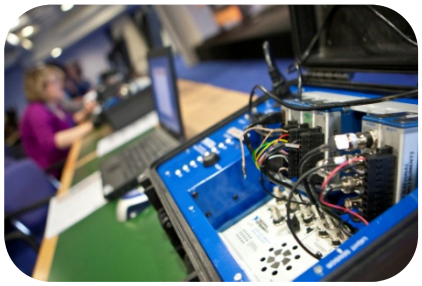
Version 14.10



Build an Automated Test System with NI TestStand and the PXI Platform Hands-On Seminar

Please do not remove this manual

You will be sent an email which enables you to download the presentations and manual

uk.ni.com
ireland.ni.com

NATIONAL INSTRUMENTS

Hardware Requirements:

4 UUT (Units Under Test) Demonstration Unit

PXIe 813x
PXIe 54xx
PXIe 512x
PXIe 654x
PXIe 4143
PXI 407x
PXIe 2532B

Software Requirements:

LabVIEW 2014
TestStand 2014

Exercise 1: Adding Tests Using the NI TestStand Sequence Editor

Goal

Create a TestStand sequence and add steps to the MainSequence that tests a filter, FET, LED, and transmission lines (BERT) on the unit under test (UUT). Figure 1-1 shows the sequence file that will be created through this exercise.

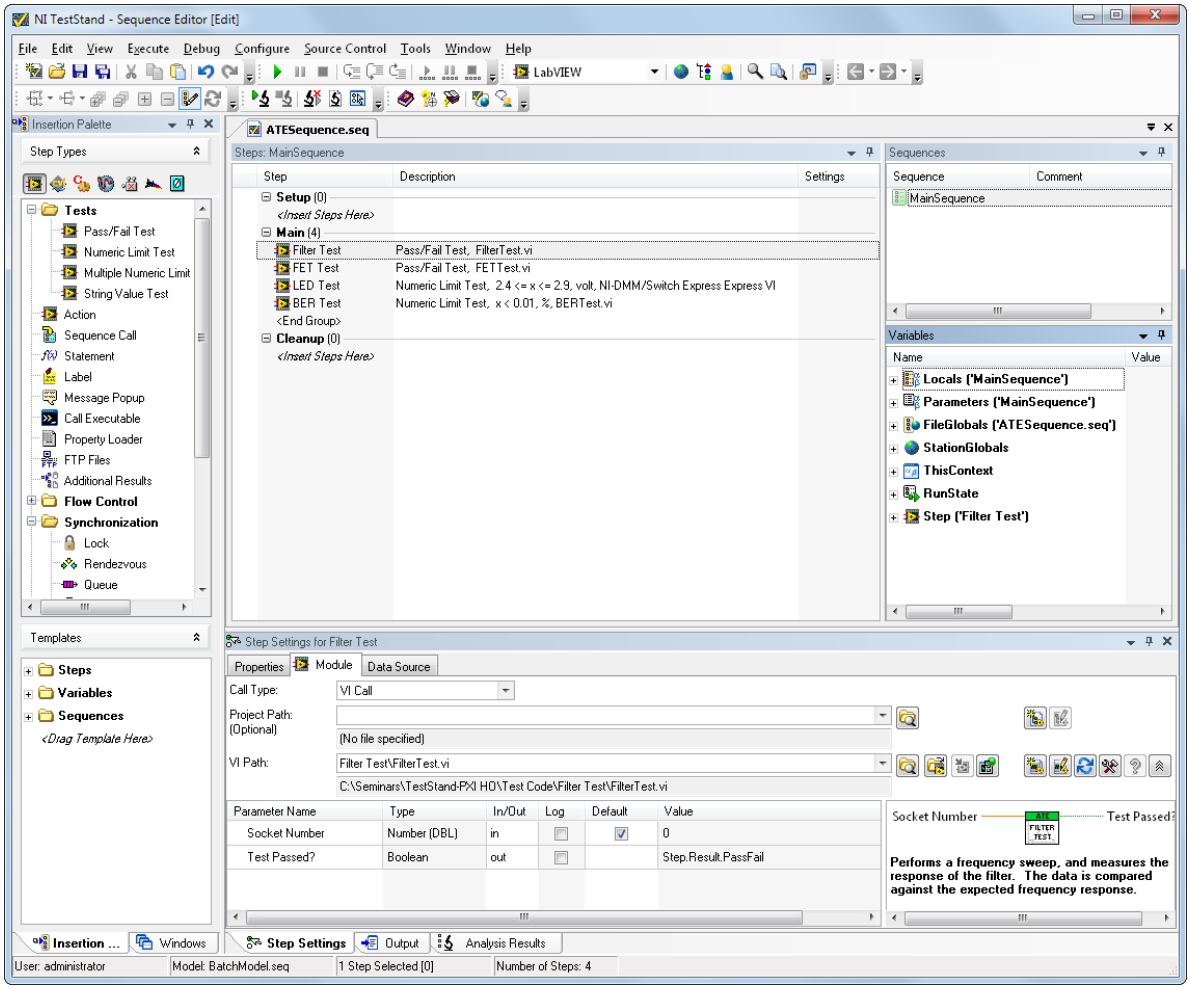


Figure 1-1. Completed Exercise

Description

Part A: Explore the Hardware

1. The PXI Chassis you are using for this seminar contains all of the hardware needed to test components of the Unit Under Test (UUT). The following components of the UUT are tested:

   - Light Emitting Diode (LED) – diode measurement.
   - Low-pass filter – frequency response measurement.
   - Field Effect Transistor (FET) – current response to given gate and sweep voltages.
   - Transmission Lines – validation of wiring integrity with high speed digital signals.

2. To test these components, the system contains the following instruments:

   - Digital Multimeter (DMM) – used to measure voltage for LED test.
   - Arbitrary Signal Generator (ARB) – used to generate a frequency sweep to test the filter response.
   - Oscilloscope – used to measure the filter signal output and measure the magnitude of the result.
   - Source Measurement Unit (SMU) – Sweeps drain voltage and measures drain current response for the FET test. Also provides a precise gate voltage.
   - High Speed Digital Module (HSDIO) – Outputs a known digital signal and checks that the return signal from the UUT matches the expected waveform.
   - Switch – Manages connections between the test sockets containing UUTs and the other hardware. Allows for software controlled switching between UUTs.

3. Open the protective cover to inspect wiring between test fixture and PXI hardware.

Part B: Connect the Hardware to the UUT in Socket 0 Using NI Switch Executive

Before developing the sequence, you must connect the hardware to the UUT to ensure that the tests run successfully.

1. Launch Measurement & Automation Explorer (MAX), located in **Start»All Programs»National Instruments**.

2. In the left-hand pane, select **My System»Devices and Interfaces»NI Switch Executive Virtual Devices»ATE**.

3. In the **Routes and Route Groups** pane, select **ATE_Switch_Configuration**. As shown in the Schematic pane Figure 1-2, this route group contains all of the necessary switch connections (routes) for the LED, Filter, and FET tests (the HSDIO lines are directly run to each UUT and are not run through the switch):

   - Connects the UUT blue LED to the Digital Multimeter (DMM)
   - Connects the UUT Filter to the Oscilloscope (Scope) and the Arbitrary Signal Generator (ARB)
   - Connects the UUT FET to the Source measure Unit (SMU)

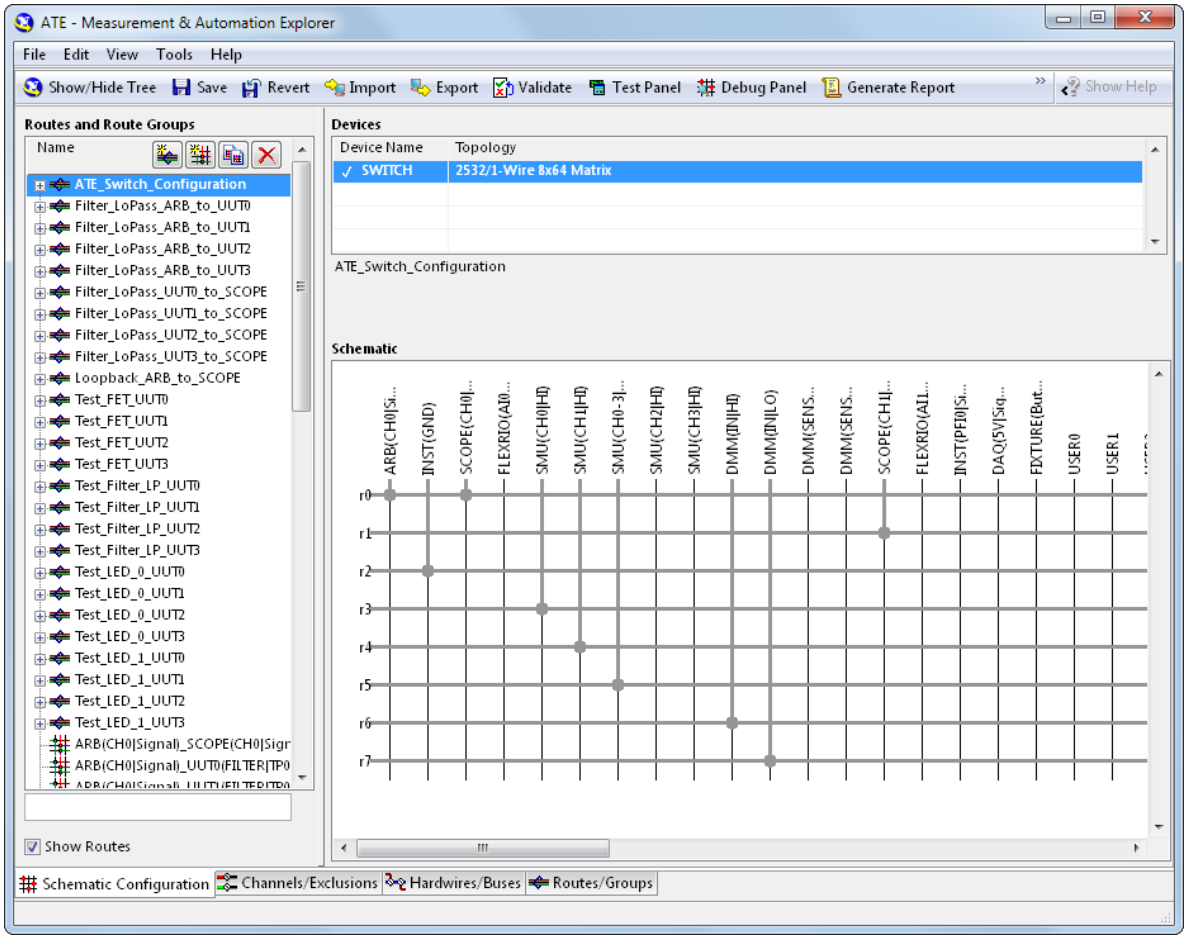


Figure 1-2. NI Switch Executive

4. To use the selected route group, click the **Test Panel…** button in the toolbar, as shown in Figure 1-3.

   Additional Information
   The NI Switch Executive Test Panel allows you to interactively select and test individual routes and route groups associated with a virtual device. In this case, the virtual device is called ATE.

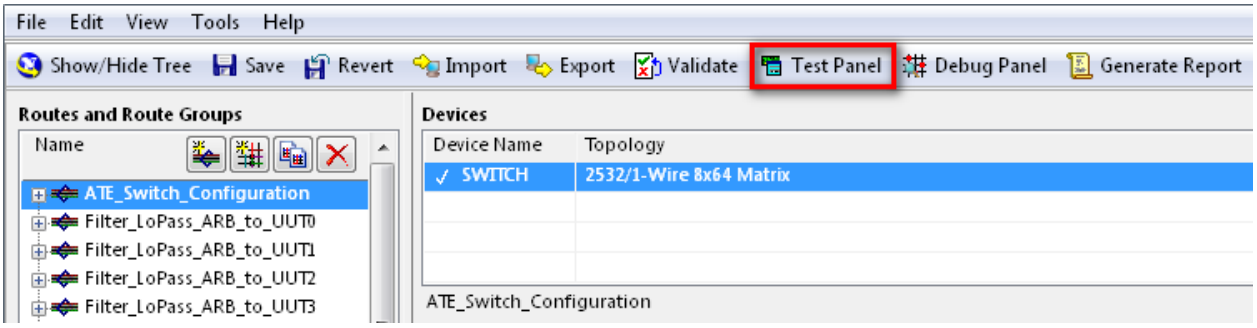


Figure 1-3. Accessing the Switch Executive Test Panel

5. Select the **ATE_Switch_Configuration** route group.

6. Click the **Connect Selected Route or Route Group** button to connect the route group in the switch hardware, as shown in Figure 1-4.

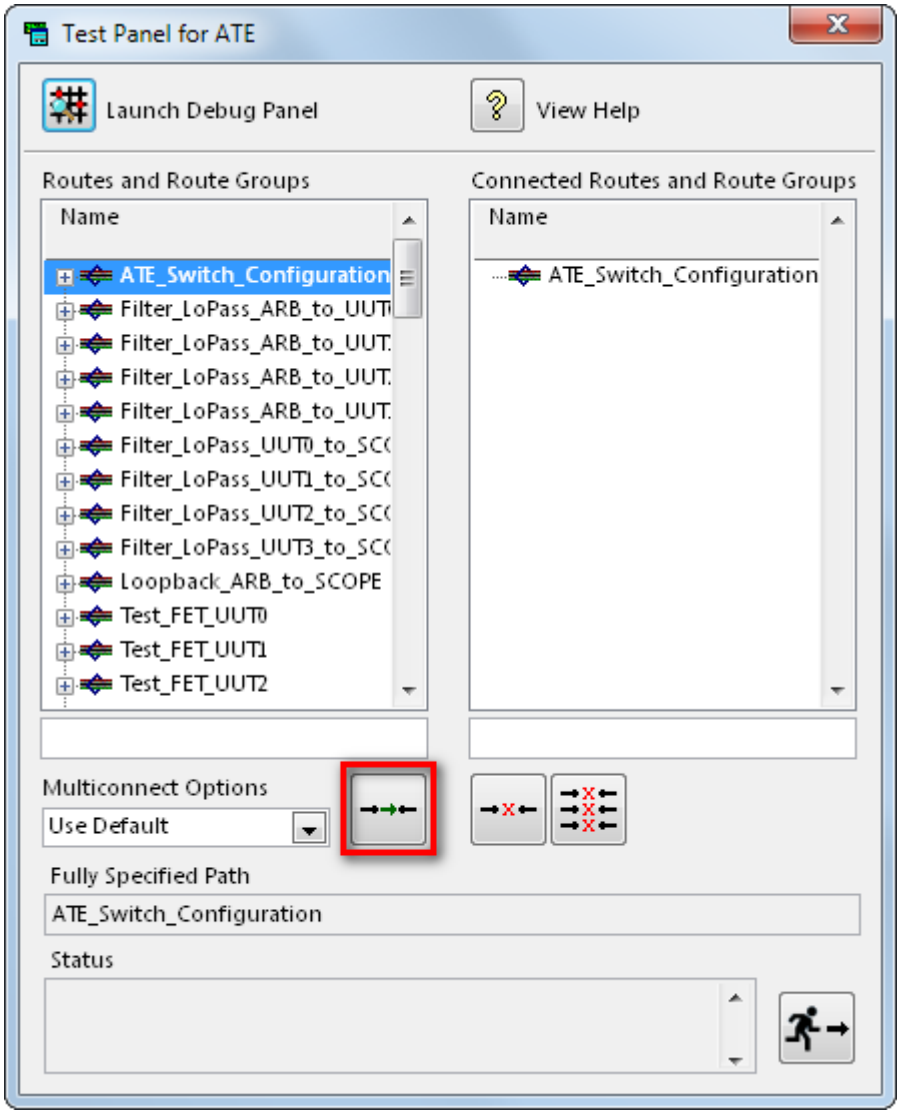


Figure 1-4. Switch Executive Test Panel

7. Click the **Exit Test Panel** button to close the test panel. The route group will stay connected which will allow the testing of each component of the UUT in Socket 0 in the next part of the exercise.

   Note: Opening the test panel disconnects all previous routes. Ensure that ATE_Switch_Configuration is listed as a connected route before exiting. In order to visualize the current connections on the switch, right-click the **NI PXIe-2532 "SWITCH"** item under **My System»Devices and Interfaces»NI PXIe-1082 "Chassis 1"** and select **Test Panels…** This will launch the Switch Soft Front Panel which shows a graphical representation of the switch connections. Minimize or move this window as needed.

Part C: Call the Filter Test VI from TestStand

1. Log in to the TestStand Sequence Editor.
   - Select **Start»All Programs»National Instruments»TestStand 20*xx*»Sequence Editor** to launch the sequence editor.

- When the Login dialog box appears, select the default, administrator, from the **User Name** control and leave the **Password** textbox empty, as shown in Figure 1-5.
- Click **OK** in the Login dialog box.

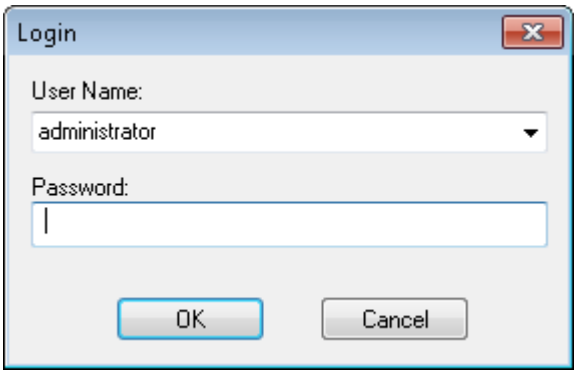


Figure 1-5. Login Dialog Box

2. Select the **New Sequence File** toolbar icon to create a blank sequence file.
3. In the **Adapters** menu, show in Figure 1-6, select the **LabVIEW** adapter. Now, any new steps you create which require a code module (such as the **Action** or **Numeric Limit Test** steps) will be configured to call a LabVIEW VI code module.

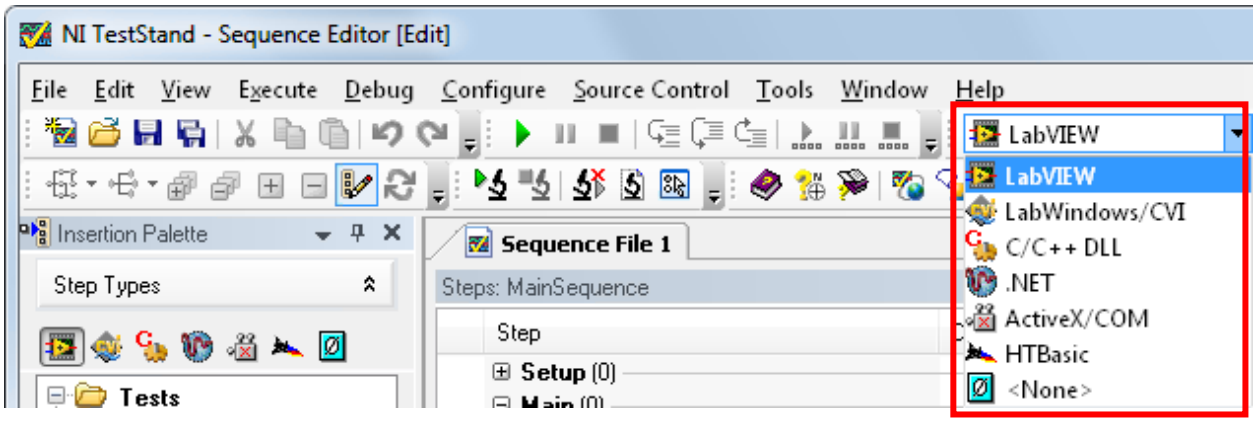


Figure 1-6. Adapter Ring Control

4. Create a new step in the test sequence to execute the Filter test.
   - Expand the **Tests** group in the **Insertion Palette** and drag the **Pass/Fail** test to the **MainSequence** as shown in Figure 1-7.

     Additional Information
     The **Pass/Fail Test** step calls the test code module, and expects a Boolean output from the code module to indicate whether the test passed or failed. Other test types expect different results from the test. For example, a **Numeric Limit Test** step expects a numeric result from the code module, which is then evaluated against limits defined in the step to determine the result of the test.

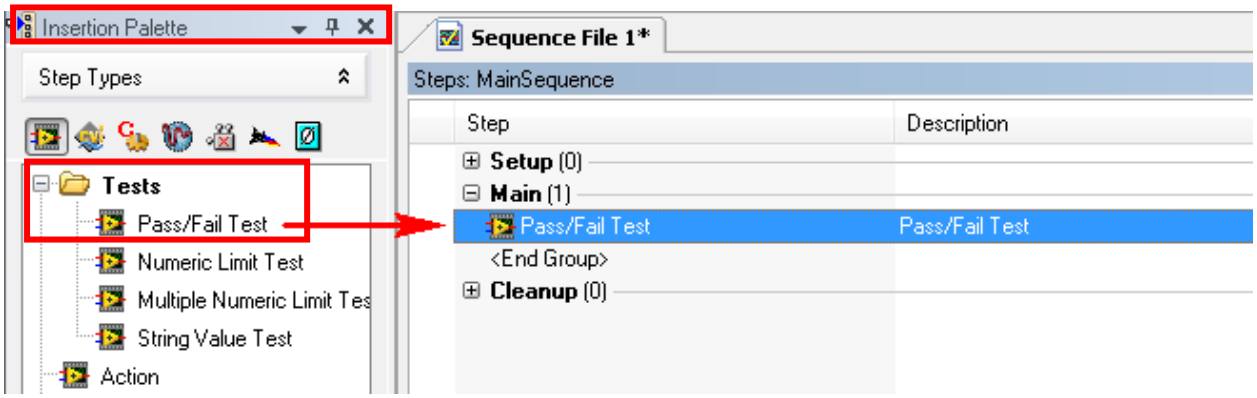


Figure 1-7. Creating the Filter Test Step

- Rename the step **Filter Test**.

5. Configure the step to call the Filter Test code module.
   - Select the **Module** tab in **Step Settings** pane (found beneath the Steps pane) (see Figure 1-8).
   - Click the **File Browse** button on the right side of the **VI Path** field.
   - Select the Filter Test.vi file from the `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Test Code\Filter Test` directory and click **Open**.

Note: If you are prompted with File FilterTest.vi Not Found dialog window, select **Use a Relative path for the file you selected**, or **add the directory that contains the file you selected to the list of search directories** and click **OK**

5. Configure the VI front panel to be displayed when the step executes by clicking the **Show Front Panel** button which will turn the icon green (see Figure 1-8).
6. Define where to store the test result in TestStand. For a Pass/Fail step, TestStand expects that the result of the test code is stored in the PassFail property.
   - Click the *f(x)* button, to the right of the *Value* field (see Figure 1-8) to launch the Expression Browser Dialog Box.

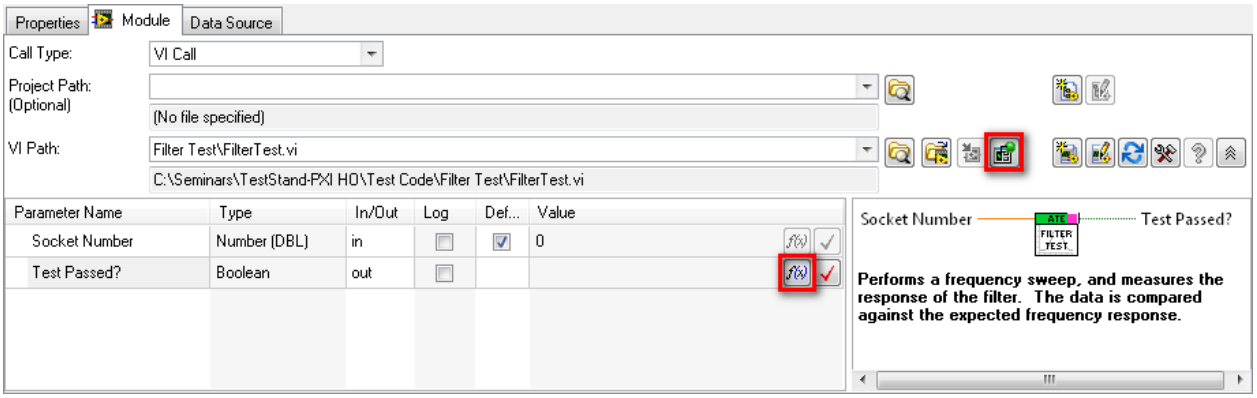


Figure 1-8. Step Settings for Filter Test Window

Additional Information

In the Expression Browser Dialog Box, you can use to interactively build an expression by selecting from lists of variables, properties, operators, and functions. Use the expression dialog to view available properties in TestStand where data can be stored.

- Navigate to **Step.Result.PassFail** as shown in Figure 1-9.

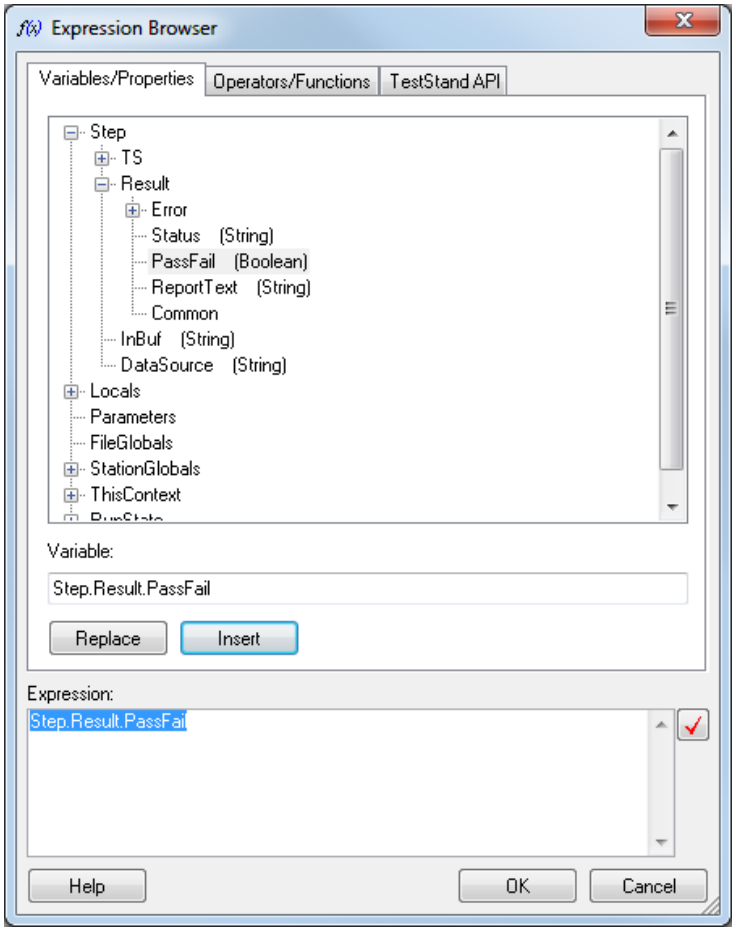


Figure 1-9. Use Expression Browser to Build an Expression

- Click **Insert** to add the selected property to the expression field.
- Click **OK** to close the Expression Browser Dialog Box. The step is now configured to store the result of the filter test in the Step.Result.PassFail TestStand property.

Additional Information
TestStand data is stored in a hierarchy. Certain properties contain subproperties, such as the **Result** property containing the **PassFail** and **Status** properties. When accessing TestStand properties, this containment relationship is represented by a period (e.g. Result.PassFail).

7. Verify that the step settings match closely to Figure 1-10.

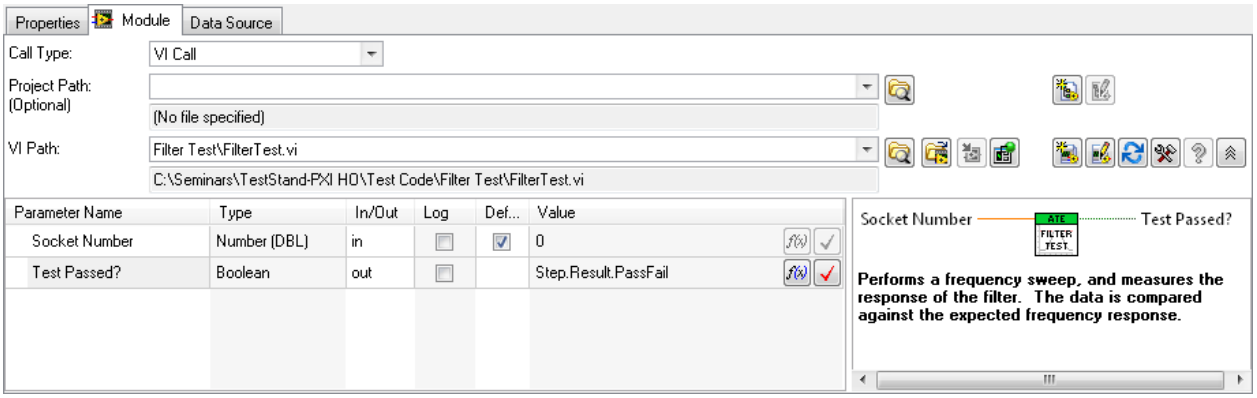


Figure 1-10. Completed Filter Test Step Configuration

Part D: Create the FET Test Step

1. Create a new **Pass/Fail** step to test the FET
   - Drag a new **Pass/Fail** step from the insertion palette to the steps pane, below the Filter Test.
   - Rename the step **FET Test**.

2. Configure the step to call the FET test code.
   - Select the **Module** tab in **Step Settings** pane.
   - Click the **File Browse** button on the right side of the **VI Path** field.
   - Select the FETTest.vi file from the `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Test Code\FET Test` directory and click **Open**.
   - If prompted, choose to use a relative path

3. Configure the VI front panel to be displayed when the step executes by clicking the **Show Front Panel** button which should turn the icon from red to green.

4. Define where to store the test result in TestStand.
   - For the **Test Passed?** Parameter, specify the expression Step.Result.PassFail which can be done by either using the Expression Browser or typing out the text.

Figure 1-11. Step Settings for the FET Test

Part E: Create the LED Test Step

1. Add a step which will test the blue LED on the UUT.
   - The test will return a numeric Voltage value, rather than a Boolean test result. For this reason, we use the **Numeric Limit Test** for this step.
   - Drag the Numeric Limit Test from the **Insertion Palette** to the **MainSequence**, as shown in Figure 1-12.

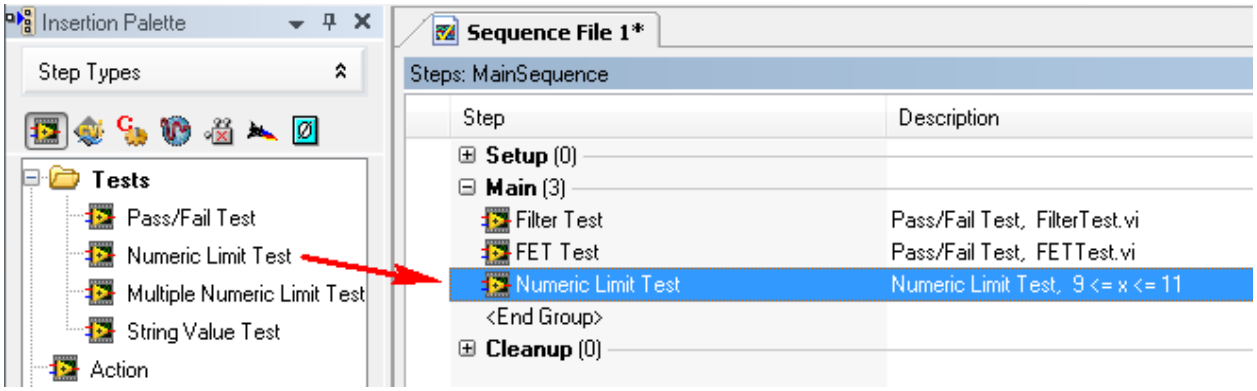


Figure 1-12. Addition of LED Test Step

   - Rename the step **LED Test**.
2. Specify the code module for the LED test. In this case, you will use an NI-DMM Express VI to execute the test.
   - Select the **Module** tab in the Step Settings for LED Test window.
   - Click the Select Express VI icon  in the top right corner.
   - Select **Measurement I/O»NI-DMM»NI-DMM/Switch Express**.

- Configure the NI-DMM Express dialog box as described in Table 1-1 below. The final configuration should match Figure 1-13.

Table 1-1. Configuration Options for the LED Test Step

Configuration Option	Value
Measurement function	Diode
Current source	1mA

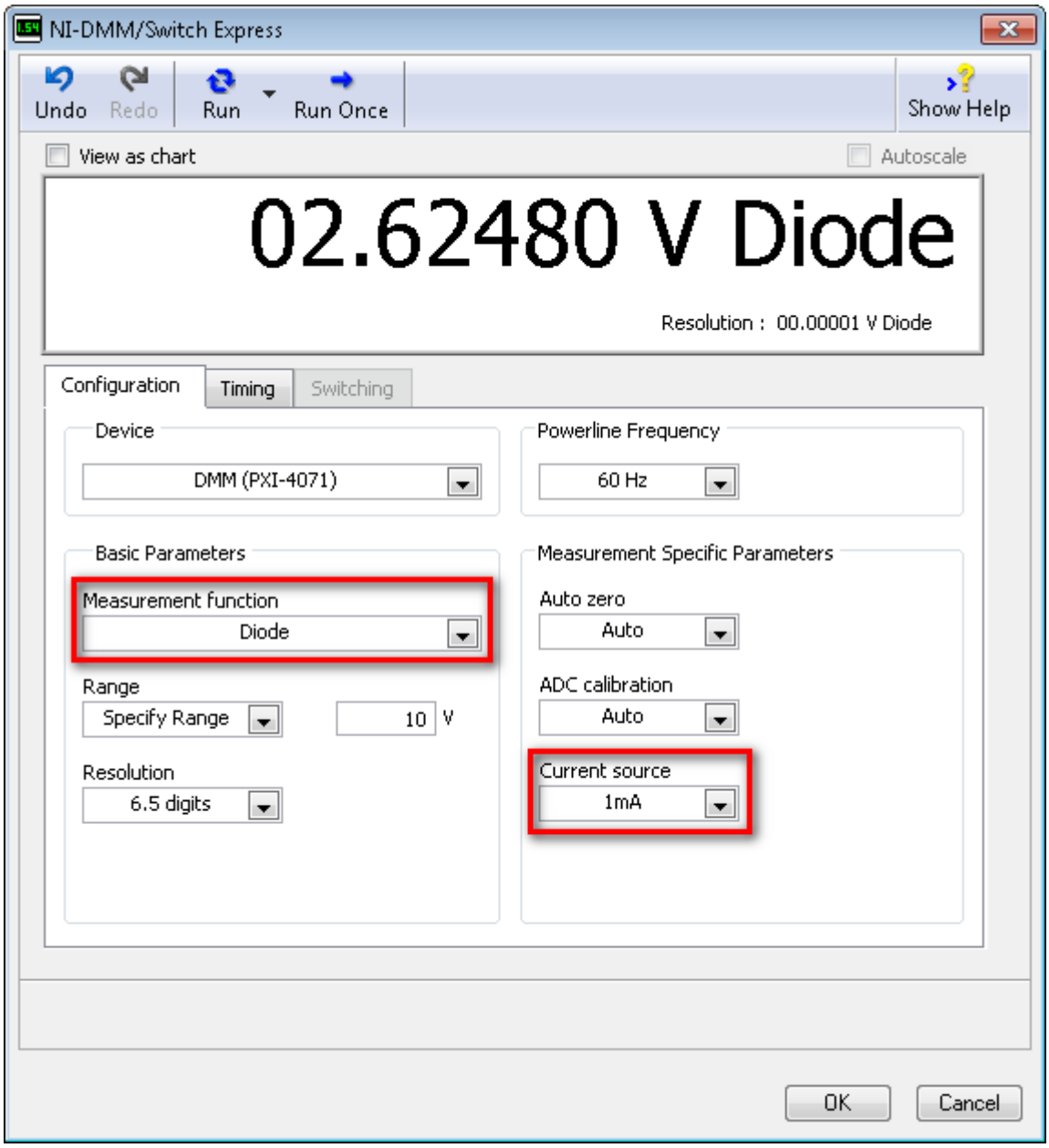


Figure 1-13. NI-DMM/Switch Express Dialog Box

3. Test the NI-DMM Express VI.
   - Select **Run Once**. If the LED is connected properly and is functional, the displayed value will be approximately **2.6 Volts**. Also notice that the blue LED on the UUT is illuminated.

- If you do not receive a result in this range or if the LED is not illuminated, verify that the switch configuration discussed in Figure 1-4 is correct.

2. Click the **OK** button to return to the Sequence Editor.
3. Define where to store the test result in TestStand.
   - In the *Value* field of the *signals out* terminal, type in Step.Result.Numeric (See Figure 1-14). This property will now contain the measured DMM value which will be compared to the limits defined in the step.

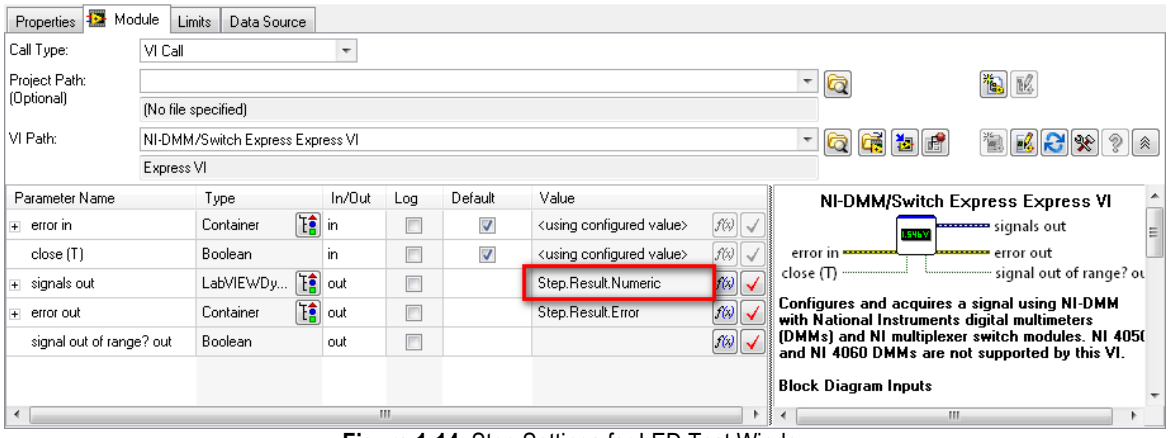


Figure 1-14. Step Settings for LED Test Window

4. Configure the step limits based on the expected measurement.
   - Click the **Limits** tab in the Step Settings for the LED Test.
   - Set the limits in the Edit Numeric Limit Test dialog box to the values shown in Figure 1-15. Ensure that Units is set to “volts”. In this case, the step will fail for measurements outside of 2.4 V and 2.9 V.

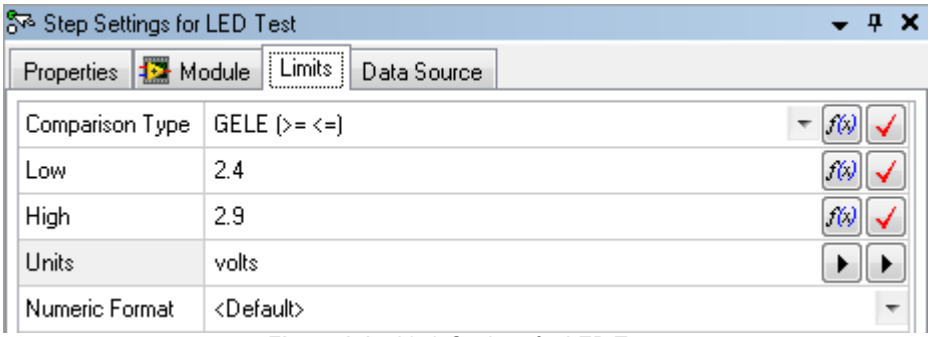


Figure 1-15. Limit Settings for LED Test

Part F: Create the BER Test Step

1. Create a new **Numeric Limit Test** step below the LED Test to execute the BER Test.
   - Rename the step **BER Test**.
2. Configure the step to call the BER test code.
   - Select the **Module** tab in **Step Settings** pane.
   - Click the **File Browse** button  on the right side of the **VI Path** field.

- Select the BERTest.vi file from the `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Test Code\BERT Test` directory and click **Open**.
- If prompted, select to use a relative path.

3. Configure the VI front panel to be displayed when the step executes by clicking the **Show Front Panel** button  .
4. Define where to store the test result in TestStand.
   - The BERTest VI has been preconfigured to pass results to TestStand.
   - For the **Bit Error Rate** parameter, specify the expression Step.Result.Numeric.
5. Configure the step limits to ensure that the Bit Error Rate is below the tolerance level of .01%.
   - Click the **Limits** tab in the Step Settings for BER Test.
   - Set the **Comparison Type** as **LT (<)**
   - Set the **Low** limit as 0.01.

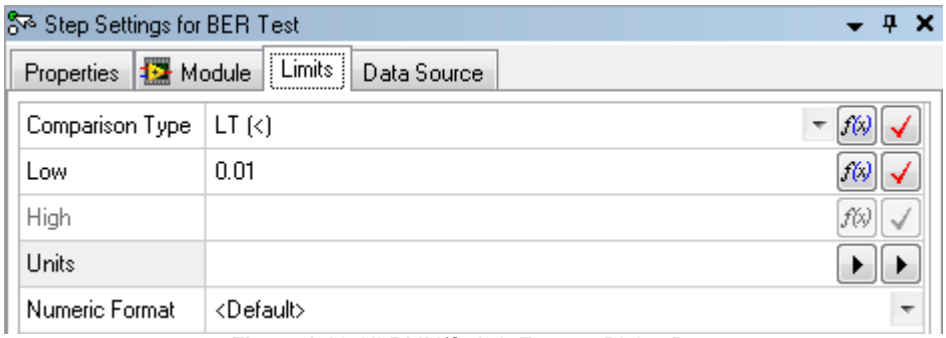


Figure 1-16. NI-DMM/Switch Express Dialog Box

6. Save the sequence as ***ATESequence.seq*** in the `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Exercises` directory.

Part G: Execute the Sequence

1. Now that the tests are added and the proper switches were manually connected in Exercise 1, you can execute your test sequence.
   - To execute the sequence once, select **Execute»Single Pass**.
   - If prompted, select **Yes** to save the sequence file.
   - As the sequence executes, observe the execution window to see each step as it executes, as shown in Figure 1-17.

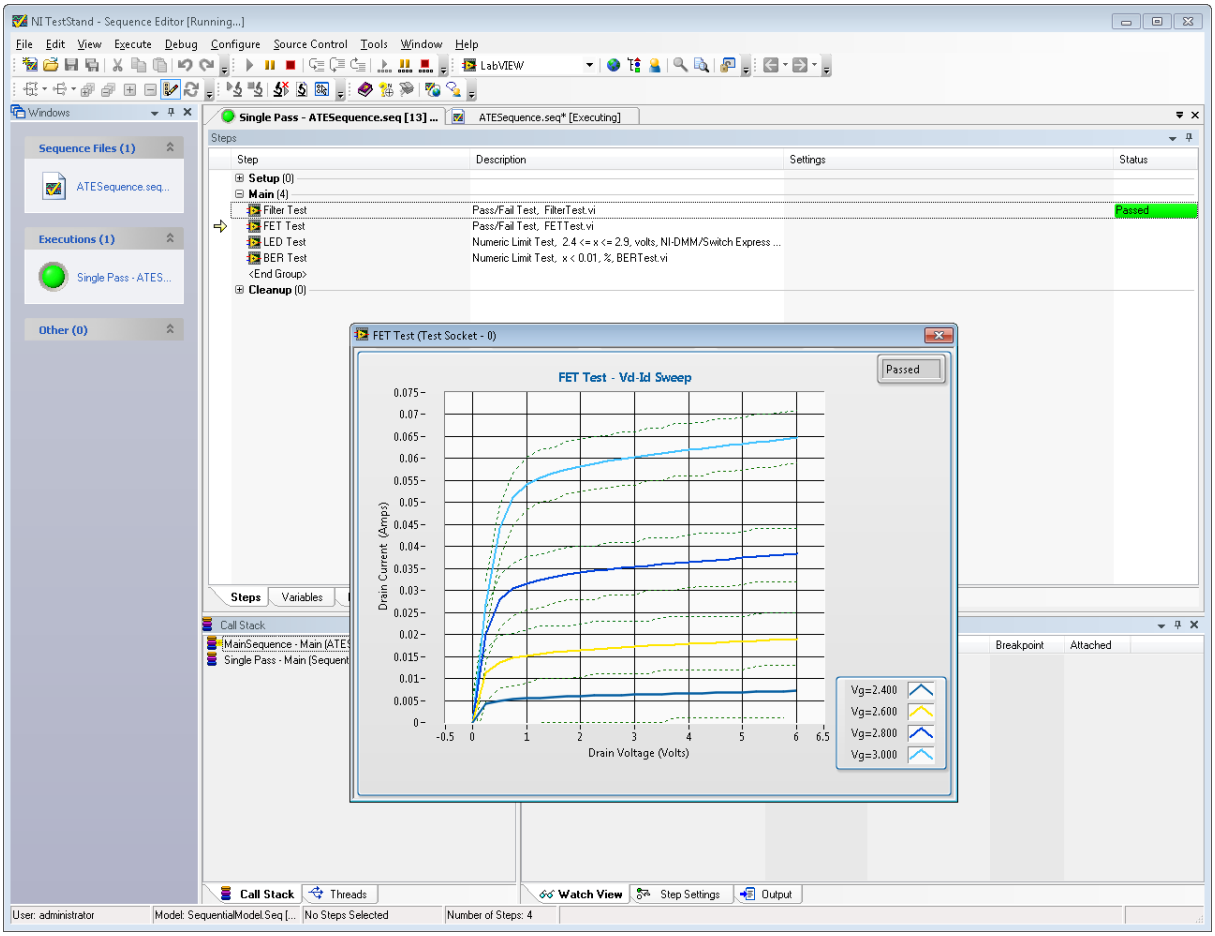


Figure 1-17. Sequence Editor Running

2. Once the execution completes, view the test results, as shown in Figure 1-18.

   Note: The UUT LED will remain on after the test is complete.

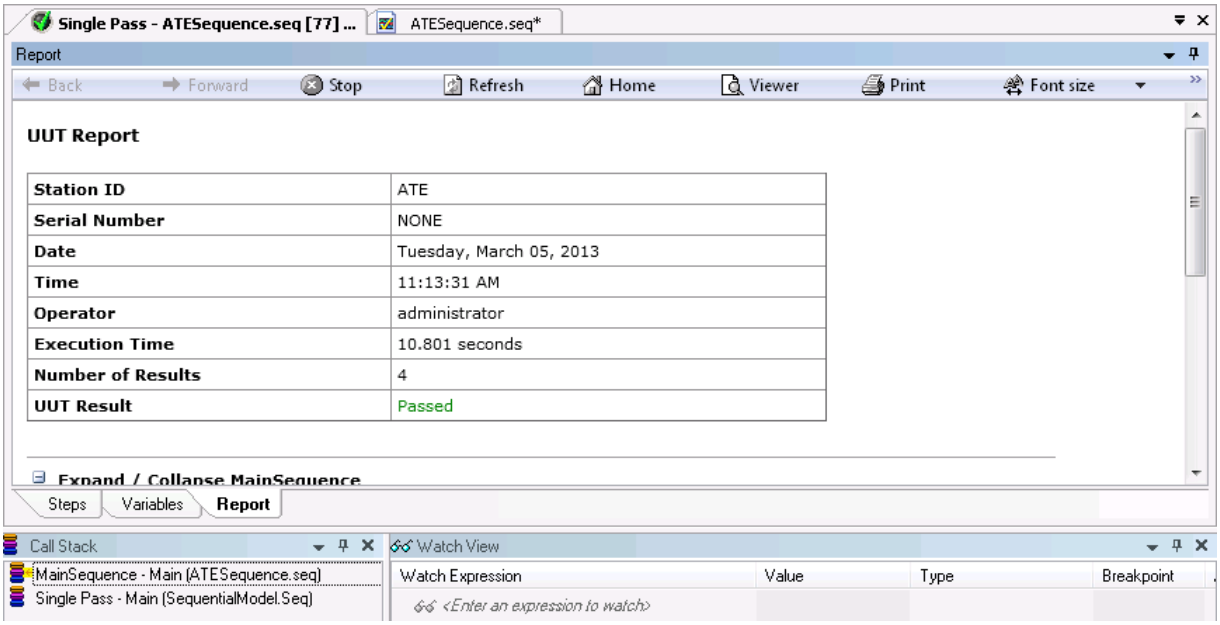


Figure 1-18. UUT Report

3. Test multiple failure cases to verify that the sequence correctly identifies a bad UUT.
   - Toggle the switch near the LED or FET on the UUT to force the corresponding test to fail. Alternatively, you can move the potentiometer dial away from the center position to force a failure on the filter, as indicated on the UUT itself. Lastly, the BER test can be failed by either changing the toggle switch (S4) or pressing the push button switch (S3).
   - Run the sequence again using **Execute»Restart**.
   - View the test results to see that the intended steps failed as expected.
   - Return all switches and knobs to the passing state on the UUT.

End of Exercise

Exercise 2: Integrating Switching into the Sequence Using NI Switch Executive

Goal

Up to this point, the ATE test sequence has used a single switch configuration to connect the UUT to the hardware. However, it is typically not cost effective to have dedicated hardware for each test point on the UUT. To simulate this issue, we will add the requirement that we also need to test the second LED on our UUT. Thus we now have two test points but only one DMM channel in our system.

To address this limitation, you will use the integrated switching feature of TestStand to dynamically change the switch configuration so that the correct LED is connected to the DMM when the test code is executed.

Description

Part A: Create a New Step to Test the Second (white) LED and Configure Switching for All Steps

1. Close the results of the previous exercise by pressing <**Ctrl+D**>
2. Create a copy of the LED test step.
   - Using the copy and paste functions, create a new copy of the LED test immediately following the original LED test.
   - Rename the copy **White LED Test**, and the original step **Blue LED Test**.

   Note: The only configuration change needed is the switch configuration. No further changes are needed, since the same test and DMM hardware will be used for both LED tests.

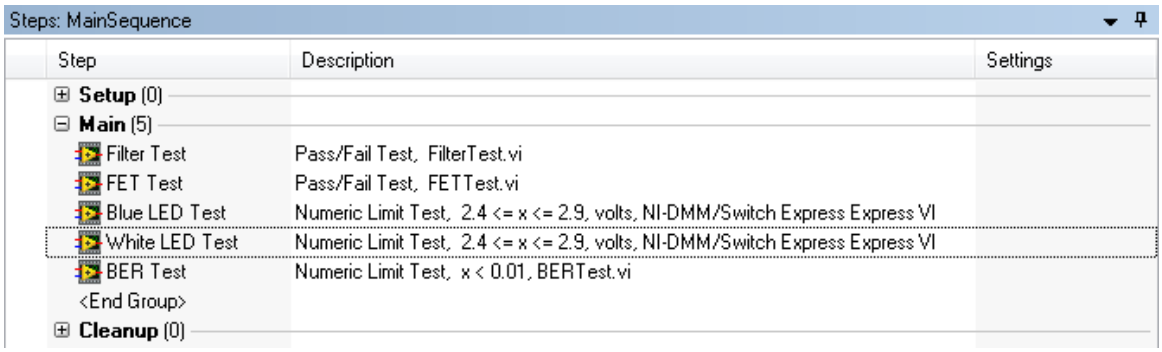


Figure 2-1. Creating the Blue LED Test

3. Configure the switching for the blue LED Test step (refer to Figure 2-2).
   - In the **Properties** tab of the Step Settings for the Blue LED Test, select the **Switching** section.
   - Select **Enable Switching**. Notice that the Blue LED Test step now has "Switch" listed in the **Settings** column of the steps pane to indicate this change from the default settings.
   - Verify that **"ATE"** is selected in the *Switch Executive Virtu*
   - *al Device* field.

     Note: This virtual device has been preconfigured with all the necessary routes and route groups in this exercise. To inspect the route groups, use the switch executive interface in Measurement & Automation Explorer that was discussed in Exercise 1.
   - In the **Route(s) to Connect** drop-down menu, select the "Test_LED_0_UUT0" switch configuration.

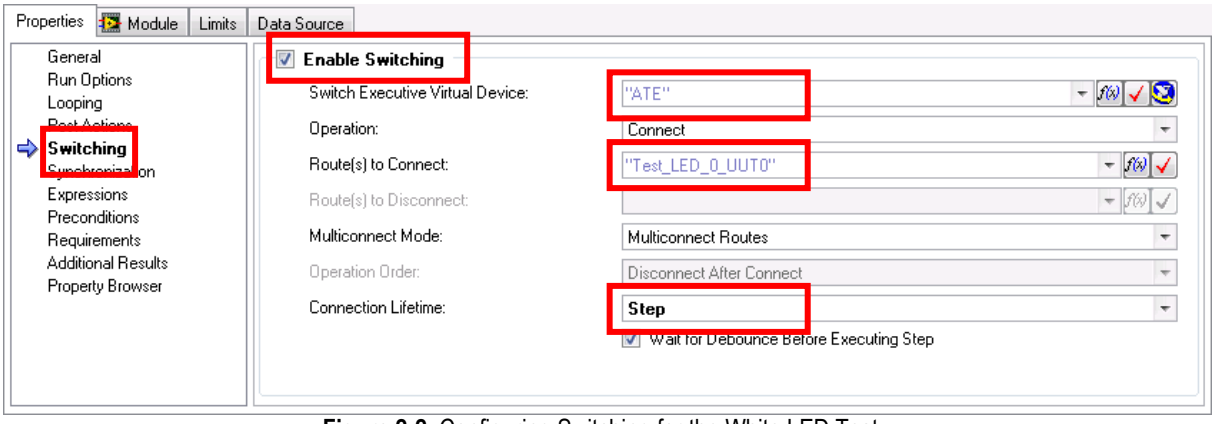


Figure 2-2. Configuring Switching for the White LED Test

- Repeat step 2 for the other tests, but using the values in the table below for the **Route(s) to Connect** field.

  Note: The BER Test step does not require a switch configuration because the hardware is directly connected to the test sockets.

Table 2-1. Route Configuration for Test Steps

Step	Route(s) to Connect Value
Filter Test	"Test_Filter_LP_UUT0"
FET Test	“Test_FET_UUT0"
White LED Test	"Test_LED_1_UUT0"

4. Test the sequence, and verify that all steps pass.
   - Select **Execute»Single Pass**.
   - View the test report. If any steps have failed, verify that the switch configuration is correct, and make sure the switch on the UUT is set such that the test will pass.

End of Exercise

Exercise 3: Performing Auto-Scheduled Parallel Testing

Goal

Now that the ATE test sequence can successfully test a single UUT, we now want to scale the system to be able to test up to four UUTs at once, using four independent test sockets. This change introduces a couple new challenges:

1. The test executive must be able to simultaneously manage four test sockets running in parallel. TestStand provides an out of the box solution to this issue: the parallel and batch process models. These models automatically handle the socket management.
2. Limited hardware resources must be effectively shared between up to four UUTs. Using the switching features of TestStand, you can quickly ensure that the correct UUT is connected to the correct hardware for each test socket. Additionally, TestStand provides an auto-scheduling feature to further optimize hardware usage.

In this exercise, you will use the batch process model, switching, and the auto-schedule feature to create an efficient parallel test system.

Description

Part A: Configure the Test Sequence to Execute in Parallel

1. Close the previous test report by pressing <Ctrl+D>
2. Change the process model to the batch process model.
   - Select **Configure»Station Options**.
   - In the **Model** tab of the Sequence file Properties dialog, change the Station Model setting to **BatchModel.seq** as shown in Figure 3-1.
   - Click **OK**. Observe that the status bar (at the bottom of the sequence editor window) indicates that the batch process model is now active.

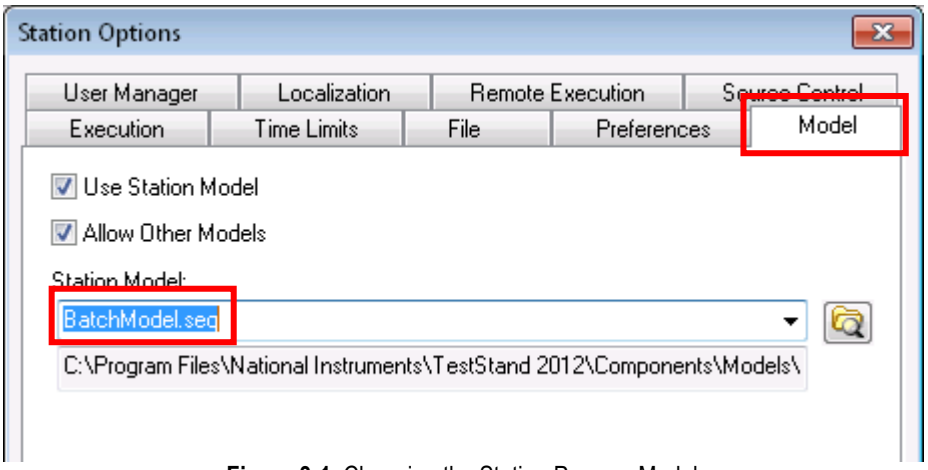


Figure 3-1. Changing the Station Process Model

Note: This setting changes the process model used for all sequence files on the machine. You can change the model for a particular sequence in the **Advanced** tab of the **Sequence File Properties** dialog by selecting **Edit»Sequence File Properties…**.

3. Specify the number of test sockets in your fixture.
   - Select **Configure»Model Options**. The Model options dialog contains settings pertaining to the currently selected process model.

- Set the Number of Test Sockets to **4**.
- Click **OK** to dismiss the model options dialog.

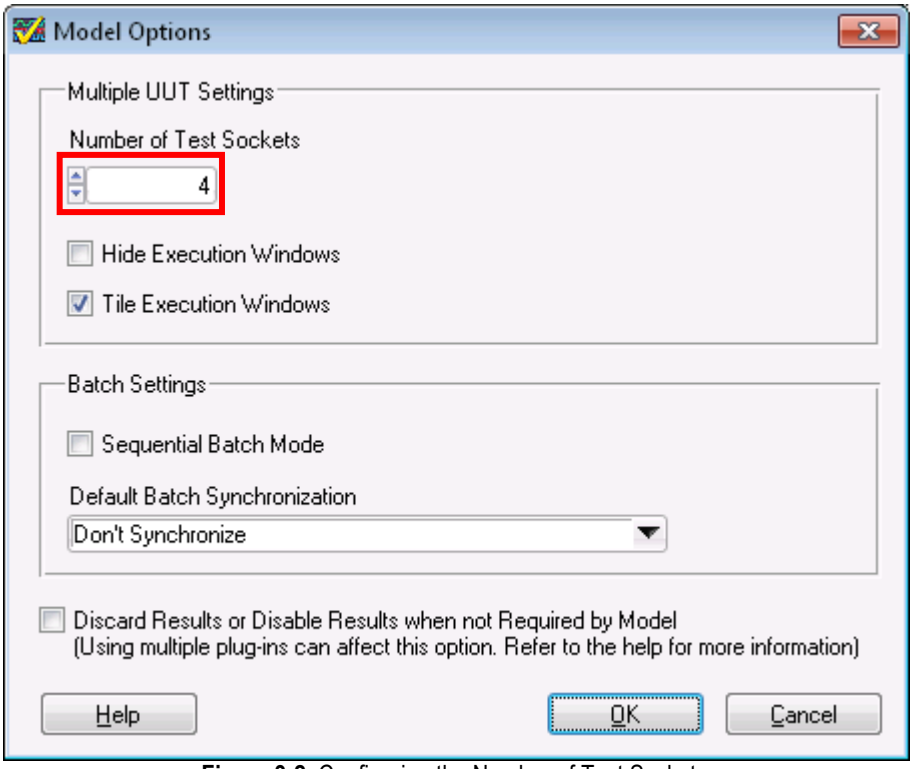


Figure 3-2. Configuring the Number of Test Sockets

4. Configure the titles panels to display socket information.
   - Select the Filter Test step in the steps pane.
   - In the **Module** tab, uncheck the **Default** setting for the Socket Number parameter as per Figure 3-3
   - In the value field, enter the property *RunState.TestSockets.MyIndex*.
   - Repeat this process for the FET Test and the BER test steps (the LED tests do not display a dialog.

     Note: *RunState.TestSockets.MyIndex* is a default TestStand property that will dynamically populate with the appropriate socket number. Each of the LabVIEW VIs have already been configured to take the socket number as an input which is then programmatically written to the title of the VI. Thus, when TestStand's batch process model creates the 4 instances (sockets) of this sequence, each VI will be called 4 times and the title of the VI will indicate which test socket is currently being tested.

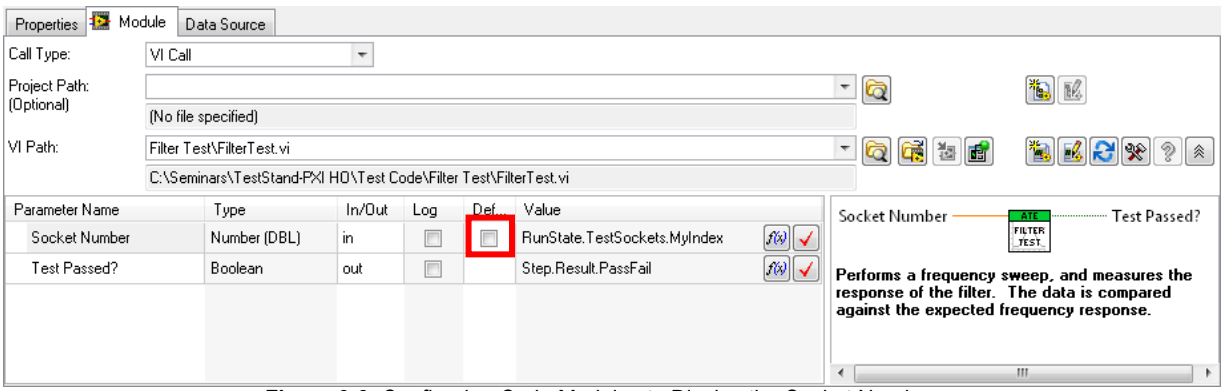


Figure 3-3. Configuring Code Modules to Display the Socket Number

Part B: Update the Switching Configuration for the Test Steps to Handle Batch Execution

1. Change the Route expression for the **Filter Test** to dynamically populate with the current test socket.
   - In the **Properties** tab of the Step Settings tab of Filter Test, select the **Switching** section.
   - Edit the **Route(s) to Connect** field, and replace the current expression with the following:

     "Test_Filter_LP_UUT" + str(RunState.TestSockets.MyIndex)

     Additional Information
     This expression is used because the switch route now depends on the socket of the UUT being tested. The possible values of this expression have been pre-defined in the switch virtual device: FilterTest_UUT0, FilterTest_UUT1, FilterTest_UUT2, and FilterTest_UUT3.

     The *Str* function converts the numeric index value to a string, which is then concatenated to the Route String.

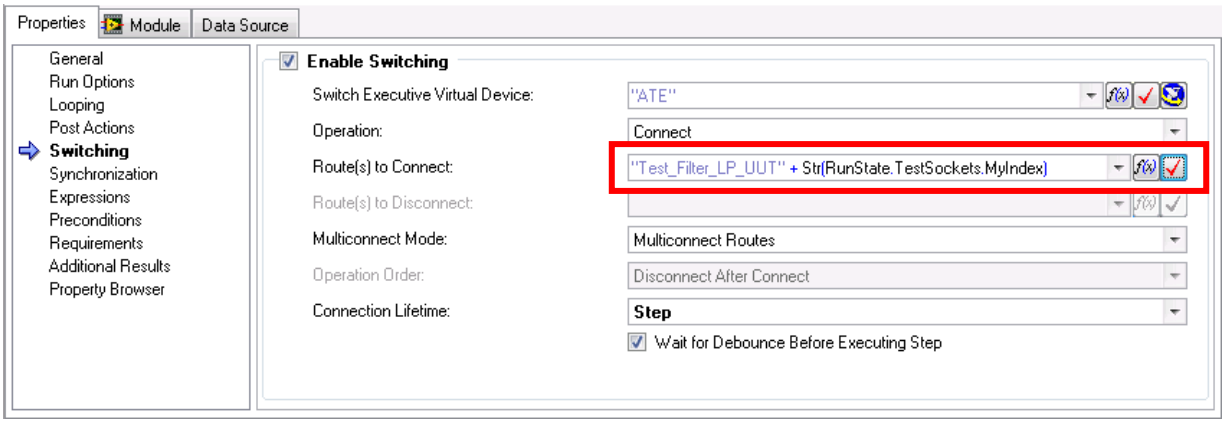


Figure 3-4. Updating the Switch Configuration to Support Multiple Test Sockets

   - Click the **Check Expression for Errors** button to confirm that the expression syntax is correct.

2. Repeat the previous step for the remaining test steps, using the expressions in the table below for the **Route(s) to Connect** field:

Table 3-1. Route Configuration for Test Steps

Step	Route(s) to Connect Value
FET Test	"Test_FET_UUT" + Str(RunState.TestSockets.MyIndex)
Blue LED Test	"Test_LED_0_UUT" + Str(RunState.TestSockets.MyIndex)
White LED Test	"Test_LED_1_UUT" + Str(RunState.TestSockets.MyIndex)

Part C: Implement Auto-Scheduling

Additional Information

In this section, you will add the Auto Schedule step type to the MainSequence. The auto schedule step allows you to configure multiple sections of your sequence that can be executed in any order. For example, if one test socket is using the DMM to measure the LED, a second socket will run the FET test, and run the LED test once the DMM is available. TestStand automatically determines the order at runtime to maximize hardware utilization and achieve the fastest testing time. The auto schedule step also ensures that each section is locked, so that only one socket can run the code at any given time.

This prevents resource conflicts, where multiple sockets are attempting to use the same hardware simultaneously.

1. Create the necessary **Auto Schedule** and **Use Auto Scheduled resource** steps by dragging them from the Insertion Palette
   - Insert an Auto Schedule step (located in the Synchronization folder of the Insertion Palette) before the *FilterTest* step. Notice that placing the Auto Schedule step also automatically creates a **Use Auto Scheduled Resource** step, as well as two **End** steps. Each end step forms a *block* of steps, which is denoted by indenting steps within the block.

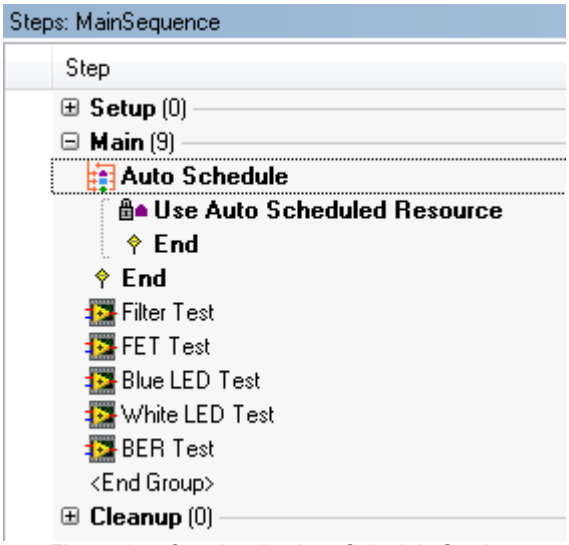


Figure 3-5. Creating the Auto Schedule Section

- Within the Auto Schedule block, create three additional **Use Auto Scheduled Resource** steps. Do not copy and paste the existing step, since this method will not create the needed **End** steps.

  Note: You will need a **Use Auto Scheduled Resource** block for each resource you are managing. Because the sequence is executing tests with four independent resources, we will need three more Auto Scheduled Resource blocks in addition to the one already created.

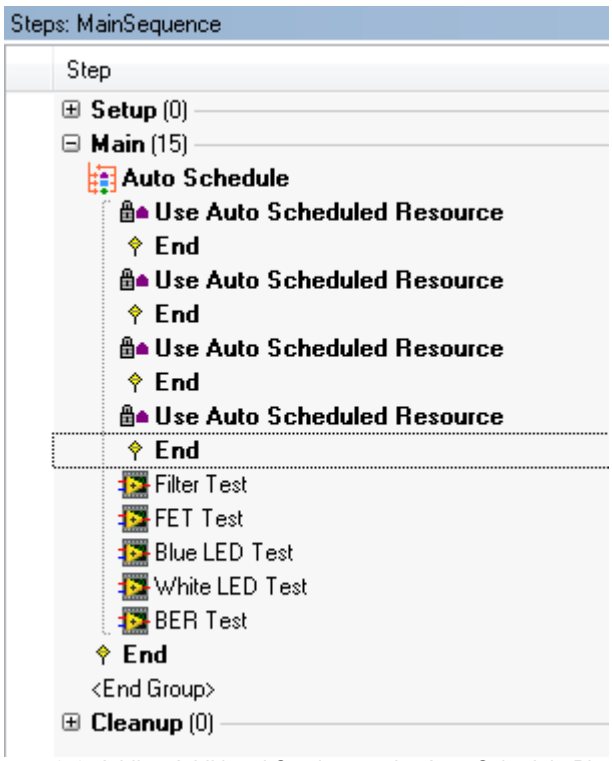


Figure 3-6. Adding Additional Sections to the Auto Schedule Block

2. Move the test steps into the **Use Auto Scheduled Resource** step blocks.
   - Drag the Filter test step into the first **Use Auto Scheduled Resource** block.
   - Drag the FET test step into the second **Use Auto Scheduled Resource** block.
   - Drag both LED test steps into the third **Use Auto Scheduled Resource** block.
   - Drag the BER test step into the fourth **Use Auto Scheduled Resource** block.

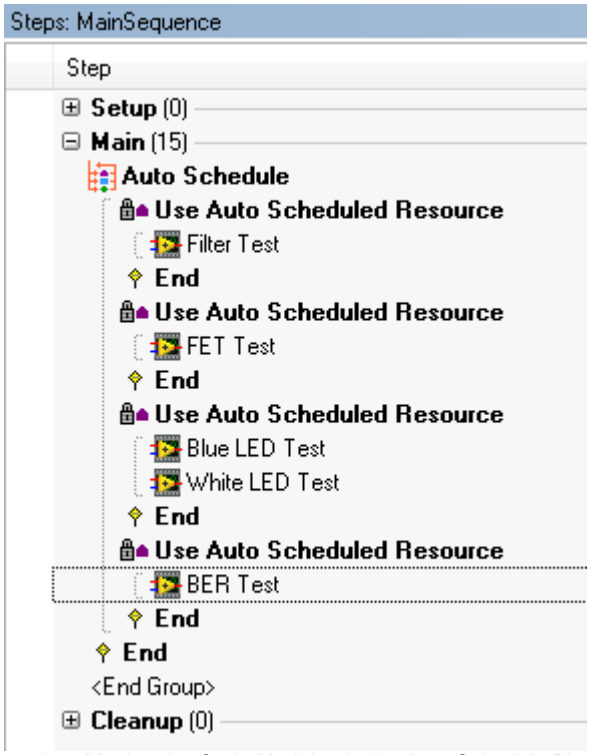


Figure 3-7. Moving the Code Modules in the Auto Schedule Blocks

3. Configure the resources used by each **Use Auto Scheduled Resource** steps.
   - Select the first **Use Auto Scheduled Resource** step. This step defines the block for the Filter Test.
   - In the **Auto Scheduled Resource Settings** tab, Enter the following expression for the Resource Lock Alternatives field:

     {"Scope", "Arb"}

     Note: This expression is an array of strings, and reserves two hardware resources when a test socket executes this **Use Auto Scheduled Resource** step. If another test socket encounters a **Use Auto Scheduled Resource** step which specifies one or both of these resource names, that socket must wait to execute until the first test socket has completed execution of the **Use Auto Scheduled Resource** step block.

Figure 3-8. Specifying the Resources Used in the First Auto Schedule Section

4. Configure the resources used in the remaining **Use Auto Scheduled Resource** steps.
   - In this test, five hardware resources are used for our four tests: the Scope, Waveform Generator (Arb), DMM, SMU, and HSDIO.
   - Use the table below to specify **Resource Lock Alternatives** for the remaining **Use Auto Scheduled Resource** steps.

Table 3-2. Resource Lock Alternatives for Each Use Auto Scheduled Resource Step.

Auto-scheduled section	Resource Lock Alternatives Value
FET Test	"SMU"
LED Tests	"DMM"
BER Test	"HSDIO"

5. In order to prevent unwanted information in the test report, disable reporting for all auto-schedule steps.
   - Select all **Auto Schedule** and **End** steps while holding the <Ctrl> key.

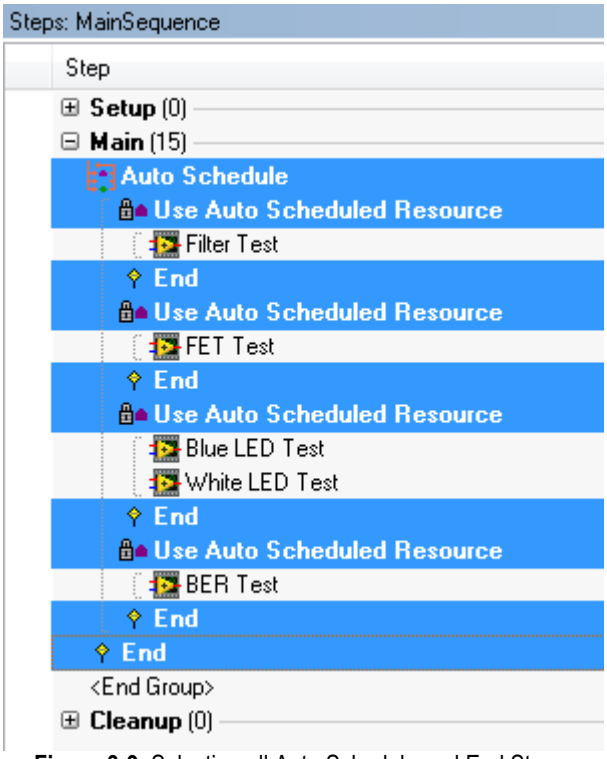


Figure 3-9. Selecting all Auto Schedule and End Steps

- Navigate to the **Run Options** section in the **Properties** tab of the **Step Settings** pane.
- Change the **Result Recording Option** to **Disabled**.

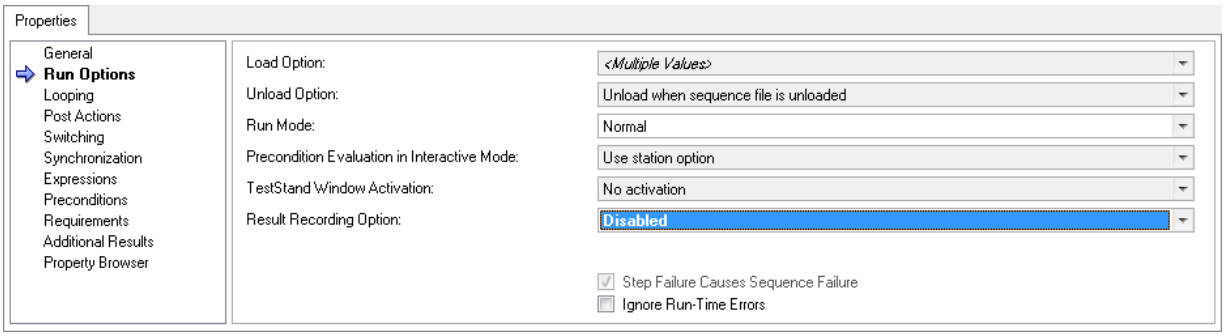


Figure 3-10. Disable Result Logging for Selected Steps

Part D: Test the Sequence

6. Verify that the four sockets are created and executing properly.
   - Execute the sequence using the Test UUTs Entry point (**Execute»Test UUTs**).
   - Verify that four execution tabs appear; one for each test socket.
   - You will be prompted to enter a batch serial number and individual UUT serial numbers in the UUT Information dialog. Enter any serial numbers and click **Go**.
   - The Execution pane should now display all four test socket executions simultaneously.
   - Click **Next Batch**, then **Stop** to end the test.
   - If any failures occurred, view the test report for each socket. Ensure that the switch configuration is correct for any steps that are failing.
   - Close all execution windows.

Hint: You can dismiss all completed execution tabs by pressing the <Ctrl-D> shortcut.

7. Use TestStand's Resource Profiler tool to visualize the hardware utilization during the test.
   - Select **Tools»Profile Resource Usage**.
   - Execute the sequence using the **Execute»Single Pass** Entry point.
   - As the sequence executes, the Resource Usage Profiler will update with information on what steps are being executed in each test socket.
   - Once the execution completes, notice that the tests were not executed in order in all threads. The auto schedule step optimized the sequence to use the hardware as efficiently as possible.

End of Exercise

Exercise 4: Execute Project with Customized TestStand User Interface

Goal

Now that the sequence is configured to run in parallel, we can now deploy it to the test floor to be used in production. When using the sequence, operators need a simplified interface to prevent confusion and accidental changes to the test sequences or systems. To accomplish this, you can create a customized user interface to run TestStand sequences.

In this exercise, you will execute the test sequence in a custom user interface.

Description

1. Launch the **ATE TestStand User Interface**, located here:
   `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Test Code\User Interface\ATE TestStand User Interface.exe`

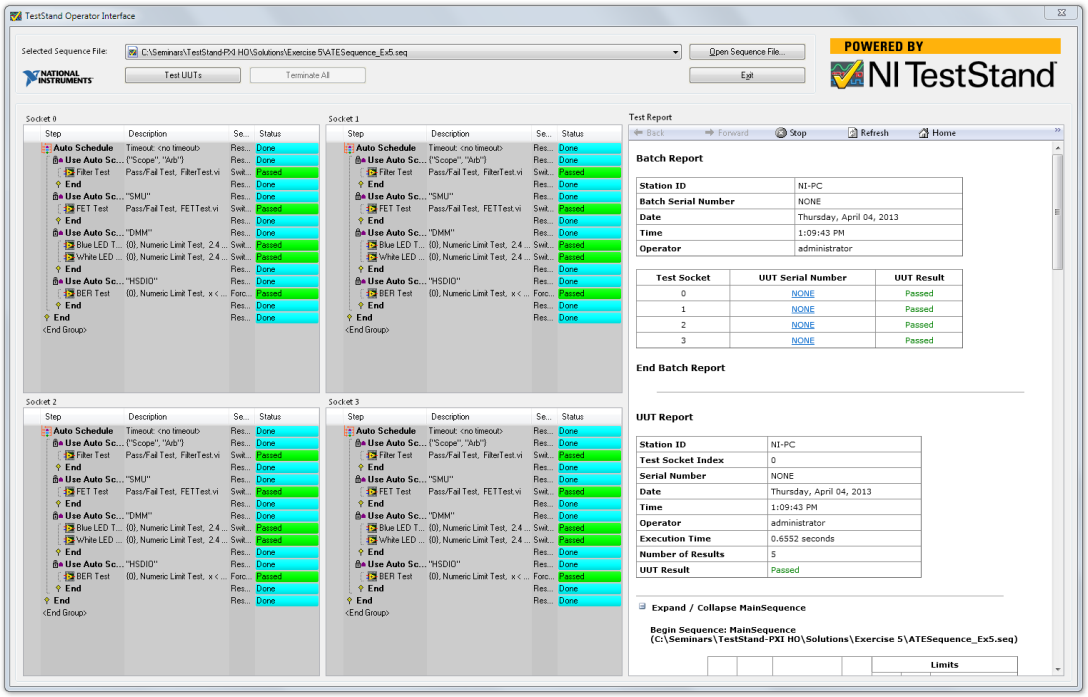


Figure 4-1. The Custom ATE TestStand User Interface

2. Inspect the user interface. Note that much of the functionality of the TestStand sequence editor is not available. This will prevent confusion and accidental system changes by the operators of the test.
3. Click the **Open Sequence File** button to open a file. Select the ATESequence.seq file that you have created in the last five exercises located here:
   `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Exercises\ATESequence.seq`
4. Click the **Test UUTs** button and click **Go** in the UUT Information window to run the sequence using the TestUUTs execution entry point. The process model setting from the sequence editor, such as the number of sockets, also applies to the user interface.
5. View the execution of each socket as the test runs. When the test completes, the batch report appears. Verify that the results are consistent with the switches on the UUT.
6. Exit the user interface.

End of Exercise

Exercise 5: (Optional) Overriding Process Model Callbacks

Goal

In this exercise, you will modify the default behavior of the process model in order to use the test fixture LEDs to show the status of each test socket. You will accomplish this by overriding a callback defined in the model, allowing the functionality to be implemented without any changes to the model file itself. You will override the following callbacks:

- PreUUT – Illuminate the yellow LED to indicate that testing is in progress.
- PostUUT – Illuminate the red or green LED to indicate that the test passed or failed.
- PostUUTLoop – Turn off all LEDs once testing is complete.

The completed sequence file is shown in Figure 5-1 below.

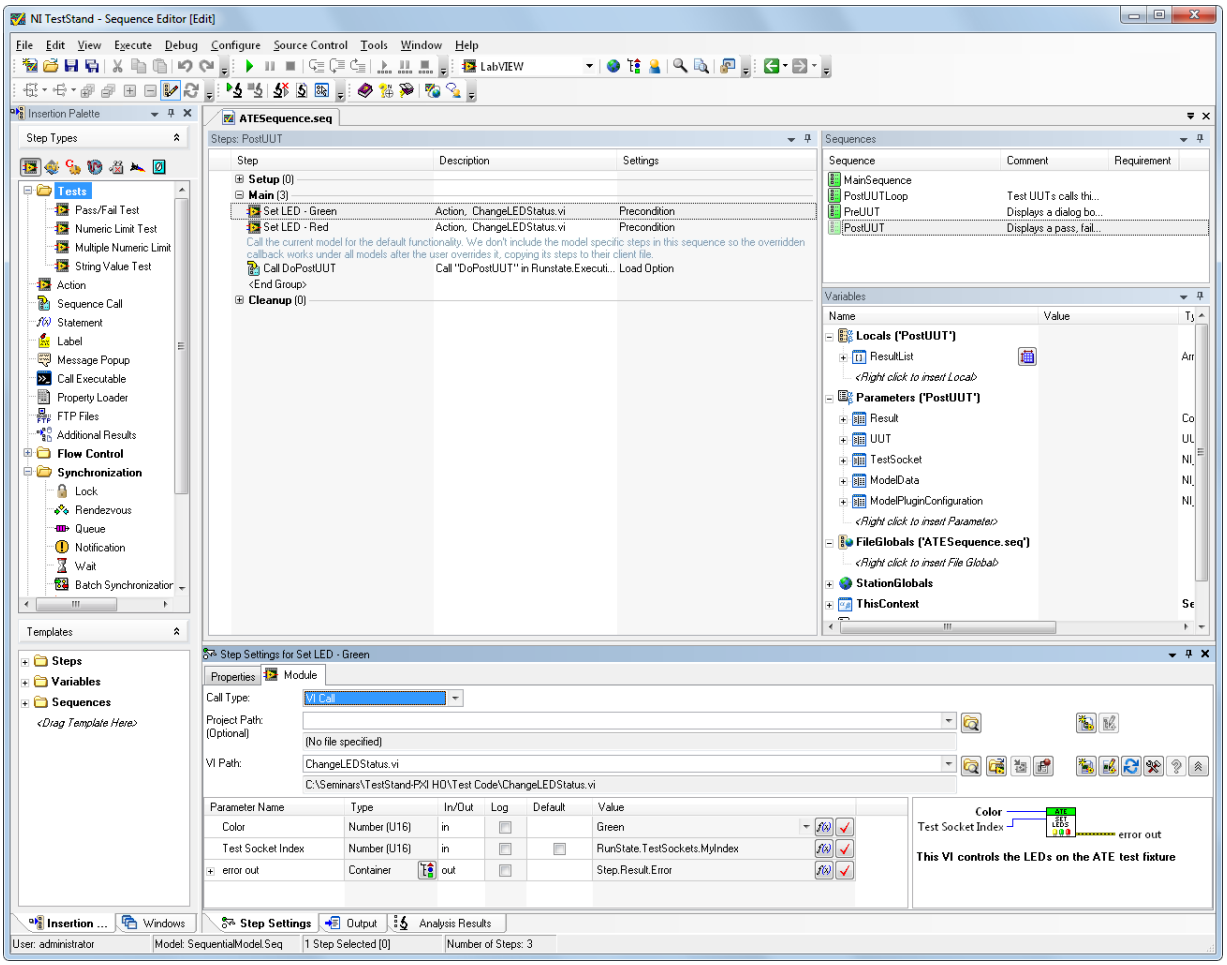


Figure 5-1. Completed Exercise

Description

Part A: Override the Model Callbacks

1. Open the ATESequence.seq file you created in the previous exercises.
2. Override the PreUUT callback.
   - Go to **Edit»Sequence File Callbacks** to open the sequence callbacks dialog, as shown in Figure 5-2.

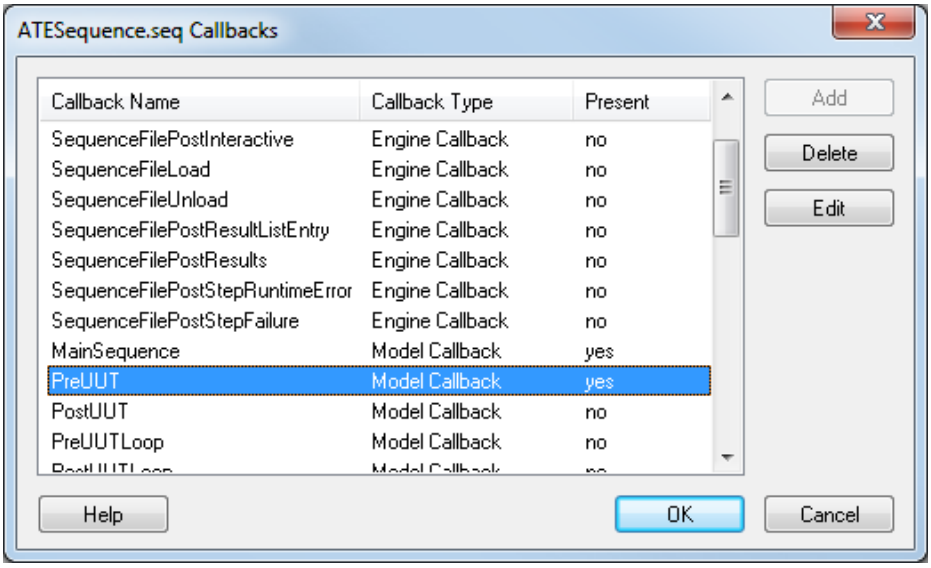


Figure 5-2. Sequence File Callbacks Dialog

- Select the PreUUT Callback, and click **Add**. Note that the text in the Present column changes to **Yes**.
- Click **OK** to dismiss the Sequence File Callbacks dialog.

Note: Notice that your sequence file now contains an additional sequence, PreUUT. The sequence icon is green, indicating that it is overriding a callback defined in the process model. Also notice that the sequence already contains a step. This step represents the default behavior of the PreUUT callback, which is displaying the serial number dialog shown in Figure 5-3.

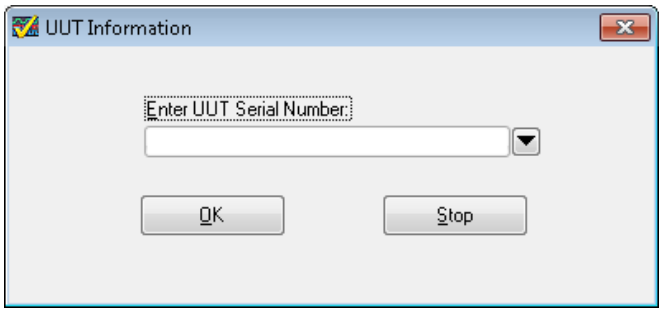


Figure 5-3. UUT Serial Number Dialog Implemented in the PreUUT Callback

Additional Information

Now that the PreUUT callback is defined in the client sequence, the process model will call this sequence instead of the PreUUT sequence defined in the process model. This allows you to define custom process model behavior unique to your test sequence.

3. Add a new step to the PreUUT callback sequence to turn on the yellow LED when the UUT begins testing.
   - Create a new Action step (drag and drop from insertion palette) after the DoPreUUT step already in the sequence and rename it **Set LED – Yellow**. Do not remove the existing Call DoPreUUT step in the sequence, since we still want the serial number dialog to appear before each UUT.

- In the Module tab for the step, select the ChangeLEDStatus.vi, located in the `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Test Code` directory.
- In the Module tab for the new step, uncheck the **Default** check box for TestSocket Index and configure the parameter values as shown in Table 5-1 below

Table 5-1. Parameter Configuration for the LED Test Step

Parameter	Value
Color	"Yellow"
Test Socket	*RunState.TestSockets.MyIndex*

- Verify that the sequence matches Figure 5-4.

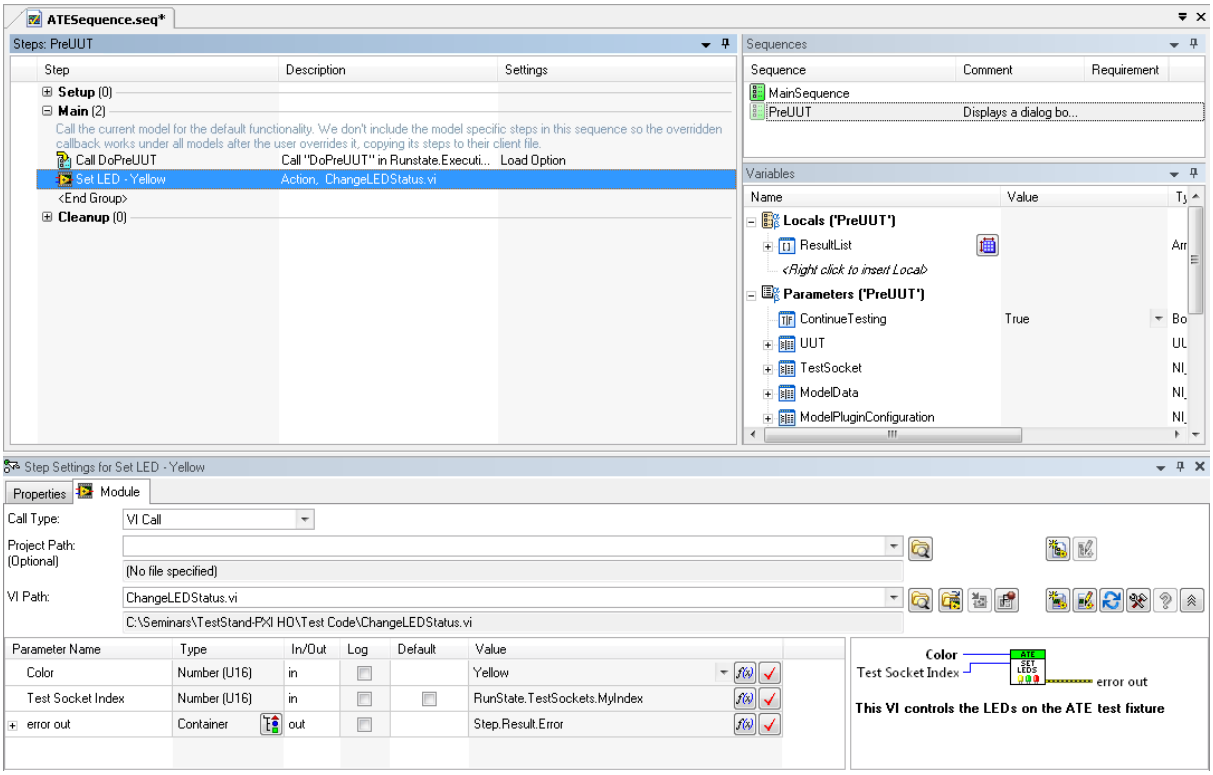


Figure 5-4. The Completed PreUUT Callback Sequence

4. Override the PostUUT callback.
   - Go to **Edit»Sequence File Callbacks** to open the sequence callbacks dialog.
   - Select the PostUUT Callback, and click **Add**.
   - Click **OK** to dismiss the Sequence File Callbacks dialog.
5. Add a new step to the PostUUT callback sequence to turn on the green LED.
   - Remove the **Call DoPostUUT** step because the status banners are no longer necessary now that we will be showing the pass/fail status with LEDs.

     Note: The default behavior of the PostUUT callback is to display a status banner indicating the test result, shown in Figure 5.5. This banner will no longer be displayed for your sequence, since the process model will now execute the callback sequence you created.

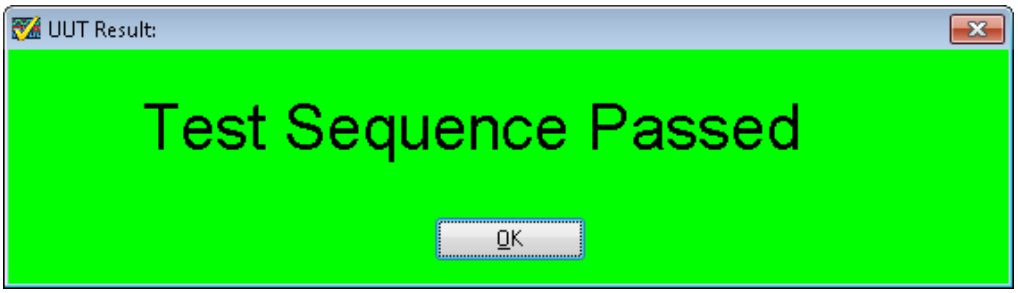


Figure 5.5. The UUT Status Banner, Implemented in the Default PostUUT Callback

   - Create a new Action step in the sequence and name it **Set LED – Green**.
   - In the Module tab for the step, select the ChangeLEDStatus.vi, located in the `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Test Code` directory.
   - In the Module tab for the new step, configure the parameter values as shown in the table below including de-selecting the **Default** check box in the Parameters section of the Module Tab.

Table 5-2. Parameter Configuration for the LED Test Step

Parameter	Value
Color	"Green"
Test Socket	*RunState.TestSockets.MyIndex*

6. Create a precondition so that the step executes only if the test passes.
   - With the step selected, click the properties tab, then the Preconditions section.
   - Enter the following expression in the Precondition expression field, as shown in Figure 5-6: Parameters.Result.Status == "Passed".

     Note: This property is not created until run-time and thus it can't be found directly using the Expression Browser and pressing the **Check Expression for Errors** button will say that it is unknown. For this reason, you must type property name manually.

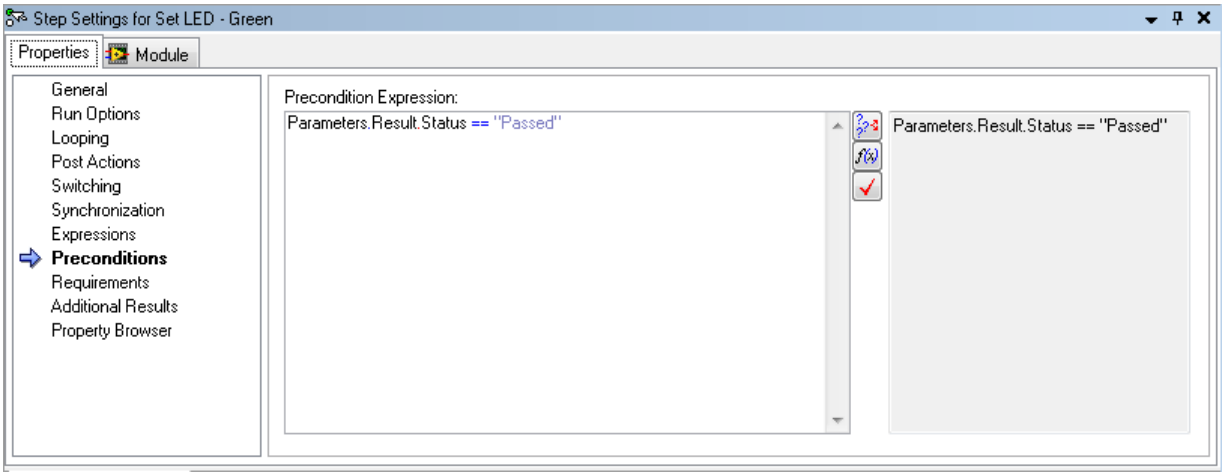


Figure 5-6. Configuring the Precondition Expression

Additional Information

If you specify a precondition for a step, the step will only execute if the expression you specify evaluates to true. In this case, the green LED will only be illuminated if the test passes.

7. Add a new step the PostUUT callback sequence to turn on the red LED if the test fails.
   - Create a copy of the **Set LED – Green** step in the PostUUT sequence using the copy and paste functions.
   - In the module tab for the new copy, change the value passed to the Color parameter to "Red".
   - Click the Properties tab, then the Preconditions section, and replace the precondition expression with the following expression:

     Parameters.Result.Status == "Failed".

   - Rename the step **Set LED – Red**.
8. Verify that your PostUUT sequence matches 5-7.

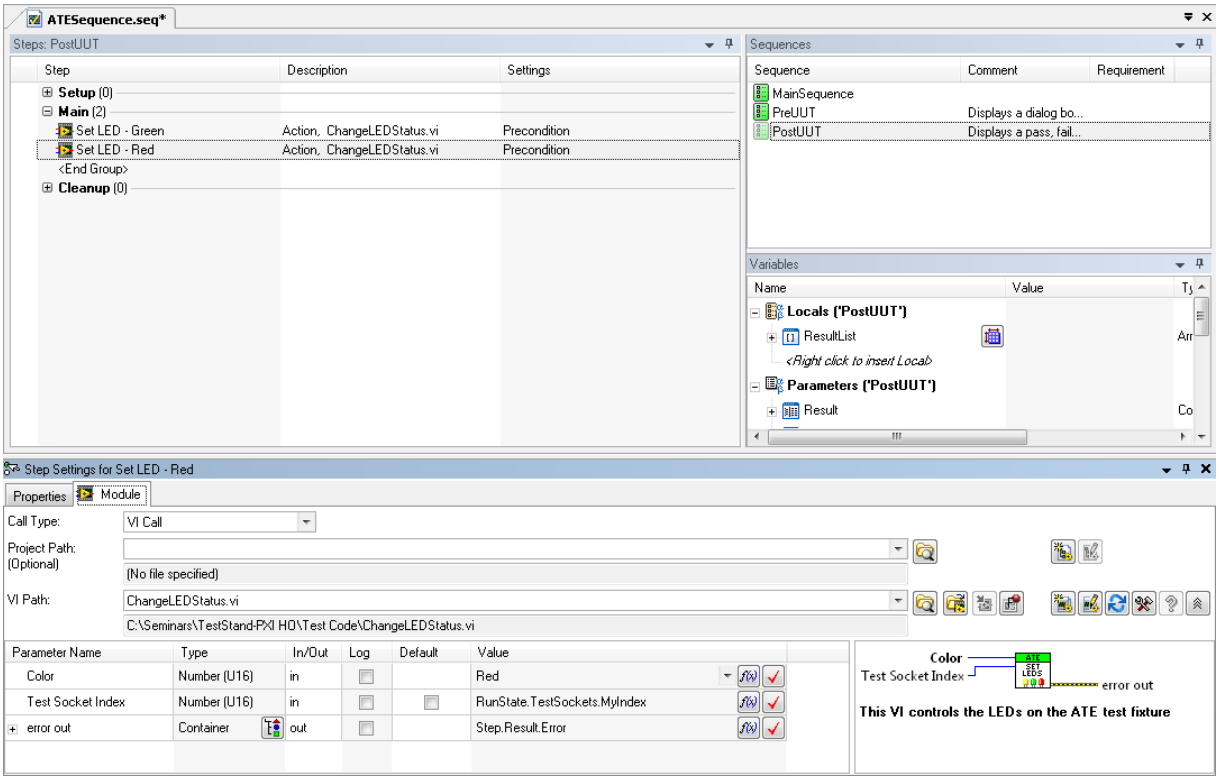


Figure 5-7. The completed PostUUT Callback Sequence

9. Override the PostUUTLoop callback.
   - Go to **Edit»Sequence File Callbacks** to open the sequence callbacks dialog.
   - Select the PostUUTLoop Callback, and click **Add**.
   - Click **OK** to dismiss the Sequence File Callbacks dialog.

   Note: The PostUUTLoop callback has no default behavior. It exists to allow users to implement their own functionality when testing is complete.

10. Add a new step the PostUUTLoop callback sequence to turn off all LEDs when all tests are done
    - Create a new Action step in the sequence and name it **Set LED – All Off.**
    - In the Module tab for the step, select the ChangeLEDStatus.vi, located in the `C:\Seminars\TestStand-PXI HO\TestStand-PXI HO\Test Code` directory.

In the Module tab for the new step, configure the parameter values as shown in Table 5-3.

Table 5-3. Parameter Configuration for the LED Test Step

Parameter	Value
Color	"All LEDs Off"
Test Socket	*RunState.TestSockets.MyIndex*

11. Verify that your PostUUTLoop sequence matches Figure 5-8.

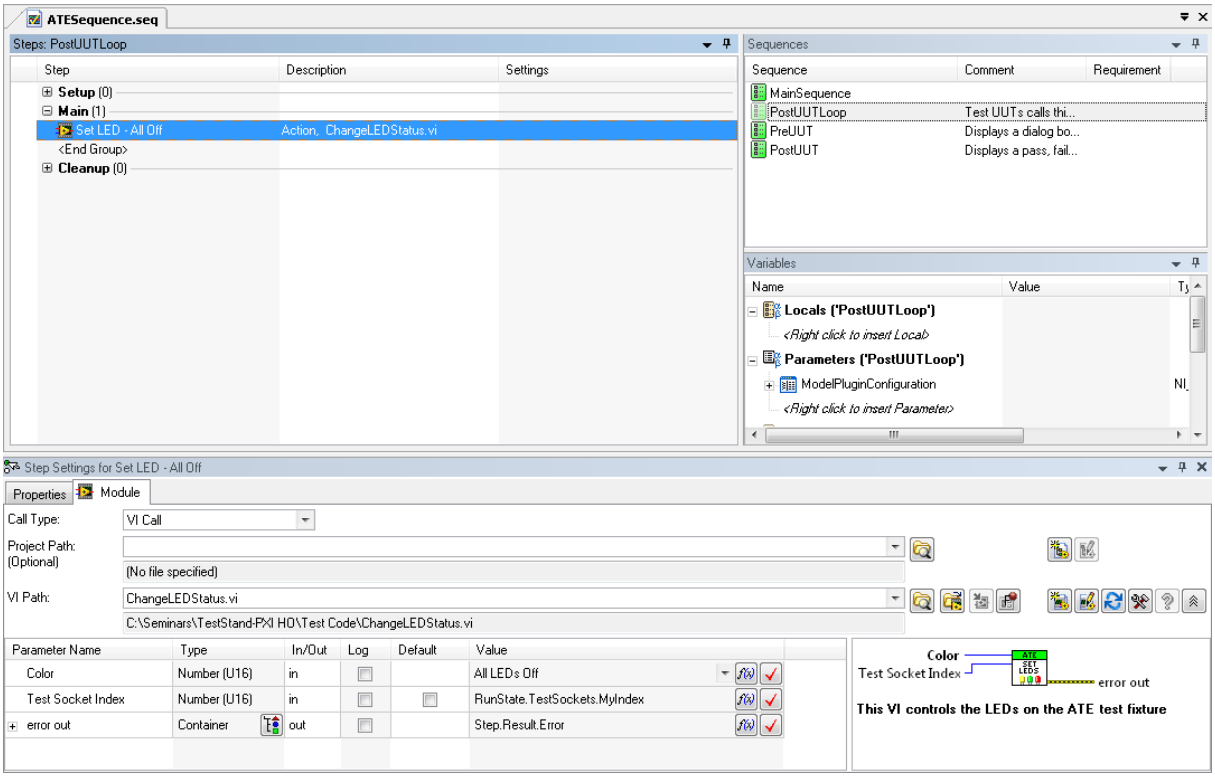


Figure 5-8. The Completed PostUUTLoop Callback Sequence

Part B: Testing

1. Test the sequence in the passing case.
   - Run the sequence using the Test UUTs Entry point (**Execute»Test UUTs**).
   - Enter UUT and Batch serial numbers when prompted. After dismissing the dialog, verify that the yellow LED illuminates on all test sockets.
   - Once the test completes, verify that the green LED is illuminated as each socket completes.
   - When prompted to enter another set of serial numbers, click **Next Batch**, then **Stop** to end the test.
   - Verify that all LEDs are now off.
2. Test the sequence again, forcing the sequence to fail.
   - Toggle the LED switch on any of the UUTs to create a failure in the hardware.
   - Run the sequence using the Test UUTs Entry point **(Execute»Test UUTs)**.
   - Once the test completes, verify that the red LED is illuminated for failing UUTs, indicating a failure.
3. Restore the UUT to the passing state by reverting the change you made for the previous step.
4. Save the sequence file.

End of Exercise