IEEE Std 1641

A Revised Signal and Test Definition Standard

Chris Gorringe
EADS Test Engineering Services (UK) Ltd
23–25 Cobham Road, Wimborne
Dorset, BH21 7PE, UK
chris.gorringe@eads-ts.com

***Abstract*—Over the past three years, the IEEE Std 1641 Signal & Test Definition standard has been evolving and maturing. This year its first revision is set to come out. This paper discusses the changes and improvements incorporated within the revised draft standard and identifies how this standard has been integrated into other test standards, such as ATML.**

In most cases, the revised standard clarifies existing practices and removes inherent ambiguity. The new revision is now based far more on its XML pedigree, deferring its original ATLAS roots, whilst still supporting those legacy signals.

The revised standard has also introduced some new components to better support digital and protocol testing. These new components bridge the gap between analogue and event signals, something that the original standard supported but did not provide any detail on. The revised standard now provides a comprehensive set of building blocks supporting analogue, events, and digital signals.

Keywords-ATE; ATS; signal modeling; test definitions; test requirements; test signals; IEEE Std 1641

I. INTRODUCTION

IEEE Std 1641[1] has now been published for 5 years and has just started its revision approval process with the IEEE. As part of this process the standard has been revamped. Most of these changes do not represent new items just clarifications and removal of ambiguities. Key to this revision process has been the work to improve the standard whist maintaining backward compatibility. What is new is how various aspects of the standard is described. Some of the key areas for this improved definition include:

- signal states
- multi channel behavior
- measurement transforms

During this period the IEEE Std 1641 has seen a growing number of standards reference it for its signal modeling capabilities. ATML [2] makes reference to this standard to help with Test Description and its Hardware Capabilities descriptions [3]. The adoption for this standard to describe signal modeling has led to IEEE Std 1641 being mandated in both MoD ATS Policy [9] and in the DoD's Information Technology Standards and Profile Registry (DISR) [4][6].

II. SIGNAL STATES

Within IEEE Std 1641 a signal can exist in one of four states. There are many types of signal, e.g., physical, event, and digital. At any time during their existence, all signals will be in one of these four states [5].

These states can be summarized as follows (see Fig. 1):

- Z-No Signal (High Impedance) (- . - . - .)
- X-Gated off (High Impedance) (……)
- L-Inactive event (- - - - -)
- H-Physical signal value

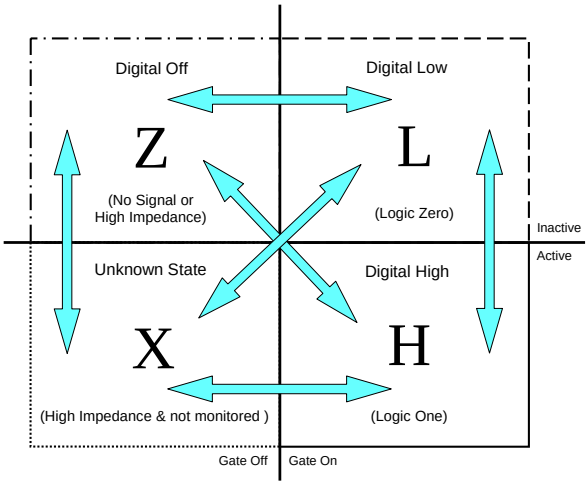

Figure 1. Available Signal States

It is worth considering examples that demonstrate each of these states. A simple Sinusoid produces a signal in the "H – Physical signal value" state. Gating the Sinusoid produces an intermittent "X-Gated Off". Synchronizing this signal highlights the Z-No Signal state (see Fig. 2).

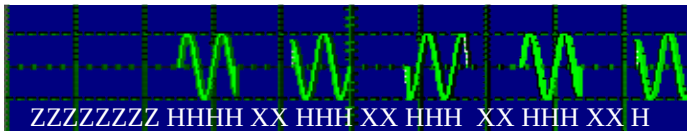

Figure 2. Physical Signal containing ZHX states

Equally we can construct a signal containing HLZ states. Consider a simple gated Clock event as shown in Fig. 3.

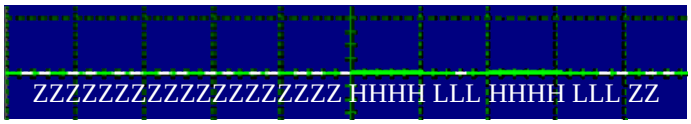

Figure 3. Event Signal containing ZHL states

These different states can be distinguished by their effect on other signals, This was shown in IEEE Std 1641.1 [7] in which the behavior of gated signals showed the difference between the X and Z states. Equally we could construct signals that contained all four states, e.g., consider combining the above signals using a logical AND as shown in Fig. 4.

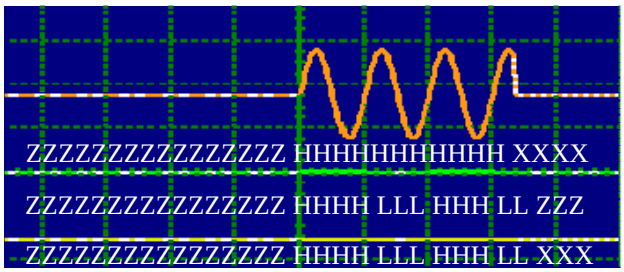

Figure 4. Signal containing four ZHLX states

The updated standard has embraced the fact that signals could have contained all four states and used these to describe digital signals [8]. It's even added new BSCs Encode and Decode to create and manipulate these signals directly. However, having four states introduces additional complexities on our existing BSCs. In IEEE Std 1641-2004 signals such as summing two events or Xor of two sinusoids was not really shown as examples. We now see that any behavior has to be fully defined for all signals states.

The following represent the rules framework which must be true.

1. A signal used as a Gate or Sync identifies the signal Active/Inactive. This means that any Signal in the HX states acts as Active and any signal in the ZL states acts as Inactive. An interesting observation for Gate & Sync is that having no connection (to the Gate or Sync) behaves differently to having a connection but with no signal (i.e. in the Z state).
2. When a Signal is used as an Input (and not a Gate or Sync), the operation performed on a signal in the Z state is identical to the operation that would have been performed had the signal not been connected.
3. Basic logical operators always apply
    - NOT NOT x = x
    - NOT(x AND y) = (NOT x) OR (NOT y)
    - NOT(x OR y) = (NOT x) AND (NOT y)
    - x XOR y = (x OR Y) AND NOT (x AND y)
    - x XOR y = (x AND y) OR NOT (x OR y)
    - x AND L = L
    - x OR H = H
4. NOT Z-state is the Z state. This was shown in the IEEE Std. 1641.1 user's guide [7], the output of Not before a signal is Sync'd (i.e. in the Z state) is in the Z state.
5. For conditioners with only one input the output state matches the input state

From these we can derive the following states for NOT, OR, XOR, AND, Sum Product, Negate. Note that we do not need to describe BSCs that can themselves be compounded. An example would be ***Diff*** which can be described as a ***Sum*** of a ***Negated*** input, since both exist in the table we can infer that ***Diff*** will have the same states as ***Sum***.

Table 1 identifies the output state of the signal derived from the state of its inputs. The last (blue) column is the state of the signal if the BSC is Gated Off (Inactive).

Within the table the following key applies:

- Items in green are specified with the 2004 standard.
- Items in white are derived from the framework rules.
- Items in blue are inferred from other BSC behavior (AM, FM & PM)
- Items in yellow are required to be specified in the standard

Using Table 1 and the standard, one can consider what happens to other BSCs when input signals are not in their usual state. As an example consider AM modulation. AM modulation could be modeled as the product of a Carrier with the sum of modulation signal and constant (see Fig 5).

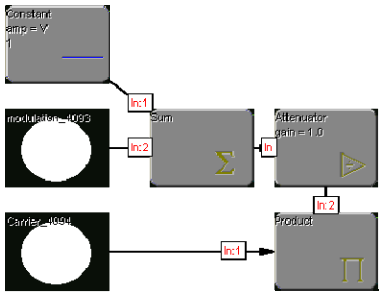

Figure 5. Simple AM Model

If the Modulation signal becomes gated off (X) we want the signal to be the Carrier; but if the Carrier becomes gated off we want the output to be Gated Off, i.e. no signal value. This is exactly the result that Table 1 provides.

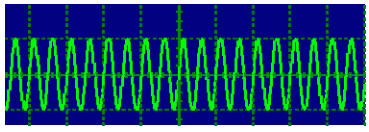

Figure 6. AM Signal with no modulation (X) just provides the Carrier

TABLE I. FOUR STATE LOGIC TABLES

	Negate		Sum(Diff)				Product				Or				And				XOr				Not	
Z	Z	Z	x	h	l	Z	x	h	l	Z	x	h	l	Z	x	h	l	Z	x	h	l	Z	Z	Z
X	X	X	X	h	X	X	X	X	l	X	X	h	l	Z	X	h	l	Z	X	0	l	Z	X	Z
H	H	X	H	H+h	H	X	X	H*h	l	X	H	H	H	Z	H	H	l	Z	0	0	H	Z	0	Z
L	L	Z	x	h	L	Z	L	L	L	Z	L	h	L	Z	L	L	L	Z	L	h	L	Z	1	Z

III. MULTIPLE SIGNAL CHANNELS

Another clarification of the standard is the formal consideration of the effect multi-channel inputs. In general BSCs (and therefore by derivation TSFs) are explained as single channels signals. However their behavior is applicable for multi-channel signals. With the advent of digital signals there is a greater prevalence of multi-channel signals with IEEE 1641 models and therefore the need to describe all potential combinations.

Multi channel signals are not new. The STDTSFLib contained two TSFs containing multiple channels, i.e. SYNCHRO and RESOLVER. With these signals we could consider conditioning the output, e.g., using Negate. What happens is each channel is negated. The axiom is true for any BSC defined as a single channel; it operates on each signal. It is basically equivalent to taking each channel in turn and applying that operation

Diagrammatically, we could consider this as shown in Fig. 7.

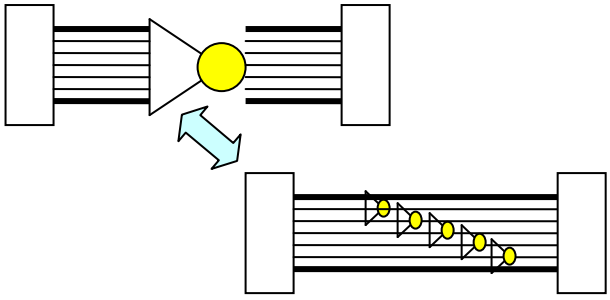

Figure 7. Unitary Operator Equivalence

Equally if we were to Sum two SYNCHRO inputs then the result would be the sum of their consecutive channels (See Fig 8). If extra channels are required then channels in the Z – No Signal state are added.

Finally, in the case where an element is combining a single channel signal (scalar signal) with a multichannel signal (vector signal) the scalar channel is applied to each multi channel in turn. An example would be applying a dc offset to the SYNCHRO signal which applies the dc offset to all channels.

In general, when applying signals with different number of channels; vector signals gain additional Z-NoSignal channels, whilst scalar channels (single) are duplicated the appropriate number of times.

Although this is true for inputs it is a different case for a ***Gate*** and ***Sync*** inputs. These are special case inputs, and can only trigger when the signal as a whole is *Active*. For a single channel (scalar) signal this represents the channel state. However for a multi-channel signal (Vector) we want to know when the signal as a whole is *Active*, not necessarily when individual channels are *Active*.

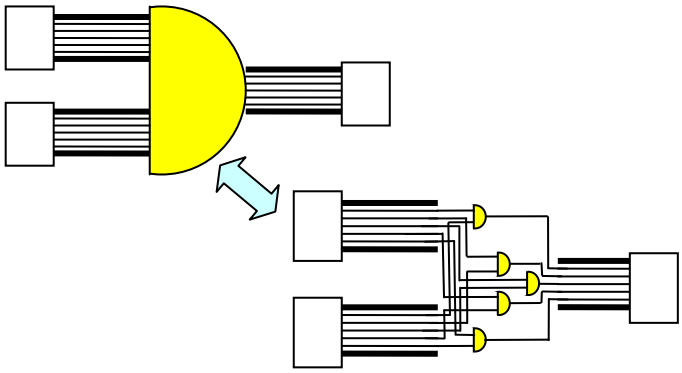

Figure 8. Multi channel operators act on each pair of channels.

In the case of a SYNCHRO signal used as a ***Sync***, this is when the signal 'starts', However with the introduction of digital signals and digital streams care needs to be taken. The solution is defined in the new standard as a multichannel signal (Vector). This signal is considered *Active* when one or more of its channels are not in the Z – No Signal state. Equally this could be represented as the multi-channel signal being *InActive* when the logical AND of all channels (x OR !x) is *InActive.*

IEEE 1641 does not have any way for a user to define a single channel vector signal. However, the effect can be achieved by creating a dual channel signal by adding an extra Z – No Signal channel. This can be modeled by using a **Connection** or **Channel** BSC to turn a single channel into a multi-channel signal as illustrated in Fig. 9 and Fig. 10.

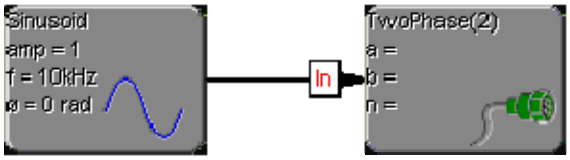

Figure 9. Use a Connection to turn single channel scalar signal into a multi-channel vector signal

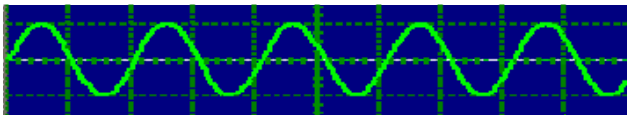

Figure 10. Dual Channel Signal with Z–NoSignal as an additional channel

IV. Measurement Transforms

One of the subtle changes introduced in the revised standard is the introduction of abstract signals. In effect an abstract signal is something that can be manipulated by 1641 but should not be directly applied to the UUT. IEEE Std 1641-2004 already had event signals that should not be applied directly to a UUT, the abstract signal basically adds values to these signals.

Within IEEE Std 1641-2004 the output of all **Sensors** was represented as an event stream. The IEEE Std 1641.1 clarified this usage and introduced the notion that the output could also have signal values (an abstract signal). When a Sensor was acting as a Measure this output was *Active* after all the measurement(s) had been taken but when the Sensor was acting as a Monitor the output was *Active* and had a value whilst the monitored condition was being met. In addition, the guide identified how the use of *condition*, when a Sensor was being used for measurement, would act as a measurement trigger in a similar way to the use of the ***Gate*** attribute.

The revised standard builds on these, and clarifies additional measurement features, which provide a sophisticated set of specification components. One of these key features is the use of generic **Measure** as a measurement transform.

As an example, consider the following model, the abstract signal coming out of the model is equivalent to the signal going into the model. The use of a generic "Measure As" with an 'As' conditioner acts as an inverse transform. In this example the inverse transform of Negate is Negate. But this technique can be applied to any model containing an input, e.g., In, Gate, Sync, Carrier, etc (see Fig. 11 and Fig.12).

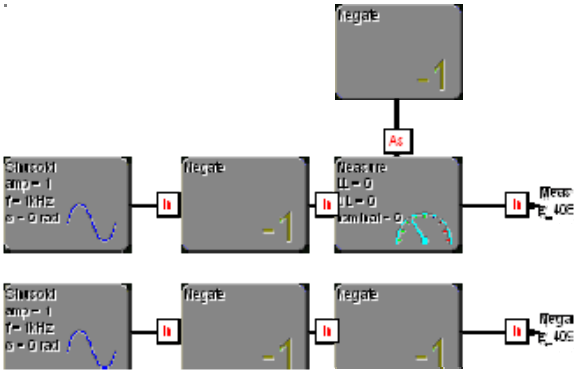

Figure 11. Simple Inverse Transform and output

```
<?xml version="1.0" ?>
<Signal Out="M4080" xmlns="STDBSC">
 <Sinusoid name="S4075" amplitude="1"
          frequency="1kHz" />
 <Negate name="N4079" In="S4075" />
 <Negate name="N4084" />
 <Measure name="M4080" samples="0" As="N4084"
          attribute="In" In="N4079" />
</Signal
```

Figure 12. Simple Inverse Transform XML

This follows the basic template shown in Fig. 13.

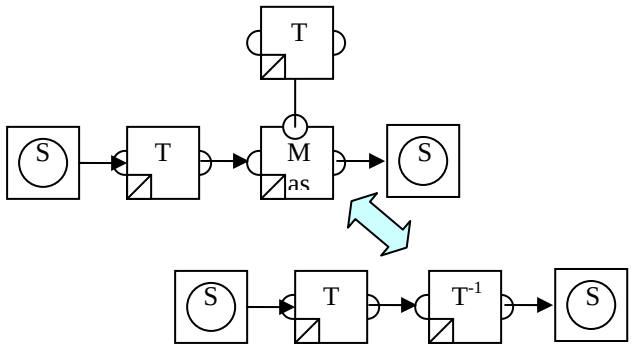

Figure 13. Inverse Transform Template

In practice, the Measure Transform can be applied to any signal, and the abstract output of the measurement will be transformed such that should the conditioner be applied the original signal would be generated. Another way to consider this is as shown in Fig. 14.

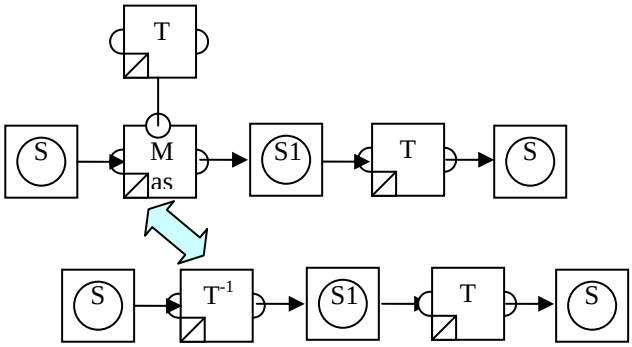

Figure 14. Inverse Transform

In the case of our 'Negate' Transform, we can transform any signal, and by apply Negate to the output we end up with the same signal. Another example would be to use an Exponential (decay) Transform. However, these are all examples where the Inverse Transform is known as shown below and in Fig. 15.

- $Negate^{-1} = Negate$
- $Exponential(x)^{-1} = Exponential(-x)$

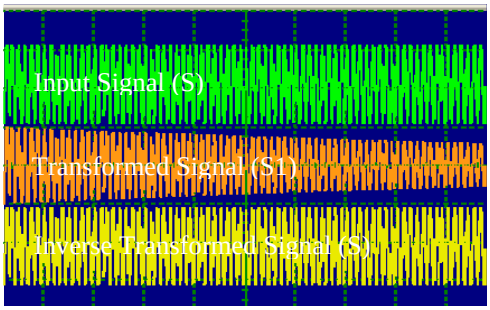

Figure 15. Inverse Transform Examples

The true power of the inverse transforms is when the Inverse transform is not known, or is implemented by some intellectual property of the test system. Measurement transforms can still be used to specify what is required but do not describe how to implement it. The technique allows us to specify what is required, not how it is achieved.

One obvious application is how to specify demodulation, specifically FM modulation. The output from the model is the demodulated signal (i.e. the signal before it was modulated). As an abstract signal we could continue to perform additional measurements or processing such as measure the frequency, harmonic tones, etc. Note that within the Model there is no indication of how this will be done, the only criteria is that should we again modulate the signal so that the final signal would match the original signal as describe in Fig. 16.

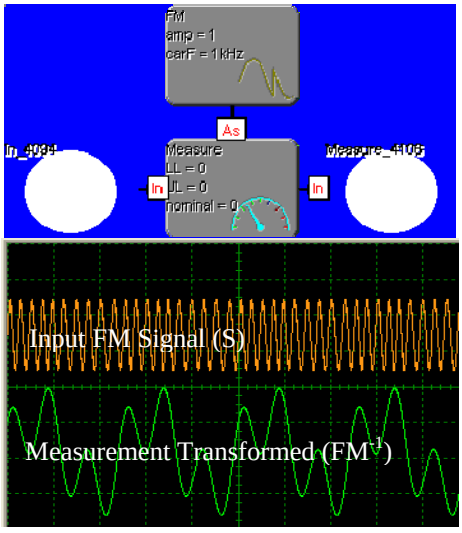

Figure 16. FM Demodulation

V. SUMMARY

The IEEE Std 1641 Standard has been revised. The new draft revision has clarified a lot of the edge cases that existed in the original standard, and unified the standard inline with the User Guide. In doing so it has clarified a new set of features that although present in the standard had not been immediately obvious or described.

The use of IEEE Std 1641 has been adopted by organizations needing to maintain signal information, and has been incorporated in part or as a whole within several interoperability demonstrations since its initial publication. This revision has taken a lot of the recommendations made from these demonstrations to ensure it continues to improve whilst maintaining backward compatibility.

The revised standard is currently within the ballot process, and is expected to be approved in late 2009. In a lot of ways it has not really changed, but so many ways we can define better Signal Models.

REFERENCES

[1] IEEE Std 1641™–2004, IEEE Standard for Signal and Test Definition. Institute of Electrical and Electronics Engineers, Inc.

[2] IEEE Std 1671™-2006, IEEE Trial-Use Standard for Automatic Test Markup Language (ATML) for Exchanging Automatic Test Equipment and Test Information via XML. Institute of Electrical and Electronics Engineers, Inc.

[3] C. Gorringe, T. Lopes; and M. Seavey, ATML and 'dot' Standards Status. IEEE AUTOTESTCON 2008 Proceedings, pp 69–73

[4] C. Gorringe, T. Lopes; and M. Seavey, ATML Completed Status - What Happens Next., IEEE AUTOTESTCON 2009, in press

[5] A. Hulme, Physical signals, events and digital streams – Their relationship and how they affect SignalFunctions in IEEE 1641. IEEE AUTOTESTCON 2009, in press

[6] M. Brown, K. Ellis, and A. Hulme, State of IEEE 1641 Standard, Applications and UK MOD Policy, IEEE AUTOTESTCON 2008 Proceedings, pp 295–300

[7] IEEE Std. 1641.1™–2006, IEEE Guide for the Use of IEEE Std 1641, Standard for Signal and Test Definition. Institute of Electrical and Electronics Engineers, Inc.

[8] C. C. Gorringe. Proposed changes to IEEE 1641™–2004 to better support digital testing. 04102/RPT/004, DE&S, UK MOD, March 2008.

[9] A. Hulme and K. Nash, Implementing IEEE 1641 – Using a complete system, IEEE AUTOTESTCON 2008 Proceedings, pp301–307

[10] C. Gorringe, I. Neag, and R. Taylor. ATML Demonstration – Readiness For Use. IEEE AUTOTESTCON 2009, in press

[11] C. Gorringe, ATML Phase I Demonstration, Final Report. October 2008. http://grouper.ieee.org/groups/scc20/ATML/Demonstrations/Phase1/index.htm

.