

Tennessee Technological University  
Programmable Logic Controllers  
Lab 1

## 1 Introduction

**In this lab you gain experience with:**

1. Analyzing how a ladder logic program will execute
2. Using the output coil (OTE)
3. Using the normally closed contact (XIO)
4. Using the normally open contact (XIC)
5. Coding various logical problems

### 1.1 Lab Files

Go to iLearn and download the PLC and HMI files for this lab to the PC. Then download the PLC project to the PLC and the HMI application to the HMI.

### 1.2 Acceptable Instructions

You may have previous experience with PLCs and that is great! However, you are only allowed to use the instructions that we have covered thus far in the lab. So, if you have experience already, consider it a challenge to restrict yourself to only use the instructions that have been covered thus far in lecture to solve the problem!

### 1.3 How to Interface with the PLC and HMI

If you are unclear on any of the following, refer to the Lab 1 Manual:

1. Download to the PLC
2. Go online with the PLC
3. Put the PLC into Run Mode
4. Make online and offline edits to the PLC program
5. Download to the HMI
6. Toggle a boolean tag

## 1.4 How to get credit

Each lab after this will require each student to submit the completed pre-lab before they are allowed to begin working on the lab. The **pre-lab must be submitted to the TA before beginning work on the lab.** If it is not complete then you will be required to complete the pre-lab before you are allowed to begin working on the lab.

In order to get credit for completing each part of this lab, **you must personally read and complete each portion of the lab and demonstrate the completion to the TA.** Each section has one or more signature slots that must be signed by the TA to confirm that the section was completed. Each section is worth equal credit.

## 1.5 20 minute grace period

To receive full credit, the lab must be completed and demonstrated during the assigned lab time. However, if you cannot complete the lab within that time, you can complete and demonstrate the lab within the first 20 minutes of the subsequent lab time and still receive full credit. **If the lab is not completed within the assigned lab time and is not completed within the 20 minute grace period, then the lab is considered late.** If you submit the lab late, then there will be a 20% deduction compounded weekly.

## 1.6 Lab agreement

The planning of a program is often a very social activity, however the actual writing of the code is always an individual pursuit. In this class it is very much the same. Students are welcome to verbally assist each other, but each person is required to write their own code and personally complete each lab. In this way each student will gain valuable experience with programming PLCs.

**The undersigned person guarantees that any and all work demonstrated to the TA in regard to this lab is a result of their own work with no unauthorized help.**

Student (Print & Sign): \_\_\_\_\_

## 2 Storefront Door Controller

This section corresponds to the Lab1\_1 object in the Lab1 PLC file.

Picture it, you have just graduated and a local grocery store contacts you and asks if you would be willing to do some freelance engineering/programming work for them. The first thing they ask you to do is to make their automatic doors work. They hired a company to install the automatic door system but the contract was disputed and the company left the work unfinished.

When you arrive at the store you find that the motor that opens the door is connected to an Allen Bradley PLC! You also find that the sensor used to detect the presence of a person in front of the door is also connected to the PLC. The PLC also gets a signal from the security system to know that the store is open for business. So, all that's left for you to do is program the PLC to control the signal that goes to the door.

### 2.1 How should the logic work?

Simple. Any time the store is open and someone is detected in front of the doors, then the doors should open.

This logic in normal sequential programming would look something like that shown in Fig.1

```

1 if (Lab1_1.Store_Is_Open and
2     Lab1_1.Person_Present_At_Front_Of_Door):
3     Lab1_1.Open_The_Door = True
4 else:
5     Lab1_1.Open_The_Door = False

```

Figure 1: Sequential logic similar to Lab1 part 1

## 2.2 The Inputs and Outputs

To access any of the signals listed in Table1, use the syntax `Lab1_1.` followed by the attribute name.

Table 1: Attributes available in Lab1\_1

Attribute Name	Data Type	Type
<code>Store_Is_Open</code>	Bool	Output
<code>Person_Present_At_Front_Of_Door</code>	Bool	Output
<code>Open_The_Door</code>	Bool	Input

Write the appropriate logic in the associated rung in the PLC file.

TA Signature 1: \_\_\_\_\_

## 3 Storefront Door Controller... again

This section corresponds to the `Lab1_2` object in the Lab1 PLC file.

When you left the grocery store, the door worked beautifully and your customer was very happy. However, after a few hours you receive a frantic phone call from the store manager asking you to return to the store because there was a problem with the doors!

When you return to the store you find that the store is packed with people but they are all standing inside looking out the front door like a hoard of zombies! What could cause such a weird situation? Well, it seems that your door programming works well at letting people in but since the sensor used to detect people at the back of the door was never connected you did not think to add it. So, people can get in but no one can get out because the `Person_Present_At_Front_Of_Door` signal doesn't work for people at the back of the door.

*—Note 1— The first automatic doors were designed by Heron of Alexandria in the first century AD! He made a mechanism driven by the heat from a fire that was kindled by priests in the temple to cause a water counter balance to open the temple door.*

You quickly install the motion detection sensor used to detect the presence of a person at the back of the door and connect the sensor to the PLC. Now you have to rewrite your code!

### 3.1 How should the logic work?

If the store is open and a person is in front of the doors or in back of the doors, then the doors should be open.

This logic in normal sequential programming would look something like that shown in Fig.2

```

1 if (Lab1_2.Store_Is_Open and
2     (Lab1_2.Person_Present_At_Front_Of_Door or
3      Lab1_2.Person_Present_At_Back_Of_Door)):
4     Lab1_2.Open_The_Door = True
5 else:
6     Lab1_2.Open_The_Door = False
7

```

Figure 2: Sequential logic similar to Lab1 part 2

### 3.2 The Inputs and Outputs

To access any of the signals listed in Table2, use the syntax `Lab1_2.` followed by the attribute name.

Table 2: Attributes available in `Lab1_2`

Attribute Name	Data Type	Type
<code>Store_Is_Open</code>	Bool	Output
<code>Person_Present_At_Front_Of_Door</code>	Bool	Output
<code>Person_Present_At_Back_Of_Door</code>	Bool	Output
<code>Open_The_Door</code>	Bool	Input

Write the appropriate logic in the associated rung in the PLC file.

TA Signature 2: \_\_\_\_\_

## 4 Sawmill controller

This section corresponds to the `Lab1_3` object in the Lab1 PLC file.

Word got out that you were a good programmer (even with the minor blunder with the zombie hoard at the grocery store). Pretty soon you are contacted by the owner of a sawmill that wants to replace the old relay system that currently controls the saw with a PLC based system.

### 4.1 How should the logic work?

The current system requires the saw operator press the start button to start the saw. But once the saw is running, it will remain running until the stop button is pressed. He wants the saw to work precisely the same way after you are done.

*—Note 2— What the sawmill owner wants is referred to in ladder logic as a **seal-in**. It is a useful way of keeping a coil energized (on) once it has been turned on. This is a very important concept in PLC programming.*

This logic in normal sequential programming would look something like that shown in Fig.3

### 4.2 The Inputs and Outputs

To access any of the signals listed in Table3, use the syntax `Lab1_3.` followed by the attribute name.

Write the appropriate logic in the associated rung in the PLC file.

```

1
2 if ((Lab1_3.Start_Saw or Lab1_3.Run_Saw) and
3     not Lab1_3.Stop_Saw):
4     Lab1_3.Run_Saw = True
5 else:
6     Lab1_3.Run_Saw = False

```

Figure 3: Sequential logic similar to Lab1 part 3

Table 3: Attributes available in Lab1_3		
Attribute Name	Data Type	Type
Start_Saw	Bool	Output
Stop_Saw	Bool	Output
Run_Saw	Bool	Input

TA Signature 3: \_\_\_\_\_

## 5 Challenge

This section corresponds to the `Lab1_4` object in the Lab1 PLC file.

You decided that to get more publicity you would enter a PLC programming competition that sends out challenges once a week. If you win there's even a guaranteed job offer! This week the contest judges decided that each competitor must code a boolean formula into a single rung of ladder logic using only the normally closed contact, normally open contact, and the coil instructions. The formula that they have given is  $D = ((A + B) \cdot C \cdot A)$ .

The reason that this is challenging is because there is no good way to negate a group of logical operations in ladder logic in a single rung. The key is to use Demorgan's theorem to transform the boolean formula into one that **doesn't have any logical NOT operations applied to a group**.

—Note 3— In electrical engineering, the `+` operator signifies a logical OR operation. The `·` operator is used to signify the logical AND operation. Lastly, the bar over top of an element(s) signifies the logical NOT operation.

—Note 4— Demorgan's theorem states that:

$$\overline{(A \cdot B)} = \overline{A} + \overline{B} \quad (1)$$

$$\overline{(A + B)} = \overline{A} \cdot \overline{B} \quad (2)$$

—Note 5— This situation comes up often in industrial PLC applications. Before a robot is allowed to interface with a sub-section of a machine, the machine must be at rest (otherwise you risk damaging the machine). This typically involves making sure that a group of things are NOT true. Negating the group is ugly in ladder logic and it is typically better to apply Demorgan's theorem to get the boolean formula into a more manageable form.

### 5.1 How should the logic work?

That's for you to figure out!

## 5.2 The Inputs and Outputs

To access any of the signals listed in Table4, use the syntax `Lab1_4.` followed by the attribute name.

Table 4: Attributes available in Lab1\_4

Attribute Name	Data Type	Type
A	Bool	Output
B	Bool	Output
C	Bool	Output
D	Bool	Input

TA Signature 4: \_\_\_\_\_