Exercise 3-2 Handshaking with Notifiers

Goal

Use notifiers to implement handshaking between parallel processes.

Scenario

In the sprint planning meeting for this iteration, the product owner chose to implement two user stories that apply to the interaction of the boiler controller application with the boiler:

- As a boiler operator, I want to ensure that when I shut down the system, the boiler controller does not suspend execution until the boiler finishes shutting down, so that the boiler controller is always aware of the current state of the boiler.
- As a boiler operator, I want to test the functionality of the application by running it with code that simulates the behavior of a boiler, so that I can verify correct operation before I implement the system.

After discussing each user story with the team, you have identified the following development tasks that must be completed to implement these user stories:

1. Obtain the notifiers that will handle handshaking among the Message Handling Loop of `Main.vi`, `Boiler Controller.vi` and the boiler to ensure that the MHL does not turn off until `Boiler Controller.vi` shuts down, which does not happen until the boiler shuts down.
2. Integrate `Boiler.lvlib` into the application.
3. Synchronize the shutdown operation among all four loops.
4. Release the handshaking notifiers when the application is shutting down.

Design

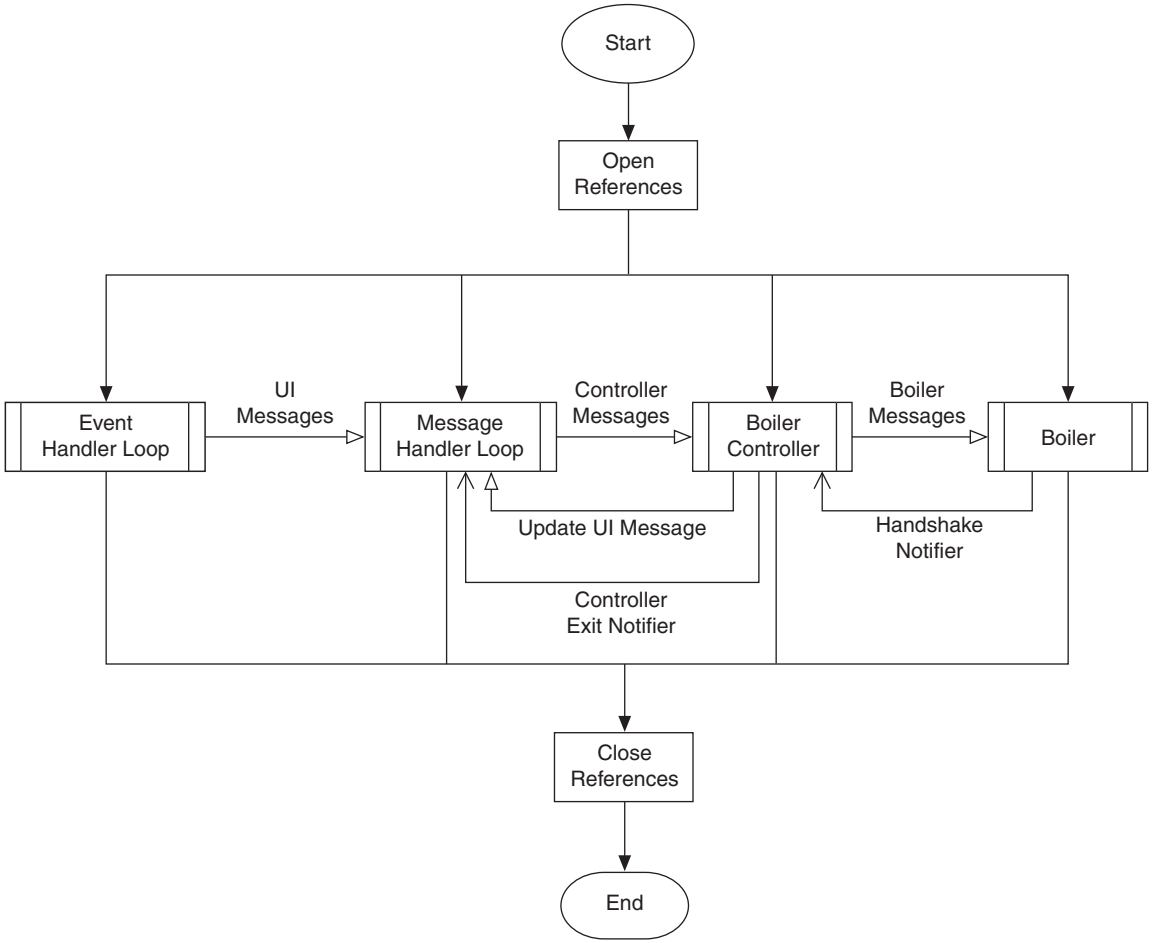
You will create two VIs to handle obtaining and releasing the notifiers that you will use in this application:

- `Create Notifiers.vi`—Obtains two notifiers: one for handshaking between the MHL and `Boiler Controller.vi` and the other for handshaking between `Boiler Controller.vi` and `Boiler.vi`. Bundle these notifiers into a type-defined cluster named `Notifiers.ctl`.
- `Close Notifiers.vi`—Releases the two handshaking notifiers to ensure that the memory for those references is deallocated.

A teammate will develop the code to send a command to the boiler and wait for it to be executed.

In Exercise 2-1, you created a project library that contains code that simulates the behavior of the boiler that the application will control. From this library, you add `Boiler.vi` as a fourth parallel process for the application.

Figure 3-13. Communication Among Event Handler Loop, Message Handler Loop, Boiler Controller, and Boiler



The integration of Boiler.vi requires the following additional changes to Main.vi:

- Add a message queue for the Boiler Loop. Boiler Controller.vi will use this queue to send instructions to Boiler.vi.
- Read configuration data for the boiler from Boiler Init.ini. To do this, you will create a subVI, Read Configuration Data.vi, that reads boiler constants from the INI file.

Due to the amount of start up code that must occur before the parallel loops begin, the team decides to create a subVI, Boiler System Open.vi, that will handle the following functionality:

- Create the Stop user event (Create User Event - Stop.vi)
- Create the message queues (Create All Message Queues.vi)
- Read boiler configuration data (Read Configuration Data.vi)
- Create the notifiers for handshaking (Create Notifiers.vi)

Implementation

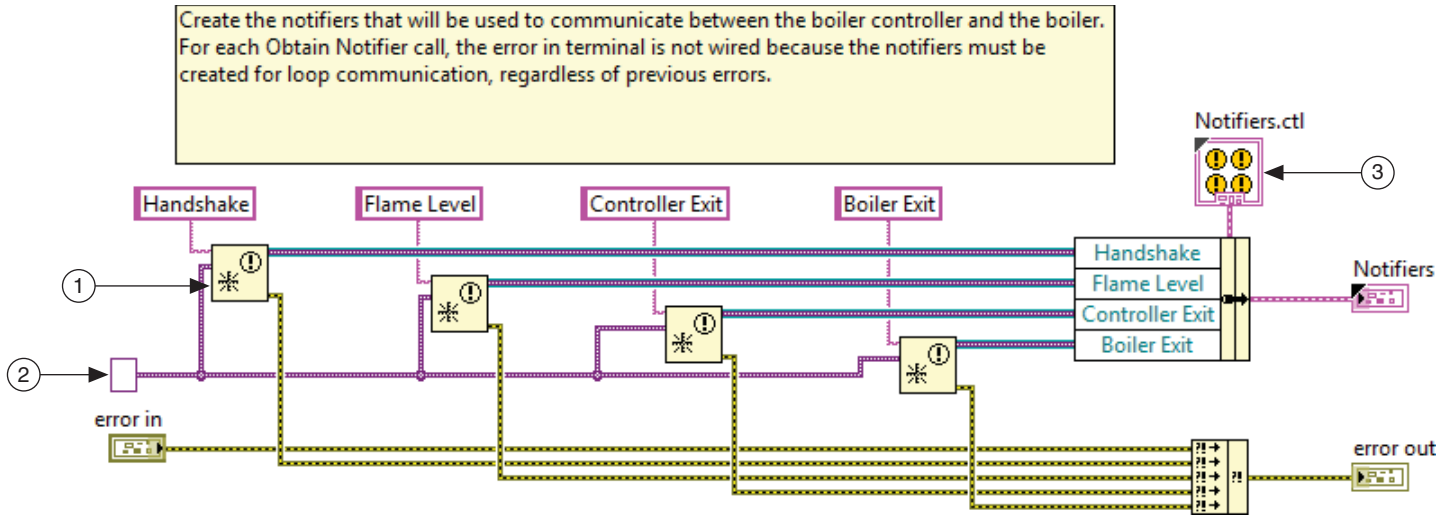
Create Notifiers for Handshaking

Obtain the notifiers that will handle handshaking among the Message Handling Loop of Main.vi, Boiler Controller.vi and the boiler to ensure that the MHL does not shutdown until Boiler Controller.vi shuts down, which does not happen until the boiler shuts down.

1. Create a VI that obtains the notifiers.
   - ☐ In the **Support VIs** directory of the Boiler Controller Project Explorer window, create a virtual folder named Notifiers.
   - ☐ Create a new VI in the **Notifiers** folder and save it as Create Notifiers.vi in <Exercises>\LabVIEW Core 3\Course Project\support\Notifiers.

☐ Create the block diagram as shown in Figure 3-14.

Figure 3-14. Create Notifiers Block Diagram



1 Obtain Notifier function

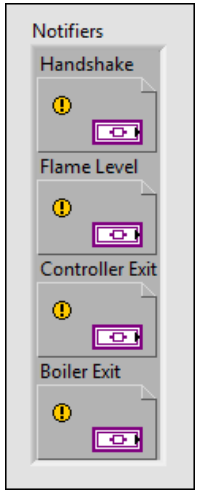
2 Variant constant

3 Notifiers Type Definition

- ☐ Place a **Cluster Constant** on the block diagram.
- ☐ Right-click the **notifier out** output of one of the Obtain Notifier functions and select **Create»Constant**.
- ☐ Create three copies of the constant.
- ☐ Move all four notifier constants into the cluster constant.
- ☐ Right-click the cluster constant border and select **AutoSizing»Arrange Vertically**.
- ☐ Right-click the cluster constant and select Make Type Def.
- ☐ Right-click the type def control and select Open Type Def.

☐ Modify the type def control as shown in Figure 3-15 and save it as `Notifiers.ctl` in the `<Exercises>\LabVIEW Core 3\Course Project\support\Notifiers` directory.

Figure 3-15. Notifiers Type Def Control



☐ Create an icon for the Notifiers type def control as shown in Figure 3-16.

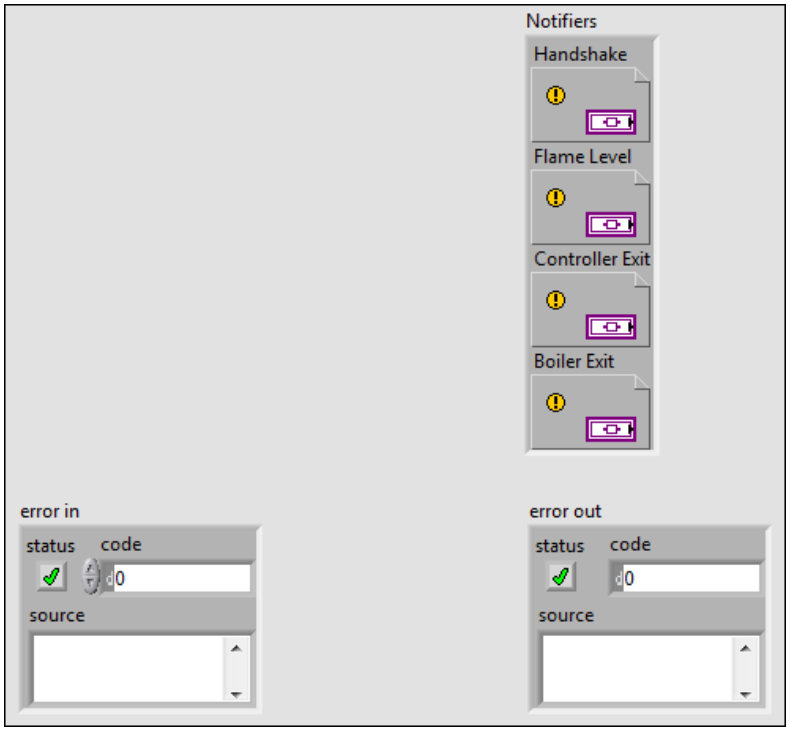
Figure 3-16. Notifiers Type Def Icon

1 Filter glyphs by keyword, `notifier`, to find an appropriate glyph for the icon.

☐ Move `Notifiers.ctl` into the **Notifiers** directory of the **Project Explorer** window.

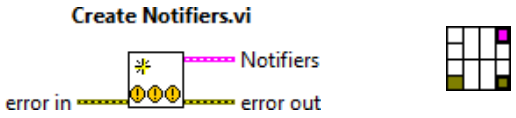
2. Complete the front panel of the Create Notifiers VI as shown in Figure 3-17.

Figure 3-17. Create Notifiers Front Panel



3. Create the an icon and connector pane for `Create Notifiers.vi` as shown in Figure 3-18.

Figure 3-18. Create Notifiers Icon



1 Filter glyphs by keywords, `create` and `notifier`, to find appropriate glyphs for the icon.

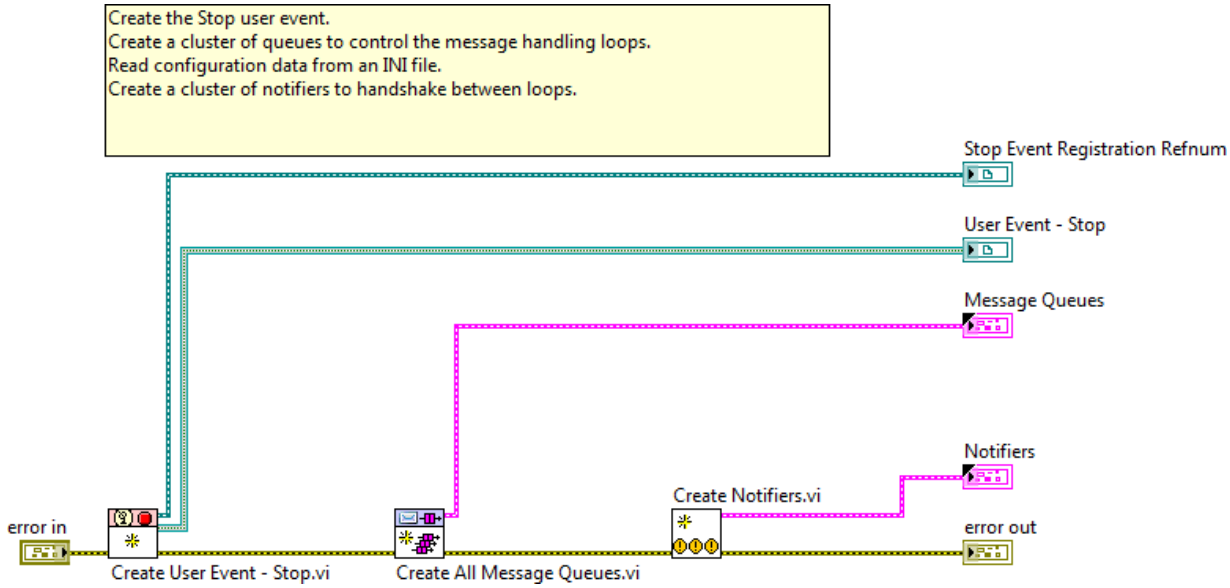
4. Save and close the VI.

5. Create a VI to call the setup VIs.

   ☐ Create a new VI in the **Support VIs** folder in the Boiler Controller **Project Explorer** window. Save the VI as `Boiler System Open.vi` in the `<Exercises>\LabVIEW Core 3\Course Project\support` directory.

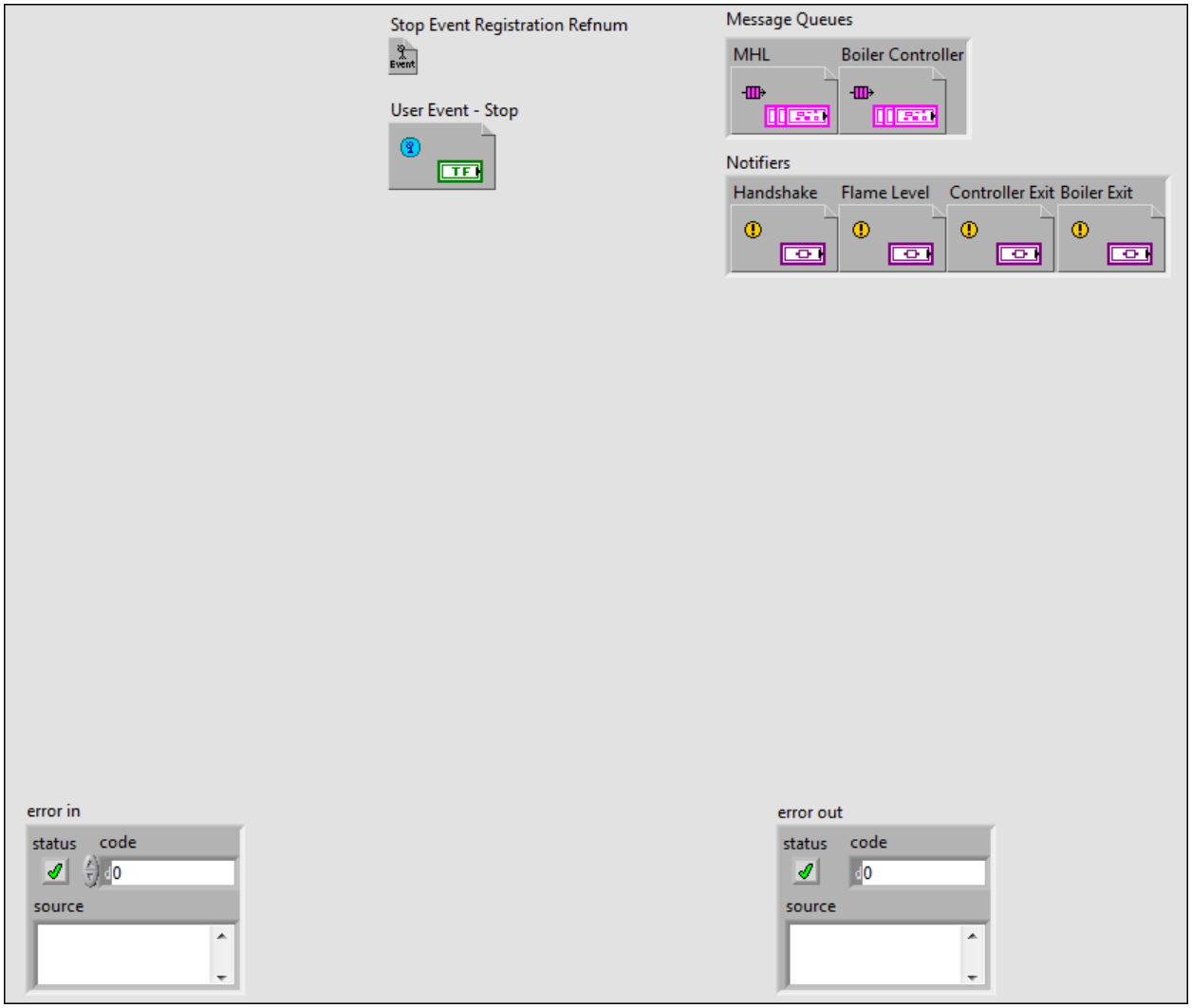
   ☐ Create the block diagram as shown in Figure 3-19.

Figure 3-19. Boiler System Open Block Diagram



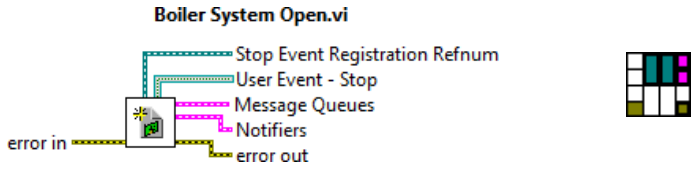
Arrange the Boiler System Open front panel as shown in Figure 3-20.

Figure 3-20. Boiler System Open Front Panel



□ Create an icon and connector pane for the Boiler System Open VI as shown in Figure 3-21.

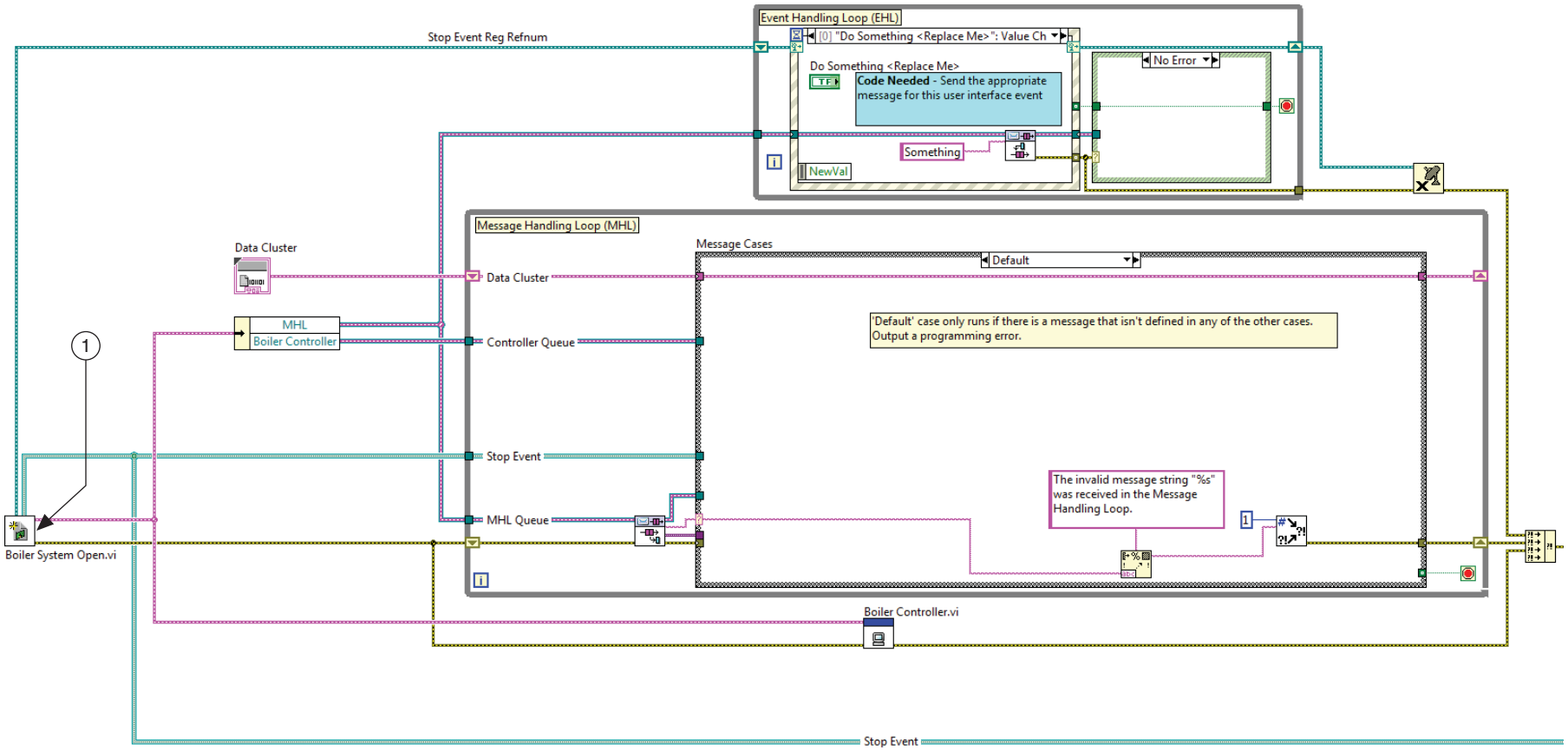
Figure 3-21. Boiler System Open Icon and Connector Pane



1 Filter glyphs by keyword, `initialize`, to find an appropriate glyph for the icon.

6. Call the Boiler System Open VI from the Main VI instead of calling each function separately.

Figure 3-22. Main VI Calling Boiler System Open



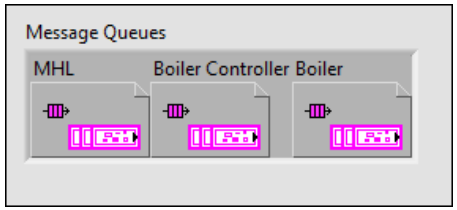
1 **Boiler System Open VI**—Replace `Create User Event - Stop.vi` and `Create All Message Queues.vi` with this VI.

Integrate Boiler.lvlib

This library contains `Boiler.vi`, the top-level VI responsible for simulating the behavior of the boiler, as well as the subVIs and type definitions required for `Boiler.vi` to run. `Boiler.vi` will serve as a fourth parallel process in the application (Event Handling Loop, Message Handling Loop, `Boiler Controller.vi`, and `Boiler.vi`).

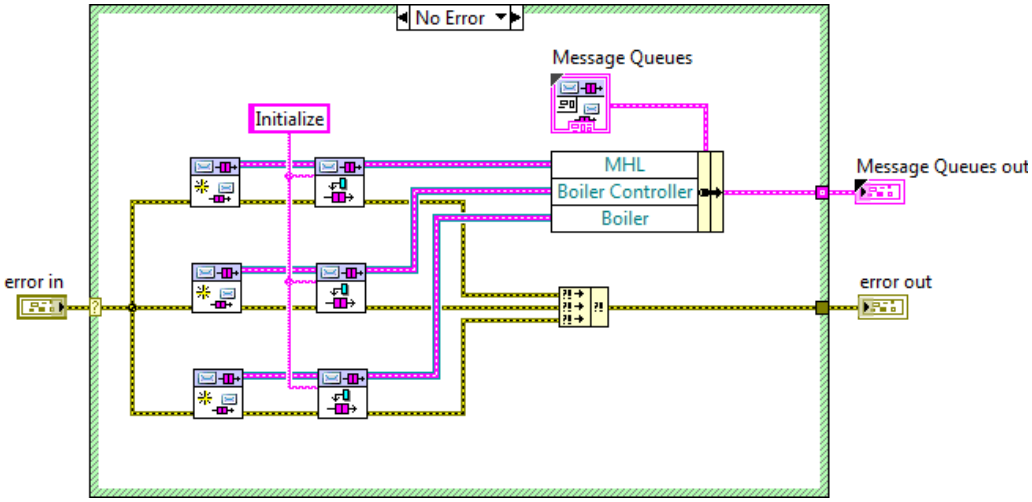
1. Modify `All Message Queues.ctl` to include a Boiler queue as shown in Figure 3-23.

Figure 3-23. All Message Queues Control with Boiler Queue



2. Modify `Create All Message Queues.vi` as shown in Figure 3-24.

Figure 3-24. Create All Message Queues Block Diagram

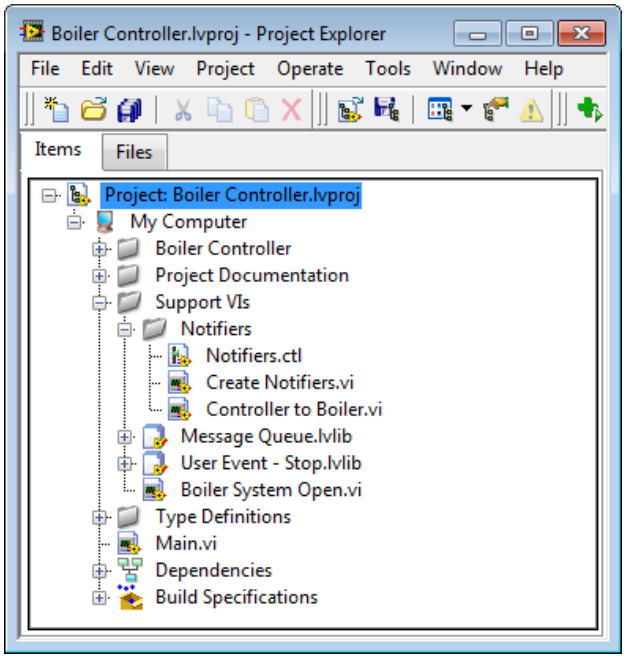


3. Add `Controller to Boiler.vi` to the project.

   ☐ Move a copy of `Controller to Boiler.vi` from `<Exercises>\LabVIEW Core 3\External\Support VIs\Notifiers` to `<Exercises>\LabVIEW Core 3\Course Project\support\Notifiers`.

   ☐ Add `Controller to Boiler.vi` to the Boiler Controller project in the **Support VIs»Notifiers** folder as shown in Figure 3-25.

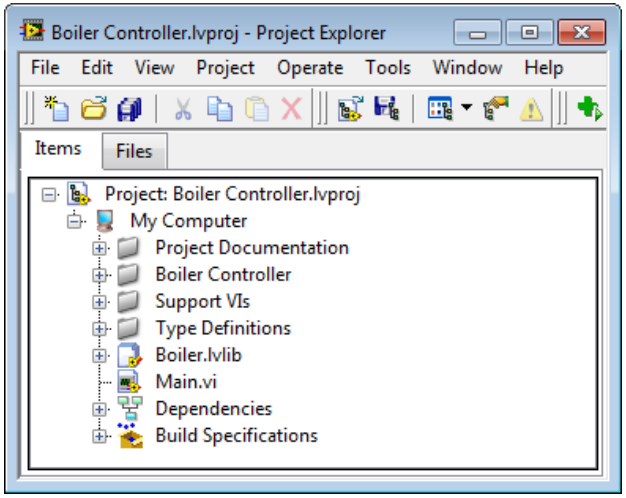
Figure 3-25. Boiler Controller Project with Controller to Boiler Notifier



4. Add `Boiler.lvlib` to the project.

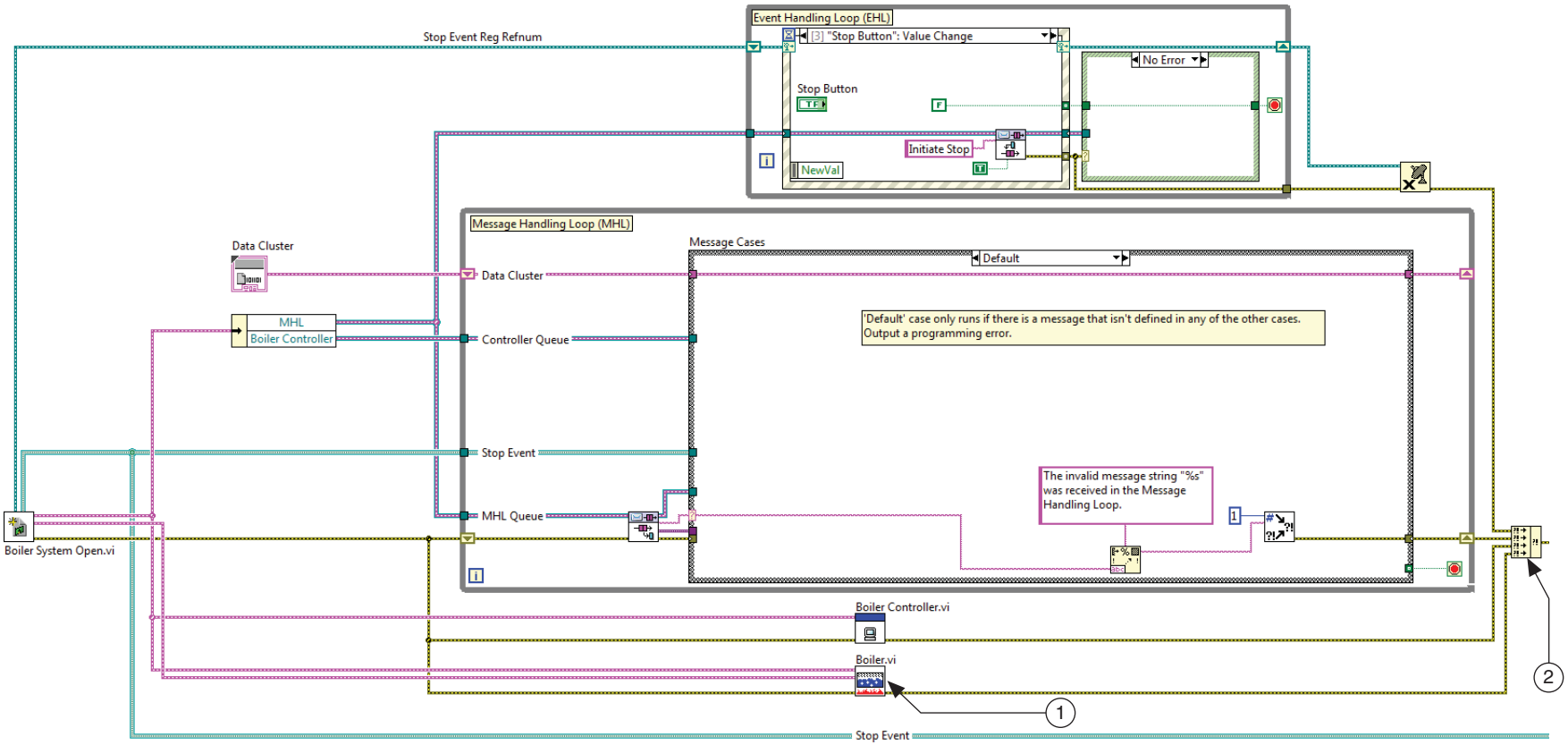
   ☐ Move the contents of the `<Exercises>\LabVIEW Core 3\External\Boiler` directory to the `<Exercises>\LabVIEW Core 3\Course Project\Boiler` directory.

Figure 3-26. Boiler Controller Project with Boiler.lvlib



5. Place an instance of `Boiler.vi` from the `Boiler.lvlib` on the block diagram of `Main.vi` and modify `Main.vi` as shown in Figure 3-27.

Figure 3-27. Main VI with Boiler VI



1 Wire the Message Queues, Notifiers, and Error cluster wires as shown.

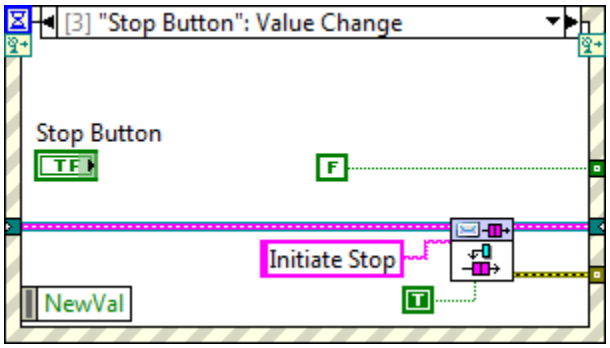
2 Expand the Merge Errors node to accommodate another input.

Synchronize the Shutdown Operation

This operation is initiated through a button-click and results in stopping all four loops.

1. Modify the Stop Button: Value Change event to enqueue the Initiate Stop message as shown in Figure 3-28.

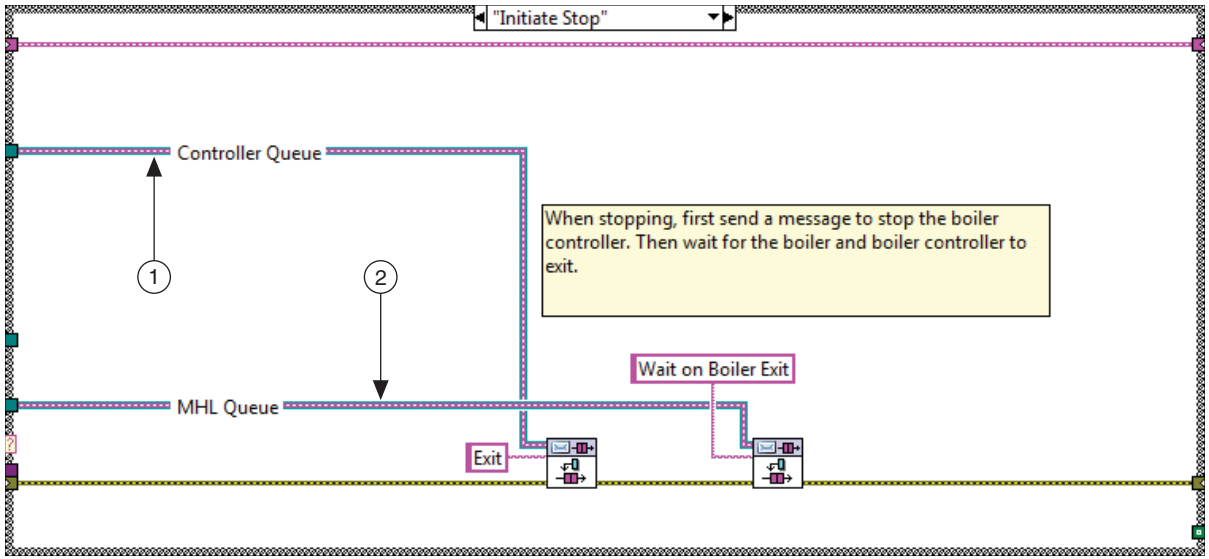
Figure 3-28. Stop Button: Value Change Event



The EHL sends a message to the MHL.

2. Create the Initiate Stop case of the Message Handling Loop as shown in Figure 3-29.

Figure 3-29. Message Handling Loop Initiate Stop Case

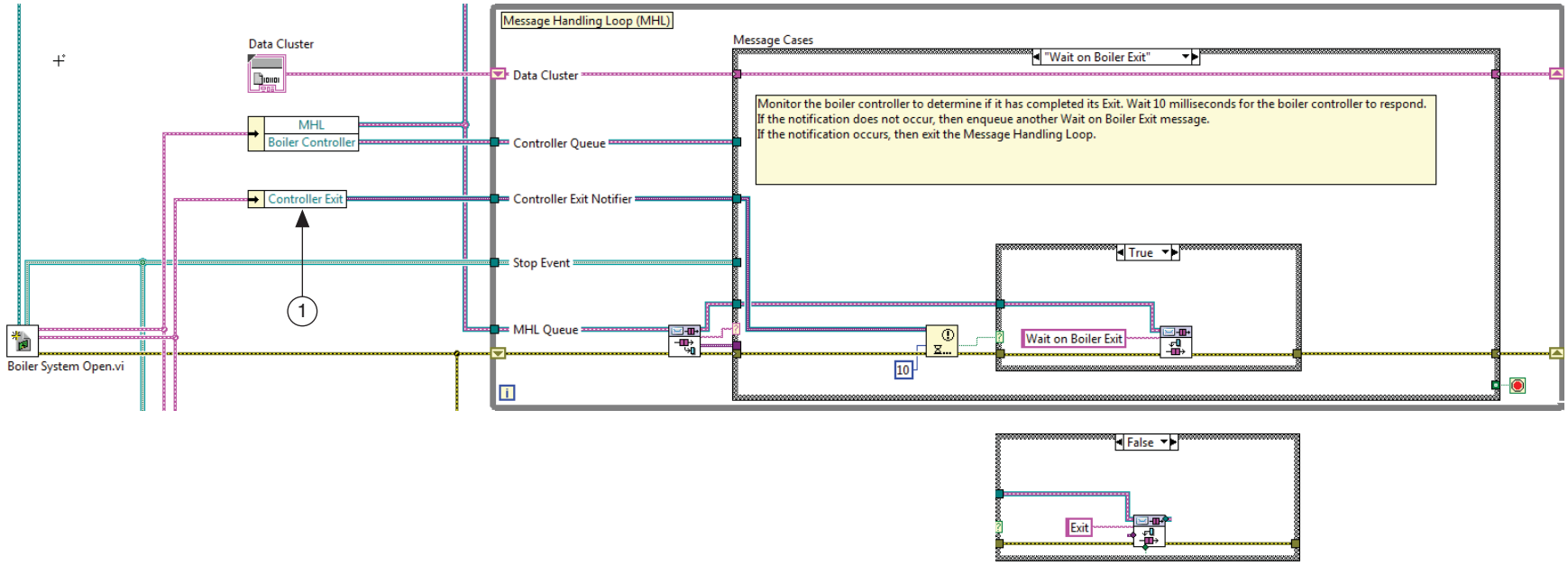


1 The Message Handling Loop sends a message to the Boiler Controller VI.

2 The Message Handling Loop waits for Boiler Controller VI to finish executing the message before proceeding.

3. Create the Wait on Boiler Exit case of the Message Handling Loop as shown in Figure 3-30.

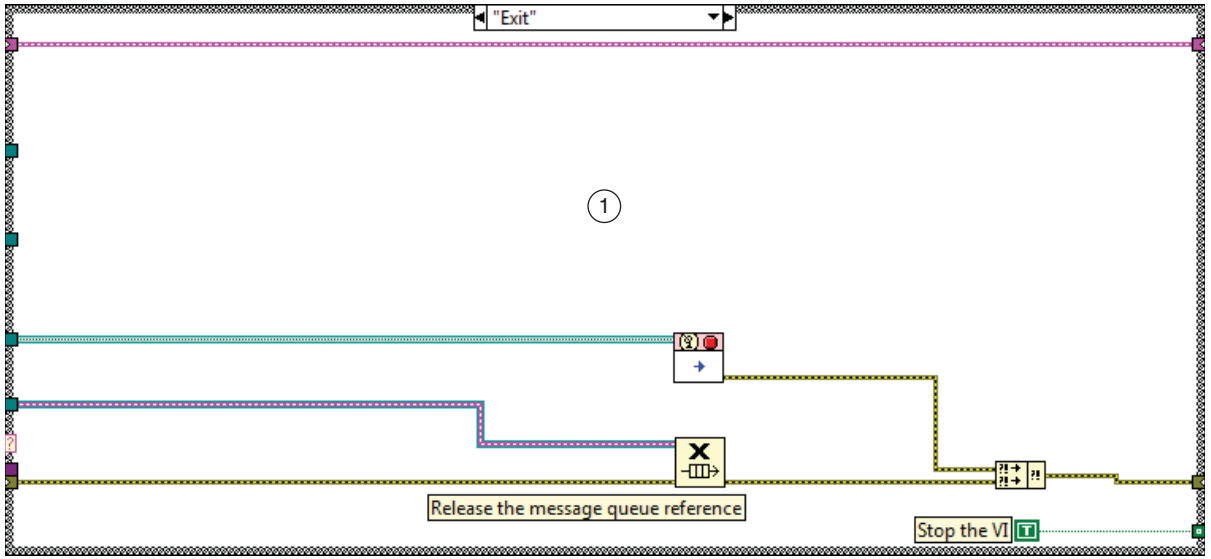
Figure 3-30. Message Handling Loop Wait on Boiler Exit Case True



1 Unbundle the Controller Exit notifier and complete the wiring as shown.

4. Modify the Exit case of the Message Handling Loop as shown in Figure 3-31.

Figure 3-31. Exit Case

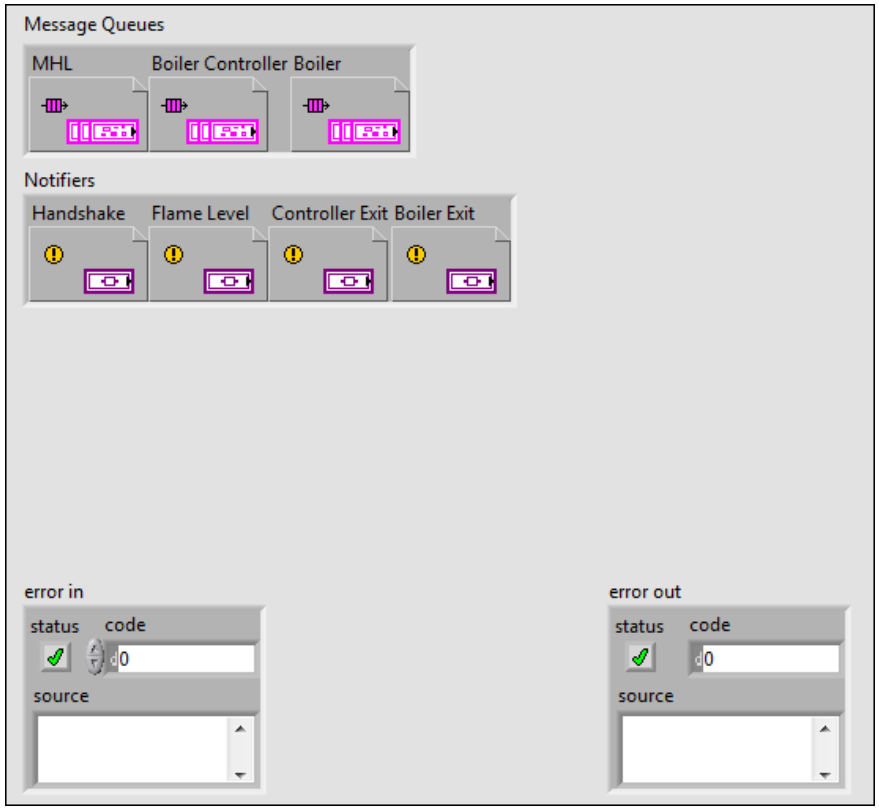


1 Delete the `Enqueue Message.vi` from this case because the Exit message is now being handled in the Wait on Boiler Exit case.

5. Update the Exit case of `Boiler Controller.vi`

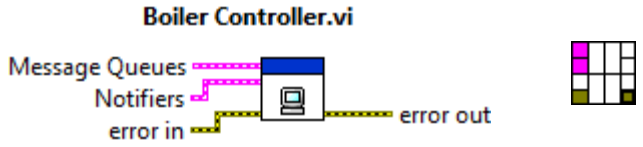
   ☐ Place a copy of the `Notifiers.ctl` type def control on the front panel of `Boiler Controller.vi` as shown in Figure 3-32

Figure 3-32. Boiler Controller Front Panel with Notifiers



6. Modify the Boiler Controller connector pane to include an input for notifiers as shown in Figure 3-33.

Figure 3-33. Boiler Controller Icon and Connector Pane with Notifiers Input

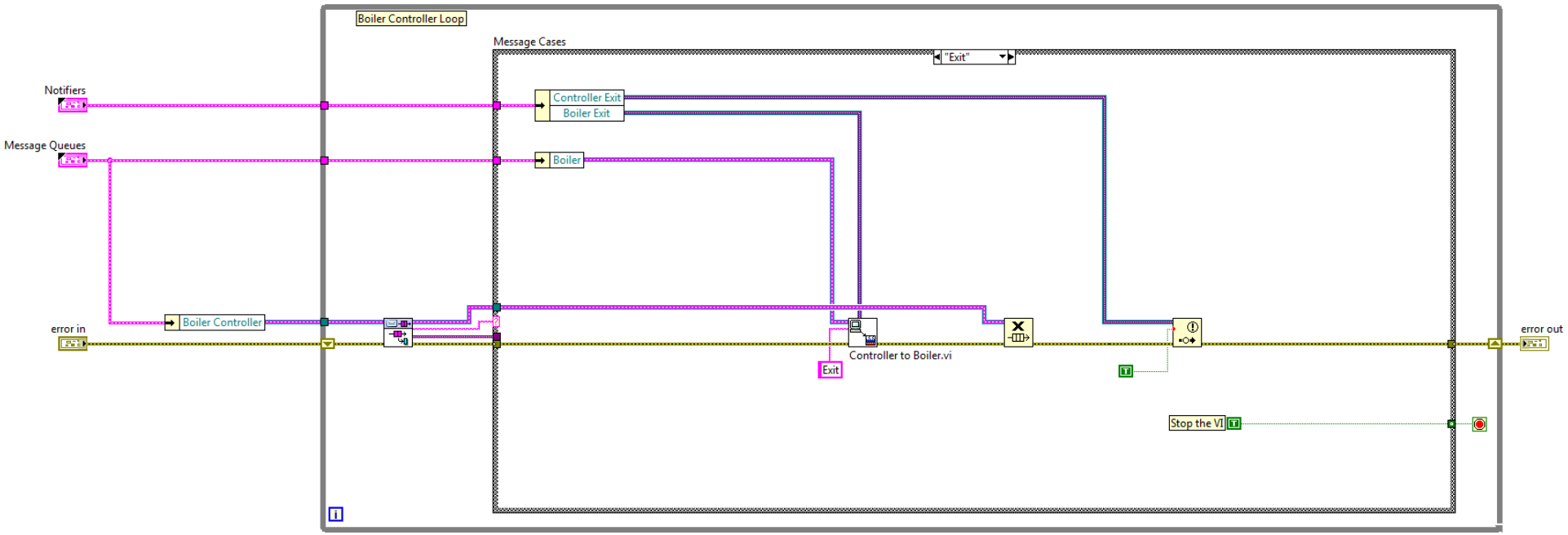


☐ On the block diagram of `Main.vi`, wire the **Notifiers** output of `Boiler System Open.vi` to the **Notifiers** input of `Boiler Controller.vi`.

7. Complete the `Boiler Controller.vi` Exit case as shown in Figure 3-34.

☐ `Boiler Controller.vi` sends a message to `Boiler.vi` and waits for `Boiler.vi` to finish executing that message before proceeding.

Figure 3-34. Boiler Controller VI Exit Case



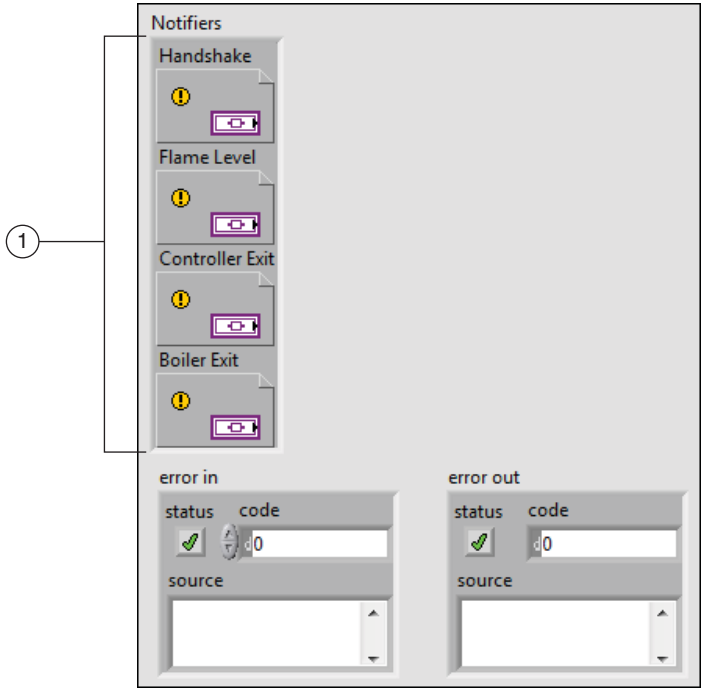
Release the Handshaking Notifiers when the Application Is Shutting Down

1. Close the notifiers that were allocated in `Create Notifiers.vi`.

   ☐ Create a new VI and save it as `Close Notifiers.vi` in the `<Exercises>\LabVIEW Core 3\Course Project\Support\Notifiers` directory and add it to the **Support VIs»Notifiers** directory of the Boiler Controller project.

   ☐ Create the `Close Notifiers.vi` front panel as shown in Figure 3-35.

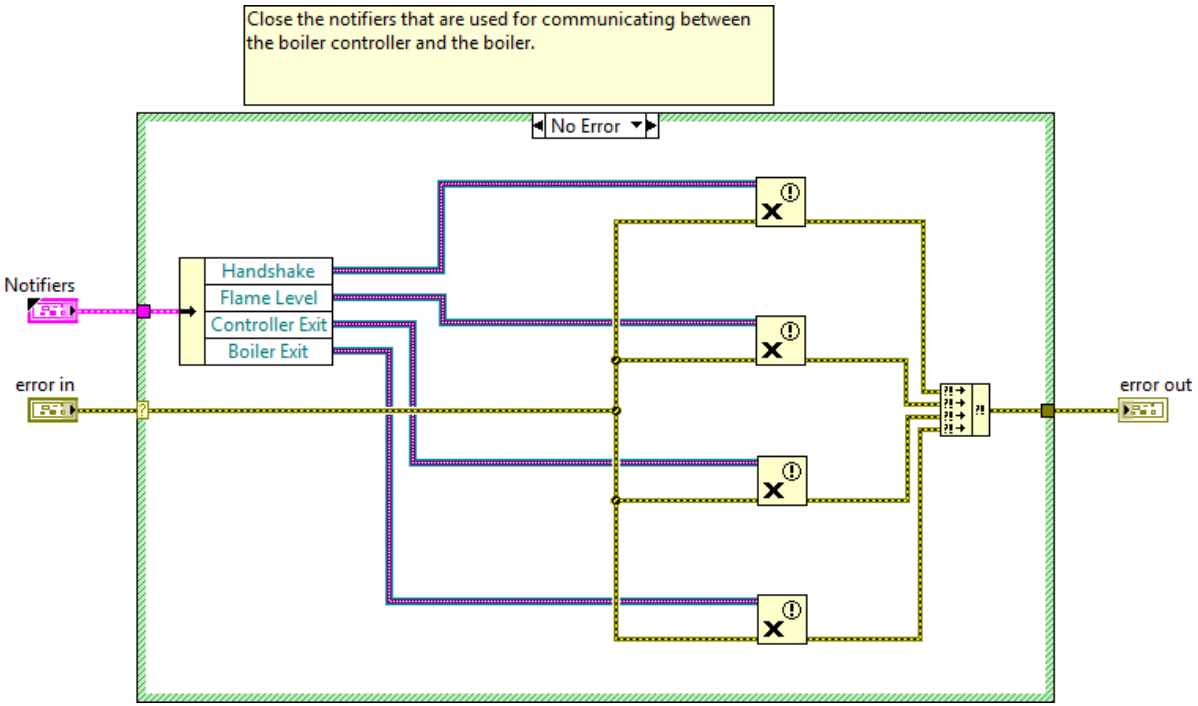
Figure 3-35. Close Notifiers VI Front Panel



1 Notifiers.ctl type def

☐ Modify the Close Notifiers VI block diagram as shown in Figure 3-36.

Figure 3-36. Close Notifiers VI Block Diagram



☐ Wire the error cluster wire directly through the Error case.

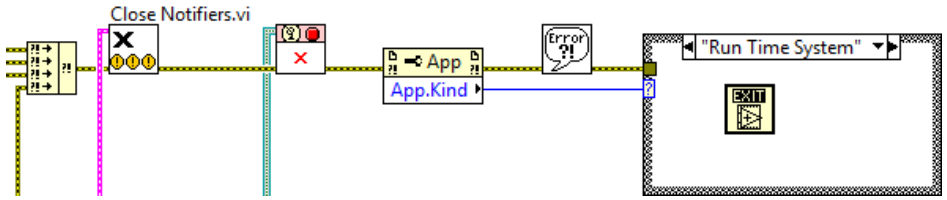
☐ Create the icon and connector pane as shown in Figure 3-37.

Figure 3-37. Close Notifiers Icon and Connector Pane

1 Filter glyphs by keyword, `notifier`, to find an appropriate glyph for the icon. Use the drawing tools to draw an X on the icon.

2. Modify `Main.vi` to wire Notifiers to `Boiler Controller.vi` and call `Close Notifiers.vi` at the end of execution as shown in Figure 3-38.

Figure 3-38. Main VI with Close Notifiers



Test the Application

Verify that the VI and the application build specification both work as expected before moving on to the next sprint.

1. Test the VI.
   - ☐ Run the VI and verify that the **Boiler Controller - Initialize** dialog box launches.
   - ☐ Click the **Do Something** and **Do Something Else** buttons and verify that the **Display** status string updates appropriately.
   - ☐ Click **Stop** and verify that all loops exit successfully.
2. Test the executable.
   a. Right-click the Main Application build specification and select **Build** from the shortcut menu to build the executable with the changes from this lesson.
   b. Click the **Explore** button when LabVIEW finishes the build.
   c. Double-click `Boiler Controller.exe` to run the application.
   - ☐ Verify that the **Boiler Controller - Initialize** dialog box launches and the **Do Something** and **Do Something Else** buttons update the **Display** status string appropriately.
   - ☐ Click **Stop** and verify all windows close.

End of Exercise 3-2