RQF LEVEL 5

CSASA501

COMPUTER SYSTEM AND ARCHITECTURE

System Automation with PLC

TRAINEE'S MANUAL

October, 2024

SYSTEM AUTOMATION WITH PLC

KOICA Korea International Cooperation Agency

TQUM TVET Quality Management Project

2024

AUTHOR'S NOTE PAGE (COPYRIGHT)

Copies available from:

- *HQs: Rwanda TVET Board-RTB*
- *Web: www.rtb.gov.rw*
- **KIGALI-RWANDA**

Original published version: October 2024

ACKNOWLEDGEMENTS

The publisher would like to thank the following for their assistance in the elaboration of this training manual:

Rwanda TVET Board (RTB) extends its appreciation to all parties who contributed to the development of the trainer's and trainee's manuals for the TVET Certificate V in in Computer System and Architecture, specifically for the module "**CSA501: System Automation with PLC**"

We extend our gratitude to KOICA Rwanda for its contribution to the development of these training manuals and for its ongoing support of the TVET system in Rwanda.
We extend our gratitude to the TQUM Project for its financial and technical support in the development of these training manuals.

We would also like to acknowledge the valuable contributions of all TVET trainers and industry practitioners in the development of this training manual.
The management of Rwanda TVET Board extends its appreciation to both its staff and the staff of the TQUM Project for their efforts in coordinating these activities.

This training manual was developed:

Under Rwanda TVET Board (RTB) guiding policies and directives

Under Financial and Technical support of

COORDINATION TEAM

RWAMASIRABO Aimable

MARIA Bernadette M. Ramos

MUTIJIMA Asher Emmanuel

Production Team

Authoring and Review

NGENDAHIMANA THEOGENE

TUYISENGE JEAN LUC

Validation

UWANYIRIGIRA JANVIERE

NTAZINDA SYLVERIEN

Conception, Adaptation and Editorial works

HATEGEKIMANA Olivier

GANZA Jean Francois Regis

HARELIMANA Wilson

NZABIRINDA Aimable

DUKUZIMANA Therese

NIYONKURU Sylvestre

KWIZERA INGABIRE Diane

Formatting, Graphics, Illustrations, and infographics

YEONWOO Choe

SUA Lim

SAEM Lee

SOYEON Kim

WONYEONG Jeong

MANISHIMWE Marc

Financial and Technical support

KOICA through TQUM Project

TABLE OF CONTENT

ACRONYMS

CPU: Central Processing Unit

DCS: Distributed Control System

FBD: Function Block Diagram

HMI: Human-Machine Interface

I/O: Input/output

IEC: International Electro Technical Commission

IL: Instruction List

IOP: Input/output Processor

LAD: Ladder Diagram

PID: Proportional-Integral-Derivative (control)

PLC: Programmable Logic Controller

PPE: Personal Protective Equipment

RP: Rwanda Polytechnic

RTB: Rwanda TVET Board

RTU: Remote Terminal Unit

SCADA - Supervisory Control and Data Acquisition

SCL: Structured Control Language

ST: Structured Text

TQUM: TVET Quality management project

TVET: Technical and Vocational Education and Training

VFD: Variable Frequency Drive

INTRODUCTION

This trainee's manual includes all the knowledge and skills required in Networking and Internet Technologies specifically for the module of **"System Automation with PLC ".** Trainees enrolled in this module will engage in practical activities designed to develop and enhance their competencies. The development of this training manual followed the Competency-Based Training and Assessment (CBT/A) approach, offering ample practical opportunities that mirror real-life situations.

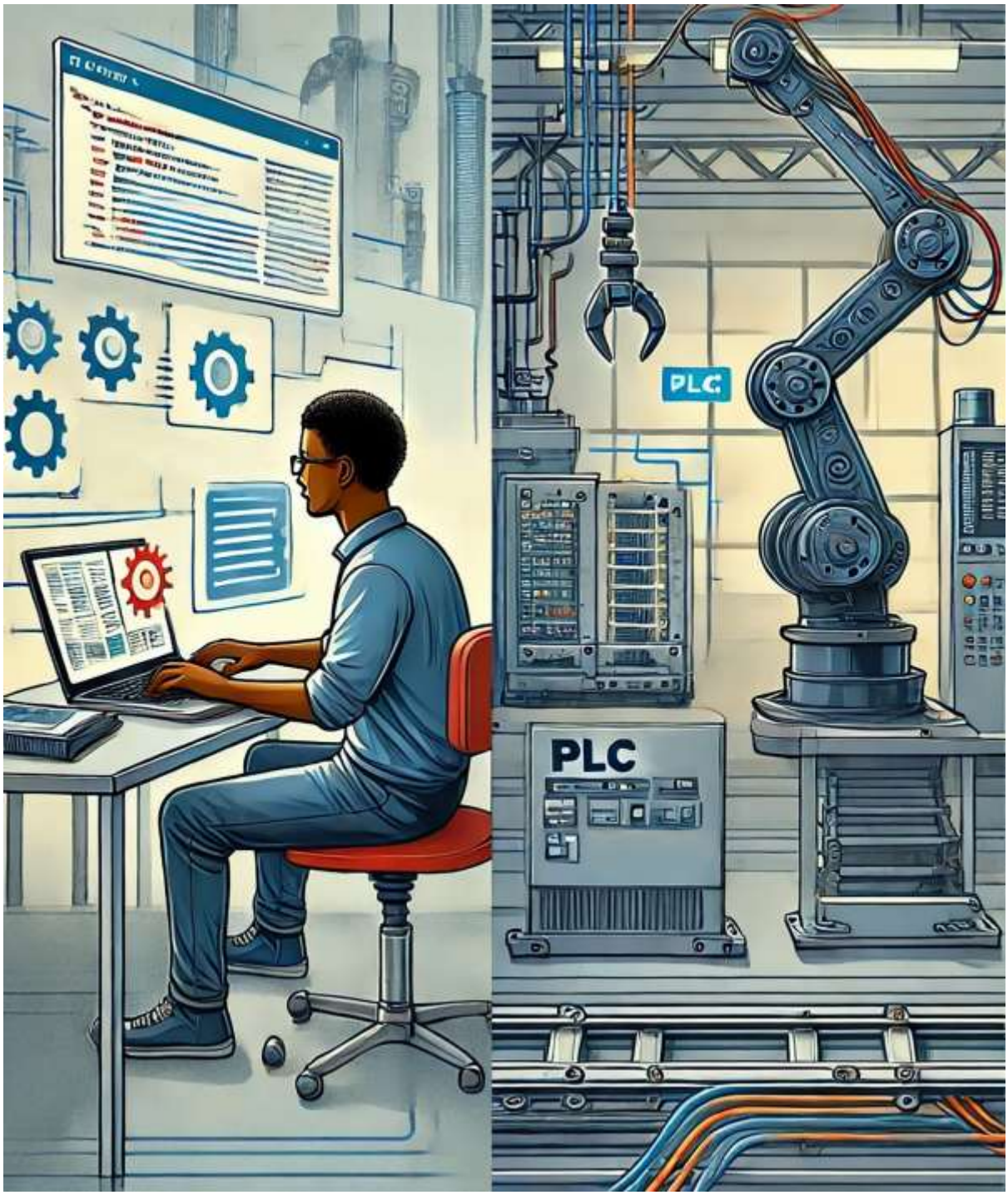
The trainee's manual is organized into Learning Outcomes, which is broken down into indicative content that includes both theoretical and practical activities. It provides detailed information on the key competencies required for each learning outcome, along with the objectives to be achieved.

As a trainee, you will start by addressing questions related to the activities, which are designed to foster critical thinking and guide you towards practical applications in the labor market. The manual also provides essential information, including learning hours, required materials, and key tasks to complete throughout the learning process.

All activities included in this training manual are designed to facilitate both individual and group work. After completing the activities, you will conduct a formative assessment, referred to as the end learning outcome assessment. Ensure that you thoroughly review the key readings and the 'Points to Remember' section.

MODULE CODE AND TITLE: CSASA501 SYSTEM AUTOMATION WITH PLC

Learning Outcome 1: Prepare automation system
Learning Outcome 2: Develop PLC Program
Learning Outcome 3: Deploy PLC program


PLC
PLC

Indicative contents

1.1 Identification of automation system requirements

1.2 Analysing Automation system

1.3 Selecting of tools, materials and equipment

1.4 Designing automation system

1.5 Installation of system hardware

Key Competencies for Learning Outcome 1: Prepare automation system

Knowledge	Skills	Attitudes
• Description of system automation • Description of Programmable Logic Controller (PLC) • Description of System analysis techniques • Description of system components • Identification of System requirements • Identification of symbols and notation • Description drawing templates	• Analysing Automation system • Selecting of tools, materials and equipment • Preparation of drawing environment • Installation of a drawing tool • Applying colour scheme and styling • Drawing system architecture • Installation of system hardware • Connecting hardware components • Testing hardware connection	• Being analyst. • Having Curiosity to explore and improve existing processes. • Being problem solver. • Being Resourcefulness • Having Precision in ensuring compatibility between selected components. • Having Confidence in making decisions. • Having Creativity in system design to optimize efficiency. • Being Patient and attention to detail when installing sensitive equipment.

Duration: 20 hrs

Learning outcome 1 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Describe properly the concept of system automation as used in various industries.
2. Describe properly programmable logic controllers (PLC) as used in system automation.
3. Analyse correctly system requirements based on existing design and user needs
4. Select properly tools, materials and equipment according to automation system requirements.
5. Design properly automation system based on analysis findings.
6. Install correctly system hardware according to manufacturer guidelines and automation system design.

Resources

Equipment	Tools	Materials
• Personal computer • PPE • PLC • Input and output modules • Power supply components • Multimeter • Contactors	• TIA Portal software • Screwdriver set • Wire strippers/cutters • Crimping pliers	• Electrical wires • Labels • Sensors • Actuators • Relays • Conduit • Zip ties • Internet

Indicative content 1.1: Identification of Automation System Requirements

Duration: 4 hrs

Theoretical Activity 1.1.1: Introduction to system automation

Tasks:

1: You are asked to answer the following questions:

i. What do you understand by system automation?
ii. Describe the types of system automation
iii. Identify he application of system automation
iv. Explain the working principles of system automation

2: Write your findings on provided paper/ flipchart

3: Present your findings to trainer or whole class

4: Ask for clarification if necessary

5: Read key reading 1.1.1 for more understanding

Key readings 1.1.1.: Introduction to system automation

Introduction to system automation

1. Definitions

a) System automation

System automation refers to the use of control systems, such as computers or robots, to handle processes or machines in industries to minimize human intervention. It plays a significant role in enhancing efficiency, accuracy, and productivity across various sectors such as manufacturing, healthcare, and logistics or **System automation** refers to the use of technology to perform tasks and processes with minimal human intervention. It involves the integration of software, hardware, and control systems to streamline operations, increase efficiency, and reduce errors.

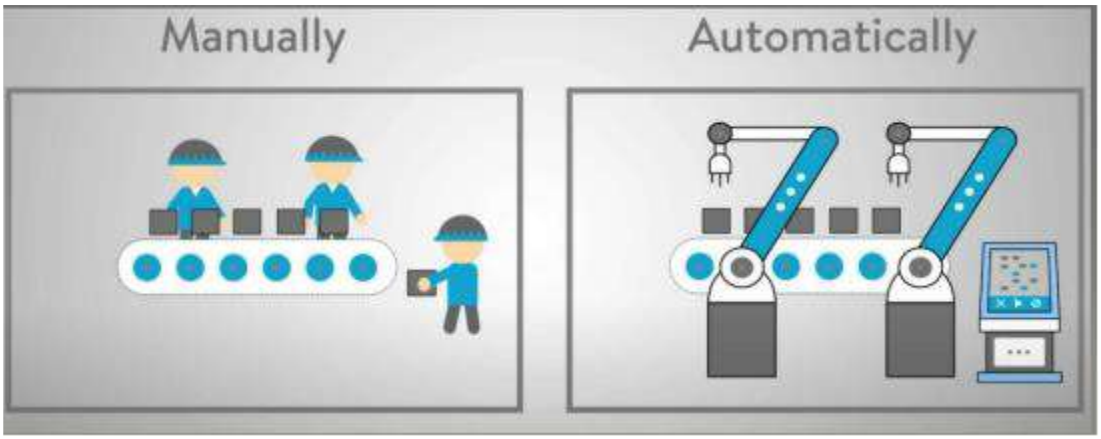
b) Automation

Automation is the technology by which processes are performed with minimal human assistance. It involves the integration of control systems to monitor and regulate devices and operations.

Automation is used across various industries, including manufacturing, telecommunications, healthcare, and finance, to improve productivity, quality, and consistency. An automation system combines sensors, controls, and actuators

to carry out functions with little to no human involvement. A Control system consisting of interconnected components is designed to achieve a desired purpose.

In the context of industrial and business processes, automation enables machines, software, and systems to operate autonomously or with minimal human oversight. It replaces manual tasks with automated workflows and decision-making processes



2. Types of Automation

Automation systems can be categorized into several types based on the complexity and flexibility of operations:

- **Fixed Automation:** Also known as hard automation, this involves high-volume production with dedicated equipment that is not easily reprogrammed. Examples include automated assembly lines and mass production systems.
- **Programmable Automation:** Suitable for batch production, this system allows for reprogramming to accommodate different production tasks. Common in flexible manufacturing e.g., manufacturing of different types of electronic devices).
- **Flexible Automation:** This is a type of programmable automation where equipment can easily switch between tasks with minimal changeover time, suited for small production batches. They are used in industries with frequent changes in products (e.g., manufacturing of custom-designed products).
- **Integrated Automation Systems (IAS):** In IAS, different machines and processes are interconnected, creating an automated network. The entire process from design, manufacturing, and quality control is fully automated (e.g., smart factories using IoT and AI).

3. Application of Automation Systems

Automation systems have widespread applications in various industries, including:

- **Manufacturing:** Automation improves production speed, product quality, and safety in industries like automotive, electronics, pharmaceuticals, and food processing.
- **Logistics and Warehousing:** Automated systems are used for sorting, packaging, and material handling to improve supply chain efficiency.
- **Building Management:** PLCs are used to control lighting, security systems, and heating, ventilation, and air conditioning (HVAC) systems in buildings.
- **Agriculture:** Automation is employed for tasks such as irrigation control, milking, and crop monitoring to improve productivity and resource management.
- **Healthcare:** Automated systems help in managing medical equipment, diagnostic tools, and even robotic surgeries.

4. Working Principles of Automation Systems Using PLC

The working principles of system automation are based on several key functions that enable machines or processes to operate automatically with minimal human intervention. These principles can be broken down into the following stages:

1. Input/Measurement (Sensing)

Automation systems begin by gathering real-time data from the environment or the system they are controlling. This data is collected through various **sensors** that detect changes in parameters such as temperature, pressure, flow, position, or speed. These sensors convert physical quantities into electrical signals for the system to process.

- **Examples of Sensors:**

- Temperature sensors (thermocouples, RTDs)
- Proximity sensors
- Pressure sensors
- Flow meters
- Light sensors

2. Processing/Decision-Making (Control Unit)

Once the input data is gathered, it is processed by the **controller** (e.g., PLC, DCS, or microcontroller), which evaluates the data against pre-programmed

instructions or control algorithms. The controller's role is to analyze the input and decide what actions are needed to achieve the desired system behavior.

- **Control Strategies:**
  - **Open-Loop Control:** A simple system where the controller sends commands to the actuators without considering feedback from the process. It does not correct for disturbances.
  - **Closed-Loop Control:** A more advanced system where feedback from the process is continuously sent back to the controller. The controller adjusts the system's operation to maintain the desired output (e.g., maintaining a set temperature, pressure, or speed).

3. Actuation (Output)

After the control unit processes the input data and makes decisions, it sends **commands** to the system's actuators to perform specific actions. These actions could involve starting or stopping a machine, adjusting valve positions, changing motor speeds, or engaging robotic arms.

- **Examples of Actuators:**

  - Electric motors
  - Hydraulic or pneumatic cylinders
  - Valves
  - Robotic arms
  - Relays and solenoids

4. Feedback Loop (Monitoring and Adjustments)

The system continuously monitors and adjusts its operations through feedback loops to ensure optimal performance.

Examples of Feedback

In a closed-loop system, the performance of the actuators and the overall process is continuously monitored through feedback sensors. The system compares the actual performance with the desired set points. If discrepancies are detected (such as a deviation in temperature or pressure), the controller adjusts the actuator commands to correct the system’s performance.

5. Communication

Automation systems often involve the **exchange of data** between different components and subsystems, which requires robust communication protocols.

This can happen via fieldbus networks, industrial Ethernet, or wireless communication. Effective communication ensures seamless operation between different components and allows for real-time data sharing.

6. Programming and Logic

At the heart of automation systems is the **programming logic** that controls the decision-making process. This logic is often implemented using programming languages (like ladder logic in PLCs) or advanced algorithms in higher-level control systems (e.g., PID controllers, machine learning algorithms in AI-driven automation systems). These programs determine how the system reacts to specific conditions and events.

- **Examples of Programming Methods:**

- Ladder Logic (commonly used in PLCs)
- Function Block Diagrams (FBD)
- Structured Text (ST)
- Sequential Function Charts (SFC)

7. Human-Machine Interface (HMI)

Most automation systems include a **user interface** to allow human operators to monitor the system and interact with it when necessary. The HMI displays critical information (like system status, performance metrics, alarms) and allows operators to input commands or adjust set points. It acts as a bridge between humans and machines, providing real-time visibility into the system's operation.

8. Safety Systems and Alarms

Automation systems often include built-in **safety mechanisms** to protect both the equipment and human operators. These systems monitor for abnormal conditions, such as overheating, overpressure, or mechanical failures, and take corrective actions (e.g., emergency shutdown, tripping a circuit breaker). They may also trigger alarms to alert operators to take necessary action.

Example: Automated Temperature Control System working principle

1. **Input/Measurement:** A temperature sensor measures the current temperature in a furnace.
2. **Processing/Decision-Making:** A PLC compares the measured temperature with the desired set point temperature.

3. **Actuation:** If the furnace temperature is below the set point, the PLC sends a signal to a relay that powers the heater.

4. **Feedback Loop:** The temperature sensor continuously provides feedback. When the set point is reached, the PLC switches off the heater.

5. **Communication:** The PLC communicates the temperature data to the HMI, where operators can monitor it in real time.

6. **Safety System:** If the furnace overheats, a safety relay trips to prevent damage or accidents

Theoretical Activity 1.1.2: Description of Programmable Logic Controller (PLC)

Tasks:

1: You are requested to answer the following questions

i. What do you understand by Programmable Logic Controller?
ii. Describe the types of Programmable Logic Controller
iii. Identify he application of Programmable Logic Controller
iv. Identify the Characteristics of Programmable Logic Controller

2: Write your findings on provided paper/ flipchart

3: Present yours findings to trainer or whole class

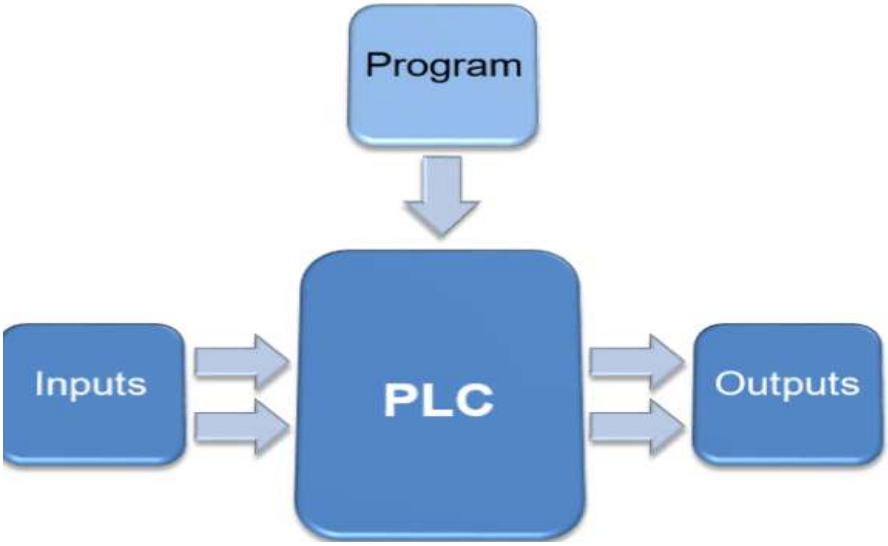
4: Ask for clarification if necessary

5: Read key reading 1.1.2 for more understanding

Key readings 1.1.2: Description of Programmable Logic Controller (PLC)

1. Definition

A **Programmable Logic Controller (PLC)** is an industrial digital computer designed for the automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures. PLCs are used in various industries due to their robustness, ease of programming, and real-time control capability. They monitor inputs, process the logic programmed by the user, and output commands to control devices like motors, lights, pumps, and valves.



2. Types of PLCs

PLCs can be classified based on various factors such as size, input/output capacity, functionality, and structure. The common types are:

✓ **Compact PLC (Fixed PLC):**

Description: All components (CPU, input/output modules, and power supply) are integrated into a single unit.

Typically used for smaller systems with limited input/output (I/O) points.

Applications: Ideal for small to medium-sized applications with limited I/O requirements. **Advantages:** Easy to install, cost-effective, and typically sufficient for simple automation tasks.

Examples: Siemens S7-1200, Allen-Bradley MicroLogix

Micro PLCs for small automation tasks.

✓ **Modular PLC:**

Description: Consist of separate modules for the processor, power supply, and I/O, which can be added or removed as needed.

Applications: Suitable for larger, more complex systems where scalability and customization are essential.

Advantages: Highly flexible and expandable, allowing for custom configurations to meet specific needs.

Examples: Siemens S7-300/400, Allen-Bradley ControlLogix

✓ **Rack-Mounted PLC:**

Description: Similar to modular PLCs but mounted in a rack, allowing for easier management of multiple modules.

Applications: Used in large-scale industrial applications requiring numerous I/O points and complex control.

Advantages: High scalability and manageability, suitable for extensive automation systems.
Examples: Allen-Bradley PLC-5, Mitsubishi MELSEC-Q Series.

✓ **Soft PLC:**

Description: PLC functions are performed by software running on a general-purpose industrial PC.
Applications: Applications that require advanced computing power, data processing, and integration with other IT systems.
Advantages: High processing power, flexibility, and ease of integration with other software systems.
Examples: Beckhoff Twin CAT, Siemens Soft PLC.

✓ **Safety PLCs**

Description: Designed specifically for safety-critical applications, incorporating features that ensure fail-safe operation.
Applications: Used in industries where safety is paramount, such as automotive, aerospace, and chemical industries.
Advantages: Meets stringent safety standards, provides redundancy, and ensures reliable operation under fault conditions.
Examples: Siemens S7-1500F, Allen-Bradley GuardLogix.

✓ **Distributed or Networked PLCs**

Description: Comprise multiple PLCs networked together to control different parts of a complex system.
Applications: Large manufacturing plants, distributed control systems (DCS), and complex automation environments.
Advantages: Enhanced flexibility, redundancy, and scalability for controlling distributed processes.
Examples: Systems using Profibus, Ethernet/IP, Modbus TCP for interconnecting multiple
PLCs.

✓ **Nano PLCs**

Description: Ultra-compact PLCs designed for very simple automation tasks with minimal I/O requirements.
Applications: Simple, low-cost applications such as small machinery, home automation, and simple repetitive tasks.
Advantages: Extremely compact, cost-effective, and easy to deploy.
Examples: Allen-Bradley Micr0800.

✓ **Specialty PLCs**

Description: PLCs designed for specific applications, incorporating specialized features and functions.
Applications: Industry-specific applications like robotics, motion control, HVAC systems, and more.
Advantages: Optimized for specific tasks, often integrating specialized I/O modules and functions.
Examples: Omron NJ/NX Series for motion control, Siemens S7-1500T for high-speed applications.

3. Applications of PLCs (Programmable Logic Controllers)

PLCs are versatile devices used in many industries to automate processes, control machinery, and ensure efficient, safe, and precise operation of systems. Here are some key application areas of PLCs:

✓ Manufacturing and Assembly Lines

- **Automated Assembly Lines:** PLCs are used to control assembly operations, ensuring that products are manufactured in the correct sequence with minimal human intervention.
- **Robotic Control:** In many industries, PLCs control robots that perform welding, painting, material handling, and packaging.
- **Packaging Machines:** PLCs manage the flow of materials, ensure accurate product packaging, and control labelling machines.

Example: In automotive manufacturing, PLCs control conveyor belts, robotic arms, and welding machines, ensuring cars are assembled efficiently and safely.

✓ Process Control in Chemical and Food Industries

- **Chemical Processing:** PLCs monitor and regulate temperature, pressure, and flow in reactors and distillation columns, ensuring product quality and safety.
- **Food and Beverage Production:** PLCs are used to control mixing, heating, cooling, and packaging processes in food and beverage manufacturing.
- **Batch Processing:** PLCs handle batch control, automating the production of chemicals, pharmaceuticals, or beverages with precise timing and ingredient measurement.

Example: A PLC-controlled brewery system automates the mixing, fermentation, and bottling processes, ensuring consistent beer production.

✓ **Water Treatment and Wastewater Management**

- **Water Treatment Plants:** PLCs control the filtration, chemical dosing, and distribution of clean water in water treatment plants.
- **Wastewater Treatment:** PLCs manage aeration, sludge processing, and effluent discharge, ensuring that environmental standards are met.
- **Pumping Stations:** PLCs control pumps and valves to regulate the flow and distribution of water or wastewater across a network.

Example: In a municipal water treatment plant, PLCs automate the purification process, ensuring a constant supply of clean water to residents.

✓ **Energy Management and Power Generation**

- **Power Plants:** PLCs control turbines, generators, and switchgear in power plants, ensuring that energy is produced and distributed efficiently.
- **Renewable Energy:** In wind farms and solar power plants, PLCs are used to control power generation, monitor equipment, and integrate renewable energy into the power grid.
- **Energy Monitoring:** PLCs are used to monitor energy consumption in factories and large buildings, helping optimize energy use and reduce costs.

Example: In a solar power plant, PLCs track sunlight, control solar panel angles, and manage power distribution to the grid.

✓ **HVAC (Heating, Ventilation, and Air Conditioning) Control**

- **Building Automation:** PLCs control HVAC systems, regulating temperature, humidity, and air quality in large buildings, ensuring comfort and energy efficiency.
- **Industrial HVAC:** In industrial facilities, PLCs manage complex HVAC systems, ensuring optimal conditions for machinery and workers.

Example: PLCs in a commercial building automate the HVAC system, adjusting heating and cooling based on occupancy and external temperature conditions.

✓ **Automotive Industry**

- **Production Line Automation:** PLCs manage processes like welding, painting, and assembly in car manufacturing plants.
- **Material Handling:** They control conveyors, elevators, and automated guided vehicles (AGVs) in car factories.
- **Testing Systems:** PLCs are used in testing and validation systems to ensure vehicle components meet safety and performance standards.

Example: PLCs control robotic arms for precision welding and painting in automotive factories, improving speed and accuracy.

✓ **Material Handling and Logistics**

- **Conveyor Systems:** PLCs control the movement of materials on conveyor belts in warehouses and manufacturing plants.
- **Automated Storage and Retrieval Systems (ASRS):** PLCs manage inventory by controlling cranes and automated systems that move products into storage.
- **Sorting and Packaging:** In logistics centers, PLCs automate sorting and packaging tasks, ensuring goods are processed and shipped efficiently.

Example: PLCs in Amazon's fulfilment centers control conveyor belts, robots, and sorting machines to handle millions of packages daily.

✓ **Mining and Metals**

- **Ore Processing:** PLCs control crushing, grinding, and flotation processes in mining operations.
- **Smelting and Refining:** They regulate temperature, pressure, and chemical processes in metal extraction and refining.
- **Conveyor and Transport Systems:** PLCs manage conveyors that transport materials in mines, ensuring safe and efficient movement.

Example: In a copper mining plant, PLCs automate ore crushing, grinding, and flotation, reducing manual labor and improving output.

✓ **Oil and Gas Industry**

- **Pipeline Monitoring and Control:** PLCs are used to monitor and control the flow of oil and gas through pipelines, ensuring safety and preventing leaks.
- **Refinery Automation:** They control complex chemical processes in refineries, such as distillation and cracking, to convert crude oil into usable products.
- **Offshore Platforms:** PLCs manage drilling, safety systems, and communication equipment on offshore oil platforms.

Example: PLCs on an offshore oil rig control drilling operations, monitor safety equipment, and ensure smooth communication with onshore teams.

✓ **Transportation Systems**

- **Traffic Signal Control:** PLCs automate traffic lights, ensuring smooth traffic flow and reducing congestion in cities.
- **Railway Systems:** They control signals, track switches, and train movement, ensuring safety and efficiency.

- **Airport Automation:** PLCs manage baggage handling systems, lighting, and security systems in airports.

Example: In airports, PLCs manage the automated baggage handling systems, reducing delays and improving efficiency in moving luggage.

✓ **Safety Systems**

- **Emergency Shutdown Systems (ESD):** PLCs control emergency shutdown systems in hazardous industries, such as oil and gas, to prevent accidents during system malfunctions.
- **Fire and Security Systems:** PLCs are used to monitor and control fire alarms, sprinkler systems, and access control in large buildings or industrial plants.

Example: In a chemical plant, PLCs are part of the safety system, ensuring emergency shutdowns are triggered if temperature or pressure exceed safe limits.

✓ **Agriculture and Irrigation**

- **Automated Irrigation Systems:** PLCs manage irrigation schedules and water distribution based on soil moisture, weather conditions, and crop requirements.
- **Greenhouse Automation:** PLCs control temperature, humidity, light, and ventilation in greenhouses to optimize plant growth.
- **Fertilizer Systems:** They automate the mixing and application of fertilizers and pesticides in large-scale farming.

Example: In modern smart farms, PLCs control automated irrigation systems, applying the right amount of water and fertilizer to crops based on real-time data.

4. Characteristics of PLCs (Programmable Logic Controllers)

PLCs possess several key characteristics that make them essential in industrial automation and process control applications. These characteristics are crucial for ensuring that PLCs can operate efficiently and reliably in demanding environments. Below are some of the most important characteristics of PLCs:

✓ **Programmability**

- A PLC is a digital device that can be programmed to perform specific tasks. Its programming can be done using standard languages like Ladder Logic, Structured Text, Functional Block Diagrams, or Sequential Function Charts (SFC).
- Users can easily modify the control logic without needing to change the physical wiring of the system.

✓ **Real-Time Operation**

- PLCs operate in real-time, meaning they can process inputs, execute the control program, and respond with outputs almost instantaneously.
- This ensures that processes are controlled in real-time, minimizing delays and ensuring proper synchronization of system components.

✓ **Ruggedness and Durability**

- PLCs are designed to withstand harsh industrial environments, including extreme temperatures, vibrations, dust, humidity, and electrical noise.
- They are built with robust hardware that ensures reliability, even under adverse conditions.

✓ **Reliability**

- PLCs are highly reliable, with long operational lifetimes and minimal downtime. They are capable of running continuously for extended periods without failure.
- Redundancy features can be incorporated into critical applications to enhance reliability further.

✓ **Scalability and Flexibility**

- PLCs can be scaled to match the complexity of the system they are controlling. Modular PLCs, in particular, allow for the addition or removal of modules to expand input/output capacity.
- They are flexible and adaptable to different system requirements, from small single-machine control to large, complex processes.

✓ **Input/output (I/O) Handling**

- PLCs interface with a wide variety of digital and analogue inputs and outputs, allowing them to interact with sensors, actuators, and other devices.
- They can monitor data from sensors and execute control signals to motors, valves, relays, and other actuators.

✓ **Communication Capabilities**

- Modern PLCs are equipped with communication interfaces to connect with other devices or systems, such as Human-Machine Interfaces (HMIs), Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other PLCs.
- PLCs support various communication protocols like Ethernet/IP, Modbus, Profibus, and Device Net for seamless data exchange and integration with higher-level systems.

✓ **High-Speed Processing**

- PLCs have fast processing speeds, enabling them to execute complex control programs and manage large volumes of I/O data in real time.
- This ensures that they can handle time-sensitive applications, such as motion control, robotic systems, and high-speed manufacturing processes.

✓ **Memory**

- PLCs have memory that stores the control program, data from sensors, and other variables needed for operation.
- This memory is non-volatile, meaning it retains data and the control program even in the event of a power failure.

✓ **Easy Troubleshooting and Maintenance**

- PLCs provide diagnostic tools that simplify troubleshooting and maintenance. Error codes, alarms, and diagnostic messages are available to help operators quickly identify and resolve issues.
- Many PLCs allow for real-time monitoring of inputs, outputs, and internal variables, making it easier to detect faults or inefficiencies.

✓ **Modular Design**

- Many PLCs have a modular design that allows users to customize the system by adding or removing components such as CPU modules, I/O modules, communication interfaces, and power supplies.
- This modularity offers flexibility and simplifies both system expansion and maintenance.

✓ **Data Handling and Monitoring**

- PLCs can collect, process, and store data from the system, which can be used for monitoring performance, generating reports, and performing predictive maintenance.
- PLCs can interface with databases and enterprise systems to provide operational data for further analysis and optimization.

✓ **Event and Fault Logging**

- PLCs often include logging capabilities to record events and fault conditions. This data is useful for diagnosing issues, improving system performance, and maintaining a historical record of operations.
- Event logs can be reviewed to identify trends or patterns that may require system adjustments or preventative maintenance.

✓ **Safety Features**

• Many PLCs are equipped with built-in safety features, including emergency stop controls, fault detection, and safe shutdown procedures.

PLCs used in safety-critical applications (e.g., Emergency Shutdown Systems) must adhere to stringent safety standards, such as IEC 61508 or ISO 13849

Points to Remember

- System automation refers to the use of control systems, such as computers or robots, to handle processes or machines in industries to minimize human intervention.
- Automation systems can be categorized into Fixed Automation, Programmable Automation, Flexible Automation and Integrated Automation Systems.
- Automation should be used across various industries, including manufacturing, Logistics and Warehousing, Building Management, Agriculture, Healthcare, etc. to improve productivity, quality, and consistency
- Automation systems work by integrating control mechanisms, sensors, actuators, and communication networks. These systems operate based on real-time feedback and control logic to achieve high efficiency, safety, and scalability in industrial or commercial setting
- By automating systems, industries can optimize productivity, enhance safety, reduce operational costs, and increase precision, creating a more reliable and efficient environment.
- A Programmable Logic Controller (PLC) is an industrial computer-based device used for controlling processes or machines in the manufacturing and production industry
- PLCs can be classified based on various factors such as size, input/output capacity, functionality, and structure. Some types include Compact PLC, Modular PLC, Rack-Mounted PLC, Soft PLC, Safety PLCs, etc.
- PLCs are versatile devices used in many industries to automate processes, control machinery, and ensure efficient, safe, and precise operation of systems. Some key application areas of PLCs are Manufacturing and Assembly Lines, Process Control in Chemical and Food Industries, Water Treatment and Wastewater Management, Energy Management and Power Generation, Automotive Industry, etc.
- PLCs possess several key characteristics that make them essential in industrial automation and process control applications. These characteristics are crucial for ensuring that PLCs can operate efficiently and reliably in demanding environments.

Most important characteristics of PLCs include Programmability, Real-Time Operation, Reliability, Scalability and Flexibility, Communication Capabilities, etc.

Application of learning 1.1.

As an automation technician at a food processing plant, your primary responsibility is to automate the packaging line to reduce waste and enhance accuracy. You start by thoroughly analysing the existing packaging process to identify inefficiencies. Based on your findings, you recommend integrating Programmable Logic Controllers (PLCs) to improve control over the packaging machinery. You then define key requirements for the new system, focusing on accessibility and maintainability to minimize downtime and facilitate quick troubleshooting by operators

Indicative content 1.2: Analysing Automation System

Duration: 4 hrs

Theoretical Activity 1.2.1: Description of automation system analysis

Tasks:

1: You are invited to answer the following questions related to Analysing the existing System:

i. What does automation system analysis means?
ii. Identify automation system analysis techniques
iii. List automation system components
iv. Analyze system requirements

2: Present your findings to the whole class and Trainer

3: Ask your trainer for any clarification if any.

4: You are asked to read the key readings 1.2.1 for more clarifications

Key readings 1.2.1.: Description of automation system analysis

1. Introduction to automation system analysis

Analysing an automation system involves a detailed examination of the existing system, its components, user needs, and technical requirements. This process ensures that the automation system is optimized for functionality, performance, and usability.

Analysing the Existing System

Before making changes or implementing new automation systems, it is crucial to analyse the existing setup. This analysis helps to identify inefficiencies, gaps, or opportunities for improvement. Key steps include:

- **Evaluating System Performance:** Measure how well the current system meets operational goals. This includes evaluating cycle times, process bottlenecks, error rates, and downtime.
- **Assessing System Architecture:** Understand how the different components of the system interact, including hardware, software, communication protocols, and data flow.
- **Identifying Redundancies:** Look for duplicate processes, unused features, or unnecessary complexity in the existing system.
- **Monitoring Data Flow and Communication:** Ensure data is accurately transmitted between sensors, controllers, and actuators in a timely manner.

2. Automation System Analysis Techniques

System analysis is essential for understanding the current functionality and identifying areas for automation improvement. In **PLC-based system automation**, the following techniques are commonly used:

✓ **Flowcharts and Diagrams:**

Represent the workflow or process in a graphical format, showing the sequence of operations, decision points, and process dependencies.

Example: Process flow diagrams, network diagrams.

✓ **Data Flow Diagrams (DFD):**

Illustrate how data flows within the system, highlighting data sources, destinations, and intermediate processes.

Useful for understanding how information is processed and stored.

✓ **System Modelling:**

Use mathematical or simulation models to predict how the system will behave under different conditions. This helps in optimizing performance and identifying potential issues.

✓ **SWOT Analysis (Strengths, Weaknesses, Opportunities, Threats):**

Helps identify the strong and weak points of the existing automation system and areas for improvement or innovation.

✓ **Root Cause Analysis:**

1. A method used to identify the root cause of problems or inefficiencies in the system, often employing techniques like the Diagrams.

3. Description of Automation System Components

A typical automation system is made up of several key components, each with a specific role:

A. Sensors:

Devices that detect changes in the environment (e.g., temperature, pressure, position) and send data to controllers.

B. Actuators:

Components like motors, valves, or solenoids that perform actions in response to control signals.

C. Programmable Logic Controller (PLC):

The brain of the automation system. It processes input signals from sensors and executes logic to control actuators.

D. Human-Machine Interface (HMI):

The interface between operators and the automation system. HMIs provide visual feedback and allow operators to interact with the system.

E. Communication Networks:

Protocols and systems (e.g., Ethernet, Modbus, Profibus) that facilitate data exchange between system components.

F. SCADA Systems:

Supervisory Control and Data Acquisition systems used to monitor and control large-scale processes. SCADA allows real-time data collection and process visualization.

G. Power Supply:

Provides stable and reliable power to the automation components, ensuring continuous operation.

H. Safety Systems:

Systems such as emergency stops, interlocks, and fail-safes designed to protect operators and equipment.

4. Identifying User's Needs

Understanding the needs of the end-users is essential for system design. This involves:

- ✓ **Engaging Stakeholders:** Conduct interviews, surveys, or focus groups with users, operators, and management to gather requirements.
- ✓ **Use Case Scenarios:** Create detailed use case scenarios that describe how users interact with the system under various conditions.
- ✓ **Task Analysis:** Break down user tasks to understand their complexity and how they can be automated or simplified.
- ✓ **Ergonomics and User Comfort:** Ensure the system is designed for ease of use, comfort, and safety.

5. System Requirements Analysis

System requirements analysis defines the specifications for the automation system. This ensures that the system meets the expectations and needs of the business and users.

5.1 Performance Requirements:

✓ **Speed and Response Time:**

How quickly the system responds to inputs and executes actions.

Example: A PLC that can process signals in milliseconds to ensure real-time control.

✓ **Capacity:**

The system's ability to handle the volume of operations, number of I/O points, and data storage.

Example: A system that manages multiple conveyor belts simultaneously.

✓ **Throughput:**

How much work the system can perform within a given time.

Example: The number of products an assembly line can complete per hour.

✓ **Scalability:**

The ability of the system to grow and handle increased workloads without significant performance degradation

Example: Adding additional sensors or actuators as production expands.

5.2 Usability Requirements

Usability requirements focus on how easily users can interact with the system and how effectively they can achieve their goals. These requirements include:

✓ **User Interface Design:**

The system should have an intuitive and user-friendly interface that allows users to navigate and operate it without extensive training.

✓ **Learnability:**

The system should be easy for new users to learn and understand quickly. This includes providing adequate documentation, tutorials, or help features.

✓ **Accessibility:**

Ensuring that the system is usable by people with diverse abilities and disabilities. This may involve supporting assistive technologies or providing alternative input methods.

✓ **Error Handling:**

The system should provide clear error messages and guidance for recovery when users make mistakes, helping them understand what went wrong and how to fix it.

✓ **Customization:**

Users should have the option to customize settings, displays, and workflows to suit their individual preferences, improving overall user satisfaction.

✓ **Feedback Mechanisms:**

The system should provide timely feedback to users on their actions, such as confirmations for commands, alerts for issues, and status updates on processes.

5.3 Recoverability Requirement

Recoverability requirements address how the system can recover from failures or errors. For PLC-based automation, important aspects include:

✓ **Backup Systems:**

The PLC should have mechanisms to back up configurations and programs to prevent loss of data in case of failures.

✓ **Error Recovery:**

The system must be able to identify faults and either correct them automatically or revert to a safe state. This includes the ability to restart processes without manual intervention when possible.

✓ **Data Logging:**

Keeping logs of system performance and errors helps in diagnosing issues and enables easier recovery after an incident.

5.4 Maintainability Requirement

Maintainability refers to the ease with which an automation system can be maintained, repaired, or updated throughout its operational life. It includes both preventive and corrective maintenance, ensuring minimal disruption and downtime.

Key Maintainability Requirements:

✓ **Modular Design:**

Components should be designed as modular units, making it easier to replace or upgrade specific parts without affecting the entire system.

Example: In a PLC system, having swappable I/O modules allows quick replacement without halting operations.

✓ **Fault Diagnosis and Troubleshooting:**

The system should include diagnostic tools to quickly identify and locate faults.

Example: Self-diagnostic capabilities, error codes, or troubleshooting guides that help maintenance staff pinpoint and resolve issues efficiently.

✓ **Documentation:**

Detailed maintenance manuals, schematics, and system architecture should be available to facilitate quick repairs and upgrades.

Example: Well-documented service logs, maintenance checklists, and detailed wiring diagrams.

✓ **Remote Monitoring and Maintenance:**

Systems should support remote diagnostics, monitoring, and updates to allow maintenance personnel to address issues without being physically present.

Example: A SCADA system that allows technicians to monitor and control the system remotely to diagnose problems.

✓ **Scheduled Maintenance Alerts:**

The system should track operating hours or cycles and automatically notify when maintenance is due.

Example: Automated alerts for routine inspections or part replacements, such as for motors or sensors.

✓ **Spare Parts Availability:**

Key components should be readily available and interchangeable to reduce downtime during repairs.

Example: Stocking essential spare parts like sensors, relays, and I/O cards for quick replacement.

✓ **Serviceability:**

Critical components should be easy to access for inspection, cleaning, and repairs.

Example: Equipment panels that open without tools, allowing easy access to wiring and components for maintenance.

✓ **Version Control and Updates:**

The system should support easy software updates and version control for firmware and application changes.

Example: A central software management system for rolling out updates or patches to multiple PLCs in a network.

5.5 Accessibility Requirement

Accessibility ensures that the automation system can be effectively used by a wide range of users, including those with disabilities. It involves designing systems that

are user-friendly, accommodating diverse abilities, and providing intuitive interfaces.

Key Accessibility Requirements:

✓ **User Interface (UI) Design:**

The UI should be designed for ease of use, with clear navigation, large buttons, and simple controls.
Example: An HMI with customizable font sizes, high-contrast colors, and touchscreen capabilities that can be adapted to the needs of different users.

✓ **Compliance with Accessibility Standards:**
The system should comply with international accessibility standards, such as the **Americans with Disabilities Act (ADA)** or **Web Content Accessibility Guidelines (WCAG)**, where applicable. o **Example:** Ensuring that software interfaces are usable by individuals with visual or hearing impairments.

✓ **Multi-Language Support:**
The system should offer multi-language support to accommodate users with different linguistic backgrounds.
Example: An HMI interface that can switch between different languages based on user preferences.

✓ **Voice Control and Audio Feedback:**
Providing voice control or audio feedback for users who may have limited mobility or visual impairments.
Example: Voice commands to control machines or receive audible alerts and system notifications.

✓ **Physical Accessibility:**
System controls, panels, and input devices should be positioned at heights and angles that accommodate a wide range of users, including those in wheelchairs.
Example: Adjustable height control panels or moveable HMI displays.

✓ **Assistive Technologies Integration:**
The system should support or integrate with assistive technologies such as screen readers, alternative input devices (e.g., Braille keyboards), or speech recognition software.
Example: HMIs and control systems that can interface with screen readers for visually impaired users.

✓ **Customizable User Profiles:**

Users should be able to personalize the interface to meet their specific needs, including adjusting controls, color schemes, or input methods.
Example: Different user profiles with tailored settings for operators with varying skill levels or accessibility needs.

✓ **Accessible Documentation:**
All user manuals, guides, and documentation should be available in accessible formats such as large print, Braille, or audio versions.
Example: Providing user manuals in multiple formats, including PDFs with text to-speech compatibility or Braille-printed versions for visually impaired users.

By addressing maintainability and accessibility requirements, automation systems can operate efficiently, accommodate diverse users, and remain easy to update and repair throughout their lifecycle

Points to Remember

- System analysis is essential for understanding the current functionality and identifying areas for automation improvement. In PLC-based system automation, the following techniques are commonly used: Flowcharting, Data Flow Diagrams (DFD), Process Mapping, Fault Tree Analysis.
- In any PLC-based automated system, understanding the components is critical to analysing and improving the system. The key components include: Input Devices like sensors, push buttons, and switches and central control unit(PLC), Output Devices like motors, actuators, lights, or valves and Human-Machine Interface (HMI), Communication Modules.
- In system automation using PLCs, identifying the user's needs helps ensure that the automated system is designed to solve real operational challenges. Key steps in identifying user needs include Process Improvement, Safety Requirements, Performance and Efficiency, Flexibility and Scalability, Integration Needs.

Application of learning 1.2: Analysing Automation system

You are tasked with designing a PLC-based automation system for a packaging plant that moves products along a conveyor belt.

i. Which adjustments could you make to ensure the system meets the performance requirement of adjusting speed without products falling off the belt?
ii. How would you design the HMI to make sure the system is easy to use for operators who need to start, stop, and adjust the speed?
iii. What features would you implement to ensure the system can recover from an unexpected motor stop?

Indicative content 1.3: Selecting of Tools, Materials and Equipment

Duration: 4hrs

Theoretical Activity 1.3.1: Identification of tools, materials and equipment

Tasks:

1: You are requested to answer the following questions

i. Define the following terms:
   a. Tool
   b. Material
   c. Equipment

ii. Identify the examples and functions of tool material and equipment used in automation system

2: Write your findings on provided paper/ flipchart

3: Present your findings to trainer or whole class

4: Ask for clarification if necessary

5: Read key reading 1.3.1

Key readings 1.3.1: Identification Tools, Materials, and Equipment used in Automation System

In **PLC-based automation systems**, selecting the right tools, materials, and equipment is essential for designing, installing, and maintaining an efficient and reliable automated system. Below is a breakdown of each category and its relevance to system automation using PLCs.

1. Tools for System Automation Using PLCs

When working with PLC systems, various tools are required for installation, programming, testing, and maintenance. Some essential tools include:

- **Programming Tools:**

✓ **PLC Programming Software:** Software such as Siemens TIA Portal, Allen-Bradley Studio 5000, or Mitsubishi GX Works is necessary to write and test programs using languages like **Ladder Logic** or **Function Block Diagrams** (FBD).

✓ **PC or Laptop:** A computer is required for programming the PLC and uploading/downloading programs to the PLC controller.

- **Wiring and Electrical Tools:**

✓ **Screwdrivers:** To connect and disconnect wiring from terminals on the PLC, sensors, and actuators.

✓ **Wire Strippers:** Used to strip the insulation off wires before connecting them to terminals or components.

✓ **Multimeter:** For testing voltage, continuity, and resistance in PLC circuits during installation and troubleshooting.

✓ **Crimping Tool:** To attach connectors to wires securely when connecting sensors, actuators, or power supply units.

- **Diagnostic Tools:**

✓ **PLC Simulator or Tester:** To simulate PLC programs before running them on the actual hardware, ensuring that the logic works as expected.

✓ **Logic Probes and Oscilloscopes:** For testing and diagnosing digital signals and verifying the operation of I/O modules.

2. Materials for System Automation Using PLCs

Materials refer to consumables and components that are used to build and maintain a PLC-based automation system. These materials are critical for setting up the electrical connections and ensuring proper operation. Key materials include:

- **Wiring and Cables:**

✓ **Control Wiring:** Used to connect sensors, actuators, and other devices to the PLC's input/output terminals. These wires must be of the right gauge to handle the electrical load.

✓ **Power Cables:** These provide power to the PLC, sensors, and actuators. They should be selected based on the voltage and current requirements of the system.

✓ **Network Cables (Ethernet, RS-485):** Required for communication between the PLC and other devices, such as HMIs or other PLCs, in an integrated automation system.

- **Connectors and Terminals:**

✓ **Terminal Blocks:** Used to connect wiring to the PLC I/O modules securely.

✓ **Wire Connectors:** Used to join wires together or attach them to terminals on PLC components.

- **Electrical Components:**

✓ **Relays and Contactors:** These are used to control high-power devices based on the low-power signals from the PLC.

✓ **Fuses and Circuit Breakers:** Provide protection to the PLC system from overcurrent conditions and electrical faults.

3. Equipment for System Automation Using PLCs

In addition to tools and materials, specialized equipment is required to ensure the successful operation and control of PLC-based automation systems. Important equipment includes:

- **Programmable Logic Controller (PLC):**

✓ The core device that executes the programmed logic, processes inputs, and controls outputs. Selecting the appropriate PLC depends on factors such as the number of I/O points, processing speed, and network communication capabilities.

- **Input Devices:**

✓ **Sensors:** Devices like temperature sensors, proximity sensors, pressure sensors, and switches are used to gather data from the environment and provide input signals to the PLC.

✓ **Push Buttons and Switches:** Used to manually start, stop, or change the operation of automated processes.

- **Output Devices:**

✓ **Actuators:** Devices like solenoids, motors, and pneumatic or hydraulic actuators are controlled by the PLC to perform physical actions in the system.

✓ **Motors:** Commonly used to drive conveyors, fans, or pumps in industrial automation.

✓ **Valves:** Controlled by the PLC to regulate the flow of liquids or gases in a process.

- **Human-Machine Interface (HMI):**

✓ The HMI provides operators with a user-friendly way to monitor and control the PLC system. HMIs can display real-time data, provide control options, and offer diagnostic information.

- **Power Supply Units (PSUs):**

✓ Provides a stable and regulated power source to the PLC and connected devices. Selecting the right PSU is important for ensuring reliable operation, especially in systems with high-power devices.

- **Communication Modules:**

✓ **Network Routers, Hubs, and Switches:** These are required in large automation systems where multiple PLCs and devices are interconnected through industrial communication protocols (e.g., Modbus, Ethernet/IP).

- **Test and Measurement Equipment:**

✓ **Power Analyzers and Meters:** To monitor the performance of the automation system and ensure it is operating within specified parameters.

✓ **Calibration Equipment:** For ensuring that sensors and measurement devices are functioning accurately.

Theoretical Activity 1.3.2: Selecting tools, materials and equipment

Tasks:

1: Referring to the key reading 1.3.1 you are requested to perform the given task. The task should be done individually

As an automation technician, you have been assigned to select tools, materials and equipment that will be used in designing automation system.

2: Present your answer to the class or colleagues

3: Ask your colleagues and trainer for any clarification if any.

4: Read the key readings 1.3.2 in trainee's manuals

Key readings 1.3.2: Selecting tools, materials and equipment

In **PLC-based automation systems**, selecting the right tools, materials, and equipment is essential for designing, installing, and maintaining an efficient and reliable automated system

When selecting tools, materials, and equipment for system automation, you can consider things like:

✓ **Automation type**: Whether you're looking for robotic process automation (RPA), artificial intelligence (AI), machine learning (ML), or another type of automation

✓ **Project needs**: What type of application you're working with, what platforms are supported, and what types of testing are required

✓ **Mobility**: Whether the tools you need are lightweight, compact, and secure enough to be moved around easily

✓ **Materials**: The materials you're working with and the best way to handle them

✓ **Vendor support**: The reputation of the vendor and the automation resources they offer

✓ **User-friendliness**: How easy it is for employees to use the new technology and integrate it into their workflow

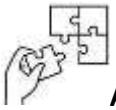
✓ **Security**: How secure the data collection is and how accurate the datasets are

Here are some other things to consider when selecting automation tools:

- ✓ **Scalability**: Whether the tools can scale to meet your needs
- ✓ **Flexibility**: How flexible the tools are
- ✓ **Integration capabilities**: How well the tools can integrate with other systems
- ✓ **Organizational maturity**: Whether you're starting with technologies that are easy to adapt

Points to Remember

- When working with PLC systems, various tools are required for installation, programming, testing, and maintenance. Some essential tools include:
  - **Programming Tools:** PLC Programming Software, PC or Laptop
  - **Wiring and Electrical Tools:** Screwdrivers, Wire Strippers, Multimeter, Crimping Tool
  - **Diagnostic Tools:** PLC Simulator or Tester, Logic Probes and Oscilloscopes
- Materials refer to consumables and components that are used to build and maintain a PLC-based automation system. These materials are critical for setting up the electrical connections and ensuring proper operation. Key materials include:
  - **Wiring and Cables:** Control Wiring, Power Cables, Network Cables (Ethernet, RS-485)
  - **Connectors and Terminals:** Terminal Blocks, Wire Connectors
  - **Electrical Components:** Relays and Contactors, Fuses and Circuit Breakers
- In addition to tools and materials, specialized equipment is required to ensure the successful operation and control of PLC-based automation systems. Important equipment includes:
  - **Programmable Logic Controller (PLC)**
  - **Input Devices:** Sensors, Push Buttons and Switches
  - **Output Devices:** Actuators, Motors, Valves
  - **Human-Machine Interface (HMI)**
  - **Power Supply Units (PSUs)**
  - **Communication Modules:** Network Routers, Hubs, and Switches
- Selecting the Right Tools for Automation, one of the most important decisions in prototype automation is choosing the right tools for the job. There are many factors to consider, such as the type of prototype, the complexity of the tasks, the budget, the skills of the team, and the desired outcomes.



Application of learning 1.3:

Imagine you are assigned to automate a water pump system in a small irrigation project. The goal is to automatically start and stop the pump based on soil moisture levels detected by a sensor. The system will also allow manual control through a push button and provide a status display for the operator. You need to select the appropriate tools, materials, and equipment to build and maintain this PLC-based system

Indicative content 1.4: Designing Automation System

Duration: 4 hrs

Theoretical Activity 1.4.1: Description of drawing environment

Tasks:

1: You are invited to answer the following questions related to the description of drawing environment:

i. What does drawing environment means?
ii. Identify standard symbols and notations for drawing system diagrams
iii. What do you understand by the term drawing templates?
iv. Explain how to apply colour scheme and styling

2: Present your findings to the whole class and Trainer

3: Ask your trainer for any clarification if any.

4: You are directed to read the key readings 1.4.1. for more clarifications

Key readings 1.4.1: Description of drawing environment

The drawing environment refers to setting up the necessary software and tools for creating system diagrams.

Before creating diagrams, it's important to set up a proper environment to work efficiently. This involves selecting the right tools and organizing the workspace.

- **Select a Suitable Tool**: Tools like **Microsoft Visio**, **Lucid chart**, **Draw.io**, **EPLAN**, or **AutoCAD** are great for creating automation-related diagrams. Each tool has its own set of features, so choose one that fits the type of diagram you need.
- **Define Standards**: Establish standards for how diagrams should be organized, including the scale, layout, and any specific notations or symbols that should be used.
- **Workspace Setup**: Ensure your workspace is properly configured for collaborative design if needed, and that your software is set up with necessary plugins or libraries.

Identification of Symbols and Notation

Using standard symbols and notations in diagrams ensures that they are universally understood by other engineers and technicians. Below are some key

symbols and notations used in different diagrams related to PLC system automation:

- **For Hardware Block Diagrams**:

✓ **PLC**: Represented by a rectangle with "PLC" labeled inside.

✓ **Input Devices (Sensors)**: Typically shown as small circles or boxes with labels like "Temperature Sensor."

✓ **Output Devices (Actuators, Motors)**: Represented as small boxes with an arrow indicating the function (e.g., motor, light, valve).

✓ **Power Supply**: Represented by a symbol of a battery or power symbol.

- **For Functional Flow Diagrams (Flowcharts)**:

✓ **Start/End**: Represented by ovals.

✓ **Process**: Represented by rectangles, indicating steps in the process.

✓ **Decision**: Represented by diamonds, indicating points where a choice is made (e.g., Yes/No).

- **For Architecture Diagrams**:

✓ **Controllers (PLCs)**: Usually represented by rectangles.

✓ **Networks**: Represented by lines or arrows showing communication between devices.

✓ **HMI (Human Machine Interface)**: Usually shown as a monitor or touchscreen icon.

- **For Data Flow and Communication (Sequence Diagrams)**:

✓ **Actors**: Represented by stick figures or rectangles (e.g., sensors, operators).

✓ **Messages**: Shown as arrows between components to represent data flow.

✓ **Lifelines**: Vertical dashed lines showing the timeline of communication.

Drawing Templates

Using templates helps standardize your diagrams and speeds up the design process. Most diagramming tools offer ready-to-use templates specific to system automation, PLC programming, and control systems.

- **Hardware Block Diagram Template**: Includes predefined blocks for PLCs, I/O modules, sensors, and actuators.
- **Functional Flow Diagram Template**: Predefined symbols for processes, decisions, and actions.
- **Architecture Diagram Template**: Pre-organized layout with areas for network devices, controllers, and communication pathways.
- **Sequence Diagram Template**: Contains lifelines, actors, and arrows for communication.

Benefits of Templates:

- Saves time by providing a basic structure.
- Ensures consistency across different diagrams.

- Enables easy modifications when system updates are required.

Applying Color Scheme and Styling

Adding colors and styles to your diagrams can improve clarity and readability. It also helps highlight different parts of the system, making it easier to understand complex processes.

- **Use Color Coding**:

✓ **Input/Output**: Use distinct colors for inputs (e.g., green for sensors) and outputs (e.g., blue for actuators).

✓ **Communication Lines**: Use different colors to represent various communication protocols (e.g., red for Ethernet, yellow for serial communication).

✓ **Power Lines**: Use solid bold colors to indicate power flow (e.g., black for ground, red for live wire).

- **Apply Styling**:

✓ **Line Types**: Use dashed lines for communication paths and solid lines for physical connections.

✓ **Text Formatting**: Use consistent font size and color for labels, making sure all labels are readable and clearly associated with their components.

- **Consistency in Colors**: Ensure that similar components or processes are consistently colored across all diagrams, so anyone looking at them will understand the relationships quickly. For example

✓ **Red:** Power lines or critical circuits.

✓ **Blue:** Communication cables.

✓ **Green:** Grounding or neutral lines.

✓ **Yellow:** Alarm or warning signals.

✓ Use **bold lines** for power circuits and dashed lines for communication paths

Practical Activity 1.4.2: Installing a drawing tool

Task:

1: Referring to the key reading 1.4.2 you are requested to perform the given task. The task should be done individually

As computer system architecture trainee, you are asked to install Draw.io

2: Present your work to the class or colleagues

3: Ask for any clarification if any.

Key readings 1.4.2: Installing a drawing tool

A drawing tool is a device or software used to create images, diagrams, or illustrations by applying marks on a surface, either digitally or on physical materials

Choosing and installing the right software for creating diagrams is crucial. Below are steps to guide you through this:

Choose the Tool: Depending on the type of diagram you want to create, select one of the following:

- **For Hardware Block Diagrams**: Use **AutoCAD**, **EPLAN**, or **Microsoft Visio**.
- **For Flow Diagrams (Flowcharts)**: Use **Lucid chart**, **Microsoft Visio**, or **Draw.io**.
- **For Architecture Diagrams**: Use **Visio**, **Lucid chart**, or **Enterprise Architect**.
- **For Data Flow and Communication (Sequence Diagrams)**: Use **Lucid chart**, **Draw.io**, or **Enterprise Architect**.

Draw.io is a popular diagramming and flowcharting software that is free to use and open source. In this tutorial, we will guide you through the steps to download and install Draw.io on Windows 11.

Step 1: Download the Draw.io Installer

The first step in installing Draw.io is to download the installer. To do this, follow these steps:

1. Open your web browser and navigate to the Draw.io website: draw.io">https://draw.io.
2. Click on the "Download" button located in the top menu of the website.
3. The download page will now appear. You can either choose to download the offline desktop version (as an EXE file) or the web-based version (as a ZIP file). We recommend downloading the offline desktop version for ease of use and convenience.

Step 2: Run the Draw.io Installer

After downloading the Draw.io installer, navigate to the download location and double-click on the EXE file to run it. Follow the on-screen prompts to complete the installation process.

Step 3: Launch Draw.io

Once the installation is complete, Draw.io should be available on your Windows 11 computer. To launch Draw.io, follow these steps:

1. Click on the "Start" menu located in the bottom-left corner of your

screen.

2. Type "Draw.io" in the search bar and click on the "Draw.io" app in the search results.

3. Draw.io should now launch and you can start working on your diagrams and flowcharts.

Practical Activity 1.4.3: Drawing system architecture

Task:

1: Referring to the key reading 1.4.3 you are requested to perform the given task. The task should be done individually

You are an automation technician tasked with enhancing the packaging line in a food processing plant. The goal is to streamline operations, reduce waste, and improve the accuracy of packaging.

Tasks:

i. Identification of System Components
ii. Create Hardware Block Diagram
iii. Create Functional Flow Diagram
iv. Create an Architecture Diagram
v. Dataflow and Communication (Sequence Diagram)

2: Present your work to the class or colleagues

3: Ask for any clarification if any.

Key readings 1.4.3: Drawing system architecture

1.Create Hardware Block Diagram

Purpose: Shows the physical setup of the system, displaying all hardware elements and their connections.

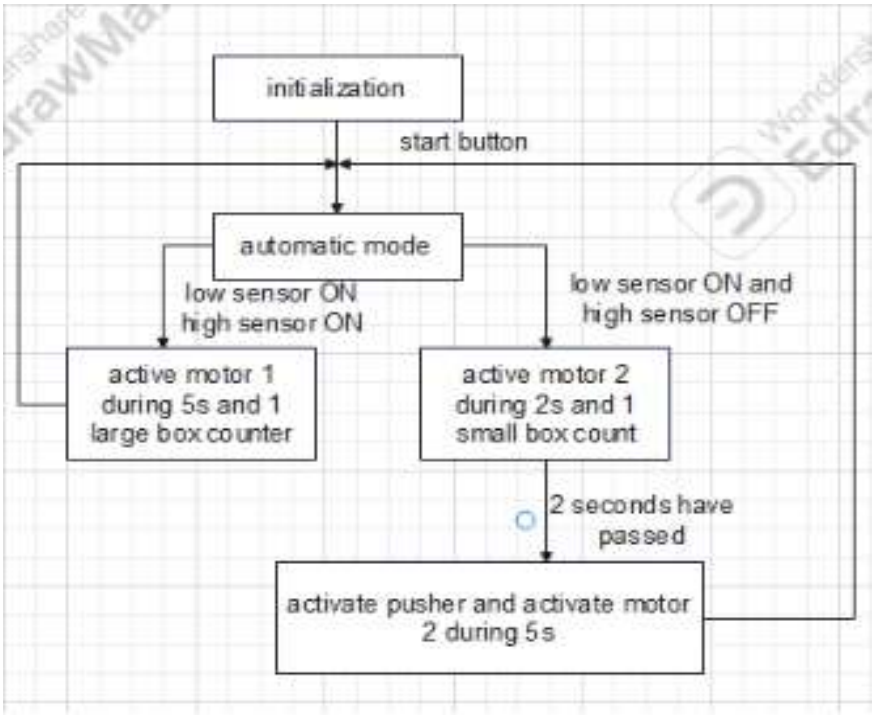
Components to Include:

Main processing unit (e.g., PLC, microcontroller).
Peripheral devices (sensors, actuators, interfaces).
Connections (e.g., communication buses, I/O lines).

Diagram Creation Steps:

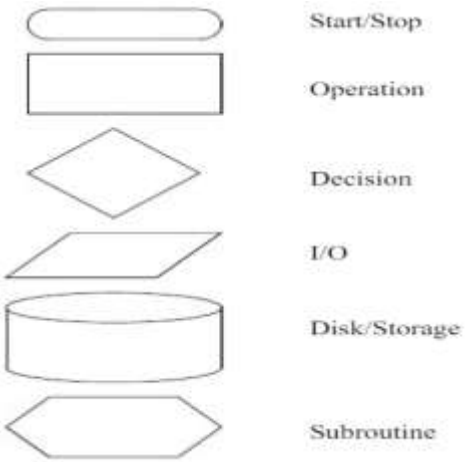
✓ Place each hardware component on the diagram.

- ✓ Draw connections (lines/arrows) showing how components are linked.
- ✓ Label each component and connection for clarity.



2.Create The Functional Flow Diagram (flowchart)

A flowchart is ideal for a process that has sequential process steps. The steps will be executed in a simple order that may change as the result of some simple decisions. The symbols used for flowcharts are shown below:



These blocks are connected using arrows to indicate the sequence of the steps. The different blocks imply different types of program actions. Programs always need a *start* block, but PLC programs rarely stop so the *stop* block is rarely used. Other important blocks include *operations* and *decisions*. The other functions may

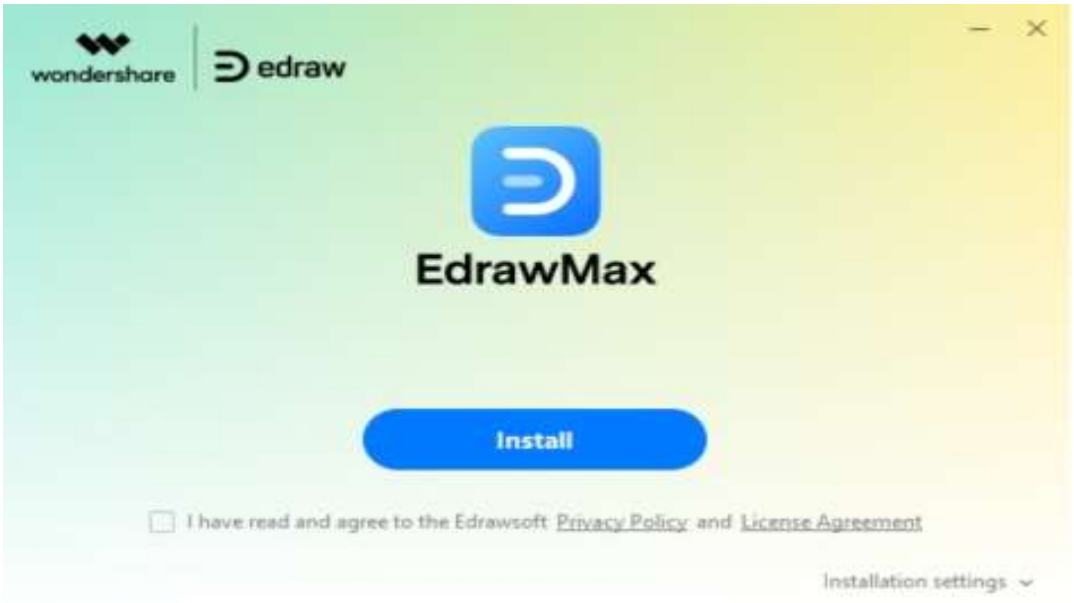
be used but are not necessary for most PLC applications. By developing logic flowchart, use **Edrow max** to design logic flowchart as follow in different steps:

Step 1. Download Edrow max setup

Type free download edrow max in your browser and select current version select download link 1

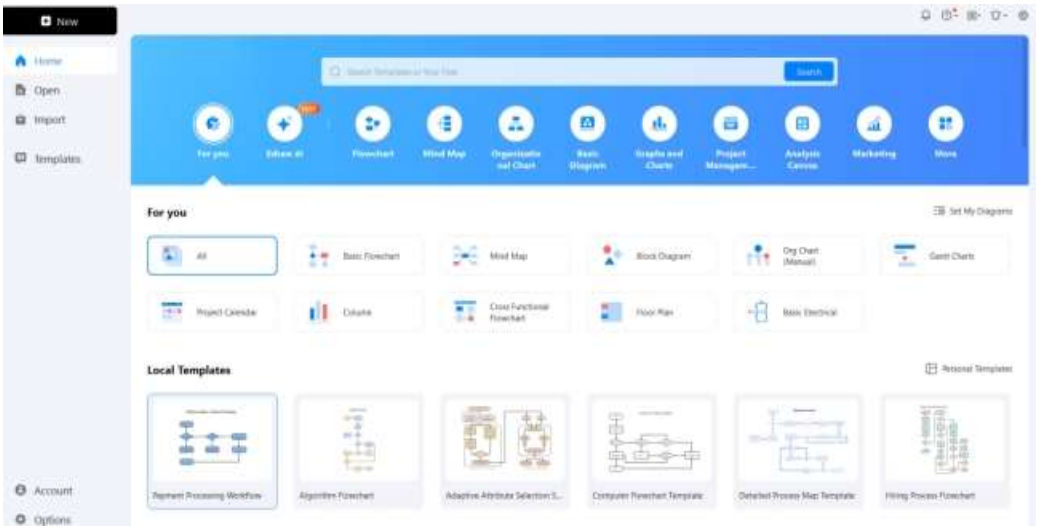
Step 2. Double click the setup file to install.

Before to install, select if you have read and agree to the edrow soft policy and license agreement

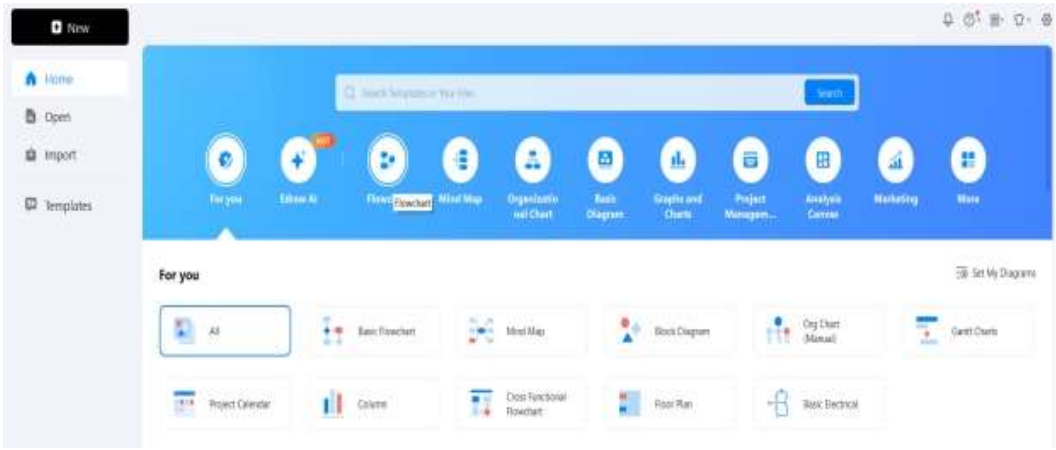


Step 3. Run the program.

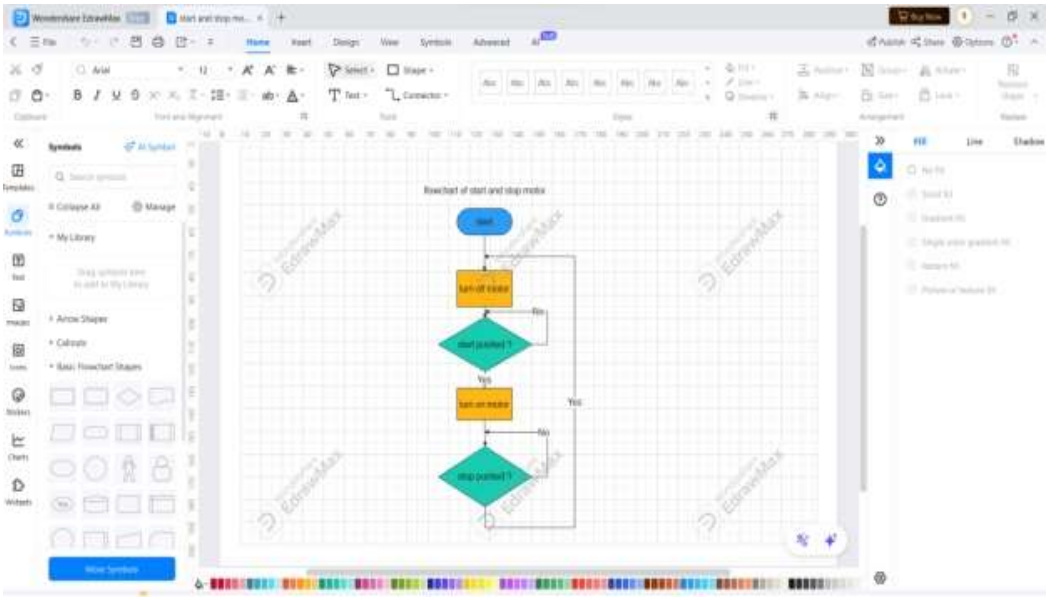
Now you are ready to develop flowchart of any related project in edrow max that select **''new''** in black colour



Step 4. Select the third tab of flowchart



Step 5. After selecting flowchart, you are allowed to start the design technics



3.Create an architecture diagram

An architecture diagram includes system components, their relationships and dependencies, the flow of data or information between components, the overall structure and organization of the system, and any external interfaces or interactions with other systems.

Step 1: Define System Layers

- **Typical Layers:**

✓ **User Interface (UI) Layer:** Front-end applications or user input devices.

✓ **Application Layer:** Core business logic or control algorithms.

✓ **Middleware Layer:** Communication protocols, data processing, and message brokers.

✓ **Database Layer:** Storage for system data.

✓ **Hardware Layer:** Physical components like sensors, actuators, and controllers.

Step 2: Identify Components within Each Layer

- Determine the components that make up each layer.
- List each component and its purpose briefly.

Step 3: Sketch the Basic Layout

Arrange each layer in the diagram to show the hierarchy from top (UI Layer) to bottom (Hardware Layer).

- **Steps:**

1. Use a rectangle for each layer, stacking them vertically or horizontally.
2. Place the UI Layer at the top and the Hardware Layer at the bottom.
3. Arrange all other layers in between based on the system flow (often top to bottom).

Step 4: Add Components to Each Layer

Place components within each layer to show what each part of the system contains.

1. Inside each layer, add boxes representing individual components (e.g., “PLC Controller” in Hardware Layer).
2. Space components out to avoid overlap for clarity.
3. Label each component with its name and primary function.

Step 5: Connect Components to Show Interactions

Illustrate how components in different layers interact and communicate.

1. Draw arrows between components to indicate the direction of data or command flow.
2. Use labelled arrows (e.g., “Sensor Data” or “Control Signal”) to make interactions clear.
3. If components communicate over a network or bus, use a network symbol or a bus line.

Step 6: Add Data Flow Labels

Describe the type of data or actions flowing between components.

1. For each arrow, add a label indicating the data or action it represents.
2. Ensure that labels are concise but descriptive.

Step 7: Enhance with Annotations or Notes (Optional)

Provide additional context for each layer or component.

1. Add short notes on the side of each layer describing its primary role.
2. If a component has a specific function, such as data encryption in Middleware, add a small note.

Step 8: Style the Diagram

Make the diagram visually clear and professional.

1. Use colors to distinguish layers or highlight important components.
2. Format arrows for clarity (e.g., dashed lines for indirect connections, solid for direct connections).
3. Ensure all text labels are legible and non-overlapping.

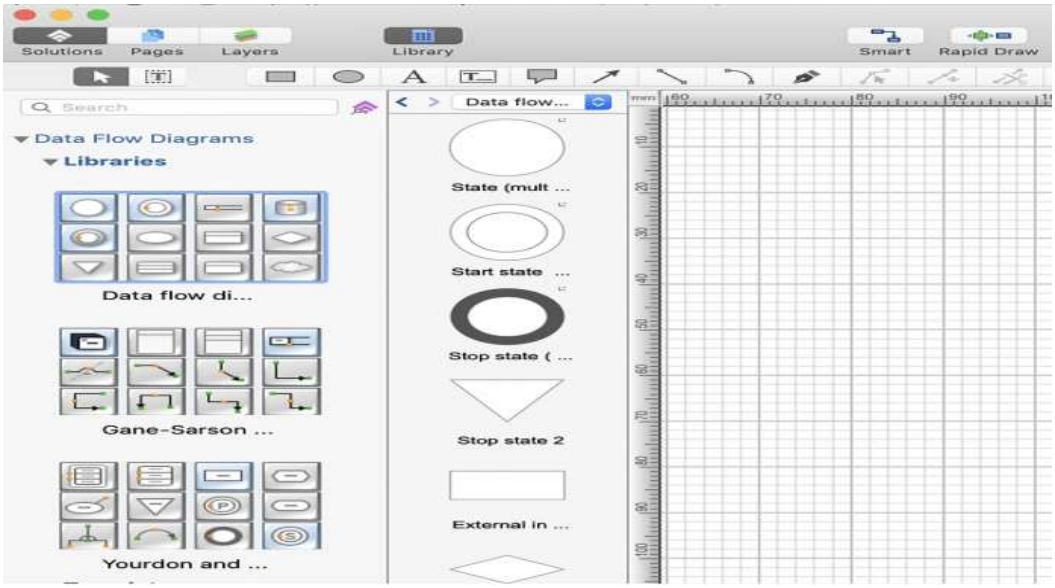
4. Dataflow and communication (sequence diagram)

Data Flow Diagrams (DFD) helps identifying business processes. At its simplest, a data flow diagram shows how data flows through a system. It displays where the data comes from and go to as well as where it will be stored.

Data Flow Diagram (DFD) can be used to visualize data flows in information systems. DFD consists of four major components: entities, processes, data stores, and data flows. The set of standard symbols is used to depict how these components interact in a system. Concept Draw DIAGRAM allows you to draw a simple and clear Data Flow Diagram using special libraries, provided by Data Flow Diagrams

The following steps are indicating how to create data **flow and communication (sequence diagram)**

Step 1. Open a Concept Draw DIAGRAM new document and select the Data Flow Diagram library.



Step 2. Drag and drop the following DFD elements from the library to your document:

Processes and functions

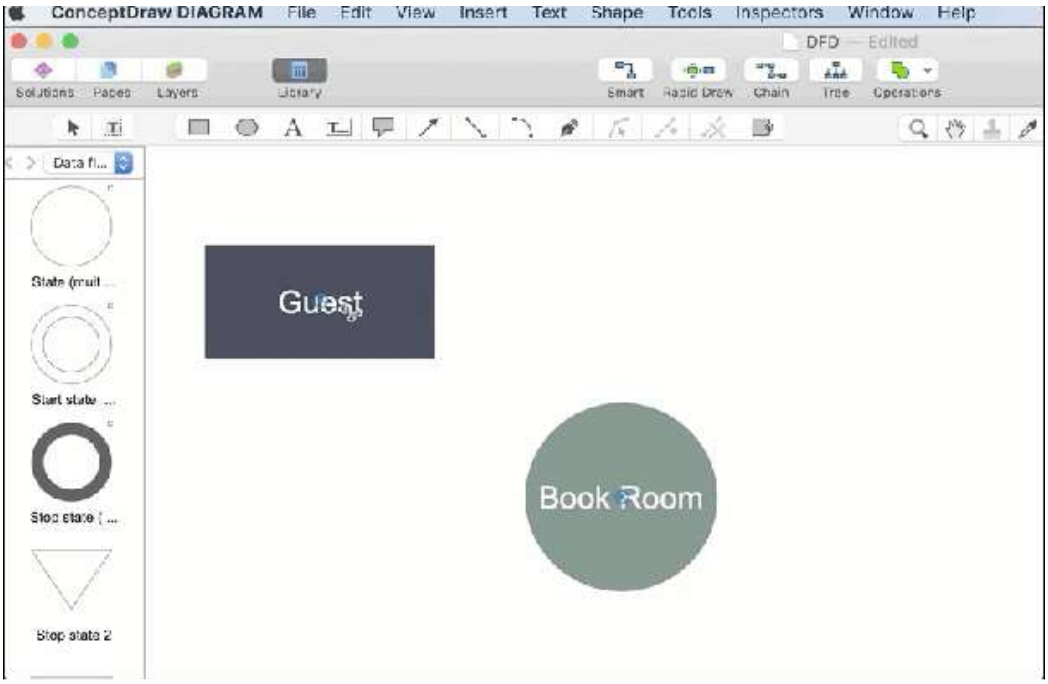
External entities

_____

_____

Data depositories

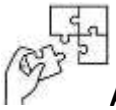
Step 3. Use connectors on Mac or PC to show the direction of data flow. In our case, we use the Arc connector. To add a text to connector select it and start typing.



Step 4. Use the Format panel to edit objects, text and even connectors.

Points to Remember

- Select a Suitable Tool: like Microsoft Visio, Lucid chart, Draw.io, EPLAN, or AutoCAD
- Using standard symbols and notations in diagrams ensures that they are universally understood by other engineers and technicians. So you have to know Hardware Block Diagrams, Flowcharts, Architecture Diagrams, Sequence Diagrams symbols
- Using templates helps standardize your diagrams and speeds up the design process. Most diagramming tools offer ready-to-use templates specific to system automation, PLC programming, and control systems.
- Adding colors and styles to your diagrams can improve clarity and readability. It also helps highlight different parts of the system, making it easier to understand complex processes.
- Steps to install Draw.io
    - Download the Draw.io Installer
    - Run the Draw.io Installer
    - Launch Draw.io



Application of learning 1.4.

As an automation technician, you are assigned to enhance the bottling line in a beverage production facility. Your primary objective is to streamline operations, minimize product spillage, and improve the accuracy of bottle filling and labelling

Indicative content 1.5: Installation of System Hardware

Duration: 4 hrs

Practical Activity 1.5.1: Installing of system hardware

Task:

1: You are requested to perform the given task. The task should be done individually.

As an automation technician, you have been assigned to install a robotic arm in an assembly line of a manufacturing facility to enhance production efficiency and accuracy. Your tasks involve preparing the installation site, connecting the necessary hardware components, and testing the hardware connections to ensure seamless integration with the existing system.

2: Present your work to the whole class and Trainer

3: Ask your trainer for any clarification if any.

4: You are directed to read the key readings 1.5.1. for more clarifications on installation of system hardware

Key readings 1.5.1: Installation of System Hardware

1. Prepare Installation Site

- **Site Assessment**:
  - Inspect the location for accessibility and suitability for hardware installation.
  - Ensure adequate space for equipment and proper ventilation.
- **Power Supply**:
  - Verify the availability and compatibility of power sources.
  - Check circuit capacity to support the hardware.
- **Safety Measures**:
  - Ensure all safety protocols are in place (e.g., grounding, ESD precautions).
  - Clear the area of obstacles and debris.

2. Connect Hardware Components

- **Gather Tools and Equipment**:

- Assemble necessary tools (screwdrivers, pliers, etc.) and hardware components (servers, cables, etc.).

- **Component Setup**:
  - Position hardware components (servers, switches, routers) according to the design layout.
- **Wiring Connections**:
  - Connect power cables to each hardware component.
  - Use appropriate cables (Ethernet, fiber optic, etc.) to connect network devices.
  - Ensure all connections are secure and labeled correctly.

3. Test Hardware Connection

- **Power On**:
  - Turn on each hardware component to check for power.
- **Initial Diagnostics**:
  - Observe indicator lights and display panels for error codes or warnings.
- **Connectivity Testing**:
  - Verify network connectivity by pinging devices and checking IP configurations.
  - Run diagnostics tools or software to ensure components communicate correctly.
- **Document Results**:
  - Record any issues encountered and the resolutions applied.

Final Steps

- **Cleanup**:
  - Remove any packaging materials and ensure the installation site is tidy.
- **User Training**:
  - Provide brief training for users on how to operate the new hardware if necessary.
- **Follow-up**:
  - Schedule a follow-up visit or check to ensure continued functionality and address any post-installation issues.

Points to Remember

- **Installing System** Hardware follow these steps:
  - ✓ Prepare Installation Site
  - ✓ Connect Hardware Components
  - ✓ Test Hardware Connection

Application of learning 1.5.

As an automation technician, you are responsible for installing a new automated sorting system in a warehouse aimed at improving inventory management and order fulfilment efficiency. Your tasks include preparing the installation site, connecting hardware components, and testing the system to ensure it operates correctly

Learning outcome 1 end assessment

Written assessment

A. Multiple-Choice Questions

1. Which of the following best defines automation?
    a) Increasing the need for human labor
    b) Implementing systems that operate automatically
    c) Decreasing productivity in manufacturing
    d) Focusing solely on aesthetic improvements
2. Which type of automation is ideal for high-volume, repetitive tasks?
    a) Flexible automation
    b) Programmable automation
    c) Fixed automation
    d) Batch automation
3. What is the primary purpose of a Programmable Logic Controller (PLC)?
    a) Controlling lighting in residential homes
    b) Managing office tasks
    c) Automating industrial machinery
    d) Enhancing manual control of systems
4. Which system requirement focuses on the system's ability to recover after a malfunction?
    a) Accessibility
    b) Usability
    c) Maintainability
    d) Recoverability
5. In system automation, the "usability requirement" primarily refers to:
    a) System affordability
    b) Ease of use for users
    c) Recovery options after failure
    d) System speed
6. When analyzing an existing system, which of the following is commonly identified?
    a) System marketing strategies
    b) User needs and system components
    c) Product pricing
    d) Employee training needs
7. Which type of system diagram shows the sequence of data and interactions between components?
    a) Functional flow diagram

b) Sequence diagram
c) Hardware block diagram
d) Architecture diagram

8. What is the first step when designing an automation system drawing environment?
    a) Choosing a color scheme
    b) Selecting tools and equipment
    c) Identifying symbols and notations
    d) Installing a drawing tool
9. In selecting tools, materials, and equipment for automation, which is considered a "tool"?
    a) PLC
    b) Flowchart software
    c) Circuit wiring
    d) Sensors
10. What is the main purpose of testing hardware connections during system installation?
    a) To ensure all components are properly connected and functioning
    b) To improve system appearance
    c) To finalize system documentation
    d) To increase system cost

B. Fill-in-the-gap questions

1. The main objective of identifying automation system requirements is to ensure that the system meets the user's _____ and performs effectively.
2. A Programmable Logic Controller (PLC) is commonly used in industrial applications to control _____ and _____ processes.
3. In system analysis, understanding the working principles helps determine how the automation system will operate and _____ in different scenarios.
4. System performance requirements are aimed at ensuring that the automation system can meet the necessary _____ standards.
5. The usability requirement of an automation system ensures that it is easy for users to _____ and operate.
6. When selecting tools, materials, and equipment for an automation project, _____ such as PLC programming software are necessary for system design and control.
7. In designing an automation system, installing a drawing tool allows the designer to create accurate _____ and _____ diagrams.
8. The hardware block diagram in an automation system is used to represent the system's _____ components and their connections.
9. System analysis techniques are used to evaluate the existing system's _____, limitations, and potential improvements.

10. Testing hardware connections during the installation phase ensures that all components are properly connected and _____ as expected.

C. OPEN QUESTIONS

1. What are the primary benefits of implementing automation in industrial processes, and how does it affect productivity and efficiency?
2. Explain the different types of automation systems and provide examples of where each type might be most effectively applied.
3. Describe the role of a Programmable Logic Controller (PLC) in automation. How does it differ from traditional control systems?
4. When analyzing an existing automation system, what system analysis techniques would you use to evaluate its effectiveness?
5. What are the key factors to consider when identifying the performance requirements of an automation system?
6. How do usability, recoverability, and maintainability requirements contribute to the overall success of an automation system?
7. What criteria would you use to select tools, materials, and equipment for designing and implementing an automation system?
8. Discuss the importance of creating an accurate hardware block diagram during the system design phase. What information should it convey?
9. How does the use of dataflow and communication diagrams (e.g., sequence diagrams) enhance the understanding of an automation system's functionality? Answer: Dataflow and sequence diagrams illustrate the flow of data and
10. Explain the process and importance of testing hardware connections during the installation of an automation system. What steps would you take to troubleshoot any issues?

Practical assessment

As an automation technician, you are tasked with Designing and install a PLC-based control system for a conveyor belt system in a manufacturing facility. The system must include the following functionalities

1. Control the conveyor belt motor based on inputs from proximity sensors.
2. Stop the conveyor when a part is detected at the end of the line.
3. Include emergency stop functionality.

END

Reference

Books

Alves, T. R., Buratto, M., De Souza, F. M., & Rodrigues, T. V. (2014, October). OpenPLC: An open source alternative to automation. In *IEEE Global Humanitarian Technology Conference (GHTC 2014)* (pp. 585-589). IEEE.

Francesco, Pasquale Chiacchio, and Diego Gerbasio. "On the implementation of industrial automation systems based on PLC." *IEEE Transactions on Automation Science and Engineering* 10, no. 4 (2012): 990-1003.

Ye, L. (2024). Design of PLC Electrical Control System. *Journal of Theory and Practice of Engineering Science*, *4*(03), 134-162.

Yuanyushkin AS, Lobanov DV, Rychkov DA. Automation tool preparation in the conditions of production. Applied mechanics and materials. 2015 Jul 20; 770:739-43.

Web links

Https://www.solisplc.com/tutorials.

https://instrumentationtools.com/plc-scan-time/#:~:text=Definition%20of%20PLC%20Scan%20Time,to%20a%20large%20till%201000ms.

Indicative contents
2.1 Installation of PLC simulation software **2.2 Selecting PLC programming languages** **2.3 Writing PLC program** **2.4 Converting PLC programming language**

Key Competencies for Learning Outcome 2 : Develop PLC program

Knowledge	Skills	Attitudes
• Description of PLC Architecture • Description of Programming Languages (Ladder Logic, FBD, ST) • Description of Control System Concepts	• Installing PLC software • Programming using Ladder Logic, FDB and ST • Using PLC Software tool • Wiring PLC program Blocks • Configuring PLC program Blocks • Testing PLC Program • Simulating PLC Program	• Being Attentive to Detail • Being a Problem-Solver • Being Adaptive • Being Safety-Oriented • Being Patience and Persistence

Duration: 40 hrs

Learning outcome 2 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Describe properly PLC architecture based on PLC type
2. Install correctly PLC simulation software based on the system to be automated
3. Select properly PLC programming languages based on the system to be automated
4. Write correctly PLC program based on automated system
5. Convert properly PLC programming language based on automated system PLC Program based on automated system required output

Resources

Equipment	Tools	Materials
• Computer • Input and output modules • Power supply components • PLC • Multimeter • Personal computer • PPE • Contactors • PLC modules (e.g., Siemens S7-1200)	• TIA Portal software • Screwdriver set • Wire strippers/cutters • Crimping pliers	• Electrical wires • Labels • Sensors • Actuators • Relays • Conduit • Zip ties • Internet • White board • Marker

Indicative content 2.1: Installation of PLC Simulation Software

Duration: 7hrs

Practical Activity 2.1.1: Installing PLC simulation software

Task:

1: You are requested to perform the given task. The task should be done individually.

You are asked to go in computer lab and perform the following tasks:

i. Prepare simulation software for installation (TIA Portal).
ii. Install simulation software (TIA Portal)
iii. Perform system compatibility check up

2: Present your work to the class or colleagues

3: Ask for any clarification if any.

4: Read the Key readings 2.1.1 in their manuals

Key readings 2.1.1.: Installing PLC simulation software

TIA Portal (Totally Integrated Automation Portal) is Siemens' all-in-one software suite designed for industrial automation, specifically for programming Siemens PLCs (Programmable Logic Controllers).

1. Steps of installing Simulation software (TIA Portal)

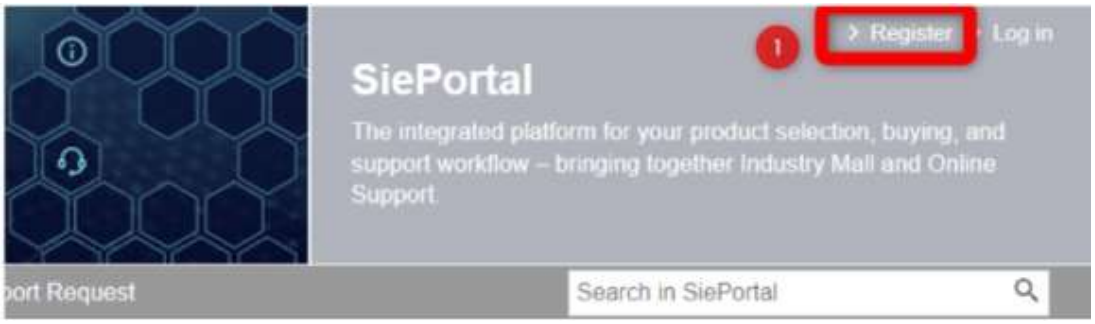
STEP 1. Create account on Siemens

Go to the link of Siemens Go to:

https://support.industry.siemens.com/cs/document/109772803/simatic-step-7-incl-safety-and-wincc-v16-trial-download?dti=0&lc=en-WW

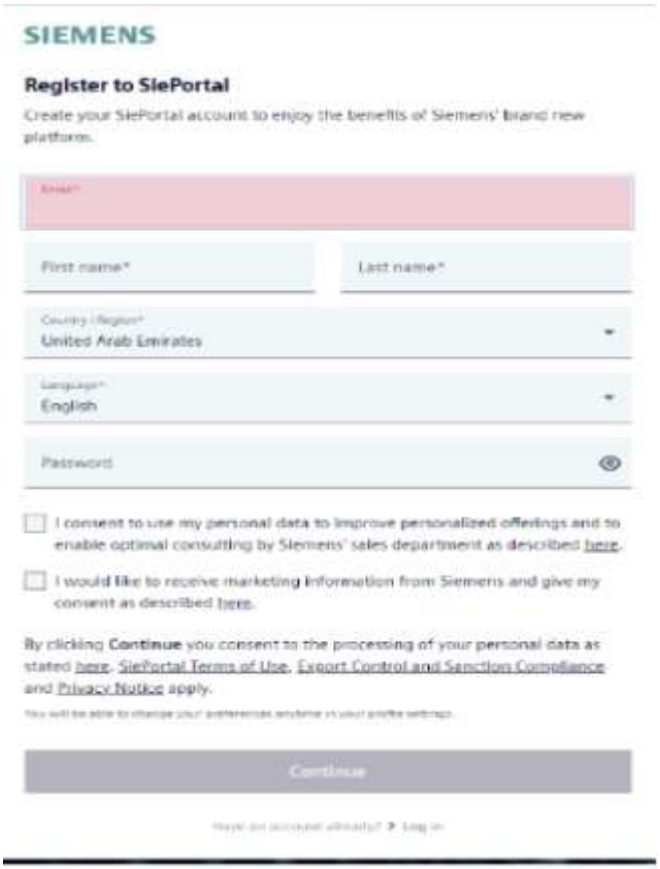
STEP 2

Click Register and create your account



STEP 3

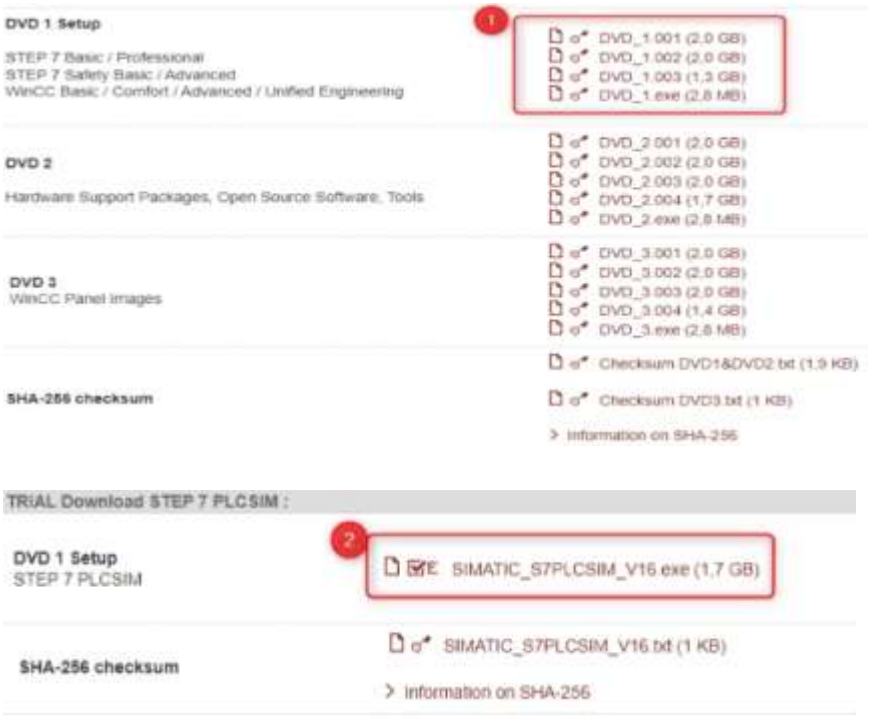
Fill the form and create your account.



STEP 4

Go back to: https://support.industry.siemens.com/cs/document/109772803/simatic-step-7-incl-safety-and-wincc-v16-trial-download?dti=0&lc=en-AE

And download zipped files marked in red (1) and (2).

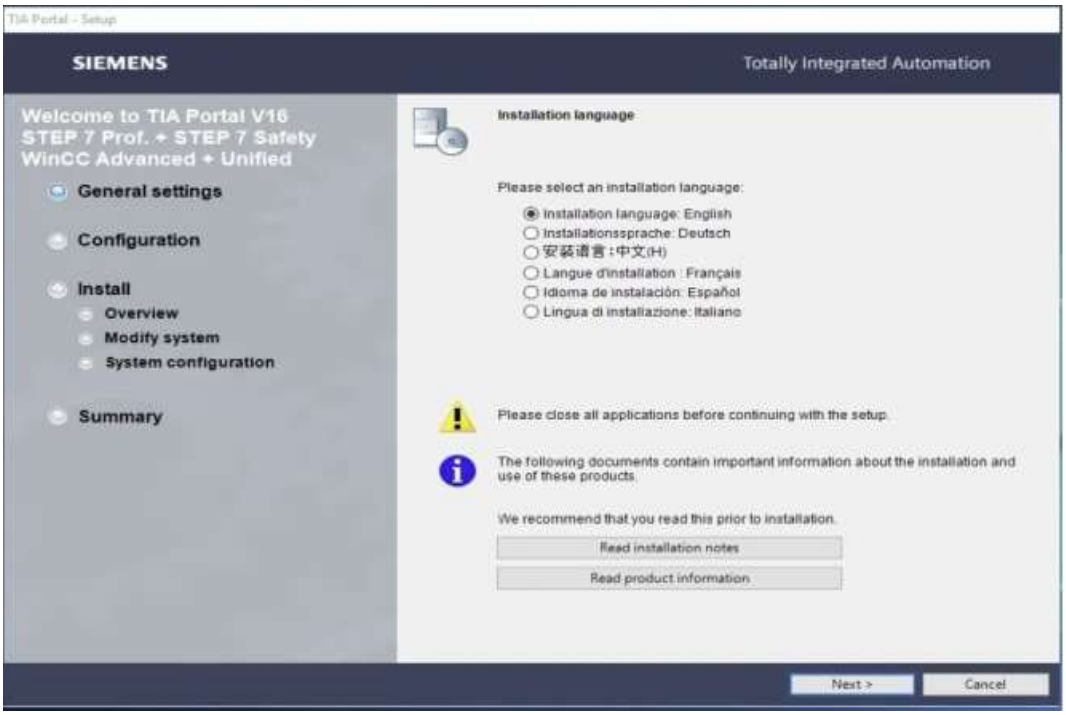


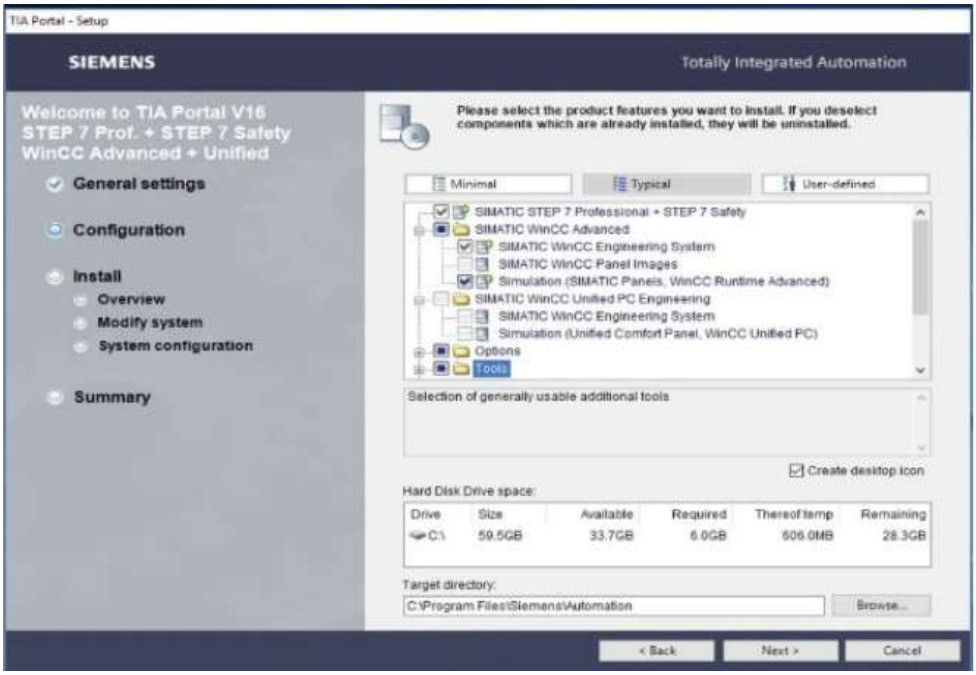
STEP 5

After downloading the files, move them all into a single directory and then double-click on the "TIA_Portal_STEP7_Prof_ Safety_WINCC_Adv_Unified_V17.exe" to start the installation

STEP 6

After selecting language, the installation will start automatically. Proceed with the installation as follows.


TIA Portal - Setup
SIEMENS
Totally Integrated Automation
Welcome to TIA Portal V16
STEP 7 Prof. + STEP 7 Safety
WinCC Advanced + Unified
General settings
Configuration
Install
Overview
Modify system
System configuration
Summary
Installation language
Please select an installation language:
Installation language: English
Installationssprache: Deutsch
安装语言：中文(H)
Langue d'installation : Français
Idioma de instalación: Español
Lingua di installazione: Italiano
Please close all applications before continuing with the setup.
The following documents contain important information about the installation and use of these products.
We recommend that you read this prior to installation.
Read installation notes
Read product information
Next >
Cancel


TIA Portal - Setup
SIEMENS
Totally Integrated Automation
Welcome to TIA Portal V16
STEP 7 Prof. + STEP 7 Safety
WinCC Advanced + Unified
General settings
Configuration
Install
Overview
Modify system
System configuration
Summary
Please select the product features you want to install. If you deselect components which are already installed, they will be uninstalled.
Minimal
Typical
User-defined
SIMATIC STEP 7 Professional + STEP 7 Safety
SIMATIC WinCC Advanced
SIMATIC WinCC Engineering System
SIMATIC WinCC Panel Images
Simulation (SIMATIC Panels, WinCC Runtime Advanced)
SIMATIC WinCC Unified PC Engineering
SIMATIC WinCC Engineering System
Simulation (Unified Comfort Panel, WinCC Unified PC)
Options
Tools
Selection of generally usable additional tools
Create desktop icon
Hard Disk Drive space:
Drive
Size
Available
Required
Thereof temp
Remaining
C:\
59.5GB
33.7GB
6.0GB
606.0MB
28.3GB
Target directory:
C:\Program Files\Siemens\Automation
Browse...
< Back
Next >
Cancel

STEP 7 – Install PLCSIM

After finishing the installation of STEP7, click on "SIMATIC_S7PLCSIM_V16.exe" and proceed with the installation of PLCSIM in the same way.

STEP 8 – Write your first PLC program

Great! You have TIA Portal installed and now you are ready to start coding!

2. Perform system compatibility check up

A **System Compatibility Checkup** in PLC (Programmable Logic Controller) simulation software is a systematic process used to verify that the software, PLC hardware, and any associated devices or systems are fully compatible and can interact without issues. This checkup is crucial to ensure that the simulation software can accurately emulate the PLC's operations and that it will function properly within the overall system environment.

Performing a system compatibility check for TIA Portal involves several steps to ensure that your hardware and software meet the necessary requirements. Here's a step-by-step guide:

Steps for System Compatibility Check:

✓ **Check System Requirements:**

Operating System: Ensure your computer's operating system is compatible with the version of TIA Portal you are installing.

§ Supported OS: Windows 10 (64-bit), Windows 7 (64-bit), or higher.

RAM: A minimum of 8 GB RAM is required, but 16 GB or more is recommended for smooth simulation and programming.

Processor: An Intel i5 or higher processor is ideal for handling the software's processing needs.

Storage: Ensure that there is enough free space (20 GB or more) on your hard drive to install TIA Portal and related files.

Graphics Card: TIA Portal requires a graphics card with a minimum resolution of 1920x1080 pixels.

✓ **Check Operating System Compatibility**

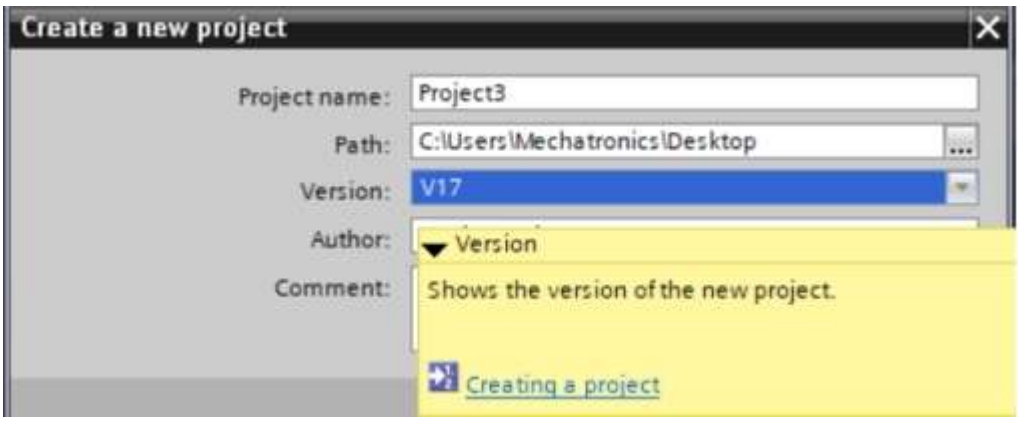
1. Operating System: Verify that your operating system (Windows version) is compatible with the TIA Portal version.

2. Service Packs/Updates: Ensure your OS has the latest service packs or updates installed.

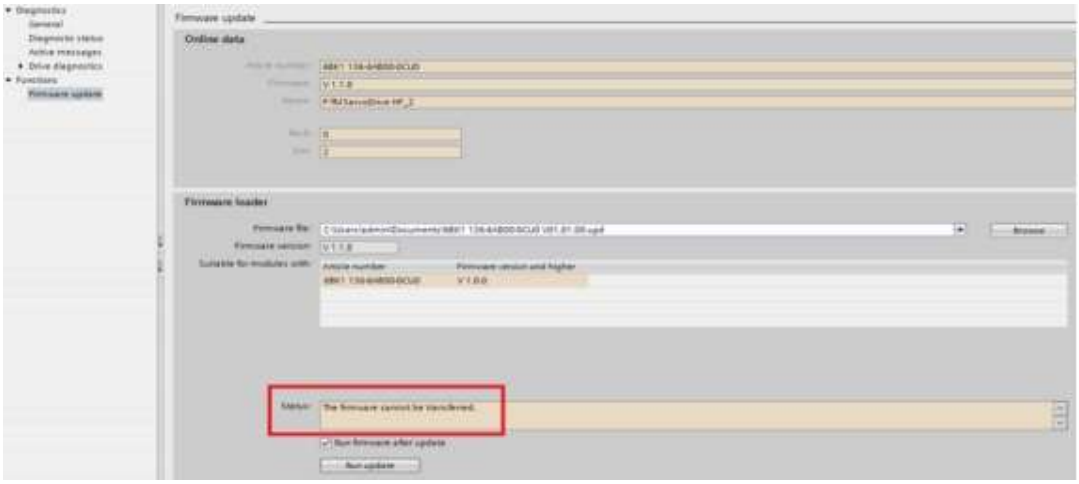
Install Required Software

Install .NET Framework: is a software development framework for building and running applications on windows.

Check if the required version of .NET Framework is installed.



Update Drivers: Ensure all hardware drivers (especially for graphics and USB devices) are up to date.



✓ **Perform Compatibility Check**

Run System Compatibility Check Tool:

- TIA Portal often includes a system compatibility check tool. Locate this tool within the installation files or the TIA Portal software suite.
- Follow the prompts to run the check. It will assess your system's hardware and software against the requirements.

Analyze Results

1. Review Compatibility Report: After the check is complete, review the report generated by the tool.
2. Address Any Issues: If the report indicates any incompatibilities, take necessary steps to address them:
   - Upgrade hardware or software.
   - Modify system settings as suggested.

Post-Installation Verification

1. Verify Installation: After installation, verify that TIA Portal runs smoothly and that all features are accessible.
2. Run Updates: Check for any available updates for TIA Portal post-installation.

Here are the key steps for performing a System Compatibility Checkup in **PLC simulation software**:

1. **Gather System Requirements**:
   - Collect all necessary specifications for both the existing hardware and software (e.g., PLC type, version of simulation software like TIA Portal, communication protocols, HMI software, etc.).
2. **Install and Update Simulation Software**:
   - Ensure the PLC simulation software (e.g., TIA Portal, STEP 7) is installed and up to date. Check if it supports the specific PLC models and components you're working with.
3. **Verify Software and PLC Compatibility**:
   - Check the PLC hardware version against the simulation software's compatibility list. Ensure the software can support the simulation for the PLC model you are using (e.g., Siemens S7-1200 or S7-1500).
   - Ensure that the software has the correct libraries and firmware versions installed to match the PLC hardware.
4. **Check Operating System Requirements**:
   - Ensure the simulation software is compatible with the operating system (e.g., Windows 10) of the computer it will be installed on. Check for any required patches or updates to ensure smooth operation.
5. **Check Communication Settings**:

- ❖ Confirm that the communication protocols used in the simulation software (e.g., PROFINET, MPI) are compatible with both the PLC hardware and any external devices (HMIs, sensors, actuators).
- ❖ Verify IP addressing, baud rates, and other communication settings in the software to ensure proper network connectivity in the simulation.

6. **Configure PLC and HMI in Simulation**:
   - ❖ Create or import the PLC program and configure the PLC model in the simulation software.
   - ❖ If using an HMI in the simulation, ensure that the software can support the HMI configuration, and check communication compatibility between the HMI and PLC.
7. **Run a Trial Simulation**:
   - ❖ Load the PLC program and perform a trial simulation to check if all configured devices (PLC, HMI, communication protocols) interact properly in a simulated environment.
   - ❖ Check for any errors or warnings during the simulation that may indicate compatibility issues.
8. **Test Hardware and Software Interactions**:
   - ❖ Test the interactions between PLC hardware (inputs/outputs) and the simulation software by simulating real-world conditions and monitoring the system's response.
   - ❖ Ensure the PLC responds as expected to control commands and feedback from the simulated environment.
9. **Document Results and Findings**:
   - ❖ Document any compatibility issues encountered during the simulation checkup, such as unsupported devices, software errors, or communication failures.
   - ❖ Recommend necessary updates, changes, or hardware adjustments to ensure full system compatibility.
10. **Optimize and Re-Test**:

- After resolving any compatibility issues, re-run the simulation to ensure the system operates correctly under various conditions.

These steps ensure that the PLC simulation software works effectively and is fully compatible with the PLC hardware and other system components.

3. Run and configure software packages

1. **Run the Installation:**

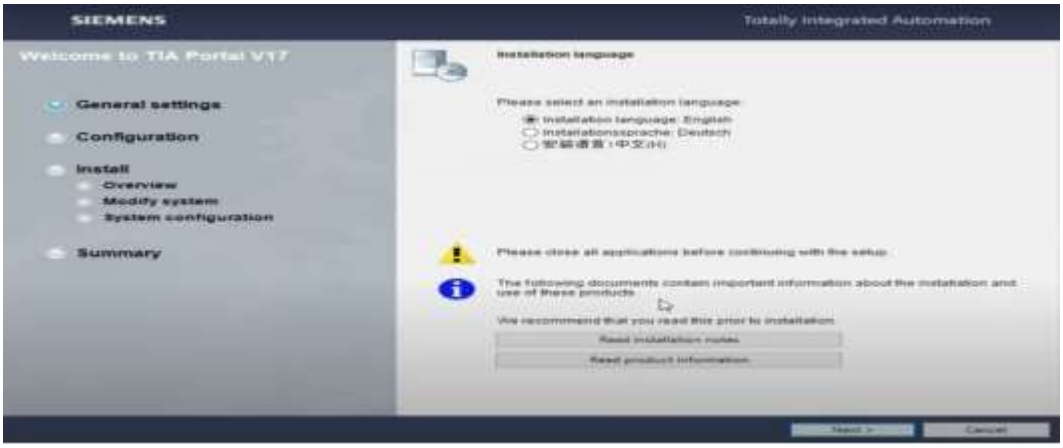
The following steps is explaining how to install TIA Portal setup in our computers as follow:

Step 1.

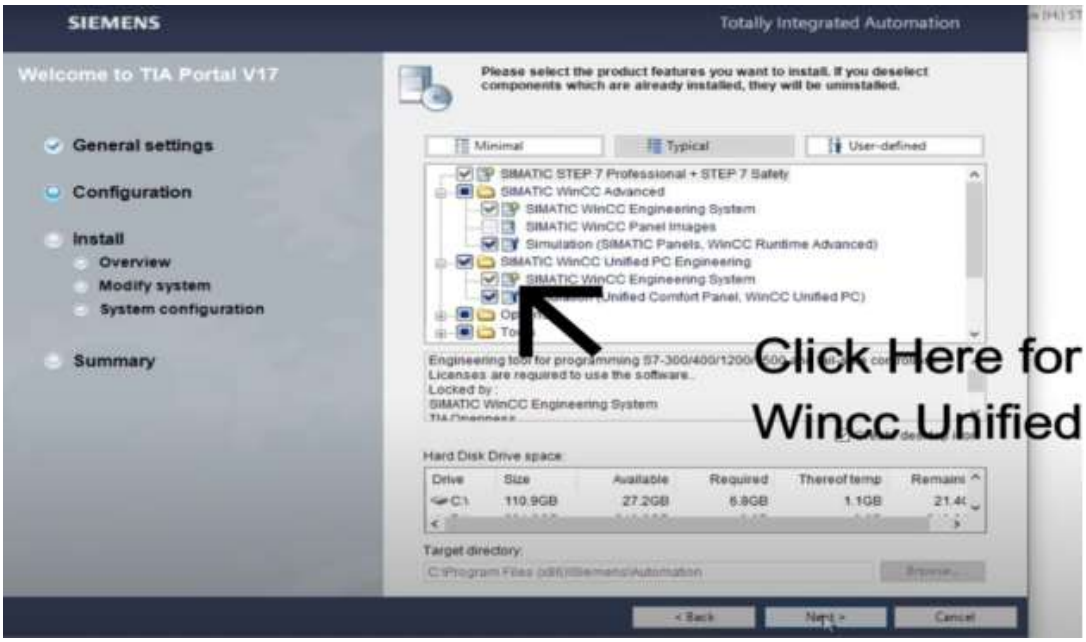
Double-click on the TIA Portal setup file to begin the installation process.

Follow the on-screen prompts to complete the installation.

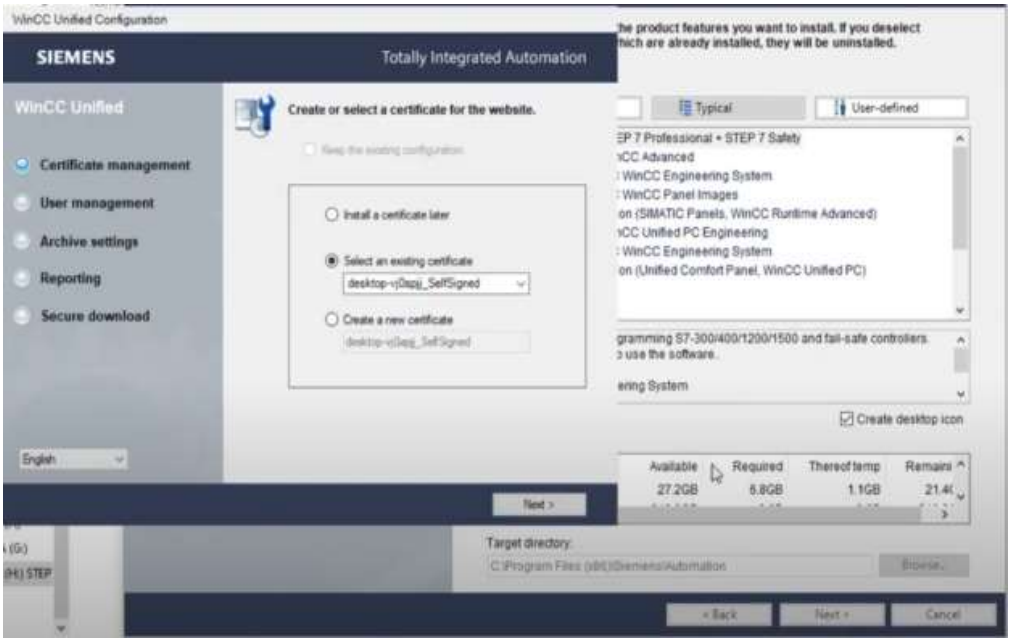
Select the components you want to install, such as **Step 7** (PLC programming), **WinCC** (HMI configuration), and the **PLC Simulator (PLCSIM)**.



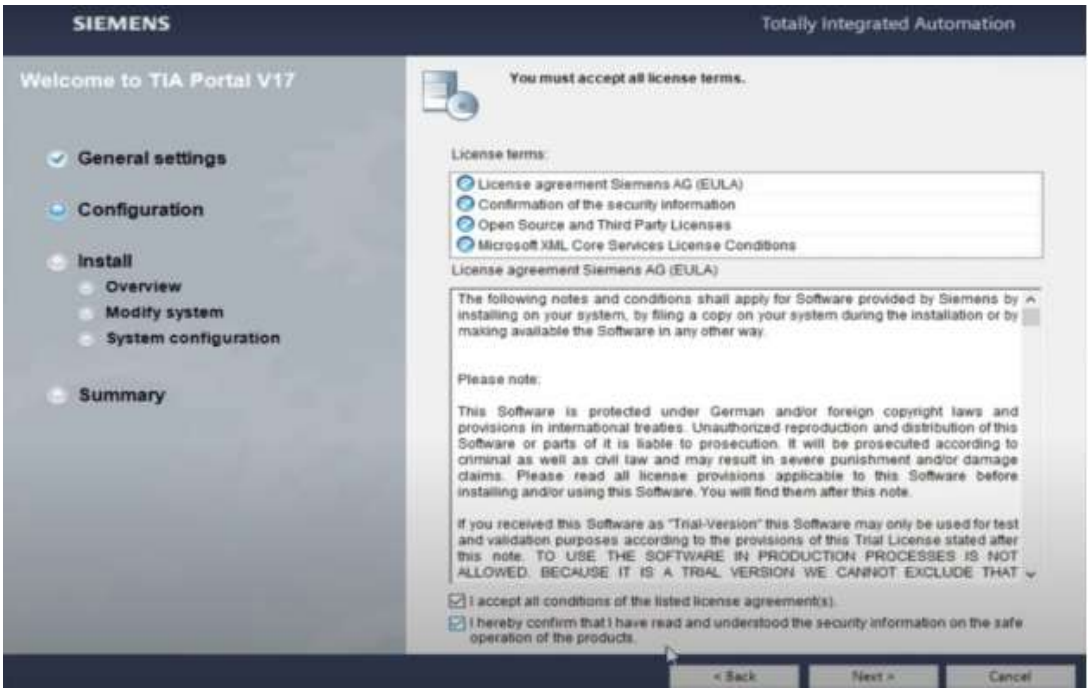
Step 2. Run the file: SIMATIC_S7PLCSIM_V17.exe and install TIA Portal PLCSIM



Step 3. Create or select a certificate for the website

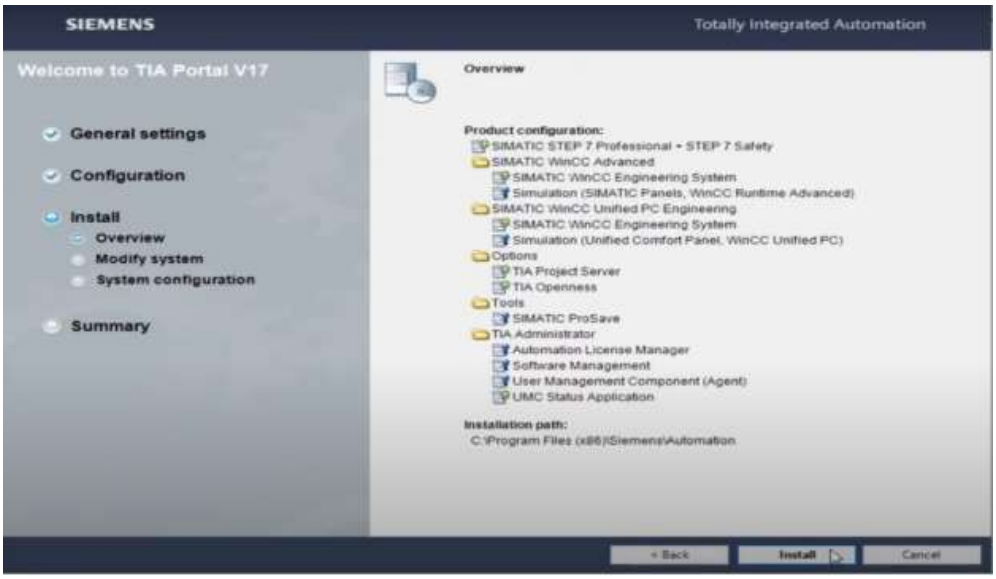


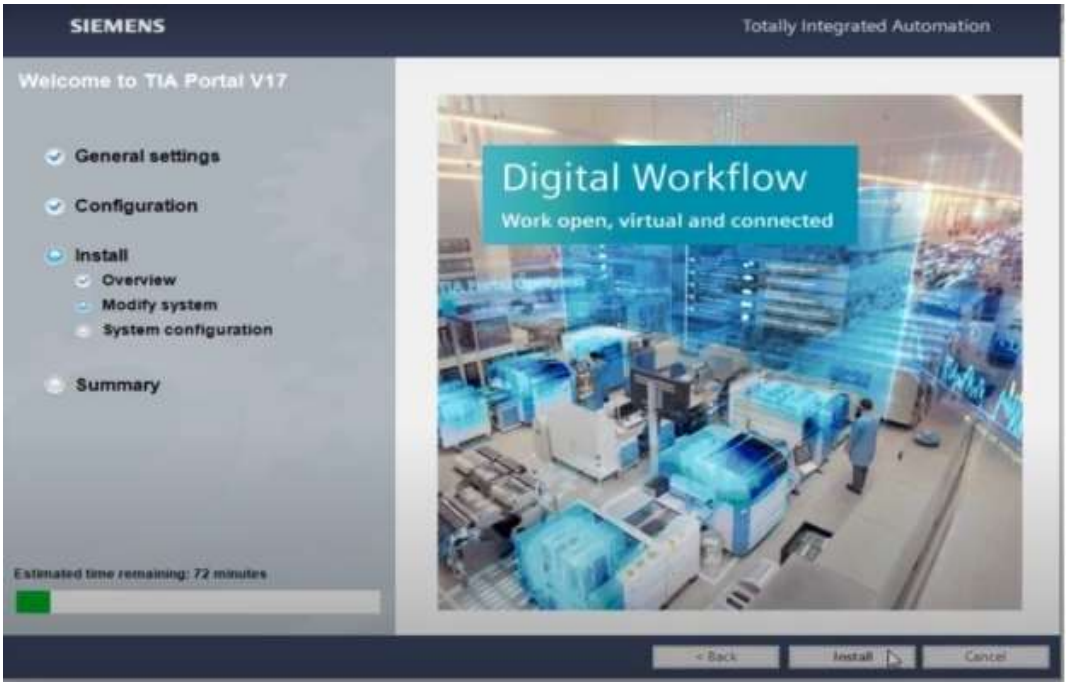
Step 4. To install TIA Portal, you must install license



Step 5. Correct functionality of TIA Portal V17 WinCC BCA Ed requires changes to some of the security and permissions settings on your system. you must accept these changes to continue the installation

Step 6. The TIA Portal setup, starting to install in a computer device that must be compatible with their operating system



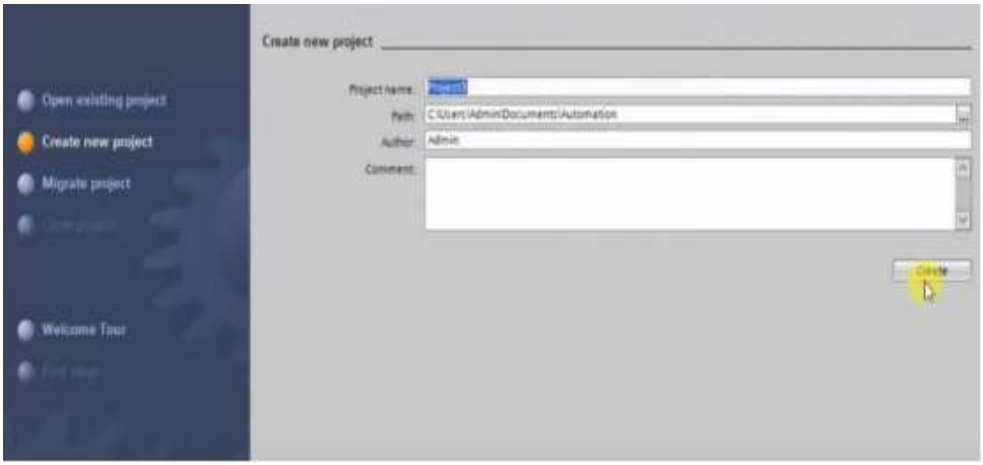


2. **Activate the License:**

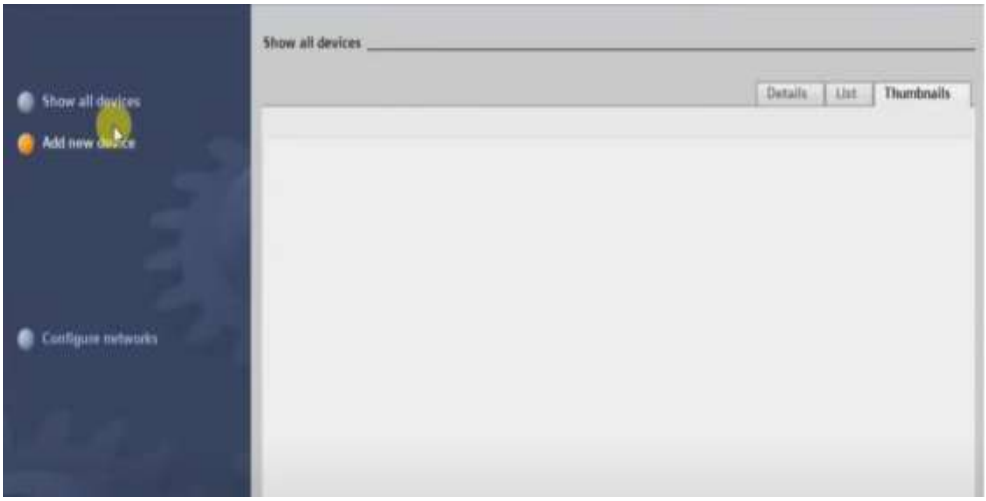
o If you have a licensed version of TIA Portal, activate the license using TIA Portal software properties.

The following steps indicating the procedures that helps to activate the license

Step 1. Open TIA Portal and create new project

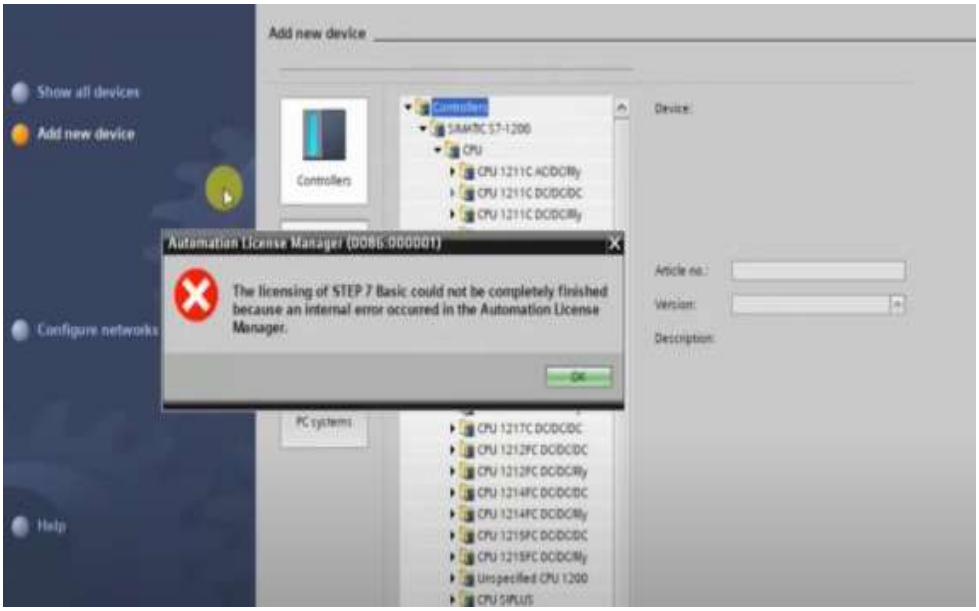


Step 2. Select configure a device and then add new device

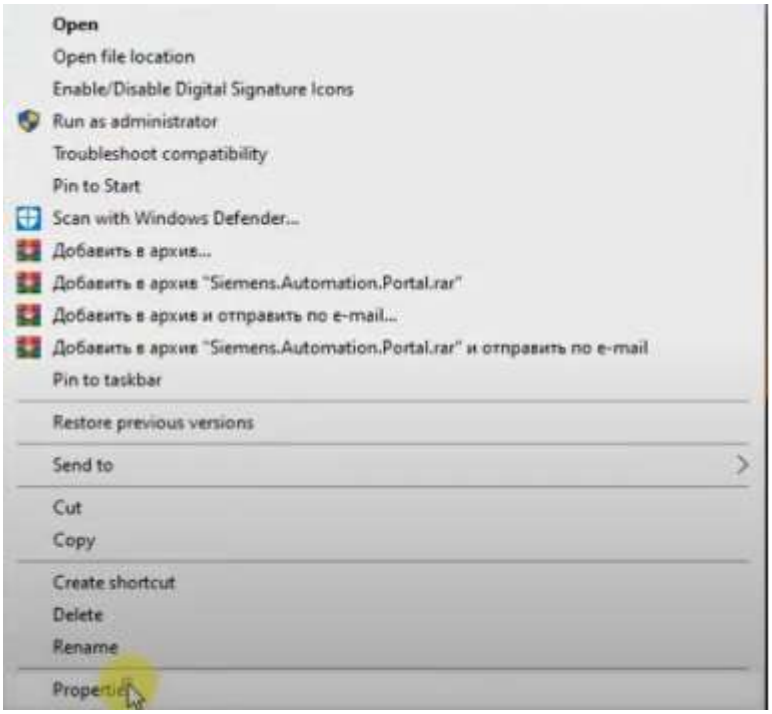


Step 3. Select controller type and its compatible of PLC device

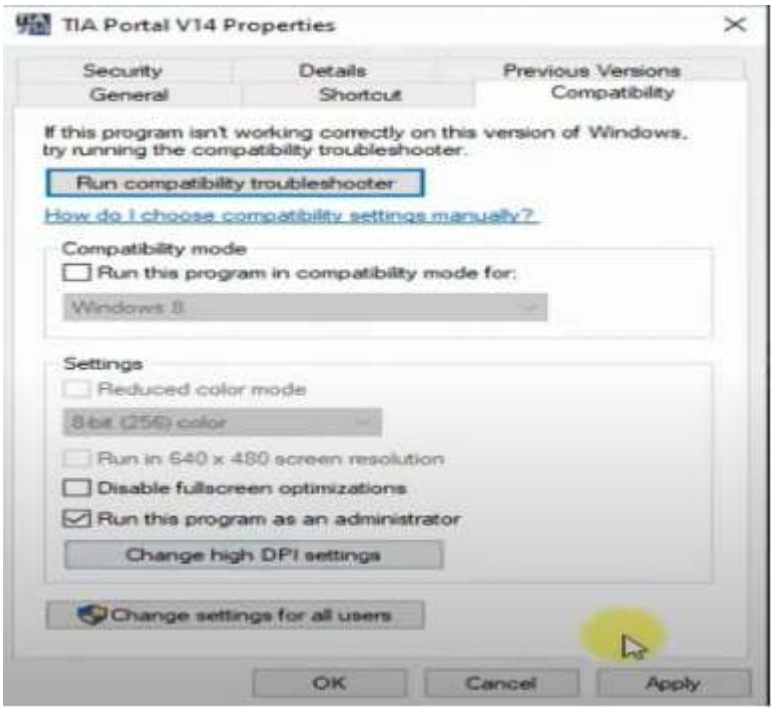
Step 4. Is not possible to select CPU because the licensing of STEP 7 Basic could not be completely finished



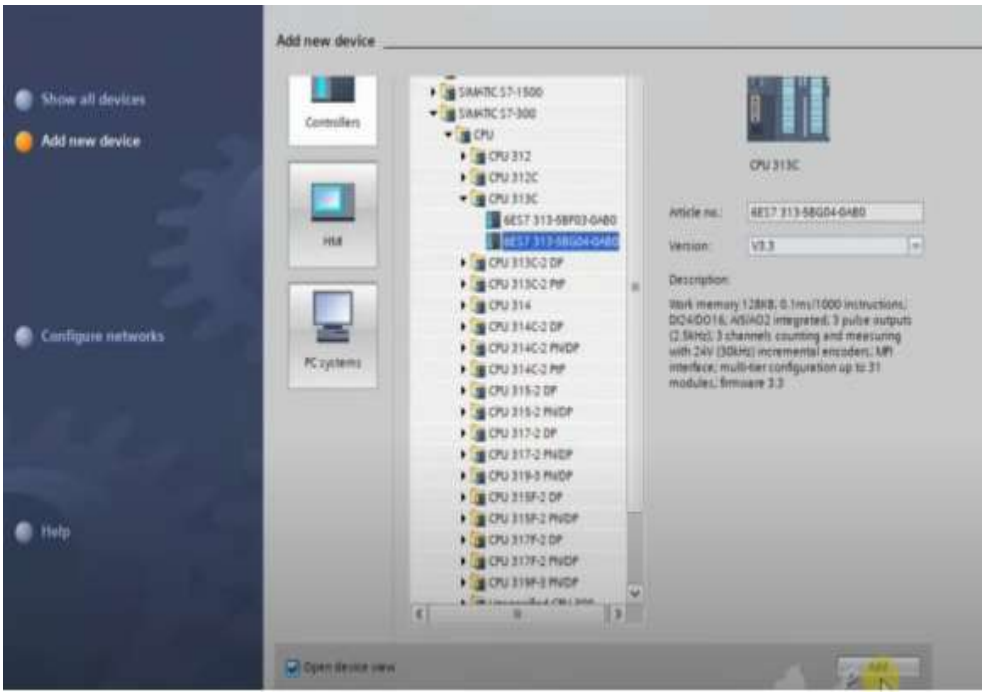
Step 5. This step indicates how to activate license Right click to the TIA Portal and select properties



Step 6. Select compatibility and Run this program as an administrator then apply



Step 7. Check if license was activated with Backing to TIA Portal and create new project, add new device, Select controller type, Select CPU type



3. **Configure Software Packages:**

After installation, open TIA Portal, and configure the environment for your simulation or project development.

Create a New Project:

- Set up a new project by selecting your PLC hardware (e.g., Siemens S71200) or select a generic PLC model for simulation purposes.
- Define the PLC configuration, including the number of inputs, outputs, and communication protocols.

Configure PLC Simulator (PLCSIM):

- Use Siemens **PLCSIM** to simulate the PLC program before uploading it to the actual hardware.
- Ensure that the simulated PLC behaves similarly to the physical PLC in terms of I/O, timers, counters, and communication.

Configure HMI (WinCC):

4. Install Updates and Add-ons:

- After the installation, check for any additional updates, patches, or add-ons available for the software to ensure the latest features and bug fixes are applied.
- Install any necessary libraries or function blocks required for your specific automation tasks.

5. **Run the Simulation:**

Once the PLC and HMI configurations are complete, run the PLC simulation using **PLCSIM**.

Test the logic, I/O, and communication with the simulated PLC to verify the correctness of your program before deployment.

6. **Test the Configuration:**
   - Perform test runs using the simulation environment to ensure that your program works as expected.
   - o Simulate various operating conditions, such as normal operation, sensor triggers, or fault conditions, to verify that the system responds appropriately.

7. **Save and Document Configuration:**
   - Save your project and maintain version control. This helps track changes and ensures you have backups of your project files.
   - Generate documentation from the TIA Portal software to provide a detailed overview of the PLC and HMI configuration.

Points to Remember

- **Steps of installing Simulation software (TIA Portal)**

Step 1. Create account on Siemens

Step 2. Click Register and create your account

Step 3: Fill the form and create your account.

Step 4: After downloading the files, move them all into a single directory and then double-click on the "TIA_Portal_STEP7_Prof_ Safety_WINCC_Adv_Unified_V17.exe" to start the installation

Step 5 – Install PLCSIM

Step 6 – Write your first PLC program

i. Performing a system compatibility check for TIA Portal involves several steps to ensure that your hardware and software meet the necessary requirements. Here's a step-by-step guide:

- **Steps for System Compatibility Check:**

Step 1: Check System Requirements

Step 2: Check Operating System Compatibility

Step 3: Install Required Software

Step 4: Perform Compatibility Check

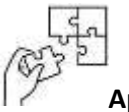
Step 5: Analyze Results

Step 6: Post-Installation Verification

- **Configure Software Packages:**

After installation, open TIA Portal, and configure the environment for your simulation or project development.

- ✓ Create a New Project
- ✓ Configure PLC Simulator (PLCSIM):
- ✓ Configure HMI (WinCC)
- ✓ Install Updates and Add-ons:
- ✓ Run the Simulation:
- ✓ Test the Configuration:
- ✓ Save and Document Configuration



Application of learning 2.1.

The XYZ Logistics Warehouse is upgrading its automated conveyor system to increase efficiency in packaging and sorting products. The existing system uses a Siemens S7-1200 PLC for control, and the new upgrade involves integrating an automated barcode scanner and a new conveyor belt drive**.** The barcode scanner comes with its own software, while the conveyor belt requires precise speed control through a Variable Frequency Drive (VFD), also connected to the PLC. Your task is to prepare simulation software installation (TIA Portal) and after perform system compatibility check up by analysing the current setup and ensuring the new barcode scanner software and conveyor VFD can communicate with the existing Siemens S7-1200 PLC

Indicative content 2.2: Selecting PLC Programming Languages

Duration: 10hrs

Theoretical Activity 2.2.1: Description of PLC programming languages

Tasks:

1: You are invited to answer the following questions related to Description of PLC programming languages:

What are the types PLC programming languages?

i. What are the application of PLC programming languages?
ii. What are the characteristics of PLC programming languages?
iii. What are the symbols and notations of PLC programming languages?

2: Present your findings to the whole class and Trainer

3: Ask your trainer for any clarification if any

4: You are directed to read the key readings 2.2.1 more details on PLC programming languages

Key readings 2.2.1.: Description of PLC programming languages

PLC programming languages are used to instruct PLCs (Programmable Logic Controllers) on how to control machinery, processes, and other automated systems. Selecting the appropriate language depends on the system's complexity, the industry standard, and the programmer's expertise.

Types PLC programming languages

- **Ladder Diagram (LD)**

Is a graphical programming language that you use to develop software for programmable logic controllers (PLCs).
It is one of the languages that the IEC 61131 standard specifies for use with PLCs.
A program in ladder diagram notation is a circuit diagram that emulates circuits of relay logic hardware.

Application:

Ladder diagrams are widely used in industrial control systems, especially in the field of automation and electrical engineering. Here are some key applications of ladder diagrams:

- ❖ Motor Control: Ladder diagrams can control the start/stop functions of motors, including sequence control for multiple motors.
- ❖ Conveyor Systems: Used to manage the operation of conveyor belts, ensuring proper start/stop sequences based on sensors.
- ❖ Temperature Control: Implementing temperature control loops for ovens, furnaces, or chillers.
- ❖ Pressure Control: Managing pressure levels in tanks and pipelines.
- ❖ Emergency Stop Circuits: Designing circuits that shut down machinery safely in case of an emergency.
- ❖ Sequential Control: Managing the steps in automated assembly processes, ensuring that operations occur in the correct order.
- ❖ Quality Control Checks: Integrating quality control measures at various stages of the assembly process.
- ❖ Lighting Control: Automating lighting systems in commercial buildings, including occupancy sensors.
- ❖ HVAC Systems: Managing heating, ventilation, and air conditioning systems based on temperature and humidity inputs.
- ❖ Pump Control: Managing pumps for water distribution and wastewater treatment systems.
- ❖ Chemical Dosing: Controlling the dosing of chemicals in water treatment processes.
- ❖ Robot Control: Implementing control logic for industrial robots in manufacturing settings.
- ❖ Elevator Control: Designing control systems for elevators, including floor selection and door operation.
- ❖ Smart Home Systems: Implementing control logic for smart lighting, security systems, and HVAC control.

❖ Teaching Tool: Used in educational settings to teach students about control systems and PLC programming.

Characteristics:

are advanced schematics widely used to record logic structures for industrial controls. These are called ladder diagrams because they mimic a ladder, with two vertical rails (supply power) and as many ‘’rungs’’ horizontal lines as there are to present control circuits and Ease of Troubleshooting, Real-Time Operation, Compatibility

Symbols and Notations:

• **Contact (--| |--)**: Represents input conditions such as switches or sensors.
• **Coil (-- () --)**: Represents output devices like motors or actuators.
• **Timers (TON, TOF)**: For delay or time-based operations.
• **Counters (CTU, CTD)**: For counting up or down events.

➢ **Function Block Diagram (FBD)**

is a **graphical language** for **programmable logic controller** design, that can describe the function between input variables and output variables. A function is described as a set of elementary blocks. Input and output variables are connected to blocks by connection lines.

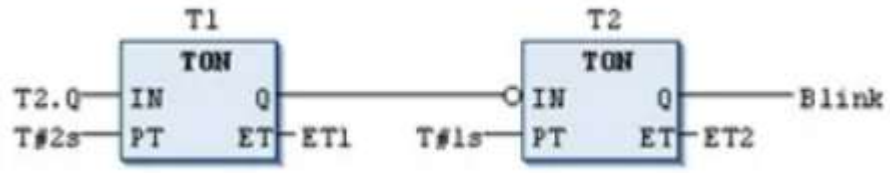
Inputs and outputs of the blocks are wired together with connection lines or links. Single lines may be used to connect two logical points of the diagram:

- An input variable and an input of a block
- An output of a block and an input of another block
- An output of a block and an output variable

The connection is oriented, meaning that the line carries associated data from the left end to the right end. The left and right ends of the connection line must be of the same type.

Multiple right connection, also called divergence, can be used to broadcast information from its left end to each of its right ends. All ends of the connection must be of the same type.

Simple function block diagram



Application:

• Used in process automation, continuous processes, and control systems involving analog signals and PID controllers.

Popular in industries like chemical, food processing, and water treatment, temperature regulation, Monitoring and Diagnostics.

Characteristics:

• Well-suited for complex systems with multiple functions and high reusability.
• Easier to implement control loops and analog processing.
• Can be difficult to troubleshoot in large systems due to the compact visual representation.

AND Gate:

- Symbol: A rectangle labelled **AND**.
- Function: The output is "true" only if all inputs are "true."

OR Gate:

- Symbol: A rectangle labelled **OR**.
- Function: The output is "true" if at least one input is "true."

❖ NOT Gate (Inverter):

- Symbol: A rectangle labelled **NOT** with a single input.
- Function: The output is the opposite of the input (if input is "true," the output is "false").

XOR Gate:

- Symbol: A rectangle labelled **XOR**.
- Function: The output is "true" if one input is "true," but not both.

❖ NAND/NOR Gates:

- Symbols: Rectangles labelled **NAND** or **NOR**.

• Function: The inverse of AND and OR operations, respectively.

➢ **Structured Text (ST)**

is one of the five languages supported by the IEC 61131-3 standard, designed for programmable logic controllers (PLCs). is a high level language, which represents a combination of three programming languages: Basic, Pascal and C. This language gives the possibility to operate with inputs and outputs, using different statements such as *for*, *while*, *if* and *case*.
The variables and function calls are defined by the common elements so different languages within the IEC 61131-3 standard can be used in the same program.

• A high-level textual programming language similar to Pascal or C, offering powerful control over the PLC.
• Allows for complex mathematical computations, data handling, and loop controls.

```
1  IF #start = 1 THEN
2      //comment
3      "Max_nr" := #Array[0];
4      FOR #i := 1 TO 10 DO
5          // Statement section FOR
6          IF #Array[#i] > "Max_nr" THEN
7              "Max_nr" := #Array[#i];
8          END_IF;
9      END_FOR;
10 END_IF;
11
```

Application:

• Ideal for complex automation systems, including large-scale industrial processes, advanced algorithms, or batch processing.
• Often used in applications that require significant data processing, such as robotics and advanced motion control systems.

Characteristics:

Allows for writing sophisticated and modular code with loops, conditions, and functions.

• Provides a lot of flexibility, making it powerful but harder to troubleshoot than graphical languages.

• Best suited for experienced programmers with knowledge of programming syntax.

Symbols and Notations:

IF, THEN, ELSE: Conditional statements.
FOR, WHILE: Loop control.

+ - * /: Arithmetic operators for mathematical operations.

: =: Assignment operator for variable storage.

➢ **Sequential Function Chart (SFC)**

Sequential Function Chart is a graphical programming language that is defined as *Preparation of function charts for control systems*. It is based on GRAFCET.
This language is used when programming a process that can be split into several steps.
A sequence is a step-by-step operation that should be used to perform a task. An example of a sequence can be a part painting process.

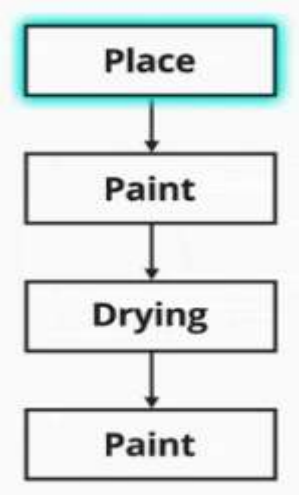
Here are the steps in a part painting process.

1) The first step is placing the part that needs to be painted in the machine.

2) And next step is the first coat of paint.

3) The next step is drying.

4) And the last step is the final coat of paint.



The actions and the conditions can be described in any PLC programming language. The SFC is basically a chart that represents an overview of the project, aimed to ease the analysis of the process.

Application:

• Typically used in batch processing, sequential operations (e.g., automated assembly lines), or any process that follows a defined sequence.
• Ideal for controlling processes that have distinct states or stages.

Characteristics:

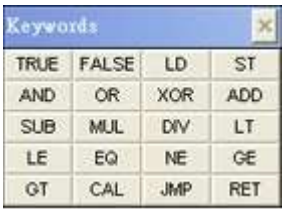
• Simplifies the representation of complex processes with many states.
• Easy to understand and troubleshoot, as the logic follows a clear sequence.
• Less suited for handling detailed mathematical operations or complex algorithms.

Symbols and Notations:

• **Step (□)**: Represents a discrete phase in the process.
• **Transition (→)**: Represents conditions that must be met to move to the next step.
• **Actions**: Associated with each step, defining what actions are to be taken.

➢ **Instruction List**

is a low level language that resembles the assembly language, a program consists of a series of instructions, listed as in an assembly program, there are some common operations, such as addition, division, multiplication and subtraction. There are also operations that make it possible to jump to some label, as well as to call a function.



Instruction List makes the program compact and offers a big processing speed. The downsides of this language are the structure and the syntax. It is difficult to debug a big list of instructions.

```
    ld      true
    st      blinker.run
    ld      t#1s
    st      blinker.cycle
    cal     blinker

    ld      blinker.q
    st      trigger.clk
    cal     trigger

    ld      trigger.q
    jmpnc   LBmodulo
    ld      counter
    add     1
    st      counter

LBmodulo:
    ld      counter
    lt      4
    jmpc    LBout
    ld      0
    st      counter

LBout:
    ld      counter
    eq      0
```

Application:

• Suitable for simple operations, particularly in legacy systems or when performance optimization is critical.
• Less commonly used in modern PLC systems but still found in certain applications requiring compact code.

Characteristics:

• Similar to assembly language, making it very efficient but harder to read and maintain.
• Useful for systems with memory or performance constraints.
• Offers precise control but is difficult for complex operations.

Symbols and Notations:

• **LD**: Load instruction.
• **ST**: Store instruction.
• **AND, OR, NOT**: Logical operations.
• **ADD, SUB, MUL**: Arithmetic instructions.

Theoretical Activity 2.2.2: Description of selecting criteria for PLC Programming language

Tasks:

1: You are invited to answer the following questions

Describe the following selecting criteria for PLC Programming language

i. Language features
ii. Compatibility and interoperability
iii. Scalability
iv. Learning curve
v. Security
vi. Advantages and disadvantages

2: Present your findings to the whole class and Trainer

3: Ask for any clarification if any

4: Read the key readings 2.2.2 more details on PLC programming languages

Key readings 2.2.1.: Description of selecting criteria for PLC Programming language

- **Key Considerations for Language Features:**

- Ease of Logic Representation
- Complexity Handling
- Libraries and Function Blocks
- Visual vs. Text-Based

- **Compatibility** refers to how well different programming languages, software, hardware, and PLC components can work together without causing conflicts or requiring significant modifications

Interoperability refers to the ability of systems, components, or devices to work together seamlessly, even if they are using different programming languages or technologies

- **Scalability** is essential when designing systems that may expand over time or require changes during operation.

The learning curve in PLC programming refers to the progression of skill and knowledge development as a person becomes proficient in writing, debugging, and understanding PLC programs

- **Key Considerations for Security in PLC programming language?**

• **Code Integrity** is the quality of software code that assures it is complete, correct, and consistent. The term is used to describe the confidence that the code will function as expected and that no unexpected problems will arise.

• **Access Control:** a fundamental component of data security that dictates who's allowed to access and use company information and resources. Through authentication and authorization, access control policies make sure users are who they say they are and that they have appropriate access to company dat

• **Validation and Verification**

Validation: Ensuring that the device meets the needs and requirements of its intended users and the intended use environment. Verification: Ensuring that the device meets its specified design requirements.

• **Error Handling and Robustness** works by identifying exceptions or issues that occur during the execution of a program or process. It employs different strategies such as exception handling, anomaly detection, and log inspection to ensure the smooth running of a system or application.

2. Advantages and Disadvantages PLC programming language:

Each programming language has its own strengths and weaknesses, making them more or less suitable depending on the application.

Ladder Diagram (LD)

• **Advantages:**

✓ Simple, easy to learn, and familiar to technicians with electrical experience.

✓ Visually intuitive, making it easy to debug and maintain.

✓ Ideal for simple discrete control applications.

• **Disadvantages:**

✓ Limited for complex algorithms, data handling, and large-scale systems.

✓ Not ideal for analog or continuous control.

Function Block Diagram (FBD)

Advantages:

✓ Reusable function blocks reduce development time.

✓ Good for analog control and systems with repeated functions.

✓ Visually clear, helping with diagnostics and troubleshooting.

Disadvantages:

✓ May become difficult to manage in large systems due to compact visual representation.

✓ Requires additional training for those unfamiliar with block diagrams.

Structured Text (ST)

• **Advantages:**

❖ Extremely powerful for complex data processing, math, and algorithmic control. o Modular and scalable, allowing for highly structured code.

❖ Similar to high-level programming languages, making it versatile.

• **Disadvantages:**

❖ Higher learning curve, requiring programming knowledge.

❖ Harder to debug and troubleshoot compared to graphical languages. o Less intuitive for those without programming experience.

Sequential Function Chart (SFC)

• **Advantages:** o Perfect for sequential and step-based processes. o Easy to visualize and understand process flow.

❖ Ideal for batch processing and systems with distinct states.

• **Disadvantages:**

❖ Not suitable for complex arithmetic or data handling. o Can become cumbersome in systems with too many transitions or conditions.

Instruction List (IL)

• **Advantages:**

✓ Compact and efficient, making it good for systems with limited memory or performance constraints.

✓ Simple to use for basic operations.

• **Disadvantages:**

✓ Hard to read, understand, and maintain.

✓ Not suitable for large-scale systems or systems requiring complex logic.

Points to Remember

PLC programming languages are used to instruct PLCs (Programmable Logic Controllers) on how to control machinery, processes, and other automated systems.

Types PLC programming languages

- **Ladder Diagram (LD**)

Application:

Ladder diagrams are widely used in industrial control systems, especially in the field of automation and electrical engineering. Here are some key applications of ladder diagrams:

- ✓ Motor Control
- ✓ Conveyor Systems Temperature Control
- ✓ Pressure Control Emergency Stop Circuits
- ✓ Sequential Control

Symbols and Notations:

- **Contact (--| |--)**: Represents input conditions such as switches or sensors.
- **Coil (-- () --)**: Represents output devices like motors or actuators.
- **Timers (TON, TOF)**: For delay or time-based operations.
- **Counters (CTU, CTD)**: For counting up or down events.

- **Function Block Diagram (FBD)**

AND **Gate**: Symbol: A rectangle labeled AND.
OR **Gate**: Symbol: A rectangle labeled OR.
NOT **Gate (Inverter)**: Symbol: A rectangle labeled NOT with a single input.
XOR **Gate**: Symbol: A rectangle labeled XOR.
NAND**/NOR Gates**: Symbols: Rectangles labeled NAND or NOR.

- **Structured Text (ST)**

Application:

- Used in process automation, continuous processes, and control systems involving analog signals and PID controllers.
- Popular in industries like chemical, food processing, and water treatment, temperature regulation, **Monitoring and Diagnostics**.

Symbols and Notations:

IF, THEN, ELSE: Conditional statements.
FOR, WHILE: Loop control.
+ - * /: Arithmetic operators for mathematical operations.

=: Assignment operator for variable storage.

- **Sequential Function Chart (SFC)**

Application:

- Typically used in batch processing, sequential operations (e.g., automated assembly lines), or any process that follows a defined sequence.
- Ideal for controlling processes that have distinct states or stages.

Symbols and Notations:

- **Step (□)**: Represents a discrete phase in the process.
- **Transition (→)**: Represents conditions that must be met to move to the next step.
- **Actions**: Associated with each step, defining what actions are to be taken.

- **Instruction List**

Application:

- Suitable for simple operations, particularly in legacy systems or when performance optimization is critical.
- Less commonly used in modern PLC systems but still found in certain applications requiring compact code.

Symbols and Notations:

- **LD**: Load instruction.
- **ST**: Store instruction.
- **AND, OR, NOT**: Logical operations.

ADD, SUB, MUL: Arithmetic instructions

- **Key Considerations for Language Features:**

  - ✓ Ease of Logic Representation:
  - ✓ Complexity Handling:
  - ✓ Libraries and Function Blocks
  - ✓ Visual vs. Text-Based

- **Compatibility** refers to how well different programming languages, software, hardware, and PLC components can work together without causing conflicts or requiring significant modifications

Interoperability refers to the ability of systems, components, or devices to work together seamlessly, even if they are using different programming languages or technologies

- **Scalability** is essential when designing systems that may expand over time or require changes during operation.

  The learning curve in PLC programming refers to the progression of skill and knowledge development as a person becomes proficient in writing, debugging, and understanding PLC programs
- **Key Considerations for Security in PLC programming language?**

- **Code Integrity:** is the quality of software code that assures it is complete, correct, and consistent. The term is used to describe the confidence that the code will function as expected and that no unexpected problems will arise.
- **Access Control:** a fundamental component of data security that dictates who's allowed to access and use company information and resources. Through authentication and authorization, access control policies make sure users are who they say they are and that they have appropriate access to company data.
- **Validation and Verification**

  Validation is ensuring that the device meets the needs and requirements of its intended users and the intended use environment. Verification: Ensuring that the device meets its specified design requirements
- **Error Handling and Robustness:** works by identifying exceptions or issues that occur during the execution of a program or process. It employs different strategies such as exception handling, anomaly detection, and log inspection to ensure the smooth running of a system or application.

Each programming language has its own strengths and weaknesses, making them more or less suitable depending on the application.

Application of learning 2.2.

As an automation technician**,** you are tasked with selecting appropriate PLC programming languages for XYZ Manufacturing's automated packaging line. The packaging system involves several operations; including barcode scanning, speed control using VFDs, and robotic arm coordination. You need to ensure that each function is handled with the most efficient programming approach for optimal system performance

Indicative content 2.3: Writing PLC Program

Duration: 15hrs

Practical Activity 2.3.1: Using Ladder Diagram (LAD) programming language

Task:

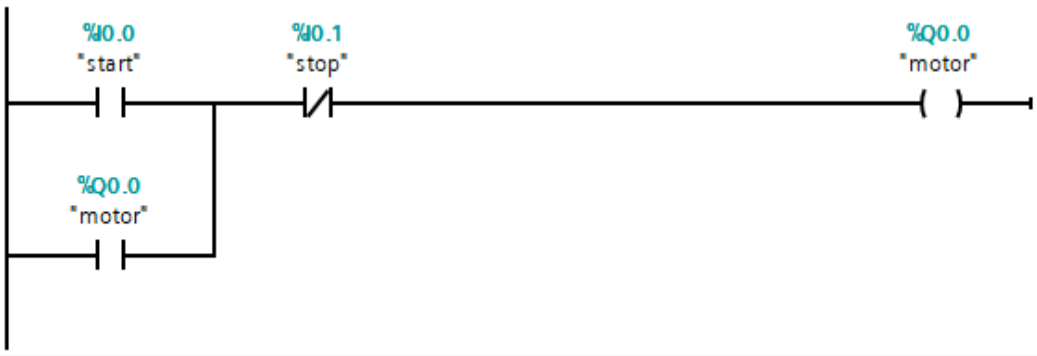
1: You are requested to perform the given task. The task should be done individually.
You are asked to go in computer lab and write PLC program by using Ladder Diagram (LAD) programming language
2: Present your work to the class or colleagues
3: Ask for any clarification if any.
4: Read the Key readings 2.3.1 in their manuals

Key readings 2.3.1.: Using Ladder Diagram (LAD) programming language

- **Defining system requirement**

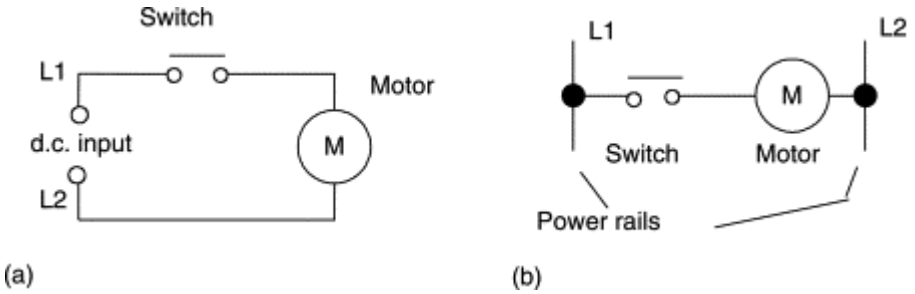
A ladder diagram is a method used in computer science to represent logical equations and simple actions. It uses contacts to represent input arguments and coils to represent output results

Ladder diagram (LD) is one of the most familiar methods of representing logical equations and simple actions. Contacts represent input arguments and coils represent output results.



As an introduction to ladder diagrams, consider the simple wiring diagram for an electrical circuit in (a). The diagram shows the circuit for switching on or off an

electric motor. We can redraw this diagram in a different way, using two vertical lines to represent the input power rails and stringing the rest of the circuit between them.
(b) shows the result. Both circuits have the switch in series with the motor and supplied with electrical power when the switch is closed. The circuit shown in (b) is termed a ladder diagram.



➢ **Developing logic ladder diagram**

Developing a logic ladder diagram involves a structured approach to visualize logical operations and control processes. Here is a step-by-step guide:

Step 1: Define the Purpose

❖ Identify the Process: Determine what process or system you are representing.
❖ List Inputs and Outputs: Define the inputs (conditions) and outputs (results) that will be included.

Step 2: Identify Logic Conditions

❖ Determine Conditions: Identify the logical conditions that affect the outputs. This could involve AND, OR, NOT, TIMER, etc.
❖ Prioritize Conditions: Organize the conditions based on their importance or sequence.

Step 3: Create the Ladder Structure

❖ Draw the Ladder: Create a vertical ladder-like structure. Each rung represents a condition or a series of conditions.
❖ Label Rungs: Label each rung with the relevant conditions.

Step 4: Define Logic Connections

❖ Use Logic Symbols: Incorporate symbols for AND (•), OR (+), and NOT (¬) operations.

❖ Connect Conditions: Draw lines to show how conditions connect to outputs based on the logical relationships.

Step 5: Test the Logic

❖ **Validate Scenarios:** Check various input combinations to ensure the logic correctly leads to expected outputs.
❖ **Adjust as Necessary**: Modify any rungs or connections based on your testing.

Step 6: Finalize the Diagram

Ensure that the diagram is easy to understand and accurately represents the logic. Include a key or legend if necessary to explain symbols used.

Step 7: Implementation

❖ **Use the Diagram**: Implement the logic ladder in the relevant system or process.

➢ **Translating logic into rungs**

Translating logic into rungs involves converting a control logic diagram (like a flowchart or a logical description) into a Ladder Diagram (LAD) format. Each rung of a ladder diagram represents a logical operation or sequence, typically corresponding to relay logic. Here are steps to be followed

1. **Understand the Logic**

Break down the process into conditions (inputs) and actions (outputs). Identify the sequence of operations. Each input could be a sensor, button, switch, or other field device, and each output could be a motor, light, valve, etc.

2. Identify Input/ Output Devices

❖ Inputs: Identify what signals you need to monitor. For example, start/stop buttons, limit switches, sensors.
❖ Outputs: Define what actions need to happen based on inputs. For example, turning on a motor, activating a solenoid, or illuminating a light.

3. **Decide on Logic Operations**

Determine the type of logic needed for each step: **AND, OR, NOT, Latch, Timer, Counter.** For example, if a motor should turn on when two switches are pressed, that's an AND logic operation.

4**. Convert to Ladder Rungs**

- ❖ Left Rail: Represents the positive power rail.
- ❖ Right Rail: Represents the ground or negative power rail.
- ❖ Rungs: Each rung contains inputs (conditions) and outputs (actions).

5. **Add Necessary Components**

Contacts: Represent your input conditions. These are normally open (NO) or normally closed (NC) switches.

Coils: Represent the output actions (turning on a motor, light, etc.).

Timers and Counters: For time-based or count-based control.

6. Finalize and Test

- Check your logic flow for any mistakes. Ensure that inputs are correctly represented as NO or NC.
- Simulate or test the program in a PLC software to see if it behaves as expected.

➢ **Program testing and debugging**

Testing is conducted to verify a software system's functionality, performance, and reliability to identify defects or errors. Here is how to test performance of ladder diagram if there is errors at the bottom of software.

✓ **Stages of Testing**

The stages of testing can vary depending on the software development lifecycle or methodology being followed. However, a general framework of testing stages typically includes the following:

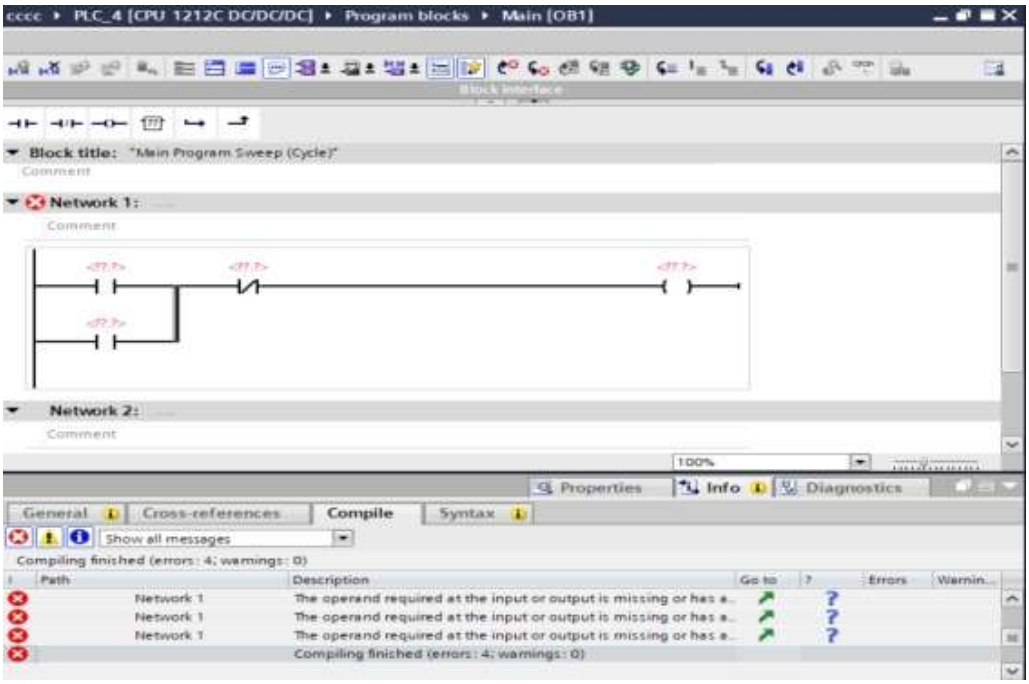
Test Planning: This stage involves creating a comprehensive test plan that outlines the testing approach, objectives, scope, resources, and timelines. It also includes identifying test requirements, test cases, and test data.

Test Design: Test cases are designed based on this stage's defined requirements and specifications. Test scenarios, scripts, and data are prepared to ensure adequate software functionality coverage.

Test Execution: Test execution involves running the designed test cases, executing test scripts, and observing the actual outcomes of the software. Testers compare the actual and expected results and report any deviations or defects.

Test Reporting: Test reporting involves documenting the test results, including any defects or issues discovered during testing. It includes detailed reports on test coverage, execution status, and defect metrics.

Test Closure: The test closure stage involves evaluating the overall testing process and determining whether the exit criteria have been met. It includes reviewing the test artifacts, identifying lessons learned, and preparing a final test report.



✓ **Debugging**

Debugging is investigating and resolving those defects, aiming to eliminate issues and ensure smooth operation. As you see at the bottom, all errors are corrected well

Stages of Debugging

The stages of debugging typically involve the following steps:

Observation: The initial debugging stage involves observing the software's behavior and symptoms to identify any issues or unexpected behavior. This may include error messages, crashes, incorrect outputs, or abnormal behavior.

Reproducing the Issue: The first step is consistently reproducing the problem. This may involve gathering information about the specific scenario, inputs, and conditions that trigger the issue.

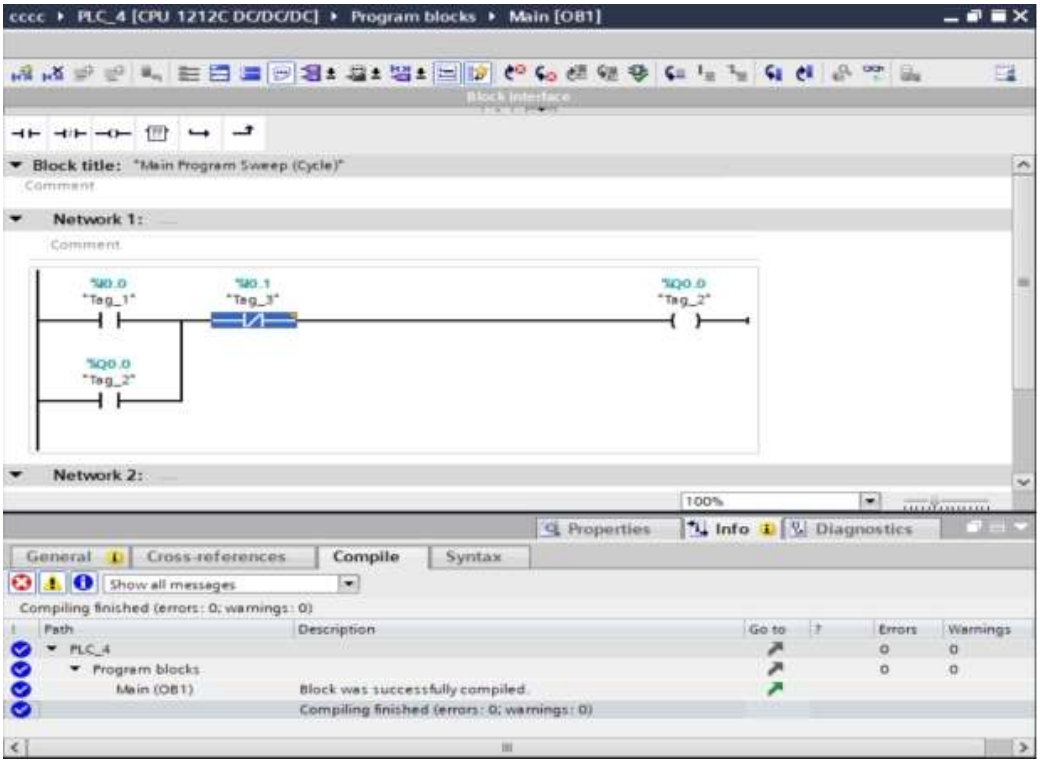
Identifying the Root Cause: Developers analyse the code, examine error messages, and debug the software to understand the underlying cause.

Isolating the Problem: This helps narrow the focus and avoid making unnecessary changes to unrelated parts of the software.

Fixing the Issue: This may involve modifying the code, adjusting configurations, or improving error handling to rectify the identified problem.

Testing and Validation: Once the fixes are implemented, thorough testing and validation are performed to ensure the issue has been resolved successfully. This includes retesting the problem scenario, running regression tests, and conducting additional tests to verify the effectiveness of the corrections.

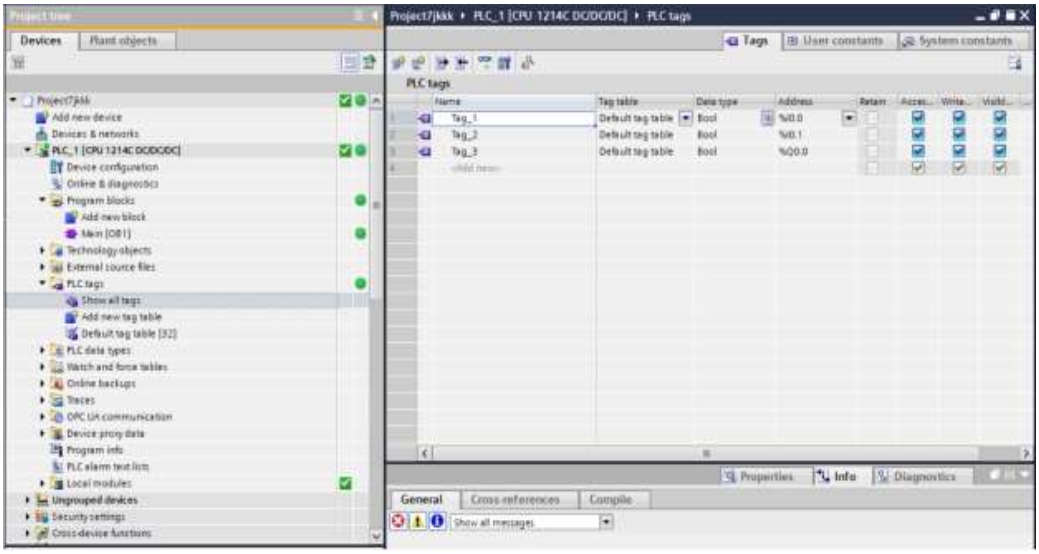
Documentation: It is crucial to document the debugging process, including the observed symptoms, root cause, applied fixes, and testing outcomes. Documentation is a reference for future debugging efforts, aids knowledge sharing, and helps maintain a record of resolved issues.



➢ **Documentation and maintenance**

PLC Program Documentation

❖ Tags and Symbols: Create a comprehensive list of variables, their addresses, data types, and descriptions.



❖ **Function Blocks (FB), Function Calls (FC), and Data Blocks (DB)**: Document the purpose, inputs/outputs, and relationships between function blocks, functions, and data blocks.

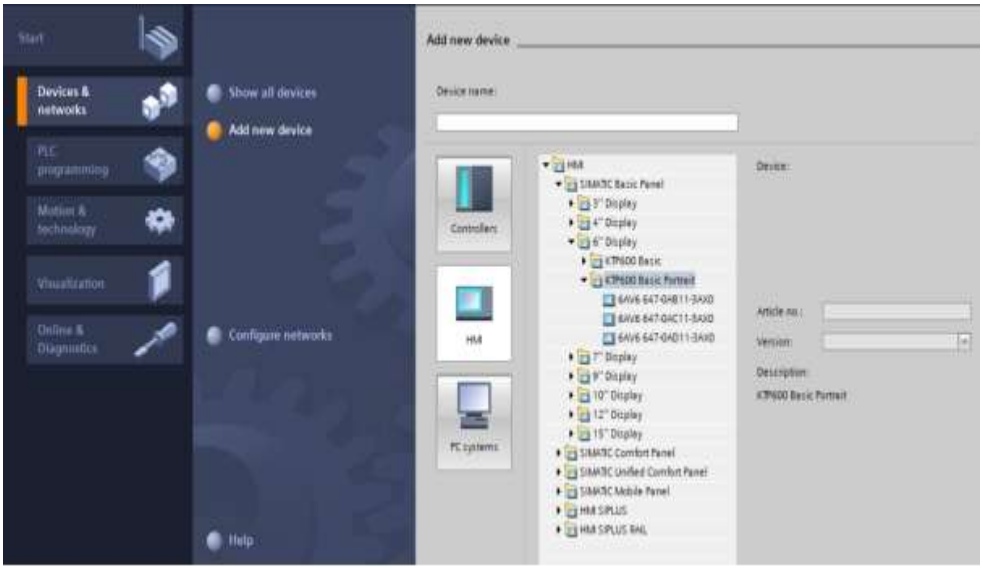
❖ **Program Structure:** Detail the organization of the program, including Main Programs (OB1, OB35, etc.), their triggers, and calling sequence.

❖ **Alarms and Diagnostics**: List all alarms, their triggers, and associated actions.

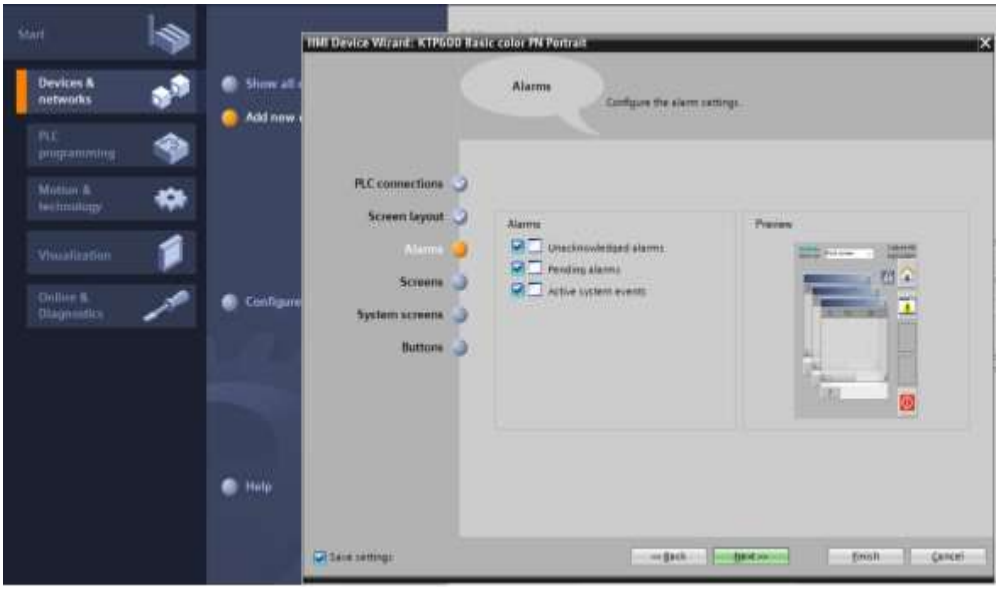
❖ **Interlocks and Safety Features**: Document any safety interlocks and fail-safes

HMI (Human-Machine Interface) Documentation

❖ **Screen Overview**: Provide a list of HMI screens with a brief description of their functions.



❖ **Tags and Alarms**: Document all the HMI tags and alarm messages that the operator can see.



User Access Levels: List the user levels and associated permissions.

I/O Configuration

❖ **Signal List:** Maintain a list of I/O signals, their types (analog/digital), and descriptions.

❖ **Wiring Diagrams**: Include or reference wiring diagrams to assist with hardware maintenance.

❖ **Input/output Addressing**: Include specific addresses for each signal.

Software Version Control

Use the built-in Version Control feature of TIA Portal to keep track of program changes.

Maintain a changelog that outlines changes, including who made them and why.

Project Maintenance

Maintenance refers to the process of ensuring that the TIA Portal project remains functional and up-to-date throughout its lifecycle. Key maintenance activities include:

a. **Regular Program Updates**

Firmware Updates: Regularly check Siemens for updates to PLC firmware and ensure compatibility with the TIA Portal project.

Program Enhancements: Update logic as per new system requirements or optimizations.

b**. Backup Strategy**

Automatic Backups: Set up regular backups of the TIA Portal project, especially after significant changes or updates.

Restoration Plan: Ensure the team knows how to restore from backups and has access to the necessary files.

c. **Diagnostic Tools and Logs**

Error Logs: Regularly review diagnostic logs from the PLC and TIA Portal to identify potential issues early.

Monitoring System Health: Use diagnostic tools in TIA Portal to continuously monitor the status of PLCs, I/O, and communication networks.

Fault Detection and Correction: Use the TIA Portal's diagnostic tools to track and resolve hardware or software issues as they arise.

d**. Network and Communication Monitoring**

Communication Checks: Periodically test network communication between PLCs, HMIs, and external systems to ensure they are working without delays or interruptions.

Security: Ensure that network security settings are maintaining and updated as per industry standards to prevent unauthorized access.

e. **Testing and Simulation**

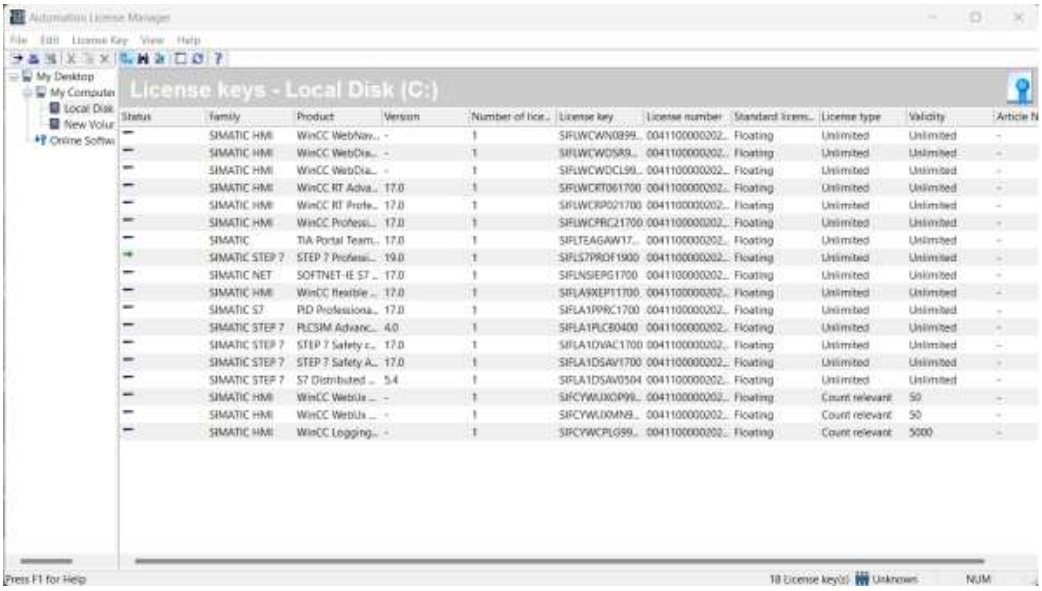
Software Simulation: Use the built-in simulation tools in TIA Portal to test changes before deploying them to the live system.

Hardware-in-the-Loop (HIL): For critical updates, test on a backup PLC or simulate with HIL to avoid disruptions to the production process.

f. **Licensing Management**

✓ Ensure that TIA Portal licenses are up to date, and keep track of the license keys.

✓ Maintain proper records of the licenses associated with the software and any hardware components (such as WinCC).



Practical Activity 2.3.2: Using Function Block Diagram (FBD) programming language

Task:

1: You are requested to perform the given task. The task should be done individually.

You are asked to go in computer lab and write PLC program by using Ladder Diagram (LAD) programming language

2: Present your work to the class or colleagues

3: Ask for any clarification if any.

4: Read the Key readings 2.3.2 in their manuals

Key readings 2.3.3: Using Function Block Diagram (FBD) programming

- **Defining system requirements**

1. **Understand the Process or System**:

Start by analysing what the system needs to accomplish.
Example: Control a motor, monitor sensor values, or activate an alarm based on certain conditions.

2. **Identify Inputs and Outputs**:

List all physical inputs (e.g. sensors, switches) and outputs (e.g. motors, alarms) required for the system.
Example: Inputs (Push button (Start), sensor (Temperature)
Outputs Motor (ON/OFF), Light (ON/OFF).

3. **Determine Control Logic**:

Break down the overall process into specific tasks or actions.
Example: Turn the motor ON when the start button is pressed and the temperature is within range.

4. **Set Timing and Safety Requirements**:

Identify any timing needs (e.g. delays or time-based operations) and safety conditions.
Example: Turn off the motor if the temperature exceeds a certain limit.

- **Selecting Function Blocks**

1. **Open Programming Environment**:

In **TIA Portal** or other PLC programming environments, open the FBD editor to begin adding function blocks.

2. **Choose Basic Function Blocks**:

Logical Blocks: For simple conditions and decision-making.
Example: Use **AND**, **OR**, and **NOT** blocks to handle control conditions (e.g. Start AND Temperature OK).

3. **Select Specialized Function Blocks**:

Timers: For time delays.
Example: Use an **On-Delay Timer** to delay motor start by 5 seconds.

Counters: For counting events.
Example: Use a **Counter** to track how many times a button is pressed.
Comparators: To compare values.
Example: Use a **Greater Than** block to turn off the motor if the temperature exceeds a certain value.

4. **Advanced Function Blocks** (if necessary):

PID Controllers: For regulating variables like temperature or pressure.
Mathematical Functions: For calculations, if your system needs to perform arithmetic operations.

- **Building the Program Logic**

1. **Create Logical Flow**:

Drag and drop the selected function blocks into the workspace and arrange them in a logical flow.

2. **Interconnect Function Blocks**:

Connect the blocks using lines to represent the flow of data.
Example: Connect the output of a **Temperature Sensor** block to a **Comparator** block (e.g. Temperature >= Limit).
Connect the result of a **Push Button** and **Temperature OK** (AND condition) to a **Motor Control** block.

3. **Ensure Proper Signal Flow**:

Inputs flow into blocks, while outputs come out of the blocks to control actuators, motors, or indicators.

4. **Add Labels and Descriptions**:

Label all inputs, outputs, and intermediate signals for clarity.
Example: Label the input as "Start Button" and output as "Motor Control" for easy understanding.

- **Configuring Block Parameters**

1. **Assign Inputs and Outputs**:

For each function block, configure the input and output assignments based on the system's I/O.
Example: Assign the **Push Button** input to %I0.0 and the **Motor** output to %Q0.0.

2. **Set Parameters**:

For specific function blocks, adjust their parameters to fit the application.

- **Timers**: Set the time delay (e.g., 5 seconds for an on-delay timer).
- **Counters**: Set the counting limit or reset condition (e.g., count to 10).
- **Comparators**: Set threshold values for comparisons (e.g., Temperature > 75°C).

3. **Initialize Variables**:

Ensure that any variables or data blocks are initialized properly.

Example: Set the starting value of a counter to zero or define the setpoint for a temperature threshold.

➢ **Step 5: Program Testing and Debugging**

1. **Simulate the Program**:

✓ Before deploying the program to actual hardware, use the simulation tool available in your PLC software (e.g., TIA Portal) to test the logic.

✓ Run the simulation to verify that:

- All inputs trigger the expected outputs.
- Timers, counters, and comparators behave as intended.

2. **Step-by-Step Testing**:

Test small portions of the program one step at a time:

✓ First, verify that the motor turns on when the button is pressed.

✓ Then, check that the motor turns off if the temperature exceeds the limit.

✓ Finally, test any additional logic like alarms or timers.

3. **Monitor Variables**:

Use the **monitoring mode** to observe real-time values of inputs, outputs, and internal variables (like counters or timers) during testing.

4. **Fix Issues and Debug**:

If the program does not behave as expected:

✓ Check the wiring or input/output assignments.

✓ Re-examine the logic flow and connections between function blocks.

✓ Adjust parameters like delay times or comparison thresholds if necessary.

5. **Deploy the Program**:

✓ Once testing is complete and all issues are resolved, download the program to the PLC for actual operation.

✓ Run the program on the real hardware and monitor the system's performance to ensure it behaves as expected in a live environment.

Example: Simple Motor Control System

1. **System Requirements**:

✓ Input: Start Button, Temperature Sensor.

✓ Output: Motor Control, Alarm Light.

✓ Logic: Turn on the motor when the button is pressed and the temperature is below 50°C. Turn off the motor if the temperature exceeds 50°C.

2. **Function Blocks Selected**:

✓ **AND** Block: To ensure both the button and temperature conditions are met.

✓ **Comparator (Less Than)**: To check if temperature is below 50°C.

✓ **Motor Output**: Controls the motor.

3. **Build the Logic**:

✓ Connect the **Start Button** and **Temperature Sensor** to the **AND** block.

✓ Link the **AND** block to the **Motor Control** output.

4. **Configure Parameters**:

Assign inputs/outputs and set the comparator threshold to 50°C.

5. **Test and Debug**:

Simulate the program to ensure the motor turns on/off correctly based on the inputs.

➢ **Testing and Debugging of Function Block Diagram (FBD)**

Testing and debugging in Function Block Diagram (FBD) programming language is a crucial part of developing a reliable and efficient control system. Here's an overview of the process, including key steps and techniques.

1. Initial Review of the FBD Design

- **Check for Logical Flow**: Review the overall structure of the FBD to ensure that the flow of logic is correct. Verify that all function blocks are connected appropriately.
- **Validate Function Blocks**: Ensure that each function block is properly configured with correct parameters and that the expected data types for inputs and outputs are used.

2. Simulation Testing

- **Run Simulations**: Utilize the built-in simulation tools in the programming environment (such as TIA Portal) to run the FBD program without connecting to the physical hardware.

- **Test Scenarios**: Create various test scenarios, simulating different input conditions to observe how the function blocks respond. This helps in validating the logic flow.
- **Check Output Responses**: Monitor the output of function blocks to ensure they are generating the expected results based on the simulated inputs.

3. Step-by-Step Debugging
- **Step Through the Logic**: Use the debugging tools to step through the program one block at a time. This allows you to observe how the program executes and identify where it might be failing.
- **Monitor Variables**: Keep an eye on input and output variables as well as internal states of function blocks during execution. This helps pinpoint any discrepancies between expected and actual behavior.

4. Error Handling and Exception Testing
- **Introduce Fault Conditions**: Simulate fault conditions (e.g., sensor failures, unexpected input values) to test how the program handles errors.
- **Validate Alarm and Fault Logic**: Ensure that any alarm or fault logic is triggered correctly under abnormal conditions.

5. Documentation Review
- **Check Documentation**: Verify that all function blocks and their parameters are documented clearly. This helps in understanding the expected behavior during testing.
- **Review Comments**: Make sure any comments in the FBD are accurate and helpful for identifying the purpose of each block.

6. System Integration Testing
- **Connect to Hardware**: Once simulation testing is successful, upload the program to the actual PLC and connect it to the relevant hardware.
- **Perform Functional Testing**: Conduct tests on the real system to ensure that the FBD operates as intended under real-world conditions.
- **Check Interfacing with Other Systems**: If the FBD interacts with other systems or components, validate that data is being exchanged correctly.

7. Final Validation
- **Conduct Acceptance Testing**: Perform a series of acceptance tests based on the system requirements to ensure that all functionality meets the design specifications.
- **User Training and Feedback**: If applicable, train users on how to operate the system and gather feedback for any potential improvements.

8. Revision and Optimization

- **Revise Based on Findings**: If issues are found during testing, revise the FBD design and logic as necessary.
- **Optimize Logic**: After successful testing, consider optimizing the FBD for efficiency, reducing complexity, or improving readability.

Practical Activity 2.3.3: Using Structured Control Language (SCL)

Task:

1: You are requested to perform the given task. The task should be done individually.
You are asked to go in computer lab and write PLC program by using Structured Control Language (SCL)
2: Present your work to the class or colleagues
3: Ask for any clarification if any.
4: Read the Key readings 2.3.3 in their manuals

Key readings 2.3.3: Using Structured Control Language (SCL)

➢ **System Requirements of Structured Control Language (SCL)**

1. **Understanding the Application**:

✓ Gain a comprehensive understanding of the automation process to be controlled, such as equipment functionality, operational environment, and specific control goals.

2. **Identifying Inputs and Outputs**:

✓ Determine all the inputs (sensors, switches, etc.) and outputs (actuators, motors, lights) that interact with the system. Define their roles and establish a mapping that will be used within the SCL program.

3. **Operational Logic**:

✓ Outline the step-by-step control logic required for the system's operation. This includes any sequences, conditions, or actions that the program should follow based on input states.

4. **Performance Metrics**:

✓ Define the performance requirements for the system, such as response time, accuracy, or efficiency, to ensure the program meets operational needs.

➢ **Program Structure in SCL**

An SCL program consists of several main components:

1. **Header Section**:
   - This section includes metadata, such as the program title, version, author, and date, helping with program identification and version control.
2. **Variable Declaration Section**:
   - Declare all variables used within the program, including inputs, outputs, timers, counters, and internal variables. Variables are typically defined with data types such as BOOL, INT, REAL, or STRING.
3. **Function and Function Block Declarations**:
   - Define any functions or function blocks that perform reusable tasks within the program. These modular blocks improve code readability and reduce redundancy.
4. **Main Program Body**:
   - The core logic is implemented here, utilizing control statements like IF-THEN-ELSE, FOR, and WHILE loops. This section directs the operations of the SCL program according to system requirements.
5. **End of Program**:
   - Concludes the SCL code, ensuring that all actions within the program are completed correctly before ending.

➢ **Coding Logic with Control Statements in SCL**

- **Control Statements**:

❖ Control statements like IF, CASE, WHILE, and FOR loops enable the program to execute conditional and iterative tasks. For instance:

- **IF-THEN-ELSE**: Used for conditional decisions, such as turning on a motor if a sensor is active.
- **FOR/WHILE Loops**: Repeats actions, useful for counting events or managing repetitive operations.
- **CASE**: Directs actions based on multiple conditions or states, enabling more efficient decision-making in complex systems.

- **Logical Operators**:

❖ Operators like AND, OR, and NOT are used to build complex logical expressions, allowing the program to evaluate multiple conditions simultaneously.

➢ **Interfacing with Input/Output (I/O) Devices in SCL**

- **Reading Inputs**:

❖ SCL can read inputs from sensors or other devices, usually through predefined memory locations or addresses. Each input is mapped to a variable that the program reads to check the device's status.

- **Controlling Outputs**:

❖ Outputs like motors, lights, and alarms are managed by writing values to specific addresses, changing the device's state (on/off or active/inactive).

- **Data Conversion and Scaling**:

❖ SCL programs often perform conversions or scaling on analog inputs (e.g., converting raw sensor data to meaningful units like temperature or pressure).

- **Communication with External Systems**:

❖ SCL can integrate communication protocols to interact with external systems, such as Human-Machine Interfaces (HMIs) or other PLCs, for coordinated control.

➢ **Testing and Debugging in SCL**

1. **Initial Code Review**:

❖ Before running the program, perform a review of the code for syntax errors and logical inconsistencies.

2. **Simulated Testing**:

❖ Use simulation tools in the SCL development environment to test the program without actual hardware. This allows checking the control logic's behavior in response to simulated inputs.

3. **Live Testing**:

❖ With hardware, monitor real input/output responses to validate that the program performs as expected in a live environment.

4. **Error Handling and Debugging**:

❖ Add debugging tools and error-handling code, such as temporary print statements or alarms, to identify and resolve any faults in the code.

5. **Iterative Testing**:

❖ After identifying issues, correct the code and re-test until the program consistently meets all system requirements.

Points to Remember

- Ladder Diagram (LAD) is a graphical programming language widely used in Programmable Logic Controllers (PLCs) for industrial automation

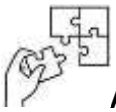
- Developing a logic ladder diagram involves a structured approach to visualize logical operations and control processes
- Translating logic into rungs involves converting a control logic diagram (like a flowchart or a logical description) into a Ladder Diagram (LAD) format. Each rung of a ladder diagram represents a logical operation or sequence, typically corresponding to relay logic
- Testing a program is conducted to verify a software system's functionality, performance, and reliability to identify defects or errors
- Debugging a program is investigating and resolving those defects, aiming to eliminate issues and ensure smooth operation. As you see at the bottom, all errors are corrected well
- Documentation: It is crucial to document the debugging process, including the observed symptoms, root cause, applied fixes, and testing outcomes. Documentation is a reference for future debugging efforts, aids knowledge sharing, and helps maintain a record of resolved issues.
- Maintenance refers to the process of ensuring that the TIA Portal project remains functional and up to date throughout its lifecycle
- System requirements on Function Block Diagram (FBD)
  - ✓ Understand the Process or System
  - ✓ Identify Inputs and Outputs
  - ✓ Determine Control Logic
  - ✓ Set Timing and Safety Requirements
- Logical Blocks Example: Use AND, OR, and NOT blocks to handle control conditions (e.g. Start AND Temperature OK).
- Program Testing and Debugging
  - ✓ Simulate the Program
  - ✓ Step-by-Step Testing
  - ✓ Monitor Variables
  - ✓ Fix Issues and Debug
- The system requirements of Structured Control Language (SCL)
  - ✓ Understanding the Application
  - ✓ Identifying Inputs and Outputs
  - ✓ Operational Logic
  - ✓ Performance Metrics
- Program Structure in SCL

An SCL program typically consists of the following main components

1. Header Section

2. Variable Declaration Section
3. Function and Function Block Declarations
4. Main Program Body
5. End of Program

- Coding logic with control statements in Structured Control Language (SCL) is essential for implementing decision-making processes and repetitive tasks in automation systems.
- Interfacing with Input/output (I/O) devices in Structured Control Language (SCL) is essential for enabling communication between the control program and the physical world.
- Testing and debugging are critical stages in the development of programs in Structured Control Language (SCL) to ensure that the code functions as intended and meets the specified requirements. Here's an overview of the testing and debugging processes in SCL.
- Debugging in PLC simulation software is the process of identifying, isolating, and resolving errors or malfunctions in a simulated PLC program or configuration.



Application of learning 2.3.

A storage tank requires an automated control system to maintain water levels efficiently. The system should activate the water pump when the water level falls below a certain threshold, as indicated by a low-level sensor, and keep it running until the water reaches the high-level threshold. Once the high level is reached, the pump should continue to operate for an additional 15 seconds to stabilize the level before shutting off. As automation technician Your task is to develop a ladder logic program for this system. The program should ensure that the pump starts when the low-level sensor detects a low water level and continues to run until the high-level sensor detects a high-water level, at which point a 15-second delay is triggered before the pump stops. Please verify that the sensors and timer function as intended, allowing for consistent water level control within the tank.

Indicative content 2.4: Converting PLC Programming Language

Duration: 8 hrs

Practical Activity 2.4.1: Converting ladder diagram to structure control language (LAD to SCL)

Task:

1: Referring to the key readings 2.4.1. You are requested to perform ladder diagram to structure control language (LAD to SCL):

As automation technician in our warehouse, you have been tasked with upgrading the conveyor belt sorting system, which currently relies on a Ladder Diagram (LAD) program to sort packages by size. The existing LAD program controls sensors, motors, and actuators but lacks the flexibility and data tracking needed to meet increasing demands. Management has assigned you to convert this LAD program into Structured Control Language (SCL) to unlock more advanced functionality. Your task involves translating LAD rungs—including start/stop functions, package size detection, diversion mechanisms, and package counting—into structured SCL code.

2: Follow instructions and procedures provided in that key readings and perform the given task

3: Present your final products to the whole class/trainer

4: Ask for clarification if any

Key readings 2.4.1: Converting Ladder Diagram (LAD) to Structured Control Language

Converting a simple Ladder Diagram with conditions controlling an output into equivalent Structured Control Language.

Example LAD Logic:

- **Inputs**: Start button (I0.0), Stop button (I0.1), and a Temperature sensor (IW64).
- **Output**: Motor (Q0.0).
- **Logic**: The motor turns ON when the Start button is pressed and the temperature is below 50°C, and it turns OFF when the Stop button is pressed.

Step 1: Understand the Ladder Logic

- **LAD Logic** might look something like this:
  - Input I0.0 (Start Button) is connected to a contact.
  - Input I0.1 (Stop Button) is connected to a normally closed contact.

- A comparator checks if the analogue input IW64, (temperature sensor) is less than 50.0.
- If all these conditions are meeting, the output coil Q0.0 (Motor) is energized.

Step 2: Translate Ladder Logic to SCL

1. **Open the Ladder Diagram in TIA Portal**:

View the LAD diagram you want to convert.

2. **Analyze the Logic**:

Note down each rung and its components (input contacts, outputs, comparators, etc.).

3. **Create a New SCL Block**:

Right-click on **Program Blocks** > **Add New Block** > Select **Function (FC)** or **Function Block (FB)** > Choose **SCL**.

4. **Define Inputs and Outputs in SCL**:

Translate the LAD inputs and outputs into variables.

- Example:

```
scl
Copy code
VAR_INPUT
  Start Button: BOOL; // Start button (I0.0)
  Stop Button: BOOL; // Stop button (I0.1)
  Temp Sensor: REAL; // Temperature sensor (IW64)
END_VAR

VAR_OUTPUT
  Motor: BOOL; // Motor output (Q0.0)
END_VAR
```

5. **Write Equivalent Logic in SCL**:

Use **IF-THEN-ELSE** statements to mimic the Ladder conditions:

```
scl
Copy code
BEGIN
  // Start motor if Start button is pressed, stop button is NOT pressed, and temperature is below 50°C
  IF Start Button = TRUE AND Stop Button = FALSE AND TempSensor < 50.0 THEN
    Motor: = TRUE; // Motor ON
```

```
  ELSE
    Motor: = FALSE; // Motor OFF
  END_IF;
END_FUNCTION_BLOCK
```

6. 6. **Simulate and Debug**:

Run the simulation in **TIA Portal** to check the functionality of the SCL block against the original LAD logic.

Practical Activity 2.4.2. Converting Ladder Diagram to Function Block Diagram (LAD to FBD)

Task:

1: Referring to the key readings 2.4.2. you are requested to Perform Ladder Diagram to Function Block Diagram (LAD to FBD).

You are a technician tasked with upgrading a motor control system to improve its readability and modularity. Currently, the system operates with a Ladder Diagram (LAD) where pressing a "Start" button turns the motor on, and pressing a "Stop" button turns it off. To make this system easier to troubleshoot, you need to convert the control logic to a Function Block Diagram (FBD). Your task is to design the FBD by selecting appropriate function blocks (such as Set/Reset latches or logic gates) to replicate the start/stop functionality of the motor.

2: Follow instructions and procedures provided in that key readings and perform the given task

3: Present your final products to the whole class/trainer

4: Ask for clarification if any

Key readings 2.4.2: Converting Ladder Diagram (LAD) to Function Block Diagram (FBD)

Example LAD Logic

- **Inputs**: Push button (I0.0) and Temperature sensor (IW64).
- **Output**: Motor (Q0.0).
- **Logic**: The motor turns ON when the button is pressed, and the temperature is below 50°C.

Step 1: Understand the Ladder Logic

- **LAD Logic** might have the following elements:

✓ Input I0.0 (Push Button) is connecting to a contact.
✓ A comparator checks if the temperature value from IW64 is less than 50.0.
✓ Both conditions must be true to energize the coil Q0.0 (Motor).

Step 2: Create a New Function Block Diagram (FBD)

1. **Open the Ladder Diagram in TIA Portal**:
   - Start by opening the existing Ladder Diagram to understand the logic you want to convert.
2. **Create a New FBD Block**:
   - Right-click on **Program Blocks** > **Add New Block** > Select **Function (FC)** or **Function Block (FB)** > Choose **FBD**.

Step 3: Translate Ladder Logic into FBD

1. **Add Input Contacts in FBD**:
   - In the FBD editor, add input contacts for the start button and comparator.
     - **Add a Contact** for the button (I0.0).
2. **Add Comparators and Logic Gates**:
   - Use a **Comparator Block** to check the temperature value:
     - In the FBD editor, insert a **Compare (LT)** block to check if the temperature is less than 50°C.
     - Configure the input as IW64 and the comparison value as 50.0.
3. **Connect Logic with AND Gate**:
   - Use an **AND** gate to ensure that both conditions (button pressed and temperature below threshold) are true.
     - Connect the output of the button and the output of the comparator to the inputs of the **AND** gate.
4. **Control the Output Coil (Motor)**:
   - Add a **Coil** at the output of the AND gate and assign it to the motor output (Q0.0).

Step 4: Assign Inputs and Outputs

1. **Map the Inputs and Outputs**:
   - Connect the variables to actual I/O addresses in the FBD editor:
     - I0.0 for the push button.
     - IW64 for the temperature sensor input.
     - Q0.0 for the motor output.
2. **Simulate and Test**:

- Simulate the FBD logic to verify that it behaves the same as the original LAD logic.
- Test with real PLC hardware to confirm correct operation.

Points to Remember

- The steps for converting a Ladder Diagram (LAD) program into Structured Control Language (SCL) for PLC programming:

  1. Understand and Analyze the Ladder Diagram (LAD)
  2. Define SCL Variables
  3. Translate Contacts to Boolean Expressions
  4. Convert Series and Parallel Logic
  5. Translate Coils as Output Assignments
  6. Translate Timers and Counters
  7. Combine and Test the Code
  8. Document the Code

- The steps for Converting Ladder Diagram (LAD) to Function Block Diagram (FBD)

  Step 1: Understand the Ladder Logic

  Step 2: Create a New Function Block Diagram (FBD)

  Step 3: Translate Ladder Logic into FBD

  Step 4: Assign Inputs and Outputs

Application of learning 2.4.

As an industrial automation technician, you are tasked with upgrading an existing motor control system for a production line. The current control logic uses a Ladder Diagram (LAD) and allows operators to start and stop the motor with two buttons: A Start button to initiate motor operation and a Stop button to halt it. To modernize this system, you need to convert the Ladder Diagram into both Structured Control Language (SCL) and Function Block Diagram (FBD) formats.

Learning outcome 2 end assessment

Written assessment

A. Open questions

1. What does a normally open (NO) contact in a ladder logic program represent?
2. Explain the function of a normally closed (NC) contact in a ladder logic program.
3. What is the purpose of a relay coil in a ladder logic program?
4. Define an On-Delay Timer (TON) in a ladder logic program.
5. What is a Counter (CTU) in a ladder logic program?

B. Multiple Choice Questions:

Read the following statement and answer by circling a letter corresponding to the correct answer

1. In Ladder Logic, what does a normally open (NO) contact represent?
    a. A closed switch when active
    b. An open switch when inactive
    c. A relay coil
    d. A condition that is always true
2. Which type of PLC memory is used to store the user program?
    a. RAM
    b. ROM
    c. EPROM
    d. Flash Memory
3. What is the purpose of a timer in a PLC program?
    a. To count the number of events
    b. To measure the time between events
    c. To delay actions in the program
    d. To control the program flow
4. Which software is commonly used for programming Siemens PLCs?
    a. TIA Portal
    b. Visual Studio
    c. MATLAB
    d. AutoCAD
5. Which of the following is a typical output device in a PLC system?
    a. Sensor
    b. Push Button
    c. Actuator

    d. Thermocouple
6. Which software is used for PLC simulation in this context?
    a. AutoCAD
    b. TIA Portal
    c. MATLAB
    d. Proteus
7. Before installing PLC software, what check should be performed?
    a. Network connection check
    b. System compatibility check
    c. Calibration check
    d. Firmware update check
8. Which programming language represents system logic with rungs?
    a. Function Block Diagram (FBD)
    b. Structured Control Language (SCL)
    c. Ladder Diagram (LAD)
    d. Sequential Function Chart (SFC)
9. In PLC programming, which language uses function blocks to construct logic?
    a. SCL
    b. FBD
    c. LAD
    d. STL
10. Which criterion evaluates how easily a programming language can be learned by new users?
    a. Scalability
    b. Interoperability
    c. Security
    d. Learning curve
11. Which of the following statements is true about SCL?
    a. It uses graphical symbols for programming.
    b. It is based on structured control statements.
    c. It is primarily used for creating ladder diagrams.
    d. It lacks debugging capabilities.
12. What is the primary goal of testing and debugging a PLC program?
    a. To ensure system compatibility
    b. To improve scalability
    c. To detect and correct program errors
    d. To update system documentation
13. In Ladder Diagram programming, what is used to represent the process logic?
    a. Function blocks
    b. Code structures
    c. Rungs

d. Statements

14. Which PLC programming language requires defining both system requirements and block parameters for configuring program logic?
    a. Ladder Diagram (LAD)
    b. Structured Control Language (SCL)
    c. Function Block Diagram (FBD)
    d. Sequential Function Chart (SFC)
15. When converting from Ladder Diagram to Structured Control Language, the logic is translated into _____.
    a. rungs
    b. control statements
    c. functions
    d. block diagrams

C. Fill-in-the-Blank Questions

Read the following statement and complete them by using correct terms: **Ladder Diagram (LAD), Function Block Diagram (FBD), scalability, parameters, system compatibility, control, TIA Portal, control, rungs, function blocks, structured, Debugging,**

1. The main software used to simulate PLC programs in this setup is called _____.
2. To ensure compatibility, it is necessary to perform a _____ check-up before installing PLC simulation software.
3. The two main PLC programming languages used to define system requirements and develop logic are _____ and _____.
4. In Ladder Diagram programming, system logic is represented in _____ that simulate electrical relay logic.
5. The SCL programming language is known for using _____ statements to control logic and flow.
6. One advantage of Function Block Diagram (FBD) is its use of _____ for building logic, making it suitable for process-oriented applications.
7. One of the selection criteria for a PLC programming language is _____, which refers to the language's adaptability to expanding applications.
8. _____ is a method of testing a PLC program to find and fix any errors before deploying it in a live system.
9. Converting a Ladder Diagram to Structured Control Language (SCL) requires translating rungs into _____ logic statements.
10. In FBD, configuring block _____ allows the program to be tailored to specific input and output requirements.

Practical assessment

As an automation technician, you are tasked with setting up PLC simulation software and program a new production line at a manufacturing plant. You must install and configure TIA Portal software to simulate the system, which will control conveyor belts, robotic arms, and sorting sensors. First, perform a system compatibility check to ensure the software runs smoothly on the available computer system. Once installed, run and configure the software packages according to the plant's specifications. After setting up the simulation environment, choose appropriate PLC programming languages to define the control logic for each part of the production line, taking into account the complexity of each task.

Reference

Books

Dzinic, J., & Yao, C. (2014). Simulation-based verification of PLC programs.

Frey, G., & Litz, L. (2000, October). Formal methods in PLC programming. In Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0 (Vol. 4, pp. 2431-2436). IEEE.

Park, C. M., Bajimaya, S. M., Park, S. C., Wang, G. N., Kwak, J. G., Han, K. H., & Chang, M. (2006, November). Development of virtual simulator for visual validation of PLC program. In *2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06)* (pp. 32-32). IEEE.

Park, S. C., Park, C. M., & Wang, G. N. (2008). A PLC programming environment based on a virtual plant. *The international journal of advanced manufacturing technology*, *39*, 1262-1270.

Web links

https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbU9qYmdrMVVnUnFMZXlNMjJ0ZVo1MDJxdTM5UXxBQ3Jtc0tucXJDOGhyTFRCXzVJdGJwNW00SHJGUi1OYUVkaW53X0Zla19Hbi0wM25LMWM2Tkx0b2ZlRTBfM1ZvX0lLa29kRy1CeHlZRUliQk5faVVRd2xmNlpTTlpyVUFsM1VWTER1R3U0akRIRmpxeDd3S3h1VQ&q=https%3A%2F%2Fdrive.google.com%2Ffile%2Fd%2F16F65N2xPyqaCC5y1p53q6nwcviejRL-0%2Fview%3Fusp%3Dsharing&v=XhpJBh5ctpk

(https://www.amazon.com/PLC-Programming-Industrial-Automation-Collins/dp/0071810455)

Indicative contents
3.1 Downloading PLC program
3.2 Testing automation system
3.3 Documenting automation system
3.4 Maintaining automation system

Key Competencies for Learning Outcome 3: Deploy PLC program

Knowledge	Skills	Attitudes
• Description of PLC programming concepts • Description of testing technics	• Downloading PLC program • Flash the program file to PLC • Testing automation system • Documenting automation system • Maintaining automation system	• Being patient • Being Analytical and details oriented • Being adaptable • Being Accurate

Duration: 20 hrs

Learning outcome 3 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Download correctly PLC program based on downloading techniques.
2. Test correctly automation system according to intended use.
3. Document properly automation system according to system operation.
4. Maintain properly automation system according to system functionality

Resources

Equipment	Tools	Materials
• Conveyor belt system with motor and drive mechanism • PLC modules (e.g., Siemens S7-1200) • PLC input and output modules • PLC Power Supply • Contactors • Safety gloves (e.g., Mechanix Wear, Ansell) • Protective clothing (e.g., coveralls, safety vests) • HMI devices (e.g Siemens wincc) • Multimeter • Personal computer	• Pliers • Wire stripper • Allen wrench • PLC programming software (TIA Portal) • Screw drivers	• Electrical wires (various gauges) • Relay • Cable ties • Sensors • Labels • Ethernet cables/USB Cable • Power cables

Indicative content 3.1: Downloading PLC Program

Duration: 4 hrs

Theoretical Activity 3.1.1: Introduction to program file preparation

Tasks:

1: You are invited to answer the following questions related to Program file preparation:

i. What do you understand by program file preparation?
ii. Describes the steps required in PLC program file preparation.
iii. Differentiate method of connection and depth connection used in PLC

2: Present your findings to the whole class and Trainer

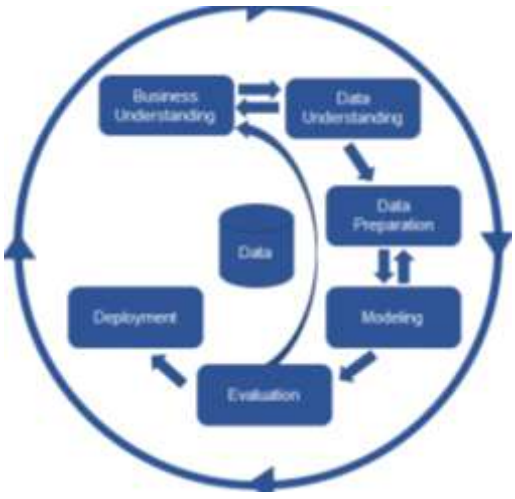
3: Ask trainer for any clarification if any

4: You are directed to read the key readings 3.1.1. for more clarifications

Key readings 3.1.1: Introduction to program file preparation

1.Program file preparation

Program file preparation is a critical stage in working with a PLC (Programmable Logic Controller) system. It involves developing, organizing, and testing the program before it is transferred to the PLC for controlling an industrial process or machinery.



The steps and considerations in plc program file preparation

1. Understand the Control Requirements

Before writing any program, you must understand the system or process to be controlled.

This includes:

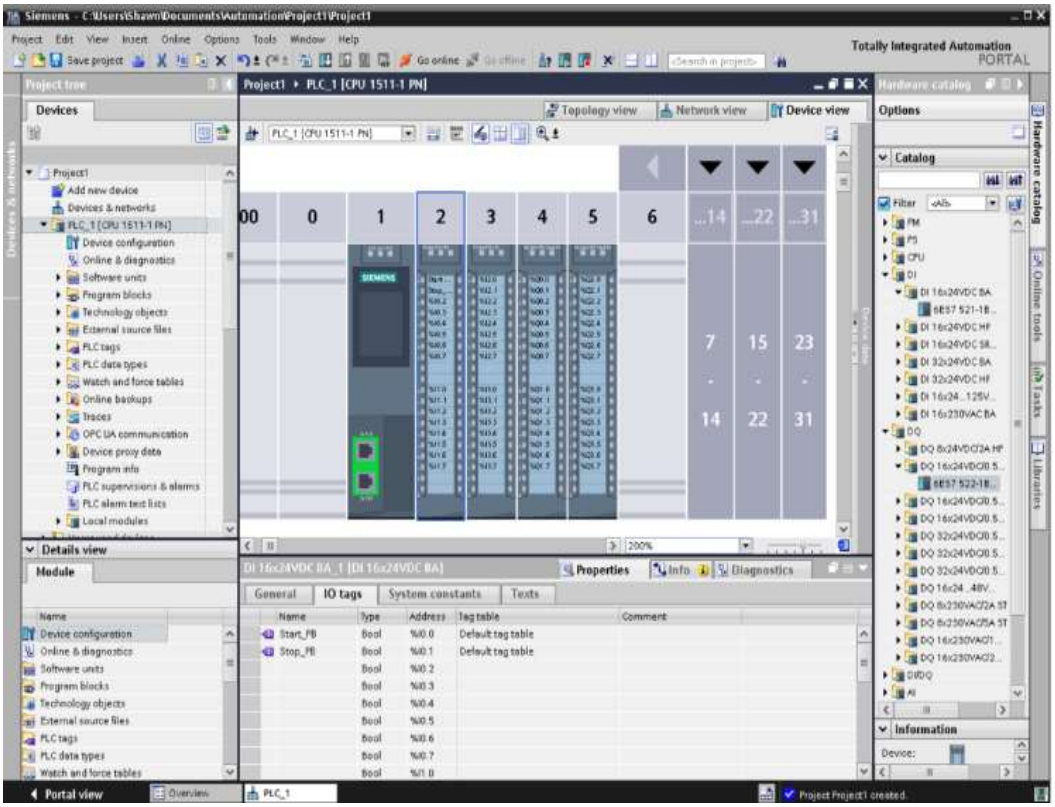
- **Specifications and objectives**: Understanding what needs to be automated or controlled.
- **I/O requirements**: Defining the inputs (sensors, switches, etc.) and outputs (motors, solenoids, etc.).

Safety considerations: Ensuring safety interlocks and emergency stop functions are included.

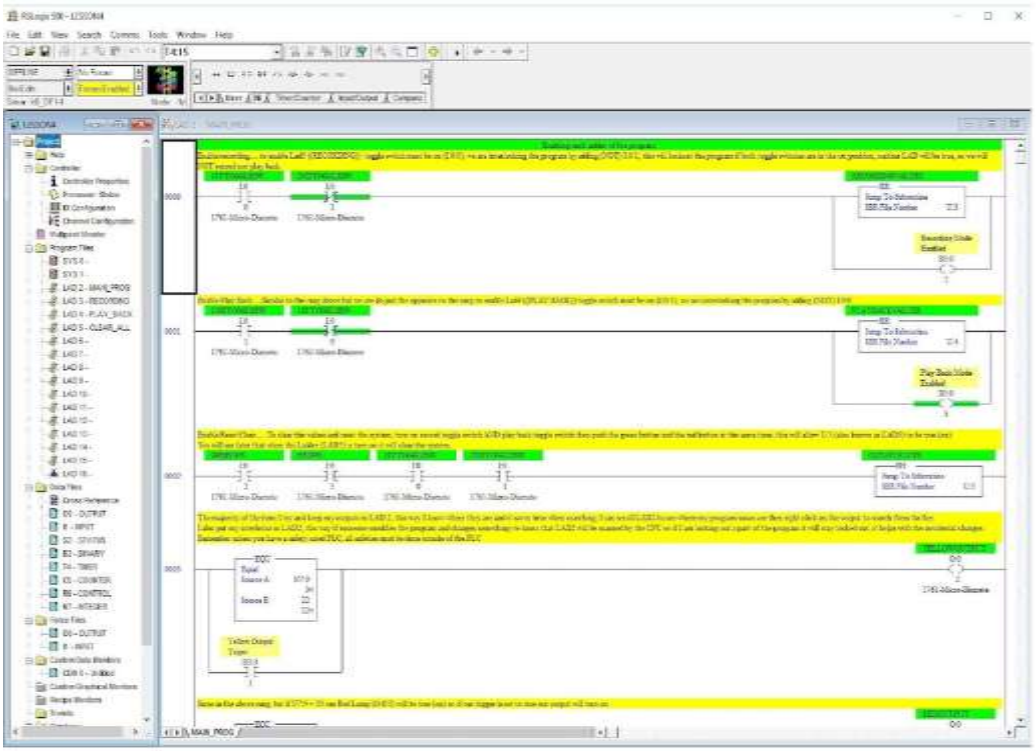
2. Selecting the Right Programming Software

The choice of programming software depends on the PLC brand and model you're using. For example:

- **Siemens**: TIA Portal or STEP 7



- **Allen-Bradley**: RSLogix 500, Studio 5000



- **Schneider Electric**: EcoStruxure Machine Expert (formerly SoMachine)

Once the correct software is installed, it's important to ensure that the **PLC type**, **firmware version**, and **communication drivers** are correctly set up for compatibility with the hardware.

3. Organizing the Program Structure

Organizing the program well ensures it is easy to understand, troubleshoot, and modify. Common practices include:

- **Modular Programming**: Break the logic into **function blocks** or **subroutines**. Each block or subroutine should handle specific tasks, such as motor control, safety, or data acquisition.
- **Naming Conventions**: Use clear and descriptive names for **variables**, **tags**, and **function blocks** (e.g., Start_Motor_Conveyor1).
- **Flowcharting/Sequential Function Charts (SFC)**: For complex processes, consider using flowcharts to map out the sequence of operations before writing the code. Some PLC programming environments allow you to implement sequences with SFC.

4. Configuring I/O Mapping

Every PLC program interacts with physical devices via inputs and outputs. Mapping these devices is crucial to ensure proper functioning. This involves:

- **Input Mapping**: Assigning physical inputs (e.g., sensors, push buttons) to memory locations within the PLC.
- **Output Mapping**: Assigning physical outputs (e.g., relays, motors) to specific output addresses in the PLC.
- **Internal Variables**: For intermediate processing or storing temporary values, define internal variables that don't correspond to actual I/O.

Most programming software includes tools to map the I/O efficiently, making this step more streamlined.

5. Writing the Control Logic

PLC programs are written using one of several languages, each suitable for different applications. The most common languages include:

- **Ladder Logic (LD)**: Resembles relay-based control systems, making it ideal for discrete control.
- **Function Block Diagram (FBD)**: Useful for continuous processes and controlling multiple devices.
- **Structured Text (ST)**: A high-level programming language resembling Pascal, good for complex calculations or data processing.
- **Instruction List (IL)**: A low-level, assembly-like language, rarely used today but useful for performance-critical applications.

The program logic should be written carefully to follow the operational sequence and meet the control requirements, which includes:

- **Logical operations** (AND, OR, NOT)
- **Timers, counters**, and **counters**
- **Math operations** (if required for process control)
- **Control loops** (such as PID control for temperature, pressure, etc.)

6. Error Checking and Simulation

After writing the program, error checking is crucial:

- **Syntax Check**: Most PLC software automatically checks for syntax errors (e.g., unmatched parentheses, invalid instructions).
- **Logical Errors**: Logical errors can be subtler. To address this, simulate the program within the PLC software. Simulation allows you to test the logic by providing mock inputs to verify that the outputs behave as expected.
- **Debugging**: Debugging tools in the software can help you step through the program logic and observe how variables change during execution. This can catch errors in the control flow or unexpected behavior.

7. Communication Settings

A properly prepared program must be able to communicate with the PLC hardware. This involves:

- **Selecting the communication protocol**: Ethernet, Modbus, Profibus, or RS232, depending on the PLC and the devices involved.
- **Setting the IP address**: For Ethernet communication, assigning a static IP to the PLC ensures consistent network access.
- **Configuring baud rate and parity settings**: For serial communication protocols like RS232 or RS485, setting correct parameters ensures data is transmitted correctly.

8. Testing the Program in Real-Time

Before downloading the program, the following should be done:

- **Load the program into a test PLC** (if possible) to run real-time tests with actual inputs and outputs.
- **Observe the behavior of I/O**: Check if inputs trigger the expected outputs and if the system behaves as designed.
- **Make adjustments**: If necessary, modify the logic based on test results and troubleshoot any hardware-related issues like sensor miswiring.

9. Finalizing and Saving the Program

Once the testing is complete, prepare the program for download:

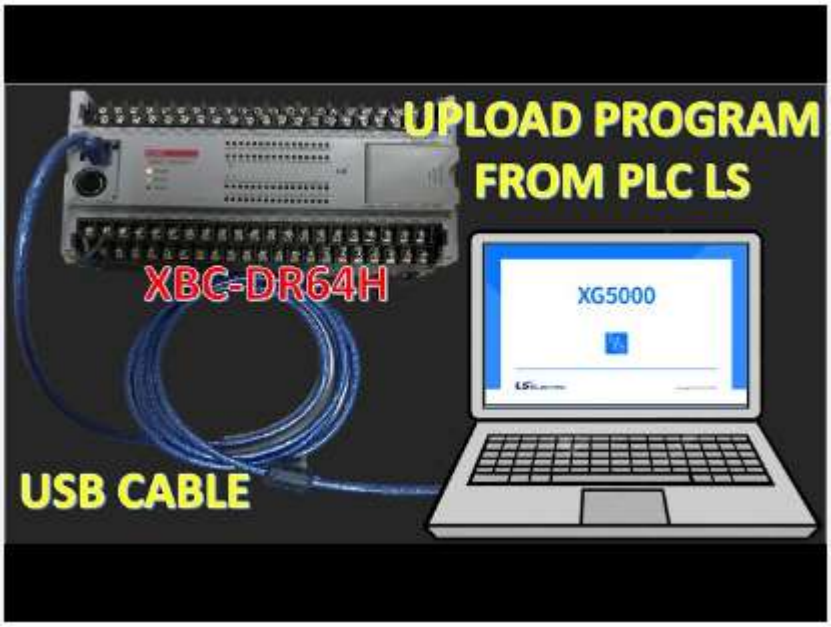
- **Backup the final version**: Ensure you save a final, tested version of the program in multiple locations, including external storage, for future reference.
- **Prepare documentation**: It's essential to document the program well, including **comments within the code**, **I/O descriptions**, and **functional explanations**. This makes future troubleshooting or modifications easier.

2.Select connection method

- **Method of connection**

When downloading a PLC program, it is crucial to select an appropriate connection method between your programming device (PC or laptop) and the PLC. Common methods include

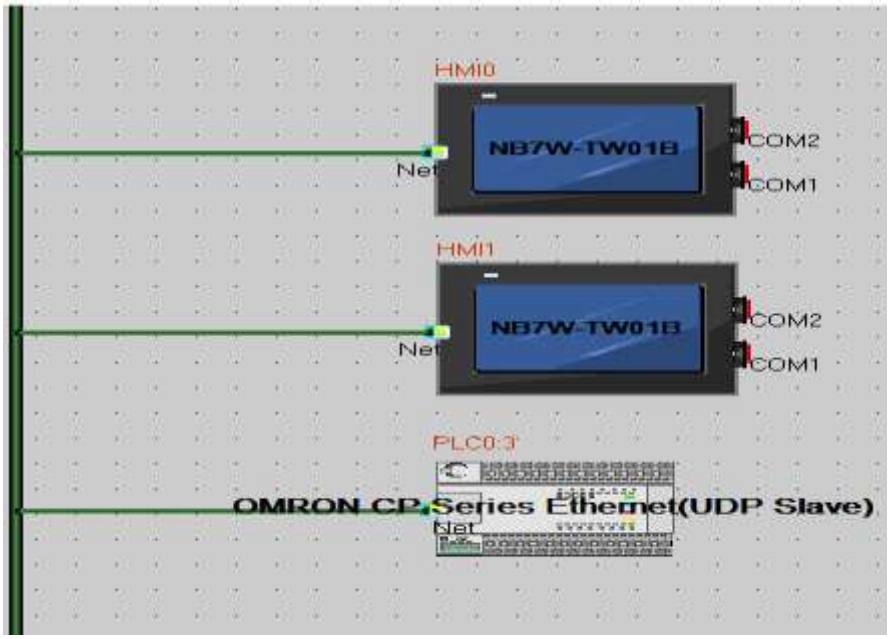
1. **USB Setup:**



- Plug the USB cable into the PLC and the PC.
- The programming software should detect the PLC automatically. If not, check that the correct drivers are installed.

2. **Ethernet Setup:**

HMI (2 pcs) to CP1 PLC over Ethernet (FINS/UDP)



•Connect an Ethernet cable from the programming device to the PLC, or connect both to the same local network.

•Assign IP addresses to both devices if needed:

▪Set the PLC and the PC on the same subnet (e.g., PC: 192.168.0.10, PLC: 192.168.0.20).

▪Ping the PLC from the PC to ensure connectivity.

3. **RS-232/RS-485 Setup:**

- Connect the serial cable (RS-232 or RS-485) between the PC's serial port (or USB-to-serial adapter) and the PLC.
- Set up communication parameters (e.g., baud rate, parity, stop bits) in the programming software to match the PLC's settings.

4. **Wireless Setup:**

- Ensure that both the PLC and the PC are connected to the same wireless network.
- Configure the wireless network parameters in the PLC, if required.
- Ping the PLC to ensure the wireless connection is stable.

Depth of connection

Depth of connection refers to the thoroughness and reliability of the communication link between the programming device (PC/laptop) and the PLC. It ensures that the connection is robust enough for transferring programs, performing diagnostics, and handling data exchanges without interruptions or errors.

Considerations for assessing and ensuring the depth of connection

1. **Connection Stability**

Stability refers to how consistently the communication is maintained without disconnections or interruptions. This is especially important for remote or long-term programming and data monitoring.

- For USB connections, check that the cable is securely plugged and not prone to disconnections due to movement.
- For Ethernet or wireless connections, monitor the network stability to ensure no packet loss or dropped connections. Tools like **ping** can help assess this.

- Test the stability of the connection by performing basic read/write operations from the PLC programming software and monitoring for any timeouts or failed data transfers.

2. **Communication Speed and Latency**

Speed is the rate of data transfer between the programming device and the PLC. Latency refers to the time delay between sending and receiving signals.

- **USB and Ethernet** typically offer high speeds and low latency, while **RS-232/RS-485** connections may be slower, with limited data rates.
- **Ethernet** and **wireless** connections should be monitored for latency, especially if connected over larger networks or remotely.

Use tools such as ping to measure the latency in an Ethernet network. Latency should be low (e.g., < 10 ms) for a strong connection.

3. Bandwidth and Data Throughput

Bandwidth refers to the capacity of the connection to handle large volumes of data. For some applications, especially those involving real-time data, a higher bandwidth may be required.

- For **high-throughput systems**, Ethernet is ideal due to its higher bandwidth capacity.
- **Wireless** connections may suffer from bandwidth limitations if too many devices are sharing the same network.

- **Check the network load or interference sources (in case of wireless) and ensure sufficient bandwidth is available for programming, diagnostics, and control**.

4**. Signal Integrity (for Serial and Ethernet)**

Signal integrity refers to the quality of the electrical signals in the connection. Degraded signals can lead to errors in communication, especially in serial (RS-232/RS-485) or long Ethernet cables.

- For **RS-232/RS-485**: Ensure proper grounding and use shielded cables to reduce noise.
- For **Ethernet**: Check for proper cable termination and verify that cables are within the recommended length (typically less than 100 meters for Cat5/6 cables).

- Monitor error rates or packet drops. A high number of communication errors may indicate poor signal integrity and require troubleshooting.

5. **Error Detection and Correction**

In many PLC communication protocols, built-in error detection (e.g., CRC or checksum) ensures that data transmitted between the programming device and the PLC is correct. If errors occur, the system can either retransmit or notify the user.

- Configure error detection settings in the communication protocol (if applicable) to ensure that corrupted data packets are detected.
- For **wireless or long-distance Ethernet**: Consider implementing additional error correction techniques, as wireless networks may suffer from interference.

- Check for communication errors or timeouts in the PLC programming environment. Reconfigure or replace cables if errors persist.

6. Redundancy and Backup Connections

Redundancy refers to having alternative communication paths to ensure that, if one connection fails, the system can switch to another. This is especially important for critical systems in automation.

- For **critical systems**, consider dual Ethernet connections or a combination of Ethernet and wireless communication for redundancy.
- If programming remotely, ensure that a backup communication method (such as a VPN over mobile data) is available in case of primary connection failure.

- Test the redundancy system by intentionally disabling the primary connection and confirming that communication switches seamlessly to the backup.

7. Real-time Monitoring of Connection Status

- Constant monitoring of the connection's health ensures that any degradation in performance is quickly identified.
- Many PLC programming environments have tools for real-time connection monitoring, which display metrics such as communication errors, latency, and bandwidth usage.
- Use diagnostics tools provided in the programming software to monitor the PLC's connection status.

- Enable real-time diagnostics during programming and testing phases. If any anomalies are detected, take corrective actions before full deployment.

Practical Activity 3.1.2: Flashing PLC Program

Task:

1: Referring to the previous activity (3.1.1) and key reading key readings (3.1.2) you are requested to perform the given task. The task should be done individually.

You are an automation technician at a beverage packaging plant. The production line is currently controlled by an outdated PLC program that doesn't account for new packaging requirements. Your task is to flash a new program into the PLC that includes updated logic for handling new bottle sizes and improving overall efficiency.

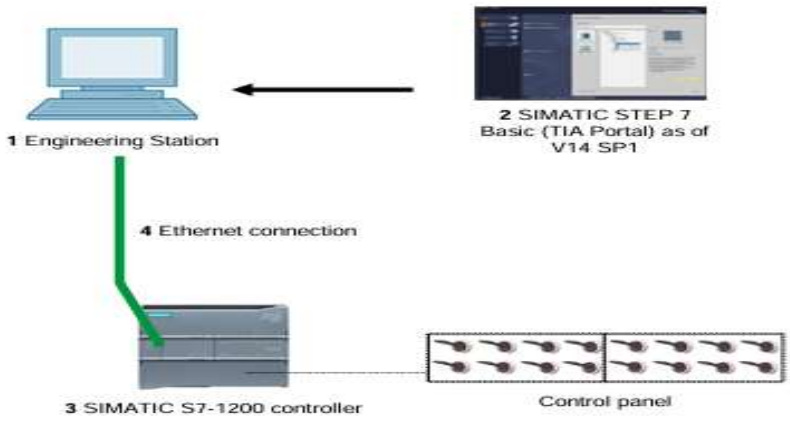
2: Present your work to the class/Trainer or colleagues

3: Ask your trainer for clarification if any.

Key readings 3.2.2: Flashing PLC Program

✓ **Flashing the Program File to a PLC**

Flashing a program file to a PLC involves transferring the control logic from the programming device (PC/laptop) to the PLC's memory.



This process is crucial for implementing and running the programmed

instructions within the PLC system.

The step-by-step process to flash a program to a PLC:

1. **Verify Program Integrity**

- **Program Compilation:**
  - ❖ Ensure the program is free of syntax and logical errors. Most PLC programming software will compile or verify the program before allowing it to be downloaded.
- **Test in Simulation (if possible)**
  - ❖ Use a simulation tool (if available) to test the program logic virtually, reducing the chance of errors during actual execution in the PLC.

2. **Connect the Programming Device to the PLC**

- Ensure that your PC/laptop is connected to the PLC using the appropriate connection method:

  - ❖ **USB** for direct connections.
  - ❖ **Ethernet** for network-based connections.
  - ❖ **RS-232/RS-485** for serial communication.
  - ❖ **Wireless** (if available).

- Confirm that the connection is active and stable. Most PLC programming software provides a status indicator showing if the PLC is connected and online.

3. **Put the PLC in the Correct Mode**

- **Stop the PLC (if necessary):**
  - ❖ Some PLCs require being in **STOP** mode before a new program can be flashed.
  - ❖ In the programming software, change the PLC's mode to **STOP** to prevent it from running any active programs while flashing the new file.
  - ❖ Some advanced systems may allow program download while the system is running (called "online download"), but it's safer to stop the PLC for significant changes.

4. **Select the Flash/Download Option**

- **In the PLC Programming Software:**
  - ❖ Open the PLC programming environment (e.g., Siemens STEP 7, Allen Bradley RSLogix, Mitsubishi GX Works).
  - ❖ Navigate to the option to download or flash the program. This is typically found in the "Communications" or "Transfer" section of the software.

- **Select the PLC/Target Device:**

❖ If multiple PLCs are connected or available on the network, ensure you select the correct target device for flashing the program.

- **Choose Program Components to Flash:**

❖ Select the program elements to flash (e.g., main program, subroutines, data blocks). In most cases, you will flash the entire program.

5. **Start the Flashing Process**

- **Initiate Flashing/Download:**

❖ Click the **Download** or **Flash** button in the programming software to begin transferring the program to the PLC.

- **Monitor the Progress:**

❖ The software will usually show the download status (e.g., percentage complete) during the flashing process. Ensure that the transfer is progressing without errors.

- **Resolve Errors (if any):**

❖ If any errors occur during the flashing process, the software will generate a message. Troubleshoot these issues by checking the connection, ensuring enough memory on the PLC, and confirming the compatibility of the program with the PLC.

6. **Verify the Flashing Process**

- **Check for Confirmation:**

❖ After the download is complete, most PLC programming software will give a message confirming that the program was successfully flashed.

- **Verify Program on the PLC:**

❖ Use the PLC software to check that the flashed program is now present on the PLC. You can usually view the program structure directly from the PLC to verify that it matches the intended logic.

7. **Put the PLC in RUN Mode**

- **Switch the PLC to RUN Mode:**

❖ After the program has been flashed, switch the PLC to **RUN** mode in the programming software or via the PLC's physical controls (if available).

- **Monitor the System:**

❖ Observe the PLC to ensure it is executing the new program correctly. Check any connected devices, outputs, or processes to ensure proper operation.

8. **Perform Testing and Debugging**

- **Test the Program:**

❖ Once the PLC is in RUN mode, test the system to ensure that all inputs,

outputs, and control logic are working as expected.

- **Online Monitoring and Debugging:**

❖ Use the online monitoring features in the PLC software to track the program execution in real-time. This helps identify any logical or operational errors during live execution.

- **Make Adjustments (if necessary):**

❖ If any issues are detected during testing, you can make minor changes and reflash the program. Some PLCs allow you to download only specific parts of the program for quick updates.

9. **Save the Program Configuration**

- **Backup the Program:**

❖ Once the program is running correctly, ensure that you save the final version of the program both locally on your computer and on external storage. This helps prevent data loss and enables future updates.

- **Store in PLC Memory (if applicable):**

❖ Some PLCs may have the option to save the program permanently in non-volatile memory (e.g., EEPROM). This ensures the program is retained even during power cycles.

Points to Remember

- Program file preparation is a critical stage in working with a PLC (Programmable Logic Controller) system. It involves developing, organizing, and testing the program before it is transferred to the PLC for controlling an industrial process or machinery.
- The steps in plc program file preparation
  - ✓ Understand the Control Requirements
  - ✓ Selecting the Right Programming Software
  - ✓ Writing the Control Logic
  - ✓ Organizing the Program Structure.
  - ✓ Configuring I/O Mapping
  - ✓ Error Checking and Simulation
  - ✓ Communication Settings
  - ✓ Testing the Program in Real-Time
  - ✓ Finalizing and Saving the Program

- Method of Connection is concerned with the physical and network-based ways to connect the programming device to the PLC, like whether you use USB, Ethernet, or Wi-Fi.

Depth of Connection refers to the level of access you have once connected, ranging from simply transferring a program to being able to fully control and monitor the PLC's operations

- **The steps process to flash a program to a PLC:**
    1. Verify Program Integrity
    2. Connect the Programming Device to the PLC
    3. Put the PLC in the Correct Mode
    4. Select the Flash/Download Option
    5. Start the Flashing Process
    6. Verify the Flashing Process
    7. Put the PLC in RUN Mode
    8. Perform Testing and Debugging
    9. Save the Program Configuration

Application of learning 3.1.

As an automation technician at a building management system company, you are tasked with updating the PLC program that controls the HVAC system for a large commercial building. The existing system has been underperforming, leading to increased energy usage and inconsistent temperature control. To address these issues, you begin by conducting an initial assessment, reviewing the current system documentation, and analysing historical performance data to identify patterns and inefficiencies.

Indicative content 3.2: Testing Automation System

Duration: 6 hrs

Theoretical Activity 3.2.1: Description of automation system testing techniques

Tasks:

1: You are invited to answer the following questions related to automation system testing techniques:

i. What are automation system testing techniques?
ii. what are the Advantages and disadvantages automation system testing techniques?
iii. Explain STLC (Software Testing Life Cycle).

2: Present your findings to the whole class and Trainer

3: Ask your trainer for any clarification if any.

4: You are directed to read the key readings 3.2.1. for more clarifications on automation system testing techniques.

Key readings 3.2.1.: Description of automation system testing techniques

Automation system testing, also known as automated testing, is a software testing technique that uses tools and scripts to automatically perform tests on a system. The goal of automation testing is to ensure that the system is working as expected and meets requirements before it is released into production

➢ **Types of automation system testing techniques**

Testing techniques for **Programmable Logic Controllers (PLCs)** are essential to ensure that control systems operate reliably and safely.

Common testing techniques specifically applicable to PLC systems:

- **Functional testing**
- **Description**: Verifies that the PLC program meets all specified functional requirements.
- **Technique**: Use a set of predefined test cases to evaluate the operation of each function (e.g., starting/stopping motors, reading sensor inputs).
- **Example**: Check if the PLC can correctly control the speed of a motor based on input signals from a speed sensor.

- **Unit Testing**
- **Description**: Tests individual components or functions of the PLC program in isolation.
- **Technique**: Focus on specific logic blocks or routines to ensure they operate correctly.
- **Example**: Testing a specific rung of ladder logic that controls a pump to ensure it activates and deactivates correctly.

3. Integration Testing

- **Description**: Tests the interaction between different PLC modules or between the PLC and external devices (e.g., sensors, actuators).
- **Technique**: Verify that all components work together as intended.
- **Example**: Test the system where multiple sensors provide input to the PLC, and the outputs control various actuators.

4. **Simulation Testing**

- **Description**: Uses simulation software to emulate the PLC program and its environment without the physical hardware.
- **Technique**: Simulate various operating conditions to observe how the PLC responds.
- **Example**: Simulating different input conditions to check if the PLC logic correctly processes them and produces the expected outputs.

5. End-to-End Testing

- **Description**: Validates the entire control process from start to finish, ensuring that all components work together seamlessly.
- **Technique**: Conduct tests that involve all parts of the control system, including PLC logic, sensors, actuators, and any connected systems.
- **Example**: Testing a complete assembly line operation where the PLC controls various machines and sensors throughout the process.

6. Performance Testing

- **Description**: Evaluates the performance of the PLC under different load conditions.
- **Technique**: Measure response times, processing speed, and system behavior under stress.
- **Example**: Assess how the PLC handles a high volume of input signals simultaneously.

7. Regression Testing

- **Description**: Ensures that new changes or updates to the PLC program do not introduce new defects or cause previously functioning parts to fail.
- **Technique**: Re-run previously executed test cases after modifications to confirm that existing functionality remains intact.
- **Example**: After modifying the control logic for one machine, test all related functions to ensure they still work correctly.

8. Boundary Value Testing

- **Description**: Tests the behavior of the PLC at the edge of input ranges or operational limits.
- **Technique**: Focus on the minimum and maximum values of inputs to ensure the PLC handles these correctly.
- **Example**: If a temperature sensor provides readings between 0°C and 100°C, test the PLC's response at 0°C, 100°C, and values just outside this range.

9. Stress Testing

- **Description**: Determines the robustness of the PLC by pushing it beyond normal operational limits.
- **Technique**: Apply extreme input conditions or rapid changes to observe how the PLC handles stress.
- **Example**: Rapidly switching inputs on and off to see if the PLC can manage without crashing or producing erroneous outputs.

10. Safety Testing

- **Description**: Verifies that safety features and interlocks in the PLC program work correctly.
- **Technique**: Check emergency stop functions, safety interlocks, and error handling mechanisms.
- **Example**: Triggering an emergency stop to ensure the PLC stops all outputs safely and as designed.

➢ **Advantages and disadvantages**

✓ **Advantages:**

1. Improved Reliability:

Thorough testing ensures that the PLC system operates as intended, reducing the likelihood of failures in real-world applications.

Benefit: Increases trust in the automation system and minimizes downtime.

2. Early Bug Detection:

Techniques such as unit testing and regression testing help identify defects early in the development cycle.

Benefit: Reduces the cost and time associated with fixing bugs later in the process.

3. Enhanced Safety:

Safety testing verifies that safety mechanisms (e.g., emergency stops, interlocks) function correctly.

Benefit: Protects operators and machinery, ensuring compliance with safety regulations.

4. Cost Efficiency:

Automated testing can be executed faster than manual testing, saving labor costs over time.

Benefit: Frees up resources for other critical tasks and can lead to quicker project completions.

5. Comprehensive Coverage:

Testing techniques can assess a wide range of scenarios, ensuring that the PLC responds correctly to various inputs and conditions.

Benefit: Increases confidence in the system's robustness and effectiveness.

6. Documentation and Audit Trail:

Testing creates records that can be referenced for compliance and quality assurance.

Benefit: Provides necessary documentation for audits and regulatory compliance.

✓ **Disadvantages:**

1. Initial Investment Costs:

Setting up a testing environment and acquiring necessary tools can involve significant upfront costs.

Drawback: This can be a barrier for small organizations with limited budgets.

2. Time-Consuming Process:

Comprehensive testing can require substantial time and resources, especially for large and complex systems.

Drawback: This may delay project timelines and increase time to market.

3. Maintenance Effort:

Test cases need to be updated as the PLC program changes, requiring ongoing maintenance.

Drawback: This can lead to additional workload and resource allocation for the testing team.

4. Complexity of Testing:

Some techniques, particularly white box testing, require detailed knowledge of the code and system architecture.

Drawback: This complexity can limit who in the team can perform effective testing.

5. Incomplete Scenario Coverage:

Some testing methods may not capture every possible scenario or edge case.

Drawback: There is still a risk of missing potential issues that may arise in live operations.

6. Dependence on Tools and Technologies:

Effective testing often relies on specific software tools for simulation and analysis.

- **STLC (Software Testing Life Cycle)**

The Software Testing Life Cycle (STLC) is a structured process that outlines the various stages of software testing. It encompasses all testing-related activities, from planning and designing to execution and closure, ensuring a systematic approach to validating software quality.

The STLC (Software Testing Life Cycle) phases:

1. Requirement Analysis

Understand and analyze the requirements for testing.

- **Activities**:
  - Identify testable requirements.
  - Review and clarify requirements with stakeholders.
  - Determine the types of testing needed (e.g., functional, performance).

2. Test Planning

Develop a comprehensive test plan to guide testing efforts.

- **Activities**:
  - Define the testing scope and objectives.
  - Identify resources (team, tools) and estimate time and costs.

–Establish testing strategies and deliverables.

3. Test Case Design

Create detailed test cases and scripts based on requirements.

- **Activities**:

–Write test cases covering various scenarios, including positive and negative cases.

–Create test data and determine test environments.

–Review and finalize test cases for execution.

4. Test Environment Setup

Prepare the testing environment for executing test cases.

- **Activities**:

–Set up hardware, software, and network configurations.

–Install the application to be tested.

–Ensure that necessary tools and resources are available.

5. Test Execution

Execute the test cases and document results.

- **Activities**:

–Run test cases as per the test plan.

–Record test results and any defects encountered.

–Conduct retesting and regression testing as needed.

6. Test Closure

Conclude testing activities and evaluate the process.

- **Activities**:

–Analyze test results and metrics to assess testing effectiveness.

–Review and document lessons learned, best practices, and areas for improvement.

–Prepare a final test summary report for stakeholders.

Practical Activity 3.2.2: Applying automation system testing techniques

Task:

1: Referring to the previous activity (3.2.1) and key reading (3.2.2) you are requested to perform the given task. The task should be done individually

As an automation engineer at a manufacturing plant, you are tasked with upgrading the PLC system that controls the conveyor system used for transporting materials between different production stages. The current system has been experiencing issues, such as inconsistent

conveyor speeds and frequent jams, leading to production delays. To address these problems, you have developed a new PLC program designed to improve the control logic for conveyor speed regulation, material detection, and jam management.

Task: Apply automation testing techniques to ensure the updated PLC program functions properly, effectively controlling the conveyor system's speed, detecting jams, and maintaining smooth material flow before the system is deployed in the live production environment.

2: Present your work to the class or colleagues

3: Ask your trainer for clarification if any.

4: Read the Key readings 3.2.2 in trainee's manuals

Key readings 3.2.2: Applying automation system testing techniques

➢ **Unit Testing**

Definition:

- Unit testing involves testing individual components or modules of the PLC program to ensure they function as intended. This step is crucial for identifying issues at a granular level before integrating the components into the larger system.

Objectives:

- Validate the correctness of each program module.
- Identify and fix bugs early in the development process.
- Ensure that each unit performs its designated function without errors.

Process:

1. **Isolate the Unit:**
   - Identify the specific function or block of code to be tested. This could be a specific ladder logic rung, function block, or subroutine.
2. **Simulate Inputs:**
   - Simulate inputs to the unit being tested to see how it responds. This may involve using the PLC programming software's simulation features to mimic real-world conditions.
3. **Check Outputs:**
   - Verify that the outputs from the unit match expected results based on the simulated inputs. This can involve checking status indicators, output signals, or logging data.

4. **Document Results:**
   - Record the outcomes of the tests, noting any discrepancies between expected and actual results. This documentation can be crucial for debugging and future reference.
5. **Iterate:**
   - If any issues are identified, revise the unit code and re-test until the unit performs correctly.

Tools:

- Most PLC programming environments (like Siemens TIA Portal, Allen Bradley RSLogix, etc.) provide built-in simulation tools for unit testing.

Advantages:

- Early identification of errors leads to reduced troubleshooting time.
- Simplifies debugging by allowing focus on individual components.

Disadvantages:

- May require additional time upfront to isolate and test units, which can delay the overall development timeline.

➢ **System Testing**

Definition:

- System testing evaluates the complete and integrated PLC program as a whole, ensuring that all components work together as intended in the actual operating environment.

Objectives:

- Validate the entire system's functionality.
- Ensure that all units interact correctly and that the PLC meets the specified requirements.
- Identify issues that may not have been evident during unit testing.

Process:

1. **Prepare the Testing Environment:**
   - Set up the physical environment or a controlled simulation that mirrors

actual operating conditions, including all connected devices and sensors.

2. **Execute Functional Tests:**
   - Run the PLC program in its entirety and perform functional tests based on the system requirements. This includes:
     - **Input Testing:** Trigger all possible input conditions and validate that the system responds appropriately.
     - **Output Testing:** Verify that the correct outputs are activated based on given inputs.
     - **Sequence Testing:** Ensure that processes occur in the correct order and that timing requirements are met.
3. **Stress Testing:**
   - Test the system under extreme conditions to ensure it can handle peak loads without failure. This may include rapidly cycling inputs or simulating fault conditions.
4. **Monitor Performance:**
   - Observe how the system behaves in real-time, checking for performance metrics such as response times, error rates, and resource utilization.
5. **Document and Analyze Results:**
   - Record all findings during system testing, including any failures or unexpected behaviors. Analyze the data to identify root causes and areas for improvement.
6. **Iterate as Necessary:**

- Based on the test results, make any necessary changes to the program, re-test, and validate that the issues have been resolved.

Tools:

- In addition to the PLC programming software, external testing and monitoring tools (such as SCADA systems or data loggers) can enhance system testing by providing real-time data and analytics.

Advantages:

- Comprehensive testing helps ensure the reliability and stability of the complete system.
- Validates that all components interact correctly in a real-world scenario.

Disadvantages:

- May require more time and resources to set up and conduct compared to unit testing.
- Difficulties in replicating all potential operating conditions can lead to undetected issues.

Points to Remember

- Automation system testing, also known as automated testing, is a software testing technique that uses tools and scripts to automatically perform tests on a system.
- Types of automation system testing techniques may include: Functional testing, Unit Testing, Integration Testing, Simulation Testing,…..
- Automation system testing has many advantages, including:
  - ✓ Improved Reliability
  - ✓ Early Bug Detection
  - ✓ Enhanced Safety
  - ✓ Cost Efficiency
  - ✓ Comprehensive Coverage
  - ✓ Documentation and Audit Trail
- However, there are also some disadvantages to automation system testing, including
  - ✓ Initial Investment Costs
  - ✓ Time-Consuming Process
  - ✓ Maintenance Effort
  - ✓ Complexity of Testing
  - ✓ Incomplete Scenario Coverage
  - ✓ Dependence on Tools and Technologies
- The Software Testing Life Cycle (STLC) is a structured process that outlines the various stages of software testing. It encompasses all testing-related activities, from planning and designing to execution and closure, ensuring a systematic approach to validating software quality.
- Unit testing involves testing individual components or modules of the PLC program to ensure they function as intended.
- Steps for applying unit testing includes: Isolate the Unit, Simulate Inputs, Check Outputs, Document Results and Iterate
- System testing evaluates the complete and integrated PLC program as a whole, ensuring that all components work together as intended in the actual operating environment

- Steps for applying system testing includes prepare the testing environment, execute functional tests, stress testing, monitor performance and document and analyse results

Application of learning 3.2.

As an automation technician at a water treatment plant, you are tasked with upgrading the existing manual system that controls various processes such as filtration, chemical dosing, and water level management. The current system relies on manual operations, which has led to inefficiencies, including inconsistent chemical dosing and frequent overflow issues in the water storage tanks. To resolve these challenges, you are hired to develop a new PLC program that will automate and improve the control logic for each stage of the water treatment process

Task:

Apply automation testing techniques to verify that the updated PLC program works as intended, ensuring the correct chemical dosing, efficient water filtration, and accurate water level management before deploying it to the plant.

Indicative content 3.3: Documenting Automation System

Duration: 4hrs

Theoretical Activity 3.2.1: Description of technical documentation

Tasks:

1: You are invited to answer the following questions related to Technical documentation

i. What is Hardware configuration and Software configuration of automation system?
ii. What are the key aspects of hardware and software configuration?
iii. Discuss Control logic of automation system.
iv. Explain the Network architecture of automation system.
v. What are the key components of network architecture?
vi. What is the automation system user manual?

2: Present your findings to the whole class and Trainer

3: you are directed to read the key readings 3.1.1. for more clarifications

Key readings 3.3.1: Description of technical documentation

Hardware configuration refers to the arrangement and setup of physical components in a PLC system, ensuring that all devices communicate effectively and operate correctly. Proper configuration is essential for the successful implementation of any automation project.

KEY ASPECTS OF HARDWARE CONFIGURATION

1. PLC Selection

- **Choose the Right PLC:**
  - Select a PLC that meets the requirements of the application, considering factors like the number of I/O points, processing speed, memory capacity, and communication capabilities.
- **Types of PLCs:**
  - **Compact PLCs:** Integrated I/O and CPU in one unit, suitable for small applications.

- **Modular PLCs:** Separate I/O and CPU modules, allowing for greater scalability and flexibility.
- **Safety PLCs:** Designed for applications requiring higher safety standards.

2. I/O Modules Configuration

- **Digital and Analog I/O:**
  - Determine the number and types of inputs and outputs needed (digital vs. analog).
  - Digital I/O: For discrete signals (on/off, sensors, switches).
  - Analog I/O: For continuous signals (temperature, pressure, flow).
- **Select Modules:**
  - Choose appropriate I/O modules that match the requirements (e.g., thermocouple input modules for temperature sensors).
- **Wiring I/O Devices:**
  - Properly wire sensors, actuators, and other devices to the corresponding I/O terminals on the PLC.
  - Ensure adherence to wiring diagrams and safety standards.

3. Communication Ports and Protocols

- **Define Communication Needs:**
  - Identify the necessary communication protocols (e.g., Ethernet/IP, Modbus, Profibus) based on system requirements.
- **Configure Communication Ports:**
  - Set up Ethernet or serial communication ports for connecting to HMI (Human-Machine Interface), SCADA systems, or other PLCs.
- **Network Configuration:**
  - Ensure proper IP addressing and subnetting for Ethernet connections and configure any necessary routing settings.

4. Power Supply Configuration

- **Determine Power Requirements:**
  - Calculate the power requirements for the PLC and all connected devices to ensure adequate power supply.
- **Choose Power Supply:**
  - Select a power supply compatible with the PLC voltage specifications (e.g., 24V DC for most PLCs).
- **Wiring:**

- Properly wire the power supply to the PLC and all associated components, ensuring that polarity is correct.

5. Peripheral Devices and Accessories

- **HMI and Operator Interfaces:**
  - Integrate HMI devices for operator interaction with the PLC system. Configure the HMI to communicate with the PLC using the defined protocols.
- **Remote I/O:**
  - If applicable, configure remote I/O stations to expand I/O capacity while maintaining communication with the main PLC.
- **Sensors and Actuators:**
  - Connect and configure sensors (e.g., photoelectric sensors, proximity sensors) and actuators (e.g., relays, solenoids) according to the application requirements.

6. Environmental Considerations

- **Select Suitable Enclosures:**
  - Choose enclosures that protect the PLC and associated hardware from environmental factors (dust, moisture, temperature).
- **Cooling and Ventilation:**
  - Ensure adequate cooling and ventilation in the installation area to prevent overheating.
- **Grounding:**
  - Properly ground the PLC and all components to prevent electrical interference and ensure safety.

7. Documentation and Labelling

- **Create Configuration Documentation:**
  - Document the hardware configuration, including wiring diagrams, IP address schemes, and I/O assignments for future reference.
- **Label Components:**
  - Label all terminals, cables, and devices to facilitate troubleshooting and maintenance.

8. Testing and Validation

- **Initial Testing:**

- Once the hardware is configured, perform initial testing to ensure that all components communicate correctly and function as expected.

- **Troubleshooting:**
  - If issues arise, use diagnostic tools and techniques to troubleshoot hardware problems, checking connections and configurations.

Software configuration involves setting up the programming environment, defining parameters, and developing control logic for a Programmable Logic Controller (PLC). This process is essential for ensuring that the PLC operates effectively within an automation system

Below are the key steps involved in software configuration, using TIA Portal as an example:

1. Create a New Project

- **Open TIA Portal:**
- Launch the TIA Portal software on your PC.
- **Create a New Project:**
- Click on "Create new project" and give your project a name and location. Click **Create** to proceed.

2. Configure the Hardware

- **Add a Device:**
  - In the Project View, right-click on "Devices & Networks" and select **Add new device**.
  - Choose the appropriate PLC model (e.g., S7-1200, S7-1500) based on your application needs.
- **Define the Configuration:**
  - After adding the device, you can configure its properties, such as CPU type, firmware version, and additional modules (I/O modules, communication processors).
- **Set Up I/O Modules:**
  - Drag and drop the necessary I/O modules from the hardware catalogue into your device configuration. Ensure that the correct modules match your physical setup.

3. Set Up Network Configuration

- **Define the Network:**

- Create a network layout by right-clicking on the PLC and selecting **Add new subnet**. Define the communication medium (e.g., PROFINET, Ethernet).

- **Configure IP Address:**
  - Assign an IP address to the PLC and any other devices in the network to enable communication.
- **Configure Remote I/O:**
  - If using remote I/O devices, add them to the network and configure their parameters similarly.

4. Programming the PLC

- **Open the PLC Programming Environment:**
  - Click on your PLC in the Project View to open the programming area.
- **Select the Programming Language:**
  - Choose the appropriate programming language (e.g., Ladder Logic, Function Block Diagram, Structured Text) based on your application requirements.
- **Create Program Blocks:**
  - Define main program blocks (OB1) and additional function blocks (FBs) or subroutines (FCs) as needed.
- **Add Logic:**
  - Drag and drop elements (contacts, coils, timers, counters) from the library to create your control logic.

5. Set Up HMI (if applicable)

- **Add an HMI Device:**
  - If your system includes an HMI, right-click on "Devices & Networks" and select **Add new device** to include an HMI panel.
- **Configure HMI Screens:**
  - Design the HMI screens by adding graphical elements, buttons, indicators, and data displays that correspond to your PLC logic.
- **Link HMI to PLC:**
  - Establish communication between the HMI and PLC by configuring tags that reference PLC variables.

6. Parameter Configuration

- **Set Device Parameters:**
  - Configure parameters for the PLC and connected devices, including:

- Communication settings (baud rate, protocol).
- I/O settings (input/output types, scaling for analog signals).
- Safety settings (if applicable).

➢ **Define Global Data Blocks:**

❖ Create global data blocks to store shared data and configurations used throughout the program.

7. Testing and Simulation

➢ **Simulate the Program:**

❖ TIA Portal offers simulation capabilities. Use the **PLCSIM** tool to test the program without needing physical hardware.

❖ Run the simulation and verify the logic, ensuring it meets operational requirements.

➢ **Debugging:**

❖ Utilize debugging tools such as breakpoints and watch tables to analyse program behavior and correct errors.

8. Download the Program to the PLC

➢ **Connect to the PLC:**

❖ Ensure that your PC is connected to the PLC (via USB, Ethernet, etc.).

➢ **Compile the Project:**

❖ Before downloading, compile the project to check for errors or warnings.

➢ **Download the Program:**

❖ Right-click on the PLC in the Project View and select **Download to device**. Follow the prompts to complete the download process.

➢ **Set PLC to RUN Mode:**

❖ Once the program is downloaded, set the PLC to RUN mode to start executing the control logic.

9. Monitoring and Maintenance

➢ **Online Monitoring:**

❖ Use the online monitoring feature to view real-time data and performance metrics.

➢ **Adjustments and Updates:**

❖ Make necessary adjustments to the program or configuration based on operational feedback and performance evaluations.

10. Documentation

- **Document the Configuration:**
  - Keep comprehensive documentation of your configuration settings, program logic, and any modifications made throughout the project.
- **Create User Manuals:**
  - Prepare user manuals that detail how to operate and maintain the automation system, including troubleshooting guides

Control logic

Control logic refers to the set of instructions and rules that dictate how a Programmable Logic Controller (PLC) responds to inputs from various sensors and devices, ultimately determining how outputs (like motors, lights, and other actuators) are activated. Effective control logic is crucial for the proper functioning of automation systems. Below are key concepts and components involved in control logic programming for PLCs.

1. Understanding Inputs and Outputs

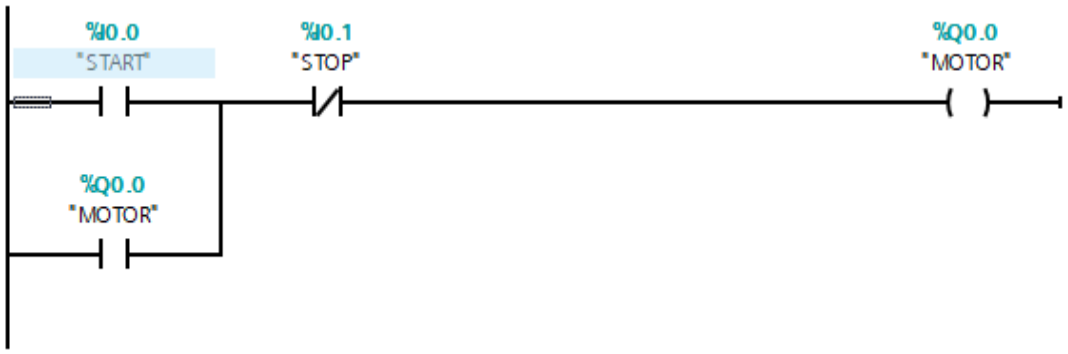
- **Inputs:**
  - Inputs are signals received from devices such as sensors, switches, and other field devices. They can be digital (on/off) or analog (varying values).
- **Outputs:**
  - Outputs are signals sent to actuators, relays, or other devices based on the control logic. Outputs can also be digital (e.g., turning a light on/off) or analog (e.g., controlling the speed of a motor).

2. Types of Control Logic

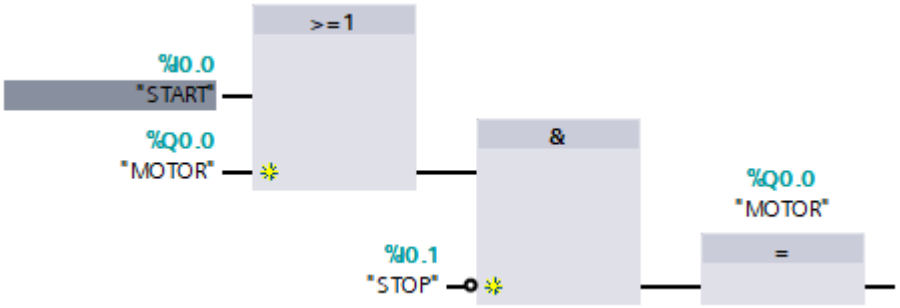
- **Discrete Control Logic:**
  - Used for controlling on/off processes. Examples include controlling lights, motors, and valves.
- **Sequential Control Logic:**
  - Governs processes that follow a specific sequence of operations, often used in manufacturing or assembly lines.
- **Continuous Control Logic:**
  - Regulates processes requiring continuous adjustment, such as temperature control in heating systems or speed control in motors.

Programming Languages for Control Logic

- **Ladder Logic (LD):**



  - A graphical programming language resembling electrical relay logic diagrams. It uses rungs and rails to represent control logic visually.
- **Function Block Diagram (FBD):**

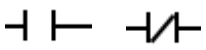


  - A graphical representation that uses blocks to represent functions. It is particularly useful for complex logic involving multiple operations.
- **Structured Text (ST):**
  - A high-level programming language similar to traditional programming languages, ideal for complex calculations and data handling.
- **Instruction List (IL):**
  - A low-level language that uses mnemonic codes for operations. It is less common but still used in some systems.

- **Basic Control Logic Elements**

- **Contacts:**

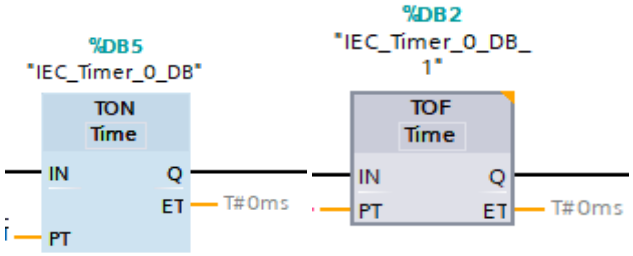
❖ Represent inputs and conditions. They can be normally open (NO) or normally closed (NC).



➢ **Coils:**

❖ Represent outputs that can be activated or deactivated based on conditions. Coils can also control outputs or trigger events.

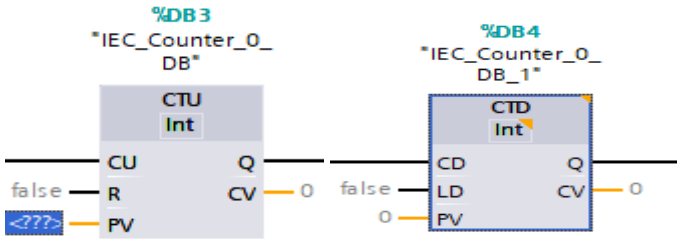
➢ **Timers:**



❖ Allow for time-based control, enabling operations to be delayed or executed after a specific duration.

➢ **Counters:**

❖ Count occurrences of events (like sensor activations) and trigger outputs based on count values.



➢ **Comparison Instructions:**

❖ Compare values (e.g., greater than, less than) and determine control actions based on those comparisons.

➢ **Developing Control Logic**

➢ **Define Objectives:**

❖ Clearly outline what the control system needs to accomplish (e.g., starting a motor, activating an alarm).

- **Create Flowcharts or Diagrams:**
  - Design flowcharts or state diagrams to visualize the process flow and decision points in the control logic.
- **Implement Logic:**
  - Using the selected programming language, implement the control logic based on the defined objectives and flow diagrams.
- **Test and Validate:**
  - Test the logic in a simulation environment or on the actual PLC to ensure it behaves as expected. Debug any issues that arise.

- **Network Architecture**

Network architecture in automation systems refers to the structured layout of hardware, software, communication protocols, and data flows that enable devices to communicate and operate together effectively. It encompasses the design and organization of network elements that facilitate communication between PLCs, sensors, actuators, HMIs, and other devices in an industrial environment. Below are the key components and considerations for establishing a robust network architecture.

1. Key Components of Network Architecture

- **Controllers:**
  - PLCs, DCS (Distributed Control Systems), and other controllers manage processes and communicate with field devices.
- **Field Devices:**
  - Sensors, actuators, and other I/O devices that gather data from the environment or perform actions based on control logic.
- **Communication Devices:**
  - Switches, routers, gateways, and hubs that facilitate data transmission between devices in the network.
- **Human-Machine Interfaces (HMIs):**
  - Devices that provide a user interface for operators to monitor and control processes.
- **SCADA Systems:**
  - Supervisory Control and Data Acquisition systems that aggregate data from multiple PLCs and devices for centralized monitoring and control.

2. Network Topologies

Network topology refers to the arrangement of different elements (links, nodes, etc.) in a computer network. Common topologies used in automation systems include:

- **Star Topology:**
  - All devices are connected to a central hub or switch. This topology is easy to manage and troubleshoot but may suffer from single points of failure.
- **Ring Topology:**
  - Devices are connected in a closed loop, with data traveling in one direction. This can provide redundancy but is more complex to manage.
- **Bus Topology:**
    - All devices share a single communication line (bus). While cost-effective, it can lead to performance issues if many devices are connected.
- **Mesh Topology:**
  - Devices are interconnected, providing multiple paths for data transmission. This increases redundancy but can be costly and complex.

3. Communication Protocols

Communication protocols define the rules and conventions for data exchange between devices. Common protocols used in industrial networks include:

- **Ethernet/IP:**
  - An open protocol that uses standard Ethernet technology for communication between industrial devices, widely used in PLC systems.
- **PROFIBUS:**
  - A standard for fieldbus communication in automation technology, allowing for fast data transmission and interoperability among devices.
- **Modbus:**
  - A serial communication protocol used for transmitting data over serial lines. It is simple and widely used for connecting PLCs to remote devices.
- **CAN Bus:**
  - A robust vehicle bus standard designed to facilitate communication among microcontrollers and devices without a host computer.
- **OPC UA (Open Platform Communications Unified Architecture):**
  - A machine-to-machine communication protocol for industrial automation, ensuring secure and reliable data exchange across different platforms.

4. Network Security

As automation systems become more interconnected, security becomes paramount. Key security measures include:

- **Firewalls:**
  - Implement firewalls to monitor and control incoming and outgoing network traffic based on predetermined security rules.
- **Virtual Private Networks (VPNs):**
  - Use VPNs to create secure connections over public networks, protecting sensitive data.
- **Access Control:**
  - Implement strict access controls to limit who can access network devices and data.
- **Regular Updates and Patching:**
  - Keep software and firmware up to date to protect against vulnerabilities.

5. Redundancy and Reliability

To ensure uninterrupted operations, redundancy should be incorporated into the network design:

- **Redundant Paths:**
  - Create multiple communication paths between devices to prevent single points of failure.
- **Backup Power Supplies:**
  - Use uninterruptible power supplies (UPS) to maintain network functionality during power outages.
- **Failover Mechanisms:**
  - Implement automatic failover mechanisms to switch to backup systems in case of a failure.

6. Monitoring and Maintenance

Regular monitoring and maintenance are essential for optimal network performance:

- **Network Monitoring Tools:**
  - Use software tools to monitor network performance, traffic, and device health.
- **Performance Analysis:**

- Analyze network performance metrics to identify bottlenecks or issues.

➢ **Documentation:**

- Maintain detailed documentation of network architecture, device configurations, and communication protocols to facilitate troubleshooting and future upgrades

✓ **Prepare system's user manual**

This manual provides instructions for operating and maintaining the automation system. It is intended for system users and technicians responsible for the system's upkeep. Please read through the manual before operating the system

Template of Automation System User Manual

Project Name: [Insert Project Name]
System Version: [Insert Version Number]
Date: [Insert Date]
Prepared by: [Insert Author's Name]

Table of Contents

Practical Activity 3.3.2: Performing hardware and software configuration in automation system

Task:

1: Referring to the previous activity (3.3.1) and key reading key readings (3.3.2) you are requested to perform the given task. The task should be done individually

As an automation technician in a logistics company, you are tasked with configuring a new automated warehouse management system (WMS) designed to streamline inventory management and improve order fulfilment processes. This system consists of automated guided vehicles (AGVs), RFID scanners, conveyor belts, and a centralized control system that oversees operations. The goal is to ensure that all hardware components are correctly installed and configured, and that the software is programmed to manage the flow of goods efficiently within the warehouse, from receiving shipments to storing and picking items for dispatch.

2: Follow instructions and procedures provided in that key readings and perform the given task

3: Present your final products to trainer /colleagues

4: Ask for clarification if any

Key readings 3.3.2: Performing hardware and software configuration in automation system

1. Hardware Configuration

1.1 System Inspection

1. **Verify the Components:** Ensure all components (controllers, sensors, actuators, communication devices) are in place and undamaged.
2. **Check the Wiring Diagram:** Reference the circuit diagram provided for the system to confirm proper connections.

1.2 Mounting the Equipment

1. **Mount the Controller (e.g., PLC or DCS):**
   - ❖ Secure the controller in the designated control panel or enclosure.
   - ❖ Ensure it is properly ventilated to avoid overheating.
2. **Install Sensors and Actuators:**

- Install sensors (e.g., temperature, pressure, or motion sensors) in the correct positions.
- Install actuators (e.g., motors, solenoids, valves) based on system requirements.
- Ensure correct alignment and orientation for effective operation.

1.3 Electrical Connections

1. **Power Supply:**
   - Connect the power supply to the controller and all peripheral devices.
   - Use the correct voltage and current ratings to avoid damaging the components (e.g., 24V DC or 230V AC).
2. **Input/Output (I/O) Wiring:**
   - Wire the sensors to the input terminals of the controller.
   - Wire the actuators to the output terminals.
   - Ensure proper grounding and shielding of cables to avoid signal interference.
3. **Communication Cables:**
   - For systems using network protocols like Modbus, Ethernet, or Profibus, connect the communication cables (Ethernet or serial) between the controller and remote devices (e.g., HMIs or SCADA systems).
   -

1.4 Powering Up

1. **Initial Power-Up:**
   - Power on the system and check for any signs of improper operation (e.g., indicator lights, status displays).
   - Use a multimeter to confirm voltage levels across different terminals.
2. **Check Sensor and Actuator Operation:**
   - Ensure all sensors and actuators are correctly connected by testing their individual functionality.
   - Look for feedback from sensors on the controller's input terminals.

2. Software Configuration

2.1 Controller (PLC/DCS) Configuration

1. **Install Programming Software:**
   - Use the manufacturer's software (e.g., Siemens TIA Portal, Allen-Bradley RSLogix) for configuring the controller.
   - Install it on a PC connected to the controller via a USB, Ethernet, or serial interface.

2. **Configure Communication Settings:**
   - ❖ Set up communication parameters (e.g., IP addresses for Ethernet-based systems, baud rate for serial communication) to ensure the controller communicates properly with external devices like HMIs, sensors, and actuators.
   - ❖ Test the communication by "pinging" the devices in the network or using diagnostics tools.

2.2 Creating the Logic Program

1. **Input/Output Mapping:**
   - ❖ Map each sensor (input) and actuator (output) to the corresponding addresses in the controller's I/O table.
   - ❖ Example: Assign sensor 1 to Input 0.1 and actuator 1 to Output 0.1 in the controller's program.
2. **Program Development:**
   - ❖ Write the control logic using programming languages such as Ladder Logic (LD), Function Block Diagram (FBD), or Structured Text (ST), depending on the controller.
   - ❖ Create conditions based on sensor inputs that activate or deactivate actuators.
   - ❖ Example: "If temperature > 50°C, turn on the cooling fan."
3. **Timers and Counters:**
   - ❖ Implement timers for delayed actions (e.g., turn off a motor after 5 minutes).
   - ❖ Use counters for tracking repetitive operations or cycles.

2.3 Testing and Simulation

1. **Offline Simulation:**
   - ❖ Before downloading the program to the controller, run a simulation in the software to test if the logic works as expected.
   - ❖ Check the response of inputs and outputs in a virtual environment.
2. **Download Program to Controller:**
   - ❖ Once tested, download the program from the PC to the controller using the software.
   - ❖ Ensure the program is successfully transferred and running.

2.4 Human-Machine Interface (HMI) Configuration

1. **HMI Software Installation:**
   - ❖ Use the HMI's configuration software (e.g., Siemens WinCC, Allen-Bradley FactoryTalk View) to set up the user interface.
2. **Create Screen Layouts:**

- ❖ Design screens that display real-time data from the automation system, including sensor values, actuator status, and alarm indications.
- ❖ Implement control buttons for starting, stopping, or adjusting system parameters.

3. **Link HMI to Controller:**
   - ❖ Set up communication between the HMI and the controller by configuring IP addresses or serial port settings.
   - ❖ Test that the HMI receives data from the controller and can send commands back.

Points to Remember

1. **Hardware configuration** refers to the arrangement and setup of physical components in a PLC system, ensuring that all devices communicate effectively and operate correctly.
   Software configuration involves setting up the programming environment, defining parameters, and developing control logic for a Programmable Logic Controller (PLC). This process is essential for ensuring that the PLC operates effectively within an automation system
2. **Key aspects of hardware configuration**

   - PLC Selection
   - I/O Modules Configuration
   - Communication Ports and Protocols
   - Power Supply Configuration
   - Peripheral Devices and Accessories
   - Environmental Considerations
   - Documentation and Labelling
   - Testing and Validation

- Key steps involved in software configuration, using TIA Portal as an example:

  1. Create a New Project
  2. Configure the Hardware
  3. Set Up Network Configuration
  4. Programming the PLC
  5. Set Up HMI (if applicable)
  6. Parameter Configuration

7. Testing and Simulation
8. Download the Program to the PLC
9. Monitoring and Maintenance
10. Documentation

3. **Control logic** refers to the set of instructions and rules that dictate how a Programmable Logic Controller (PLC) responds to inputs from various sensors and devices, ultimately determining how outputs (like motors, lights, and other actuators) are activated
4. **Network architecture** in automation systems refers to the structured layout of hardware, software, communication protocols, and data flows that enable devices to communicate and operate together effectively.

Key Components of Network Architecture

- Controllers:
- Network Topologies
- Communication Protocols
- Network Security
- Monitoring and Maintenance

5. Creating an automation system user manual involves providing comprehensive instructions and guidelines to help users operate, maintain, and troubleshoot the system effectively.

1. Hardware Configuration

1.1 System Inspection

- Verify the Components**.**
- Check the Wiring Diagram

1.2 Mounting the Equipment

- Mount the Controller (e.g., PLC or DCS)
- Install Sensors and Actuators

1.3 Electrical Connections

- Power Supply
- input/output (I/O) Wiring
- Communication Cables

1.4 Powering Up

- Initial Power-Up
- Check Sensor and Actuator Operation

2.Software Configuration

2.1 Controller (PLC/DCS) Configuration

- Install Programming Software
- Configure Communication Settings

2.2 Creating the Logic Program

- Input/output Mapping
- Program Development
- Timers and Counters

2.3 Testing and Simulation

- **Offline Simulation:**
- **Download Program to Controller:**

2.4 Human-Machine Interface (HMI) Configuration

- **HMI Software Installation:**
- **Create Screen Layouts:**
- **Link HMI to Controller:**

Application of learning 3.3.

As an automation technician at a food processing company, you are tasked with configuring a new automated packaging system designed to streamline the packaging process for various food products. The system consists of multiple components, including conveyor belts, packaging machines, sensors, and a programmable logic controller (PLC) that coordinates the entire operation.

The packaging process must be efficient and adaptable to handle different product sizes and packaging types. You need to ensure that the hardware components are correctly configured, and that the software is programmed to operate seamlessly with the hardware. Your goal is

to minimize downtime during the setup and ensure that the system meets production requirements.

Indicative content 3.4: Maintaining Automation System

Duration: 6hrs

Practical Activity 3.4.1: Applying preventive maintenance, diagnosing and troubleshooting in automation system

Task:

1: Referring to the key reading key readings (3.4.2) you are requested to perform the given task. The task should be done individually

> The beverage manufacturing plant has identified intermittent failures in the automated bottling line, particularly with the bottle detection sensors and the programmable logic controller (PLC) controlling the mixer. As the automation technician, please apply preventive maintenance**,** diagnosing and troubleshooting in automation system by inspecting the bottling line, lubricating moving parts, calibrating sensors, and updating the control system software.

2: Follow instructions and procedures provided in that key readings and perform the given task

3: Present your final products to trainer /colleagues

4: Ask for clarification if any

Key readings 3.4.1.: Applying preventive maintenance, diagnosing and troubleshooting in automation system

Definition of key terms

- **Preventive maintenance** is a proactive approach aimed at keeping automation systems in optimal condition by performing regular checks and maintenance tasks
- **System diagnosing** is the process of identifying and analyzing faults or issues within an automation system
- **System troubleshooting** is the process of diagnosing and resolving issues within an automation system to restore its normal operation

Steps for Preventive Maintenance in an Automation System

Develop a Maintenance Schedule:

- Create a detailed schedule outlining when maintenance tasks will be performed (e.g., daily, weekly, monthly, quarterly, annually).
- Include all equipment and components in the automated system.

Conduct Regular Inspections:

- Perform visual and operational checks of all equipment, including sensors, motors, PLCs, and other automation components.
- Look for signs of wear, corrosion, or damage.

Lubricate Moving Parts:

- Regularly lubricate gears, bearings, and other moving components according to manufacturer specifications to reduce friction and prevent wear.

Calibrate Sensors:

- Check and calibrate sensors to ensure accurate readings and performance.
- Follow manufacturer guidelines for calibration procedures.

Clean Equipment:

- Remove dust, debris, and contaminants from equipment and components to prevent interference with operation.
- Pay special attention to electrical connections and control panels.

Check Electrical Connections:

- Inspect all electrical connections for signs of wear or corrosion.
- Tighten loose connections and replace damaged wiring as needed.

Update Software and Firmware:

- Ensure that all control system software and firmware are up to date.
- Apply patches or updates provided by manufacturers to enhance performance and security.

Test System Performance:

- Conduct routine tests to verify that the entire automation system is functioning correctly.

- Simulate operational conditions to check for any abnormalities.

Document Maintenance Activities:

- Maintain detailed records of all maintenance tasks performed, including dates, observations, and any corrective actions taken.
- Use this documentation to track maintenance history and identify recurring issues.

Train Personnel:

- Provide training for staff on maintenance procedures and the importance of preventive maintenance.
- Encourage reporting of any unusual observations or issues that may arise during operations.

Review and Adjust Maintenance Plans:

- Regularly review maintenance schedules and procedures based on performance data and system feedback.
- Adjust plans as necessary to improve efficiency and effectiveness.

Steps for Diagnosing in an Automation System

Identify Symptoms:

- Gather information from operators and logs about the specific issues being experienced, such as erratic behavior, failures, or alarms.
- Document the symptoms and the conditions under which they occur.

Check System Documentation:

- Review system schematics, manuals, and previous maintenance records to understand the setup and history of the equipment.
- Familiarize yourself with the design, components, and functions of the automation system.

Verify Power Supply:

- Ensure that all components have a proper power supply and that there are no interruptions in the electrical feed.
- Check circuit breakers, fuses, and any other protective devices.

Inspect Physical Components:

- Conduct a visual inspection of all equipment and components for signs of wear, damage, or overheating.
- Look for loose connections, frayed wires, and any foreign objects that may be obstructing operation.

Test Inputs and Outputs:

- Use multimeters or diagnostic tools to measure input signals (from sensors or switches) and output signals (to actuators or displays).
- Confirm that all inputs are within expected ranges and that outputs respond correctly to inputs.

Monitor System Performance:

- Use monitoring tools to track the performance of the automation system in real time.
- Identify patterns or anomalies that may indicate the source of the problem.

Check for Alarms and Error Codes:

- Review the control system's alarm history and error codes to identify specific issues reported by the system.
- Use this information to guide the troubleshooting process.

Isolate the Problem:

- Systematically isolate sections of the automation system to pinpoint the area where the issue originates.
- Disconnect components as necessary to identify whether they are contributing to the problem.

Review Control Logic:

- If using a PLC, examine the control logic and program for any errors or misconfigurations.
- Test the program in a simulation environment, if available, to replicate the issue.

Consult Manufacturer Support:

- If the issue remains unresolved, consult the manufacturer's technical support or documentation for additional troubleshooting guidance.
- Consider reaching out to online forums or user groups for similar experiences.

Implement Solutions:

- Once the root cause is identified, implement the appropriate corrective actions to resolve the issue.
- Replace faulty components, adjust settings, or reprogram as necessary.

Document Findings:

- Record the diagnostic process, findings, and actions taken for future reference.
- Update maintenance records to include any changes made to the system.

Steps for Troubleshooting in an Automation System

1. **Define the Problem:**
   - Clearly describe the issue, including symptoms, error codes, and specific conditions under which the problem occurs.
   - Gather input from operators and review any logged data.
2. **Prioritize the Issue:**
   - Assess the impact of the problem on production and prioritize troubleshooting based on urgency.
   - Determine whether the issue needs immediate attention or if it can be scheduled for later.
3. **Review System Documentation:**
   - Consult system manuals, schematics, and previous troubleshooting logs to understand the design and functionality of the automation system.
   - Familiarize yourself with the operational parameters of the affected components.
4. **Check for Recent Changes:**
   - Identify any recent modifications to the system, such as software updates, hardware installations, or configuration changes, that may have triggered the issue.
   - Consider reverting to previous settings if recent changes are suspected to be the cause.

5. **Inspect Physical Components:**
   - ❖ Conduct a thorough visual inspection of relevant components, including sensors, actuators, wiring, and control panels.
   - ❖ Look for signs of wear, damage, loose connections, or foreign objects that could affect performance.
6. **Test Electrical Connections:**
   - ❖ Use a multimeter to verify power supply and continuity across connections.
   - ❖ Check for voltage drops or irregularities that could indicate faulty wiring or connections.
7. **Evaluate Inputs and Outputs:**
   - ❖ Test the input signals to the automation system (e.g., from sensors or switches) to ensure they are functioning correctly.
   - ❖ Check that output signals (e.g., to motors or valves) are responding appropriately to inputs.
8. **Analyze Control Logic:**
   - ❖ Review the control program for the automation system, focusing on the logic that governs the affected operations.
   - ❖ Identify any potential programming errors, logical flaws, or unintentional overrides.

Points to Remember

- Preventive maintenance is a proactive approach aimed at keeping automation systems in optimal condition by performing regular checks and maintenance tasks
- System diagnosing is the process of identifying and analyzing faults or issues within an automation system
- System troubleshooting is the process of diagnosing and resolving issues within an automation system to restore its normal operation

Steps for Preventive Maintenance in an Automation System

- Develop a Maintenance Schedule
- Conduct Regular Inspections
- Lubricate Moving Parts
- Calibrate Sensors
- Clean Equipment

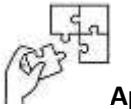
- Check Electrical Connections
- Update Software and Firmware
- Test System Performance
- Document Maintenance Activities
- Review and Adjust Maintenance Plans

Steps for Diagnosing in an Automation System

- Identify Symptoms:
- Check System Documentation
- Verify Power Supply
- Inspect Physical Components
- Test Inputs and Outputs
- Monitor System Performance
- Check for Alarms and Error Codes
- Isolate the Problem
- Review Control Logic
- Consult Manufacturer Support
- Implement Solutions
- Document Findings

Steps for Troubleshooting in an Automation System

1. Define the Problem
2. Prioritize the Issue
3. Review System Documentation
4. Check for Recent Changes
5. Inspect Physical Components
6. Test Electrical Connections
7. Evaluate Inputs and Outputs
8. Analyze Control Logic
9. Use Diagnostic Tools
10. Isolate and Test Components
11. Implement Solutions
12. Document the Process



Application of learning 3.4.

In a food packaging facility, an automated packaging system that includes conveyors, filling machines, labelling systems, and sensors has been experiencing inconsistent performance, leading to frequent stoppages and occasional mislabelling of products. To address these issues, the automation technician is tasked with applying preventive maintenance, diagnosing the underlying problems, and troubleshooting the system

Learning outcome 3 end assessment

Written assessment

I. Open Questions

1. What steps are involved in preparing a program file for downloading to a PLC?
2. Explain the significance of selecting the appropriate connection method when downloading a PLC program.
3. What are the different methods of connecting to a PLC, and how do they differ?
4. Describe the process of flashing a program file to a PLC. What precautions should be taken?
5. What are the key techniques used for testing automation systems, and how do they contribute to system reliability?
6. Discuss the advantages and disadvantages of various testing techniques applied in automation systems.
7. What is the Software Testing Life Cycle (STLC), and what are its key phases?
8. Differentiate between unit testing and system testing in the context of automation systems.
9. What elements should be included in the technical documentation for an automation system?
10. Explain the role of preventive maintenance in maintaining an automation system.

II.Multiple choice question

1. **What is the first step in program file preparation for a PLC?**

A) Flashing the program

B) Writing the program code

C) Selecting connection method

D) Testing the program

2. **Which of the following is NOT a method of connection for downloading a PLC program?**

A) USB

B) Ethernet

C) HDMI

D) RS-232

3. **What is the primary advantage of using Ethernet for PLC connections?**

A) Higher compatibility

B) Greater distance coverage

C) Simplicity of use

D) Lower cost

4. **Which technique is primarily used for identifying issues in individual modules?**

A) System testing

B) Unit testing

C) Integration testing

D) Performance testing

5. **In the STLC, what phase follows test planning?**

A) Test case design

B) Test execution

C) Requirement analysis

D) Test closure

6. **What is a primary disadvantage of exhaustive testing techniques?**

A) They are time-consuming

B) They are cost-effective

C) They guarantee bug-free software

D) They are easy to implement

7. **What should technical documentation include regarding hardware?**

A) Programming languages used

B) Hardware configuration

C) User experience

D) Testing results

8. **What is the purpose of a user manual in an automation system?**

A) To outline the maintenance schedule

B) To provide end-users with operational guidance

C) To describe testing procedures

D) To detail programming techniques

9. **Which of the following is a method of diagnosing system issues in an automation system?**

A) Preventive maintenance

B) Fault tree analysis

C) User training

D) User manual review

10. **What is the main focus of preventive maintenance?**

A) Fixing failures as they occur

B) Enhancing system performance

C) Preventing failures before they happen

D) Upgrading software regularly

Practical assessment

You are tasked with designing and implementing an automation system for a small manufacturing facility that uses a conveyor belt to transport products from one workstation to another. The system will be controlled by a Programmable Logic Controller (PLC) and will include sensors to detect the presence of products on the conveyor.

END

Reference

Books

Darvas, D., Blanco, E., & Molnár, S. V. (2019). PLCverif re-engineered: An open platform for the formal analysis of PLC programs. In *Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems, JACoW* (pp. 21-27).

Ghaleb, A., Zhioua, S., & Almulhem, A. (2018). On PLC network security. *International Journal of Critical Infrastructure Protection*, *22*, 62-69.

Mellado, J., & Núñez, F. (2022). Design of an IoT-PLC: A containerized programmable logical controller for the industry 4.0. *Journal of Industrial Information Integration*, *25*, 100250.

Schofield, B., Borrego, J., & Blanco Viñuela, E. (2022). Continuous Integration for PLC-based Control System Development. *JACoW*,478-483.

Web links

https://www.mathworks.com/help/plccoder/ug/deploy-ladder-diagram.html

https://www.electroniclinic.com/plc-timers-and-counters-their-types-and-practical-uses/

https://www.realpars.com/blog/plc-timer

RTB | RWANDA TVET BOARD

October, 2024