PROTOCOL INSIGHT

Falcon Series - Stimulus Test Editor User Manual

Contact Protocol Insight at:

sales@protocolinsight.com
support@protocolinsight.com
www.protocolinsight.com

Contents

The Protocol Insight® Stimulus Test Editor™ allows users to create UniPro and UFS stimulus files with the same tool used for creating Advanced Trigger and Trace Validation™ analysis tests.

Protocol Insight's new Microsoft Visual Studio based Test Editor tool replaces the original UniPro and UFS Stimulus Builders.

Test Editor is available for the Falcon G350/450 Exerciser/Analyzer series instruments only, it is not available for the Falcon G300/G400 series or the Raptor M300/M400 series.

General Operation

Stimulus Tests are created and edited using the Test Editor, which can be accessed from the Falcon application software from **Tools →Editor →Open Editor**.

For Test Editor license verification, the Falcon application software must be connected to the Falcon instrument. Launch the Falcon application software and click the Connect button on the Toolbar to connect to the instrument.

Once created, Stimulus tests are saved in the \CustomTests directory and are automatically added to the list of available Tests in the Configuration window. If the Stimulus test isn't listed in the Test list choosing **File→Refresh Stimulus Test List** will update the Test list.

Sample Stimulus tests are provided in the \CustomTests\Samples directory.

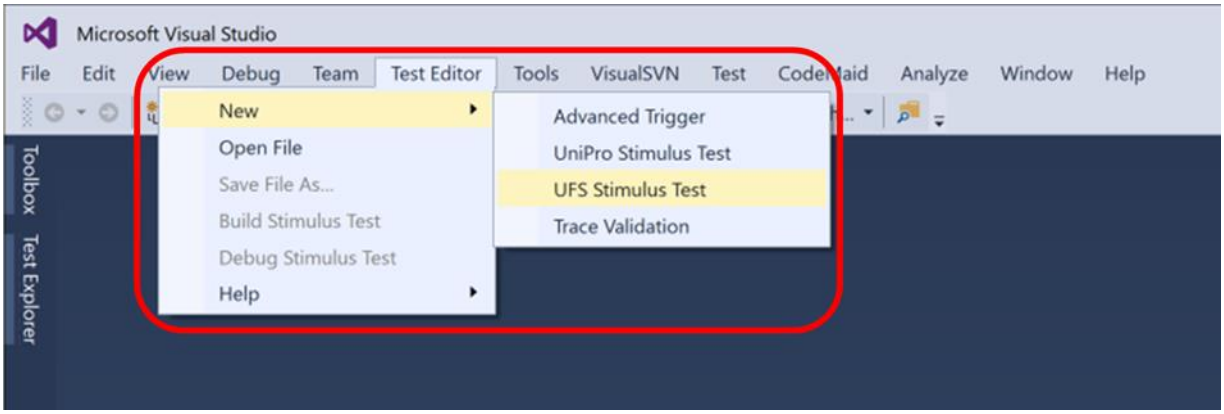
Go to the Stimulus Test Editor playlist on Protocol Insight's YouTube channel to view the Test Editor Training Series videos on how to create a stimulus test.

Using Protocol Insight's Test Editor

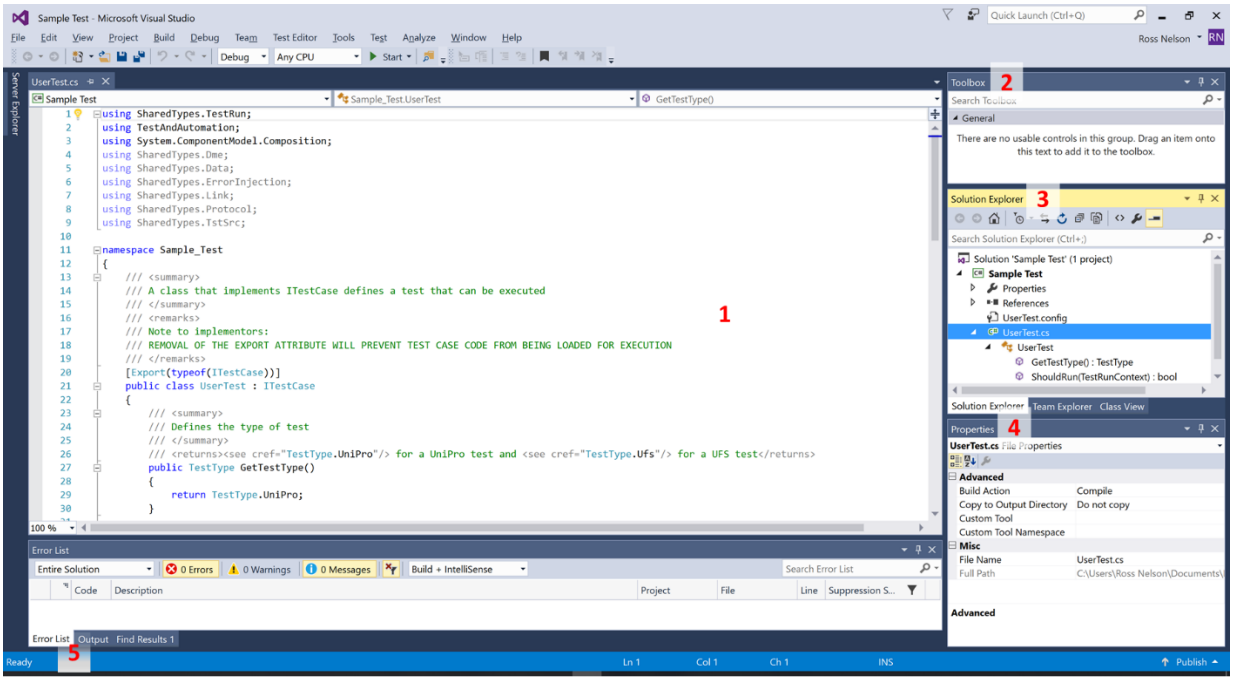
Stimulus tests are created using the C# programming language. Each test is represented by a Microsoft Visual Studio solution that contains a C# Class Library project. This project is built by Visual Studio using the Microsoft Build Engine (MSBuild) to create a Dynamically Linked Library or .dll file. The Falcon application software will find all tests in the \CustomTests folders that were created this way and make them available in the Test list in the Configuration window. The methods defined in step 4 of the "Editing the Test" section run directly from the Falcon software for test management, run condition checking and execution of the test.

To create or edit a Stimulus test, launch Protocol Insight's Test Editor from the Falcon application software from **Tools→Editor→Open Editor**. When Test Editor is launched an instance of Microsoft

Visual Studio will open and the Protocol Insight Test Editor plug-in will appear as a menu choice:



The default solution space for the Stimulus test solution includes the following components:

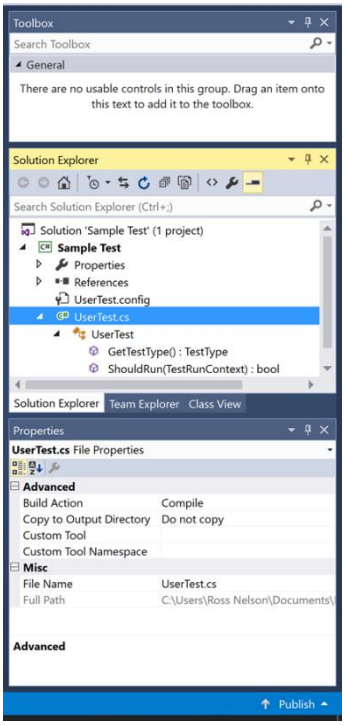


UserTest.cs editor 1

The UserTest.cs editor is where the stimulus test is written. When a new test is created the editor is populated from a default template. If UserTest.cs is not displayed in the editor double click on UserTest.cs in the Solution Explorer pane to bring its contents into the editor. If the Solution Explorer pane is not already open, select **View→Solution Explorer** from the Visual Studio menu.

The Toolbox 2

The Toolbox contains objects that are used for editing Advanced Triggers and Trace Validation tests. It is not used for Stimulus Test editing.



The Solution Explorer 3

The Solution Explorer provides a simple way to find all the objects from your Stimulus test in a tree view and is the quickest way to navigate to UserTest.cs. If the Solution Explorer pane is not already open, select **View→Solution Explorer** from the Visual Studio menu.

The Properties Window 4

The Properties Window is used for editing Advanced Triggers and Trace Validation tests. It is not used for Stimulus test editing.

Error List and Output 5

When the Stimulus test is built using **Test Editor→Build Stimulus Test**, the Output pane will show the output of the build and whether the build succeeded. If the build operation fails, then the Error List pane is used to get details on errors that are preventing the build from completing. If the Error List and Output panes are not already open, select **View→Error List** or **View→Output** from the Visual Studio menu.

Panes in Visual Studio can be docked and pinned in a similar way to panes in the Falcon application software. Pin or unpin panes by toggling the pin icon in the top right corner of the pane.

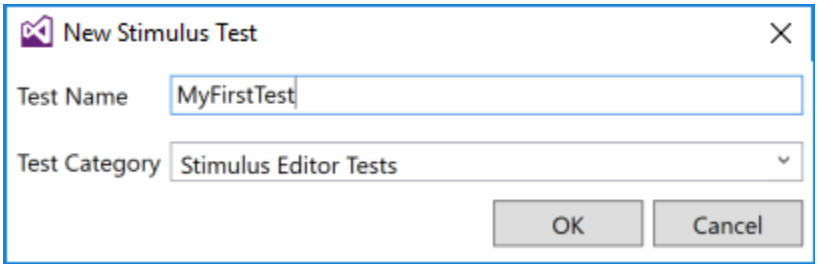
Creating a New Test

In Visual Studio select **Test Editor→New→Stimulus Test** and a prompt will appear to pick a test name and category.

Categories exist as folders under <User>\Documents\Protocol Insight\Falcon\CustomTests and are displayed in the Tests section of the Configuration window. New categories can be created by typing the new category name into the Category field in the New Stimulus Test dialog before saving and or can be created by manually adding new sub-directories in <User>\Documents\Protocol

Insight\Falcon\CustomTests with File Explorer

Choose an existing test category or create a new one and supply a name for the test:



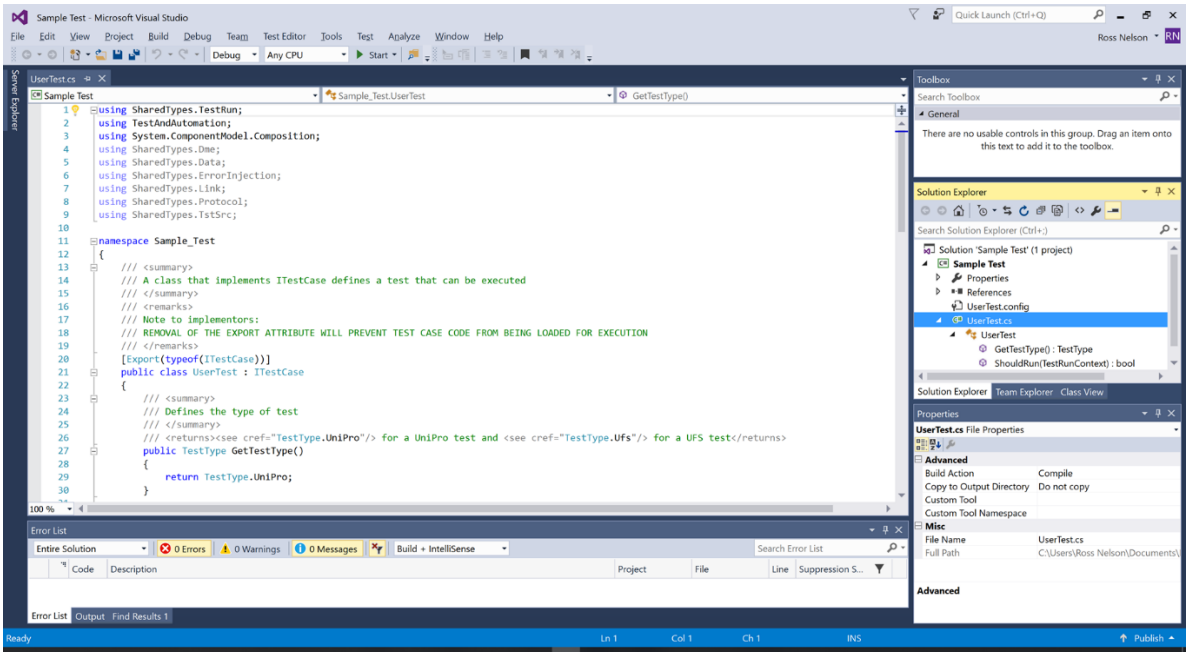
Opening an Existing Test

In Visual Studio select **Test Editor→Open File** and select a file with the .pie extension to open the Stimulus Test solution.

Editing the Test

1. To open an existing Stimulus test or create a new one:
    1. From the Falcon application software select **Tools→Editor→Open Editor** or **Tools→Editor→New UniPro Stimulus** or **Tools→Editor→New UFS Stimulus**.
    2. If Visual Studio has already been launched from the Falcon application software, select **Test Editor→Open File** or **Test Editor→New→Stimulus Test**.
2. The newly opened or created test solution will be listed in the Solution Explorer pane of the solution space as UserTest.cs and the template will be displayed in the editor. UserTest.cs is where the stimulus test is developed. If UserTest.cs is not displayed in the editor double click on UserTest.cs in the Solution Explorer pane to bring its contents into the editor. If the Solution Explorer pane is not already open, select **View→Solution Explorer** from the Visual Studio menu.

3. Visual Studio should look similar to this:



4. There are three methods defined in the UserTest.cs file:

    1. GetTestType() method
    This method should return a TestType enum value so that the Falcon application software can manage the test appropriately.

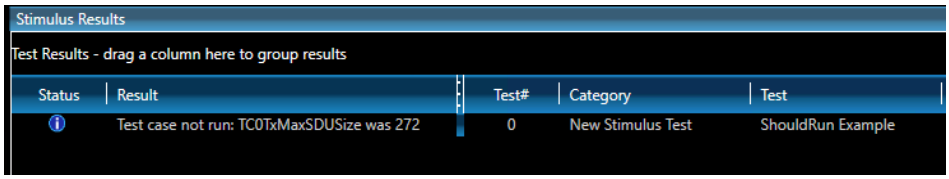
    2. ShouldRun(TestRunContext context) method
    This method should return a true or false boolean value indicating whether or not this test should be ran by the Falcon application software. By default this method returns true and the Falcon application software will always run the test. However, a user can restrict running a test to certain scenarios using the TestRunContext parameter. The following example shows a ShouldRun method that returns false when TC0TxMaxSDUSize is 272.

```
public bool ShouldRun(TestRunContext context) {
  /*This method is set up to return true which indicates
  * You may also return false to indicate that the test c
  * If you uncomment following lines, the test will only
  */

  if (context.DeviceValues.TC0TxMaxSDUSize == 272) {
    context.Error = "TC0TxMaxSDUSize was 272";
    return false;
  }

  return true;
}
```

    As seen in the example, the Error string property of the TestRunContext parameter can be set. The Falcon application software will now report the given error and indicate that the test was not run when the TC0TxMaxSDUSize is set to 272 in the Device Configuration.



To get a better idea of how to utilize the TestRunContext class, please refer to the Exerciser Commands API documentation.

3. TestBody(TestRunContext context, IExerciserAndStatusToolkit toolKit) method
   This method is where the Stimulus test is defined. The IExerciserAndStatusToolkit parameter has a variety of tools that can be used to interact with the exerciser. As seen in the example, the UniProTools property of the toolKit object is accessed in order to call the UniProExerciserAndStatusTools.BootUniPro() method.

```
public bool TestBody(TestRunContext context, IExerciserAndStatusToolkit toolKit) {
  UniProExerciserAndStatusTools tools = toolKit.UniProTools;
  /* Here you define the commands that your test will run.
  * the tools object has been created for you from the toolkit.
  * Using tools you can access the UniProExerciserAndStatusTools class which allows you to run UniPro commands
  * and retrieve UniPro status. This method is set up to return true which indicates success.
  * You may also return false to indicate a failure condition.
  * Uncomment the Sample line below and your test will issue a Boot UniPro command
  */

  tools.BootUniPro();

  return true;
```

   The full capabilities of the UniProExerciserAndStatusTools class is outlined in the Exerciser Commands API documentation.

5. Build the test to make it executable by the Falcon application software. Do this by selecting **Test Editor→Build Stimulus Test** from the Visual Studio menu.
   1. The Output pane (**View→Output** from VS menu) will give the output of the build and will indicate whether the build succeeded.
   2. If the build operation fails, the Error List (**View→Error List**) pane can be accessed to get details on errors that are preventing the build from completing.
6. Following a successful build, the Falcon application software will refresh its Stimulus tests and the new test will be displayed.

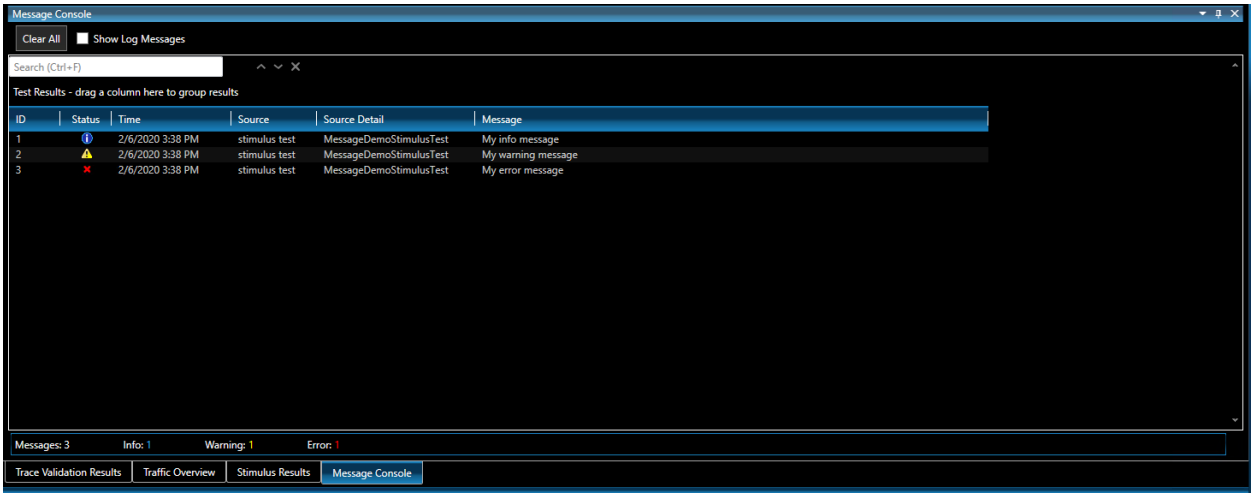
Outputting debug\status information to Message Console in the application software

The TestRunContext parameter to the TestBody() method can be used to output information to the message console in the application software. The following example shows the use of the TestRunContext.AddMessage() method in a test.

```
/// <summary>
/// The TestBody method contains the code for running a stimulus test
/// </summary>
/// <param name="context">The <see cref="TestRunContext"/> of the current test</param>
/// <param name="toolKit">A <see cref="IExerciserAndStatusToolkit"/> implementation
/// to access protocol specific Tools objects
/// </param>
/// <returns>True is the test succeeds. Otherwise returns false</returns>
public bool TestBody(TestRunContext context, IExerciserAndStatusToolkit toolKit)
{
  context.AddMessage("My info message", "stimulus test", "MessageDemoStimulusTest", LogTypes.Info);
  context.AddMessage("My warning message", "stimulus test", "MessageDemoStimulusTest", LogTypes.Warning);
  context.AddMessage("My error message", "stimulus test", "MessageDemoStimulusTest", LogTypes.Error);

  return true;
}
```

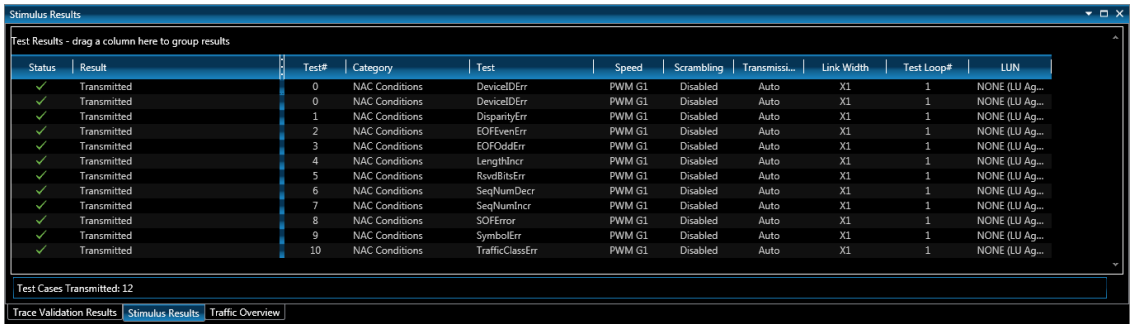
The AddMessage() method takes four arguments. The first argument is the text of the message. The second argument is the source of the message. In this case the text “stimulus test” is being used to give an indication of where the message was created. The third argument is the source detail which can be used to add detail about the message source. The name of the stimulus test is used to give specific information about which test generated the message. The fourth argument is a LogTypes enumeration value to indicate the level of the message. A message can be of the Info, Warning, or Error level. Running the test results in the following output to the Message Console.



Running the Test

1. In the Falcon application software select a Stimulus test from the Test list in the Configuration window. Tests created with the Stimulus Test Editor tool will have an orange S icon. Note: legacy Stim tests developed prior to Test Editor will be listed with a red S and are not editable with Test Editor.

2. Start the analyzer by pressing Start in the Toolbar.
3. Click the Start Stim button in the Toolbar.
4. When the Stimulus tests have completed the Application Status Bar in the Toolbar will show "Stimulus Tests Completed" and the Stimulus Results window will show the results of all individual tests:



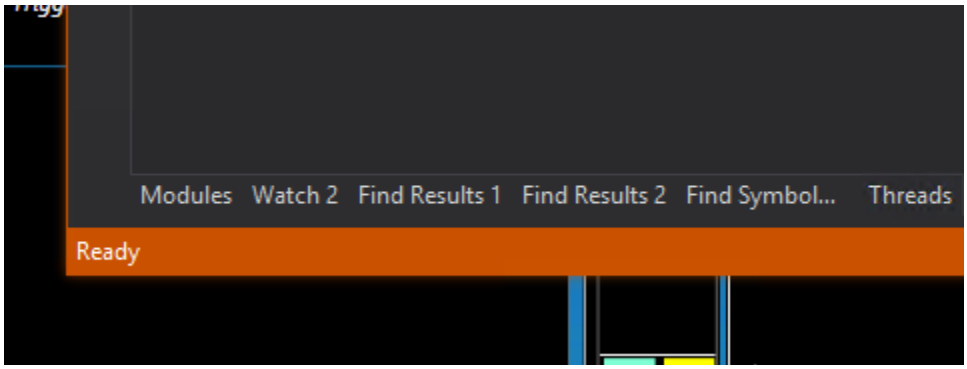
5. Stop the analyzer, wait for the trace to unload and view it in the Listing windows.

Debugging the Test

1. Make sure the Falcon application software is running and connected to HW.
2. Open the test from Visual Studio from **Test Editor→Open File** or from the Falcon application software from **Tools→Editor→Open Editor**.
3. Build the stimulus test with **Test Editor→Build Stimulus Test**.
4. Set a breakpoint on a line of code (for more information on setting a breakpoint in Visual Studio see this link): https://docs.microsoft.com/en-us/visualstudio/debugger/using-breakpoints?view=vs-2017
5. Visual studio should have a red dot in the left margin as seen in the example below

```
50    public bool TestBody(TestRunContext context, IExerciserAndStatusToolkit toolKit) {
51      UniProExerciserAndStatusTools tools = toolKit.UniProTools;
52      /*
53       * Example code:
54       * tools.BootUniPro();
55       */
56      tools.BootUniPro();
57      DmeConfigResult result = tools.DmePeerGet(ProtocolInsight.SharedTypes.Protocol.UniproAttrID.DL_TC1TxBufferSize, 0);
58      uint attributeValue = 0;
59      if (result.ConfigResultCode == ConfigResultCode.Success) {
60        attributeValue = (uint)result.AttributeValue;
61      }
62      return true;
63    }
64  }
```

6. In Visual Studio select **Test Editor→Debug Stimulus Test**.
7. When your test is ready to debug there will be a "Ready" indication in the bottom left corner of Visual Studio



8. In the Falcon application software select the test that is being debugged.
9. Start the analyzer.
10. Hit the Start Stim button.
11. The breakpoint that was set in step 4 should be hit as seen here

```
50    public bool TestBody(TestRunContext context, IExerciserAndStatusToolkit toolKit) {
51      UniProExerciserAndStatusTools tools = toolKit.UniProTools;
52      /*
53       * Example code:
54       * tools.BootUniPro();
55       */
56      tools.BootUniPro();
57      DmeConfigResult result = tools.DmePeerGet(ProtocolInsight.SharedTypes.Protocol.Uni
58      uint attributeValue = 0;
59      if (result.ConfigResultCode == ConfigResultCode.Success) {
60        attributeValue = (uint)result.AttributeValue;
61      }
62      return true;
63    }
64  }
```

12. Use the F10 button to advance the execution of the TestBody method. More information on debugging in Visual Studio can be found here https://docs.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour?view=vs-2017
13. To stop Debugging the test select **Test Editor→Stop Debugging Stimulus Test**.

Renaming the Test

1. Open the project properties of the Stimulus test project by right clicking the project in the Solution Explorer to open the project context menu.
2. Select the **Properties** menu item in order to display the project properties in the editor.
3. Under the Application section of the project properties edit the Assembly field. This property corresponds to the test name in the Falcon application software Test list.

Additional Resources

1. Source code samples

   The full source code for the UniProExerciserAndStatus.cs, UfsExerciserAndStatusTools.cs and UfsPackets.cs tools classes are available for reference while writing tests. The source code gives examples of how methods can be composed into other methods. For example HibernateForTime(int time) is built up using HibernateEnter(), HibernateExit(), and the Thread.Sleep(TimeSpan time)

methods.

The source code can be opened in the editor from
Test Editor→Help→Open UniProExerciserAndStatusTools.cs source
Test Editor→Help→Open UfsExerciserAndStatusTools.cs
Test Editor→Help→Open UfsPackets.cs source:

```
/// <summary>
/// Enter Hibernate for a specific amount of time
/// </summary>
/// <param name="time">Time in milliseconds</param>
public void HibernateForTime(int time) {
  HibernateEnter();
  Thread.Sleep(time);
  HibernateExit();
}
```

2. Tests are written using the C# programming language. There are a lot of resources on the internet for learning C#, including: https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/intro-to-csharp/index
3. Since Stimulus Editor is an extension to Visual Studio, customers are highly encouraged to take advantage of the Intellisense feature. Users can find a guide here: https://docs.microsoft.com/en-us/visualstudio/ide/using-intellisense?view=vs-2017

Exerciser Commands API Help

Stimulus Test Editor uses classes for developing stimulus tests. The classes provide methods that can be called from user-defined tests to exercise the DUT. The API commands are documented in a separate *Exerciser Commands API Help* document that is accessable from **Test Editor→Help**.

Stimulus Test Editor File Types

Protocol Insight Exerciser File (.pie) *(indicated in the Test list with an orange S icon)*

The Protocol Insight Exerciser .pie file contains all the information needed to create, edit and run Stimulus Test Editor tests. The .pie file is used by Test Editor and runs in the Falcon application software.

Protocol Insight Stimulus File (.pis) *(indicated in the Test list with a red S icon)*

Legacy UniPro and UFS Stimulus Builder *.pis files can be executed with the v1.8.0 or later Falcon application software but cannot be edited using Test Editor. The original UniPro and UFS Stimulus Builder tools will continue to be available in the Falcon application for users to update or to help with migration of their original stim tests from the Builder to the Editor, however it will no longer be maintained or supported.

General Information - Protocol Insight's Test Editor

Test Editor™ is used to create and modify Advanced Trigger, Trace Validation and Stimulus Test files.

For proper Test Editor license verification, the Falcon application software must be connected to the Falcon instrument. Launch the Falcon application software and click the Connect button on the Toolbar to connect to the instrument and verify the license.

Multiple Advanced Trigger and Trace Validation files can be opened simultaneously, but only one Stimulus Test file can be opened at a time.

While the Advanced Trigger and Trace Validation files are edited using the graphical .sm.diagram drag-and-drop model, currently all Stimulus Tests must be edited in the solution space editor in C#. .sm.diagram drag-and-drop editing of Stimulus Tests will be added in a future release.

Host PC requirements

- Intel® Core™ i7 or i9 processor or equivalent.
- 32 GB RAM recommended, 16 GB minimum.
- NVMe solid state drive with 500GB free space recommended, 256GB minimum free space.
- Thunderbolt 3 enabled type-c connector is required if connecting to a Falcon instrument.
- Microsoft Windows 10 64-bit operating system

Software requirements

- Windows 10 64-bit operating system
- Microsoft SQL Server 2014 or later
- Microsoft Visual Studio 2015 Community or Professional editions
  Note: Stimulus Test Editor is not supported on Microsoft Visual Studio 2015 Isolated Shell
- Protocol Insight application software and firmware version 1.8.0.8170 or later.