

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**DEPARTMENT OF ELECTRONICS AND  
INSTRUMENTATION ENGINEERING**



**15EI401L - AUTOMATION LABORATORY**

**LABORATORY MANUAL**

# **CONTENTS**

- 1. Study of PLC**
- 2. Implementation of Code converters  
(Binary to gray & gray to Binary)**
- 3. Traffic light control system**
- 4. Water level control system**
- 5. Material handling system**
- 6. Bottle filling system**
- 7. Sequential operation of motor**
- 8. Star to delta starter**
- 9. DC motor speed control system**
- 10. Temperature control system**
- 11. Implementation of PLC programming through SCADA**

# 1. STUDY OF PLC

## Aim:

To study the Programmable Logic controller (PLC) and its logic operations.

## Introduction:

Controllers may consist of logical components and connections among them. Depending on the current logical value of input, output is produced to change the status of the system. PLC may realize such controllers. Today, the command and feedback control systems of industrial automation systems are realized by programmable logic controllers (PLCs). Siemens Simatic S7-200 is one of the PLC brands widely used in industry.

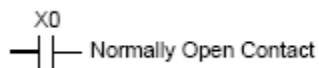
In order for PLCs to work as controllers, they must be able to realize some functions. These functions are basic and combinational logic operations such as AND, OR, AND-NOT, OR-NOT, timer and counter operations. In addition to these, PLCs may have the ability to realize several transfer, mathematical, and PID operations.

PLC consists of three main parts: CPU, memory and I/O units. CPU is the brain of PLC. It reads the input values from inputs, runs the program existed in the program memory and writes the output values to the output register. Memory is used to store different types of information in the binary structure form. The memory range of S7-200 is composed of three main parts as program, parameter, and retentive data fields. I/O units provide communication between PLC control systems.

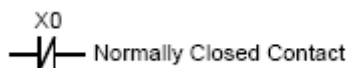
## Constructing of PLC Program:

There are mainly two methods for composing PLC programs: Ladder Logic Diagram (LAD) and Statement List (STL). LAD method is commonly used to implement the programs for process controls. A network of LAD is a row of connected elements that form a complete circuit between the left and right power rail. The left power rail represents the energized conductor whereas the right power rail represents the return path conductor of the circuit. Power flows from the left rail, through the closed contacts to the coils or boxes connected to the right power rail. You can then use the power flow to activate the outputs according to your program. The instructions from a ladder diagram, mnemonic are translated to machine code that can be stored in the PLC memory. Each horizontal rung on the ladder in a ladder program represents a line in the program and the entire ladder gives complete program in “ladder language”. There are three basic symbols used in ladder logic.

The first one is **NO - NC contacts** : NO contact is an instruction that tells the processor to look at a specific bit in its RAM memory. If the bit is 1, the instruction is true. and if it is 0, the instruction is false. The determining factor in choosing which bits in its memory to look at is the address. It could be some auxiliary bit (M), a timer contact (T), a counter contact (C), a state bit (S), or it might be connected to an external input (X)



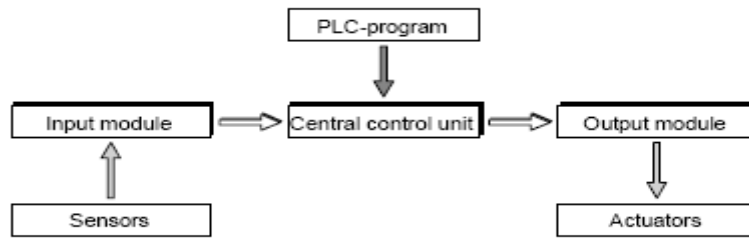
NC contact plays the same role as the previous one, except that if the bit addressed is 1, the instruction is false and if it is 0, the instruction is true.



The second symbol is **output**: for outputting to the output module. If the instructions to the left on its rung have a true path to the leftmost vertical rail, then the PLC will set the bit to which it is referenced via the address to 1. If no complete true path is available, it will set the bit to 0.



## System components of a PLC:



The function of an input module is to convert incoming signals into signals which can be processed by the PLC and to pass these to the central control unit. The reverse task is performed by an output module. This converts the PLC signal into signals suitable for the actuators. The actual processing of the signals is effected in the central control unit in accordance with the program stored in the memory. The program of a PLC can be created in various ways: via assembler type commands in 'statement list', in higher-level, problem-oriented languages such as structured text or in the form of a flow chart such as represented by a sequential function chart. In Europe, the use of function block diagrams based on function charts with graphic symbols for logic gates is widely used. In America, the 'ladder diagram' is the preferred language by users. Depending on how the central control unit is connected to the input and output modules, differentiation can be made between compact PLCs (input module, central control unit and output module in one housing) or modular PLCs.

### VersaPro Software:

- VersaPro, GE Fanuc's Windows-based programming software for the Series 90 and VersaMax PLCs.
- VersaPro is designed to install and run under Windows 95, Windows 98, Windows NT 4.0, and Windows 2000. With VersaPro, you can:
  - i. Create PLC logic and information associated with that logic in a folder
  - ii. Configure PLC Hardware
  - iii. Create and edit variables
  - iv. Create, edit, and monitor the execution of Ladder or Instruction List logic

### VersaPro's Functions Library:

- a) LD Bit Operations
- b) LD Boolean Functions
- c) LD Control Functions
- d) LD Conversion Functions
- e) LD Counter Functions
- f) LD Data Move Functions
- g) LD Data Table Functions
- h) LD Math Functions

- i) LD Numerical Functions
- j) LD Relational Functions
- k) LD Timer Functions

**TIMER:**

The simple on-delay timer (TMR) function increments while it receives power flow and resets to zero when power flow stops. Time may be counted in tenths of a second (the default selection), hundredths of a second, or thousandths of a second. The range is 0 to +32,767 time units, therefore the timing range is 0.001 to 3,276.7 seconds. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up.

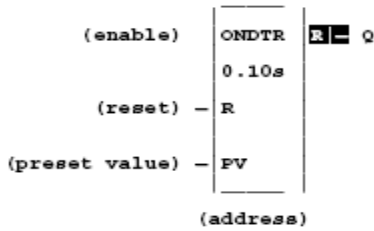
When the TMR receives power flow, the timer starts accumulating time (current value). The current value is updated when it is encountered in the logic to reflect the total elapsed time the timer has been enabled since it was last reset.



Parameter	Description
address	<p>The TMR uses three consecutive words (registers) of %R memory to store the following:</p> <ul style="list-style-type: none"> <li>• Current value (CV) = word 1.</li> <li>• Preset value (PV) = word 2.</li> <li>• Control word = word 3.</li> </ul> <p>When you enter a TMR, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function.</p> <p><b>Note:</b> Do not use this address with other instructions.</p> <p><b>Caution:</b> Overlapping references will result in erratic operation of the timer.</p>
enable	When enable receives power flow, the timer's current value is incremented. When the TMR is not enabled, the current value is reset to zero and Q is turned off.
PV	PV is the value to copy into the timer's preset value when the timer is enabled or reset.
Q	Output Q is energized when TMR is enabled and the current value is greater than or equal to the preset value.

**ONDTR:**

A retentive on-delay timer (ONDTR) increments while it receives power flow and holds its value when power flow stops. Time may be counted in tenths of a second (the default selection), hundredths of a second, or thousandths of a second. The range is 0 to +32,767 time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up. When the ONDTR first receives power flow, it starts accumulating time (current value). When this timer is encountered in the ladder logic, its current value is updated.

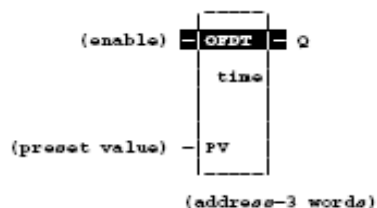


Parameter	Description
address	<p>The ONDTR uses three consecutive words (registers) of %R memory to store the following:</p> <ul style="list-style-type: none"> <li>• Current value (CV) = word 1.</li> <li>• Preset value (PV) = word 2.</li> <li>• Control word = word 3.</li> </ul> <p>When you enter an ONDTR, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function.</p> <p><b>Note:</b> Do not use this address with other instructions.</p> <p><b>Caution:</b> Overlapping references will result in erratic operation of the timer.</p>
enable	When enable receives power flow, the timer's current value is incremented.
R	When R receives power flow, it resets the current value to zero.
PV	PV is the value to copy into the timer's preset value when the timer is enabled or reset.
Q	Output Q is energized when the current value is greater than or equal to the preset value.
time	Time increment is in tenths (0.1), hundredths (0.01), or thousandths (0.001) of seconds for the low bit of the PV preset value.

## OFDT:

The off-delay timer (OFDT) increments while power flow is off, and resets to zero when power flow is on. Time may be counted in tenths of a second (the default selection), hundredths of a second, or thousandths of a second. The range is 0 to +32,767 time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up.

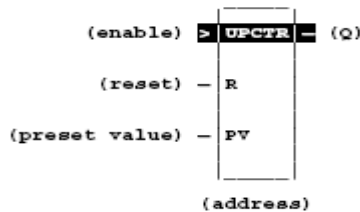
When the OFDT first receives power flow, it passes power to the right, and the current value (CV) is set to zero. (The OFDT uses word 1 [register] as its CV storage location—see the “Parameters” section on the next page for additional information.) The output remains on as long as the function receives power flow. If the function stops receiving power flow from the left, it continues to pass power to the right, and the timer starts accumulating time in the current value.



Parameter	Description
address	<p>The OFDT uses three consecutive words (registers) of %R memory to store the following</p> <ul style="list-style-type: none"> <li>• Current value (CV) = word 1.</li> <li>• Preset value (PV) = word 2.</li> <li>• Control word = word 3.</li> </ul> <p>When you enter an OFDT, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function.</p> <p><b>Note:</b> Do not use this address with other instructions.</p> <p><b>Caution:</b> Overlapping references will result in erratic operation of the timer.</p>
enable	When enable receives power flow, the timer's current value is incremented.
time	Time increment is in tenths (0.1), hundredths (0.01), or thousandths (0.001) of seconds for the low bit of the PV preset value.
PV	PV is the value to copy into the timer's preset value when the timer is enabled or reset.
Q	Output Q is energized when the current value is less than the preset value. The Q state is retentive on power failure; no automatic initialization occurs at power-up.

## UPCTR:

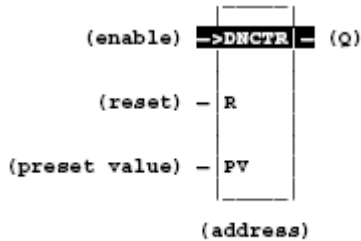
The Up Counter (UPCTR) function is used to count up to a designated value. The range is 0 to 32,767 counts. When the up counter reset is ON, the current value of the counter is reset to 0. Each time the enable input transitions from OFF to ON, the current value is incremented by 1. The current value can be incremented past the preset value PV. The output is ON whenever the current value is greater than or equal to the preset value. The state of the UPCTR is retentive on power failure; no automatic initialization occurs at powerup.



Parameter	Description
address	<p>The UPCTR uses three consecutive words (registers) of %R memory to store the following:</p> <ul style="list-style-type: none"> <li>• Current value (CV) = word 1.</li> <li>• Preset value (PV) = word 2.</li> <li>• Control word = word 3.</li> </ul> <p>When you enter an UPCTR, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function.</p> <p><b>Note:</b> Do not use this address with another up counter, down counter, or any other instruction or improper operation will result.</p> <p><b>Caution:</b> Overlapping references will result in erratic operation of the counter.</p>
enable	On a positive transition of enable, the current count is incremented by one.
R	When R receives power flow, it resets the current value back to zero.
PV	PV is the value to copy into the counter's preset value when the counter is enabled or reset.
Q	Output Q is energized when the current value is greater than or equal to the preset value.

## DNCTR:

The Down Counter (DNCTR) function is used to count down from a preset value. The minimum preset value is zero; the maximum present value is +32,767 counts. The minimum current value is -32,768. When reset, the current value of the counter is set to the preset value PV. When the enable input transitions from OFF to ON, the current value is decremented by one. The output is ON whenever the current value is less than or equal to zero. The current value of the DNCTR is retentive on power failure; no automatic initialization occurs at power-up.

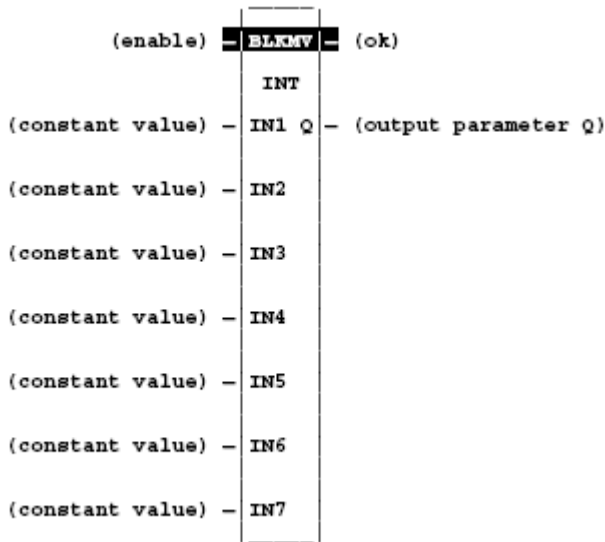


Parameter	Description
address	<p>The DNCTR uses three consecutive words (registers) of %R memory to store the following:</p> <ul style="list-style-type: none"> <li>Current value (CV) = word 1.</li> <li>Preset value (PV) = word 2.</li> <li>Control word = word 3.</li> </ul> <p>When you enter an DNCTR, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function.</p> <p><b>Note:</b> Do not use this address with another down counter, up counter, or any other instruction or improper operation will result.</p> <p><b>Caution:</b> Overlapping references will result in erratic operation of the counter.</p>
enable	On a positive transition of enable, the current value is decremented by one.
R	When R receives power flow, it resets the current value to the preset value.
PV	PV is the value to copy into the counter's preset value when the counter is enabled or reset.
Q	Output Q is energized when the current value is less than or equal to zero.

## BLKMOV:

Use the Block Move (BLKMOV) function to copy a block of seven constants to a specified location. The BLKMOV function has eight input parameters and two output parameters. When the function receives power flow, it copies the constant values into consecutive locations, beginning at the destination specified in output Q. Output Q cannot be the input of another program function.

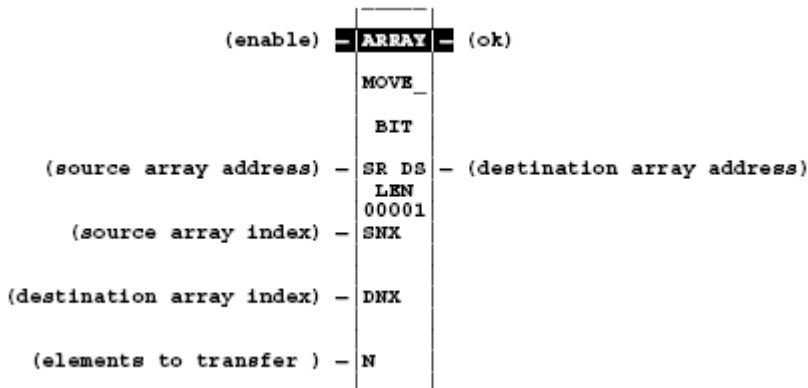




Parameter	Description
enable	When the function is enabled, the block move is performed.
IN1— IN7	IN1 through IN7 contain seven constant values.
ok	The ok output is energized whenever the function is enabled.
Q	Output Q contains the first integer of the moved array. IN1 is moved to Q.

### ARRAY\_MOVE:


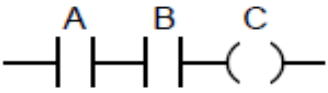

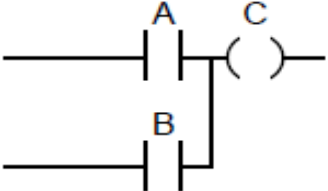
Use the Array Move (ARRAY\_MOVE) function to copy a specified number of data elements from a source array to a destination array. The ARRAY\_MOVE function has five input parameters and two output parameters. When the function receives power flow, the number of data elements in the count indicator (N) is extracted from the input array starting with the indexed location (SR + SNX — 1). The data elements are written to the output array starting with the indexed location (DS + DNX — 1). The LEN operand specifies the number of elements that make up each array.

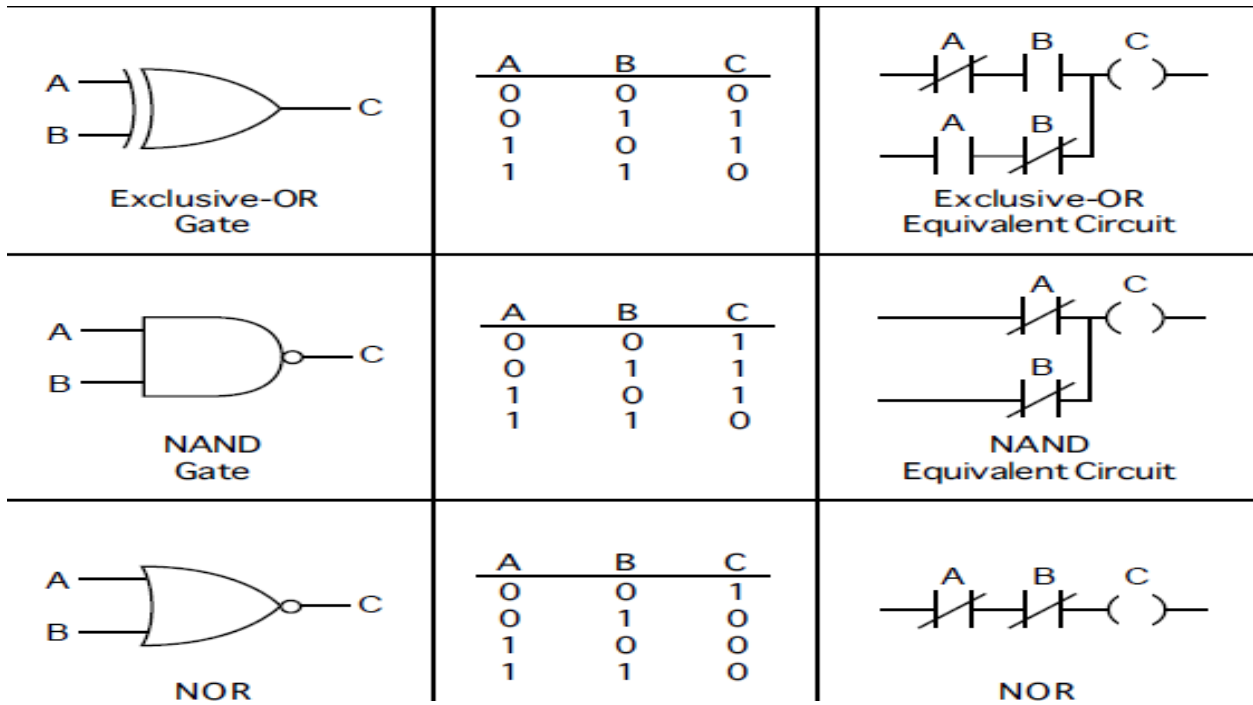


Parameter	Description
enable	When the function is enabled, the operation is performed.
SR	SR contains the starting address of the source array. For ARRAY_MOVE_BIT, any reference may be used; it does not need to be byte aligned. However, 16 bits, beginning with the reference address specified, are displayed online.
SNX	SNX contains the index of the source array.
DNX	DNX contains the index of the destination array.
N	N provides a count indicator.
ok	The ok output is energized whenever enable is energized.
DS	DS contains the starting address of the destination array. For ARRAY_MOVE_BIT, any reference may be used; it does not need to be byte aligned. However, 16 bits, beginning with the reference address specified, are displayed online.
LEN	LEN specifies the number of elements starting at SR and DS that make up each array.

### Verification of logic gates:

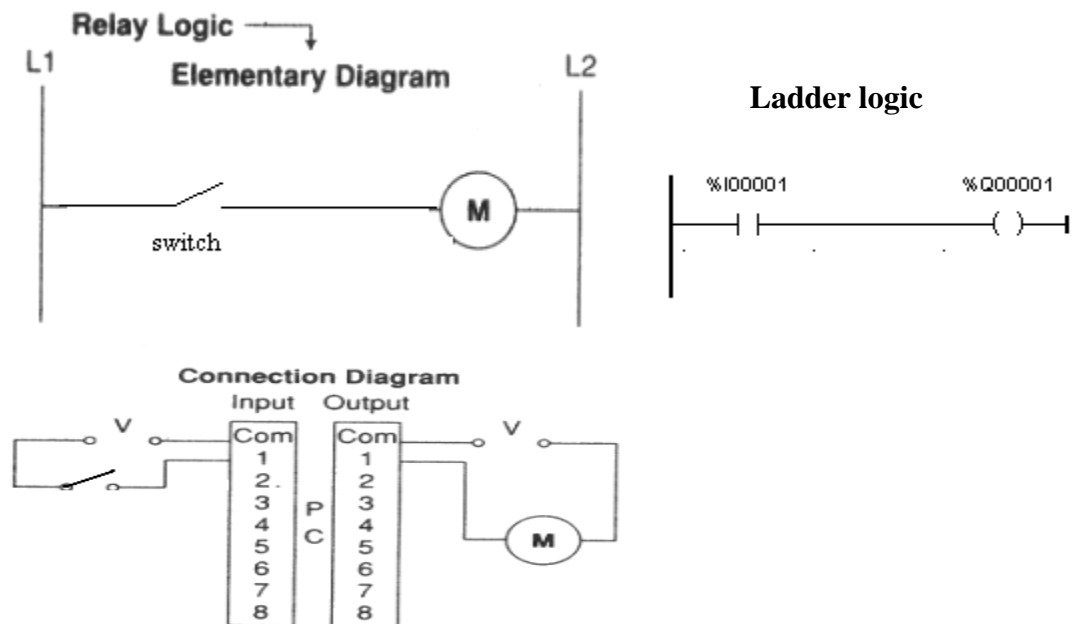
## EQUIVALENT LADDER/LOGIC DIAGRAMS

Logic Diagram	Truth Table	Ladder Diagram															
 <p>AND Gate</p>	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	C	0	0	0	0	1	0	1	0	0	1	1	1	 <p>AND Equivalent Circuit</p>
A	B	C															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
 <p>OR Gate</p>	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	C	0	0	0	0	1	1	1	0	1	1	1	1	 <p>OR Equivalent Circuit</p>
A	B	C															
0	0	0															
0	1	1															
1	0	1															
1	1	1															



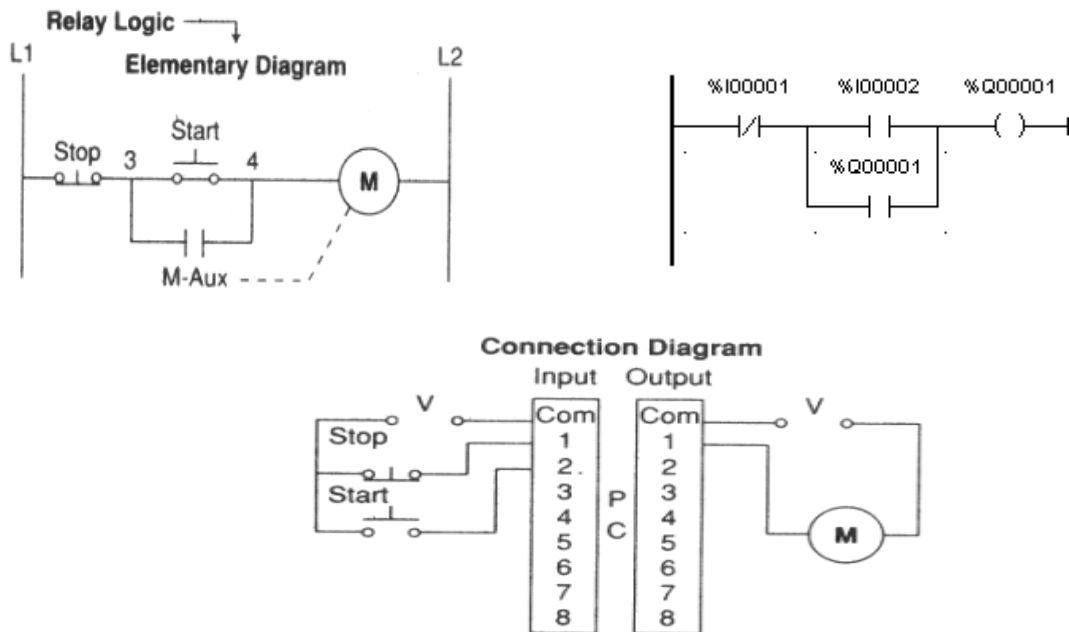
### Example:1

A simple circuit with one switch as a contact and one output as a coil. As the switch is opened or closed, the output goes on or off.



## Example:2

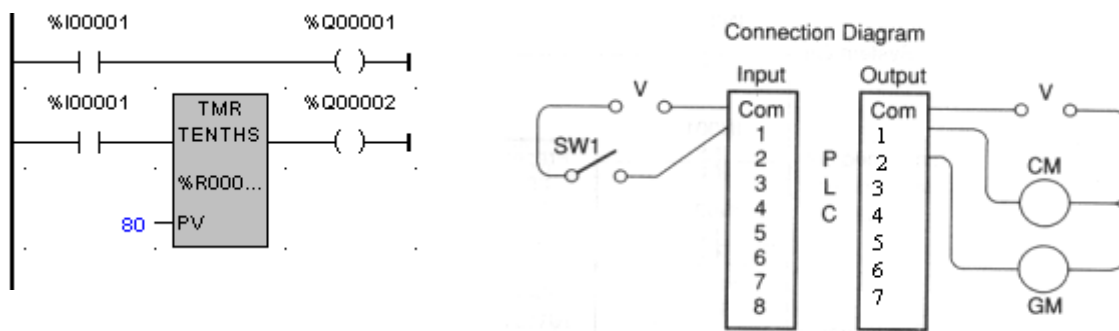
When the start button is depressed, the coil energizes. When the button is released, the coil remains on.



## Example:3

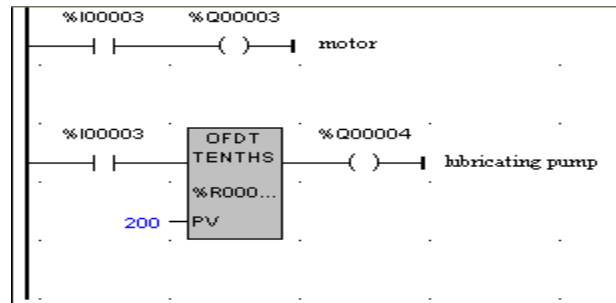
For a grinding operation on a metal part, the coolant flow on a part must be on for an interval before the grinding process starts. When the process circuit is turned on, the coolant motor is turned on. Eight seconds later the grinding process starts.

### Ladder logic



### Example:4

It is an off delay circuit. A motor and its lubrication pump motor are running. Lubricating pump remains ON for 20 secs after the main motor is shut off.



### Result:

Thus the Programmable Logic controller (PLC) has been studied and its various logic operations are verified.

## 2. IMPLEMENTATION OF CODE CONVERTORS (Binary to Gray & Gray to Binary)

### **Aim:**

To implement the following code convertors(i) binary to gray (ii) gray to binary

### **Apparatus Required:**

1. PC
2. Ge Fanuc PLC
3. Versa Pro Software

### **Procedure:**

1. Double click on the Versa pro icon.
2. Create a new folder from File- New folder- Next- Finish.
3. In the new folder create the ladder logic as shown in the diagram.
4. Save the file.
5. From the PLC menu, select Connect to connect the PC to the PLC.
6. Store the program in the PLC by selecting PLC- Store- Accept.
7. Load the program in the PLC by selecting PLC- Load- OK.
8. Run the program.
9. Give the inputs and verify the outputs.
10. To make changes, select PLC- Clear- Select All- OK and clear the program from the PLC memory

**(i) Binary to gray code**

SL.NO	BINARY CODE				GRAY CODE			
	B <sub>4</sub>	B <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>	G <sub>4</sub>	G <sub>3</sub>	G <sub>2</sub>	G <sub>1</sub>
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

B1-%I00001

G1-%Q00001

B2-%I00002

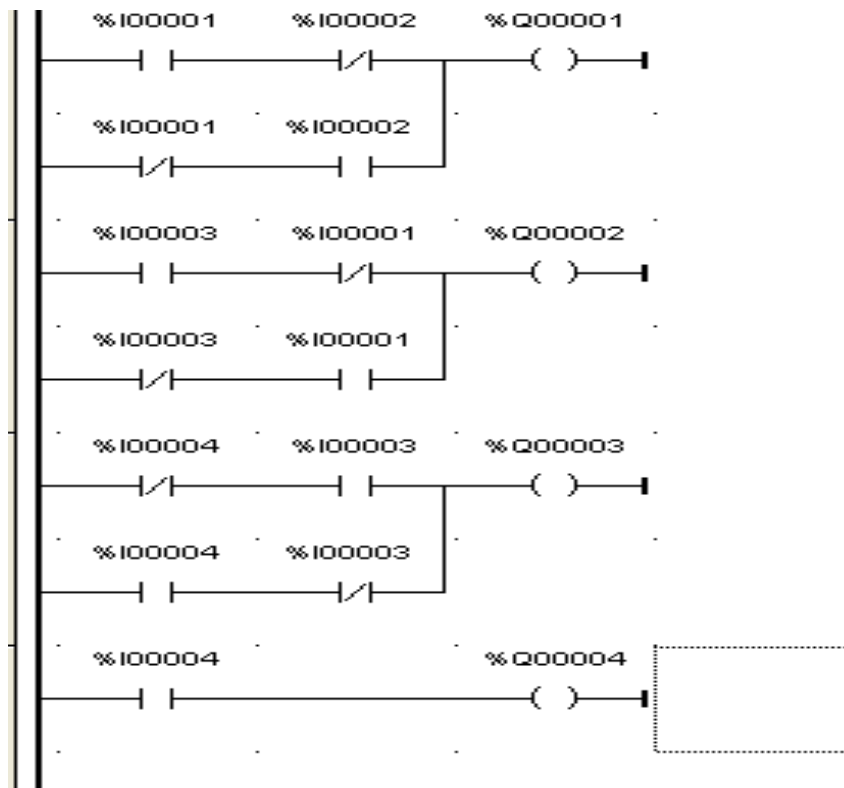
G2-%Q00002

B3-%I00003

G3--%Q00003

B4-%I00004

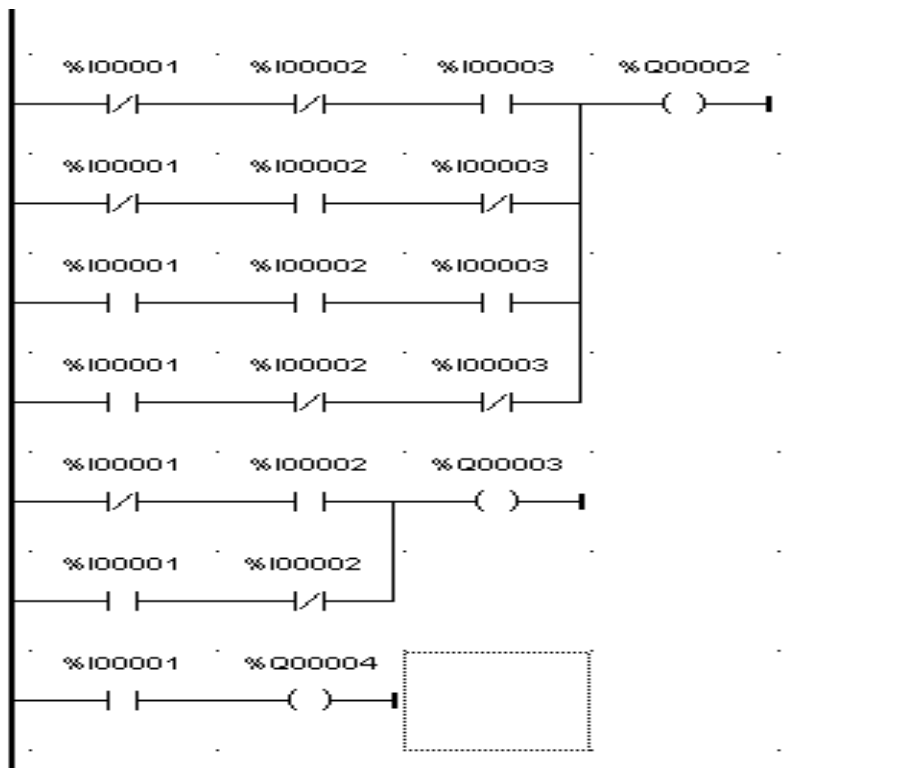
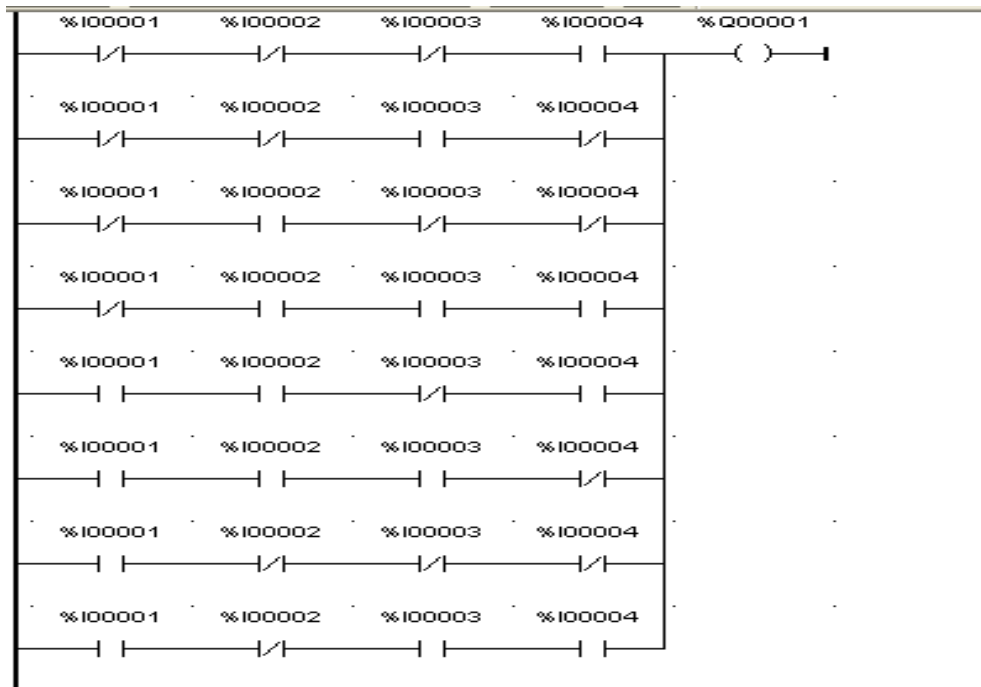
G4--%Q00004





**(ii) Gray to binary code**

GRAY CODE				BINARY CODE			
G <sub>4</sub>	G <sub>3</sub>	G <sub>2</sub>	G <sub>1</sub>	B <sub>4</sub>	B <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1



**Result:**

Thus the code convertors are verified using ladder logic programming.

### 3. TRAFFIC LIGHT CONTROL SYSTEM

**Aim:**

To implement a one way and two way traffic light control system.

**Apparatus Required:**

- 4. PC
- 5. Ge Fanuc PLC
- 6. Versa Pro Software

**Procedure:**

- 11. Double click on the Versa pro icon.
- 12. Create a new folder from File- New folder- Next- Finish.
- 13. In the new folder create the ladder logic as shown in the diagram.
- 14. Save the file.
- 15. From the PLC menu, select Connect to connect the PC to the PLC.
- 16. Store the program in the PLC by selecting PLC- Store- Accept.
- 17. Load the program in the PLC by selecting PLC- Load- OK.
- 18. Run the program.
- 19. Give the inputs and verify the outputs.
- 20. To make changes, select PLC- Clear- Select All- OK and clear the program from the PLC memory

**ONE WAY DIRECTION:**

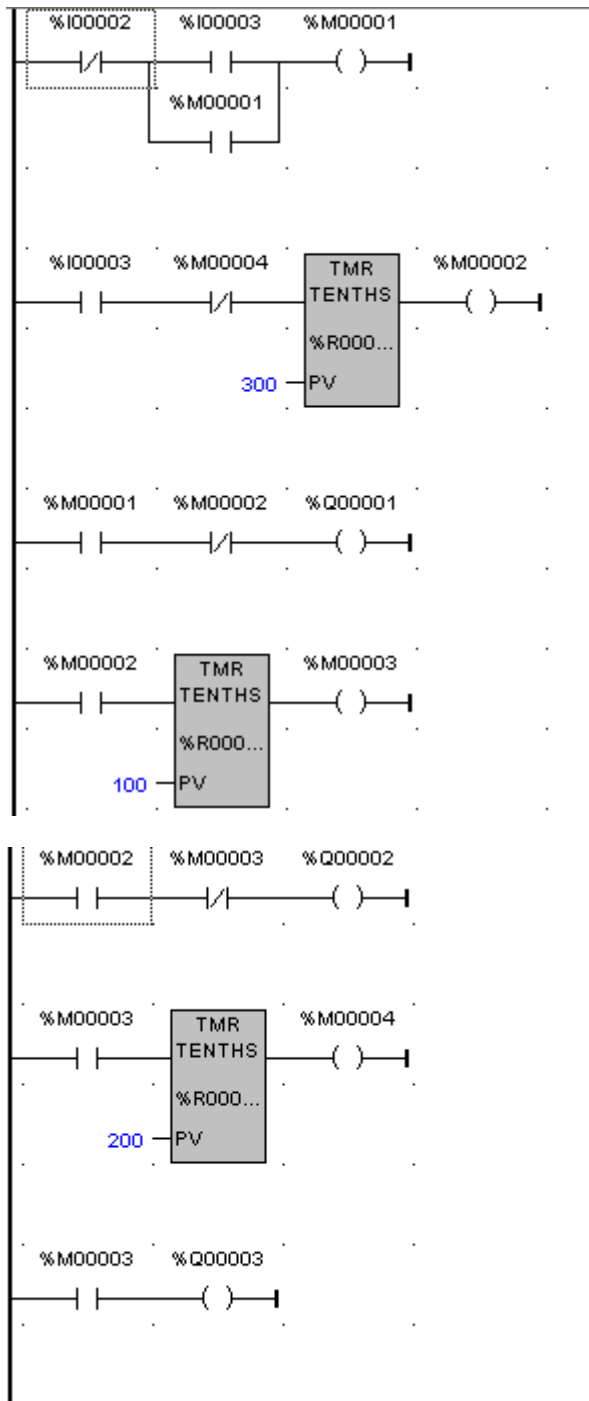
**Problem statement:**

Red light glows for 30s, yellow light for 5 s, green light for 25s and this process continues. There is a start button and a stop button. Draw the ladder logic to implement the above process.

**Flow diagram:**



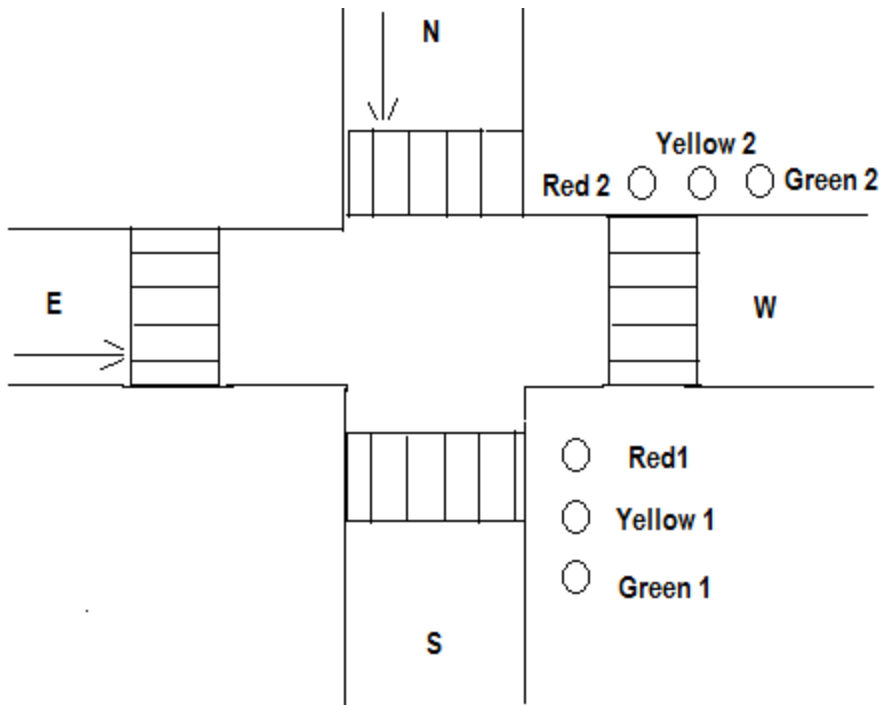
Ladder Logic:



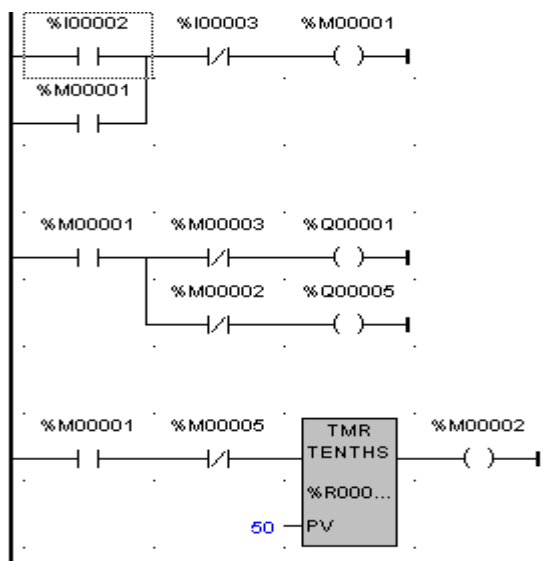
## TWO WAY DIRECTION:

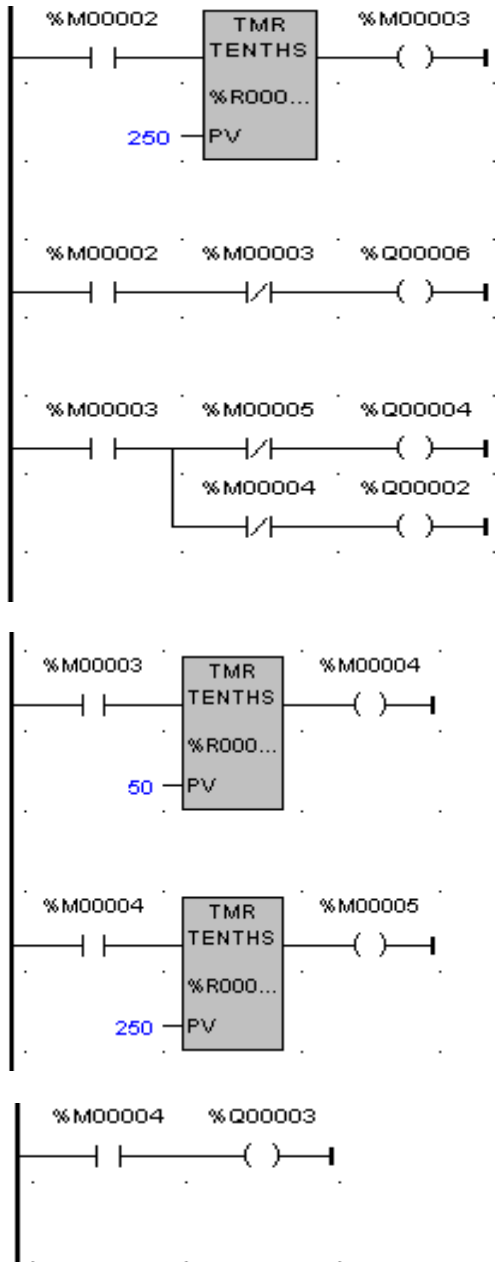
When Red 1 is ON for 30s, E-W traffic flows. At this instant, Yellow 2 becomes ON for 5s after which Green 2 becomes ON for 25s. After 30s, when Red 2 is ON for 30s, N-S traffic flows. At this instant, Yellow 1 becomes ON for 5s after which Green 1 becomes ON for 25s and this process continues. There is a start button and a stop button. Draw the ladder logic to implement the above process.

### Flow diagram:



### Ladder Logic:





**Result:**

Thus the one way and two way traffic light control was executed and verified using Ge Fanuc

## 4. WATER LEVEL CONTROL SYSTEM

### AIM:

To design a ladder logic program to maintain the water level in a tank.

### APPARATUS REQUIRED:

1. VPLCT - 01 [PLC Trainer kit].
2. Water level indicator [Float switches - 2].
3. PC.
4. RS - 232 cable.
5. Patch Chords.

### PROCEDURE:

1. Load the Versapro Software to the PC.
2. Open the Versapro Software.
3. Switch ON the PLC Trainer and water level system.
4. Connect PLC and water level System kit.
5. Open the New folder and draw the ladder logic program.
6. Connect the PLC to PC.
7. Select the correct hardware Configuration [Sl. No].
8. Store the Program to PLC.
9. Run the Program.
10. Verify the performance of the water level System.

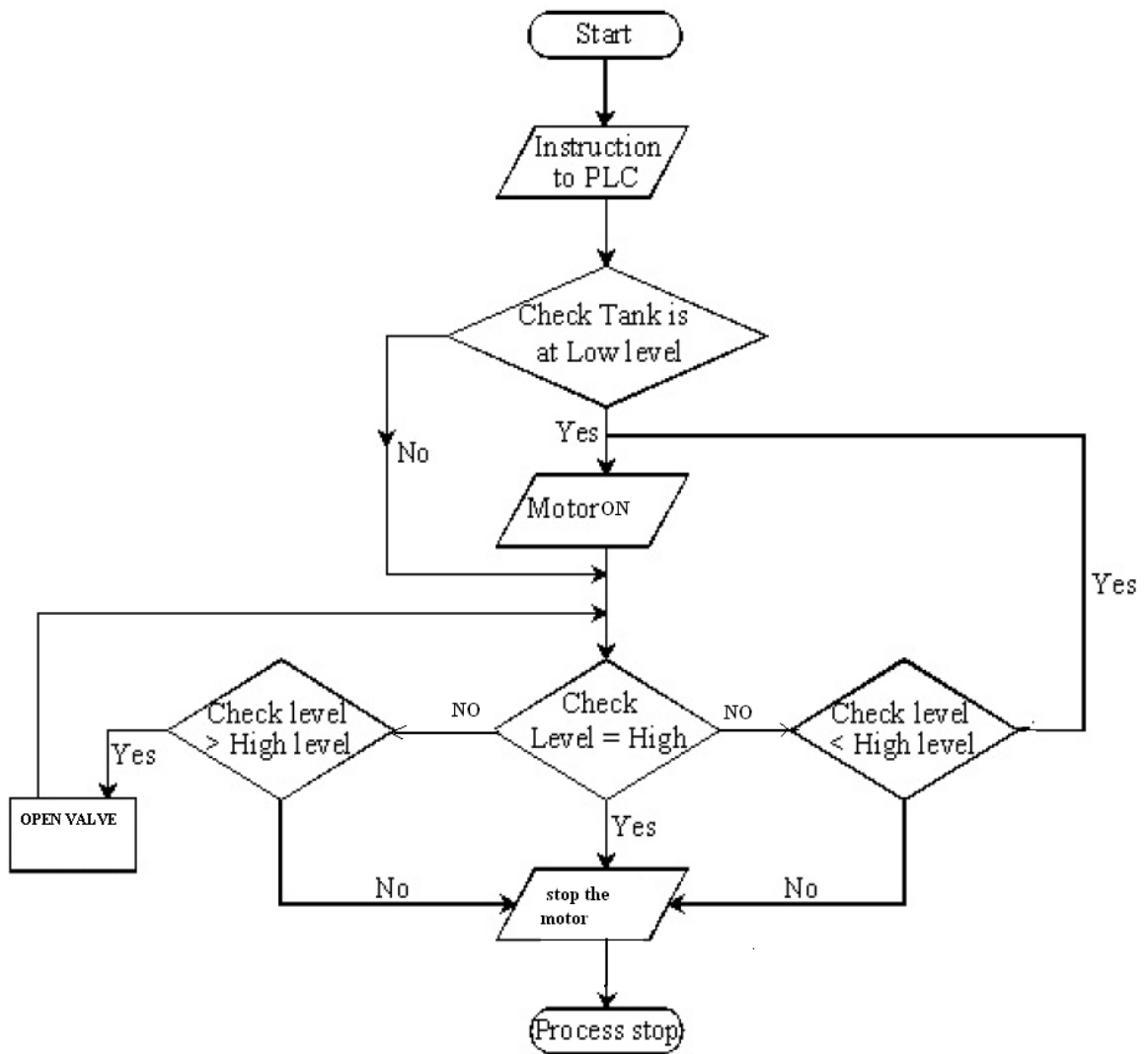
### PROBLEM STATEMENT:

Water Level controller is used to Fill/drain the tank, which is depends upon high/low level in tank water. If the tank is in low level the solenoid valve will open & motor starts - ON Position if the tank is in high level, the solenoid valve will be close & motor stop - OFF position.

### BLOCK DIAGRAM:

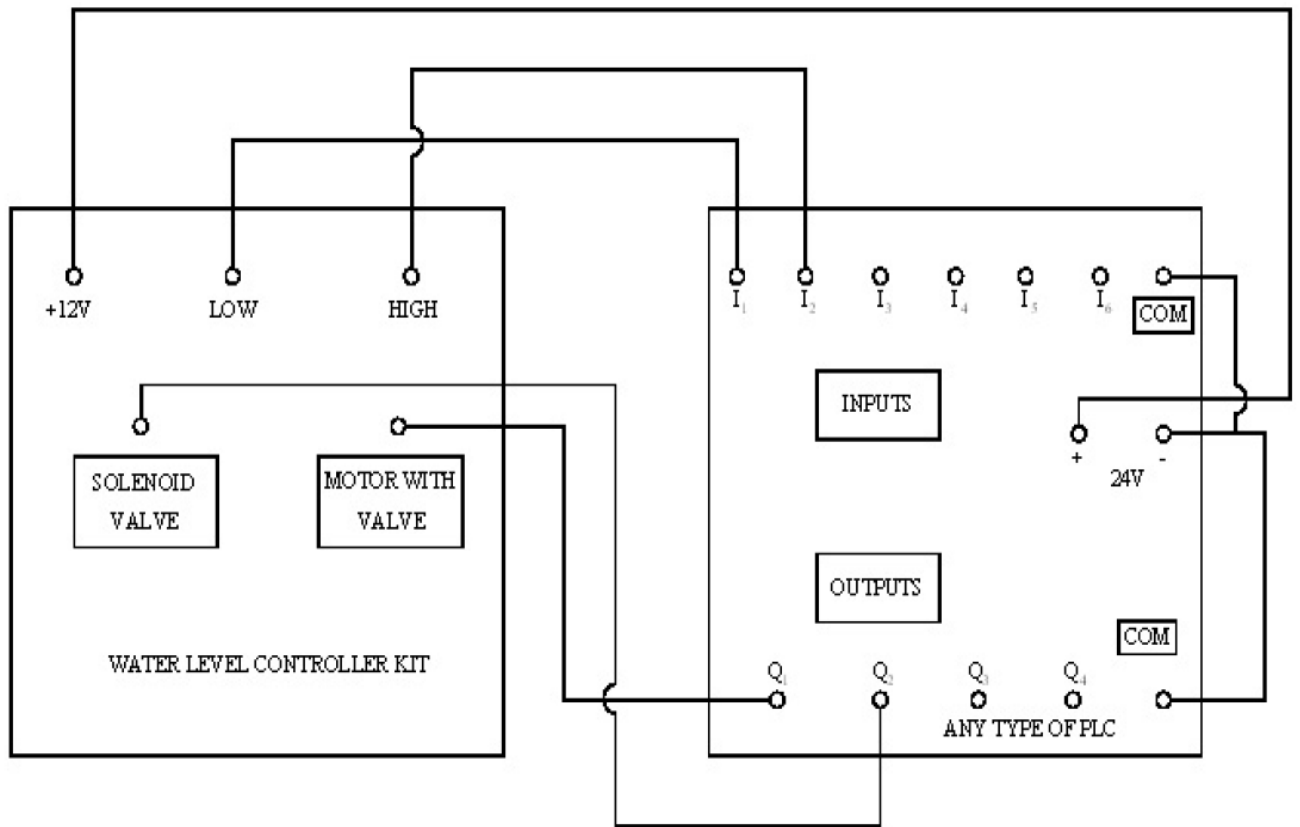


# FLOWCHART:

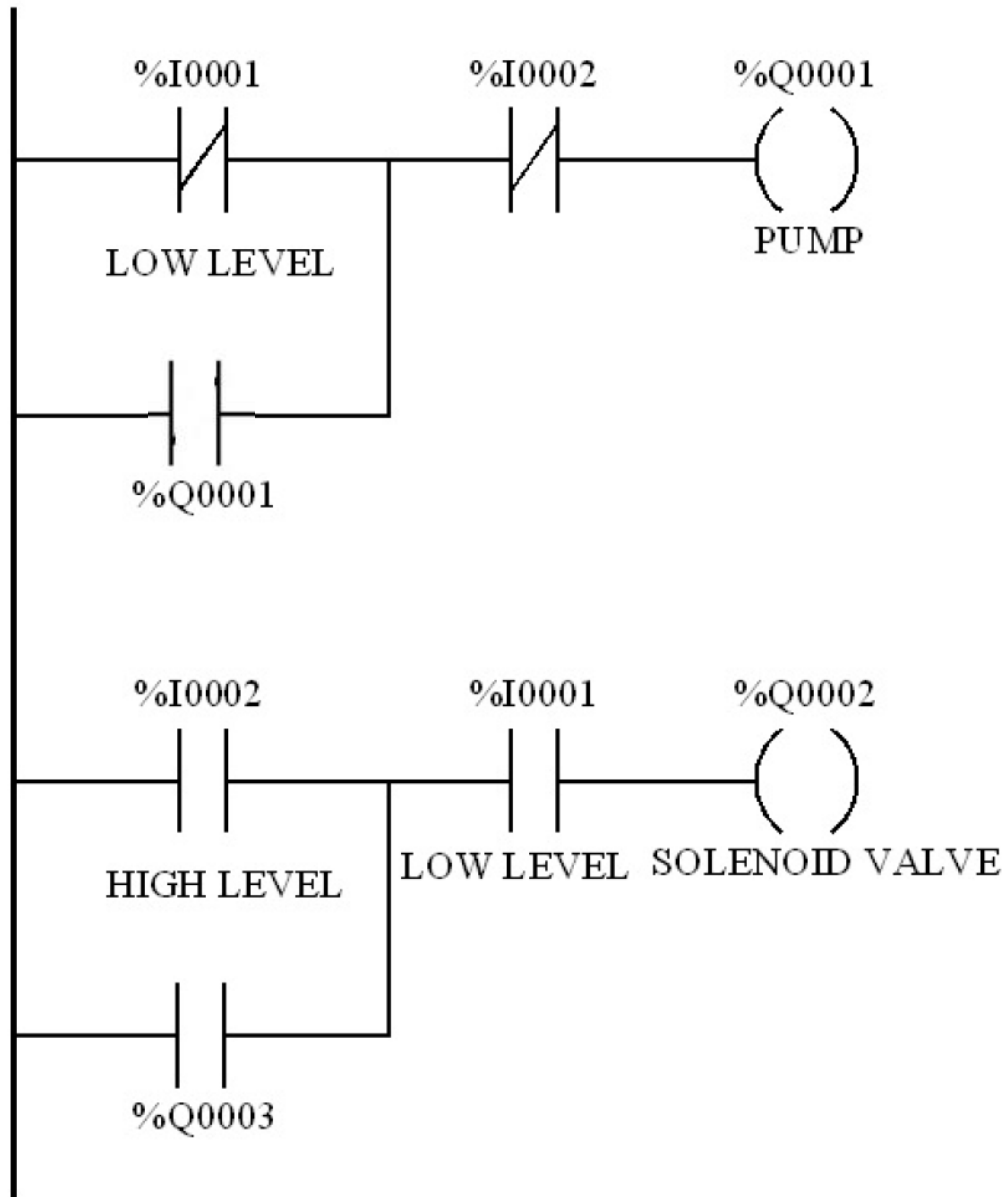




## WIRING DIAGRAM:



## LADDER DIAGRAM:



## RESULT:

Thus the ladder logic program to maintain the water level in a tank has been designed and tested using the water level kit.

## 5. MATERIAL HANDLING SYSTEM

### AIM:

To develop a ladder logic program for the material handling system.

### APPARATUS REQUIRED:

1. Material Handling system.
2. PLC
3. Versapro Software
4. PC
5. RS-232 Cable.
6. Patch Chords

### PROCEDURE:

1. Load the Versapro Software to the PC.
2. Open the Versapro Software.
3. Switch ON the PLC Trainer and Material handling system.
4. Connect PLC and Material handling System kit.
5. Open the New folder and draw the ladder logic program.
6. Connect the PLC to PC.
7. Select the correct hardware Configuration [Sl. No].
8. Store the Program to PLC.
9. Run the Program.
10. Verify the performance of the Material handling System.

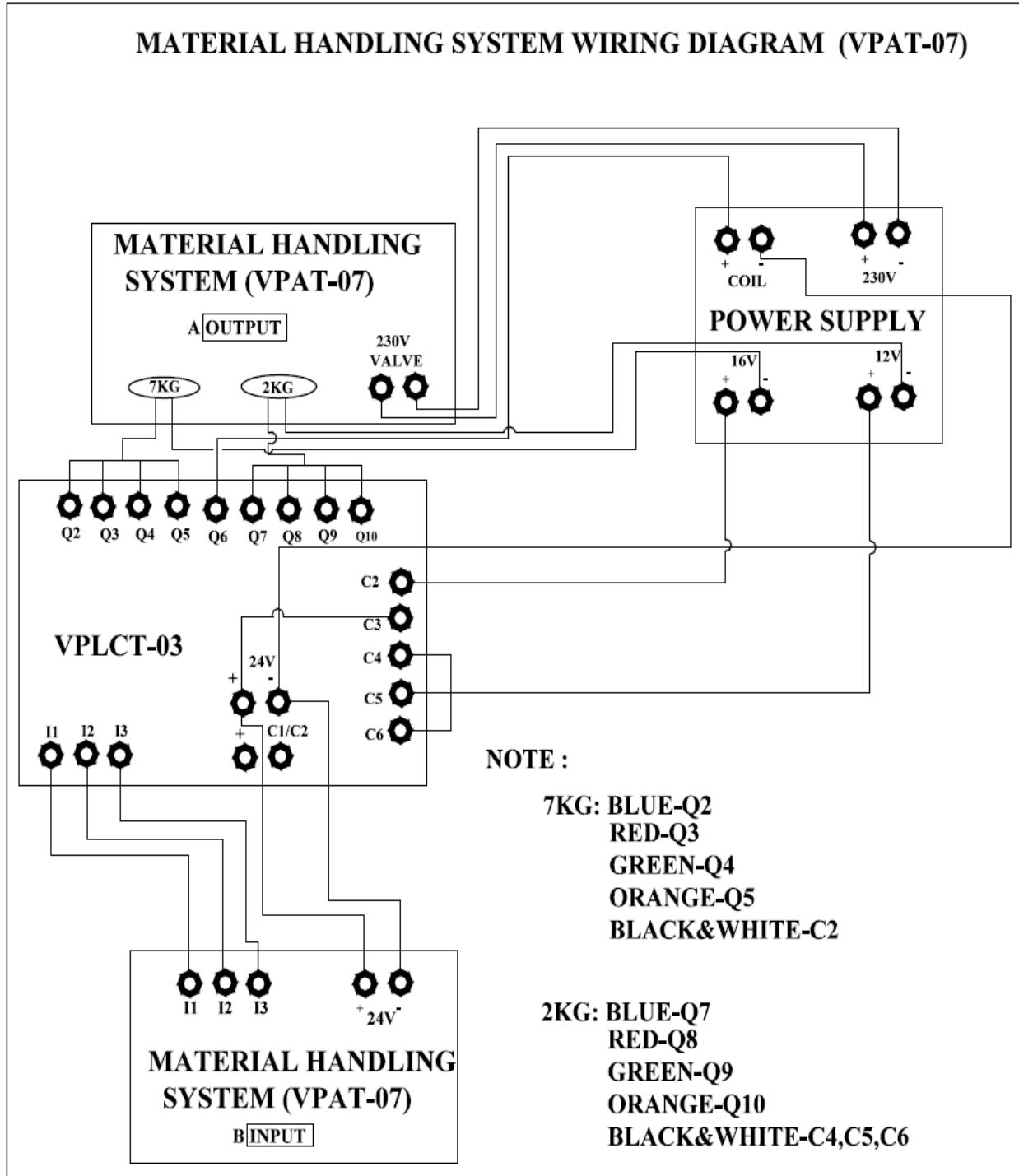
### Problem statement:

In the material handling system first the arm is go to the initial position that means it is placed in the(i1) limit switch Point . Now the program in run the conveyer settings is role whenever the sensor (i3) is high at the time the conveyer setting action in stop now the arm is rotate the initial point to (i2) position. Whenever the input (i2) is high at the time the coil output is high now the coil is magnetized and to catch the material and rotate the reverse direction. Whenever as the arm is touch the I/P (i1) at the time the I/P 1 is high the coil O/P in low. So, the coil is demagnetized and the material is drop in the storage box.

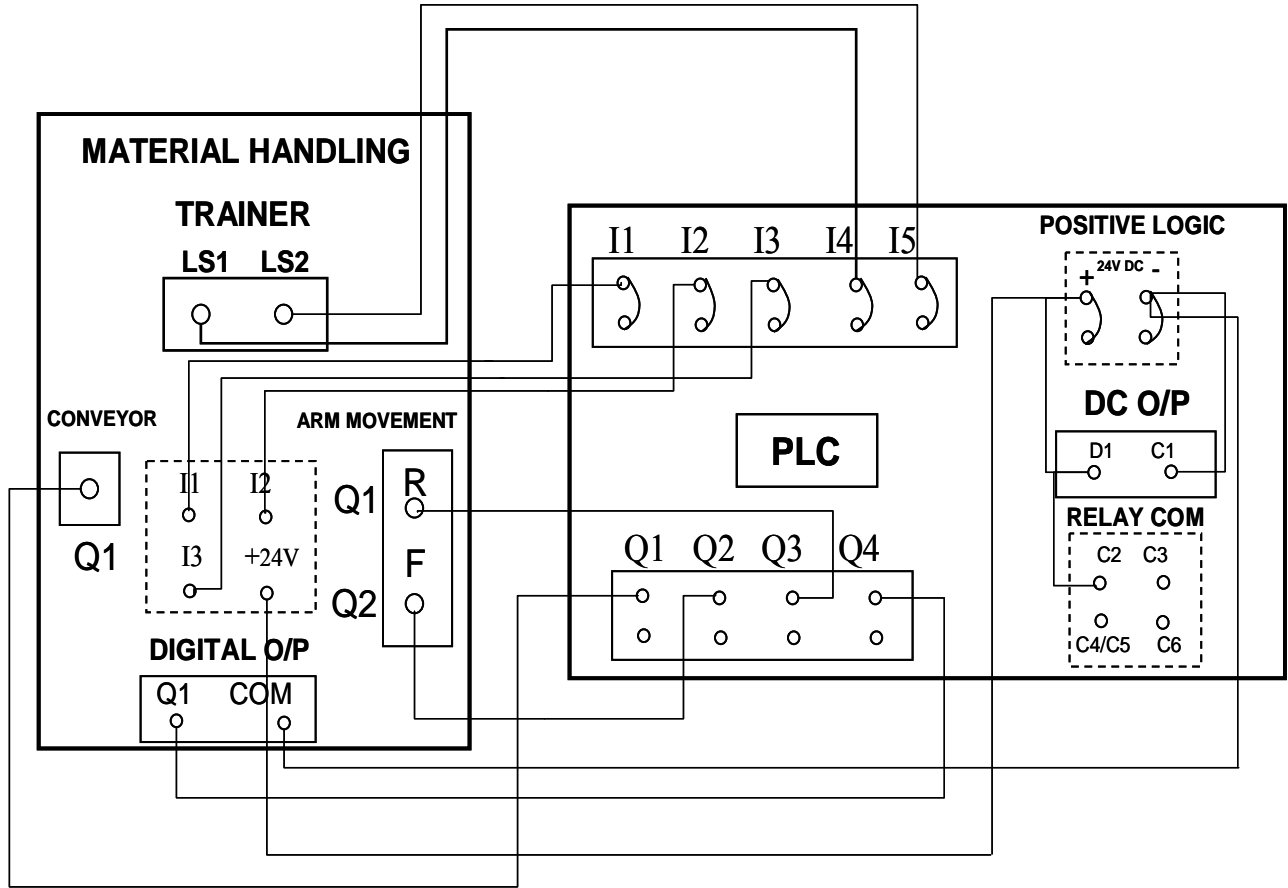
The material handling system consists of the following interfaced to a micro PLC.

- (i) Stepper motor 1- To rotate a container belt
- (ii) Stepper motor 2- To move a arm
- (iii) Electromagnet - To pick a material (It is mount in arm end)
- (iv) Limit switches - To Control the arm position

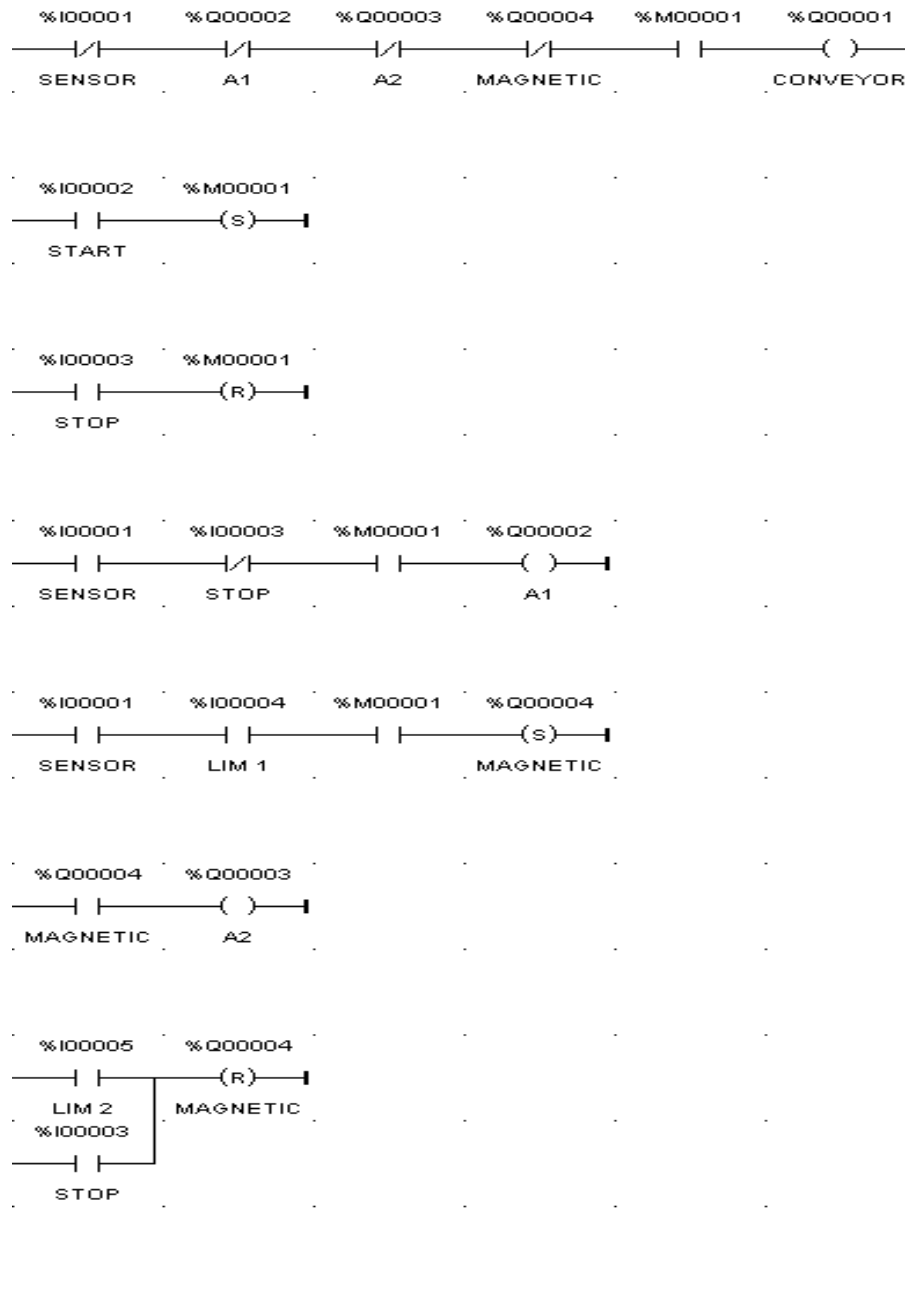
**WIRING DIAGRAM:**



**WIRING DIAGRAM:**



## LADDER DIAGRAM:



## RESULT:

Thus the ladder logic program for the material handling system is developed and tested using hardware kit.

## 6. BOTTLE FILLING SYSTEM

### AIM:

To design a ladder logic program to perform Bottle Filling System.

### APPARATUS REQUIRED:

1. Bottle filling system.
2. PLC
3. Versapro Software
4. PC
5. RS-232 Cable.
6. Patch Chords

### PROCEDURE:

1. Load the Versapro Software to the PC.
2. Open the Versapro Software.
3. Switch ON the PLC Trainer and Bottle filling system.
4. Connect PLC and Bottle Filling System kit.
5. Open the New folder and draw the ladder logic program.
6. Connect the PLC to PC.
7. Select the correct hardware Configuration [Sl. No].
8. Store the Program to PLC.
9. Run the Program.
10. Verify the performance of the Bottle Filling System.

### PROGRAM FOR BOTTLE FILLING SYSTEM:

I<sub>1</sub> = Input from proximity (Capacitive Pickup) sensor - Bottle indicator.

I<sub>2</sub> = Input from position sensor.

I<sub>3</sub> = Input from low level sensor in process tank.

I<sub>4</sub> = Input from high level sensor in process tank.

Q<sub>1</sub>, Q<sub>2</sub>, Q<sub>3</sub>, Q<sub>4</sub> = Output for stepper motor coil.

Q<sub>5</sub> = Output for drives a pump.

Q<sub>6</sub> = Output for drive a solenoid valve.

A motor (pump) is used to fill the process tank from reservoir. The motor is switched ON and OFF by two sensors which are placed in the process tank at two levels [LL & HL].

When the water level is below to the low level, both of the sensors output are low (I<sub>3</sub> / I<sub>4</sub>).

So the motor is switched ON and water is poured into the process tank from reservoir.

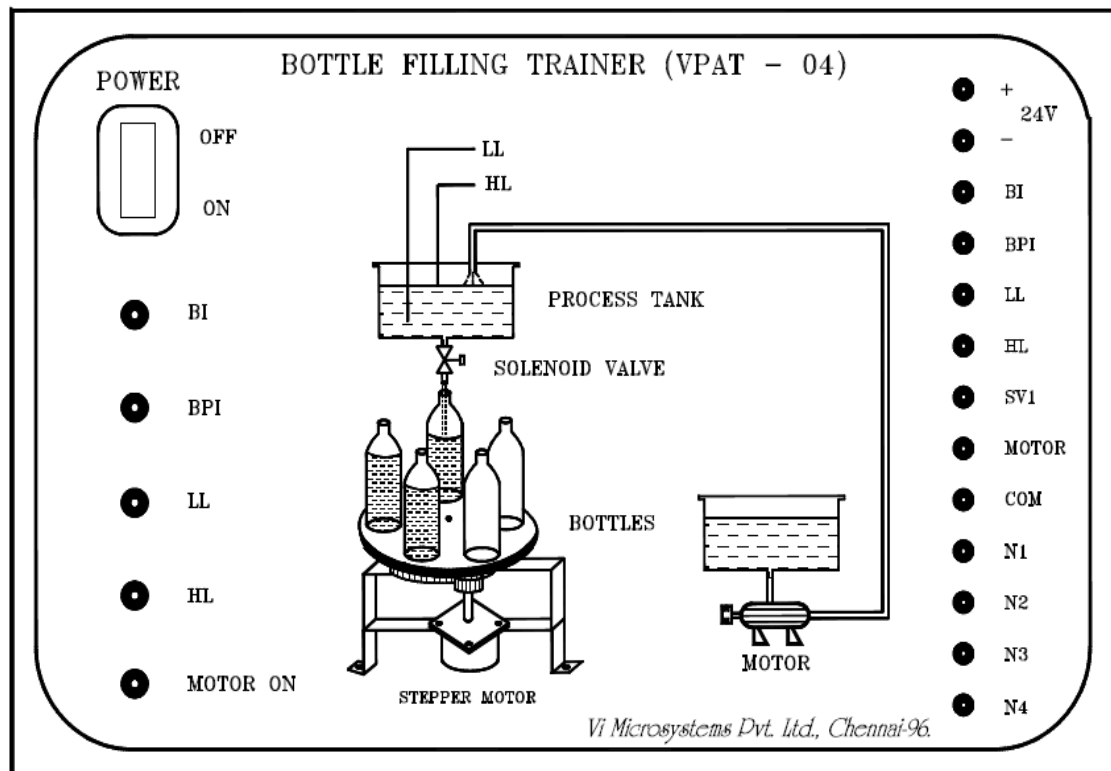
Even though the water level is increased above the low level sensor, the motor is at running condition, because of a feedback connected parallel to the low level sensor (I<sub>3</sub>).

When the water level reaches the high level sensor (I<sub>4</sub>) the motor is turned Off as high level sensor broke the connection. When I<sub>1</sub>, I<sub>2</sub> are high then M1 (Internal memory coil) energised which in turn energise solenoid valve Q<sub>6</sub>. The output for the Q<sub>6</sub> is for specified

timing, which is determined by on-delay timer PV (Preset Value). Next step is to rotate stepper motor by energising four coils of it. The starting address of data are stored in sequence as explained below. SNX is to increment the starting register (SR) in array move function. The input to SNX is given through up counter register R1. The timing is decided with the help of on-delay timer present value.

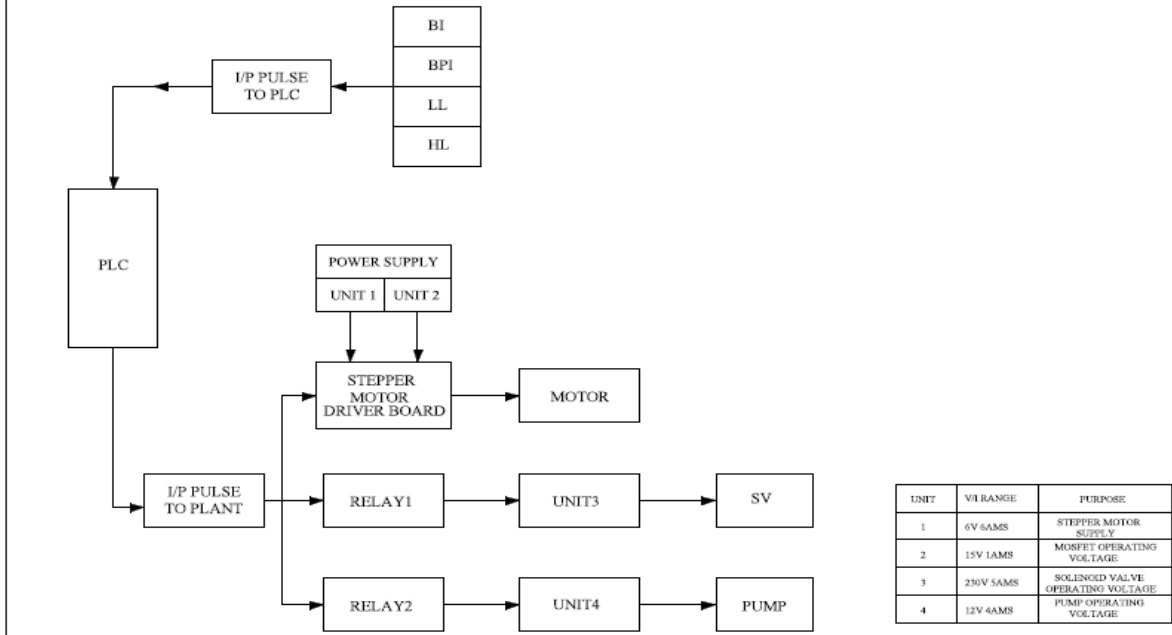
First register data will be given in Q1 and incremented till Q4 which is clearly shown in the table. The rotation of the stepper motor stops when Q6 is low as I1 and I2 are high. The operation continuous in these sequence.

Coil 1	Coil 2	Coil 3	Coil 4	Decimal Value
1	0	0	0	1
0	1	0	0	2
0	0	1	0	4
0	0	0	1	8

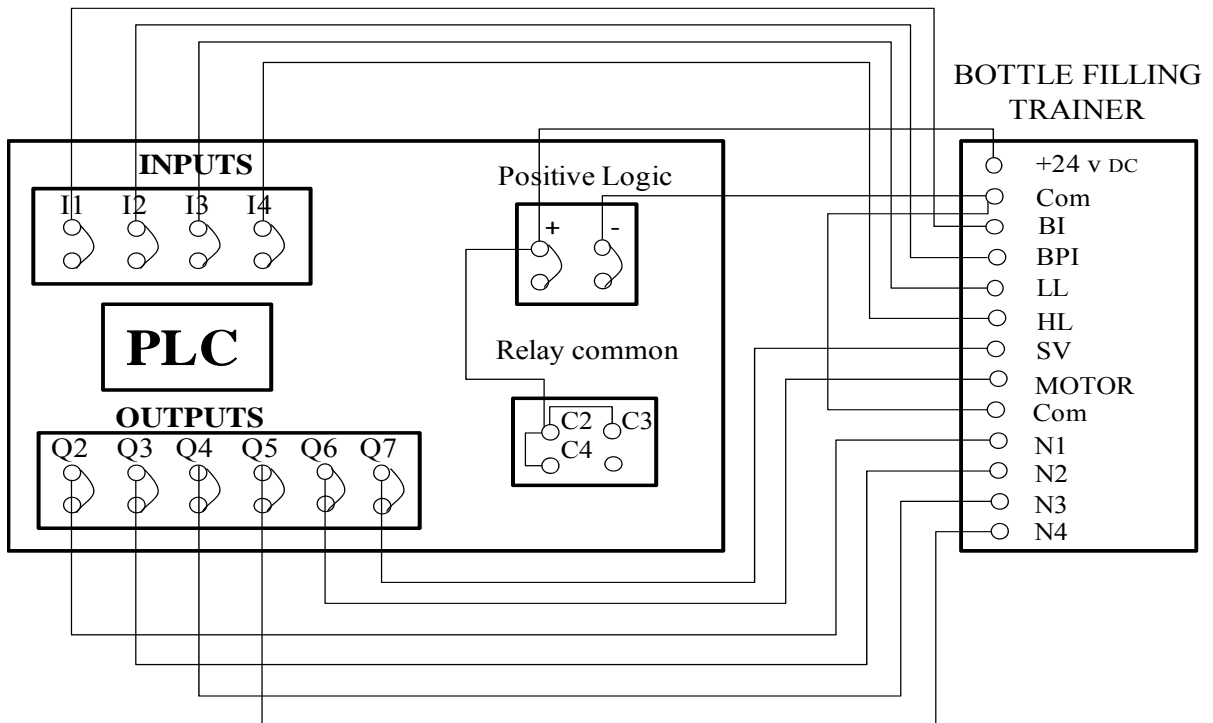




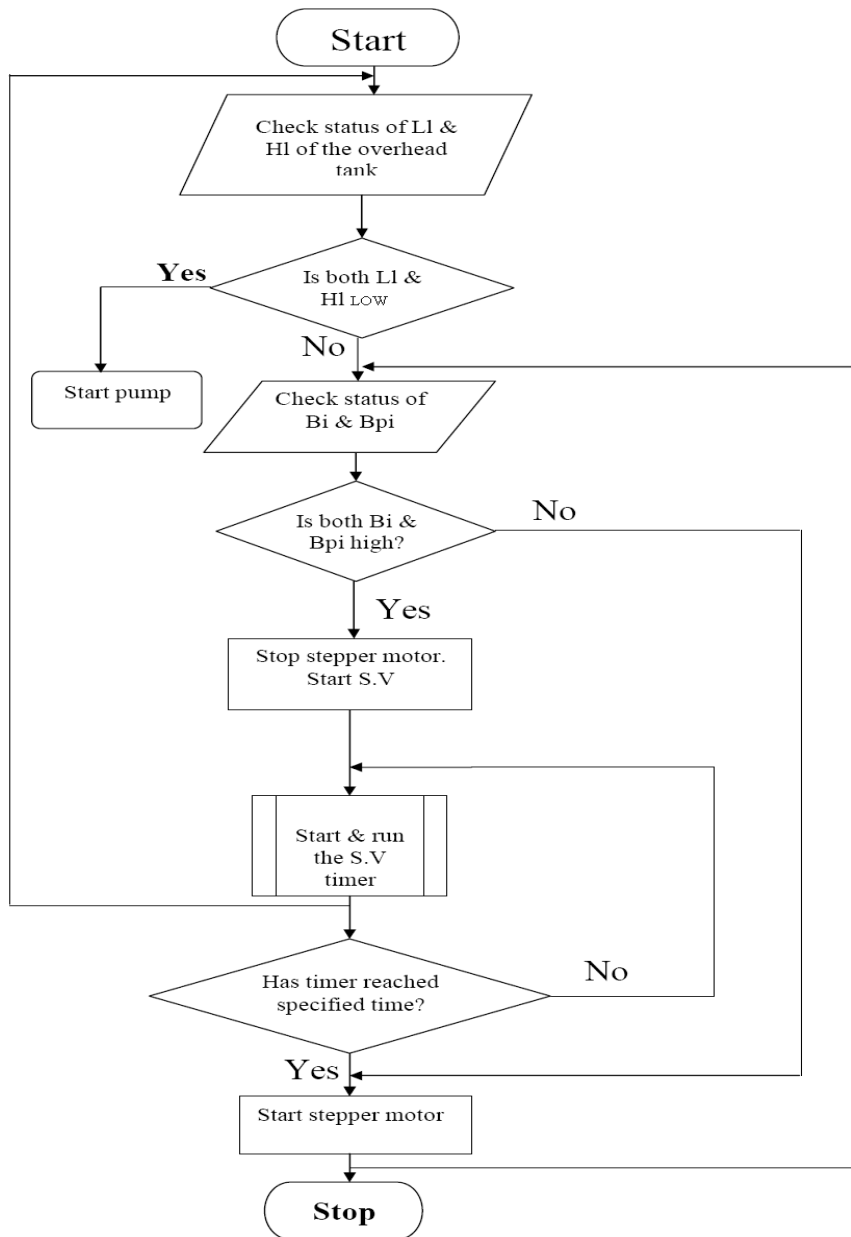
## BOTTLE FILLING SYSTEM BLOCK DIAGRAM



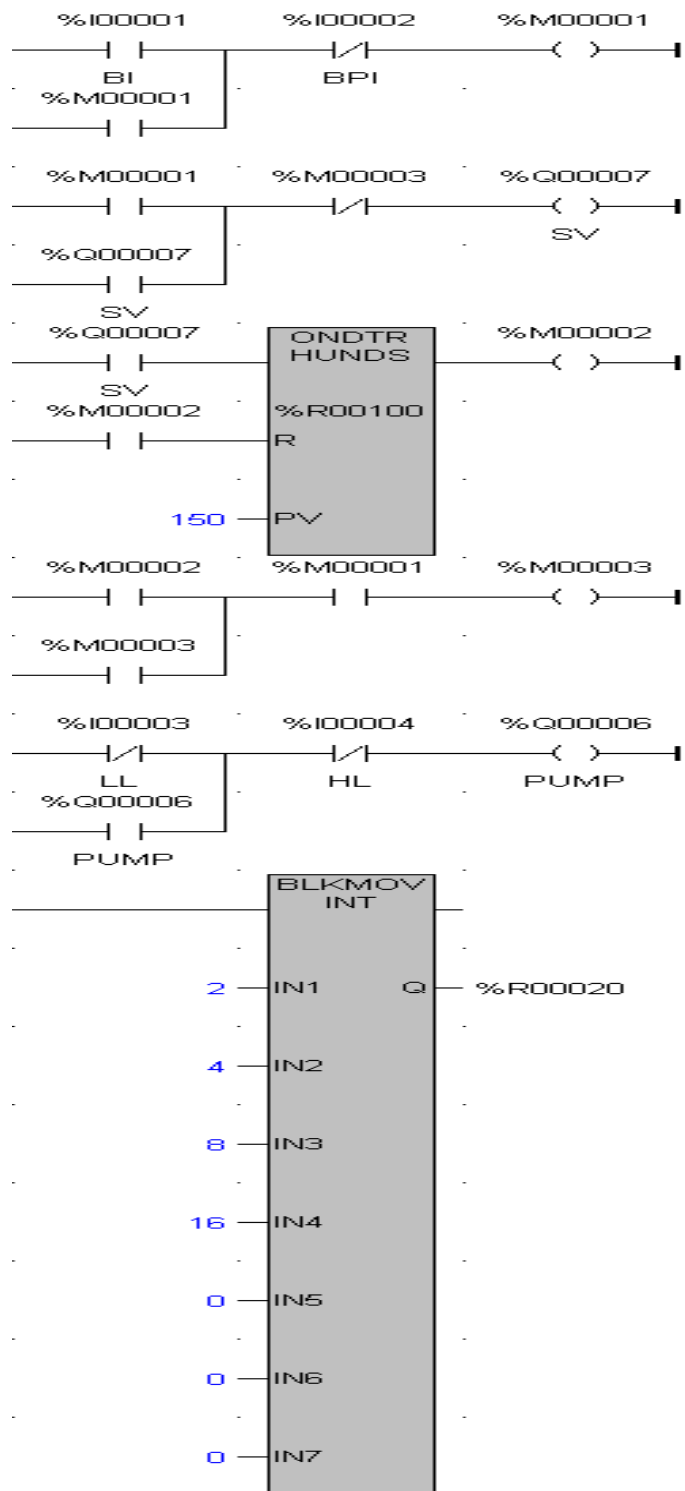
### WIRING DIAGRAM:

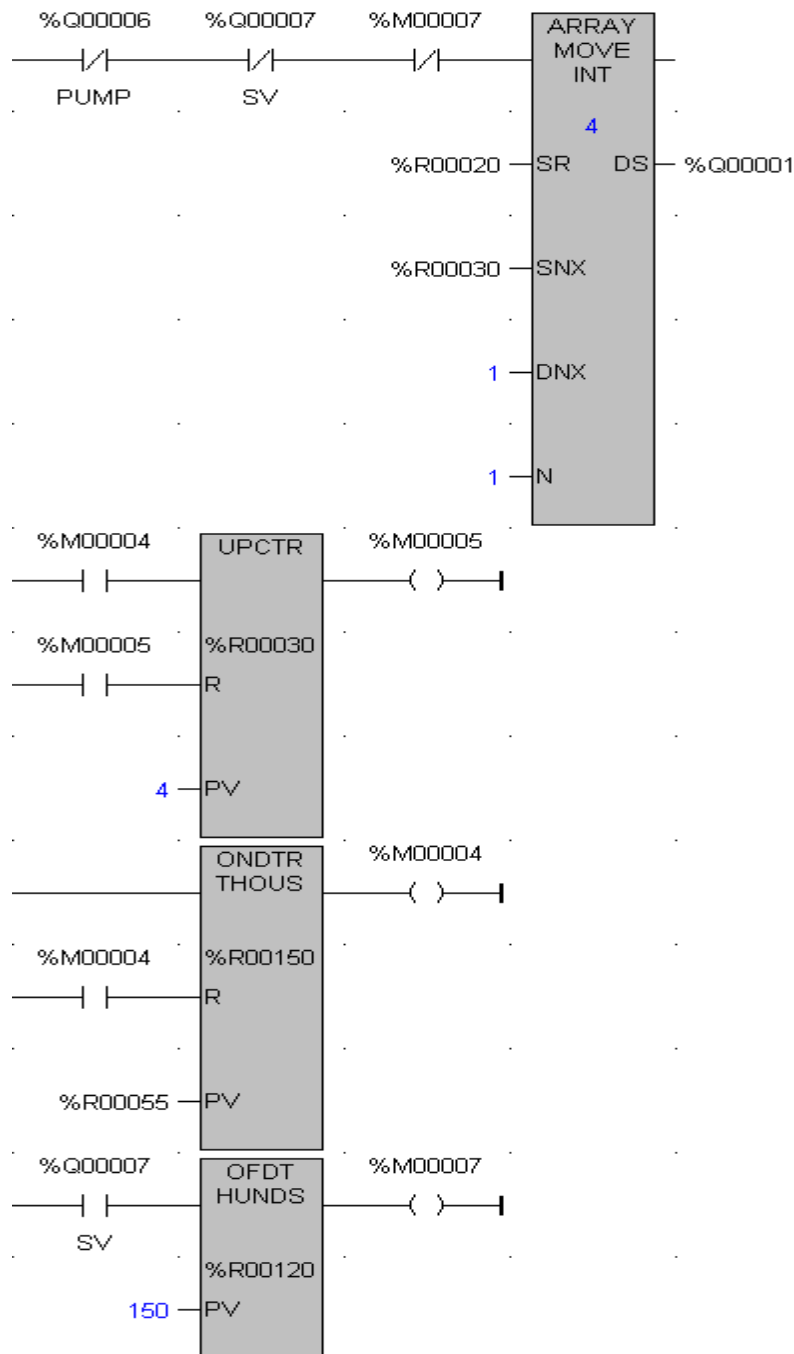


FLOWCHART:



# LADDER LOGIC PROGRAM:





**RESULT:**

Ladder logic program for Bottle Filling System has been designed and the performance is tested using trainer kit.

## 7. SEQUENTIAL OPERATION OF MOTOR

### **AIM:**

To design a ladder logic program to perform the sequential operation of a stepper motor.

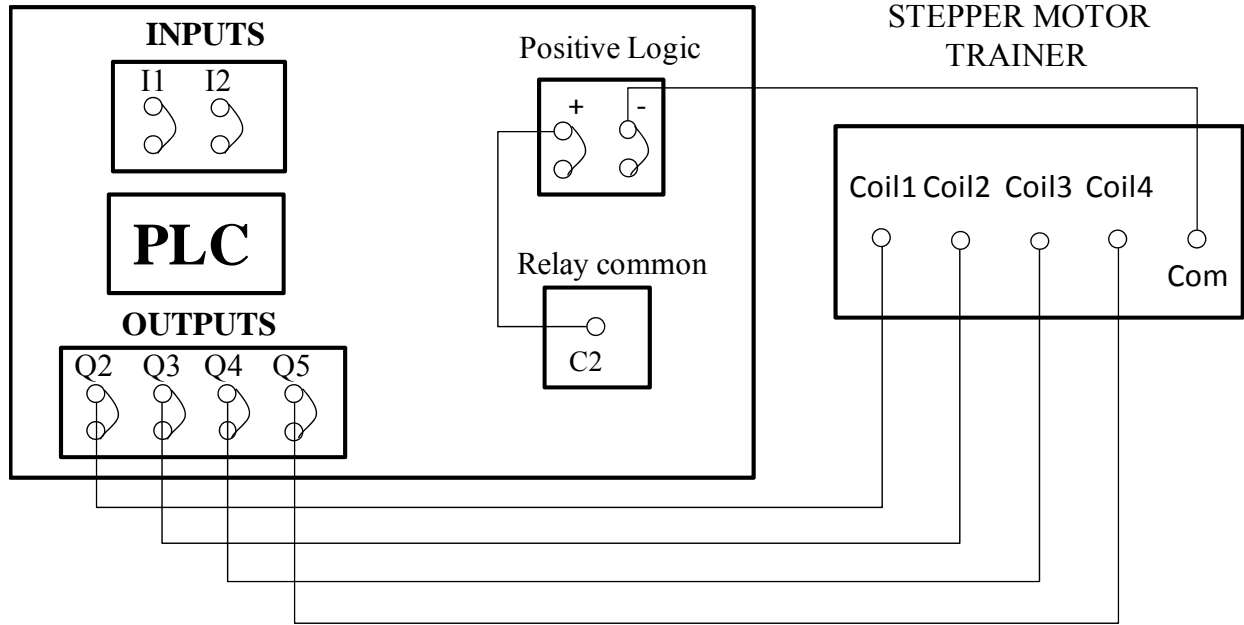
### **APPARATUS REQUIRED:**

1. Stepper motor
2. PLC
3. Versapro Software
4. PC
5. RS-232 Cable.
6. Patch Chords

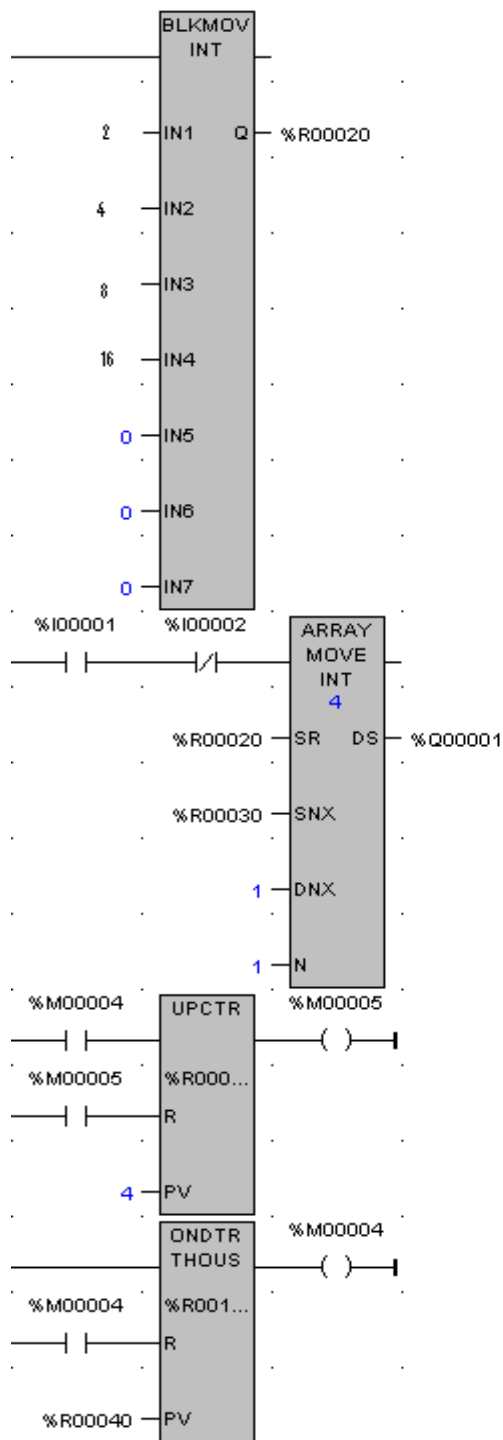
### **PROCEDURE:**

1. Load the Versapro Software to the PC.
2. Open the Versapro Software.
3. Switch ON the PLC Trainer and stepper motor .
4. Connect PLC and stepper motor.
5. Open the New folder and draw the ladder logic program.
6. Connect the PLC to PC.
7. Select the correct hardware Configuration [Sl. No].
8. Store the Program to PLC.
9. Run the Program.
10. Verify the performance of the stepper motor.

**WIRING DIAGRAM:**



## LADDER LOGIC PROGRAM:



## RESULT:

Ladder logic program for sequential operation of stepper motor has been designed and the performance is tested using trainer kit.

## **8. STAR TO DELTA STARTER**

### **AIM:**

To design a ladder logic program to perform the star to delta operation.

### **APPARATUS REQUIRED:**

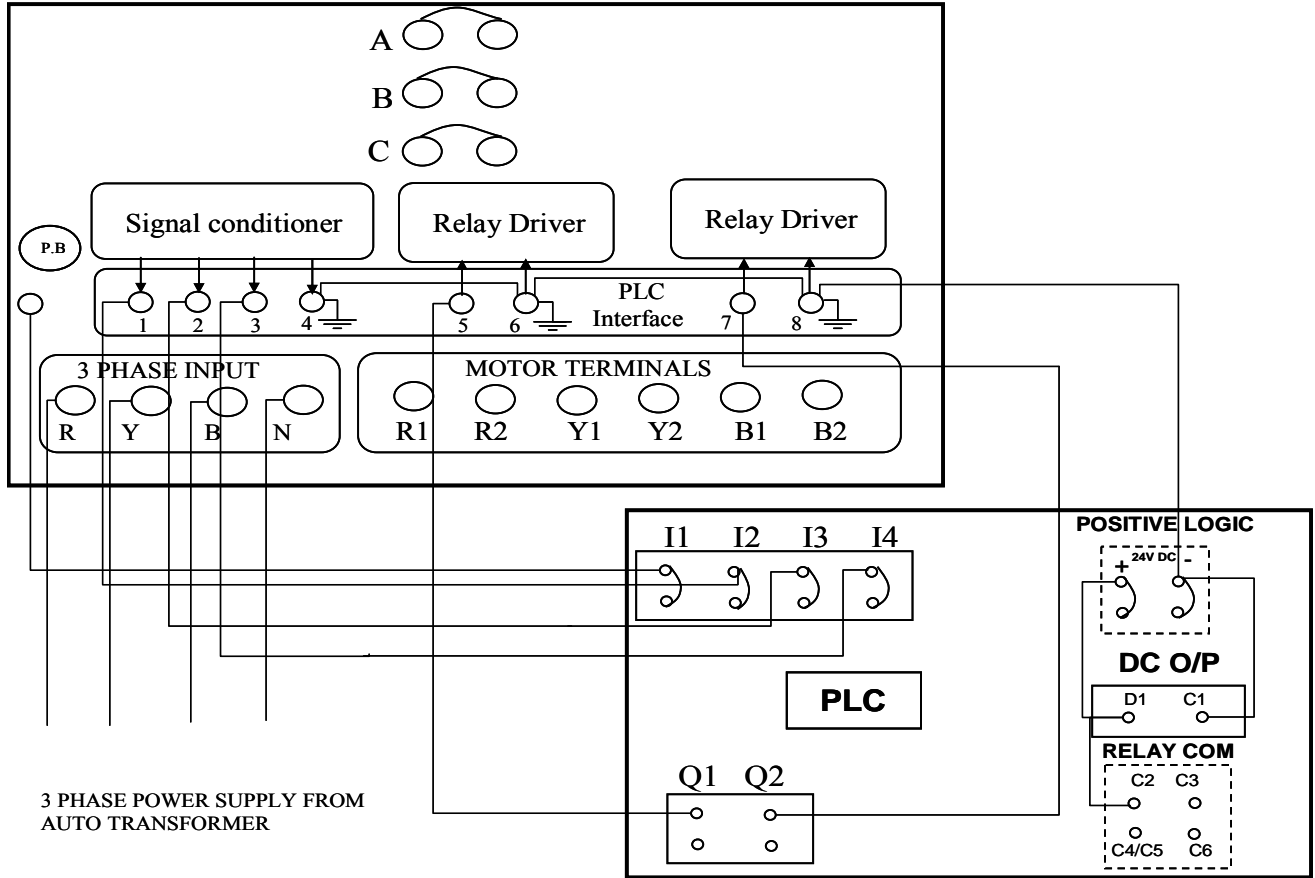
1. Star to Delta starter
2. PLC
3. Versapro Software
4. PC
5. RS-232 Cable.
6. Patch Chords

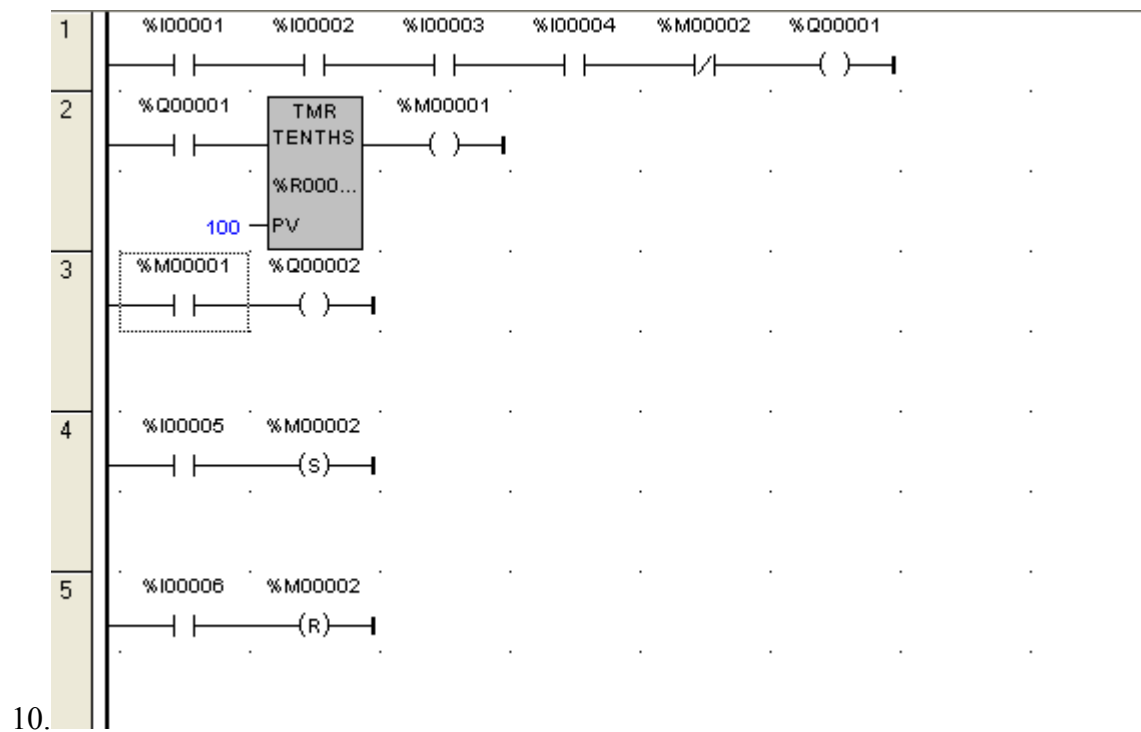
### **PROCEDURE:**

1. Load the Versapro Software to the PC.
2. Open the Versapro Software.
3. Switch ON the PLC Trainer and Star to Delta starter
4. Connect PLC and Star to Delta starter.
5. Open the New folder and draw the ladder logic program.
6. Connect the PLC to PC.
7. Select the correct hardware Configuration [Sl. No].
8. Store the Program to PLC.
9. Run the Program.
10. Verify the performance of the Star to Delta starter



**WIRING DIAGRAM:**





RESULT:

Ladder logic program for star to delta starter has been designed and the performance is verified using trainer kit.

## **9. DC MOTOR SPEED CONTROL SYSTEM**

### **AIM**

To control the speed of the PMDC motor using PLC

### **APPARATUS REQUIRED**

1. Speed Control Module Trainer (VSAT-01).
2. Versamax Micro PLC.
3. PC
4. RS-232Cable
5. 12V DC Motor
6. Patch Chords.

### **PROCEDURE**

1. Write your application program in the programming area.
2. Store the program to PLC.
3. After storing to set the P,I,D values in the PID function tuning Parameters task.
- 4.. Right click over the PID function click the Tuning Parameters task.
5. Now the PID tuning Parameter control dialogue box appear in the screen.

Here to set the following values in the particular parameters

$P = 1$

$I = 0.1$

$D = 0.1$  to  $7$  (Respective your application output oscillation to change the D value)

Sample Period =  $0.01$  Sec

Upper clamp =  $32000$

SP/PV Range =  $32000$  Min

slew Time=  $1$  Sec

6. Using patch cards, connect the PLC PWM output to the input of Analog output point in the speed control module trainer kit.
7. Using Patch cards connect the speed control module trainer kit analog input to PLC HSC input.
8. Run the program
9. Set motor speed in PC through a Versapro Software.

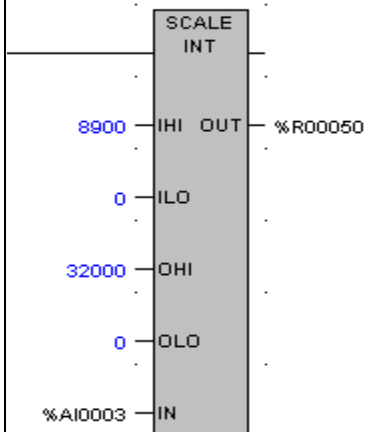
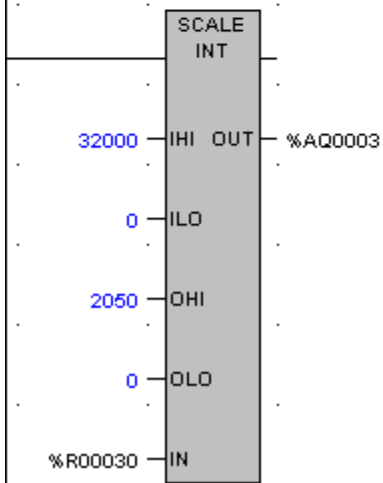
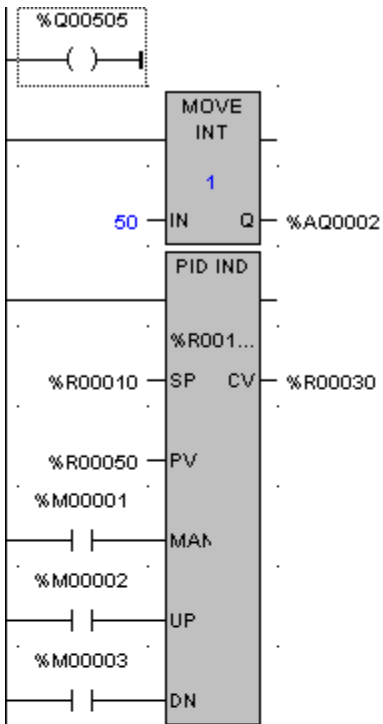
## **PID IND BLOCK**

The independent algorithm (PID IND) is the close loop control algorithm. The PID function has six input parameter: a process set point (SP), a process variable (PV) , a manual / auto Boolean switch (MAN), a manual mode up adjustment input (UP), and manual mode down adjustment (DN). It also has an address, which specifies the location of a block of parameters associated with the function. It has two output parameters, a successful Boolean output and the control variable result (CV). If the manual input is true, the PID block is placed in manual mode and the output control variable is set from the Manual command parameter %Ref + 13. If either the UP or DN inputs are true, the manual command word is incremented or decremented by one CV count every PID solution. For faster manual changes of the output control variable, it is also possible to add or subtract any CV count value directly fo / from the manual command word.

<b>Parameter</b>	<b>Description</b>
<b>Address</b>	The Variable's address is the location of the PID control block information, which consists of 40 consecutive registers of %R, %P, or %L memory.
<b>SP</b>	SP is the control loop set point. Data Type: WORD
<b>PV</b>	PV is the control loop process variable. Data type : word.
<b>MAN</b>	When energised, the PID function is in MANUAL mode.
<b>UP</b>	When energised, if in MANUAL mode, the CV output is adjusted up.
<b>DN</b>	When energised, if in MANUAL mode, the CV output is adjusted down.

## **SCALE INT**

Use this function to scale an input value and place it in an accumulator. This function has five inputs/parameters. For the integer based version of the function (SCALE-INT) all the parameters be integer based (signed).



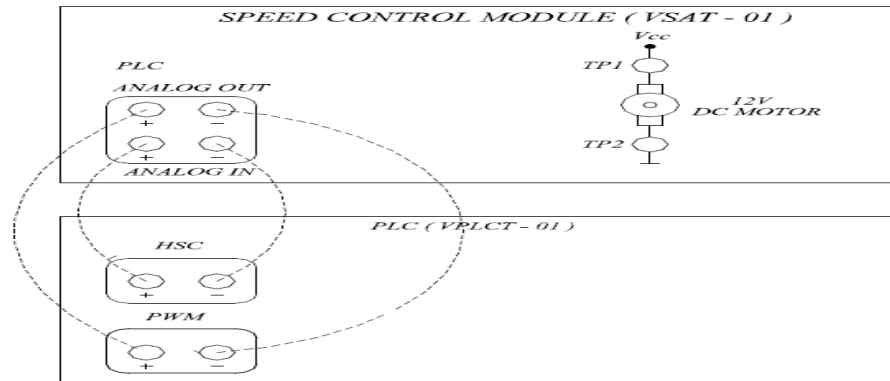
Parameter	Description
IHI	This input can either be %R, %AI, %AQ reference memory or constant. It represents the maximum input value.
ILO	This input can either be %R, %AI, %AQ reference memory or constant. It represents minimum input value.
OHI	This input can either be %R, %AI, %AQ reference memory or constant. It represents maximum scale value. When the IN input is at the IN Hi value, the accumulator value is the same as the OHI value.
OLO	This input can either be %R, %AI, %AQ reference memory or constant. It represents the minimum scaled value. When the IN input is at ILO value, the accumulator value is the same as of the OLO value.
IN	This input can either be %R, %AI, %AQ reference memory or constant. It represents the value to be scaled.

Analog Input to PLC  
 Analog output from PLC

Process Variable (PV) from the 'motor.  
 Control variable (CV) to the motor.

The set point is given to PLC. The- process variable (PV) from the motor is given as input to the Analog input of PLC. The PLC performs, the control operation and gives the control variable (CV) to control PMDC motor.

**CONNECTION DETAILS**



**RESULT:**

Ladder logic program for temperature control has been designed and the performance is verified using trainer kit.

## 10. TEMPERATURE CONTROL SYSTEM

### AIM:

To study the PID - characteristics of a temperature controller.

### APPARATUS REQUIRED

1. Temperature Controller kit,
2. PLC Trainer kit,
3. Water path,
4. Thermometer,
5. Multimeter

Temperature measurement plays a major role in industrial application. The various sensors, which is used to measure the temperature are thermocouple, Thermistor, RTD. Due to the salient features of thermocouple it is being widely used in industries. In thermocouple based on thermoelectric principle, it senses the temperature of the medium, the junction of two junction temperature difference is directly proportional to emf, which is used to measure of temperature.

In temperature application trainer, output of pulse width modulation [PWM] in PLC is fed into control input, which is near by 5V DC power drive circuit is converted 230V AC, which is used to heat the water by heater. Thermocouple sensor senses the temperature level of water and converted mV into voltage level (5V), which is fed into PLC & Display unit. We will control the desired temperature by ladder logic program in PLC.

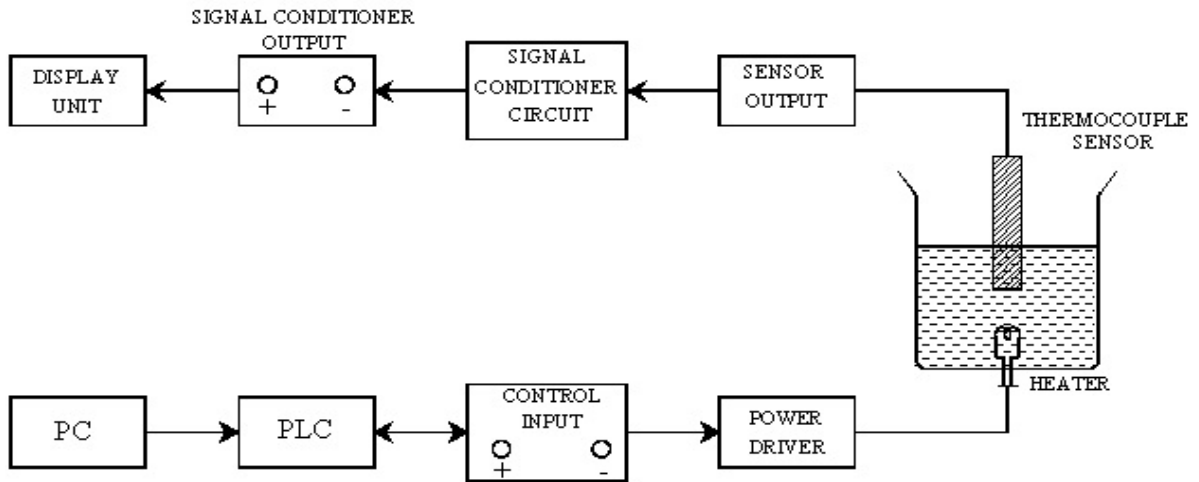
In temperature application trainer, output of PWM in PLC is patched into control input. Signal conditioner output is patched into input of Hsc in PLC. For the above two values for scale integer block in ladder logic, which is used to desired process variable and control voltage. PV & CV is determined by our desired temperature under control. Hence for above manipulation PLC used to control the process stable and accurately.

### WORKING OF THE SYSTEM

Output of pulse width modulation in PLC is patched into control input unit is fed into power driver unit. This unit is converted 5V DC (to) 230V AC by using SCR circuit. Heater is get the output value of power driver unit, which is heating the water. Thermocouple sensor senses the temperature and gives output in millivolts range. AD 590 [Temperature transducer] converts temperature in degree in to an output current, that is I:A for every degree kelvin. It requires a supply voltage exceeding 4V to bias internal transistor circuitry, signal conditioner unit is used to converted mV range into voltage, such as (0-100°C) as respective as (0-5V). These range of voltage is get from signal conditional circuit and given input of High Speed Counter (HSC) in PLC.

In high speed counter input [3AI] get from PLC as like as process variable [PV], hence PLC again produced PWM, it will control the temperature. This process was repeated again and again, which is used to controlled the temperature for our desired range.

## BLOCK DIAGRAM



## THERMOCOUPLE

It is based on thermoelectric principle, two junction temperature difference is directly proportional to the generated emf, which is a measure of temperature. Here we are using J-Type thermocouple. Temperature compensation is performed by AD-590 temperature sensor which is having two compensations are load compensation and reference junction compensation. The reference junction is not held at 0°C, the observed value must be corrected by adding to it a voltage that has resulted from temperature difference equal to amount by which the reference junction is above 0°C. Connecting wire made of the same material as thermocouple wires, which is chosen such that the relationship between emf and temperature is same. These compensations are called lead compensation.

### SPECIFICATION

#### Thermocouple

Type	-	J-type
Material	-	Iron constantan
Working Temp.	-	(-200°C) to (760°C)
Coating	-	Nickel, chromium,

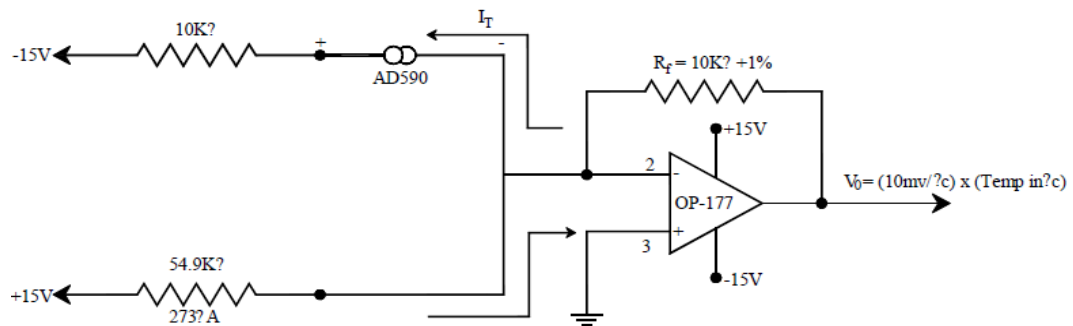


## LED

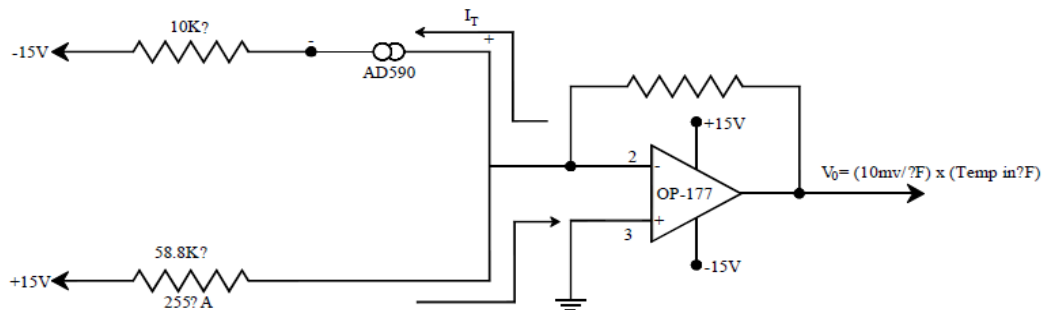
- Size - 50 × 20mm
- Type - Common anode
- Display - 3.5 digit
- Segment - 7 - Segment

## Power Supply

- Input - 230V AC/ 50Hz
- Output - 5V/500mA
- 12V/500mA



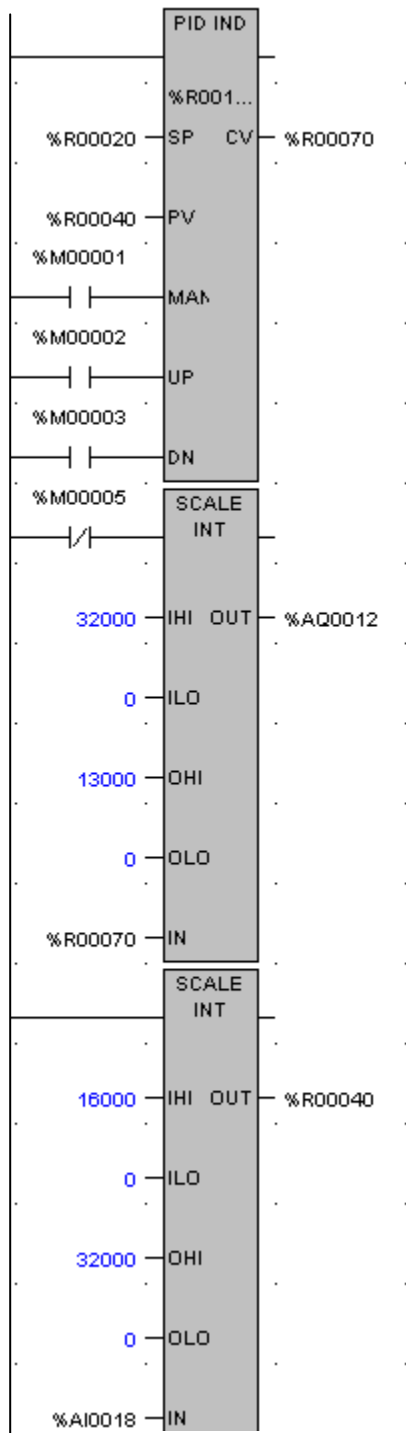
a)  $V_0 = 0$  at  $0^\circ\text{C}$  &  $1000\text{mV}$  at  $100^\circ\text{C}$



b)  $V_0 = 0$  at  $0^\circ\text{F}$  &  $1000\text{mV}$  at  $100^\circ\text{F}$

## LADDER LOGIC PROGRAM EXPLANATION

In this temperature controller needs three analog outputs and one analog input, but nano and fourteen point PLCs doesn't have these features. Hence we are using PWM for analog outputs &



Hsc for analog input. But at that same time analog input are assigned in one channel & Analog outputs are assigned in another one channel.

CHANNEL 1 [PWM]		CHANNEL 2	
2AQ[15-5000]	Duty cycle	3AI (Hsc)	Frequency
3AQ[0-10000]	Frequency OUTPUT Enabled	[0-32000]	
505Q			

For the above table used to insert the values of scale integer block, which is used for PID function. We can't give directly for analog inputs, hence we are using two scale integer blocks.

In rung - 1 directly using open coil for output enabled function in pulse width modulation (PWM).

In rung-2 duty cycle, we are using move integer block to moves the duty cycle's value [example : 50].

In rung-3, PID register is having SP - setpoint, PV - Process variable, CV - Control variable. Man-Manual, UP - UP, and Dn - down. Man, UP and DN ports are used for manual mode, hence we are inserting normally open contacts (No). SP, Pv & Cv are inserted some register values, due to on-line changes during process in running - mode.

In rung-4 and rung-5 are using for scale into blocks, which is automatically scaling our values.

This block is get four values from one register, which is two values for input high and input low values. Then another two values for output high & output low values, these four values are scaling automatically from input register and gives the scaling values to out port - register.

In rung-4, scale integer block is used for scaling to 3AQ values [control variables]. Input values of rung-4, IL0 - 0, IH1 - 32000, assigned, because PLC output is (0-24V) range. But we need only 5V hence we are inserting values in register [R100]& get the output value from 3A register.

In rung-5 scale integer block is used for scaling to PV register values [Process Variable]. Input values of rung-5 IL0, IH1-10000 assigned, because the frequency value of HSC. Hence we are inserting OL0-0, OH0-30000 values in register [3AI-HSC] and get output value from PV register [R120]. For the above process was repeated again and again set the PID value which is depends upon our desired value.

#### RESULT:

Ladder logic program for temperature control has been designed and the performance is verified using trainer kit.

# **11. IMPLEMENTATION OF PLC PROGRAMMING THROUGH SCADA**

## **INTRODUCTION**

The system consists of four main parts:-

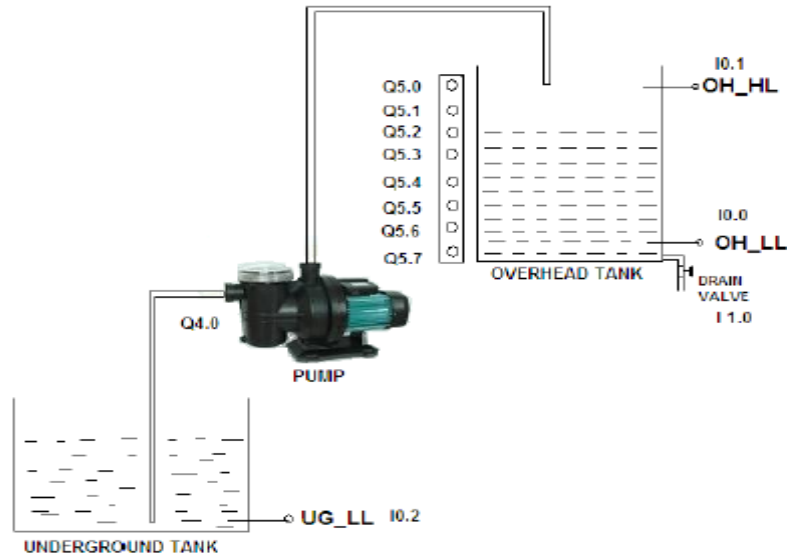
1. Level Sensor
2. PLC(Programmable Logic Controller)
3. Relay and Motor
- 4.HMI (Human Machine Interface)

The level sensor communicates the present level of the tank to the PLC. The PLC decides whether to turn the motor ON or OFF. The status of the system is communicated to the computer and is viewed and remotely controlled by the user through the HMI.

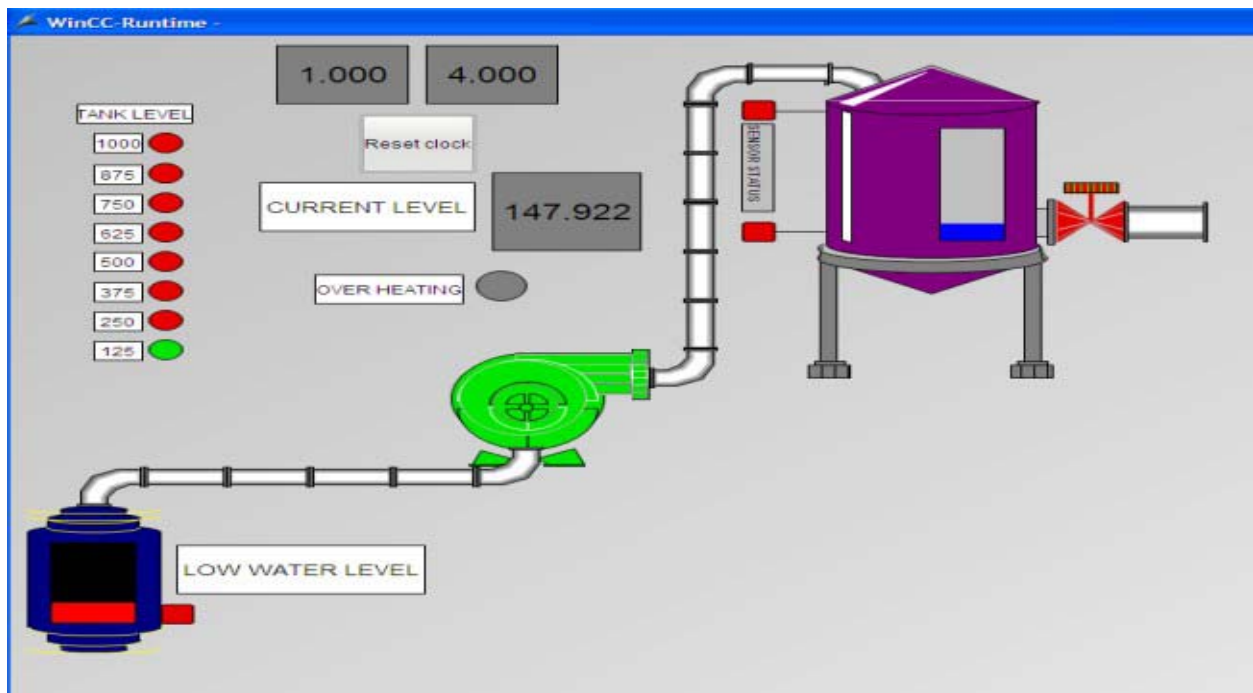
The overhead Tank is to be filled by a Pump. The pump will automatically start when the water level of Over Head Tank reaches below Low Level and stop when the level reaches High Level. Dry run is checked by the Low Level sensor of the Under Ground Tank. In that case Pump will not run. Run time monitoring of the pump in Second and minute is recorded and Reset Switch is also provided. Provision of Manual Start/Stop switch is incorporated which will totally override the automatic system. Provisions are also made for various alarms, such as “Underground Tank Empty Alarm” and “Pump run time exceeded 10 Minutes” (If pump runs continuously for 10 Minutes)

The implementation is divided into four parts:-

1. Sensor Positioning
2. Circuit Design
3. Ladder Programming
4. HMI Creation



Four inductive Proximity sensors were used to sense presence of water at required levels. The sensors are UG\_LL – Low Level Sensor Underground tank (I 0.2), OH\_LL – Low Level Sensor Overhead tank (I 0.0), OH\_HL – High Level Sensor Overhead tank (I 0.1) are placed as shown in above figure.



RESULT:  
Ladder logic program for water level control has been designed and the performance is verified using SCADA.