Event-Driven Programming in LabVIEW

Publish Date: Jul 20, 2008

Overview

This document describes the differences between procedural-driven and event-driven programming, and it describes briefly how to use event-driven programming in LabVIEW. An event-driven program executes in an order determined by the user at run-time. In LabVIEW, you can use the Event structure to handle events in an application. Using the Event structure simplifies your block diagram, minimizes CPU usage, and handles user interface events that you could not handle in previous versions of LabVIEW.

Table of Contents

1. Procedural-Driven and Event-Driven Architectures

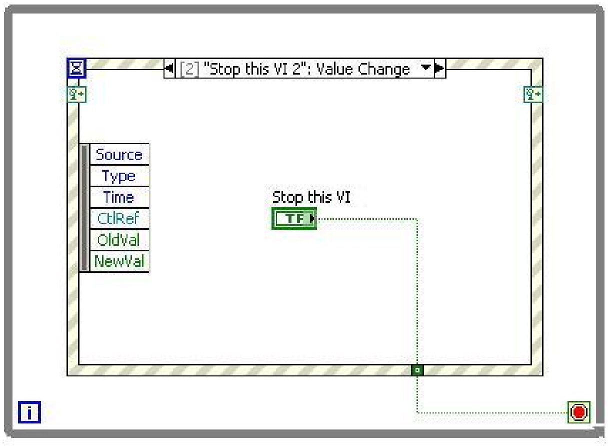
Events are caused by actions the user performs. For example, clicking the mouse generates a mouse event, pressing a key on the keyboard generates a keyboard event, and so on. When a system event occurs, the operating system is responsible for identifying and responding to the event. Different operating systems implement this process in various ways. However, all operating systems respond to the system event by broadcasting a system event message. Any application running on the system can react to the system event by executing code written for that specific system event.

In procedural-driven or top-down architectures, the application executes a set of instructions in a specified sequence to perform a task. The structure and sequence of the program, not user actions, control the execution order of a procedural-driven application. The program execution begins in main and then flows through method calls and control statements in a fairly predictable manner.

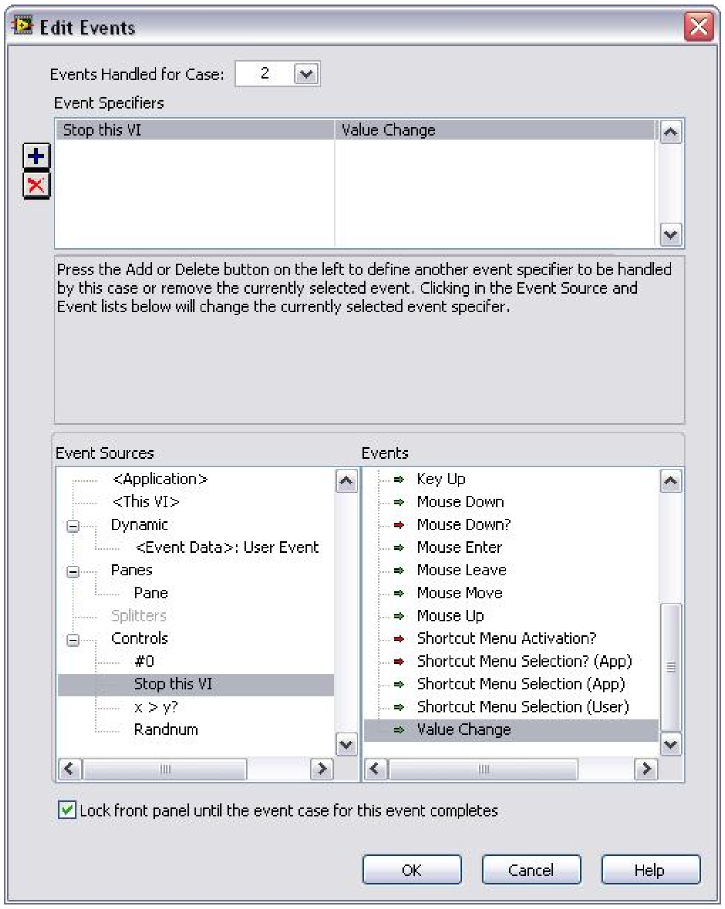
In an event-driven program, the program first waits for events to occur, responds to those events, then returns to waiting for the next event. How the program responds depends on the code written for that specific event. The order in which an event-driven program executes depends on which events occur and on the order in which those events occur. While the program waits for the next event, it frees up CPU resources that might be used to perform other processing tasks.

2. Using Events in LabVIEW

In LabVIEW, you can use the Event structure to handle events in an application. As with a Case structure, you can add multiple cases to the Event structure. You can then configure those cases to handle one or more events. When those events occur, LabVIEW executes the corresponding case. The following block diagram shows an example Event structure.



You configure events by right-clicking the Event structure border and selecting **Edit Events Handled by This Case** from the shortcut menu. Use the **Edit Events** dialog box that appears to edit single or multiple cases.


Edit Events

Events Handled for Case: 2

Event Specifiers

Stop this VI	Value Change

Press the Add or Delete button on the left to define another event specifier to be handled by this case or remove the currently selected event. Clicking in the Event Source and Event lists below will change the currently selected event specifer.

Event Sources
- <Application>
- <This VI>
- Dynamic
  - <Event Data>: User Event
- Panes
  - Pane
- Splitters
- Controls
  - #0
  - Stop this VI
  - x > y?
  - Randnum

Events
- Key Up
- Mouse Down
- Mouse Down?
- Mouse Enter
- Mouse Leave
- Mouse Move
- Mouse Up
- Shortcut Menu Activation?
- Shortcut Menu Selection? (App)
- Shortcut Menu Selection (App)
- Shortcut Menu Selection (User)
- Value Change

[✓] Lock front panel until the event case for this event completes

Using the Event structure minimizes the CPU usage because the VI no longer must continually poll the front panel for changes. In contrast to polling, the Event structure does not lose user events because the structure uses an event queue to store user events and handle the user events in the order in which they occur.

3. Types of Events

A given event can either be static or dynamic and a static event can be either a notify event or a filter event. Static events are only for interactions with the front panel. Static, notify events are the most common and only react to what has occurred on the front panel. An example for a static, notify event would be a “Value Change” for an OK button. Static, filter events catch an event performed by the user before LabVIEW processes that event. The code within the Event Structure can then decide if that event should be processed or not. An example of a filter even would be a “Panel Close?” for the user clicking the close on the front panel. In the Edit Events window, filters have red arrows and notifies have green arrows.

There are times when you want to include events that don’t occur on the front panel. This is when you need to register a dynamic event. Using dynamic events will allow your Event Structure to react to code happening anywhere on your block diagram. An example of all these events is attached to this webpage.