



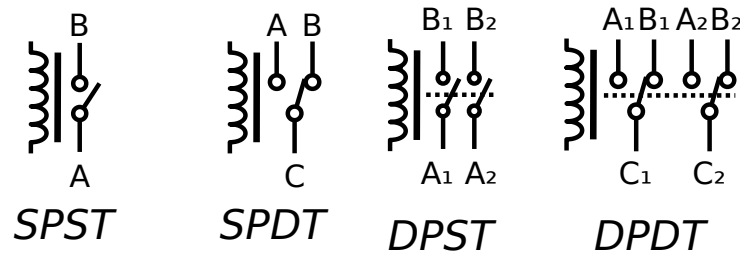
Programmable Logic Controllers

Automation
Course CO23-320203



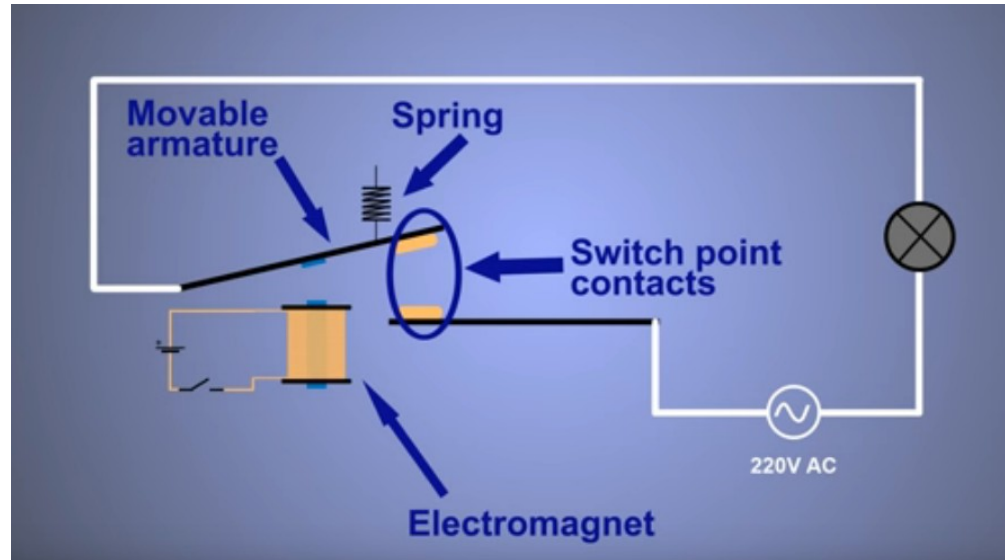
Relays

- Relays are electronic elements which use an electromagnet (typically a small AC or DC voltage) to close or open an electric contact (potentially with a higher current/voltage).



- Relays practically have not changed since the invention of telegraph (Henry, Davy, Morse, circa 1840). Why not replace them by transistors?
 - Transmitted current and operating voltage
 - Simple behaviour: on or off, no linear regions, no junction voltage, etc.
 - Convenient arrangements

Relays



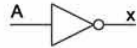






What is a Relay? How does a Relay works!
https://www.youtube.com/watch?v=1_YfuH_AcxQ

Relays – types and classification

- “*DPST NC*”?!? When resistor color code suddenly seems almost user friendly...
- Classification of relays
 - **N**ormally **O**n / **N**ormally **C**losed
 - **S**ingle **T**hrow / **D**ouble **T**hrow
 - **S**ingle **P**ole / **D**ouble **P**ole
 - + Bi-stable relays (coil is only activated when changing state)
 - + Stepping relays (each activation switches to another state)

Logic elements – logic gates

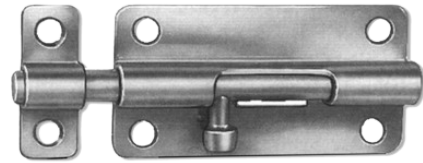
- Relays are extremely useful but how to deal with situations where there are multiple inputs which control one output?
- Logic gates!
 - NOT
 - AND
 - (NAND)
 - OR
 - (NOR)
 - XOR
 - (XNOR)

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	\overline{A}	AB	\overline{AB}	$A + B$	$\overline{A + B}$	$A \oplus B$	$\overline{A \oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

[Source: https://cdn-images-1.medium.com/max/1200/1*zq3cbyx-xd_SRq8EwzER0w.jpeg]

Logic elements – latches and flip-flops

- Basic element of logic circuit construction which is used for storing information (precisely: the high/low state, equivalent of a bit)
- What differentiates latches from flip-flops?
 - **Latches** – asynchronous operation. Can be polled/modified at any time
 - **Flip-flops** – clocked (= synchronous) operation. Require a falling or rising clock pulse to be read/written.

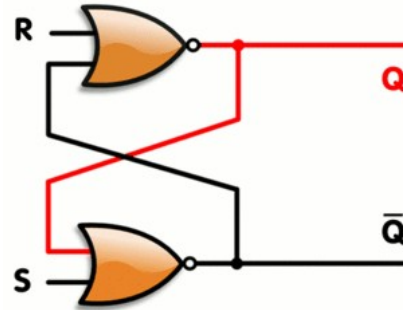


Types of flip-flops

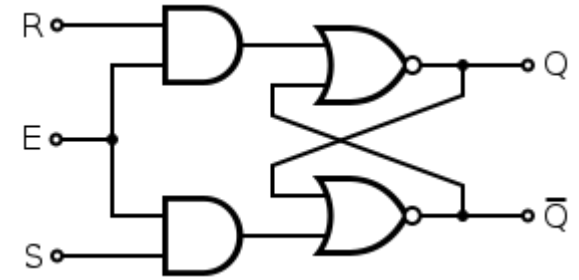
- SR ("set-reset")

- Truth table:

SR Flip-flop			
S	R	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined



SR Latch. [Source: https://upload.wikimedia.org/wikipedia/commons/c/c6/R-S_mk2.gif]



SR Flip-flop. [Source: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e1/SR_%28Clocked%29_Flip-flop_Diagram.svg/300px-SR_%28Clocked%29_Flip-flop_Diagram.svg.png]

Types of flip-flops

- D ("data" or "delay")

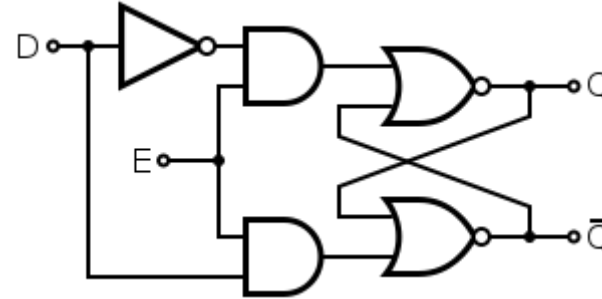
D Flip-flop		
D	Q(t+1)	Operation
0	0	Reset
1	1	Set

- T ("toggle")

T Flip-flop		
T	Q(t+1)	Operation
0	Q(t)	No change
1	$\bar{Q}(t)$	Complement

- JK - like SR but changes state upon the restricted combination $J \equiv S=1$, $K \equiv R=1$

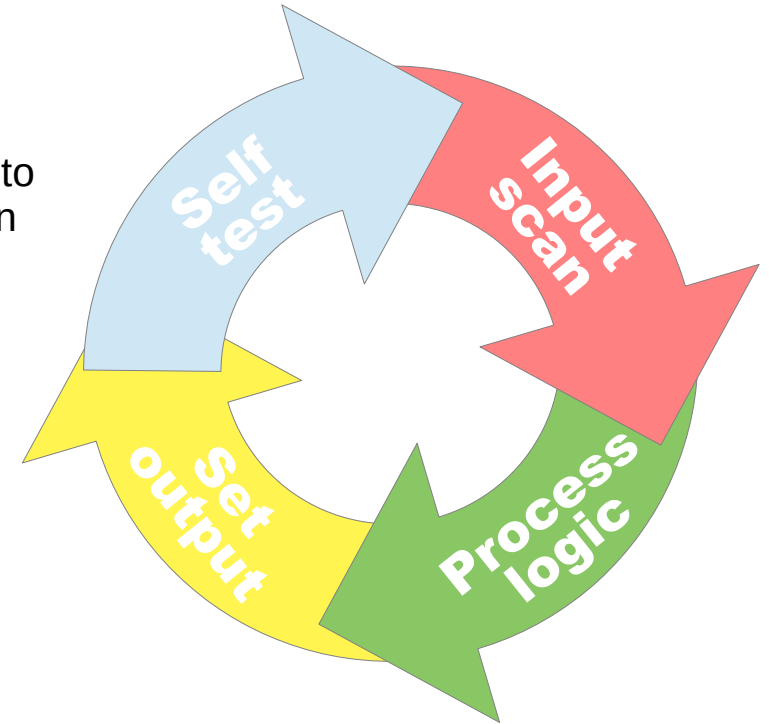
JK Flip-flop			
J	K	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}(t)$	Complement



D flip-flop. [Source: [https://en.wikipedia.org/wiki/Flip-flop_\(electronics\)#/media/File:D-type_Transparent_Latch_\(NOR\).svg](https://en.wikipedia.org/wiki/Flip-flop_(electronics)#/media/File:D-type_Transparent_Latch_(NOR).svg)]

Programmable Logic Controller

- Programmable Logic Controller (PLC) is a CPU-based controller with a simple and efficient real-time processing architecture (often named “1-bit”).
- It processes input from various sensors or buttons is mapped to memory locations in the input block. Setting output variables in the output memory block cause signals to be transmitted to actuators.
- The CPU constantly runs a fixed-interval (“scan time”) loop (“scan cycle”)
 - Self test detect any errors of the PLC hardware
 - CPU reads memory area where the inputs values are
 - It sequentially processes all instructions of the currently loaded program
 - It updates output variables in the output memory area



Brief history of PLCs

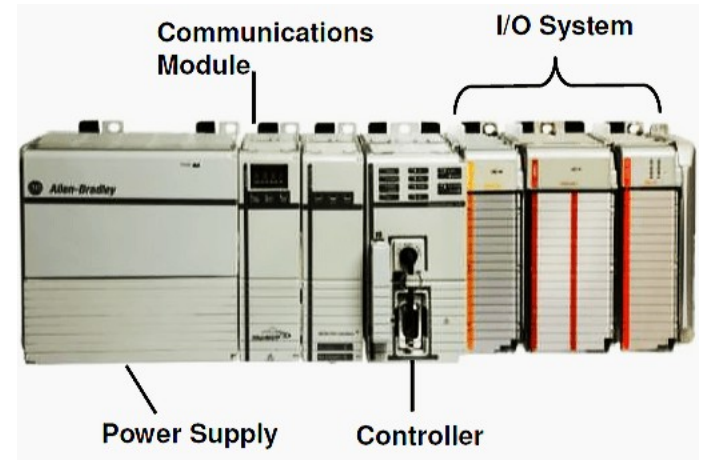
- With rapid industrialisation, many new mechanisms appeared and engineers tried to make existing one function in an automatic fashion.
 - Result: complex mechanic and electric systems interconnecting relays, logic circuits (cathode ray tubes, logic gates, etc) with cams, indexers, etc.
 - Difficult to maintain but even more difficult to upgrade, e.g. in case of design changes
- In 1968 GM Hydra-Matic (the automatic transmission division of General Motors) issued a request for proposals for an electronic replacement for hard-wired relay systems.
- Bedford Associates (today part of Schneider Electric) came up with MODular DIGital CONTroller (MODICON), with its 084 controller and won the bid.
- MODICON 984 was introduced in the 90'. One of the original 084 PLCs was presented to the producer by the GM plant after nearly 20 years in operation.

Today: why PLC?

- While their appearance was an obvious advance, why still use them today?
 - General purpose computers can do much more advanced and flexible calculations
 - Specialised computer (GPGPU, FPGA) can be extremely efficient
 - General-purpose controllers (e.g. Arduino) are very cheap and can be easily programmed
- Answer: because of the specific needs of the industry
 - Guarantee of real time performance
 - Can be programmed by a non-specialist
 - Is rugged enough to survive factory floor conditions
 - Consistent interface over lifetime and consecutive generations

Components of PLC

- Power Supply
- CPU (a.k.a. Controller)
- I/O modules
 - Digital
 - Analog
- Communication modules
- Integration rack/rail/housing (e.g. DIN rail)
- Specialised modules (e.g. motor controller)
- (External programming tool / PC)

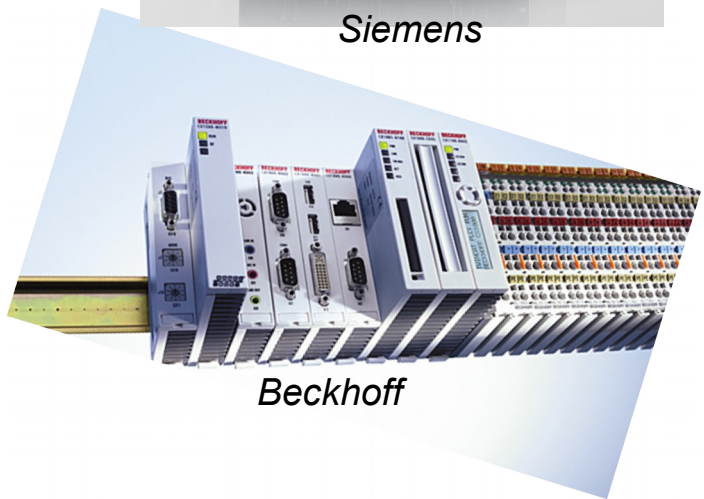


[Source: <http://electrical-engineering-portal.com/wp-content/uploads/2016/09/plc-components.png>]

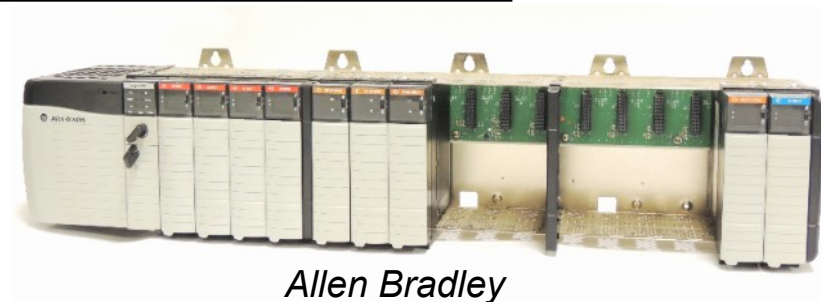
PLC examples and integration



Siemens



Beckhoff



Allen Bradley



Mitsubishi

Something missing in these images? → Tons of cables!

PLC examples and integration

Control cabinet – typical integration

Spring 2019

Automation - PLCs 1

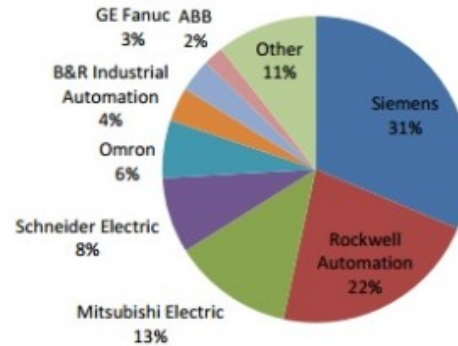


PLC sizes

- Design parameters – number of I/Os and program memory
 - Small – up to 128 Inputs/outputs and 2Kb of memory
 - Medium – up to 2048 I/Os and up to 32Kb of memory
 - Large – 8192 I/Os and 750Kb of memory
- The PLC size is chosen w.r.t. the machinery to control and the position in the hierarchy of control (will be discussed later)

Major PLC manufacturers

- Siemens (Simatic)
- ABB
- Schneider (Modicon)
- Rockwell (Allen-Bradley)
- Mitsubishi
- GE Automation
- Beckhoff
- ...



Warning: almost definitely false data!

- Each producer has their own hardware design, conventions, programming tools and setup procedures
- How can one possibly work with more than one type?

Common standards - IEC 61131

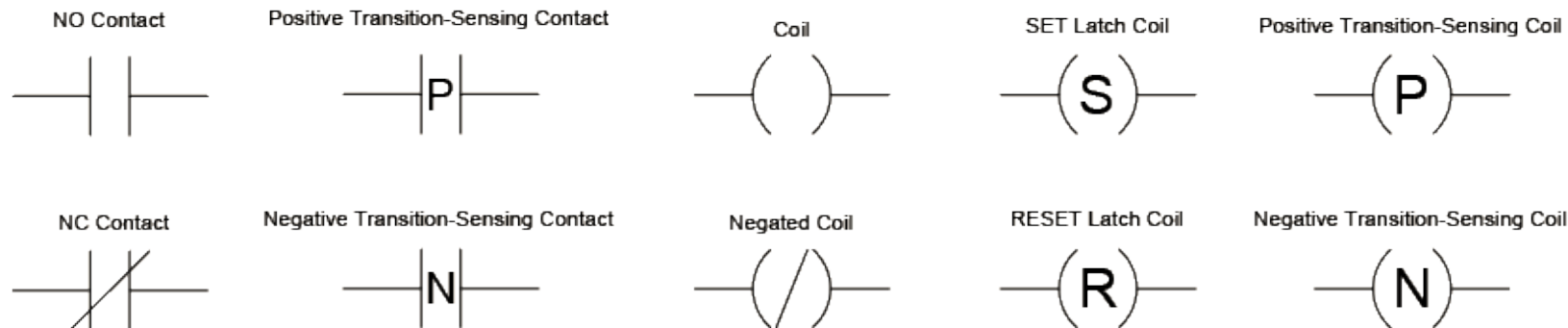
- Producers follow common specifications: IEC 61131
- Part 3: programming languages
 - **Ladder diagram** (LD), graphical
 - Circuit analogy encoding of each function of the controller
 - **Structured text** (ST), textual
 - Pascal-like programming with specific constraints
 - **Function block diagram** (FBD), graphical
 - Logical blocks
 - **Instruction list** (IL), textual (deprecated)
 - Assembly-like low level language
 - **Sequential function chart** (SFC), for sequential and parallel control processing programmes
- Another good news: all PLC languages are interoperable
- Other common standards:
 - Part 5: Communication
 - Part 7: fuzzy logic programming

PLC programming – overview

- Some elements that can be used in the PLC programs:
 - **Normally open/Normally closed contacts** (input variables = tags = symbols)
 - **Coils** (output; can be **normal** or **latched**; highlighted means they are energized).
 - **Timers** (coil can either be ON or OFF for the specified delay).
 - **Counters** (can count by increments either up or down).
 - **Bit shift registers** (can shift data by one bit when active).
 - **One-shot** (meaning active for one scan time; useful for pulse timer).
 - **Drums** (can be sequenced based on a time or event).
 - **Data manipulation instructions** (enable movement, comparison of digital values).
 - **Arithmetic instructions** (enable addition, subtraction, multiplication, and division of digital values).

Ladder diagrams

- Input/output tables: textual symbols for memory image locations
- Standardised set of symbols expressing the electrical engineering metaphores



- A “ladder” of lines (= “rungs”), each expressing a single function / interconnection

STEP 7-Micro/WIN 32 - Hidrofor2-3 - [SIMATIC LAD]

File Edit View PLC Debug Tools Windows Help

View

Program Block

Symbol Table

Status Chart

Data Block

System Block

Cross Reference

Communications

Instructions

Bit Logic

Clock

Communications

Compare

Convert

Counters

Floating-Point Math

Integer Math

Interrupt

Logical Operations

Move

Program Control

Shift/Rotate

String

Table

Timers

Libraries

Call Subroutines

Tools

Hidrofor2-3(CPU 226 REL 02.00)

Program Block

MAIN (OB1)

SBR_0 (SBR0)

INT_0 (INT0)

Symbol Table

USR1 (USR1)

POU Symbols (SYS1)

Status Chart

Data Block

System Block

Cross Reference

Communications

Instructions

Bit Logic

Clock

Communications

Compare

Convert

Counters

Floating-Point Math

Integer Math

Interrupt

Logical Operations

Move

Program Control

Shift/Rotate

String

Table

Timers

Libraries

Call Subroutines

Network 6

Szivattyú hibajelzés időzítése

S4000:I0.0

Q11:Q1.1

I17:I1.7

T37

IN

TON

+1200

PT

B4015:I1.5

T44

IN

TON

+50

PT

I17:I1.7

T38

IN

TON

+1200

PT

Symbol	Address	Comment
B4015	I1.5	Áramlás érzékelő a szivattyú szívó ágán (áramlás van)
I17	I1.7	Szivattyú az előírt sebességgel forog
Q11	Q1.1	Szivattyú start (frekvenciaváltó)
S4000	I0.0	Automata kapcsoló

Network 7

Szivattyú hibajelzés

I16:I1.6

/

M00:M0.0

S1

OUT

SR

SM0.5

H03:Q0.3

()

T37

T38

T44

S4003:I0.3

R

MAIN SBR_0 INT_0

Total Errors: 0

Ready

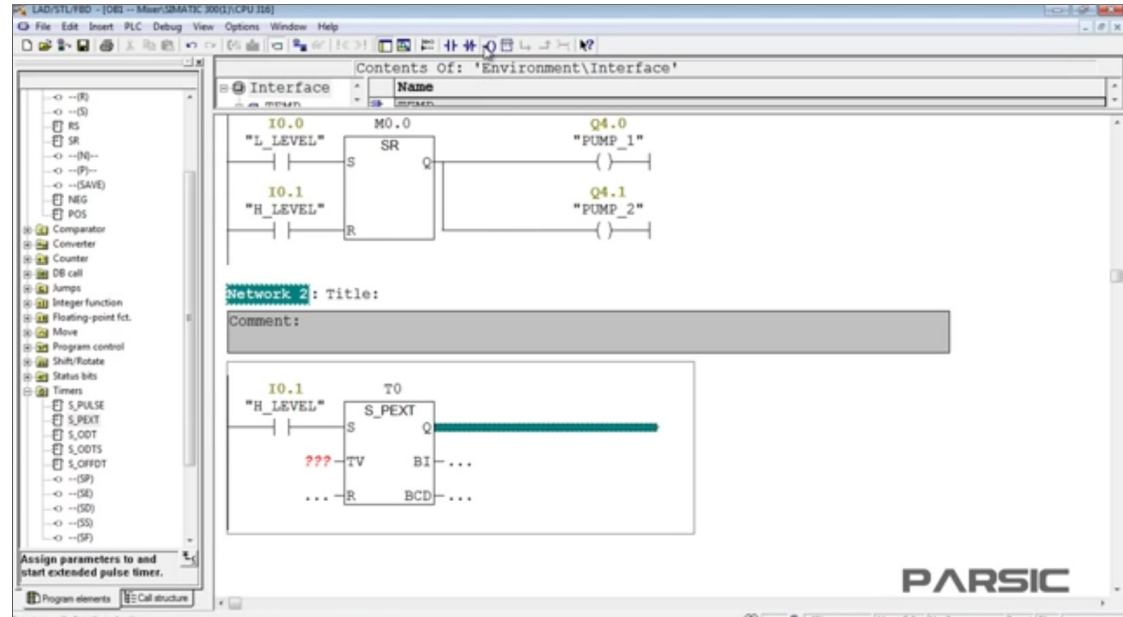
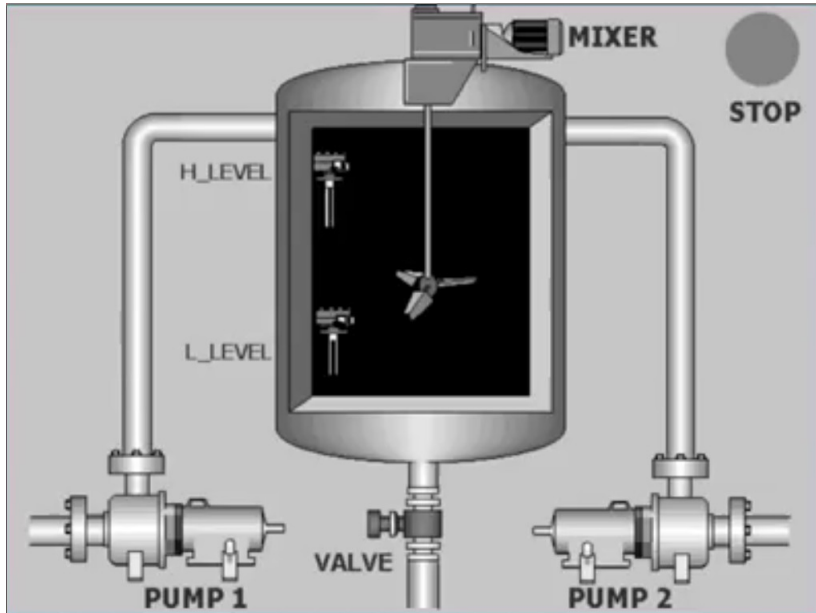
Network 6 Title INS

Ladder logic – programming workflow

- Configuring the programming environment
 - Define the PLC model and its physical configuration, i.e. which slots are occupied by particular components
- Setting up the user-friendly tags (or symbols or variable names) for the input and output memory locations
 - For example: *Q1.3* (fourth bit of the second byte in the output memory zone) → *OUTPUT_VALVE_OPEN*
- Creating rungs for each output
- Testing using provided simulation tool, usually by manually altering the values stored in the input/output memory

Ladder logic example

- Liquid mixing tank



- Tutorial Part 1: <https://www.youtube.com/watch?v=y2eWdLk0-Ho>
- Tutorial Part 2: <https://www.youtube.com/watch?v=nYr8Q21nG0k>

Structured text example

Example: a simple function to find the index of a first array element greater than a given threshold

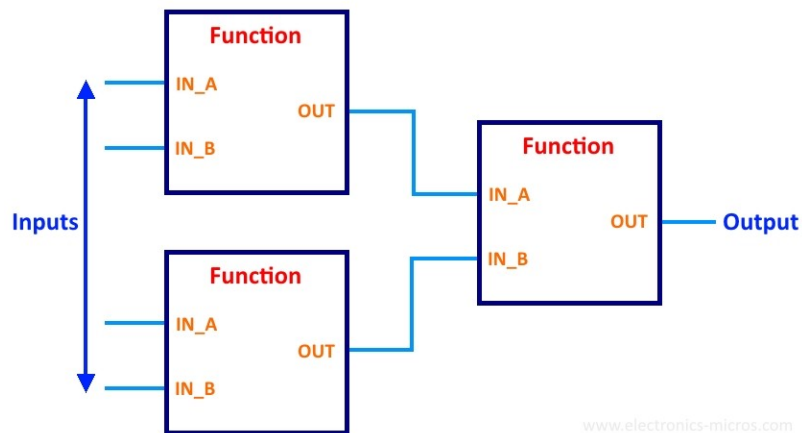
```
FUNCTION FindGreaterThen : INT
VAR_INPUT
    Threshold: REAL;
END_VAR
VAR_IN_OUT
    Values: ARRAY[1..1000] OF REAL;
END_VAR
VAR
    Found: BOOL;
    Index: INT;
END_VAR
Found := FALSE;
Index := 1;
```

```
WHILE NOT (Found) AND Index <= 1000 DO
    IF Values[Index] > Threshold THEN
        Found := TRUE;
    ELSE
        Index := Index + 1;
    END_IF;
END_WHILE;
IF Found THEN
    FindGreaterThen := Index;
ELSE
    FindGreaterThen := 0;
END_IF;
```

- The main loop is managed by the PLC – the program covers just one cycle
- The PLC also furnishes the input and output memory variables

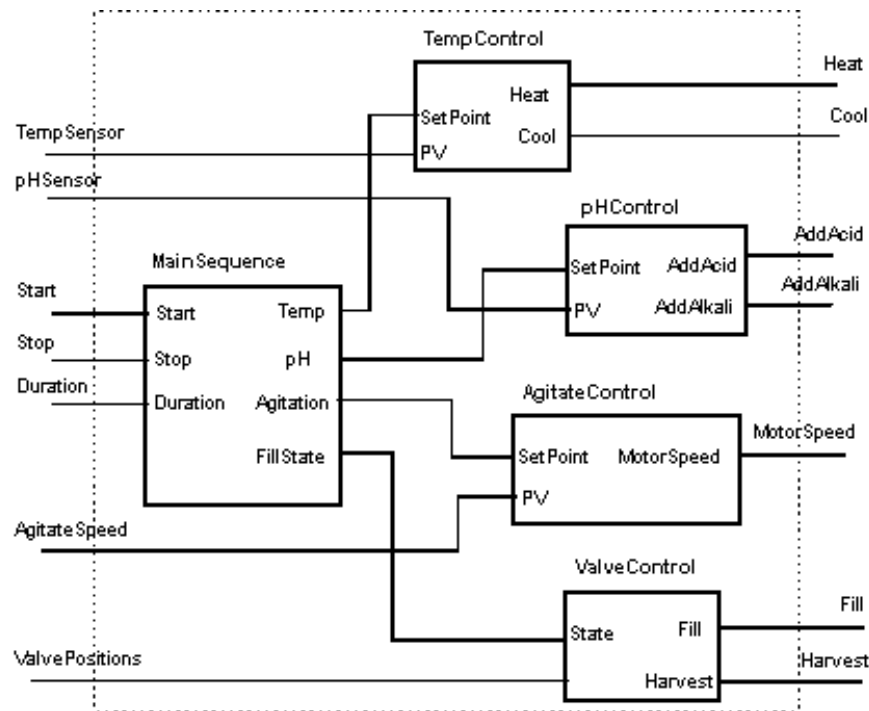
Function block diagram example

- General idea:



[Source: <http://wiringdiagramcircuit.co/wp-content/uploads/2018/02/unique-function-block-diagram-tutorial-functional-block-diagram-of-plc-yhgfdmuor-net.jpg>]

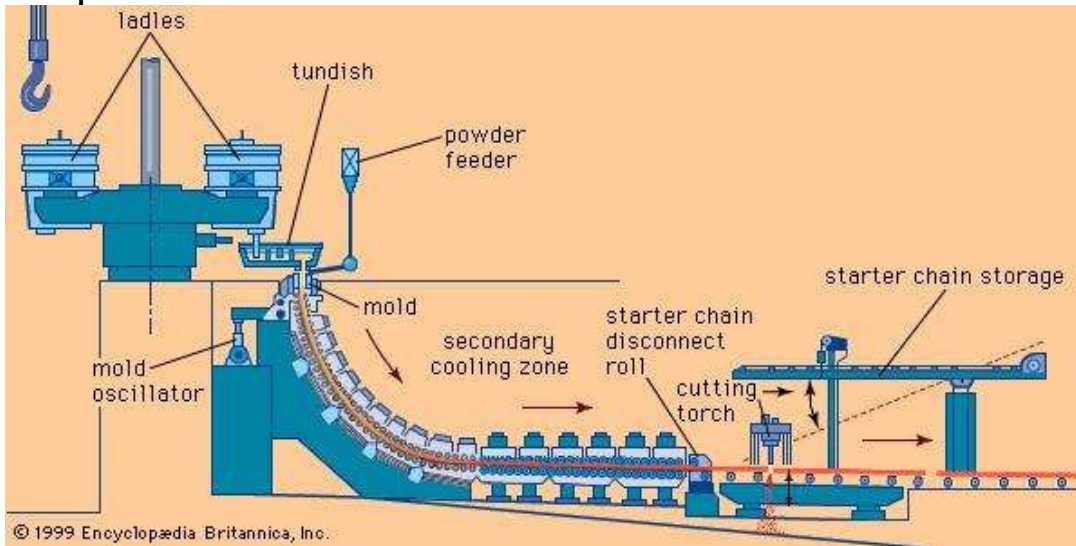
- Mathematical, logical operations or simple programs are present in the blocks.
- Blocks can be written in Structured Text!



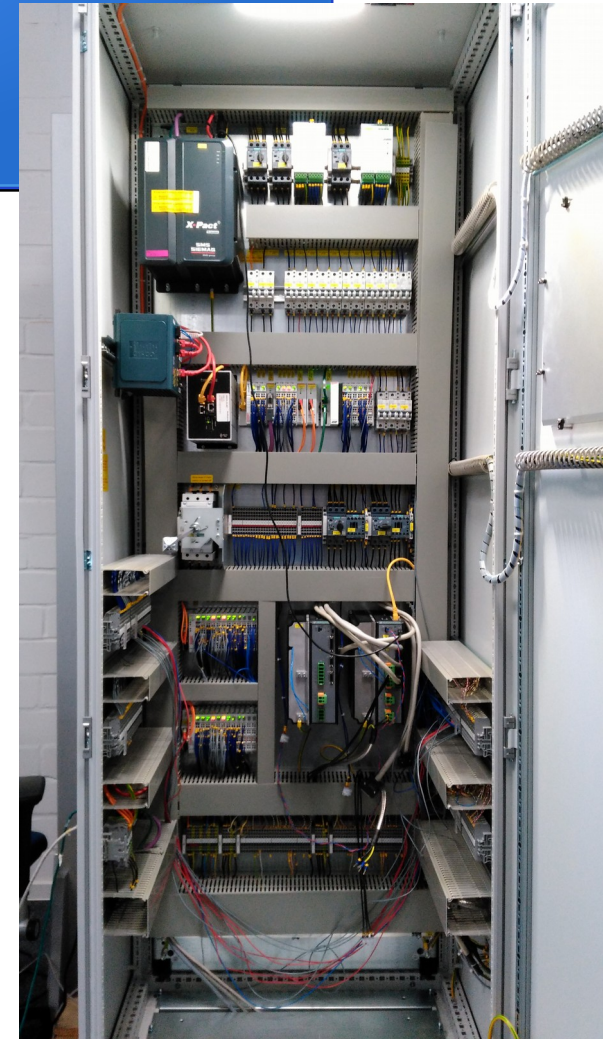
[Source: <http://www.plcopen.org/pages/images/benefits/struct4.gif>]

PLCs in operation

- In the picture: the electrical control cabinet of a hydraulically actuated plunger governing the flow rate of molten steel from the tundish in the slab casting process.



[Source: <https://amedia.britannica.com/700x450/48/1548-004-13E12F99.jpg>]



Want to know more?

- Good sources of further information:
- <http://automationprimer.com/>
- <http://www.plcacademy.com>