POLITECNICO DI MILANO





Automatic Control

Control system technologies for automation

Control system design, communication systems, Programmable Logic Controllers

Prof. Luca Bascetta (luca.bascetta@polimi.it)

Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria

Motivations (I)

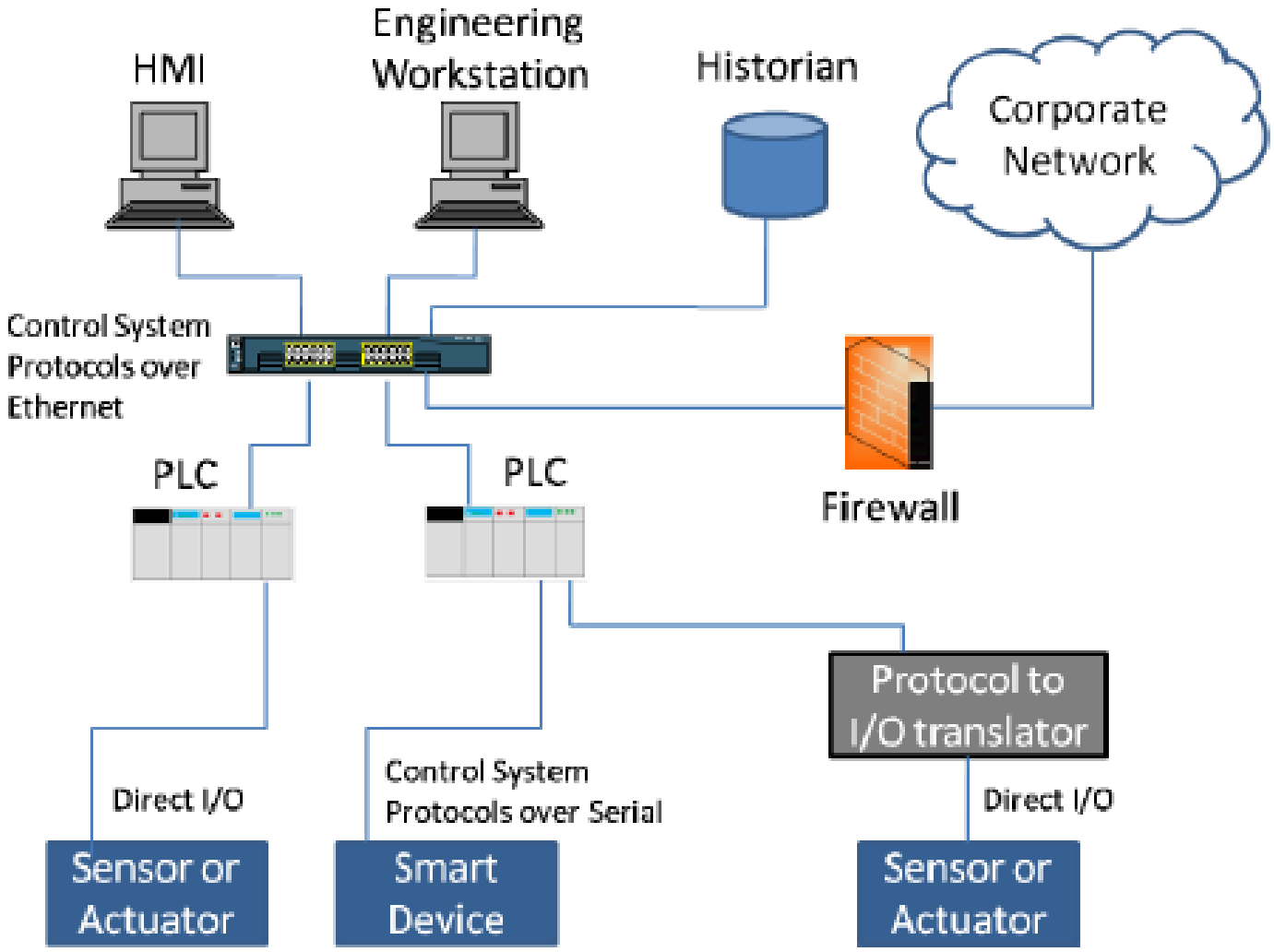
In the design and realization of factory automation and process control systems many aspects should be considered, ranging from the design of the system architecture to the selection of the communication network, and the suitable hardware and software platforms.

Even focusing only on the control system, in a complex plant we must control either continuous time physical quantities and action sequencing.

For these reasons we will face the fundamental aspects of:

- communication systems
- fieldbus network systems
- Programmable Logic Controllers
- Real-time systems

Motivations (II)



Communication systems (I)

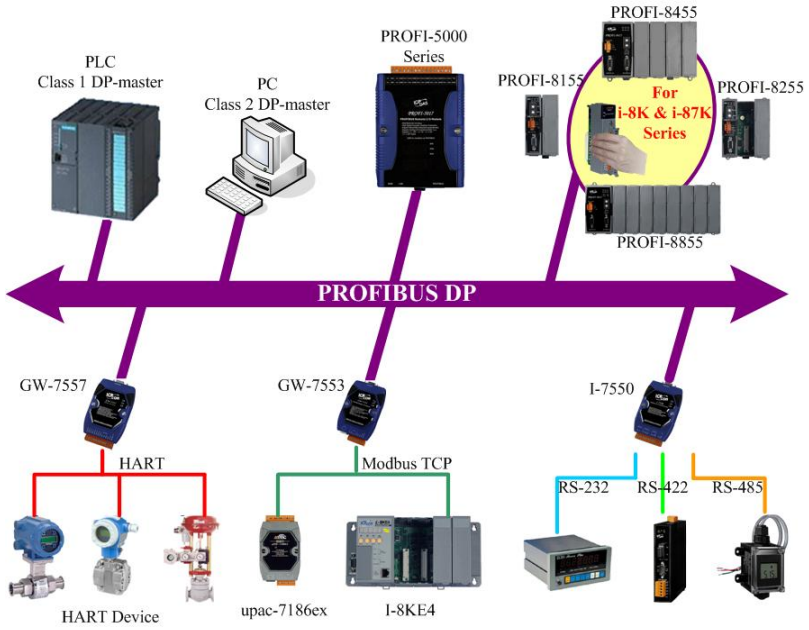
In a control system composed by a control unit and a few sensors and actuators, communication can be performed using analog voltage/current signals.

What happens, however, if we have many control units connected to many sensors/actuators?

Instead of using an analog point-to-point connection, we can choose a digital bus communication system connecting all the sensors/actuators to one or more control units.

Using this communication system has obvious advantages:

- simpler and less expensive cabling
- flexibility
- adding/removing devices is easier
- resource sharing
- redundancy
- distributed functionalities



Communication systems (II)

There are also some disadvantages of using bus communication systems:

- devices are more expensive
- it is not easy to connect together devices from different manufacturers
- more complex design methodologies

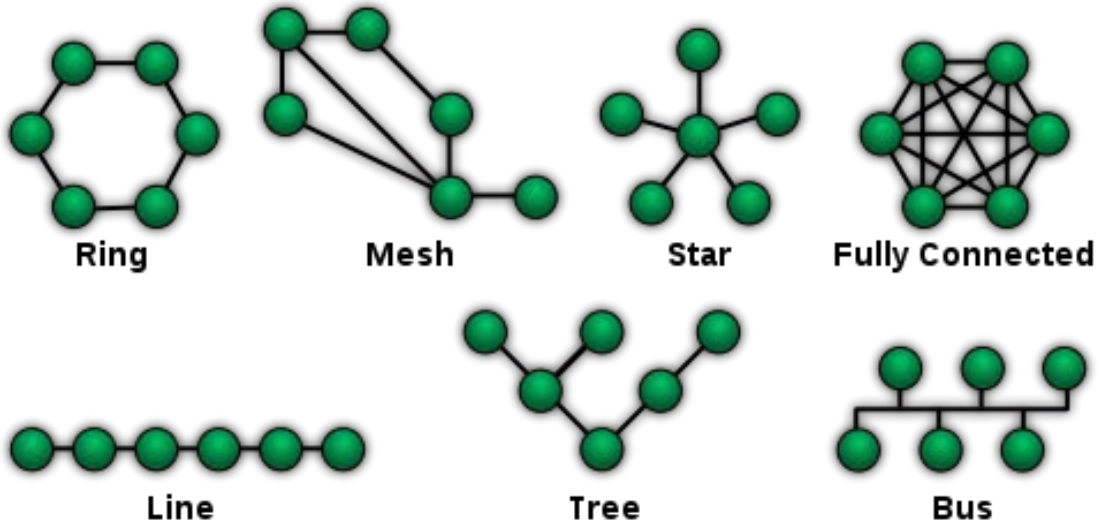
We will now first introduce some fundamentals of communication networks, and then analyze the most important buses used in control systems.

Communication systems (III)

A computer network is a system constituted by two or more computers exchanging data through a communication network.

One important characteristic of a network is its topology, the layout or organizational hierarchy of the interconnected nodes.

Different network topologies exist, each one characterized by a different throughput and reliability. In general the more interconnections there are, the more robust the network is; but the more expensive it is to install.



Communication systems (IV)

Another important characteristic is the network protocol, the set of rules used for exchanging information over network links.

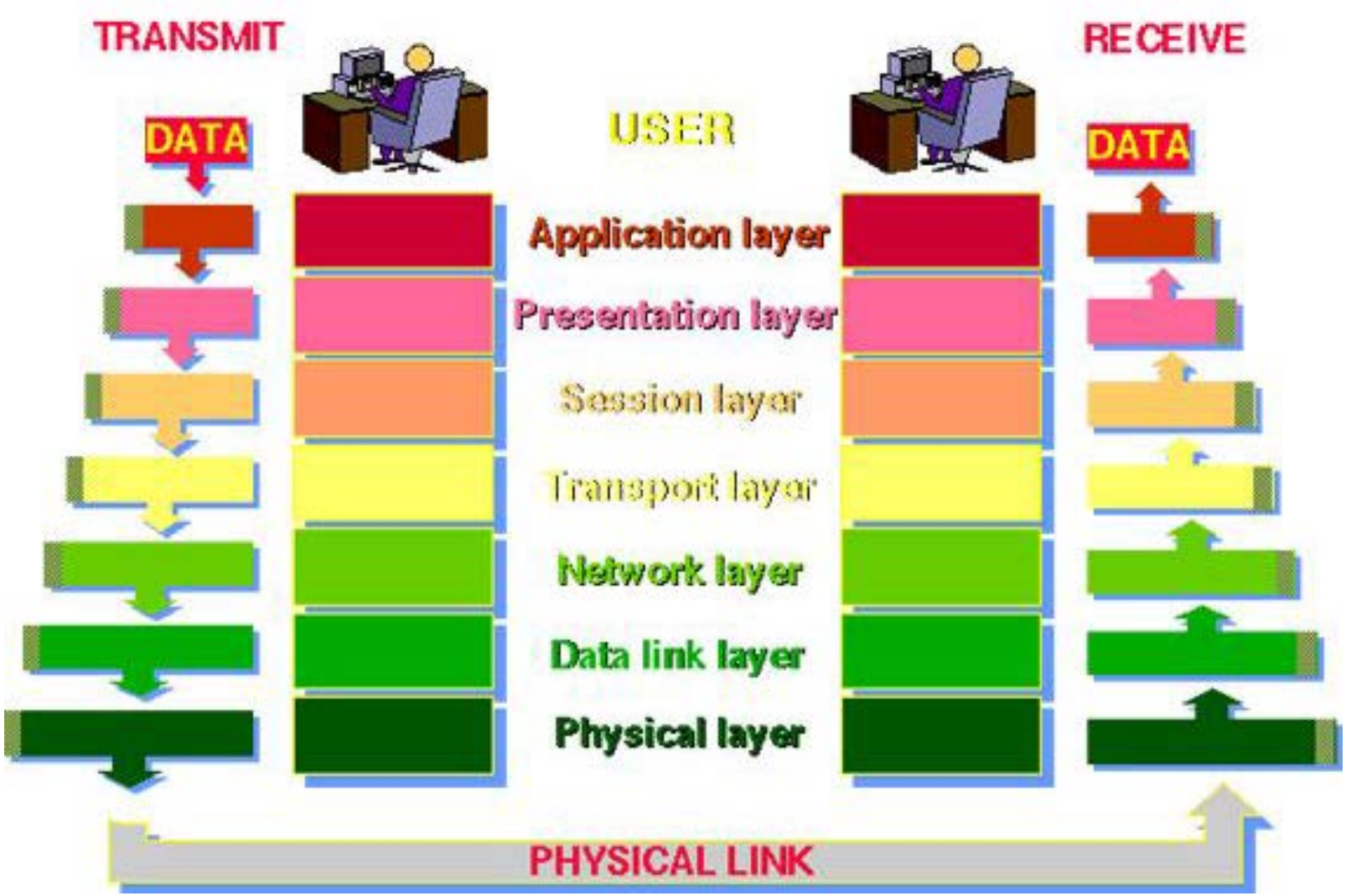
Protocols are organized in a hierarchical way in a protocol stack. In this way each protocol leverages the services of the protocol below it.

A conceptual model that standardizes the communication functions of a communication network, without regard to their underlying internal structure and technology, is the Open Systems Interconnection model (OSI model).

The model partitions a communication system into seven abstraction layers. A layer serves the layer above it and is served by the layer below it.

The model is a product of the Open Systems Interconnection project at the International Organization for Standardization (ISO), maintained by the ISO/IEC 7498-1 standard.

Communication systems (V)



Communication systems (VI)

OSI layer	Protocol Data Unit	Function
Application	Data	High-level APIs, including resource sharing, remote file access, directory services and virtual terminals
Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
Transport	Segment (TCP) Datagram (UDP)	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
Physical	Bit	Transmission and reception of raw bit streams over a physical medium

Not all the seven layers need to be implemented.

In the case of control systems the most used levels are: physical, data link and application.

Communication systems – Example

Let’s use the OSI model to describe the Internet protocol.

Layer	Implementation
Application	Internet protocol suite (telnet, FTP, SMTP, HTTP,…)
Presentation	
Session	
Transport	TCP (Transmission Control Protocol) UDP (User Datagram Protocol)
Network	IP (Internet protocol)
Data link	Not specified (usually Ethernet)
Physical	

Let’s know analyze the functionalities of the layers that are most important in control systems.

Communication systems (VII)

Physical layer

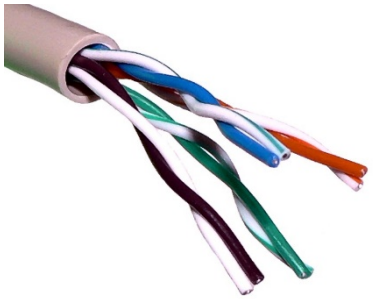
The physical layer has the following major functions:

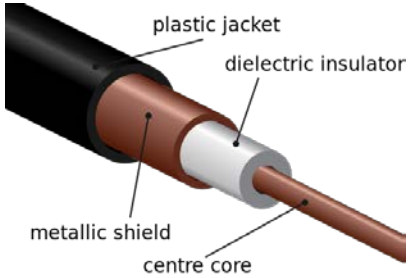
- it defines the electrical and physical specifications of the data connection
- it defines the relationship between a device and a physical transmission medium (e.g., a copper or fiber optical cable, radio frequency). This includes the layout of pins, voltages, line impedance, cable specifications, signal timing and similar characteristics for connected devices, and frequency (5 GHz or 2.4 GHz etc.) for wireless devices
- it defines transmission mode, i.e. simplex, half duplex, full duplex
- it defines the network topology

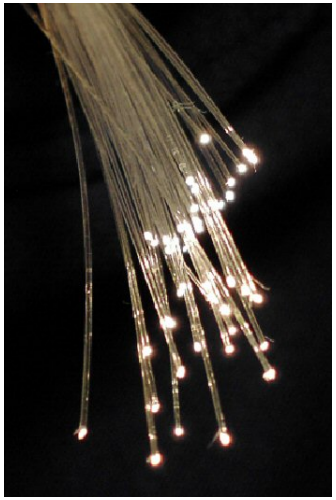
Communication systems (VIII)

Typical transmission medium are:

- twisted pair wires
  - two conductors twisted together for the purpose of canceling out electromagnetic interference
  - can be shielded or unshielded
  - used for telephone, modem lines, Ethernet
  - mid-high speed (up to 100 Mbps in local area networks)
- coaxial cable
  - an inner conductor surrounded by a tubular insulating layer, surrounded by a tubular conducting shield
  - mid-high speed (up to 100 Mbps in local area networks)
- optical fiber
  - transparent fiber made by drawing glass (silica) or plastic to a diameter slightly thicker than that of a human hair
  - permits transmission over longer distances and at higher bandwidths (data rates) than wire cables, but is more expensive







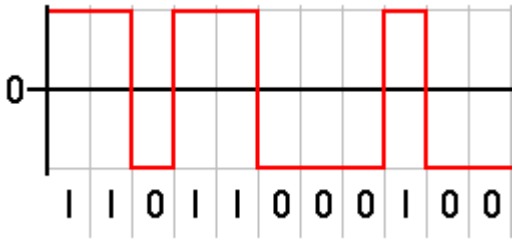
Communication systems (IX)

Bits are transmitted through the physical medium using a line code.

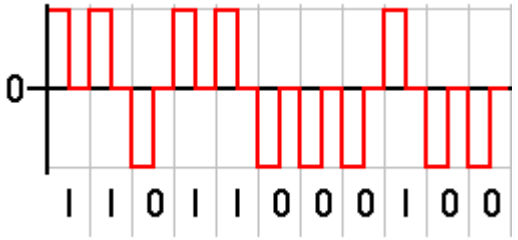
Line coding consists of representing the digital signal to be transported, by a waveform that is optimally tuned for the specific properties of the physical channel.

The common types of line encoding are:

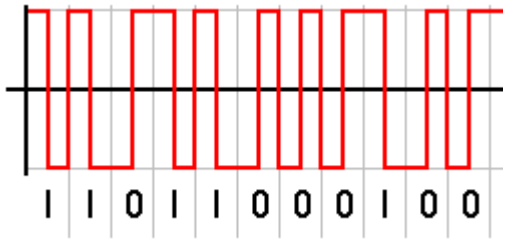
- non-return-to-zero (NRZ), is a binary code in which ones are represented by a positive voltage, while zeros are represented by a negative voltage, with no other neutral or rest condition



- return-to-zero (RZ), the signal drops (returns) to zero between each pulse



- Manchester, the encoding of each data bit is either low then high, or high then low, of equal time (it is self-clocking)



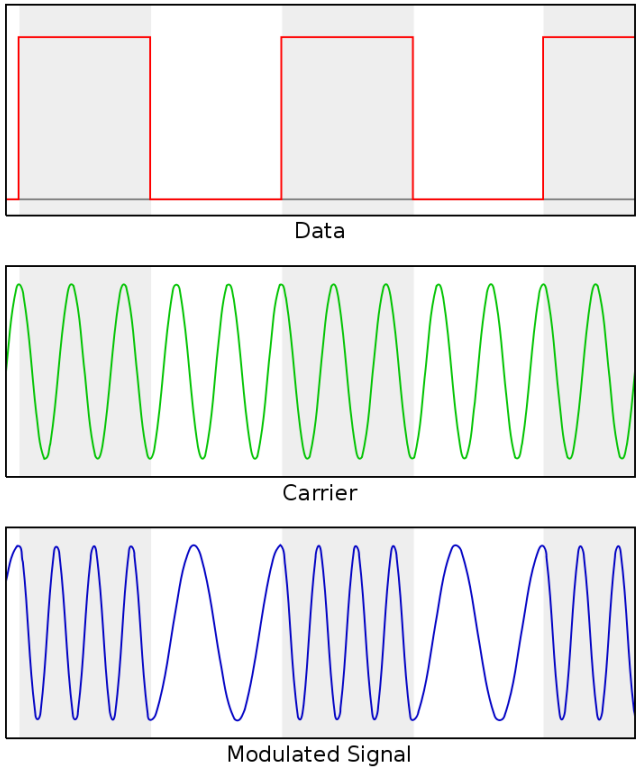
Communication systems (X)

Bits can be transmitted through the physical medium using a frequency modulation technique.

For example the HART protocol uses the FSK method (Frequency Shift Keying) to encode bits.

The two digital values “0” and “1” are assigned to the following frequencies:

- logical “0”, $2200\ Hz$ sinusoidal signal
- logical “1”, $1200\ Hz$ sinusoidal signal



Communication systems (XI)

Let's consider now some examples of electrical specifications:

- EIA RS-232C
  - is one of the oldest standard, but is still in use
  - maximum transmission distance: $15\ m$
  - maximum transmission speed: $20\ kbit/s$
  - 3 wires: GND (ground), TXD (transmit), RXD (receive)
- EIA RS-422
  - maximum transmission speed: $115\ kbit/s$ up to $1200\ m$, or $10\ Mbit/s$ up to $12\ m$
  - differential (instead of single-ended) transmission
  - no more than 10 receivers
- EIA RS-485
  - maximum transmission speed and distance equivalent to RS-422
  - up to 32 transmitters and 32 receivers
  - transmitters can disconnect from the network

Communication systems (XII)

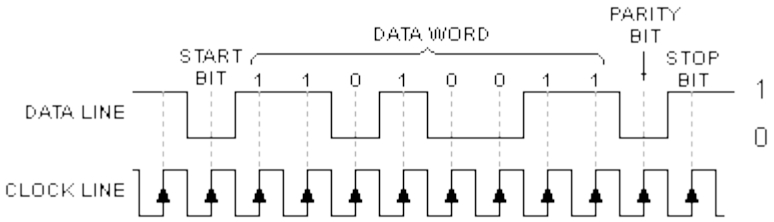
As previously mentioned we can have different transmission modes:

- simplex, data can be sent only through one direction (unidirectional communication)
- half duplex, data can be sent in both directions but it is done one at a time (when the sender is sending data, then at that time the receiver cannot send the sender a message)
- full duplex, data can be sent in both directions simultaneously (bidirectional communication)

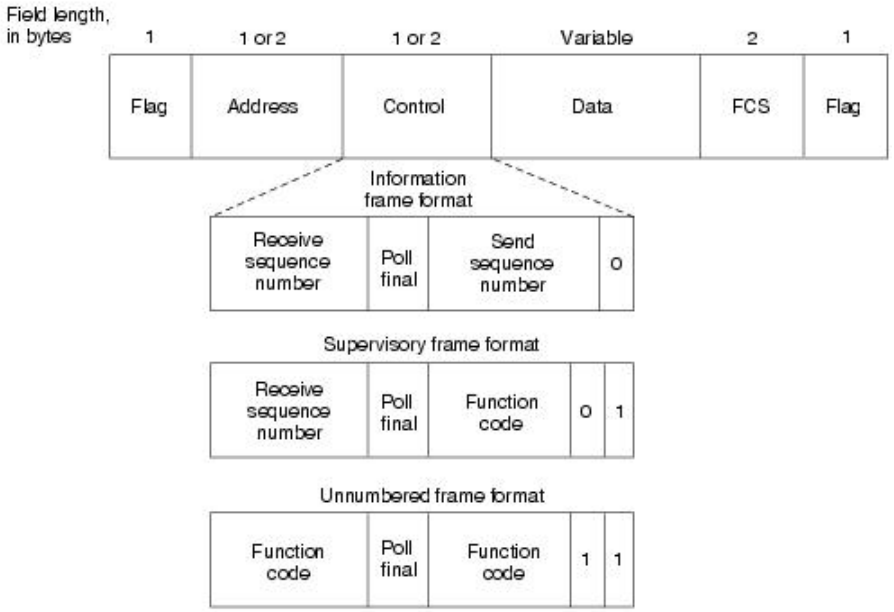
Finally, data transmission can be synchronous or asynchronous.

Communication systems (XIII)

Asynchronous communication is transmission of data blocks, generally without the use of an external clock signal, coded as words of a certain word length (e.g., 8 bytes or ASCII characters).



Synchronous communication requires that the clocks in the transmitting and receiving devices are synchronized, so the receiver can sample the signal at the same time intervals used by the transmitter. No start or stop bits are required. For this reason synchronous communication allows more information to be passed over a circuit per unit time than asynchronous communication.



Communication systems (XIV)

Data link layer

Data link layer provides node-to-node data transfer, and

- it detects and possibly corrects errors that may occur in the physical layer
- it defines the protocol to establish and terminate a connection between two physically connected devices
- it also defines the protocol for flow control between them

IEEE 802 divides the data link layer into two sublayers:

- Media Access Control (MAC) layer, responsible for controlling how devices in a network gain access to medium and permission to transmit
- Logical Link Control (LLC) layer, responsible for identifying Network layer protocols and then encapsulating them. It controls error checking and frame synchronization

The MAC and LLC layers of IEEE 802 networks such as 802.3 Ethernet, 802.11 Wi-Fi, and 802.15.4 ZigBee, operate at the data link layer level.

Communication systems (XV)

The MAC sublayer provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multiple access network that incorporates a shared medium.

Examples of common multiple access protocols for wired networks are:

- CSMA/CD (used in Ethernet and IEEE 802.3)
- Token bus (IEEE 802.4)
- Token ring (IEEE 802.5)
- Token passing (used in FDDI)

Communication systems (XVI)

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

- if the medium is idle any node can start a transmission at any time
- the transmitter monitors for collisions during transmission and, if a collision is detected, the frame is transmitted again
- not suitable for real-time applications as no bound on the transmission delay exists (due to the retransmission mechanism)
- when the protocol is used with fast Ethernet links this issue is less important

Token bus/token ring/token passing

- uses a special frame called a “token” that travels around a logical “ring”
- token passing is a channel access method providing fair access for all nodes, and eliminating the collision problem
- there can be master and slave nodes (slave nodes transmit only when asked by a master node)
- access is more deterministic compared to CSMA/CD

Another way to control the access to the network is by using an arbiter.

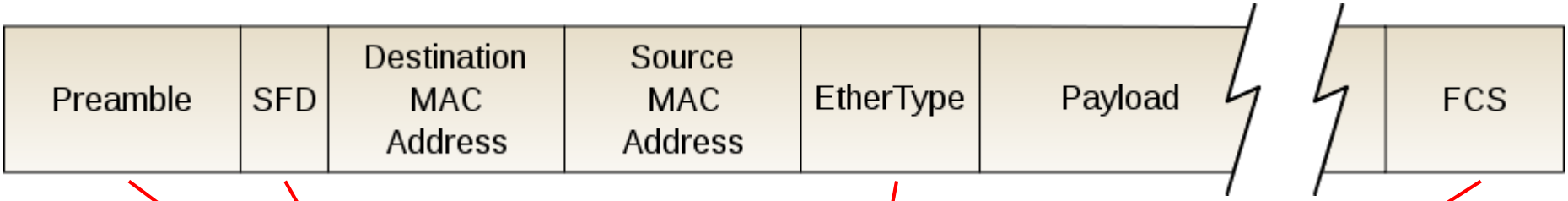
Arbiters are electronic devices that allocate access to shared resources (in this case the physical medium).

Communication systems – Example (I)

Let's consider, as an example, Ethernet.

- it was developed at Xerox PARC between 1973 and 1974
- Ethernet protocol includes levels 1 and 2 of the OSI model
- many different physical medium were used during time
  - coaxial cable with extra braided shielding (thick Ethernet), $10\ Mbit/s$ up to $500\ m$
  - coaxial cable (thin Ethernet), $10\ Mbit/s$ up to $200\ m$
  - twisted pair (twisted pair Ethernet), $10\ Mbit/s$ up to $500\ m$
  - twisted pair (fast Ethernet), $100\ Mbit/s$ up to $100\ m$
  - optical fiber, $10\ Mbit/s$ up to $2\ Km$
  - Gigabit Ethernet, $1\ Gbit/s$ up to $100\ Km$ (single-mode optical fiber)
- signals are transmitted using Manchester encoding
- different network topologies
- MAC layer: CSMA/CD
- LLC layer: frame Ethernet II

Communication systems – Example (II)



Synchronization pattern

Start of Frame Delimiter

Frame Check Sequence: CRC that allows detection of corrupted data

Values of 1500 and below mean that it is used to indicate the size of the payload, while values of 1536 and above indicate that it is used as an EtherType, to indicate which protocol is encapsulated in the payload of the frame

Communication systems – Example (III)

MAC address is the address assigned to each Ethernet card.

It is a 48-bit address composed by:

- a 24-bit Organizational Unique Identifier (OUI), identifies the organization that issued the identifier
- a 24-bit Network Interface Controller (NIC), assigned by that organization in nearly any manner they please, subject to the constraint of uniqueness

Fieldbus (I)

Fieldbus is an industrial network system for real-time distributed control, connecting together sensors, actuators and controllers.

Fieldbus is the equivalent of LAN-type connections, which require only one communication point at the controller level and allow multiple of analog and digital points to be connected at the same time, while 4-20 mA communication requires that each device have its own communication point at the controller level.

Length of cables and number of cables required are both reduced.

Fieldbus is a good solution to send small amount of information having real-time constraints.

There were many competing technologies for fieldbus and the original hope for one single unified communication mechanism has not been realized. This should not be unexpected since fieldbus technology needs to be implemented differently in different applications.

With reference to the OSI model, fieldbus standards are determined by the physical media of the cabling, and layers one, two and seven of the reference model.

Fieldbus (II)

IEC 61158 specification includes eight different standard protocol sets called “Types” as follows:

- Type 1: Foundation Fieldbus H1
- Type 2: ControlNet
- Type 3: PROFIBUS
- Type 4: P-Net
- Type 5: FOUNDATION fieldbus HSE (High Speed Ethernet)
- Type 6: SwiftNet (a protocol developed for Boeing, since withdrawn)
- Type 7: WorldFIP
- Type 8: Interbus

FOUNDATION Fieldbus and PROFIBUS PA are the most common solutions for process control. They share the same physical layer but are not interchangeable.

PROFIBUS DP is a common solution for factory automation, and CAN for automotive applications.

Fieldbus (III)

PROFIBUS (Process Field Bus) is a standard for fieldbus communication in automation technology and was first promoted in 1989 by BMBF (German department of education and research) and then used by Siemens.

PROFIBUS is openly published as part of IEC 61158.

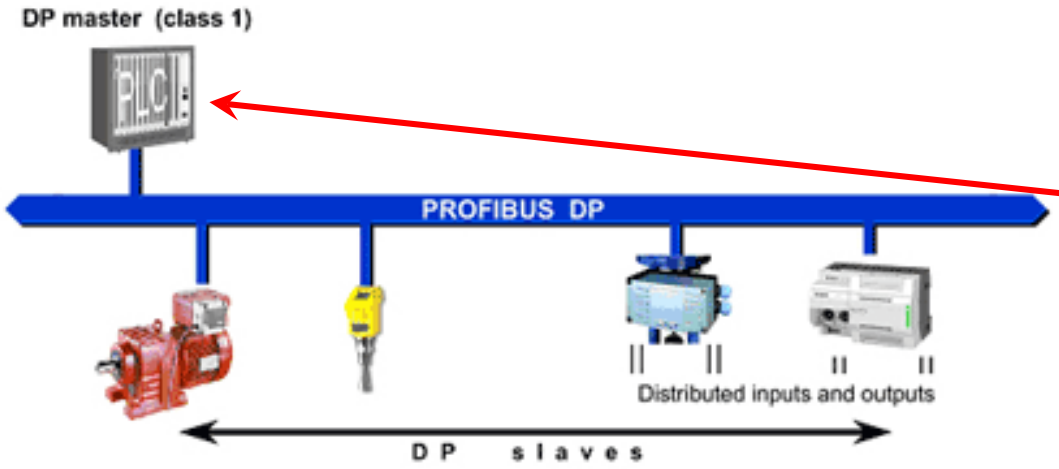
There are two variations of PROFIBUS in use today:

- PROFIBUS DP (Decentralized Peripherals) is used to operate sensors and actuators via a centralized controller in production (factory) automation applications
- PROFIBUS PA (Process Automation) is used to monitor measuring equipment via a process control system in process automation applications

PROFIBUS DP is the most commonly used.

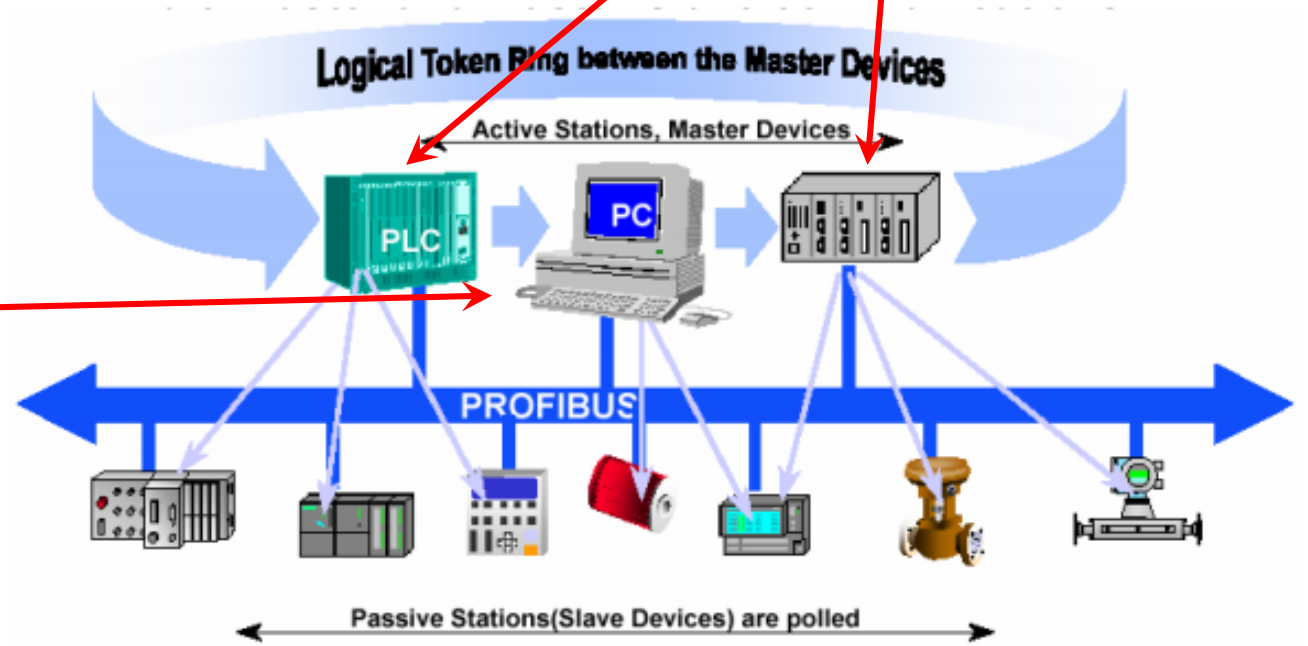
It is based on a RS-485 or a fiber optics communication and defines levels 1 and 2 of the OSI model. It defines also user profiles that can be used to describe properties and behavior of the devices connected to the network.

Fieldbus (IV)



Class 1 masters always connected

Class 2 master occasionally connected for configuration purposes



Fieldbus (V)

A Controller Area Network (CAN bus) is a vehicle bus standard designed to allow a real-time serial communication.

Development of the CAN bus started in 1983 at Robert Bosch GmbH, the protocol was officially released in 1986.

CAN is a multi-master serial bus standard to connect controllers, sensors and actuators at a speed up to $1\ Mbit/s$.

It has many advantages:

- low design and realization costs
- suitable for harsh environments (mechanical vibrations, electromagnetic disturbances,…)
- easily configurable
- automatic error detection

In industrial automation there are two versions of CAN bus in use today:

- CANOPEN
- DeviceNet

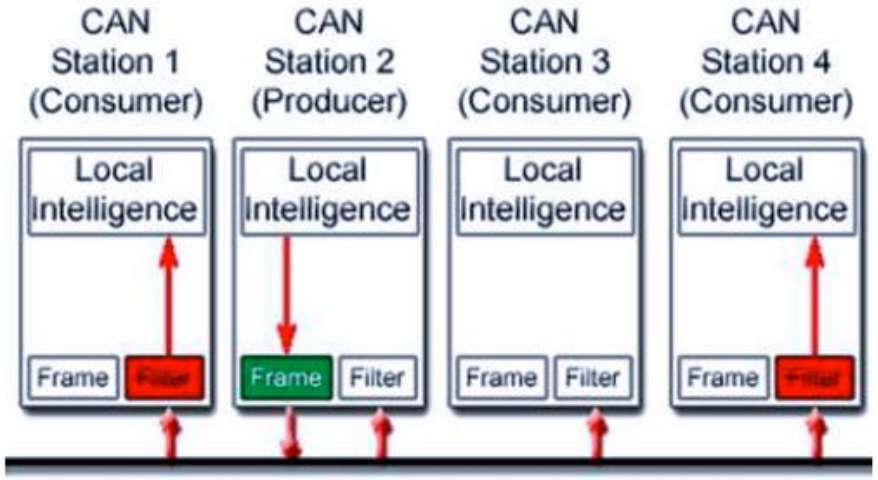
Fieldbus (VI)

At physical level CAN bus has the following characteristics:

- shielded twisted pair wires are used
- two different transmission speeds can be adopted
  - low speed: $125\ kBit/s$, up to $40\ m$, from 2 to 20 nodes
  - high speed: from $125\ kBit/s$ to $1\ MBit/s$, up to $40\ m$, from 2 to 30 nodes
- error correction is performed retransmitting every message until all the receivers received it without errors

At data link level CAN bus has the following characteristics:

- broadcast communication
- each message includes a unique node identifier which also represents the message priority
- every node receives all messages and selects the ones including its identifier



Fieldbus (VII)

CAN data transmission uses a lossless bit-wise arbitration method of contention resolution.

CAN specifications use the terms “dominant” bits and “recessive” bits, where dominant is a logical 0 and recessive is a logical 1.

If one node transmits a dominant bit and another node transmits a recessive bit then there is a collision and the dominant bit “wins”.

This means there is no delay to the higher-priority message, and the node transmitting the lower priority message becomes a receiver and automatically attempts to re-transmit after the end of the dominant message.

This makes CAN very suitable as a real-time prioritized communication system.

Let’s see an example.

Fieldbus (VIII)

	Start Bit	ID Bits										Rest of frame
		10	9	8	7	6	5	4	3	2	1	0	
Node 15	0	0	0	0	0	0	0	0	1	1	1	1	
Node 16	0	0	0	0	0	0	0	1	Stopped transmitting				
CAN data	0	0	0	0	0	0	0	0	1	1	1	1	

If a logical 1 is transmitted by all transmitting nodes at the same time, then a logical 1 is seen by all nodes, including both the transmitting nodes and receiving nodes.

If a logical 0 is transmitted by all transmitting nodes at the same time, then a logical 0 is seen by all nodes.

If a logical 0 is transmitted by one or more nodes, and a logical 1 is transmitted by one or more nodes, then a logical 0 is seen by all nodes including the nodes transmitting the logical 1. When a node transmits a logical 1 but sees a logical 0, it realizes that there is a contention and it quits transmitting.

By using this process, any node that transmits a logical 1 when another node transmits a logical 0 loses the arbitration. This means that the node that transmits the first 1 loses arbitration. The node with the lowest identifier transmits more zeros at the start of the frame, and that is the node that wins the arbitration or has the highest priority.

Fieldbus (IX)

We conclude this part mentioning industrial Ethernet that, in the last years, is rapidly spreading in factory automation and process control.

Industrial Ethernet refers to the use of standard Ethernet protocols with rugged connectors and extended temperature switches in an industrial environment.

Different protocols exist that make Ethernet suitable for a real-time fieldbus communication system:

- EtherCAT
- EtherNet/IP
- Powerlink
- PROFINET
- SERCOS III

Programmable Logic Controllers (I)

A process automation system is constituted by many different controllers.

Part of them are digital control systems, i.e., PIDs, used to control continuous time physical quantities.

Another important part is constituted by devices used to control action sequencing, safety interlock logics, etc. This devices are called Programmable Logic Controllers (PLCs).

Action sequencing, for example, is the control of a set of actions, triggered by events, that should be executed in the correct order (e.g., a robot taking parts from a conveyor belt).

It is rather common that the automation system is organized in hierarchical layers, e.g., the highest level is constituted by one or more PLCs doing action sequencing, the lowest by one or more digital control systems controlling the execution of each action.

Programmable Logic Controllers (II)

As for continuous time control systems, we need to introduce formal methodologies for

- writing requirements specification
- design an action sequencing control system

For requirements specification, suitable formal languages can be used (we will not discuss this aspect).

To devise a design methodology, one should make reference to discrete event system theory.

A discrete event system is a dynamical system characterized by

- the state space is discrete (each state variable takes value in a discrete set)
- the state evolution depends entirely on the occurrence of asynchronous discrete events over time

A classical example is a line of people who are waiting at a counter, where the state is the number of people in the line and the events are represented by people leaving or joining the line.

Programmable Logic Controllers (III)

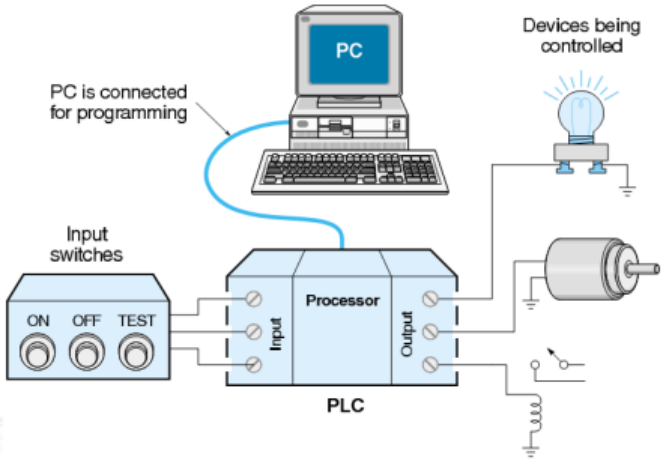
Discrete event systems can be modelled using finite state automata and Petri nets.

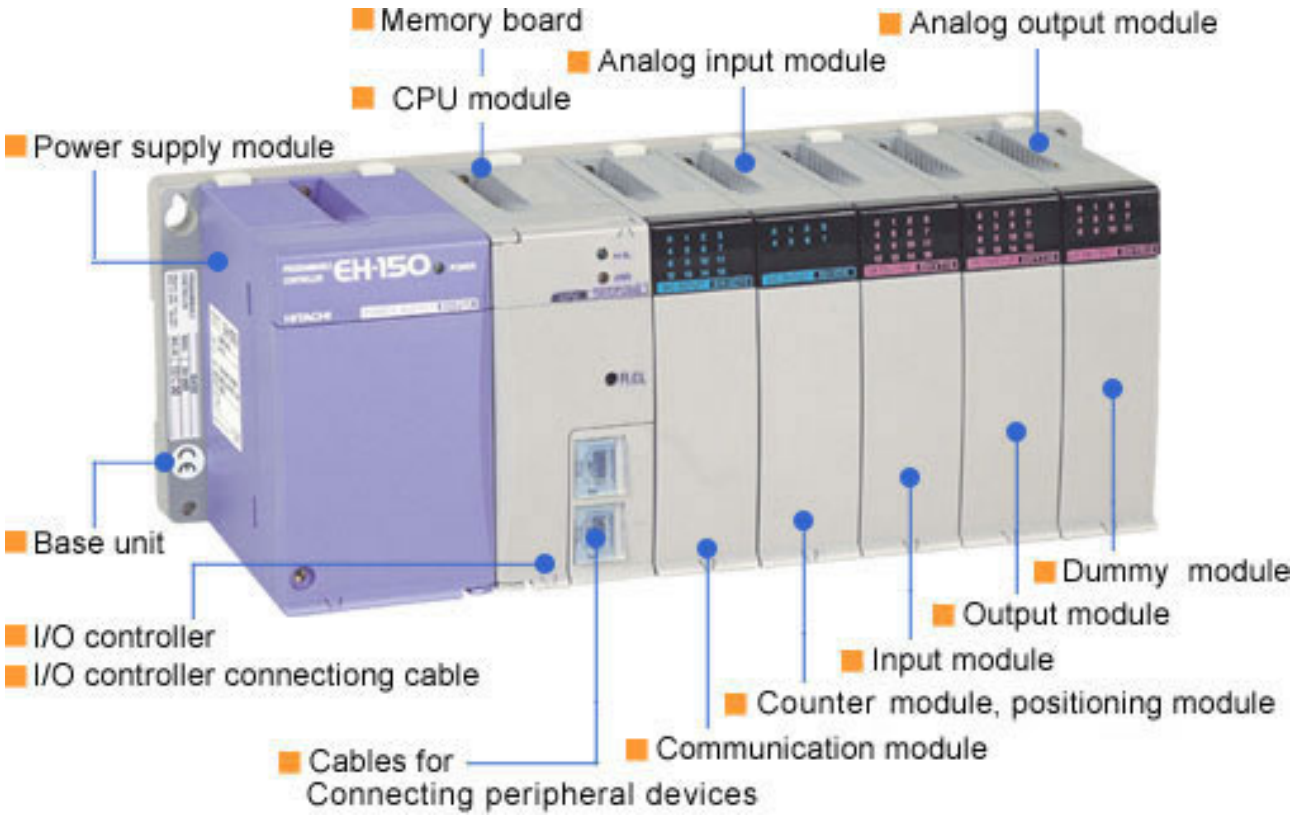
We will not discuss the details of discrete event system theory.

Instead we will introduce the fundamentals of PLCs and PLC programming languages.

Programmable Logic Controllers (IV)

We start describing the components of a PLC.





Programmable Logic Controllers (V)

A PLC program is generally executed repeatedly, every T_C milliseconds, as long as the controlled system is running, and is constituted by the following steps:

- the status of physical inputs is copied to an area of memory accessible to the processor (I/O Image Table)
- the user program is run from its first instruction rung down to the last rung, and the I/O image table is updated with the status of outputs
- operating system services are executed
- the status of outputs is copied from the I/O image table to physical outputs

T_C is called cycle time and is related to the specific application.

The previous steps are executed as a sequence of instructions, as a consequence the input signals can be acquired only at the beginning of the PLC cycle (no input variations during the cycle time can be detected).

Programmable Logic Controllers (VI)

Under the IEC 61131-3 standard, PLCs can be programmed using standards-based programming languages.

IEC 61131-3 currently defines five programming languages for programmable control systems:

- Function Block Diagram (FBD)
- Ladder Diagram (LD)
- Sequential Functional Chart (SFC)
- Structured Text (ST)
- Instruction List (IL)

FBD, LD and SFC are graphical programming languages.

ST and IL are text programming languages.

We will now introduce the first PLC programming language, the Ladder Diagram.

Programmable Logic Controllers (VII)

Ladder logic was originally a written method to document the design and construction of relay racks as used in manufacturing and process control.

Ladder logic has then evolved into a programming language that represents a program by a graphical diagram based on the circuit diagrams of relay logic hardware.

Ladder logic has contacts that make or break circuits to control coils. Each coil or contact corresponds to the status of a single bit in the programmable controller's memory.

The language, initially devised only for Boolean functions, has been then extended in order to support integer and real functions as well.

The motivation for representing sequential control logics in a ladder diagram was to allow factory engineers and technicians to develop software without additional training to learn a programming language.

Programmable Logic Controllers (VIII)

A ladder diagram is constituted by:

- two vertical rails, a power supply wire on the left and a ground wire on the right
- a set of rungs that typically has one or more contacts on the left side and one or more coils on the right side
- current can only flow from power supply (left rail) to ground (right rail)
- logical variables are associated to each contact and coil
- rungs are executed from the top to the bottom of the ladder diagram
- if a path can be traced between the left side of the rung and ground, through closed contacts, the rung is true and the output coil storage bit is asserted (1), otherwise is false (0)

Let’s now introduce the main components of a ladder diagram.

Programmable Logic Controllers (IX)

Rung input:

--| |-- a regular contact, if the corresponding bit is asserted the contact is closed (open contact at rest)

--|/|-- a "not" contact, if the corresponding bit is asserted the contact is open (closed contact at rest)

--|P|-- rising edge contact, if the corresponding bit makes a low-to-high transition the contact is closed for one cycle

--|N|-- falling edge contact, if the corresponding bit makes a high-to-low transition the contact is closed for one cycle

Rung output:

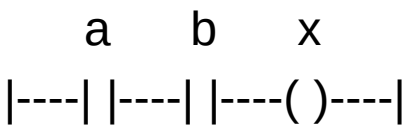
--( )-- a regular coil, energized (bit asserted) whenever its rung is closed (inactive at rest)

--(/)-- a "not" coil, energized (bit asserted) whenever its rung is open (active at rest)

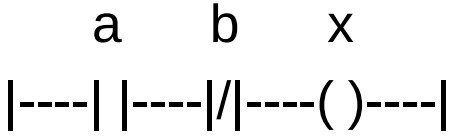
--(L)-- latch coil, energized (bit asserted) whenever its rung is closed, it keeps energized until an unlatch coil associated to the same bit is energized

--(U)-- unlatch coil, energized (bit set to 0) whenever its rung is closed, it keeps energized until a latch coil associated to the same bit is energized

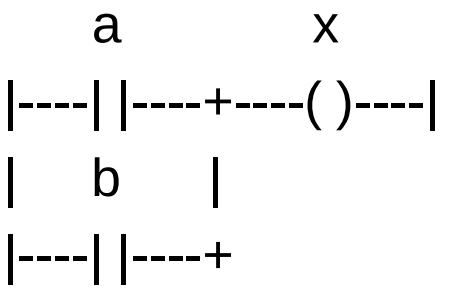
Programmable Logic Controllers – Examples



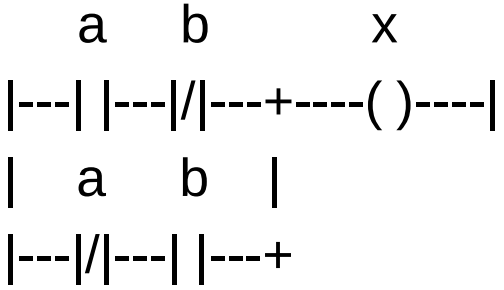
x = a AND b



x = a AND NOT b



x = a OR b

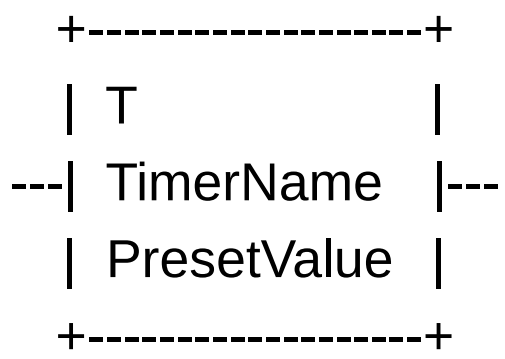


x = a XOR b

Programmable Logic Controllers (X)

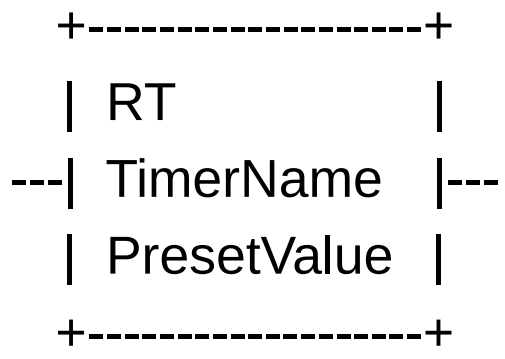
Additional functionalities includes timers (used to delay actions) and counters (used to count events and trigger actions when predefined values are reached).

Timer



- if the timer is energized it starts counting
- when the PresetValue is reached variable TimerName is asserted and remains set until the timer is de-energized
- otherwise variable TimerName is set to 0

Retentive timer



- when timer is de-energized it stops counting but it is not reset
- when timer is energized again it starts counting from the last value
- to reset the timer a suitable command is used

TimerName
---(RES)---

Programmable Logic Controllers (XI)

Counter

```
   +--------------------+
   |  CU                |
---|  CounterName       |---
   |  PresetValue       |
   +--------------------+
```

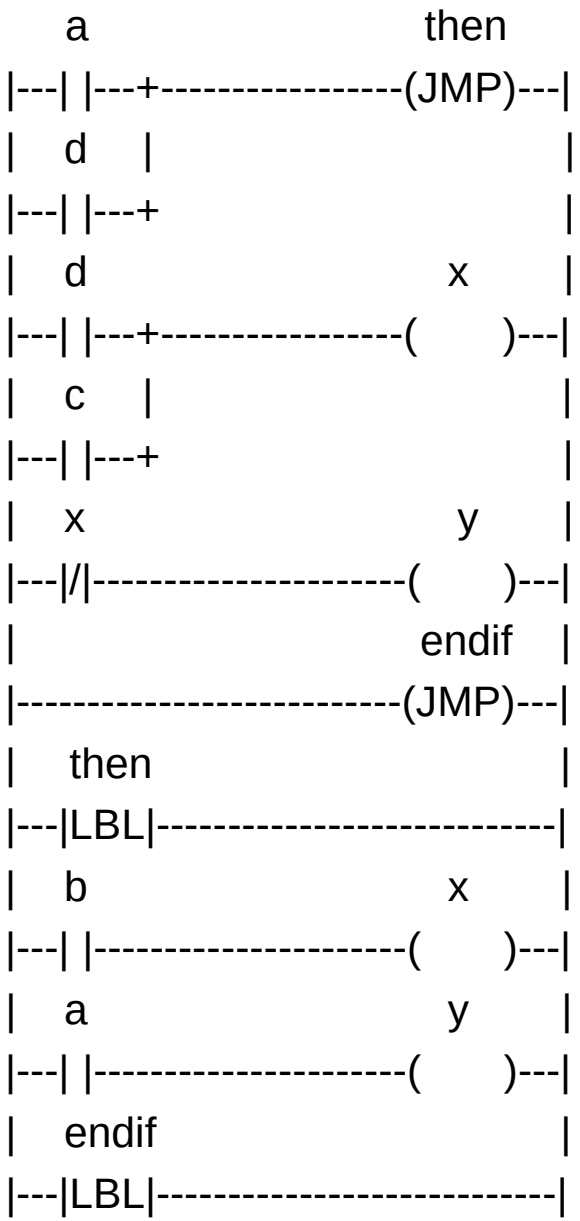
- is similar to the retentive timer
- counts the low-to-high transitions on the input line up to the PresetValue
- when the PresetValue is reached, variable CounterName is asserted and remains set until the counter is reset
- to reset the counter a suitable command is used

```
CounterName
---(RES)---
```

Finally, we can introduce jumps in the ladder using the following instructions:

---(JMP)---	if coil JMP is energized the execution jumps to the rung
---|LBL|---	immediately after the rung that includes LBL (no other instructions can be put on this rung)

Programmable Logic Controllers – Examples (I)

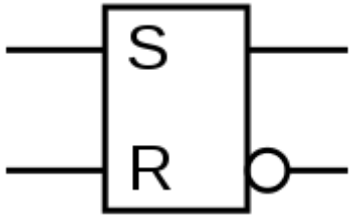


This ladder diagram implements the following code

```
if (a OR d)
 {
   x = b;
   y = a;
 }
else
 {
   x = d OR c;
   y = NOT x;
 }
```

Programmable Logic Controllers – Examples (II)

A set-reset flip-flop is a circuit that has two stable states and can be used to store state information.



The output of the flip-flop is asserted (1) if there is a low-to-high transition on S (set) and remains asserted until R (reset) is set to 1.

The set-reset flip-flop is equivalent to the following equation

$$Q = (NOT\ R)\ AND\ (Q\ OR\ S)$$

It can be implemented using the following ladder diagram

```
    S        R        Q
|---| |---+---|/|-------( )----|
|   Q   |
|---| |---+
```

This diagram can be used to enable a motor using an enabling switch (S) and a disabling switch (R).

Q represents the state of the motor (1 if the motor is moving).

Real-time systems (I)

A discrete time control law periodically computes the control variable given a measure of the controlled variable and the reference signal. This computation is executed every sampling interval.

To replicate the same behavior on a computer system we need an operating system that always takes the same amount of time to complete the control task. We will call such a system a real-time system.

How can we define a real-time system?

- a system in which the time at which output is produced is significant, as the input usually corresponds to some event in the physical world, and the output has to relate to that same event;
- a system that must process information and produce a response within a specified time, else risk severe consequences, including failure;
- a system whose correctness is based on both the correctness of the outputs and their timeliness;
- a system which has to respond to externally generated input stimuli within a finite and specified period.

Real-time systems (II)

Real-time does not mean fast!

A real-time system does not have to be fast: the deadline may be days or weeks… it means that it must produce the response by the required time.

To measure the timeliness of a real-time system we introduce the notion of jitter.

We define jitter the amount of error in the timing of a task over subsequent iterations of a program or loop.

A good real-time system provides a low amount of jitter when programmed correctly.

We can classify real-time systems into two groups:

- soft real-time systems, are systems that still function even if deadlines are sometimes not met (failure to meet response time constraints leads only to a decrease of performance)
- hard real-time systems, are systems where a failure to meet response time constraints leads to a catastrophic system failure

Real-time systems (III)

From a hardware point of view, a real-time system should have:

- one or more processors with suitable computational power
- known and predictable instruction execution time (at least for the worst case)
- low and deterministic memory and I/O latency
- automatic fault detection capabilities
- hardware redundancy

From a software point of view, a real-time operating system should have:

- a task scheduler allowing different process priorities
- a multitasking pre-emptive scheduler (a task can be temporarily interrupted and later resumed) allowing context switches
- techniques to avoid deadlock conditions
- mechanisms to support inter-process communication and process synchronization
- mechanisms to handle interrupts

Real-time systems (IV)

Writing software for a real-time system is far more complex than writing standard software.

Rigorous techniques and tools are required, for the design and for the testing phase as well.

To simplify the development of real-time applications, supporting inter-process communication, synchronization, data exchange and distributed computing, middleware exists (e.g., the OROCOS framework).

Finally, we mention some real-time operating systems:

- commercial
  - QNX
  - VxWorks
  - WindowsCE
- open source
  - RTAI
  - Xenomai
  - RTLinux