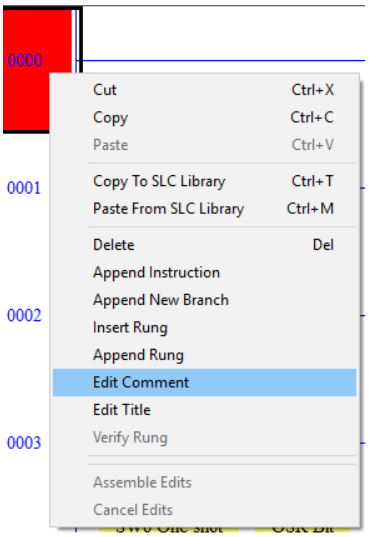
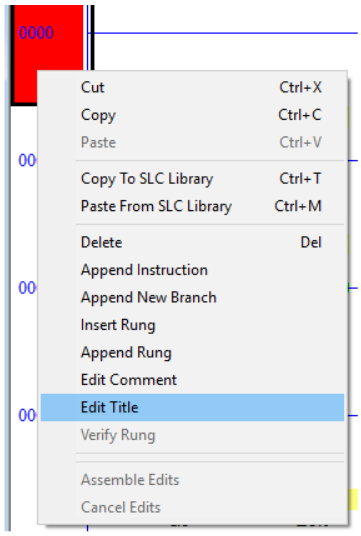
EET165 Lecture #5

1) **Review:** Answer any questions from last week.

2) **Comments:** A comment is a note in a computer program that you can leave yourself or a co-worker. Comments are ignored by the computer and you can put anything you want into them. Normally, the person who writes the code, the date it was started, and a brief description of what the program is doing.

   You can comment a PLC program by highlighting the rung number (when it turns red) and right clicking on the red area you will see a menu show up. You can add either a title or a comment (or both).





   a. **Title:** A title is a single line that is centered. It is used to break the ladder logic into "chunks". These rungs work together to perform a task (such as toggle a motor power on and off).

   b. **Comments:** Comments are multiple lines of text that are NOT centered. This is used to give a better explanation of what is going on in each line or segment.

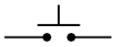
c. **Example:** A code without comments and titles and the same code with comments and titles are both shown below.

3) **Symbols:** We have covered some of the symbols used in a PLC program already. They are broken into two groups, inputs and outputs.

   a. **Inputs:** Some of the input symbols that we have already covered are shown below. These inputs are switches. No matter what kind of switch, they make or break connections.

N.O. Toggle Switch

N.C. Toggle Switch



N.O. Push Button

N.C. Push Button

N.O. Pressure Switch

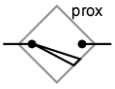
N.C. Pressure Switch

N.O. Float Switch

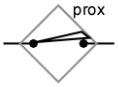
N.C. Float Switch

N.O. Temperature Switch

N.C. Temperature Switch



N.O. Proximity Switch

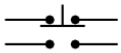


N.C. Proximity Switch

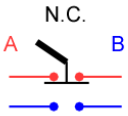
N.O. Limit Switch

N.C. Limit Switch

b. **More inputs:** There are many more types of inputs below are a few more. Just like the other inputs, all of these inputs make and break connections in different ways.



Break and Make Button



Selector Switch

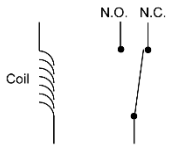
Overload Switch

c. **Sensors:** Sensors are also switches, but the switch is moved but it does not touch anything. Some examples are:

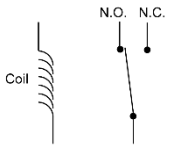
   i. **Proximity:** There are many common types of proximity sensors. Inductive will sense ferrous metals (similar to the way a metal detector works), capacitive disrupts an electric field, ultrasonic will send out a sound wave and it will sense an echo or no echo, and a light sensor will send a light and sense a reflection or the lack of a reflection. Hall effect sensors sense a magnetic field.

   ii. **Other types of sensors:** There are other types, like a beam break sensor. It will have a transmitter and a receiver separated by an air gap. A 1 or 0 is generated when the beam is not broken or broken. Flow sensors have a little propellor that moves when water rushes through a pipe and it generates a signal based on the speed of the flow.

d. **Output coils:** We have also covered a few outputs. These outputs are all types of relays. A relay is just a coil of wire that can have current flow through it. When current flows, the coil becomes an electro-magnet. It will then pull a piece of metal away from its normally closed pin and makes a connection with the normally open pin.



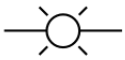
Unenergized



Energized

This is the basis for most of the outputs that are used in controls. A small coil signal can control a normally open and a normally closed contact at the same time.

e. **Output Coils:** We have also covered two output coils, a lamp and a motor contactor.

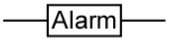


Lamp

Motor

f. **Other output coils:** There are many other types of output coils. But it does not really matter, they all work the same. If power is applied, the device does its job.

Control Relay



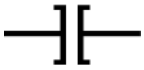
Alarm

Heater

Horn

Solenoid Valve

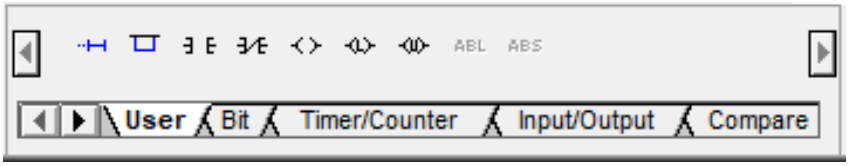
4) **PLC Symbols:** We have covered a few different symbols as shown below.

Symbol	Name	Instruction	Description	
	XIC	Examine if closed	Pass a 1 when closed	Open = pass a 0 Closed = pass a 1
	XIO	Examine if open	Pass a 1 when open	Open = pass a 1 Closed = pass a 0
	OTE	Output Energized	Power device when 1	0 in = No power out 1 in = Power out

Now there are three new symbols that will be introduced.

Symbol	Name	Instruction	Description
	OSR	One Shot Rising	Create a high pulse for 1 program cycle. *(Under the Bit Tab)*
	OTL	Output Latch	Turn on the device or bit and keep it on even if the signal goes away.
	OTU	Output Unlatch	Turn off the device.

Tabs: The One Shot Rising (OSR) is under the Bit tab, so it works on Bits. The Latch (OTL) and Unlatch (OTU) are under the Bit tab and the User tab, so these two commands work on both Bits and Coils.



OSR: One Shot Rising. When the signal feeding the OSR goes from low (0) to high (1), the OSR sets a bit in file B3 to 1, but only for one programming cycle. Even if the input of the OSR remains a 1, it will not send out another 1 until it gets reset. A reset is when it goes low and then high again. This OSR must feed an output, so I just use another bit from file 3. *Note: If you are using https://app.plcsimulator.online/ the button and the OSR symbol are combined into a single symbol, but the OSR is replaced by a **P**. The **P** stands for positive edge trigger.*

OTL: When the OTL receives a 1, it outputs a 1 and it stays that way, even if the input goes to zero. The only way to clear the output or bit is using the associated OTU command. *Note: If you are using https://app.plcsimulator.online/ the OTL symbol is the same, but the **L** is replaced by an **S,** where the **S** stands for Set.*

OTU: When the OTU receives a 1, it outputs a 0 and it stays that way, even if the input goes to zero. The only way to set the output or bit is using the associated OTL command. *Note: If you are using https://app.plcsimulator.online/ the OTU symbol is the same, but the **U** is replaced with an **R**, where the **R** stands for Reset.*

Power Loss: If an OTL /OTU loses power and then regains power, it will power up in the last state it was in. So, if it is latched on and it loses power; it will power up latched. If it is unlatched and it loses power; it will power up unlatched.

Last one wins: The second rung determines the state when both inputs are true. If the OTL and the OTU both receive a 1, the second rung erases the work done by the first rung.

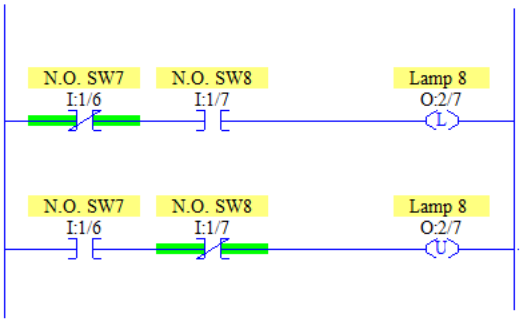
5) **<u>Seal – In Examples:</u>** Below is an example of how to use the Latch and Unlatch symbols to create a seal-in circuit. The same bit is the output of two rungs. That looks like it broke one of the rung rules, but it doesn't. Think of this as an output with input pins. One pin (L) is connected to the first rung and the second pin (U) is connected to the second rung.

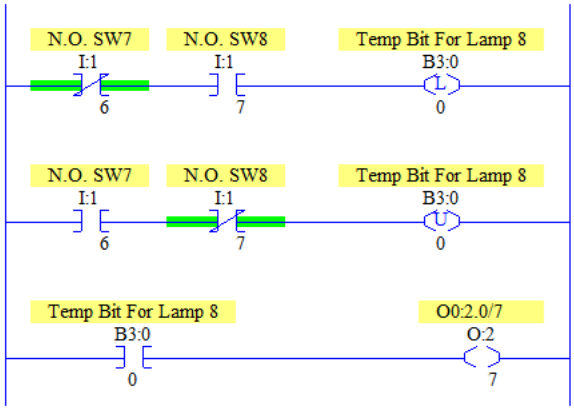
   The Latch and Unlatch can be used on an output coil or a bit. In the image below, the left side shows the Latch and Unlatch driving an output coil. The right side shows the Latch and Unlatch driving an internal bit.

   It doesn't matter if the latch is driving a coil or bit, they work the same. The When the Latch receives a 1 it is latched on. It stays on even if the input signal goes back to 0. The only way it gets reset to 0 is if the Unlatch for the bit goes high.

   <u>Because the rungs are checked from left to right and from top to bottom this means this bit can receive an L and a U in the same scan. If both L and U receive a 1, the second rung determines the state it stays in.</u>

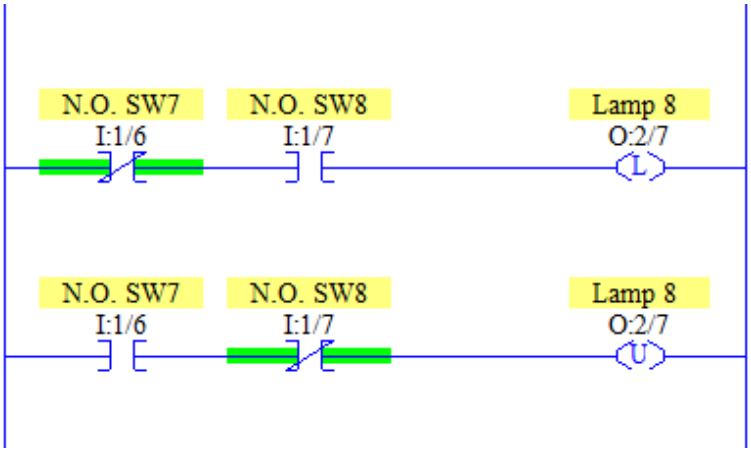
   In this example, assume both buttons are pressed at the same time. The first run will latch the bit and make it a 1. But then the second rung will unlatch the bit making it a 0. Because the Unlatch is the last rung and that is the state is stays in. So, if both buttons are pressed at the same time, the bit is unlatched (set to 0) because that is the second rung.



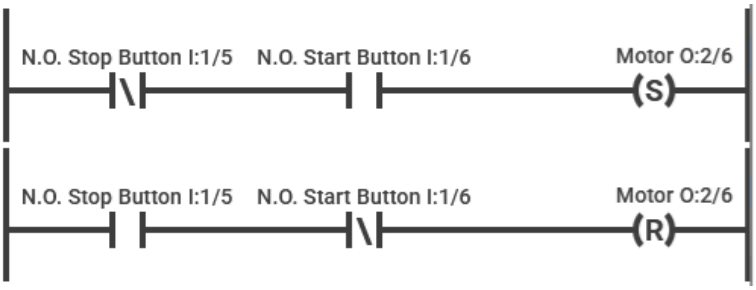


I personally don't like it when both L and U can be on at the same time and the PLC program determines what happens based on the order of the rungs.

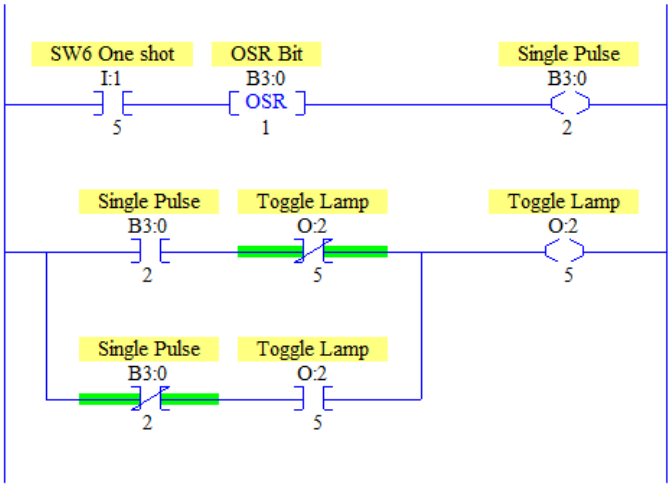
I prefer to lock out the circuit using both buttons. In this example, both SW7 and SW8 are in both rungs but with opposite logic. Therefore, if SW8 is pressed, it activates the top rung, and locks out the bottom rung. If SW7 is pressed, it activates the bottom rung, and locks out the top rung. When both buttons are pressed at the same time, they each lock out the other rung. This configuration makes it impossible for L and U to both be active at the same time. If both buttons are pressed, the bit stays in the last valid state.



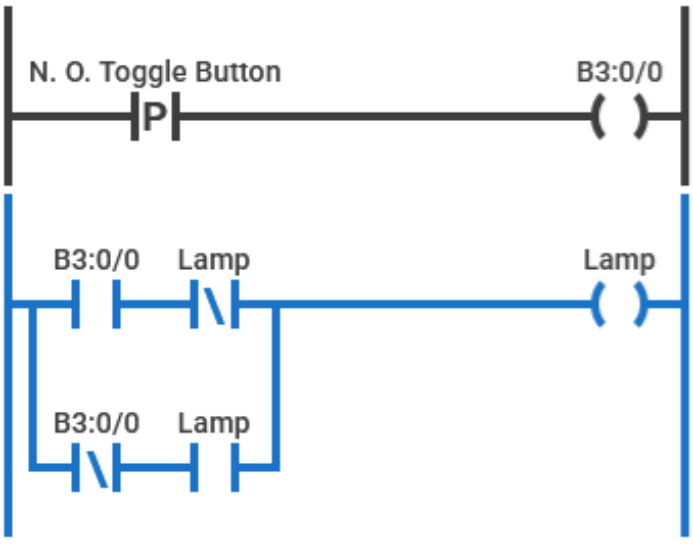
This screen capture was from *https://app.plcsimulator.online/* so the OTL and OTU commands have an S (instead of L) and an R (instead of U) in them – but they work the same way. This website will let you practice programming and test your code at home. As you can see, it may have a few small differences, but it still very useful.



6) **Toggle Example:** Below is an example of how to use the OTR command to make a button toggle an output on and off. When SW6 is pressed, the OSR receives a high signal, this makes bit B3:0/1 go high for one program cycle then go low. B3:0/2 echoes B3:0/1. When the "Signal Pulse" goes high; the upper path in the second rung is activated (and the lower run is de-activated). If the lamp is off it gets turned on – if the lamp is on it gets turned off. Then Signal Pulse goes low and drops to the lower path. If the lamp is on, it creates a seal-in circuit. Without the OSR, the lamp would start to flicker on and off with every cycle.



These screen captures were from *https://app.plcsimulator.online/* so the OTL and OTU commands have an S (instead of L) and an R (instead of U) in them. Also, the simulator combines SW6 and the OSR into a single symbol. But the simulation works the same.



7) **<u>Multiple Outputs:</u>** Multiple outputs can be turned on and off at the same time. This is done by stacking the outputs.

   a. When inputs are stacked it is an OR, so only one needs to be true for the output to be true.

   b. When inputs are in series, it is an AND, so all of them need to be true for the output to be true.

   c. However, outputs are different. If you want all of the outputs to be latched or unlatched you need to stack them (similar to an input OR).

Think of the rung as a path for water. The inputs are controlled by an outside source. These are valves that turn on and turn off. That is why stacked (parallel) inputs are an or, because only one needs to be true for the water to get through that barrier. Inputs in series are an and, because they all need to be on to let the water get through that barrier.

Once the water passes all the inputs, it reaches the outputs. All of the outputs need to have access to the water source, so for all of the outputs to get power they have to be stacked (in parallel). An example is shown below.