PROGRAMMABLE LOGIC CONTROLLERS

✅ 1. PLC Fundamentals (Core Concepts)

- **What is a PLC?**
  - Definition & role in automation
  - Advantages over hard-wired relay logic
- **PLC Architecture**
  - CPU (control + scan cycle)
  - Memory (program memory, data table, I/O image table)
  - Input modules (digital/analog)
  - Output modules (digital/analog)
  - Power supply
- **PLC Scan Cycle**
  - Input scan → Program execution → Output update
  - Relation to continuous loops in software
- **PLC Programming Languages (overview)**
  - Ladder Diagram (main focus)
  - Mention of others: FBD, ST, IL, SFC (just awareness)

✅ 2. Ladder Logic Essentials (Main Exam Meat)

- **Contacts and Coils**
  - Normally Open (XIC)
  - Normally Closed (XIO)
  - Output Coil (OTE)
  - Latch (OTL) and Unlatch (OTU)
- **Logical Operations**
  - AND (series contacts)
  - OR (parallel contacts)
  - NOT (using NC contacts)
- **Internal Memory Bits**
  - How to use internal relays (B3, M bits) for intermediate logic
- **Basic Programs**
  - Start/Stop motor latch circuit
  - Simple interlock (two conditions must be true)

✅ 3. Timers (Time-Based Control)

- On-Delay Timer (TON)
- Off-Delay Timer (TOF)
- Retentive Timer (RTO)
- Timer Reset (RES)
  (Know their use-cases and how to wire them in ladder logic)

✅ 4. Counters (Count-Based Control)

- Count Up (CTU)
- Count Down (CTD)
- Counter Reset (RES)
  (Be able to create simple counting circuits)

✅ 5. Data Handling & Math

- **Comparison Instructions**
  - Greater Than (GRT)
  - Less Than (LES)
  - Equal (EQU)
- **Math Instructions**
  - ADD, SUB, MUL, DIV
  - Scaling analog inputs (basic understanding)
- **Move Instruction (MOV)**
  - Copy data between memory locations

✅ 6. Program Structuring Techniques

- Sequencing (step-by-step control, e.g. washing machine cycle)
- Subroutines / JSR (jump to subroutine)
- State machines (using memory bits to track steps)

✅ 7. Troubleshooting & Simulation

- ### Diagnostics
  - Going online with PLC software
  - Forcing I/O (know concept)
  - Monitoring and interpreting real-time values
- ### Common Faults
  - Inputs not wired or misaddressed
  - Timers/counters not resetting as expected
- ### Simulation Practice *(you already reached this in class)*
  - Start/Stop motor latch
  - Traffic light (timers)
  - Bottle filling/counting (counters)
  - Door interlock (AND/OR logic)

✅ 8. Extra (If Time Allows / Bonus Points)

- Analog I/O handling (4–20 mA sensors, scaling raw values)
- Safety interlocks (why certain conditions must be enforced before outputs energize)

✨ Your Study Priority

If time is tight, focus in this order:
Ladder Logic Basics → Timers → Counters → Comparisons/Math → Structuring → Troubleshooting/Simulation.

💡 Practical Tip

✅ Spend most of your time in a simulator (OpenPLC or LogixPro).
✅ Build small circuits for each topic.
✅ Write tiny notes after each topic with a quick ladder snippet.

✅ 9. Minor Topic Additions for a Full House 🏠

Data Addressing & Naming Conventions: Understand how different PLC brands label inputs, outputs, timers, and internal bits (e.g., I:1/0, O:2/1, M0.0), so you can read and write ladder logic accurately.

Basic Error Handling & Safety Chains: Learn how to use latches/unlatches or fault bits to flag abnormal conditions, and design safety interlocks (like E-stops) that override all other logic when triggered.

PLC Operating Modes: Know the difference between RUN, PROG/STOP, and REMOTE modes, and how each affects program execution and editing.

Troubleshooting Scenarios: Be ready to diagnose practical issues (e.g., motor not starting despite input signal, counters double-counting) by tracing logic, wiring, and I/O behavior.

✨ WHAT IS A PLC?

PLCs are a specialized industrial computer that controls and automates processes in various industries.

PLCs are designed to be rugged, reliable, and easily programmed to **monitor** and **control** machines and processes.

A **Programmable Logic Controller (PLC)** is basically a rugged, specialized industrial computer built to run automation tasks. Rugged means they can operate in harsh environments.

👉 It *takes in signals* from sensors and switches, processes them according to a stored program, and then sends commands to outputs like motors, valves, or lights.

Key real-world use cases:

- Controlling machines on a factory assembly line.
- Managing amusement ride rollercoasters as they twist and turn.
- Automating food-processing machinery that mix ingredients for your favorite snack.

Why PLCs instead of normal PCs?

Designed to Handle Digital & Analog I/O:

PLCs have built-in I/O modules to directly handle digital and analog signals from industrial devices, while regular PCs need extra hardware and software, making them slower and harder to use for real-time control.

Survive Extreme Temperatures:

PLCs are rugged and built to survive extreme temperatures, while regular PCs can't handle the heat—or the cold—of harsh industrial environments.

Immune to Electrical Noise:

PLCs are built to resist electrical noise from heavy machinery, while regular PCs can glitch or fail in such interference-heavy environments.

Resist Vibration and Impact:

PLCs are designed with sturdy, shock-resistant casings and components, enabling them to withstand vibrations, impacts, and rough handling while regular PCs are too fragile and prone to hardware failures under physical stress.

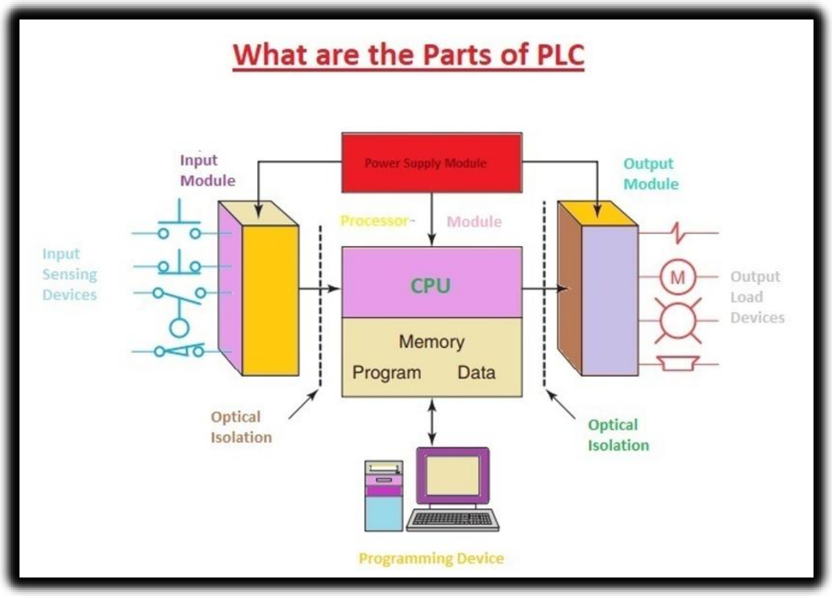
🧠 Core Sections of a PLC

CPU (Central Processing Unit)

The PLC's "brain."

Contains a microprocessor, memory chips, and circuits for control logic, monitoring, and communication.

- **Microprocessor**: The brain of the PLC, executing tasks and calculations quickly.
- **Memory Chips**: Store the program and real-time data, like sensor states.
- **Control Logic & Communication Circuits**: Internal pathways for CPU communication with memory, I/O, and other devices.

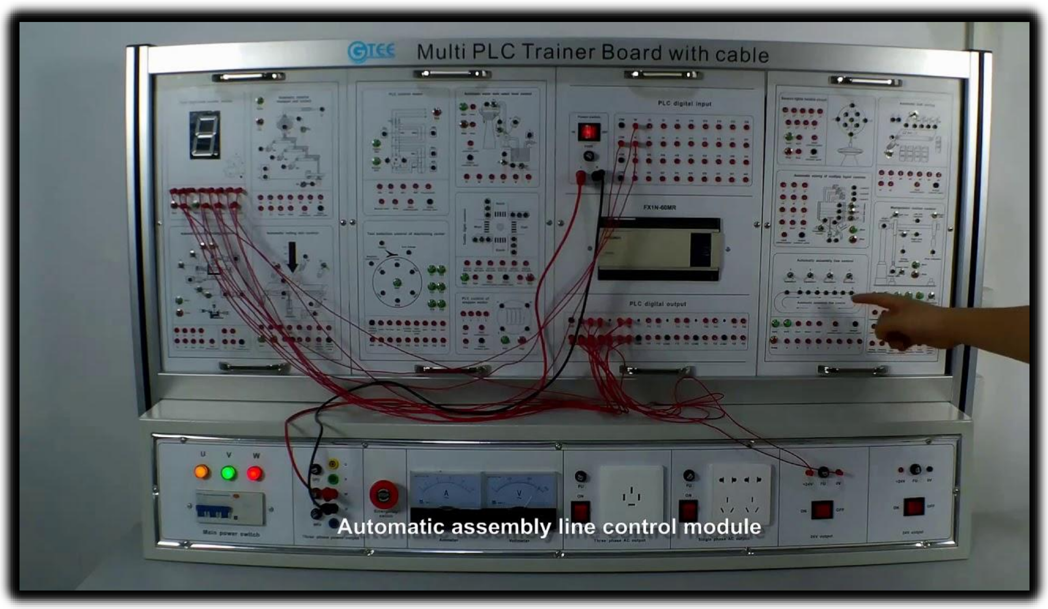


⚡ I/O Interface System

PLC's "nerves": These are the input/output (I/O) systems that receive signals from sensors/switches and send commands to actuators (motors, lamps, relays).

Inputs (The Senses – How the PLC Gathers Info): 👂 👀

- This part of the PLC is constantly **listening and watching** for signals from the outside world.
- **Field devices** are like the "reporters" sending info to the PLC.
- **Sensors:** Imagine a sensor on a factory line detecting if a product is in the right spot. That's an input telling the PLC, "Hey, item confirmed!" Or a temperature sensor letting the PLC know, "It's getting too hot in here!" 🔥
- **Switches:** Think of a simple "ON/OFF" button a worker presses. That button sends an electrical signal (an input) to the PLC, saying, "Let's get this show on the road!" 🟢
- It's like your smartphone getting a notification – someone sent you a message, or an alarm just went off. The PLC receives similar digital or analog "notifications."



PLC Training Board.

Outputs (The Muscles – How the PLC Takes Action): 🦾 🗣️

- Once the PLC's brain (CPU) has processed the inputs and made a decision, the output interface is how it **sends commands** back out to the real world to make things happen.
- **Actuators** are the "doers" that receive commands from the PLC and perform a physical action.
- **Motors:** If the PLC decides to start a conveyor belt, it sends an electrical signal (an output) to the motor, telling it to spin. It's like you telling your smart speaker to play music. 🎶
- **Lamps/Lights:** If a machine completes a task, the PLC might turn on a "Task Complete" light. Or, if there's an error, it might flash a warning lamp. 🚨
- **Relays:** These are like remote-controlled switches. The PLC might activate a relay, which in turn switches on a high-power device (like a large industrial heater or a powerful pump) that the PLC itself can't directly power.
- It's like your smart home system turning on the lights when you walk into a room, or adjusting the thermostat based on the temperature. The PLC is giving commands to its environment.

⚡ CPU Operating Modes

Just like you might switch between "work mode" and "chill mode," the PLC's CPU has distinct operating modes:

- **Programming Mode:** This is where the CPU gets its **new instructions**. When you connect a PC and "download" your PLC program, the CPU is in this mode, **ready to accept and save those changes** to its memory. It's like updating the PLC's brain with new skills.
- **Run Mode:** Once the program is downloaded and confirmed, you switch the CPU to Run Mode, where the PLC **springs into action**! In this mode, the CPU constantly executes the program stored in its memory.

💾 Memory in the CPU

- Stores the **program logic**
- Tracks the current **status of inputs and outputs**
- Holds **values and data** (timers, counters, internal bits)

The Scan Time: Blazing Fast Automation! 🚀

Because a PLC is a **dedicated controller** (meaning its sole purpose is to run that one control program), it cycles through its tasks tirelessly and incredibly fast.

1. It first **reads the status of all connected input devices** (like checking if any switches are pressed or sensors are activated).
2. Then, it **processes this input data** through the stored control program.
3. Finally, based on the program's logic and the input status, it **updates the status of its output devices** (like turning a motor on or off).

The time it takes for the CPU to complete **one full cycle** – reading inputs, executing the program, and updating outputs – is called the **scan time**.

This happens with mind-blowing speed, often in the range of **1/1000th of a second** (that's 1 millisecond!).

Imagine blinking, and the PLC has already completed several hundred scans!

This rapid cycling is crucial for ensuring that industrial processes respond instantly to changes in the real world.

🔥 Why this matters for your exam

- You can now clearly define what a PLC is, where it's used, and why it's built tough.
- You can explain its main parts (CPU and I/O), modes (program/run), and how the scan cycle works.
- You understand what "scan time" means and why memory is crucial.