Programmable Logic Controllers

DASE7503 Robotic Systems Integration

Contents

- Introduction
- Classification of PLCs
- Architecture of a PLC
- Principle of Operation of a PLC
- Programming the PLCs
- Logical Operators
- Role of PLCs in Robotic Systems
- Advantages of Using PLCs in Robotic Systems
- Typical Applications of PLCs integrated with Robotic Systems
- Limitations of PLCs in Robotic Systems

Introduction

- **Programmable Logic Controllers (PLCs)** are used in the equipment of industrial automation and not only there.
- PLCs are used in a modern factory or a smart warehouse.
- The first programmable logic controllers entered the market in the late 60's of the twentieth century. They replaced conventional control systems built using **contactors** and **relays** in which the cabling plays the role of a program.
- A Programmable Logic Controller (PLC) is a universal microprocessor-based device designed to control the operation of various machines, devices or process lines.
- The **cyclic program memory circuit** is a characteristic feature of PLCs distinguished them from other computer-based controllers.

Introduction

- The PLC operation algorithm is developed using a special programming language intended for it.
- To change a control algorithm, it is necessary to change the **memory content**.
- It is possible to write a program using either a **programmer** or a **computer with the dedicated software**.
- PLCs are equipped with appropriate number of **input devices** collecting information about the state of a controlled object and service requests as well as the appropriate number of **output devices** connected to executive, signalling or data transmission elements.

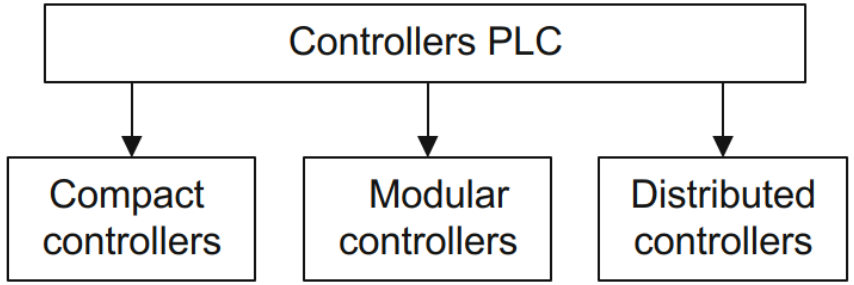
Introduction

- There are a lot of companies that produce programmable logic controllers for sale.
- They offer potential users a wide range of basic PLCs, as well as specialized controllers targeting specific purposes.
- The following companies are the most popular PLC producers: ABB, Allen-Bradley, Delta Electronics, Fatek, GeFanuc, Honeywell, Kinco, Mitsubishi, Moeller Electric, Omron, Panasonic, Schneider Electric Modicon, SIEMENS, Unitronics, Vipa.

Classification of PLCs

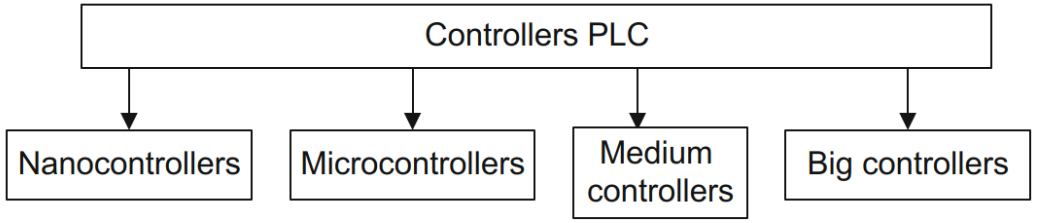
- PLCs offered on the market form a large and diverse product group.
- They start from the simplest models that resemble the programmable relays and end by the very complex devices with capabilities resembling what industrial computers offer.
- PLCs could be classified using two criteria, including (1) **construction** and (2) **the numbers of inputs and outputs**.

Classification of PLCs



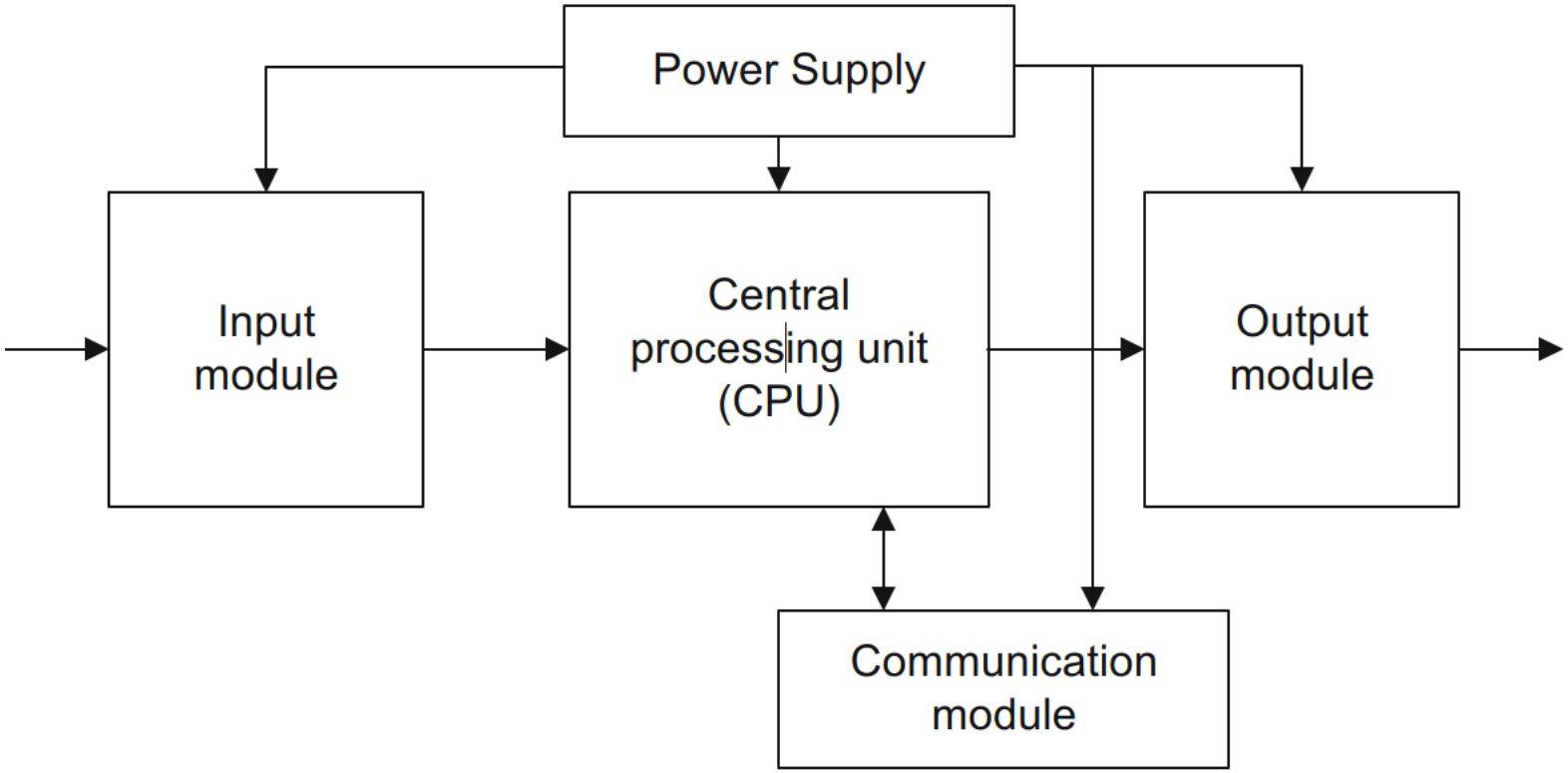
- The division of PLCs according to the applied construction is shown in the Figure.
- A **compact PLC** is a device in which the main elements such as CPU, power supply and a small number of inputs and outputs are integrated in one housing. Some of them possess a small display and keyboard.
- **Modular controllers** are
  - more complicated devices.
  - used where it is required to connect a larger number of input or output blocks.
  - characterized by a flexible construction which adapts to the needs of a desired control object through the selection of appropriate modules such as a CPU, input-output modules, or communication modules.
- **Distributed controllers** are the devices with elements placed in the distributed manner.

Classification of PLCs



- The division of PLCs according to **the number of inputs and outputs** is shown in the Figure.
- **Nanocontrollers** are the smallest devices available on the market. These include controllers having **up to 32** input/output points.
- **Microcontrollers** have from **33 to 128** input/output points. As a rule, both nano- and microcontrollers belong to the class of ***compact*** controllers. So, they have a compact, uniform construction.
- **Medium controllers** have from **129 to 1024** input/output points.
- **Big controllers** have more than 1024 input/output points.
- As a rule, both medium and big controllers have ***modular construction***. They have no a uniform construction; it is selected to meet the specifics of a given control object.
- In the case of the expansion of a control system, it is possible to extend the controller with additional modules.
- However, **the largest devices** have the ***distributed construction***, where the modules are connected with each other through a suitable bus.

Architecture of a PLC



Architecture of a PLC

- **Power Supply** – The PLCs are most often supplied with 24 V DC voltage. There are also constructions adapted to be supplied directly from the 110 V AC or 230 V AC power grid.
- **Central Processing Unit (CPU)** - Responsible for execution of the program, i.e. tasks specified by the user. The task of the CPU is to perform the tasks described in the control program based on information from input modules. The result of program implementation is some information for the control object in the form of appropriate output states.
- **The Module of Digital and Analog Inputs** - An element enabling the input to the controller of input signals from sensors, room units (traces) and measuring devices. These signals may be either **digital** or **analog**. The task of input modules is a transformation of signals from different devices in the form understandable by CPU.

Architecture of a PLC

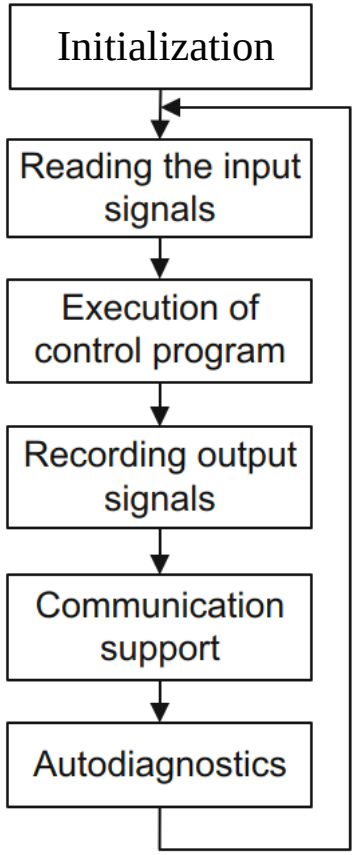
- **The Module of Digital and Analog outputs –** A subunit enabling transferring control signals to different actuators and executive elements. These signals may be either digital or analog. The output modules transform the results generated by a program into the object signals
- **The Communication Module** is used to connect the controller with other devices such as a programmer or some local area network devices, for example, such as Mod- bus, PROFIBUS, Profinet, ControlNet, Genius, and so on.

Principle of Operation of PLC

- The PLC controller analyses input signals and executes control algorithms in the real-time mode.
- The PLC controller operates according to **a specific control program** kept in its **memory**.
- The control program contains **a sequence of logic instructions** that control the operation of a device.
- Creating a control program consists in distributing a control task to individual instructions.
- An **instruction** is the smallest part of a control program.
- After uploading the program into a controller, these instructions are processed by **the microprocessor devices**, which performs appropriate actions on the base of these instructions (for example, it may be executed either **arithmetic operations** or **logical operations** or writing into the memory, and so on).

Principle of Operation of PLC

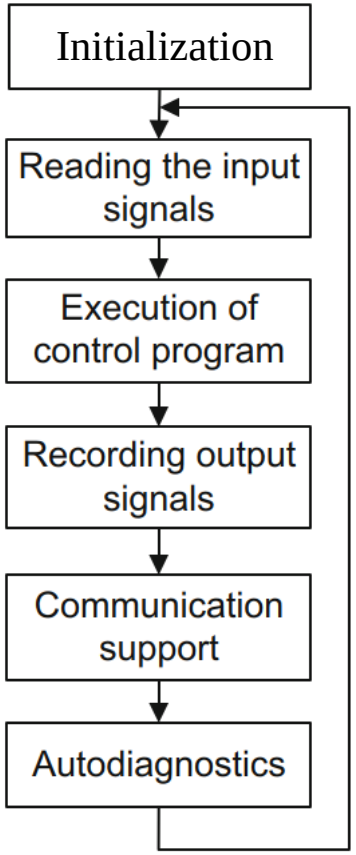
- **Initialization** - In this phase, the controller executes a series of preliminary and control actions. It checks the correctness of the configuration of the structure and basic elements.
- **Reading the input signals** - Each program cycle begins from the reading the states of each controller input. Next, the state is written into so called picture of process inputs. During the phase of actualization of the outputs, the calculated values of outputs are copied from the appropriate area of data memory to the output modules which generate the control signals.
- **Execution of control program** - During this phase, only the user program is executed. The program is executed instruction by instruction. Subsequent instructions are transferred from the program memory to the microprocessor, which decodes them into the machine code, performs appropriate actions, and writes the outcomes into the data memory. When all instructions are executed and corresponding output states are generated, the controller writes the output states in the area of its memory used as the picture of process outputs.



The controller cycle's phases

Principle of Operation of PLC

- **Recording output signals -** The outputs are operated by downloading from the area of memory with the picture of process outputs. The operating system activates appropriate outputs controlling the actuators. During the phase of actualization of the outputs, the calculated values of outputs are copied from the appropriate area of data memory to the output modules which generate the control signals.
- **Communication support -** During the phase of communication support, two- ways transmission data with external devices is executed. For example, there is the transmission to other controllers, or the operator panel, or the programmer.
- **Autodiagnostics -** The last phase of each cycle is devoted to internal diagnostic operation. For example, such parameters are checked as the battery voltage supplying the RAM memory of the controller, the power supply voltage, or the occurrence the errors in the program execution.



The controller cycle's phases

Programming the PLCs

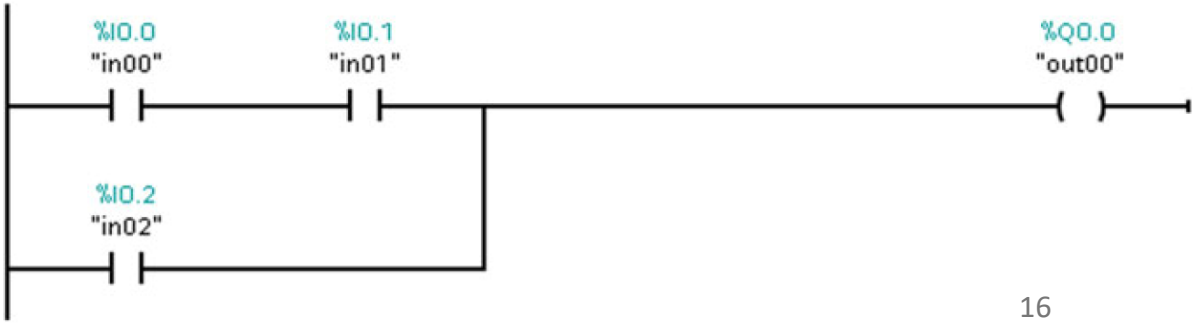
- With the increasingly common use of PLCs, there has been a need to standardize programming methods or device connection standards.
- For the programmer, the most important is the third part of the IEC 61131 standard , which concerns the principles of the programming controllers.
- There are two groups of programming languages identified by the IEC 61131-3 standard, namely, the **text languages** and **graphic languages.** The following languages are the textual:
  - Instruction List (IL),
  - Structured Text (ST).
- There are the following graphic languages:
  - Ladder Diagram (LD),
  - Function Block Diagram (FBD).

Programming the PLCs

The Ladder Diagram Language

- It allows implementation of control task using standard graphic symbols.
- These symbols are placed in circuits in a manner similar to ladders in ladders diagrams for control relay circuits.
- The appearance of a written program is practically the same as the appearance of a contactor-relay circuit.
- There are two basic symbols in this language, namely, contacts and two-state outputs.
- The contacts present logic values of input signals and Boolean variables.
- The two-state outputs are reflections of relay coils which are used to drive discrete outputs and assign logic values to Boolean variables.
- In addition, the ladder diagrams use **function blocks** describing **complex functions** such as counters, timers, comparators, arithmetic operations, and so on.
- There is a fragment of a program in LD language.

The practical examples of their use in programming the **SIMATIC S7 - 1200 controller**.



Introduction to Logical Operators

- **Boolean algebra** is a formal system composed of non-empty set including two elements, namely, one and zero. Two operations with two arguments (binary operations) are defined for this set:
  - logical sum (Boolean sum, disjunction, alternative),
  - logic product (Boolean product, conjunction),

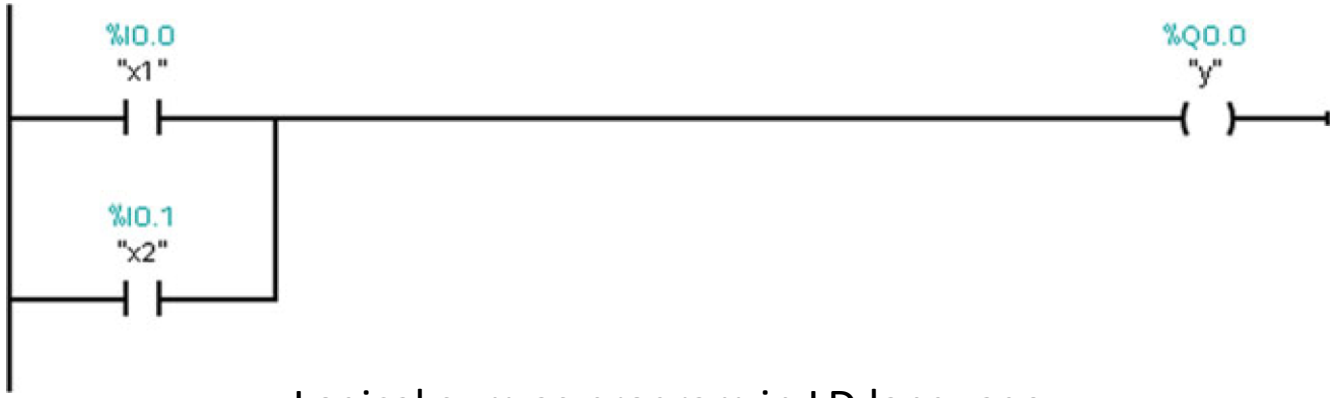
  and a single **unary** operation (with a single argument):
  - negation (inversion).
- It is possible to represent the basic logical operations using either an expression or a truth table. The logical expressions may be implemented in PLC controllers using, for example, the **LD** language.

Introduction to Logical Operators

- **The operation of logical sum** is defined in the following way.
- If at least one argument is equal to 1, then the outcome is equal to 1. If both operands are zeros, then the sum is equal to zero.
- This operation is represented using the logical expression: $y = x1 + x2$, the truth table and the program in **LD language**, as shown in the figure.

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Truth table for logical sum



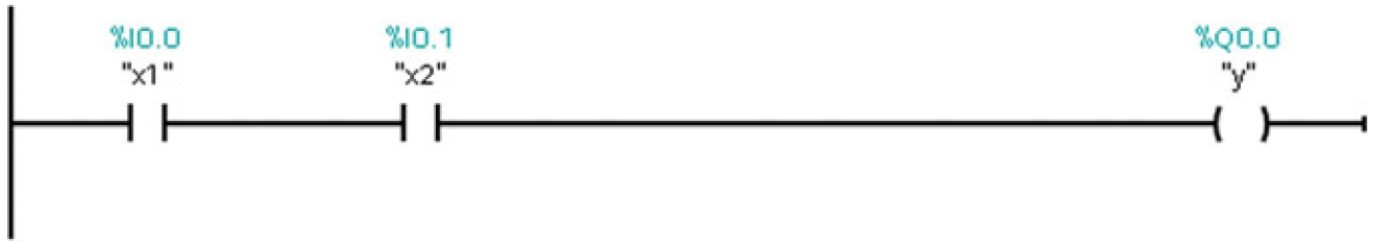
Logical sum as program in LD language

Introduction to Logical Operators

- **The operation of logical product** is defined in the following way.
- The outcome is equal to 1 if all arguments are equal to 1.
- This operation is represented using the logical expression: $y = x1 \cdot x2$, the truth table and the program in LD language, as shown in Figure.

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

Truth table for logical product



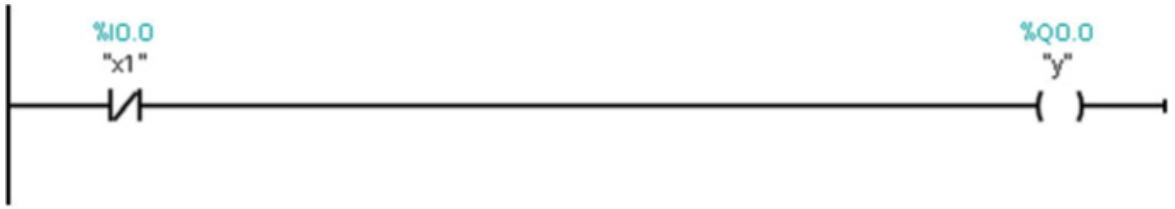
Logical product as program in LD language

Introduction to Logical Operators

- **The operation of negation** changes the value of argument to the opposite.
- This operation is represented using the logical expression: $y = \overline{x}1$, the truth table and the program in LD language, as shown in Figure.

x1	y
0	1
1	0

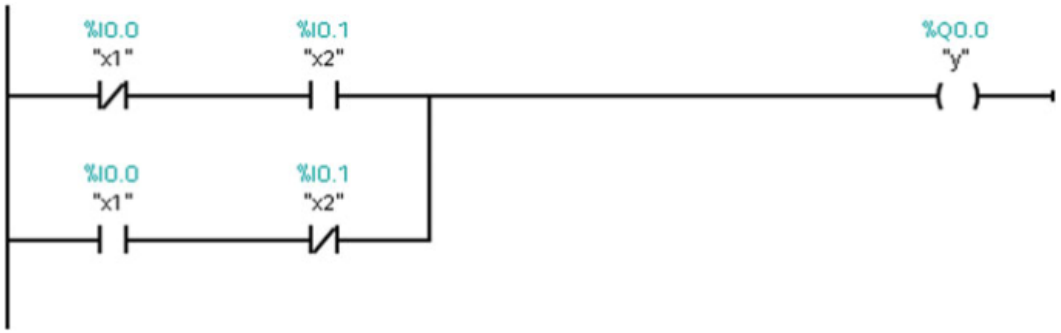
Truth table for negation



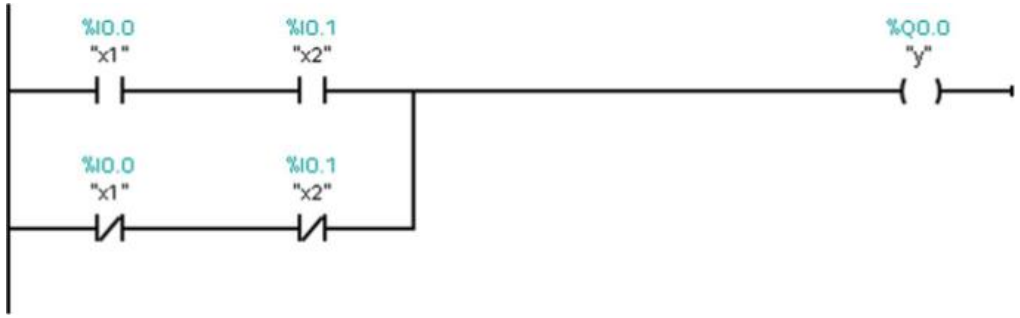
Logical negation as program in LD language

Introduction to Logical Operators

- There are logic gates **XOR** and **XNOR** shown in Figures, respectively.



XOR gate as program in LD language



XNOR gate as program in LD language

Introduction to Logical Operators

- They are represented as logical expressions: XOR $y = \bar{x}1 \cdot x2 + x1 \cdot \bar{x}2$ and XNOR $y = x1 \cdot x2 + \bar{x}1 \cdot \bar{x}2$, truth tables and programs in LD language.

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Truth table for XOR logic gate

x1	x2	y
0	0	1
0	1	0
1	0	0
1	1	1

Truth table for XNOR logic gate

Introduction to Logical Operators

- For any elements x, y, z of Boolean algebra, the following laws take places.

Nr	Law	AND form	OR form
1.	Identity	$1 \cdot x = x$	$0 + 1 = x$
2.	Null	$0 \cdot x = 0$	$1 + x = 1$
3.	Idempotent	$x \cdot x = x$	$x + x = x$
4.	Inverse	$x \cdot \bar{x} = 0$	$x + \bar{x} = 1$
5.	Commutative	$x \cdot y = y \cdot x$	$x + y = y + x$
6.	Associative	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$(x + y) + z = x + (y + z)$
7.	Distributive	$x + y \cdot z = (x + y) \cdot (x + z)$	$x \cdot (y + z) = x \cdot y + x \cdot z$
8.	Absorption	$x \cdot (x + y) = x$	$x + x \cdot y = x$
9.	DeMorgan	$\overline{(x \cdot y)} = \bar{x} + \bar{y}$	$\overline{(x + y)} = \bar{x} \cdot \bar{y}$
10.	Double evolution	$\overline{(\bar{x})} = x$	

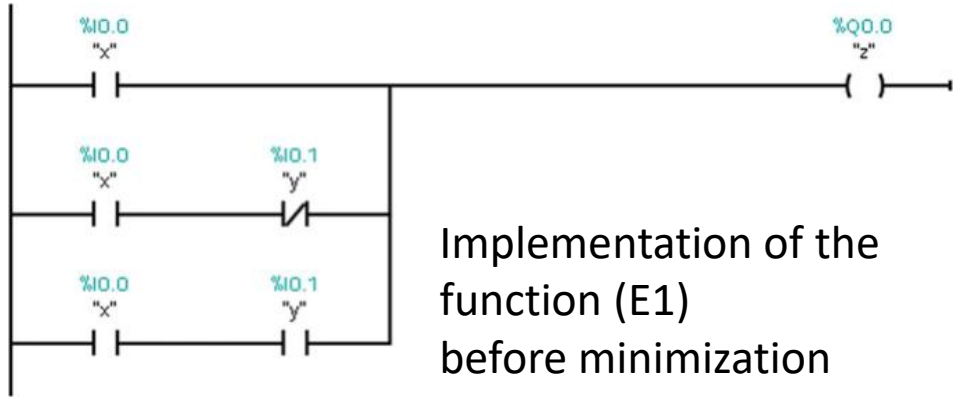
Introduction to Logical Operators

- These laws are used for **minimizing Boolean functions**. The minimization is used to simplify logical expressions in such a way that they include the minimum possible number of arguments and Boolean operations.
- They are used mainly in digital technique, in which the functions are represented using logic gates.
- In practice, the simpler expression means less complexity of digital circuit, and, thus lower cost of execution, greater reliability, and faster operation.

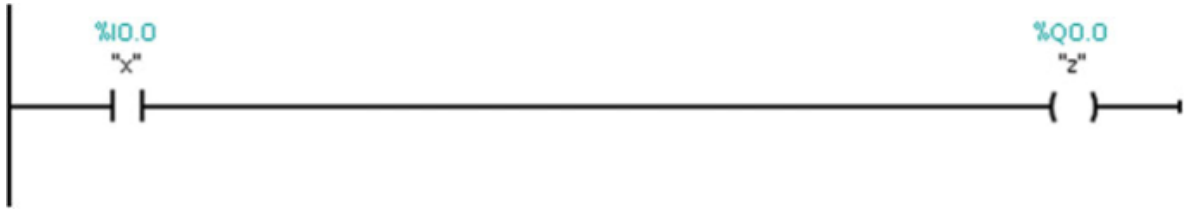
Introduction to Logical Operators

- In the case of logical controllers, using **minimization** allows simplifying a program code and increasing its transparency.
- Let us consider the following function: $z = x + x \cdot \bar{y} + x \cdot y.$ (E1)
- Let us minimize it using the laws from Table in Slide 23

$$\begin{aligned} z = x + x \cdot \bar{y} + x \cdot y &= x + x \cdot (\bar{y} + y) &&= Law\ 7, \\ &= x + x \cdot (1) &&= Law\ 4, \\ &= x + x &&= Law\ 1, \\ &= x &&= Law\ 3. \end{aligned}$$



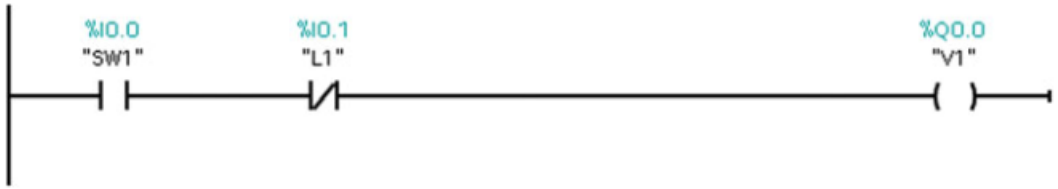
Implementation of the function (E1) before minimization

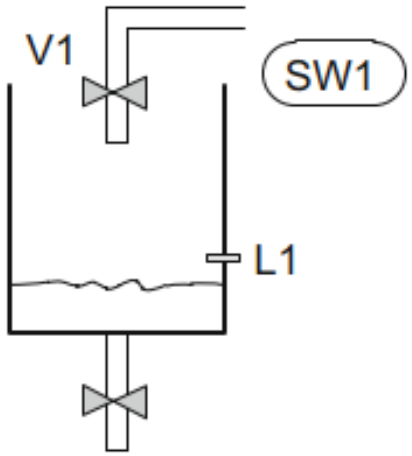


Expression (E1) after minimization

Introduction to Logical Operators

- This example shows a way of moving from a word description to a description with logic functions.
- Filling the tank is carried out by opening the solenoid valve *V* 1. The tank should be filled if the level of liquid is bellow the sensor *L*1. In this case, the button *SW* 1 should be switched on.
- This dependence may be described as the following:
  - $V\,1 = SW\,1 \cdot \overline{L}1$. An example of a program in LD language is shown in the following Figure.





Tank filling process

Role of PLCs in Robotic Systems

- **Automation and Control**: PLCs serve as the central control units that automate the movement, operations, and coordination of robotic arms and mechanisms, ensuring precise and repeatable tasks.

- **Interfacing with Sensors and Actuators**: They process signals from various sensors (proximity, pressure, vision, etc.) and send commands to actuators (motors, grippers, etc.) to execute desired actions.

- **Sequence Control**: PLCs manage complex sequences of operations, ensuring that each step occurs in the correct order and timing, which is essential in manufacturing and assembly lines.

- **Data Logging and Monitoring**: They monitor system performance, collect data for analysis, and facilitate diagnostics and troubleshooting.

Typical Applications of PLCs integrated with Robotic Systems

- Assembly lines (e.g., welding, pick-and-place)
- Material handling and logistics
- Packaging and palletizing
- Quality inspection automation
- Automated machining and drilling

Advantages of Using PLCs in Robotic Systems

- **Flexibility:** Easily reprogrammed to adapt to different tasks or processes.
- **Reliability:** Designed to operate continuously in industrial environments.
- **Integration:** Can communicate with other control systems, HMI (Human-Machine Interface), and enterprise software.
- **Scalability:** Suitable for small to large robotic systems with multiple axes and functionalities.

Limitations of PLCs in Robotic Systems (1)

Understanding these constraints is important for designing effective robotic applications. Here are some common limitations of PLCs in robotic systems:

- **Limited Processing Power:**
  - PLCs are optimized for logic control rather than complex computations or high-speed processing.
  - They may struggle with tasks requiring intensive data processing, real-time image analysis, or complex mathematical calculations.
- **Speed Constraints:**
  - PLCs often have slower cycle times compared to specialized motion controllers or embedded systems.
  - This can limit the responsiveness of robotic operations, especially in high-speed or high-precision applications.
- **Limited Advanced Motion Control:**
  - While some PLCs support basic motion control, they may lack the advanced features of dedicated servo controllers or CNC systems, such as smooth trajectory planning or multi-axis synchronization.
- **Scalability and Complexity:**
  - Managing very complex or highly synchronized multi-robot systems can become cumbersome with standard PLCs.
  - Large-scale systems might require multiple PLCs or integration with other control platforms, increasing complexity and cost.

Limitations of PLCs in Robotic Systems (2)

Understanding these constraints is important for designing effective robotic applications. Here are some common limitations of PLCs in robotic systems:

- **Cost for High-Performance Applications:**
  - High-performance PLCs or those with advanced motion and vision capabilities can be expensive.
  - For highly sophisticated robotic tasks, dedicated controllers may offer better performance-to-cost ratios.
- **Limited Integration with High-Level Systems:**
  - PLCs may have limitations in integrating seamlessly with advanced vision systems, AI modules, or cloud-based analytics without additional interfaces or modules.
- **Programming Complexity:**
  - Programming PLCs often requires specialized languages (ladder logic, function block diagrams), which might be less intuitive for complex robotic algorithms compared to high-level programming languages.
- **Lack of Flexibility for Certain Tasks:**
  - Tasks involving adaptive learning, complex path planning, or computer vision may be beyond the capabilities of standard PLCs, necessitating supplementary systems.