Homework 4 Guide

Background

In this homework, you will create a state machine and use an event structure to help determine your states. If you are not taking the CLAD add-on part of the workshop, you probably haven't heard of event structures; don't worry about it. They are a more advanced concept but if you follow along step by step, you should be able to get the correct solution anyhow.

Task

This homework is an alarm clock application that allows you to set an alarm, turn the alarm off, and hit snooze to get an extra five minutes of rest.

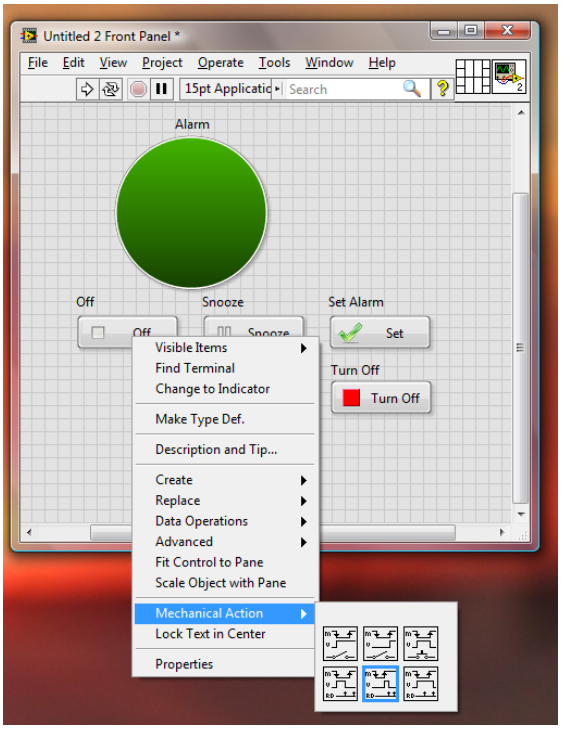
Steps

Front Panel

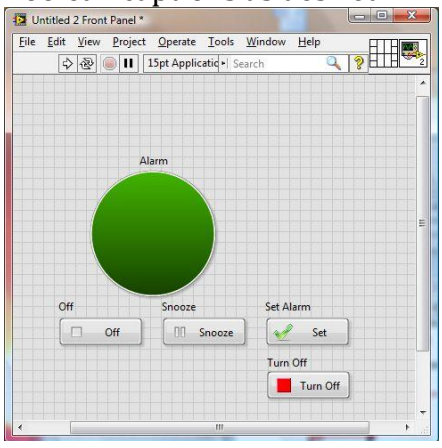
- Note: This homework uses silver controls because of how they look; modern controls could be used if preferred.

1) Create a Boolean Indicator to indicate when the alarm is turned on
    a. You can choose to make a customized one (recommended) but the one shown is found under Controls> Silver> Boolean> LED (silver)

    b. Enlarge the Boolean and rename it "Alarm"
2) Create Boolean Controls for the following tasks: Set Alarm, Off, Snooze, and Turn Off
    a. The choice each Boolean is your own, but make sure that the mechanical action is Latch When Released



b. The Booleans that are shown are found under Controls> Silver> Boolean> Buttons> OK Button, Media Stop, Pause, and Stop Button respectively
c. Rename the labels of the Booleans to Set Alarm, Off, Snooze, and Turn Off
d. Rename the Boolean captions as desired

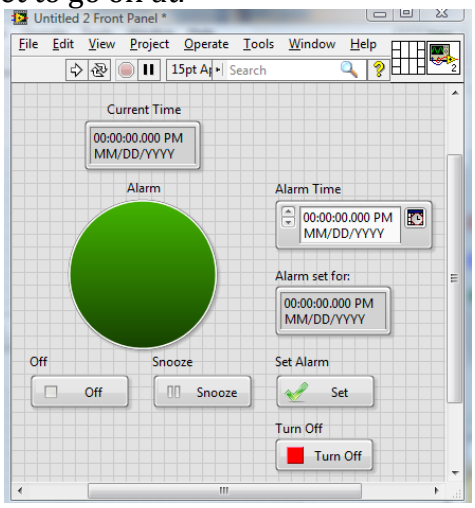


3) Create an indicator for the current time and what time the alarm is set for
    a. Place a Time stamp control on the front panel
        - Go to Controls> Silver> Numeric> Time Stamp Control (silver)
    b. Right click and select “Change to Indicator”
    c. Rename the Indicator “Current Time”
    d. Create a copy of the control (Ctrl-drag)
    e. Rename the copy “Alarm set for:”
4) Create a controls and indicators for the time
    a. Place a Time stamp control on the front panel

- Go to Controls> Silver> Numeric> Time Stamp Control (silver)

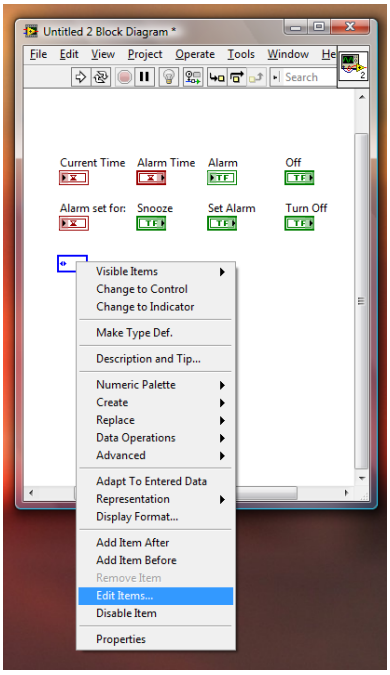
b. Rename the control “Alarm Time”. This will be where the user can enter then alarm time that they desire.

c. Do this for the “Current Time” (indicator) and “Alarm set for:” (indicator). “Current Time” will display to the user the current time, and “Alarm Set for:” will display to the user the time that the alarm is currently set to go off at.



Block Diagram

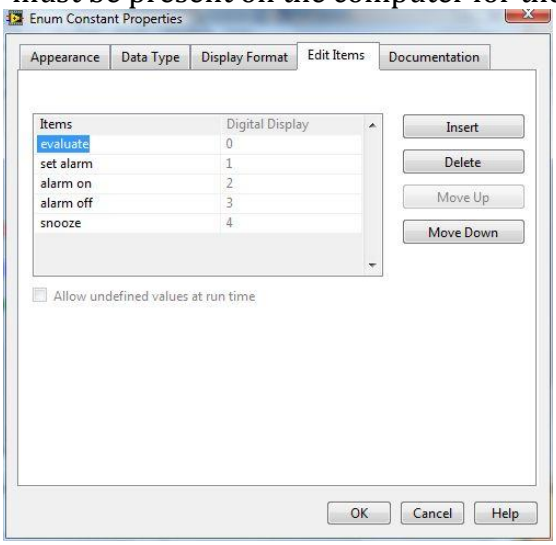
1. Create the framework of a state machine
   a. Create an enum that contains all of the desired states and make it a type def
      i. Create an enum (Functions> Programming> Numeric> Enum Constant
      ii. Right Click the enum and select “Edit Items”



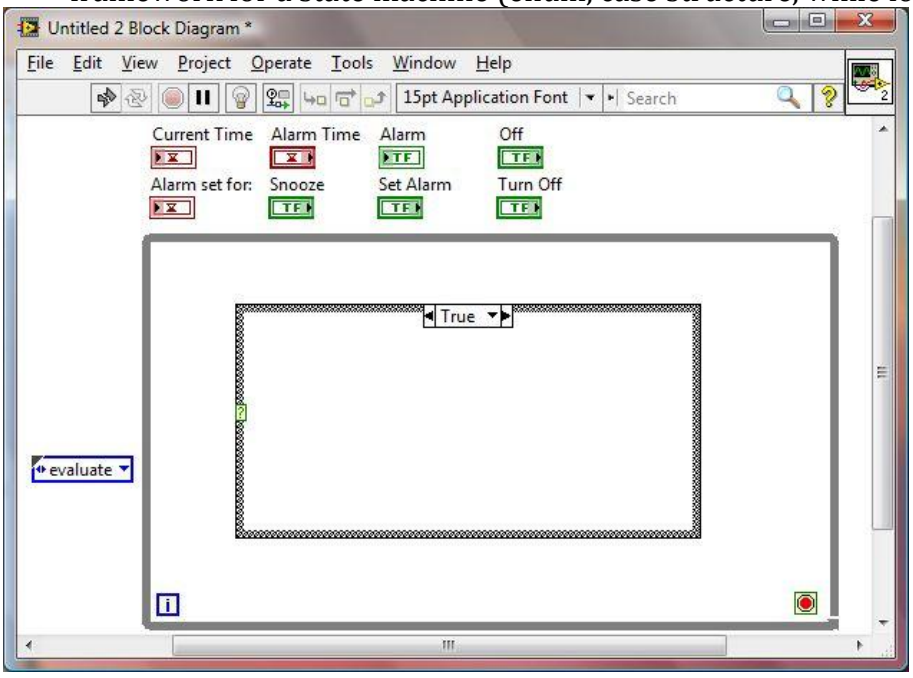
iii. Under Items, create values for: evaluate, set alarm, alarm on, alarm off, and snooze; this represents all the different states that the state machine can have.

iv. Right click the enum and select "Make Type Def"

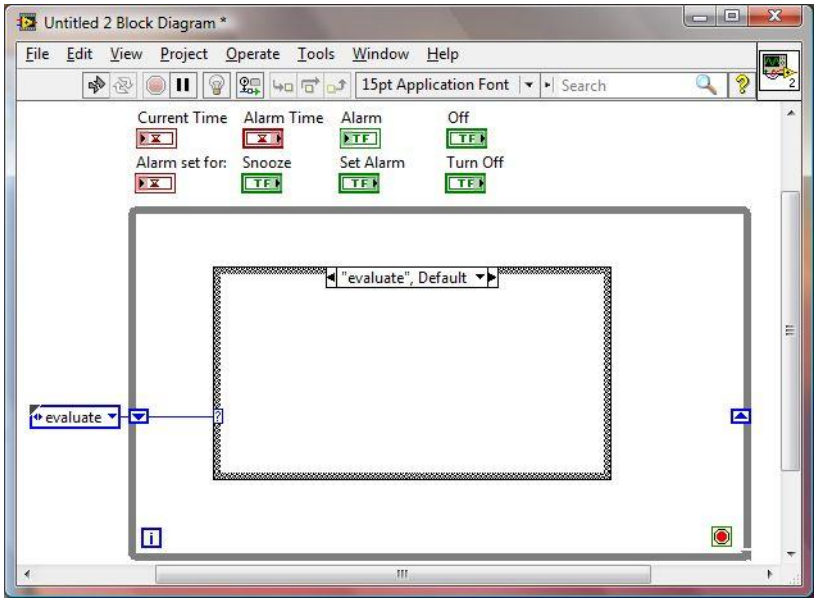
- Note: this creates a .ctl file, and any future changes that you wish to make must be made to the .ctl file. The .ctl file also must be present on the computer for the VI to work



b. Create a While Loop with a Case Structure inside. This is the basic framework for a state machine (enum, case structure, while loop)

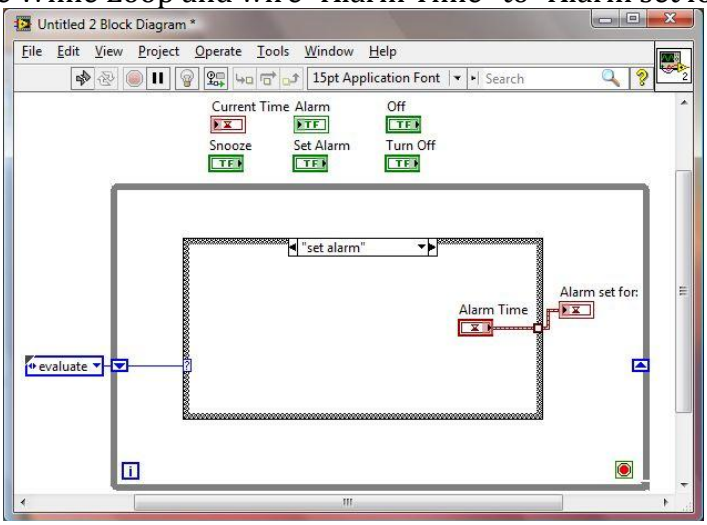


c. Wire the enum to the Case Selector Terminal
    i. Right click the tunnel created on the While Loop and select "Replace with Shift Register". This will allow the while loop to remember what state it was in and what state it should go to.
    ii. Right click the Case Structure and select "Add Case for Every Value"

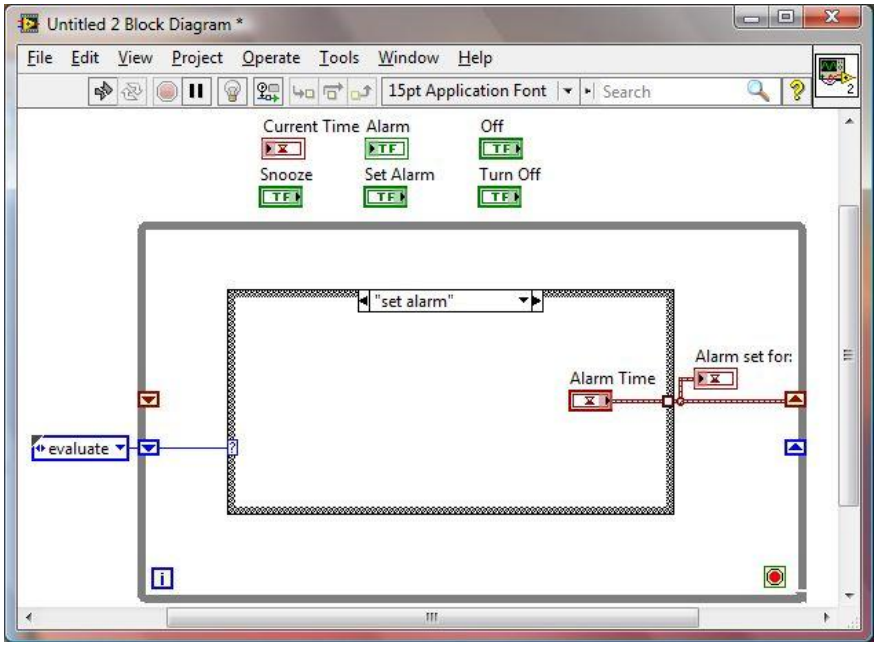


2. Create an uninitialized shift register to contain what time the alarm is set for
    - Note: Because the shift register is not initialized, turning the VI off and on will not cause you to lose the value that the alarm is set for
    - This step is the "functionality code" portion of the state "set alarm"

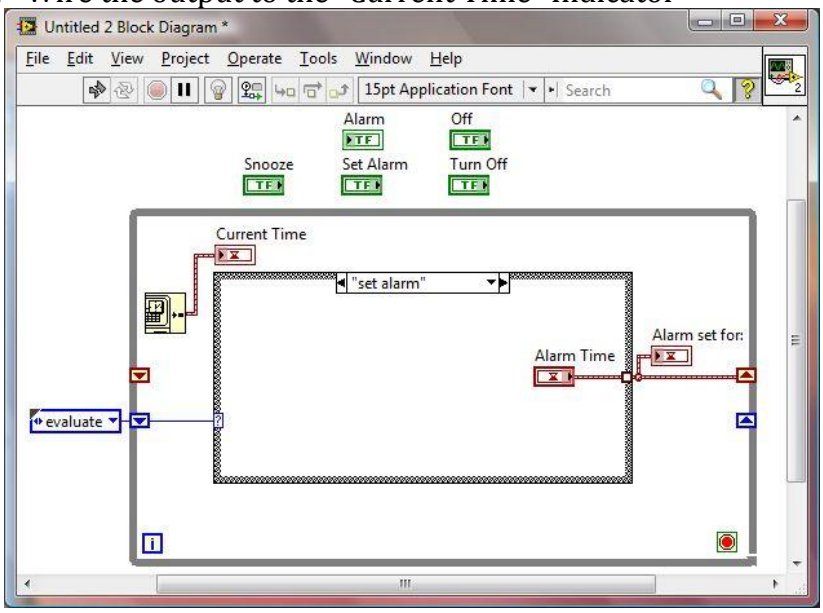
a. Go to the "set alarm" case of the case structure and place the "Alarm Time" control inside the Case Structure
b. Place the "Alarm set for:" indicator outside the Case Structure, inside the While Loop and wire "Alarm Time" to "Alarm set for:"



c. Wire the output tunnel of Alarm Time on the Case Structure to the right side of the While Loop
d. Right Click the tunnel that has just been created and select “Replace with Shift Register”. This shift register will allow for the program to remember the alarm time that was set.

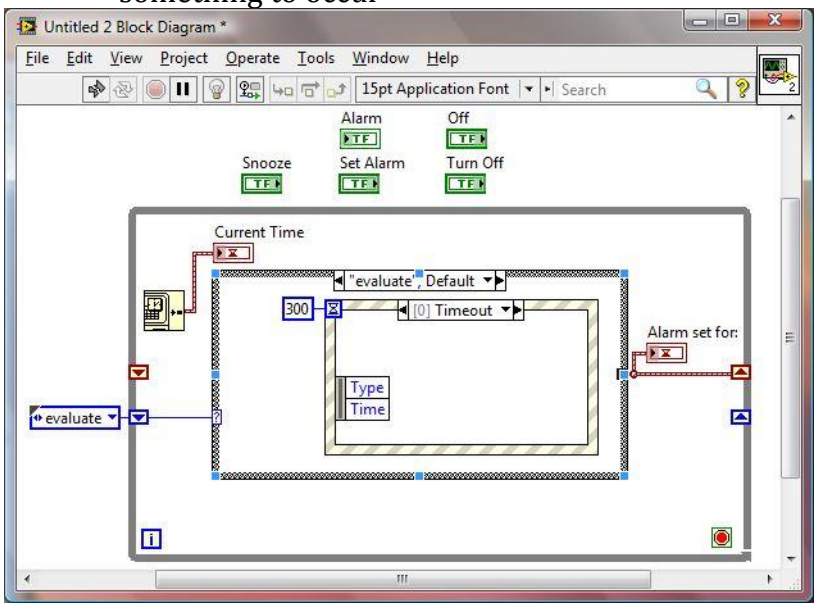


3. Get the current time
   a. Place a Get Date/Time In Seconds VI inside the While Loop, but outside the left side of the Case Structure (Functions> Programming> Timing> Get Date/Time in Seconds)
   b. Wire the output to the “Current Time” Indicator

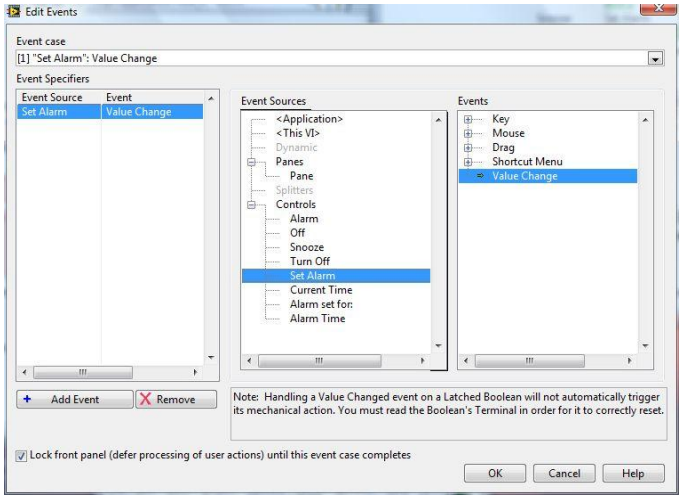


4. Create the code for the “evaluate” case
    - This case contains transition code (code that decides which state to go to) that decides which case is most relevant to execute. As we create the other cases, the transition code will all be “deciding” to go back to the evaluate case.
    - Because an alarm clock responds to situations such as the current time being at or past the alarm time, hitting a snooze button, or other events, an event structure will be employed in the transition code to “sense” what situation is occurring.

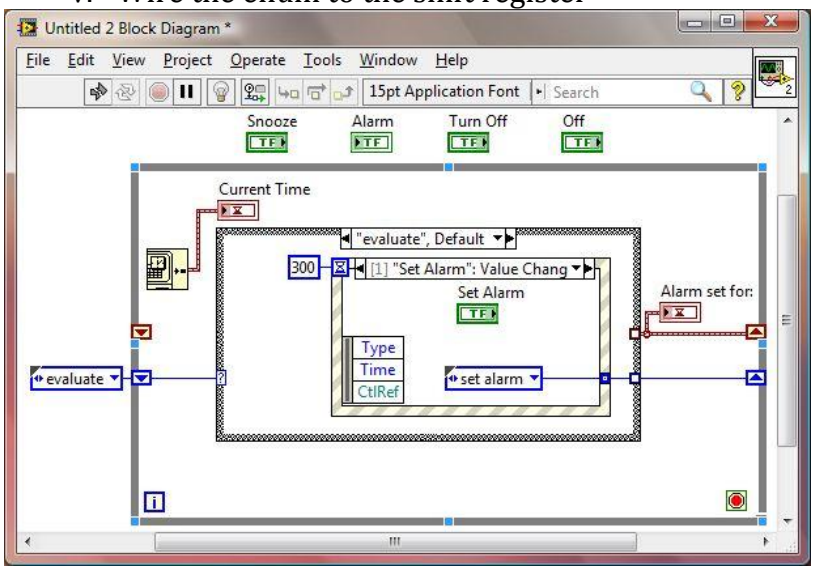
    a. Place an event structure with a timeout of 300
    ** If you are not taking the CLAD add-on part of the workshop, you probably haven’t heard of event structures; don’t worry about it. They are a more advanced concept but if you follow along step by step, you should be able to get the correct solution. **
        i. The event structure is found under Functions> Structures> Event Structure
        ii. Right click the Timeout terminal and select “Create Constant” and then change the value to 300
        ** This will tell the event structure to wait 300ms for something to occur **



    b. Create an event case in the event structure that goes to the “set alarm” state when the “Set Alarm” Boolean is clicked
        i. Right click the event structure and select “Add Event Case…”
        ii. A window will pop up, under the center column (Event Sources) select the “Set Alarm” control and leave the right column (Events) as “Value Change” and hit OK to exit

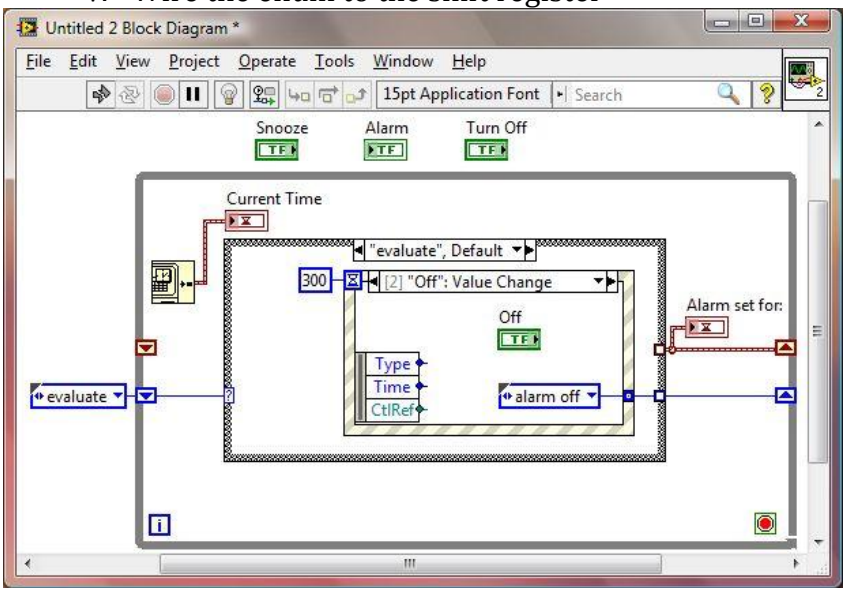


iii. Place the "Set Alarm" control inside this case of the Event Structure
iv. Add a copy (Ctrl-Drag) of the enum to this case and select the "set alarm" value for it (left click and choose from the list)
v. Wire the enum to the shift register

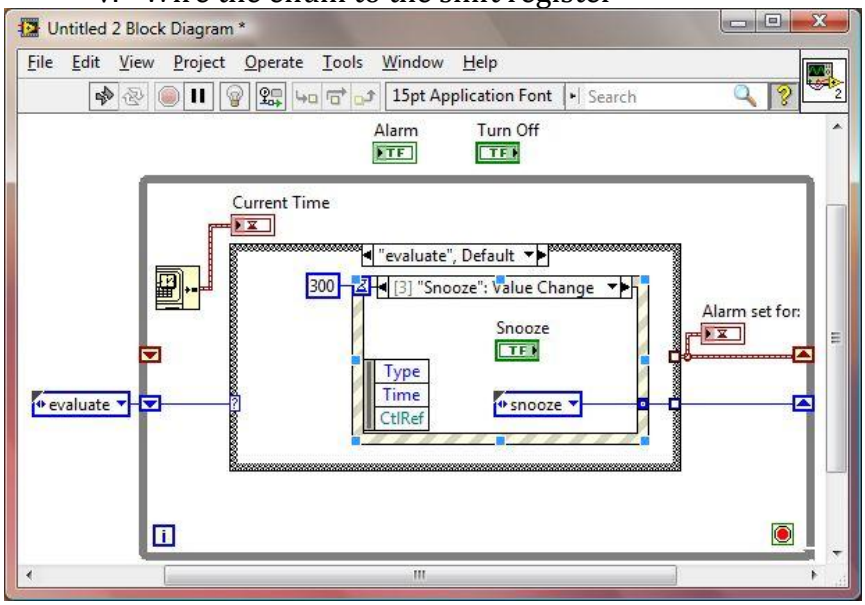


c. Create an event case to go to the "alarm off" state when the "Off" boolean is clicked
   i. Right click the event structure and select "Add Event Case…"
   ii. A window will pop up, under the center column (Event Sources) select the "Off" control and leave the right column (Events) as "Value Change" and hit OK to exit
   iii. Place the "Off" control inside this case of the Event Structure
   iv. Add a copy (Ctrl-Drag) of the enum to this case and select the "alarm off" value for it (left click and choose from the list)

v. Wire the enum to the shift register



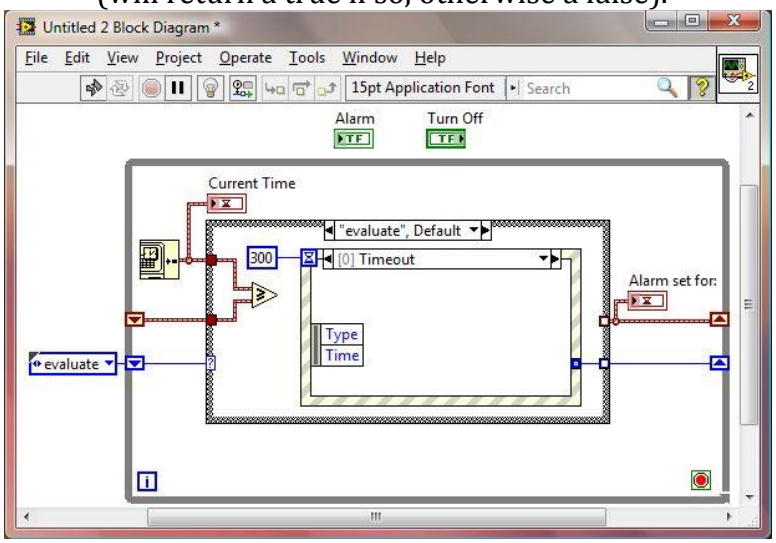
d. Create a case to go to the "snooze" state when the "Snooze" button is clicked
   i. Right click the event structure and select "Add Event Case…"
   ii. A window will pop up, under the center column (Event Sources) select the "Snooze" control and leave the right column (Events) as "Value Change" and hit OK to exit
   iii. Place the "Snooze" control inside this case of the Event Structure
   iv. Add a copy (Ctrl-Drag) of the enum to this case and select the "snooze" value for it (left click and choose from the list)
   v. Wire the enum to the shift register



e. Create the logic for the timeout event case
   - This case decides whether the code should go to “alarm on” or continue to evaluate the code

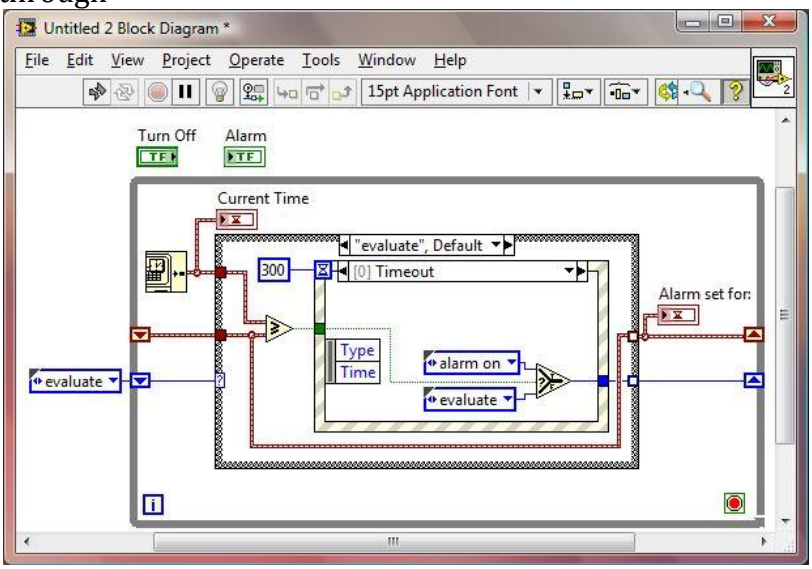
   i. We only want to turn the alarm on when the current time is equal to or later than when the alarm is set for. Because of this add a “Greater Or Equal” Function (Functions> Programming> Comparison> Greater Or Equal) to the inside of the Case Structure to the left of the Event Structure.

   ii. Wire the top input of the Greater Or Equal function to the current time and the bottom input to the shift register containing the time that the alarm is set for
   ** This checks if the alarm time is happening or has passed (will return a true if so, otherwise a false). **

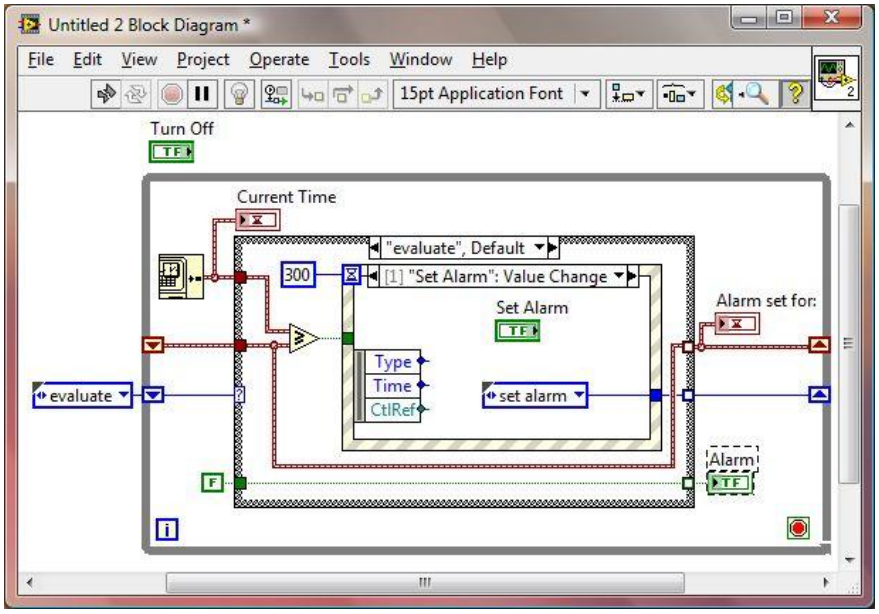


   iii. Place a “Select” function (Functions> Programming> Comparison> Select) inside of the Timeout case with the following inputs and outputs:
      1. Top input: copy of the enum with “alarm on” selected
      2. Middle input: output of Greater Or Equal function
      3. Bottom input: copy of enum with “evaluate” selected
      4. Output: connect to the enum shift register
   - This function passes through the top input if the Boolean is true and the bottom if the Boolean is false.
     ** If the alarm time has passed, the Alarm On state will execute next, otherwise, the Evaluate state will execute **

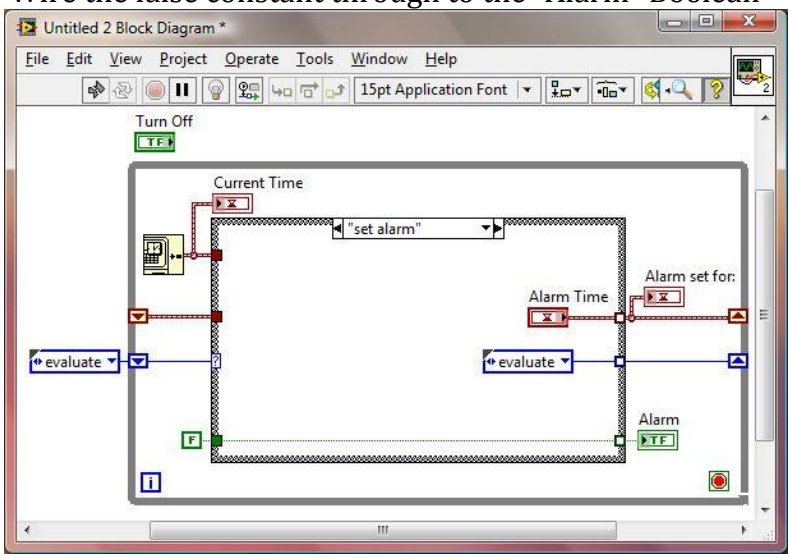
iv. Connect the left side of the alarm shift register through the event structure to the right shift register to pass the alarm time through



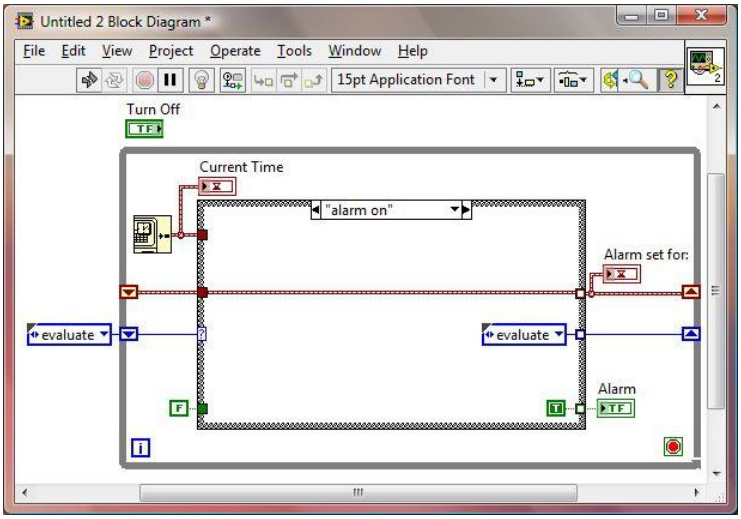
f. Finish the "Evaluate" case
   i. Place the Alarm indicator to the right side of the case structure
   ii. All cases (in the case structure) will output out a false to the Alarm indicator except for when the "Alarm On" state occurs, so in order to minimize the amount of constants needed, place a False constant to the left of the case structure, and only output a true during the Alarm On state.
   iii. Wire the false constant straight through the state machine's case structure and out to the "Alarm" Boolean on the other side



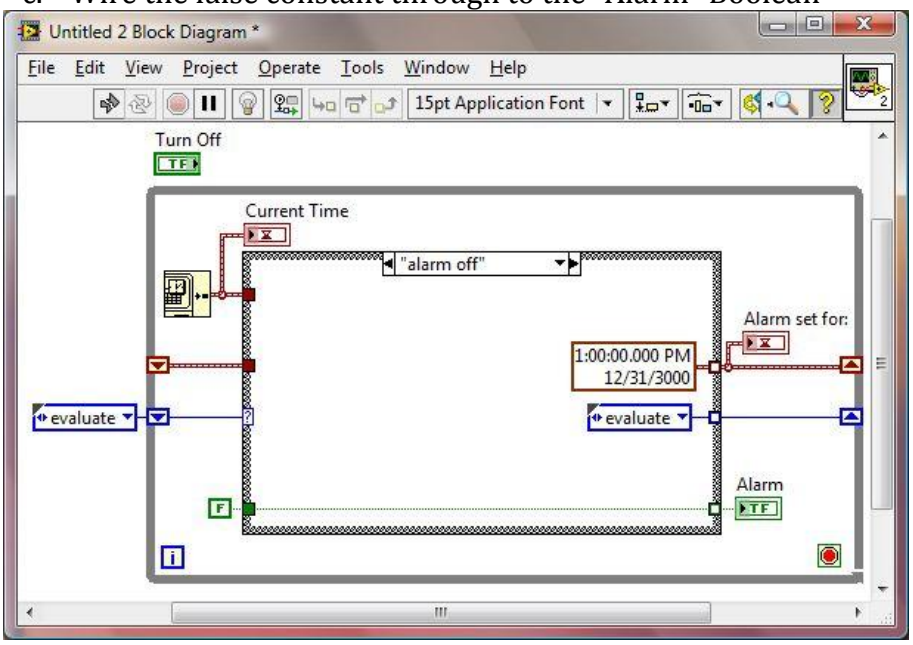
5. Finish the “set alarm” case; this is where the Alarm Time control value gets set into the “Alarm set for:” indicator and shift register
    a. Go to the “set alarm” case and create a copy of the enum set to “evaluate” and wire it to the shift register; once the alarm is set, the program needs to go to the evaluate case to wait for the alarm to go off or for the user to press a button
    b. Wire the false constant through to the “Alarm” Boolean



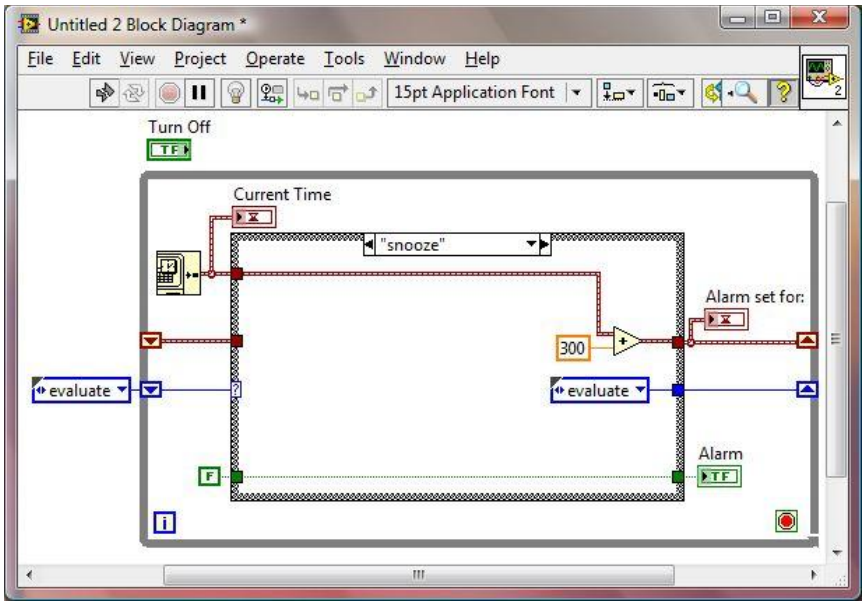
6. Finish the “alarm on” case
    a. Go to the “alarm on” case and wire the alarm through since it is not actually changing
    b. Create a copy of the enum set to “evaluate” and wire it to the shift register
    c. Create a true constant inside the case structure wired to the Boolean tunnel



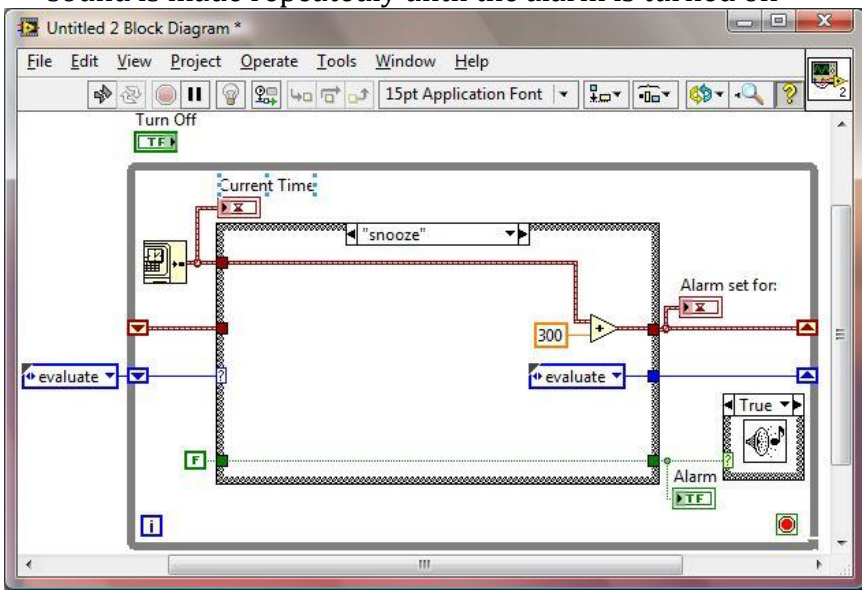
7. Finish the “alarm off” case, which gets executed when the user pressed the “Off” button
    a. Go to the “alarm off” case and create a time stamp constant that will effectively turn off the alarm
        i. Right click the output terminal containing the time stamp, and select Create Constant
        ii. Highlight the portion of the timestamp that is YYYY and type 3000. If this is not long enough for your purposes, you can come back to it at a later time
        ** This constant is in the year 3000, which won’t happen anytime soon, so it acts as if we turned off the alarm **
    b. Create a copy of the enum set to “evaluate” and wire it to the shift register so that the program returns to waiting for user input
    c. Wire the false constant through to the “Alarm” Boolean



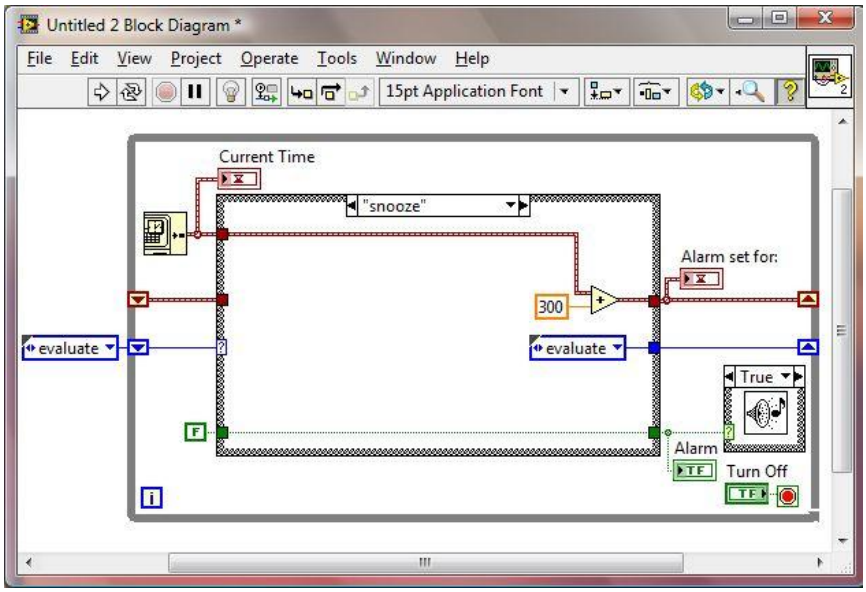
8. Finish the “snooze” case, which will change the alarm time to 5 minutes in the future
    a. Go to the “snooze” case and create a five minute snooze
        i. Five minutes is 300 seconds, so to create a five-minute snooze button, take the current time and wire it into an Add function. For the other input, create a numeric with a value of 300
    b. Create a copy of the enum set to “evaluate” and wire it to the shift register so that the program returns to waiting for the alarm time to occur or for the user to press a button
    c. Wire the false constant through to the “Alarm” Boolean



9. Add some noise to the alarm
    a. Create a case structure to the right side of the current one, and wire the Alarm Boolean wire (also connected to “Alarm” indicator) to the conditional terminal of the case structure
    b. Add a Beep.vi (Functions> Programming> Graphics & Sound> Beep.vi) to the “True” case so that when the Alarm Boolean is true, a beep sound is made repeatedly until the alarm is turned off



10. Wire the “Turn Off” Boolean to the conditional terminal of the While Loop so that the user can control when to stop this Alarm program altogether



Note: the first time you turn it on, the alarm will be on because the unitialized shift register returns a default value when it hasn’t stored anything in the past; click the “Off” button and proceed to setting the alarm.

Optional Extra add-ons:

- Create an exclusion so that if the alarm is set for all zeros, the alarm is off
- Create custom controls so that the front panel looks more like a real alarm clock
- Allow for multiple alarms
- Allow the user to have a more customized snooze button (change time)
- Document your code with tip strips and free labels