Using Case Structure in LabVIEW

Steve Zuraski

Executive Summary

Case structure in LabVIEW is a structure where you may have two or more subdiagrams. You would want to use this when you need to choose between different functional paths that continue through to the same next block. To chose a case or subdiagram to follow one would either utilize Boolean logic or numeric case selectors. In this application note I will go through reasons for implementing case structure, using different features of case structure, and an example of case structure.

Key Words: Case Structure, Subdiagram/Case, Case Selector, Input Tunnel, Output Tunnel, Error Handling

Introduction and Objective
A common structure that can be used in the block diagram of LabVIEW is a case structure. An example of what a case structure looks like can be seen below.

Case structures are used to implement one of two or more subdiagrams based on an input to the structure. This can be useful when different operations are required based upon what input variables are present at the start of the case structure. This is useful in a variety of applications like controlling the selection of a test based upon user input specifications or implementing true and false logic. The objective of this application note is to explain how to properly use the case structure in LabVIEW and to explain the properties of case structures.

**Implementation**

To create a case structure in the block diagram in LabVIEW you can follow the following instructions:

1. First you will need to find the case structure and place it in the block diagram. It can be found following the path **All Functions – Structures – Case Structure**.
2. Once the case structure is placed in the block diagram, the selector control will need to be set up. The selector node is the small green question mark that is always found on the left side of the case structure. There are some options in setting up the selector to the case structure. You can use an
integer, Boolean value, string, or enumerated type value as an input to the selector. Depending on what type is selected, the case structure will allow for a different number of subdiagrams. If you choose a Boolean selector control you will only have two cases, true and false. If any of the other choices for selector control you can have as many subdiagrams as you choose. To add a new case you can right click with the mouse and choose from the menu that pops up add case before or add case after.

3. Next add the desired items inside your case structure. These items will be what will run when the selector chooses that case. Switching between cases/subdiagrams can be done by clicking on the arrow by the title of the case.

4. Make sure your title of each case is labeled correctly. The title can be changed by double clicking it and typing in a new title. The title for each case needs to correspond to exactly what the indicator will see. This means that if you have a Boolean selector, you should have the title of either true or false.

5. If you prefer you can add a default case. This can be done by double clicking the title of the case and add the word default (will look like “true, default” for the default set to true in a Boolean case structure).

Features and Properties of Case Structure

1. **Auto Grow** – this will allow the size of the case structure to automatically size itself to fit all of the blocks you place inside of it.

2. **Add Case After** – will add another case after the one that is currently selected.

3. **Add Case Before** – will add another case before the one that is currently selected.

4. **Case Indicator** – this will choose the case that will be implemented while the program is running.
   Appearance of case indicator: [Image]

5. **Duplicate Case** – this will add another case that will have the same internal structure as the case that is currently selected.

6. **Error Handling** – you can wire in error clusters. The case selector will be green if there are no errors and it will turn red if there are errors. If an error occurs the case structure will execute the error subdiagram, if one is made.

7. **Input Tunnel** – this is a node where data lines enter into the case structure. For every input tunnel there needs to be a connecting path that goes to an output tunnel. Also for every tunnel created it needs to have a path in each
case. If it does not the node will be white and you may have errors unless you set the default for unwired terminals. This can be found when you right click on the tunnel in the short cuts menu.

8. **Output Tunnel** – this is a node where data lines leave the case structure. For every output tunnel there needs to be a connecting path that comes from an input tunnel. Also for every tunnel created it needs to have a path in each case. If it does not the node will be white and you may have errors unless you set the default for unwired terminals. This can be found when you right click on the tunnel in the short cuts menu.

9. **Title Box** – this shows what case is currently selected. Also note that this has to coordinate with the case selector.

   Appearance of title box: 

   Example of Case Structure

The following is an example of a case structure that is used to separate write functions for different tests. In the subsequent figures properties of case structures can be seen; these properties include case indicator, input tunnels, output tunnels, default case, and the title box.

Let’s first look at some of the properties of figure 2. On the bottom left edge of the case structure there is the case indicator (blue color). It is blue colored because it corresponds to a numeric value. It is controlled by a three way switch on the front panel, and there is also a default state built into the logic. Figure two shows case 0 as scene in the title block at the top. Case zero corresponds to when the switch sends the number zero to the case structure. Another property we see on figure two is the input and output tunnels. These are the square boxes on the left and right edges of the case structure. The purple line that passes through the case structure is the VISA resource name. This parameter needs to be given to all the write functions inside the case structure. It can be seen that this line passes all the way though the case structure. Now if we look at the same line in figure 4 we see that it still passes all the way through the case structure even though it is not connected to anything. This must pass through even though it does not connect to anything because it is being used in the other case states. If the purple line wasn’t connected the input and output tunnels would turn white and there would be an error message. In figure three we can see a case structure inside a case structure. This case structure is controlled via a Boolean state. It can be seen that the case indicator in this case is green, and that the case structure title is “True”. Another feature that we see in all of the figures below is the yellow line that passes through the case structure. This is an error handling line. In the VISA Write operation there can be error involved when LabVIEW runs the operation. The error handling line will filter out this error so it is not scene by the user of the program. It can be scene that it enters the case structure, connects to each write block and then leaves the structure. Even in a case like in figure 4, we can that the error handling line must still pass through the case structure even if it is
not used. If this was not present we would see the input and output nodes in all cases would be white, indicating an error in the input and output tunnels. If this is a problem you can fix this by either connecting the line in all cases or you can use the setup menu to create a default state if a node does not have a complete connection in a state.

Figure 2: VISA Write - Case 0
Figure 3: VISA Write - Case 1

Figure 4: VISA Write - Case 3, Default
Conclusion

Case structures are very useful tools provided by LabVIEW when dealing with multiple subdiagram implementations. Case structures are controlled by Boolean logic or case sequencing as seen in the example. Provided in this application note were the properties of case structures and how they were used. An example of how case structure is used and what it looks like was seen in the example that showed the setup for a write sequence for different tests. In conclusion, case structure is a useful tool when separating blocks/subdiagrams by a logical or numerical input.
References

