



**BITS Pilani**  
Pilani Campus

(Merged -  
DEZG516/DMZG511/ESZG51  
1)

**Lecture**



Type	Content Ref.	Topic Title	Study/HW Resource Reference
Pre CH			
During CH	T1	<p>Digital logic: Logic gates, Boolean algebra, applications</p> <p>Programmable logic controllers: Components, programming using ladder logic</p> <p>Programmable logic controllers: Use of relays, timers, counters, registers in ladder logic, application examples etc.</p>	<p>T1: Chapter 5, T2: Chapter 6</p> <p>T1: Chapter 21</p>
Post CH			Chapter end problems



# Digital Logic

# AND Gate

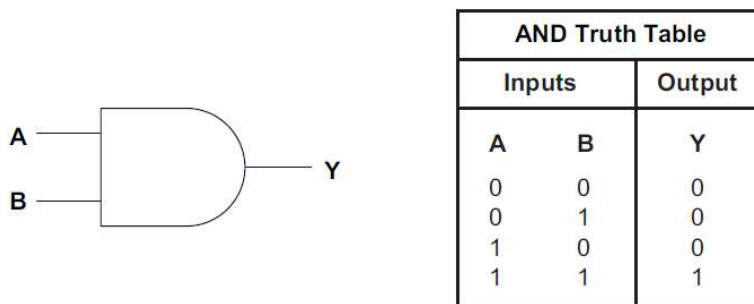
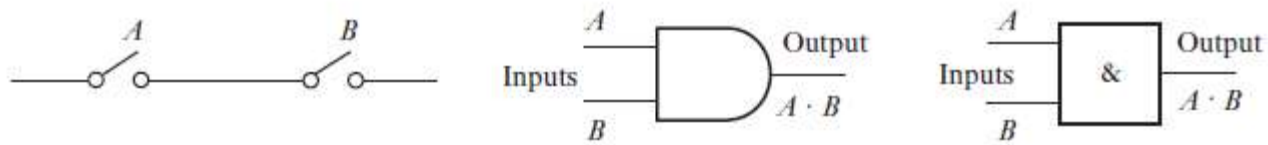
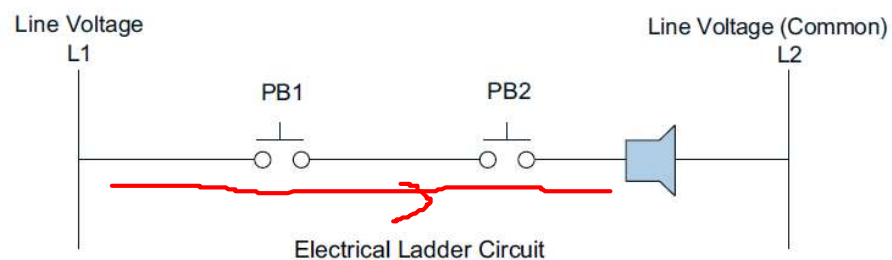
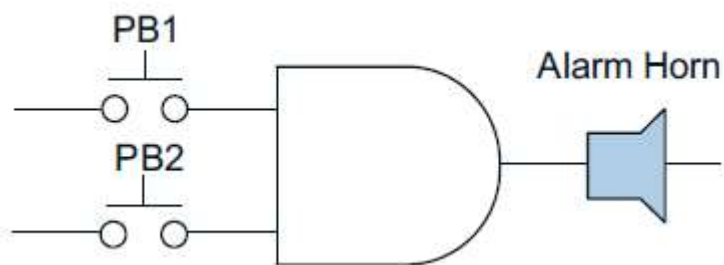
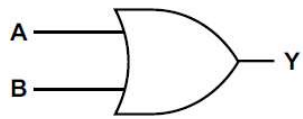


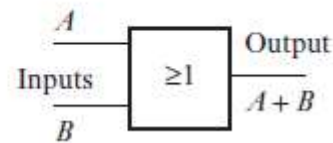
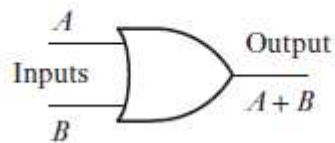
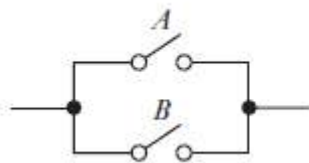
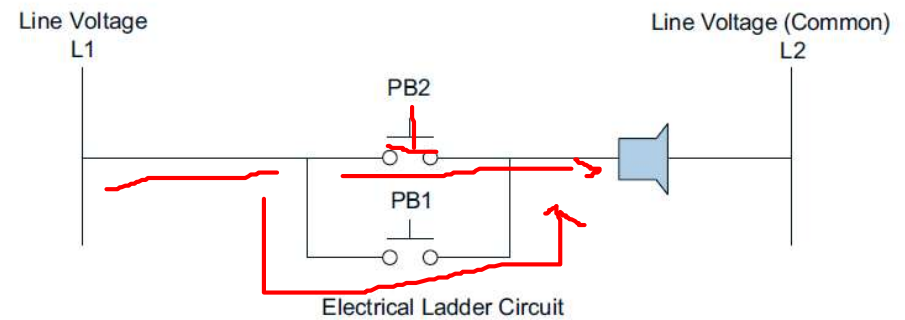
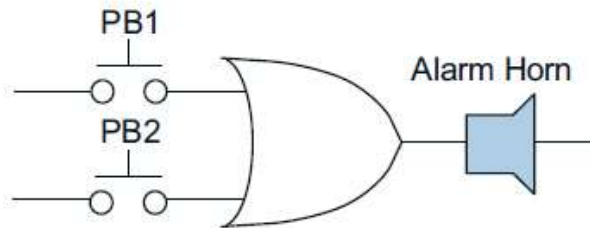
Figure 3-2. Two-input AND gate and its truth table.



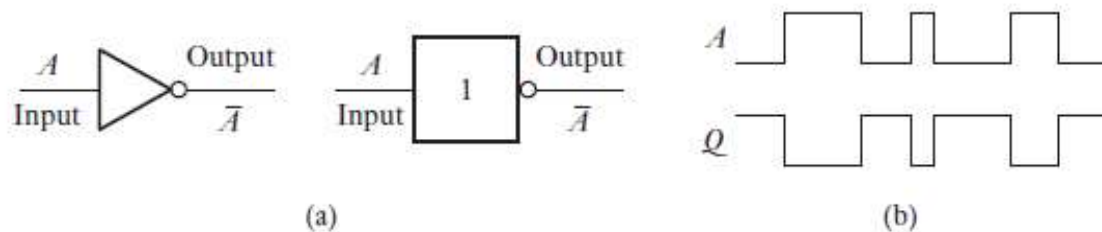
# OR Gate



OR Truth Table		
Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

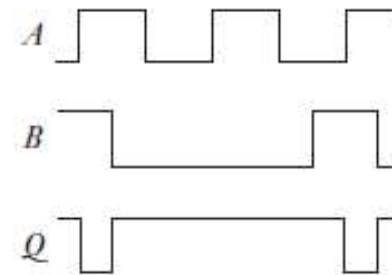
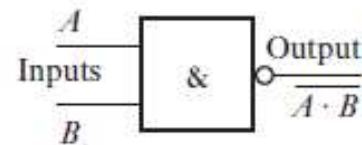
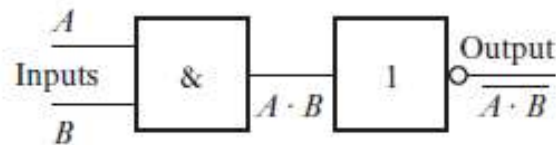
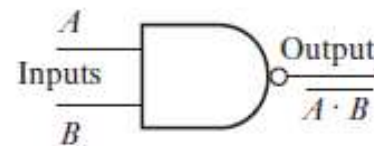
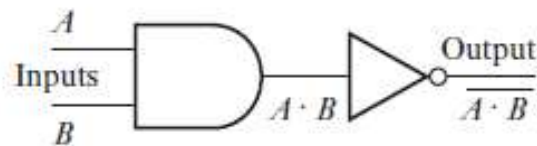


# NOT Gate — Inverter



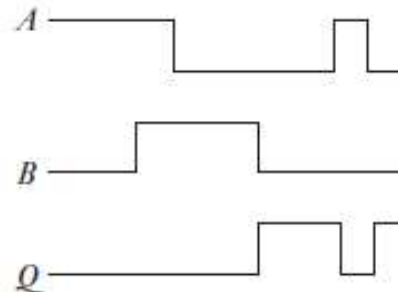
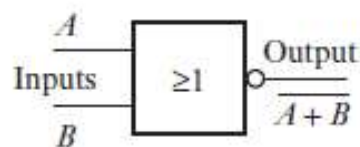
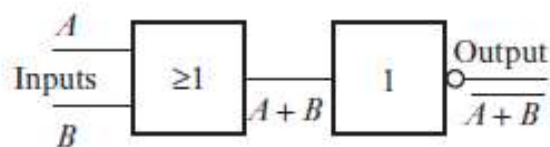
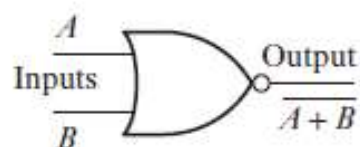
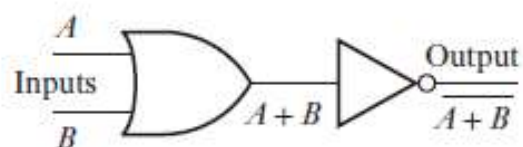
Input $A$	Output $Q$
0	1
1	0

# NAND Gate (AND + NOT)



Inputs		Output $Q$
$A$	$B$	
0	0	1
0	1	1
1	0	1
1	1	0

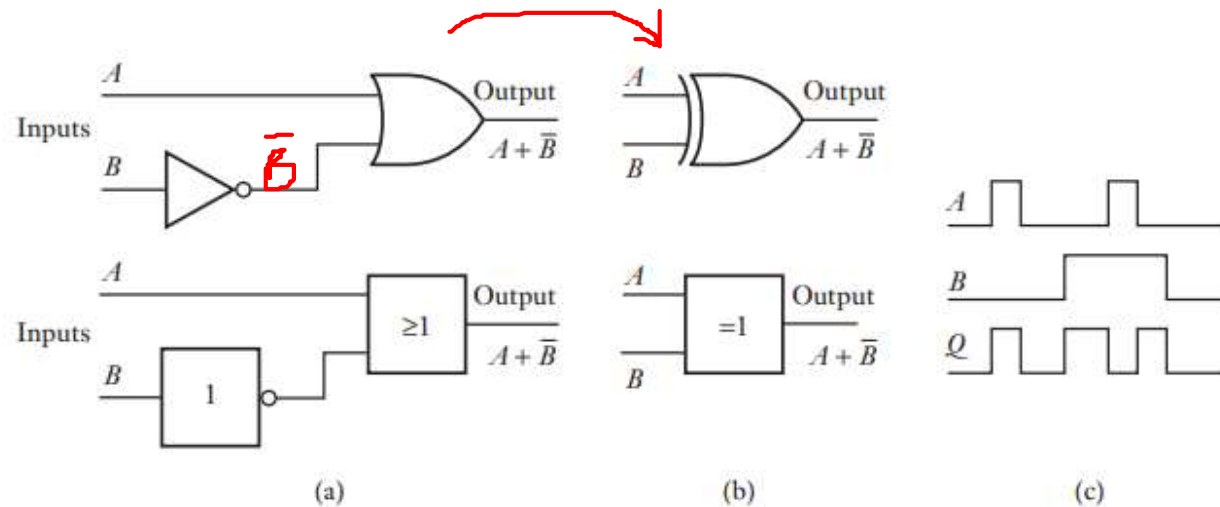
# NOR Gate (OR + NOT)



Inputs		Output $Q$
$A$	$B$	
0	0	1
0	1	0
1	0	0
1	1	0



# XOR (Exclusive OR)

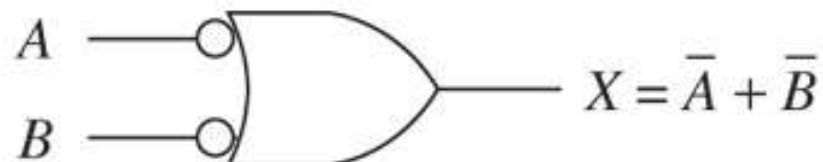
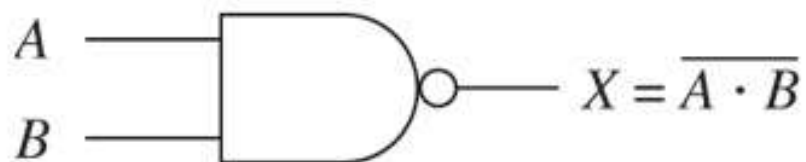


A	B	B-	Y
0	0	1	1
1	0	1	1
0	1	0	0
1	1	0	1

- OR gate with a NOT gate applied to one of the inputs to invert it before the inputs reach the OR gate.
- It can be considered as an AND gate with a NOT gate applied to one of the inputs to invert it before the inputs reach the AND gate.



# NAND equals “Negative OR”



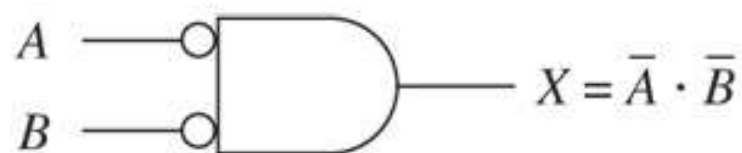
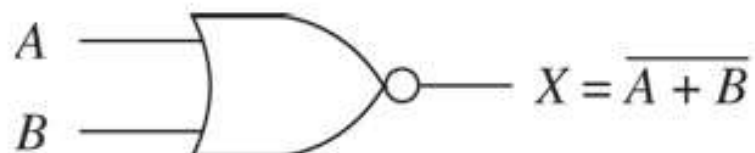
$A$	$B$	$X = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

$A$	$B$	$X = \overline{A} + \overline{B}$
0	0	1
0	1	1
1	0	1
1	1	0

Equivalent  
result



# NOR equals “Negative AND”



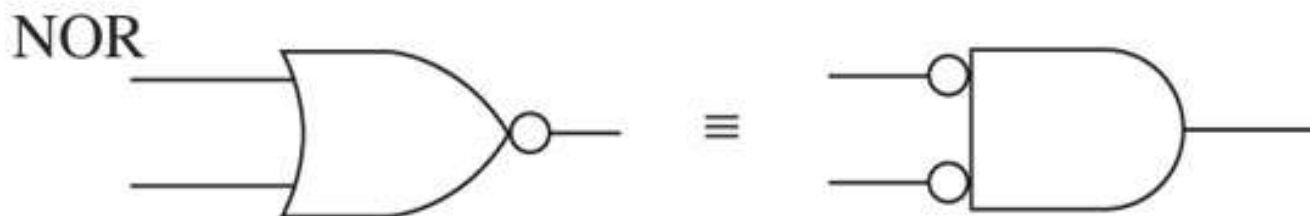
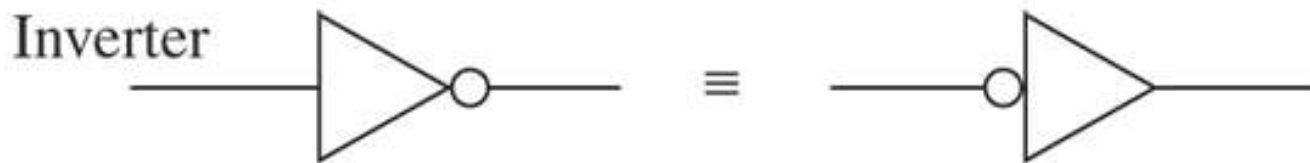
A	B	$X = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

A	B	$X = \bar{A} \cdot \bar{B}$
0	0	1
0	1	0
1	0	0
1	1	0

Equivalent  
result

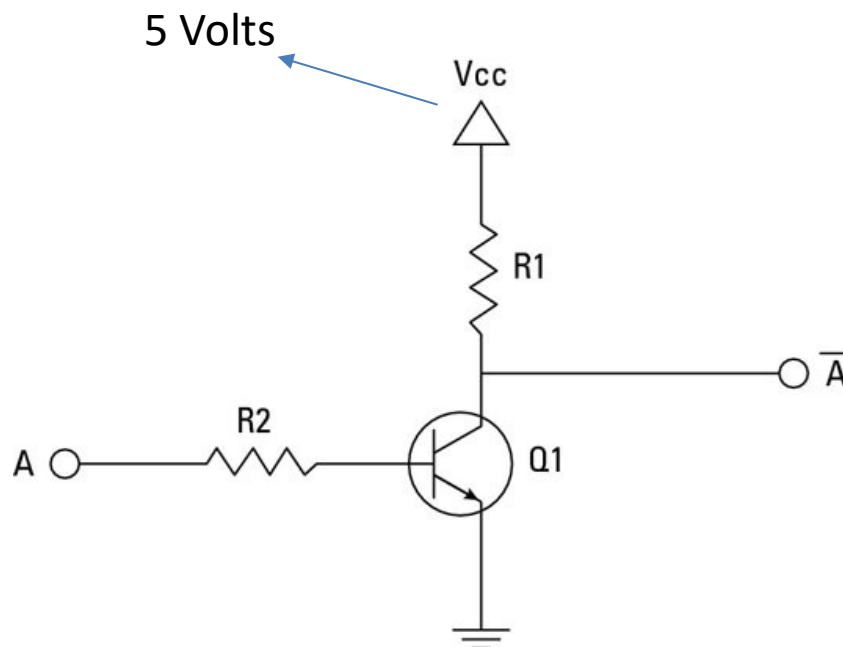
(a)

# Alternative Symbols for Inverter, NAND, and NOR



(c)

# NOT Gate Hardware



When the transistor is off, no current flows through the collector-emitter path. In this way, the circuit's output is HIGH when its input is LOW.

When voltage is present at the input, the transistor turns on, allowing current to flow through the collector-emitter circuit directly to ground. This ground path creates a shortcut that bypasses the output, which causes the output to go LOW.

In this way, the output is HIGH when the input is LOW and LOW when the input is HIGH.



# Boolean Algebra

**Boolean algebra** is a division of mathematics which deals with operations on **logical** values and incorporates binary variables.

$$\left. \begin{array}{l} A + 0 = A \\ A \cdot 1 = A \end{array} \right\}$$

$$\begin{array}{l} A + A' = 1 \\ A \cdot A' = 0 \end{array}$$

$$\left. \begin{array}{l} A + A = A \\ A \cdot A = A \end{array} \right\}$$

$$\left. \begin{array}{l} 1 + A = 1 \\ 0 \cdot A = 0 \end{array} \right\}$$

$$\left. \begin{array}{l} A + B = B + A \\ A \cdot B = B \cdot A \end{array} \right\}$$

$$\left. \begin{array}{l} A \cdot (B + C) = A \cdot B + A \cdot C \\ A + B \cdot C = (A + B) \cdot (A + C) \end{array} \right\} \text{Distributive Law}$$

$$\left. \begin{array}{l} A + (B + C) = (A + B) + C \\ A \cdot (B \cdot C) = (A \cdot B) \cdot C \end{array} \right\}$$

$$\left. \begin{array}{l} \overline{A \cdot B} = \overline{A} + \overline{B} \\ \overline{A + B} = \overline{A} \cdot \overline{B} \end{array} \right\} \text{De Morgan's theorem}$$

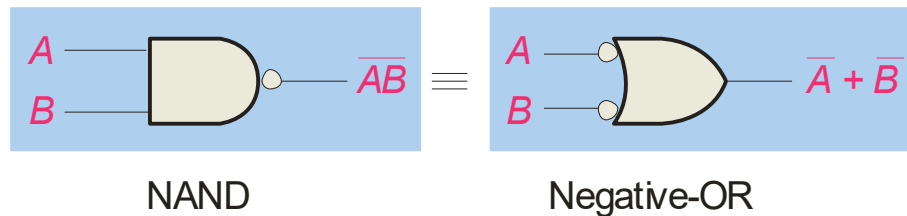
## DeMorgan's Theorems

### DeMorgan's 1<sup>st</sup> Theorem

**The complement of a product of variables is equal to the sum of the complemented variables.**

$$\overline{AB} = \overline{A} + \overline{B}$$

Applying DeMorgan's first theorem to gates:



Inputs		Output	
A	B	$\overline{AB}$	$\overline{A} + \overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

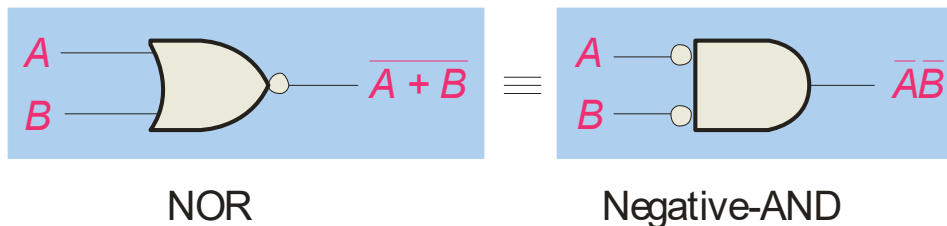
## DeMorgan's Theorems

### DeMorgan's 2<sup>nd</sup> Theorem

**The complement of a sum of variables is equal to the product of the complemented variables.**

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

Applying DeMorgan's second theorem to gates:



Inputs		Output	
A	B	$\overline{A + B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0



## Boolean Addition and Multiplication

The OR operation is often called **Boolean addition**. Variables that are ORed together form a **sum term**.

The AND operation is often called **Boolean multiplication**. Variables that are ANDed together form a **product term**.

**Example #1** The expression  $(A+B+C)(D+E)$  is the product of two sum terms.

**Example #2** The expression  $AB + CD + AD$  is the sum of three product terms.

## Commutative Laws

The **commutative law of addition** states that

**The order in which variables are ORed makes no difference.**

$$A + B = B + A$$

The **commutative law of multiplication** states that

**The order in which variables are ANDed makes no difference.**

$$AB = BA$$

## Associative Laws

The **associative law of addition** states that

**When ORing more than two variables, the result is the same regardless of the grouping of the variables.**

$$A + (B + C) = (A + B) + C$$

The **associative law of multiplication** states that

**When ANDing more than two variables, the result is the same regardless of the grouping of the variables.**

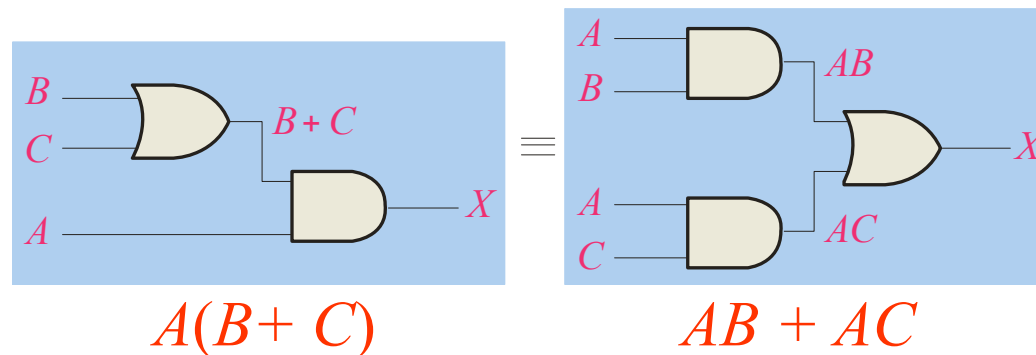
$$A(BC) = (AB)C$$

## Distributive Law

The **distributive law** is the factoring law. A common variable can be factored from an expression just as in ordinary algebra. That is

$$AB + AC = A(B + C)$$

The distributive law can be illustrated with equivalent circuits:



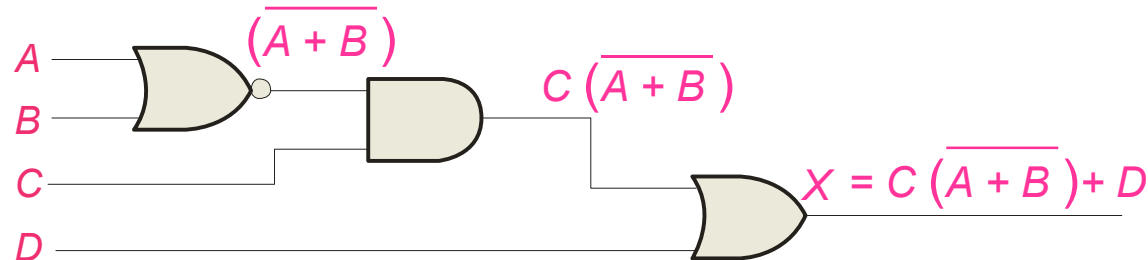
## Boolean Analysis of Logic Circuits

Combinational logic circuits can be analyzed by writing the expression for each gate and combining the expressions according to the rules for Boolean algebra.

### Example Solution

Apply Boolean algebra to derive the expression for  $X$ .

Write the expression for each gate:



Applying DeMorgan's theorem and the distribution law:

$$X = C(\bar{A} \bar{B}) + D = \bar{A} \bar{B} C + D$$

# Example

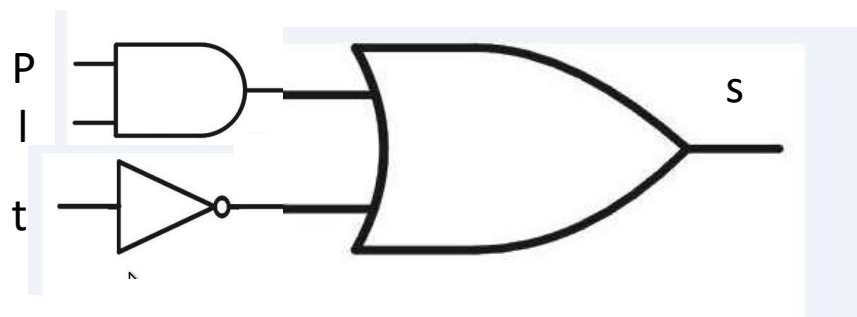
An electronic switch that uses digital logic is to be developed for switching the lights on and off in an art gallery. The switch has to be turned on at 7:00 p.m. and turned off at 6:00 a.m. Also, if there are people in the gallery and the light that enters the gallery from outside is inadequate, the lights have to be turned on regardless of the time of the day.

There are 3 sensors (Human, Light, Timer) – Timer goes high if time is between 6.00 AM to 7.00 PM.

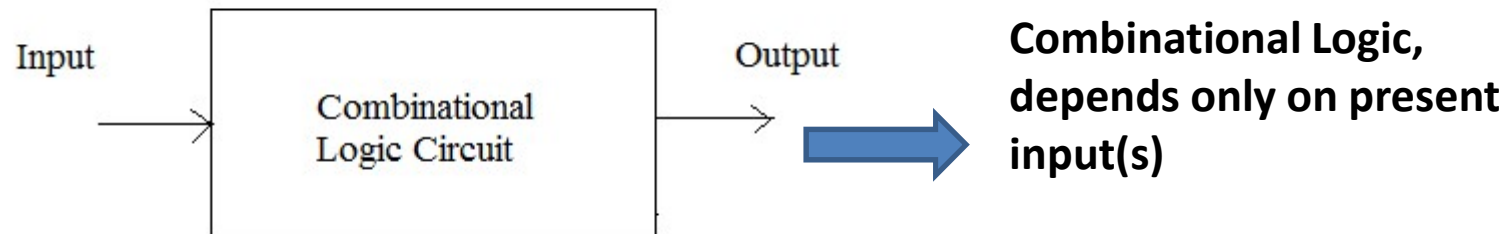
The logic state of the people sensor is denoted by  $p$ , that of the light sensor is denoted by  $l$ , and that of the time sensor is denoted by  $t$ .

The output of the logic circuit is denoted by  $s$

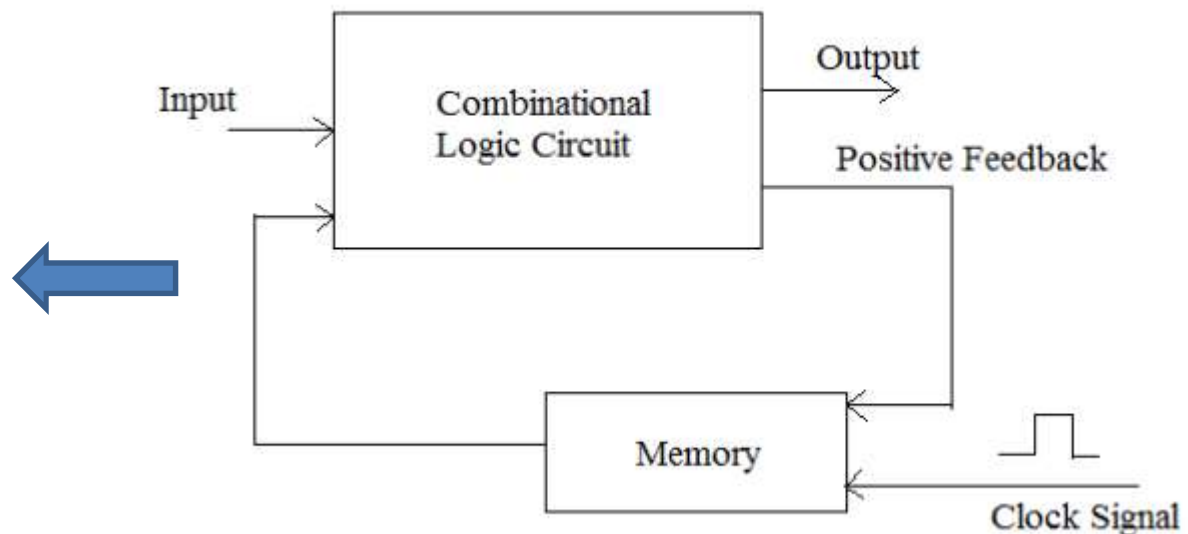
$$s = p \cdot l + \bar{t}$$



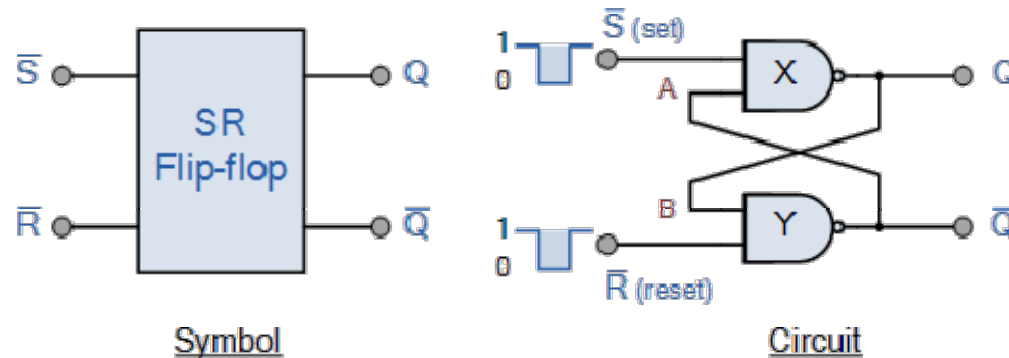
# Logic types



**Sequential Logic,  
Considers earlier data  
also, therefore needs  
memory**



# SR flip-flop - cross-coupled 2-input NAND gates

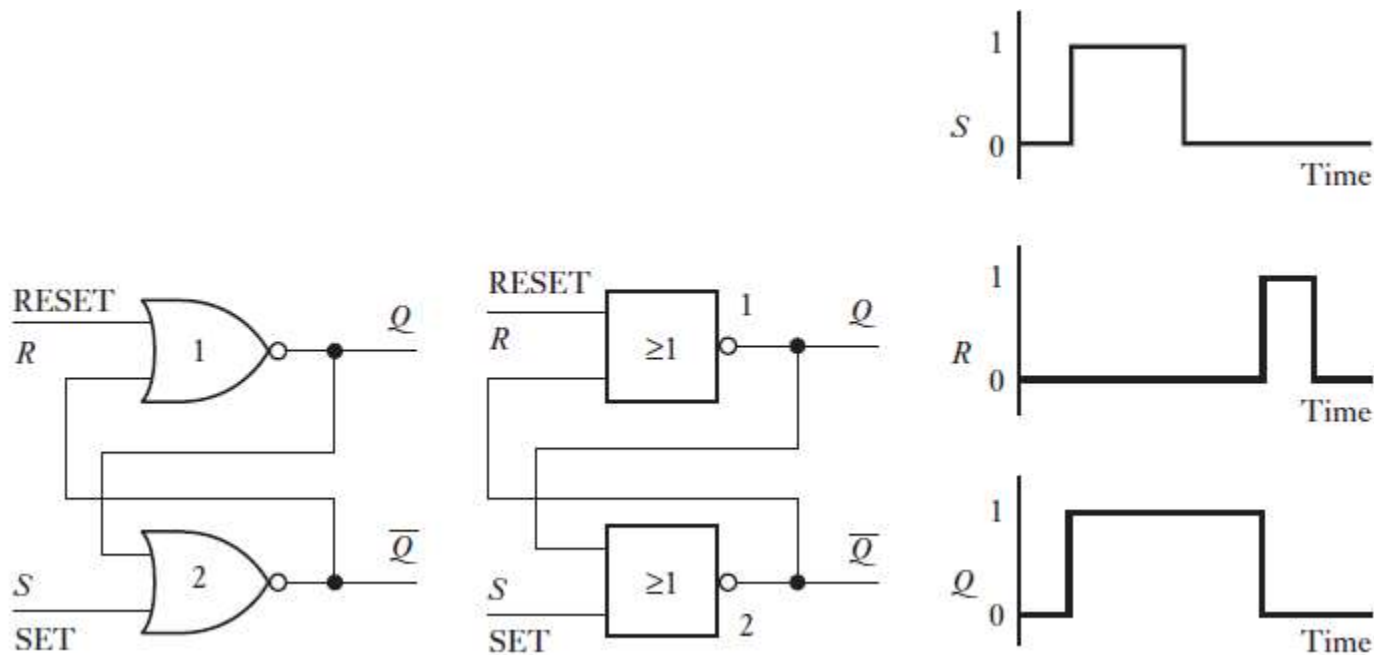


Truth Table for this Set-Reset Function

State	S	R	Q		Description
Set	1	0	0	1	Set $\bar{Q} \gg 1$
	1	1	0	1	no change
Reset	0	1	1	0	Reset $\bar{Q} \gg 0$
	1	1	1	0	no change
Invalid	0	0	1	1	Invalid Condition

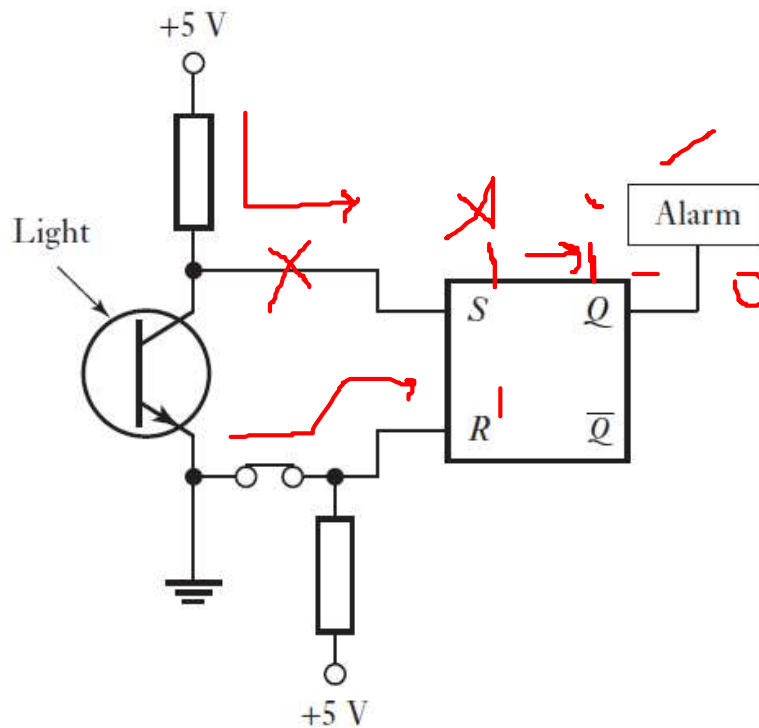


# S – R Flip Flop (Memory Device)



$S$	$R$	$Q_i \rightarrow Q_{i+1}$	$\bar{Q}_i \rightarrow \bar{Q}_{i+1}$
0	0	$0 \rightarrow 0$	$1 \rightarrow 1$
0	0	$1 \rightarrow 1$	$0 \rightarrow 0$
0	1	$0 \rightarrow 0$	$1 \rightarrow 1$
0	1	$1 \rightarrow 0$	$0 \rightarrow 0$
<u>1</u>	0	$0 \rightarrow 1$	$1 \rightarrow 0$
1	0	$1 \rightarrow 1$	$0 \rightarrow 0$
1	1	Not allowed	
1	1	Not allowed	

# Alarm circuit using Flip flop



A phototransistor might be used as the sensor.

When it is illuminated it gives a virtually 0 V input to S,

When the illumination ceases it gives about 5 V input to S.

When the light beam is interrupted, S becomes 1 and the output from the flip-flop becomes 1 and the alarm sounds.

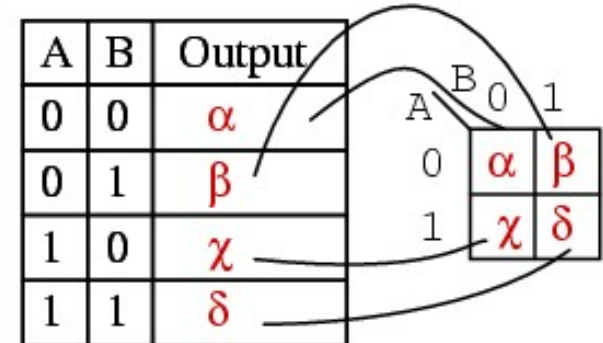
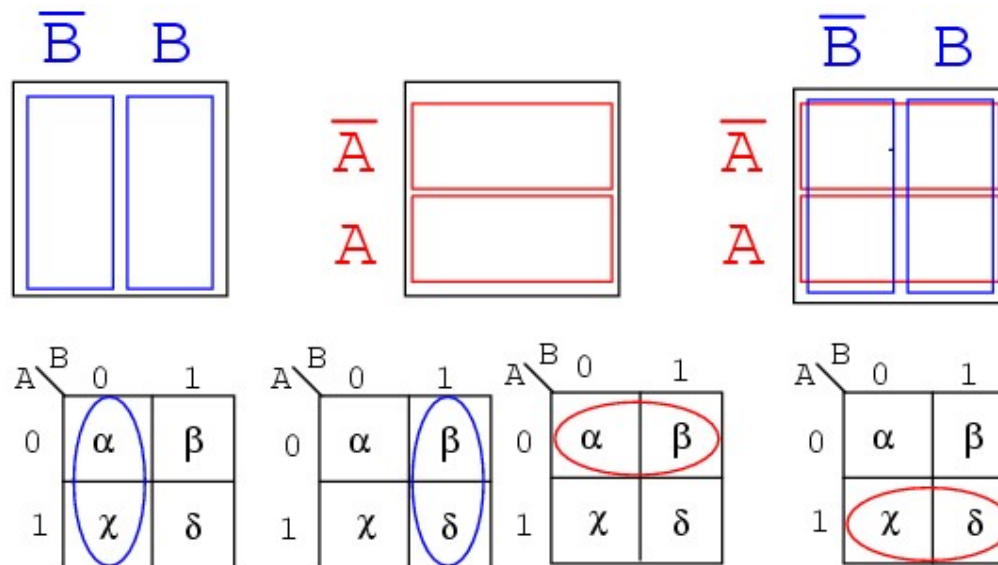
The output will remain as 1 even when S changes to 0.

The alarm can only be stopped if the reset switch is momentarily opened to produce a 5 V input to R

# Karnaugh Map

- Special form of a truth table which enables easier pattern recognition
- Pictorial method of simplifying Boolean expressions
- Good for circuit designs with up to 4 variables

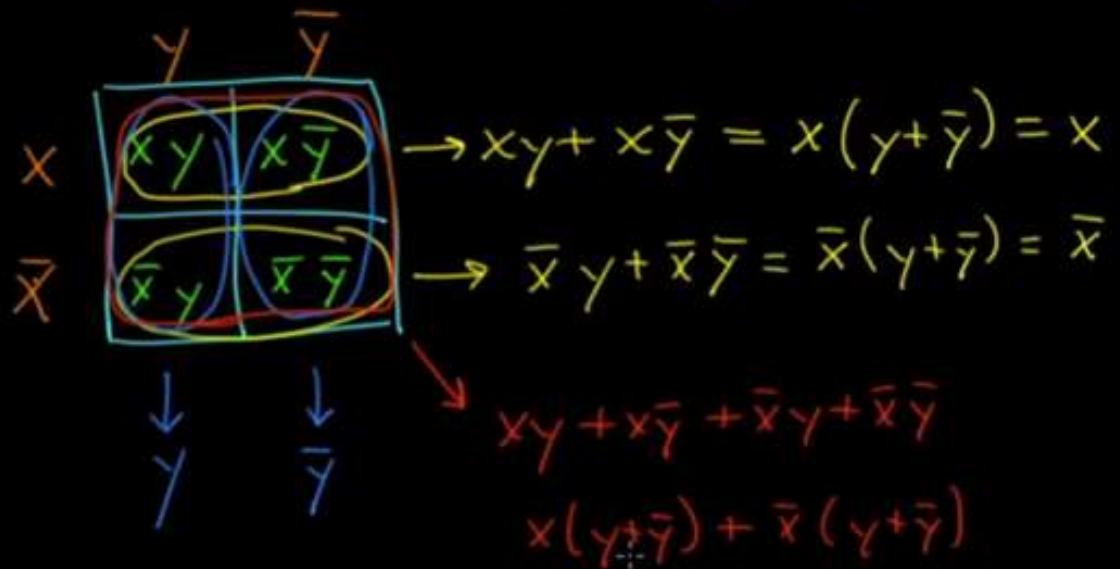
A	B	Output
0	0	$\alpha$
0	1	$\beta$
1	0	$\chi$
1	1	$\delta$

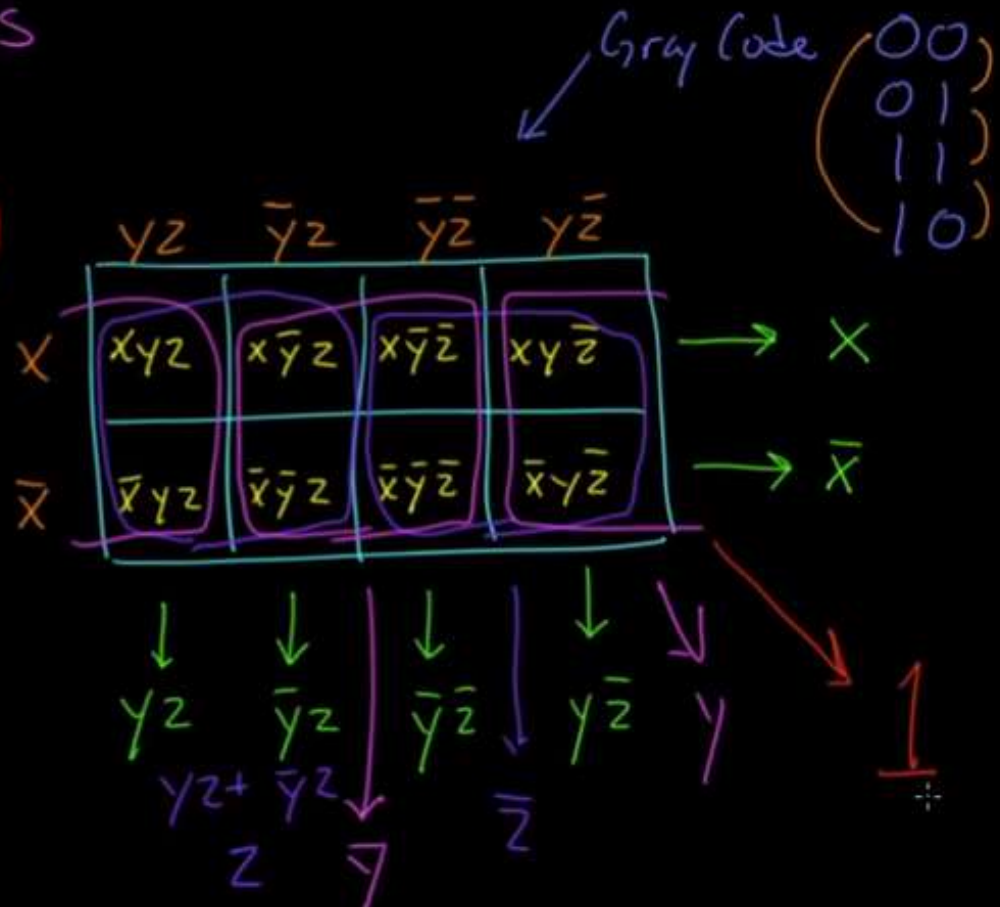
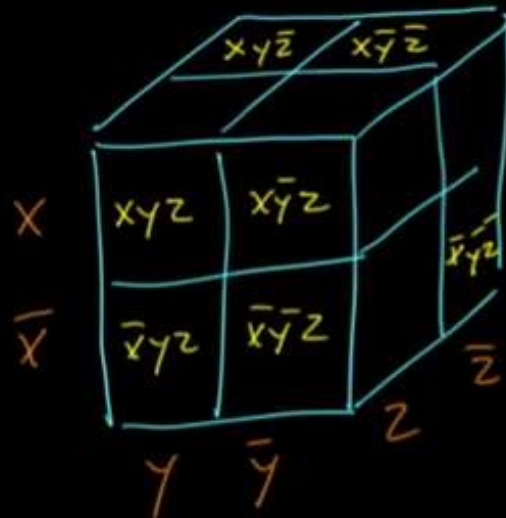
# Karnaugh Maps

$\cdot$  AND  $\cap$   
 $+$  OR  $\cup$

$xy$   
 $x\bar{y}$   
 $\downarrow$   
 $\bar{x}y$   
 $\bar{x}\bar{y}$



# Karnaugh Maps

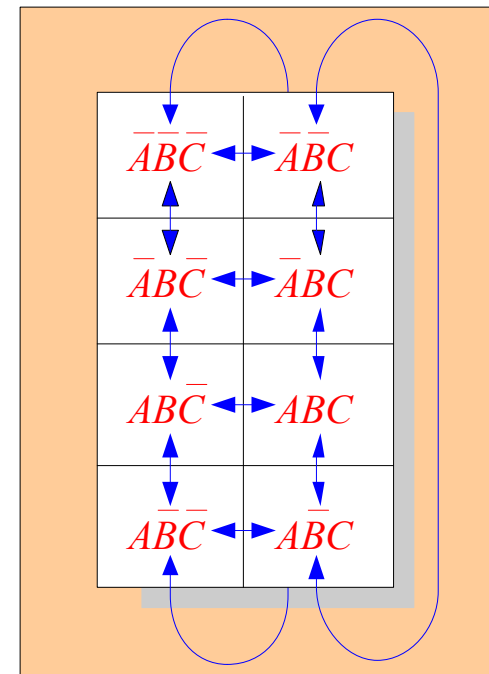


## Karnaugh maps

The Karnaugh map (K-map) is a tool for simplifying expressions with 3 or 4 variables. For 3 variables, 8 cells are required ( $2^3$ ).

The map shown is for three variables labeled  $A$ ,  $B$ , and  $C$ . Each cell represents one possible product term.

Each cell differs from an adjacent cell by only one variable.





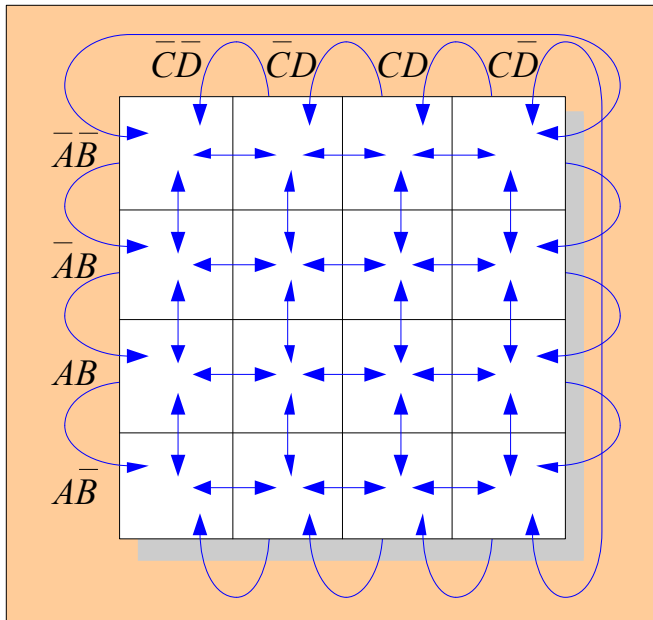
# Karnaugh Map Procedure

---

1. If you're starting with a Boolean expression that is not in SOP form, convert it to SOP form.
  2. Set up the K-map, labeling its rows and columns.
  3. Place 1s in the appropriate squares.
  4. Group adjacent 1s in groups of 8, 4, 2, or 1. You want to **maximize the size of the groups** and **minimize the number of groups**. Follow this order:
    - a. Circle any octet.
    - b. Circle any quad that contains one or more 1s that haven't already been circled, using the minimum number of circles.
    - c. Circle any pair that contains one or more 1s that haven't already been circled, using the minimum number of circles.
    - d. Circle any isolated 1s that haven't already been circled.
  5. Read off the term for each group by including only those complemented or uncomplemented variables that do not change throughout the group.
  6. Form the OR sum of the terms generated in Step 5.
-

## Karnaugh maps

A 4-variable map has an adjacent cell on each of its four boundaries as shown.



Each cell is different only by one variable from an adjacent cell.

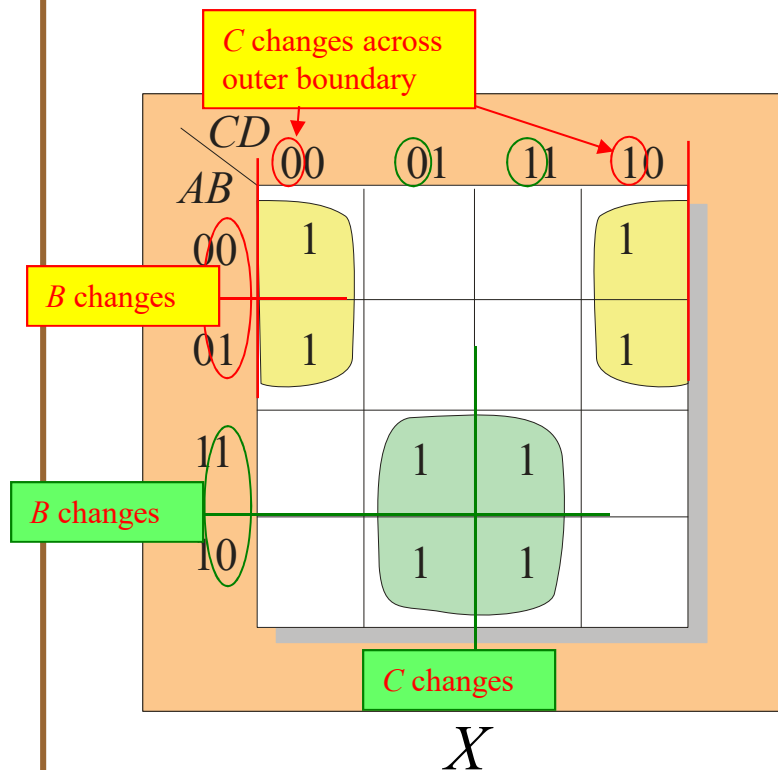
Grouping follows the rules given in the text.

The following slide shows an example of reading a four variable map using binary numbers for the variables...



## Karnaugh maps

**Example** Group the 1's on the map and read the minimum logic.



## Solution

1. Group the 1's into two separate groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The upper (yellow) group is read as  $\bar{A}\bar{D}$ .
4. The lower (green) group is read as  $AD$ .

$$X = \bar{A}\bar{D} + AD$$



# Karnaugh Maps

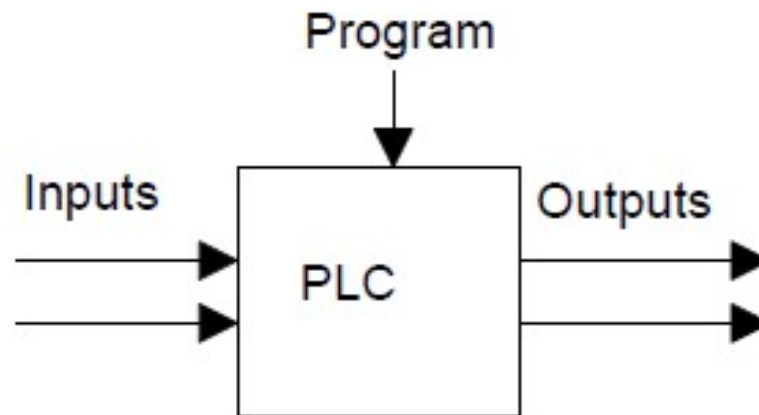
---

## Summary of Grouping Rules

- A group must only contain 1s, no 0s
- A group can only be horizontal or vertical, not diagonal
- A group must contain  $2^n$  1s (1, 2, 4, 8, etc.)
- Each group should be as large as possible
- Groups may overlap
- Groups may wrap around a table
- Every 1 must be in at least one group
- There should be as few groups as possible

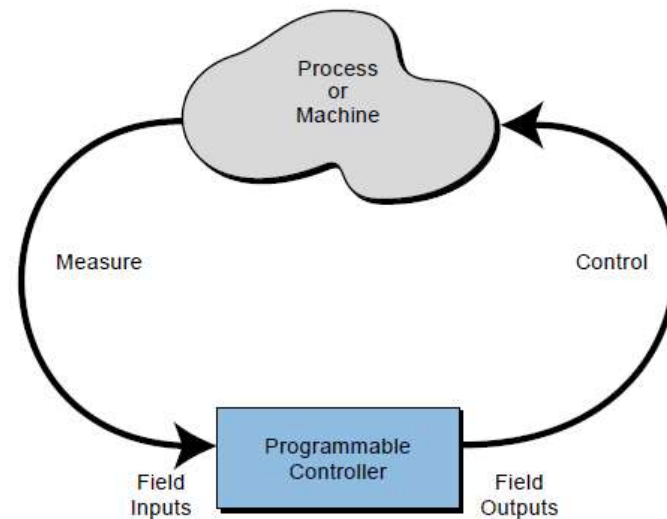
# Programmable logic controller

---

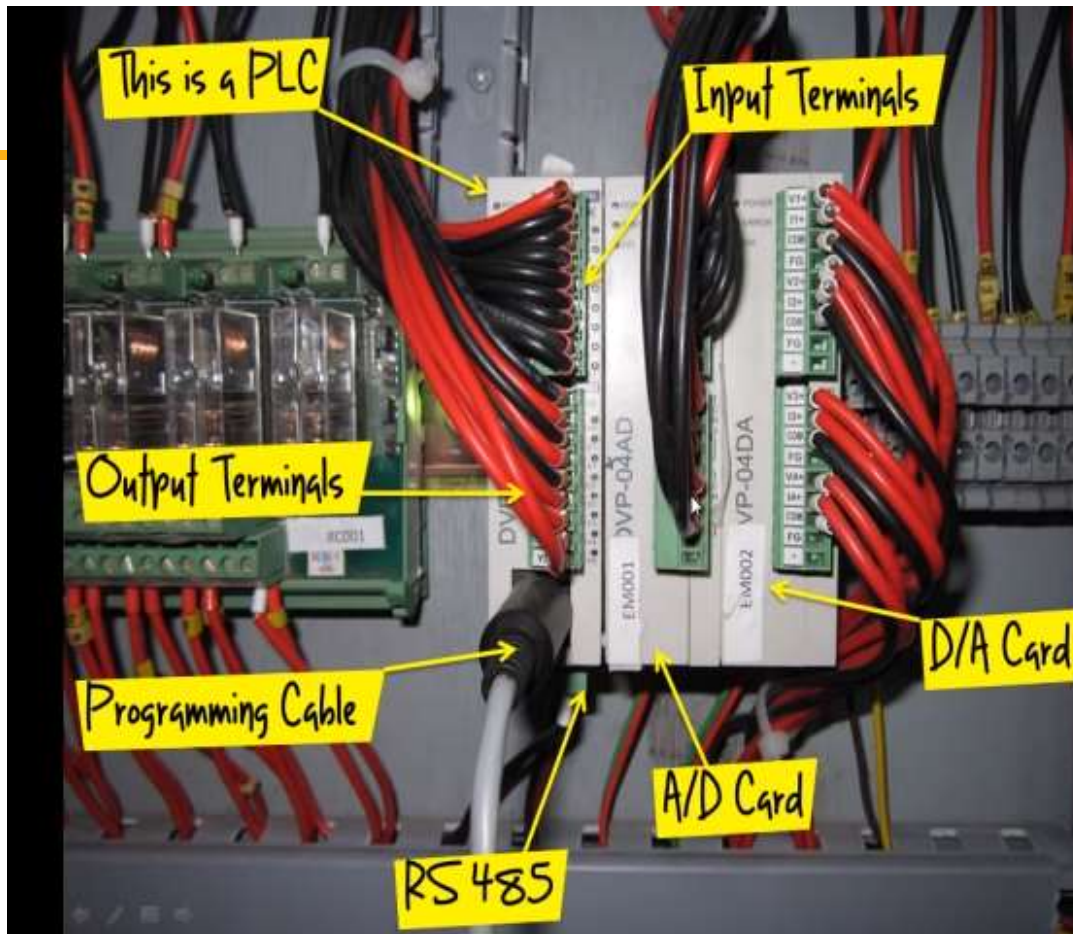


- 1 Rugged and designed to withstand vibrations, temperature, humidity and noise.
- 2 Have interfacing for inputs and outputs already inside the controller
- 3 Are easily programmed and have an easily understood programming language which is primarily concerned with logic and switching operations.

# Introduction



- PLCs are members of the computer family, using integrated circuits instead of electromechanical devices to implement control functions.
- They are capable of storing instructions, such as sequencing, timing, counting, arithmetic, data manipulation, and communication, to control industrial machines and processes. Figure illustrates a conceptual diagram of a PLC application.



Eg. - Control of machinery on factory assembly lines, as well as heating, ventilation, air-conditioning, plastic injection moulding machines, commercial washing machines...



# Why PLC?

---

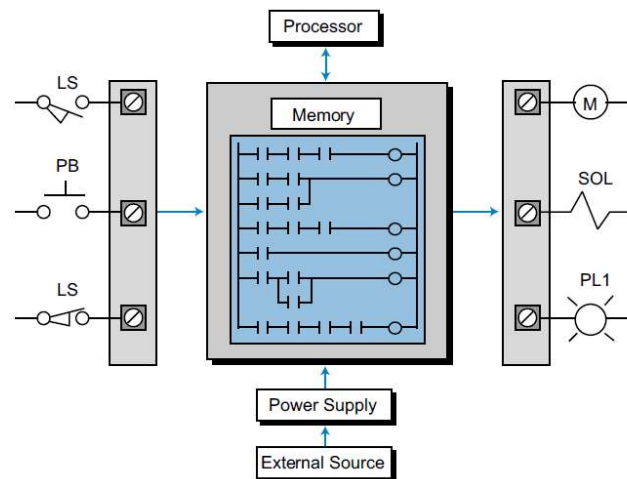
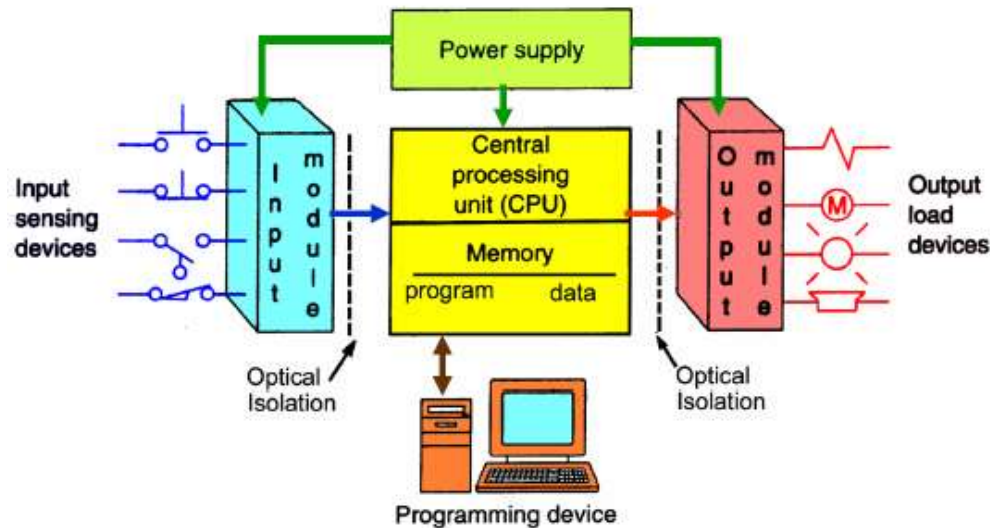
- Hard wired panels / relay logic are very time consuming to implement due to wiring and debugging related issues.
  - In need of another controller which:
    - Is faster and operates in real time
    - Withstands vibrations, temperatures, humidity, noise
    - Has inbuilt interfacing for inputs and outputs
    - Is flexible and adaptable (easy to program/re-program), easy to troubleshoot and maintain
    - Consumes less space, requires less wiring, comparatively cheaper
-

# Components in PLC



## Components

- Input Module
- Output Module
- CPU
- Memory
- Power Supply
- Programming Device
- Communication Channel



Programming Software – e.g. Automation studio



# Sourcing and Sinking of PLC

Field devices connected to the positive side of the field power supply are classified as sourcing field devices.

Conversely, field devices connected to the negative side or DC common of the field power supply are sinking field devices

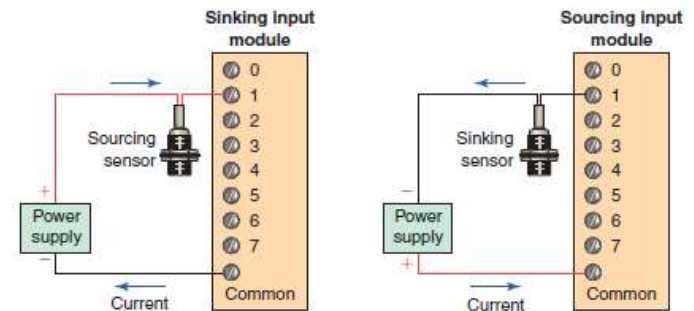


Figure 2-20 Sinking and sourcing inputs.

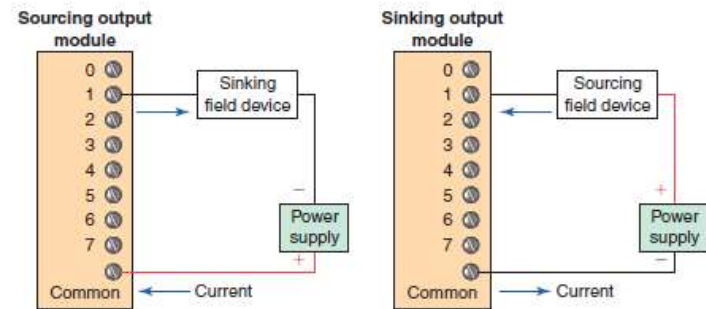


Figure 2-21 Sinking and sourcing outputs.

It is important to know the type of input or output concerned so that it can be correctly connected to the PLC. Thus, sensors with sourcing outputs should be connected to sinking PLC inputs and sensors with sinking outputs should be connected to sourcing PLC inputs. The interface with the PLC will not function and damage may occur if this guideline is not followed.





# Input and Output module & I/O address format

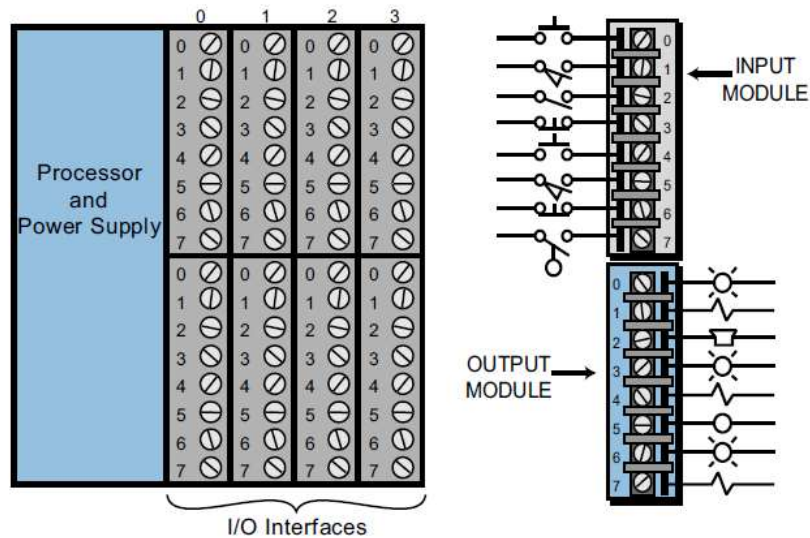


Figure 1-8. Input/output interface.

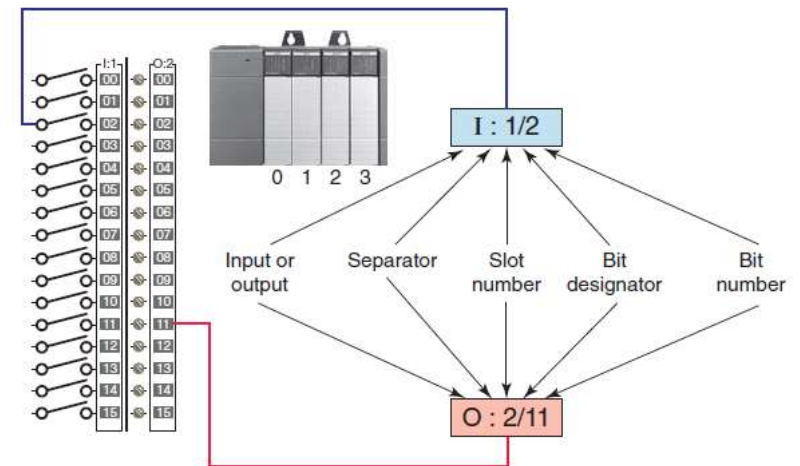


Figure 5-2 I/O address format for the SLC family of PLCs.  
Source: Image Used with Permission of Rockwell Automation, Inc.

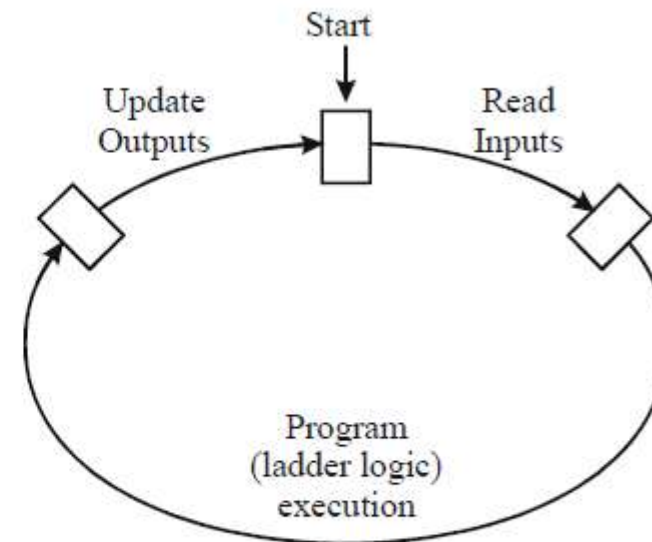
# PLC Processor Scan

## ■ Order of PLC Processor Scan

- Read Physical Inputs
- Scan ladder logic program
- Write the physical outputs

## ■ Scan Time

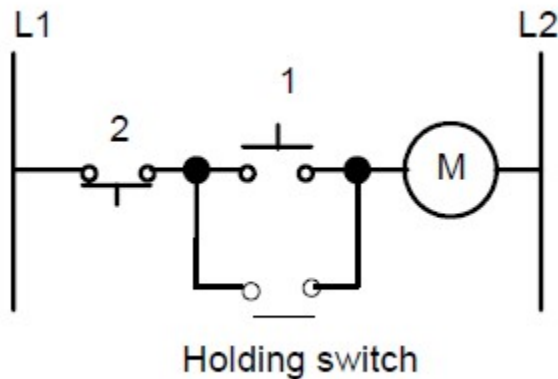
- Time to complete above cycle
- Order of 1-200 milliseconds



The operation of the PLC can be considered in two modes:

1. The I/O scan mode
2. The Execution mode

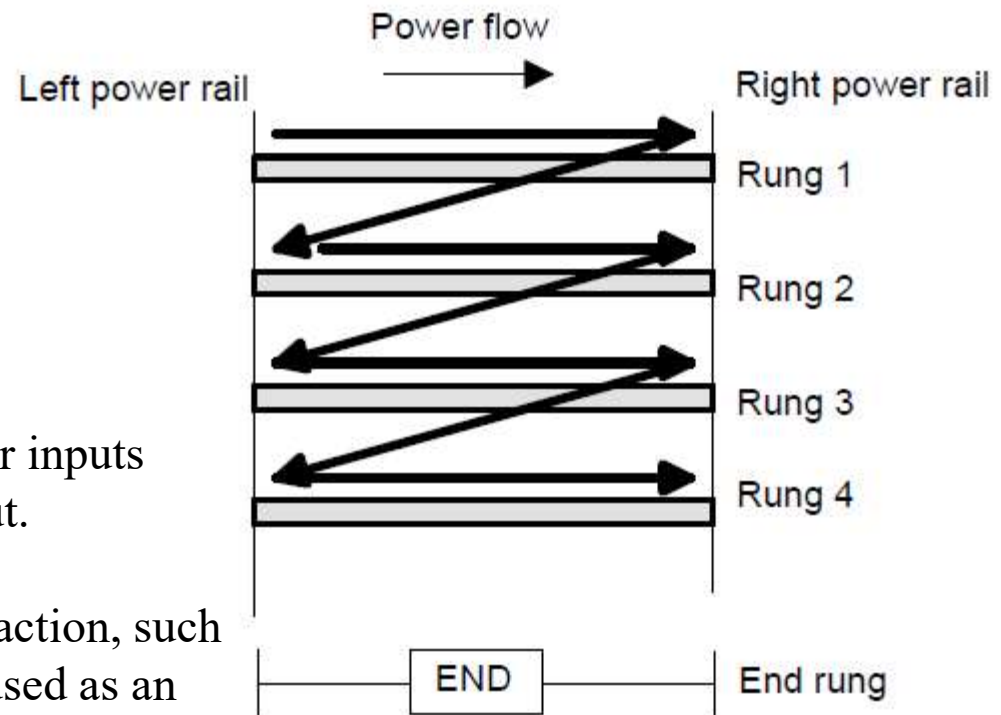
# Ladder Programming



Each rung must start with an input or inputs and must end with at least one output.

The term input is used for a control action, such as closing the contacts of a switch, used as an input to the PLC.

The term output is used for a device connected to the output of a PLC, e.g. a motor.





# Ladder Programming

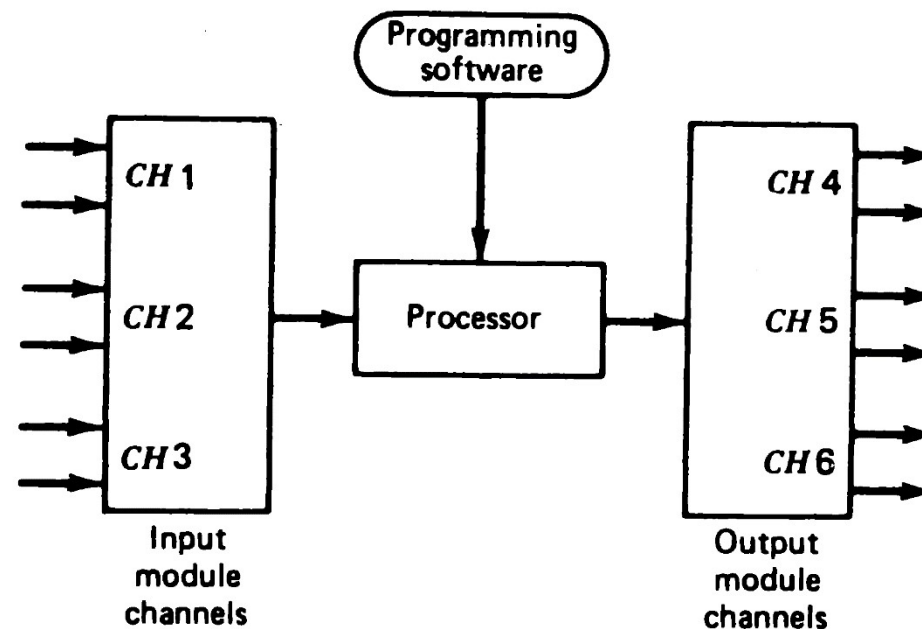
---

- The sequence followed by a PLC when carrying out a program can be summarised as follows.
- Scan the inputs associated with one rung of the ladder program.
- Solve the logic operation involving those inputs.
- Set/reset the outputs for that rung.
- Move on to the next rung and repeat operations 1, 2, 3.
- Move on to the next rung and repeat operations 1, 2, 3.
- Move on to the next rung and repeat operations 1, 2, 3.
- And so on until the end of the program with each rung of the ladder program scanned in turn.
- The PLC then goes back to the beginning of the program and starts again.

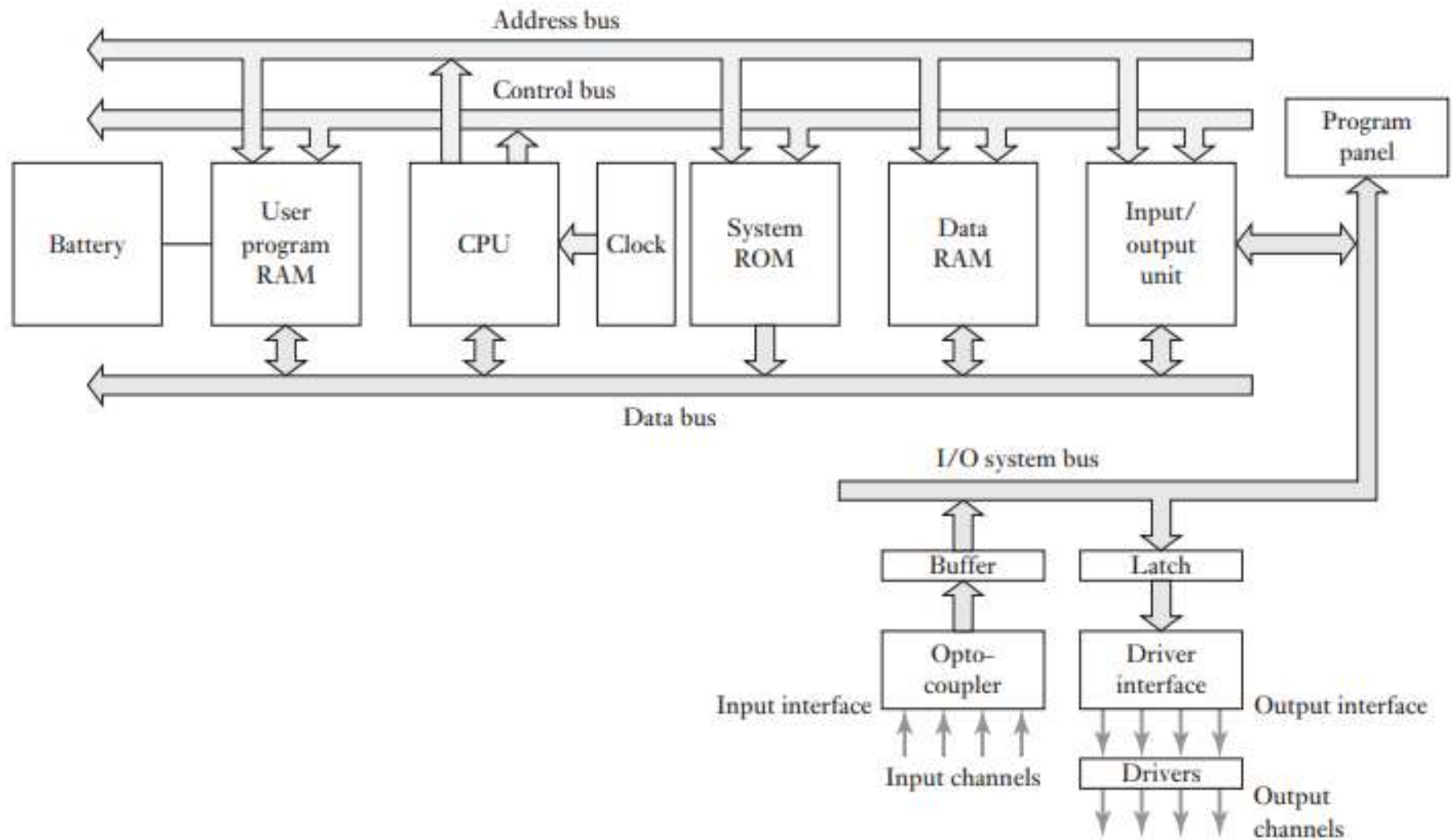


# Architecture of a PLC

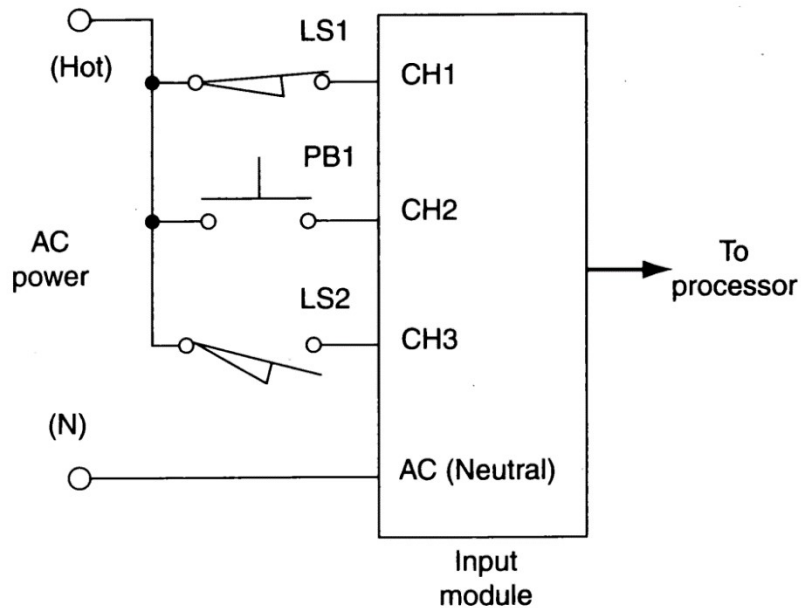
A Programmable controller can be studied by considering the basic elements shown in Figure : the processor, I/O modules and the software.



# Architecture of a PLC



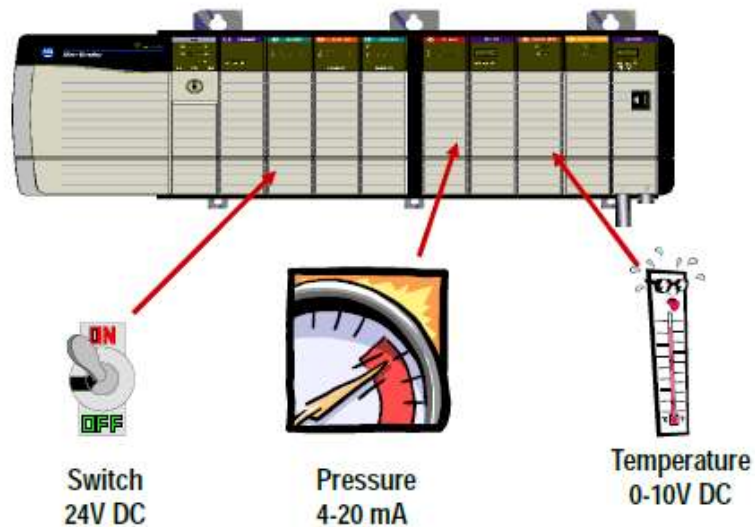
# Input Module



- Figure shows the typical wiring to a PLC input module.
- The input module examines the state of the physical switches and other input devices and puts their state into a form suitable for the processor.
- It is able to accommodate a number of inputs called channels.
- If the switch is closed, the input will be 24 V DC, and if open, 0 V DC. The input module converts this into the 1 or 0 state needed by the processor.

# Input Module

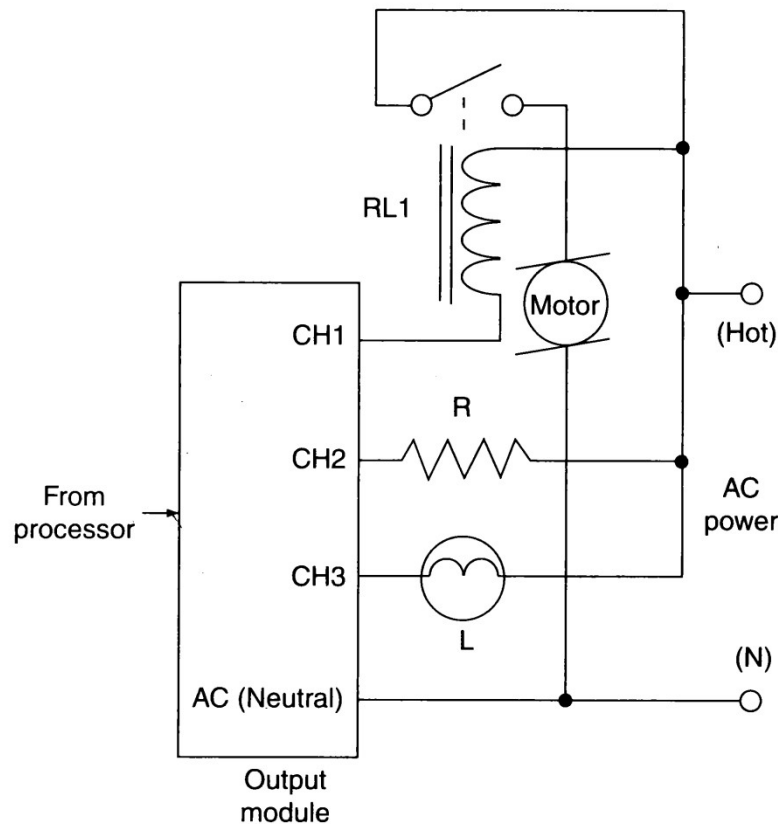
- **Input Module:** Convert real world voltage and currents to signals the PLC can understand. Since there are different types of input devices, there is a wide variety of input modules available, including both digital and analog modules.
- **Discrete / Digital Module:** Use 0 or 1 to depict state
- **Analog Module:** Use numbers to depict state e.g. 30 degree



- Limit switches – open or closed,
- Comparators – high or low,
- Push buttons – depressed or not depressed



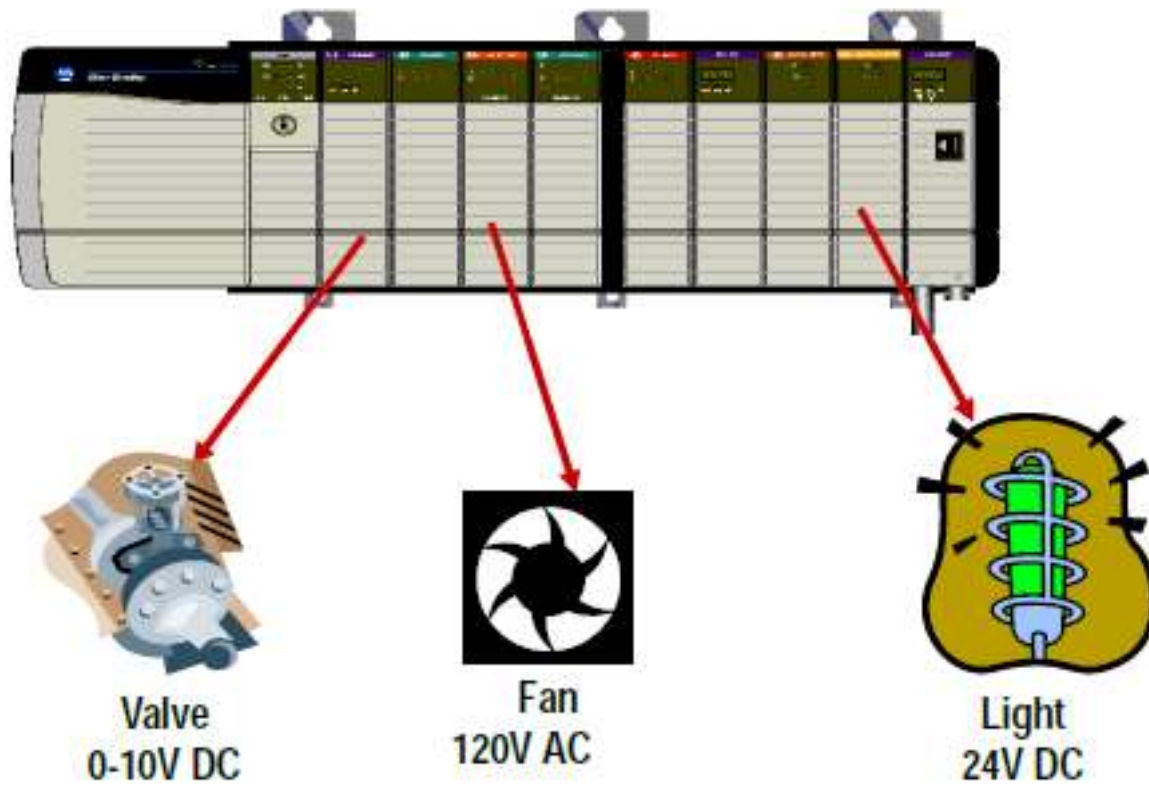
# Output Module



- Lights: On / OFF,
- Motors : Rotating or not rotating,
- Solenoids : Engaged or not engaged

- Figure shows the typical wiring to the PLC output module.
- The Output Module supplies ac power to the external devices such as motors, lights, solenoids, etc.
- Internally, the output module accepts a 0 or 1 from the processor, and uses it to turn ON or OFF a device.
- An output module can have one or several channels per unit. Each channel is usually provided with an indicator light to show whether the particular channel is being driven ON/OFF.

# Output Module



Valve  
0-10V DC

Fan  
120V AC

Light  
24V DC

# I/O Devices

## Input Devices



a) Momentary action push button



b) Physical limit



c) Pressure limit



d) Temperature limit



e) Level limit

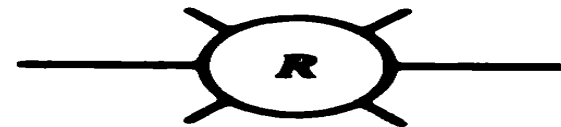
## Output Devices



(a) Motor



(b) Solenoid



(c) Light (red)



# Processor

---

- The processor is a computer that executes a program to perform the operations specified in a ladder diagram.
  - It performs arithmetic and logic operations on input variable data and determines the proper state of the output variables.
-

# CPU & Memory



- **CPU:** The brain of PLC is the central processing unit (CPU).
- It executes the various logic and sequencing functions by operating on the PLC inputs (sensor information) to determine the appropriate output signals for the actuator.
- The processor is microprocessor very similar in its construction to those used in personal computers and other data-processing equipment.
- **Memory:** Tied to the CPU is the PLC memory, which contains the program of logic, sequencing, and other input/output operations.
- The memory for a programmable logic controller is specified in the same way as for a computer, and may range from 1k to over 48 k of storage capacity.
- Memory types are ROM, RAM, EPROM



# Power Supply

- **Power Supply:** Power supplies come in various forms:
  - Power supply modules that fit into one of the slots in a chassis
  - External power supplies that mount to the outside of a chassis
  - Stand alone power supplies that connect to the PLC or I/O through a power cable
  - Embedded power supplies that come as part of the PLC block.





# Programmable Device

---

- **Programming Device:** The PLC is programmed by means of a programming device.
  - The programming device is usually detachable from the PLC cabinet so that it can be shared between different controllers.
  - Different PLC manufactures provide different devices:
    - Simple teach pendant-type devices, similar to those used in robotics
    - PLC programming keyboards and CRT displays.
-



# Communication Channel

---

**Communication Channel:** The CPU uses the:

- the **data bus** for sending data between the constituent elements,
  - the **address bus** to send the addresses of locations for accessing stored data
  - the **control bus** for signals relating to internal control actions
  - the **system bus** is used for communications between the input/output ports and the input/output unit.
-





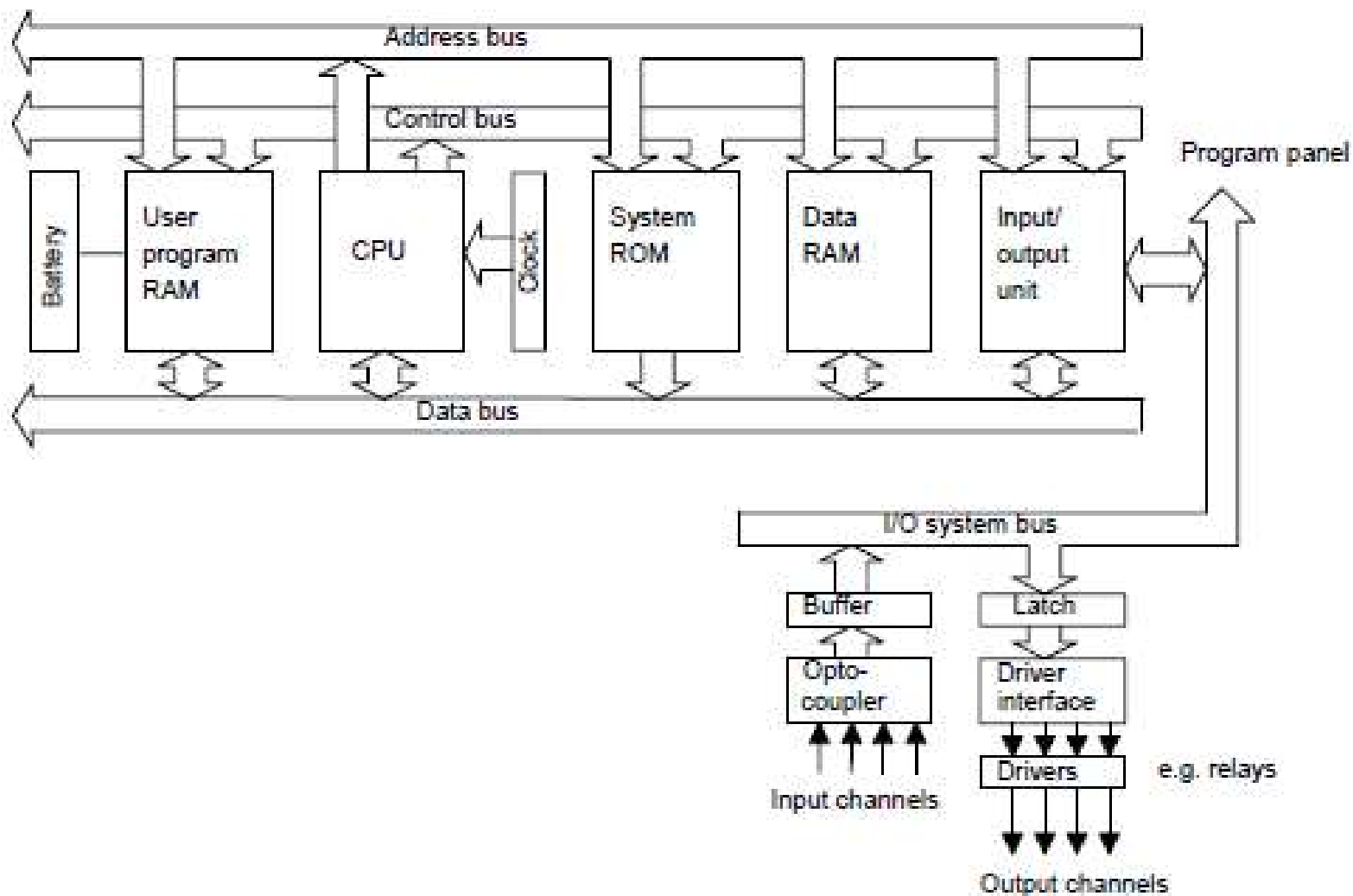
# Communication Channel

---

- The buses are the paths used for communication within the PLC.
  - The information is transmitted in binary form, i.e. as a group of bits with a bit being a binary digit of 1 or 0, i.e. on/off states.
  - The term word is used for the group of bits constituting some information. Thus an 8-bit word might be the binary number 00100110.
  - Each of the bits is communicated simultaneously along its own parallel wire.
-



# Communication Channel





# Selection Criterion for PLC

---

- Application Requirement
  - Input / Output Requirement
  - Memory Requirement
  - CPU Requirement
  - Software Requirement & Operator Interface
  - Communication Requirement
  - Environmental Requirements
-

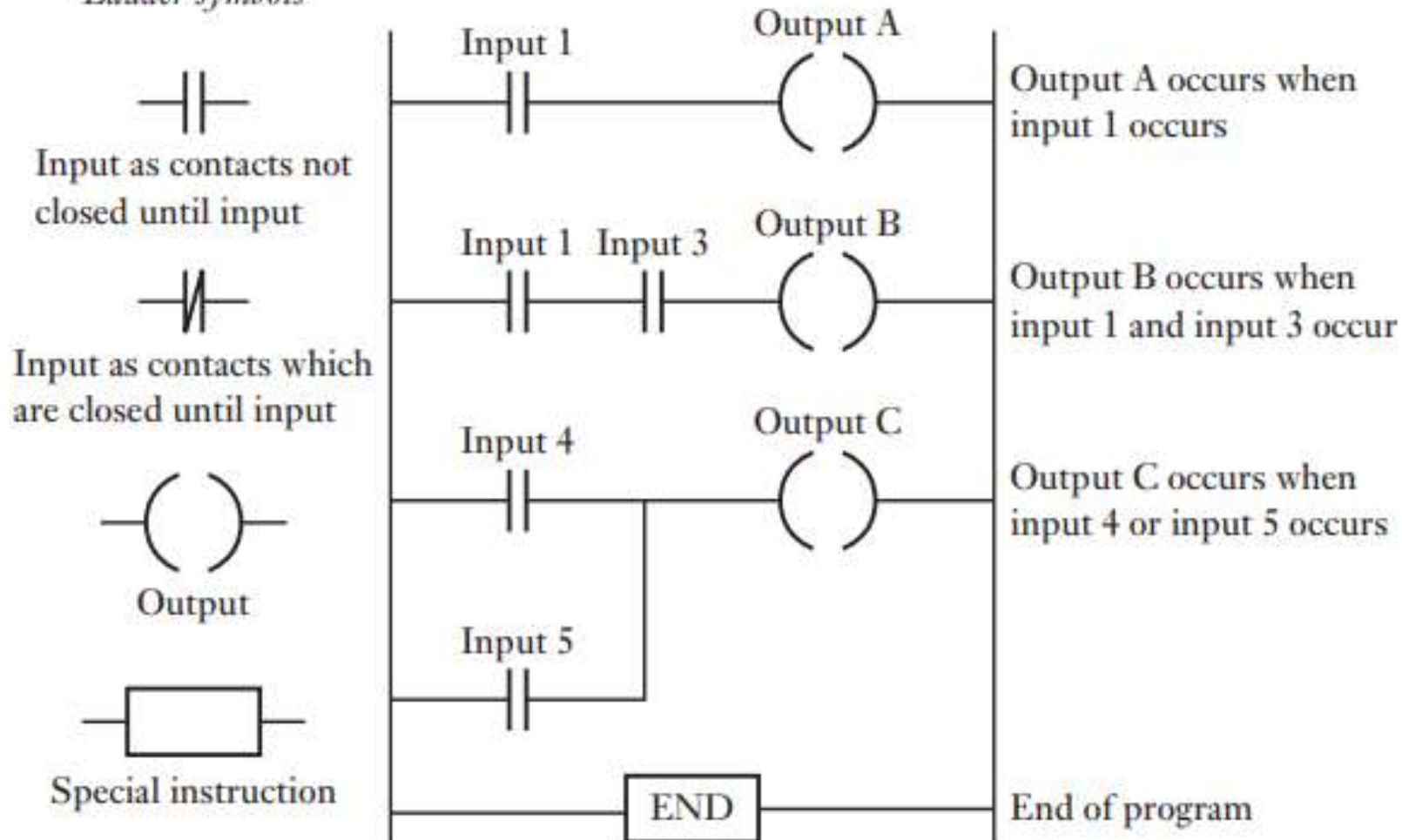
# PLC Addressing

Function	Address
Input channels	00 to 07
Output channels	08 to 15
Internal relays	16 to 31
Timers and counters	32 to 39

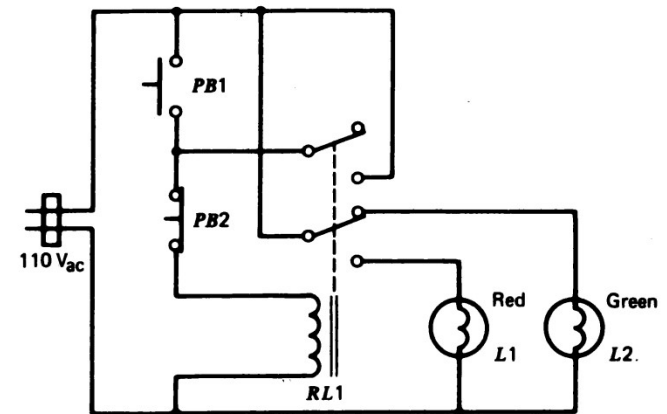
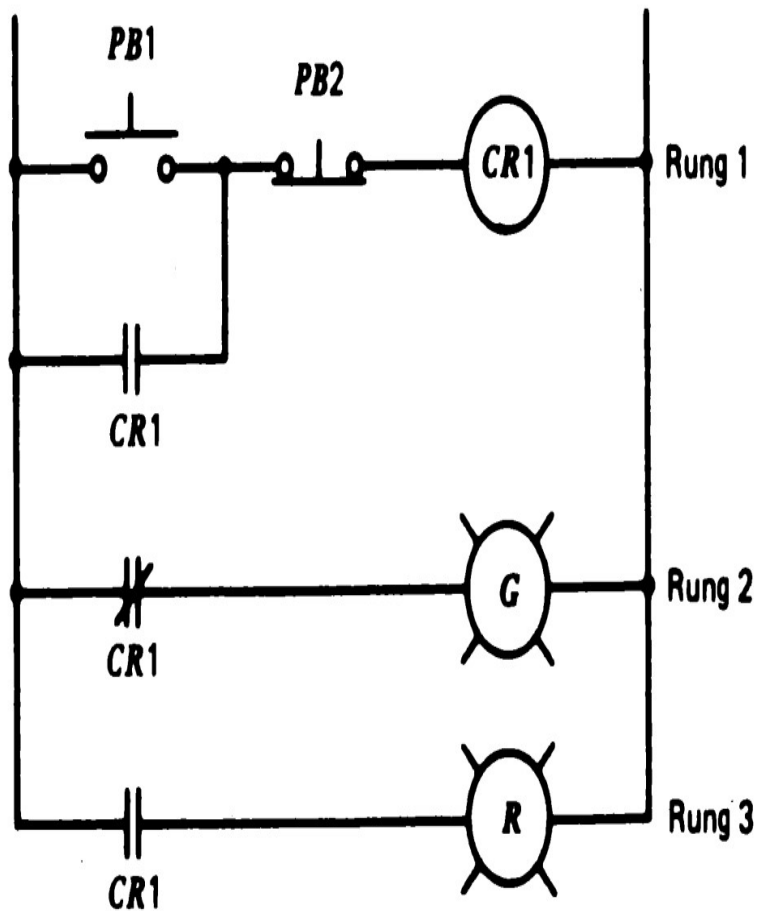
- To identify the I/O devices, the PLC uses the device address or channel.
- Its address designation depends on the type of programmable controller.
- Table shows a typical address designation for different I/O devices.

# Representation of PLC circuit

## Ladder symbols



# Use of Relay Controllers and Ladder Diagram



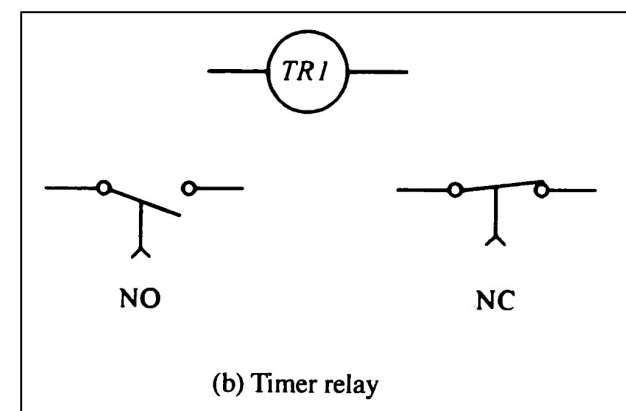
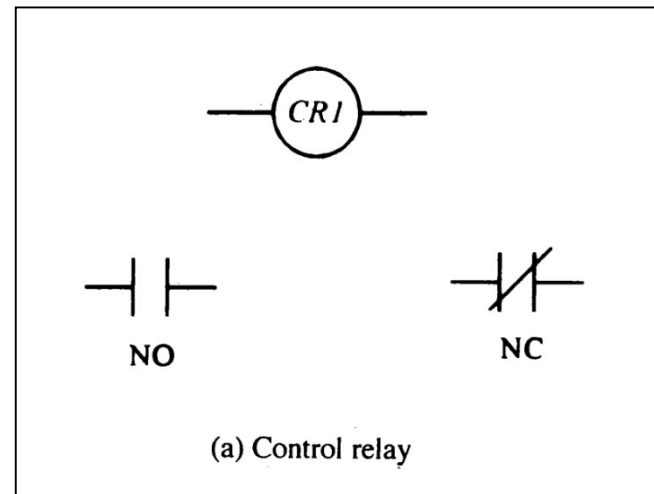
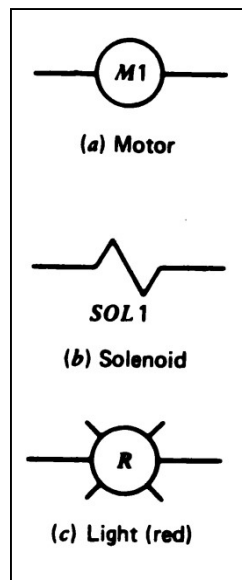
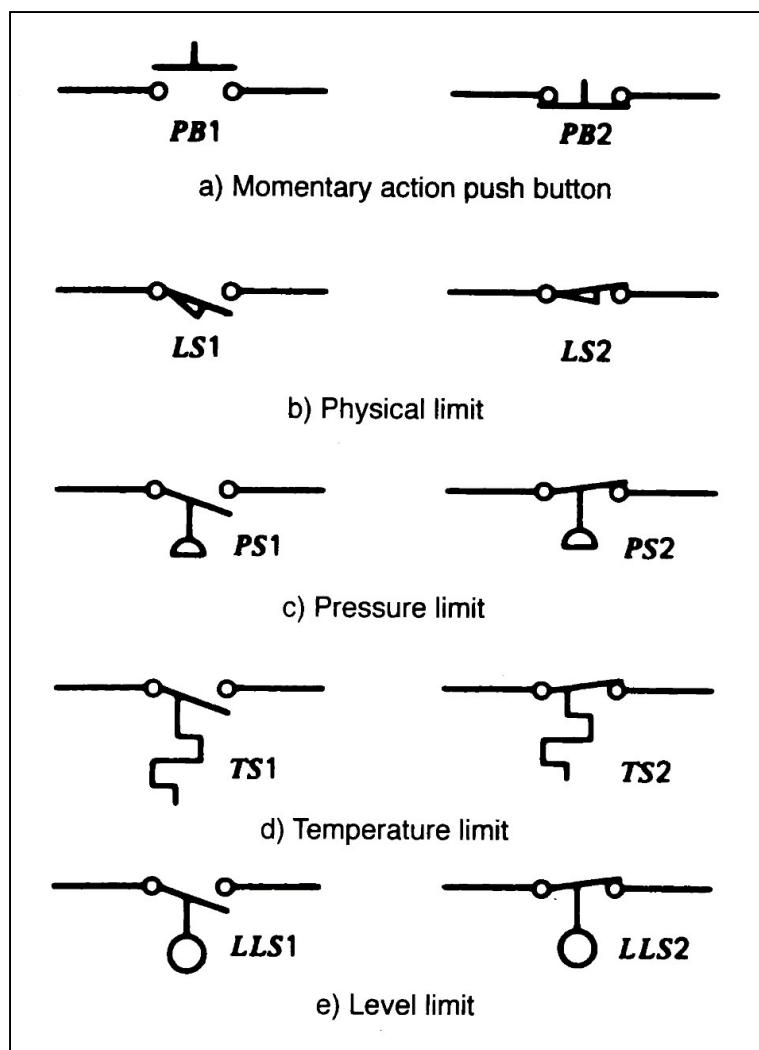
Use of momentary push-button switches and a relay to implement a latch

# Relay Sequencers



- Before the PLC, control, sequencing, and safety interlock logic for manufacturing automobiles was mainly composed of relays, cam timers and dedicated closed-loop controllers.
- One way to provide a discrete state controller is to use physical relays to put together a circuit that satisfies the requirements of the ladder diagram.
- When a program has been wired into the relays that make up the relay logic panel, it has been programmed to meet the ladder diagram.
- If the event sequence is to be changed, it is necessary to rewire all or part of the panel. It may even be necessary to add more relays to the system, or to use more relays than the previous program.

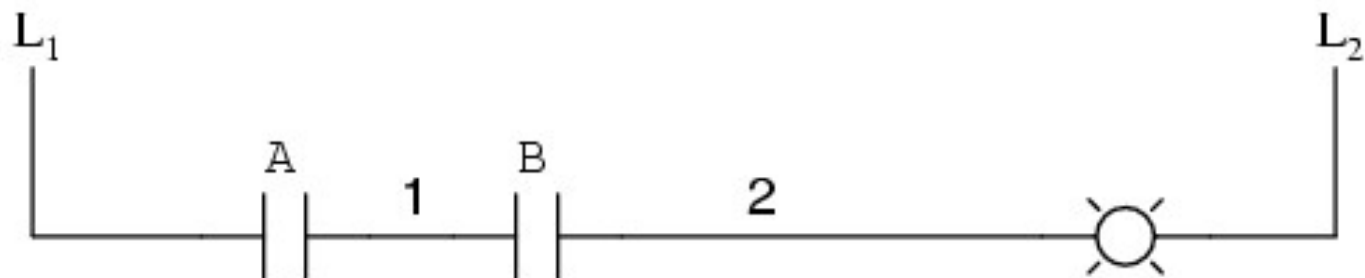
# Elements of Ladder Diagram



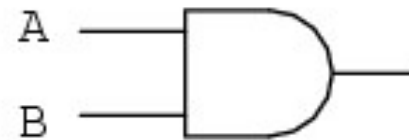




# Ladder Diagram: AND Logic



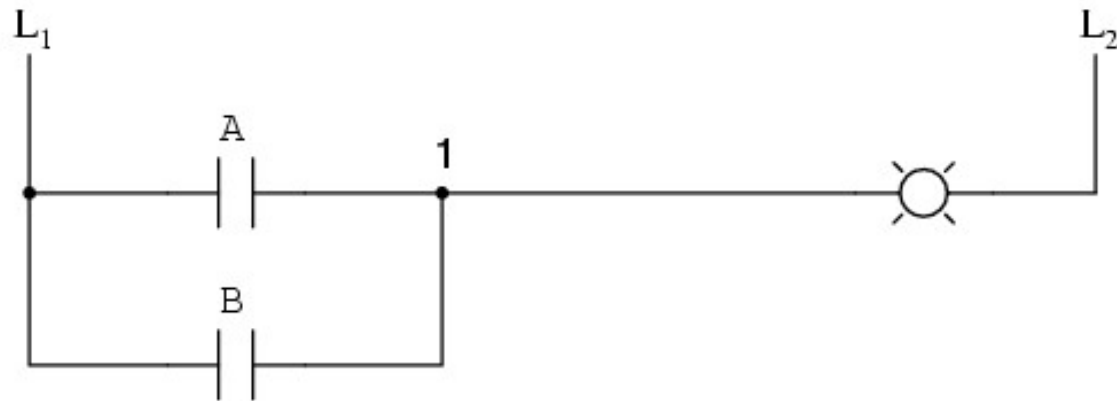
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



The lamp energizes only if contact A and contact B are simultaneously actuated.



# Ladder Diagram: OR Logic



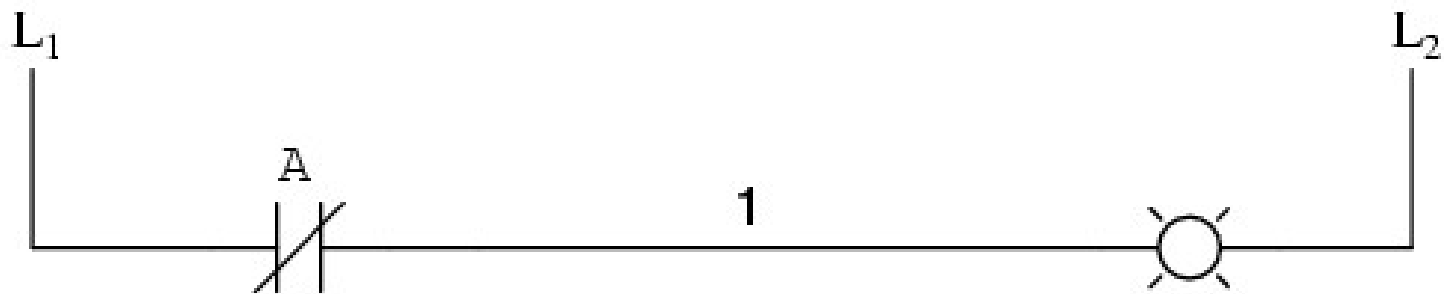
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



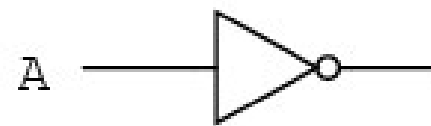
The lamp energizes if either of contact A or contact B are actuated.



# Ladder Diagram: NOT Logic

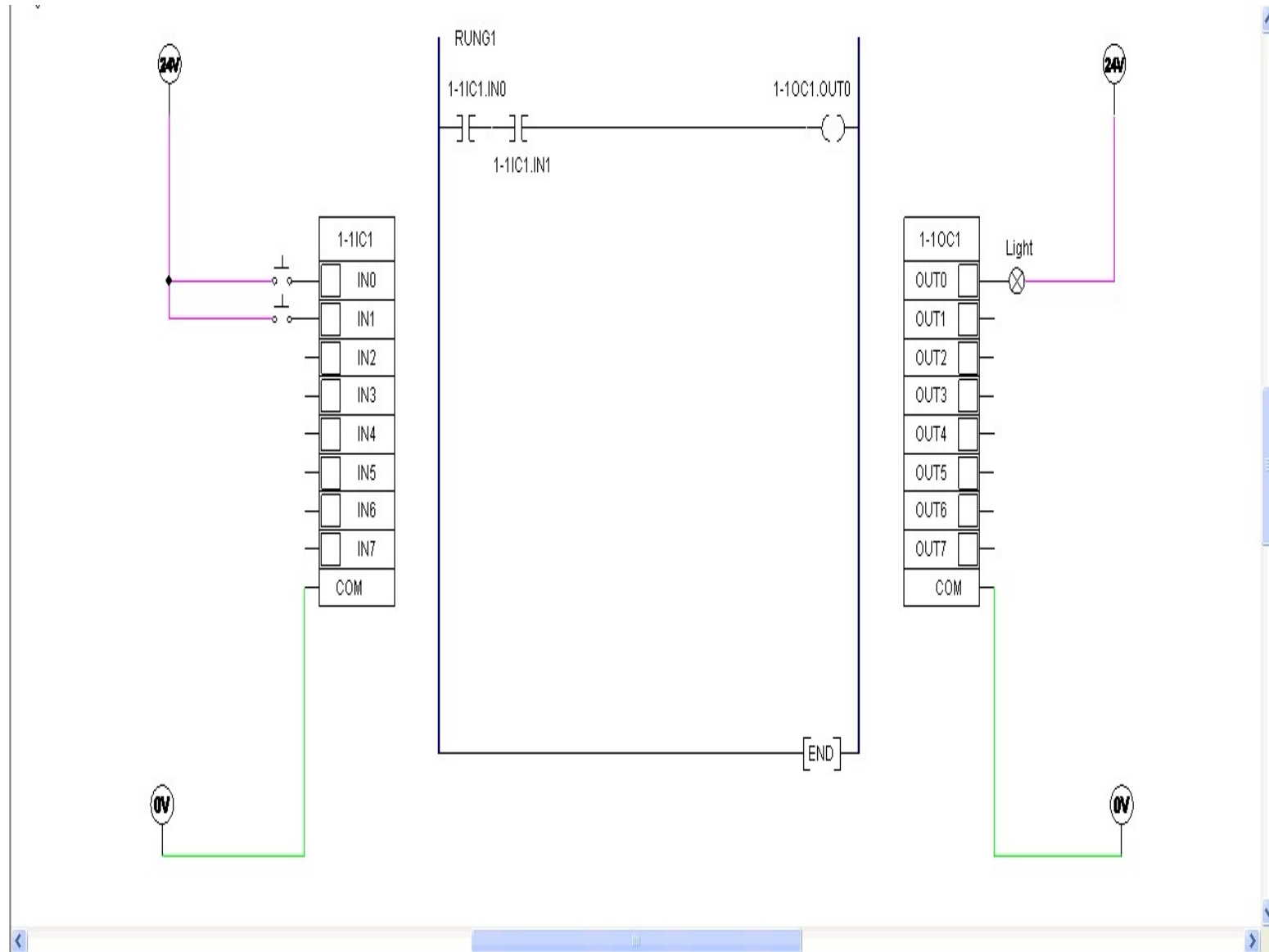


A	Output
0	1
1	0

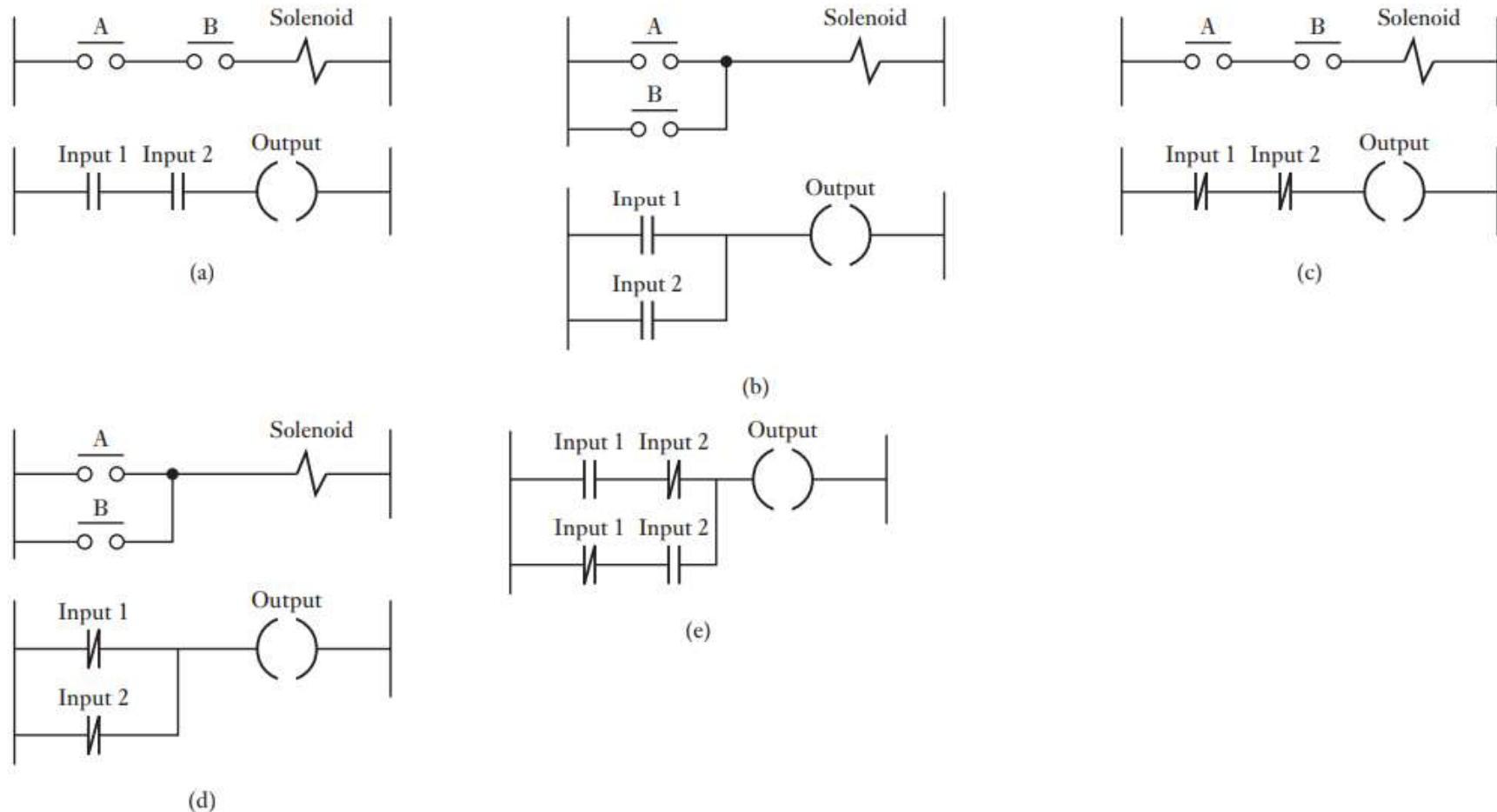


The lamp energizes if the contact is not actuated, and de-energizes when the contact is actuated.

# Ladder Logic Diagrams for AND Gate



# Logics in Ladder diagram form



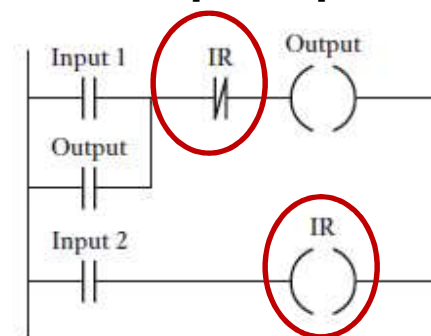
**Figure 14.8** (a) AND, (b) OR, (c) NOR, (d) NAND, (e) XOR.

# Internal Relays

The term internal, auxiliary relay or marker is used for what can be considered as internal relay in PLC.

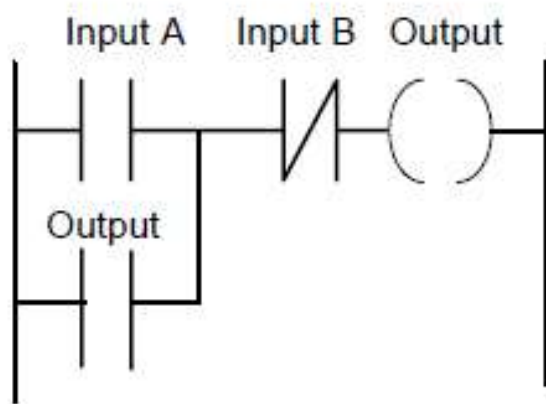
It behaves like relays with their associated contacts, but in reality are not actual relays but simulation by the software of the PLC.

- Internal can be very useful aids in the implementation of switching sequences.
- They are often used when there are programs with multiple input conditions.



**Figure 14.16** Resetting a latch.

# Latching circuits



*Latched circuit*

When the input A contacts close, there is an output.

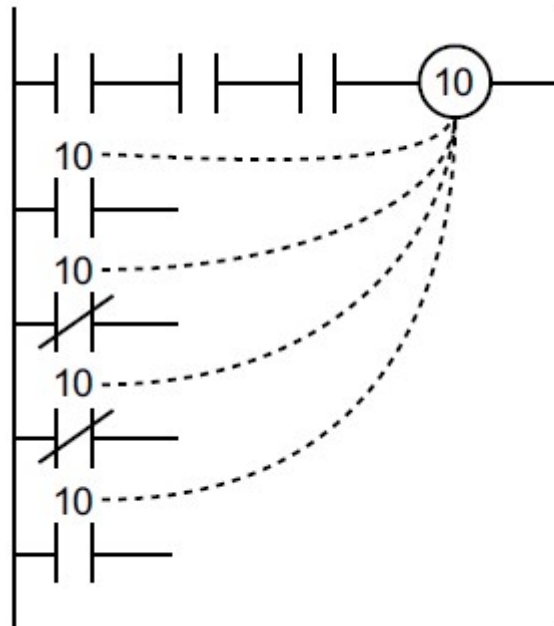
However, when there is an output, another set of contacts associated with the output closes.

These contacts form an OR logic gate system with the input contacts. Thus, even if the input A opens, the circuit will still maintain the output energised.

The only way to release the output is by operating the normally closed contact B.

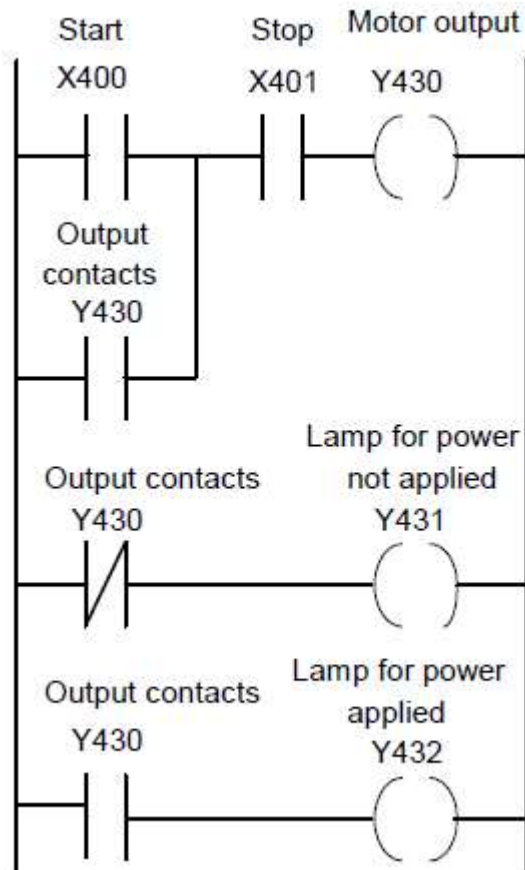


# Multiple contacts from one coil 10





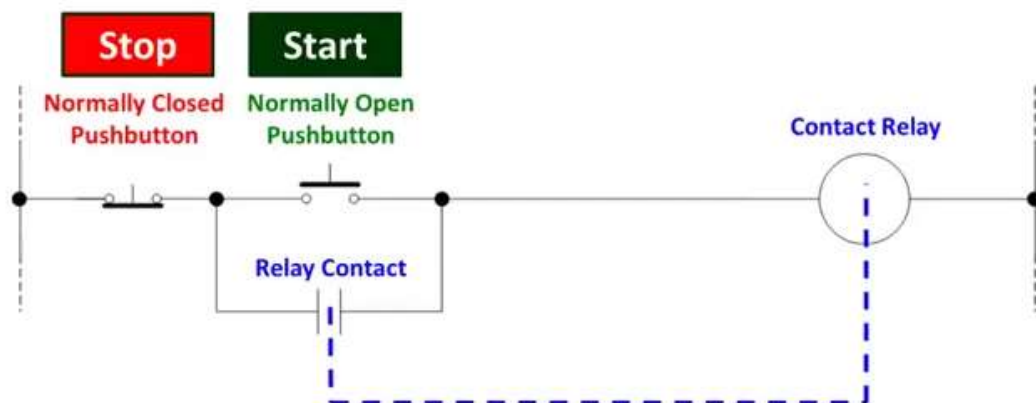
# Latching circuit Application



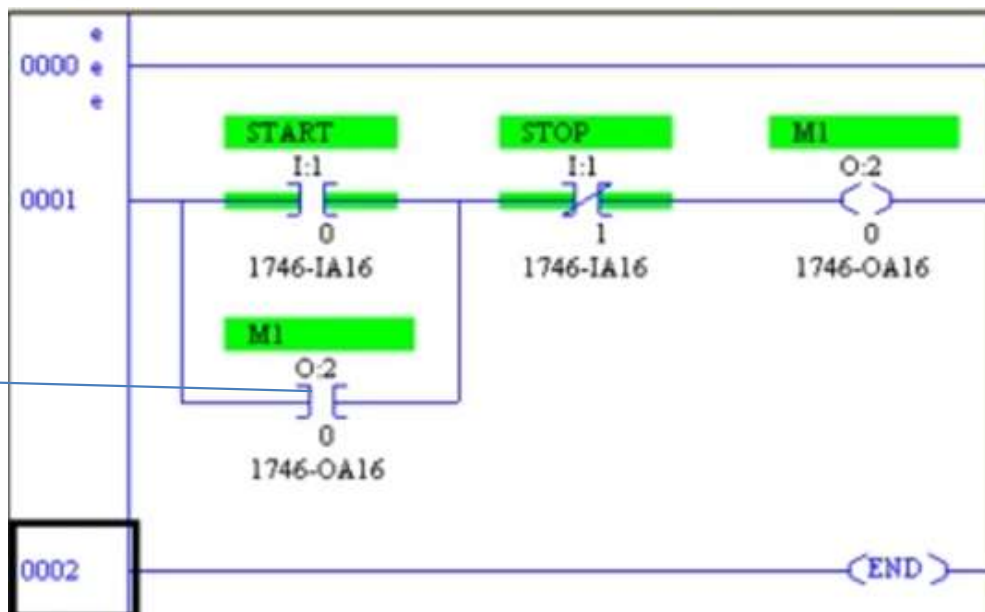
There are often situations where it is necessary to hold an output energised, even when the input ceases.

A simple example of such a situation is a motor which is started by pressing a push button switch.

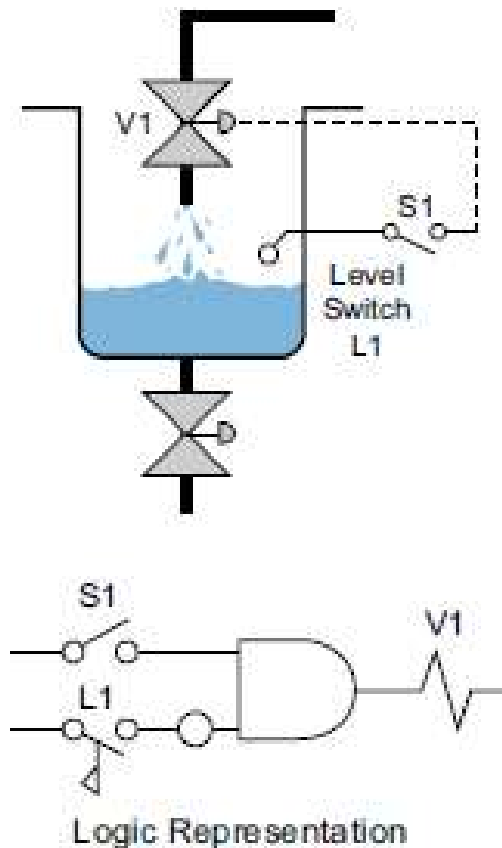
# Programmable logic controller



Internal  
Relays

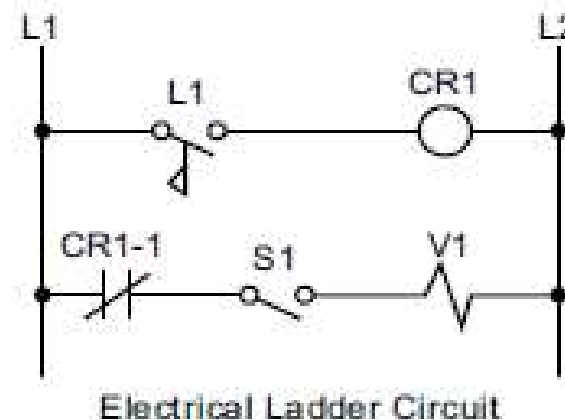


# Programmable logic controller

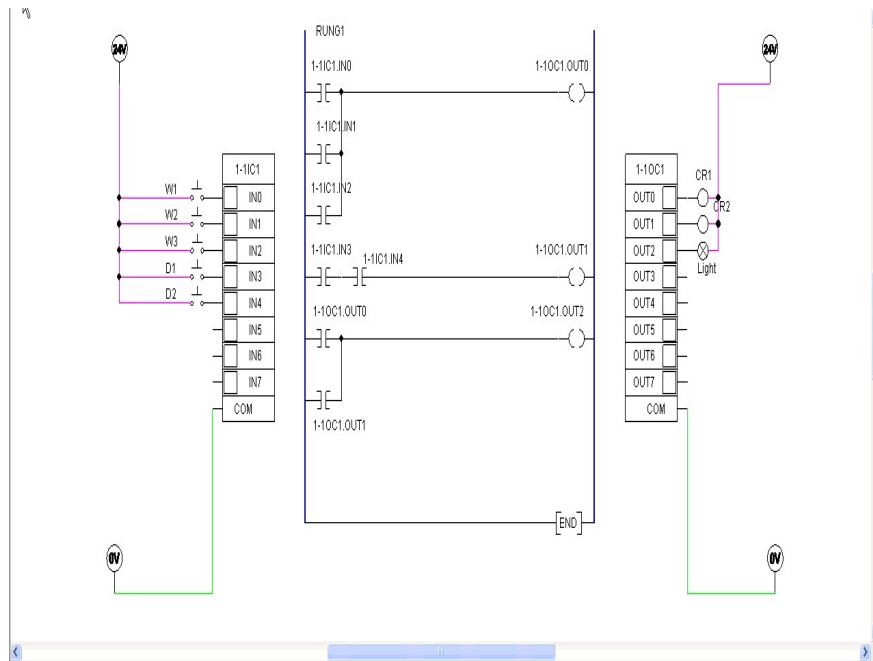


S1	L1 ( $\overline{L1}$ )		V1
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0

Truth Table







A,B,C as windows

D, E doors

A or B or C

D and E

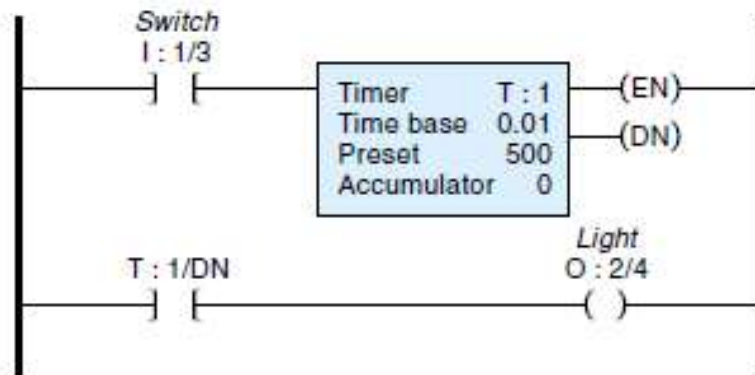
- A small house has 3 windows and 2 doors. Each window and door has a switch attached such that the contacts close when a door or window opens. Develop a PLC ladder that will turn ON a light if one or more windows are open, or if both doors are open.

# PLC Timer Function



- The Programmed timer function plays an important role in PLC applications to provide for needed delays in some manufacturing sequence and to specify the period of time that some operation is to last.
- While activated by a true path, the timer begins to accumulate time in form of “ticks”.
- Each “tick” is worth a certain amount of time.
- The timer is preloaded with a certain number of these ticks.
- When the accumulated time ticks equals the preload value, the timer itself becomes TRUE.
- The timer only counts while it has a TRUE input. If the input becomes FALSE and then TRUE again, the timer will reset to ZERO and start to count again.

# PLC Timer Function



**DN** A bit that goes TRUE when the timer is done—in other words, when the count gets to the preset value.

**T : 1** The timer address (actually the first of three addresses in RAM) where the first address holds the status bits EN, TT, and DN; the second address holds the preset value; and the third address holds the accumulator value.

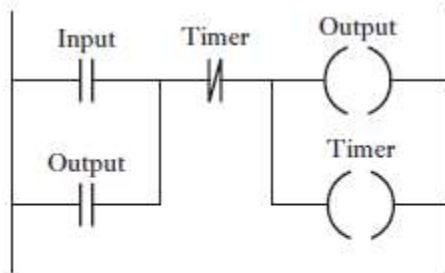
**Time base** The value of 0.01 means that each count corresponds to 0.01 seconds.

**Preset** The value of 500 means that the delay will last 500 counts, which in this case is 5 s ( $0.01\text{ s} \times 500 = 5\text{ s}$ ).

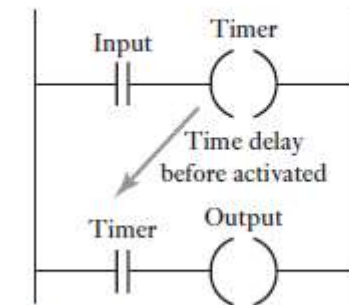
**Accumulator** Holds the value of the current count.

**EN** A bit that is TRUE as long as the timer rung is TRUE.

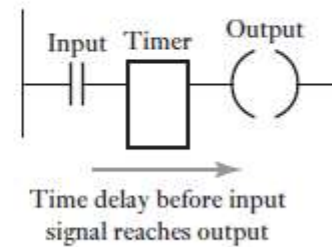
# TIMERS



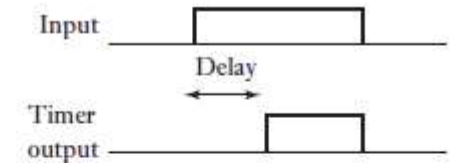
**Figure 14.24** Delay-off timer.



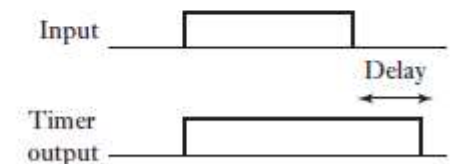
(a)



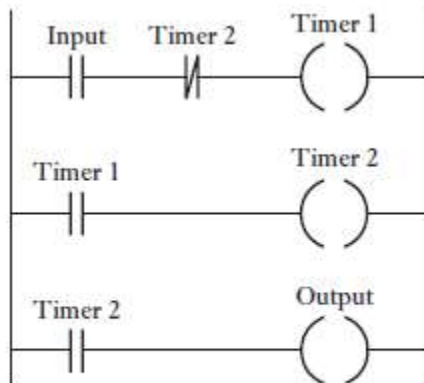
(b)



(c) On-delay timer TON



(d) Off-delay time TOFF



**Figure 14.23** On/off cyclic timer.



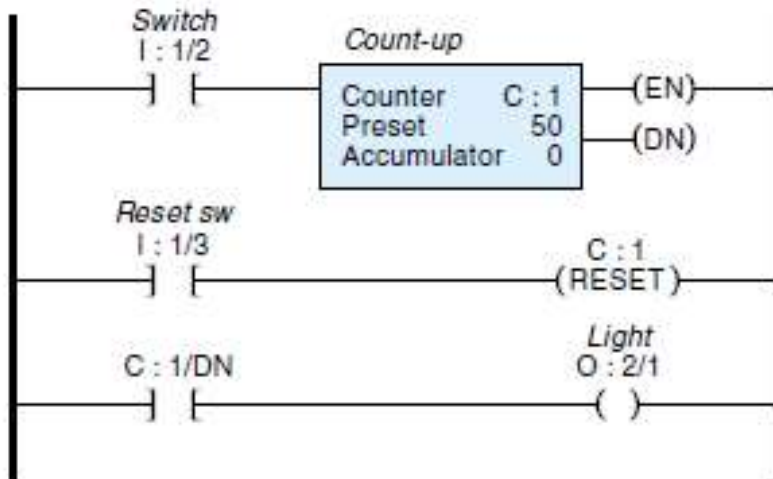


# PLC Counter Function

---

- A counter is a programmed function that counts (increments) every time the input changes from FALSE to TRUE.
  - If in one scan, the input is False, and in the next scan, the input is True, the counter increments. No further counts will occur until the input goes False again and then True.
  - The counter has an address and a preset number of counts. When the preset number of counts have been accumulated, the counter becomes TRUE and can activate some other part of the ladder program.
-

# PLC Counter Function



C : 1

The counter address (actually the first of three addresses) where the first address holds such bits as CU and DN; the second address holds the preset value; and the third address holds the accumulated value.

Preset

When the counter gets to the Preset value, it makes the DN bit go TRUE and keeps on counting.

Accumulator

Holds the value of the current count.

EN

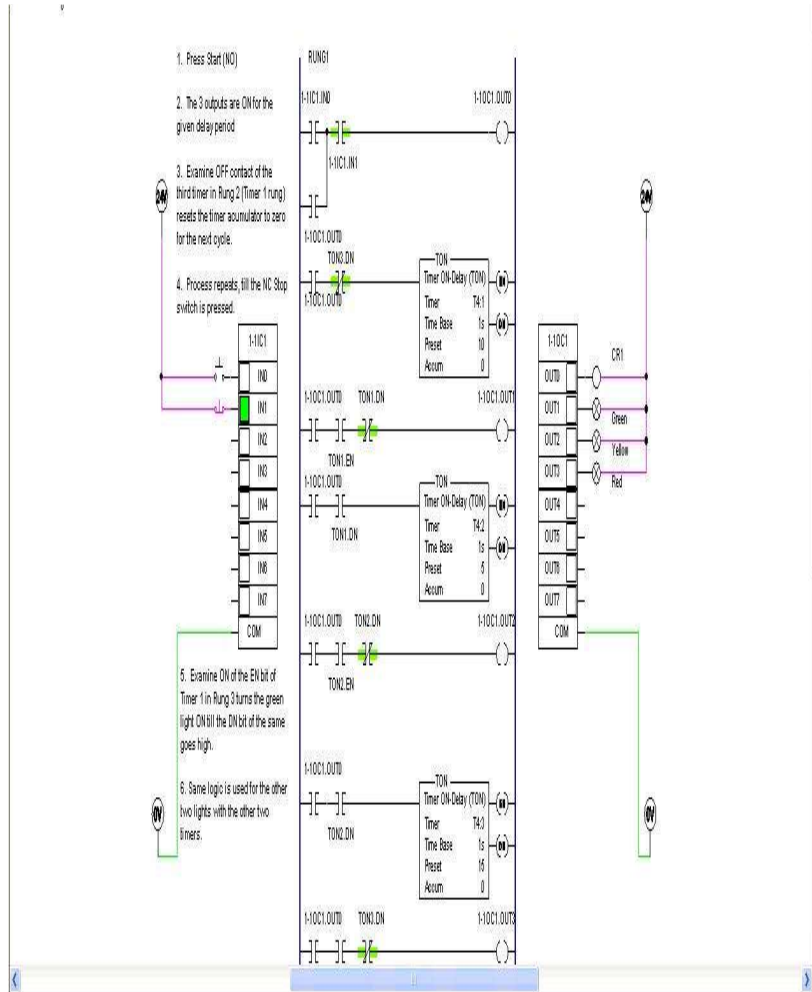
Goes TRUE when the counter rung is TRUE.

DN

Stands for *done*—goes TRUE when the count meets or exceeds the PRESET value.

RESET

A separate instruction with the same counter address; when this bit goes TRUE, the Accumulator resets to zero (resets the counter).



Design and write ladder logic for a simple traffic light controller for following

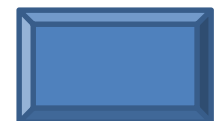
sequence of operation as below,

Step 1 : Turn Green on for 35 seconds

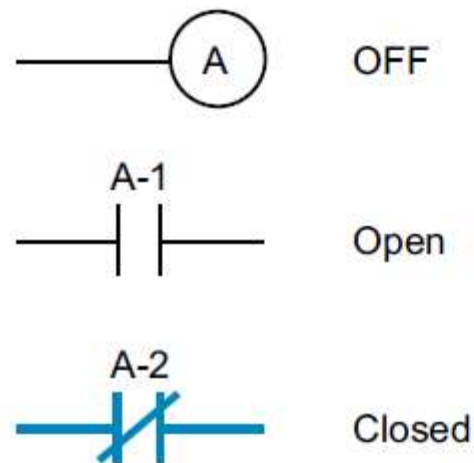
Step 2 : Turn Yellow 1 on for 5 seconds.

Step 3 : Red 2 on for 40 seconds.

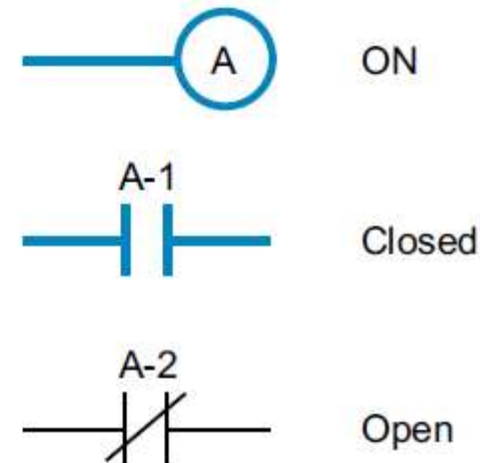
Step 4 : Sequence repeats thereafter.



# N.O / N.C



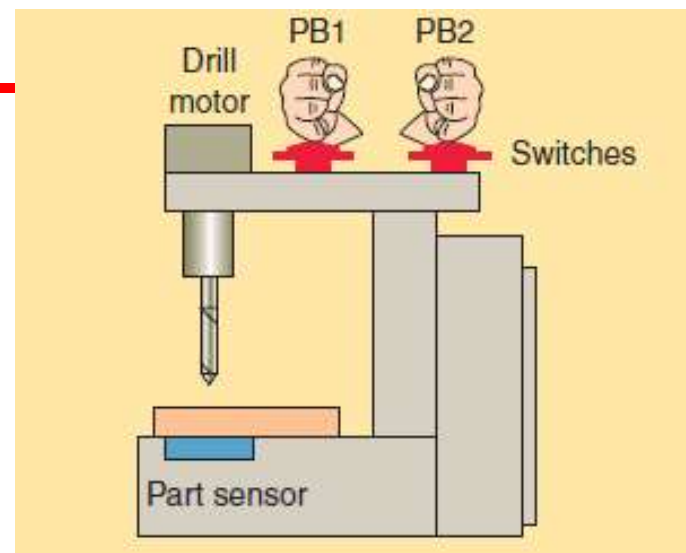
(b) Coil A de-energized.



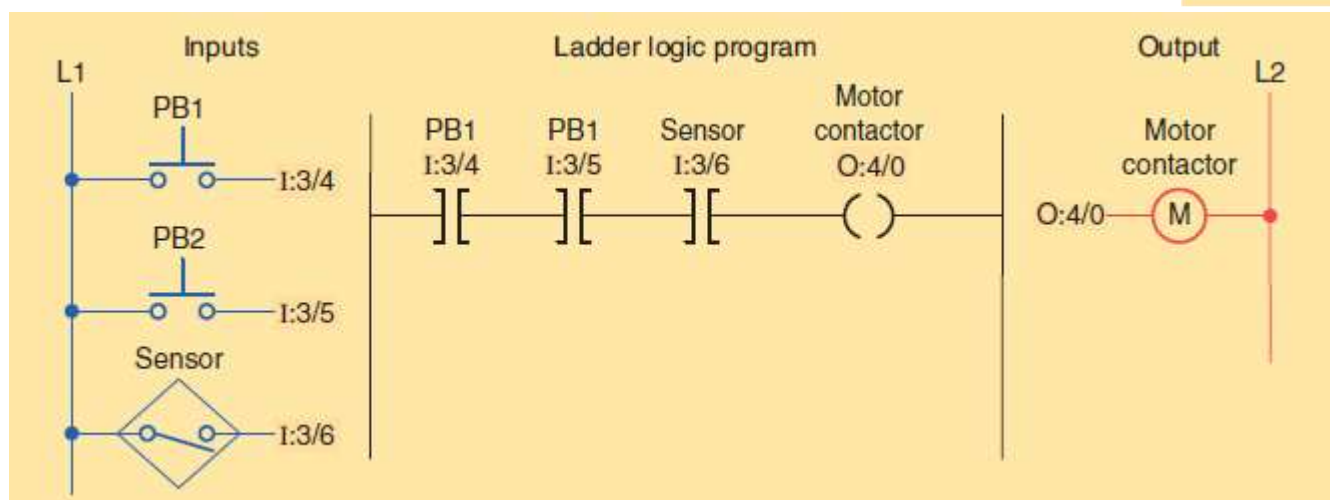
(c) Coil A energized.

A1 – Normally Open or closed?

# Ladder Diagram



Sketch of the drilling process.



# Ladder Diagram

For a system where there has to be no output when any one of four sensors gives an output, otherwise there is to be an output, Figure 6.10 shows the ladder program and the instruction list.

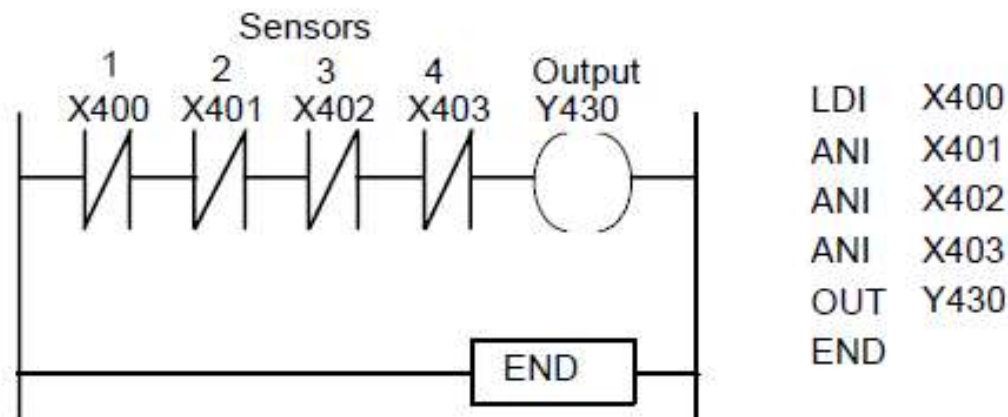


Figure 6.10 *Output switched off by any one of four sensors being activated*

# Ladder Diagram

For a valve which is to be operated to lift a load when a pump is running and either the lift switch is operated or a switch operated indicating that the load has not already been lifted and is at the bottom of its lift channel, Figure 6.9 shows the ladder program and the related instruction list.

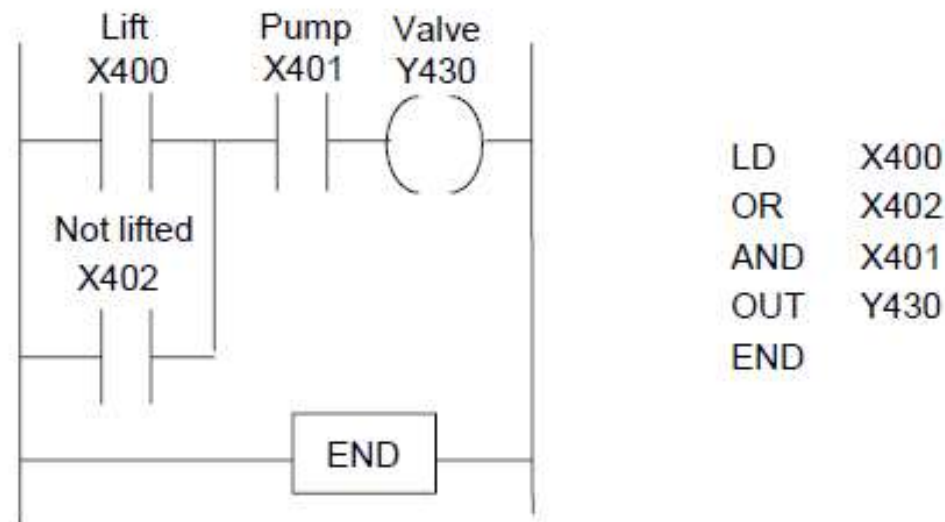
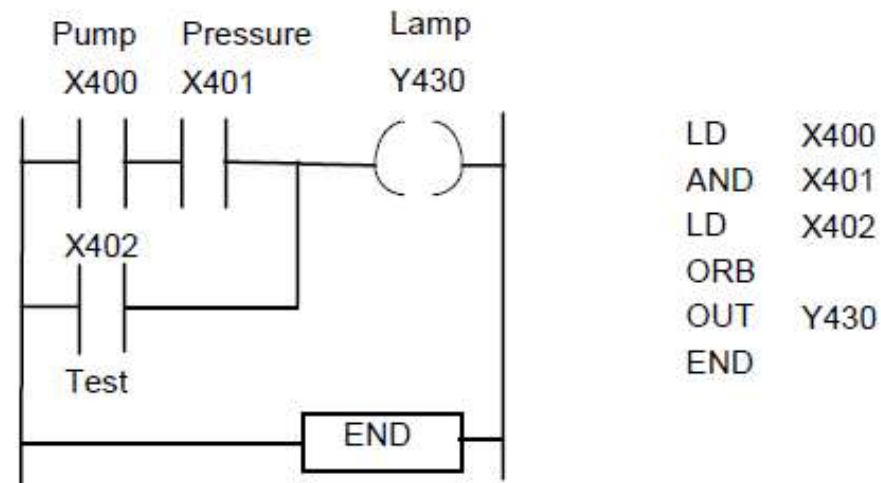


Figure 6.9 *Valve operation program*

# Ladder Diagram

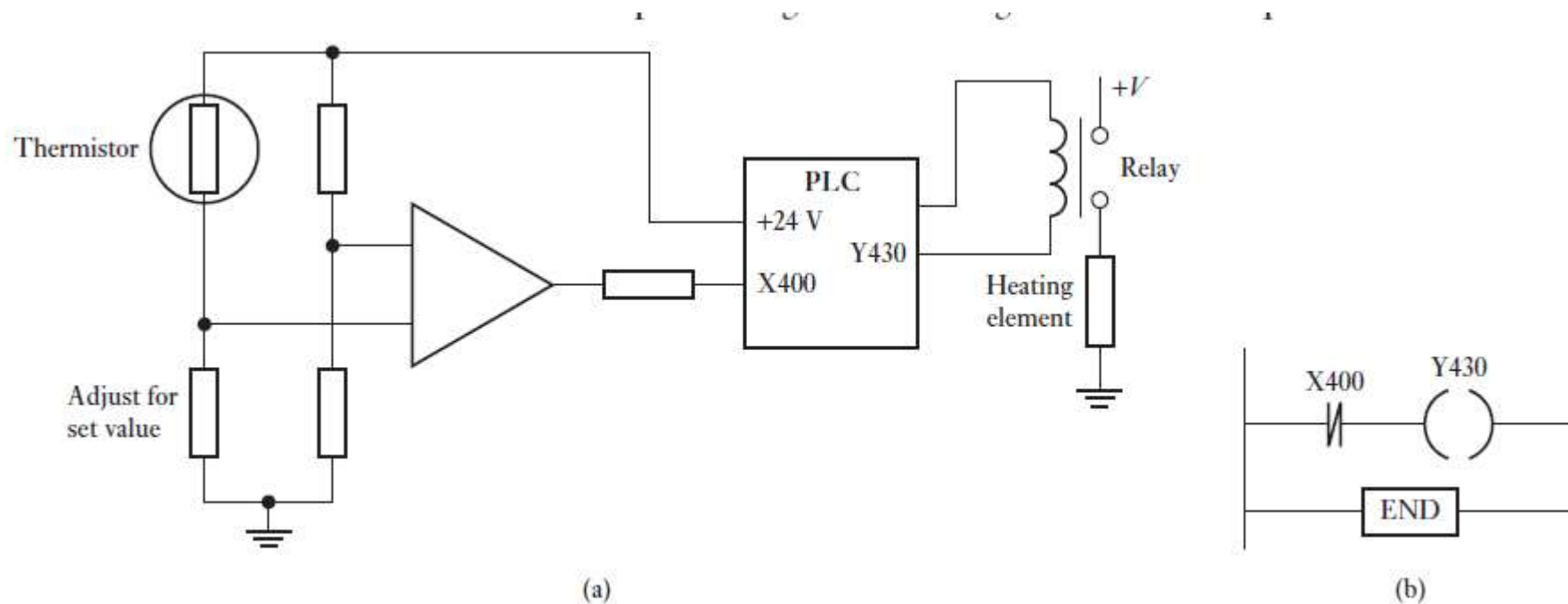
A signal lamp is required to be switched on if a pump is running and the pressure is satisfactory, or if the lamp test switch is closed. Figure 6.8 shows the ladder program and the related instruction list.



Test circuits, where you would like to check operation of system before commissioning



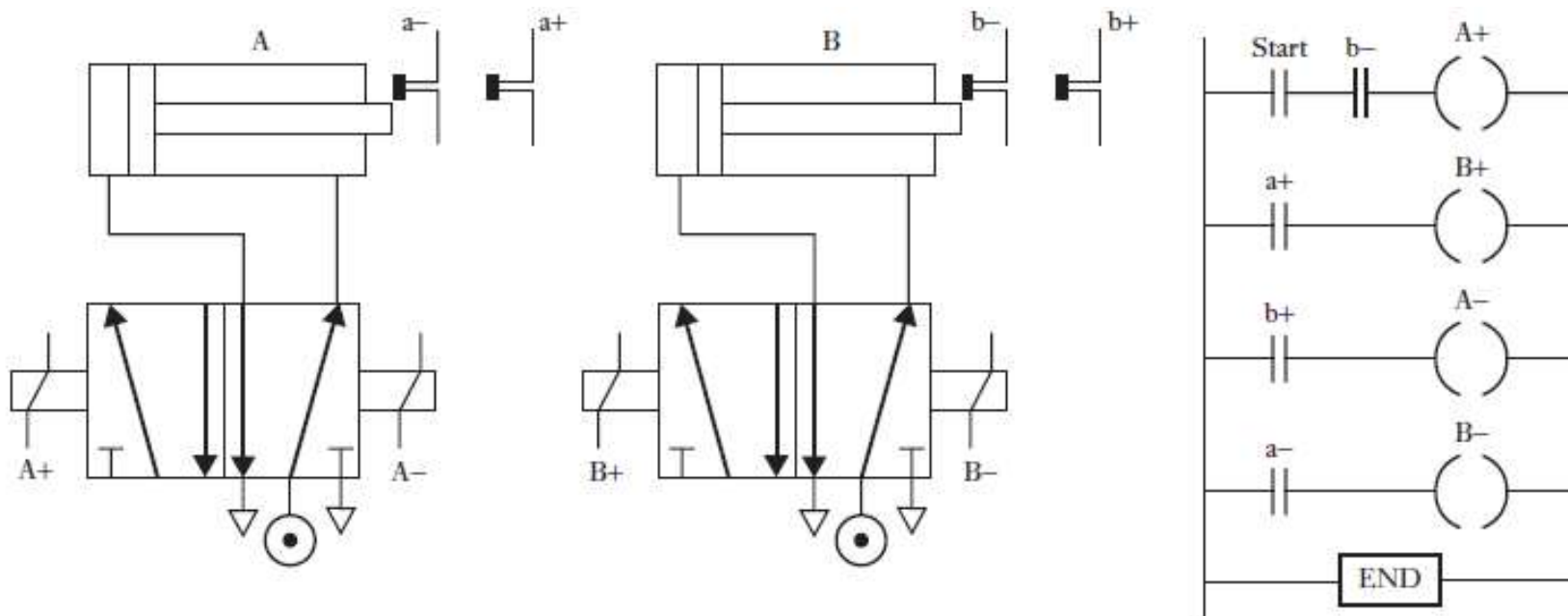
# Ladder Diagram



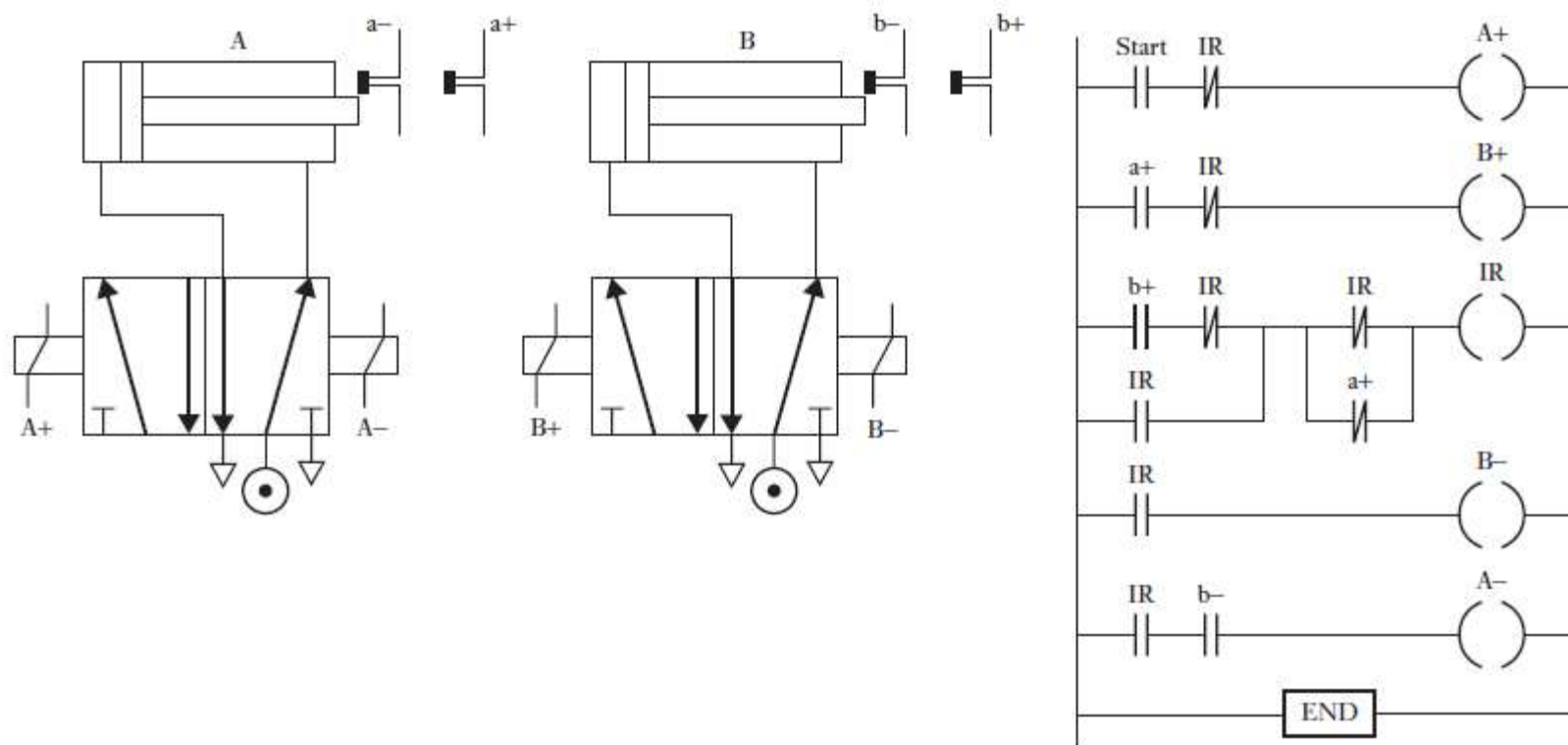
**Figure 14.7** Temperature control system.



# Sequencing of cylinders (A+ B+ A- B-)

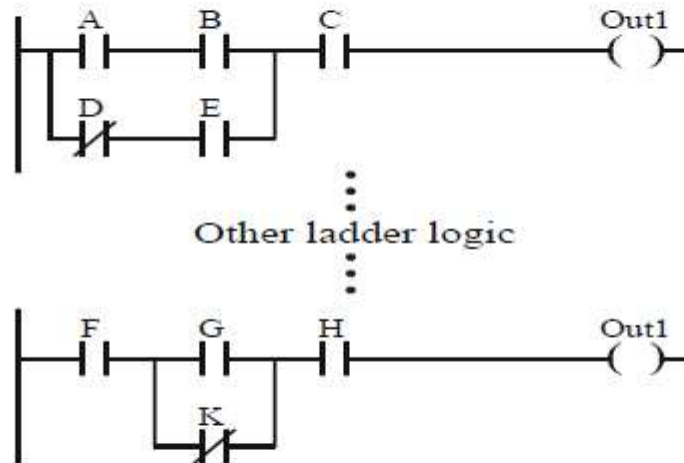


# Sequencing of cylinders (A+ B+ B- A-)



# Repetition of Output coils

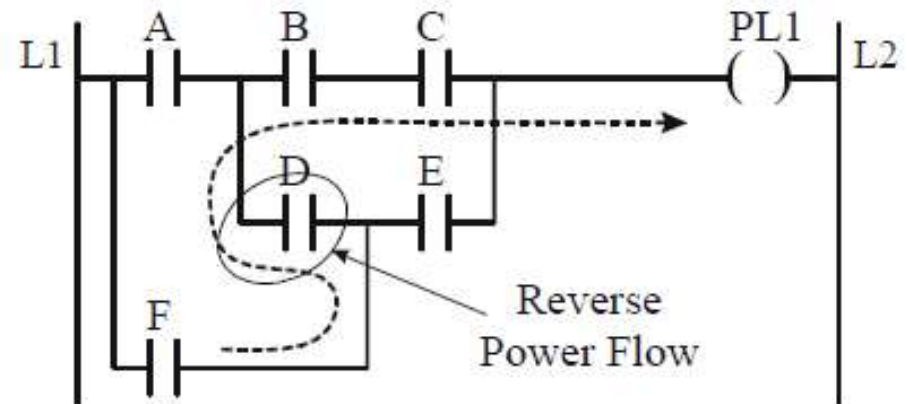
- Do not repeat normal output coils that refer to the same address



- The coils for first and second rung refer to **Out1**
  - Second rung overrides the logic in first rung

# Reverse Power flow

- This is **not** allowed:



# Shift registers

A register is a number of internal relays grouped together, normally 8, 16, or 32. Each internal relay is either effectively open or closed, these states being designated 0 and 1. The term *bit* is used for each such binary digit. Therefore, if we have eight internal relays in the register, we can store eight 0/1 states. Thus we might have, for internal relays:

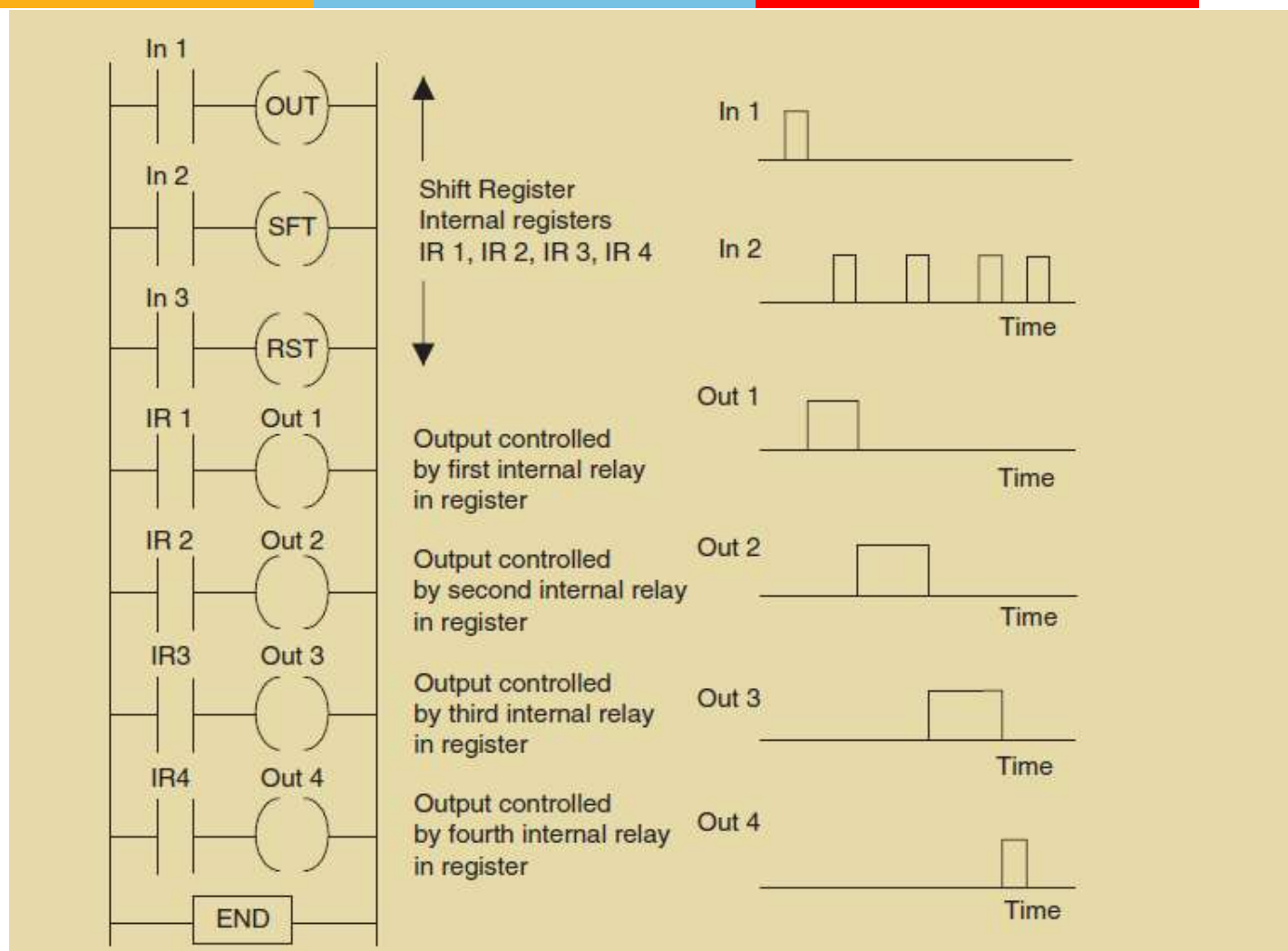
1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

and each relay might store an on/off signal such that the state of the register at some instant is:

1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

that is, relay 1 is on, relay 2 is off, relay 3 is on, relay 4 is on, relay 5 is off, and so on. Such an arrangement is termed an *8-bit register*. Registers can be used for storing data that originate from input sources other than just simple, single on/off devices such as switches.

# Shift registers



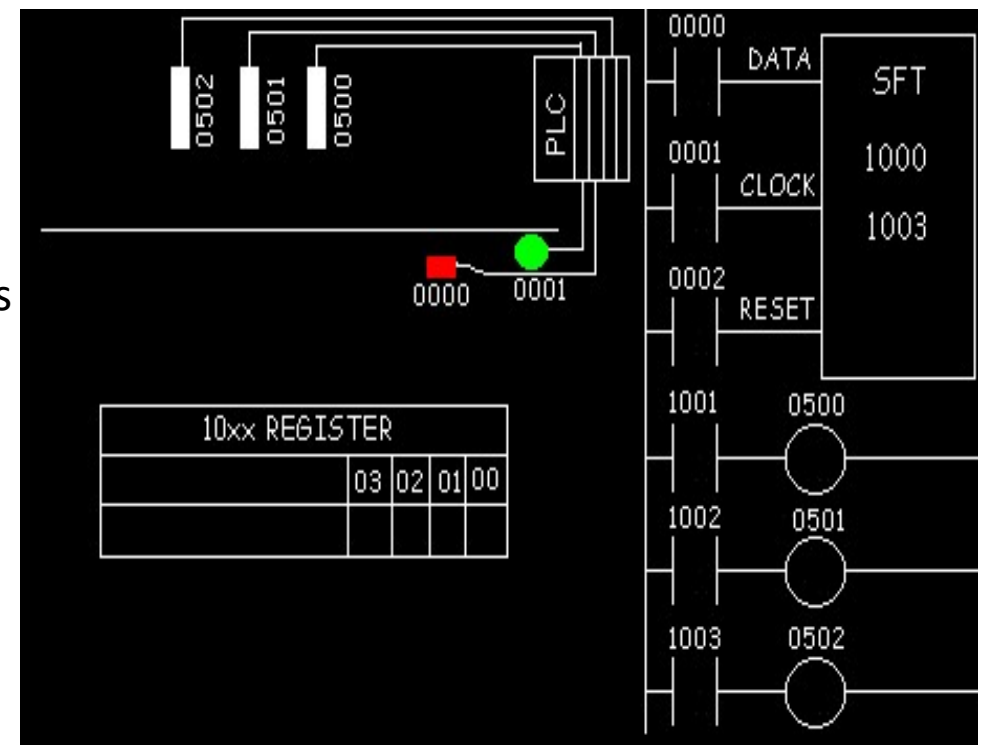


# Shift registers – Ice cream cone Machine

We have 4 steps.

- 1) First we verify the cone is not broken. Next we put ice cream inside the cone.(turn on output 500)
- 2) Next we add peanuts.(turn on output 501)
- 3) And finally we add sprinkles.(turn on output 502)
- 4) If the cone is broken we obviously don't want to add ice cream and the other items

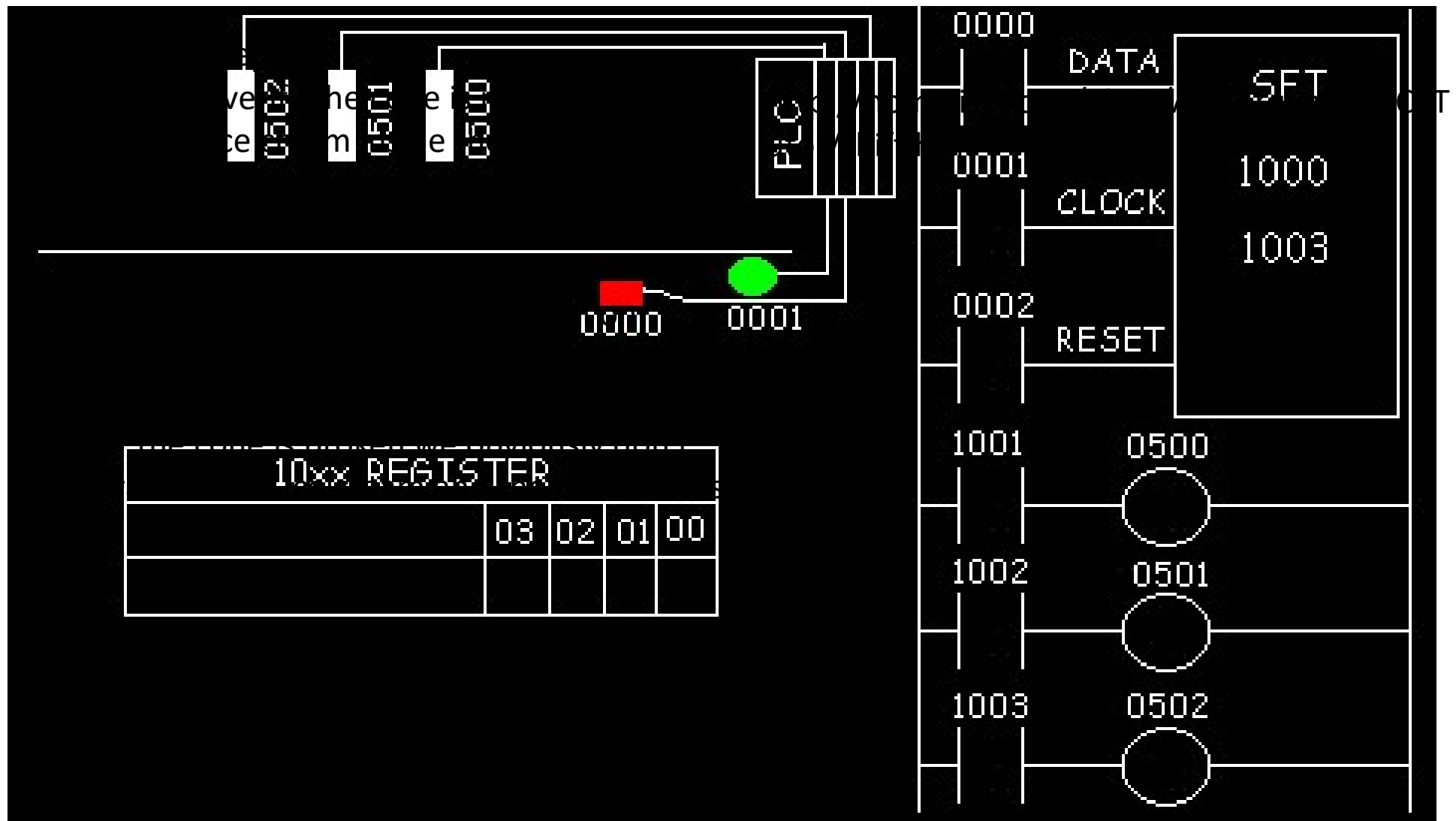
[http://home.isr.uc.pt/~lino/AIR/Arquivo/PLC\\_Tutor/shift.htm](http://home.isr.uc.pt/~lino/AIR/Arquivo/PLC_Tutor/shift.htm)



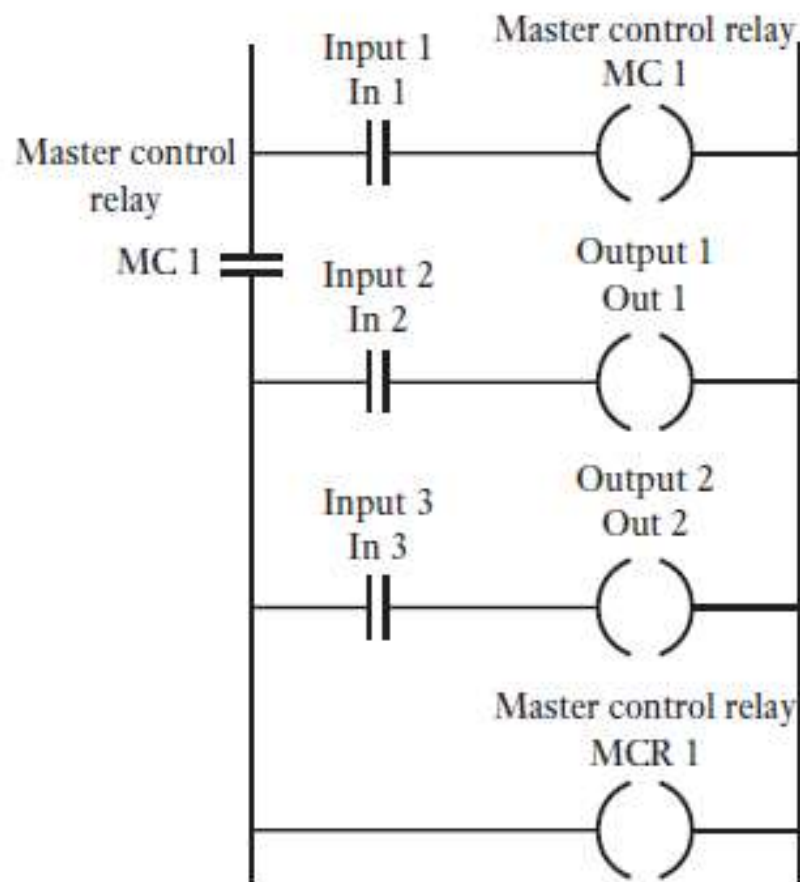




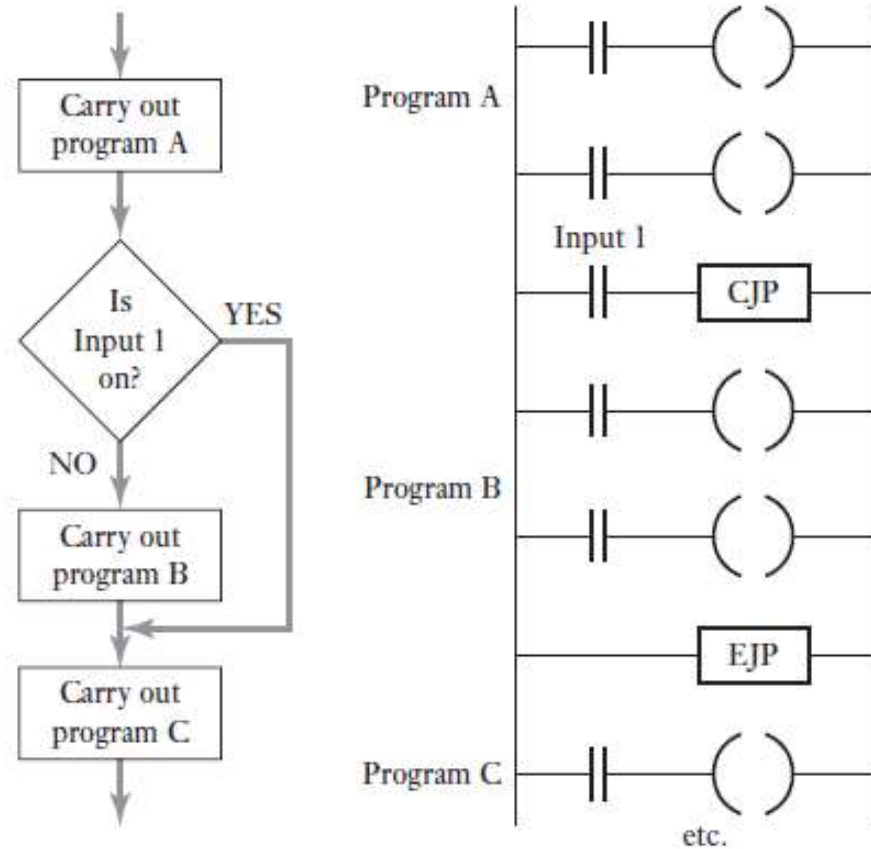
# Shift registers – Ice cream cone Machine



# Master Control



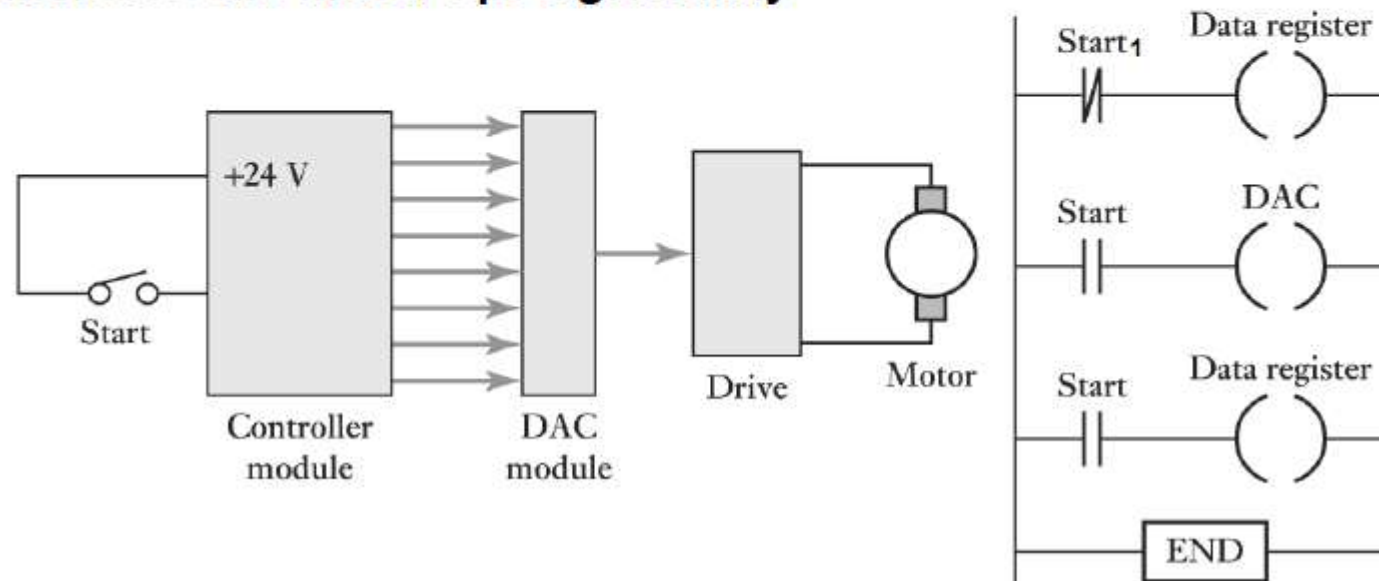
# Jump control Relays



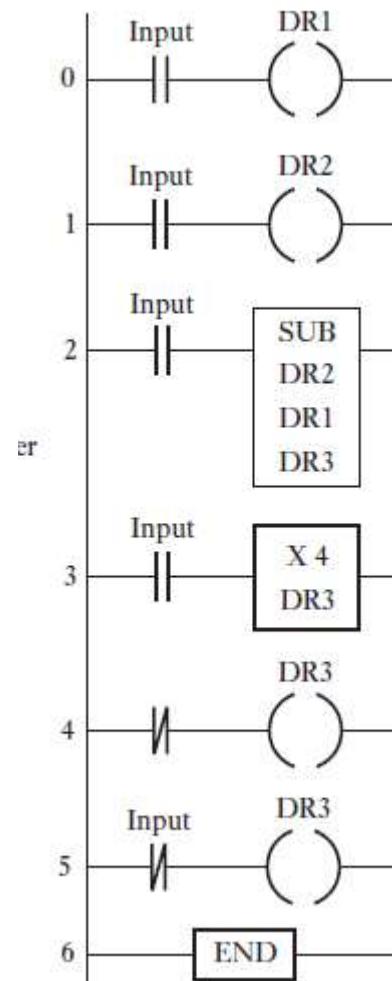
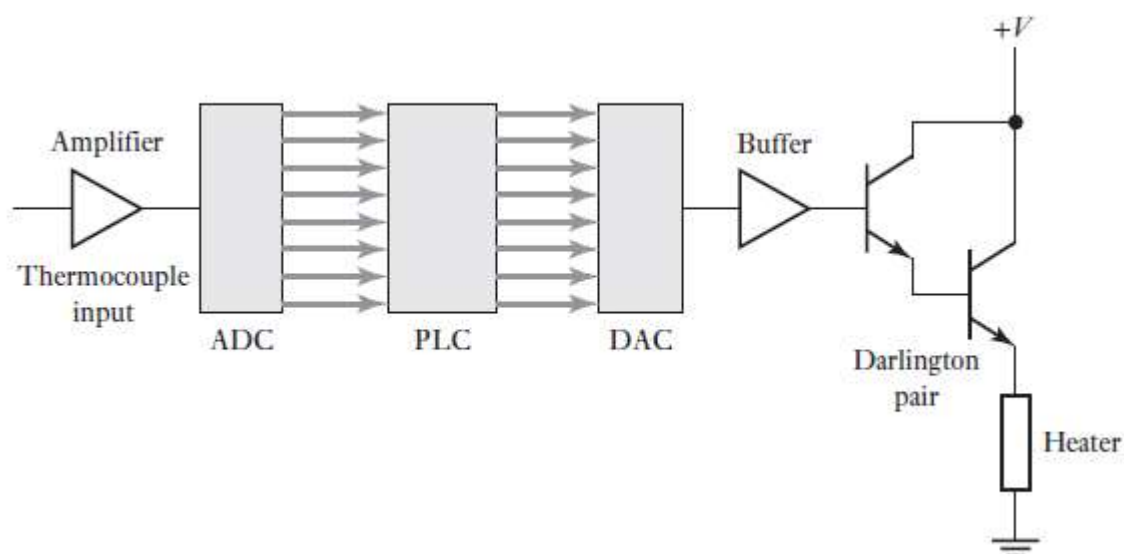
# Analog – Digital converter

Many sensors and actuators working on Analog signals so, ADC are required at the PLC input while DAC are required in the Output interface of PLC

Figure show how DAC can be used in motor speed control to increase the motor speed gradually



# Proportional controller



# Proportional controller

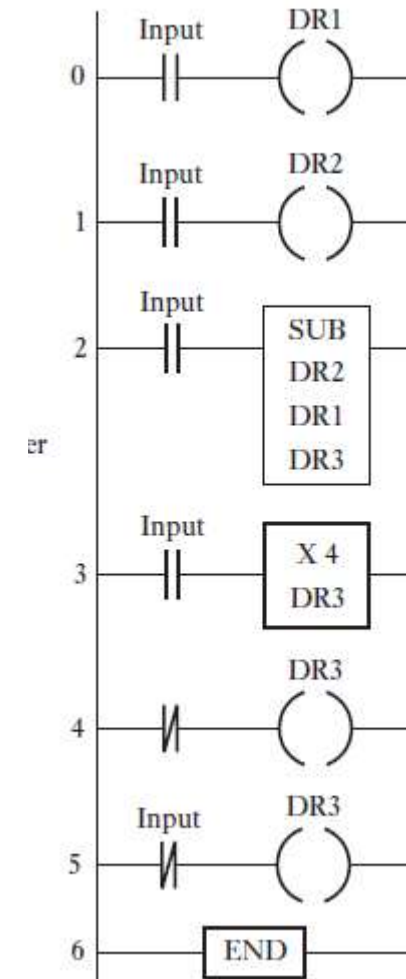
Rung 0 reads the ADC and stores the temperature value in data register DR1.

Rung 1, the data register DR2 is used to store the set point temperature. Rung 2 uses the subtract function to subtract the values held in data registers DR1 and DR2 and store the result in data register DR3, i.e. this data register holds the error value.

Rung 3 multiply function is used, in this case to multiply the value in data register DR3 by the proportional gain of 4.

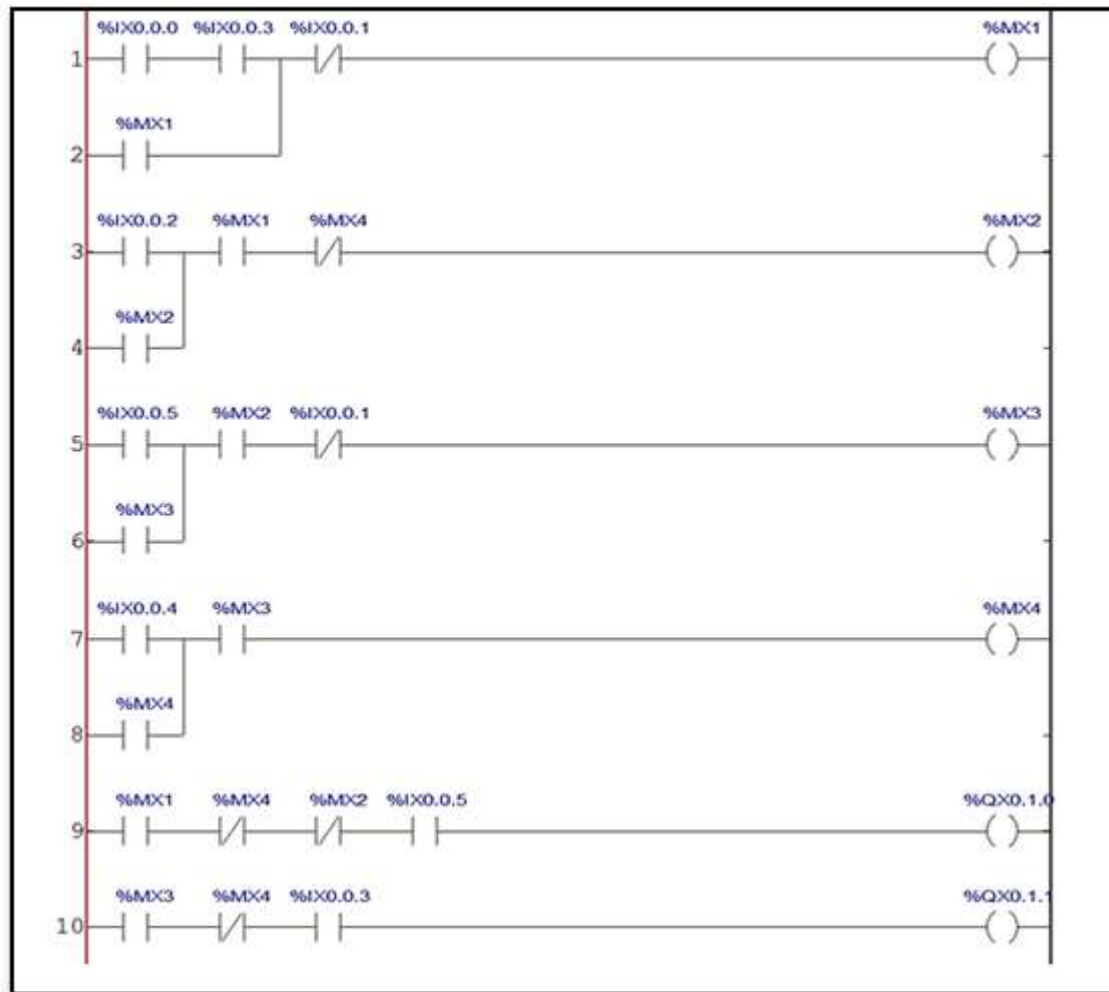
Rung 4 uses an internal relay which can be programmed to switch off DR3 if it takes a negative value.

Rung 5 the data register DR3 is reset to zero when the input is switched off.





# Assignment problem A- A+ B- B+



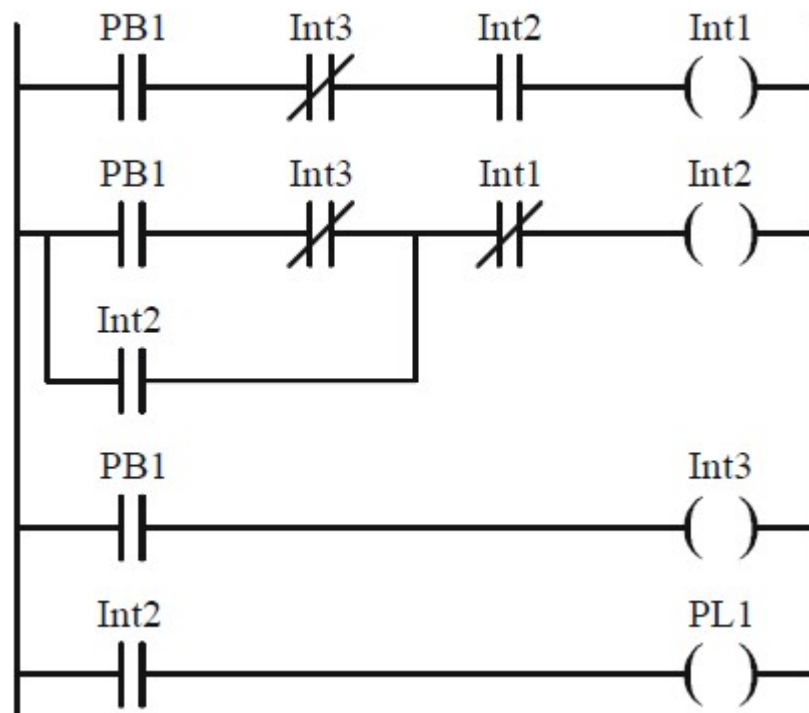
Input 3 - a+

Input 2 - a-

Input 5 - b+

Input 4 - b-

# PLC Processor Scan



Home Work!





# Types of PLCs

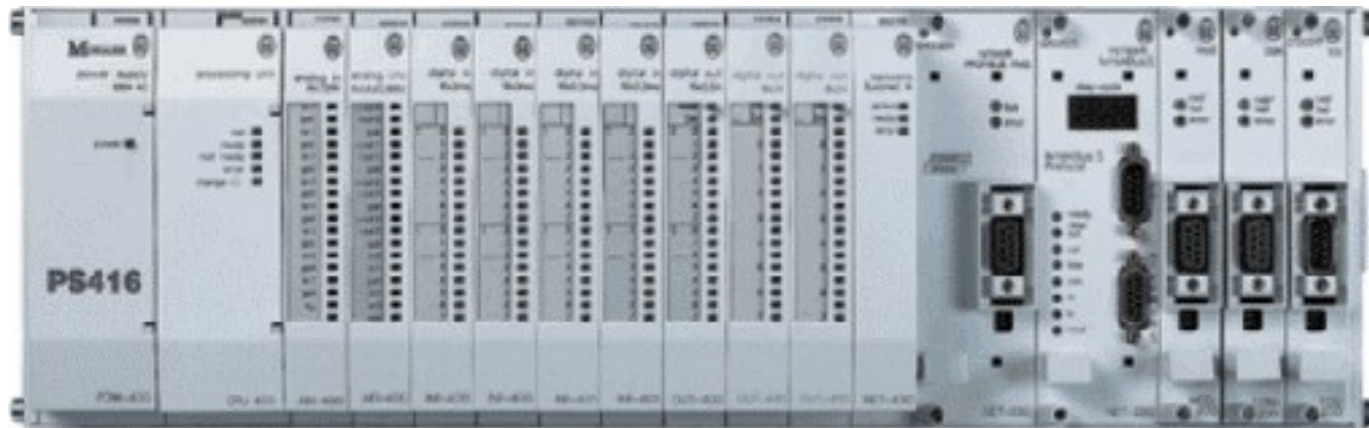
---

- **Single Box Type:** The single box type (or, as sometimes termed, brick) is commonly used for small programmable controllers and is supplied as an integral compact package complete with power supply, processor, memory, and input/output units. Typically such a PLC might have 6, 8, 12 or 24 inputs and 4, 8 or 16 outputs and a memory which can store some 300 to 1000 instructions.
  - **Modular:** The modular or the Rack type consists of separate modules for power supply, processor, input module, output module, memory.
-

# Types of PLCs



Brick Type PLC



Rack Type PLC



---

# Thank you