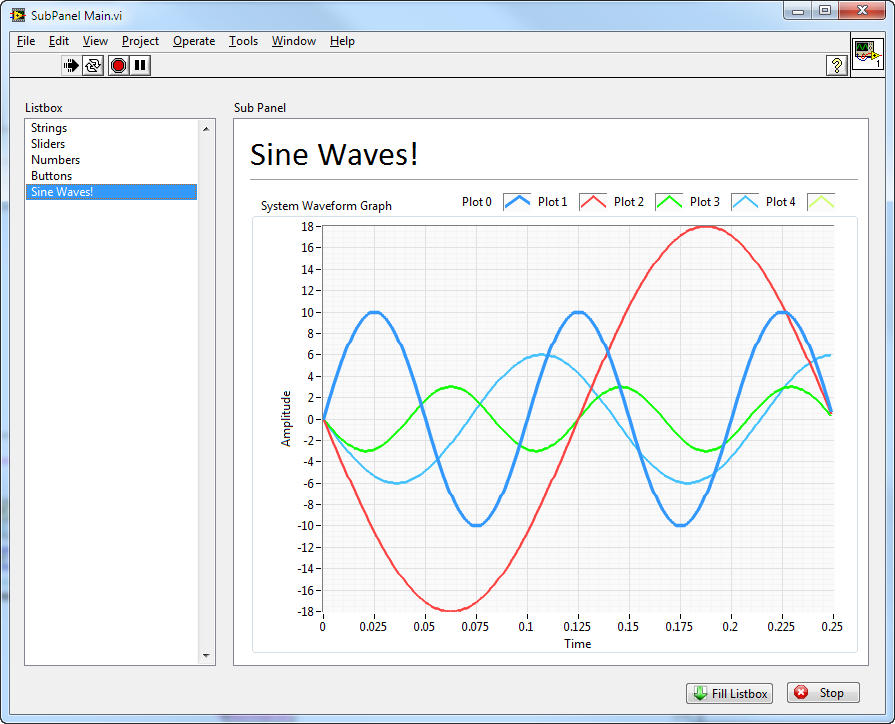
Sub Panel Tutorial

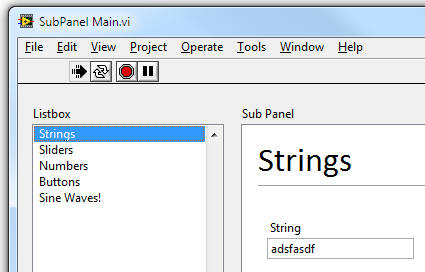
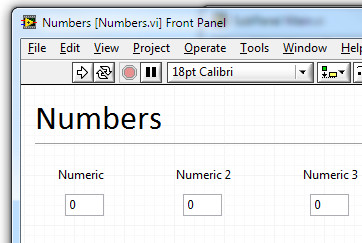
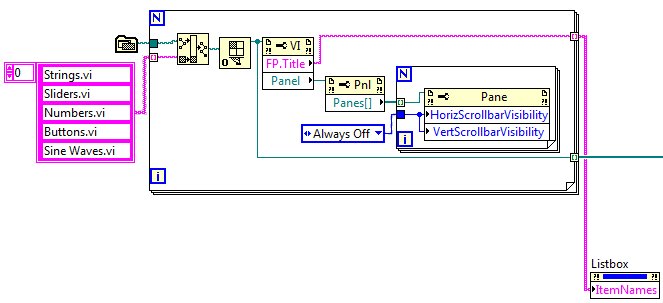
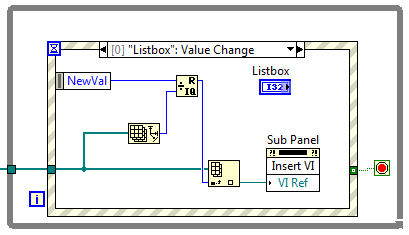
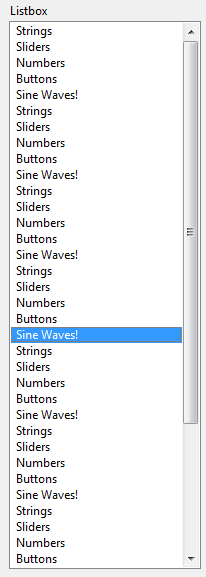
# Background

In this tutorial we will modify examine a more robust approach than tab controls to compartmentalizing a UI. A Listbox coupled with a Sub Panel allows for a large number of dynamically loaded panes to be displayed on demand. This effect can commonly be seen in options dialogs where may settings are organized and displayed through one dialog (see **Tools»Options** in LabVIEW for one such example).  
  


# Setup

You’ll need LabVIEW installed (my VIs were made in 2010 but you can follow along in almost any version).

# Steps

1. Open **SubPanel Main.vi** and run it.
2. Click on different entries in the Listbox and note how the contents of the Sub Panel change.   
     
   
3. Each “page” that displays in the Sub Panel is really it’s own VI. You can open up one of the Sub VIs and see that we aren’t doing anything special (do not open the one currently being displayed or else you’ll get an error message).  
     
   
4. Looking at the block diagram for SubPanel Main.vi you can see we have two main sections in the code. The first loads references to all of the subVIs and sets the Scrollbar Visibility to always off so that you don’t seen scrollbars within the sub panel. We store the names of each of the subVIs in an array to use as the items in the Listbox and we keep the array of references around as we’ll need it later.   
     
   
5. The second section of the code simply reacts to a user changing the value of the Listbox by picking the corresponding reference out of the array of references and inserting it into the Sub Panel.  
     
   
6. Return to the running SubPanel Main.vi and click the **Fill Listbox** button () -- you’ll see many more items added to the Listbox. This is the real benefit of this approach over a tab control based approach. Namely we can dynamically change the selectable items at runtime (whereas with a tab control you must populate the tabs at edit time) and we can have an almost unlimited number of items and still be able to read the text of each.  
     
   
7. END.