Concept: Refactoring a VI

Goal

Improve an existing VI that is poorly designed.

Description

You receive a VI that is used as a subVI in a larger project. You must improve the VI for readability and user friendliness.

Evaluate the VI

The files that you need to complete this exercise are here:
`<NI eLearning>\LV Core 2\Refactoring Issues\Exercise`.

1. Open the `Determine Warning Bad One.vi` located in the `<Exercise>` directory. Figure 1 shows the block diagram of this VI.

   ❑ Run this VI with a variety of input values and observe the varying warning text.

2. Use the following list to evaluate the VI. Place a checkmark for all issues that apply.

   ❑ Disorganized block diagram

   ❑ Overly large block diagram

   ❑ Poorly named objects and poorly designed icons

   ❑ Unnecessary logic

   ❑ Duplicated logic

   ❑ Lack of dataflow programming

   ❑ Complicated algorithms

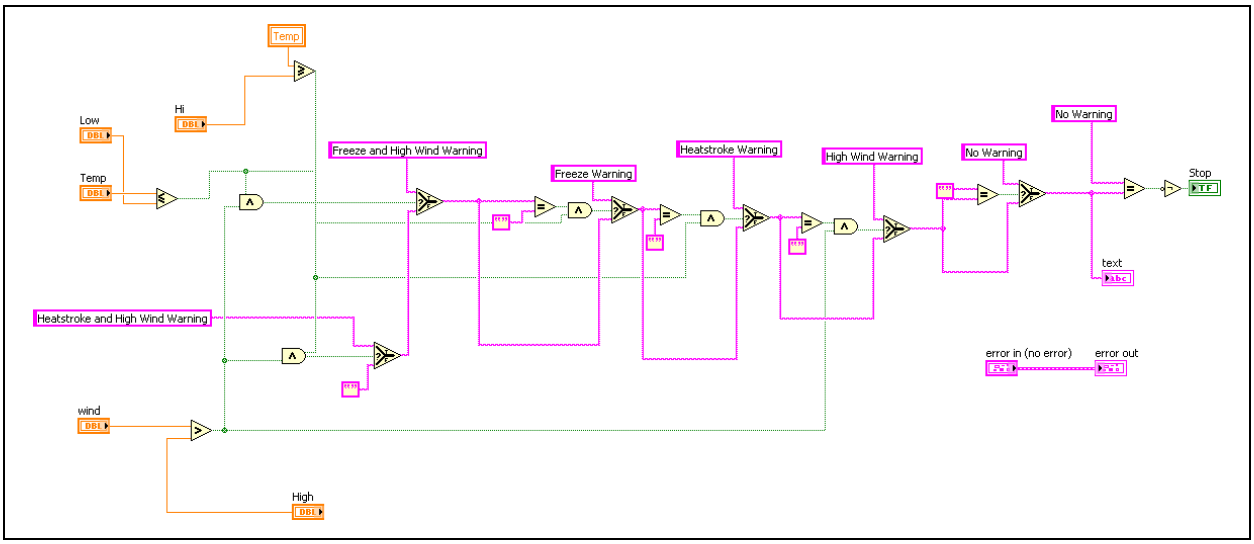

Figure 1. Poorly Designed Block Diagram

Improve the VI

Improve the VI in stages. Begin with the first checkmark: The block diagram is too disorganized.

1. Use the following tips to help you organize the block diagram:

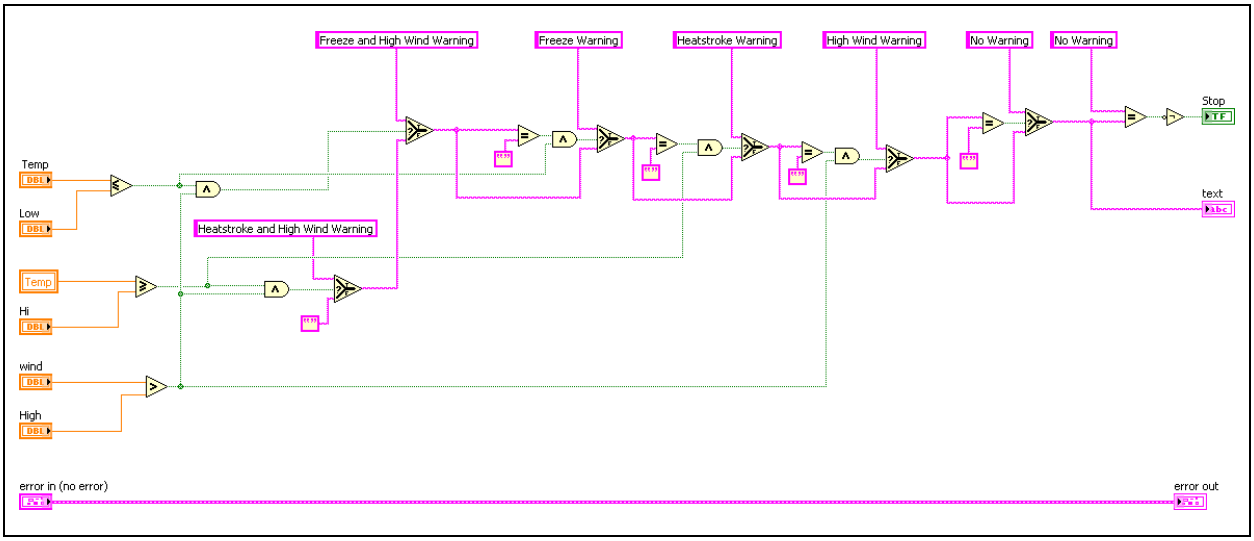

Figure 2. Reorganized Block Diagram

- ❑ Move all controls to the left of the block diagram.
- ❑ Move all indicators to the right of the block diagram.
- ❑ Use the **Align Objects** and **Distribute Objects** toolbar buttons toarrange the controls and indicators.

- ❑ Rearrange wires so that they do not overlap.
- ❑ Rearrange wires so that no input wires are running from right to left.
- ❑ Reduce the number of bends in wires.
- ❑ Do not allow wires to run under objects.

2. After the block diagram is better organized, rename controls and indicators using names that are more descriptive.
   - The purpose of this VI is to determine whether the current temperature and wind speed are at a level requiring a warning to generate. The VI also lights an LED if a warning occurs.
   - Suggested input names are `Current Temperature`, `Low Temp`, `High Temp`, `Current Wind Speed`, and `High Wind Speed`.
   - Suggested output names are Warning Text and Warning?.
3. Remove any unnecessary logic from the block diagram. Figure 3 shows the last few functions in the block diagram.

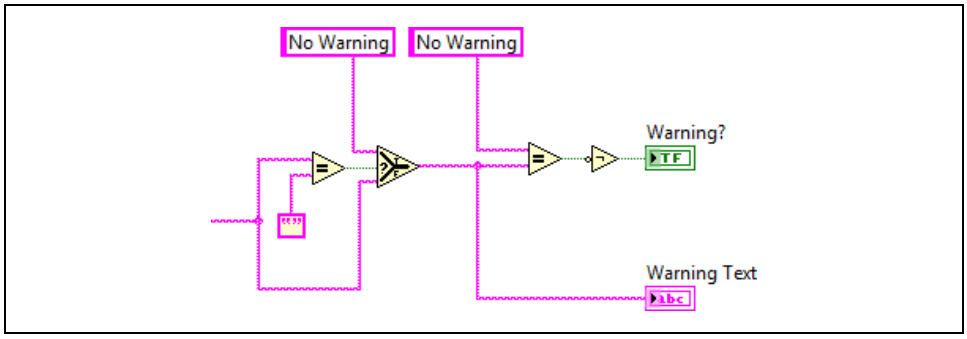

Figure 3. Unnecessary Logic

Notice that the Equal? function is followed by a Not function. One approach is to replace this with a Not Equal? function as shown in Figure 4.

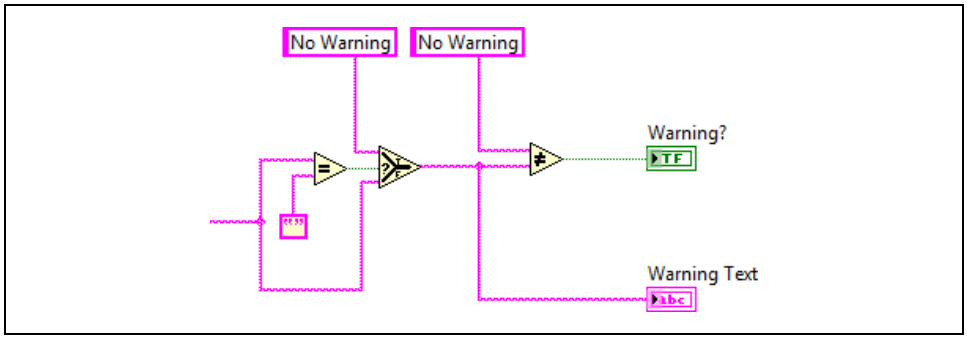

Figure 4. Unnecessary Logic Simplified

Notice you can reduce unnecessary logic even further by removing the Not Equal? function and inverting the Boolean input of the Select function, as shown in Figure 5.

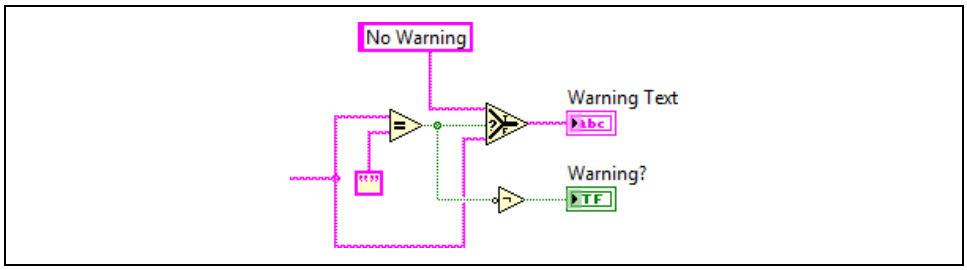

Figure 5. Unnecessary Logic Simplified Further

Refer to Figure 6 for assistance with wiring this duplicated function that occurs near the end of the VI.

- ❑ Delete the **Equal?** function.
- ❑ Delete the input wire to the Not function.
- ❑ Wire the input of the Not function to the input wire of the Select function.
- ❑ Test the edited VI to be sure the functionality has not changed.

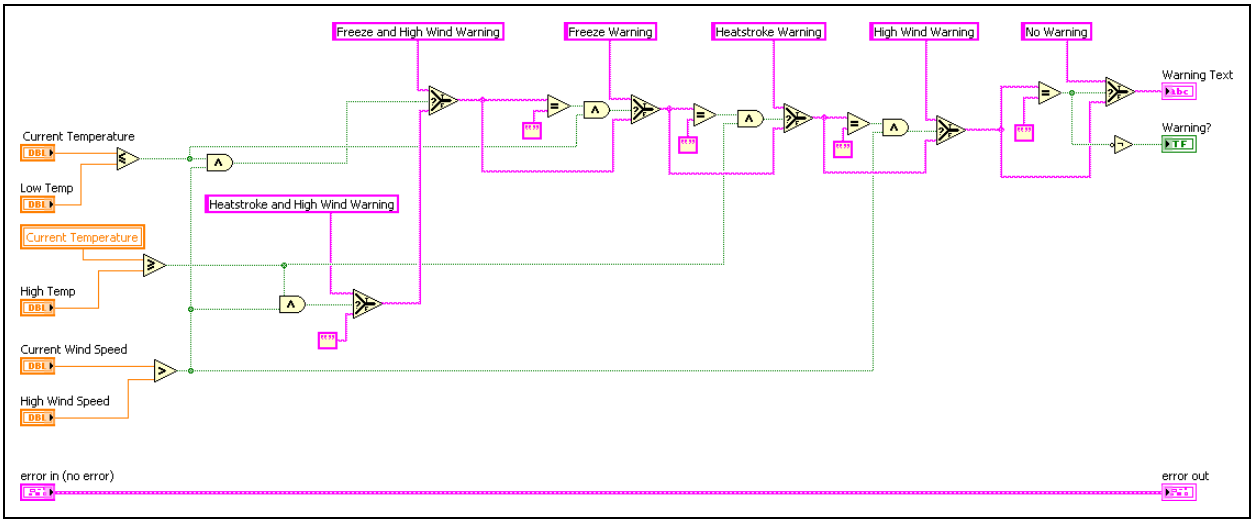

Figure 6. Well-Named Controls and Unnecessary Logic Removed

4. Save the VI as `Determine Warnings Good One.vi`.

Optional

1. Replace duplicated logic on the block diagram with subVIs. Figure 7 shows an example of the algorithm in the VI that is repeated. You can replace this algorithm with a subVI.

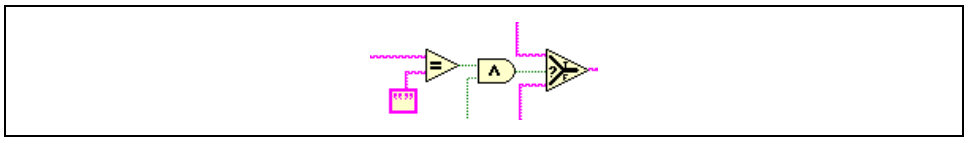

Figure 7. Repeated Algorithm

- ❑ Select the repeated algorithm by drawing a selection box around the objects.
- ❑ Select **Edit»Create SubVI**.
- ❑ Double-click the new subVI to open it.
- ❑ Edit the new subVI as necessary. Some things to consider: create an appropriate icon, recreate the connector pane, and rename the controls and indicators.
- ❑ Save the subVI.
- ❑ Close the subVI.

Note If the subVI is dimmed, right-click the subVI icon on the block diagram and select **Relink to SubVI** from the shortcut menu.

- ❑ Delete the duplicated logic in other locations and replace with the new subVI.
- ❑ Test the edited VI.

2. Remove unnecessary local variables and wire to the appropriate control or indicator instead.

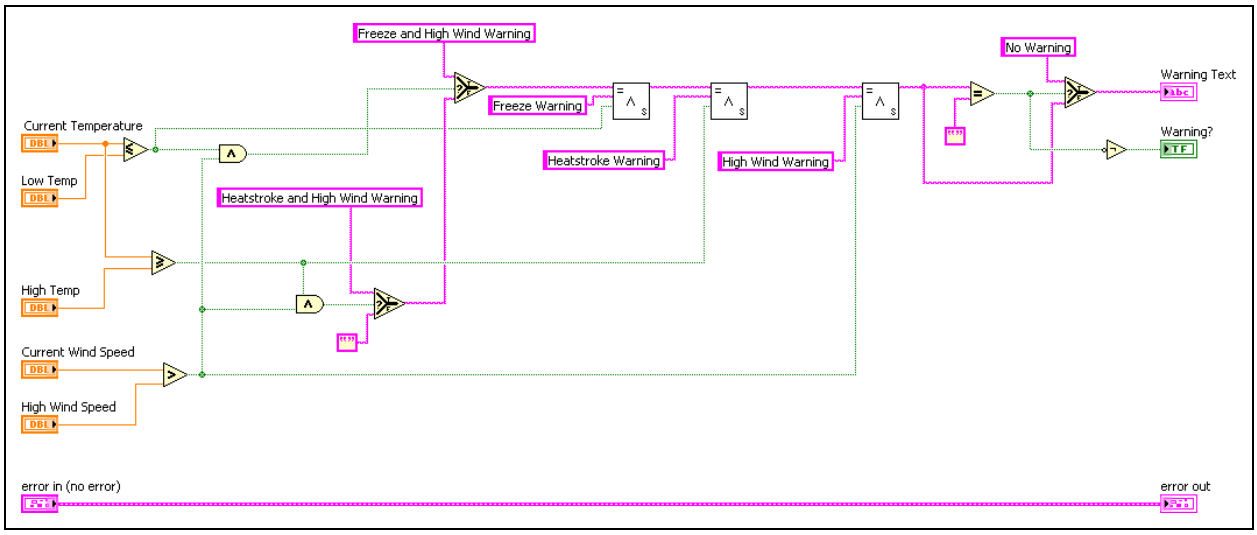

Figure 8. Duplicated Logic Placed in a SubVI and Local Variables Removed

3. Save the VI as `Determine Warnings Good One.vi`.

Challenge: Simplify Algorithm

If you have time remaining in this exercise, try to determine a way to simplify the algorithm and rewrite the code so that is easier to modify later.

An example solution is shown in Figure 9 using a state machine. The states contained are: Heatstroke, Freeze, High Wind, and Generate Warning. You can explore alternate solutions in the `<Exercise>\Challenge` directory.

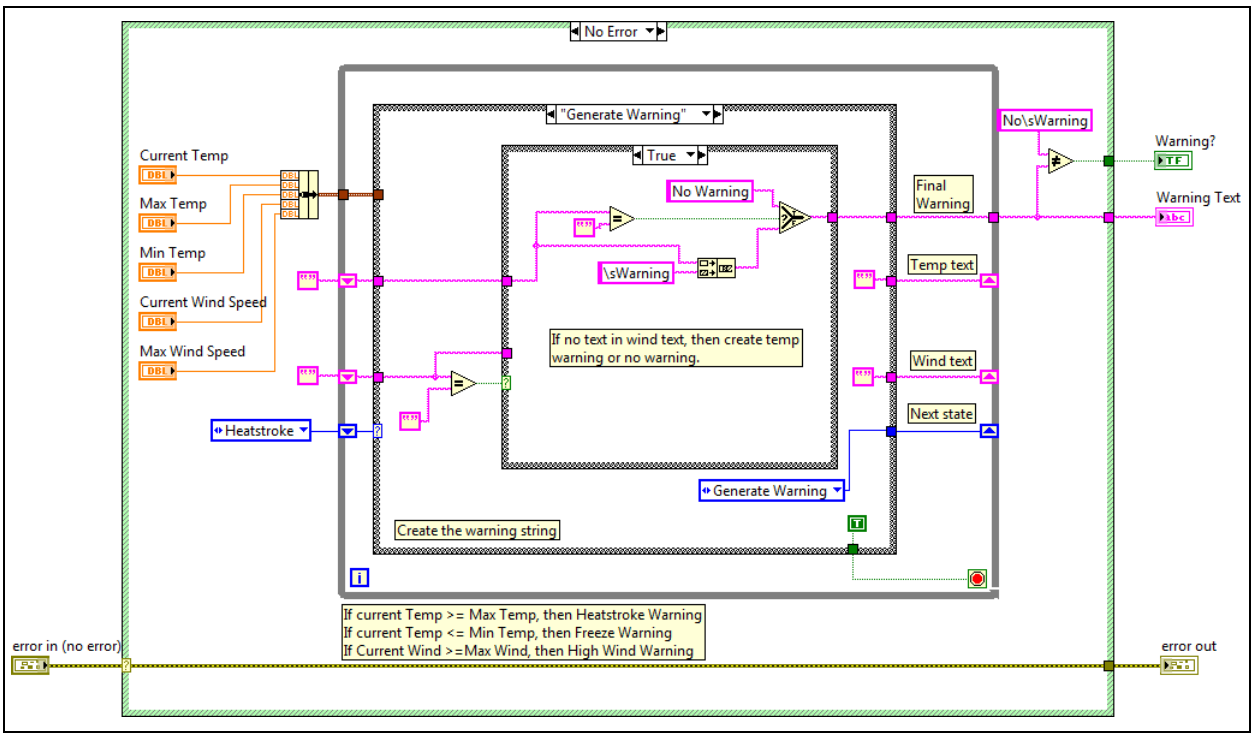

Figure 9. Alternate Algorithm that is Readable and Maintainable

End of Exercise

Notes