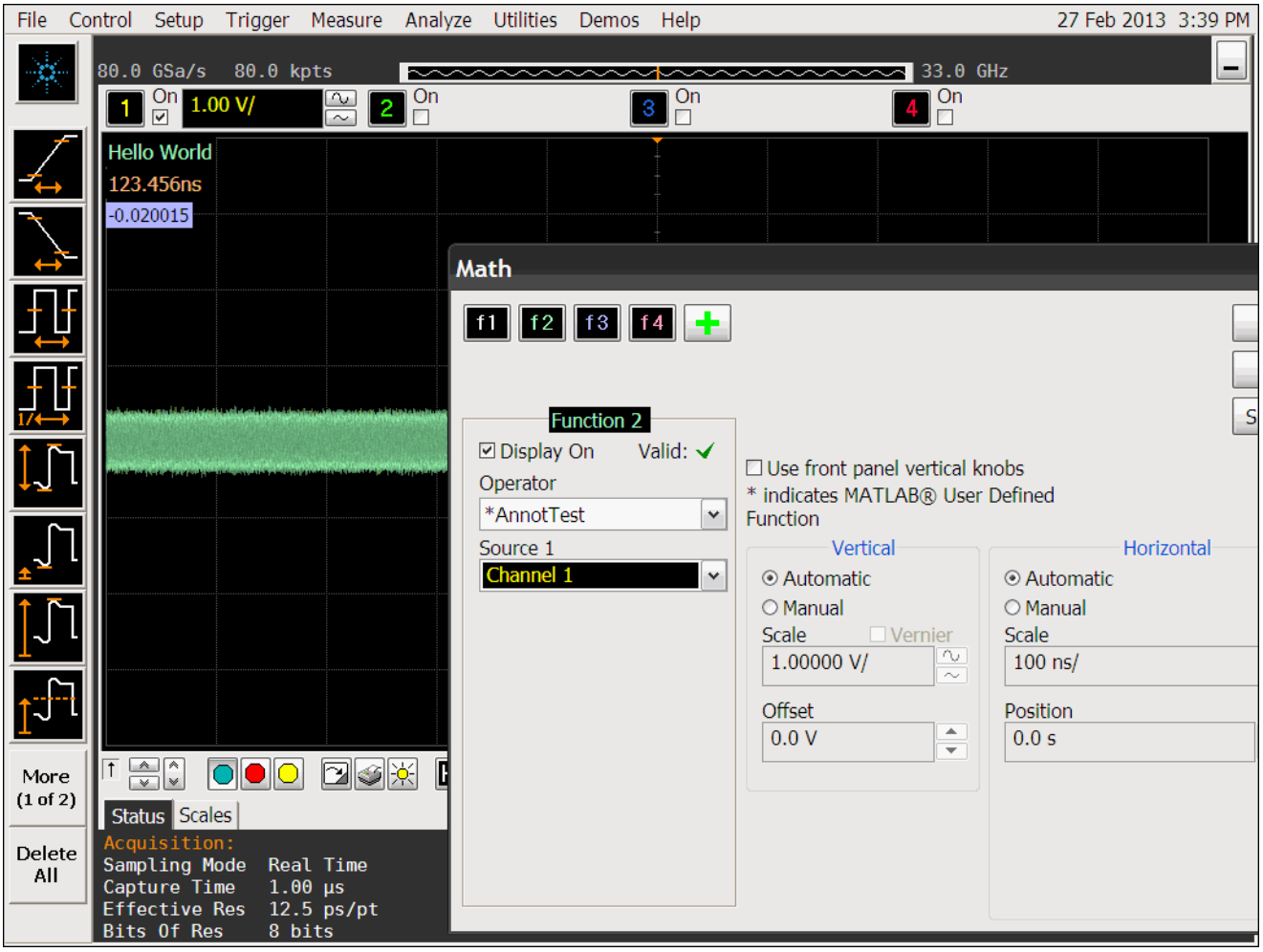
MATLAB Script File

```
FnXFirst = SrcXFirst;
FnAutoYDispScale = SrcYDispScale;
FnAutoYDispMiddle = SrcYDispMiddle;
FnData = SrcData;
Annotation1 = 'Hello World';
Annotation2 = [num2str(123.456), 'ns'];
Annotation3 = num2str(SrcData(1));
```

XML File

```
<?xml version="1.0" encoding="utf-8"?>
<Functions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="Functions.xsd"
           Version="1">
    <Function>
        <Name>AnnotTest</Name>
        <Abbreviation>AT</Abbreviation>
        <MATLABName>AnnotTest</MATLABName>
     </Function>
</Functions>
```

Result


File Control Setup Trigger Measure Analyze Utilities Demos Help
27 Feb 2013 3:39 PM
80.0 GSa/s 80.0 kpts
33.0 GHz
1 On 1.00 V/
2 On
3 On
4 On
Hello World
123.456ns
-0.020015
Math
f1 f2 f3 f4
Function 2
Display On Valid:
Operator
*AnnotTest
Source 1
Channel 1
Use front panel vertical knobs
* indicates MATLAB® User Defined Function
Vertical
Automatic
Manual
Scale
Vernier
1.00000 V/
Offset
0.0 V
Horizontal
Automatic
Manual
Scale
100 ns/
Position
0.0 s
More (1 of 2)
Delete All
Status Scales
Acquisition:
Sampling Mode Real Time
Capture Time 1.00 µs
Effective Res 12.5 ps/pt
Bits Of Res 8 bits

4 XML File

XML File Format

The XML file provides the framework for your mathematical transformation MATLAB script and describes to the User Defined Function the necessary information to create the user interface. This file must be placed in the proper directory:

- For the 8000 Series and 80000B Series oscilloscopes: C:\SCOPE\MATLAB
- For the 90000A Series or 9000 Series oscilloscopes with Windows XP: C:\Documents and Settings\All Users\Documents\Infiniium\User-Defined Functions
- For the 90000A Series, 9000 Series, 90000 X-Series, 90000 Q-Series, Z-Series, and S-Series oscilloscopes with Windows 7: C:\Users\Public\Public Documents\Infiniium\User-Defined Functions

The maximum size of the file is 4096 characters.

The file contains tags and field values. The following is an example of the file format.

```
<?xml version="1.0" encoding="utf-8"?>
<Functions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="Functions.xsd" Version="1">
   <Function>
      <Name>Butterworth</Name>
      <Abbreviation>BTR</Abbreviation>
      <MATLABName>Butterworth</MATLABName>
      <FunctionType>1 Source</FunctionType>
      <FileIO>false</FileIO>
      <SourceWindow>All</SourceWindow>
      <XUnits>Same</XUnits>
      <YUnits>Same</YUnits>
      <Control Name = "Data Rate">
         <Double>
            <Min>1e9</Min>
            <Max>10e9</Max>
            <Resolution>1000</Resolution>
            <Units>bitspersec</Units>
            <Default>2.5e9</Default>
         </Double>
      </Control>
      <Control Name = "Low Pass">
         <Enumeration>
            <Default>Off</Default>
            <Selection>Off</Selection>
            <Selection>On</Selection>
         </Enumeration>
      </Control>
   </Function>
</Functions>
```

Tags for XML File

All tags in the XML file have a starting and terminating tag. The terminating tag has a forward slash. For example, the terminating tag for the <Function> tag is the </Function>. The tags can be in any order except where indicated otherwise. Also, the tags are case sensitive.

The following table shows a list of all of the tags and their descriptions.

Tag	Description	Required Field?
<Functions Version = "1">	This is the root tag. All XML files must have one and only one <Functions> tag. The <Functions> block can contain one or more <Function> blocks. The Version attribute is optional but when used, must be 1.	Yes
<Function>	This tag defines a MATLAB math function. One or more Function tags can be used within a Functions tag. Up to 20 user-defined functions are allowed. They can all be presented in a single file or in individual files. All .xml files in c:\scope\MATLAB are parsed.	Yes
The following tags must be contained in the <Function> block.		
<Name>	This describes the name of the math operator that will appear in the user interface. Infiniium will display from 1 to 16 character names. Names with more than 16 characters in length will be truncated. This is also the string that is passed to the GPIB :FUNCtion<N>:MATLab:OPERator command to select this function.	Yes
<Abbreviation>	This tag is the abbreviation for the operator that will appear in the user interface. Infiniium will display as much of the abbreviation as it can. From 1 to 6 characters are allowed.	Yes
<MATLABName>	This tag lists the filename of the MATLAB script without a suffix. The MatLabName can be up to 24 characters.	Yes
<CheckAbort>	This tag produces an abort dialog box after a specified number of acquisition points have been passed to your MATLAB script. This provides a way for the user to cancel the function if it is taking to long to finish. The following is an example where after 500,000 points are passed a dialog box is produced. `<CheckAbort>500000</CheckAbort>` There also needs to be an entry in the MATLAB script file in any section that is likely to take a lot of time where a user may want to cancel the function.	No

Tag	Description	Required Field?
<FunctionType>	This tag determines the type of math function. This can be one of the following values. ▪ 1 Source ▪ 2 Source ▪ Clock Data ▪ Sampled Clock Data If this field is omitted, the value 1 Source is used. For 1 Source types, your function should conform to the single source function interface. For 2 Source types, your function should conform to the two source function interface. For Clock Data types, your function should conform to the clock data function interface.	
<FileIO>	When 'true', file based IO is used instead of memory based IO. File based IO is useful when processing deep waveforms (greater than 8M points). See Chapter 6, “Deep User Defined Functions,” starting on page 41.	
<SourceWindow>	This tag determines how much data is passed to your mathematical transformation function. Valid values are All or Display. If All is selected, all data in the source waveform record will be passed to MATLAB. If Display is selected, only the portion of the source waveform that is on screen will be passed to MATLAB. If this tag is omitted, Display is assumed.	
<XUnits>	This tag allows you to select the units for the horizontal axis of your function. If this field is omitted, same units are assumed. same units uses the units that the source is using. The following is a list of valid units. same, Volt, second, meter, inch, Ohm, decibel or dB, Hertz, Ampere, Watt, percent, ratio, unknown, constant, sample, point, division, dBm, dBj, hour, waveform, hits, bit, gain, feet, Henry, Farad, minute, degreeC, degreeF, UI, bitspersec, dBc, radian, degree The units are not case sensitive.	
<YUnits>	This tag allows you to select the units for the vertical axis of your function. If this field is omitted, same units are assumed. same units uses the units that the source is using. The following is a list of valid units. same, Volt, second, meter, inch, Ohm, decibel or dB, Hertz, Ampere, Watt, percent, ratio, unknown, constant, sample, point, division, dBm, dBj, hour, waveform, hits, bit, gain, feet, Henry, Farad, minute, degreeC, degreeF, UI, bitspersec, dBc, radian, degree The units are not case sensitive.	
<Source1Type>, <Source2Type>	May be Digital, Bus, Channel, Function, Memory.	
<Source1Title>, <Source2Title>	String title of the label above the source control.	

Tag	Description	Required Field?
<Interpolate>	false if SrcData should not be interpolated.	
<Variable>	Pass a value from the XML file into the MATLAB script. For example <Variable Name = "Bandwidth" Value = "20e9" /> will pass a variable Bandwidth assigned the value 20 GHz to the MATLAB script.	
<Control Name="CtrlName">	This tag defines a control block. Zero, one, two, or three control blocks are allowed for each function. The control block allows the user interface entry associated with your function to be passed to your MATLAB function. This allows users to pass values from the user interface to your MATLAB function. The controls will be presented in the scope user interface in the order presented here. The following tags are all contained in the <Control> block. The Name attribute is the name of the control that shows up in the user interface. As much of the name as possible will be displayed. The name can be up to 24 characters. There are four types of controls. ▪ Double – This control offers a range of floating point values. ▪ Integer – This control offers a range of integer values. ▪ Enumeration – This control offers the user to select from a list of selections. Up to 10 selections can be offered. ▪ String – This control allows the user to input a string. ▪ File – This control is similar to the String control, but brings up a file browse dialog in the user interface.	
<Double>		
<Min>	This is the minimum inclusive double value that the user can enter.	Yes
<Max>	This is the maximum inclusive double value that the user can enter.	Yes
<Resolution>	This is the double resolution and increment for the up/down stepper.	Yes
<Default>	This is the default value for the control. The default value is applied whenever the user selects your function operator. The default should be between min and max.	Yes
<Units>	This is the units associated with the control. The <XUnits> shows a list of valid unit values except that the value same should not be used.	Yes
<Integer>		
<Min>	This is the minimum inclusive integer value that the user can enter.	Yes
<Max>	This is the maximum inclusive integer value that the user can enter.	Yes
<Resolution>	This is the integer resolution and increment for the up/down stepper. The default resolution is 1.	
<Default>	This is the default value for the control. The default value is applied whenever the user selects your function operator. The default should be between min and max.	Yes

Tag	Description	Required Field?
<Units>	This is the units associated with the control. The <XUnits> shows a list of valid unit values except that the value same should not be used.	Yes
<Enumeration>		
<Default>	This is the default value for the control. The default value is applied whenever the user selects your function operator. The default should be the same value as one of the selections.	Yes
<Selection>	This tag contains a selection string for an Enumeration type control. Up to 10 selections are available. The selections will be presented in the order listed here. The value that gets passed to your MATLAB script is the 1 based index of the selection. For example, if the user selects the 5th selection, the value 5 will get passed to MATLAB. At lease one selection must be provided for an Enumeration type control.	Yes
<String>	The string control is particularly powerful as MATLAB variables can be entered and evaluated in MATLAB. This allows a large number of variable values to be passed into MATLAB. The user can enter a string of up to 128 characters.	
<Default>	This is the default string for the control.	
<File>	This control is similar to the String control, but brings up a file browse dialog in the user interface.	
<Default>	This is the default filename for the control.	

5 GPIB Function Commands

The FUNCtion subsystem defines functions 1 to 16. These FUNCtion commands and queries are implemented in the Infiniium oscilloscopes.

:FUNCtion<F>:MATLab

Command `:FUNCtion<F>:MATLab <operand>[,<operand>]`

The :FUNCtion<F>:MATLab command sets the operand(s) for these user-defined functions:

- Butterworth
- FIR
- LFE
- RTEye
- SqrtSumOfSquare

And these InfiniiSim functions:

- InfiniiSim 2 Port
- InfiniiSim 4 Port 1 Src
- InfiniiSim 4 Port CM
- InfiniiSim 4 Port Diff
- InfiniiSim 4 Port Src1
- InfiniiSim 4 Port Src2

<F> An integer, 1-16, representing the selected function.

<operand> {CHANnel<N> | DIFF<D> | COMMonmode<C> | FUNCtion<F> | EQUalized<L> | WMEMory<R> | MTRend | MSPectrum | XT<X> | PNOise | INPut | CORRected | ERRor | LFPR}

See the discussion of possible operands in the introduction to “Function Commands” in the programmer’s guide.

Example This example sets the "InfiniiSim 2 Port" math function, operands, and controls.

```
myScope.WriteString ":FUNCtion1:MATLab:OPERator 'InfiniiSim 2 Port'"
myScope.WriteString ":FUNCtion1:MATLab CHANnel1"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol1 'c:\users\public\
documents\infiniium\filters\cable only.tf2'"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol2 5e-9"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol3 10e9"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol4 2"
```

See Also

- ":FUNCtion<F>:MATLab:OPERator" on page 39
- ":FUNCtion<F>:MATLab:CONTrol<N>" on page 37

History Legacy command (existed before version 3.10).

Version 4.30: Up to 16 functions supported.

:FUNCtion<F>:MATLab:CONTrol<N>

Command `:FUNCtion<F>:MATLab:CONTrol<N> {<value> | <string>}`

The :FUNCtion<F>:MATLab:CONTrol<N> command sets control values for these user-defined functions:

- Butterworth
- FIR
- LFE
- RTEye
- SqrtSumOfSquare

And these InfiniiSim functions:

- InfiniiSim 2 Port
- InfiniiSim 4 Port 1 Src
- InfiniiSim 4 Port CM
- InfiniiSim 4 Port Diff
- InfiniiSim 4 Port Src1
- InfiniiSim 4 Port Src2

<F> An integer, 1-16, representing the selected function.

<N> An integer, 1-6, representing the user-defined or InfiniiSim function control.

<value> A double, integer, or enumerated type value. For an enumerated type, the 1 based index is passed to select the enumeration.

<string> A character array.

Example This example sets the "InfiniiSim 2 Port" math function, operands, and controls.

```
myScope.WriteString ":FUNCtion1:MATLab:OPERator 'InfiniiSim 2 Port'"
myScope.WriteString ":FUNCtion1:MATLab CHANnel1"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol1 'c:\users\public\
documents\infiniium\filters\cable only.tf2'"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol2 5e-9"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol3 10e9"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol4 2"
```

Query `:FUNCtion<F>:MATLab:CONTrol<N>?`

The :FUNCtion<F>:MATLab:CONTrol<N>? query returns the value or string of the user-defined control.

Returned Format `[:FUNCtion<F>:MATLab:CONTrol<N>] {<value> | <string>}<NL>`

Example This example places the current returned value for function 1 control 1 in the string variable, strSelection, then prints the contents of the variable to the computer's screen.

```
Dim strSelection As String   ' Dimension variable.
myScope.WriteString ":FUNCtion1:MATLab:CONTrol1?"
strSelection = myScope.ReadString
Debug.Print strSelection
```

See Also

- ":FUNCtion<F>:MATLab:OPERator" on page 39
- ":FUNCtion<F>:MATLab" on page 36

History Legacy command (existed before version 3.10).

Version 4.30: Up to 16 functions supported.

Version 5.60: Up to 6 user-defined controls supported.

:FUNCtion<F>:MATLab:OPERator

Command `:FUNCtion<F>:MATLab:OPERator <string>`

The :FUNCtion<F>:MATLab:OPERator command sets the Function dialog box operator. Any math function operator name can be specified, not just user-defined or InfiniiSim math functions.

<F> An integer, 1-16, representing the selected function.

<string> A character array that is the name of the math function as it appears in the Function dialog box.

Example This example sets the "InfiniiSim 2 Port" math function, operands, and controls.

```
myScope.WriteString ":FUNCtion1:MATLab:OPERator 'InfiniiSim 2 Port'"
myScope.WriteString ":FUNCtion1:MATLab CHANnel1"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol1 'c:\users\public\
documents\infiniium\filters\cable only.tf2'"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol2 5e-9"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol3 10e9"
myScope.WriteString ":FUNCtion1:MATLab:CONTrol4 2"
```

Query `:FUNCtion<F>:MATLab:OPERator?`

The :FUNCtion<F>:MATLab:OPERator? query returns the string of the math function operator.

Returned Format `[:FUNCtion<F>:MATLab:OPERator] <string><NL>`

Example This example places the current operator string for function 1 in the string variable, strSelection, then prints the contents of the variable to the computer's screen.

```
Dim strSelection As String   ' Dimension variable.
myScope.WriteString ":FUNCtion1:MATLab:OPERator?"
strSelection = myScope.ReadString
Debug.Print strSelection
```

See Also

- ":FUNCtion<F>:MATLab" on page 36
- ":FUNCtion<F>:MATLab:CONTrol<N>" on page 37

History Legacy command (existed before version 3.10).

Version 4.30: Up to 16 functions supported.

6 Deep User Defined Functions

For the 90000A, 90000 X-Series, or 9000A Series oscilloscopes, software version 3.11 and later has the ability to perform file based IO for user defined functions (UDF). This permits applying a function to a longer memory depth than would otherwise be allowed. While there are no hard constraints on the length of a waveform to pass in or out of a function, you will likely run out of memory in the oscilloscope application when trying to pass deep waveforms (greater than 8M points).

For deep waveforms, you can instead use a file to pass the data in and out of a UDF. To use file based IO, the MATLAB script needs to read from and write to a file.

In most cases, you will want to process a part of the source data at a time so as to not create internal MATLAB arrays that are too large.

Also, the .XML file must contain a keyword to tell the oscilloscope to use file IO. This line should be added to the .XML file that configures the oscilloscope:

```
<FileIO>true</FileIO>
```

Instead of passing SrcData, Src1Data, or Src2Data variables to MATLAB, the oscilloscope passes SrcDataFile, Src1DataFile, or Src2DataFile variables. These variables are strings that are the filename and path to the files that contain source data.

In the MATLAB script, create a variable 'FnDataFile' that is the path and filename of the resultant data.

The source and function data files are binary files of type 'double'.

Here is an example script that creates the square root of the sum of squares of the input waveforms:

```
% Open the source files with read access.
Src1Fid = fopen(Src1DataFile,'r');
Src2Fid = fopen(Src2DataFile,'r');

% Get the size of the files to determine the number of points.
File1Details = dir(Src1DataFile);
File2Details = dir(Src2DataFile);

% In this example, process the data in chunks of 100000 points.
PacketSize = 100000;
```

```
% The files contain binary double data, or 8 bytes per point.
BytesPerDouble = 8;

% Find the smaller of the two sources.
Bytes = min(File1Details.bytes, File2Details.bytes);

% Compute how may chunks to process.
NumPackets = int32((Bytes / BytesPerDouble) / PacketSize);

% Round up. Could use ceil().
if NumPackets * PacketSize < (Bytes / BytesPerDouble)
   NumPackets = NumPackets + 1;
end

% Get the path of the source data files.
[PathStr, Name] = fileparts(Src1DataFile);

% Using the same path, create a FnData file.
FnDataFile = [PathStr,'\FnData'];

% Open the output file with write access.
[FnFid, Message] = fopen(FnDataFile,'w');

% For each chunk of data:
for i = 1:NumPackets
   % Read the source data from the file.
   Src1Data = fread(Src1Fid,PacketSize,'double');
   Src2Data = fread(Src2Fid,PacketSize,'double');

   % Compute result.
   FnData = sqrt((Src1Data .* Src1Data) + (Src2Data .* Src2Data));

   % Write the result.
   fwrite(FnFid, FnData, 'double');
end

% Scaling for this particular function.
Src1YScale = Src1YScale / 2.0 + Src1YMiddle;
Src2YScale = Src2YScale / 2.0 + Src2YMiddle;

FnYScale = sqrt(Src1YScale * Src1YScale + Src2YScale * Src2YScale);
FnYMiddle = FnYScale / 2.0;

% Don't forget to close all files.
fclose(Src1Fid);
fclose(Src2Fid);
fclose(FnFid);
```

Index