Exercise 14-1: Creating a State Machine

Goal

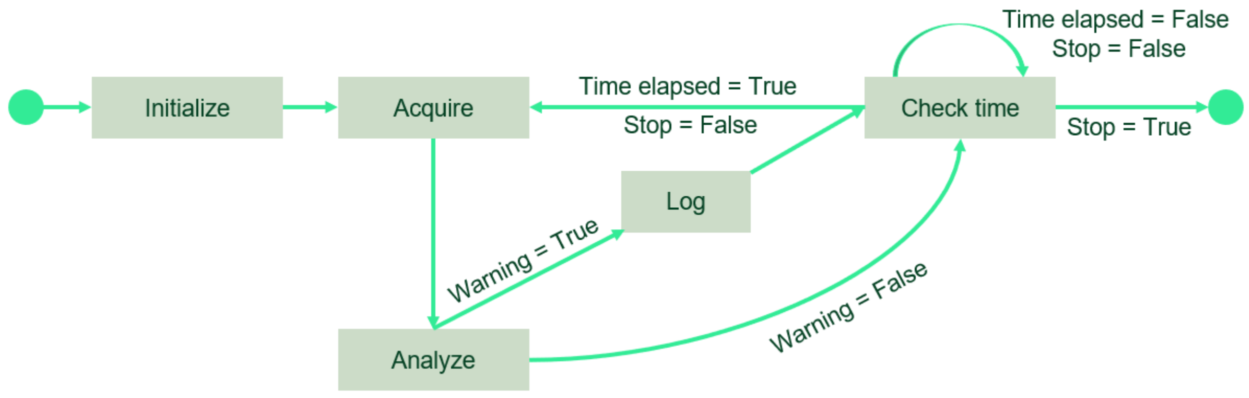
- Create a VI that implements a state machine using a type definition enum.

Scenario

You must design a VI for a user interface state machine. The VI acquires a temperature every half a second, analyzes each temperature to determine if the temperature is too high or too low and alerts the user if there is a danger of heatstroke or freeze. The program logs the data if a warning occurs. If the user has not clicked the **Stop** button, the entire process repeats. The state machine must also allow for expansion, because processes may be added in the future.

Design

Use a state machine to create the VI in this exercise. The state transition diagram in the following figure describes the logic for this application.



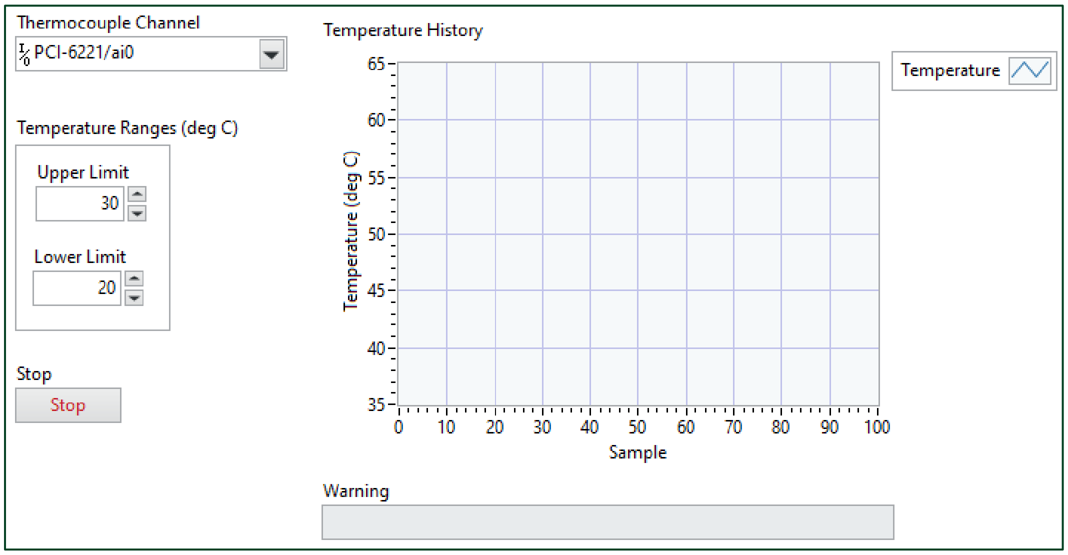
The following table describes the states in this state machine.

State	Description	Next State
Acquire	Set time to zero and acquire data from the temperature sensor	“Analyze”
Analyze	Read front panel controls and determine the warning level	“Log” if a warning occurs, “Check Time” if no warning occurs
Log	Log the data panel in a tab-delimited ASCII file	“Check Time”
Check Time	Check whether time elapsed is greater than or equal to 0.5 seconds	“Acquire” if time has elapsed, “Check Time” if time has not elapsed

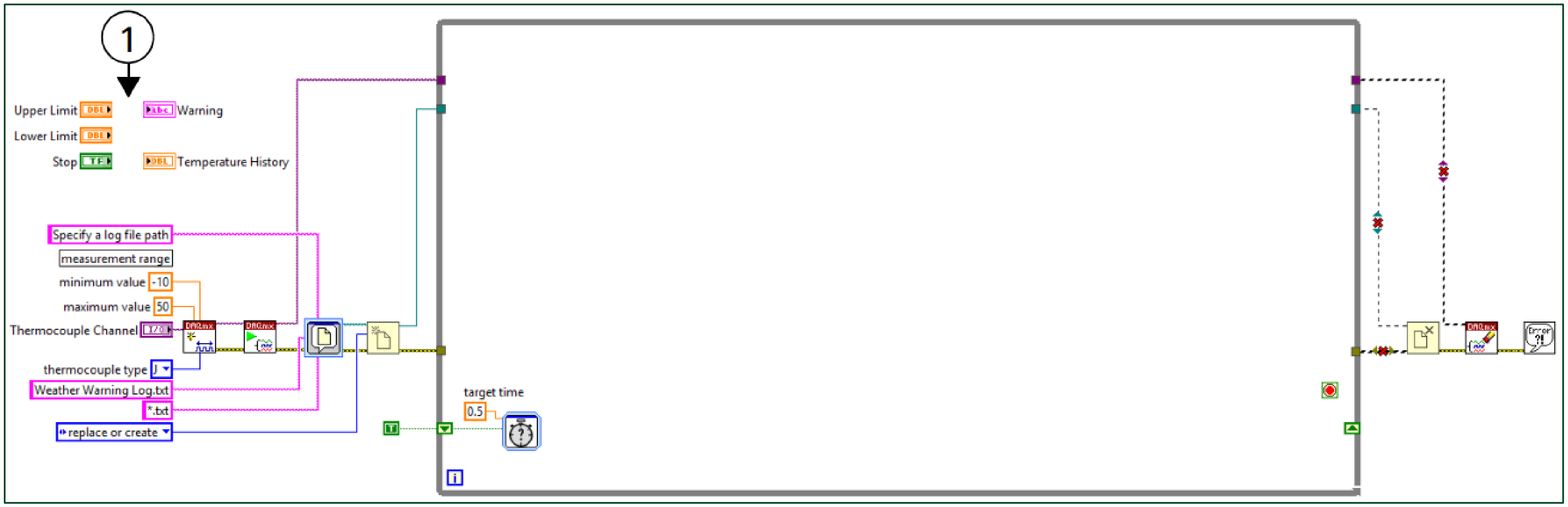
Implementation

1. Open `Weather Station.lvproj` in the `C:\Exercises\LabVIEW Core 1\State Machine` directory.
2. Open the Weather Station UI VI from the **Project Explorer** window.

   The figure below shows the front panel of the Weather Station UI VI that has been provided for you. You will modify the block diagram to create a state machine for the weather station.

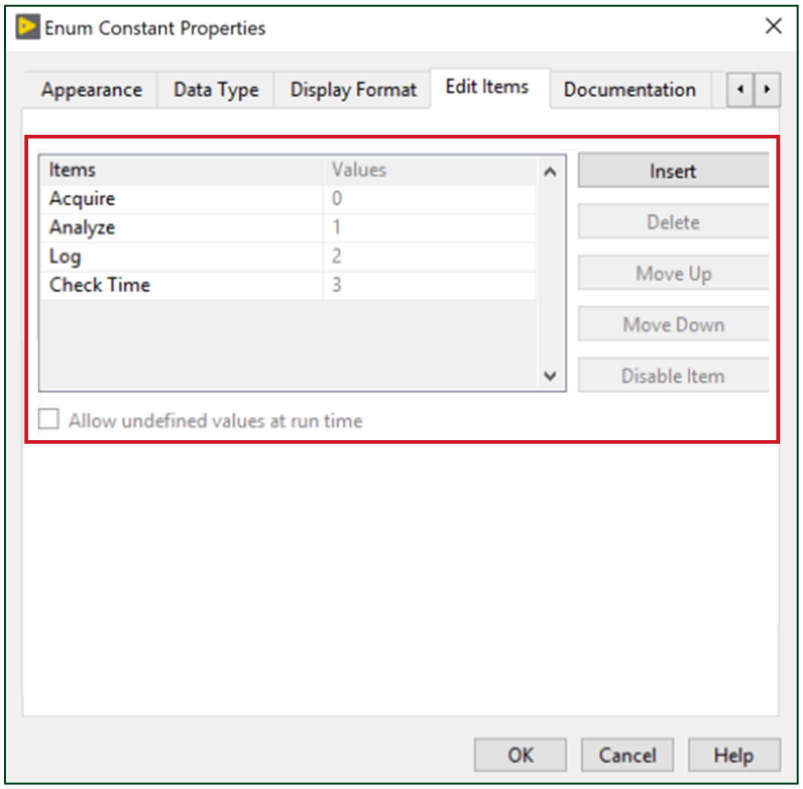


The following figure shows the starting point of the block diagram for the Weather Station UI VI. You will have to edit this block diagram to implement a state machine for the weather station application.



1. These are the unused controls and indicators from the front panel. You will use these controls and indicators to program different cases.

3. Create a new type definition to control the weather station application.
   - Open the block diagram and create an enum constant to the left of the While Loop.
   - Right-click the **enum constant** and select **Edit Items** to open the **Enum Constant Properties** window. On the **Edit Items** tab, add the enum items, as shown in the following figure.

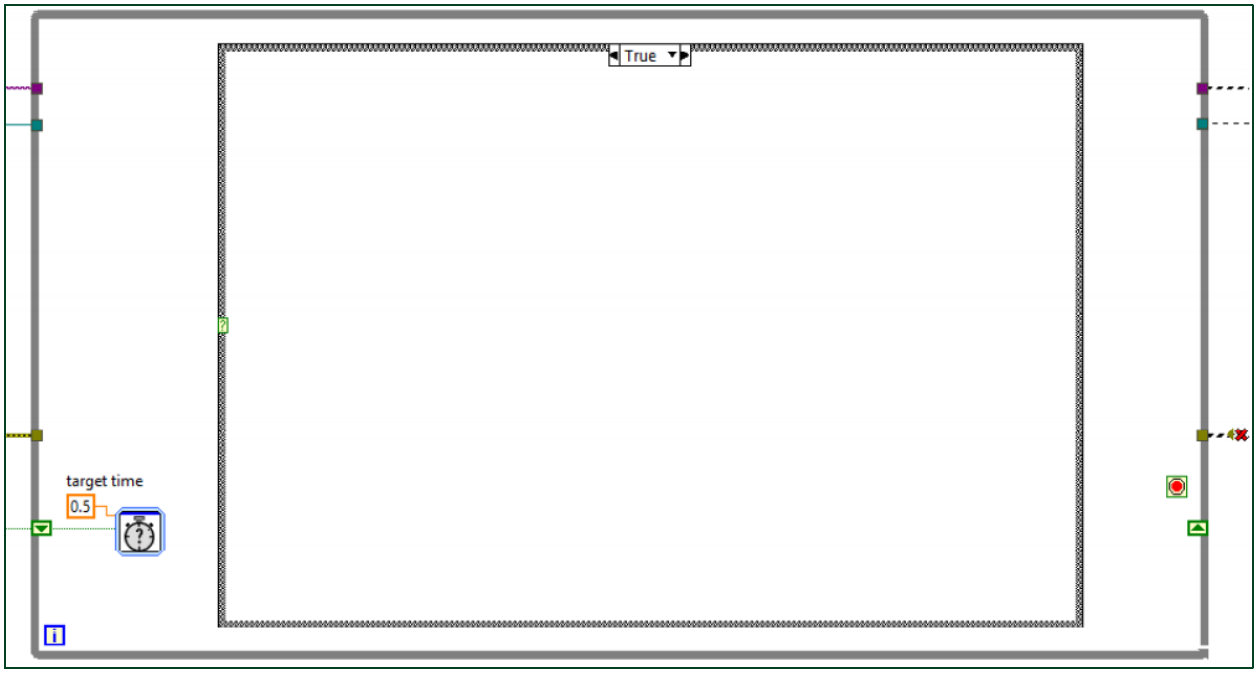


Items	Values
Acquire	0
Analyze	1
Log	2
Check Time	3

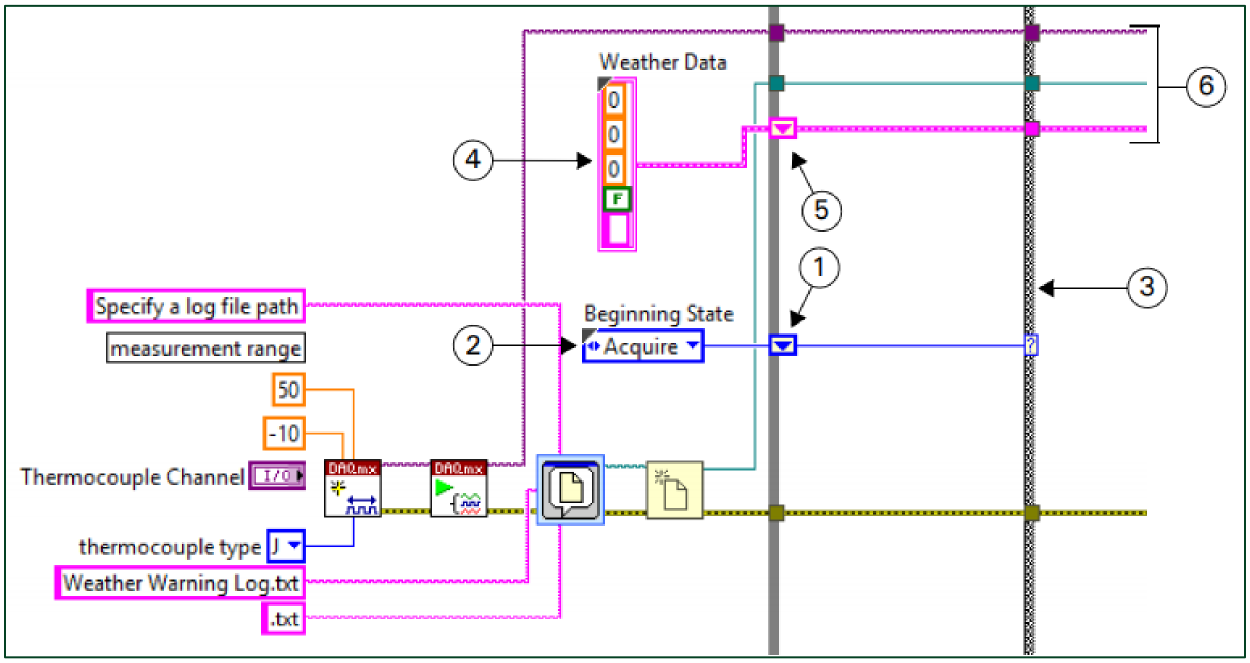
   - Right-click the **enum constant** on the block diagram and select **Make Type Def**.

4. Modify the new type definition and add it to the Weather Station project.
   - Right-click the **enum constant** and select **Open Type Def**.
   - Rename the label on the enum control to `States`.
   - Apply the changes made to type definition (File»Apply Changes).
   - Save the type definition as `Weather Station States` in the `C:\Exercises\LabVIEW Core 1\State Machine\Supporting Files` directory.
   - Close the **Type Def** window.
   - In the **Project Explorer** window, drag the Weather Station States type definition to your Supporting Files folder.

5. Place a Case structure inside the while Loop, as shown in the following figure. This is the part of the State Machine design pattern.



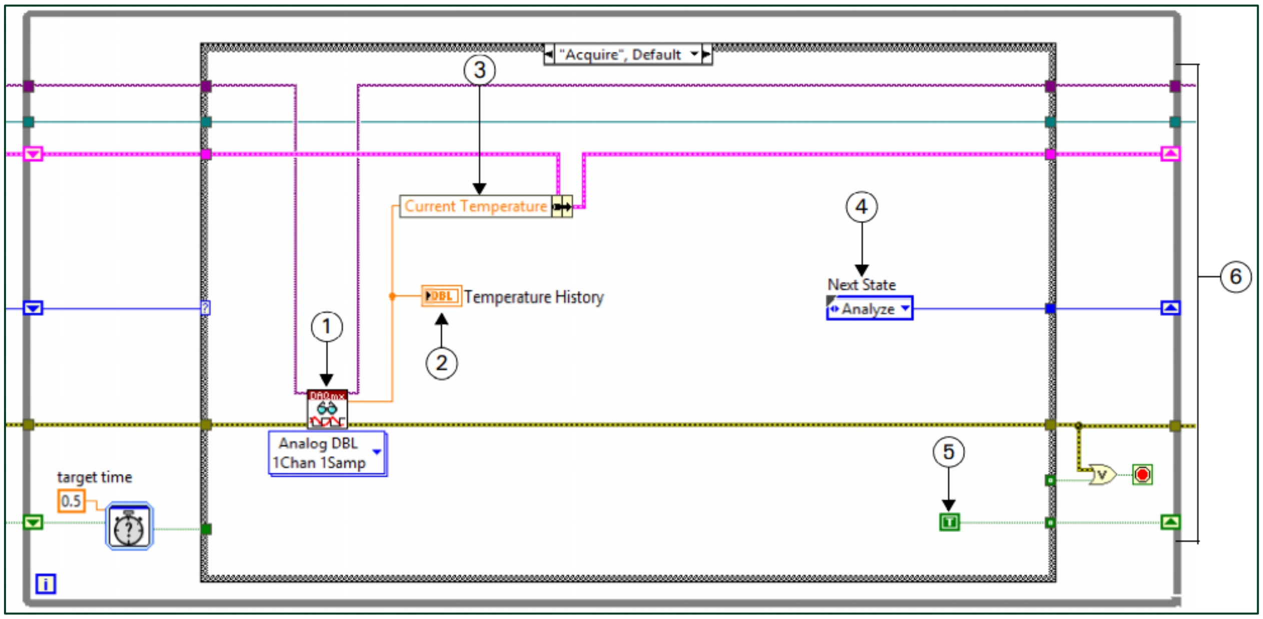
6. Control the state machine with the type definition enum and update the frame work as shown in the following figure.



1. **Shift Register** – Right-click the **While Loop** and select **Add Shift Register**.
2. **Type definition enum constant** – In the **Properties** window, enable the **Show Label** checkbox. Rename the label as `Beginning State`. Wire the **Beginning State** constant to the shift register to initialize the shift register to the Acquire state. Wire the shift register to the **Selector** input of the Case structure.
3. Add more cases. Right-click the **Case structure** and select **Add Case For Every Value** to create different cases for each value in the enum.
4. **Weather Data** – Drag the **Weather Data** type definition from the **Project Explorer** window to the block diagram to create a type definition cluster constant.
5. **Shift Register** – Place another shift register on the While Loop and wire the **Weather Data** constant to it.
6. **Wire as shown**.

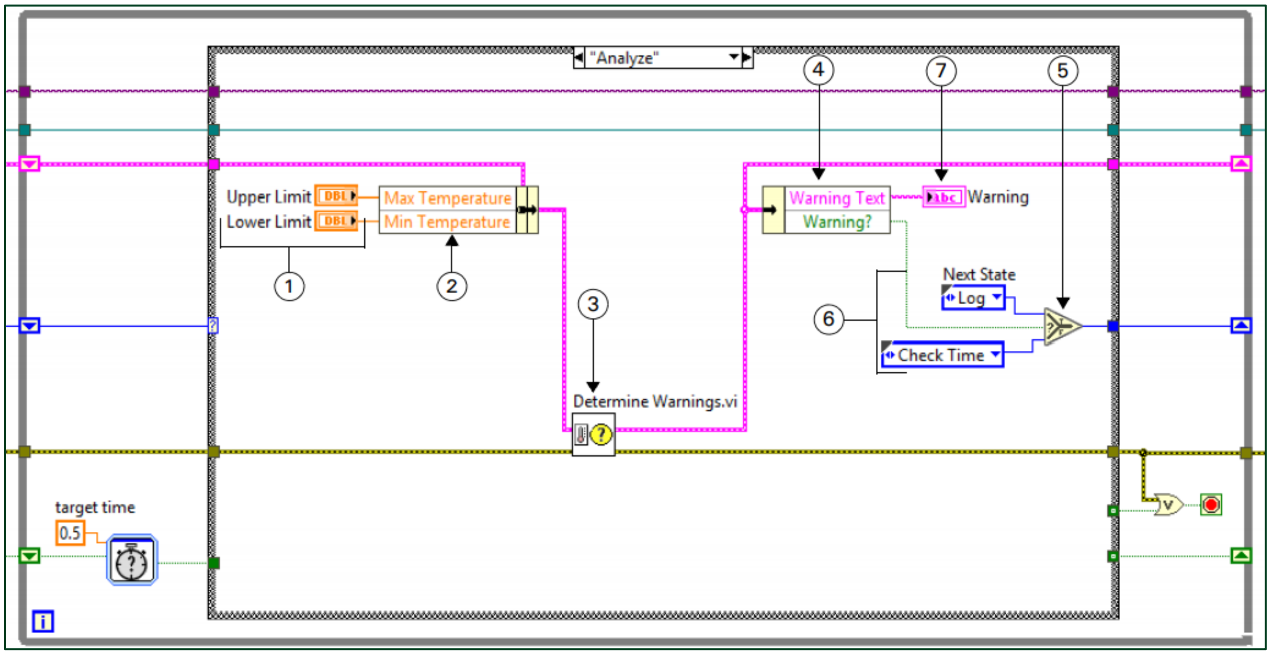
Note: After you finish wiring the Acquire case in the following step, some tunnels are empty because not all cases are wired yet.

7. Complete the Acquire state as show in the following figure.



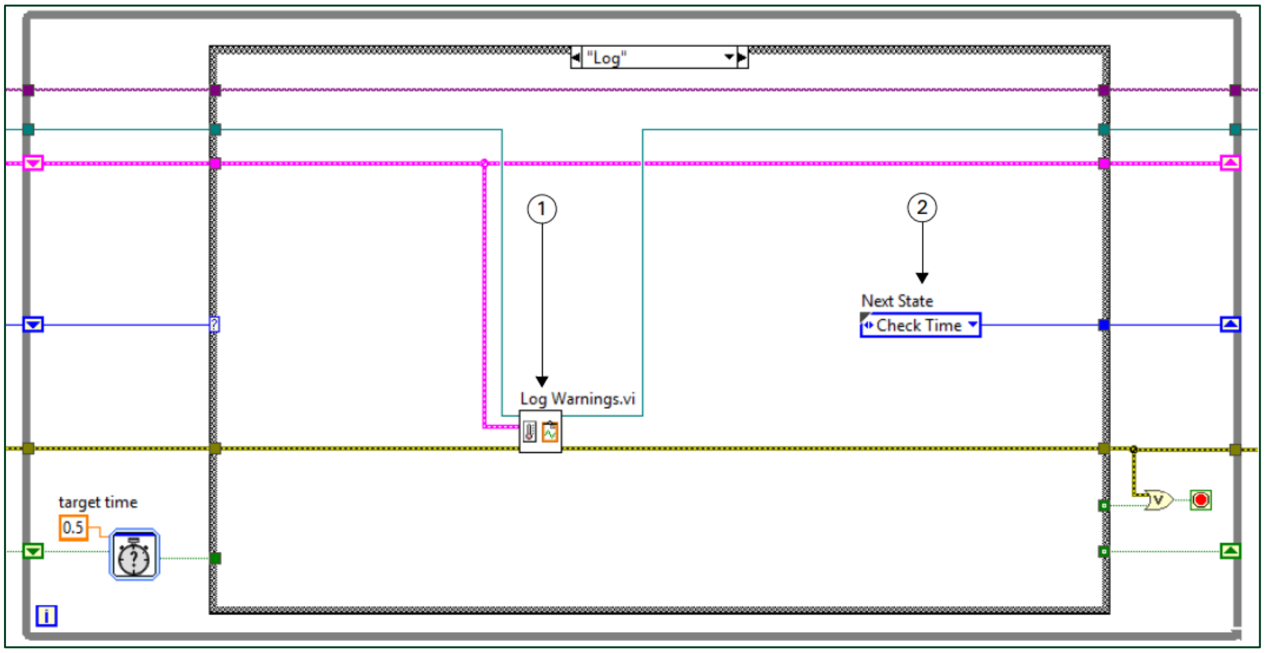
1. **DAQmx Read** – In the Polymorphic VI Selector, set the following values:
   - **Channel Type**: Analog
   - **Channel Count**: Single Channel
   - **Sample Count**: Single Sample
   - **Data Format**: DBL
2. **Temperature History** – Move this indicator into the Acquire state of the Case structure.
3. **Bundle By Name** – Set the element to **Current Temperature**. Wire the data output of the DAQmx Read VI to the **Current Temperature** input.
4. **Next State enum** – Press <Ctrl> and click the **Beginning State** enum and drag a copy into the Acquire case. Rename this copy of the Weather Station States type definition as `Next State`. Set the enum to **Analyze** and wire it through a tunnel on the Case structure to the shift register on the While Loop.
5. **TRUE Constant** – Create a **TRUE** constant and wire it through the Case structure to the **Elapsed Time** shift register. The **TRUE** constant resets the **Elapsed Time** counter every time the VI executes the Acquire case.
6. Wire the DAQmx task, file reference, **Weather Data** cluster, enum, error, and Boolean through the block diagram to the corresponding shift registers as shown.

8. Complete the Analyze case as shown in the following figure.



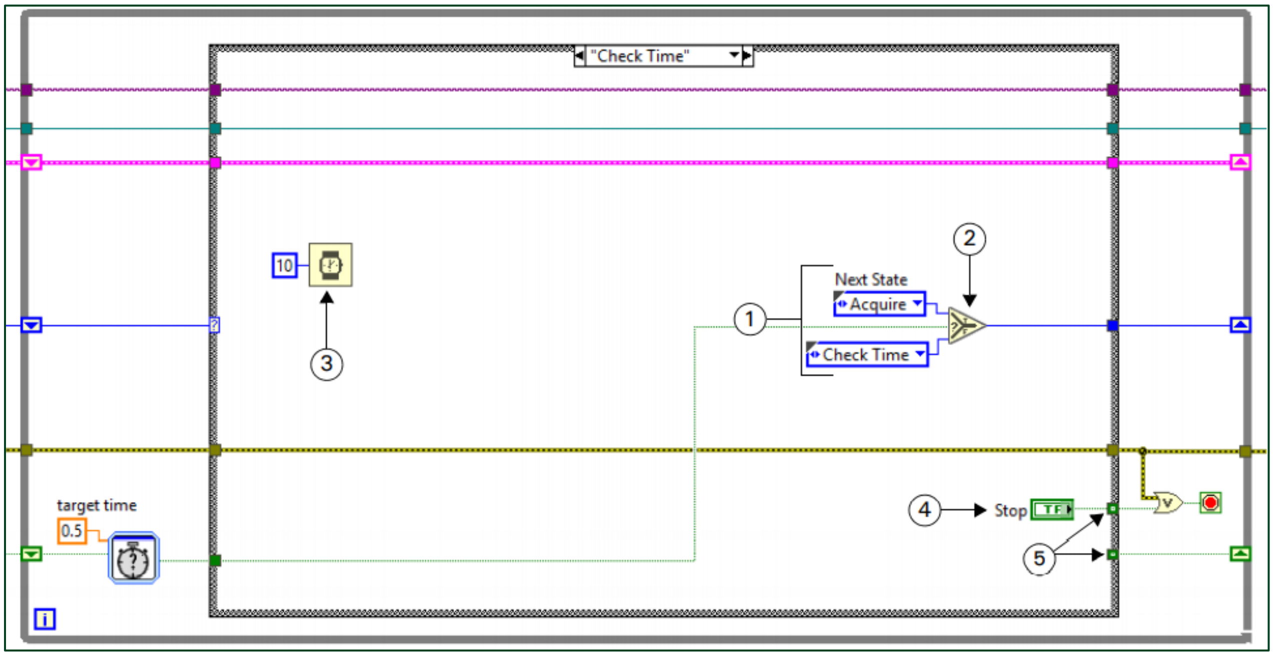
1. **Upper Limit and Lower Limit –** Move these controls into the Analyze state of the Case structure.
2. **Bundle By Name** – Replaces the **Max Temperature** and **Min Temperature** items with the values from the **Upper Limit** and **Lower Limit** controls. The Bundle By Name function makes it possible to wire the **Upper Limit** and **Lower Limit** values to the **Weather Data** In input of the Determine Warnings subVI.
3. **Determine Warnings** – In the **Project Explorer** window, find Determine Warnings VI inside the Supporting Files folder and drag it to the block diagram.
4. **Unbundle By Name –** Returns the values of specific items from the cluster.
5. **Select –** Determines which state to execute next depending on whether or not a warning occurs.
6. **Weather Station States –** Wire two copies of the **Weather Station States** type definition enum renamed as `Next State` to the Select function. You can create these copies from the **Beginning State** enum.
7. **Warning –** Move this indicator into the Analyze state of the Case structure.
8. Wire the DAQmx task and file reference through the case as shown.

9. Complete the Log case as shown in the following figure.



1. **Log Warnings** – From the **Project Explorer** window, find Log Warnings VI inside the Supporting Files folder and drag it to the block diagram. This subVI logs weather data to file.
2. **Next State** – Create a copy of the **Weather Station States** type definition enum, label it `Next State`, and set the next state to **Check Time**.
   Wire the DAQmx task and **Weather Data** cluster through the case as shown.

10. Complete the Check Time case as shown in the following figures.



1. **Next State** – Wire two copies of the **Weather Station States** type definition enum to the Select function.
2. **Select –** Determines which state to execute next depending on whether or not time has elapsed.
3. **Wait (ms) –** This function allows the CPU to do other tasks while this VI repeatedly executes the Check Time case until the Elapsed Timer VI returns a TRUE value when 0.5 seconds have elapsed.
4. **Stop –** Move the **Stop** button terminal from outside the While Loop. Wire the Stop button terminal to the Or function outside of the Case structure.
5. **Use Default if unwired** – Right-click these **tunnels** and select **Use Default If Unwired**. These Boolean tunnels will output a value of FALSE in cases where these tunnels are unwired.

   Wire the DAQmx task, file reference, and **Weather Data cluster** through the case as shown.

11. Save and test the VI.

Test

1. Run The VI.
   - Name the log file `Weather Warning Log.txt` when prompted.
   - Enter values for the **Upper Limit** and **Lower Limit** controls and observe the behavior of the VI. Does it behave as expected?
2. Stop the VI.
3. Navigate to the `Weather Warning Log.txt` file and open it.
4. Notice the changes in the upper and lower limit values and the placement of tabs and line breaks.
5. Close the log file.
6. Save and close the VI and the project.

On the Job

1. Back at your job, can your application's logic be described by a state transition diagram? If so, then draw the state transition diagram below:

   ____

   ____

   ____

2. Will you use a state machine to implement your state transition diagram in LabVIEW code?

   ____

   ____

   ____

End of Exercise 14-1