THE LXI IVI PROGRAMMING MODEL FOR SYNCHRONIZATION AND TRIGGERING

Lynn Wheelwright
3751 Porter Creek Rd
Santa Rosa, California 95404
707-579-1678
lynnw@sonic.net

Abstract - The LXI Standard provides three synchronization and trigger methodologies in addition to what system designers are familiar with on rack and stack instruments:

- ***A high speed LVDS trigger bus that is eight lanes wide***
- ***LAN events and triggers which can be used in place of wires or in addition to them***
- ***Very accurate absolute time keeping, time stamping, and time based triggers using IEEE 1588 clocks***

An IVI programming model for controlling these methodologies is explored and an example measurement scenario presented based on the use of synthetic instruments working together to make a stimulus-response measurement. The sample code presented shows the ease with which a test engineer can switch between the LXI trigger lines and the equivalent LAN trigger mechanism. Logic models and a state machine example are presented describing necessary arming, triggering, and event generation logic. Tradeoffs between hardware and software implementations are outlined and all of these models are wrapped together to illustrate what is needed in an LXI device to use these capabilities.

INTRODUCTION

Historically, instruments have generally provided triggering and synchronization that matched the intended use for the measurement capability provided. This led to many varieties of triggering and many different capabilities associated with each type of trigger. Naturally, for each of these different types of triggering and synchronization a different and unique set of programming commands was developed. In order to achieve some semblance of order and transfer of learning to test system program development, the IVI (Interchangeable Virtual Instruments) Foundation was established. With its concept of instrument classes and the notion of inherent capabilities that each instrument's programming interface should support, the stage was set for further improvements in test system architecture.

The LXI (LAN Extensions for instruments) standard brings both new triggering capability as well as a unified programming model for the various types of triggering found within a test system. In addition, using Ethernet as the communication and control mechanism allows test system designers to address difficult measurement scenarios (such as large physical devices or distributed measurement problems) with greater ease. The new capabilities are:

1. An eight lane wide improved hardware trigger bus using LVDS technology. It can handle higher speed signals and provide better noise immunity.
2. An Ethernet signaling protocol that matches up to the improved hardware trigger bus so that measurement scenarios requiring greater separation among the measuring devices, or that don't necessarily need the ultimate in speed can be more easily implemented.
3. Very accurate time base synchronization among measuring devices utilizing IEEE 1588. This allows each device to maintain accurate time with respect to all of the other devices in the system. Based on this, very accurate time based triggering is possible for large numbers of measuring devices. This is also the basis

for coupling accurate timestamps to data so that correlations among many instruments' results can be computed.

EXAMPLE MEASUREMENT SCENARIO

For illustration purposes, let's examine a stimulus response measurement using synthetic modular instruments as the measurement hardware. The device under test is an amplifier that needs to be checked for distortion over its operating range using a digitally modulated test signal of 550 microsecond pulses. Each pulse is 2 dB lower than its predecessor covering a dynamic range of 90 dB. Each amplifier is tested over 101 equally spaced frequency points between 2 GHz and 3 GHz. Our test assets are:

1. An arbitrary waveform generator.
2. An up converter.
3. A down converter.
4. A digitizer.

A block diagram of how the test assets may be connected is shown in figure 1. The trigger connections may be implemented using either the LXI trigger bus, or Ethernet signaling.

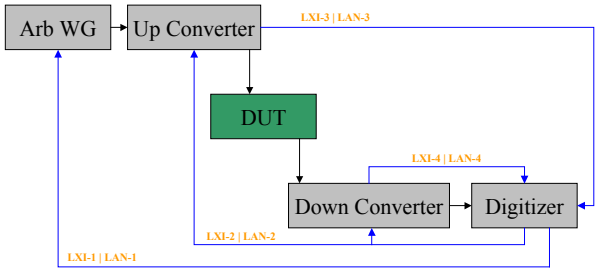

Figure 1. Block Diagram

The test signal is created in the arbitrary waveform generator and routed to the up converter where it is translated up in frequency. The output of the up converter is routed to the input of the DUT. After passing through the DUT, the signal is routed to the down converter where it is translated down to a frequency with in the range of the digitizer. The digitizer samples the incoming signal for use by the test program.

The synchronization or trigger signals (denoted in blue on the diagram), allow the modules to synchronize their operations as follows:

1. The digitizer will use an internally generated signal based on the rising edge of the input signal to start its acquisition.
2. The digitizer shall wait for both the up and down converters to settle before entering the 'Waiting for Trigger' state.
3. Upon entering the 'Waiting for Trigger' state, the digitizer shall signal the arbitrary waveform generator to output the test signal.
4. When the acquisition is complete, the digitizer transmits the data to the controller and signals the up and down converters to move to the next frequency.

Since the digitizer has the most interactions with other modules, let's examine the trigger logic used to accomplish this (see figure 2). Here we see the symmetrical layout of the LXI trigger bus and the Ethernet signaling.

In this top level view, event signals are received from the left via the LXI Trigger Bus or the network interface (UDP port listener and TCP socket listeners). The LAN0..7 input registers capture the values of the event signals received over the network for use by the Arm or Trigger logic. This logic determines which signals the Arm-Trigger State Machine uses once a measurement sequence has been initiated by the module controller. The state machine is responsible for starting the triggered action and notifying other entities of the progress of the measurement sequence. The Event logic monitors the signals from the state machine (and other sources as appropriate) and is responsible for sending events out over the LXI Trigger Bus or over the network.

While it is possible that all of these pieces could be implemented totally in hardware or totally in software, it is expected that most implementations will be a mixture. This would be typical of modules requiring tight timing tolerances or fast response to a trigger—especially when received over the LXI Trigger Bus. For the purposes of this example the items in green are assumed to be software and the items in blue and yellow are a mixture of hardware and software. In order to issue network events the hardware must be capable of notifying the software to initiate the transmission. Typically, connecting the appropriate signals to the processor's interrupt circuitry enables this.

As we look at a flow chart in figure 3 of the Arm-Trigger state machine, the sections are color coded for easier identification. In this illustration of a digitizer, all of these sections are needed in

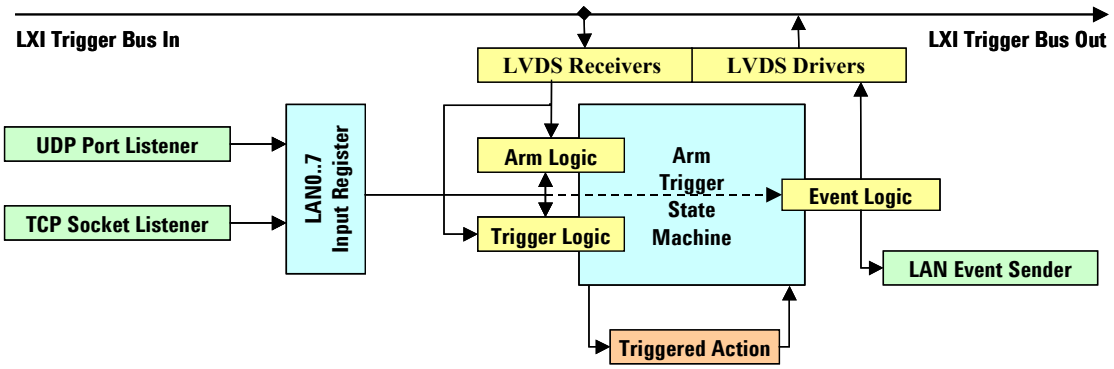

Figure 2. Trigger Logic Top Level View

order to perform the example measurement. Other modules such as the Up Converter or Down Converter, may not need the Arm portion of the state machine since the only triggerable action may be to step the frequency as part of a frequency sweep. Before leaving out sections of the state machine, be sure to evaluate all of the use cases for the module.

This flow chart is based on the *SCPI* [1] trigger state machine model. One feature to be aware of is that it is possible to stack multiple Arm sections and multiple Trigger sections on top of each other if the application warrants it (a logic analyzer comes to mind as a more complex example of triggering where this might be done).

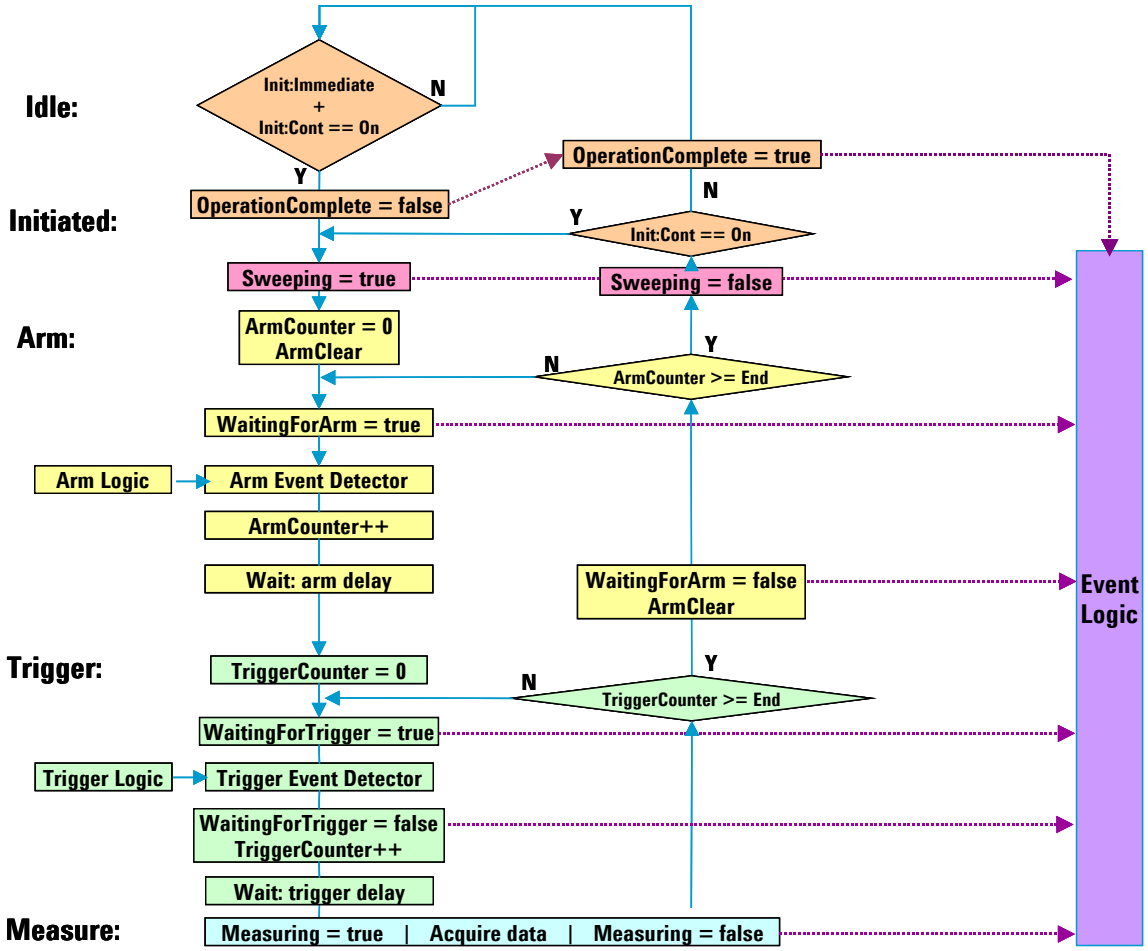

Figure 3. Digitizer Trigger State Machine Model

While this diagram shows various interesting state machine signals connecting to the Event logic, there are other signals which one might also wish to route onto the LXI Trigger Bus or send as LAN Events. This might include synchronization clocks, cross routing LXI Trigger Bus signals, or mapping the trigger bus to LAN events, etc. The only requirement for these signals is that they are connected to the Event logic so that they can be programmatically routed in a common interface.

Figure 4 shows the relationships among the state machine signals as it sequences through a measurement. Due to the looping capabilities of the various sections of the state machine, it is possible to have several levels of repetition (as denoted by braces around groups of signals).

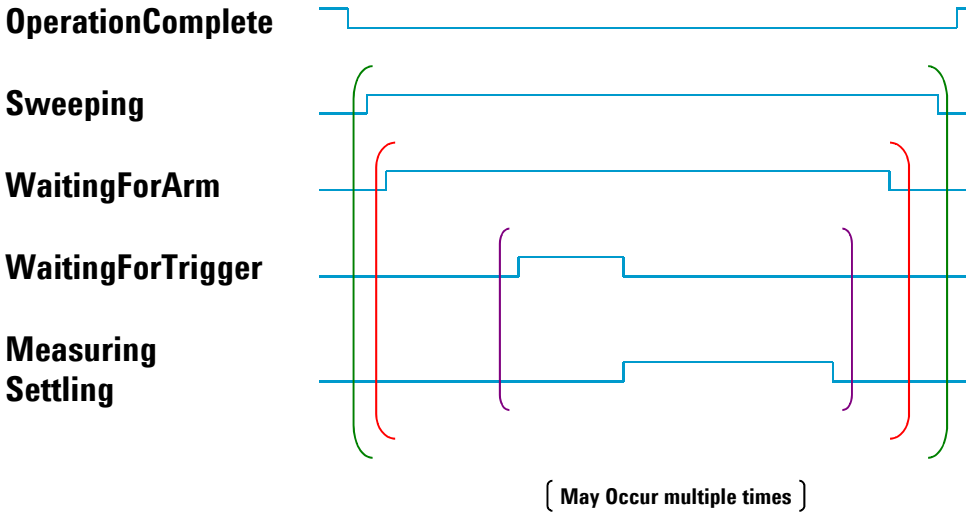

Figure 4. Trigger State Machine Signal Relationships

These signals are referenced in the programming example and the Arm, Trigger and Event logic diagrams to follow. Figures 5 and 6 are the Arm logic and a pictorial representation of the programming interface. Figures 7 and 8 are the Trigger logic and its programming interface. Figures 9 and 10 are the Event logic and programming interface.

The Arm logic provides the ability to selectively enable any of the inputs as well as selections for edge or level sense and positive or negative slope (or level). This logic includes the capability of OR summing as well as AND summing for those cases when a measurement may be initiated from multiple sources independently.

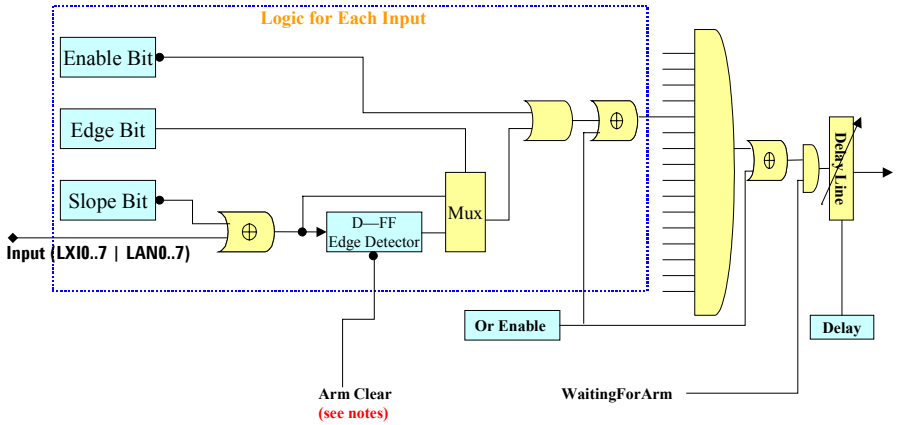

Figure 5. Arm Logic

The delay built into these models is to compensate for electrical delay differences between the arm signal path and the signal path through the device under test. There are devices such as narrow band crystal filters that have significant delay that must be taken into account in order to make valid measurements.

Figure 6 is a representation of an IVI-COM interface for this logic. The

mapping of the methods and properties onto the logic can be readily seen.

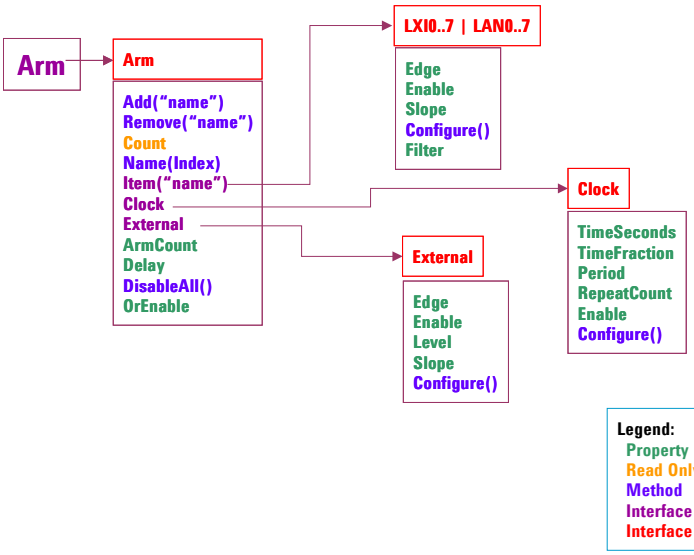

Figure 6. IVI-COM Arm Interface

The Arm, Trigger, and Event interfaces are designed to be extensible so that more LAN based items may be added at runtime when needed. This is the purpose of the Add and Remove methods.

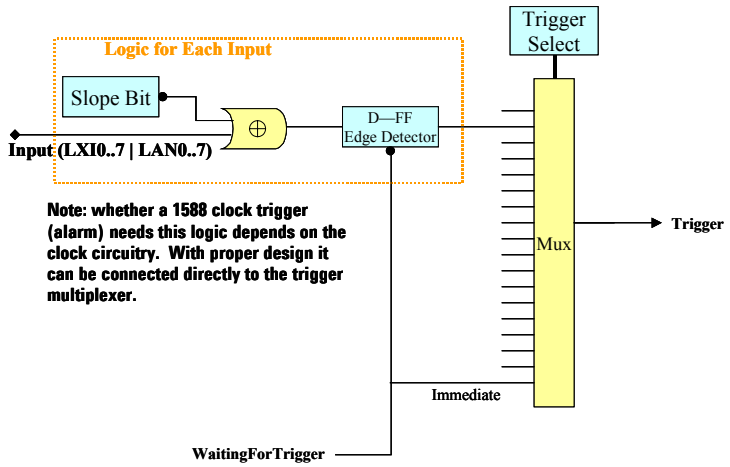

Figure 7. Trigger Logic

The Trigger logic (above) is a bit simpler than the Arm logic. This is mainly because we select only one trigger at a time, and most triggers happen on an edge. In some cases, where we want to OR together events and use the result to trigger the module, this can be done by using the OR summing in the Arm logic and selecting an Immediate trigger in the trigger block. A similar technique can be used for trigger gating (controlling the acquisition process with the level of a signal). All that is needed to set the Edge property to false in the Arm logic and again select the Immediate trigger in the Trigger logic.

For both Arm and Trigger logic, external triggers without level attributes may be treated in the same repeated capability group as LAN and LXI triggers. The trigger interface in figure 8 illustrates the case where the external triggers do have a settable level attribute.

The last major block is the Event logic. It is responsible for routing signals to the appropriate Event transmitter (either LXI Trigger Bus line or a LAN event packet). All signals (not just signals from the Arm-Trigger state machine) which are intended to be utilized for sending events or routed to the LXI Trigger Bus, need to be connected to the input multiplexers in the Event logic shown in figure 9.

Also included in this logic is the ability to invert the signal. The Enable signal is a three state signal—Off, On, and WireOr. This is applied to both the LAN and the LXI trigger bus outputs. In the case of LAN events in the WireOr mode, this effectively gives the test system programmer the ability to select which edge of the signal to use to generate an event (thereby reducing the LAN traffic by a factor of two). Note, while the analogy of wired-or logic works both on LAN and the LXI Trigger Bus, the use of negative logic to achieve a wired-and function is very problematic over LAN and is not

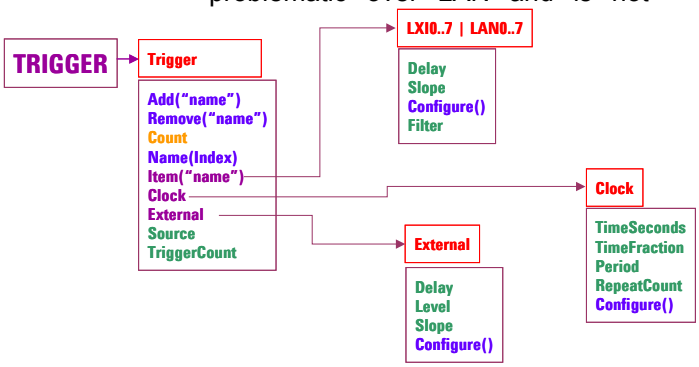

Figure 8. IVI-COM Trigger Interface

supported in this model. Instead, use the AND capability in the Arm logic and route each event source to a different input.

The Event interface diagram in figure 10 shows how the methods and properties are mapped onto the logic.

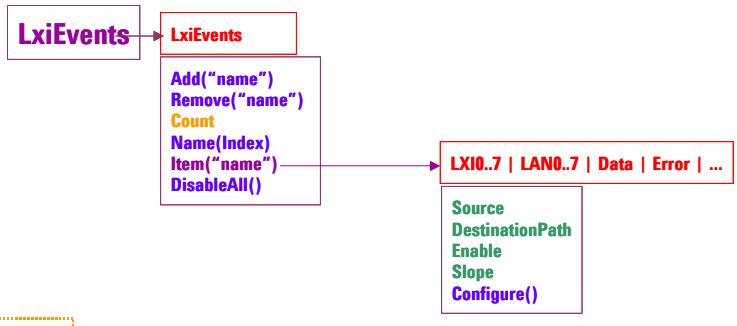

Figure 10. IVI-COM Event Interface

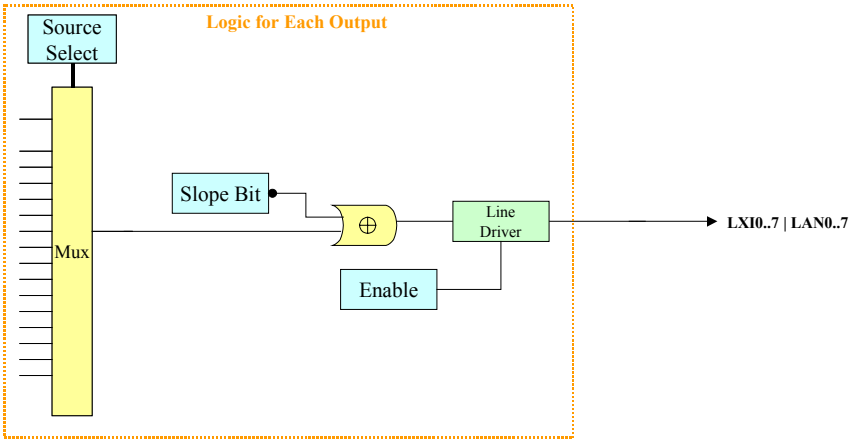

Figure 9. Event Logic

SAMPLE CODE IN C#

```
// tell the digitizer to output WaitingForTrigger on LXI1—tell Arb to use LXI1 to trigger.
Digitizer.Events.Item("LXI1").Configure( WaitingForTrigger, "", On, Positive );
Arb.Trigger.Item("LXI1").Configure( 0, Positive );
Arb.Trigger.Source = "LXI1";

// tell digitizer to output WaitingForArm falling edge as trigger for up and down converters
Digitizer.Events.Item("LXI2").Configure( WaitingForArm, "", On, Negative );
UpConverter.Trigger.Item("LXI2").Configure( 0, Positive );
UpConverter.Trigger.Source = "LXI2";
DownConverter.Trigger.Item("LXI2").Configure( 0, Positive );
DownConverter.Trigger.Source = "LXI2";

// tell up-converter to output Settling on LXI3 for use by digitizer
UpConverter.Events.Item("LXI3").Configure( Settling, "", On, Negative );
Digitizer.Arm.Item("LXI3").Configure( true, true, Positive );

// tell down-converter to output Settling on LXI4 for use by digitizer
DownConverter.Events.Item ("LXI4").Configure( Settling, "", On, Negative );
Digitizer.Arm.Item("LXI4").Configure( true, true, Positive );

// load signal file into arb and get it ready to go
…
// setup rest of digitizer parameters (acquisition length, SignalLevel trigger, etc.)
…
// start digitizer - acquire 101 signal packets
Digitizer.Arm.ArmCount = 101;
```

```
Digitizer.Initiate();
// digitizer will now wait until it sees Settled from the up and down converters.

// setup frequencies for up and down converter
UpConverter.Frequency.Sweep.Configure( 2e9, 3e9, 10e6 );
DownConverter.Frequency.Sweep.Configure( 2e9, 3e9, 10e6 );

// as soon as both converters settle, the digitizer will trigger the arb and start measuring

// loop and read back data
double [,] data = new double[101, 10000];
for( int index = 0; index <=100; index++) data[index] = Digitizer.Trace.Fetch();
// Done
```

The above code example uses the LXI Trigger Bus for measurement synchronization. To modify this code for LAN triggering simply requires replacing the string "LXI" with "LAN" in all instances highlighted in violet and underlined above. This symmetry between the LXI Trigger Bus and LAN triggering illustrates the simplicity of using the proposed arm and trigger interfaces in a traditional functional test scenario.

For other scenarios, having an accurate time of day clock based on IEEE 1588 for triggering allows the additional capability of simultaneously triggering hundreds or thousands of measurement devices with an accuracy in the tens of nanoseconds. The combination of all of these triggering methods enables the test system designer to deal with a wider range of measurement issues in a consistent and straight forward manner.

Some rules of thumb for selecting which trigger resources to use include:

- For tight timing requirements needing hard wired signals within a test system, the LXI Trigger Bus is recommended.
- If the timing requirement accuracy need is a few tens of nanoseconds, or the measurement hardware is widely dispersed, then triggering based on IEEE 1588 clocks is a viable solution.
- If the timing and synchronization requirements can tolerate a latency of tens to a few hundred microseconds (or greater), LAN triggering is a very convenient solution.

SUMMARY

The example measurement illustrates just a few of the triggering and synchronization capabilities that can be addressed by the LXI triggering facilities. It was chosen because it is typical of measurements made in functional test systems. The capability and flexibility of the LXI triggering interfaces can also be applied to data acquisition and monitoring situations as well.

A key element in the design of the LXI trigger interfaces, is that the use of the LXI Trigger Bus and the LAN triggers should be as similar as possible. The reason for this is so that test system programmers can switch between the two easily and quickly as their needs change—thus allowing the same programming techniques to address a wider range of situations.

Likewise, standardizing the trigger state machine names used in the interfaces improves portability and shortens test system development time. Lastly, the extensible nature of the LAN triggering and events interfaces allows the system designer to add logical trigger channels on an as-needed basis.

REFERENCES

[1] Standard Commands for Programmable Instrumentation (SCPI) Consortium, *Volume 2: Command Reference*. 1999, pp. 24-1 to 24-4. http://www.scpiconsortium.org/SCPI-99.pdf