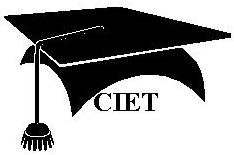
**COIMBATORE INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**Coimbatore -641109**

**An Autonomous Institution**

**Accredited with ‘A’ Grade by NAAC**



**DEPARTMENT OF MECHATRONICS ENGINEERING**

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Reg. No. : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Year / Branch:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Semester : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PLC AND AUTOMATION LAB**

|  |
| --- |
| **LABORATORY MANUAL** |

**COIMBATORE INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**Coimbatore -641109**

**Academic Year: 2024-2025**

**BONAFIDE CERTIFICATE**

|  |  |
| --- | --- |
| **Name:** | **Register Number:** |
| **Year/Sem:** | **Branch:** |

Certified that this is the bonafide record of work done by the above student during the academic year/semester: **2024-2025/ 6th**

Place:

Date: Staff In-charge

Submitted for the end semester practical examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of Internal Examiner Signature of External Examiner

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exp No.** | **Date** | **Name of the Experiment** | Marks | **Page No.** | **Sign** |
| 1 |  | **Study of Programmable Logic Controller(PLC)** |  |  |  |
| 2 |  | **Development of an application by using timer of PLC** |  |  |  |
| 3 |  | **Development of an application by using counter of PLC** |  |  |  |
| 4 |  | **Structured Text Programming Using CODESYS V3.5** |  |  |  |
| 5 |  | **Solving simple problems using Functional Block Diagram (FBD)**  **programming in PLC** |  |  |  |
| 6 |  | **Traffic Light Control using PLC** |  |  |  |
| 7 |  | **Boolean logic simulation using PLC software** |  |  |  |
| 8 |  | **Motor speed control logic simulation using PLC Software** |  |  |  |
| 9 |  | **Start and Stop operation of motor using PLC software** |  |  |  |
| 10 |  | **SCADA system for Tank Fluid Level Control** |  |  |  |

**Exp No: 1 Study of Programmable Logic Controller(PLC)**

**Aim:**

To study about the programmable logic controller (PLC) field interface module and hardware working and application.

**Theory:**

Initially industries used relays to control the manufacturing processes. The relay control panels had to be regularly replaced, consumed lot of power and it was difficult to figure out the problems associated with it. To sort these issues, Programmable logic controller (PLC) was introduced.



*Fig. 1: Graphical Representation Of PLC*

**What is PLC?**

**Programmable Logic Controller** (**PLC**) is a digital computer used for the automation of various electro-mechanical processes in industries. These controllers are specially designed to survive in harsh situations and shielded from heat, cold, dust, and moisture etc. **PLC** consists of a microprocessor which is programmed using the computer language. The program is written on a computer and is downloaded to the PLC via cable. These loaded programs are stored in non – volatile memory of the PLC. During the transition of relay control panels to PLC, the hard wired relay logic was exchanged for the program fed by the visual programming language known as the Ladder Logic was created to program the PLC.

**PLC Hardware**

The hardware components of a PLC system are CPU, Memory, Input/Output, Power supply unit, and programming device. Below is a diagram of the system overview of PLC.

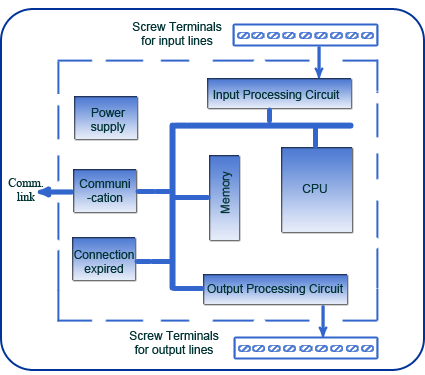


Fig. 2: An Overview Of Hardware Components Of A PLC System

 **CPU** – Keeps checking the PLC controller to avoid errors. They perform functions including logic operations, arithmetic operations, computer interface and many more.

 **Memory** – Fixed data is used by the CPU. System (ROM) stores the data permanently for the operating system. RAM stores the information of the status of input and output devices, and the values of timers, counters and other internal devices.

 **I/O section** – Input keeps a track on field devices which includes sensors,

switches.

 **O/P Section –** Output has a control over the other devices which includes motors, pumps, lights and solenoids. The I/O ports are based on Reduced Instruction Set Computer (RISC).

 **Power supply** – Certain PLCs have an isolated power supply. But, most of the PLCs work at 220VAC or 24VDC.

 **Programming device** – This device is used to feed the program into the memory of the processor. The program is first fed to the programming device and later it is transmitted to the PLC’s memory.

**System Buses** – Buses are the paths through which the digital signal flows internally of the PLC. The four system buses are:

· Data bus is used by the CPU to transfer data among different elements.

·Control bus transfers signals related to the action that are controlled internally.

· Address bus sends the location’s addresses to access the data.

· System bus helps the I/O port and I/O unit to communicate with each other.

**Working & Application:**

**Working of PLC (Programmable Logic Controller)**

The Programmable logic controller functions in four steps.

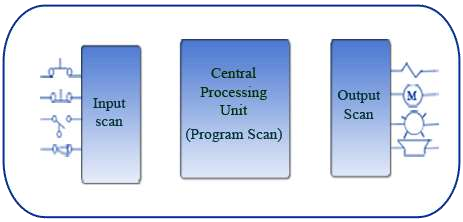


Fig. 3: Typical Block Diagram Of Programmable Logic Controller Functions

* **Input scan**: The state of the input is scanned which is connected externally. The inputs include switches, pushbuttons, and proximity sensors, limit switches, pressure switches. Ideally, they are transformers and not relays.
* **Program scan**: The loaded program is executed to carry out the function appropriately.
* **Output scan**: The input sources have a control over the output ports to energize or de- energize them. The outputs include solenoids, valves, motors, actuator, and pumps. Depending on the model of PLC, these relays can be transistors, triacs or relays.

 **Housekeeping**

**PLC Applications**

The simple suitable application is a conveyor system. The requirements of the conveyor systems are as follows:

 A programmable logic controller is used to start and stop the motors of the conveyor belt.

 The conveyor system has three segmented conveyor belts. Each segment is run by a motor.

 To detect the position of a plate, a proximity switch is positioned at the segment’s end.

 The first conveyor segment is turned ON always.

 The proximity switch in the first segment detects the plate to turn ON the second conveyor segment.

 The third conveyor segment is turned ON when the proximity switch detects the plate at the second conveyor.

 As the plate comes out of the detection range, the second conveyor is stopped after 20 secs.

 When the proximity switch fails to detect the plate, the third conveyor is

stopped after 20 secs.

**History**

Programmable Logic Controllers were discovered by the automotive industry to substitute the re-wiring of the machine’s control panel.Prior to the invention of PLC, automobiles were manufactured using plenty of relays, cam timers, and closed loop controllers. The electricians had to re-wire every part of the machine daily which was time consuming and highly expensive on the financial front.

Later in the year 1968, a request for an electronic device for the hard-wired relay systems was made by GM hydramatic. Bedford Associates won the proposal and started a new company to develop, fabricate, sell, and service this new launched product. The first PLC launched was designated 084 as it was the eighty fourth projects of Bedford Associates. Dick Morley worked on this project and is being considered as the Father of PLC. In the year 1977, the brand invented by Modicon was sold to Gould Electronics. The Gould Electronics later sold it to German Company AEG which was later taken over by French Schneider Electric.

The first 084 model of PLC was revealed in North Andover, Massachusetts at the

Modicon headquarters. The automotive industry is one of the largest users of PLC.

**Advantages**

 PLCs can be programmed easily which can be understood clearly well.

 They are fabricated to survive vibrations, noise, humidity, and temperature.

 The controller has the input and output for interfacing.

**Disadvantages**

 It is a tedious job when replacing or bringing any changes to it.

 Skilful work force is required to find its errors.

 Lot of effort is put to connect the wires.

 The hold up time is usually indefinite when any problem arises.

**Exp No: 2 Development of an application by using timer of PLC**

**AIM:**

To simulate ladder logic programs using Allen Bradley RSLogix Micro English Software

**APPRATUS REQUIRED:**

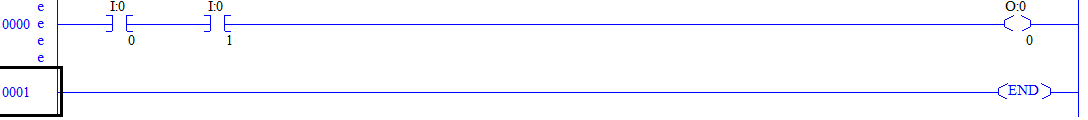
1. PC with PLC software

**PROCEDURE:**

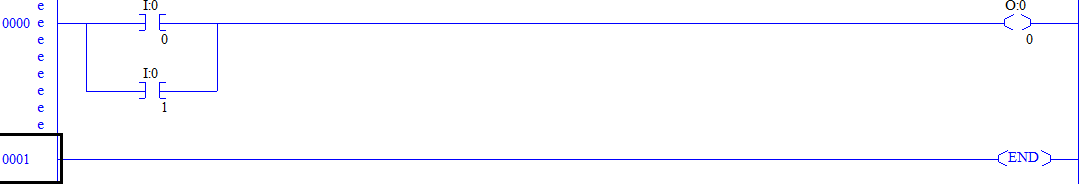
1. Create the ladder logic program in the program space of Allen Bradley RSLogix Micro English Software
2. Simulate the program
3. Download the program to emulator and run
4. Verify the results

**1) Realization of Logic Gates**

**AND GATE:**

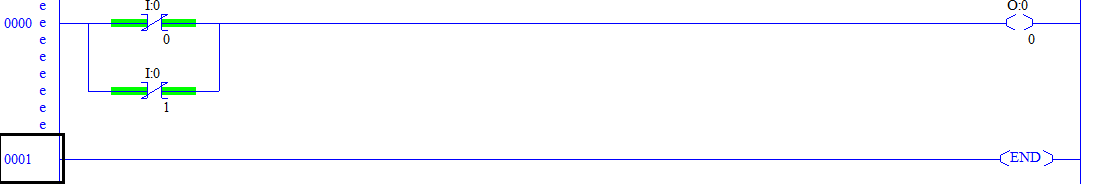


**OR GATE:**



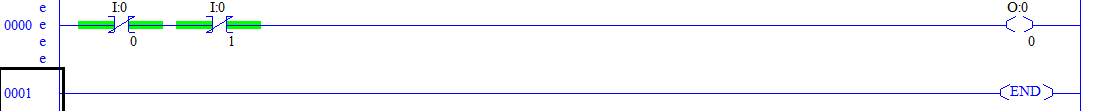
**NAND GATE:**

Y=(AB)’=A’+B’



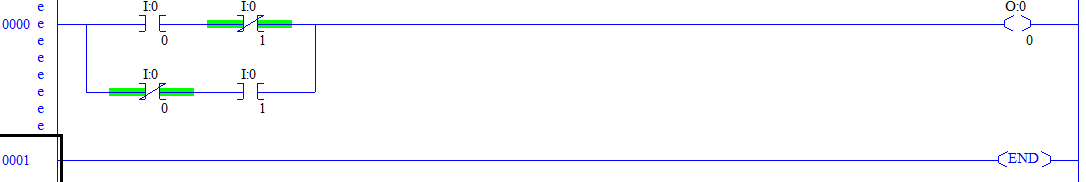
**NOR GATE:**

Y=(A+B)’=A’.B’



**XOR GATE:**

Y=A’B+AB’

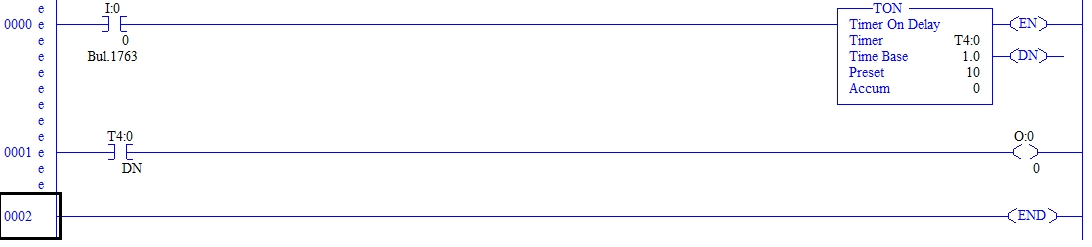


**XNOR GATE:**

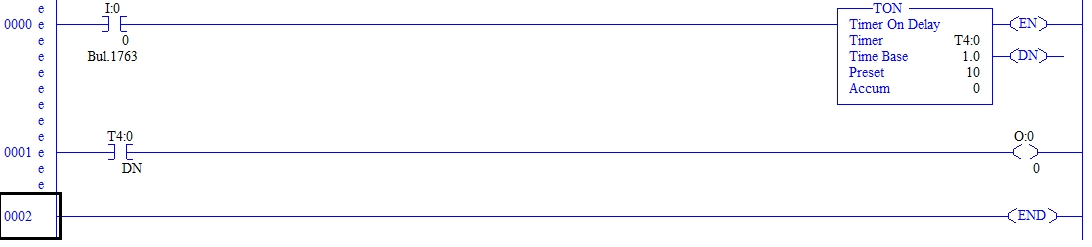
Y=AB+A’B’



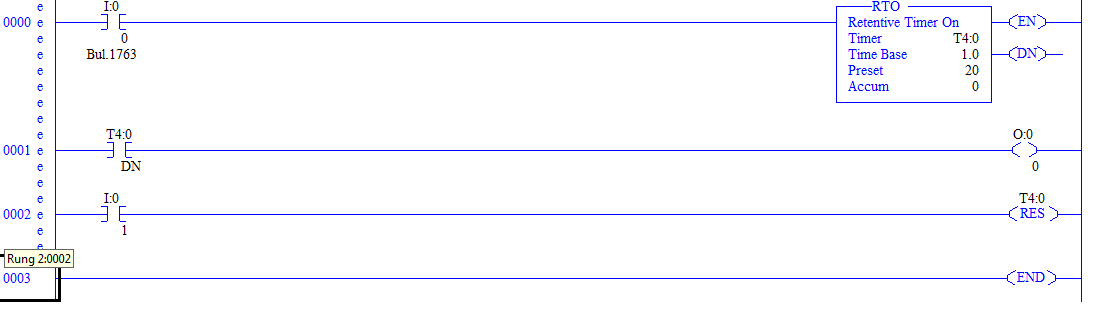
**On Delay Timer:**



**OFF Delay Timer:**



**Retentive Timer:**



**Exp No: 3 Development of an application by using counter of PLC**

**AIM:**

To simulate ladder logic programs using Allen Bradley RSLogix Micro English Software

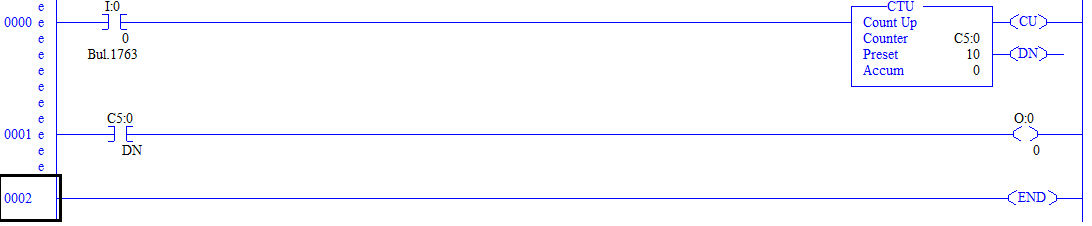
**APPRATUS REQUIRED:**

1. PC with PLC software

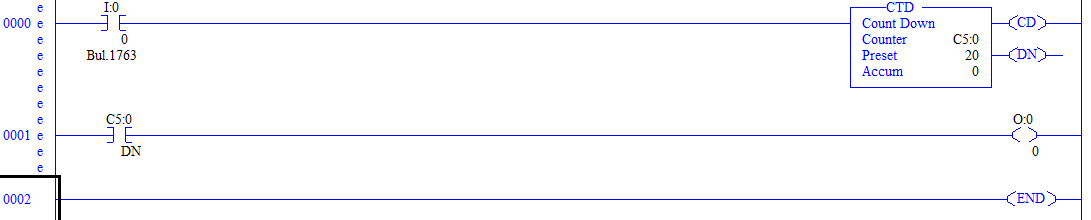
**PROCEDURE:**

1. Create the ladder logic program in the program space of Allen Bradley RSLogix Micro English Software
2. Simulate the program
3. Download the program to emulator and run
4. Verify the results

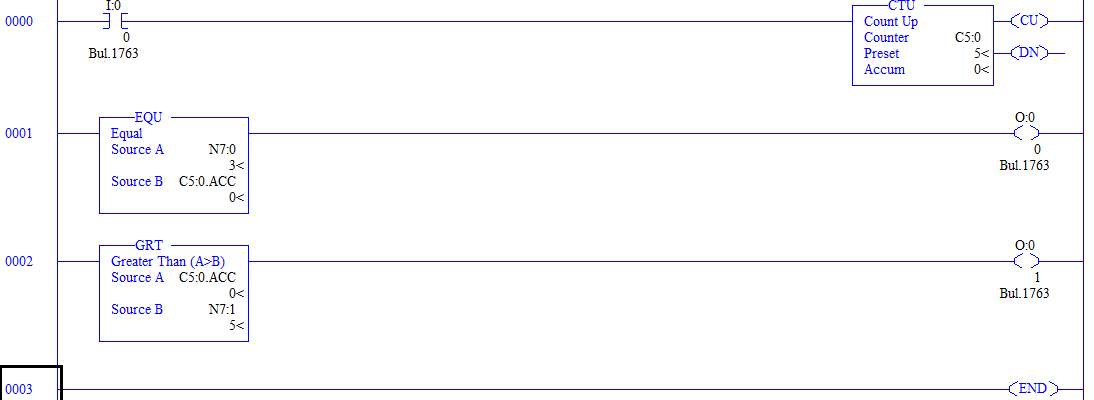
**Up Counter:**



**Down Counter:**



**Comparator:**



**Exp No: 4 Structured Text Programming Using CODESYS V3.5**

**Aim:**

To write and execute structured text programs using CODESYS V3.5

**Software used:** CODESYS V3.5

**Theory:**

**Structured Text Programming:**

**Structured text**, abbreviated as **ST** or STX, is one of the five languages supported by the IEC 61131-3 standard, designed for programmable logic controllers (**PLCs**). By using a text-based PLC programming language, your program will take up much smaller space, and the flow/logic will be easier to read and understand.

**Syntax of Structured Text:**

* **All statements are divided by semicolons**  
  Structured Text consists of statements and semicolons to separate them.
* **The language is not case-sensitive**  
  Even though it is good practice to use upper- and lowercase for [readability](https://en.wikipedia.org/wiki/Readability), it’s not necessary.
* **Spaces have no function**  
  But they should be used for readability.

A **statement** is you telling the PLC what to do

*X : BOOL;*

PLC to **create a variable** called **X** and that variable should be a **BOOL** type

**Variables in Structured Text:**

variable X is defined in between two other keywords – **VAR** and **END\_VAR**.

X : BOOL;

**Data Types in Structured Text:**

* Integers
* Floating points
* Time
* Strings
* Bit strings

**Types of Operators**

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators

## Conditional Statements

**(i) IF Statements**

IF [boolean expression] THEN

<statement>;

ELSIF [boolean expression] THEN

<statement>;

ELSE

<statement>;

END\_IF ;

**(ii) CASE Statements**

CASE [numeric expression] OF

result1: <statement>;

resultN: <statemtent>;

ELSE

<statement>;

END\_CASE;

**Iteration with Repeating Loops**

**(i) FOR Loop**

FOR count := initial\_value TO final\_value BY increment DO

<statement>;

END\_FOR;

**(ii) WHILE Loops**

WHILE [boolean expression] DO

<statement>;

END\_WHILE;

**(iii) REPEAT Loops**

REPEAT

<statement>;

UNTIL [boolean expression]

END\_REPEAT;

**Steps:**

(i) Open new project as standard project from the file

(ii) Save the project and select Structured Text(ST) method

(iii) Double click and open PLC\_PROG file for Programming

(iv) Write the program and build to check any errors

(v) Go to simulation mode and login to execute the program

(vi) Force the variable values using F7 button and check the results

**Programs:**

STEPS: SIMULATION, BULID ALL, LOGIN, FORCE VALUES(F7 Key and ALT+F7), RUN)

**\*clear all to remove all the values assigned to variables**

**1) IF ELSE: PROGRAM**

PROGRAM PLC\_PRG

VAR

s1:BOOL;

s2:BOOL;

m1:BOOL;

END\_VAR

IF(s1) AND (s2) THEN

m1:=TRUE;

END\_IF

**2) FOR LOOP**

PROGRAM PLC\_PRG

VAR

I:INT;

f:ARRAY[0..4] OF INT:=[10,20,30,40,50];

avg:INT;

END\_VAR

avg := 0;

FOR i := 0 TO 4 DO

avg := avg + f[i];

END\_FOR;

avg := avg / 5;

**3) while loop**

PROGRAM PLC\_PRG

VAR

count: INT:=20;

a: INT:=0;

END\_VAR

WHILE(count>6) DO

a:=a+1;

count:=count-1;

END\_WHILE

**4) repeat**

PROGRAM PLC\_PRG

VAR

count: INT:=20;

a: INT:=0;

END\_VAR

REPEAT

**5) IF ELSE Statement**

a) IF (sensor1) AND (sensor2) THEN

motor1 := TRUE;

END\_IF;

b) sensor1:=0;

sensor2:=0;

solenoid:=0;

IF sensor1>=100 THEN

solenoid:=10;

ELSIF sensor2<=99 THEN

solenoid:=1000;

ELSE

solenoid:=9;

END\_IF;

**Exp No:5 Solving simple problems using Functional Block Diagram (FBD)**

**programming in PLC**

**Aim:**

To develop and execute Functional Block Diagram (FBD) programs using STEP 7 programming software

**Software used:** STEP 7 programming software

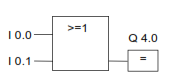
**Theory:**

**Functional Block Diagram (FBD):**

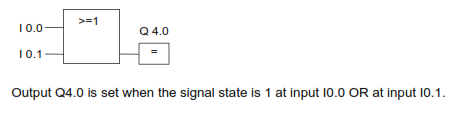
The primary concept behind an FBD is data flow. A Functional block diagram describes a function between input and output through a functional block. A FBD program is built using function blocks connected together to define the data exchange. The connecting lines will have a compatible information type at both ends.

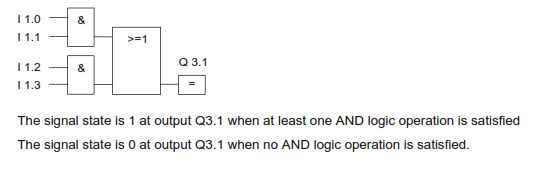
**(1) Bit Logic Instructions**

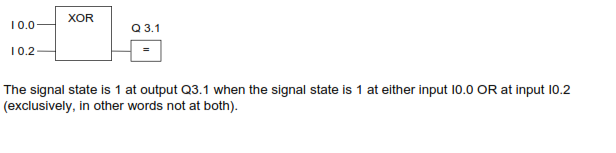
i) OR Logic Operation (**>=1)**

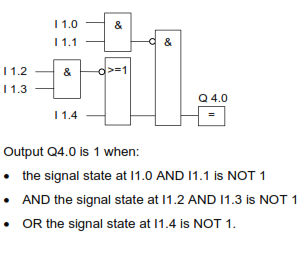


ii) AND

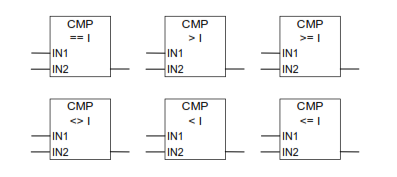


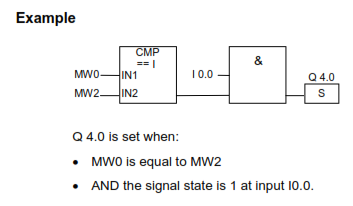




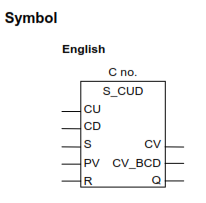


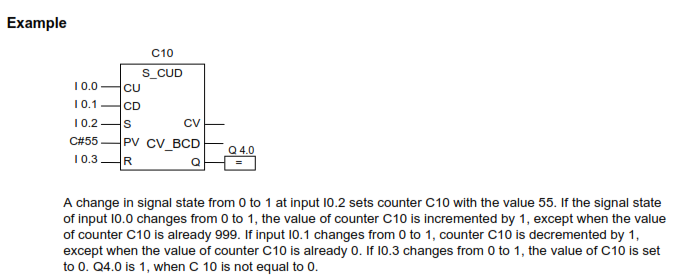
**(ii) Comparison Instructions**



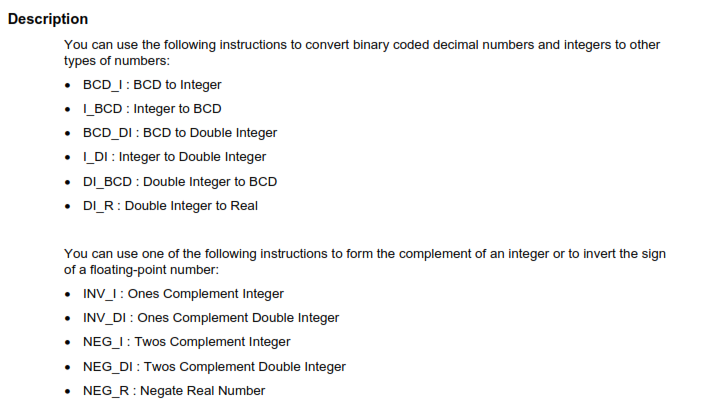


**(iii) Counter Instructions**

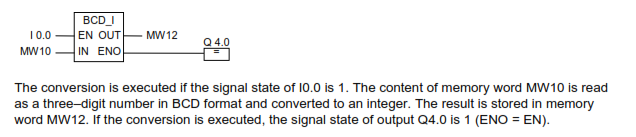




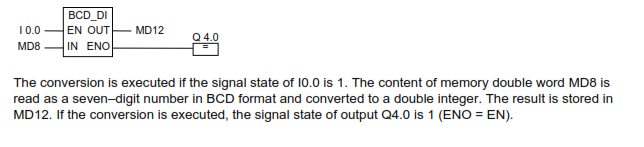
**(iv) Conversion Instructions**



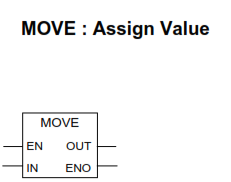
Example:

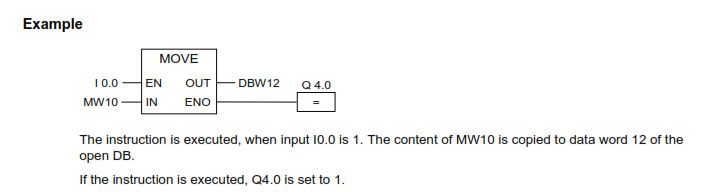


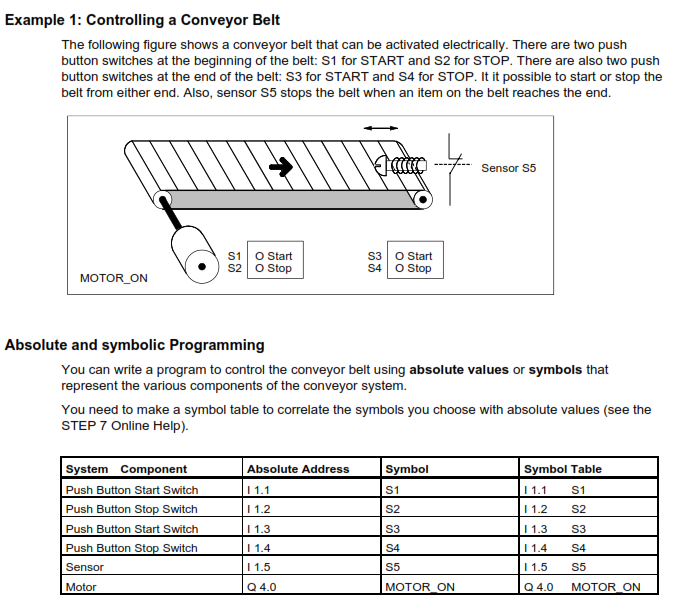


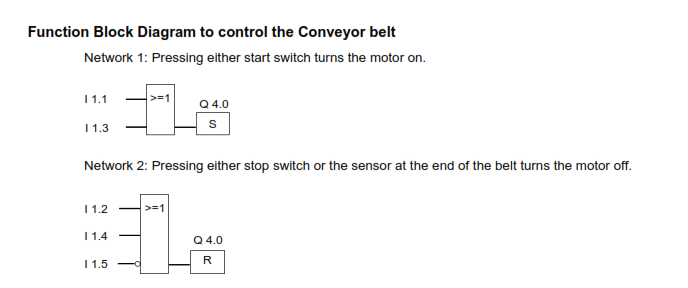


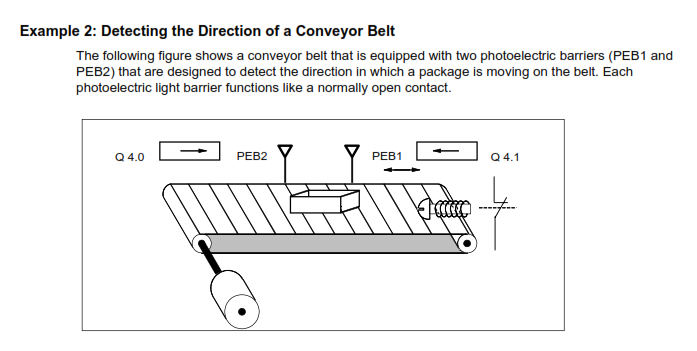
**(v) MOVE instructions**

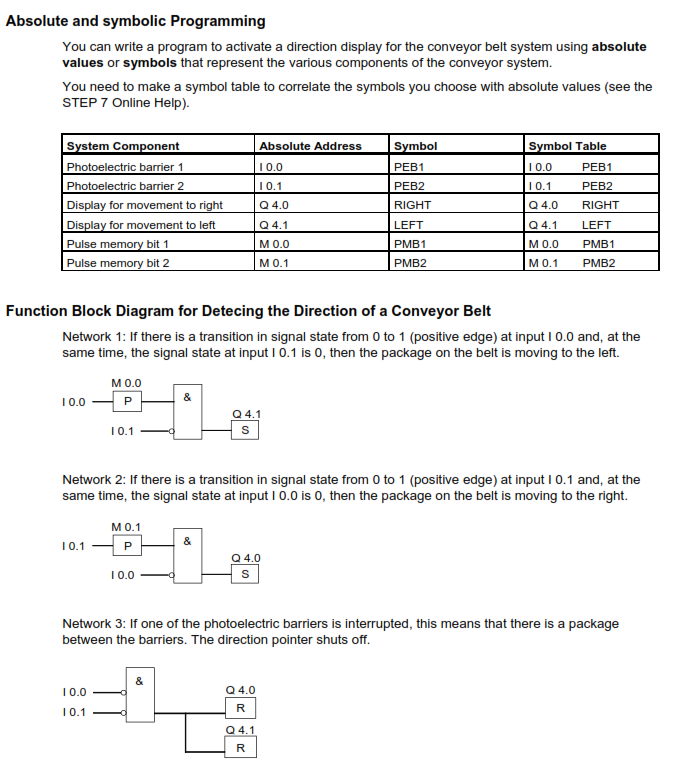












**Exp No:6 TRAFFIC LIGHT CONTROL USING PLC**

**Aim:**

To design and control the simple Traffic Light system by using PLC.

**Apparatus Required:**

1. VPAT - 03 Trainer kit

2. GE FANUC Versa MAX PLC

3. PC with Versa pro software

4. Communication cable PC/PPI

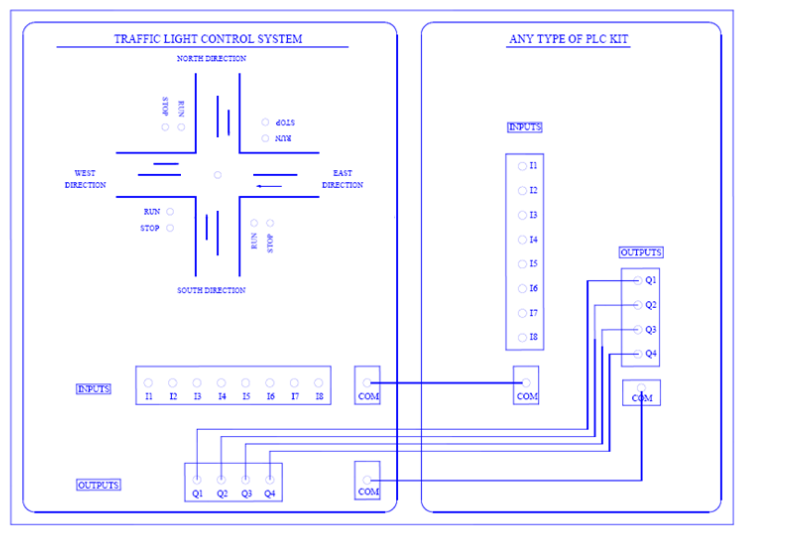
5. Patch chords

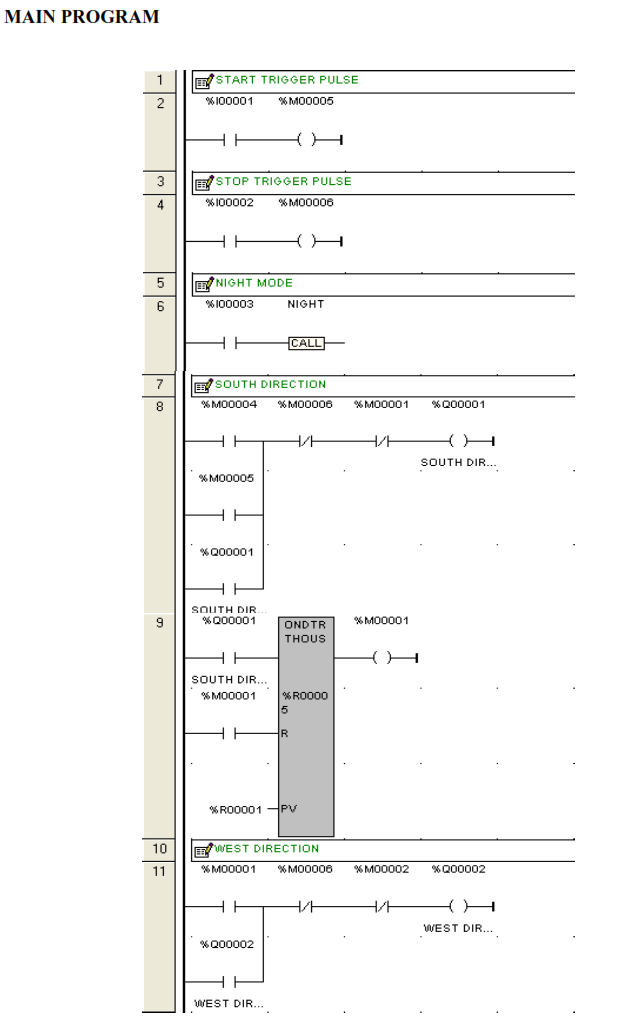
6. PC Power chords

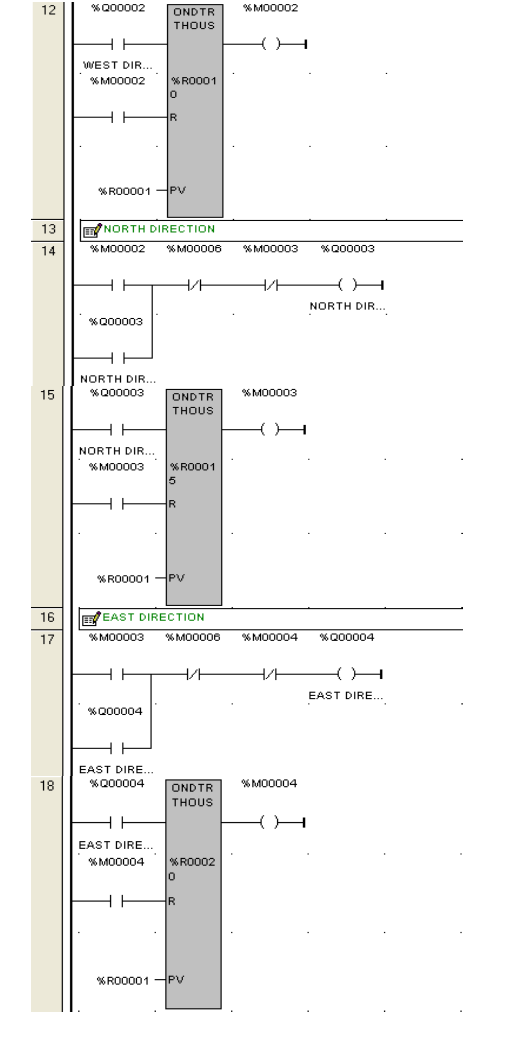
**Theory:**

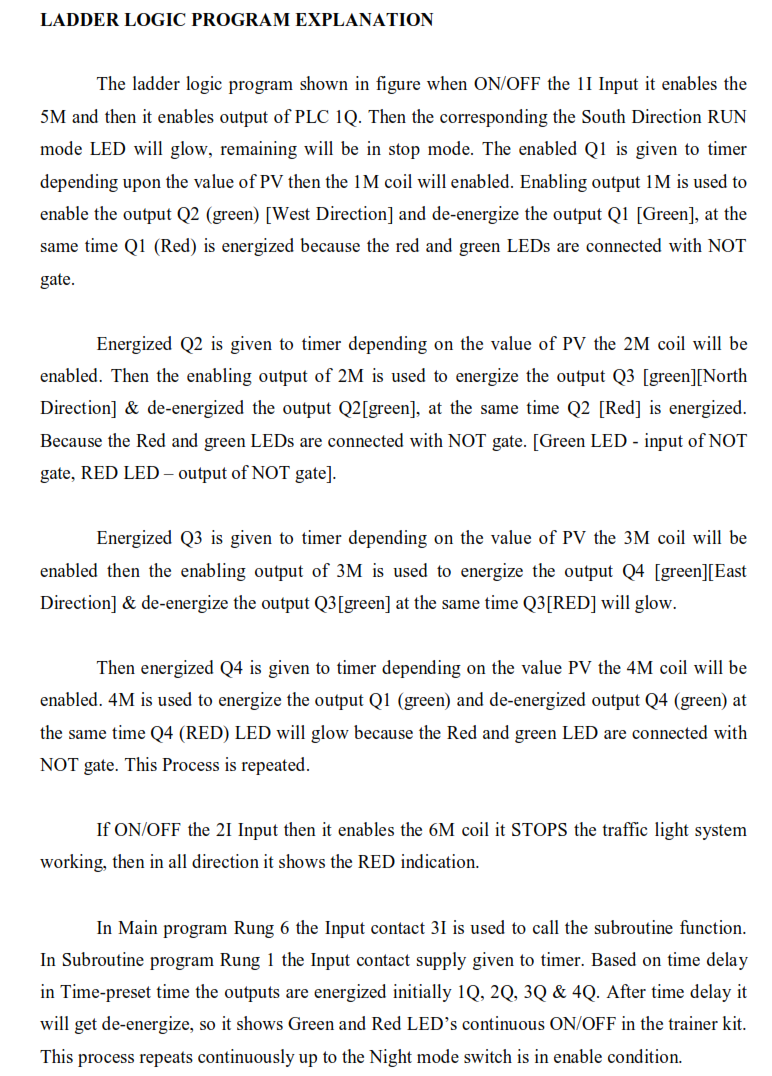
**Working of the system:**

In this traffic light controller, having four directions [North, South, East & West] each having Run & Stop mode. Hence 8 inputs, but the outputs are four. Such as 3 directions are STOP mode, one direction is RUN mode. For eight input get from Traffic Light controller kit and patched into PLC such as I1, I2, I3, I4, I5, I6, I7, I8. The ladder logic program can be drawn depending upon the application used. For the output of PLC such as Q1, Q2, Q3, Q4, are patched to output of Traffic Light controller trainer kit, then the LEDs will glow in one direction to RUN mode remains in STOP mode.









**Exp No:7**  **SPEED CONTROL OF DC MOTOR USING PLC**

**AIM**

To control the speed of the PMDC motor using Versamax Micro PLC.

**APPARATUS REQUIRED**

1. Speed Control Module Trainer (VSAT-01).

2. Versamax Micro PLC.

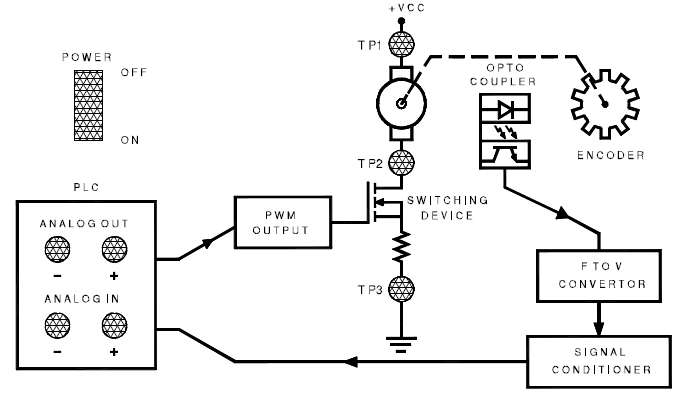
3. PC

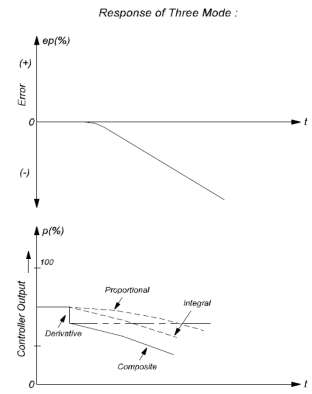
4. RS-232Cable

5. 12V DC Motor

. Patch Chords.

**PID CONTROL MODULE**





**THEORY:**

**PID CONTROLLER:**

One of the most powerful but complex controller mode operations combines the proportional, integral and derivative mode. This system can be used for virtually any process condition. The analytical expression is,



**where,**

KP = Proportional gain between error and controller Output (% per %)

dep / dt = Rate of change of error (%S) KD = Derivative gain constant (% - S/%)

KI = Scaling between error and controller output.

This mode eliminates the offset of the proportional mode and still provides fast response. The three mode controller action exhibits proportional, Integral and derivative action

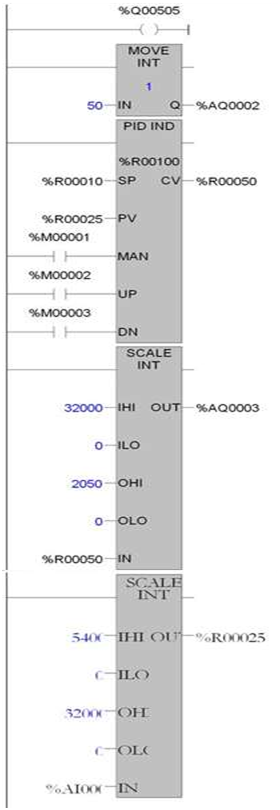
**PROGRAM EXPLANATION:**

**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 count every PID solution.

**LADDER LOGIC PROGRAM:**



**Parameter Description:**

Address: 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.

SP: SP is the control loop set point. Data Type: WORD

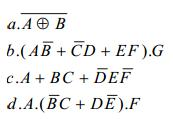
PV: PV is the control loop process variable. Data type: word. MAN : When energised, the PID function is in MANUAL mode.

UP: When energised, if in MANUAL mode, the CV output is adjusted up. DN: When energised, if in MANUAL mode, the CV output is adjusted

down.

**Exp No: 7 Boolean logic simulation using PLC software**

Construct a Ladder Logic for the following Boolean functions operations using any suitable PLC. Transfer and run your programs on the PLC and verify that the PLC operates according to the Boolean functions.



**AIM:**

To simulate ladder logic programs using Allen Bradley RSLogix Micro English Software

**APPRATUS REQUIRED:**

1. PC with PLC software

**PROCEDURE:**

1. Create the ladder logic program in the program space of Allen Bradley RSLogix Micro English Software

2. Simulate the program

3. Download the program to emulator and run

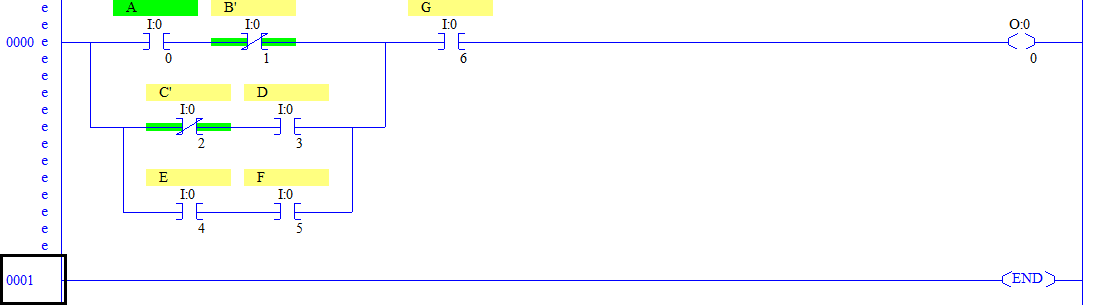
4. Verify the results

a. EX NOR Logic

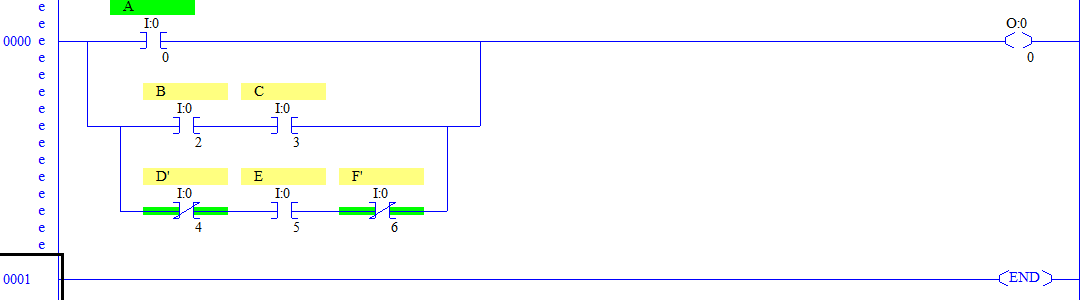
Y=AB+A’B’



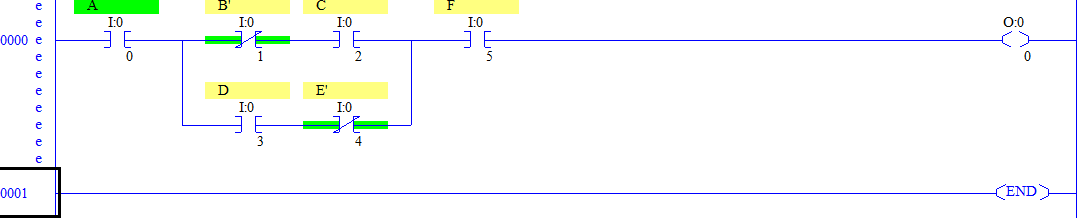
b. (AB’+C’D+EF).G



c. A+BC+D’EF’



d. A.(B’C+DE’).F



**Exp No: 8 Motor speed control logic simulation using PLC**

**Software**

Construct ladder logic for the following: A DC motor need to be started automatically with 30 second time delay. The motor is to be switched off at any desired point of time by one stop switch. Start switch is normally open and momentary contact switch. One output relay must control the supply to the motor. The stop switch is normally close and momentary contact. The motor should have a off and run light to indicate that the motor is running or at rest.

**AIM:**

To simulate ladder logic programs using Allen Bradley RSLogix Micro English Software

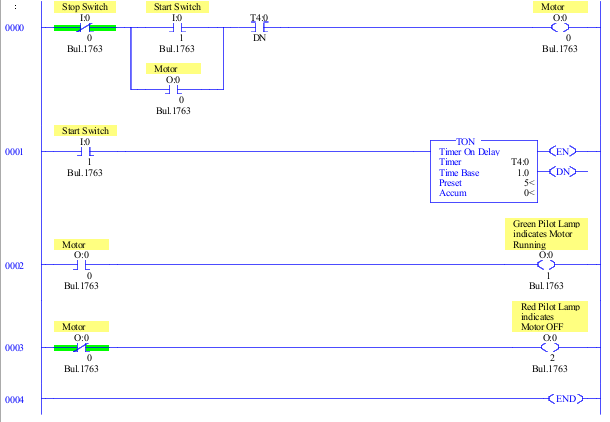
**APPRATUS REQUIRED:**

1. PC with PLC software

**PROCEDURE:**

1. Create the ladder logic program in the program space of Allen Bradley RSLogix Micro English Software
2. Simulate the program
3. Download the program to emulator and run
4. Verify the results

**Ladder Logic Program:**



**Exp No: 9 Start and Stop operation of motor using PLC software**

A PLC motor controller has two START buttons and two STOP buttons. The motor is to run if two RUN buttons depressed simultaneously. The motor should run when the buttons are released. The motor should stop by depressing any STOP button . Construct a LAD for this motor control task.

**AIM:**

To simulate ladder logic programs using Allen Bradley RSLogix Micro English Software

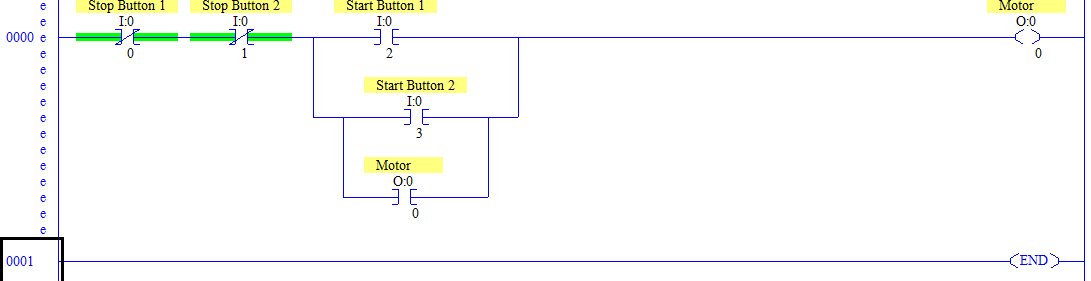
**APPRATUS REQUIRED:**

1. PC with PLC software

**PROCEDURE:**

1. Create the ladder logic program in the program space of Allen Bradley RSLogix Micro English Software
2. Simulate the program
3. Download the program to emulator and run
4. Verify the results

**Ladder Logic Program:**



**Exp No: 10 SCADA system for Tank Fluid Level Control**

**AIM:**

To design and simulate tank fluid level control system using open source SCADA software

**Software Used:**

Wonderware Intouch SCADA

**Program Code and Setting:**

on show:

START=0;

STOP=0;

VALVE1=0;

VALVE2=0;

While showing:

IF START==1 THEN VALVE1=1; ENDIF;

IF VALVE1==1 AND TANK<100 THEN TANK=TANK+1; ENDIF;

IF TANK==100 THEN VALVE1=0; VALVE2=1; ENDIF;

IF VALVE2==1 AND TANK>0 THEN TANK=TANK-1; ENDIF;

IF TANK==0 THEN VALVE2=0; ENDIF;

IF STOP==1 THEN VALVE1=0; VALVE2=0; TANK=0; ENDIF;

