Tennessee Technological University
Programmable Logic Controllers
Lab 3

1 Introduction

In this lab you gain experience with:

1. Arithmetic instructions
2. Compare instructions
3. Implementing counters without using a counter instruction
4. Implementing modulo functionality without using a modulo instruction

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!

You are **not** allowed to use the modulo instruction, the counter instructions, the compute instruction, or the compare instruction. And as always you are **not** allowed to use the oneshot instruction (or any of the other rising/falling edge instructions).

1.3 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 Transmission Makers of America

This section corresponds to the `Lab3_1` object in the Lab3 PLC file.

Transmission Makers of America (not a real company) is a tier 1 supplier of transmissions to automakers in the US. Their transmissions contain multiple gear packs which engage and disengage to provide the varying speed to torque relationship that cars require.

Transmissions are difficult to manufacture because they require the gears to be meshed with very little backlash, and they must not be under a large amount of squeeze. One way to achieve such a precise fit would be to build every component absolutely perfectly... But that is extremely cost prohibitive. So, instead of counting on perfection, Transmission Makers of America use precisely sized shims to adjust the overall mesh depth of the gears, brilliant!

The problem they are having is that the formula to calculate the correct size shim is quite complicated. Often when their employees calculate the necessary shim thickness, there are small math errors that render the transmission unusable. They have contracted you to code the fomula into a PLC and display the correct shim thickness on the HMI. They have already created the HMI and it has been provided to you so all you have to do is move the correct shim thickness into the `Shim_Thickness` attribute from Table1.

*—Note 1— Backlash is the term used to describe the amount of separation between two gears. So, given gear A and gear B which are meshed together, the backlash is the rotation gear A can have **without** causing gear B to rotate.*

—Note 2— Tier 1 auto suppliers are those which are under contract to supply automotive components directly to automakers like GM, Ford, Toyota, etc.

In reality, Transmission Makers of America only has shims in a few sizes. So, they use the shim that is closest in size to the ideal which will be calculated in the PLC.

$$\frac{Gear\_Diameter \cdot \sin\left(Backlash\right) \cdot Pinion\_Gear\_Depth}{\left(Drag\_Torque \cdot 20\right) + 3} - 2.32 \qquad (1)$$

2.1 How should the logic work?

That's for you to figure out.

2.2 The Inputs and Outputs

To access any of the signals listed in Table1, use the syntax `Lab3_1.` followed by the attribute name.

Table 1: Attributes available in Lab3_1

Attribute Name	Data Type	Type
`Backlash`	Real	Output
`Pinion_Gear_Depth`	Real	Output
`Gear_Diameter`	Real	Output
`Drag_Torque`	Real	Output
`Shim_Thickness`	Real	Input

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

TA Signature 1: ______________________

3 Challenge - Modulo

This section corresponds to the `Lab3_2` object in the Lab3 PLC file.

Another weekly challenge from the competition for which you signed up! Write the logic necessary to calculate the modulo of two operands while only using the truncate, add, subtract, multiply, and divide instructions. Store the modulo of `OperandA` and `OperandB` in `Calculated_Modulo`.

You may choose to create a new tag to store a temporary result. This is acceptable. If you need help creating a new tag, refer to Fig.1. If the data you intend to store in the tag is not a boolean value **make sure that the datatype of the tag you create matches the datatype which you intend to store**. For this problem you will probably need to create a tag with datatype "Dint" or possibly "Real".

—*Note 3— The modulo operation is typically denoted by the % symbol.*

—*Note 4— The TRN (truncate) instruction can be found in the Allen Bradley instruction set manual that is available. The truncate instruction removes the decimal portion of a number and leaves it whole without any rounding. ie.* $TRN(6.1) = 6.0$ *and* $TRN(6.99) = 6.0$

3.1 How should the logic work?

The modulo command calculates the remainder from a division operation. So, given `OperandA` and `OperandB`, calculate the remainder from $OperandA/OperandB$. As an example, if `OperandA` is 5 and `OperandB` is 3, then $OperandA\%OperandB = 2$. As a second example, if `OperandA` is 13 and `OperandB` is 6, then $OperandA\%OperandB = 1$. As a third example, if `OperandA` is 14 and `OperandB` is 7, then $OperandA\%OperandB = 0$. As a final example, if `OperandA` is 5 and `OperandB` is 2, then $OperandA\%OperandB = 1$.

—*Note 5— One of the powerful uses of the modulo operation, is it's ability to identify positive and negative numbers. Notice in the final example,* $5\%2 = 1$. *If the remainder of any number is non-zero after being divided by 2, then that number is odd.*

3.2 The Inputs and Outputs

To access any of the signals listed in Table2, use the syntax `Lab3_2.` followed by the attribute name.

Table 2: Attributes available in Lab3_2

Attribute Name	Data Type	Type
`OperandA`	Dint	Output
`OperandB`	Dint	Output
`Calculated_Modulo`	Dint	Input

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

TA Signature 2: ______________________________

4 7 Boxes and Counting

This section corresponds to the `Lab3_3` object in the Lab3 PLC file.

The online retailer (You signed an non-disclosure agreement so you can't speak their name) which contracted you to automate their box moving process has now become a repeat customer! They now want you to implement a counting system to count how many boxes have been transferred to trucks.

They intend to use the box transfer counter as a means of keeping each shift on track. Each shift is intended to transfer 7 heavy parcels. However, some shifts are not hitting their goal. So, by having a counter on the screen to keep track of their progress, the retailer hopes to increase productivity.

They also want you to reset the count when the shift leader hits the reset button on the HMI. Moreover, they want you to send a signal to the HMI when the shift goal is met.

4.1 How should the logic work?

Each time `Start_Transfer` goes from false to true, they want you to increment the value stored in `Current_Count` to keep track of the number of boxes that have been transferred. They also want you to reset the counter to 0 whenever the `Reset_Counter` tag is true. Finally, if the value in `Current_Count` is greater than or equal 7, you should turn on the `Shift_Goal_Met` bit.

4.2 The Inputs and Outputs

To access any of the signals listed in Table3, use the syntax `Lab3_3.` followed by the attribute name.

Table 3: Attributes available in Lab3_3

Attribute Name	Data Type	Type
`Start_Transfer`	Bool	Output
`Reset_Counter`	Bool	Output
`Current_Count`	Dint	Input
`Shift_Goal_Met`	Bool	Input

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

TA Signature 3: ______________________

Open Parameters and Local Tags

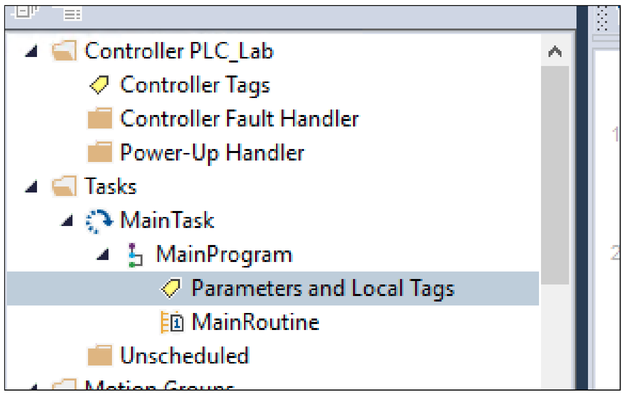
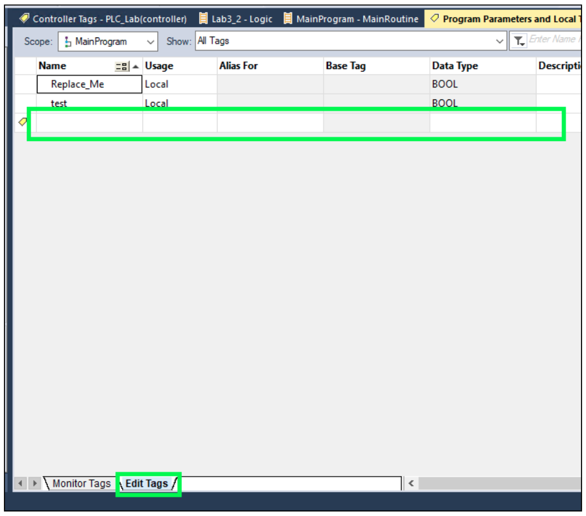


Figure 1: First step to creating a new boolean tag

Create new Boolean Tag


Controller Tags - PLC_Lab(controller)
Lab3_2 - Logic
MainProgram - MainRoutine
Program Parameters and Local T
Enter Name
Monitor Tags
Edit Tags

Scope: MainProgram Show: All Tags

Name	Usage	Alias For	Base Tag	Data Type	Descripti
Replace_Me	Local			BOOL	
test	Local			BOOL	

Figure 2: Second and Third step to creating a new boolean tag

5 How to create a boolean tag

To create another boolean tag to store the result of a boolean operation. Go to the left hand controller organizer menu. Under Main Program, double the item named Parameters and Local Tags. Refer to Fig.1.

Next, in the window that appears insure that you are on the edit tab of the parameters and tags window. In Fig.2 you can see that the edit tab is in a green box for visibility at the bottom left.

Finally, in the bottom entry in the list of tags enter the details for the tag that you are creating. The only two items that you should enter are the name and the datatype. The name must be a name that is not already taken and the datatype must be BOOL.