OBJECTIVE

After completing this exercise, the student will be able to:

a) Describe the function of "OR" logic in the PLC
b) Enter a basic program using "OR" logic with RSLogix 500 software

MATERIALS NEEDED

Allen-Bradley MicroLogix 1400 programmable controller with simulator
RSLogix 500 and RSLinx Software

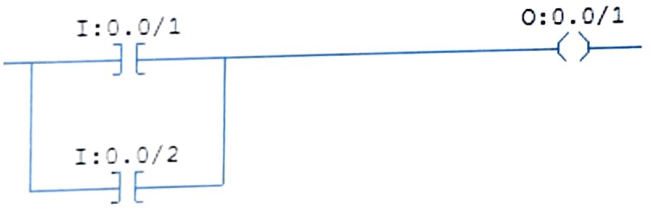
REFERENCES

Cox, Technician's Guide to Programmable Controllers
Allen-Bradley, Bulletin 1766 User Manual

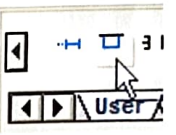
DISCUSSION

Boolean Logic is another way of expressing a PLC program. Boolean differs from Ladder Logic primarily in its use of branching instructions, known in Boolean as "OR" logic. A branch is a method of putting instructions in parallel on a rung.

In the program below, XIC "1" is in parallel with XIC "2". In Boolean this means that the rung can be made true by more than one combination of conditions. This rung would be true if XIC "1" was true OR if XIC "2" was true OR if they were both true. As we found in the last lab AND circuits only had one combination that gave a true rung condition.



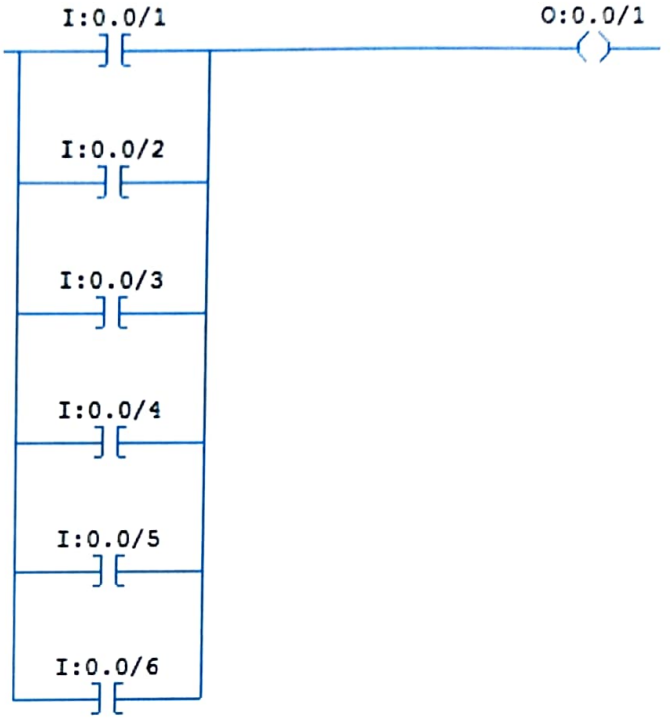
1. Using the techniques shown in the previous labs, create a new program in RSLogix 500 and name the program LAB04.
2. Add a new rung in the program.
3. From the User Tool Bar drag a Rung Branch on to the rung.
4. Add the two XIC's and the OTE on to the rung.
5. Drag the appropriate addresses on to the instructions.
6. Download the program to the controller, put it in run mode and go online with it.



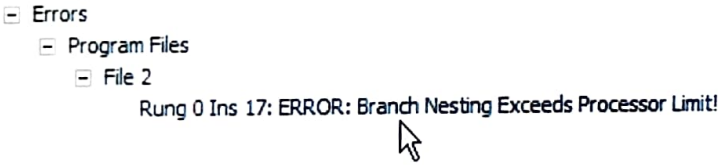
Demonstrate that Output 1 will energize if either Input 1 OR Input 2 is activated.
When finished, go offline and continue with the lab.

Using what you just learned, create the following program:

1. In the offline mode drag another Rung Branch under XIC "2" and release it on a green box. Notice it did not go around XIC "2". This is what A-B calls target branching. Click on the tall red box and drag it to the right side of XIC "2" and release it on a green target.
2. Drag another XIC from the tool bar on that new branch.
3. Use steps 1 and 2 to add three more XIC branches on the rung.
4. Now drag the appropriate addresses on to all of the XIC's that you added.



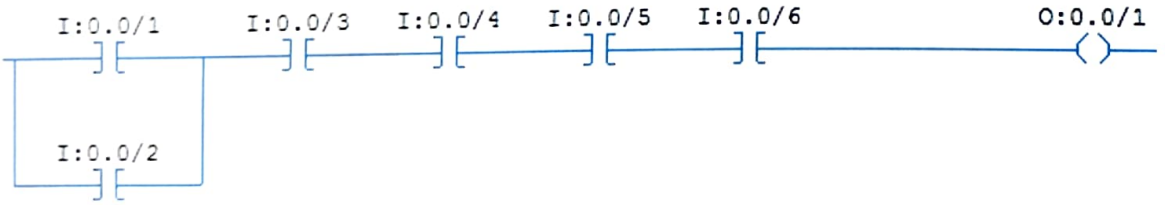
All of the XIC's are now parallel and the right addresses are on them but it looks like an inverted wedding cake instead of like the picture. Oh well, electrically it is the same thing, so click on the Verify File (green check mark on the left) on the tool bar. Interesting, it didn't compile. Instead you got an error that looks like this:



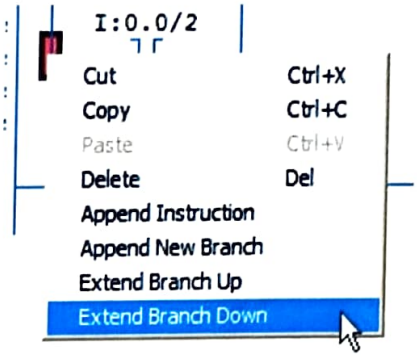
Here's the Concept

The Rung Branch instruction automatically opens the branch and closes the branch, then you drag the end where you want it to end. This is called a "nested" branch and you can only have 5 of these nested together. Nested branches are useful if you need to have an extra condition ahead of an existing branch, but that isn't what needs to be done at the moment.

5) Drag your XIC instructions to the upper rung and delete the nested branches by clicking on the lower left corner and pressing the delete key on the keyboard, so your program looks something like this:



6) Now right click on the lower corner of the remaining branch and select "Extend Branch Down", as shown on the right:
7) Repeat step 6 three more times.
8) Now drag XIC instructions 3 through 6 on to the appropriate branches.
9) Run Verify File again. Notice that not only are the branches lined up nicely but that bad error message went away.



With this method of branching you are no longer limited to 5 branches. It becomes a memory limitation.

10) Download the program, put the controller back in run mode and go online with it.

Demonstrate that Output 1 will energize if any one of the Inputs 1, 2, 3, 4, 5 or 6 is on. Demonstrate that no other inputs will energize Output 1. Call your instructor if your program is behaving differently.

Q1. Complete the Truth Table below for this program. (Parallel Circuit on Pg.2 Top of page)

XIC 1	XIC 2	XIC 3	XIC 4	XIC 5	XIC 6	OTE 1
1	1	0	0	0	0	1
1	1	0	0	0	1	1
1	0	0	0	1	0	1
1	0	0	0	1	1	1
1	0	1	1	0	0	1
1	0	1	1	0	1	1
1	0	1	1	1	1	1
1	0	1	0	0	0	1
1	0	1	0	0	1	1
1	0	1	0	1	0	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
0	1	0	0	0	0	1
0	1	0	0	0	1	1
0	1	0	0	1	0	1
0	1	0	0	1	1	1
0	1	0	1	0	0	1
0	1	0	1	0	1	1
0	1	0	1	1	0	1
0	1	0	1	1	1	1
0	1	1	0	0	0	1
0	1	1	0	0	1	1

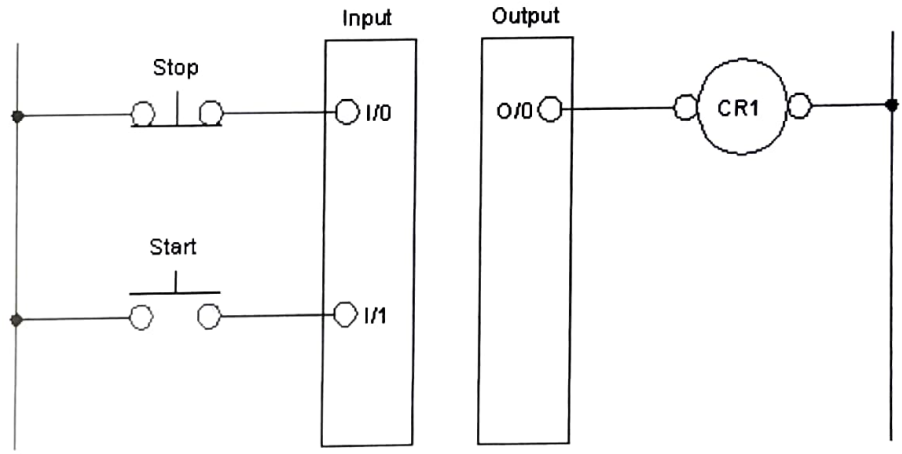
Note: this table is incomplete but should give you a good feel for creating a complete table, should the need ever arise.

Combining AND And OR Logic

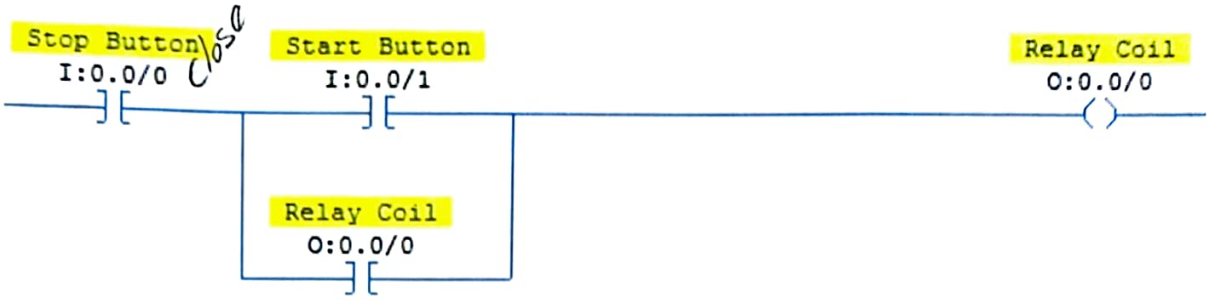
DISCUSSION

AND and OR logic can be combined to create very useful control circuits. A very common circuit that we know from motor control is a three wire start/stop circuit and these contacts should be arranged so that the N.C. stop contact is before the N.O. start contact and that a sealing contact or holding contact is wired in parallel with the start contact.

To do this in a PLC program a good understanding of XIC and XIO is very important. Let's start by walking through the logical process of a 3 wire start/stop circuit. Assuming the circuit is currently off, as long as nobody is pressing the stop button there is a complete circuit across it. If the start button is pressed the contact closes, completing the circuit and a coil energizes. A N.O. contact operated by the coil, wired parallel to the start contact closes, completing that circuit. When the start button is released, that circuit opens, but the circuit still has the path through the relay contact that acts as a holding circuit, so the relay stays energized. When the stop button is pressed that part of the circuit opens, so the relay coil opens, causing the holding contact to open, so when the stop button is released the relay stays off. In the program we are going to write there is a N.C. stop button wired to input 0, a N.O. start contact wired to input 1 and a relay coil wired to output 0. Always start with a drawing of the hardware before starting to write a program.



If the stop button is not being pressed, then there will be voltage on input 0 and there will be a 1 in the data table at that address. We want our circuit to be able to operate as long as no one is pushing the stop button, meaning that in our logic we want to test to see if there is a 1 in the data table so we know that the stop button is not being pressed. With that said the instruction that is used to test for a 1 in the data table is the XIC instruction, because it has the green handles around it when a 1 is present. This is why it is very important to have a wiring diagram of all of the devices connected to the PLC BEFORE you start programming. For the PLC logic to operate properly, the following program needs to be written:



Write the above program, using the steps you've already learned. If you right button click on each instruction and select "*Edit Description*" you can type in the names of the devices connected to these addresses. This makes your program much easier to read and troubleshoot than programs without documentation. This cannot be done with the HHT.

When you have finished writing and documenting the program, download it, put the processor back in the run mode and go online with it.

Q2. To simulate a N.C. Stop PB, we will start by closing switch 0 with all other switches off. Looking at the computer screen you should now see green handles on XIC I:0.0/0, the stop button. Now close switch 1. What instructions have green handles now?

All of them, All instructions have green handle

Q3. What are the data table values for switch 0 and 1?

I:0.0/0 = 1
I:0.0/1 = 1

Q4. To simulate releasing the Start button, open switch 1 now. Looking at the computer screen what is the true logic path when switch 1 is opened?

I:0.0/0 = 1 STOP close
O:0.0/0 = 1 → Relay coil Latch contact
O:0.0/0 = 1 → Relay coil

STOP	START	LATCH	RC
1	0	1	1

Q5. What is the data table value of switch 1 now? Notice how the data table and the truth tables you've been creating relate to each other.

I:0.0/1 = 0

Q6. To simulate the Stop PB being pressed, open switch 0 and observe what happened to the green handles on all of the instructions. Now simulate releasing the Stop button by closing switch 0. What instructions now have a true logic path?

Green handle all instructions dissapear

Switch 0 = I:0.0/0 = True logic green handle

Q7. With switch 0 closed, simulating the stop button not being pressed, close switch 1 momentarily then turn it off, simulating the start button being pressed and released. What is the true logic path now?

STOP	START	latch	coil
1	0	1	1

coil Relay on

Q8. Momentarily open switch 0, then close it again, simulating pressing the Stop PB. The true logic path is now XIC I:0.0/0 and all other instructions false. In other words the circuit is now stopped and waiting for a start command before it will do anything more. This very common circuit is a great example of, series AND logic and parallel OR logic, being combined together to perform a practical task. Describe in your own words the operation of the program for a three wire start/stop circuit.

This is a START - STOP circuit very useful TO START and STOP motor, The current go Through The STOP button (closed) when we push The START button curren energize The Relay coil That close The NO seal/latch contact TO KEEP The coil energized The STOP button cut The power To The coil and open The latch auxiliary contact Which Turn OFF The Relay coil