Exercise 2-1: UI Event Handler Design Pattern

Goal

- Develop an application using the User Interface Event Handler design pattern.

Hardware Setup

(Hardware) In the exercises where we work with Analog Input/Output channels, we use PCI-6221/USB-6212 multifunction I/O device paired with the BNC-2120 shielded connector block. Analog Input 2 should be connected to the Sine/Triangle BNC connector. Analog Input 3 should be connected to the TTL Square Wave BNC connector. The Sine/Triangle waveform switch should be set to Sine.

Scenario

You want to create an application that has a front panel with multiple buttons. Clicking **each** button should trigger specific code to execute. You will use the User Interface Event Handler design pattern to implement this application.

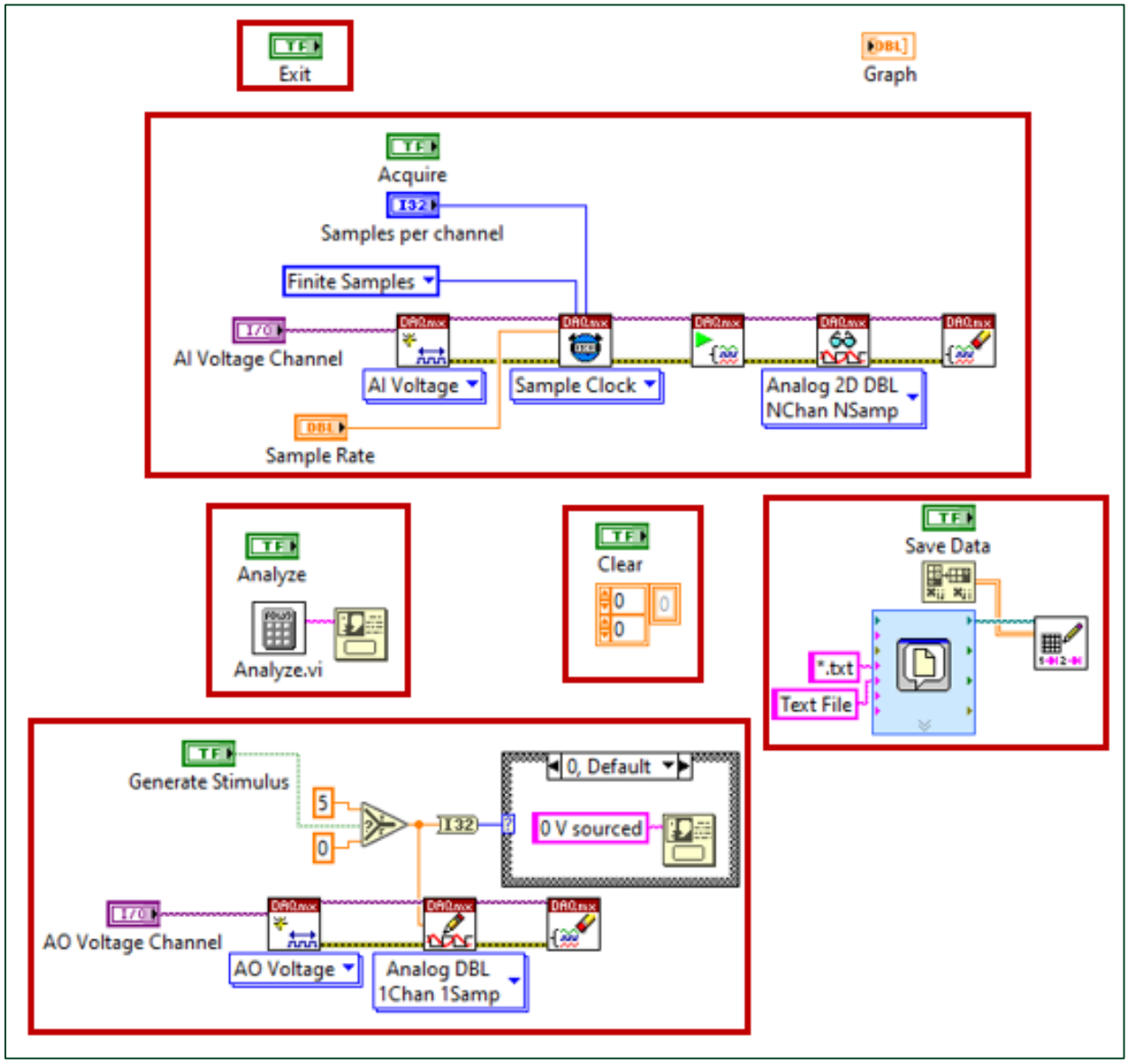
The following table lists the events you will implement in the User Interface Event Handler VI you create.

Event	Event Description
“Exit”: Value Change	Stops the While Loop
“Acquire”: Value Change	Acquires a finite acquisition
“Analyze”: Value Change	Analyzes data
“Save Data”: Value Change	Logs data to a text file
“Clear”: Value Change	Clears the graph data.
“Generate Stimulus”: Value Change	Updates the Analog Output channel signal.

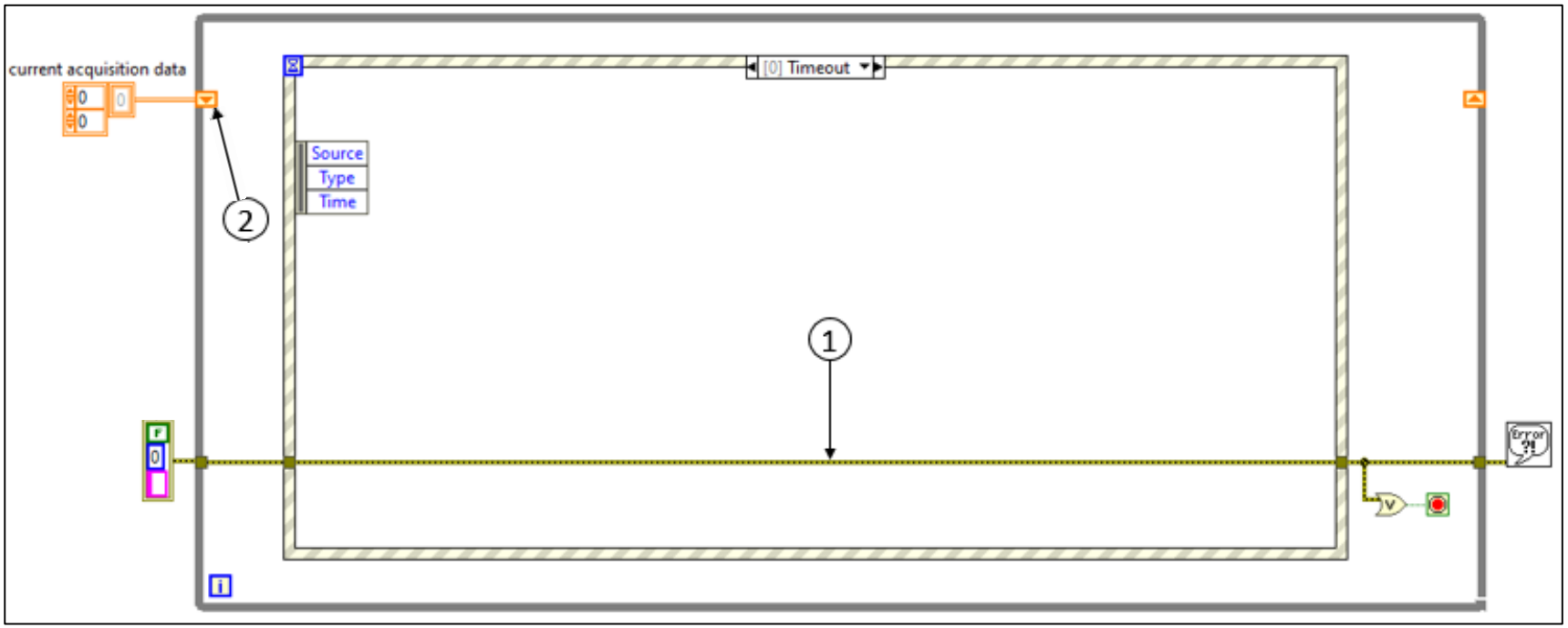
Guided Instruction

1. Open `C:\Exercises\LabVIEW Core 2\User Interface Event Handler\User Interface Event Handler.lvproj`.
2. From the **Project Explorer** window, open the User Interface Event Handler VI. Notice that the front panel has already been created for you. In this application, clicking each **Boolean** button should trigger its corresponding code to execute.

3. Explore the block diagram.
   - Notice that the block diagram contains Boolean button terminal next to the code that corresponds to each Boolean button.

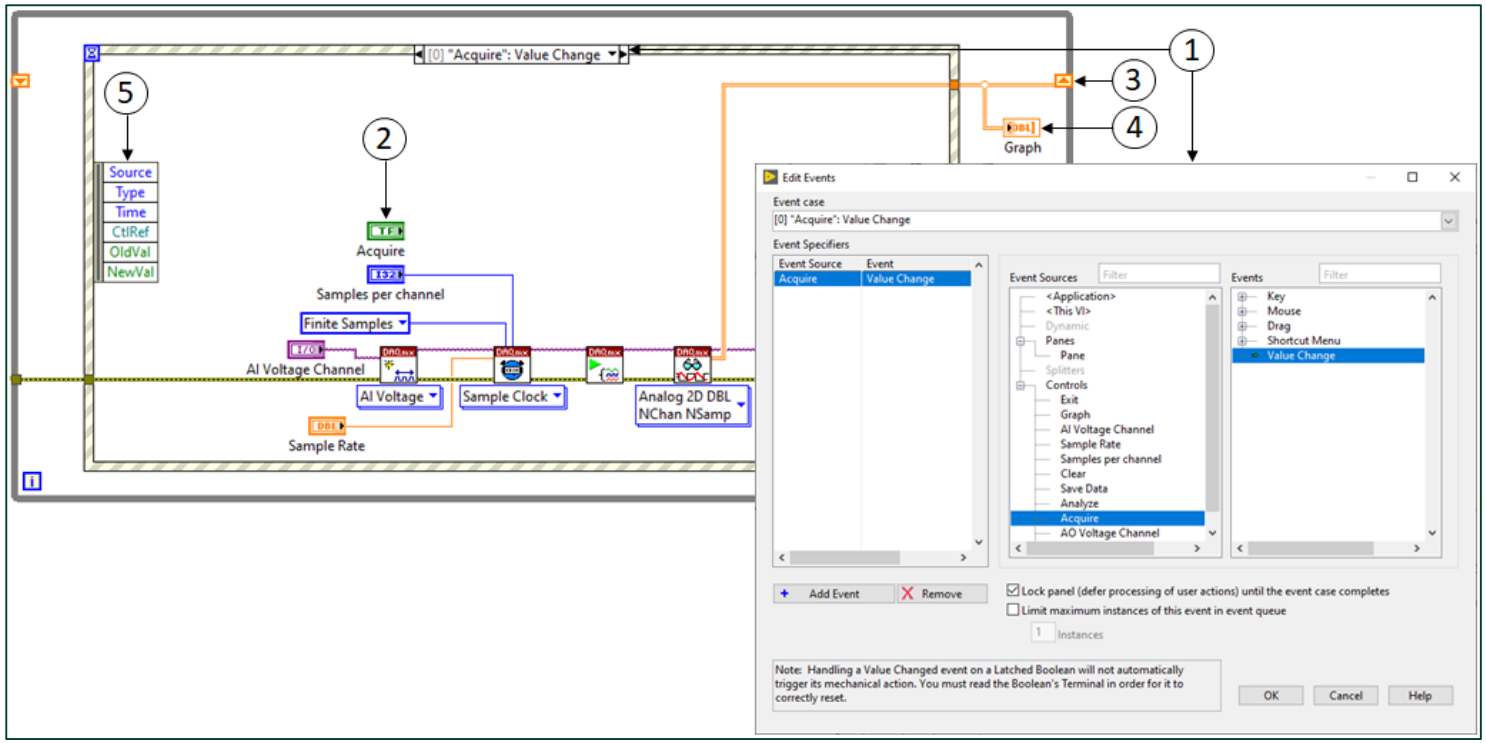


4. Locate the **While Loop** already on the block diagram. Place an **Event** structure inside the **While Loop**.
5. Wire the error wire through, as shown in the following figure.



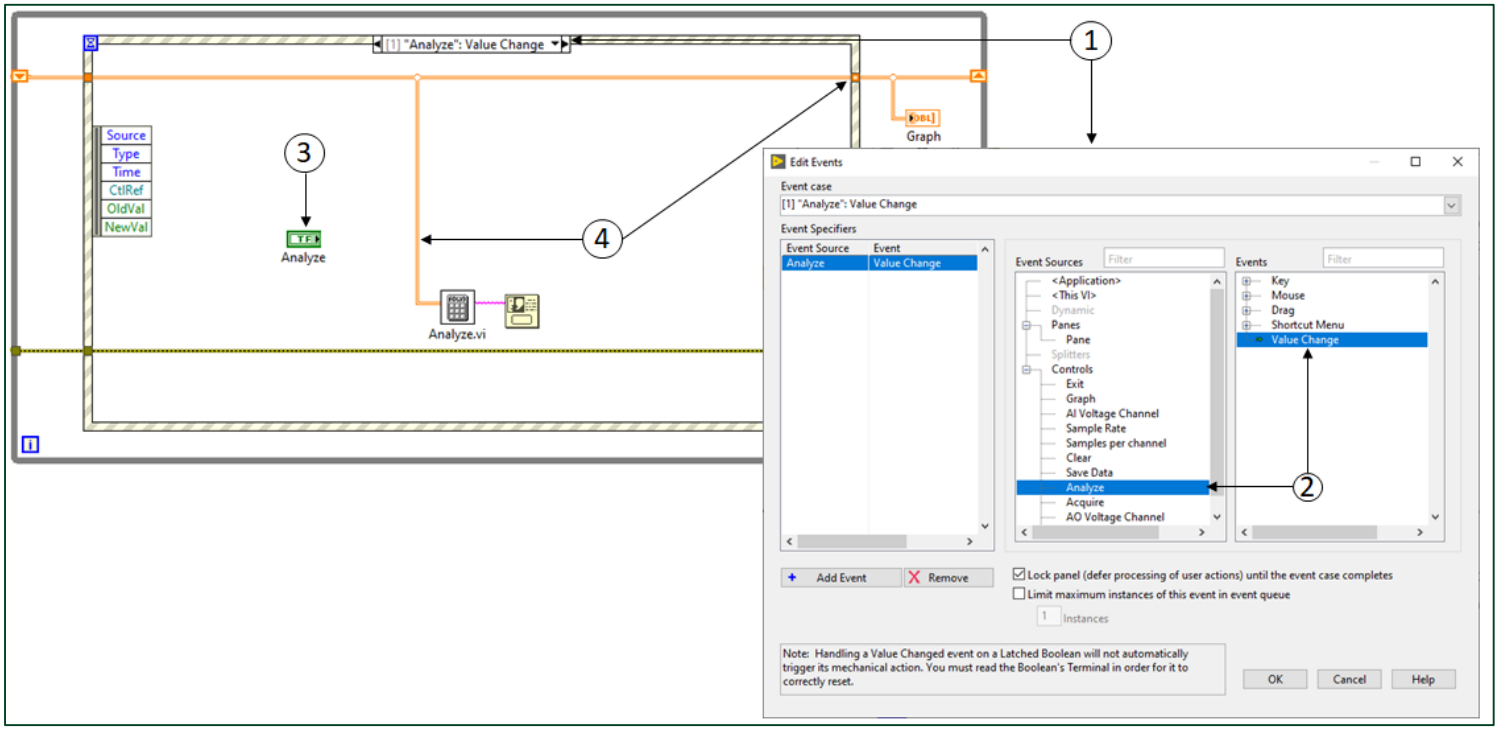
1. **Error Wire** – Wire the error wire through the Event structure so that it creates input and output tunnels.
2. **Shift register** – Create shift register. Storing the acquired data in the shift register allows multiple event cases to update and read the current acquisition data.

6. Configure the Event structure to handle the **“Acquire”: Value Change** event.



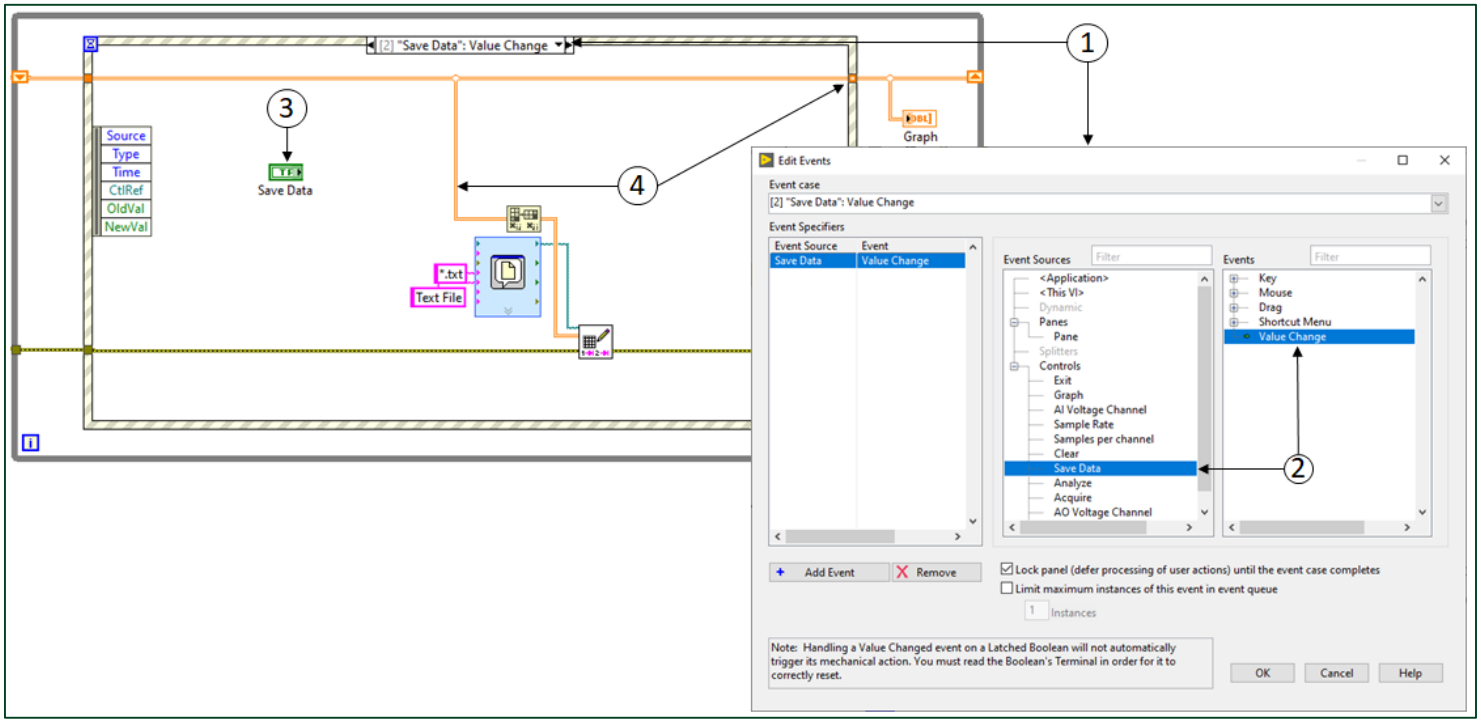
1. **“Acquire”: Value Change event** – Right-click the **Event structure border**, then click Edit **Events Handled by This Case**. In the Event Sources section of the Edit Events window, click the **Controls»Acquire**, and in the Events section select **Value Change**.
2. **Acquire control and corresponding code** – Move this control and code inside the event case.
3. **Shift register** – Wire the data output of the DAQmx Read VI into the shift register. This updates the current acquisition data stored in the shift register. You will update and read this data in other event cases.
4. **Graph indicator** – Move and wire this indicator as shown. Rewire the error wires as shown.
5. **Event Data Node** – Identifies the data that the Event structure returns when an event occurs. You can resize the node to display only one or multiple elements.

7. Configure the Event structure to handle the **“Analyze”: Value Change** event.



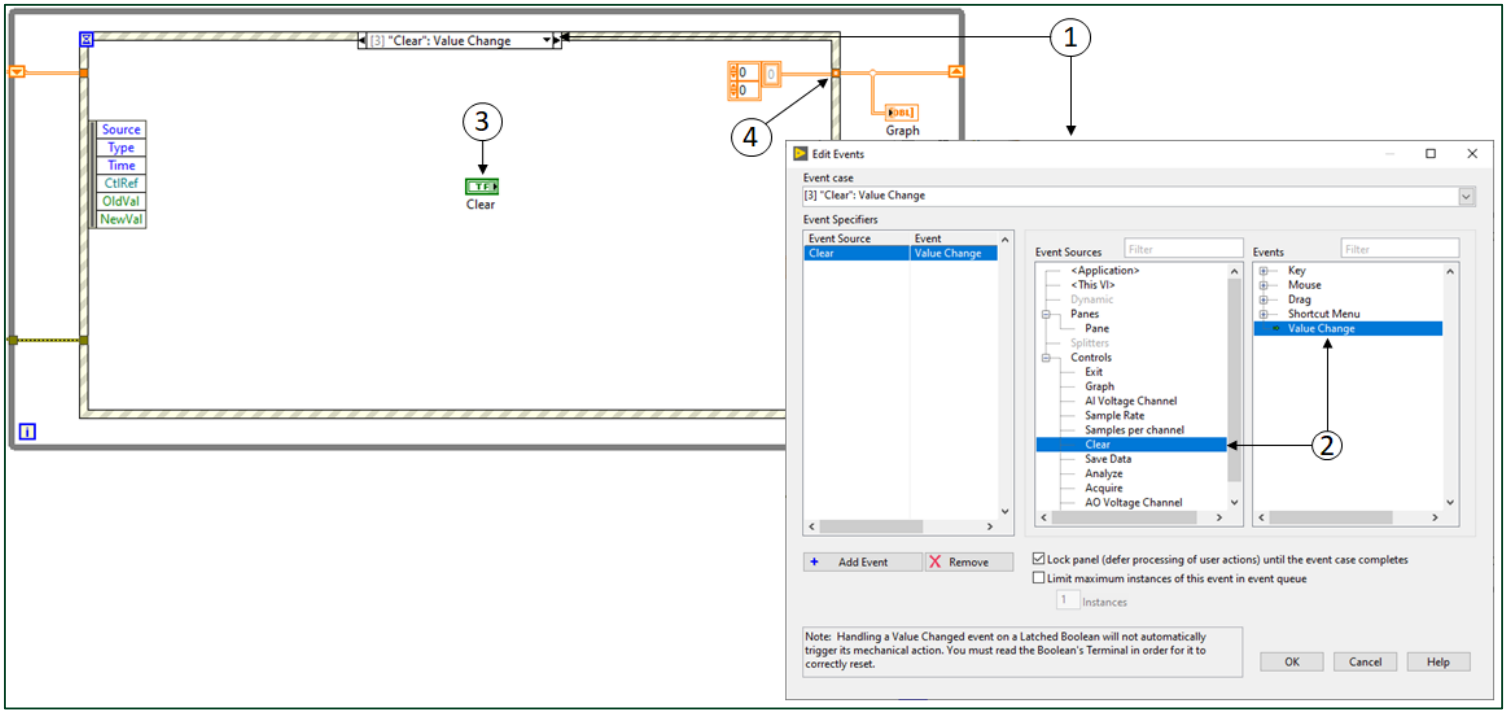
1. **“Analyze”: Value Change event** – Right-click the **Event structure border** and select **Add Event Case**.
2. **Edit Events window** – In the Event Sources section of the Edit Events window, click the **Controls»Analyze**, and in the Events section select **Value Change**.
3. **Analyze control and corresponding code** – Move this control and code inside the event case.
4. **Shift register** – Wire the shift register data to the *N*-Channel Data input of the Analyze subVI to read the current acquisition data. Wire the shift register data through the event case to store the data back in the shift register.

8. Configure the Event structure to handle the **“Save Data”: Value Change** event.



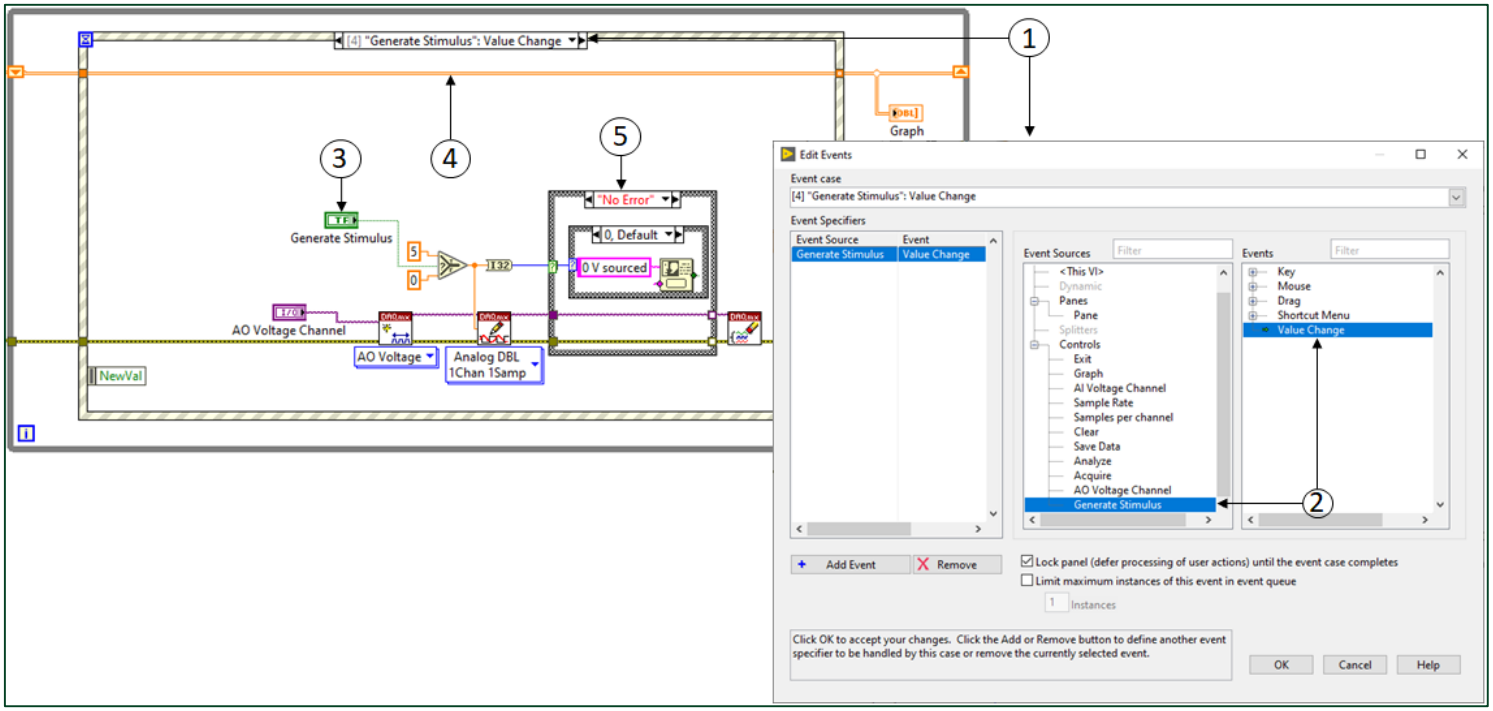
1. **“Save Data”: Value Change event** – Right-click the **Event structure border** and select **Add Event Case.**
2. **Edit Events window** – In the Event Sources section of the Edit Events window, click the **Controls»Save Data**, and in the Events section select **Value Change**.
3. **Save Data control and corresponding code** – Move this control and code inside the event case.
4. **Shift register** – Wire the shift register data to the input of the Transpose 2D Array function to read the current acquisition data. Wire the shift register data through the event case to store the data back in the shift register.

9. Configure the Event structure to handle the **“Clear”: Value Change** event.



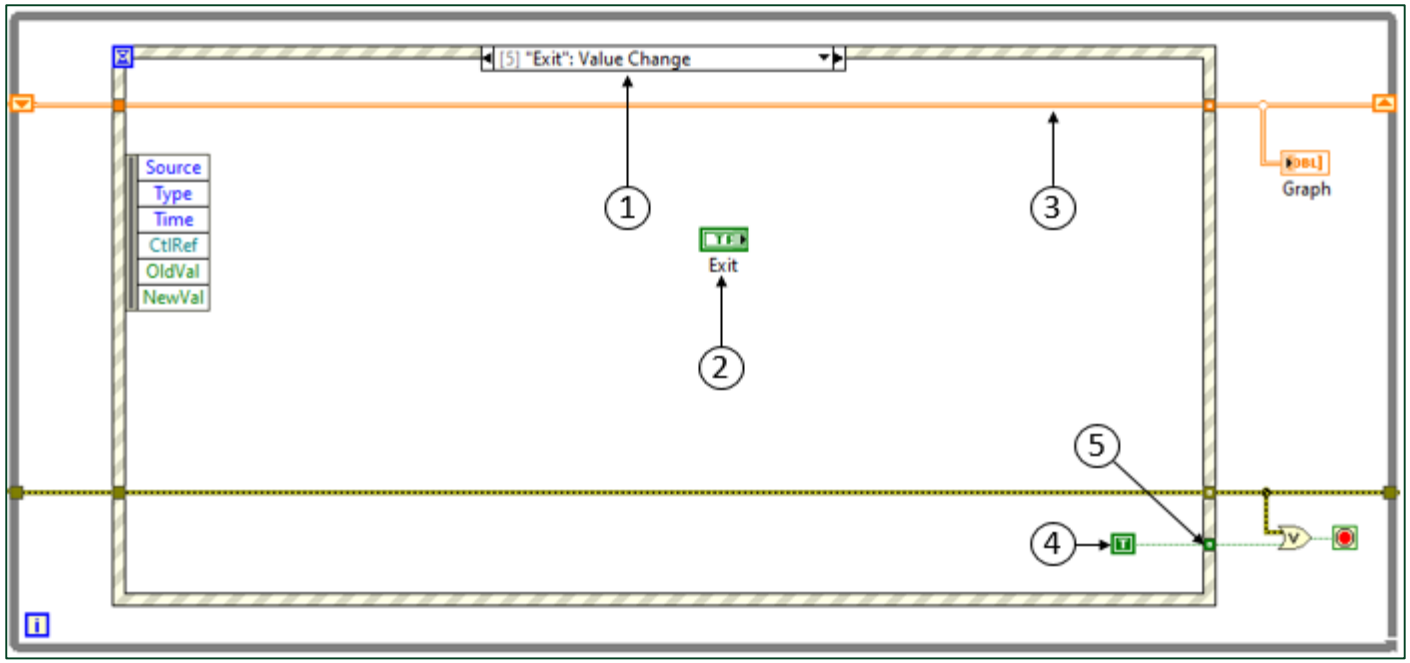
1. **“Clear”: Value Change event** – Right-click the **Event structure border** and select **Add Event Case.**
2. **Edit Events window** – In the Event Sources section of the Edit Events window, click the **Controls»Clear**, and in the Events section select **Value Change**.
3. **Clear control and corresponding code** – Move this control and code inside the event case.
4. **Shift register** – Wire the empty 2D DBL array constant to the shift register. This clears the stored acquisition data in the shift register by replacing it with empty data.

10. Configure the Event structure to handle the **“Generate Stimulus”: Value Change** event.



1. **“Generate Stimulus”: Value Change event** – Click the **Event structure border** and select **Add Event Case**.
2. **Edit Events window** – In the Event Sources section of the Edit Events window, click the **Controls»Generate Stimulus**, and in the Events section select **Value Change**.
3. **Generate Stimulus control and corresponding code** – Move this control and code inside the event case.
4. **Wire the tunnels** – This case does not update or read the acquisition data.
5. **Error Case structure** – Wrap the case structure containing the dialog boxes in another one and connect error wire to its conditional terminal, then connect error and DAQmx task wires throughout the case structure in the **Error** and **No Error** cases. We do this to prevent the dialog box from appearing in case of a VI error.

11. Configure the Event structure to handle the Exit event.



1. **"Exit": Value Change event** – Right-click the **Event structure border** and select **Add Event Case**. In the Event Sources section of the Edit Events window, click the **Controls»Exit** and in the Events section select **Value Change**.
2. **Exit control** – Move this control inside the event case.
3. **Wire the tunnels** –This case does not update or read the acquisition data.
4. **True constant** – This causes the Exit event case to stop the While Loop.
5. **Event Tunnel** – Wire the True constant through an event tunnel to the OR function. Make sure that when right-clicking the **tunnel**, the **Use Default If Unwired** option is enabled, which means that any case with no constant wired defaults to False.

12. Save the VI.

Test the VI

1. Run the VI.
2. Set the following control values.

AI Voltage Channel	PCI 6221/ai0, PCI 6221/ai3
Sample Rate (Hz)	1000
Samples per channel	100
AI Voltage Channel	PCI 6221/ao0

3. Click the **Acquire** button. Verify that acquisition data appears on the Graph.
4. Click the **Analyze** button. Verify that a dialog window appears showing analysis data.
5. Click the **Save Data** button. Verify that a file dialog appears and that the log file is created.
6. Click the **Generate Stimulus** button. Verify that a file dialog appears and that the Analog Output channel signal is updated.
7. Click the **Clear** button. Verify that the VI clears the Graph.
8. Click the **Exit** button. Verify that the VI exits.

Your Turn

1. Add a new control.
2. Configure an event for the control.
3. Add code that executes each time the event occurs.
4. Test if your new code executes as expected.

On the Job

Do you have any UI event-based applications? If so, describe the desired event-based functionality for the application.

End of Exercise 2-1