Connected Component Workbench

Connected Component Workbench is the software used to program Allen Bradley Mico-810 PLC's. Copies can be obtained fom the following link:

http://www.rockwellautomation.com/global/products-technologies/connected-components/tools/workbench.page

Note that if you are using a Windows XP machine, you need an older version from a CD ROM. Please see the instructor if this is you.

Notation

Inputs:

- Connect COM-0 to -DC24 (0V) to enable inputs 0..3
- Connect -DC24 to -DC24 to enable inputs 4..7
- Inputs 0..3 are digital (0V = logic 0, 24V = logic 1)
- Inputs 4..7 and digital (24V) or analog (0..10V), depending upon the input voltage

Constant Inputs:

Time:

- T#300x
- d = day
- h = hour
- m = minutes
- s = seconds
- ms = milliseconds

Binary: 2#

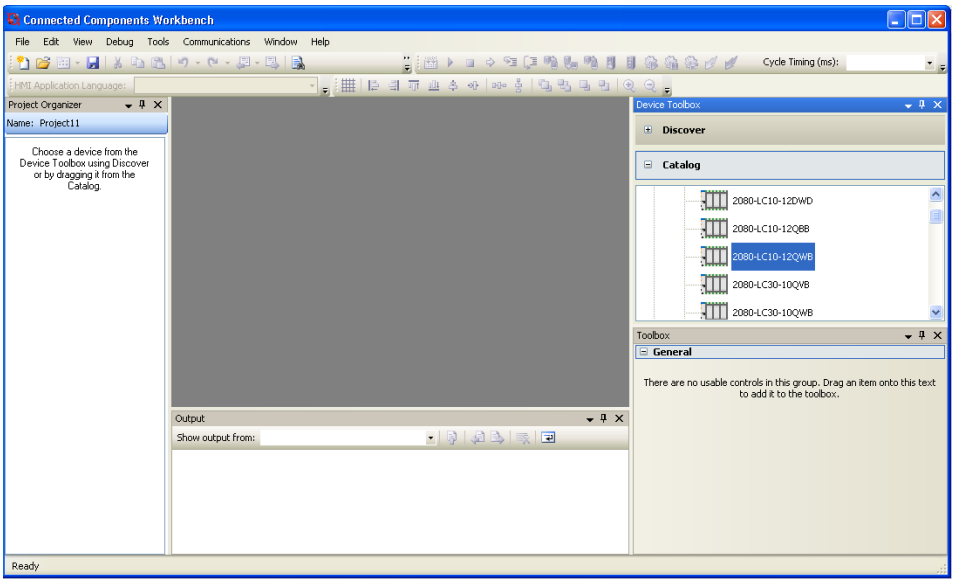
- 2#1001_0010 binary number 0x52

Hex: 16#

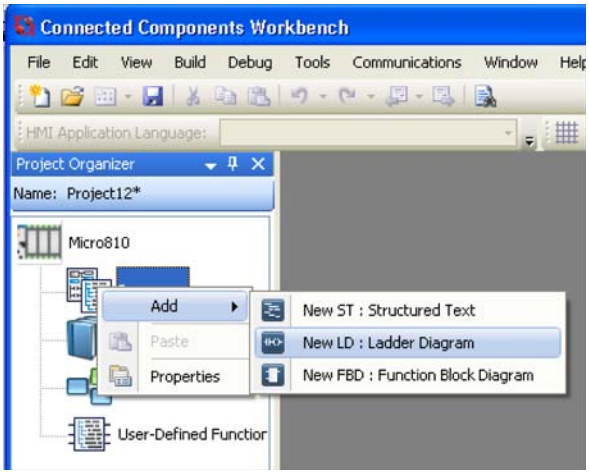
Decimal: 10#

Getting Started:

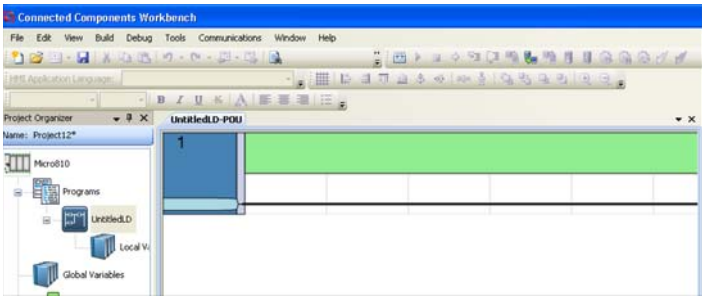
Start Connected Components Wizard. You should get the following screen



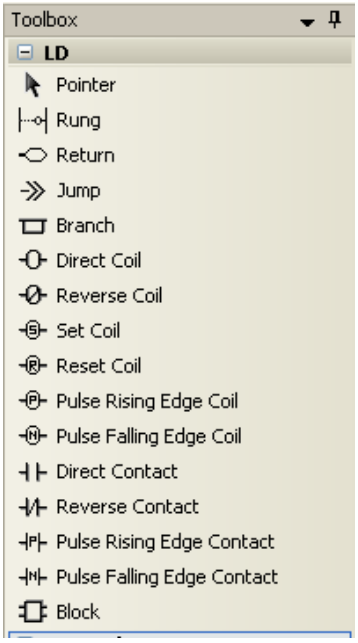
Select the PLC we're using: 2080-LC10-12QWB, and draag it left to the Program window. Double click on Programs and select New Ladder Diagram:



This should give you the following display: an empty program



Program elements are shown in the lower right corner: these are drag and drop elements:



- Rung: Add a new line to the program

Outputs: These must be placed on the far right of the ladder diagram

- Direct Coil: Boolean variable. When energised, the variable becomes true. If the variable is an output relay, the relay closes. When de-energised, the coil opens. Can also be an internal state.
- Reverse Coil: Booleam vairable. Energised is open (true), de-energised is closed (false).
- Set Coil: When energised, the coil closes and remains closed.
- Reset Coil: When energised, the coil opens and remains open.
- Pulse Rising Edge Coil: The coil closes momentarilly when the output is energised (rising edge)
- Pulse Falling Edge Coil: The coil opens momentarilly when the output is de-energised (falling edge)

Inputs: These must be placed on the left side of the ladder diagram

- Direct Contact: True means the switch closes, false means the switch is open.
- Reverse Contact: Closed when false, open when true.
- Pulse Rising Edge Contact: Momentarilly closes when the signal goes high (rising edge)
- Pules Falling Edge Contact: Momentarilly closes when the signal goes low (falling edge)

Blocks:

- 100+ operations, such as a time delay. More on this later.

Example Code:

Write a prorgram so that the red LED turns on whenever you press button 0.

Function:

Y = A

Inputs:

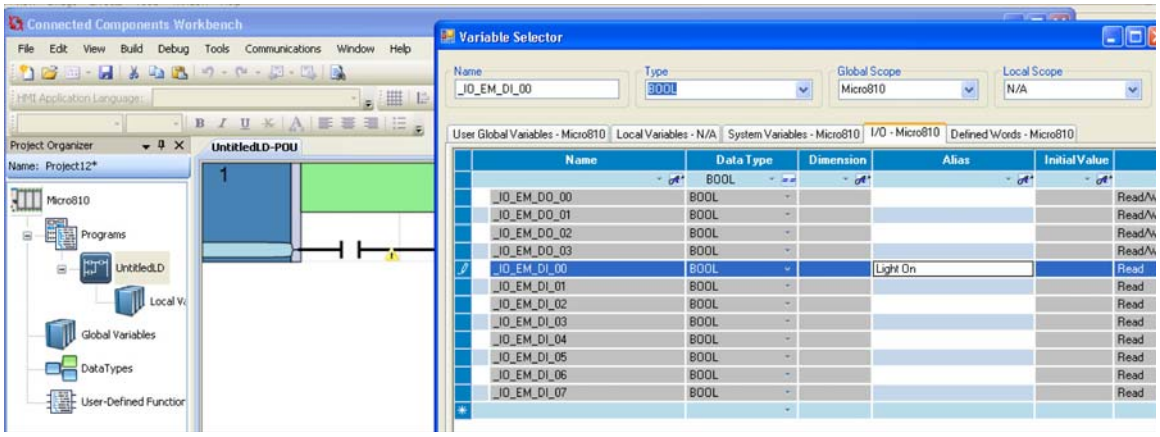
A = IN0

Output:

Y = OUT0

Program:

Click on the Input Coil icon and drag it to the left of the ladder diagram. To define this input as IN0, double click on the coil and select I/O Micro 810

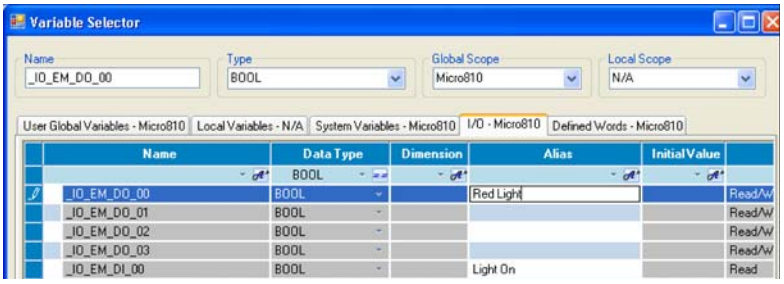


This lets you assign a coil to the input pins and output relays:

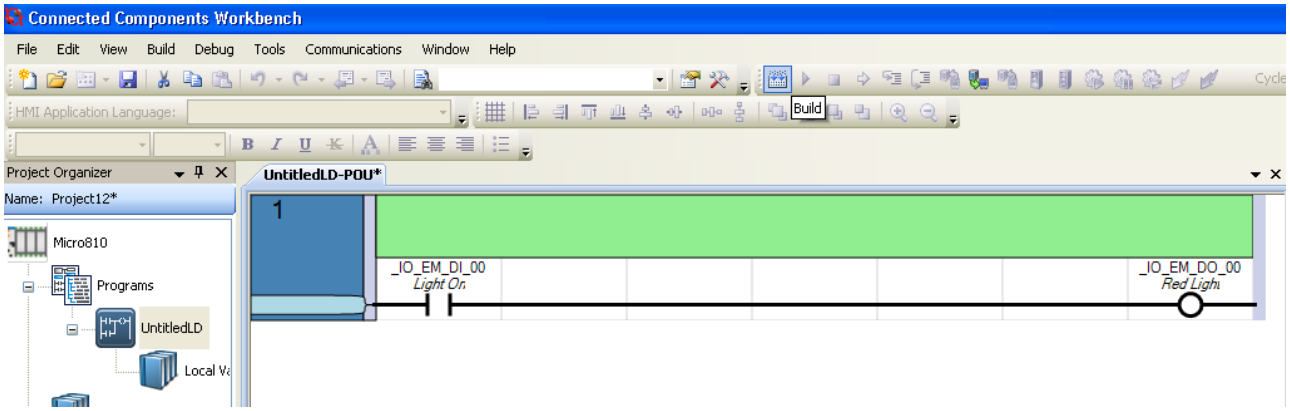
- _IO_EM_DO_0x Digital Output #x (relay output x)
- _IO_EM_DI_0x Digital Input #x (input pin x)

Select Digital Input 0 for pin #0. You can give it an Alian (name) if you like.

Click on the Output Coil icon and drag it to the right side of the display and assign it to Digital Output 0.



Your program should then look like the following:

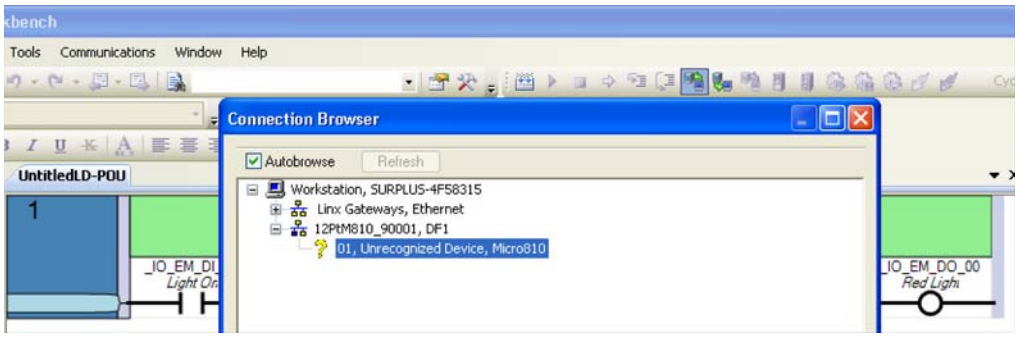


Compilation:

Click on the Build icon. This translates the ladder diagram to machine code. With any luck. you'll get a successful message:

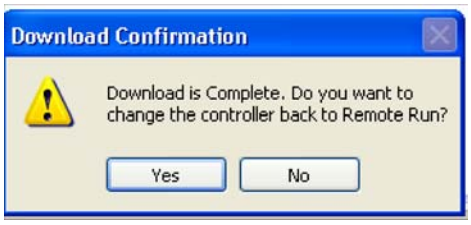
Program:

Connect the Micro-810 to a PC through the USB port. When you click on the Download icon, you'll get a message asking which device. Select the USB port and Micro810



It will then ask if you want to halt the current program:

say yes. About 30 seconds later, after the program is downloaded, it will ask if you want to restart the (new) program. Say yes.



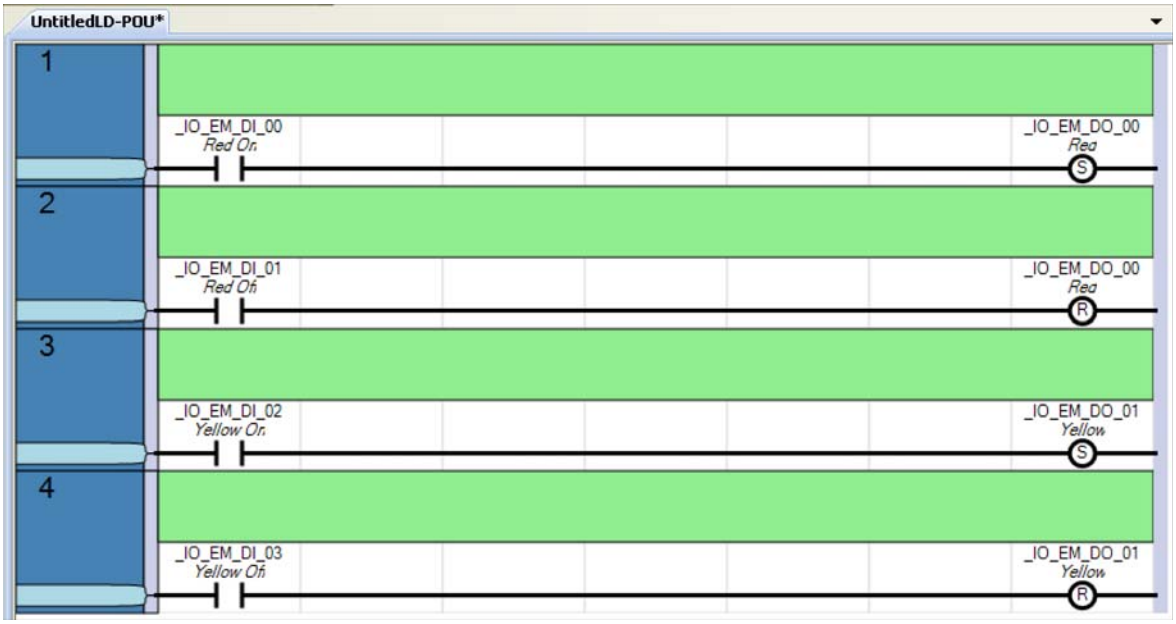
At this point, the program should be running: when you press button 0, the red light turns on.

Example 2: Two On-Off Switches

Write a program with the following functionality:

- IN0: Turn on the red LED (out 0)
- IN1: Turn off the red LED
- IN2: Turn on the yellow LED (out 1)
- IN3: Turn off the yellow LED

Create four rungs by dragging the rung icon left. Add four input coils and four output coils as follows:



Example 3: Logical AND and OR

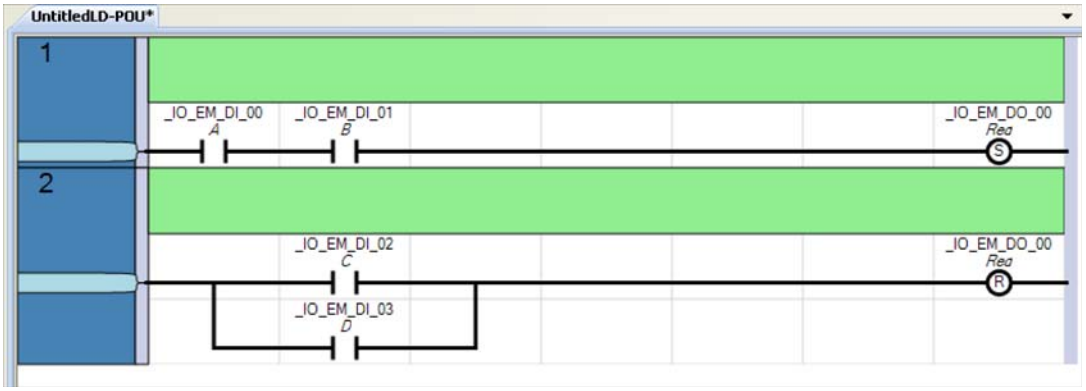
Write a program which

- Turns on the red LED when button 0 and 1 are pressed, and
- Turns on the yellow LED when buttons 2 or 3 are pressed.

Program: To create the OR function, drag the Branch inco left and place in in the second rung. Add to Direct Contacts for input 2 and 3.

Note that

- Two switches in series is an AND function
- Two switches in parallel is an OR function



Combinational Logic on a PLC

From ECE 275, if you can implement AND, OR, and NOT functions, you can implement any truth table. One way to do this is using Karnough maps.

Example: Write a program which

- Turns off all LEDs when no buttons are pressed
- Turns on the blue LED (DO3) when one or more buttons are pushed
- Turns on the green LED (DO2) when two or more buttons are pressed, and
- Turns on the yellow LED (DO1) when three or more buttons are pressed

Assume only buttons 0 / 1 / 2 are used and defined as

- A = input 0
- B = input 1
- C = input 2

First, set up a Karnough map for each output:

A \ BC	00	01	11	10
0	0	1	1	1
1	1	1	1	1

Blue: 1 or more

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Green: 2 or more

A \ BC	00	01	11	10
0	0	0	0	0
1	0	0	1	0

Yellow: 3 or more

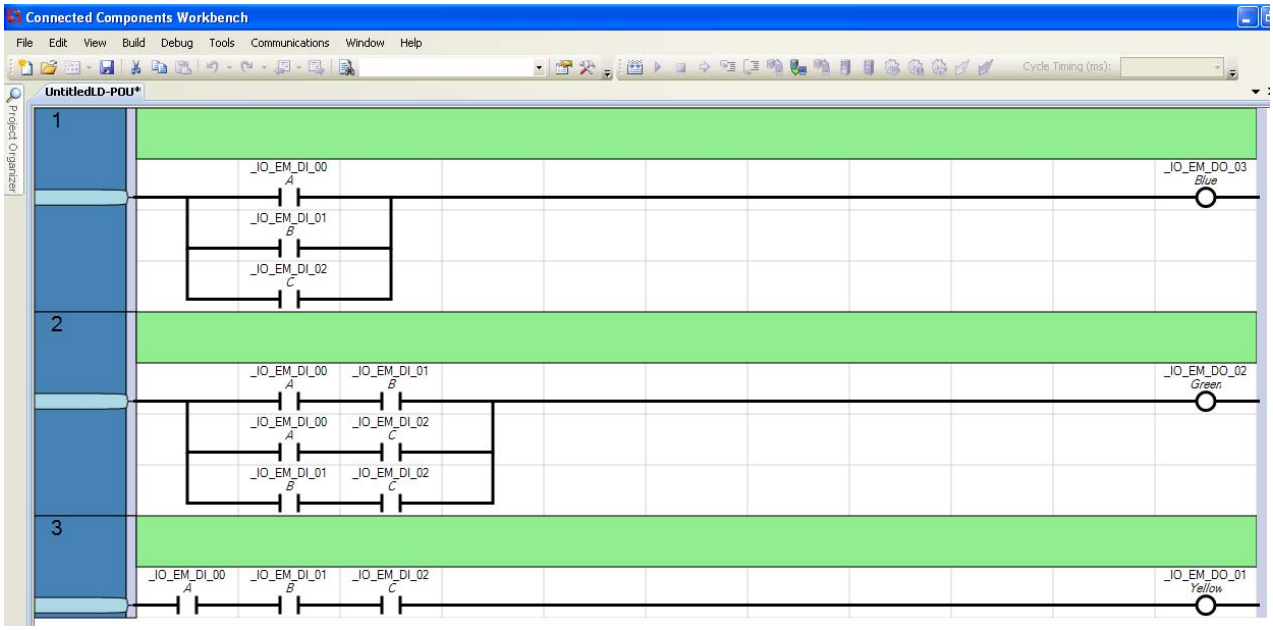
Next, circle the ones in groups of 2n (shown in red above). This gives

$$Blue = A + B + C$$

$$Green = AB + AC + BC$$

$$Yellow = ABC$$

Implement this in Ladder Logic. In Connected Component Workbench, the code will look like the following:



Note that there are other solutions. For example, you could create a local variable for when one, two, or three buttons are pressed - with the following logic for each:

$$One = A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C$$

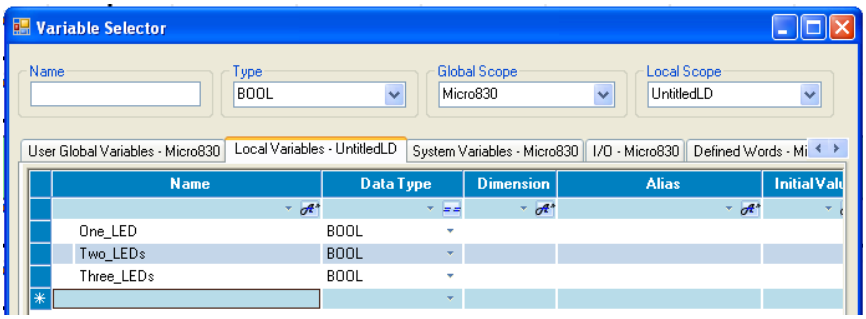
$$Two = AB\overline{C} + A\overline{B}C + \overline{A}BC$$

$$Three = ABC$$

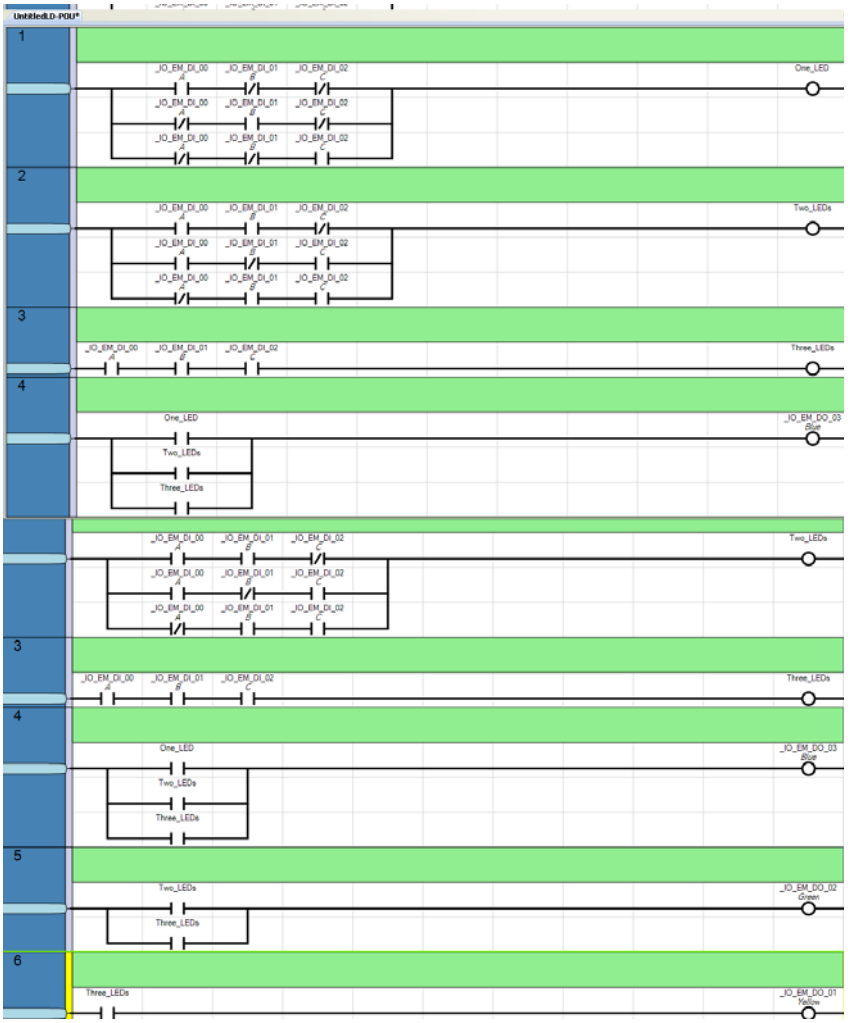
- The blue LED (one or more) is on if any of these three are true

- The green LED (two or more) is on if Two or Three is true
- The yellow LED is on if three is true

Create a local variable

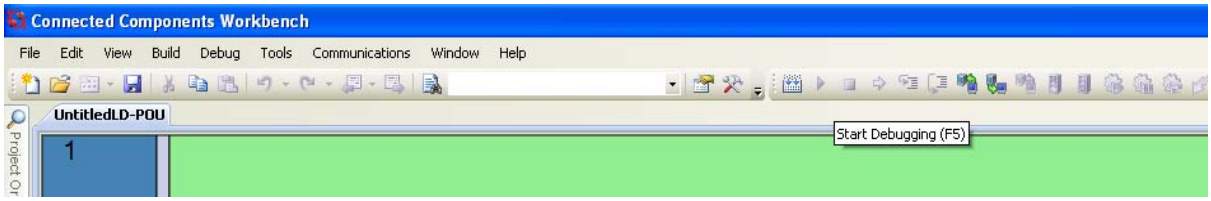


Then use this local variable along with logic for one, two, or three buttons being pressed



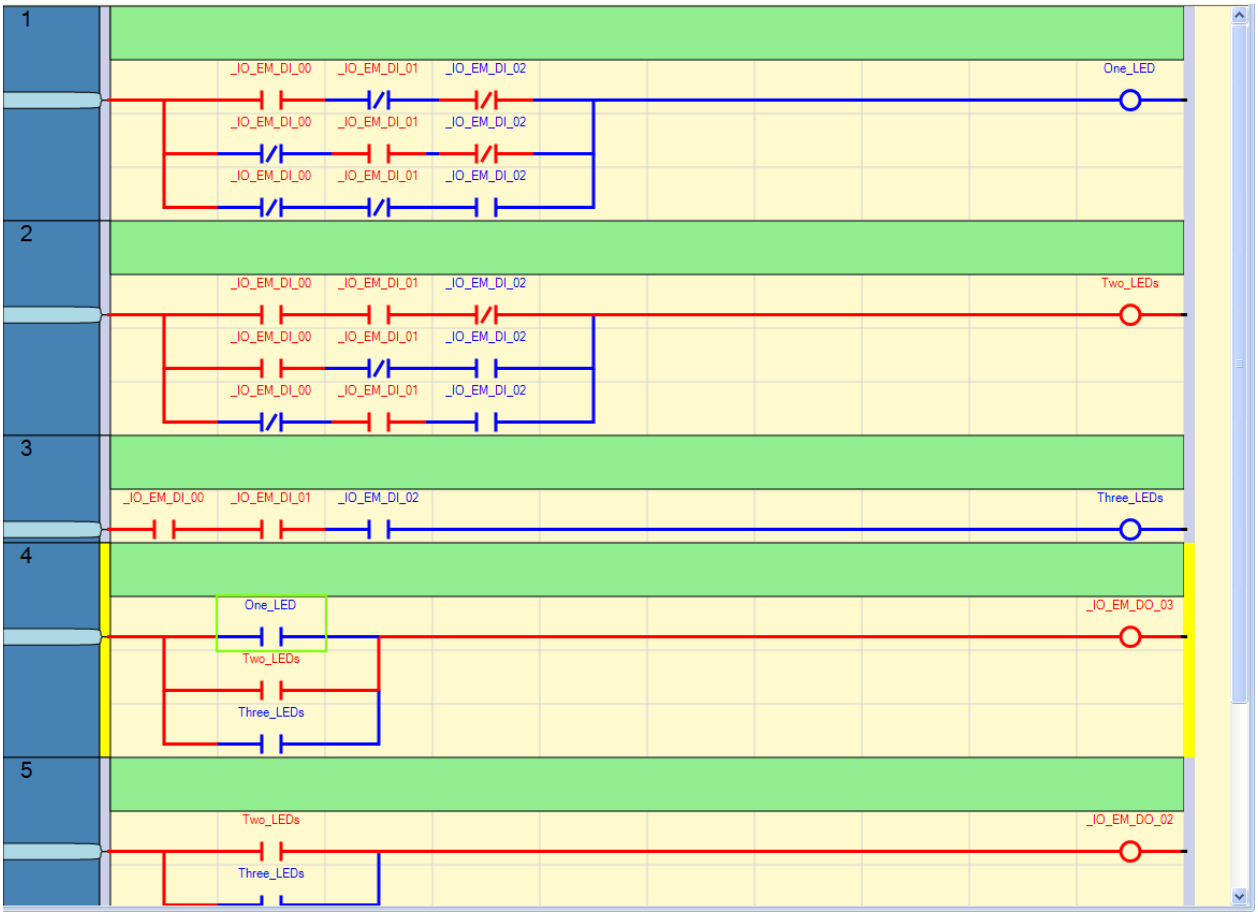
Debug Feature:

Once you download your code, click the Debug button (F5)



This will then show you on the screen which lines of code are turned on as you press the buttons. It helps understand what is going on (and we'll use this feature a lot).

For example, when you press A and B, the display under Debug mode looks like the following:



Red shows which paths are turned on (true)