LabVIEW

Hands-On: Designing Reusable LabVIEW Code Using Software Architectures (State Machines)

Introduction to State Machines

The State Machine is one of the fundamental architectures LabVIEW developers frequently use to build applications quickly. State Machine architecture can be used to implement complex decision-making algorithms represented by state diagrams or flow charts.

State Machines are used in applications where distinguishable states exist. Each state can lead to one or multiple states, and can also end the process flow. A State Machine relies on user input or in-state calculation to determine which state to go to next.

State Machines are most commonly used when programming user interfaces. When creating a user interface, different user actions send the user interface into different processing segments. Each of these segments will act as states in the State Machine. These segments can either lead to another segment for further processing or wait for another user event. In this example, the State Machine constantly monitors the user for the next action to take.

LabVIEW State Machines

Each state in a State Machine does something unique and calls other states. State communication depends on some condition or sequence. To translate the state diagram into a LabVIEW diagram, you need the following infrastructure:

- While loop – continually executes the various states
- Case structure – each case contains code to be executed for each state
- Strict Type Def Enum Control – used to represent the states
- Shift register – contains state transition information
- Transition logic – determines the next state in the sequence

LabVIEW Vending Machine

The LabVIEW Vending Machine Application is designed to accept change and distribute a soda when the change has reached the appropriate amount. This application was developed using a state machine diagram and LabVIEW state machine architecture.

LabVIEW Vending Machine State Machine

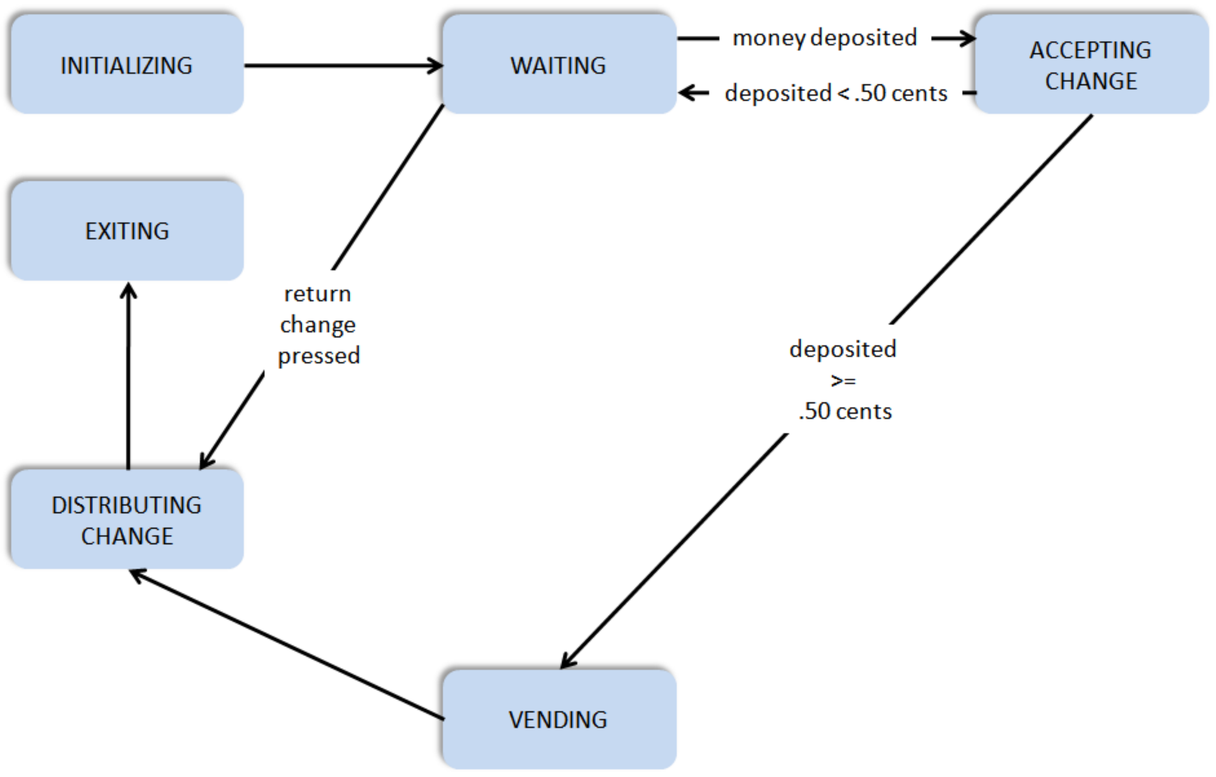
The LabVIEW Vending Machine application has the following requirements:

- All Soda products are sold for 50 cents.
- The machine only accepts nickels, dimes, and quarters.
- Exact change is not needed.
- Change can be returned at anytime during the process of entering coins.

LabVIEW Vending Machine State Diagram

For this example, our state diagram will have six states. The states we will be using are:

1.) **Initializing** – The Initializing state will initialize the Vending Machine
2.) **Waiting** – The Waiting state will wait until a user performs an action
3.) **Accepting Change** – The Accepting Change state will accept the deposited change and update the deposited amount in the state information. The transition logic for the next state is decided based on a comparison to the deposited amount and the cost of a soda. If the user has entered enough money, the next state will be Vending, otherwise the next state will be Waiting.
4.) **Vending** – The Vending state will distribute the soda and transition to the next state of Distributing Change.
5.) **Distributing Change** – The Distributing Change state will determine the change that is owed to the user and will distribute this amount.
6.) **Exiting** – The Exiting state will shut down and exit the Vending Machine application



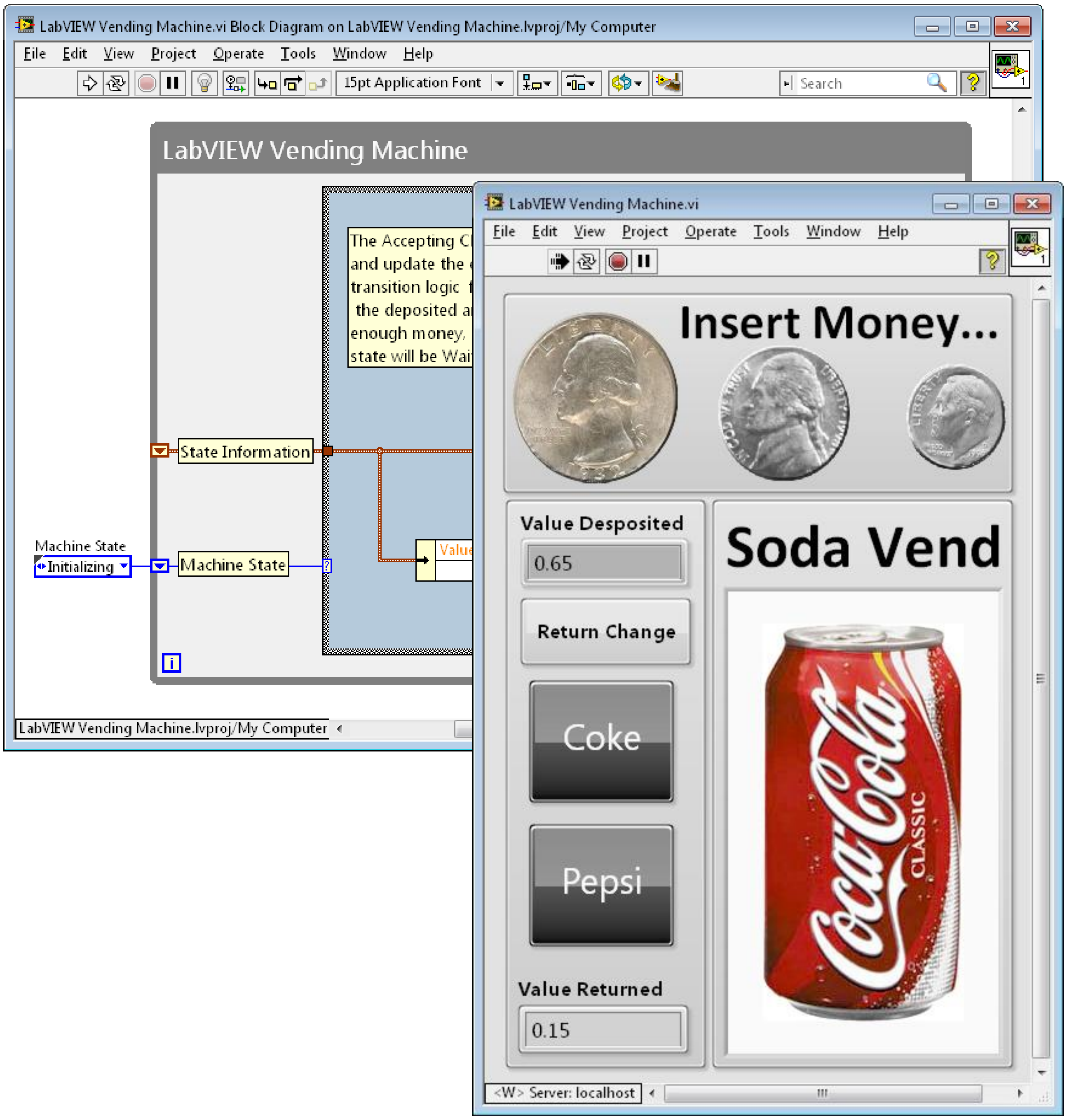
To download the exercise manual and LabVIEW code, go to http://bit.ly/StateMachineLV.

Evaluating the LabVIEW Vending Machine Code

To get started evaluating the behavior of the LabVIEW Vending Machine application, open the LabVIEW project **LabVIEW Vending Machine.lvprj** and then double-click the LabVIEW VI **LabVIEW Vending Machine.vi** in the project window. **Run** the LabVIEW Vending Machine application and complete the following series of actions: to observe the behavior of a soda being dispensed once the change reaches .50 cents.

a. Select a Quarter, Dime, Nickel, Quarter: You will notice that a Coke is dispensed and .15 cents is returned.

b. Select a Quarter, Dime, Nickel. Select the Pepsi button and then enter another Quarter: You will notice that a Pepsi is dispensed and .15 cents is returned.



Exercise 1: Add A New State to State Machine

Goal

We would like to add a new state to the LabVIEW state machine. This new state will accept a soda selection (Coke or Pepsi).

Scenario

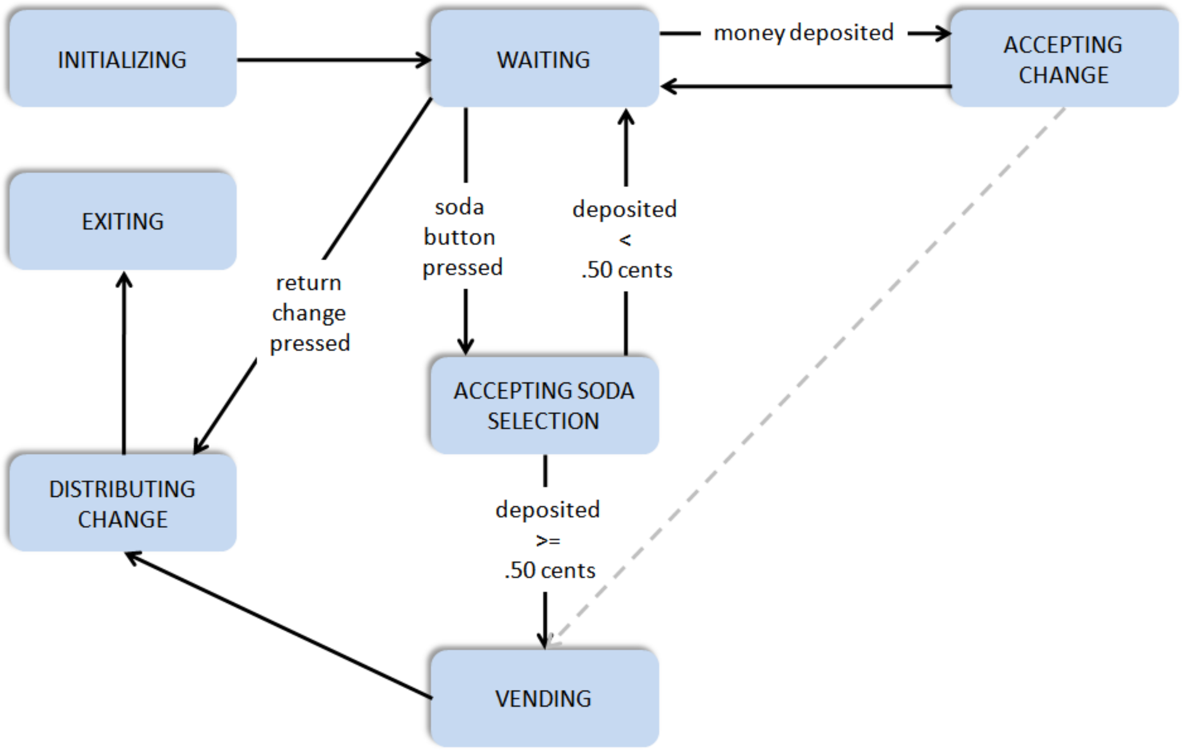
We have added two buttons to the LabVIEW application to allow the user to select between a Coke and a Pepsi, but have we do not have a state to represent the machine accepting the user soda selection. We would like to add a new state to the LabVIEW state machine. This new state will accept a soda selection (Coke or Pepsi). We will call this new state “Accepting Soda Selection”

Description

The State Machine.ctl is a Strict Type Def Enum Control that is being used in the LabVIEW Vending Machine application to transition between the states. Strict Type Def controls enforce that when any changes are made to the control file, all calling applications will bind and update to reflect these changes. This means that when we add a new item to the Enum to represent a new state, all references of this control in the LabVIEW Vending Machine application will be updated to include the new state as well.

New LabVIEW Vending Machine State Diagram

The Accepting Soda Selection state will accept the soda selection and determine if enough money has been deposited to receive this soda. The transition logic for the next state is decided based on a comparison to the deposited amount and the cost of a soda. If the user has entered enough money, the next state will be Vending, otherwise the next state will be Waiting.



Concepts Covered

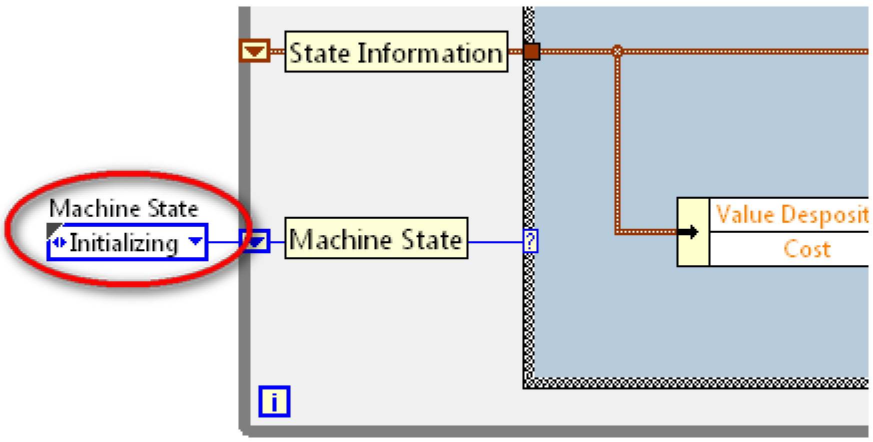
- Adding a new state to a LabVIEW state machine
- Editing Strict Type Def Enum Controls

Setup

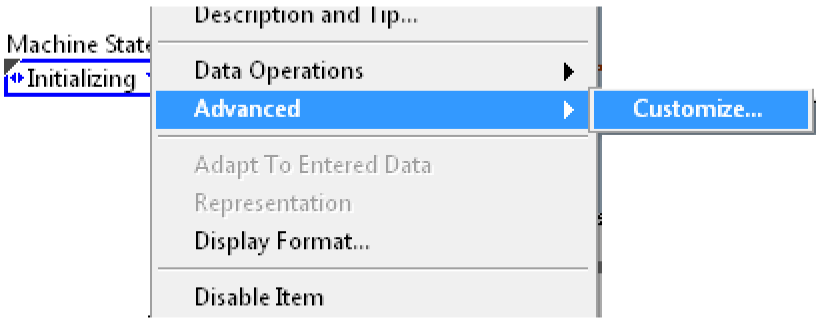
1. Open the **LabVIEW Vending Machine.vi** from the **LabVIEW Vending Machine.lvproj.**

2. On the Block Diagram, open the Machine State enum in which a new state should be added

    a. Locate the **Machine State.ctl** control on the LabVIEW Block Diagram.



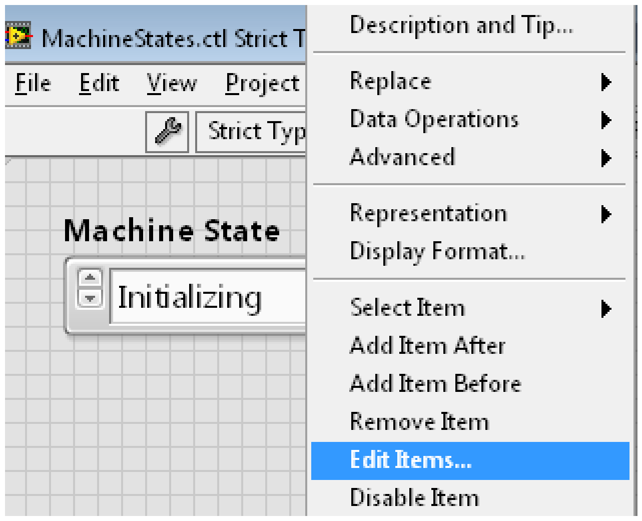
    b. To open the control for customizing, right-click on **Machine State.ctl** and select **Advanced»Customize.**



Note: *Machine State.ctl can also be accessed in the project Window.*

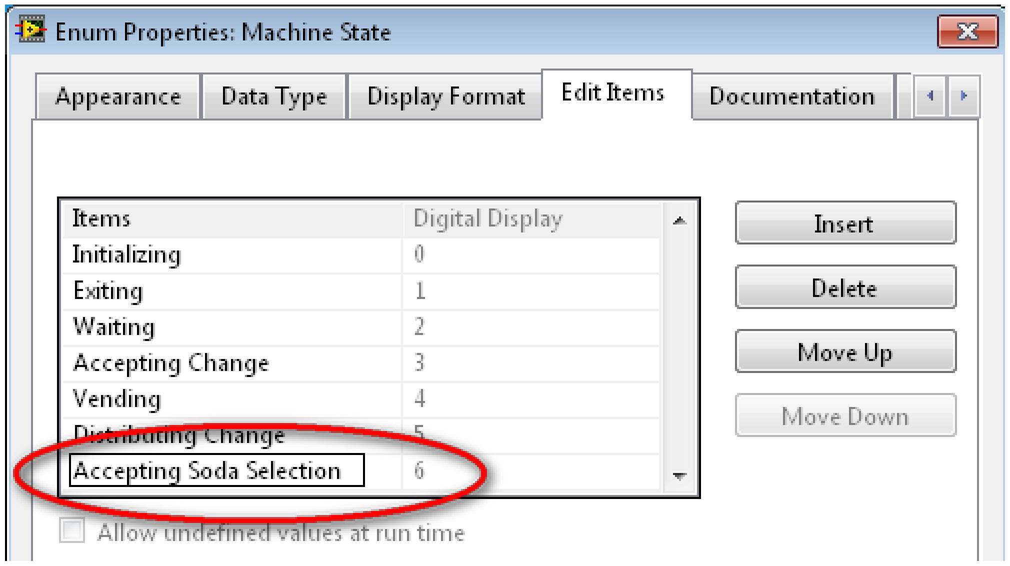
3. Edit the items in the **Machine States.ctl Strict Type Def Enum Control** to represent a new state of **"Accepting Soda Selection"**

   a. To open up the Enum Properties Window that will allow you to edit items in the Machine State Enum control, right-click and select **Edit Items.**



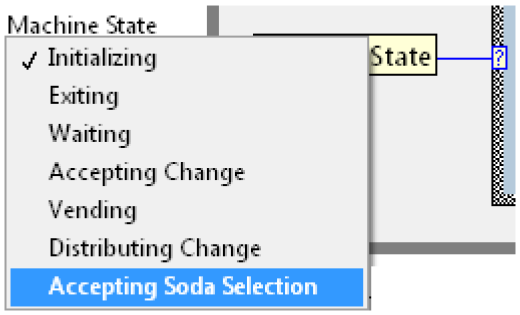
   b. Select the **Edit Items** tab In the Enum Properties: Machine State editor window.

   c. In the Edit Items Tab, double-click under "Distributing Change" to add a new entry at the bottom of the **Items** list. Name this new item **"Accepting Soda Selection"** Click **OK** when you are finished.



Items	Digital Display
Initializing	0
Exiting	1
Waiting	2
Accepting Change	3
Vending	4
Distributing Change	5
Accepting Soda Selection	6

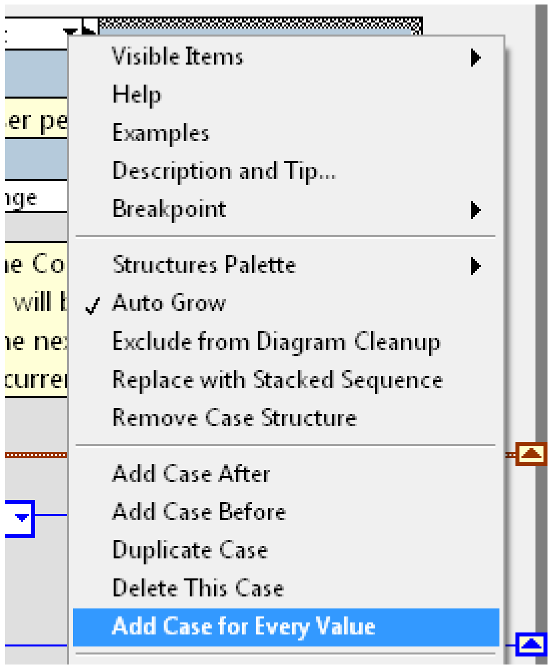
d. Save and close **Machine States.ctl.** Notice that when you click the Machine State control on the LabVIEW Vending Machine Block Diagram, you now have a new option for **"Accepting Soda Selection."**



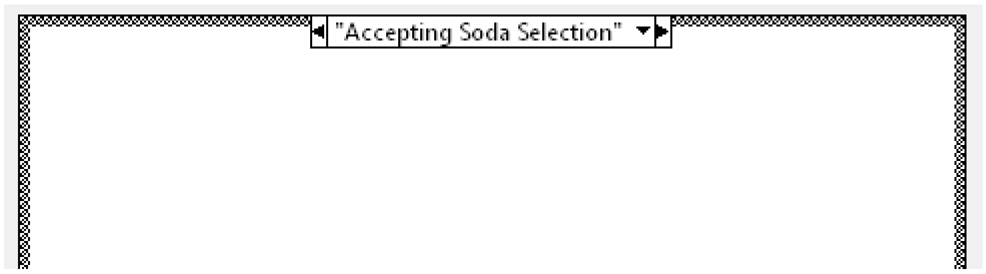
Important! *Ensure that you do not actually select the "Accepting Soda Selection" item from the Machine State control on the LabVIEW Vending Machine Block Diagram***.** *The beginning state should remain "Initializing."*

4. Add the new state **"Accepting Soda Selection"** to the Case Structure on the LabVIEW Vending Machine application. Adding this state to the Case Structure will add it to the implemented LabVIEW state machine.

a. To easily add the new state to the LabVIEW Vending Machine application, right-click the **Case Structure** on the LabVIEW Block Diagram and select **Add Case for Every Value.**



b. Notice that you can now select **“Accepting Soda Selection”** from the Case Structure. This is because this state has been added to the Case Structure through the changes made to the Strict Type Def Enum. Transition logic will be added to this state in the next Exercise.



EXERCISE 2: ADD TRANSITION LOGIC TO A NEW STATE

GOAL

Now that we have added the new **Accepting Soda Selection** state, we need to add transition logic to it.

SCENARIO

We have added the new **Accepting Soda Selection** state to the state machine and now we need to add transition logic to it. This new state will accept the soda selection and determine if enough money has been deposited to receive this soda. The transition logic for the next state is decided based on a comparison to the deposited amount and the cost of a soda. If the user has entered enough money, the next state will be Vending, otherwise the next state will be Waiting.

DESCRIPTION

This new state will accept the soda selection and determine if enough money has been deposited to receive this soda. The transition logic for the next state is decided based on a comparison to the deposited amount and the cost of a soda. If the user has entered enough money, the next state will be Vending, otherwise the next state will be Waiting.

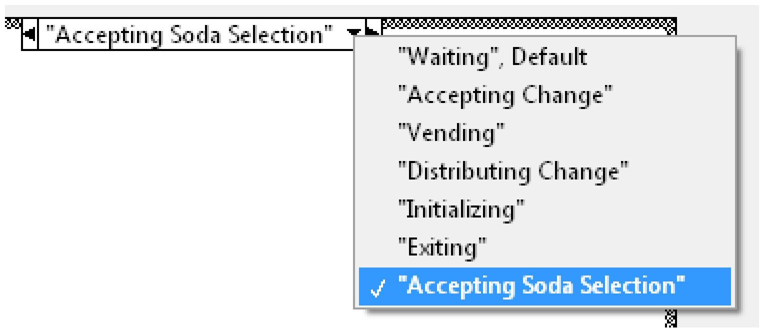
CONCEPTS COVERED

- Adding logic to a new state within a LabVIEW state machine

SETUP

1. Add transition logic to the **"Accepting Soda Selection"** state to determine the next state: "Vending" or "Waiting"

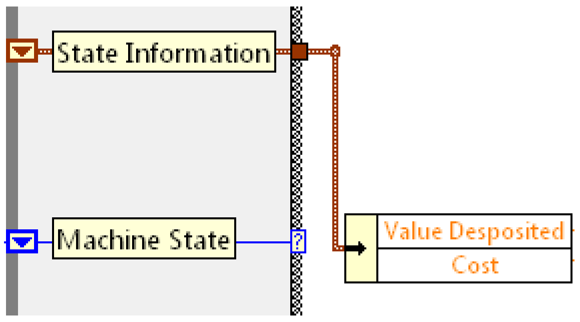
   a. Go to the Accepting Soda Selection state by clicking the Case Structure and selecting **"Accepting Soda Selection"**



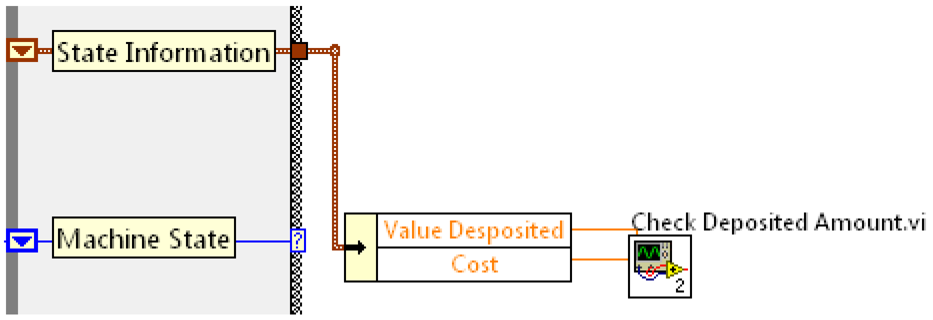
   b. The "**Accepting Soda Selection"** state will accept the soda selection and determine if enough money has been deposited to receive this soda. If the user has entered enough money, the next state will be Vending, otherwise the next state will be Waiting. The following code implements this transition logic and should be added to the "**Accepting Soda Selection**" state in the Case Structure.

i. Drop down the Unbundle by Name control. To do this, navigate to **Programming»Cluster, Class & Variant** and select the **Unbundle by Name** function. Drop this function on the Block Diagram in the "**Accepting Soda Selection** "state in the Case Structure.

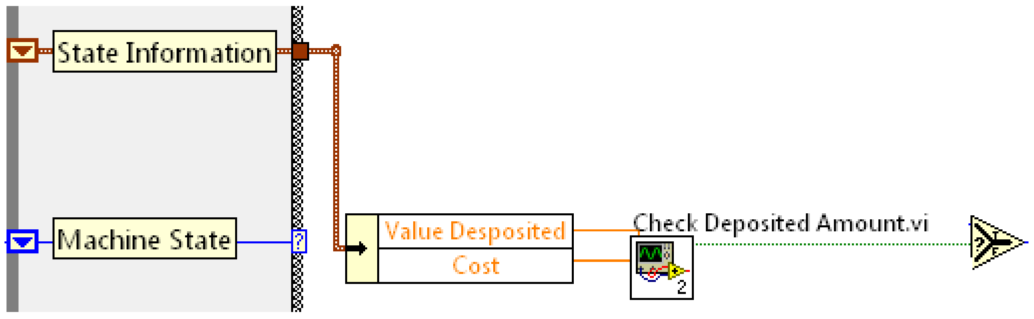
ii. Wire the **State Information data cluster** into an **Unbundle by Name** function and expand the **Unbundle by Name** function to expose the **Value Deposited** and **Cost** data.



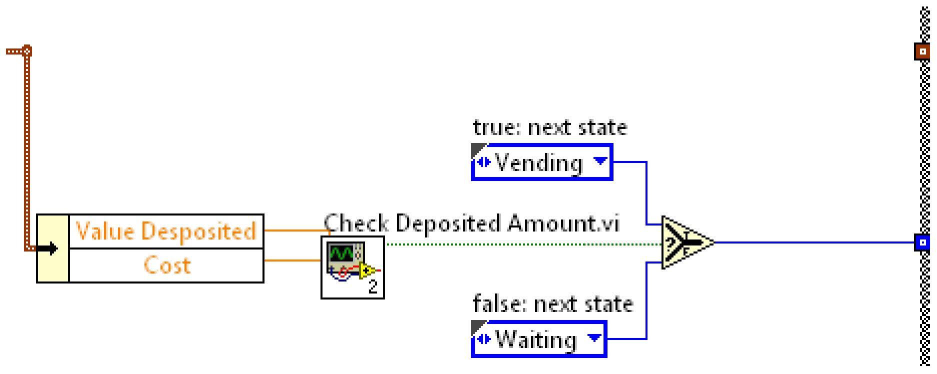
iii. Use the SubVI **Check Deposited Amount.vi** to compare **Value Deposited** and **Cost.** To do this, drag **Check Deposited Amount.vi** from the Project Explorer and drop it to the right of the Unbundle by Name function. Wire the **Value Deposited** and **Cost** values from the Unbundle by name into the **Check Deposited Amount.vi** inputs.



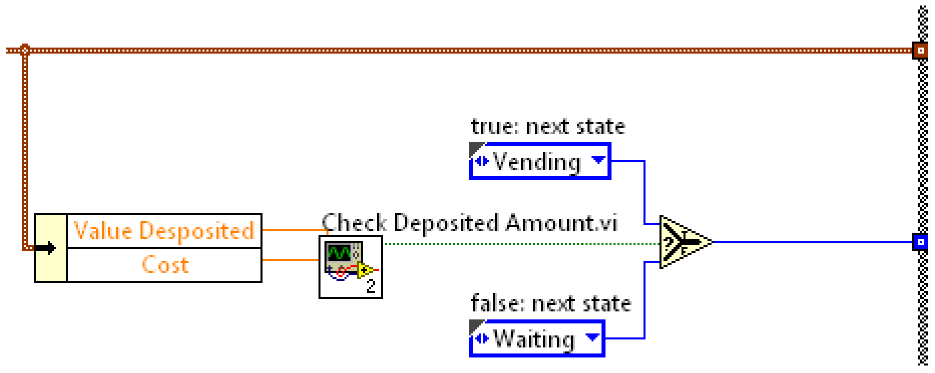
iv. Use the **Select** function to determine the next state. To do this, navigate to **Programming»Comparison** and select the **Select** function. Drop this function to the right of the **Great or Equal?** function. Wire the **output** of the SubVI **Check Deposited Amount.vi** into the **Select** function.



v. Use the **Machine States.ctl** controls to dictate the next state. To do this, drag two copies of the **Machine States.ctl** from the **Project Explorer** and drop them to the left of the **Select** function. Wire one into the top input terminal and the other into the bottom input terminal of the Select function. Select **Vending** on the top input and **Waiting** on the bottom input. Wire the output terminal of the **Select** function into the **Machine State** terminal.



vi. Wire the **State Information** cluster through the **Accepting Soda Selection** state in the Case Structure.



2. **Save** and **Run** the LabVIEW Vending Machine application with the new **"Accepting Soda Selection"** state and transition logic. While the application is running select any combination of change to be entered into the LabVIEW Vending Machine. Complete the following steps while the application is running to observe the behavior of a soda being dispensed once the change reaches .50 cents.

   a. Select a Quarter, Dime, Nickel, Quarter: You will notice that a Coke is dispensed and .15 cents is returned.

   b. Select a Quarter, Dime, Nickel. Select the Pepsi button and then enter another Quarter: You will notice that a Pepsi is dispensed and .15 cents is returned.

Exercise 3: Change the State Machine Order

Goal

Now that we have added the new **"Accepting Soda Selection"** state and implemented new transition logic to it, we need to change the transition logic in the Accepting Change state, Coke and Pepsi selections.

Scenario

Upon adding the new state "Accepting Soda Selection", the state machine transition logic and order should be modified.

Description

The **"Accepting Change"** state should be updated to transition to the next state of **Waiting**. The state machine will achieve the **"Accepting Soda Selection"** state when a user presses either the **Coke** or **Pepsi** button, so the transition logic in the Coke and Pepsi Events should be updated to reflect this change.

Concepts Covered

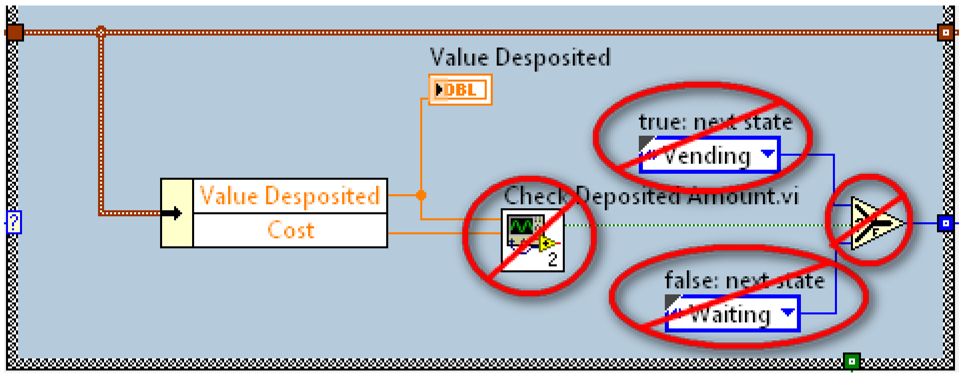
- Changing the transition logic of a LabVIEW state machine

Setup

1. Change the next state of **"Accepting Change"** to be **Waiting**

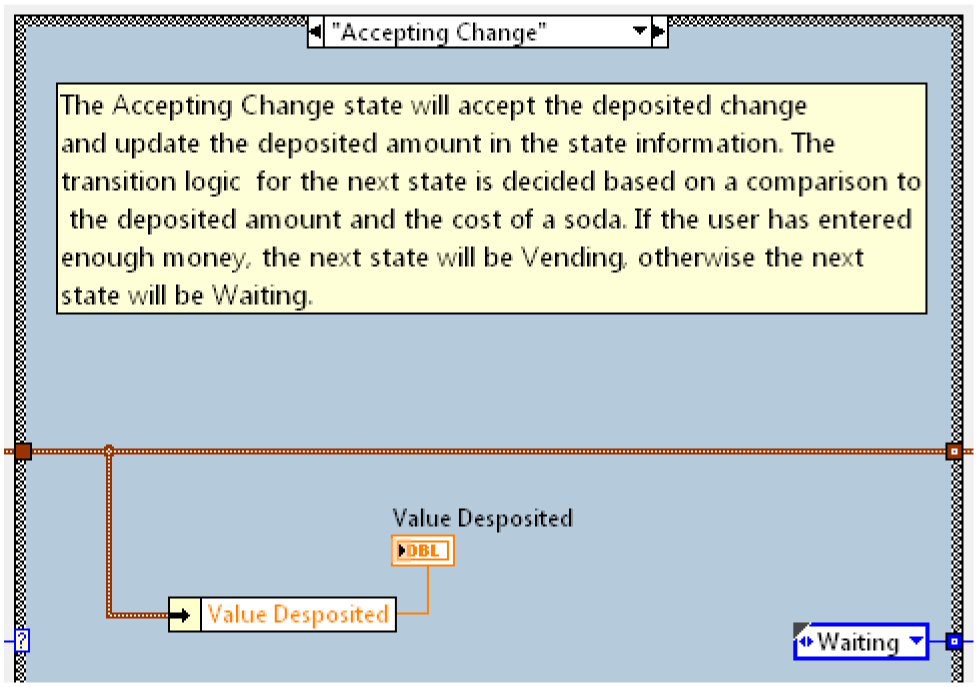
   a. Select the **"Accepting Change"** state in the Case Structure

   b. Delete the following functions:
   - **Check Deposited Amount.vi** (Sub VI)
   - **true:next** (Strict Type Def Enum)
   - **false:next** (Strict Type Def Enum)
   - **Select** (Boolean)



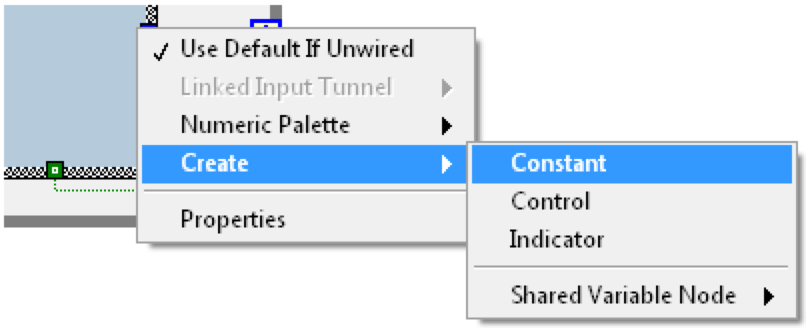
   Make sure to delete the broken wires after completing this.

   c. Collapse the **Unbundle by Name** function to <u>only</u> expose **Value Deposited**

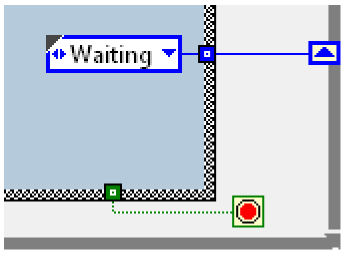


The Accepting Change state will accept the deposited change and update the deposited amount in the state information. The transition logic for the next state is decided based on a comparison to the deposited amount and the cost of a soda. If the user has entered enough money, the next state will be Vending, otherwise the next state will be Waiting.

d. Create the transition for the next state by right-clicking on the **Machine State exit terminal** and selecting **Create»Constant.**



e. Declare the transition logic to drive the next state to **Waiting**. To do this, change the **Machine State** constant value to be **"Waiting"**

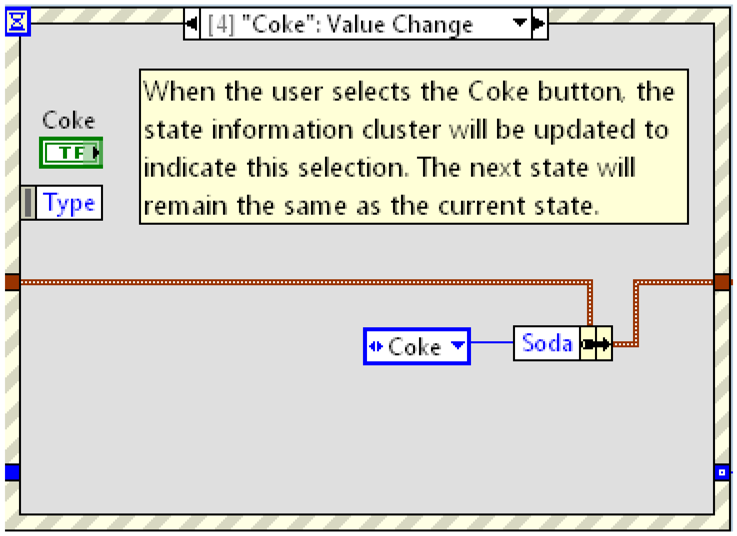


2. Change the Transition logic of the **Pepsi** and **Coke** buttons to dictate the next state of **"Accepting Soda Selection"**

   a. Select the **"Waiting"** state in the Case Structure.

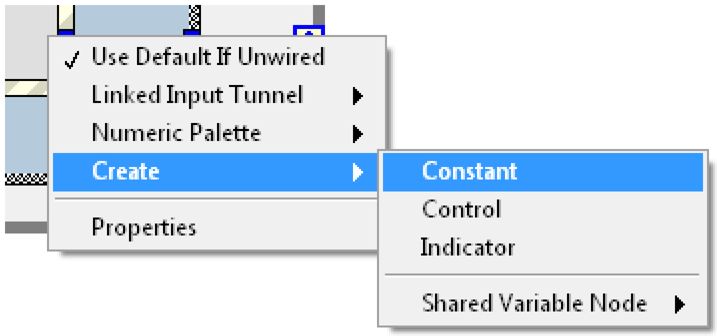
b. Select **“Coke”: Value Change** event in the Event Structure.

c. Delete the Machine State wire that is running through the **“Coke”: Value Change** event

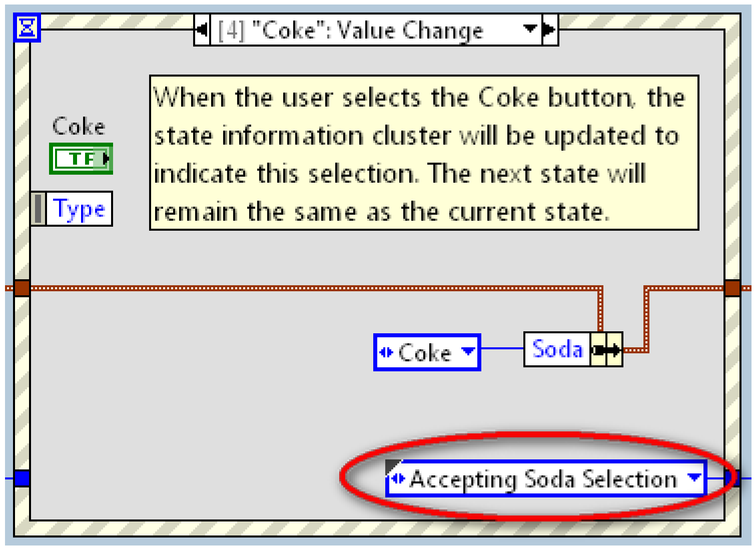


When the user selects the Coke button, the state information cluster will be updated to indicate this selection. The next state will remain the same as the current state.

d. Create the transition for the next state by right-clicking on the **Machine State exit terminal** and selecting **Create»Constant.**

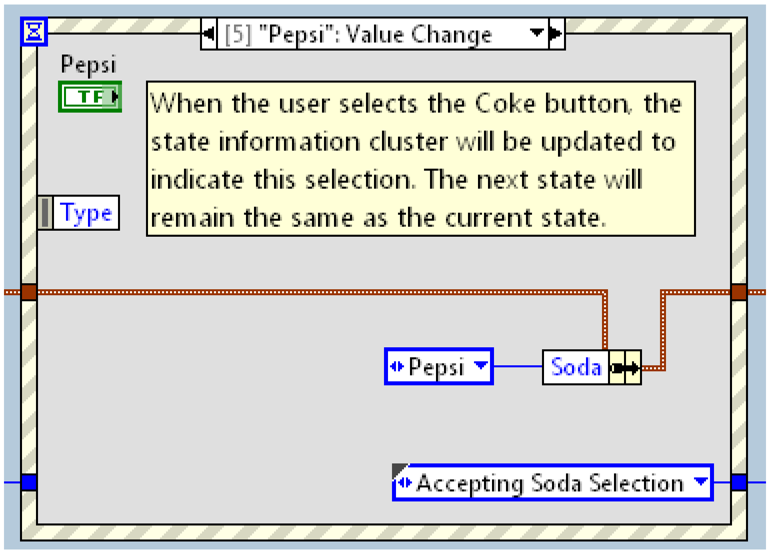


e. Declare the transition logic to drive the next state to **Accepting Soda Selection**. To do this, change the **Machine State** constant value to be **“Accepting Soda Selection”**



When the user selects the Coke button, the state information cluster will be updated to indicate this selection. The next state will remain the same as the current state.

f. Repeat **steps c-e** for the **"Pepsi": Value Change** event



When the user selects the Coke button, the state information cluster will be updated to indicate this selection. The next state will remain the same as the current state.

2. **Save** and **Run** the LabVIEW Vending Machine application with the new **"Accepting Soda Selection"** state and transition logic as well as modified transition logic to the Accepting Change state, Coke and Pepsi selections. While the application is running select any combination of change to be entered into the LabVIEW Vending Machine. Complete the following steps while the application is running to observe the behavior of a soda being dispensed once the user makes a soda selection and the change has reaches .50 cents.

   a. Select a Quarter, Dime, Nickel, Quarter. Then select the Coke button: You will notice that a Coke is dispensed and .15 cents is returned.

   b. Select a four Quarters. Select the Pepsi button: You will notice that a Pepsi is dispensed and .50 cents is returned.

Optional - Exercise 4: Implement LabVIEW State Machine

Goal

Implement a LabVIEW State Machine based on a State Diagram.

Scenario

You have a state diagram for a simple data acquisition application that you would like to implement in LabVIEW.

Description

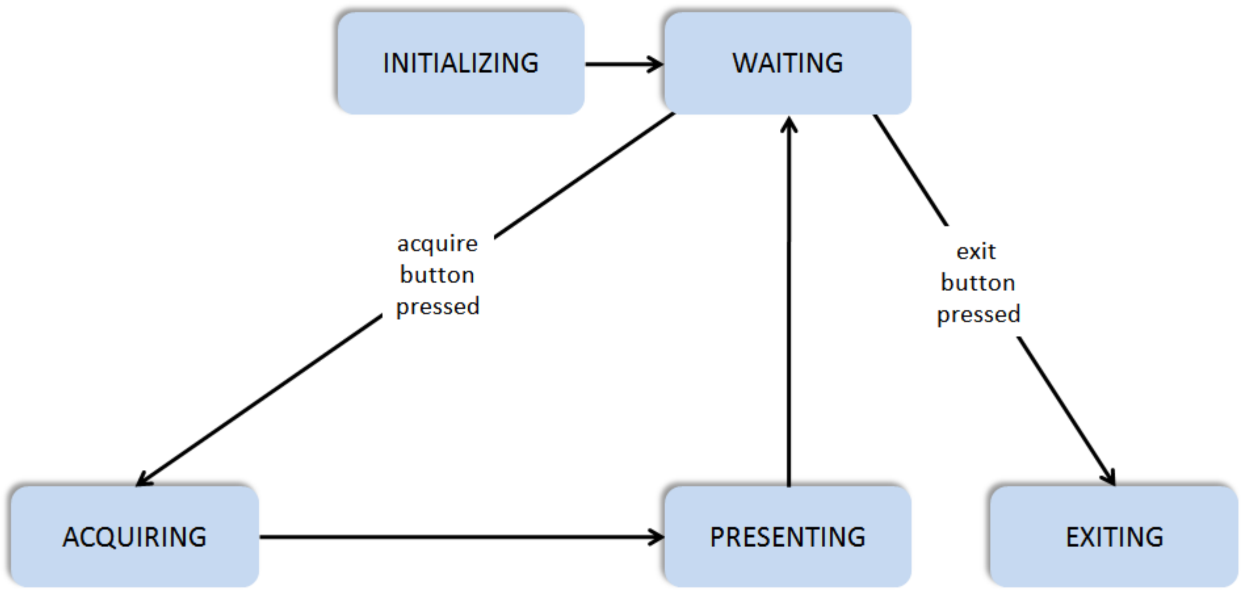
Evaluate the State Diagram provided for a simple data acquisition application and write a LabVIEW application that implements the program flow.

Concepts Covered

- Implementing a LabVIEW state machine from a state diagram

Setup

Use the following state diagram to implement a LabVIEW state machine to acquire data and present it.



Optional - Exercise 5: Add a New State to the Implemented LabVIEW State Diagram

Goal

Implement a new state in the LabVIEW State Machine based on a new state in the State Diagram.

Scenario

You have a state diagram and an implemented LabVIEW state machine for a simple data acquisition application, but you would like to add analysis to the LabVIEW state machine.

Description

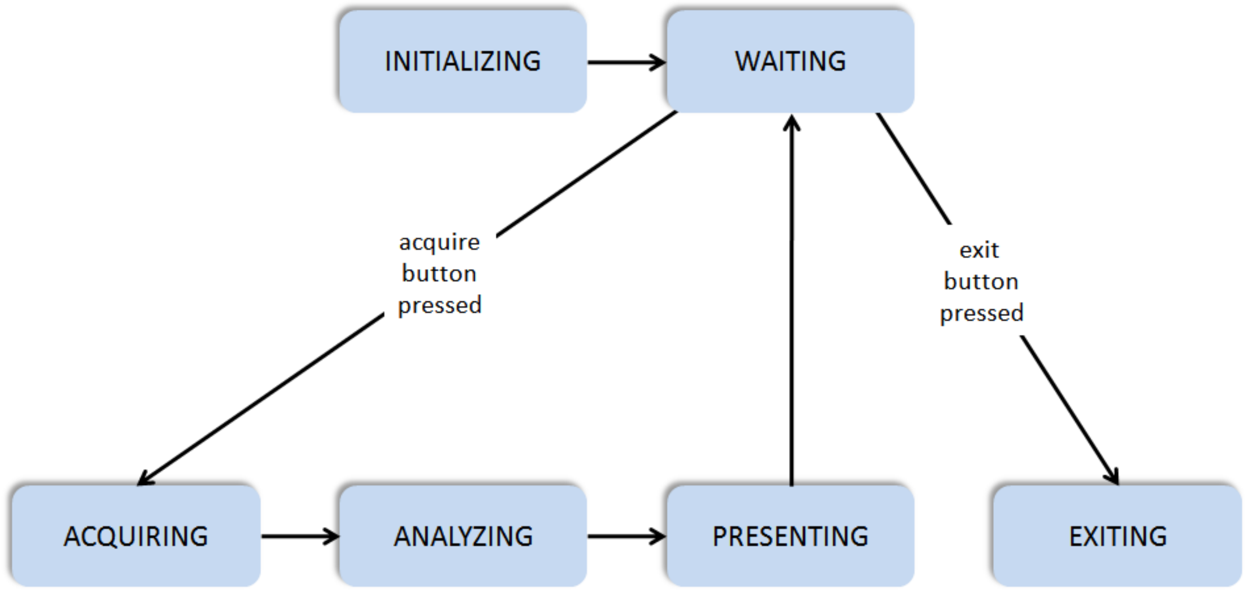
Evaluate the updated data acquisition State Diagram with a new Analyzing state. Update the implemented LabVIEW state machine to include the new Analyzing state. Ensure that you change the transition logic to necessary states.

Concepts Covered

- Adding a state to a LabVIEW state machine based on a new State Diagram

Setup

Use the following updated state diagram to add the new Analyzing state to the LabVIEW data acquisition state machine.


INITIALIZING
WAITING
acquire button pressed
exit button pressed
ACQUIRING
ANALYZING
PRESENTING
EXITING