# Tennessee Technological University
# Programmable Logic Controllers
# Lab 4 & 5

## 1 Introduction

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

1. More arithmetic instructions
2. More compare instructions
3. Implementing counters using the counter instruction
4. Timers

### 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!

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 7 Boxes and Counting... Again

This section corresponds to the `Lab5_1` object in the Lab5 PLC file.

The online retailer likes the work counting system you put in place to help them meet productivity. However, when their onsite technicians opened up the PLC code to make an adjustment, they were confused by what you had written. The online retailer has requested that you return to adjust your logic to use an Allen Bradley counter instruction instead of your hand implemented instruction.

You are a bit annoyed by their request, but they are an important customer. So, you decided the best course of action is to put on a happy face and adjust your code.

### 2.1 How should the logic work?

You will have to create a tag of type COUNTER with the name `Box_Counter`. To create a tag of type counter, refer to Fig.3. The instructions there are for creating a boolean tag. So, the only difference will be that you will select the datatype COUNTER rather than BOOL.

After creating the counter tag, you will have to insert a count up (CTU) instruction into the logic. When you insert the CTU instruction, there are 3 available fields to customize the instruction to your particular needs. In Fig.1 notice the fields adjacent to the words counter, preset, and accum.

**CTU Instruction**

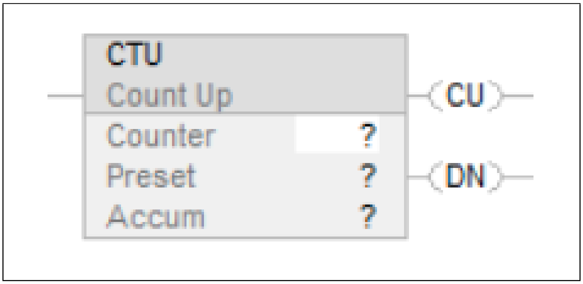


Figure 1: Count up instruction

The field beside the word counter is where you are to type the name of the COUNTER type tag you have created. The preset is the value which you would like to count up to. So, in this case you are counting up to the value 7. So, the preset should be 7. You can enter the value 7 directly into the preset field.

The .DN bit which is part of the COUNTER type tag associated with the CTU instruction will be false until the counter value reaches the preset value. When the count value reaches the preset value, the .DN bit will become true. The .DN bit should be useful in deciding when to turn on `Shift_Goal_Met`.

*—Note 1— If you wanted to programmatically set the the preset value, you can do so by moving a value into the .PRE attribute which is a part of the COUNTER type tag associated with the timer instruction.*

To reset the counter, you will use a reset instruction (RES). This should be controlled by the `Reset_Counter` signal.

### 2.2 The Inputs and Outputs

To access any of the signals listed in Table1, use the syntax `Lab5_1.` followed by the attribute name.

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

TA Signature 1: ____________________________

Table 1: Attributes available in Lab5_1

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

## 3 Challenge - The Maze Runner

This section corresponds to the `Lab5_2` object in the Lab5 PLC file.

This week the challenge activity is going to be a bit *time* consuming (Pun!). This week the contest organizers decided that each contest will have to write a ladder logic program to guide a ball through a maze... automatically! The maze and the ball will appear on the HMI and you have to make the ball successfully navigate the maze!

### 3.1 How should the logic work?

The logic might seem tough at first glance but it's nothing you can't handle. The way to solve this problem is by writing a bit latch sequence. Each step in the sequence will control a timer and should turn on the appropriate direction control from Table2. You stay on that step until the timer completes. Then you move on to the next step. Hint: You will need one step for each change in direction that that the gold ball must make. ie. if there were 3 turns, then you would need a sequence with 3 steps.

Hint: The length of time that each step should be on will be a bit of a guessing game. You'll have to figure out how long it takes for the ball to travel from where it starts to where you want it to be.

As a further requirement, when the `Reset` becomes true you must unlatch all the steps in your sequence. When `Go` becomes true the sequence should start and continue until the `Reset` becomes true. Follow the approach to bit latch sequencing that has been discussed in lecture.

Hint: You will need to use timers to accomplish this task, obviously. You can complete it with either the TON timer or the RTO timer.

### 3.2 The Inputs and Outputs

To access any of the signals listed in Table2, use the syntax `Lab5_2.` followed by the attribute name.

Table 2: Attributes available in Lab5_2

| Attribute Name | Data Type | Type |
|---|---|---|
| `Go` | Bool | Output |
| `Reset` | Bool | Output |
| `Go_Up` | Dint | Input |
| `Go_Down` | Bool | Input |
| `Go_Left` | Bool | Input |
| `Go_Right` | Bool | Input |

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

TA Signature 2: ______________________

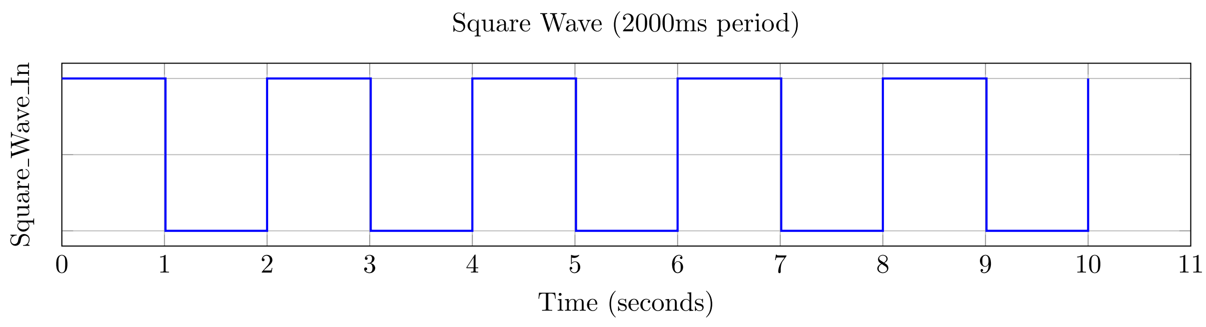


Figure 2: Square Waveform with a 2 second (2000 ms) period

## 4 Double Challenge - Simple Waveform

This week we have a rare double challenge. For the second challenge you are to use timers to make the waveform shown in Fig.2 appear in the axes on the HMI. However, the period of the square wave should be editable from the HMI.

### 4.1 How should the logic work?

The axes shown on the HMI are continuously plotting the state of the `Lab5_3` attribute called `Square_Wave_In`. So, to make a squarewave appear on the axes, you must set `Square_Wave_In` to true for an amount of time equal to one half of the period and then set the tag false for the same amount of time.

There is a field on the HMI that allows the user to edit the value stored in the `Lab5_3` attribute called `Period`. The value stored in this tag should be used as the period for your square wave.

### 4.2 The Inputs and Outputs

To access any of the signals listed in Table3, use the syntax `Lab5_3.` followed by the attribute name.

Table 3: Attributes available in Lab5_3

| Attribute Name | Data Type | Type |
|---|---|---|
| `Period` | Real | Output |
| `Square_Wave_In` | Bool | Input |

TA Signature 3: ________________________

## 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.3.

Next, in the window that appears insure that you are on the edit tab of the parameters and tags window. In Fig.4 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.

**Open Parameters and Local Tags**

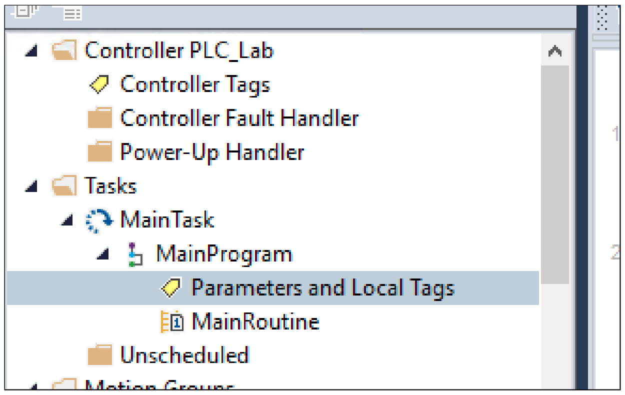
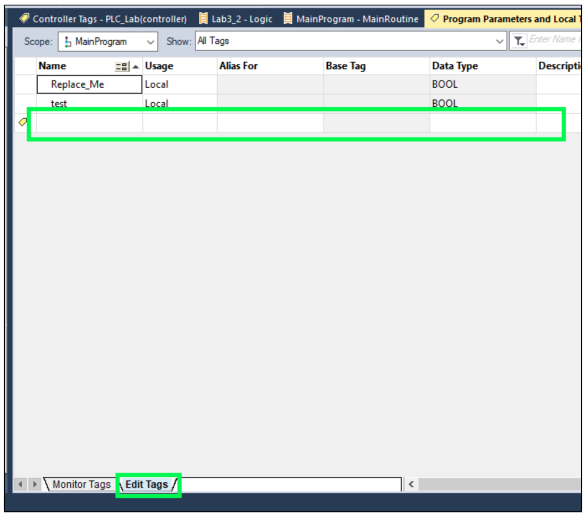


Figure 3: First step to creating a new boolean tag

**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 4: Second and Third step to creating a new boolean tag