IEEE 1641 supports test and design

Matt Cornish - February 01, 2007

Companies in the aerospace and defense industries use equipment for dozens of years, and these line-replaceable units (LRUs)—such as radar, radio, and positioning systems—require test and calibration throughout their lifetime. Yet, building a test system that will serve an LRU for its entire life is a nearly impossible task.

While test equipment has a longer life span than, say, PCs, an instrument is likely to need replacement before an LRU is taken out of commission—but after the manufacturer has ceased producing that particular model. Thus, engineers often must replace obsolete test equipment with different models, and such replacements usually require software changes to both the test system and the test itself.

IEEE 1641-2004, “IEEE Standard for Signal and Test Definition,” addresses the problem by defining a structure for test signals that makes software portable among equipment (Ref. 1). By employing the IEEE 1641 structure, you can replace a piece of test equipment with minimal impact on a test system.

The IEEE 1641 structure may also be the basis for synthetic instruments, which use analog-to-digital and digital-to-analog converters (ADCs and DACs) to provide the functionality of multiple test instruments. With the structure, you can also develop simulations, which are useful during product design. IEEE 1641 is also incorporated into other software standards, such as Automatic Test Markup Language (Ref. 2).

What is IEEE 1641?

IEEE 1641 lets you define signals purely in terms of the LRU under test, which makes it easier to
replace instruments in a test system and makes the system itself more resistant to obsolescence.

You can define test signals by connecting Basic Signal Components (BSCs), which form the basic building blocks of any test signal; their attributes let you define a test signal. The main groups of BSCs are sources, conditioners, events, measurements, digital, and connections. A Test Signal Framework (TSF) is a high-level grouping of BSCs designed to provide a reusable block of signal functionality.

A mathematically based Signal Modeling Language (SML) lets you precisely define the signals within a BSC. You can use SML to form a complete functional model of a signal. Thus, you can use it with traditional test instruments or synthetic instruments to create live signals, or you can use it for simulations (virtual signals).

You can arrange BSCs and TSFs together to form a complete test signal, known as a signal graph. You can define signal graphs either graphically or through text. If you use text, use Extensible Markup Language (XML), which is both machine readable and easily formatted for human reading.

You can also use Interface Definition Language (IDL), which lets you create a standardized programming object such as an ActiveX object to represent a signal graph. You can then call the object from most common programming languages.

![Figure 1](image)

**Figure 1.** A resource manager is the intermediary between signals and instrument drivers.

Because IEEE 1641 defines signals as they relate to an LRU, you need a way to translate the signal definitions onto physical signals. You can use a resource manager to map the BSCs and TSFs to a system’s instruments and route signal data to the instruments through drivers. **Figure 1** shows the resource manager as the intermediary that maps signal definitions to instrument drivers. “Implementing IEEE 1641,” p. 48, outlines the steps required to put IEEE 1641 into practice.

**The details of Basic Signal Components**

BSCs contain the attributes that control the behavior of a signal. Examples of AC-signal attributes
are amplitude, frequency, and phase. A BSC for an FM signal could consist of a signal source and a signal conditioner (Figure 2). Both the source and the conditioner have attributes that define the FM signal, such as amplitude, frequency, and frequency deviation.

![Figure 2](image)

**Figure 2.** A BSC for an FM signal consists of a signal source and a signal conditioner, both of which contain attributes of the signal.

Some commercial software programs offer a library of BSCs that can make it easier to construct an IEEE 1641-compliant test program. Compliant programs assign default values to the attributes in the BSCs, which simplify your work even further. For example, the phase attribute of a sinusoid is frequently not important and you can ignore it. The default value for amplitude is zero for safety reasons. An unexpected output from a device where amplitude has inadvertently not been specified could cause damage or injury. Attributes can also have default units. For example, a sinusoid’s frequency has hertz as its default unit. You can, though, specify a signal’s frequency as a period in seconds instead.

BSCs also have “In” connections between them. The “In” connection applies to a signal passed to one BSC from another. An exception to this rule is made for the case of Carrier, where you must distinguish between the modulating signal and the carrier signal. Usually, no such distinction is required, as in the case of summing several signals. An “Out” connection defines the output of the complete signal, which should be applied to, or measured from, a pin on the LRU under test.

Just as BSCs can define signals, they also can define events. Thus, you can use events to synchronize or “gate” physical signals. For example, you can gate a sinusoid (used as an RF carrier) with a clock to produce RF bursts as part of a radar signal.

**Tests require measurements**

Measurement BSCs can be set up to let instruments capture signals and produce a required value, such as RMS or Peak (Figure 3). BSCs can also compare a measured value with upper or lower limits to yield a pass/fail result. They also raise an event for use as a gate or for synchronization when a particular condition is met.
Digital BSCs include logic levels, data rate, and clock rate. They address the digital characteristics of the signal, such as the voltage for a logic 1 (5 V) and a logic 0 (0 V). Don’t, however, confuse these nominal voltage levels with the “analog” attributes of digital signals such as rise time and overshoot.

To create connections between signal graphs and the physical test points on an LRU, you can use connection BSCs. A connection BSC’s attributes are typically pin names that describe a contact within a connector on what would commonly be the unit under test (UUT). Once you define your signal model using BSCs, you can create a TSF. **Figure 4** illustrates two radar TSFs.

The SML description of signals contained in BSCs and, thus, TSFs lets you build a signal that accurately simulates and generates waveform data. You can use SML to verify signals or produce waveform data for synthetic instrumentation.

SML is not, however, readily interpreted at a rate sufficient to meet many real-time applications. Simulations produced using SML are effectively static.

**Streaming signals**

IEEE 1641 lends itself to a streaming framework that provides an efficient method of dealing with complex signal definitions. A common example of streaming is in multimedia systems, such as MPEG data pipelined between “filters” that decode a data stream, split the stream into audio and video streams, and render these video and sound streams to the screen and speakers.

Filters are the streaming equivalent of BSCs, and a filter graph is directly analogous to a signal graph. Coding of BSCs as streaming filters, through an existing framework such as DirectX, provides a system for simulating and synthesizing changing signals using ADCs or DACs.

The streaming framework lets you produce and measure dynamic signal data and plot the results. With most current PCs, you can simulate, but not reproduce, a real-time signal up to about 4 MHz.

With streaming, you can use a PC sound card to test communication headsets and telephony devices. Recently, PCI Express instrument cards brought fast DACs and ADCs to a faster data bus. Signals above 1 Msample/s can be generated live and on-the-fly from IEEE 1641 test programs through the
Many applications, however, require more accuracy and precision than a PC sound card or data-acquisition card can produce. For those applications, you could use an arbitrary waveform generator and a digitizer.

When you reproduce an IEEE 1641 signal-graph within a streaming-filter framework, you can transform signal definitions into signal data. By rendering this signal data visually, you can create a simulation against which you can verify a signal definition. Typically, a simulation requires points of reference and measures of scale, such as the voltage, frequency, and time scales found on oscilloscopes and spectrum analyzers.

**Figure 5.** This resource description of an RF synthesizer was constructed from signal graphs that contain BSCs.

Using BSCs to create resource descriptions creates a common interface between test programs and instrumentation. **Figure 5** shows a signal graph that describes a test resource (an RF synthesizer).

Typically, a resource manager views the definition of the test signal and maps it onto a list of descriptions it carries for the instrumentation. Here, the XML defined in the standard becomes useful. Because XML is well structured and supported by software tools, you can easily parse data to find a match between the signal definition and the resource description.

IEEE 1641 test programs fully define the signals you must apply to, or measure from, an LRU. Thus, you have available all the information you need to source new instrumentation. Furthermore, SML enables simulations that provide validation that the output or measurement from a replacement instrument meets your requirements.

That said, many instrument-replacement programs falter because test specifications aren’t described rigorously enough. For example, a replacement instrument with an improved frequency response may seem to be a “better” choice. Unfortunately, an unspecified (or, more likely, unconsidered) part of the test might have been to filter off high-frequency harmonics. Now that they can be measured, the test fails.

IEEE 1641 does not force you to rigorously specify all of the details in a test program, but it does enable you to specify the attributes that improve obsolescence management. Of course, as in all cases, you still need good engineering practices to be assured of success.
References


2. Automatic Test Markup Language (ATML) project information, IEEE, Piscataway, NJ. grouper.ieee.org/groups/scc20/tii/