1 Overview

Industrial control system (ICS) is a generic term for the broad class of automation systems associated with controlling and monitoring of manufacturing and industrial facilities. A Programmable Logic Controller (PLC) is an industrial digital computer used to control the manufacturing processes. Traditionally, discussions concerning ICS and PLCs refer to the control systems that regulate critical infrastructure.

1.1 Information Technology vs Operational Technology

Information Technology (IT) refers to the use of systems (especially computers) to store, retrieve and send information. Operational Technology (OT) refers to the hardware and software to operate Industrial Control Systems. OT differs from IT in that OT (1) has a much longer lifespan (10 years) (2) is designed with improved shock resistance (3) runs with little human interaction (4) requires 100% availability.

1.2 Understanding PLCs

A PLC contains a CPU, for logic and processing, memory, and input/output circuits. A running PLC continually scans a program. The scanning cycle is composed of three phases: (1) Checking inputs - the PLC will read information from any connected sensors (2) Execute Program Logic - the PLC will execute the uploaded program logic (3) Writing outputs - the PLC will write to the output devices (actuators) based on the executed logic.

1.3 IEC-61131-3 Standard Protocol

The International Electrotechnical Commission (IEC) 61131-3 standard protocol specifies how PLCs are manufactured and programmed. IEC-61131-3 specifically defines the five programming languages supported by PLCs. The defined languages are (1) Ladder Diagrams (LD), (2) Function Block Diagram (FBD), (3) Structured Text (ST), (4) Instruction List (IL), and (5) Sequential Function Chart (SFC). This lab uses the most popular programming language for PLCs, Ladder Diagrams also known as Ladder Logic.

1.4 Lab Environment

Figure 1 depicts the lab environment.

In this lab, you will use two machines that are connected to the same LAN. The ICS Security machine contains the OpenPLC editor software, for creating and editing ladder diagrams, and the OpenPLC runtime software, which executes the ladder logic created from the OpenPLC editor.The SCADABR machine executes the HMI builder software. The SCADABR machine runs in the background. The HMI builder software is accessed using the OpenPLC machine, via a standard web browser using the SCADABR IP address and port 8080. For this lab the URL to access the HMI Builder software is http://192.168.10.25:8080/ScadaBR.

1.5 Lab Objectives

There are four objectives for this lab

1. introduce Industrial Control Systems
2. introduce Programmable Logic Controllers
3. introduce Programming with Ladder Diagrams
4. write simple ladder diagram programs

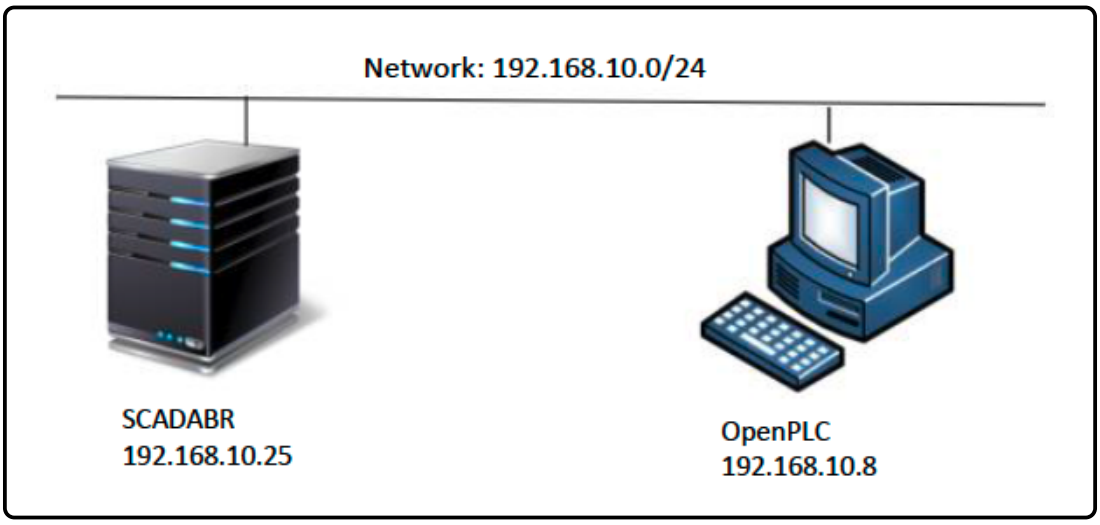

Figure 1: Lab Environment

2 Introduction to Ladder Logic

Figure 2 depicts the ladder diagram for a button and a LED. When the button is pressed the LED will illuminate, otherwise if the button is not pressed, then the LED will not illuminate.

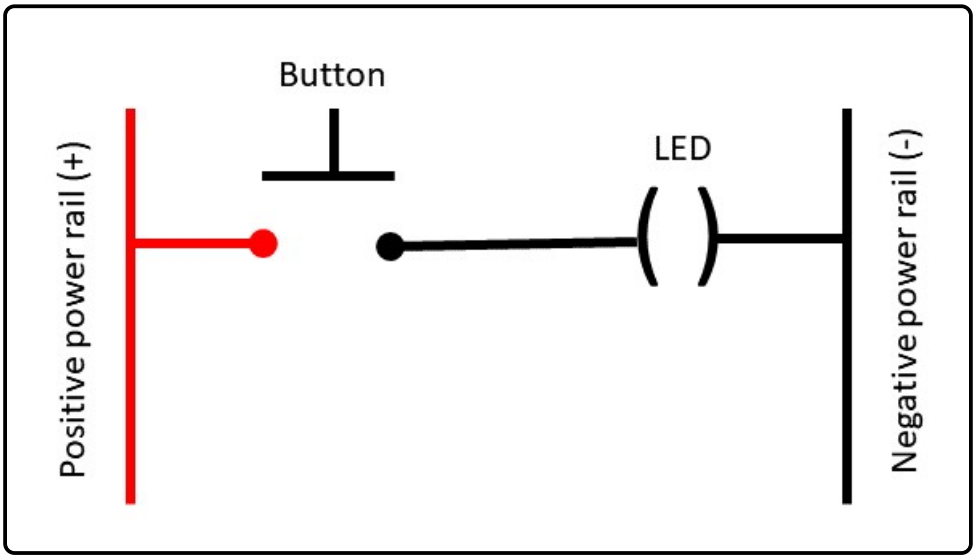

Figure 2: Ladder Diagram

Notice the symbols of a Ladder Diagram (LD) are very similar to an electrical diagram. There are a few key differences between LD and electrical diagrams. (1) Ladder diagrams are read top to bottom from left to right, similar to how the pages in a book are read. (2) Order is very important for a LD, which resembles a ladder shape, of rungs of logic.

Starting from the top the LD is parsed and evaluated based on the current value of any inputs (example: contacts under the button). After the complete LD is read, the outputs are written (example: LED coil). If there is a discrepancy for the value of an output in the ladder logic, the last occurring rung determines the

correct value for the output.

2.1 Relay Contacts

A contact is a fundamental element of a ladder diagram. Figure 3 depicts the different types of relay contacts. A contact is similar to a push button, and it has only two states: open or closed. A closed contact allows the the current flow to the next element. An open contact does not allow the current to flow to the next element.

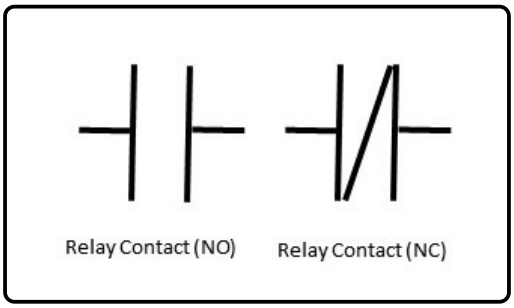

Figure 3: Relay Contacts

Relay Contact (Normally Open)

A normally open relay contact referred to as examine when closed/on. If this contact is activated, usually by PLC input, then the current flows and the result is true.

Relay Contact (Normally Closed)

A normally closed relay contact does not all the current to flow if the contact is closed. This contact behaves exactly opposite of the normally open relay contact. When activated, usually by PLC input, this normally closed relay contact will be false.

2.2 Coils

A coil is a fundamental element of a ladder diagram. Figure 4 depicts the different types of relay contacts. A coil is similar to a relay contact in that is has two states. If the coil is activated it is said to be energized. If the coil is not activated it is said to be de-energized.

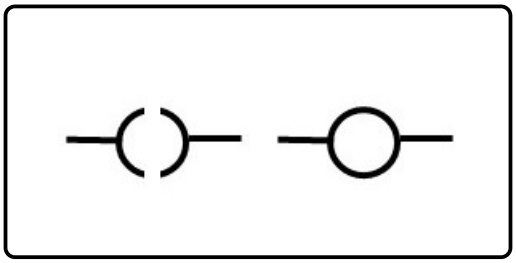

Figure 4: Coils

The Open Coil

Contacts are associated with inputs, coils are associated with outputs. If the series of instructions that precede the coil evaluate to true, the coil is energized. The activation of a coil can occur from writing to the PLC's I/O output device.

The Closed Coil

The closed coil is similar to a normally closed rely contact. The coil closed coil is only energized when the series of instructions that precede the coil evaluate to false.

2.3 PLC Logic and Scanning

Similar to other programming languages, ladder diagrams can evaluate logical operations, including logical OR and logical AND. Figure 5 represents two common scenarios when evaluating a ladder diagram.

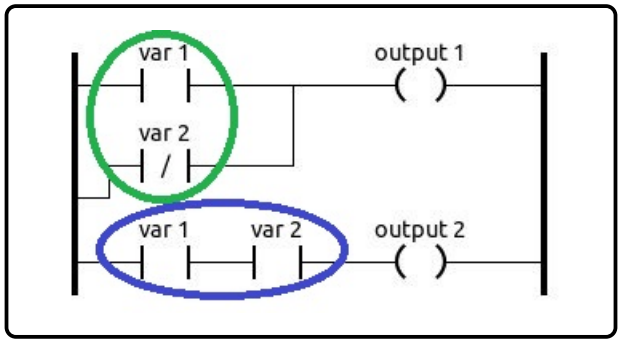

Figure 5: PLC Logic and Scanning

Logical OR

The green circle in Figure 5 represents a logical OR. The variable *var1* can either be set or TRUE, or not set or FALSE. The variable *var2* is a negated variable that can be set as TRUE; however because of the negation the output from *var2* will be FALSE. If *var2* is not set or FALSE, then because of the negation of *var2* the output will be TRUE. In the event that *var1* is NOT set or FALSE and if negated *var2* is not set the output of *var2* will be TRUE causing the value for *output1* will be energized or TRUE.

Logical AND

The blue circle in Figure 5 represents a logical AND. In order for *output2* to be energized or TRUE, then BOTH *var1* and *var2* MUST be activated or TRUE.

Scan Cycle

The PLC Scan Cycle is the cyclic process by which PLCs operate. Recall there are three phases of a PLC Scan Cycle. In phase 1, all of inputs are read, and the value of each input is determined. In phase 2, the logic is processed. This logic relies on the value of each input. In phase 3, after the logic is processed, the outputs are determined and written. The output is determined by the last rung in the ladder to write to that output.

Understanding Check #1

Using the logic from Figure 5 create a truth table for the possible inputs for *var1* and *var2* and the possible outputs for *output1* and *output2*. After you have completed the truth table answer the following questions.

Truth Table for inputs and output of Figure 5

var1	*var2*	neg *var2*	*output1*	*output2*
False	*False*			
False	*True*			
True	*False*			
True	*True*			

1. What are the values for *var1* and *var2* for only *output1* to be energized?
2. What are the values for *var1* and *var2* for only *output2* to be energized?
3. What are the values for *var1* and *var2* for both *output1* and *output2* to be energized?
4. What are the values for *var1* and *var2* for both *output1* and *output2* to NOT be energized?

2.4 The Function Block

Similar to other programming languages, ladder diagrams allow for relational operations. Relational operations are provided through the use of function blocks. Figure 5 represents the different function blocks available for ladder diagrams. The behavior of each function block varies greatly and provides a great deal of functionality to Ladder Logic.

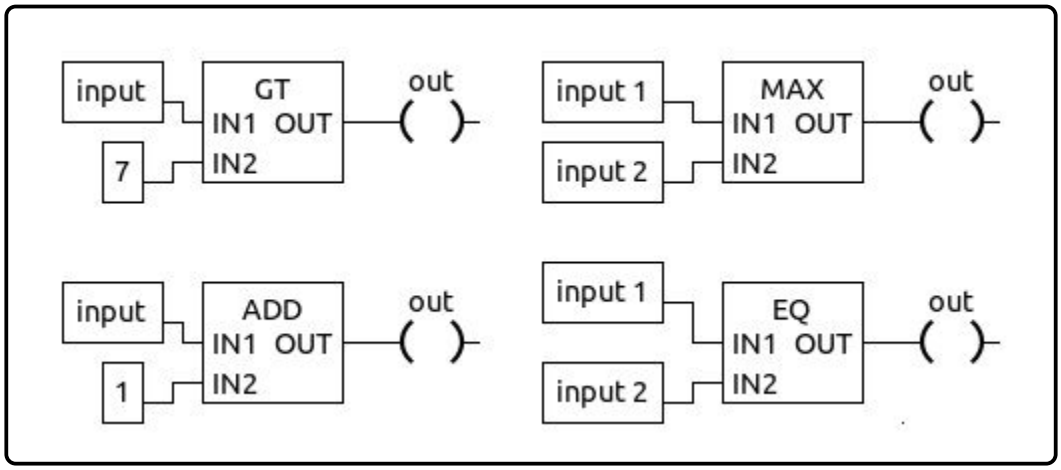

Figure 6: Function Blocks

The Function blocks defined here (left to right, top to bottom) are: Greater Than (GT), Maximum (MAX), Addition (ADD), and Equality (EQ). Each of these function blocks accepts two inputs. The inputs can be a constant value such as a number, TRUE/FALSE, etc. or the inputs can be connected to the preceding instructions. The results of a function block can either be a boolean or numeric value, that is sent to the output labeled OUT.

Understanding Check #2

Using the logic from Figure 7. Given all inputs labeled *input2* and all outputs labeled *output* are in the '0' or FALSE state answer the following questions.

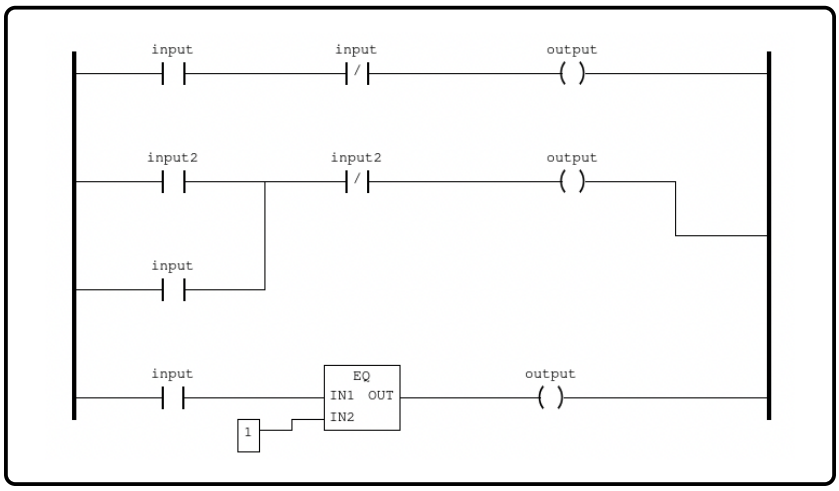

Figure 7: Ladder Diagram for Understanding Check #2

1. What will be the value of output after this ladder diagram is completed running?
2. What are the values for *input* and *input2* are required for *output* to be energized?
3. Would the original output change if the function block IN2 also started at '0'?

Similar to Understanding Check #1, create a truth table for each of the inputs and outputs. Use this completed the truth table to justify your answers.

3 Introduction to OpenPLC Editor

Creating, testing and executing ladder diagrams requires a PLC editor. OpenPLC Editor allows for writing PLC programs according to the IEC 61131-3 standard.

3.1 Installing OpenPLC Editor

OpenPLC Editor is installed in the ICSSecurity OVA available through the ICS Security GitHub. The directions for a clean installation can be found at https://www.openplcproject.com/plcopen-editor. Currently, the only two versions available for installation are for Windows and Linux.

3.2 Opening OpenPLC Editor

There are two ways to open the OpenPLC editor.

- If you created a virtual machine from the provided OVA, there is an icon on the desktop. Clicking the icon will open the OpenPLC editor.
- Open a terminal and execute the following command /Downloads/OpenPLC_Editor/openplc_editor.sh

The editor is divided into different quadrants. Figure 8 displays the editor without an open project. Figure 9 displays an open project containing variable declarations, the ladder diagram, configuration information, and a debugger window.

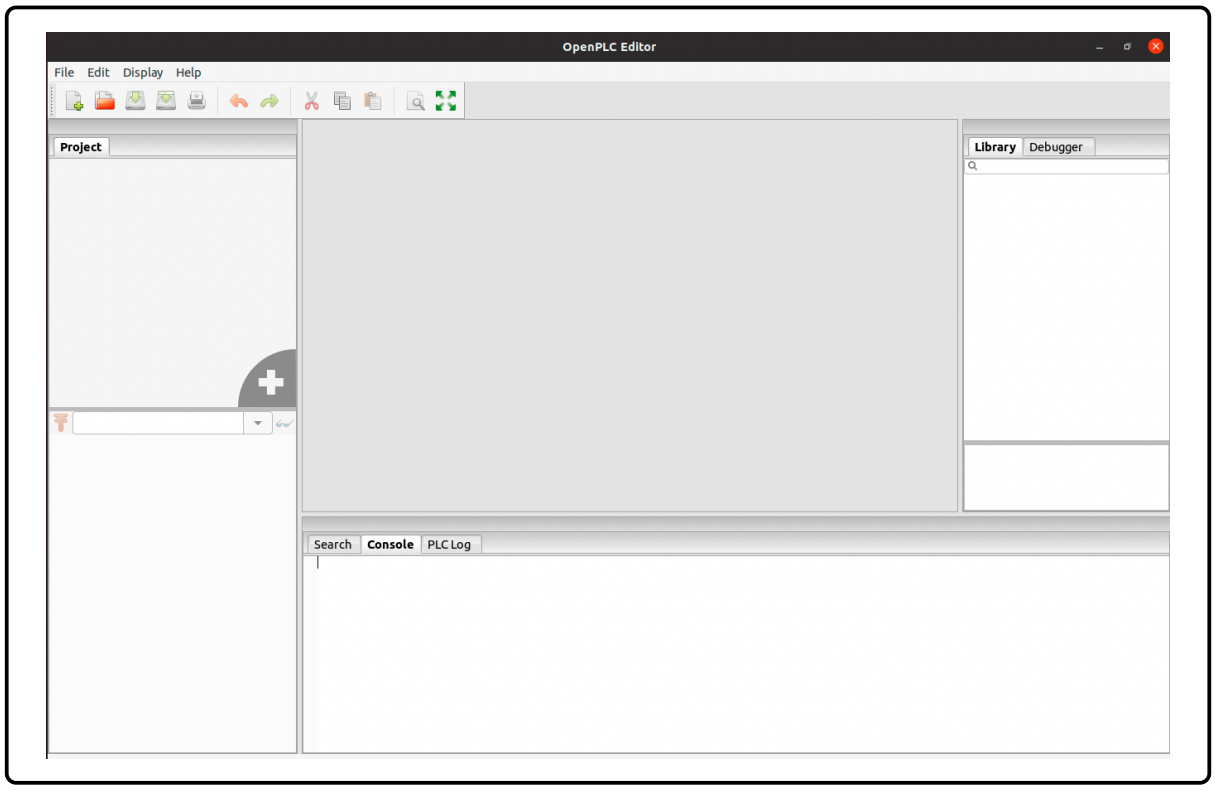

Figure 8: The OpenPLC Editor without a project or ladder diagram

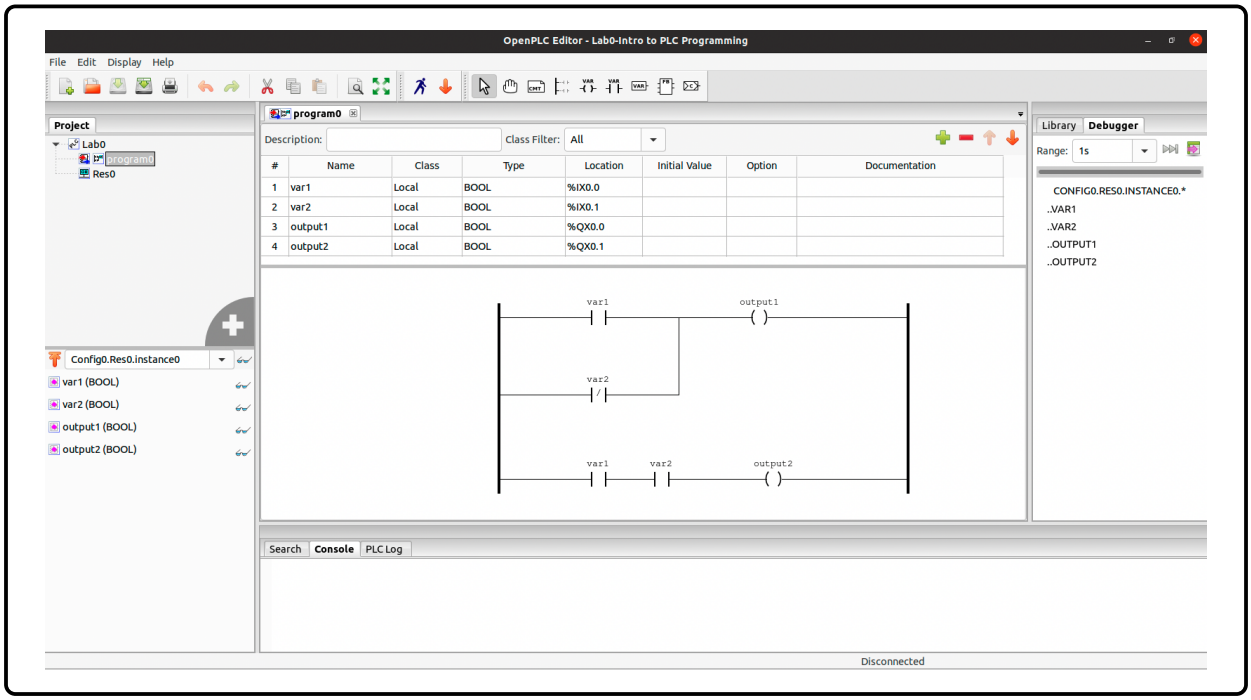

Figure 9: The OpenPLC Editor with a project, variable declarations and a ladder diagram

A tutorial video explaining the OpenPLC Editor, including creating projects, adding variables, creating ladder diagrams, and running the debugger can be found at [Insert URL here].

3.3 PLC Addressing for Variables

Figure 10 displays a ladder diagram and the corresponding variable declarations. Normally, a PLC application interacts with the outside world through communication protocols. For the OpenPLC Editor, these interactions occur through variables. When designing a ladder diagram, one area of consideration is which variables are available for external interaction. Any variables available for external interaction must by properly addressed in memory. The memory addressing for OpenPLC is explained below.

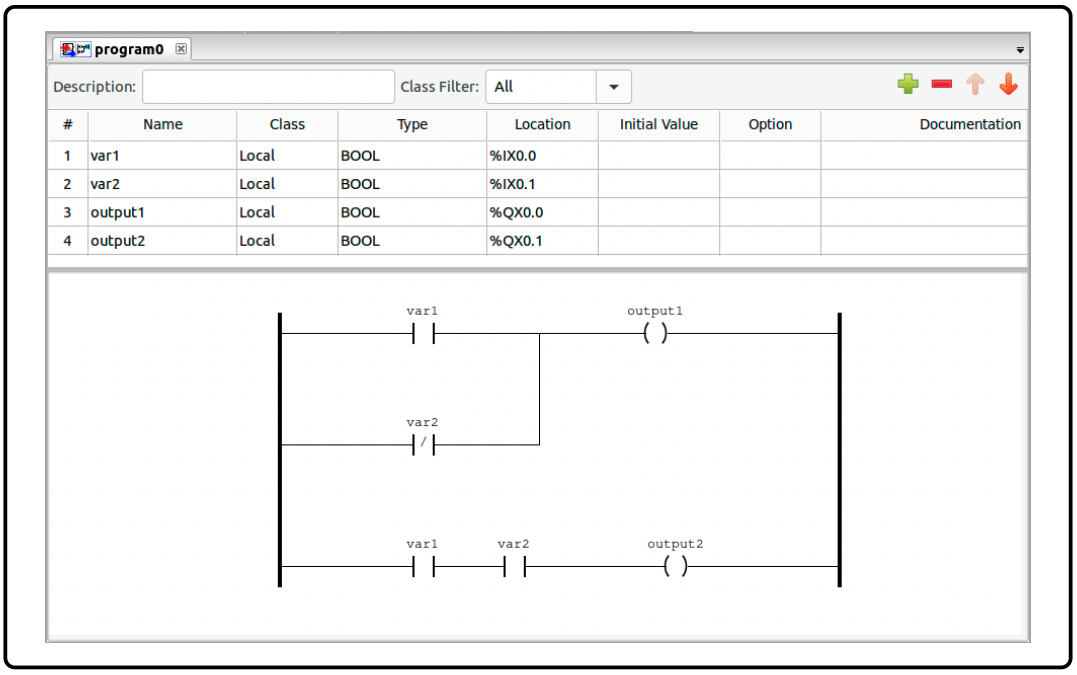

Figure 10: The OpenPLC Editor with a project, variable declarations and a ladder diagram

OpenPLC are composed of four ordered parts:

- Starts with a % symbol.
- Second is the **storage class**, indicating input or output.
- Third is the **data size**.
- Forth is the **hierarchical address** location

The **storage class**, generally, has the following interpretation: (I) inputs, (Q) outputs, and (M) memory. The **data size** requires a data type with a predefined number of bits. OpenPLC has a set of predefined data types with corresponding sizes.

Elementary Data Types and Data Sizes

Symbol	Data Type	Number of Bits	Elementary Data Types
X	Bit	1	BOOL
B	Byte	8	BYTE, SINT, USINT
W	Word	16	WORD, INT, UINT
D	Double Word	32	DWORD, DINT, UDINT, FLOAT
L	Long Word	64	LWORD, LINT, ULINT, DOUBLE

The **hierarchical address** is dependant on the data type. For a Bit the **hierarchical address** has two-parts. The value immediately following the symbol (X) must be a number in the range 0 to 1023. After the number is a period, and then following the period must be a number in the range 0 to 7. Consider this right-most value as a position in a byte.

The **hierarchical address** for other data sizes (Byte, Word, Double Word, Long Word), the value immediately following the symbol must be a number in the range of 0 to 1023.

Understanding Check #3

Given the following addresses: (1) determine which addresses are valid, (2) for the valid addresses determine the storage class, and the data type.

1. %IX0.8
2. %IX0.1
3. %IB1.1
4. %IB11
5. %QB1
6. %QX0.0
7. %IX0.0
8. %IB1
9. %QD100
10. %ML10

4 Lab Task - First OpenPLC Program

In **Understanding Check #2** you were asked to calculate the output for Figure 7. Using the OpenPLC Editor, your task is to verify your answers from **Understanding Check #2**, by creating and executing a ladder diagram. Figure 7 is repeated here for your reference.

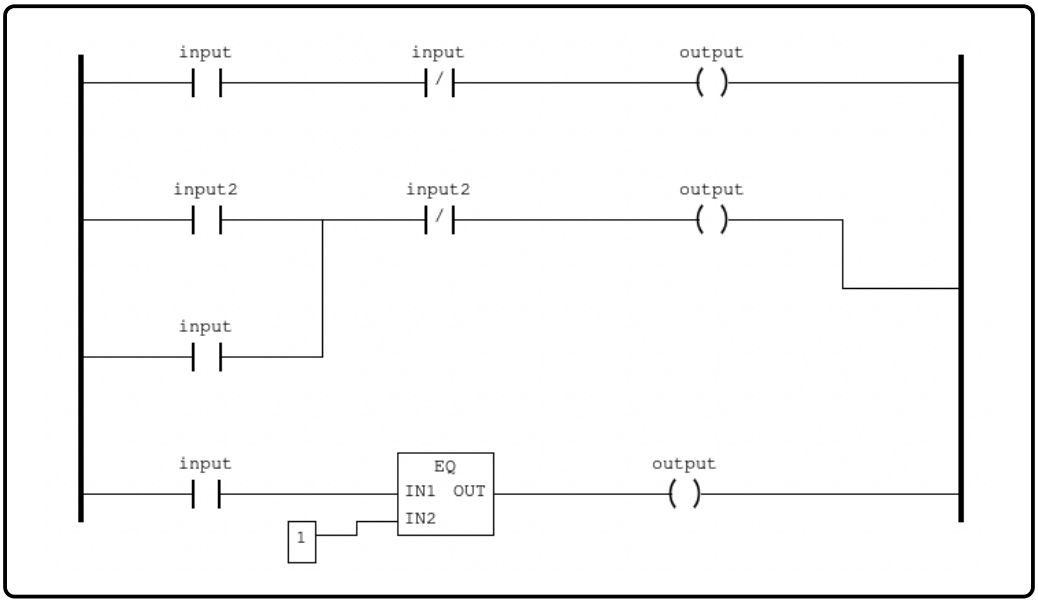

Figure 7: Repeated From Page 6

4.1 Specifications

You will submit a single PDF that will contain your answers from **Understanding Check #2** and screen captures from the OpenPLC Editor. Place your name and the lab number at the top of the page in the upper left hand section of the header. Your sections will be labeled using the title from each bullet below.

- **Section 1 - Understanding Check #2 Answers**

  Copy your truth table and answers to the questions from Understanding Check #2.

- **Section 2 - Ladder Diagram**

  Open the OpenPLC Editor and create a project named Lab0. Only create the ladder diagram shown in Figure 7. Take a screen capture of the ladder diagram and paste the image under this section.

- **Section 3 - Variable Declarations**

  In the OpenPLC editor add the variables as specified in the ladder diagram. You must determine the appropriate types for input, input2, and output. Once you have declared all your variables take a screen capture of the variable section and paste the image under this section. Write at least two sentences justifying why you chose the variable type for input, input2, and output.

- **Section 4 - Executing and Debugging your Ladder Diagram**

  In the upper left project window click on the program named lab0. In the configuration window on the lower left your variables will appear. Next click on the glasses next to each variable. clicking on the glasses next to the variable, will add your variables to the debugger window.

  Next run your program by clicking on the Running Figure to the left of the variable bar. Watch your variables in the debugging window and not the initial state of the variables and output. Based on your answers to Understanding Check #2 toggle the variables and verify your answers.

  Take at least two screen captures of the debugger window illustrating the toggling of your variables, and changing of your output value.

- **Section 5 - Lab Summary**

  Write a summary paragraph explaining your Understanding Check #2 answers versus the actual results from your PLC program. This discussion should include any challenges you encountered in creating the ladder diagram, variables, and running your code.

4.2 Submission

Submit your single PDF before the required due date and time. Name your PDF your last name first letter of your first name Lab0.pdf. (Example: steinersLab0.pdf)