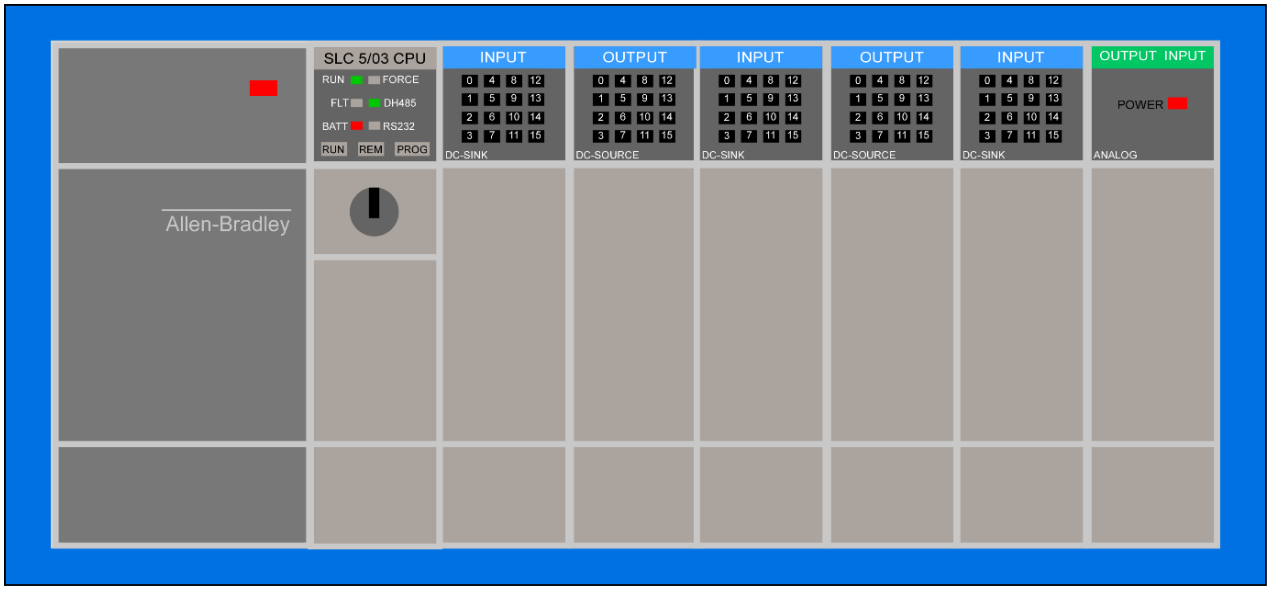
EET165 Lecture #4

1) **Review:** Review logic if needed.

2) **Hardware:** The PLC parts are attached to a backplane. The backplane is attached to the case using a DIN rail. Below is a diagram of the PLC used in this class.

   On the far left is the power supply. Next is the CPU, and it is in slot 0. The CPU is always in slot 0. Next there are 6 slots (numbered 1-6) that can hold any of the IO cards. In the lab configuration, slot 1 holds an input card that is attached to the switches on the panel. Slot 2 is an output card that is connected to the lamps on the same switch panel. Slot 3 is an input card that connects to the BCD rotary switches on the right panel and slot 4 is an output card that connects to the 7 segment displays above the rotary switches.



   The cards in this lab have 16 inputs (or 16 outputs) that are numbered 0 – 15. The cards are connected to the field devices.

   Input cards are connected to field devices such as buttons, switches, sensors, limit switches, and proximity sensors. The input cards convert the voltages into an on/off digital signal that the PLC can use.

   Output cards are connected to field devices such as motor contactors, relay coils, lamps, alarm sirens, and pilot lights. The output cards convert the digital on/off digital signal into voltages that the field device can use.

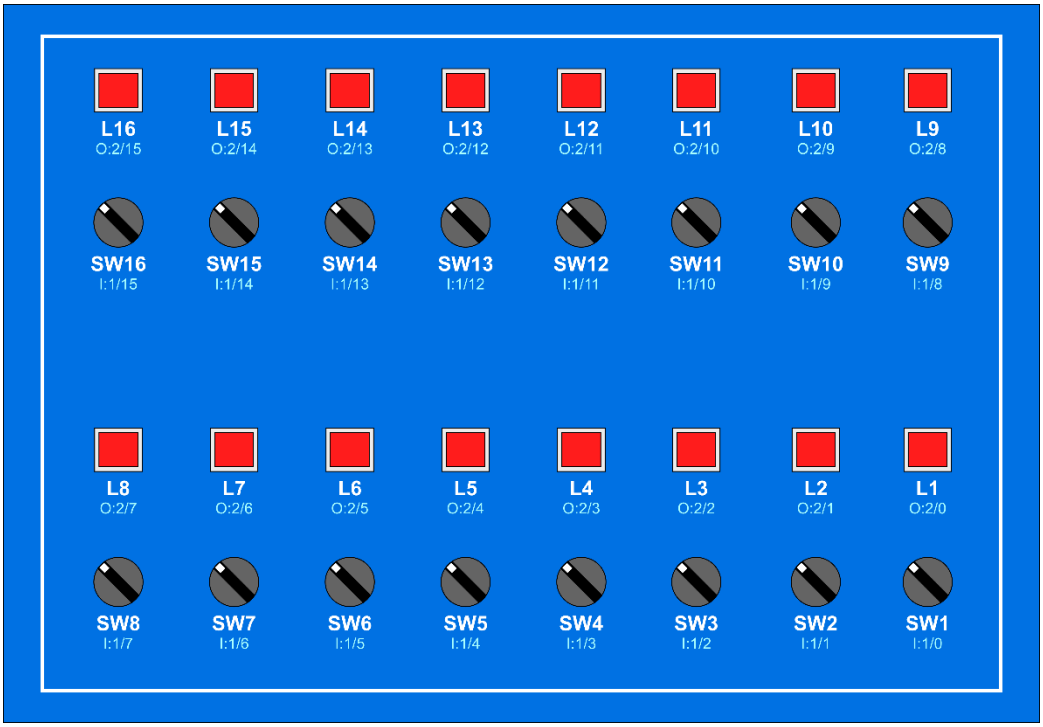
To access the field device, the program needs to address the pin on the card that the device is attached to. The command starts with I or O to tell the PLC if it is accessing an input or output card. Then there is a colon, followed by the slot number where the card is attached. There may be many different input or output cards, this number is how the specific card is selected. Lastly, there is a forward slash followed by the pin number. For example:

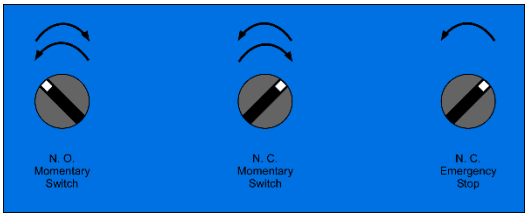
To access a switch on input card 1, on pin 3, the command would be: **I:1/3**

To access a pilot light on output card 2, on pin 7, the command would be: **O:2/7**

To access a proximity switch on input card 3, on pin 0, the command would be: **I:3/0**

3) **<u>Control Panel:</u>** Below is a drawing of the control panel with the card and pin number is listed below each field device. For example, Lamp 12 (L12) is connected to **O:2/11**.





The switches are toggle switches. In order to simulate a momentary switch, you will need to flip the switch on and then off (or off and then on). A normally closed switch will be in the on position and switching it to off would simulate a button press.

4) **Files:** The PLC program is broken into two file types, program files and data files.

   a. **Program Files:** Program files have 256 files broken in to four parts.

      i. **System functions (File 0):** These files include all the parts the system needs to run the program. It includes information such as the processor type, how the I/O is configured, the processor type, and the password.

      ii. **Reserved (File 1):** This is a file that is used by the system. You do not have access to this file.

      iii. **Main Ladder Program (File 2):** This is the main ladder logic program that you write.

      iv. **Subroutines (Files 3-255):** These are where subroutines you write are stored.

   b. **Data Files:** There are also 256 files broken into nine parts.

      i. **Output (File 0):** This file stores the states (on/off) of the field devices that are connected to the output cards. Start with O: and after the colon is the slot the card is in. Finally, there is a slash and a number (0-15) for the output number. For example: **O:1/0**.

      ii. **Input (File 1):** This file stores the state (on/off) of the field devices that are connected to the input cards. Start with I: and after the colon is the slot the card is in. Finally, there is a slash and a number (0-15) for the input number. For example: **I:2/15**.

      iii. **Status (File 2):** This file stores the status of the CPU and is used for debugging.

      iv. **Internal Bits (File 3):** This file holds bits that can be used to temporarily hold a bit. If you need to hold a bit, you could use an output relay but it would use up a valuable pin. These bits are not connected to a pin. To access a bit you would use **B3:0/0**. B3 replaces the I or O and is followed by the colon. Next is a row number in the data file (0-255). Finally, a slash and the column number of the bit (0-15).

v. **Timers (File 4):** This file holds the information used to control timers.

vi. **Counter (File 5):** This file holds the information used to control counters.

vii. **Control (File 6):** This file holds the information used to control many different commands.

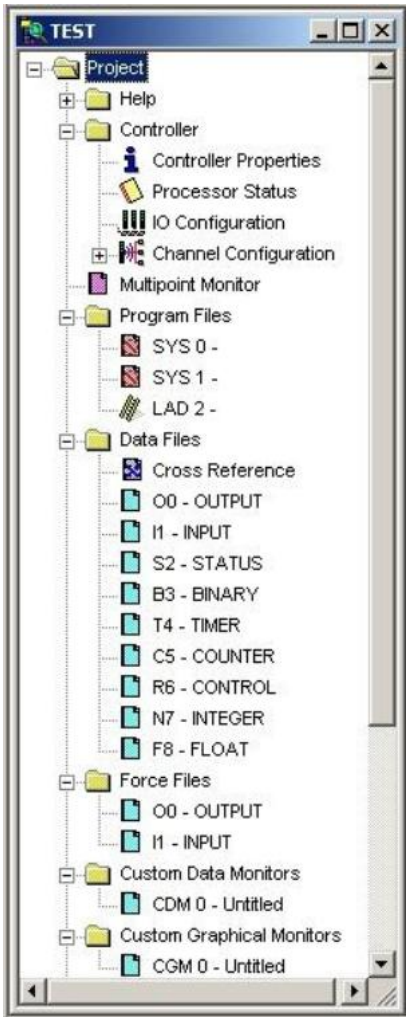
viii. **Integers (File 7):** This file holds integer variable values.

ix. **Float (File 8):** This file holds float variable values.

x. **File 9 through File 255:** These are used for user defined files. These can be configured to hold bits, integers, counters, etc.

c. These files are listed on the left side of the screen. If you double click on the file, it will bring up the file and you can view what is in it.

*NOTE: The output file is really **O0:** and the input file is really **I1:**, however on these two files you do not need the 0 and 1. You **do** need the number on the remaining files.*



5) **Different programming methods:** There are a few different ways you can program a PLC. They fall into two categories, graphical and text based. Some of the different ways are listed below.

   a. **Graphical Methods:** These programming methods use images to program the PLC.

      i. **Ladder Diagram:** This is what we are using in this class. The program is made up of pictures that look like an engineering diagram.

      ii. **Functional Block Diagram:** This uses interconnecting functional blocks to program the PLC. The blocks have inputs and outputs with a description of what the block does display on it. These are similar to code made out of Legos.

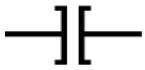
      iii. **Sequential Functional Chart:** This graphical programming method makes a program that looks like a flow chart.

   b. **Text Methods:** There are also programming methods that use text to program the PLC.

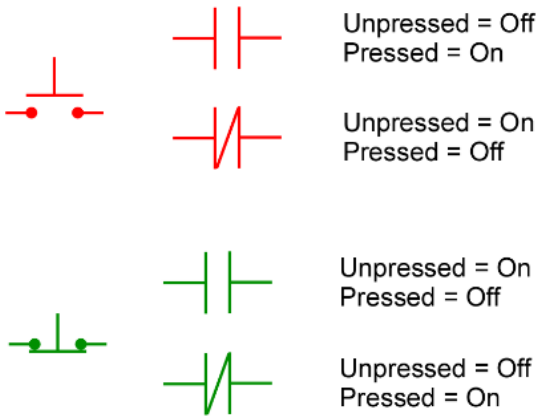
      i. **Instructions List:** Structured text gives you the commands that you stack and fill in the required information.

      ii. **Structured Text:** Structured text looks like a programming language with variables, ifs, and loops.

6) **Symbols:** When using ladder logic, there are three PLC program symbol we will be using for now. As time goes on, we will introduce more.

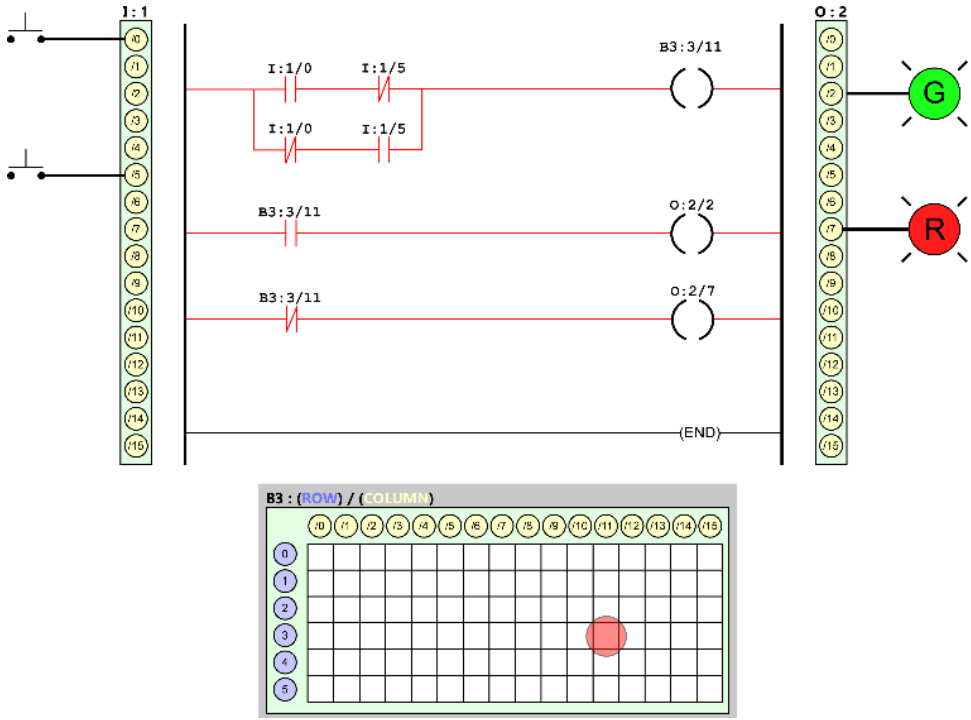
Symbol	Name	Instruction	Task	Description
	XIC	Examine if closed	Excite if Closed	Open = pass a 0 Closed = pass a 1
	XIO	Examine if open	Excite if Open	Open = pass a 1 Closed = pass a 0
	OTE	Output Energized	Output Energy	0 = No power 1 = Power

An example of how the XIC and XIO react to different push buttons is shown below.



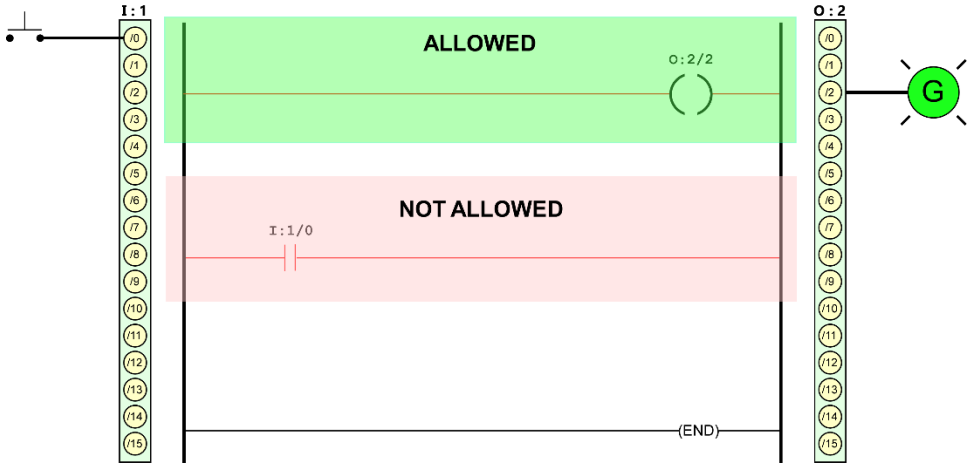
7) **<u>Rungs:</u>** Rung are used to create the program. Each run will look at the inputs and decide if an output is high or low. The runs will use the XIC, XIO, and OTE (for now) to determine if it should output a 1 or 0. These symbols will be associated with the parts attached to the input file (File 1) and the output file (File 0). Below is a drawing of the rungs of a PLC program along with file 0, 1, and 3.

   Below there are two momentary buttons connected to screw 0 and screw 5. This is the implementation of an exclusive or (XOR). Before, we used the output pins to hold the logical operations. This is not a good idea. Now we will save the logical output to binary location 3/11. This bit can then be used to turn on the green and red lights connected to output screw 2 and 7.



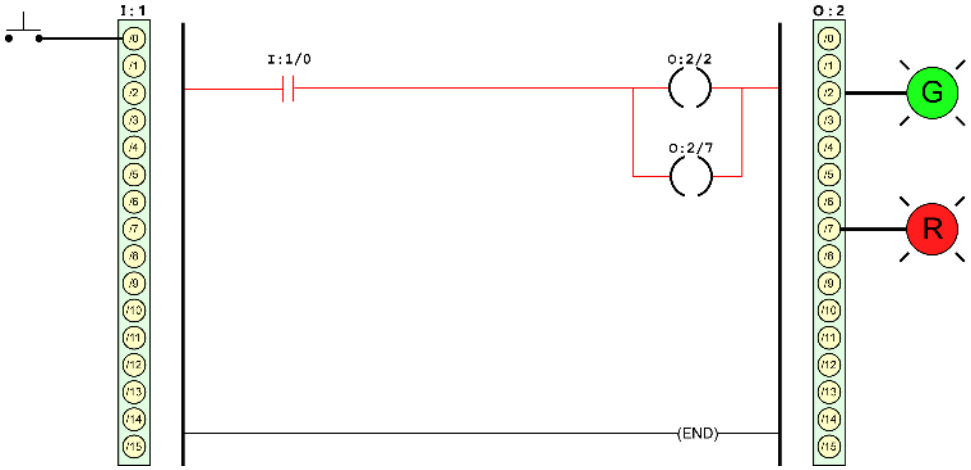
8) **Rung rules:** you should follow these rules when creating ladder logic.

a. **No inputs:** You can have just an output lamp, without an input. This is used for a power button. But, you can not have an input without an output.

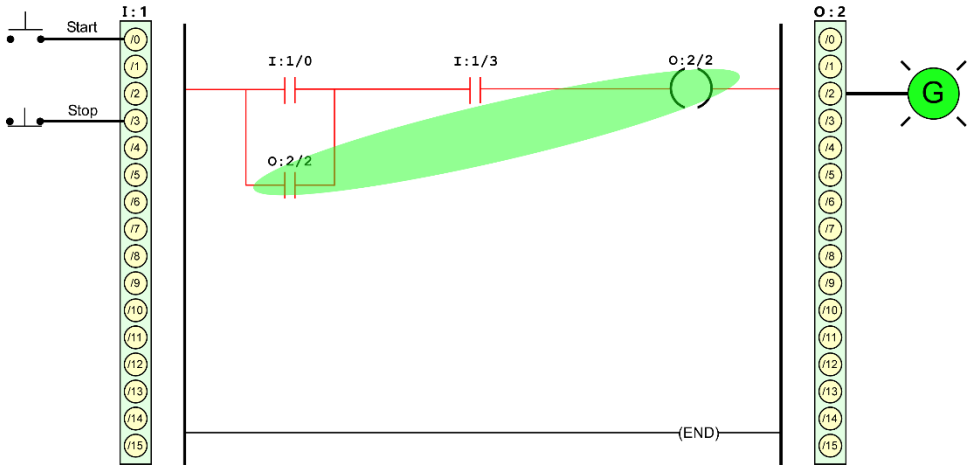


b. **Two rungs should NEVER control the same output:** The rule explains it all. You can have a very complex rung, but you should not have more than one rung control the same output. If you do, the outputs are unpredictable.

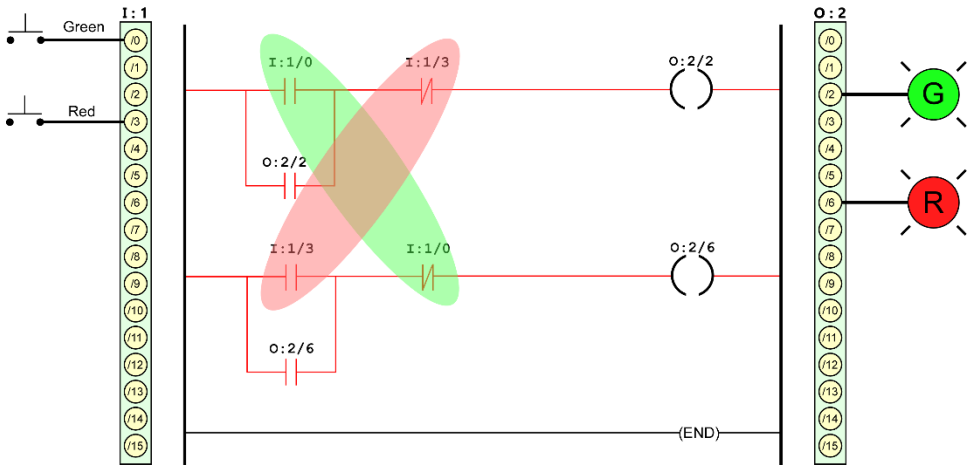
c. **One rung can control multiple outputs:** As shown below, you can drive as many outputs as you wish. This is not an "OR" statement. The parallel inputs are an OR. But parallel outputs means that all of them will be executed.



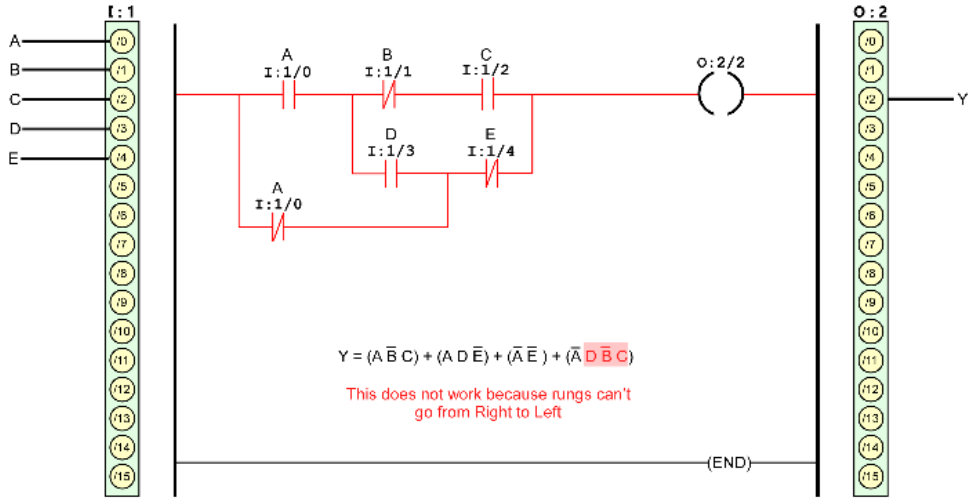
d. **Seal-In Circuit:** A seal in circuit is used to keep a motor running when it is activated by a momentary switch. When the push button is pressed, the motor turns on. This will activate the path around the push button.

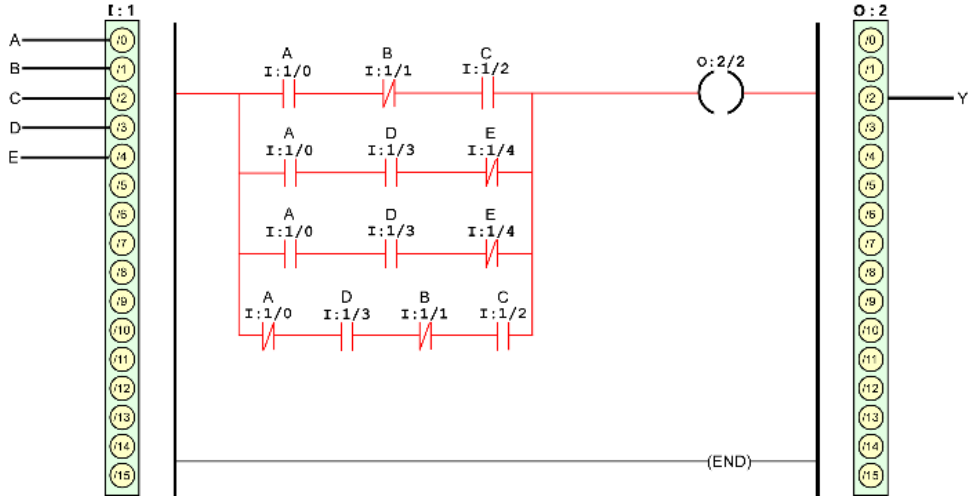


e. **Latching Circuit:** A latching circuit is the opposite of a seal-in circuit. It will disable another part of the circuit. When the green button is pressed, it turns off the red bulb. When the red button is pressed it disables the Green button.



f. **<u>Logic only moves from Left to Right:</u>** The logic in a rung runs from left to right, it will never "backtrack" and check from right to left. Sometimes people start to get fancy with their rungs and they assume if they can draw a line from one rail to the other, then it is a valid path. But in the example below A'DB'C is a path from left to right. But because there is a right to left logical comparison – it will not work. Below it is one possible solution that will work.





9) **<u>Execution Sequence:</u>** The program is executed in the following order.

a. Read in all of the field devices and copy their states into Input File 1.

b. Evaluate the rungs. The rungs are evaluated from left to right, and top down. This will update Output File 0 as the rungs are updated.

c. When the rungs are all evaluated and the Output File is updated, the Output File will update the output field devices.

d. There is a clean-up section that is used for network communication (also called housekeeping).