IMPLEMENTING IEEE 1641 - A DEMONSTRATION OF PORTABILITY

Matt Cornish  
Racal Instruments Group Ltd, UK  
matt.cornish@racalinstrumentsgroup.co.uk

Malcolm Brown  
MOD, UK  
ctstmats1b@wsa.dlo.mod.uk

ABSTRACT

For some years now, the Standards Technical Working Group for Automatic Test (STWGAT) has been developing the Signal & Test Definition (STD) standard under the designation of IEEE 1641 [1]. In the year that 1641 has been formally released, a UK Ministry of Defence (MoD) sponsored study into the uses of the standard has been undertaken; producing a 1641 compliant Test Program Set (TPS) and proving its portability across multiple, unique Automatic Test Equipment (ATE) platforms.

IEEE 1641 may be expressed in many Commercial Off-The-Shelf (COTS) languages, including Visual Basic (VB), eXtendable Mark-up Language (XML) or C++. Two Test Program Sets (TPS) have been produced, one in VB and one in XML (based on Automatic Test Mark-up Language (ATML) [2] – Test Description). Through reference to the 1641 standard, these two test programs are identical and unambiguous in functionality.

Just as tests can be described using 1641, so, too, can test resources [3]. 1641 based resource descriptions have been produced for each of the ATE platforms used in this study.

Resource management and allocation can be achieved automatically based upon a match of test signal requirements and resource signal descriptions.

INTRODUCTION

This paper describes a study and demonstration, sponsored by the MoD, with the aim of producing a TPS conforming to IEEE 1641 STD and showing its portability (through signal based test definition) across several test platforms in order to:

• Provide a means of meeting a test requirement.
• Promote portability across both ATE and software test environment.

And, also, to assess what 1641 could offer in terms of:

• Overcoming existing obsolescence problems.
• Reducing IPR issues.

Equipment Selection

In order to carry out this study and to ensure its validity, the following requirements were identified:

• The UUT should be a real, in service, Line Replaceable Unit (LRU).
• Testing should require a range of analogue and digital tests.
• Testing should not infringe any Intellectual Property Rights (IPR).
• An in-service Application Test Package (ATP) should be available:
  o From which a set of baseline results could be obtained.
  o To minimize time spent designing & building Interface Test Adaptors (ITA).
• Multiple ATE platforms, capable of running the same ATP must be available.

Test Set First Line (Electronics Unit)

After a short review, the TSFL LRU was found to meet these criteria.

TSFL is used to verify operation of High Velocity Missile (HVM):

• Aiming Unit.
• Launcher.
• Guidance
System.
Unfortunately, there were less digital tests than initially apparent, though the unit does require serial bus testing.

ATE

The following ATE platforms were identified as possessing a common set of functionality needed to test the TSFL:

- Factory ATE – currently used for fourth line maintenance of the TSFL.
- Spectrum – a COTS General Purpose ATE (GPATE).
- DIANA – a deployable GPATE for which a TSFL ATP is available.

IMPLEMENTATION

Signal Extraction

By referring to the Product Acceptance Specification (PAS) it was initially simple to derive 1641 signals. This is because a PAS is broadly written in terms of UUT.

E.g. the PAS statement:

‘Monitor SK2-F with respect to 0V at SK2-s and verify that the voltage is +5.0V +/-0.2V. Note that, to avoid power-up transients, this measurement is to be taken greater than 1s after applying power.’

Translates to the 1641 signal definition:

```xml
<TwoWire name="TW1" hi="SK2_F" lo="SK2_s"/>
<TimedEvent name='TE1' delay='1s' repetition='1'/>
<Instantaneous name="Inst1" In="TW1" Gate='TE1' type='Voltage' samples="1" nominal="5.0V" LL="4.8V" UL="5.2V"/>
```

where the mapped elements are emboldened.

Signal Modeling

Figure 1 illustrates how the signal elements from the PAS were modeled graphically, using newWaveX, then simulated and, finally, VB or XML source code automatically generated.

COTS Language Test Definition (.NET VB)

Code 1 shows how test programs have been captured in a COTS language - in this case, VB. The VB test program controls the ATE through an ActiveX object complying with the IDL interface defined in 1641 (in this case, provided by the newWaveX run-time engine).

Note that the PSU has been created by calling the Require method to return individual BSCs.

![Figure 1 – Signal Modelling and Generation of 1641 Test Program Code](image)
' Create EU Power Supply Signal
Set psu = ate.Require("PSU")
psu.name = "PSU1"
psu.ampl = "45.5 errlmt ±0.5V"
psu.current_limit = "550mA errlmt ±30mA"
Set tw = ate.Require("TwoWire")
tw.Name = "TW1"
Set tw.In = psu.Out
tw.lo = "BatteryNeg"
tw.hi = "BatteryPos"
Set tw = ate.Require("TwoWire")
psu.current_limit = "550mA errlmt ±30mA"
psu.ampl = "45.5 errlmt ±0.5V"
psu.name = "PSU1"
Set psu = ate.Require("PSU")

't Create Measurement
Set monitor = ate.Require("<Signal name='T030-0060' Out='Inst1'..."
"<TwoWire name='TW1' hi='SK2_F' lo='SK2_s'/>
"<Instantaneous name='Inst1' In='TW1' Gate='TE1' type='Voltage' samples='1' nominal='5.0V' LL='4.8V' UL='5.2V'/>
<!--[CDATA[<TimedEvent name='TE1' delay='1s' repetition='1'/>]]-->
"</Signal>")
Set monitor = Nothing

psu.Out.Run

Code 1 – newWaveX generated VB Sample

However, it is also possible to Require an entire signal in XML - as in the Measurement in the lower half of the code sample.

XML TestDescription

Code 2 shows a sample of how test programs have been captured in XML. This XML is a version of the ATML standard, currently in development. However, it is to include 1641 as a method of describing signals. It is particularly useful, as many tools exist to help process, manipulate and extract information from XML files.

Code 2 – XML Test Description Sample

TSFs

Code 3 shows a sample of the TSF library that has been created. Further understanding of the Test Signal Framework can be obtained from the standard itself. However, it is worth noting that this particular TSF references out to an existing standard – RS422. RS422 is already fully defined and standardized, so there would
be no point in redefining it in terms of BSCs.

**Resource Description**

Resources (instruments) on the Factory & Spectrum ATEs have been described in terms of the signals (i.e. BSCs) that they support. Resources can then be matched onto test signal definitions by the resource manager on the ATE, to automatically allocate suitable test resources. When a resource is allocated, a connection to the relevant driver is created. Code 4 is a sample of the Resource description that was created for the Factory ATE.

At the time of writing, 1641 is being considered for inclusion in both ATML [2] and Resource Adaptor Interface (RAI) [4] to provide capability description in terms of signals.

**Signal Extraction from Existing Source Code**

As part of this study, the implications of 1641 for existing ATPs were to be considered. Two sources of test program source code were available; these being for the Factory ATE and the DIANA.

In older ATPs, source code is often the only means of describing the signals used within the test program.
source of test information available. Where the test program is written in an older, proprietary or obsolescent language, the code, style and structure may be unfamiliar. The non-standardized development of many ATPs means that test programs often contain ATE specific information. This irregular structure and ATE specifics mean that automatic extraction of signals is not viable. And, without automated extraction, the test program is being virtually re-written, which, in the case of larger ATPs, is very laborious and, thus, prone to error.

Use of Source Code and PAS for existing systems

As these tests had already been implemented on two of the ATEs in the study, it was also necessary to refer to the source code for that ATE to resolve amendments that had not been put back into the PAS. E.g. where a current measurement had been made using a current shunt, the measurement had become a voltage measurement with a formula.

Figure 2 – Sources of Test Information

For the purposes of this study, it was found most useful to use the PAS to derive 1641 signals representing each of the individual stimuli and measurements that make up the tests.

Source code also provides run-time information for the hardware on which the test program is running, that must be transferred to a driver, under the control of the run-time system. E.g. ITA functions controlled by the test program, where the test program should be independent of the hardware, even the ITA.

Figure 2 illustrates the production of TSF libraries & test signals from which XML test descriptions can be obtained and the COTS test language program produced.

1641 Run-time System

Factory ATE

Figure 3 illustrates the implementation of the 1641 test program on the Factory ATE:

1. The COTS test program Requires a signal, in terms of BSCs and TSFs. This starts a sequence of actions in the run-time system:

2. The signal definition is compared with the resource description that the resource manager holds for the ATE.

3. As resources are matched to signals, a driver to control that resource is loaded.

4. UUT pins referred to by the signal definitions’ connection BSCs are matched to the pin map and a driver is loaded to control the switch configuration on the ATE.

5. BSCs that can be implemented on the controller, such as a TimedEvent, are integrated into the implementation from a software library.

6. Where signal definitions contain TSFs, these are referred to for details on which BSCs make up the signal.

7. Finally, results are logged in ATML TestResults format, which can later be re-formatted to match the result format of the base-line test program. As is illustrated, it is not the test program, but the run-time system that is logging the test results. This means that the test program does not need to carry
environment-specific logging instructions. In fact, it is the measurement BSC, itself, that makes the call to the logging application; so any test incorporating a measurement will be logged.

**Spectrum ATE**

Portability of the COTS Language test program is shown in Figure 4. This architecture is identical to that for the Factory ATE (Figure 3). The differences on the ATE are in the **instruments**, **switches** and **pin map** through the ITA.

So, by utilizing **Drivers** for the Spectrum ATE and producing a new **Pin Map & Resource Description**, the **Resource Manager** is able to generate the same **signals**, but on a different ATE.

What is critical for test program portability is that the 1641 signal capabilities of the host ATE are accurately described. Furthermore, experiences of obsolescence management make it clear that though accurate characterization of instrument capabilities is not easy, it is **essential**.

**Alternate Test Executive**

Though the test program was initially executed through Genie, to show portability across system software, the test program was also executed using the Spectrum’s native test executive. As the test program is carried in a COTS language that is supported by Test Studio, it was possible to transfer the test program to this alternate test executive with no changes to the test program.

**Such portability has benefits in maintaining a familiar test environment for the operator.**

It is the inclusion of an IDL description in 1641 that enables the run-time system to be embedded in these and other different test executives.

**DIANA**

In the case of a SABRE test program, all the instrument allocation and switching information is contained within the test program, itself. So, to be able to allocate resources and switch paths for 1641 signals, ‘snippets’ of code that do this are extracted from a SABRE test program. Now, when a 1641 signal definition is received, the ‘snippets’ of code that go together to create that signal on the ATE must be assembled and executed.

An XML style sheet was created to assemble SABRE code ‘snippets’ to form a new SABRE test program from a 1641 signal-based test program. SABRE test programs are compiled to REBATE, off-line, before being transferred to the DIANA for execution.

In the dashed box of Figure 5 is a 1641 compliant system for translating 1641 based test descriptions into SABRE code. This is the equivalent of the run-time environment that has been shown for the Factory and Spectrum ATEs. However, this is an off-line activity; i.e. not occurring at run-time or on the same machine.
Finally, the REBATE interpreter on the DIANA executes this code to test the UUT and prints out the results.

**Comparison of Off-Line and Run-Time System Techniques**

The two approaches covered in this paper represent significantly different applications for 1641: New systems utilizing new technology and existing systems utilizing older technology, typical of many ATE in service today. A summary of benefits ☑ and detriments ❌ in the application of 1641 to these situations is provided below.

**Run Time System**

☑ Operates on current languages & platforms
☑ Supports multiple COTS languages
☑ Supports multiple test executives
☑ Separate Resource capability descriptions
☑ Requires Resource Allocation
☑ Requires ITA and UUT connectivity

**Off-line Translation**

☑ Can use existing (proprietary) languages
☑ Do not translate at run-time
☑ Translator has all the ATE knowledge

![Figure 6 – Architectural Overview](image)

Figure 6 shows an architectural overview of the elements that are included in this study. It shows the routes taken to get from an initial test specification, in terms of the PAS and existing test programs, to four sets of comparable test results. Three ATEs are identified (1, 2 & 3), with a fourth (4) being the same ATE as 3, but utilizing an alternate test executive.

![Figure 7 – Sample Test Results Comparison](image)

Figure 7 shows a sample of the test results produced by each of the ATEs. It can be seen that these are presented in a very similar format. To make a fair comparison of the different

<table>
<thead>
<tr>
<th>Baseline DIANA Test Results</th>
<th>DIANA (XML via style sheet)</th>
<th>Factory ATE (COTS Language)</th>
<th>Spectrum (COTS Language)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST PASS LOW SOURCE</td>
<td>TEST PASS STD DERIVED TESTS</td>
<td>TEST PASS STD DERIVED TESTS</td>
<td>TEST PASS LOW SOURCE</td>
</tr>
<tr>
<td>UNITS = VOLTS</td>
<td>UNITS = VOLTS</td>
<td>UNITS = VOLTS</td>
<td>UNITS = VOLTS</td>
</tr>
<tr>
<td>3.2.2.2.d</td>
<td>3.2.2.2.d</td>
<td>3.2.2.2.d</td>
<td>3.2.2.2.d</td>
</tr>
</tbody>
</table>

| TEST PASS HIGH SOURCE       | TEST PASS STD DERIVED TESTS | TEST PASS STD DERIVED TESTS | TEST PASS HIGH SOURCE     |
| MV = 5.061                 | MV = 5.061                 | MV = 5.061                 | MV = 5.061                 |
| LL = 4.800                 | LL = 4.800                 | LL = 4.800                 | LL = 4.800                 |
| UNITS = VOLTS              | UNITS = VOLTS              | UNITS = VOLTS              | UNITS = VOLTS              |
| 3.2.2.1.b                   | 3.2.2.1.b                   | 3.2.2.1.b                   | 3.2.2.1.b                   |

| TEST PASS POST STATUS LED   | TEST PASS MEASURED VOLTAGE  | TEST PASS POST             | TEST PASS POST             |
| TEST T040-0120              | TEST T040-0120              | TEST T040-0120              | TEST T040-0120              |
| UNITS = VOLTS              | UNITS = VOLTS              | UNITS = VOLTS              | UNITS = VOLTS              |
| 3.2.2.3.a                   | 3.2.2.3.a                   | 3.2.2.3.a                   | 3.2.2.3.a                   |

☑ Boiler Plate code per ATP
☑ ITA specific translators or translate to the test head
methods used, a similar amount of effort was applied to each.

The results themselves show success for the completed tests.

The last test (POST) shows a partial success: The successful elements are noted, as are the elements that caused the failure. Resolution of these issues would simply involve spending more time incorporating the missing elements into the XML style sheet.

Results have been logged in all three environments in a manner which is independent of implementation and have retained the same format through use of ATML TestResults and XML style sheets.

By incorporating the logging mechanism within the measurement BSCs implemented in the run-time system, no logging instructions are required in the test program. This promotes a more structured logging behavior, which can otherwise lean toward the logging of informal data. These structured results have proven to be easily incorporated into ATML TestResults; which have, in turn, provided the facility to maintain test results in a common, definable format, across multiple platforms.

**DIGITAL TESTS**

It is acknowledged that though a digital bus was part of this study, in the form of the RS422 bus, no actual digital BSCs formed part of any tests. Below is a summary and example of the use of the 1641 Digital BSCs and TSFs:

- **Digital**
  - Static Digital
  - Parallel Digital (Digital Bursts)
    - Stimulus
    - Response
  - Measurement Windows
  - DTIF (TSFs)
- **Digital busses**
  - RS422 (TSFs)
- **Components**
  - Parallel Digital
  - Serial Digital
  - Digital Bus
- **Formulae**
  - Translations
  - Synchronized response to events
  - Timed responses

**Digital Example:**

*Keep the UUT in test mode by sending “LHLHLHLH” to PL1-1 every 50ms*

DigitalBus pins="PL1-1"
SerialDigital sync='clk' data="LHLHLHLH"
Clock name='clk' clockRate="50ms"

**CONCLUSIONS**

IEEE 1641 has successfully enabled the development of an ATP test program that has been executed on three different ATEs and across three different operating environments. IEEE 1641 has additionally enabled the resource descriptions of ATEs and configuration of a run-time system to support the execution of the test program.

As well as .NET VB, XML has also been used to successfully capture the test program through ATML Test Description. This shows synergy with other like standards and improves the validation and scope of 1641.

In addition to being implemented on a 1641 compliant ATE, the XML test program has been used to reach a non-COTS test language through the use of style sheets. Though this approach is not recommended, it emphasizes the good structure of the standard, as it was possible to automate the interpretation of the test program in this way.

IEEE 1641 has shown itself to be a new standard suitable for new TPSs. The benefits to existing TPSs are dependent on the quality and availability of support documentation and IPR; though this is not unique to 1641. However, IEEE 1641 compliant TPSs ensure quality of information at a non-proprietary level, thereby radically improving obsolescence management in the future. The availability of non-proprietary information for new TPSs also offers a way to overcome IPR issues in future contracts.

The process of capturing the signal requirements for the tests from the PAS has highlighted that it is important to accurately define what signals must be applied or measured, in terms of the UUT. Since the standard accurately defines signals (rather than simply being another executable programming
language) it is ideally suitable to form part of such test specifications.

From the measure of effort that has been expended on implementing the 1641 on these three systems, it is clear that the profile of 1641 test systems must be quite different from many existing test platforms. There is a greater emphasis on accurate description of instrumentation to support resource allocation and thus portability for tests.

Given accurately defined resources, 1641 has definite benefits for matching tests to resources. This is further enhanced through the use of the XML format within 1641, which provides ready parsing and extraction of the values to be matched.

**NEXT STEP**

In order to capitalize on the findings and developments of this study, it is suggested that a further study is initiated to report on the provision of comparative costs between the recommended STD approach and contractor preferred ATP development. This study should:

- Use 1641 to shadow the development of a new or converted ATP including digital signals
- Allow the test program to be developed by the contractor using their preferred method.
- Produce a 1641 signal definition and test program supporting the ATP that can be run on the target ATE.
- Compare both performance and costs of the two approaches.
- Provide a fuller Resource Capability of the target ATE, using IEEE 1641.

**REFERENCES**