PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

NOUR BACHIR EL BAYADH UNIVERSITY CENTER

Nour Bachir University Center, El Bayadh, Algeria

Institute of Technology

Department of Electrical Engineering

COURSE HANDOUT

TITLE:

Programmable Logic Controllers(PLC)

Réalisé par : Dr. BENALI Abdelkrim

University Year: 2024 / 2025

Field: Science and Technology

Sector: Electronics

Specialization: Embedded Systems Electronics

Semester S2:

Course Unit: UEF 1.2.2

Subject 2: Programmable Logic Controllers (PLC)

VHS: 45 hours/15 weeks (Lectures: 1.5 hours/week, Tutorials: 1.5 hours/week)

Credits: 4

Coefficient: 2

Course Objectives:

This course allows students to understand the hardware and software structure of Programmable Logic Controllers (PLCs), to select a PLC and associated components based on the desired application, and to use a programming language appropriate for the PLC.

Recommended Prior Knowledge:

Combinational and Sequential Logic, Microprocessors, Microcontrollers, Sensors, Industrial Networks and Communications.

Course Contents:

Chapter 1: Programmable Logic Controllers PLCs (2 weeks)

Definition of a PLC, Internal and External Architecture of a PLC and Characteristics. Selection of a PLC. Types of Inputs/Outputs of a PLC and their Characteristics.

Chapter 2: Materialization of Industrial Processes by PLCs (3 weeks)

Definition of an Automated System. The Essential Parts of an Automated System (PO, PC, HMI, Interfacing). Operating Principle of a PLC and an Automated Order-Information System. Wiring. Sensor-Actuator Concepts, Industrial Networks, etc.

Chapter 3: PLC Programming (5 weeks)

Introduction to Grafcet. Introduction to the languages: LD, IL, FBD, SFC, SCL. Application: Definition of PO-PC Parts, Development of the Grafcet, Ladder Programming. Application Exercises.

Chapter 4: Process Visualization (3 weeks)

Introduction to HMI (Human Machine Interface) and SCADA systems, process representation and control, alarm display, recipes, archiving, user management, etc.

Chapter 5: Programmable Logic Controller (PLC) for Safety (2 weeks)

Architecture, process and machine control, and safety function management.

Assessment Method:

Continuous Assessment: 40%; Exam: 60%.

Chapter I:

Programmable Logic Controllers PLCs

1. Introduction

The Programmable Logic Controller (PLC) is a programmable electronic device, suitable for the industrial environment, which performs automation functions to ensure the control of sensors and actuators based on logical, analog, or digital information.

The Industrial Programmable Logic Controller (PLC) is a programmable electronic device designed for the industrial environment. It performs automation functions to control actuators based on logical, analog, or digital information. A PLC is a specific type of microprocessor controller that utilizes programmable memory to store instructions and implements various functions such as logical operations, sequencing, timing, counting, or arithmetic operations to control machines and processes.

Industrial PLCs emerged in the late 1960s, driven by demands from the American automotive industry (GM) for more adaptable control systems. The decreasing costs of electronics at the time made it advantageous to replace existing technologies. PLCs are widely used across industrial sectors for controlling machinery (conveying, packaging) or production lines (automotive, food processing), as well as for process regulation (metallurgy, chemistry). They are increasingly employed in building automation (commercial and industrial) for tasks like heating control, elevator operation, lighting, security, and alarm systems.

Previously, electromechanical relays and pneumatic systems were used for control, known as hardwired logic. However, they were expensive, inflexible, and lacked communication capabilities. The solution was the adoption of microprocessor-based systems, enabling easy modification of automated systems, known as programmed logic.

Since the computers of that time were expensive and unsuitable for industrial constraints, programmable controllers were developed to meet the needs of the industry.

The industrial Constraints can be cited as :

a) External Influences:

- Dust,
- Temperature,
- Humidity,
- Vibrations,
- Electromagnetic interference, etc.

b) Personnel:

- Easy hardware implementation (no complex programming language required).
- Troubleshooting feasible by electromechanical technicians.
- Ability to modify the system during operation.

c) Hardware:

- Scalable

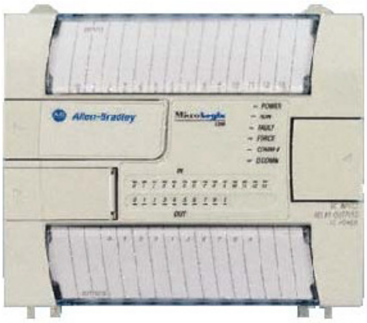
- Modular
- Easy implementation

2. PLC Architecture:

a. PLC External Aspect:

Automation systems can be either compact or modular. Among the compact types, programming modules such as Siemens' LOGO, Schneider's ZELIO, or Crouzet's MILLENIUM are distinguished. They integrate the processor, power supply, inputs, and outputs. Depending on the models and manufacturers, they may perform additional functions (rapid counting, analog I/O...) and accept a limited number of extensions. These automation systems, with simple operation, are generally intended for controlling small-scale automation.

On the other hand, modular types have the processor, power supply, and input/output interfaces housed in separate units (modules) fixed on one or more racks containing the "backplane" (bus and connectors). These automation systems are integrated into complex automation setups where power, processing capacity, and flexibility are required.

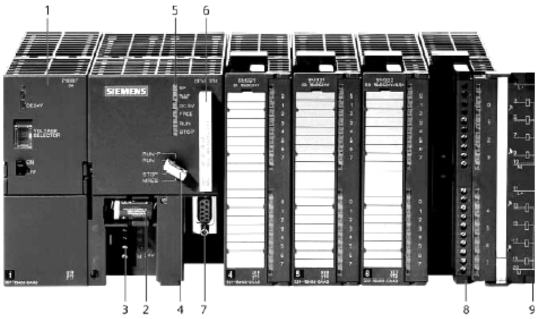


(Allen-bradley)

(Zelio of Schneider)

Figure I.1: Compact PLCs

(Modicon)



(Siemens)

1 Power module
2 Backup battery
3 Connection to 24V DC
4 Mode switch (key-operated)
5 Status and fault indication LED
6 Memory card
7 Multipoint Interface (MPI)
8 Front connector
9 Front panel flap

Figure I.2: Modular PLC

b. Internal Structure:

The internal structure of the PLC consists of:

- Power module: it ensures the distribution of power to the different modules.
- Central processing unit (CPU): based on a microprocessor, it performs all logical, arithmetic, and digital processing functions (transfer, counting, timing, etc.).
- Internal bus: it enables communication between all blocks of the PLC and any extensions.
- Memories: They allow storage of the operating system (ROM or PROM), the program (EEPROM), and system data during operation (RAM). The latter is generally backed up by a battery or a capacitor. Memory capacity can typically be increased by adding PCMCIA memory cards.
- Input/output interfaces:
- Input interface: it receives information from the automated production system (APS) or the control panel and formats (filters, etc.) this signal while electrically isolating it (optical coupling).
- Output interface: it controls various actuators and signaling elements of the automated production system while ensuring electrical isolation.

Compact PLCs can control All-or-nothing outputs and sometimes handle counting and analog processing functions.

Modular PLCs enable the realization of numerous other functions through intelligent modules placed on one or more racks. These modules have the advantage of not overloading the CPU's workload as they often have their own processor.

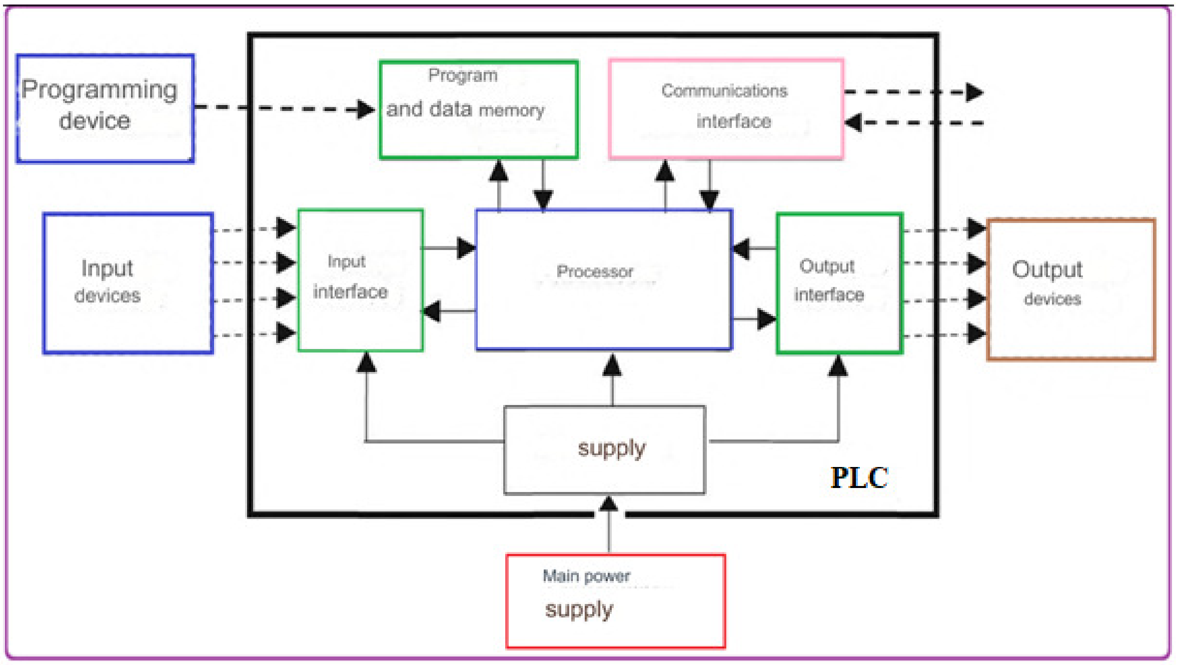


Figure I.3: PLC general structure

3. Principle of operation of a PLC:

When dealing with a computer, program execution generally occurs line by line and asynchronously. One characteristic of a PLC is to operate differently, i.e., cyclically. Indeed, before executing anything, the PLC reads its entire program, and once the execution is complete, it repeats the same operations. The notion of a cycle and cycle time (between approximately 1ms and 30ms) is then defined. There are several types of cycles, but the most common one is the one represented in Figure 1.6.

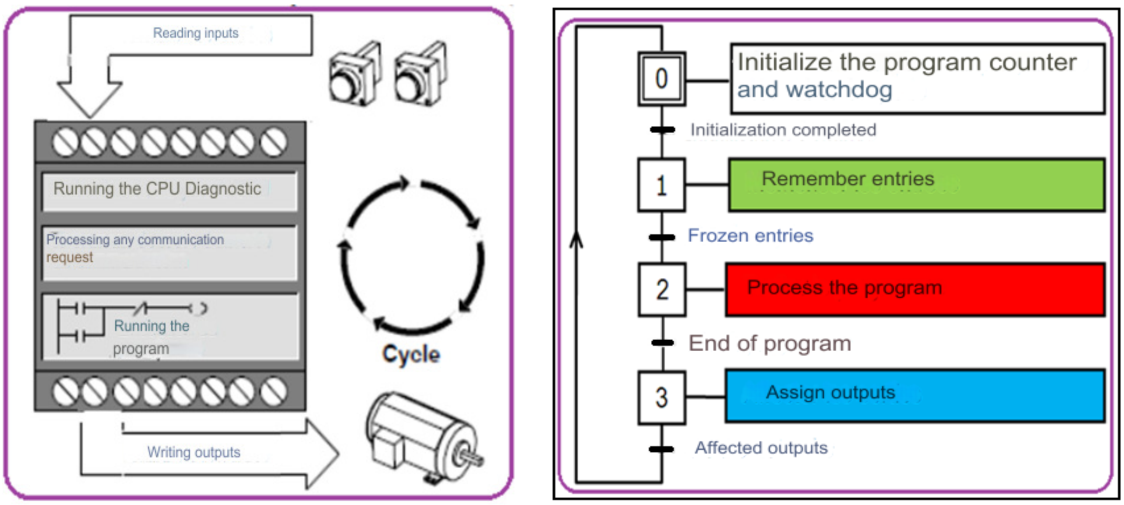


Figure I.4: Typical cycle of program execution of an Industrial Programmable Logic Controller.

This cycle comprises 5 phases:

• **Phase 1:** Input Reading or Acquisition: Incorporation of input module information and writing their values into RAM (DATA area).

• **Phase 2:** Program Execution or Data Processing: Reading of the program (located in the program RAM) by the processing unit, reading of variables (data RAM), processing, and writing of variables (internal, outputs...) into the data RAM.

• **Phase 3:** Processing of any communication requests.

• **Phase 4:** Execution of self-diagnostic test (Management of the PLC's self-control system).

• **Phase 5:** Output Writing: Reading of output variables in the data RAM and transfer to the output module. The cycle scanning time is checked by a timer called Watchdog, which triggers an alarm procedure in case of exceeding (set by the user).

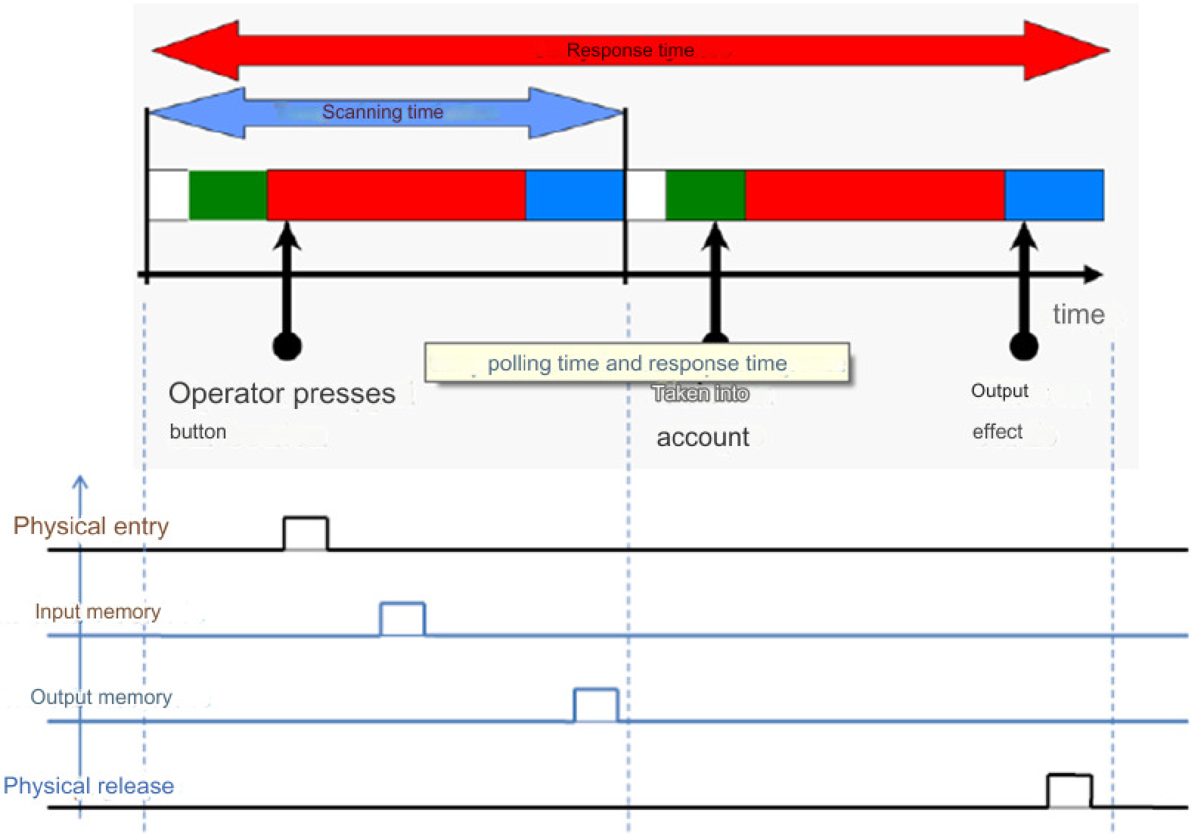


Figure I.5: The stages of response time of a PLC

4. Main functions of PLC cards:

- **Input/output cards:** Available in quantities of 4, 8, 16, or 32, they can perform input, output, or both functions. They are the most commonly used and the available voltages are standardized (24, 48, 110, or 230V DC or AC...). Channels can be independent or have "common" connections. Input cards gather information from sensors, buttons, etc., and represent it with a bit image of the sensor's state. Output cards offer two types of technologies: electromagnetic relay outputs (coil plus contact) and static outputs (based on transistors or triacs).
- **Fast counting cards:** They allow the acquisition of high-frequency information incompatible with the PLC's processing time. Example: signal from a position encoder.
- **Axis control cards:** They ensure precise positioning of mechanical elements along one or more axes. For example, the card can control a servo motor and receive position information from an encoder. Closed-loop position control can be achieved.
- **Analog input/output cards:** They allow the acquisition of an analog signal and its digital conversion (ADC), necessary for processing by the microprocessor. The inverse function (analog output) is also available. Analog quantities are standardized: 0-10V or 4-20mA.

Other cards:

- PID control cards
- Weighing cards

- Communication cards (Ethernet...)
- Remote input/output cards

5. PLC inputs and outputs interface

The inputs of the PLC receive information from detection elements (sensors) and the operator panel (HMI).

The outputs of the PLC transmit information to actuators (relays, solenoid valves...) and signaling elements (indicators) of the operator panel.

a) Input interfaces:

They are intended to:

- Receive information from sensors
- Process the signal by shaping it, eliminating noise, and electrically isolating the control unit from the operating part.

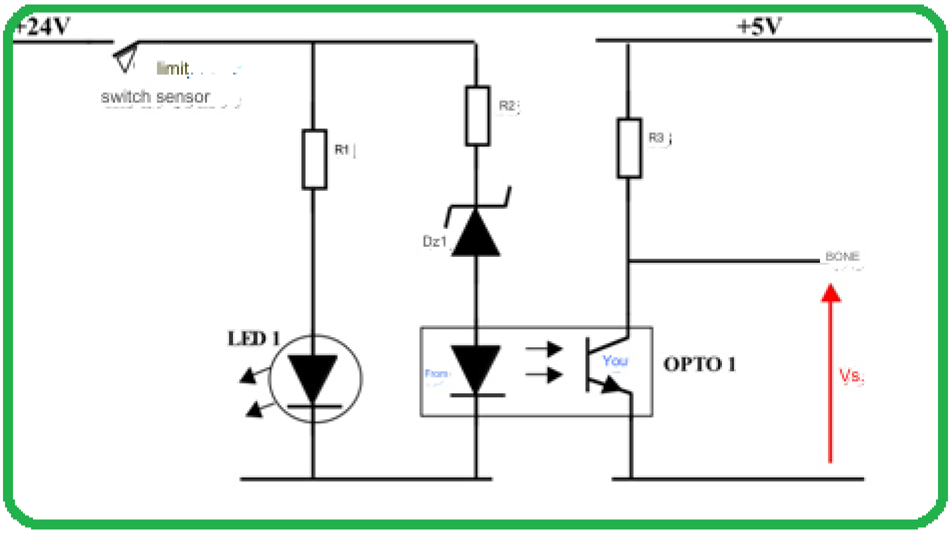


Figure I.6: Example of input interface

When the sensor is closed:

- LED1 indicates that the PLC input is activated
- The photocoupler LED lights up
- The phototransistor T' of the photocoupler becomes conductive
- The voltage Vs = 0V

Thus, when a PLC input is activated, the input interface sends a logical 0 to the processing unit and a logical 1 when the sensor contact is open (input not activated).

b) Output Interfaces:

They are designed to:

- Control the pre-actuators and signaling elements of the system
- Adapt the voltage levels of the control unit to those of the system's operative part while ensuring galvanic isolation between them.

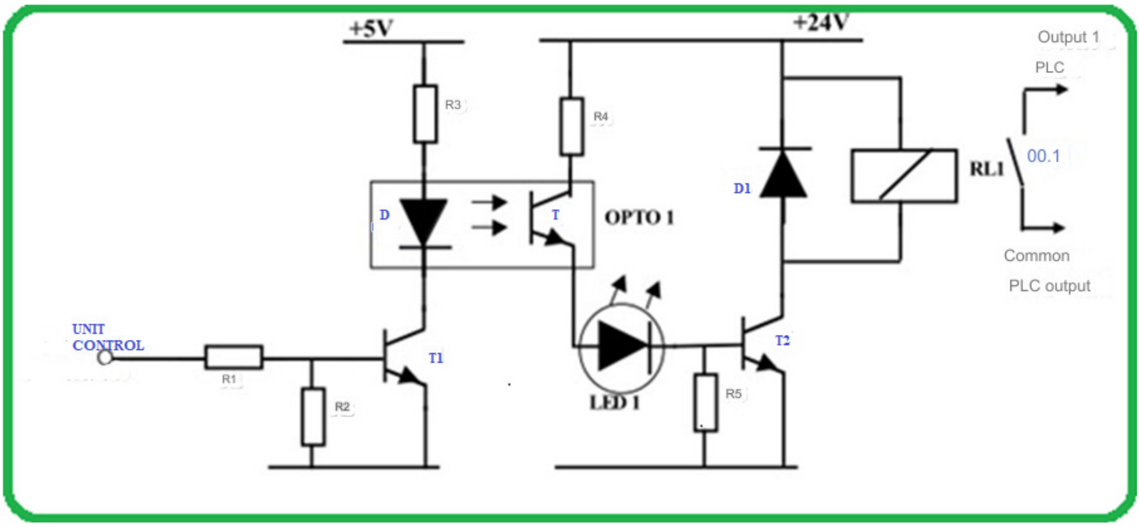


Figure I.7: Example of output interface

Operation of the output interface:

When commanding a PLC output:

- The control unit sends a logic 1 (5V).
- T1 becomes conducting, so D' lights up.
- The phototransistor T' of the photocoupler becomes conducting.
- LED 1 lights up, informing us of the command of output O0,1.
- T2 becomes conducting.
- Relay coil RL1 becomes energized and controls the closing of the contact of output O0,1.

Therefore, to command a PLC output, the control unit must send:

- A logic 1 to activate a PLC output.
- A logic 0 to stop the command of a PLC output.

Choice of a PLC

The choice of a PLC depends on the control part to be programmed. Several criteria must be considered.

- Number of integrated inputs/outputs: The first parameter to consider when choosing a PLC is the number of inputs and outputs required. There may be a base unit and extensions, or a central unit and input or output cards. Therefore, an inventory of inputs and outputs should be made.

- Type of inputs/outputs: Inputs and outputs can be:

i) Digital: on/off inputs and outputs,

ii) Analog: connection with a tachymetric generator as input and a speed controller as output, for example.

iii) Numeric: fast counting on an incremental encoder. Each input or output must be adapted to the sensor or actuator.

- The cards ensure galvanic isolation between the central unit and the system. Output cards can be relay-based or transistor-based. Relay-based cards provide a break between the power supply and the actuator but are relatively slow. Transistor-based cards switch more quickly but do not provide electrical isolation.
- Processing time (scanning).
- Memory capacity.
- Number of counters.
- Number of timers.
- Special functions or modules: Some cards (axis control, weighing, etc.) will "relieve" the processor and should offer the desired characteristics (resolution, etc.).
- Communication functions: The PLC must be able to communicate with other control systems (PLCs, supervision systems, etc.) and offer communication possibilities with standardized standards (Profibus, etc.).

Chapter II:

Materialization of Industrial Processes through PLCs

1. Introduction

A Programmable Logic Controller (PLC) serves as the cornerstone of automated production systems by providing a versatile, reliable, and programmable means to control and coordinate various machinery, devices, and processes. PLCs act as the central control unit within automated production systems. They receive inputs from sensors, switches, and other devices, process these inputs based on pre-programmed logic, and send commands to outputs such as motors, valves, actuators, or alarms. This centralized control allows for precise coordination of various components within the production line, ensuring that each step is executed accurately and in the correct sequence.

2. Automated Production Systems:

The objective of automating systems is to produce products of quality while minimizing human intervention and cost.

An automated system is a set of interacting elements organized with a specific purpose: to act upon raw materials to add value.

Automated systems are subject to various constraints: energy, configuration, adjustment, and operational constraints, which affect all modes of operation and shutdown of the system.

Automated Production Systems refer to a comprehensive setup where various processes and operations are automatically controlled and managed by machines, computers, and robots with minimal human intervention. These systems are designed to enhance efficiency, accuracy, and consistency in manufacturing, assembly, or packaging processes. Components of an automated production system may include PLCs, robotics, conveyors, sensors, Human-Machine Interfaces (HMIs), and other control equipment. Automated systems aim to improve production speed, reduce costs, ensure quality, and increase overall productivity in industries like automotive, electronics, food processing, and pharmaceuticals.

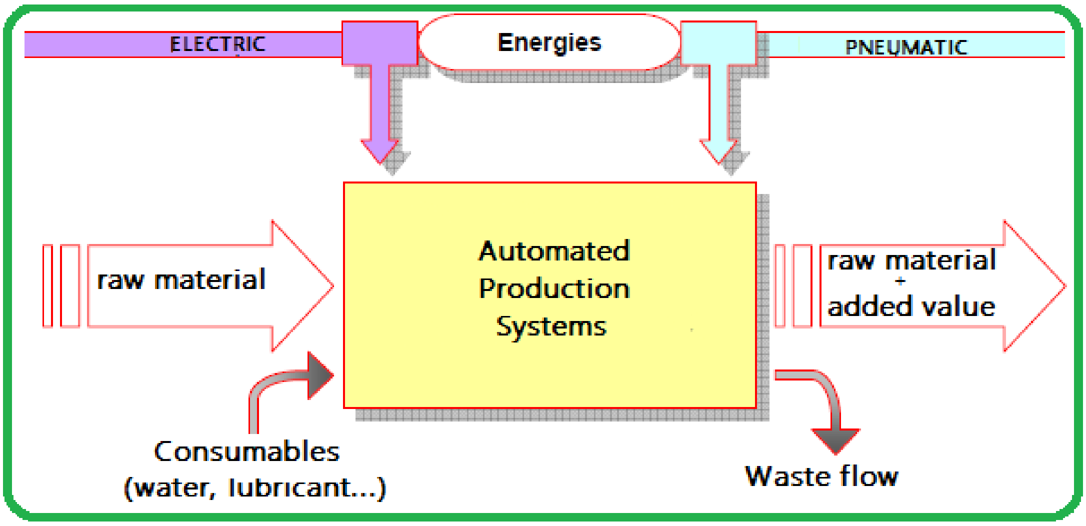

ELECTRIC
Energies
PNEUMATIC
raw material
Automated Production Systems
raw material + added value
Consumables (water, lubricant...)
Waste flow

Figure II.1: Principle of an Automated Production System

The objectives pursued by automation can be quite varied. Some of them include:

- Seeking lower costs by reducing labor expenses, material economy, energy savings, etc.
- Eliminating dangerous or arduous tasks and improving working conditions.
- Performing operations that are impossible to control manually.

Structure of an Automated System:

Any automated system can be decomposed according to the following scheme**:**

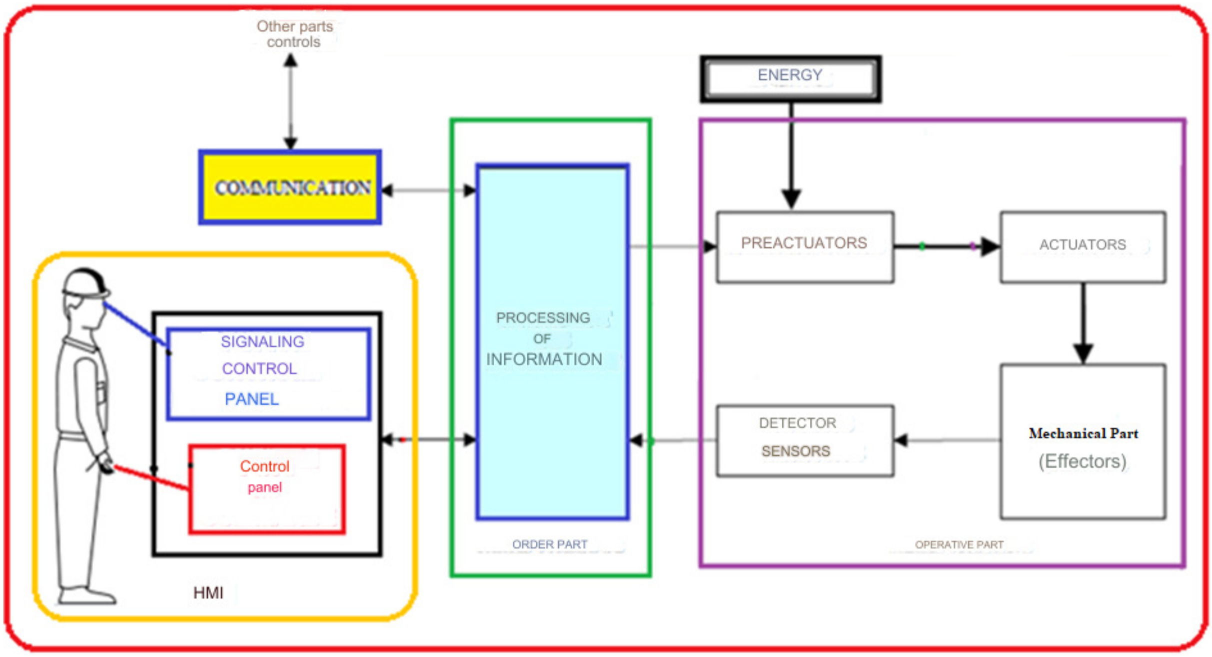


Figure II.2: Structure of an Automated System

1- Operational Part:

This is the visible part of the system. It acts on the raw material to add value.

Actuators (motors, cylinders) act on the mechanical part of the system, which in turn acts on the raw material.

Sensors/detectors acquire various states of the system.

Pre-actuators control the actuators; they transfer energy between the power source (electrical network, pneumatic, etc.) and the actuators. Example: contactor, distributor, etc.

These pre-actuators are controlled in turn by the information processing block.

2- Control Part:

It gives operational orders to the operational part then pre-actuators receive instructions from the control panel (operator) and information from the operational part transmitted by the sensors/detectors.

Based on these instructions and its task management program (implemented in a programmable logic controller or realized by relays - referred to as wired logic), it commands the pre-actuators and sends information back to the signaling panel or to other control and/or supervision systems using a network and communication protocol.

3- Control Station:

Comprising control and signaling panels, it allows the operator to control the system (start, stop, cycle start, etc.).

It also allows visualizing the different states of the system using indicators, dialogue terminals, or human-machine interfaces (HMI).

Nature of Information Processed by the Controller:

The information can be of the following types:

- On/Off: the information can only have two states (true/false, 0 or 1, etc.). This is the type of information delivered by a detector, a push button, etc.
- Analog: the information is continuous and can take a value within a well-defined range. This is the type of information delivered by a sensor (pressure, temperature, etc.).
- Digital: the information is contained in coded words in binary or hexadecimal form. This is the type of information delivered by a computer or an intelligent module.

2. Automation Component Technology

Detectors are part of the sensor family; their function is to detect the presence of an object, so the output information of a detector is binary type.

Characteristics of a sensor:

- Measurement range: Extreme values that can be measured by the sensor.
- Resolution: Smallest variation in measurable quantity by the sensor.
- Sensitivity: Variation of the output signal compared to the variation of the input signal.
- Accuracy: Ability of the sensor to give a measurement close to the true value.
- Speed: Reaction time of the sensor.
- Linearity: Represents the sensitivity deviation over the measurement range.
- Influence quantities: Physical quantity other than the measurand whose variation can modify the sensor's response.

Examples of Use:

- Detection of the position of workpiece pallets.
- Detection of the presence or absence of basic components at workstations.
- Detection of actuator position.
- Detection of vacuum on the cap head.
- Detection of package presence.

a) Electromechanical Detectors:

Position switches are present in all automated installations as well as in various applications due to the many inherent advantages of their technology. They transmit to the processing system information about:

- Presence/absence,
- Passage,
- Positioning,
- End of travel.

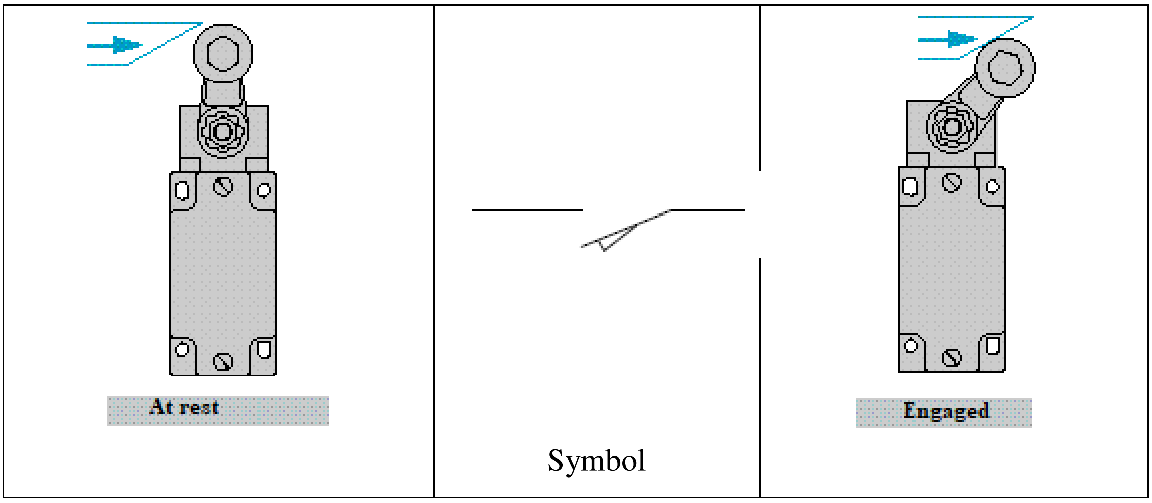


Figure II.3: Position switches

Devices of great simplicity in implementation, offering many advantages.

From an electrical point of view:

- Galvanic separation of circuits,
- Very good ability to switch low current loads, depending on the model, combined with high electrical endurance,
- Very good short-circuit resistance in coordination with appropriate fuses,
- Total immunity to electromechanical interference,
- High operating voltage.

From a mechanical point of view:

- Positive opening operation of contacts,
- High resistance to various industrial environments (standardized and specific tests in the laboratory),
- Good fidelity, up to 0.01 mm at the switching points,
- Simple visualized operation.

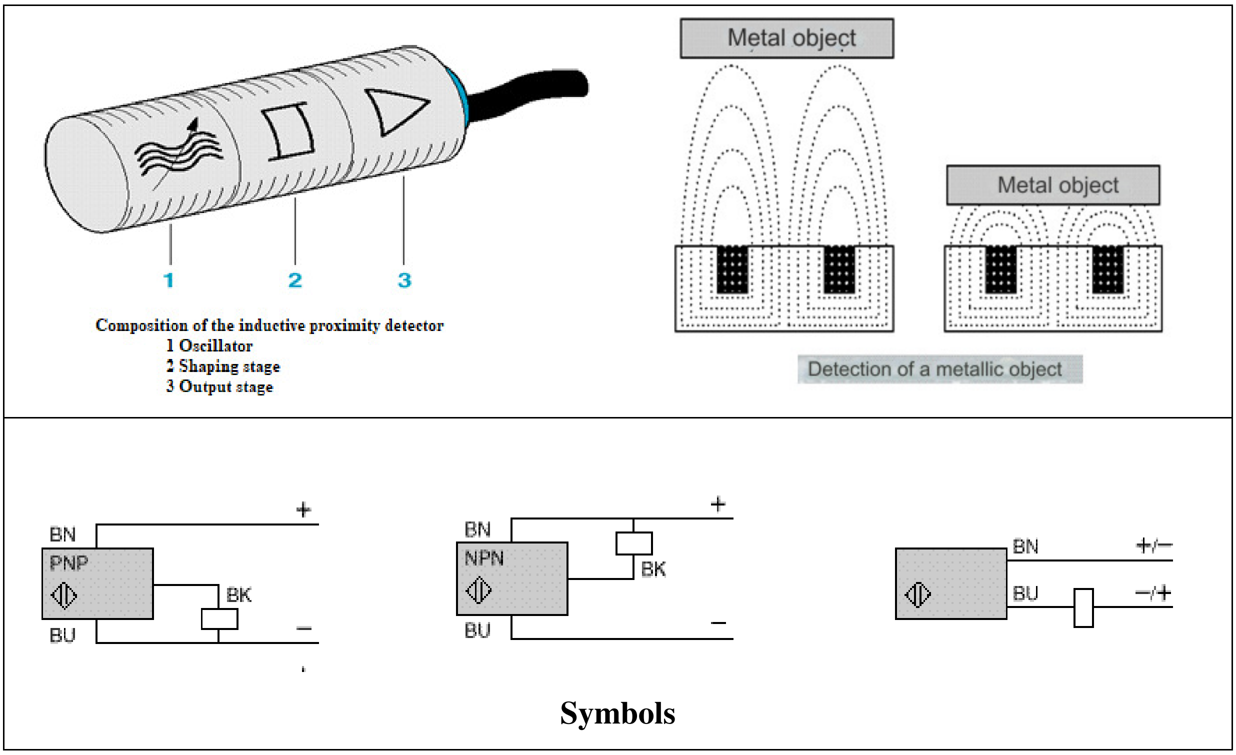
b) Inductive Proximity Sensors

This type of sensor is used for the detection of metallic objects. It allows detection of the object without contact.

When a metal screen is placed in the magnetic field of the sensor, induced currents constitute an additional load that stops the oscillations.

After shaping, an output signal corresponding to a normally open (NO) contact closure, normally closed (NC) contact opening, or complementary NO + NC is delivered.

Constitution of these sensors Operation



Composition of the inductive proximity detector
1 Oscillator
2 Shaping stage
3 Output stage

Detection of a metallic object

Symbols

Figure II.4: Constitution of an Inductive Proximity Sensor

c) Capacitive Proximity Sensors

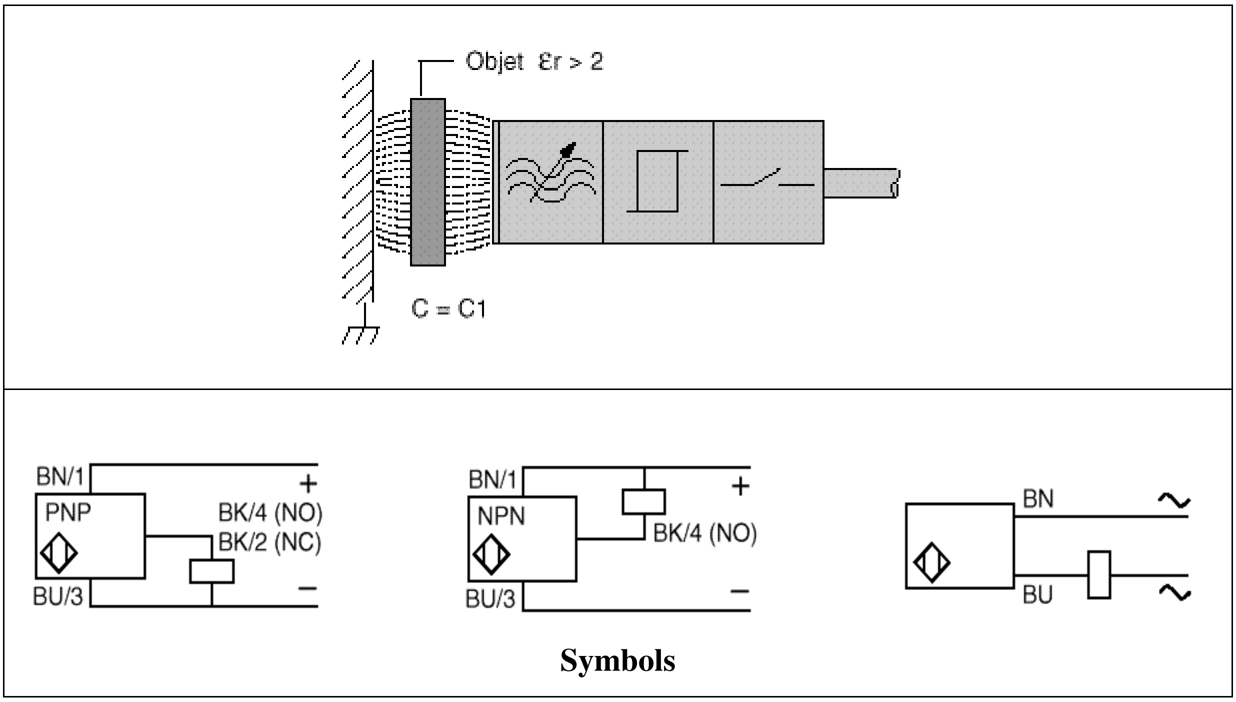
This type of sensor is used for the detection of objects of all types. It allows detection of the object without contact.

A capacitive proximity sensor is mainly constituted of an oscillator, with the capacitor formed by 2 electrodes placed at the front of the device. In the air (εr = 1), the capacitance of this capacitor is C0.

εr is the dielectric constant, which depends on the material's nature.

Any material with εr > 2 will be detected.

When an object of any nature (εr > 2) is in front of the sensitive face of the sensor, this results in a variation of the capacitive coupling (C1). This capacity variation (C1 > C0) triggers the oscillator. After shaping, an output signal is delivered.



Symbols

Figure II.4: Constitution of a Capacitive Proximity Sensor

d) Photoelectric Detector

A photoelectric detector essentially consists of a light emitter (light-emitting diode) combined with a receiver sensitive to the amount of light received (phototransistor).

Detection occurs when the target enters the light beam emitted by the detector and sufficiently changes the amount of light received by the receiver to cause a change in the output state.

Principle of Barrier-Type Sensors

The sensor consists of a transmitter and a receiver. When the object to be detected interrupts the beam, in the absence of a light beam, the receiver switches the output.

This detector has a long range.

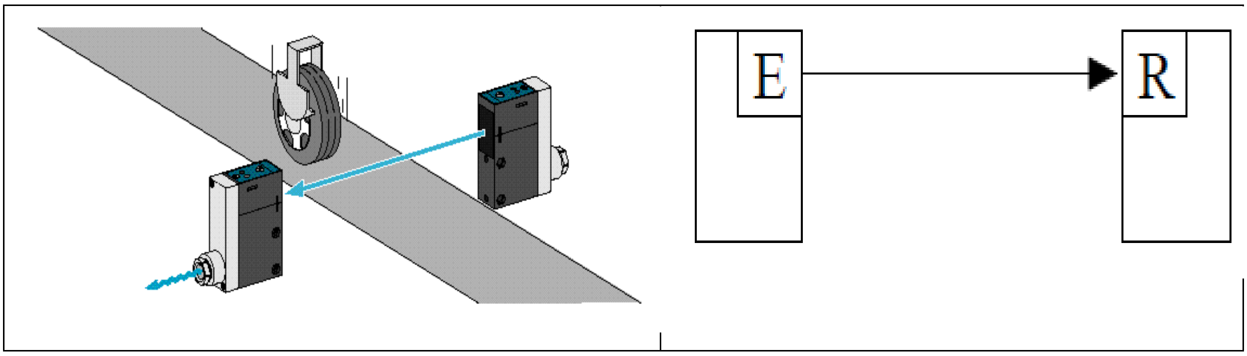


Figure II.5: Barrier-type sensor

Principle of Reflex-Type Sensors

The sensor consists of a transmitter and a receiver placed in the same housing. The beam is reflected by a reflector. When the object to be detected interrupts the beam, in the absence of a light beam, the receiver switches the output. This detector primarily detects dark objects.

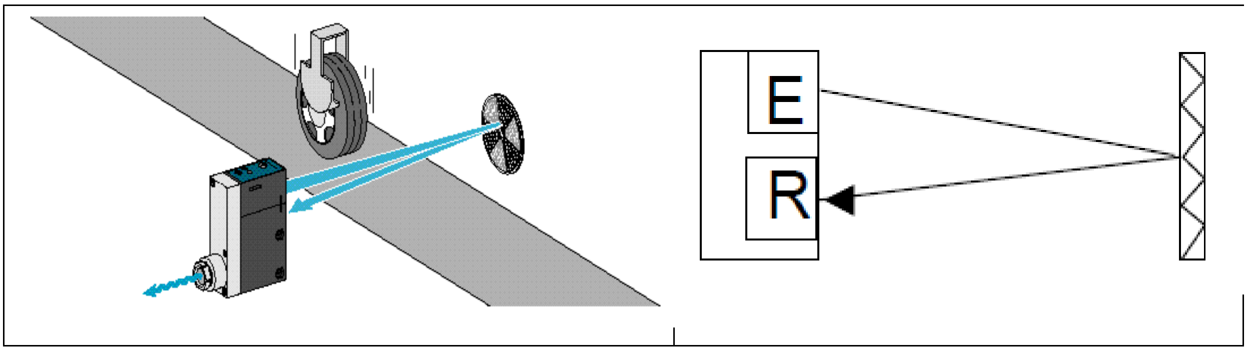
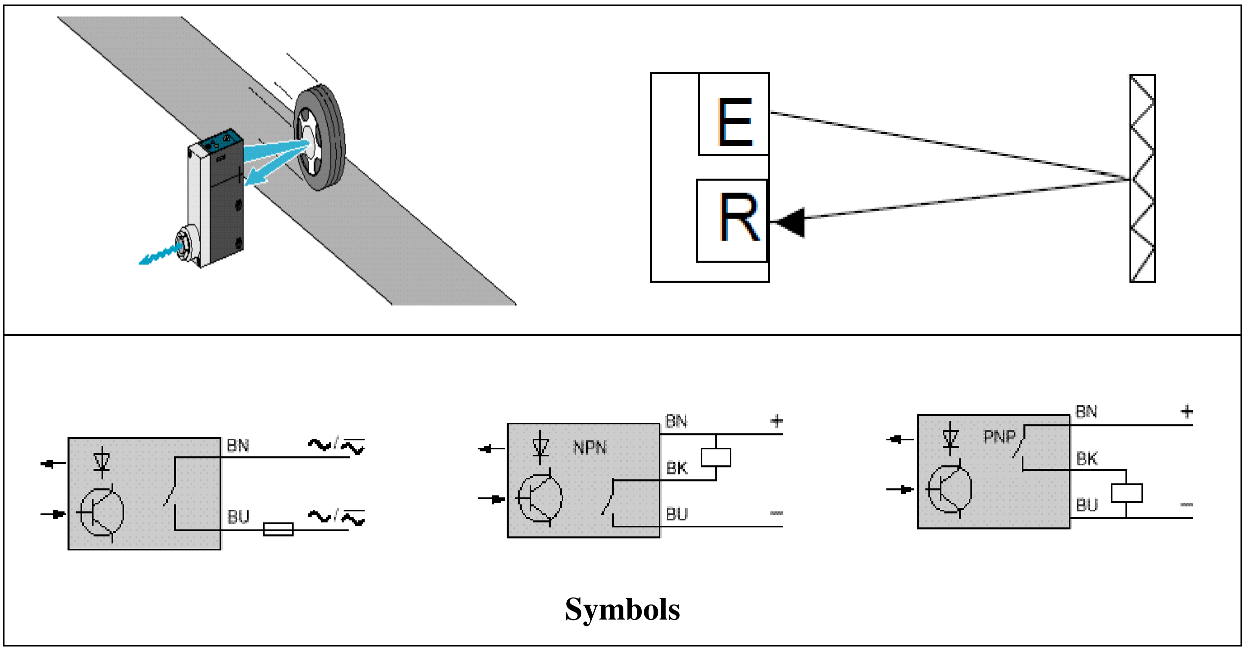


Figure II.6: Reflex-Type Sensor

Principle of Proximity Sensors

The sensor consists of a transmitter and a receiver placed in the same housing. The beam is reflected by the object to be detected. When the object reflects the beam, in the presence of the light beam, the receiver switches the output. This detector detects light-colored objects.



Symbols

Figure II.6: Reflex-Type Proximity Sensor

The emitted signal is provided by a light-emitting diode. This signal can be emitted directly (conventional detectors, as in the case of the postal sorting system) or transmitted to the emission zone via an optical fiber (transfer system). Fiber optic sensors allow the detection of very small parts.

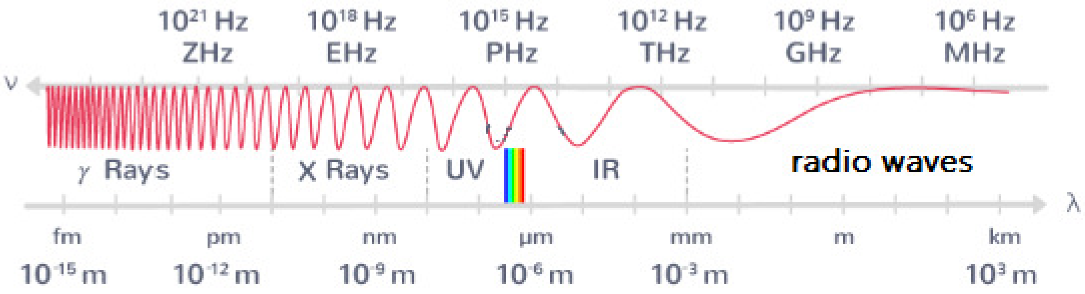


Figure II.7: Wavelengths of Emitted Signals

Photoelectric sensors using optical fibers have light-emitting diodes emitting in the red spectrum. To prevent this signal from being disturbed by ambient light, these signals are modulated before emission.

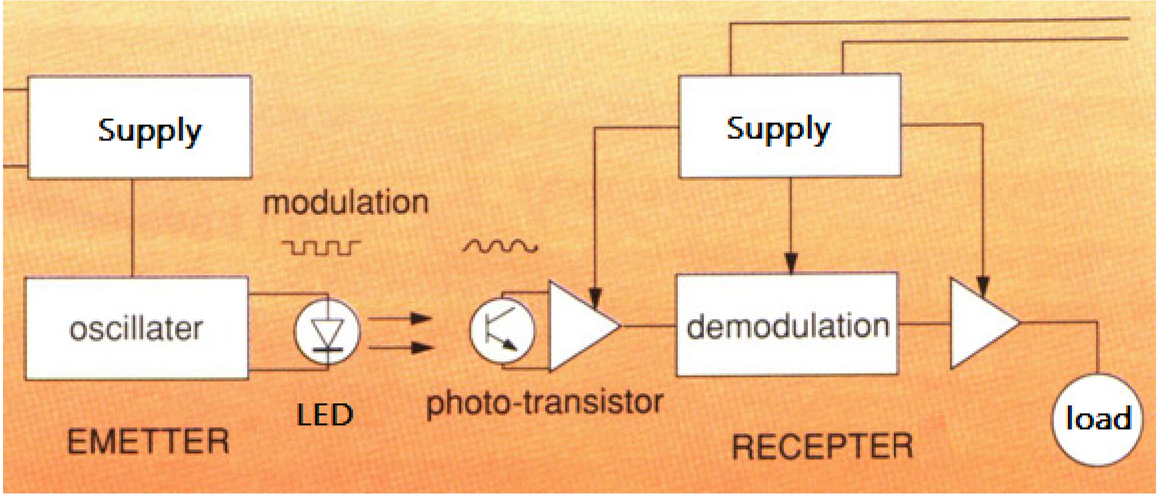


Figure II.8: Principle of Signal Transmission and Reception via Optical Fiber

Principle of Transmission via Optical Fiber:

In detectors using optical fibers, the amplifier is remote, which allows for a very compact detector. The signal is transmitted from the amplifier to the detection area via optical fibers.

Principle of Signal Propagation:

The core and the cladding of the optical fiber have different refractive indices, with the refractive index n2 of the cladding being slightly lower than n1 of the core. If the angle of incidence of the light beam entering the fiber is below a certain limit, transmission from the core to the cladding is prevented, and the light beam remains confined within the core, propagating through reflection. The propagation conditions depend on the material used.

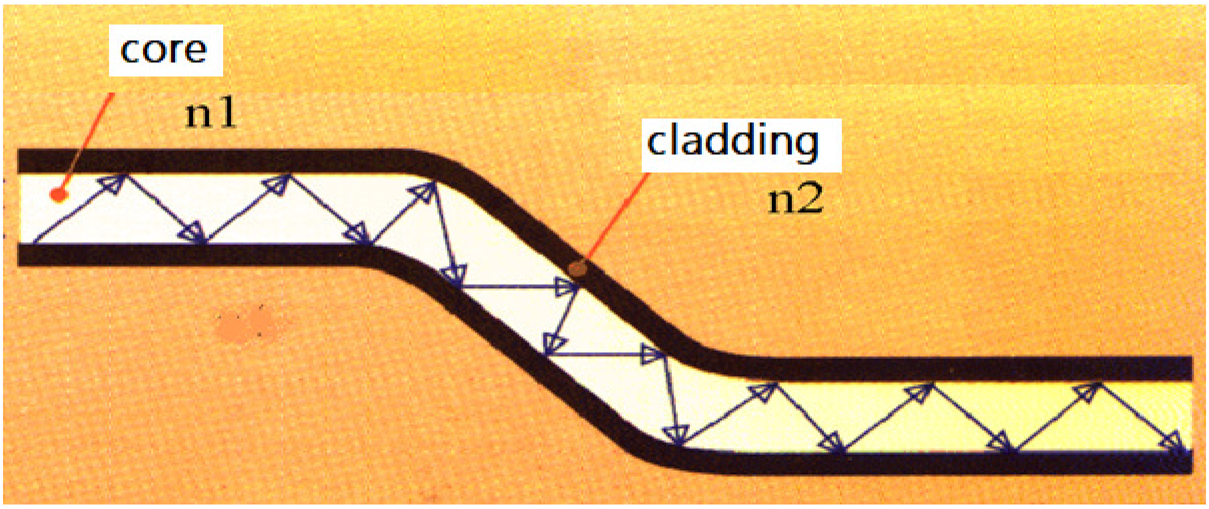


Figure II.8: Principle of Signal Propagation in Optical Fiber

Detector Output Stage

The information provided by a detector can be given in the form of an electrical contact or as a static output. In the latter case, there are two main families: 2-wire detectors, which are connected in series with the load, and 3-wire detectors.

2-Wire Detectors: These devices are powered in series with the load to be controlled. Consequently, they are subject to:

- Residual current (in the open state)
- Voltage drop (in the closed state)

Advantages:

- They can be connected in series like mechanical position switches.
- For certain series, they can be connected to either positive (PNP) or negative (NPN) logic inputs without risk of wiring errors.

Disadvantages:

- It is important to verify the possible influence of residual current and voltage drop on the controlled input device (engagement and disengagement thresholds).

3-Wire Detectors: These devices have two wires for DC power and one wire for transmitting the output signal.

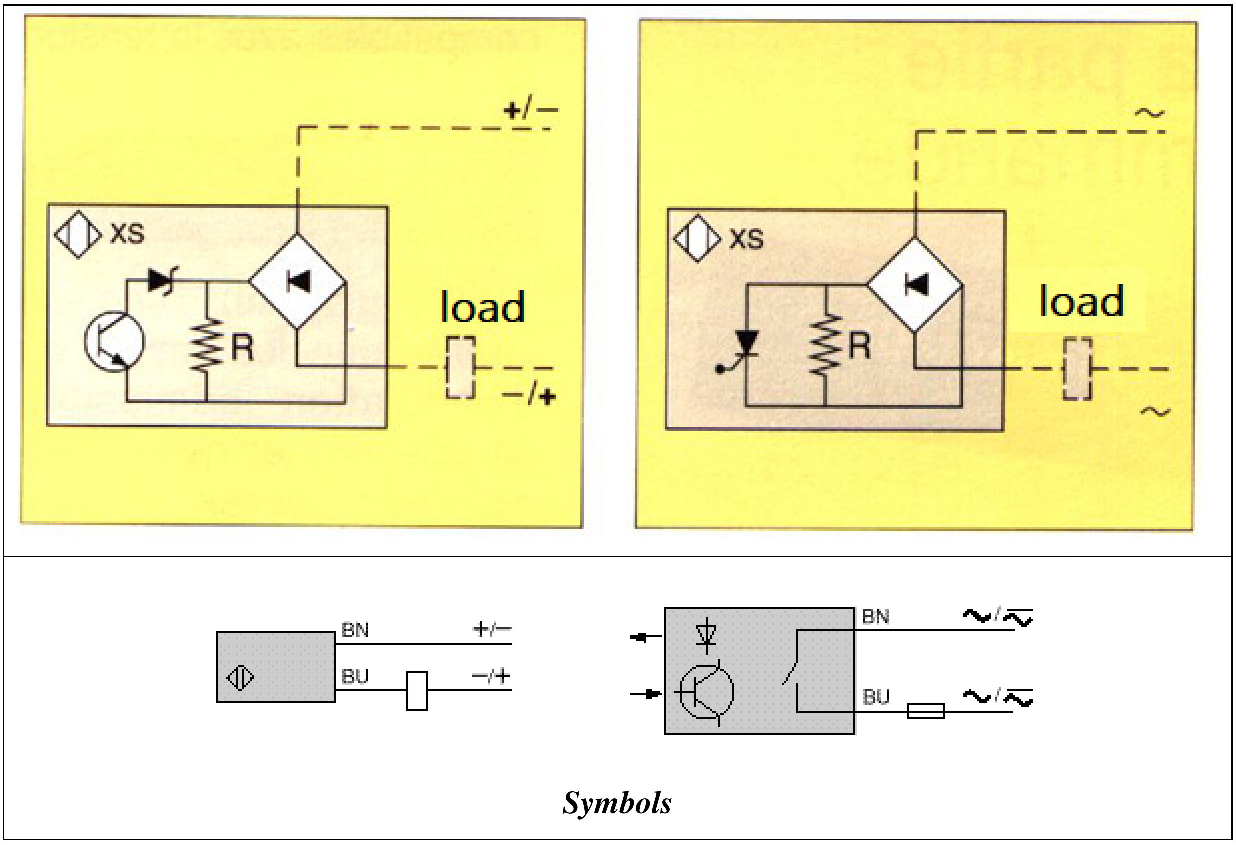
- **PNP type:** Switching to the positive potential load.
- **NPN type:** Switching to the negative potential load.

Advantages:

- Adaptable output signal, no residual current, and low voltage drop.
- Programmable versions reduce the number of models in stock.

Disadvantages:

- For some models, it is necessary to use the device adapted to the logic of the input device, PNP or NPN.



Symbols

Figure II.9: Wiring Diagram of a 2-Wire Detector

However:

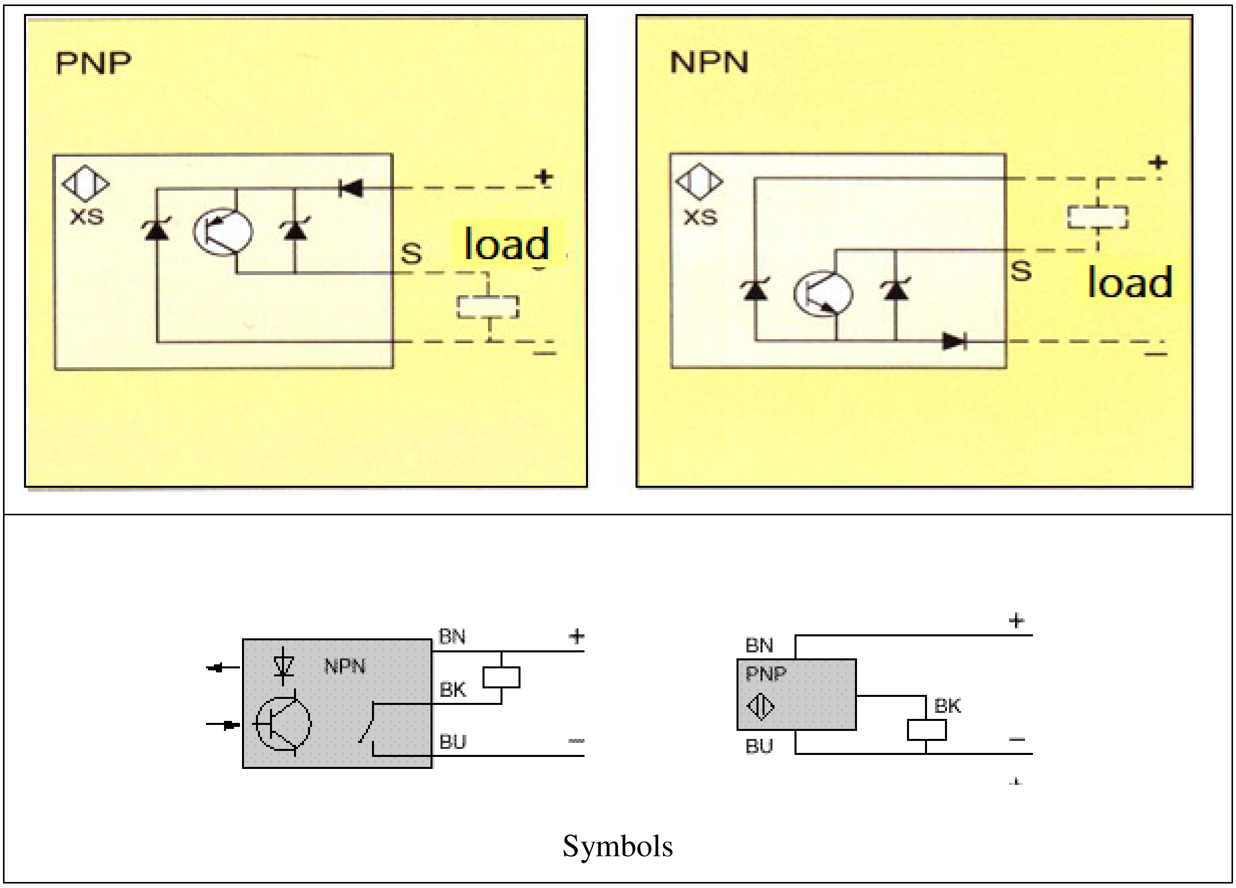
Verify the potential influence of residual current and voltage drop on the controlled input device (engagement and disengagement thresholds).

3-Wire Detectors

These devices include two wires for DC power supply and one wire for transmitting the output signal.

- PNP Type: Switching to the positive potential load.
- NPN Type: Switching to the negative potential load.

Programmable universal devices can perform the following functions: PNP/NO, PNP/NC, NPN/NO, NPN/NC.



Symbols

Figure II.10: Wiring Diagram of a 3-Wire Detector

Advantages and Disadvantages of 3-Wire Detectors

Advantages:

- Adaptability of the output signal: No residual current, low voltage drop.
- Programmable versions: Reduces the variety of models in stock.

Disadvantages:

- Compatibility: For some models, it is necessary to use the device suited to the input logic of the component (PNP or NPN).

Static Output Detectors

For static output detectors, there are two types:

- Clear function: The output is '1' logic when the beam is received by the receiver.
- Dark function: The output is '1' logic when the beam is not received by the receiver.

e) Detection of Cylinder Rod Positions with Reed Switches

Reed switches are used to detect the position of cylinder rods. These switches are coupled with cylinders whose pistons have a magnetic pellet. When the pellet is aligned with the switch, the two reeds touch, and the contact is established.

Construction of Reed Switches:

- Inert gas: The protective tube contains inert gas to prevent oxidation.
- Glass tube: The tube containing the reeds is made of glass.
- Magnetic reed: The two metal reeds touch under the magnetic attraction of the cylinder's pellet.

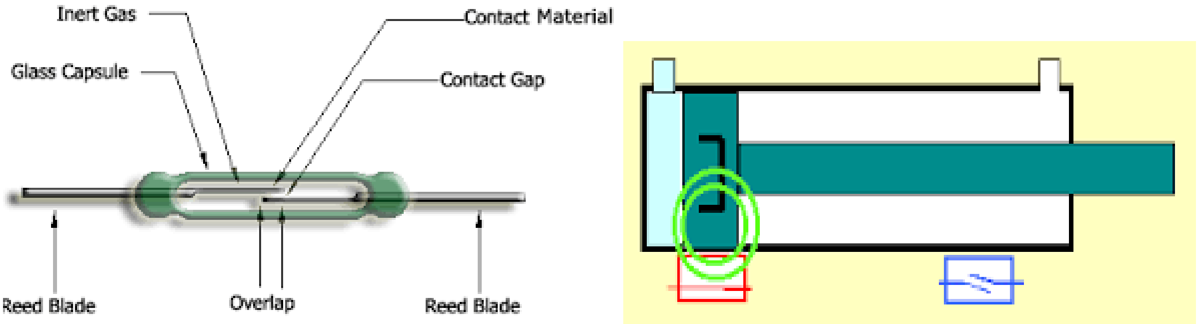


Figure II.11: Constitution and placement of reed switch on a jack

f) Incremental encoders

The periphery of the encoder disk is divided into "x" regularly spaced slots. A light beam is located behind these slots directed towards a photosensitive diode. Whenever the beam is interrupted, the sensor sends a signal that allows knowing the variation in the position of the shaft. To determine the direction of rotation of the encoder, a second light beam is used, which will be offset from the first. The first beam that sends its signal will also indicate the direction of rotation of the encoder.

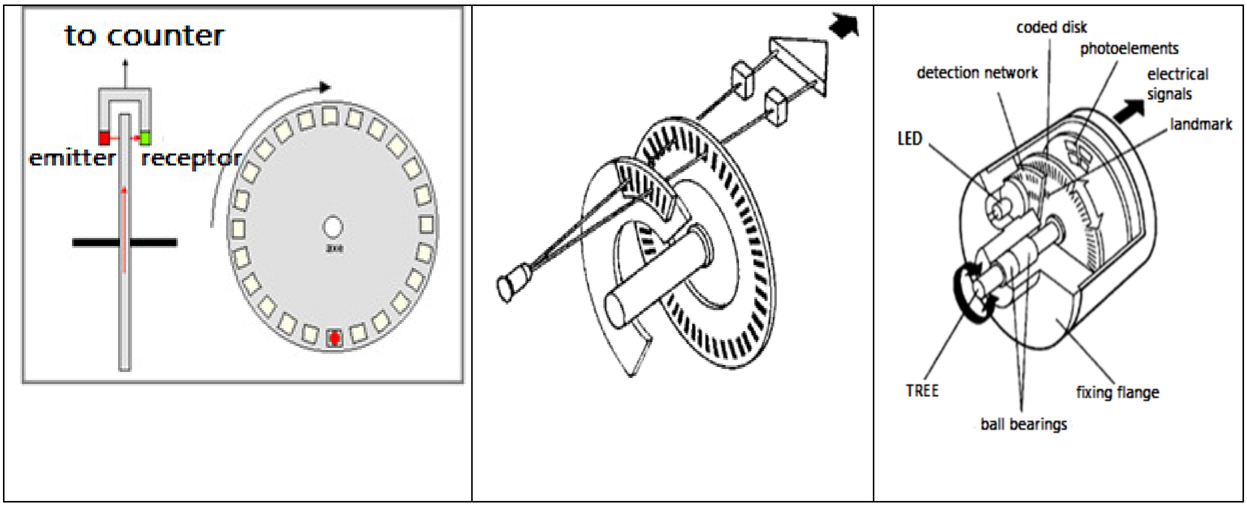


Figure II.11: An incremental encoder and its operating principle

This time, the disk has a large number of tracks, and each track is equipped with an emitting diode that emits a light beam and a photosensitive diode. The central track is the main track; it determines in which half-turn the reading is performed. The next track determines in which quarter-turn we are, the next one the eighth of a turn, and so on. The more tracks there are, the more precise the angular reading will be. There are single-turn absolute encoders that allow knowing a position within one turn and multi-turn absolute encoders that also provide information about the number of turns completed.

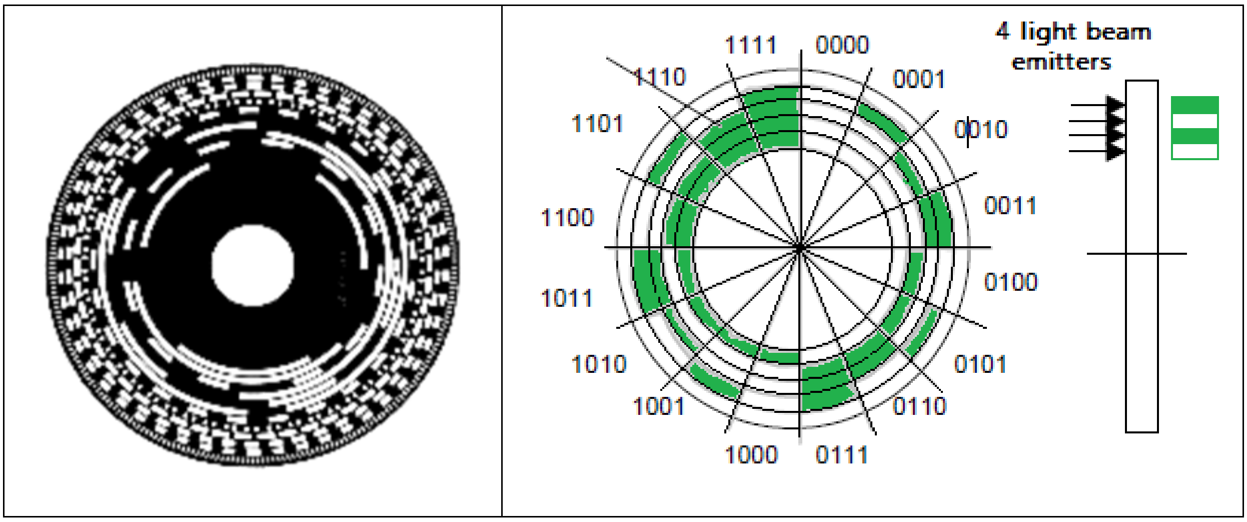


Figure II.12: Incremental encoder with 4 bits

g) Electrical strain gauges

Electrical strain gauges typically consist of a thin wire or foil, often made of materials like constantan or Karma, attached to a flexible backing material, such as paper or plastic. When subjected to mechanical strain or deformation, such as stretching or compression, the wire or foil experiences a change in resistance proportional to the applied strain.

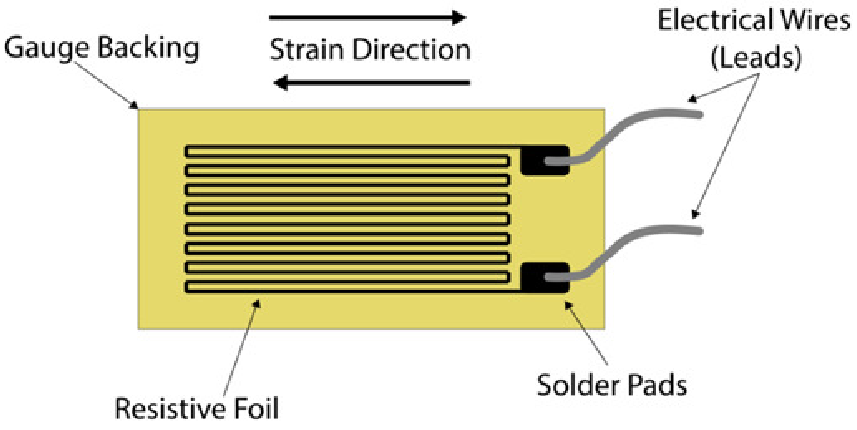


Figure II.13: Electrical strain gauges

This change in resistance is then measured using a Wheatstone bridge circuit or similar setup to quantify the strain on the material being monitored. Additionally, the gauge may be bonded or attached to the surface of the material under investigation using an adhesive or similar method to ensure accurate strain measurement.

Chapiter III :

PLCs Programmation

1. Introduction

It was in 1975 that an idea emerged, faced with the increasing complexity of logical automations, to create a tool that would allow the representation of a system's specifications, thus overcoming the drawbacks of various existing methods (primarily their heaviness). This reflection was conducted within the AFCET (French Association for Economic and Technical Cybernetics) between academics and industry professionals, and in 1977, an initial report on the GRAFCET tool was published.

GRAFCET: **F**unctional **G**raph of **C**ommand **S**tep **T**ransition. The uppercase writing. GRAFCET corresponds to the GRAFCET model, and lowercase "grafcet" corresponds to the graphical result of a study on a system's behavior.

Since the GRAFCET model has been standardized (an international standard), it has been expanded to be directly implemented in the control part of a system (programmed GRAFCET), using various languages specific to programmable logic controllers.

During the course of a project, different GRAFCETs will be developed, more or less detailed, depending on the progress of the study. They will allow description from various perspectives: a global view of the automated system, the description of the system's operation from an operative part viewpoint, or even the description of the expected operation of an identified control part.

Programming an API involves translating the operating equations of the system to be automated into the specific language of the controller.

2. Programming Languages of PLCs

PLCs (Programmable Logic Controllers) are programmed using specialized languages that allow engineers and technicians to create control logic for industrial automation processes. The most commonly used PLC programming languages are standardized by the **IEC 61131-3** standard, which defines five primary programming languages:

2.1. Instruction List Language (IL):

This low-level textual language is an instruction per line language. It resembles, in some aspects, the assembly language used for programming microprocessors. It is defined by the IEC 61131-3 standard.

Instruction List (IL) is an assembly-like programming language conforming to IEC 61131-3.

This language supports accumulator-based programming. IEC 61131-3 operators are supported, as well as multiple inputs/multiple outputs, negations, comments, definition/redefinition of outputs, and conditional/unconditional jumps. Each instruction starts with loading values into the accumulator using the LD operator. The operation is then performed with the first parameter extracted from the accumulator. The result of the operation is available in the accumulator, from which you must store it with the ST instruction.

To enable programming loops or conditional executions, IL supports comparison operators (such as EQ, GT, LT, GE, LE, and NE) and jumps. These can be unconditional (JMP) or conditional (JMPC/JMPCN). For conditional jumps, the value of the accumulator is referenced for TRUE or FALSE.

Syntax:

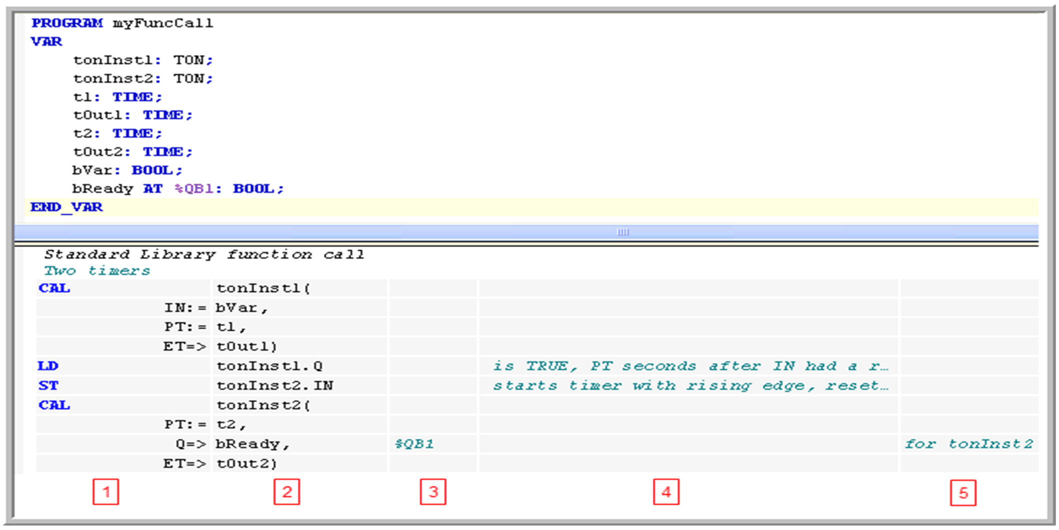
An Instruction List (IL) consists of a series of instructions. Each instruction starts on a new line and contains an operator and, depending on the type of operation, one or more operands separated by commas. You can extend the operator with a modifier.

On a line preceding an instruction, there may be an identification mark (label) followed by a colon (:) (e.g., "ml:" in the example below). A label can be the target of a jump instruction (e.g., "JMPC m1" in the example below).

Program Example:

```
LD            BVar1
ST            tonInst1.IN
CAL           tonInst1(
          PT:=t1,
          ET=>tOut2)
LD            toninst1.Q
JMPC          mark1
ST            tonInst2.IN
markl:
LD            iVar2
ADD           230
```

IL editor Structure :



```
PROGRAM myFuncCall
VAR
    tonInst1: TON;
    tonInst2: TON;
    t1: TIME;
    tOut1: TIME;
    t2: TIME;
    tOut2: TIME;
    bVar: BOOL;
    bReady AT %QB1: BOOL;
END_VAR
```

```
 Standard Library function call
 Two timers
CAL              tonInst1(
          IN:=   bVar,
          PT:=   t1,
          ET=>   tOut1)
LD               tonInst1.Q                    is TRUE, PT seconds after IN had a r…
ST               tonInst2.IN                   starts timer with rising edge, reset…
CAL              tonInst2(
          PT:=   t2,
           Q=>   bReady,        %QB1                                                  for tonInst2
          ET=>   tOut2)
```

Figure III.1: IL table editor

Each program line is written in a table row, structured into fields by the table columns:

Coloumn	Content	Description
1	operator	This field contains the IL operator (LD, ST, CAL, AND, OR, etc.) or a function name. In case of a function block call, the corresponding parameters are also specified here. Enter the prefix field := or =>.
2	operande	This field contains exactly one operand or a jump label. If multiple operands are required (multiple/extensible operators AND A, B, C, or function calls with multiple parameters), write them on the following lines and leave the operator field empty. In this case, add a comma to separate the parameters. In case of a function block, program, or action call, add the appropriate opening and/or closing parentheses.
3	address	This field contains the address of the operand defined in the declaration part. You cannot modify this field. Use the "Show symbol address" option to enable or disable it.
4	symbol comment	This field contains the comment defined for the operand in the declaration part. You cannot modify this field. Use the "Show symbol address" option to enable or disable it.
5	operand comment	This field contains the comment for the current line. It is editable and can be enabled or disabled via the "Show operand comment" option.

2.2. Language: Structured Text (ST)

This high-level textual language is an advanced language that allows programming of variously complex algorithms.

Structured Text (ST) is one of the five languages of the IEC 61131-3 standard. ST is a higher-level textual programming language, akin to PASCAL or C. The code of programs consists of expressions and instructions. Unlike IL (Instruction List), it allows the use of multiple constructs for programming loops, facilitating the development of complex algorithms.

Various complex instructions are supported, for example:

- Iteration loops (REPEAT-UNTIL; WHILE-DO)
- Conditions (IF-THEN-ELSE; CASE)
- Functions (SQRT(), SIN())

Example program :

```
IF value < 7 THEN
 WHILE value < 8 DO
  value:=value+1;
 END_WHILE;
END_IF;
```

2.3. Ladder Diagram (LD):

Ladder Diagram (LD) is a graphical programming language primarily dedicated to programming Boolean equations (true or false).

Ladder Diagram (LD) is a graphically-oriented programming language that resembles the structure of an electrical circuit.

LD is suitable for constructing logical switches but also allows for the creation of networks like in FBD.

Ladder Diagram (LD) consists of a series of networks, each of which is bounded by a vertical power line (power rail) on the left. A network contains a circuit diagram consisting of contacts and coils.

On the left side is a contact or a series of contacts passing from left to right, representing the ON or OFF condition corresponding to the Boolean values TRUE and FALSE. A Boolean variable is assigned to each contact. If this variable is TRUE, the condition will be passed from left to right along the connection line. Otherwise, the condition will be passed OFF. Thus, the coils placed on the right side of the network receive an ON or OFF condition from the left. Consequently, the TRUE or FALSE value will be written into an assigned Boolean variable.

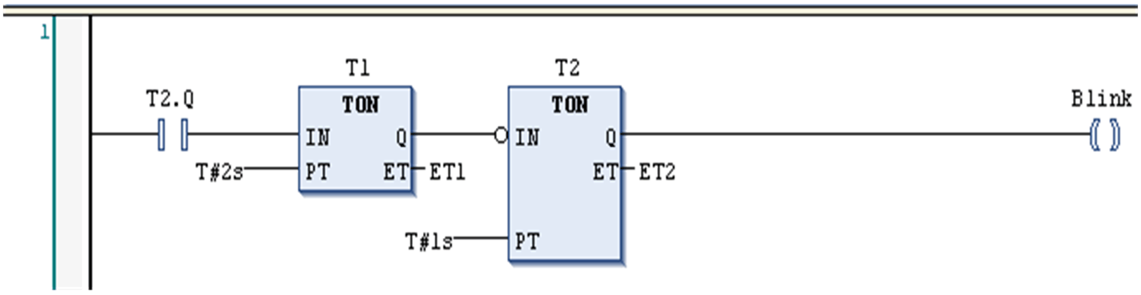


Figure III.2 : Ladder Diagram

2.4. Sequential Function Chart Language (SFC) :

Sequential Function Chart (SFC) is a high-level language derived from GRAFCET, allowing easy programming of all sequential processes.

SFC is a graphical language that describes the chronological order of actions within a program. These actions are available as distinct programming objects, written in any available programming language. In SFC, they are assigned to step elements, and the processing sequence is controlled by transition elements.

Example of step sequence in an SFC module:

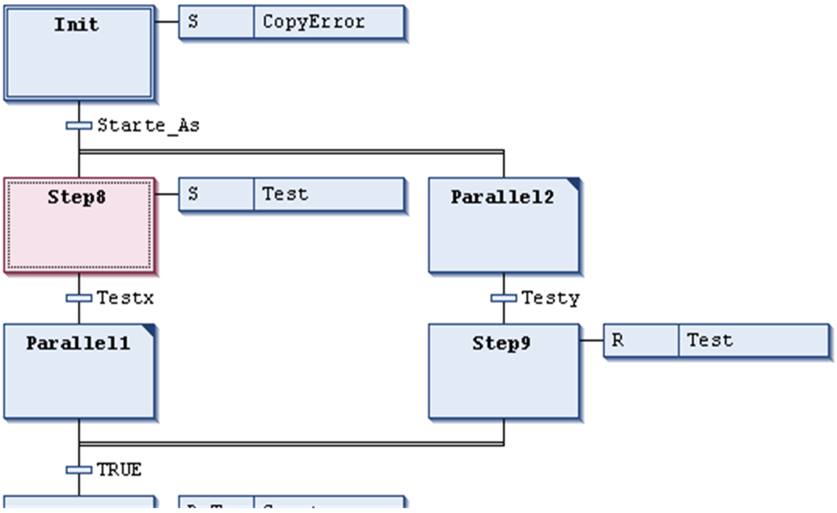


Figure III.3 : Sequential Function Chart Language

2.5. Function Block Diagram (FBD) Language :

The Ladder Diagram language is based on symbolism very close that used in traditional wiring diagrams.

Function Block Diagram (FBD) is a graphically-oriented programming language. It operates with a list of networks. Each network contains a graphical structure of boxes and connection lines representing a logical or arithmetic expression, a function block call, a jump, or a return instruction.

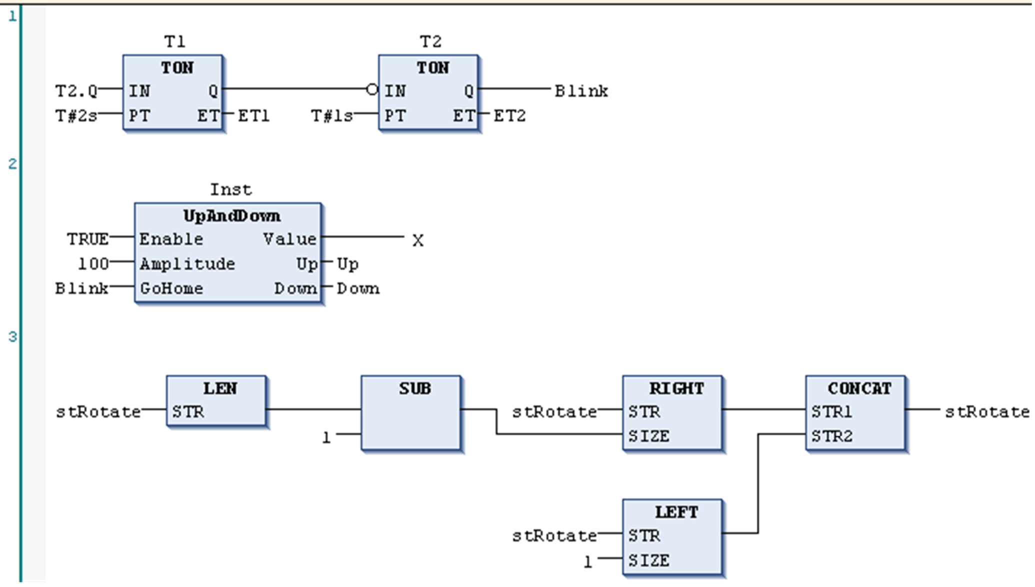


Figure III.4 : Function Block Diagram Language

3. Design of a Sequential Automation System:

In the design of a sequential automated system, several steps must be taken to develop the functional diagram of an automated system from the initial specification document.

3.1. Specifications Document (Standard X 50-151)

The specifications document for an automation system is a document that governs the relationships between the designer of the control system and the user of that system. The clauses of this document may contain considerations:

- Legal,
- Commercial,
- Technical. It also includes:
- A clear description of the functions and performance requirements of the automation system to be designed.
- Specifications for the behavior of the "control" part with respect to the "operative" part.

3.2. Specification Levels:

3.2.1. Functional Specifications:

These specifications describe, in terms of function, the behaviors that the control part must exhibit:

- In response to information from the operative part, the participants, or other control parts.
- They describe the operation of the "operative" part of the sequential automation;
- They also determine the reactions of the "control" part based on the information received from the "operative" part. These specifications must therefore clearly and precisely define the various functions, information, and commands associated with the

automation of the "control" part, without considering the technologies that will be applied or used.

3.2.2. Technological Specifications:

These specifications describe, in terms of means, i.e., technological solutions, the operation of the control part. Technological specifications specify how the automation must be incorporated into the system consisting of the automated system and its environment. These specifications provide additional details to the functional specifications to enable the design of an automation system that truly controls the "operative" part:

- Information about the exact nature of the sensors, pre-actuators, and actuators used,
- Their characteristics and the constraints that may result from them only come into play at this stage. In addition to interface specifications, such as pre-actuators, environmental specifications such as temperature, humidity, supply pressure, etc., can also be added. At this stage of automation design, we move from general data on its operation to specific specifications. These specifications accurately determine the actions to be taken at each stage of the automation, taking into account the technological tools, their limitations, and constraints.

3.2.3. Operational Specifications

They describe:

- The behaviors that the control part and the automated system must possess in the production context,
- The overall performance of the automated system, maintenance, operating modes, and stops...

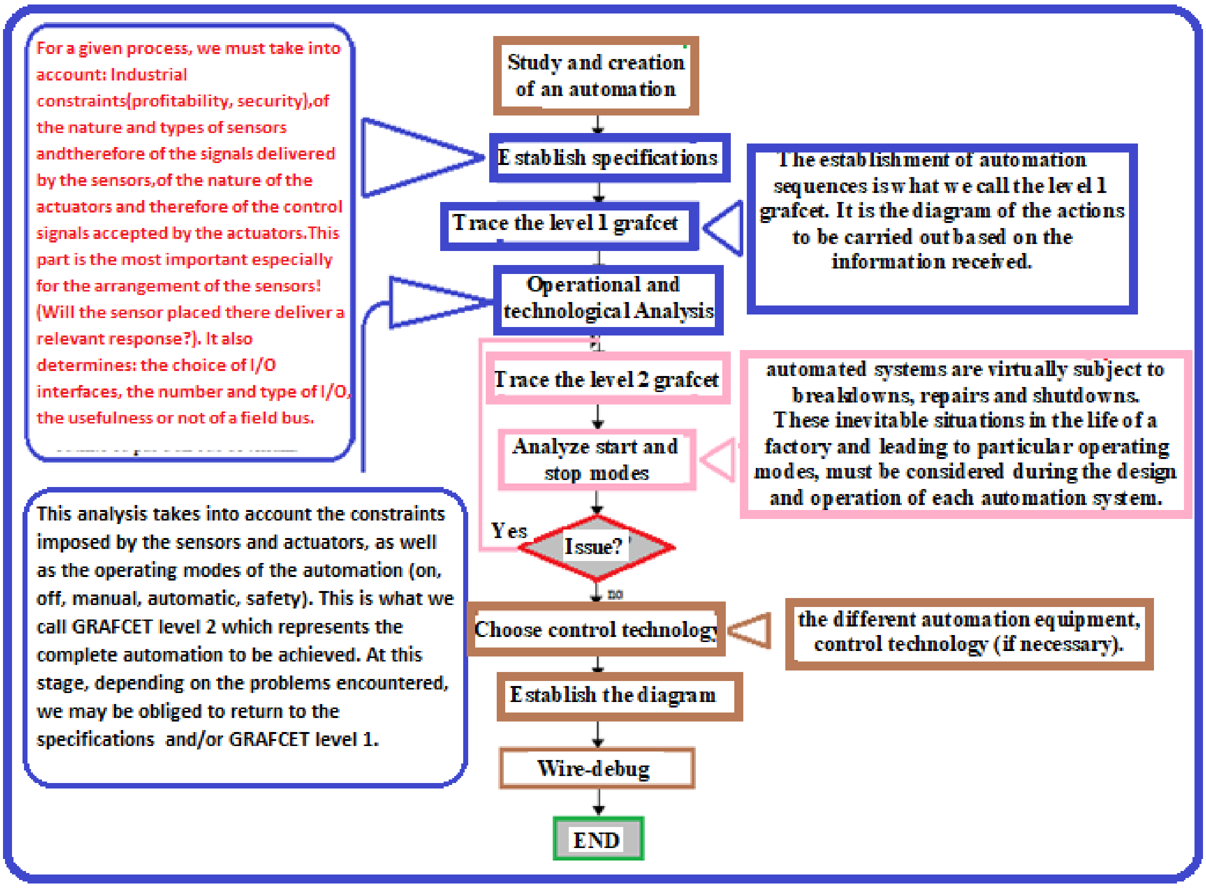

For a given process, we must take into account: Industrial constraints(profitability, security),of the nature and types of sensors andtherefore of the signals delivered by the sensors,of the nature of the actuators and therefore of the control signals accepted by the actuators.This part is the most important especially for the arrangement of the sensors! (Will the sensor placed there deliver a relevant response?). It also determines: the choice of I/O interfaces, the number and type of I/O, the usefulness or not of a field bus.
Study and creation of an automation
Establish specifications
Trace the level 1 grafcet
The establishment of automation sequences is what we call the level 1 grafcet. It is the diagram of the actions to be carried out based on the information received.
Operational and technological Analysis
Trace the level 2 grafcet
automated systems are virtually subject to breakdowns, repairs and shutdowns. These inevitable situations in the life of a factory and leading to particular operating modes, must be considered during the design and operation of each automation system.
Analyze start and stop modes
Yes
Issue?
no
This analysis takes into account the constraints imposed by the sensors and actuators, as well as the operating modes of the automation (on, off, manual, automatic, safety). This is what we call GRAFCET level 2 which represents the complete automation to be achieved. At this stage, depending on the problems encountered, we may be obliged to return to the specifications and/or GRAFCET level 1.
Choose control technology
the different automation equipment, control technology (if necessary).
Establish the diagram
Wire-debug
END

Figure III.5: Design Flowchart of an Automated System

3.3. Development of the GRAFCET:

The GRAFCET model: It consists of a set of graphical elements (with syntax), an interpretation, and evolution rules.

Graphical Elements:

- **STEPS**
- **TRANSITIONS**
- **ORIENTED CONNECTIONS**

The alternating step/transition network forms the backbone of the GRAFCET.

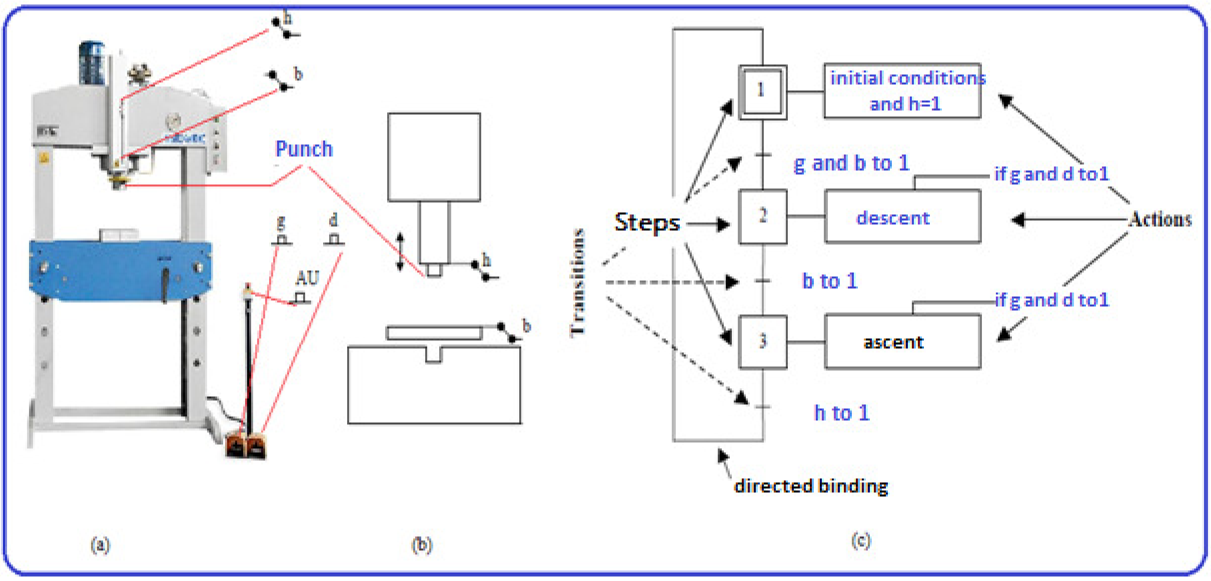


Figure III.6: Example of a Stamping Press , (a) Hydraulic Press, (b) Functional Diagram of the Press, (c) GRAFCET Associated with the Operation of the Press

A hydraulic press is a machine that uses a hydraulic cylinder to generate a compressive force. It is commonly used in manufacturing processes to shape or mold metal. The functional diagram illustrates the operational stages of the press, highlighting the flow from the initial state to the end state.

The GRAFCET diagram provides a visual representation of the sequential steps and transitions in the press operation.

Operation Description

In the resting state, the punch is in the upper position (h = 1) as the initial condition. When both g and d are actuated simultaneously, the punch descends to position b, then ascends back to h, the resting state.

Note: This GRAFCET is incomplete; it lacks the emergency stop and manual control. It is important to note that:

- Depending on the sensors, the transition conditions change (open or closed at rest).
- Depending on the actuators, the actions are different.
- Operator safety is ensured by an action if the condition g and d = 1 is true.

4. GRAFCET Elements:

- **STEP:** A situation where the behavior of the control part is invariant concerning its inputs and outputs. A step is represented by a square. Step i, numerically identified, has a state variable called step variable Xi. This variable is a boolean, being 1 if the step is active, 0 otherwise. A step is either ACTIVE or INACTIVE. A dot inside the square is sometimes used for studying the dynamic behavior of the system when the step is active.

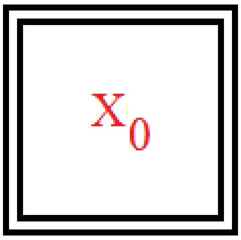
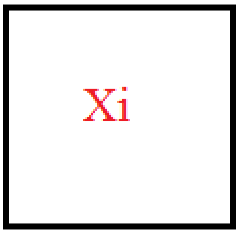
Example Steps and Transitions:

1. **Step 1 (Initial State):** Punch is at the upper position (h = 1).
    - **Transition 1:** Condition g and d are both 1.
2. **Step 2:** Punch descends to position b.
    - **Transition 2:** Punch reaches position b.
3. **Step 3:** Punch ascends back to the upper position (h).

Remarks

- Emergency stop and manual control should be integrated into the GRAFCET for a complete operational diagram.
- Safety measures must ensure that the operator is protected by validating conditions like g and d both being 1 before activating certain steps.
- Each step and transition must be clearly defined to ensure accurate and safe operation of the press.

This setup ensures that the press operates in a controlled and sequential manner, maintaining safety and efficiency throughout the process.

 	**Initial Step:** It represents the system in its initial resting state. It is activated at the beginning of the cycle.
 	**Step:** Each step is associated with one or more actions, meaning an order directed towards the operative part or other GRAFCETs.

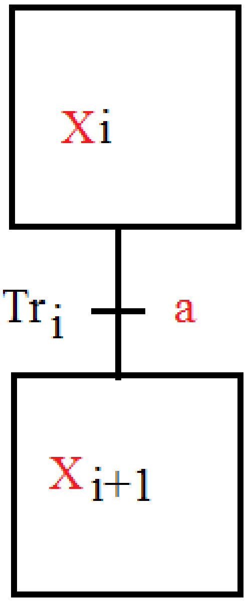
4.1. Step:

A binary variable *xixi* can be associated with each step *ii*, where the states "0" and "1" correspond respectively to the inactivity and activity of step *i*.

The step corresponding to the initialization of the system is called the initial step. It is represented by a double square. There can be multiple initial steps in the same GRAFCET.

4.2. Transitions:

Transitions indicate the possibility of evolving from one situation to another. The transition from one situation to the next occurs by crossing a transition from top to bottom. The evolution can occur between two or more steps. A transition is represented by a perpendicular bar on the connection line.

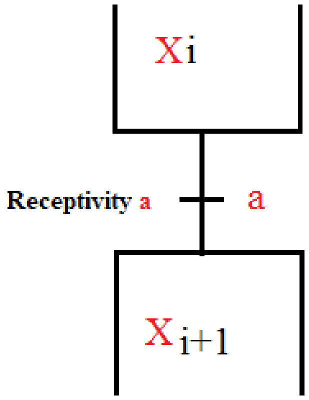
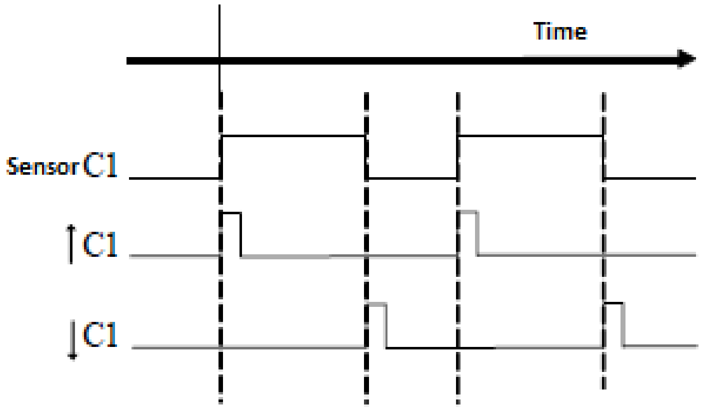
	A transition indicates the possibility of evolution that exists between two steps, and thus the succession of two activities in the operative part. When crossed, it allows the system to evolve. Each transition is associated with a receptivity a, which expresses the necessary condition for moving from one step to another.

4.3. Receptivity:

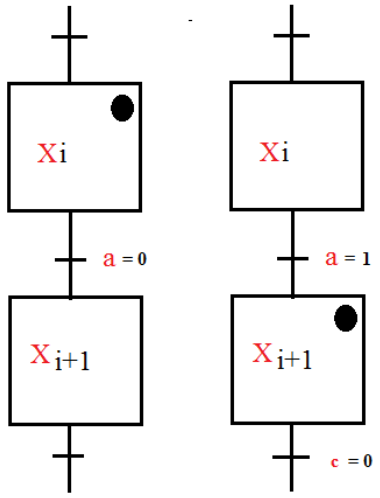
A Boolean equation associated with a transition. It is a logical function of the inputs, auxiliary variables, and/or the activity of steps. It distinguishes, among all the variables of the system, those that are likely to cause the command part to evolve by crossing a transition. This condition is written as a logical proposition, a combinatorial function calculated from:

- Input variables reflecting the state of sensors, push buttons, etc.
- Time
- The current state of the GRAFCET steps (the $\boldsymbol{Xi}$).

If the receptivity is not specified, it means it is always true.

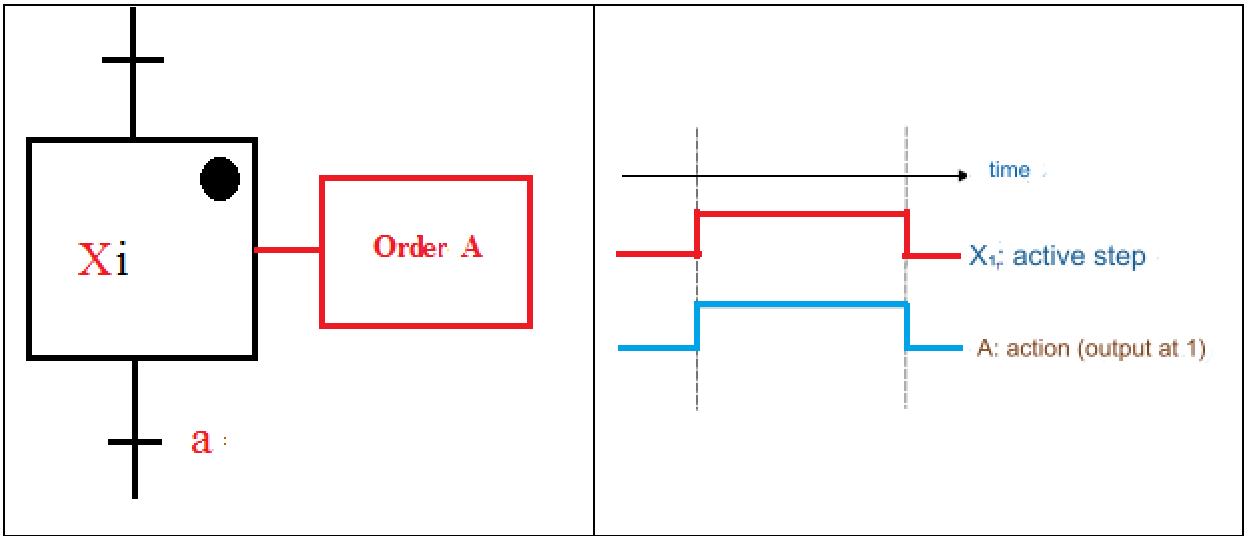
Rule: If step i is inactive (Xi=0), transition Tij has no effect. However, it's important to note that validating a transition without reason can have serious consequences, disrupting the cycle in some cases!

	If step i is active (Xi=1), transition a is validated, then: • If a=0, then wait. • If a=1, then step i is deactivated (Xi=0), and the next step i+1 is activated (Xi+1=1).

4.4. LIAISONS: They connect steps and transitions, and they are oriented. The general direction is from top to bottom if not indicated otherwise. Arrows should be used when needed, or when it facilitates readability. Sometimes diagonal lines can be used to clarify the GRAFCET.

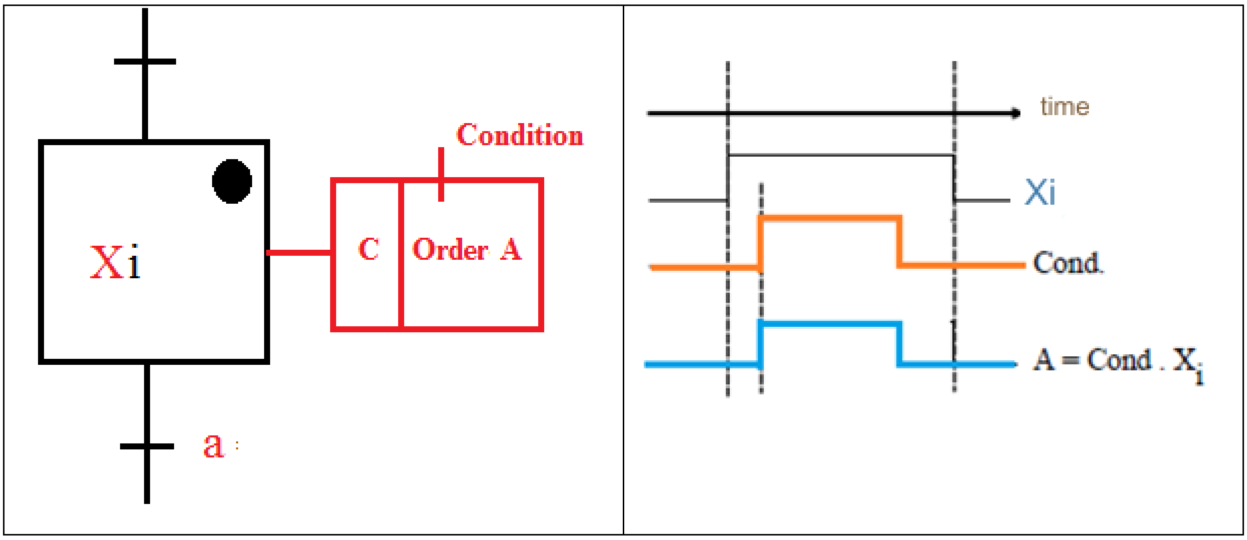
4.5. Actions: The action associated with the step can be of 3 types: continuous, conditional, or memorized.

4.5.1. Continuous Actions

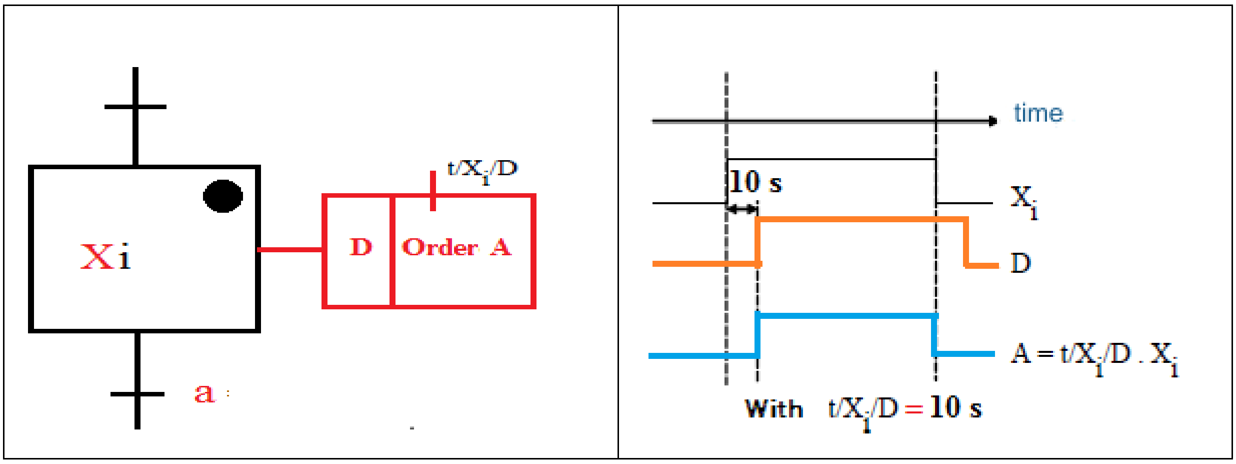


4.5.2. **A conditional action** is executed only if the associated step is active and if the associated condition is true. They can be decomposed into 3 particular cases:

a) Simple conditional action**: Type C**

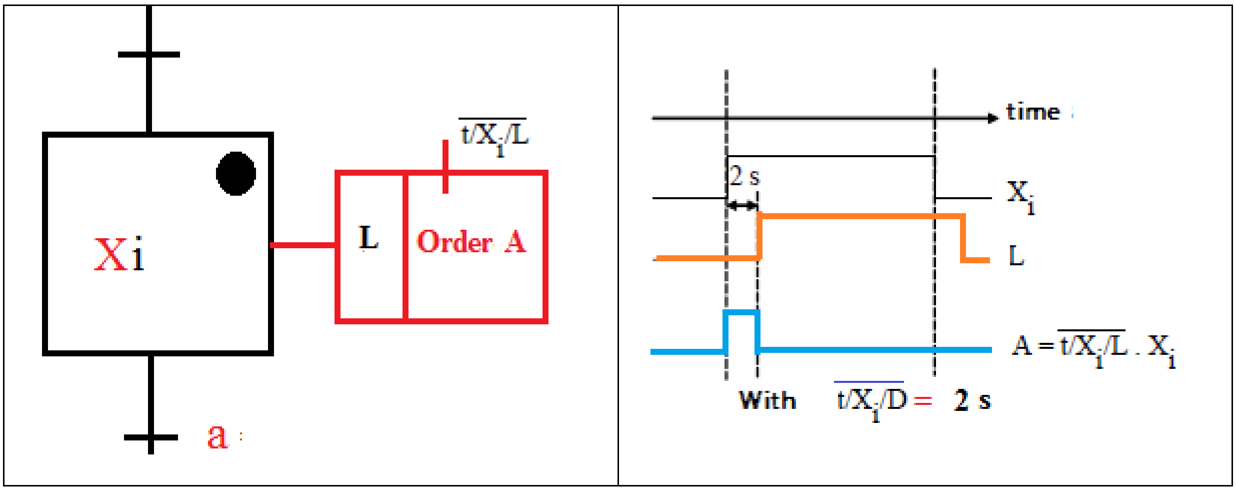


b) Simple conditional action: Type D



D is the duration of the delay, i.e., the time taken to switch to 1 after Xi switches to 1.

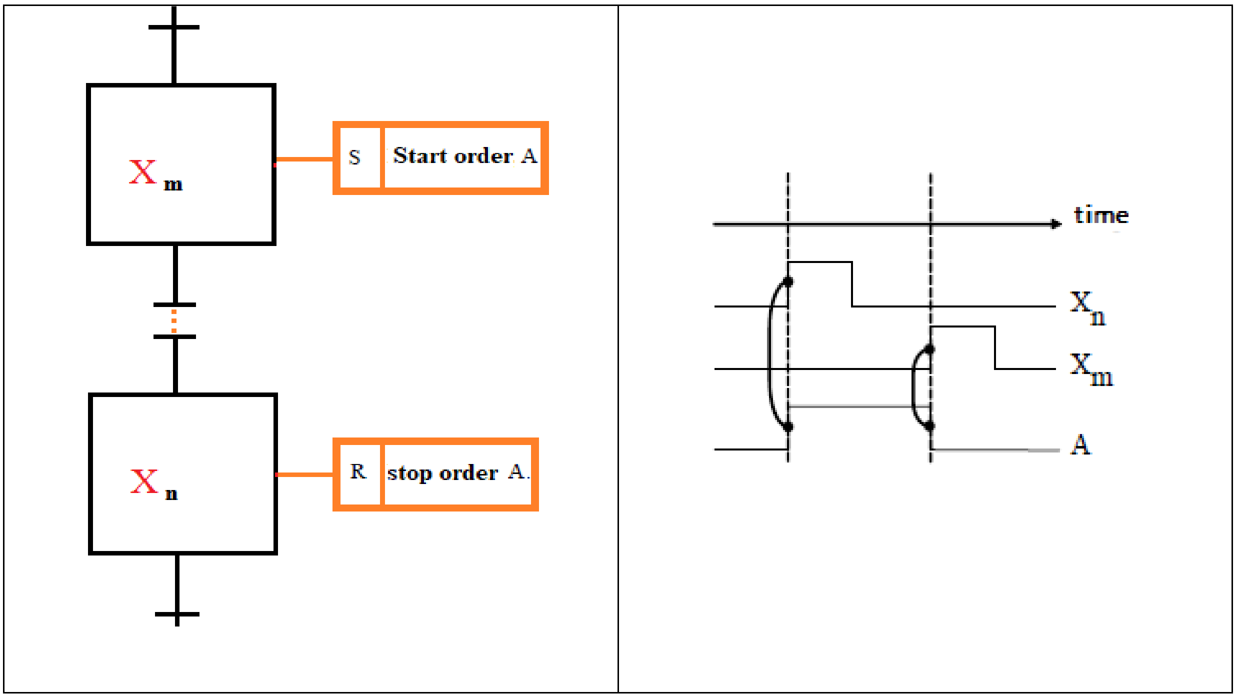
c) Simple conditional action: Type L



Xi
a
L
Order A
$\overline{t/X_i/L}$
time
2 s
X_i
L
$A = \overline{t/X_i/L} . X_i$
With $\overline{t/X_i/D} =$ 2 s

Action limited in time: Type L

4.6. Memorized actions:



4.7. Timers: Timers used in GRAFCET refer to step variables (Xn). A timer variable Sn is a Boolean variable whose evaluation mode takes time into account. It can be generally written as $Sn=t1/Xn/t2$ where:

- En denotes the input variable.
- $t1$ denotes the delay applied to the change from logical state 0 to 1 of input variable En. If $t1$=0, then the timer is written as $Sn=Xn/t2$.
- $t2$ denotes the delay applied to the change from logical state 1 to 0 of input variable En. If $t2$=0, then the timer is written as $Sn=t1/Xn$.

Syntax Rule: The alternation of STEP-TRANSITION and TRANSITION-STEP must always be respected regardless of the sequence followed:

- Two steps should never be directly connected; they should be separated by a transition.

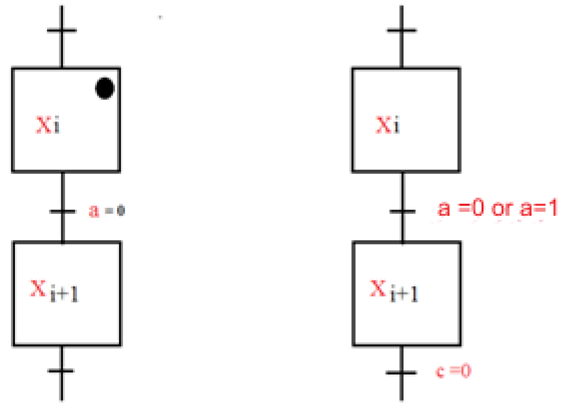
- Two transitions should never be directly connected; they should be separated by a step.

The rule may seem obvious, but errors are often made.

The Five Evolution Rules:

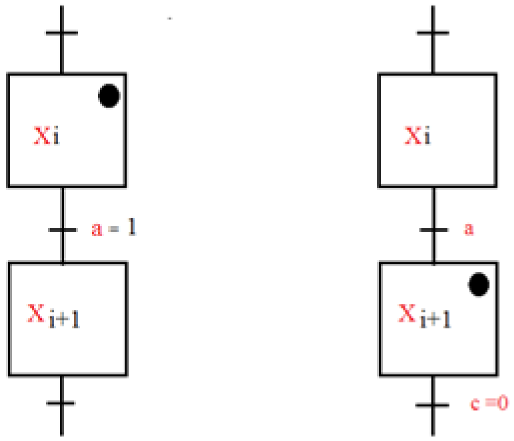
Rule No. 1 - Initial Condition: At the initial moment, only the initial steps are active. Rule

Rule No. 2 - Crossing a Transition: For a transition to be validated, all its upstream steps (immediately preceding steps connected to this transition) must be active. The crossing of a transition occurs when the transition is validated, AND only if the associated receptivity is true.



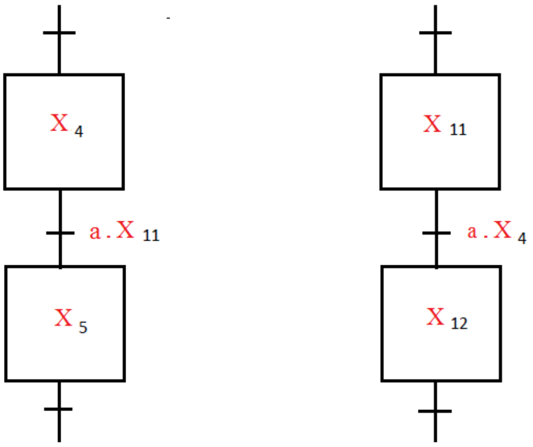
Transition validated Transition not validated

Rule No. 3 - Evolution of Active Steps: The crossing of a transition necessarily leads, in this order, to the deactivation of all its upstream steps and the activation of its downstream steps.



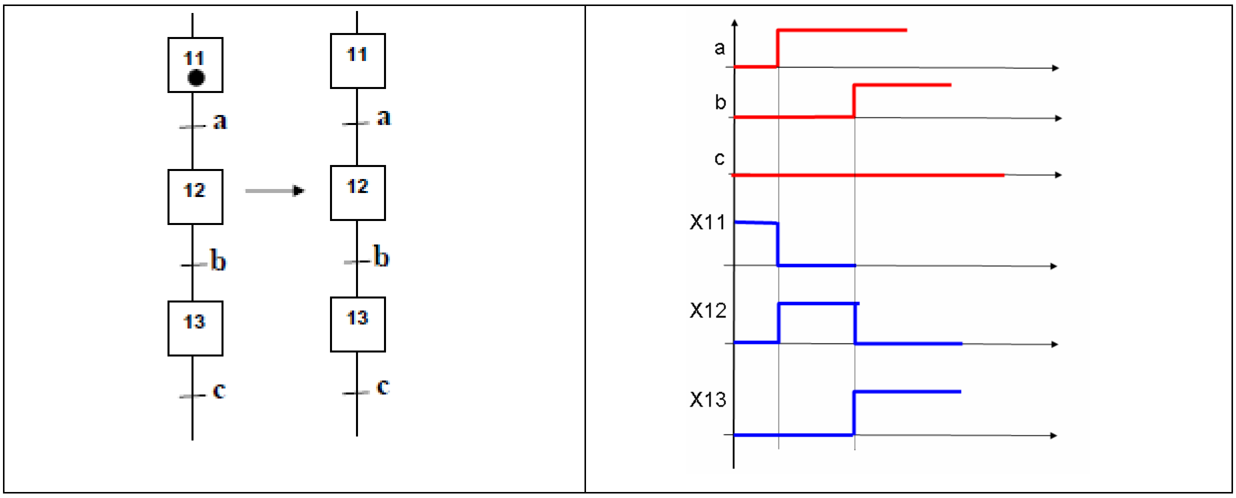
Passable Crossed

Rule No. 4 - Simultaneous Crossing: All transitions that can be crossed simultaneously at a given moment are simultaneously crossed.

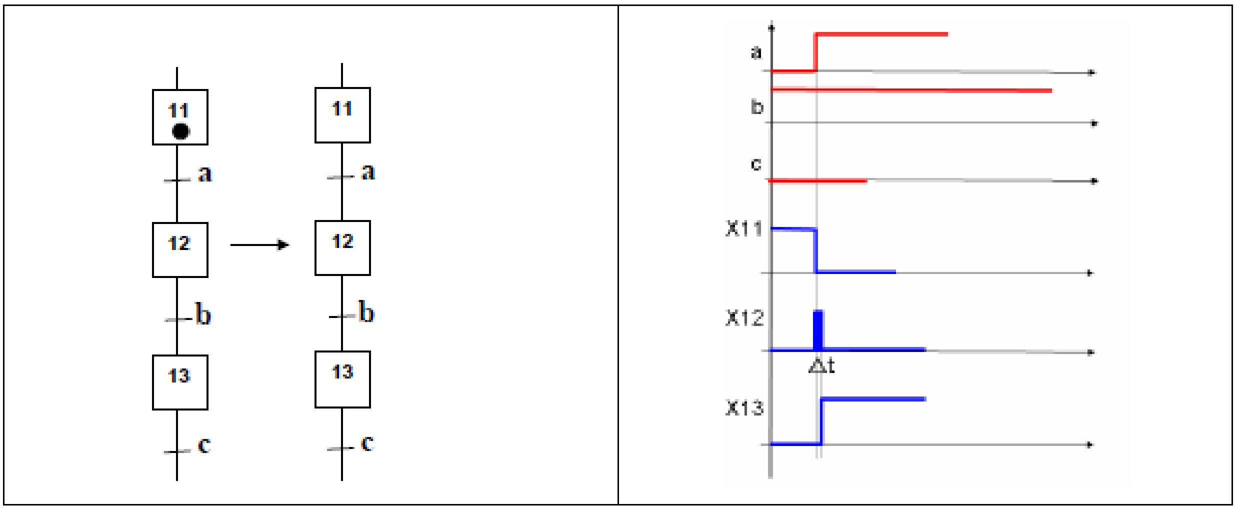


Rule No. 5 - Activation Conflict: If a step must be simultaneously deactivated by the crossing of a downstream transition and activated by the crossing of an upstream transition, then it remains active. This avoids unwanted transient commands (which can be harmful to the process).

5. **Transient and Non-Transient Evolution:** This is the general case where the input event triggers only one step of evolution (simultaneous crossing of one or more transitions). The resulting state is stable.



# **Transient Evolution:** In certain cases, applying the evolution rules may lead to successively crossing transitions (in multiple steps of evolution) if the receptivities associated with subsequent transitions are already true during the crossing.



The corresponding evolution, known as transient, involves a succession of unstable states during which the unstable steps are not activated. It's said that they have been virtually activated and deactivated, just like the associated transitions have been virtually crossed. An important consequence: during a transient evolution, the actions associated with the virtually activated steps are not executed (the state assignment does not occur). Only the memorized actions will be considered.

6. General Structure of a GRAFCET

The general structure of a GRAFCET consists of **steps, transitions, actions, and conditions** arranged in a flowchart-like diagram. Here are the primary components that make up a GRAFCET:

This example of Description Using the GRAFCET Model Cycle of a Single Sequence

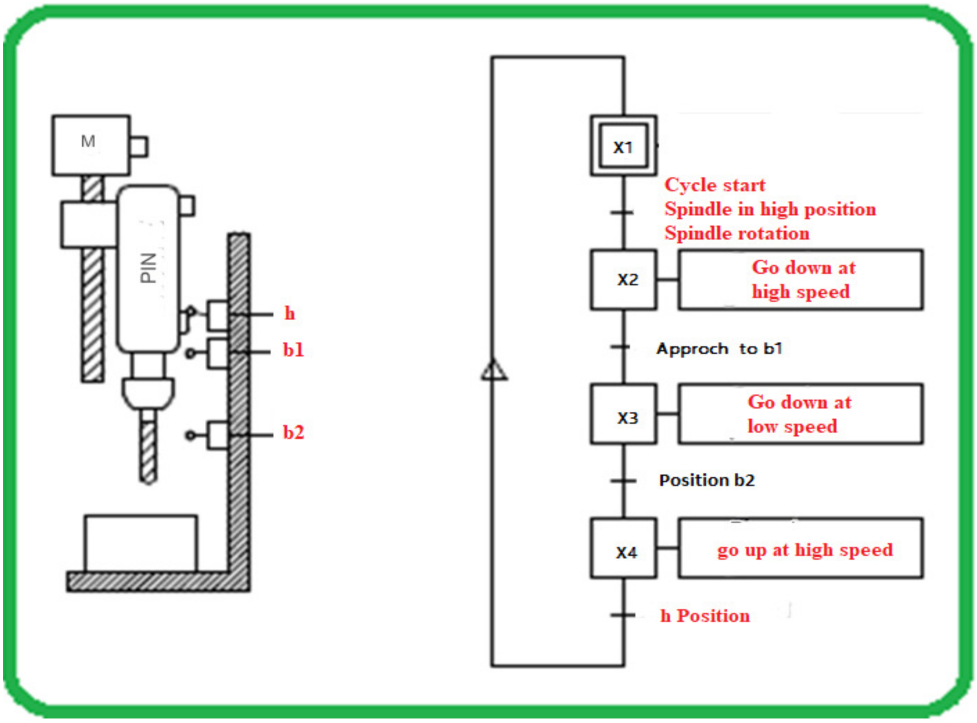


Figure III.9: Cycle of a Single Sequence for a Drill

Requirements for Drill Operation:

- The spindle runs continuously.
- The workpiece is fixed by the operator, who then gives the start cycle command.
- High-speed approach (h, b1).
- Slow-speed drilling (b1, b2).
- High-speed retraction.

7. Main Structures of a Grafcet

1. Step Jump and Sequence Resumption

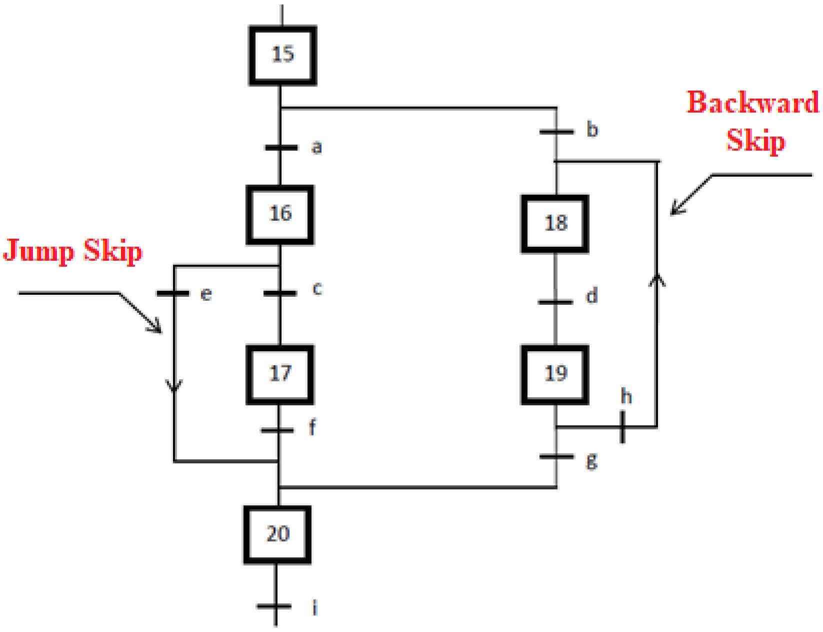
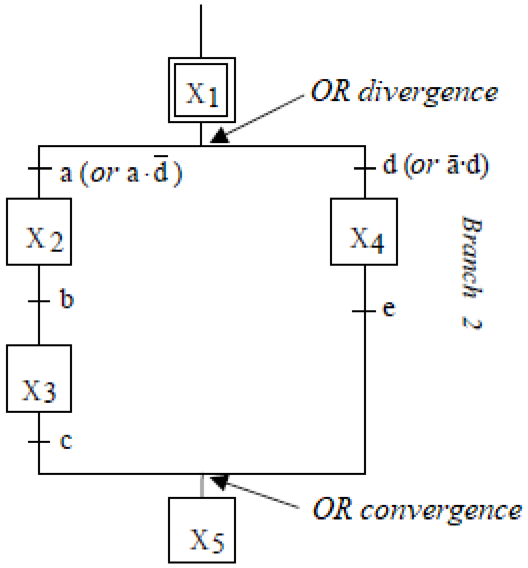


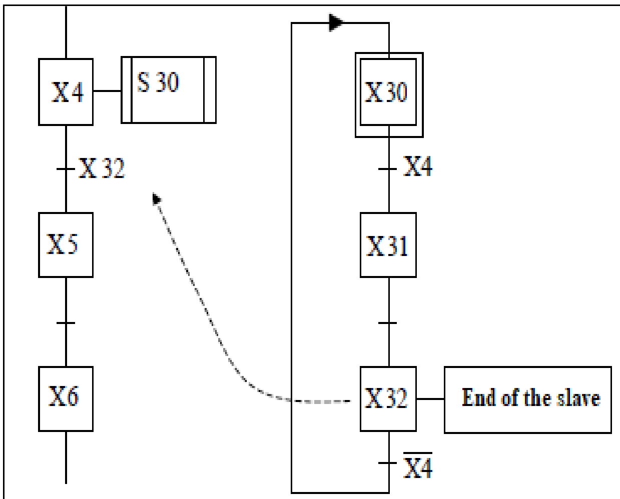
Figure III.10: The step-jump and sequence-resume structure

2. Switching between two or more sequences:

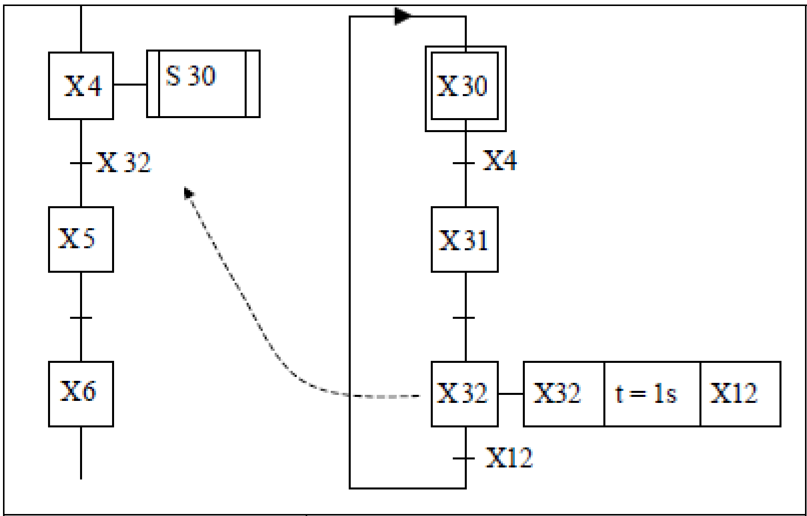
 X1
OR divergence
a (or a·d̄)
d (or ā·d)
X2
X4
Branch 2
b
e
X3
c
OR convergence
X5
	Let X1 be active and a = 0, d = 0 !
	wait until a or d becomes 1
	Branch 1:
	if a = 1 → X2
	if b = 1 → X3
	if c = 1 → X5
	Branch 2:
	if d = 1 → X4
	if e = 1 → X5
	Note: each branch of the divergence can be terminated by a jump, in which case there is no OR convergence.

Note: If a and d are both 1 simultaneously, we encounter a case of interpreted parallelism. Steps 2 and 4 will become active simultaneously, thereby causing the execution of actions in parallel, which is not desired by the designer. a and d must be exclusive conditions: one or the other will be true, but not both at the same time. We resolve this issue by setting, for example, a condition on Branch 1.

8. **Sub-GRAFCET** or repeated sequence: In an automated system, certain sequences can recur repeatedly in the cycle. To avoid repeating the same actions, it is possible to use a sub-program. This is written in the form of an independent GRAFCET, connected to the main GRAFCET.

<table>
<tr><td colspan="2"> </td><td rowspan="2">Initially, the initial steps of the two GRAFCETs are activated. In the master, the activation of step 4, X4 = 1, will trigger the execution of the slave GRAFCET by validating the first transition and causing the receptivity X32 to be triggered. In the slave, the activation of step 32, X32 = 1, causes the appearance of receptivity X32 in the master, which allows the execution of the main program by deactivating X4. This step 32 "hands control back to the master."</td></tr>
<tr><td>GRAFCET
Master</td><td>GRAFCET
Slave</td></tr>
</table>

It exists another method to hand control back to the master GRAFCET, which is actually the most commonly used one. It is illustrated below.



GRAFCET **Master**

GRAFCET **Slave**

We use a timer to keep the variable X32 at 1 for 1 second in order to hand control back to the master

Slave GRAFCETs are very useful for repeated sequences, emergency stop, and manual control. They allow controlling multiple automatisms on the same PLC.

9. Ladder Programming:

Ladder logic, or Ladder programming language, is a method for drawing electrical logic diagrams. It is a highly popular graphical language for programming industrial programmable logic controllers (PLCs). It was originally invented to describe relay logic. Its name is based on the observation that programs in this language resemble a ladder, with two vertical "rails" and a series of "rungs" between them. In Germany and elsewhere in Europe, the style involves placing horizontal rails, one at the top of the page and the other at the bottom, with vertical rungs drawn sequentially from left to right.

A program in Ladder logic, also called a Ladder diagram, resembles the diagram of a set of relay-based electrical circuits. The major advantage of Ladder logic is its ability to be understood and used by a wide variety of technical personnel such as engineers, electrical technicians, etc., without additional training due to this similarity.

Ladder logic is widely used for programming PLCs, where sequential control of manufacturing processes is required. Ladder programming is useful for simple but critical control systems, or for retrofitting old wired relay circuits. As programmable logic controllers have become more sophisticated, they have also been successfully used in highly complex automation systems.

Ladder language can be considered a rule-based language rather than a procedural language. Each "rung" in Ladder logic represents a rule. When implemented with electromechanical elements, the various rules are all "executed" simultaneously and immediately. When implemented in the logic of a programmable logic controller, the rules are executed sequentially by the software, in a loop. By executing the loop quickly enough, typically several times per second, the effect of simultaneous and immediate execution is achieved.

To introduce the contact language, let's take the circuit diagram illustrated in the Figure below, which allows starting and stopping an electric motor.

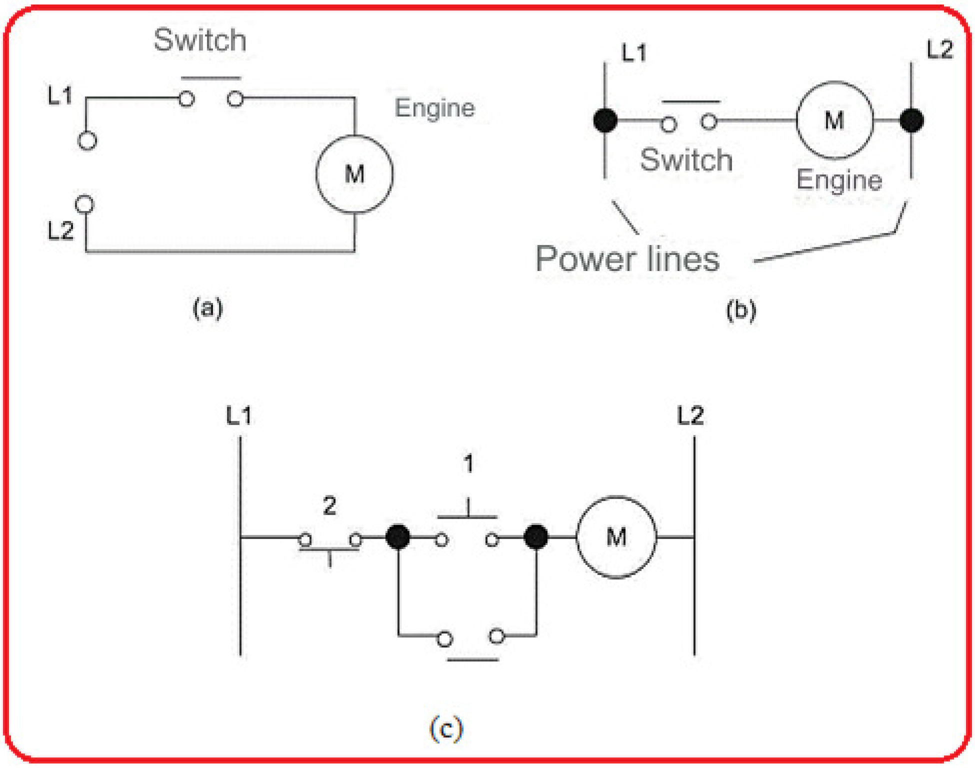


Figure III.11: Contact language (Ladder)

In a contact language diagram, the name of the variable associated with each element and its address are added to the symbol. It is preferable to choose a descriptive name, for example, "motor control switch of the pump" instead of a simple "input," and "pump motor" instead of a simple "output." Figure 4 presents the contact language diagram from Figure 4a as it would be drawn with addresses in the format of (Mitsubishi), (Siemens), (Allen Bradley), and (Schneider). Figure 4a shows that this line of the contact language diagram has an input at address X400 and an output at address Y430. When connecting inputs and outputs, the corresponding devices must be connected to the ports with these addresses.

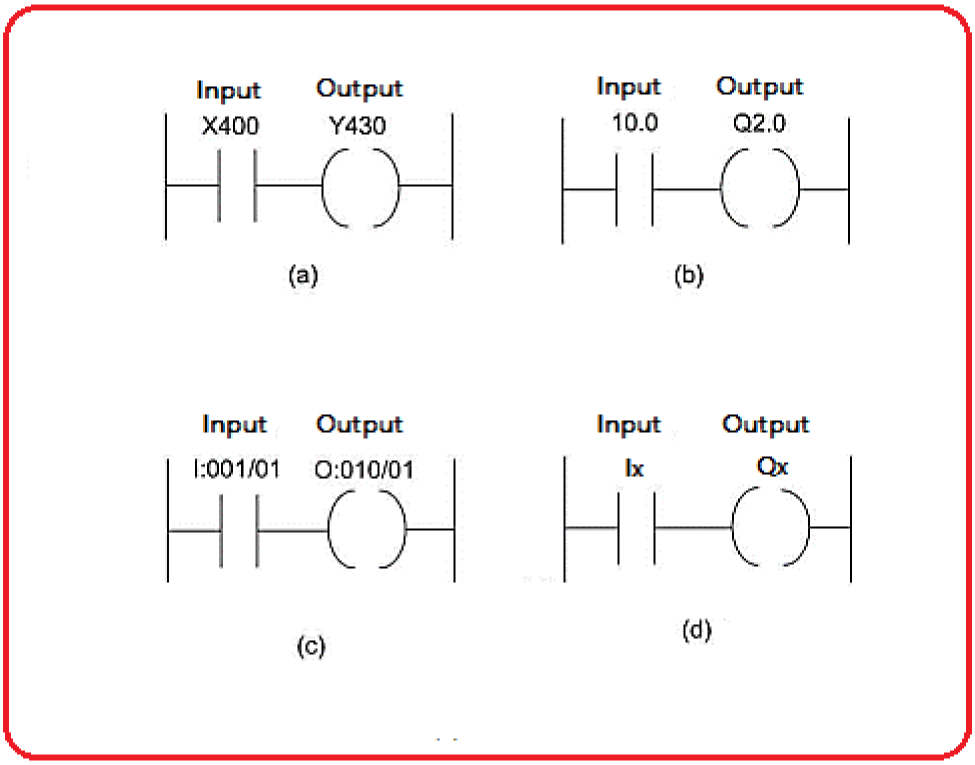


Figure III.12: Adding addresses in the format (a) Mitsubishi, (b) Siemens, (c) Allen Bradley and (d) Schneider.

The programming of PLCs is often based on contact language diagrams. Writing a program is equivalent to drawing a switching circuit. For drawing a contact language diagram, here are the conventions to follow:

- The vertical lines of the diagram represent the power bars between which circuits are connected. The current flow starts from the left vertical line and crosses a horizontal line.
- Each horizontal line defines an operation of the control process.
- Each line must start with one or more inputs and must end with at least one output.

The term "input" corresponds to a control action, such as closing the contacts of a switch used as input. The term "output" corresponds to a device, such as a motor, connected to the output of a PLC. Outputs are not immediately affected during the program analysis. The results of the instructions are placed in memory, and all outputs are affected simultaneously at the end of the program analysis.

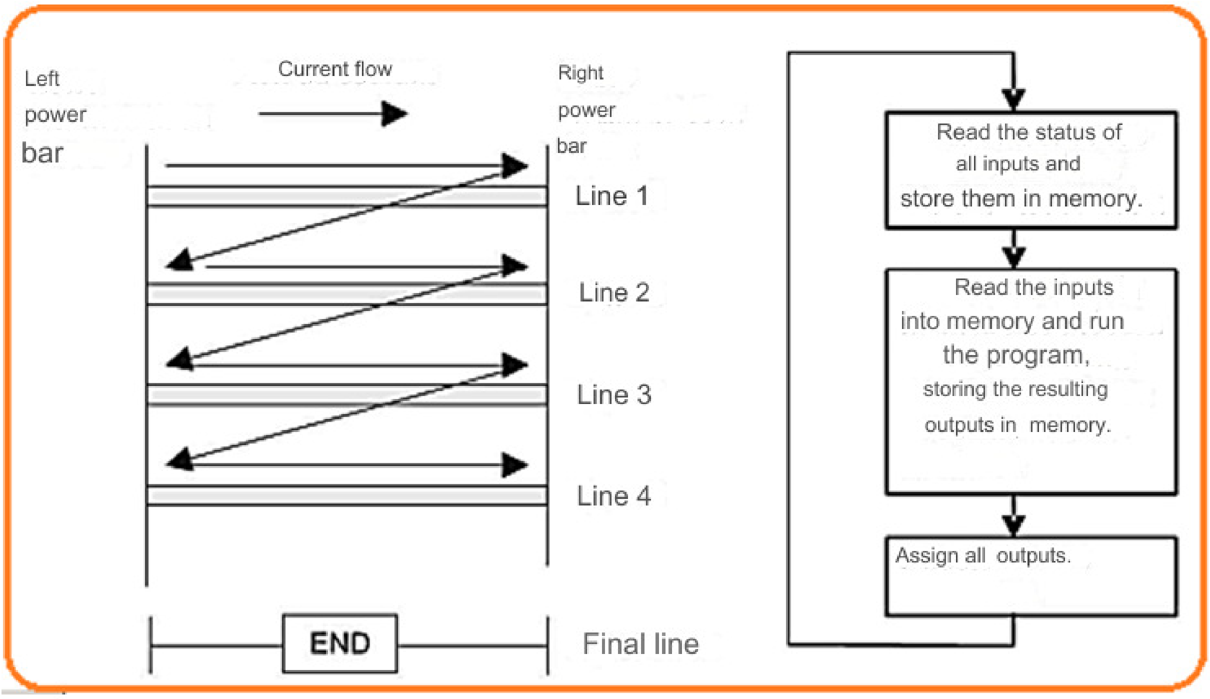
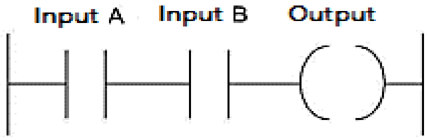


Figure III.13: Analysis of the contact schema

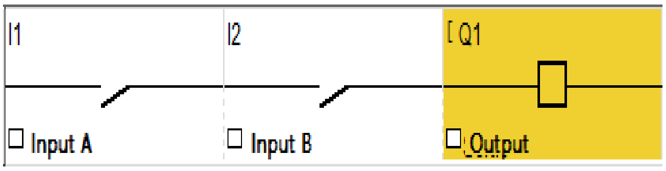
9.1. Representation of logical functions in Ladder

1. The AND function:

An AND gate is used, for example, in the locking system of a machine tool so that it only starts if the safety device is in place and if it is powered. Figure 2.13 shows an AND gate system on a contact schema. The schema begins with -| |-, a set of normally open contacts labeled 'Input A', to represent switch A. It is placed in series with another symbol -| |-, also a set of normally open contacts, labeled 'Input B', to represent switch B. The line ends with -( )-, representing the output. For the output to be active, inputs A and B must occur simultaneously.



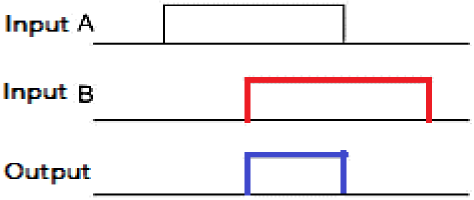
AND function standard Représentation



AND function standard Représentation in **Zelio**

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

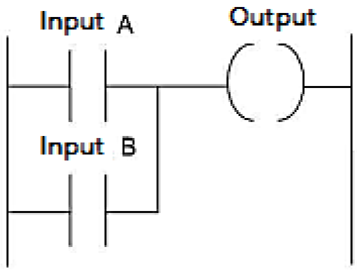
Truth Table



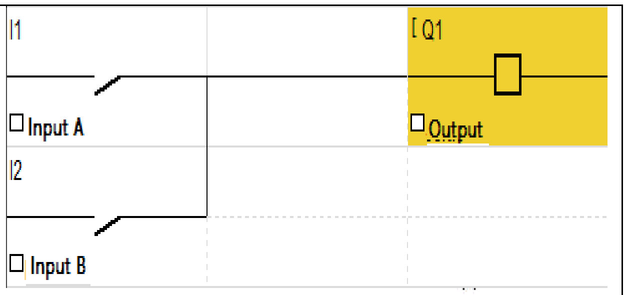
Timing Diagram of the **AND** Function

Figure III.14 : Representation of **AND** function

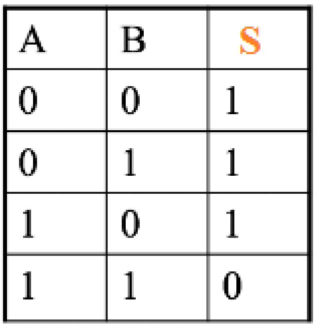
2. **The OR function:** An OR gate is, for example, used in a conveyor belt that transports bottles for packing into cartons.



OR function standard Représentation

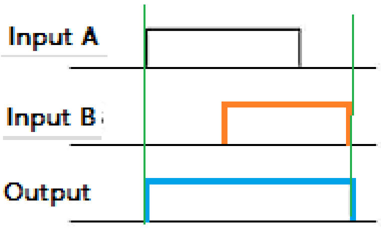


OR function standard Représentation in **Zelio** of **Schneider**



A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

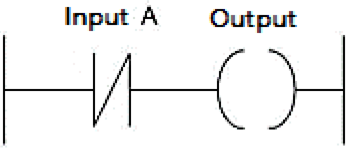
Truth Table



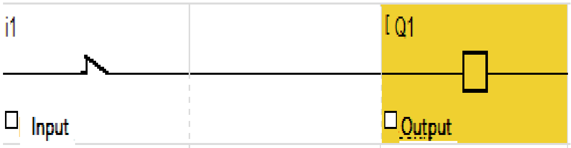
Timing Diagram of the **OR** Function

Figure III.15 : Representation of **OR** function

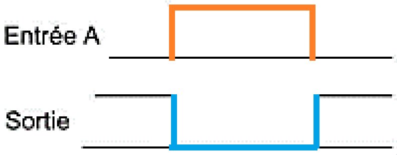
3. **The NOT function:** A NOT gate is, for example, used with a lamp that turns on when it is night. In other words, when there is no light on the light sensor, the output is active**.**

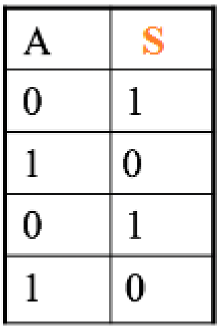


NOT function standard Représentation



NOT function standard Représentation in **Zelio**





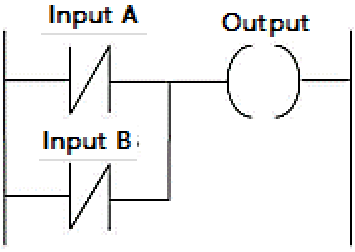
A	S
0	1
1	0
0	1
1	0

Truth Table

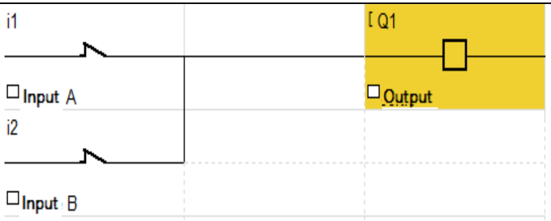
Timing Diagram of the **NOT** Function

Figure III.16 : Representation of **NOT** function

4. **The NAND function:** The figure below illustrates a contact diagram that represents a NAND gate. When both Input A and Input B are at 1, the output is 0. When Input A and Input B are both 0 or when one is 0 and the other is 1, the output is 1.

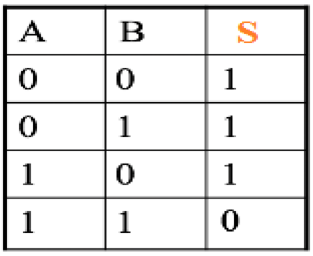


NAND Représentation standard de la function



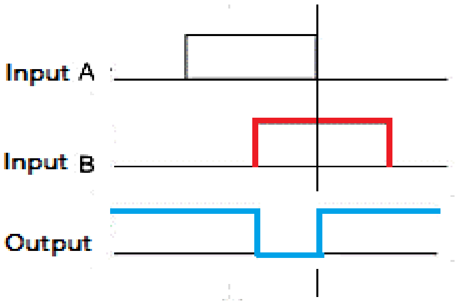
NAND function standard Représentation in **Zelio**

Zelio de Schneider



A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

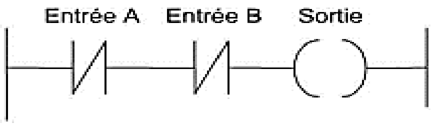
Truth Table



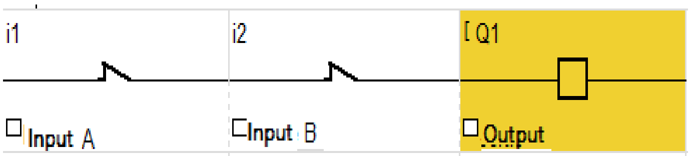
Timing Diagram of the **NAND** Function

Figure III.17 : Representation of **NAND** function

5. **The NOR function:** The figure illustrates a contact diagram that represents a NOR gate. When neither Input A nor Input B is active, the output is 1. When Input A or Input B is 1, the output is 0.



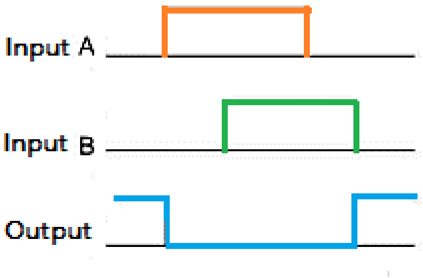
NOR function Représentation



NOR function Représentation in **Zelio**

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

Truth table



Timing Diagram of the **NOR** Function

Figure III.18 : Representation of **NOR** function

9.2. Equations and concepts associated with GRAFCET:

These equations specify the conditions under which each step in the GRAFCET diagram is activated or deactivated. Activation happens when the necessary conditions for the step are met (e.g., the preceding step is active and certain transition conditions are satisfied). Deactivation happens when the subsequent step is activated.

These equations ensure that the GRAFCET progresses logically from one step to the next, based on the conditions and transitions specified.

Definitions:

- *Xi*=1 if step *i* is active.
- *Xi*=0 if step *i* is inactive.
- *Ri*=1 if the receptivity *Ri* is true.
- *Ri*=0 if the receptivity *Ri* is false.

Rules:

2nd Rule: Transition Validation and Activation

A transition is validated when all the immediately preceding steps are active. The transition can only be crossed if:

- It is validated,
- AND the receptivity associated with the transition is TRUE.

3rd Rule: Transition Crossing and Step Activation/Deactivation

The crossing of a transition results in the activation of all the immediately following steps and the deactivation of all the immediately preceding steps.

9.3. Logical Equations:

Using these rules, we can write the logical equations for the activation and deactivation of each step in the GRAFCET.

1. **Activation Condition (Set, *S*):**

A step *Xi* is activated if the preceding step X_{i-1} is active and the receptivity R_i is true.

$$SX_i = X_i - 1 \cdot R_i$$

Where:

- SX_i is the activation condition for step *i*.
- X_{i-1} is the state of the immediately preceding step $i-1$ (1 if active, 0 if inactive).
- R_i is the receptivity condition for the transition to step *i* (1 if true, 0 if false).

2. **Deactivation Condition (Reset, *R*):**

A step $Xi-1$ is deactivated when the following step *Xi* is activated.

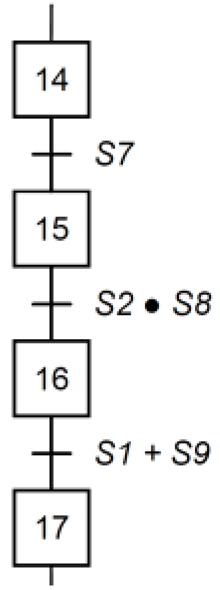
$$RXi-1=Xi$$

Where:

- $RXi-1$ is the deactivation condition for step $i-1$.
- *Xi* is the state of the step *i* (1 if active, 0 if inactive).

Example:

Let's consider a simple sequence with three steps $X14$, $X15$, and $X16$.

	**Activation Equations:** 1. **Step *X*15**: $SX15= X14\cdot S7$ 2. **Step X16**: $SX16= X15\cdot S2\cdot S8$ 3. **Step X17**: $SX17= X16\cdot(S1+ S9)$ **Deactivation Equations:** 1. **Step X14**: $RX14= X15$ 2. **Step X15**: $RX15= X16$ 3. **Step X16**: $RX16= X17$

Based on the provided information, let's summarize the rules and derive the Karnaugh map for step *i* in the GRAFCET.

Rules:

2nd Rule: Activation Condition

The step *Xi* is activated if the preceding step $Xi-1$ is active and the receptivity $Ri-1$ is true.

$$CAXi=Xi-1\cdot Ri-1$$

3rd Rule: Deactivation Condition

The step *Xi* is deactivated if the receptivity *Ri* is true and the subsequent step $Xi+1$ is active.

$$CDXi=Xi+1$$

4. **Effect Memory**

If both the activation condition (CA) and the deactivation condition (CD) for step *i* are false, step *i* remains in its current state (memory effect). The state of *Xi* at time $t+\delta t$ depends on its previous state at time t.

Truth Table for *Xi*:

To derive the truth table and subsequently the Karnaugh map, we need to consider all possible combinations of the states of $Xi-1$, Xi, $Xi+1$, $Ri-1$ and Ri.

Xi	***CAXi***	***CDXi***	***Xi*(*t*+*δt*)**	Note

Xi	*CAXi*	*CDXi*	*Xi*(*t*+*δt*)	Note
0	0	0	0	The step remains inactive (memory effect)
0	0	1	0	The step remains inactive
0	1	0	1	Step activation
0	1	1	1	Activation AND deactivation = Activation
1	0	0	1	The step remains active (memory effect)
1	0	1	0	Deactivation of the step
1	1	0	1	The step remains active
1	1	1	1	Activation AND deactivation = Activation

Karnaugh Map for $Xi(t+\delta t)$:

Here is the Karnaugh map setup for $Xi(t+\delta t)$:

CAXi.CDXi *Xi*	**00**	**01**	**11**	**10**
0	0	0	1	1
1	1	0	1	1

Simplified Boolean Expression:

To derive the simplified Boolean expression from the Karnaugh map, group the 1s in the map and write the expression.

From the Karnaugh map, we get:

$$Xi = CAXi + \overline{CDXi}.Xi$$

$$Xi = (X_{i-1} \cdot R_{i-1}) + (\overline{X_{i+1}}.Xi)$$

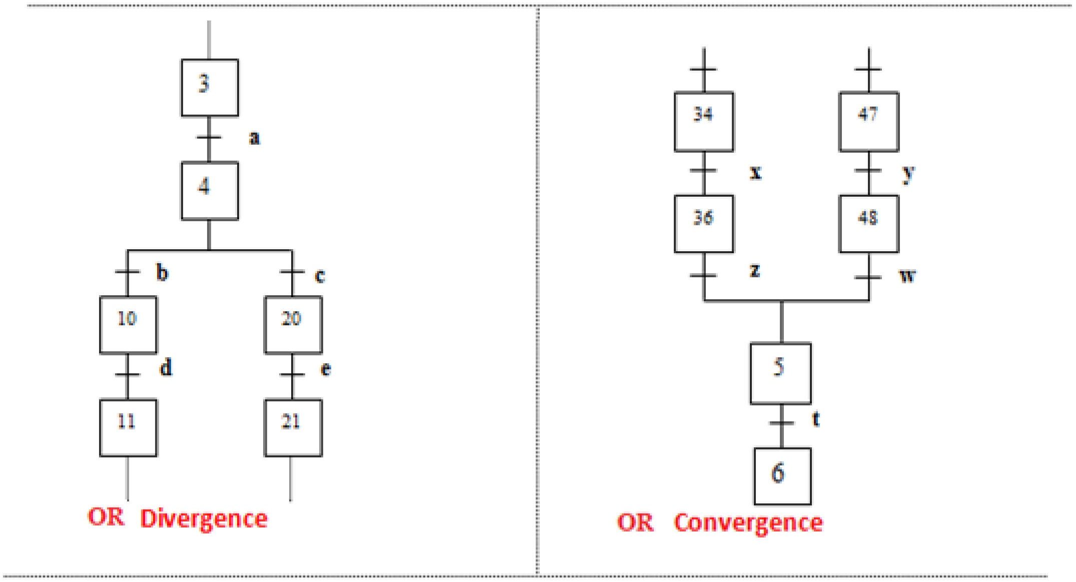
This expression captures the conditions under which step *Xi* will be active at time $t+\delta t$.

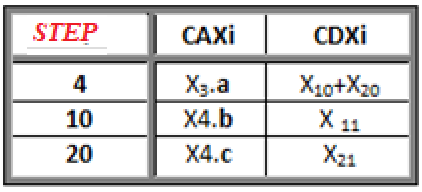
9.4. Alternative Sequences LADDER representation

Alternative sequences are used when the GRAFCET could evolve through one (or more) available sequence(s). This sequence starts with a OR divergence.

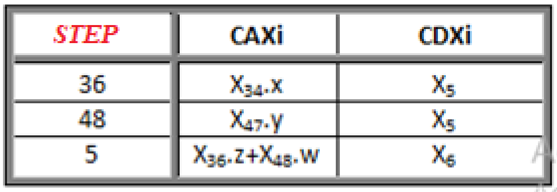
1. **OR Divergence**

In a divergent "OR" sequence, a single step can lead to multiple alternative subsequent steps. This is often used to model processes where a decision point leads to different possible paths





STEP	CAXi	CDXi
4	$X_3.a$	$X_{10}+X_{20}$
10	X4.b	X_{11}
20	X4.c	X_{21}



STEP	CAXi	CDXi
36	$X_{34}.x$	X_5
48	$X_{47}.y$	X_5
5	$X_{36}.z+X_{48}.w$	X_6

2. **OR Convergence**

In a convergent "OR" sequence, multiple alternative preceding steps can lead to a single subsequent step. This is often used to model processes where multiple different paths can lead to the same outcome.

9.5. Simultaneous Sequences LADDER representation

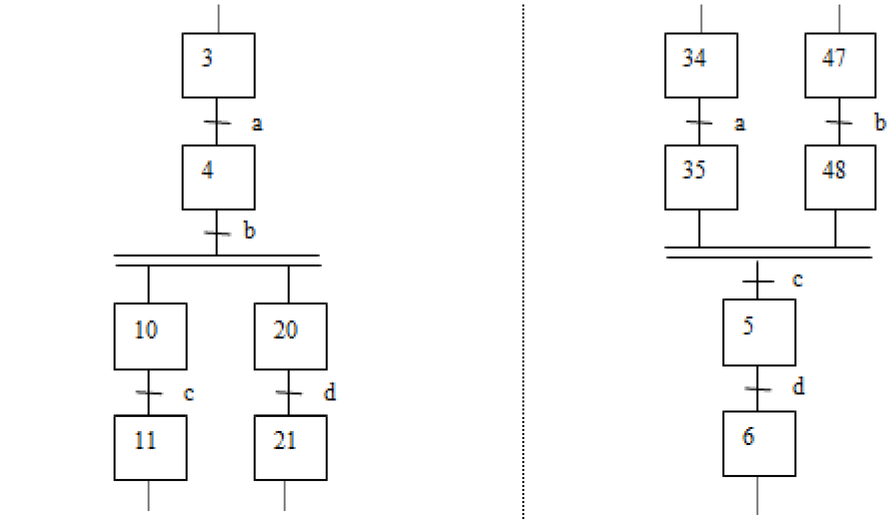
Simultaneous sequences are used when several sequences are required to be activated in parallel, from a given step. These sequences start with an AND divergence.

1. **AND Divergence**

Simultaneous sequences are used when several sequences are required to be activated in parallel, from a given step. These sequences start with an AND divergence.

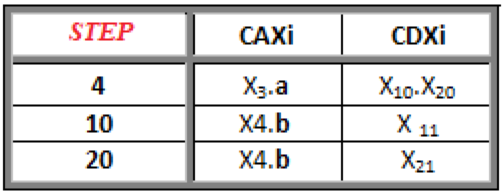
2. AND Convergence

Its representation is throught a horizontal straight double receiving the input links and with one single output link. In order as the AND convergence may evolve it is required that all the preceding steps of the double line (synchronization) are active and their output transition-condition is true.

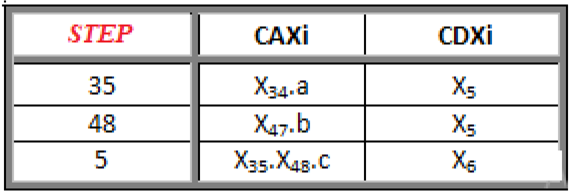


AND Divergence

AND Convergence



STEP	CAXi	CDXi
4	X_3.a	$X_{10}.X_{20}$
10	X4.b	X $_{11}$
20	X4.b	X_{21}



STEP	CAXi	CDXi
35	X_{34}.a	X_5
48	X_{47}.b	X_5
5	$X_{35}.X_{48}$.c	X_6

Chapter IV:

Process Visualization

1. Introduction

Process visualization refers to the use of graphical interfaces to display the status, behavior, and control of automated processes in real-time. It is a crucial part of modern industrial automation systems, enabling operators and engineers to monitor, control, and optimize processes. PLCs (Programmable Logic Controllers) play a vital role in collecting data and providing the logic necessary for process visualization.

2. Automated Networks

With the development of automated systems and electronics, the search for lower costs and the current need to be able to manage production as well as possible, and from the moment when all the equipment is of the computer type, it becomes interesting to interconnect them to a mini-computer or a supervision automaton (Figure IV.1).

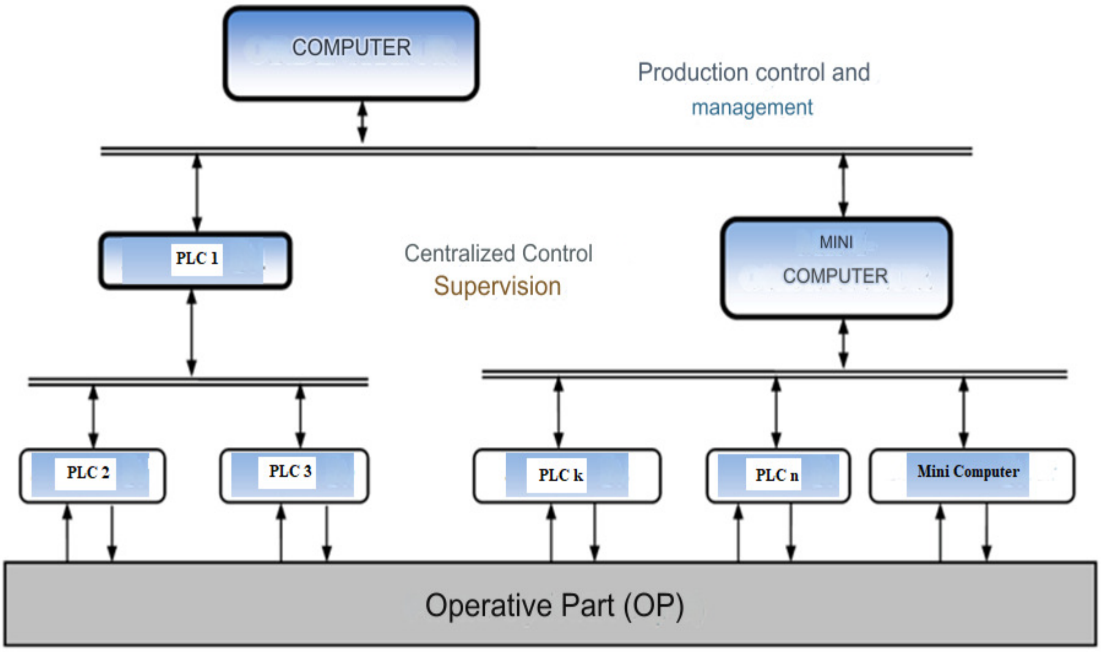


Figure IV.1: Example of a production control and management structure

The interconnection between two PLCs can be achieved very simply by connecting one or more outputs of one PLC to inputs of the other and vice versa (Figure IV.2).

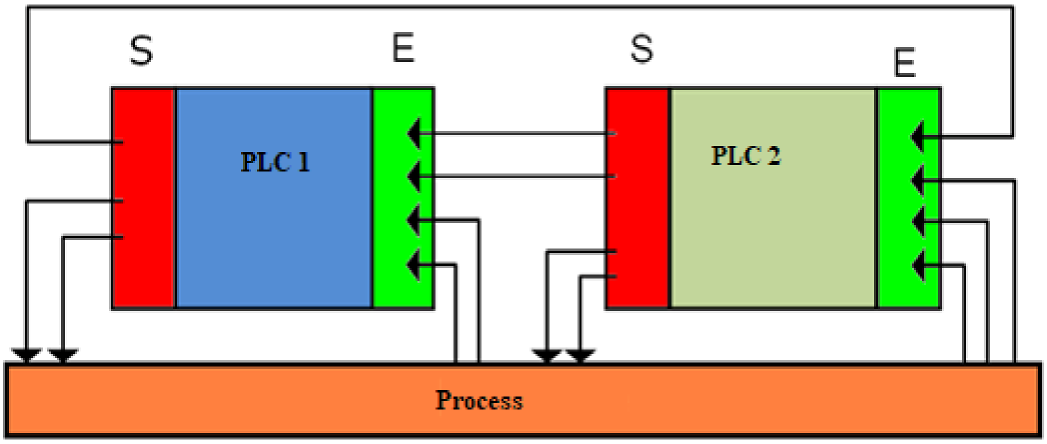


Figure IV.2: Simple interconnection (Inputs/Outputs) between two automatons (API)

This method does not allow the direct transfer of internal variables from one PLC to another, so that they must be converted by program into output variables before their transfer. It becomes costly in terms of the number of inputs/outputs mobilized for this use and cumbersome from a wiring point of view, when the number of variables that must be exchanged becomes large.

2. Fieldbus

To reduce the wiring costs of PLC inputs/outputs, field buses appeared. The use of remote input/output blocks first made it possible to meet this requirement.

The input/output interfaces are moved as close as possible to the sensors. With technological development, sensors, detectors, etc. have become "smart" and have made it possible to connect directly to a bus.

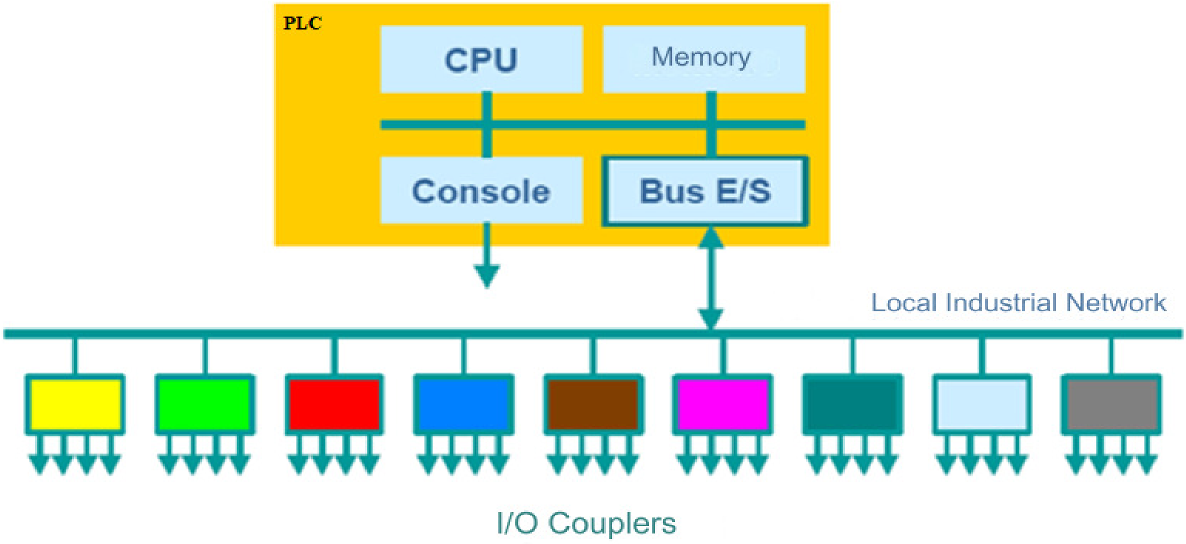


Figure IV.3: Interconnection by remote inputs/outputs

Several communication protocols and standards have appeared to ensure the "multiplexing" of all information coming from the sensors/preactuators, for example the ASi bus (Actuators

Sensors interface) is a Master/Slave type sensor/actuator bus which allows 31 slaves (sensors or preactuators) to be connected to a specific cable (two wires) carrying data and power. This bus is completely standardized and allows the use of technologies from several manufacturers.

Advantages of fieldbuses:

- Reduced cabling costs and ability to reuse existing hardware
- Reduced maintenance costs

Disadvantages of fieldbuses:

- Limited network size
- Latency in time-critical applications
- Total cost

3. Different types of automaton networks:

3.1 Star network:

A common processing center exchanges with each of the other stations. Two stations cannot exchange directly with each other (Figure IV.4). Example the BITBUS field network of the INTEL company

Benefits :

- High exchange speed.
- Different types of transmission media.
- No support access management.

Disadvantages:

- High overall cost.
- Limited developments.
- It all comes down to the central station

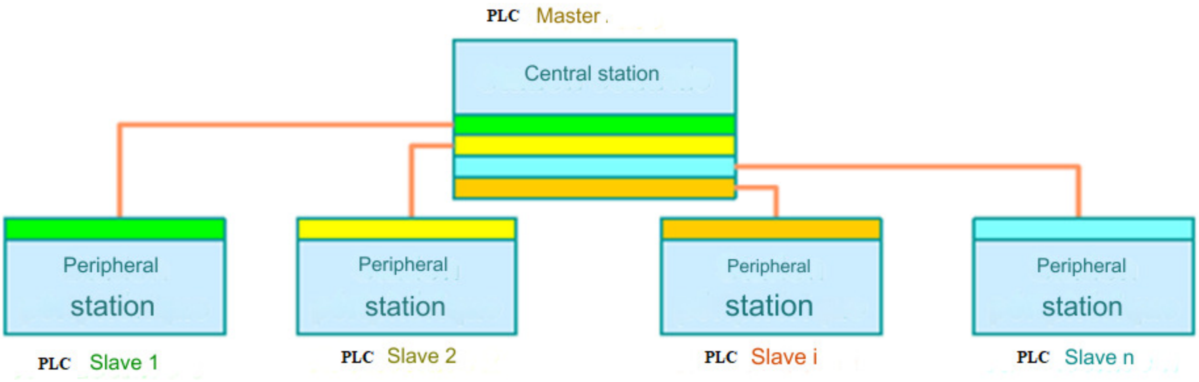


Figure IV.4: Interconnection by remote inputs/outputs

3.2 Ring network:

Each station can communicate with its neighbor. This solution is interesting when a station must receive information from the previous station or transmit it to the next one (Figure IV.5).

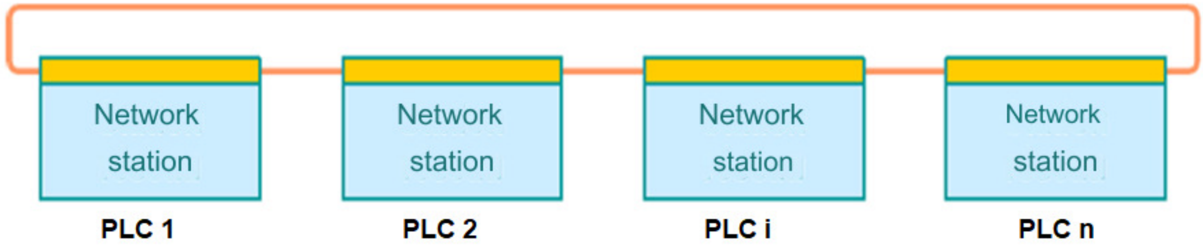


Figure IV.5: Ring Topology

Benefits :

- Signal regenerated therefore reliable.
- Easy control of exchanges (the message returns to the sender).

Disadvantages:

- Each station is blocking.
- An extension temporarily interrupts the network.

3.3 Hierarchical network:

This is the most efficient form of network. It offers great flexibility of use, information can circulate between stations of the same level or circulate from the most advanced station (generally a computer) to the simplest, and vice versa (Figure IV.6).

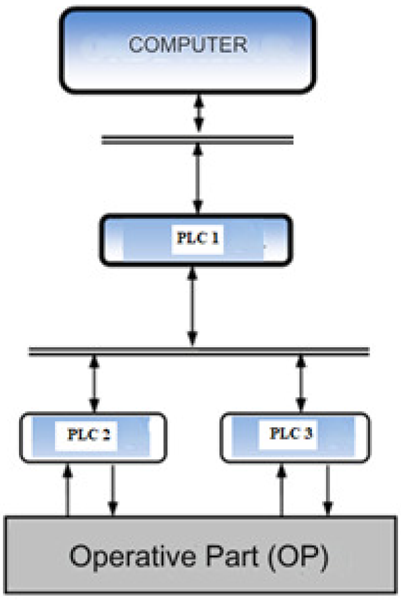
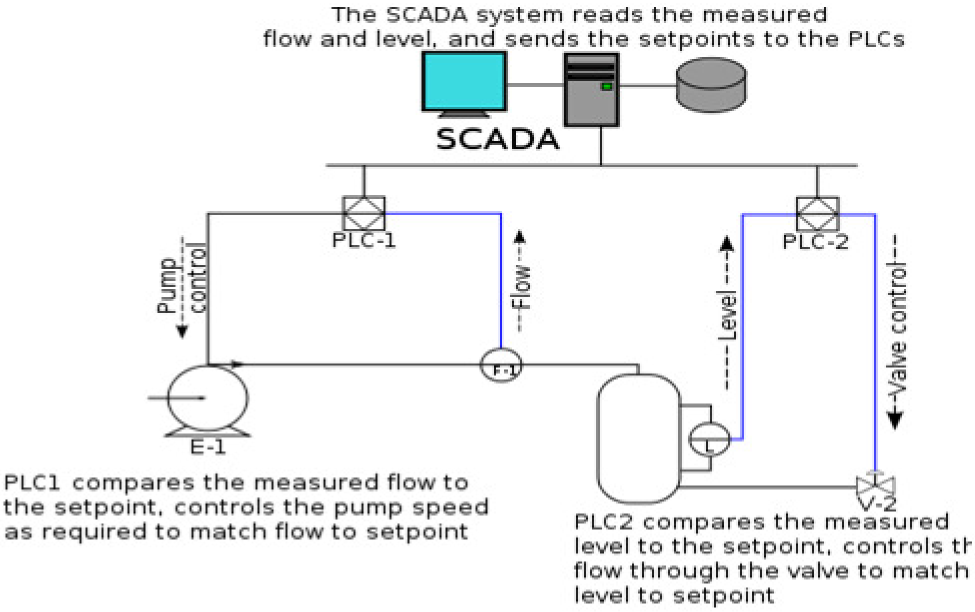


Figure IV.6: Hierarchical network

4. The SCADA system

SCADA stands for Supervisory Control and Data Acquisition. It is a type of software application program for process control. SCADA is a central control system consisting of controllers network interfaces, input/output, communication equipment and software. SCADA systems are used to monitor and control the equipment in the industrial process which includes manufacturing, production, development and manufacturing. Infrastructural processes include gas and oil distribution, electric power and water distribution. Public utilities include a bus traffic system, an airport. SCADA system takes meter readings and checks the status of sensors at regular intervals so that it requires minimum human interference.



The SCADA system reads the measured flow and level, and sends the setpoints to the PLCs
SCADA
PLC-1
PLC-2
Pump control
Flow
Level
Valve control
F-1
L
E-1
V-2
PLC1 compares the measured flow to the setpoint, controls the pump speed as required to match flow to setpoint
PLC2 compares the measured level to the setpoint, controls tł flow through the valve to match level to setpoint

Figure IV.7: General SCADA network

4.1 History of SCADA

Earlier, the control of industrial plants and manufacturing workshops could be done manually using analog equipment and push buttons. As the size of the industry increases, so they used timers and relays to provide supervisory control at a fixed level for minimum automation. Thus, a fully automated system with a more efficient system was required for all industries.

We know that for industrial control purposes, computers were implemented in 1950. After that, the concept of telemetry was implemented for data transmission as well as communication. In 1970, SCADA system was developed with microprocessors along with PLC.

These concepts were thus fully aided in the development of remotely operated automation in industries. Distributed SCADA systems were implemented in 2000. Subsequently, new SCADA systems were developed to monitor and control data in real time anywhere in the world.

4.2 SCADA system architecture

Generally, SCADA system is a centralized system that monitors and controls the entire area. It is a pure software package that is positioned above the hardware. A supervisory system gathers data about the process and sends the control commands to the process. SCADA is a remote terminal also known as RTU.

Most of the control actions are performed automatically by RTUs or PLCs. RTUs consist of the programmable logic converter which can be set according to specific needs. For example, in the thermal power plant, the water flow rate can be set to a specific value or it can be changed according to the needs.

The SCADA system allows operators to change the flow setpoint and activate alarm conditions for loss of flow and high temperature, and the condition is displayed and recorded. The SCADA system monitors the overall performance of the loop. The SCADA system is a centralized system that can communicate with wired and wireless technology with Clint devices. The SCADA system controls can perform all kinds of industrial processes.

For example, if too much pressure builds up in a gas pipeline, the SCADA system can automatically open a relief valve.

4.2.1 Hardware architecture

SCADA system generally can be classified into two parts:

- Client layer
- Data Server Layer

The Clint layer responds to human-machine interaction.

The data server layer handles most of the data activity processes.

SCADA station refers to servers and is composed of a single PC. Data servers communicate with field equipment through process controllers such as PLCs or RTUs. PLCs are connected to data servers either directly or through networks or buses. SCADA system uses WAN and LAN network, WAN and LAN consist of internet protocols used for communication between master station and devices.

Physical equipment such as sensors connected to PLCs or RTUs. RTUs convert sensor signals into digital data and send digital data to the master. Depending on the feedback from the master received by the RTU, it applies the electrical signal to the relays. Most of the monitoring and control operations are performed by RTUs or PLCs as can be seen in the figure.

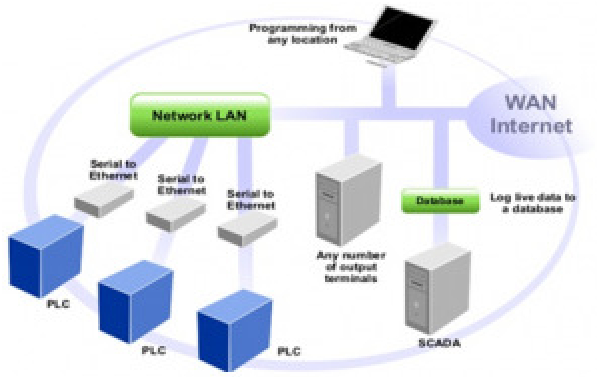


Figure IV.8: Hardware architecture of SCADA system

4.2.2 Software architecture

Most of the servers are used for multitasking and real-time database. Servers are responsible for data collection and processing. SCADA system consists of software program to provide trends, diagnostic data and manage information such as planned maintenance procedures, logistics information, detailed schematics for a particular sensor or machine and troubleshooting guides from the expert system. This means that the operator can see a schematic representation of the controlled plant.

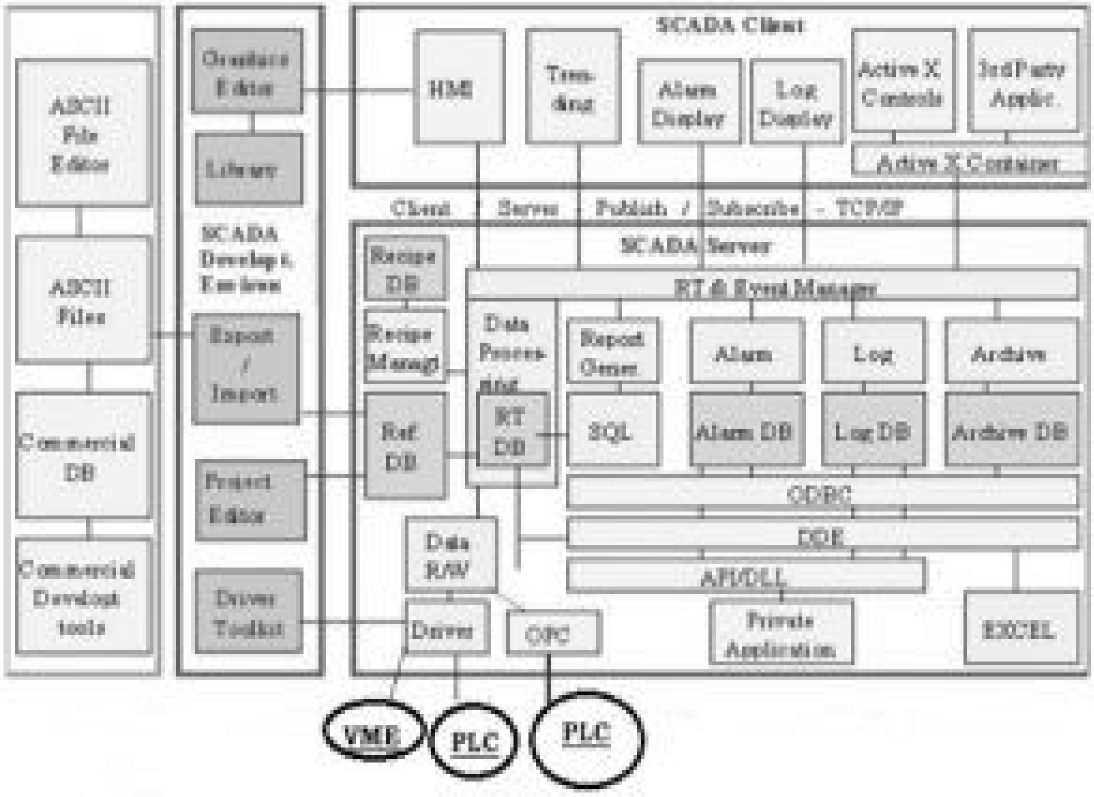


Figure IV.9:SCADA Software Architecture

Examples are alarm checking, calculations, logging and archiving polling controllers on a set of parameters, these are usually connected to the server.

In principle, the SCADA system is composed of the following elements:

- ✓ **RTU (Remonte Terminal Unit) or API**

SCADA starts with Remote Terminal Units (RTU) and Programmable Logic Controllers (PLC). It is used to collect information from the field instrumentation and transmit it to the MTU through the communication system. RTU can be used and replaced by Programmable Logic Controllers (PLC) in case we want to control some points of the system by a program because it is advantageous to write a program to control the system unlike an RTU.

- ✓ **MTU (Master Terminal Unit)**

We can say that the main station has several stations (RTU) connected by different means of communication. Its role is to periodically collect field data from the RTU stations and allows remote control and transmission of information to the operator via the HMI and send the operator's instructions to the field instrumentation.

4.3 Communication system

Communication between MTU and different RTUs can be different means for example internet, wireless network, cabling…

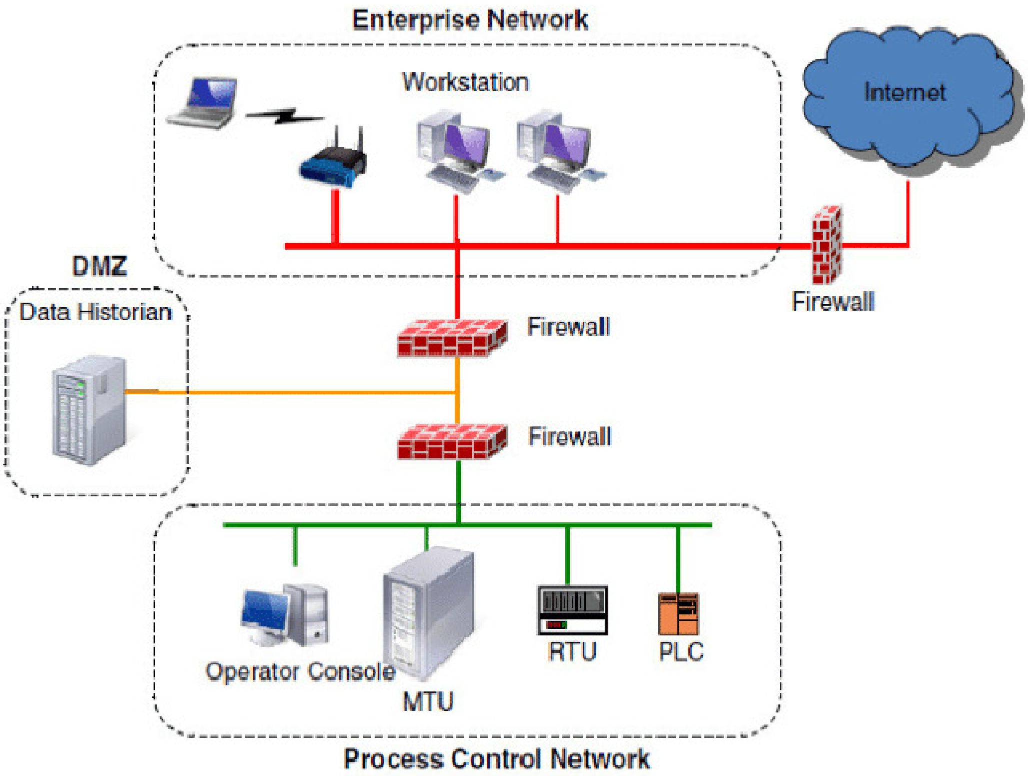


Figure IV.10: General diagram of a SCADA system.

SCADA encompasses data transfer between the server (MTU, master terminal units) and one or more remote terminal units (RTUs) and between the server and the operators' terminals. The figure below shows a diagram of the architecture of a SCADA network that uses routers to reach the control center via the Internet.

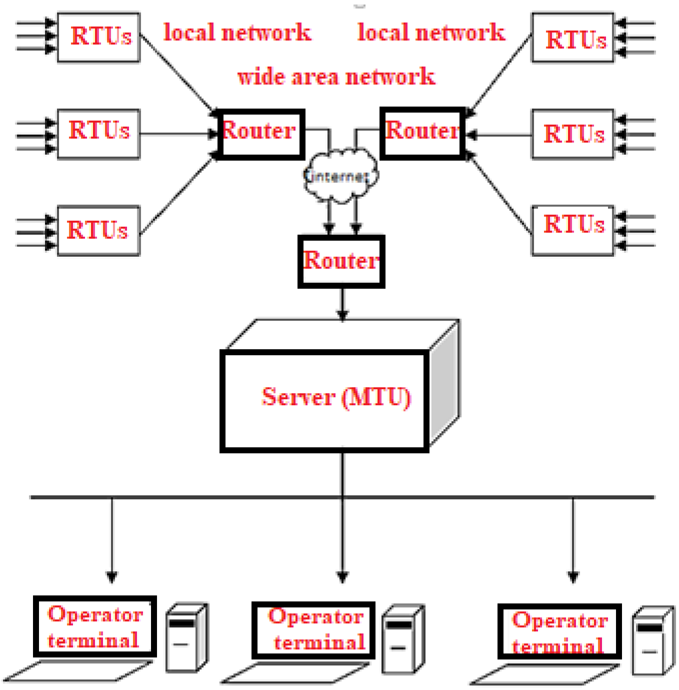


Figure IV.11: Supervision architecture in a SCADA environment.

4.4 Protocol used in a SCADA environment

In the real world of device-to-device process automation communication, a dialogue or conversation between devices takes place systematically, in many cases, over different types of communication networks and in different languages.

These are the open protocols that many manufacturers adapt to easily integrate their products into a market, an open protocol means that the specifications are published and can be used by anyone freely or by License.

Open protocols are usually supported by a combination of user groups, professional societies and government. This gives users a much wider choice of devices.

Communication protocols in a SCADA environment evolve following the need to send and receive data deemed critical generally for long distances and in real time, this perspective has given rise to several protocols that we will develop the most used.

4.4.1 Modbus protocol

Modbus is one of the oldest and by far the most popular automation protocol in process automation and SCADA used in industrial electronics today.

MODBUS is a communications protocol published by Modicon in 1979 for use with its programmable logic controllers. Modbus provides a common language for devices and equipment to communicate with each other, for example, MODBUS enables devices on a system that measures temperature and humidity to be connected on the same network or communicate the results to a supervisory computer or PLC.

Address Field	Field Function Code	Order Data Field	Error checking field

Address Field: corresponds to the address of the Slave station receiving the Request.

Field Function Code: this is the command to read or write data to the slave.

Data Field: if a write command was launched by the master.

Error checking field: is a value created by the master or slave at the beginning of the transmission or response, then checked when the message is received to verify that the contents are correct.

A slave's response consists of fields confirming that it received the request, the data to return, and a data verification error.

No error occurs, the slave's response contains the requested data, if an error occurs in the message request received by the slave, or if the slave is unable to perform the requested action, the slave will return an exception message in response.

The error checking field of the slave telegram allows the master to confirm that the message content is valid.

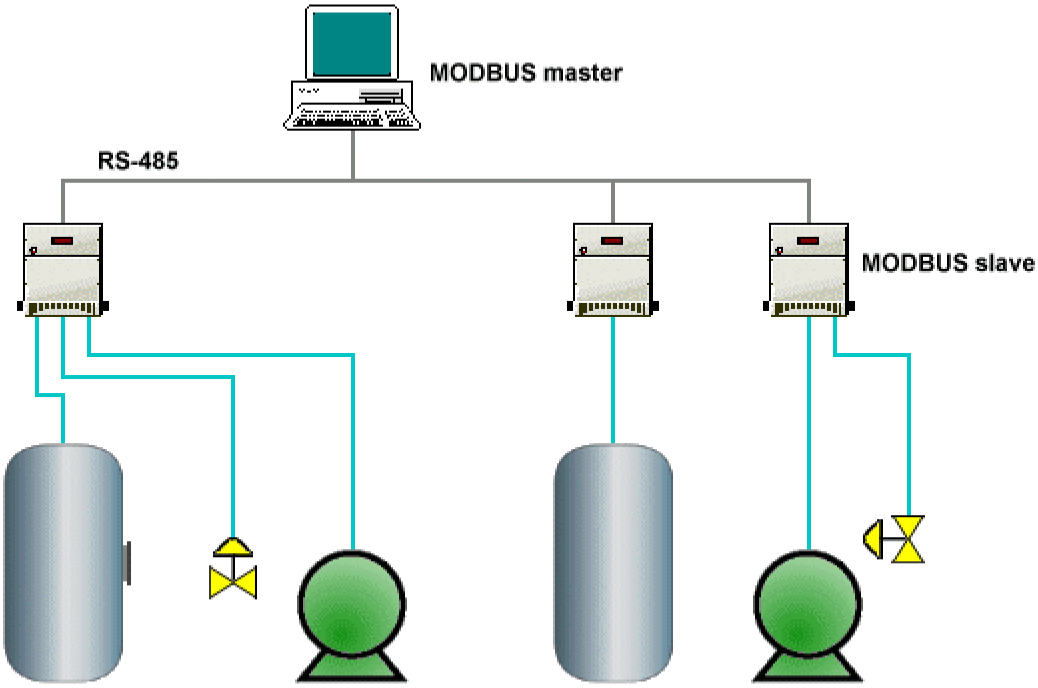


Figure IV.12: Communication by MODBUS Protocol.

4.4.2 DNP3 Protocol

It is a communication protocol designed for transferring data and control commands from a master to one or more devices. DNP3 was developed by Westronic Inc., especially for use in SCADA system.

DNP3 manages a communication in balance function which means that the master station and slave station can transmit information. It is built on the EPA profile (Enhanced Performance Architecture) which is a simplified version of the OSI model (Open System Interconnections). It has 3 layers: Physical, link and application. However, to allow the transmission of large messages (2 kilobytes or more), segmentation and data assembly functions have been added. All of these functions constitute a pseudo-Transport layer.

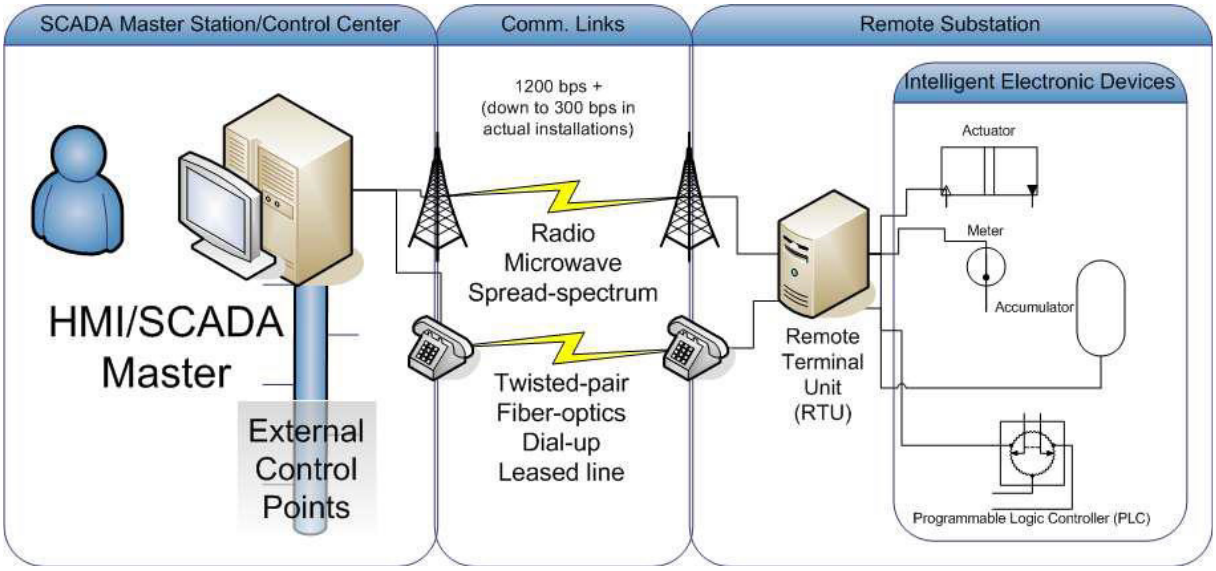


Figure IV.13: Communication via DNP3 protocol.

4.4.3 PROFIBUS protocol

PROFIBUS: (Process Field bus) is a protocol for fieldbus communication in technological automation, connecting automation systems and controllers to decentralized field devices such as sensors, actuators and encoders.

PROFIBUS networks exchange data using a single bus cable, users can combine various types of PROFIBUS protocols with own software and other requirements, resulting in a unique application profile. It was first developed in 1989 by BMBF and then used by SIEMENS.

PROFIBUS is also a Master-slave type protocol like MODBUS but with additional token ring which is a protocol to allow multiple masters to connect. It can take three transmission media or physical media.

The PROFIBUS-DP: used to operate the sensor and actuators via a centralized controller in a production plant, Automation application. The many standard diagnostic options, in particular, are centralized here. Typically the standard cable color will be purple, but it can be any other color.

The PROFIBUS-PA: it is the protocol designed for automation processes. In reality, PROFIBUS PA is a type of PROFIBUS DP application profile. It standardizes the measured data transmission process. It has been designed specifically for use in hazardous environments.

The PROFIBUS-FMS:is used for non-deterministic communication.

PROFIBUS: meets unanimously recognized international standards. Its architecture is based on 3 layers inspired by the 7-layer model of OSI, layer 1, physical, describes the physical characteristics of the transmission.

Layer 2, data link, specifies the rules for accessing the bus. Finally, layer 7, application, defines the common mechanisms useful for distributed applications and the meaning of the information exchanged, the following figure represents the architecture of PROFIBUS communication.

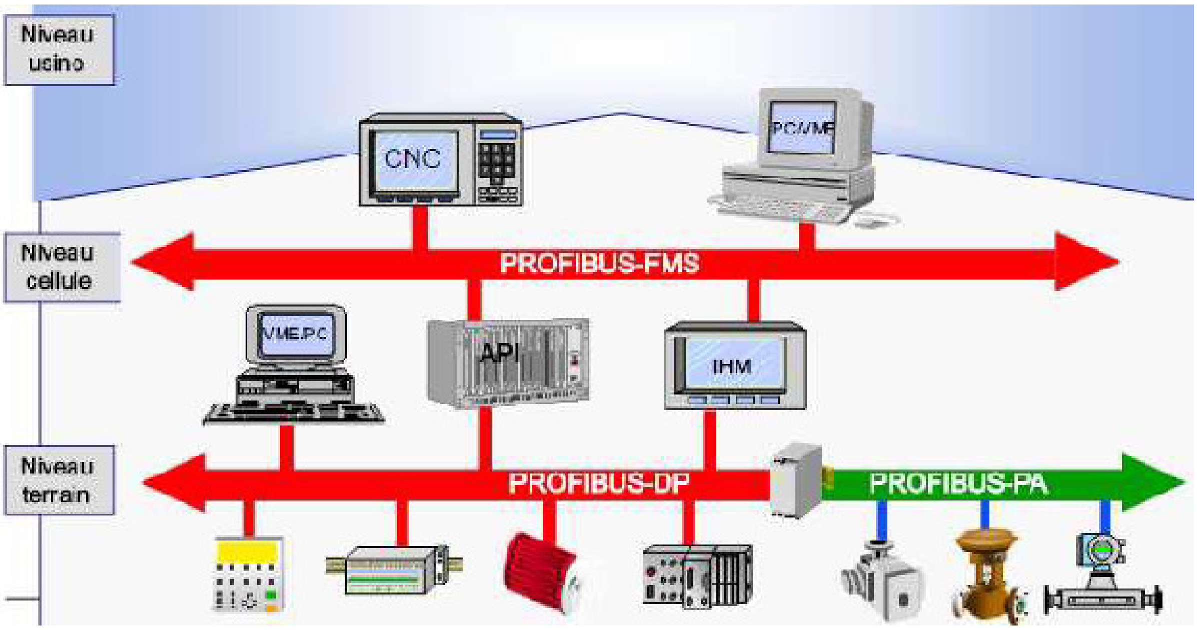


Figure IV.14: Communication via PROFIBUS Protocol.

4.5 SCADA HMI Software

HMI stands for Human Machine Interface. We use HMIs in industry to control and monitor machines. Many times an HMI will be in the form of a screen, much like a computer monitor, and more times than not they are touch screen.

An operator or maintenance personnel can operate and monitor the machine from the HMI. They can include information such as temperature, pressure, process steps and material count. They can also display very precise levels in the tank and the exact position of the machines. Machine information previously used for multiple indicators can now be viewed on a single screen.

To confirm the HMI connection to the machine for control and monitoring, special software must be used so that engineers can program them properly. The software allows the engineer to design what the operator will actually see on the screen, what they can monitor on the screen, what buttons can be pressed, and how the operator can manipulate the machine. The person programming the HMI must program each indicator and button to a specific input or output address of a PLC. The HMI and the PLC must be compatible, this means that they must be able to talk to each other through the protocols.

4.6 Operation of the SCADA system

The SCADA system performs the following functions

- Data acquisitions
- Data communication
- Presentation of information/data
- Monitoring / Control

These functions are performed by sensors, RTUs, controllers, communication network. Sensors are used to collect important information and RTUs are used to send this information to the controller and display the system status. Depending on the system status, the user can give the command to other components of the system. This operation is performed by the communication network.

4.6.1 Data acquisitions

The real-time system consists of thousands of components and sensors. It is very important to know the status of some components and sensors. For example, some sensors measure the water flow from the tank to the water tank, and some sensors measure the pressure value when water is released from the tank.

4.6.2 Data communication

SCADA system uses wired network to communicate between users and devices. Real-time applications use many sensors and components that need to be controlled remotely. SCADA system uses internet communications. All information is transmitted over the internet using specific protocols. Sensors and relays are not able to communicate with network protocols, so RTUs are used to communicate sensors and network interfaces.

Presentation of information/data

Normal circuit networks have indicators which can be visible to control but in real time SCADA system there are thousands of sensors and alarm which are impossible to manage simultaneously. SCADA system uses Human Machine Interface (HMI) to provide all the information collected from the different sensors.

4.6.3 Monitoring / Control

SCADA system uses different switches to operate each device and displays the status of the control area. Any part of the process can be enabled/disabled from the control station using these switches. SCADA system is implemented to work automatically without human intervention but in critical situations it is managed by manpower.

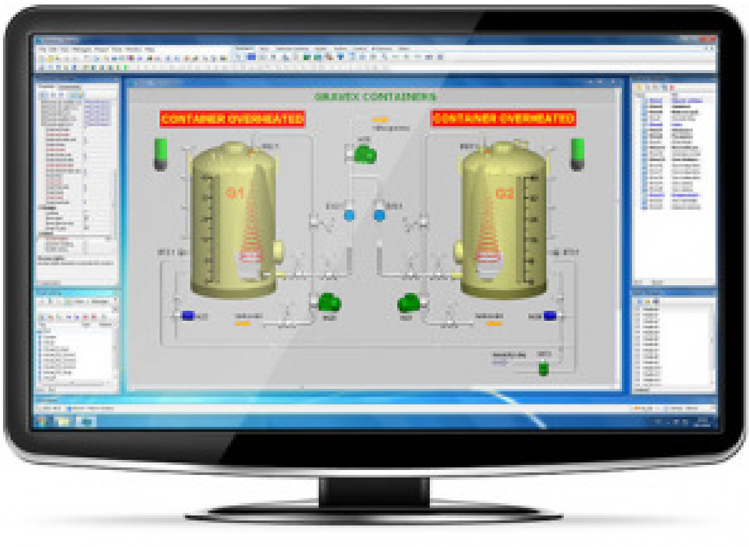


Figure IV.15: Human-machine interface

For example, the tank water level alarm is set to values of 60% and 70%. If the water level reaches more than 60%, the alarm gives a normal warning and if the water level reaches more than 70%, the alarm gives a critical warning.

At present, SCADA networks are widely used in today's industries to check and review real-time data, industrial processes can be controlled, communicate with devices. So SCADA systems are essential for industrial organizations because these systems include hardware and software. So, SCADA security is also essential in industries.

The term SCADA security is used to protect SCADA networks that are made with computer hardware. The SCADA networks used by some of the systems are electricity, natural gas, etc. Private and government organizations have taken the measures of these networks because of the valuable role of ensuring the security of SCADA systems.

4.7 SCADA Security Examples

The threats that occur in SCADA systems are as follows.

- The Pirates
- The terrorists
- Malware
- Error inside

SCADA security weakness occurs mainly due to the following reasons.

- Bad training
- Development of application vulnerabilities

- Problems while monitoring
- Less maintenance

The SCADA system can be protected by mapping all the systems present, monitoring and detecting the institute, and creating processes for network security.

4.8 SCADA for remote industrial installation

In large industrial plants, many processes are happening simultaneously and each one needs to be monitored which is a complex task. SCADA systems are used to monitor and control equipment in industrial processes which include water distribution, oil distribution and power distribution. The main objective of this project is to process the data in real time and control the remote industrial environment on a large scale. In the real time scenario, a temperature recording system for a remote plant operation is used.

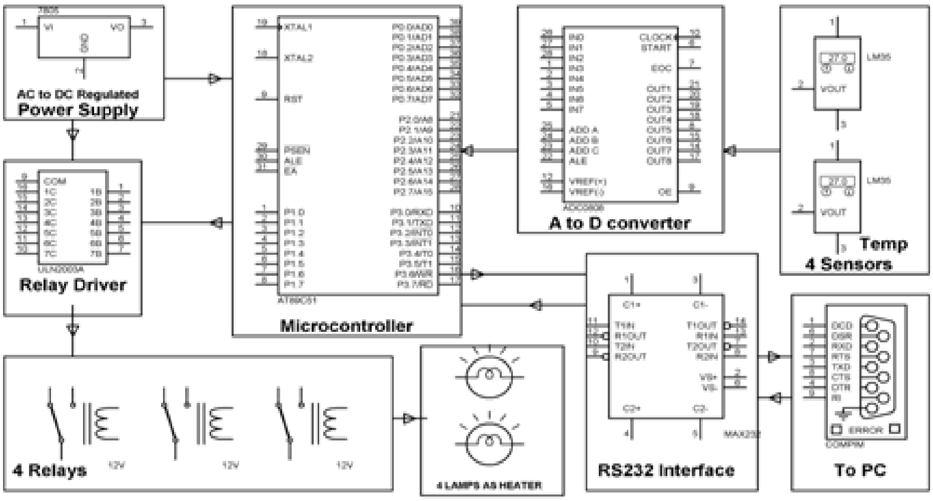


Figure IV.16: Functional diagram of industrial temperature control plant

The temperature sensors are connected to the microcontroller, which is connected to the PC at the front, and the software is loaded on the computer. The data is collected from the temperature sensors. The temperature sensors continuously send the signal to the microcontroller which accordingly displays these values on its front panel.

The parameters like low limit and high limit can be set on the computer screen. When the temperature of a sensor exceeds the set point, the microcontroller sends a command to the corresponding relay. The heating elements connected through the relay contacts are turned off and on.

This is a temperature recording system. Here 8 temperature sensors in multiplexing mode are connected to the microcontroller via ADC 0808. Then the values of all the sensors are sent serially by the microcontroller via Max 32 to the com port of the PC. A software “DAQ System” loaded on the PC takes these values and displays them on its front panel, and also saves them in the database “daq.mdb”.

Some parameters like set point, low limit and high limit can be set interactively on the computer screen. When the temperature of some sensors increases beyond the set point, the microcontroller sends commands to the relay driver IC. The heating elements connected via relay contacts are (specific to that sensor) turned OFF (or ON otherwise). The high and low limits are for alarm. When the temperature exceeds the high limit or below the low limit, the alarm is activated.

Benefits

The advantages of SCADA system are as follows.

- Quality of service can be improved
- Reliability can be improved
- The maintenance cost is lower
- The operation can be reduced
- Major system parameters can be monitored
- Staff numbers may be reduced
- Repair time can be reduced
- Fault detection and localization
- It stores a large amount of data
- Depending on the user's needs, it displays data in various formats.
- Thousands of sensors can be interfaced with SCADA for control and monitoring
- Simulations of real data can be obtained by operators
- Give a quick answer
- It is flexible and scalable while adding additional resources.
- The SCADA system provides mechanical and graphical information on board
- The SCADA system is easily expandable. We can add a set of control units and sensors as needed.
- The SCADA system is capable of operating in critical situations.

Disadvantages

The disadvantages of SCADA system are as follows.

- It is complex in terms of dependent modules and hardware units.
- It needs skilled analysts, programmers and operators to maintain
- High installation cost
- Unemployment rates may be increased
- This system supports restricted hardware devices and software

Applications

The applications of SCADA system are as follows.

- Energy production and distribution
- Public transport
- Water and sewer system
- Manufacturing
- Industries and buildings
- Communication networks
- Oil and gas industries
- Production, transport and distribution of electricity
- Water distribution and reservoir system
- Public buildings such as electric heating and cooling system.
- Generators and turbines
- Traffic light control system

So, this is an overview of SCADA (Supervisory Control and Data Acquisition) system. This system is controlled by a computer, used to control and monitor different processes in factories. This system uses GUI (Graphical User Interface), data communications and extensive management for monitoring systems.

Chapter V:

programmable logic controller dedicated to security

1. Introduction

A **Safety PLC (Programmable Logic Controller)** is a specialized type of PLC used to ensure the safe operation of machinery and processes in industrial environments. Safety PLCs are designed to perform both standard automation functions and safety-critical tasks, such as emergency shutdowns, machinery interlocks, and other safety measures required to protect workers, equipment, and the environment.

The Applications of Safety PLCs in industry are:

- **Manufacturing and Assembly Lines**: To ensure safe operation around robots, conveyors, and other automated systems.
- **Process Industries**: In chemical, oil and gas, and power plants, where failures could lead to hazardous situations.
- **Automotive**: For controlling automated welding, painting, and assembly processes.
- **Food and Beverage**: To manage the safety of machinery like mixers, ovens, and packaging equipment.

Overall, Safety PLCs are critical in any environment where machinery poses a potential risk to human health or the environment.

A Safety PLC is just like a regular PLC and is programmed the same way. The logic in the Safety PLC is locked. It has safety signatures ensuring that coding has not been changed. Physically, other than they might be a different color, they look exactly the same! But…that's where the similarity ends.

2. Safety Instrumented System (SIS)

A Safety Instrumented System (SIS) is a crucial component in industrial processes that is designed to prevent hazardous events by reducing the risks associated with process operations to an acceptable level. An SIS consists of various safety instrumented functions (SIFs) and is implemented to protect people, equipment, and the environment from accidents such as explosions, fires, or toxic releases.

SIS is used across various industries where hazardous processes are involved:

- Oil and Gas: To prevent explosions, fires, and toxic releases by controlling critical process variables like pressure and flow.
- Chemical and Petrochemical: For emergency shutdowns, pressure relief, and isolation of hazardous chemicals.
- Power Generation: To prevent overheating, overpressure, and other risks in steam boilers, reactors, and turbines.

- Pharmaceuticals: To ensure safe mixing, heating, and cooling of reactive chemicals.

- Food and Beverage: For ensuring safe operation of cooking, sterilization, and chemical handling equipment.

- Nuclear Facilities: To prevent radiation leaks, reactor failures, and other nuclear hazards.

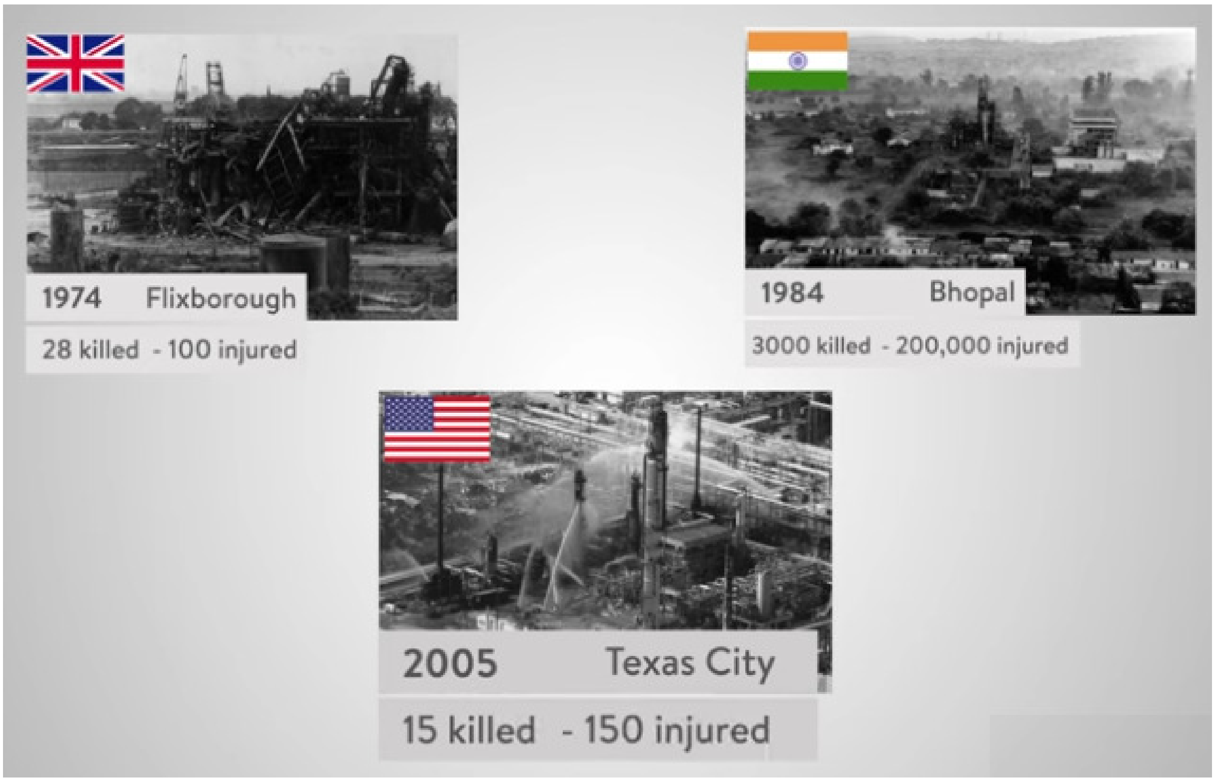


Figure V.1 : Accidents led to establishement safety measures in industry

2.1 Components of a Safety Instrumented System:

1. Sensors: Devices that continuously monitor the process conditions, such as temperature, pressure, flow, level, or gas concentration. Sensors detect any deviations from normal operating conditions that could lead to unsafe situations.

2. Logic Solver: Often a Safety PLC (Programmable Logic Controller) or a dedicated microprocessor-based system, the logic solver receives signals from the sensors, processes these inputs based on pre-configured safety logic, and determines whether a corrective action is necessary.

3. Final Control Elements: Devices that implement the safety action, such as shutting down a valve, stopping a pump, or initiating an alarm. These elements execute the actions directed by the logic solver to bring the process back to a safe state.

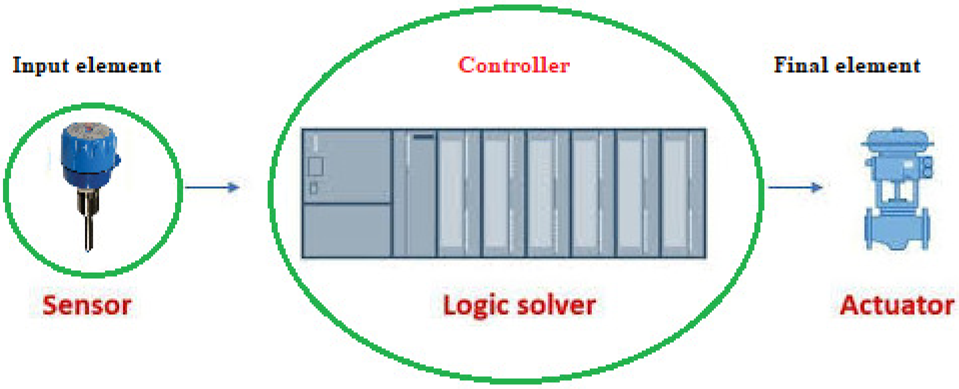


Figure V.2 : Components of a Safety Instrumented System (SIS)

Safety Instrumented System (SIS) and Basic Process Control System (BPCS) are both critical components in industrial automation, but they serve distinct purposes within a process plant or facility.

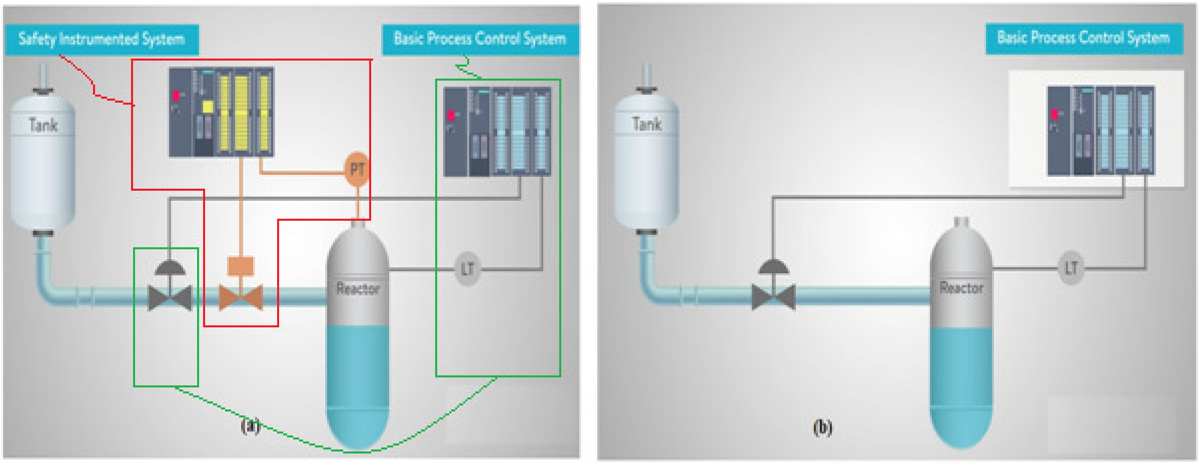

Safety Instrumented System
Basic Process Control System
Tank
PT
Reactor
LT
(a)
Basic Process Control System
Tank
Reactor
LT
(b)

Figure V.3 : Safety Instrumented System (SIS) and Basic Process Control System (BPCS) (a)SIS with BPCS, (b) BPCS alone

a- Redundancy and Reliability:

- BPCS may include some redundancy to improve availability and minimize downtime (e.g., redundant controllers or communication links). However, this redundancy is aimed at maintaining operational performance rather than ensuring safety.
- SIS incorporates redundant components and fault-tolerant architectures (e.g., dual CPUs, redundant I/O modules) to maintain safety functions even in the event of component failures. This ensures that safety functions remain operational when they are needed.

b- Risk Reduction:

- BPCS offers minimal or no risk reduction for safety; its focus is on optimizing control performance and process efficiency.

- SIS provides substantial risk reduction by implementing Safety Instrumented Functions (SIFs) that detect and respond to hazardous conditions. Each SIF is designed to meet a specific Safety Integrity Level (SIL), which indicates the required level of risk reduction.

c- Programming and Configuration:

- BPCS uses standard automation programming tools and languages, such as ladder logic or function blocks, with fewer restrictions on changes to logic.
- SIS uses specialized safety programming tools and often requires safety-certified engineers for configuration. The programming environment includes restrictions, such as password-protected changes and safety-rated function blocks, to prevent unauthorized or unsafe modifications.

d- Integration:

- BPCS is often tightly integrated with human-machine interfaces (HMIs) and other plant control systems for real-time monitoring, control, and data logging.
- SIS may be integrated with the BPCS but operates independently to ensure that safety functions are not compromised by faults in the control system. It may share information with the BPCS, such as status updates, but maintains independence to ensure reliable safety performance.

2.2 How a Safety Instrumented System Works:

- The SIS continuously monitors the process for any deviations from safe operating conditions.

- When a deviation is detected by the sensors, the information is sent to the logic solver.

- The logic solver evaluates the inputs against the predefined safety logic and decides whether a safety response is required.

- If necessary, the logic solver sends a signal to the final control elements to take action, such as shutting down equipment, venting excess pressure, or isolating part of the process.

3. Safety Integrity Levels (SIL):

Safety Instrumented Systems are designed to meet specific Safety Integrity Levels (SIL), which are standards defined by the probability of a safety function failing on demand. The SIL level indicates the level of risk reduction provided by the SIS:

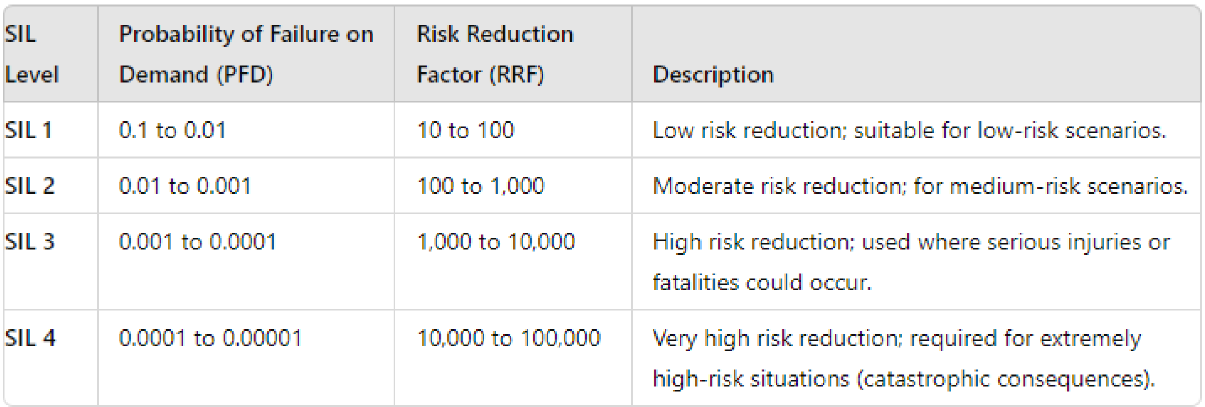
Safety Integrity Levels (SIL) are a measure of the risk reduction provided by a Safety Instrumented System (SIS). SIL levels are part of standards such as IEC 61508 and IEC 61511, which define the requirements for functional safety in industrial processes. The SIL level indicates the reliability of a safety function in performing its intended action when a dangerous condition arises.

SIL quantify the probability that a safety-related system will satisfactorily perform its safety functions under all stated conditions within a specified period of time. Each SIL level represents a range of risk reduction, from SIL 1 (lowest) to SIL 4 (highest).

The required SIL level for a safety function is determined through a **risk assessment** process, where potential hazards are identified, and the associated risks are evaluated. The SIL level represents how much risk reduction is needed to bring a process's risk to an acceptable level.

Here is a table summarizing the **Safety Integrity Levels**, their corresponding Probability of Failure on Demand (PFD), and Risk Reduction Factor (RRF).

Table V.1 : Safety Integrity Levels



SIL Level	Probability of Failure on Demand (PFD)	Risk Reduction Factor (RRF)	Description
SIL 1	0.1 to 0.01	10 to 100	Low risk reduction; suitable for low-risk scenarios.
SIL 2	0.01 to 0.001	100 to 1,000	Moderate risk reduction; for medium-risk scenarios.
SIL 3	0.001 to 0.0001	1,000 to 10,000	High risk reduction; used where serious injuries or fatalities could occur.
SIL 4	0.0001 to 0.00001	10,000 to 100,000	Very high risk reduction; required for extremely high-risk situations (catastrophic consequences).

4. Key Terms Related to SIL:

- **Probability of Failure on Demand (PFD):** The likelihood that a safety function will fail when it is needed.
- **Risk Reduction Factor (RRF):** The inverse of the PFD, representing the amount of risk reduction provided by the safety function.
- **Mean Time to Failure (MTTF):** The average time expected until a failure occurs.
- **Mean Time to Repair (MTTR):** The average time required to repair a failed safety function.
- **The MTBF** is a basic measure of the reliability in repairable items of a piece of equipment. It may be expressed in hours or years. It is commonly used in systems reliability and sustainability analysis.
- MTBF: can be calculated by the following formula:
- **MTBF =** MTTR + MTTF

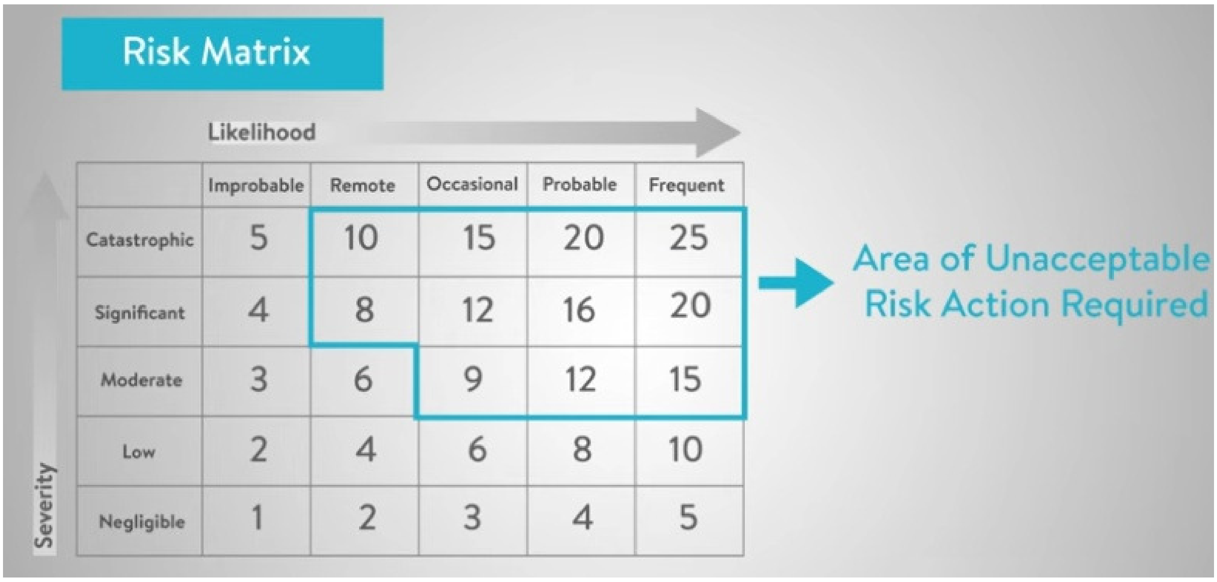
A **Risk Matrix** or **Layer of Protection Analysis (LOPA)** is often used to assess these factors and determine the appropriate SIL level.

5. SIL Verification and Validation:

Once the required SIL level is established, the safety function must be designed and tested to meet the SIL requirements:

- **Verification**: Ensures that the safety function design meets the SIL requirements by calculating the PFD and assessing if it falls within the acceptable range for the intended SIL level.
- **Validation**: Involves testing and evaluating the safety function to confirm that it performs correctly under all anticipated operating conditions and that it meets the required SIL level.

Table V.2 : Risk matrix



Risk Matrix

Likelihood

Severity	Improbable	Remote	Occasional	Probable	Frequent
Catastrophic	5	10	15	20	25
Significant	4	8	12	16	20
Moderate	3	6	9	12	15
Low	2	4	6	8	10
Negligible	1	2	3	4	5

Area of Unacceptable Risk Action Required

5. MooN Voting Logic Concept :

M out of N is a concept used in Safety Instrumented Systems (SIS) to describe the level of redundancy and voting logic required to achieve a certain level of safety. The "MooN" terminology represents how many out of a total number of elements (sensors, controllers, actuators, etc.) must agree or operate correctly for the system to take a particular action, such as triggering an emergency shutdown or maintaining normal operation.

- **MooN** stands for **M out of N**, where:
  - **M** = The minimum number of components that must be in agreement or functioning correctly for the system to take action.
  - **N** = The total number of redundant components in the system.

For example, in a **2oo3 (2 out of 3)** configuration:

- Out of 3 redundant components (such as sensors), at least 2 must detect the same condition (e.g., a high pressure) to trigger a safety response (e.g., shutting down a valve).

This table highlights the trade-offs between reliability, safety integrity, and nuisance trip likelihood for different MooN configurations.

Table V.3 : Common MooN Configurations in SIS.

Configuration	Description	Reliability	Safety Integrity	**Nuisance Trip Likelihood**
1oo1 (1 out of 1)	A single component is used to detect a hazardous condition.	Low	Lowest, as failure of the single component results in loss of safety function.	High
1oo2 (1 out of 2)	Two components are used; only one needs to detect a fault for the system to act.	Moderate	Higher than 1oo1; one failure can be tolerated.	Moderate
2oo2 (2 out of 2)	Two components are used; both must detect the same fault for the system to act.	High	Very high; both components need to fail or detect a hazard simultaneously.	Low
2oo3 (2 out of 3)	Three components are used; at least two must detect the fault for the system to act.	High	High; can tolerate one failure without triggering a shutdown.	Low
1oo3 (1 out of 3)	Three components are used; only one needs to detect a fault for the system to act.	Very High	Very high; single detection is sufficient.	Higher
3oo4 (3 out of 4)	Four components are used; at least three must detect the fault for the system to act.	Very High	Extremely high; can tolerate up to one failure.	Low

6. Choosing the Right MooN Configuration

The choice of MooN configuration in an SIS depends on several factors:

- **Risk Tolerance**: Higher-risk processes may require a more fault-tolerant configuration like **2oo3** to balance safety and reliability.
- **Safety Integrity Level (SIL)**: The required SIL determines the needed level of redundancy and fault tolerance.
- **Cost and Complexity**: More redundant components increase costs and complexity. For example, a **3oo4** configuration requires four components, which might be cost-prohibitive for some applications.
- **Process Sensitivity**: Processes that are sensitive to nuisance trips may favor configurations like **2oo3** or **3oo4** to minimize false alarms.

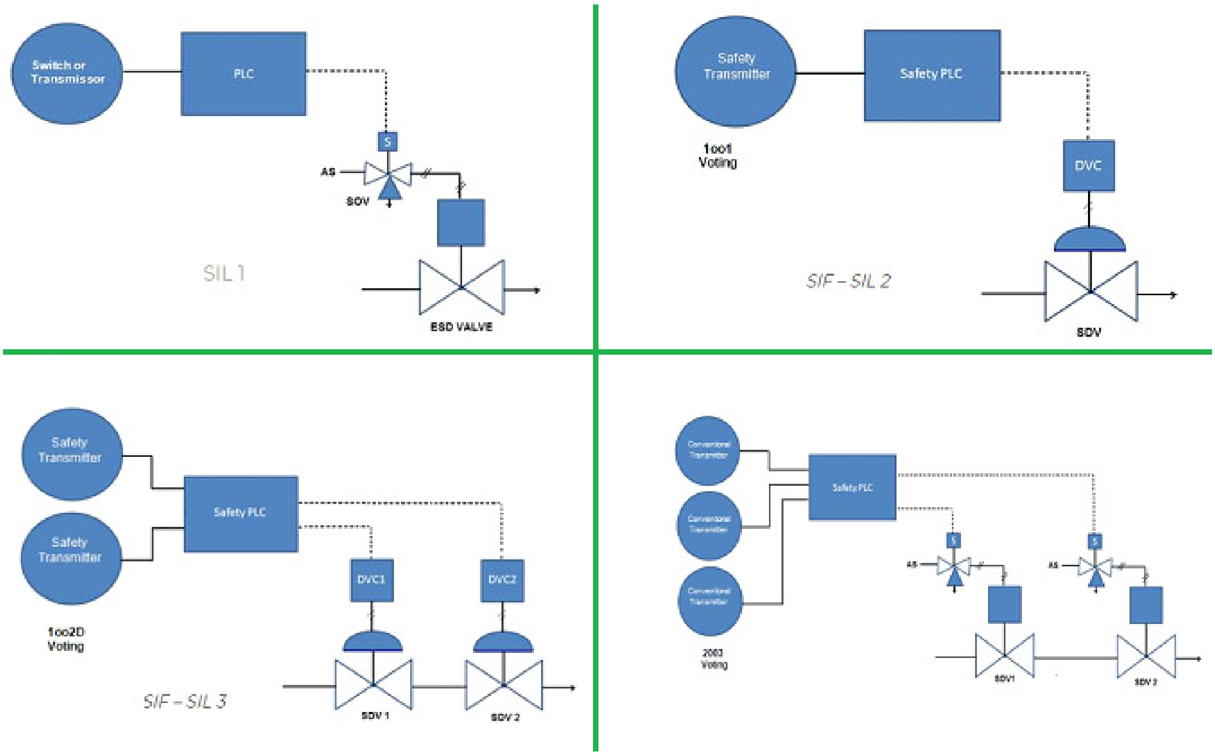


Figure V.5 : Architecture Examples

7. Modeling the Probability of Failure on Demand (PFD) for Safety Instrumented Systems (SIS) involves calculating the average likelihood that a safety function will fail to perform correctly when needed. This process considers factors such as the failure rates of components, testing intervals, system architecture, and redundancy.

Key Factors in Modeling PFD for SIS

a) **Failure Rate of Components (λ)**:

- Each component (e.g., sensors, logic solvers, actuators) in the SIS has a certain failure rate, typically measured in failures per hour.
- Failures can be categorized as **safe failures** (where the system fails into a safe state) or **dangerous failures** (where the system fails to respond correctly to a hazardous condition).

b) **Test Intervals (T)**:

- The **proof test interval (T)** is the period between scheduled tests or inspections to detect hidden failures. The longer the interval, the higher the probability that a dangerous failure may remain undetected.

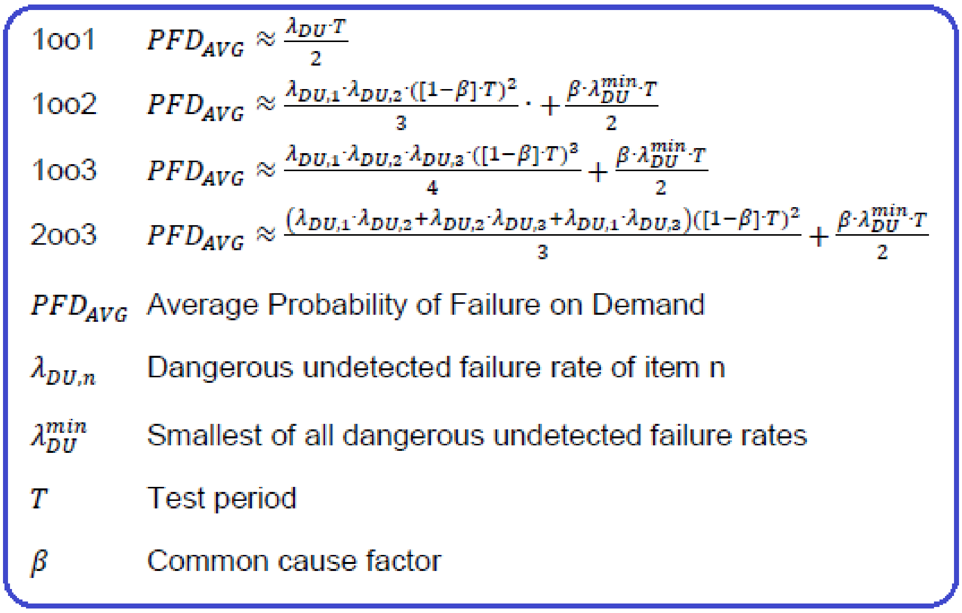
c) **System Architecture and Redundancy (MooN Configuration)**:

- Different system architectures (like 1oo2, 2oo3) have different levels of redundancy, which impact the probability of failure.
- For example, a 1oo2 (1 out of 2) system may have a higher probability of spurious trips but lower PFD than a 2oo2 (2 out of 2) system.

The simple calculation of PFD are based on the following formulas illustrated in Table V.4.

For a simple SIS configuration, the PFDavg can be approximated using the following formulas in table bellow:

Table V.4 : PFDavg for a differents MooN voting configuration.



1oo1	$PFD_{AVG} \approx \frac{\lambda_{DU} \cdot T}{2}$
1oo2	$PFD_{AVG} \approx \frac{\lambda_{DU,1} \cdot \lambda_{DU,2} \cdot ([1-\beta] \cdot T)^2}{3} \cdot + \frac{\beta \cdot \lambda_{DU}^{min} \cdot T}{2}$
1oo3	$PFD_{AVG} \approx \frac{\lambda_{DU,1} \cdot \lambda_{DU,2} \cdot \lambda_{DU,3} \cdot ([1-\beta] \cdot T)^3}{4} + \frac{\beta \cdot \lambda_{DU}^{min} \cdot T}{2}$
2oo3	$PFD_{AVG} \approx \frac{(\lambda_{DU,1} \cdot \lambda_{DU,2} + \lambda_{DU,2} \cdot \lambda_{DU,3} + \lambda_{DU,1} \cdot \lambda_{DU,3})([1-\beta] \cdot T)^2}{3} + \frac{\beta \cdot \lambda_{DU}^{min} \cdot T}{2}$
PFD_{AVG}	Average Probability of Failure on Demand
$\lambda_{DU,n}$	Dangerous undetected failure rate of item n
λ_{DU}^{min}	Smallest of all dangerous undetected failure rates
T	Test period
β	Common cause factor

Figure V.6 shows a simple process where a fluid is added in a continuous and automatic way to a process vessel. If the control system fails due to a very high pressure condition, a safety relief occurs, producing an undesirable smell out of the plant. Na acceptable risk rate for such event is 0.01/year or less (once every one hundred years or 1 change in 100 per year). Let's specify a Safety Instrumented System (SIS) reaching such safety requirements.

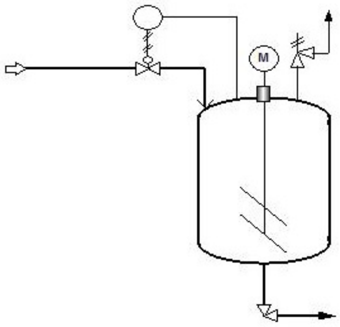


Figure V.6 : Process with basic control

In order to define the safety integrity requirements, the demand rate regarding the SIS should be estimated. In such example, the SIS demand rate should be the dangerous failure rate of the control loop.

The overall failure rate for the control loop may be estimated from the failure rates for components, which, in the example, we will assume as:

	Failures/year
Pressure transmission	0.6

Controller	0.3
I/P	0.5
Control valve	0.2
Failure overall	1.6

The control loop for such example may fail in any direction, assuming that both are equally probable. Since the active control loop is under the operator supervision, it is assumed that only 1 failure out of 4 would be suddenly enough to cause a demand for a shut down condition without a previous intervention by the operator. That causes the overall result of (1 out of 2) x (1 out of 4) or 1/8 of overall failure rate, which should be used as the demand rate for a stop. Different assumptions should be made based on the specific knowledge on the equipment and conditions.

Therefore, the demand rate = 1.6/8 = 0.2/year

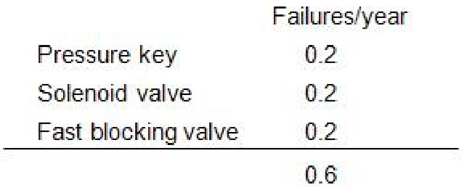
The acceptable unavailability =

$$\frac{\text{Risk Rate}}{\text{Demand Rate}} = \frac{0{,}01}{0{,}2} = 0{,}05$$

The required availability is = 1-0.05 = 0.95

SIS with simple and direct connection is proposed to cut the supply when the system pressure reaches 80% of the safety valve setting value.

Compliance should be evaluated by the estimation of loop unavailability. Follows below the failure rates, showed as example and that could be consulted by each manufactured:



	Failures/year
Pressure key	0.2
Solenoid valve	0.2
Fast blocking valve	0.2
	0.6

The loop is designed to fail in the safe direction, therefore, it is admitted that only 1 out of 3 failures would be in the unsafe direction. All those passive system failures would not be diagnosed.

Therefore, the non-diagnosed failures rate = 0.6/3 = 0.2/year

With a test annual frequency,

$$\text{FDT} = \tfrac{1}{2}\ fT = \tfrac{1}{2} \times 0.2/\text{year} \times 1 \text{ year} = 0.1$$

That provides an availability of 0.9, which still does not comply with the safety requirements. However, the availability may be increase with a higher frequency of tests. With monthly tests we have,

$$\text{FDT} = \tfrac{1}{2} \times 0.2/\text{year} \times (1/12) \text{ year} = 0.0083$$

Reaching an availability >0.99. The project test frequency should be specified as part of the project documents.

According to table 1, a SIL 1 system with frequent tests should provide an availability of 0.99 complying with the 95% availability required.

Simplified example of PFD_{avg} calculation :

We want to calculate the PFD_{avg} for a low demand mode Safety Loop involving a pressure transmitter, an analogue barrier, two electronic safety modules and the STO (Safe Torque off) of a Variable Speed Drive.

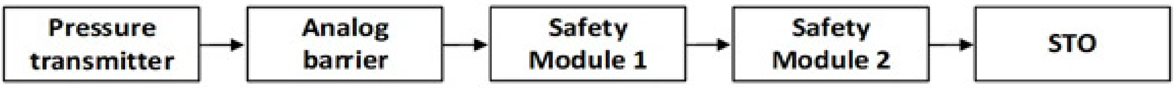


Figure V.7 : example of safety instrumental function

For a low demand mode, the total PFD_{avg} can be approximated as the sum of the PFD_{avg} values for each component, assuming they are in series.

$$\text{PFD}_{avg} \approx \frac{\lambda_{PT} \times T}{2} + \frac{\lambda_{AB} \times T}{2} + \frac{\lambda_{SM1} \times T}{2} + \frac{\lambda_{SM2} \times T}{2} + \frac{\lambda_{STO} \times T}{2}$$

λ_{PT}, λ_{AB}, λ_{SM1}, λ_{SM2} and λ_{STO} are the dangerous failure rates for each component.

We need to know the safety data from the technical data sheet of each component; for example, we need to know λ_{DU}, *SFF* and component type. It is then necessary to establish the structure of each subsystem and the proof test period Ti.

Bellow are examples of data:

Component	λ_{DU}	SFF	Type	Arch. Constr.	Structure	T_i
Pressure Transmitter	$9{,}9\cdot10^{-8}$	87 %	B	2	1oo1	10 years
Analogue Barrier + Converter	$9{,}13\cdot10^{-8}$	93 %	B	2		
Safety Module	$1{,}14\cdot10^{-11}$	99 %	A	3		
STO / VSD	$3{,}65\cdot10^{-9}$	99,66 %	A	3		

The PFD_{avg} of the 1oo1 structure can be calculated with the following formula, resulted from the integral in *par. 2.2 Terms and definitions from 61508-4*

The SIL architecture of each subsystem has been fixed using table 2 and table 3 of the so called Route 1_H.

Component	PFD_{Avg}
Pressure Transmitter	$4{,}34\cdot10^{-3}$
Analogue Barrier + Converter	$4\cdot10^{-3}$
Safety Module 1	$5\cdot10^{-7}$
Safety Module 2	$5\cdot10^{-7}$
STO / VSD	$1{,}6\cdot10^{-4}$
TOTAL (SIL 2)	**$8{,}5\cdot10^{-3}$**

PFD_{avg_TOT} of the loop is the sum of the PFD_{avg} of each component. It is $10^{-2} < 8{,}5\text{x}10^{-3} < 10^{-3}$

The system reaches SIL 2.

8. Internal Architecture of Safety PLC

The fundamental difference between a general-purpose programmable logic controller (PLC) and a safety PLC can be summarized in one word: "diagnostics." Although there are differences in terms of internal architecture, software, and certification, the safety PLC is distinguished from traditional PLCs by the integration of numerous diagnostic functions that detect any potential internal failures in the hardware or firmware, ensuring that a PLC failure does not cause any "dangerous" situation.

Nowadays, even though standard PLCs incorporate diagnostic functions, they have far fewer than safety PLCs. This diagnostic capability of the programmable controller reduces undetected dangerous faults, thereby reducing the probability of failure, which is one of the requirements of the Safety Integrity Level (SIL) standard that determines the safety level of a device as detailed in previously .

To simplify, let’s take a small example to explain the difference between a standard PLC and a safety PLC: suppose a digital output of a PLC that controls a solenoid valve is burnt out. This could constitute a dangerous fault because if the output controlling the solenoid valve fails, the valve will not close, which could create a dangerous situation.

So, how do we avoid such a situation? This is where safety PLCs come in. To detect the failure of the digital output, the safety PLC introduces a diagnostic routine using micro-pulses and reads the output state. In case of failure, the PLC firmware triggers an alarm.

To enhance the safety level of the system, redundant architectures can be used, such as duplicating the control circuit of the solenoid valve by using two CPUs mounted in parallel.

There are many diagnostic functions in a safety PLC, both in the CPU and in the memories, as well as at the input and output levels. This is why safety PLCs are relatively expensive, as these additional functions logically incur extra costs.

However, keep in mind that statistically, most system faults originate much more from sensors and actuators than from the PLC itself. International standards classify applications based on their risk level: SIL-1, SIL-2, SIL-3, and SIL-4 (Safety Integrity Level) and are part of the risk analysis that must be done during the design of the automated system.

In practice, safety PLCs are used to connect emergency stop buttons, light curtains, and other safety devices. In fact, it is a mistake to use a standard PLC to connect safety devices because the risk it could pose is high. In process industries, such as oil and gas or nuclear, the required safety level is very high. These systems will obviously require more safety devices.

Today, there are many safety PLCs on the market. Examples include Siemens Fail-Safe systems, Sick's Flexi-Soft, Rockwell Automation's GuardLogix, Triconex safety PLCs, and Pliz. These safety PLCs are often yellow or red in color.

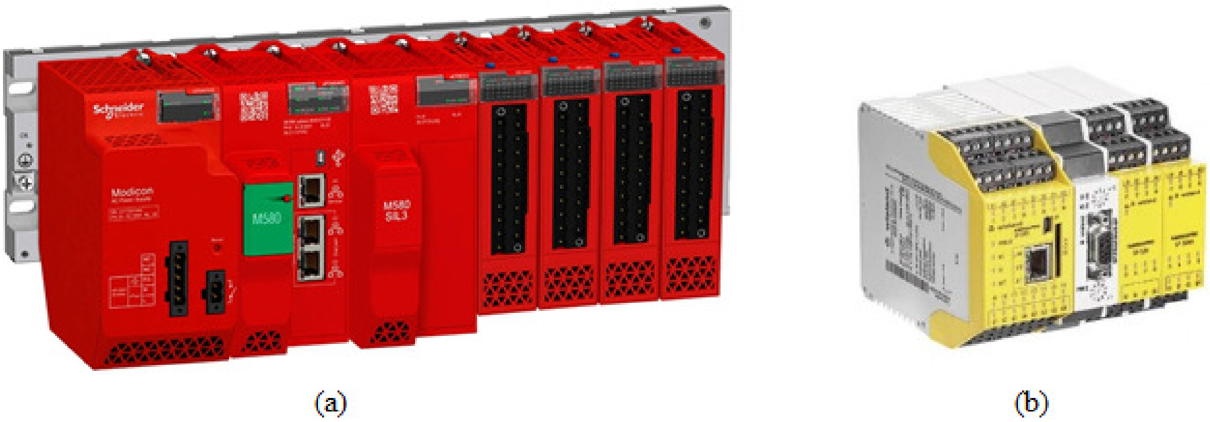


Figure V.7 : Safety PLCs (a) Modicon M580 the new safety PLC from Schneider, (b)WIELAND ELECTRIC compact safety automation systems

9. Internal architecture of a safety PLC microprocessor

The internal architecture of a safety PLC microprocessor is designed to provide high reliability and safety by detecting and mitigating potential faults. Unlike a standard PLC, a safety PLC is specifically engineered to meet stringent safety standards (such as IEC 61508, IEC 62061, or ISO 13849) and includes features to ensure that failures do not lead to dangerous situations.

Here’s an overview of the key components and architectural elements commonly found in a safety PLC microprocessor:

1) Dual or Redundant Processors

- Dual-Core or Dual-CPU Architecture: Safety PLCs often utilize two or more independent processors that run in parallel. These processors execute the same safety program simultaneously and continuously cross-check each other's operations. If any discrepancy is detected, the system can safely shut down or switch to a safe state.

- Lockstep Operation: In some safety PLCs, the processors work in lockstep, meaning that they perform identical operations at the same time and their outputs are continuously compared to detect any errors.

2) Memory Protection and Error Checking

- Error-Correcting Code (ECC) Memory: Safety PLCs use ECC memory to detect and correct single-bit errors and detect multi-bit errors in the data stored in memory. This reduces the risk of data corruption leading to dangerous failures.

- Memory Segmentation and Protection: The internal architecture includes hardware-based memory protection mechanisms that prevent accidental overwriting or corruption of critical safety data and program code.

3) Dedicated Safety I/O Modules

- Safe Input/Output Processing Units: The microprocessor interfaces with dedicated safety input/output modules that are designed with built-in redundancy and diagnostics. These modules check the integrity of the input signals (e.g., emergency stop, light curtains) and output signals (e.g., safety relays) to ensure they are functioning correctly.

- Input and Output Checks: Safety PLCs constantly monitor the status of inputs and outputs using diagnostic routines, such as pulse testing or heartbeat signals, to ensure that they are operating correctly.

4) Diagnostic and Monitoring Functions

- Watchdog Timers: Safety PLCs use hardware watchdog timers to monitor the execution of programs. If the processor fails to complete its tasks within a predefined timeframe, the watchdog triggers a safe shutdown.

- Self-Test Routines: The microprocessor performs regular self-diagnostic checks, including memory tests, arithmetic logic checks, and register integrity tests, to detect internal faults.

- Cyclic Redundancy Checks (CRC): CRC checks are used to verify the integrity of both software and data in transit. This prevents data corruption from affecting the safety-critical processes.

5) Redundant Communication Interfaces

- Safety Communication Protocols: Safety PLCs use specialized communication protocols (e.g., PROFIsafe, SafetyNET p, Ethernet/IP with CIP Safety) to ensure secure data transmission between components. These protocols include features like message sequencing, checksums, and time-stamped data to detect and handle communication errors.

- Redundant Network Paths: Multiple network paths may be provided for communication redundancy, ensuring that if one path fails, another can maintain the communication without loss of critical safety data.

6) Hardware Security Features

- Tamper Detection: The microprocessor often includes tamper detection mechanisms that monitor for unauthorized changes to safety-critical settings or configurations.

- Cryptographic Modules: Some safety PLCs incorporate cryptographic modules to authenticate firmware and prevent the execution of unauthorized code.

7) **Fail-Safe Mechanisms**

- Safety Relays and Safe Outputs: The architecture includes safety relays and safe outputs that ensure, even in the case of a processor failure, the system will transition to a safe state, such as shutting down machinery or activating brakes.
- Safe State Logic: In case of fault detection, the microprocessor is programmed to bring the system to a predetermined "safe state" to minimize risk.

8) **Deterministic Real-Time Operating System (RTOS)**

- Real-Time Scheduling: A deterministic RTOS ensures that safety-critical tasks are executed with precise timing to meet strict safety requirements. The RTOS may prioritize safety tasks over non-safety tasks to ensure prompt and reliable response to critical events.
- Separation Kernel: Some safety PLCs use a separation kernel architecture that provides strong isolation between different tasks and memory spaces, reducing the risk of faults propagating from one task to another.

9) Redundant Power Supply and Power Management

- Dual Power Supply: Safety PLCs often include redundant power supplies to maintain operation in case of power failure.
- Voltage Monitoring: The microprocessor continuously monitors voltage levels and power integrity to ensure stable operation. Any abnormality in power supply triggers an alarm or initiates a safe shutdown.

10) Compliance with Safety Standards

- Certified Safety Components: All components in a safety PLC, including the microprocessor, must be certified according to safety standards like IEC 61508 or ISO 13849. This certification involves rigorous testing to verify that the design meets the required safety integrity levels (SIL).

References

1. Frank D. Petruzella, Programmable Logic Controllers, 4th edition, Ed. Mc Graw Hill 2004.

2. William Bolton, Les automates programmables industriels, Editions Dunod, l'Usine Nouvelle, 2010.

3. Ian G. Warnock, Programmable Controllers: Operation and Application, Prentice Hall.

4. Gilles Michel, Architecture et applications des automates programmables industriels, Dunod.

5. G. Michel, Automates Programmables industriels, Dunod, 1979.

6. S. Thelliez et J.M.Toullote, Grafcet et logique industrielle programmée, Eyrolles, 1980.

7. J.C Bossy, P. Brard, P. Faugère, C. Merlaud, Le Grafce : sa pratique et ses applications, Educalivre Ed. Casteilla, 1995.

8. Henri Ney, Eléments d'automatismes, Collection Electrotechnique et normalisation, Edition Nathan, 1996.

9. M. Diaz, Les Réseaux de Pétri - Modèles fondamentaux. Traité IC2 - Série Informatique et Systèmes d'Information, Hermès Science 2001

10. A. Choquet-Geniet, Les réseaux de Pétri – Un outil de modélisation, Dunod, 2006.

11. P. Ladet, Outils de modélisation des automatismes séquentiels, Les réseaux de Pétri, Techniques de l'ingénieur, 1990.