PC Based, IEEE Signal & Test Description Standard, Synthetic Instruments

Matt Cornish, Racal Instruments Ltd, UK, +44 (0)1202 872800, matt.cornish@racalinst.co.uk
Richard Hazlewood, Racal Instruments Ltd, UK, +44 (0)1202 872800, richard.hazlewood@racalinst.co.uk
Chris Gorringe, Racal Instruments Ltd, UK, +44 (0)1202 872800, chris.gorringe@racalinst.co.uk

ABSTRACT

There exists a trend in Automatic Test Environments (ATE) towards the synthesis of signals using digital memory devices, which, through recent advances in technology, are now able to achieve output at frequencies that were previously the domain of analogue devices.

This paper will discuss the use of a recent trend in PC technology, DirectX®, to enable real-time streaming of signal definitions directly to digital memory devices onboard a standard PC that is also running the Automated Test Program (ATP).

We will consider the use of the emerging IEEE Signal & Test Description (S&TD) standard to allow us to define stimulus signals using the STD Basic Components (e.g. Sinusoid, AM, Clock).

We have devised a system that will enable us to design, simulate and implement complex stimulus signals at useful frequencies on a single machine. We have used STD to ensure an international standard, providing flexibility, portability and a natural source for simulation. By mapping STD SML onto a DirectX® framework, we have created a real-time simulation and, through onboard PC hardware, a real world signal.

Keywords. S&TD, signal, synthetic, instrument, ATE, ATP, A2k.

1 Introduction

How nice it would be if we could make use of the computing power of a modern PC to combine Automated Test Program (ATP) design, implementation and Automatic Test Environment (ATE)!

AUTOTESTCON 2002 introduced us to the development of a new standard known as S&TD, through a number of papers and presentations. ‘Signal Definition and Test Description – An IEEE Standard’ presented the idea of a standard which provides a framework for a component library of Basic Signal Components (BSC), which is extendable to incorporate subsequently emerging

technologies. ‘Synthesis Of Complex Signals On Test Equipment’ [2] introduced the concept of the Signal Graph to describe complex, ‘user defined’ signals which can be proven through simulation and then mapped onto real hardware to provide a portable ATE (Automatic Test Environment) signal descriptions. "UUT Test Requirements Capture in Any Language You Like… And Still Compliant with the IEEE Signal Definition and Test Description" described the language independence of the S&TD standard. Finally, "Modeling Complex Signals Using Basic Signal Components" went into the details of the Test Signal Framework (TSF) for creating custom signal components from the Basic Signal Components.

"Synthesis of Complex Signals on Test Equipment", in particular, described ways in which the Signal Modeling Language (SML) lends itself to use in synthesis of the designed signal on computers. And, it is that concept that this paper intends to take further.

2 Signal Definition & Simulation

S&TD defines Basic Signal Components (BSC), selected to provide the most basic signal and measurement functions that might conceivably required in any ATP. BSCs include such as Sine, Square, Sum, AM, RMS, FFT. To create more complex signals, these BSCs can be connected together. The standard describes BSCs and the links between them (e.g. In, Carrier, Gate) mathematically, through SML. SML allows S&TD signal definitions to be easily be interpreted to produce simulations of defined waveforms.

"Synthesis of Complex Signals on Test Equipment" [2] introduced the concept of Signal Graphs, to take advantage of the nature of Basic Signal Components. Just such a Signal Graph can be seen in figure 1.

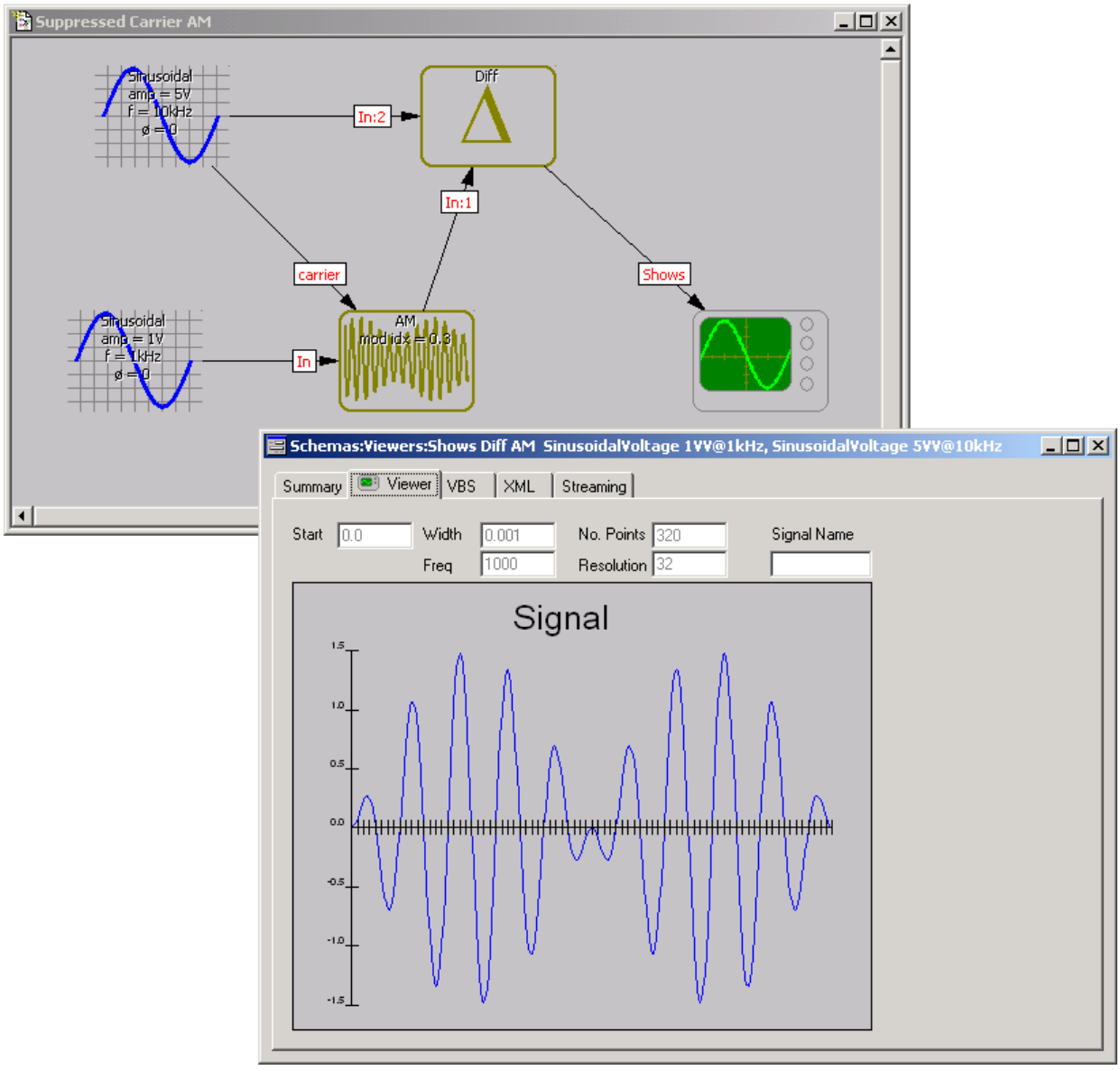

Figure 1. Signal Graph and Simulation

Figure 1 shows an example of how the Basic Signal Components might be connect to form a more complex signal. A Signal Graph gives a very good visual cue as to the nature of the signal that is describes. Also shown in Figure 1 is a simulation of the signal described in the Signal Graph, produced using the Signal Modeling Language from the STD standard.

SML provides the possibility of extremely accurate simulation results. Simulations created with SML can be used to prove a signal design and even generate points to populate digital memory devices, such as an arbitrary waveform generator [2]. As is often the case with discrete

mathematics, however, SML is not readily interpreted at a rate sufficient to meet many real-world, real-time applications, and simulations are effectively static.

3 Real-Time Simulation

3.1 Streaming Background

With respect to real-time simulation of STD defined signals, the most significant principle of Streaming technology is that of rendering linear data to a particular device, or devices (just as, say, a movie data file or a sound data clip might be rendered to a VDU and/or speakers). The mechanism by which the data is transferred from its original source to a 'renderer' is known as 'Streaming'. Streaming involves taking the linear data and passing it, as 'stream', through various 'Filters' until it reaches a final Filter that can perform the rendering to an output device. The number of Filters required to transfer data of a particular given type from its source to a particular given output device is dependant on the format of the data and the characteristics of the output device.

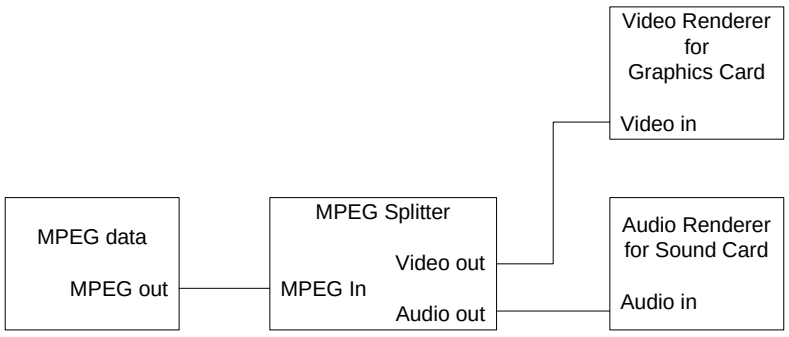

Figure 2. A Simplified Filter Graph for Rendering an MPEG File

Figure 2 shows a common example of streaming in multimedia, whereby an MPEG data stream is separated by a filter into separate audio and video streams, before being rendered to audio and video devices. Such an arrangement is known as a Filter Graph.

Connections in a Filter Graph are made between dedicated inputs and outputs on each filter, such that any given input type will only accept input from an output of the same type.

Each Filter in a Filter Graph extracts or modifies the data in such a way as to make it more useful to Filters further on down the sequence (downstream).

3.2 Application of Streaming to Signal Graphs

In streaming we have an efficient method of dealing with complex signals, in a framework already similar to that defined by the STD standard, in that it connects basic building blocks together to create a more complex function.

It was envisaged that real-time simulation of STD defined signals could be realized by levering the streaming Filter Graph mechanism to meet the requirements of the Signal Graph. Filters could be created that would perform the same mathematical functions as Basic Signal Components, with an additional filter being defined to render the signal to the PC screen in the format of, say, a digital oscilloscope.

In SML we have a very precise functional description of how each of the STD Basic Signal Components behaves and connects to other Basic Signal Components. As we have stated above, the streaming framework works in a very similar way. In essence, all that is required is to map the SML for each of our Basic Signal Components onto Filters in a Streaming framework and we will have created an efficient new way to synthesize each of the Basic Signal Components in real-time, in a PC environment that is available to anybody.

It seems that the Filter Graph of multimedia streaming maps very nicely onto the Signal Graph of signal definition.

3.3 Implementation

After a brief period of investigation, it was decided that the DirectShow® [8] Streaming technology framework already provided a lot of the functionality that we required in order to implement a real-time signal Streaming framework. To this end, a set of code-classes, derived from the DirectShow® classes, were produced to provide behavior that modifies and controls the Streaming technology, such that it models the framework laid out by the STD standard.

Filters were developed to represent each of the STD standard's Basic Signal Components. The mathematics behind each Filter being described by the standard's SML.

Each Basic Signal Component can have a number of different inputs; Two inputs that are common to all filters (Sync and Gate), but which need not be connected, and one or more inputs that are specific to individual Basic Signal Components (or specific group of Basic Signal Components), but which must always be connected (for example, Carrier). Each Basic Signal Component produces a single output signal, but it may be duplicated for connection to more than one other Basic Signal Component. By contrast, Filters must always have a pre-defined number of inputs, and usually a single output.

To achieve the Basic Signal Components' connection configurations in Filters, it was necessary to create default connections for those inputs which need not be connected and make number of output connections dynamic so that signals could be routed to multiple destinations. These default

connections had to be made to Filters, specially created to have no effect on the behavior of the Basic Signal Component. A special Filter was also created to dynamically split the output signal, whenever more than one destination is specified in the Signal Graph. For example, the Sum Basic Signal Component has Sync, Gate, Input0 and Input1 inputs, but whenever the last input pin is connected, another one is created, allowing the summation of more than two signals.

In order to create source BSCs, a generic 'signal generator' filter was created. This, typically, acts as the starting point in the Filter Graph. Because, unlike the Signal Graph, Streaming is acting on real digital signals, a sampling rate has to be specified for the data, so this is specified in the signal generator and is communicated down through the Filter Graph so that each Filter is configured for that rate.

The derived signal streaming filter classes had to be designed to handle the synchronization of the digital signal data as it flows through a Filter Graph. This is required because there could be several routes that different signal streams follow before they reach, say, a Sum Filter, and Sum can only perform its function when it has received the data from all streams. To deal with such situations, it was necessary to add a time stamp to the data stream to enable synchronization of the data to be combined. This can be likened to such multimedia Filters as video encoders, where, for example, the audio stream must coincide with the video stream to maintain lip-sync.

3.4 Defining the Signal

Within the STD standard are a number guides for transport languages. Although, so far, Signal Graphs have been used to represent signal definitions, the same definition can be achieved through such as XML [5] or VB [6]. Figure 3 show the same signal definition as that shown in figure 1, but, this time, represented in XML.

```
<?xml version="1.0" ?> <!--encoding="UTF-8"-->
<!-- created with Racal Instruments newWave (http://www.racalinstruments.com) -->
<!-- W3C Schema generated by newWave (http://www.racalinstruments.com) -->
<Signal name="" Out="Diff3"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:noNamespaceSchemaLocation="..\signals.xsd">
       <SinusoidalVoltage name="Sinusoidal9" amplitude="5V" frequency="10kHz" />
       <SinusoidalVoltage name="Sinusoidal10" amplitude="1V" frequency="1kHz" />
       <AM name="AM5" carrier="Sinusoidal9" In="Sinusoidal10"/>
       <Diff name="Diff3" In="AM5 Sinusoidal9"/>
</Signal>
```

Figure 3. Signal Graph and Real-Time Simulation

As an extendable, generic standard, XML seemed an ideal transport language to carry the signal definition to our real-time simulator. It makes possible a simulation tool that is independent from any signal design environment, and that can be incorporated into any application that conforms the STD standard and is able to parse XML.

3.5 Realization

Figure 4 shows the same Signal Graph as seen earlier, but here the Basic Signal Components are being simulated through Filters, written using code classes derived from the DirectShow® framework. A real-time graphical display Filter has also been produced to fit within the DirectShow® framework, and this is being used to render the simulation of the signal. Though it is not entirely evident from the figure, the simulated signal changes dynamically, in response to any changes that are made to the Signal Graph. This is a significant improvement on the previous, 'static', simulation, in which the whole simulation would have to be restarted for the sake of a single change on a single node. It is because each individual node in the real-time simulation generates its signal independently of the others, rather than each node's output being calculated on the basis of the nodes that came before it, that this is possible.

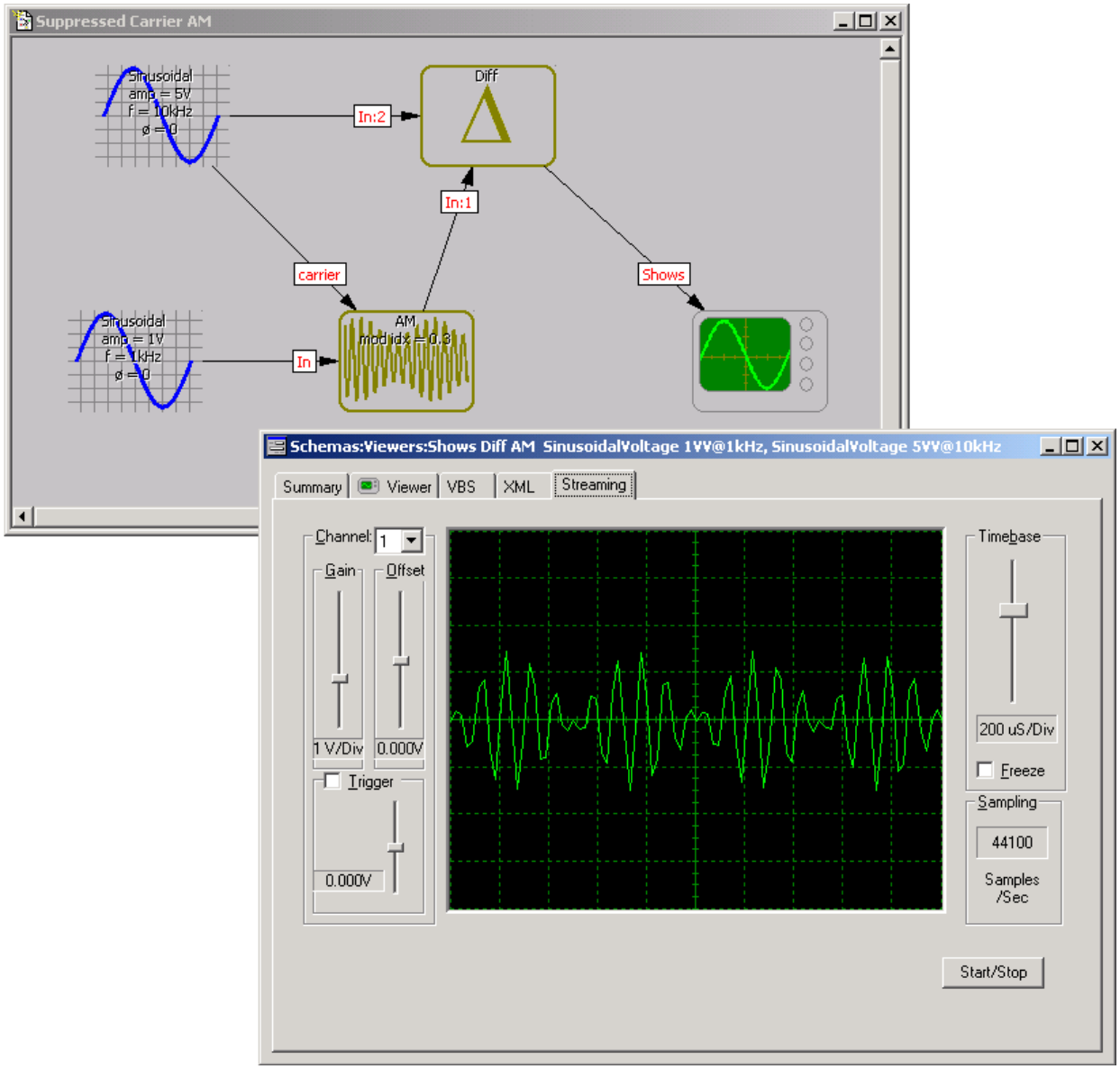

Figure 4. Signal Graph and Real-Time Simulation

With a relatively small amount of effort, since an efficient framework has already been provided, in the form of DirectShow®, it has been shown how it is possible to simulate STD Basic Signal Components in real-time. 'Real-time' is obviously limited by the speed of the host computer's processor and the efficiency of its peripherals. However, in tests, real-time simulations of STD signals have been we have successfully created up to 1 MHz on a 600 MHz PC; Well into the rf band.

4 Real-Time Signal Synthesis

A significant difference between the original simulation (figure 1) and the Streaming simulation is that, in Streaming, data is actually flowing between the Filters (Basic Signal Components) of our Filter Graph (Streaming equivalent to the Signal Graph); Everything is happening at the same time. In the original, functional SML, the final output signal is a product of each node (Basic Signal Component) in the Signal Graph acting on those nodes that are up-stream of it. In the streaming simulation, data is flowing between each of the Filters at the same time, just as it would in real-world instruments.

In terms of a simulation for the real world, this notion of independent nodes is certainly more like the hardware on which the signal might be implemented. We know this because in the real world we can connect a voltmeter to any of the terminals in our rack. So, we are gradually getting closer to the real world, but we still haven't left our design tool.

So far, it has been shown that by mapping STD Basic Signal Components' SML onto DirectShow® Filters, an STD signal definition may be Streamed in real-time to, for example, a graphical display for simulation. Now that we see our simulation is not just mimicking the final output of our Signal Graph, but that we could look at any connection between any two nodes and see the signal actually passing between them, we can start to wonder whether we can stream that signal to any devices other than the screen. And, why not; The DirectX® framework doesn't care whether it is moving data into the screen memory or memory on a digital to analog converter card, plugged into the PC's bus.

Since the DirectX framework is particularly efficient at getting streamed data to PC hardware, why not a digital to analogue converter (D to A). This would mean being able to actually generate the signal in the real world, through the same environment that was used to design and prove the signal. And, not just signals, but also measurements, since a Streaming Filter may just as well take data from an analogue to digital converter (A to D) and provide is as a Stream to the Filter Graph.

A relatively easy to make first step towards proving this idea was to use one of the Filters already provided by the manufacturer of the sound card, on the test PC, to render a signal design to the sound card. Of course, there is no Basic Signal Component for a sound card, so a special tag was used to denote this in the XML. Now, signal definitions made in the Signal Graph were output from the PC sound card. Measurements could also be made through the sound card's analog to digital converter.

Of course, the frequency range of a PC sound card is not great, but they are produced in such quantity as to make them very affordable and high quality, accurate models are commonly available. Suddenly, our generic signal definition tool has become an audio analyzer.

There is clearly a quantum leap from being able to manipulate audio frequencies to transferring data at the rates required for to produce radio frequencies, or higher, but this solution is a very flexible one. The use of a D to A or A to D converter to this system means that literally any signal can be generated or measurement made. Moreover, the definition is portable, through an internationally recognized standard, and can be implemented on any number of other devices as need necessitates.

Should a signal be required of greater frequency than can be delivered by this system, it has already been shown [2] how the stream data can be down-loaded onto a digital memory device, such as an arbitrary waveform generator, for later output in real-time. The benefit of using this Streaming mechanism to generate the data points required for this operation is that the data can be generated far more quickly (certainly as fast as it can be downloaded to the ARB) than it could, using the functional SML synthesis method.

5 Conclusions

We have devised a system that will enable us to design, simulate and implement complex stimulus signals and measurements at useful frequencies on a single machine. We have used the IEEE STD to ensure an international standard, providing flexibility, portability and a natural source for simulation, through the standard's SML.

Through a DirectX® framework, equivalents for the STD Basic Signal Components have been implemented in a mathematically oriented language, rather than the functional language and real-time Streaming simulations of our signal definitions produced on an on-screen display.

Using commonly available PC hardware, we have created real-time, a real-world signals with genuine applications. Where the physical limits of the PC's data though-put are met, Streaming can be used to populate digital memory devices virtually instantaneously.

6 References

[1] *Signal Definition and Test Description – An IEEE Standard*, AUTOTESTCON 2002, Keith Ellis, et al.
[2] *Synthesis of Complex Signals on Test Equipment*, AUTOTESTCON 2002, Matt Cornish, et al.
[3] *UUT Test Requirements Capture in Any Language You Like… And Still Compliant with the IEEE Signal Definition and Test Description*, AUTOTESTCON 2002, Chris Gorringe et al.
[4] *Modeling Complex Signals Using Basic Signal Components*, AUTOTESTCON 2002, Dick Delany, et al.
[5] *Extensible Markup Language (XML) 1.0 (Second Edition)*
[6] *VBScript Language Reference, Microsoft, 1997*
[7] *DirectShow, MSDN Library – July 2001*, Microsoft Corp.