



Falcon Series

Creating and Using Advanced Triggers User Manual

Copyright © Protocol Insight. All rights reserved. Licensed software products are owned by Protocol Insight or its suppliers, and are protected by national copyright laws and international treaty provisions.

Protocol Insight products are covered by U.S. and foreign patents, issued and pending. Information in this manual supersedes all previously published material. Details, specifications and pricing subject to change.

Protocol insight is a registered trademark of Protocol Insight, LLC.

MIPI and the MIPI logo are a licensed trademark of the MIPI Alliance.

UFS and UFS Logo are a trademark of the Universal Flash Storage Association

JEDEC® and the JEDEC logo are registered trademarks of JEDEC Solid State Technology Association.

Contact Protocol Insight at:

sales@protocolinsight.com

support@protocolinsight.com

www.protocolinsight.com



Contents

General Operation	1
Using Protocol Insight's Test Editor	1
Creating an Advanced Trigger	3
Creating a simple single-state Advanced Trigger	3
Creating a multi-state Advanced Trigger	4
Testing a new Advanced Trigger	4
Setting an Advanced Trigger to Capture a Trace	4
Advanced State Machine Concepts	5
State Machine	5
AdvancedTriggerState	5
AdvancedTriggerEvent	6
Transitions	6
Validators	6
Actions	6
Counters	6
Advanced Test Editor Concepts	6
State Machine	6
States	7
Events	7
Transitions	10
Comments	10
Advanced Trigger File Types	10
Advanced Trigger Specifications	11
Advanced Trigger Limitations	11
General Information - Protocol Insight's Test Editor	11



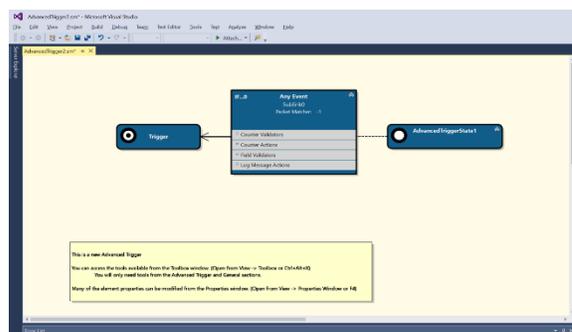
The Protocol Insight® Advanced Trigger Test Editor™ allows users to create advanced triggers using the same Test Editor graphical editor that is used for creating Trace Validation™ analysis tests.

Once created, advanced triggers are set in the Trigger Setup window so they trigger when the analyzer is running. They can also be run on previously captured traces from the Configuration window to verify correct trigger logic or perform post-capture analysis.

Advanced Trigger is available for all Falcon G300/G400 series instruments but is not available with the Raptor M300/M400 series.

General Operation

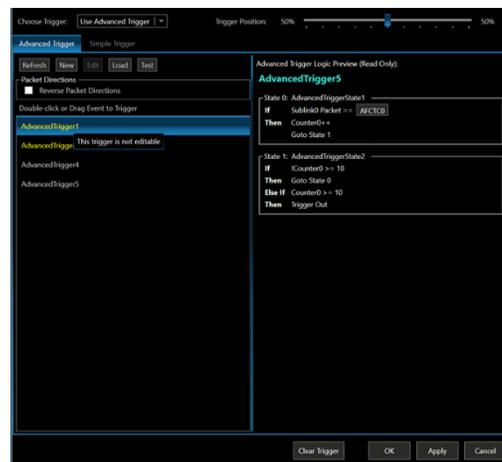
Advanced Triggers are created and edited using the Test Editor, which can be accessed from the Falcon application software from **Tools → Editor → Open Editor** or by selecting Advanced Trigger from the Choose Trigger dropdown in the Trigger Setup window and clicking New or Edit. A simple Advanced Trigger example is provided in the \AdvancedTrigger directory.



Once created, Advanced Triggers in the \AdvancedTrigger directory are automatically added to the list of available triggers in the left pane of the Advanced Trigger tab in the Trigger Setup window.

Note: Advanced Triggers that were saved with and then loaded from a *.pitx trace do not include the full *.piat Advanced Trigger information and thus are not editable; these are listed in yellow with a ToolTip "This trigger is not editable". To modify a non-editable Advanced Trigger the user must open the *.piat file in the \AdvancedTriggers directory.

Setting an Advanced Trigger is handled from the Trigger Setup window, by double click or dragging the trigger of interest from the left pane of the Trigger Setup window into the "Drag Advanced Trigger Here" pane on the right side.



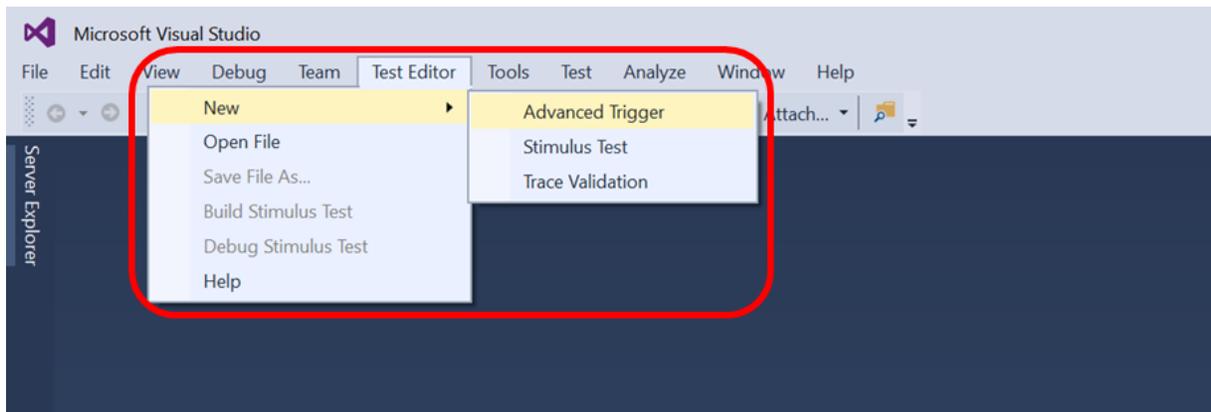
Using Protocol Insight's Test Editor

To create or edit an Advanced Trigger, launch Protocol Insight's Test Editor from the Falcon application software from **Tools → Editor → Open Editor** or by selecting Advanced Trigger from the Choose Trigger dropdown in the Trigger Setup window and clicking New or Edit.

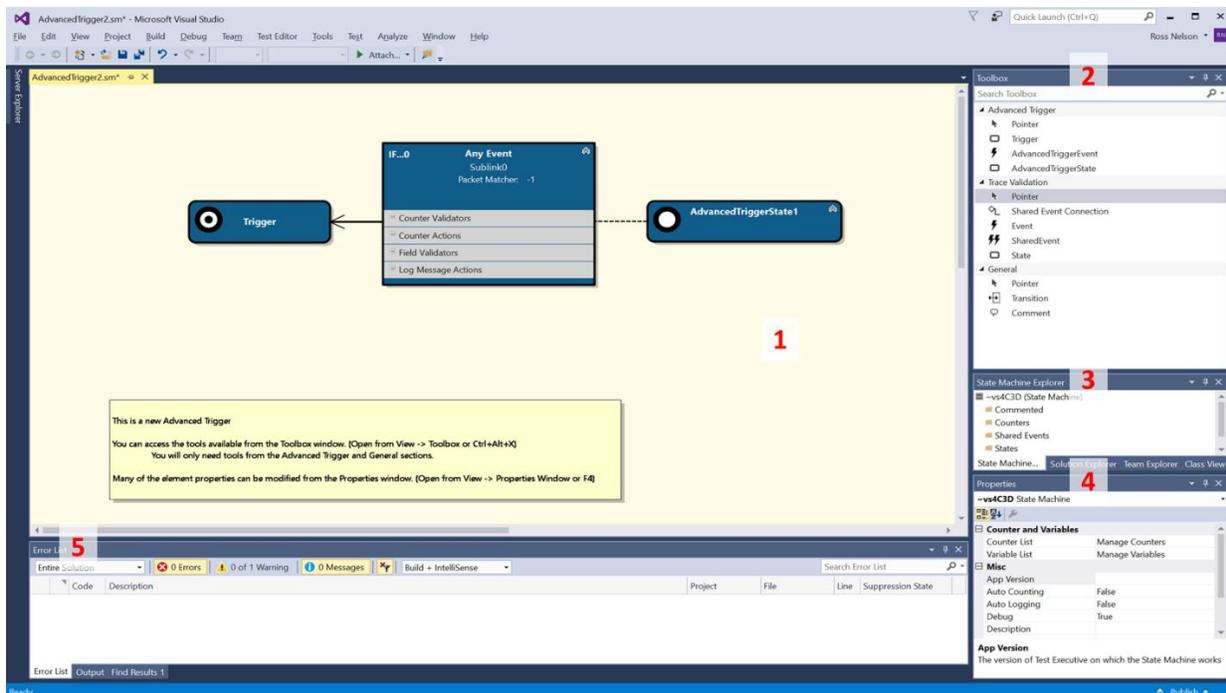
When Test Editor is launched an instance of Microsoft Visual Studio will open and the Protocol Insight



Test Editor plug-in will appear as a menu choice:



Select **Test Editor**→**New**→**Advanced Trigger** and an Advanced Trigger template will be opened in the solution space. The solution space for the Advanced Trigger template includes the following components:



The Diagram 1

The Diagram is the visual representation of a State Machine. It is the creative space in the file and is represented by the yellow background. Each template contains information about one State Machine and therefore only has one Diagram.



The Toolbox 2

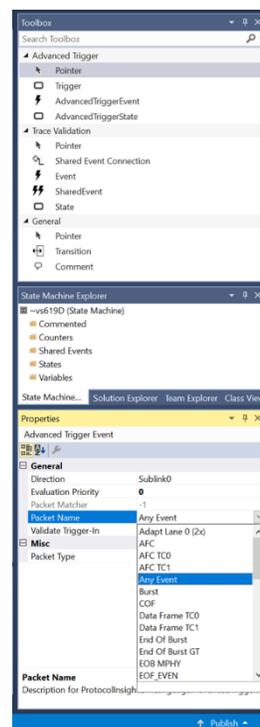
The Toolbox, which is on the right side of the application by default, contains most of the objects that you can add to a State Machine. To add an AdvancedTriggerState, AdvancedTriggerEvent, Transition or Comment just click and drag that item from the toolbox to the Diagram. Events can only exist as children of States, so to add an Event, click and drag the Event toolbox item onto an existing State on the Diagram. Transitions define the transition between Events and States. To add a Transition, first select Transition in the toolbox, click the event you want the transition to come from, and finally click the State the want the Transition to go to.

The State Machine Explorer 3

The State Machine Explorer provides a simple way to find all the objects from your State Machine in a tree view.

The Properties Window 4

The Properties Windows is where the vast majority of the editing takes place in a State Machine. Therefore it is important that you leave this window open and available. If you should close it, you can open it again from the View → Properties menu option or by pressing F4. The information displayed in the Properties Window will correspond to the item you have selected in the diagram.



Error List 5

The Error List will display a list of Warnings and Errors present in the State Machine. Warnings provide information on things that could cause unexpected behavior but will not prevent the Advanced Trigger from running in the Falcon software. Errors are items that need to be corrected as they will cause problems in the Falcon software. The executable Advanced Trigger files will not be generated if you have errors but you will be still be able to save your State Machine file.

The Advanced Trigger template includes an Advanced Trigger State, an Advanced Trigger Event and a Trigger. These are the minimum recommended objects that should be included in every Advanced Trigger file.

Creating an Advanced Trigger

Creating a simple single-state Advanced Trigger

1. In Test Editor select **Test Editor** → **New** → **Advanced Trigger**. This will open an Advanced Trigger template that has one state, one event and a trigger.
2. The AdvancedTriggerState1 will be pre-defined as an Initial state.
3. Modify the AdvancedTriggerEvent by right-clicking to add Counter Validators, Counter Actions, Field Validators and Log Message Actions.
4. Set the desired values for the Event validators and actions in the Properties.
5. If desired add additional AdvancedTriggerEvents and set the desired properties.
6. Connect the additional Events to the Trigger using Transitions from the Toolbox.



7. Save the advanced trigger in the default \AdvancedTrigger directory. Any triggers saved in the \AdvancedTrigger directory will automatically populate in the list of available triggers when the Trigger Setup window is opened in the Falcon application software.

Creating a multi-state Advanced Trigger

1. In Test Editor select **Test Editor**→**New**→**Advanced Trigger**. This will open an Advanced Trigger template that has one state, one event and a trigger.
2. The AdvancedTriggerState1 will be pre-defined as an Initial state.
3. Add additional AdvancedTriggerStates from the Toolbox by dropping them anywhere in the yellow state machine solution space.
4. Add AdvancedTriggerEvents to each state from the Toolbox by dropping them on the AdvancedTriggerState in the solution space.
5. Use Transitions from the Toolbox to connect all AdvancedTriggerEvents to subsequent AdvancedTriggerStates.
6. Modify each AdvancedTriggerEvent by right-clicking to add Counter Validators, Counter Actions, Field Validators and Log Message Actions.
7. Set the desired values for the Event validators and actions in the Properties.
8. Save the advanced trigger in the default \AdvancedTrigger directory. Any triggers saved in the \AdvancedTrigger directory will automatically populate in the list of available triggers when the Trigger Setup window is opened in the Falcon application software.

Testing a new Advanced Trigger

Once an advanced trigger has been created and saved, it can be run against previously saved traces in the same fashion as a Trace Validation test. Thus, if a user has a trace that exhibits the behavior that the trigger was designed to find in new acquisitions, the user can verify the new advanced trigger by running it against the previously saved trace.

In the Falcon application software open the saved .pitx trace containing the trigger event. Then open the Configuration window from the menu at **View**→**Configuration**. Scroll to the bottom of the window to the Tests section. In the category *Advanced Trigger Offline TV* the new trigger will be listed, select that trigger and click Start TV in the Toolbar. If the trigger is properly constructed and the trigger event exists in the saved trace the message “Triggered” will be displayed in the Rule column of the Trace Validation Results window.

If the new Advanced Trigger isn’t listed, choosing **File**→**Refresh Advanced Trigger Tests** will update the Test list.

Setting an Advanced Trigger to Capture a Trace

Once an Advanced Trigger has been created using the Editor, it can be set to trigger the Falcon instrument from the Trigger Setup window.

The Trigger Setup window is accessed from the Tools menu or the Toolbar. The Choose Trigger dropdown allows selection of different trigger actions, including **No Trigger**, **Trigger Immediate**, **Simple Trigger** and **Advanced Trigger**.





Choosing Advanced Trigger will open the Advanced Trigger tab. The left pane of the Advanced Trigger tab will display a list of all Advanced Triggers that are available in the \AdvancedTrigger directory, with buttons to Refresh that list, launch the Test Editor to create a New trigger or Edit an existing one, or Load triggers from other directories.

The right pane Advanced Trigger Logic Preview is Read Only and displays the trigger state logic of the selected trigger. Clicking on a packet listed in the trigger logic will open the Packet Editor as Read-Only to display the header values loaded into the Advanced Trigger.

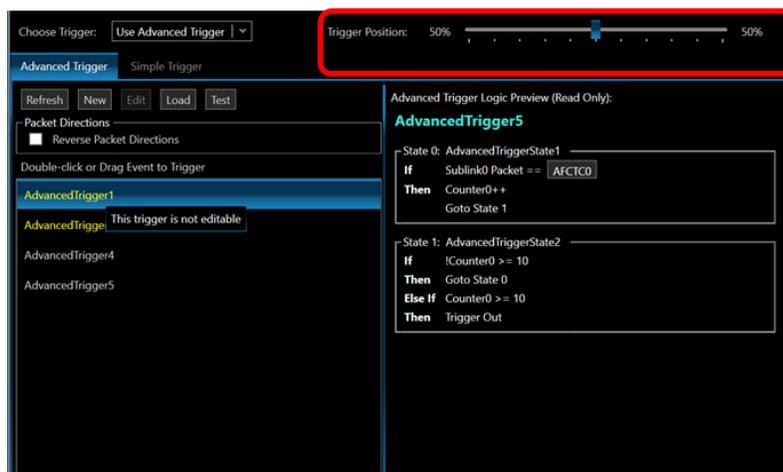
To set an Advanced Trigger, double click or drag the trigger of interest from the left pane into “Drag Advanced Trigger Here” in the right pane.

Trigger IN/OUT

The TRG IN connector on the Falcon front panel can be selected as an Analyzer Resource in the Simple Trigger window or in the Advanced Trigger Editor and can be used with other packet events. When a trigger event has occurred a TRG OUT is always asserted to the front-panel TRG OUT connector.

Trigger Position

The **Trigger Position** is used to set how much acquisition memory is reserved to save trace data prior to the trigger event vs. after the trigger event. If a 20% trigger position is set the analyzer will reserve 80% of the memory for post-trigger fill. This means that if the trigger occurs deep in the trace then 100% of the buffer will fill, 20% before and 80% after. But if the trigger event occurs very early in the trace, say after only filling 5% of the total memory, the capture will total less than 100% because 5% will be before the trigger and 80% after the trigger for 85% total.



Advanced State Machine Concepts

Advanced Triggers are built on a State Machine model. A state machine is a system that stores the status of something at a given time and, based on an input, can change the status. State machines can be described as discreet states with a set of possible events that will cause the system to move to a new state. States can also contain a set of actions or output events that trigger based on changing input.

State Machine

The highest level of an Advanced Trigger is the State Machine. There is only one State Machine in each template, and it represents the overall system through which data is passed for validation.

AdvancedTriggerState

All State Machines must begin with an Initial AdvancedTriggerState in which the State Machine will wait for an Event to trigger in order to move to another State. Up to four additional AdvancedTriggerStates



can be defined in addition to the initial state. An `AdvancedTriggerState` is a discreet status indicting where in the State Machine the validation process is at present. As events occur, the system will move from State to State and actions can trigger that update information in the system. State Machines can only be in one State at any given time.

AdvancedTriggerEvent

An Event defines a set of conditions (implemented as Validators) that, when all are true, will trigger a transition to another State. States can have more than one event but only one Event can ever fire for a given State. See the *Event Evaluation Priority* section for more detail.

Transitions

Transitions define which State the State Machine should enter after an Event has occurred. Each Event can only have one associated Transition. More than one Transition may feed into a single State.

Validators

There are several kinds of Validators available in `AdvancedTriggerEvents`. These validators are used to look at packets or variables to validate the data. When all the Validators on an Event are true, it will cause the Event to fire.

Actions

Actions can occur when an Event fires or when a State is entered. Test Editor provides actions to Log Messages, Update Counters, and Set or Modify Variables.

Counters

Counters are used for statistics and allow the user to provide detail on how many times something happens in a State Machine. For example, how many of a particular type of packet were found in a trace.

Advanced Test Editor Concepts

When users create a new Advanced Trigger, they are provided with a basic State Machine template. Starting with this template, users define States and Events to move assert a trigger condition.

State Machine

State Machines are the highest level parent in the file and contain all other objects. In order to select the State Machine, click anywhere on the yellow background of the diagram.

State Machine Properties

- **App Version** – A String value containing the application version for which this state machine was written. (Default: None)
- **Auto Counting** – A Boolean value which tells the State Machine whether or not to automatically add counters. When true, the engine will automatically add counters for Success, Fail, and Total. When false, the user must add any counters in the State Machine by using Counters and Counter Actions. (Default: False)
- **Auto Logging**– A Boolean value which tells the State Machine whether or not to automatically log messages. When true, the engine will automatically log success and failure messages. When false,



the user must log all messages in the State Machine by using Log Message Actions. (Default: False)

- **Debug** – A Boolean value which tells the State Machine whether or not it can be run in debug mode. (Default: True)
- **Description** – A string value containing a description of the State Machine. (Default: None)
- **Inactivity Timeout** – A Boolean value which tells the State Machine whether to timeout when the Global Inactivity Timeout is reached. (Default: True)
- **Max Instances** – Controls the number of instances that may be run at once. If MaxInstances is 0 then an unlimited number may run.
- **Protocol** – A string value containing the protocol for which this was written. This applies to Trace Validation tests only. (Default: UniPro)
- **Stimulus Test Case** – A string value containing the name of the corresponding Stimulus Test Case Category and Name that will generate an appropriate trace for this State Machine to validate. For Example: "MyCustomCases\MyCustomStimulusFile" will establish a link to the MyCustomStimulusFile test case in the MyCustomCases category. This applies to Trace Validation tests only. (Default: None)
- **Counter List** – Clicking the ellipsis in this field opens the Counter Manager dialog. See the *Counters* section later for more detail.
- **Variable List** – Clicking the ellipsis in this field opens the Variable Manager dialog. See the *Variables* section later for more detail.

States

States are the primary building blocks of a State Machine. States define the current status of the State Machine and their Events define the next State to which the State Machine will move. When entering into a State, any actions on the State will run and then the State Machine simply waits for an Event to fire. There are two kinds of States available in an Advanced Trigger State Machine.

- **Initial** – This is the beginning State. Each State machine may only have one Initial State. In general, no events will transition back into an Initial State.
- **Normal** – The bulk of the States in your State Machine will be Normal. These States are transitioned to after an Event and they have their own Events which transition to other states.

State Properties

- **Kind** – A String value which identifies the type of State as describe above. (Default: Normal)
- **Name** – A String value containing the name of the State. This is for identification and each State must have a unique name. (Default: StateX)

Events

Events are the main driving force behind a State Machine. You use Events to identify the data you are looking for in a trace, and they then allow you to move to the next State in the State Machine.

Events have actions available to log messages and work with counters. You can define actions that will be executed when the Event is entered during validation. You can select the new action and edit its properties in the Properties Window. Actions are deleted via Right Click and Delete or by selecting them and pressing the Delete key. The Actions available are described below.



Event Properties

- **Direction** –
- **Evaluation Priority** – An Integer value that identifies the order in which the Events are to be evaluated, see *Event Evaluation Priority*. (Default: 10)
- **Packet Matcher** –
- **Packet Name** –
- **Validate Trigger-In** –
- **Packet Type** –

Event Evaluation Priority

Event Evaluation Priority allows the user to control how Trace Validation will handle Events when there are more than one attached to a single State. Since only one Event can ever fire for a State, this is an issue if there are multiple Events with Validators that all evaluate to True at the same time. In this case, the event with the lowest Evaluation Priority will fire and the others will be ignored. If multiple Events are true and have the same Evaluation Priority, there is no guarantee as to which Event will fire. This is more clearly explained with the following examples.

Example 1: One Event with all Validators True

Note: Evaluation Priority doesn't affect Events in this case.

Event Name	All Validators True?	Evaluation Priority	Event Fires?
Event 1	Yes	13	Yes
Event 2	No	1	No
Event 3	No	2	No
Event 4	No	3	No

Example 2: Multiple Event with all Validators True

Note: Evaluation Priority does affect Events in this case.

Event Name	All Validators True?	Evaluation Priority	Event Fires?
Event 1	Yes	13	No
Event 2	No	1	No
Event 3	Yes	2	Yes
Event 4	Yes	3	No



Example 3: Multiple Event with all Validators True and Same Priority

Event Name	All Validators True?	Evaluation Priority	Event Fires?
Event 1	Yes	10	?
Event 2	No	10	No
Event 3	Yes	10	?
Event 4	Yes	10	?

These examples illustrate how Event Evaluation Priority affects which Event fires. If there is ever a case where multiple Events can have all their Validators evaluate to true (as is the case in Example 3), then specifying an Evaluation Priority is important to avoid unexpected results.

Counter Validators

Counter Validators validate if the condition of Counter0 or Counter1 is true or false.

Counter Actions

Counter actions are used to update the value of a counter. When defining a Counter action, choose the counter from the list of available counters and then set a value. The Action property can be set to ++ or - - to tell the State Machine to add or subtract 1 to the current value of the counter.

Field Validators

Field Validators are used to compare specific packet header bits to a defined value. When all the Validators in an Event are True, the Event will fire and transition to the next State.

Field Validators allow you to compare the data in a field of a packet to other data.

Log Message Actions

Log Message Actions generates messages that are displayed in the Falcon software with the given Status and Text. The text is displayed in the Trace Validation Results window along with an icon representing the status. For more information.

The possible Status values are:

-  **Failed**
Used to indicate a failure in the Trace Validation. Messages of this type are usually logged just before transitioning to a Fail State or in a Fail State.
-  **Warning**
Warning messages are logged to indicate an issue that needs attention but that does not cause the validation to fail.
-  **Passed**
The Passed status is used to log messages after a successful validation has occurred. These types of messages are most often logged in a Success State.
-  **Info**
Info messages are used to provide further information to the Falcon software. These messages are not meant to indicate an issue with validation but simply to provide additional details or an



explanation as to why something may not have been tested.

Transitions

Transitions simply tell the State Machine which State to move to after an Event fires. Transitions are represented in the Diagram as lines between Events and States. In order to tell the difference between a Transition and the connector between States and their child events, transitions have an arrowhead on the end that connects to a State.

Comments

Comments allow you to add a simple comment box to the diagram. This can be used as documentation or notes on the diagram to provide detail or explain a complicated state.

Advanced Trigger File Types

Protocol Insight Advanced Trigger File (.piat)

The Advanced Trigger .piat file contains all the information needed to create, edit and run Advanced Triggers. The .piat file is used by Test Editor and by the Falcon trigger machine.

.piat files consist of four sub-files, which can be accessed by renaming the .piat file to a zip file. **These sub-files should not be accessed except by expert users.** The four sub-files are .sm, .sm.diagram, .piv and .adv. The .sm and .piv files use XML and are therefore readable and editable in any XML or Text editor, however due to their complexity it is highly recommended that these files are only modified using Test Editor.

State Machine File (.sm)

The State Machine file has a .sm extension. They contain all the pertinent data related to the State Machine and its objects.

State Machine Diagram File (.sm.diagram)

The State Machine Diagram file has a .sm.diagram extension. These files are not directly edited, but they are required. They contain all the location and size data necessary to display objects visually on the diagram in Test Editor.

Protocol Insight Trace Validation File (.piv)

A Trace Validation XML output file is generated by Test Editor when you save an Advanced Trigger with no errors, and is saved as a *.piv file. This file contains all the information that the Falcon software requires in order to run Advanced Trigger as a Trace Validation test against a previously captured trace. These files are generated each time a user saves and will only be generated if the State Machine contains no errors.

Advanced Trigger File (.adv)

The Advanced Trigger .adv file contains the actual trigger logic generated from the Test Editor that is read into the Falcon trigger machine. These files are generated each time a user saves and will only be generated if the State Machine contains no errors.



Advanced Trigger Specifications

States: 4 N way branching, Trigger In, Trigger Out, Go To State

Packet Matchers: 5 per direction (8 Symbol Matchers each).

Additional State/FLR/SKP matcher: 2 per direction

Counters: 2

Compare Actions: < (Less Than), == (Equal), > (Greater Than), >= (Greater Than OR Equal)

Modify Actions: Hold (No Change), Count + 1 (Increment), Count - 1 (Decrement)

Only one Compare Action per Counter can be defined in a trigger. That one Compare Action is available in all states. You can't program "Counter1 < 5" in one state and then "Counter1 > 5" in another state. If you program "Counter1 < 5" in one state then that is the only Compare Action available for Counter1 in any state of the trigger. Either or both counters can be incremented or decremented in each state.

Advanced Trigger Limitations

Matching Multiple Small Packets

Trigger on small packets (<8 bytes for 2 lanes, <4 bytes for 1 lane) may not work as expected in some cases. If the packets are back to back (immediately following each other) the second packet may not be accounted for correctly. Furthermore if packets from opposite directions are seen at the exact same pipeline stage (clock cycle) the same issue occurs.

Direction Skew

There is an indeterministic skew between the 2 directions, this skew can be multiple pipeline stages (clock cycle). When trying to trigger on a packet A followed by packet B in the opposite direction, it is possible that the trigger state machine could see packet B before packet A.

Counter Latency

The counter match logic has a 1 pipeline stage (clock cycle) latency from the command to increment or decrement to when the counter compare occurs.

Count Accuracy

The counter is only able to increment or decrement once per pipeline stage (clock cycle). Thus if 2 packets are seen in the same clock cycle the counter will only increment or decrement once. Thus it cannot accurately count packets that are <8 bytes for 2 lanes, <4 bytes for 1 lane.

General Information - Protocol Insight's Test Editor

Test Editor™ is used to create and modify Advanced Trigger, Trace Validation and UniPro Stimulus Test files. UFS Stimulus Test file creation will be added with a future release.

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 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
- Protocol Insight application software and firmware version 1.0.4872 or later.

