Customize your oscilloscope to simplify operations

Arthur Pini - September 23, 2014

Today's digital oscilloscopes have an impressive set of acquisition, measurement, and analysis tools that let engineers and technicians perform tests. Even so, there are cases where you'd like to streamline the setup and automate operations, particularly for people who just need simple or repetitive measurements. Oscilloscopes let you customize their operations to limit the function of the instrument to a few, selected tests or to semi-automate these tests. Here are a few examples of where a customer user interface or test automation can help.

- You need to simply the operation of the oscilloscope to allow an inexperienced user to run repetitive testing.
- You need to automate a series of tests without hooking the instrument up to a controller or automatic test system.
- You need to analyze or measure the acquired data using proprietary processes or algorithms.
- Acquired waveforms need to be processed more rapidly than is possible by transferring the data into an external computer.

Simplified Setups and Test Flow
Oscilloscope manufacturers handle customization differently. I'll use a Teledyne LeCroy HDO6054, which uses a feature called CustomDSO to either present user configurable buttons on the display to evoke the desired operations or to replace the user interface with a custom GUI. Another oscilloscope manufacturer also offers the ability to program the existing user interface to remove or limit existing selections, thus simplifying instrument operation. Others offer macro programming based upon a series of stored panel setups. You can program a series of setups that compose the elements of your test. In some cases, you can even program dynamic operations.

All digital oscilloscopes have the ability to store and recall panel setups. This lets you set up the instrument and save the setup for future use or for use by a less experienced user. These stored setups can be recalled when needed. Basic mode CustomDSO (optional in some models) extends this capability and lets you link multiple setup files into a cohesive test sequence where each is called by a single user-defined button press. The called setups can themselves include calls to other setups, enabling you to create a hierarchy of tests. **Figure 1** shows an example of the application of basic CustomDSO to automate the setup of a test of a SPI low speed serial interface.

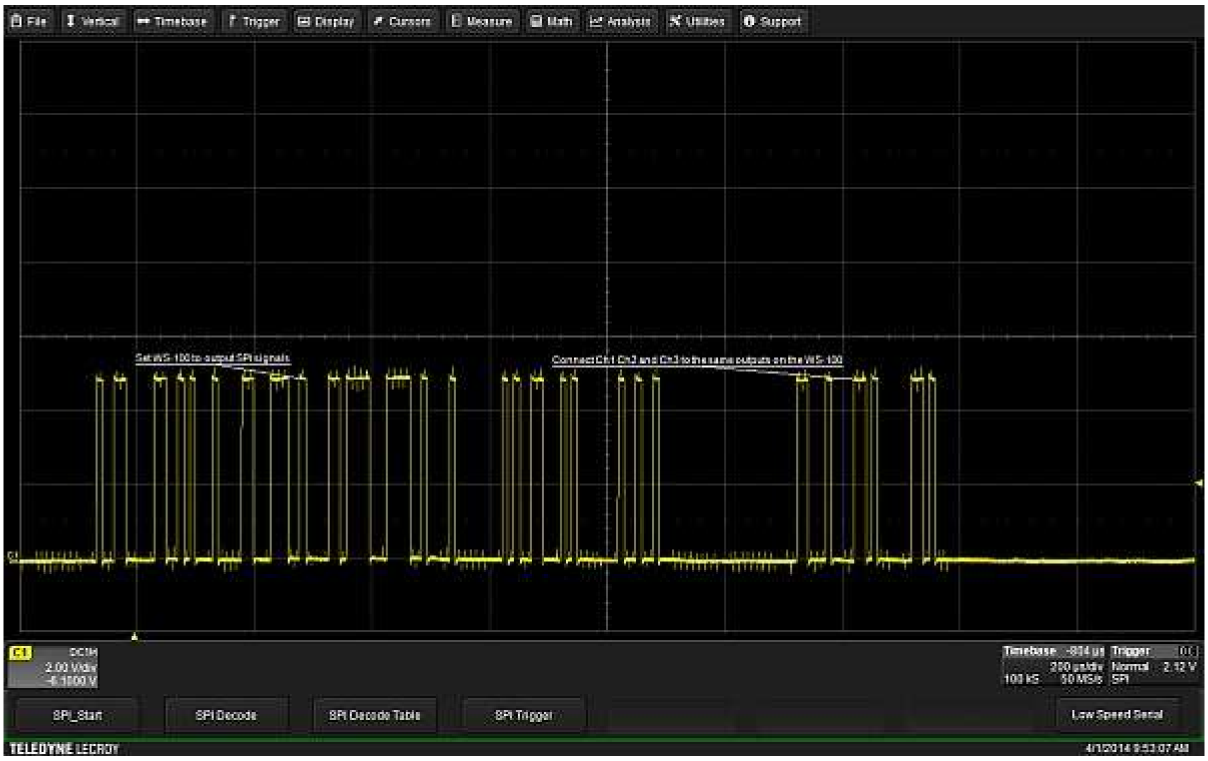
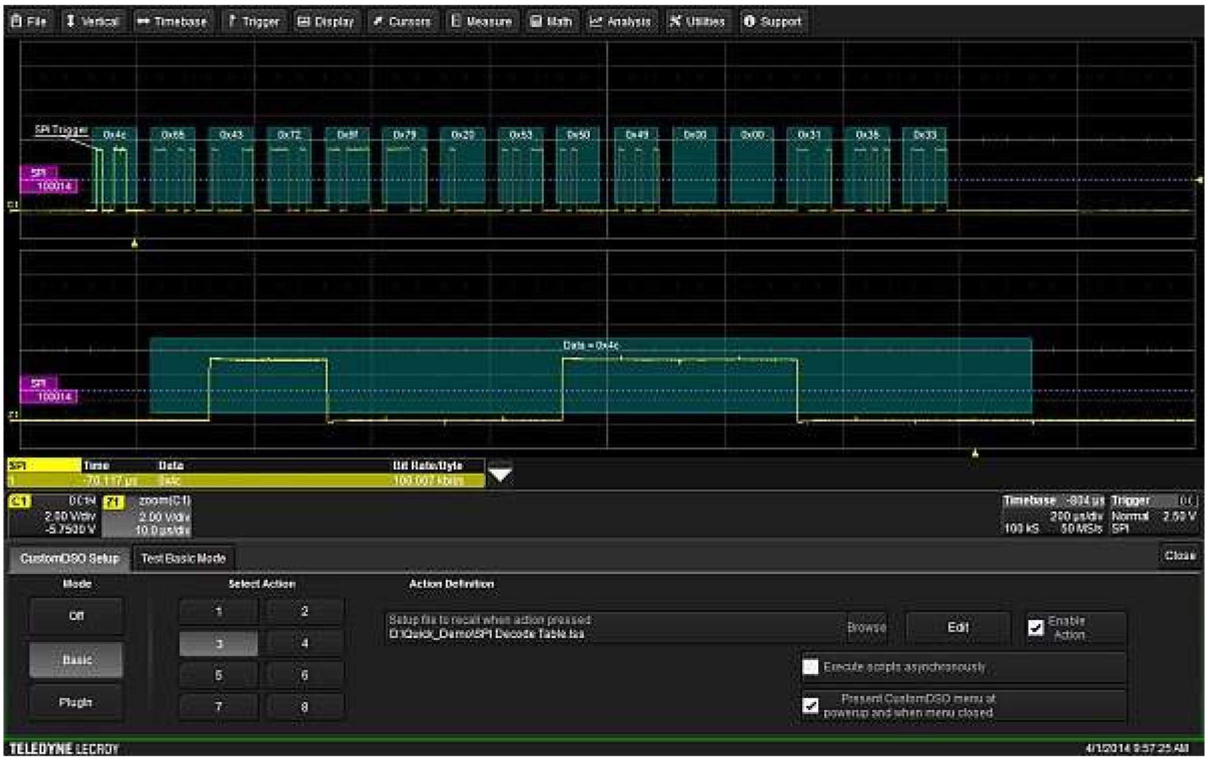


Figure 1. An example of a CustomDSO button bar with a selection of SPI tests . Up to eight buttons can be defined, each one calling a specific test setup.

Each of the eight buttons located at the bottom of the display are created using CustomDSO and each can setup a specific test. Operator prompts can be inserted using trace labels. In Figure 1, only five of the buttons have been assigned. Pressing any button can bring up additional sets of similar buttons to guide the user through a complete test.

Figure 2 shows the CustomDSO setup menu. In basic mode, each of up to eight CustomDSO buttons is associated with a scope setup file. The root of the file name is assigned as the button label. In the example, button 3 will call the setup file named SPI Decode Table.lss and the button will be labeled SPI Decode Table. Panel setups in this scope are actually VB (Visual Basic) scripts that define the state of the instrument. Each of these setup files can include additional CustomDSO button definitions so that multiple setups can be chained and called in an infinite variety of topologies.



SPI	Time	Data	Bit Rate/Byte
1	-70.117 µs	0x4c	100.007 kbit/s

C1 DC1M 2.00 V/div -5.7500 V
Z1 zoom(C1) 2.00 V/div 10.0 µs/div
Timebase -804 µs 200 µs/div 100 kS 50 MS/s
Trigger DC Normal 2.50 V SPI

Setup file to recall when action pressed
D:\Quick_Demo\SPI Decode Table.lss

[✓] Enable Action
[ ] Execute scripts asynchronously
[✓] Present CustomDSO menu at powerup and when menu closed

4/1/2014 9:57:25 AM

Figure 2. The CustomDSO setup shows the association of the setup file "SPI Decode Table" to the button 3. Pressing this button executes the setup file setting the oscilloscope to the desired setting. It can also bring up other CustomDSO buttons for additional tests.

Additionally, the panel setups can contain dynamic operations that automate scope operations. I'll discuss this further in the section on VBscripts.

Note that CustomDSO has two modes of operation (shown on the left) in the CustomDSO descriptor box in Figure 2. The second mode is "plug-In" and is a more powerful feature that let you add your own ActiveX controls to a setup. These controls are defined by routines written in Visual Basic, Visual C++, or other ActiveX compatible programming languages. With ActiveX controls, you can create your own custom user interfaces. A large number of interactive devices are available, including button, check box, radio button, list box, picture box, and common dialogue box. **Figure 3** shows an example of a plugin GUI using two push buttons. The first, Single Trigger, initiates a single acquisition. The second transfers the waveform in channel 1 (C1) to waveform memory M1 while progressively transferring M1 to M2, M2 to M3 and M3 to M4, effectively moving previously stored waveforms through the existing waveform memories. This latter function would require almost a dozen button pushes if done manually.

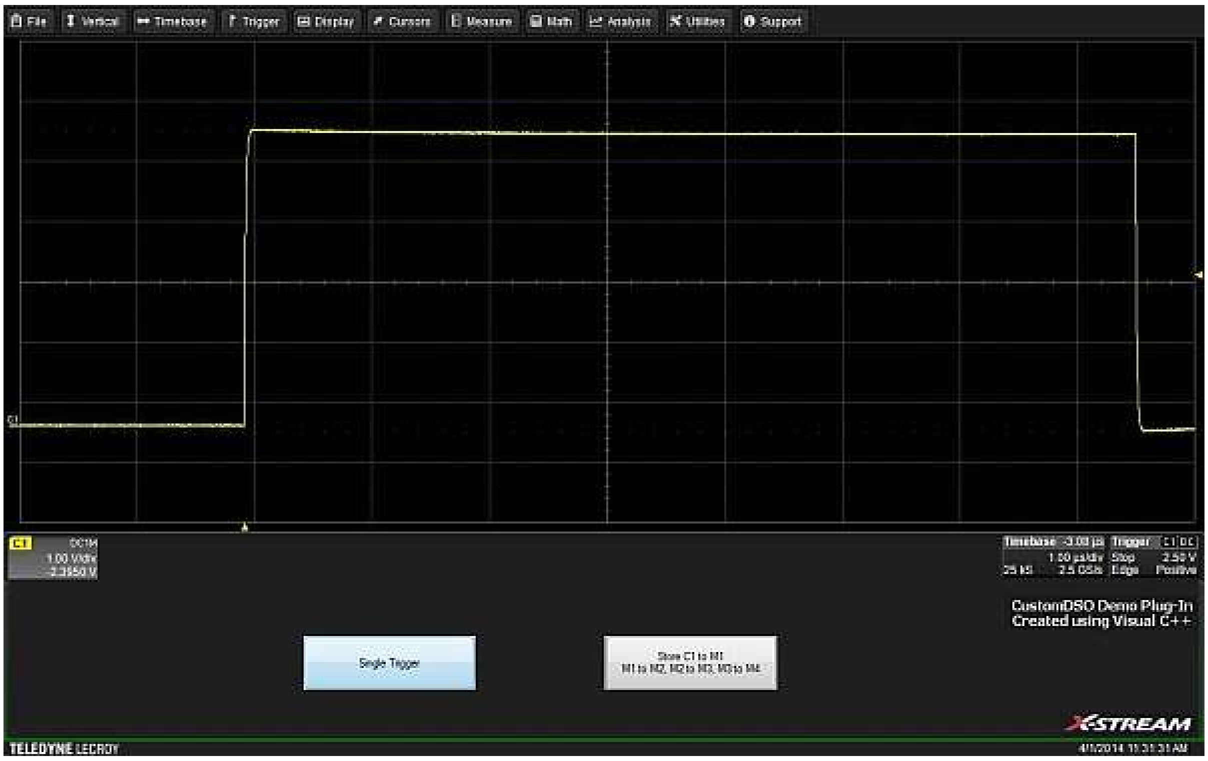


Figure 3. An example of a CustomDSO PlugIn used to create a user defined GUI that can contain buttons, check boxes, radio buttons, list boxes, picture boxes, or a common dialogue box.

Visual Basic stripts

Visual Basic Scripts
In this oscilloscope, the setup files are ASCII text files that contain a complete Visual Basic Script "program" that, when executed, will restore the instrument to a previously recorded state. In effect, each time a panel is saved, the instrument effectively writes you a program that, when executed, returns the instrument to the saved state.

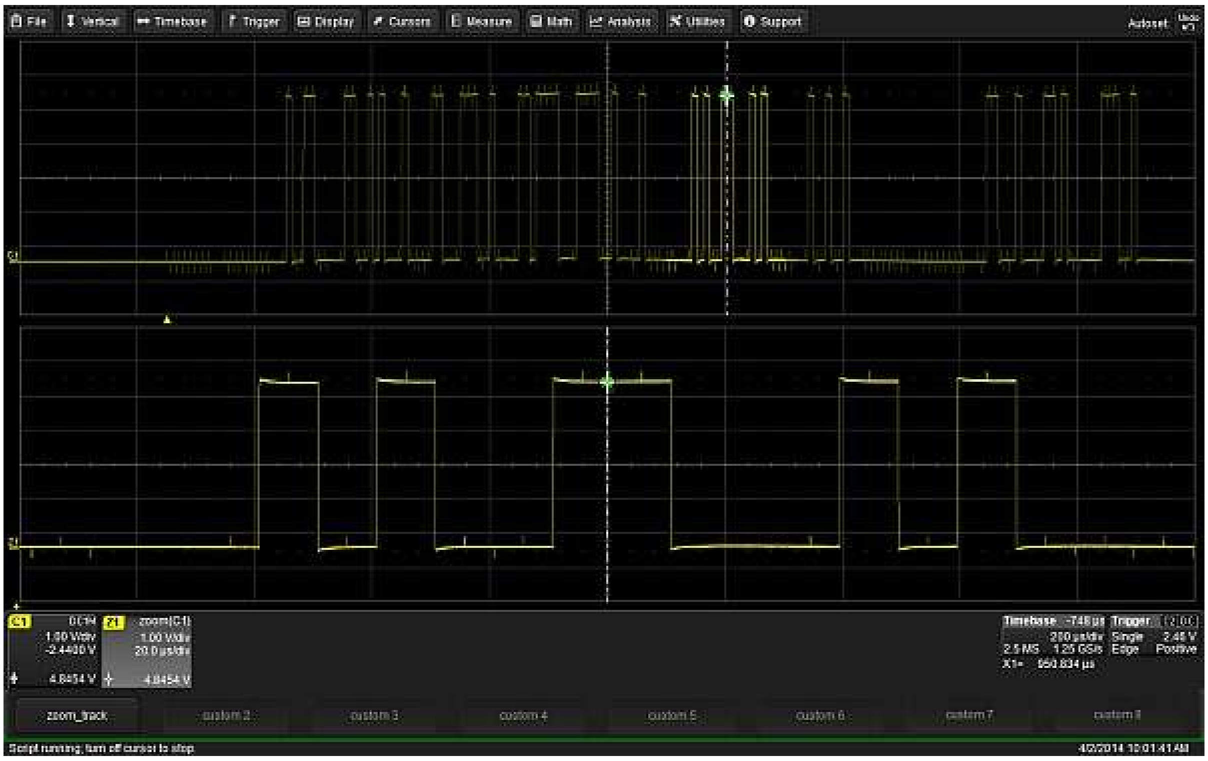
In addition to recalled setup states, VB scripting can be used to create programs that control the oscilloscope using standard remote commands. You can "automate" oscilloscope operations from an internally run setup file.

For instance, you can write a VB script that will move the center of a zoom trace to the position set by the placement of a cursor. In essence, the zoom trace follows the cursor location. The script in shown in **Listing 1**. The script is saved with the same file extension as a setup file (.lss) and is evoked by recalling a panel setup or by linking it to a CustomDSO button. By using VB scripts in conjunction with CustomDSO very interactive tests can be programmed. **Figure 4** shows CustomDSO being used to launch the VB script.

Listing 1. A Visual Basic Script (zoom_track.lss) to have the zoom trace center track a cursor.

```
set app = CreateObject("LeCroy.XStreamDSO")
' Display a message on the display
app.SystemControl.PersistentMessage = "Script running; turn off cursor to
stop."
' Change the trigger mode to stopped
app.Acquisition.TriggerMode = "Stopped"
```

```
' set the cursor type to horizontal absolute
app.Cursors.Type ="HorizAbs"
' Turn the cursor on
app.Cursors.View = True
'set the zoom 1 trace horizontal expansion to 10:1
app.Zoom.Z1.Zoom.HorZoom = 10
' Turn on the zoom 1 trace
app.Zoom.Z1.View = True
' Loop to have zoom center track cursor horizontal location, exit when cursor
is turned off
While app.Cursors.View = True
' force trigger immediately
'set arguments to 0,False to wait for a triggerable event.
app.Acquisition.Acquire -1,True
'Read cursor horizontal location
curtime=app.Cursors.XPos1
'Set Zoom Z1 center to cursor location
app.Zoom.Z1.Zoom.HorCenter=curtime
Wend
' Clear the message on the screen
app.SystemControl.PersistentMessage = ""
' Disconect the automation link
Set app = Nothing
```



C1 DC1M 1.00 V/div -2.4400 V 4.8454 V
Z1 zoom(C1) 1.00 V/div 20.0 µs/div 4.8454 V
Timebase -748 µs 200 µs/div 2.5 MS 1.25 GS/s X1= 950.834 µs
Trigger Single Edge 2.46 V Positive
Script running; turn off cursor to stop
4/2/2014 10:01:41 AM

Figure 4. Using CustomDSO to launch the VB script "zoom_Track.lss". When the script is running the center of the zoom trace will track the horizontal absolute cursor position. As seen above the cursor will always be in the center of the zoom trace Z1.

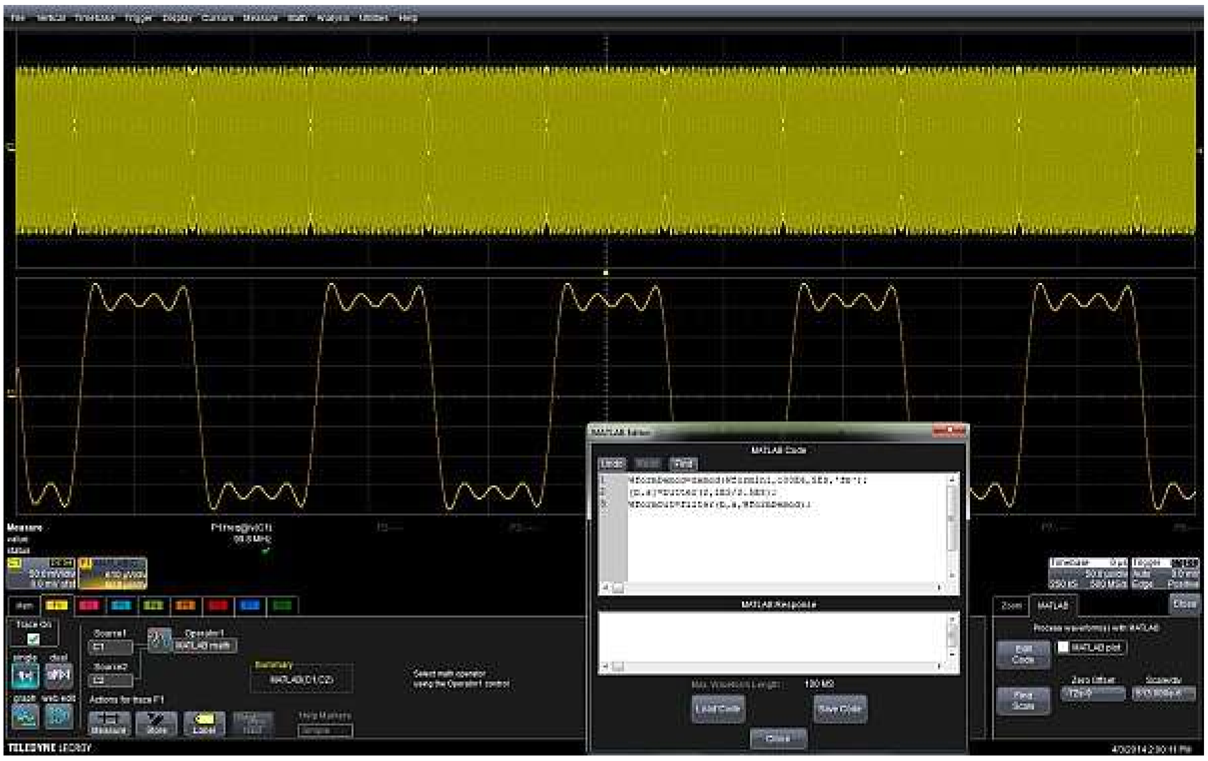
The availability of panel setups, CustomDSO, and VB script-based setups provides plenty of flexibility in controlling the flow of oscilloscope measurements. It also provides a simple way to simply the scope operation by having inexperienced uses follow a script based on CustomDSO button presses.

Custom Processing and Measurements
Customization can also add proprietary processing to the scope's toolset allowing you to use proprietary analysis and measurements to an existing scope. We'll look at these and other customization techniques that can be applied to mid-range bench oscilloscopes.

Midrange and high-end oscilloscopes from all major manufacturers offer some sort of math customization based on the insertion of a third-party program within the processing path of the oscilloscope as an optional feature. All of them support MATLAB. The oscilloscope I used for this article supports MATLAB, Visual Basic, C/C++, Excel, or Java. This feature is based on a COM-based architecture that provides a high level of customization. User-defined algorithms can be created and inserted into the oscilloscope processing path and the result returned as a processed math function or measurement that can be displayed like any other oscilloscope trace or measurement. Such traces can be further processed by internal or user-defined algorithms.

Because the oscilloscope uses Microsoft Automation commands for remote control, they can also be embedded into the user-defined algorithm enabling "smart algorithms" capable of making real-time decisions during calculation as we saw already with VB scripting.

As an example, consider the oscilloscope measurement set up in **Figure 5**. Here a 100 MHz sinusoidal carrier is frequency modulated by a 10 kHz square wave. The waveform is demodulated using a simple MATLAB script and the demodulated waveform is shown in math trace F1. The edit box allows the users to create or edit or load the script without leaving the scope environment making convenient to create or change the script.



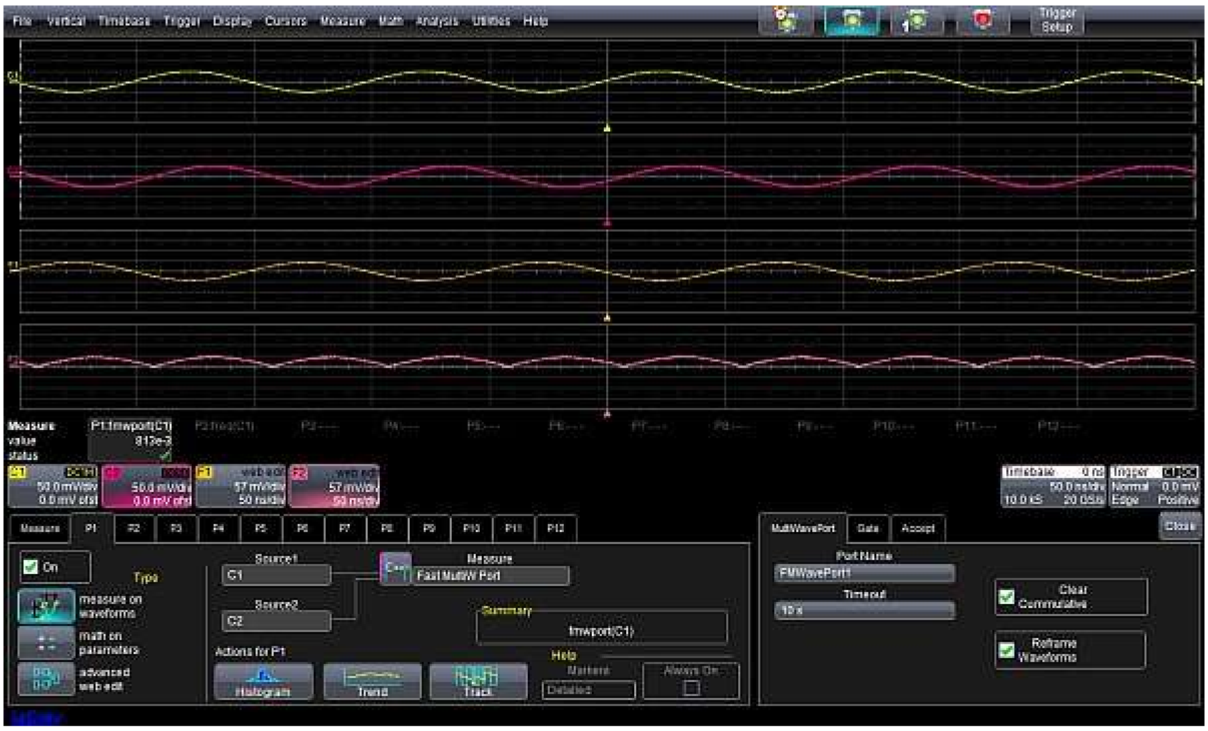
```
WformDemod=demod(WformIn1,100E6,5E9,'fm');
[b,a]=butter(2,1E5/2.5E9);
WformOut=filter(b,a,WformDemod);
```

Figure 5. This embedded MATLAB script demodulates an FM carrier and displays the modulation waveform. The edit box allows the users to create or edit the script without leaving the scope environment.

Analog demodulation of the FM signal is accomplished using MATLAB's "demod" function shown in the first line of edit box in Figure 5. This function takes as arguments the source waveform, carrier frequency, sampling frequency, and modulation type, respectively. The output from the demod function has to be filtered and a 1 MHz, second order Butterworth low-pass filter is implemented in MATLAB by the next two lines of code. A similar technique can be used to create custom measurements as well as math functions, the process is identical.

Fast Custom Processing
An advanced feature of this oscilloscope’s customization option provides memory mapped access to the scope’s data for user based waveform or measurement computation. This feature is called FastMultiWavePort (FMWP) and enables insertion of custom processing algorithms, written in the C/C++ language, into the oscilloscopes processing stream. FMWP maximizes data throughput from the acquisition system to your processing by using a shared memory window. Results can be sent back to the scope or processed independently. This feature can support up to four waveform inputs and outputs along with up to eight parameter outputs.

Figure 6 shows an example of using FMWP to acquire two waveforms from the scope and output two processed waveforms and a parameter back to the scope. The outputs are computed by a C++ program accessing the input waveform via memory mapped data. The output in F1 is simply the inversion of the input C1. F2 is the absolute value of input C2. The parameter P1 is the correlation coefficient of the two inputs. The biggest advantage of FMWP is that it permits the engineer to create his own proprietary code and apply it to the scope data at the highest possible processing speed.


Measure P1:fmwport(C1)
value 813e-3
status ✓

Source1: C1
Source2: C2
Measure: Fast MultiW Port
Summary: fmwport(C1)
Port Name: FMWavePort1
Timeout: 10 s
✓ Clear Commulative
✓ Reframe Waveforms

Figure 6. The FastMultiWavePort parameter setup in P1 along with outputs F1 (inversion of input C1) and F2 (absolute value of input C2), created using a C++ program operating

on memory-mapped data from the inputs.

Conclusion
The oscilloscope is one of the workhorse instruments of electronic design and test due to its extensive tool set and application flexibility. Customization can simplfy testing, proprietary processing, and internal automation to the oscilloscope's toolset further extending the usefulness of the instrument.

Also see
Create Your Own ActiveX Controls

Other oscilloscope articles by Arthur Pini
Why are oscilloscope probe amps at the tip?
Perform pass/fail tests with an oscilloscope
Digitizers: Finer resolution is better
10 Tricks that Extend Oscilloscope Usefulness
Electromechanical measurements with an oscilloscope
MSOs probe analog and digital
How to select a modular waveform digitizer
Read sensors with an oscilloscope
Measure vector and area with an oscilloscope X-Y display
Product How To: Calculate power with a scope
Improve power supply reliability
How to measure instantaneous RF power
How to perform histogram analysis on your oscilloscope