PHYC 500: Introduction to LabView

M.P. Hasselbeck, University of New Mexico

Exercise 12 (v 1.1) State Machine

A State Machine is a powerful programming structure that provides logical branching to different operations. Use of a State Machine in LabView is very advantageous because it can make complicated programs easier to read and troubleshoot. In addition, the use of Type Definitions (Type Defs) give flexibility to expand and modify a VI with a minimal amount of effort. The State Machine should probably be the structure of choice for the majority of your new LabView programs.

A State Machine is almost always a better choice than a (Stacked/Flat) Sequence structure because:

• A State Machine allows decisions to change the program flow. The execution of a Sequence is fixed.

• A State Machine allows a specific window of code to run more often than others. All code must run the same in a Sequence.

• A VI can break out of a State Machine and abort. A Sequence must run to completion.

A State Machine in LabView has these basic components: i) While Loop, ii) Case Structure inside the While Loop, iii) at least one Shift Register. A State Machine is typically configured with an Enumerated control or "Enum". The Enum is setup as a Type Definition, usually called a "Type Def". With an Enum Type Def, the State Machine can be upgraded, expanded, and modified with minimal changes to the code. A Type Def is a custom control and the changes to it automatically appear throughout the VI. The above concepts will be demonstrated in this exercise.

Download the temperature.vi from the class website and save it to disk. Open LabView and select New: VI: From Template: Frameworks: Standard State Machine. In the Block Diagram, configure the While Loop to stop only with a Front Panel Boolean control. Right-click on the Enum control on the Shift Register input and Open Type Def. Configure this control for three states: Acquire, Analyze, and Alarm. Do this by right-clicking on the Enum control and editing the menu to make these changes. Select File: Apply Changes and then save this control to disk. Close the Type Def control.

Go back to the Block Diagram of the State Machine. Right-click on the Case Structure and select Add Case for Every Value. Scroll through the Case Structure to verify that all three Enum values specified in the Type Def are now present. Notice how the two default

states (Initialize and Stop) have been replaced. If a new state needs to be added or an existing one removed, the programmer only needs to open the Type Def to make the changes.

The Type Def Enum directs the VI to its next state. The various cases do not have to be arranged in a required order or sequence on the Block Diagram. Logical control is entirely determined by the value of the Enum constant that is connected to the Shift Register.

Configure the three states Acquire, Analyze, and Alarm. In the Acquire case, place the downloaded temperature VI (use Select a VI in the Block Diagram menu). Connect its output to i) a numeric indicator and ii) a Waveform Chart that displays individual points (not a continuous line). Set the next state to the Analyze case by selecting it on the Type Def connected to the output terminal of the Shift Register. Pass the temperature data to the Analyze case with a second Shift Register.

In the Analyze case, write code to set a Boolean TRUE condition if the temperature is above 25 or below 15. If TRUE, the next state is Alarm. If FALSE, the next state is Acquire. Branching into two different states requires two Type Def constants. Duplicate the Type Def (CTRL-drag) and use the Select function to set the next desired state that is connected to the Shift Register output terminal.

Go to the Alarm case and write code to blink a red LED on the Front Panel three times at 2 Hz. This is easily accomplished with a For Loop and a Shift Register with Boolean (T/F) data. On/off action is implemented with the Boolean NOT operation. Ensure that the LED remains dim when flashing is completed. Also set up an I32 indicator outside the Case Structure to count the alarm events. This will require a third Shift Register. An increment counter is available on the Numeric palette to simply this code.

Exit the Alarm case and go to Acquire. Note that all output tunnels and Shift Registers must be wired for the VI to run. If a state does not use Shift Register data, simply make a connection between the input and output. This passes the data through the program and makes it available to the next state. The VI will run indefinitely until the Stop button is pressed.

Save this VI as it will be used again.