EET165 Lecture #3

1) **Review:** Review base conversion if needed.
   a. **Base 10 to Base 2 -** Successive division, divide by 2 and save the remainder
   b. **Base 10 to Base 16 -** Successive division, divide by 16 and save the remainder
   c. **Base 2 to Base 10 -** Multiply the digit by the place value, and add them all up
   d. **Base 16 to Base 10 -** Multiply the digit by the place value, and add them all up
   e. **Base 2 to Base 16** - Direct conversion using the chart
   f. **Base 16 to Base 2** - Direct conversion using the chart

2) **Digital Logic:** Base 10, Base 2, and Base 16 was covered last week. Let's focus in on Binary. Binary is just a way to express anything that has two states, such as heads/tails, on/off, positive/negative, and more. **When working with logic gates, we use 0 and 1**.

3) **Logic Gates:** There are entire classes that are dedicated to Digital Logic and Logic Gates. This will only be an introduction to a very large and complex topic. A Logic Gate is a specialize chip that will take in one or more input signals and make a decision on what the output should be.

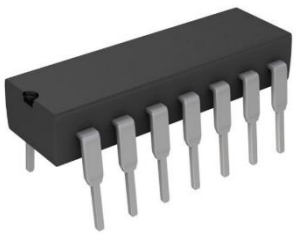
   Each of these gates has a unique symbol that is used in an electrical diagram or schematic.

   Each of these gates has a truth table to help you understand how it works. A truth table takes every possible input combination and shows you what the output is. You can calculate how many possible inputs by using the equation.

$$\textbf{Possible combinations} = 2^{\text{Number of Inputs}}$$

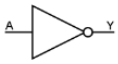
For 1 input: Possible combinations = $2^1 = 2$
For 2 inputs: Possible combinations = $2^2 = 4$
For 3 inputs: Possible combinations = $2^3 = 8$

   Each of these gates has a real-world chip that can be used to perform the job. Most of these chips are a 14-pin Duel In-Line Package (14 pin DIP).



4) **Logic gate, Truth table, and Logical expression:** The following are some of the basic logic gates used to control devices. When talking about logic gates, it is convention to assume the letters early in the alphabet are inputs and letters at the end of the alphabet are outputs. Think of the 1 as ON or as CONNECTED, and think of 0 as OFF or DISCONNECTED. Normally in a digital circuit, 1 is 5V and 0 is Ground (or 0V).

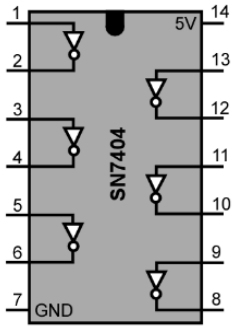
   a. **NOT Gate:** The NOT gate is the easiest of the gates to learn, it just flips a single bit. So, 0 (off) becomes 1 (on) and 1(on) becomes 0 (off). Below is the logic symbol, the truth table, and the logical expression. All three contain the same information.



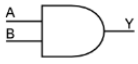
NOT

A	Y
0	1
1	0

$Y = (A)'$ or $Y = (\overline{A})$



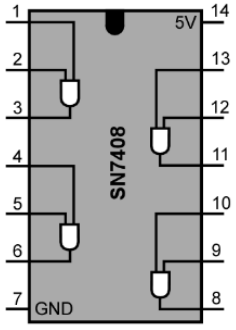
   b. **AND Gate:** The AND gate has to have all of its inputs as 1 (on) for it to have an output of 1(on). If ANY input is 0 (off), the output is also 0 (off).



AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = (A*B)$ or $Y = (AB)$

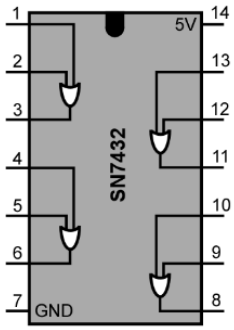


c. **OR Gate:** The OR gate will output a 1 (on) if ANY input is a 1 (on).

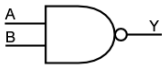
OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

$$Y = (A+B)$$

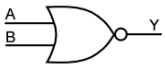


d. **Other Gates:** There are many other gates that can be used. Some are shown below.



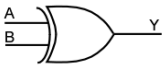
NAND

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



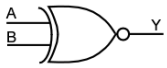
NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



NXOR

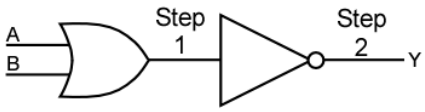
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

5) **Symbols, Truth tables, and diagrams:** The logic gates can be combined to create a customized circuit. There are three ways this information be conveyed. The there can be a logical diagram, a truth table, and a Boolean expression. All three methods hold the same information and you can convert one into another.

We will not be converting between the different methods in class, but there will be a brief demonstration of some these conversions so you can understand all three methods hold all of the same information.

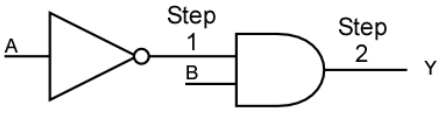
a. **Logic diagrams to Truth Tables:** These logic gates lock together like Legos to create a logic diagram. The output of one gate will feed the input of another to build a circuit. Stepping thorough the gates, you can generate the truth table.

Example 1: This diagram can be used to generate the truth table for this circuit.



		Step 1	Step 2	
A	B	OR	NOT	Y
0	0	0	1	1
0	1	1	0	0
1	0	1	0	0
1	1	1	0	0

Example 2: This diagram can be used to generate the truth table for this circuit.

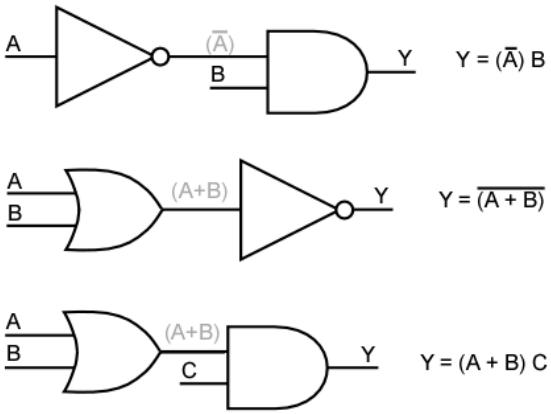


		Step 1		Step 2	
A	B	A'	B	AND	Y
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	1	0	1	0	0

b. **<u>Logic Diagram to Boolean Expression (or equation):</u>** Instead of drawing the gates, you can also write the logic diagram as an equation. To write the equation, you need to follow the inputs through the gates.

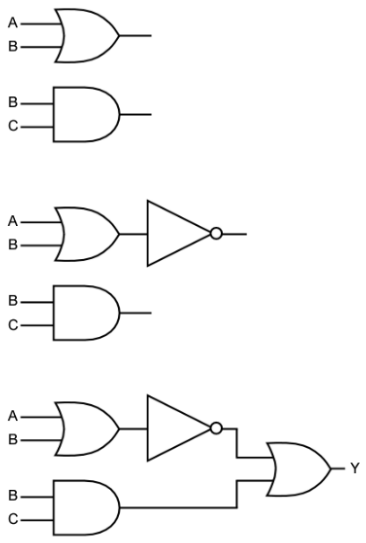
Every time you got through a gate, put the output in a set of parentheses. You do not need parentheses on the output of the last gate. You also do not need parentheses when there is only one item (such as A', B', C', and so on.)

For example:

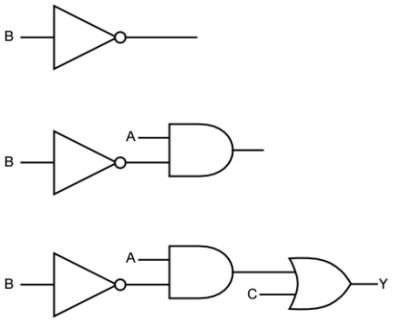


c. **<u>Boolean Equation to Logic Diagram:</u>** When converting an equation into a logic diagram, you just draw the gates following the order of operations:
   - Parentheses
   - NOT
   - AND
   - OR

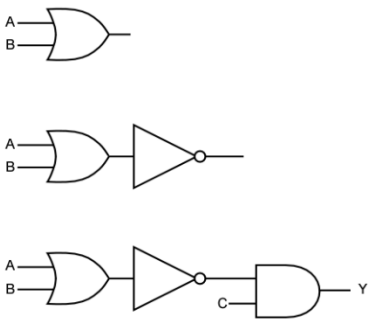
$$Y = (A+B)' + (C*B)$$



$$Y = (A*B') + C$$

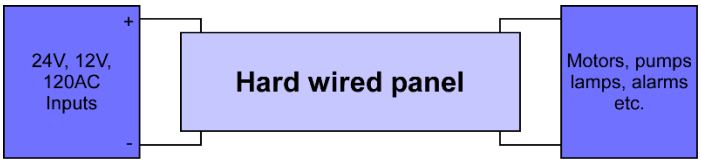


$$Y = (A+B)' * C$$



6) **Boolean Algebra:** There are many different ways to combine and simplify these types of circuits so you use the fewest number of gates. These include Boolean Algebra, De Morgan's Theorem, and Karnaugh Mapping.

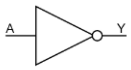
7) **Why 5V logic?:** You may be wondering why we are learning about 5V logic chips when we are controlling 3-phase 480V motors. That is a very good question. If you have experience wiring up a relay panel, you are familiar with this type of setup. The buttons and inputs are connected to relays and the relays turn the motors, pumps, and alarms on.



When we work with PLC's we are taking in buttons and inputs, but we are converting them into 5V/Ground (1/0). These 1/0 inputs are used to perform the calculations. When the logic has done its job, the ones and zeros are written to an output that us used to turn on the motors, pumps and alarms.

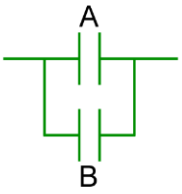
8) **<u>Gates to PLC:</u>** The gates can be converted into PLC inputs so the Logical Diagrams can be used to create a PLC program. The symbol that is a "normally closed" button in an electrical drawing is a "NOT" gate in PLC programming. The AND is two inputs in series. The Or is two inputs in parallel. The XOR and XNOR are a combination of all three of the previous gates.

NOT

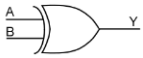


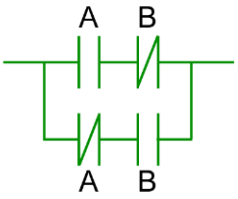
AND

OR

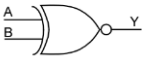


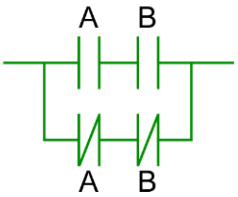
XOR



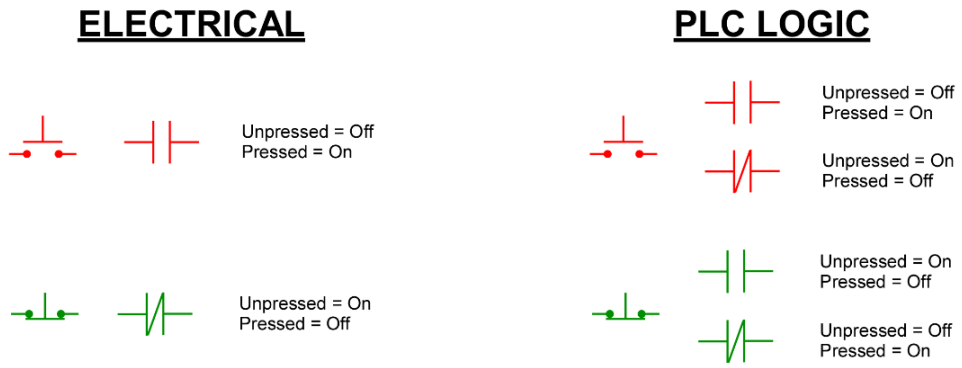


XNOR

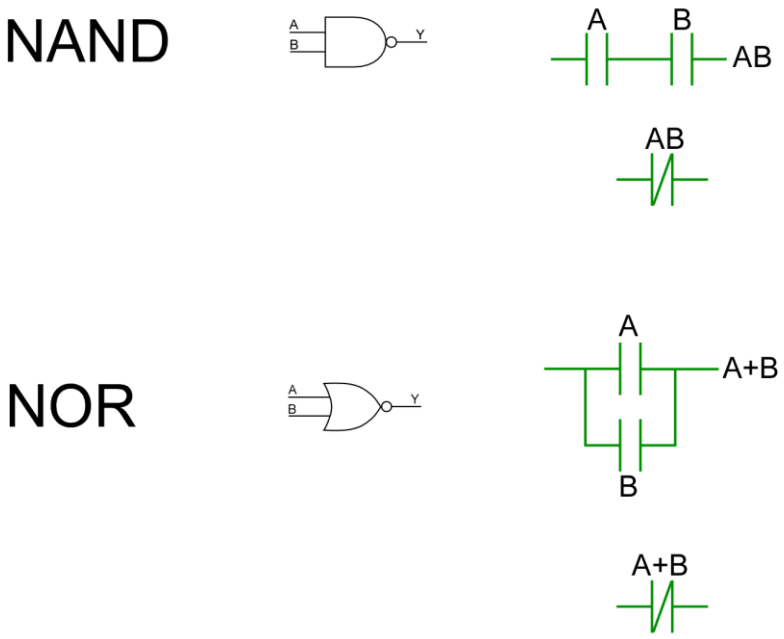




9) **NOT:** The biggest problem comes when someone who is used to electrical diagrams starts to program a PLC. Some of the symbols look identical but have a very different meaning. The most confusing gate to learn is the NOT gate. The NOT looks like a normally closed relay, but it inverts the action of the button.

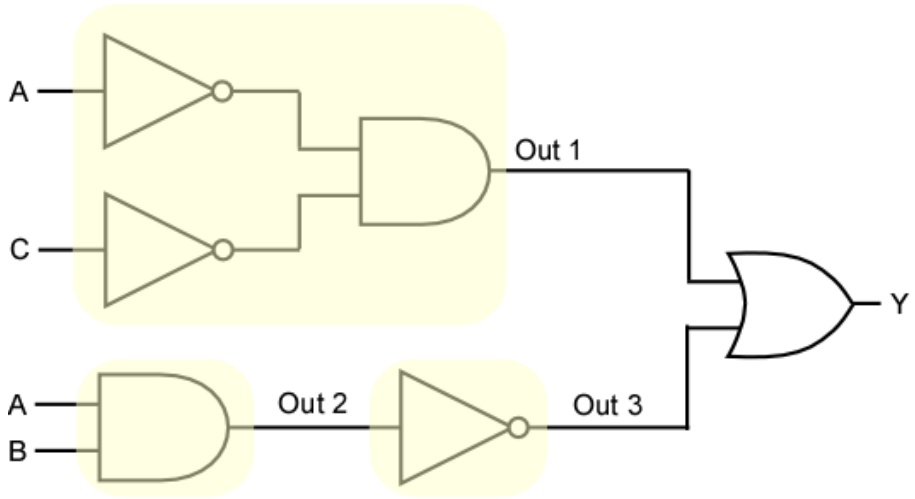


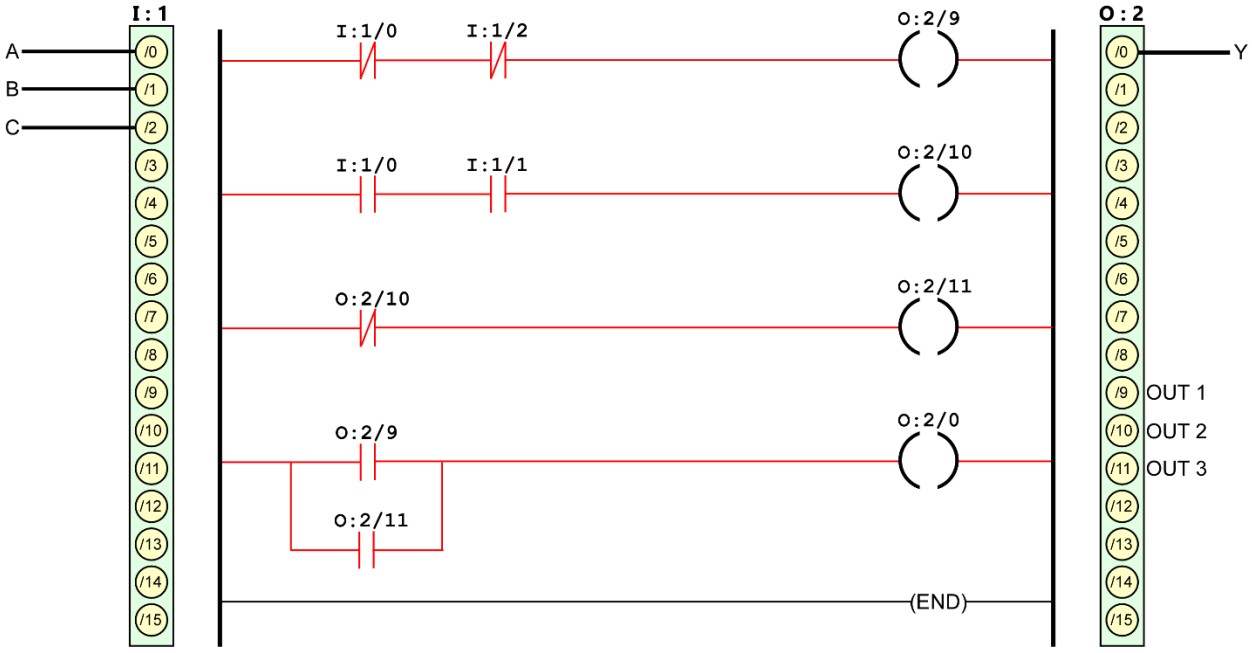
10) **Multiple Rung Logic:** The output of a logic gate can be directly connected to the input of the next logic gate. This can't be done with a PLC. A PLC can turn on or turn off a relay coil Some logic gates, such as the NAND and NOR, and some logical diagrams will need to be implemented with two or more rungs.



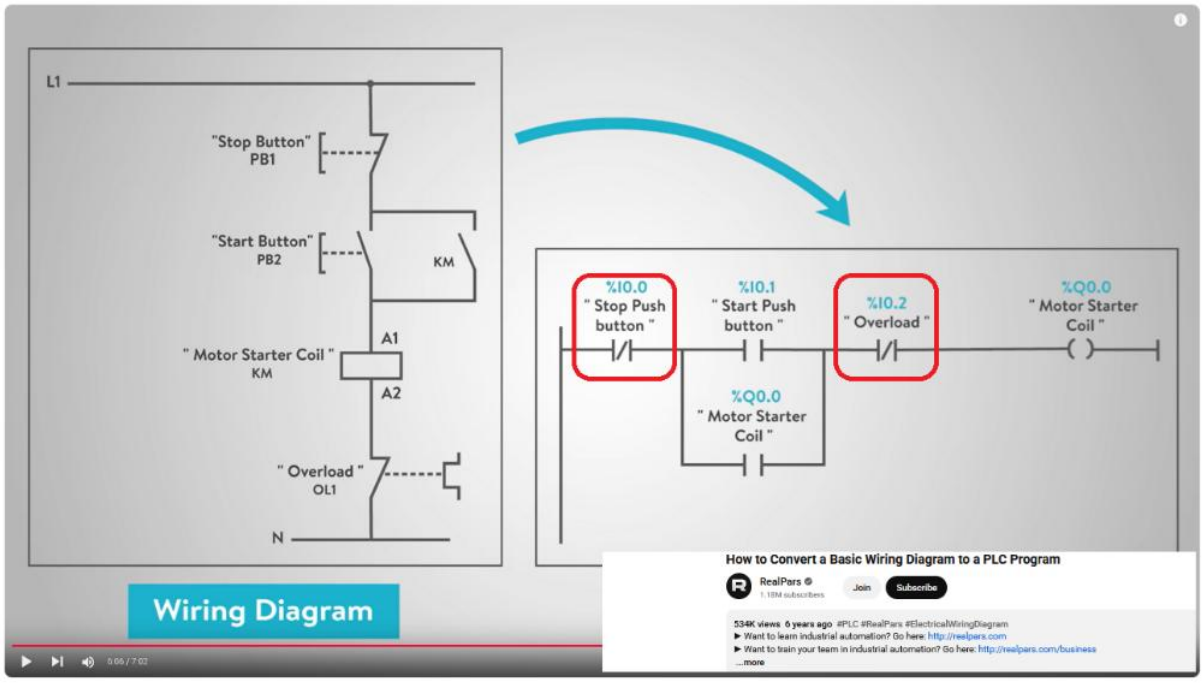
When converting a logic gate into a PLC program your logic may span more than one rung. Every time you complete a gate, you will need to save the result. For now, this will be saved on an output coil. Later, we will cover a better method.

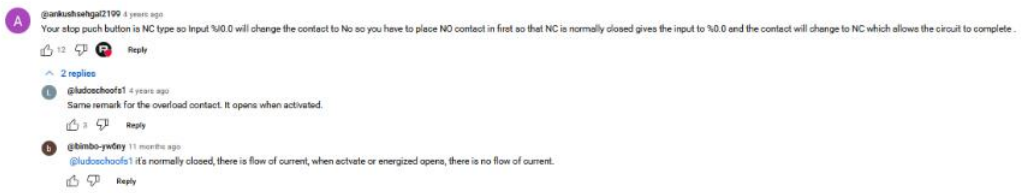
An example is shown below.





11) **<u>On Line Errors:</u>** Some information on the internet is wrong. In fact, this error is the same error I just warned you about. (I added the red squares.)





12) **<u>Simulation:</u>** You can practice your PLC programming at home using the following website:

https://app.plcsimulator.online/

Next, we will simulate the circuit above and see why it is wrong.