State Machine VI

Goal

Create a VI that implements a state machine using a type-defined enum.

Scenario

You must design a template for a user interface state machine. The state machine must allow the user to activate Process 1 or Process 2 in any order. The state machine must also allow for expansion, because processes may be added in the future.

Design

Inputs and Outputs

Table 1. Inputs and Outputs

Type	Name	Properties
Cancel Button	`Process 1`	Boolean Text: `Process 1`
Cancel Button	`Process 2`	Boolean Text: `Process 2`
Stop Button	`Stop`	—

State Transitions

Table 2. State Transitions

State	Action	Next State
Idle	Poll user interface for selection	No button clicked: `Idle State` Process 1 clicked: `Process 1 State` Process 2 clicked: `Process 2 State` Stop clicked: `Stop State`
Process 1	Execute Process 1 code	Idle State
Process 2	Execute Process 2 code	Idle State
Stop	Stop the state machine	Stop State

Implementation

In the following steps, you will create the front panel window shown in Figure 1.

The files that you need to complete this exercise are here: `<NI eLearning>\LV Core 1\Using State Machines\Exercise`.

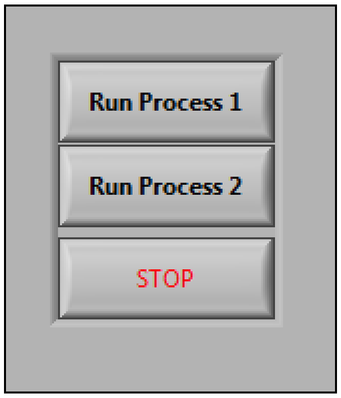

Figure 1. State Machine VI Front Panel Window

1. Create a new project containing a blank VI.

   ❑ Select **Empty Project** from the **Getting Started** window.

   ❑ Select **File»Save Project**.

❑ Name the project `State Machine.lvproj` in the `<Exercise>` directory.

❑ Select **File»New VI**.

❑ Save the new VI as `State Machine.vi` in the `<Exercise>` directory.

2. Create a menu cluster containing buttons for running process 1, running process 2, and stopping the VI.

❑ Place a cluster shell on the front panel window.

❑ Relabel the cluster `Menu`.

❑ Add a Cancel button to the cluster shell.

❑ Relabel the Cancel button `Process 1`.

❑ Change the Boolean text `Run Process 1`.

❑ Make a copy of the Process 1 button, and place the copy within the cluster shell.

❑ Rename the copied button `Process 2`.

❑ Change the Boolean text on the copied button to `Run Process 2`.

❑ Right-click each button and select **Visible Items»Label** to hide the labels.

❑ Add a stop button to the cluster shell.

❑ Right-click the **Stop** button and select **Visible Items»Label** to hide the label.

❑ Modify the Boolean text on the button using the **Text Settings** on the toolbar.

Suggested text settings: 24 point bold Application Font.

❑ Enlarge and arrange the buttons within the cluster using the resizing tool and the following toolbar buttons: **Align Objects**, **Distribute Objects**, and **Resize Objects**.

❑ Right-click the border of the cluster and select **Autosizing»Size to Fit**.

❑ Right-click the border of the cluster and select **Visible Items»Label** to hid the label.

3. Create the type-defined enum to control the state machine.

   ❑ Add an enum to the front panel window.

   ❑ Right-click the enum and select **Edit Items**. Modify the list as follows:

Items	Digital Display
`Idle`	0
`Process 1`	1
`Process 2`	2
`Stop`	3

   ❑ Select **OK** to exit the **Enum Properties** dialog box.

   ❑ Relabel the enum control `State Enum`.

   ❑ Right-click the State Enum and select **Advanced»Customize**.

   ❑ Select **Type Def.** from the Control Type pull-down menu.

   ❑ Right-click the Enum and select **Representation»U32**.

   ❑ Save the control as `State Enum.ctl` in the `<Exercise>` directory.

   ❑ Close the Control Editor window.

   ❑ Click **Yes** when asked if you would like to replace the control.

   ❑ Switch to the block diagram.

   ❑ Right-click the State Enum and select **Change to Constant**. The enumerate control no longer appears on the front panel window.

In the following steps, you create the block diagram shown in Figure 2. This block diagram contains four states—Idle, Process1, Process 2, and Stop.

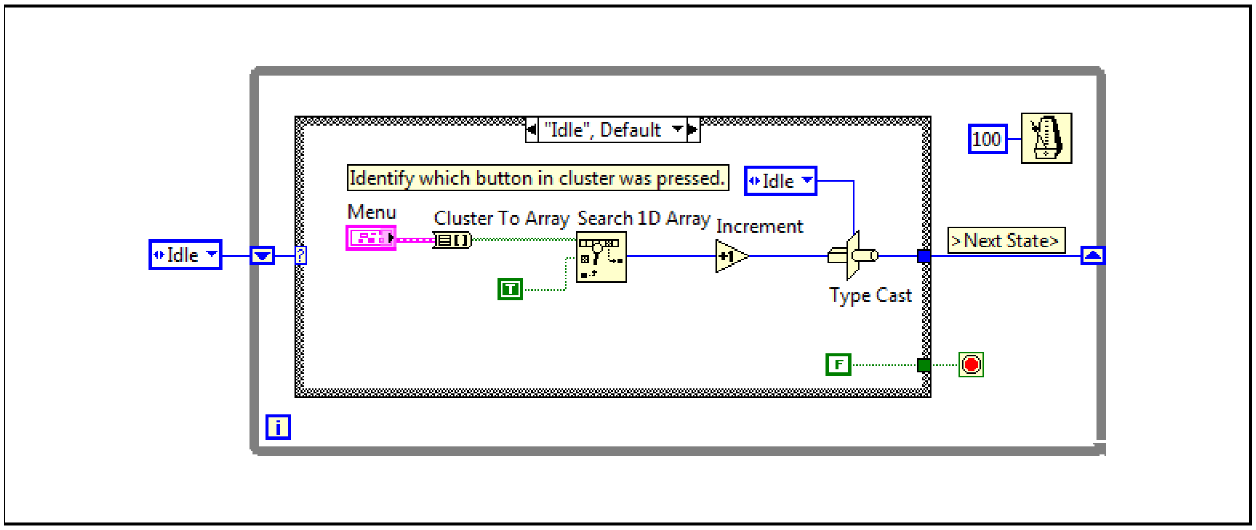

Figure 2. Idle State

4. Create the block diagram shown in Figure 2.

   ❑ Place a Case Structure inside a While Loop on the block diagram as shown in Figure 2.

   ❑ Wire the enum to the case selector terminal of the Case structure using a shift register on the While Loop.

   ❑ Right-click the Case structure and select **Add Case for Every Value** to automatically add a case for each item in the enum.

   ❑ Copy the enum to use within the Case structure. The copy is also linked to the type-defined enum.

   ❑ Switch to the Idle case of the Case structure, and wire a False constant to the conditional terminal; the state machine should not stop when exiting the Idle state.

   ❑ In the Idle state, you convert the cluster to an array so that the array can be searched for any button clicked. The Search 1D Array function returns the index of the button clicked. Because the Idle State does not have a button associated with it, this index must be incremented by one.

   ❑ Add an **Index Array** function to the Idle case.

   ❑ Add an **Array Constant** to the Idle case and move the copy of the enum into the array

❑ Expand the Array to display four items. Select a different state for each item in the following order as shown in Figure 2.

- Idle
- Process 1
- Process 2
- Stop

It is very important that the order of the cluster matches the order of the items in the enumerated constant.

❑ Complete the wiring for the Idle case as shown in Figure 2.

5. Complete the Process 1 state shown in Figure 3.

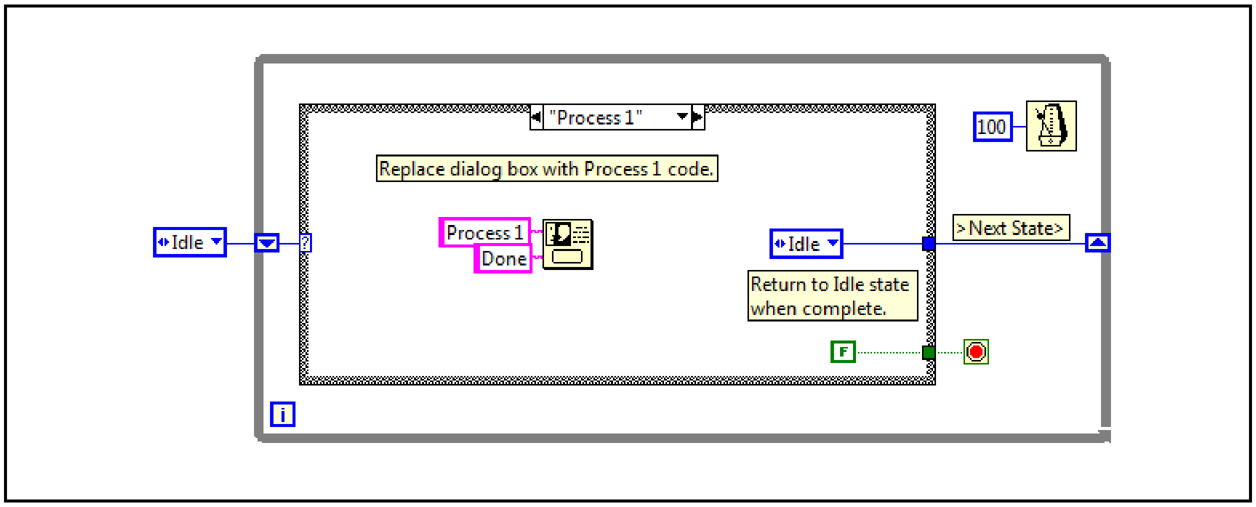
"Process 1"
Replace dialog box with Process 1 code.
Process 1
Done
Idle
Return to Idle state when complete.
100
> Next State>
F
i

Figure 3. Process 1 State

❑ Use a One Button Dialog function to simulate the Process 1 code.

❑ Wire a False constant to the conditional terminal; the state machine should not stop when exiting the Process 1 state.

6. Complete the Process 2 state shown in Figure 4.

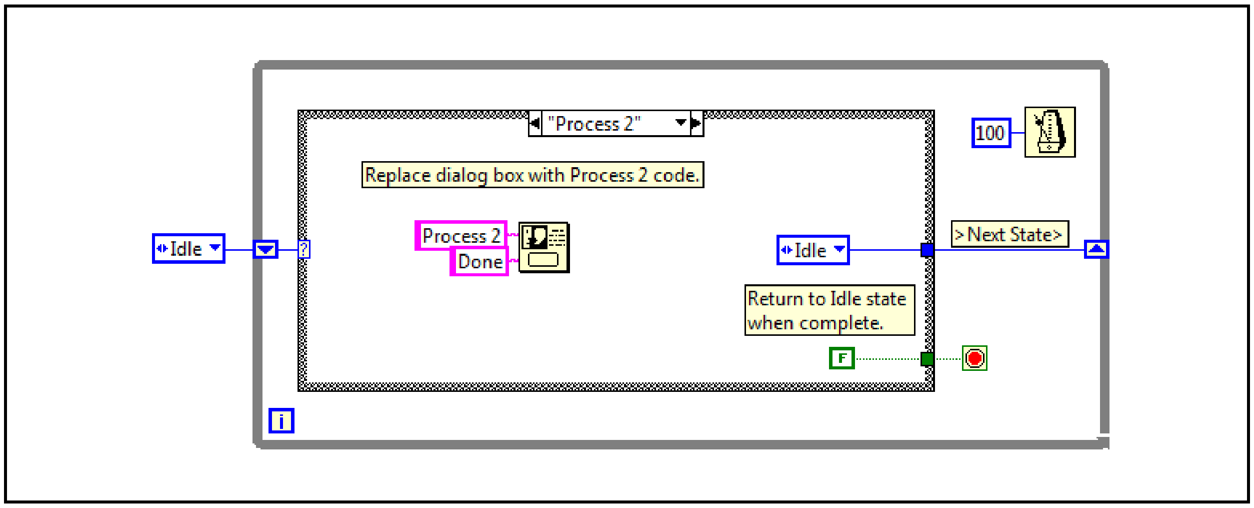
"Process 2"
Replace dialog box with Process 2 code.
Process 2
Done
Idle
Idle
Return to Idle state when complete.
100
>Next State>
F
i

Figure 4. Process 2 State

- ❑ Use a One Button Dialog function to simulate the Process 2 code.
- ❑ Wire a False constant to the conditional terminal; the state machine should not stop when exiting the Process 2 state.

7. Complete the Stop state shown in Figure 5.

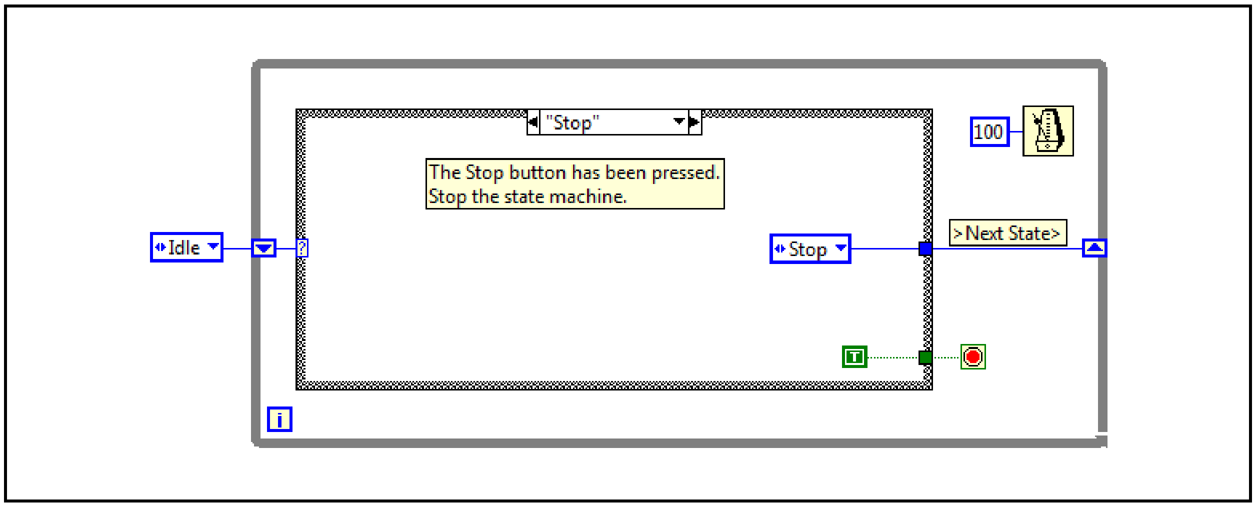

Figure 5. Stop State

- ❑ Pass a True constant to the conditional terminal; the state machine should stop only from this state.

8. Save the VI when you have finished.

Test

1. Switch to the front panel window.
2. Run the VI. Experiment with the VI to be sure it works as expected. If it does not, compare your block diagram to Figures 2 through 5.
3. Save and close the VI when you are finished.

End of Exercise

Notes