# Digital Signals in IEEE 1641 and ATML

## Fact or Fiction


Chris Gorringe
EADS Test Engineering Services (UK) Ltd
Wimborne, UK
chris.gorringe@eads-ts.com

Malcolm Brown
DES SE TLS SP ATS
MoD (UK)
dessetls-sp-ats@mod.uk

Teresa Lopes
Teradyne, Inc.
North Reading, MA, USA
teresa.lopes@teradyne.com



***Abstract*— The paper discusses the rational and benefits behind the recent MoD demonstration, using a fully compliant IEEE 1641 test system for both digital and analogue test programs, in line with the MoD ATS procurement policy.**

**This paper considers the support provided by IEEE 1641 to help define Digital Signals, within an overall signals framework. IEEE 1641 does not look to supplant other digital standards such as IEEE Std. 1445 (DTIF)[6], rather it looks to embrace these within its signal framework, allowing signals to be defined through board simulation or user defined digital signals all within the same standard framework.**

**The paper covers the new IEEE 1641 building blocks and presents a new set of Digital TSF components suitable for providing high level abstract digital, which have been used on a digital tester.**



***Keywords-IEEE Std. 1641; Digital Test; Digital Signal; Signal Modeling***


### I. INTRODUCTION

IEEE Std 1641 [3] has found itself being used in a variety of cases and referenced by other standards such as ATML where a signal orientated approach is required.

One area where there has not been significant usage or examples is the use of IEEE Std 1641 to create digital signal definitions.

However the MoD Automatic Test System (ATS) Policy, states all test programs should use standards as part of the Open System Architecture. Key to this policy is the use of IEEE Std 1641 to define tests used both for analogue and digital testing.

Following on from an initial "RAF 1641 Demonstration" during 2008[1][2], the MoD commissioned a programme to add IEEE Std 1641 digital test signals onto their existing General Purpose Automatic Test Equipments (GPATE) colloquially known as the Teradyne Spectrum, which utilize the M9 Series Digital Test Instrument for the digital test subsystem. These systems are used and operated through DSG, the RAF's repair center (formerly known as DARA & RAF Sealand).

The result of this joint effort has been a new set of digital Test Signal Framework (TSF) components specifically tailored to meet the requirements and work processes performed at DSG whilst being underpinned and defined using IEEE Std 1641. This approach represents a marked difference from previous signal related developments. In this programme we looked to build and expand upon the work practices already in use, such that we could incorporate many of the existing practices and experiences used at DSG within the ATS Policy, rather than embark on a change program with a new set of processes and digital tools.

The outcome has been to define and implement a working set of digital signal component TSFs that allow access to the inherent M9 digital signal capabilities, whilst maintaining a true signal abstraction that allows them to be defined and modeled by IEEE Std 1641. These form the core components that could be used for re-hosting onto different digital subsystems (a future requirement). In addition, the digital components defined within the IEEE Std 1641 basic components and ATLAS 716 TSF Libraries have been added so that the system can import native digital 1641 signals.

### II. DIGITAL SIGNAL VS STREAMS – A BIT OF TERMINOLOGY

Within the Basic Signal Components (BSC) and Test Signal Framework (TSF) defined by IEEE Std 1641 there are several predefined digital signals, namely; the SerialParallel and ParallelDigital BSCs together with the DIGITAL_SERIAL, DIGITAL_PARALLEL and the new bidirectional DIGITAL_TEST TSFs. All of these components are examples of physical digital signals that can be sourced or sensed directly onto the UUT, through a DigitalBus connector, and represent a physical signal such a voltage or current, where the various digital logic levels are represented by different physical values, e.g. -2 V to +5 V. (See Figure 1.)

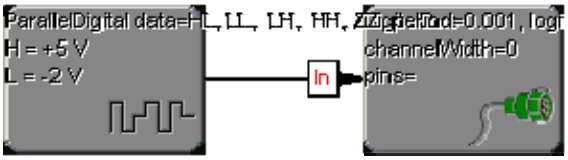


Figure 1. Example of a Parallel Digital Signal

The revised draft for IEEE Std. 1641 also describes the concept of abstract signals derived from events, which when grouped together create a special type of signal called a "digital stream". As an abstract signal a digital stream can not

be connected directly to a UUT. The digital stream must be first converted into (or from) a physical digital signal. Initially the use of these digital streams may appear superficial or even over-complex, and for simple stimulus or response examples, the original digital components offer a simple direct programming model. However, digital streams can be combined and manipulated to provide a method to produce complex digital signals (representing complex digital patterns Figure 2.), which can then be converted into digital signals (e.g. voltages -2 V to +5 V), something that became over complex when using the original physical digital components.

It is this ability to combine and manipulate digital streams and then convert them into digital signals for the UUT which is core to the development of new digital TSF components.

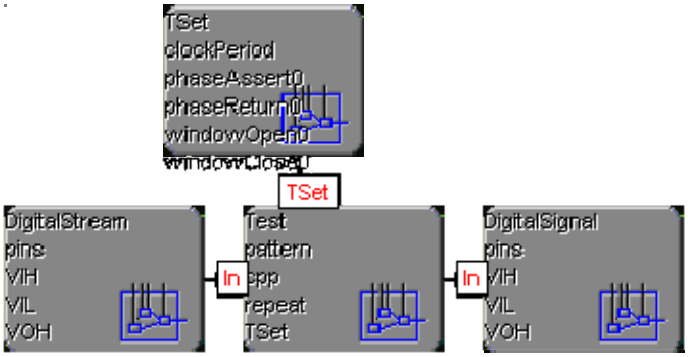


Figure 2. Equivalent Digital Stream Example

## III. The Digital DSG TSF Signal Components

The DSG TSF library consists of a set of new Digital Stream components used for the DSG digital signal modeling and cover the following:

- Timing Sets – **TSet**/**TSet4**
- Digital tests patterns – **Test**
- Pattern synchronization – **WaitFor**
- Failure jump patterns – **IfFail**
- Delays – **Wait**
- Pin Logic levels, channels and names – **DigitalStream** & **DigitalSignal**
- Import Simulation – **DTIF**

Each digital channel may use one of the following pulse classes:

- NRet – Non Return (Default)
- RZero – Return to Zero
- ROne – Return to One
- RComp – Return to Complement
- CompS – Complement Surround

Integrating with IEEE Std 1641 events:

- Use of standards Sync input provides initialization synchronizations (generally using power supply triggering)
- Use of channel input synchronizing provides user defined sync triggering e.g. 'listening'
- Use of output channels provides external synchronizing e.g. 'talking'

These digital components are used to create a digital signal model that can be used as a test program library component or TSF. The TSF provides an IEEE Std 1641 XML definition, and provides a standard programming interface (IDL) for using the digital signal within a test program. In addition the tool set converts these digital signal model directly into a native XML format that can be loaded directly onto the M9 through the API or Soft Front Panel using Teradyne's XML format of their digital test binary (DTB).

A DTB file is a binary file which stores all the information associated with a digital test. DTB files are created by the LSRTAP Importer, the SVF Importer and the M9 Soft Front Panel. The M9 Soft Front Panel can also be used to view and execute DTB files.

The digital signal models build up a sequence of digital components to create a digital stream. A digital stream represents an abstract multi-channel logical digital signal where each channel is considered to be in one of the four event states ZXHL (See Figure 3.).

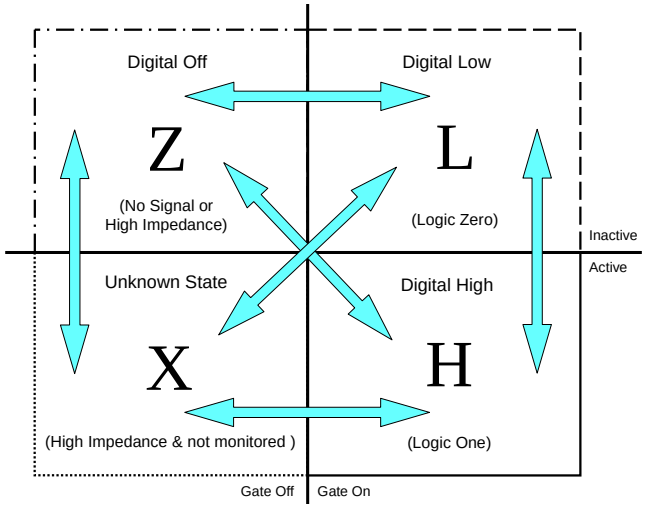


Figure 3. State diagram for a digital stream

The digital stream is then converted to a physical digital Signal through the DigitalSignal components, that uses the associated pulse class and window information defined in the TSet to create a digital signals suitable for connecting to a UUT.

By using these components DSG can capture and program their digital requirements, but at a higher abstract level. This leads to fewer unforeseen setups in the final instruments, where individual settings could be inadvertently altered or

needlessly copied from test programs to test program without effect.

### A. *DigitalStream*

Creates a digital stream from a physical signal based upon voltage/current threshold levels.

A digital stream output can be consumed by one of the following: **DigitalSignal**, **Test**, **WaitFor**, **Wait**, **IfFail**, **Channels.**

DigitalStream supports the following attributes:

- channels, pins
- VIH , VIL, VOH, VOL, IOH, IOL, VCom
- impedance, format, phase, window, capture

In addition to single pin names, pin groups can be defined and subsequently used by the pattern attributes.

Pins groups can be defined as a group of pins, e.g., Data [D3 D2 D1 D0] Add [A4 A3 A2 A1] Enable

Corresponding Channels are assigned either in the normal way or grouped for convenience, e.g., [1 2 3 4] [10 11 12 13] 100 is the same as 1 2 3 4 10 11 12 13 100

### B. *DigitalSignal*

This converts a digital stream back into a digital physical signal for use by a UUT by adding voltage/current levels, channels and pin names.

The output of DigitalSignal is a physical signal conforming to the correct logic voltage and pulse class.

Multiple DigitalSignals can be used with a digital signal to represent different groups of patterns, e.g., Pattern Sets.

The DigitalSignal should always be the last component of a digital signal model TSF.

The inputs for DigitalSignal should be a digital stream from one of **DigitalSignal, Test**, **WaitFor**, **Wait**, **IfFail**, or **Channels.**

DigitalSignal supports the following read-only attributes. If they are provided they should match the definitions in the corresponding DigitalStream component:

- channels, pins
- VIH , VIL, VOH, VOL, IOH, IOL, VCom

The pins, channels and voltage levels are inherited from the DigitalStreams in the rest of the digital model.

### C. *Test*

Provides basic test pattern data e.g. IOX _READY,

The **Test** component supports the following attributes:

- pattern – pattern data, logic opcode (IOX, IH, IL, OH, OL, MH, ML etc) followed by list of pins, terminated by semi-colon for next step
- cpp – clocks per pattern. Applies to all pattern steps within the test
- repeat – number of times to repeat the test pattern(s)
- delay – delay after pattern(s). Represents the dead time that will be added after repeated pattern. No test performed during delay
- TSet* – step(s) timing set, if its not provided its inherited from previous **Test**, **WaitFor** or **IfFail**

### D. *IfFail*

Tests for a specific pattern and jumps to a failing pattern sequence if the pattern fails

The **IfFail** component supports the following attributes:

- pattern – pattern data, logic opcode (IOX, IH, IL, OH, OL, MH, ML etc) followed by list of pins, terminated by semi-colon for next step.
- cpp – clocks per pattern: Applies to all pattern steps
- delay – delay after testing pattern(s); Represents the dead time that will be added after testing pattern(s).Only applies to GO Path. no test performed during delay
- TSet* – step(s) timing set, if its not provided its inherited from previous Test, WaitFor or IfFail
- failPattern – pattern to run if test fails – No additional patterns are then run.

Multiple Steps; only the first step containing a testable pattern is considered.

### E. *WaitFor*

Waits for a particular sequence of patterns on input pins.

The **WaitFor** component supports the following attributes:

- pattern – pattern data, logic opcode (IOX, IH, IL, OH, OL, MH, ML etc) followed by list of pins, terminated by semi-colon for next step.
- cpp – clocks per pattern Applies to all pattern steps
- repeat – maximum number of attempts to wait for pattern
- delay – delay after pattern(s). Represents the dead time that will be added after repeated pattern. No test performed during delay
- TSet* – step(s) timing set, if its not provided its inherited from previous Test, WaitFor or IfFail

Multiple Steps; Only the first step containing a testable pattern is considered.

### F. *Wait*

Waits a specific time without testing any pattern

The **Wait** component supports the following attribute:

- delay – delay without changing pattern(s). Represents the dead time that will be added. No test performed during delay

The Wait component finds the most appropriate TSet definition within the model to implement the delay. **Wait** uses a combination of Tset clock period, CPP and repeating the pattern to achieve the desired delay.

While waiting, **Wait** turns off testing and results capture.

### *G. TSet/TSet4*

Represents a timing set with either a single or group of four phases. The single TSet allows digital signal models to be defined that can then be ported onto digital platforms that support different TSet definitions. The TSet4 group requires a minimum hardware specification.

These components support the following attributes:

- period: Clock Period – must be multiple of the system or pattern clock
- phaseAssert(0-3) – time when pattern asserts
- phaseReturn(0-3) – time when pattern returns
- windowOpen(0-3)– time when measure window opens
- windowClose(0-3)– time when measure window closes

## IV. THE DIGITAL PATTERN STRING FORMAT

The digital pattern string format for IEEE Std 1641 is defined as a simple sequence of the characters "HLZX10" representing the corresponding digital states. The characters represent the digital signals as follows:

- H logic high (or logic 1)
- 1 logic high (or logic 1)
- L logic low (or logic 0)
- 0 logic low (or logic 0)
- Z high impedance (absence of logic signal)
- X unknown or indeterminate tri-state level
- , delimiter between blocks
- ; delimiter between blocks

The digitalString comprises a list of digital characters separated with delimiters. Each comma ',' or semi-colon ';' is treated as a delimiter between blocks. It may also include whitespace characters, namely space, new-line, carriage-return, line-feed, and tab. These whitespace characters are available for formatting purposes to make the data more readable. They are ignored when the digitalString is processed.

Even though IEEE Std 1641 defines a digital pattern string format, it was never intended that this would be the only format for all digital patterns strings defined within the IEEE Std 1641 framework. It was always envisaged that, through the use of TSFs and TSF attribute formulae, any digital pattern string could be defined with the proviso that the mapping onto the standard's simplified format was always provided as part of the TSF definition.

DSG have extensive experience in various digital languages and features used across various different digital test systems. From this they were able to define a digital pattern string format that provided the flexibility required and which could be mapped back onto the IEEE Std 1641 format through the TSF.

The digital pattern string format selected is based on the L200 format but also has elements borrowed from the Membrain format

The pattern supports two distinct formats:

- pin assignment
- pin state assignment

Pin assignment is written as "pinName=pinState" e.g. _READY=IL

Pin state assignment is written as "pinState pinList" where pinList is a list of 0 or more pin names e.g. IL READY RESET DATA[x5]

The following constraints also apply:

- pin names may not contain whitespace
- commas are ignored
- semi-colons represents new digital steps.
- Square Brackets[] represent a bus definition

TABLE I. SUPPORTED PIN STATES

| **Pin Codes** | **IEEE Std 1641 Pinstates** | | |
|---|---|---|---|
| | ***Description*** | ***Stim*** | ***Resp*** |
| IOX | High Impedance Tristate | X | X |
| IH | Input to UUT High | H | X |
| IL | Input to UUT Low | L | X |
| OH | Output from UUT High | X | H |
| OL | Output from UUT Low | X | L |
| MH | Input and monitor to UUT High | H | H |
| ML | Input and monitor to UUT Low | L | L |
| IHOL | Input to UUT High and Output from UUT Low | H | L |
| ILOH | Input to UUT Low and Output to UUT High | L | H |
| Keep | Keep same state as previous | - | - |
| TOG | Toggle previous states | X➔X<br>H➔L<br>L➔H | X➔X<br>H➔L<br>L➔H |

The pattern format also supports Pin groups. The logic codes for a group of pins (e.g., Data) can be defined as follows:

- IH Data -All data pins go to IH, e.g., IH D0 D1 D2 D3
- IH Data[x3] -Data pins are set IH D0 D1 IL D2 D3
- IH Data[b0011] -Data pins are set IH D0 D1 IL D2 D3
- IH Data[3] -Data pins are set IH D0 D1 IL D2 D3
- IH Data[0] 'Data pins are set IL D0 D1 IL D2 D3
- IH Data[] 'Data pins are set IL D0 D1 D2 D3

Note the format also supports Octal (if anyone remembers that), e.g., [o134]==[x5c]

The following Logic states pairs are supported:

- IH IL
- OH OL
- MH ML
- IHOL ILOH
- KEEP TOG
- IOX (IOX)

SIG* is not supported by IEEE Std 1641.

## V. Bringing It All Together

To try and put this in context lets consider a digital signal model using these components.

On the example TPS there were two set of digital pins each using different voltage levels. The digital test does an initial digital sequence at 50 ms, waits for a UUT synchronous response and then performs three test patterns with one repeated 20 times.

- Define multiple DigitalStreams combined using a Channel component. Each DigitalStream defines a different set of pins and channels, and different voltage levels (Figure 4.)

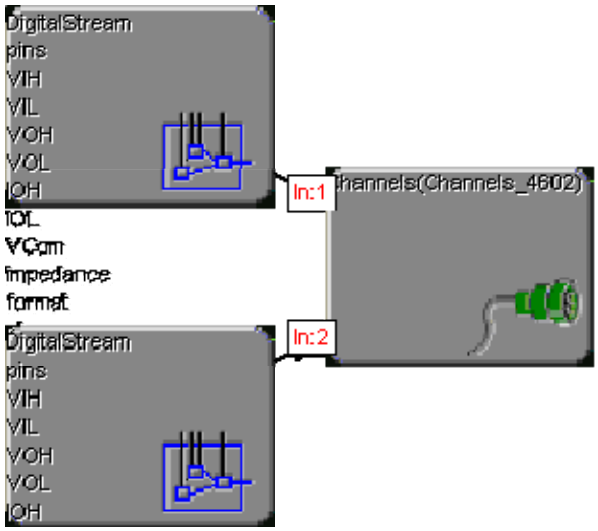


Figure 4. Defining Multiple pin sets

- Define the initial pattern and associated timing set. The initial pattern must have a timing set defined, subsequent tests can either define their own timing sets or inherit the previous timing set (Figure 5.).

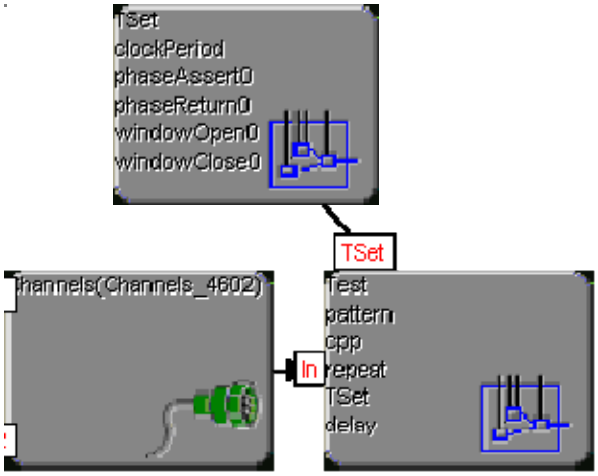


Figure 5. Initial Test

- The UUT Synchronization waits for a specific UUT response, in our case "_CMSTRB = OL;". WaitFor can be set to wait 'indefinitely or keep looking for a maximum number of times using the ***repeat*** attribute (Figure 6).

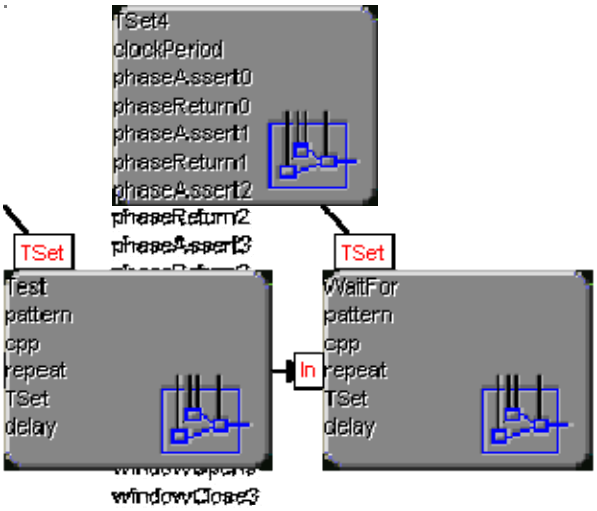


Figure 6. UUT Synchronize

- Finally, the remaining test sequence can be defined. In this case it inherits the TSet used in the **WaitFor**. It needs to be defined in three sections as the middle one is marked to repeat (20 times). A single component can take as many digital patterns as required. However, because the other attributes, e.g., ***cpp***, ***delay***, ***repeat***, etc. are not the same for all patterns, additional components are required (Figure 7).


Figure 7. Remaining Tests

## VI. Something For the Student

This paper contains examples where the output of digital streams are being concatenated (e.g. Figure. 7). Such that each digital component is performed in sequence one after the other.

This concept was probably so intuitive it was just accepted. However, within IEEE Std 1641 signal modeling rules we need to look a little bit closer to understand and convince ourselves this approach is both valid and reliable.

As an example lets consider three tests (A, B, C) that concatenate three patterns. As a single test the pattern would be "IL A B; OL A IH B; IOX *". An equivalent digitalString for a parallel stim and response sequence would be to stimulate "LL, XH, XX" and response "XX, LX, XX". Table II shows the pattern from each test.

TABLE II. Sequencing Digital Stream Components

| Component | IEEE Std 1641 Pinstates | | |
|---|---|---|---|
| | ***Pattern*** | ***Stim*** | ***Resp*** |
| TestA | IL A B | LL | XX |
| TestB | OL A IH B | XH | LX |
| TestC | IOX * | XX | XX |

In order for all three examples to be equivalent our three Tests (A, B, C) must be sequenced in order such that B starts when A completes and C starts when B completes, etc. This is achieved by all digital streams detecting the 'end of signal" Z state at their inputs and using that to trigger their own pattern.

The output of each digital stream component is a multi-channel signal. In effect each digital pin is represented as a channel and while the stream is operational it has one of the values HLX. However, both before and after the test is operational, after it's finished and stopped, the component reverts to the passive Z state. Effectively, the sequencing of each digital stream is controlled by detecting the stopped state (channels' transition to Z) and using it to synchronize or trigger the next digital pattern.

This same technique has general applicability and would be extremely useful for creating arbitrary waveform signal sequences for use in traditional arbitrary waveform generators; this however is outside the scope of this paper

Detecting the end of a digital streams Z state ends up being is a relatively simple model, basically we use the multichannel Sync and Gate properties to trigger an event when the input signal stops. Fig 8. identifies an IEEE 1641 signal for such a task.

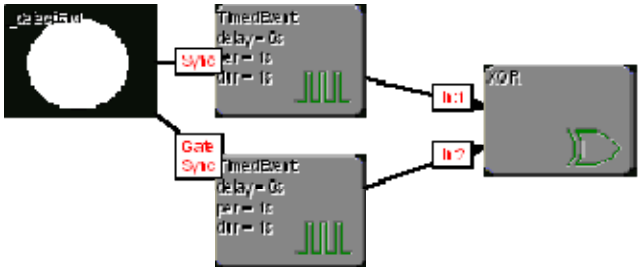


Figure 8. Detecting temporary end of stream

## VII. Conclusion

These digital signals are being used by DSG to define digital test program sets (TPS) in line with the UK MoD ATS Policy.

By defining and building components based around existing features and practices already being used, this approach has allowed significant reuse of knowledge and experience, whilst underpinning the development process with IEEE Std 1641 in line with Open System Architecture.

The M9 digital system has an ability to provide far greater flexibility than that supported by these digital signals components. However, by identifying the right level of abstractions used by the DSG team we were able to simplify the interface used for the majority of use cases. The digital components defined cover about 97% of all digital programming features used by the DSG team to program their M9 digital systems, whilst mapping completely onto the IEEE Std 1641 through the TSF library. Additional digital components could be added as and when use cases are defined.

Because ATML Test Description uses the IEEE Std 1641 to define the signals within a test description by embedding these digital signal models (TSFs) with an ATML test description, ATML can directly use these to define digital signals for use within ATML. This is the exact approach already being performed by the ATML demonstration team [5] across multiple ATEs with different digital platforms.

## REFERENCES


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

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

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

[4] 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.

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

[6] IEEE Std 1445™–1998, IEEE Standard for Digital Test Interchange Format (DTIF)