

School of Electrical Engineering

Department of Electrical Engineering and Automation

ELEC 8201 Control & Automation

Industrial Automation Software

Valeriy Vyatkin

Course logistics

This part contributes to 1/3 of the mark

4 lectures

4 quizzes

2 homeworks

2 exam questions

Course team

Prof. Valeriy Vyatkin



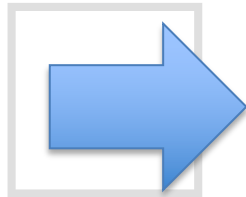
Dr. Udayanto Dwi Atmojo



Mr. Pranay Jhunjhunwala



Accessing the E-book



IEC 61499 function blocks for embedded and distributed control systems design

☆☆☆☆☆ (0)

EBSCOhost
Vyatkin, Valeriy
ISA

Holdings

Reviews

Internet

EBSCOhost E-kirja/E-book

Searching: [eBook Collection \(EBSCOhost\)](#), [Show all](#) | [Choose Databases](#)

AN 1201596 [Select a Field \(option...\)](#) [Search](#) [Create Alert](#) [Clear](#) [?](#)

AND [Select a Field \(option...\)](#)

AND [Select a Field \(option...\)](#) [+](#) [-](#)

[Basic Search](#) [Advanced Search](#) [Search History](#)

[Result List](#) | [Refine Search](#) [1 of 1](#)

IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design


Authors: [Vyatkin, Valeriy](#)

Publication Information: Ed.: Third Edition. Research Triangle Park, NC : International Society of Automation. 2016

Resource Type: eBook.

Description: IEC 61499 is the standard for distributed control systems that follows on from the IEC 61131 standard for programmable logic controllers (PLC). This book is a practical guide for component-based development of distributed embedded and control systems as proposed by the new international standard. Each chapter is designed as an independent study unit, making the book ideal for use in university courses, industrial training, or self-study. Working knowledge of the IEC 61499 standard can be achieved in approximately 10 to 15 learning hours. For the control, automation, or software engineer and the embedded systems developer, this book provides concrete directions on how to specify and implement a distributed system according to the IEC 61499 standard and how to create an IEC 61499-compliant control device. The text also sheds some light on the broader embedded systems arena since the IEC 61499 standard provides the higher level (yet executable!) abstraction appropriate for model-based engineering of distributed embedded systems.

Subjects: [Embedded computer systems--Standards](#)



[Detailed Record](#)

[PDF Full Text](#)

[Download This eBook \(Offline\)](#)

[SFX](#)

Related Information

[Table of Contents](#)

[Find Similar Results](#)
using SmartText Searching.

A

Tools

- We will be using NxtStudio
- Install it on your personal laptops, or use the virtual machine image
 - Refer to the **VDI Instructions** guide on MyCourses

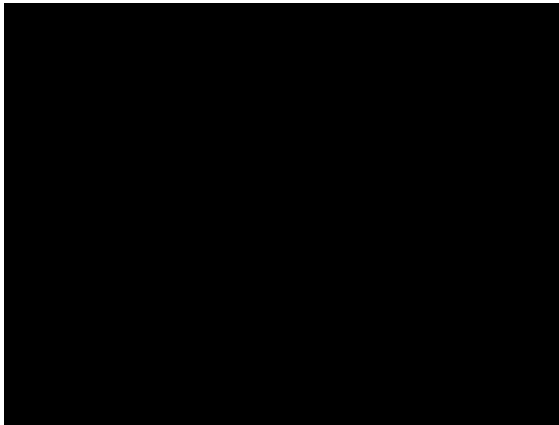
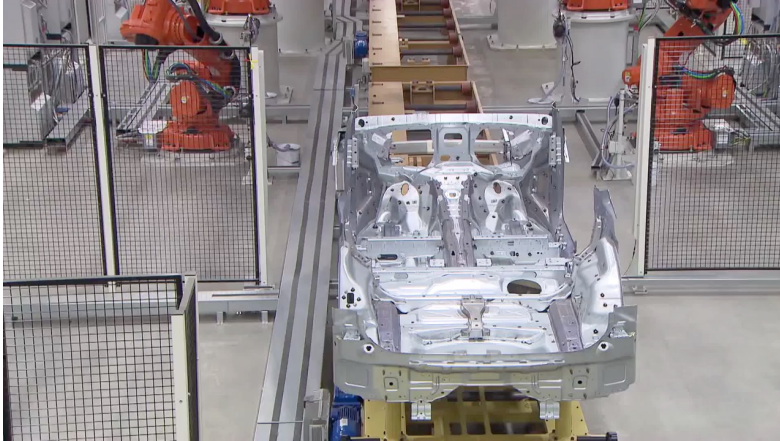
Online course on IEC 61499 – free access

- Subscribe to the following YouTube channel https://www.youtube.com/channel/UCAXM7DKhEqJlo_zxOscz7rA and watch the introductory video.
- Enrol to this online study platform (for free) and go through the available material watching and repeating on your laptops.
 1. <https://training.flexbridge.se/courses/basic-introduction-to-iec-61499?coupon=aalto-2022-spring>
 2. <https://training.flexbridge.se/courses/distributed-automation-programming-with-iec-61499-basic-introduction?coupon=aalto-2022-spring>

Quizzes in the Automation part

- Quizzes are released after each lecture
- Submission deadline is in 24 hours

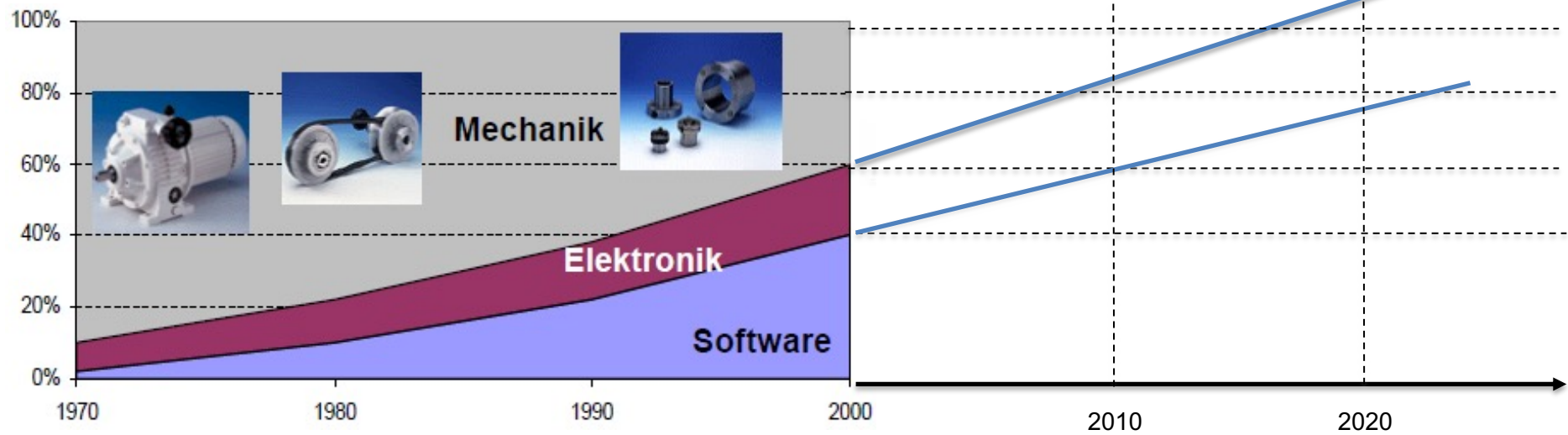
What is Industrial Automation?



What is Industrial Automation?



Software Ratio in Industrial Automation

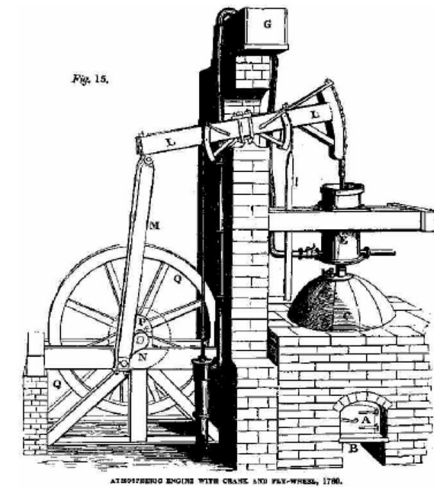


According to Verband Deutscher Maschinen- und Anlagenbau e.V. (VDMA)

Generations of Automation Systems

1. Relay based controllers (40s – 50s)
2. Microprocessor based PLCs (70s)
3. Multifunctional PLCs
4. Industrial Networks
5. Internet of Things

Generation 0. Mechanical regulators

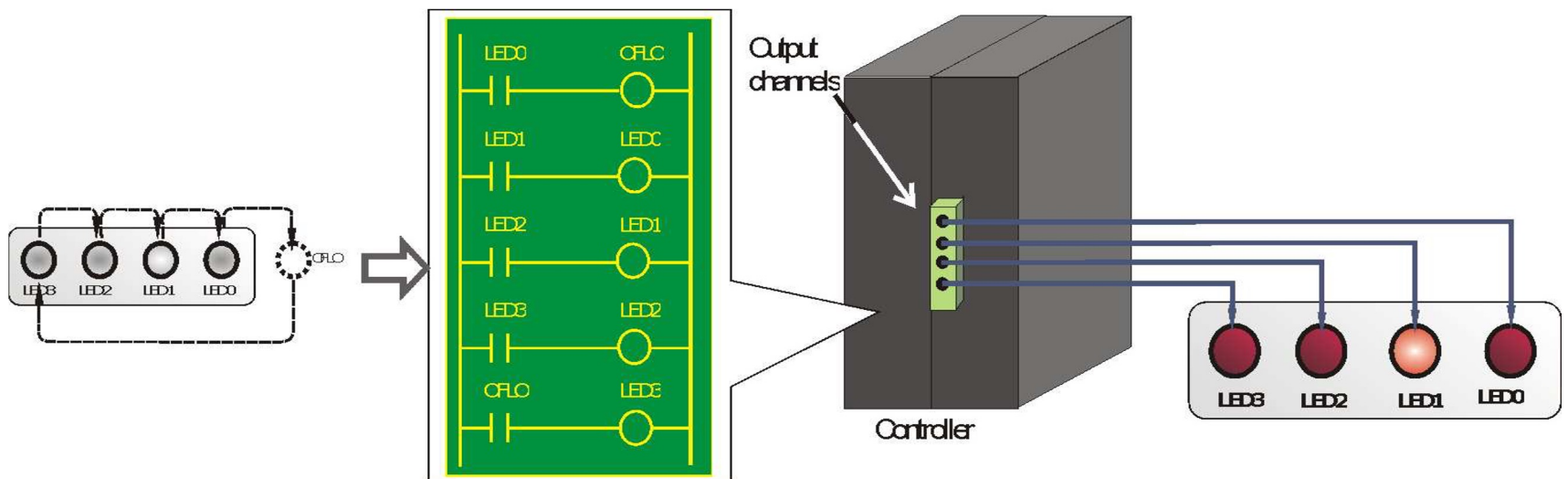


Mine in Cornwall equipped with first steam engine in 1790. First industrial regulator invented by James Watt to keep the wheel's speed constant.

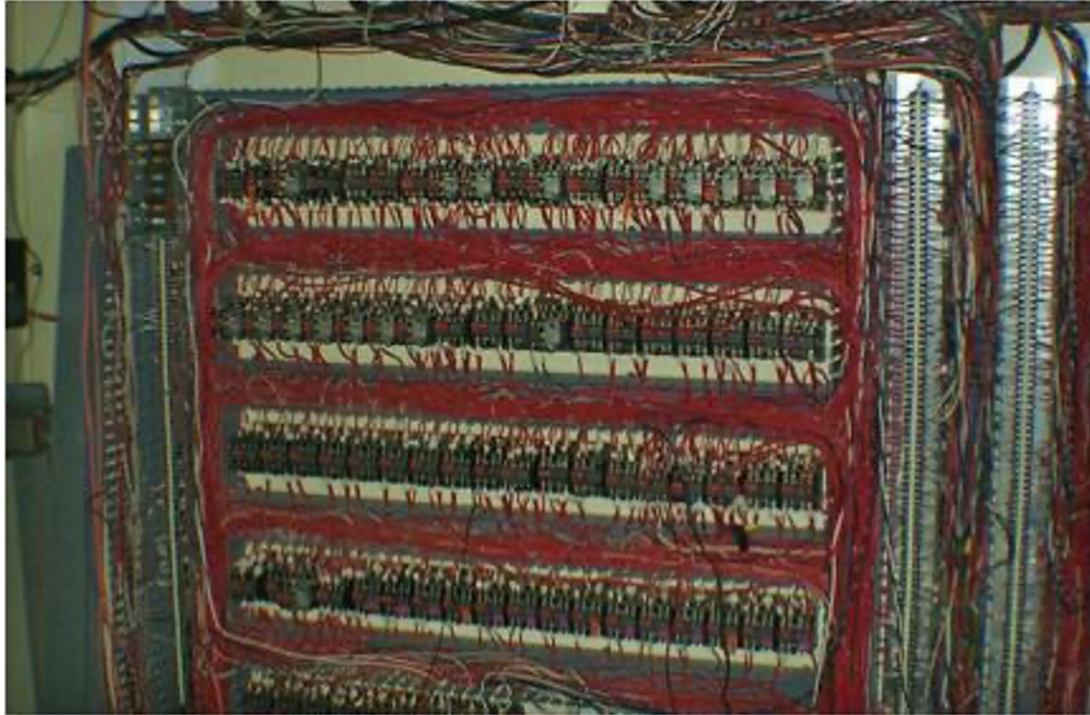


Generation 1: Relay Ladder Circuits

Hard-wired ladder logic circuits were widely used to control industrial equipment. This explains current popularity of the Ladder Diagram language for programming industrial controllers

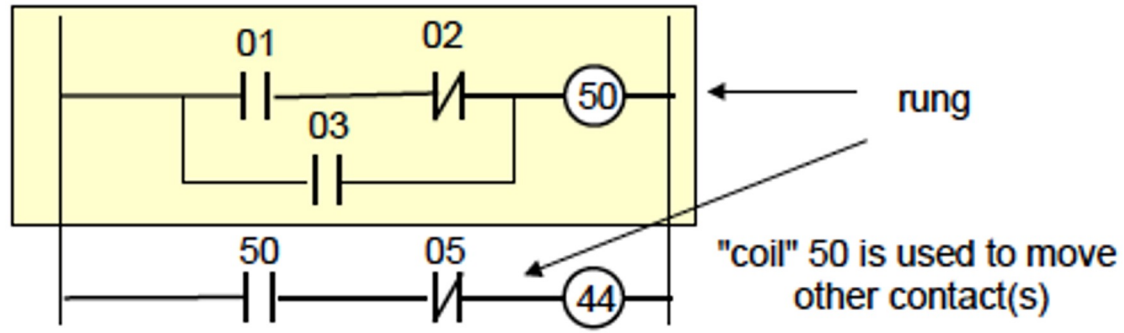
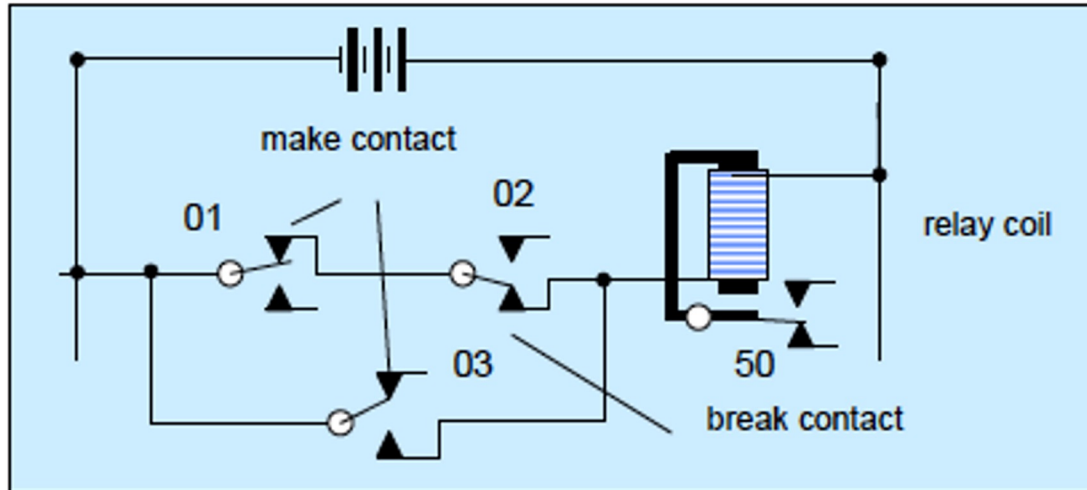


Relay Ladder Logic



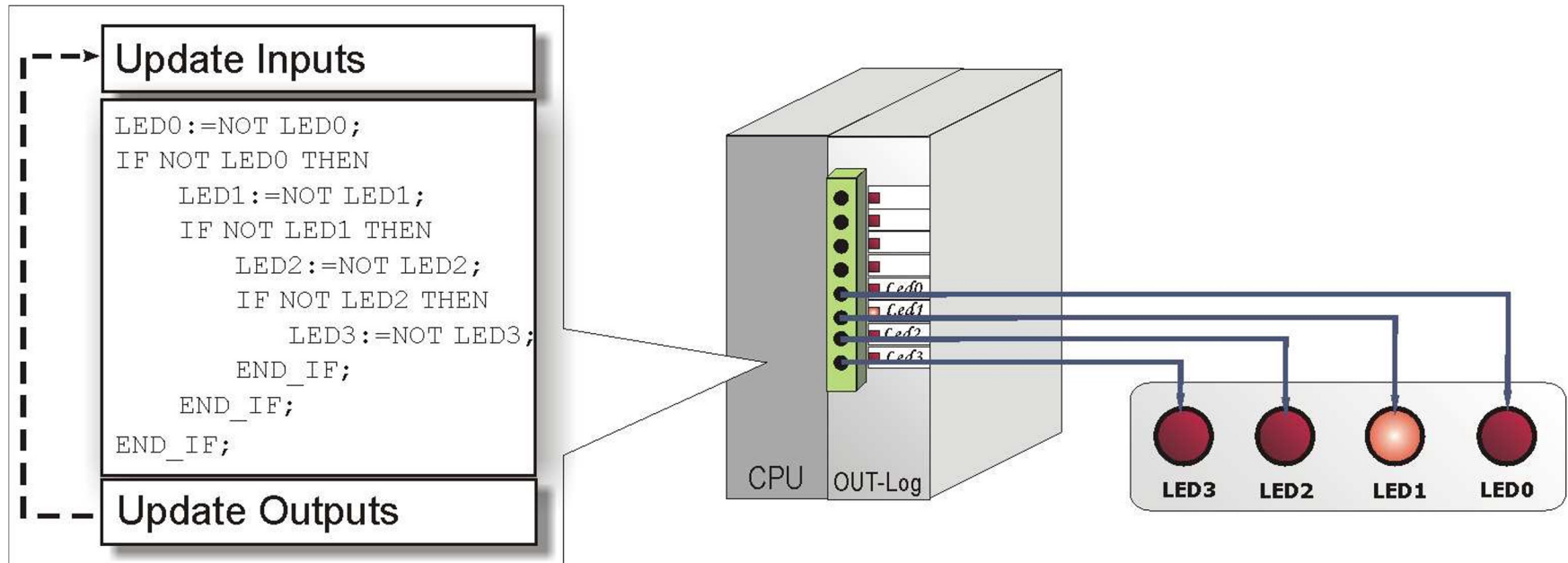
Control logic implemented as hard-wired **relays**

Electric Circuit and Ladder Diagram

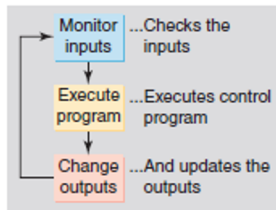
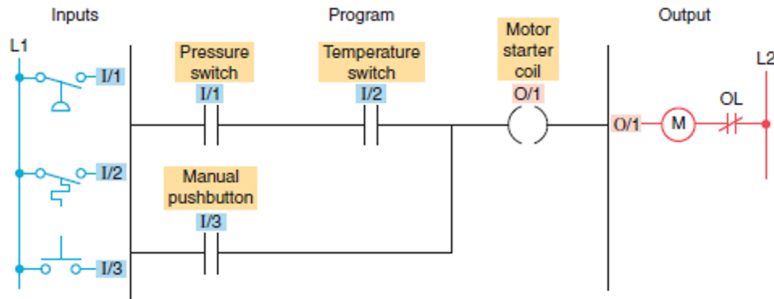


Generation 2: Programmable Logic Controllers (PLCs)

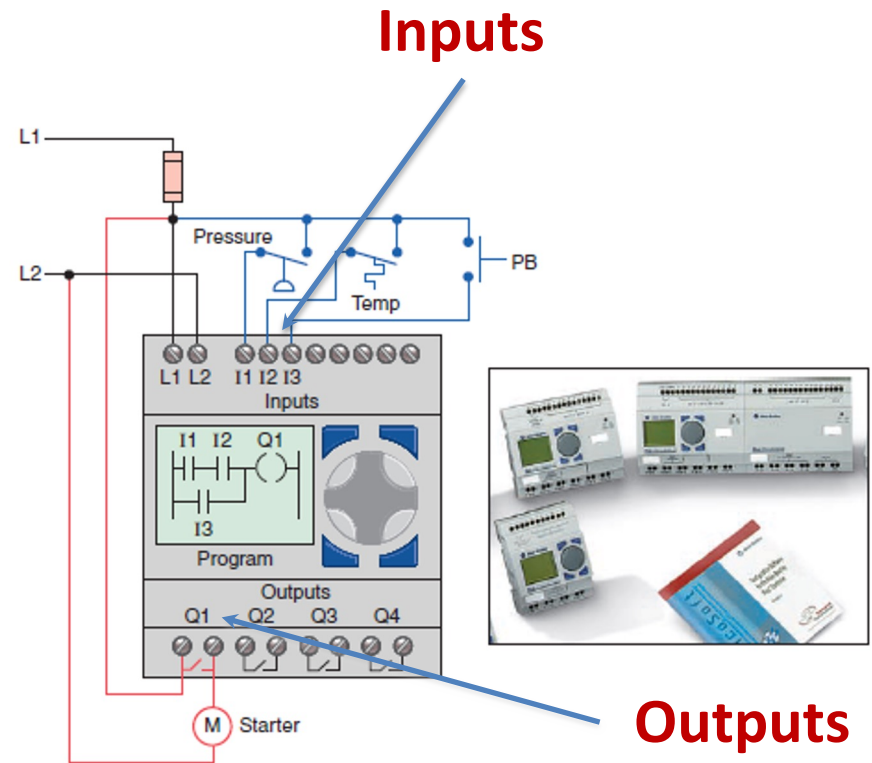
PLCs – specially hardened industrial computers - tremendously improved flexibility of automation systems



PLC Programming with Ladder Logic



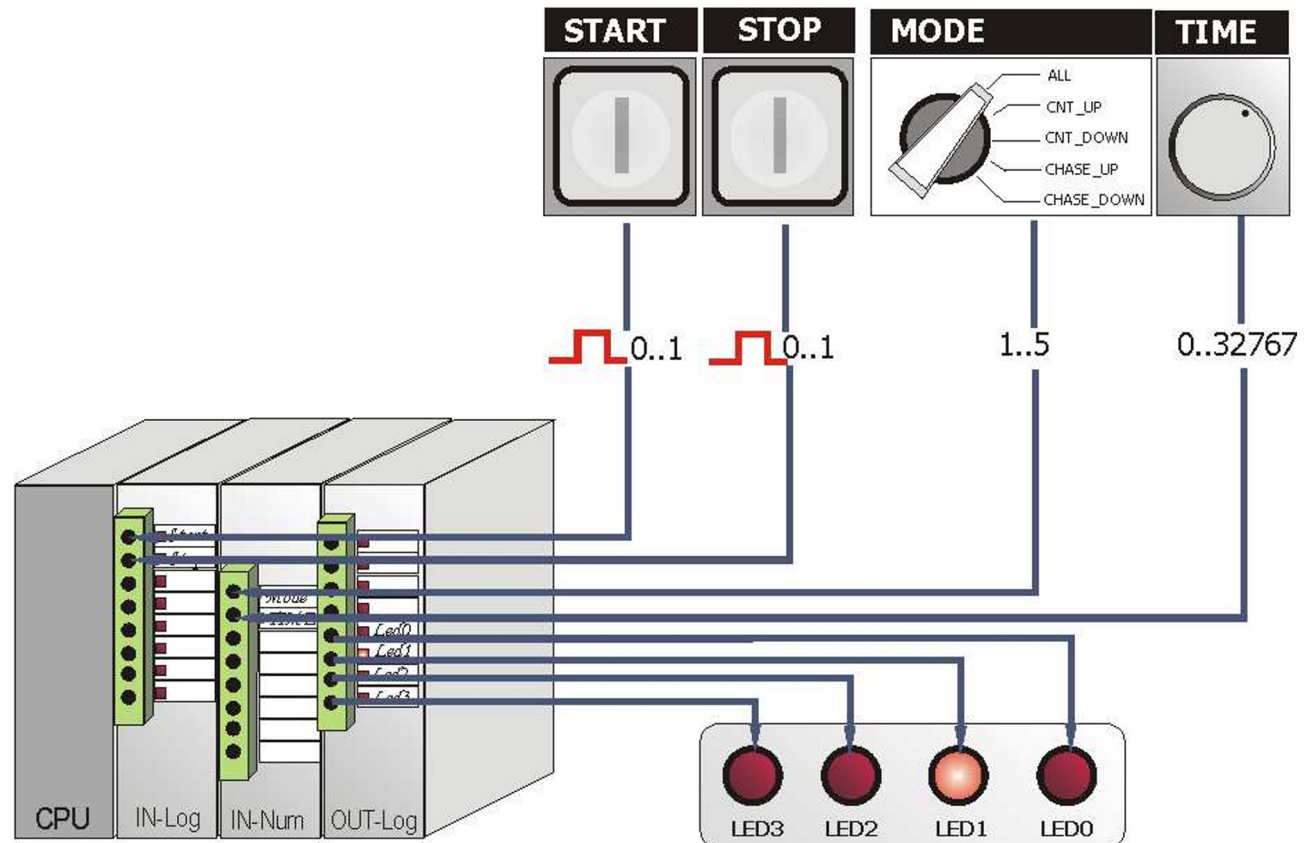
Process control PLC ladder logic program with typical addressing scheme.



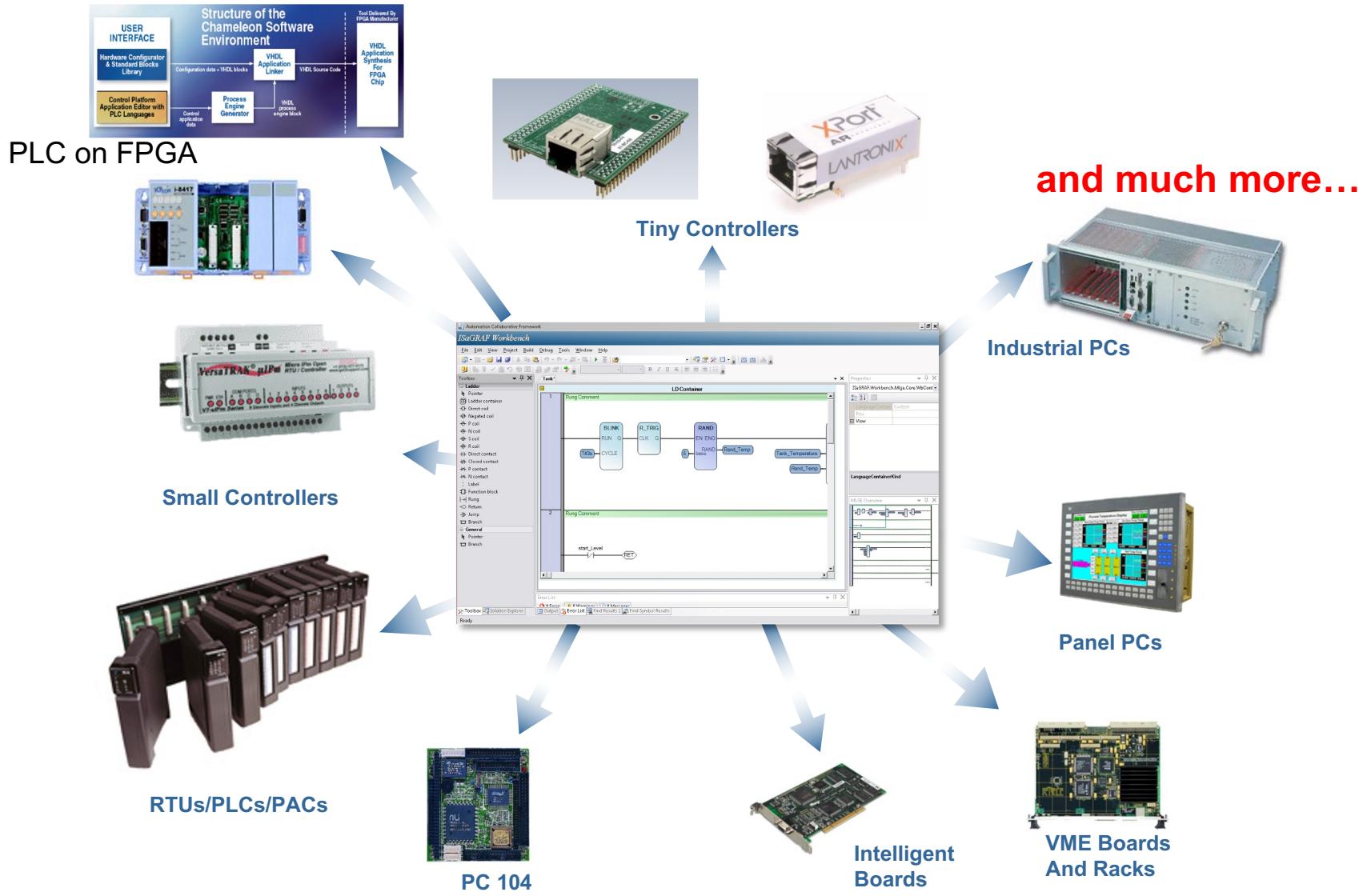
Typical wiring required to implement the process control scheme using a fixed PLC controller.

Generation 3: Multifunctional PLCs

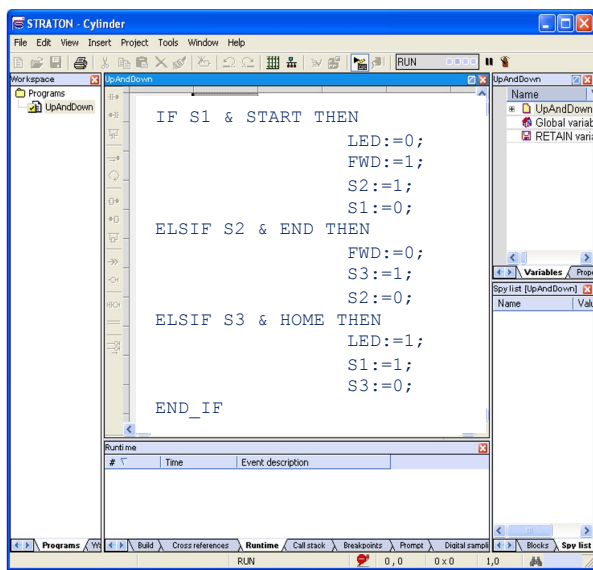
Modern PLCs know many programming languages and have versatile and easily expandable architecture.



Programmable Logic Controllers: Form Factors



Generation 4: Industrial Networks



```
IF S1 & START THEN
    LED:=0;
    FWD:=1;
    S2:=1;
    S1:=0;

ELSIF S2 & END THEN
    FWD:=0;
    S3:=1;
    S2:=0;

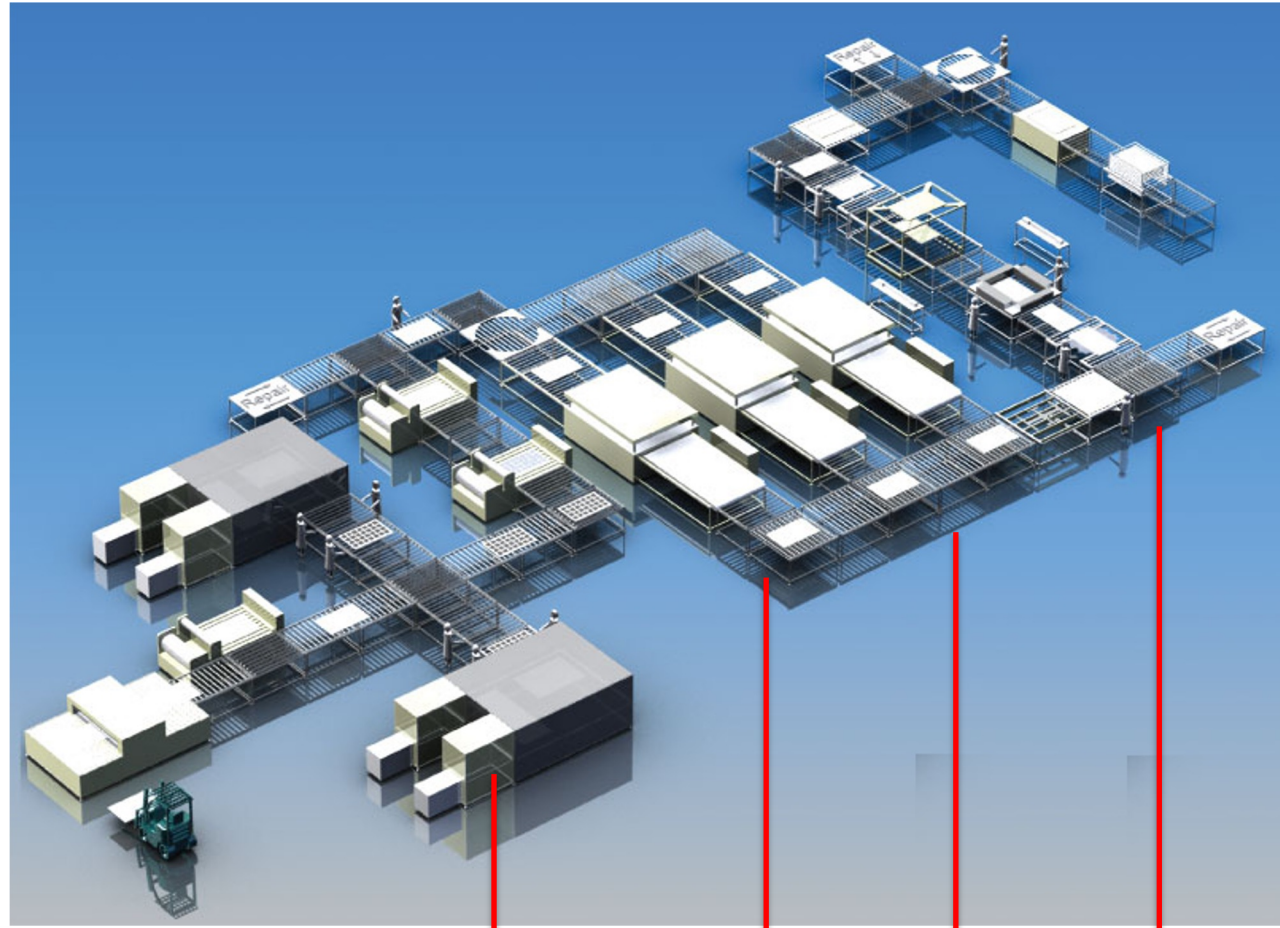
ELSIF S3 & HOME THEN
    LED:=1;
    S1:=1;
    S3:=0;

END_IF
```

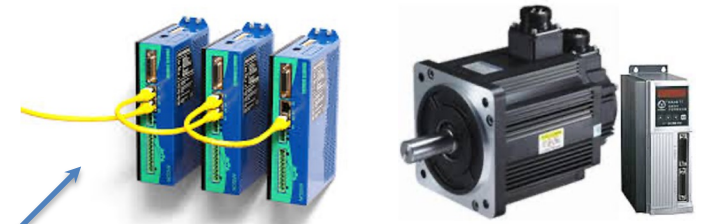
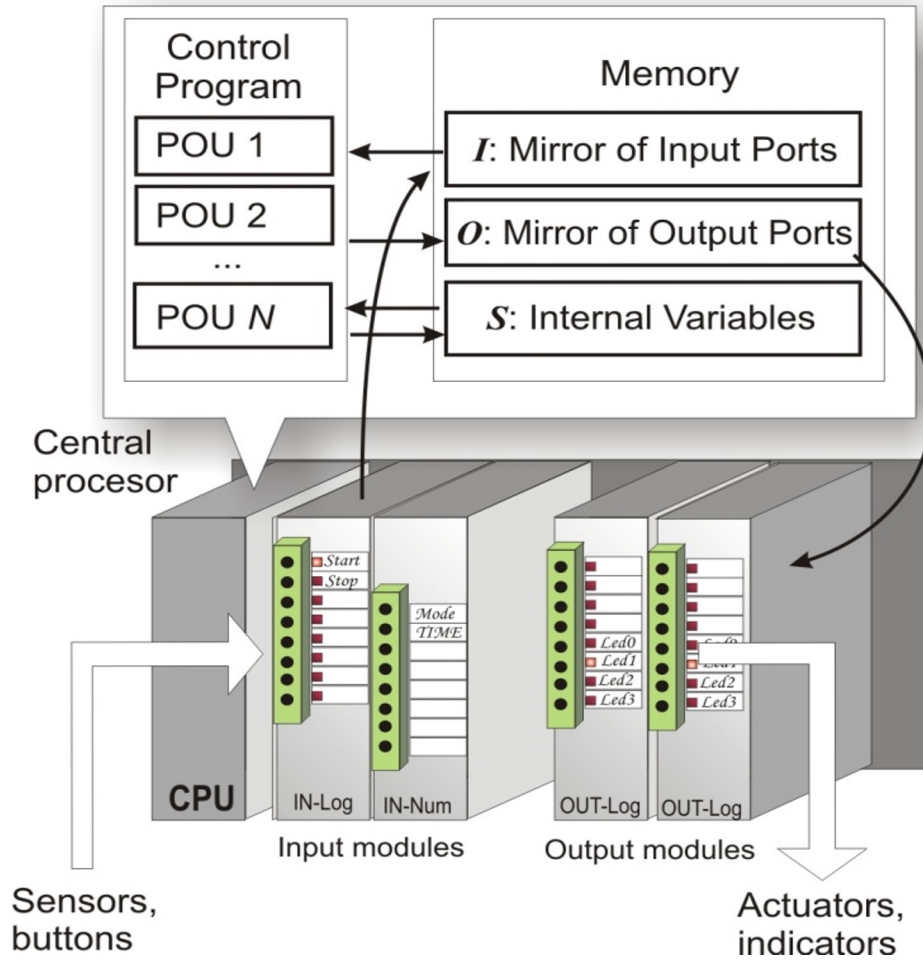
The screenshot shows the STRATON - Cylinder software interface. The main window displays a ladder logic program for an 'UpAndDown' system. The program consists of three main branches: 1. If S1 and START are true, then LED is set to 0, FWD is set to 1, S2 is set to 1, and S1 is set to 0. 2. If S2 and END are true, then FWD is set to 0, S3 is set to 1, and S2 is set to 0. 3. If S3 and HOME are true, then LED is set to 1, S1 is set to 1, and S3 is set to 0. The program ends with END_IF. The interface also shows a workspace with 'UpAndDown' programs, a runtime table, and a status bar at the bottom.



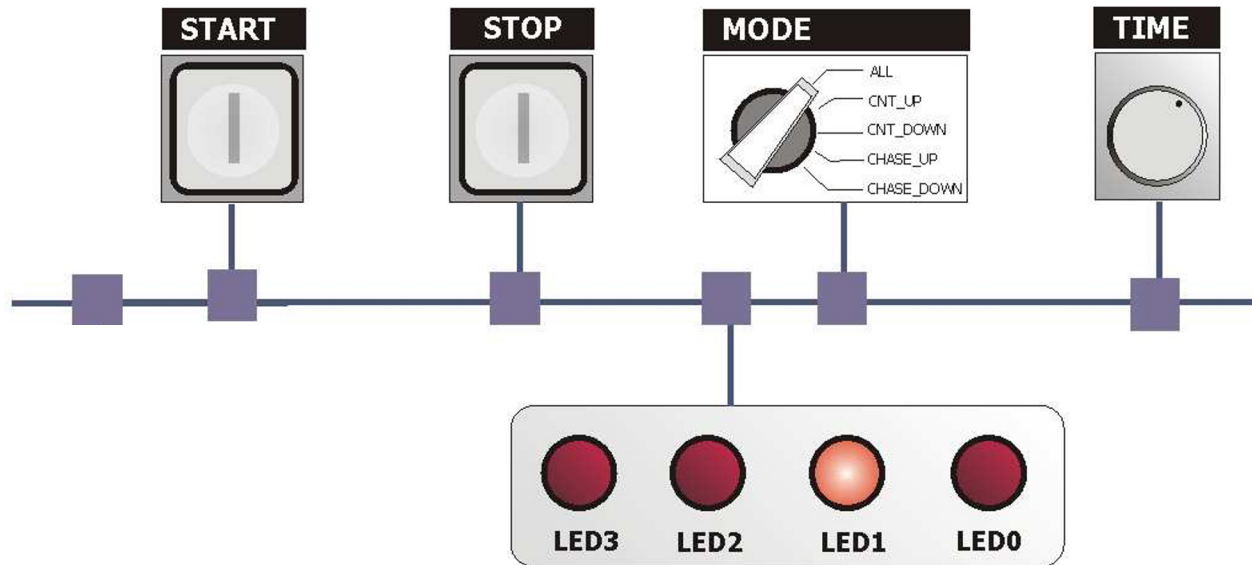
PLC



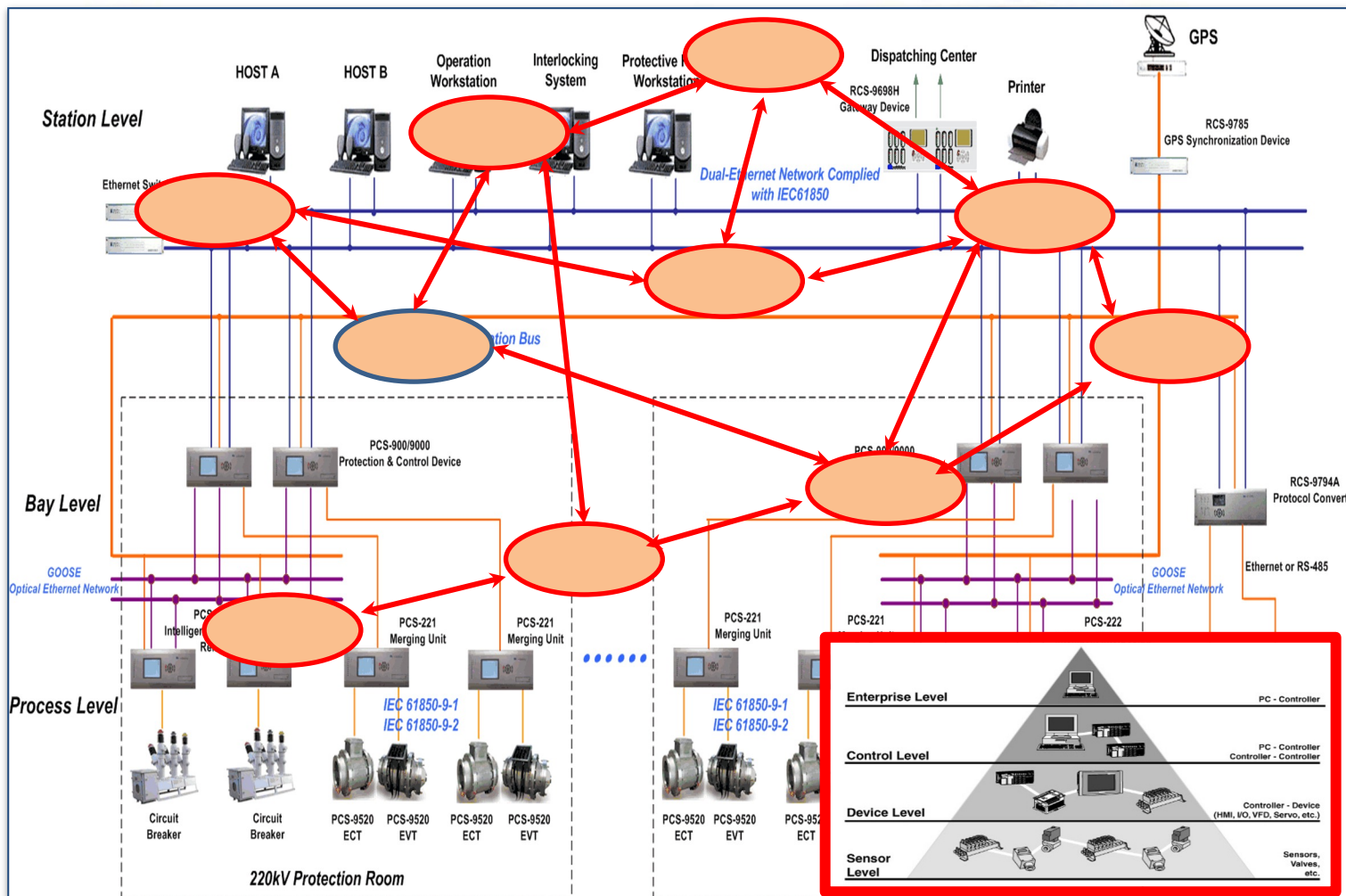
PLC as Integration Platform



Generation 5: From Networking to Internet of Things

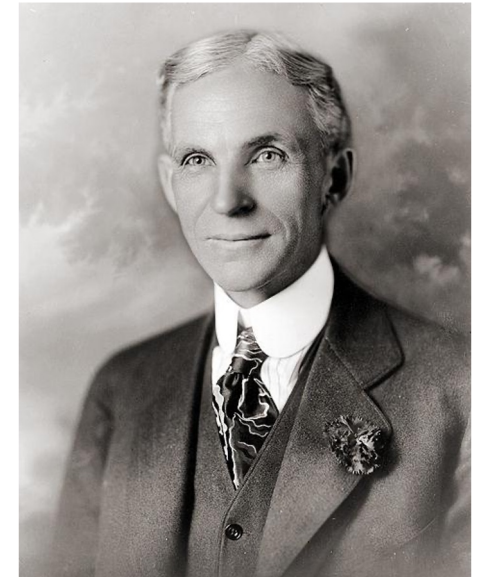


Machine To Machine Communication, Internet of Things



Production in the past...

“Any customer can have a car painted any colour that he wants so long as it is black!”



Henry Ford (1863-1947)

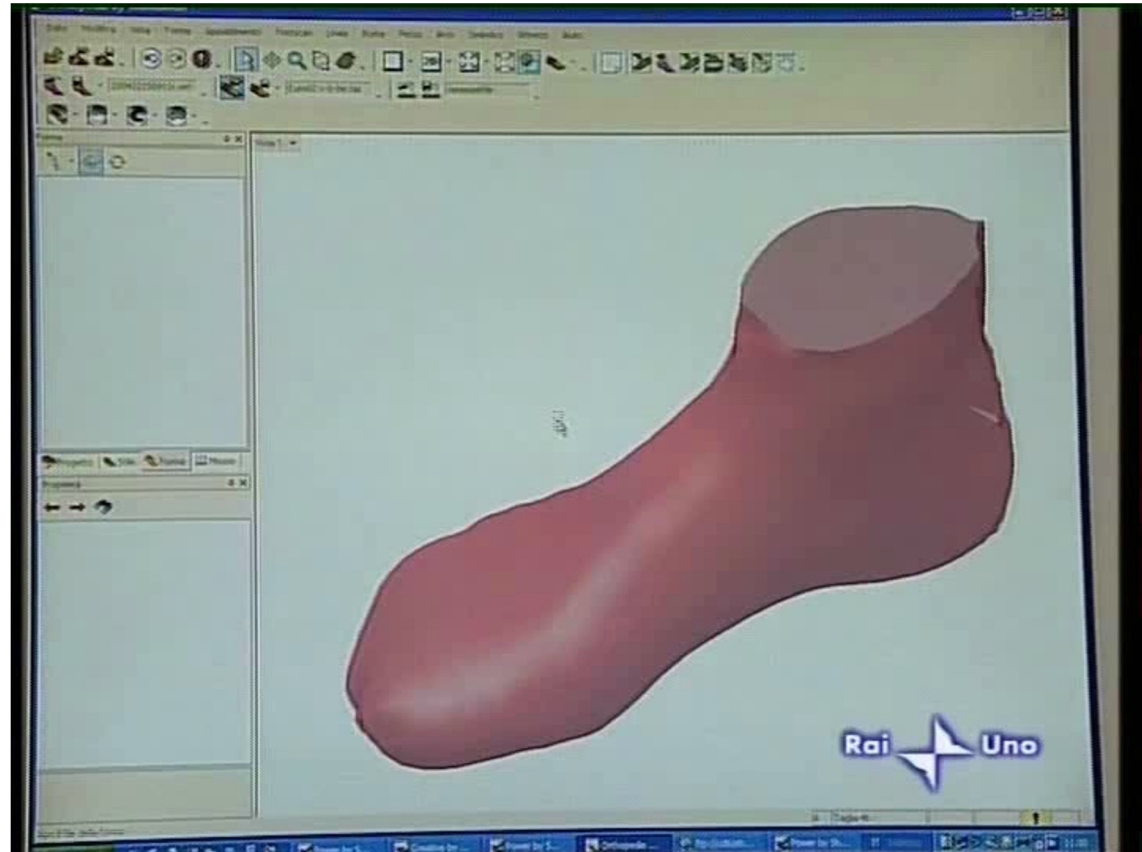
Source of photo: Wikipedia



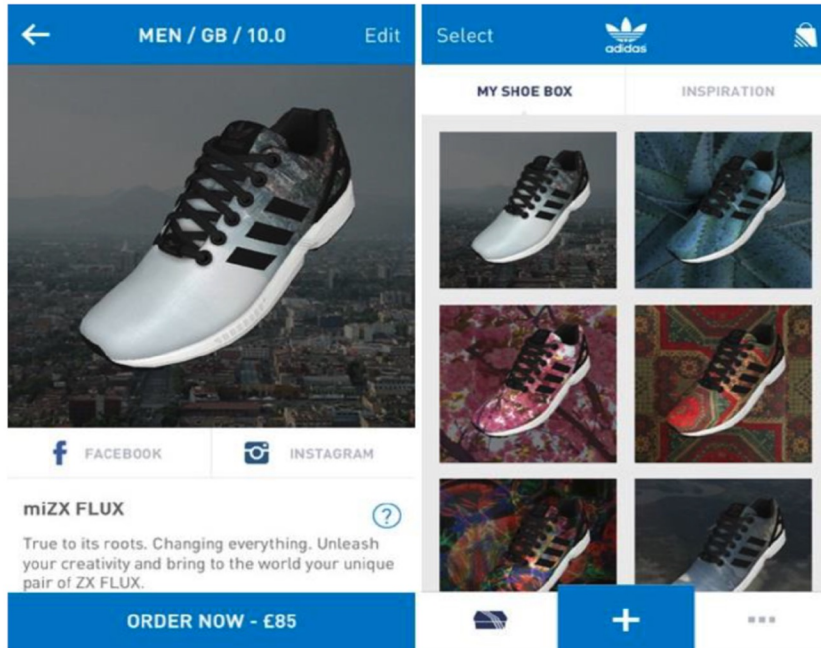
1926 Ford Model T

Source of photo: Boldride

XXI Century: Manufacturing to order!

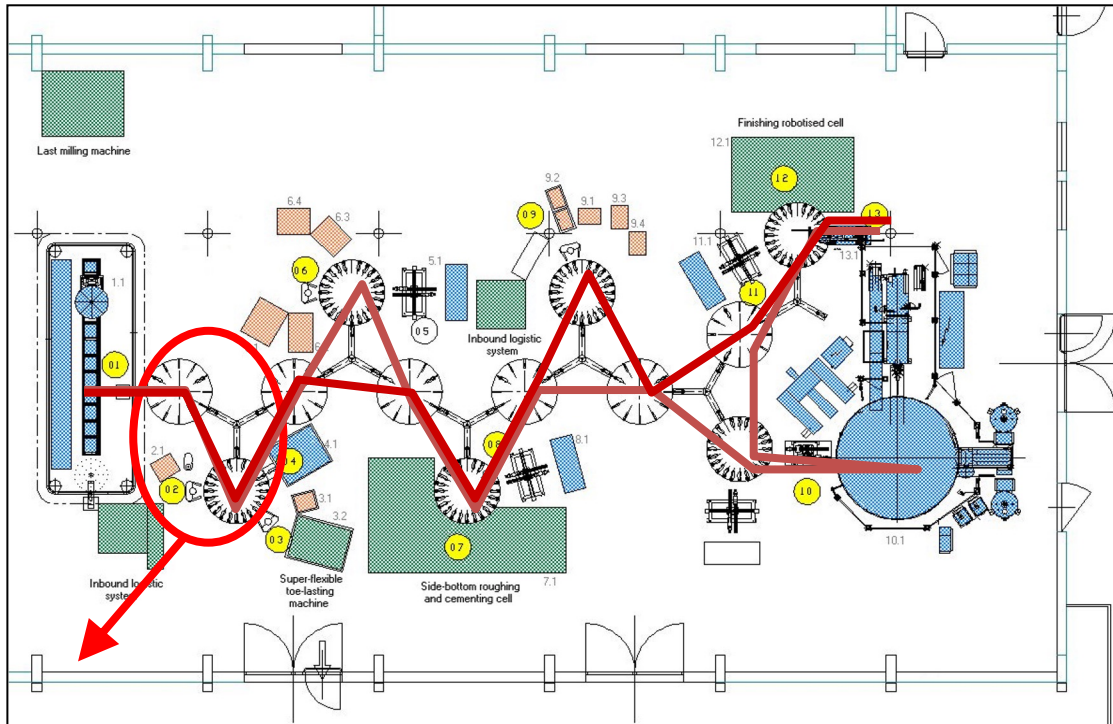


The Adidas Speedfactory: Bringing Sports Shoes Production back to Germany by Industrie 4.0 for Mass Customization



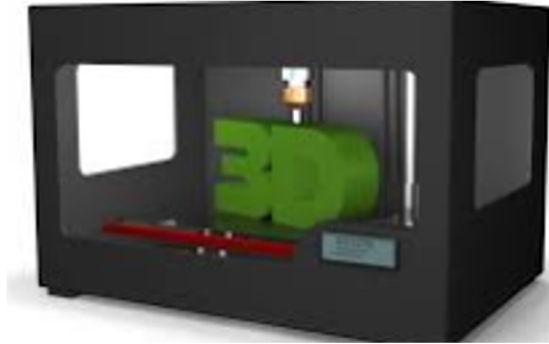
- The costumers can design their own short shoes using an App.
- Since the customer wants to receive his personalized product on the next day or faster, long logistic chains from low-wage countries are no longer acceptable in the era of mass customization.
- Thus, adidas decided to open various "speedfactories" for personalized shoes in Germany close to the customer, using Cyber-physical production systems (CPPS).

Flexible Manufacturing

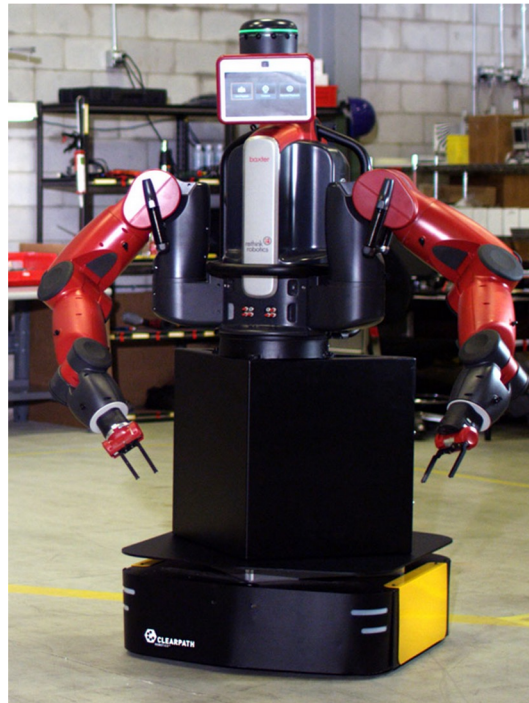


Industry 4.0: New Factory Floor

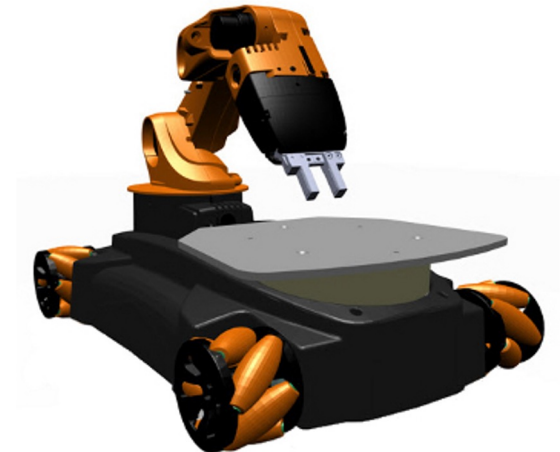
Industrial robots



3D printers



Mobile machines



Aalto Factory of the Future at Scanautomatic Fair 2018



Computations in Controllers

IEC 61131-3 Programming Languages

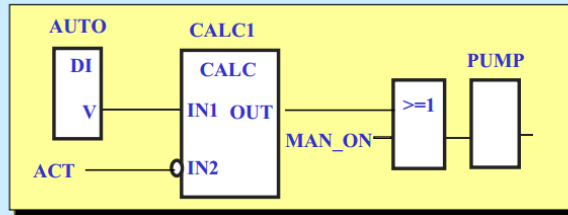
The five IEC 61131-3 Programming languages

<http://www.isagraf.com>

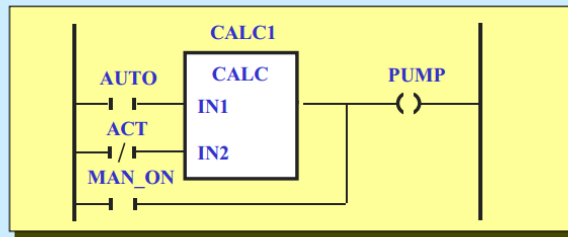
Function Block Diagram (FBD)

graphical languages

Sequential Flow Chart (SFC)

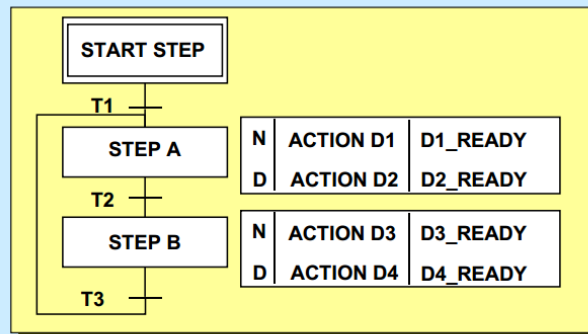


Ladder Diagram (LD)



Instruction List (IL)

```
A: LD  %IX1 (* PUSH BUTTON *)
    ANDN %MX5 (* NOT INHIBITED *)
    ST  %QX2 (* FAN ON *)
```



textual languages

Structured Text (ST)

```
VAR CONSTANT X : REAL := 53.8 ;
Z : REAL; END_VAR
VAR aFB, bFB : FB_type; END_VAR

bFB(A:=1, B:='OK');
Z := X - INT_TO_REAL (bFB.OUT1);
IF Z>57.0 THEN aFB(A:=0, B:="ERR");
ELSE aFB(A:=1, B:="Z is OK");
END_IF
```

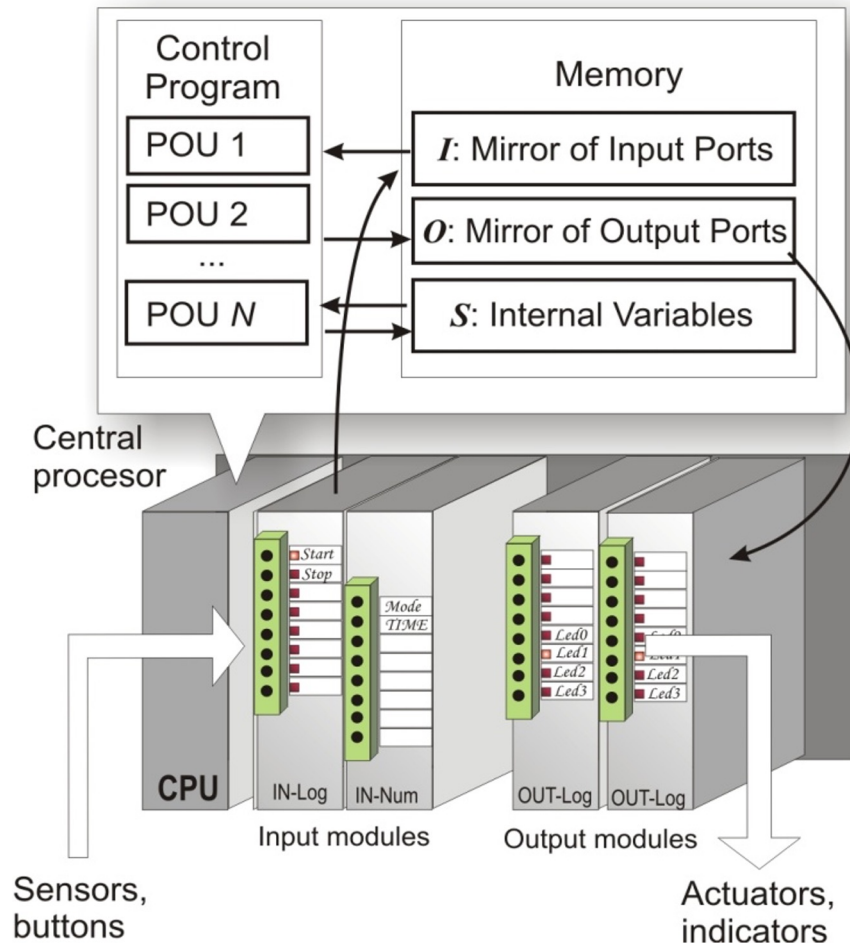
Software Tools

- *CoDeSys*
 - Software tool for developing and engineering IEC 61131-3 controller applications
 - a soft PLC from 3S-Smart Software Solutions GmbH
- *TwinCAT*
 - OEM version of CoDeSys for Beckhoff, under Visual Studio Shell
- *ISaGRAF*
- *KW*
- Tools of *SIEMENS, Rockwell, ABB...*

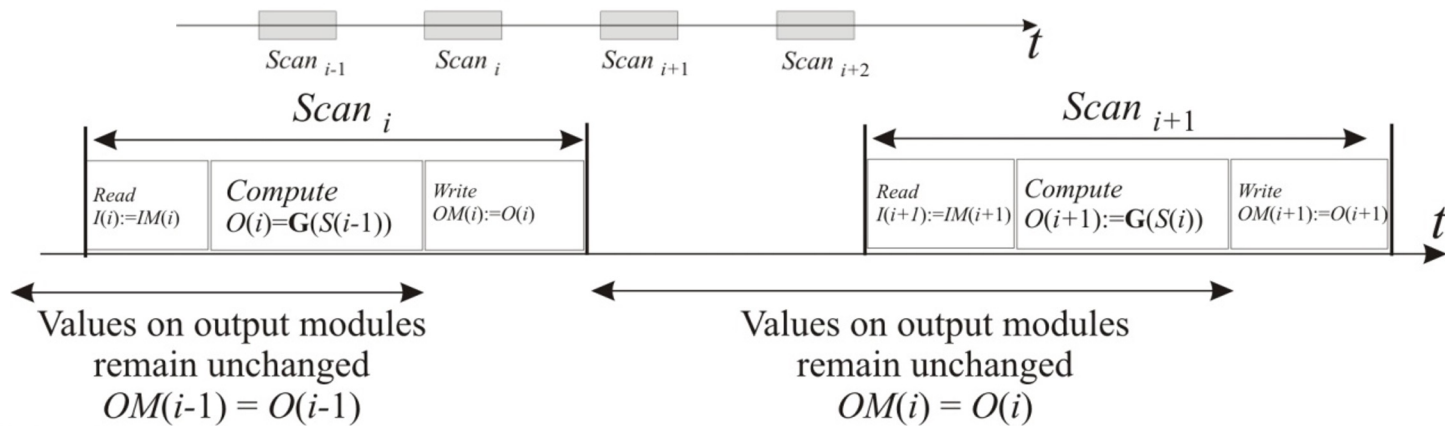
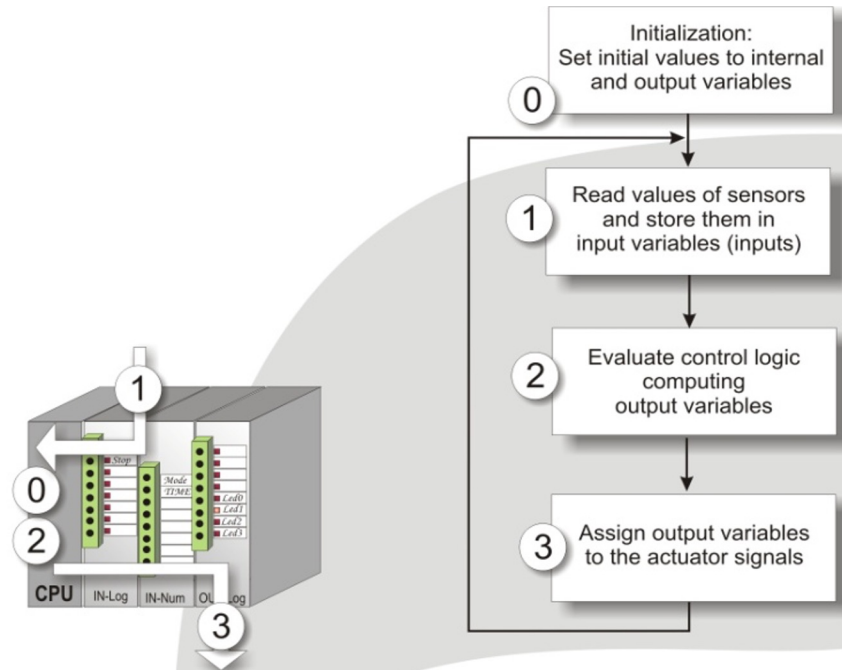
Key Features of IEC 61131-3

- IEC 61131-3 is the most important automation standard in industry.
- 80% of all PLCs support it, all new developments base on it.
Depending on the country, some languages are more popular.
- **Structured software** - *through use of Configuration, Resource, and Program Organization Units (POU)*
- **Software encapsulation** - *through use of POU, and complex data types*
- **Strong Data Typing** - *through languages that restrict operations to only apply to appropriate types of data*
- **Execution control** - *through use of tasks*

PLC



Cyclic Program Execution



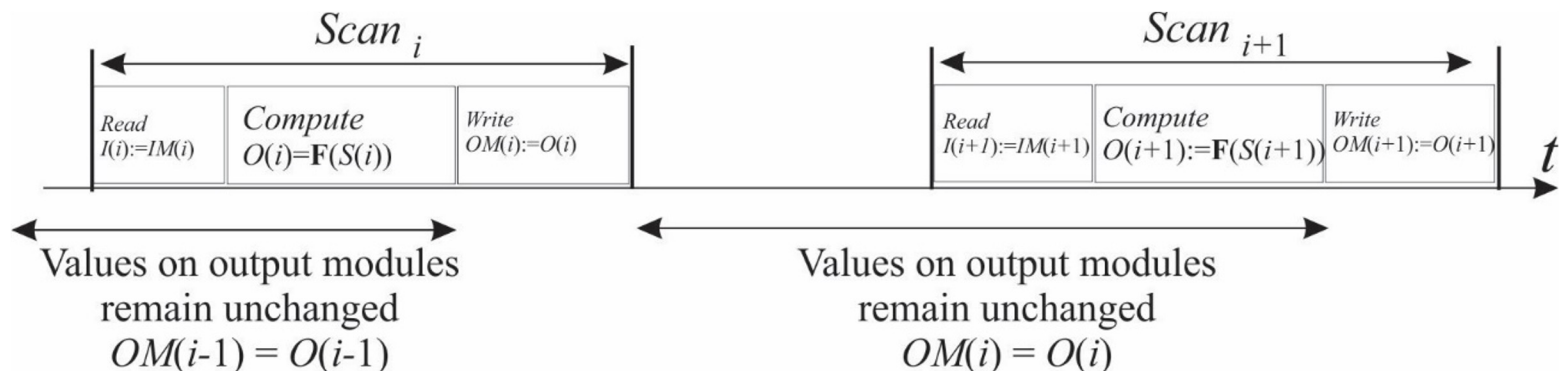
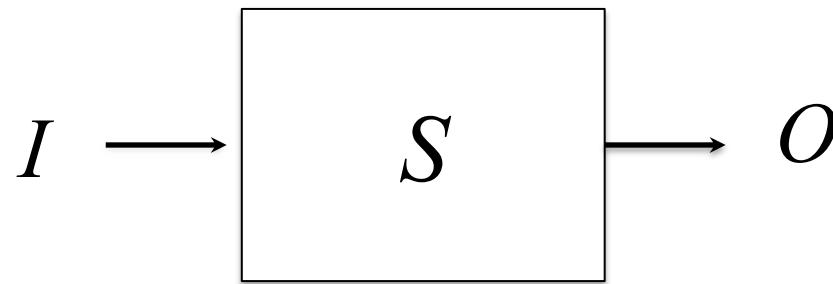
Reasons

- Reactivity:
- Timeliness:
- Reliability: if input is read with an error, the error will be corrected in the next scan.

Formal Models of Automation Logic

Let us denote by I and O Boolean vectors of inputs and outputs, and S is Boolean vector of state variables.

Then the semantics of the controller can be described by the following system of Boolean assignments:



General Models of PLC execution

Combinatorial:

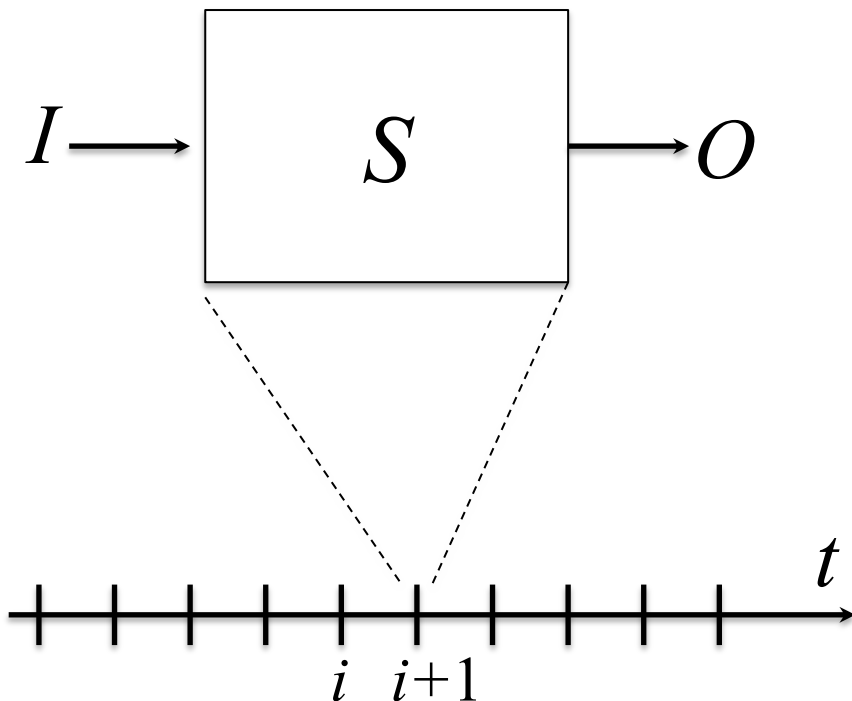
$$O(i) = \mathbf{F}(I(i))$$

Moore type state machine:

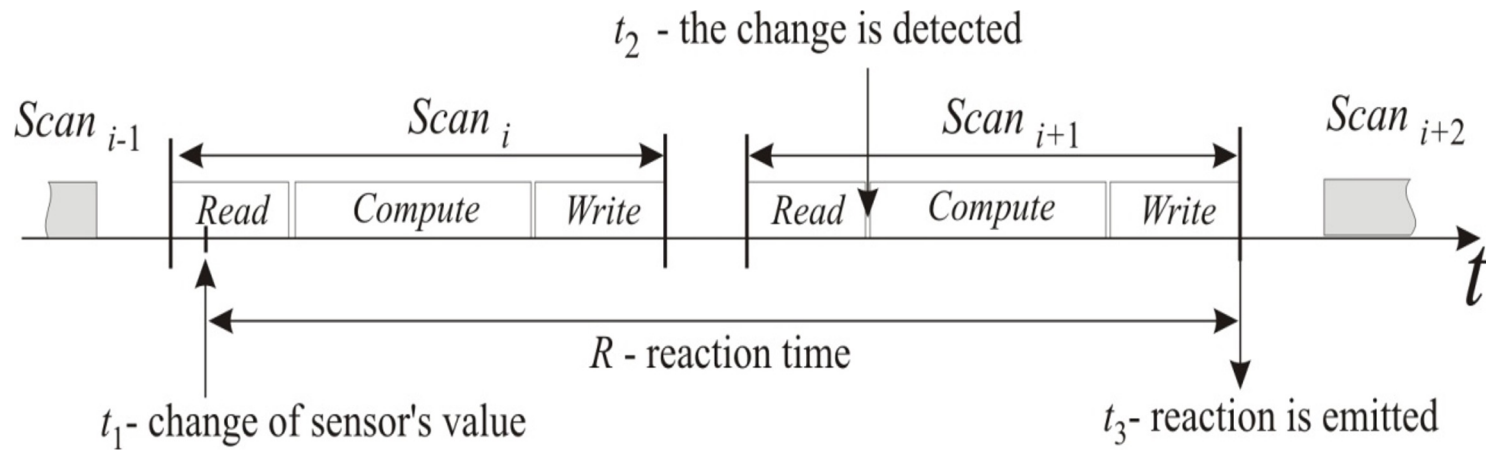
$$\begin{cases} S(i+1) := \mathbf{T}(I(i+1), S(i)); \\ O(i+1) := \mathbf{F}(S(i+1)), \end{cases}$$

Mealy type state machine:

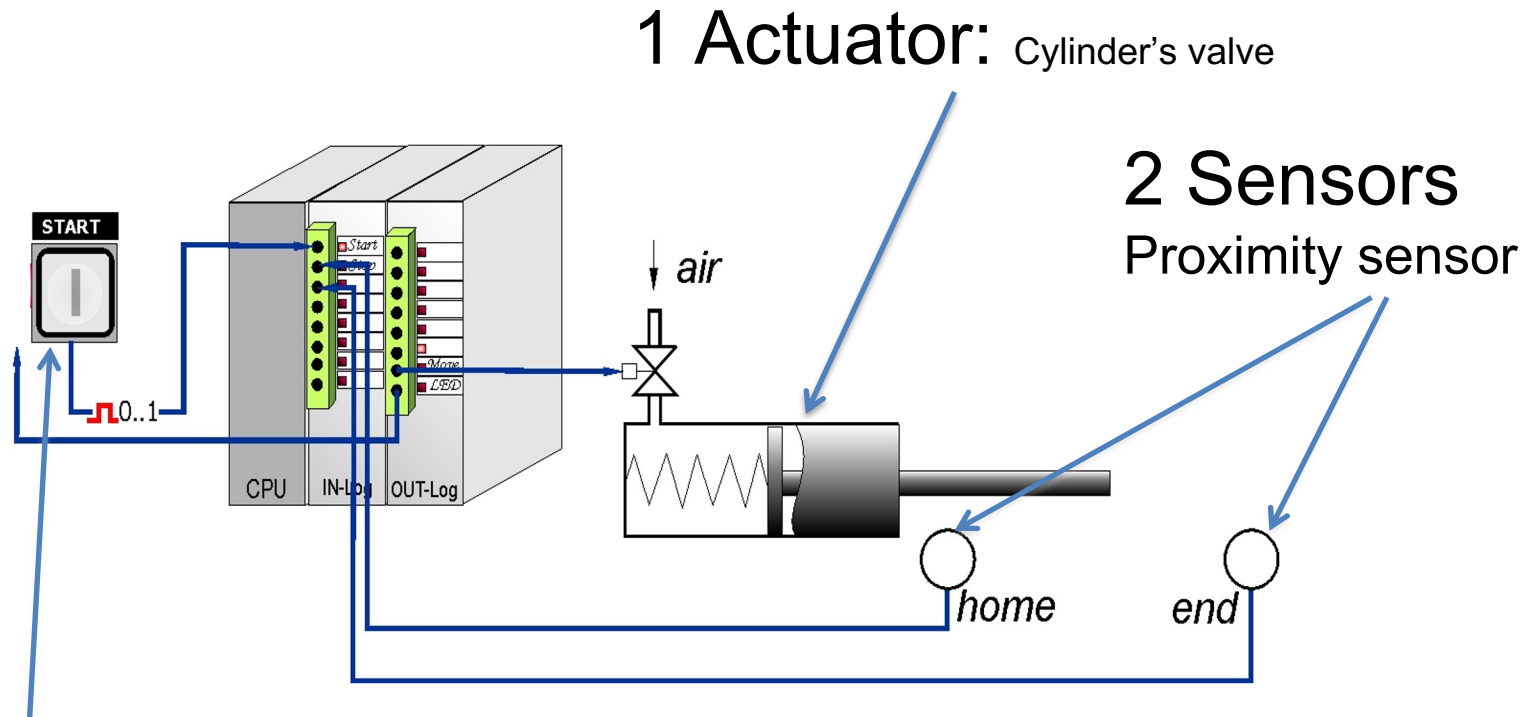
$$\begin{cases} S(i+1) := \mathbf{T}(I(i+1), S(i)); \\ O(i+1) := \mathbf{F}(I(i+1), S(i)); \end{cases}$$



Reaction of PLC



Case-study: Single acting Pneumatic Cylinder

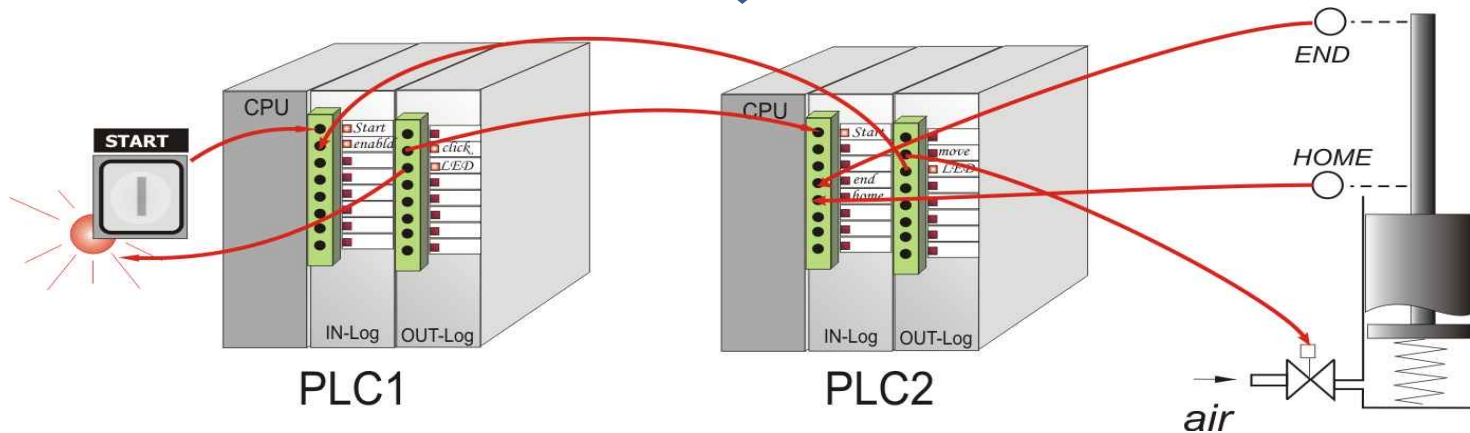
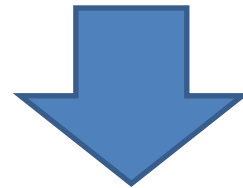
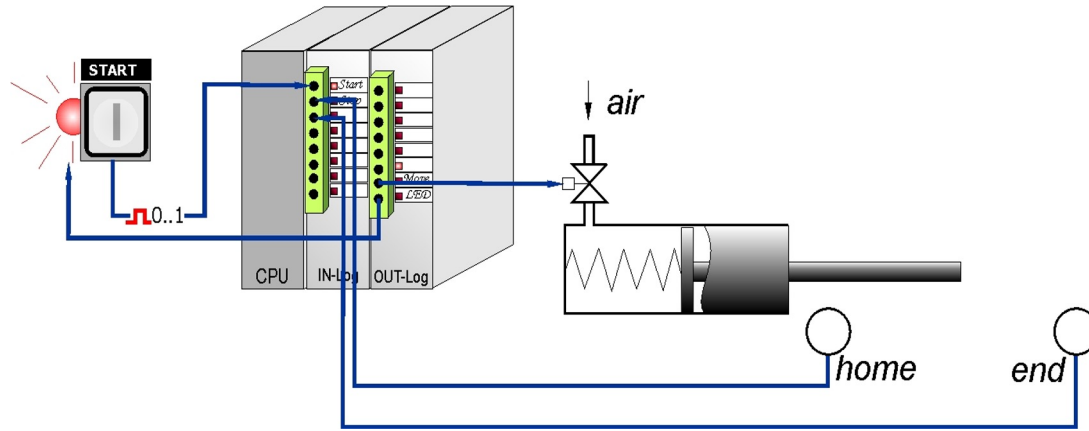


3rd Sensor

A Push Button, which generates the start event

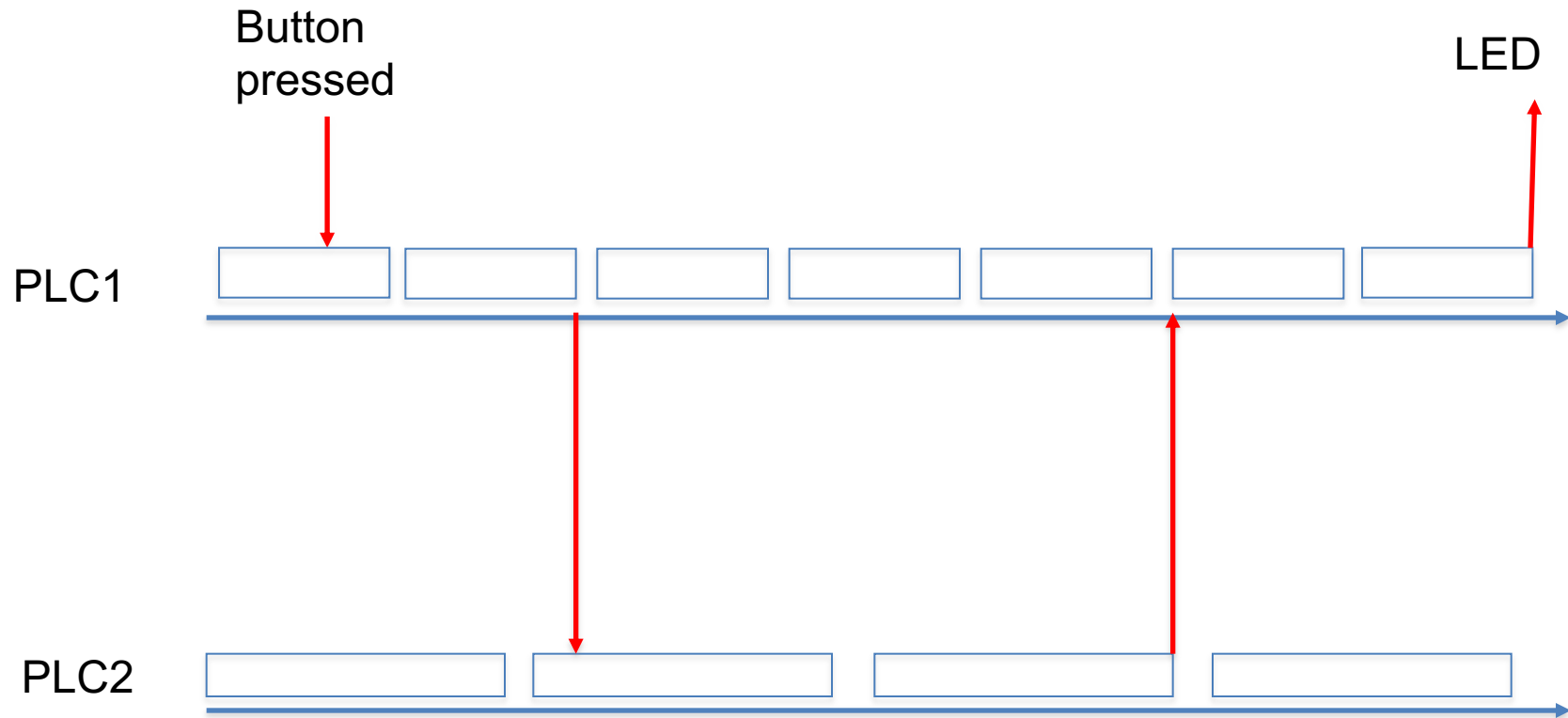
A **Proximity Sensor** is a **sensor** able to detect the presence of nearby objects without any physical contact.

Distributed PLC Systems



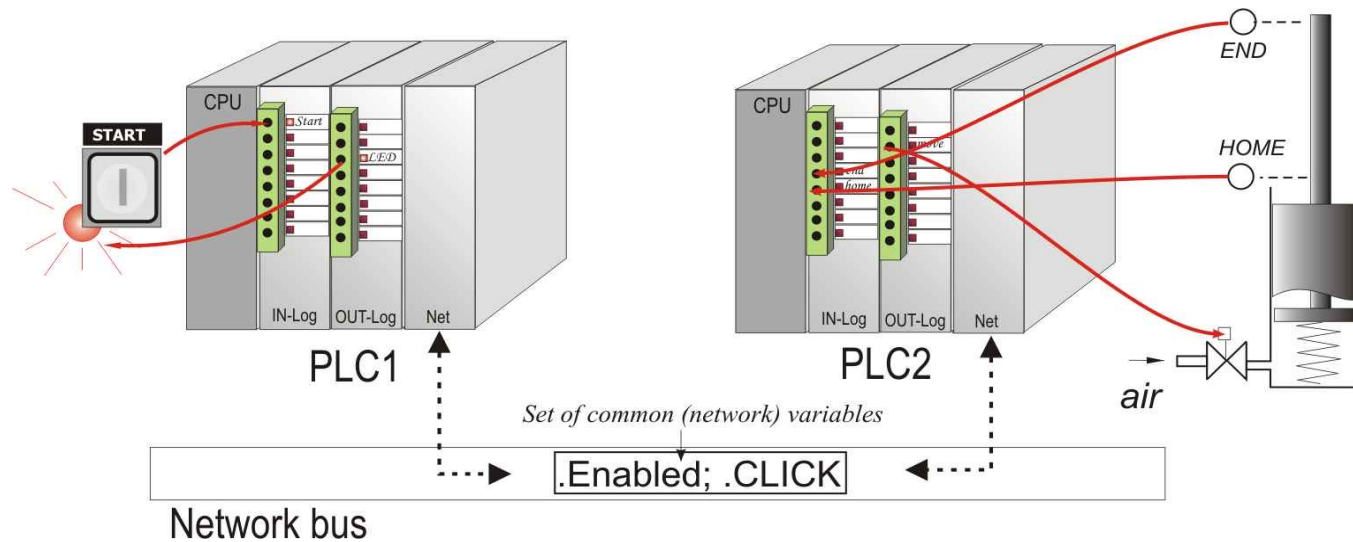
CLICK:= START;
LED:=enable;

Working and Reaction Time

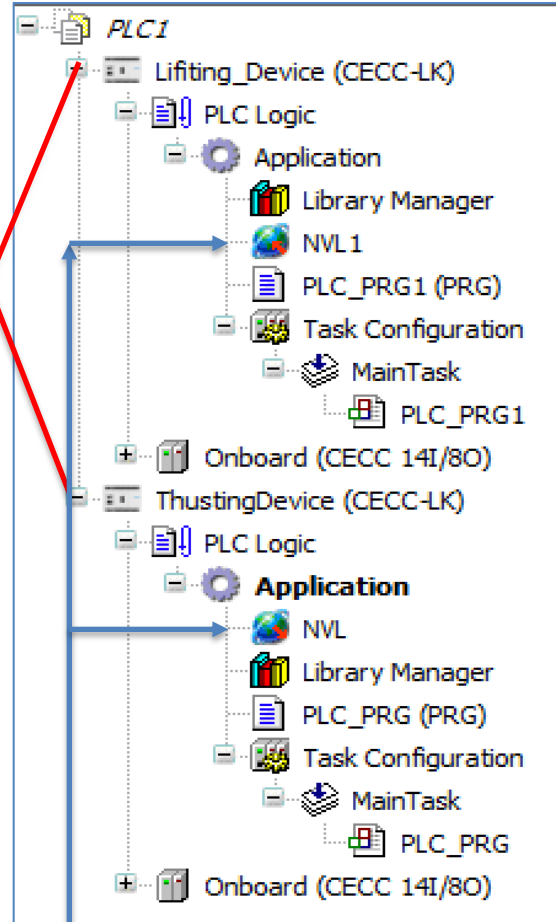
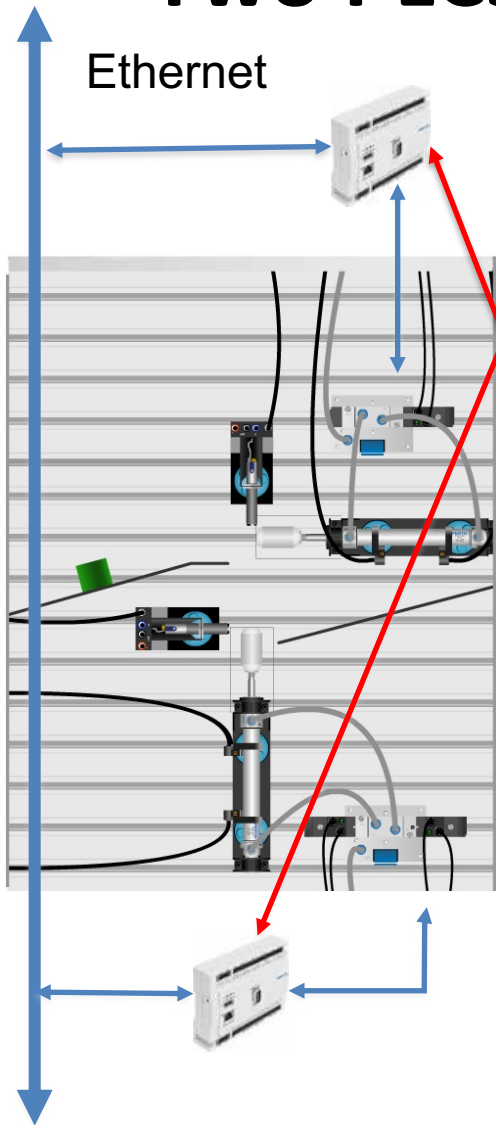


Networking PLCs

Declare `.Enabled` and `.CLICK` as shared variables



Two PLCs in the Lifting Luggage example



Network variable declaration

```

VAR_GLOBAL
  SharedVariable: BOOL;
END_VAR
    
```

```

PROGRAM PLC_PRG1
VAR
  LCExtended : BOOL;
  LCRetracted: BOOL;
  LuggageArrived: BOOL;
  ExtendLC: BOOL;
  RetractLC: BOOL;
END_VAR
IF LuggageArrived AND NOT LCExtended THEN
  ExtendLC := TRUE;
  RetractLC := FALSE;
ELSIF LCExtended AND SharedVariable THEN
  ExtendLC := FALSE;
  RetractLC := TRUE;
END_IF
    
```

Sensors (LCExtended, LCRetracted, LuggageArrived)
Actuator (ExtendLC, RetractLC)

```

PROGRAM PLC_PRG
VAR
  TCRetracted: BOOL;
  TCExtended: BOOL;
  LuggageRaised: BOOL;
  ExtendTC: BOOL;
  LuggageAway: BOOL;
END_VAR
IF LuggageRaised AND TCRetracted THEN
  ExtendTC:= TRUE;
  LuggageAway := FALSE;
  SharedVariable := FALSE;
ELSIF TCExtended AND NOT LuggageRaised THEN
  ExtendTC:= FALSE;
  SharedVariable := TRUE;
END_IF
    
```

Sensors (TCRetracted, TCExtended, LuggageRaised)
Actuator (ExtendTC, LuggageAway)

Using PLCs in a Distributed System: Working

The screenshot displays a software interface for developing a PLC application. It is divided into several main sections:

- Devices:** A tree view on the left showing the project structure. The 'Application [run]' folder is expanded, showing various programs like 'LiftCylProd (PRG)', 'ThrustCylProd (PRG)', and 'VisualizationManager'.
- Code Editor:** The central pane shows the ladder logic for 'Device.Application.LiftCylProd'. The code is as follows:

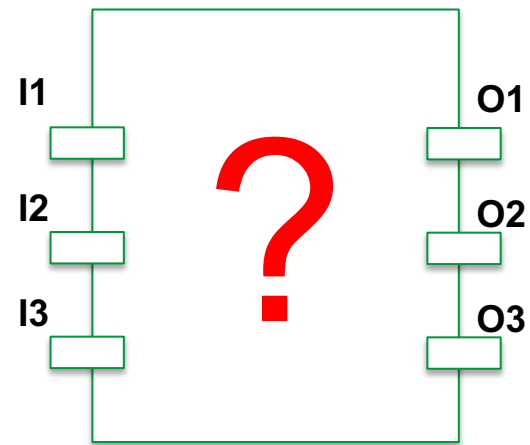
```
1 IF LuggageArrived FALSE AND NOT LCExtended TRUE THEN
2   ExtendLC TRUE := TRUE;
3   RetractLC FALSE := FALSE;
4 ELSIF LCExtended TRUE AND SharedVar FALSE THEN
5   ExtendLC TRUE := FALSE;
6   RetractLC FALSE := TRUE;
7 END_IF RETURN
```
- Configuration:** A window titled 'Lifting_Device' is open, showing settings for the task. The 'Priority (0..31)' is set to 10. The 'Type' is 'Cyclic' with an 'Interval (e.g. t#200ms):' of t#100ms. The 'Watchdog' section has 'Enable' unchecked and 'Time (e.g. t#200ms):' set to an empty field with 'ms' as the unit. 'Sensitivity' is set to 1.
- Visualization:** A window on the right shows a 3D schematic of a lifting mechanism. Below the schematic is a legend for testing purposes:
 - Extend lifting cylinder
 - Retract lifting cylinder
 - Extend thrust cylinder

At the bottom of the interface, a status bar indicates: 'Messages - Total 4 error(s), 0 warning(s), 4 message(s)'.

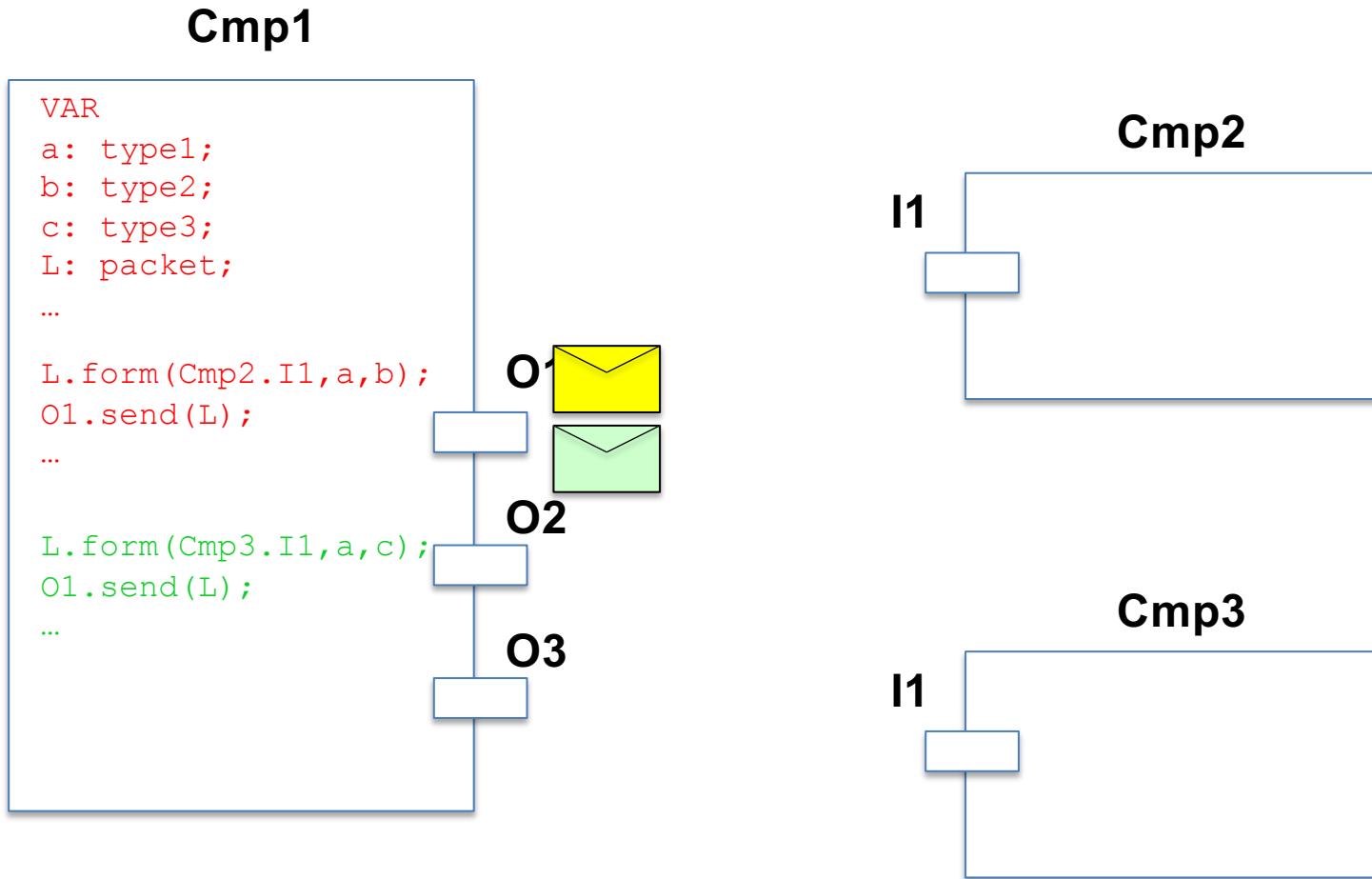
Intelligent Automation Component



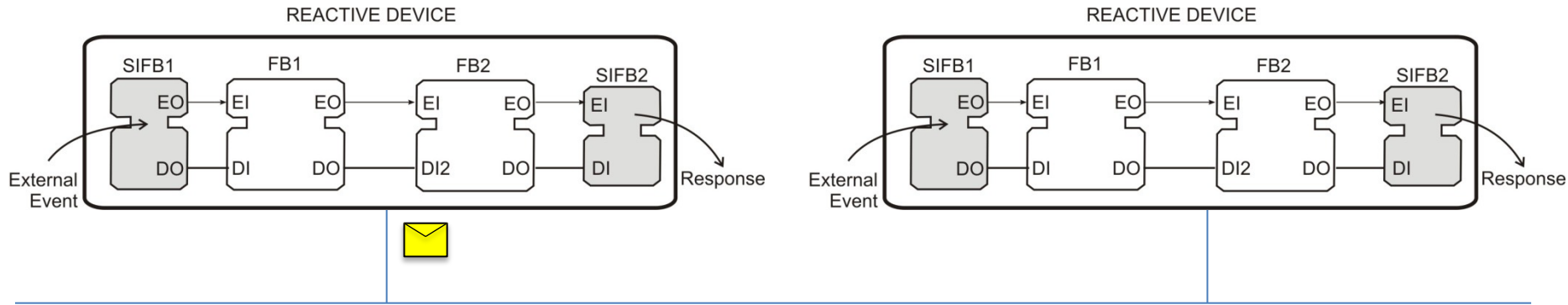
Cylinder Software Component



Communicating components



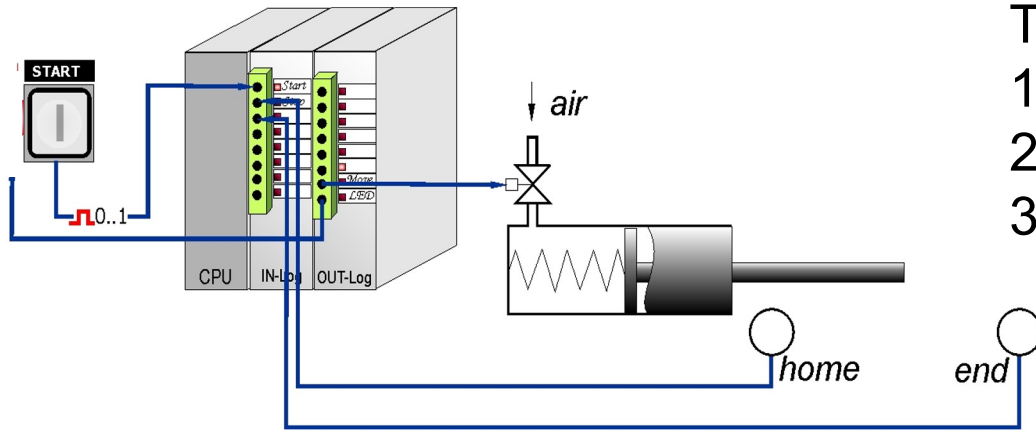
Event-driven interaction



Application Logic Design

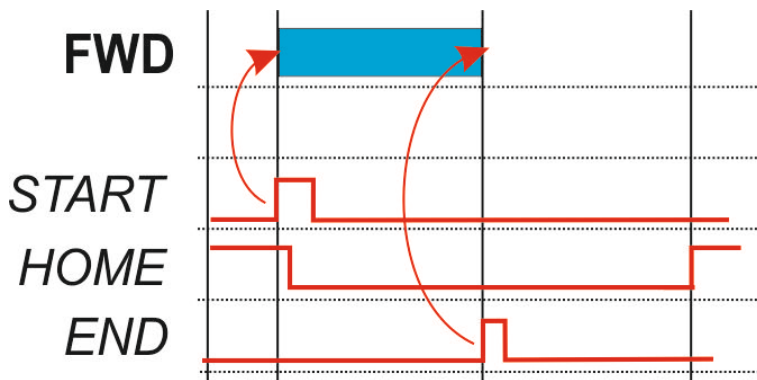
Example: Pneumatic Cylinder with Return Spring

Single acting cylinder has only one control signal **FWD** (move forward)



This system has 3 input signals

- 1- HOME
- 2- START
- 3- END

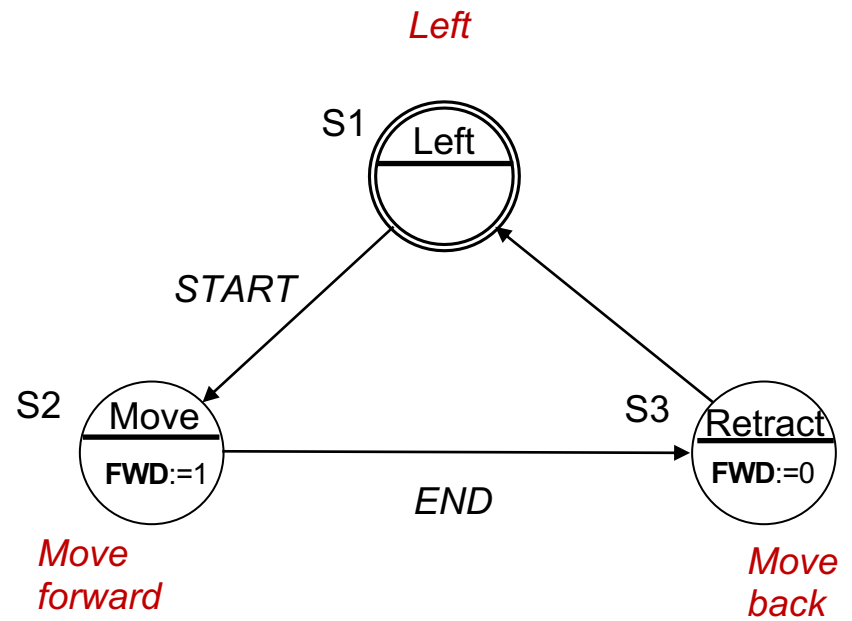
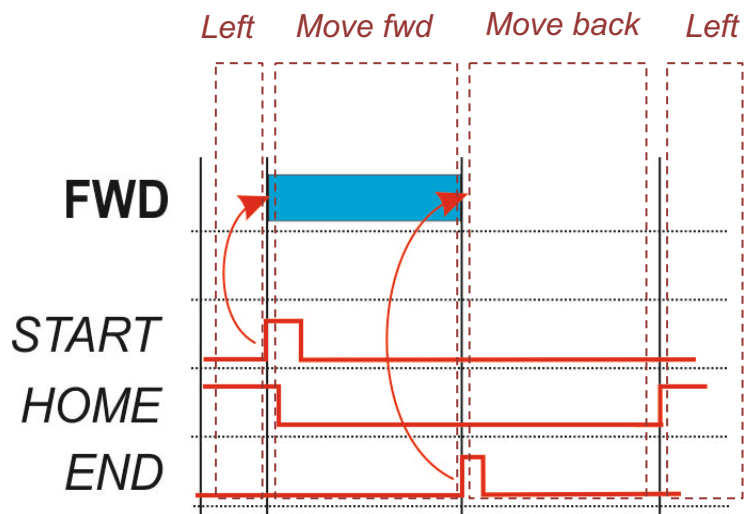


Actions and Signals:

- **HOME and END** signals are generated by 2 proximity sensors that are located in the 2 ending positions
- **START** signal is generated by the Start button.

Resulting state-machine

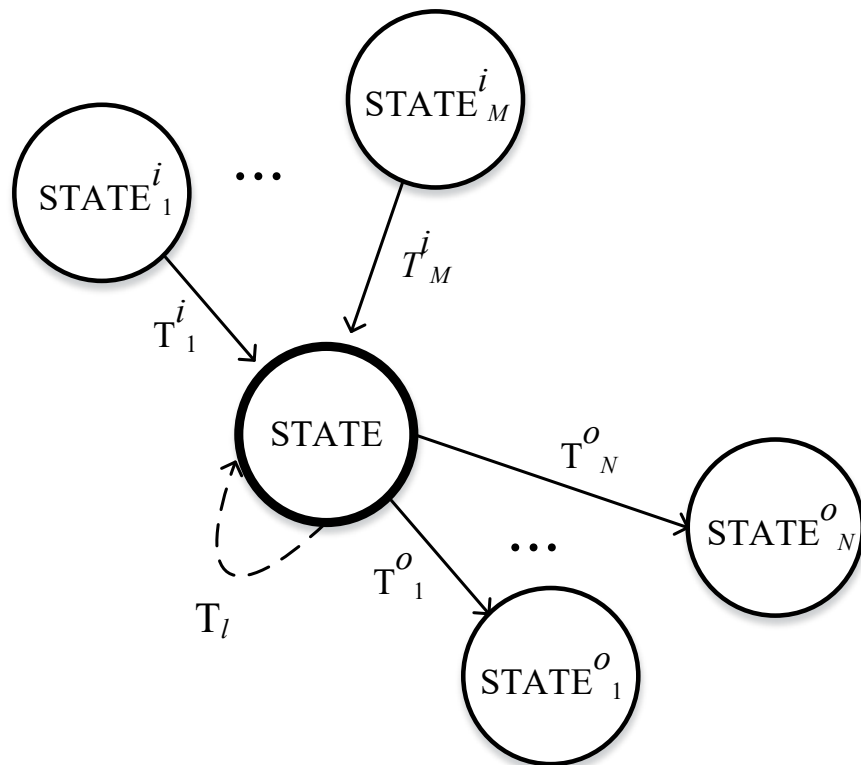
Scenario



Algorithm

1. Identify stable states in the system's behavior.
 - A valid scenario from the system functionality is a good starting point
2. For each state define the output signals that shall be true in that state.
3. Define transitions from state to state.
4. Define transitions conditions

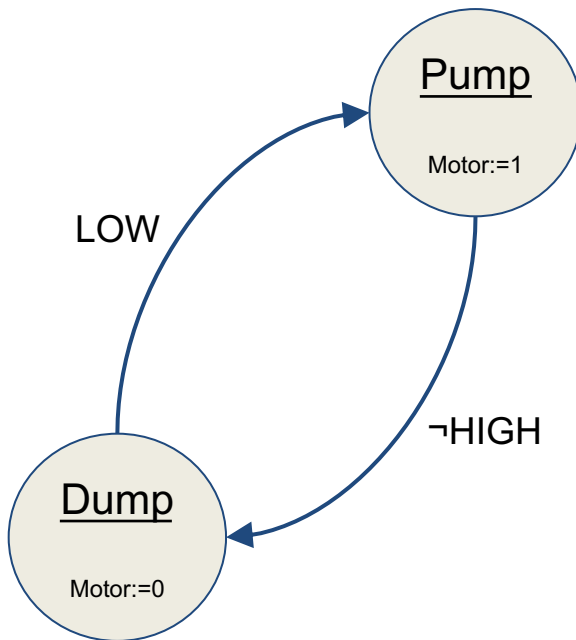
State machine design conventions



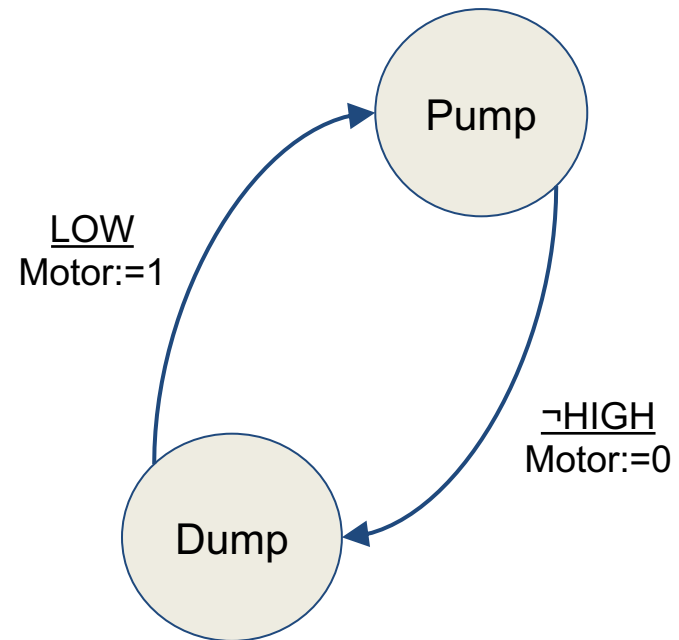
1. We are drawing state machines without loopback arcs.
2. Completeness of the outgoing conditions is guaranteed by the ELSE assumption
 1. at least one outgoing transition is true.
3. Orthogonality of outgoing transitions
 1. at most one outgoing transition is true
 2. provision of determinism.

Moore vs. Mealy state machines

Moore-type FSM

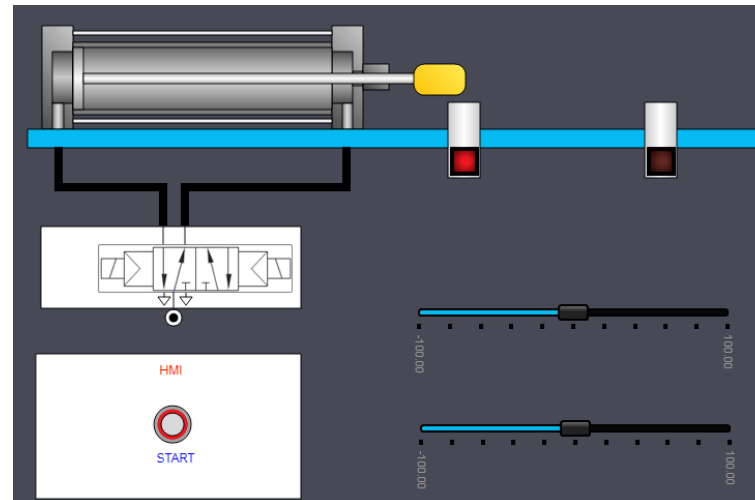
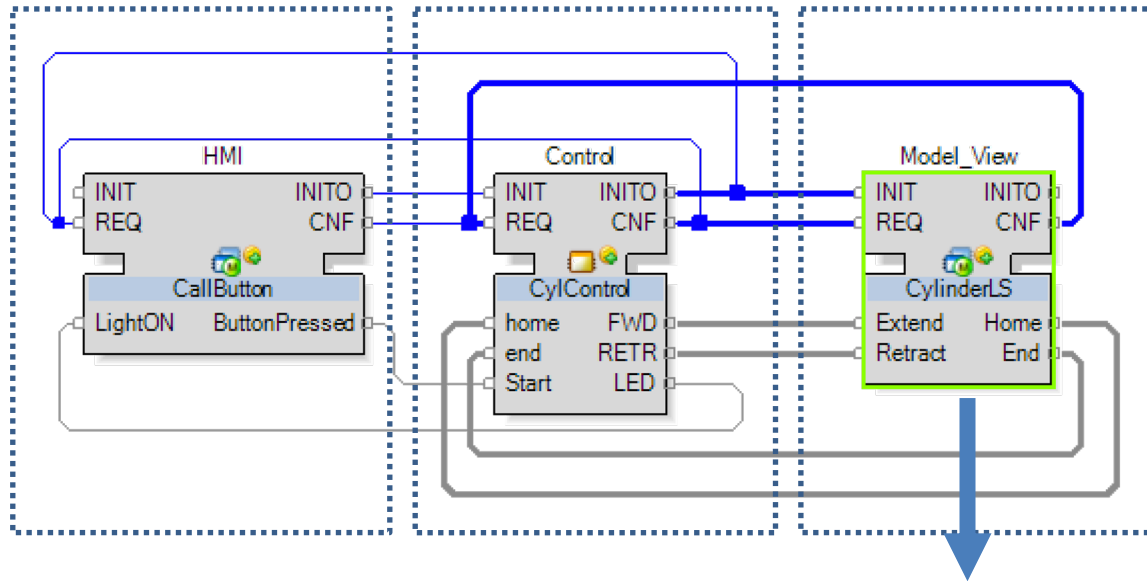


Mealy-type FSM



We will be using Moore machines throughout this lecture.

Simulation-based testing framework



What to remember?

- Generations of automation systems.
- What is the advantage of flat architecture compared to the “ICT pyramid”?
- Why intelligent machines are needed at the factory floor?
- What is main difference of combinatorial logic from state-based logic?
- What are pros and contras of using simulation in the loop?