

# PLC

**P**rogrammable

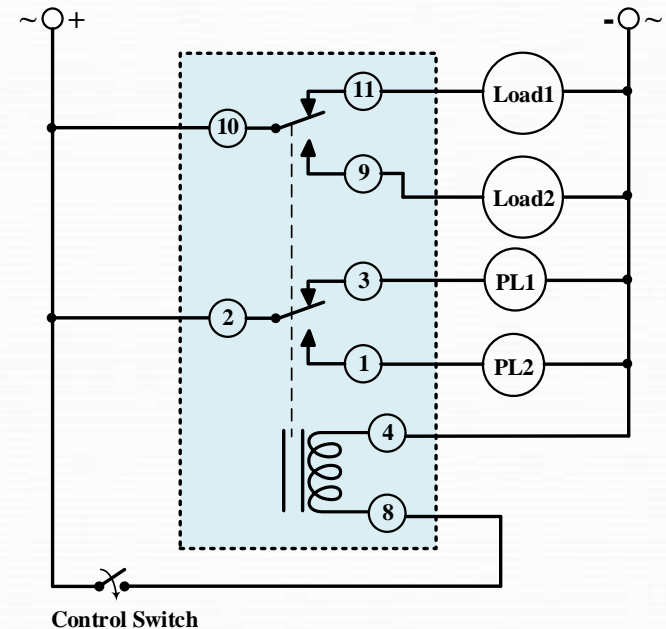
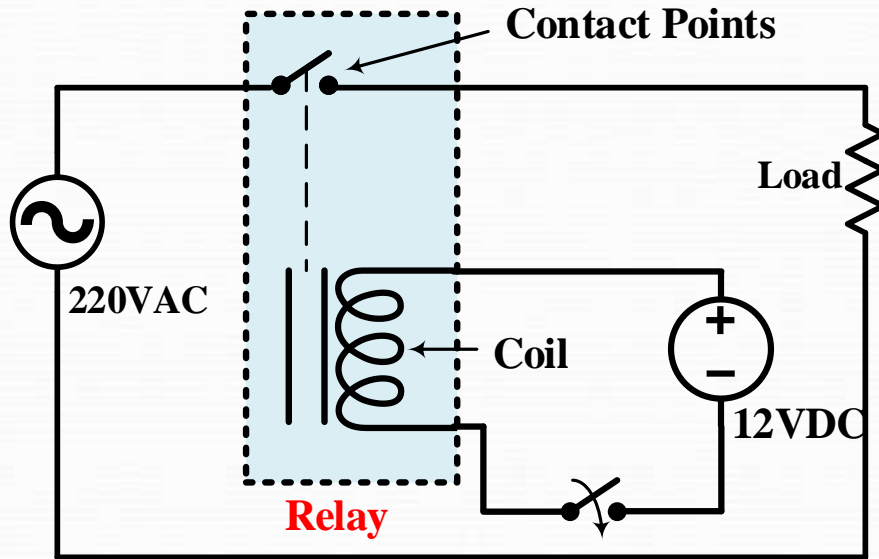
**L**ogic

**C**ontroller

**PLCs are small computers**, dedicated to automation tasks in industrial environment

# Control Relay

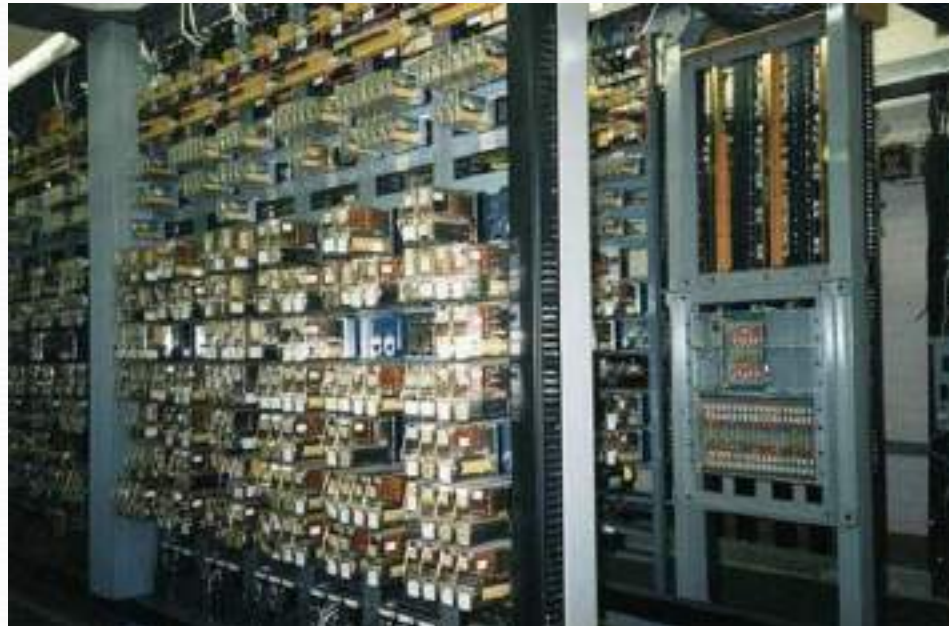
- A Control Relay is an electromagnetic switch aims to closing and opening the circuits electromechanically.



# Solid State Logic Circuits


- A solid state logic circuit is usually made **from arrangements of transistors.**
- Using just transistors, we can build logic gates (AND, NAND, OR, NOR, MUX etc.) as well as memory devices (flip-flops, registers, etc.) and from there on, the combinations, applications, and implementations are endless.
- Solid-state logic circuits are **small, reliable, low cost,** and **can operate at very high speeds** with a **high life expectancy.** In the average control circuit, the inputs may change from high to low and vice versa several times in a second.

- Before the advent of solid-state logic circuits, logical control systems were designed and built exclusively around electromechanical control relays.
- A relay control panel is comprised of **a single to thousands of relays.**



# The Need for PLCs

- **Hardwired panels were consuming very long time to wire, debug and change.**
- **GM identified the following requirements for computer controllers to replace hardwired panels.**
  - **Solid-state not electromechanical**
  - **Easy to modify input and output devices**
  - **Easily programmed and maintained by plant electricians**
  - **Be able to function in an industrial environment**



Relays are far from obsolete in modern design, but have been replaced in many of their former roles as logic-level control devices, relegated most often to those applications demanding high current and/or high voltage switching.

# What is a PLC ?

- **PLCs** are often defined as miniature industrial computers that contain hardware and software used to perform control functions.
- **PLCs** are designed for **multiple arrangements** of digital and analog inputs and outputs with extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact.

## The most important and essential characteristics of a PLC that portray its unique aspects are:

- 1) It is field programmable by the user.
- 2) It contains programmed functions.
- 3) It scans memory and I/O in a deterministic manner.
- 4) It provides error checking and diagnostics.
- 5) It could be monitored.
- 6) It is packaged appropriately.
- 7) It has general purpose suitability.

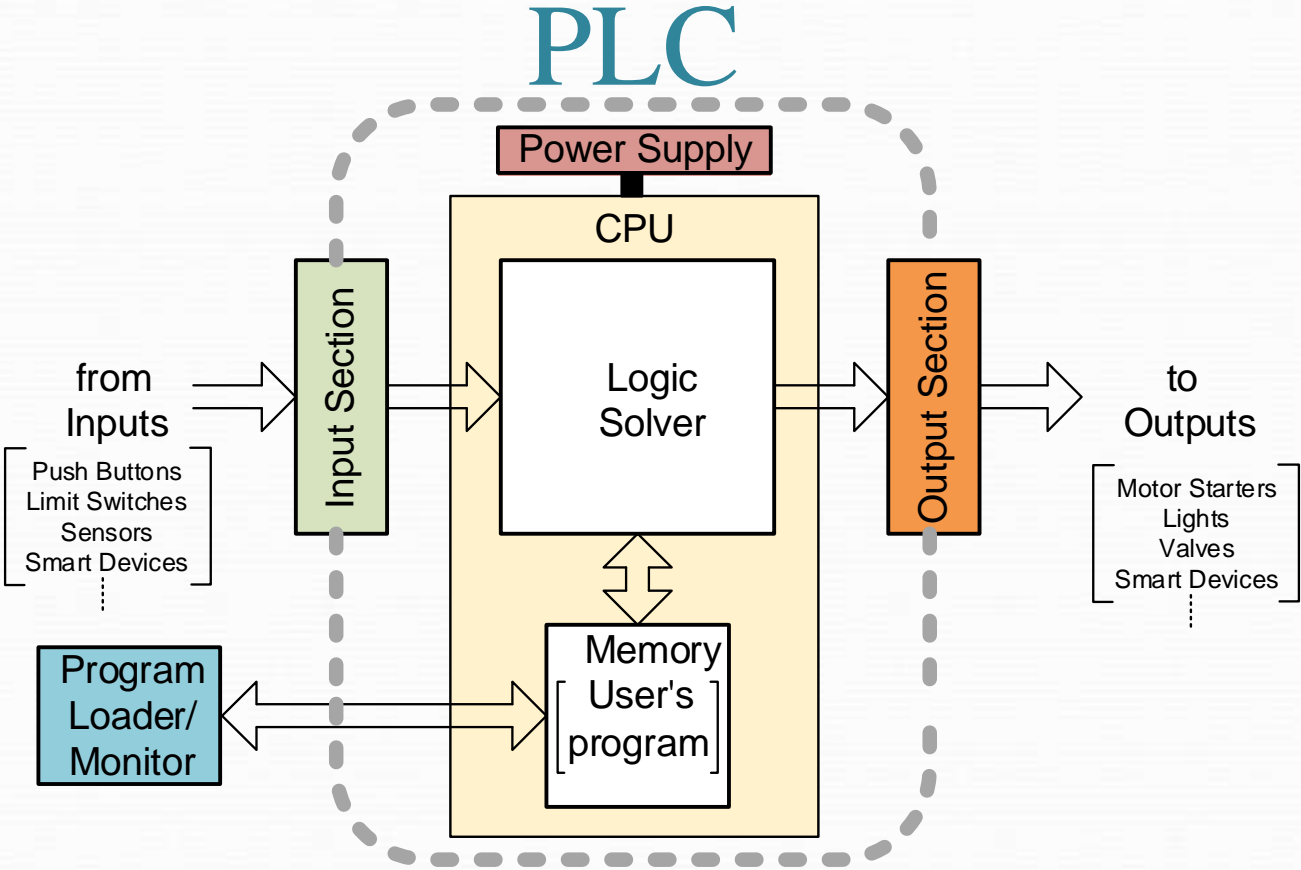


# **Advantages of PLCs**

**In addition to cost savings, PLCs provide many other benefits including:**

- 1) Increased Reliability.**
- 2) More Flexibility.**
- 3) Lower Cost.**
- 4) Communications Capability.**
- 5) Faster Response Time.**
- 6) Easier to Troubleshoot.**

# PLC Basics



# **PLC Basics**

**cont.**

**All the PLC components get their required power from the PLC power supply.**

**External power supply is usually used to provide the external input / output devices with the necessary power to operate them.**

**The PLC power supply could be used to power the external small input / output devices like sensors and indicators.**

# **PLC Basics**

**cont.**

**The logic solver reads these inputs and decides what the output states should be, based on the user's program logic.**

**The input modules convert the high-level signals that come from the field devices to logic-level signals that the PLC's processor can read directly.**

**The output modules convert the logic-level output signals from the logic solver into the high-level signals that are needed by the various field devices.**

# **PLC Basics**

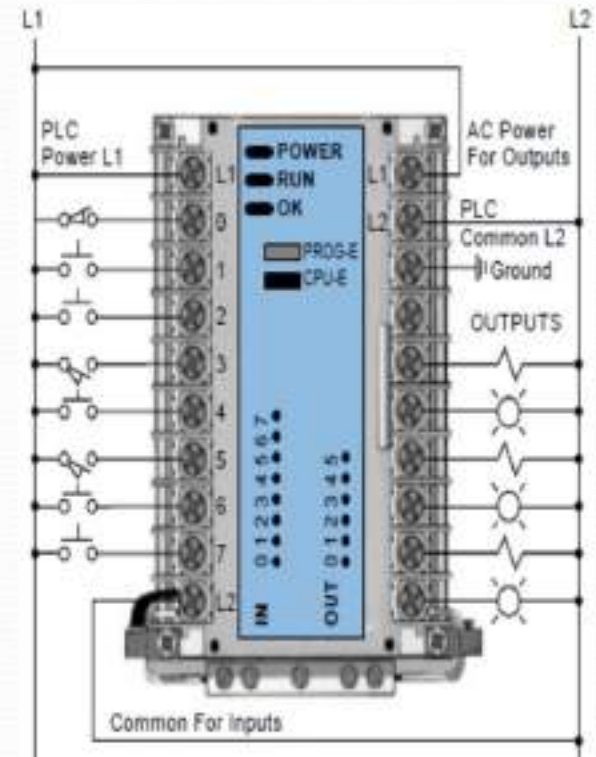
**cont.**

**The program loader is used to enter the user's program into the memory or change it and to monitor the execution of the program.**

# I / O Configuration

## 1) Fixed I / O

- Is typical for small PLCs
- Comes in one package, with no separate removable units.
- The CPU and I/O are packaged together.
- Lower in cost, but lacks flexibility.
- For some models, if any part in the unit fails, the whole unit has to be replaced.



# I / O Configuration cont.

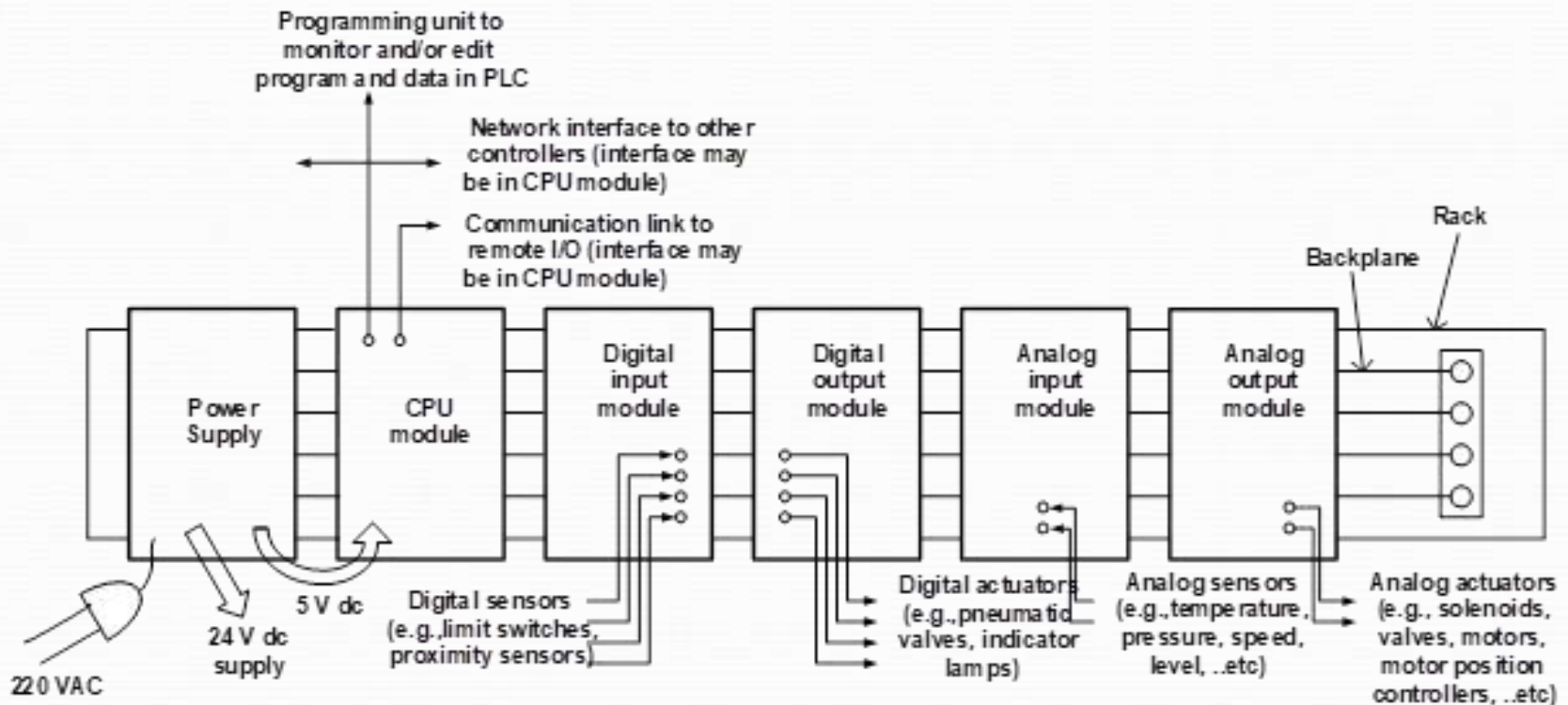
## 2) Modular I / O



- Is divided by compartments into which separate modules can be plugged.
- This feature greatly increases the unit's flexibility.
- All the modules are available and could be chosen and mixed in any way.

# I / O Configuration cont.

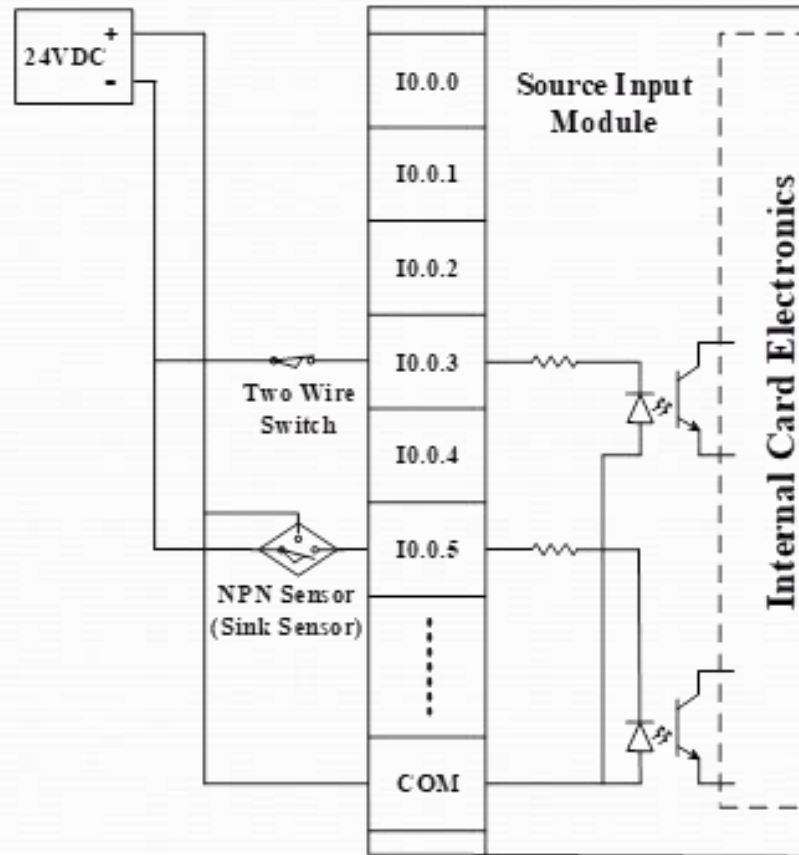
## 2) Modular I / O





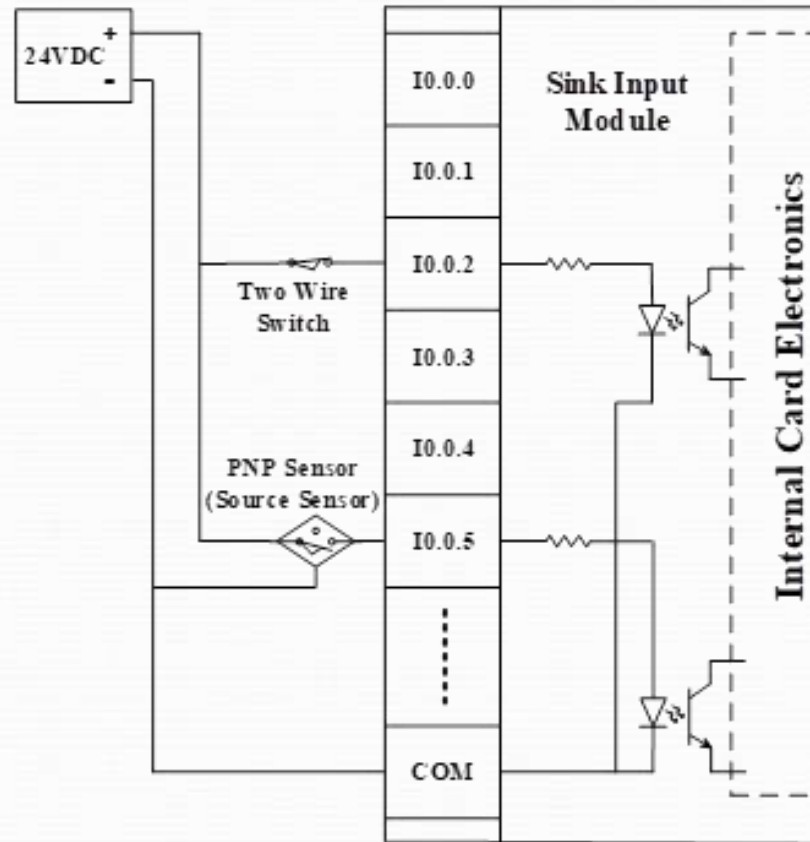
# Input Modules Wiring

- 1) Wiring of Sink Sensor and Two Wire Sensor to a PLC Source Inputs Module.



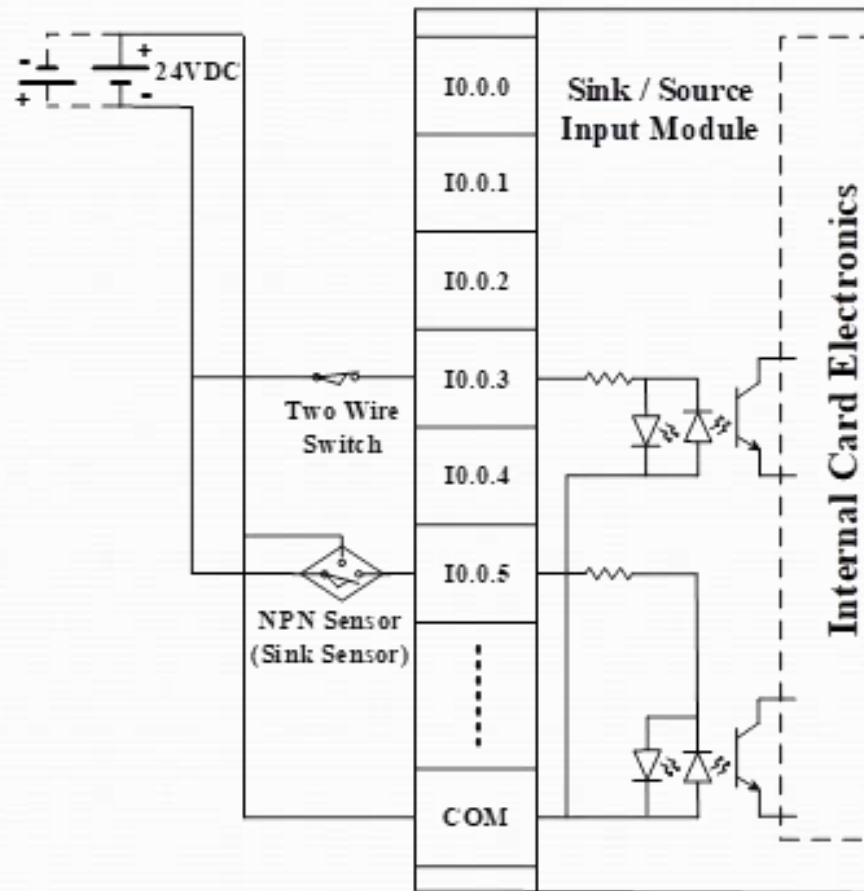
# Input Modules Wiring cont.

2) Wiring of Source Sensor and Two Wire Sensor to a PLC Sink Inputs Module.

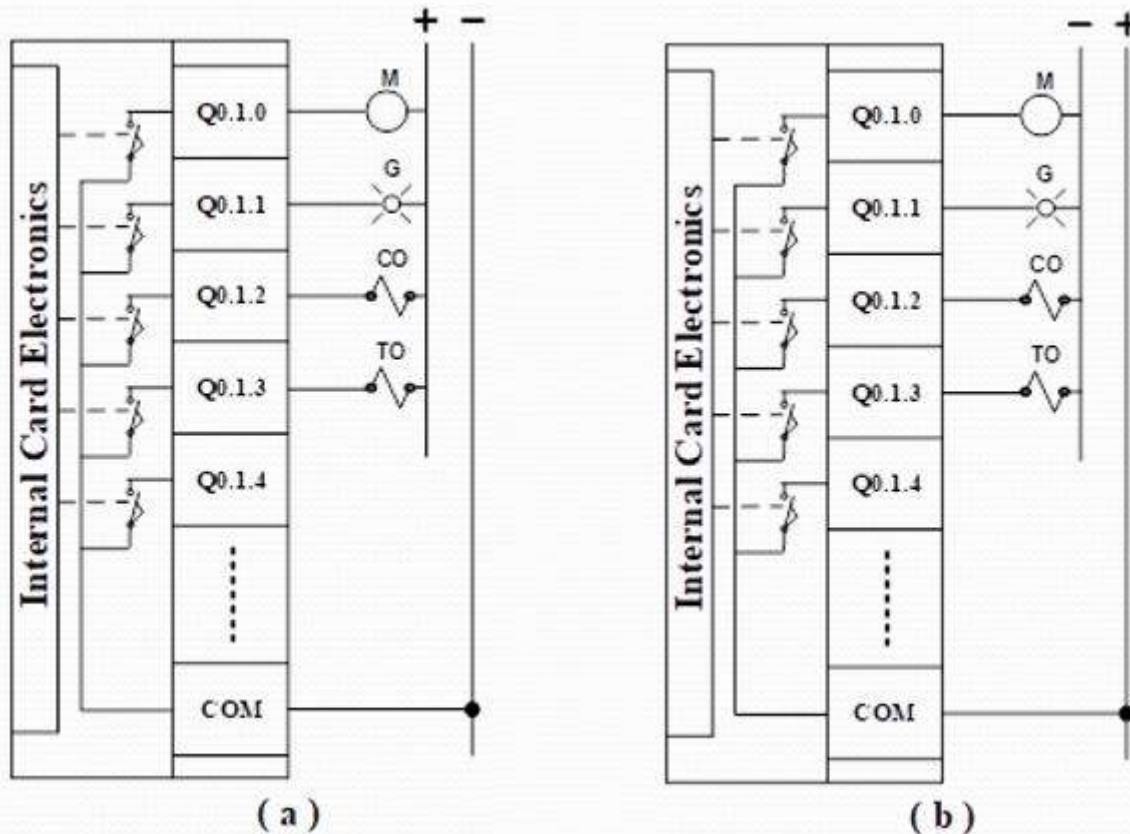


# Input Modules Wiring cont.

## 3) Wiring of Sink / Source PLC Inputs Module.



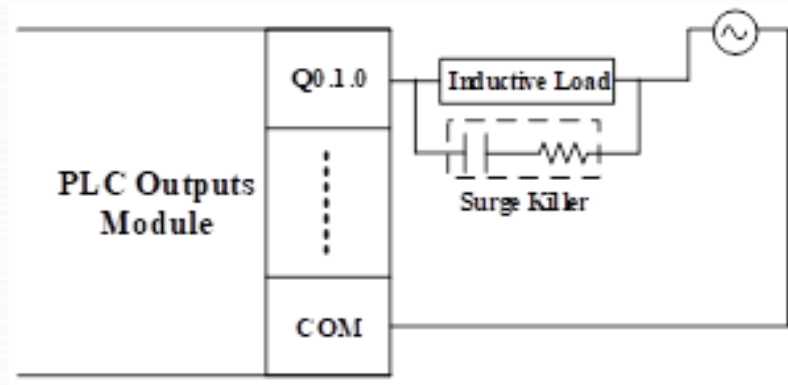
# Wiring of Output Modules



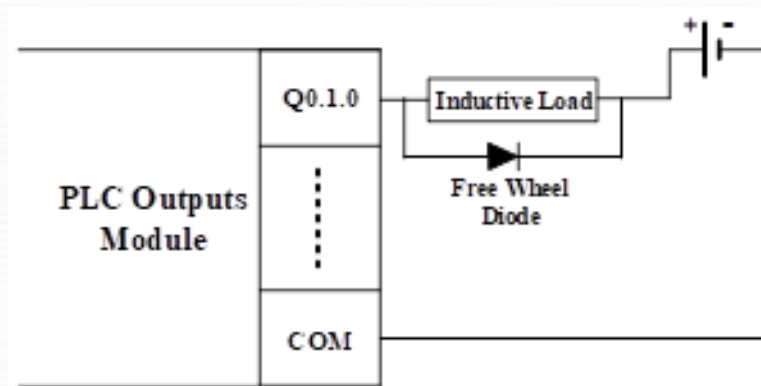
a) Wiring of a PLC Sink Outputs Module

b) Wiring of a PLC Source Outputs Module

# Wiring of Inductive Loads



## 1) Wiring of Inductive Load to AC Output Module



## 2) Wiring of Inductive Load to DC Output Module

# PLC Software

## □ Operating System and Application Programs

The operating system program causes **the PLC to:**

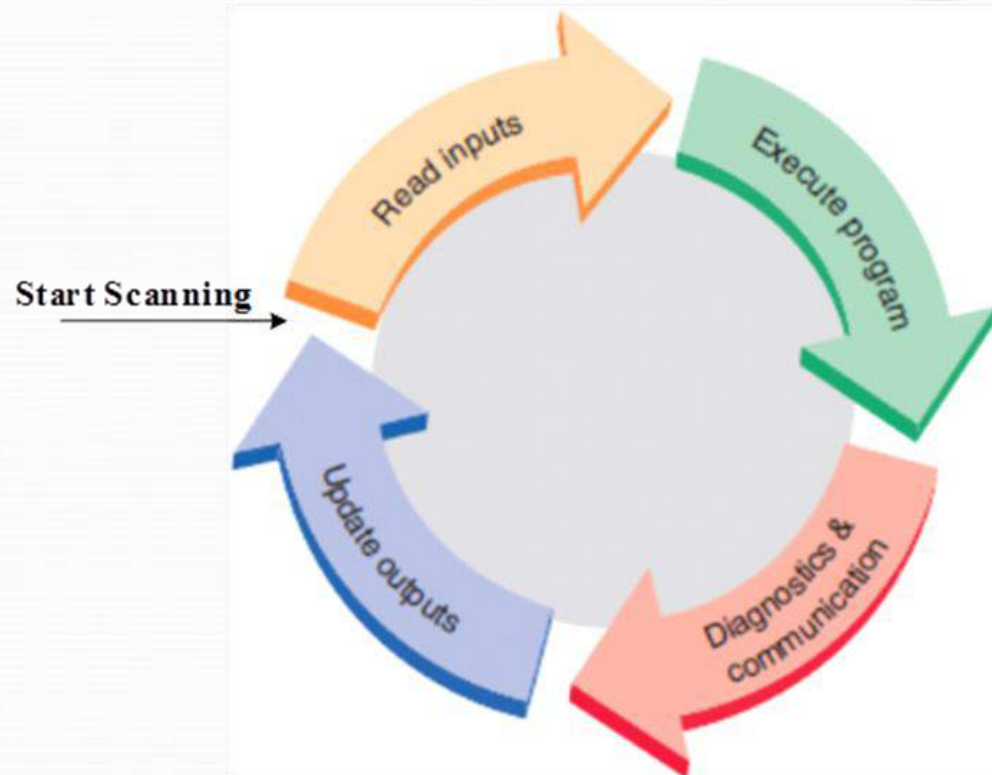
- **start** when power is **turned on**,
- **to run the user-program** when the PLC is switched **into run mode**,
- **to respond to the user commands** by running the appropriate application programs.

The **application programs** allow the user to enter programs and data into the PLC's memory.

## PLC Software cont.

The PLC retains its operating system, application programs, user programs, and some data **in retentive memory** while the PLC is turned off and even when disconnected from the power supply.

The PLC operating system makes the PLC run user-programs very differently from the way other computers run user-programs.



The operating system executes an **initialization step once each time** the PLC changed **into run mode**, and then repeatedly makes the PLC executes a **scan cycle sequence as long as the PLC remains in run mode**.



Every time the PLC finishes one scan cycle and starts another, the operating system also **restarts a watchdog timer**.

The watchdog timer runs while the scan cycle executes.

If the watchdog timer reaches its pre-set value before being restarted (if a scan cycle **takes unusually long to complete**), **the PLC will immediately fault, and stop running**.

After faulting, the PLC usually **needs operator intervention** before it can resume running.

# PLC User-Programs

- The user program is created and downloaded to the CPU by a programmer using a programming unit, which can then be disconnected from the PLC.
- The user-program remains in the PLC's memory until a programming unit is used to change it.
- The user program contains all the functions required to process the specific automation task.

# PLC User-Programs cont.

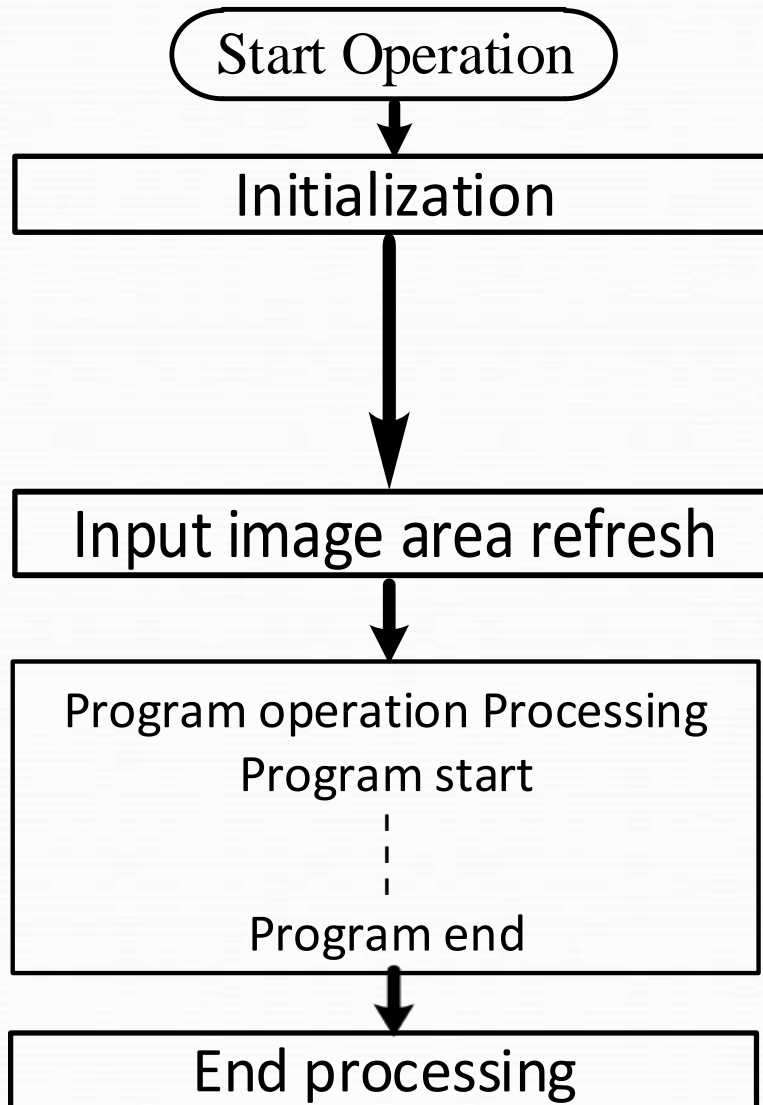
The tasks of the user program include:

- Specifying the conditions for a restart (cold restart) or (warm restart) on the CPU (for example, **initializing signals with a particular value**).
- Processing process data (for example, generating logical links of binary signals, fetching and evaluating analog signals, specifying binary signals for output, output of analog values).
- Reaction to interrupts.
- Handling disturbances in the normal program cycle.

# Processing Methods

- 1) Cyclic operation
- 2) Event driven interrupt operation method
- 3) Time driven interrupt operation method

# 1) Cyclic operation



Stage for the start of a scan processing.

It is executed only one time when the power is applied or reset is executed.

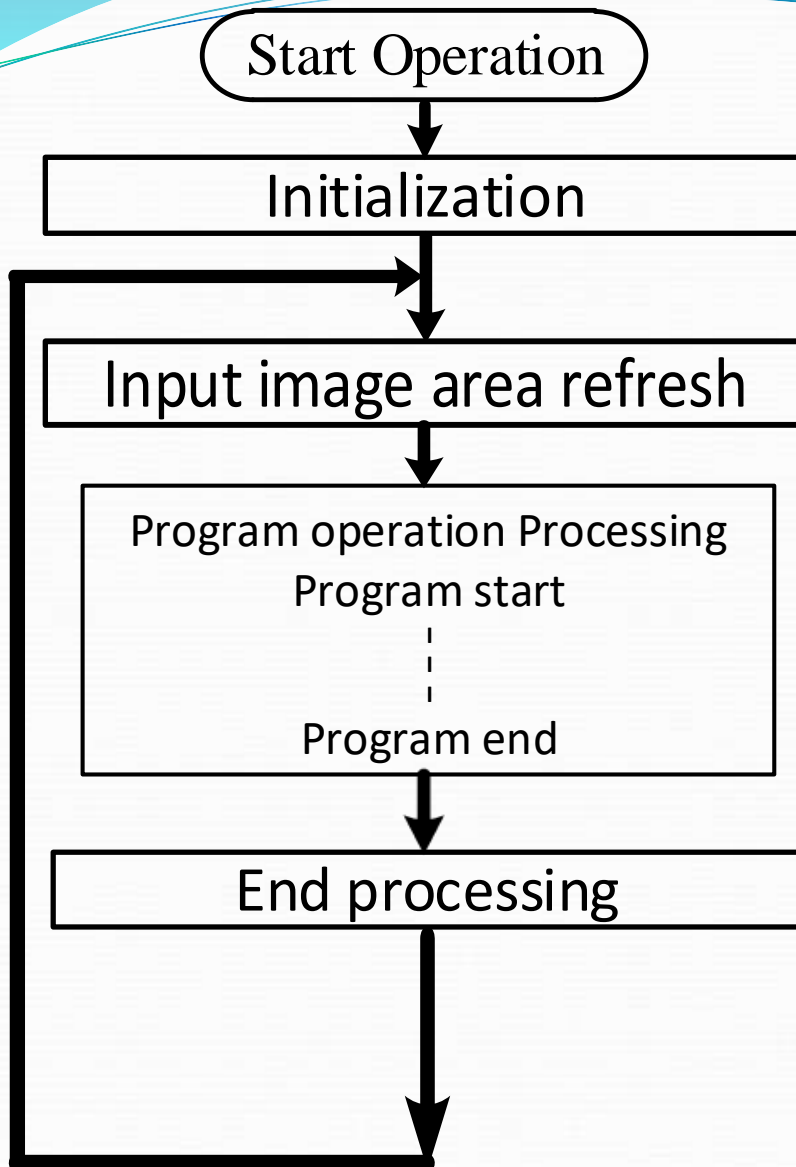
It executes the following processing:

1) Reset the I/O modules. 2) Execution of self-diagnostics 3) Data clear. 4) I/O address allocation or type registration.

Input module conditions are read and stored in the input image area before program operation processing of the program.

Program is executed sequentially from the first step to the last step.

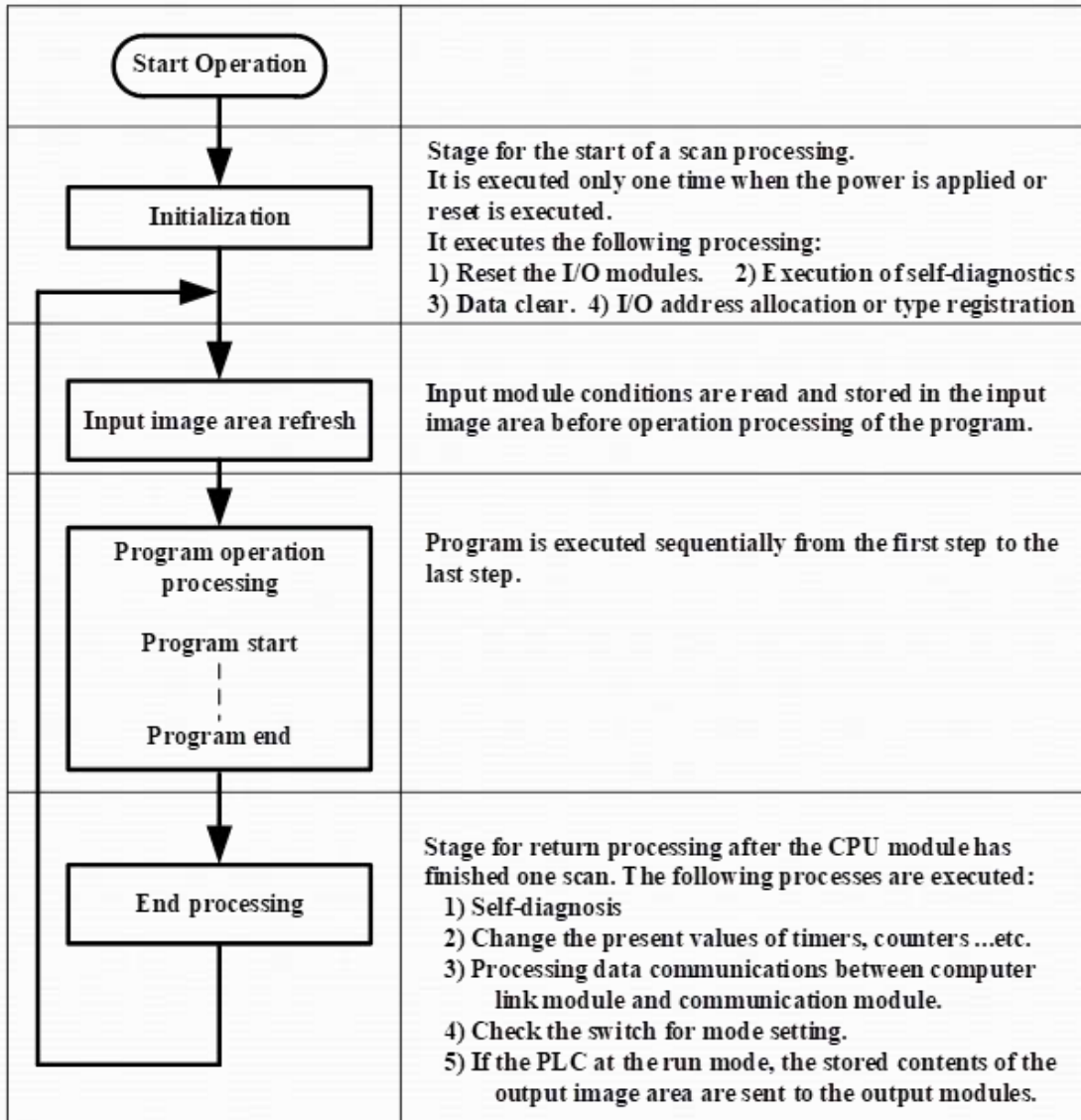
Stage for return processing after the CPU module has finished one scan.



Stage for return processing after the CPU module has finished one scan.

- 1) Self diagnosis.
- 2) Change the present values of timers, counters ...etc.
- 3) Processing data communication between computer and link module and communication module.
- 4) Check the switch for mode setting.
- 5) If the PLC at the “run” mode, the stored contents of the image area are send to the output modules

# 1) Cyclic operation



# Processing Methods cont.

## 2) Event driven interrupt operation method

- If a situation occurs which is requested to be **urgently processed** during execution of a PLC program, this operation method processes immediately the operation which corresponds to **interrupt program**.
- The signal which informs the CPU module of those urgent conditions is called **interrupt signal**.
- The CPU module has two kind of interrupt operation methods, which are **internal** and **external** interrupt signal methods.



# Processing Methods cont.

## 3) Time driven interrupt operation method

In time driven interrupt operation method, operations are processed not repeatedly **but at every pre-set interval**. This operation is used to process operation with a constant cycle.

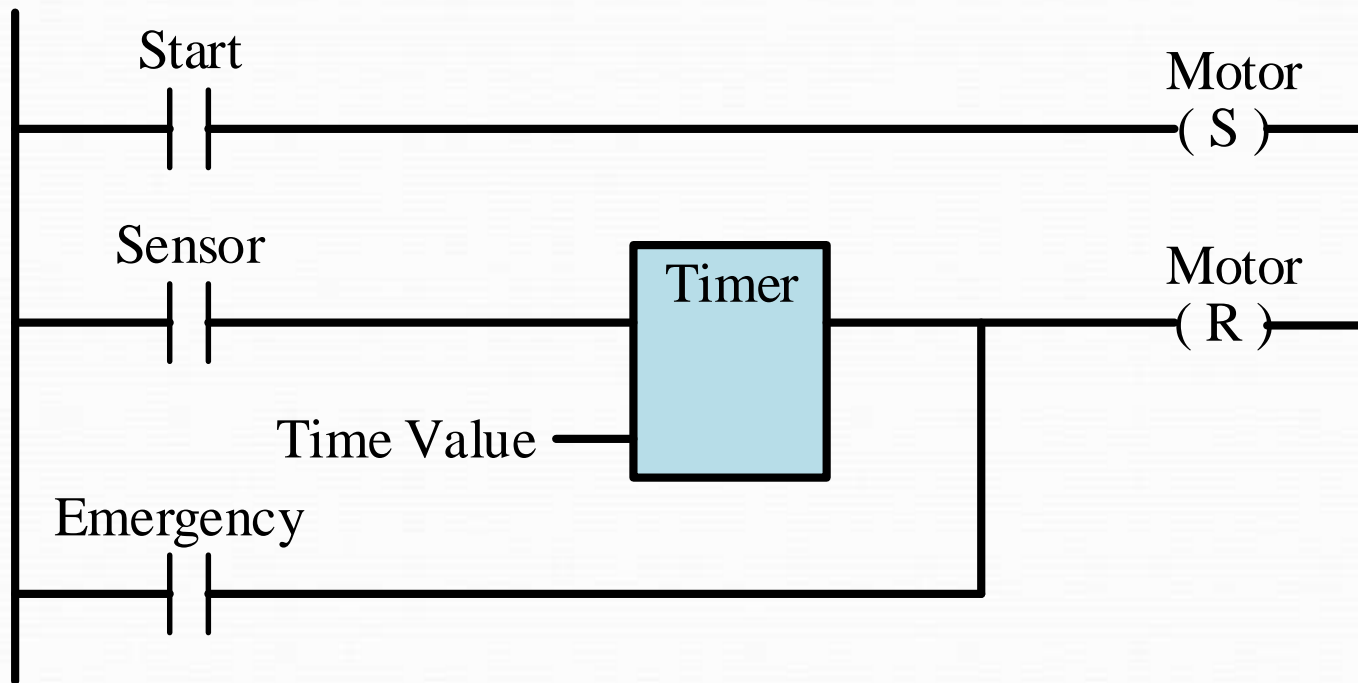
# PLC Programming Languages

The **I**nternational **E**lectrotechnical **C**ommission (IEC) is the international standards and conformity assessment body for all fields of electrotechnology.

IEC 61131-3 standard was established to standardize the multiple languages associated with PLC programming by defining the following five standard languages:

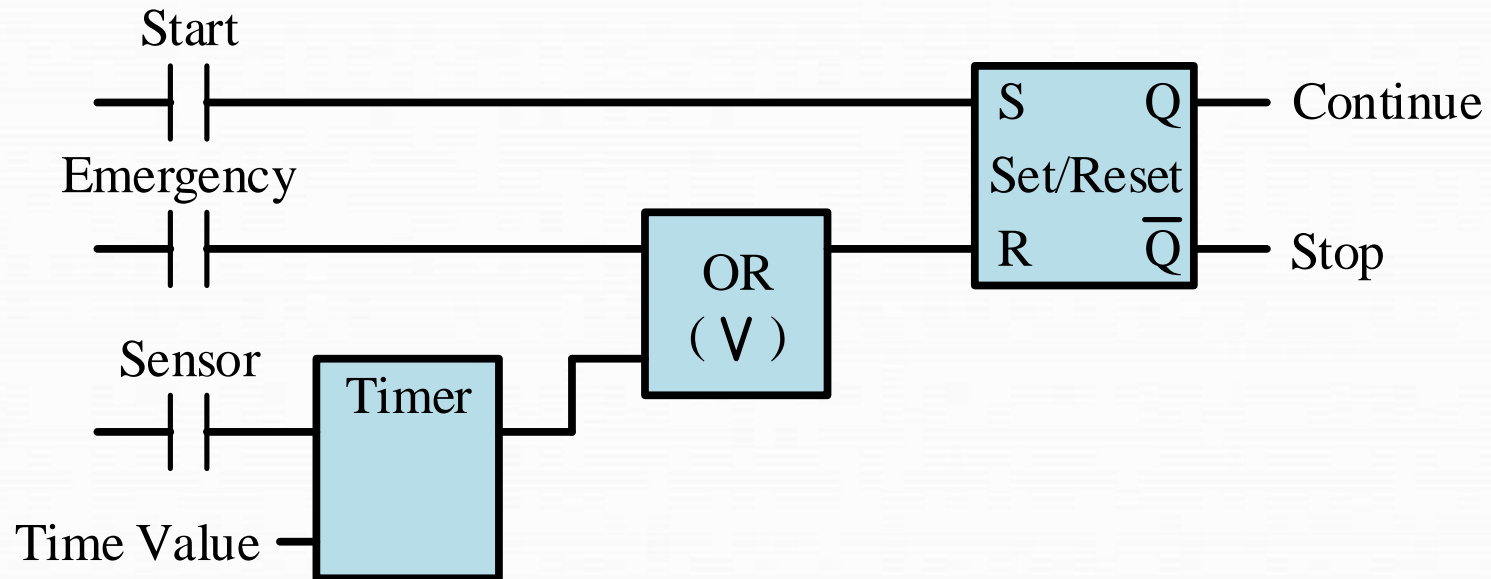
# PLC Programming Languages

**1) Ladder Diagram (LD):** It's a graphical depiction of a process with rungs of logic, similar to the relay ladder logic schemes that were replaced by PLCs.



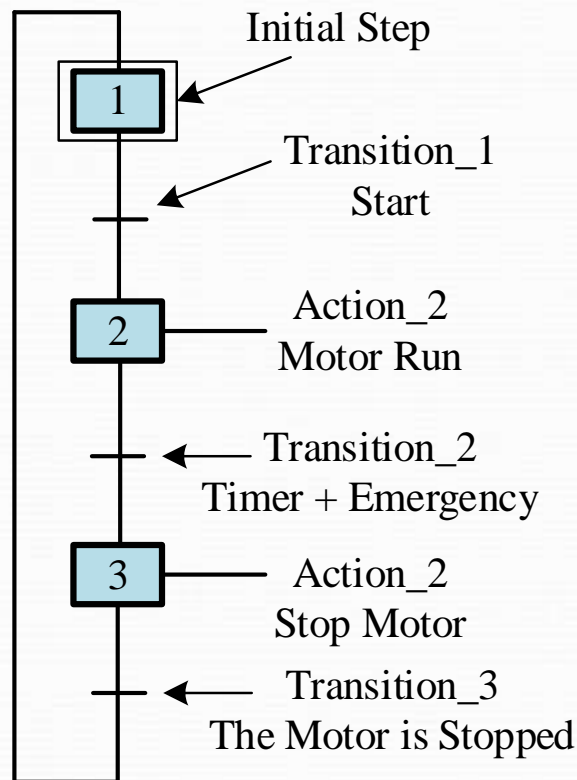
# PLC Programming Languages

2) **Function Block Diagram (FBD):** It's a graphical depiction of process flow using simple and complex interconnecting blocks.



# PLC Programming Languages

**3) Sequential Function Chart (SFC):** It's a graphical depiction of interconnecting steps, actions, and transitions.



# PLC Programming Languages

**4) Instruction List (IL):** It's a low-level, text-based language that uses mnemonic instructions.

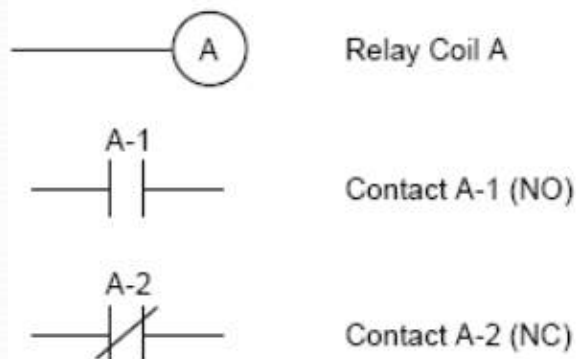
<u>Instruction</u>		<u>comment</u>
LD	Start	(* start button is pressed*)
ANDN	Stop	(* Stop button unpressed*)
ST	RUN	(* Run Motor*)

**5) Structured Text (ST):** It's a high-level, text-based language such as BASIC, C, or PASCAL specifically developed for industrial control applications.

# PLC Logic & Hardwired Relay Logic - I

Programmable Logic controller contacts and electromechanical relay contacts operate in a very similar fashion.

Standard Configuration for relay A



PLC Configuration for relay A



# PLC Logic & Hardwired Relay Logic - I cont.

## Contact Symbols used in PLCs

- | |— Normally opened contact represents any input to the control logic.
- | / |— Normally closed contact represents any input to the control logic.
- ( )— Output represents any output that is driven by some combination of input logic. An output can be a connected device or an internal output.



## PLC Logic & Hardwired Relay Logic - I cont.

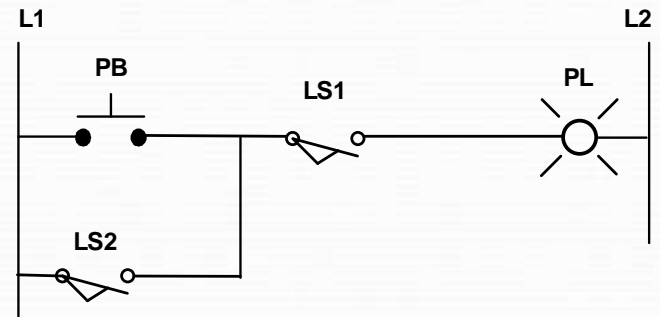
—( / )— NOT Output represents any output that is driven by some combination of input logic. An output can be a connected device or an internal output.

**Note that** a PLC can have as many normally open and normally closed contacts as desired; whereas in an electromechanical relay, only a fixed number of contacts are available.

# PLC Ladder Diagram Language

The original ladder diagrams were established to represent hardwired logic circuits used to control machines or equipment.

This figure illustrates a simple electrical ladder diagram.



Due to wide industry use, they became a standard way of communicating control information from the designers to the users of equipment.

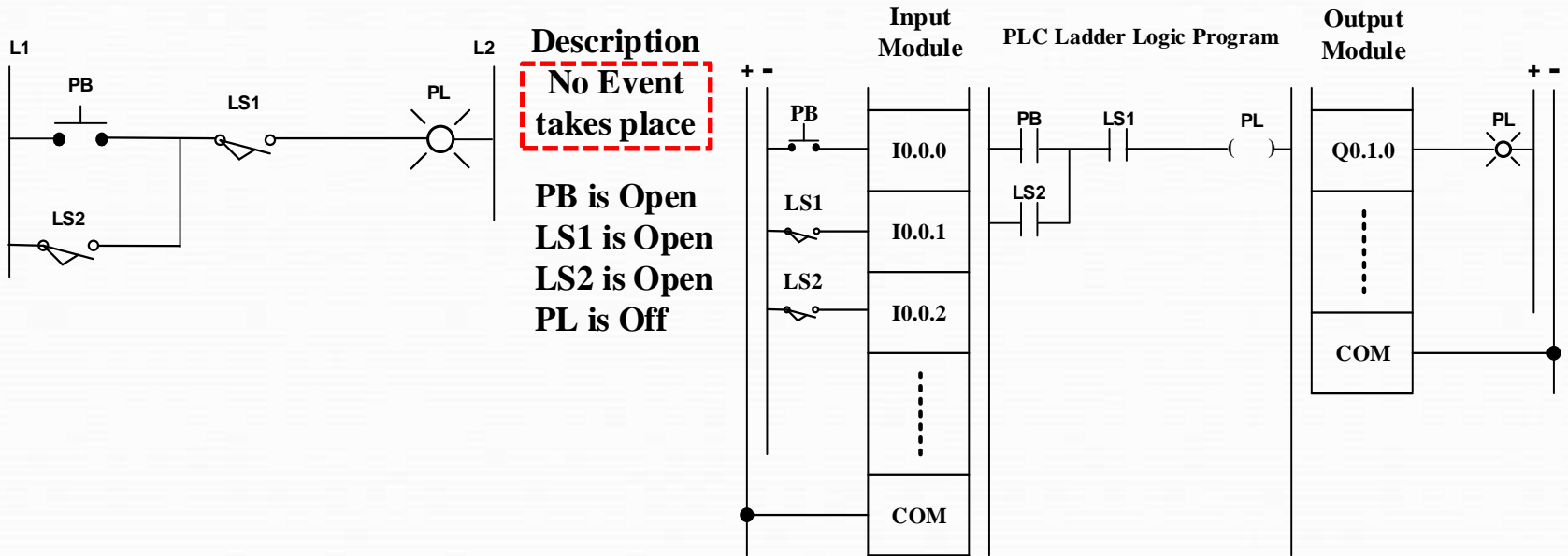
# PLC Ladder Diagram Language **cont.**

Programmable logic controllers can implement all of the “old” ladder diagram conditions **and much more.**

Their purpose is to perform these control operations in a **more reliable** manner at a **lower cost.**

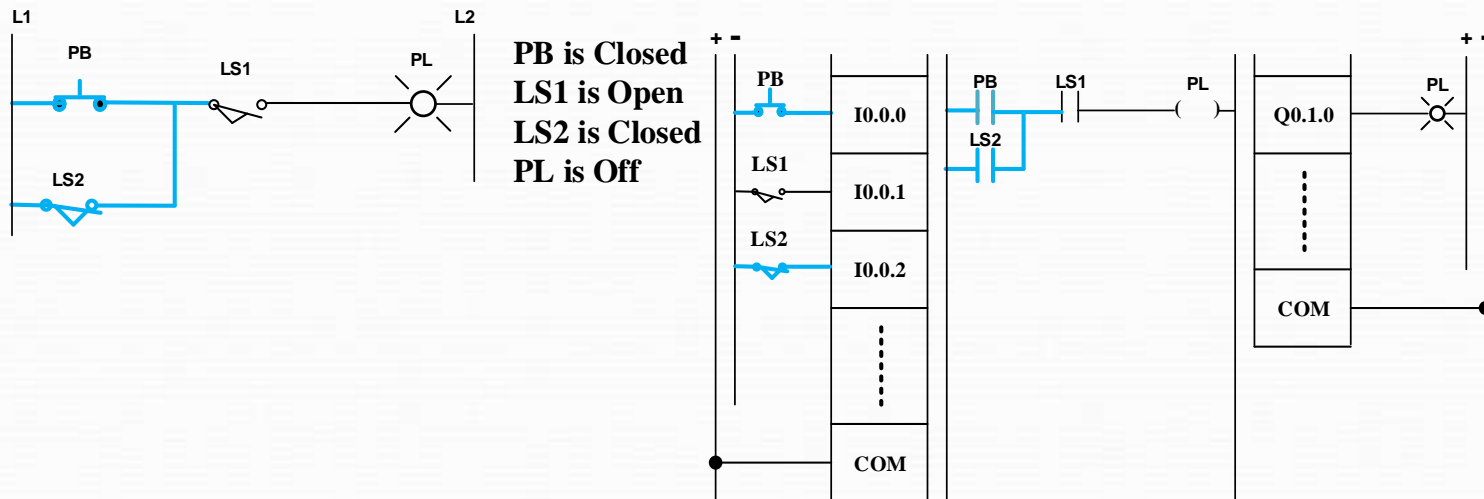
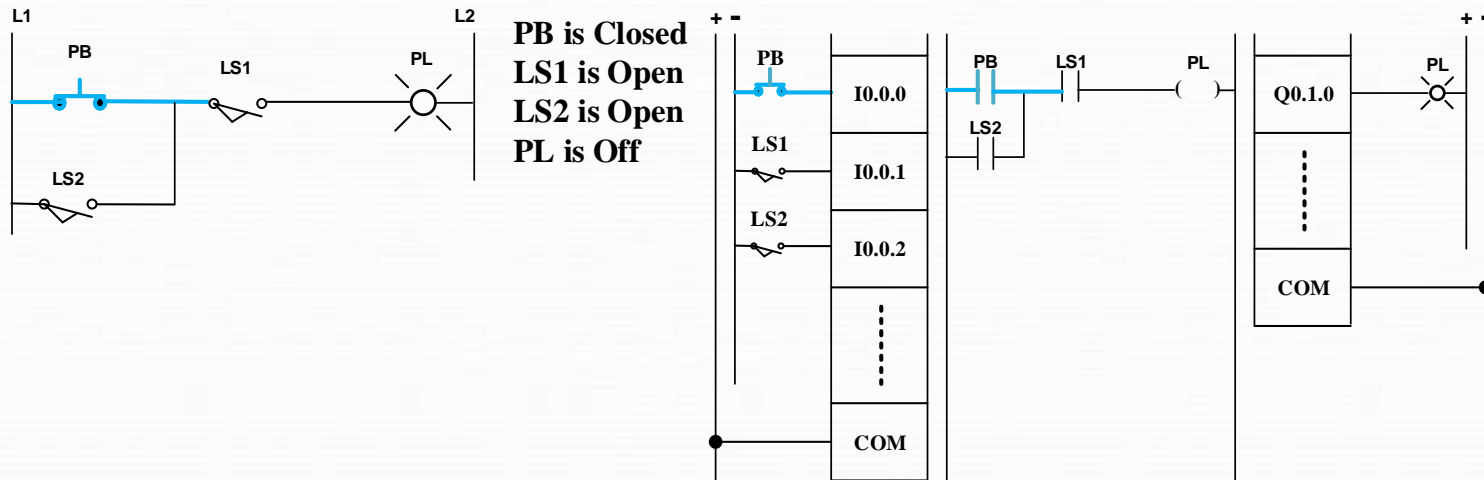
The PLC implements, in its CPU, all of the old hardwired interconnections using its software instructions.

# PLC Ladder Diagram Language cont.

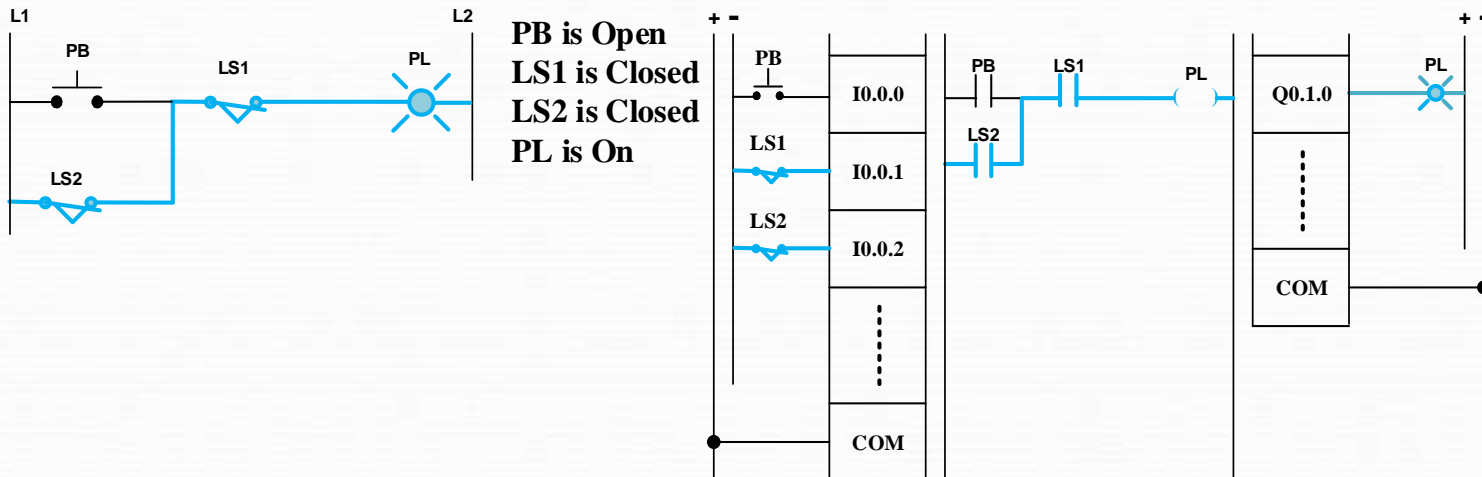
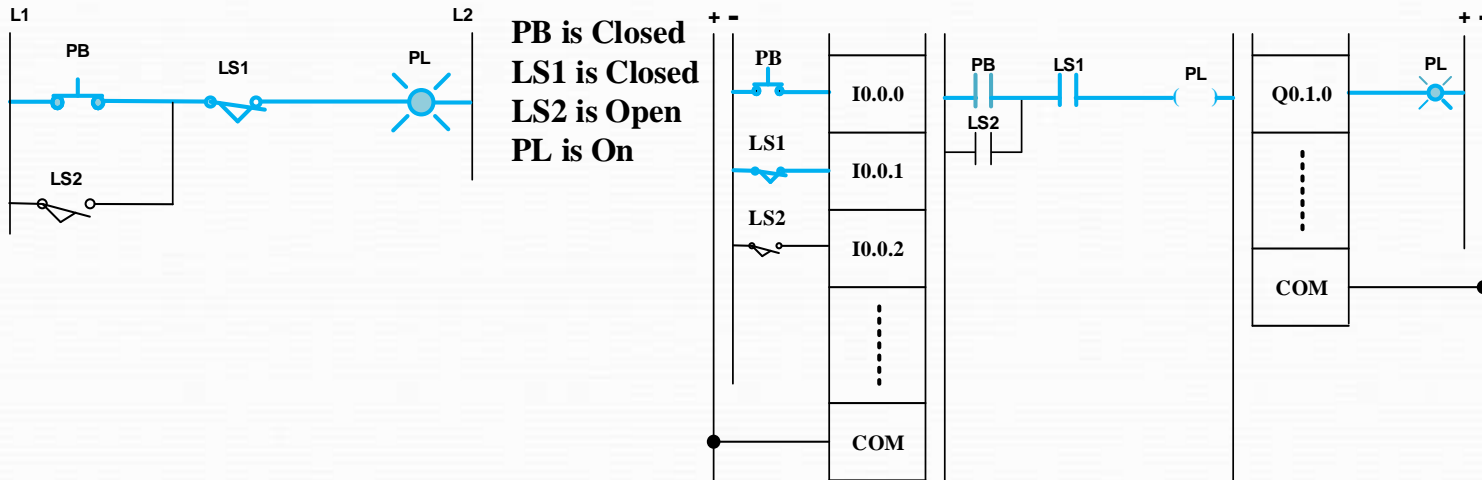


This figure illustrates the PLC transformation of the simple diagram shown in the previous figure to a PLC format.

# PLC Ladder Diagram Language **cont.**

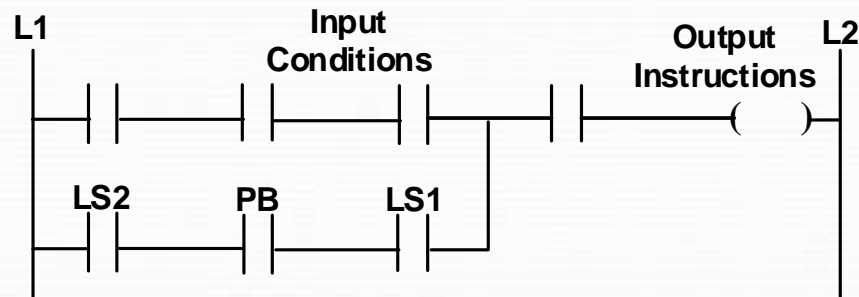


# PLC Ladder Diagram Language **cont.**



# Ladder Diagram Format

The main functions of a ladder diagram program are to **control outputs** and perform functional operations based on **input conditions**.

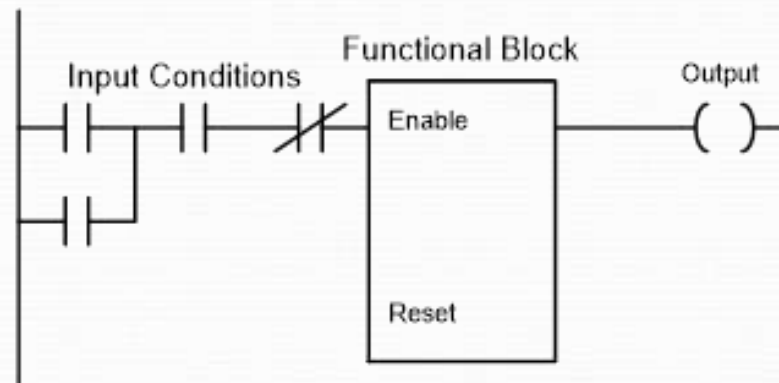


In general, a rung consists of **a set of input conditions** (represented by contact instructions) and **an output instruction at the end of the rung** (represented by a coil symbol).

# Ladder Diagram Format **cont.**

The ladder rung is TRUE when it has logic continuity.  
Logic continuity exists when power flows through the rung from left to right.

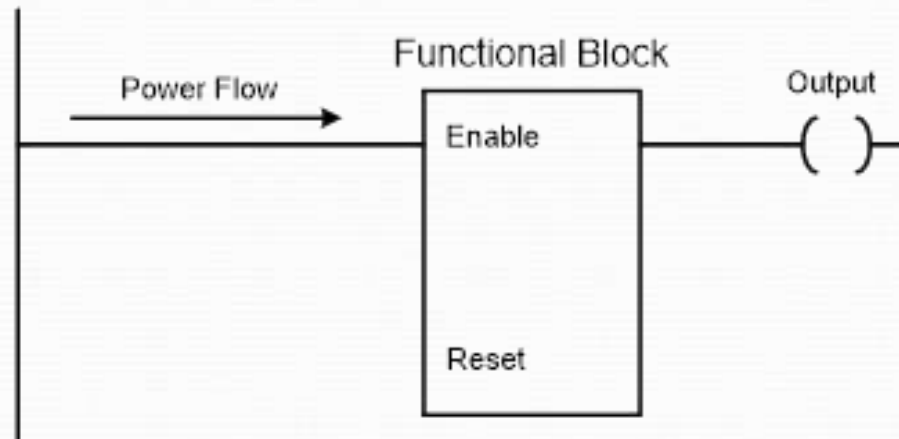
When a ladder diagram contains a functional block, contact instructions are used to represent the input conditions that **drive (or enable)** the block's logic.





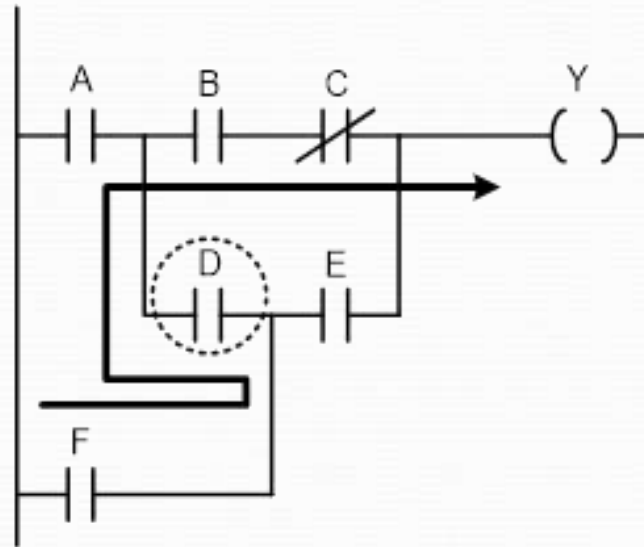
# Ladder Diagram Format **cont.**

To make a block **active at all times without any driving logic**, the user can omit all contact logic and place a continuity line in the block during programming as shown in Figure .



# Ladder Diagram Format **cont.**

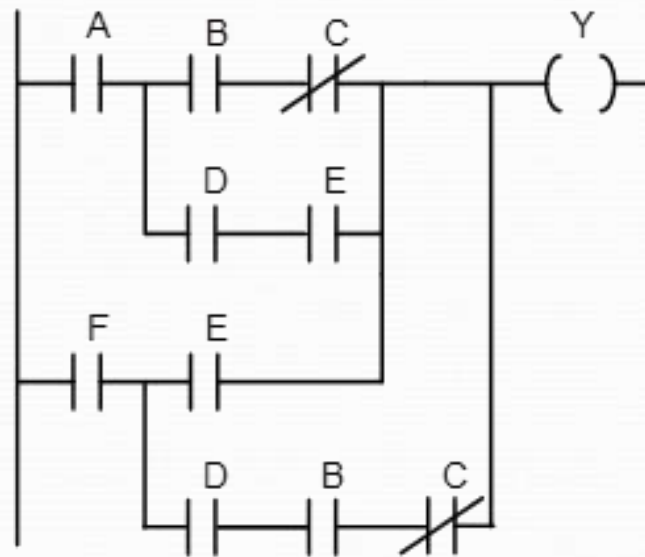
**Rule One**, which is present in almost all PLCs, prevents reverse (i.e., right-to-left) power flow in a ladder rung.



PLC logic does not allow reverse power to avoid sneak paths.

# Ladder Diagram Format **cont.**

If a PLC's logic requires reverse power flow, the user must reprogram the rung with forward power flow to all contact elements.



# Addresses Used in PLCs

Every input/output point has a unique address that can be used by the CPU.

When a field signal is connected to an input or an output interface, its address will be related to the terminal where the signal wire is connected.

# Addresses Used in PLCs **cont.**

The address for a given input/output can be used throughout the program **as many times as required** by the control logic.

This PLC feature is an advantage when compared to relay-type hardware, where **additional contacts often mean additional hardware.**

Most controllers reference input/output devices using numeric addresses with octal (base 8) or decimal (base 10) numbering.

## PLC Logic & Hardwired Relay Logic -II

**Hardwired logic** refers to logic control functions (timing, sequencing, and control) that are **determined by the way devices are interconnected.**

In contrast to PLCs, in which logic functions are **programmable and easily changed**, **hardwired logic** is **fixed** and can **be changed** only by altering the way devices are physically connected or inter-wired.

## PLC Logic & Hardwired Relay Logic –II cont.

A prime function of a PLC is to replace existing hardwired control logic and to implement control functions for new systems.

The ladder circuit connections of the hardwired relay circuit are implemented in the PLC via software instructions, thus all of the wiring can be thought of as being inside the CPU (software as opposed to hardwired).

## PLC Logic & Hardwired Relay Logic –II cont.

The logic implemented in PLCs is based on the three basic logic functions (**AND, OR, and NOT**). These functions are used **either alone** or **in combination** to form instructions that will determine if a device is to be switched on or off.

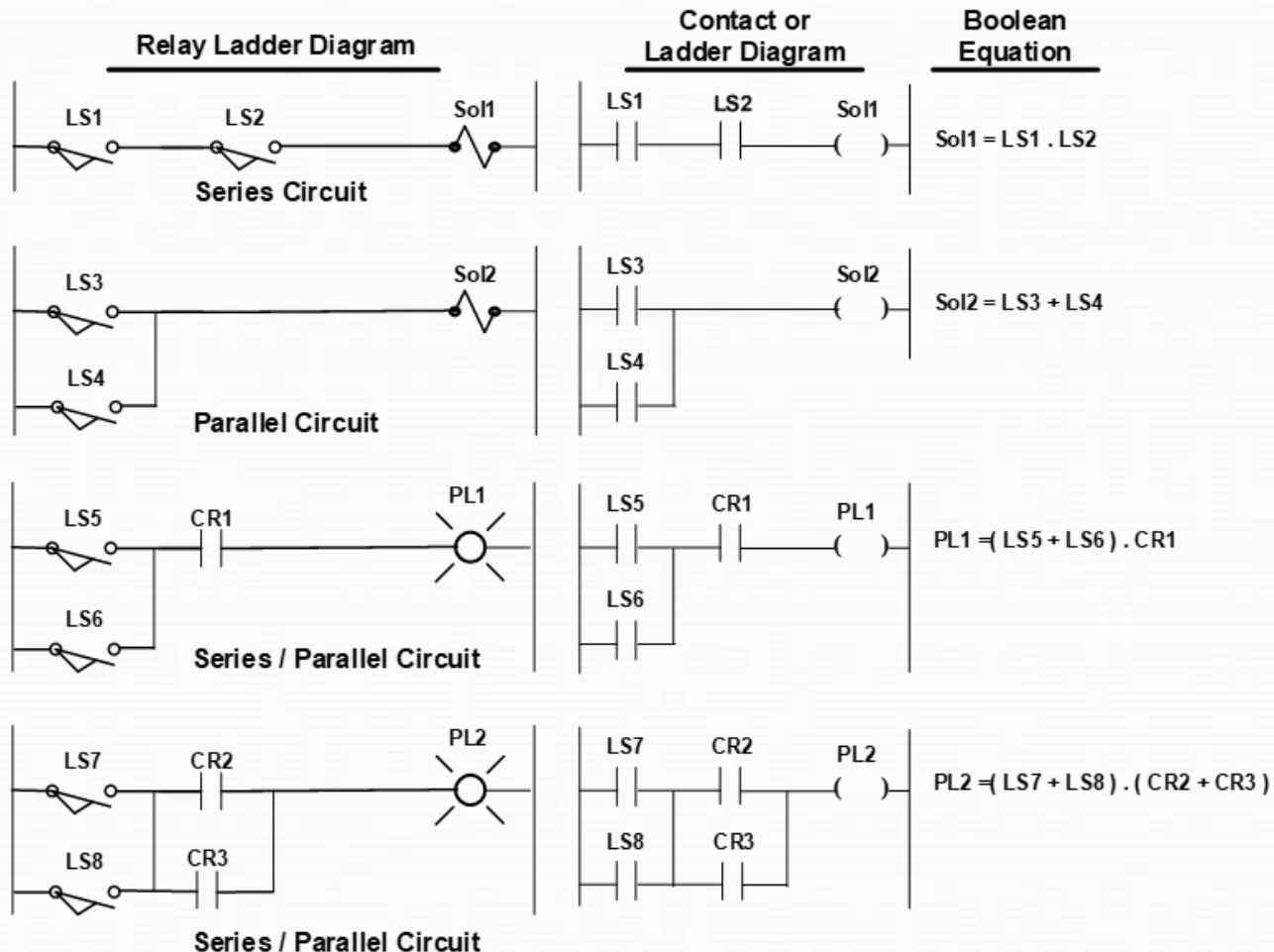
Ladder diagrams are also called **contact symbology**; since their instructions are relay-equivalent contact symbols (i.e., normally open and normally closed contacts and coils).



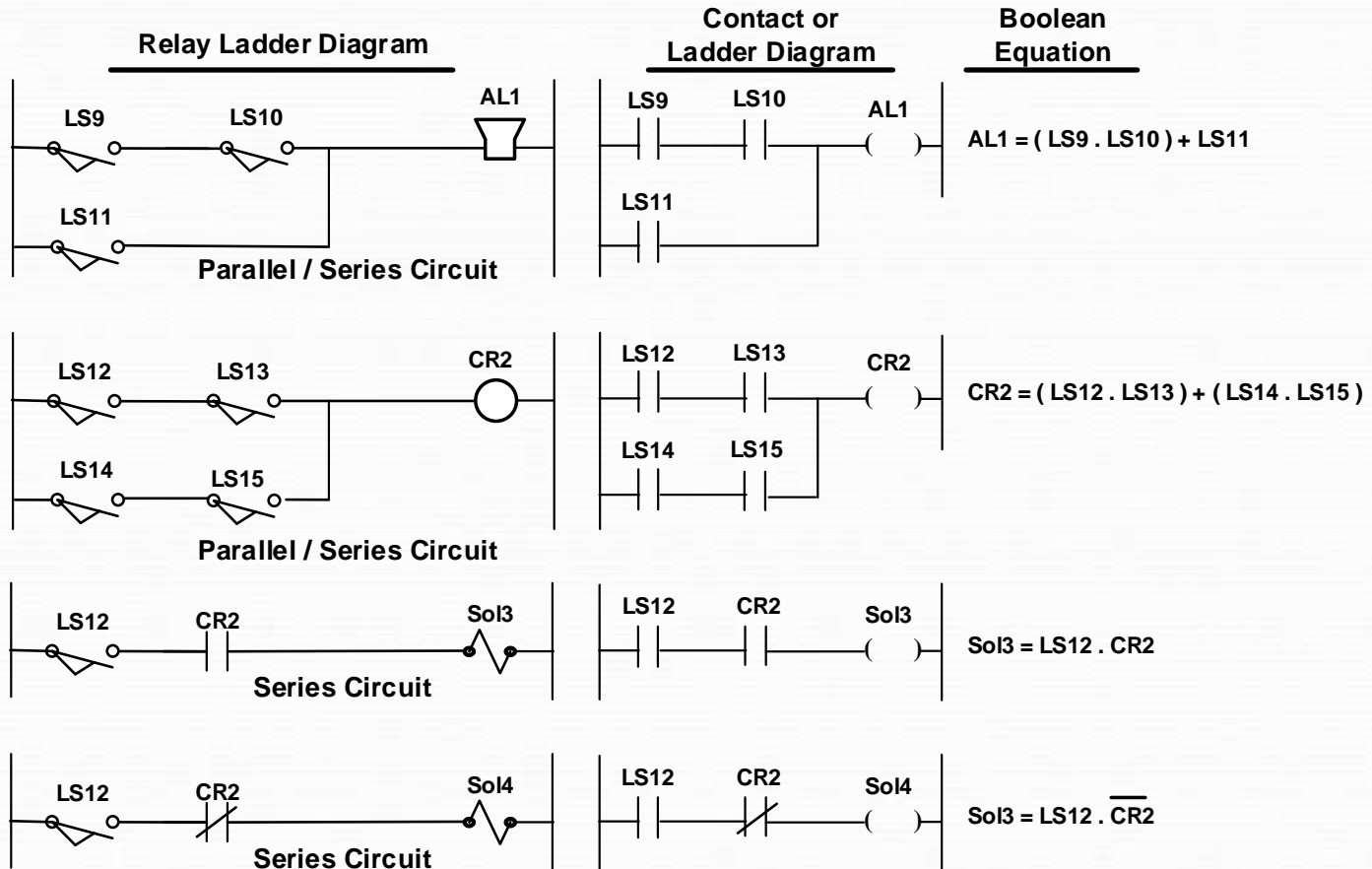
## **PLC Logic & Hardwired Relay Logic –II cont.**

Contact symbology is a very simple way of expressing control logic in terms of symbols that are used on relay control schematics. If the controller language is ladder diagram, the translation from existing relay logic to programmed logic is a one-step translation to contact symbology.

# Translation of Hardwired Relay Logic into PLC Logic

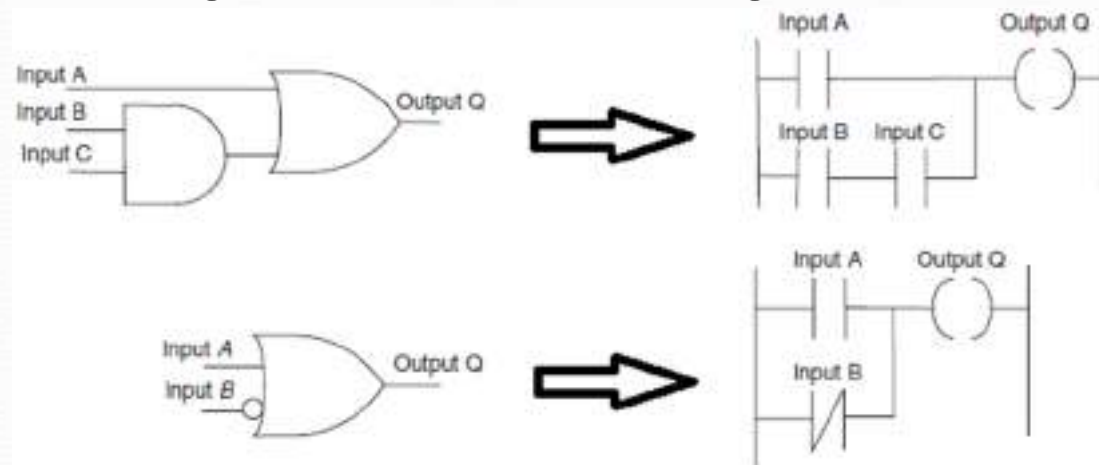


# Translation of Hardwired Relay Logic into PLC Logic



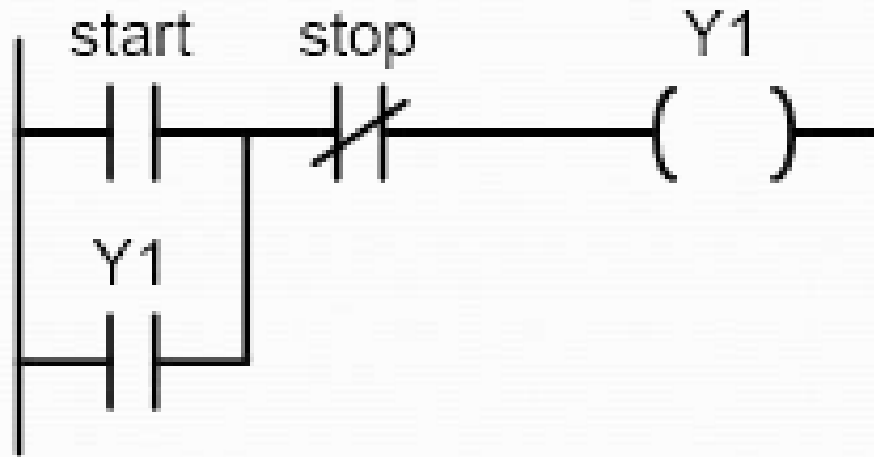
# Logic Design

Design ideas can be converted to Boolean equations directly. The Boolean equation form **can then be simplified or rearranged**, and then converted into logic gates circuit or ladder logic. For example, the following Boolean equation ( $A + B.C = Q$ ) could be directly converted to its equivalent logic gates circuit and ladder logic as shown in Figure



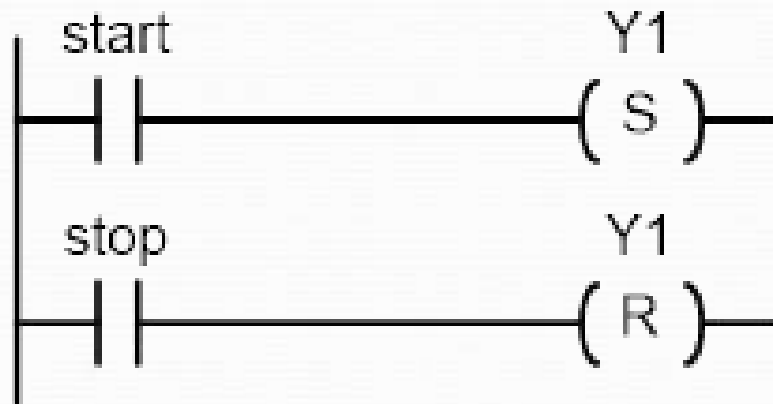
# Common Ladder Programming Techniques

## 1) Start, Stop & Latched



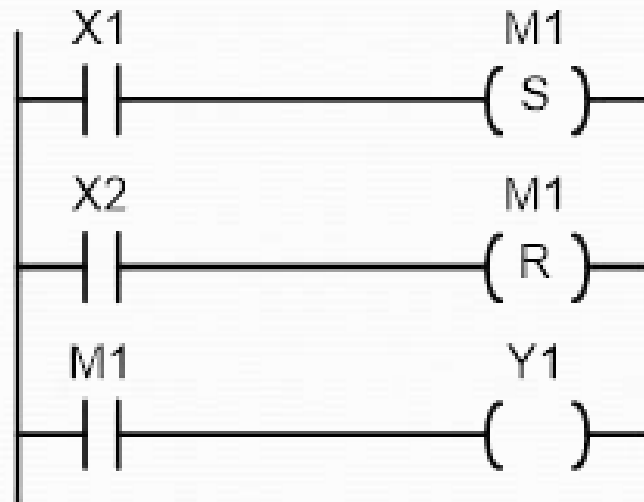
## Common Ladder Programming Techniques cont.

2) Latched circuit by using set & rest instructions



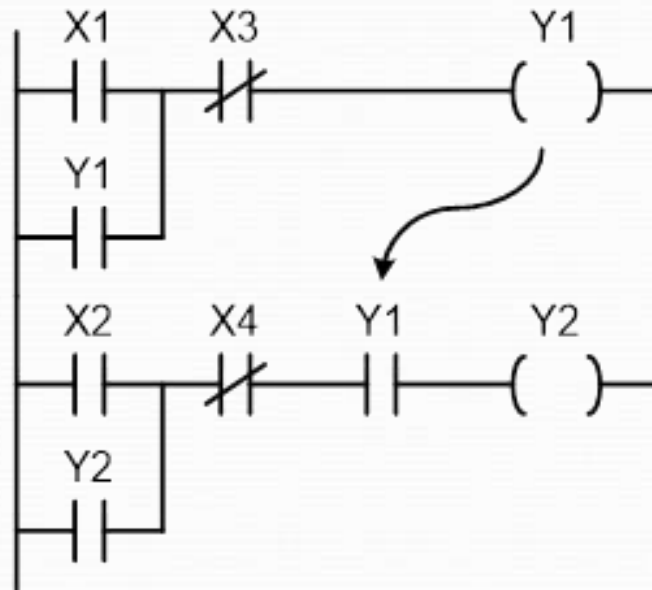
## Common Ladder Programming Techniques **cont.**

### 3) Power Shutdown Latched



## Common Ladder Programming Techniques cont.

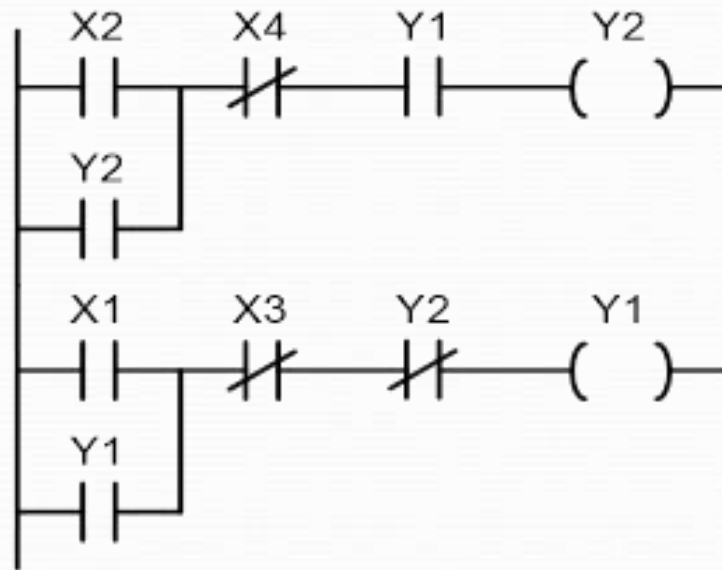
### 4) Conditional Control





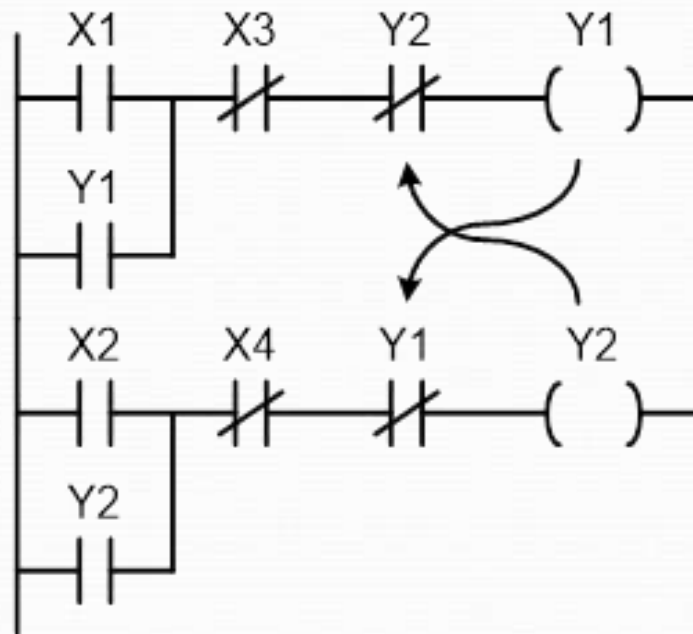
## Common Ladder Programming Techniques **cont.**

### 5) Sequential Control



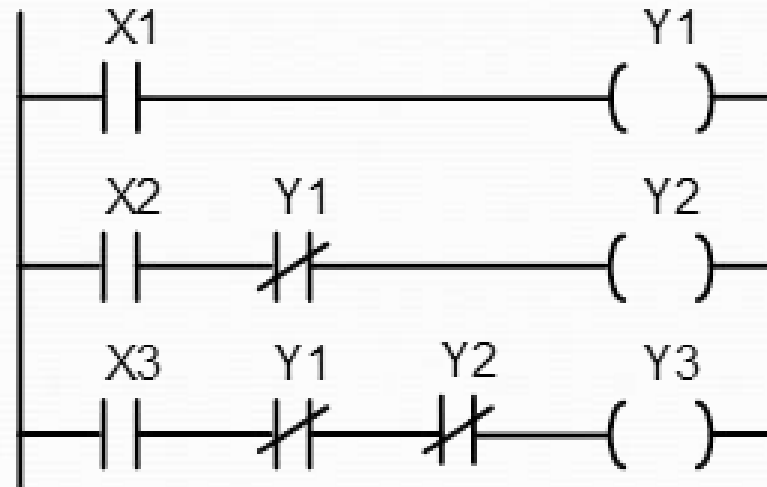
## Common Ladder Programming Techniques cont.

### 6) Interlock Control



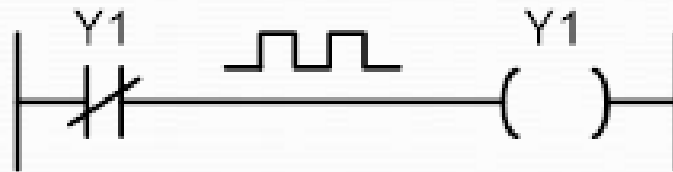
# Common Ladder Programming Techniques **cont.**

## 7) Priority Conditions

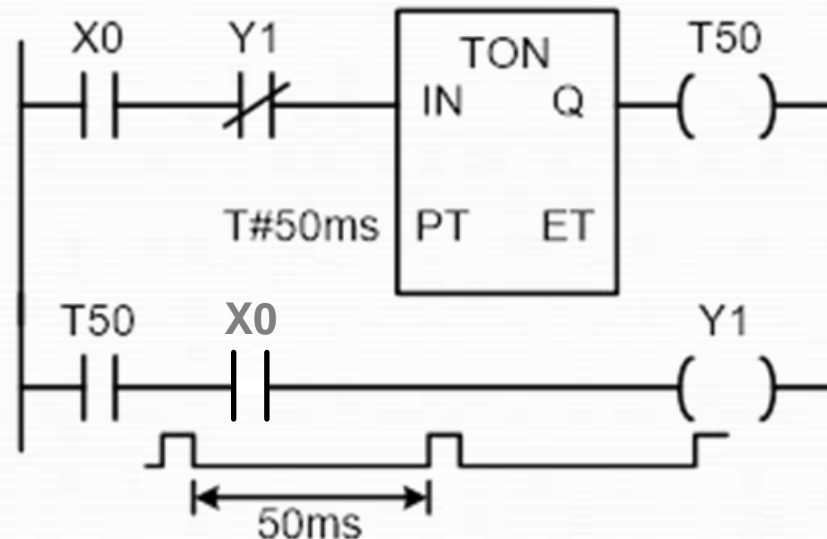


## Common Ladder Programming Techniques cont.

### 8) Oscillating Circuit

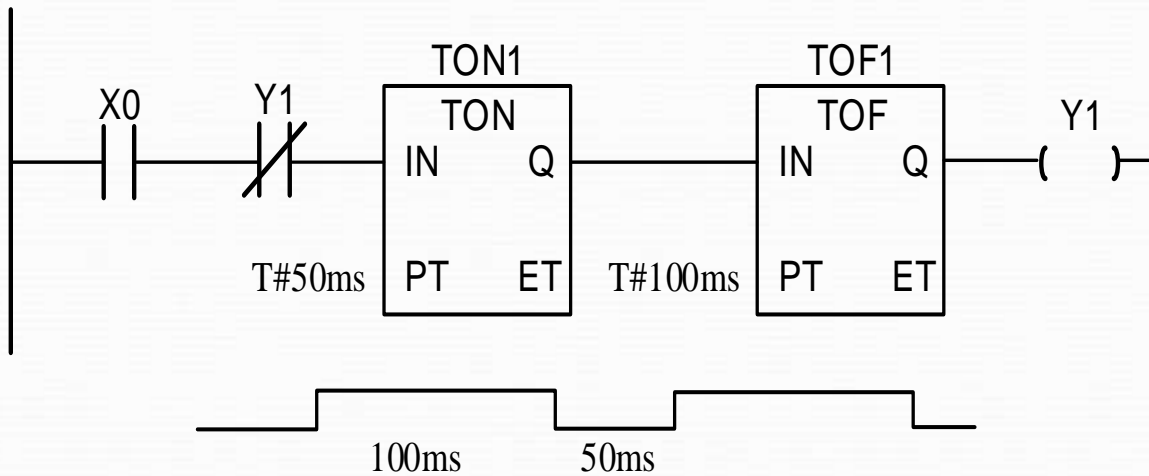


### 9) Oscillating Circuit with Controlled off Period



# Common Ladder Programming Techniques cont.

## 10) Flashing Circuit



# Timers

Many control tasks require the programming of time. For example, the refrigerator protection device.

## Timers in PLC

The timers of a PLC are realized in the form of software modules and are based on the generation of digital timing.

The counted clock pulses are derived from the quartz generator of the microprocessor. The desired time duration is set in the control program.

There are many timer function blocks are used but the most used timers which were defined by IEC 61131-3 are:

- TP Pulse timer
- TON On-delay timer
- TOF Off-delay timer

Time duration is specified by means of a defined character format, **T# or t#**, followed by a number and the time elements, i.e. days, hours, minutes, seconds, milliseconds.

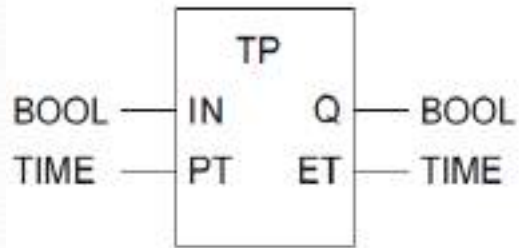
d	day
h	hour
m	minute
s	second
ms	millisecond

Examples

T#2h30m
T#3m15s
t#20s400ms
T# 50ms
t#2h_20m_30s



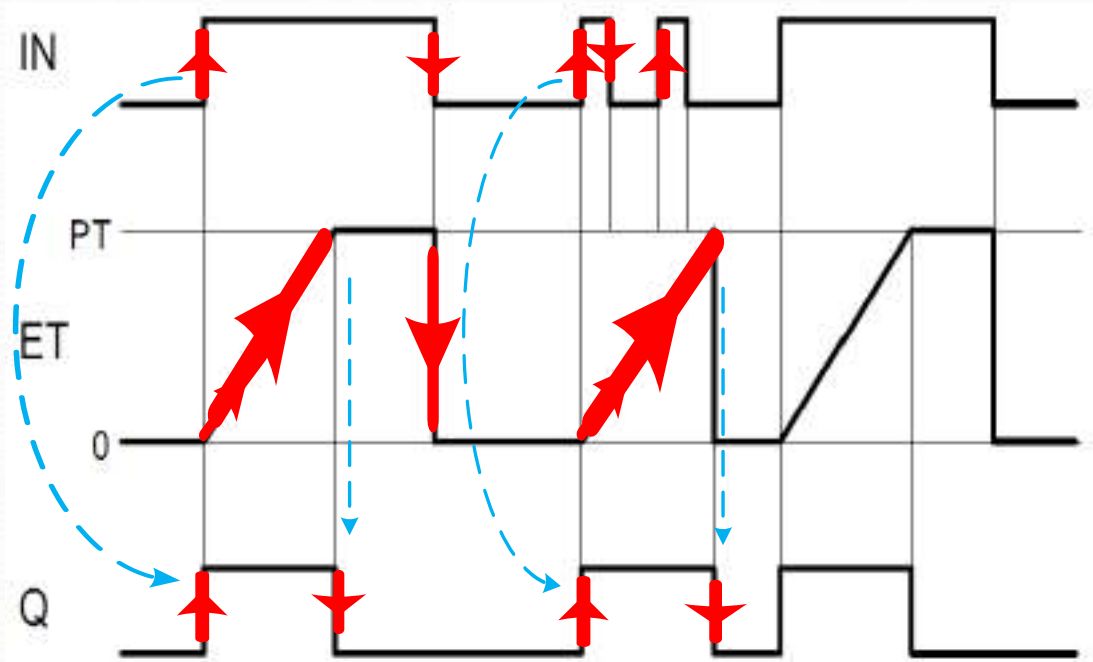
## a) Pulse Timer (TP)



TP Function Block

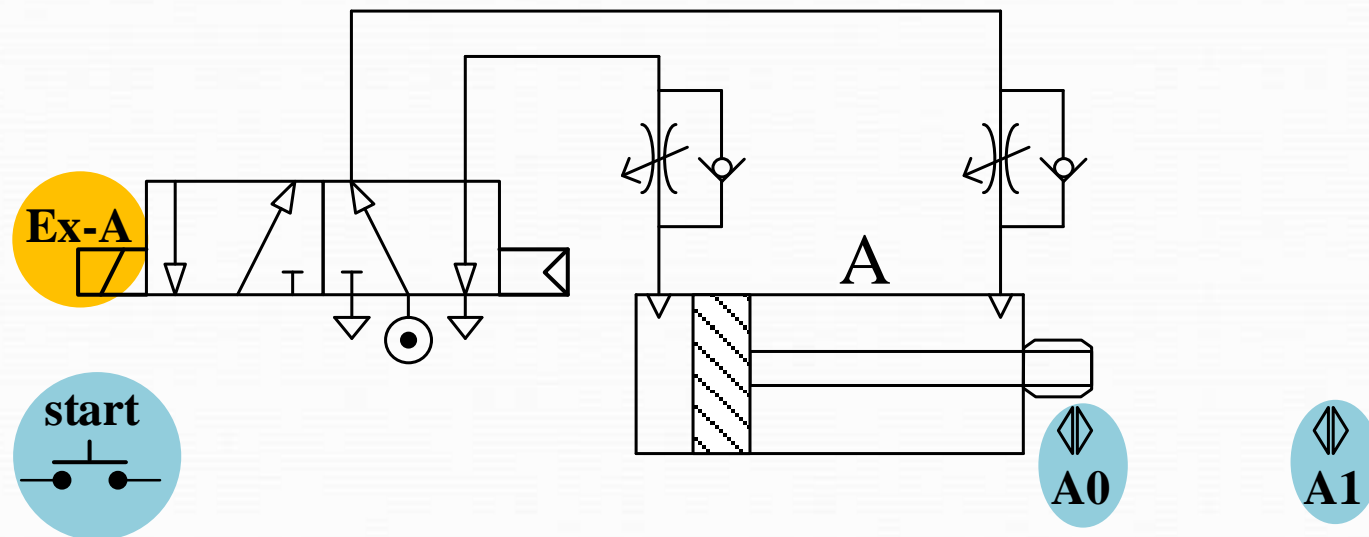
PT : Preset Time

ET : Elapsed Time

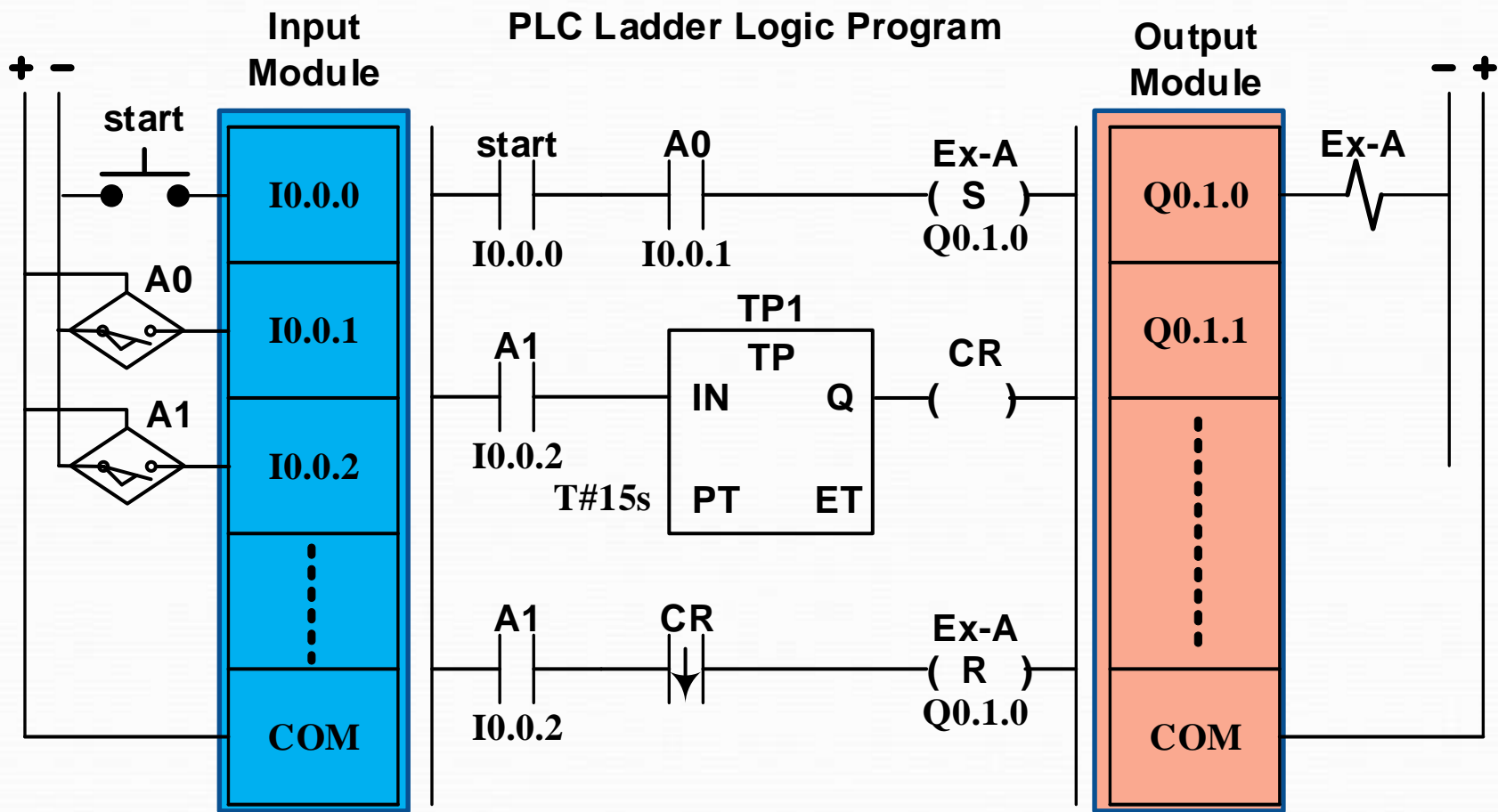


TP Timing Diagram

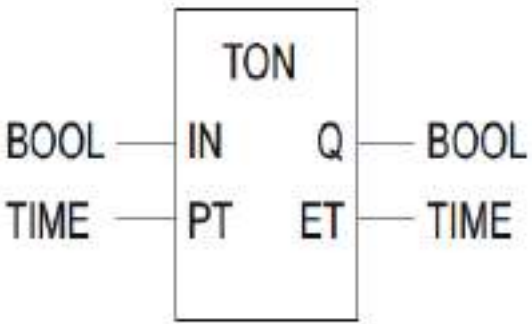
## Example (Electro-pneumatic clamping mechanism)



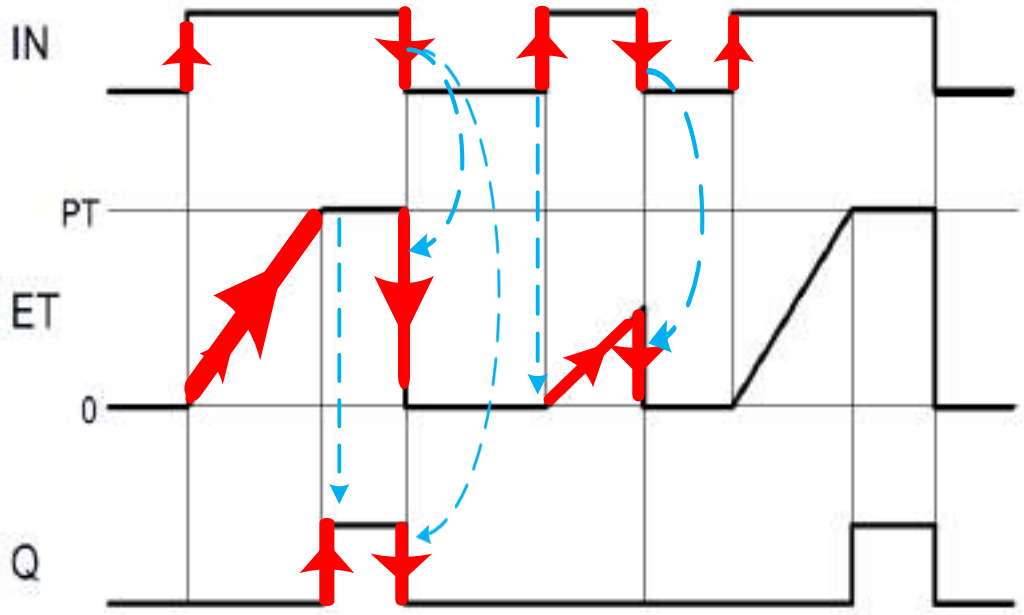
Pressing the start button causes the piston of cylinder A to advance. When the piston has advanced fully, **it is to remain in this position for 15 seconds.** The cylinder then returns to its **initial position.**



# b) On Delay Timer(TON)

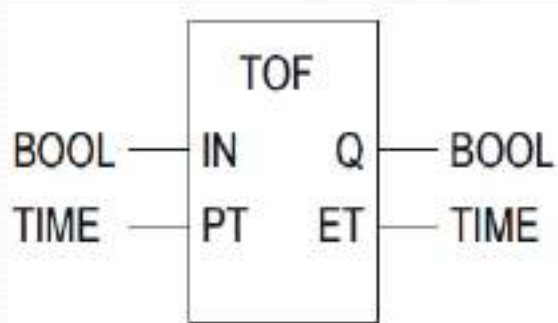


TON Function Block

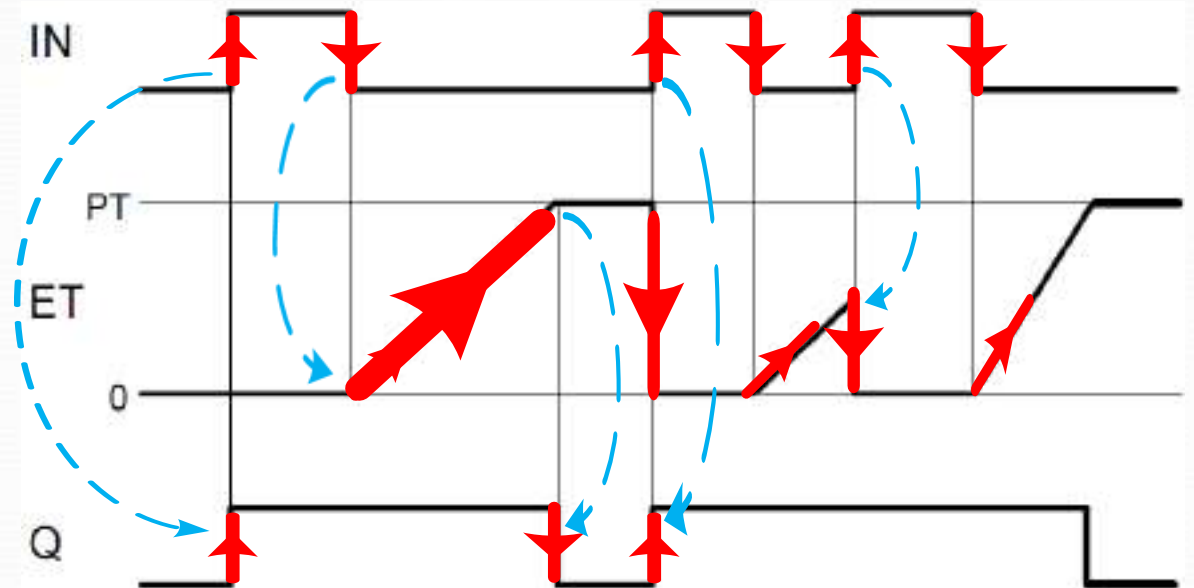


TON Timing Diagram

## c) Off Delay Timer(TOF)



TOF Function Block



TOF Timing Diagram

**Example** By using modular PLC type LG write the PLC LLD that generates a square wave of **2Hz** and **60% duty cycle**. Use **NO Start Button** and **NC Stop Button** and a **LED** as indicator.

**Solution:**

$$T = \frac{1}{f} = \frac{1}{2} = 0.5s = 500ms$$

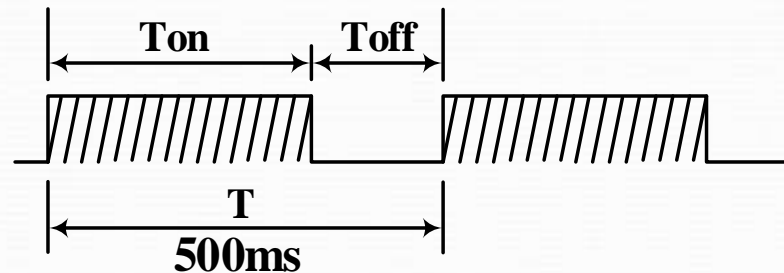
$$\text{Duty Cycle} = \frac{T_{on}}{T}$$

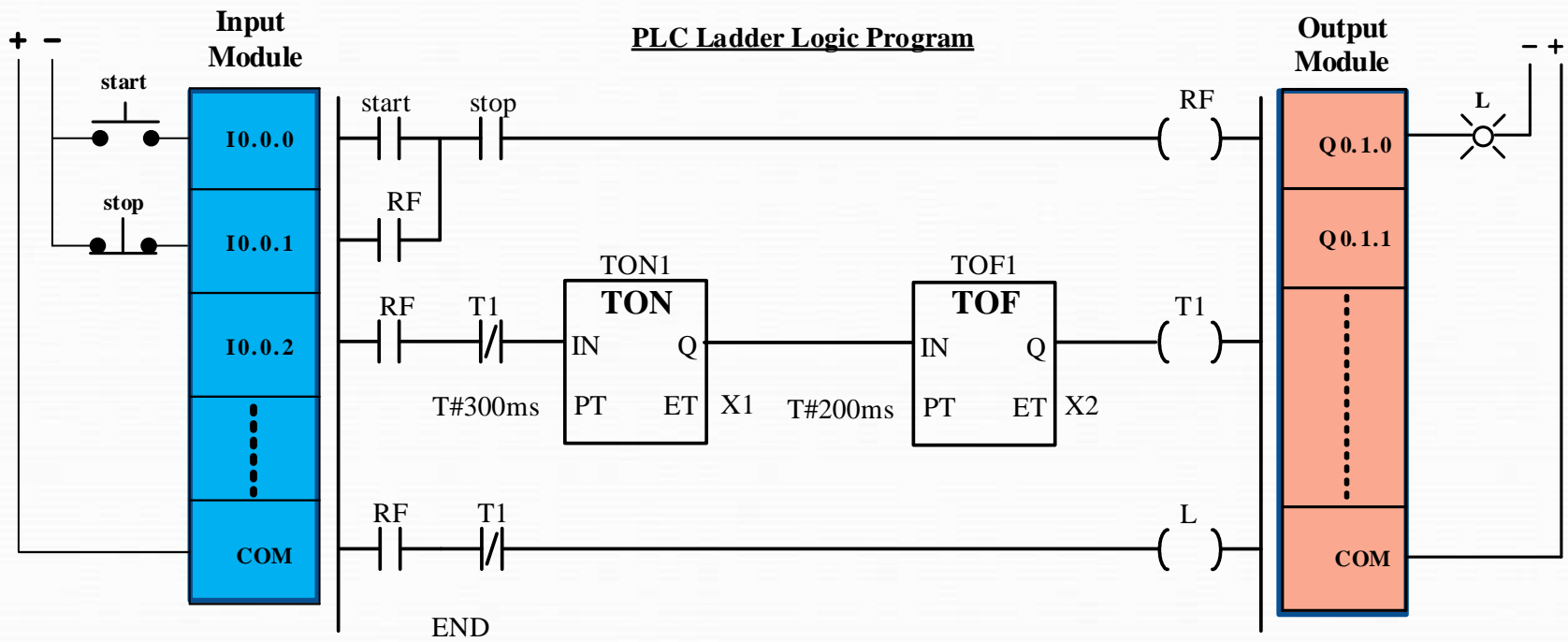
$$T_{on} = T * \text{Duty Cycle}$$

$$= 500 * 0.60 = 300ms$$

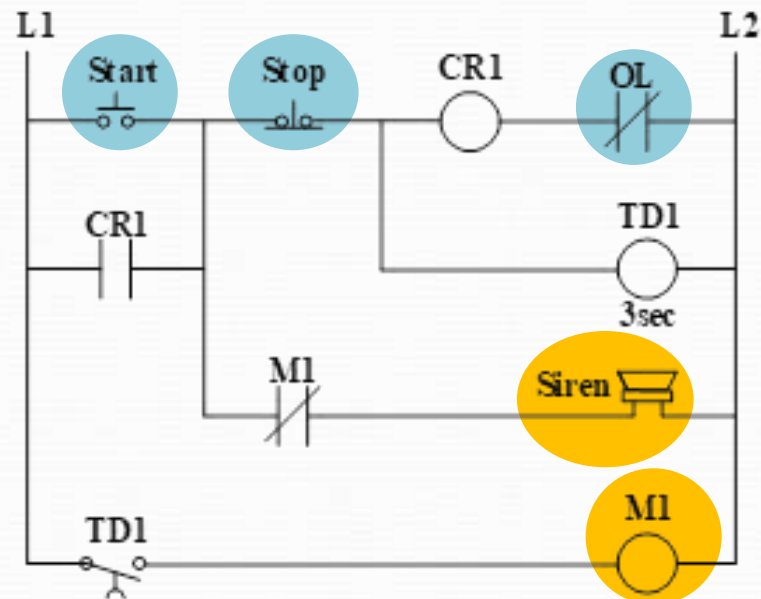
$$T_{off} = T - T_{on}$$

$$= 500 - 300 = 200ms$$

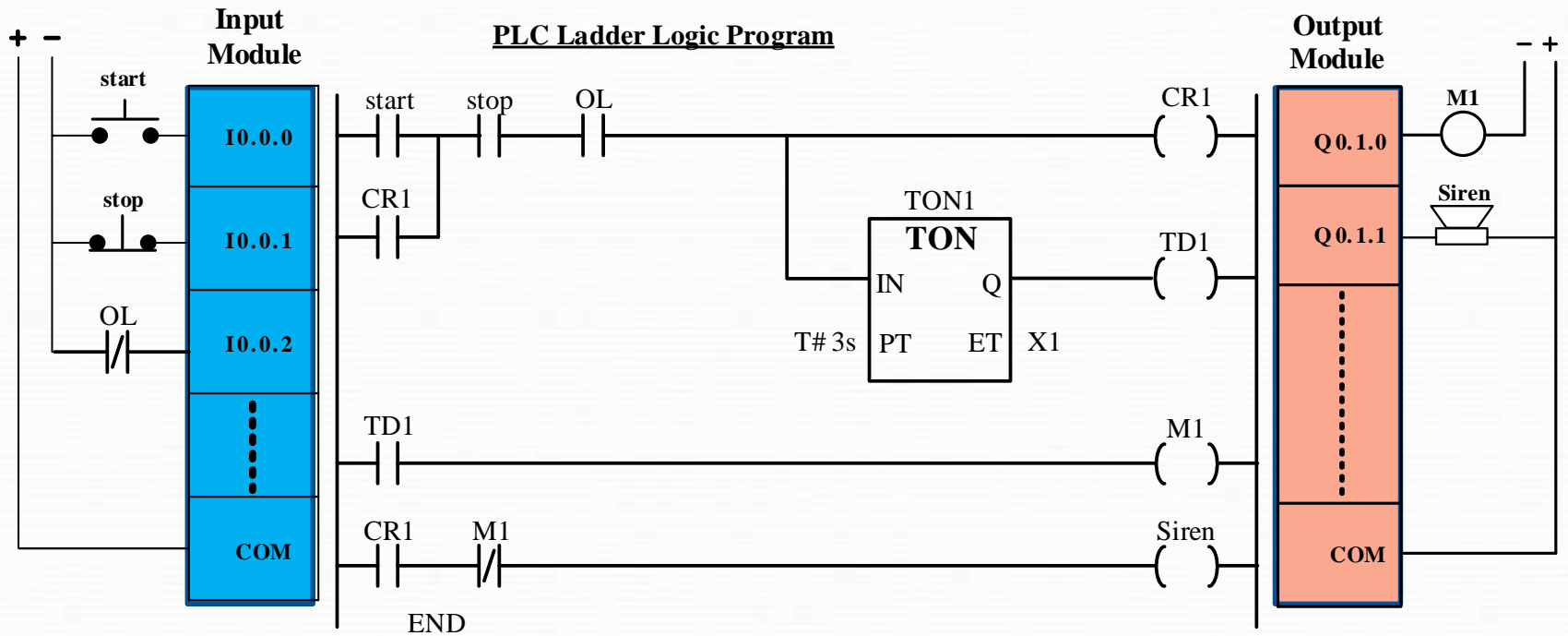




**Example**: An **electric motor** is used to power a large **conveyor belt**. Before the motor actually starts, a **warning siren** activates **to alert workers** of the conveyor's **forthcoming** action. The following relay circuit is supposed to accomplish both tasks (motor control plus siren alert). Convert this circuit to a PLC based control system.

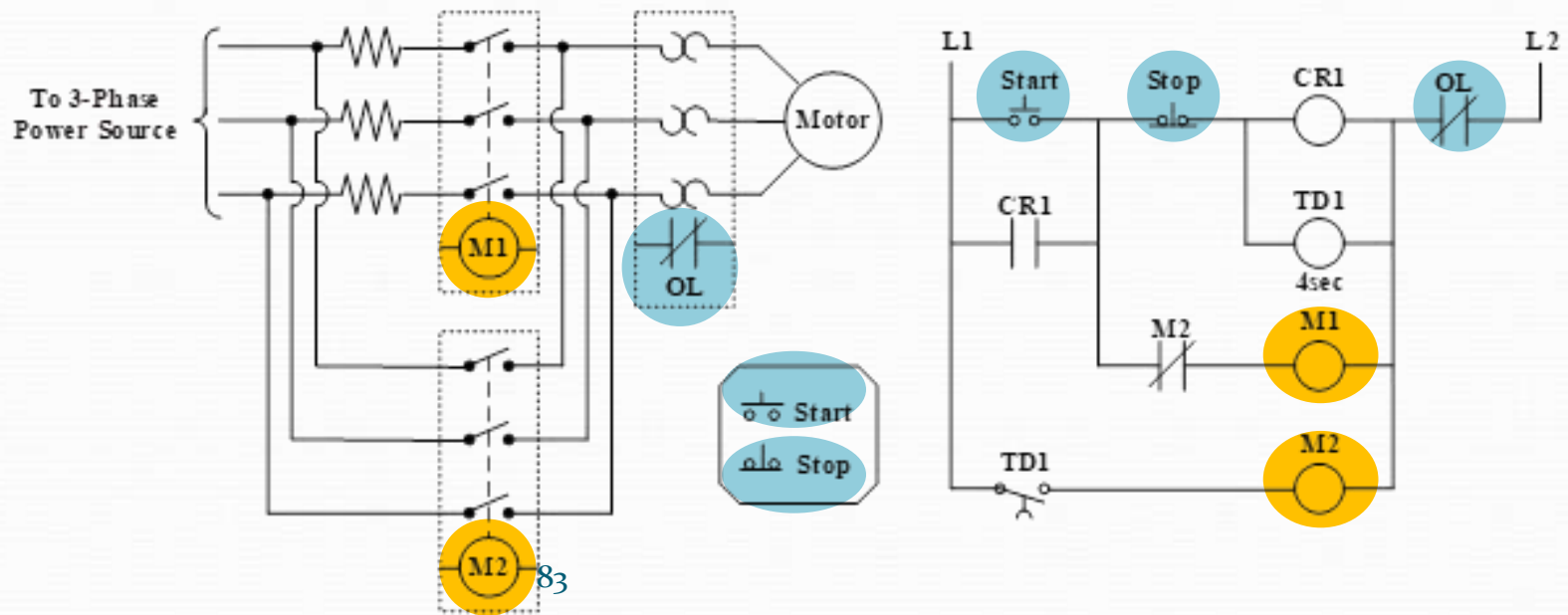




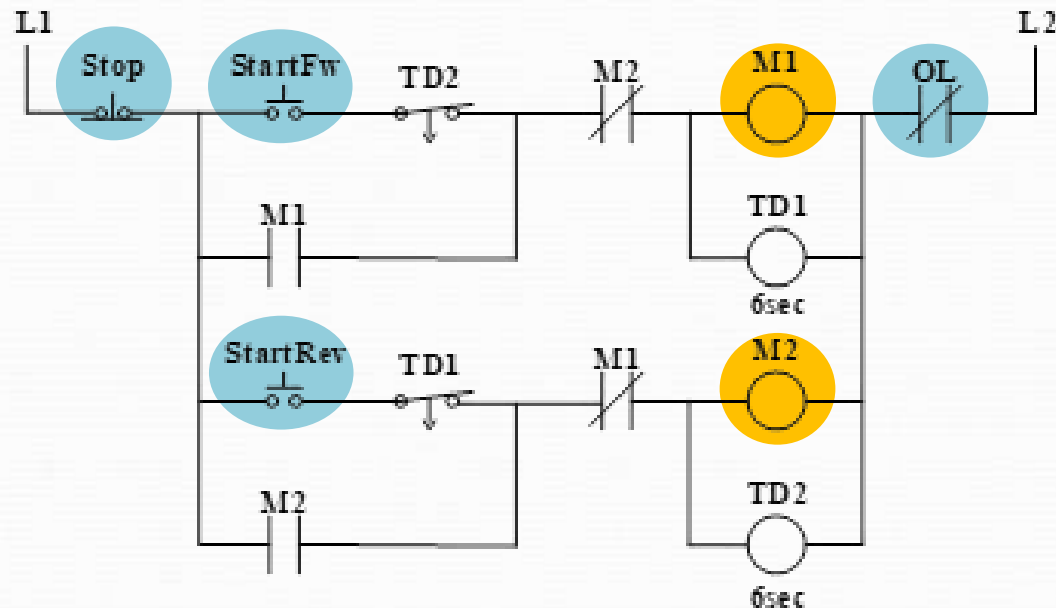


Q1: By using modular PLC type LG write the PLC LLD that generates a square wave of 4Hz and 70% duty cycle. Use NO Start Button and NC Stop Button and a Led as indicator.

Q2) Large electric motors are often equipped with some form of **soft-start control**, which **applies power gradually** instead of all at once (as in “across the line” starting). Here is an example of a simple “**soft start**” control system. Convert this circuit into a PLC based control system.



Q3) The following figure shows the **reversing motor** control circuit. Convert this circuit into a PLC based control system.



## 2) Counters

Counters are used to detect piece numbers and events.

A counter is for instance required, if exactly 10 identical parts are to be conveyed to a conveyor belt via a sorting device.

EN 61131-3 (IEC 61131-3) differentiates between three different counter modules:

**CTU** : Up Counter

**CTD** : Down Counter

**CTUD** : Up / Down Counter

These standard function modules are used to detect standard, non time-critical counting.

With many control tasks it is however necessary to use so-called **high speed counters**.

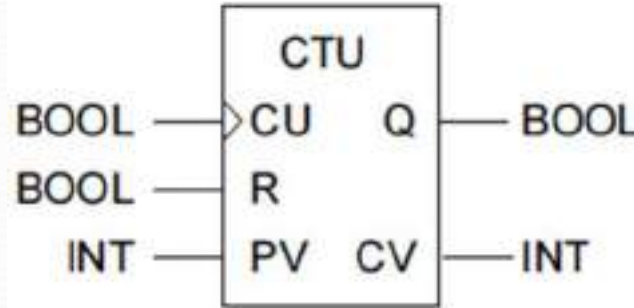
**"High-speed"** in this case generally refers to a counter frequency in excess of 50 Hz, i. e. more than 50 events are counted per second.

The limitations of counter frequency in counter function blocks are due to the output signal delays.

Each input signal (i.e. also the counter signal) is delayed by a certain time, before it is released for processing in the PLC. This prevents interference.

A further limitation is the cycle time of the PLC.

## a) Up Counter



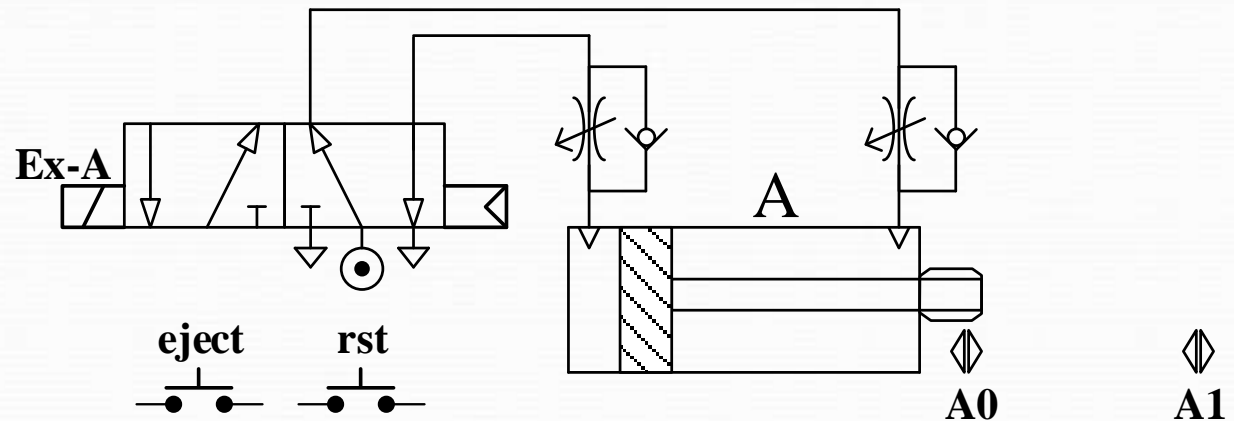
The Up Counter is **known as a CTU (count up)**. The counter is set at the initial value 0 by a signal at reset input R.

The output **current value (CV)** increased by 1 with each **positive edge** at counter input **CU (count up)**.

As soon as the current value **CV is equal to or greater than the preset value (PV)**, the output signal assumes the value “1”.

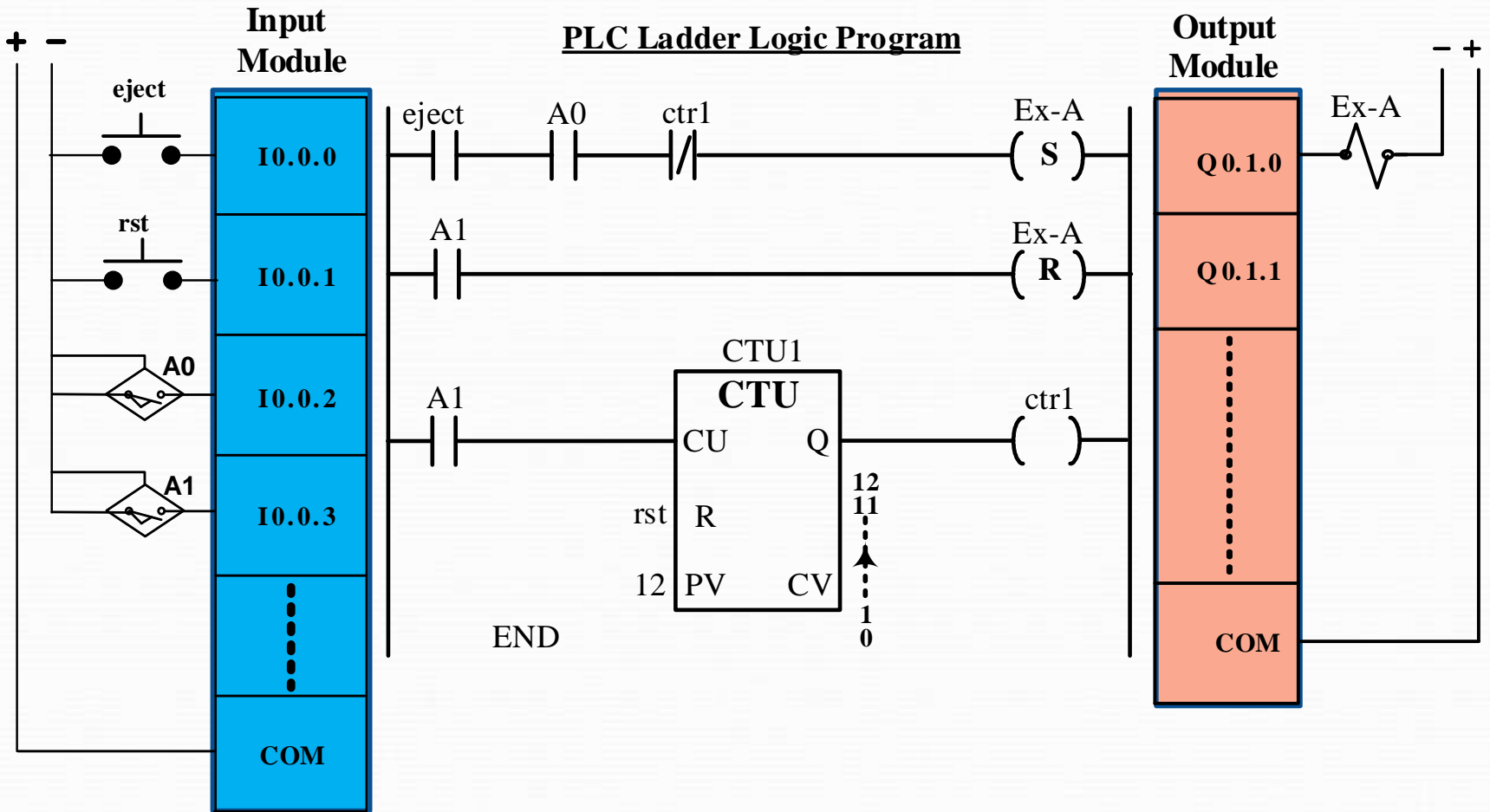
Prior to reaching this value, output Q has a “0” signal.

**Example:** Ejecting parts from a gravity-feed magazine via a cylinder.

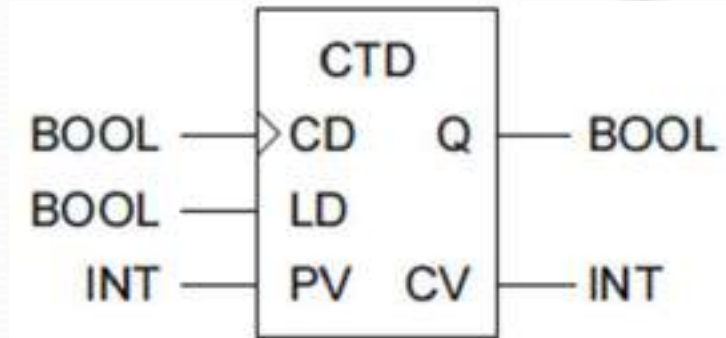


IF push button “eject” is actuated, cylinder A is to advance, ejecting a workpiece and then retract again. **12 parts** are to be ejected in this way. When **12 parts** have been ejected, it should no longer be possible to trigger a cylinder movement via push button “eject”. First the counter must be reset by actuating push button “rst”.





## b) Down Counter



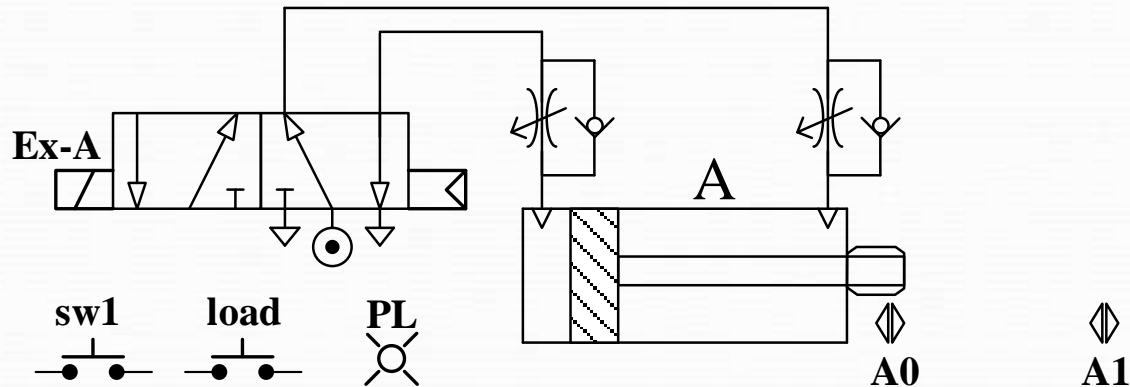
The Down Counter is known as a CTD (count down) and represents the counterpart of the up counter.

The down counter with preset value (PV) is loaded with a “1” signal at input load (LD).

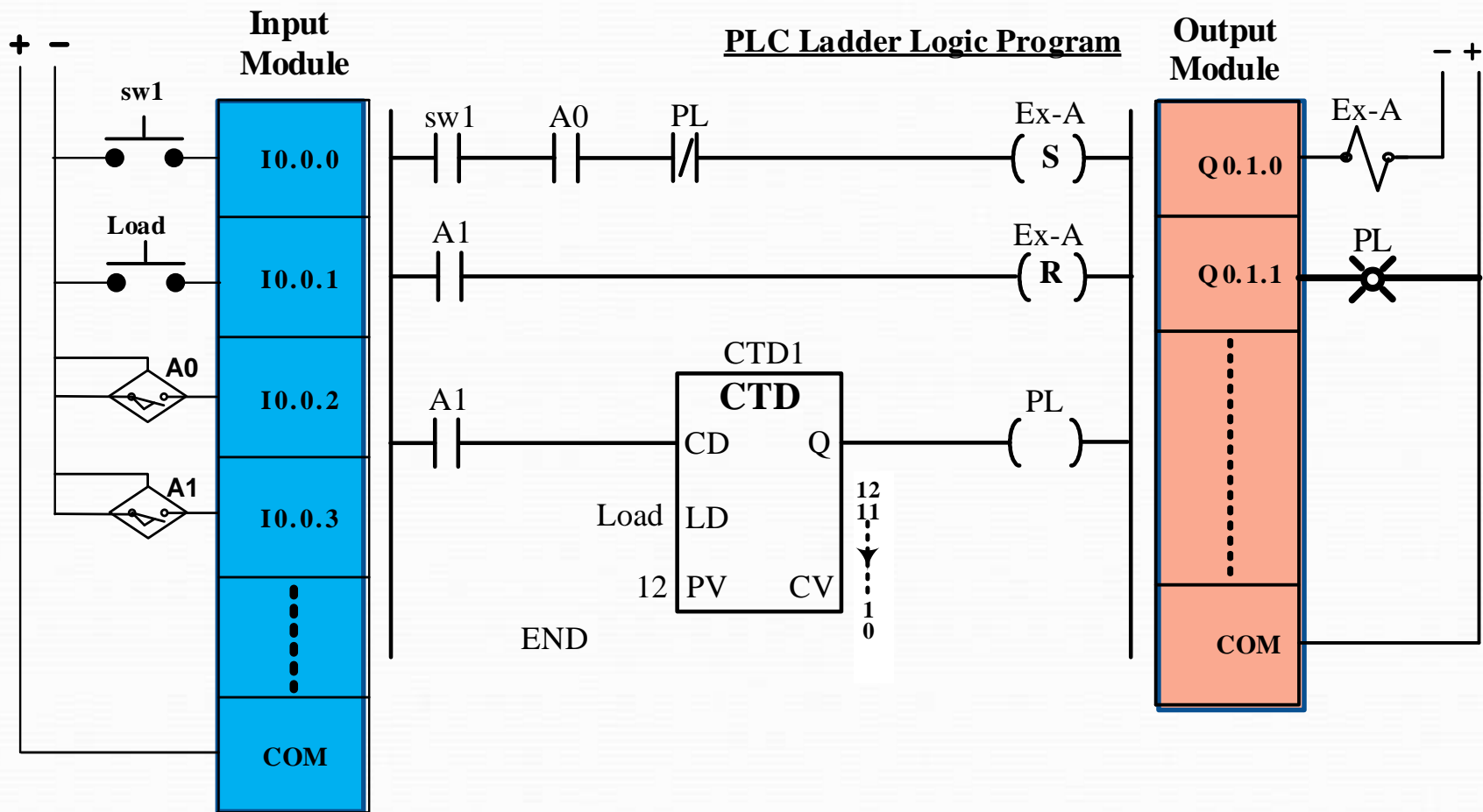
During normal operation, each positive edge at input CD (count down) reduces the counter reading.

Output Q of function block CTD is “0”, until the current counter reading CV becomes less than or equal to “0”.

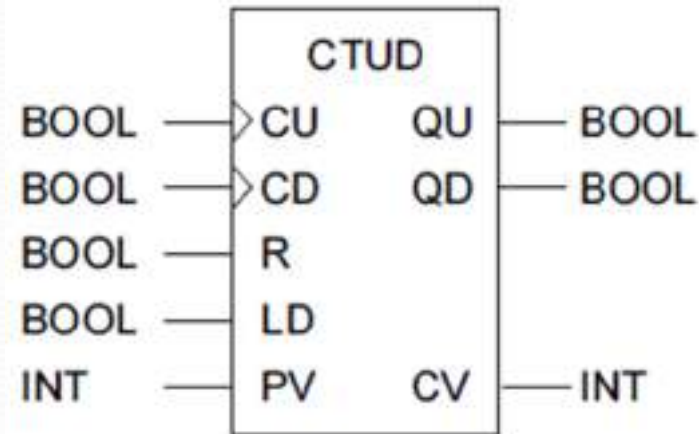
## Example: (Electro-pneumatic mechanism)



cylinder A is moved via a valve Ex-A. The position of the cylinder is signaled via the sensors A0 (retracted) and A1 (extended). The cylinder is to **advance, if push button sw1 is pressed. When 12 strokes have been executed, pilot lamp PL is illuminated** and the counter has expired. The counter must be reloaded with the preset value, before any cylinder movements can be executed further. This is effected by means of actuating the push button “load”.



## c) Up/Down Counter



The Up/Down Counter is known as a CTUD. Function block CTUD, Up/Down counter, combines an Up and a Down counters.

The value of output QU is calculated in accordance with the equation:  $CV \geq PV$ , the value of output QD in accordance with the equation  $CV \leq 0$ .

It should be noted that the function of the down counter is used only after the preset value has been loaded to the counter via the command LD.

## 2) Counters

Counters are used to detect piece numbers and events. For instance, The counter is required, if exactly 10 identical parts are to be conveyed to a conveyor belt via a sorting device.

EN 61131-3 (IEC 61131-3) differentiates between three different counter modules:

**CTU** : Up Counter

**CTD** : Down Counter

**CTUD** : Up / Down Counter

With many control tasks it is however necessary to use so-called **high speed counters**.

"**High-speed**" in this case generally refers to a counter frequency in **excess of 50 Hz**, i.e. more than 50 events are counted per second.

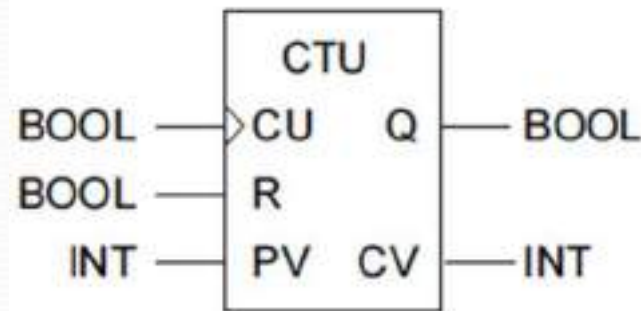
**The limitations** of counter frequency in counter function blocks are **due to the input and output signals delays**, as well as **the cycle time of the PLC**.

## a) Up Counter

**CU : Count Up**

**R : Reset**

**PV : Preset Value**



**Q : Output**

**CV : Current Value**

The counter is set at the initial value 0 by a signal at reset input R.

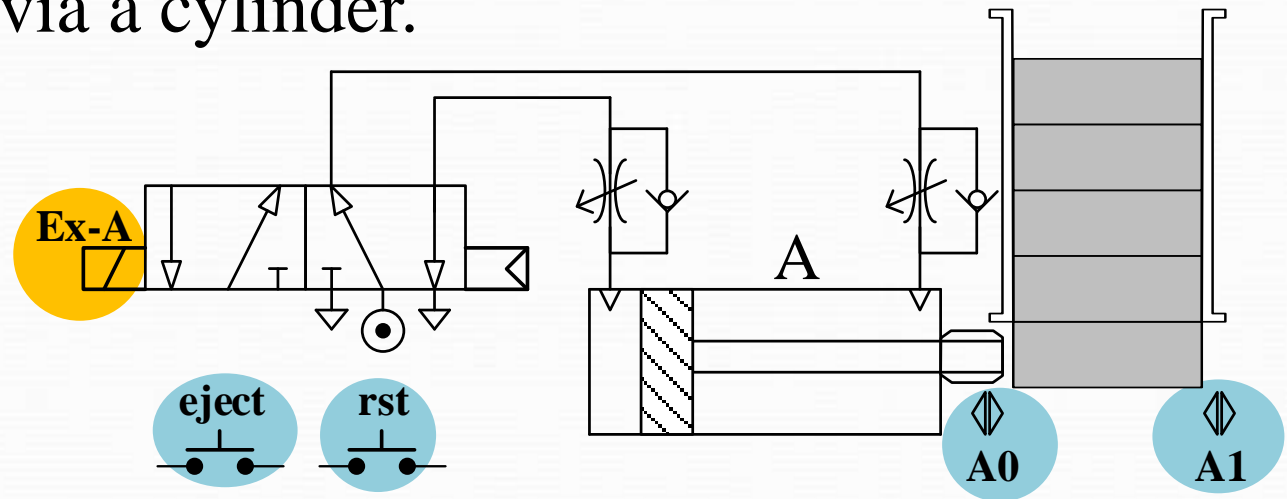
The **current value (CV)** increased by 1 with each **positive edge** at counter input **CU (count up)**.

As soon as the current value **CV is equal to or greater than the preset value (PV)**, the output signal assumes the value “1”.

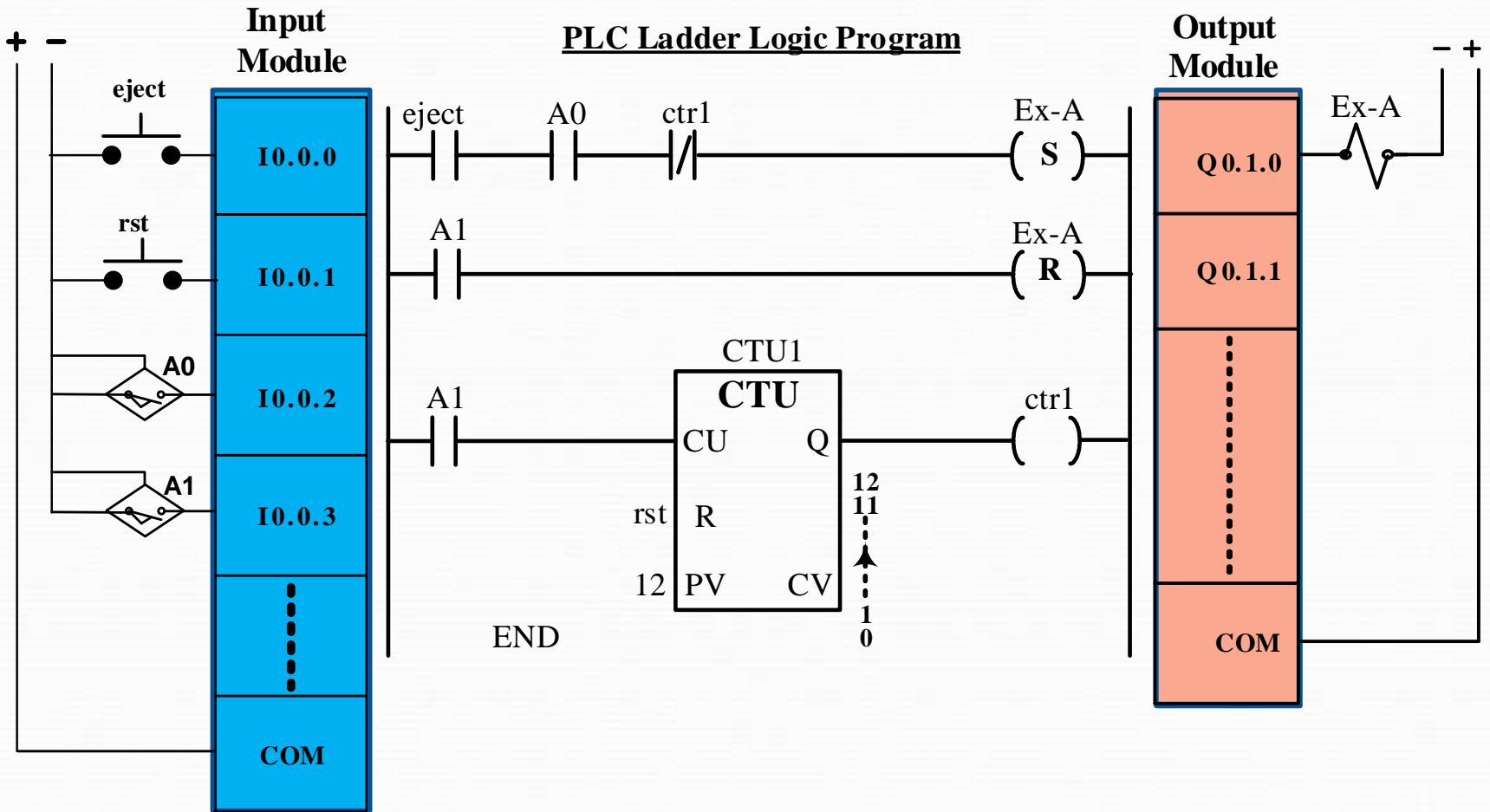
Prior to reaching this value, output Q has a “0” signal.



**Example:** Ejecting parts from a gravity-feed magazine via a cylinder.



IF push button “eject” is actuated, cylinder A is to advance, ejecting a workpiece and then retract again. 12 parts are to be ejected in this way. When 12 parts have been ejected, it should no longer be possible to trigger a cylinder movement via push button “eject”. First the counter must be reset by actuating push button “rst”.

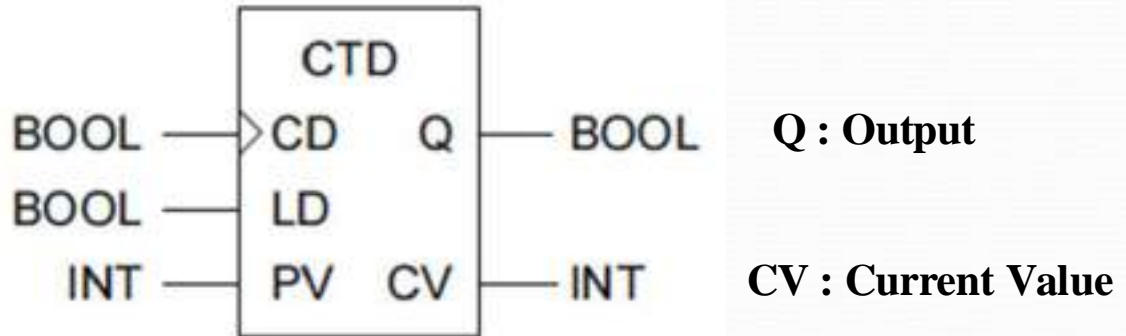


## b) Down Counter

**CD : Count Down**

**LD : Load**

**PV : Preset Value**



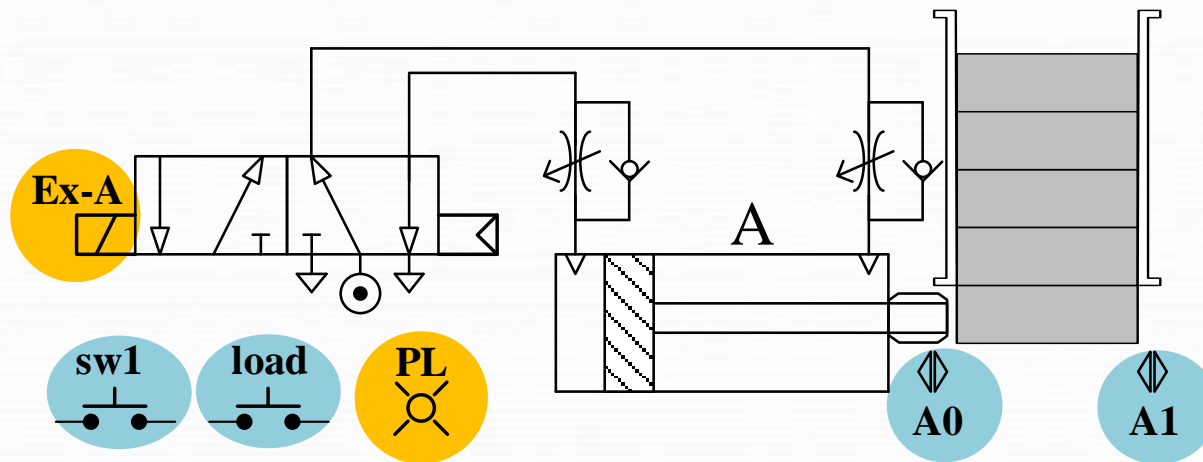
The Down Counter **is the counterpart** of the up counter.

The down counter with preset value (PV) is loaded with a “1” signal at input **load (LD)**.

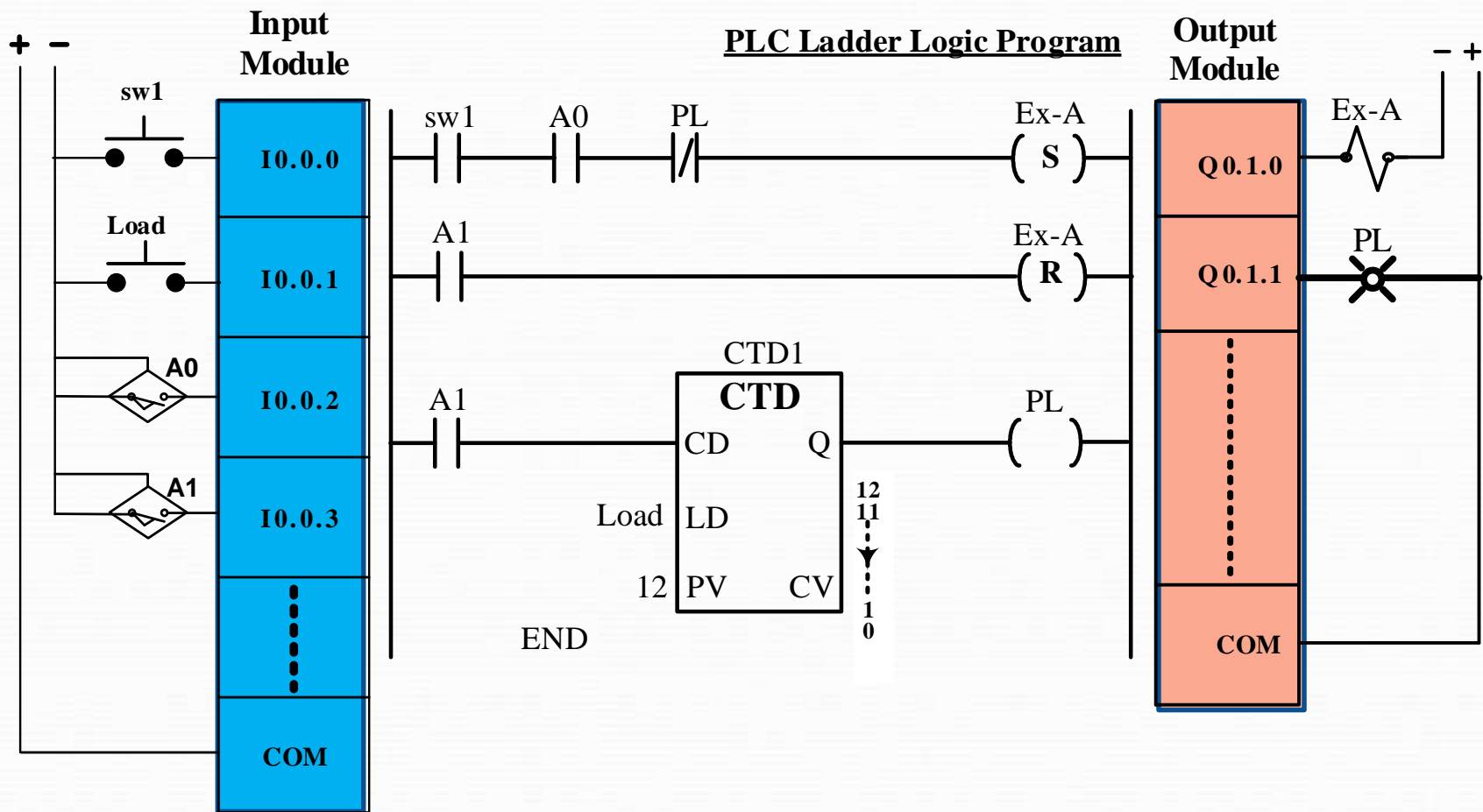
During normal operation, each positive edge at input CD (count down) reduces the counter reading.

**Output Q of function block CTD is “0”, until the current counter reading CV becomes less than or equal to “0”.**

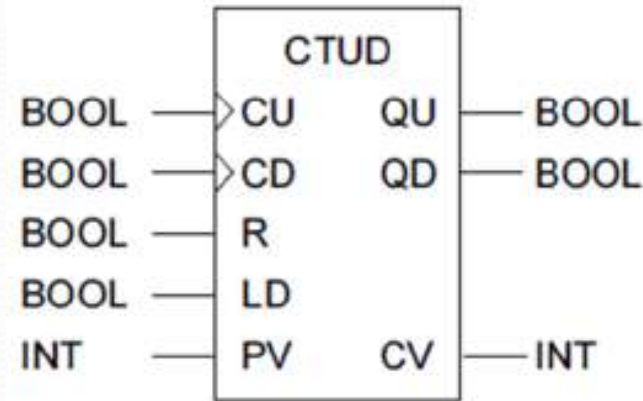
**Example:** Ejecting parts from a gravity-feed magazine via a cylinder.



The cylinder is to **advance**, if push button **sw1** is pressed. **When 12 strokes** have been executed, **pilot lamp PL** is illuminated and the **counter has expired**. The counter must be **reloaded** with the preset value, before any cylinder movements can be executed further. This is effected by means of actuating the push button “load”.



## c) Up/Down Counter

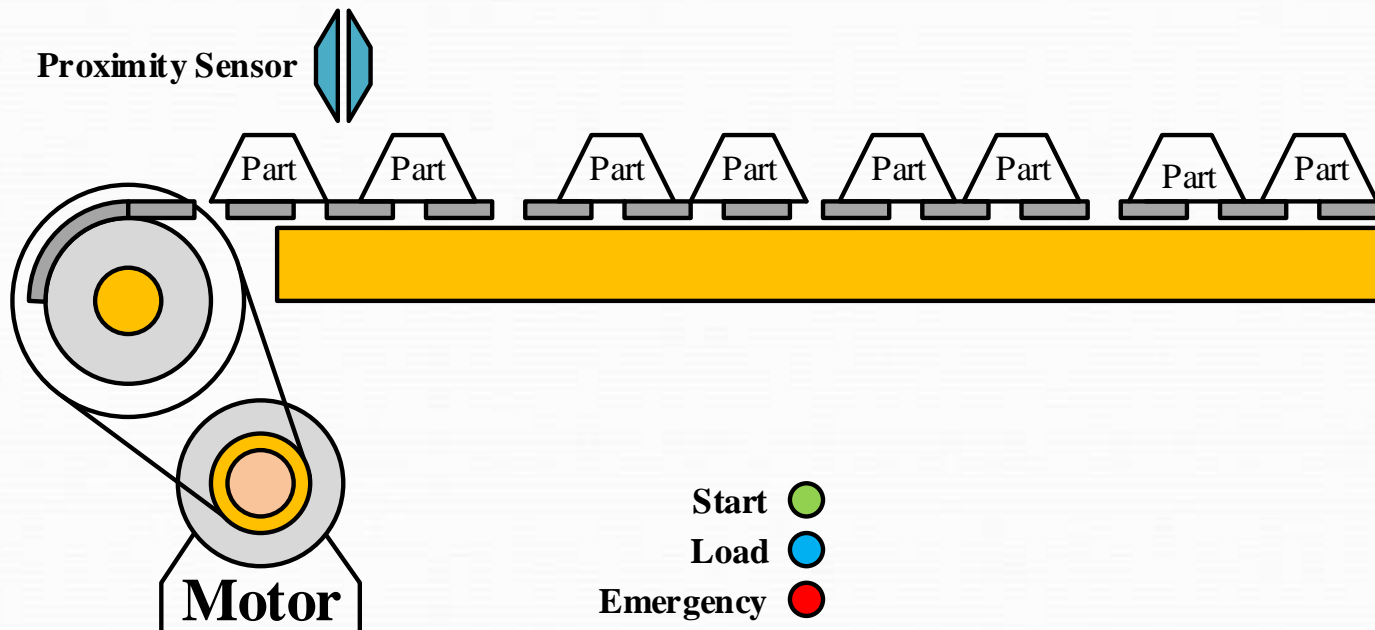


The function block CTUD, Up/Down counter, combines an Up and a Down counters.

The value of output QU is calculated in accordance with the equation:  $CV \geq PV$ , the value of output QD in accordance with the equation  $CV \leq 0$ .

It should be noted that the function of the down counter is used only after the preset value has been loaded to the counter via the command LD.

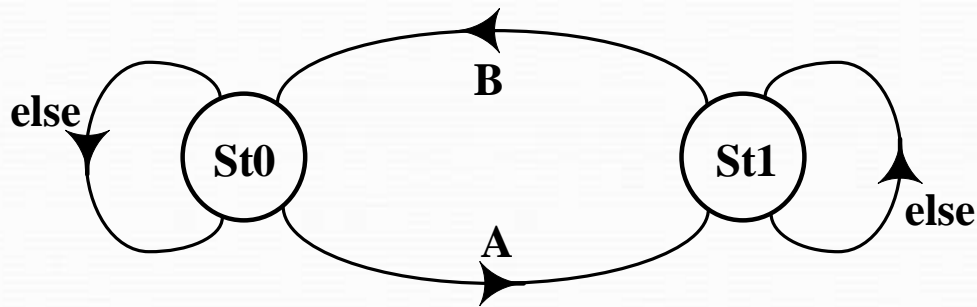
**Q1:** It is required to supply 24 parts at each time the start button is pressed. An emergency button is used to stop the process if an accident occurs. By using modular PLC / Lsis GLOFA with GM4 CPU devise the PLC LLP that used to control this process using down counter and Load button.



# State-Based Design

The state based system consists of system states, and the transitions between those states.

**State : St**

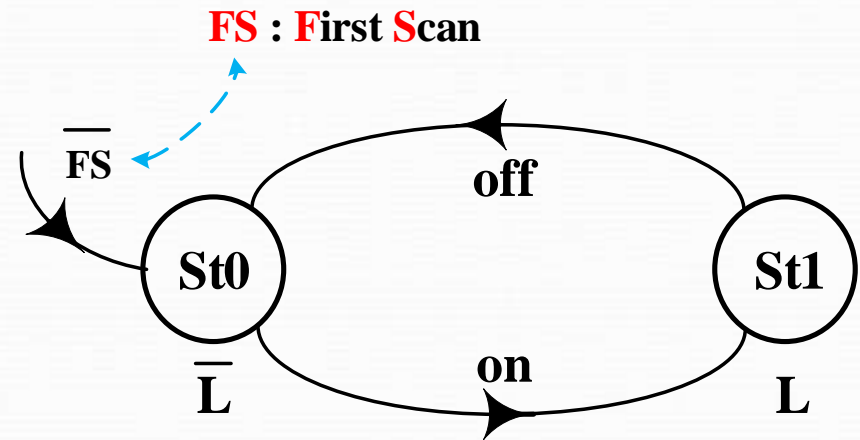


This state diagram has two states, State (0) and State (1). If the system is in State (0) and event (A) occurs, the system will then go into State (1), otherwise it will remain in State (0). Likewise if the system is in State (1) and event (B) occurs, the system will return to State (0).



Inputs  
on - NO  
off - NO

Outputs  
Light (L)



Outputs

States

L

St0

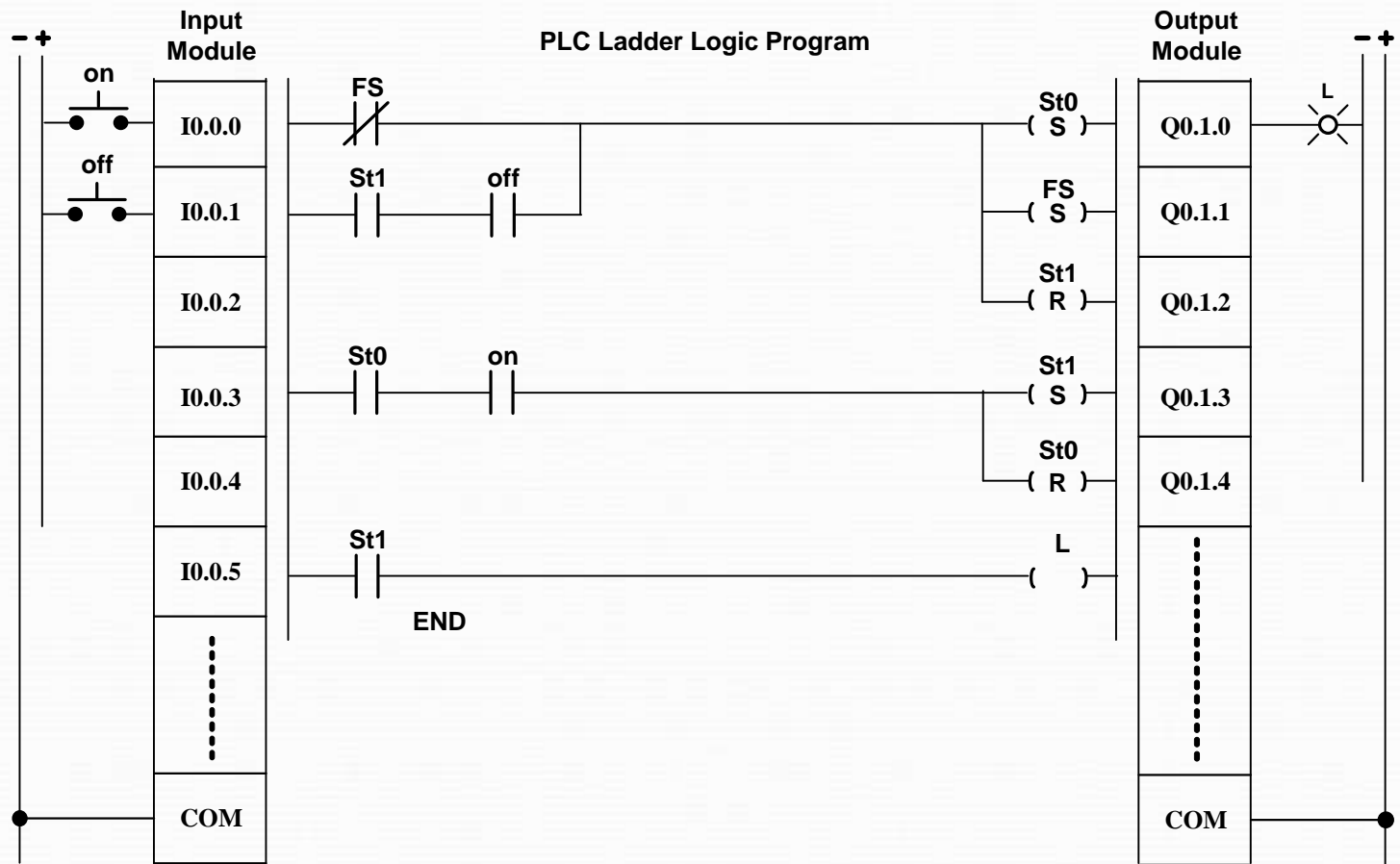
0

St1

1

Outputs Equations

$$L = St1$$



How the PLC program will be if one push button is used instead of two buttons for the same example

Inputs

Push Button (PB) - NO

Outputs

Light (L)

Outputs

States

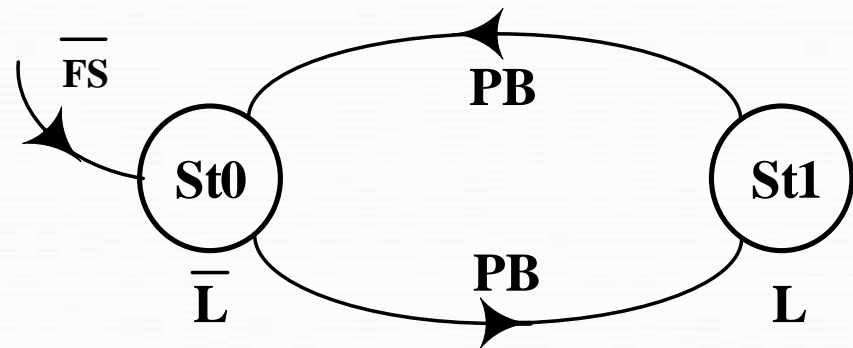
L

St0

0

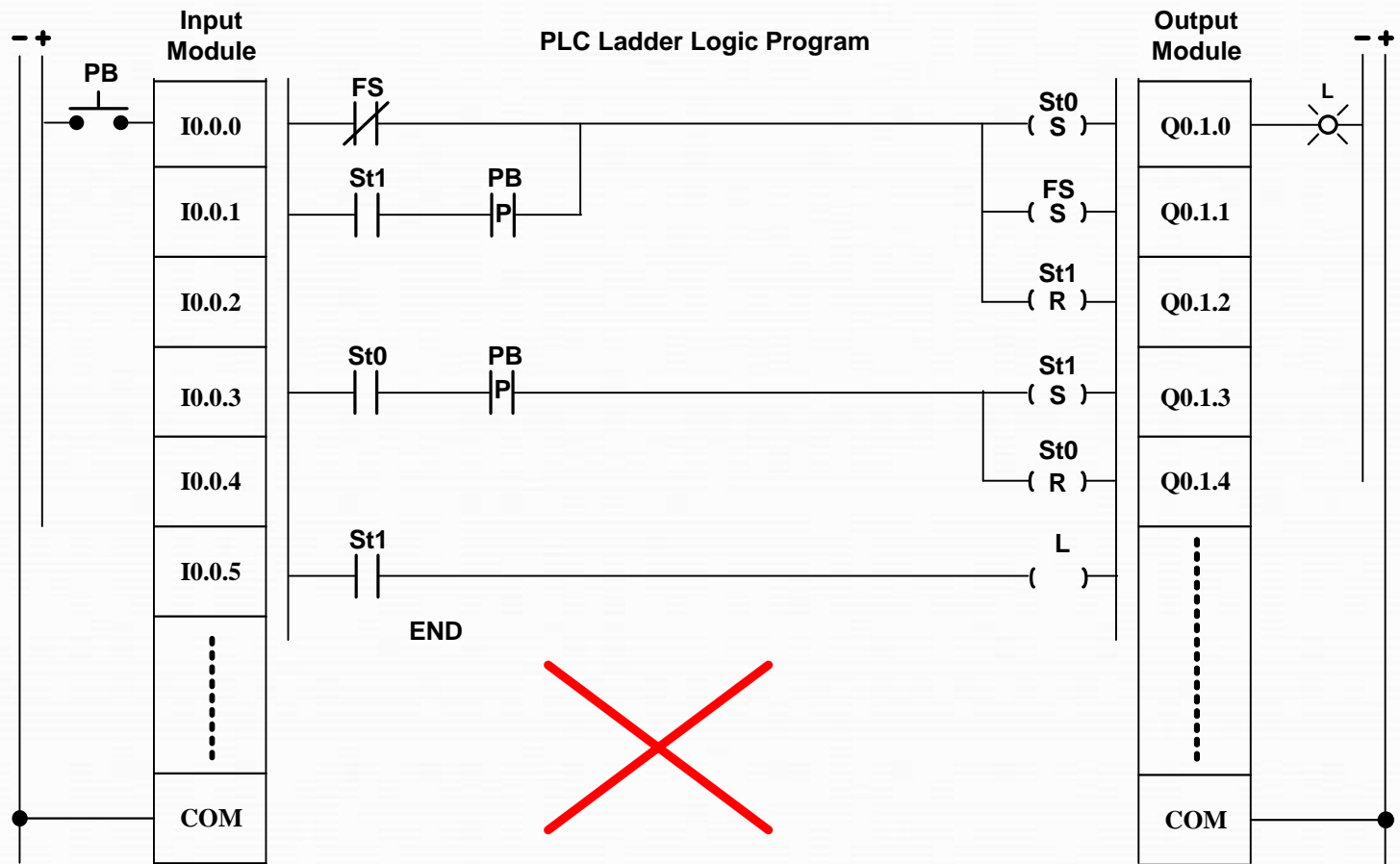
St1

1



Outputs Equations

$$L = St1$$



How the PLC program will be if one push button is used instead of two buttons for the same example

Inputs

Push Button (PB) - NO

Outputs

Light (L)

Outputs

States

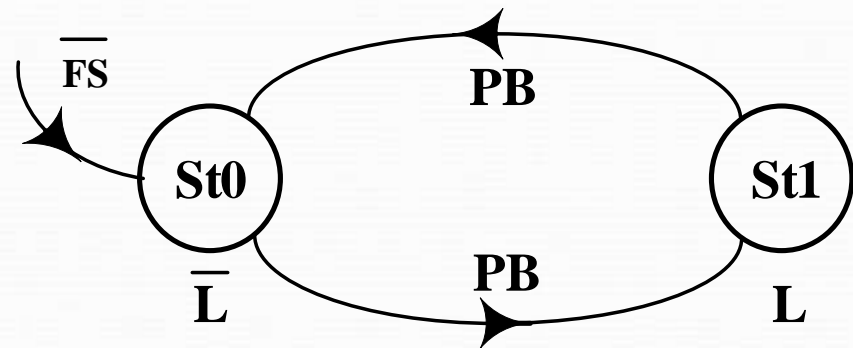
L

St0

0

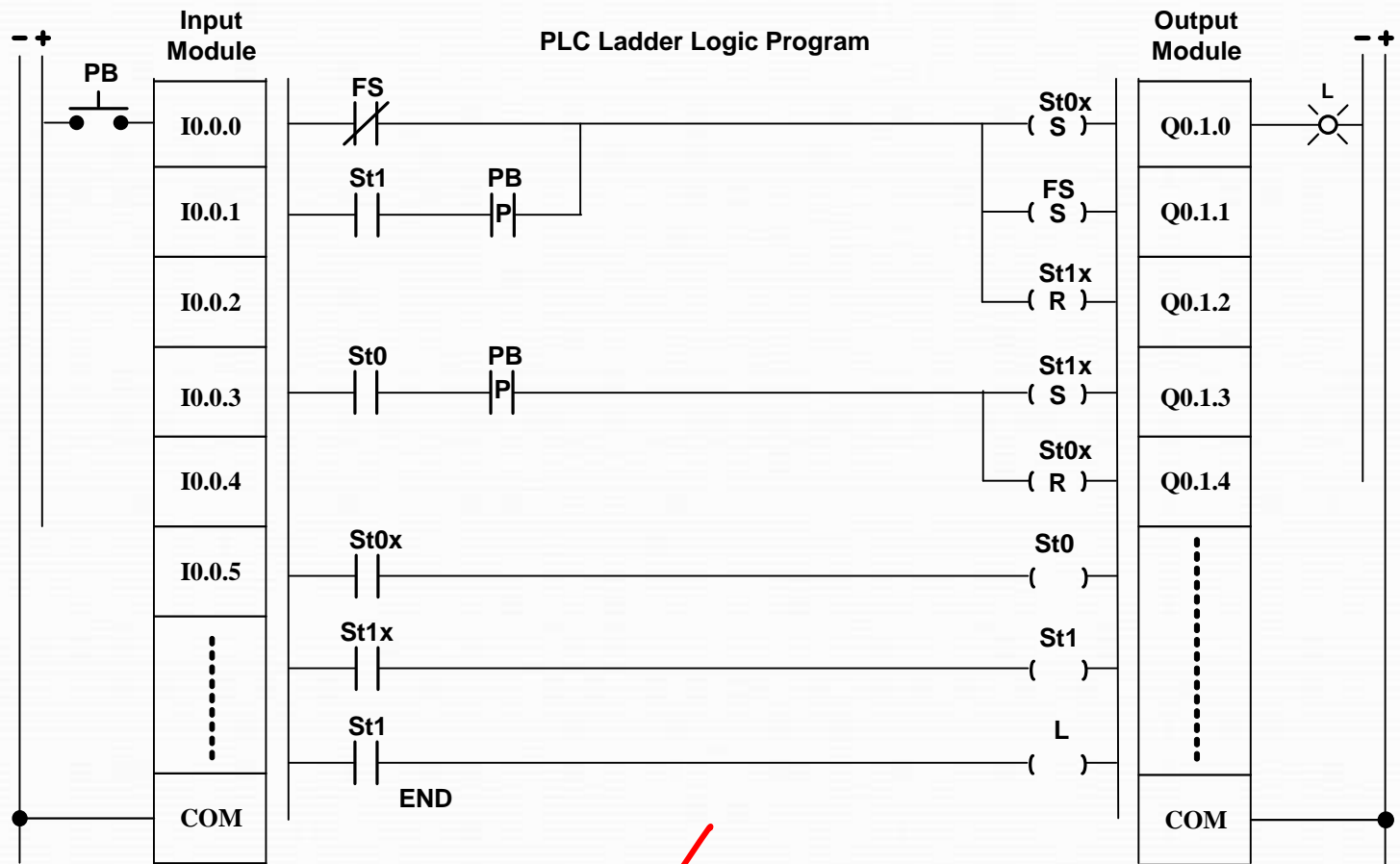
St1

1



Outputs Equations

$$L = St1$$



## Example

Devise the PLC program that could be used to control a store light as follows:

- The light is **turned off** at system start up.
- When the **light at "off" state**, the system **turns it on** as the **push button is activated** or a **movement is detected**.
- When the **light at "on" state**, the system **turns it off** as the **push button is activated** or when **no movement is detected** for 10 minutes or more.

### Inputs

Push Button (PB) - NO

Movement Detector (MD) -NO

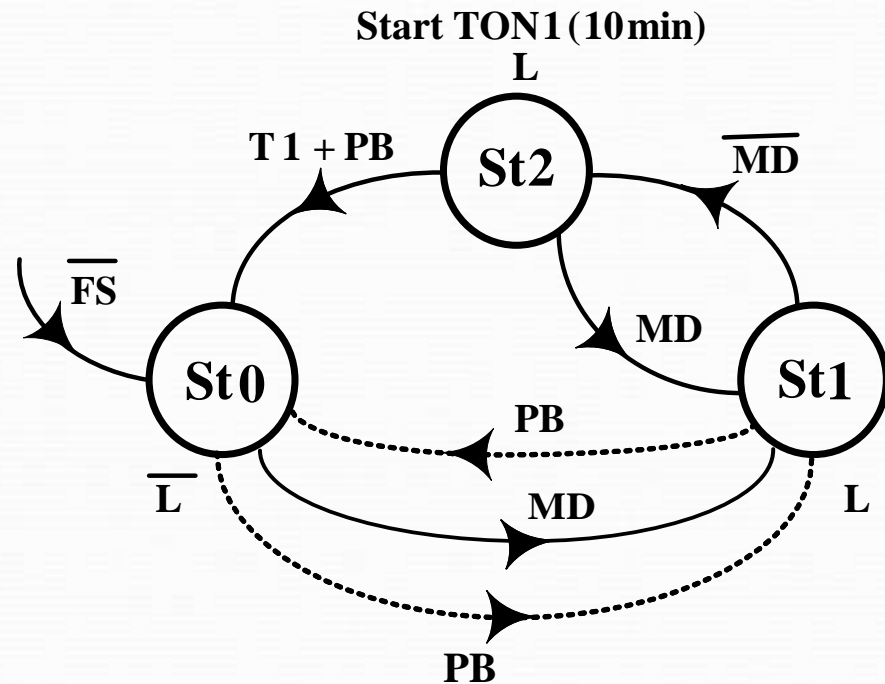
### Outputs

Light (L)

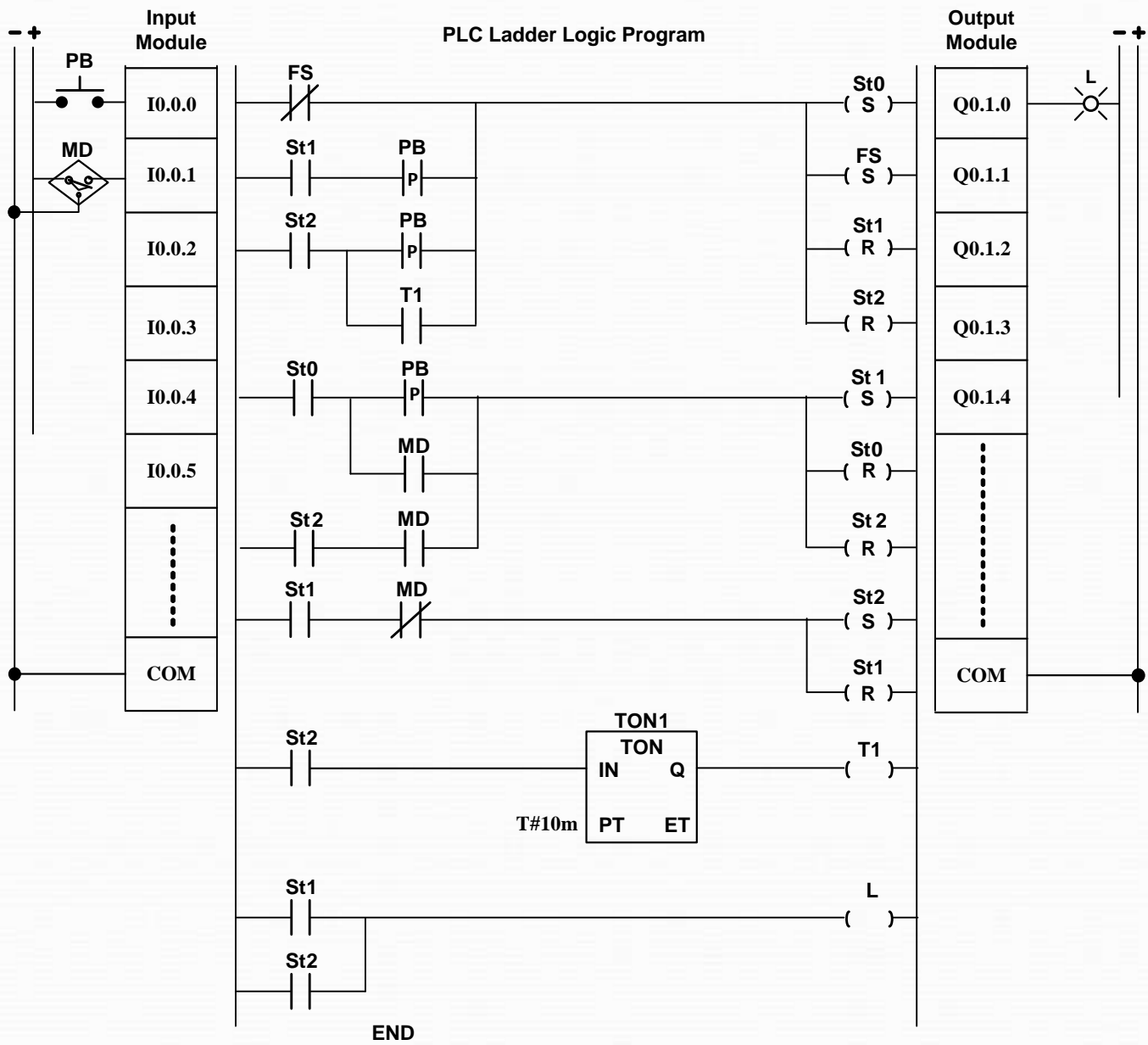
<u>States</u>	<u>Output</u> <u>L</u>
St0	0
St1	1
St2	1

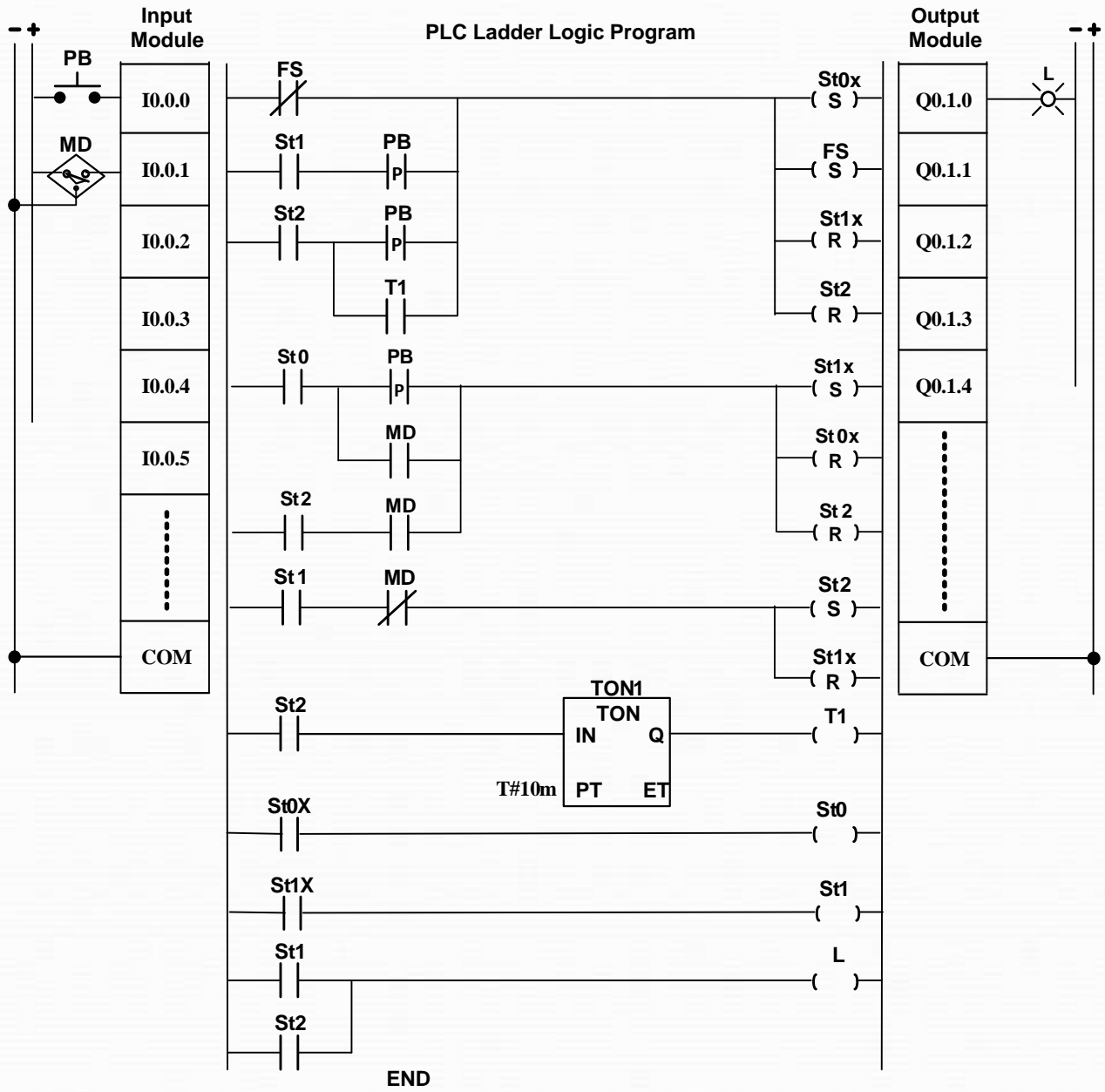
### Output Equation

$$L = St1 + St2$$









## State-Based Design (2)

**Q1:** By using modular PLC type LG write the PLC LLP that generates a square wave of **2Hz** and **60% duty cycle**. Use **NO Start** Button and **NC Stop** Button and a **LED** as indicator using state-based method.

### Solution:

$$T = \frac{1}{f} = \frac{1}{2} = 0.5s = 500ms$$

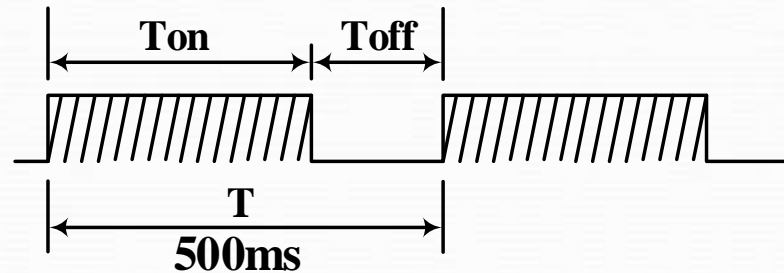
$$\text{Duty Cycle} = \frac{T_{on}}{T}$$

$$T_{on} = T * \text{Duty Cycle}$$

$$= 500 * 0.60 = 300ms$$

$$T_{off} = T - T_{on}$$

$$= 500 - 300 = 200ms$$

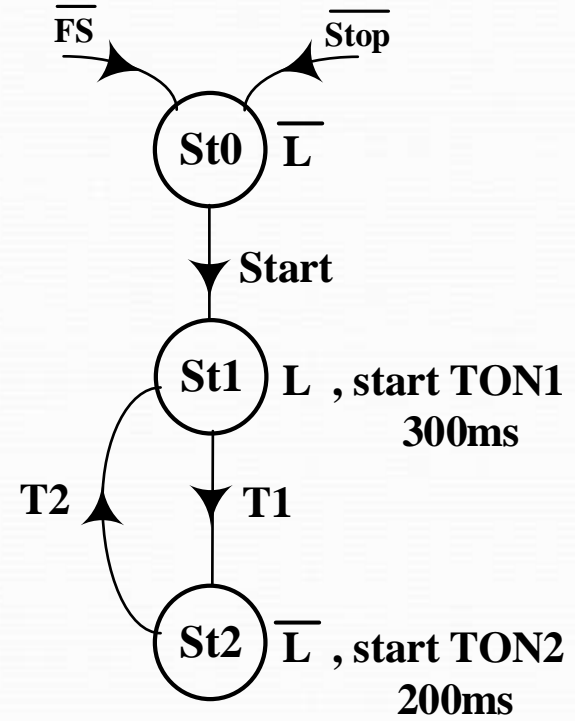


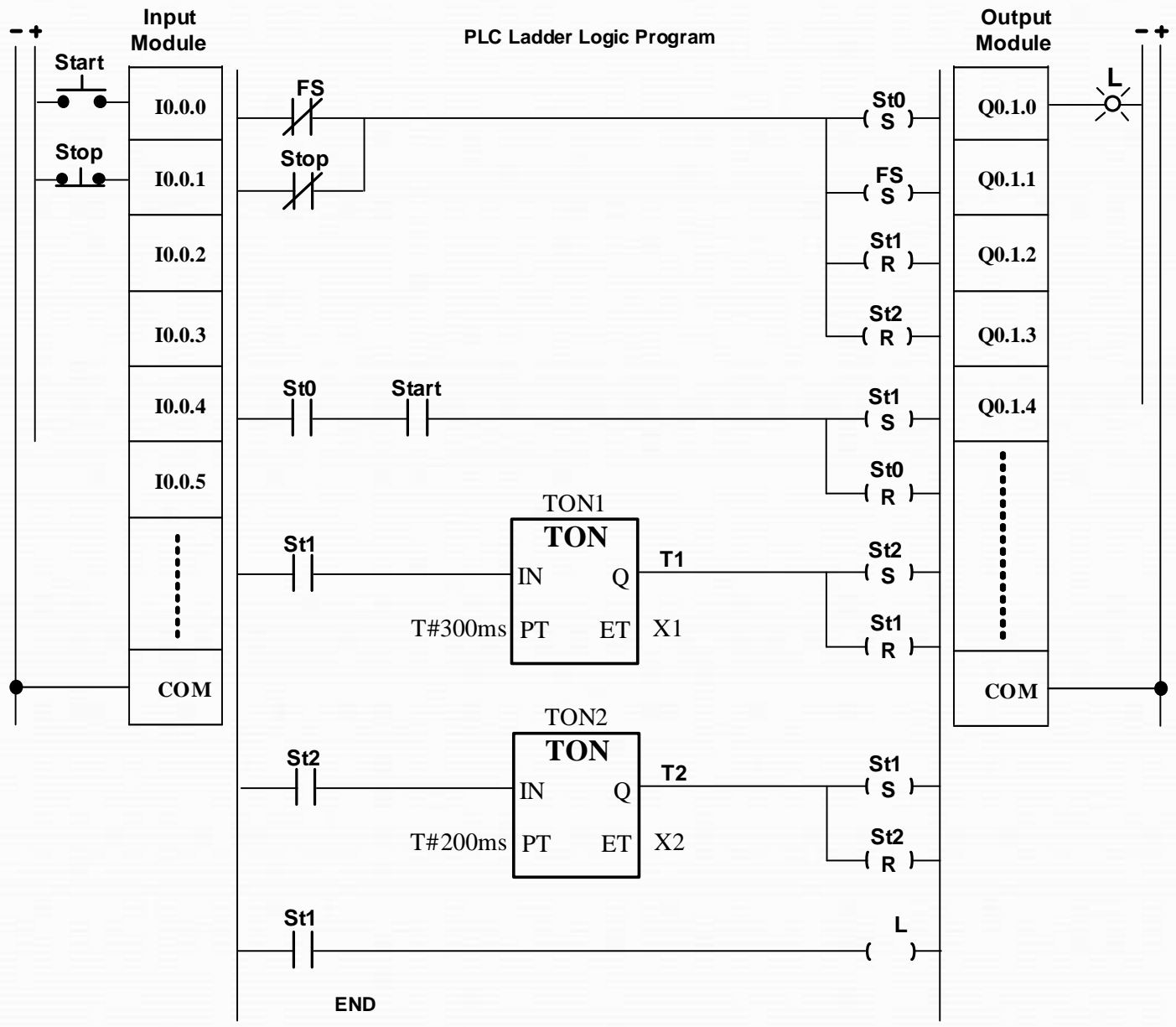
Inputs            Output  
**Start - NO**      **Light ( L )**  
**Stop - NC**

<u>States</u>	<u>Output</u>
<b>St0</b>	<b>0</b>
<b>St1</b>	<b>1</b>
<b>St2</b>	<b>0</b>

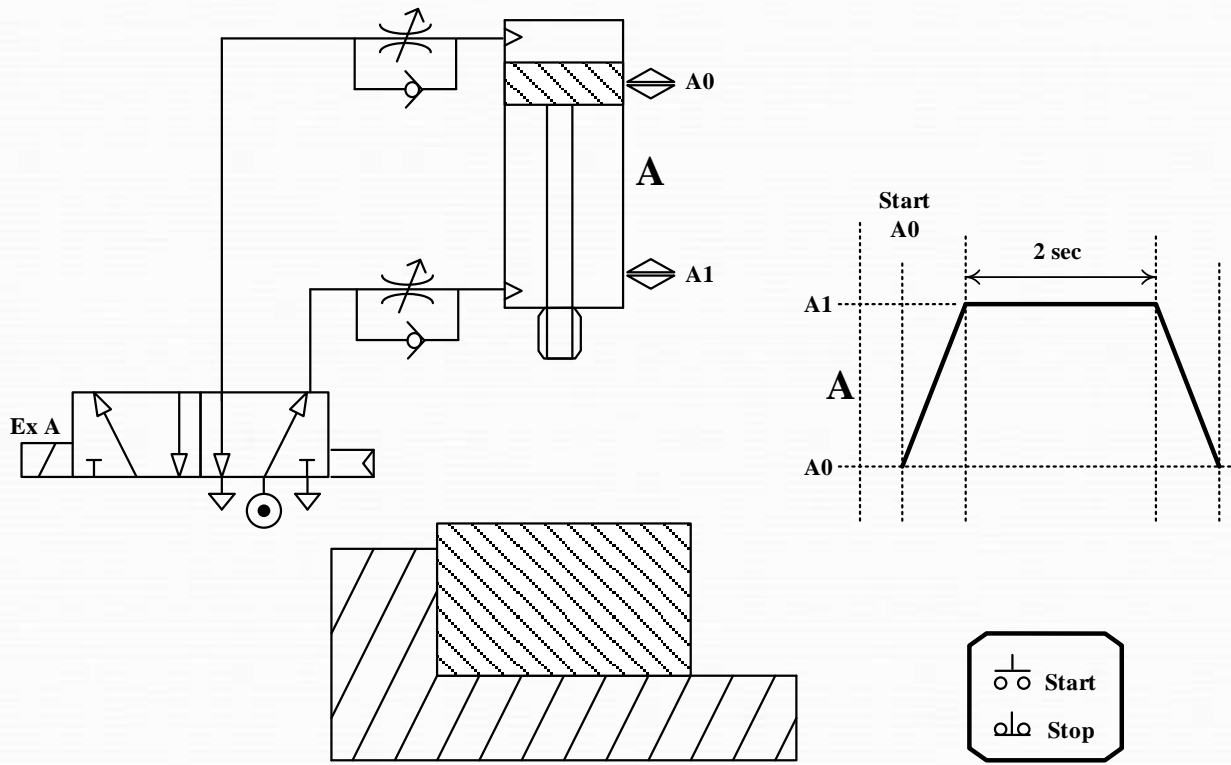
Output Equation

$$L = St1$$





**Q2:** Design the PLC based control system by using modular PLC / Lsis GLOFA with GM4 CPU and devise the PLC ladder logic program that controls the operation of cylinders A in the sequence mode explained below (the sequence is repeated **six times** when start button is pressed and then waits another start to repeat). The system could be stopped at any time and the counter also could be reset at any time.



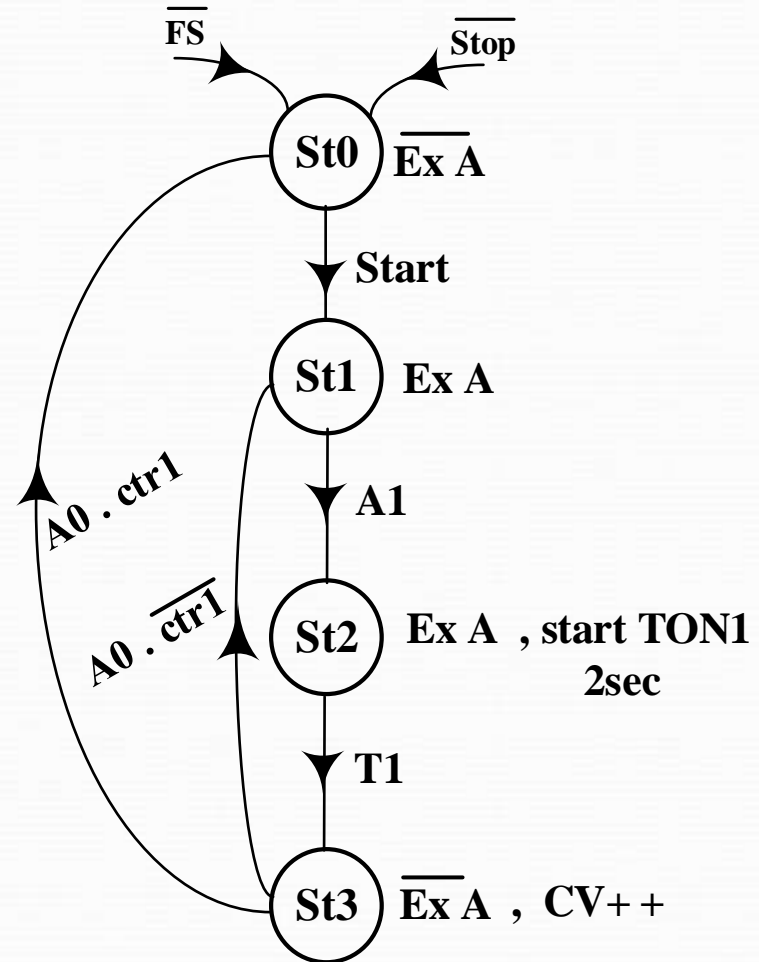
# Solution:

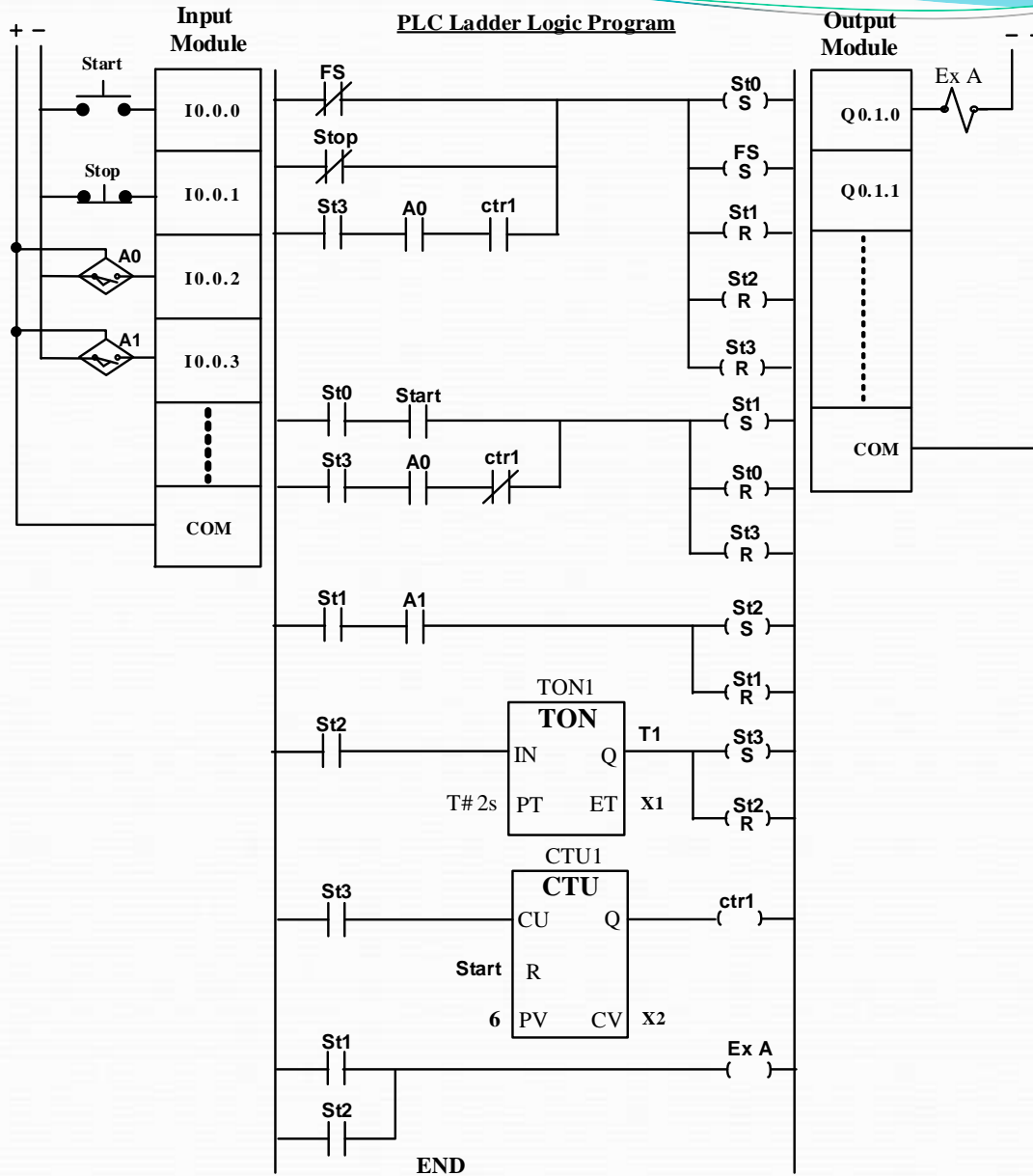
<u>Inputs</u>	<u>Output</u>
Start - NO	Ex A : Extend Sol.
Stop - NC	
A0 - NO	
A1 - NO	

<u>States</u>	<u>Output</u>
	<u>Ex A</u>
St0	0
St1	1
St2	1
St3	0

## Output Equation

$$\text{Ex A} = \text{St1} + \text{St2}$$







# State-Based Design – Case Studies

**Case-1:** By using modular PLC type LG write the PLC LLP that generates a square wave of **2Hz** and **60% duty cycle**. Use **NO Start** Button and **NC Stop** Button and a **LED** as indicator using state-based method.

## Solution:

$$T = \frac{1}{f} = \frac{1}{2} = 0.5s = 500ms$$

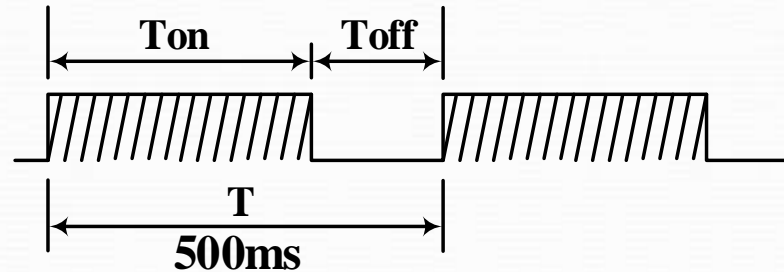
$$\text{Duty Cycle} = \frac{T_{on}}{T}$$

$$T_{on} = T * \text{Duty Cycle}$$

$$= 500 * 0.60 = 300ms$$

$$T_{off} = T - T_{on}$$

$$= 500 - 300 = 200ms$$

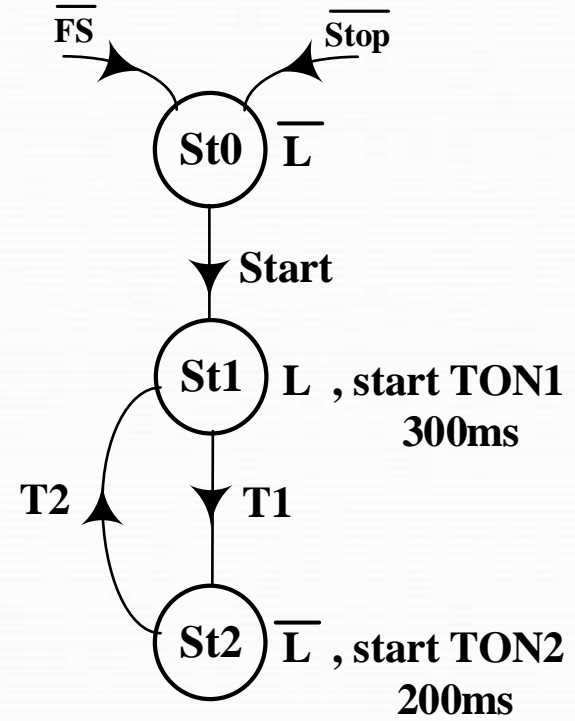


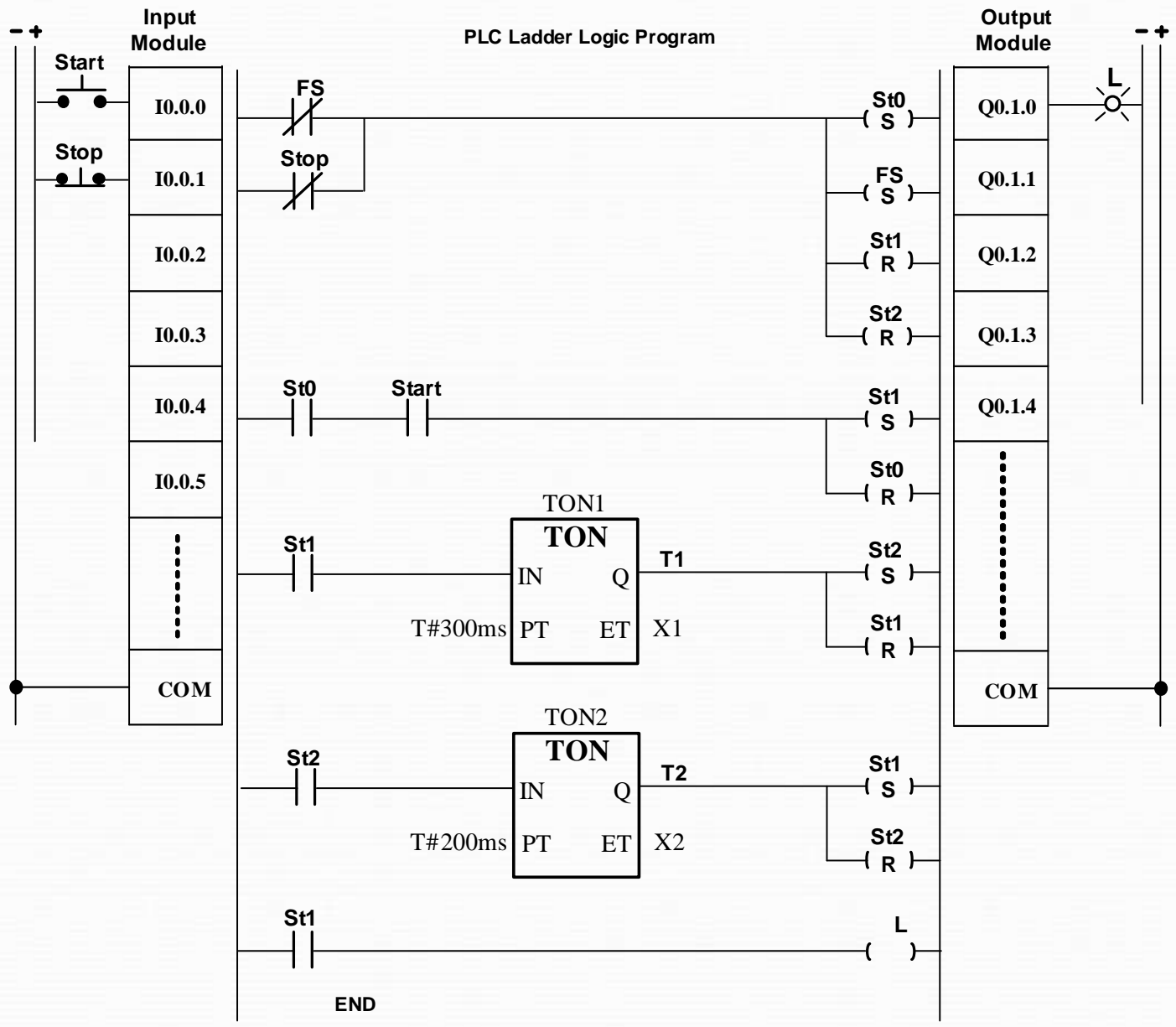
Inputs            Output  
**Start - NO**      **Light ( L )**  
**Stop - NC**

<u>States</u>	<u>Output</u>
<b>St0</b>	<b>0</b>
<b>St1</b>	<b>1</b>
<b>St2</b>	<b>0</b>

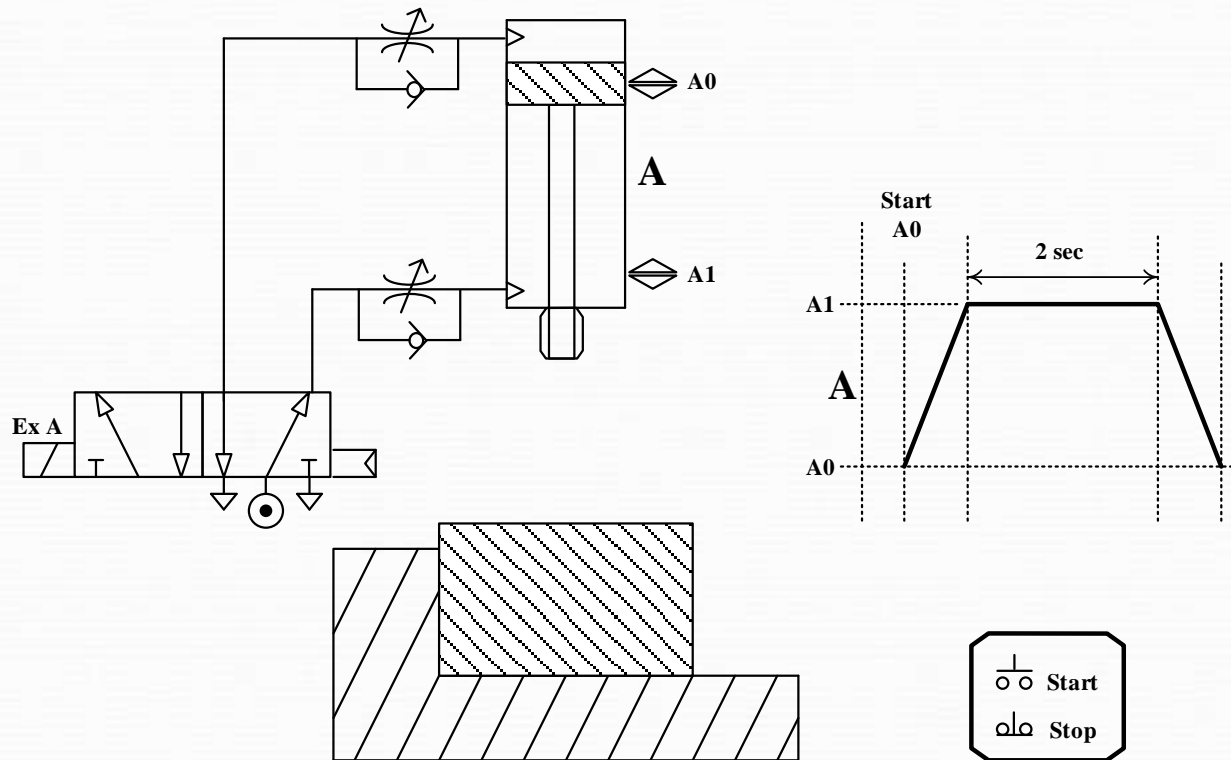
Output Equation

$$L = St1$$





**Case-2:** Design the PLC based control system by using modular PLC / Lsis GLOFA with GM4 CPU and devise the PLC ladder logic program that controls the operation of cylinders A in the sequence mode explained below (the sequence is repeated **six times** when start button is pressed and then waits another start to repeat). The system could be stopped at any time and the counter also could be reset at any time.



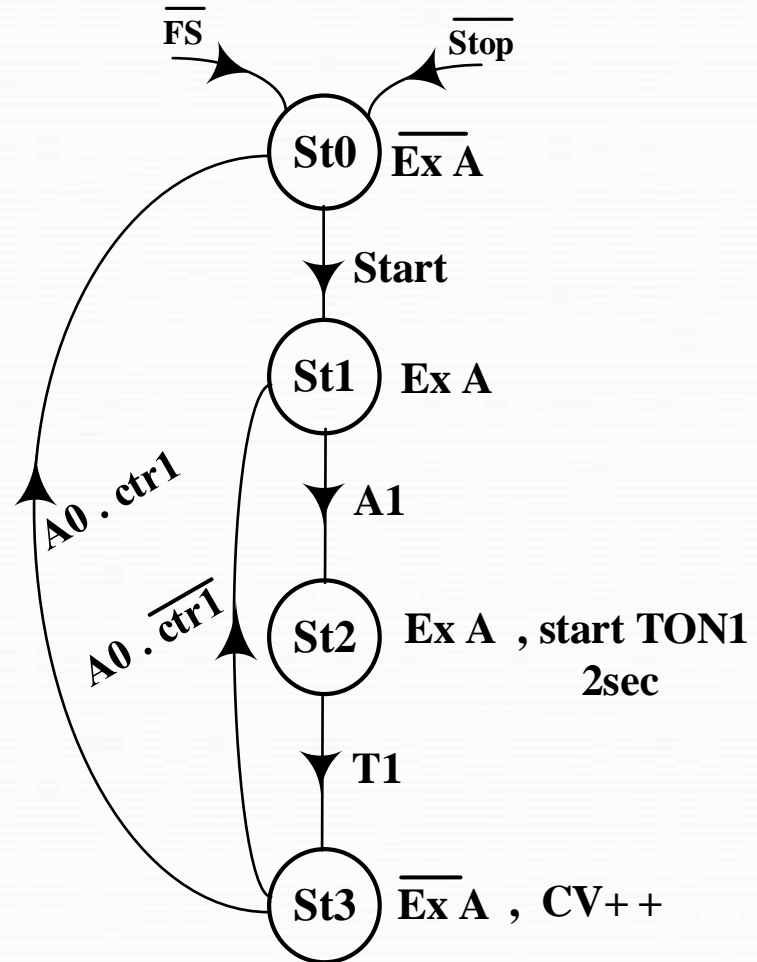
# Solution:

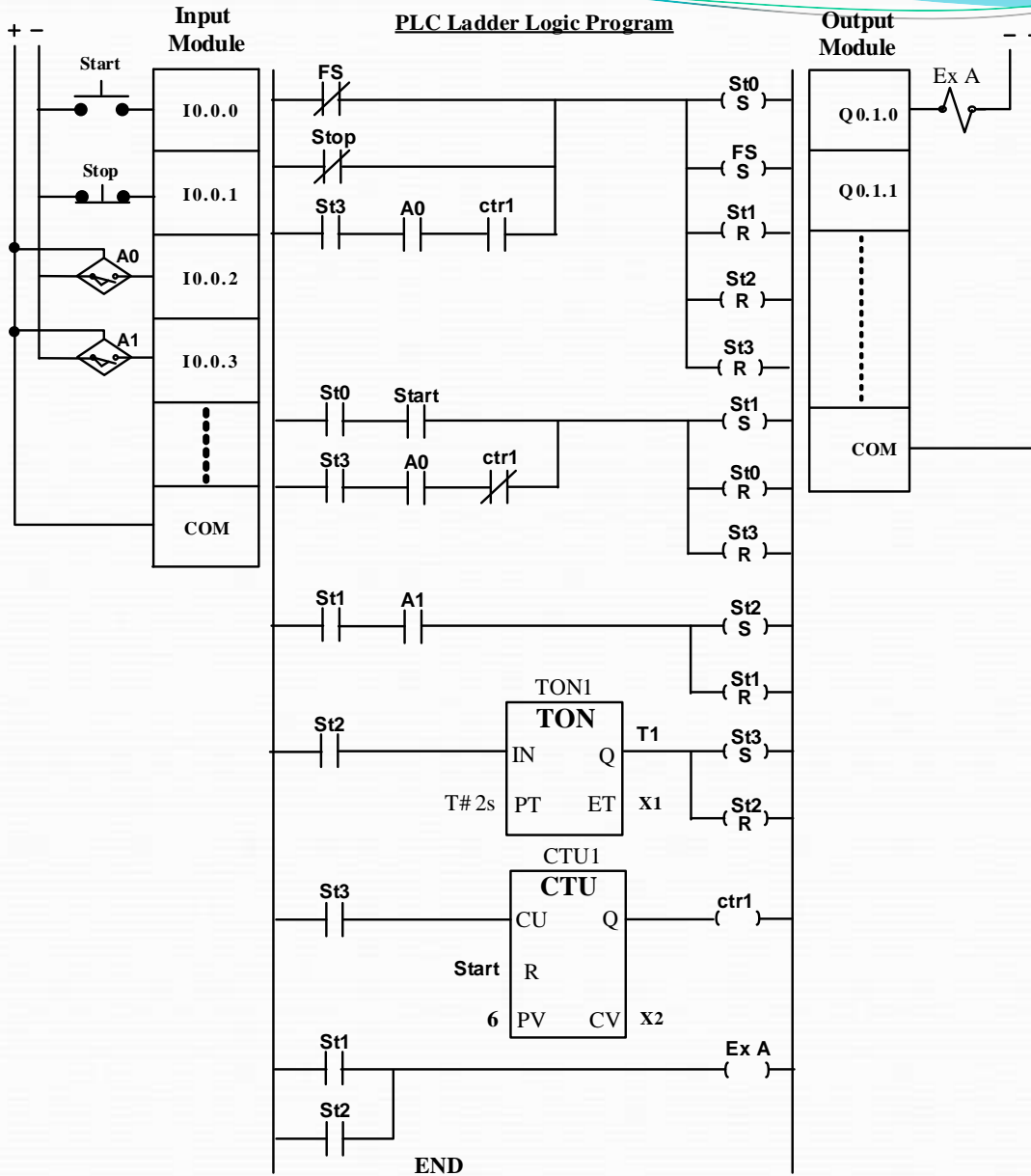
<u>Inputs</u>	<u>Output</u>
Start - NO	Ex A : Extend Sol.
Stop - NC	
A0 - NO	
A1 - NO	

<u>States</u>	<u>Output</u>
	<u>Ex A</u>
St0	0
St1	1
St2	1
St3	0

## Output Equation

$$\text{Ex A} = \text{St1} + \text{St2}$$



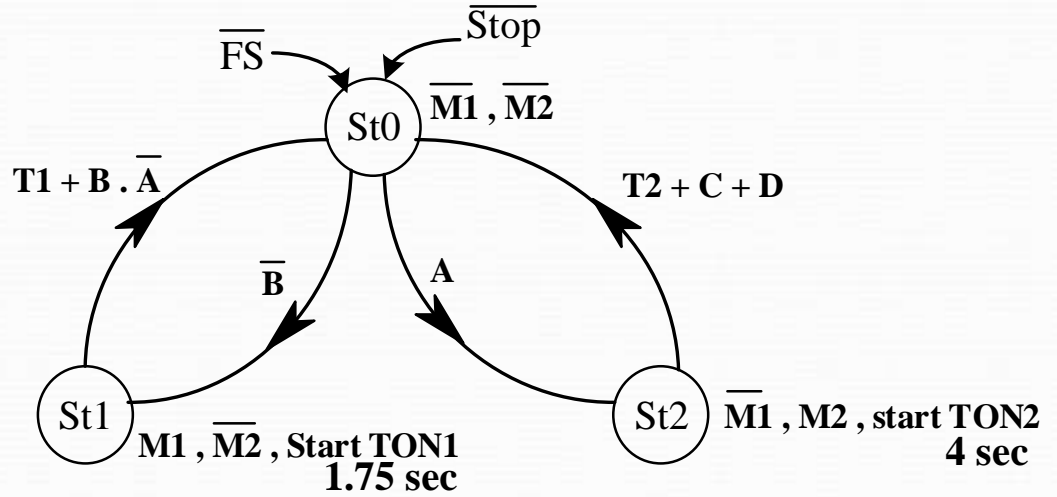


**Case-3:** Implement the PLC LLP for the following state diagram.

<u>Inputs</u>	<u>Outputs</u>	
Stop (NC)	M1	
A	M2	
B		
C		
D		

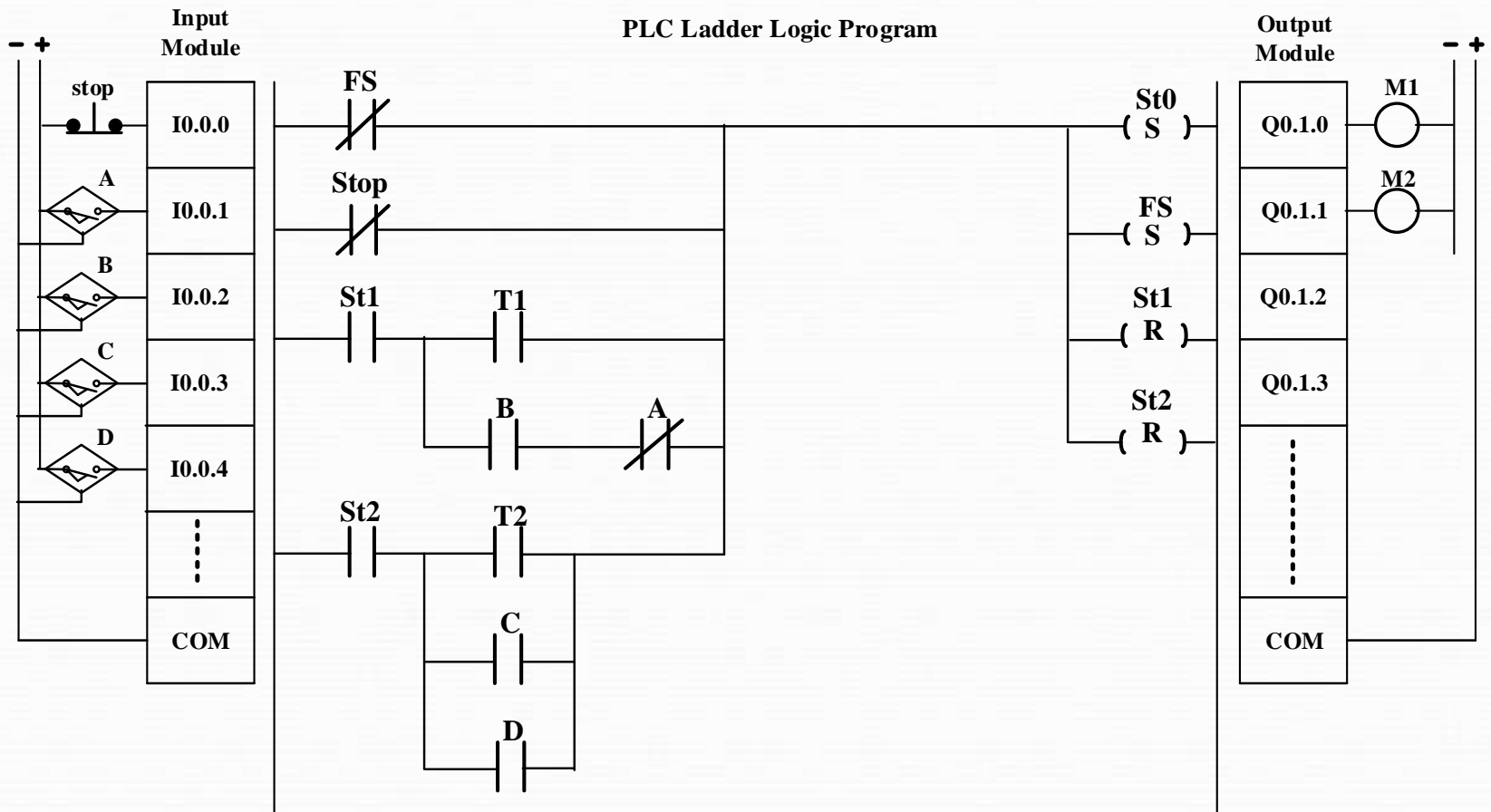
<u>States</u>	<u>Outputs</u>	
	<u>M1</u>	<u>M2</u>
St0	0	0
St1	1	0
St2	0	1



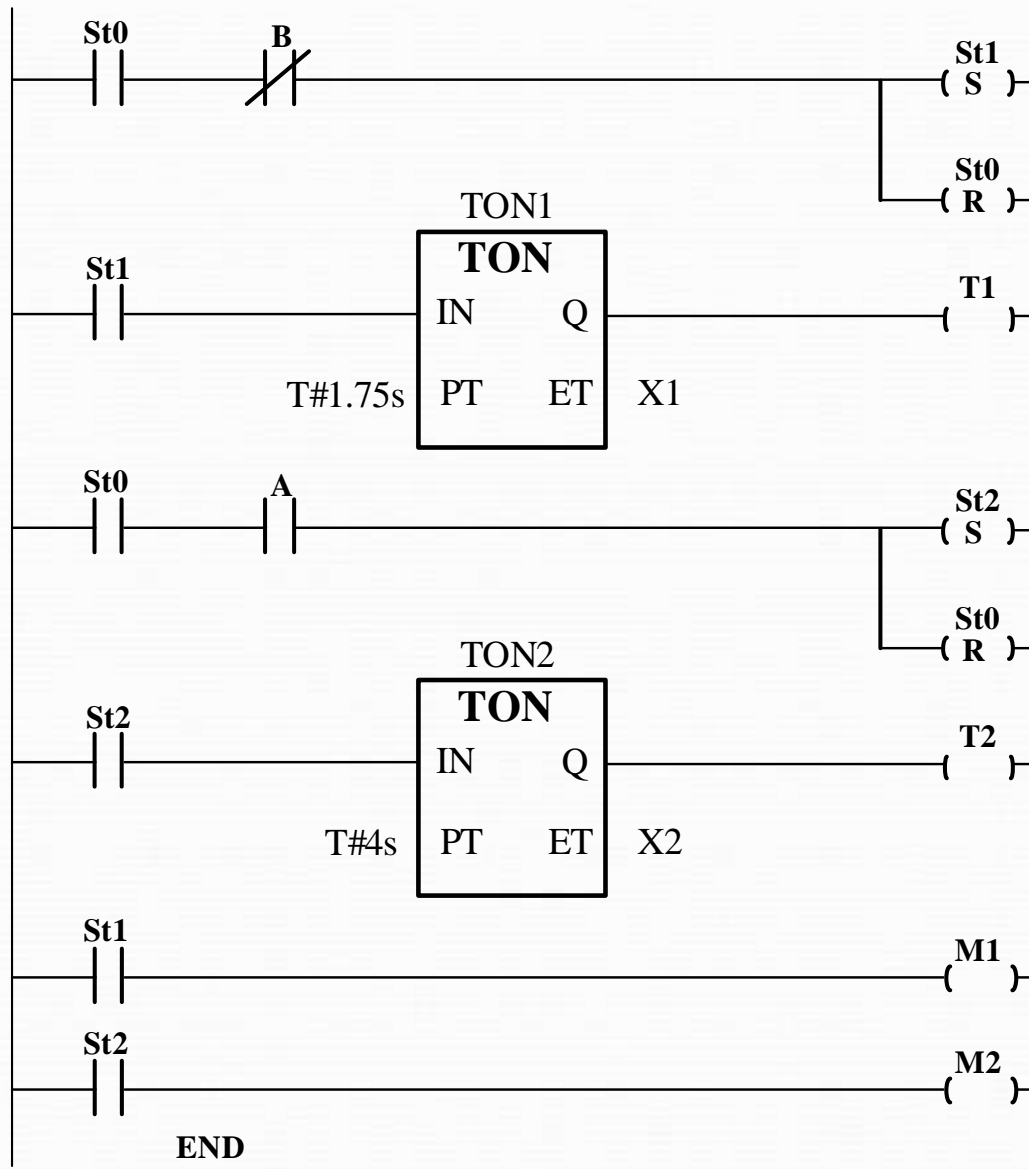
**Outputs Equations**

$M1 = St1$

$M2 = St2$







**Case-4**: Design the PLC based control system by using modular PLC / Lsis GLOFA with GM4 CPU and devise the PLC ladder logic program that controls a wrapping process using state based design. The general sequence of operations is described below.

1. The folder is idle until a part arrives.
2. When a part arrives it triggers the part sensor and the part is held in place by actuating the hold actuator
3. The first wrap is done by turning on output paper for 1 second.
4. The paper is then folded by turning on fold output for 0.5 seconds.
5. An adhesive is applied by turning on output tape for 0.75 seconds.
6. The part is release by turning off output hold.
7. The process pauses until the part sensors goes off, and then the machine returns to idle.

### Inputs

Stop (NC)  
Part Sensor (NO)

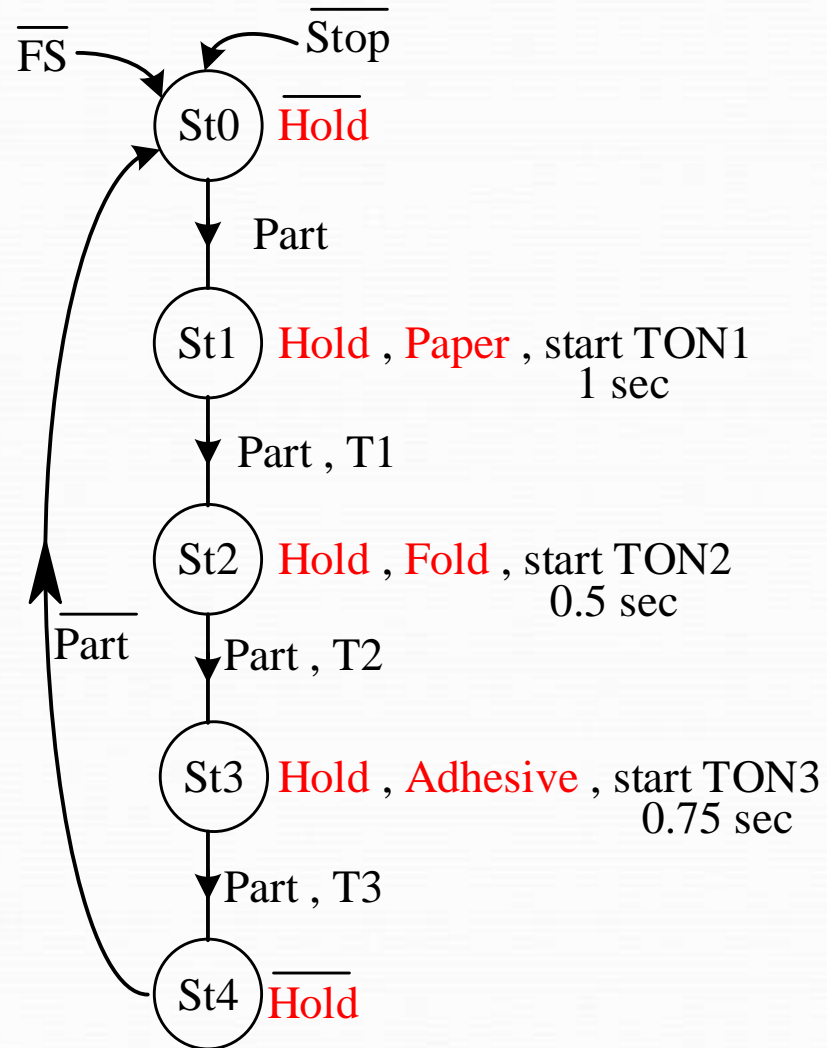
### Outputs

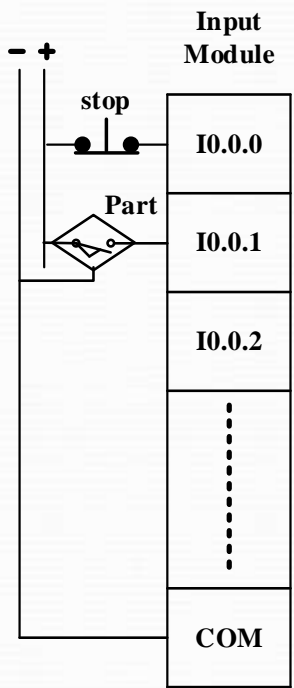
Hold  
Paper  
Fold  
Adhesive

<u>States</u>	<u>Outputs</u>			
	<u>Hold</u>	<u>Paper</u>	<u>Fold</u>	<u>Adhesive</u>
St0	0	0	0	0
St1	1	1	0	0
St2	1	0	1	0
St3	1	0	0	1
St4	0	0	0	0

### Outputs Equations

Hold =  
Paper =  
Fold =  
Adhesive =





PLC Ladder Logic Program

