UNIVERSITY OF JORDAN
FACULTY OF ENGINEERING & TECHNOLOGY

MEASUREMENTS and CONTROL LAB. 0908448

EXPERIMENT N0. 5

Introduction to PLC and Ladder Logic

*Prepared by **Eng. Hussam Khasawneh** in March, 2009*

OBJECTIVES:

The aim of this experiment is to familiarize the student with Programmable Logic Controllers (PLC) as hardware and the software used to program it, also to familiarize him with programming PLC using ladder logic diagrams.

EQUIPMENT and APPARATUS:

1. Elettronica Veneta, Programmable Logic Controller Trainer, mod. PLC-8/EV, see figure 1.
2. 2x 24V solenoids.
3. Multimeter.
4. Personal computer.
5. Electrical wires.

Figure 1. *Programmable Logic Controller Trainer, mod. PLC-8/EV.*

INTRODUCTION TO PLC:

A computer system for automation has to satisfy many requirements that we take more or less for granted. It has to run all around the day and night since a production break may cost huge amounts of money. It has to work in real time taking various time requirements and process disturbances into consideration.

Whatever happens, the system has to behave in a predictable way. It has to be safe, both for the process and for humans.

In most automation systems there are events that will bring the process into another state of operation. Furthermore, there are a lot of applications in both the process and manufacturing industries where control involves primarily switching and sequencing. In both the process and manufacturing industries there is a wealth of applications of switching circuits for combinatorial and sequencing control.

Switching theory, which provides the foundation for binary control, is not only used in automation technology but is also of fundamental importance in many other fields. This theory provides the very principle on which the function of digital computers is based. In general, binary combinatorial and sequencing control is simpler than conventional feedback (analog and digital) control, because both the measurement values and the control signals are binary. However, binary control also has specific properties that have to be considered in more detail.

Programmable logical controllers (PLCs) have been in use since the 1960s and are still the basis for the low level control in many automation systems. Today PLCs can handle not only the lowest levels of control but also advanced control of hybrid systems, where time-driven continuous controllers have to be integrated with event-driven controllers.

The Development of PLCs

The modern computer control system of today is the result of two parallel developments, one from relay technology to implement logical circuits and the other from continuous instrumentation and pneumatic proportional integral derivative (PID) controllers developing into software realizations of continuous controllers.

Logical circuits have traditionally been implemented with different techniques.

The primary reason for designing a PLC was to eliminate the large cost involved in replacing the complicated relay-based machine control systems.

When production requirements changed, so did the control system. This becomes very expensive when the changes are frequent. Since relays are mechanical devices they also have a limited lifetime, which required strict adhesion to maintenance schedules. Troubleshooting was also quite tedious when so many relays were involved. Usually, the machine control panel includes many, possibly hundreds or thousands of, individual relays. Then it is easily recognized that alternative solutions were sought.

PLC brands

Siemens, Allen-Bradley, IDEC, ABB, Mitsubishi, Omron, Honeywell, Schneider Electric, Saia-Burgess Controls, and General Electric.

In this experiment we will use:

I. PLC SIEMENS SIMANTIC S7-200, see figure 2.

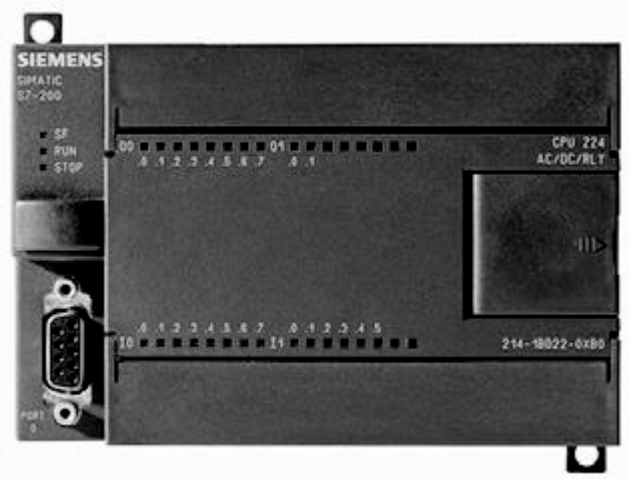


Figure 2. *PLC SIEMENS S7-200.*

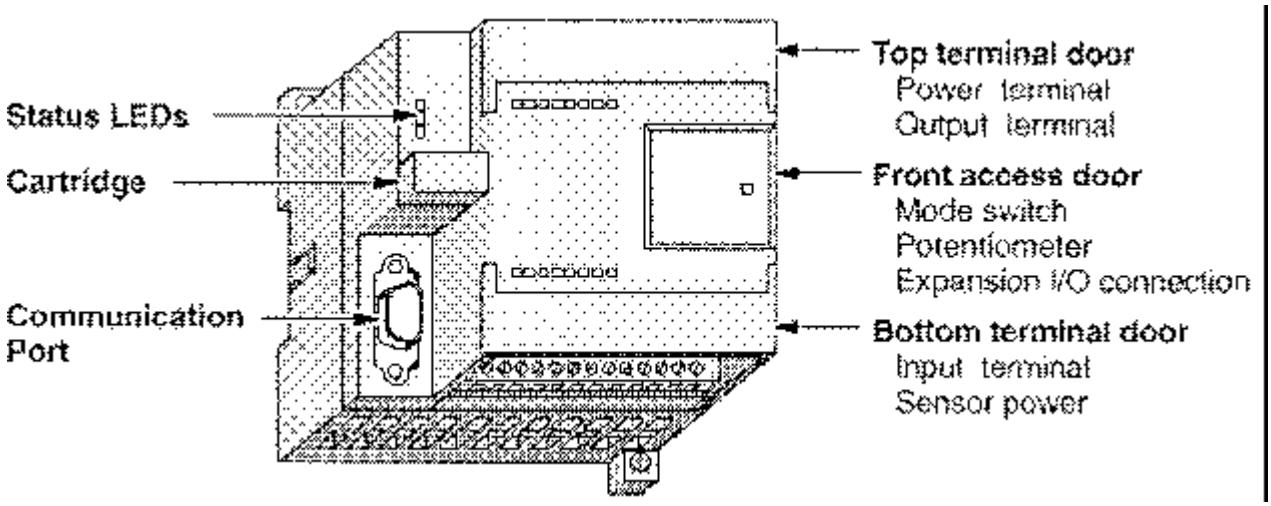


Figure 3.*PLC SIEMENS S7-200 parts.*

PLC SIEMENS S7-200 has the following characteristics:

Program 4096 words
User data 2560 words
User program storage type EEPROM
Local I/O 14 In/10 Out

- Digital Inputs: I0.0-I0.7 and I1.0-I1.5
- Digital Outputs: Q0.0-Q0.7 and Q1.0-Q1.1

Internal relays 256: M0.0-M31.7
Counters/Timers 256/256: C0-C255, T0-T255.
Number of expansion modules 7 modules, types of expansion modules:

- Discrete Input / Output
- Analog Input / Output
- RTD / Thermocouple input
- PROFIBUS-DP slave
- AS-Interface master
- Ethernet
- Internet
- Modem
- Positioning

Modules can be added directly next to the CPU, or in an extended arrangement with a 0.8m Expansion Cable, either with panel mounting or standard DIN rail mounting.

DC input features
Input type Sink/source (IEC type 1 sink).
Input Voltage Maximum continuous 30 VDC.
Surge 35 VDC for 0.5 s.
Rated value 24 VDC at 4 mA, nominal.
Logic 1 signal (minimum) 15 VDC at 2.5 mA, minimum.
Logic 0 signal (maximum) 5 VDC at 1 mA, maximum.

DC output features

Output specifications	DC output	Relay output
Output type	Solid State-MOSFET	Relay, dry contact
Permissible range	20.4 to 28.8 VDC	5-30VDC or 5-250VAC
Rated value	24 VDC	-
Logic 1 at max. current	20 VDC, minimum	-
Logic 0 with 10 K W load	0.1 VDC, maximum	-

II. Digital expansion EM 222

8 Points, 24 VDC Digital Output Module, see figure 4.

III. Analog input/output module EM 235

4 AI / 1 AO 12 bit, see figure 5.

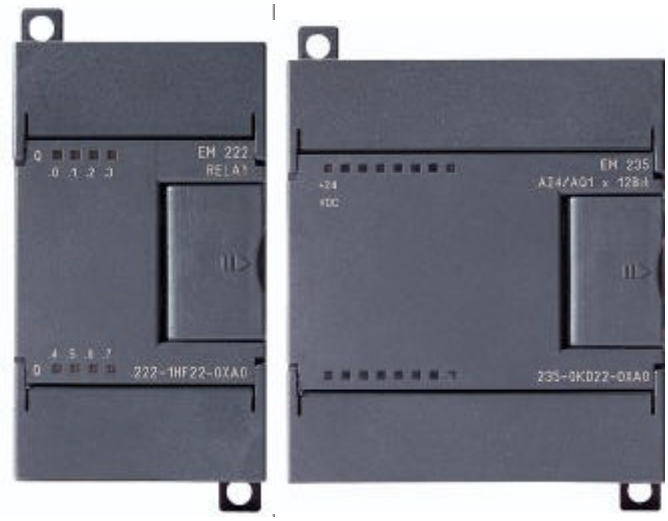


Figure 4. *EM222.* ***Figure 5.*** *EM 235.*

LADDER LOGIC:

It is a very popular graphical language for programming Programmable Logic Controllers (PLCs), Ladder Logic based on drawing electrical logic schematics to implement Boolean expressions. The Ladder Diagrams consists of graphic symbols, representing logic expressions, and contacts and coils, representing outputs.

A program in ladder logic, ladder diagram, is similar to a schematic for a set of relay circuits. The relay circuits are usually drawn in the form of wiring diagrams that show the power source and the physical arrangement of the various components of the circuit (switches, relays, motors, etc.) as well as their interconnections. The wiring diagrams are used by technicians to do the actual wiring of a control panel. The Ladder Diagram is a widely used representation form for logical circuits. It represents a conventional wiring diagram in schematic form, without showing each electrical connection explicitly. In an LD each branch of the control circuit is shown on separate horizontal rows (the "rungs" of the "ladder"), between two vertical "Rails", see figure 6.

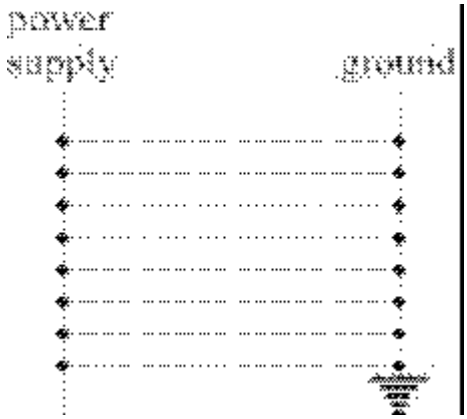


Figure 6. *Rungs and Rails.*

Each branch reflects one particular function and the related sequence of operations. In this drawing frame it is implicitly assumed that one of the vertical lines is connected to a voltage source and the other to ground. Note that all the rungs of the ladder are "executed" simultaneously in a wiring diagram.

In the LD are shown relay contacts that can be either of normally open or normally closed type (the normal state is the one in which the coil is not energized). The output consists of a relay (a coil) that could also symbolize a more complex circuit or a flip-flop. The drawing symbols for the switches and an actuator (relay) are shown in figure 7.

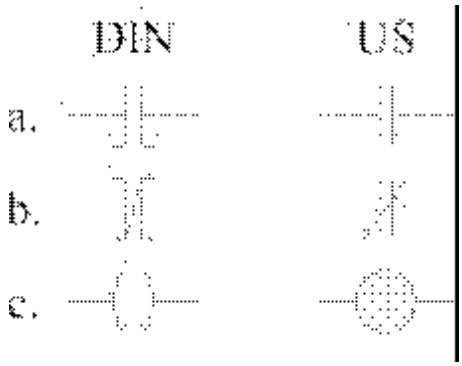


Figure 7. *DIN and the U.S. standards for symbols for (a) a normally open contact, (b) a normally closed contact, and (c) an output element.*

Normally Open Contact

A normally open (NO) switch is one that normally prevents current and which allows current when it is actuated.

In figure 8, normally the lamp is turnned OFF since the switch is Normally Open, it will turnned ON only if the switch is closed.

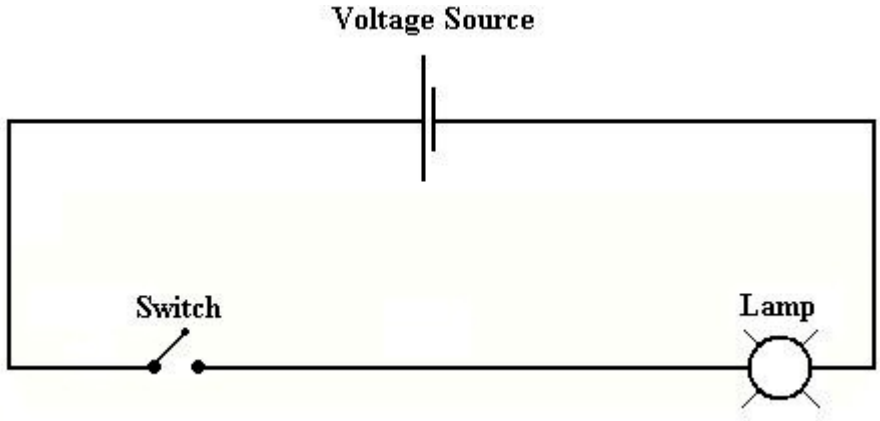


Figure 8. *A circuit consists of NO switch and Lamp.*

In figure 9, the equivalent circuit of the previous one in Ladder Logic Diagrams.

Figure 9. *A rung contains a NO contact and an output.*

Normally Closed Contact

A normally closed (NC) switch is one that normally allows current and which prevents current when it is actuated.

In figure 10, normally the lamp is turnned ON since the switch is Normally Closed, it will turnned OFF only if the switch is activated.

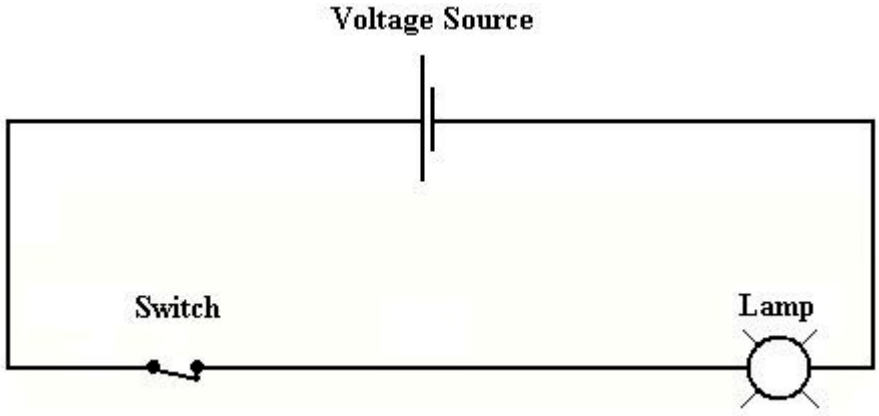


Figure 10. *A circuit consists of NC switch and Lamp.*

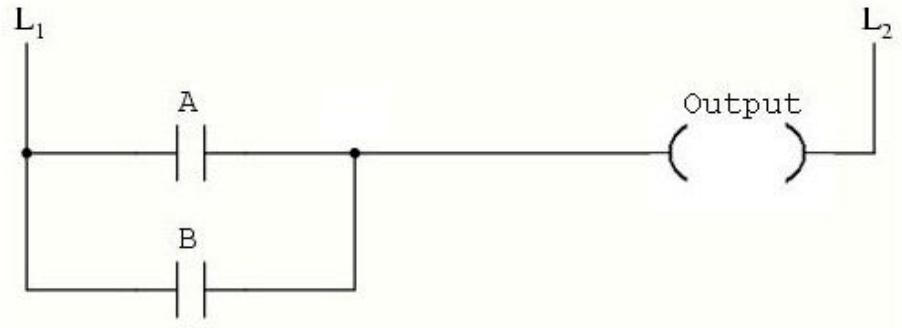
In figure 11, the equivalent circuit of the previous one in Ladder Logic Diagrams.

Figure 11. *A rung contains a NC contact and an output.*

Digital logic functions

Using multiple contacts, simple and complex logic functions can be constructed easily. It is useful to use standard binary notation for the status of the switches and lamp (0 for un-actuated or de-energized; 1 for actuated or energized), a truth table can be made to show how the logic works:

OR Gate



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

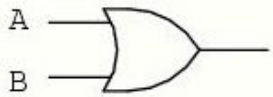
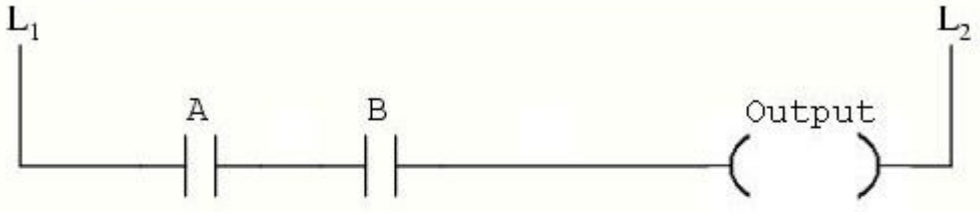


Figure 12. *OR.*

Now, the output will come on if either contact A or contact B is actuated, because all it takes for the lamp to be energized is to have at least one path for current from wire L1 to the wire L2. What we have is a simple OR logic function, implemented with nothing more than contacts and an output.

AND Gate



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

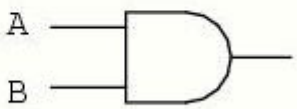


Figure 13. *AND.*

Now, the output energizes only if contact A and contact B are simultaneously actuated. A path exists for current from wire L1 to the wire L2, if and only if both switch contacts are closed.

NOT Gate

A	Output
0	1
1	0

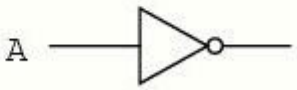


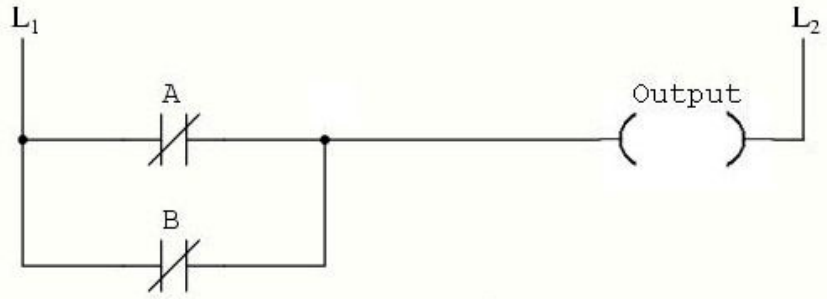
Figure 14. *NOT.*

The logical inversion, or NOT, function can be performed on a contact input simply by using a normally-closed contact.

Now, the output energizes if the contact is not actuated, and de-energizes when the contact is actuated.

NAND Gate

If we take our OR function and invert each "input" through the use of normally-closed contacts, we will end up with a NAND function. In a special branch of mathematics known as Boolean algebra, this effect of gate function identity changing with the inversion of input signals is described by DeMorgan's Theorem.



A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

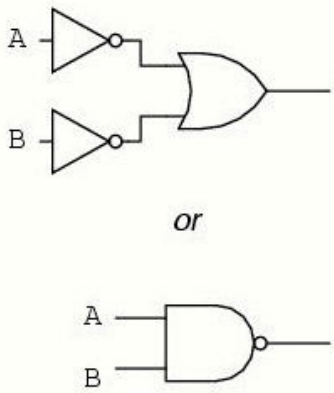
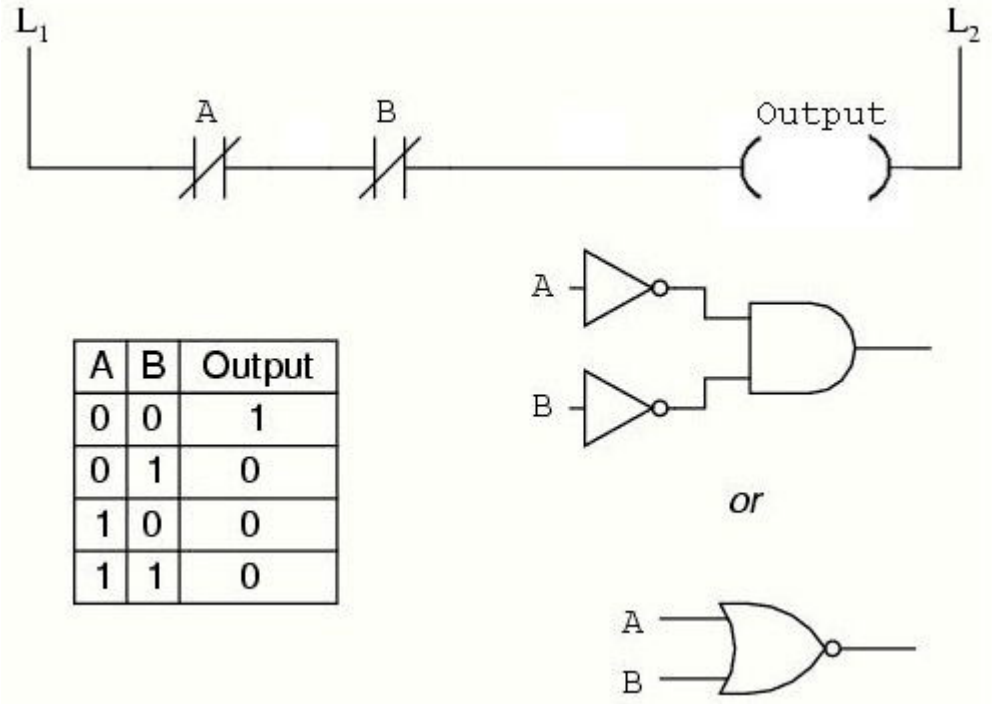


Figure 15. *NAND.*

The output will be energized if either contact is un-actuated. It will go out only if both contacts are actuated simultaneously.

NOR Gate

Likewise, if we take our AND function and invert each "input" through the use of normally-closed contacts, we will end up with a NOR function:



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

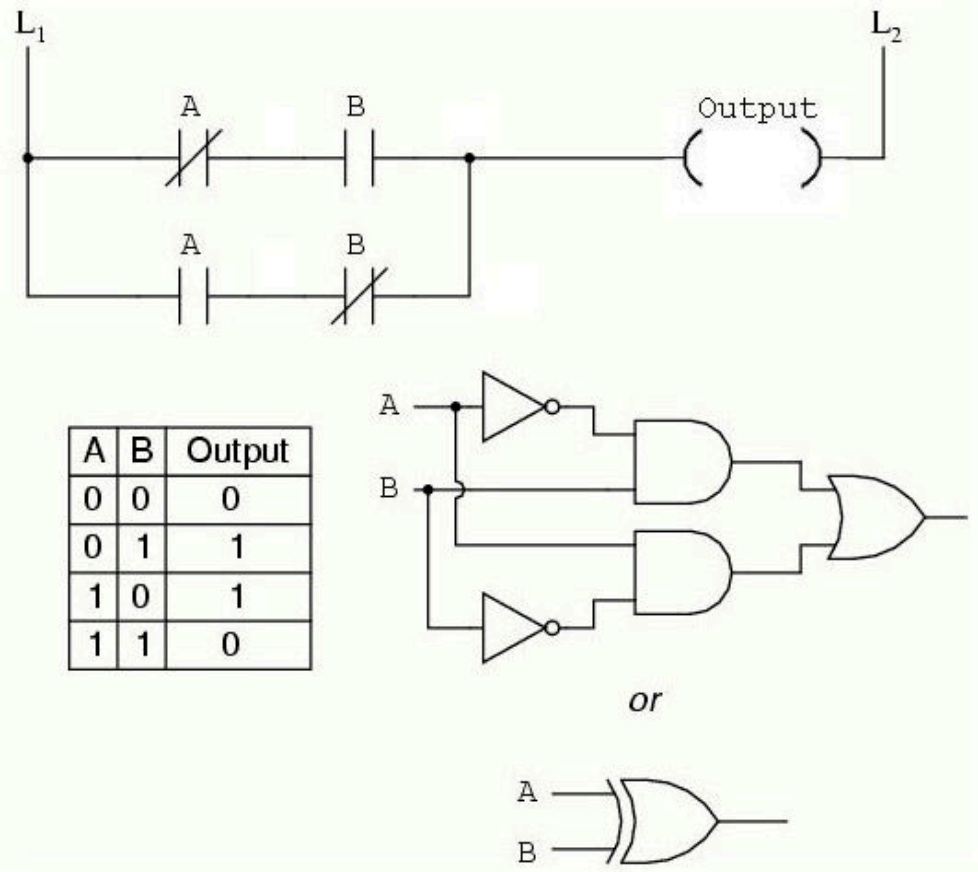
Figure 16. *NOR.*

A pattern quickly reveals itself when ladder circuits are compared with their logic gate counterparts:

- Parallel contacts are equivalent to an OR gate.
- Series contacts are equivalent to an AND gate.
- Normally-closed contacts are equivalent to a NOT gate (inverter).

XOR Gate

We can build combinational logic functions by grouping contacts in series-parallel arrangements, as well. In the following example, we have an Exclusive-OR function built from a combination of AND, OR, and inverter (NOT) gates:



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Figure 17. *XOR.*

Latching

If we have an electrical motor and a pushbutton for starting the motor, we can simply connect the pushbutton to one of the PLC's inputs and the motor to an output relay. The ladder will be as shown in figure 18.

Figure 18. *Pushbutton operates motor.*

But the problem here is that the output relay (motor) will turn off once you remove your finger from the pushbutton, but this is not the purpose of the pushbutton.

To solve this problem, we need to latch the output in parallel to the push button, as shown in figure 19.

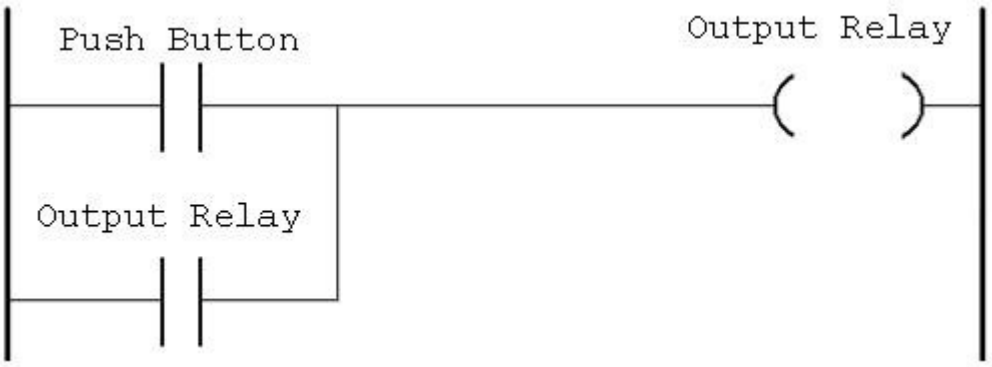


Figure 19. *Latching.*

Now, when you activate the push button, the output relay will be energized, so the electrical motor will turn ON, at the same moment the normally open contact (Output Relay) will be activated. As a result, even if you hold the pushbutton the output relay will stay ON due to the other contact.

But here the motor will stay ON forever; to stop it you need another pushbutton, let say Stop PB as shown in figure 20.

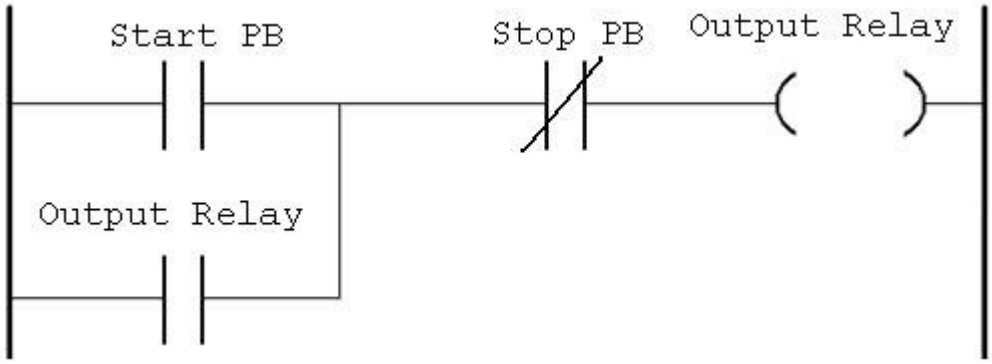


Figure 20. *Latching the start with stop pushbutton.*

Now, to stop the motor, all what you need to is pressing the stop pushbutton, so the output relay will disconnect the power of the motor.

HOW TO PROGRAM THE PLC:

To program the PLC, we will use the STEP7-Micro/WIN software.

1. Turn the PC ON.
2. Turn the PLC ON.
3. Start V4.0 STEP 7 Micro WIN SP5.
4. Select View>>Ladder, see figure 21.

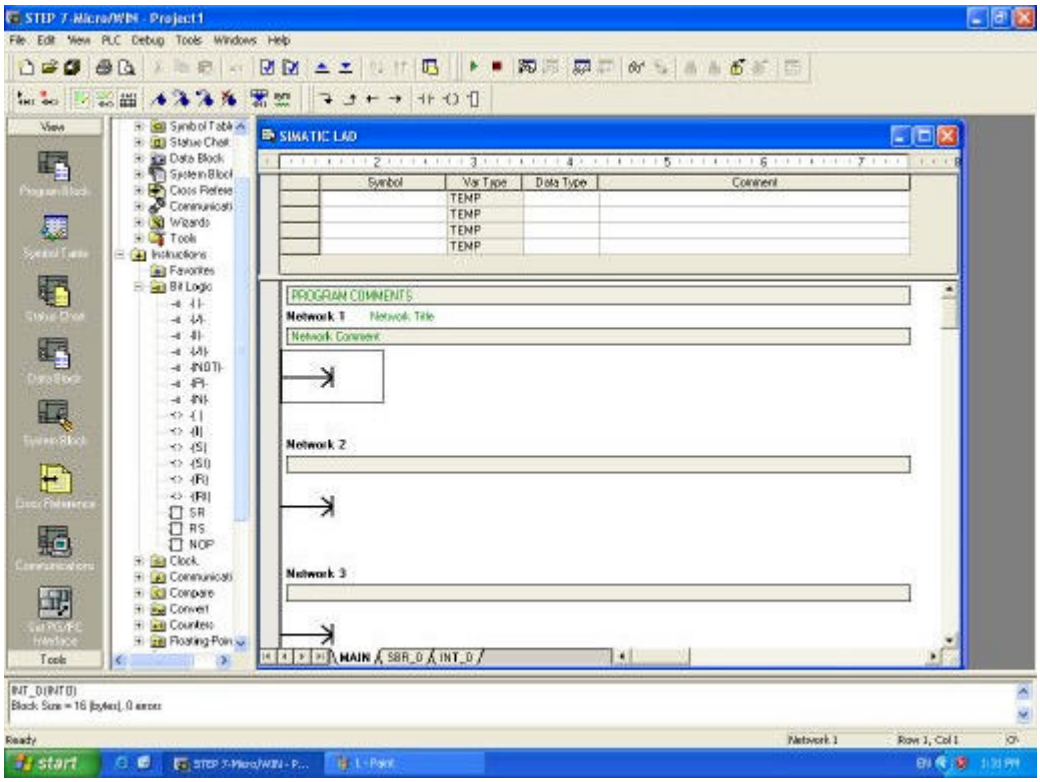


Figure 21.

5. Form Bit Logic double click the normally open contact.

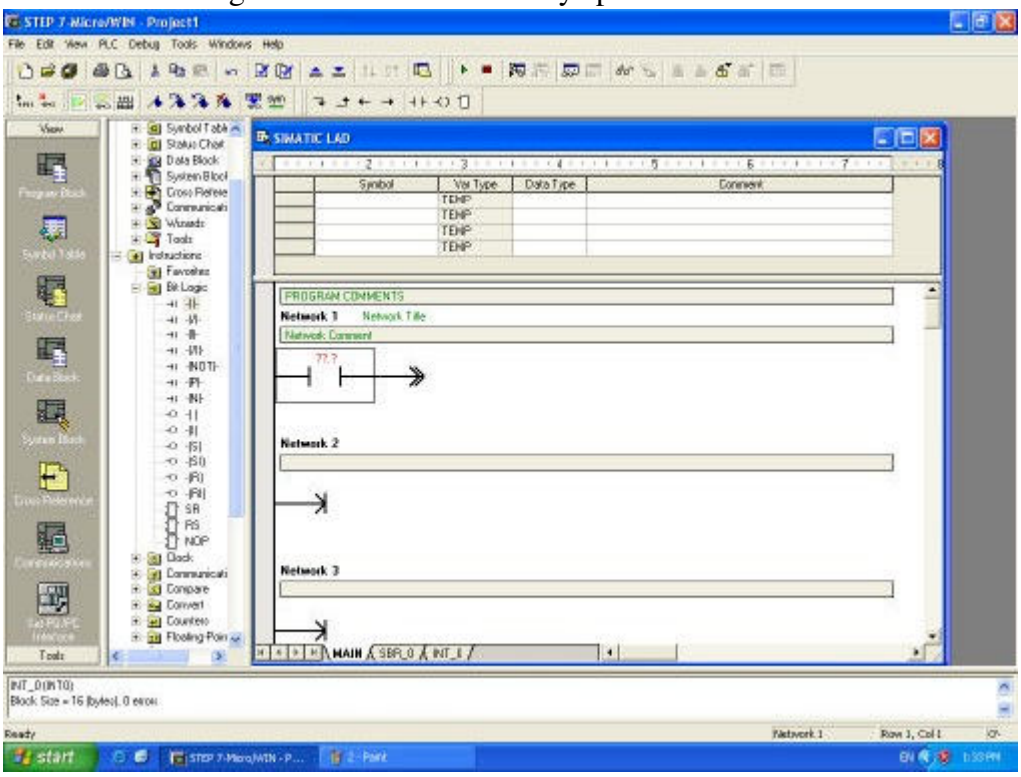


Figure 22.

6. Click on the red question marks, and rename the normally open contactor with a physical input name, I0.0-I0.7 or I1.0-I1.5, let us say I0.0.

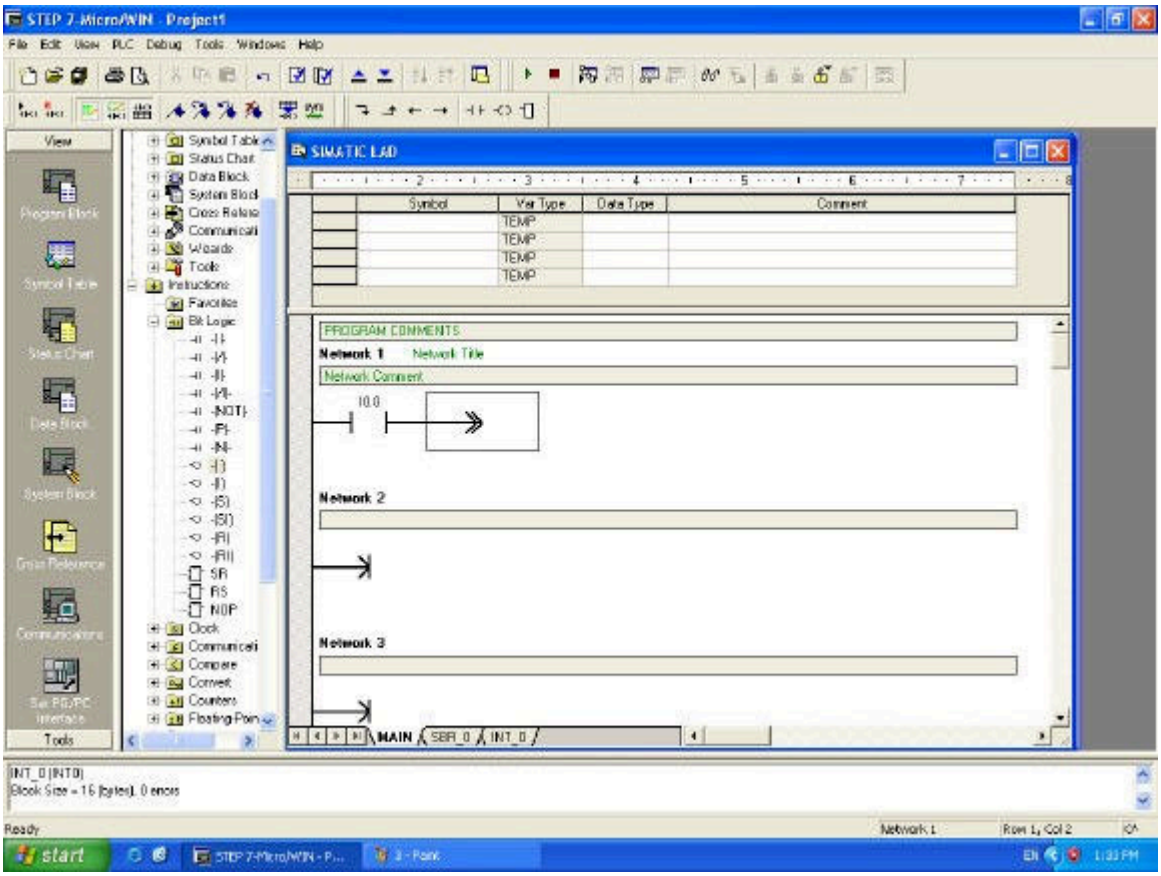


Figure 23.

7. Now from the Bit Logic, double click the output, and name it with a physical output name, Q0.0-Q0.7 or Q1.0-Q1.1, let us say Q0.0, the result rung will be as shown in figure 24.

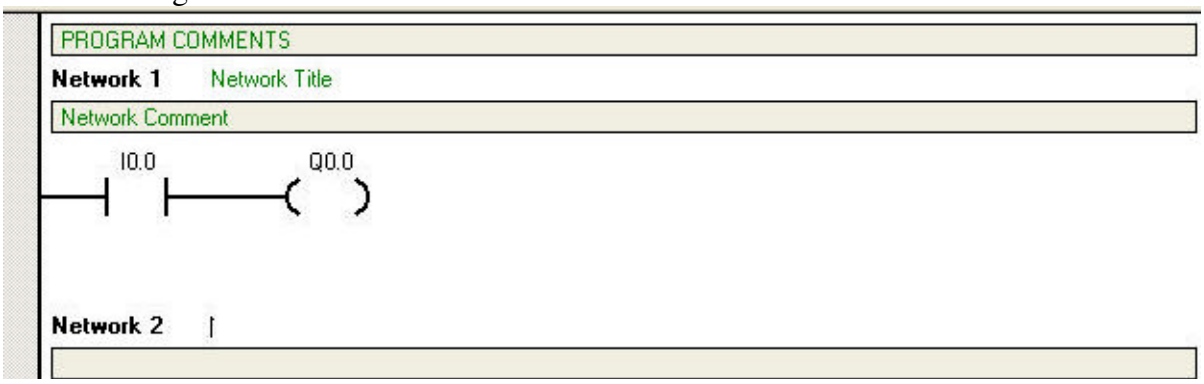


Figure 24. *The result rung.*

8. Select PLC>>Compile All; make sure that the "Total Errors" is zero.

9. Select PLC>>STOP to place the PLC in the stop mode.

10. To download the ladder diagram to the PLC, click download (shown to the right), the window for downloading appears as shown in figure 25, click download.

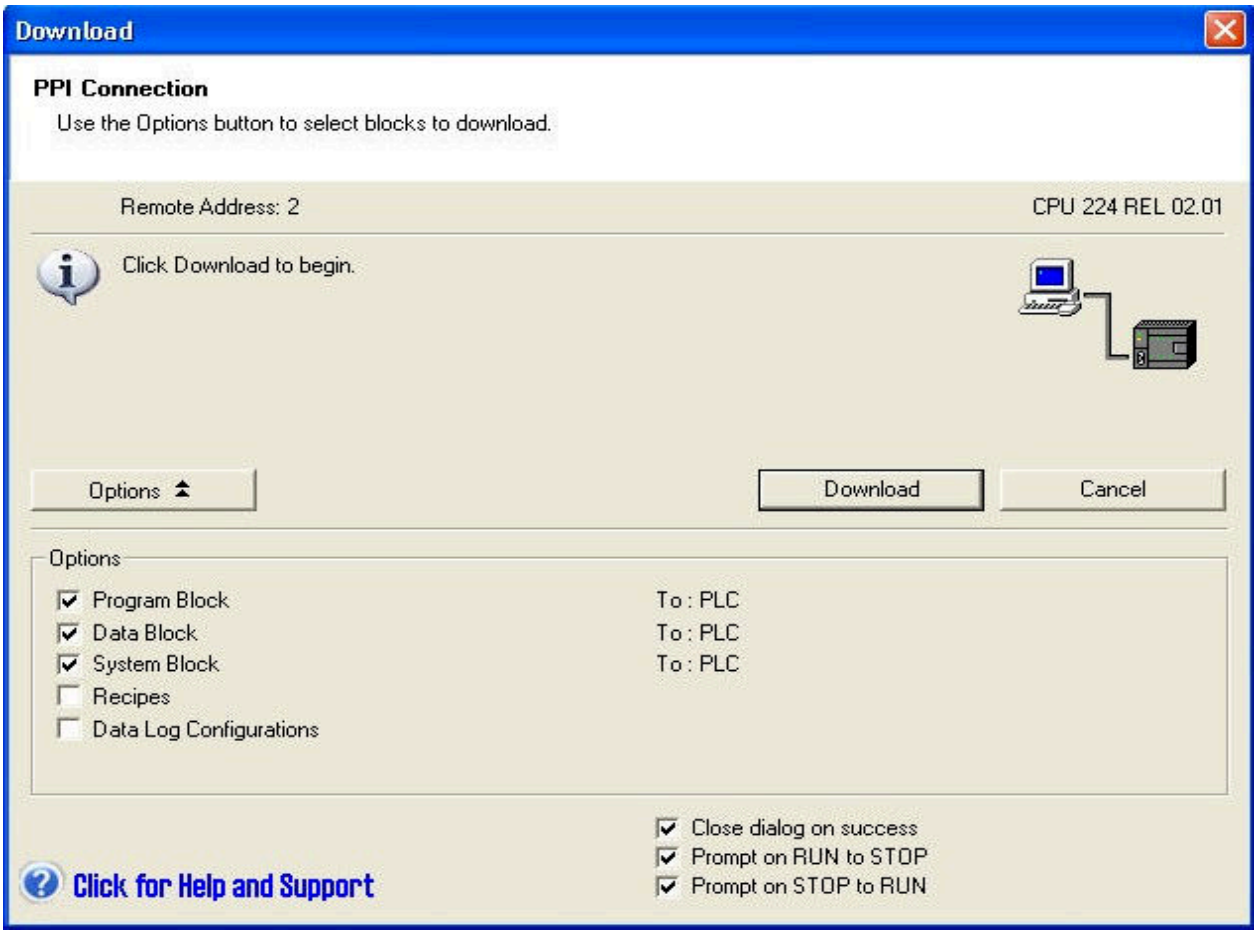


Figure 25. *Downloading window.*

11. Now place the PLC in the run mode, by selecting PLC>>RUN.
12. Now the PLC is ready for use performing the function of the ladder diagram that you have drawn.

OUTPUT RELAY:

You will use solenoids that operate using larger current than what the PLC can source, so you need output relays, as shown in figure 26.

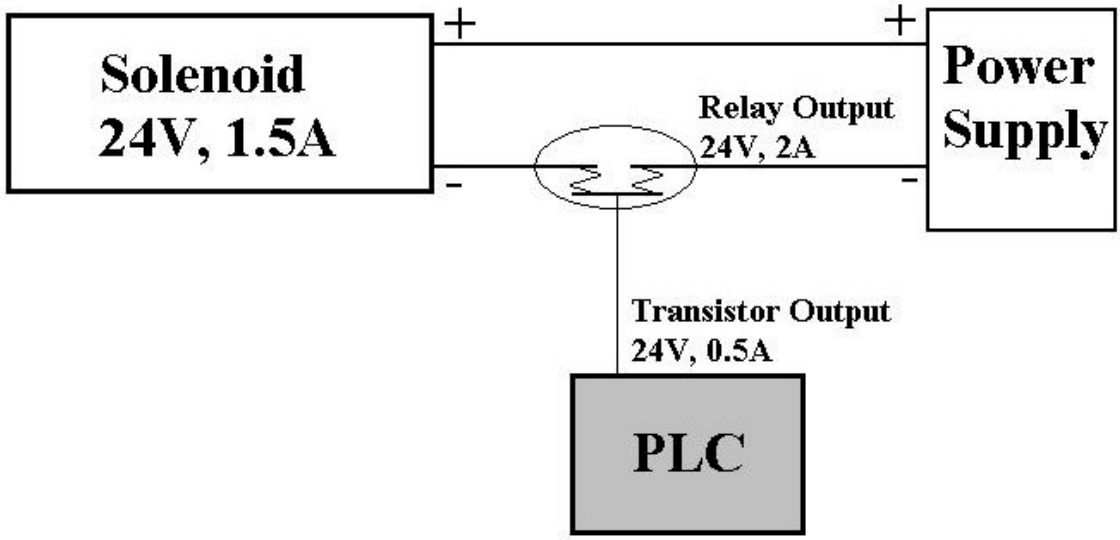


Figure 26. *Relay's function.*

MOTOR CONTROL CIRCUITS:

One of the applications of the PLC is to control the direction of rotation for the 3-Phase Induction Motor, see figure 27.

When you select M1, The sequence of phases will be **A, B, C** and that will occur forward rotation. But when you select M2, the sequence of phases will be **B, A, C** and that will occur reverse rotation.

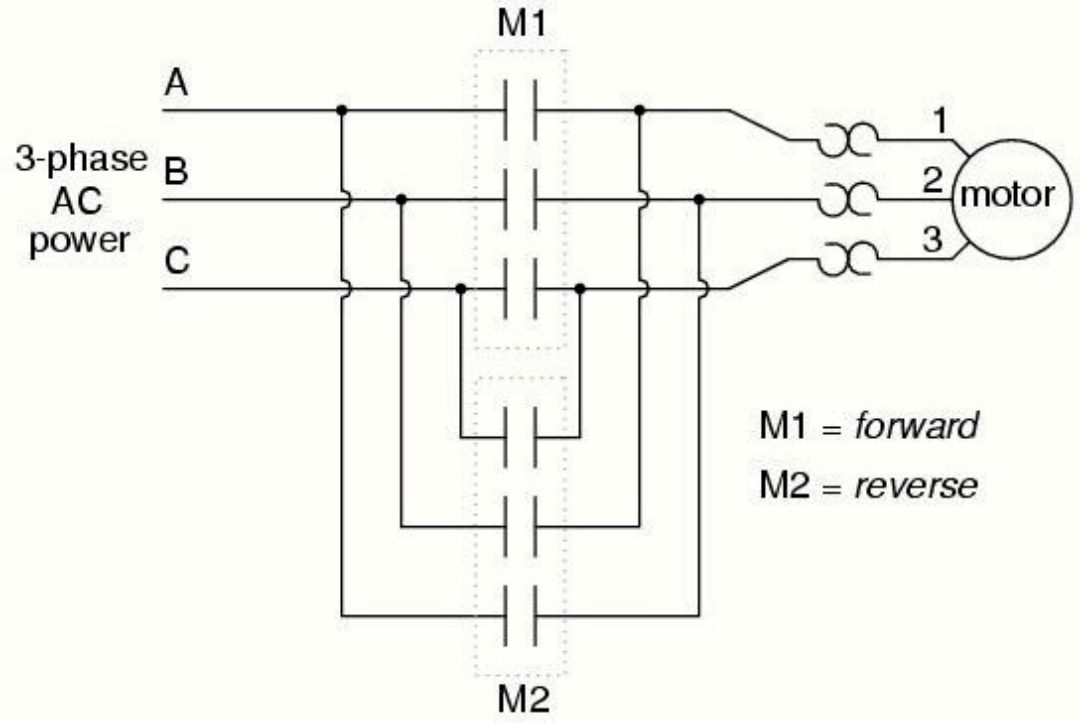


Figure 27. *The circuit that is used to control the direction of rotation for 3-phase motor.*

To control the direction of rotation for this motor, you should connect the three phases into **starter** with two control signals (M1) and (M2).

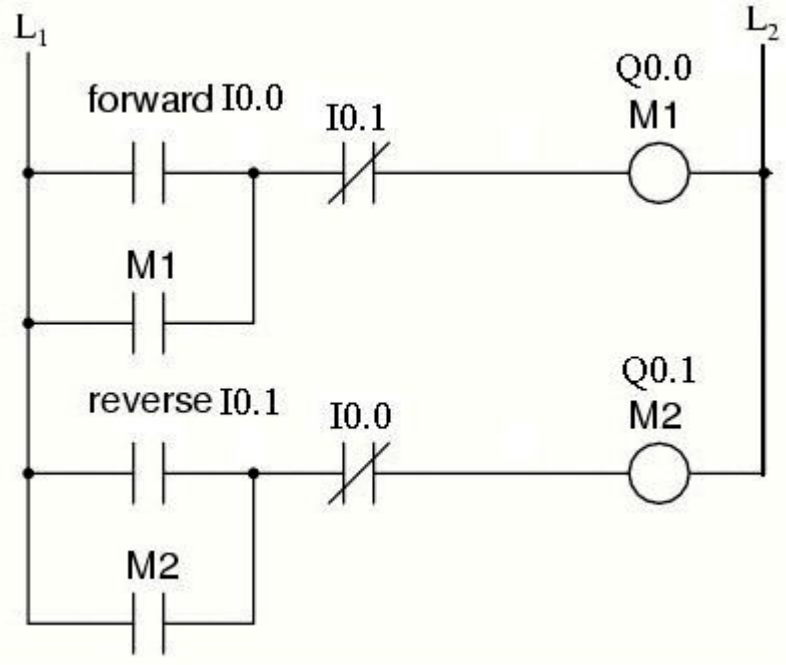


Figure 28. *The ladder logic diagram to control the direction of rotation for 3-phase motor.*

Note:
The rung can contain many inputs, but should contain ONE output only.

TIMERS/COUNTERS

On-Delay Timer

The On-Delay Timer (TON) instruction counts time when the enabling input is ON. When the current value (Txxx) is greater than or equal to the present time (PT), the timer bit is ON.

The On-Delay timer current value is cleared when the enabling input is OFF.

This timer continues counting after the Preset is reached, and it stops counting at the maximum value of 32767.

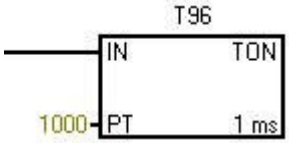
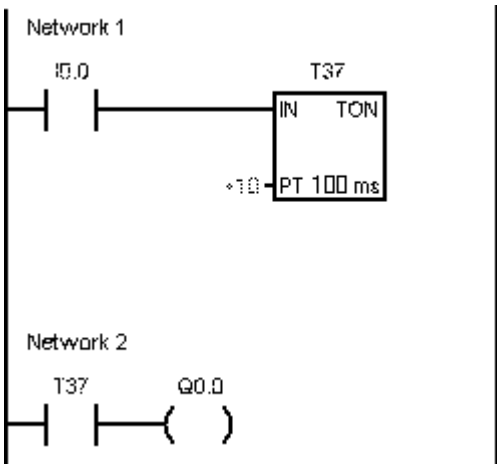


Figure 28. *TON.*

The timer shown in figure 28, will delay for 1000*1ms= 1000ms = 1 second then it will be activated ON.



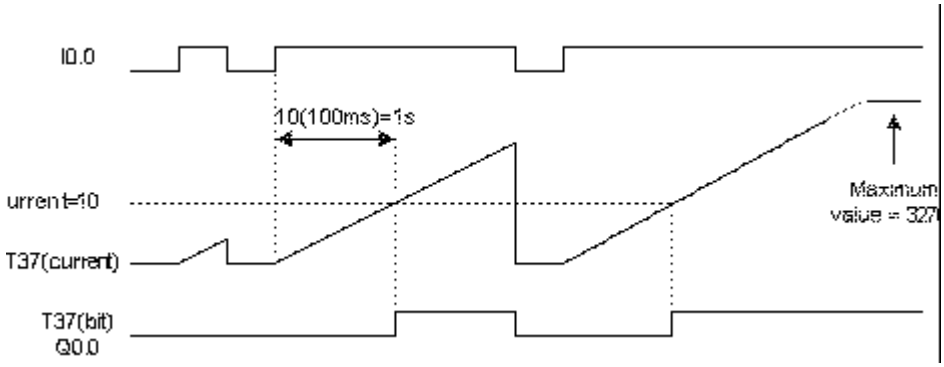


Figure 29. *(a) Ladder logic diagram for on-delay timer. (b) the timing diagram.*

Here when you activate the input I0.0, the timer T37 starts delaying time, until it reaches 1000ms, then T37 bit turns ON, hence Q0.0 will be ON. As a result Q0.0 will be activated after 1 second activating I0.0.

Off-Delay Timer

The Off-Delay Timer (TOF) is used to delay turning an output OFF for a fixed period of time after the input turns OFF. When the enabling input turns ON, the timer bit turns ON immediately, and the current value is set to 0. When the input turns OFF, the timer counts until the elapsed time reaches the preset time. When the preset is reached, the timer bit turns OFF and the current value stops counting. If the input is OFF for a time shorter than the preset value, the timer bit remains ON. The TOF instruction must see an ON to OFF transition to begin counting.

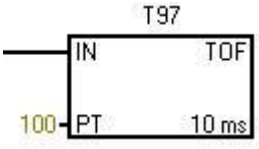


Figure 30. *TOF.*

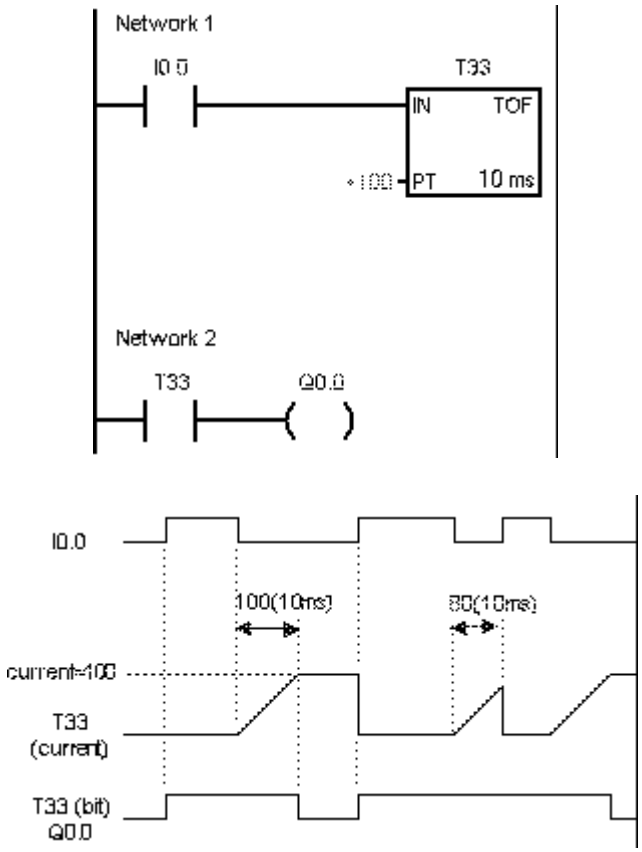


Figure 31. *(a) Ladder logic diagram for off-delay timer. (b) The timing diagram.*

Here when you switch the input I0.0 OFF, the timer T33 will delay for 1000ms before the T33 bit turns OFF.

Notes:

- You cannot share the same timer numbers for TOF and TON. For example, you cannot have both a TON T32 and a TOF T32.
- The Reset (R) instruction can be used to reset any timer. The Reset instruction performs the following operations Timer Bit = OFF and Timer Current = 0
- TON, TONR, and TOF timers are available in three resolutions. The resolution is determined by the timer number as shown in the chart below. Each count of the current value is a multiple of the time base. For example, a count of 50 on a 10-ms timer represents 500 ms.

Timer Type	Resolution	Maximum Value	Timer Number
TON, TOF	1 ms	32.767 s	T32, T96
	10 ms	327.67 s	T33-T36, T97-T100
	100 ms	3276.7 s	T37-T63, T101-T255

Count Up Counter

The Count Up (CTU) instruction counts up from the current value each time the count-up input CU makes the transition from off to on. When the current value (Cxxx) is greater than or equal to the Preset Value (PV), the counter bit (Cxxx) turns on. The counter is reset when the Reset (R) input turns on, or when the Reset instruction is executed. The counter stops counting when it reaches the maximum value (32,767).

Counter ranges: Cxxx=C0 through C255

Count Down Counter

The Count Down (CTD) instruction counts down from the current value of that counter each time the count down input CD makes the transition from off to on. When the current value Cxxx is equal to zero, the counter bit (Cxxx) turns on. The counter resets the counter bit (Cxxx) and loads the current value with the preset value (PV) when the load input (LD) turns on. The Down Counter stops counting when it reaches zero, and the counter bit Cxxx turns on.

Counter ranges: Cxxx=C0 through C255

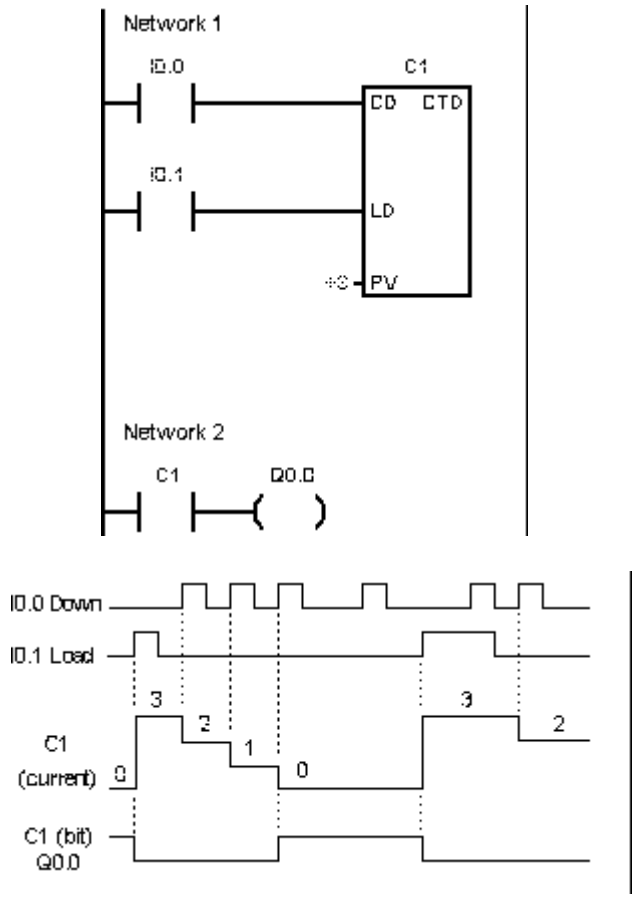


Figure 32. *(a) Ladder logic diagram for Count Down Counter. (b) The timing diagram.*

Additional exercises

Note

1. **The rung can contain many inputs, but should contain ONE output only.**
2. **Let the Engineering staff check what you have achieved**

1. Program the PLC with ladder diagram that do all the following logical functions:
   - Q0.1 = NOT I0.1
   - Q0.2 = I0.2 (OR) I0.3
   - Q0.3 = I0.4 (AND) I0.5
   - Q0.4 = I0.6 (NAND) I0.7
   - Q0.5 = I1.0 (NOR) I1.1
   - Q0.6 = I1.2 (XOR) I1.3

Check the results:

I0.1	Q0.1
0	
1	

I0.2	I0.3	Q0.2
0	0	
0	1	
1	0	
1	1	

I0.4	I0.5	Q0.3
0	0	
0	1	
1	0	
1	1	

I0.6	I0.7	Q0.4
0	0	
0	1	
1	0	
1	1	

I1.0	I1.1	Q0.5
0	0	
0	1	
1	0	
1	1	

I1.2	I1.3	Q0.6
0	0	
0	1	
1	0	
1	1	

2. Now build a new ladder diagram for turning the output Q0.0 ON when you activate the start pushbutton I0.0, and stop when you activate the pushbutton I0.1, recall figure 20.

Check the results (keep the sequence):

I0.0	I0.1	Q0.0
0	0	
0	1	
1	0	
1	1	

3. Program the PLC with the ladder diagram in figure (28).

Check the results (keep the sequence):

I0.0	I0.1	Q0.0	Q0.1
0	0		
0	1		
1	0		
1	1		

4. Draw the same ladder in step.3 again but add a new pushbutton (STOPM) that will stop the motor whenever the user activates it. Program the PLC with the new ladder diagram, after adding STOPM**.**

Check the results (keep the sequence):

I0.3 (stop)	I0.0	I0.1	Q0.0	Q0.1
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

5. Now program the PLC to do the following sequence:
   When you press the pushbutton (I0.0), (Q0.0) activated immediately, for 5 seconds, then it turns OFF.
   After it turns off, wait 10 seconds, and then turn Q0.1 ON.

Check the results (keep the sequence):

I0.0	Q0.0 Immediately	Q0.0 After 5 seconds	Q0.1 Immediately	Q0.1 After 10 seconds
0				
1				
0				

6. Now program the PLC to do the following sequence:
   When activating the pushbutton I0.0 five times, the output Q0.0 turns ON.
   When activating the pushbutton I0.1, Q0.0 turns OFF, pushing I0.1 one more time Q0.0 turns ON again.

Check the results (keep the sequence):

I0.0	I0.1	Q0.0
Pushed 1,2,3 and 4 times	0	
Pushed 5 times	0	
Pushed 5 times	Pushed 1 time	
	Pushed for the second time	

What will happen if Q0.0 has been already Off and pushed I0.1 once or twice?