



Benha University

Prof. Mohamed Ahmed Ebrahim

Benha University
Faculty of Engineering at Shoubra
Electrical Engineering Dept.



Industrial Controls (1)

By



Associate Prof. / Mohamed Ahmed Ebrahim Mohamed

E-mail: mohamedahmed_en@yahoo.com

mohamed.mohamed@feng.bu.edu.eg

Web site: <http://bu.edu.eg/staff/mohamedmohamed033>





Lecture (7)
09– 05 - 2021



Function Blocks

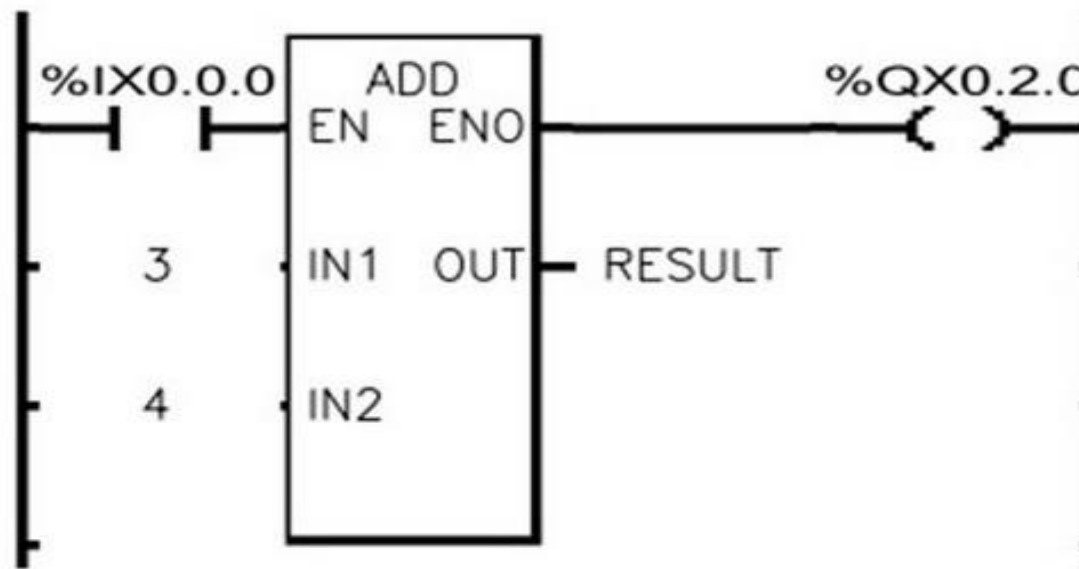
Prof. Mohamed Ahmed Ebrahim

Function Blocks

- ❑ **There are several types of function blocks in ladder programming that implement:**
 1. Logic operations (AND, OR, NOT, etc.).
 2. Math operations (addition, multiplication, etc.)
 3. Timers.
 4. Counters.

2. Math operations

In the ladder programming, there are blocks to implement different types of mathematical operation such as addition and subtraction.

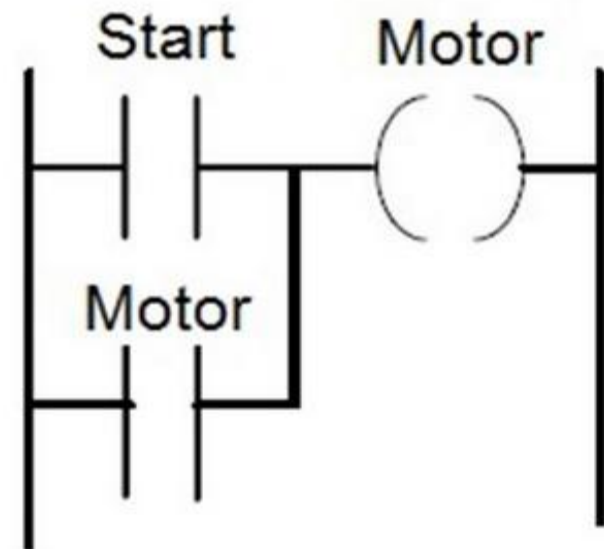


Latching

- There are often situations where it is necessary to hold an output energized, even when the input ceases.
- A simple example of such a situation is a motor which is started by pressing a push button switch. Though the switch contacts do not remain closed, the motor is required to continue running until a stop push button switch is pressed.
- Latching is particularly useful for making a momentary pushbutton switch perform as if it were a maintained switch.
- The *latch circuit* is a self-maintaining circuit in that, after being energized, it maintains that state until another input is received.

Latching circuit with **start** button

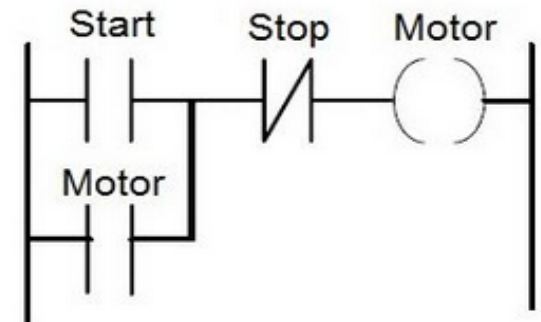
- The circuit shown can be used to start a motor using the **Start** push button.
- When **Start** is pressed, the motor starts.
- When **Start** is released, the holding contacts maintain the circuit and hence the power to the motor.
- **But how to stop the motor?**



Latched circuit

Latching circuit with **start** and **stop** buttons

- To stop the motor, we add another push button (**Stop**) which is normally closed.
- When **Stop** button is pressed, this disconnects the power to the motor and the holding contacts open. Thus when **Stop** is released, there is no power connected to the motor.
- In conclusion, we have a motor which is started by **Start** and stopped by **Stop** push buttons.

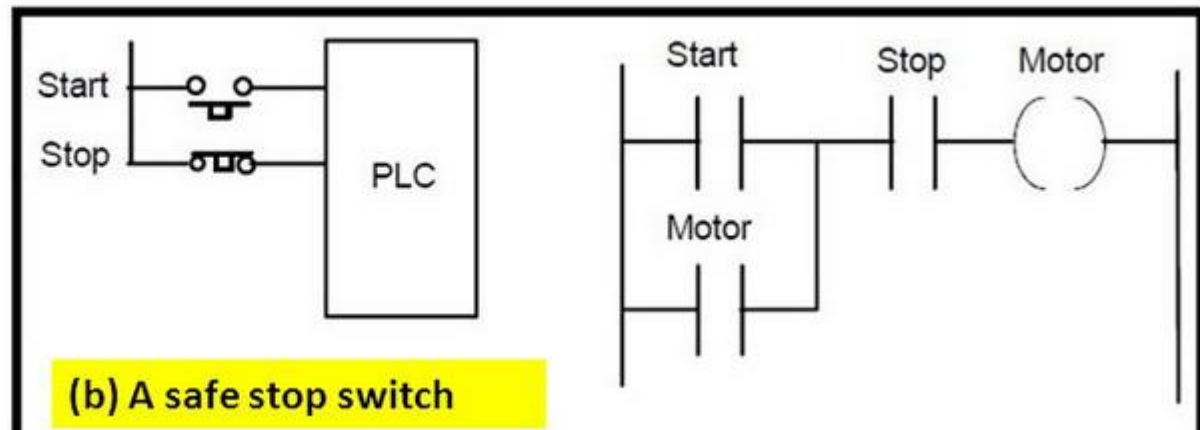
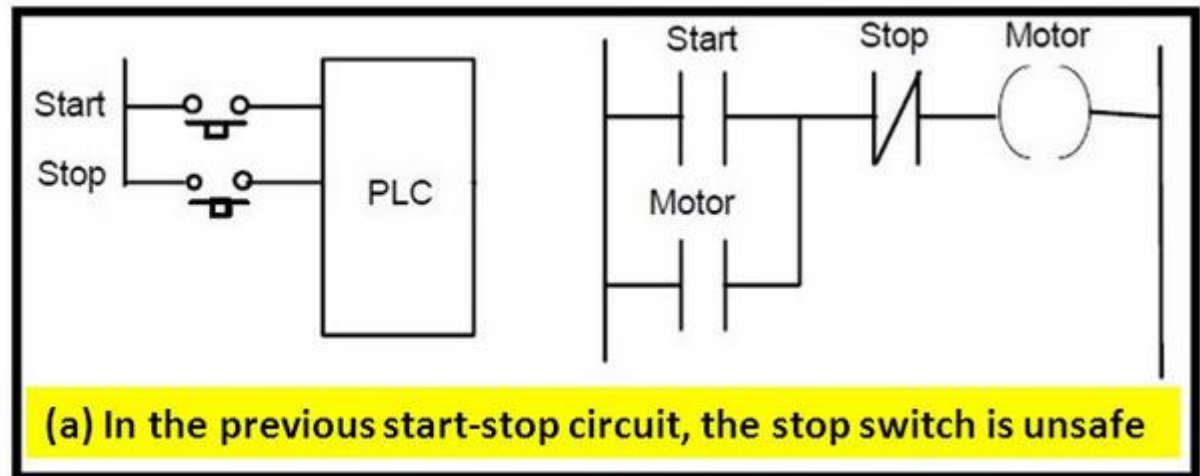


Latched circuit

Safe and Unsafe Stop switches

To ensure **safety**, stop switches has to be very carefully considered.

We have to ensure that the Stop switch will do its job even when it fails.

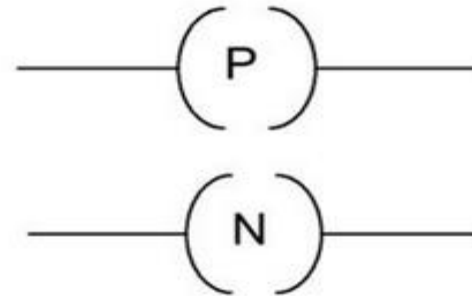
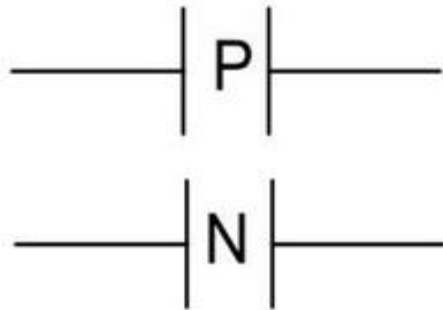


Cont.

- In electrical circuits, the "open" failures (open switch contacts, broken wire connections, open relay coils, blown fuses, etc.) are statistically more likely to occur than any other type of failure.
- A stop switch is *not* safe if it is normally open and has to be closed to give the stop action. Imagine that the stop switch fails. It most likely fails open (becomes open permanently). In this case, the system cannot be stopped.
- A better arrangement is to program the stop switch in the ladder program as open and use a stop switch that is normally closed and operating opens it.

Special types of **contacts** and **coils**

- There are special types of contacts and coils which are associated with whether the input variable or power flow is having a +ve or -ve edge changes.
- They have the following symbols.



I. Positive and Negative transition-sensing **Contacts**

Positive transition-sensing contact:

when the associated variable changes from 0 to 1,
power flows for one ladder rung evaluation.



Negative transition-sensing contact:

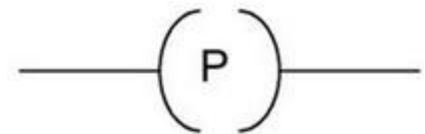
when the associated variable changes from 1 to 0,
power flows for one ladder rung evaluation.



II. Positive and Negative transition-sensing coils

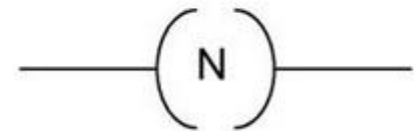
Positive transition-sensing coil:

if the power flow to it changes from off to on, the output is set on for one ladder rung evaluation.



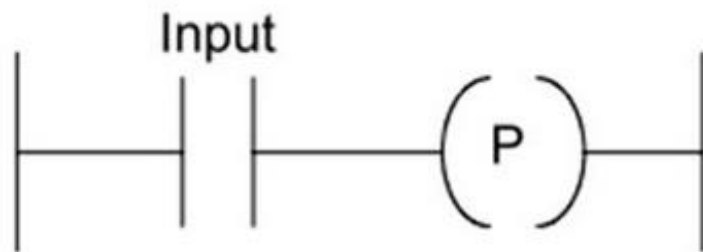
Negative transition-sensing coil:

if the power flow to it changes from on to off, the output is set on for one ladder rung evaluation.



Example

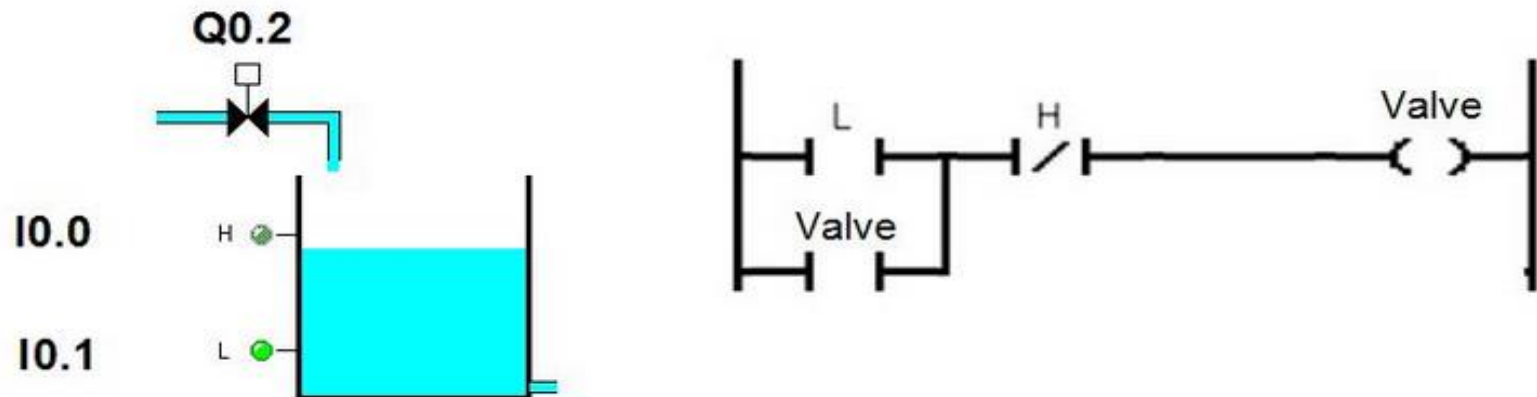
- For the ladder rung shown, with the input off there is no output. When the input switches on, there is an output from the coil.
- However, the next and successive cycles of the program do not give outputs from the coil even though the switch remains on. The coil only gives an output for *one ladder rung evaluation* the first time the switch is on.



Evaluation	Input	P output
1	Off	Off
2	On	On
3	On	Off
4	On	Off

Example (On-off liquid level control)

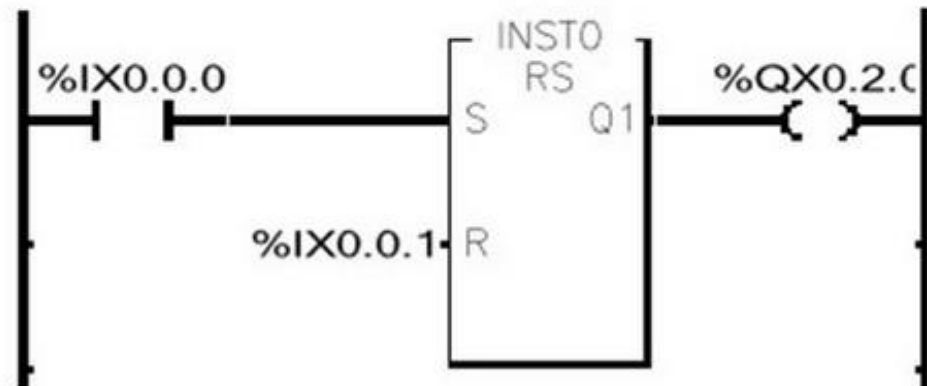
- It is required to keep the water level between two levels: **L** (Low) and **H** (High).
- There are two sensors: one that gives 1 if the level is below **L** and the other gives 1 if the level is above **H**.
- If the level is below **L**, the valve is opened to fill the tank till the level reaches **H** when the valve closes.
- *Design the ladder diagram for this control system.*



SR Flip-Flop

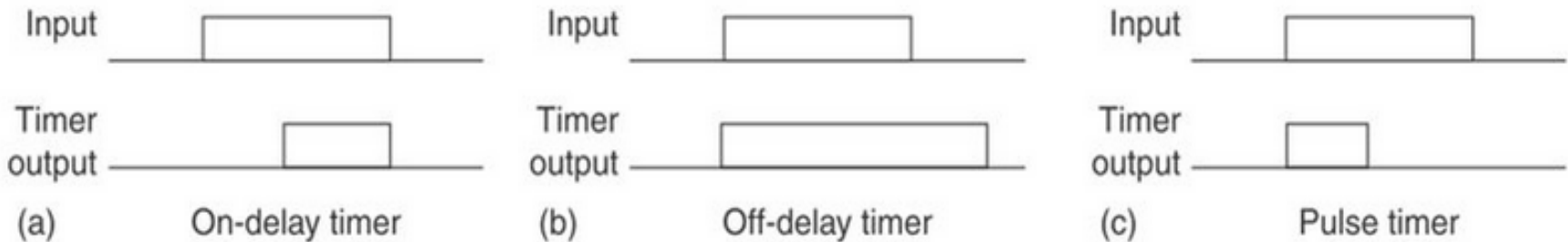
- You may have noted that the previous example may be solved using the SR flip-flop memory action.
- It is interesting that there is a function block which implements the SR Flip-flop function with following truth table.

S	R	Q_n
0	0	Q_{n-1}
0	1	0
1	0	1
1	1	0



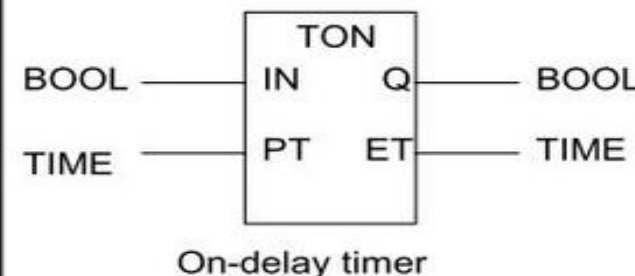
3. Timers

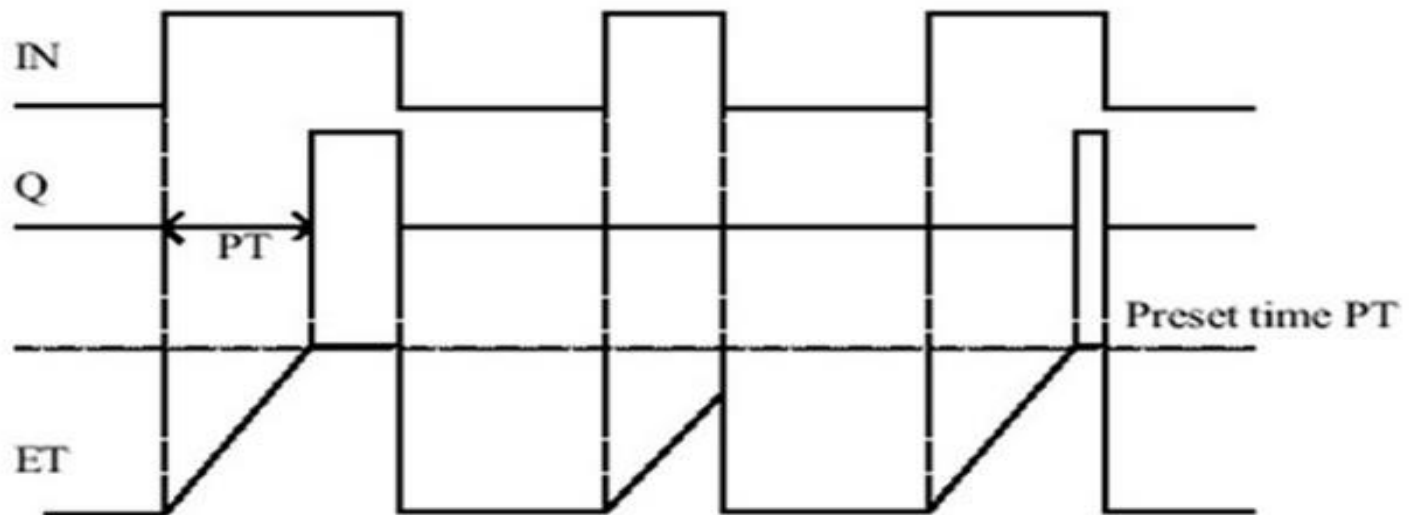
- **Timers:** are used to operate devices for certain period of time.
- **PLC contains three basic types of timers:**
 - a) On-delay timer (TON).
 - b) Off-delay timer (TOF).
 - c) Pulse-timer (TP).



Timers: (a) on-delay (b) off-delay (c) pulse

a) ON – Delay Timer

Function	Description
 <p>On-delay timer</p>	<p>Input</p> <ul style="list-style-type: none"> IN : Timer operation condition PT : Preset Time <p>Output</p> <ul style="list-style-type: none"> Q : Timer output ET : Elapsed Time

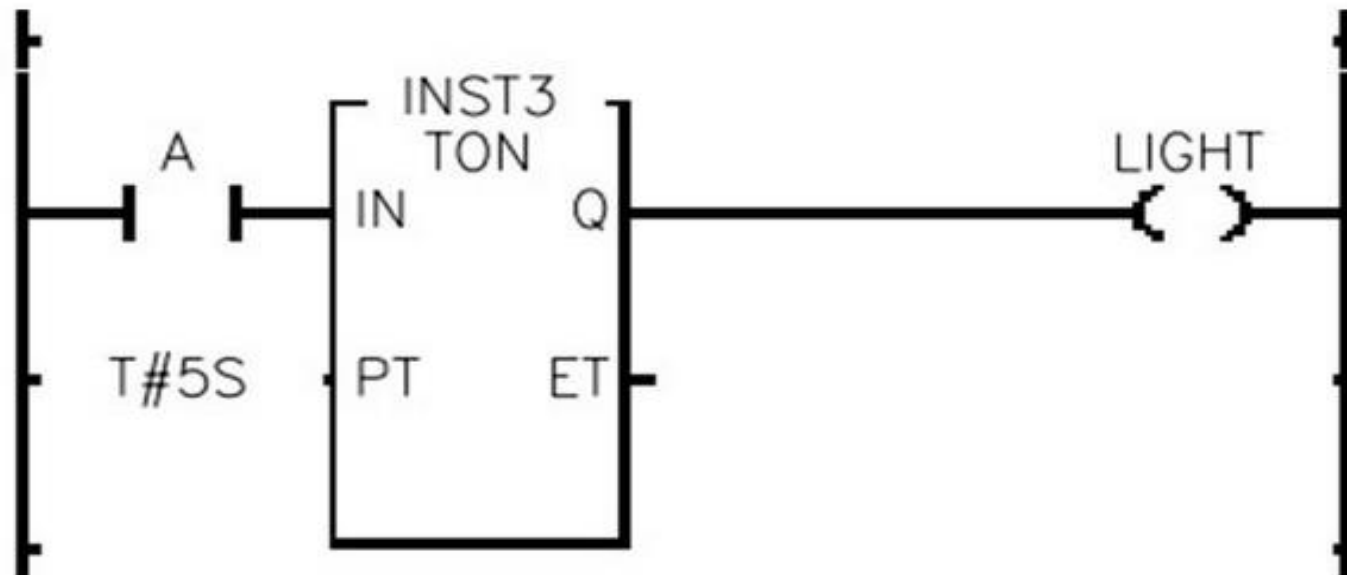


Cont.

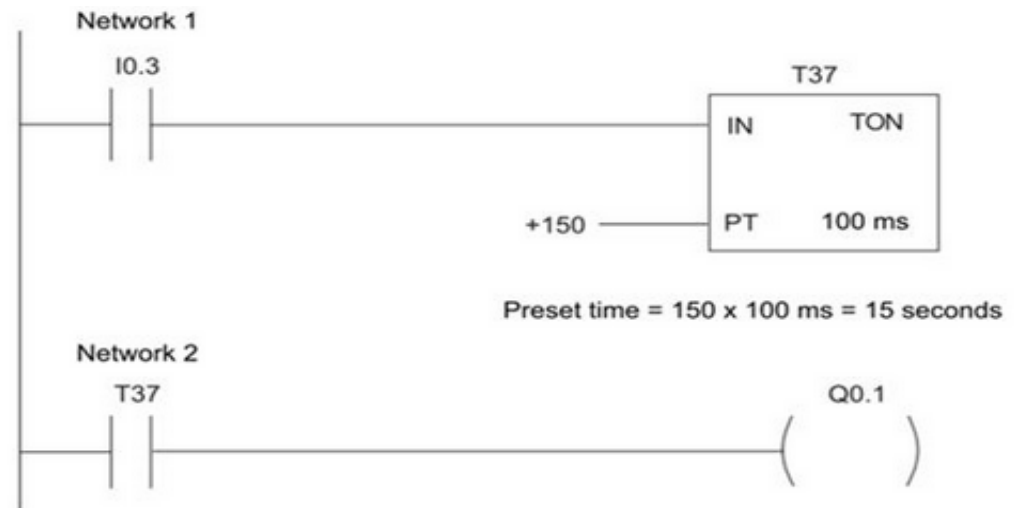
- When the input **IN** changes from 0 to 1, the output **Q** changes from 0 to 1 after a time interval set at **PT** (preset time). During this interval, **ET** outputs the elapsed time.
- If **IN** is 0 before **ET** reaches the preset time, the elapsed time becomes 0.
- If **IN** is 0 after **Q** is 1, **Q** will be 0.

Example (1): LG

Develop the ladder logic that will turn on an output light, 5 seconds after switch A has been turned on.



Example (2): Siemens



- In this example, when input I0.3 turns on, timer T37 begins timing.
- T37 has a time base (resolution) of 100 ms (0.1 seconds). The preset time (PT) value has been set to 150. This gives 15 seconds delay (150 x 100 ms).
- Therefore, 15 seconds after the I0.3 contact closes, timer output becomes a logic 1, and output coil Q0.1 turns on. If the switch opens before 15 seconds has elapsed, the elapsed time (ET) resets to 0.

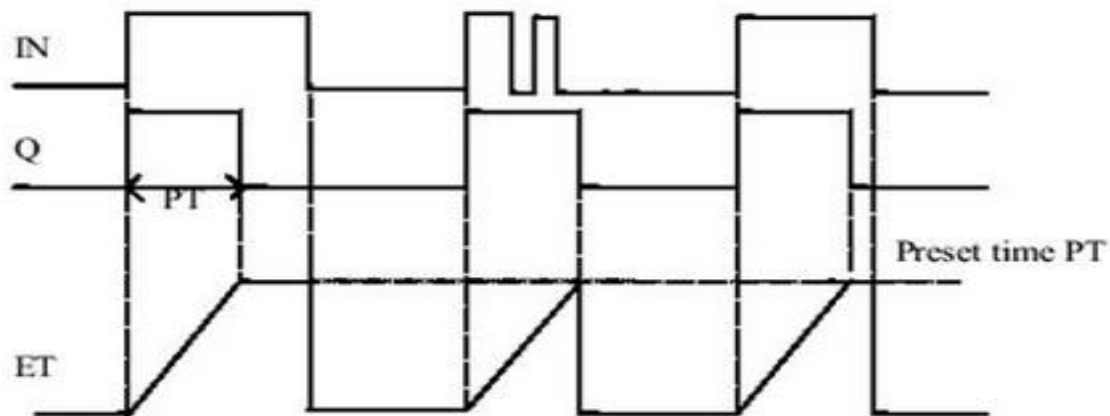
b) Off – Delay Timer

In Off-Delay timer, if the input **IN** turned off (change from 1 to 0), the timer waits for a certain time interval and then turns off its output **Q**.

c) Pulse Timer

Function	Description
	<p>Input</p> <p>IN : Timer operation condition PT : Preset Time</p> <p>Output</p> <p>Q : Timer output ET : Elapsed Time</p>

Time chart



Cont.

- If **IN** is changes from 0 to 1, **Q** becomes 1 during the preset time, and if **ET** reaches **PT**, **Q** becomes 0 automatically.
- Elapsed time **ET** increases when **IN** is 1 and holds the value when it reaches **PT** and becomes 0 when **IN** is 0.
- It does not matter whether **IN** is 0 or 1 while the output **Q** is 1.

4. Counters

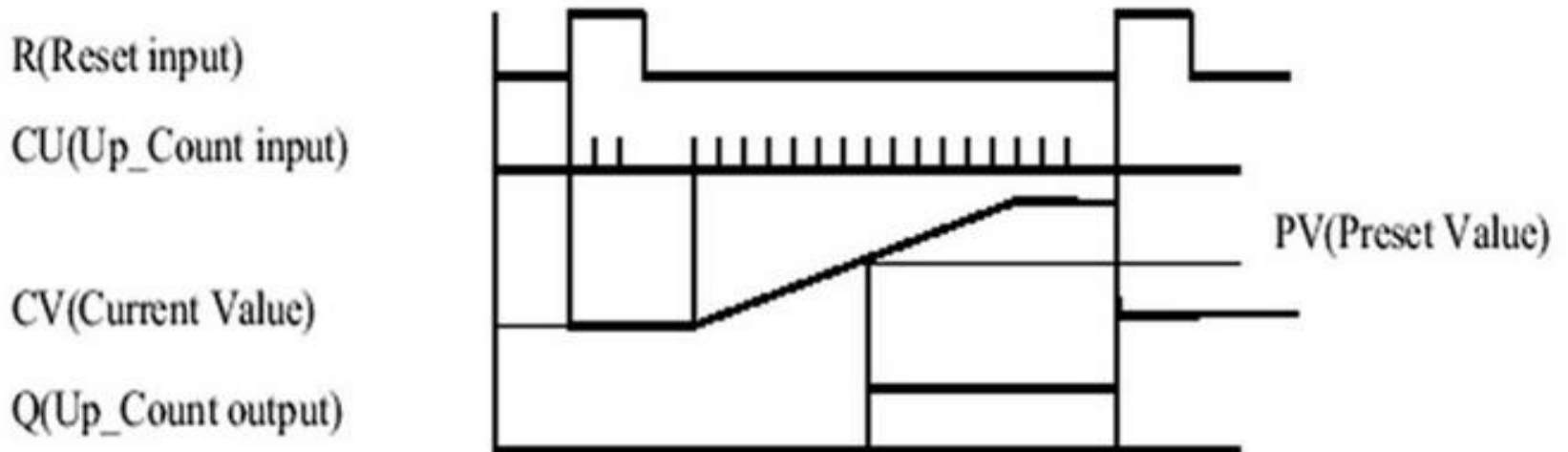
- ❑ PLC counter instructions **keep track** of events. As it counts, a counter instruction compares an **accumulated count** value to a **preset value** to determine when the desired count has been reached.
- ❑ **Counters** can be used to start an operation when a count is reached or to prevent an operation from occurring until a count has been reached.

Cont.

- **Types of counters in PLC:**
 - a) Up counter (CU).
 - b) Down counter (CD).
 - c) Up-down counter (CUD).

a) UP – Counter (CTU)

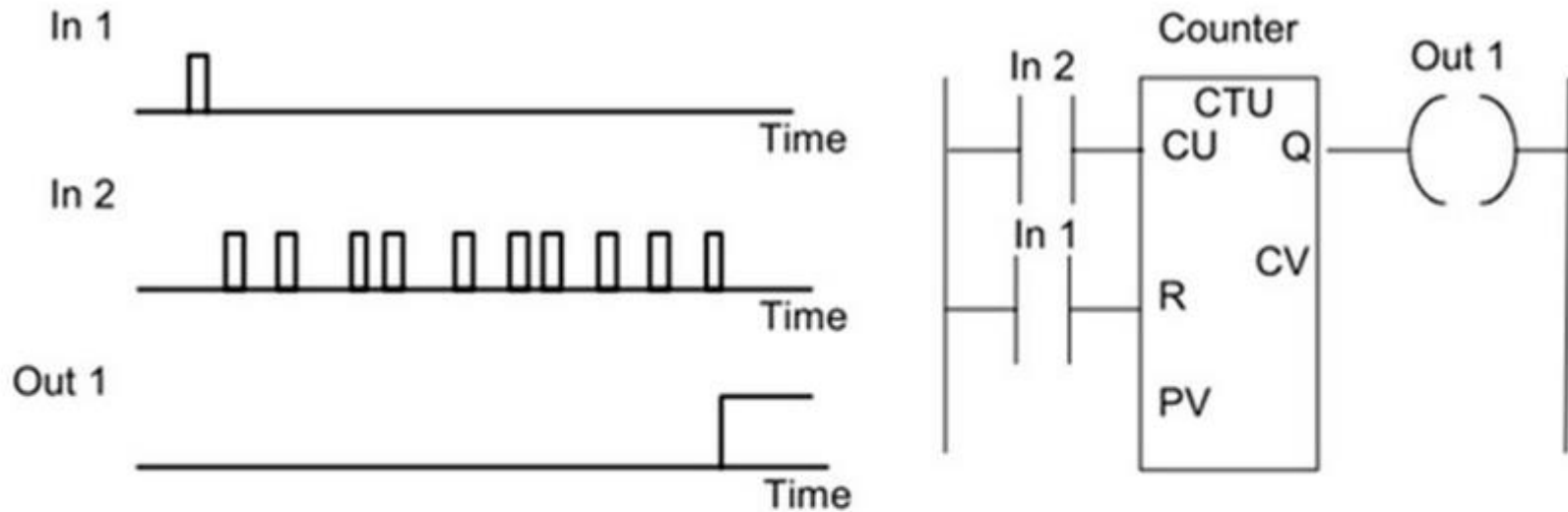
Function	Description
	<p>Input</p> <ul style="list-style-type: none"> CU : Up_Count pulse input R : Reset input PV : Preset Value <p>Output</p> <ul style="list-style-type: none"> Q : Up_Count output CV : Current Value



Cont.

- **PV** (preset value) is the count value to be stored in the counter.
- Up counter CTU increases **CV** (current value) by 1 when input **CU** changes from 0 to 1 (i.e. positive edge).
- Output **Q** is 1 when **CV** \geq **PV**.
- When reset input **R** is 1, **CV** is cleared (becomes 0).

Example (3)

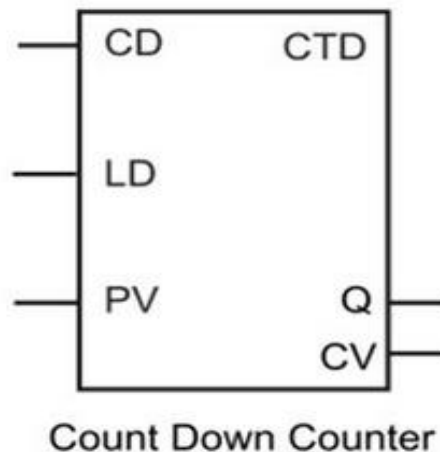


Basic counter program

In this example, **Out1** will turn on after switch **In2** has been closed 10 times. Push button **In1** will reset the counters.

b) Down – Counter (CTU)

- The down counter starts its counts from a value **PV**.
- **PV** is loaded to **CV** (current value) using the input **LD**.
- Each time a positive edge occurs on **CD** (count down) terminal the **CV** is decremented by one.
- If **CV** \leq 0, **Q** turns on.

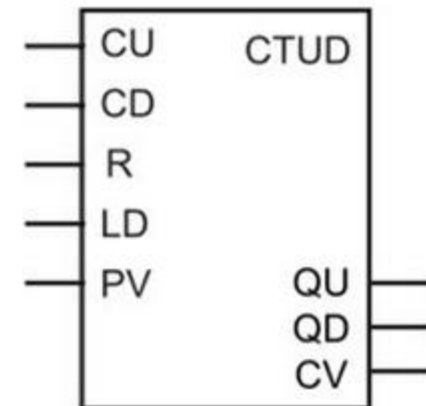


c) UP / Down – Counter (CTU)

- Every +ve edge on **CU** increment **CV** by 1.
- Every +ve edge on **CD** decrement **CV** by 1.

CV	QD	QU
$CV \leq 0$	1	0
$CV \geq PV$	0	1

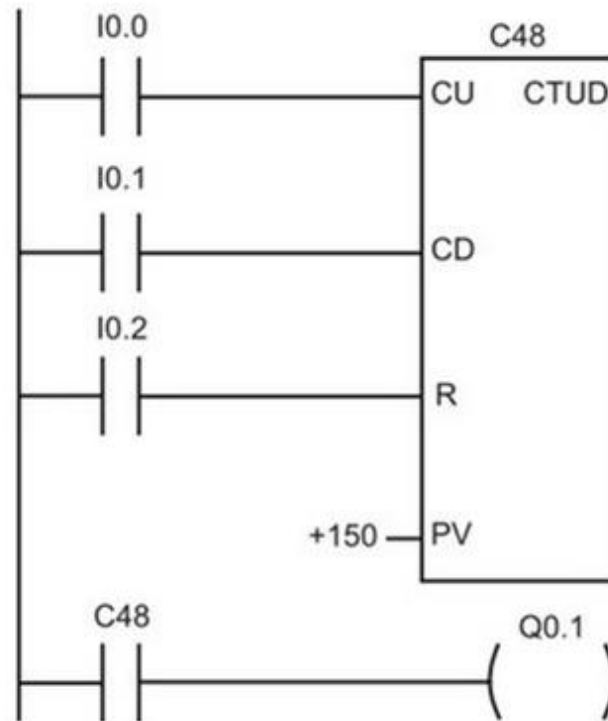
- **R**: reset the counter
- **LD**: load CV with the value PV.



Count Up/Down Counter

Example (4)

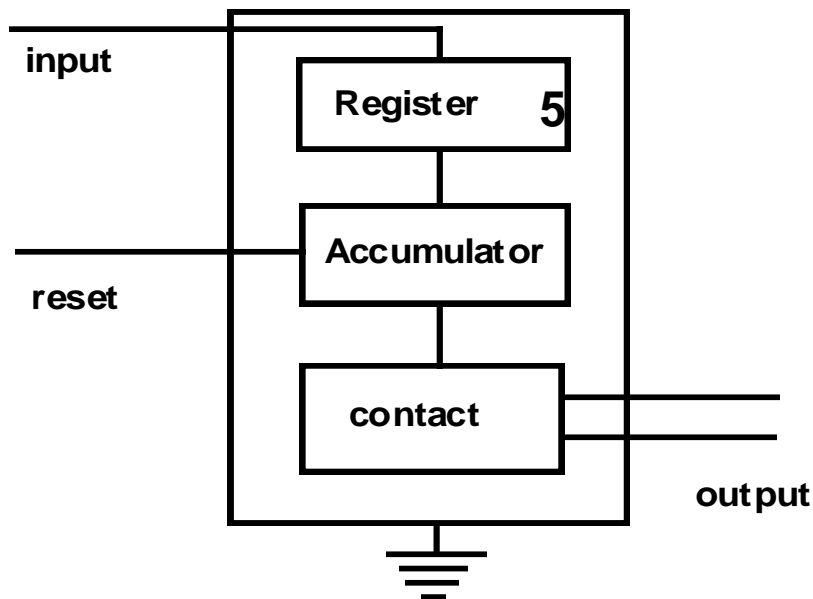
A counter might be used to keep track of the items in an inventory storage area.



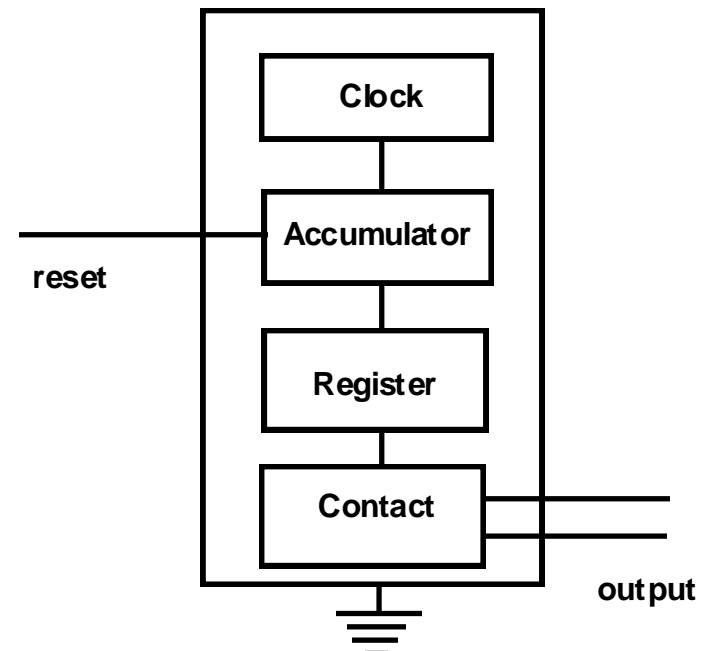
Example (4) – Cont.

- In the previous slide, Counter **C48** is reset to zero when contact **I0.2** closes. This could be triggered automatically or manually to indicate that the storage location is empty.
- When contact **I0.0** closes, the counter counts up by 1. This could be triggered by a proximity switch sensing that an item has been placed in the storage location.
- When contact **I0.1** closes, the counter counts down by 1. This could be triggered by a proximity switch sensing that an item has been removed from the storage location.
- When the accumulated count reaches 150, the counter turns on, contact **C48** closes, and output **Q0.1** turns on. This could trigger other logic in the program to divert new items to another location until an item is removed from this location.

TIMERS AND COUNTERS(continue)



COUNTER



TIMER

PLC Industrial applications

Prof. Mohamed Ahmed Ebrahim

EX:(1): PLC Program for Motor Starter

- **PLC for motor starter It should have the following provisions:**
 1. **Push button:** to start the motor, The motor should continue to rotate even when the push button is released.
 2. **Stop Push button:** to halt the motor after it started.
 3. **Over current protection :** In case of over load, the motor should stop automatically by the signal coming from contactors of overload relay.
 4. **Limit switch :** It should prevent the motor from starting and can also stop the running motor.
 5. The motor starter should also have **indicator** (Lights) to show ON or OFF status of motor.

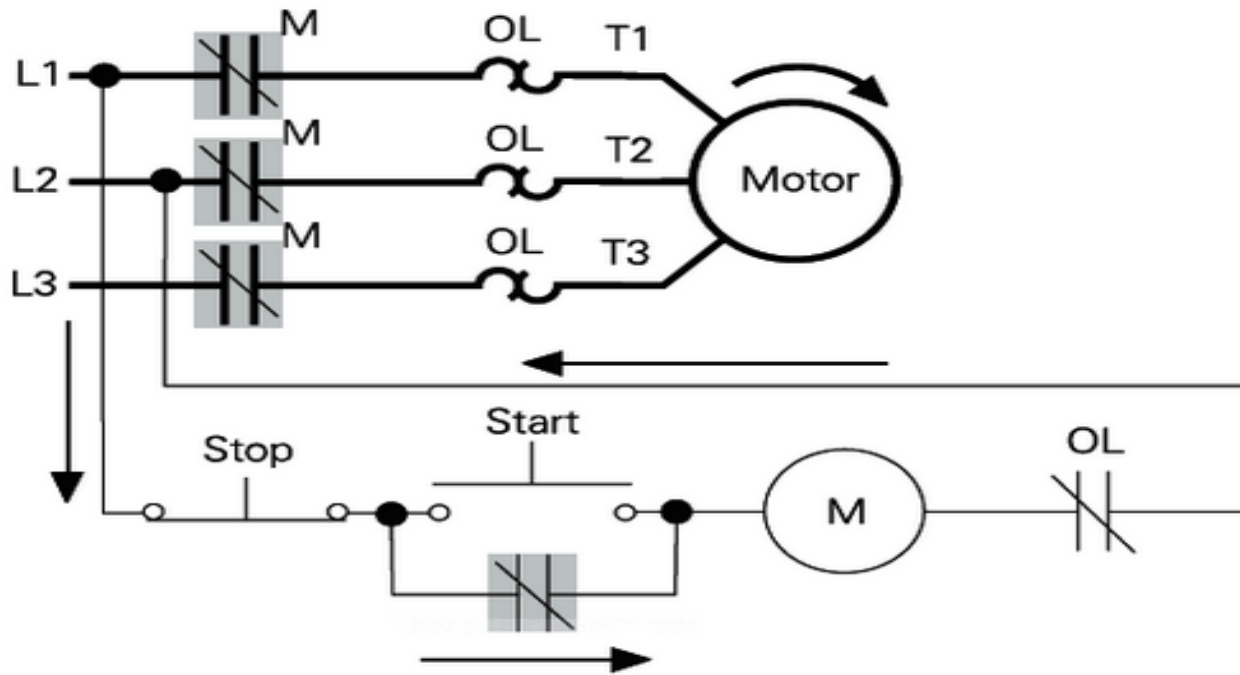
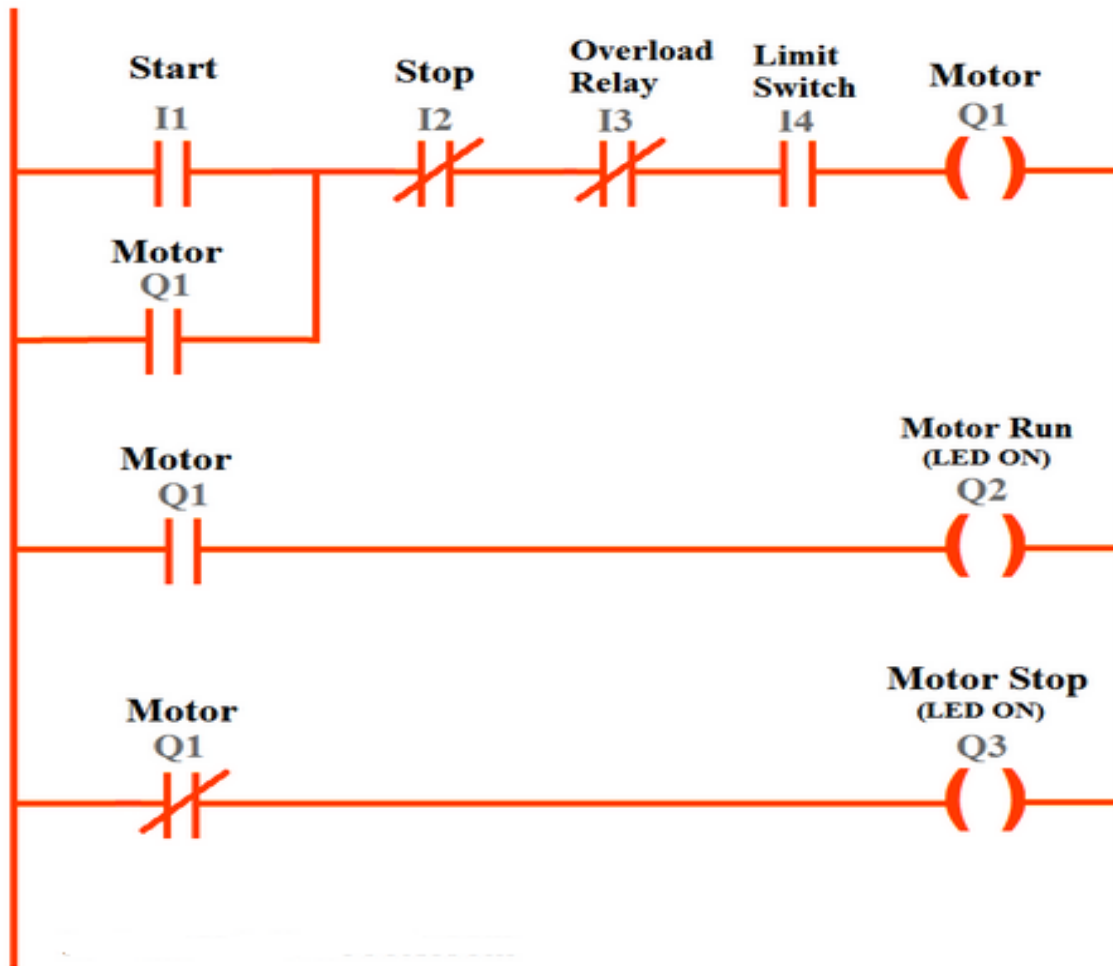


Figure (1): Motor Electrical Schematic



Ladder (1): Ladder diagram for Motor starter

- **Observation:-**

1. **Start Button I1:**

Normally open contact (Make contact) is used because the motor should only **start when the button is pressed**.

2. **Stop Button I2 :**

Normally close (break contact) contact is used because the button should **normally be closed** or high so that the motor keeps on running. It should open when the button is pressed. It is opposite to start push button.

3. **Overload relay I3 :**

In normal condition, this relay should allow the motor to rotate so normally close contact is selected for it. In case of overload it will stop the motor by opening its contact.

4. **Limit switch I4 :**

The motor should only rotate when the limit switch is closed therefore normally open contact is used.

4. Output Q1, Q2, Q3 :

Relay coil Q1, Q2 and Q3 represent motor output, motor indication ON and indication OFF respectively.

ON indicator gets input from normally open input which depends upon output Q1. OFF indicator is fed by normally close input which depends upon output Q2.

5. Input Q1 (for continuous rotation):

Since it is required that once push button is pressed, motor should run continuously even if the push button is released.

EX:(2): Continuous bottle filling system

- **Objective:**
 1. will implement a control program that detects the position of a bottle via a **limit switch** then waits for 0.5 secs.
 2. and then fills the bottle until a **photodetector** detects the filled condition of the bottle.
 3. After the bottle is filled ,the **buzzer** sounds and the control program will again wait for 0.7 secs. before moving to the next bottle.
 4. Until the limit switch signals ,the feed motor,**M1** runs while there are fixed rollers which carries the filled bottles. Motor,**M2** keeps running after the process has been started.

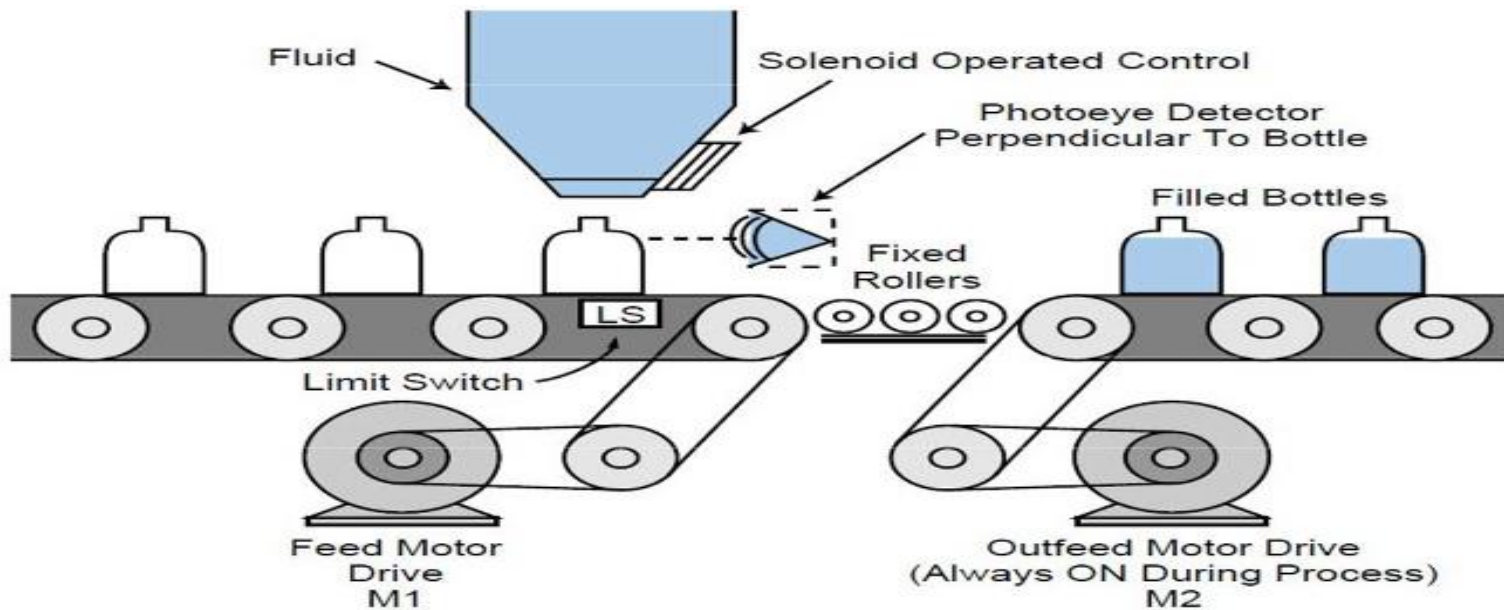
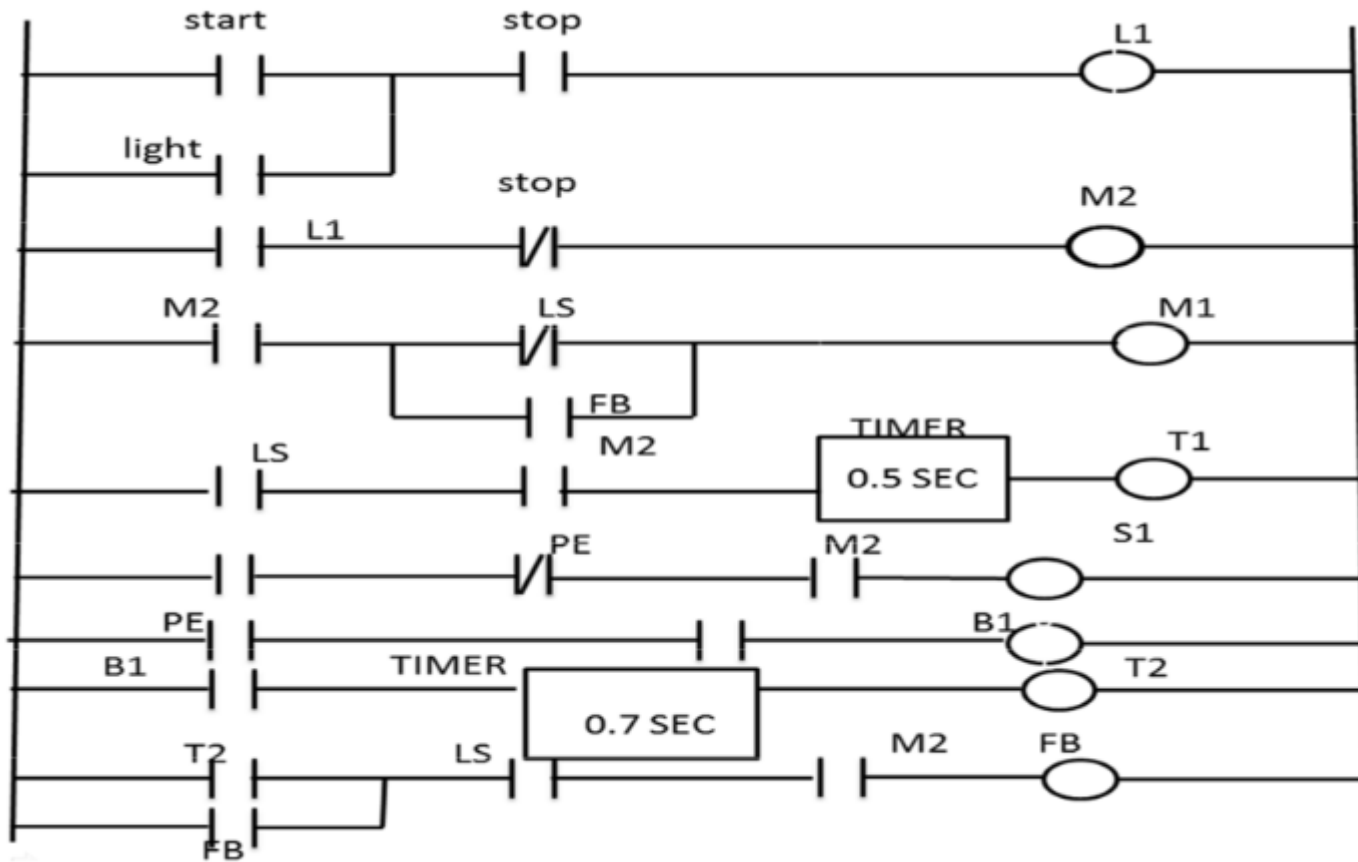


Figure (2): Bottle filling system

Inputs	address
Start	I0:15
Stop	I1:15
Limit switch(LS)	I2:15
Photo detector(PE)	I3:15

Outputs	address
Feed motor(M1)	O0:15
Outfeed motor(M2)	O1:15
Solenoid valve(S1)	O2:15
Light(L1)	O3:15
Buzzer(B1)	O4:15

Table (2): Inputs and outputs employed



ladder (2): ladder diagram for bottle filling system

- **Observation:-**

1. Once the start button is pressed the green light (L1) turns ON and remains ON until stop button is pressed. As light turns ON outfeed motor(M2) starts running.
2. After M2 runs and if either limit switch(LS) has not signaled or filled bottle condition is fulfilled motor(M1) starts.
3. After limit switch has signaled timer,T1 gets activated.
4. After T1 gives done (DN) signal and photo eye detector (PE) is disabled ,solenoid valve gets in operation. As PE signals solenoid stops and buzzer(B1) sounds after which timer,T2 gets enabled which stops the process for 0.7 seconds.
5. Once the filled bottle condition is activated the cycle starts again.

Thank You
For Your Attention



*Mohamed Ahmed
Ebrahim*