

How To Create A Latching Button In Pathfinder

June 17, 2008
Bruce Wilkinson

Pathfinder Version:

All versions that support stacking events. Tested specifically on ver. 4.19.

Overview

Latching buttons require specific steps and the use of memory slots. Setting them up is not too difficult; however it is not something you can simply figure out “on the fly”.

With any latching switch, there are two switch states: ON and OFF. In addition there are two states for the button - PRESSED or NOT PRESSED (call these UP or DOWN for simplicity). Our stacking event logic must sort this out in order to provide latching control. Three stack events are required. There are many variations that may involve GPIO, mini panels and user panels however the general process is described below.

Stacking Event A

- will run if switch state is OFF and button is DOWN and memory is BLANK
- actions will write a memory value, perform action A and set switch state to ON

Stacking Event B

- will run if switch state is ON and button is DOWN and memory is BLANK
- actions will write a memory value, perform action B and set switch state to OFF

Stacking Event C

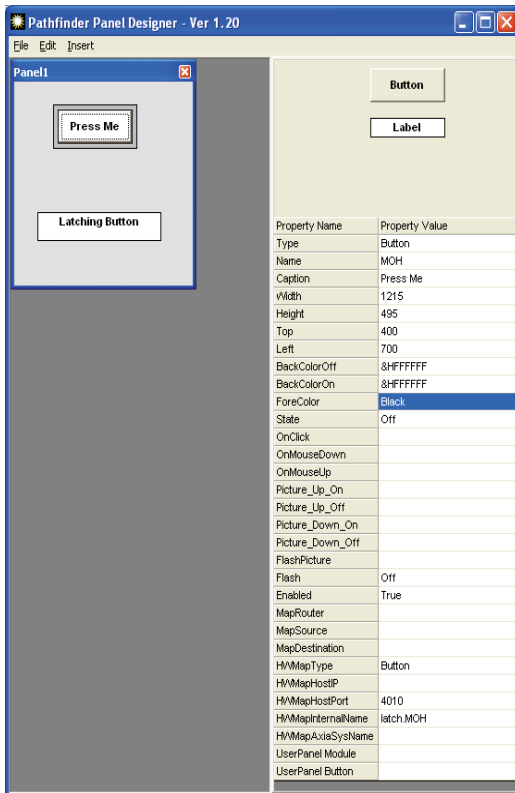
- runs if button is UP
- action clears memory

In this example, we will use a simple latching mini panel button to change an audio route and a GPO.

Remember, these same principles apply whether you have a hardware or software button, or even a couple of GPI's that you want to use to create a latching state.

Detailed Example

Now, let's get to a specific example. Let's control an audio route and a GPO with a mini panel latching switch and let's have the background color of the switch indicate the on-off status. In the real world, we might use a switch like this to control the audio feed delivered to our “music on hold” for our office phone switch or to change the feed to a recording device.



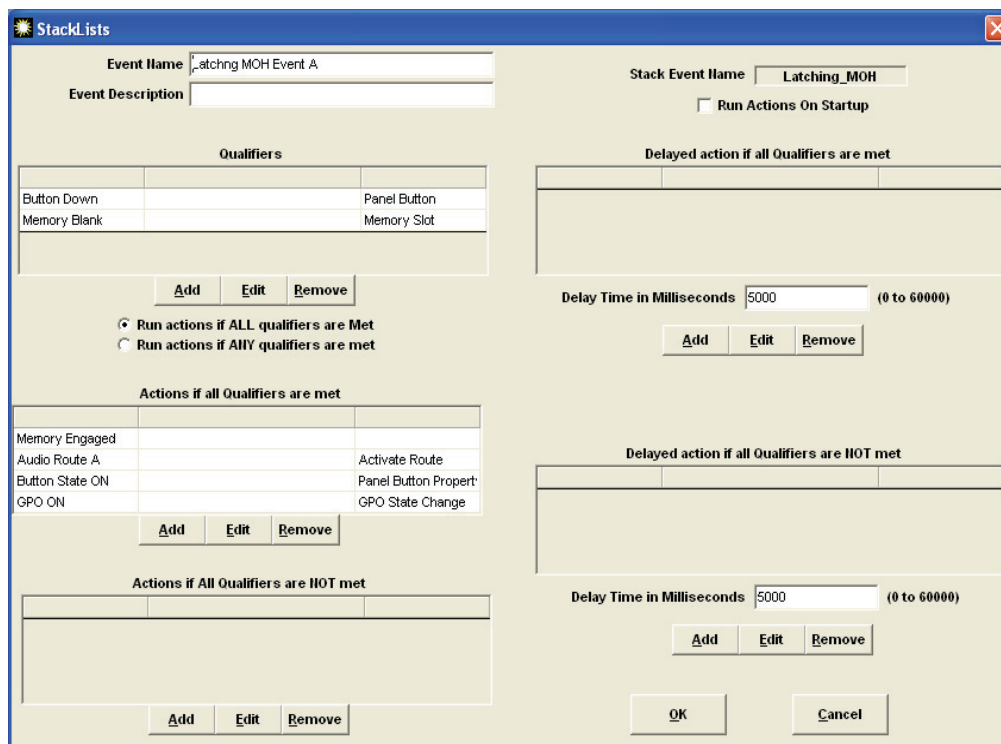
First, create a mini panel with a button. In this example, our panel is named LATCH and our button is named MOH.

Note that all we really need to define is the button itself, and give it a name. The button colors and button state will be defined as actions in the stacking events.

Now we will setup the three stack events that we need to make this behave as a latching button and to perform the desired actions.

IMPORTANT NOTE: The sequence order of the actions is very important. The memory value must be set first. Otherwise as soon as the route change in Stack A happens, stack event B's qualifiers are met and we will find ourselves in a loop.

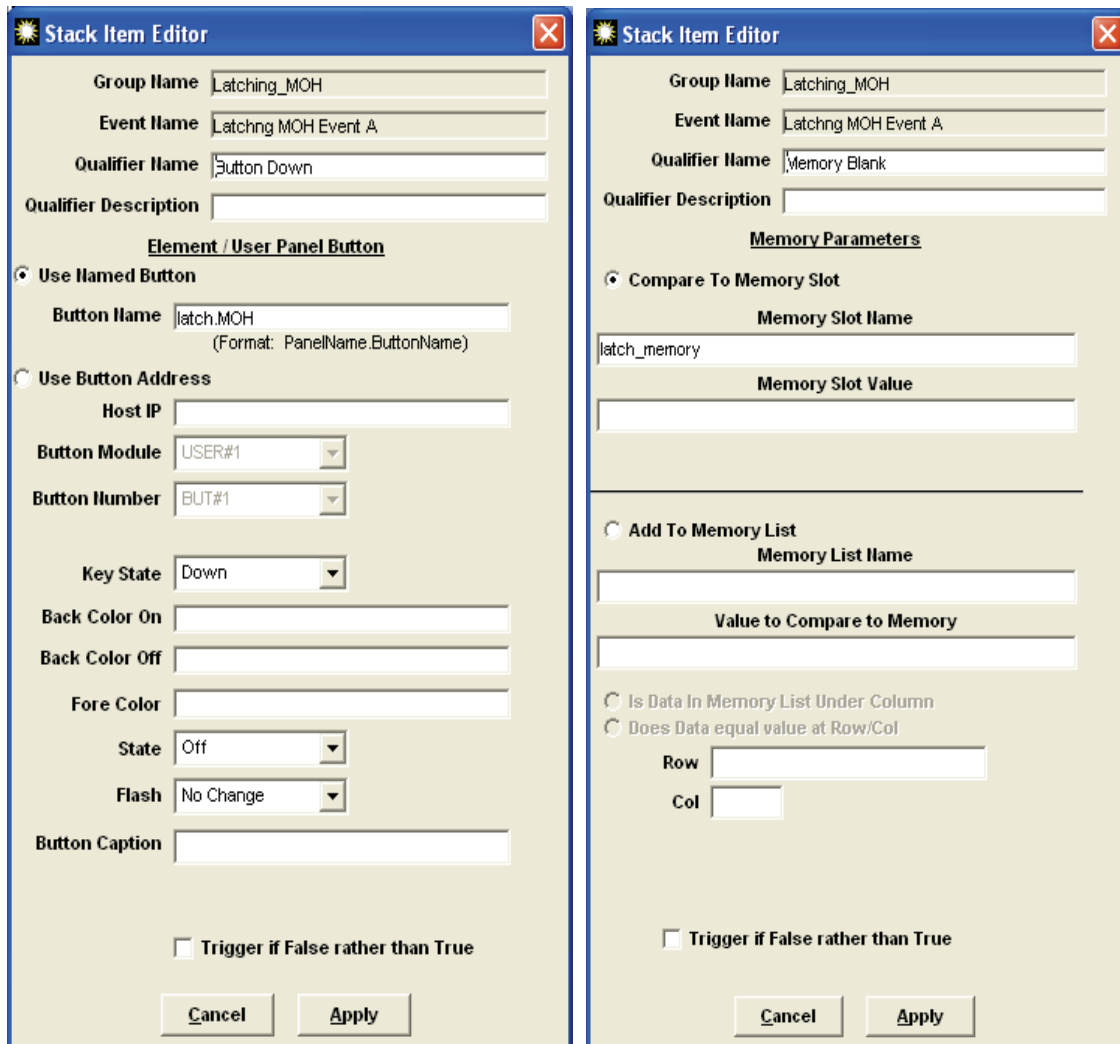
Stacking Event A will run if the latching button state is OFF and it looks like this:



In this case, the Stacking Event has two qualifiers. The first one will determine whether or not the button is pressed and check to see if the button state is OFF. That qualifier looks like the example shown to the left, below.

The second qualifier is simply a check to make sure that there is no memory value. We have decided to use a single memory location for this and we have named it "latch_memory"

The memory qualifier looks like the example shown to the right, below.



Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event A
 Qualifier Name: Button Down
 Qualifier Description:
Element / User Panel Button

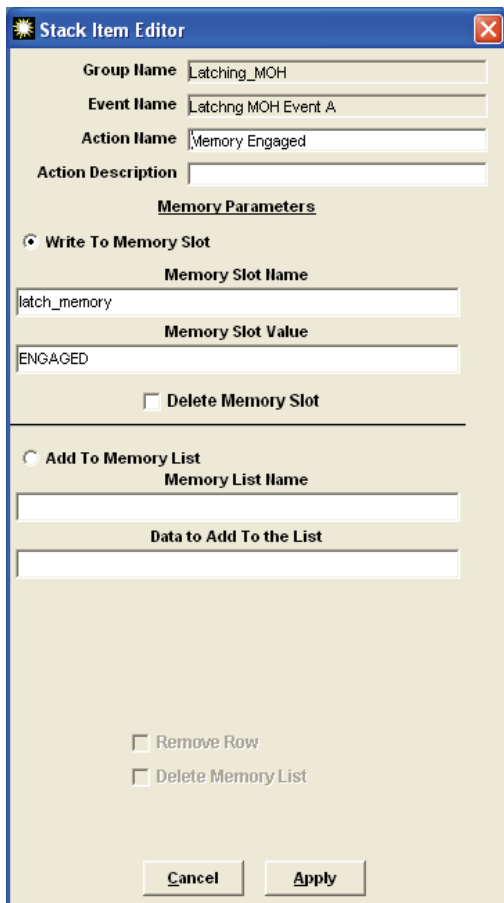
Use Named Button
 Button Name: latch.MOH
(Format: PanelName.ButtonName)

Use Button Address
 Host IP:
 Button Module: USER#1
 Button Number: BUT#1
 Key State: Down
 Back Color On:
 Back Color Off:
 Fore Color:
 State: Off
 Flash: No Change
 Button Caption:
 Trigger if False rather than True
 Cancel Apply

Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event A
 Qualifier Name: Memory Blank
 Qualifier Description:
Memory Parameters

Compare To Memory Slot
 Memory Slot Name: latch_memory
 Memory Slot Value:
 Add To Memory List
 Memory List Name:
 Value to Compare to Memory:
 Is Data In Memory List Under Column
 Does Data equal value at Row/Col
 Row:
 Col:
 Trigger if False rather than True
 Cancel Apply



Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event A
 Action Name: Memory Engaged
 Action Description:

Memory Parameters

Write To Memory Slot

Memory Slot Name: latch_memory
 Memory Slot Value: ENGAGED
 Delete Memory Slot

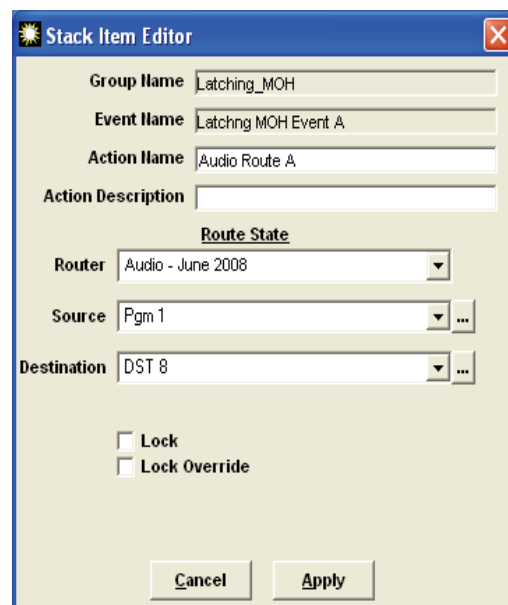
Add To Memory List

Memory List Name:
 Data to Add To the List:
 Remove Row
 Delete Memory List

Cancel Apply

Now for the actions of Event A. First we will write something to memory - in this example, we write ENGAGED, as shown in the example to the left.

Then, we will establish an audio route. In this case, we will route PGM 1 to destination 8 of a microphone node, as shown in the example on the right.



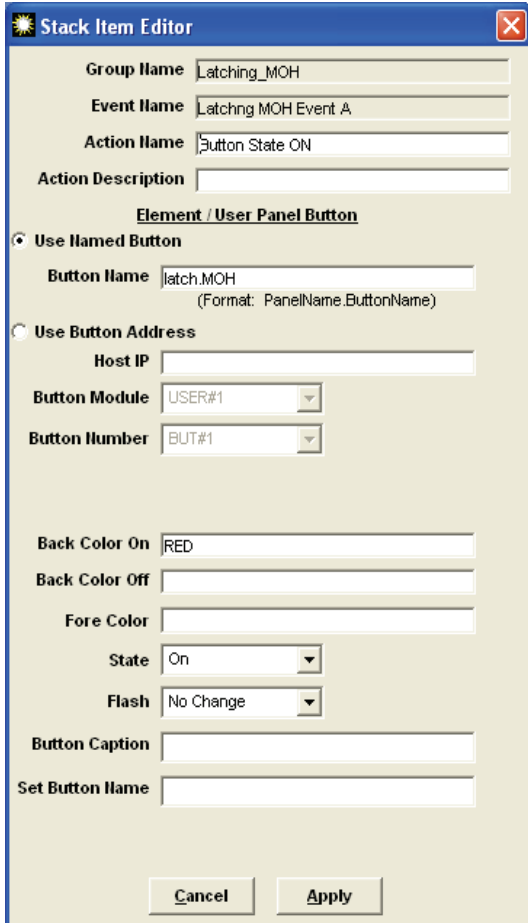
Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event A
 Action Name: Audio Route A
 Action Description:

Route State

Router: Audio - June 2008
 Source: Pgm 1
 Destination: DST 8
 Lock
 Lock Override

Cancel Apply



Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event A
 Action Name: Button State ON
 Action Description:

Element / User Panel Button

Use Named Button
 Button Name: latch.MOH
 (Format: PanelName.ButtonName)

Use Button Address
 Host IP:
 Button Module: USER#1
 Button Number: BUT#1

Back Color On: RED
 Back Color Off:
 Fore Color:
 State: On
 Flash: No Change

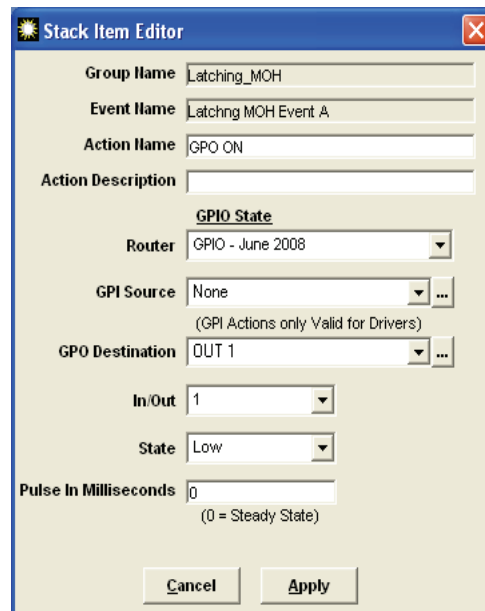
Button Caption:
 Set Button Name:

Cancel Apply

Next - change the button state to ON and give it some color - RED in this example.

Just to illustrate another option, we have decided that we will also change the state of a GPO to ON and this will also reflect our button state.

We've just created Stacking Event A! Now let's move on to Stacking Event B.



Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event A
 Action Name: GPO ON
 Action Description:

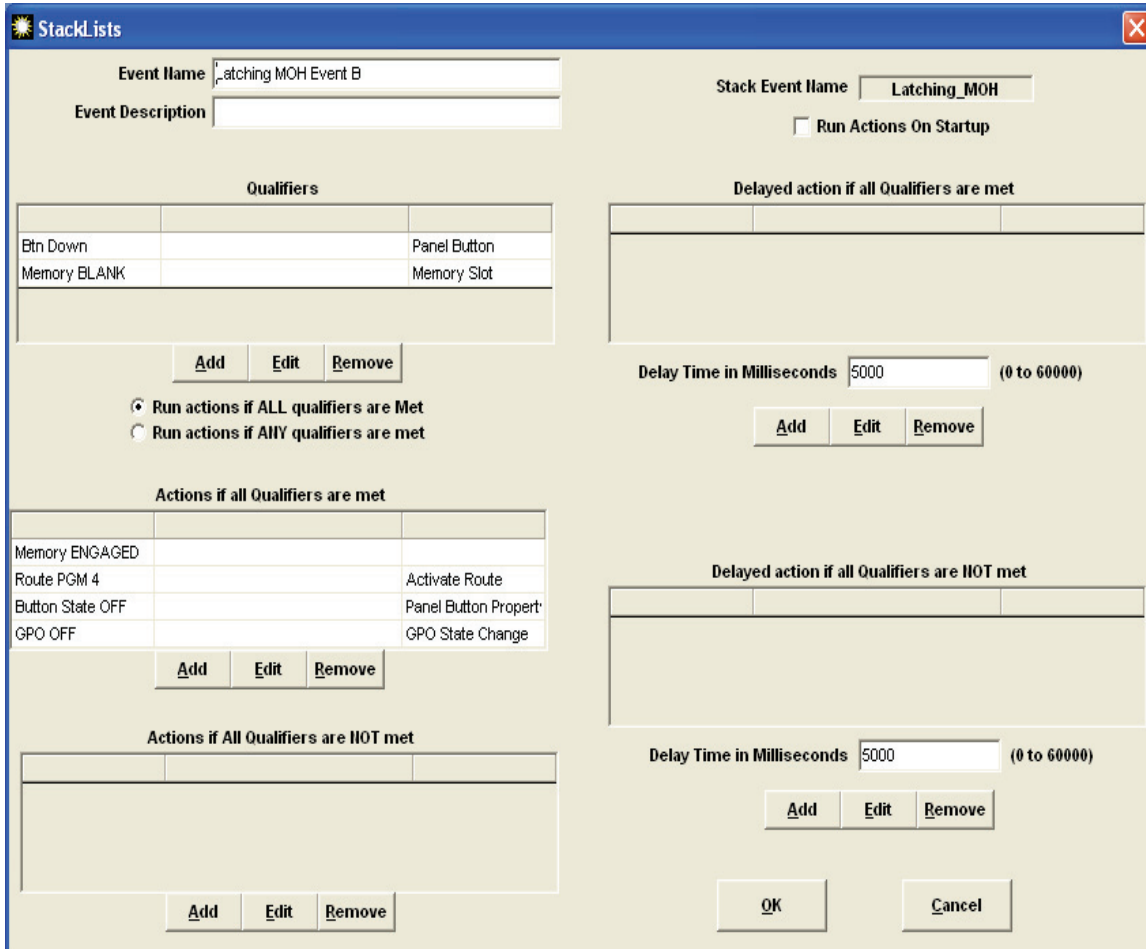
GPIO State

Router: GPIO - June 2008
 GPI Source: None
 (GPI Actions only Valid for Drivers)
 GPO Destination: OUT 1

In/Out: 1
 State: Low
 Pulse In Milliseconds: 0
 (0 = Steady State)

Cancel Apply

Now we will create another stacking event called Event B to create a set of different actions if the latching switch state is already ON. It will look like this.



The screenshot shows the 'StackLists' configuration window for an event named 'Latching MOH Event B'. The window is divided into several sections for configuring event logic and actions.

Event Name: Latching MOH Event B
Event Description: (Empty field)

Qualifiers:

| | |
|--------------|--------------|
| Btn Down | Panel Button |
| Memory BLANK | Memory Slot |

Buttons: Add, Edit, Remove

Logic:
 Run actions if ALL qualifiers are Met
 Run actions if ANY qualifiers are met

Actions if all Qualifiers are met:

| | |
|------------------|----------------------|
| Memory ENGAGED | |
| Route PGM 4 | Activate Route |
| Button State OFF | Panel Button Propert |
| GPO OFF | GPO State Change |

Buttons: Add, Edit, Remove

Actions if All Qualifiers are NOT met:

| | |
|--|--|
| | |
|--|--|

Buttons: Add, Edit, Remove

Stack Event Name: Latching_MOH
 Run Actions On Startup

Delayed action if all Qualifiers are met:

| | |
|--|--|
| | |
|--|--|

Delay Time in Milliseconds: 5000 (0 to 60000)
Buttons: Add, Edit, Remove

Delayed action if all Qualifiers are NOT met:

| | |
|--|--|
| | |
|--|--|

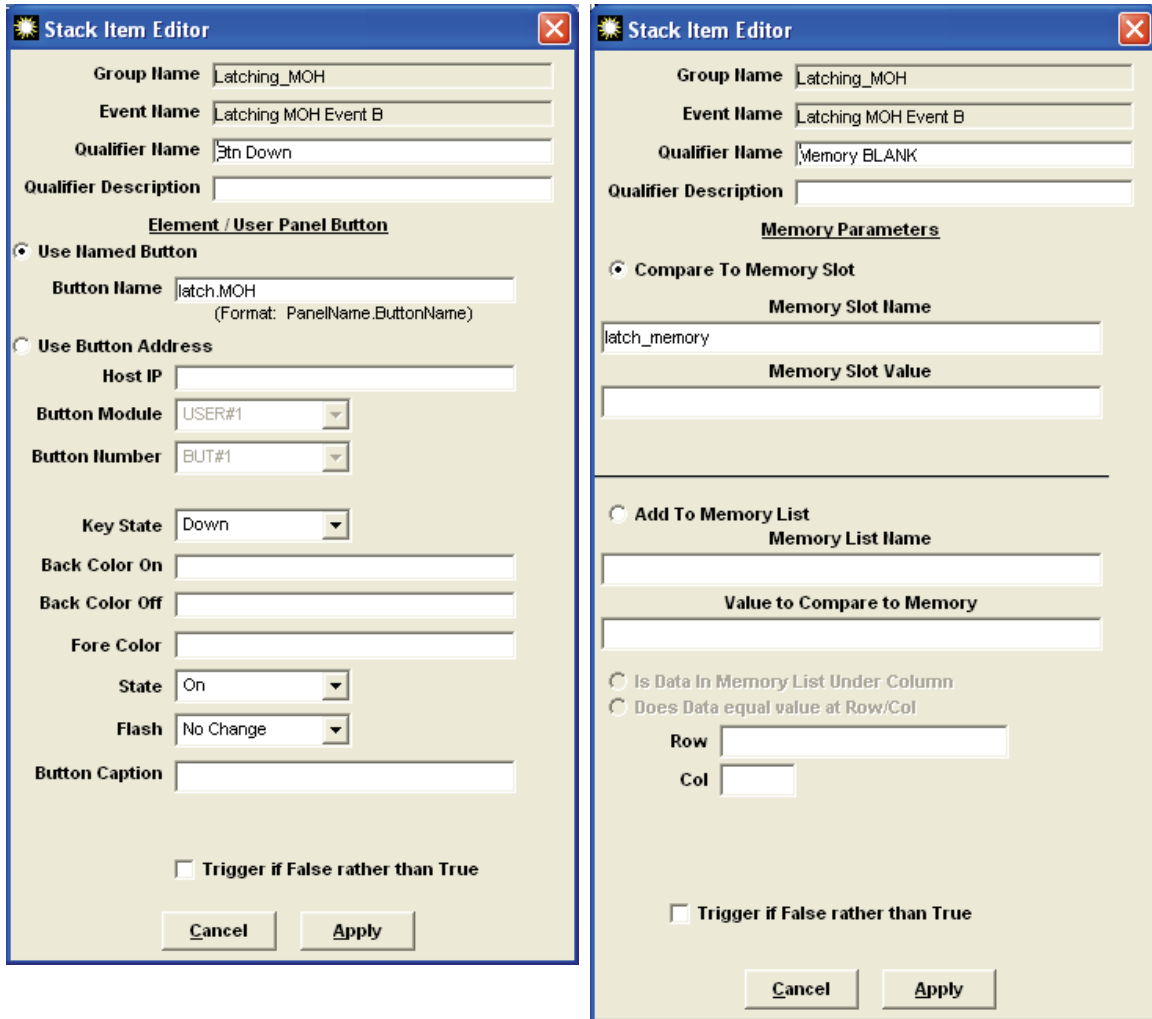
Delay Time in Milliseconds: 5000 (0 to 60000)
Buttons: Add, Edit, Remove

Buttons: OK, Cancel

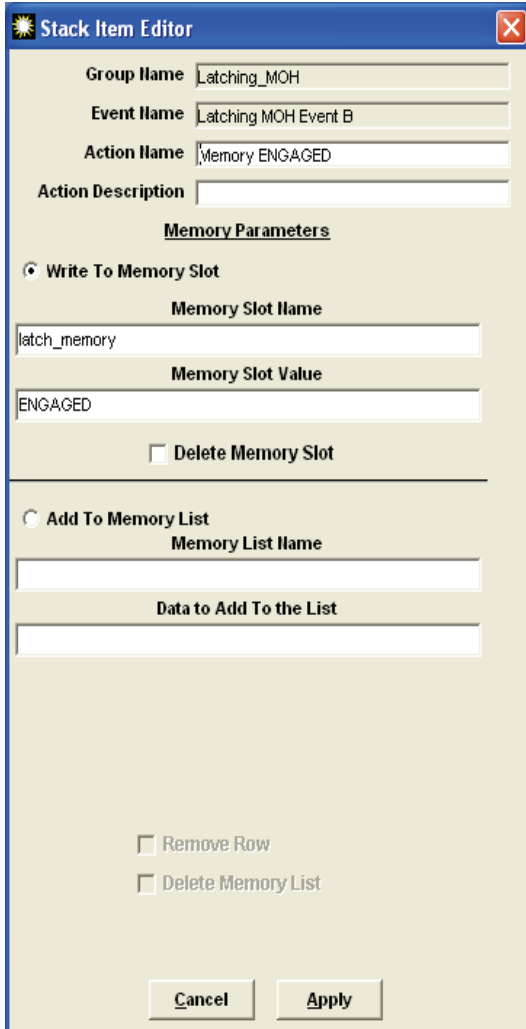
Event B has very similar qualifiers and actions to Event A which we just created. The differences are that this event runs if the switch state is ON and it will create a different audio route.

Below left is the Button Down qualifier for Stacking Event B that checks for the button press and the button state of ON.

Then, a memory check, identical to the one in Event A, takes place to make sure there is no memory value, as shown below right.



Now, we'll create our Actions.



Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event B
 Action Name: Memory ENGAGED
 Action Description:

Memory Parameters

Write To Memory Slot

Memory Slot Name: latch_memory
 Memory Slot Value: ENGAGED

Delete Memory Slot

Add To Memory List

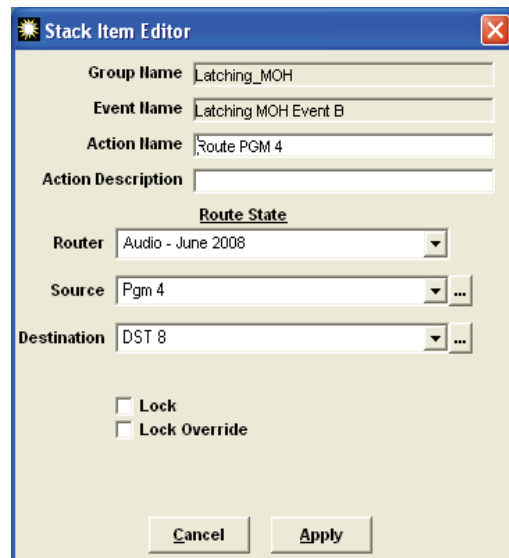
Memory List Name:
 Data to Add To the List:

Remove Row
 Delete Memory List

Cancel Apply

First, as with Event A, we will write a memory value.

Then we will establish an audio route. In this example, we are routing PGM 4 to destination 8 of the microphone node which changes the route we had established in Event A.



Stack Item Editor

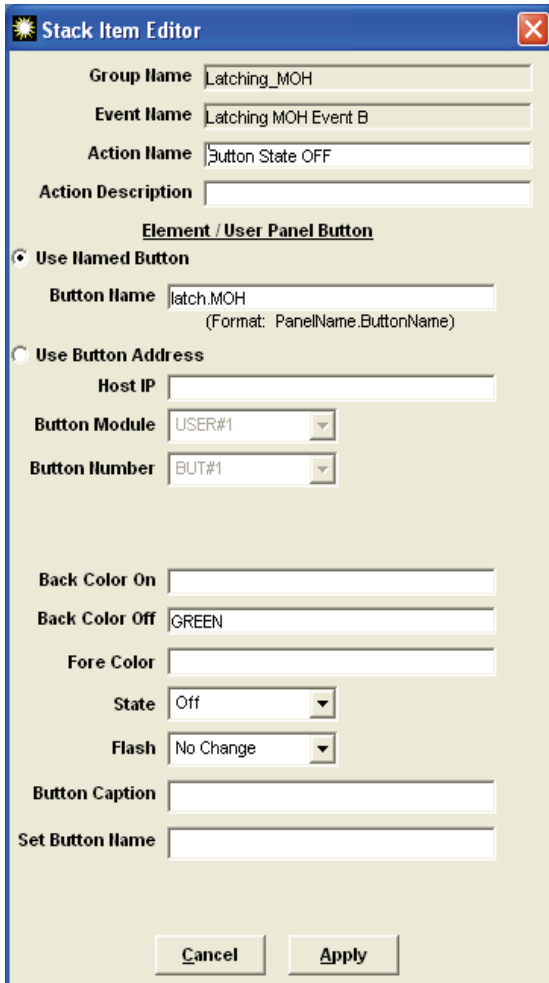
Group Name: Latching_MOH
 Event Name: Latching MOH Event B
 Action Name: Route PGM 4
 Action Description:

Route State

Router: Audio - June 2008
 Source: Pgm 4
 Destination: DST 8

Lock
 Lock Override

Cancel Apply



Stack Item Editor

Group Name: Latching_MOH

Event Name: Latching MOH Event B

Action Name: Button State OFF

Action Description:

Element / User Panel Button

Use Named Button

Button Name: latch.MOH
(Format: PanelName.ButtonName)

Use Button Address

Host IP:

Button Module: USER#1

Button Number: BUT#1

Back Color On:

Back Color Off: GREEN

Fore Color:

State: Off

Flash: No Change

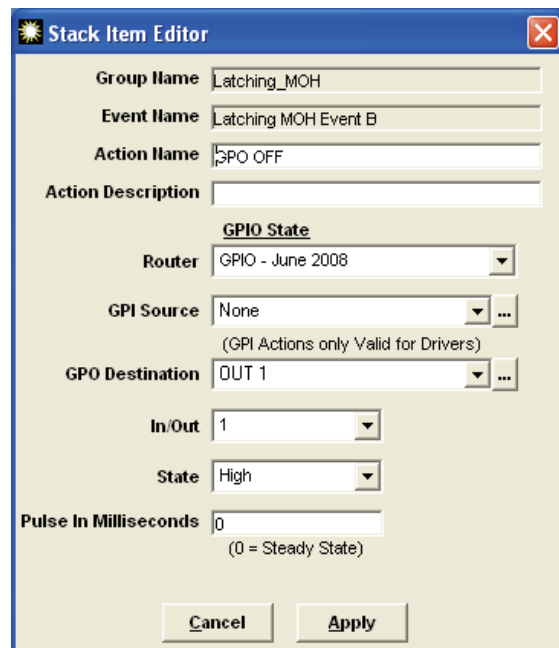
Button Caption:

Set Button Name:

Buttons: Cancel, Apply

Next, we will change the button state. We checked to see if it was ON before we ran this operation, and now we will set it to OFF plus change our background color to GREEN.

Lastly, we will change the GPO state to OFF to reflect our button state.



Stack Item Editor

Group Name: Latching_MOH

Event Name: Latching MOH Event B

Action Name: GPO OFF

Action Description:

GPIO State

Router: GPIO - June 2008

GPI Source: None

(GPI Actions only Valid for Drivers)

GPO Destination: OUT 1

In/Out: 1

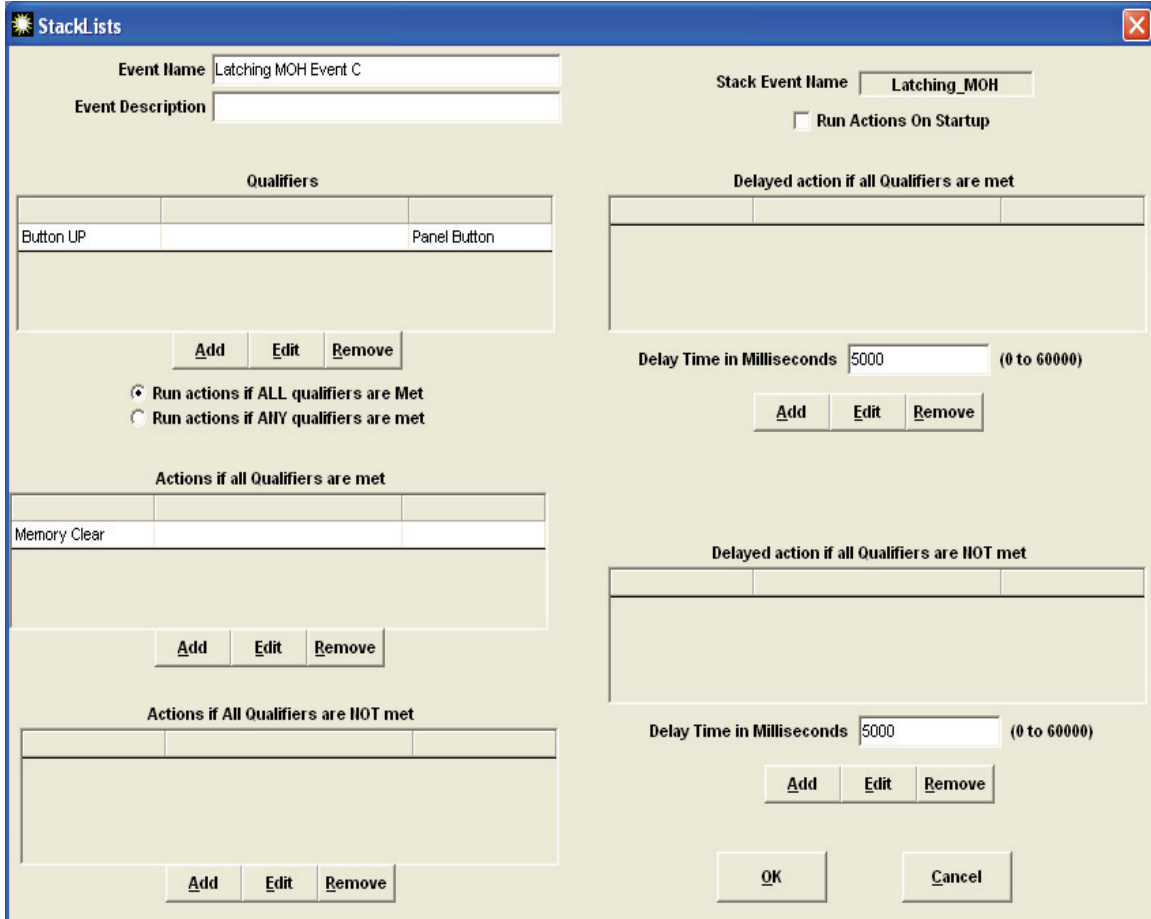
State: High

Pulse In Milliseconds: 0
(0 = Steady State)

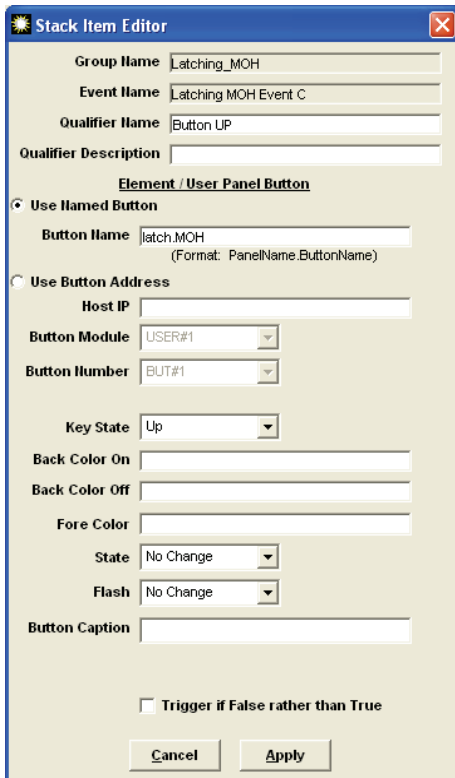
Buttons: Cancel, Apply

So far, so good! We have determined what happens if a button is pressed. Of course, most of the time the button will not be pressed so we also need to determine what happens in that case. So, we'll need to construct a Stacking Event C.

Stacking Event C is a very simple event that just clears the memory value whenever the button is not pressed. It will look something like this:



The screenshot shows the 'StackLists' application window. The 'Event Name' is 'Latching MOH Event C' and the 'Stack Event Name' is 'Latching_MOH'. The 'Event Description' field is empty. Under 'Qualifiers', there is one entry: 'Button UP' with 'Panel Button' in the adjacent field. Below this is a table for 'Actions if all Qualifiers are met' with one entry: 'Memory Clear'. The 'Run actions if ALL qualifiers are Met' radio button is selected. On the right side, there are two sections for 'Delayed action if all Qualifiers are met' and 'Delayed action if all Qualifiers are NOT met', both with a 'Delay Time in Milliseconds' of 5000. The 'Run Actions On Startup' checkbox is unchecked. 'Add', 'Edit', and 'Remove' buttons are present for each table. 'OK' and 'Cancel' buttons are at the bottom right.



Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event C
 Qualifier Name: Button UP
 Qualifier Description:

Element / User Panel Button

Use Named Button
 Button Name: latch_MOH
 (Format: PanelName ButtonName)

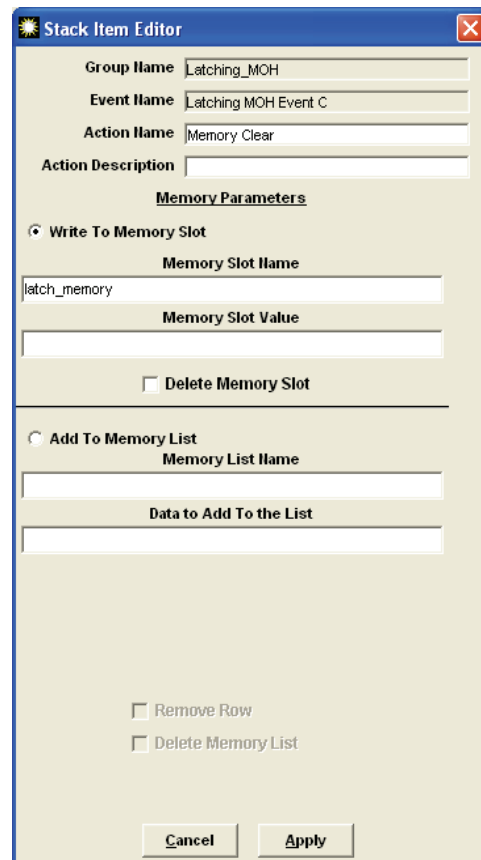
Use Button Address
 Host IP:
 Button Module: USER#1
 Button Number: BUT#1
 Key State: Up
 Back Color On:
 Back Color Off:
 Fore Color:
 State: No Change
 Flash: No Change
 Button Caption:

Trigger if False rather than True

Cancel Apply

Here is the qualifier looking for a button state of UP...

...and here is the action of clearing the memory.



Stack Item Editor

Group Name: Latching_MOH
 Event Name: Latching MOH Event C
 Action Name: Memory Clear
 Action Description:

Memory Parameters

Write To Memory Slot
 Memory Slot Name: latch_memory
 Memory Slot Value:
 Delete Memory Slot

Add To Memory List
 Memory List Name:
 Data to Add To the List:

Remove Row
 Delete Memory List

Cancel Apply

Just to prove this actually works, here are the results illustrating the two states of the latching mini panel button. Of course, you can get a lot prettier than this example. We want to show you the basics so you can use the example as a guide for more complex cases.

Here is Button ON state:



Here is Button OFF state:



Conclusion

There are lots of variations to the Latching Button theme. For instance, we could use hardware buttons on user panels or GPI as qualifiers. The resulting actions can be anything you want. The important things to remember are:

1. For a latching button you should use three stacking events
2. You must write a memory value
3. You must pay attention to the order of the qualifiers and actions – specifically, write the memory value *first*.

See how easy that was? Have fun!

For questions or assistance, please contact Axia Support at Support@AxiaAudio.com.