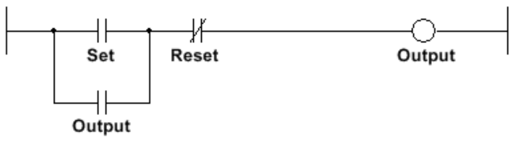
Hardware Engineer:	Basil Alsulami	Lab:	4
Software Engineer:	Andrew Struthers	Due:	January 31st, 2019

Lab 4: Boolean Logic Simplification

Purpose:

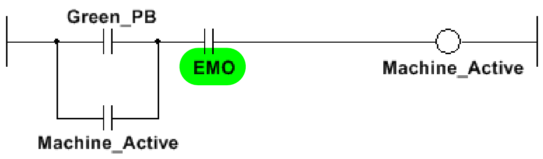
Boolean simplification allows programs to be significantly reduced in complexity by removing redundant or otherwise unnecessary elements from logic expressions. As seen in class, there are three primary methods to simplify logic circuits. These are by graphically analyzing the circuits, algebraically analyzing the Boolean expression, or by analyzing the original or alternate form of the truth table. This lab requires each group to use one or more of these methods to simplify the logic associated with the scenario described below.

In addition to simplifying logic, this lab will use, for the first time, a latching circuit. The ladder diagram below illustrates a typical latching circuit.



As is evident, when the **Set** contact is activated, the **Output** coil becomes active. The state of **Output** coil is placed in parallel with the **Set** contact to latch the output until the **Reset** button is pressed.

This circuit can be used in a variety of applications, however will only be used to establish a safety interlock system for this week's lab. The safety interlock circuit will be configured to disable the program any time the EMO (Red Mushroom) button is pressed and re-activate the program when the green pushbutton is pressed. The illustration below shows the latching circuit in this application. Notice that a program tag will need to be created to represent the **Machine_State** variable. This should be defined as a Bool variable type and **should be in series with each subsequent rung in the program.**



This safety interlock circuit will be used on all subsequent programs and will appear in alternate applications in the future.

Objectives:

Upon completion of this lab, you should be able to:

- Demonstrate momentary contact and relay interfacing techniques
- Implement control panel wiring solutions using NFPA 79
- Establish network connectivity using RSLinx Classic
- Apply Boolean Simplification Techniques to implement logic circuits
- Use latching circuits as safety interlocks

Materials:

1 *x*	PLC Trainer	1 *x*	Digital Multi-Meter Fluke 179 or equivalent

Procedure:

The Hardware and Software Engineer should evenly distribute the workload associated with this lab.

1. Before beginning any lab, always inspect the equipment to verify that all connections, including the chassis ground wires are secure and properly connected and that the equipment is safe to operate.

2. If necessary, use **Network Connectivity and Configuration** procedure from Lab 2 to configure the network connection for your PLC from the PC that you plan to use for the remainder of this lab.

3. Launch **Studio 5000** and create a new Project named "Lab4_Unit#" where # corresponds to your Unit number. For example, Unit 6 should contain a project called "Lab4_Unit6". **Save the project to your N: drive.**

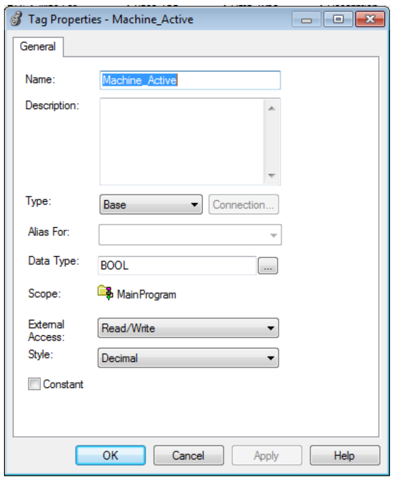
4. Configure the Controller I/O Configuration as outlined in Lab 2.

5. Configure the Program Tags using the same tag labels as used outlined in Lab 3.

   Note:

   Because the Red Flush Pushbutton will not be used for this lab, a tag does not need to be created.

6. Create a program tag called **Machine_Active**.
   a. Select Type: Base
   b. Select Data Type: BOOL


Tag Properties - Machine_Active

General

Name: Machine_Active

Description:

Type: Base

Alias For:

Data Type: BOOL

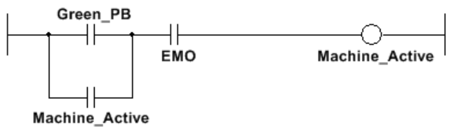
Scope: MainProgram

External Access: Read/Write

Style: Decimal

Constant

7. Use the illustration below to construct the safety interlock circuit on the first rung of the program.



8. Use a normally open (XIC) contact in series with each of your subsequent rungs to enable or disable the rungs accordingly.

9. Use the truth table below and the simplification techniques demonstrated in class to write a simplified Boolean expression for each output. Note that the 'x' represents a "Don't Care" state. From the truth table, we see that if **Machine_Active** = 0, then we don't care what the state is of the other inputs. In this case, the Red indicator will be active and the remaining indicators will be inactive.

Inputs					Outputs				
Machine_Active	**Black**	**Yellow**	**Blue**	**White**	**Red**	**Green**	**Yellow**	**Blue**	**White**
0	x	x	x	x	1	0	0	0	0
1	0	0	0	0	0	1	0	1	1
1	0	0	0	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1
1	0	0	1	1	0	1	1	0	1
1	0	1	0	0	0	1	0	0	0
1	0	1	0	1	0	1	1	0	1
1	0	1	1	0	0	1	1	0	0
1	0	1	1	1	0	1	1	0	0
1	1	0	0	0	0	1	0	1	1
1	1	0	0	1	0	1	1	1	1
1	1	0	1	0	0	1	1	1	1
1	1	0	1	1	0	1	1	1	1
1	1	1	0	0	0	1	0	1	1
1	1	1	0	1	0	1	1	1	1
1	1	1	1	0	0	1	1	0	0
1	1	1	1	1	0	1	1	0	0

10. Record the Boolean expressions in the space provided. Use "!" for **negation**, e.g., "not A" = !A.

Red_Indicator =	!Machine_Active
Green_Indicator =	Machine_Active
Yellow_Indicator =	Bl_PB + W_PB
Blue_Indicator =	!Bk*!Y*!Bl + Bk*!Bl + Bk*!Y*Bl + !Bk*!Y*Bl*!W
White_Indicator =	!Y + Bk*Y*!Bl + Y*!Bl*W

11. Implement the each Boolean expression as a separate rung of your program. As mentioned above, the **Machine_Active** contact should be in series with each logic circuit.

12. Use the truth table to test and troubleshoot your program.

13. A sign-off is not required for this lab.

Note:

Students enrolled in ETSC 522 must implement each of the logic circuits using both Ladder Diagrams and Function Block Diagrams. The Safety Interlock circuit can be implemented in the ladder.

For Submission:

- **Each Boolean expression as recorded in step 10.**
- **A copy of the Program submitted electronically as a PDF document.**