

LabVIEW TAB or SubPanel?

Category: Senza categoria

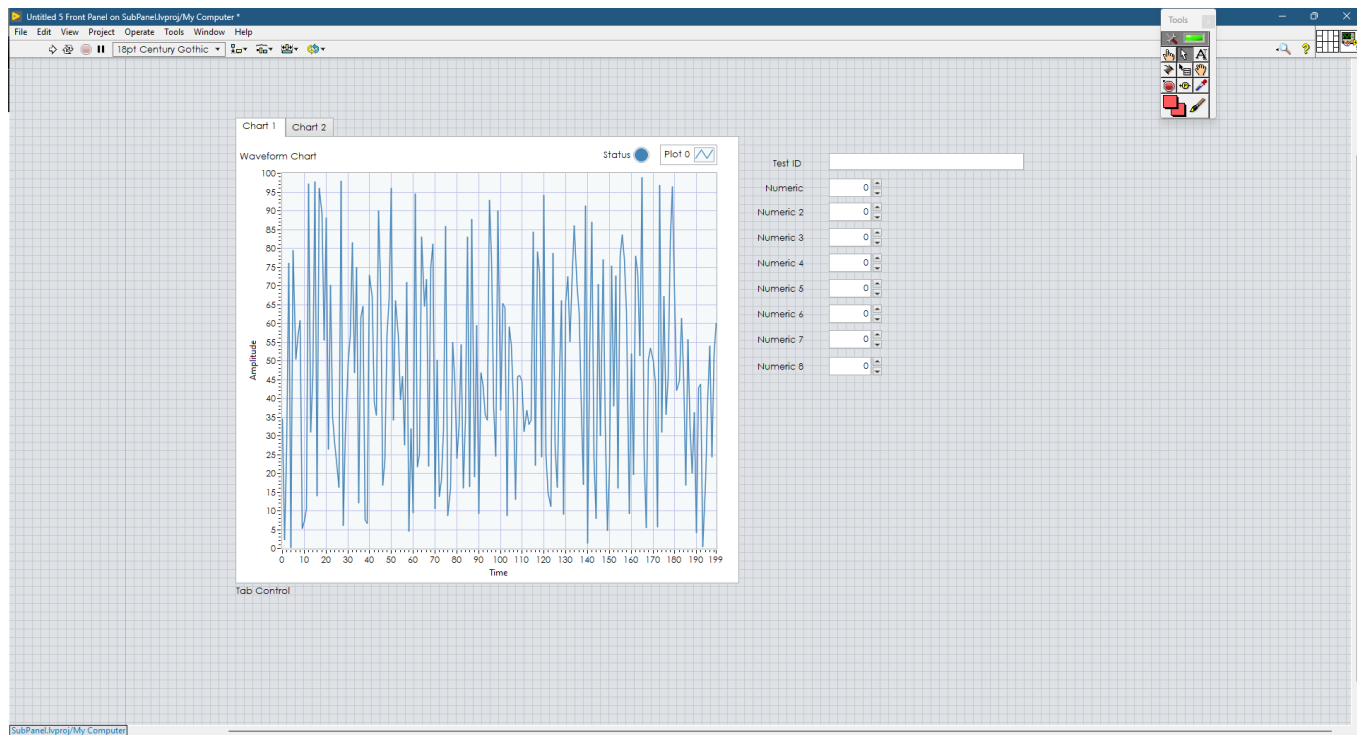
written by Nicola Bavarone | February 8, 2025

Surely, you have found yourself in a situation where your front panel feels like it's bursting with too many controls.

The easiest and quickest solution (and you know when I say easiest, it's probably not the best or most efficient) is the TAB Control, a tool that allows you to organize controls across multiple pages, helping to structure your interface and manage the clutter by displaying controls only when relevant.



Disorgannized Panel without TAB



A slight improvement by placing graphs on different TAB Control pages.

TAB Control: Simple, but be cautious

Implementing a TAB control is straightforward and intuitive: just position it and drag the controls onto the active pages.

However, the problem lies in the fact that while your panel might look more organized—controls are only visible when their respective page is displayed—they still exist in the diagram, and you need to manage them there. Before you know it, you could end up creating 10 pages, each with 10 controls. Wait, 100 controls in a UI, all managed from your diagram?

Another advanced issue is scalability. Imagine needing to make major modifications—you'll probably end up redesigning the UI and redoing one or more tabs.

When to use TAB controls?

A general rule is to use this option when there are few controls to organize, especially “Fittable” controls (explained below), and when you are confident that you won't need to make significant changes to the VI later.

Fittable Controls

Though this might not be a widely recognized term in the LabVIEW developer community, it has started emerging with the use of split bars and panels to create resizable interfaces.

Fittable controls include charts and other graph types, list boxes, multilist boxes, and tables. Even the TAB Control can be fittable, provided it contains only fittable controls. In essence, they are housed within a container that,

when set to “Fit to Pane,” automatically resizes within the panel.

On the other hand, controls like strings or numeric inputs, when positioned alongside others, could become awkwardly large or misaligned within a resizable TAB, unless you add some code to realign them properly.

Quick Example: Implementing a TAB Control

Here’s a quick example, sped up 4x for brevity, just to show you how you can organize your controls using tabs, placing graphs on different pages and using a list box as a menu to switch between pages.

SubPanels: Complex but Versatile

SubPanels offer a more complex yet versatile solution. They only load into memory when needed, enabling plugin architectures, modularity, and an effective separation of the GUI from the main code through abstraction.

However, this added flexibility brings increased complexity and the need for interprocess communication.

A Quick Example: SubPanels

Let’s try to replicate the previous example, but this time using SubPanels.

The first challenge: dynamically loading external VIs into our main VI. For this, you’ll need familiarity with dynamic VI calls, asynchronous methods for passing data between called VIs, and a solid understanding of reentrancy, which allows a VI to launch fully independent instances of itself.

My preferred method—though not the only one—for passing data between VIs is using User Events.

What’s the benefit after all this effort?

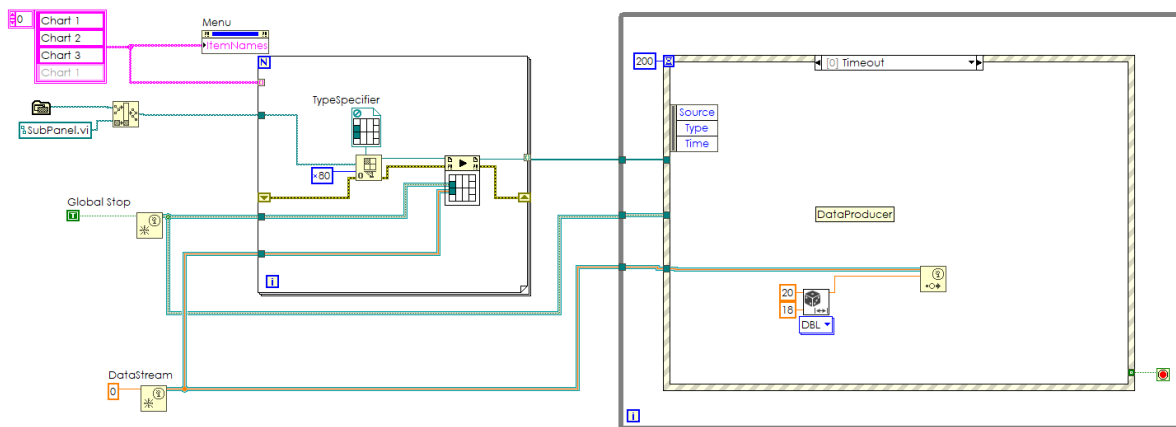
I know, it seems like more effort than benefit, but let’s look at the advantages:

You’ve created a “SubPanel.vi” with a resizable chart (sure, it could look prettier, but focus on the fact that it’s a fittable control). Even better, the VI is reentrant, meaning each time you call it, you create a completely independent copy. So, with a bit of creativity, you could have named it “Chart.vi” and created a reusable VI that can be used anytime you need to add a chart to your SubPanel (and it’s already fitted to the panel it’s housed in). Better, right?

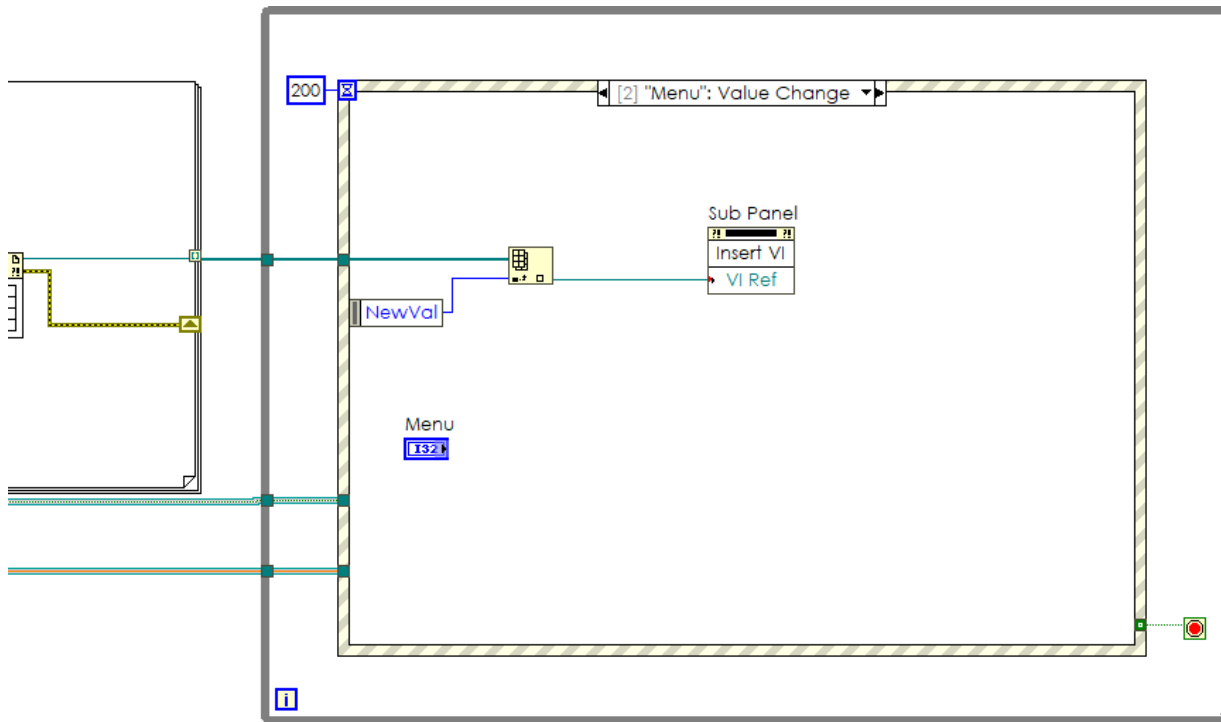
Now, you might ask, “But how do I pass data to it or control it?”

This is why I prefer using events for communication between dynamic VIs.

We can use the **“Call and Forget”** method. To do this, we open the VI using a Type Specifier, allowing us to expose the terminals in the “Start Asynchronous Task.”



In the “TimeOut” event, I pass the data, though I could have passed it in different ways (for example, creating a data stream event for each clone, allowing data to be passed corresponding to the desired plot).



Which to choose?

The choice between Tabs and SubPanels in LabVIEW depends on the project's complexity and specific needs.

Both options offer unique advantages, and making a wise decision will contribute to system efficiency and performance.

If you have few controls to manage and no specific requirements, the TAB Control is a valid choice.

When the number of controls grows or when you need modularity and scalability, SubPanels offer a strong alternative.