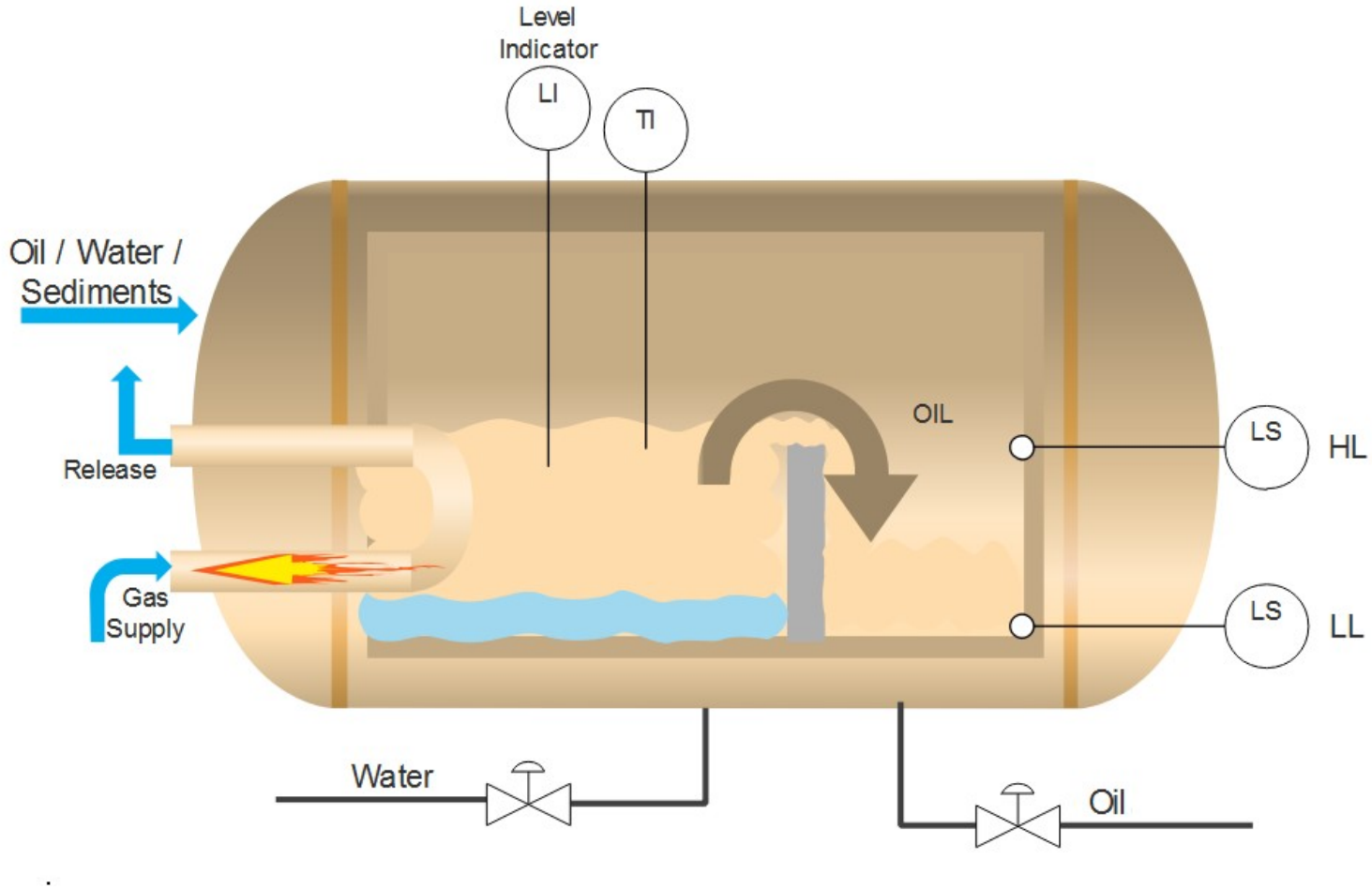
# PLC Program for Oil and Water Separation Process

This is a PLC Program for Oil and Water Separation Process.

## Problem Description

Implement programming of Oil and Water separation process in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

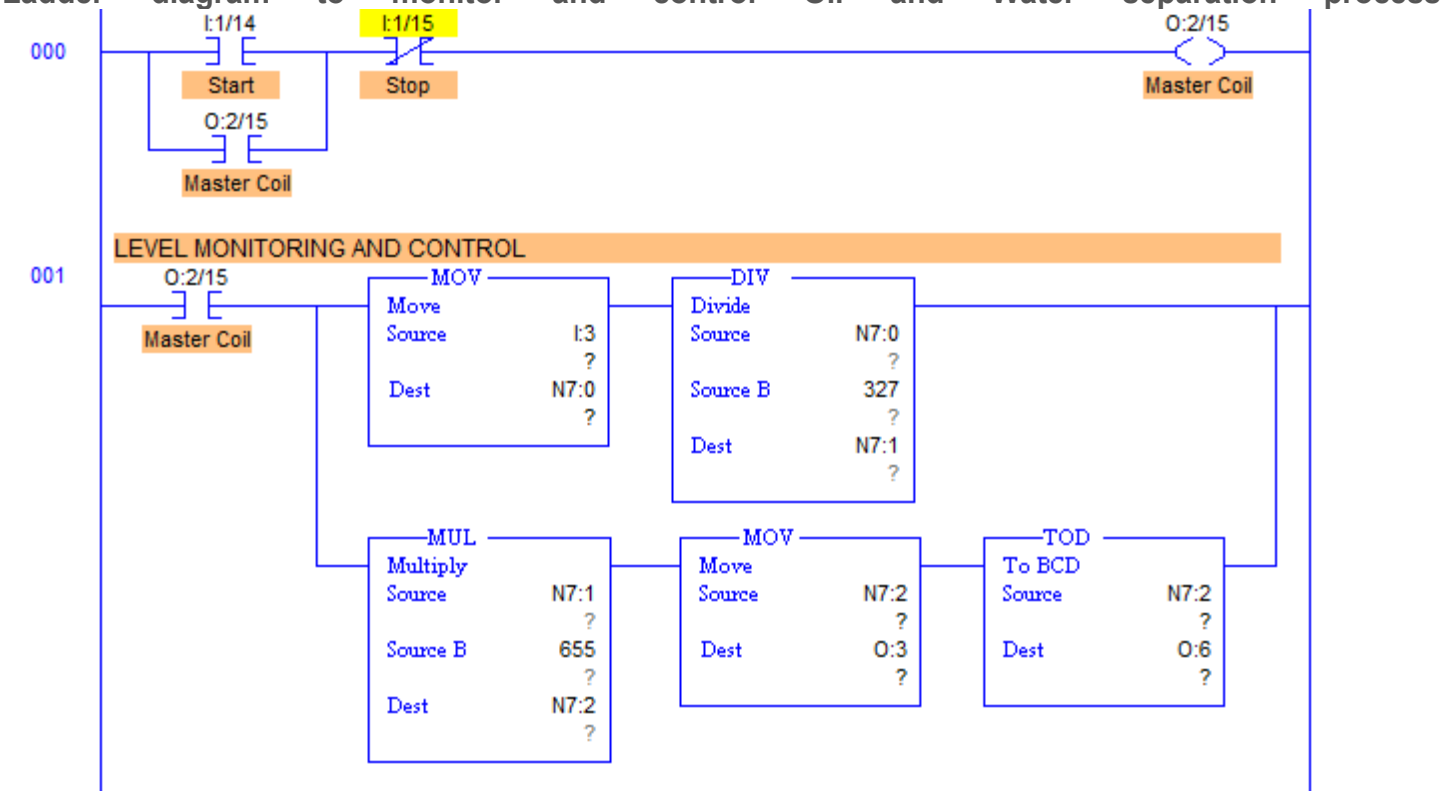
- Due to the gravitational force and different liquid density of Water and Oil, when water and oil both are present in the liquid, oil always reside on top of the liquid. By following this theory and making an arrangement shown in the diagram above, Oil and Water can be separated.
- However, this process does not completely remove water particles but only less than 10% of water contents are present after passing through this process.
- Level of the tank is monitored using level sensor. To control level of the system Inlet is controlled.
- To control the temperature of this system, ignition is to be controlled. Temperature sensor is used and gas supply flow is controlled to control temperature. Temperature set point here is 50 degree centigrade.
- Oil outlet is controlled by control valve which is operated by two level switches.

# PLC Program & Program Description

Here is PLC program for Oil and Water Separation Process, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14          = Start                                                (Input)
 I:1/15          = Stop                                                 (Input)
 O:2/15          = Master Coil                                          (Output)
 I:3             = Level sensor output                                  (Input)
 N7:0-N7:2       = Level monitoring and controlling registers           (Registers)
 O:3             = Inlet controlling valve                              (Output)
 O:6             = Level Display                                        (Output)
 I:1/1           = High level (Oil)                                     (Input)
 I:1/0           = Low level (Oil)                                      (Input)
 O:2/0           = Oil output valve (Single actuating)                  (Output)
 I:4             = Temperature output from sensor              (Input)
 N7:3 to N7:5    = Temperature monitoring and controlling registers (Registers)
 O:5             = Gas supply controlling valve                         (Output)
 O:7             = Temperature Display                                  (Output)
```

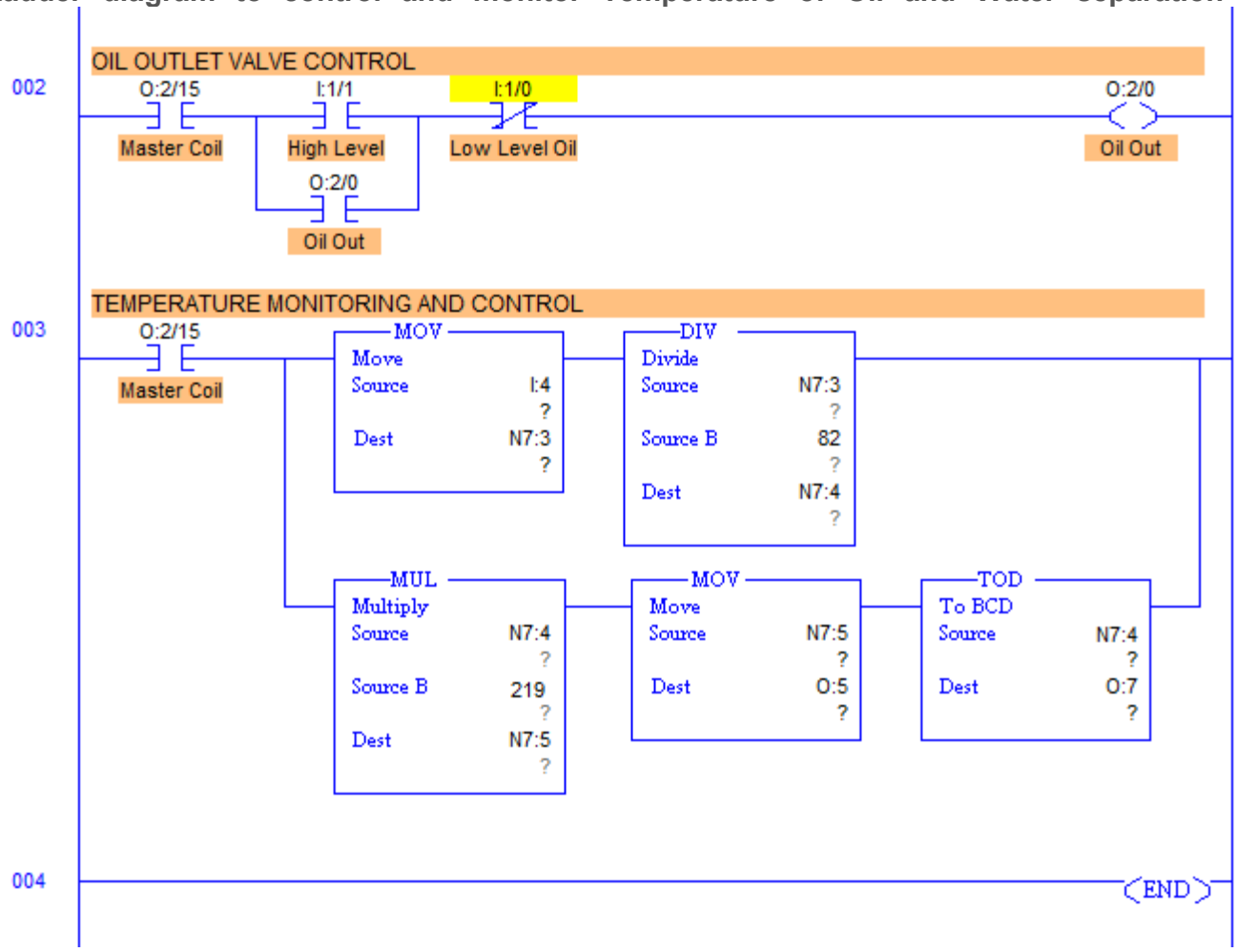
**Ladder diagram to monitor and control Oil and Water separation process**




- RUNG000 is the master start and stop control rung.
- RUNG001 is for Level Monitoring and control of Oil and Water separation process tank.
- I:3 is an input where Level sensor is connected with. Height of the tank is 200cms. And we need to control the level till 100cm. hence output of the sensor is divided by 327 which means that per cm, input is added with 327.
- This data is further processed and multiplied by 655, because when output is 100cm, we have to throttle inlet valve to be fully closed. So when output is 100cm in N7:1 register, it is multiplied by 655 and the full throttling is received at O:3 which is connected to Inlet valve through I-P converter.
- N7:1 stores the level of the tank which is moved to O:6 (Display address) through Decimal to BCD converter.

- Similarly Temperature monitoring and controlling is done as shown in the Ladder diagram below.

**Ladder diagram to control and monitor Temperature of Oil and Water separation process**




- Let's say, temperature to be controlled is 50 degree centigrade. According to used sensor RTD PT100, data are processed and required temperature is controlled through Gas Supply for ignition.
- Here display output address is O:7.
- RUNG002 is to operate Oil Outlet Valve with address O:2/0.
- When Level low is detected by Low level sensor with address I:1/1, outlet valve is opened and it is closed when I:1/0 is detected that is high level.

## Runtime Test Cases

```
Inputs          Outputs  Physical Elements
I:1/1 = 1       O:2/0 = 1       Open oil outlet valve
I:1/0 = 1       O:2/0 = 0       Close oil outlet valve
I:3 = 8000h     O:3 = 0000h     Close inlet valve fully
I:3 = 4000h     O:3 = 7FFFh     Inlet Valve 50% open
I:4 = 6018h     O:5 = 0000h     Gas supply valve fully closed
I:4 = 300Ch     O:5 = 8000h     Gas supply valve fully closed
```

# PLC Program to Store Temperature Data of a Process at Different Time

This is a PLC Program to Store Temperature Data of a Process at Different Time.

## Problem Description

Temperature of two tanks are being controlled. To display and compare temperature of both the tanks, graphical display is used. Update temperature of these tanks on the display every 5secs for comparison with respect to time. Implement this process in PLC using Ladder Diagram programming language.

## Problem Solution

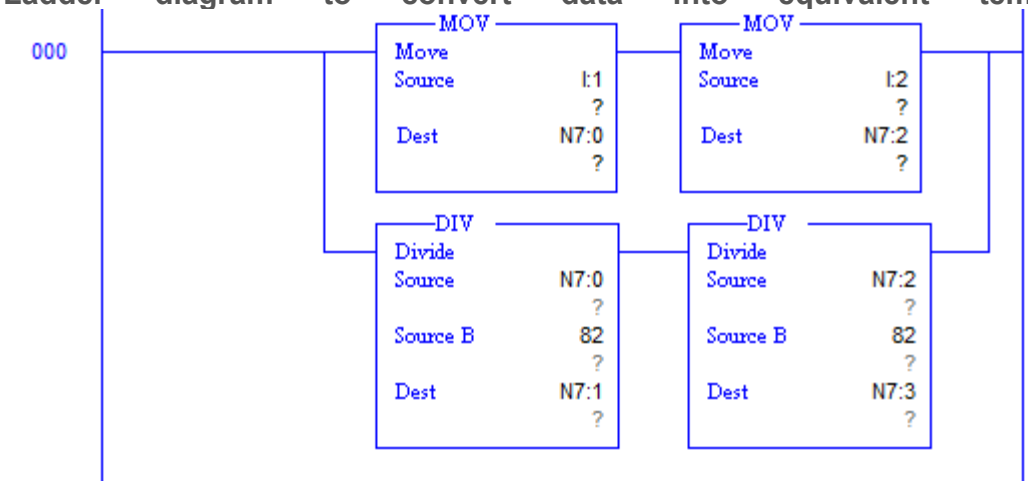
- Using RTD PT100, we can calibrate value 00000 to 65536 into -2500-6500C and divide into 800 equal parts.
- Use MOV instruction to transfer Input data into register for further conversion usage.
- Convert outputs of temperature transmitters into equivalent Temperature in Centigrade.
- To do this, divide input by value change per centigrade, we get temperature inside that tank.

## PLC Program & Program Description

Here is PLC program to Store Temperature Data of a Process at Different Time, along with program explanation and run time test cases.

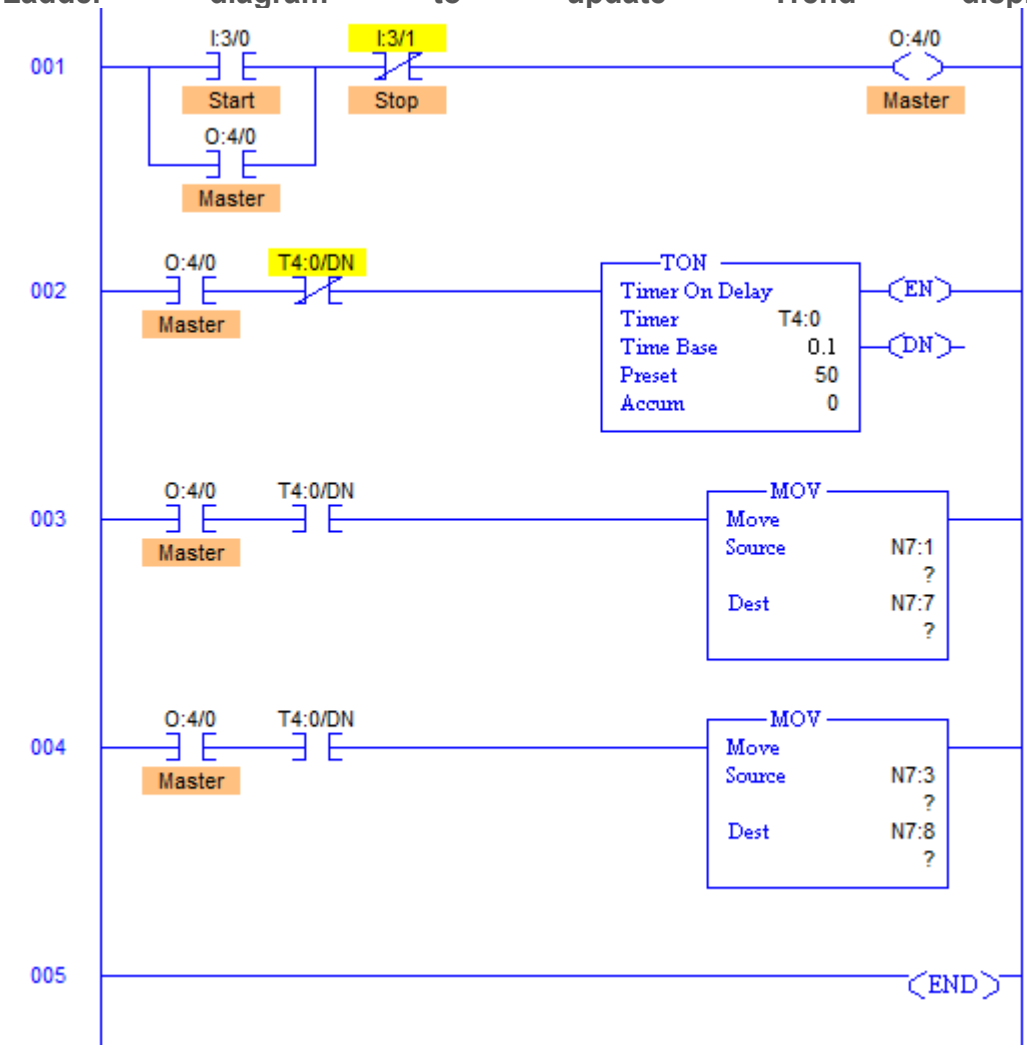
```
List of Inputs and Outputs
 I:1             = Input from temperature transmitter1     (Input)
 I:2             = Input from temperature transmitter2     (Input)
 I:3/0           = Start PB                                (Input)
I:3/1            = Stop PB                                 (Input)
 O:4/0           = Master coil                             (Output)
 N7:0 to N7:8    = Registers to store and process data     (Register)
```

**Ladder diagram to convert data into equivalent temperature in Centigrade**




- When divided by total range of RTD PT100 that is -250 to 650 degree centigrade, we get 82 value change per Centigrade.
- Hence when divided by 82, we get answer in centigrade.
- Important thing to note here is that the input here will always be greater 23124 which is value when temperature is 32 degree centigrade that is room temperature.

**Ladder diagram to update Trend display every 5secs**




- RUNG002, RUNG003 and RUNG004 updates temperature data in corresponding registers every 5secs.
- Timer is in auto reset mode, hence timer will keep running till it is manually stopped by operator.

## Runtime Test Cases

```
Inputs       Outputs                              Physical elements
I:3/0 = 1    O:4/0 = 1                            Start temperature displaying
T4:0/DN = 1  N7:7 = N7:1, N7:8 = N7:3, Accum = 0  Update temperature & reset timer
```

# PLC Program to Store Data of Various Process Sequentially

This is a PLC Program to Store Data of Various Process Sequentially.

## Problem Description

There are total two parameters to be monitored of a tank. Display level and temperature of this tank for 5secs each. Implement this in PLC using Ladder Diagram programming language.

## Problem Solution

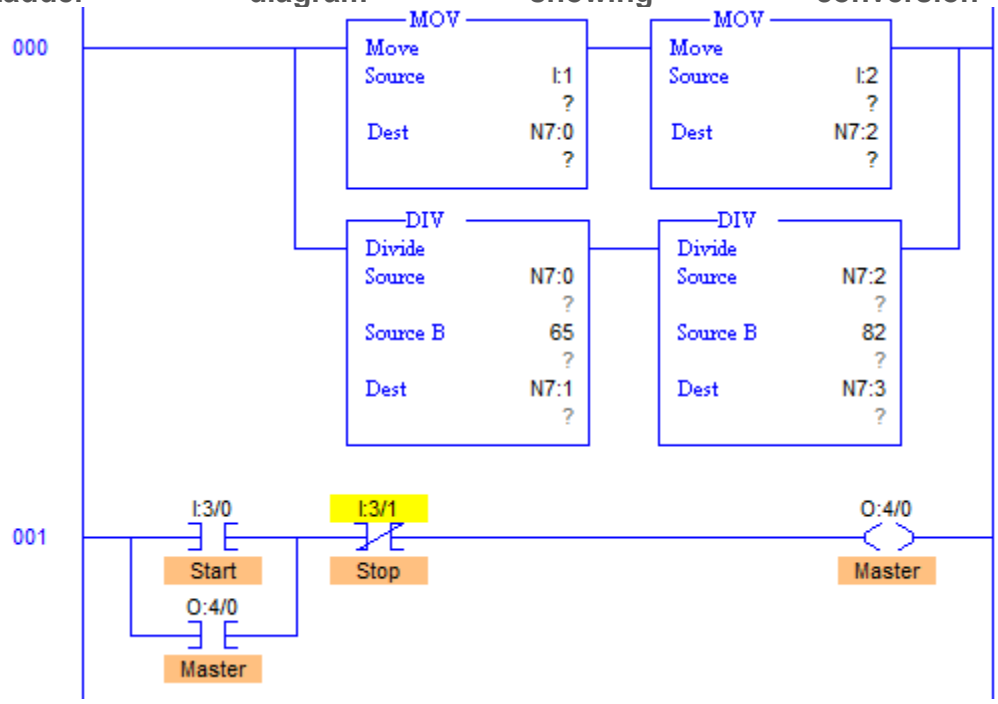
- Continuous measurement is necessary in this case.
- Continuous level measuring devices such as Ultrasonic or Pressure level sensor can be used and for temperature, RTD or Thermocouple can be used.
- Output of these sensors are in analog form, so to deal with such analog data, analog modules are used in PLC.
- Analog modules convert analog signal into equivalent hex form.
- Convert this into Temperature in degree centigrade and level in height in cm.
- Generate time base to MOV output storing register value to display address after converting into BCD equivalent.
- Use either number of timers or Sequential Output instruction to solve this problem.

## PLC Program

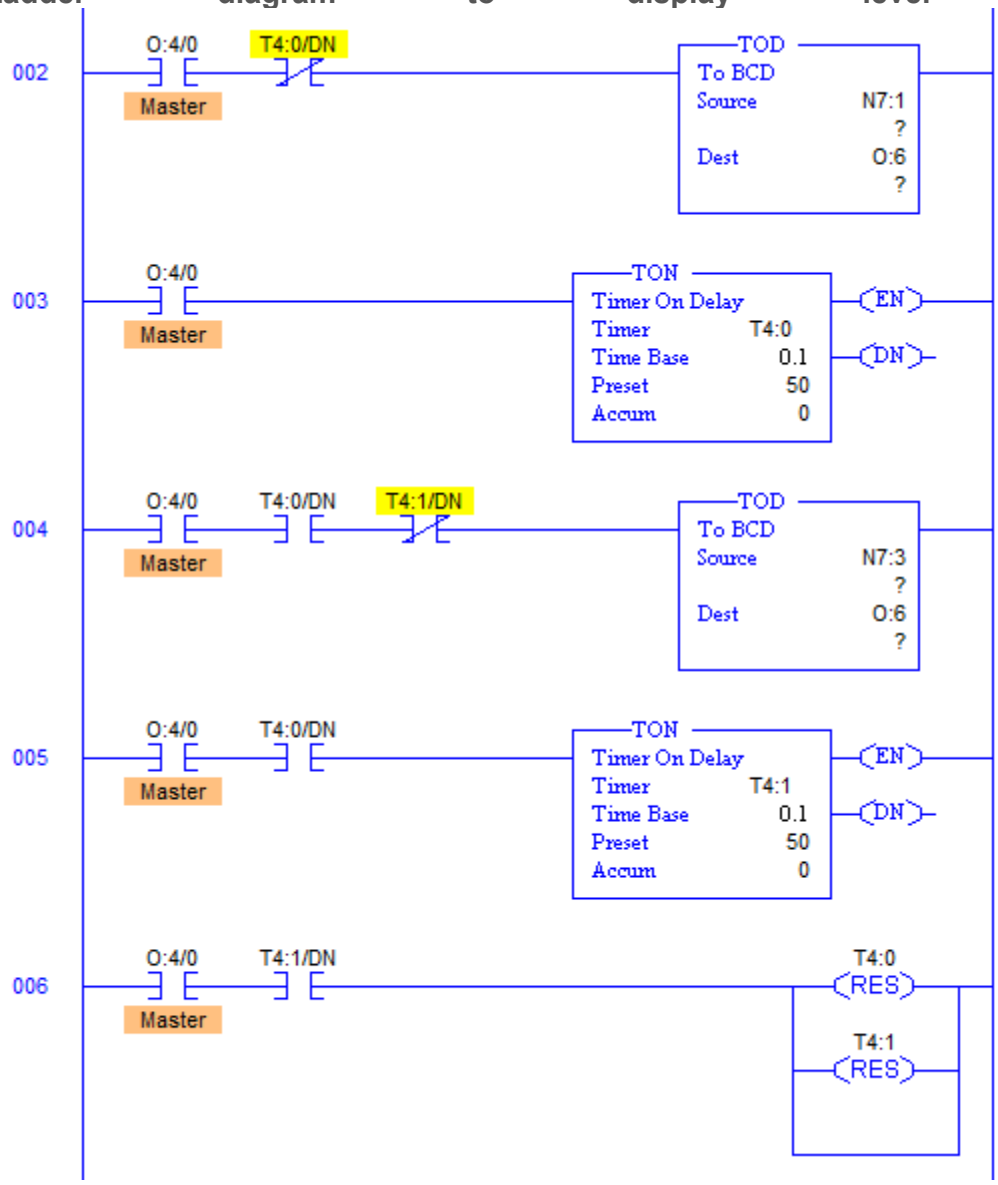
Here is PLC program to Store Data of Various Process Sequentially, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1             = Level Input from Tank                  (Input)
 I:2             = Temperature Input from of the Tank              (Input)
 N7:0 and N7:1   = Store and  process data of tank level           (Register)
 N7:2 and N7:3   = Store and process data of tank Temperature      (Register)
 T4:0, T4:1      = Timers to switch display data every 5secs       (Timers)
```

**Ladder diagram showing conversion of inputs**



**Ladder diagram to display level and temperature**




## Program Description

- When level in tank rises, output of sensor increases accordingly.
- Suppose we have a tank with height 1000cm.
- Converting 16bit data into 1000cm height, 65 is the answer which means, increment of 65 is obtained per centimeter rise in tank level.
- Input data from I:1 is continuously moved to N7:0 which means data in N7:0 register continuously varies.
- Value stored in N7:0 is divided by 65 and stored in register N7:1.
- Value stored in N7:1 register is first converted into equivalent BCD number before feeding it to display which is connected with output module at address O:6.
- Similarly output of temperature is processed.
- Temperature range is 800 (-150 to 650 degree centigrade), hence when input is divided by 82, output is in temperature in degree centigrade.
- Work of timers here is just to change the input register address from which data is to be moved to output display every 5secs.
- After completion of each cycle, timers are reset by RUNG006.
- It just shows two process of a tank, more processes can be displayed similarly.

## Runtime Test Cases

```
Inputs          Outputs  Physical Elements
O:2/0 = 1        O:6 = N7:1      Display level of the Tank
T4:0/DN = 1      O:6 = N7:3      Display temperature of the Tank
T4:1/DN = 1      O:6 = N7:1      Switch back to displaying level of the Tank
```

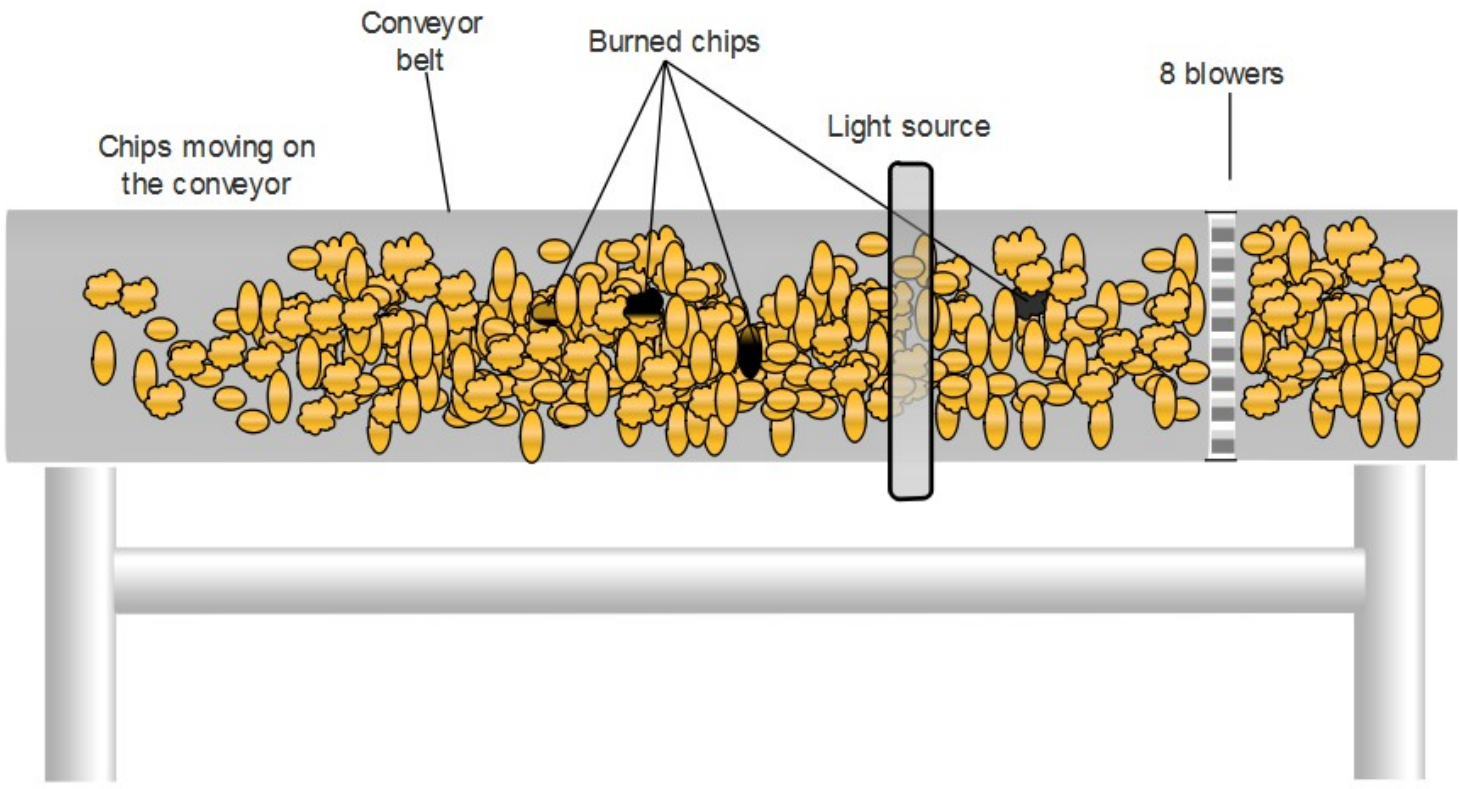
# PLC Program to Detect Burned Chips and Remove Them

This is a PLC Program to Detect Burned Chips and Remove Them.

## Problem Description

Potato chips are made and ready to be packed. But before that, it goes through a conveyor in which final quality check is done, burnt chips are detected and removed from the process line. Implement automation of this process in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- To detect burned chips, light source and sensors are used.
- Light source is used so light detectors such as Light Dependent Resistors are used to detect the burned chips.

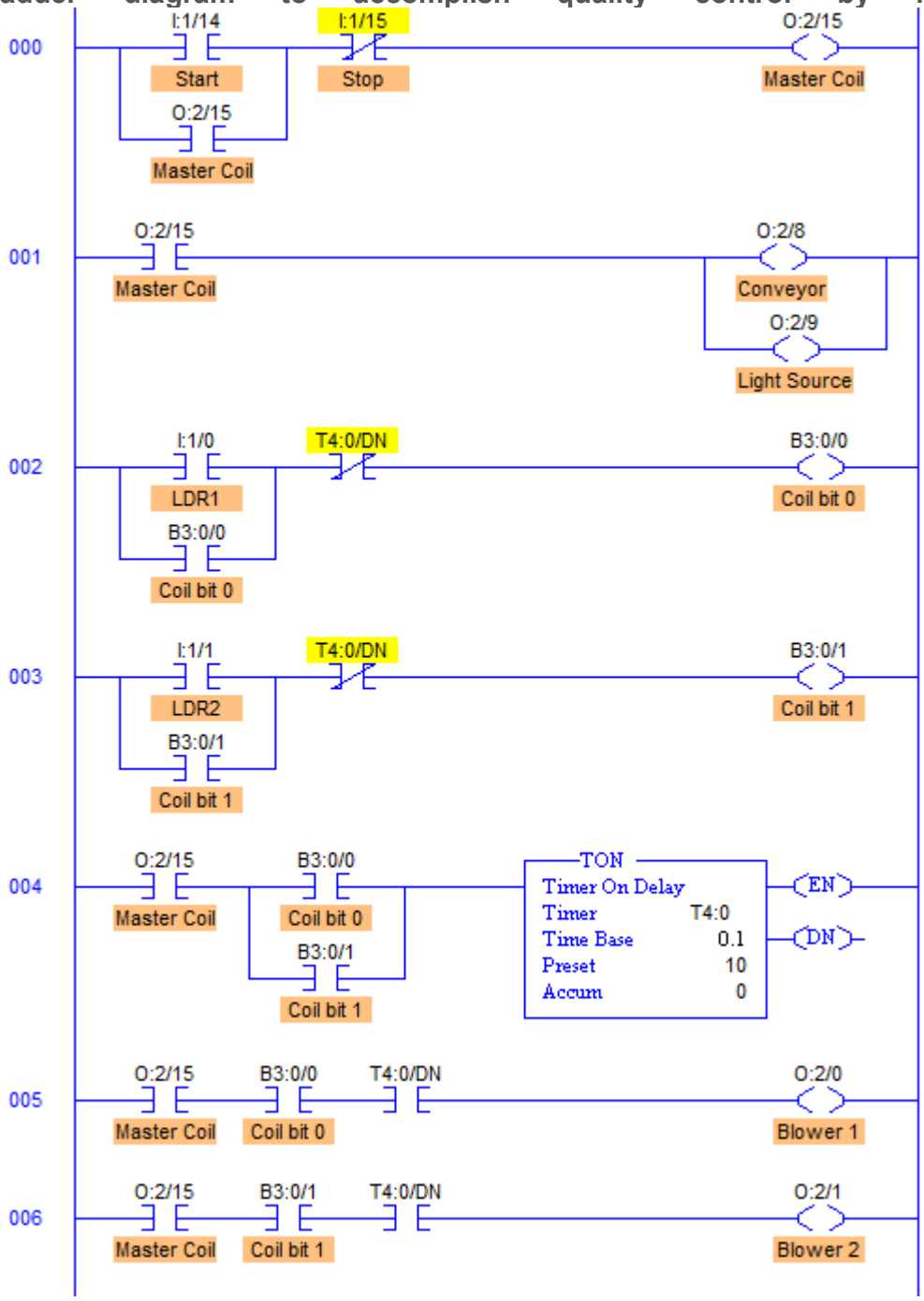
- Blowers are used to throw burned chips away from the conveyor when detected.
- There are total number of 8 blowers. Number of blowers to be used depends on the width of a conveyor belt.
- Time measurement of an event to take place can be used here to measure what time burned chips take to reach from light source to blowers when detected.
- Set this time as preset of a timer to operate particular blower.
- There are 8 blowers, so 8 light detecting circuits must be used in order to operate all blowers.
- Let us assume we are using Light Dependent Resistor. To use this resistor, threshold has to be set that is darkest color to be passed as a good quality product. If chips are darker than the desired level, light source detects it and activates corresponding circuit.
- So output of this circuit is normally high and to activate blower, normally low logic has to be set while programming or we can even invert output from LDR circuit.

## PLC Program

Here is PLC program to Detect Burned Chips and Remove them, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14            = Start                                  (Input)
 I:1/15            = Stop                          (Input)
 O:2/15            = Master Coil                            (Input)
 O:2/8             = Conveyor                               (Output)
 O:2/9             = Light Source                  (Output)
 I:1/0 to I:1/7    = Light Dependent Resistors              (Input)
 B3:0/0 to B3:0/7 = Bit latching                   (Bits)
 O:2/0 to O:2/7    = Blowers                                (Output)
 T4:0              = Timer to activate blower after 1sec   (Timer)
```

**Ladder diagram to accomplish quality control by removing defected chips**




## Program Description

- RUNG001 operates conveyor and light source. When Start PB is pressed, it energizes conveyor motor coil and light source.
- There are total 8 LDR circuits are mounted next to each other. LDR1 activates Blower1, LDR2 activates Blower2 and so on.
- Ladder diagram has shown just two LDRs and Blowers. Further programming for all other blowers and LDRs remain same.
- When any defected part is detected, for example if LDR1 detects a defected part, then it latches the corresponding coil bit B3:0/0 which in turn activates timer.
- 1sec is assumed time taken by a part to reach from Light Source to Blower and hence timer has preset 10.

- When 1sec is over, it activates Blower1 with address O:2/0. Blower energizes and immediately de-energizes giving just a push to detected part.
- Similarly it works for all LDR circuits.
- Many PLCs provide Time Base of 0.01 which may be used for more accuracy.

## Runtime Test Cases

```
Inputs                    Outputs                    Physical Elements
I:1/14 = 1                O:2/8 = O:2/9 = 1          Run conveyor and turn ON lights
I:1/0 = 1 & T4:0/DN = 1   O:2/0 = 1 (Momentarily)   Blower1 energizes momentarily
I:1/1 = 1 & T4:0/DN = 1   O:2/1 = 1 (Momentarily)   Blower2 energizes momentarily
_____________________Similarly all other blowers are operated_____________
```

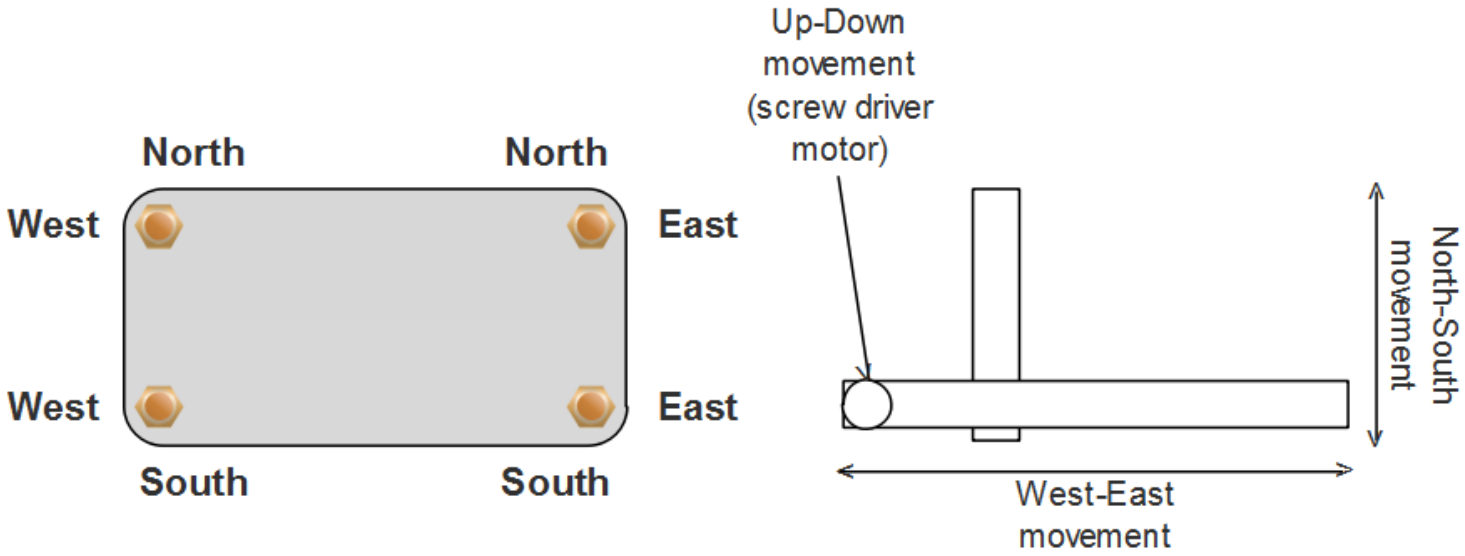
# PLC Program to Operate Screwing of Parts

This is a PLC Program to Operate Screwing of Parts.

## Problem Description

When a plate is placed, screws are to be fit at each corner of the plate. Use robotic arm to perform this action and implement automation of this in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- Robotic arm used to perform this screwing action.
- Robotic arm used here is of 3-axis arm.
- Sequence of the arm is to be controlled.
- SQO instruction must be used in order to fulfill the requirements.
- Sequence to be followed is as given below.

```
1-  Arm is initially at South-East position
2-  Arm lowers and screws.
```

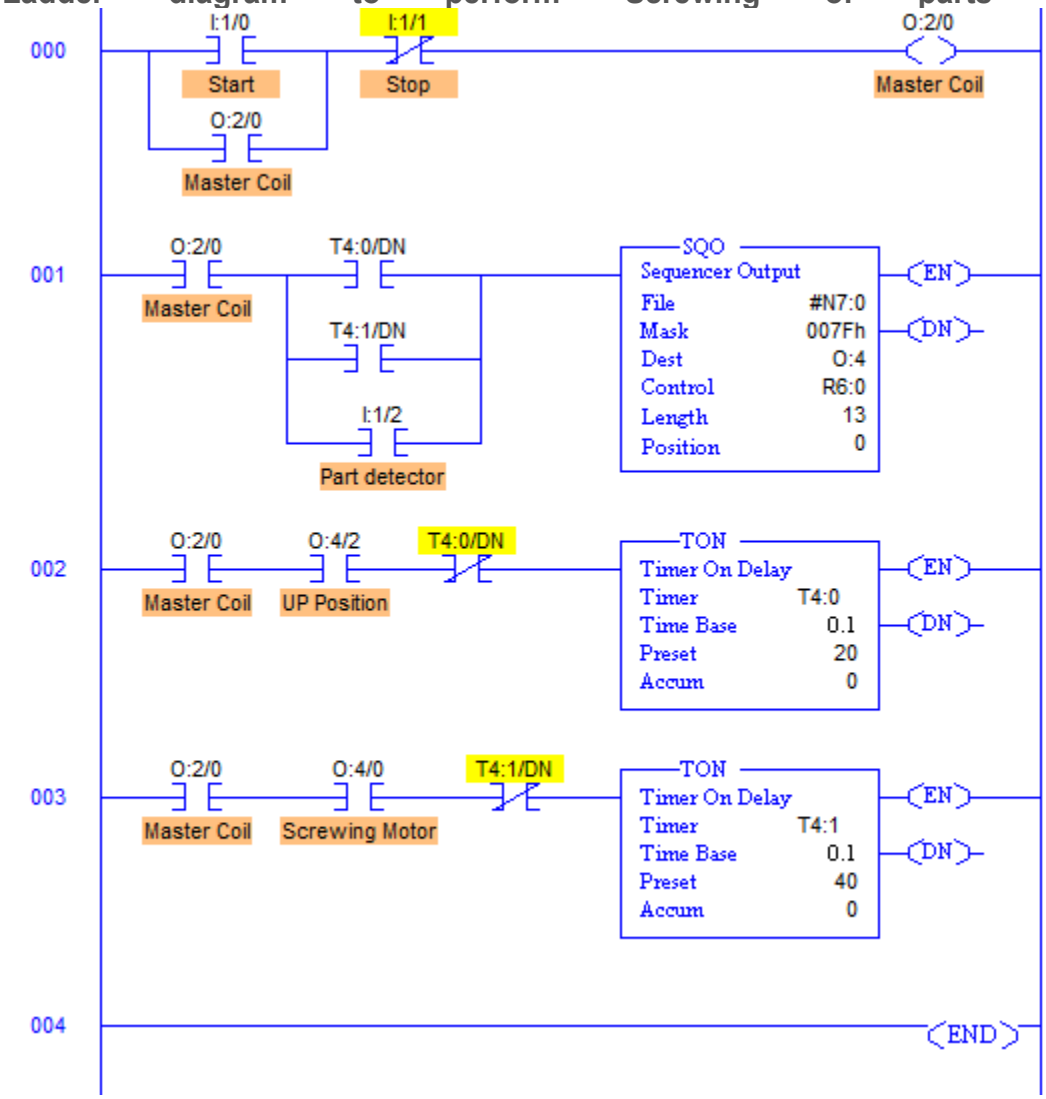
```
  3-  Raise arm.
  4-  Move to West-South position
  5-  Arm lowers and screws.
  6-  Raise arm.
  7-  Move to West-North position.
  8-  Arm lowers and screws.
  9-  Raise arm.
  10- Move to North-East position.
  11- Arm lowers and screws.
  12- Raise arm.
  13- Move to South-East position.
```

## PLC Program

Here is PLC program to Operate Screwing of Parts, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0           = Start PB                       (Input)
 I:1/1           = Stop PB                        (Input)
 I:1/2           = Part detector proximity        (Input)
 O:2/0           = Master coil                    (Output)
 O:4/0           = Screwing Motor coil            (Output)
 O:4/1           = Down position                 (Output)
 O:4/2           = Up position                    (Output)
 O:4/3           = West position                  (Output)
 O:4/4           = East position                  (Output)
 O:4/5           = South position         (Output)
 O:4/6           = North  position        (Output)
 N7:0            = File register                  (Register)
 N7:1 to N7:13   = Sequence storing registers     (Register)
 R6:0            = Control Register               (Register)
```

**Ladder diagram to perform Screwing of parts with robotized arm**




## Program Description

- RUNG000 again here is for Master Start and Stop.
- File; #N7:0 and File length is 13, hence output sequence is varied from N7:0 to N7:12 with each input.
- Destination is set to O:4 hence with each transition, N7:0 to N7:12 are moved to O:4 with masking.
- O:4/0 to O:2/6 are used as the output address to Different positions and Motor coil, mask has value 007Fh which means data flow of N7:0/0…N7:12/0 to N7:0/5…N7:10/5 is passed and the remaining N7:0/7…N7:12/7 to N7:0/15…N7:12/15 are blocked.
- Control parameters are assigned to register R6:0.
- Sequence of robotic arm to be operated are stored in the registers from N7:0 to N7:12 as following.

| Step-by-step sequence of operation | | | | | | | | | | NORTH | SOUTH | EAST | WEST | UP | DOWN | MOTOR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N7:0/ | o | o | o | o | o | o | o | o | o | o | o | o | o | o | o | o |
| N7:1/ | o | o | o | o | o | o | o | o | o | o | 1 | 1 | o | o | 1 | 1 |
| N7:2/ | o | o | o | o | o | o | o | o | o | o | 1 | 1 | o | 1 | o | o |
| N7:3/ | o | o | o | o | o | o | o | o | o | o | 1 | o | 1 | 1 | o | o |
| N7:4/ | o | o | o | o | o | o | o | o | o | o | 1 | o | 1 | o | 1 | 1 |
| N7:5/ | o | o | o | o | o | o | o | o | o | o | 1 | o | 1 | 1 | o | o |
| N7:6/ | o | o | o | o | o | o | o | o | o | 1 | o | o | 1 | 1 | o | o |
| N7:7/ | o | o | o | o | o | o | o | o | o | 1 | o | o | 1 | o | 1 | 1 |
| N7:8/ | o | o | o | o | o | o | o | o | o | 1 | o | o | 1 | 1 | o | o |
| N7:9/ | o | o | o | o | o | o | o | o | o | 1 | o | 1 | o | 1 | o | o |
| N7:10/ | o | o | o | o | o | o | o | o | o | 1 | o | 1 | o | o | 1 | 1 |
| N7:11/ | o | o | o | o | o | o | o | o | o | 1 | o | 1 | o | 1 | o | o |
| N7:12/ | o | o | o | o | o | o | o | o | o | o | 1 | 1 | o | 1 | o | o |

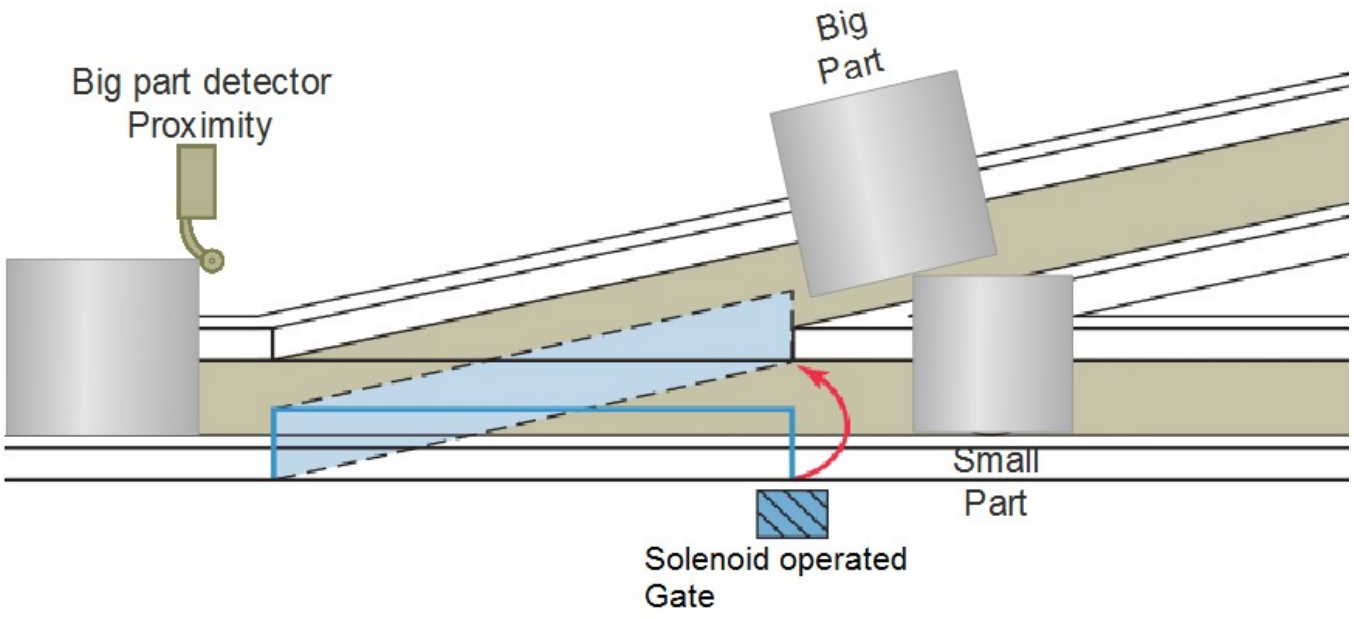
# PLC Program to Separate Big and Small Parts

This is a PLC Program to Separate Big and Small Parts.

## Problem Description

Two different sized particles are being moved on the conveyor belt. To pack these particles, two different boxes are used separately. These products must not be packed in the same box. Separation of these particles is to be controlled. Implement automation to perform this operation in PLC using Ladder Diagram programming language.

## Problem Diagram



## Problem Solution

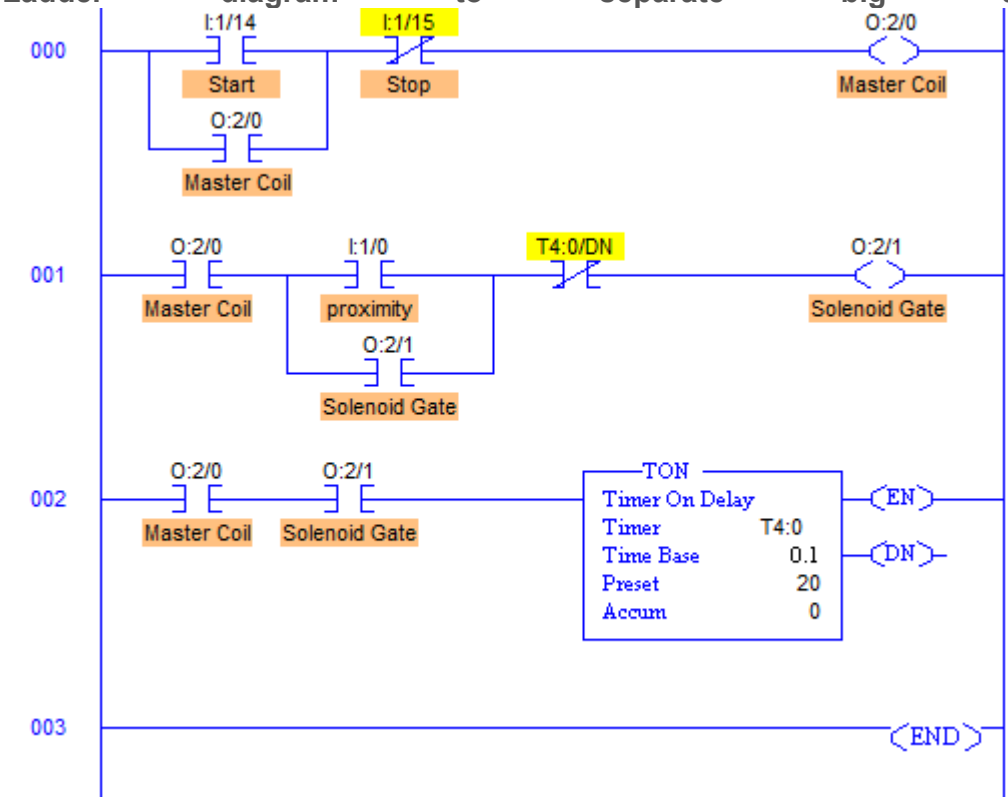
- Small parts do not need to be detected as it has a straight conveyor path.
- Big parts are to be diverted to the adjacent conveyor which is connected with the main conveyor as shown in diagram above.
- Detection of big particles has to be done in order to divert the path of such particles.
- To detect big parts, Proximity Limit switch is used.
- Conveyor has a fixed speed, hence timer is to be set to energize Gate Solenoid which diverts big parts to the adjacent conveyor.

## PLC Program

Here is PLC program to Separate Big and Small Parts, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                          (Input)
 I:1/15  = Stop                           (Input)
 O:2/0   = Master Coil                    (Output)
 I:1/0   = Proximity to detect big parts  (Input)
 O:2/1   = Gate solenoid coil             (Output)
 T4:0    = Timer to operate gate          (Timer)
```

**Ladder diagram to separate big and small parts**




## Program Description

- RUNG000 is to Master Start and Stop the separation process.
- RUNG001 and RUNG002 operates solenoid coil.
- When Proximity switch detects a big part, it energizes and latches Solenoid Coil which is connected to O:2/1.
- Let us assume it takes approximately 2secs for a big part to transfer from main conveyor to the adjacent one.

- When Solenoid coil is energized, Timer T4:0 is started.
- Solenoid coil remains energized for 2secs that is when part is being diverted to the adjacent conveyor.
- After 2secs, when a big part is diverted to the adjacent conveyor, XIO of T4:0/DN de-energizes O:2/1 and the Gate is returned back to its main position to let Small parts move on the main conveyor.

## Runtime Test Cases

| Inputs | Outputs | Physical Elements |
|---|---|---|
| I:1/14 = 1 | O:2/0 = 1 | Start separation process |
| I:1/0 = 1 | O:2/1 = 1 | Energize solenoid gate |
| T4:0/DN = 1 | O:2/1 = 0 | De-energize solenoid gate |

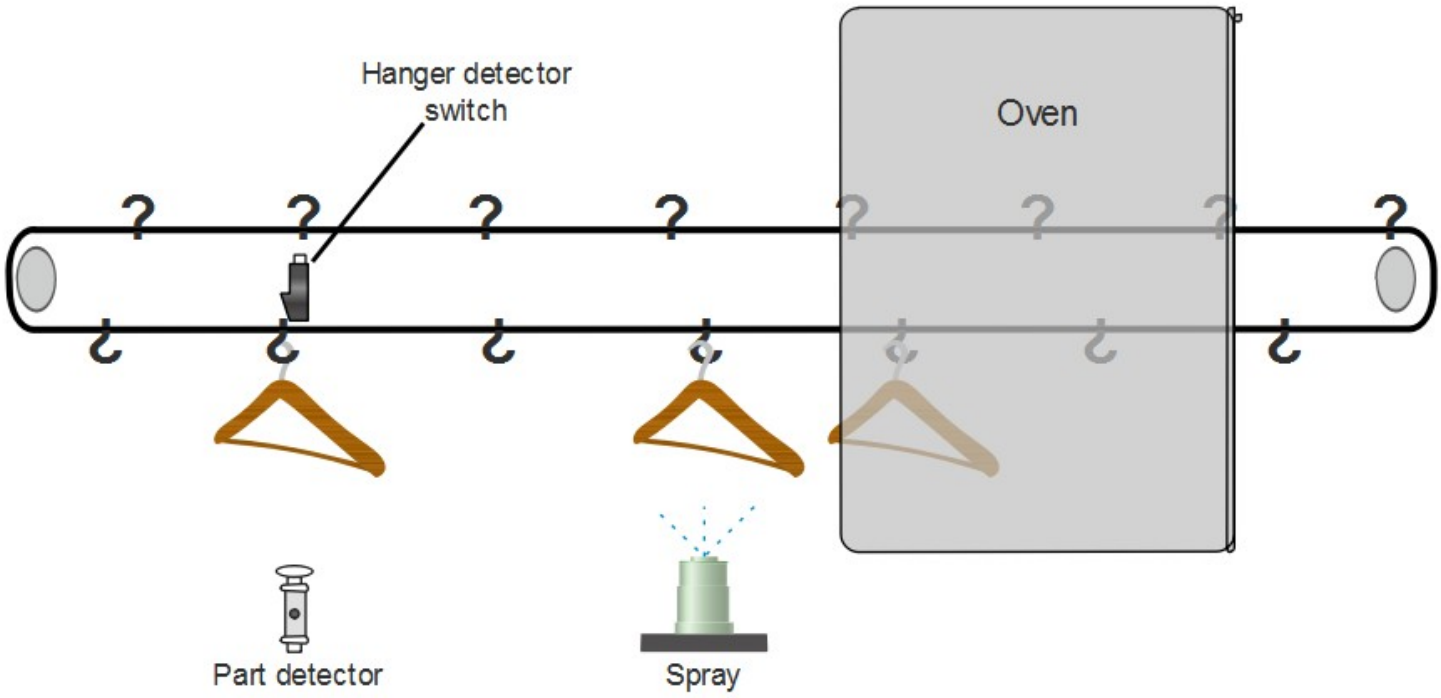
# PLC Program to Control Spray-Painting of Parts

This is a PLC Program to Control Spray-Painting of Parts.

## Problem Description

Control Spray-Painting operation in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- Spray-Paining operation is similar to the bottle capping operation. The only difference is that Bottle Capping operation used timers while Spray-Paining operation do not require timers.
- Two limit switches are used, one to detect hangers and other to detect parts.
- Both are mounted exactly opposite to each other so that when part is not present, outputs are complement of each other.

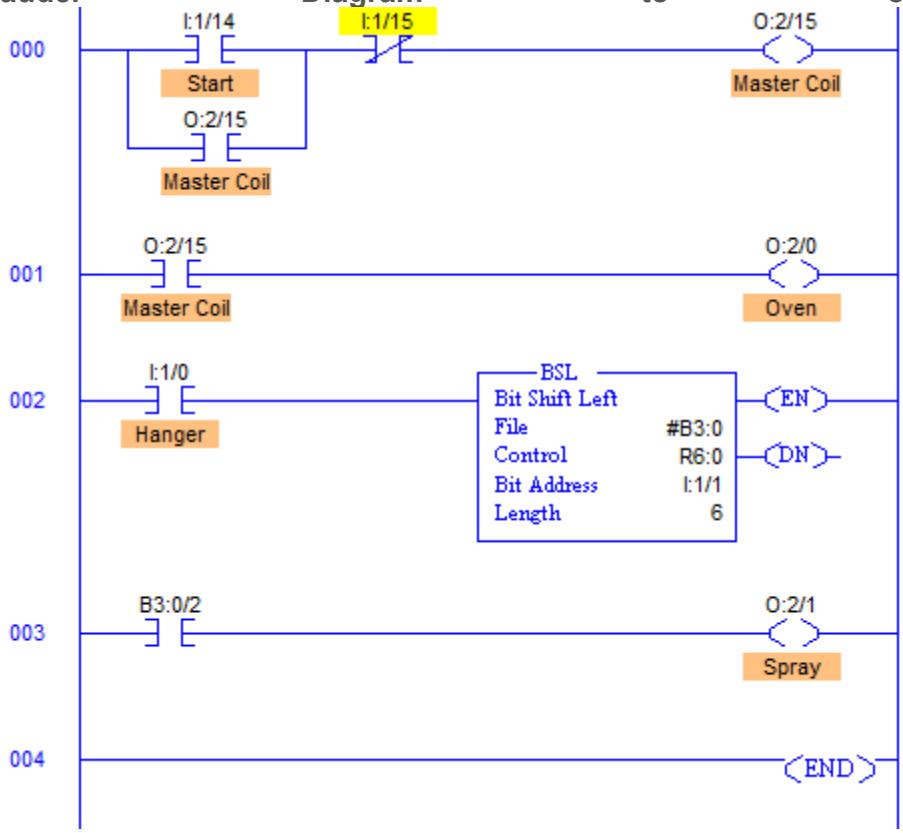
- Bit shift register is used to spray parts.
- Oven is used to dry the parts after they are sprayed. This output is continuously ON and is not controlled.

## PLC Program

Here is PLC program to Control Spray-Painting of Parts, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                                  (Input)
 I:1/15  = Stop                                   (Input)
 I:1/0   = Limit switch (Hanger)                  (Input)
 I:1/1   = Proximity (Part detection)             (Input)
 O:2/0   = Master coil / Run                      (Output)
 O:2/1   = Oven                                   (Output)
 O:2/2   = Spray                                  (Output)
 BSL     = Bit shift left instruction             (Logical)
 B3:0    = Bit shift Register                     (Register)
 B3:0/2  = Bit to energize capping machine (Bit)
 R6:0    = Control register                       (Register)
```

**Ladder Diagram to operate Spray-Paining**




## Program Description

- When Start PB is pressed, Oven with address O:2/0 is started.
- RUNG002 and RUNG003 are used to operate bit shift register and Spray with address O:2/1.
- I:1/0 and I:1/1 are mounted such that they are detected together. When Part is present, I:1/0 and I:1/1 both goes high setting bit B3:0/0. Shifting of this itself is operated by I:1/0 Hanger switch. So every time a hanger is detected, bit is shifted left.
- From proximity to capping machine, station distance is 3 steps. Hence bit B3:0/2 of B3:0 register operates spray solenoid.
- R6:0 is a control register where all the operations are handled.

## Runtime Test Cases

```
Inputs              Output        Physical Elements
I:1/14 = 1          O:2/0 = 1     Turn ON oven
I:1/0 = 1           BSL = 1       Shift bit to left
I:1/0 = I:1/1 = 1   B3:0/0 = 1    Set bit to operate Spray
B3:0/2 = 1          O:2/1 = 1     Activate spray
```

# PLC Program to Display Level of Three or More Tanks

This is a PLC Program to Display Level of Three or More Tanks.

## Problem Description

There are total 3 tanks of which level is being controlled. Display level of these tanks for 5secs of one by one. Implement this in PLC using Ladder Diagram programming language.

## Problem Solution

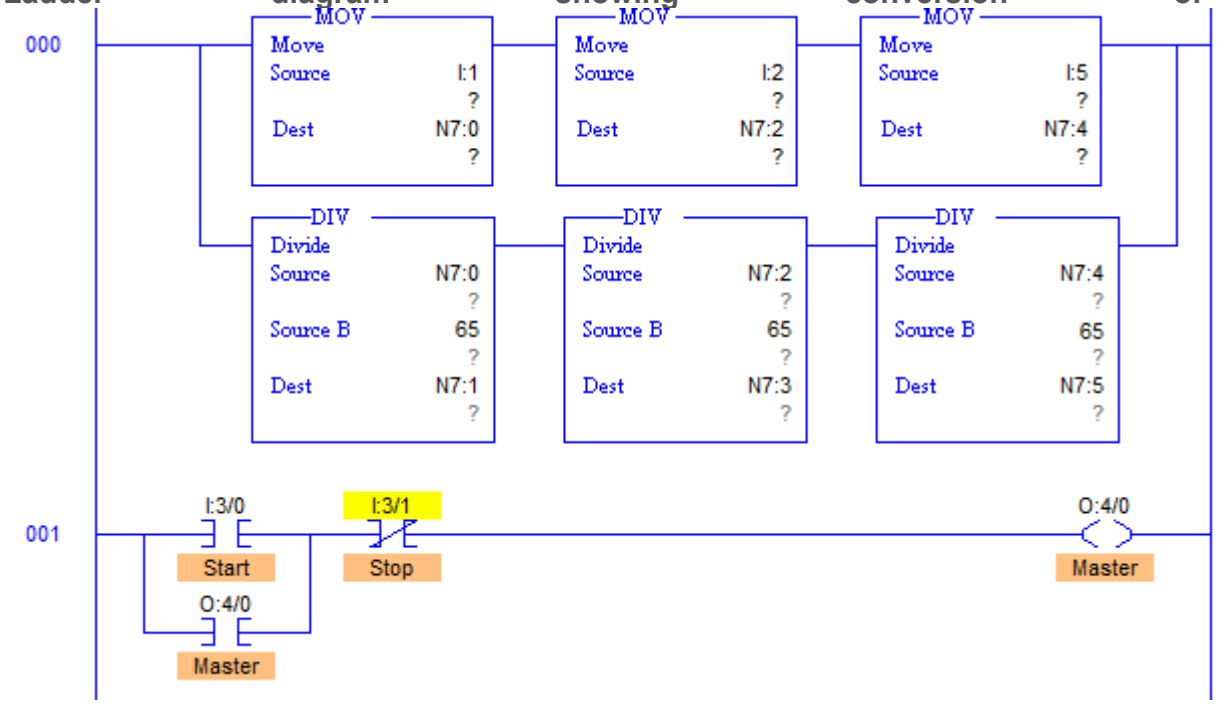
- Continuous measurement is necessary in this case.
- Continuous level measuring devices such as Ultrasonic, Radar, Capacitive or Pressure gauge level sensor can be used.
- Output of these sensors are in analog form, so to deal with such analog data, analog modules are used in PLC.
- Analog modules convert analog signal into equivalent hex form.
- Convert this into level height in Meter or in any other parameter to display on the screen.
- Generate time base to MOV output storing register value to display address after converting into BCD equivalent.
- Use either number of timers or Sequential Output instruction to solve this problem.

## PLC Program

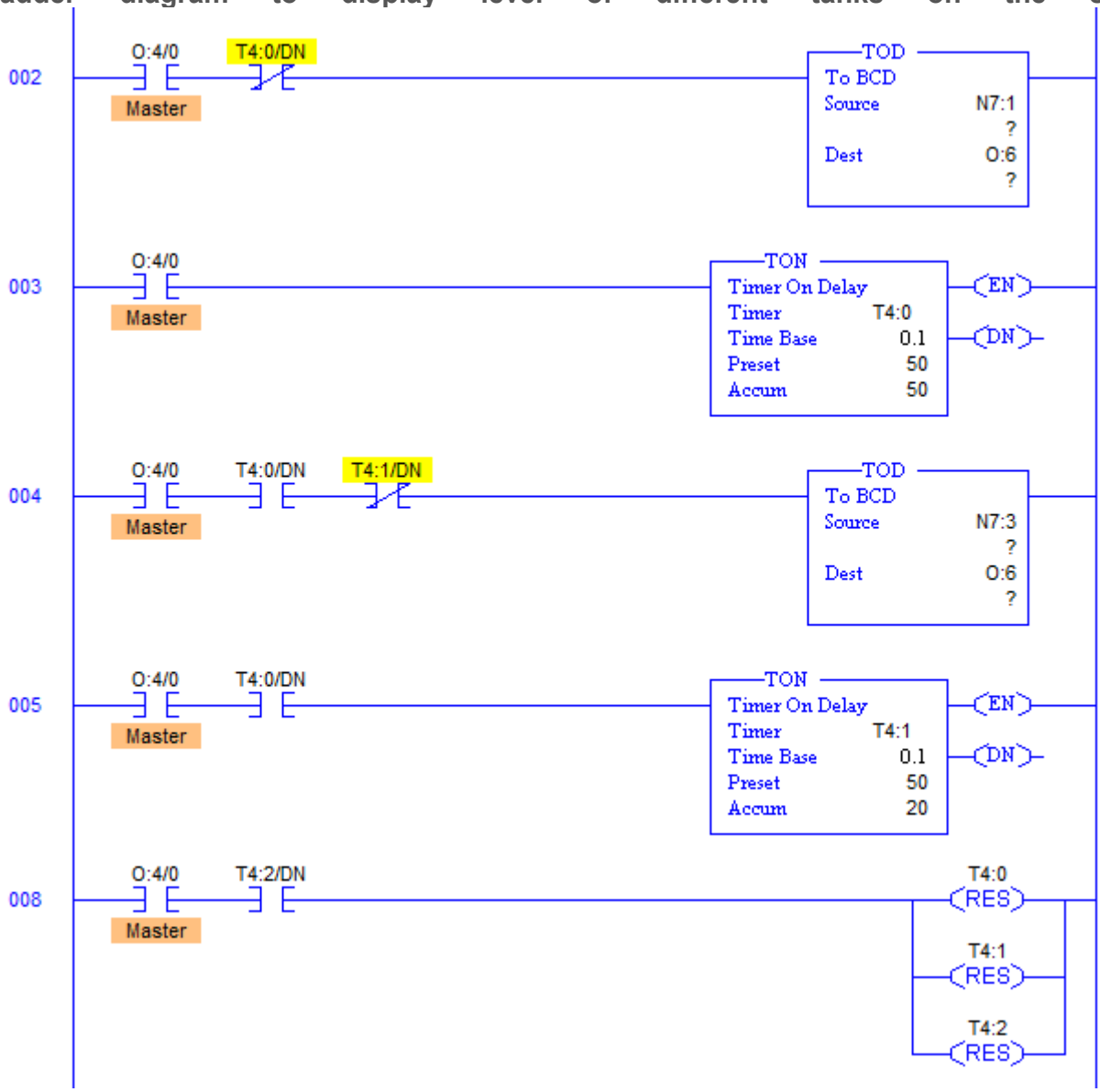
Here is PLC program to Display Level of Three or More Tanks, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1             = Input from Tank 1                              (Input)
 I:2             = Input from Tank 2                              (Input)
 I:3             = Input from Tank 3                              (Input)
 N7:0 and N7:1   = Store and  process data of tank 1              (Register)
 N7:2 and N7:3   = Store and process data of tank 2               (Register)
 N7:4 and N7:5   = Store and process data of tank 3               (Register)
 T4:0,T4:1,T4:2 = Timers to switch display data every 5secs       (Timers)
```

Ladder diagram showing conversion of input



**Ladder diagram to display level of different tanks on the same display**




## Program Description

- When level in tank rises, output of sensor increases accordingly.
- Suppose we have a tank with height 1000cm which is 10m.
- Converting 16bit data into 1000cm height, 65 is the answer which means, increment of 65 is obtained per centimeter rise in tank level.
- Input data from I:1 is continuously moved to N7:0 which means data in N7:0 register continuously varies.
- Value stored in N7:0 is divided by 65 and stored in register N7:1.
- Value stored in N7:1 register is first converted into equivalent BCD number before feeding it to display which is connected with output module at address O:6.
- Similarly output of remaining two tanks are processed.
- Work of timers here is just to change the input register address from which data is to be moved to output display after every 5secs.
- After completion of each cycle, timers are reset by RUNG008.
- Ladder diagram shows operation timers for just two tanks, one more timer should be added to display level of third tank.
- This problem can also be solved by using Sequential Output instruction SQO.

## Runtime Test Cases

```
Inputs          Outputs  Physical Elements
O:2/0 = 1       O:6 = N7:1      Display level of Tank1
T4:0/DN = 1     O:6 = N7:3      Display level of Tank2
T4:1/DN = 1     O:6 = N7:5      Display level of Tank3
T4:2/DN = 1     O:6 = N7:1      Switch back to display level of Tank1
```

# PLC Program to Display Level of a Tank

This is a PLC Program to Display Level of a Tank.

## Problem Description

There is a tank of which level is controlled automatically. Display level of this tank continuously. Perform this in PLC using Ladder Diagram programming language.

## Problem Solution

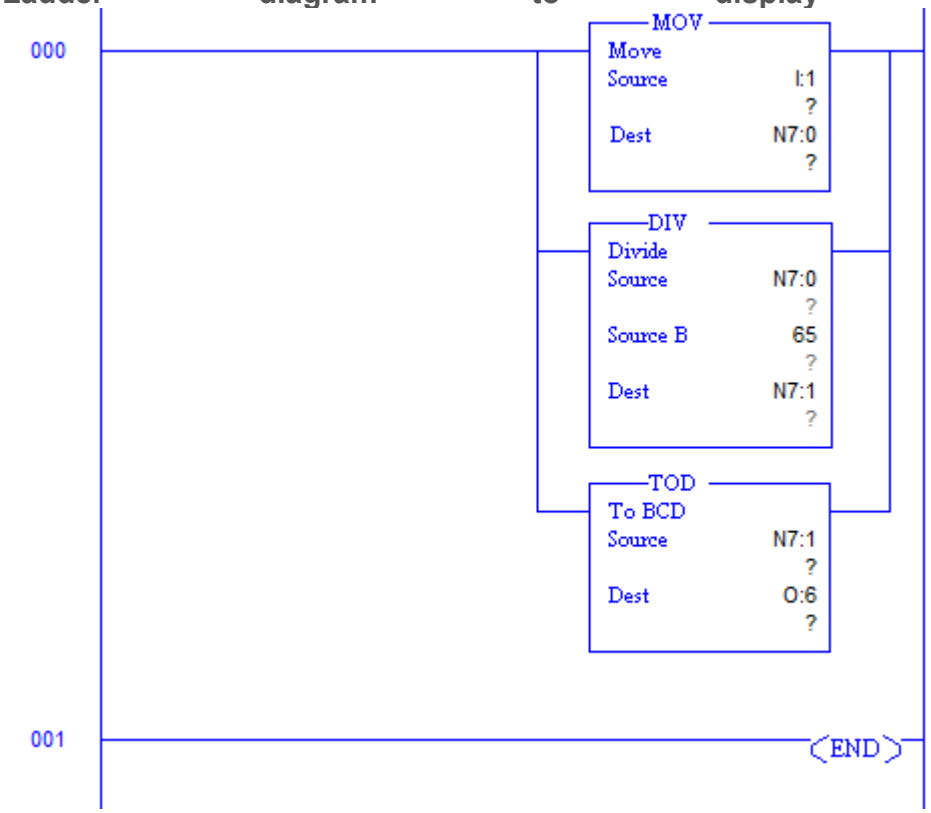
- Continuous measurement is necessary in this case.
- Continuous level measuring devices such as Ultrasonic, Radar, Capacitive or Pressure gauge level sensor can be used.
- Output of these sensors are in analog form, so to deal with such analog data, analog modules are used in PLC.
- Analog modules convert analog signal into equivalent hex form.
- Convert this into level height in Meter or in any other parameter to display on the screen.

## PLC Program

Here is PLC program to Display Level of a Tank, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1    = Input from transmitter (Input)
 N7:0   = Register to store input data   (Register)
 N7:1   = Register data to be displayed  (Register)
 O:6    = Display address                (Output)
 TOD    = Decimal into BCD conversion    (Compute)
```

**Ladder diagram to display level of tank**




## Program Description

- There is no any input is given to start the process since we want continuous displaying of tank level.
- When level in tank rises, output of sensor increases accordingly.
- Suppose we have a tank with height 1000cm which is 10m.
- Converting 16bit data into 1000cm height, 65 is the answer which means, increment of 65 is obtained per centimeter rise in tank level.
- Input data from I:1 is continuously moved to N7:0 which means data in N7:0 register continuously varies.
- Value stored in N7:0 is divided by 65 and stored in N7:1 register to display tank level in Centimeters.
- Value stored in N7:1 register is first converted into equivalent BCD number before feeding it to display which is connected with output module at address O:6.
- Accuracy of this method for 1000cm is ±0.8%.

## Runtime Test Cases

```
Input           Output                    Physical Elements
I:1 = 65536     N7:0 = FFFFh                      Store input to register
I:1 = 65536     N7:1 = 03F0h, O:6 = 1008(In cm)   Displays 1008cm on the screen
I:1 = 32768     N7:1 = 01F8h, O:6 = 504(In cm)    Displays 504cm on the screen
```

# PLC Program to Drain Two Different Products from 2 Tanks

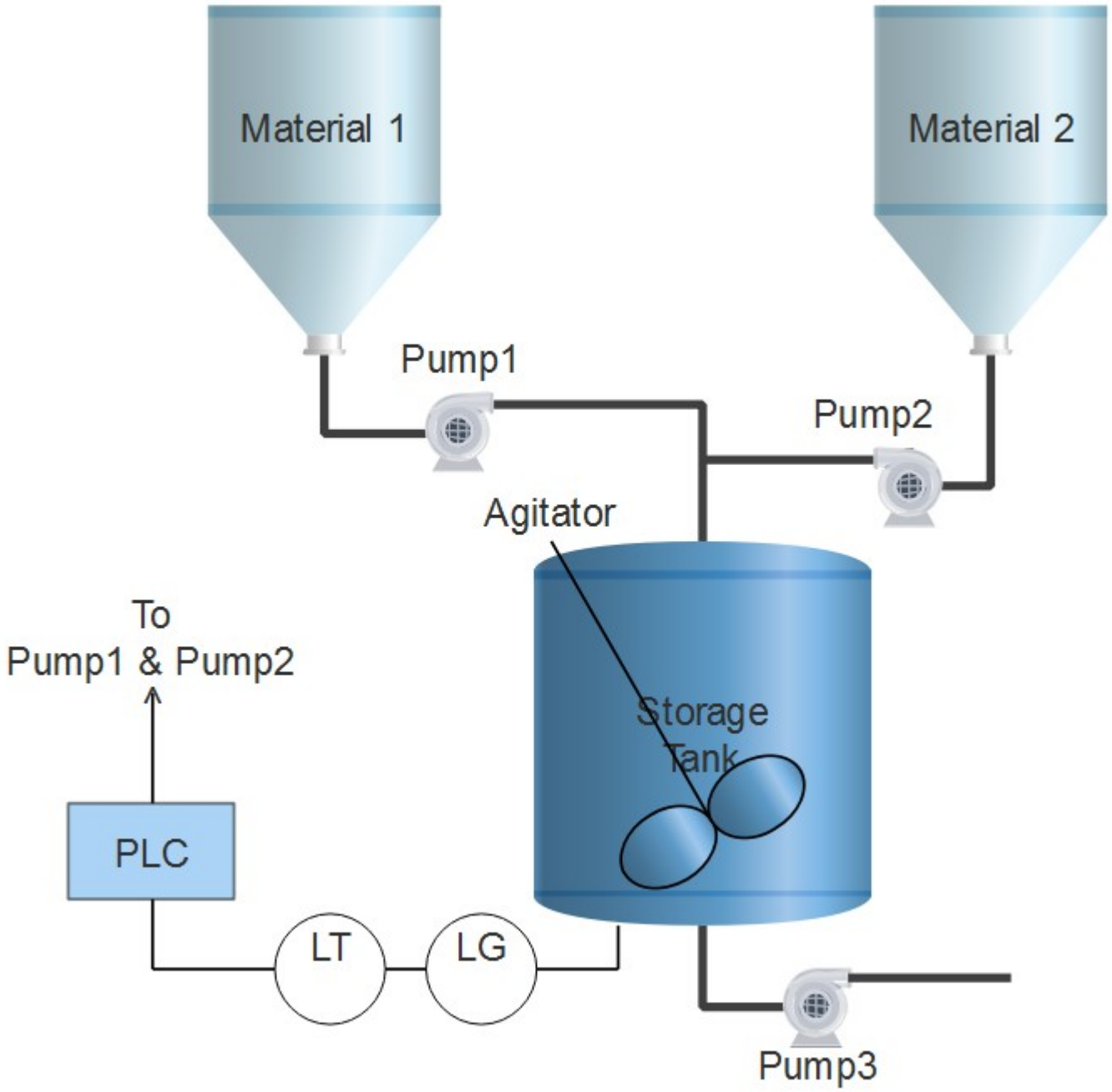
This is a PLC Program to Drain Two Different Products from 2 Tanks.

## Problem Description

There two different tanks which are to be operated. Draining of these two materials are controlled. Mixing in the ratio of 2:1 is to be done. Implement automation of this in PLC using Ladder Diagram programming language.

## Problem Diagram

**Diagram showing mixing of two materials continuously**




## Problem Solution

- Level gauge is used to measure level of the storage tank continuously
- Level gauge is connected with Level Transmitter which converts corresponding level output in 4-20mA equivalent.
- Analog I/O Modules are chosen to deal with Analog signals.

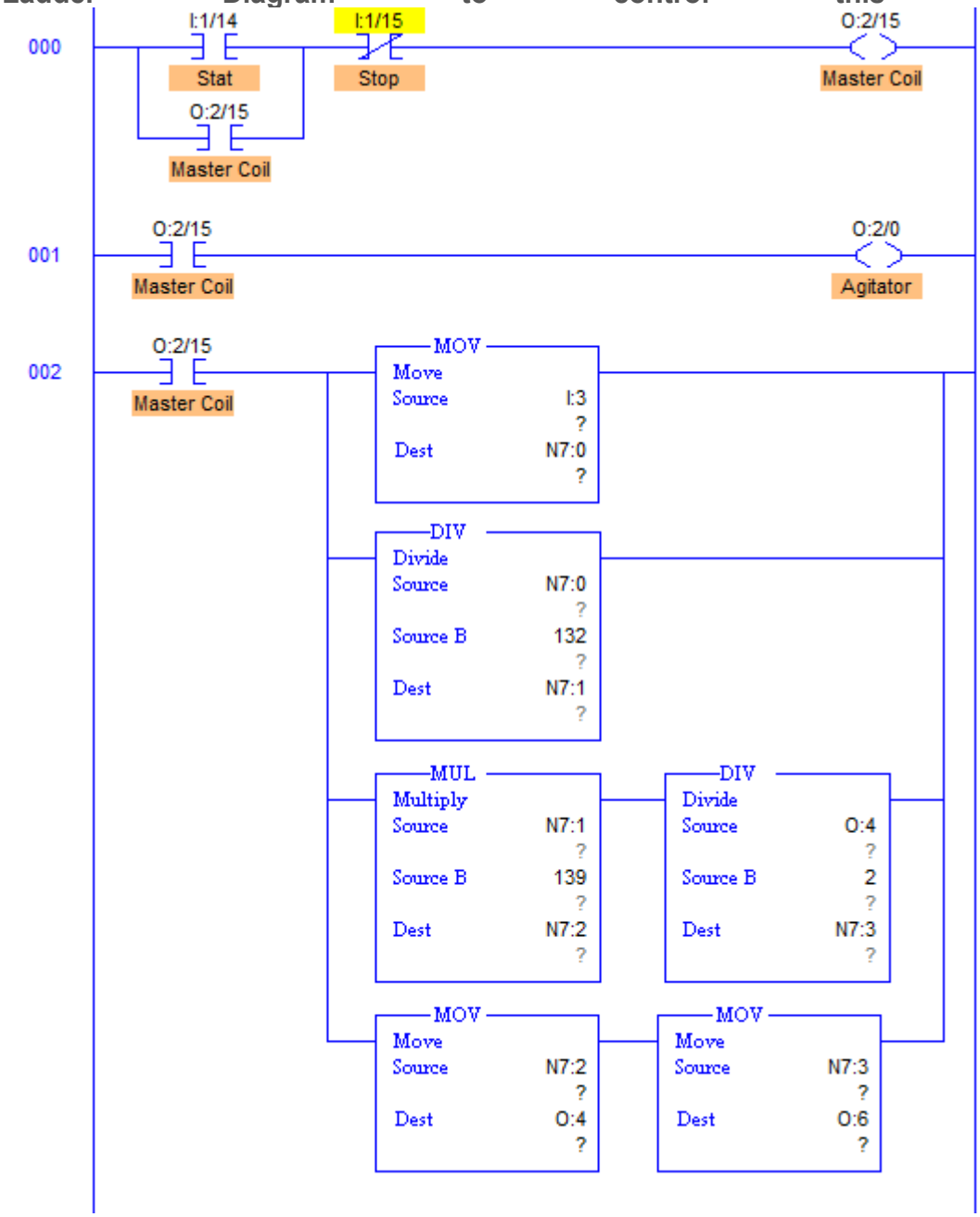
- Centrifugal pumps are used to drain material from both the tanks at the same time.
- Height of storage tank is 5meters that is 500cm and the level which is to be maintained is 470cm.
- Calculate necessary conversions and use registers to store data and to do arithmetic operations.

## PLC Program

Here is PLC program to Drain Two Different Products from 2 Tanks, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                                                  (Input)
 I:1/15  = Stop                                                   (Input)
 O:2/15  = Master Coil                                            (Output)
 O:4     = Output to I-V converter of Pump1                       (Output)
 O:6     = Output to I-V converter of Pump2                       (Output)
 O:2/0   = Agitator to mix                                        (Output)
 I:3     = Input to which transmitter is connected         (Input)
 N7:0    = Register to store input data                           (Register)
 N7:1    = answer of division by value change per centimeter      (Register)
 N7:2    = Multiplication answer                                  (Register)
 N7:3    = Division to obtain 2:1 ratio                           (Register)
```

**Ladder Diagram to control this mixing process**




## Program Description

- RUNG001 comprises all the conversion needed to control pumps.
- Output of transmitter is in current signals which is 4-20mA.
- When output is 4mA, Analog Input Module converts it into 16bit equivalent hex numbers. Hence when input at I:3 to Analog module is 4mA, it moves 0000h into register and when 20mA, it moves FFFFh into register. Here register N7:0.
- Here height of the tank is 5m or 500cm. By converting it into equivalent hex, change in value per centimeter is 132.
- Value of N7:0 is then multiplied with 139 because when Level reaches 470cm, output is F0C0h. So when output at 470cm is multiplied with 139, we get full FFFFh at N7:2 to operate pump2 in full speed and the pump2 half the speed of pump1 to achieve 2:1 mixing ration.

- This multiplication is stored into N7:2 register. Digital to Analog conversion of value stored in N7:2 is performed inside the processor and equivalent mA current is received from terminal O:4 and O:6.
- Current to Voltage converter then converts current signals into voltage signal and adjusts motor speed.
- Agitator with output O:2/0 is activated as soon as the Master Start PB is pressed.

## Runtime Test Cases

```
Inputs          Outputs                    Physical Elements
I:3 = F258h      O:4 = O:6 = 0000h           Pump1 and Pump2 are off
I:3 = 792Ch      O:4 = 8000h, O:6 = 4000h   Pump1 = 4000RPM, Pump2 = 2000RPM
I:3 = 50C8h      O:4 = 5474h, O:6 = 2A3Ah   Pump1 = 2640RPM, Pump2 = 1320RPM
```

# PLC Program to Drain Same Products from 2 Tanks

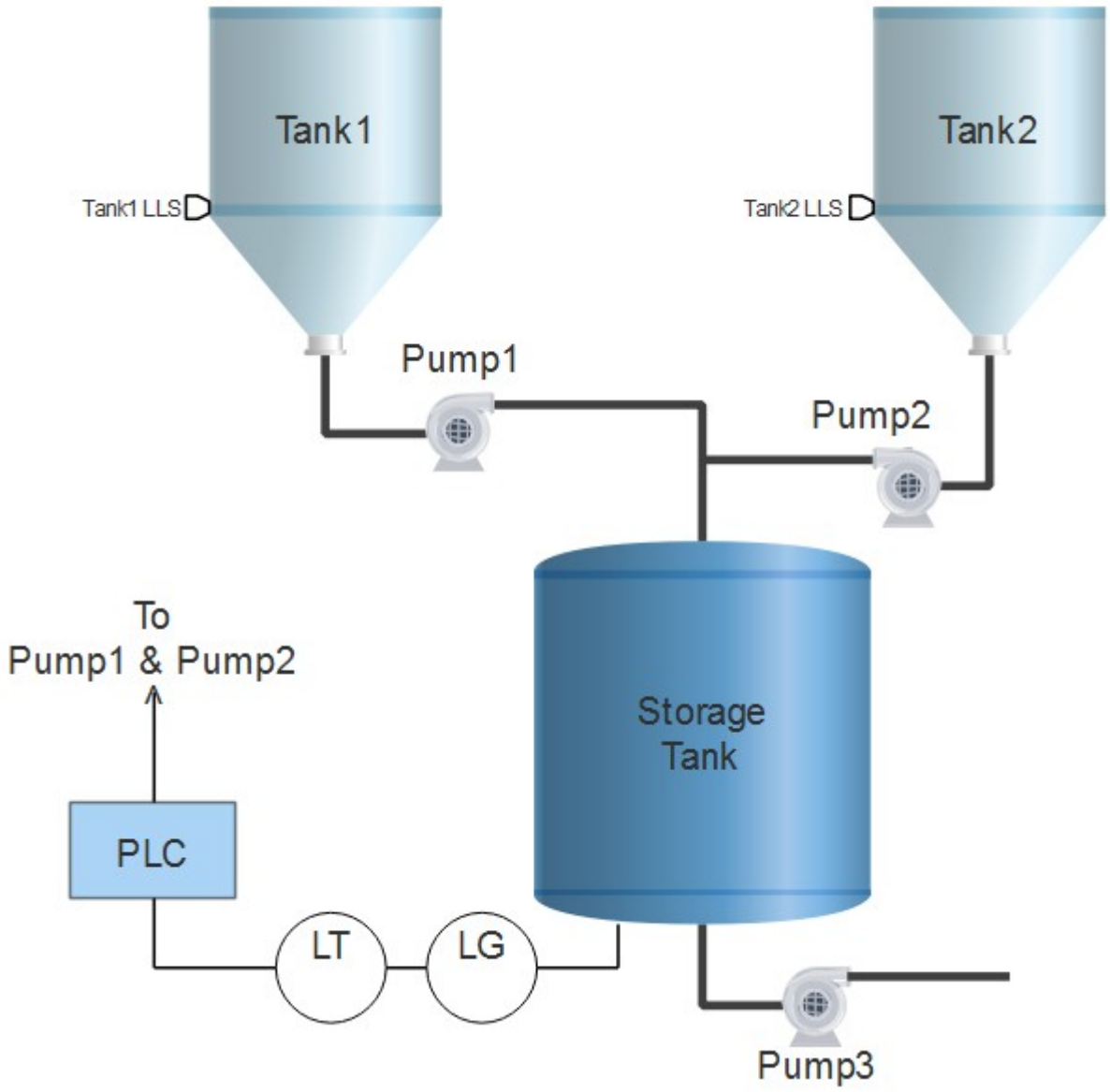
This is a PLC Program to Drain Same Products from 2 Tanks.

## Problem Description

Two tanks have same products filled. Draining from these depends on the requirement from the storage tank. Implement automation of this draining system in PLC using Ladder Diagram programming language

## Problem Diagram

**Diagram showing draining through pumps**




## Problem Solution

- Level gauge is used to measure level of the storage tank continuously
- Level gauge is connected with Level Transmitter which converts corresponding level output in 4-20mA equivalent.
- Analog I/O Modules are chosen to deal with Analog signals.
- Centrifugal pumps are used to drain material from both the tanks at the same time.
- Two low level switches are used to detect low level of tanks 1 and 2 which turns Pumps OFF when low level is reached.
- Height of storage tank is 5meters that is 500cm and the level which is to be maintained is 470cm.
- Calculate necessary conversions and use registers to store data and to do arithmetic operations.

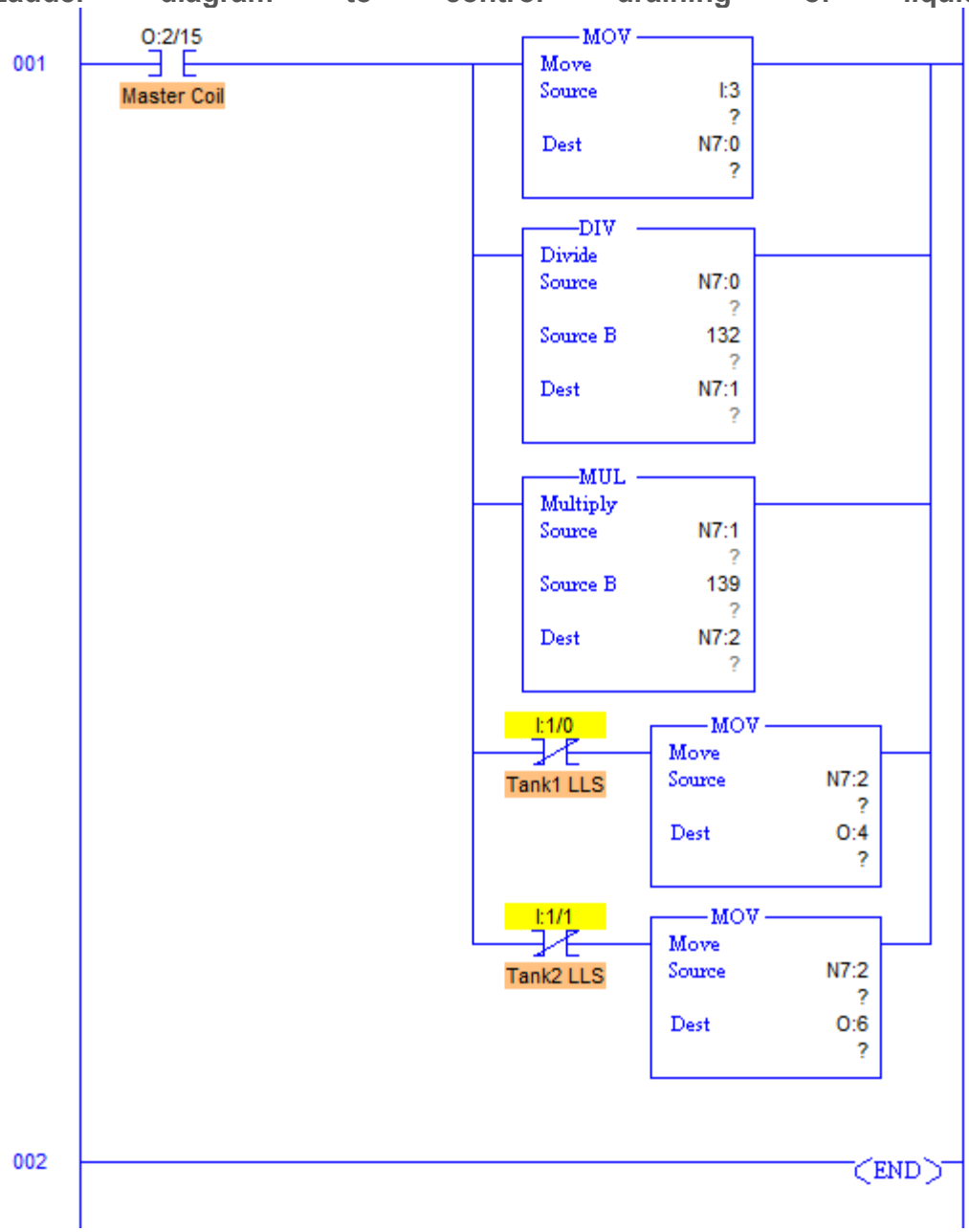
## PLC Program

Here is PLC program to Drain Same Products from 2 Tanks, along with program explanation and run time test cases.

```
List of Inputs and Outputs
```

```
I:1/14  = Start                                                 (Input)
I:1/15  = Stop                                                  (Input)
O:2/15  = Mater Coil                                            (Output)
I:1/0   = LLS of Tank1                                          (Input)
I:1/1   = LLS of Tank2                                          (Input)
O:4     = Output to I-V converter of Pump1                      (Output)
O:6     = Output to I-V converter of Pump2                      (Output)
I:3     = Input to which transmitter is connected         (Input)
N7:0    = Register to store input data                          (Register)
N7:1    = answer of division by value change per centimeter     (Register)
N7:2    = Multiplication answer                                 (Register)
```

**Ladder diagram to control draining of liquid from the tank**

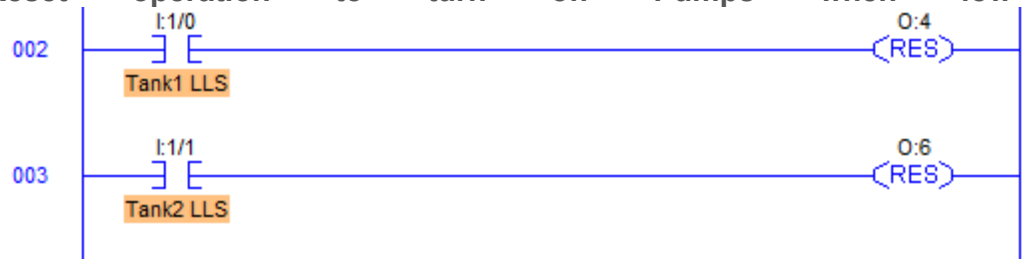



## Program Description

- RUNG001 comprises all the conversion needed to control pumps.
- Output of transmitter is in current signals which is 4-20mA.

- When output is 4mA, Analog Input Module converts it into 16bit equivalent hex numbers. Hence when input at I:3 to Analog module is 4mA, it moves 0000h into register and when 20mA, it moves FFFFh into register. Here register N7:0.
- Here height of the tank is 5m or 500cm. By converting it into equivalent hex, change in value per centimeter is 132.
- Value of N7:0 is then multiplied with 139 because when Level reaches 470cm, output is F0C0h. So when output at 470cm is multiplied with 139, we get full FFFFh at N7:2 to operate pumps in full speed.
- This multiplication is stored into N7:2 register. Digital to Analog conversion of value stored in N7:2 is performed inside the processor and equivalent mA current is received from terminal O:4 and O:6.
- Current to Voltage converter then converts current signals into voltage signal and adjusts motor speed.
- Two pumps are used here hence speed of response is very good and smooth operation is achieved.

**Reset operation to turn off Pumps when low level is detected**




- Both tanks have a Low Level switch at the bottom. When there is no material in the tank or very less material is present, pump should not run in order to prevent it from failure. Limit switch with address I:1/0 is for tank1 and I:1/1 is for tank2.

## Runtime Test Cases

```
Inputs          Outputs          Physical Elements
I:3 = F258h     O:4 = O:6 = 0000h        Pump1 and Pump2 are off
I:3 = 792Ch     O:4 = O:6 = 8000h        Pump1 and Pump2 = 4000RPM
I:3 = 50C8h     O:4 = O:6 = 5474h        Pump1 and Pump2 = 2640RPM
I:1/0 = 1       O:4 = 0000h      Pump1 off
I:1/1 = 1       O:6 = 0000h              Pump2 off
```

# PLC Program to Change Preset Value of Counter According to Various Products

This is a PLC Program to Change Preset Value of Counter According to Various Products.

## Problem Description

Different objects are moving on a conveyor belt. Let's say, circular balls, square blocks and rectangular blocks. All three types of objects are collected in the same sized box. As these types are different in size and shapes as well, number of objects to be placed are different for each type. Set counter value according to different sized objects' detection. Implement automation of this in PLC using Ladder Diagram programming language.

## Problem Diagram

**Diagram showing counting process of different objects**


## Problem Solution

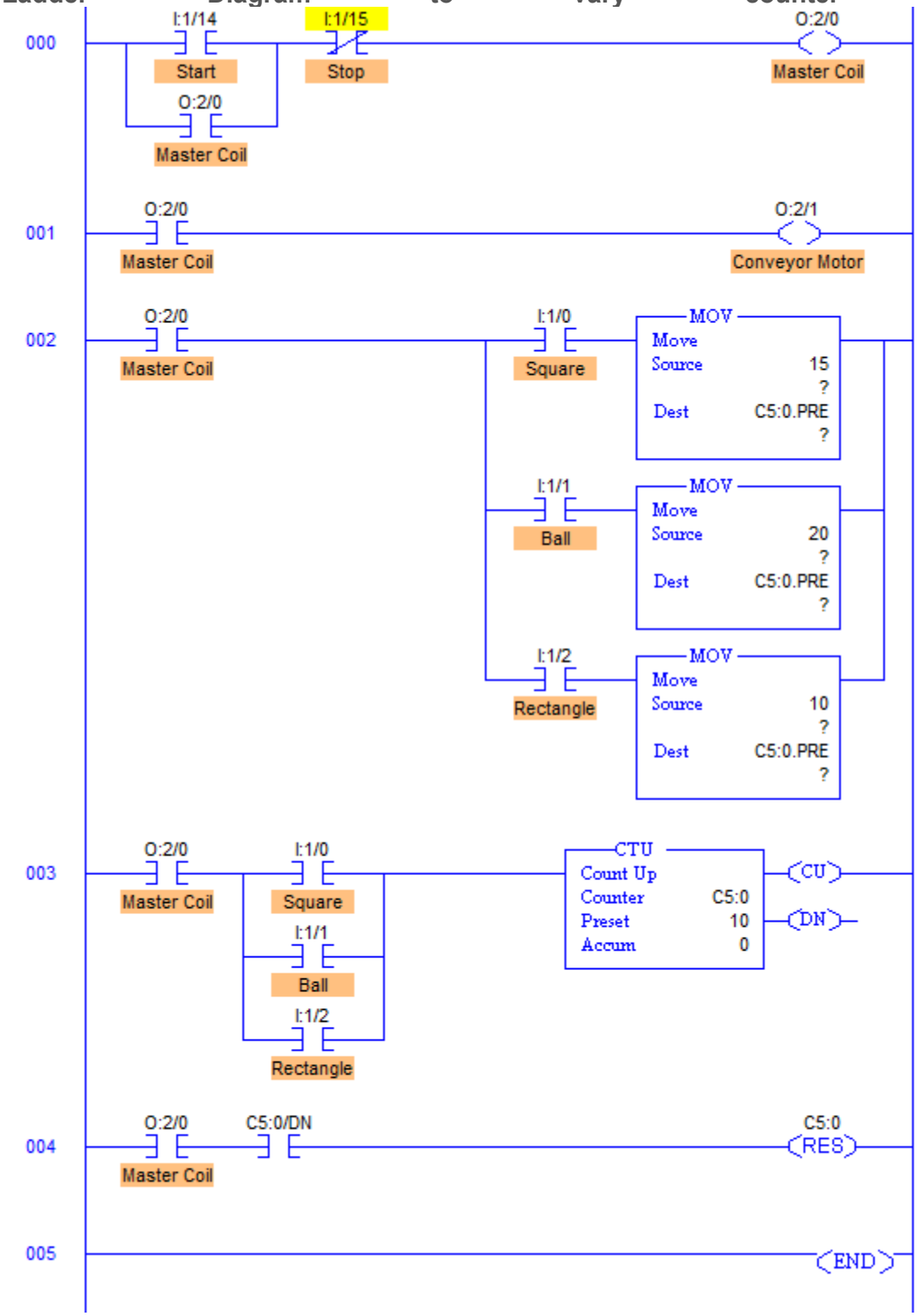
- Use three different proximity switches to detect all three different objects.
- Mount these switches such that switches detect assigned object only. For example, mount Square detector proximity such that it neither detects Rectangular blocks nor Balls.
- Load counter values in registers for different objects. And load this value as soon as a particular type of object is detected.

## PLC Program

Here is PLC program to Change Preset Value of Counter According to Various Products, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                                                  (Input)
 I:1/15  = Stop                                                   (Input)
 O:2/0   = Master Coil                                            (Output)
 O:2/1   = Conveyor motor                                  (Output)
 I:1/0   = Square block detector (Proximity)                      (Input)
 I:1/1   = Ball detector Proximity                                (Input)
 I:1/2   = Rectangle block detector Proximity                     (Input)
 MOV     = To move corresponding source value to counter preset   (Logical)
 CTU     = Up Counter to count the number of objects              (Counter)
-(RES)-  = To reset counter accumulator value                     (Reset)
```

**Ladder Diagram to vary counter preset value**




## Program Description

- RUNG001 is to run conveyor motor when the process starts.
- RUNG002 is to Move and Vary preset value of counter whenever a particular type of object is detected.
- When I:1/0 goes high, a square is detected, box has a capacity to pack 15 square blocks, hence #15 is moved to the preset value of counter with address C5:0.PRE. Similarly the same box has capacity to fill 20 balls or 10 rectangular blocks.

- When C5:0 is done counting and C5:0.PRE = C5:0.ACC, C5:0/DN goes high momentarily which resets counter immediately for next counting operation.
- C5:0/DN bit can be used to take further actions such as moving filled box and packing it.

## Runtime Test Cases

```
Inputs          Outputs             Physical elements
I:1/14 = 1      O:2/0 = O:2/1 = 1   Start conveyor motor
I:1/0 = 1       C5:0.PRE = 15        Collect 15 square objects
I:1/1 = 1       C5:0.PRE = 20        Collect 20 balls
I:1/2 = 1       C5:0.PRE = 10        Collect 10 rectangular blocks
```

# PLC Program to Reset all Non-Retentive Outputs

This is a PLC Program to Reset all Non-Retentive Outputs.

## Problem Description

Reset all Non-Retentive outputs of a process. Implement this by using MCR instruction in PLC using Ladder Diagram programming language.

## Problem Solution

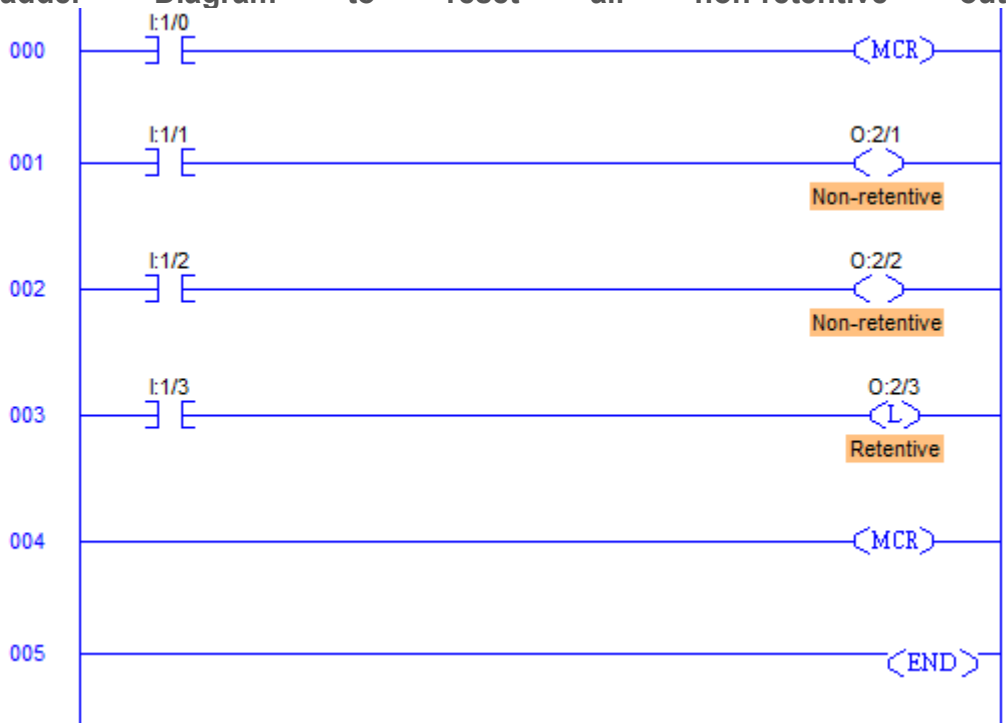
- By using Master Control Relay MCR instruction, we can solve this problem.
- When this instruction is enable, line is energized, it turns on.
- When MCR is OFF, the number of following ladder diagram lines specified are turned off.
- In many PLCs, all outputs are turned off while in many PLCs, only Non-retentive outputs are turned off.
- In PLCs such Allen Bradley and Siemens, all non-retentive outputs are turned off.
- The MCR must be turned on to be inactive. If the function goes off for some reason, it is active and turns the specified lines off.

## PLC Program

Here is PLC program to Reset all Non-Retentive Outputs, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0  = Activate MCR                    (Input)
 I:1/1  = Activate Non-retentive output1 (Input)
 I:1/2  = Activate Non-retentive output2 (Input)
 I:1/3  = Activate retentive output       (Input)
 O:2/1  = Non-retentive output1           (Output)
 O:2/2  = Non-retentive output2           (Output)
 O:2/3  = Retentive output                (Output)
```

**Ladder Diagram to reset all non-retentive outputs (Allen Bradley)**

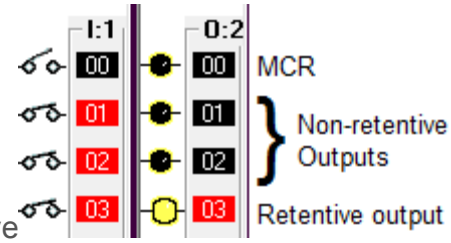



## Program Description

- As the theory of MCR describes, it needs to be turned on to be inactive.
- When I:1/0 is high, all the rungs perform operation normally which are in between RUNG000 to RUNG004. Others are independent of MCR.
- When I:1/0 goes low, it forces all the NON-RETENTIVE outputs O:2/1 and O:2/2 to be de-energized.
- However O:2/3 is not affected by MCR since it is –(L)– instruction.

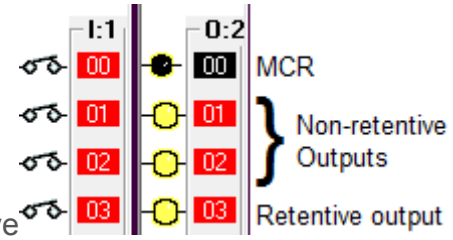
## Runtime Test Cases

MCR




MCR active

inactive




# PLC Program to Operate Light as an Emergency Signal

This is a PLC Program to Operate Light as an Emergency Signal.

## Problem Description

Boiler is being operated and monitored using a PLC. Temperature set point of boiler is 600°C and maximum temperature limit is 610°C. If temperature rises above its maximum limit. Hooter and emergency lights are activated in order to inform operator. Implement this in PLC using Ladder Diagram programming language.

## Problem Solution

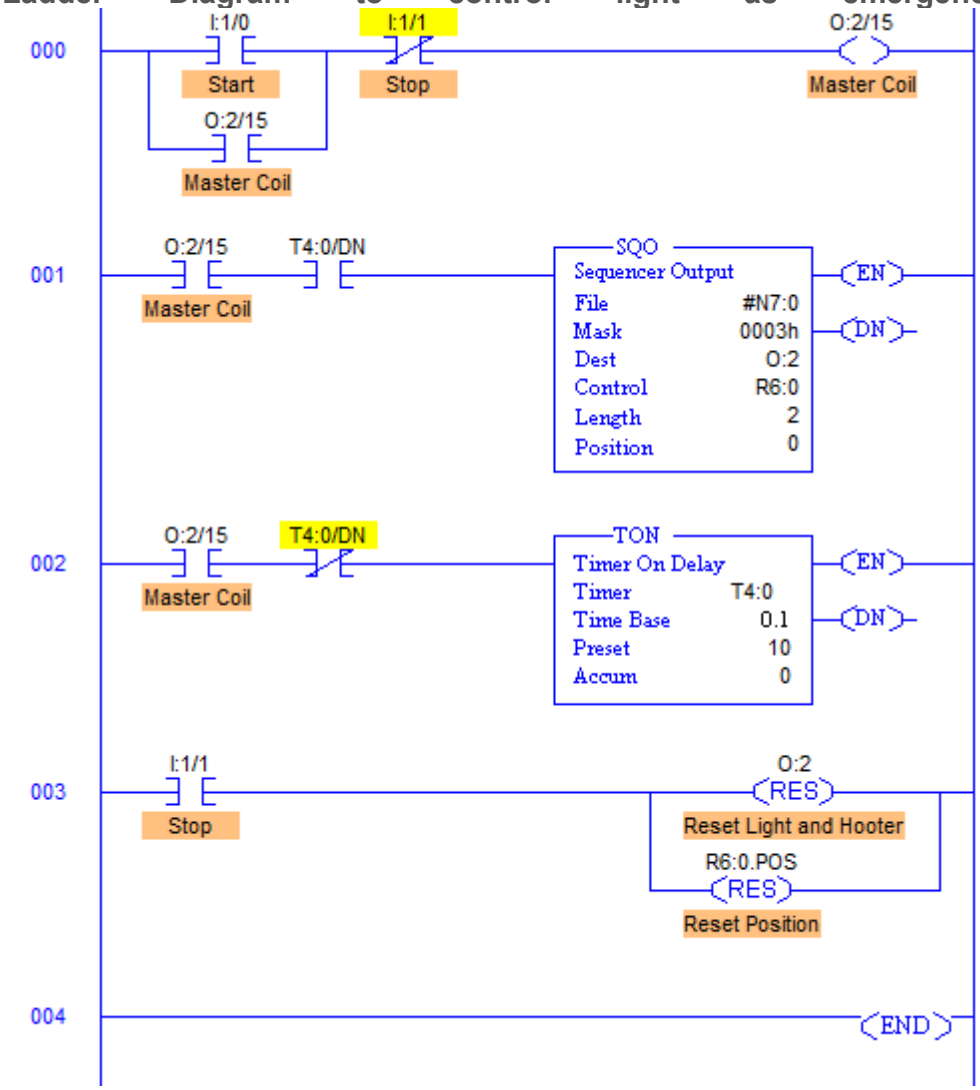
- There are two methods to solve this problem. One is by using stack operation and the other one is by using sequencer output method.
- Sequencer output method is best suited for this problem since very less configuration is needed and program length is also reduced.
- In this method, we need to assign SQO instruction by configuring all the parameters given in the instruction.
- Alarm signal must be continuous until operator turns it off manually.
- Sequencing of Lights and Hooter is done here by using SQO Sequential Output to operate lights and hooter.
- Set timer to activate and deactivate hooter and lights or to be précised, change sequence of output.
- Let's say, these both outputs are energized for 1sec and de-energized for next 1sec.
- In sequencer output, the start position is all zeros. So to start the actual function of output sequence, Position 1 is determined as starting sequence while programming.

## PLC Program

Here is PLC program to Operate Light as an Emergency Signal, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start                           (Input)
 I:1/1   = Stop                                    (Input)
 O:2/0   = Master Coil                             (Output)
 T4:0    = Timer to update output sequence (Timer)
 SQO     = Sequencer output                        (Sequencer)
 O:2/0   = Emergency Red Light (From O:2)  (Output)
 O:2/1   = Hooter       (From O:2) (Output)
 R6:0    = Control Register                        (Register)
 N7:0    = File register                           (Register)
-(RES)- = Reset outputs and Position of SQO        (Logical)
```

**Ladder Diagram to control light as emergency signal with hooter**




## Program Description

- File; #N7:0 and File length is 2, hence output sequence is varied from N7:0 and N7:1 with each input.
- Destination is set to O:2 hence with each transition, N7:0 and N7:1 are moved to O:2 with masking.
- O:2/0 and O:2/1 are used as the output address to Emergency Light and a Hooter, hence Mask has value 0003h which means data flow of N7:0/0, N7:0/1, N7:1/0 and N7:1/1 are passed to O:2/0 and O:2/1, and remaining bits are blocked.
- Control parameters are assigned to register R6:0.
- Sequence of Light and Hooter to be operated are stored in the registers N7:0 and N7:1 as following.

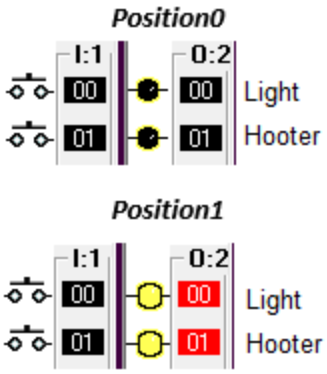
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N7:0/ | o | o | o | o | o | o | o | o | o | o | o | o | o | o | o | o |
| N7:1/ | o | o | o | o | o | o | o | o | o | o | o | o | o | o | 1 | 1 |

Hooter Light

- Time base is set to 1sec, hence after every 1sec, output sequence is changed to its next register pattern output which is then transferred to O:2 and O:2/0-O:2/1 are energized accordingly.
- As we can see, N7:1 and N7:2 have the exact opposite bit pattern. So, these bits are set to 1 for 1 cycle and reset for the next cycle. These bits are used to operate Hooter and a Light.
- So when I:1/0 is pressed, during position 0, outputs and remain in OFF condition and are energized when SQO is at position 1.

- When Stop PB with address I:1/1 is pressed, Position is reset to 0 and all the outputs are de-energized.

## Runtime Test Cases



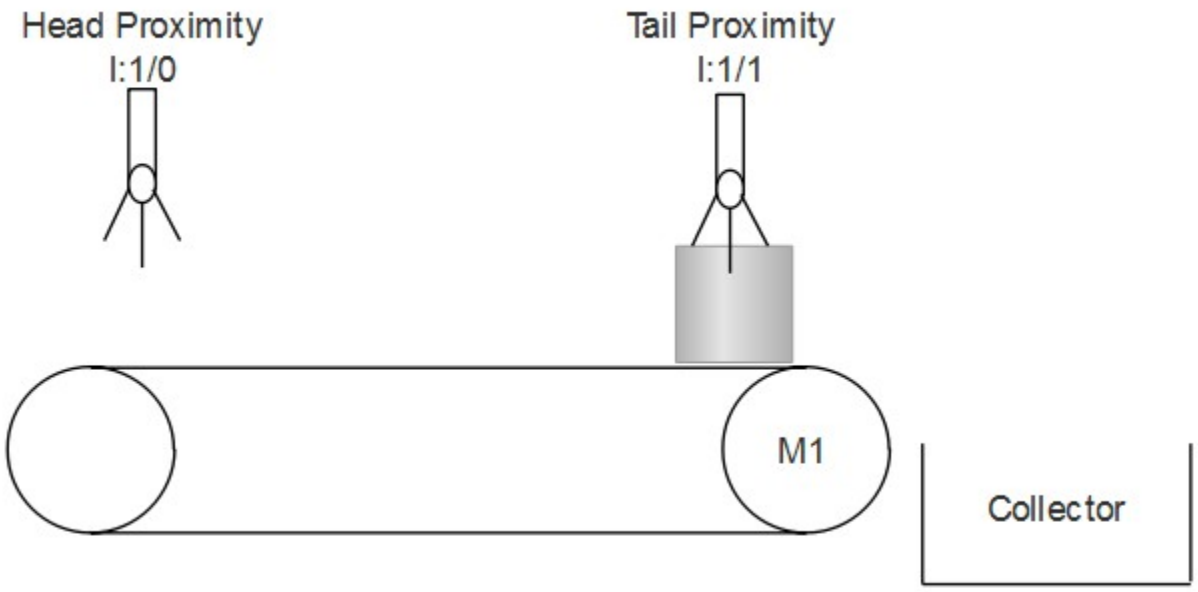

# PLC Program to Measure Time Taken by an Event

This is a PLC Program to Measure Time Taken by an Event.

## Problem Description

There are certain objects moving on conveyor belt. Time of an object to reach from one end to another end of the conveyor is to be measured. Implement this in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- Install two proximity switches, one at the head end and another at the tail end.

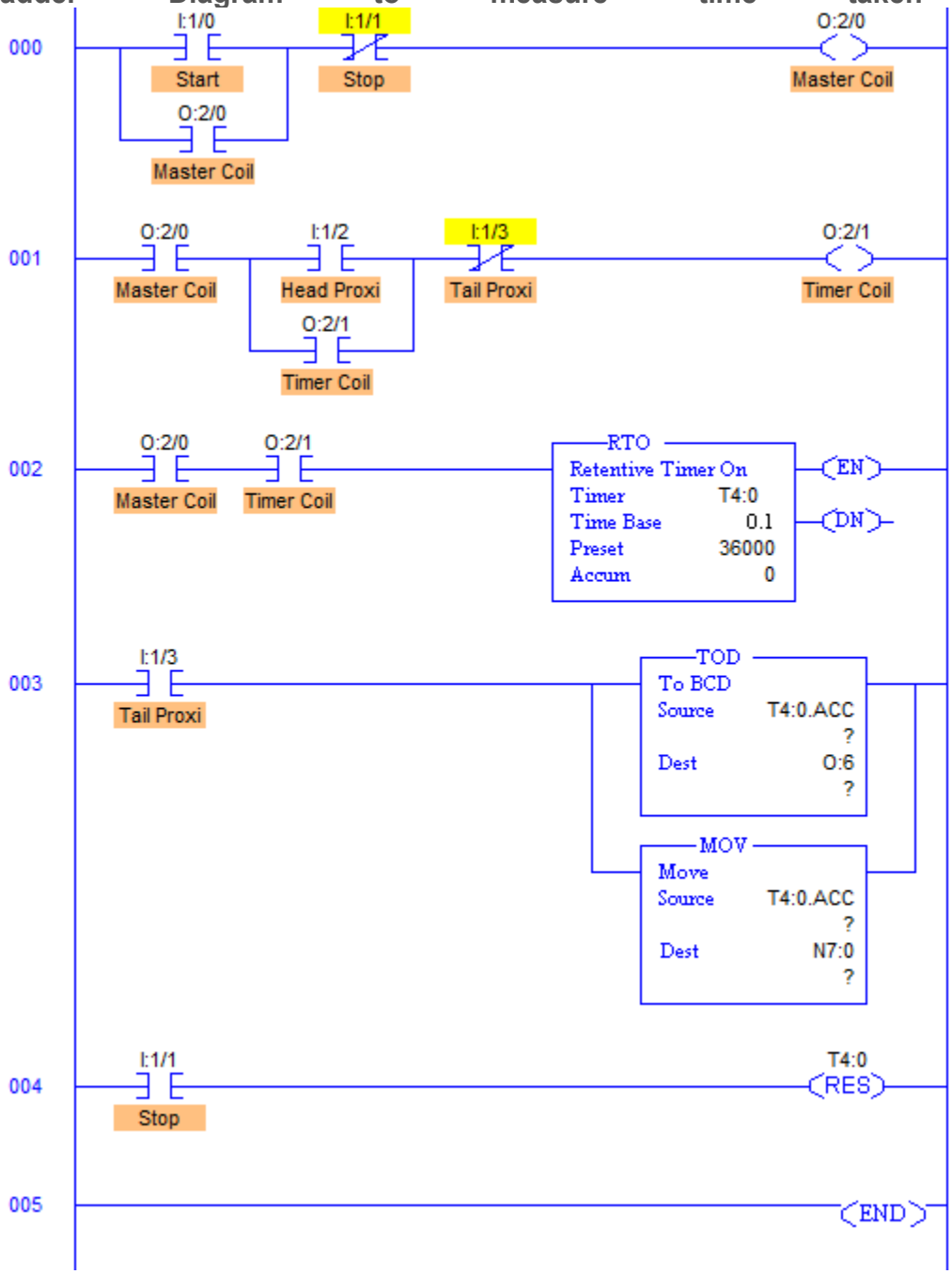
- When first proximity detects an object, it latches and output starting timer.
- When another proximity placed at the tail end detects the same object, timer is stopped.
- Move preset value to any register or output displays.
- This shows time taken by an event.
- Similar phenomena can be applied to measure time taken to fill an empty tank.

## PLC Program

Here is PLC program to Measure Time Taken by an Event, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start Measurement                      (Input)
 I:1/1   = Stop Measurement                       (Input)
 I:1/2   = Head proximity                (Input)
 I:1/3   = Tail Proximity                (Input)
 T4:0    = Retentive Timer to measure time(Timer)
 N7:0    = Register to store measured time(Register)
 O:6     = (Display) To display measured time     (Output)
-(RES)-  = Reset Retentive timer                  (Reset)
```

**Ladder Diagram to measure time taken by an event**




## Program Description

- RUNG001 & RUNG002 operate Timer T4:0 through Timer Coil O:2/1. When Head Proximity Switch detects an object, it latches timer coil activating Timer T4:0.
- RUNG003 operates moving and converting of timer accumulator value. When Tail Proximity detects an object, this de-energizes Timer Coil stopping timer T4:0. And as soon as timer stops when I:1/3, it activates RUNG003 and TOD and MOV instruction are executed.
- MOV instruction moves T4:0.ACC value intro register N7:0 and TOD instruction converts Decimal data into equivalent BCD code to operate Display with address O:6. Digits displayed on the display is time taken by an event.
- Retentive timer is used here so that when Tail Proximity detects the object and Master Coil is de-energized, value in the accumulator do not change.

## Runtime Test Cases

```
Inputs          Outputs  Physical Elements
I:1/2 = 1        O:2/1 = 1        Enable Timer
I:1/3 = 1        O:2/1 = 0        Disable Timer, Convert into BCD, Move data
I:1/1 = 1       -(RES)- = 1       Stop Measurement, Reset Timer
```

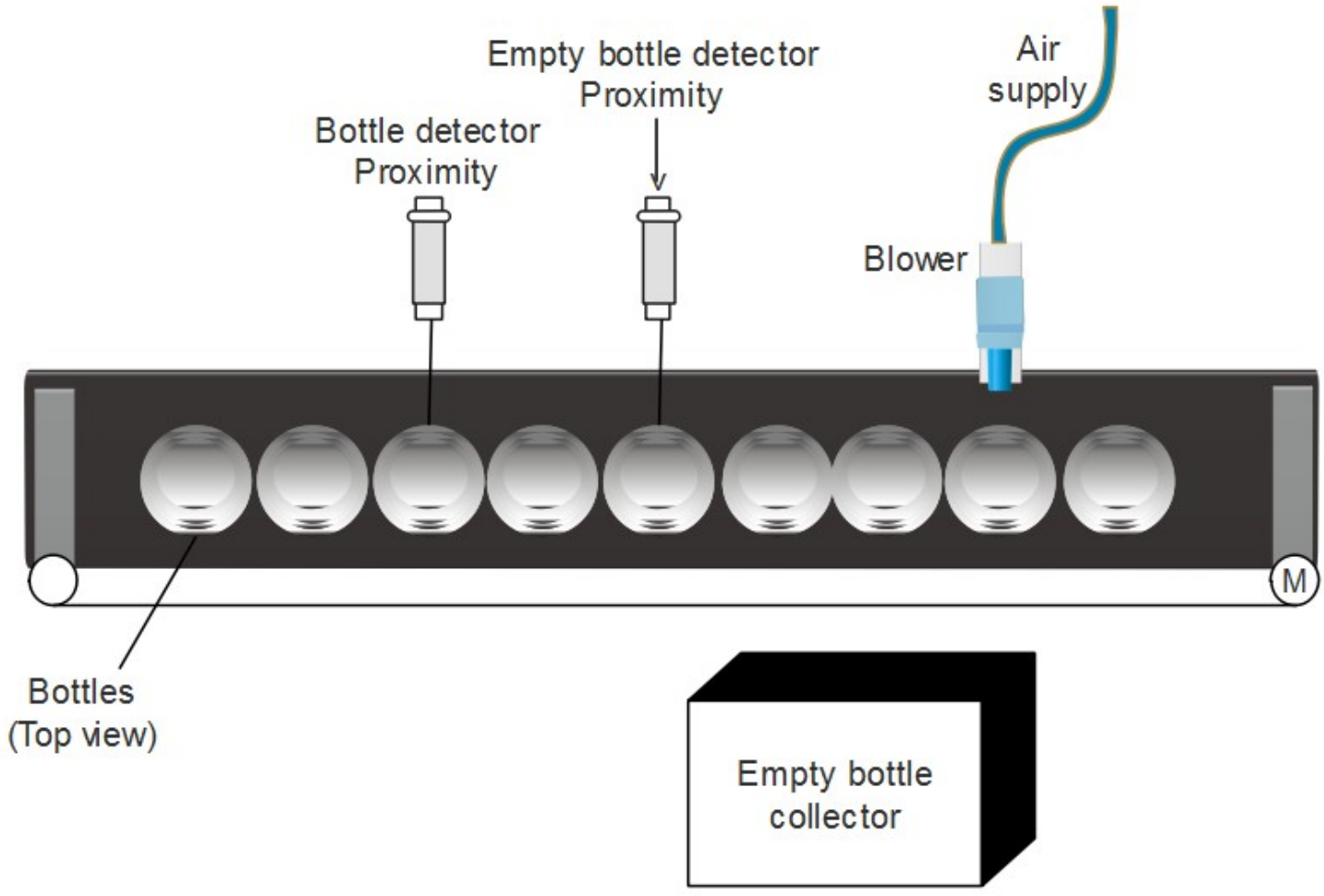
# PLC Program to Remove Empty Detected Bottle on Conveyor

This is a PLC Program to Remove Empty Detected Bottle on Conveyor.

## Problem Description

After filling process, bottles are moved on the conveyor belt for packing process. Detect if any empty bottle is left on the conveyor and remove it from the conveyor. Implement automation of this in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

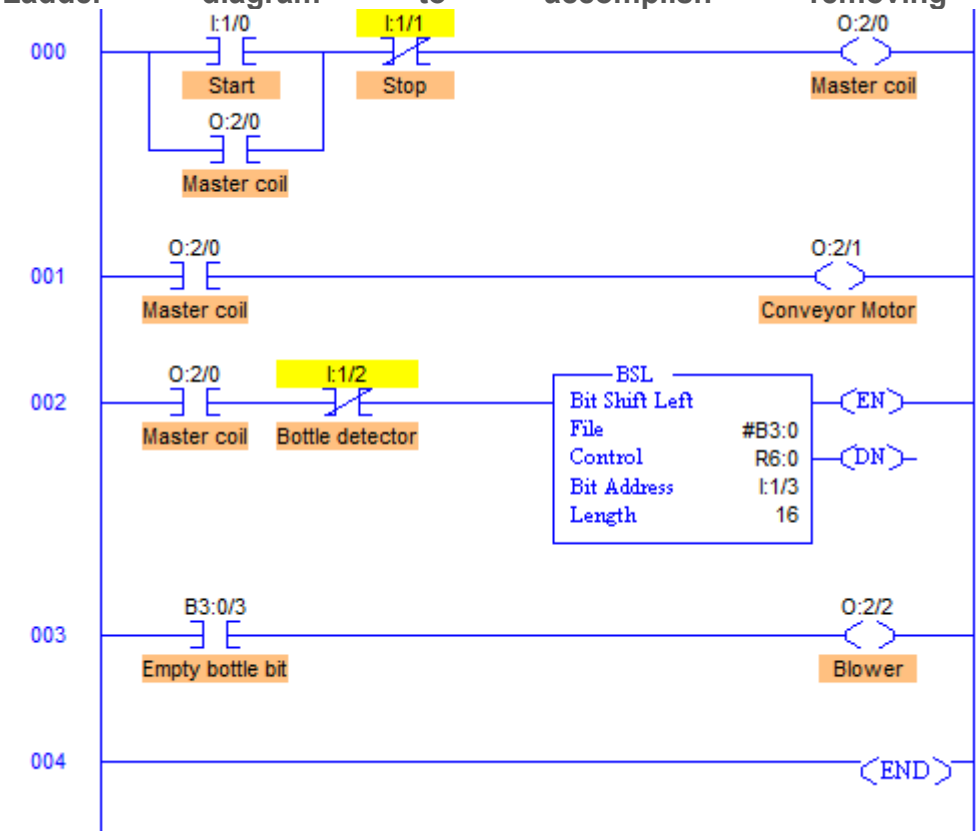
- Proximity sensors are used to detect bottles.
- One proximity is calibrated such that it detects all the bottles passing on the conveyor. And other proximity is used such that it detects only empty bottle.
- Use Bit Shift Register to shift a bit which is set when an empty bottle is detected.
- Use a piston or blower is used to throw an empty bottle out of the conveyor.

## PLC Program

Here is PLC program to Remove Empty Detected Bottle on Conveyor, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start                                  (Input)
 I:1/1   = Stop                                   (Input)
 I:1/2   = Bottle Proximity                       (Input)
 I:1/3   = Empty bottle proximity         (Input)
 O:2/0   = Master coil / Run                      (Output)
 O:2/1   = Conveyor motor                 (Output)
 O:2/2   = Blower                         (Output)
 BSL     = Bit shift left instruction             (Logical)
 B3:0    = Bit shift Register                     (Register)
 B3:0/3  = Bit to energize capping machine (Bit)
 R6:0    = Control register                       (Register)
```

**Ladder diagram to accomplish removing of empty bottle**




## Program Description

- When the system is started, conveyor motor coil with address O:2/1 is energized.
- RUNG002 and RUNG003 are used to operate bit shift register and Blower with address O:2/2.
- Whenever conveyor motor is in RUN mode, empty bottles detected by the proximity sensor with input I:1/3, it sets B3:0/0 bit and is shifted left every time a bottle is detected by bottle proximity with address I:1/2.

- From proximity to blower, distance is 4 steps. Hence bit B3:0/3 of B3:0 register is used to operate blower.
- When B3:3/0 bit is set that is when empty bottle is detected by input I:1/3, after 4 steps, blower is activated and the empty bottle is removed.

## Runtime Test Cases

```
Inputs                  Output    Physical Elements
I:1/0 = 1 (Start PB)   O:2/1 = 1  Run conveyor motor
I:1/3 = 1               B3:0/0 = 1        Set first bit of bit register
I:1/2 = 1               BSL = 1   Shift bit to left
B3:0/3 = 1              O:2/2 = 1 Activate blower
```

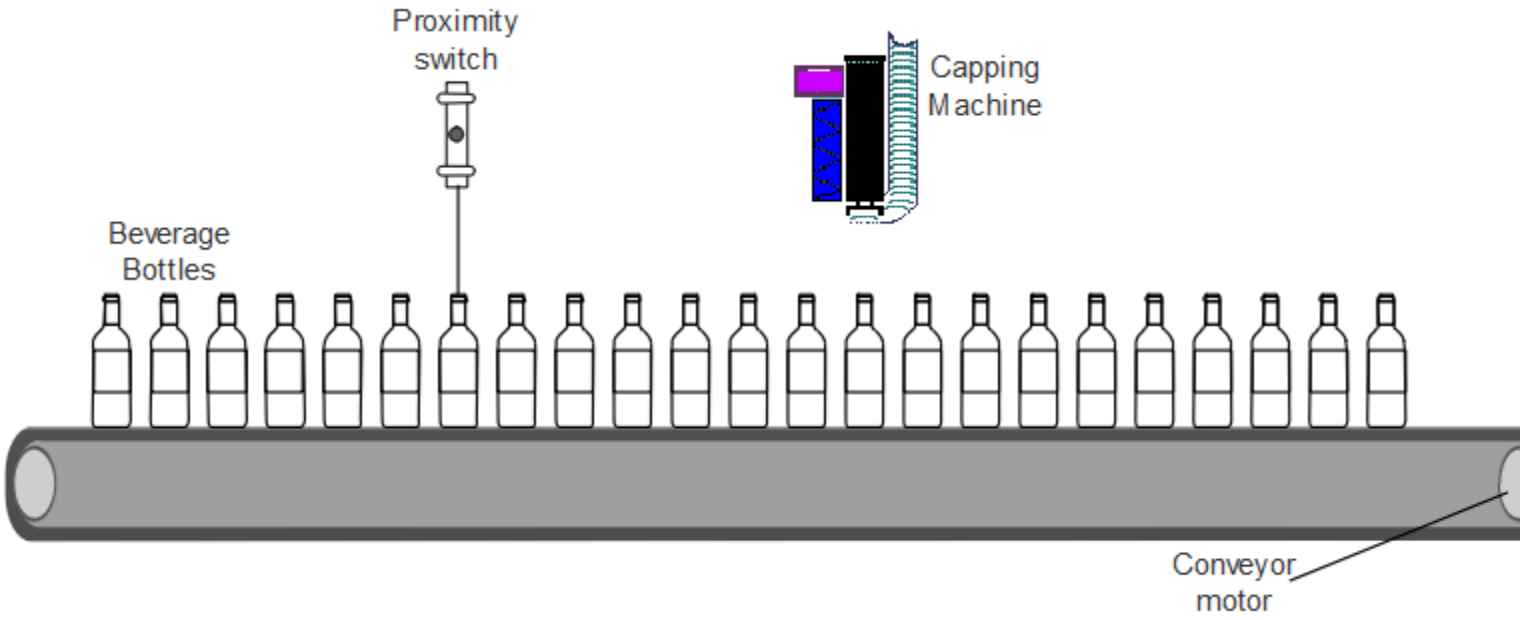
# PLC Program to Perform Capping of Beverage Bottles

This is a PLC Program to Perform Capping of Beverage Bottles.

## Problem Description

Beverage bottles are moving on a conveyor belt. Capping of these bottles is performed. Crown cork caps are used for these bottles. Implement automation of this process in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- To sense the bottle, proximity sensor is used.
- Timer is used to stop the conveyor for 1sec for capping procedure.
- Bit Shift register is also used to perform this operation.

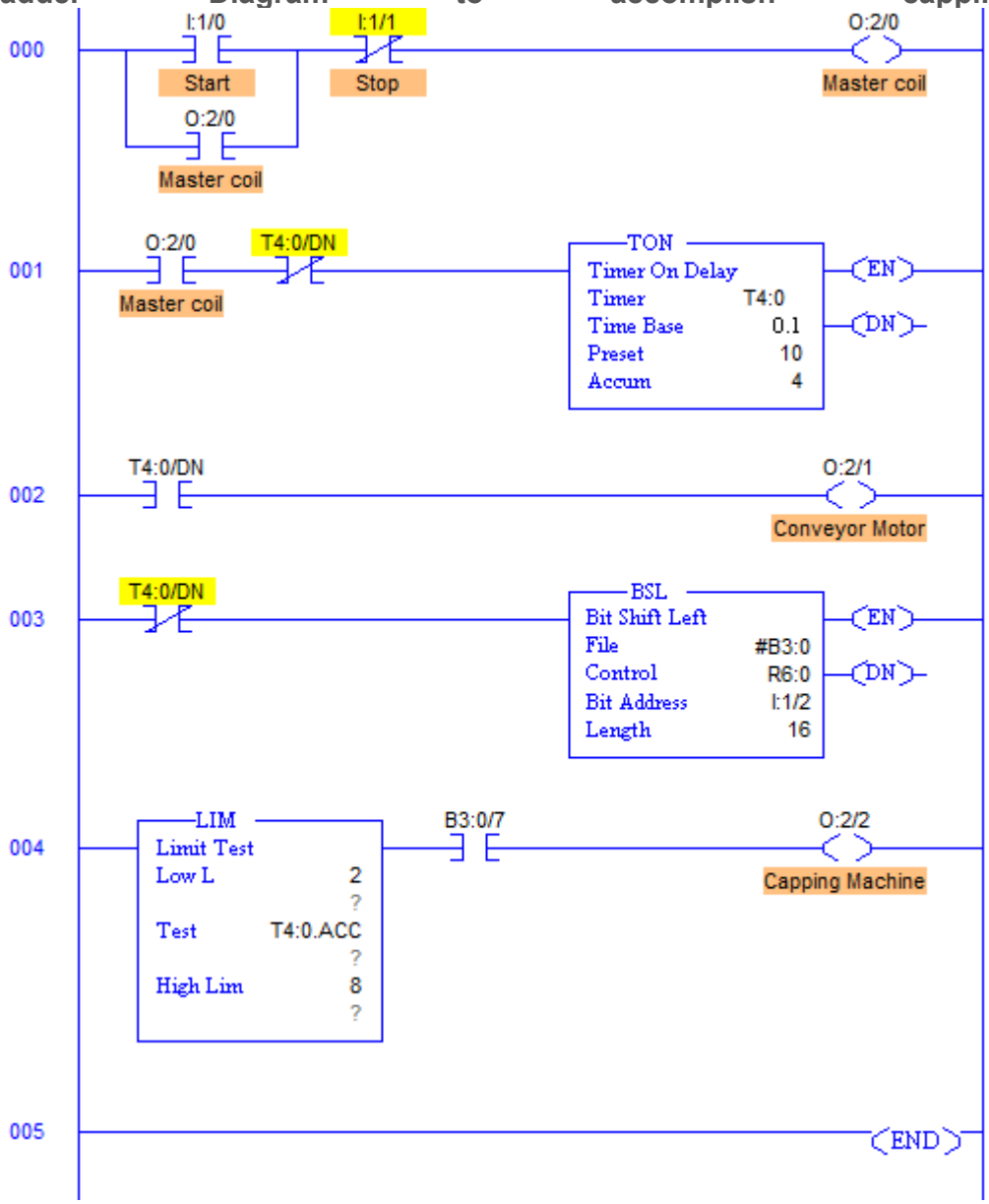
- Count the number of steps capping machine is placed from the sensor and set bit position to operate capping machine accordingly.
- In this example as you can see, bottle is 8 steps away from the proximity switch, so if Bit register B3:0 is used, then capping machine should be operated when B3:0/0 is shifted to B3:0/7.
- Capping machine is operated by pneumatic system. It is activated by air supply when energized.
- Similarly we can even make an arrangement where two capping machines are operated, one to cap small bottles and other for larger bottles.

## PLC Program

Here is PLC program to Perform Capping of Beverage Bottles, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start                                                    (Input)
 I:1/1   = Stop                                                     (Input)
 I:1/2   = Proximity                                                (Input)
 O:2/0   = Master coil / Run                                        (Output)
 O:2/1   = Conveyor motor                                  (Output)
 O:2/2   = Capping machine                                          (Output)
 BSL     = Bit shift left instruction                               (Logical)
 B3:0    = Bit shift Register                                       (Register)
 B3:0/7  = Bit to energize capping machine                 (Bit)
 R6:0    = Control register                                         (Register)
 T4:0    = Timer to stop conveyor for capping                       (Timer)
 LIM     = Limit output to perform capping before conveyor starts again     (Compare)
```

**Ladder Diagram to accomplish capping of bottles**




## Program Description

- RUNG001 & RUNG002 are used to operate conveyor motor.
- Assuming it takes 0.8secs to cap a bottle, so capping is performed conveyor is stopped for a sec.
- When the system is started, conveyor motor with address O:2/1 runs for a moment and stops for a sec. Timer is in auto reset mode by giving XIO of T4:0/DN in series to it.
- RUNG003 and RUNG004 are used to operate bit shift register and Capping Machine with address O:2/2.
- Whenever conveyor motor is in RUN mode and bottles is detected by the proximity sensor with input I:1/2, it sets B3:0/0 bit and is shifted left when conveyor motor runs for a moment.
- From proximity to capping machine, distance is 8 steps. Hence bit B3:0/7 of B3:0 register operates capping machine.
- When Accumulator value is between 2 to 8 that is when conveyor is stopped for a sec and timer is between 0.2-0.8secs, capping machine is operated.

## Runtime Test Cases

| Inputs | Output | Physical Elements |
|---|---|---|

```
I:1/2 = 1 & T4:0/DN = 0   B3:0/0 = 1      Set first bit of bit register
T4:0/DN = 1               O:2/1 = 1       Run conveyor motor
T4:0/DN = 0               O:2/1 = 0       Stop conveyor motor, shift bit left
T4:0.ACC = 2<Accum<8      O:2/2 = 1       Perform capping
```

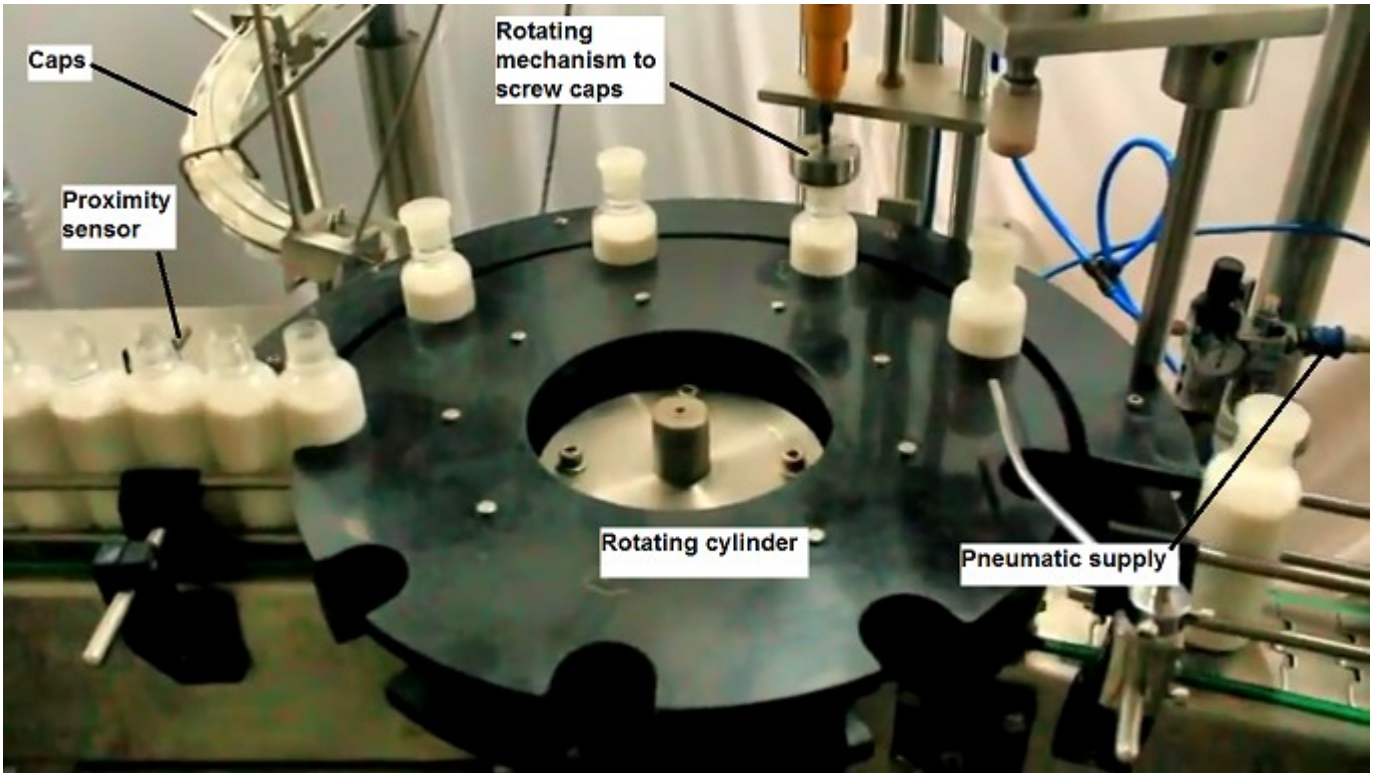
# PLC Program to Perform Bottle's Capping with Rotating Mechanism

This is a PLC Program to Perform Bottle's Capping with Rotating Mechanism.

## Problem Description

Water bottles are moved on a conveyor for capping. Screw caps are screwed to close the opening end of the bottle using rotating mechanism. Implement this process in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- To sense the bottle, proximity sensor is used.
- Used a timer to stop the cylinder motor for 2secs to screw caps.
- Used one more timer to run the motor for 1sec to rotate the cylinder.
- Bit Shift register is also used to perform this operation.
- Count the number of steps capping machine is placed from the sensor and set bit position to operate capping machine accordingly.
- In this example as you can see, bottle is 7 steps away from the proximity switch, so if Bit register B3:0 is used, then capping machine should be operated when B3:0/0 is shifted to B3:0/6.

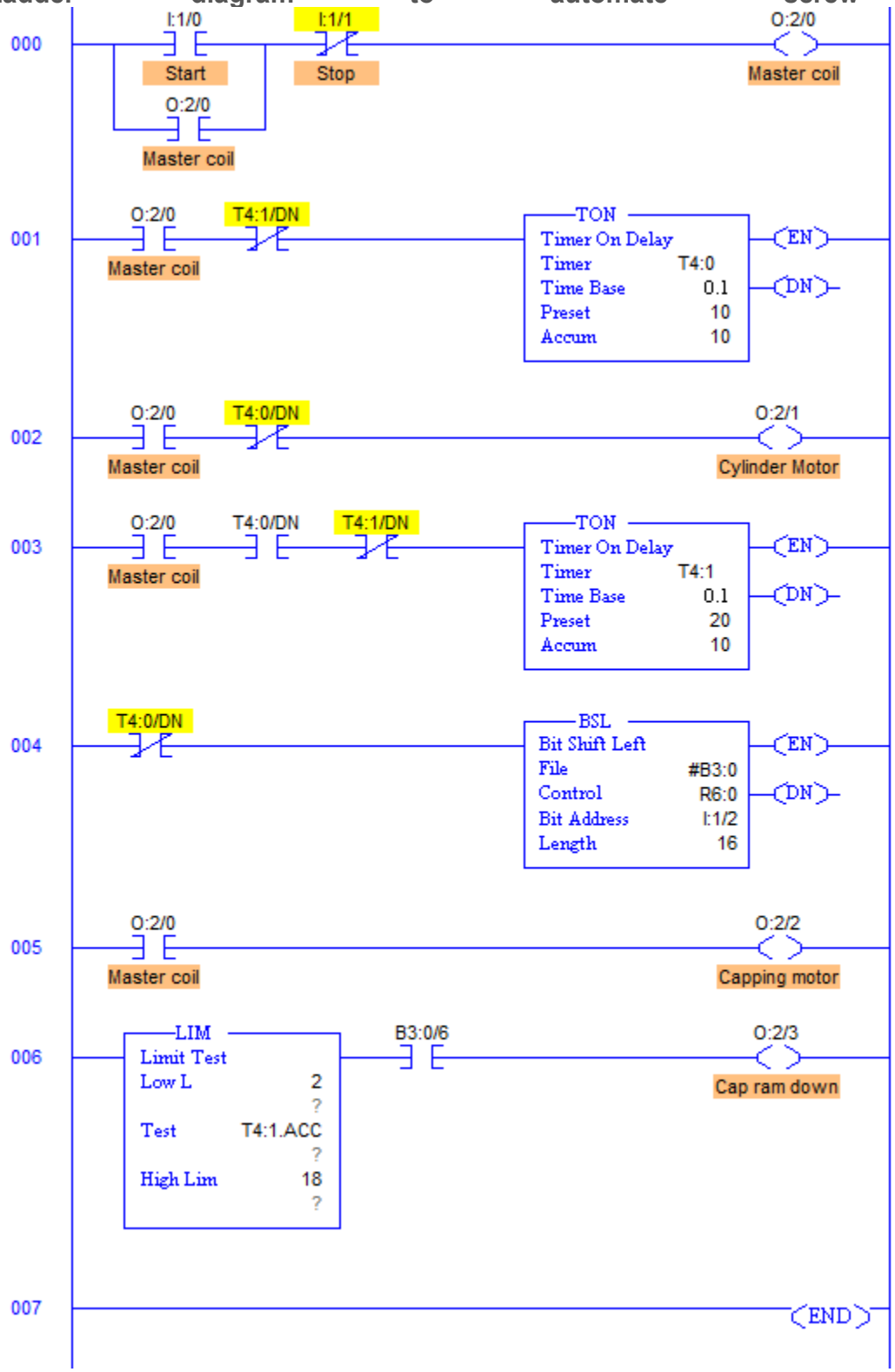
- Two inputs are given to this Capping machine, electric supply to run motor and pneumatic supply to push machine down cap ram.

## PLC Program

Here is PLC program to Perform Bottle's Capping with Rotating Mechanism, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start                                          (Input)
 I:1/1   = Stop                                           (Input)
 I:1.2   = Proximity                                      (Input)
 O:2/0   = Master coil / Run                              (Output)
 O:2/1   = Cylinder motor                          (Output)
 O:2/2   = Electric supply to capping machine motor       (Output)
 O:2/3   = Pneumatic supply to cap ram                    (Output)
 BSL     = Bit shift left instruction                     (Logical)
 B3:0    = Bit shift Register                             (Register)
 B3:0/6  = Bit to energize capping machine         (Bit)
 R6:0    = Control register                               (Register)
 T4:0    = Timer to run cylinder motor                    (Timer)
 T4:1    = Timer to stop cylinder motor timer for capping (Timer)
 LIM     = Limit output to perform capping               (Compare)
```

**Ladder diagram to automate screw cap process**




## Program Description

- RUNG001, RUNG002 and RUNG003 are used to operate cylinder motor.
- Assuming it takes 1.6secs to screw cap a bottle, conveyor is stopped for 2secs and capping is done.
- When the system is started, cylinder driving motor with address O:2/1 runs for one 1sec and stops for 2secs. Timer is set to auto reset mode by giving XIO of T4:1/DN in series to it. T4:0 is used to de-energize cylinder driving motor coil with address O:2/1 after 1sec and T4:1 is used to energize the same coil after 2secs.

- RUNG004, RUNG005 and RUNG006 are used to operate bit shift register BSL and Capping Machine with address O:2/2.
- Whenever proximity sensor with input I:1/2 detects bottle and O:2/1 is energized, I:1/2 sets B3:0/0 bit. This bit is shifted when again this coil is energized.
- From proximity to capping machine, distance is 7 steps. Hence bit B3:0/6 of B3:0 register is used to operate capping machine.
- Capping machine motor is connected with output O:2/2 which is energized till the process is running. Hence, simply letting master coil run the capping machine motor.
- When Accumulator value is between 2 to 18, pneumatic supply is activated and screw capping is operated that is when cylinder motor is stopped for 2secs and timer is between 0.2-1.8secs.

## Runtime Test Cases

```
Inputs                   Output        Physical Elements
I:1/2 = 1                B3:0/0 = 1   Set first bit of bit register
O:2/0 = 1                O:2/1 = 1    Start cylinder motor
T4:0/DN = 1      O:2/1 = 1    Stop cylinder motor after 2secs
T4:1/DN = 1              O:2/1 = 1    Start cylinder motor again
T4:0/DN = 0              BSL/EN = 1   Shift bit to left
O:2/0 = 1                O:2/2 = 1    Start capping machine motor
T4:0.ACC = 2<Accum<16    O:2/3 = 1    Activate pneumatic supply to
                                      push down cap ram for capping
```

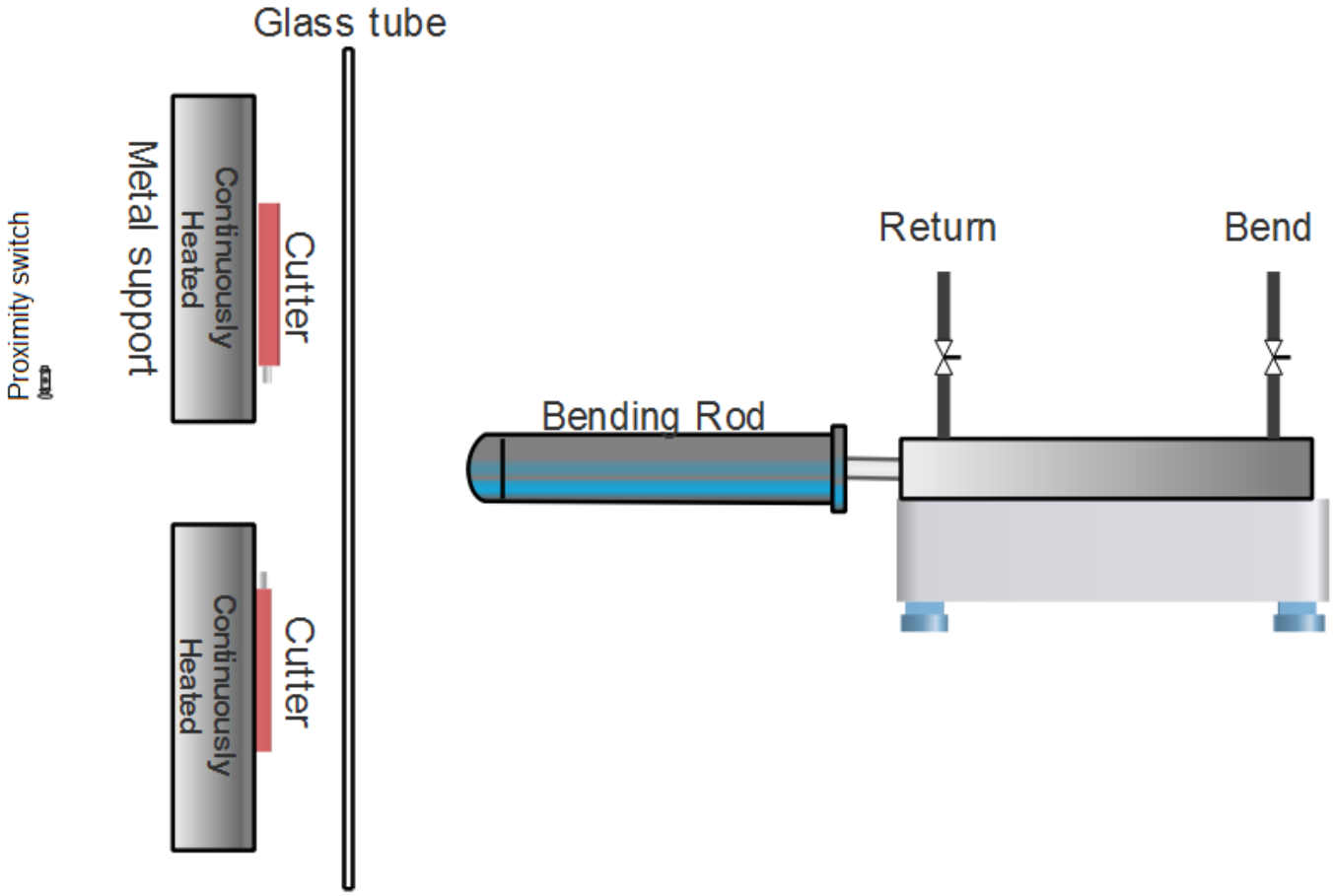
# PLC Program to Heat and Bend Glass Tubes

This is a PLC Program to Heat and Bend Glass Tubes.

## Problem Description

Heated glass tubes are passing in a process line having a particular length which are to be bent. To manufacture fluorescent bulbs, these tubes are to be bent in U-Shape. Automate this process in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- Diagram is an example of how bending process is done.
- Pneumatic operated rod moves in right and left direction which forces glass tube to bend through metal support.
- Metal support is continuously heated so that glass tubes do not break while bending.
- When bending is done, two cutters are used to cut the bent glass tube.
- Proximity switch detects a length to be cut by the cutter.
- Metal support are mounted close to each other such that only bending rod and glass tube together can pass through it.
- Metal support is not exactly square as shown in the diagram, it has curvy ends for smoother bending operation.

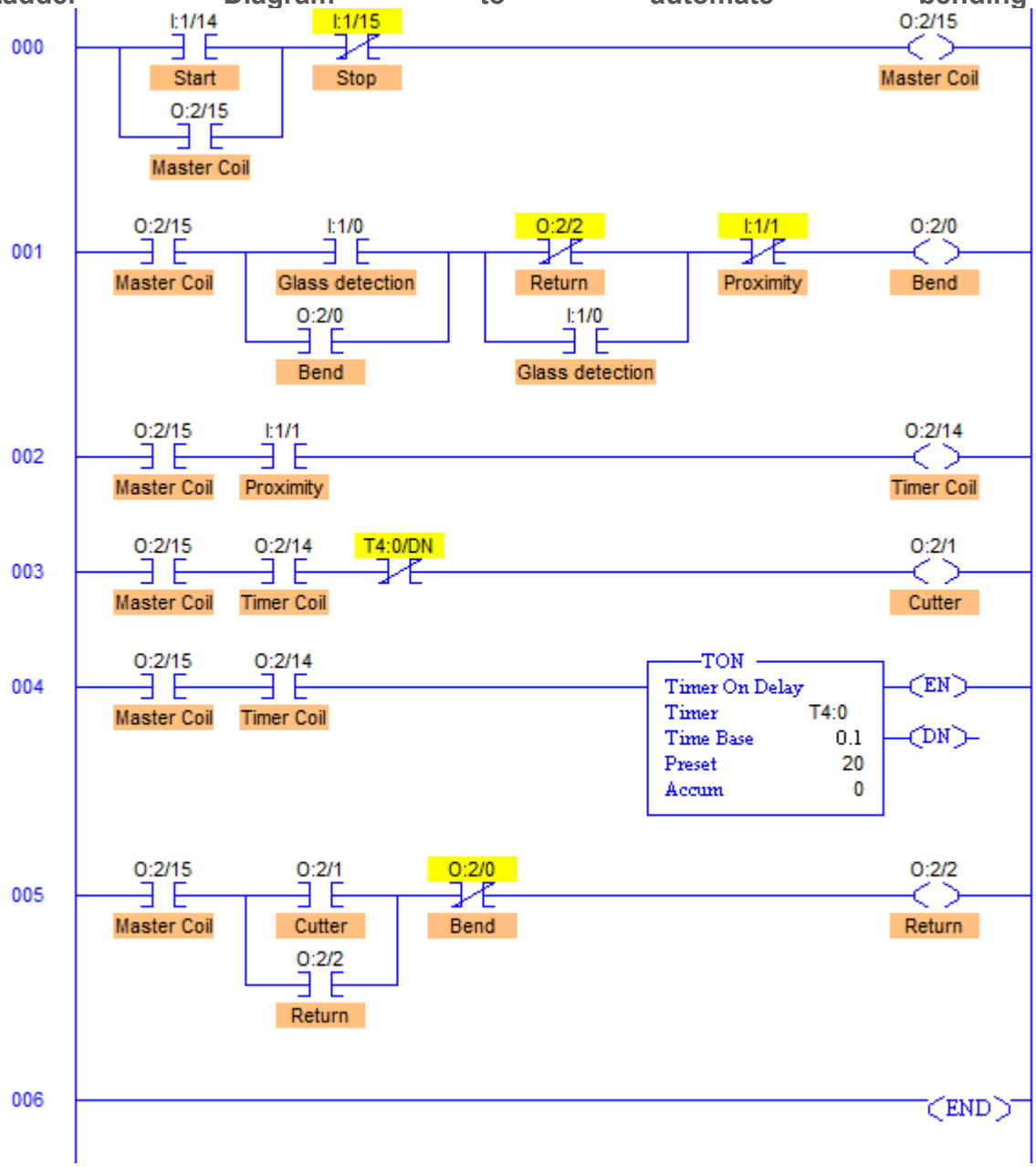
## PLC Program

Here is PLC program to Heat and Bend Glass Tubes, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                                        (Input)
 I:1/15  = Stop                                         (Input)
 O:2/15  = master Coil                                  (Output)
 I:1/0   = Glass presence detector                      (Input)
 I:1/1   = Glass length detector (Proximity)            (Input)
 O:2/0   = Bend (Air supply)                            (Output)
 O:2/1   = Cutter                                (Output)
 O:2/2   = Return (Air supply)                          (Output)
```

```
O:2/3   = Current supply (To heat metal support)  (Output)
O:2/14  = Timer coil                              (Output)
T4:0    = Timer for cutter                        (Timer)
```

**Ladder Diagram to automate bending operation**




## Program Description

- RUNG001 is to supply air to move rod toward left when Glass Tube is detected by I:1/0. This air supply is provided through air regulator which is controlled by contactor or an SCR firing circuit which is fired when Input I:1/0 goes true.
- Air supply is provided till fixed length of bent glass is detected by Proximity I:1/1.
- RUNG002, RUNG003 and RUNG004 are to operate cutter. Cutter is activated when I:1/1 goes high that is when desired length is detected. Timer is used to stop cutter after 2secs (assuming cutter takes approx. 2secs to cut the bent tube).
- When timer is over, T4:/0DN goes high turning cutter off and activating Return O:2/2. This is again operated by air supply through air regulator. When O:2/2 is activated, air supply from Bend is diverted to Return to take the rod back to its normal position.

- In case of malfunction, when both Bend and Return are energized together, it does not let rod move in any direction. To avoid this, interlocking is provided to Bend and Return operating rungs so that air supply is fed to perform any one operation, either Bend or Return.

## Runtime Test Cases

```
Inputs       Outputs               Physical Elements
I:1/0 = 1    O:2/0 = 1              Air Supply to Bend
I:1/1 = 1    O:2/0 = 0, O:2/1 = 1  Withdraw Air Supply from Bend, Activate Cutter
T4:0/DN = 1  O:2/1 = 0              Deactivate Cutter
T4:0/DN = 1  O:2/2 = 1              Air Supply Return
I:1/0 = 1    O:2/2 = 0              Withdraw Air Supply from Return
```

# PLC Program to Measure Unknown Frequency

This is a PLC Program to Measure Unknown Frequency.

## Problem Description

Measure and display unknown frequency. Implement this in PLC using Ladder Diagram programming language.

## Problem Solution

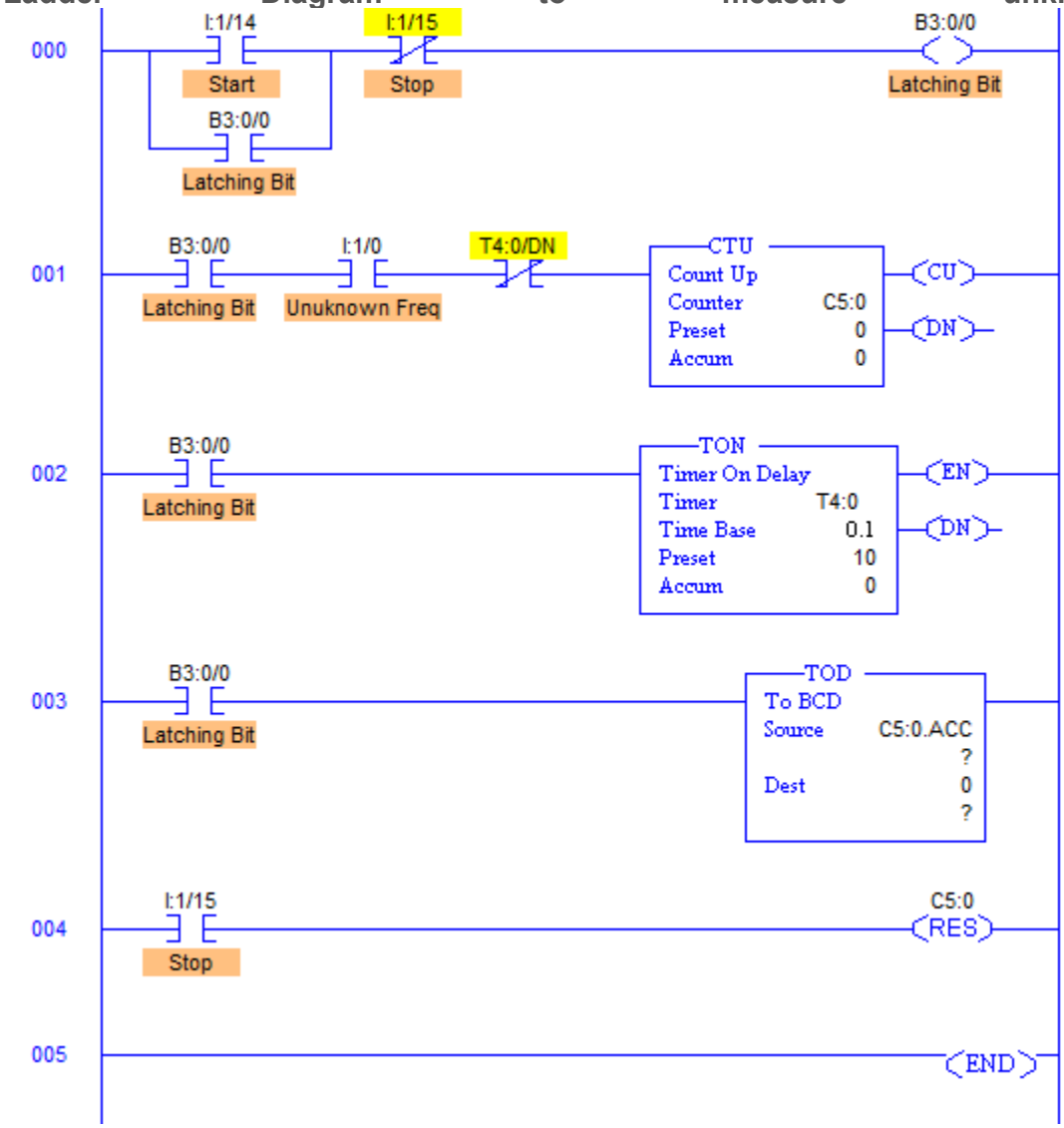
- Define an address to which Unknown frequency is given as an input.
- This address is set to 1 whenever a positive pulse is detected otherwise remain 0.
- And Frequency may be defined as number of cycles per second. This cycle includes three terms, high (positive) pulse, zero and negative pulse.
- Frequency input detects positive pulses.
- Give this input to Up Counter to determine the number of positive pulses per second with unknown frequency.
- Use timer to stop the counter from counting after 1sec.
- Reset counter after measuring of frequency is done.
- However to define a particular Amplitude signals, Limit instruction may be useful.

## PLC Program

Here is PLC program to Measure Unknown Frequency, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 B3:0/0   = Latching Bit                                  (Bit)
 I:1/0    = Unknown frequency input                       (Input)
 I:1/14   = Start PB                                      (Input)
 I:1/15   = Stop PB and reset counter                     (Input)
 T4:1     = 1sec time delay                               (Timer)
 C5:0     = Up counter to increment each positive pulse   (Counter)
 O:6      = Display address                               (Output)
 TOD      = Hex to BCD conversion               (Compute)
 T4:0/DN = Stop counting                         (Timer)
-(RES)-   = Reset Counter coil                            (Output)
```

**Ladder Diagram to measure unknown frequency**




## Program Description

- Latching bit is connected in series with all the rungs except -(RES)- in order to master stop the measuring process.
- Whenever it is required to measure the unknown frequency, I:1/14 Start PB is pressed which activates the Latching rung hence timer is started and counter also allows the positive pulses to increment accumulator value.
- This is stopped by T4:0/DN bit when 1sec is over.
- Stop PB I:1/15 input resets counter and timer both.
- Accumulator value is converted into corresponding BCD number and fed to Display with address O:6.

## Runtime Test Cases

**When it was tested using timer, Time Base = O.1 and Preset = 0.5 replacing I:1/0 with Timer DN bit.**

# PLC Program to Measure the Scan Cycle of PLC

This is a PLC Program to Measure the Scan Cycle of PLC.

## Problem Description

Measure the scan cycle of a PLC using Ladder Diagram programming language.

## Problem Solution

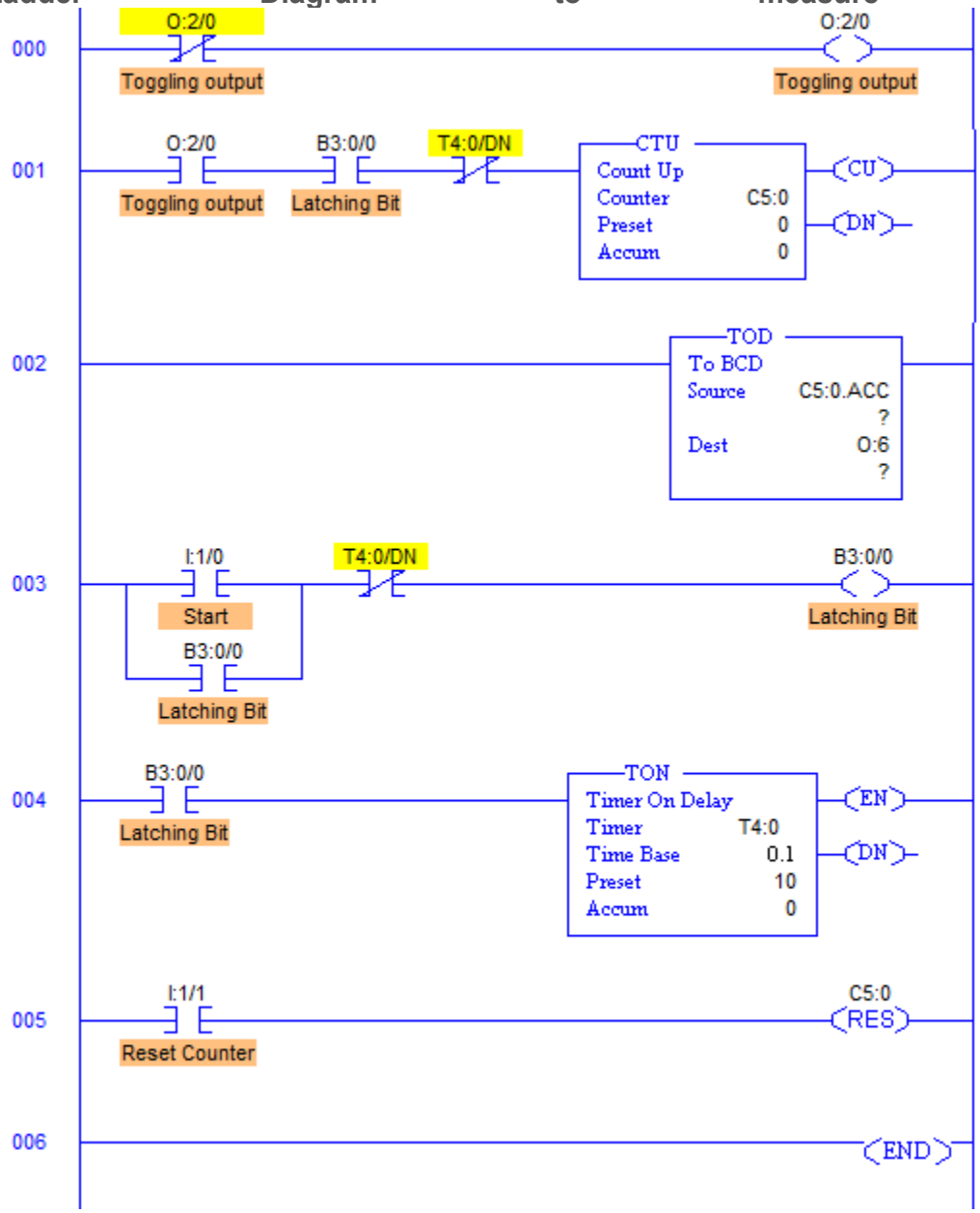
- Use toggling output which toggles every time scan cycle is completed.
- This can be done by connecting an XIO contact and an output in series with the same addresses.
- Use this toggling bit as an input to up counter to increment number of times the bit toggles.
- Use 1sec timer to stop the counter from counting, because the bit is toggling continuously and it might unnecessarily increment counter value even after 1sec is over.
- Reset counter to clear the accumulator data to measure the scan cycle again.

## PLC Program

Here is PLC program to Measure the Scan Cycle of PLC, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 O:2/0    = Toggling output                        (Output)
 B3:0/0   = Latching Bit                           (Bit)
 I:1/0    = Start measurement PB                  (Input)
 I:1/1    = Reset Counter PB              (Input)
 T4:1     = 1sec time delay                        (Timer)
 C5:0     = Up counter to increment each toggle    (Counter)
 O:6      = Display address                        (Output)
 TOD      = Hex to BCD conversion         (Compute)
 T4:0/DN = Stop counting                  (Timer)
 -(RES)- = Reset Counter coil                      (Output)
```

**Ladder Diagram to measure scan cycle**




## Program Description

- When this program is Run, during first scan cycle, output image table memory bit O:2/0 is detected as 0 hence it allows the current through RUNG000 and updates output image table memory bit O:2/0 to 1. During the second scan cycle, O:2/0 is detected 1, hence it blocks current flow through the RUNG000 and output image table memory bit O:2/0 is reset to 0. This process continues till it is stopped manually.
- This toggling of a memory bit is fed to Up Counter with address C5:0 and every time the bit toggles, counter is incremented by 1.
- Scan cycle is defined as the Scans per second (Scans/Sec).
- To fulfill this definition requirement, 1sec timer is added in this program to stop the incrementing of Counter after 1sec when Star PB is pressed which is done by placing XIO of T4:0/DN in series with Up Counter CTU.
- Accumulator value is converted into BCD and fed to Display with address O:6.

## Runtime Test Cases

**Average value of scan cycle in LogixPro simulation software was detected 16 when it was set to Minimum Scans.** 

**Average value of scan cycle in LogixPro simulation software was detected 235 when it was set to Maximum Scans.** 

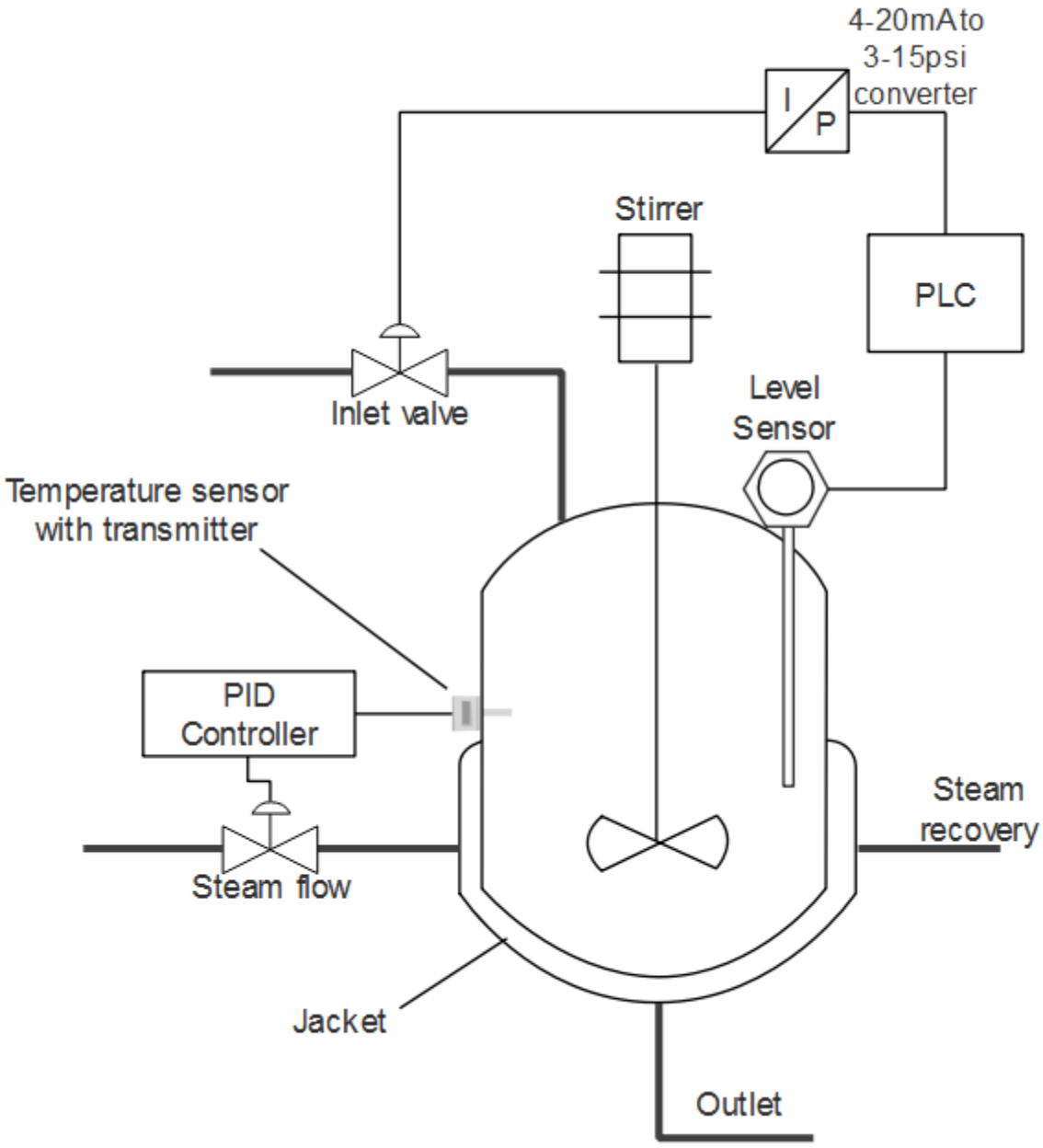

# PLC Program for Continuous Stirred Tank Reactor

This is a PLC Program for Continuous Stirred Tank Reactor.

## Problem Description

Control Continuous Stirred Tank Reactor of a chemical plant. Implement automation to control this CSTR in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

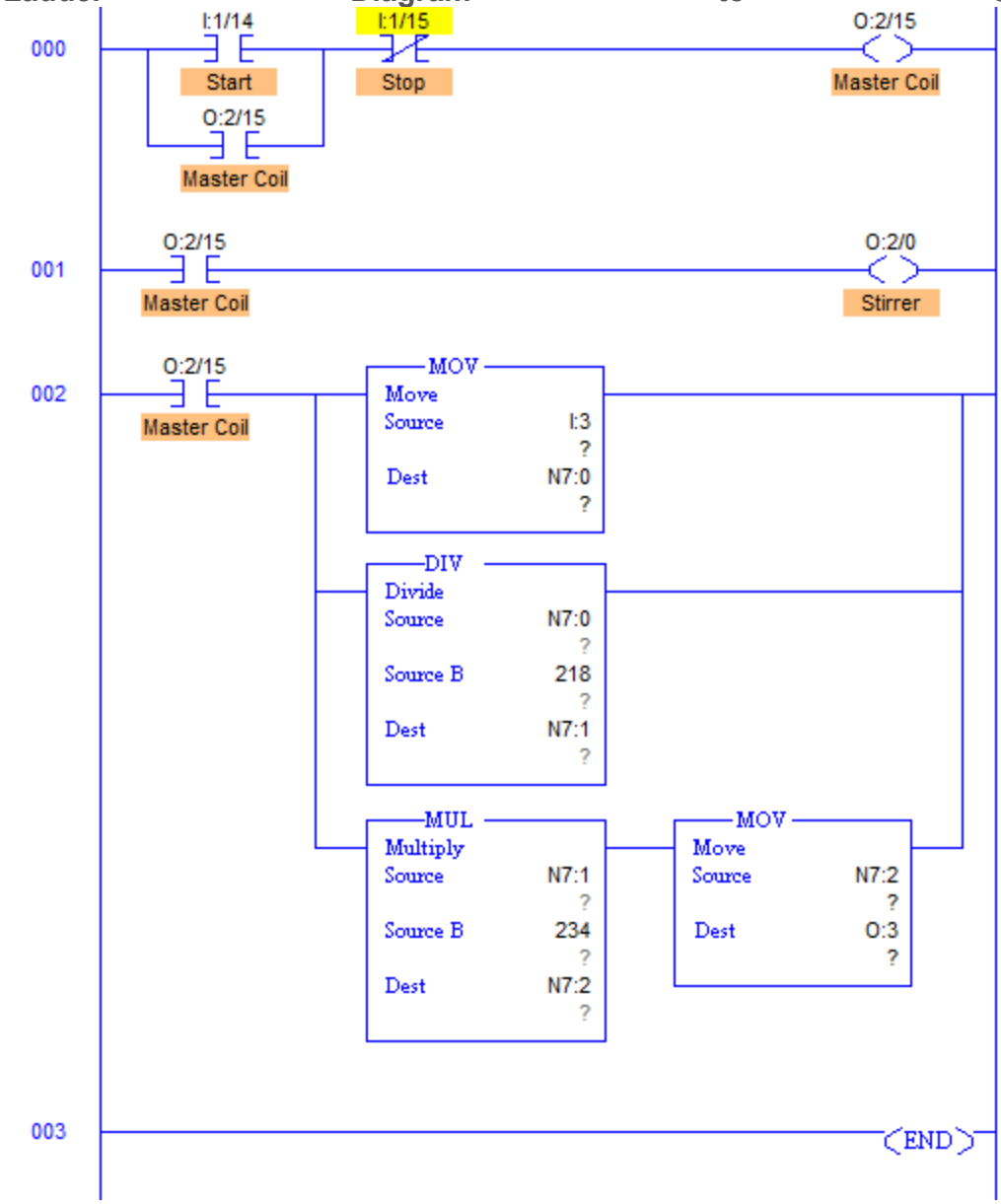
- Basically three parameters are controlled in this reactor. Temperature, Flow and Level of the tank.
- Temperature controlling is best done by PID Temperature controller, so in many industries, controlling of Temperature is assigned to PID control loop as shown in the diagram above.
- Continuous level measurement is required. To do this, capacitance level measurement technique is used because the tank may be Open or Close depending upon the process application. Capacitance Measurement method works for both Open and Closed tanks.
- Capacitance Level Measurement sensor comes along with Transmitter to convert level output into equivalent standard current signal 4-20mA. If level sensor does not have a transmitter, calibration has to be done in order to achieve 4mA for Low level and 20mA for High level.
- Analog I/O Modules are used to deal with analog input signals and analog output final control elements.
- To process data, necessary conversions are made on the basis of desired output requirement.

## PLC Program

Here is PLC program for Continuous Stirred Tank Reactor, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                                                    (Input)
 I:1/15  = Stop                                                     (Input)
 O:2/15  = Latching Coil                                            (Output)
 DIV     = Division by the value which is changed per centimetre    (Compute)
 MUL     = Multiplication                                  (Compute)
 MOV     = Move instruction to move data                          (Logical)
 I:3     = Input from transmitter                          (Input)
 N7:0    = Input data stored in Hex form                            (Register)
 N7:1    = Storing computed value                          (Register)
 N7:2    = Storing conversion of preset cms into equivalent hex     (Register)
 O:3     = Output to which I-P converter is connected               (Output)
```

**Ladder Diagram to control CSTR**




## Program Description

- RUNG001 comprises all the conversion needed to control level of the tank.

- Output of transmitter is in current signals which is 4-20mA.
- When output is 4mA, Analog Input Module converts it into 16bit equivalent hex numbers. Hence when input at I:3 to Analog module is 4mA, it moves 0000h into register and when 20mA, it moves FFFFh into register. Here register N7:0.
- Here height of the tank is 3m or 300cm. By converting it into equivalent hex, change in value per centimeter is 218.
- Value of N7:0 is then multiplied with 234 because when Level reaches 280cm, output is 61167 in decimal (EEEFh). So when output at 280cm is multiplied with 234, we get full FFFFh at N7:2 to operate valve to fully close.
- This multiplication is stored into N7:2 register. Digital to Analog conversion of value stored in N7:2 is performed inside the processor and equivalent mA current is received from terminal O:3.
- Current to Pneumatic converter then converts current signals into equivalent 3-15psi pneumatic signal and adjusts valve opening.

## Runtime Test Cases

```
Inputs          Outputs          Physical Elements
I:3 = 0000h      O:3 = 0000h      Valve fully open
I:3 = EEEFh      O:3 = FFFFh      Valve fully closed
I3 = 7778h       O:3 = 8000h      Valve 50% open
```

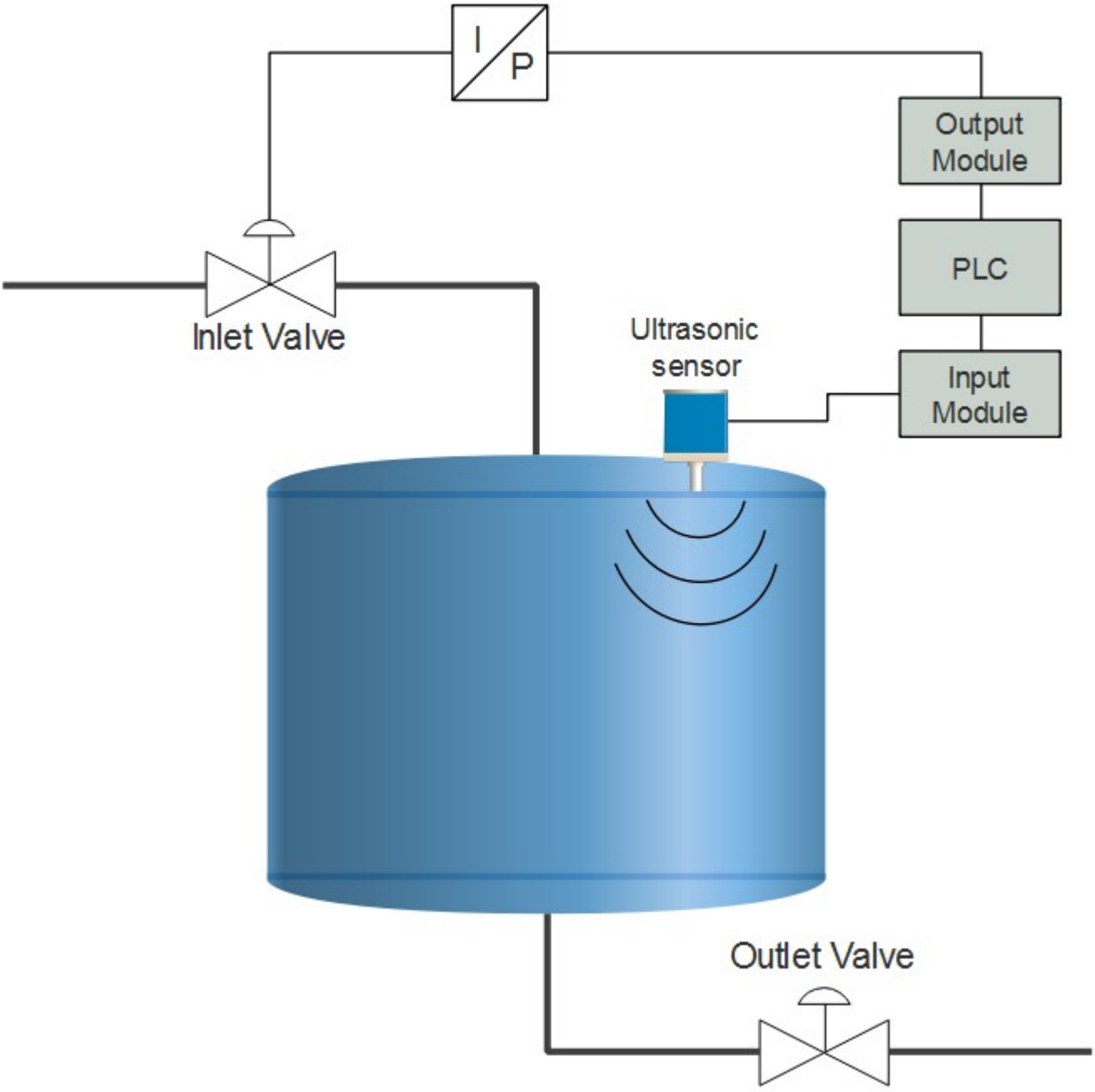
# PLC Program to Maintain Level of a Tank

This is a PLC Program to Maintain Level of a Tank.

## Problem Description

By using a control valve which is operated with standard pneumatic signal 3-15psi, level of a tank is to be maintained. Implement automation of this tank level system in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- To measure the continuous level, Ultrasonic Sensor may be used.
- Rosemount 3100 series Ultrasonic Level Transmitters can be used to measure the level of the tank.
- Range of these transmitters is 9-11m.
- Output of these transmitters is in standard electrical signal 4-20mA. Convert this standard signal into digital signal by connecting level transmitter to Analog Input Module of PLC.
- Calibrate the transmitter and set output such that it gives 4mA when tank is empty and 20mA when tank is full. Convert this data in accordance to level in Centimeters.

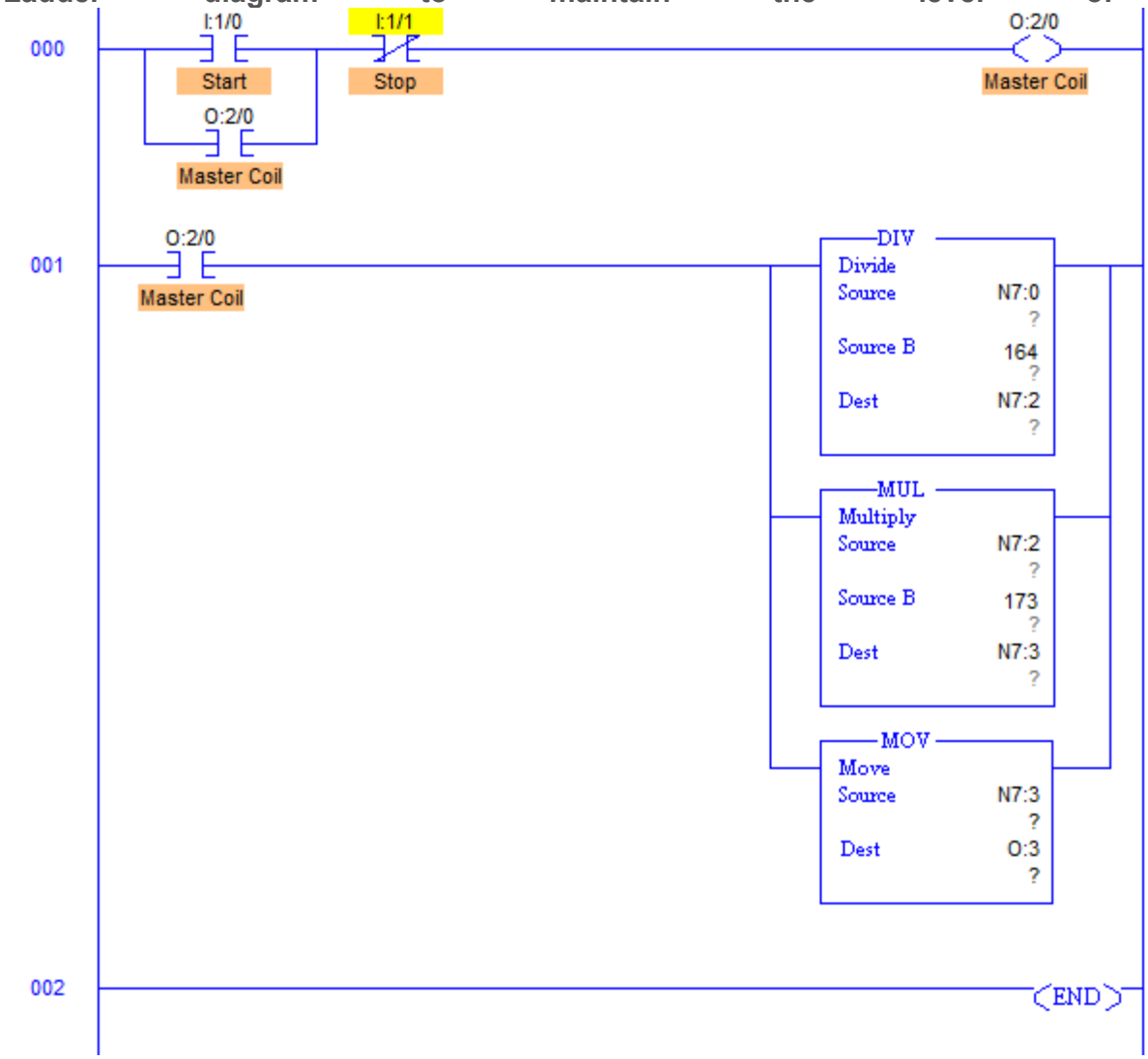
- This gives output such as 0000h when input is 4mA and FFFFh when output is 20mA. Determine the level of tank to be maintained. Convert preset value to be the maximum output that means if the level to be maintained is 370cm, then convert value such that output is FFFFh when level is 370cm.
- At low level, valve must be fully open, so air to close valve is used here to maintain the inlet flow.

## Program

Here is PLC program to Maintain Level of a Tank, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start                                              (Input)
 I:1/1   = Stop                                               (Input)
 O:2/0   = Latching Coil                                      (Output)
 DIV     = Division block                                (Compute)
 MUL     = Multiplication block                          (Compute)
 MOV     = Move instruction to move data                      (Logical)
 N7:0    = Input from transmitter in Hex form                 (Register)
 N7:2    = Storing conversion FFFFh into equivalent 400cms  (Register)
 N7:3    = Conversion of preset cms into equivalent hex       (Register)
 O:3     = Output to which I-P converter is connected         (Output)
```

**Ladder diagram to maintain the level of a tank**




## Program Description

- RUNG000 is a latching rung to operate the system through Master Start and Stop PB.
- RUNG001 comprises all the conversion needed to control level of the tank.
- Output of transmitter is in current signals which is 4-20mA.
- When output is 4mA, Analog Input Module converts it into 16bit equivalent hex numbers. Hence when input to Analog module is 4mA, it stores 0000h into register and when 20mA, it stores FFFFh into register. Here register N7:0.
- Let's say here height of the tank is 4m or 400cm. By converting it into equivalent hex, change in value per centimeter is 164.
- Value of N7:0 is then multiplied with 173 because when Level reaches 380cm, output is 62259 in decimal. So when output at 380cm is multiplied with 173, we get full FFFFh at N7:2 to operate valve to fully close.
- This multiplication is stored into N7:2 register. Digital to Analog conversion of value stored in N7:2 is performed inside the processor and equivalent mA current is received from terminal O:3.
- Current to Pneumatic converter then converts current signals into equivalent 3-15psi pneumatic signal and adjusts valve opening.

## Runtime Test Cases

```
Input           Output   Valve percentage open
N7:0 = 0000h     O:3 = 0000h      Valve fully open
N7:0 = F333h     O:3 = FFFFh      Valve fully closed
N7:0 = 7999h     O:3 = 7FFFh      Valve 50% open
```

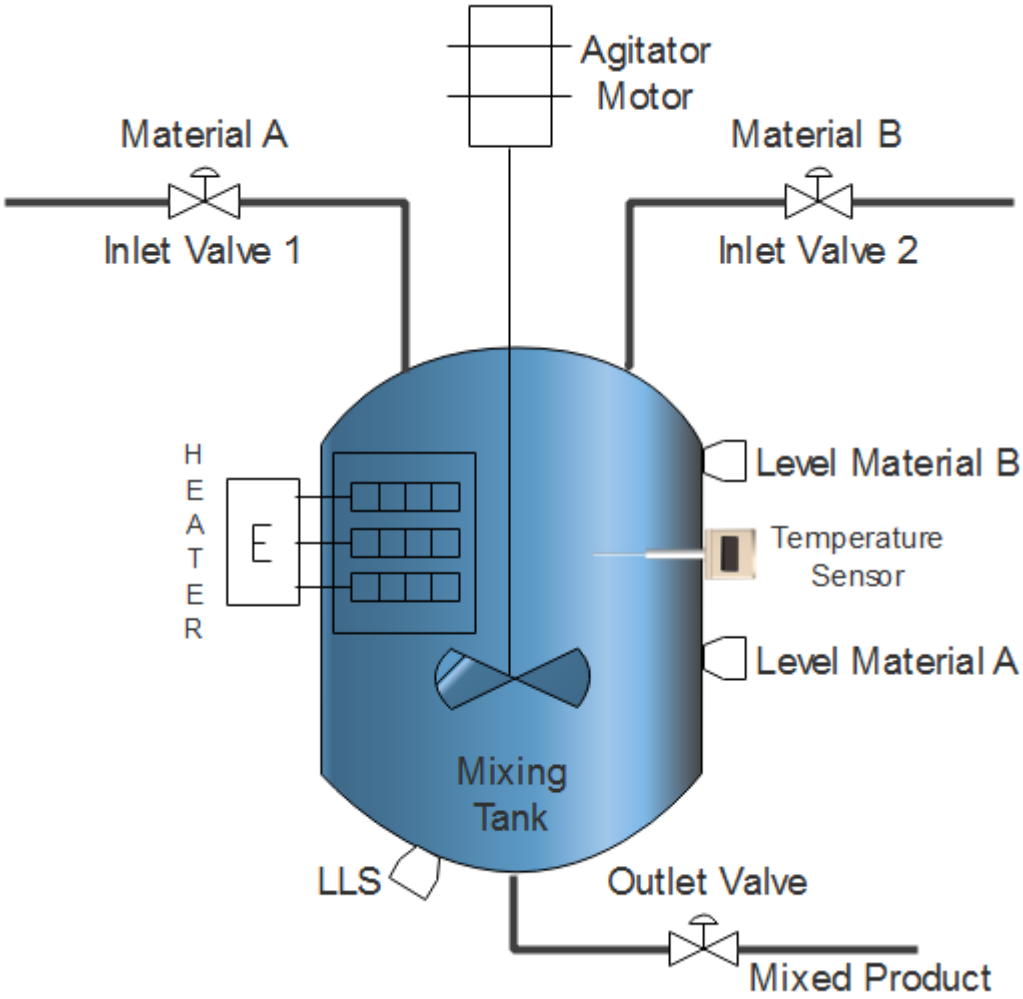
# PLC Program for Heating and Mixing of Products

This is a PLC Program for Heating and Mixing of Products.

## Problem Description

Material A and Material B are collected in a tank. These products are to be mixed and heated till it reaches set point temperature value. Implement automation of this system in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- To detect level of Material A and Material B, two separate level switches are used.
- And to detect low level, one more level switch is used at the bottom of the tank.
- These give output in digital terms that is when corresponding levels are detected.
- To control level of this system, Single Acting Piston valve can be used which has two states, either fully open or fully close.
- To control mixing, agitator is used which is connected with Motor shaft.
- Heater is installed inside the tank and temperature sensor such as RTD or Thermocouple is used to detect the temperature of liquid in the tank.
- This mixer is mixed until set point temperature is achieved.
- Outlet valve is then operated to drain the mixed product.

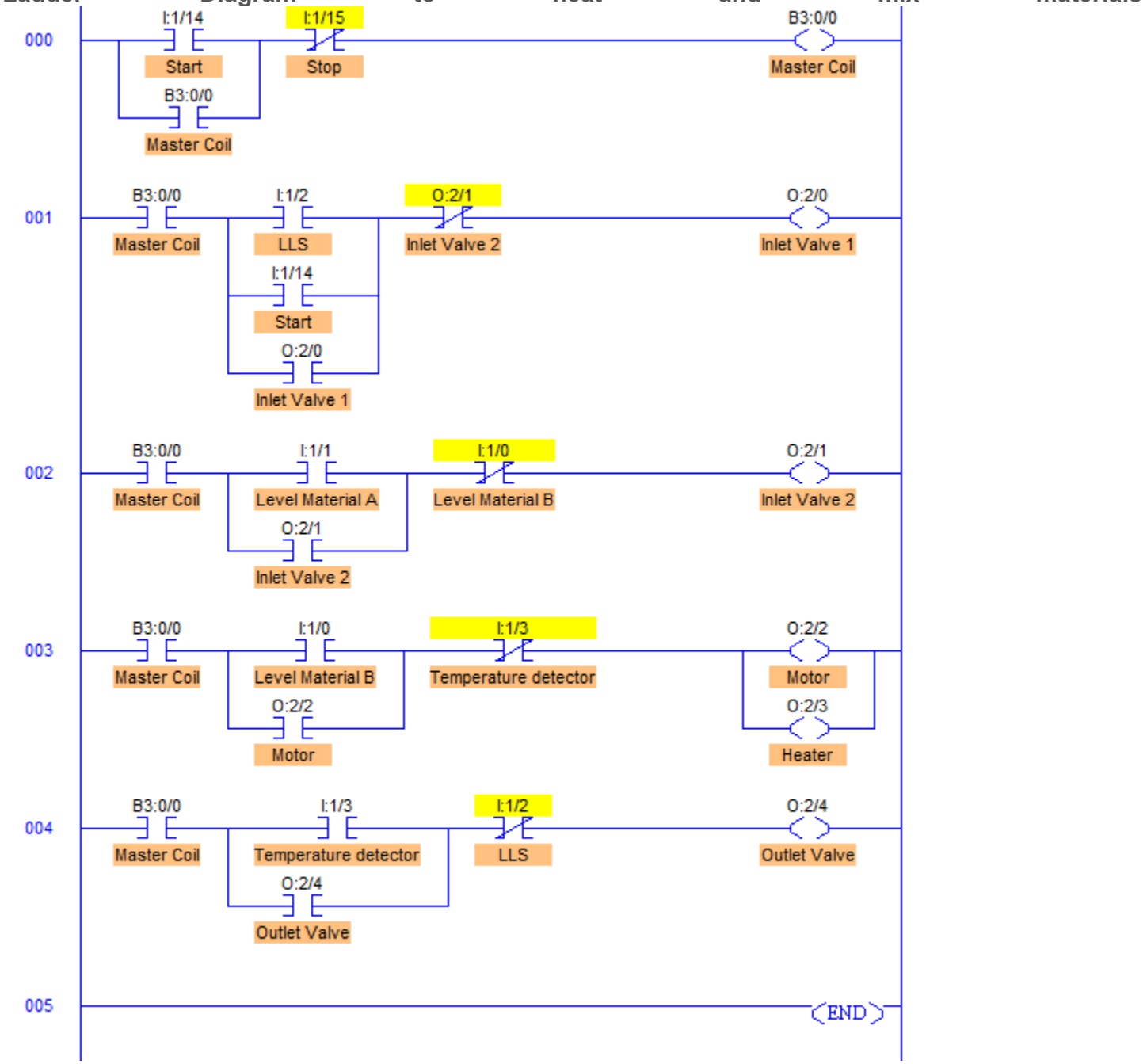
## PLC Program

Here is PLC program for Heating and Mixing of Products, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                          (Input)
 I:1/15  = Stop                                  (Input)
 B3:0/0  = Master Coil Bit                       (Bit)
 I:1/0   = Level of Material B                   (Input)
 I:1/1   = Level of Material A                   (Input)
```

```
I:1/2   = Low Level Switch (empty tank)          (Input)
O:2/0   = Inlet Valve 1 (Material A Feed)(Output)
O:2/1   = Inlet Valve 2 (Material B Feed)(Output)
O:2/2   = Agitator Motor (Mixing)                (Output)
O:2/3   = Heater                          (Output)
O:2/4   = Outlet valve to drain the product      (Output)
```

**Ladder Diagram to heat and mix materials**




## Program Description

- RUNG000 contain master start/stop with address of Start PB I:1/14 and Stop PB I:1/15.
- RUNG001 is to operate Inlet Valve of Material A with address O:2/0. It is operated when Low Level of the tank is detected by Level Low Switch I:1/2. And it is closed when Inlet Valve of material B starts that is when Level Material A is detected by a switch with address I:1/1. Start PB is also connected in parallel with the LLS, it is done so that when LLS or level of Material A is not detected, Inlet Valve is operated by Start PB.

- RUNG002 is to operate Inlet Valve of Material B with address O:2/1. It is opened when Material A is filled to its desired level (Level Material A). In other words, Valve of Material B is opened when Level of Material A is detected by I:1/1 and it is closed when Level of Material B is detected or Tank is full.
- RUNG003 operates Agitator Motor and Heater with address O:2/2 and O:2/3 respectively. When the tank is full with Material A and B, Level High (Level Material B) is detected. Agitator Motor is connected to the address O:2/2 and Heater to O:2/3. Agitator and Heater are run when I:1/0 is set to 1 that is when tank is full.
- RUNG004 is for Outlet Valve with address O:2/3. It is operated when the entire mixing and heating process is completed that is when temperature set point is reached. And this valve closed again is closed when LLS is again detected.

## Runtime Test Cases

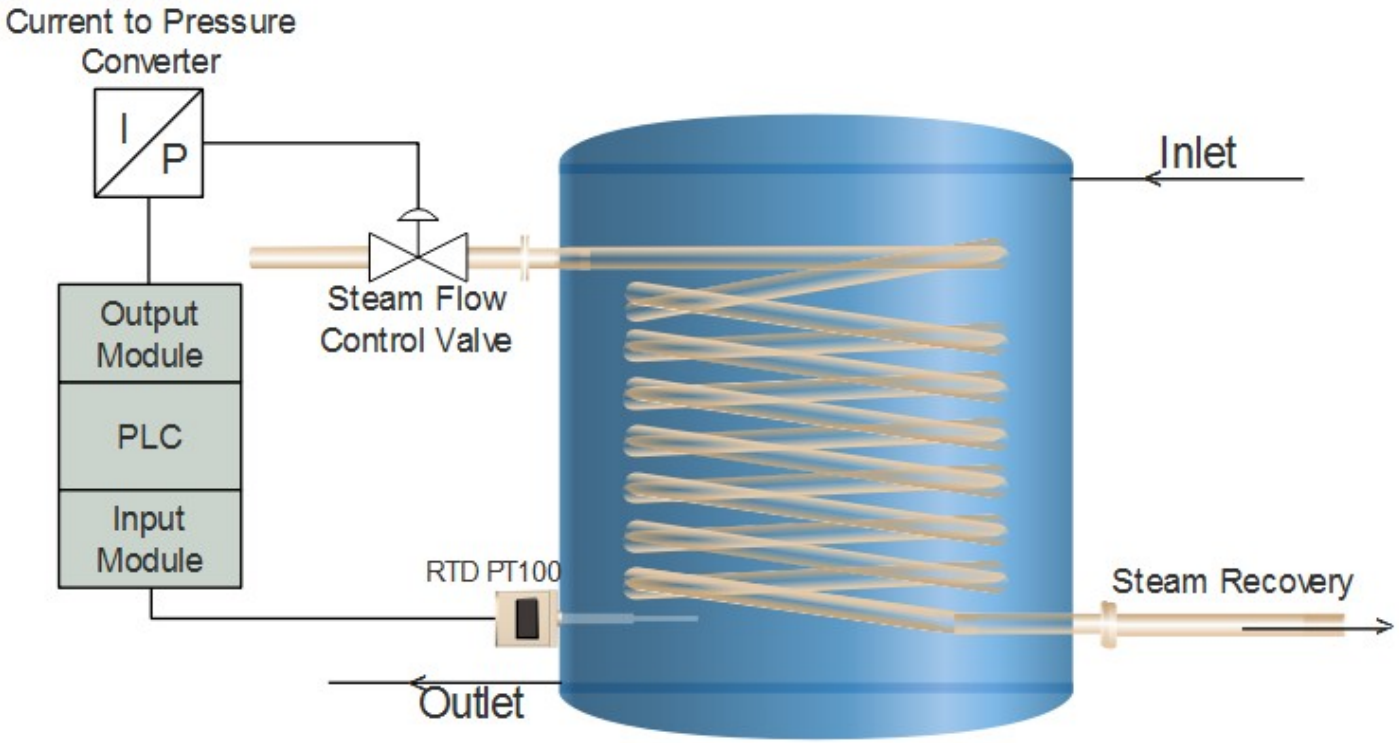
| Inputs | Outputs | Physical Elements |
|---|---|---|
| I:1/2 = 1 | O:2/0 = 1 | Open Inlet Valve 1 |
| I:1/2 = 0 and I:1/1 = 0 | O:2/0 = 1 | Open Inlet Valve 1 |
| I:1/1 = 1 | O:2/0 = 0 | Close Inlet Valve 1 |
| I:1/1 = 1 | O:2/1 = 1 | Open Inlet valve 2 |
| I:1/0 = 1 | O:2/1 = 0 | Close Inlet Valve 2 |
| I:1/0 = 1 | O:2/2 = 1, O:2/3 = 1 | Run Agitator Motor and Heater |
| Temperature ≥ 70°C | O:2/2 = 0, O:2/3 = 0 | Stop Agitator Motor and Heater |
| Temperature ≥ 70°C | O:2/4 = 1 | Open Outlet Valve |
| I:1/2 = 1 | O:2/4 = 0 | Close Outlet Valve |

# PLC Program to Heat the Liquid in Tank by Steam Flow

This is a PLC Program to Heat the Liquid in Tank by Steam Flow.

## Problem Description

**Heating of the liquid in the tank is to be performed. To heat this liquid, steam flow is controlled. If the temperature is detected less than the set point, increase the steam flow and vice-versa. Implement automation of this process in PLC using Ladder Diagram programming language**



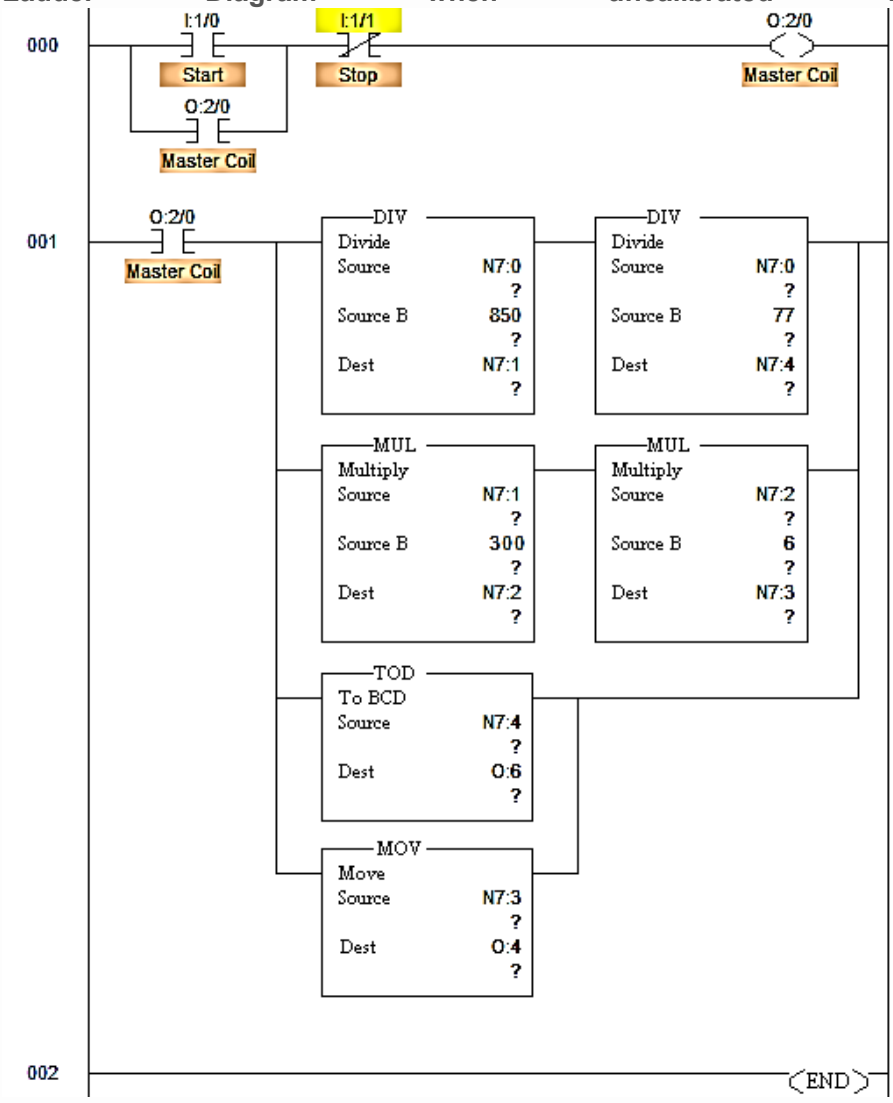

## Problem Solution

- To detect temperature, RTD PT100 is used. This RTD has a range from -200°C to 650°C.
- Let's say that the temperature of liquid in the tank is to be maintained at 100°C. Calibrate RTD to operate in this range.
- RTD PT100 has a resistance of 100Ω when it is at 0°C and 138.4Ω when at 100°C.
- Calibrate 4-20mA output such that when input is 0°C, it gives 4mA output and when it is 100°C, it gives output 20mA.
- Feed this data to Analog Input Module of PLC. Input module converts 4-20mA into equivalent 0000h to FFFFh hex number. This means when temperature is 0°C, data stored in the register is 0000 0000 0000 0000, and when output is 100°C, data stored in register is 1111 1111 1111 1111.
- Analog Output module converts this data into equivalent 4-20mA which is then converted into 3-15psi to operate control valve.
- One more method to solve this problem is by calculating Output in terms of 4-20mA for -200°C-650°C that when -200°C, it gives 4mA and when 650°C, it gives 20mA.
- But then this would not be precise control action because data conversion gives minor errors while performing conversion.

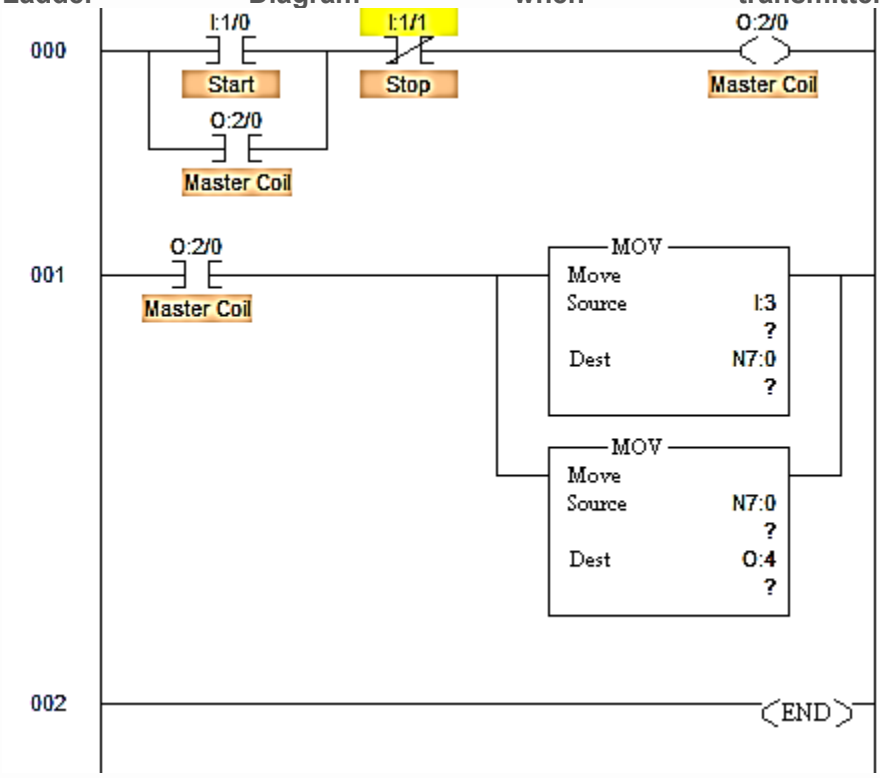
## PLC Program

- Here is PLC program to Heat the Liquid in Tank by Steam Flow, along with program explanation and run time test cases.
- `List of Inputs and Outputs`

- I:1/0 = Start (Input)
- I:1/1 = Stop (Input)
- O:2/0 = Mater Coil (Output)
- O:4 = Output to I-P converter (Output)
- O:6 = Display (Output)
- N7:0 = Register input data (Register)
- N7:1 = answer of division by total range (Register)
- N7:2 = Answer after multiplied by the set point (Register)
- N7:3 = Convert set point as maximum value (Register)
- N7:4 = Conversion to display temperature (Register)
- I:3 = Input data (4-20mA) (Input)
- N7:0 = Register input data (Register)
- Ladder Diagram when uncalibrated Transmitter is used



- Ladder Diagram when transmitter is calibrated




## Program Description

- RUNG001 comprises all the conversion needed to control level of the tank.
- Output of transmitter is in current signals which is 4-20mA.
- When output is 4mA, Analog Input Module converts it into 16bit equivalent hex numbers. Hence when input to Analog module is 4mA, it stores 0000h into register and when 20mA, it stores FFFFh into register. Here register N7:0.
- Temperature range of RTD PT100 is -200°C-650°C. By converting it into equivalent hex, change in value per centimeter is 77.
- Value of N7:1 is then multiplied with the preset value of liquid temperature. That is here 300°C.
- This multiplication is stored into N7:2 register. This gives data which is again multiplied by 6 and stored in register N7:3 which gives full output when preset is reached. Digital to Analog conversion of value stored in N7:3 is performed inside the processor and equivalent mA current is received from terminal O:4.
- Current to Pneumatic converter then converts current signals into equivalent 3-15psi pneumatic signal and adjusts valve opening.
- In other method, when transmitter is calibrated, transmitter is tested in the laboratory and calibrated such that at 0°C, it gives output 4mA and at 100°C it gives 20mA.
- This 4-20mA can be directly converted into hex from 0000h to FFFFh. Then programming would be to just move this converted data into output which again converts 0000h-FFFFh into 4-20mA equivalent to actuate control valve.

http://www.sanfoundry.com/plc-program-heat-liquid-tank-steam-flow/

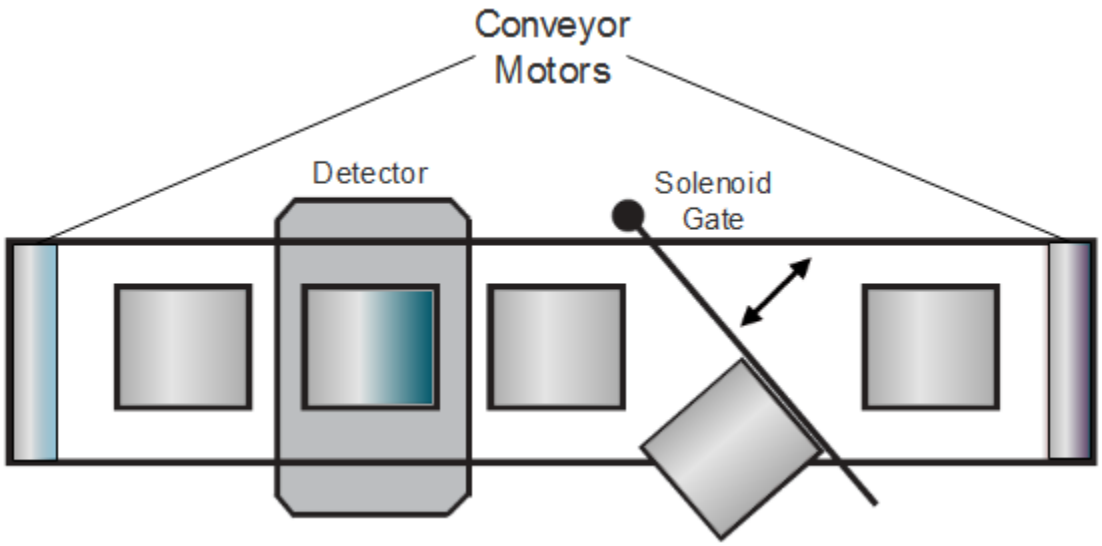
# PLC Program to Sort Parts for Quality Control on Conveyor

This is a PLC Program to Sort Parts for Quality Control on Conveyor.

## Problem Description

Parts are moving on the conveyor from one process line to other with a constant speed. Out of 1000 part, one part is taken out for quality check. Implement automation of this process in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

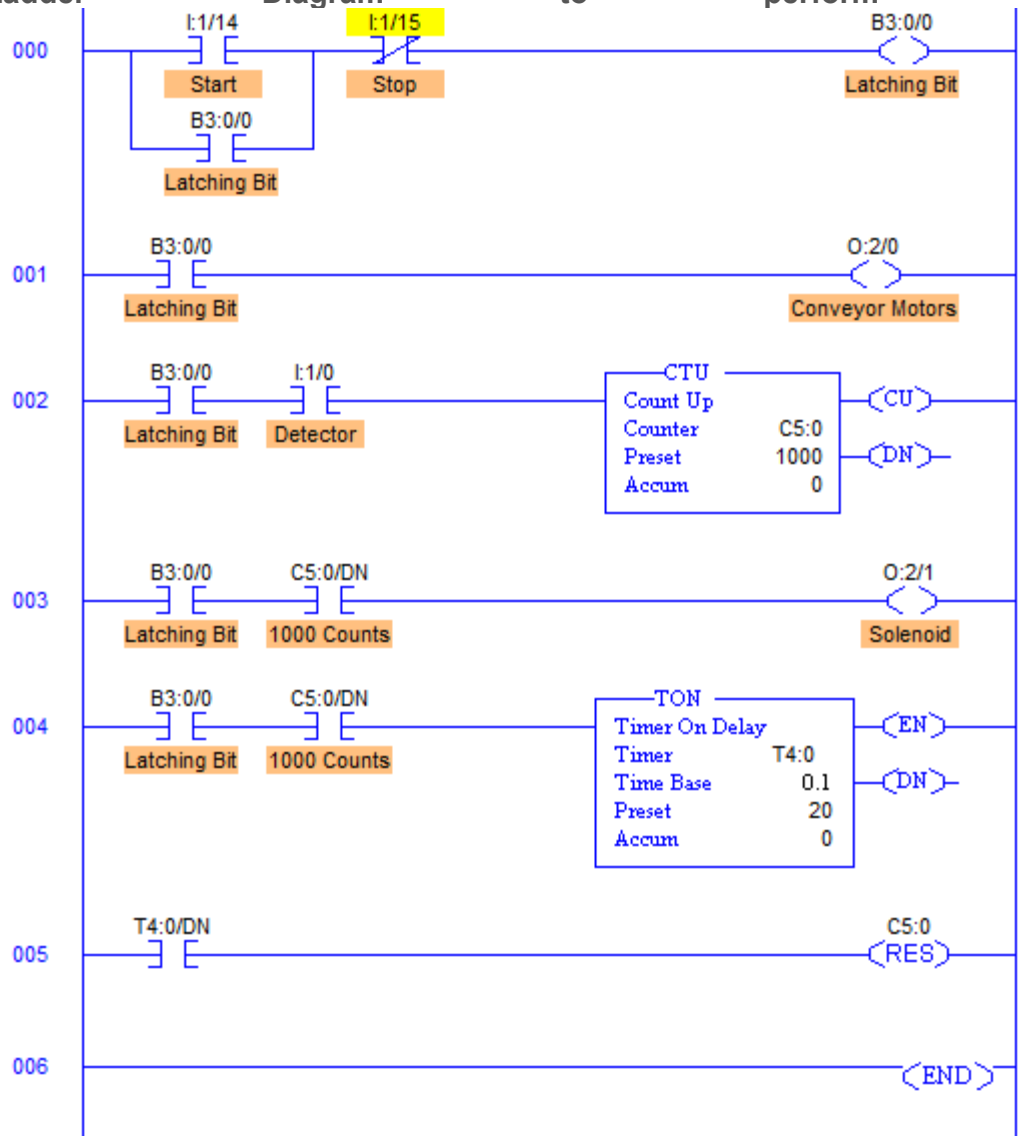
- To detect the parts, detector such as proximity switch, optical sensors or any other sensor is used.
- Connect output of this detector to Input Module of PLC which sets and resets image memory according to parts' detection.
- Give this detection, as an input to Up Counter which is incremented with each part's detection.
- Set counter preset value to 1000.
- Operate Solenoid for a few seconds until the part is diverted for quality check.

## PLC Program

Here is PLC program to Sort Parts for Quality Control on Conveyor, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14   = Start                          (Input)
 I:1/15   = Stop                                  (Input)
 I:1/0    = Detector input                        (Input)
 B3:0/0   = Latching Coil                  (Bit)
 O:2/0    = Conveyor Motors                       (Output)
 O:2/1    = Solenoid to operate gate              (Output)
 C5:0     = Up Counter to count 1000 parts (Counter)
 T4:0     = Timer to operate solenoid             (Timer)
 -(RES)- = Reset counter value                    (Timer/Counter)
```

**Ladder Diagram to perform this operation**




## Program Description

- RUNG000 is Master Start and Stop the process.
- RUNG001 operates Conveyor Motors with address O:2/0 to start moving parts to other process. This is started as soon as Start PB I:1/14 is pressed.
- RUNG002 comprises Up Counter which counts the number of parts detected by the detector which is connected to I:1/0. Whenever a part is detected, I:1/0 goes high incrementing accumulator value of C5:0 Counter.
- When 1000 parts are counted, done bit is generated which is used to operate Solenoid Coil in RUNG003. It allows the current to pass and solenoid is operated.
- Assuming that it takes 2secs to divert the part for quality check, 2secs of timer T4:0 is used. This timer bit T4:0/DN resets the counter value to 0 which in turn unlatches solenoid coil O:2/1 taking gate to its main position.

## Runtime Test Cases

```
Inputs                    Outputs                Physical Elements
B3:0/0 = 1                O:2/0 = 1              Run Conveyor Motors
I:1/0 = 1 (Momentarily)    Accumulator = +1     Increment Counter
C5:0.ACC = 1000           O:2/1 = 1              Energize Solenoid
T4:0/DN = 1               C5:0.ACC = 0(Reset)  Reset Counter, De-energize Solenoid
```

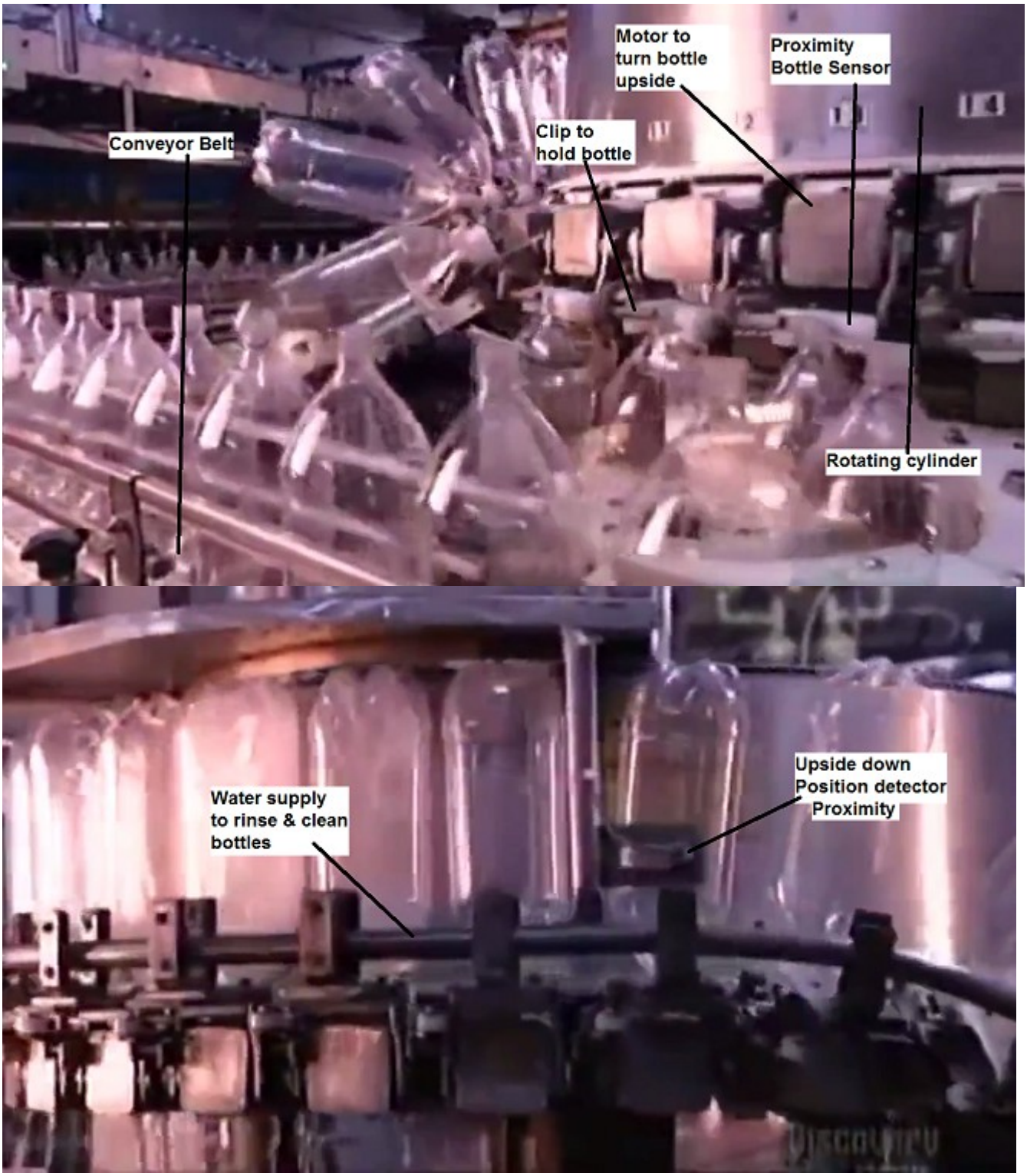
# PLC Program for Cleaning and Rinsing Bottles in Beverage Industries

This is a PLC Program for Cleaning and Rinsing Bottles in Beverage Industries.

## Problem Description

Number of bottles are moving on the conveyor belt. Cleaning of Rinsing of these bottles is to be performed. Implement automation of this process in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- A huge cylinder carries all the bottles through this process which rotates on a fixed RPM.
- Use a switch to sense the bottle from the conveyor belt.
- Conveyor belt also has a fixed speed.
- Mechanical clip is used to hold the bottle tightly throughout the process. This mechanical switch has two positions, hold and return.
- Motor is used to turn the bottles. Motor is used such that it rotates only at 180o angle turning bottles upside down.

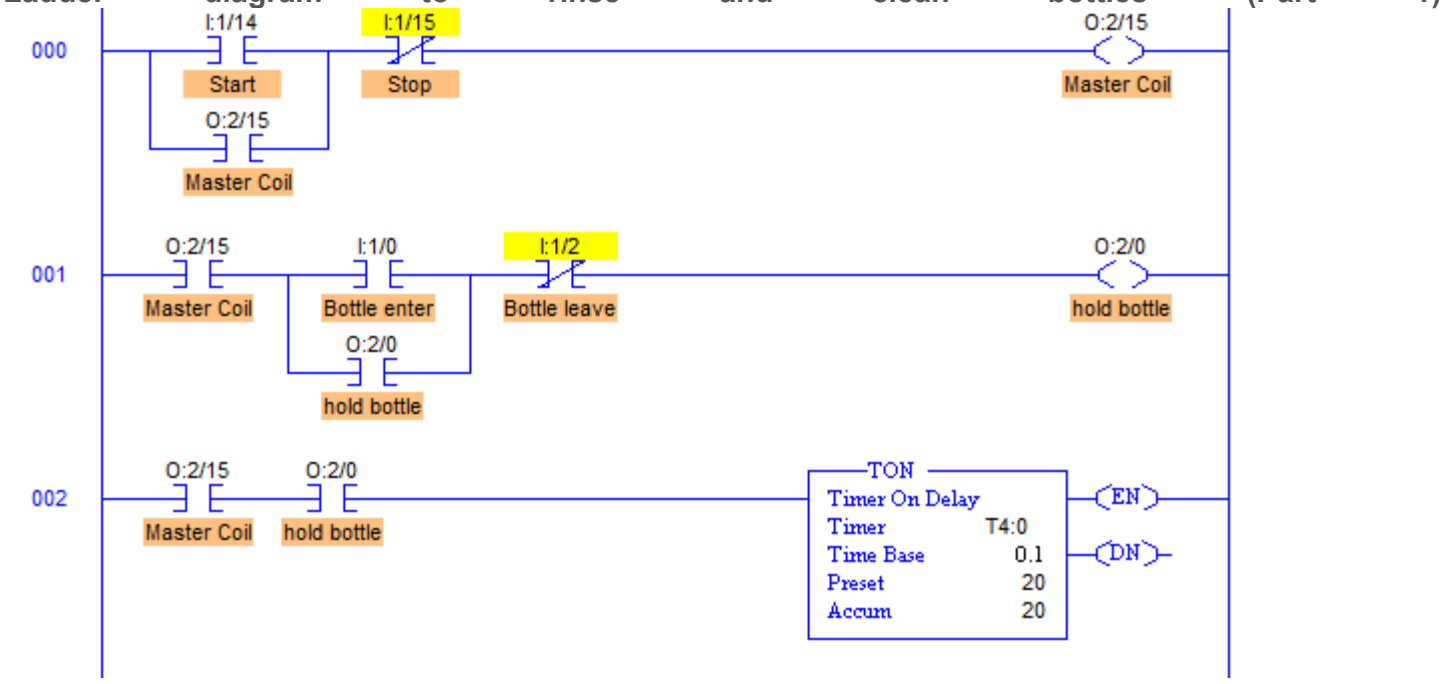
- Along with the cylinder, water supply tubes are fixed in the position and solenoid valves are used to sprinkle water inside the bottles.
- Proximity switch is used to detect the upside position of a bottle to perform rinsing and cleaning.
- There are total 30 number of bottles cleaned and rinsed simultaneously in a process stream. Let us program cleaning of a single bottle.

## PLC Program

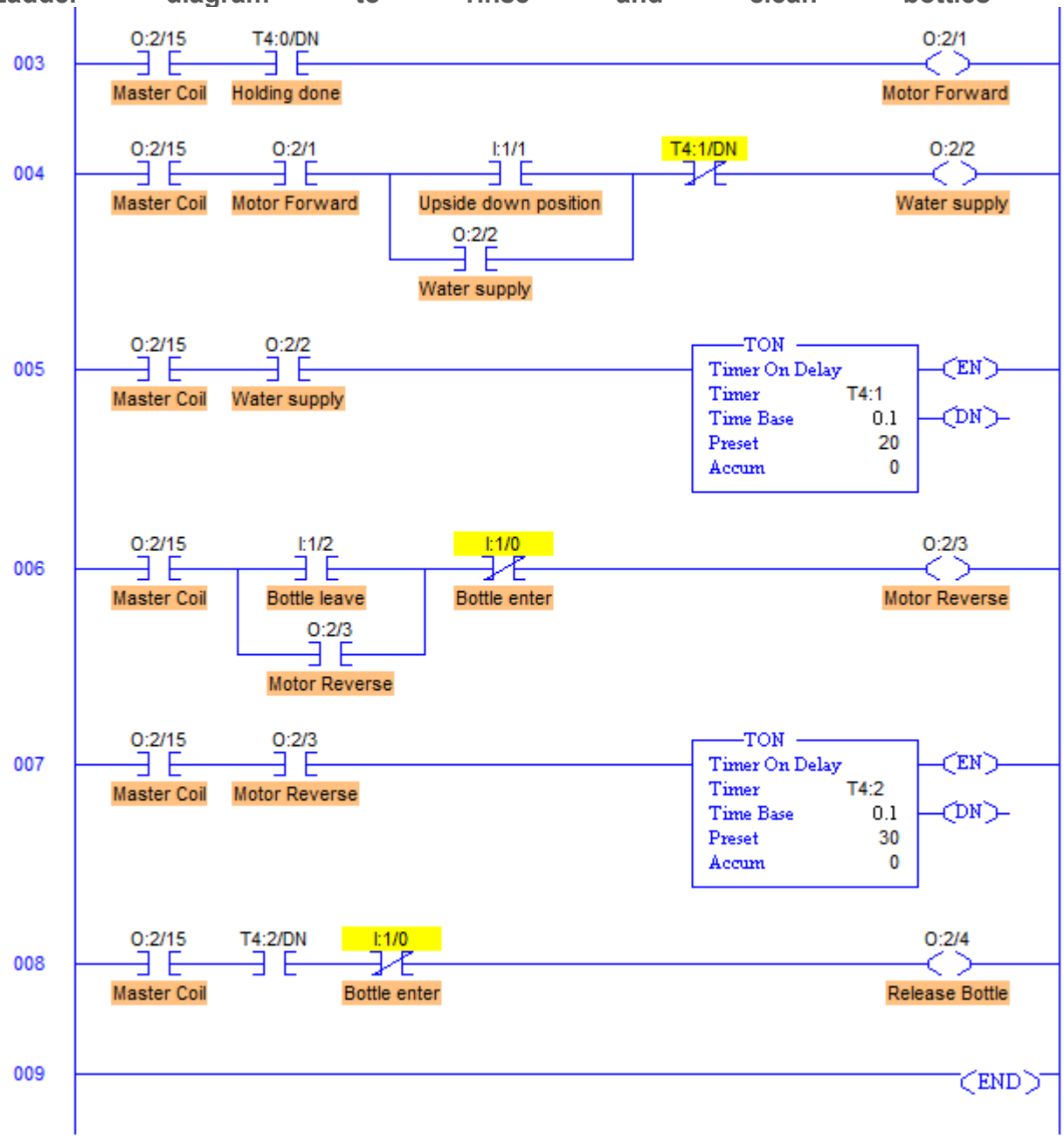
Here is PLC program for Cleaning and Rinsing Bottles in Beverage Industries, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                                          (Input)
 I:1/15  = Stop                                           (Input)
 O:2/15  = Master Coil                                    (Output)
 I:1/0   = Proximity bottle sensor (Enter)         (Input)
 I:1/1   = Upside down position detector (proximity)      (Input)
 I:1/2   = Proximity bottle sensor (Leave)         (Input)
 O:2/0   = Clip bottle (Hold)                             (Output)
 O:2/1   = Motor to turn bottle (Forward)          (Output)
 O:2/2   = Water supply (Solenoid coil)                   (Output)
 O:2/3   = Motor to turn bottle (Reverse)          (Output)
 O:2/4   = Clip bottle (Release)                          (Output)
 T4:0    = Timer to turn bottle                           (Timer)
 T4:1    = Timer for solenoid coil to sprinkle water      (Timer)
 T4:2    = Timer to leave bottle                          (Timer)
```

**Ladder diagram to rinse and clean bottles (Part 1)**



**Ladder diagram to rinse and clean bottles (Part 2)**




## Program Description

- RUNG001 is to operate Clipping of a bottle. When I:1/0 is detected which is an address of Bottle Enter proximity switch, O:2/0 energizes which is connected to perform first position of the clip which holds the bottle.
- RUNG002 & RUNG003 are to energize O:2/1 which operates rotating of a bottle by rotating motor in forward direction by 1800, bottle is turned upside down after 2secs bottle is held.
- RUNG004 & RUNG005 are used to energize O:2/2 which is connected to the solenoid valve coil to sprinkle water inside from the bottom of upside down bottle. This is done when I:1/1 proximity detects the presence of upside down bottle. Water sprinkling is done for 2secs and solenoid coil is de-energized.
- RUNG006 is used to de-energize motor forward coil and energize motor reverse coil with which is connected to O:2/3 to take bottle back to its main position. This is performed when I:1/2 detects the presence of a bottle to leave.
- RUNG007 and RUNG008, when O:2/3 is energized, after 3secs, Bottle is released by the clip. O:2/3 is the output to activate 2nd position of the clip which is to leave a bottle.

## Runtime Test Cases

```
Inputs                    Outputs    Physical Elements
I:1/0 = 1                 O:2/0 = 1  Operate clip to position1 to hold bottle
O:2/0 = 1 & T4:0/DN = 1   O:2/1 = 1  Turn bottle upside down
I:1/1 = 1                 O:2/2 = 1  Sprinkle water inside the bottle
T4:1/DN = 1               O:2/2 = 0  Stop sprinkling
I:1/2 = 1                 O:2/3 = 1  Turn bottle back to its main position
T4:2/DN = 1               O:2/4 = 1  Operate clip to position2 to release bottle
```

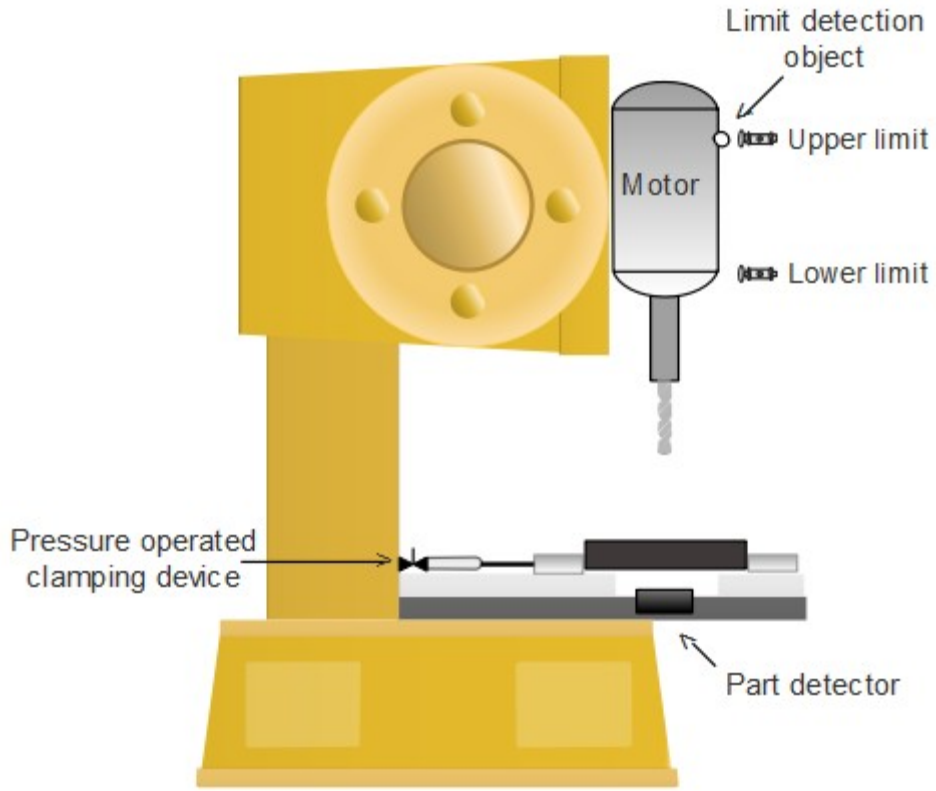
# PLC Program to Operate Drilling of Parts

This is a PLC Program to Operate Drilling of Parts.

## Problem Description

Whenever a part is placed on the drilling table, pneumatic clamper clamps the part and drilling process is done. When drilling is done, clamper releases the part by releasing pressure. When another part is detected, the process is repeated. Implement this in PLC using Ladder Diagram programming language.

## Problem Diagram



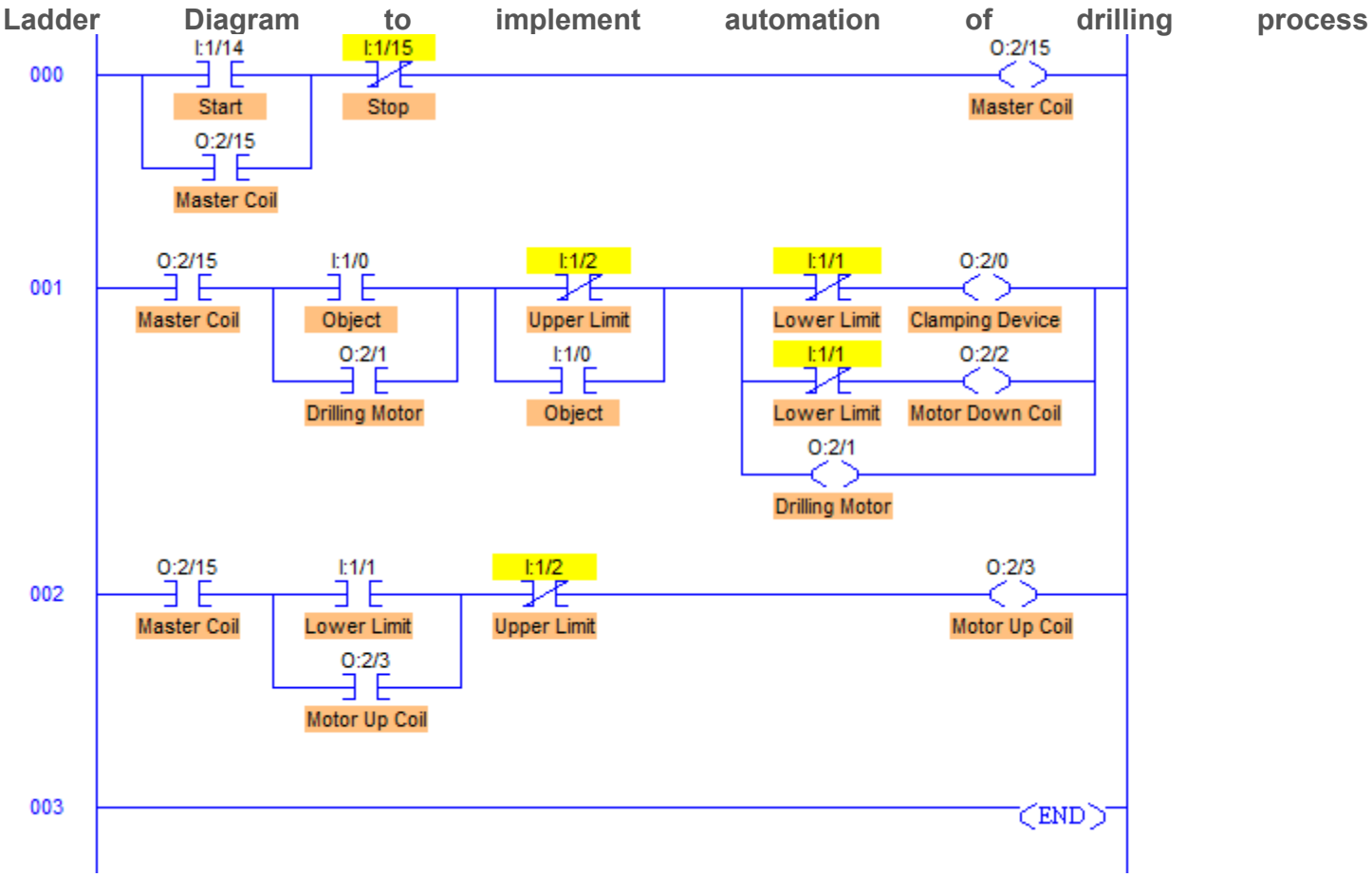

## Problem Solution

- Set lower and upper limit of a motor to stop and start the drilling process. This is done for precise drilling and to obtain uniformity.
- Pressure operated clamping device is used to hold the objects firmly. This is operated by 20psig air supply which is provided when an object is detected.
- Limit detection object is placed on the motor to detect upper and lower limit by the switches.

# PLC Program

Here is PLC program to Operate Drilling of Parts, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                  (Input)
 I:1/15  = Stop                   (Input)
 O:2/15  = master coil            (Output)
 O:2/0   = Clamping Device        (Output)
 O:2/1   = Drilling Motor (Output)
 O:2/2   = Motor Down             (Output)
 O:2/3   = Motor Up               (Output)
 I:1/0   = Object detect switch   (Input)
 I:1/1   = Lower Limit            (Input)
 I:1/2   = Upper Limit            (Input)
```




# Program Description

- RUNG001 is to operate Clamping device with address O:2/0, Motor Down Coil O:2/2 and Drilling Motor O:2/1.
- These outputs are operated when an object is detected.
- Clamping device and Motor Down coil is de-energized when Lower Limit of drilling motor is detected which is connected to I:1/1. And Drilling Motor O:2/1 is de-energized when Upper Limit I:1/2 is detected.
- RUNG002 is to operate Motor Up coil with address O:2/3. When Lower limit is reached that means drilling is completed. Next operation is to take Drilling motor back to its main position and for that reverse coil Motor Up coil with address O:2/3 is energized.
- When it is reached to its main position that is Upper Limit I:1/2, reversing Motor Up coil is de-energized.

## Runtime Test Cases

```
Inputs      Outputs                     Physical Elements
I:1/0 = 1   O:2/0 = O:2/1 = O:2/2 = 1  Activate Clamper, ON Motor, Energize Motor Down
I:1/1 = 1   O:2/0 = 0, O:2/2 = 0       Deactivate Clamper, De-Energize Motor Down
I:1/1 = 1   O:2/3 = 1                   Energize Motor Up Coil
I:1/2 = 1   O:2/3 = 0                   De-Energize Motor Up Coil, OFF Motor
```

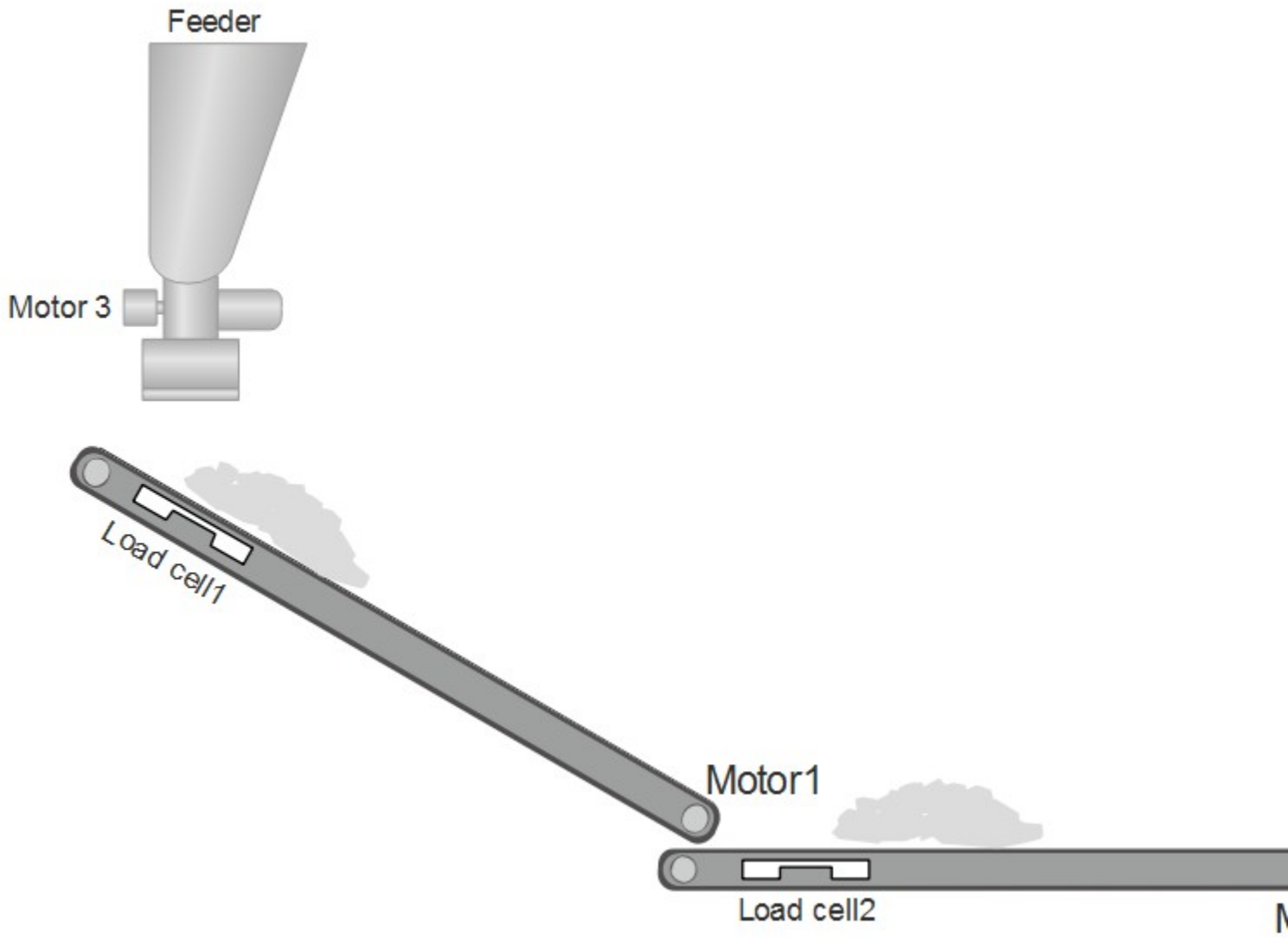
# PLC Program to Control the Sequence of Conveyors and Interlocking Them

This is a PLC Program to Control the Sequence of Conveyors and Interlocking Them.

## Problem Description

A feeder drops material on the conveyor which sends material for further process through one more conveyor. Conveyor must start automatically when material is dropped on it. Implement automation of this in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- Feeder has a motor mounted to feed material on conveyor belts.
- Load cells are installed at the bottom of conveyor belts to detect if material is present on the conveyor belt.
- When material falls on conveyor belt 1, motor 1 should start, and when material in present on conveyor belt 2, motor 2 remain On.
- Switches can also be used sometimes to detect material's presence. But for more reliable operation, Load cells can be used as shown in the diagram above.

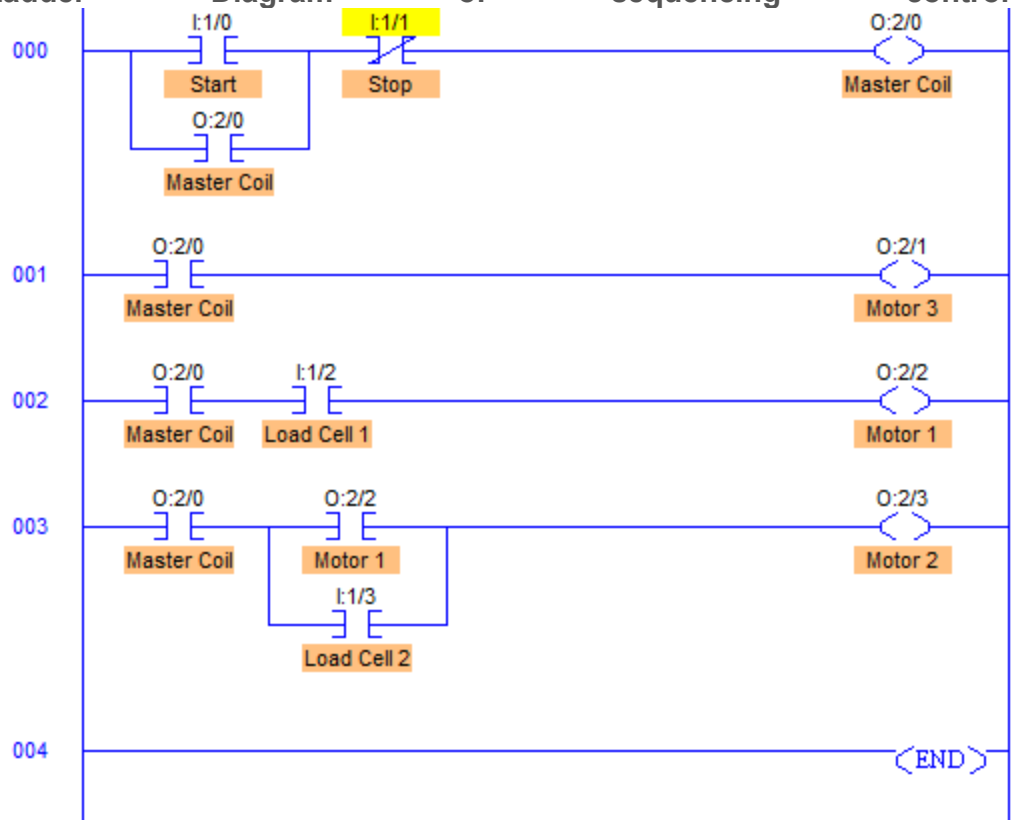
## PLC Program

Here is PLC program to Control the Sequence of Conveyors and Interlocking Them, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start                  (Input)
 I:1/1   = Stop                           (Input)
 I:1/2   = Load cell of conveyor 1        (Input)
 I:1/3   = Load cell of conveyor 2        (Input)
 O:2/0   = Latching Coil                  (Output)
 O:2/1   = Motor 3 (feeder)               (Output)
 O:2/2   = Motor 1 (Conveyor 1)           (Output)
```

```
O:2/3   = Motor 2 (Conveyor 2)           (Output)
```

**Ladder Diagram of sequencing control of conveyor**




## Program Description

- RUNG000 is for Master Start/Stop the process.
- RUNG001 is to operate feeder with output address O:2/0 which is operated when Start PB is pressed.
- RUNG002 is to operate Motor 1 of Conveyor 1 which is operated when Load cell 1 detects the presence of material. As long as material is on the conveyor, Motor 1 remains energized.
- RUNG003 is to operate Motor 2 of Conveyor 2 O:2/3 which is operated whenever Motor 1 is ON AND/OR as long as material is presence on the conveyor 2 which is detected by Load cell 2 (I:1/3).

## Runtime Test Cases

```
Inputs                    Outputs       Physical Elements
I:1/0 = 1                 O:2/1 = 1    Start Motor 3
I:1/0 = 1, I:1/2 = 1      O:2/2 = 1    Start Motor 1
I:1/0 = 1, O:2/2 = 1      O:2/3 = 1    Start Motor 2
I:1/0 = 1, I:1/3 = 1      O:2/3 = 1    Start Motor 2
```

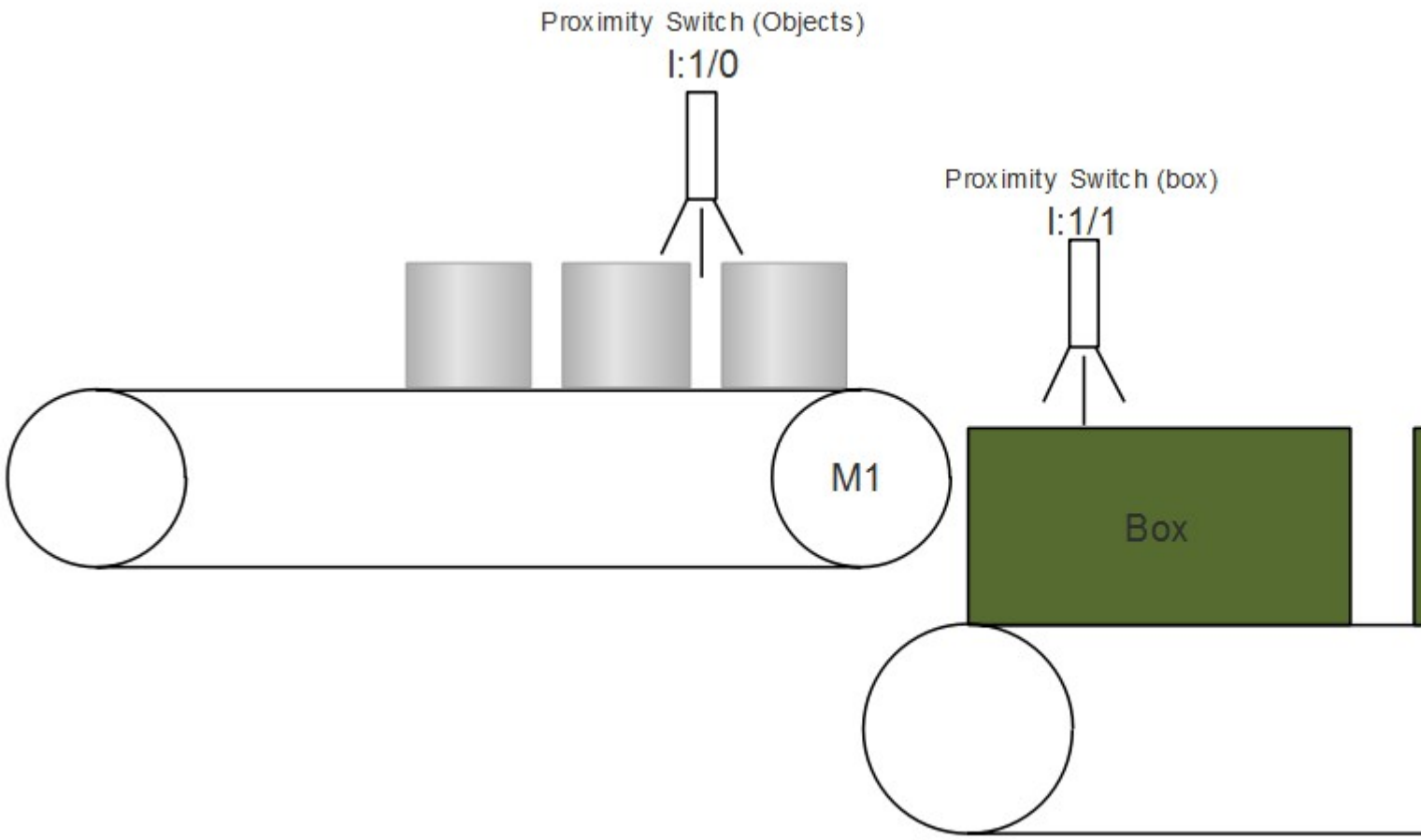
# PLC Program to Count and Pack Parts from Conveyor

This is a PLC Program to Count and Pack Parts from Conveyor.

## Problem Description

Objects are moving on a conveyor belt 1. When an empty box is detected, conveyor belt starts and 5pcs are packed in a box. When box is filled, it is carried to the storage area via conveyor belt 2. Implement automation of this process in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- Use proximity switches to detect moving objects on the conveyor belt 1 and to detect an empty box on conveyor belt 2.
- Use counter to count number of objects to be packed.
- Use timer such that when 5pcs are detected, conveyor runs for a while and stops when 5th object is finally collected in the box. Assume time by calculating conveyor belt speed.
- When number of parts to be packed are detected timer is activated. When timer is over, it stops the conveyor until next empty box is detected.
- Assuming time taken by the last 5th object is 2secs to be collected.

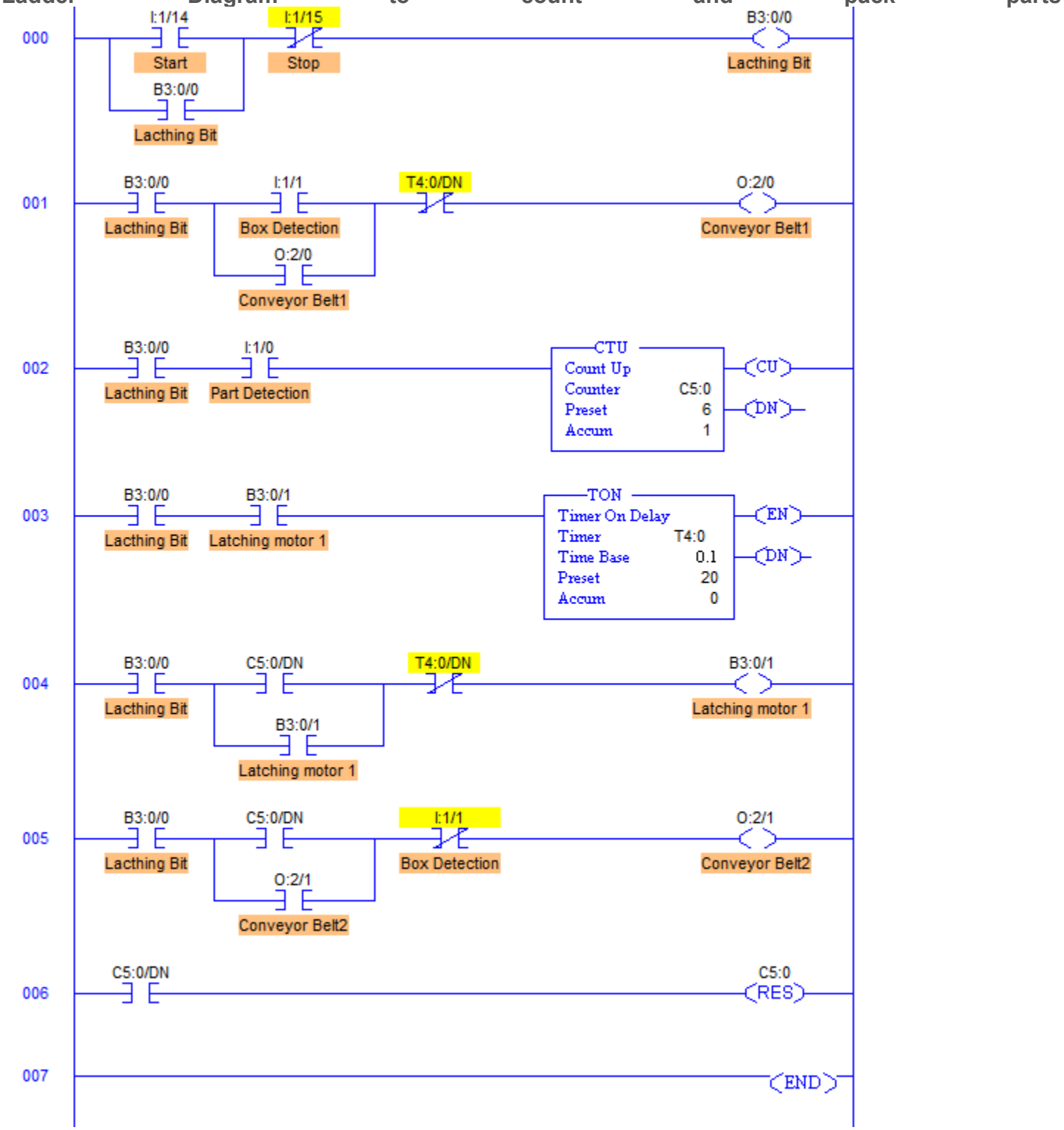
## PLC Program

Here is PLC program to Count and Pack Parts from Conveyor, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                 (Input)
 I:1/15  = Stop                          (Input)
 B3:0/0  = Latching Bit                  (Bit)
 B3:0/1  = Latching Motor 1 bit          (Bit)
 I:1/0   = Part detection                (Input)
 I:1:1   = Box detection         (Input)
 O:2/0   = Conveyor Belt 1               (Output)
```

```
O:2/1   = Conveyo2 Belt 2                (Output)
C5:0    = Part counter                   (Counter)
T4:0    = Timer to stop conveyor (Timer)
```

**Ladder Diagram to count and pack parts**




## Program Description

- RUNG000 is for master start and stop the process.

- RUNG001 is for controlling Conveyor Belt 1 motor M1 which is operated by Box Detection proximity switch. When the process is started and empty box is detected by Proximity having address of I:1/1, power supply to the motor of Conveyor Belt 1 is given. This drives the Motor M1 moving objects on the conveyor belt 1.
- Every time a part is detected by the I:1/1 Part detector, Counter C5:0 is incremented by 1. Counter has a preset value of 6. Here that means when total 5 parts are detected, counter done bit C5:0/DN is set to 1.
- When C5:0/DN bit is set, timer T4:0 is activated for 2secs which is an assumption that the part will take approximately 2secs to fall into the box after it is detected by the Pat Detector proximity switch I:1/0.
- Counter is immediately reset when C5:0/DN goes true. So one shot is generated by C5:0/DN bit. It latches the Latching Motor1 bit before it goes false.
- Conveyor Belt 2 motor M2 (O:2/1) is operated when counter value reaches to 6 that is when total 5 parts are detected. And it is stopped when an empty box is detected again.
- And the process is repeated.
- Important thing to note here is that Counter Preset is set 6 that is because when the counter is reset, the value is the accumulator is set to 1 because of latching bit.

## Runtime Test Cases

```
Inputs                  Output                  Physical Elements
I:1/1 = 1               O:2/0 = 1, O:2/1 = 0  Start Conveyor1, Stop Conveyor2
I:1/0 = 1(Momentarily)   C5:0.ACC = +1          Part is detected, Increment Counter
C5:0.ACC = 6            T4:0.EN = 1             Enable Timer T4:0, Reset Counter
T4:0/DN = 1             O:2/1 = 1       Start Conveyor2, Stop Conveyor1
```

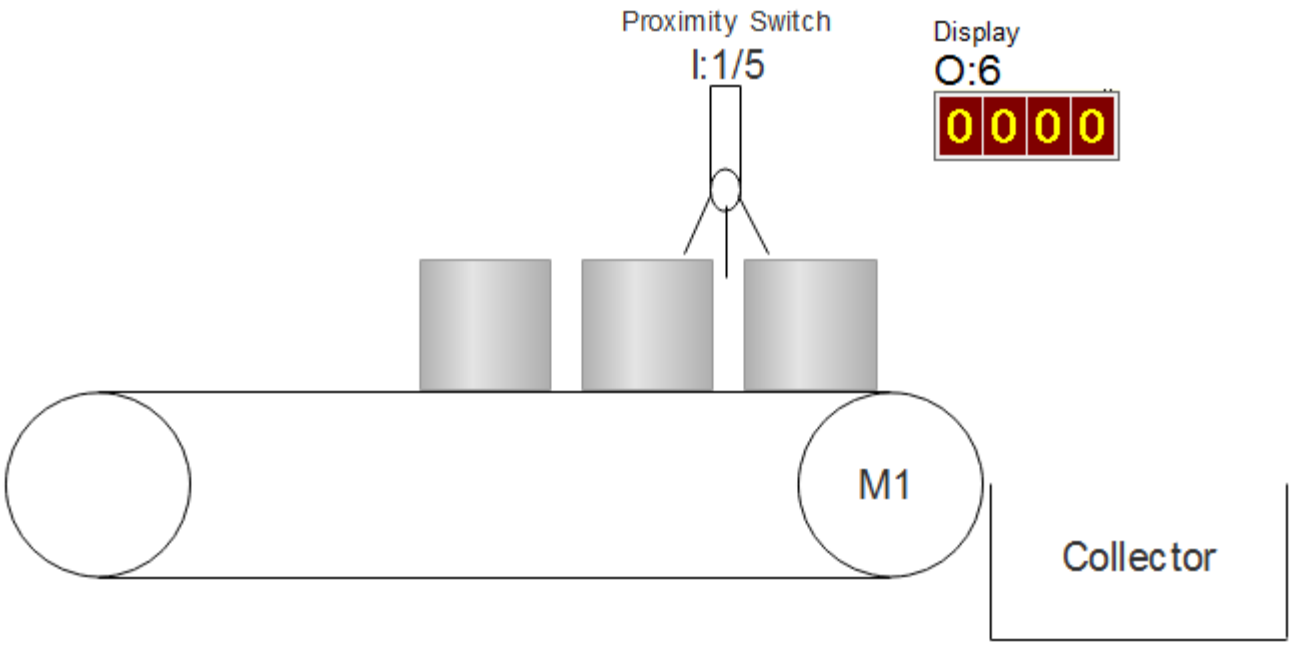
# PLC Program for Counting of Parts from Conveyor

This is a PLC Program for Counting of Parts from Conveyor.

## Problem Description

Parts are moved on the conveyor. Count the number of parts collected at the end of the conveyor and display it on the display in PLC using Ladder Diagram programming language.

## Problem Diagram



## Problem Solution

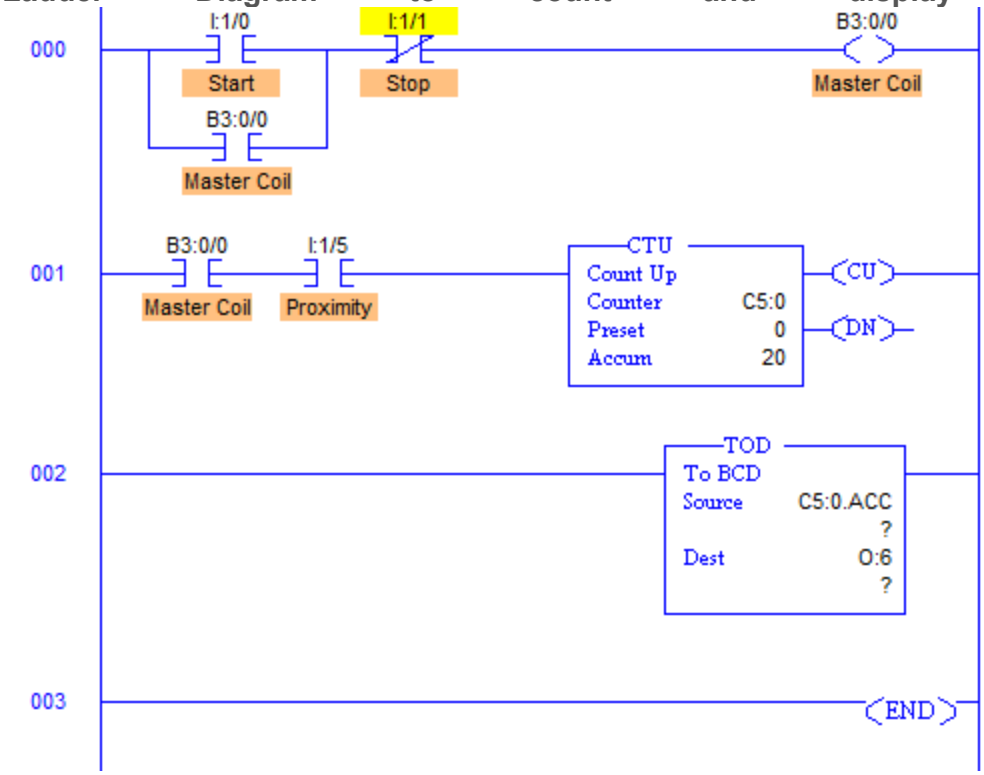
- Mount Proximity Switch to detect the parts.
- Use output of proximity to counter as an input to increment data.
- Convert this number into appropriate numerical and show number of parts collected.
- Most widely Inductive and Capacitive Proximity switches are used to detect parts.
- Inductive Proximity are used to detect metal objects while to detect other objects, Capacitive Proximity Switch is most widely used.
- Capacitive Proximity detection capability ranges from 1 to 25mm distance.
- Mount this sensor according to the size of parts present on the conveyor and width of conveyor so that this sensor can detect parts easily.
- CUP is used to increment the number of parts collected.

## PLC Program

Here is PLC program for Counting of Parts from Conveyor, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start                                          (Input)
 I:1/1   = Stop                                           (Input)
 I:1/5   = Proximity (Part detection)                     (Input)
 C5:0    = Counter Up                                     (Counter)
 O:6     = Display address                                (Output)
 TOD     = Hexadecimal to BCD conversion instruction      (Compute)
 B3:0/0  = Latching Bit                                   (Bit)
```

**Ladder Diagram to count and display number of parts**




## Program Description

- RUNG000 is the master start/stop rung used to start and stop the counting process.

- RUNG001 is to count the number parts collected. Counter Up with address C5:0 is to count the number of parts collected. Whenever a part is detected by Proximity Switch, I:1/5 goes true momentarily and it increments accumulator value of the counter.
- RUNG002 is used to convert Decimal numbers into Binary Coded Decimal. Source address is given as C5:0.ACC and destination address is given as O:6 which is of Display. Converted BCD number is decoded by the display and the value present in accumulator of CUP C5:0 is displayed.

## Runtime Test Cases

```
Inputs                   Outputs          Elements
I:1/5 = 1 (Momentarily)  Counter Incremented      Counter
C5:0.ACC = 0             O:6 = 0                 Display = 0
C5:0.ACC = 10            O:6 = 16         Display = 10
C5:0.ACC = 25            O:6 = 37         Display = 25
```

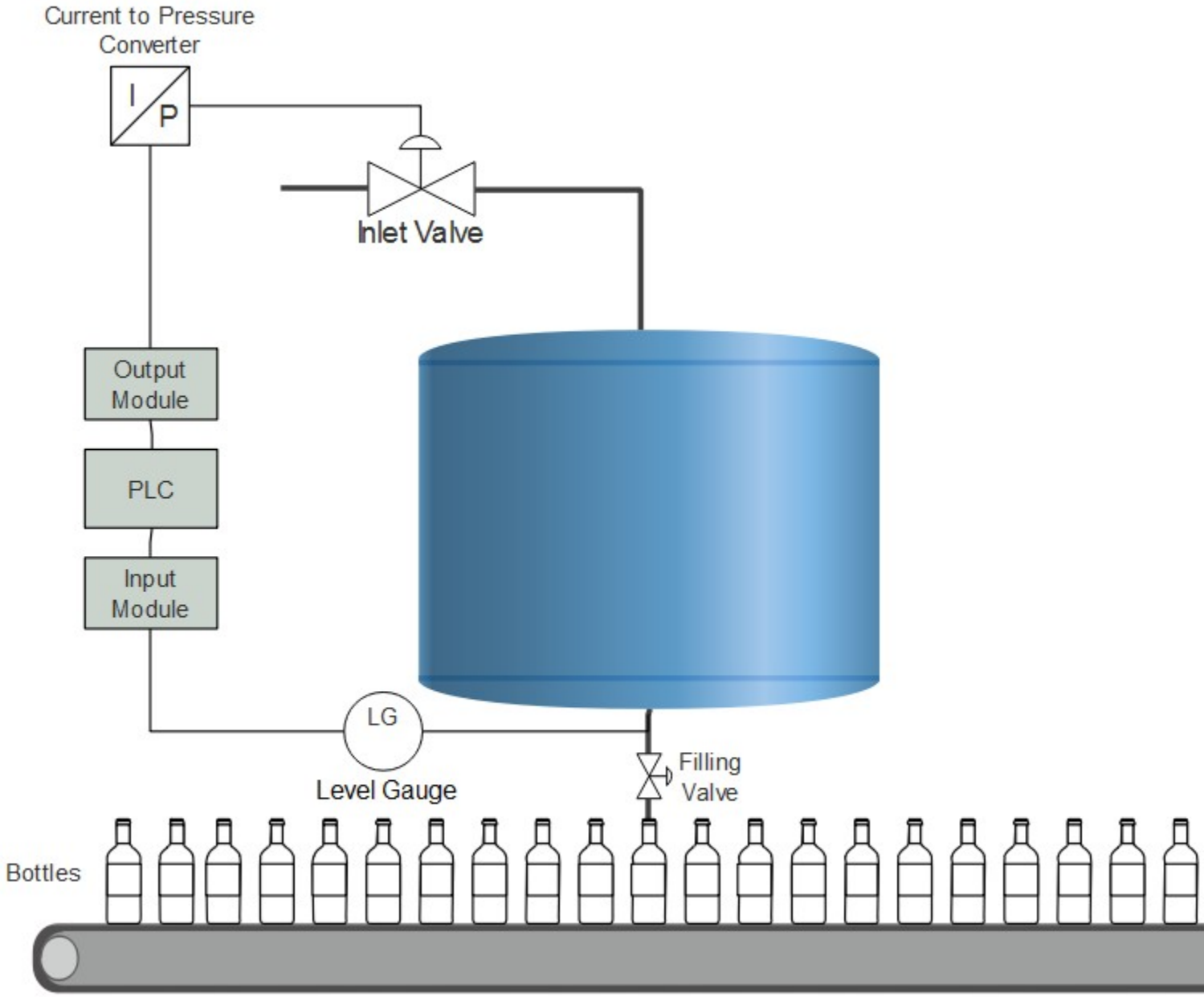
# PLC Program to Maintain the Pressure Head in a Bottle Filling System

This is a PLC Program to Maintain the Pressure Head in a Bottle Filling System.

## Problem Description

Bottle filling has a constant speed of filling 20 bottles per minute. This speed depends on level of the tank due to its head pressure. To maintain this speed, pressure head of the filling tank has to be maintained at a particular. Implement this automation in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- Solid state level switches cannot be used here since level has to be continuously monitored.
- Pressure is proportional to level. As level of a tank increases, pressure also increases.
- Level gauges are highly sensitive to very small variations. Many companies such as Rockwall Automation and Endress+Hauser manufacture pressure gauges to measure liquid level of a tank.
- Output of this gauge is terms of pressure so we have to convert pressure into equivalent current output. But let us assume here that maximum pressure that means when tank is full, it gives 20mA output and when tank is empty, it gives 4mA output which is in standard form 4-20mA.
- Use conversion instructions to convert this 4-20mA data into registers. To do this, Analog modules for PLCs are used.
- These modules convert 4-20mA into equivalent digital level signals.
- Output of this Analog modules are stored in Hex form which are then processed by the processor and hex output is generated again.

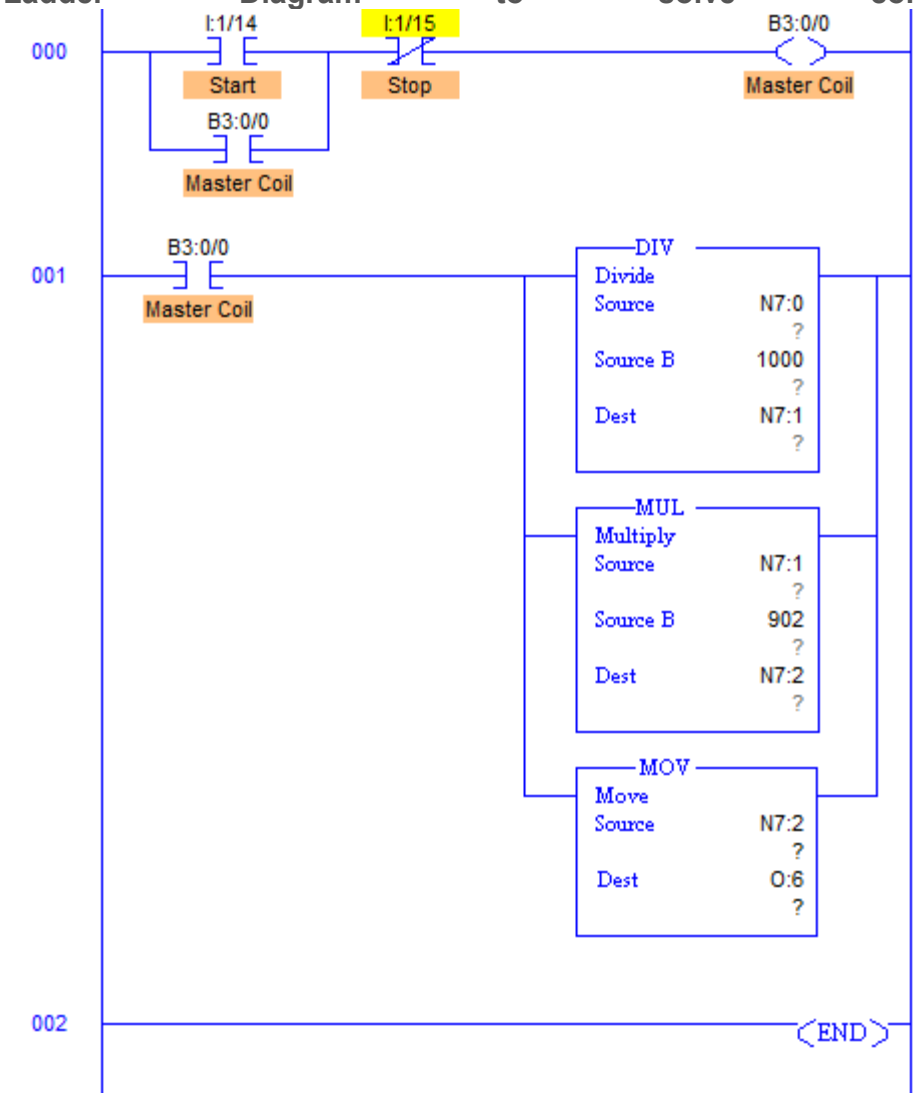
- Just like Analog input modules, Analog output modules are used convert digital output data into equivalent current signals to operate power supply circuit which varies output accordingly, to drive the final control element, here control valve.

## PLC Program

Here is PLC program to Maintain the Pressure Head in a Bottle Filling System, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                                                  (Input)
 I:1/15  = Stop                                                   (Input)
 B3:0/0  = Master Coil Bit                                        (Bit)
 DIV     = To divide total height of the tank                     (Compute)
 MUL     = To multiply with the tank level to be maintained       (Compute)
 N7:0    = Input from Level Gauge                          (Register)
 N7:1    = Result (Variation per cm)                              (Register)
 N7:2    = Result of multiplication                               (Register)
 O:6     = I-P Converter                                          (Output)
```

**Ladder Diagram to solve control pressure head**




## Program Description

- RUNG000 is a latching rung to operate the system through Master Start and Stop PB.

- RUNG001 comprises all the conversion needed to control level of the tank.
- Output of transmitter is in current signals which is 4-20mA.
- When output is 4mA, Analog Input Module converts it into 16bit equivalent hex numbers. Hence when input to Analog module is 4mA, it stores 0000h into register and when 20mA, it stores FFFFh into register. Here register N7:0. This conversion is done internally by the A-to-D converter in Analog Input Module.
- Height of the tank is 10m or 1000cm. By converting it into equivalent hex, change in value per centimeter is 66 approximately which is stored in register N7:1.
- Value of N7:1 is then multiplied with the preset value of tank level that is 900cm here.
- This multiplication is stored into N7:2 register. Digital to Analog conversion of value stored in N7:2 is performed inside the processor and equivalent mA current is received from terminal O:6.
- Current to Pneumatic converter then converts current signals into equivalent 3-15psi pneumatic signal and adjusts valve opening.

## Runtime Test Cases

```
Input           Output          Valve percentage open
N7:0 = 0000h    O:6 = 0000h     Valve 100% Open
N7:0 = FFFFh    O:6 = E506h     Valve 90% Open
```

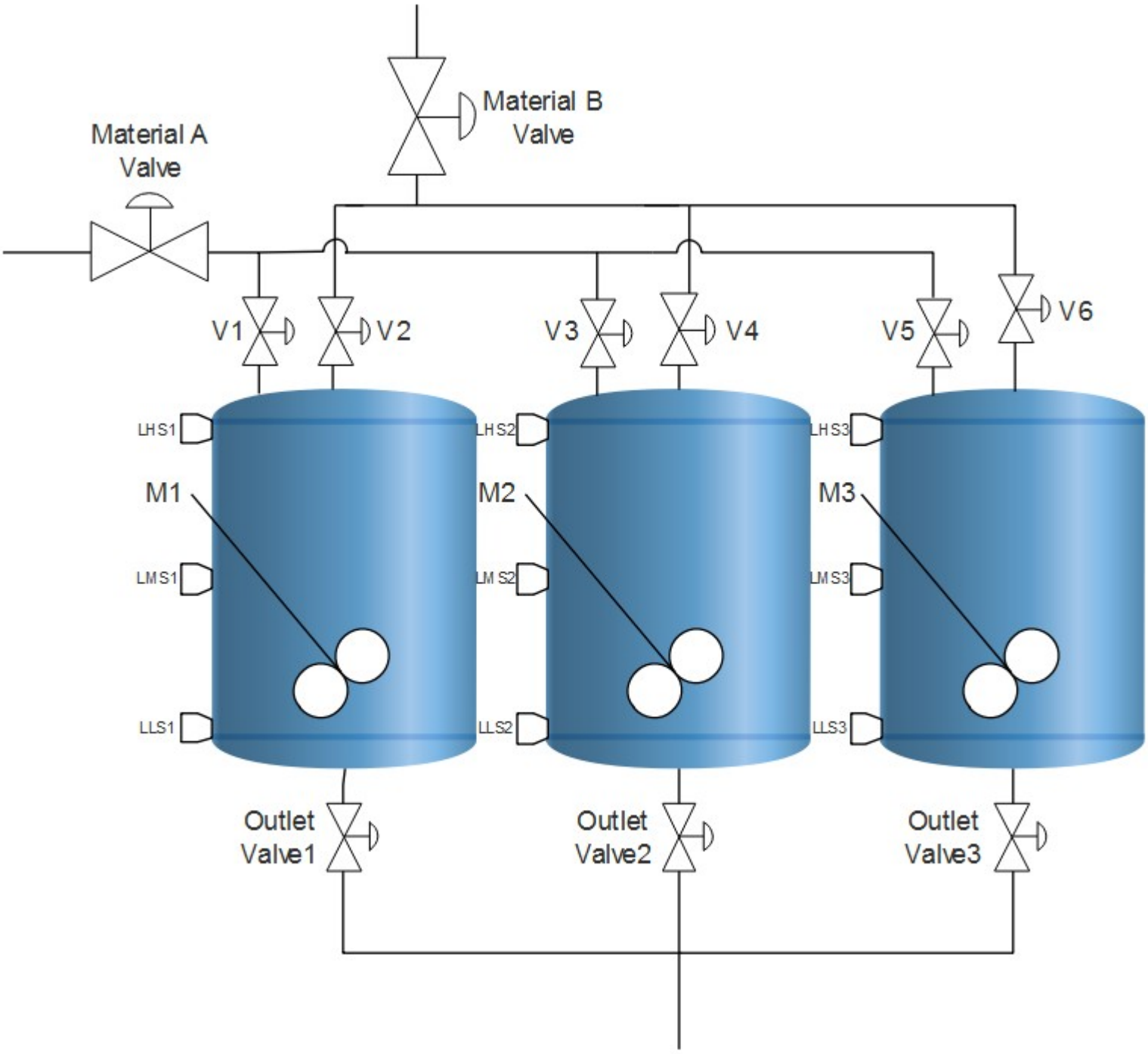
# PLC Program to Control Three Mixing Devices in a Processing Line

This is a PLC Program to Control Three Mixing Devices in a Processing Line.

## Problem Description

Control three mixing devices in a process line. Two mixing devices are running in the parallel to each other. When any one of the mixing process stops, individual stop switches turn the third mixing device ON and the device operated by the stop button is stopped. Implement automation of this system in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

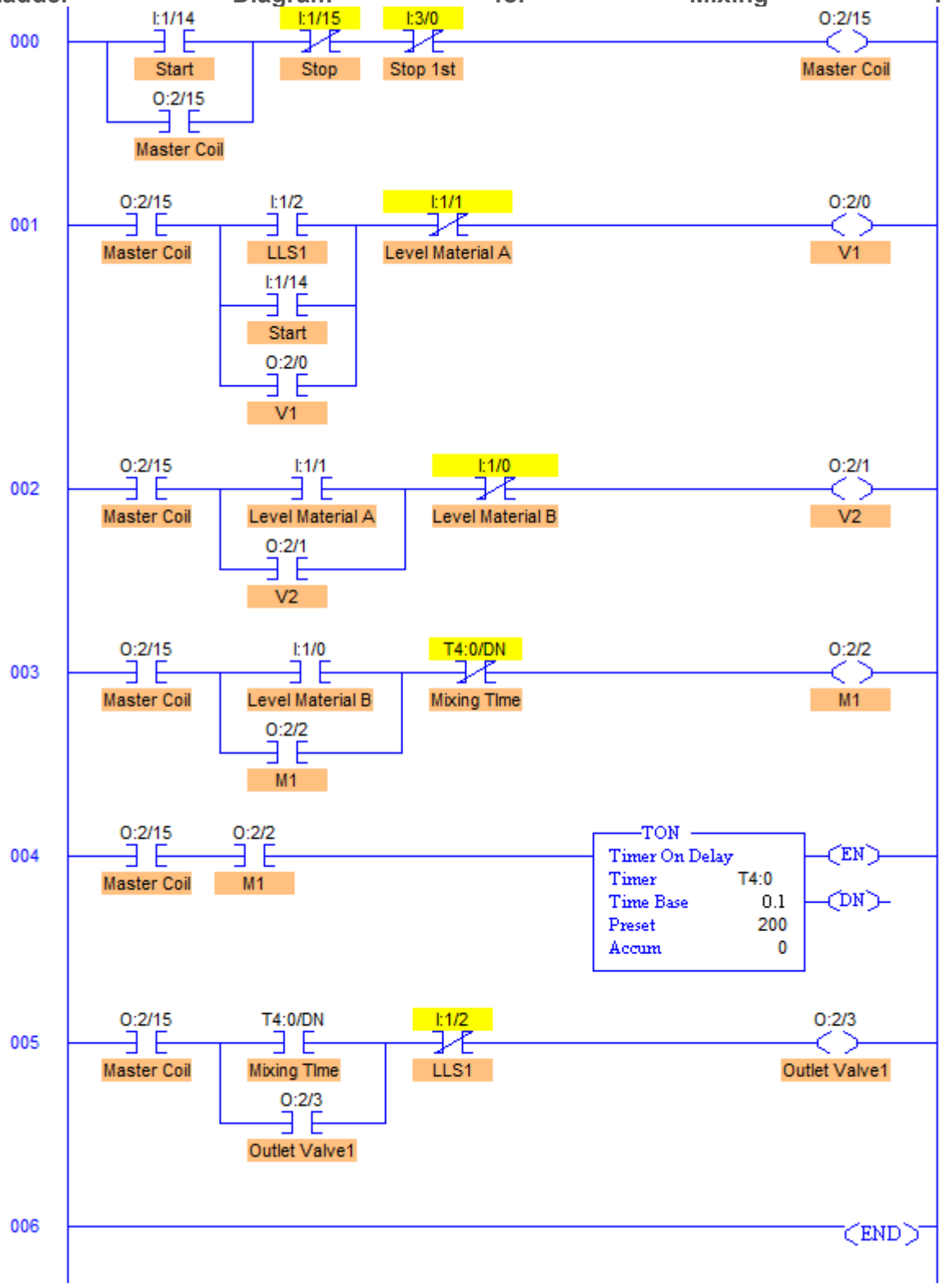
- Use limit switches, control valves and motor agitators separately for all three mixing tanks.
- The only common input to all these tanks is Material feeding by two valves.
- Use separate timers for each tank for mixing.
- First two tanks are tagged as primary mixing tanks and the third one is said to be secondary mixing tank.
- Use switches of stop buttons of both Primary mixing processes as an input to start the tank third to enter into the process sequence.

## PLC Program

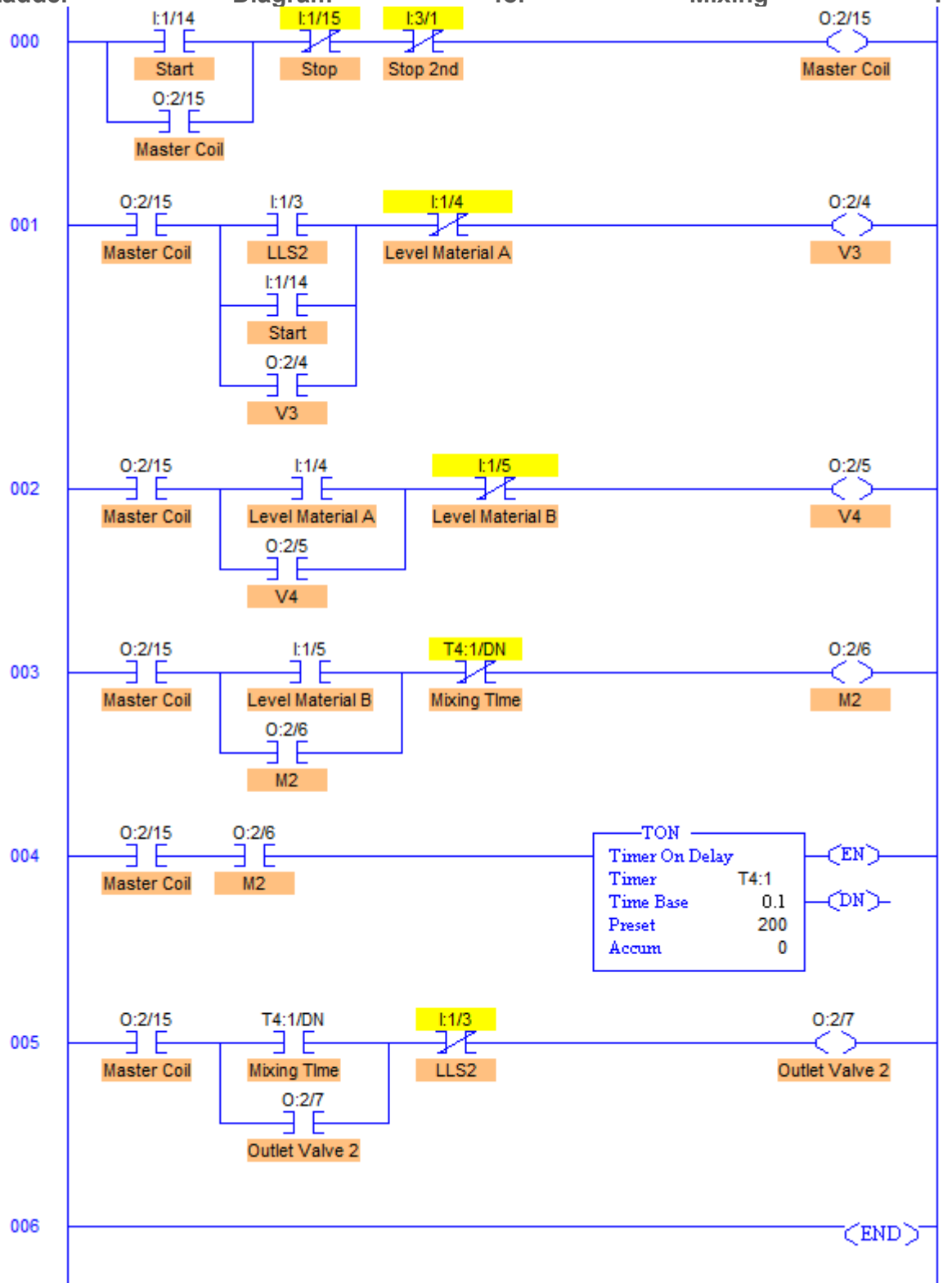
Here is PLC program to Control Three Mixing Devices in a Processing Line, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14                  = Start                           (Input)
 I:1/15                  = Stop                                    (Input)
 O:2/15                  = Master Coil                             (Output)
 I:1/0, I:1/5, I:1/7     = Level of Material B Switches            (Input)
 I:1/1, I:1/4, I:1/8     = Level of Material A switches            (Input)
 I:1/2, I:1/3, I:1/6     = Low Level Switch (empty tank)           (Input)
 O:2/0, O:2/4, O:2/8     = Inlet Valves to feed Material A (Output)
 O:2/1, O:2/5, O:2/10    = Inlet Valves to feed Material B (Output)
 O:2/2, O:2/6, O:2/9     = Agitator Motors M1, M2 and M3          (Output)
 O:2/3, O:2/7, O:2/11    = Outlet Valves for product outlet        (Output)
 T4:0, T4:1, T4:2        = Timers to mix materials                 (Timer)
```

# Ladder Diagram for Mixing Tank 1



**Ladder Diagram for Mixing Tank 2**

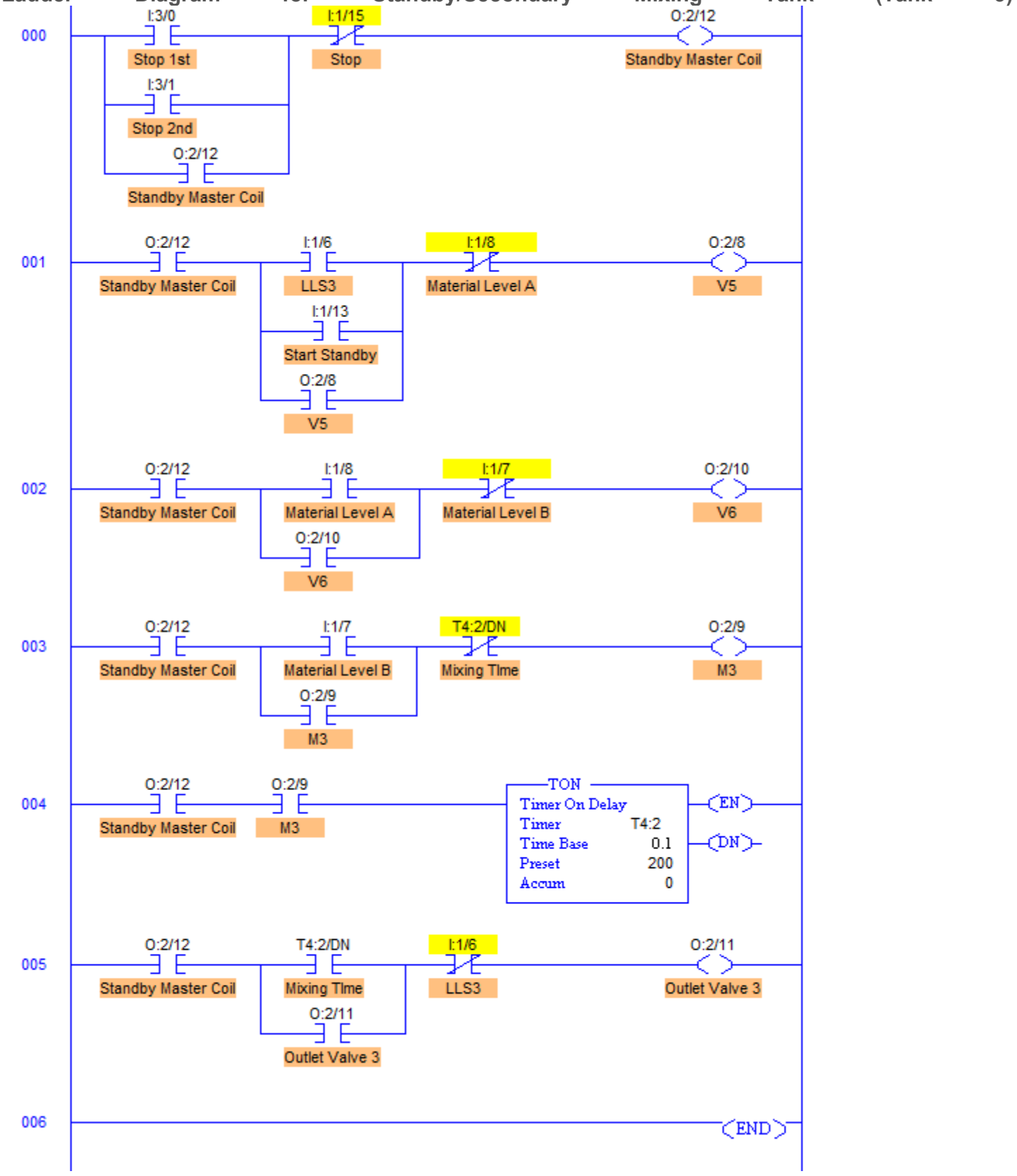



## Program Description

- RUNG000 contain master start/stop with address of Start PB I:1/14 and Stop PB I:1/15.
- RUNG001 is to operate V1 of Material A with address O:2/0. It is operated when Low Level of the tank 1 is detected by Level Low Switch1 I:1/2. And it is closed when Level Material A of tank 1 is detected by a switch with address I:1/1. Start PB is also connected in parallel with the LLS1, it is done so that when LLS1 or level of Material A is not detected, Inlet Valve is operated by Start PB.

- RUNG002 is to operate V2 of Material B with address O:2/1. It is opened when Material A is filled to its desired level (Level Material A). In other words, V2 of Material B is opened when Level of Material A is detected by I:1/1 and it is closed when Level of Material B of tank 1 is detected or Tank 1 is full.

**Ladder Diagram for Standby/Secondary Mixing Tank (Tank 3)**



- RUNG003 & RUNG004 operates Agitator Motor M1 with address O:2/2. When the tank is full with Material A and B, Level High (Level Material B) of tank 1 is detected. This detection energizes O:2/2 and enables ON timer with address T4:0. Agitator Motor M1 is connected to the address O:2/2. So when O:2/2 energizes, Motor Agitator M1 starts the mixing process and mixes Material A and B for 20secs which is a preset of Timer ON. When Pre = Acc, T4;0/DN bit goes high turning off the agitator motor M1.
- RUNG005 is for Outlet Valve 1 with address O:2/3. It is operated when the entire mixing process is completed that is when Pre= Acc = 20secs. And this is closed when LLS is again detected.
- Similarly Mixing Process Tank 2 is operated.

**Extra Inputs and Outputs for Standby Process**

```
I:3/0  = Individual Stop for tank 1 (Start Tank 3)       (Input)
I:3/1  = Individual Stop for tank 2 (Start Tank 3)       (input)
```

- In Mixing Tank 3, when any of the Primary tanks Tank1 or Tank2 fails, operation of mixing tank 3 takes place.
- Input of this Mixing tank 3 is Individual stop buttons of Tank 1 and Tank 2 which has to be pressed in case of failure in either of the primary mixing tanks.

## Runtime Test Cases

```
Inputs                  Outputs                 Physical Elements
I:1/2 = 1               O:2/0 = 1               Open V1
I:1/2 = 0 & I:1/1 = 0   O:2/0 = 1               Open V1
I:1/1 = 1       O:2/0 = 0       Close V1
I:1/1 = 1               O:2/1 = 1       Open V2
I:1/0 = 1               O:2/1 = 0               Close V2
I:1/0 = 1       O:2/2 = 1               Run Agitator Motor M1
T4:0/DN = 1             O:2/2 = 0       Stop Agitator Motor M1
T4:0/DN = 1     O:2/3 = 1               Open Outlet Valve1
I:1/2 = 1       O:2/3 = 0       Close Outlet Valve1

Similarly for Mixing Tank 2
I:3/0 = 1               O:2/0 = 0, O:2/8 = 1    Start Process Tank 3, Open V5
I:3/1 = 1               O:2/4 = 0, O:2/8 = 1    Start Process Tank 3, Open V5
```

# PLC Program to Control Mixing in a Tank

This is a PLC Program to Control Mixing in a Tank.

## Problem Description

Material A and Material B are collected in a tank. These materials are mixed for a while. Mixed product is then drained out through Outlet valve. Implement this in PLC using Ladder Logic programming language.

## Problem Diagram

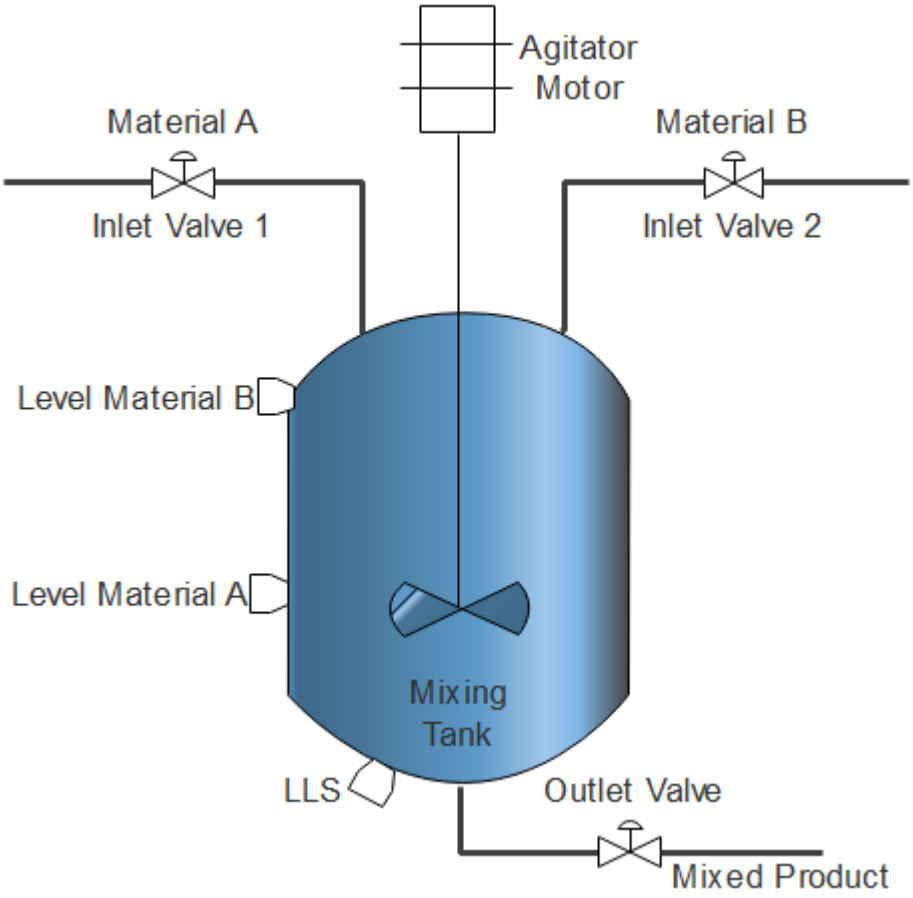



**Diagram of a mixing tank**

## Problem Solution

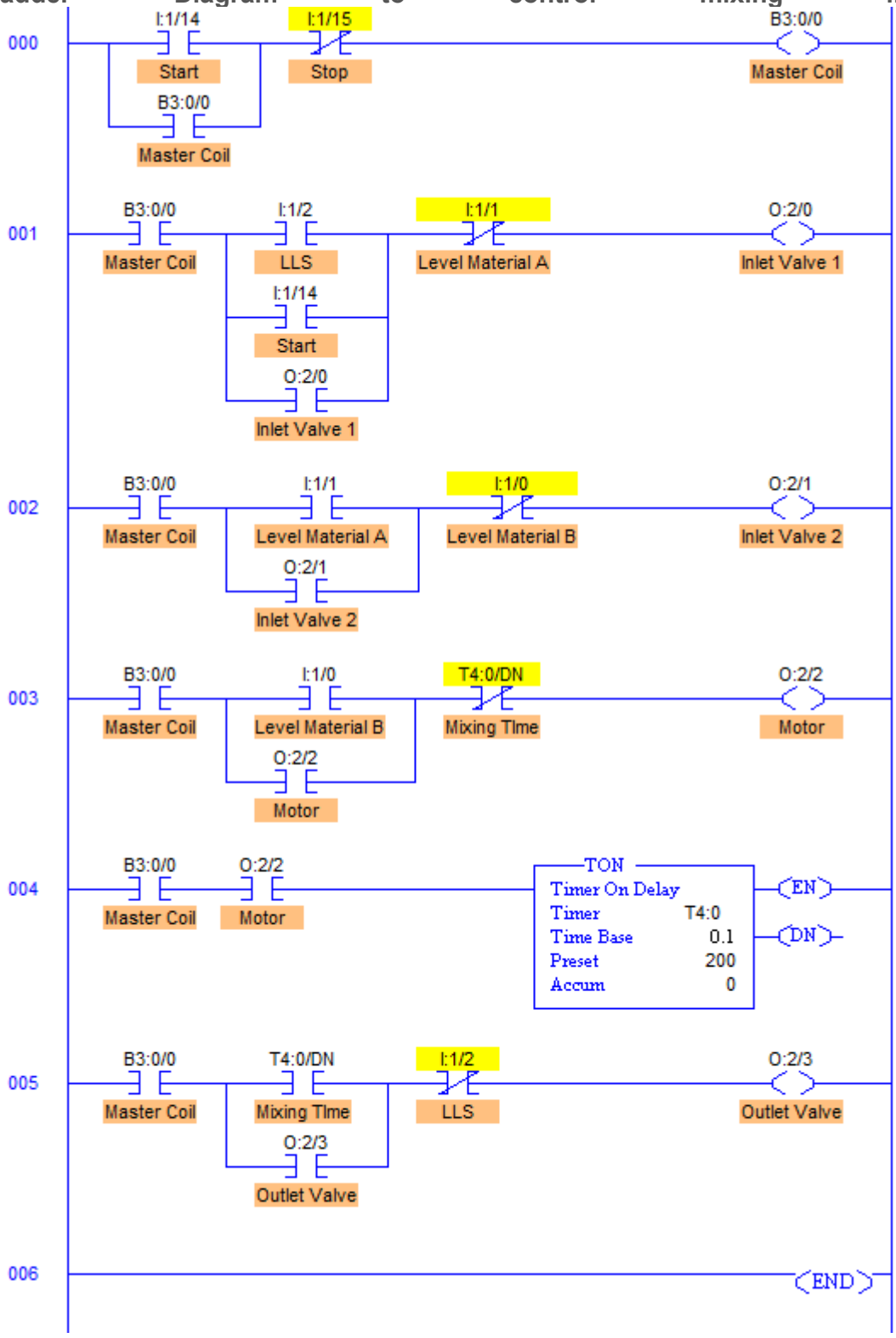
- To detect level of Material A and Material B, two separate level switches are used.
- And to detect low level, one more level switch is used at the bottom of the tank.
- These give output in digital terms that is when corresponding levels are detected.
- To control level of this system, Single Acting Piston valve can be used which has two states, either fully open or fully close.
- To control mixing, agitator is used which is connected with Motor shaft.
- Particular time delay is generate to mix the materials for a definite time.
- Control inlet valves on the basis of Level Material switches A and B.
- Outlet valve is then operated to drain the mixed product.

## PLC Program

Here is PLC program to Control Mixing in a Tank, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14  = Start                          (Input)
 I:1/15  = Stop                                  (Input)
 B3:0/0  = Master Coil Bit                       (Bit)
 I:1/0   = Level of Material B                   (Input)
 I:1/1   = Level of Material A                   (Input)
 I:1/2   = Low Level Switch (detects empty tank)   (Input)
 O:2/0   = Inlet Valve 1 (Material A Feed)(Output)
 O:2/1   = Inlet Valve 2 (Material B Feed)(Output)
 O:2/2   = Agitator Motor (Mixing)               (Output)
 O:2/3   = Outlet Valve (Product Outlet)         (Output)
 T4:0    = Time to mix Materials                 (Timer)
```

**Ladder Diagram to control mixing in the tank**




## Program Description

- RUNG000 contain master start/stop with address of Start PB I:1/14 and Stop PB I:1/15.
- RUNG001 is to operate Inlet Valve of Material A with address O:2/0. It is operated when Low Level of the tank is detected by Level Low Switch I:1/2. And it is closed when Level Material A is detected by a switch with address I:1/1. Start PB is also connected in parallel with the LLS, it is done so that when LLS or level of Material A is not detected, Inlet Valve is operated by Start PB.

- RUNG002 is to operate Inlet Valve of Material B with address O:2/1. It is opened when Material A is filled to its desired level (Level Material A). In other words, Valve of Material B is opened when Level of Material A is detected by I:1/1 and it is closed when Level of Material B is detected or Tank is full.
- RUNG003 & RUNG004 operates Agitator Motor with address O:2/2. When the tank is full with Material A and B, Level High (Level Material B) is detected. This detection energizes O:2/2 and enables ON timer with address T4:0. Agitator Motor is connected to the address O:2/2. So when O:2/2 energizes, Motor Agitator starts the mixing process and mixes Material A and B for 20secs which is a preset of Timer ON. When Pre = Acc, T4;0/DN bit goes high turning off the agitator motor.
- RUNG005 is for Outlet Valve with address O:2/3. It is operated when the entire mixing process is completed that is when Pre= Acc = 20secs. And this is closed when LLS is again detected.
- Timer On is set to auto reset mode.

## Runtime Test Cases

```
Inputs                  Outputs        Physical Elements
I:1/2 = 1               O:2/0 = 1     Open Inlet Valve 1
I:1/2 = 0 & I:1/1 = 0    O:2/0 = 1     Open Inlet Valve 1
I:1/1 = 1               O:2/0 = 0     Close Inlet Valve 1
I:1/1 = 1               O:2/1 = 1     Open Inlet valve 2
I:1/0 = 1                O:2/1 = 0     Close Inlet Valve 2
I:1/0 = 1               O:2/2 = 1     Run Agitator Motor
T4:0/DN = 1             O:2/2 = 0     Stop Agitator Motor
T4:0/DN = 1             O:2/3 = 1     Open Outlet Valve
I:1/2 = 1               O:2/3 = 0     Close Outlet Valve
```

# PLC Program for Heating Liquid in the Tank by Heater

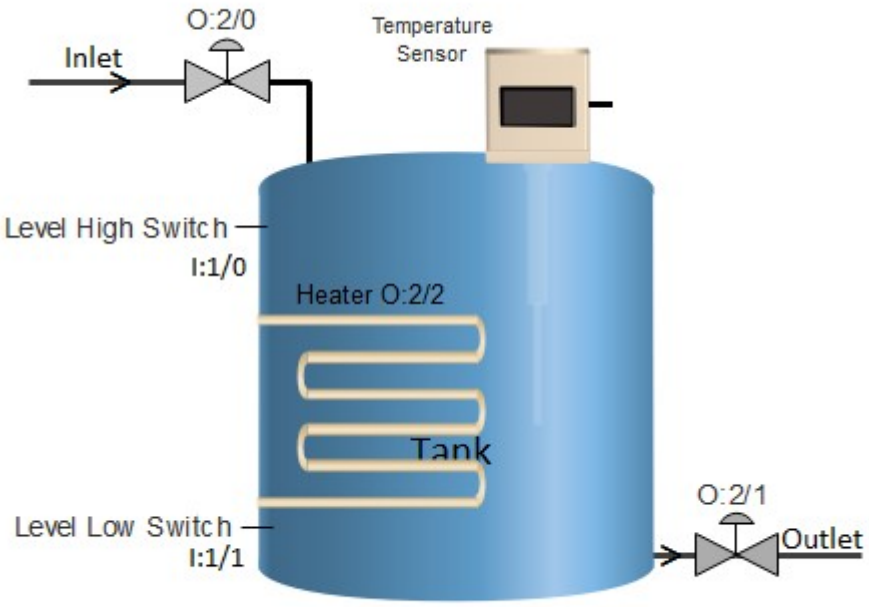
This is a PLC Program for Heating Liquid in the Tank by Heater.

## Problem Description

Controlling heating of Liquid in the tank using heater. Implement this process in PLC using Ladder Diagram programming language.

## Problem Diagram

**Diagram to control the heating of liquid in the tank**




## Problem Solution

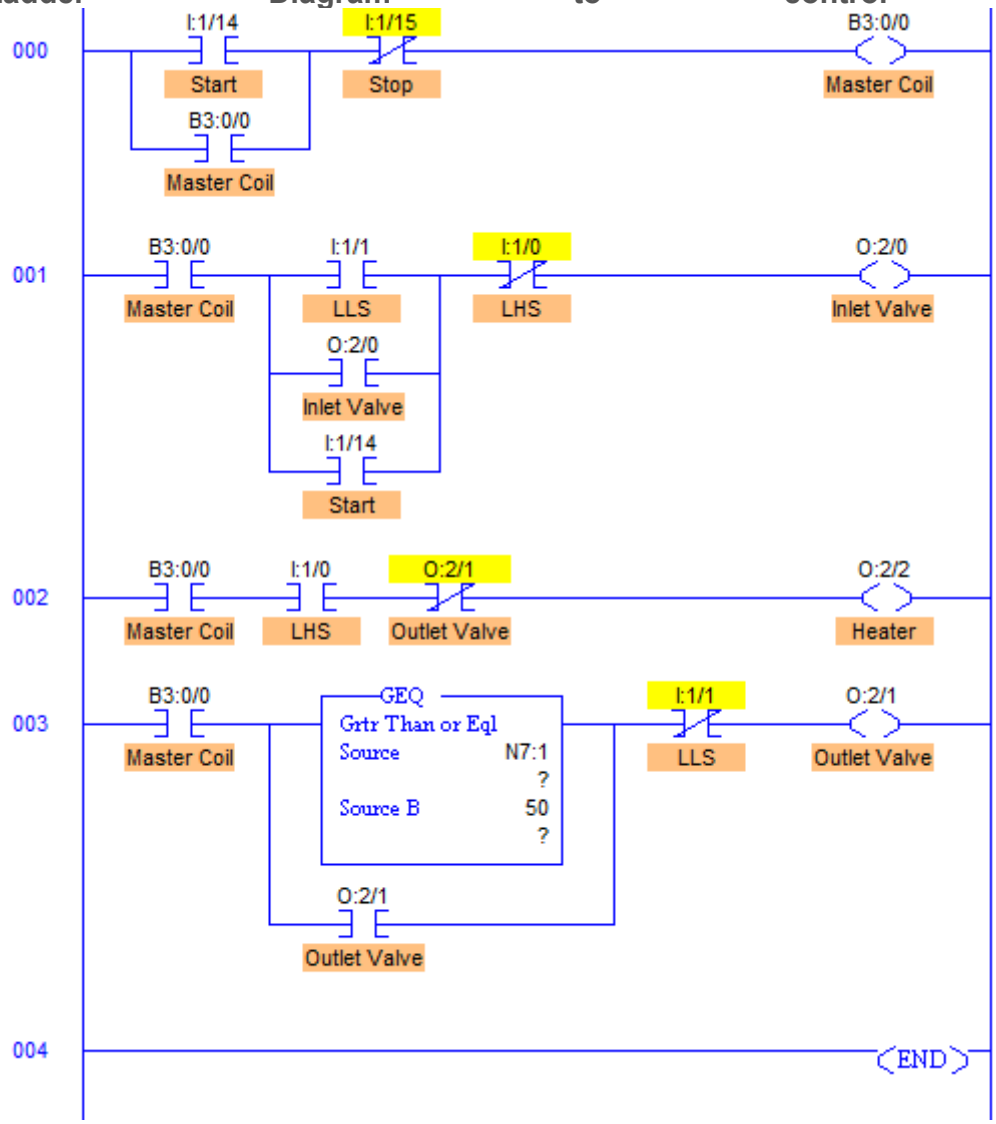
- To detect high and low level of liquid in the tank, two level switches are used which gives output in digital terms, that is when corresponding levels are detected, it gives output high otherwise remain low.
- To control level of this system, Single Acting Piston valve can be used which has two states, either fully open or fully close and to heat the liquid, heater is used.
- Low Level Switch is mounted at the bottom of the tank and Level High switch is mounted at the side-upper most position.
- Heater is installed inside the tank and temperature sensor such as RTD or Thermocouple may be used to detect the temperature of liquid in the tank.

## PLC Program

Here is PLC program for Heating Liquid in the Tank by Heater, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Level High Switch      (Input)
 I:1/1   = Level Low Switch       (Input)
 O:2/0   = Inlet Valve            (Output)
 O:2/1   = Outlet Valve           (Output)
 O:2/2   = Heater         (Output)
 I:1/14  = Start                  (Input)
 I:1/15  = Stop                   (Input)
 N7:1    = Temperature Data       (Register)
```

**Ladder Diagram to control this process**




## Program Description

- RUNG000 is simply for latching a coil and master start-stop buttons.
- RUNG003 is to control the outlet valve through O:2/1. This is done when Temperature is above 50°C. When temperature is greater than 50°C, outlet valve starts draining the liquid. XIO of Level Low Switch is connected in series so that when Level Low is detected, it goes true closing the outlet valve.
- Similarly in RUNG001, it works exactly same. It is energized when Level Low is detected. The only difference in RUNG001 is that extra I:1/14 contact in parallel with LLS.
- Suppose when the system is started and the tank is partially filled, neither LHS nor LLS is detected, in this case heater output does not get energized and heating is not controlled.
- To eliminate this error, I:1/14 (Start) is connected in parallel to LLS I:1/1 contact. This checks if LHS (I:1/0) is detected or not. If LHS is not detected, then it opens the inlet valve until LHS is detected and heating is started.
- In RUNG002, heater coil O:2/2 energizes when LHS I:1/0 is detected and heating is started. This is done until temperature reaches 50°C, O:2/1 is energized de-energizing O:2/2 which in turn stops heating.

## Runtime Test Cases

| Inputs | Outputs | Physical Elements |
|---|---|---|
| I:1/1 = 1 | O:2/0 = 1 | Open Inlet Valve, Close Outlet Valve |
| I:1/1 = 0, I:1/0 = 0 | O:2/0 = 1 | None detected, Open Inlet Valve |

```
I:1/0 = 0 & Temp. < 50°C O:2/2 = 1        Heater ON
Temp. ≥ 50                O:2/1 = 1       Open Outlet Valve
O:2/1 = 1                 O:2/2 = 0       Heater OFF
```

# PLC Program to Control Level of Series Tanks

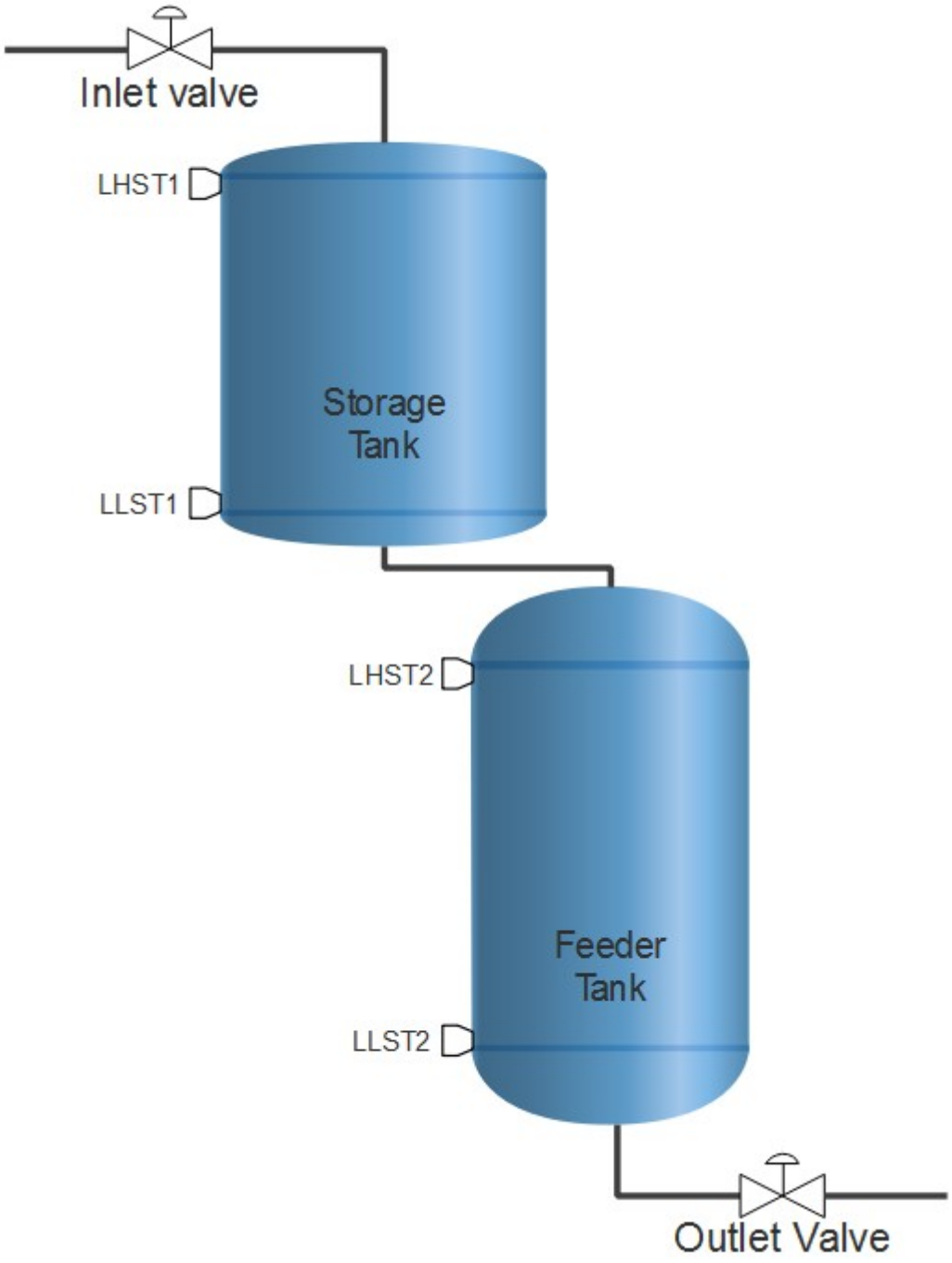
This is a PLC Program to Control Level of Series Tanks.

## Problem Description

Two tanks are connected with each other through a pipeline in series. Level of both tanks is to be controlled. Implement this operation in PLC using Ladder Diagram programming language.

## Problem Diagram

**Diagram showing tanks in series operation**




## Problem Solution

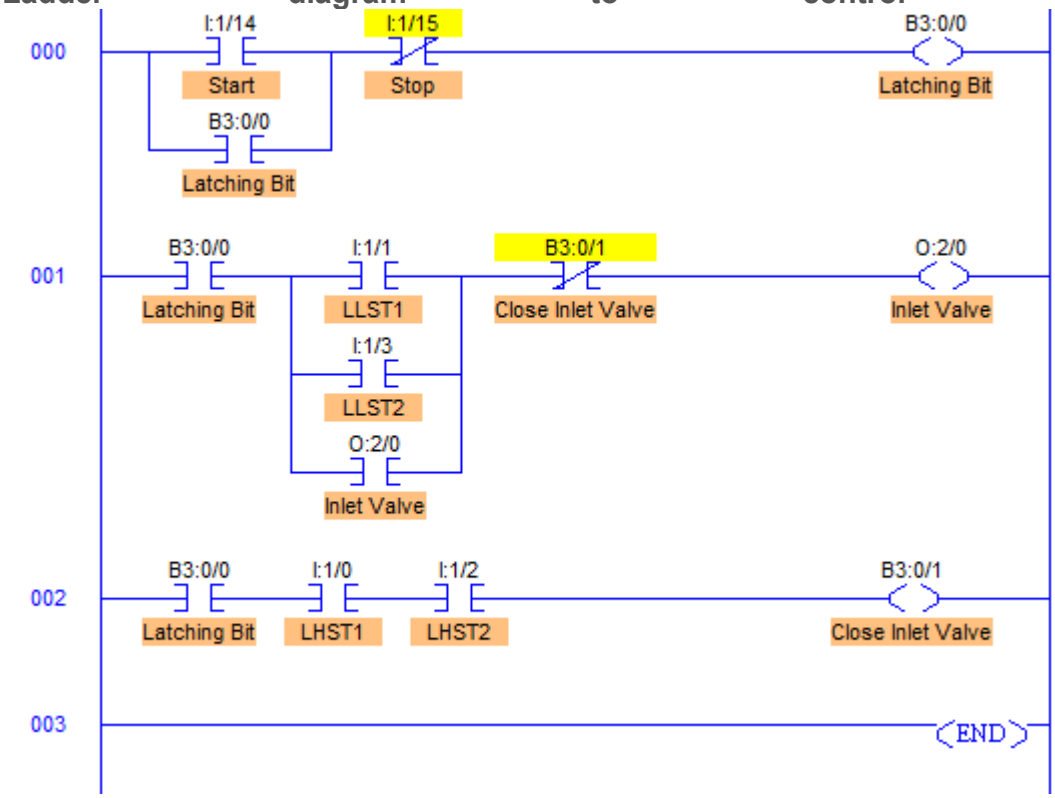
- Assume that the storage tank has more capacity than the feeder tank.To measure level of both tanks, level switches are installed.
- Outlet valve is operated by another process and it is operated according to the requirement. So outlet flow varies. Inlet valve is to be operated by PLC to control the level.
- Use interlocking to control the inlet valve.
- Connect Level Switches to the input module of PLC. Connect inlet valve with the output module of PLC.

## PLC Program

Here is PLC program to Control Level of Series Tanks, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/14 & I:1/15 = Start & Stop                                    (Input)
 I:1/0           = LHST1 high level switch of storage tank         (Input)
 I:1/1           = LLST1 low level switch of storage tank (Input)
 I:1/2           = LHST2 high level switch of feeder tank (Input)
 I:1/3           = LLST2 low level switch of feeder tank  (Input)
 O:2/0           = Inlet valve                                     (Output)
```

**Ladder diagram to control series tanks**




## Program Description

- RUNG000 is used for latching the bit B3:0/0 through which Master Start and Stop can be performed. Memory Bit is used here to save an output terminal.
- RUNG001 performs controlling of Inlet Valve which is connected to O:2/0 of output module.
- O:2/0 energizes when I:1/1 or I:1/3 goes high which is set by Low level switch of storage tank or by low level switch of feeder tank respectively.
- Similarly it is closed when high level of storage and feeder tank detected.

## Runtime Test Cases

```
Inputs                    Outputs         Physical Elements
I:1/1 = 1 or I:1/3 = 1     O:2/0 = 1      Open Inlet Valve
I:1/1 = 1 and I:1/2 = 1   O:2/0 = 0       Close Inlet Valve
```

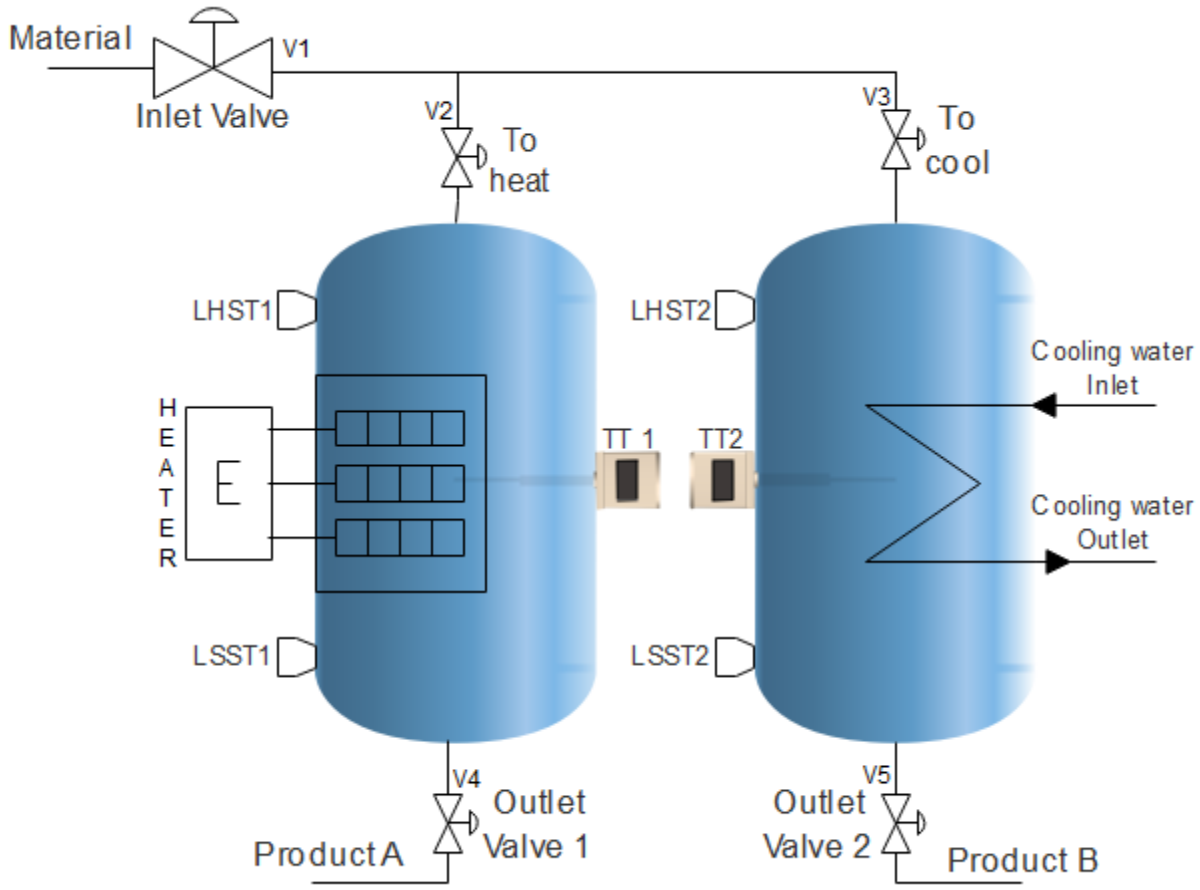
## PLC Program to Control Level of Parallel Tanks

This is a PLC Program to Control Level of Parallel Tanks.

## Problem Description

Two tanks are connected in parallel with a stream line. Heat and Cool the same material and control level of both tanks. Implement this in PLC using Ladder Diagram programming language.

## Problem Diagram




## Problem Solution

- Heater is used to heat the material and cooling water supply is used to cool down material temperature.
- Inlet valve feeds Material to both tanks.
- One tank stores material for heating purpose and the other stores material for cooling purpose.
- Level switches are used to detect Low level and High level in both the tanks.
- Outlet valves are installed in the bottom of both the tanks to drain products for further process.
- Mount temperature sensors to measure temperature of both the tanks.
- Controlling of these tanks is independent of inlet flow assuming inlet flow to be constant.

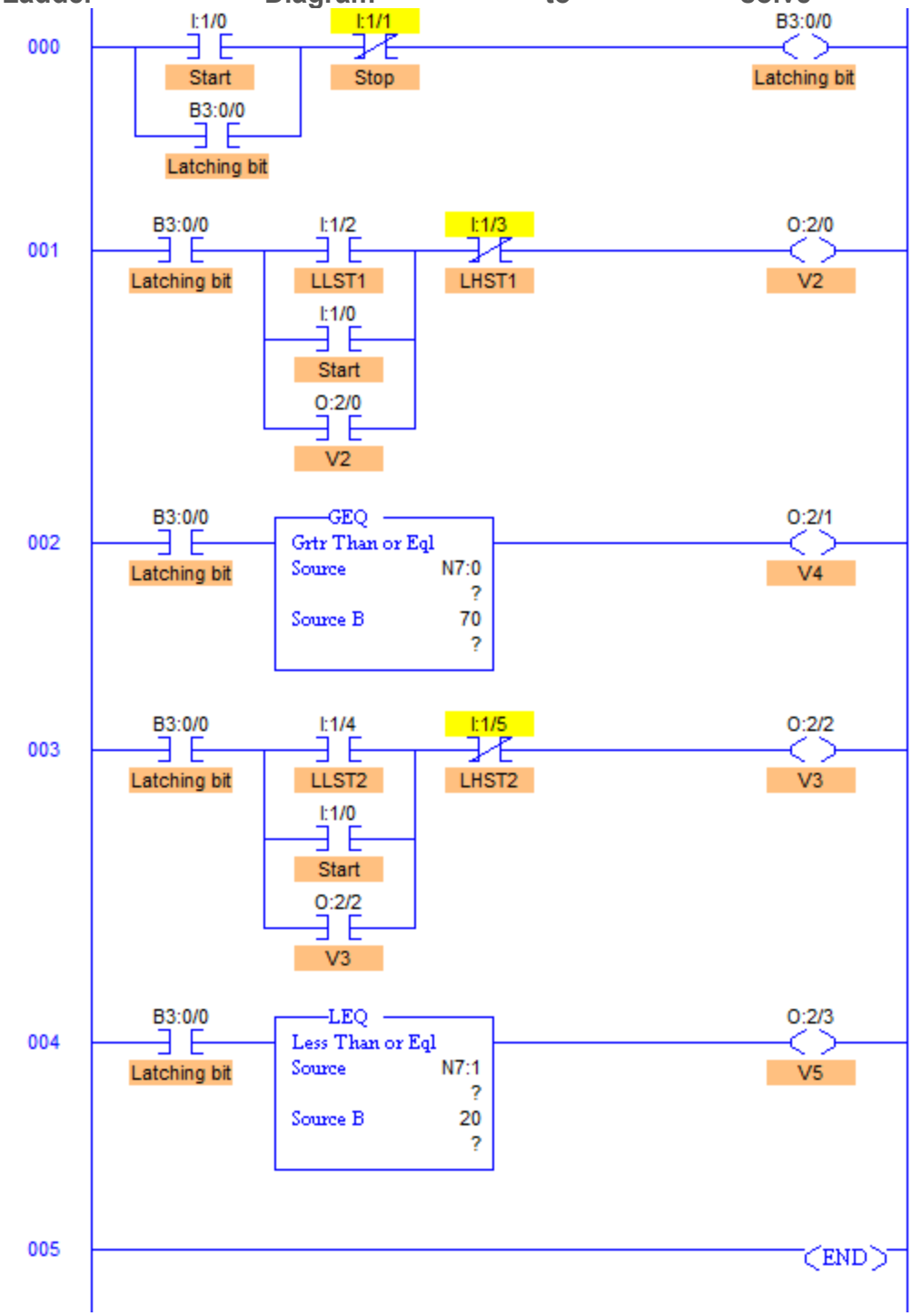
## PLC Program

Here is PLC program to Control Level of Parallel Tanks, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start PB                                                (Input)
 I:1/1   = Stop PB                                                 (Input)
 I:1/2   = LLST1 Low level of Tank 1                               (Input)
 I:1/3   = LHST1 High Level of tank 2                              (Input)
```

```
I:1/4   = LLST2 Low level of Tank 2                                  (Input)
I:1/5   = LHST2 High level of Tank 2                                 (Input)
B3:0/0  = Latching bit                                               (Bit)
O:2/0   = V2 To heating tank                                         (Output)
O:2/1   = V4 outlet from heating tank                                (Output)
O:2/2   = V3 To cooling tank                                         (Output)
O:2/3   = V5 outlet from cooling tank                                (Output)
N7:0    = Register to store temperature reading of heating tank      (Register)
N7:1    = Register to store temperature reading of cooling tank      (Register)
```

**Ladder Diagram to solve this problem**




## Program Description

- RUNG001 is to control V2 Valve. This is operated whenever Low level of Tank1 is detected or if Low level or high level both are not detected.
- RUNG002 is to control V4 valve. This is operated whenever Temperature data of tank 1 is greater than or equal to 70o C.
- Latching of V4 is not provided because it is independent of Inlet valves. Even if the tank is half empty but temperature is more than 70oC, outlet valve of the tanks open.
- Similarly RUNG003 and RUNG004 work and V3 and V5 are operated.

## Runtime Test Cases

| Input | Output | Physical Elements |
|---|---|---|
| I:1/2 = 1 | O:2/0 = 1 | Open V2 Valve |
| I:1/3 = 1 | O:2/0 = 0 | Close V2 Valve |
| N7:0 ≥ 70 | O:2/1 = 1 | Open V4 Valve |
| N7:0 < 70 | O:2/1 = 0 | Close V4 Valve |
| I:1/4 = 1 | O:2/2 = 1 | Open V3 Valve |
| I:1/5 = 1 | O:2/2 = 0 | Close V3 Valve |
| N7:1 ≤ 20 | O:2/3 = 1 | Open V5 Valve |
| N7:1 > 20 | O:2/3 = 0 | Close V5 Valve |

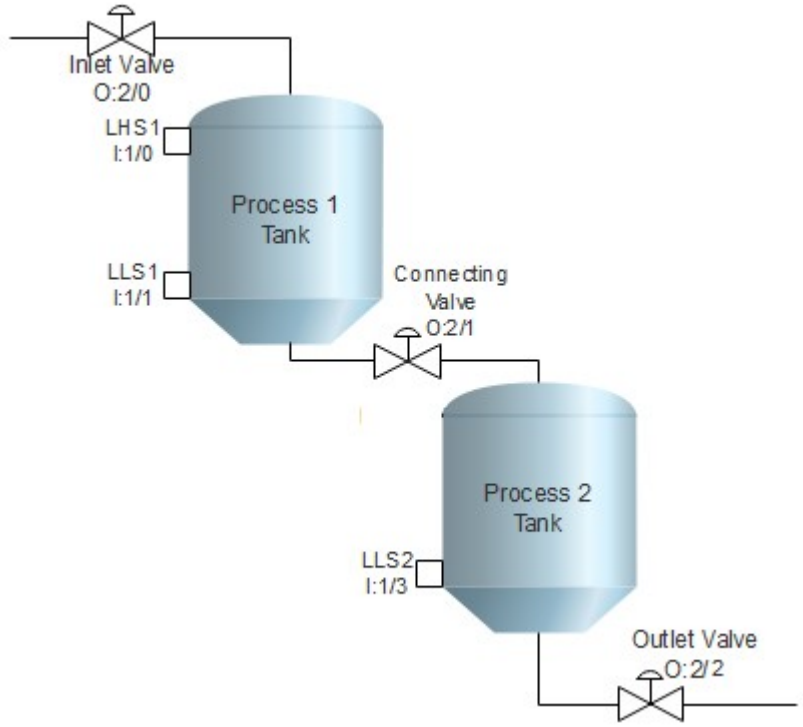
# PLC Program to Control Level of Two Tanks

This is a PLC Program to Control Level of Two Tanks.

## Problem Description

Two simultaneous processes are to be performed in two separate tanks which are connected through a valve. Process 1 takes place in the 1st tank and Process 2 takes in the 2nd tank. Control level of these tanks in PLC using Ladder Diagram programming language.

## Problem Diagram



## Problem Solution

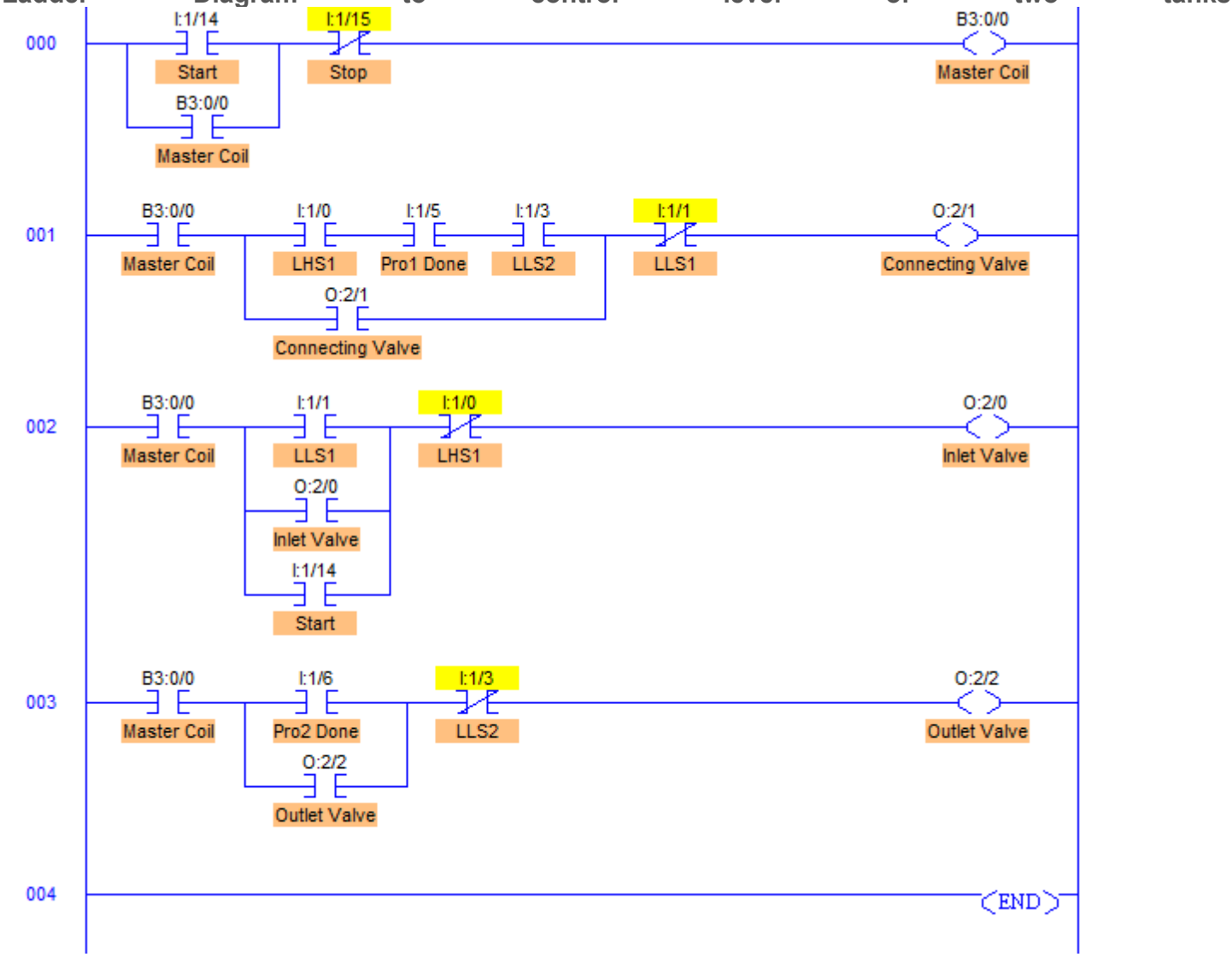
- Mount two level switches in the first tank and two switches in the second tanks. Both switches of tanks detect High and Low level of the tanks.
- Install inlet valve to control inlet process flow and outlet valve to control outlet process flow.
- Install one more connecting valve to control draining of process material from Process 1 Tank to Process 2 Tank.
- Provide interlocking to prevent from malfunctioning or overflowing.

## PLC Program

Here is PLC program to Control Level of Two Tanks, along with program explanation and run time test cases.

```
List of inputs or Outputs
 I:1/0  = Level High Switch-Tank 1, LHS1 (Input)
 I:1/1  = Level Low Switch-Tank 1, LLS1   (Input)
 I:1/3  = Level Low Switch-Tank 2, LLS2  (Input)
 O:2/0  = Inlet Valve                     (Output)
 O:2/1  = Tank Connecting Valve           (Output)
 O:2/2  = Outlet Valve                    (Output)
 I:1/14 = Start                           (Input)
 I:1/15 = Stop                            (Input)
 B3:0/0 = Master Coil Bit                 (Bit)
```

**Ladder Diagram to control level of two tanks**




## Program Description

- RUNG000 is a master start/stop rung to Start and Stop the entire process.
- RUNG001 is for controlling Connecting Valve output O:2/1. It is opened when LHS1 I:1/0, Pro1 Done I:1/5 and LLS2 I:1/3 are detected. And it is closed when LLS1 I:1/1 is detected, or in other words, when Process 1 Tank is empty.
- Important thing to note here is that if Tanks' size/shape were not same and any one XIC contact from O:2/1, I:1/0 or I:1/5 was not connected, there would have been a chance of malfunctioning/overflowing.
- RUNG002 is for controlling Inlet valve O:2/0. It is allows the inlet flow by opening Inlet Valve whenever Low Level switch of Process1 Tank LLS1 with address I:1/0 is detected.
- RUNG003 is for controlling Outlet Valve with address O:2/2. It allows the process material to flow out when Pro1 Done I:1/6 is detected.
- Consider any set point of Process1 and Process2 as I:1/5 and I:1/6.

## Runtime Test Cases

```
Inputs                          Outputs  Physical Elements
I:1/0 = 1, I:1/1 = 0             O:2/2 = 1       Open Outlet Valve
I:1/0 = 0, I:1/1 = 1             O:2/0 = 1       Open Inlet Valve
I:1/0 = 1, I:1/5 = 1, I:1/3 = 1   O:2/1 = 1       Open Connecting Valve
I:1/0 = 0, I:1/1 = 0             O:2/0 = 1       Open Inlet Valve
```

```
I:1/0 = 1                O:2/0 = 0       Close Inlet Valve
I:1/1 = 1                O:2/1 = 0       Close Connecting Valve
I:1/3 = 1                        O:2/2 = 0       Close Outlet Valve
```

# PLC Program to Control Level of a Single Tank

This is a PLC Program to Control Level of a Single Tank.

## Problem Description

One open tank is installed in the plant of which liquid level is to be controlled. When level reaches the Level Low, Outlet flow is blocked and inlet flow is allowed until high level is achieved. And when Level High is detected, outlet flow is allowed and inlet flow is blocked.

## Problem Diagram

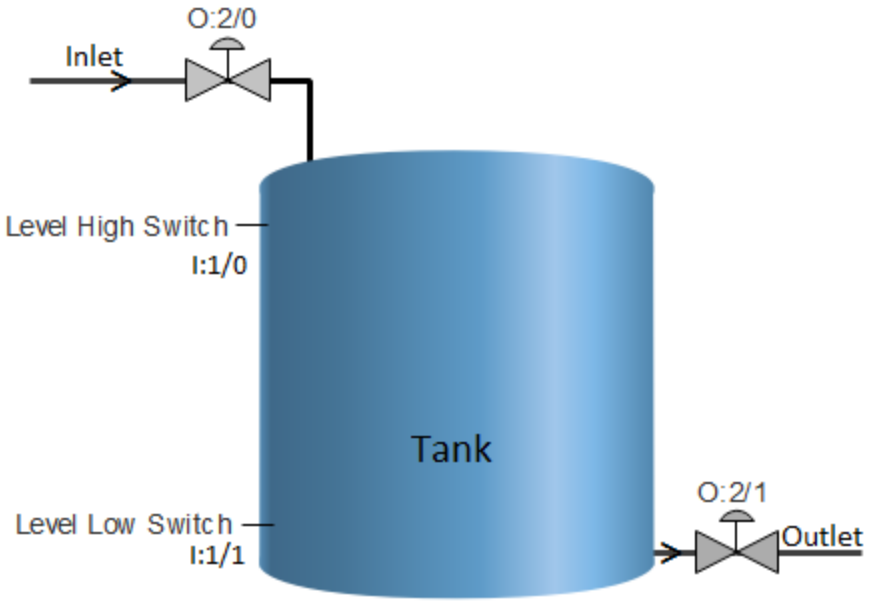



**Diagram of a single tank level control**

## Problem Solution

- To detect high and low level of liquid in the tank, two level switches are used which gives output in digital terms, that is when corresponding levels are detected, it gives output high otherwise remain low.
- To control level of this system, Single Acting piston valve can be used which has two states, either fully open or fully close.
- Low Level Switch is mounted at the bottom of the tank and Level High switch mounted at the side upper most position.
- When this inputs are detected, output to Control Valve has to be latched in order to continuously fill or empty the system.
- Master start/stop is also provided to shut down or start the entire process.

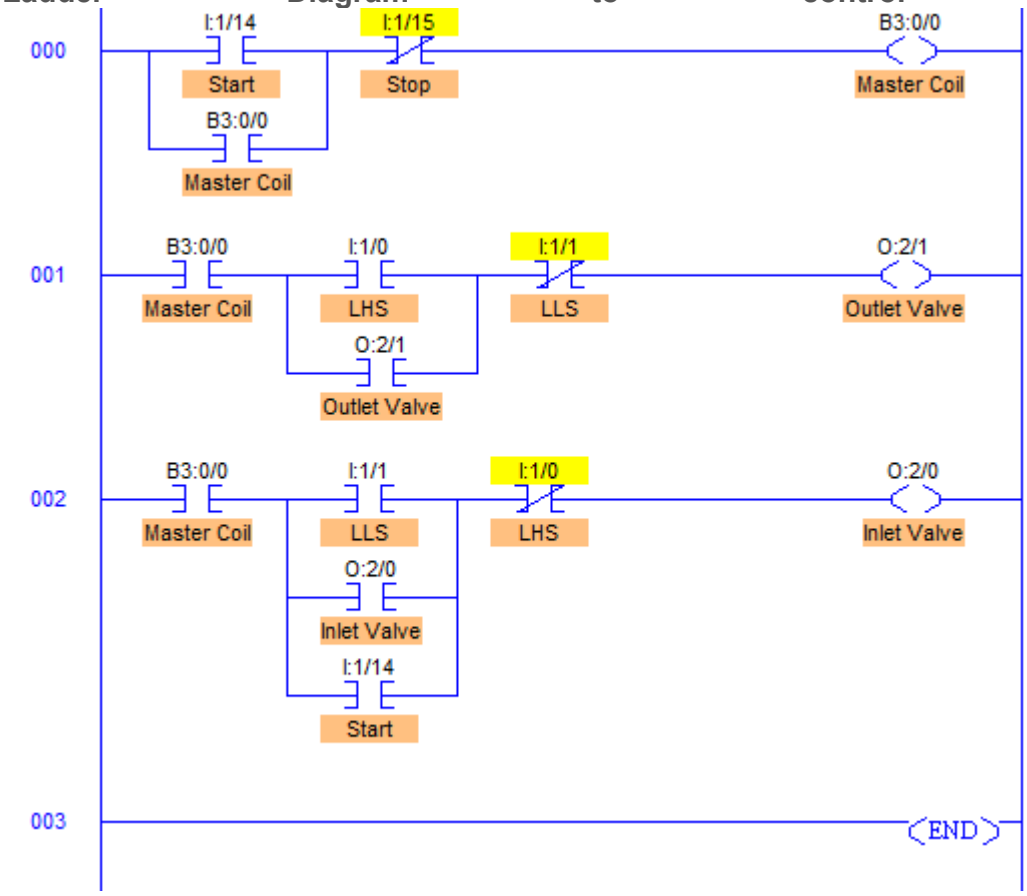
## PLC Program

Here is PLC program to Control Level of a Single Tank, along with program explanation and run time test cases.

```
List of Inputs and Outputs
```

```
I:1/0  = Level High Switch      (Input)
I:1/1  = Level Low Switch       (Input)
O:2/0  = Inlet Valve            (Output)
O:2/1  = Outlet Valve           (Output)
I:1/14 = Start                  (Input)
I:1/15 = Stop                   (Input)
```

**Ladder Diagram to control this process**




## Program Description

- RUNG000 is simply for latching a coil and master start-stop buttons.
- RUNG001 is to control the outlet valve through O:2/1. This is done when Level High is detected.
- Latching of Output O:2/1 is done because when High Level is detected, input to RUNG001 is temporary, like Push Button. So in order to keep outlet valve open until the Level Low I:1/1 is detected, latching is done. XIO of Level Low Switch is connected in series so that when Level Low is detected, it goes true closing the outlet valve.
- Similarly in RUNG002, it works exactly same. The only difference in RUNG002 is that extra I:1/14 contact in parallel with LLS.
- Suppose when the system is started and the tank is partially filled, neither LHS nor LLS is detected, in this case, outlet and inlet valves remain closed while inlet valve should open to start filling the tank because it's partially filled.
- To eliminate this error, I:1/14 (Start) is connected in parallel to LLS I:1/1 contact. This checks if LHS (I:1/0) is detected or not. If LHS is not detected, then it opens the inlet valve until LHS is detected.

## Runtime Test Cases

```
Inputs                  Outputs                  Physical Elements
I:1/0 = 1, I:1/1 = 0     O:2/1 = 1, O:2/0 = 0     LHS Detected, Open Outlet Valve
I:1/0 = 0, I:1/1 = 1     O:2/0 = 1. O:2/1 = 0     LLS Detected, Inlet Valve
I:1/0 = 0, I:1/1 = 0     O:2/0 = 1. O:2/1 = 0     None detected, Open Inlet Valve
```

# PLC Program to Control Traffic Lights and Pedestrian Lights

This is a PLC Program to Control Traffic Lights and Pedestrian Lights.

## Problem Description

Implement controlling of Traffic Lights and Pedestrian Lights using PLC in Ladder Diagram programming language.

## Problem Solution

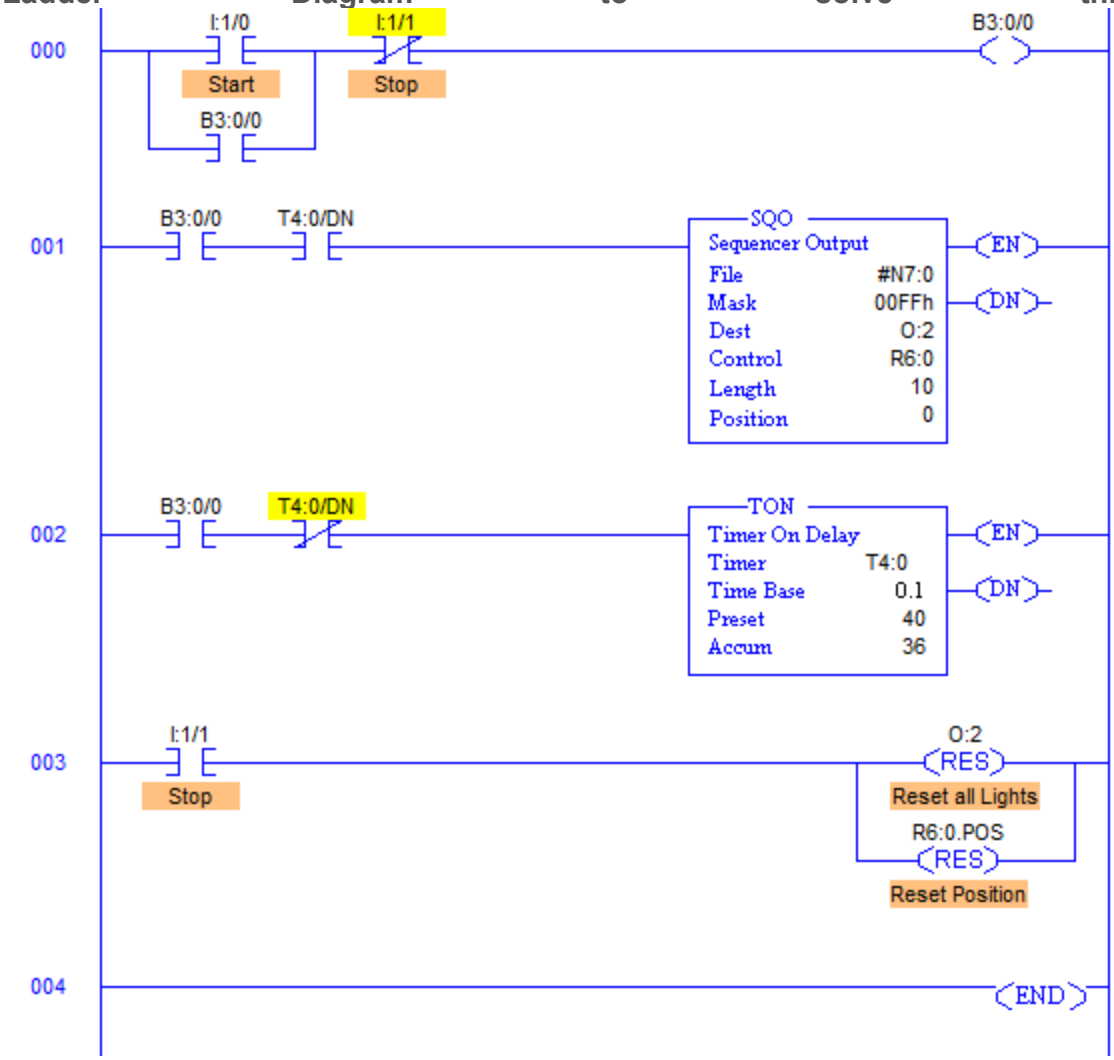
- As we have solved Traffic Lights problem, similarly we can solve this problem using Sequencer Output SQO instruction.
- In this program, two bits and outputs are added.
- File length, Control, File and Destination remain same.
- Mask data changes due to 2 added outputs.
- As 2 outputs are added, Mask will now have value 00FFh as total number of bits used are 8. In order to pass all 8bits, data flow is masked with 11111111 and moved to output.
- When Green light of South-North is ON, Pedestrian Light and Red light of East-West should be ON and vice-versa.
- Use Coil to Master Start and Stop the entire process.
- While using ordinary method to Master Start and Stop, when stop is pressed, the process is just paused and is not entirely reset, hence resetting of Position in SQO instruction and Outputs using the same Stop PB can be done manually.

## PLC Program

Here is PLC program to Control Traffic Lights and Pedestrian Lights, along with program explanation and run time test cases.

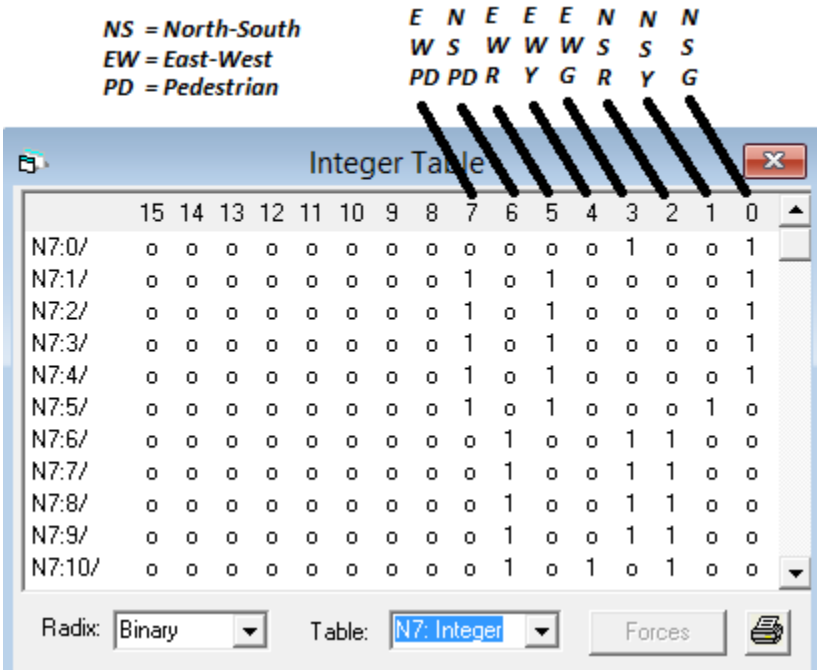
```
List of Inputs and Outputs
 I:1/0   = Start                          (Input)
 I:1/1   = Stop                                  (Input)
 B3:0/0  = Latched Coil Bit                      (Bit)
 T4:0    = Timer to update output sequence(Timer)
 SQO     = Sequencer output                      (Sequencer)
 O:2/0   = North-South Green Light               (Output)
 O:2/1   = North-South Yellow Light              (Output)
 O:2/2   = North-South Red Light                 (Output)
 O:2/3   = East-West Green Light                 (Output)
 O:2/4   = East-West Yellow Light         (Output)
 O:2/5   = East-West Red Light                   (Output)
 O:2/6   = South-North Pedestrian Light          (Output)
 O:2/7   = East-West Pedestrian Light            (Output)
```

**Ladder Diagram to solve this problem**

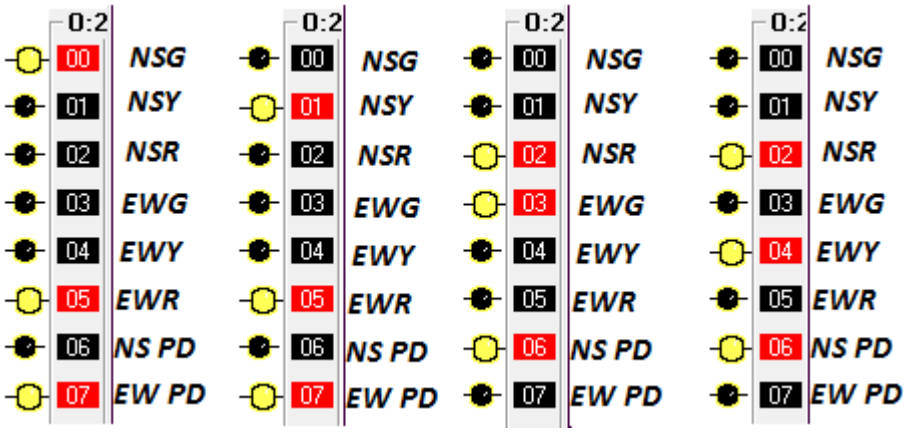



## Program Description

- O:2/0 to O:2/7 are used as the output address to Traffic Lights and Pedestrian Lights. Hence Mask has value 00FFh which means data flow of N7:0/0…N7:10/0 to N7:0/7…N7:10/7 is passed and the remaining N7:0/8…N7:10/8 to N7:0/15…N7:10/15 are blocked.
- Control parameters are assigned to register R6:0.
- Sequence of traffic lights and pedestrian lights to be operated are stored in the registers from N7:0 to N7:10 which is shown in the figure below.
- The entire operation is similar to that of Traffic Lights controlling.
- The only difference here is extra added outputs and different value to mask the data flow from N7:0… to O:2.
- B3:0/0 is used because it just uses a memory bit instead of wasting an output contact to use as a coil.




## Runtime Test Cases




# PLC Program to Control Traffic Lights

This is a PLC Program to Control Traffic Lights.

## Problem Description

Implement controlling of Traffic Lights in PLC using Ladder Diagram programming language.

## Problem Solution

- There are two methods to solve this problem. One is by using stack operation and the other one is by using sequencer output method.
- Sequencer output method is best suited for this problem since very less configuration is needed and program length is also reduced.
- In this method, we need to assign SQO instruction by configuring all the parameters given in the instruction.
- File, Mask, Dest, Control, Length and Positions are parameters which we need to configure.
- File : It is the starting address for the registers in the sequencer file.

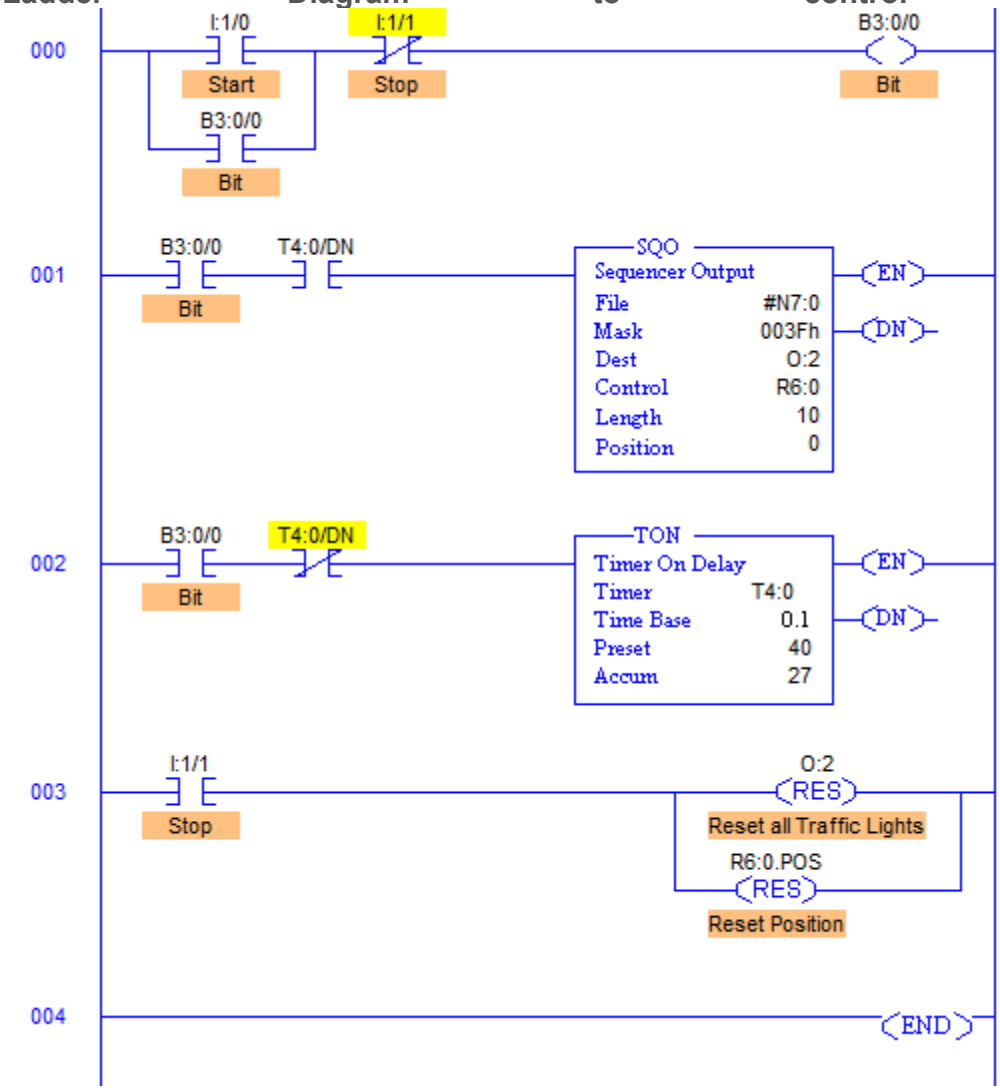
- Mask : Mask is the bit pattern through which data flow happens from source to the destination address. If there is 1 in the masking, it passes values and if 0, it blocks the data flow.
- Dest : It is the address of the input to which the Sequencer Output instruction moves the data.
- Control : Is the address that contains the parameters with control information for the instruction. EN, DN and ER are bit which sets according to the status of sequencer output. EN and DN bits are set just as in timers. ER bit stands for Error bit, it is set when a negative position/length value is detected by the processor, or zero length value.
- Length : It is the number of steps of the sequencer file starting at position 1. Position 0 is the start-up position.
- Position: It indicated the steps that is desired to start the sequencer instruction. The start position is all zeros, this is represented as the neutral position; so no outputs will be turned ON in position 0.
- So to start the actual function of output sequence, Position 1 is determined as starting sequence while programming.
- Integers or Bit Registers are used as Destination Address.

## PLC Program

Here is PLC program to Control Traffic Lights, along with program explanation and run time test cases.

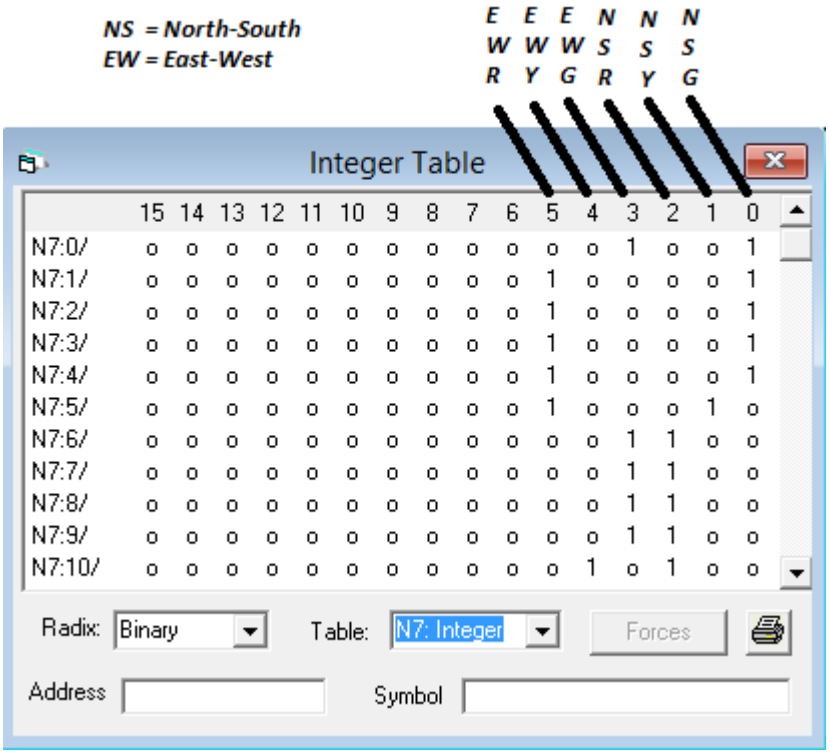
```
List of Inputs and Outputs
 I:1/0   = Start                          (Input)
 I:1/1   = Stop                                  (Input)
 B3:0/0  = Latched Coil Bit                      (Bit)
 T4:0    = Timer to update output sequence(Timer)
 SQO     = Sequencer output                      (Sequencer)
 O:2/0   = North-South Green Light               (Output)
 O:2/1   = North-South Yellow Light              (Output)
 O:2/2   = North-South Red Light                 (Output)
 O:2/3   = East-West Green Light                 (Output)
 O:2/4   = East-West Yellow Light         (Output)
 O:2/5   = East-West Red Light                   (Output)
```

**Ladder Diagram to control Traffic Light**



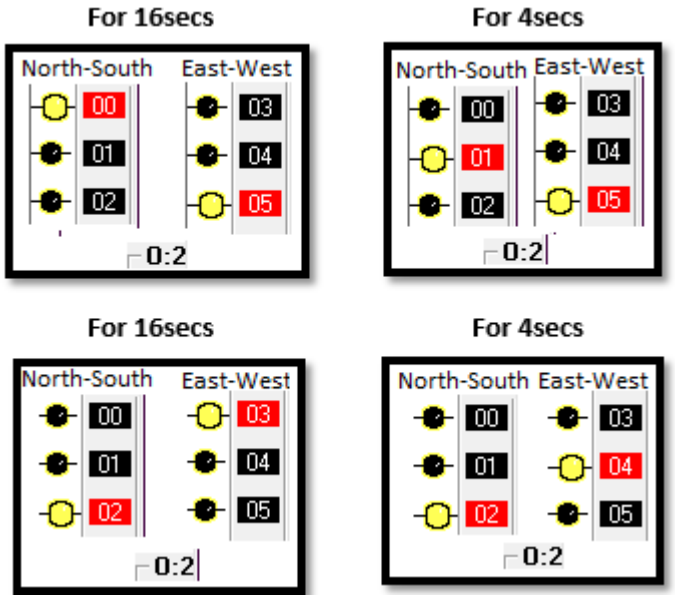

## Program Description

- RUNG000 again here is for Master Start and Stop the process.
- File; #N7:0 and File length is 10, hence output sequence is varied from N7:0 to N7:10 with each input.
- Destination is set to O:2 hence with each transition, N7:0 to N7:10 are moved to O:2 with masking.
- O:2/0 to O:2/5 are used as the output address to Traffic Lights and hence Mask has value 003Fh which means data flow of N7:0/0…N7:10/0 to N7:0/5…N7:10/5 is passed and the remaining N7:0/6…N7:10/6 to N7:0/15…N7:10/15 are blocked.
- Control parameters are assigned to register R6:0.
- Sequence of traffic lights to be operated are stored in the registers from N7:0 to N7:10 as following.




- Time base is set to 4secs, hence after every 4secs, output sequence is changed to its next register pattern outputs which is then transferred to O:2 and O:2/0 to O:2/5 are energized accordingly.
- As we can see, from N7:1 to N7:4 have the same bit pattern. So, these bits are set to 1 for 4 cycles that is 16secs. These bits are used for South-North Green light and East-West Red light.
- Similarly the entire sequence is followed.
- When Stop I:1/1 is pressed, Position is reset to 0 and all the outputs are de-energized.

## Runtime Test Cases




# PLC Program to Call a Subroutine for a Different Process

This is a PLC Program to Call a Subroutine for a Different Process.

## Problem Description

Add temperature outputs of two temperature transmitters and display the total temperature. Update display every 10 seconds when this system is active using Subroutine program.

## Problem Solution

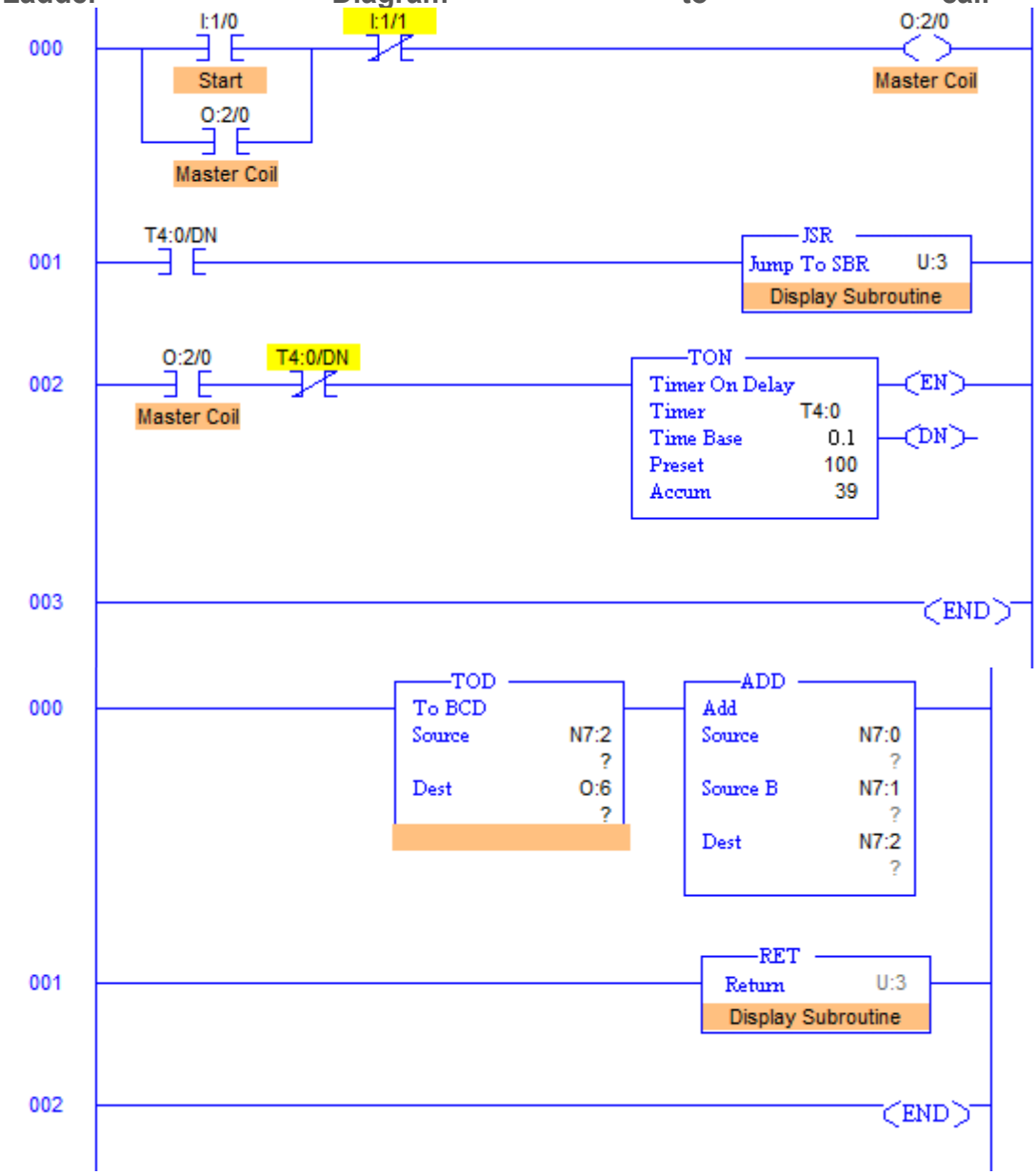
- Using subroutine has an advantage that process is not affected by any other data in subroutine program unless and until the subroutine is called.
- This is very useful in displaying various data on the same display with a particular time delay.
- For example, we can use the same display to show Pressure readings for the first 5 seconds and Temperature reading for the next five seconds and so on.
- When we use subroutine instruction, Subroutine address must be same in JSR instruction of main program and RET instruction of subroutine program.

## PLC Program

Here is PLC program to Call a Subroutine for a Different Process, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0       = Start                                              (Input)
 I:1/1       = Stop                                               (Input)
 O:2/0       = Master Coil                                        (Output)
 T4:0        = Time delay to update data                 (Timer)
 O:6         = Display address                                    (Output)
 N7:0 & N7:1 = Temperature data of transmitters          (Register)
 T4:0/DN     = Update display calling Subroutine, reset timer     (Timer Bit)
 U:3         = Address of Subroutine                             (Subroutine)
```

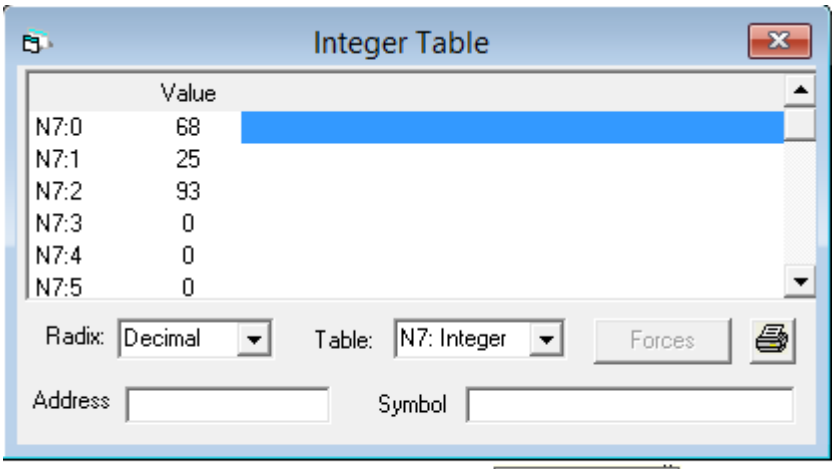
**Ladder Diagram to call subroutine**

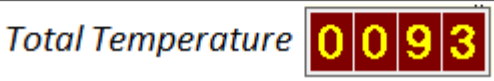



## Program Description

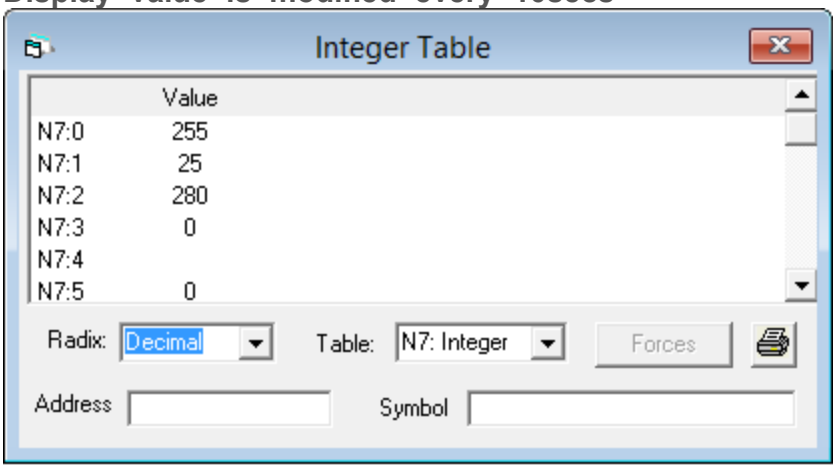
- Main program comprises of simple Master Coil rung, rung for timer and rung to call subroutine.
- When Start PB I:1/0 is pressed, Master coil energizes and the TON timer T4:0 is enabled.
- After 10 seconds when PRESET=ACCUMULATOR, T4:0/DN bit goes high calling subroutine.
- Subroutine has a program that adds two data from addresses N7:0 and N7:1 which are outputs from Temperature transmitters. These outputs are added and output of addition is converted into BCD and sent to address O:6 which is connected to Display.
- Output data are in Hexadecimal and Displays accept BCD inputs. So TOD instruction is used here to convert data of temperature addition to BCD.
- T4:0/DN is also used as an XIO input to Timer itself, hence it automatically resets to 0 whenever preset and accumulator values are equal or in other words, it updates the display every 10secs resetting timer.

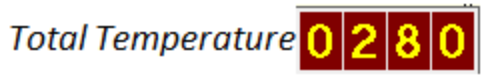
## Runtime Test Cases

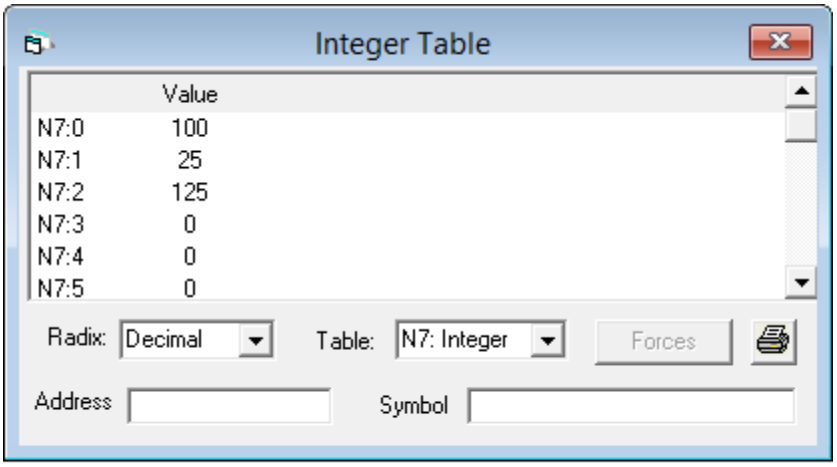


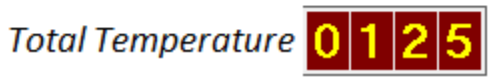




**Display value is modified every 10secs**




# PLC Program to Perform Pulse Width Modulation

This is a PLC Program to Perform Pulse Width Modulation.

## Problem Description

Implement Pulse Width Modulation in PLC using Ladder Diagram programming language.

## Problem Solution

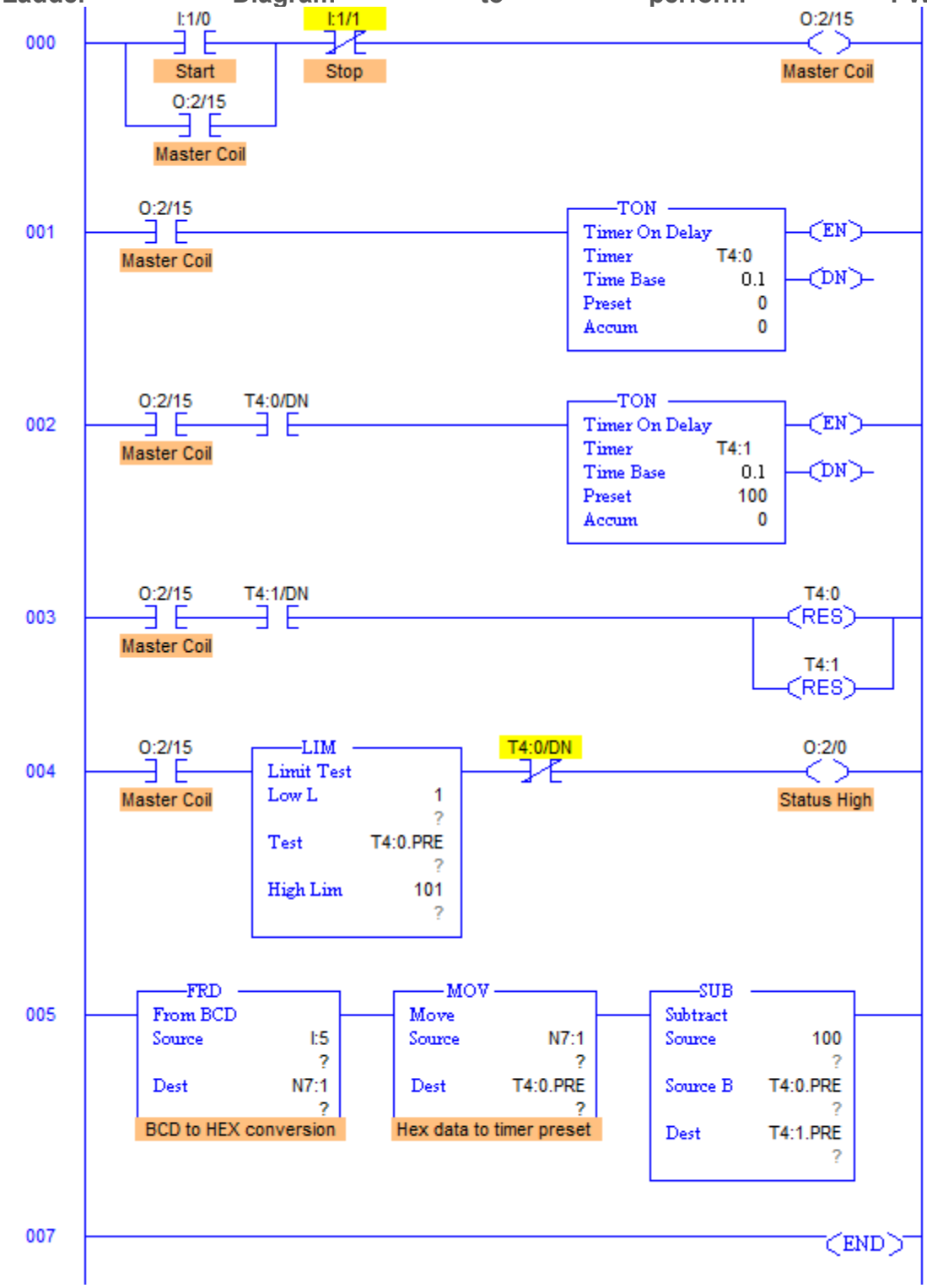
- To perform this two timers are used to Turn ON and OFF an output according to the length of a pulse.
- Timer preset value should vary such that when preset of one timer is increased, preset value of other timer should decrease in order to maintain Turn ON and OFF time of output.
- Select input bits such that we can directly enter digits and place it into Preset value of a timer.
- This can be done by using a digital input device which generates 0-9 numerical digits.
- Output of this Digital device is always in BCD form and Timers preset values store data in Hexadecimal, so whichever data are sent to preset register of a time, it has to be converted into Hexadecimal form.
- FRD instruction can be used to perform BCD to Hex conversion.
- Output of this conversion is directly moved to preset register of timer which can be performed by MOV instruction.

## PLC Program

Here is PLC program to Perform Pulse Width Modulation, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0    = Master Start PB                      (Input)
 I:1/1    = Master Stop PB                       (Input)
 O:2/15   = Master Coil                          (Output)
 T4:0     = Timer 0, ON time                     (Timer)
 T4:0.PRE = Timer 0 Preset value address  (Timer)
 T4:1     = Timer 1, OFF time                    (Timer)
 T4:1.PRE = Timer 1 Preset value address  (Timer)
 -(RES)-  = Timer reset coil                     (Output)
 I:5      = 0-100 BCD input                      (Input)
 N7:1     = BCD-Hex conversion storing register  (Register)
```

**Ladder Diagram to perform PWM operation**

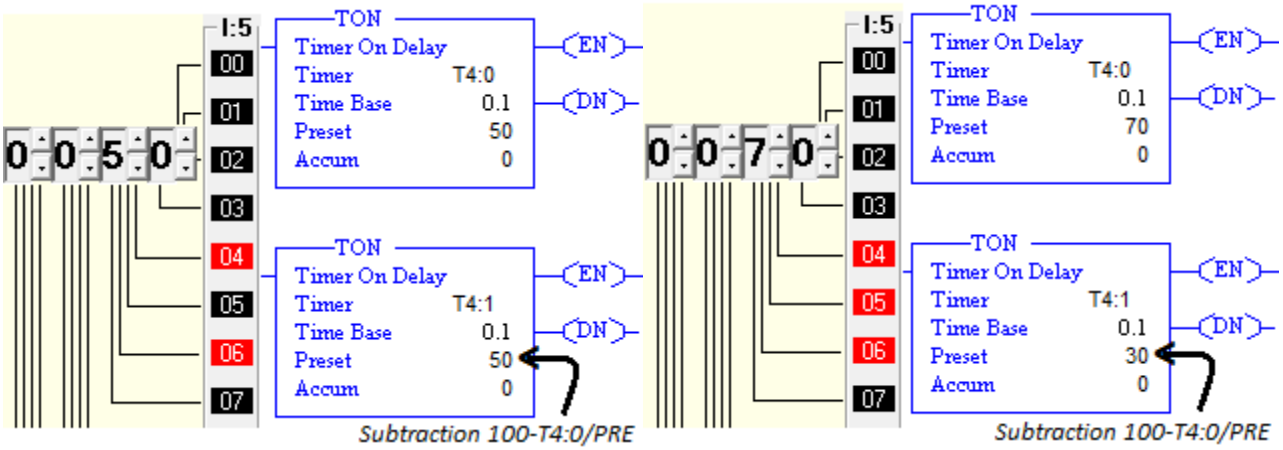



## Program Description

- RUNG000 to RUNG003 are used for timers T4:0 and T4:1 which can be understood easily.
- RUNG004 is for output to which Pulse of a particular width is provided.
- I:5 is used to give 2 digit input to preset value address which alters Preset of Timer T4:0 and determines ON time of output O:2/0.
- Preset value of T4:1 depends on T4:0.PRE.

- 2 digit output has a limit from 0-99, hence subtraction is performed to set the preset value of T4:1 timer which is OFF time of output O:2/0.
- This input is BCD number.
- Data stored in any registers in PLC are in the form of Hexadecimal. So BCD input from I:5 has to be converted into Hexadecimal which is performed by FRD instruction and moved to address T4:0.PRE.
- Now when 2 digit input is set as a preset of Timer T4:0, it is subtracted from 100 and the output is set as a preset of Timer T4:1.
- 100 – ON time = OFF time. (ON time = 0-99)
- If the input is 50, then both the timers hold the same value, hence square wave of 50% duty cycle is obtained.
- So by varying input from 0-99, width of pulse (ON time of output) is varied.
- However, when input from I:5 is 0 and OFF time is completed, output O:2/0 blinks for a while. To eliminate this error, LIM instruction is used.
- We can even use 3 and 4 digit inputs but then these digits must be subtracted from 1000 and 10000 respectively.

## Runtime Test Cases




# PLC Program to Jump to Other Process

This is a PLC Program to Jump to Other Process.

## Problem Description

Implement Jumping from one process to another process in PLC using Ladder Diagram programing language.

## Problem Solution

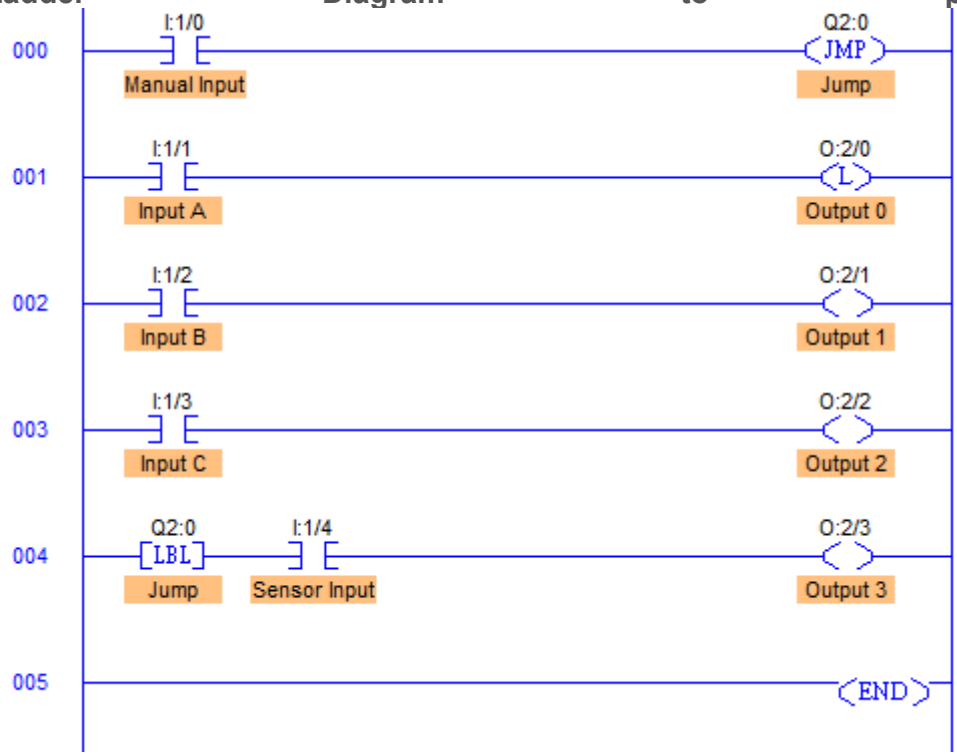
- This type of operation is most widely used in hazardous area where some action must be taken immediately despite of the status of currently running process.
- Use JMP instruction to perform this task.
- This Jump instruction does not differ from any Microcontroller Jump instruction, this Jump instruction also must be used with LBL (Label) instruction.
- Simply explaining, when JMP is activated, all the outputs between JMP and LBL are disabled until this is active. By saying ‘all outputs’, it does not include latch –(L)– outputs.

# PLC Program

Here is PLC program to Jump to Other Process, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Input to Jump to Label (Input)
 I:1/1   = Input A                (Input)
 I:1/2   = Input B                (Input)
 I:1/3   = Input C                (Input)
 I:1/4   = Sensor Input           (Input)
 O:2/0   = Output 0               (Output)
 O:2/1   = Output 1               (Output)
 O:2/2   = Output 2               (Output)
 O:2/3   = Output 3               (Output)
 Q2:0    = Jump (JMP and LBL)     (I/O)
```

**Ladder Diagram to perform Jumping**




# Program Description

- RUNG000 comprises of a Manual input which may be a toggle button or sensor output by which operation is performed.
- Until I:1/0 is pressed, JMP does not energize, that means program is being scanned rung by rung and it does not affect program or process at all.
- When I:1/0 is pressed, program is directly jumped to the LBL and scans RUNG004.
- If Sensor Input I:1/4 is true, it energizes O:2/3 turning ON Output 3.
- As long as JMP is energized, all the output between JMP and LBL rungs are skipped and Input I:1/1 to I:1/3 do not affect O:2/0 to O:2/2 at all.
- Important thing to note here is that if Latch Output is used between JMP-LBL rungs and is active then it is not unlatched when JMP energizes.

# Runtime Test Cases

**Fig.1 -Output O:2/0 is latched even if JMP is energized.**

Fig.1

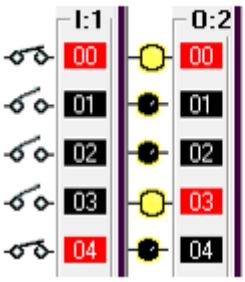



Fig.2

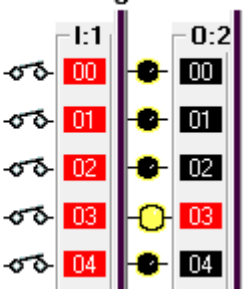



**Fig.2 -When Sensor Input is TRUE and JMP is energized.**

# PLC Program to Generate Outputs Based on Equations

This is a PLC Program to Generate Outputs Based on Equations.

## Problem Description

The following calculation will be made when input I:1/0 is true.

```
                    x=2^y  √(2+Sin y)
```

If the result x is between 1 and 10 then the Output 0 will be turned on. The value of x will be output as an Analog voltage. Perform this operation in PLC using Ladder Diagram programming language.

## Problem Solution

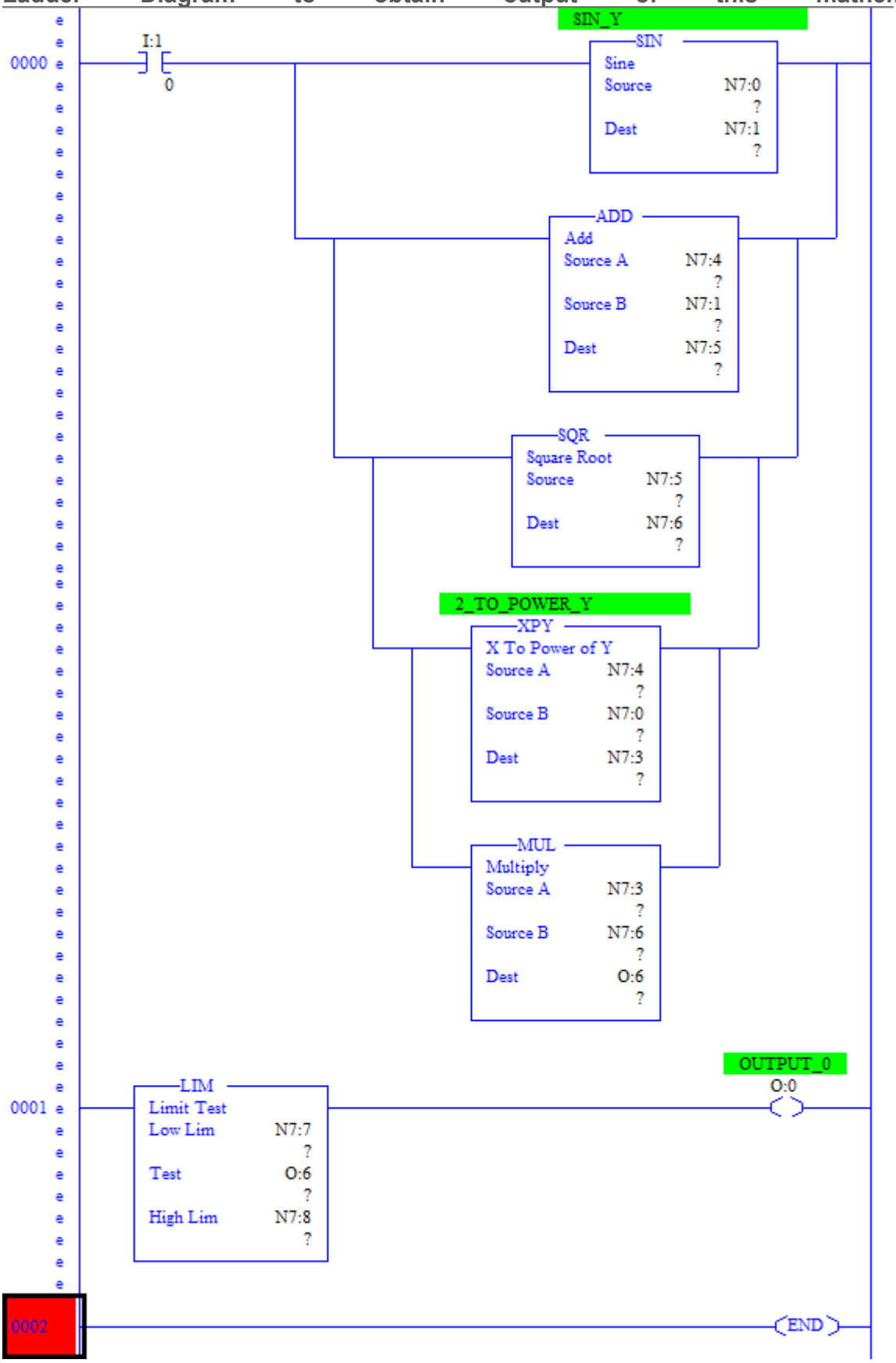
- Perform tasks individually as defined below.
- Assign integer number addresses to store the 16/32bit data into registers.
- Use this addresses in the mathematical instructions as Source and Destination.
- Use XPY (X to power Y) instruction to find the input 2y.
- Use SIN instruction to find Sin y. Add this with 1 by using ADD instruction.
- Square Root this 1 + Sin y by using SQR instruction and multiply the output of XPY and SQR.
- Compare this data with the lower and upper limit and control output.

## PLC Program

Here is PLC program to Generate Outputs Based on Equations, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Start Input                                  (Input)
 N7:0    = Y input to SIN function                      (Input)
 N7:1    = Sin Y operation data                         (Input)
 N7:4    = Numeric Value “2” (To perform 2+Sin y)  (Input)
 N7:5    = Output of addition 2+Sin Y                   (Input)
 N7:6    = Square Root of 2+Sin Y                  (Input)
 N7:4    = Numeric Value “2” (To perform 2y)            (Input)
 N7:3    = Output of X to power Y (2y)                  (Input)
 N7:7    = Lower Limit (01)                             (Input)
 N7:8    = Higher Limit (10)                            (Input)
 O:6     = Output x, multiplication of N7:6 & N7:3 (Output)
 O:0     = Output 0                                     (Output)
```

**Ladder Diagram to obtain output of this mathematical equation**




## Program Description

- As we can see, step by step procedure is done.

- Output of individual operations are first obtained.
- Output of all these data are stored into Integer Registers.
- Operation starts from the smallest function as shown in the RUNG0000 that first Sin Y(N7:0) is obtained, then Sin Y is added with the integer 2.
- Output of this 2+Sin Y is then square rooted using SQR instruction and result is stored into the register N7:6.
- Then 2y is performed. This is obtained by using XPY (X to power Y) instruction where X is N7:4 (digit 2) and Y is N7:0 (y).
- Output of this both are Multiplied to finally obtain the output of the equation X (O:6).
- LIM instruction is then used to compare output of the equation with its lower limit and upper limit. Accordingly output is generated.
- If O:6 is between 1 to 10, output O:0 energizes turning ON the Output 1.

## Runtime Test Cases

**You can check output by varying y (N7:0) input.**

*This program is tested and performed using RSLogix 500 Pro by Rockwell Automation*

# PLC Program to do Mathematical Functions

This is a PLC Program to do Mathematical Functions.

## Problem Description

Implement various mathematical functions in PLC using Ladder Diagram programming language.

## Problem Solution

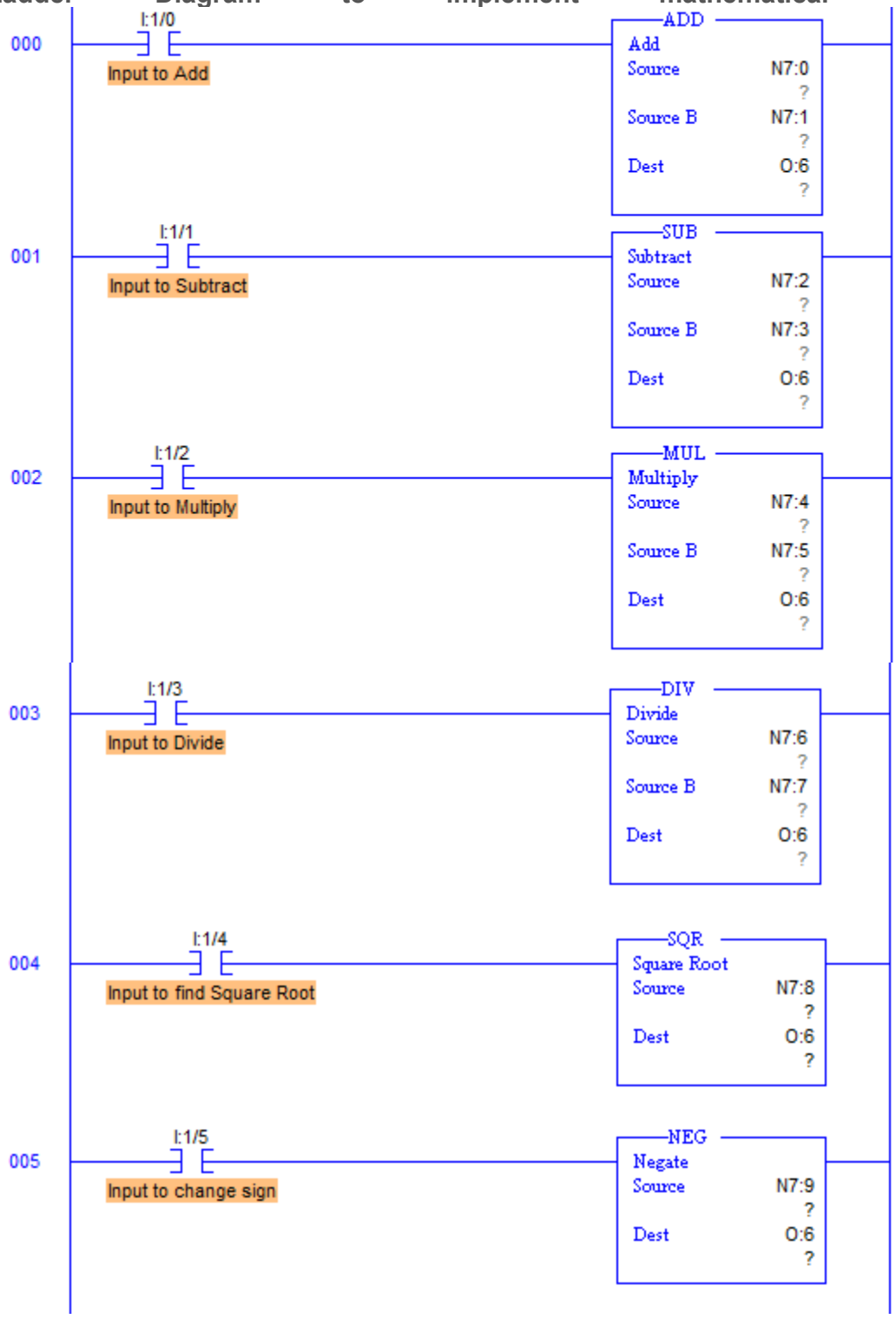
- Use all the Math Instructions to implement various Mathematical Arithmetic Functions.
- Use ADD to add one piece of data to another.
- Use SUB to subtract one piece of data from another.
- Use MUL to multiply one piece of data by another.
- Use DIV to divide one piece of data by another.
- Use SQR to find the square root of a piece of data.
- Use NEG to change the sign of a piece of data.

## PLC Program

Here is PLC program to do Mathematical Functions, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0           = Input to Add                          (Input)
 I:1/1           = Input to Subtract                     (Input)
 I:1/2           = Input to Multiply                     (Input)
 I:1/3           = Input to Divide                       (Input)
 I:1/4           = Input to find Square Root             (Input)
 I:1/5           = Input to change the sign of a Number  (Input)
 N:7/0 to N:7/9  = Integer Number Location               (Input)
 O:6             = Display                               (Output)
```

**Ladder Diagram to implement mathematical arithmetic functions**




## Program Description

- Program is very clear to understand.
- N7:0 to N7:9 contain Integers of which operation is to be performed.
- In the ADD instruction, data of N7:0 and N7:1 are added and sent to destination which is Display with output address O:6.
- Similarly all other operations are performed.

- Though, we must know that the answers stored in the destination are in Hexadecimal form, hence we have to convert this data into BCD form and then send it to display output O:6.

# PLC Program to Control Lights in a Sequence (2)

This is a PLC Program to Control Lights in a Sequence (2).

## Problem Description

Implement controlling of various lights in PLC using Ladder Diagram programming language using Bit Shift Registers.

## Problem Solution

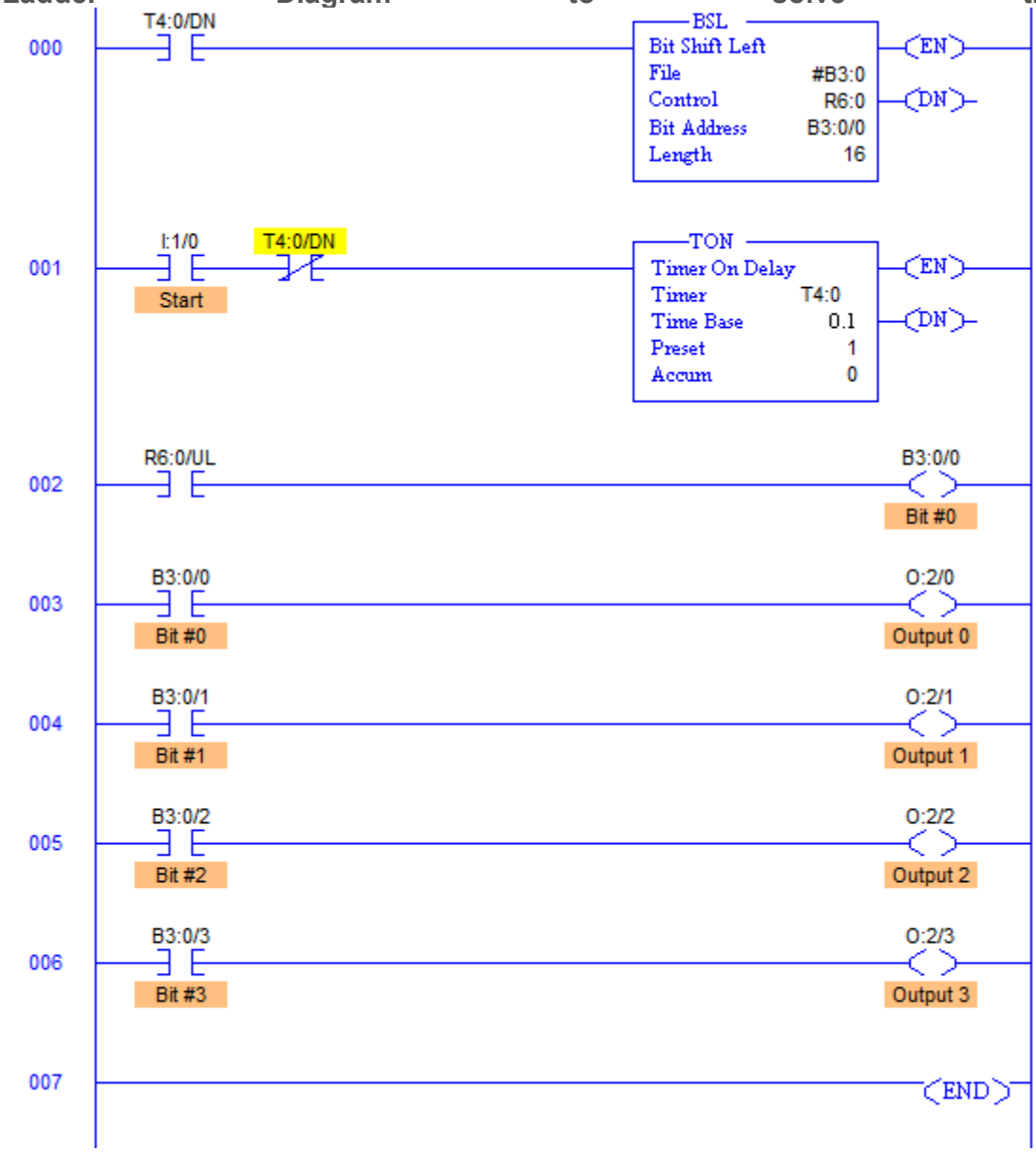
- Define order of lights.
- Use Bit Shift Registers to implement any sequence of lights.
- Double check if the order of light is made correctly and connections are made properly.
- Use latching coil for Master Start and Stop for prevention against malfunctioning.
- 0.1 Time Base function availability is useful to shift register bits very quickly.
- By using this, we can make lights blink.
- Check if bit addresses provided to Light output addresses are correctly chosen or not.
- Provide 16bit of length to Bit Shift Register, by adding this total 16 number of lights can be controlled.
- This limitation can be overcome by using more than one registers or with same length, more than one shift registers.

## PLC Program

Here is PLC program to Control Lights in a Sequence, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Master Start                   (Input)
 B3:0    = Altering Register              (Output Register)
 R6:0    = Control Register               (Storing Register)
 R6:0/UL= Used for Wraparound operation   (Unload Bit)
 B3:0/0  = Bit input to Output 0          (Input Bit)
 O:2/0   = Output 0                       (Output)
 B3:0/1  = Bit input to Output 1          (Input Bit)
 O:2/1   = Output 1                       (Output)
 B3:0/2  = Bit input to Output 2          (Input Bit)
 O:2/2   = Output 2                       (Output)
 B3:0/3  = Bit input to Output 3          (Input Bit)
 O:2/3   = Output 3                       (Output)
 T4:0    = Timer  to shift bits           (Timer)
```

**Ladder Diagram to solve this problem**



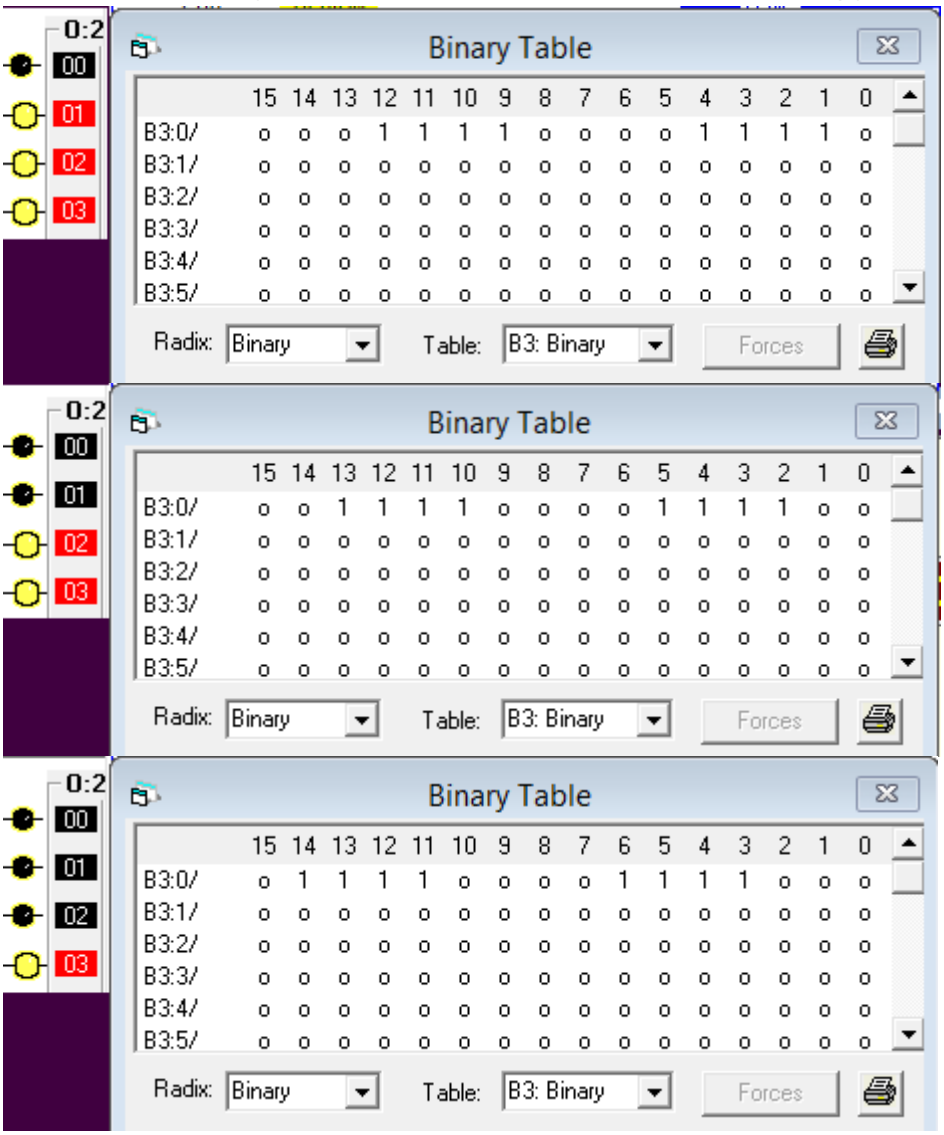

## Program Description

- Here as it can be observed in the Ladder Diagram, Outputs are controlled by the bits set in the register B3:0.
- This outputs are controlled in accordance to shifting of bits from Right to Left.
- This shifting is controlled by Timer T4:0.
- Timer is set to auto reset mode by setting XIO T4:0/DN bit of the same timer to its input. Master Start is included as well. This controls the entire process. This has to be a toggle switch otherwise latching rung is used in case Push Buttons are available and one more Stop PB must be added to Master Stop.
- Whenever Timer Done bit is set, that bit sends False to True signal to Bit Shift Register which performs Shifting of bits to Left.
- Time delay is 0.1secs here, so that shifting is real quick.
- Length indicates the number of bits to be shifted, or the file length in, in bits.
- The Last bit is shifted out of the array and stored in the unload bit, R6:0/UL. The status that was previously in the unload bit is lost. So for wraparound condition, last bit of the array B3:0/0 is set to the position to the UL bit.
- After every 0.1secs, bits are shifted and orders outputs are controlled.

- The sequence of this outputs can be set by setting sequence of bits in the register B3:0 manually or automatically storing sensors' outputs.

## Runtime Test Cases

- Bits are shifted and Outputs are changed after every 0.1secs.
- Outputs are true in the sequence and go false in the same sequence if this bit pattern in the Register B3:0 is followed.
- You can set your own pattern and operate outputs accordingly.




# PLC Program to Control Lights in a Sequence (1)

This is a PLC Program to Control Lights in a Sequence (1).

## Problem Description

Implement controlling of various lights in PLC using Ladder Diagram programming language using timers. Retentive Timer is suggested to use.

## Problem Solution

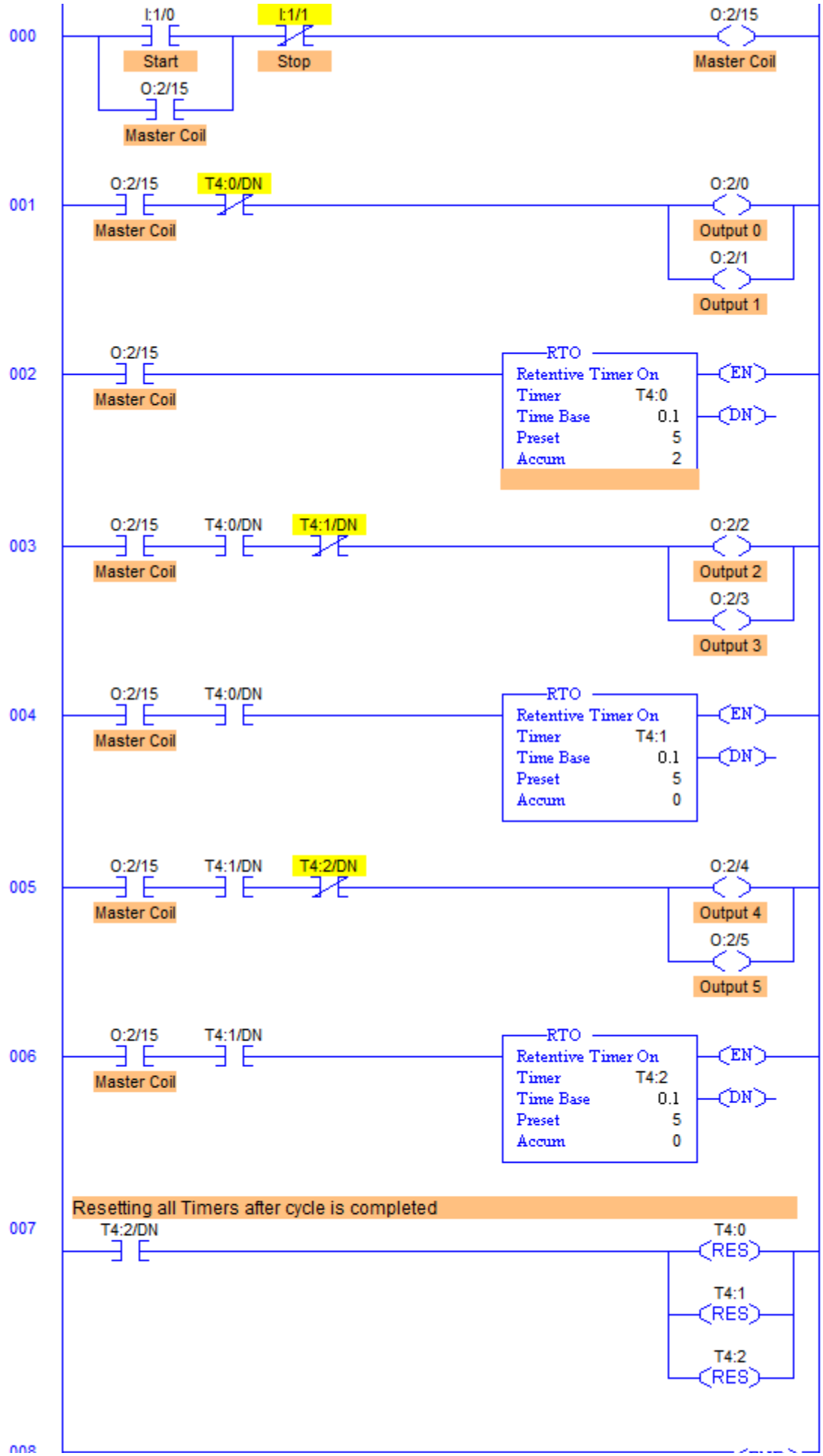
- Define order of lights.
- Provide timers to lights, to each individually if necessary.
- Reset timers automatically or use reset coil to reset timers.
- Double check if the order of light is made correctly and connections are made properly.
- Use latching coil for Master Start and Stop for prevention against malfunctioning.
- 0.1 Time Base function availability is useful to turn ON and OFF a light.
- By using this, we can make lights blink.
- This is one method to solve this problem by using timers.
- Retentive Timers have a capability of storing the previous values at which timer was stopped or input was withdrawn.
- Hence Retentive Timer RTO can be used here so that in case of power failure, program can be restarted from where it was left previously.

## PLC Program

Here is PLC program to Control Lights in a Sequence, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Master Start                  (Input)
 I:1/1   = Master Stop                   (Input)
 O:2/15  = Master Coil                   (Output)
 O:2/0   = Output 0                      (Output)
 O:2/1   = Output 1                      (Output)
 O:2/2   = Output 2                      (Output)
 O:2/3   = Output 3                      (Output)
 O:2/4   = Output 4                      (Output)
 O:2/5   = Output 5                      (Output)
 T4:0    = Timer for Outputs 0 & 1       (Timer)
 T4:1    = Timer  for Outputs 2 & 3      (Timer)
 T4:2    = Timer for Outputs 4 & 5       (Timer)
 -(Res)-= Reset coil to reset timers     (Output)
```

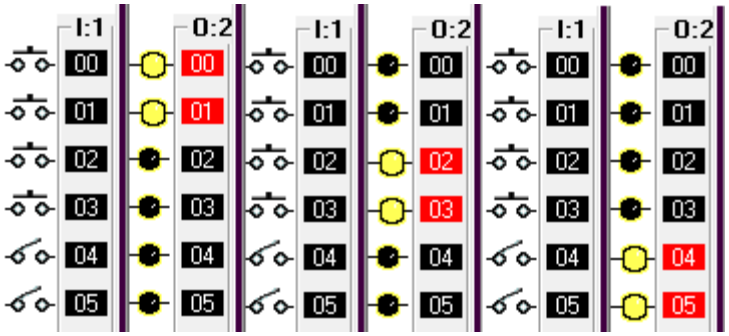
**Ladder Diagram to control sequence of lights**


000
I:1/0
Start
I:1/1
Stop
O:2/15
Master Coil
O:2/15
Master Coil
001
O:2/15
Master Coil
T4:0/DN
O:2/0
Output 0
O:2/1
Output 1
002
O:2/15
Master Coil
RTO
Retentive Timer On
Timer T4:0
Time Base 0.1
Preset 5
Accum 2
EN
DN
003
O:2/15
Master Coil
T4:0/DN
T4:1/DN
O:2/2
Output 2
O:2/3
Output 3
004
O:2/15
Master Coil
T4:0/DN
RTO
Retentive Timer On
Timer T4:1
Time Base 0.1
Preset 5
Accum 0
EN
DN
005
O:2/15
Master Coil
T4:1/DN
T4:2/DN
O:2/4
Output 4
O:2/5
Output 5
006
O:2/15
Master Coil
T4:1/DN
RTO
Retentive Timer On
Timer T4:2
Time Base 0.1
Preset 5
Accum 0
EN
DN
Resetting all Timers after cycle is completed
007
T4:2/DN
T4:0
RES
T4:1
RES
T4:2
RES
008

## Program Description

- This program is just a simple combination of Timers and Outputs.
- The only difference here is that Retentive Timer is used instead of Timer ON TON or Timer OFF TOF.
- Various outputs Output 0 to 5 are used and made a pair of two outputs.
- One timer is provided to each pair of 2 outputs. Timers DN bits are used to Turn ON and OFF lights.
- The only difference here in this program is that Retentive Timers are used so that when Input is withdrawn or power fails, Timers do not reset itself because this Timer has a capability of storing last accumulated value and hence when again input is given, the program starts from where it was left previously.
- Simply time delay of 0.5 is generated here for each pair of lights.

## Runtime Test Cases




**Simple Lights' Sequence**

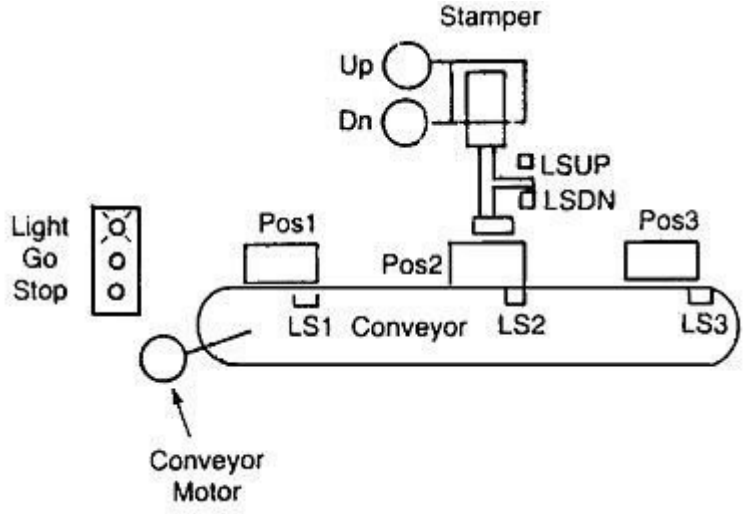
# PLC Program to Operate Stamping of Parts

This is a PLC Program to Operate Stamping of Parts.

## Problem Description

When a part is placed on the conveyor at position 1, and when a start button is pressed it moves to position 2. Upon reaching position 2, it stops for the stamping operation to take place. After stamping it automatically moves to position 3. It stops at position 3, where the part is removed manually from the conveyor. Assume only one part is on the conveyor at a time. Add limit switches, interlocks, push buttons, etc. as required.

## Problem Diagram



## Problem Solution

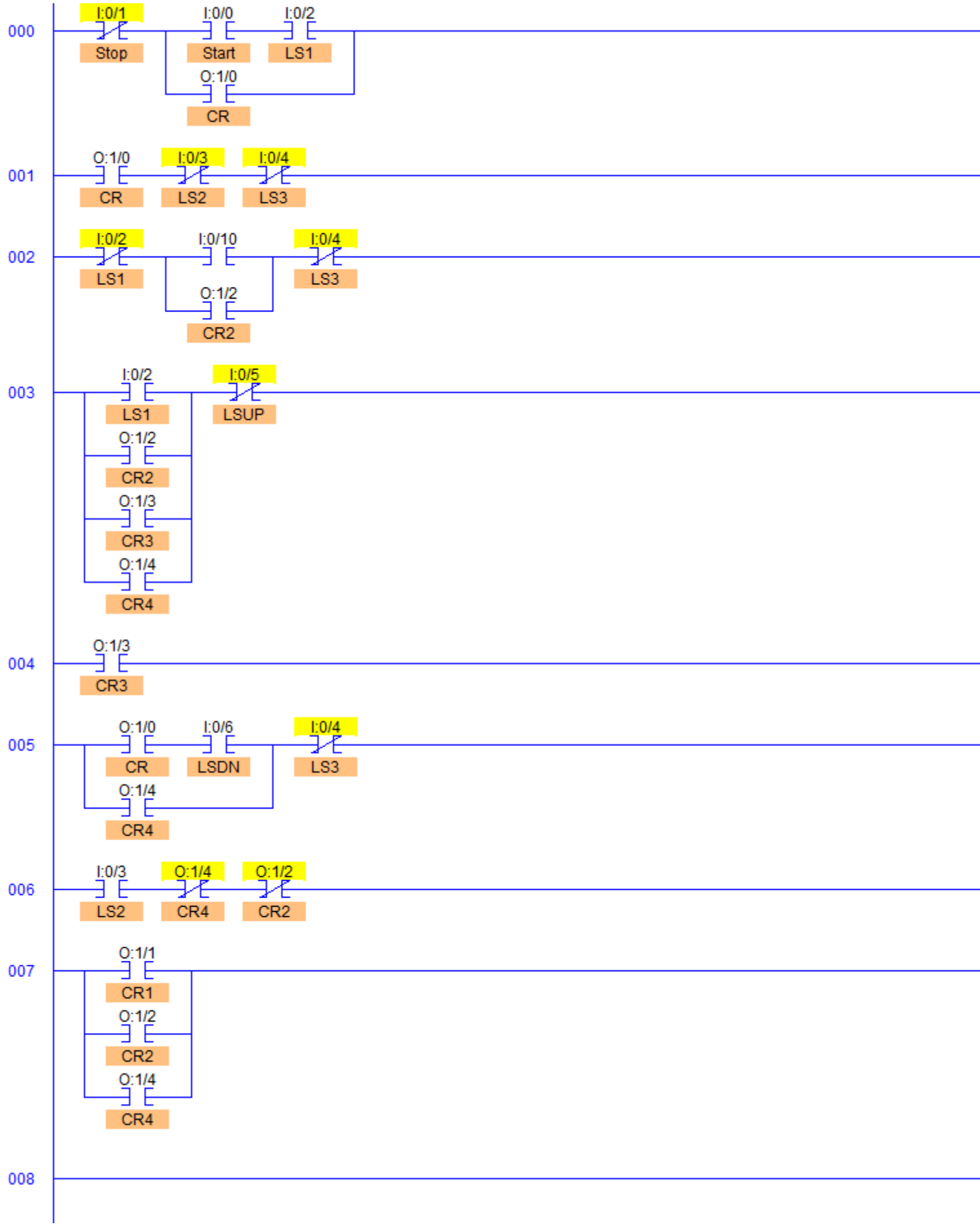
- Assuming all the contacts available are of Normally Open type.
- Push Buttons to Start and Stop the process in case of malfunctioning.
- Use of Level Switches to detect the positions 1, 2 and 3 as shown as LS1, LS2 and LS3 respectively.
- Reversible motor with UP_MOTOR coil for reverse direction and DN_MOTOR coil for forward direction control.
- Conveyor Motor to move the part from position 1 to 2 and after stamping, 2 to 3.
- LSDN and LSUP are two other limit switches which detect the lower most and upper most position of the stamper arm.
- Additional use of relay contacts in the software during programming to store the various bits in order to run Motor continuously.
- Interlocking by using XIC contacts in the software in order to prevent from Malfunctioning.

## PLC Program

Here is PLC program to Operate Stamping of Parts, along with program explanation and run time test cases.

```
List of Inputs & Outputs with addresses
I:0/0  = Start                                      (Input)
I:0/1  = Stop                                               (Input)
I:0:2  = Level Switch to detect position 1                  (Input)
I:0/3  = Level Switch to detect position 2                  (Input)
I:0/4  = Level Switch to detect position 3                  (Input)
I:0/6  = Level Switch to detect lower most position         (Input)
I:0/5  = Level Switch to detect upper most position         (Input)
O:1/10 = Motor coil for reverse direction rotation          (Output)
O:1/11 = Motor coil for forward direction rotation          (Output)
```

**Ladder Logic Diagram for the given problem is shown below.**


000
I:0/1
Stop
I:0/0
Start
I:0/2
LS1
O:1/0
CR
001
O:1/0
CR
I:0/3
LS2
I:0/4
LS3
002
I:0/2
LS1
I:0/10
O:1/2
CR2
I:0/4
LS3
003
I:0/2
LS1
I:0/5
LSUP
O:1/2
CR2
O:1/3
CR3
O:1/4
CR4
004
O:1/3
CR3
005
O:1/0
CR
I:0/6
LSDN
I:0/4
LS3
O:1/4
CR4
006
I:0/3
LS2
O:1/4
CR4
O:1/2
CR2
007
O:1/1
CR1
O:1/2
CR2
O:1/4
CR4
008

## Program Description

- Rung000- When start button is pressed and position 1 is detected by LS1, CR1 relay goes ON and is latched.
- Rung001- As long as CR is high, it latches CR1 and CR1 sends continuous true pulse to O:0/12 of Rung007 which energizes motor coil starting the MOTOR. This MOTOR is stopped only when LS2 and/or LS3 are detected.
- Rung002- CR2 is used when the process is stuck in the middle. In this case, manual switch with address I:0/10 is used to restart the MOTOR.
- Rung003- CR3 is energized whenever LS1, CR2 or CR4 goes high. CR3 relay is used to operate the UP_MOTOR coil of the stamping motor.
- Rung004- UP_MOTOR coil is energized due to CR3 and stamper comes in the main position. This is de-energized when LSUP is detected which is the upper most and main position of the stamper.
- Rung005- When stamping is done that is when lower most position is detected, CR4 is energized and latched starting the MOTOR until LS3, position 3 of the part is detected.
- Rung006- DN_MOTOR coil is energized in order to stamp the part which is when LS2, position 2 of the part is detected. It stays de-energized whenever CR4 (CR*LSDN) and CR2 (MOTOR) are ON/Energized.
- Rung007- Conveyor belt is started whenever MOTOR coil is energized that is whenever LS1 and LSDN is detected that is whenever Position 1 is detected and again when stamping is done.
- Rung008- End of the program and the scan cycle is repeated scanning from the Rung 000 again.

## Runtime Test Cases

```
Input            Relay Outputs    MOTOR
LS1=1,Start=1    CR=ON, CR1=ON, CR3=ON    ON
LS2=1    CR1=OFF, CR3=OFF OFF

Input            Relay Outputs    UP_MOTOR DN_MOTOR
LSDN=1,LS3=0     CR4=1            OFF       OFF
LSUP=0,LS1=1     CR3=1            ON        OFF
LS2=1            C2=0,C4=0       OFF        ON
```

# PLC Program to Drive Motor in Forward and Reverse Direction

This is a PLC Program to Drive Motor in Forward and Reverse Direction.

## Problem Description

A motor is connected to PLC. Run this motor in the Forward and Reverse direction using Ladder Diagram programming language.

## Problem Solution

- For any three-phase AC motor, reversing can be accomplished by reversing any two leads. For single-phase motor, reversing start lead with respect to the main leads. And for DC motor, reversing the field leads with respect to the armature leads.
- There are certain ways to reverse the motor. One is to use DPDT (Double Poles, Double Throw) switch and another one is by using Reversing Contactors.
- DPDT switch is best suitable for reversing a small DC motors while Reversing Contactor is used to reverse Three-Phase AC motors.

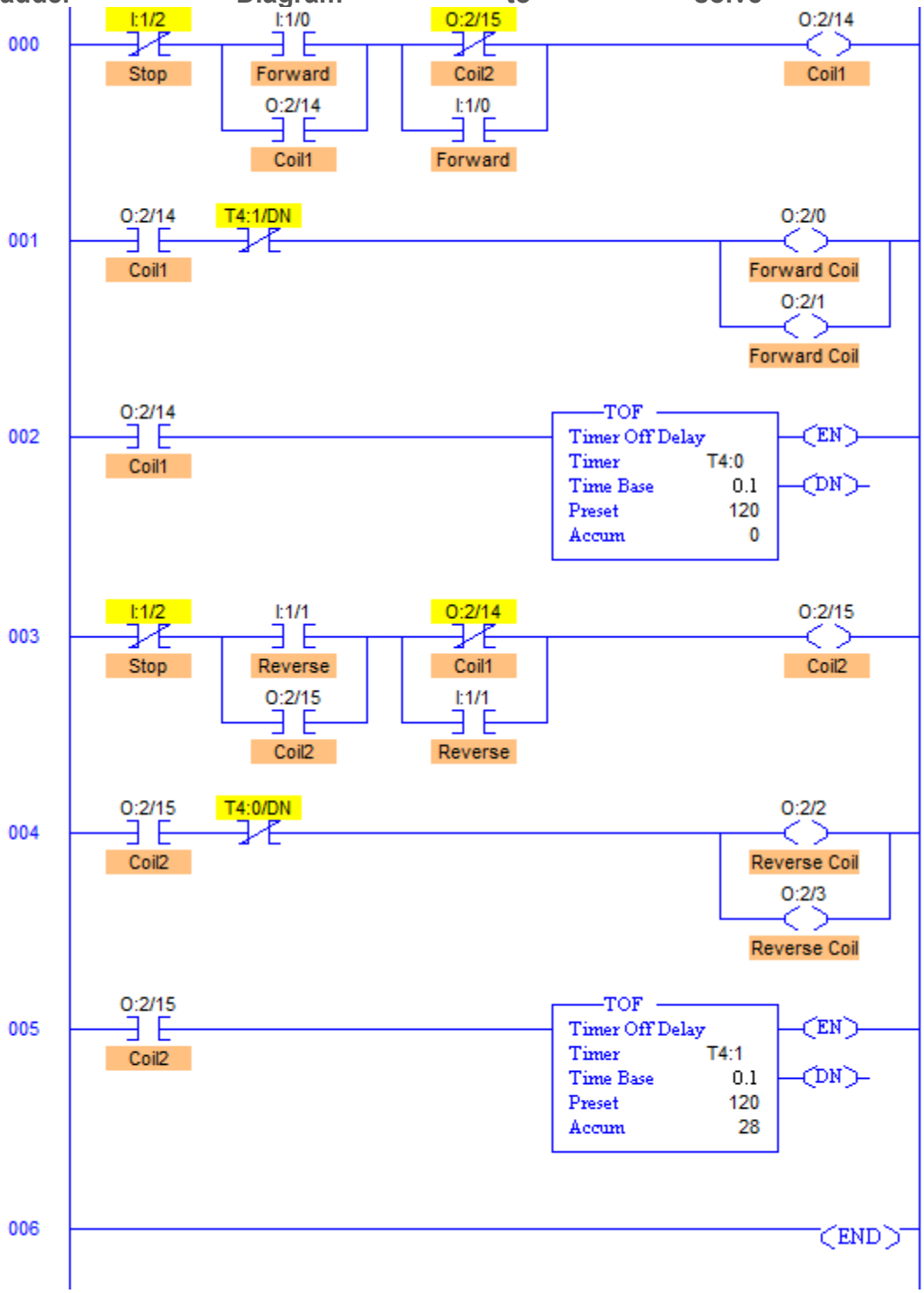
- Single-phase motors are not widely used for reversing operation. In fact, these are not even available widely with reversing capability.
- As here we have a three-phase AC motor, reversing any two leads will drive the motor in reverse direction.
- Contactor is an electrical switch used for switching an electrical power circuit.
- Two magnetic contactors are used, one for forward connections and the other for reverse connections.
- Only Push Button switches are used to control the direction of this three-phase AC motor.
- We have the input commands to these Push Button switches which are then internally processed by PLC and then there is the output terminal which activates the corresponding relay to energize the relevant magnetic contactors.
- Configure forward and reverse wiring of the motor with contactors such that forward contactor is connected directly in the normal direct phasing of the motor terminal and reverse contactor is connected with two of the motor terminals in the opposite phase.
- When it is switched to reverse direction, forward rotation does not stop instantaneously hence we have to determine what time it takes to completely stop one particular direction. Then provide time delay of a second or two and activate the other contactor.

## PLC Program

Here is PLC program to Drive Motor in Forward and Reverse Direction, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0           = Forward Start                          (Input)
 I:1/1           = Reverse Start                          (Input)
 I:1/2           = Stop                                   (Input)
 O:2/14          = Latched coil 1 for forward direction   (Output)
 O:2/0 & O:2/1   = Forward Contactor                      (Output)
 T4:1            = Delay before forward direction (Timer)
 O:2/15          = Latched coil 2 for reverse direction   (Output)
 O:2/2 & O:2/3   = Reverse Contactor                      (Output)
 T4:0            = Delay before reverse direction (Timer)
```

**Ladder Diagram to solve this problem**




## Program Description

- Important thing to note here that basically programmers do not provide time delay for such applications where motor has to be run in forward and reverse directions.
- But when motor supply is cut down, it has an actual breaking during which its speed reduces and then comes to rest.
- So assuming that the motor takes approximately 10secs to come down to its rest state.

Suppose the motor is currently running in the forward direction.

- While it is running in the forward direction, O:2/15 coil2 remains de-energized and reverse action is not activated. TOF is used here so input to T4:0 is true hence timer is not activated.
- When I:1/1 is pressed, O:2/14 coil1 energizes making coil2 O:2/15 de-energized.
- When O:2/15 is de-energized, input to timer T4:1 goes true from false and input to T4:0 goes false from true.
- Since TOF is used, when input goes true to false, T4:0 is activated. XIO of T4:0/DN is given to Reverse Coils (O:2/2 and O:2/3) are not energized until timer count is completed. Completion of time delay sets Done bit to low energizing Reversing Coils (O:2/2 and O:2/3).
- Similar operation happens when motor is running in the reverse direction and forward direction input is given.

## Runtime Test Cases

- Simulation of this problem was successfully performed in software LogixPro of Allen Bradley and verified using I/O Simulator.
- Instead of actual contactors or motor outputs, simple LED outputs were used to perform this in I/O Simulator.

# PLC Program to Drive Motors Simultaneously with Interlocking

This is a PLC Program to Drive Motors Simultaneously with Interlocking.

## Problem Description

Two Motors are running in a sequence one by one for a particular time. If the start button is pressed Motors run in sequence such that 1st Motor stays ON for 5secs and then 2nd Motor is turned ON and stays ON for 5secs. And the cycle is repeated until it is interrupted. While motors are running in the sequence, if one motor is running and the button of other motor is pressed, then the running Motor should stop and the other motor should run. Implement this logic in PLC using Ladder Diagram programming language.

## Problem Solution

- Define addresses of motors.
- Create latching seal in contact to start and stop the sequential operation.
- Use TON timer to generate a particular time delay, same or different.
- Use DN bit of first timer to energize 2nd motor coil and activate second timer.
- No need to use Reset coils, program will reset the timers itself after completion of each cycles.
- Use Parallel Start Motor contact to start one motor when other motor is running.

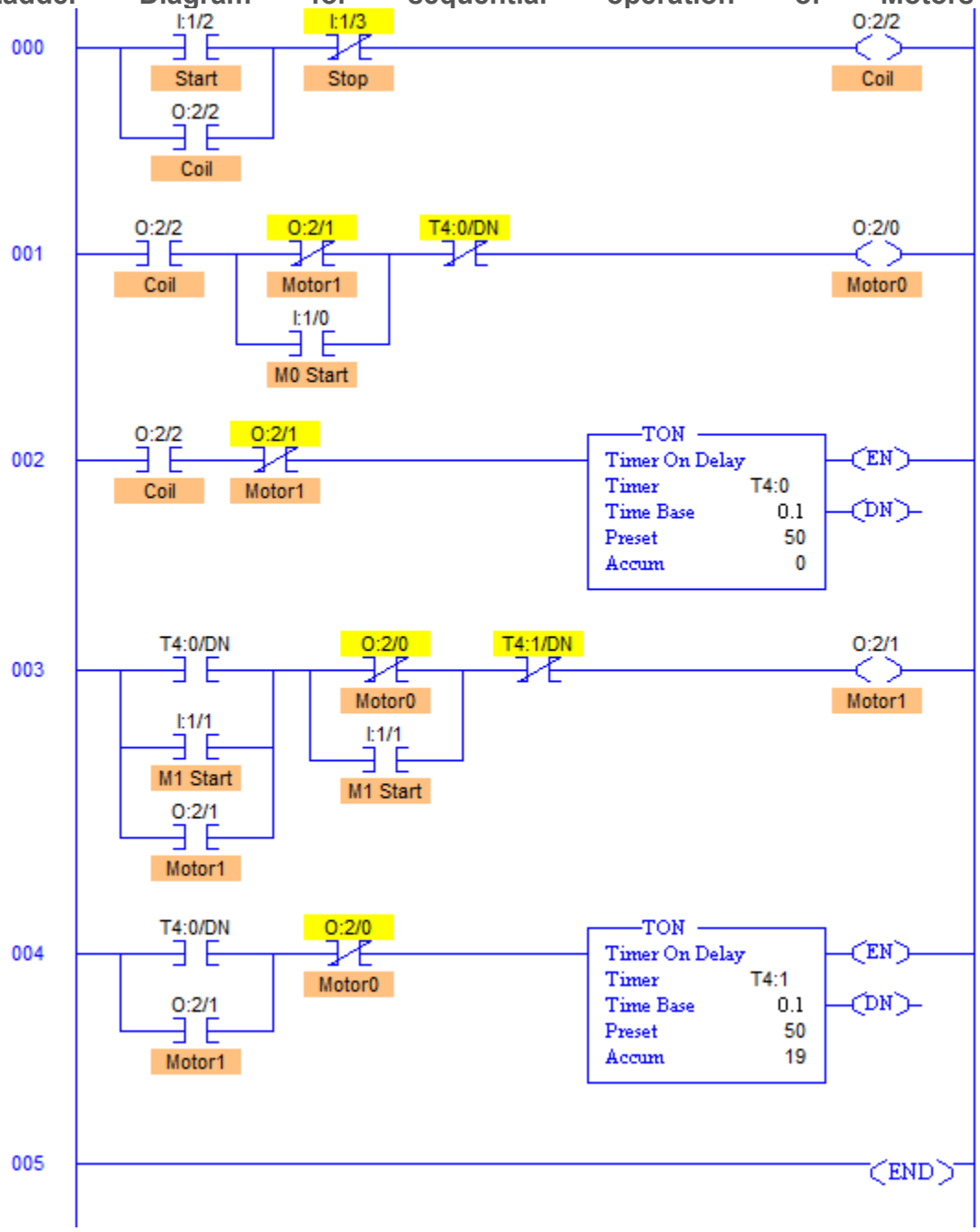
## PLC Program

Here is PLC program to Drive Motors Simultaneously with Interlocking, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Motor0 Start           (Input)
 I:1/1   = Motor2 Stop            (Input)
 I:1/2   = Master Start           (Input)
 I:1/3   = Master Stop            (Input)
 O:2/2   = Master Coil            (Output)
 O:2/0   = Motor0         (Output)
```

```
T4:0    = Motor0 Timer           (Timer)
O:2/1   = Motor1         (Output)
T4:1    = Motor1 Timer           (Timer)
```

**Ladder Diagram for sequential operation of Motors with Interlocking**



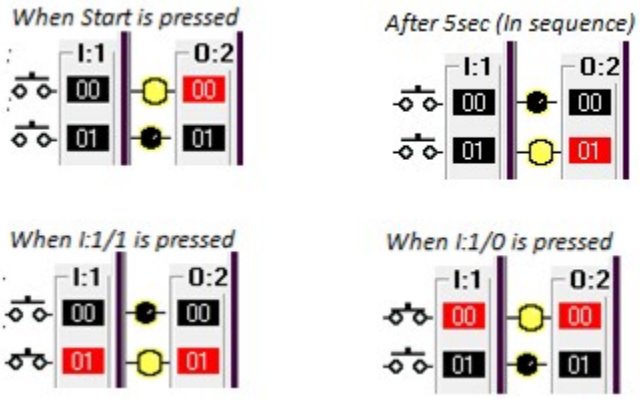

## Program Description

- RUNG000 is Latching coil for Master Start and Stop.
- RUNG001-RUNG002 and RUNG003-RUNG004 are arrangements to operate Motor0 O:2/0 and Motor1 O:2/1 respectively.
- When Start button is pressed momentarily, it latches the coil starting the energizing the output O:2/0, hence turning Motor0 ON.
- Timer T4:0 is used to run Motor0 for 5secs (Less because of simulation purpose).
- Similarly T4:1 is used to run Motor1 for 5secs.

- While Motor0 is running, and M1 Start I:1/1 is pressed, it stops the Motor0 (O:2/0 de-energizes) and Motor1 (O:2/1 energizes) starts and timer t4:1 is activated. And vice-versa.
- XIO contact of each motor's output is used to turn other motor off by de-energizing other motor's output which cuts off the current supply in the field.

## Runtime Test Cases

- LogixPro is used to Run Test Cases. You can also make this program in any simulation software and check all the cases.
- Worst cases were taken into account and solved.




# PLC Program to Operate 4 Outputs Simultaneously with Time Delay

This is a PLC Program to Operate 4 Outputs Simultaneously with Time Delay.

## Problem Description

There are total four number of outputs which should be run one by one with a particular time delay. Implement this in PLC using Ladder Diagram programming language.

## Problem Solution

- Generate Master Start and Stop buttons to activate the sequence and define four outputs.
- Use TON timer to generate a particular time delay, same or different.
- Use Done bit of first timer to energize other output and activate second timer.
- Repeat this process until the final output is energized and last timer is activated.
- Reset timers after the completion of first cycle if necessary.
- Use LEDs as output to test the program.

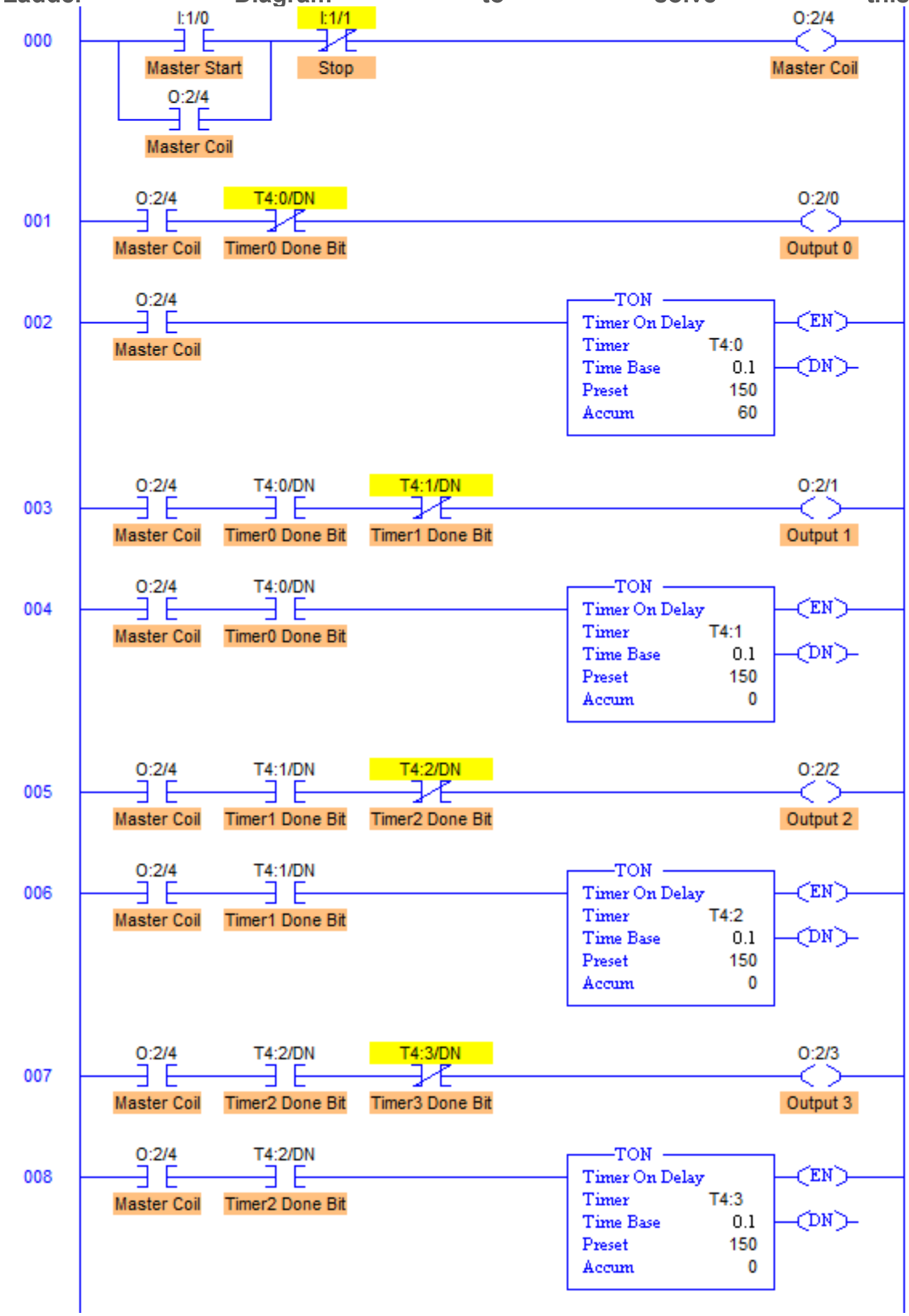
## PLC Program

Here is PLC program to Operate 4 Outputs Simultaneously with Time Delay, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Master Start           (Input)
 I:1/1   = Master Stop            (Input)
 O:2/4   = Master Coil            (Output)
 O:2/0   = Output0                (Output)
 T4:0    = Output0 Timer          (Timer)
 O:2/1   = Output1                (Output)
 T4:1    = Output1 Timer          (Timer)
```

```
O:2/2   = Output2               (Output)
T4:2    = Output2 Timer         (Timer)
O:2/3   = Output3               (Output)
T4:2    = Output3 Timer         (Timer)
```

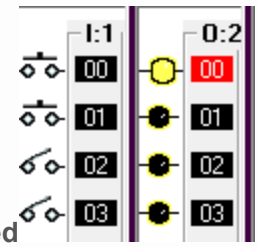
**Ladder Diagram to solve this problem**



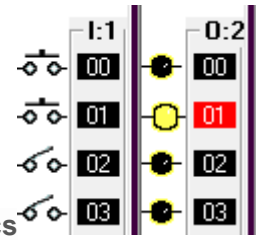

## Program Description

- RUNG000 again here is to create Master Start and Stop coil with address O:2/4.
<=”” |i=””>
- RUNG002 comprises Timer0 T4:0 with 15secs of Time Delay which is used to de-energize O:2/0 output0 after 15secs.
- When Master Start I:1/0 is momentarily pressed, output master coil O:4/4 is latched energizing the Output0 O:4/0 and activating Timer0.
- When Timer counts upto 15secs, Timer Done bit goes high turning off the Output0 and energizing Output1 O:2/1.
- This process gets repeated till the end.
- All the timers have the same preset value here so all the outputs remain ON for 15secs.
- To repeat this entire system again automatically, all the timers must be reset when Timer3 is done counting and generates T4:3/DN bit as shown in diagram below (RUNG009 of main program).

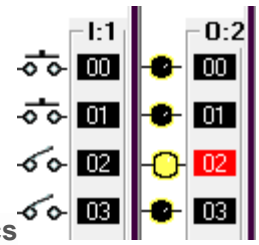

## Runtime Test Cases



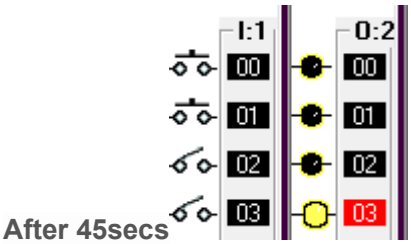

**First 15secs after Start is pressed**




**After 15secs**




**After 30secs**




# PLC Program to Implement an Automatic Car-Wash Process

This is a PLC Program to Implement an Automatic Car-Wash Process.

## Problem Description

When a Car enters the hall, a certain sequence is to be followed automatically. Steps are, 1) Soaping, 2) Washing, 3) Rinsing and 4) Drying. Implement this process sequence in PLC using Ladder Diagram programming language.

## Problem Solution

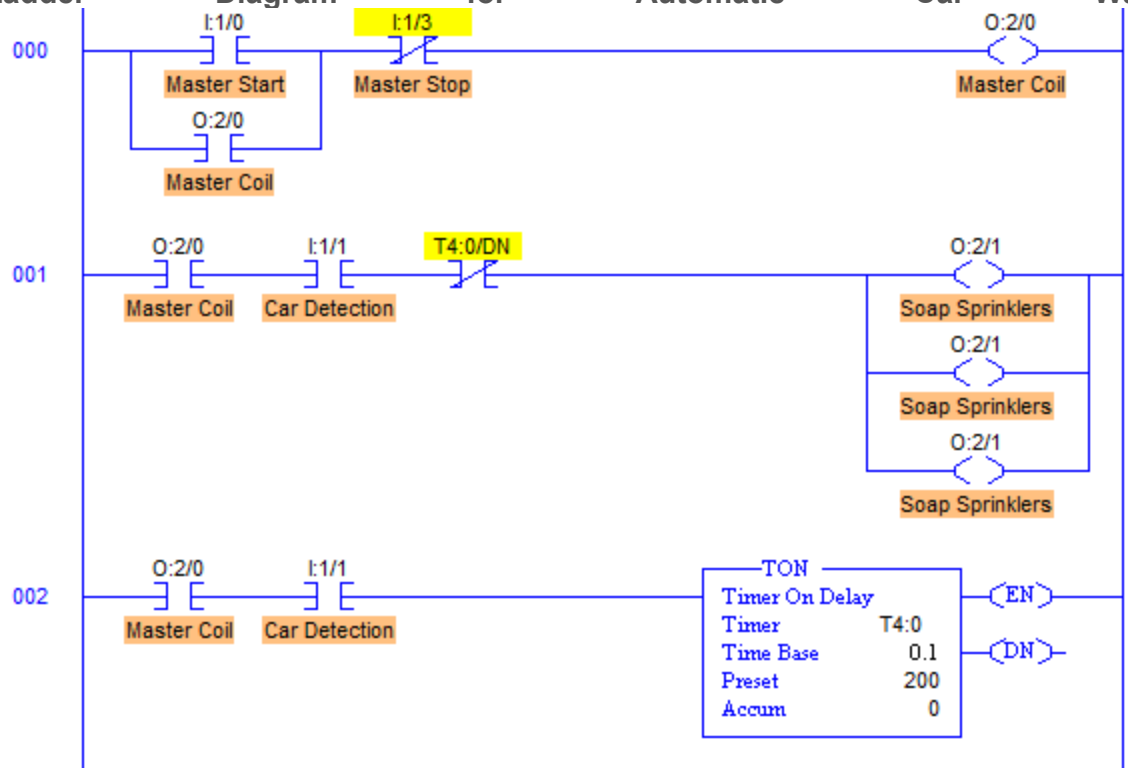
- To detect the car automatically, load cells can be used, or any other sensor such as Infrared Sensor can also be used.
- Soaping, Washing, Rinsing and Drying are performed for a particular time, hence to generate time delay for these outputs become mandatory.
- To operate this process, for soaping, washing, and drying, four different timers are used.
- IR sensor detects everything whatever restricts the signal but in load cell, particular Low Level and High Level can be set to detect heavily weighted cars only. Load Cell can be here more effective here than IR sensors.

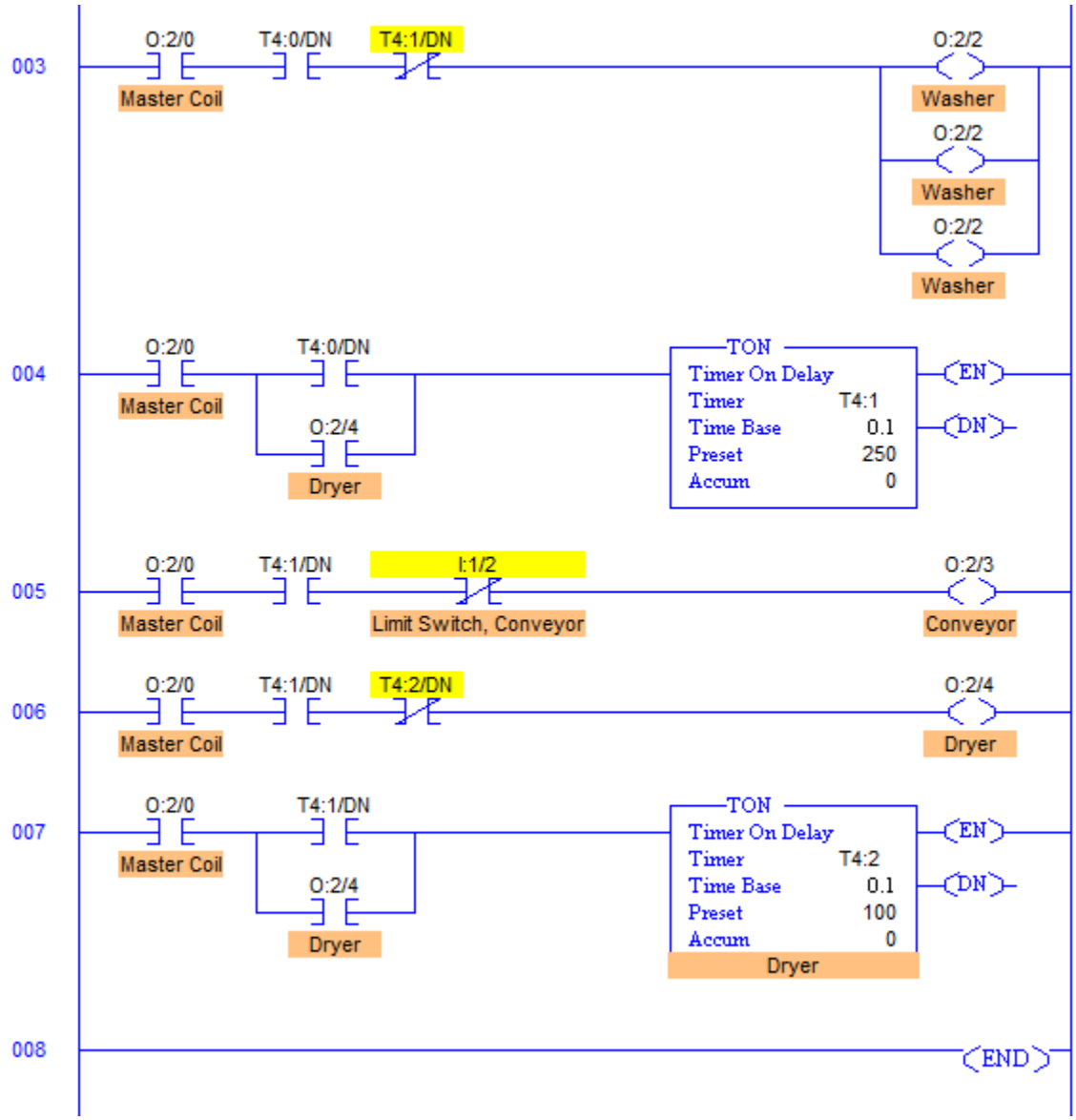
## PLC Program

Here is PLC program to Implement an Automatic Car-Wash Process, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Master Start                    (Input)
 I:1/1   =  Car Detection         (Input)
 I:1/2   = Limit Switch, Conveyor (Input)
 I:1/3   =  Master Stop                    (Input)
 O:2/0   = Master Coil                     (Output)
 O:2/1   = Soap Sprinkler         (Output)
 O:2/2   = Washer                 (Output)
 O:2/3   = Conveyor                        (Output)
 O:2/4   = Dryer                           (Output)
 T4:0    = Soaping Time                    (Timer)
 T4:1    = Washing Time                    (Timer)
 T4:2    = Drying Time                     (Timer)
```

# Ladder Diagram for Automatic Car Wash Process






## Program Description

- RUNG000: This rung is used to create a Master Coil, in other words, Master Start and Stop which controls the entire system. To activate this system, this Master Start has to be pressed.
- RUNG001 is used to activate Soaping process which is operated by a switch input I:1/1, when car is detected, Soap sprinklers are activated. And RUNG002 is used to generate a time delay which terminates the Soaping process by using XIO of T4:0/DN bit after a definite time.
- As soon as Soaping is done, RUNG003 has Washing outputs which washes the car by water sprinklers and RUNG004 is used to generate a time delay which terminates the washing process by using XIO of T4:1/DN bit after a definite time.
- RUNG005 is activated as soon as the washing process is done. It has motor coil's output to run the conveyer on which car moves to the last step of car washing, drying of car. At the end of the conveyor, a limit switch is mounted which stops the conveyor and till then Drying is done through RUNG006 and RUNG007 similar to previous operations.

## Runtime Test Cases

- You can simulate this process on LogixPro and check all the Test Cases.
- Choose I/O Simulation and simulate this program. Worst cases are checked.

# PLC Program to Maintain the Capacity of a Particular Classroom

This is a PLC Program to Maintain the Capacity of a Particular Classroom.

## Problem Description

A classroom has a capacity of maximum 120 students. There are two doors, one for Entry and the other for Exit. When number of students in the classroom is less than 120, Entry door has a Green light on it which remains ON. When number of students in the classroom is 120 or more than that, Red light goes ON turning OFF the Green light which indicates that the classroom has reached its maximum capacity and is full.

## Problem Description

- Considering the availability of two separate doors for Entry and Exit, two separate Proximity Switches can be used to detect entry and exit of students.
- One proximity switch is mounted at the Entry door and the other at the Exit door.
- Both the switches will generate two different outputs which can be then fed to PLC to operate the lights according to the Ladder Logic Program written in its memory.
- Counter must be used to count the number of students entering and exiting.
- Comparator must also be used to compare the count value with the given maximum capacity of 120.

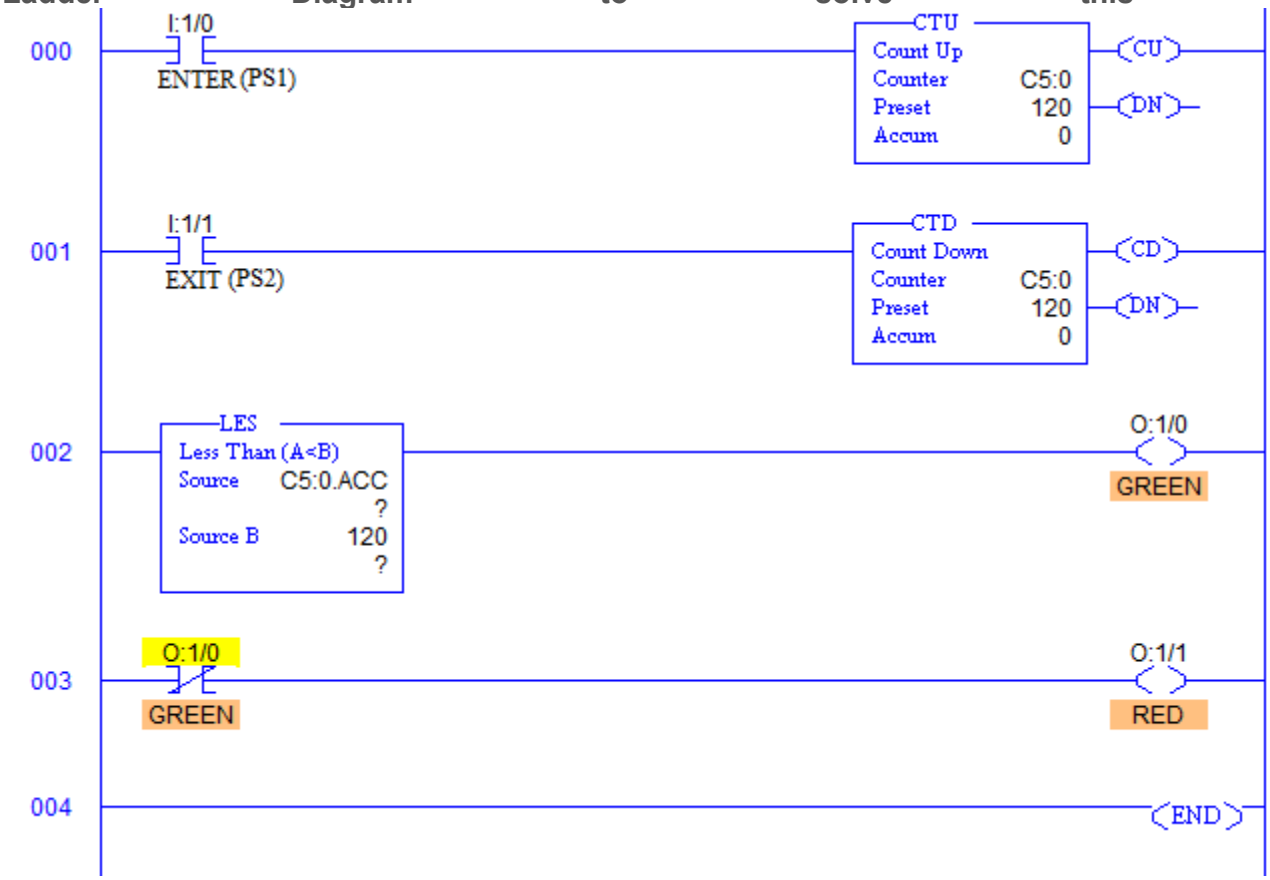
## PLC Program

Here is PLC program to Maintain the Capacity of a Particular Classroom, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0 = Proximity Switch to detect Entry of a student.           (Input)
 I:1/1 = Proximity Switch to detect Exit of a student.            (Input)
 O:1/1 = Red Light to indicate availability in the classroom      (Output)
 O:1/0 = Light to indicate classroom’s maximum capacity.  (Output)
 C5:0  = Counter to count the number of students entering.(Counter)
 C5:0  = Counter to count the number of students exiting.  (Counter)
 LES   = Comparator to compare the counter value
```

*NOTE: Note here that the address of Counter Up and Counter Down is kept same. This must be kept same in order to decrement and increment the same value in the accumulator of CTU and CTD.*

**Ladder Diagram to solve this problem**




## Program Description

- For this problem maximum limit is set to 120.
- RUNG000- Whenever a student enters in the class, Proximity Switch 1 generates a pulse which increments counter by 1.
- RUNG001- Whenever a student exits from the class, Proximity Switch 2 generates a pulse which decrements counter by 1.
- RUNG002- When the number of students in the class is less than 120, Green Light goes ON.
- RUNG003- When the number of students in the class is 120 which is the maximum capacity of the class, Red Light goes ON.
- RUNG004- It terminates the program and the scan cycle is repeated again.

## Runtime Test Cases

```
Input                    C5:0.ACC
PS1 goes Low to High      Incremented by 1
PS2 goes Low to High      Decremented by 1

Comparator                                Output
Source A(C5:0.ACC) < Source B (120)       Green=ON, Red=OFF
Source A(C5:0.ACC) = Source B (120)       Green=OFF, Red=ON
Source A(C5:0.ACC) ≥ Source B (120)       C5:0.DN bit goes true.
```

# PLC Program for a Car Parking System

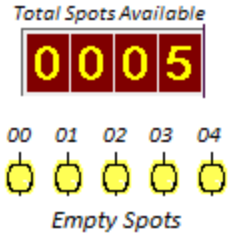
This is a PLC Program for a Car Parking System.

## Problem Description

A parking plot has total capacity of Cars. Number of empty spots are displayed on the display outside the Parking Plot and which spots are available is to be indicated by LEDs. Implement this in PLC using Ladder Diagram programming language.

## Problem Solution

- Counter is used to count the number of empty spots.
- Proximity Sensors or IR Sensors are used to detect the presence of car.
- Here in this system IR Sensor can be well installed to make this system cost efficient since Proximity Sensors are costly than IR Sensors.
- Value of counter is displayed on the display which is mounted outside the parking plot.
- This counter value is converted into decimal.



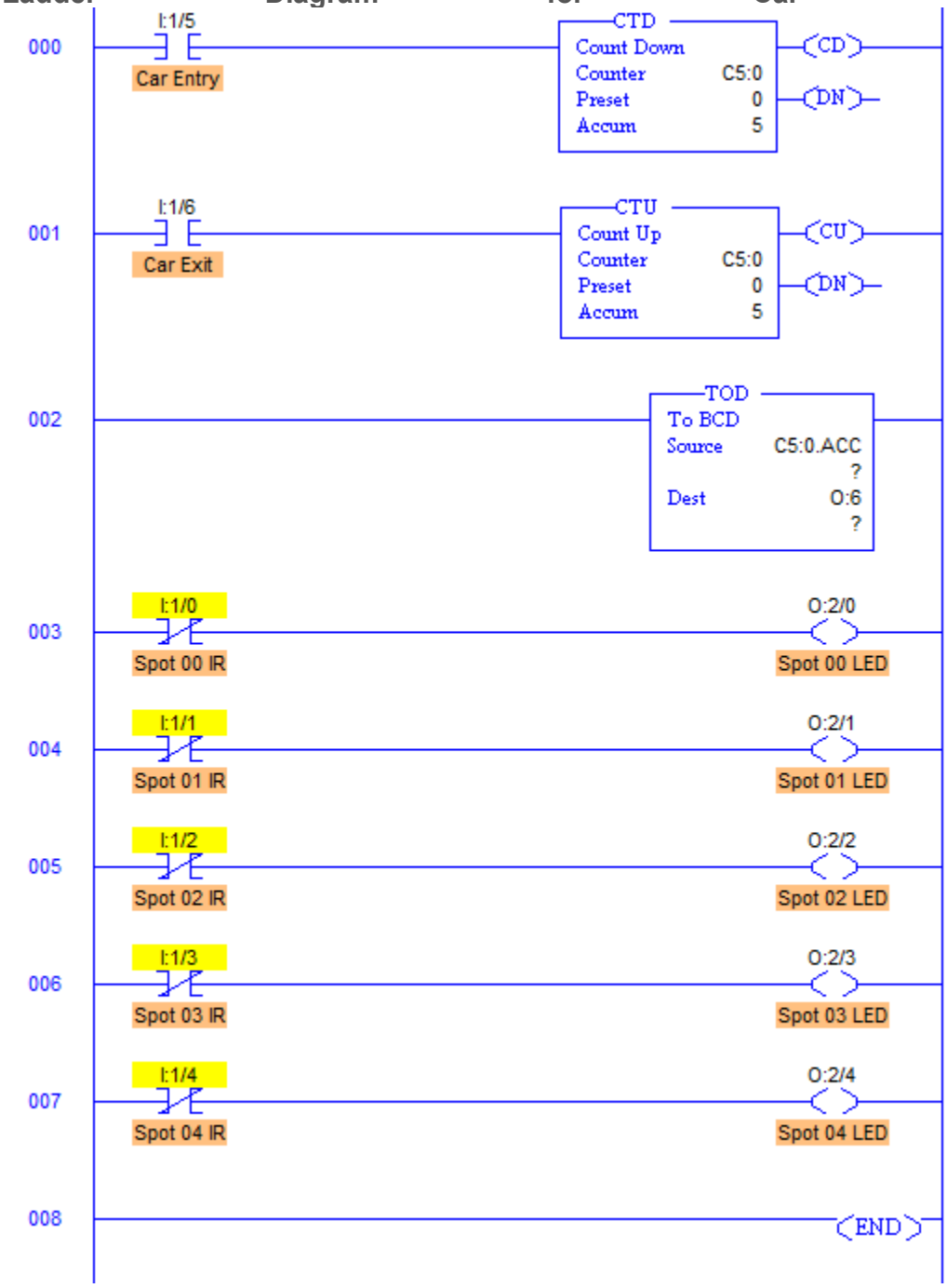

**Display arrangement**

## PLC Program

Here is PLC program for a Car Parking System, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0 to I:1/4  = IR Sensor to detect the presence of cars (Inputs)
 O:2/0 to O:2/4  = LEDs to indicate presence of car spots   (Outputs)
 C5:0            = To increment when Car exits               (Counter Up)
 C5:0            = To decrement when Car enters              (Counter Down)
 O:6             = Display address                           (Output)
```

## Ladder Diagram for Car Parking System

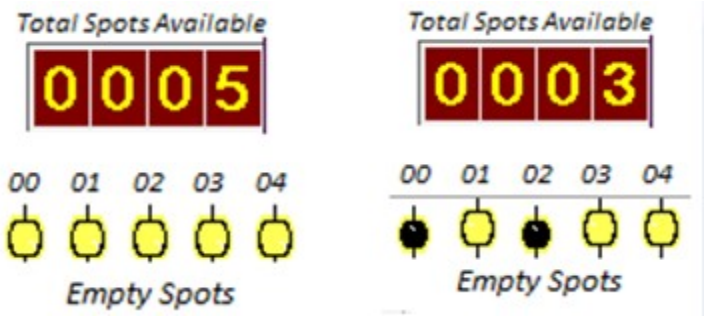



## Program Description

- Counter Up CTU and Counter Down CTD are used to determine the Exit and Entry of cars respectively.
- Value 5 is already stored in the accumulator since only 5 number of spots are there in this Parking Plot.
- So whenever car enters or exits from the Parking area, the value in the counter is incremented and decremented accordingly.
- Accumulator holds decimal values, this value thus sent to the display through BCD converter which converts Decimal digits into equivalent Binary Coded Decimal signals.
- Display receives whatever the value Accumulator holds, in terms of BCD.
- I:1/5 and I:1/6 are two inputs from other two IR Sensors to detect the exit and entry of cars accordingly.
- Here again, CTU and CTD both have the same address in order to vary accumulator value of both counters according to Exit and Entry of cars.

- XIO (Normally Closed) contact is used here for IR Sensor outputs so that LED is ON when the spot is empty.

## Runtime Test Cases




# PLC Program for Burglar Alarm Security System

This is a PLC Program for Burglar Alarm Security System.

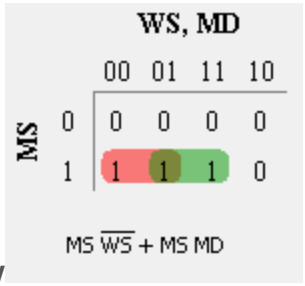
## Problem Description

Consider the design of a Burglar Alarm for a house. This alarm will be activated if an unauthorized person is detected by a Window Sensor or a Motion Detector. Implement this Alarm System in PLC using Ladder Diagram programming language.

## Problem Solution

- Basically two sensors are used, one is Motion Detector and other one Window Sensor. Window sensor is nothing but a loop of wire that is a piece of thin metal foil which encircles the window.
- The motion detector is designed such that when a person is detected, the output of sensor goes true.
- Important thing to note here is that in Window Sensor, current is always passing until there is a breakage in glass of a window. Hence output is always true. When alarm system is active and someone tries to break the window, current does not flow through the metal foil causing output to go false.
- Karnaugh-Map can be used here to solve the equation and then to implement its logic in Ladder Diagram.

**Solved K-Map and its equation is shown below**



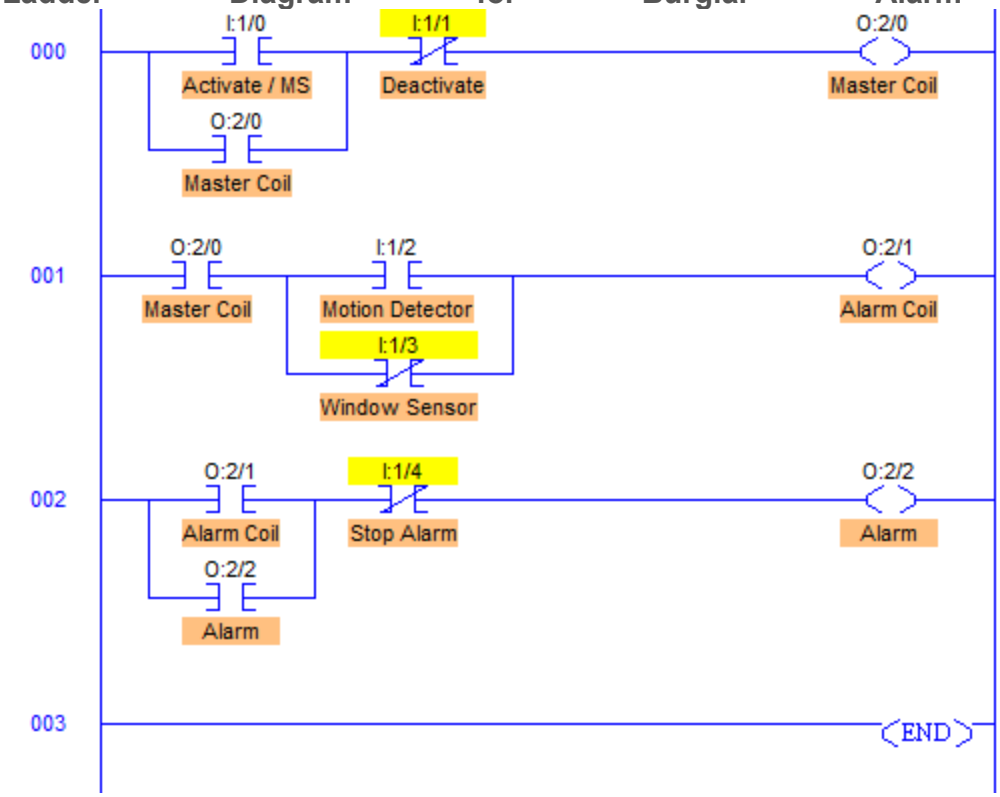

*Note: This security system is normally not implemented using PLC since the hardware cost of the entire system will be effectively high. Purpose of this problem is just to learn how to implement such logics in PLC.*

## PLC Program

Here is PLC program for Burglar Alarm Security System, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   = Master Switch                          (Input)
 I:1/1   = Deactivate system                      (Input)
 I:1/2   = Motion Detector                        (Input)
 I:1/3   = Window Sensor                          (Input)
 I:1/4   = Button to Stop Alarm (not the system)  (Input)
 O:2/0   = Master Coil                            (Output)
 O:2/1   = Alarm Coil                             (Output)
 O:2/2   = Alarm                                  (Output)
```

**Ladder Diagram for Burglar Alarm Security System**




## Problem Description

- RUNG000 simply shows a latching of a coil O:2/0 to activate the entire security system.
- When Activate button I:1/0 is pressed momentarily, the security system is activated.
- If the system is not activated, alarm does not indicate anything since sensors will have no effects on the Alarm Coil O:2/1.
- When system is active and Motion detector detects a person, the alarm coil will momentarily go high activating the Alarm O:2/2 which stays ON until I:1/4 is pressed manually.
- As we can see in RUNG001 that XIO (Normally Closed) contact is used for Window Sensor input I:1/3 because it is normally in true state when not activated. So when the breakage of a window is detected, it goes false from its true condition allowing Alarm Coil O:2/1 to go high for a moment which in turn activates Alarm O:2/2.
- In RUNG002, latching has to be provided in order to keep the alarm ringing even if the detection by the sensors are momentary or to be accurate, pulsating.

## Runtime Test Cases

```
Master  Window   Motion   Alarm
Switch
0       x        x        0
0       x        x        0
```

```
0       x       x       0
0       x       x       0
1       0       0       1
1       0       1       1
1       1       0       0
1       1       1       1
```

# PLC Program to Latch and Unlatch Output With Time Delay

This is a PLC Program to Latch and Unlatch Output With Time Delay.

## Problem Description

Implement Latching and Unlatching of output with a particular time delay in PLC using Ladder Diagram.

## Problem Solution

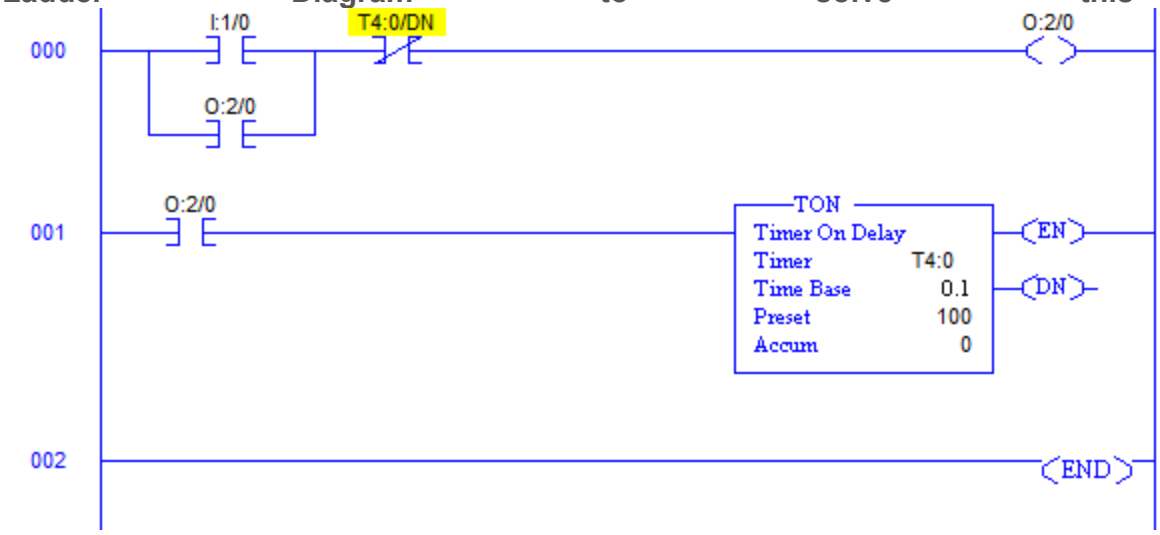
- After solving previous problem, we know how to Latch and unlatch a coil.
- Previous problem can be modified to solve this problem.
- There are total three types of Timers available known as TON, TOFF and RTO timers.
- TON -Timer On
- TOFF -Timer Off
- RTO -Retentive Timer On
- TON timer is used here to solve this problem.

## PLC Program

Here is PLC program to Latch and Unlatch Output With Time Delay, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0           = Start Push Button      (Input)
 T4:0/DN = Timer 4.0 done bit     (Input)
 O:2/0           =                        (Output)
 T4:0            =                        (Timer)
```

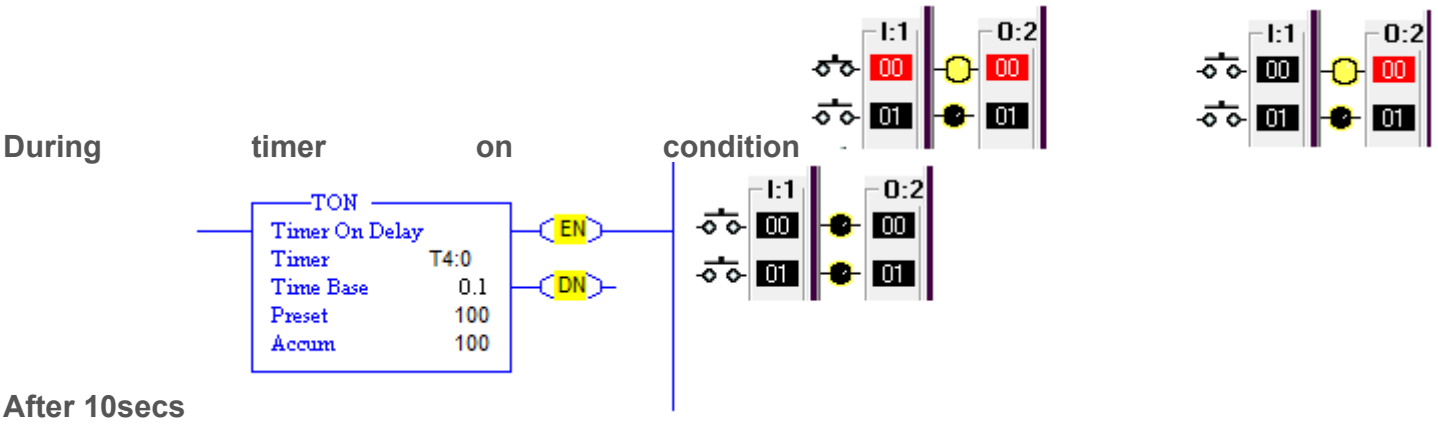
**Ladder Diagram to solve this problem**




## Program Description

- Latching is done the same way as in previous program. The only difference in this program is that Timer On is added to unlatch the output after 10secs. And to operate unlatching of output, instead of using stop PB, Timer On done bit passed to Stop XIO contact which operates automatically.
- Time Delay to turn output off can be decided by operator in terms of Multiplication of Time Base and Preset. It means if time base is 0.1 and we want to generate 10secs of time delay then Preset value must be set to 100 so that “100 * 0.1 = 10secs”.
- When input I:1/0 is pressed momentarily, output goes true and it is latched.
- When output is latched, Timer T4:0 is enabled and EN bit goes high.
- As long as O:2/0 is true, Timer T4:0 is active and accumulator value is incremented.
- When Timer count (Accumulator value) is equal to Preset value, Timer Done Bit T4:0/DN goes high which works as a Stop PB of previous problem causing output to go false.
- Important thing to note here is that when timer is ON and you press Start PB as soon as Time delay is over, output is latched again. So during those 10secs, Start PB has no effect on output at all.

## Runtime Test Cases




# PLC Program to Latch and Unlatch an Output by Sealing

This is a PLC Program to Latch and Unlatch an Output by Sealing.

## Problem Description

Implement Latching and Unlatching of Output without using manufacturer provided Latching and Unlatching Instructions. That is called Latching by Sealing of contacts.

## Problem Description

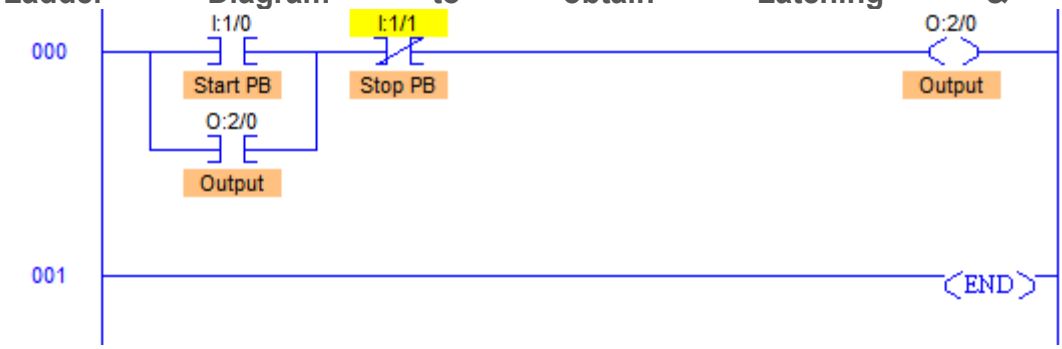
- In many PLCs, Latching and Unlatching instructions are not available to latch and unlatch outputs directly.
- To achieve this, Seal in contact must be used to obtain similar level of Latching.
- This technique of latching is called Latching by Seal in contact.
- This seal in contact must be used in parallel with the main input Start Push Button and address of this parallel contact must be same as Output of which we want to latch.
- By performing this simple technique, we can obtain Latching of output that of similar to Latch/Set/Keep and Unlatch/Reset/Bit instructions of Allen Bradley/Siemens/Omron PLCs respectively.

## PLC Program

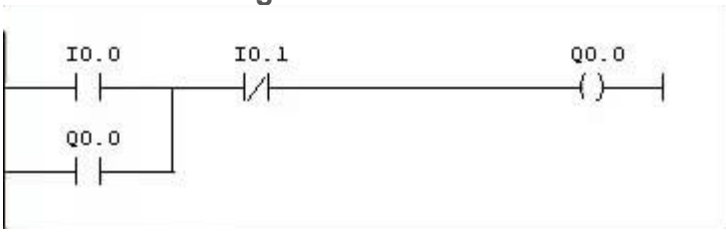
Here is PLC program to Latch and Unlatch an Output by Sealing, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 I:1/0   =       Start Push Button (Input)
 I:1/1   =       Stop Push Button (Input)
 O:2/0   =       Output (Input & Output)
```

**Ladder Diagram to obtain Latching & Unlatching Fig.1**



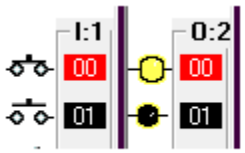

**Ladder Diagram to obtain Latching & Unlatching Fig.2**

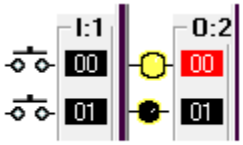


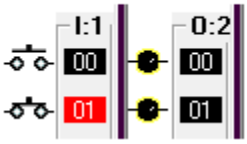

## Program Description

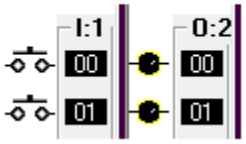
- Fig.1 shows latching of output for Allen Bradley PLCs and fig.2 shows similar operation for Siemens PLCs. Similarly in Omron, same contacts may be used to obtain this operation with the only difference of Input-Output addresses.
- There are two types of Scanning Order: Horizontal and Vertical. Allen Bradley, Siemens, Omron and such PLCs come with Horizontal Scanning order and PLCs such as AEG Modicon come with Vertical Scanning order.
- As we now know from the theory of PLC Programming using Ladder Diagram that in most of the PLC manufacturers provide Horizontal Scanning order, input and output image tables are updated accordingly.
- So here when Start PB is pressed, due to the sequence of Ladder Logic in this program, it immediately latches the output with its own bit stored in Output Image Table due to Horizontal Scanning order.
- Stop PB, which is XIO/Normally Closed contact, simply breaks the current path causing the Output to go false from its latched condition.
- Similarly it works in Siemens and Omron PLCs as well.

## Runtime Test Cases










# PLC Program to Simply Latch and Unlatch an Output

This is a PLC Program to Simply Latch and Unlatch an Output.

## Problem Description

Perform Latching and Unlatching of an output (Pilot Light, Motor, Solenoid Coil etc.).

## Problem Solution

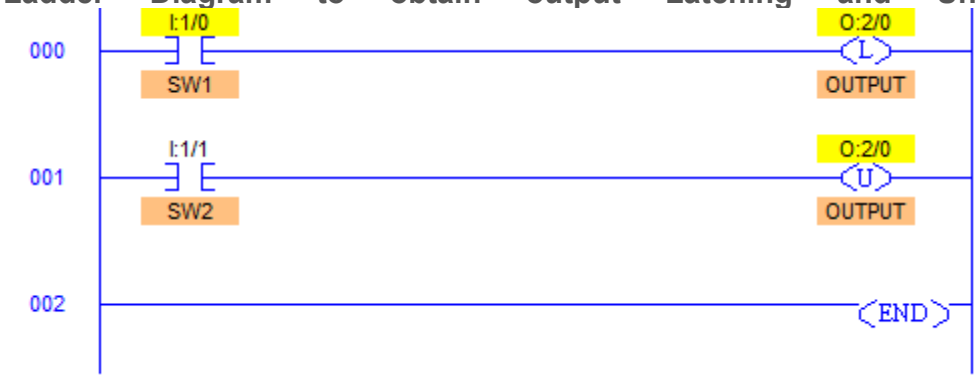
- Almost all the PLCs come along with Latch and Unlatch instructions inbuilt.
- They all must be used in two rungs in order to operate an output.
- Different manufactures have provided different names to these Instructions.
- Allen Bradley PLCs have -(L)- for Latching and -(U)- for Unlatching instructions, Siemens PLCs have (S) for Set and (R) for Reset instructions. Similarly, Omron PLCs have KEEP instruction for latching.
- All these instructions perform the same task of Latching when one input is pressed and Unlatching when another input is pressed.
- Two Push Buttons are required to perform this task.
- Any output to be Latched is required which is connected physically as an output device.

## PLC Program

Here is PLC program to Simply Latch and Unlatch an Output, along with program explanation and run time test cases.

```
List of Inputs & Outputs
 SW1 = Switch1                  I:1/0 & 0.00     (Input to Latch)
 SW2 = Switch2                  I:1/1 & 0.01     (Input to Unlatch)
 Bit stored to latch            100.00           (Stored Bit Address)
 Output                         O:2/0 & 200.00   (Output Light)
```

**Ladder Diagram to obtain output Latching and Unlatching in Allen Bradley**

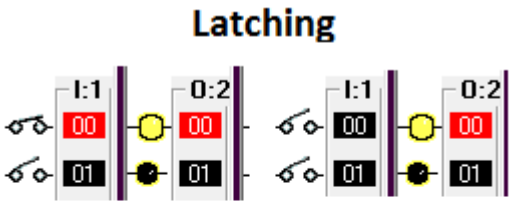


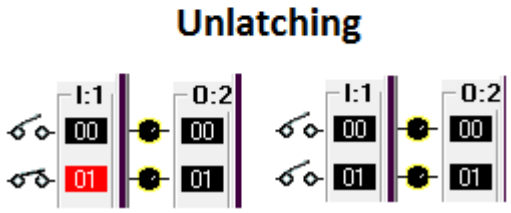

**Ladder Diagram to obtain output Latching and Unlatching in Omron**


## Program Description

- first figure shows how programming of Latching and Unlatching is done in Allen Bradley PLCs and 2nd shows how Latching and Unlatching is done in Omron PLCs.
- Siemens PLCs and Allen Bradley have similar instruction types. The only difference is that Latch is replaced by Set and Unlatch is replaced by Reset. Addressing varies as well.
- In first figure, when SW1 is pressed momentarily, the Output goes True even if the input from SW1 is withdrawn, this happens because whenever the input SW1 is pressed, O:2/0 stores the bit and output remains True even if from Switch SW1 is withdrawn.
- This output goes False only when the switch SW2 is pressed resetting the O:2/0 bit which Unlatches the Output.
- In 2nd figure, as we can see, Omron PLCs use KEEP instruction which is a single rung instruction comprising of two different inputs, one to set the bit and other to reset the bit. Latching and Unlatching is here done by KEEPing the bit and accordingly Output is Latched and Unlatched.

## Runtime Test Cases




# PLC Program to Operate Seven Segment Display

This is a PLC Program to Operate Seven Segment Display.

## Problem Description

Implement displaying 0-9 digits in 7 Segment LED Display interfacing with PLC using Ladder Diagram programming language.

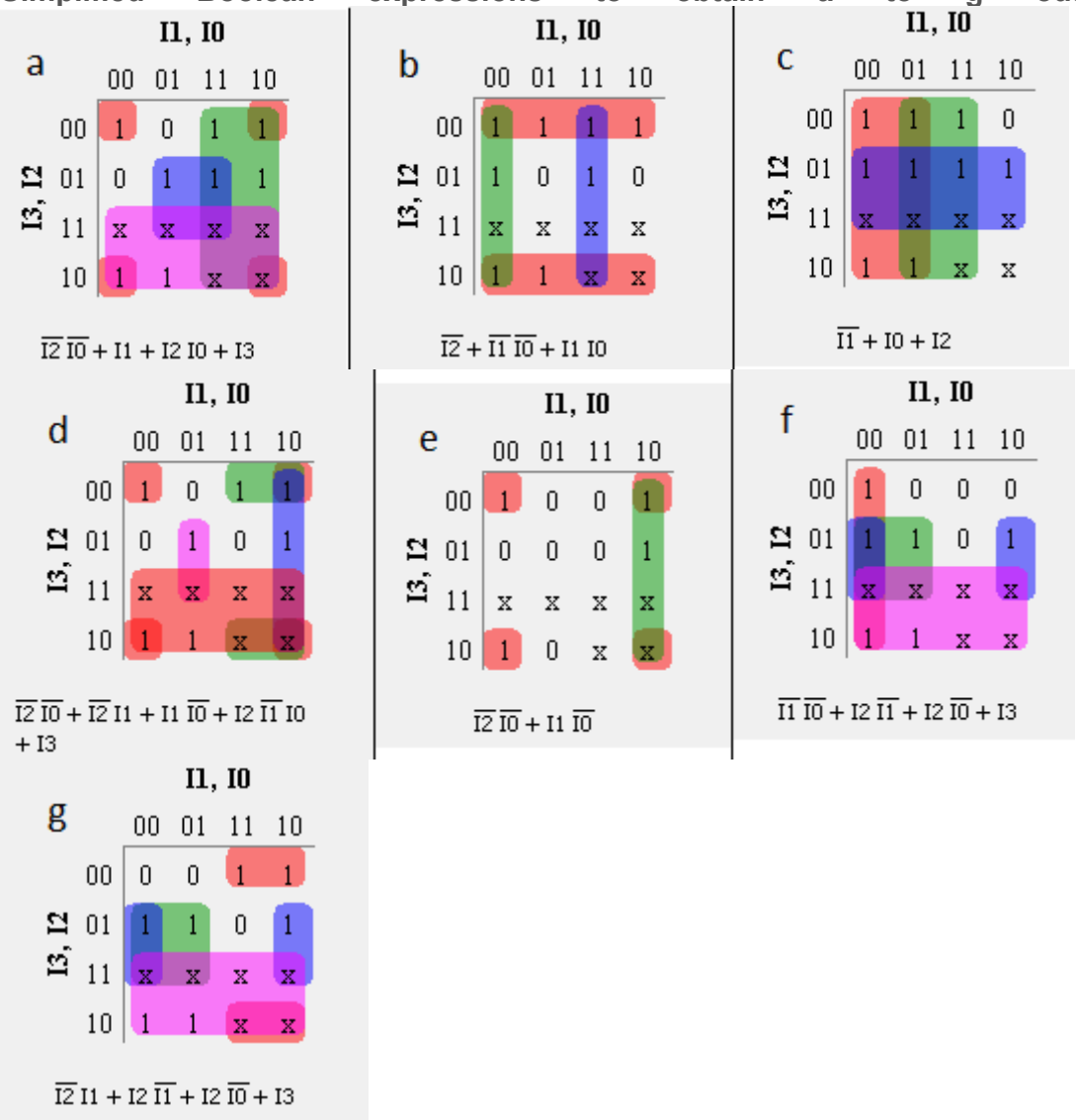
## Problem Solution

- 7 Segment LED Displays are also known as BCD to 7 Segment decoder.
- These displays are readily available or can be made easily organizing simple LEDs in the same structure as in actual display.
- Input to this display is BCD number which are decoded into digits 0 to 9.
- Connect this display with Output card of a PLC.
- Write a truth table to energize various outputs according to BCD input applied.
- This displays do not come with internal latches, we can latch inputs in PLC using either seal in contact or Latch-Unlatch instructions in Allen Bradley and Set-Reset instructions in Siemens PLCs.
- Use K-Map to obtain output equations.
- Implement this equations in Ladder Diagram format using AND and OR operations.

### Truth Table of 7 Segment Display

```
Inputs   Outputs
         a b c d  e f g    Display
0000     1 1 1 1  1 1 0    0
0001     0 1 1 0  0 0 0    1
0010     1 1 0 1  1 0 1    2
0011     1 1 1 1  0 0 1    3
0100     0 1 1 0  0 1 1    4
0101     1 0 1 1  0 1 1    5
0110     1 0 1 1  1 1 1    6
0111     1 1 1 0  0 0 0    7
1000     1 1 1 1  1 1 1    8
1001     1 1 1 1  0 1 1    9
```

**Simplified Boolean expressions to obtain a to g outputs using K-Map**

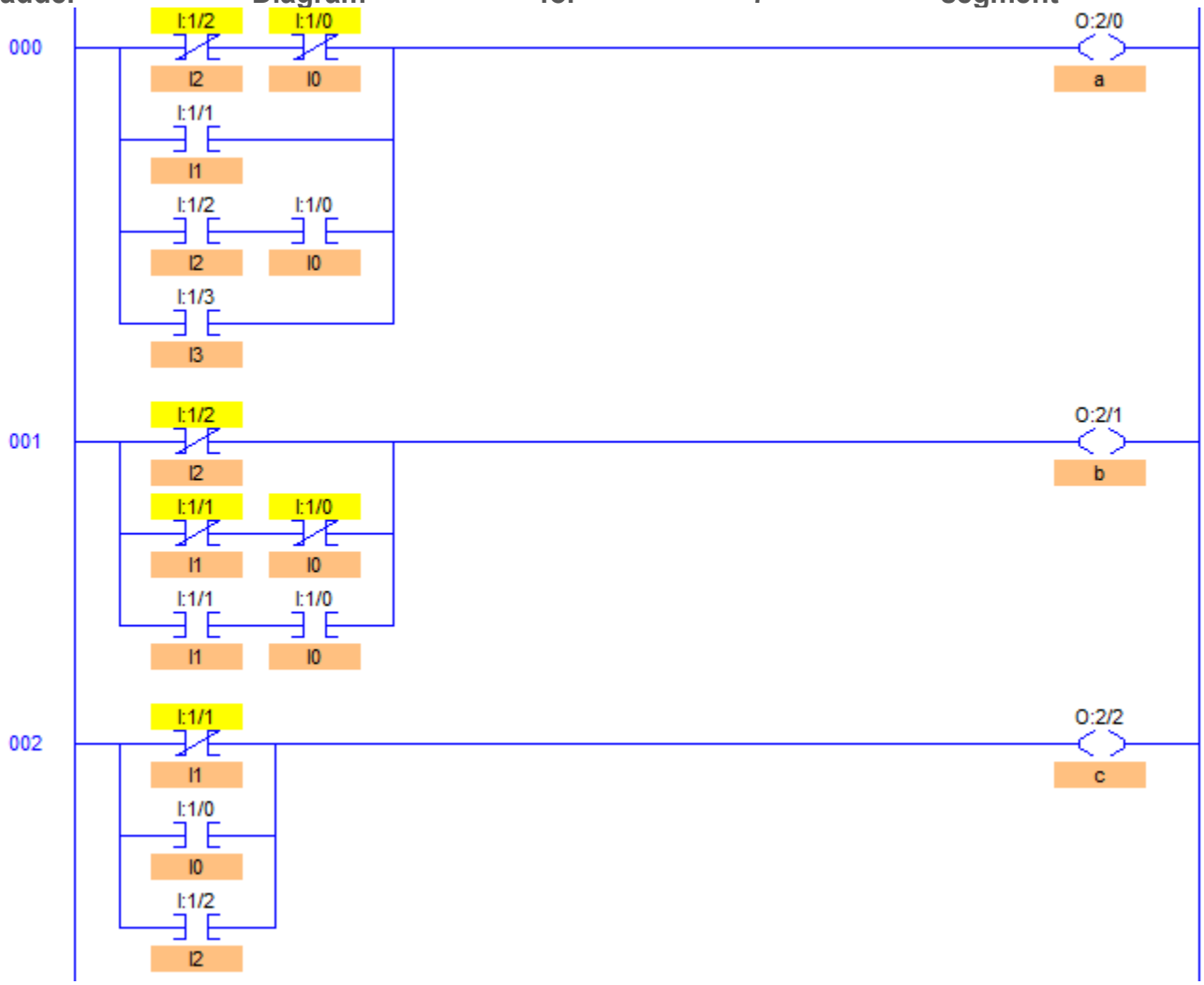


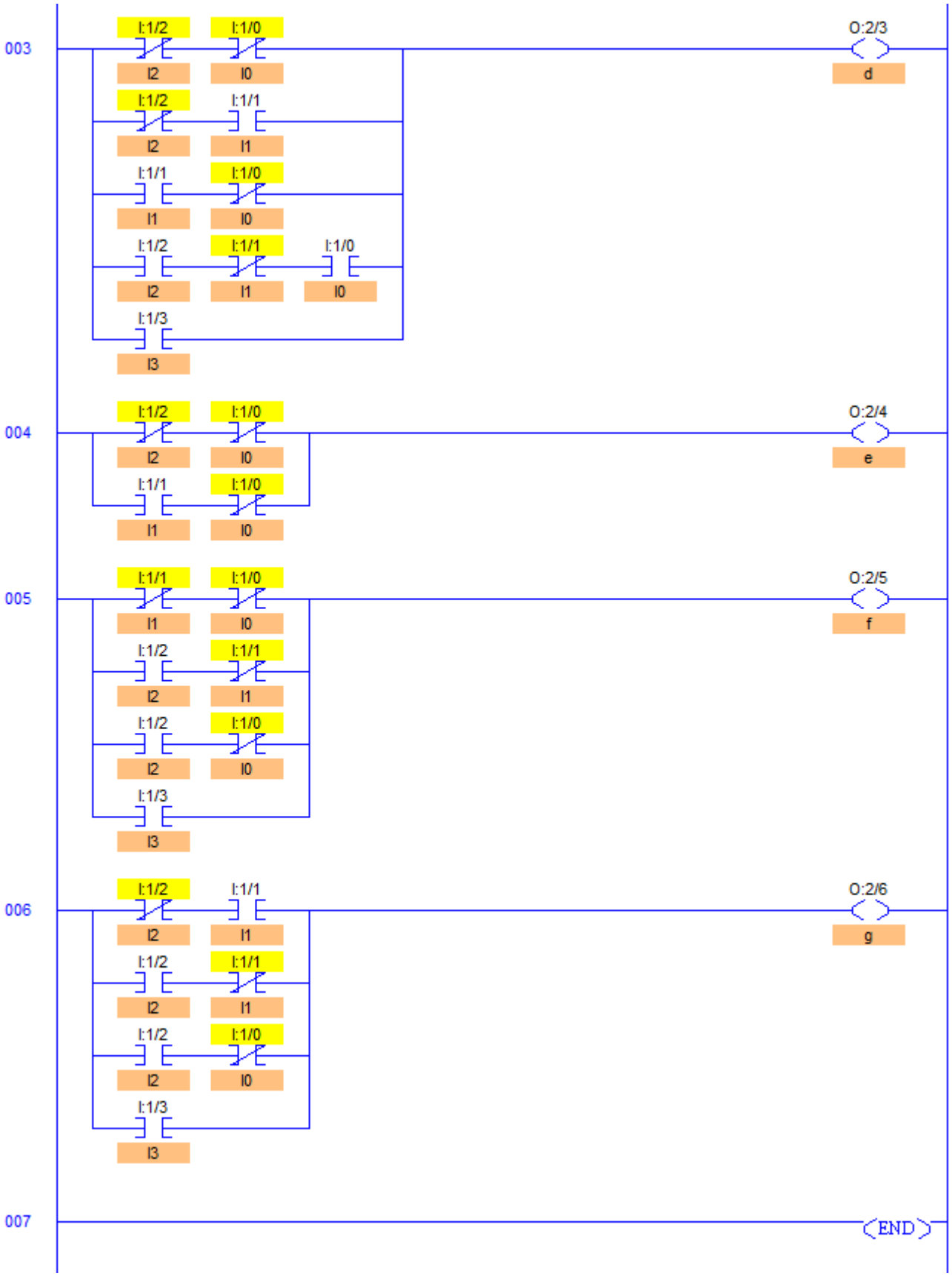

## PLC Program

Here is PLC program to Operate Seven Segment Display, along with program explanation and run time test cases.

```
List of Inputs and Outputs
I:1/0 to I:1/3   = I0 Input 0 to I3 Input 3 respectively.  (Input)
O:2/0 to O:2/6   = a to g respectively.                    (Outputs)
```

Ladder Diagram for 7 segment display





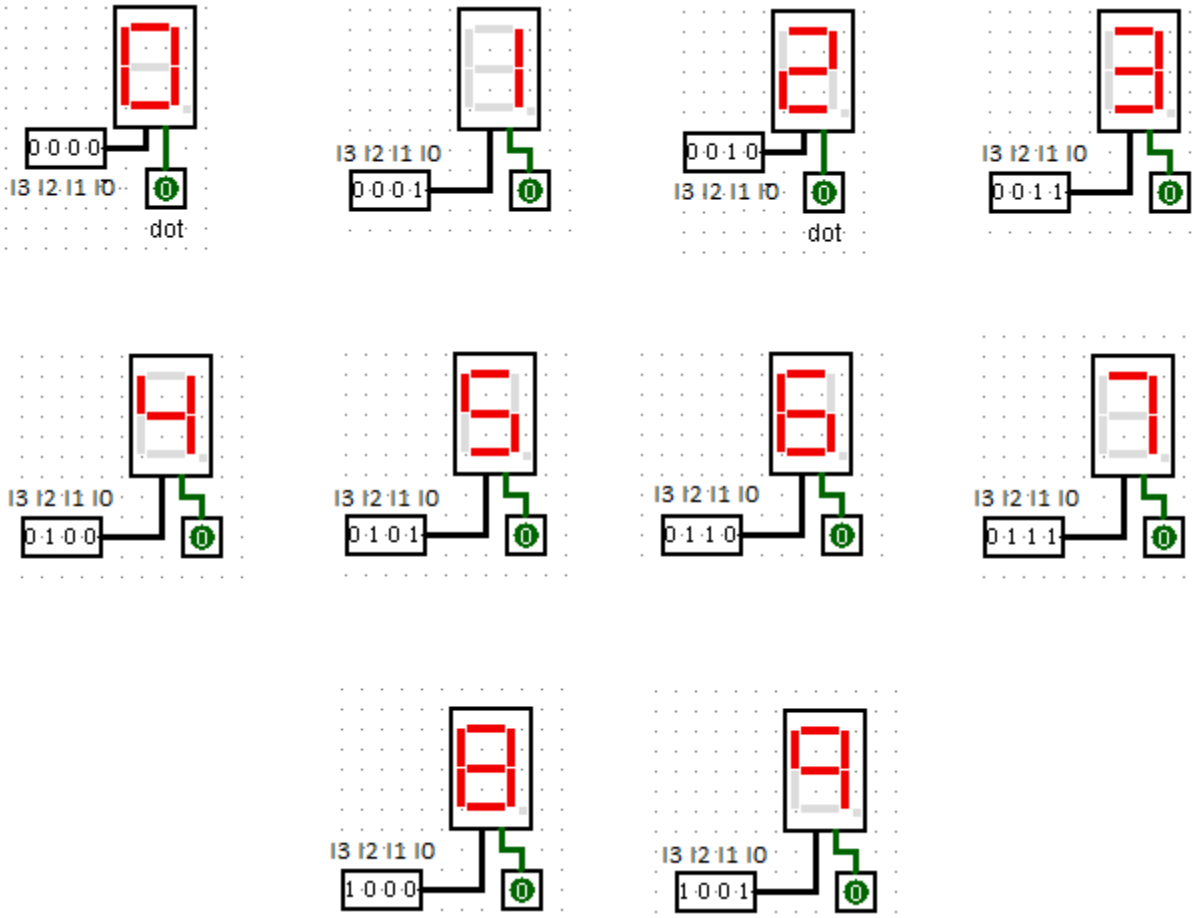

## Program Description

- After performing K-Map simplification method, equations for all the 7 outputs are obtained.
- I:1/0 to I:1/3 are the inputs I0 to I3 respectively and O:2/0 to O:2/6 are the outputs ‘a’ to ‘g’ respectively.

- Output ‘a’ 0:2/0 energizes in 4 cases, when I2 and I0 are low, I1 is high, I0 and I2 are high or when I3 is high.
- Output ‘c’ O:2/2 energizes in 3 cases, when I1 is low, I0 is high, or when I2 is high.
- Similarly all the remaining outputs b, d, e, f and g are obtained.
- Simply this is a method to convert BCD inputs into Seven Segments to display 0-9 digits.

*Separated by commas are different cases of outputs being energized.*

## Runtime Test Cases




# PLC Program to Implement T Flip Flop

This is a PLC Program to Implement T Flip Flop.

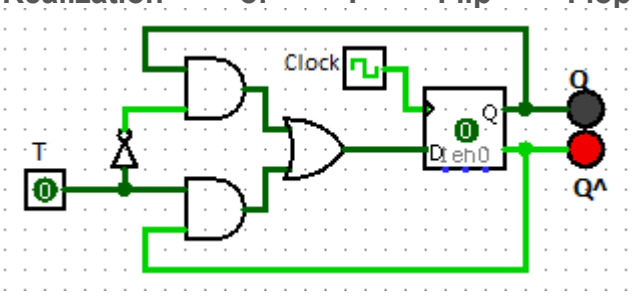
## Problem Description

Implementing T(Toggle) Flip Flop in PLC using Ladder Diagram programming language.

## Problem Solution

- The T Flip Flop has two inputs.
- One is the Clock Input and the other one is T input. Clock input initiates the Flip Flop action. The second input T input enables or disables the trigger operation. Circuit diagram of T Flip Flop is shown in figure below.

**Realization of T Flip Flop using gates and D Flip Flop**




**Truth table for the T Flip Flop**

```
T  Clock Qn Qn+1
0  0     X  Qn
0  1     X  Qn
1  0     Q  Qn
1  1     Q  Q^
```

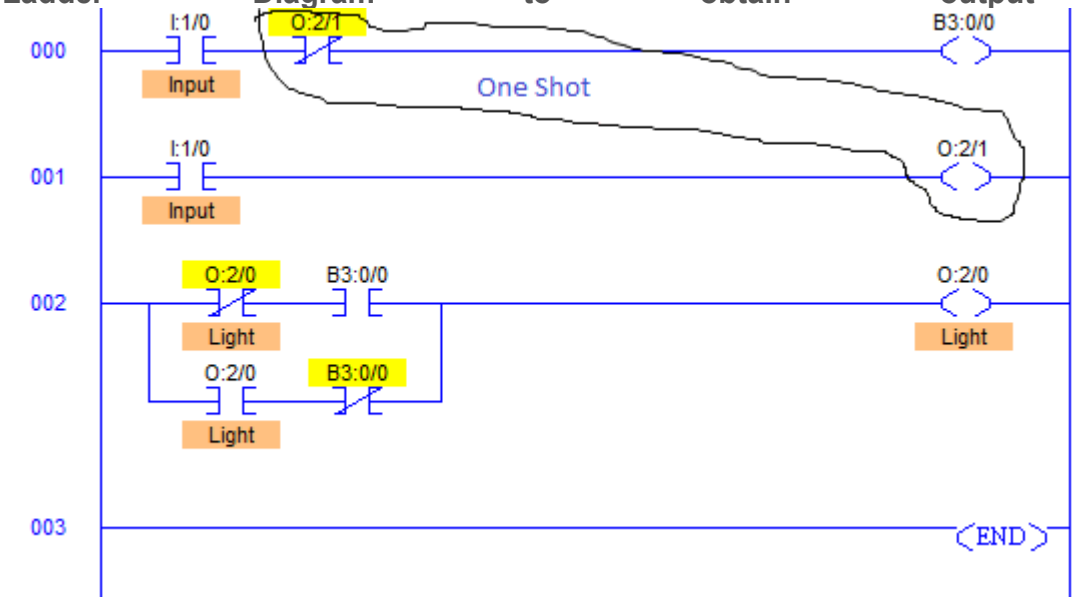
- A High state in the clock column indicates that the clock generates square waves making a 0 to 1 and 1 to 0 transitions.
- The Qn column is the state of the flip flop prior to the clock application and the Qn+1 column is the state of the flip flop after the clock.
- An x indicates don't care condition.
- Ladder Logic Diagram of this can be obtained by creating One Shot of T Input.
- This can be done in PLCs such as Allen Bradley directly by using OSR instruction. OSR stands for One Shot Rise which when input is given, it simply triggers an event to occur one time.
- Many PLCs do not have this instruction. By adding one more extra relay or by storing bit status keeping in mind the scan cycle and order of rungs in programming, One Shot can be obtained.

# PLC Program

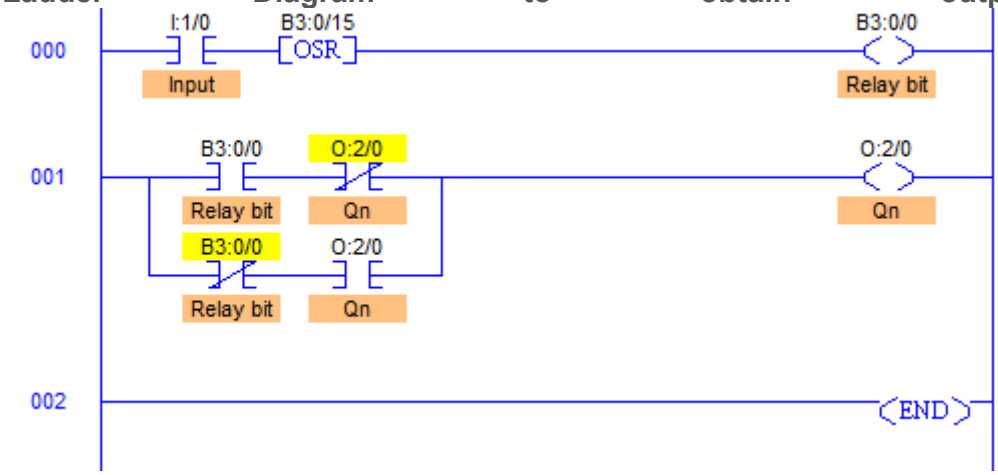
Here is PLC program to Implement T Flip Flop, along with program explanation and run time test cases.

```
List of Inputs and Outputs
 T(Toggle)         =     I:1/0    (Input)
 Qn Output/Light   =     O:2/0    (Output)
 Relay Bit         =     B3:0/0   (Bit 0 Output)
 One Shot          =     O:2/1    (OS logic Output)
 OSR Bit   =     B3:0/15 (OSR instruction)
```

Ladder Diagram to obtain output Figure 1




Ladder Diagram to obtain output Figure 2




*Note: Assuming Clock is always present, so it only depends on the state of T input as stated in Truth Table of T Flip Flop above.*

## Program Description

- I:1/0 is T input which is used to toggle the output.
- We are assuming that the clock is always present, so output is dependent only on T input I:1/0.
- In the Ladder Diagram Figure. 1, One Shot is created by ordering rungs such a way that during first scan cycle when T input I:1/0 is triggered, Bit B3:0/0 goes high immediately setting output image table of B3:0/0 to 1.
- During first scan cycle, it first scans the RUNG000 and then RUNG001.
- When the RUNG002 is scanned, it receives input “1” from the image table which was set while scanning RUNG000. It triggers output O:2/0.
- As soon as output O:2/0 goes high, it is sealed.
- During the second scan cycle, when RUNG000 is scanned, status of O:2/1 is observed “1” which breaks the flow in RUNG000 resetting image table of memory bit B3:0/0 to “0”.
- This does not occur output O:2/0 to be de-energized due to latching logic.
- Even if the T input is withdrawn, the output O:2/0 does not change.
- When T input I:1/0 is again triggered, it repeats the same scan cycle sequence De-Energizing output O:2/0.

- The only difference in Ladder Diagram of Figure.1 and Figure.2 is that in Figure.1, one extra rung is added to create One Shot but the same rung is eliminated in Figure.2 by using One Shot Rise instruction.
- Operation remains the same.

## Runtime Test Cases

```
T Clock  Qn Qn+1  Comments
0 0      X  Qn    Clock is not present
0 1      X  Qn    Unchanged output
1 0      Q  Qn    Output Toggles
1 1      Q  Q^    Output Toggles
```

# PLC Program to Implement D Flip Flop

This is a PLC Program to Implement D Flip Flop.

## Problem Description

Implementing D Flip Flop in PLC using Ladder Logic programming language.

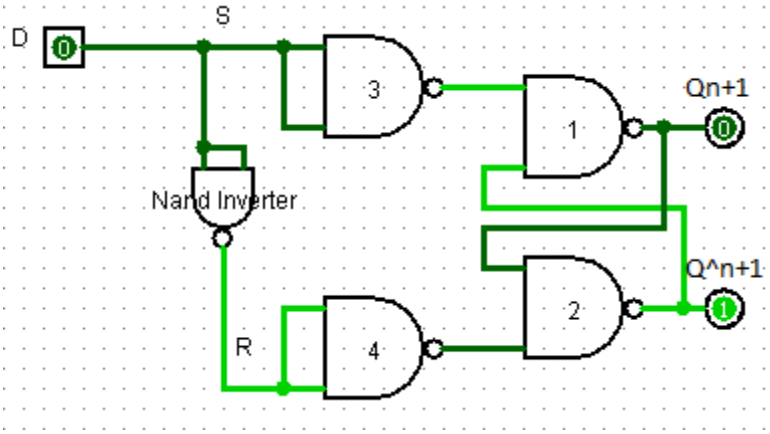
## Problem Solution

- This latch has only 1 input denoted as D.
- D latch is the simple gated S-R latch with a NAND inverter connected between its S and R inputs.
- In S-R Flip Flop when S=R=0 or S=R=1, the outputs Q and Q^ either don't change or they are invalid (indeterminate) due to race condition.
- This disadvantage of S-R latch can be overcome by using the D latch.
- As we can see in the diagram below, S and R inputs will always be the complements of each other. Hence S = R = 0 or S = R = 1 condition will never occur. This will avoid the problems associated with S-R=0-0 and S-R=1-1 conditions.
- Truth table can be obtained as given below

**Truth Table for the D Latch**

```
Inputs   Outputs          Comment
D        Qn+1     Q^n+1
0        0        1        Reset Condition
1        1        0        Set Condition
```

**Realization of D Latch using Logic Gates**

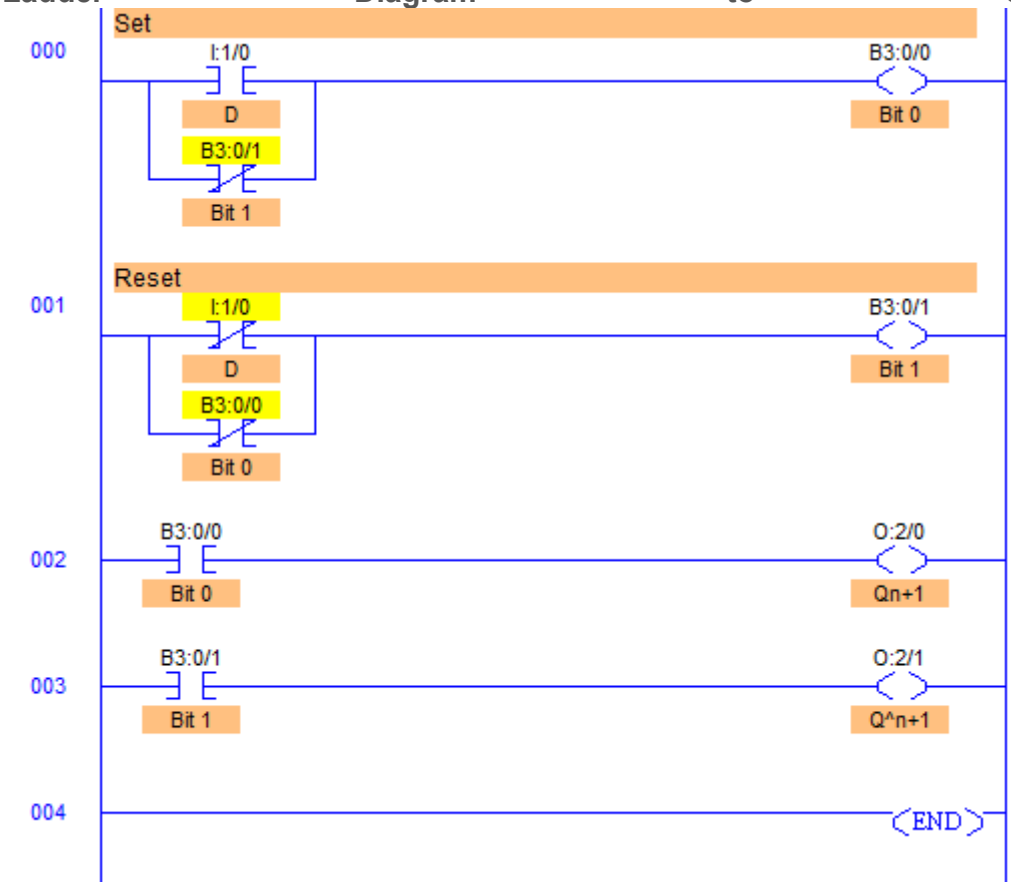



## PLC Program

Here is PLC program to Implement D Flip Flop, along with program explanation and run time test cases.

```
List of Inputs and Outputs
D (Set)   =       I:1/0    (Set Input)
D (Reset)=        I:1/0    (Reset Input)
Qn+1      =       O:2/0    (Q Output)
Q^n+1     =       O:2/1    (Q^ Output)
Bit 0     =       B3:0/0   (Bit 0 output)
Bit 1     =       B3:0/1   (Bit 1 output)
```

**Ladder Diagram to obtain output**




## Program Description

- By definition, a condition of Qn+1 = 1 and Q^n+1 = 0 is Set and a condition of Qn+1 = 0 and Q^ n+1 = 1 is Reset.
- As we can see from the circuit diagram as well as in the ladder diagram, the only difference between S-R latch and D Latch is that it uses it uses inverted value of S.
- So as we can see in the ladder diagram, R is replaced by inverted input of D (Set) I:1/0 and denoted as D (Reset) I:1/0.
- During power up, Q^n+1 (O:2/1) will go high because of its order scan cycle follows.
- In a D latch, we can simply say that when we activate the D input I:1/0 sets the circuit, and when we de-activate the D input I:1/0 resets the circuit.
- Both signal states can never be set to high as the Reset input is replaced by Inverted input of D I:1/0, so Race Condition is eliminated.
- Slight delay may occur in inputs and resulting changes in outputs due to PLC's program scan time which is unobtrusive.

## Runtime Test Cases

```
Inputs   Outputs          Comment
D        Qn+1 Q^n+1
0        0    1           Reset Condition
1        1    0           Set Condition
```

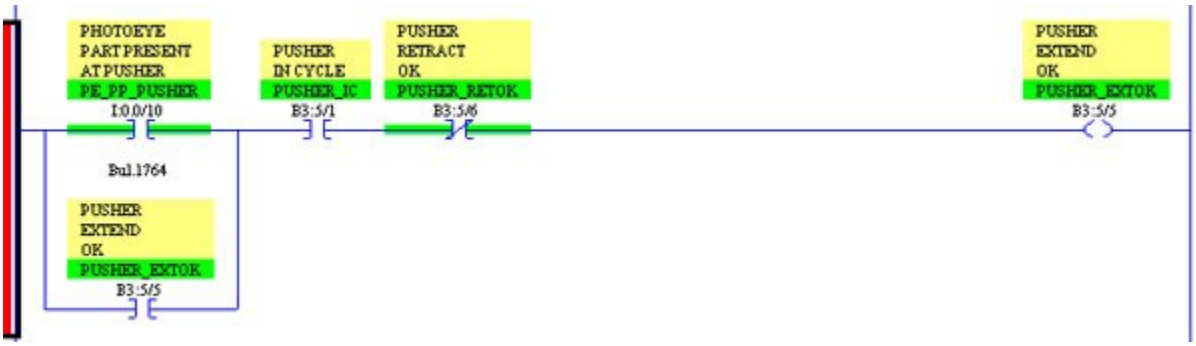
# Programming Application - Photoeye Make and Break Logic (Debounce)

Article ID: 15

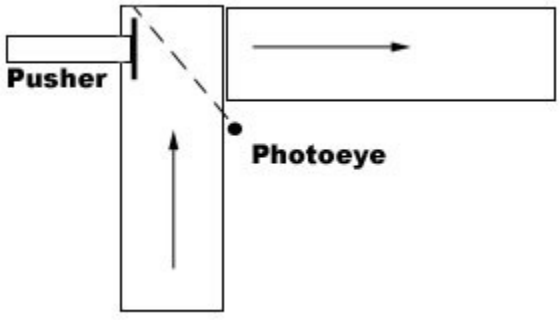
Last updated: 13 Oct, 2010

How many times have you seen your photoeye lens get dirty and think there was a part to sense but really it was just a false input? Hopefully in this example I can explain to you how to implement make / break into your control logic.

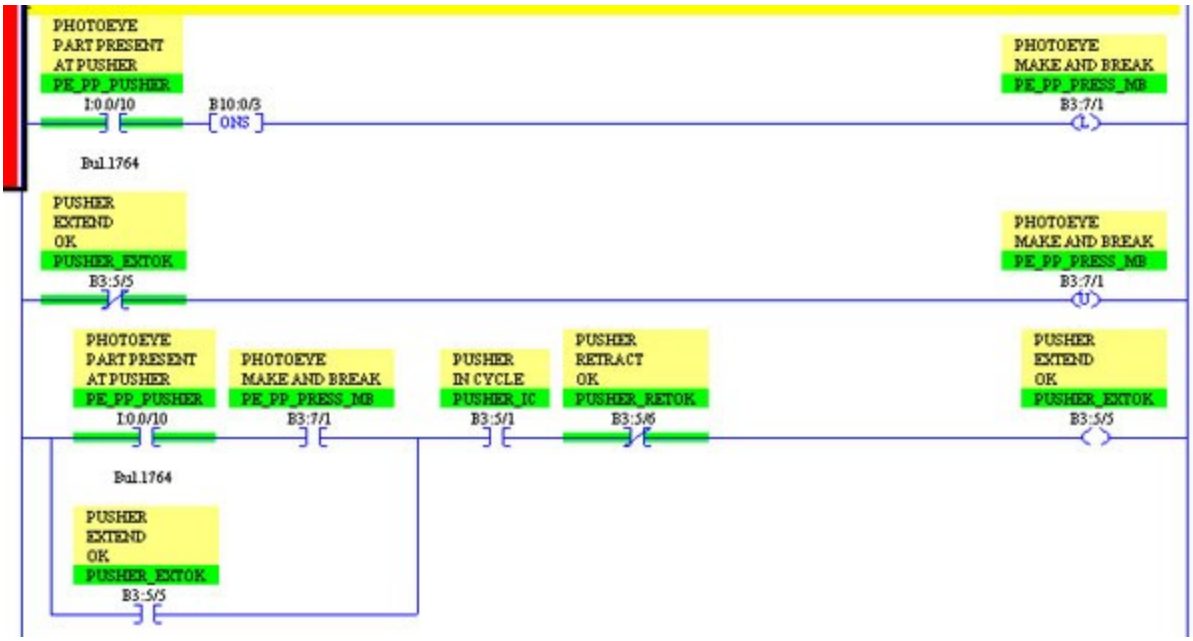
Let's say that you have a photoeye that makes sure your part is present before it fires a pusher cylinder that moves your part down stream 90 degrees. See the example logic below.




Most everyday, the photoeye will function properly by seeing your part on the conveyor, sending a signal to the PLC, and then the PLC will tell the pusher to push the part to the other side. However, let's say one day the photoeye lens gets dirty. Now no matter if you have a part in front of the sensor or not, the input to the PLC is still going to be true. So when it comes time to evaluate this rung again, your pusher would function just like normal even with no part present and the maintenance guys are going to give you a goofy look because your pusher has just cycled with not part on the conveyor.




Designing make / break logic for devices such as a pusher to verify the part has indeed left the automated area is a common practice easy to implement. The concept is to use an internal latch bit in conjunction with a single shot bit. Once your pusher has retracted it will unlatch the "make" bit. See example logic below.

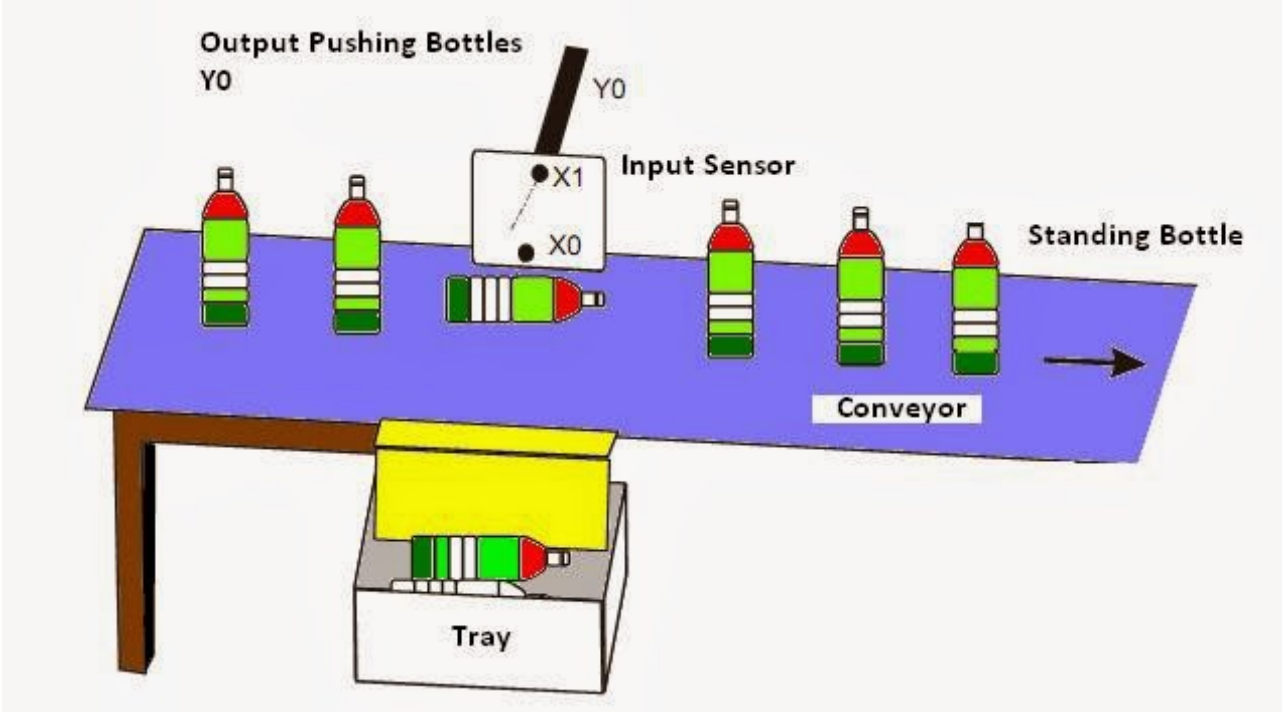



So now the next time the rung is evaluated, the one shot will fail to set the "make" bit again, because your photoeye never turned off. Your photoeye must turn off, then turn back on again to set the "make" bit and allow your pusher to cycle once again.

Some other situations that arise when make / break logic would come in handy:

1. Cable cut and photoeye shorted to a high signal for PNP or shorted to a low signal for NPN.
2. Using photoeye in a gravity application and the part does not drop, but hangs in the automation and does not drop down.
3. Photoeye lens gets dirty or sees dirt from common industrial exposure.
4. Person standing by the photoeye anf "flags" the eye with their white shirt.

Detecting the standing bottles on the conveyor and pushing falling bottles in tray.




PLC LADDER Programming Practice Problems 1

Topics Covered in this example is **using Contacts in series.**

Number of PLC Inputs Required

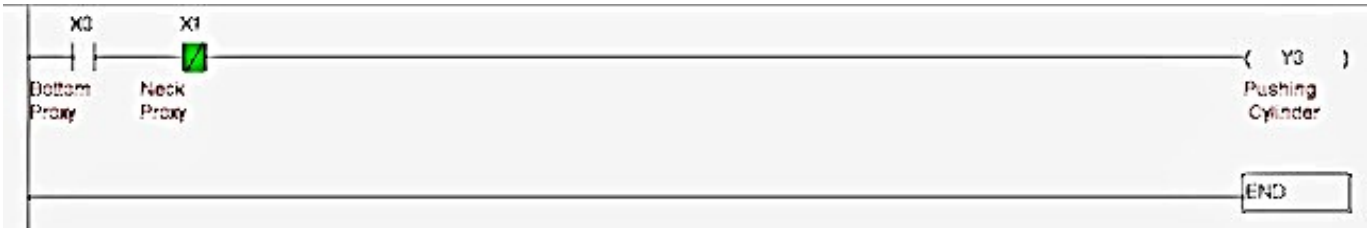
X0 – Proximity Sensor to sense bottom of the Bottle i.e. X0 = ON when the detected input signal from the bottle-bottom is sheltered.

X1 – Proximity Sensor to sense upper part of the Bottle i.e. X1 = ON when the detected input signal from the bottle-neck is sheltered.

Number of PLC Outputs Required

Y0 – To operate Pushing Cylinder/Rod

PLC Ladder Programming:

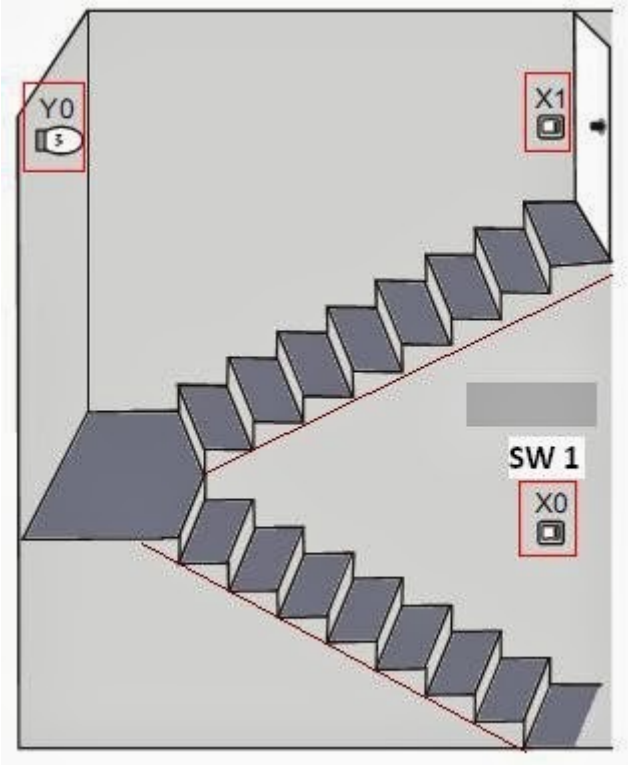



PLC LADDER Programming Practice Problems 1

PLC Ladder Program Description:

- If the bottle on the conveyor belt is upstanding, the input signal from monitoring photocell at both bottle-bottom and bottle-neck will be detected. In this case, X0 = ON, and X1 = ON. The normally open (NO) contact X0 will be activated as well as the normally closed (NC) contact X1. Y0 remains OFF and pneumatic pushing pole will not perform any action.
- If the bottle from the conveyor belt is down, only the input signal from monitoring photocell at the bottle-bottom will be detected. In this case, X0 = ON, X1 = OFF. The state of output YO will be ON because the NO contact X0 activates and the NC contact X1 remains OFF. The pneumatic pushing pole will push the fallen bottle out of the conveyor belt.

Switching on/off the Lamp whether they are at the bottom or the top of the staircase.




PLC Ladder Programming Practice Problem 2

Topics Covered in this example is **using Contacts in parallel.**

**Number of PLC Inputs Required**

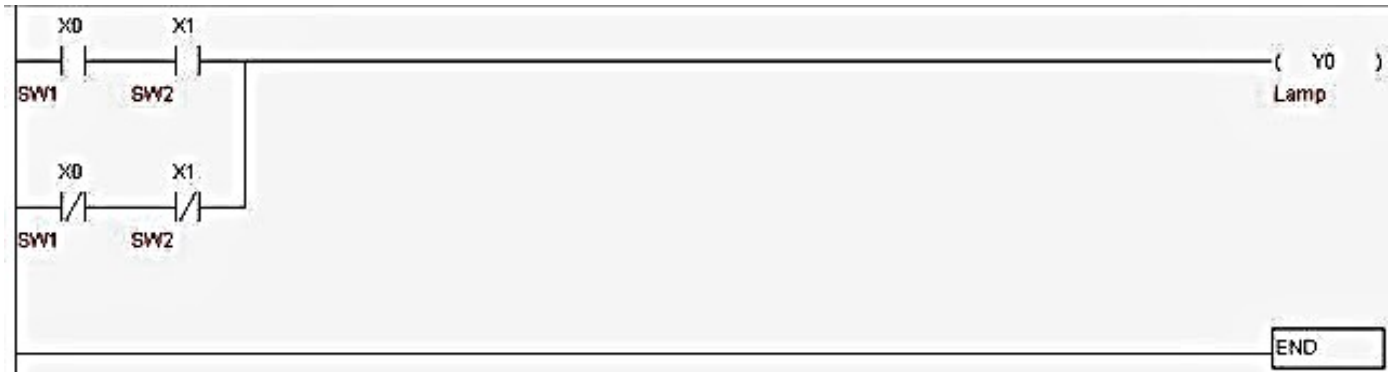
X0 – Switch at the bottom of Staircase i.e. X0 turns ON when the bottom switch is turned to the right.

X1 – Switch at the top of Staircase i.e. X1 turns ON when the top switch is turned to the right.

**Number of PLC Outputs Required**

Y0 – Lamp

PLC Ladder Programming:

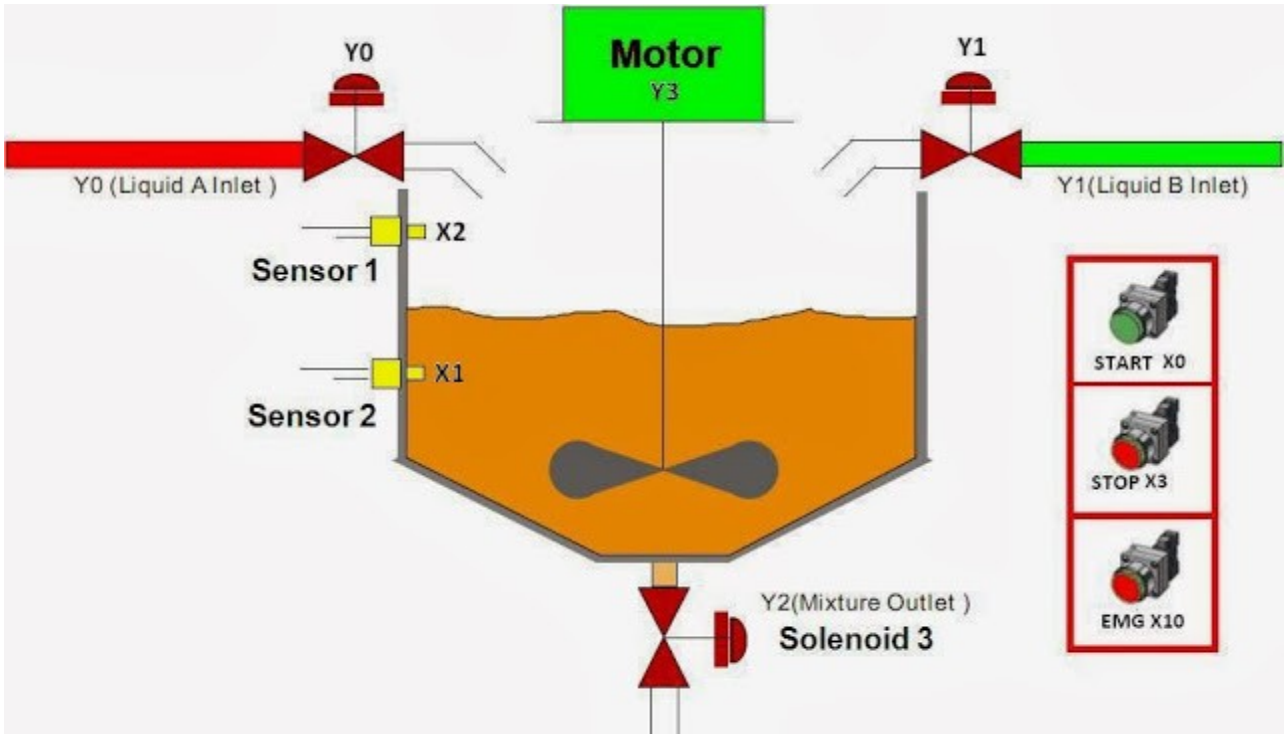



PLC Ladder Programming Practice Problem 2

PLC Ladder Program Description:

- If the states of the bottom switch and the top switch are the same, both ON or OFF, the light will be ON. If different, one is ON and the other is OFF, the light will be OFF
- When the light is OFF, users can turn on the light by changing the state of either top switch at the bottom switch of the stairs. Likewise, when the light is ON, users can turn off the light by changing the state of one of the two switches.

**Automatically infusing the container with liquids A and B in order when START is pressed.When it reaches the set level, mix the two liquids evenly then open the valve to let out the mixture.**



Topics Covered in this example is **PLC based Batch Process.**

## Number of PLC Inputs Required

X1 – Start Switch.

X1 – Low level float sensor. X1 = ON when the liquid level reaches X1.

X2 – High level float sensor. X2 = ON when the liquid level reaches X2.

X3 – Stop Switch.

X10 - EMERGENCY STOP button. X10 = ON when the button is pressed.

## Number of PLC Outputs Required

Y0 – Liquid A Inlet

Y1 – Liquid B Inlet

Y2 – Mixture Outlet

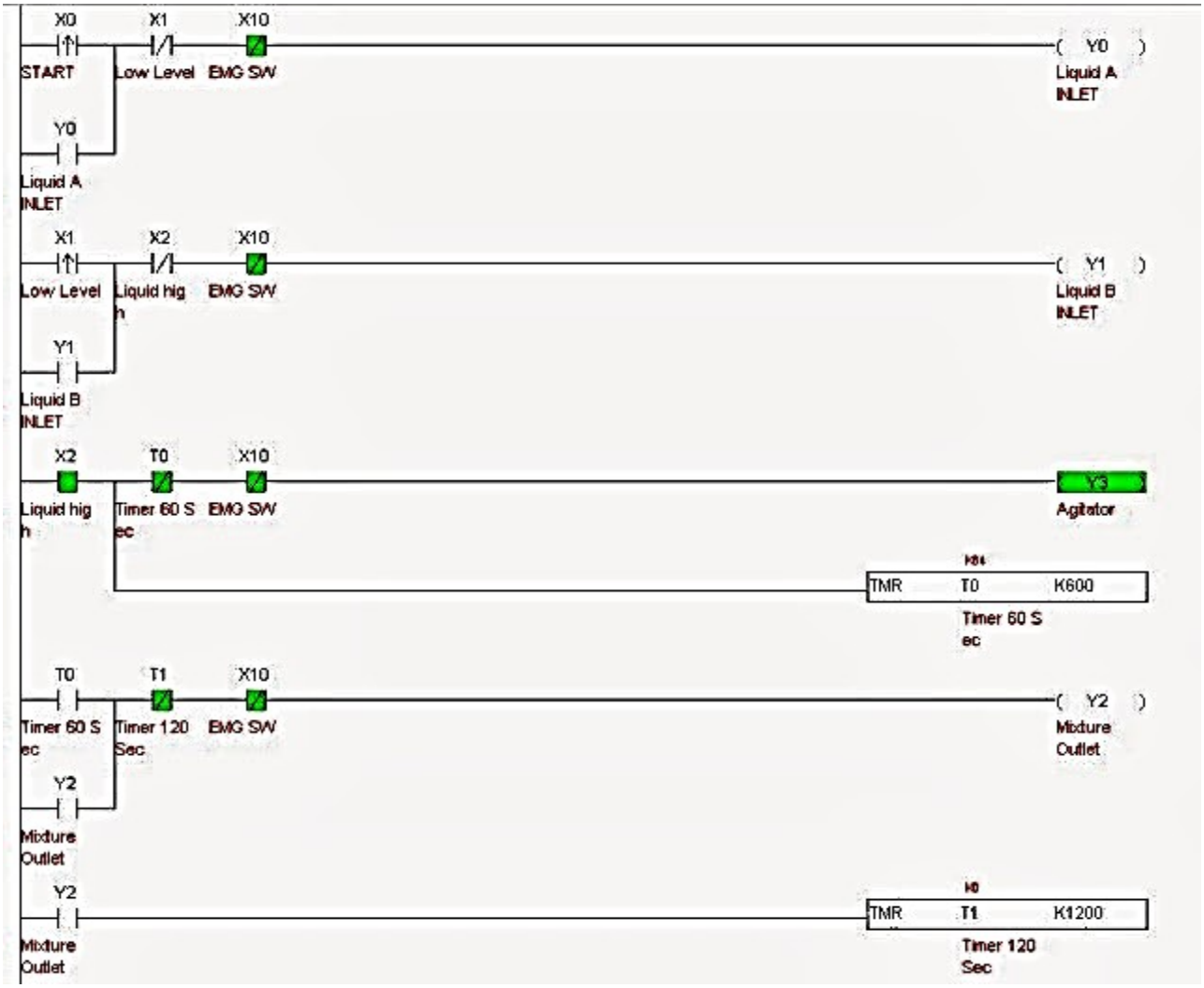
Y3 – Agitator /Stirrer

## Number of PLC Timer Required

T0 – 60 second Timer, 100 ms Time Base. (See K60 Preset Value for Timer)

T1 – 120 second Timer, 100 ms Time Base. (See K1200 Preset Val. for Timer)

PLC Ladder Programming:




PLC Ladder Program Description:

- X0 = ON when START is pressed. Y0 will be ON and latched, and the valve will be opened for infusing liquid A until the level reaches the low-level float sensor.

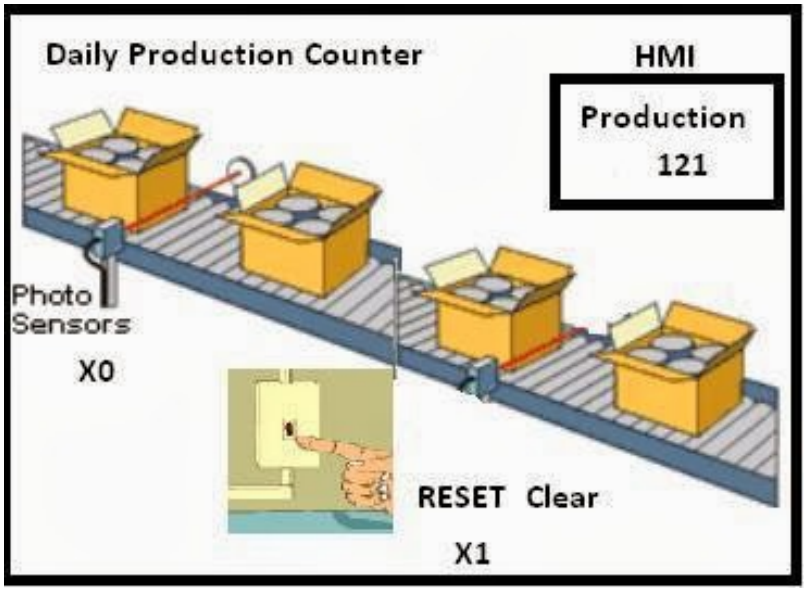
- X1 = ON when the level reaches the low-level float sensor. Y1 will be ON and latched, and the valve will be opened for infusing liquid B until the level reaches the high-level float sensor.

- X2 = ON when the level reaches the high-level float sensor. Y3 will be ON and activates the agitator. Also, timer T0 will start to count for 60 sec. After 60 sec, T0 will be ON, and the agitator motor Y3 will stop working. Y2 will be ON and latched, and the mixture will drain out of the container.

- When Y2 = ON, timer T1 will start to count for 120 sec. After 120 sec, T1 will be ON and Y2 will be OFF. The draining process will be stopped.

- When an error occurs, press EMERGENCY STOP button X10. The NC contact X10 will be ON to disable all the outputs. The system will then stop running.

**The production line may be powered off accidentally or turned off for noon break. The program is to control the counter to retain the counted number and resume counting after the power is turned ON again. When the daily production reaches 500, the target completed indicator will be ON to remind the operator for keeping a record. Press the Clear button to clear the history records. The counter will start counting from 0 again.**




PLC Ladder Programming Practice Problem 5

**Topics Covered in this example is** Latched 16 bit UP counter.

**Number of PLC Inputs Required**

X0 – Product Detecting Sensor.

X1 – Production Counter RESET/Clear

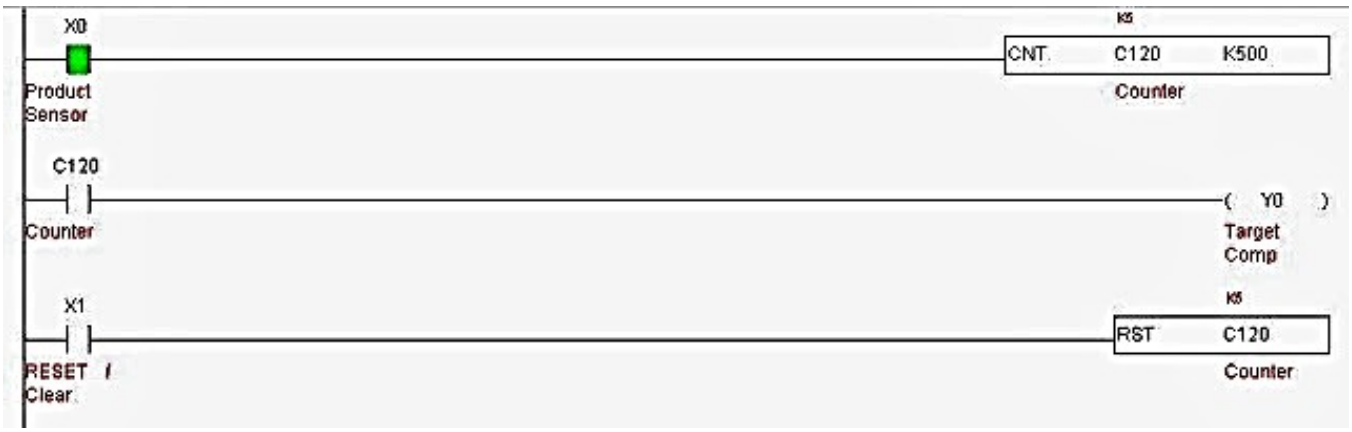
## Number of PLC Outputs Required

Y0 – Production Counter Target Completed.

## Number of PLC Counter Required

C120 – 16 Bit Latched Counter. (Max Count =32,768)

PLC Ladder Programming:




PLC Ladder Programming Practice Problem 5

PLC Ladder Programming Description:

- The latching counter is demanded for the situation of retaining data when power-off.
- When a product is completed, C120 will count for one time. When the number reaches 500, target completed indicator Y0 will be ON.
- For different series of PLC, the setup range of 16-bit latching counter is different.

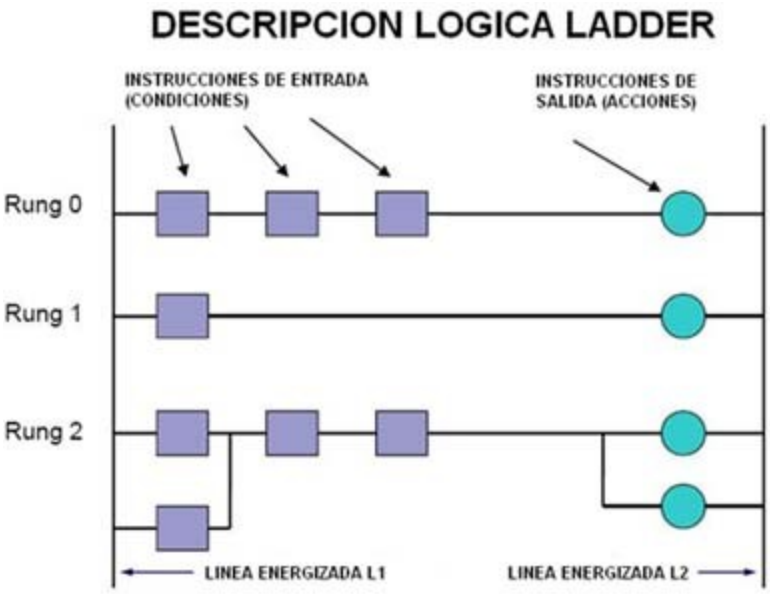
## BASIC PLC LADDER PROGRAMMING

**Description of language ladder**

There are different types of programming languages for PLCs. Perhaps the most common is the ladder programming. The ladder diagrams are commonly used schemes to represent the control logic of industrial systems. Is called "ladder" diagram because they resemble a ladder, with two vertical rails (supply) and "rungs" (horizontal lines), in which there are control circuits that define the logic through functions. In this way the main ladder language features are:

- Input instructions are entered on the left.
- Output Instructions are located on the right.
- Power rails are the power supply lines L1 and L2 for alternating current circuits, and 24 V - earth for DC circuits.

- Most PLC allow more than one output for each row (Rung).
- The Processor (or "Controller") explores rungs of the ladder from top to bottom and from left to right.

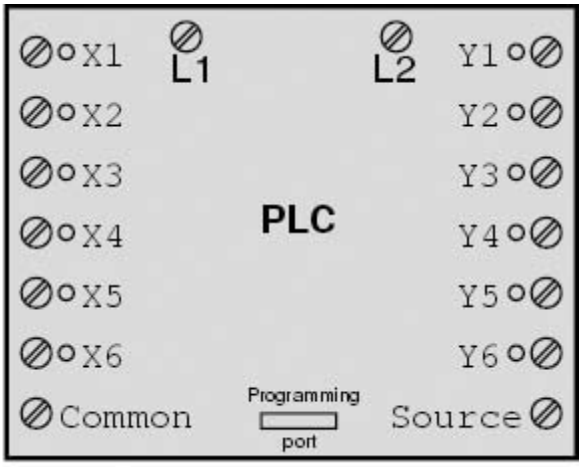



The input instructions are the conditions that the circuit has to let or not let to pass current from one line to another. These conditions are commonly handled with normally-open or normall-closed contacts which interpret the high and low signals of sensors or switches. If conditions are true the current reaches the output instructions, which generate actions such as energizing a motor coil or power on a lamp. In this way the flow of current to the output coils is conditioned by the logic managed by the input instructions.

A PLC has many input and output terminals, through which "high" or "low signals are produced " to be transmitted to power lights, solenoids, contactors, small motors and other devices provided to on / off control. In an effort to make PLCs easy to program, ladder programming language was designed to resemble ladder logic diagrams. Therefore, an industrial electrician or electrical engineer accustomed to read ladder logic diagrams will be more comfortable programming a PLC if the ladder language is handled.

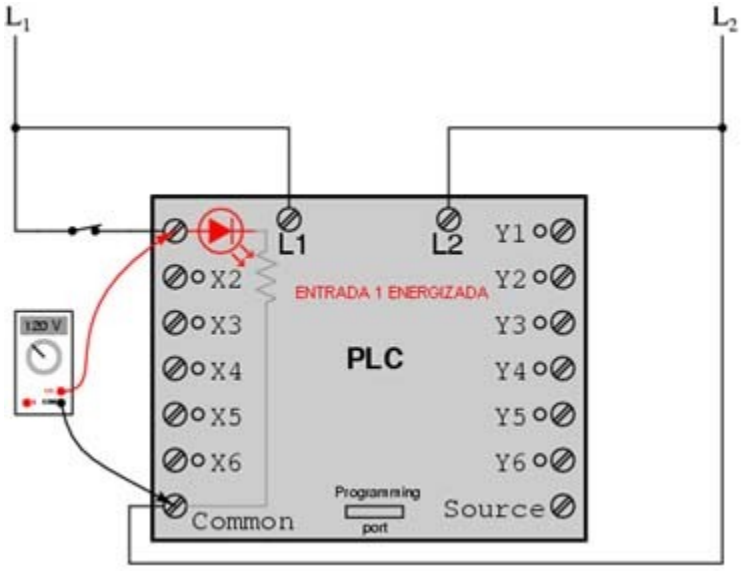
**Ladder logic and wiring**

The signal connections and programming standards vary somewhat between different models of PLC, but the concepts are the same, so both power wiring and programming are somewhat generic

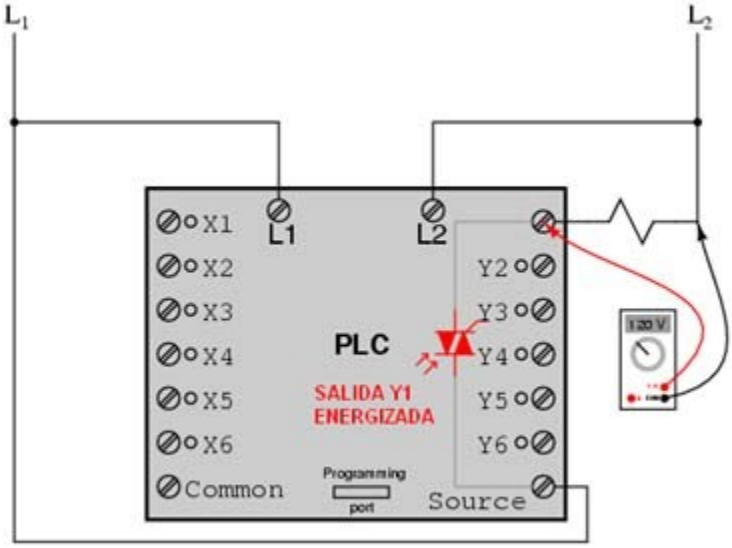



The following illustration shows a simple PLC, as might appear from a front view. Two screw terminals, L1 and L2, provide a 120 volts AC connection to supply the internal circuitry of the PLC. Six screw terminals on the left side allow you to connect input devices, each terminal representing a different input channel with its own "X" label. The lower left screw terminal is a "common" connection, which is usually linked to L2 terminal (neutral) of 120 VAC power supply.

Within the PLC, connected between the input terminals and the common terminal, is an optocoupler device that provides a high signal to the PLC's internal circuitry when a 120 VAC signal is applied between the corresponding input terminal and the common terminal. A LED indicator on the PLC front panel gives visual indication of an energized input:

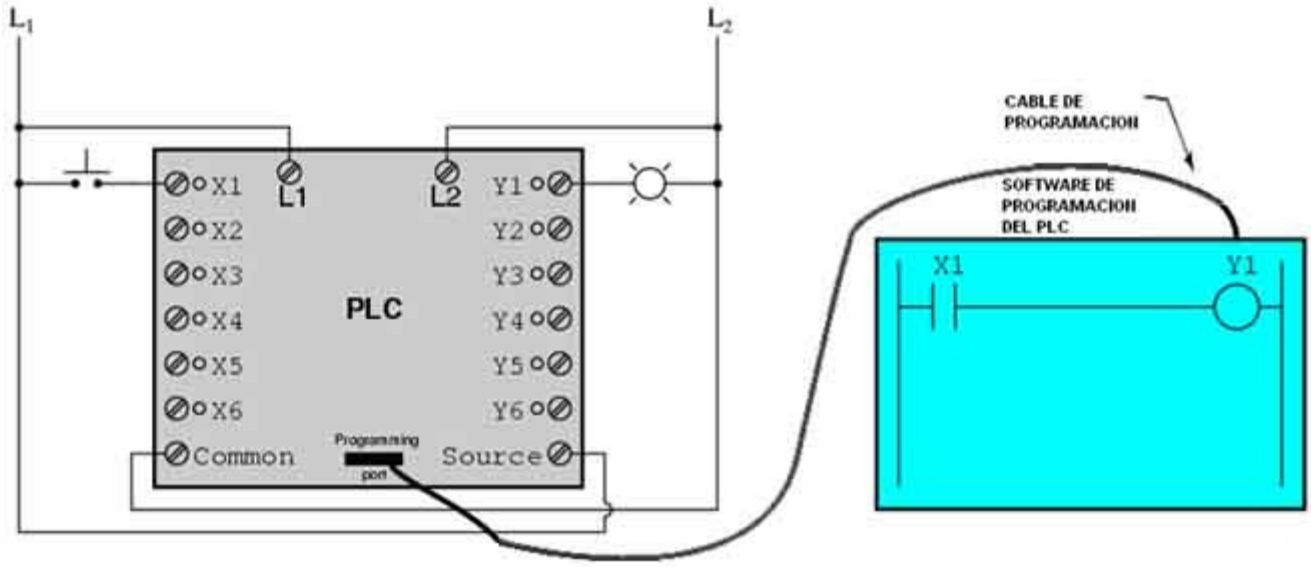



The output signals are generated by the CPU circuit of the PLC that activates a switching device (transistor, TRIAC, or even an electromechanical relay), connecting the "source" to any of the terminal outputs "Y". The terminal "Source", therefore, is usually associated with L1 from the 120 VAC power supply. As with each entry, a LED indicator on the PLC front panel gives a visual indication of an energized output:



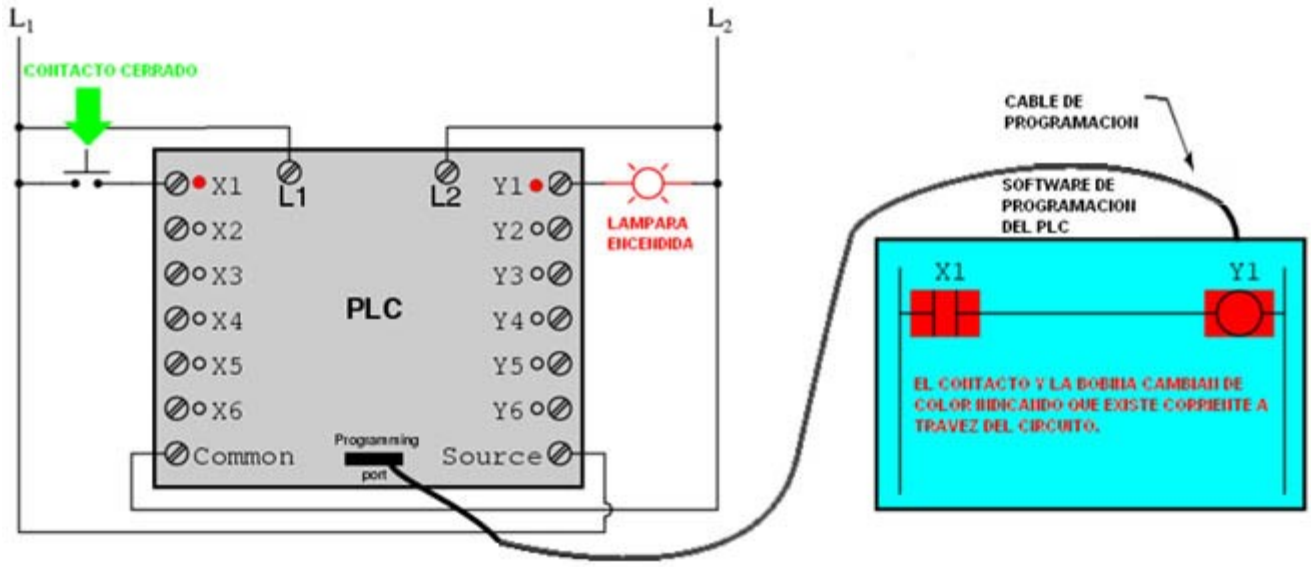

The actual logic of the control system is set in the PLC by means of a software. This software determines which output is energized under what conditions of entry. Although the program itself seems to be a ladder logic diagram, with the symbols of switches and relays, there is no actual switch contacts or relay coils in the PLC to create the logical relationships between input and output. These are imaginary contacts and coils. The program is loaded into the PLC and is seen through a personal computer connected to the PLC programming port.

To clarify how the ladder logic is related with the PLC wiring consider the following circuit and PLC program:

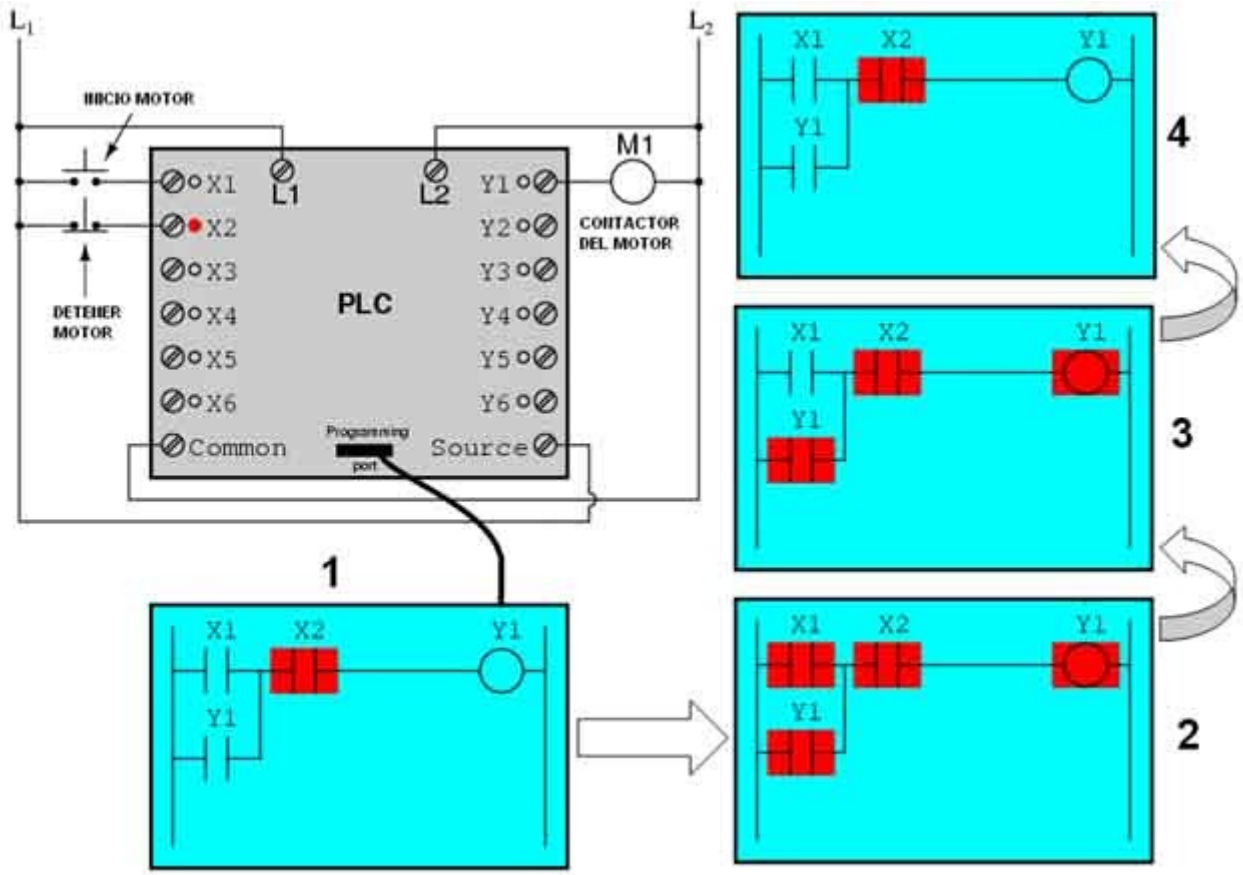



When the switch button is not pressed (off), there is no current in the PLC X1 input. The software shows a normally-open X1 contact in series with a Y1 coil. While the X1 input signal is not "high" is not sent current to the Y1 coil since the contact is normally open. Therefore, the associated output to Y1 remains de-energized and the lamp remains turned off.

If the power button is pressed the current flows through contact, which now changes state to closed, and sent a signal "high" to the PLC input X1. Each and every one of the X1 contacts appearing in the program will assume the drive (not normal), as if it were relay contacts actuated by the excitation of a relay coil named "X1". In this case, the activation of input X1 X1 will normally open contact is closed and thus permit the passage of current to the coil Y1. When the Y1 coil program "energizes" the real Y1 output energized, and thus the lamp has energy to light.




The true power and versatility of a PLC is revealed when we want to change the behavior of a control system. A PLC is a programmable device that can alter its behavior by changing its internal logical instructions without having to reconfigure the electrical components connected to it. For example, suppose that what we wanted to do with the lamp was an inverted switching: by pressing the button to turn off the lamp, and releasing it to turn it on. The "hardware" solution would require that a normally closed switch is replaced by a normally open switch. The "software" solution is much easier: simply change the program so that X1 contact is normally closed instead of normally open. Besides, since each PLC output is nothing more than a bit in its memory, we can assign contacts in the PLC program "commanded" by an output status (Y). Let's Take, for example, a control circuit for the start-stop of a motor:




The switch button connected to X1 input serves as the "start" switch, while the switch connected to X2 input serves as the "stop". Another contact in the program, named Y1, uses the output coil state as a contact seal so that the motor contactor will remain energized after the "Start" button is released. In the initial state (sequence 1) we can see the normally closed contact X2 in a color block, showing that it is in a closed state ("conducting electricity")..

Pressing the "Start" button (sequence 2) PLC input X1 is energized, so that contact X1 is closed in the program, and thus current to the Y1 coil is sent. In this way the Y1 output also is energized and the 120 volts AC are applied to the motor contactor coil. The parallel Y1 contact is also "closed", which latches the "circuit", ie, if the start button is released, the normally open X1 contact will return to "open", but the motor will continue running due to Y1 contact continues providing "continuity" to the Y1 coil current, keeping the Y1 output energized (Sequence 3).

To stop the motor, is necessary to press the "Stop" button, which activates the X2 input and open the normally closed contact, breaking the current continuity to Y1 coil. When the "Stop" button is released the X2 input is disabled, and X2 contact is back to its normal state, closed. The motor, however, will not resume until the "Start" button is activated, because the contact that was holding it on, is de-energized with the circuit continuity break, when is pressed the Stop button.