

Implementation

The files that you need to complete this exercise are here:

<NI eLearning>\LV Core 1\Wiring Debug\Exercise.

1. Open the Broken VI.vi from the <Exercise> directory.
2. This VI is not executable. Click the broken **Run** button to open the Error list window.
 - ☐ Enable the **Show Warnings** checkbox.
 - ☐ Double-click the item **Terminal: object is hidden** to highlight the cause of this warning. Notice that a tunnel is highlighted which appears to be the location of crossing wires.
 - ☐ Press the **Down Arrow** key on the keyboard to align the tunnel with the wire entering the function in the Case structure.
 - ☐ Click the broken **Run** button again to bring up the Error list window.
 - ☐ Click the **Show Error** button to highlight the cause of this error. Notice that a wire which is accidentally connecting objects of different data types is highlighted.
 - ☐ Press the **Delete** key on the keyboard to delete this wire.
3. Clean up the wires that do not follow straight lines and are confusing to read.
 - ☐ Select the tunnel that appears on the right side of the outer Case structure.
 - ☐ Drag this tunnel to the left side of the outer Case structure, approximately in line with the left output of the shift register, as shown in Figure 1.
 - ☐ Right-click the wire coming out of the left side of the shift register and select **Clean Up Wire**.
 - ☐ Now that the wire is routed properly, move the tunnel on the left side of the outer Case structure to exactly align with the output of the left side of the shift register.
 - ☐ Select the tunnel that appears on the top side of the outer Case structure.

- ☐ Drag this tunnel to the right side of the outer Case structure, approximately in line with the right input of the shift register, as shown in Figure 1.
 - ☐ Wire from this tunnel to the right input of the shift register.
 - ☐ Now that this tunnel is wired to the shift register, move it to exactly align with the right input of the shift register.
 - ☐ Wire this tunnel into the Running Result indicator and the Not a Number/Path/Refnum function.
 - ☐ Wire the output of the Not a Number/Path/Refnum function into the conditional terminal. Notice that the VI is no longer broken.
4. Run the VI.
- ☐ Click the **Run** button.
 - ☐ Enter a value of 5 into the X control.
 - ☐ Select the value **Add** from the **Action** control.
 - ☐ Click the **Calculate** button. Notice that the value in the Running Result indicator has not appeared to change. This VI is behaving unexpectedly.
5. Enable execution highlighting to see where an expected value is coming from.
- ☐ Click the **Highlight Execution** button.
 - ☐ On the front panel, click the **Calculate** button again, and then watch the code execute.

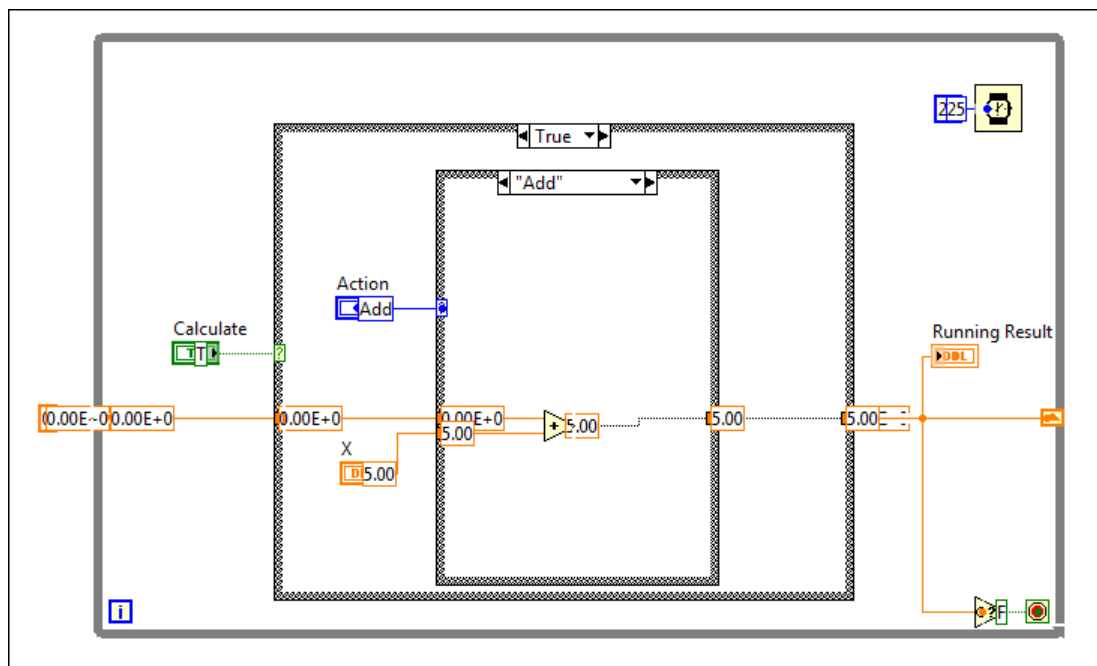


Figure 2. Loop Giving Expected Result

- ☐ Notice that when the VI runs the True case, a value of 5 is being output from the tunnel in the outer Case structure, as seen in Figure 2, so this is not the piece of code which is giving unexpected results.

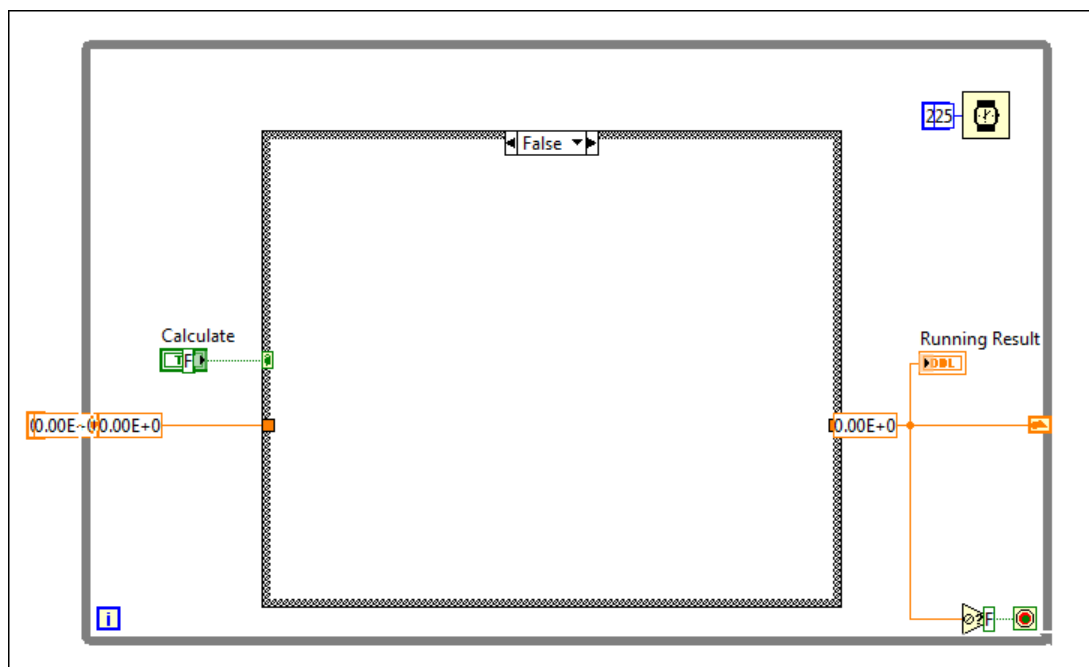


Figure 3. Loop Giving Unexpected Result

- ☐ Notice that when the VI runs the False case, a value of 0 is being output from the tunnel in the outer Case structure, as seen in Figure 3. Therefore, the Running Result indicator and the shift register is being overwritten with unexpected data.
- ☐ This is happening because Use Default If Unwired is selected for this tunnel. Click the **Abort** button to stop this VI.
- ☐ Right-click the tunnel on the right side of the outer Case structure and deselect **Use Default If Unwired**.

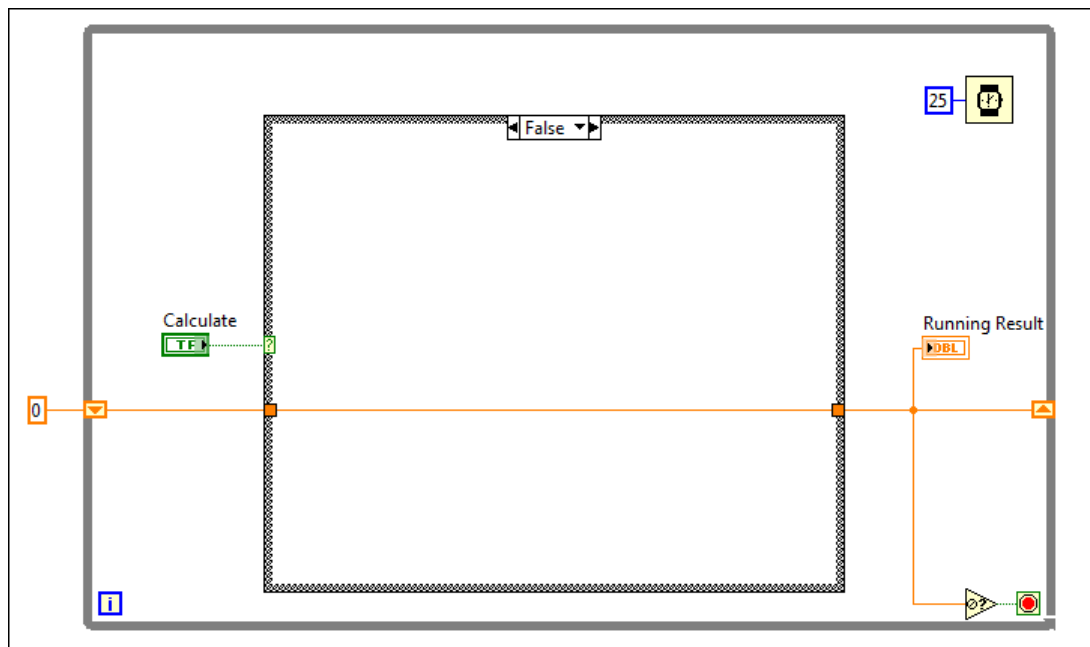


Figure 4. Properly Wired Case Structure

- ☐ The VI is broken again. Wire the tunnel on the left side of the outer Case structure to the tunnel on the right side, as seen in Figure 4.
6. Save the VI as `Fixed VI.vi`.
 7. Run the VI.
 - ☐ Click the **Run** button.
 - ☐ Click the **Calculate** button. Notice that a value of 5 appears in the Running Result Indicator.

End of Exercise

Notes
