Why did we add LabVIEW applications to our ATLAS TPSs?

Larry V. Kirkland
WesTest Engineering
810 Shepard Lane
Farmington, Utah 84025
larryk@westest.com

Abstract: It is important to keep an open mind and be practical with testing software and hardware. Test Program Set (TPS) transportability is a primary consideration for long term customer support goals. Our ATLAS TPSs are robust and user friendly, however, LabVIEW offers several features and state-of-the-art routines and graphical representations that are valuable in many testing techniques or applications. Also, an interoperable open system software test package is an excellent approach to state-of-the-art technology. Vital testing techniques require expansive open systems.

I. Introduction

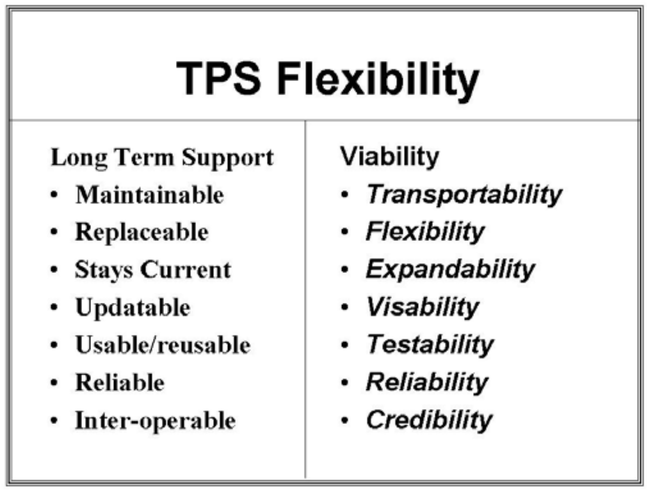
Adding LabVIEW applications brings the power & flexibility of graphical programming to your TPS. LabVIEW gives you the ability to develop your own virtual testing environment. LabVIEW is a graphical programming environment used by millions of engineers and scientists to develop sophisticated measurement, test, and control systems using intuitive graphical icons and wires that resemble a flowchart[1]. LabVIEW includes hundreds of powerful graphical and textual measurement analysis, mathematics, and signal processing functions that seamlessly integrate with LabVIEW data acquisition, instrument control, and presentation capabilities[2].

TPS transportability has become a critical issue. Transportability is a major issue for long term support and viability. We need to support and develop applications for *transportability*, *flexibility* and *expandability* in order to meet customer's needs and requirements. As an example, LabVIEW and LabWINDOWS is highly supported and desired by many customers. As defined below, customer needs and requirements for LabVIEW applications can be achieved:

LabVIEW/LabWINDOWS CVI:

- You can call LabVIEW VIs in LabWindows CVI.
- WesTest's Test Executive can call LabVIEW applications.
- LabVIEW highly supported and desired by many.
- LabVIEW is viable and long term supportable.

WesTest is developing multiple applications which support ATLAS and LabVIEW as well as C++, etc. Likewise this supports LabWINDOWS CVI applications. Figure 1 is a synopsis of our TPS Flexibility to satisfy customer needs and requirements:



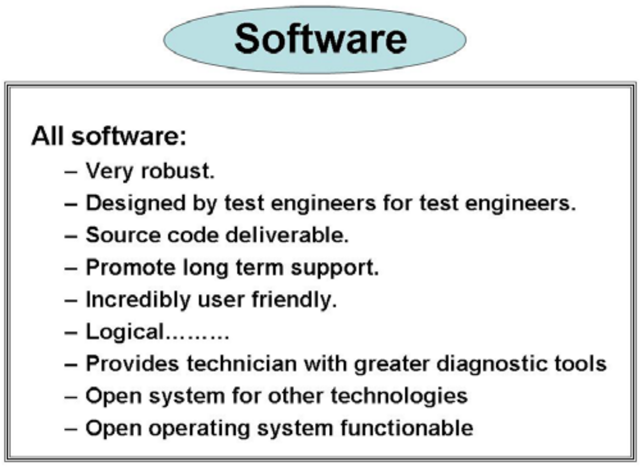
TPS Flexibility

Long Term Support	Viability
• Maintainable	• *Transportability*
• Replaceable	• *Flexibility*
• Stays Current	• *Expandability*
• Updatable	• *Visability*
• Usable/reusable	• *Testability*
• Reliable	• *Reliability*
• Inter-operable	• *Credibility*

Figure 1. Flexibility

Flexibility is critical in terms of Long Term Support and Viability. Maintainability is a must and operating system changes or updates can place your application in a down cycle. Your application should function under all current and updated operating systems. If for any reason an operating system change shuts you down then you must reconsider the maintainability of your application. If your application is not reliably maintainable you will need to reconsider using a technology. Also, turnaround time for application updates to make operating system changes work should be a few days not 6 months.

It is also important to note that a test executive should be maintainable, backward compatible and capable of future enhancements to the system. As shown in figure 2 all software must meet certain criteria. Software should function so that all applications can be employed. Using various software packages together to achieve a total testing environment where an engineer can truly use various techniques to optimize the test and diagnosis process.



Software

All software:
- Very robust.
- Designed by test engineers for test engineers.
- Source code deliverable.
- Promote long term support.
- Incredibly user friendly.
- Logical………
- Provides technician with greater diagnostic tools
- Open system for other technologies
- Open operating system functionable

Figure 2. Software

II. VI's

We are creating VIs for everything and reducing duplicating work. Shown below are applications that can be used. Of course, our VIs can be updated as new techniques are developed.

- **VI's for Instruments**
- **VI's for Diagnostics**
- **VI's for Analysis**
- **VI's for Patterns**
- **VI's for Analog**
- **VI's for Digital**
- **VI's for Probing**
- **VI's for Control**
- **VI's for Routing**
- **VI's for Special Tests**
- **VI's for Unique Applications**
- **VI's for Tricks of the Trade**
- **VI's for Problem-Solving**
- **VI's for Inquiring**
- **VI's for Searching**
- **Etc.**

WesTest's Software Development Environment (SDE) automatically generates ATLAS code. The TPS developer need not write actual test software. This allows the engineer to focus on the test requirements rather than spending wasted time writing software. In contrast, LabVIEW TPS developers program the graphical software for the tests. We use LabVIEW in our applications by performing a function call to a LabVIEW executable, thus maintaining the integrity of the automated code generation feature of our SDE.

LabVIEW VI's can be somewhat intense or they can be written to focus in on certain tests or applications. We do not focus in on one language or limit ourselves to one technique to perform test and diagnosis. An open system allows usage of many applications. We like to use the applications that give us the optimal results. After all, it is the TPS which tests and diagnosis our Units Under Test (UUT).

III. TEST EXECUTIVE and DEBUG

A major factor we considered is the profound power of the features of the Test Executive. We have coupled the Test Executive features with LabVIEW features. LabVIEW features are very useful for specific testing applications likewise; the WesTest Test Executive features are very useful for specific applications. This paper will show some of the features associated with LabVIEW and some of the features associated with WesTest's Test Executive. These features are vast so I will focus on some of the most useful.

In reality, the Test executive is the strength of testing and diagnosis. It is certain we must have a strong instrument sweet to test UUTs; however, the test executive is just as important as strong instrumentation. A limited or weak test executive is actually a hindrance to test/diagnosis.

The system software consists of the operating system (OS), the Test Executive, instrument drivers and associated dynamic link libraries (DLL), and the TPS executables. While all are important, the piece of software that pulls the others pieces together to create a high performance test environment is the Test Executive. The Test Executive must take advantage of a multitasking, multithreaded operating system in order to maximize execution performance. Look heavily at the Test Executive System; its' primary purpose is to control the flow of testing. A weak Test Executive can undermine testing effectiveness and productivity. Remember, the Test Executive first reads in the test program and then controls the instruments, makes decisions and directs the testing in accordance with the instructions contained in the test program and can provide user test flow requests. Providing a powerful Test Executive full of debug capabilities and integration tools is what is needed for efficiency and high level performance in the world of automated test equipment. It is also important to understand that backward compatibility, built-in securities and ease of operator use is also essential for a Test Executive to be complete and maximize support for the test program developer.

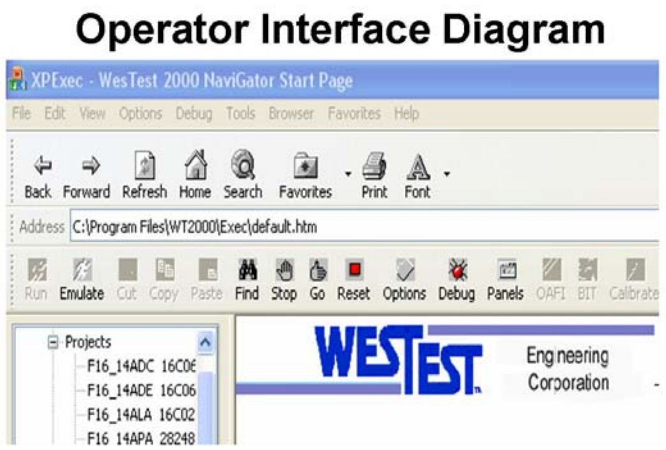
Now let's correlate Test Executive Integration and User tools to LabVIEW integration into a TPS. As shown in figure 3, the test executive should provide integration and User capabilities. Runtime commands are very helpful with controlling TPS execution. Debug commands provide the engineer with tools needed to effectively and economically integrate a TPS. Soft Front Panels are helpful when setting up instrumentation, especially the scaling and timing characteristics of certain instruments. Soft commands are time saving techniques. All these tools are not only beneficial but almost mandatory for proper integration and testing.

Integration & User Tools

- Run time commands
- Debug commands
- Soft front panels
- Soft commands

Figure 3. Integration and User Tools

Run Time Commands are shown in figure 4. Emulate is an important command; emulate runs the test program without power applied. Emulate tests the flow of the test program and can be used to verify certain tests are being performed. After test performance you can use the "Find" command to locate the test output of concern. The "Stop" command will stop the test program on-the-fly while the programs runs (this command is active whether power is applied or not). The "Go" command restarts or resumes the test programs after a "Stop"; this helps if everything is working as planned and you wish to continue testing. The "Reset" command is a station reset and shuts down all instrumentation and power.



Operator Interface Diagram
XPExec - WesTest 2000 NaviGator Start Page
File Edit View Options Debug Tools Browser Favorites Help
Back Forward Refresh Home Search Favorites Print Font
Address C:\Program Files\WT2000\Exec\default.htm
Run Emulate Cut Copy Paste Find Stop Go Reset Options Debug Panels OAFI BIT Calibrate
Projects
F16_14ADC 16C06
F16_14ADE 16C06
F16_14ALA 16C02
F16 14APA 28248
WESTEST Engineering Corporation

Figure 4. Run Time Commands

Other critical run-time commands are associated with the "Options" menu selection. When we are integrating LabVIEW applications it is important to monitor the signal routing. The Test Executive has a selection to show the flow diagrams as shown in figure 5. Signal routing knowledge is another absolute requirement of a proficient test executive. Users and engineers must have this knowledge to mitigate problems.

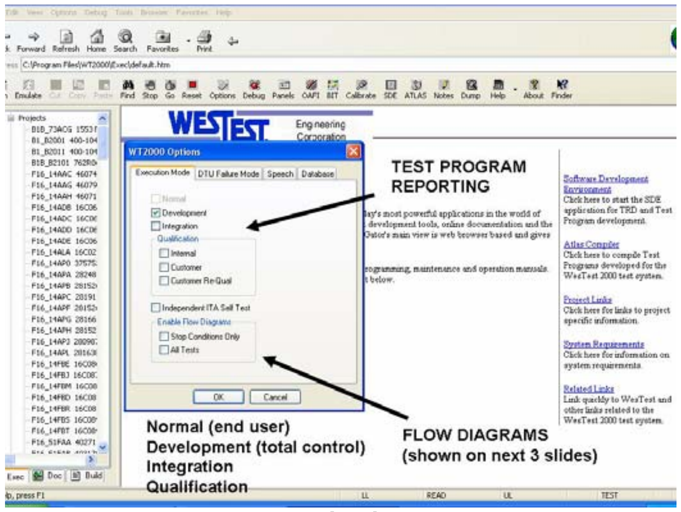


Figure 5. Flow diagrams

The dynamic flow generator aids the developer with a visual display of present conditions including switching data for system switch banks and relays. Updates are continuously made by the test executive. Figure 6 shows a selection of "Test Point Routing"; test point routing is critical when integrating LabVIEW graphical technology. If the engineer wants to use LabVIEW for sensor applications only and not worry about the signal being applied to the instrument then the signal routing must be verified.

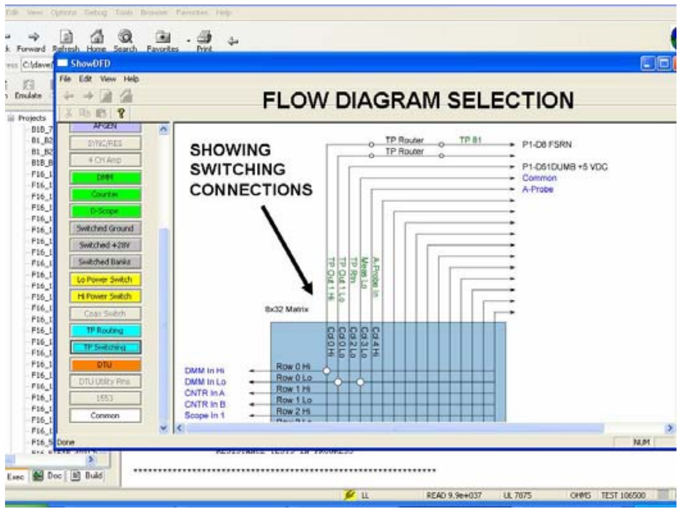


Figure 6. TP Routing

Now, let's consider the Debug tools as shown in figure 7. There are several debug tools that can be used to integrate LabVIEW applications. We need to decide where to place our LabVIEW application in our test program. We have proven it is best to integrate a sensor application after all the routing has taken place and the instrument is connected to the signal we wish to evaluate. The various debug selections are interactive and can be utilized at the engineer's discretion. As you see from figure 7, the flow can be controlled at the discretion of the test engineer.

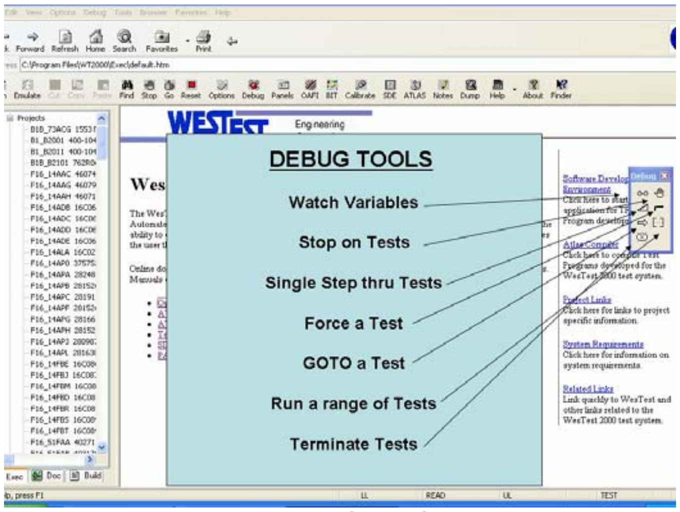
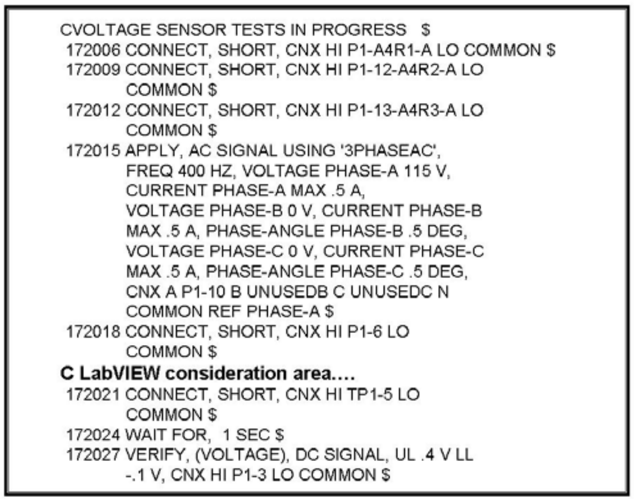


Figure 7. Debug Tools

If the engineer wishes to add a LabVIEW application after statement 172018 as shown below, they can use the debug tools in conjunction with their LabVIEW VI. As shown in figure 8 the engineer is considering adding a LabVIEW application after test number 172018.

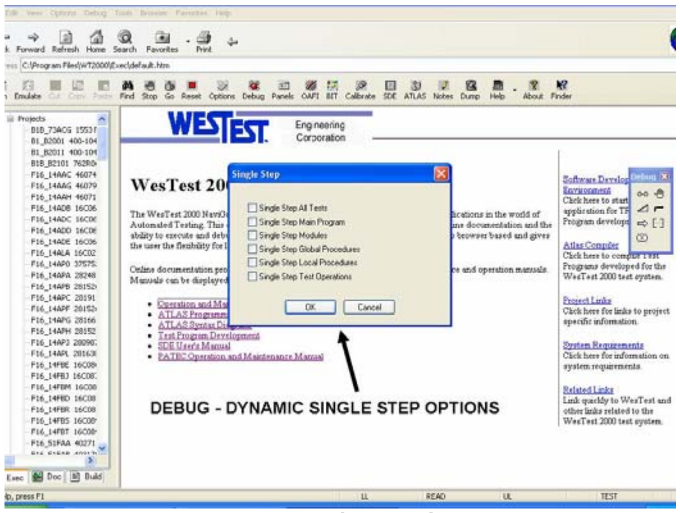


```
CVOLTAGE SENSOR TESTS IN PROGRESS   $
 172006 CONNECT, SHORT, CNX HI P1-A4R1-A LO COMMON $
 172009 CONNECT, SHORT, CNX HI P1-12-A4R2-A LO
        COMMON $
 172012 CONNECT, SHORT, CNX HI P1-13-A4R3-A LO
        COMMON $
 172015 APPLY, AC SIGNAL USING '3PHASEAC',
        FREQ 400 HZ, VOLTAGE PHASE-A 115 V,
        CURRENT PHASE-A MAX .5 A,
        VOLTAGE PHASE-B 0 V, CURRENT PHASE-B
        MAX .5 A, PHASE-ANGLE PHASE-B .5 DEG,
        VOLTAGE PHASE-C 0 V, CURRENT PHASE-C
        MAX .5 A, PHASE-ANGLE PHASE-C .5 DEG,
        CNX A P1-10 B UNUSEDB C UNUSEDC N
        COMMON REF PHASE-A $
 172018 CONNECT, SHORT, CNX HI P1-6 LO
        COMMON $
C LabVIEW consideration area....
 172021 CONNECT, SHORT, CNX HI TP1-5 LO
        COMMON $
 172024 WAIT FOR,  1 SEC $
 172027 VERIFY, (VOLTAGE), DC SIGNAL, UL .4 V LL
        -.1 V, CNX HI P1-3 LO COMMON $
```

Figure 8. LabVIEW Consideration Area

You can use the debug tools to test, debug and integrate the LabVIEW VI. The "Watch Variable" debug tool shows a variable(s) value during a TPS run. Debug "STOP" conditions: CAN STOP ON SPECIFIC TEST or CAN ENTER MULTIPLE TESTS TO STOP ON or CAN STOP WHEN A VARIABLE IS AT A CERTAIN VALUE or CAN STOP ON MULTIPLE VARIABLES or CAN STOP WHEN A TEST FAILS.

Single step debug options are shown in figure 9. The single step selections cover many possible scenarios which can be utilized.

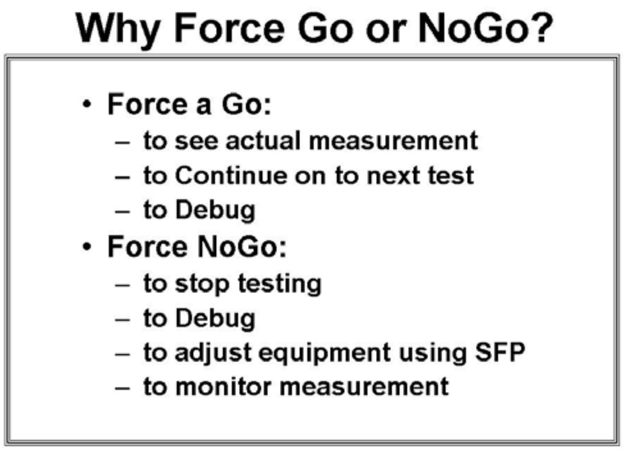


Single Step
- ☐ Single Step All Tests
- ☐ Single Step Main Program
- ☐ Single Step Modules
- ☐ Single Step Global Procedures
- ☐ Single Step Local Procedures
- ☐ Single Step Test Operations

OK Cancel

Figure 9. Single Step Debug

The FORCE GO issued by the TPS developer can be used to FORCE MAINLINE TESTING TO CONTINUE AFTER A FAILURE; the FORCE NOGO USED BY TPS DEVELOPER CAN BE USED TO VERIFY DIAGNOSTICS. Refer to figure 10 for a detailed explanation.



Why Force Go or NoGo?

- **Force a Go:**
  - **to see actual measurement**
  - **to Continue on to next test**
  - **to Debug**
- **Force NoGo:**
  - **to stop testing**
  - **to Debug**
  - **to adjust equipment using SFP**
  - **to monitor measurement**

Figure 10. Force Go or NOGO

The debug "GO TO A SPECIFIC TEST" is USED TO BYPASS PREVIOUS TESTING this is ESPECIALLY USEFUL FOR STAND ALONE TESTS to DEBUG. The debug "RUN A RANGE OF TESTS" can be USED TO BYPASS PREVIOUS TESTING this is ESPECIALLY USEFUL TO DEBUG INTERACTIVE TESTS. The stop execution is used to stop the test program and power down.

LabVIEW provides debug tools like Execution Highlight, Single-Stepping, Probes, and Breakpoints. The correlation between the Test Executive debug tools and the LabVIEW debug tools will be discussed.

LabVIEW Execution Highlighting allows one to view an animation of the execution. Execution highlighting shows the flow of data on the block diagram from one node to another by using bubbles that move along the wires. We use execution highlighting in conjunction with single-stepping to see how data moves from node to node through a VI.

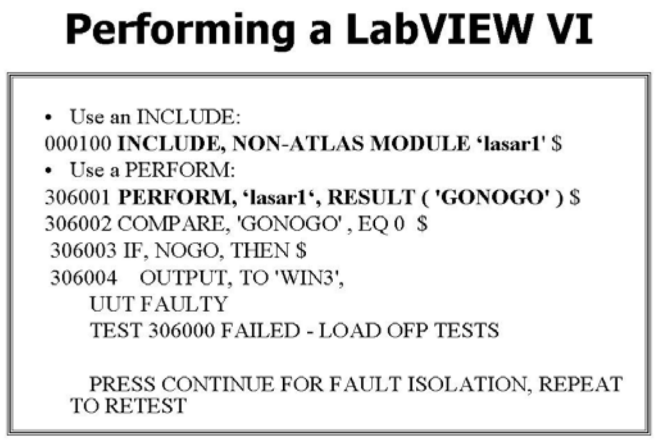
LabVIEW Single-Stepping allows one to Single-step through a VI to view each action of the VI on the block diagram as the VI runs. The single-stepping buttons affect execution only in a VI or subVI in single-step mode.

LabVIEW probes allow one to use the Probe tool to check intermediate values on a wire as a VI runs. When execution pauses at a node because of single-stepping or a breakpoint, you also can probe the wire that just executed to see the value that flowed through that wire.

LabVIEW Breakpoints allow one to use the Breakpoint tool to place a breakpoint on a VI node or wire on the block diagram and pause execution. When you set a breakpoint on a wire, execution pauses after data passes through the wire. You can place a breakpoint on the block diagram workspace to pause execution after all nodes on the block diagram during execution.

Using the STOP on test function with a LabVIEW application is an excellent way to use the power of both the Test Executive and LabVIEW debug tools together. We use Stop on Test to debug LabVIEW or pull up the VI and trace through the subvi's.

To integrate a LabVIEW application into one of our TPSs involves a direct function call as entered into the SDE as shown in figure 11.



Performing a LabVIEW VI

- Use an INCLUDE:

```
000100 INCLUDE, NON-ATLAS MODULE 'lasar1' $
```

- Use a PERFORM:

```
306001 PERFORM, 'lasar1', RESULT ( 'GONOGO' ) $
306002 COMPARE, 'GONOGO' , EQ 0  $
 306003 IF, NOGO, THEN $
 306004    OUTPUT, TO 'WIN3',
      UUT FAULTY
      TEST 306000 FAILED - LOAD OFP TESTS

      PRESS CONTINUE FOR FAULT ISOLATION, REPEAT
   TO RETEST
```

Figure 11. Performing a LabVIEW Application

As shown in figure 11, statements 000100 and 306001 are the statements associated with the LabVIEW application. In this case we are performing a LASAR Test and the result is a GO/NOGO flag. We use our Test Executive LASAR diagnostic routines should a NOGO occur as shown in figure 12. One of the nice things about this approach is the reuse of our LASAR VI. Also, LabVIEW single-step debug is excellent for LASAR TPS development and debug and some engineers prefer this method.

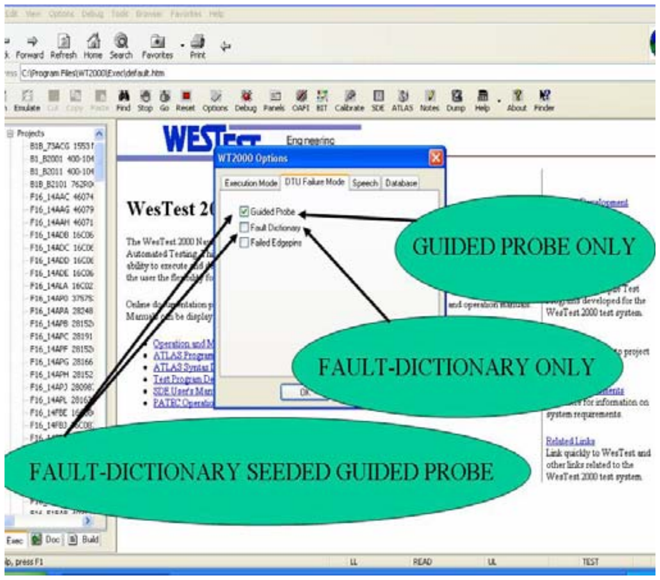


Figure 12. LASAR Diagnostics

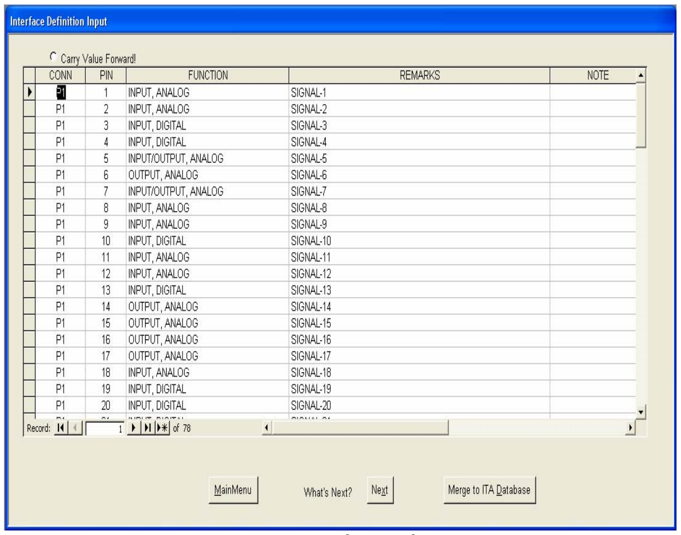
IV. SDE INTEGRATION

We used the automatic routing feature of WesTest's Software Development Environment (SDE) to obtain the actual CONNECT/DISCONNECT statements for signal routing of our LabVIEW application. Signal routing automation is one of the more critical and time consuming issues associated with TPS development. Since, our SDE automatically creates the signal routing and syntax we can utilize this information for stimulus and sensor LabVIEW applications. This saves considerable LabVIEW programming requirements; we do not need to add the connection VI into our LabVIEW VI. We use our "VERIFY W/D" (VERIFY WITH DELAY) SDE feature for sensor signal routing.

The LabVIEW application was programmed using the Software Development Environment (SDE). A robust SDE with Interface Test Adapter (ITA) to instrument signal routing is critical in reducing the complexity of the LabVIEW and making the desired application measurement.

The ITA data base entry form is used to create the Interface Test Adapter Database. The ITA Database is used by the ATLAS Compiler to locate UUT connections, make switch connections and assign resources. The database is built by entering the UUT and tester connections to the ITA. The UUT connection information is imported from the Interface Definition Form.

This Interface Definition Entry Form provides a means of entering the UUT pins and their test requirements into the database as shown in figure 13. The data entered in this form is used in the TPS Test Strategy Report. It is also used in building the ITA database.

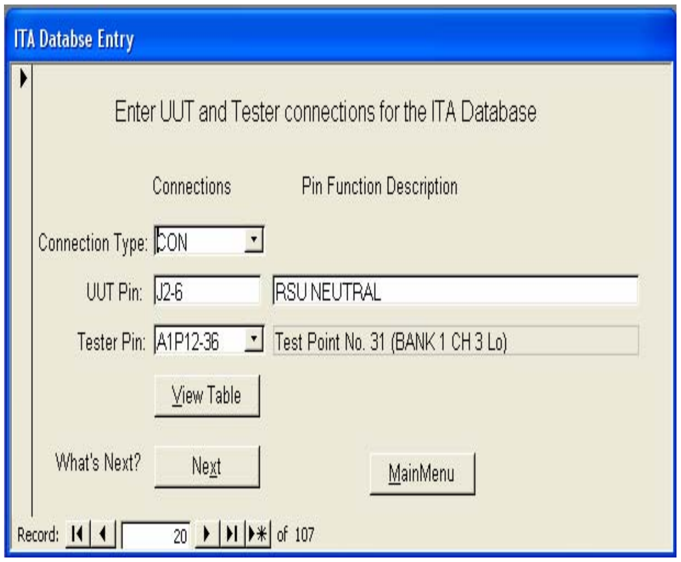


CONN	PIN	FUNCTION	REMARKS	NOTE
P1	1	INPUT, ANALOG	SIGNAL-1	
P1	2	INPUT, ANALOG	SIGNAL-2	
P1	3	INPUT, DIGITAL	SIGNAL-3	
P1	4	INPUT, DIGITAL	SIGNAL-4	
P1	5	INPUT/OUTPUT, ANALOG	SIGNAL-5	
P1	6	OUTPUT, ANALOG	SIGNAL-6	
P1	7	INPUT/OUTPUT, ANALOG	SIGNAL-7	
P1	8	INPUT, ANALOG	SIGNAL-8	
P1	9	INPUT, ANALOG	SIGNAL-9	
P1	10	INPUT, DIGITAL	SIGNAL-10	
P1	11	INPUT, ANALOG	SIGNAL-11	
P1	12	INPUT, ANALOG	SIGNAL-12	
P1	13	INPUT, DIGITAL	SIGNAL-13	
P1	14	OUTPUT, ANALOG	SIGNAL-14	
P1	15	OUTPUT, ANALOG	SIGNAL-15	
P1	16	OUTPUT, ANALOG	SIGNAL-16	
P1	17	OUTPUT, ANALOG	SIGNAL-17	
P1	18	INPUT, ANALOG	SIGNAL-18	
P1	19	INPUT, DIGITAL	SIGNAL-19	
P1	20	INPUT, DIGITAL	SIGNAL-20	

Figure 13. Interface Definition

Connection information defines the type of connection that will be routed to the ITA. Connection Type may be CON, or JMP. No other connection types are valid. CON (connect) defines UUT to Tester connections and digital UUT inputs and digital UUT outputs. JMP (jumper) defines tester to tester connections.

In the Tester Pin field enter the tester pin that the Interface Test Adapter will connect to the UUT Pin. A description of the Tester Pin will automatically be displayed to the right of the tester Pin field. Note: This form must be completed before any tests can be generated with the Test Input Forms. An actual entry data is shown in figure 14.



Connection Type: CON
UUT Pin: J2-6 RSU NEUTRAL
Tester Pin: A1P12-36 Test Point No. 31 (BANK 1 CH 3 Lo)

Figure 14. Actual SDE Entry

This ATLAS View of the Test Input Form allows the engineer to view the ATLAS code that the SDE will generate for the data entered in the current test number. This code may not be edited on the screen; it is for informational purposes only.

When this view is selected in the SDE you will see the words “Double Click Here to Generate Atlas View for This Test”. Double click inside of the dialog area and the SDE will display the code for the selected test.

The Operator Instructions View in the SDE of the Test Input Form is used to input operator instructions that will be displayed at the beginning of the test. This may include, but not be limited to, instructions for the operator to turn on UUT cooling air, set UUT switches, or move test clips. Any instructions entered here will appear on the test station display panel at run time. Instructions for UUT probing during fault isolation are defined here.

The word PROBE, followed by a UUT test point, is a keyword for the SDE. Based upon the keyword PROBE the SDE will include the correct UUT point as the test point in the TRD test point connection field. In this case, the operator instruction entered was “Did the NOISE level settle to LESS THAN .03 Volts for each run of the NOISE test.” The noise test operator instruction is a crucial part of this unique test and the operator interaction.

It is favorable to add explicit Test Requirements Document (TRD) Notes. The TRD Notes View of the Test Input Form of the SDE contains test notes that are required in the TRD. Notes are generally used when test flow does not follow the standard TRD format. These notes must make the TRD a stand-alone document to properly generate the ATLAS code. The notes should add sufficient readability to aid in future revisions or TPS updates. Also, notes can be used to explain the reasoning associated with unique tests and the usefulness of these tests.

When using some unique instruments, it may be necessary to generate the ATLAS code manually. This is true when using the Summing Amplifier (ATLAS Complex Signal) and when writing directly to a device such as the 1553 bus simulator.

Use the Sequence Editor form to create a sequence for the code. If the purpose of the manual code is to apply stimulus to the UUT, create a Stimulus sequence. If the purpose of the manual code is to perform a measurement on the UUT, create a Measurement sequence. You can manually enter ATLAS code by clicking on the **Manual Code** tab.

In the Sequence drop down box enter the sequence number, or P if this code should be inserted in the PREAMBLE. Enter the ATLAS code in the text box. The first line of code entered MUST be a COMMENT. This comment will be inserted in the TRD as a description for the sequence. Do not enter test numbers for the ATLAS code; this will be filled in by the SDE.

The SDE is a powerful development tool. Much of its power is provided by the programs underlying structure. The ability to generate a desired sequence of tests may not be obvious to the developer. A section is designed to provide examples of development techniques for special applications. As techniques are developed for special situations they will be added to the examples section.

Some non-atlas modules return a GO/NOGO flag to the Test Executive, but do not generate a UUT callout. In these cases, it is necessary to have the SDE generate the NOGO path in the ATLAS code and the TRD. Since the SDE will only generate this path if there is an Upper or Lower Limit entered in the Measurement sequence, a N/A should be entered in both the Upper Limit and Lower Limit text boxes.

V. Noise

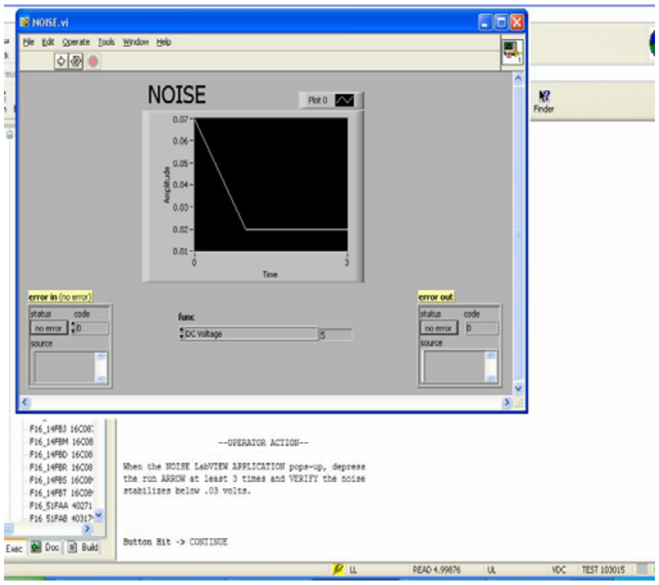
Noise can be measured as root mean square (RMS) or peak to peak. Low frequency noise with a low peak to average ratio is often measured as RMS root mean value of the square of the signal level. A National Instruments (NI) Digital Multimeter uses an onboard Digital Signal Processor (DSP) to compute the RMS value from digitized samples of the AC waveform. The result is quiet, accurate, and fast-settling AC readings[3]. Many precision digital multimeters (DMM) have true RMS capability; the meter does this by collecting thousands of samples[4].

Knowing our connection requirements are handled outside our LabVIEW application, we prepare our LabVIEW VI for signal stimulus or sensor applications. We use a PERFORM statement like "PERFORM, '\NOISE' $ to activate the LabVIEW application. In conjunction with the "PERFORM" statement we must identify the NON_ATLAS application by identifying the LabVIEW application in the preamble section of the TPS in the SDE. We use the following syntax to identify the LabVIEW application in the preamble "INCLUDE, NON-ATLAS, \'dmm\noise' $."

Below is shown the actual code created for a LabVIEW application:

```
B BRANCH FOR REPEAT OF TEST 1035 $
 103500 CONNECT, (VOLTAGE), DC SIGNAL,
CNX HI P1-64 LO COMMON $
103501 WAIT FOR, 1 SEC $
103502 PERFORM, '\DMM\NOISE' $
C (this is the LabVIEW application) $
103503 DISCONNECT, (VOLTAGE), DC SIGNAL,
CNX HI P1-64 LO COMMON $
```

The noise display from the LabVIEW application is shown in figure 15.



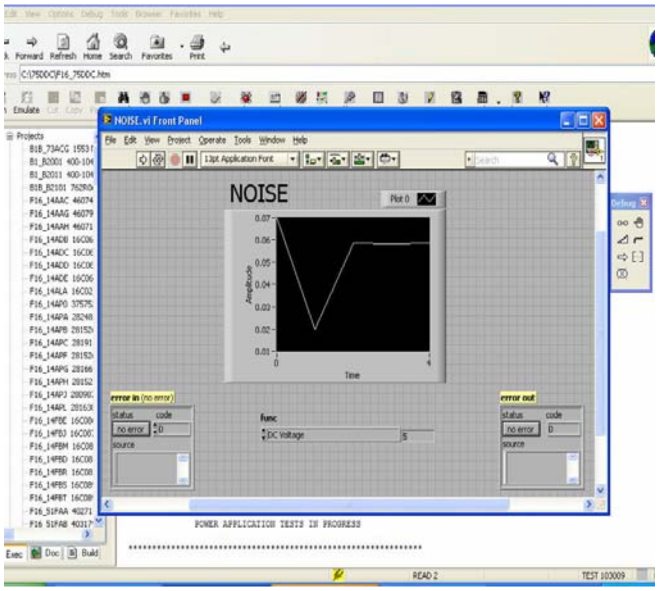
```
--OPERATOR ACTION--

When the NOISE LabVIEW APPLICATION pops-up, depress
the run ARROW at least 3 times and VERIFY the noise
stabilizes below .03 volts.

Button Hit -> CONTINUE
```

Figure 15. LabVIEW application showing a passing test

Figure 15 shows a UUT passing the noise test. Figure 16 shows a UUT failing the noise test. The author is merely showing some of the unique testing techniques that can be used with LabVIEW and is not necessarily implying this test should be performed.



```
POWER APPLICATION TESTS IN PROGRESS

**************************************************************
```

Figure 16. LabVIEW application showing a failing test

Interesting measurements and techniques add credence to using LabVIEW for certain tests. In this case, displaying a true RMS DMM display is quite a bit different than looking at thousands of samples gathered by the DMM.

A Noise figure or relative application definition term is a key performance parameter in many systems. As a parameter in a in the overall UUT, a lower or stable noise figure gives a key indication of system performance. In a development laboratory, noise figure measurements are essential to verify new designs and support existing equipment. In a repair environment, noise signals can be measured in unique fashion. Even so, it might be necessary to measure noise to demonstrate that the UUT meets specifications.

The term noise can be used to describe anything that obscures a desired signal. It is true that noise can be divided into two main types: interference and random noise. Interference occurs when an undesired signal obscures a desired one. Interference noise can be generated by test equipment or your ITA design. It is important to determine if the noise is being generated by an external source or something outside the UUT. Random noise can arise from physical sources, such as from thermal fluctuations. It is often associated with resistance and is expressed in terms of voltage fluctuations across, or current fluctuations in parallel with resistors.

There are two main characteristics of a noise source. The first is its frequency distribution. The second is the physical phenomena which is producing it.

Noise is commonly measured in units of energy per frequency, much like the measurement of electro- magnetic radiation. For a circuit element, this quantity is often expressed in units of V^2rms=Hz. The variable for noise voltage is "E," which distinguishes it from "V," the variable for signal and power voltage. In addition, noise adds in quadrature. If there are two independent noise sources A and B, then the total noise energy created by them is $E^2_{tot} = E^2_A + E^2_B$.[6]

The conversion of electrical energy to heat causes noise. For resistors of different compositions, this process occurs in physically different ways. Each dissipative process is accompanied by characteristic fluctuations which can be measured with a spectrum analyzer. This can yield a "noise signature" much like the spectral emission lines of an atom or molecule. The tunnel junction is a resistive structure with interesting physical properties. A tunnel junction consists of two conducting thin films with a thin layer of insulating dielectric in between. Placing an electrical potential across the conducting films causes electrons to tunnel through the dielectric. If impurities are introduced into the dielectric, the tunneling rate can be affected, either positively or negatively, thereby affecting the resistance. In addition, impurities affect the noise signature by introducing peaks and valleys into an otherwise smooth curve. By studying the noise spectra of samples, insight can be gained into the different mechanisms by which electrons travel through the substance[6].

The author is not attempting to define noise as a specific entity. Noise is used quite randomly and can be specified for any given application. What is important about noise is that random signals or variances can occur in UUTs that are indicative of problems with components or the connections between components.

VI. GLITCHES

Capturing rare glitches is important for analyzing random failures. LabVIEW features are excellent at showing glitches by utilizing LabVIEW's graphical power and analytical techniques. Troubleshooting signal instability can be rather involved and somewhat impractical however, LabVIEW lends itself well in resolving these types of testing problems.

The best way to characterize the amount of instability in a key signal parameter is to use a histogram of the parameter. You can see the maximum range of parameter values, the RMS variation and the shape of the distribution[7].

In addition to finding intermittents or glitches you may also want to look for rare signal shapes due to reflections, noise bursts, frequency drift, unstable *power quality* or other factors that produce rare signal events and cause failures in the weapon system. There is technology which can isolate rare events, measure them and provide both measurements and views and this is one of the great features of LabVIEW. Using LabVIEW for certain applications you can find rare abnormalities in a signal. The result is the isolation of signal faults which typically could not be found or realized using traditional methods.

There are a variety of new and better techniques for viewing and troubleshooting the source of rare/intermittent circuit misbehaviors using LabVIEW. Using unique graphs and data analysis will reveal time domain information about circuit instability. LabVIEW has a variety of tools for finding rare/intermittent patterns. We can even find things that cannot seemingly be defined. Therefore LabVIEW can provide unique ways to test for unstable conditions.

The LabVIEW waveform chart is a special type of numeric indicator that displays one or more plots of data typically acquired at a constant rate. The following figure shows an example of a waveform chart[5].

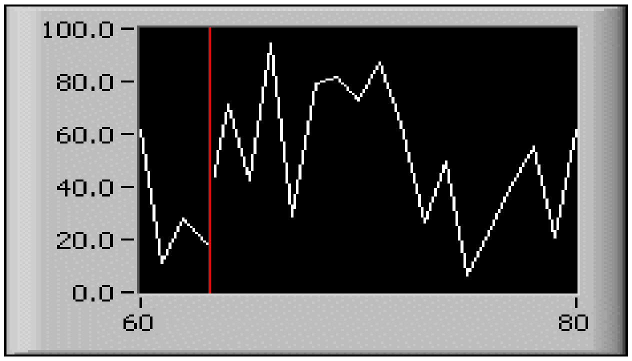


Figure 17. Waveform Charts

We can use LabVIEW to add special graphical displays of signal characteristics which are usually unseen. This information which was previously not used can be useful for test/diagnosis. We look forward to seeing information previously unseen and determining its usefulness.

Overall, signal disturbances are not always detected when performing standard measurements. Standard measurements are the accepted or most understood measurements which tend to provide the desired information as to the UUT functionality. Standard measurements are a must when performing test and diagnosis but sometimes we need to think outside the box to correct or detect a weird problem with certain UUTs.

Standard diagnostic schemes sometimes don't do enough analysis to focus in on the actual cause of a test failure. At times measurements will be border-line on test sequences prior to an actual test failure. These border-line measurements can be used to aid in the determination of an actual fault or used to determine if additional testing should be performed. An actual fault and the associated test that should detect that fault can be deceiving. The specific test in question can pass but be right on the border-line. The failure might not show up in testing until a later test is performed. This later test assumes all the prior tests passed and therefore the circuitry associated with these prior tests is good. This is not necessarily the case if some of the output measurements prior to the actual failing test were right on the border-line. There are many other factors which can show problems.

What can we do? Take advantage of the order in which the faults are simulated and add additional up-front testing to help determine problems quickly and more accurately. We should structure our TPSs such that a review of preliminary tests should be evaluated before the R/R component list is presented. The review of the preliminary test can be rather straight forward. We can look at signals that are within 8-10% of the lower or upper limit. We can then use an inter-related test scheme to evaluate the test(s) that can be associated with the actual failing test. We can add a LabVIEW diagnostic test to aid in fault isolation.

With current generation and next generation test systems focusing on testing efficiency, it is critical to develop test strategies or methods that maximize testing throughput, make better use of the increasingly expensive instruments used in test station and drive down test costs.

Test Strategy analysis should include evaluating existing legacy code or test specifications, at times it might seem the legacy code does not seem to properly examine Input/Output pin coverage, component failure mode coverage and ambiguity group size assessment. However, the legacy TPSs test strategy often is based on factors unbeknownst to the re-host test engineer. It is good to question a test strategy, but if a test sequence is of good quality and does the job, it should be used. However, additional testing could still be required to optimize the repair process.

A TPS Re-host design, development, manufacture, integration and debug for a Line Replaceable Unit (LRU) and/or Shop Replaceable Unit (SRU) includes the analysis, assessment and improvement of the legacy test program or test specification, component fault coverage and diagnostic isolation capabilities. Also, a TPS re-host should include an analysis of Cannot Duplicate (CND) and Re-test OK (RTOK) occurrences.

Terminology such as Cannot Duplicate (CND), Re-test OK (RTOK), No Fault Indicated (NFI), No Fault Found (NFF), and No Trouble Found (NTF), are used to describe the inability to replicate field failures during repair shop test/diagnosis. This paper uses CND to refer to all such failures. On specific Units Under Test, CND failures can make up more than 85% of all observed field failures in avionics and account for more than 90% of all maintenance costs. These statistics can be attributed to a limited understanding of root cause failure characteristics of complex systems, inappropriate means of diagnosing the condition of the system, and the inability to duplicate the field conditions in the laboratory.

When a unit is tested outside its' operating system, it has normally been removed due to a fault. Sometimes these faults are extremely difficult to detect and at times unique testing must be performed to isolate the fault. The external test may not discover any fault and a CND event may occur. The CND occurrence is a major problem when dealing with complex technical systems, and its consequences may be manifested in system down-time and increased life cycle costs. There are multiple interacting causes of CND, demanding tough requirements for successful solutions. There are ways and technologies that provide possible improvements for the prevention of causes of CND and the reduction of its consequences. The identified causes and solutions are related to life cycle stages, availability performance factors, test technologies, and system stakeholders.

The depot shop or the actual users of TPSs have hands-on knowledge of the quality of the TPS and provide great insight into weaknesses or things that randomly occur. The actual test program expert is the actual user. The user becomes familiar with the program weaknesses and strengths. Often, they can perceive actual component failures based on test program failures or higher level failures without using the diagnostic routine. In reality, the user performs actual TPS validation. Validation refers to evaluating system performance to establish compliance with functional requirements and assess system accuracy and correctness. Verification, on the other hand, establishes structural correctness and process effectiveness by evaluating the test program for logic; while testing executes a piece of software with the goal of finding errors; actual testing needs to exercise a set of test cases on many paths, although it does not guarantee that each path or signal characteristic is tested. Functional testing validates problem specifications by comparing system output with known results. Both functional and actual testing is necessary to build reliable test programs.

VII. Conclusion

Although we pride ourselves in the quality of the TPS using proven techniques, further signal analysis does improve the quality of a TPS and improves the diagnostic process. Adding LabVIEW tests can improve the Re-Test Okay (RTOK) rates by seeing things previously hidden from view.

LabVIEW can enhance the quality of a TPS. Using LabVIEW in our Test Executive has proven an effective way to detect failures previously not seen. It is nice to use LabVIEW instrument VI's in conjunction with LabVIEW analysis tools without coding in signal routing. Since our SDE automatically sets-up the signal routing, we do not need to add this to the VI. We connect the instrument path so the VI merely needs to acquire the signal and perform evaluation or analysis.

We do not know the extent to which we can utilize VIs for signal analysis in the future, but we foresee the possibility of many unique tests to resolve issues like Could-Not-Duplicate (CND) or RTOK problems. Also, we foresee better diagnostic routines using LabVIEW for certain diagnostic evaluation and analysis. All in all, we have proven the integration of high quality tools like LabVIEW into testing applications is a wise decision. We must continue to improve our test/diagnosis to make certain our REPAIR PROCESS IS OPTIMAL.

Techniques employed up-front in the TPS development process can save many dollars over the life of a weapon system in areas like changing good parts, man-hours expended in determining UUT faults, downtime for weapon systems, etc.

Overall, more flexibility and more money spent up front on TPS development will pay for itself during the overall weapon system life-cycle.

If we utilize our skill up-front and completely utilize knowledge gained as TPSs are developed and as units are tested, we can produce quality improvements. There is no question that the overall test performance and strength of a test system lays within the Test Executive. This is a case in which the Test Executive has the features that aid the test program developer. Providing a powerful Test Executive full of debug capabilities and integration tools is what is needed for efficiency and high level performance in the world of automated test equipment. Without a doubt, such a Test Executive has a big impact on diagnostics, reliability and maintainability. It is also important to understand that backward compatibility, built-in securities and ease of operator use is also essential for a Test Executive to be complete and maximize support for the test program developer.

References:

1. National Instruments on the web, 2010, What is NI LabVIEW?, http://www.ni.com/labview/whatis/
2. Connections on the web, 2010, NI LabVIEW Getting Started FAQ, http://cnx.org/content/m15428/latest/
3. National Instruments on the web, 2010, Low Frequency and DC Measurement Fundamentals, http://zone.ni.com/devzone/cda/tut/p/id/3610#toc2
4. Kay, Art, Analysis and Measurement of Intrinsic Noise in OP AMP Circuits, 4/26/07
5. National Instruments on the web, 2010, Types of Graphs and Charts, http://zone.ni.com/reference/en-XX/help/371361B-01/lvconcepts/types_of_graphs_and_charts/
6. Leon, F.R. et al, Low-current Noise Measurement Techniques, August 1999.
7. Lauterbach, Michael, Capturing and Troubleshooting Intermittent Signal Faults, October 2008