Picture This: Creating Controls on the Fly

by
Benjamin A. Rayner
Architect, Measurement & Automation
Data Science Automation, Inc.
USA

Category:
Prototype/Test

Products Used:
LabVIEW 8.2.1

The Challenge:
A developer of custom control and monitoring systems needed an application that would allow a standard set of hardware to be monitored and controlled using multiple large format touch screens.

The Solution:
Data Science Automation exploited the power and flexibility of LabVIEW dynamic event registration and the picture control to implement an application that supports the creation of re-configurable virtual user interfaces. The final application realized one of the "holy grails" of LabVIEW: Creating controls on the fly.

Abstract:
A large US government sub-contractor recognized a need for an application that would reduce the time associated with designing, prototyping and evaluating the ergonomic and logical layout of their control panels. Data Science Automation developed a solution that utilized Dynamic Event Registration and the LabVIEW Picture control, yet allowed users with no knowledge of programming to use and create controls on the fly.

Doing the impossible
Over the years LabVIEW has earned a well-deserved reputation as a powerful, efficient and flexible tool for representing the physical systems that by an application is controlling or monitoring. One constraint on this functionality is that LabVIEW cannot programmatically create controls or indicators. These can only be instantiated on a front panel during development. While property nodes give you great flexibility to manipulate controls once they are present, you can't create controls "on-the-fly" (Figure 1).

Figure 1 – LabVIEW Champion Christian Altenbach's artistic interpretation of "Controls on the Fly".

However, as with many things in LabVIEW, what the programming environment is actually capable of is not nearly important as what ingenuity and skill can make it look like LabVIEW can do. As a case in point, the underlying technology for generating the appearance of dynamically instantiated control and indicators is the picture control and has existed for many years. Unfortunately, in earlier versions of LabVIEW the process of interacting with images in a picture control such that they appear to respond to button clicks, and update to reflect changes in external data inputs would have been prohibitively complex for all but the most trivial application of the technique.

Clearly the advent of the event structure simplified the process. However, as we shall see, the introduction of "Dynamic Event Registration" was the thing that really made this approach practical. They allow arbitrarily complex event functionally to be encapsulated in a subVI that provides the requisite "smoke and mirrors" to make the technique work while hiding this complexity from higher-levels in the application's code hierarchy. The remainder of this paper discusses how these concepts are being used to develop an application that will model and test control panels before the customer invests any time is in creating a physical prototype.

Conjuring Controls

As illustrated in Figure 2, Data Science Automation has under development an application that allows system designers to quickly and easily compose control panel designs by creating, editing and repositioning virtual controls and indicators. Once a panel design is completed the designer can put the interface into operational mode and evaluate it for organization and ease of use. In addition, because the application is being developed assuming a touch panel interface, operators will be able to interact with this virtual control panel in a manner similar to how a physical control panel would be used. Finally, the application can save a dataset defining the design so it can be reloaded or reused.

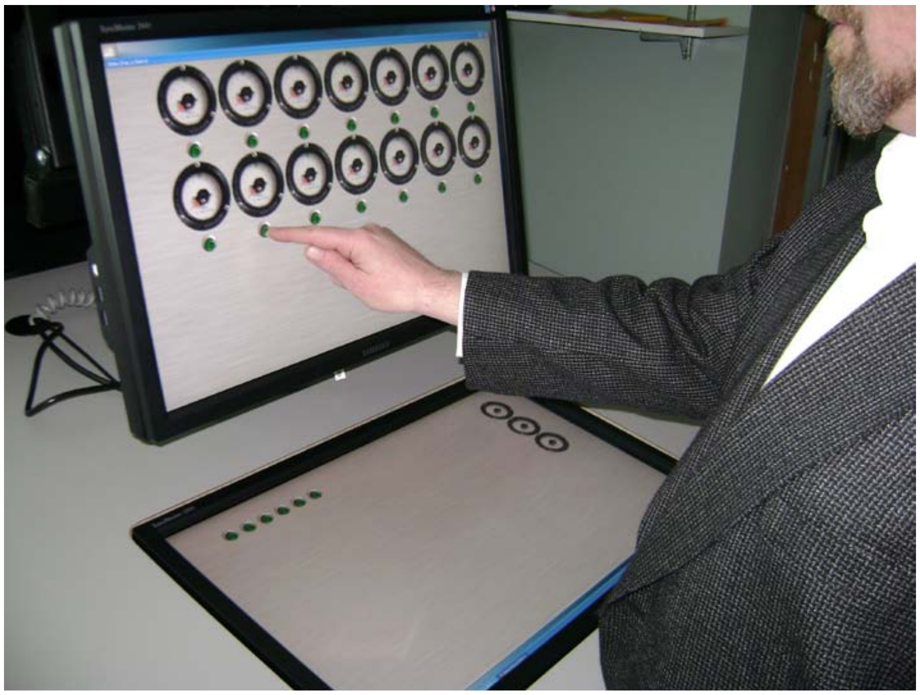


Figure 2 – An operator clicks a virtual push-button to evaluate a possible panel layout.

A key aspect of this functionality is that its use assumes (and requires) no knowledge of LabVIEW to design new control panels. Moreover, because panel designs can be saved and reloaded, the user can quickly switch between different designs. The customer has recognized this ability to switch quickly between designs as a powerful feature that may some day allow operators to quickly respond to system

© Copyright, Data Science Automation, Inc. All Rights Reserved.

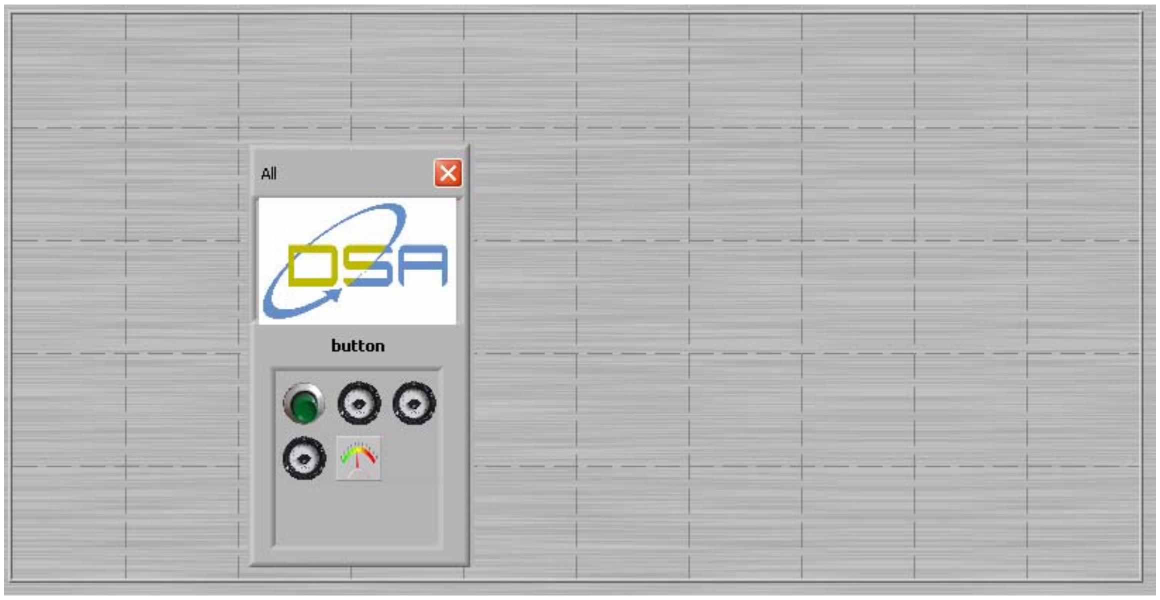
375 Valley Brook Road • Suite 106 • McMurray, PA 15317 • T 724-942-6330 • F 724-942-8390 • DSAutomation.com

failures by transferring critical control and monitoring functions from a failed or damaged system to hardware that had previously been monitoring a less critical process.

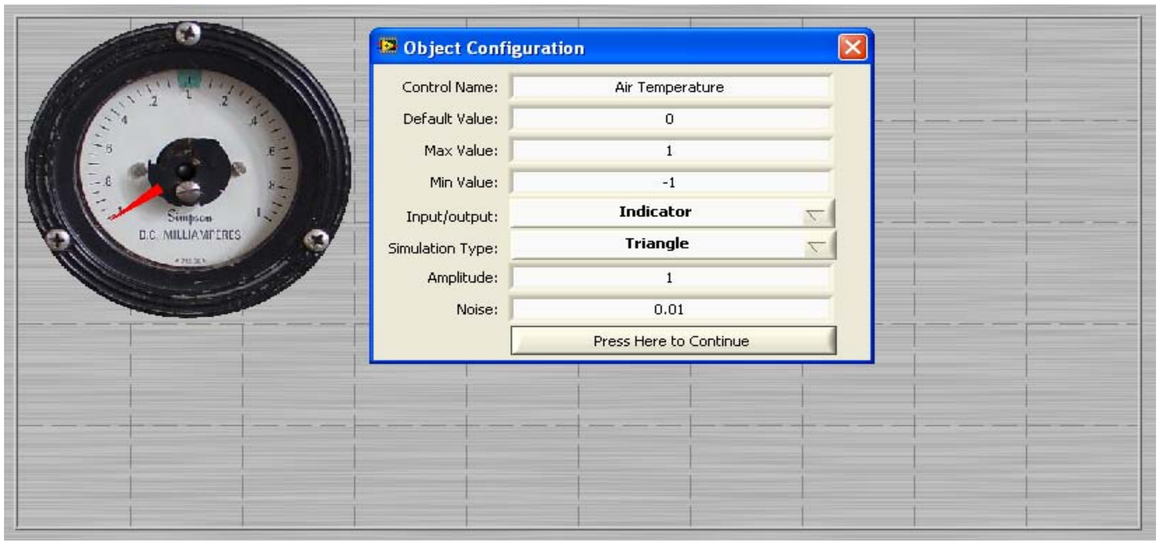
Putting it Together Virtually

When the application is in “Design” mode, laying out a new panel design is analogous to creating a front panel in LabVIEW. After starting the application and choosing to create a new design a blank “Control Panel” is presented. Right-clicking on the control panel invokes a palette from which the designer can choose one of the available controls (Figure 3). As illustrated in the figure, the control panel has optional grid-lines while in design mode which help in distributing objects.

Figure 3 – When the application is in design mode right clicking in free space on the virtual control panel presents a palette from which the designer can select the object to create.



After selecting the desired control from the palette, the control panel designer clicks on a blank portion of the control panel area to identify where the application should place the control object. This action also opens a configuration screen (Figure 4) that allows the designer to specify the control’s name, default value, and data range. In addition, it presents pop-up menus for defining the object as a control or indicator, and identifying what signal simulation (if any) is desired for the object. This latter feature is valuable for creating control panel designs in the absence of the physical system it will control.

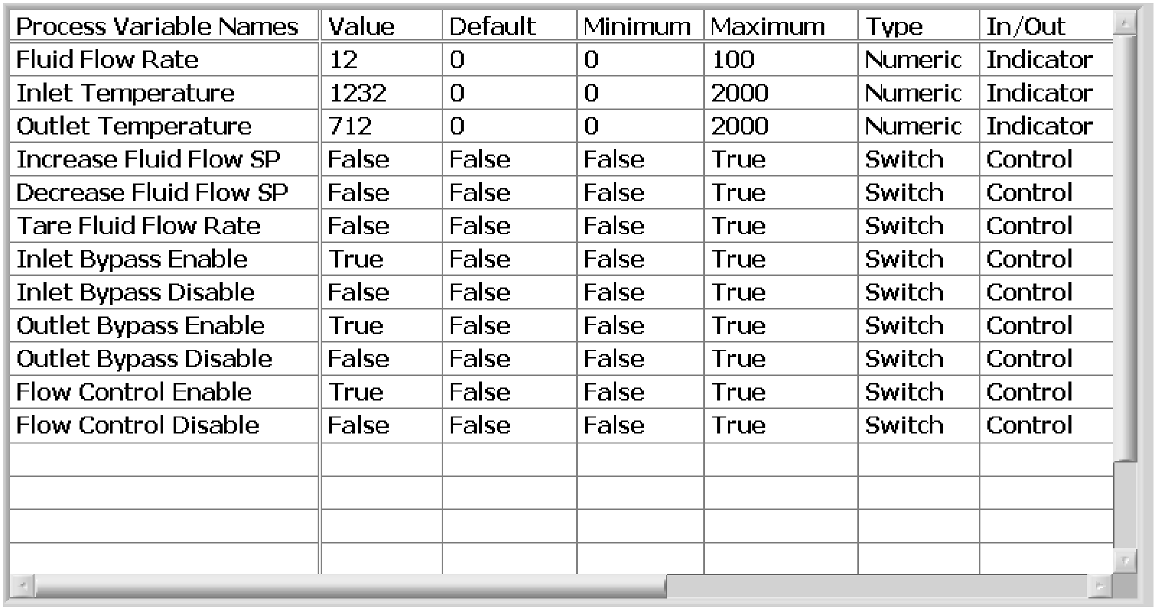


Control Name: Air Temperature
Default Value: 0
Max Value: 1
Min Value: -1
Input/output: Indicator
Simulation Type: Triangle
Amplitude: 1
Noise: 0.01

Figure 4 – Once the control object is placed, a dialog box opens to configure the virtual interface object’s characteristics.

When the design is complete, the designer places the application in “Operate” mode. From this point on, the controls and indicator operate in a manner very similar to the actual physical device: meters and gauges will update, buttons and switches will toggle, and knob will turn.

To further aid in development of control panels and their physical interfaces, an Information screen (Figure 5) is available that shows the current state of all I/O and provides supports the ability to over-ride any of the values.



Process Variable Names	Value	Default	Minimum	Maximum	Type	In/Out
Fluid Flow Rate	12	0	0	100	Numeric	Indicator
Inlet Temperature	1232	0	0	2000	Numeric	Indicator
Outlet Temperature	712	0	0	2000	Numeric	Indicator
Increase Fluid Flow SP	False	False	False	True	Switch	Control
Decrease Fluid Flow SP	False	False	False	True	Switch	Control
Tare Fluid Flow Rate	False	False	False	True	Switch	Control
Inlet Bypass Enable	True	False	False	True	Switch	Control
Inlet Bypass Disable	False	False	False	True	Switch	Control
Outlet Bypass Enable	True	False	False	True	Switch	Control
Outlet Bypass Disable	False	False	False	True	Switch	Control
Flow Control Enable	True	False	False	True	Switch	Control
Flow Control Disable	False	False	False	True	Switch	Control

Figure 5 – To aid in troubleshooting the virtual interface, the designer can at any time open an information screen showing the state and configuration of all I/O.

Diving Under the Hood

The heart of the logic implementing this functionality is a subVI (Figure 6) that responds to clicks in the control panel area. When the subVI starts, it registers to receive two events. One is for "Mouse Down" events on the picture control that forms the virtual control panel. The other is a global shutdown event that is used to stop the application.

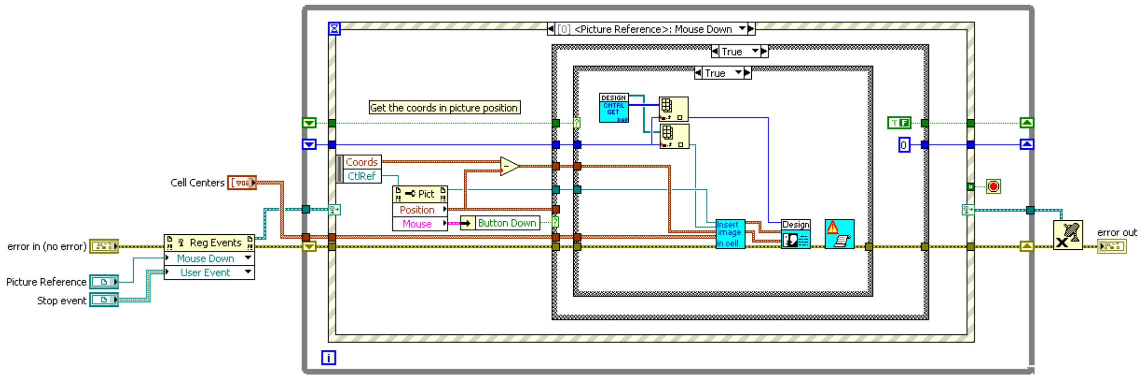


Figure 6 – Thanks to Dynamic Event functions for interacting with the virtual interface can be implemented in a subVI, thus keeping the main application clean and allowing for re-use in other applications.

When the user clicks on the picture, the coordinates of the click are retrieved and a check is made to determine if it was a right or left mouse click. Note that if the "Button Down" property is False when the event fires, it was a right-click that occurred. Conversely, a True value indicates a left click. Therefore the figure shows the code used to handle a left-click event.

Functionally, the code uses the CtrlRef event data to determine where the mouse was clicked and whether it was a right or left click. With data in hand it can then select cases appropriate for each case. Without the power and flexibility of the

© Copyright, Data Science Automation, Inc. All Rights Reserved.

Dynamic Events this logic would have to be exposed to the main application. This exposure would in turn result in a high degree of coupling between the main application logic and the logic managing the control panel, and low cohesion in the main application itself – both of which are Very Bad Things.

© Copyright, Data Science Automation, Inc. All Rights Reserved.

375 Valley Brook Road • Suite 106 • McMurray, PA 15317 • T 724-942-6330 • F 724-942-8390 • DSAutomation.com