Implementing IEEE 1641 – Using a complete system

Ashley Hulme and Keri Nash
EADS Test Engineering Services (UK) Ltd
29-31 Cobham Road, Wimborne,
Dorset, BH21 7PF, UK
Phone: +44 1202 872800, Fax +44 1202 870810
Email: *ashley.hulme@eads-ts.com, keri.nash@eads-ts.com*

Abstract *– This year, new products have been introduced onto a Royal Air Force (RAF) test system, which provide an integrated IEEE 1641 development and run-time system. The program involved the integration of Commercial of the Shelf (COTS) products to provide a facility to enable the creation of a TPS using 1641 signal definitions, through to the running of the test program using established instruments and driver software. This paper discusses the process involved in achieving the objectives from the point of view of the TPS developer and integrator, and will cover the complete process, from requirements capture, via TPS integration, to program operation and results verification. The paper will also outline the lessons learned and the resulting feedback to the COTS product developers and 1641 working groups.*

Keywords *– ATE, ATS, signal modeling, test definitions, test requirements, test signals, UUT.*

I.INTRODUCTION

A. Background

IEEE Std 1641™–2004 [1] has been established now for four years and several demonstrations and trials have been completed to test its utility. These trials [2][3][4] have been conducted on behalf of the UK Ministry of Defence (MoD). The MoD has supported the development of the 1641 standard and has adopted it as part of its policy for new test requirements. The results and findings of these trials have been reported back to the IEEE working groups via the UK Standards Technical Working Group for Automatic Test (STWGAT). These activities have made a major contribution to the improvements to the 1641 standard, which are currently being processed under the IEEE SCC20 project P1641.

This project was the natural evolution of this process, and would provide a prototype of a complete integrated development and run-time environment for 1641 test programs.

B. Project Overview

The project used COTS products to provide the RAF with a toolset to create, develop, and execute 1641 test programs on the RAF's General Purpose Automated Test Equipments (GPATE) at the Defence Support Group's (DSG) repair facility at Sealand in North Wales. The contract was managed by EADS Test Engineering Services (UK) Ltd on behalf of Defence Equipment & Support (DE&S). DE&S equips and supports the UK's armed forces for current and future operations. It acquires and supports through-life, including disposal, equipment and services ranging from ships, aircraft, vehicles and weapons, to electronic systems and information systems.

The requirement was for an integrated development and runtime environment which could not only run on the GPATEs but would also provide an offline development and simulation environment that would run on a stand-alone personal computer. In addition, the project would provide a complete working example of a TPS, which would be used as part of the acceptance process.

A high degree of parallel development was necessary due to time pressures imposed by the MoD's purchasing and finance management cycle. This required good communications and a high degree of cooperation from all the parties involved, the project teams were split between three companies over three sites on two continents. Video and telephone conferencing facilities were frequently used to maintain the pace of development.

II. PRODUCT AND TEST SUBJECT SELECTION

A. Product Selection

Two complimentary products were selected for the project, newWaveX™–SD and SigBase®. Prior to this DE&S had commissioned a survey to determine what 1641 related products were available for system integrators. Although several suppliers could support 1641 on their own test equipment, there were few stand-alone products that were commercially available. The selected products work together and also satisfy the requirements to run both on the GPATEs and on standalone PCs.

newWaveX–SD provides all the necessary facilities for developing 1641 compliant signals required to test a Unit Under Test (UUT). SigBase provides the test program structure development and runtime environments.

B. Test Subject Selection

The intention was to select a test subject that used most of the facilities provided on the target GPATEs. Due to

constraints imposed project timescales, cost implications and IPR considerations, the list of suitable subjects was considerably reduced. The unit eventually selected was one that had already been used for a previous 1641 related project. This had the following advantages:

- an existing Interface Test Adapter (ITA), reducing both cost and timescales
- the Production Acceptance Specification (PAS) was owned by one of the participating companies, which not only reduced cost but meant that it may be possible to get any technical queries answered quickly
- the signal requirements were a reasonable match with capabilities of the GPATE instruments.

The test subject was a small unit which required a series of analogue signals as inputs and outputs and required an EIA/TIA 422 communication bus. Several of the tests required time related signals and measurements. There was a display which would require operator interaction. Overall, the test subject would require a good cross section of 1641 signals.

III. PROGRAM DEVELOPMENT

A. Source Material

The 1641 program was developed from the original PAS with no reference to other implementations, either for the DSG GPATE or any other test equipment.

B. Reusable Signals

Examination of the PAS revealed that only a limited set of different signals was required throughout the whole test sequence. This requirement aligned with the 1641 philosophy of defining reusable signals or Test Signals Framework (TSF) models. It also helped with the project constraints in that the full development facility was not available at the outset of the project and work on TSFs could start in parallel with the other activities.

The TSFs were created using newWaveX, which can be used stand-alone or as part of an integrated facility. The initial group of TSFs were created using newWaveX in stand-alone mode.

Table 1. List of TSFs created for Project

TSF Name	Function
ResistanceCheck	Measurement
dcVoltageCheck	Measurement
BatterySupply	Source
RS422_Send	Source
RS422_Receive	Measurement
PrecisionDCSource	Source
TestLoad	Source
ShortingLink	Source
Source380Hz	Source
CollimatorSignal	Source
SupplyCurrent	Source/Measurement

C. Project TSFs

As the project was for a single test subject the TSFs created were only those required for that test subject. A TSF may have fixed attributes, or variable attributes with or without default values. This is very useful from a programmer's point of view as it reduces the amount of data that has to be entered.

The list of TSFs created for the project is shown in Table 1. Some of these are described in detail to illustrate the various features of a TSF signal.

Source380Hz TSF

The Source380Hz TSF is a straightforward source TSF which comprises a Sinusoid and TwoWire Basic Signal Components (BSCs). This is shown in figure 1.

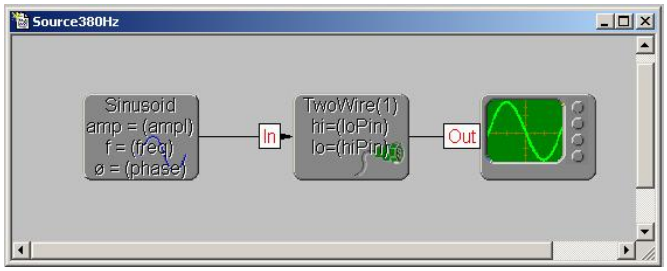

Figure 1. Source380Hz TSF

This TSF had all its attributes defined as variables with default values provided. This allowed the TSF to be connected to any of the UUT pins and to be given any value of voltage and frequency. As the TSF was intended for use at a specific frequency (380 Hz ±2 Hz) and at a predefined voltage (19.7 V ±0.1 V), these values were specified as the default values. Similarly the connection pins were defaulted to the UUT pins that it would normally be connected to. When the TSF was being used in its default configuration, the programmer only has to reference the TSF and does not have to enter all the attribute values.

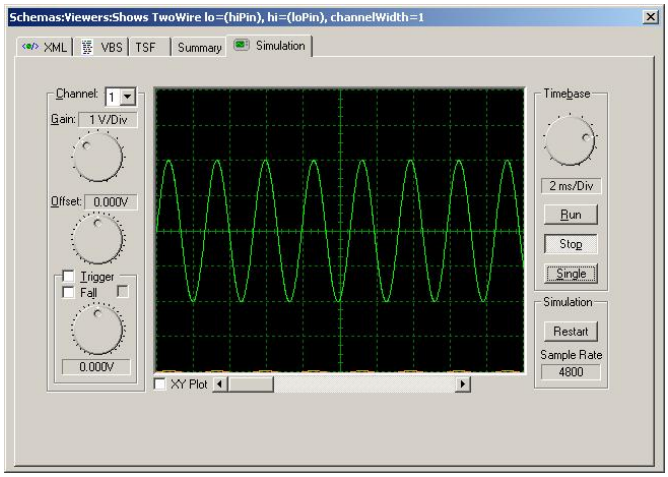

Figure 2. Source380Hz Simulation

The signal was also simulated to verify that the correct output is generated. This is illustrated in figure 2.

In this instance the signal is obvious and does not require a great deal of verification. In the case of more complicated signals this feature is very valuable, the user can use the simulation to confirm that the signal provided by the test equipment is correct.

ResistanceCheck

The ResistanceCheck TSF is a straightforward example of a measurement TSF.

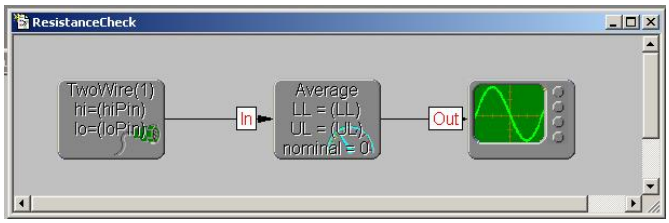

Figure 3. ResistanceCheck TSF

This TSF also used variable values for its attributes so that it could be used on any UUT pins and for any value of resistance. No defaults were provided so that the programmer is required to enter the pin numbers and measurement limits each time the TSF was used.

Measurement TSFs cannot be simulated as a stand-alone element as there is nothing to measure. If necessary, to check the validity of the TSF during development, a source TSF or BSC may be connected temporarily to the input to enable a simulation to occur.

SupplyCurrent

The SupplyCurrent TSF provides both a source and a measurement function in a single TSF. It also illustrates a timing element as the current measurement occurs a specified time after the supply voltage is applied. This is illustrated in figure 4.

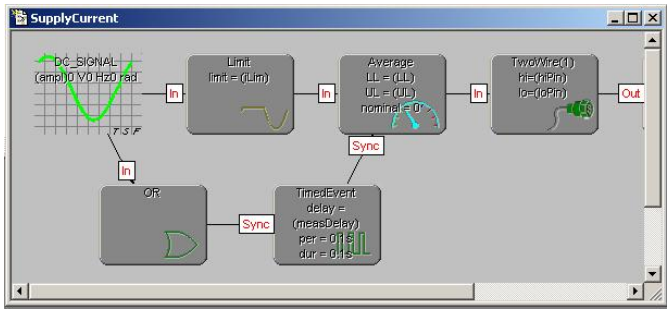

Figure 4. SupplyCurrent TSF

The TSF operates as follows. When the TSF is invoked, the DC_SIGNAL provides the supply voltage required. The Limit BSC limits the current to a safe value in case of a UUT fault. The TimedEvent BSC is triggered (via the OR BSC) by the DC_SIGNAL starting (the OR BSC is only there to provide a link between the signal Out from the DC_SIGNAL and the Sync Event input to the TimedEvent). The DC_SIGNAL voltage is applied to the UUT via the connection BSC, which specifies the connection pins. When the TimedEvent BSC time is completed, a Sync Event is applied to the Average BSC. The Average BSC is set to measure the current (Type = Current), so when the Sync occurs the current is measured.

The measured value is compared with the specified limits (the values set for UL and LL) and the GO/NOGO flags set accordingly.

Again, it is not possible to accurately simulate the measurement but it is possible to show the current measurement being triggered after the specified delay (figure 5).

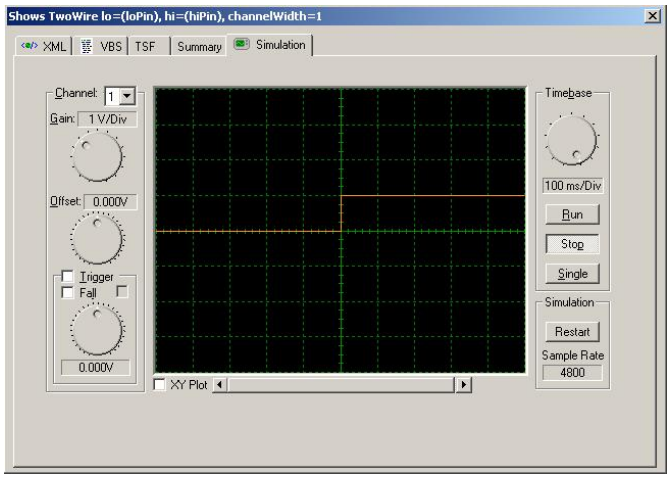

Figure 5. Current Measurement Simulation

RS422_Send

The RS422_Send TSF makes extensive use of the defaults as it not necessary to specify the protocol settings at each use. The programmer only has to provide the message text as the protocol information is all provided in the defaults. The RS422 TSFs are not built up from low-level BSCs as it is specified in detail in another standard. Also, for the same reason it is not simulated. Figure 6 shows the RS422_Send TSF in XML format.

D. Level of TSF Usability

These TSFs were designed for this UUT only, with no consideration for other similar or related test subjects. If the TSFs had been for a wider range of test subjects, more thought would have to be given to which values are fixed, variable with defaults, or variable with required entry of values. Figure 7 shows the Collimator signal with its fixed values. If this was being designed for more general use, the fixed values would be replaced by variables, maybe with defaults.

```
<!-- generated with newWaveX v3.0.8
   (http://www.newWaveX.com) -->
<tsf:TSF name='RS422_Send'
  uuid='{9DF2F399-2EFA-475A-843E-A8915C5F0987}'
  xmlns:tsf='STDTSF' xmlns='STDBSC'>
 <tsf:interface>
  <xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    elementFormDefault="qualified">
   <xs:element name='RS422_Send'>
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="SignalFunctionType">
       <xs:attribute name="messageText" type="string" >
       </xs:attribute>
       <xs:attribute name="baudRate" type="int"
        default="19200" >
       </xs:attribute>
       <xs:attribute name="wordLength" type="int"
        default="8" >
       </xs:attribute>
       <xs:attribute name="parity" type="string"
        default="Even" >
       </xs:attribute>
       <xs:attribute name="startBits" type="int" default="1">
       </xs:attribute>
       <xs:attribute name="stopBits" type="int" default="1">
       </xs:attribute>
       <xs:attribute name="handshake" type="string"
        default="None" >
       </xs:attribute>
       <xs:attribute name="loPin" type="string"
        default="SK1-L" >
       </xs:attribute>
       <xs:attribute name="hiPin" type="string"
        default="SK1-t" >
       </xs:attribute>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
  </xs:schema>
 </tsf:interface>
 <tsf:model>
  <Signal name='RS422_Send' Out='RS422SendPins'
    xmlns='STDBSC'>
   <RS422_Send name='RS422_Send8'
    messageText='messageText' baudRate='baudRate'
    wordLength='wordLength' parity='parity'
    stopBits='stopBits' handshake='handshake' />
   <TwoWire name='RS422SendPins' lo='loPin' hi='hiPin'
     channelWidth='1' In='RS422_Send8 '/>
  </Signal>
 </tsf:model>
</tsf:TSF>
```

Figure 6. RS422_Send TSF (XML format)

An alternative view for creating the project TSFs would be to describe the GPATE facilities in terms of TSFs. This approach was not used for this project as the TSFs could not have been created until have been create until the full details of the system instruments were known. Using TSFs based on the Test Requirement not only allows the TSF design to commence before the details of the test equipment are known, but are also more portable.

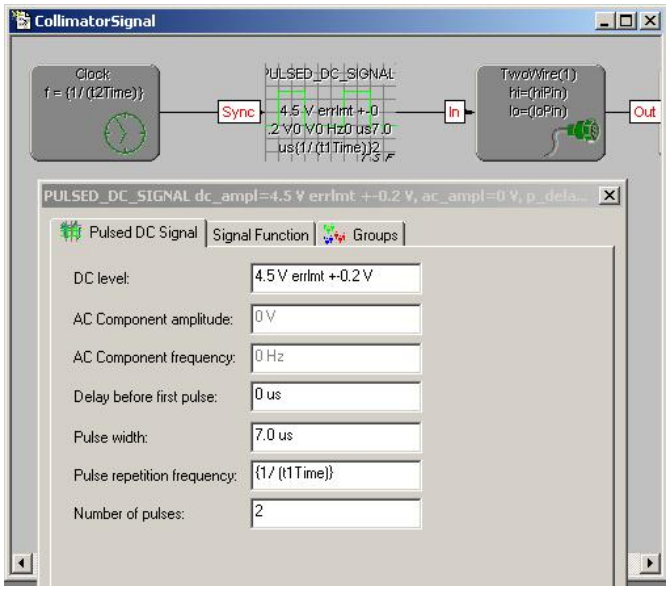

Figure 7. Collimator Signal Showing fixed values

E. Test Strategy.

SigBase may be used to define the test strategy before the detail of the individual tests is known. The test strategy was entered using the test order specified in the PAS. The elements of the test strategy, known as Diagnostic Procedures (DPs) in SigBase, are entered as a sequence in flow chart format. This top level test strategy represents the complete end-to-end test program which is executed to determine the overall PASS/FAIL status of the UUT.

F. Test Development

Each of the DPs is then developed separately in flow chart format. A DP may be quite simple, just simple application and measurement of signals, or quite complex, with multiple sequences of signals and measurements with decisions determining the route through the program flow.

Each flow chart element represents the application/ modification/removal of a signal, a measurement, a decision or a message, etc. The flow chart symbols are provided in a palette (see figure 8) and the appropriate symbol is dragged into position and then linked in the desired sequence.

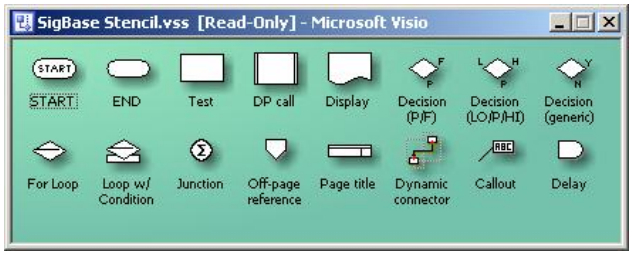

Figure 8. Palette showing available DP elements

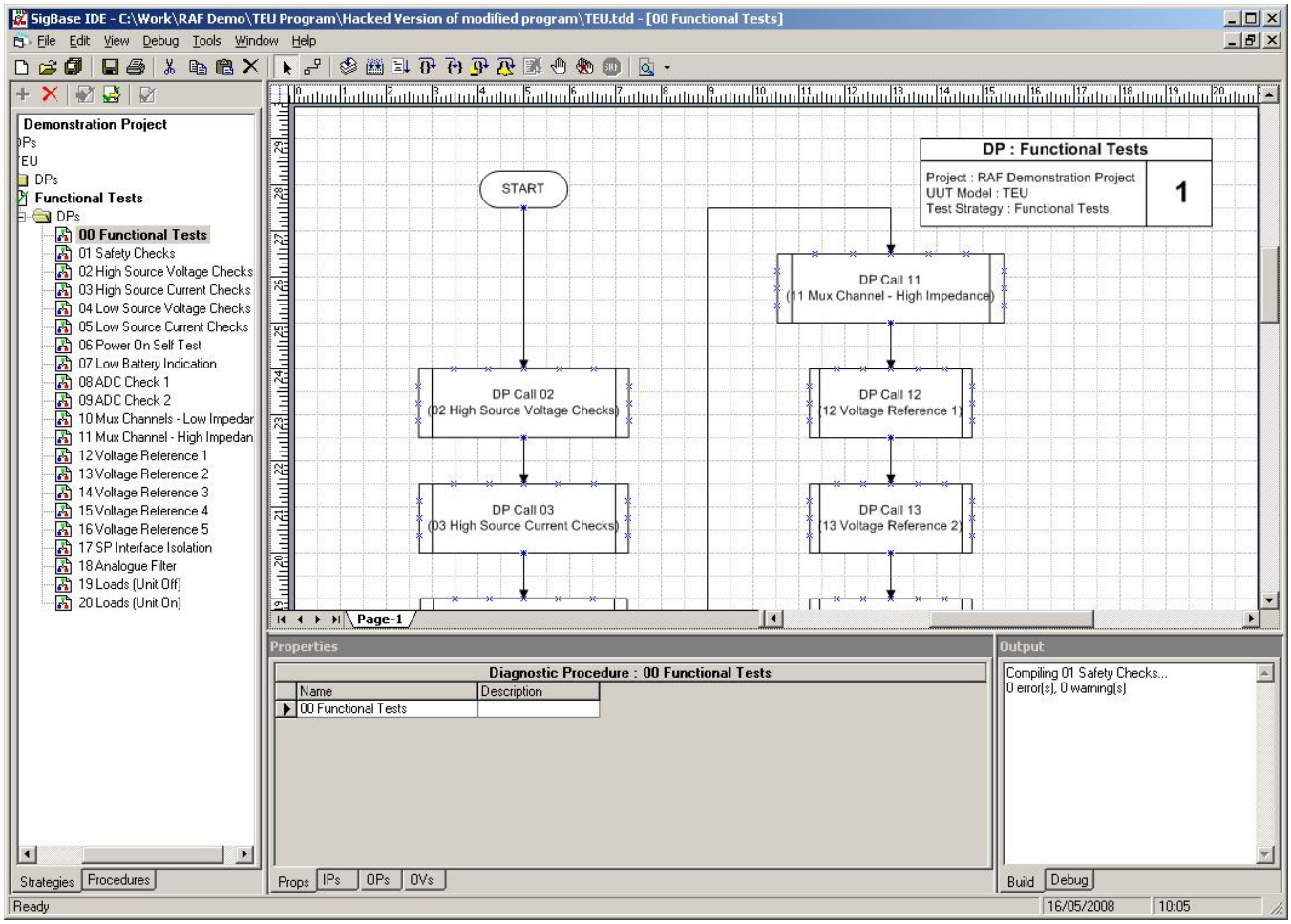

Figure 9. Top-level DP showing sequence of tests

Figure 9 shows the top-level DP, named Functional Tests, which calls all the lower level DPs in the desired order. The order of execution may be changed simply by reordering the elements of the top-level DP.

Each of the Test elements in the flow chart relate to a signal in the Procedures section of SigBase. Therefore, although a test sequence can be constructed in a DP without any signals being defined, the Test elements cannot be completed until an appropriate signal exists for each one.

Figure 10 illustrates a section of a DP in which three measurements are made and the values checked to determine whether they are within specification. If any of the tests fail a message is output to the operator. The stimulus signals would have been applied before the first visible Test element. The Display element is highlighted in the figure, and the contents of the properties pane at the bottom refers to that element.

The treeview pane shows all the DPs in the project. The top-level DP (Functional Tests) is in bold and the current DP is highlighted (High Source Voltage Checks).

G. Test Signals

Selecting the Procedures tab in SigBase opens a work area in which signals are created.

These signals provide the link between the test sequences defined in the strategy section and the TSFs. They are referenced by the Test elements in a DP flow chart and in their turn reference TSFs which provide the signal definition. They also provide any additional attribute values that may be required.

When creating a new signal, a TSF is selected from a palette and the required attribute values are set. Each different signal is given a unique name so that it can be referenced by a DP Test element.

A library of existing TSFs may be loaded and then appear in the TSF palette. If a suitable TSF is not available on the palette, newWaveX may be invoked directly from SigBase to enable the user to create a new TSF. Once a new TSF has been created and added to the library it is available for use in any other signal. The list of signals is complete when there is a signal to satisfy every flow chart Test element in the test strategy DPs.

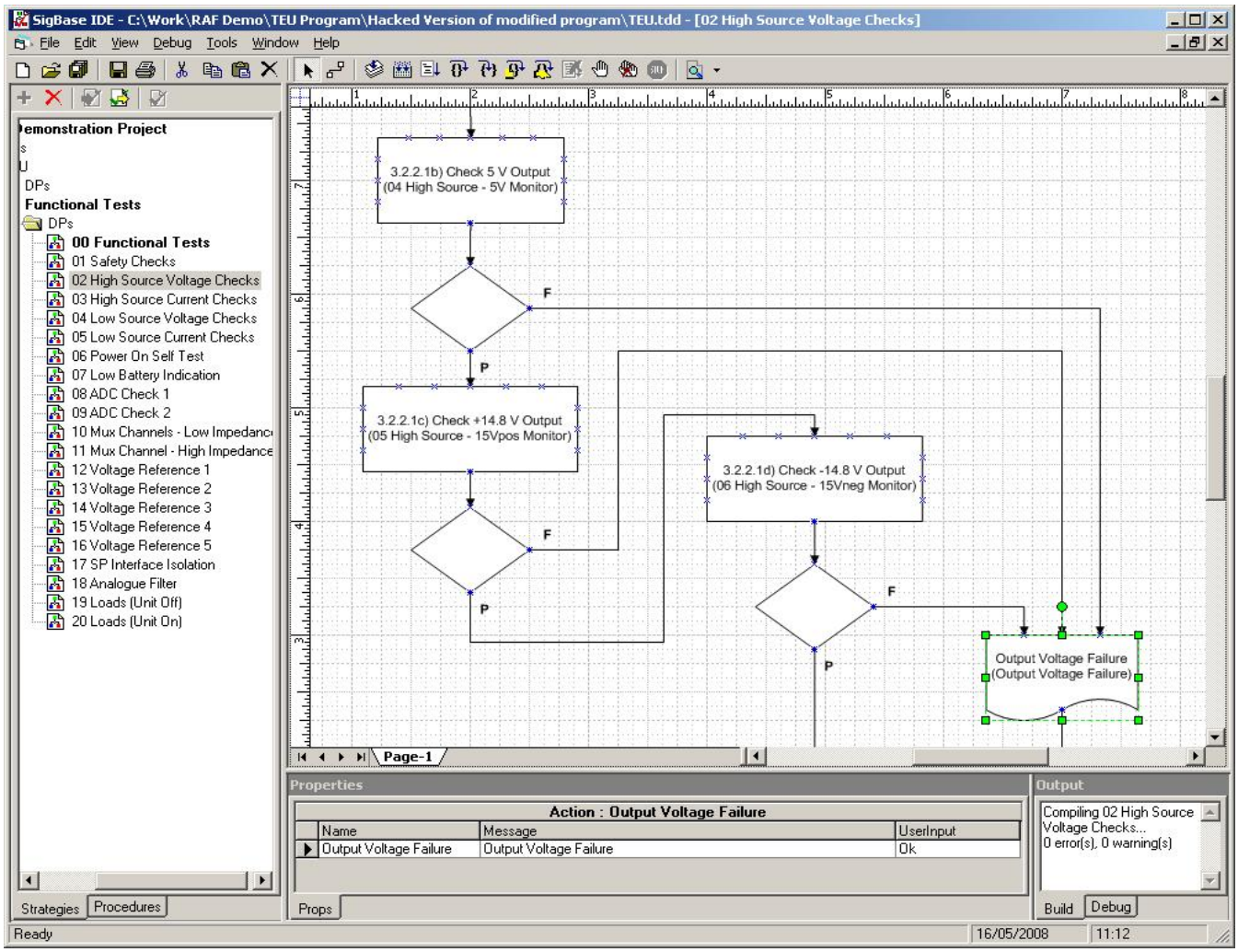

Figure 10. DP showing detail of program flow

H. Compiling and Building

When the Test Strategy is complete and all the Test elements are satisfied, each DP may be compiled and built. This process verifies that all references are complete. If there are any errors or warnings, these are displayed in the Output pane (see lower right corner of Figure 10). The flow path was then simulated with manual entry for PASS and FAIL decisions. This allowed all the program paths to be verified offline.

IV. SYSTEM INTEGRATION

DSG (at Sealand) made a GPATE PC system available so that the SigBase and other relevant software could be installed. Part of the configuration included a removable hard disk drive, which simplified the integration process as it was then a simple procedure to convert the GPATEs between 1641 operation and their normal operation mode. Over the integration period, three different GPATEs were used (simplified by the single hard disk drive) and this ensured that the solution was not unintentionally dependant on any unique features of a single system.

A. Test Station Description

The description of the test station was completed using the standard, well established TYX PAWS® methods. This included files to describe the following:

- Instrument capabilities (including the system switching)
- Interconnections between the instruments (including switch) and the interface panel
- The ITA connections.

This process is in widespread use both in commercial and military test systems and a detailed description is outside of the scope of this paper.

B. Instrument Drivers

Wherever possible, the standard drivers provided with the standard GPATEs were used. Wrappers were created to link

the 1641 data with these standard drivers. The 1641 signals were provided in XML format and contained the attribute values required by the drivers. This can be seen in the example of XML data below.

```
<?xmlversion="1.0" encoding+"UTF-8"?>
<!—generated with newWaveX v3.0.4
     (http://newWaveX.com) -->
<Signal xmlns='STDBSC' xmlns:tsf716=' STDTSFLib'
  name='' Out='TwoWire46'>
  <RS422 name='Signal131'
   messageText='TN0100,20,0&#13;&#10;'
   baudRate='19200' wordLength='8'
   startBits='1' stopBits='1'
   parity='Even' handshake='None'/>
  <TwoWire name='TwoWire46' lo='SK1-L' hi='SK1-t'
   channelWidth='1' In='Signal131'/>
</Signal>
```

The 1641 attributes in the XML were then mapped directly to the driver attributes for execution at runtime.

C. Runtime System

The standard PAWS Run Time System was utilized as the runtime executive. New features were employed to provide dynamic resource allocation and dynamic switch path mapping.

With this runtime environment the developed 1641 test program produced an end-to-end execution time consummate with previous test program variants for the test subject.

V. TEST PROGRAM INTEGRATION

When the first test program for any newly integrated system is commissioned, some of the issues encountered may be system issues rather than program integration issues. However, as the PAWS elements of the system were well proven, there no significant system related issues discovered during this process.

To state that there were no issues encountered during program commissioning would be incorrect, but most of the problems were due to a lack of experience with the specific instrument set used, or a misunderstanding of the detail of the PAS. Typical of the issues that had to be resolved were:

- The PAS left some actions unspecified, such as a requirement to power the UUT off and on between certain tests.
- Using an existing ITA led to a less than ideal instrument selection in one case.
- One signal could not be realized with the instrument set available. His test was eventually omitted.
- Parallel development led to compromises being made in the flow chart (test sequence) design and with output listing.

None of these issues were particularly related to the use of 1641, but were typical of any new development with similar timescales pressures and budget constraints. These could all have been fixed with a little more time and effort, but if the ITA had been updated to allow a different instrument selection and the test flow modified, it would not have added any benefit to the implementation of 1641 programs or signals.

VI. FUTURE DEVELOPMENTS

A. Products

Since the completion of this specific project, the products have continued to be developed and improved, and at the time of writing (July 2008) a further revision of the product suite has been released. The original project was for a test program using analog signals only. Plans are in place to enhance the products further to include a full digital capability. This will be the subject of a further planned demonstration project.

B. 1641 Standard

This project did not highlight any significant issues with the current 1641 standard. The development of a new revision of IEEE Std 1641 is well underway by the SCC20 TAD subcommittee and this effort will continue. This project (P1641) includes further work on the specification of digital signals and a host of small issues which have been collected since work finished on the current 2004 version.

VII. CONCLUSIONS

This project illustrated that it is possible to produce an integrated environment using the 1641 standard to define all the signals. The product suite is designed to be available as a complete 1641 solution in its own right, with the ability to produce 1641 IDL output and to interface with other ATML compliant products. It also has the advantage that it can be integrated with existing PAWS ATLAS systems to provide a dual program environment capability.

From the IEEE Standards point of view, it has shown (once again) that the 1641 standard is viable in its original version and may be used successfully to specify test programs in a test equipment independent manner. This reinforces the work and conclusions from other implementations [2][3][4] and demonstrations of the use of IEEE Std 1641–2004.

VIII. REFERENCES

[1] IEEE Std. 1641™–2004, IEEE Standard for Signal and Test Definition.

[2] Brown, M., M. Cornish, 2005, Implementing IEEE 1641 – a demonstration of portability, *IEEE AUTOTESTCON 2005 Proceedings*, pp 144-152

[3] Brown, M., M. Cornish, D Poole, Implementing IEEE 1641 – RF Stimulus and Measurement, *IEEE AUTOTESTCON 2006 Proceedings*.

[4] Brown, M., and J. M. González Pascual, 2007, A Case Study: Developing a Complete Test Program using IEEE 1641, *IEEE AUTOTESTCON 2007 Proceedings*, pp 718-727