Exercise 5-2: Handle Specific Errors Immediately

Goal

Handle a specific error immediately by asking the user to retry an action.

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.

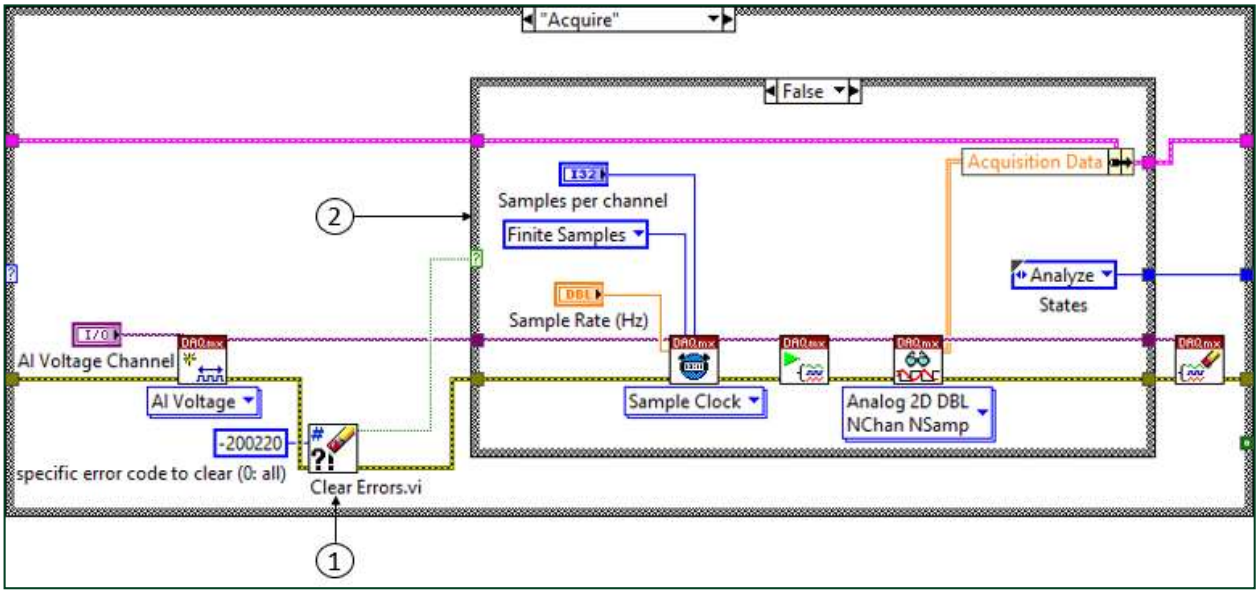
Scenario:

In Exercise5-1, you tested how this application responds to the user entering an invalid DAQmx channel. The application manages it by displaying an error dialog and exiting.
In this exercise, you want to define how the application handles this error.
The application will display a “AI Voltage Channel is invalid. Please enter valid channel(s)” dialog box to the user and then allow the user to re-enter a different channel.

Instructions:

Handling an Error Locally:

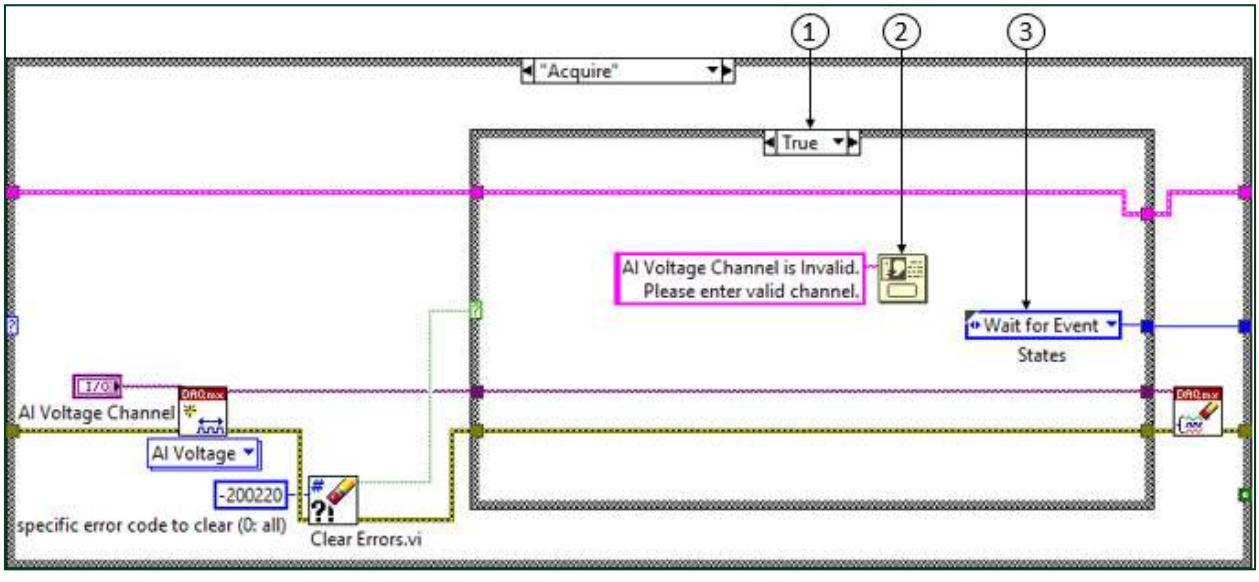
1. Open `C:\Exercises\LabVIEW Core2\Handle Error Locally\Handle Error Locally.lvproj`.
2. From the **Project Explorer** window, open the Event-Driven State Machine – Handle Error Locally VI.
3. On the block diagram, go to the Acquire case of the state diagram.
4. Modify the block diagram as shown in the following figure, to perform a finite acquisition if the DAQmx Create Channel VI does not output Error -200220.



1. **Clear Errors VI** – Clears a specific error or warning from the error wire if it matches a code you specify. Use the context help to learn more about this VI.
2. **Case Structure** – If the Clear Errors VI did not find error -200220 in the error wire, then perform a finite acquisition.

Note: To make the current case the False case, right-click the **border of the Case structure** and select **Make This Case False.**

5. Modify the block diagram as shown in the following figure, to locally handle Error -200220 by



1. **Case Structure** – Select the **True case**.
2. **One Button Dialog** – Display a dialog to the user to specify a valid channel. Right-click the message input and select **Create» Constant**. Set the constant to AI Voltage Channel is invalid. Please enter valid channel.
3. **States Enum Constant (type definition)** – right-click the **blue enum output tunnel** on the Case structure and select **Create» Constant**. Set the constant to Wait for Event. If Error -200220 occurs, there is no need to go to the Analyze state, so the application should go back to the Wait for Event state instead.

6. Examine the new error handling behavior of the application.
   - Run the VI.
   - Set the AI Voltage Channel control to a valid channel, such as `PCI-6221/ai0, PCI-6221/ai2:3`.
   - Click the **Acquire** button. The VI should run.
   - Double-click the **text** in the AI Voltage Channel control and set it to an invalid channel, such as `Undefined_PCI-6221/ai0:8`.
   - Click the **Acquire** button. This will cause an error in the Acquire case of the state machine because Undefined_PCI-6221/ai0:8 does not exist in your system.
   - Verify that the application handles this error by displaying a dialog to the user and allowing the user to enter the AI Voltage Channel again.

- Click the **Exit** button to stop the VI when finished.

7. Examine the error handling behavior of this application for an error that is not handled locally.
   - Run the VI.
   - Double-click the **text** in the AO Voltage Channel control and set it to an invalid channel, such as `Undefined_PCI-6221/ao1`.
   - Click the **Generate Stimulus** button.
   - This will cause an error in the Update Stimulus case of the state machine because Undefined_PCI-6221/ao1 does not exist in your system.

Preventing Incorrect Behavior

In the Analyze case in this application, the Analyze subVI calculates the RMS values of the acquired data and compares them to the **Threshold (rms)** control value. However, because the calculated RMS values will always be zero or positive values, it does not make sense for the **Threshold (rms)** control to pass in a negative value.

To prevent a potential user error or confusion, edit the **Threshold (rms)** control to have a minimum value of 0, to prevent the user from entering a negative value.

1. On the front panel, right-click the **Threshold (rms)** control and select **Properties**.
2. In the **Data Entry** tab, uncheck the **Use Default Limits** checkbox, then set **Minimum** to 0 and select Coerce under the **Response to value outside limits** drop-down list.
3. Test the new behavior.
   - Try setting the **Threshold (rms)** control to a negative value, such as -5.
   - Notice that the **Threshold (rms)** control coerces a negative value to 0.

Your Turn

1. Modify this VI to also handle an error of invalid AO Voltage Channel locally by displaying an instructive dialog of "`AO Voltage Channel is invalid. Please enter a valid analog output channel`" to the user.

   _____
   _____
   _____

2. To view the solution, refer to **[Your Turn Solution] Event-Driven State Machine - Handle Error Locally VI** in the `C:\Solutions\LabVIEW Core 2\Exercise 5-2` directory.

   _____
   _____
   _____

On the Job

1. Would any of your applications benefit from handling a specific error locally?

   _____
   _____

2. If so, what specific errors do you want to handle locally?

_____
_____

3. Describe how you would handle each of those errors (Ignore error, retry action, fix source of error, etc.).

   _____
   _____
   _____
   _____
   _____
   _____

End of Exercise 5-2