Course Outline

- ~~ICS OVERVIEW~~
  - ~~Terms & Definitions~~
  - ~~Generic architectures~~
  - ~~History of ICS~~

- **Hands on: Basic PLC Programming**
  - Creating a first Flowchart-based program
  - Creating visualisation

- **Commonly used ICS protocols**
  - Overview of ICS protocols
  - Security considerations for commonly used protocols
  - Hands-on: Wireshark captures

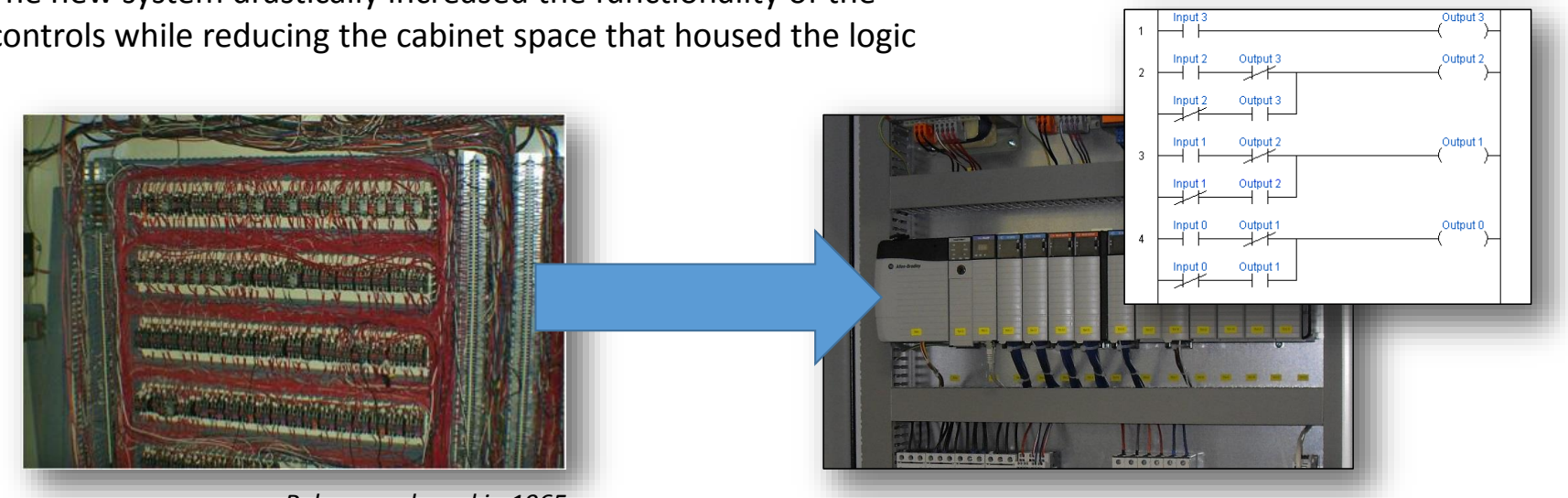
- **Introduction to ICS Security**
  - Basics of a ICS security penetration test
  - Red team Excercise & Demo's

Hands on: Basic PLC Programming

- **Hardware overview**
- Creating a first PLC program
- Creating visualisation

Birth of the PLC

- Originally logical Control Systems were designed and built exclusively around **electromechanical relays**
- First PLC designed by Modicon as a **relay re-placer** for GM and Landis
  - These controllers eliminated the need for rewiring and adding additional hardware for each new configuration of logic
  - The new system drastically increased the functionality of the controls while reducing the cabinet space that housed the logic

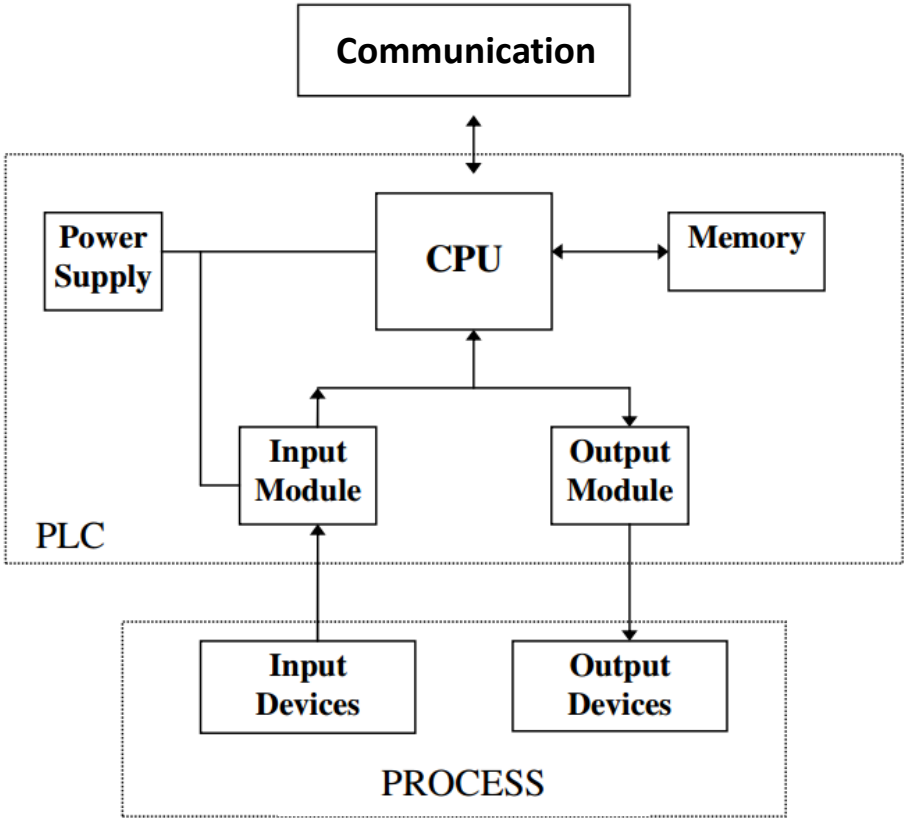


Relay panel used in 1965

Basic building blocks of a PLC

A PLC can be divided in four (five) building blocks

- Input/output modules
- Central Processing Unit
- Memory
- Communication
  - Programming Terminal
  - Other Automation Devices
- (Power supply)



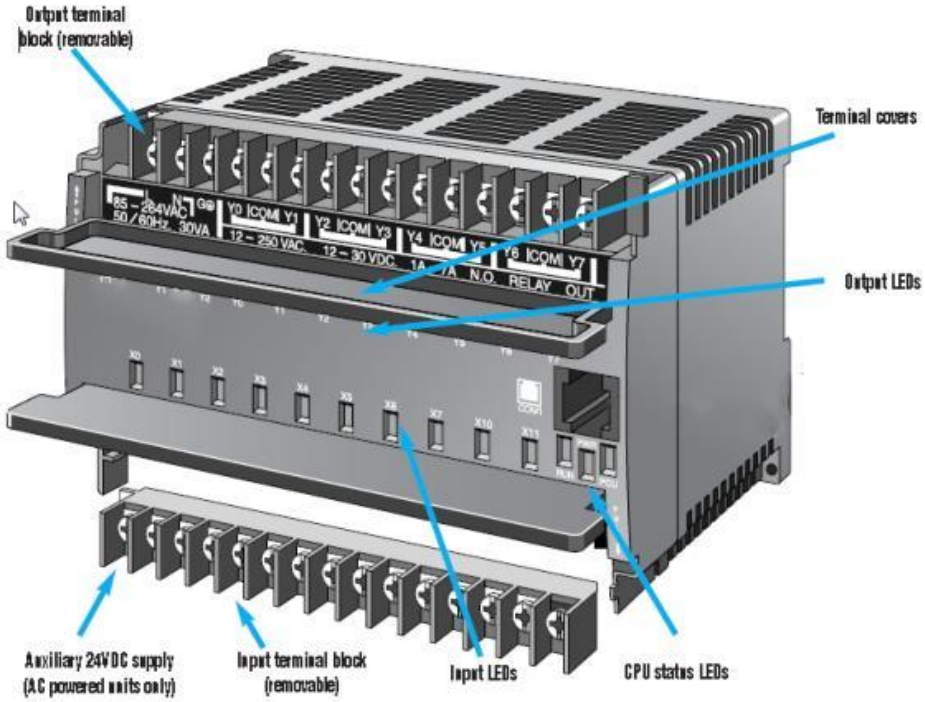
Basic building blocks of a PLC

Specific features required for industrial control

Robust
- Rugged, noise immune equipment
- Modular plug-in construction, allowing easy replacement or addition of units

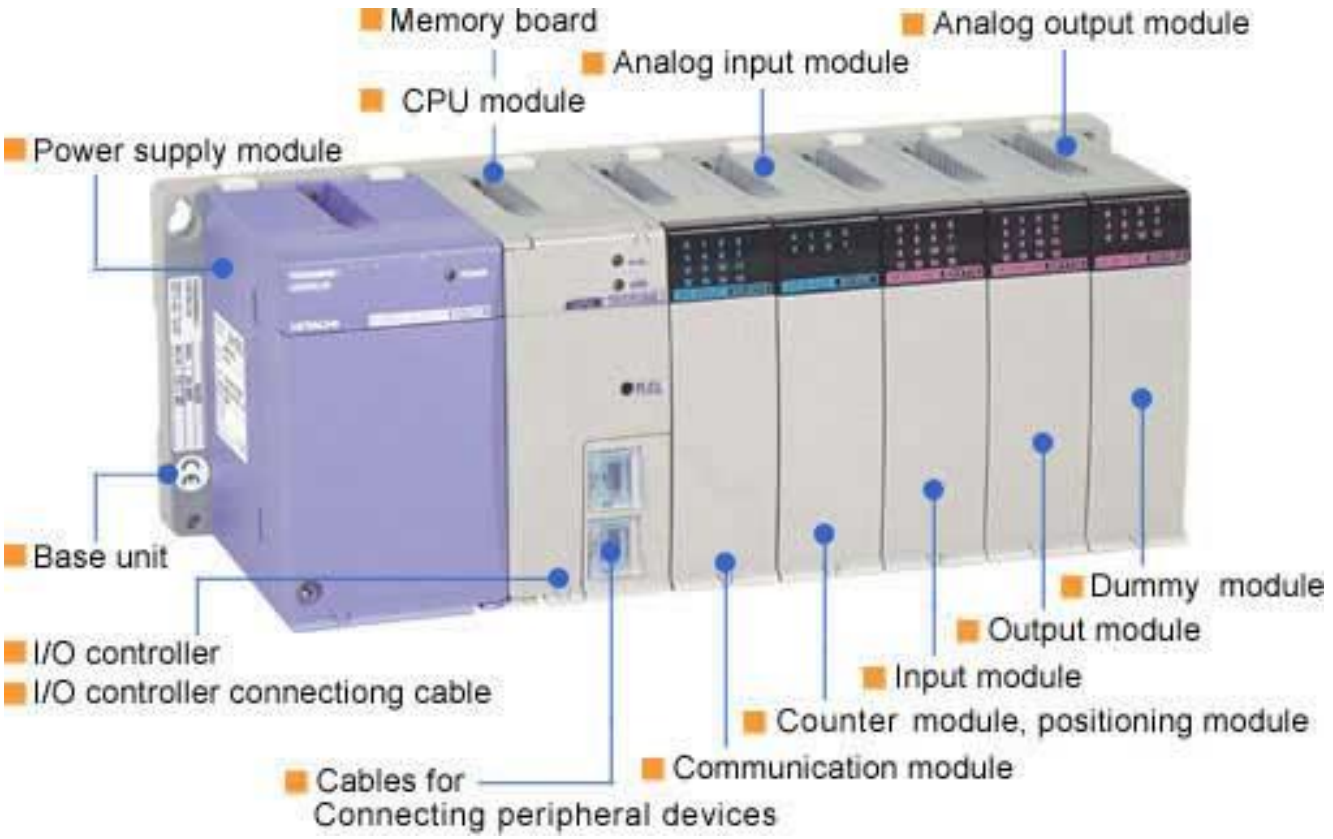
Simple to use
- Standard input/output connections and signal levels
- Easily understood programming language
- Ease of programming and reprogramming in-plant

- Capable of communicating with other PLCs, computers and intelligent devices
- Competitive in both cost and space occupied with relay and solid-state logic systems

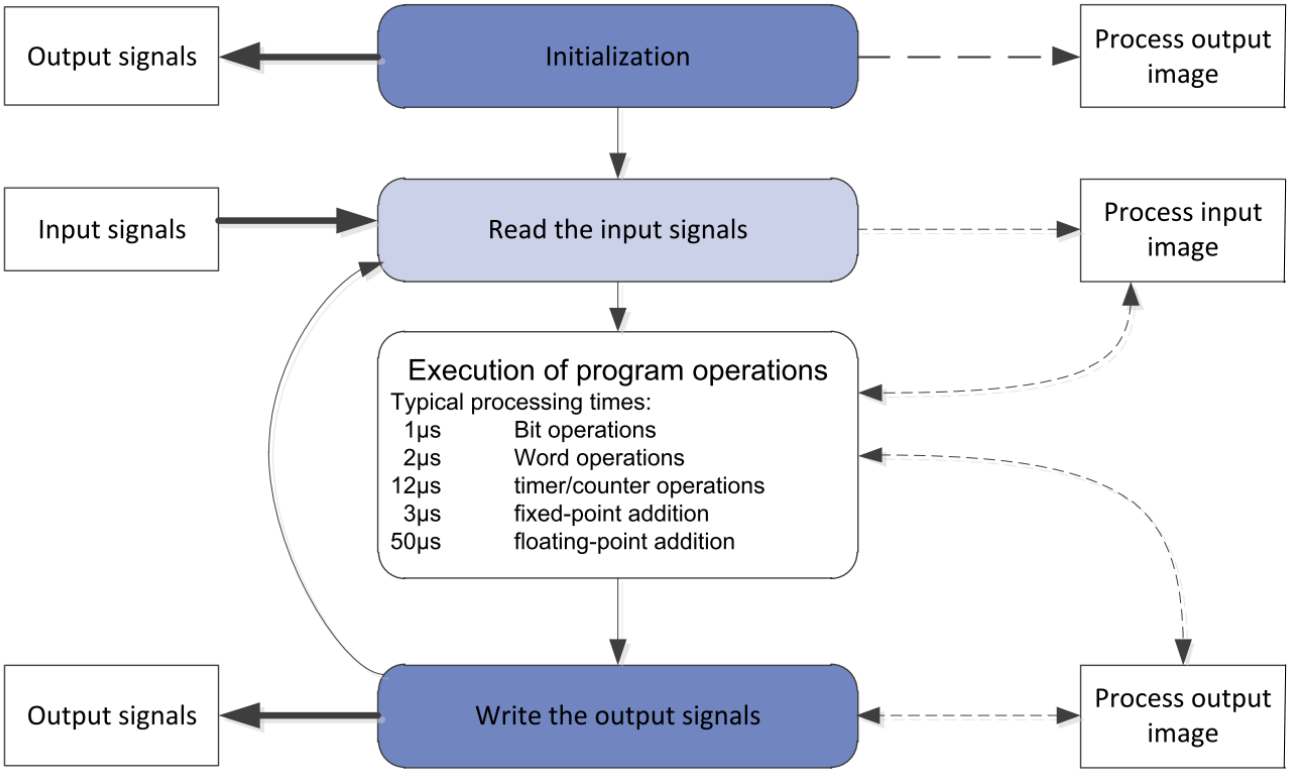


Basic PLC programming: Hardware overview

Basic building blocks of a PLC



PLC work flow

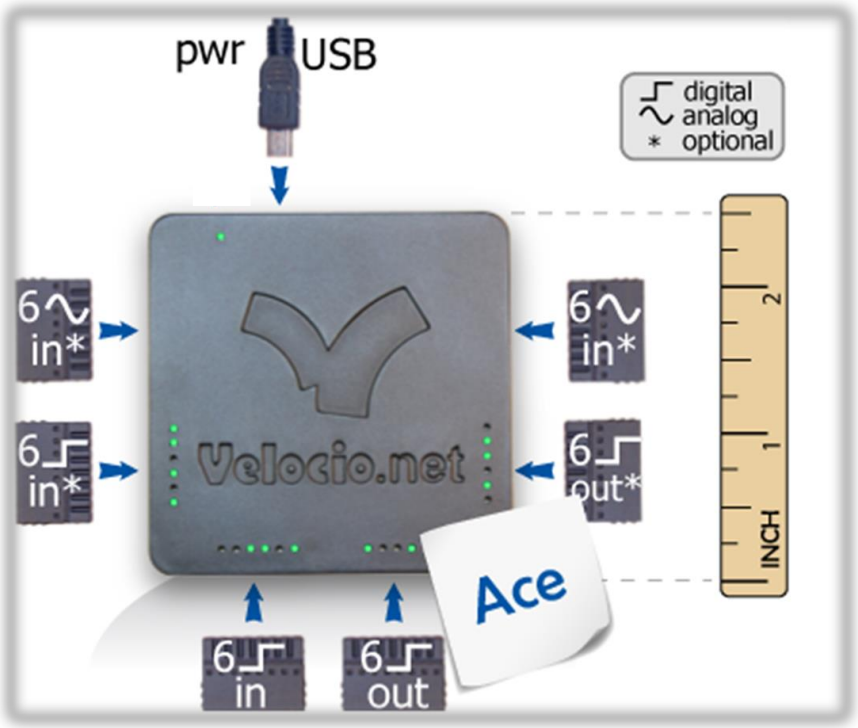


Basic PLC programming: Hardware overview

Hardware used for hands-on:

Basic PLC programming: Hardware overview

Hardware used for hands-on:



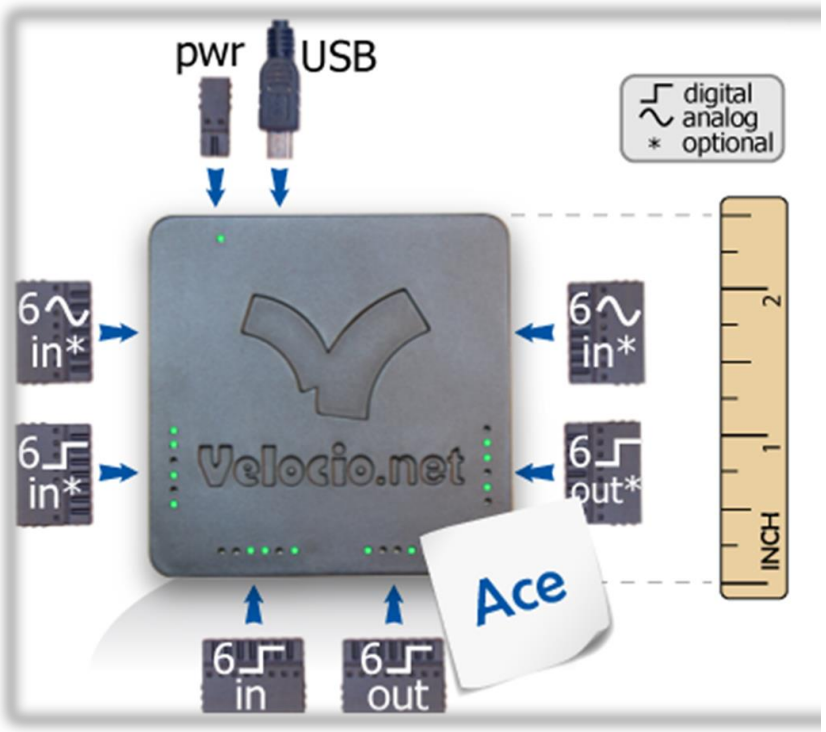
Ace 11

- 6 Digital Inputs
  - Protected, 3-30VDC
  - High speed pulse counting capability
- 6 Digital Outputs
  - Sinking transistor outputs
  - Step and direction motion control enabled
  - PWM capable on all outputs
- USB programming port
- USB port Modbus RTU slave enabled
- **USB POWERED***

Only our version! Not Default !

Basic PLC programming: Creating a first Flowchart-based program

Hardware used for hands-on:



Hardware Specifications

Power :
- Voltage: 4.75 - 5.5VDC
- current: 300mA maximum; < 100mA typical

Digital Inputs :
- Type : DC voltage input
- Input range : 3 to 30 VDC
- Input low (or 0) signal : 0 to 0.8V, or open connection
- Input high (or 1) siganl : 3 to 30VDC
- Pulse counter input frequency : up to 100 KHz (typical); up to 250 KHz (maximum)

Digital Outputs :
- Type : Sinking transistor
- Voltage range : 3 to 30VDC
- Current : 500 mA maximum
- Motion output pulse frequency: 0 to 100 KHz (typical); 0 to 250 KHz (maximum)

Analog Inputs :
- Type : v5 = 0 to 5VDC; v10 = 0 to 10VDC; c = 0 to 20 mA
- resolution : 12 bit

Analog Outputs :
- Types : selectable ; 0-5V, 0-10V, 0-20mA
- resolution : 16 bit

Thermocouple Inputs :
- Types : selectable ; J, K, T and N
- Output value : floating point value in degrees C

Communications :

- Upstream : USB Device; mini USB connector
- RS232 : 3 wire (TX, RX and ground)
  - baud rates: selectable ; 9600 baud; 19200 baud; 38400 baud; 57600 baud
  - parity: selectable
  - stop bits: selectable

Physical Dimensions :

2.5"H x 2.5"W x 0.5" deep

Environmental

Operating temperature : -40 to +85C

Software Specifications

Application Program Limits (in Ace PLC)

Program Memory :		34K Words
Maximum rungs or function blocks		4K
Maximum # Subroutines		68
Maximum Tagnames		950
Main Program data memory		
Bits		2,048
unsigned 8 bit integers		512
unsigned 16 bit integers		512
signed 16 bit integers		512
signed 32 bit integers		256
floating point numbers		256
Object Memory (used for subroutine data)		
object words		4,096
object bits	up to	65,536
object 8 bit integers	up to	8,192
object signed 16 bit	up to	4,096
object unsigned 16 bit	up to	4,096
object signed 32 bit	up to	2,048
object floating point	up to	2,048
Maximum # objects		292

Terminal Block Connections

Terminal type	Socket connectors and Spring cage capture plug
Terminal spacing :	2.50 mm
Wire AWG	20 to 26 AWG*

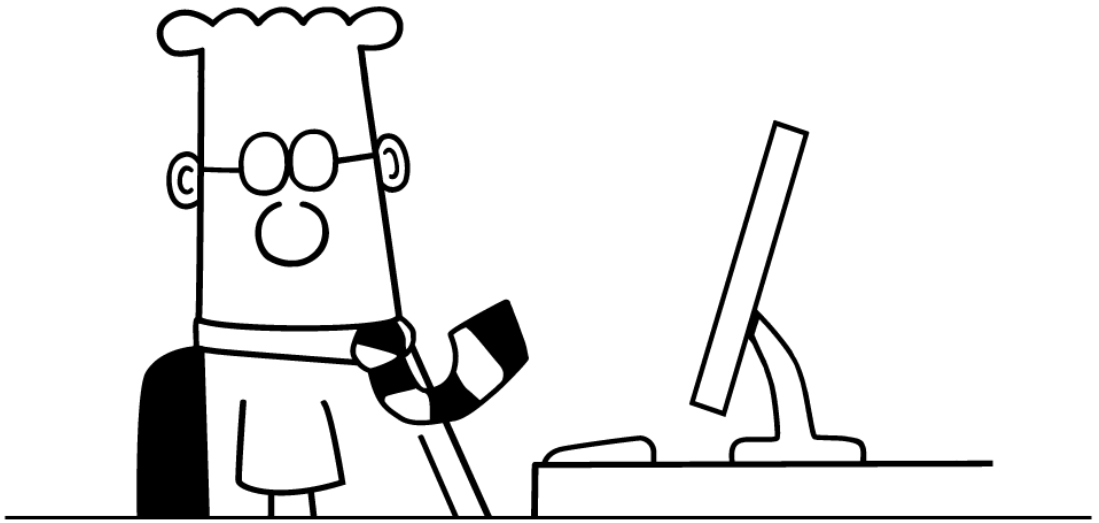
* best wire fit is with 22 or 24 AWG

Ports included :

- Ace 11 : Ports C, D and power
- Ace 22 : Ports B, C, D, E and power
- Ace 222 : Ports A, B, C, D, E, F and power
- Ace 3090 : Ports A, B, C, D, E, F and power
- Ace 5190 : Ports A, B, C, D, E, F and power

Hands on: Basic PLC Programming

- Hardware overview
- **Creating a first PLC program**
- Creating visualisation



PLC programming Languages

IEC 61131-3, the third part of the open standard for PLCs, defines **five programming language**s

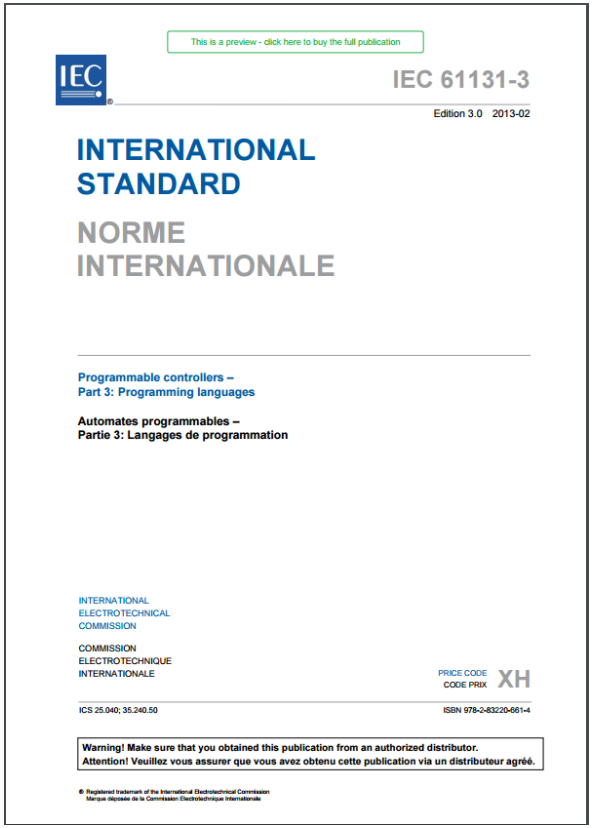
- Instruction List Programming (IL)
- Structured Text Programming (ST)
- Functional Block Programming (FBD)
- Ladder Logic Programming (LD)
- Sequential Function Charts (SFC)

Text: Instruction List Programming (IL), Structured Text Programming (ST)

Graphical: Functional Block Programming (FBD), Ladder Logic Programming (LD), Sequential Function Charts (SFC)

IEC 61131-3 Defines **Elementary Data types for a PLC**

- Bit Strings
- Integers
- Reals
- Time
- Date and Time
- String
- Arrays
- …



PLC programming Languages

Instruction List Programming (IL)

- Low level language (resembles assembly)
- Program control achieved by jump instructions and function calls
- Is a 'accumulator oriented' language
  - Each instruction uses or alters the current content of the accumulator (a form of internal cache)
- Each instruction begins on a new line and consists of:
  - a label *(optional)*
  - An operator *(here Opcode)*
  - If necessary a modifier
  - If necessary one or more operands

Label	Opcode	Operand	Comment
START:	LD	%I:000/00	(* Load input bit 00 *)
	AND(	%I:000/01	(* Start a branch and load input bit 01 *)
	OR(	%I:000/02	(* Load input bit 02 *)
	AND	%I:000/03	(* Load input bit 03 and invert *)
	)		
	)		
	ST	%O:001/00	(* SET the output bit 00 *)

read as O:001/00 = I:000/00 AND (I:000/01 OR (I:000/02 AND I:000/03))

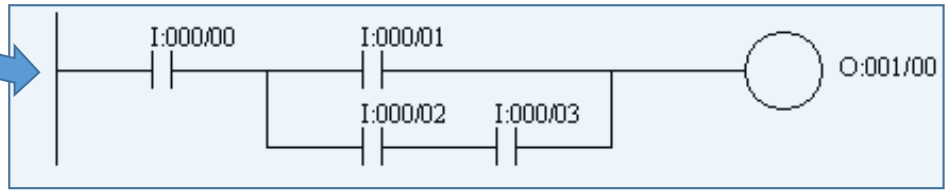
PLC programming Languages

Instruction List Programming (IL)

- Low level language (resembles assembly)
- Program control achieved by jump instructions and function calls
- Is a 'accumulator oriented' language
  - Each instruction uses or alters the current content of the accumulator (a form of internal cache)
- Each instruction begins on a new line and consists of:
  - a label *(optional)*
  - An operator *(here Opcode)*
  - If necessary a modifier
  - If necessary one or more operands

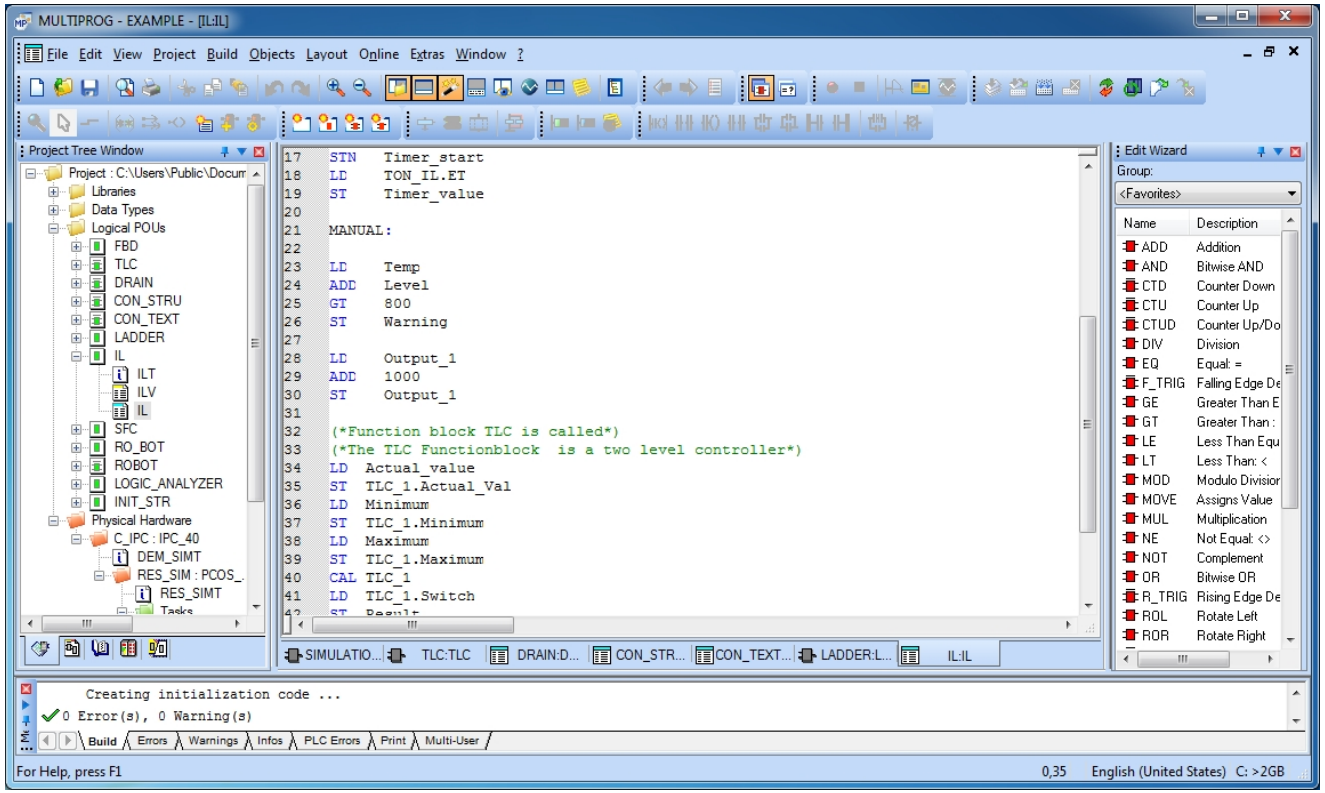
Label	Opcode	Operand	Comment
START:	LD	%I:000/00	(* Load input bit 00 *)
	AND(	%I:000/01	(* Start a branch and load input bit 01 *)
	OR(	%I:000/02	(* Load input bit 02 *)
	AND	%I:000/03	(* Load input bit 03 and invert *)
	)		
	)		
	ST	%O:001/00	(* SET the output bit 00 *)

read as O:001/00 = I:000/00 AND (I:000/01 OR (I:000/02 AND I:000/03))



PLC programming Languages

Instruction List Programming (IL)



PLC programming Languages

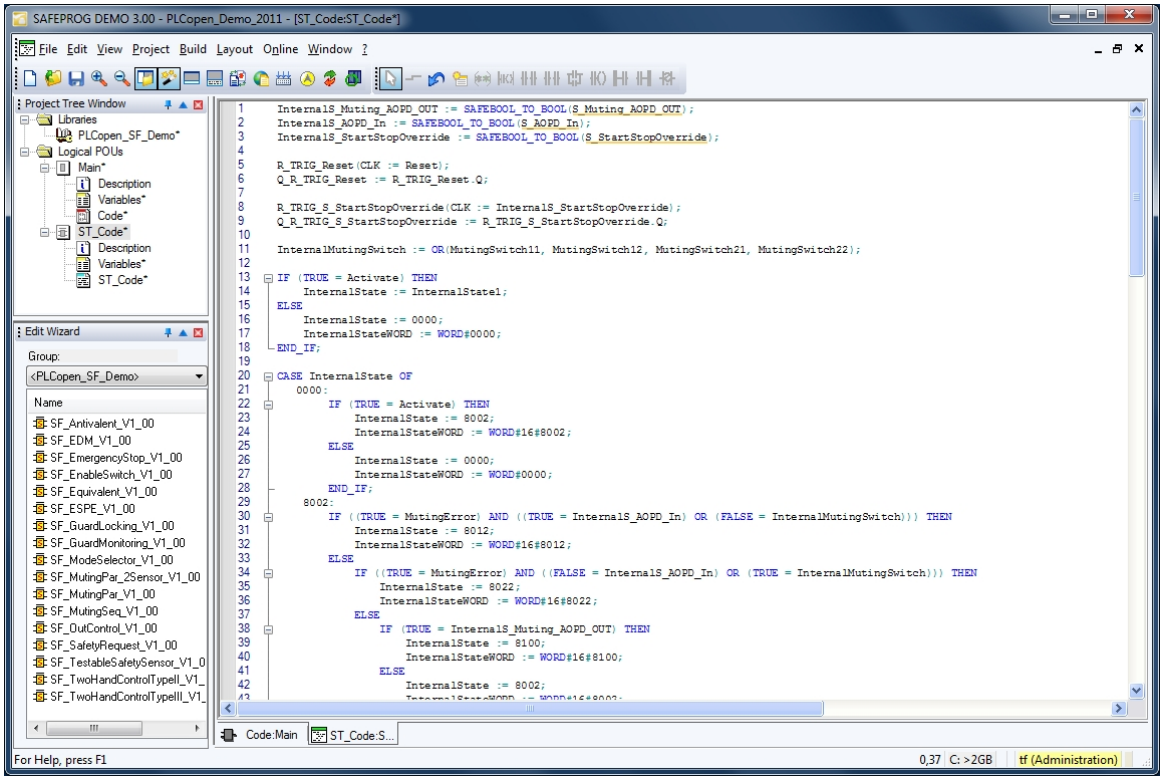
Structured Text Programming (ST)

- High level programming language (based on Pascal)
- Ability to use control flow statements to conditionally execute, loop over or jump to statements in the program
- Allows for the following structures:
  - *IF-THEN-ELSIF-ELSE-END_IF*
  - *CASE-value:-ELSE-END_CASE*
  - *FOR-TO-BY-DO-END_FOR*
  - *TAN(A), COS(A), A**B,…*
- Used Syntax
  - All Statements are divided by semicolons
  - Not case-sensitive
  - Spaces have no function (but should be used for user reliability)

```
PROGRAM main
VAR
      i : INT;
END_VAR
i := 0;
REPEAT
      i := i + 1;
      UNTIL i >= 10;
END_REPEAT;
END_PROGRAM
```

PLC programming Languages

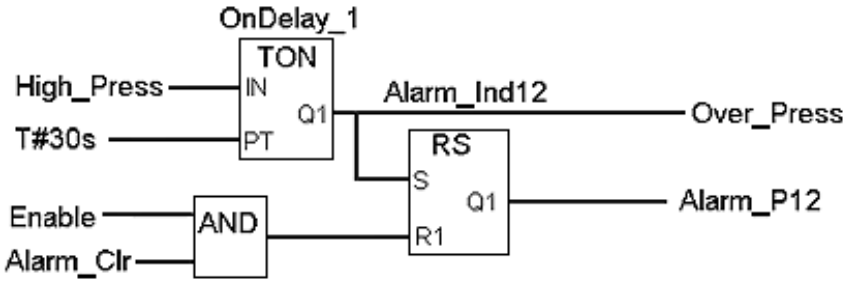
Structured Text Programming (ST)



PLC programming Languages

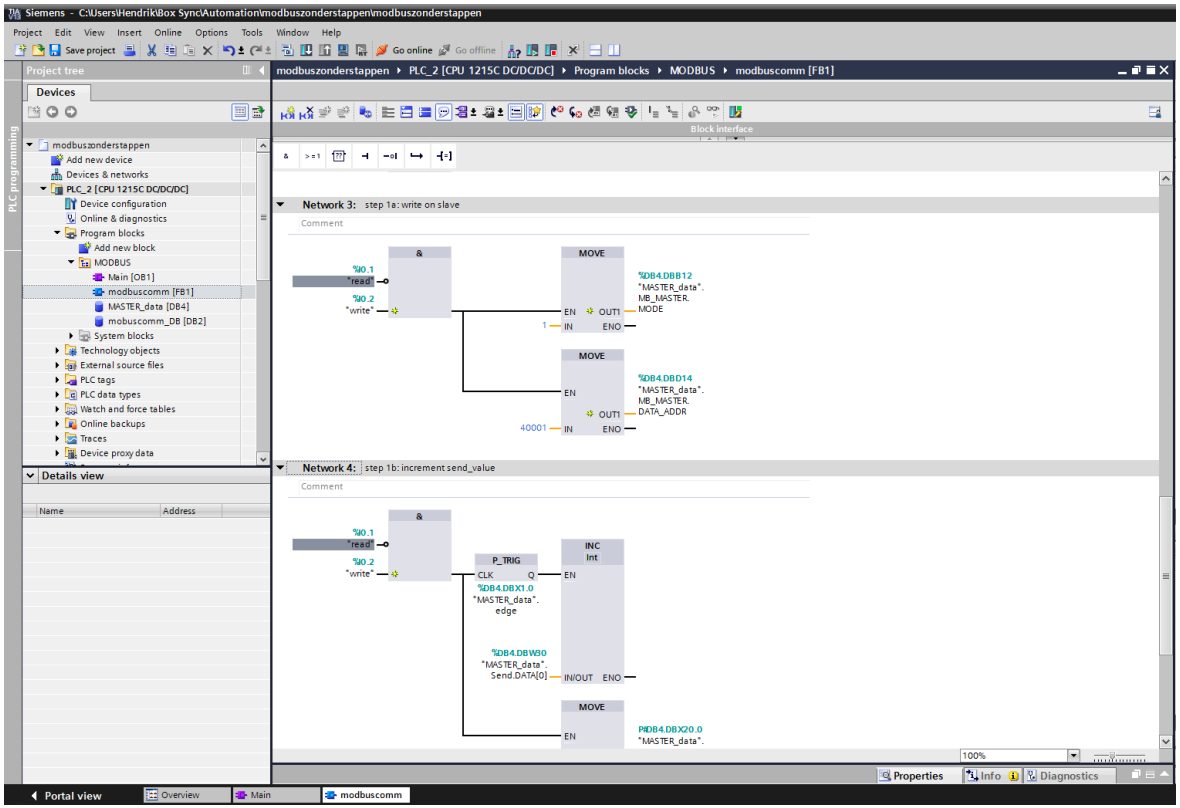
Functional Block Programming (FBD)

- A Graphical language
- A function is described as a set of elementary Blocks
- Input and output variables are connected to blocks by connection lines
- Connections are oriented (data is carried from left to right)
- Programmers can make their own Function Blocks (and reuse them in other projects)
- Function blocks can represent actual pieces of equipment (with encapsulated I/O)



PLC programming Languages

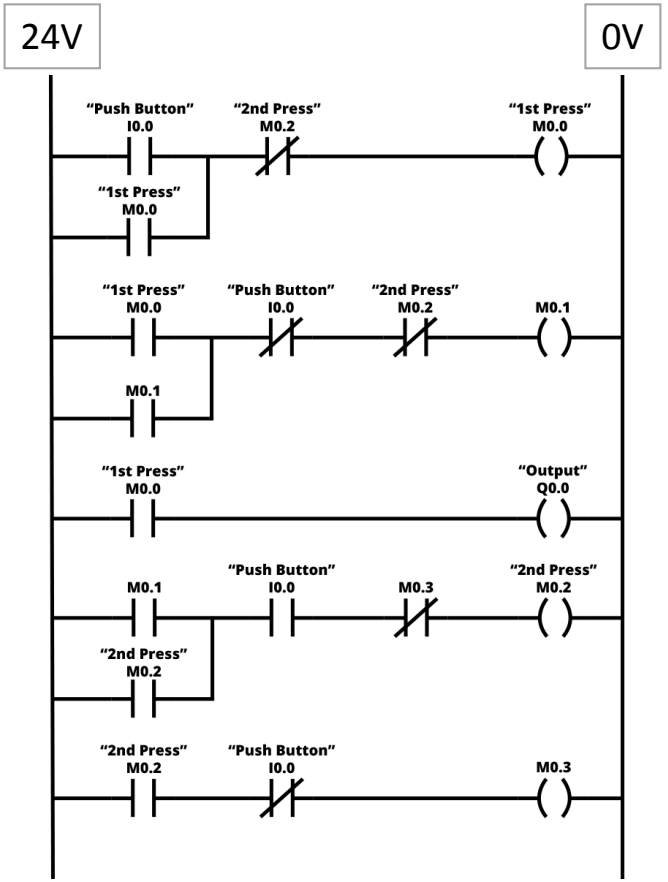
Functional Block Programming (FBD)



PLC programming Languages

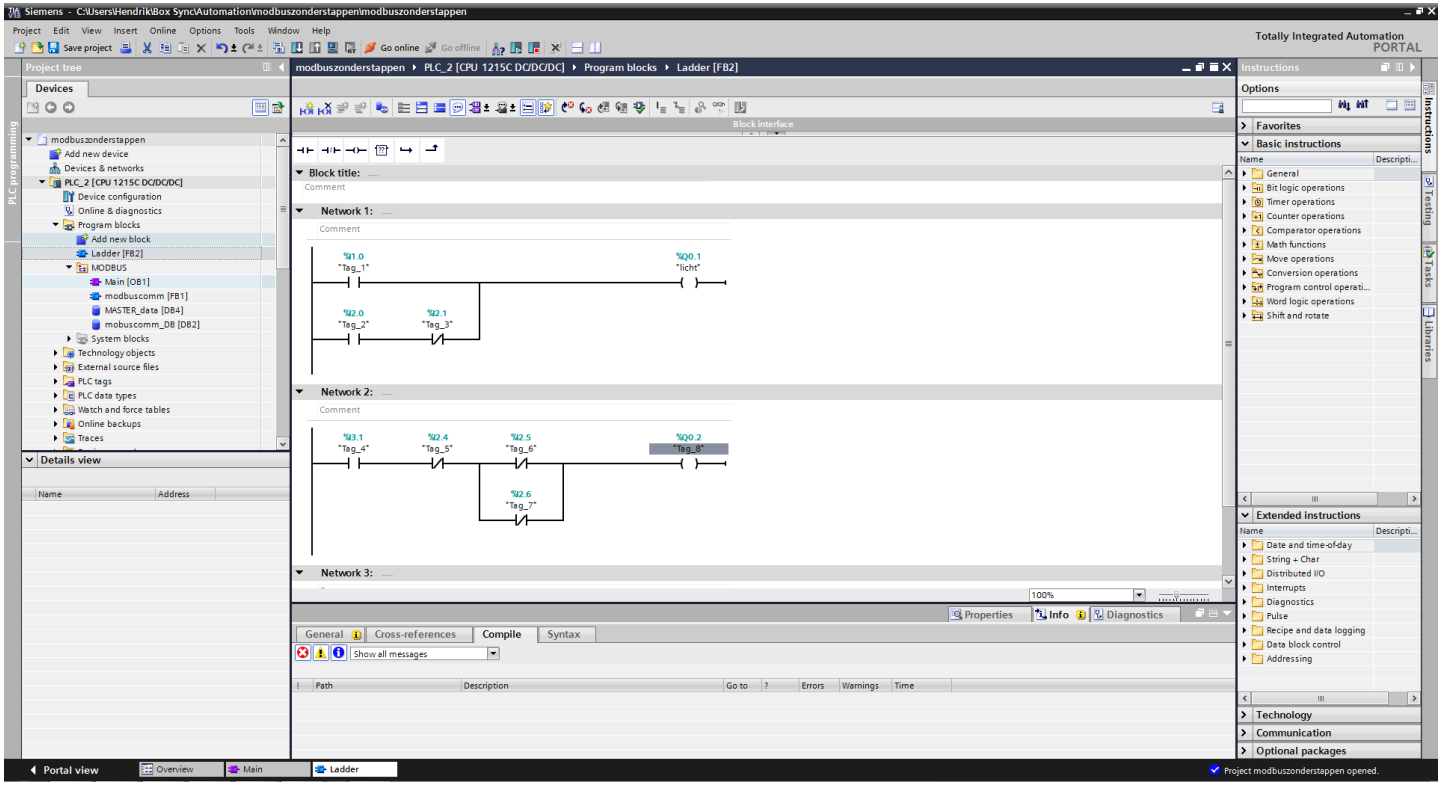
Ladder Logic Programming (LD)

- Originally written to document the design and construction of relay racks
- Each device in the rack would be represented by a symbol on the ladder diagram
- Evolved into a graphical programming language (First language implemented in PLCs)
- Ladder logic had contacts that make or break circuits to control coils
  - Each coil corresponds to the status of a single bit in the programmable controller's memory (and can be used in multiple "rungs")



PLC programming Languages

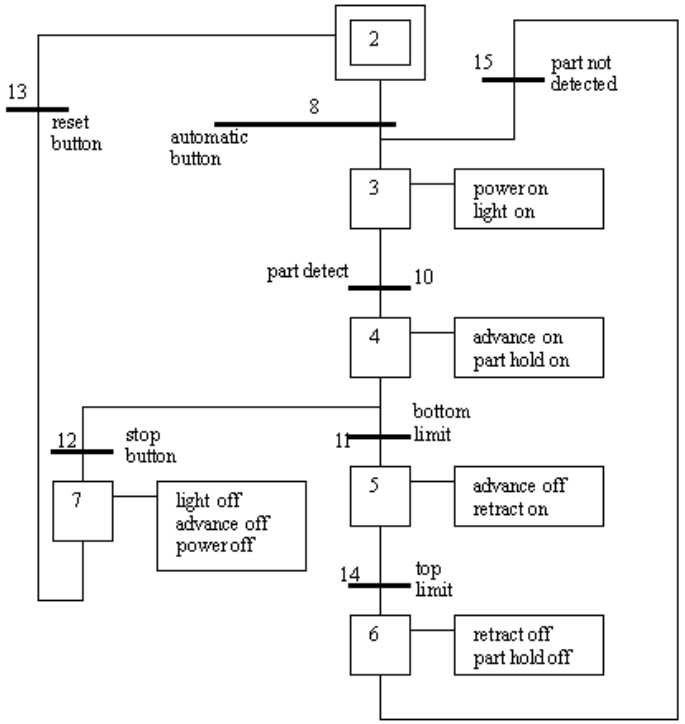
Ladder Logic Programming (LD)



PLC programming Languages

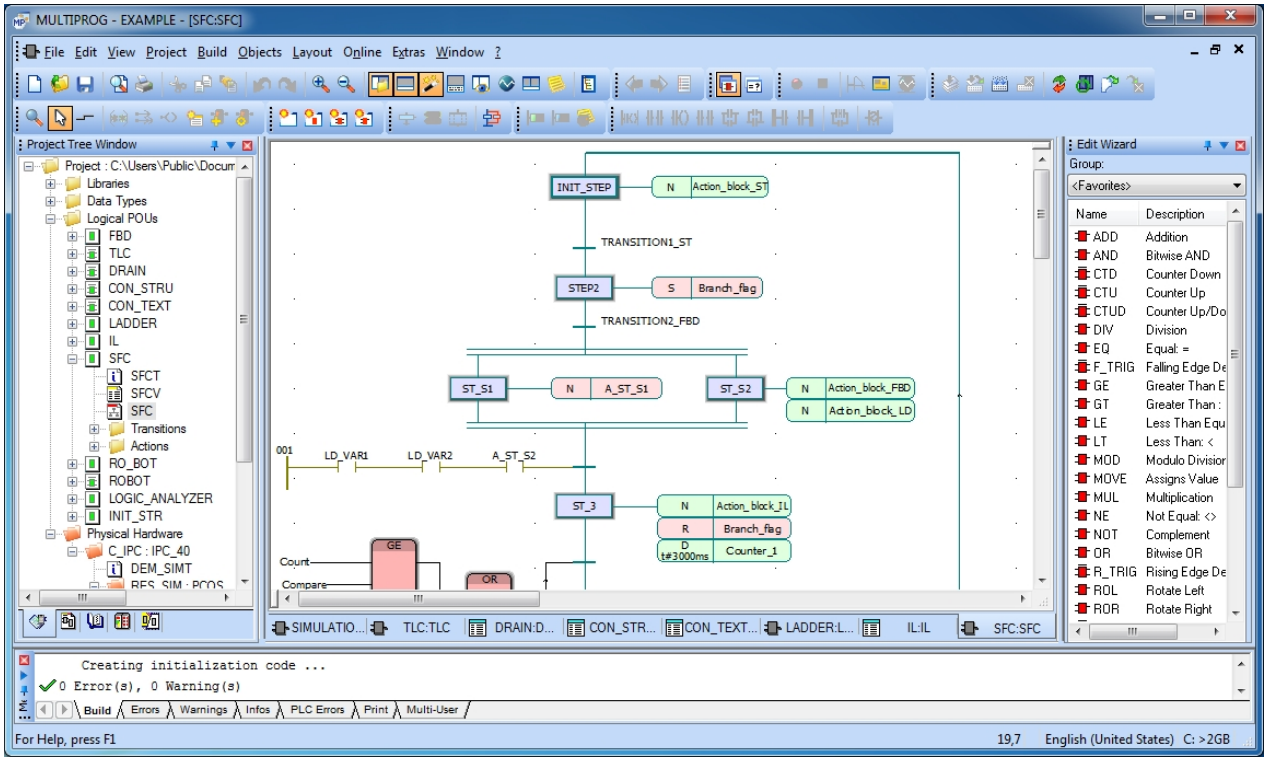
Sequential Function Charts (SFC)

- Graphical programming language
- Breaks sequential tasks down into:
  - Steps
  - Transitions
  - Actions
- These are drawn graphically to describe a sequence of interactions



PLC programming Languages

Sequential Function Charts (SFC)

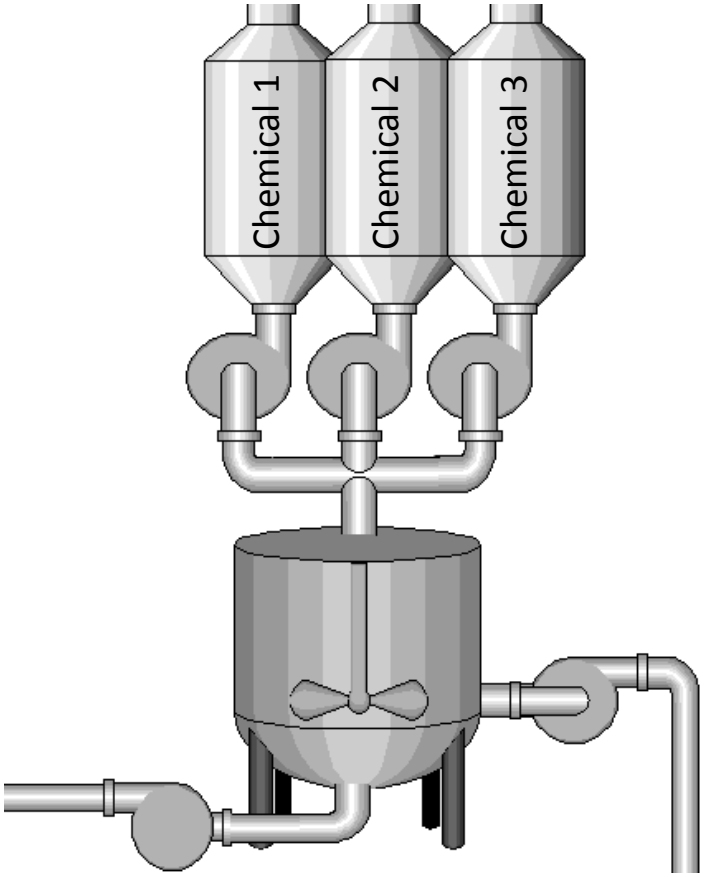


Basic PLC programming: Creating a first Flowchart-based program

Hands-on Creating a basic PLC program

http://velocio.net/software/

Hands-on Creating a basic PLC program

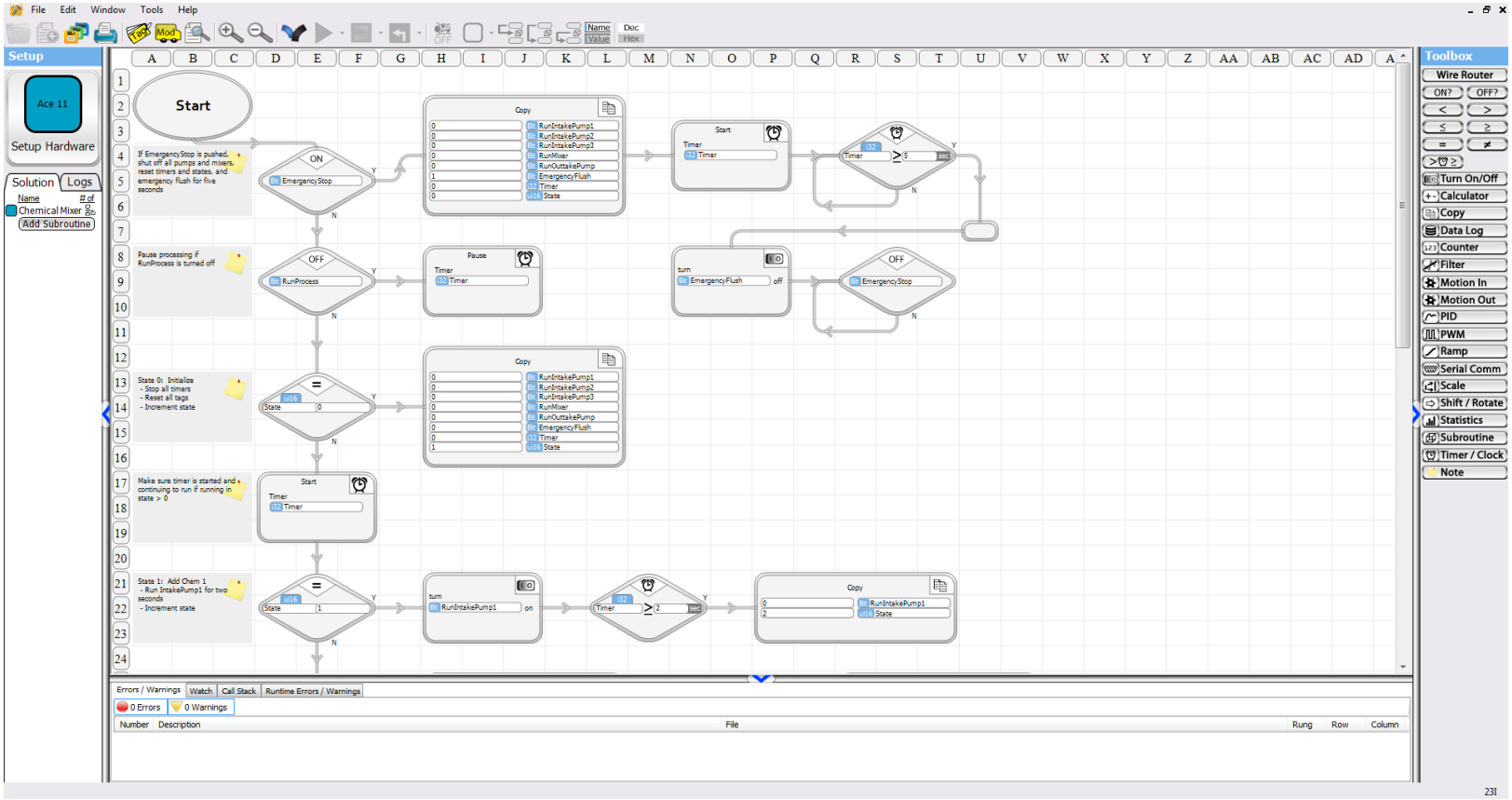


Process Requirements:

- One part chemical 1&2 to two parts chemical 3
  - Chemical 1 and 2 intake pumps must run 2 seconds
  - Chemical 3 intake pump must run 4 seconds
- Chemical 1 must be added before the others
- Chemical 2 and 3 must be added while mixer is running
- After all chemicals are added, they must be mixed for 4 seconds
- One mixed, outtak pump must run 5 seconds to process batch
- RunProcess switch can start and pause the process
- EmergencyStop switch must run emergency flush pump for 5 seconds before resetting systems

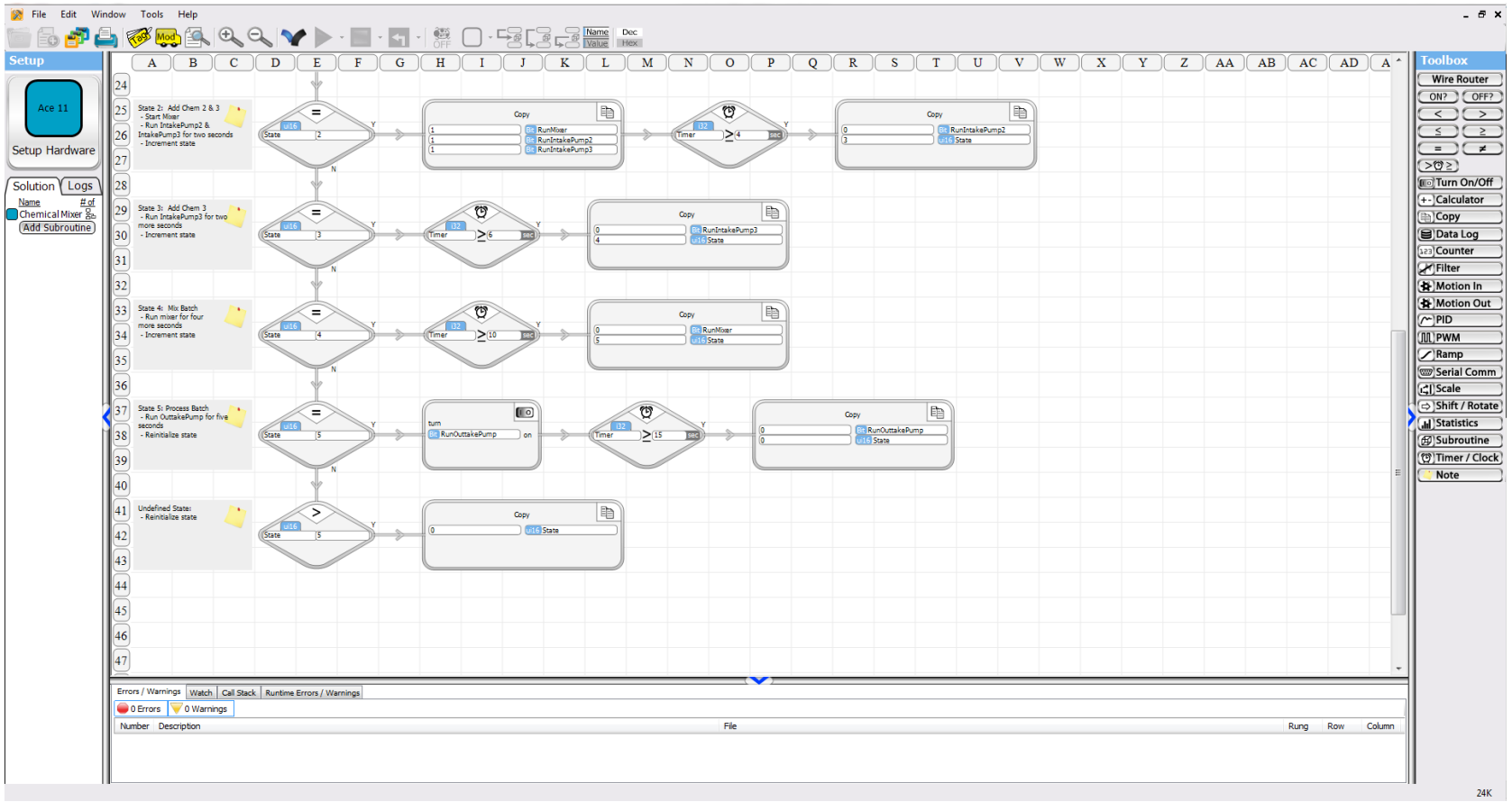
Basic PLC programming: Creating a first Flowchart-based program

Hands-on Creating a basic PLC program



Basic PLC programming: Creating a first Flowchart-based program

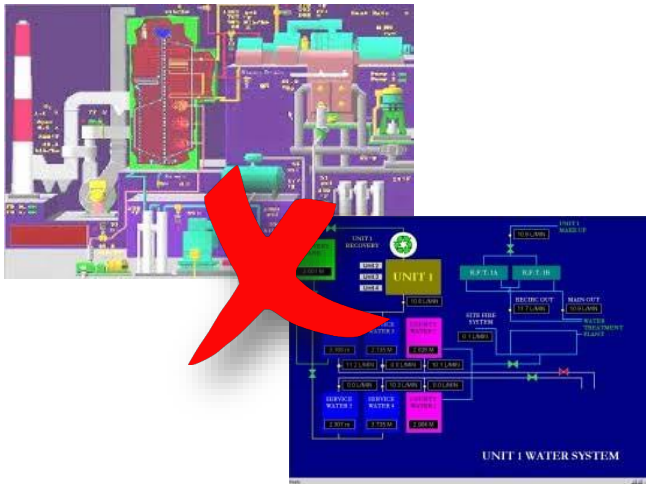
Hands-on Creating a basic PLC program



Hands on: Basic PLC Programming

- Hardware overview
- Creating a first Flowchart-based program
- **Creating visualisation**

Hands-on Creating a basic Visualisation

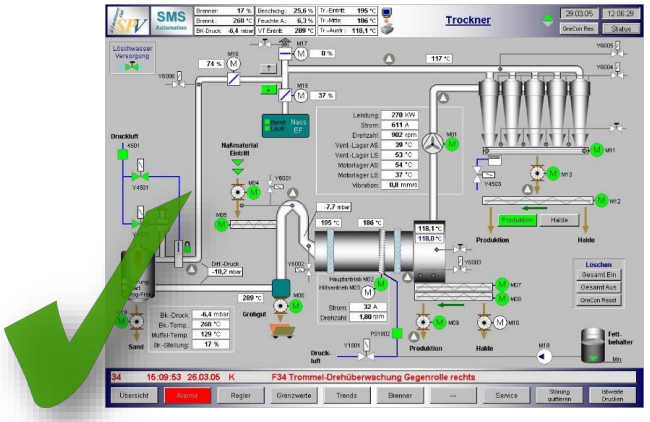


Requirements for a good HMI

- the screen must be able to hold operator's attention with maximum display clarity
- the design must allow a person with no training or little experience to be able to successfully operate a machine.

Fundamental factors critical to a display design

- Screen Layout
- Color Issues
  - Background Color
  - Display colors for objects
- Graphics and picture issues
- Display Tekst
- Presenting Data Values
- Alarms and Events
- Navigation and controls
- Operational Security



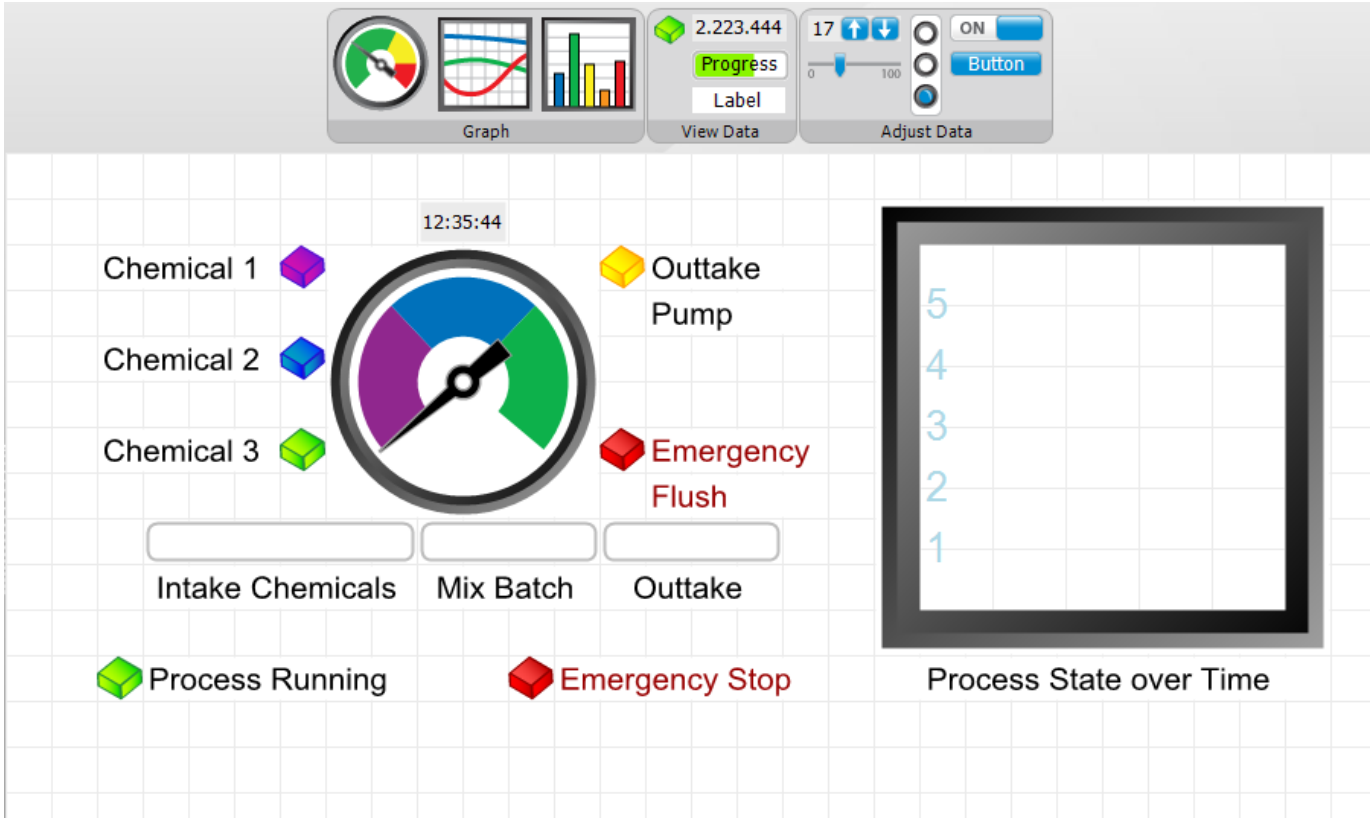
Basic PLC programming: Creating a first Flowchart-based program

Hands-on Creating a basic Visualisation

http://velocio.net/software/

Basic PLC programming: Creating a first Flowchart-based program

Hands-on Creating a basic Visualisation



https://youtu.be/F2Fe-licqqM


GONE FOR
LUNCH