1 Lab Objective
In this lab you will continue working with programmable logic controllers and the use of relay ladder logic.

2 Ladder Logic Elements

2.1. Coils
A relay ladder logic coil is used to provide an output that corresponds directly to the state of the rung on which it is placed. It can be thought of as an internal variable whose value is constantly updated depending on the value of its input logic.

In most ladder logic software, latching coils may also be available. This type of coil will turn on when the state of the rung on which it’s placed is true, but will stay on even if that rung later becomes false. With this coil type, an additional rung is required to define logic to reset each coil. We will not use these latching type coils in this lab.

2.2. Finish Flags
A finish-flag is merely an internal coil that is designed carefully to indicate the end of a sequence of events. At the end of the sequence, the internal coil should trigger on and stay on. It could be used subsequently to start or inhibit another event or sequence of events. The design of a finish flag using RLL relies on a latching mechanism and past inputs and event (similar to the operation of coils with latching capabilities).

3 Ladder Logic Examples
3.1. Alternating Red Lights
A controller is needed to simulate an intersection where both sides are flashing red lights (the equivalent of a four-way stop). The timing diagram is as shown below.

1a. Timing Diagram

Figure 1: Alternating Red Lights - Timing Diagram
For this scenario, the red light on street 1 is one output and the red light on street 2 is the second output. According to the timing diagram, the red lights should alternate on and off at two-second intervals.

1b. Sequence of Events
1. Turn Red 1 on and Red 2 off for 2 seconds
2. Turn Red 1 off and Red 2 on for 2 seconds
3. Repeat steps 1 through 2 indefinitely.

3.2. Washing Machine

![Washing Machine - Schematic](image)

The washing time of the washer comprises three cycles: wash, rinse, and spin cycles. During the wash cycle, water is added until a high level switch is triggered on. Then, the water valve is shut off and the agitator is activated for 12 minutes. After the 12 minutes timer has expired, the dirty water is drained out through a pump that is activated as soon as the agitator stops. The drain pump is stopped when the low-level switch is triggered low. Then, the rinse cycle starts.

The rinse cycle starts when the low-level switch is triggered low (end of wash cycle). The cycle starts by adding water into the tub until the high level switch is triggered. As soon as the high level switch is triggered, the water valve is shut off and the agitator is activated for 6 minutes. When the 6-minute clock has expired, the agitator is stopped and the water is drained out. The pump continues to drain the water until the low-level switch is triggered low, then the spin cycle starts.

During the spin cycle, the motor connected to the agitator disengages from the gearbox permitting the agitator to spin at a higher speed (lower torque) for 4 minutes. To simplify the problem use two separate motors to simulate the spin and agitate cycles as indicated below.
Here, the cycle time for each event sequence has been decreased for simplicity.

2a. **Sequence of Events**

1. Press the start button
2. Add water until the high level switch is triggered.
3. Shut off the water valve.
4. Turn the agitator on for 12 seconds.
5. Stop the agitator and start the pump until the low level switch is triggered off.
6. Rinse: Repeat steps 2 through 5 (the rinse time for step 4 is 6 seconds).
7. Spin: Turn the spin motor on for 4 seconds.

4 **Prelab**

4.1. **Alternating Red Lights**

Using the outputs given in Table 1, create a ladder logic diagram for the alternating red lights described in Figure 1.

<table>
<thead>
<tr>
<th>Output</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red 1</td>
<td>Y001</td>
</tr>
<tr>
<td>Red 2</td>
<td>Y004</td>
</tr>
</tbody>
</table>

Table 1: Alternating Lights - Input/Output

Use the appropriate ladder logic representation and naming convention (found in the Appendix) for each element in the diagram.

Unlike the previous lab, here we will place the artificial constraint that only 1 timer is available on the PLC. Accordingly, the single timer’s output can only be used as an indicator when your outputs should change but cannot be used to change your outputs directly. (Hint: Use a coil)

4.2. **Washing Machine**

Using the outputs given in Table 2, create a ladder logic diagram for the washing machine described in the second ladder logic example.

<table>
<thead>
<tr>
<th>Output</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill</td>
<td>Y101</td>
</tr>
<tr>
<td>Agitate</td>
<td>Y102</td>
</tr>
<tr>
<td>Drain</td>
<td>Y103</td>
</tr>
<tr>
<td>Spin</td>
<td>Y104</td>
</tr>
<tr>
<td>Complete</td>
<td>Y105</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power</td>
<td>X001</td>
</tr>
<tr>
<td>Low Level</td>
<td>X002</td>
</tr>
<tr>
<td>High Level</td>
<td>X003</td>
</tr>
</tbody>
</table>

Table 2: Washing Machine - Input/Output
Notice that this process, unlike the previous lab, is both time and event driven. Also, notice that some particular sequence of events is repeated during this process. One very effective way to design this feature is to use finish-flags. Design finish-flags to indicate the end of the first two cycles (wash and rinse). Use these finish-flags (and others if needed) to control the entire process. Do not use the counters in this lab.

The start/stop switch is used to simulate power on/off in some sense. When power is turned on, all timers should be enabled, and the wash cycle started. When power is turned off, all timers should be reset, and all outputs shut off. When power is turned back on, the process should start over beginning, optionally, with the wash cycle, or at a later stage.

Once again, use the appropriate ladder logic representation and naming convention (found in the Appendix) for each element in the diagram.

5  Lab Procedure

5.1.  Alternating Red Lights
Implement the new alternating red lights program from Prelab Exercise 1. Show your working PLC program to the TA. Save or print a copy of the ladder logic diagram for the postlab.

5.2.  Washing Machine
Implement the washing machine from Prelab Exercise 2. Show your working PLC program to the TA. Save or print a copy of the ladder logic diagram for the postlab.

6  Postlab

6.1.  Ladder Logic Diagrams
Print out the Ladder Logic Diagrams for both exercises.

6.2.  Comment Diagrams
On each diagram write comments for every rung explaining what is happening (logic, counter, finish flag etc.)
Appendix

1. **Start New Project**
   1. Open CLICK Programming Software
      - Start → All Programs → Local → AutomationDirect → Click_Programming_Software
   2. Select “Start a new project”

2. **Write to PLC**
   1. Write Project
      - PLC → Write Project into PLC
      - Click “OK” to begin read/write program
      - If PLC is in RUN Mode, click “YES” to change to STOP Mode
      - Click “OK” to acknowledge transfer complete
      - Click “OK” to set PLC mode to RUN
   2. Connect to PLC (Only if Write Project fails)
      - PLC → Connect
      - Verify settings
      - PC COM Port No.: COM1
      - Baud Rate: 38400
      - Address: 1
      - Parity Bit: Odd
      - Stop Bit: 1
      - Click “OK” to connect to PLC

3. **Address Naming Convention**
   Each element has an independent name. The following address naming convention is used by the CLICK Software.
   - X001-X003: Input
   - Y001-Y008: Output
   - C1-C2000: Coil
   - T1-T500: Timer
   - CT1-CT250: Counter
4. **Ladder Logic Elements**

The following standard ladder logic elements are available in the CLICK Software.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Ladder Logic Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normally Open Contact</td>
<td>Is ON when the defined address (e.g. X001, Y001, C1, or T1) is ON</td>
<td><img src="image" alt="Normally Open Contact" /></td>
</tr>
<tr>
<td>Normally Closed Contact</td>
<td>Is ON when the defined address (e.g. X001, Y001, C1, or T1) is OFF</td>
<td><img src="image" alt="Normally Closed Contact" /></td>
</tr>
<tr>
<td>Out Coil</td>
<td>Turns ON/OFF a given address (e.g. Y001 or C1) when the rung is true/false.</td>
<td><img src="image" alt="Out Coil" /></td>
</tr>
<tr>
<td>Timer</td>
<td>When enabled, measures the elapsed time. Turns on output address once it reaches the set point.</td>
<td><img src="image" alt="Timer" /></td>
</tr>
<tr>
<td>Counter</td>
<td>When enabled, counts up or down until it reaches the set point</td>
<td><img src="image" alt="Counter" /></td>
</tr>
<tr>
<td>End Instruction</td>
<td>Marks the termination point of a program</td>
<td><img src="image" alt="End Instruction" /></td>
</tr>
</tbody>
</table>