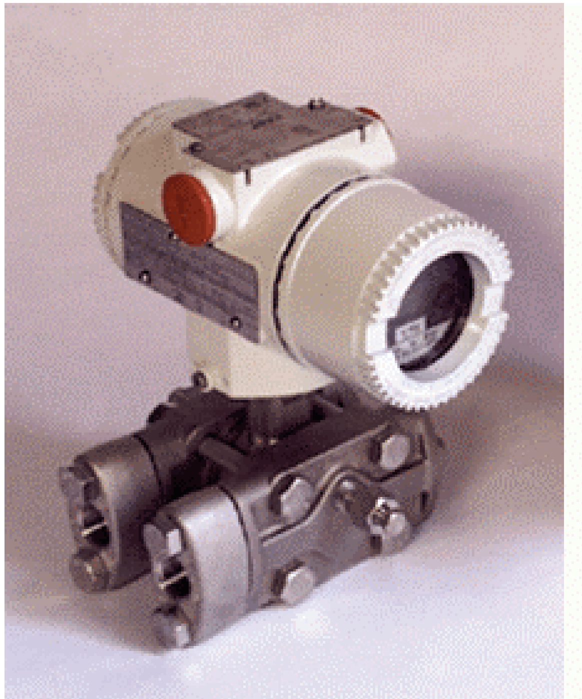
Industrial Automation
Automation Industrielle
Industrielle Automation



2. Instrumentation and Control

2.1 Instrumentation - Sensors and actors

Instrumentation - Capteurs et actionneurs

Instrumentierung - Sensoren und Aktoren

Prof. Dr. H. Kirrmann

ABB Research Center, Baden, Switzerland

2.1.1 Market

The instrumentation market

Emerson (Fisher-Rosemount): 27 %
Invensys: 4-5%
ABB: 4-5%
Honeywell: 3-4%

one dominant player a lot of small players…

Concepts

instruments = sensors (***capteurs,*** *Messgeber*) <u>and</u> actors (***actionneurs,*** *Stellglieder*)

<u>binary</u> (on/off) and <u>analog</u> (continuous) instruments are distinguished.

industrial conditions:

- temperature range commercial: (0°C to +70°C)
  industry (-40°C..+85°C)
  extended industrial(–40°C..+125°C)

- mechanical resilience (shocks and vibrations) EN 60068
- protected against Electro-Magnetic (EM)-disturbances EN 55022, EN55024)
- sometimes NEMP-protected (Nuclear EM Pulse) - water distribution, civil protection
- protection against water and moisture (IP67=completely sealed, IP20 = normal)
- easy mounting and replacement
- robust connectors
- DC-powered (24V= because of battery back-up, sometimes 48V=)

2.1.2 Binary Instruments

Binary position measurement

binary sensors (*Geber*, "*Initiator*", ***indicateur "tout ou rien"***):

•micro-switch (*Endschalter*, ***contact fin de course***) +cheap, -wear, bouncing

•optical sensor (*Lichtschranke*, ***barrière optique***) +reliable, -dust or liquid sensitive

•magnetic sensor (*Näherungsschalter*, ***détecteur de proximité***) +dust-insensitive, - magnetic

Binary Signal processing

Physical attachment

- Level adaptation,
- Galvanical separation
- EMC barrier (against sparks, radio, disturbances)

Acquisition

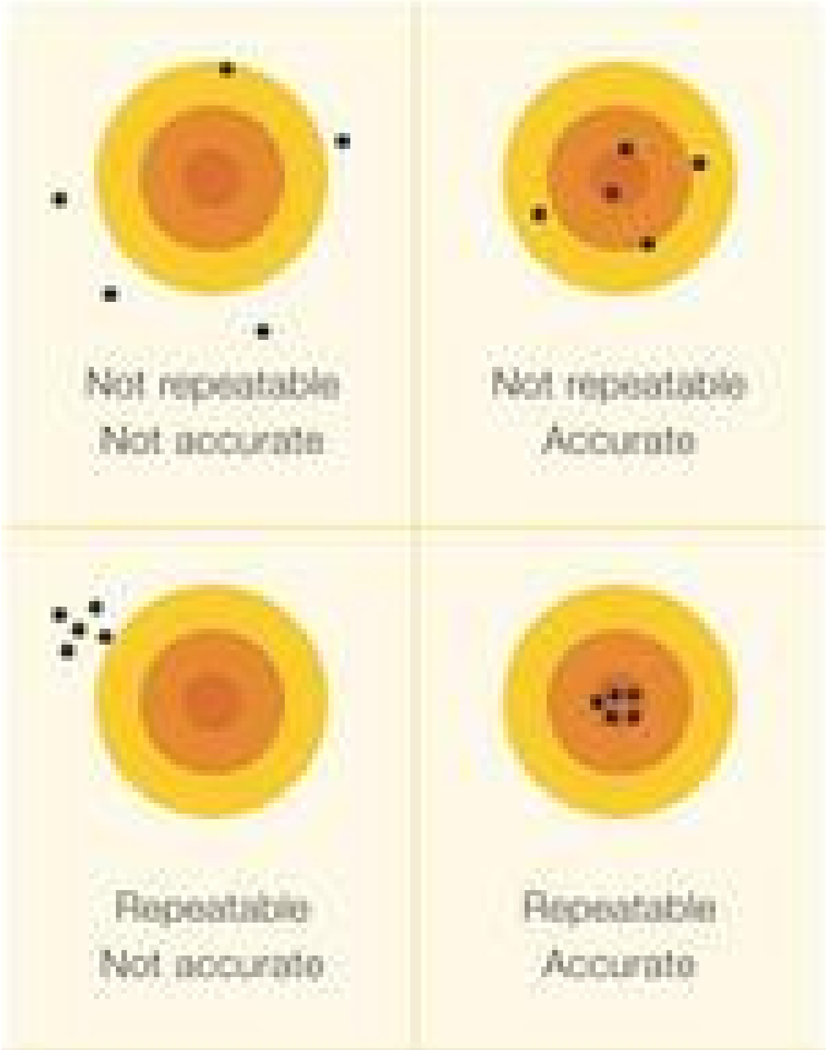
- Convert to standard levels
- Relay contacts 24V (most frequent), 48V, 110V (electrical substations)
- Electronic signals 24V —>10V-60V,
- Output: 0..24V@100mA
- Counter inputs: Gray, BCD or binary

Processing

- Filtering (e.g. 0..8 ms filter),
- Plausibility (*Antivalenz*, ***Antivalence***),
- Bounce-free (*Entprellen, Anti-rebond*)

2.1.3 Analog Instruments

Precision: repeatability and accuracy

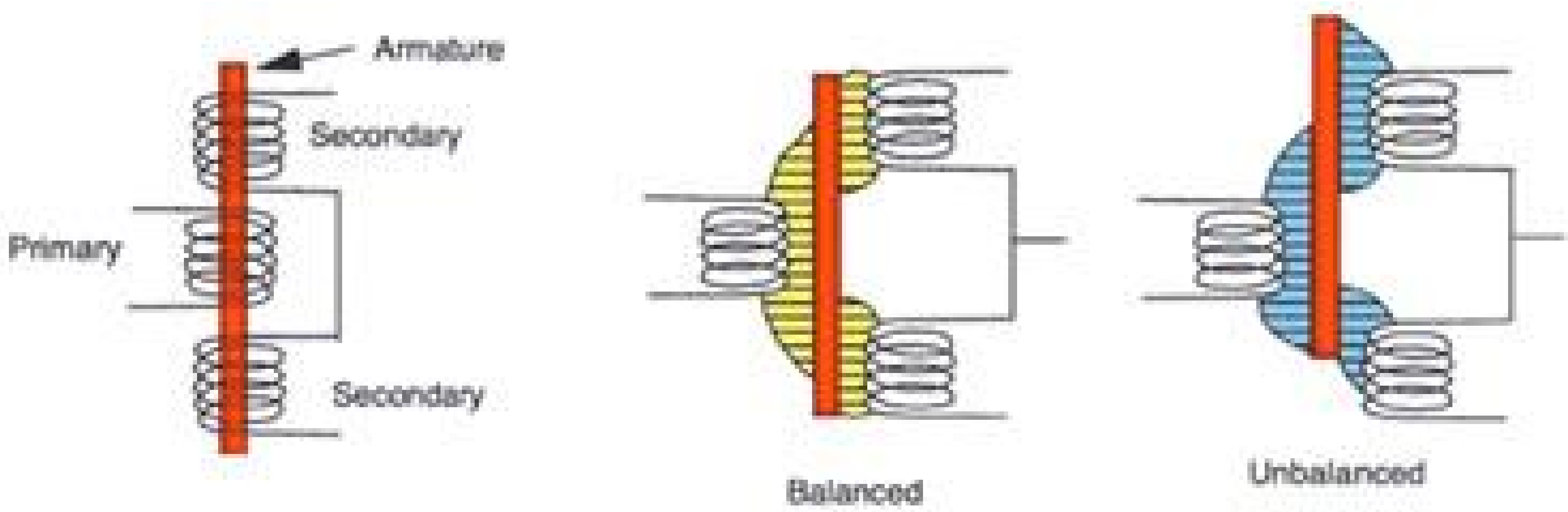


2.1.3.1 Analog mechanical position

potentiometer	+cheap, -wear, bad resolution
capacitive	+cheap, -bad resolution
balanced transformer (LVDT) (linear or sin/cos encoder)	+reliable, robust - small displacements
strain gauges	+reliable, very small displacements
piezo-electric	+extremely small displacements

Variable differential transformer (LVTD)

The LVDT is a variable-reluctance device, where a primary center coil establishes a magnetic flux that is coupled through a mobile armature to a symmetrically-wound secondary coil on either side of the primary.
Two components comprise the LVDT: the mobile armature and the outer transformer windings. The secondary coils are series-opposed; wound in series but in opposite directions.

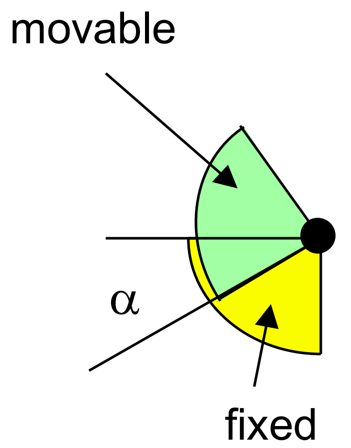


When the moving armature is centered between the two series-opposed secondaries, equal magnetic flux couples into both secondaries; the voltage induced in one half of the secondary winding is 180 degrees out-of-phase with the voltage induced in the other half of the secondary winding.
When the armature is moved out of that position, a voltage proportional to the displacement appears

source: www.sensorland.com

Capacitive angle or position measurement

$$C = \frac{A}{d} \sim \alpha$$

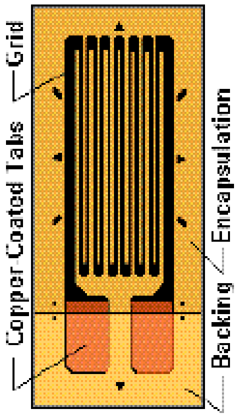


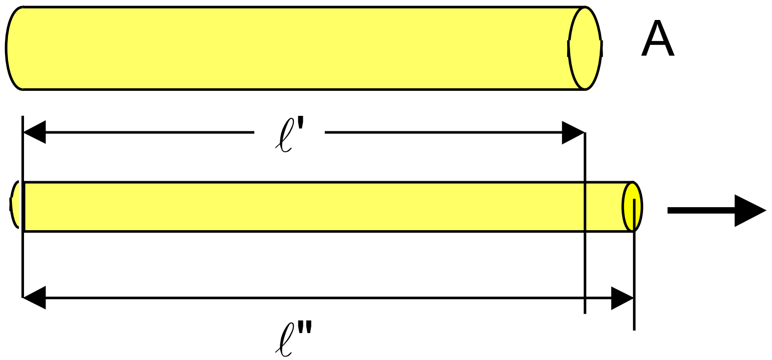
capacitance is evaluated modifying the frequency of an oscillator

Small position measurement: strain gauges

Dehnungsmessstreifen (DMS), *jauges de contrainte*

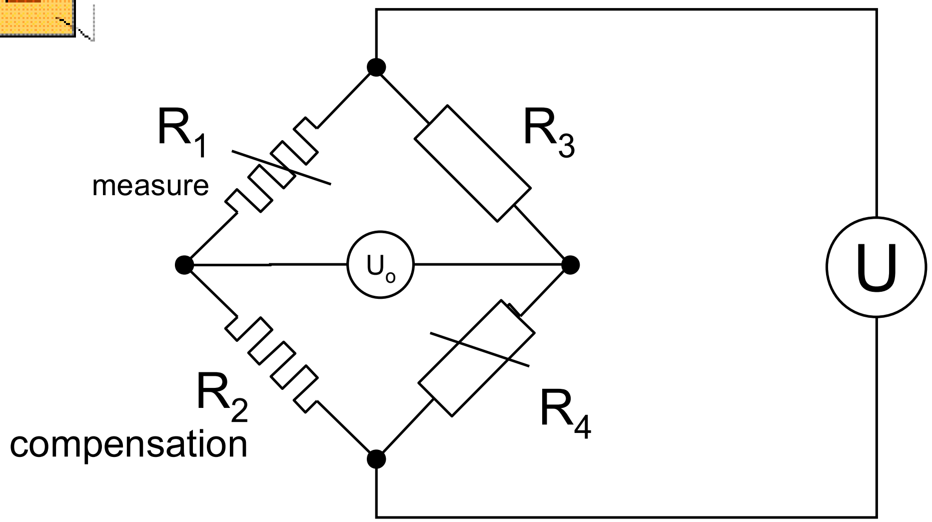
Principle: the resistance of a wire increases when this wire is stretched:





$$R = \rho \frac{\ell}{A} = \rho \frac{\ell^2}{V} \sim \ell^2$$

volume = constant, ρ = constant



measurement in bridge
(if $U_0 = 0$: $R_1 R_4 = R_2 R_3$)

temperature compensation by “dummy” gauges

frequently used in buildings, bridges, dams for detecting movements.

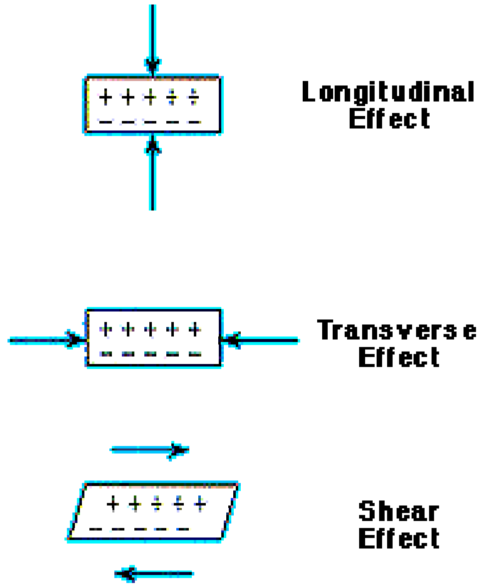
Piezo-electrical effect

Piezoelectric materials (crystals) change form when an electrical field is applied to them. Conversely, piezoelectric materials produce an electrical field when deformed.

Quartz transducers exhibit remarkable properties that justify their large scale use in research, development, production and testing.
They are extremely stable, rugged and compact.

Of the large number of piezoelectric materials available today, quartz is employed preferentially in transducer designs because of the following excellent properties:

- high material stress limit, around 100 MPa (~ 14 km water depth)
- temperature resistance (up to 500C)
- very high rigidity, high linearity and negligible hysteresis
- almost constant sensitivity over a wide temperature range
- ultra high insulation resistance (10^{+14} ohms) allowing low frequency measurements (<1 Hz)



source: Kistler

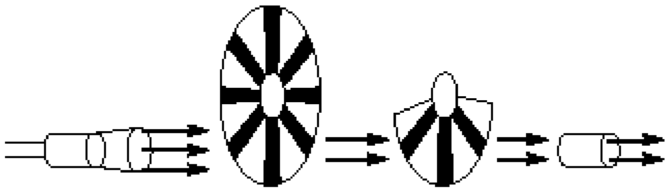
Force measurement

Force / Torque / Weight / Pressure is measured by small displacements ($F = k \cdot x$):

- piezo-electrical transducers
- strain gauges

Acceleration is measured by way of force / displacement measurement ($F = M \cdot \gamma$)

Principle of optical encoding



Optical encoders operate by means of a grating that moves between a light source and a detector. The detector registers when light passes through the transparent areas of the grating.

For increased resolution, the light source is collimated and a mask is placed between the grating and the detector. The grating and the mask produce a shuttering effect, so that only when their transparent sections are in alignment is light allowed to pass to the detector.

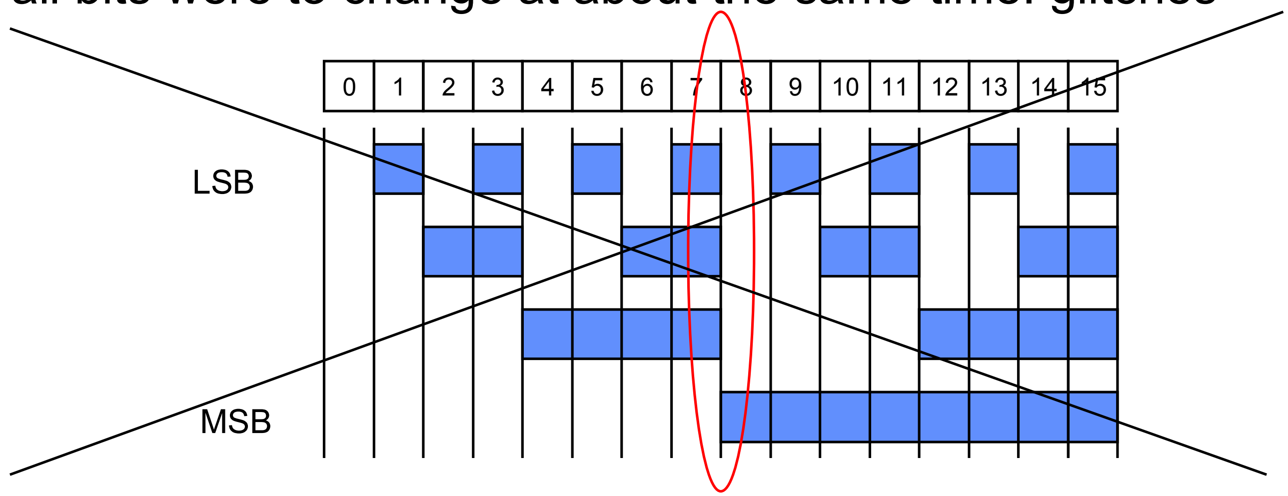
An incremental encoder generates a pulse for a given increment of shaft rotation (rotary encoder), or a pulse for a given linear distance travelled (linear encoder). Total distance travelled or shaft angular rotation is determined by counting the encoder output pulses.

An absolute encoder has a number of output channels, such that every shaft position may be described by its own unique code. The higher the resolution the more output channels are required.

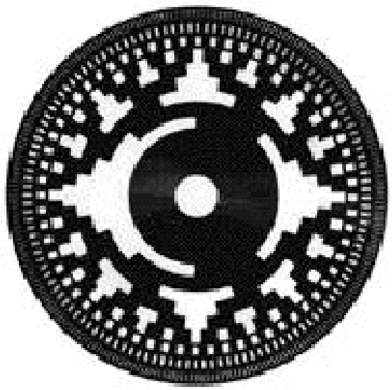
courtesy Parker Motion & Control

Absolute digital position: Grey encoder

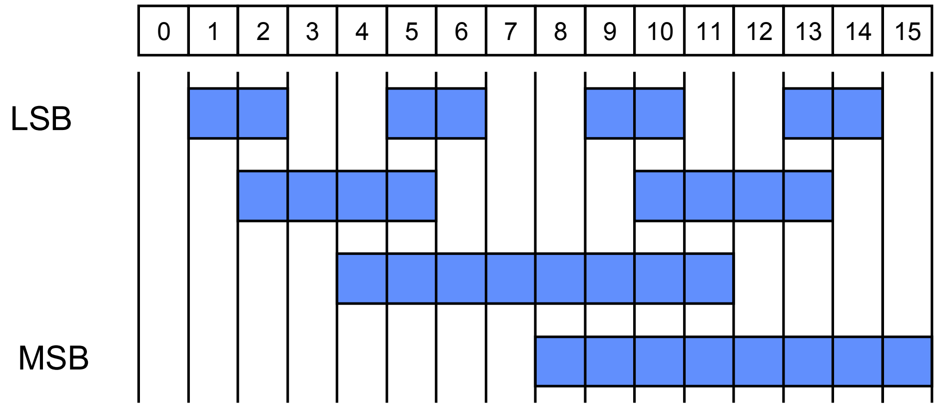
straight binary: if all bits were to change at about the same time: glitches



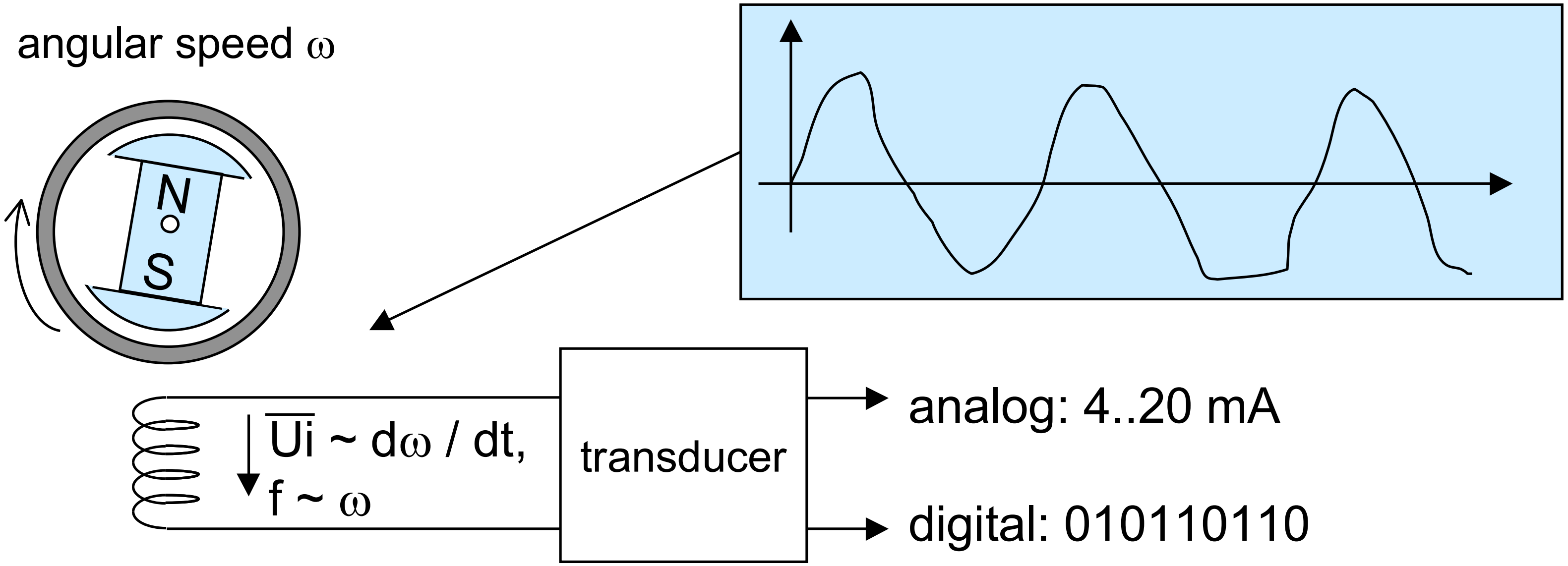
Grey: only one bit changes at a time: no glitch



Grey disk (8 bit)



Analog speed measurement: tachometer



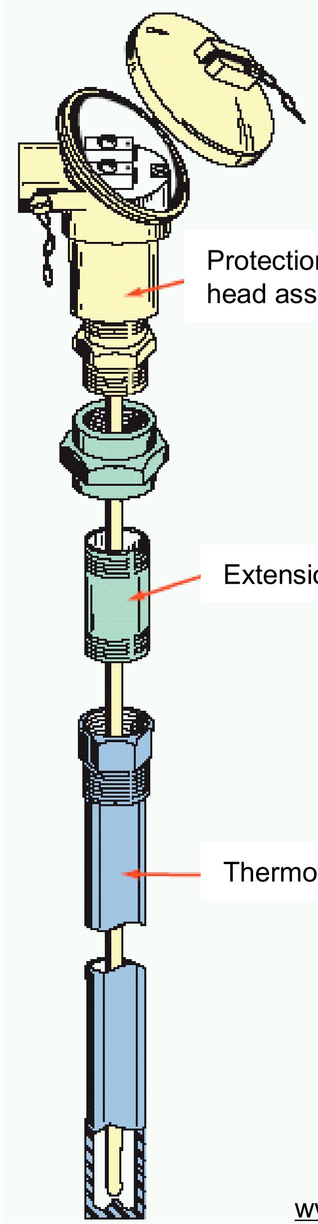
a simple tachometer is a rotating permanent magnet that induces a voltage into a stator winding.

this voltage is converted into an analog voltage or current, which can later be converted to a digital value,

alternatively, the frequency can be measured to yield directly a digital value

2.1.3.2 Temperature measurement

the most frequently measured value in industry



Temperature measurement

Thermistance (RTD - resistance temperature detector):

metal whose resistance depends on temperature:
+ cheap, robust, high temperature range (-180ºC ..600ºC),
- require current source, needs linearisation.

Thermistor (NTC - negative temperature coefficient):

semiconductor whose resistance depends on temperature:
+ very cheap, sensible,
- low temperature, imprecise, requires current source, strongly non-linear

Thermo-element (*Thermoelement, **thermocouple***):

pair of dissimilar metals that generate a voltage proportional to the temperature difference between warm and cold junction (Seebeck effect)
+ high precision, high temperature, punctual measurement
- low voltage, requires cold junction compensation, high amplification, linearization

Spectrometer:

measures infrared radiation by photo-sensitive semiconductors
+ highest temperature, measures surfaces, no contact
- highest price

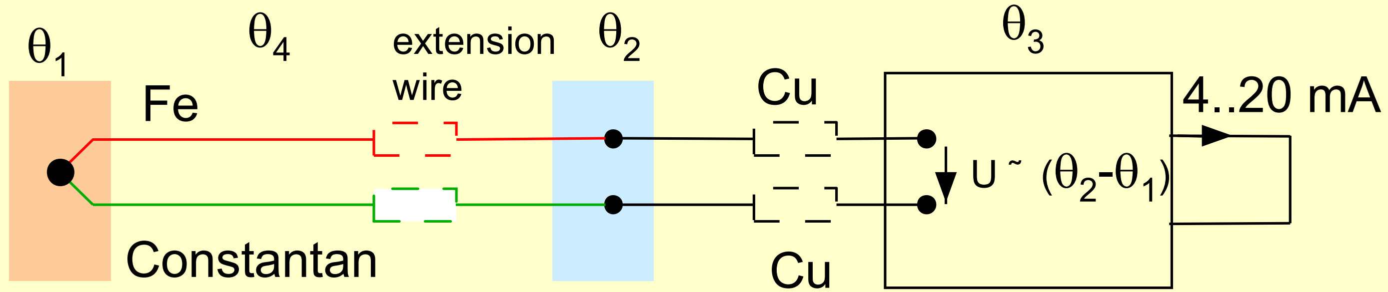
Bimetal (*Bimetall, **bilame***):

mechanical (yes/no) temperature indicator using the difference in the dilatation coefficients of two metals, very cheap, widely used (toasters...)

Thermo-element and Thermo-resistance

Thermo-element
(*Thermocouple*)

two dissimilar electrical conductors



Fe-Const

also: Pt/Rh - Pt

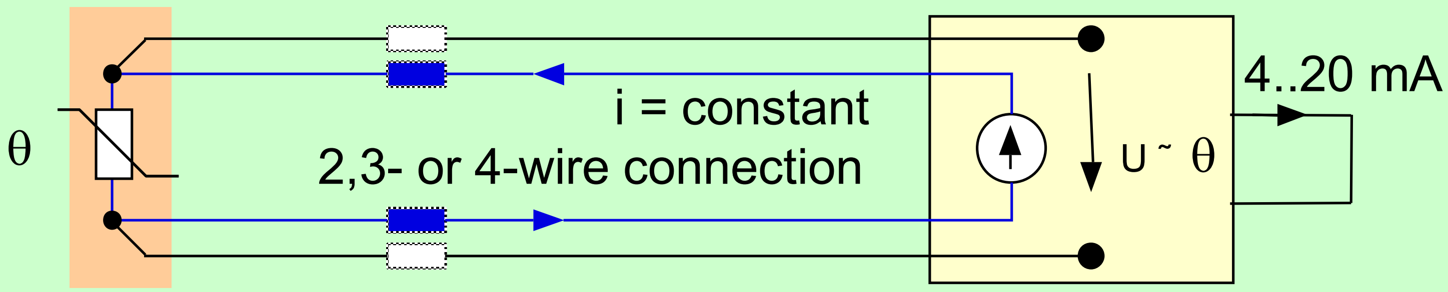
measured temperature
(hot junction)

reference temperature
(cold junction)

Thermoresistance
(semiconductor or metal)

Platinum (Pt 100)

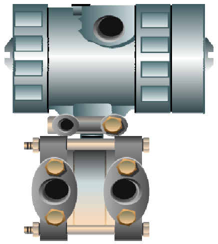
one material whose resistance is temperature-dependent

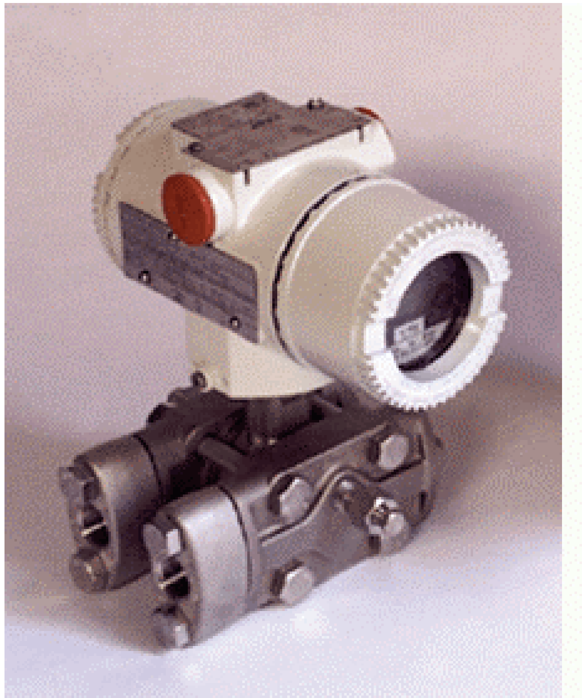


2 or 4 wire connection (to compensate voltage drop)

2.1.3.3 Hydraulic measurements

- Flow,
- Mass Flow,
- Level,
- Pressure,
- Conductivity,
- pH-Sensor,
- Viscosity,
- Humidity,

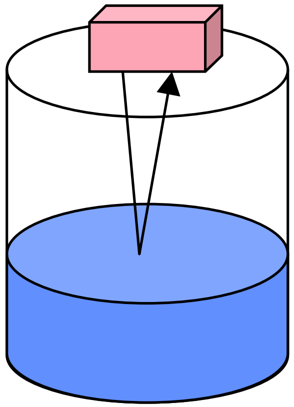


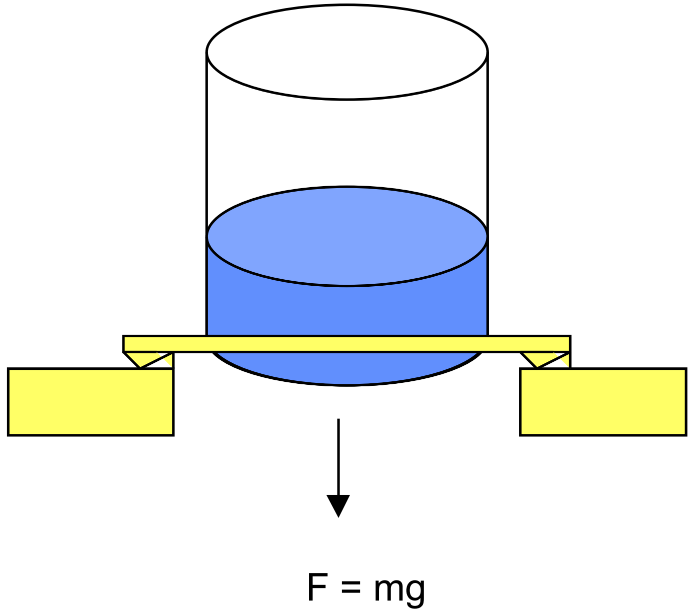


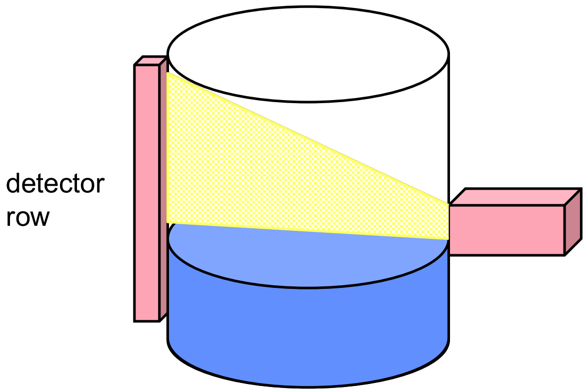
special requirements: intrinsic safety = explosive environment, sea floor = high pressure

Level measurement

- pulsed laser
- load cell
- pulsed microwave
- nuclear
- ultrasonic (40-60 kHz)
- low power ultrasonic

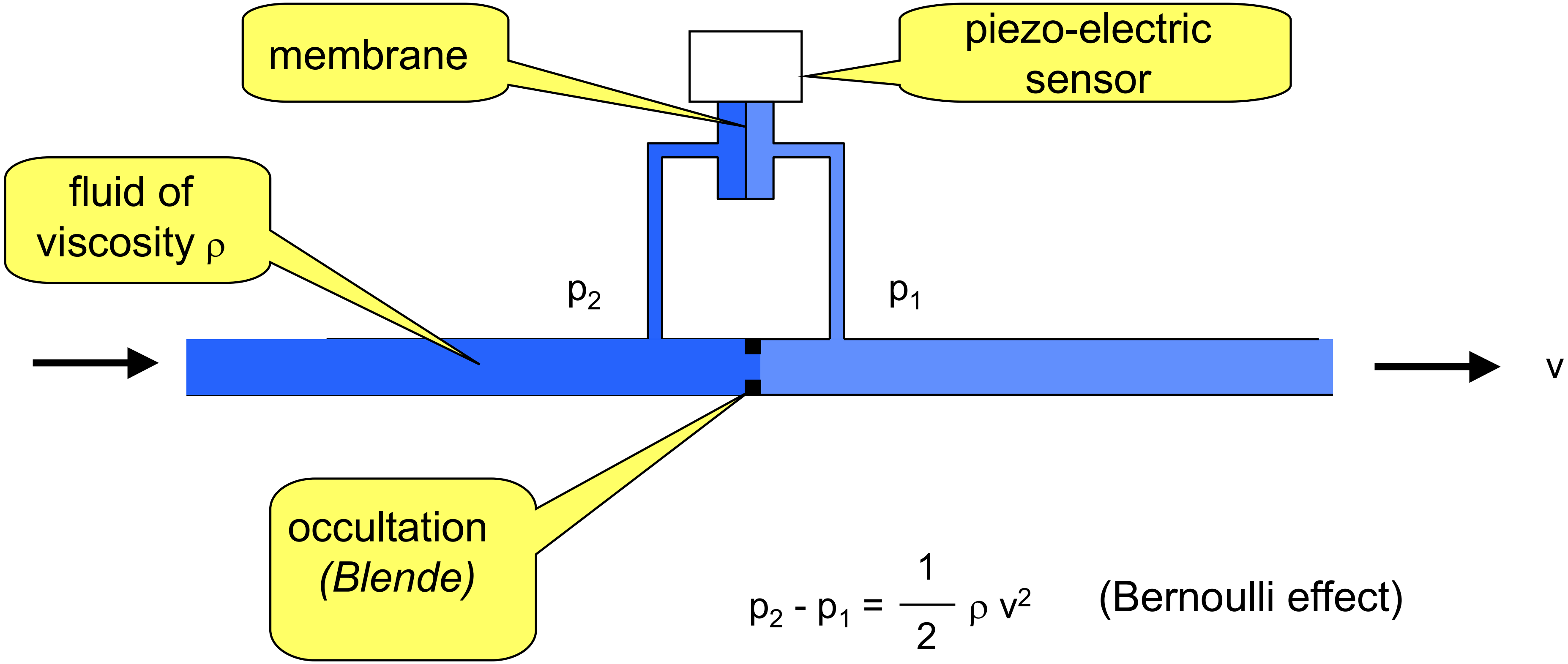






see Control Engineering, Aug 2003

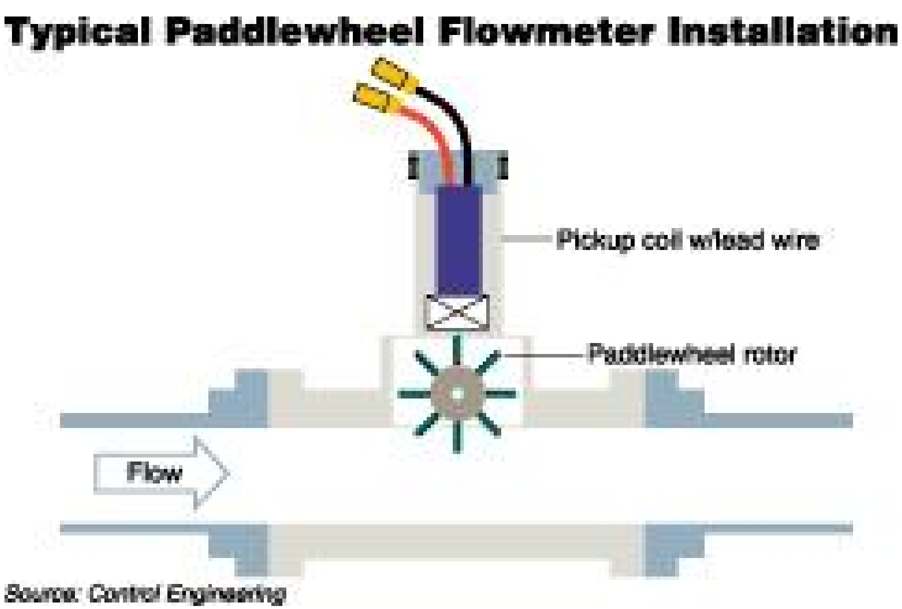
Flow velocity measurement: differential pressure

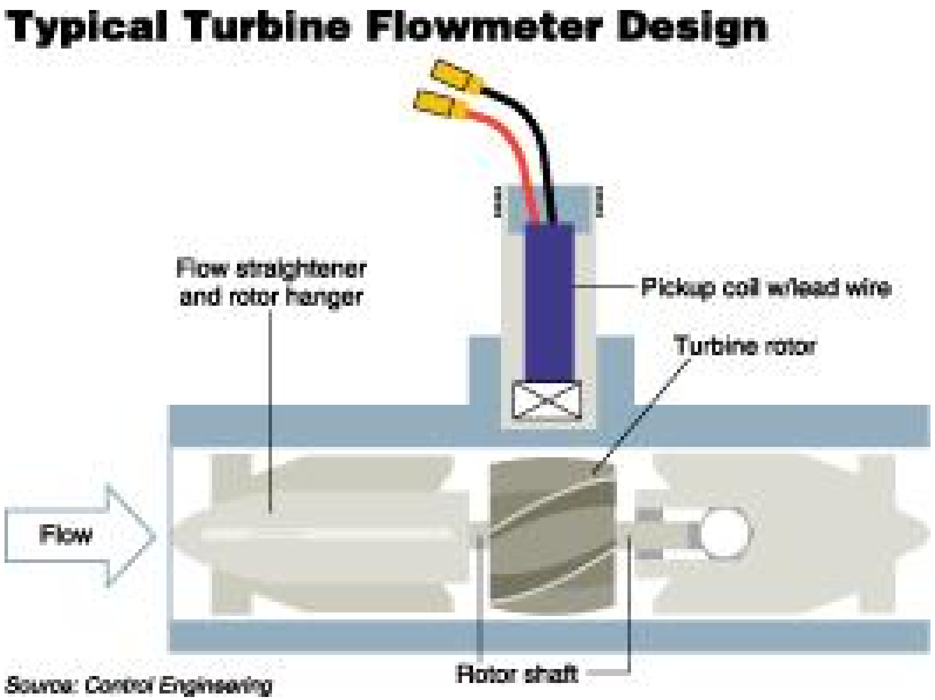


$$p_2 - p_1 = \frac{1}{2} \rho v^2$$ (Bernoulli effect)

the flow velocity is proportional to the square root of the pressure difference

Flow measurement





Other means:

Magnetic-dynamic
Coriolis
Ultra-sound

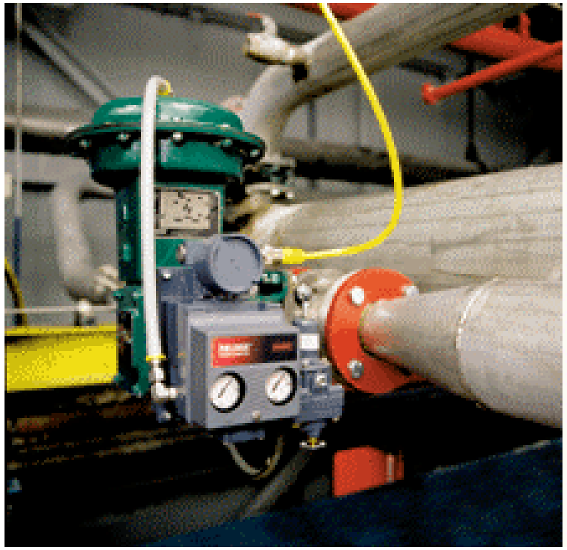
Flow measurement in a plant

2.1.4 Actors

Actors (Actuators)

Stellantriebe, *Servomoteurs*

About 10% of the field elements are actors (that influence the process).
Actors can be binary (on/off) or analog (e.g. variable speed drive)



The most common are:
- electric contactors (relays)
- heating elements
- pneumatic and hydraulic movers (valve, pump)
- electric motors (rotating and linear)
  - Solenoids,
  - DC motor
  - Asynchronous Motors (Induction)
  - Synchronous motors
  - Step motors, reluctance motors

Actors are controlled by the same electrical signal levels as sensors use (4..20mA, 0..10V, 0..24V, etc.) but at higher power levels (e.g. to directly move a contactor (*disjoncteur*).

Drives (variateurs de vitesse, Stellantriebe)

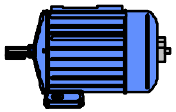
Variable speed drives control speed and acceleration and protect the motor (overcurrent, torque, temperature).
High-power drives can feed back energy to the grid when braking (inverters).
Drives is an own market (“Automation & Drives”)

simple motor control

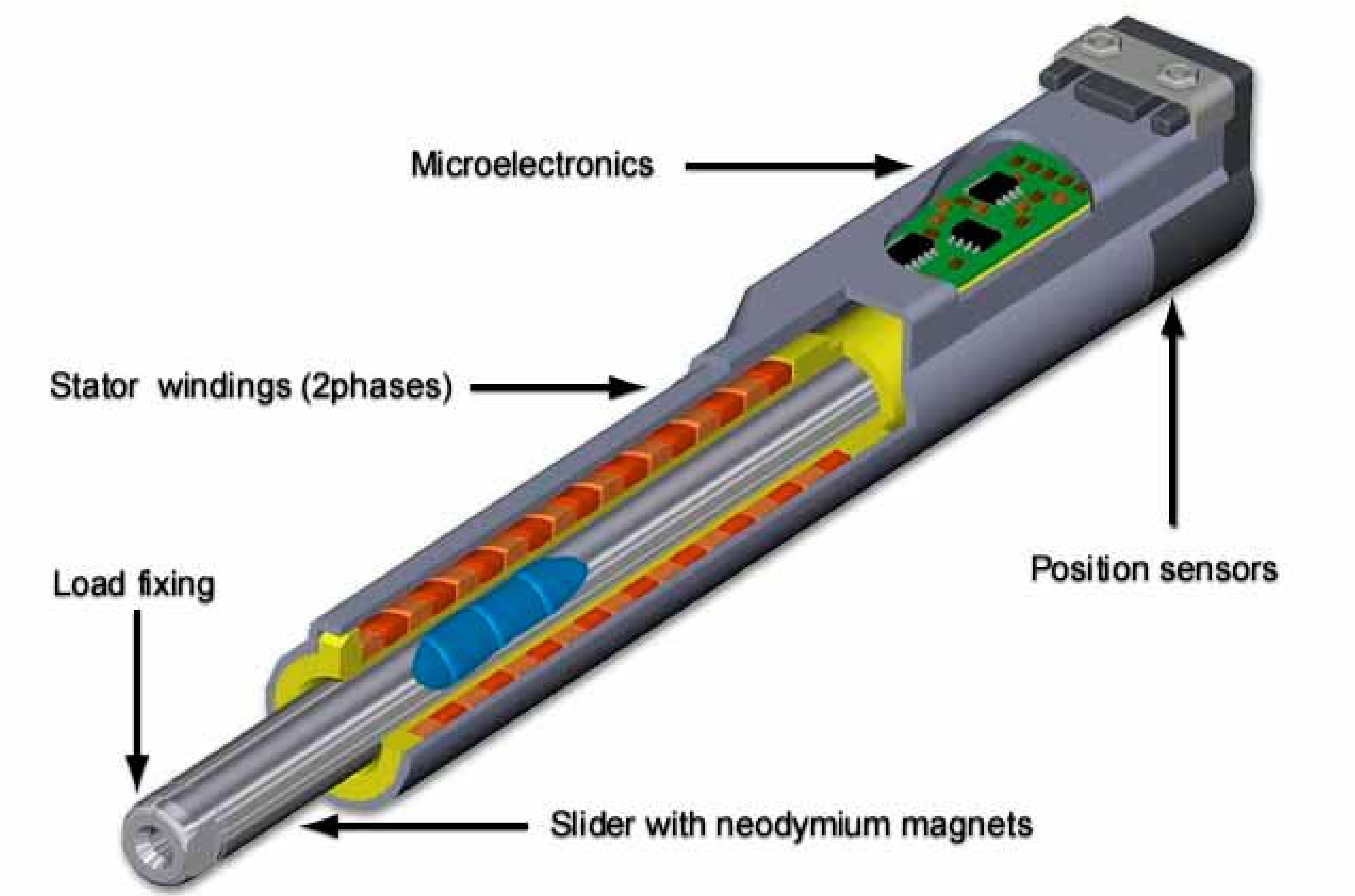
cabinet for power of > 10 kW

small drive control < 10 kW (Rockwell)

Motors are a separate business



Linear Motors



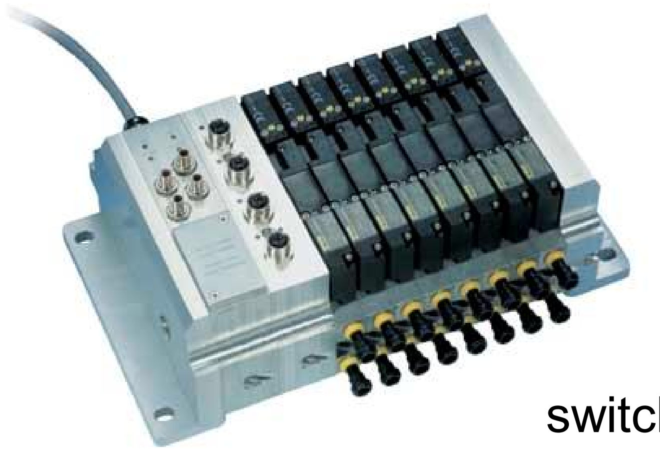
source: LinMot (/www.linmot.com)

Hydraulics and fluidics…

Pumps, valves, rods,…

I/P or E/P = electro-pneumatic transducers

fluidic switches



switchboard ("Ventilinsel")

source: www.bachofen.ch

2.1.5 Transducers

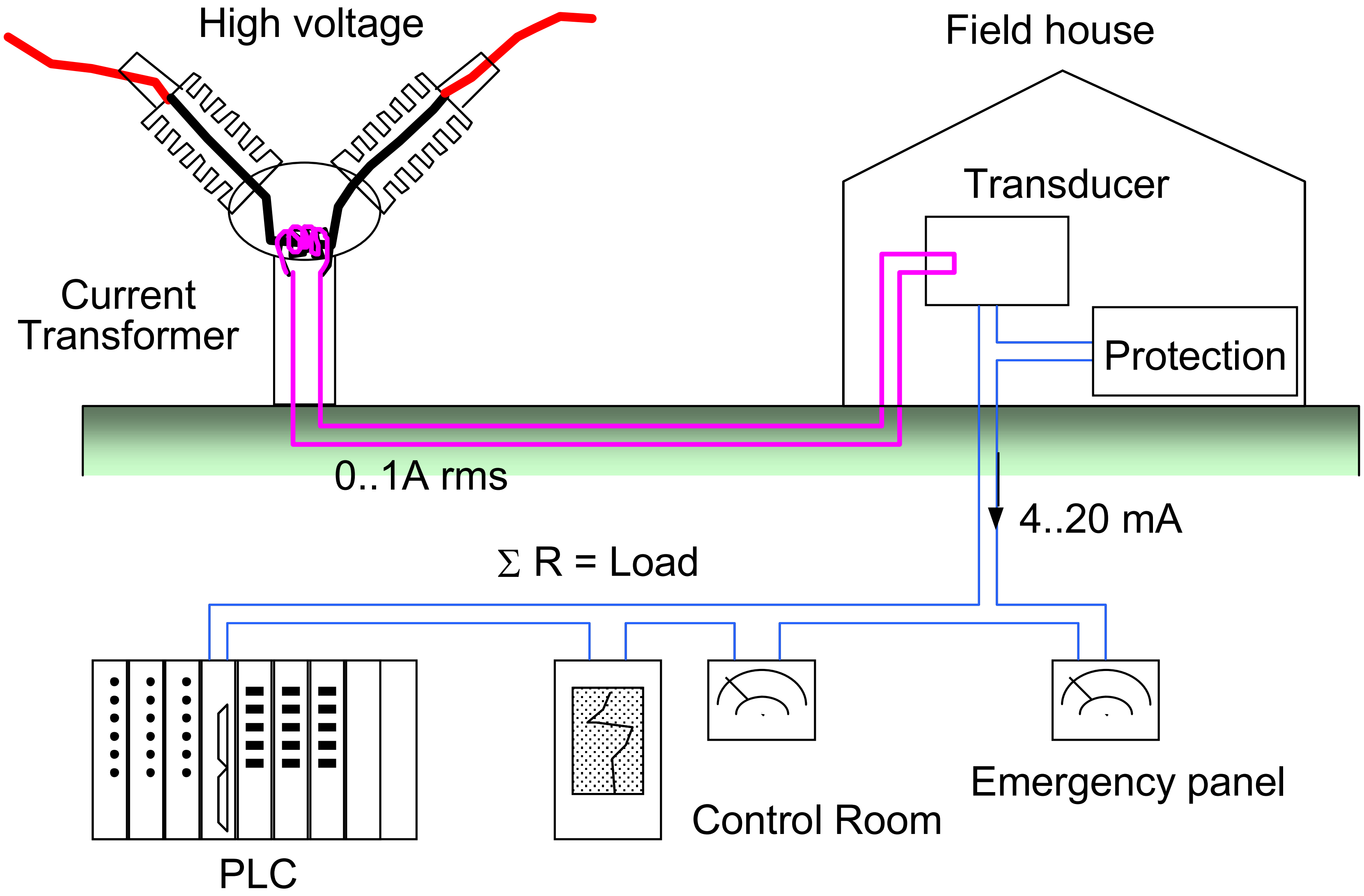
Transducer

A transducer converts the information supplied by a sensor (piezo, resistance,…) into a standardized signal which can be processed digitally.

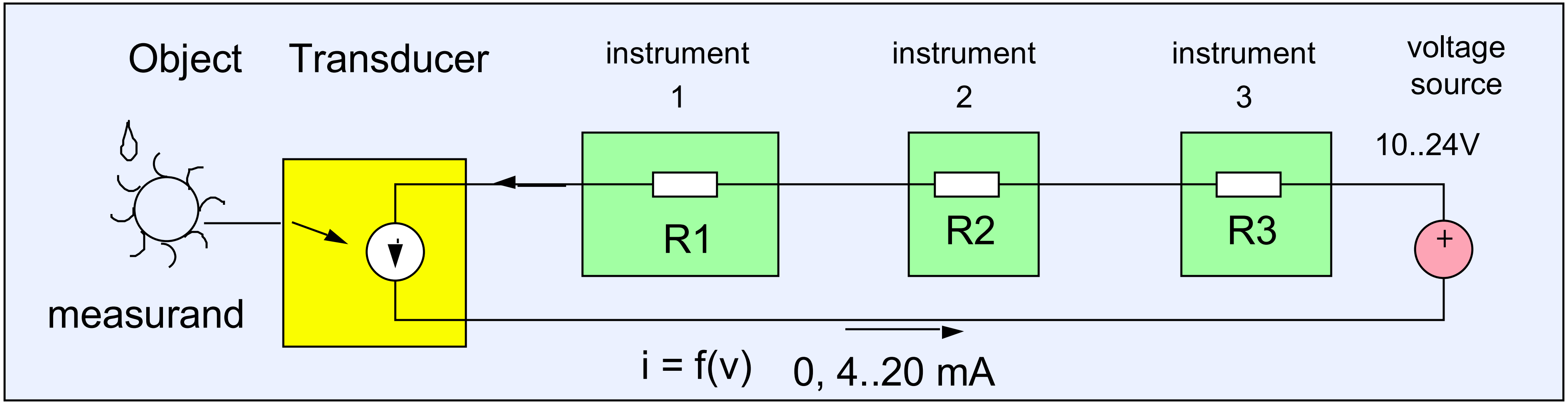
Some transducers have directly a digital (field bus) output and are integrated in the sensor.

Other are located at distances of several meters from the sensor.

Example of analog transducer



4-20 mA loop standard



The transducer acts as a current source which delivers a current between 4 and 20 mA, proportional to the measurand (*Messgrösse, valeur mesurée*).

Information is conveyed by a current, the voltage drop along the cable induces no error.

0 mA signals an error (wire disconnection)

The number of loads connected in series is limited by the operating voltage (10..24 V).
e.g. if (R1 + R2+ R3) = 1.5 kΩ, i = 24 / 1.5 = 16 mA, which is < 20 mA: NOT o.k.)

Simple devices are powered directly by the residual current (4mA) allowing to transmit signal and power through a single pair of wires.

Analog measurements processing in the transducer

Acquisition (*Erfassung*/Saisie)

Normalized Signals: 0-10V, 2-10V, (0/4-20mA), ±20mA,
Resistance thermometer (Pt100),
Thermoelement

Shaping (*Aufbereitung*/conditionnement)

Filtering against 50Hz/60Hz noise and its harmonics
Scaling,
Linearisation of sensors (Pt100, FeConst), correction (square root for flow).
Averaging and Computation of Root Mean Square (Effektivwert, valeur efficace),
Analog-Digital Conversion

Plausibility

Range, Limit supervision, Wire integrity
Error report, diagnostic, disabling.

Combined measurement

Correction of pressure and temperature measurement for moist gases,
correction of level in function of pressure,
power and energy computation, cumulative measurements

2.1.6 Instrumentation diagrams: P&ID

Instrumentation Diagrams

Similarly to electrical schemas, the control industry (especially the chemical and process industry) describes its plants and their instrumentation by a

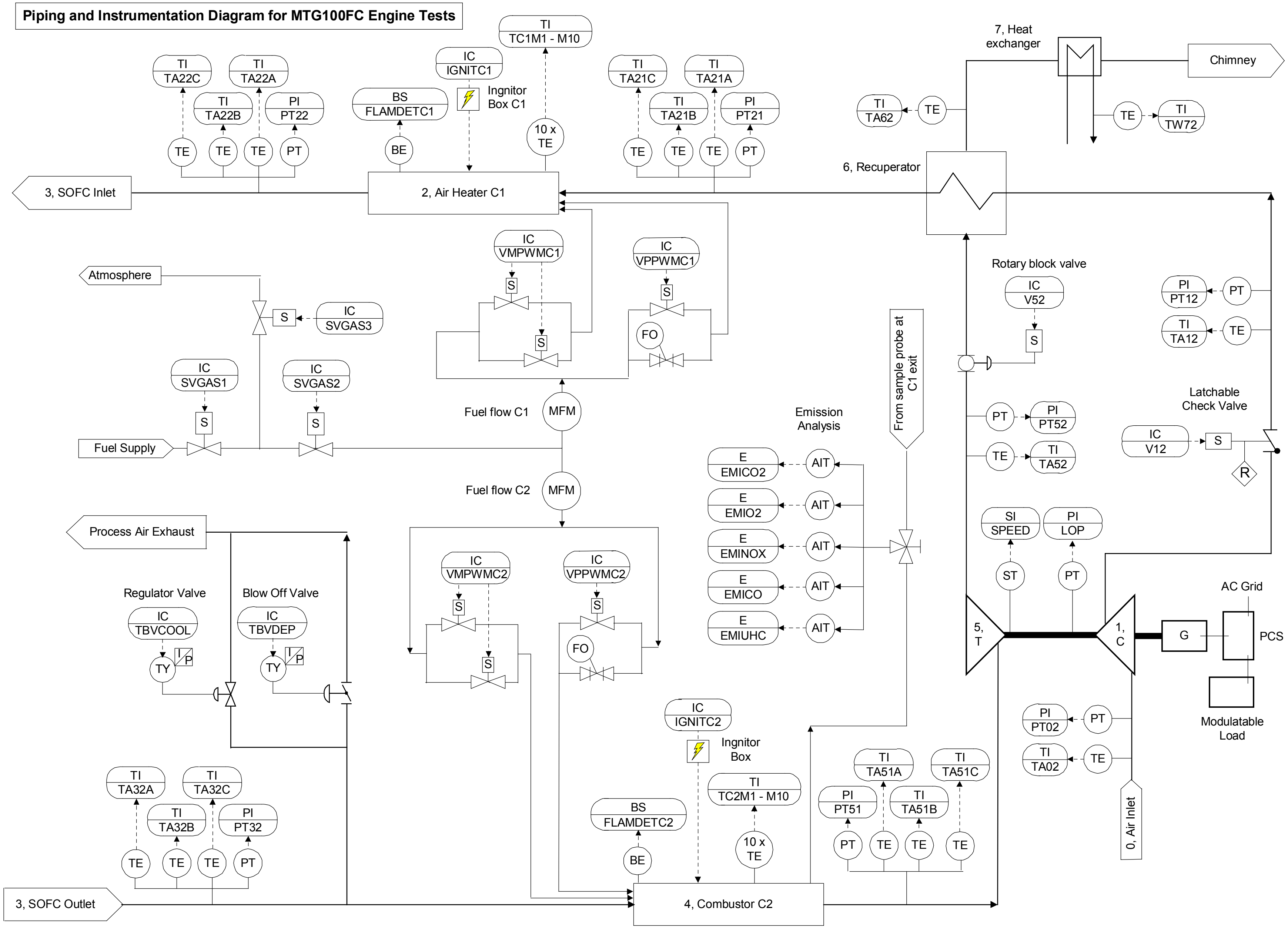
P&ID (pronounce P.N.I.D.) (Piping and Instrumentation Diagram),
sometimes called P&WD (Piping and wiring diagrams)

The P&ID shows the flows in a plant (in the chemical or process industry) and the corresponding sensors or actors.

At the same time, the P&ID gives a name ("tag") to each sensor and actor, along with additional parameters.

This tag identifies a "point" not only on the screens and controllers, but also on the objects in the field.

P&ID example


Piping and Instrumentation Diagram for MTG100FC Engine Tests
TI TC1M1 - M10
IC IGNITC1
Ingnitor Box C1
10 x TE
BS FLAMDETC1
BE
TI TA22C
TI TA22A
TI TA22B
PI PT22
TE
PT
TI TA21C
TI TA21A
TI TA21B
PI PT21
3, SOFC Inlet
2, Air Heater C1
7, Heat exchanger
Chimney
TI TA62
TI TW72
6, Recuperator
IC VMPWMC1
IC VPPWMC1
S
FO
Atmosphere
IC SVGAS3
IC SVGAS1
IC SVGAS2
Fuel Supply
Fuel flow C1
MFM
Fuel flow C2
Rotary block valve
IC V52
PI PT12
TI TA12
PT
PI PT52
TI TA52
Latchable Check Valve
IC V12
R
Emission Analysis
From sample probe at C1 exit
E EMICO2
E EMIO2
E EMINOX
E EMICO
E EMIUHC
AIT
Process Air Exhaust
IC VMPWMC2
IC VPPWMC2
Regulator Valve
IC TBVCOOL
Blow Off Valve
IC TBVDEP
TY
I/P
SI SPEED
PI LOP
ST
AC Grid
5, T
1, C
G
PCS
Modulatable Load
PI PT02
TI TA02
IC IGNITC2
Ingnitor Box
TI TC2M1 - M10
BS FLAMDETC2
TI TA51A
TI TA51C
PI PT51
TI TA51B
TI TA32A
TI TA32C
TI TA32B
PI PT32
0, Air Inlet
3, SOFC Outlet
4, Combustor C2

P&ID

The P&ID mixes pneumatic / hydraulic elements, electrical elements and instruments on the same diagram

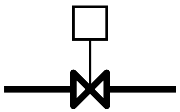
It uses a set of symbols defined in the ISA S5.1 standard.

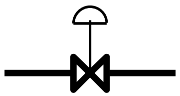
Examples of pneumatic / hydraulic symbols:

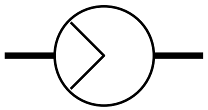
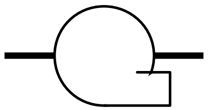
pipe

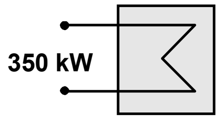

valve


one-way valve (diode)

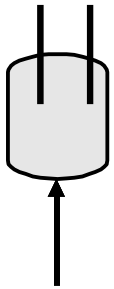

binary (or solenoid) valve (on/off)

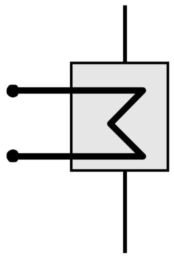

analog valve (continuous)


pump, also


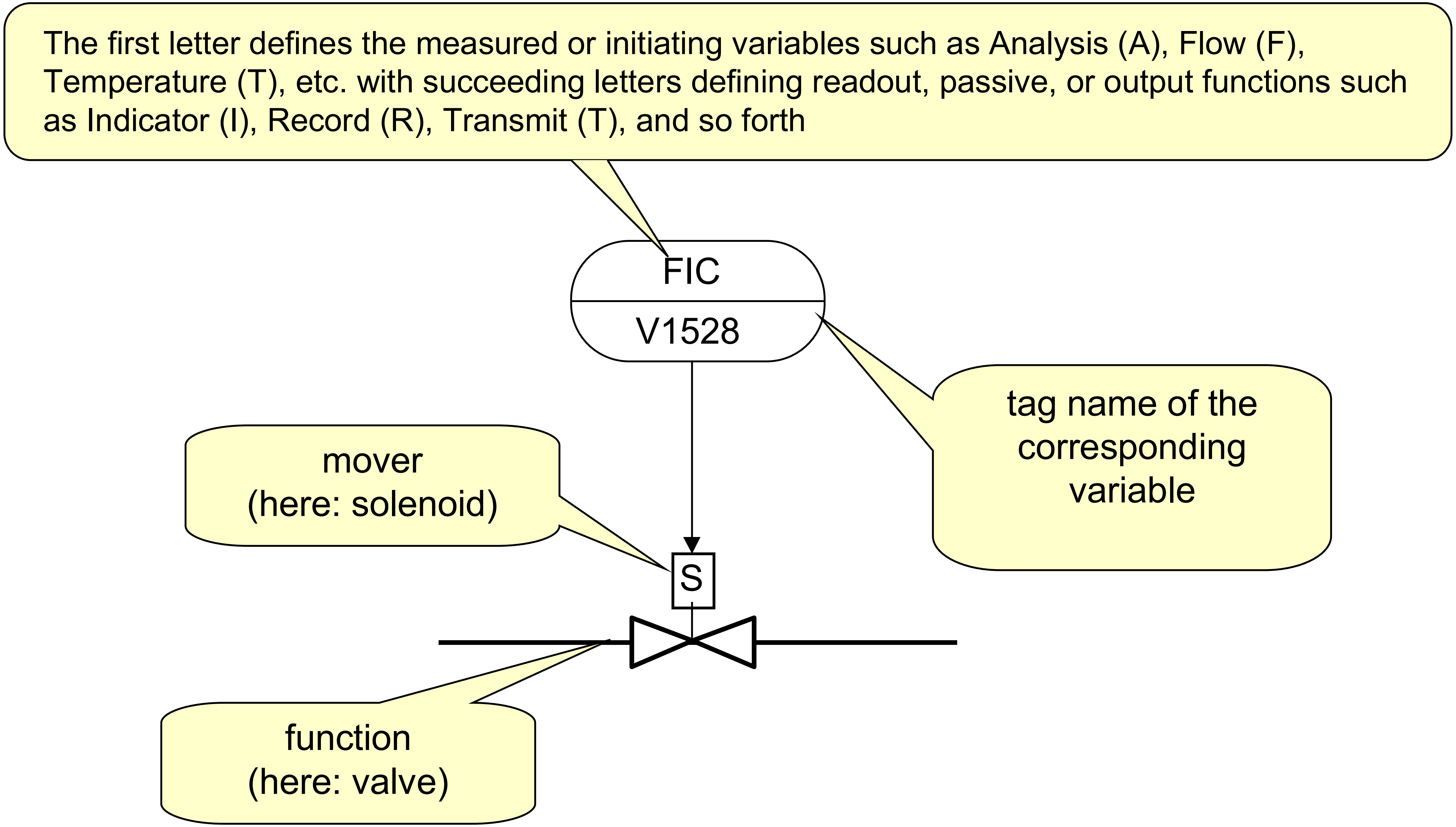


heater

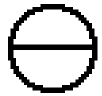
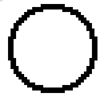
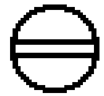
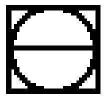
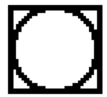
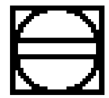
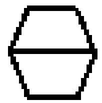
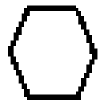
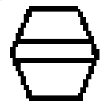
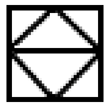
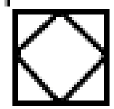
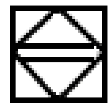

vessel / reactor


heat exchanger

Instrumentation identification



ISA S5.1 General instrument or function symbols

	Primary location accessible to operator	Field mounted	Auxiliary location accessible to operator
Discrete instruments	1 	2 	3 
Shared display, shared control	4 	5 	6 
Computer function	7 	8 	9 
Programmable logic control	10 	11 	12 

1. Symbol size may vary according to the user's needs and the type of document.
2. Abbreviations of the user's choice may be used when necessary to specify location.
3. Inaccessible (behind the panel) devices may be depicted using the same symbol but with a dashed horizontal bar.

Source: Control Engineering with data from ISA S5.1 standard

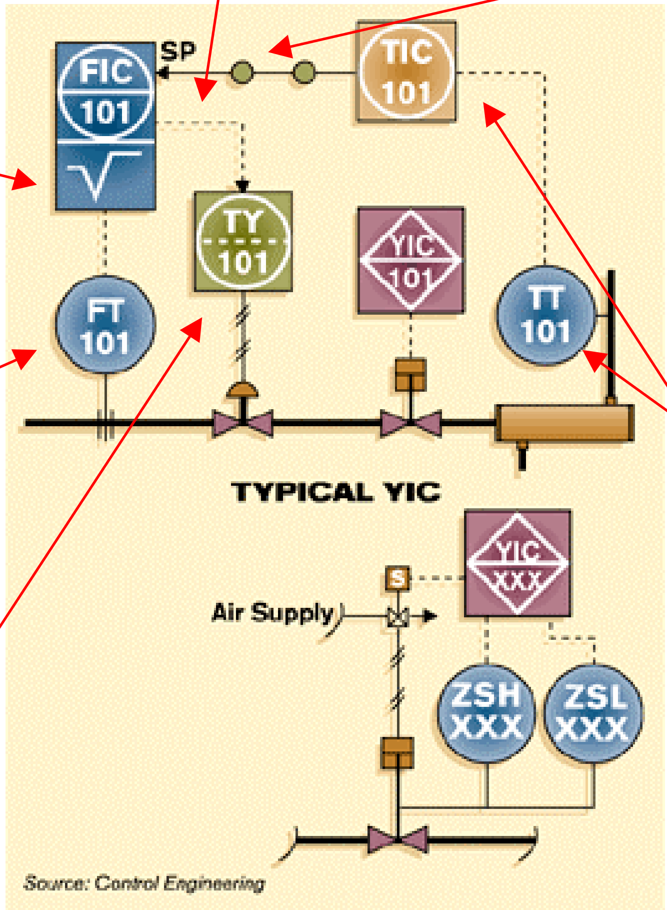
Example of P&ID

The output of FIC 101 is an electrical signal to TY 101 located in an inaccessible or behind-the-panel-board location.

Square root extraction of the input signal is part of FIC 101's functionality.

FT101 is a field-mounted flow transmitter connected via electrical signals (dotted line) to flow indicating controller FIC 101 located in a shared control/display device

The output signal from TY 101 is a pneumatic signal (line with double forward slash marks) making TY 101 an I/P (current to pneumatic transducer)



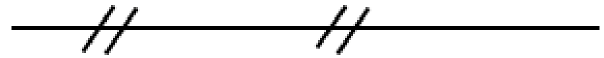
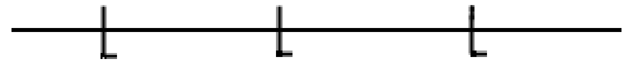
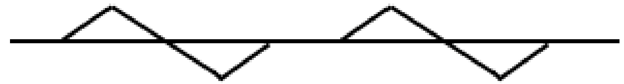
TIC 101's output is connected via an internal software or data link (line with bubbles) to the setpoint (SP) of FIC 101 to form a cascade control strategy

TT 101 and TIC 101 are similar to FT 101 and FIC 101 but are measuring, indicating, and controlling temperature

The ISA code for instrument type

	First letter	
	Measured or initiating variable	**Modifier**
A	Analysis	
B	Burner, combustion	
C	User's choice	
D	User's choice	Differential
E	Voltage	
F	Flow rate	Ration (fraction)
G	User's choice	
H	Hand	
I	Current (electrical)	
J	Power	Scan
K	Time, time schedule	Time rate of change
L	Level	
M	User's choice	Momentary
N	User's choice	
O	User's choice	
P	Pressure, vacuum	
Q	Quantity	Integrate, totalizer
R	Radiation	
S	Speed, frequency	Safety
T	Temperature	
U	Multivariable	
V	Vibration, mechanical analysis	
W	Weight, force	
X	Unclassified	X axis
Y	Event, state, or presence	Y axis
Z	Position, dimension	Z axis

Common connecting lines

Connection to process, or instrument supply	
Pneumatic signal	
Electric signal	
Capillary tubing (filled system)	
Hydraulic signal	
Electromagnetic or sonic signal (guided)	
Internal system link (software or data link)	

Source: Control Engineering with data from ISA S5.1 standard

2.1.7 Protection Classes

German IP-Protection classes

Erste Kenn-ziffer	Berührungsschutz	Fremdkörperschutz	Zweite Kenn-ziffer	Wasserschutz
0	Kein besonderer Schutz		0	Kein besonderer Schutz
1	Gegen große Körper-flächen	Große Fremdkörper Durch-messer > 50 mm	1	Gegen senkrecht fallendes Tropfwasser
2	Gegen Finger oder ähnlich große Ge-genstände	Mittelgroße Fremdkörper Durchmesser > 12 mm	2	Gegen schräg fallendes Tropfwasser (bis 15° Ab-weichung von der Senk-rechte)
3	Gegen Werkzeug, Drähte und ähnliches mit einer Dicke > 2,5 mm	Kleine Fremdkörper Durchmesser > 2,5 mm	3	Gegen Sprühwasser (be-liebige Richtung bis 60° Abweichung von der Senk-rechte
4	Vollständiger Schutz	Kornförmige Fremdkörper Durchmesser > 1 mm	4	Gegen Spritzwasser aus allen Richtungen
5		Staubgeschützt, Staubab-lagerungen sind zulässig, dürfen aber in ihrer Menge nicht die Funktion des Geräts gefährden	5	Gegen Strahlwasser aus einer Düse aus allen Rich-tungen
6	Vollständiger Schutz	Staubdicht	6	Gegen Überflutung
			7	Gegen Eintauchen
			8	Gegen Untertauchen

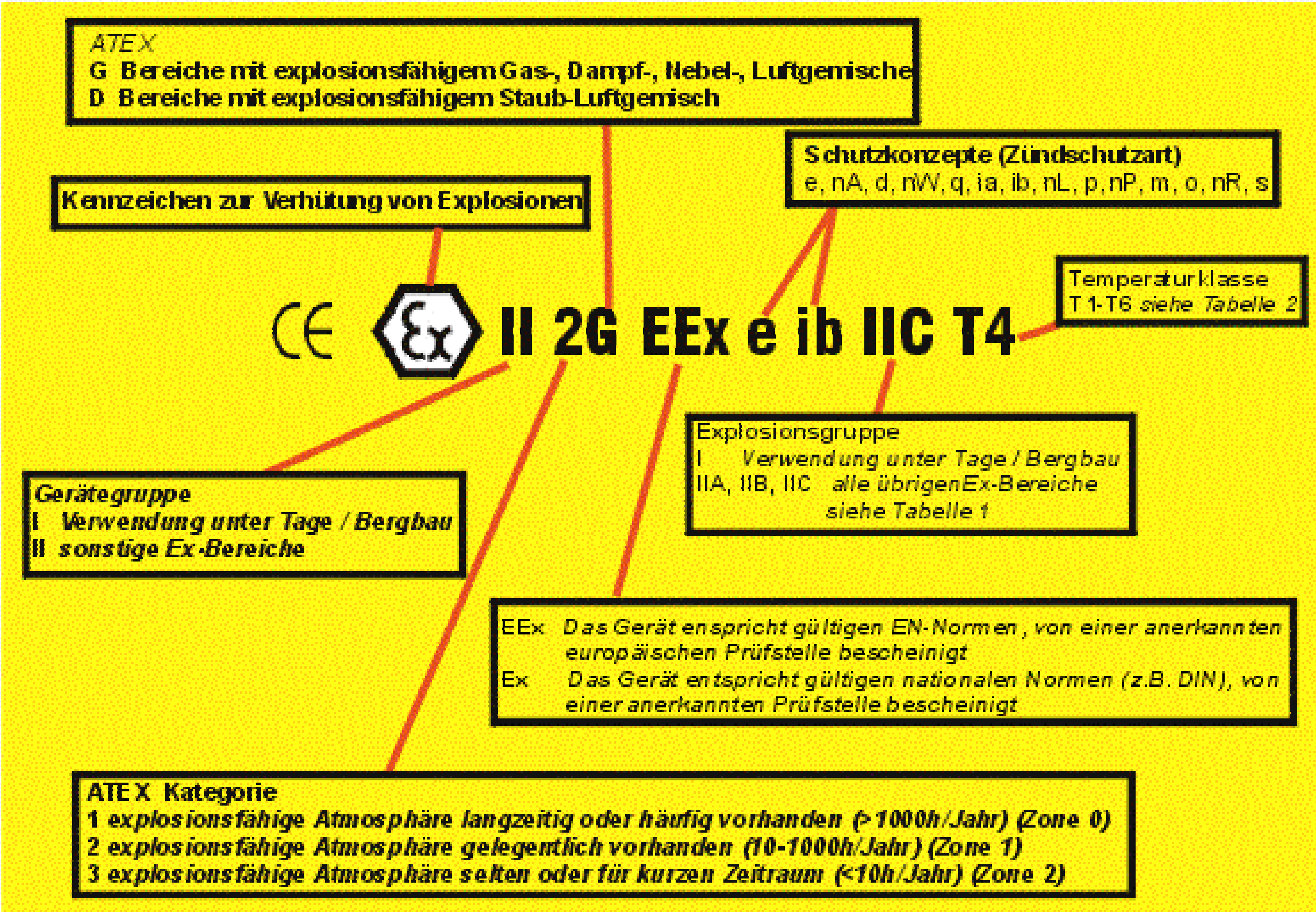
Explosion protection

Instruments that operate in explosive environments
(e.g. petrochemical, pharmaceutical, coal mines,...) are subject to particular restrictions.

e.g.
They may not contain anything that can produce sparks or high heat,
such as electrolytic capacitors or batteries without current limitation.
Their design or programming may not be altered after their acceptance.
Their price is higher than that of standard devices because they have to undergo
strict testing (Typentest, type test) by a qualified authority (TÜV in Germany)

Such devices are called Eex - or "intrinsic safety devices" (*Eigensichere Geräte*, *"Ex-Schutz", **protection anti-déflagrante, "Ex"***) and are identified by the following logo:

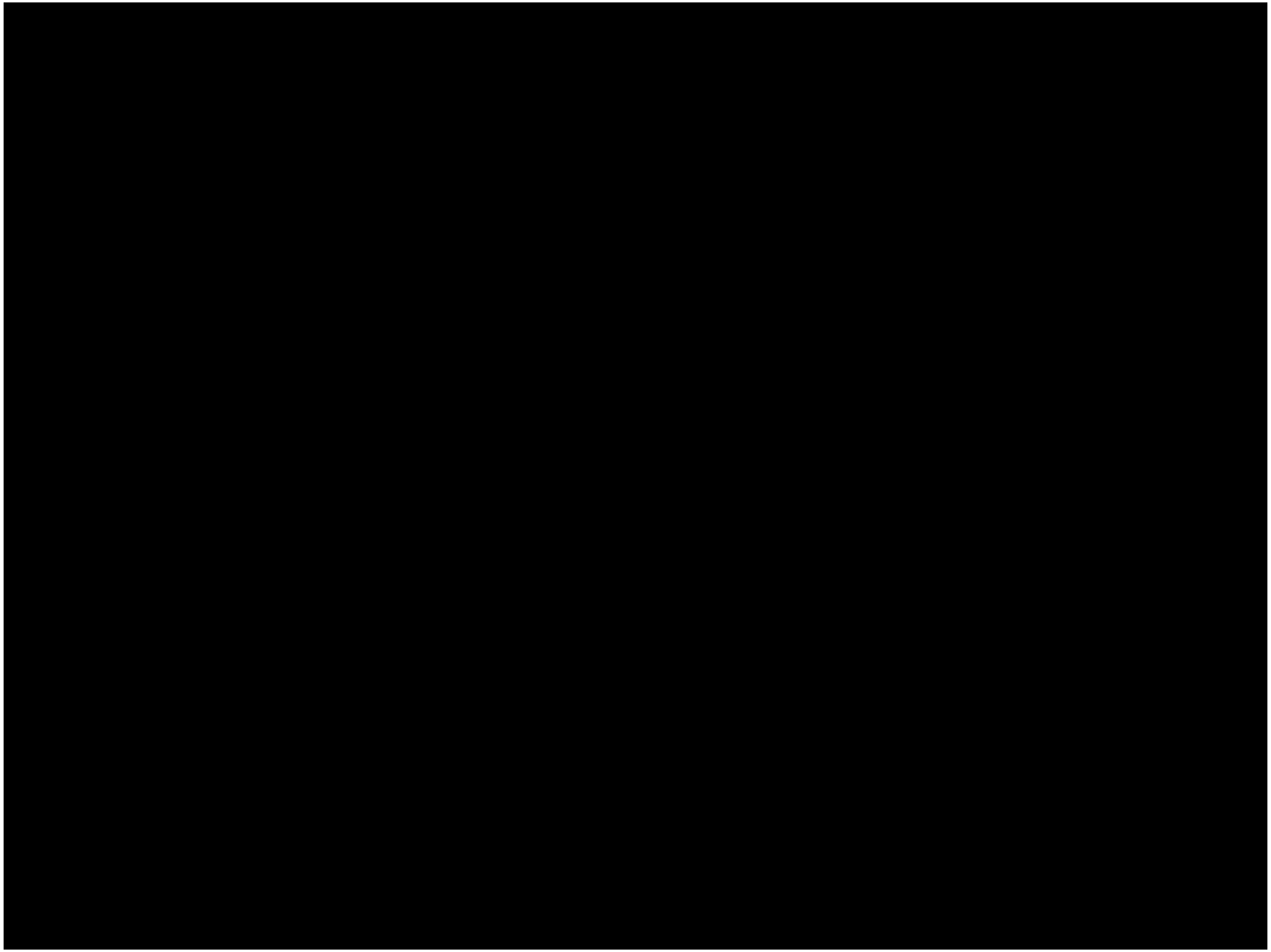
European Explosion-Proof Code



Eex-devices are "safe" (certified) to be used in an explosive environment.
They must have passed a type test at TÜF (Germany), UL (USA),...

Swiss Norm: "Verordnung über Geräte und Schutzsysteme in explosionsgefährdeten Bereichen"

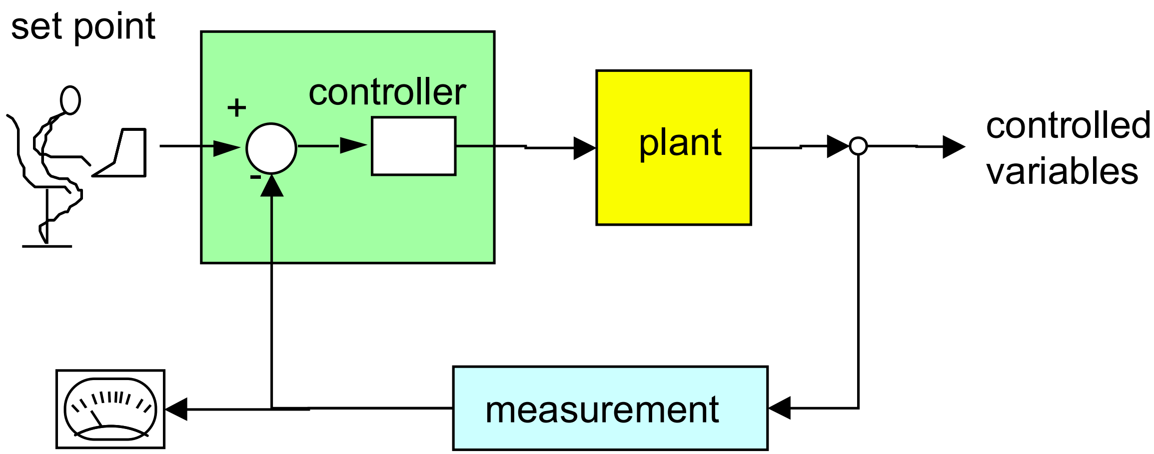
Field Device: faceplate (movie)



Assessment

How are binary process variables measured ?
How are analogue process variables measured ?
How is temperature measured ?
What is the difference between a thermocouple and a thermoresistance ?
How is position measured (analog and digital) ?
What is a Grey encoder ?
How is speed measured ?
How is force measured ?
What is a P&ID ?
What is a transducer ?
How does a 4..20 mA loop operate ?

EPFL



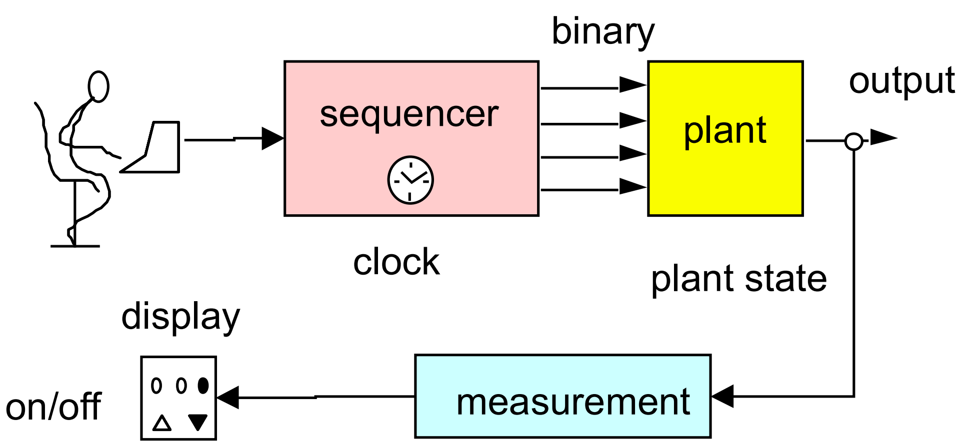
2.2 Control of continuous processes
Régulation de systèmes continus
Regelung stetiger Strecken

Prof. Dr. H. Kirrmann
EPFL / ABB Research Center, Baden, Switzerland

Open loop and closed loop

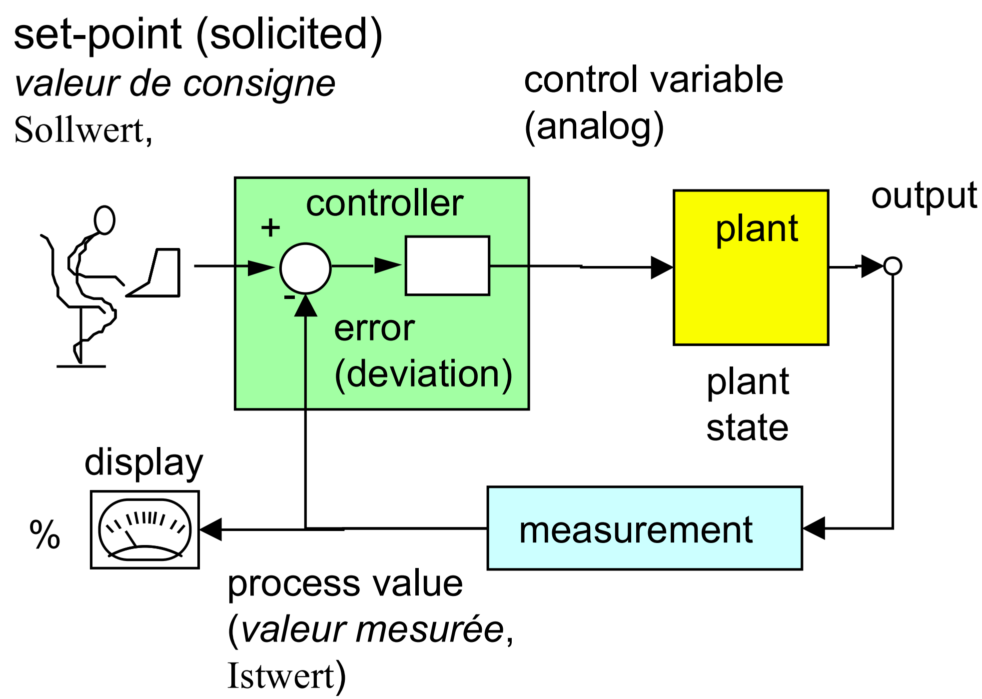
open-loop control / command
(commande / pilotage, steuern,)

keywords: sequential / combinatorial, binary variables, discrete processes, "batch control", "manufacturing"



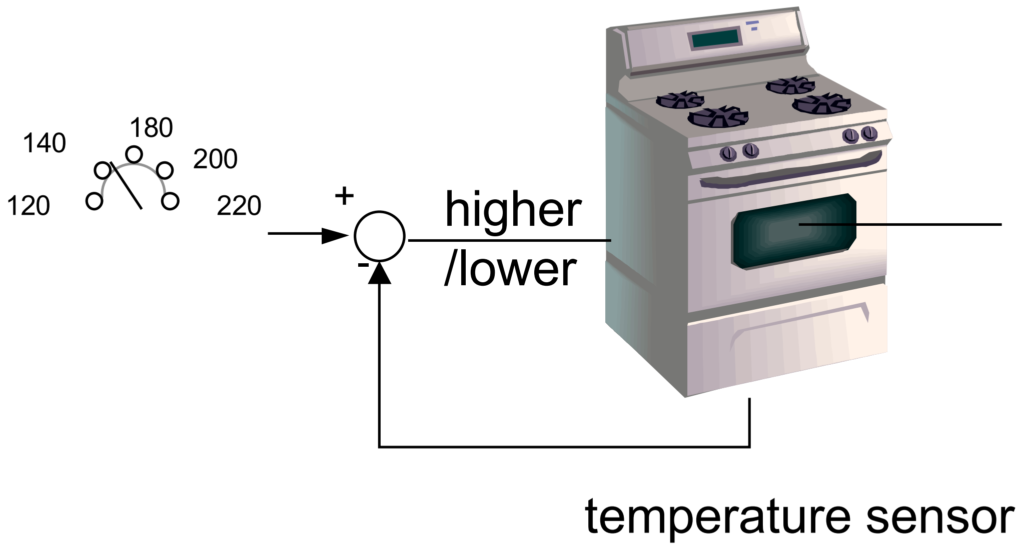
closed-loop control / regulation
(régulation, Regelung)

keywords: feedback, analog variables, continuous processes, "process control"



Why do we need closed loop in continuous processes ?

open loop: temperature is imprecise, depends on ambient temperature and cooking quantity
but time of heating can be modulated.



closed loop:
temperature closely controlled, requires measurement of the output variable (temperature)

2.2 Control

Controllers

This is an intuitive introduction to automatic control.
The approach shown here is only valid with benign plants (most are).

For a theoretical development, the courses of Prof. Longchamp and Prof. Bonvin are recommended.

The two most popular controllers in industry are presented:

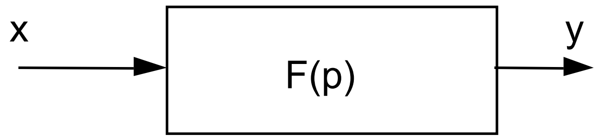
- the two-point controller

-the PID controller

2.2.1 Plant Modeling

Modeling: Continuous processes

Examples: Drives, Ovens, Chemical Reactors



Continuous plants have states which can be described by a continuous (analog) variable (temperature, voltage, speed,...)

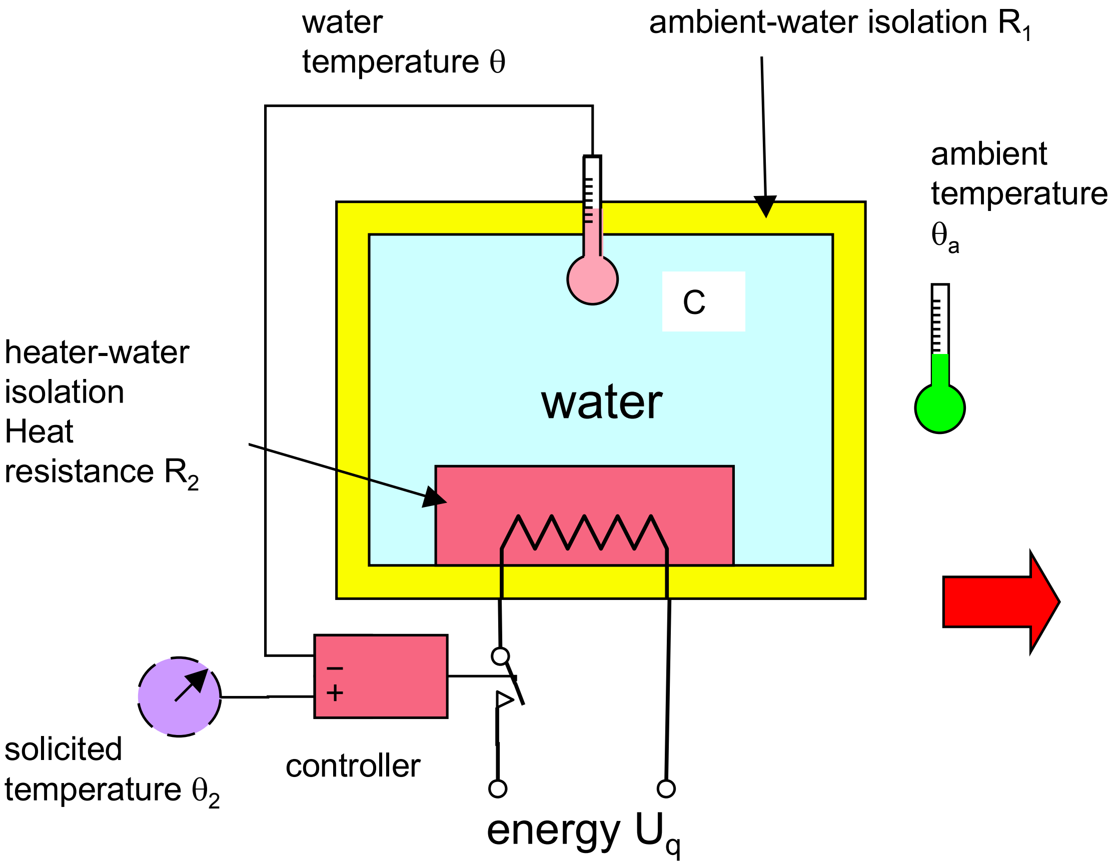
Between plant input and plant output, there exists a fixed relation which can be described by a continuous model (transfer function).

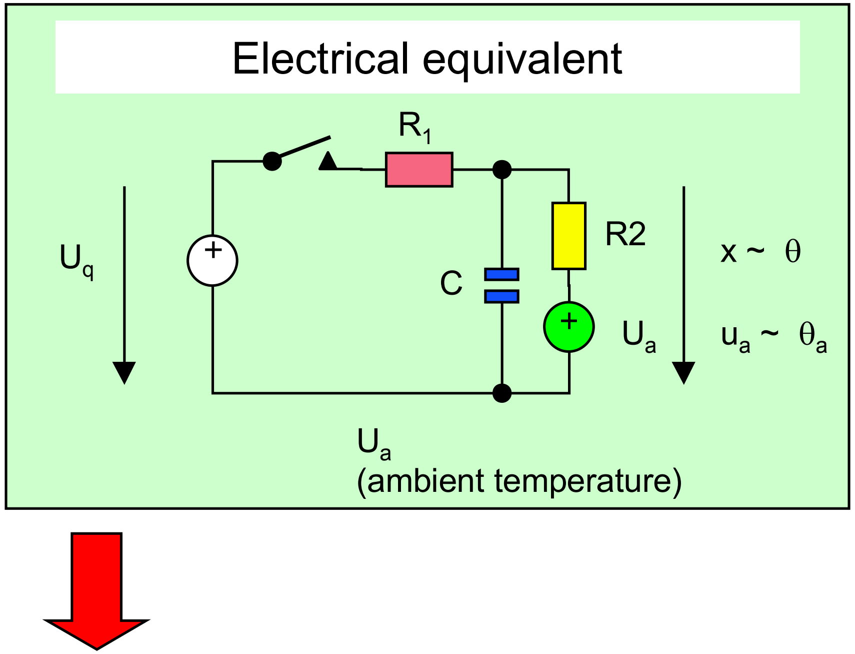
Continuous plants are mostly reversible and monotone:
This is the condition necessary to control them.

The transfer function may be described by a differential equation, simplified to a Laplace or a z-transform when the system is linear.

The principal control task in relation with a continuous process is its *regulation* (maintain the state at a determined level)

Modeling: a non-linear plant and its electrical equivalent





heating

$$\frac{dx}{dt} = -x\frac{(\frac{1}{R_1}+\frac{1}{R_2})}{C} + \frac{1}{R_1 C}U_q + \frac{1}{R_2 C}Ua$$

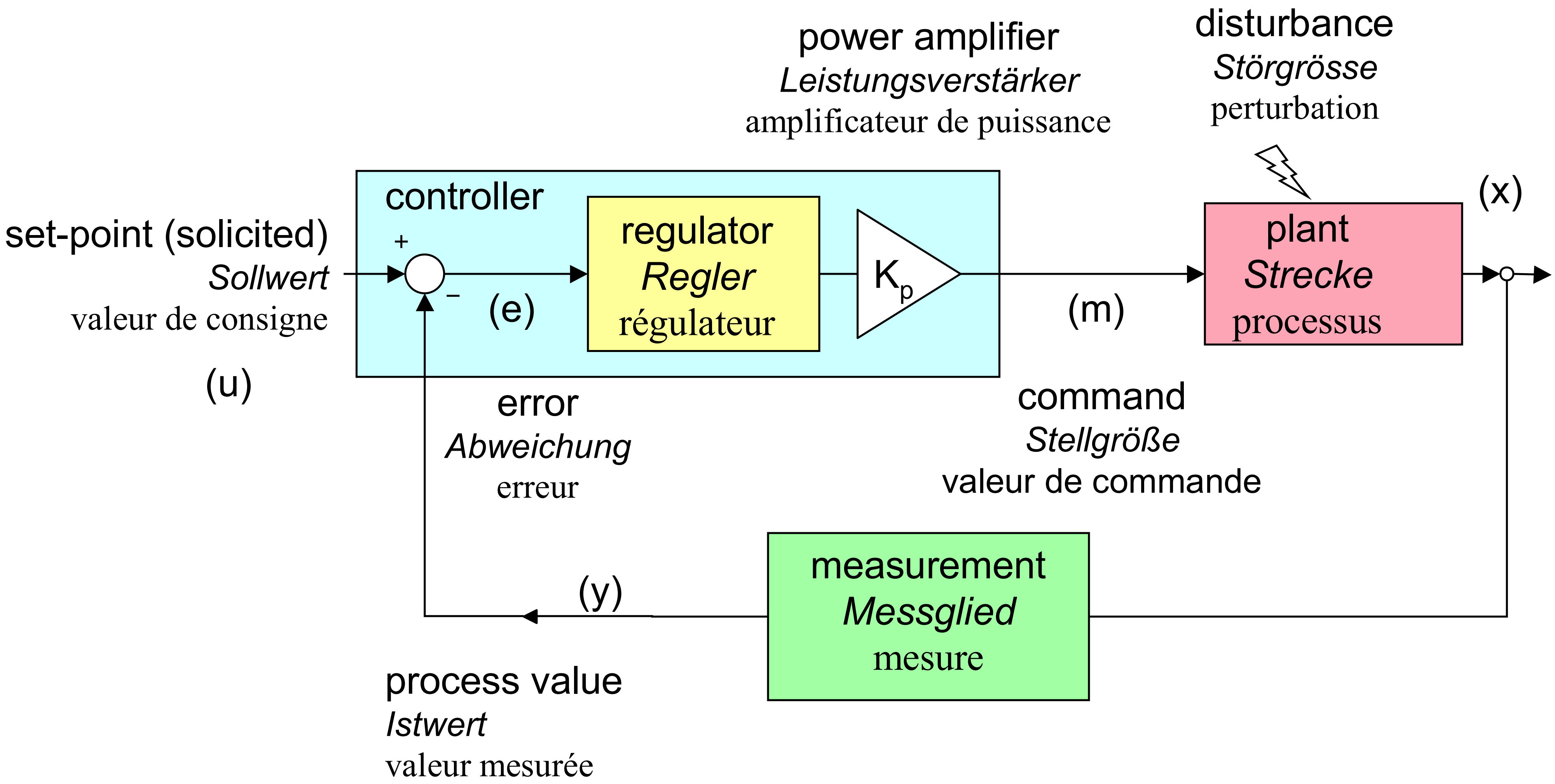
cooling

$$\frac{dx}{dt} = -x\frac{1}{R_2 C} + \frac{1}{R_2 C}Ux$$

$$\frac{dx}{dt} = -Ax + Bu + C$$

$$\frac{dx}{dt} = -Dx + E$$

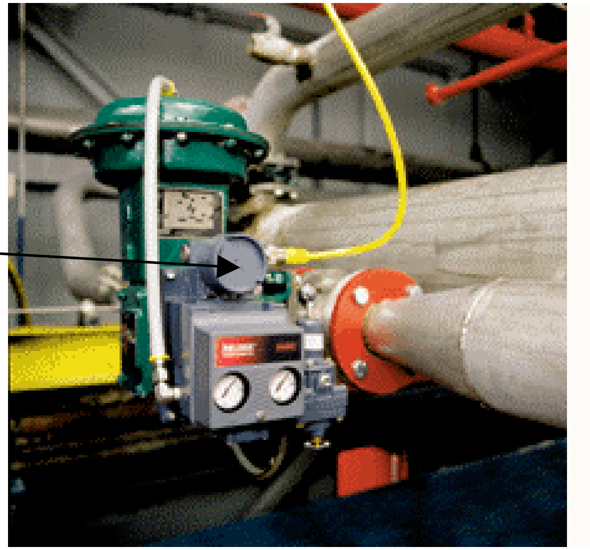
Controller (régulateur, Regler)



the regulator (controller) can be implemented by mechanical elements, electrical elements, computers,...

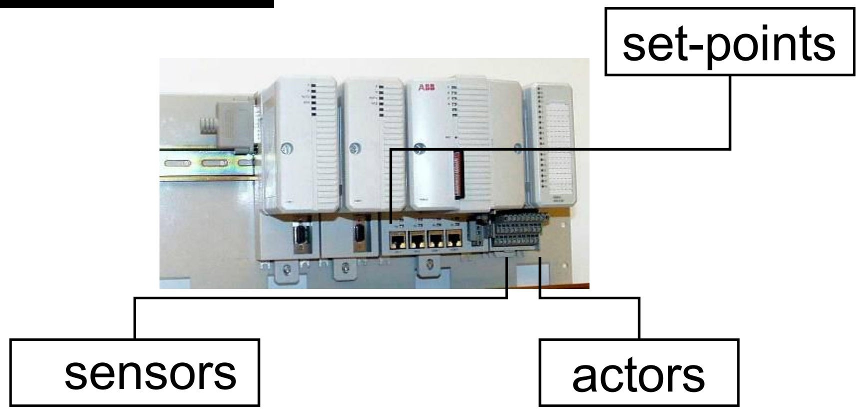
Where is that controller located ?

- in the sensor
or in the actuator
(analog PIDs)

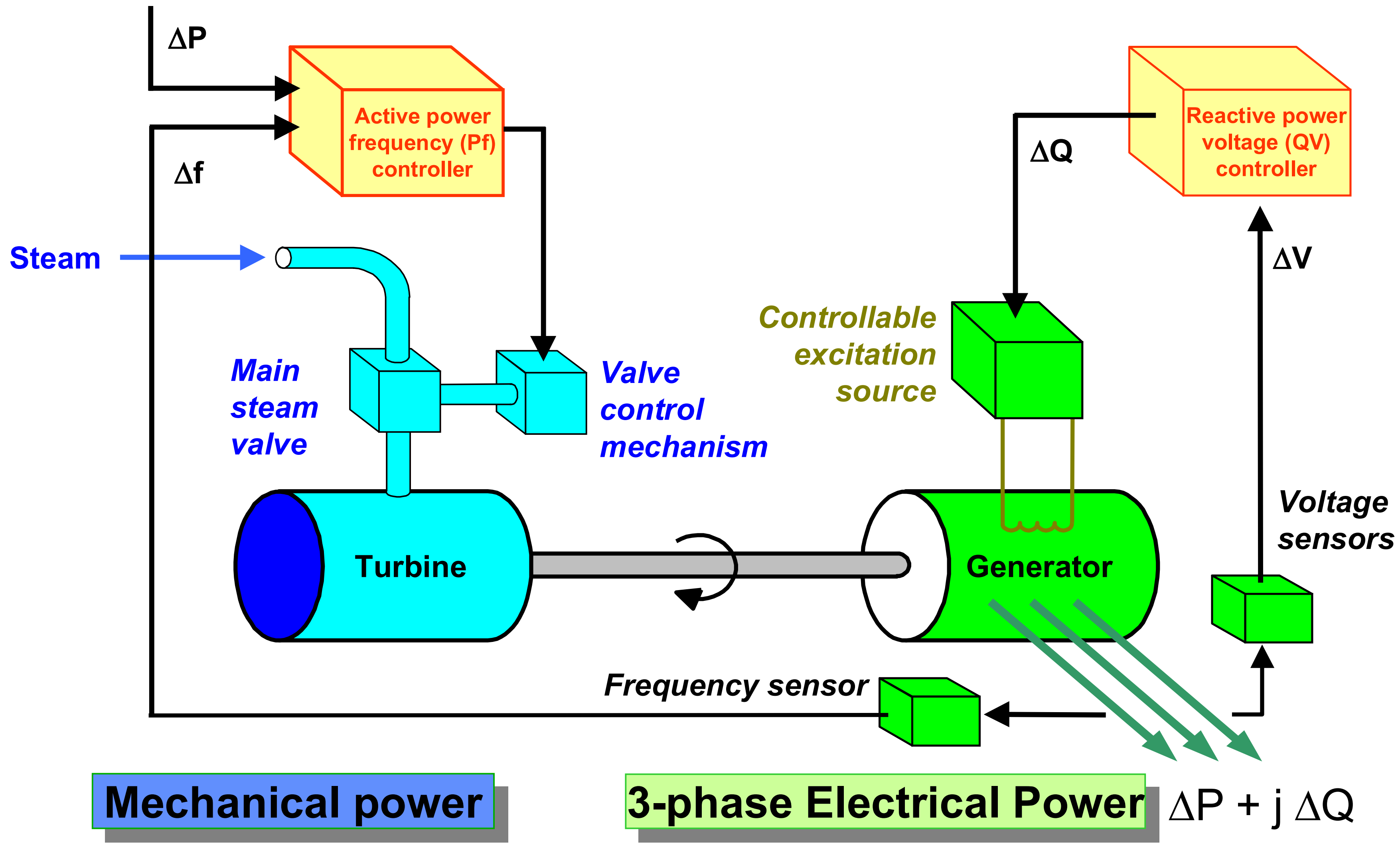


- as a separate device (analog PIDs)
(some times combined with a recorder)

- as an algorithm in a computer
(that can handle numerous "loops").



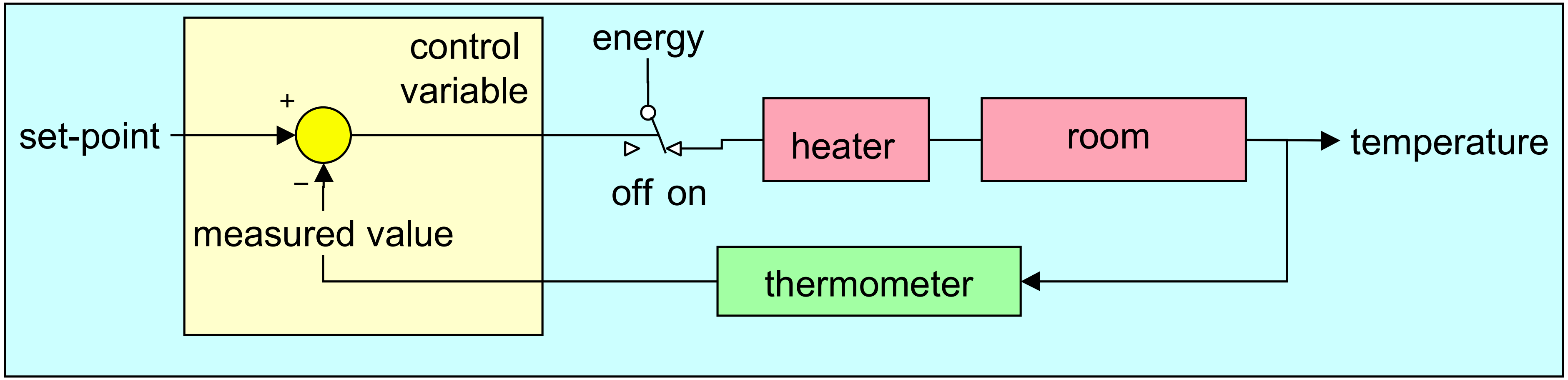
Electricity Generator



2.2.2 On/Off (two-point) controller

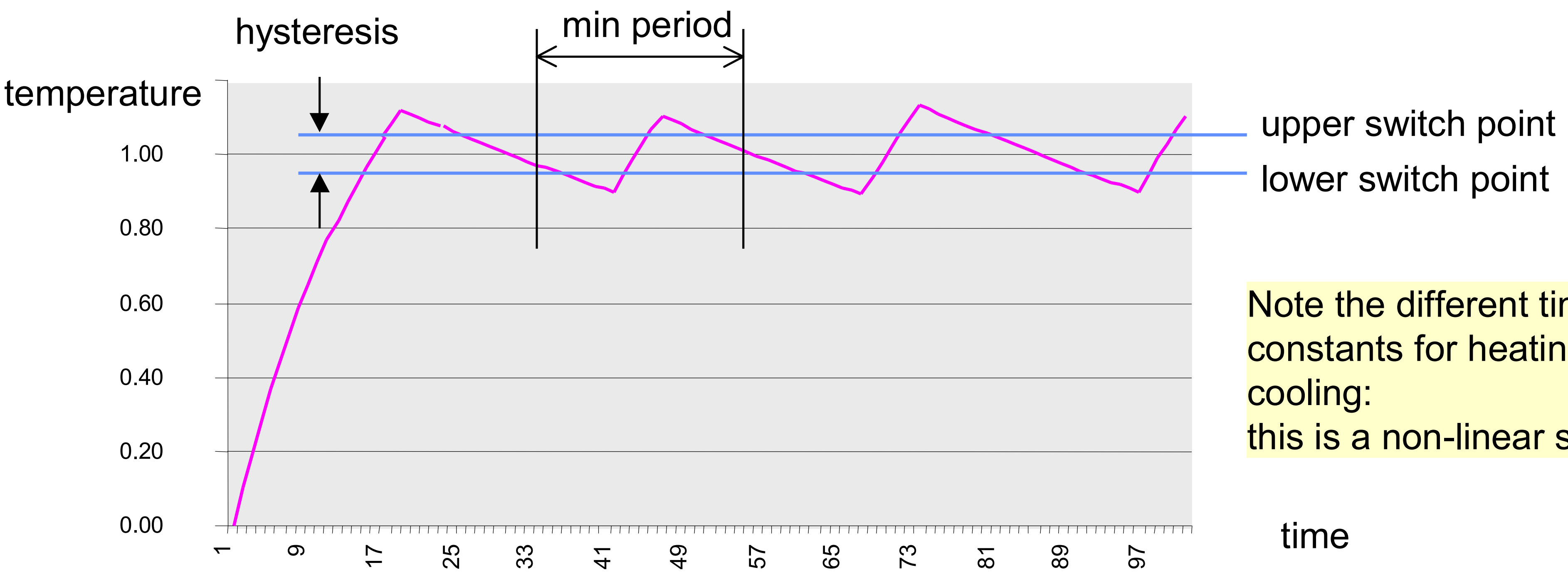
Two-point controller: principle

The two-point controller (or regulator, *Zweipunktregler*, Régulateur tout ou rien)
has a binary output: on or off (example: air conditioning)



commercial controller with integrated
thermocouple or thermistance transducer

Two-point controller: time response

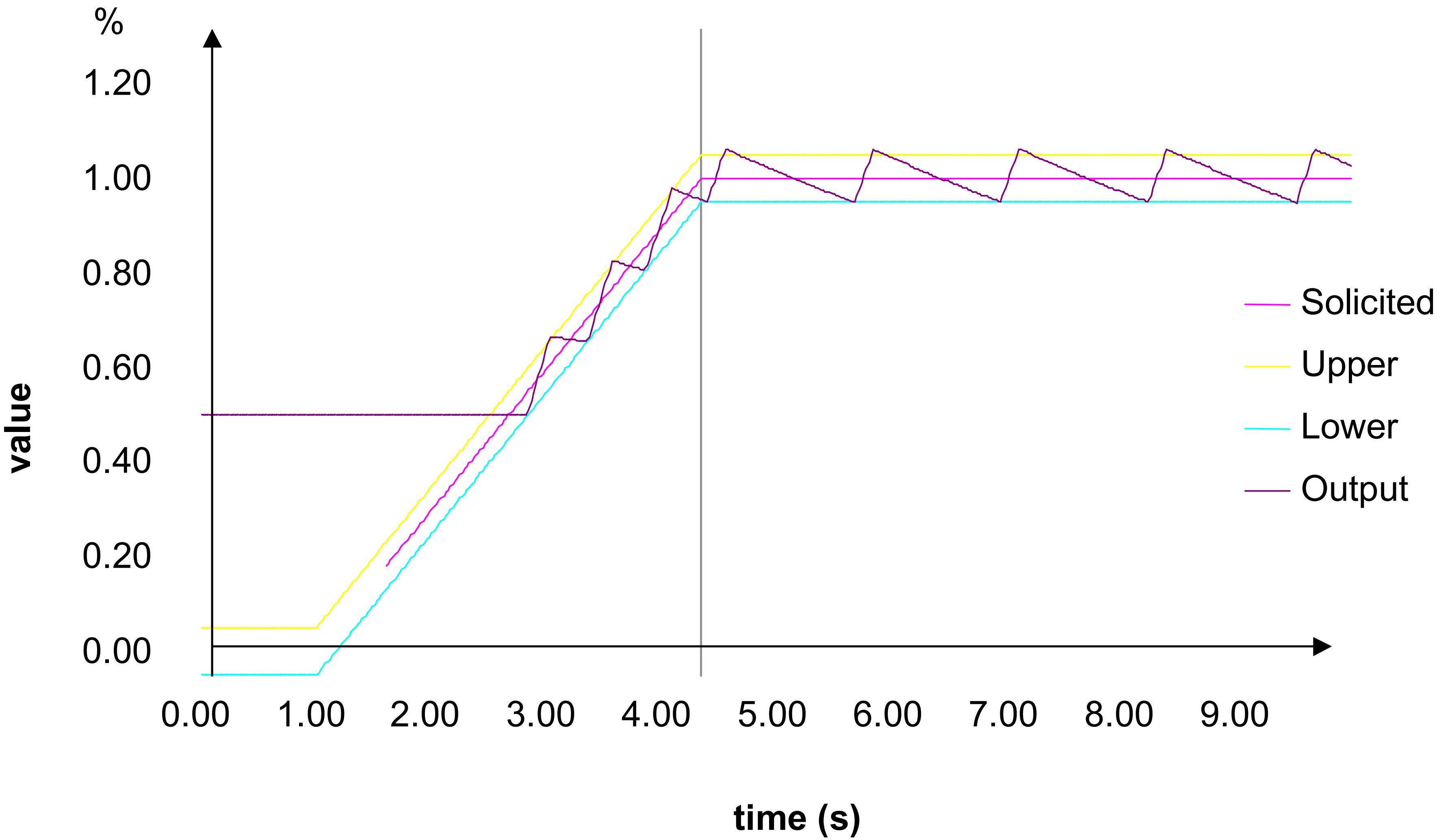


Note the different time constants for heating and cooling:
this is a non-linear system

If the process is not slow enough, <u>hysteresis</u> or switching period limit are included to limit switching frequency and avoid wearing off the contactor.

(thermal processes are normally so inertial that no hysteresis is needed)

Two-point controller: Input variable as ramp



2.2.3 PID Controller

A glance back in time...

*Tin = Sn
étain, Zinn,
stannum

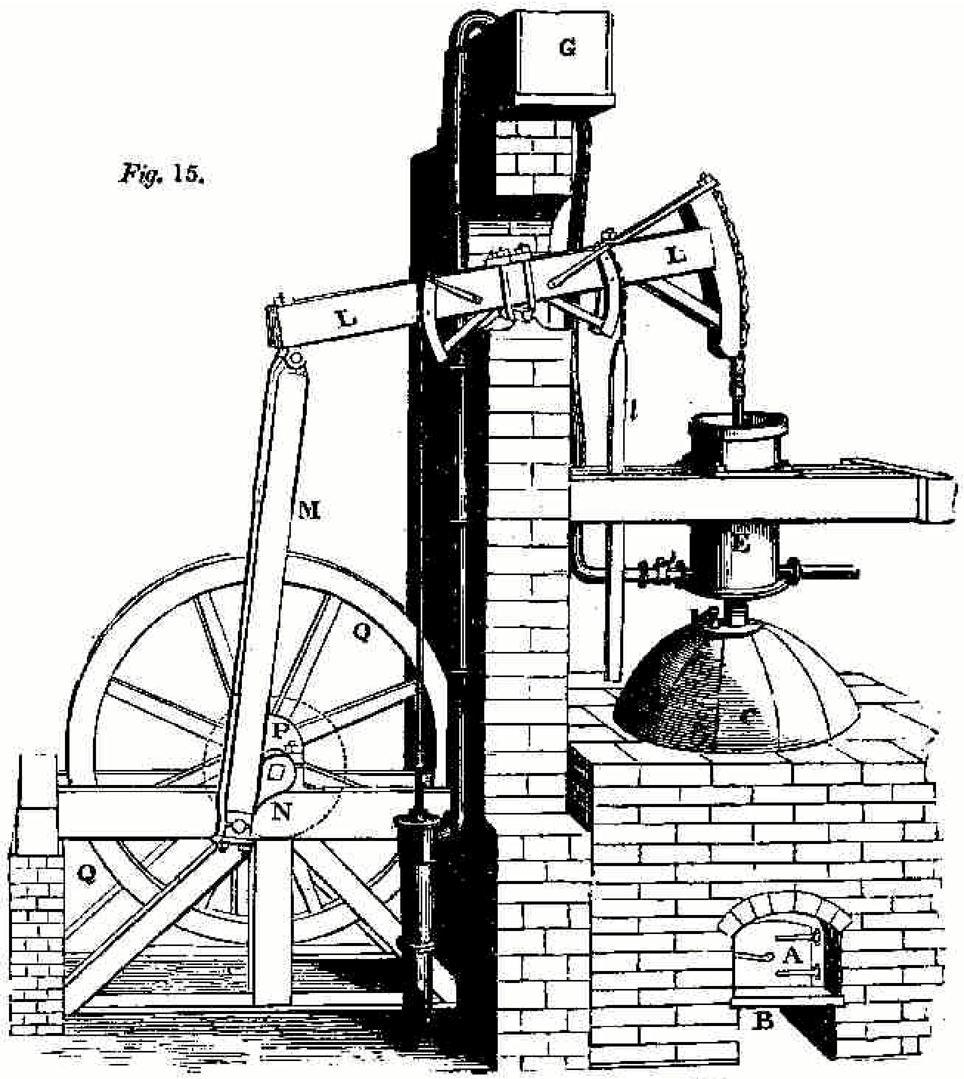
≠ Zk, Zink, zinc

ruins of a tin* mine in Cornwall (England), with the machine house for pumping, where the first steam engines were installed (1790)

Birth of the steam machine (1780 - Thomas Newcomen)

used for:

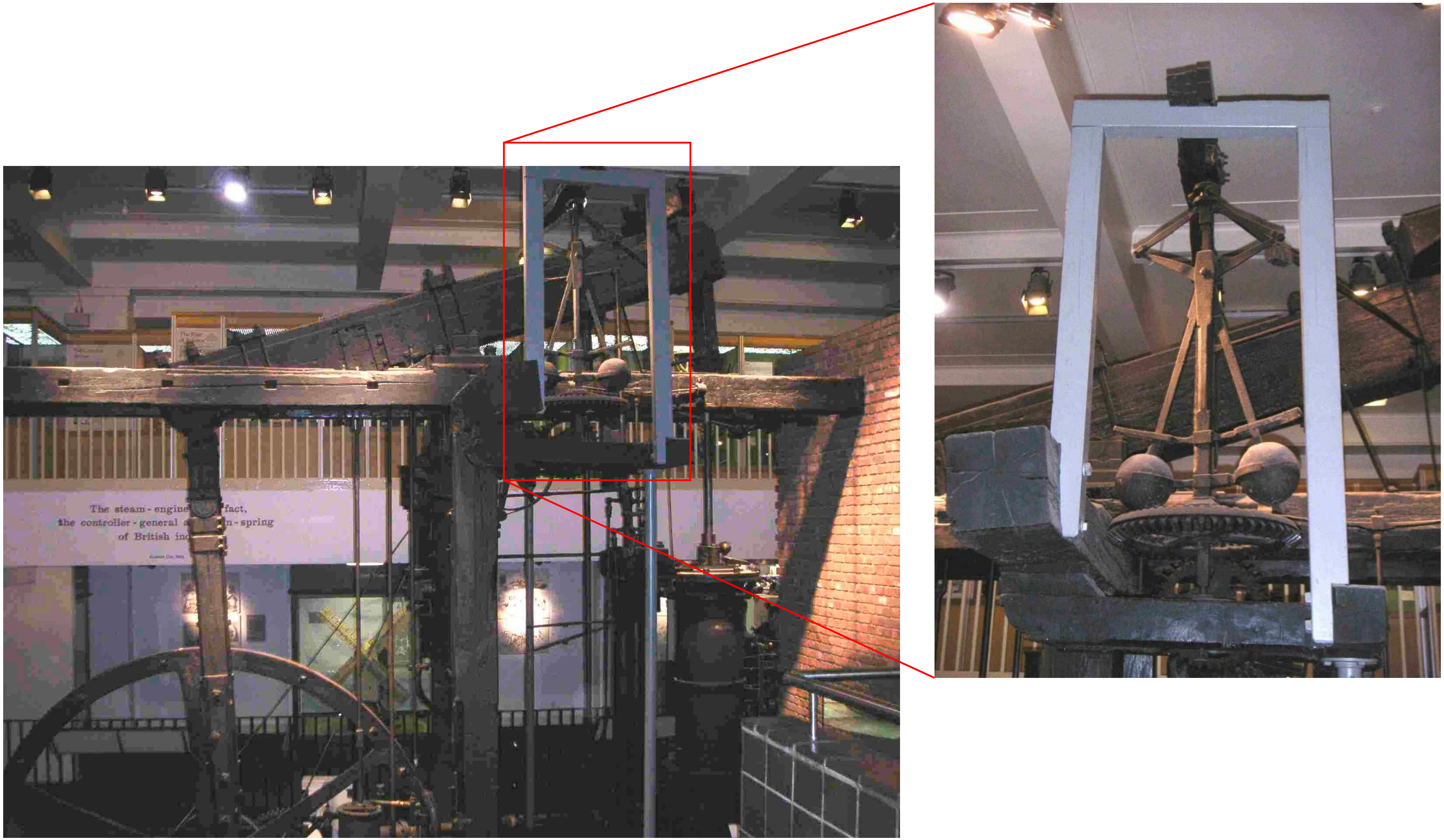
pump water
winches
ore crashing



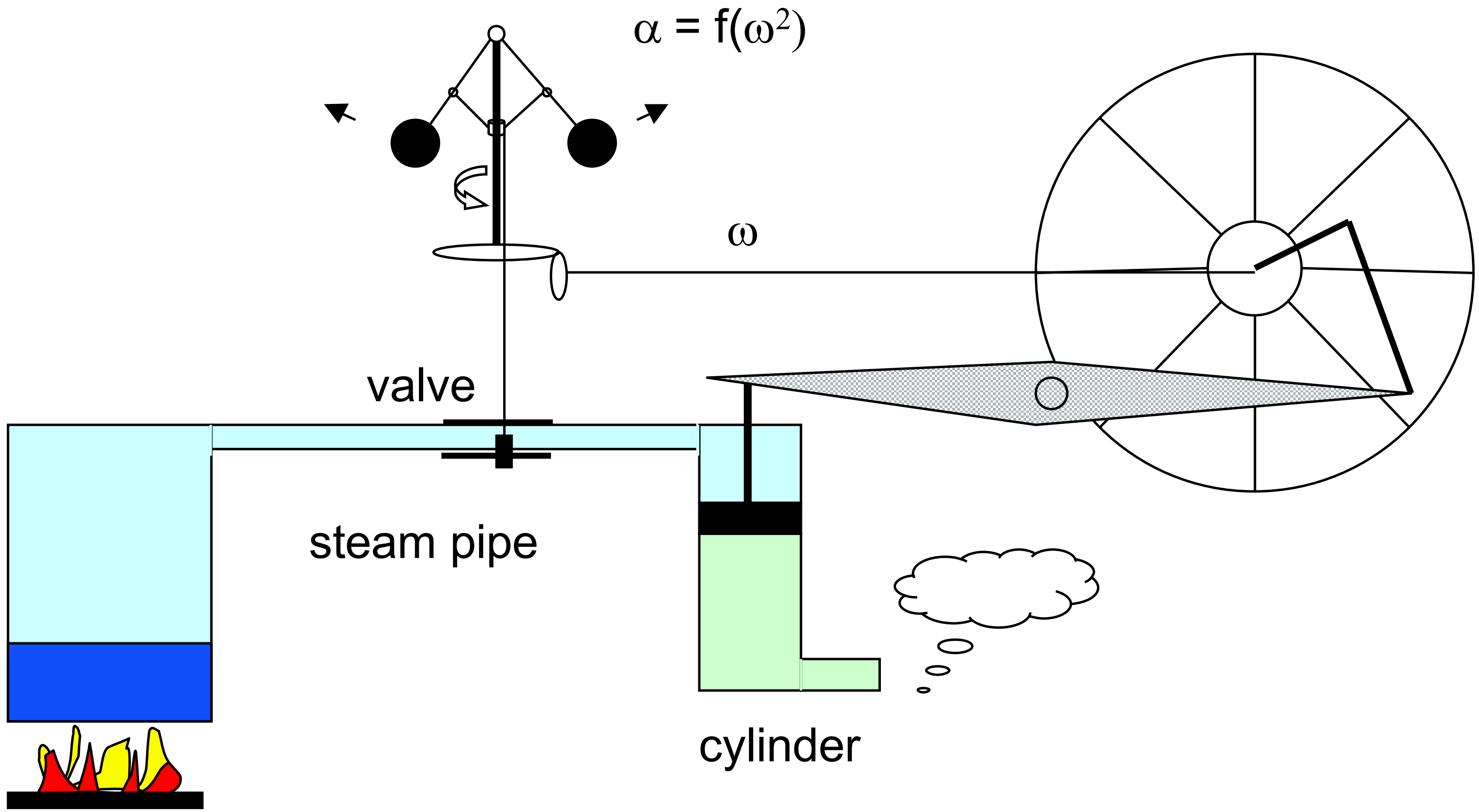
ATMOSPHERIC ENGINE WITH CRANK AND FLY-WHEEL, 1780.

Problem: keep the wheel speed constant.

The Watts "governor" (1791) - the first industrial regulator



Flywheel governor



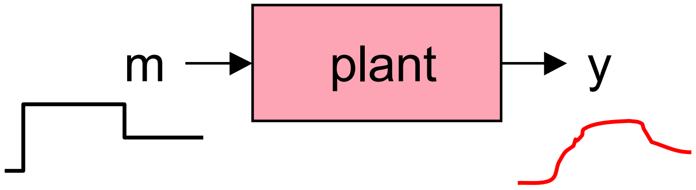
ancestor of automatic control...

Plant model for the following example

The following examples use a plant modeled by a 2_{nd} order differential equation:

$$y + y' T_1 + y'' TT_2 = m$$

differential equation



Temporal response

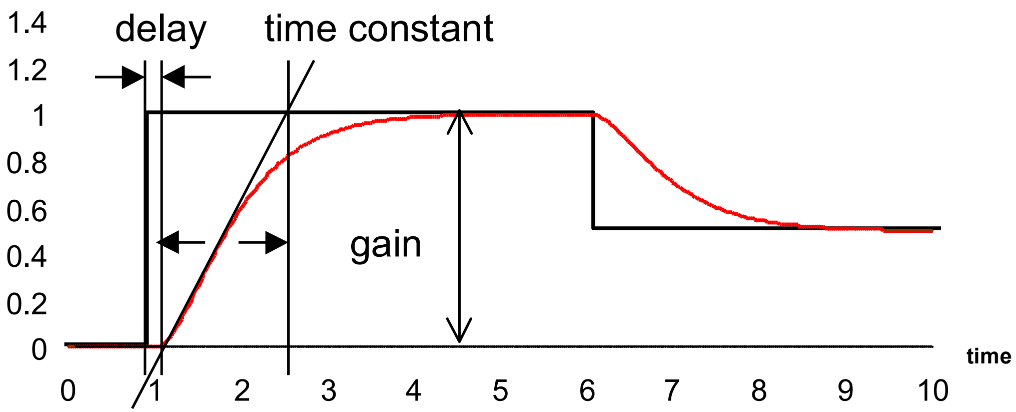
$$\frac{y}{m} = \frac{1}{1 + sT_1 + s^2 TT_2}$$

Laplace transfer function

This transfer function is typical of a plant with slow response, but without deadtime

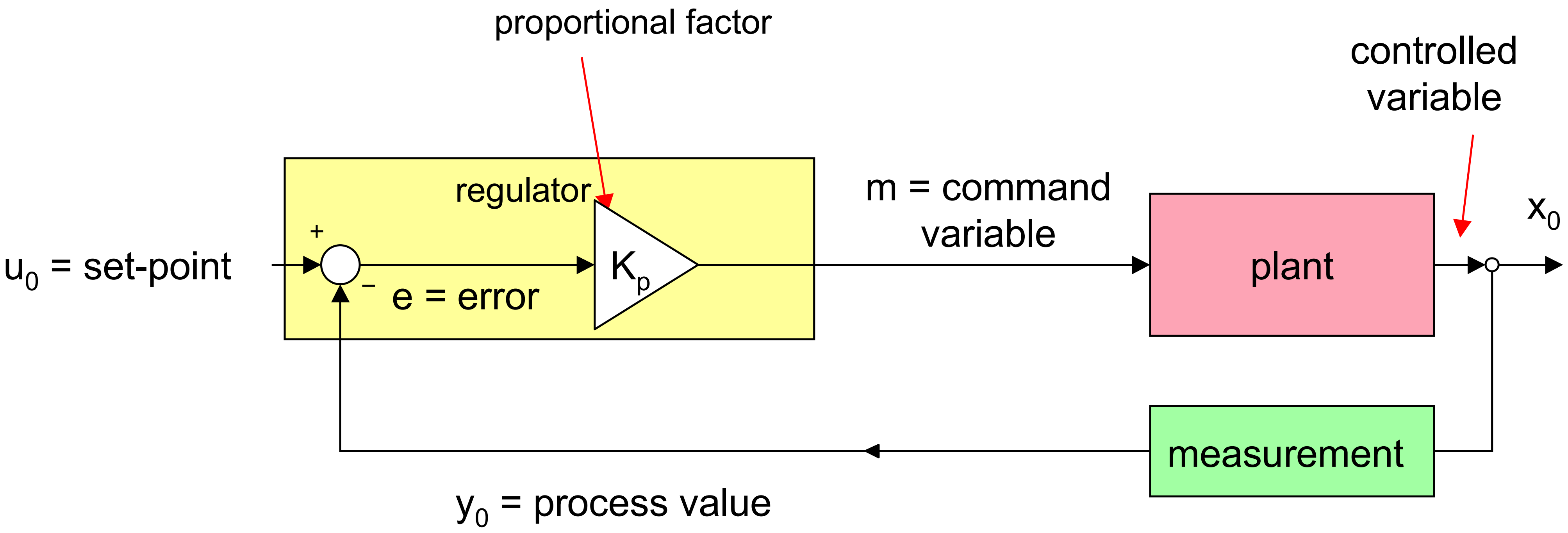
In the examples:
$T_1 = 1$ s
$TT_2 = 0.25$ s^2

step response



d ~ 0.2, T= 1.5s

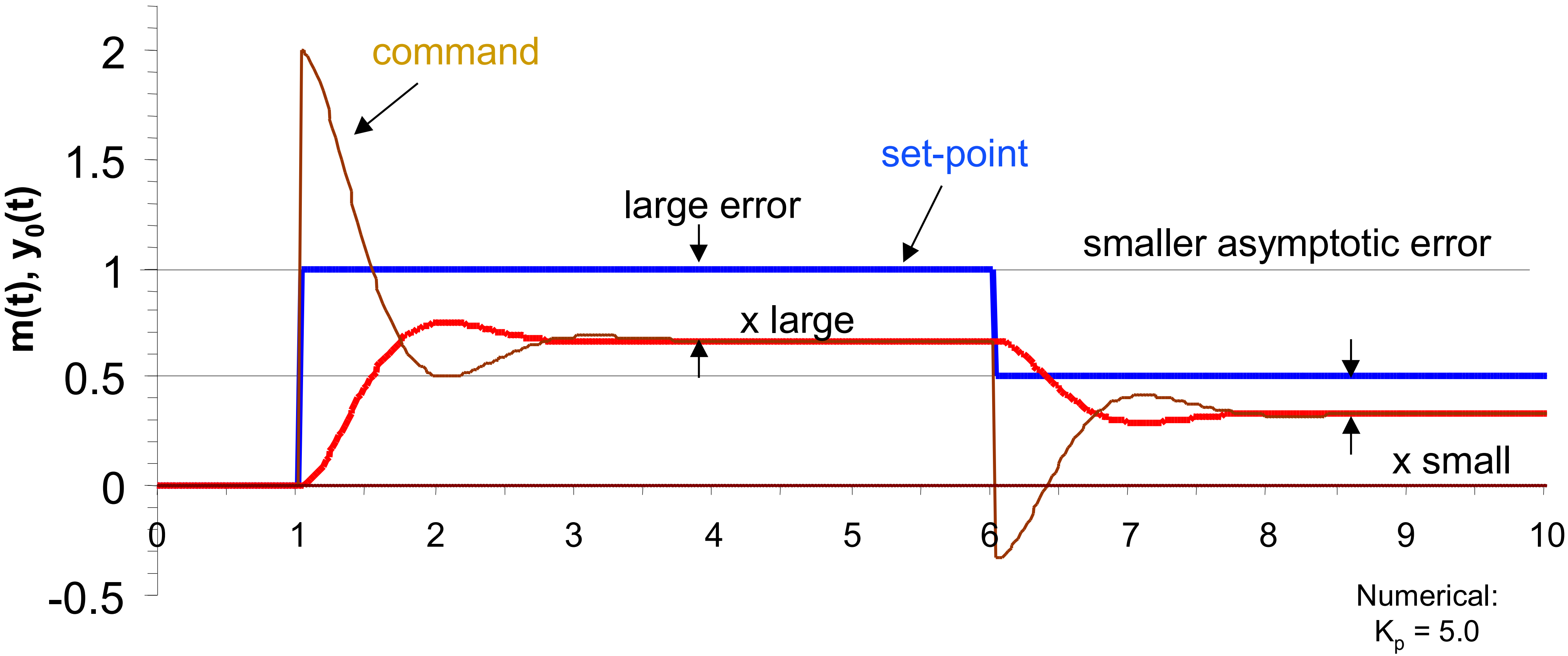
P-controller: simplest continuous regulator



the error is amplified to yield the command variable

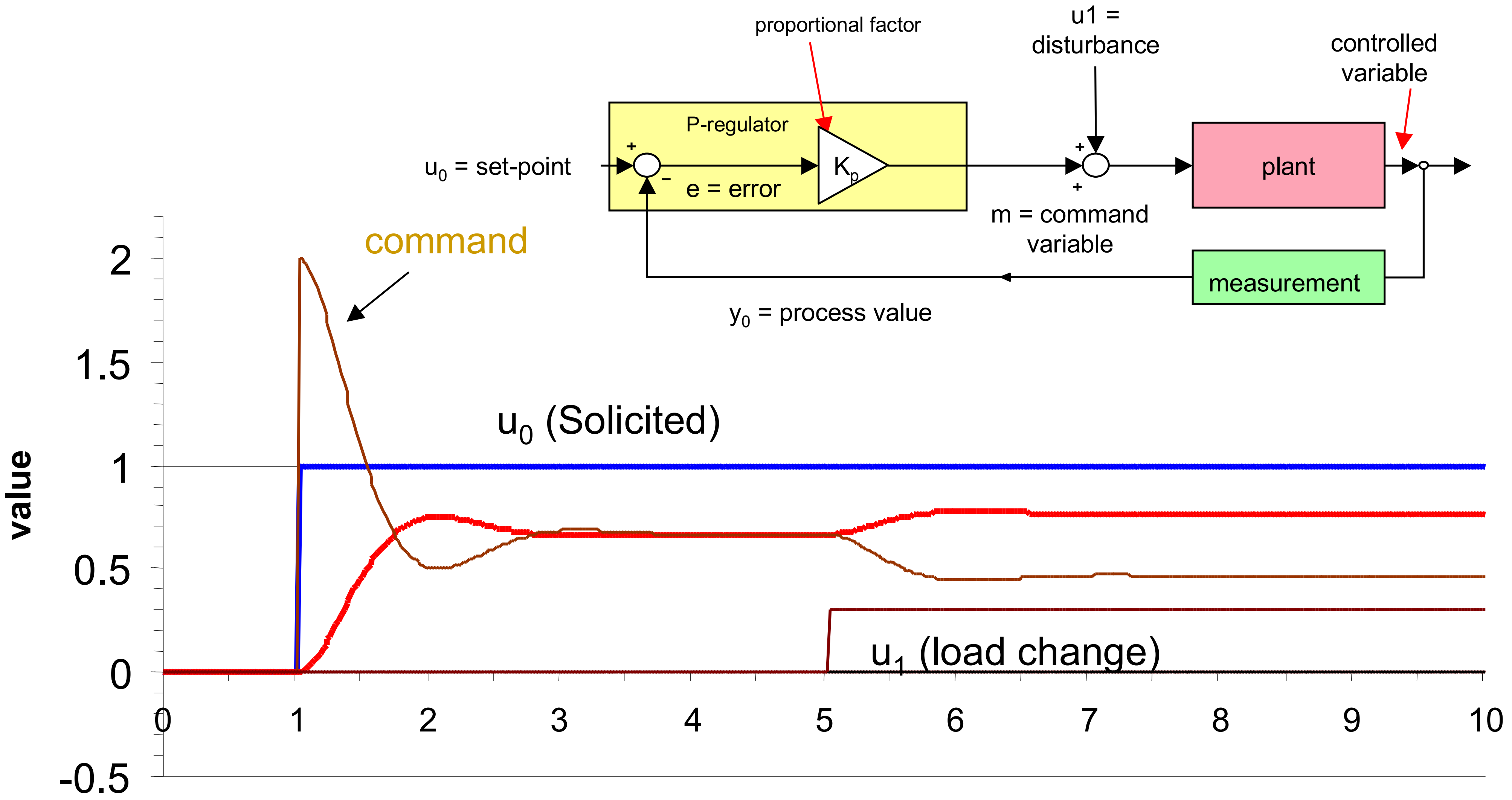
$$m = K_p \cdot e = K_p \cdot (u_0 - y_0)$$

P-Controller: Step response



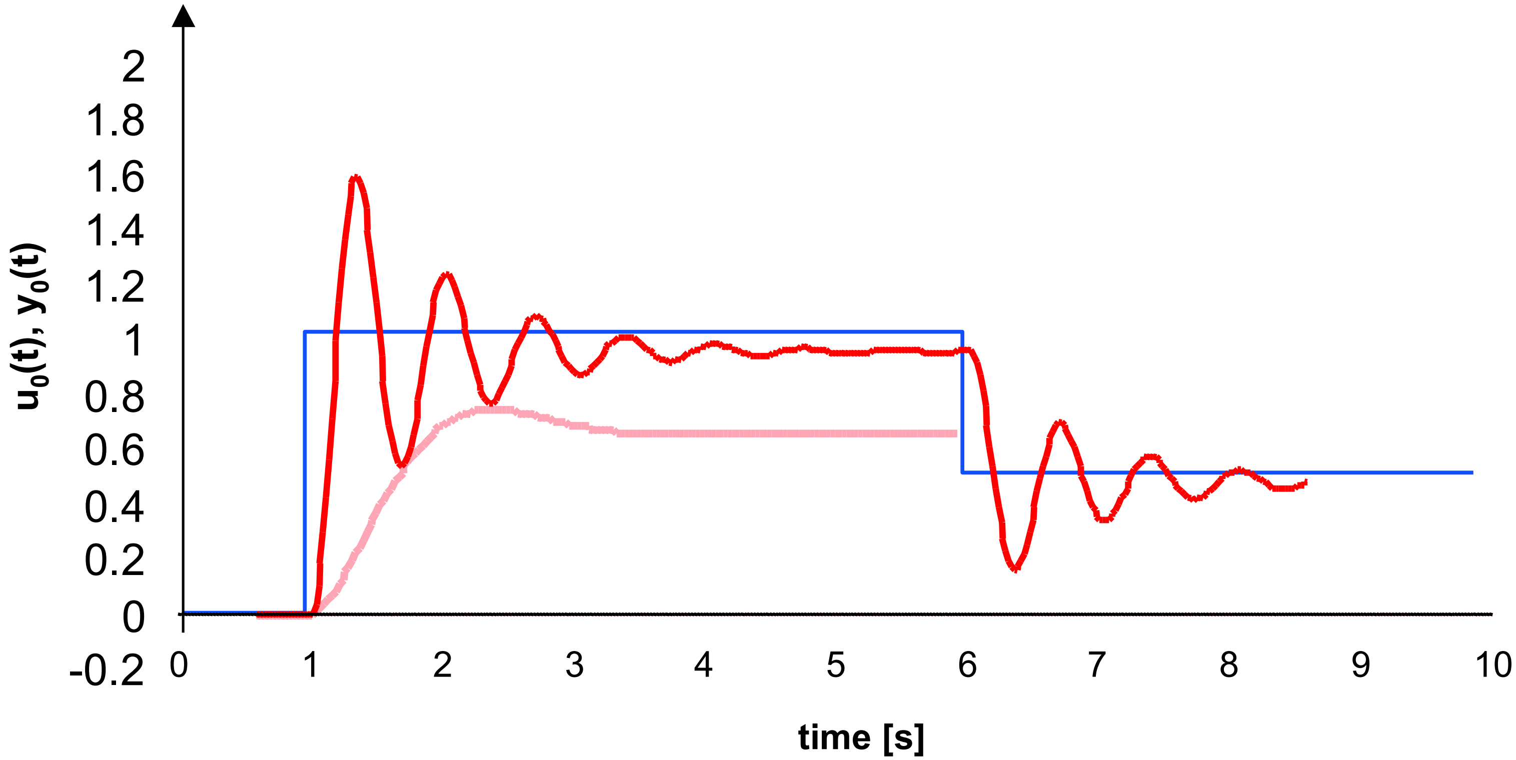
The larger the set-point, the greater the error.
The operator was used to "reset" the control

P-Controller: Load change



Not only a set-point change, but a load change causes the error to increase (or decrease).
A load change (disturbance u_1) is equivalent to a set-point change

P-Controller: Increasing the proportional factor



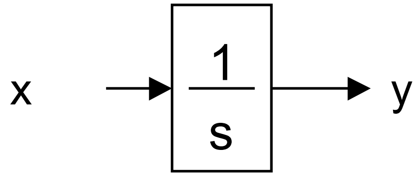
increasing the proportional factor reduces the error, but the system tends to oscillate

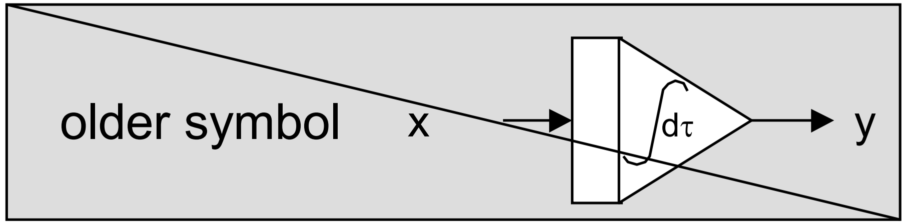
PI-Controller (Proportional Integral): introducing the integrator

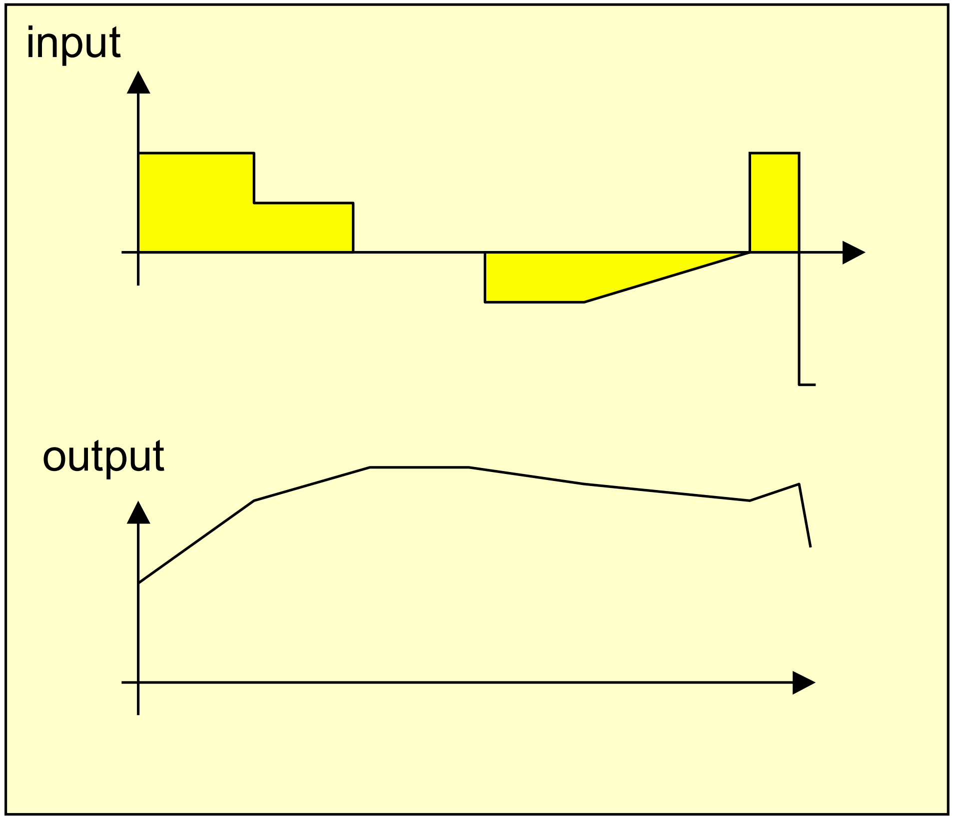
equation

$$y = \int_{t0}^{t} x(\tau)d\tau$$

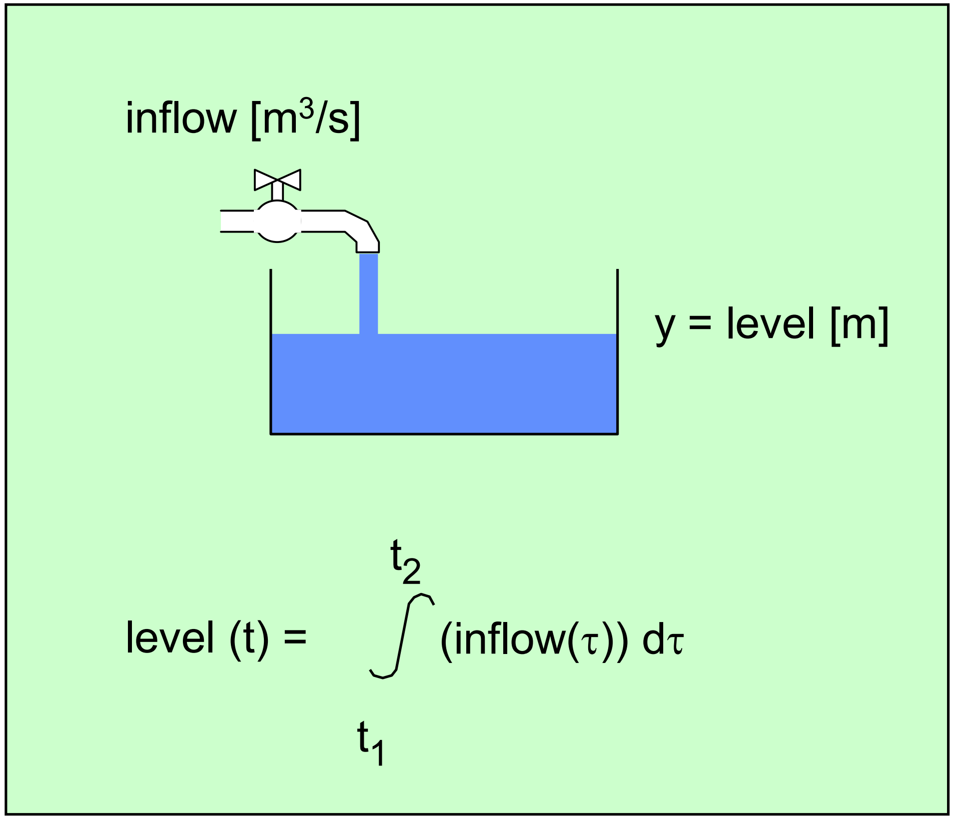
symbol





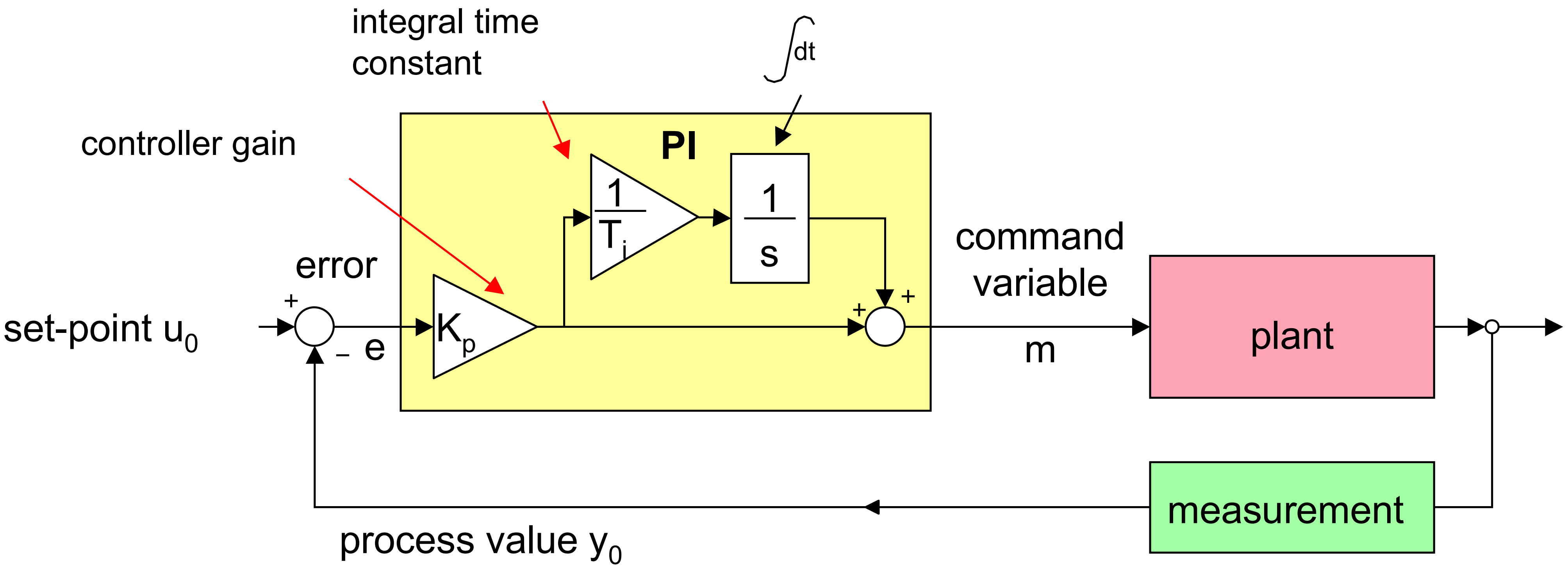


Time response of an integrator



Example of an integration process

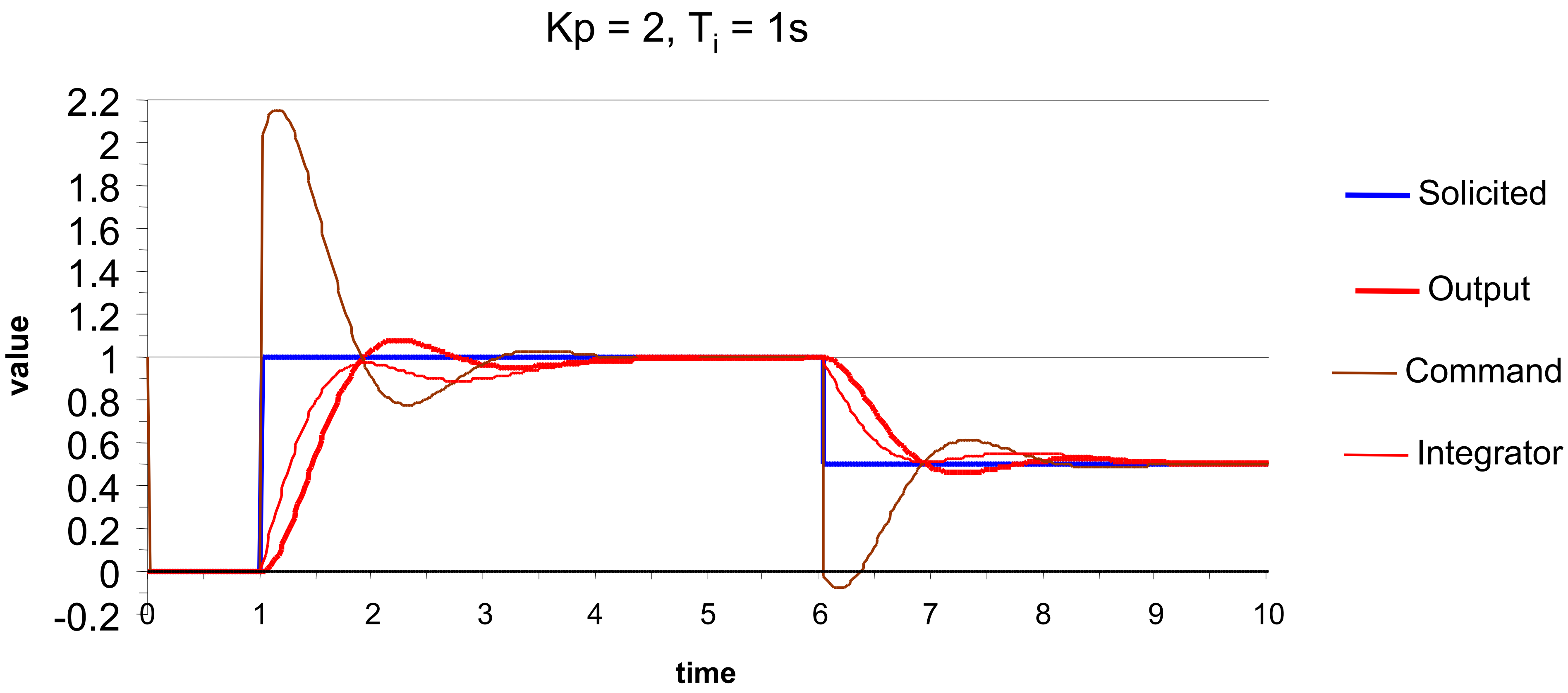
PI (Proportional-Integral) Controller



The integral factor Ki produces a non-zero control variable even when the error is zero.

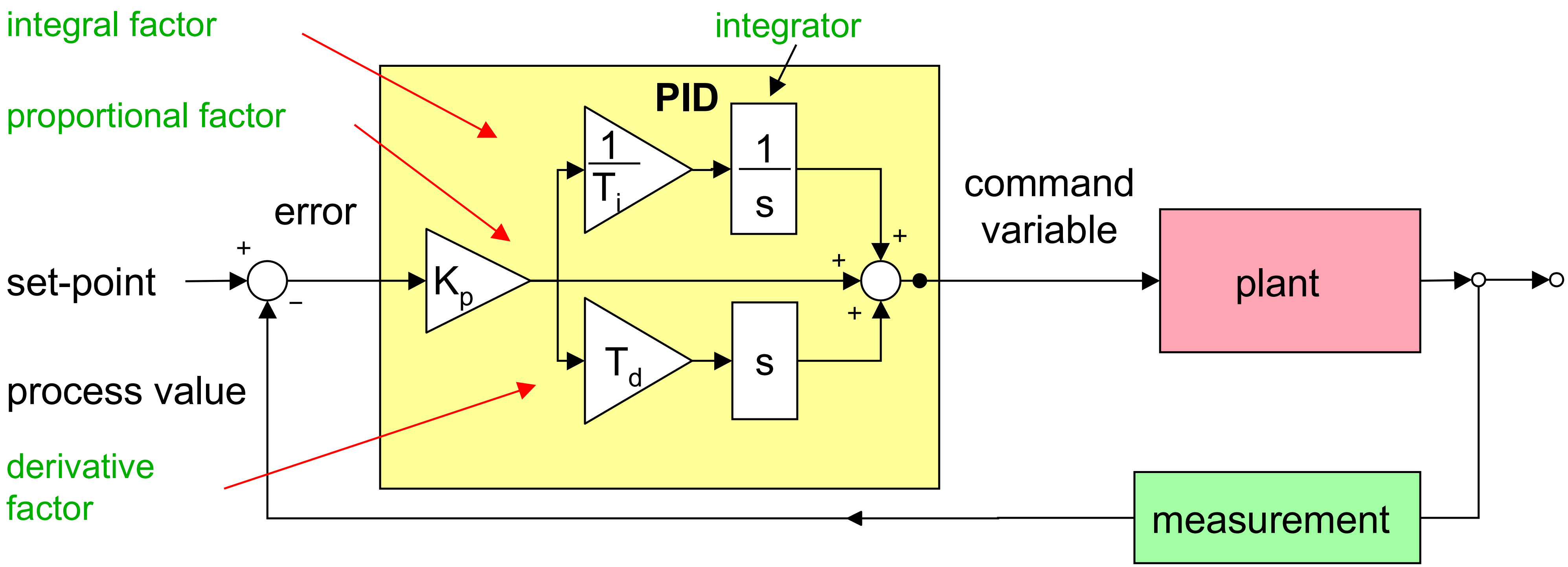
$$m = K_p \left(e(t) + \frac{1}{T_i} \int_{t0}^{t} e(\tau) d\tau \right)$$

PI-Controller: response to set-point change



The integral factor reduced the asymptotical error to zero, but slows down the response

PID-Controller (Proportional-Integral-Differential): introducing the differentiator



The proportional factor K_p generates an output proportional to the error, it requires a non-zero error to produce the command variable.

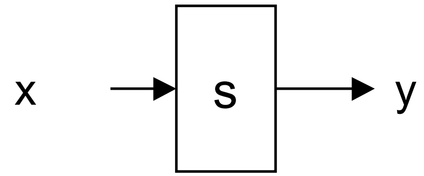
Increasing the amplification Kp decreases the error, but may lead to instability

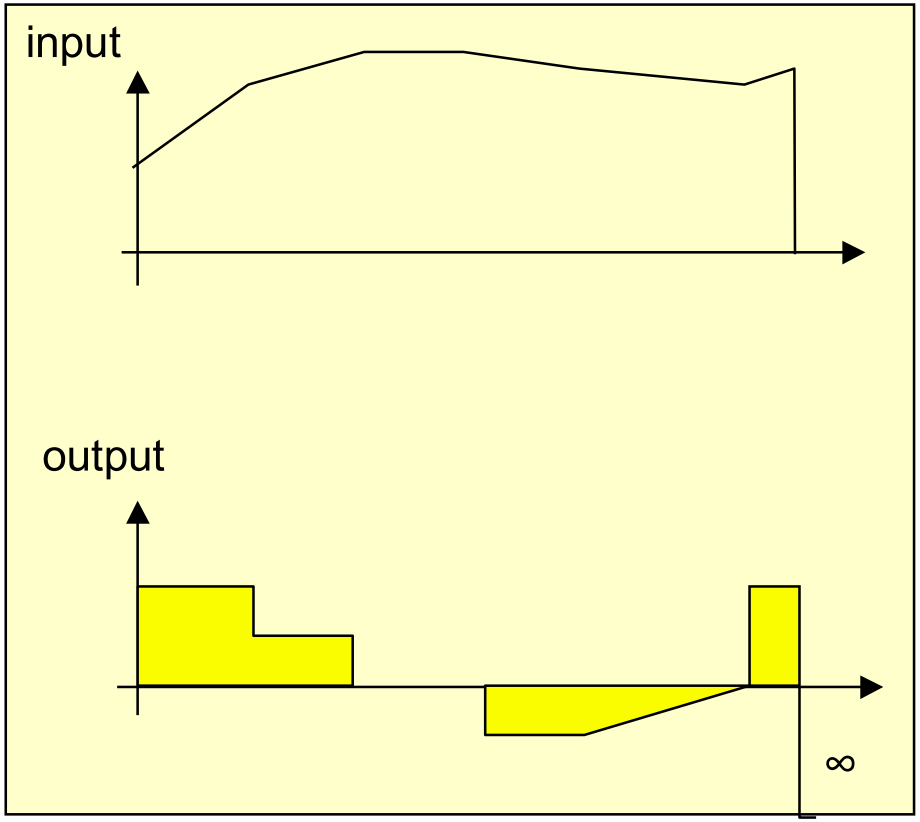
The integral factor Ki produces a non-zero control variable even when the error is zero, but makes response slower.

The derivative factor K_d speeds up response by reacting to an error step with a control variable change proportional to the step (real differentiators include filtering).

PID-Controller: Implementation of differentiator

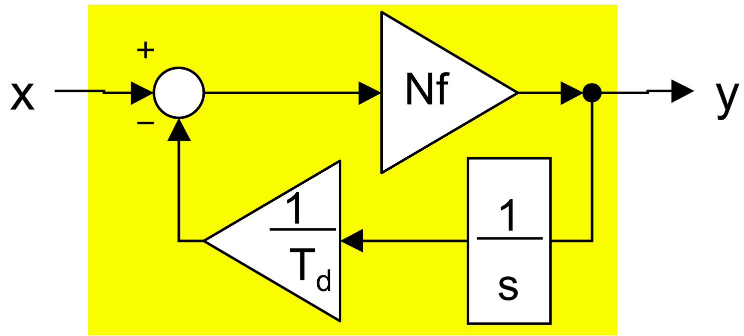
$$y = \frac{dx}{dt}$$





Time response of a differentiator

A perfect differentiator does not exist.
Differentiators increase noise.
Differentiators are approximated by integrators (filtered differentiator):



Use instead an already available variable:
e.g. the speed for a position control

PID controller: Equations

time domain

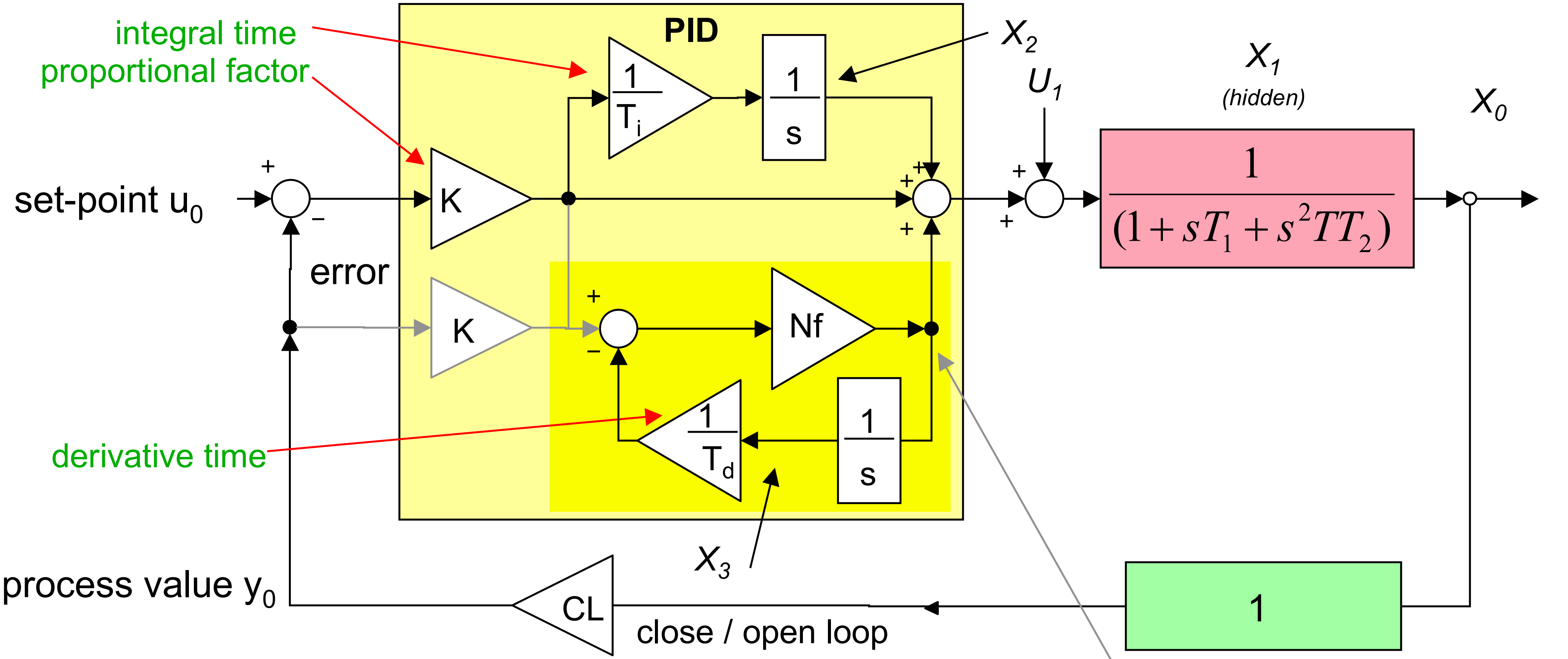
$$m = K_p \left(e(t) + \frac{1}{T_i} \int_{t0}^{t} e(\tau) d\tau) + T_d \frac{de(t)}{dt} \right)$$

Laplace domain

$$F(p) = K_p \left(1 + \frac{1}{sT_i} + \frac{T_d s}{(1 + \frac{Nf}{T_d} s)} \right)$$

Real differentiators include a filtering

PID and Plant Simulation (Excel sheet)



$$\frac{dx_0}{dt} = x_1$$

$$\frac{dx_2}{dt} = \frac{K}{T_i}(u_0 - y_0)$$

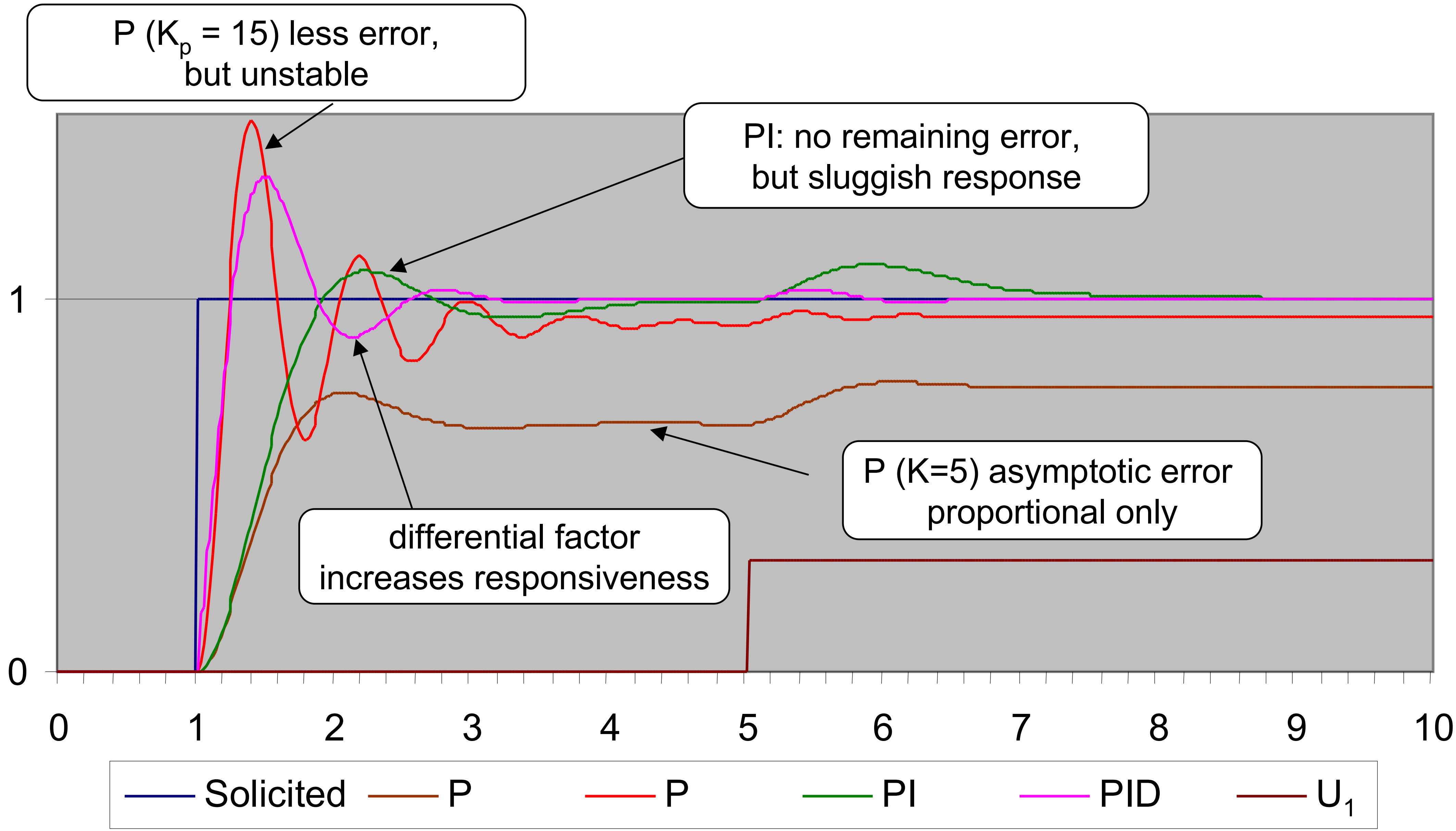
$$\frac{dx_1}{dt} = \frac{1}{TT_2}\left(x_2 + D_f + K(u_0 - x_0) - T_1 x_1 - x_0 + u_1\right)$$

$$\frac{dx_3}{dt} = Nf\left(-\frac{1}{T_d}x_3 + K(u_0 - y_0)\right)$$

$$D_f = Nf\left(-\frac{1}{T_d}x_3 + K(u_0 - x_0)\right)$$

filtered derivative

PID response summary

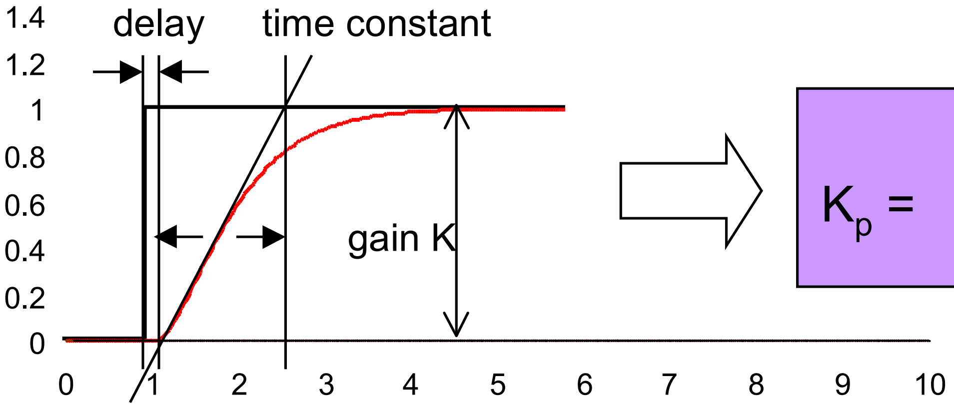


PID-Controller: influence of parameters

increasing	Rise time	Overshoot	Settling time	Steady-State Error
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	Small Change

Empirical formula of Nichols (1942 !)

step response (open loop)



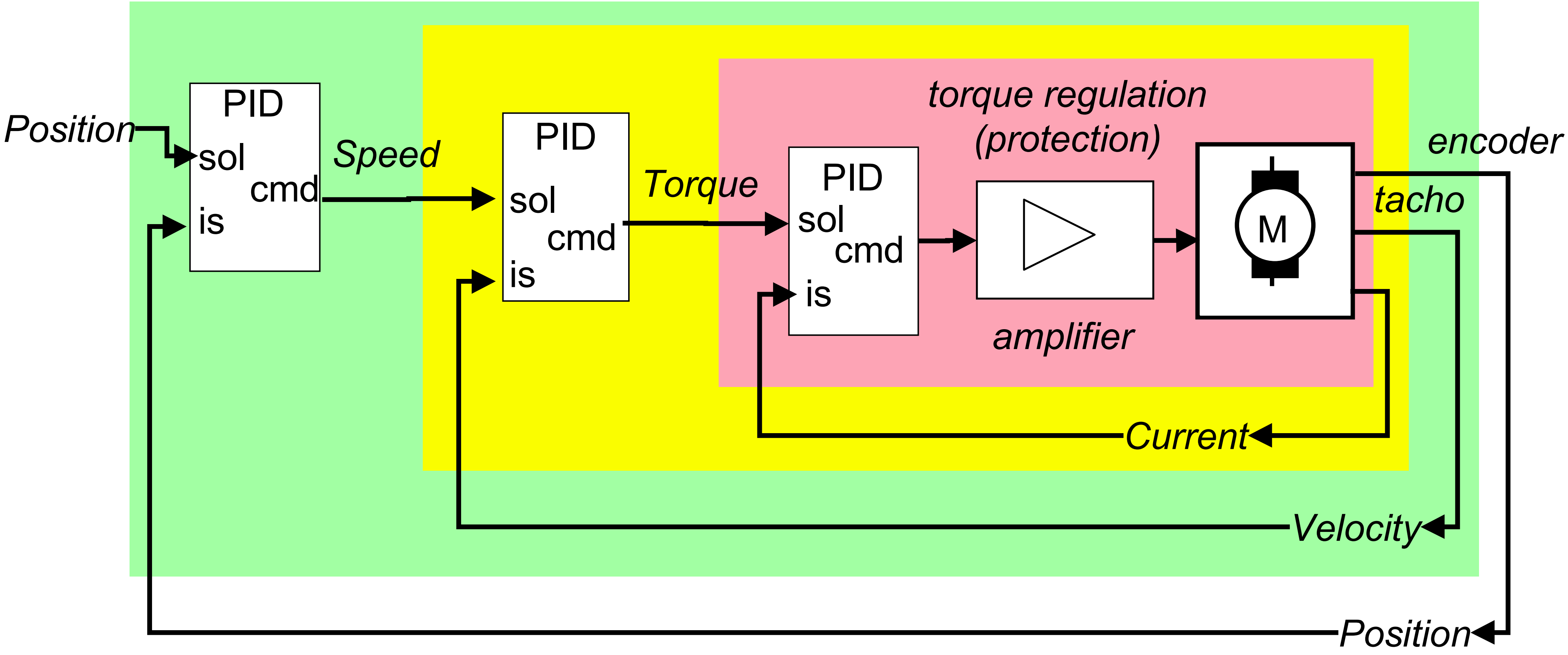
d ~ 0.2, T= 1.5s

$$K_p = \frac{1.2\,T}{K\,d} \qquad T_i = 2.0\,d \qquad T_d = 0.5\,d \qquad (N_f = 10)$$

2.2.4 Nested controllers

Nested control of a continuous plant - example

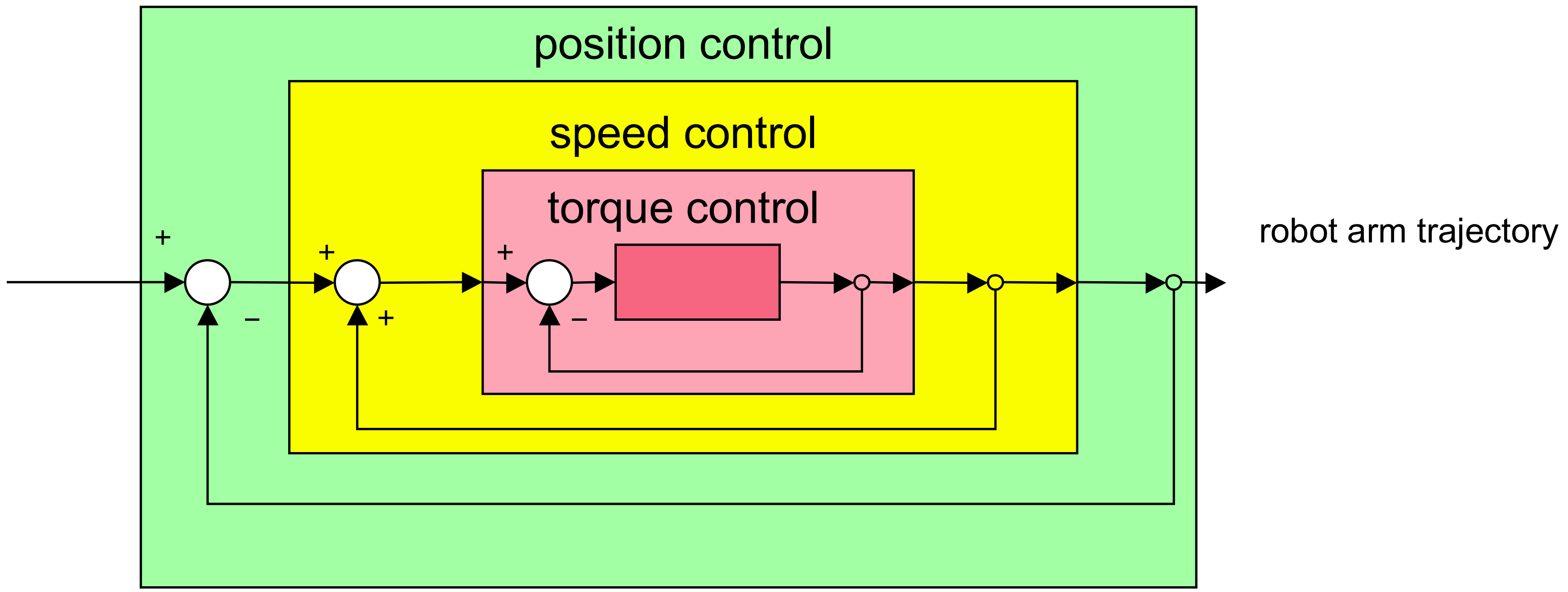
Example: position control of a rotating shaft



Nesting regulators allow to maintain the output variable at a determined value while not exceeding the current or speed limitations

Nested loops and time response

A control system consists often of nested loops, with the fastest loop at the inner level



Assessment

How does a two-point regulator works ?

How is the a wear-out of the contacts prevented ?

How does a PID regulator works ?

What is the influence of the different parameters of a PID ?

Is a PID controller required for a position control system (motor moves a vehicle)

Explain the relation between nesting control loops and their real-time response

To probe further

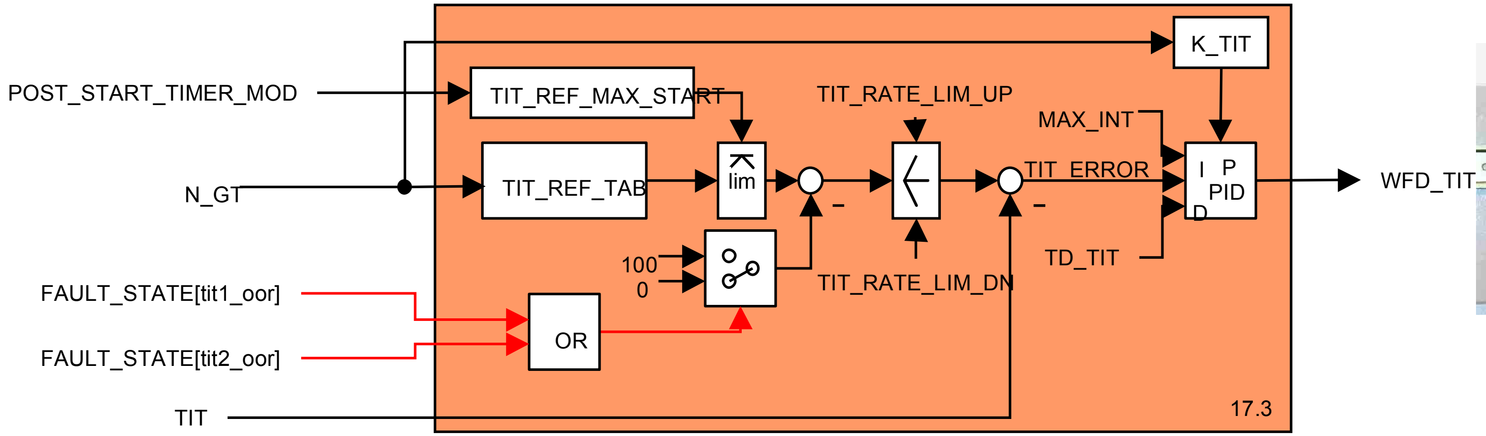
"Computer Systems for Automation and Control", Gustaf Olsson, Gianguido Piani, Lund Institute of Technology

“Modern Control Systems”, R. Dorf, Addison Wesley

Courses of Prof. Roland Longchamp and Dominique Bonvin

EPFL

Industrial Automation
Automation Industrielle
Industrielle Automation



2.3 Programmable Logic Controllers

Automates Programmables

Speicherprogrammierbare Steuerungen

Prof. Dr. H. Kirrmann

ABB Research Center, Baden, Switzerland

2005 March, HK

2.3.1 PLCs: Definition and Market

Programmable Logic Controller: Definition

AP = Automates Programmables industriels
SPS = Speicherprogrammierbare Steuerungen

Definition: “small computers, dedicated to automation tasks in an industrial environment"

Formerly: cabled relay control (hence 'logic'), analog (pneumatic, hydraulic) governors

Today: specialized computer performing control and regulation

Function: Measure, Command, Control

Distinguish **Instrumentation**

flow meter, temperature, position,…. but also actors (pump, …)

Control

programmable logic controllers with digital peripherals & field bus

Visualization

HMI in PLCs (when it exists) is limited to control of operator displays

PLC: functions

(*Messen, Steuern, Regeln = MSR*)

- Measure
- Command
- Regulation
- Protection

•Event Logging
•Communication
•Human interface

PLC: Characteristics

• large number of peripherals: 20..100 I/O per CPU, high density of wiring, easy assembly.

• binary and analog Input/Output with standard levels

• located near the plant (field level), require robust construction, protection against dirt, water and mechanical threats, electro-magnetic noise, vibration, extreme temperature range (-30C..85C)

• programming: either very primitive with hand-help terminals on the target machine itself, or with a lap-top able to down-load programs.

• network connection is becoming common, allowing programming on workstations.

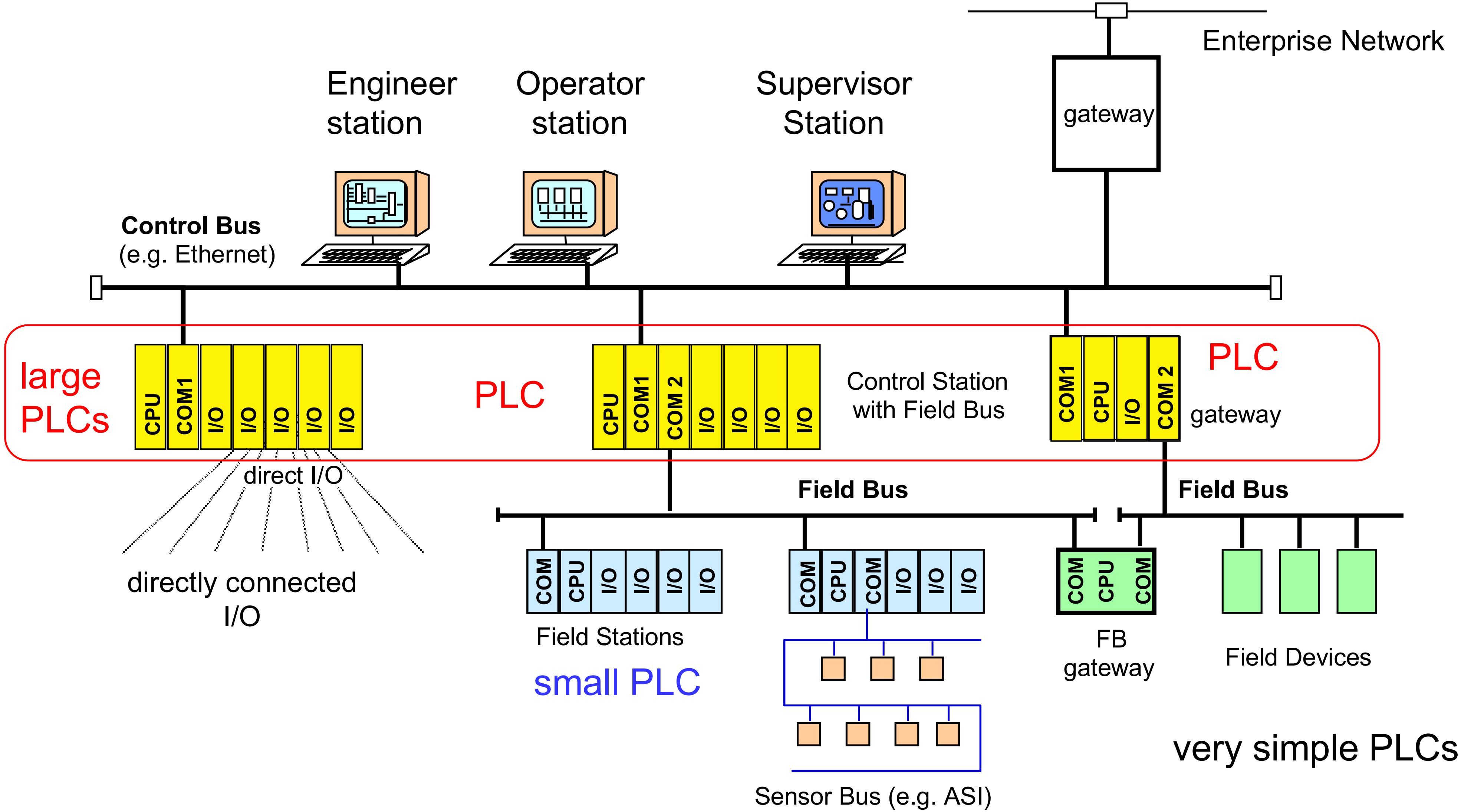
• field bus connection for remote I/Os

• primitive Man-Machine interface, either through LCD-display or connection of a laptop over serial lines (RS232).

• economical - €1000.- .. €15'000.- for a full crate.

• the value is in the application software (licenses €20'000 ..€50'000)

PLC: Location in the control architecture



PLC: manufacturers

Switzerland

SAIA, Weidmüller

Europe:

Siemens (60% market share) [Simatic],
ABB (includes Hartmann&Braun, Elsag-Bailey, SattControl,…) [Industrial^{IT}],
Groupe Schneider [Télémécanique],
WAGO,
Phoenix Contact ...

World Market:

GE-Fanuc,
Honeywell,
Invensys (Foxboro)
Rockwell, (Allen-Bradley,…)
Emerson (Fisher Control, Rosemount, Westinghouse)
Hitachi, Toshiba, Fujitsu, Yokogawa
…

large number of bidders, fusions and acquisitions in the last years.
Distinguish PLCs for the open market (OEM) and proprietary PLCs

2.3.3 PLCs: Kinds

Kinds of PLC

(1) Compact

Monolithic construction
Monoprocessor
Fieldbus connection

Fixed casing

Fixed number of I/O (most of them binary)

No process computer capabilities (no MMC)

Typical product: Mitsubishi MELSEC F, ABB AC31, SIMATIC S7

(2) Modular PLC

Modular construction (backplane)
One- or multiprocessor system
Fieldbus and LAN connection

3U or 6U rack, sometimes DIN-rail

Large variety of input/output boards

Connection to serial bus

Small MMC function possible

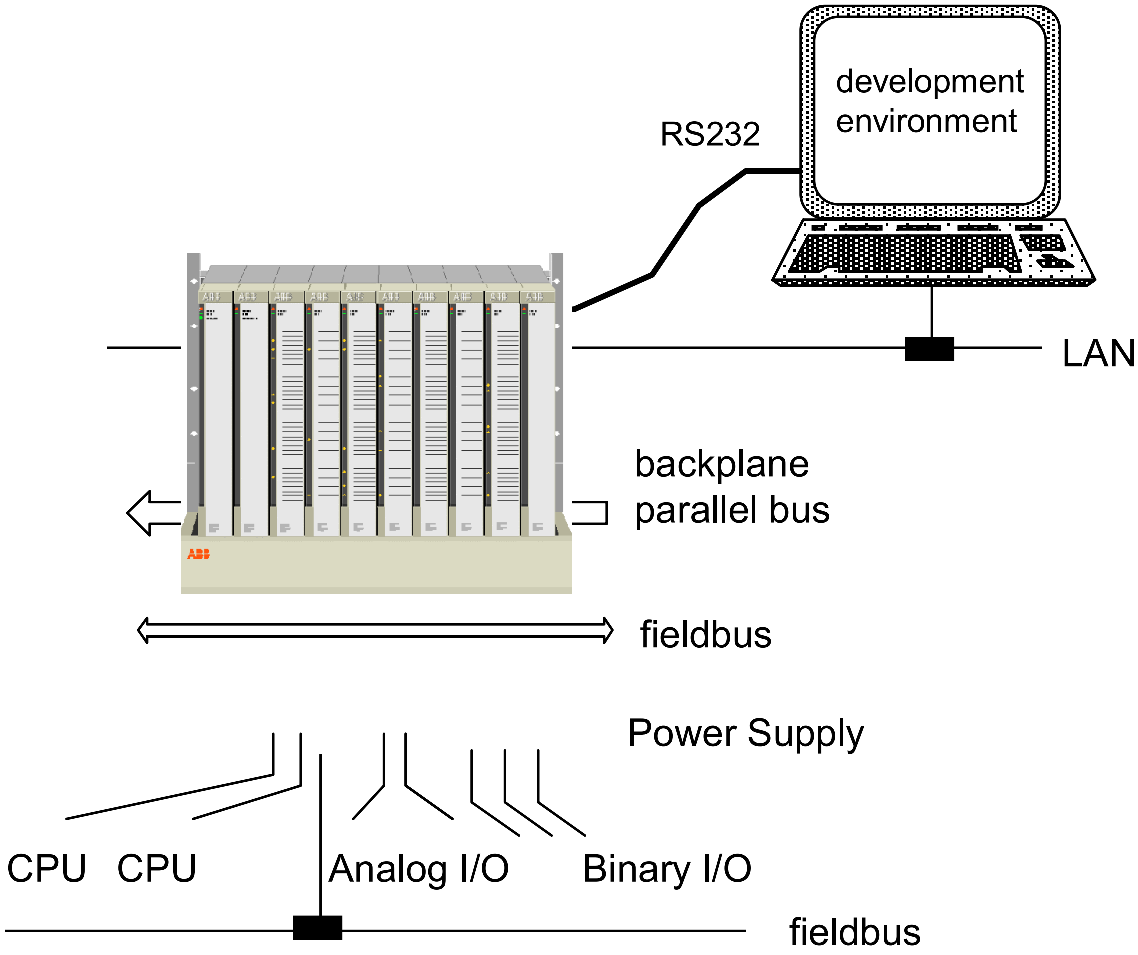
Typical products: SIMATIC S5-115, Hitachi H-Serie, ABB AC110

(3) Soft-PLC

Windows NT or CE-based automation products
Direct use of CPU or co-processors

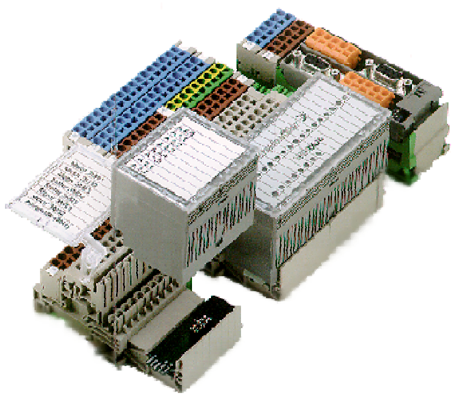
Modular PLC

- tailored to the needs of an application
- housed in a 19" (42 cm) rack
  (height 6U (= 233 mm) or 3U (=100mm)
- high processing power (several CPU)
- large choice of I/O boards
- concentration of a large number of I/O
- interface boards to field busses
- requires marshalling of signals
- primitive or no HMI
- cost effective if the rack can be filled
- supply 115-230V~ , 24V= or 48V= (redundant)
- cost ~ €10’000 for a filled crate



Small modular PLC

courtesy ABB

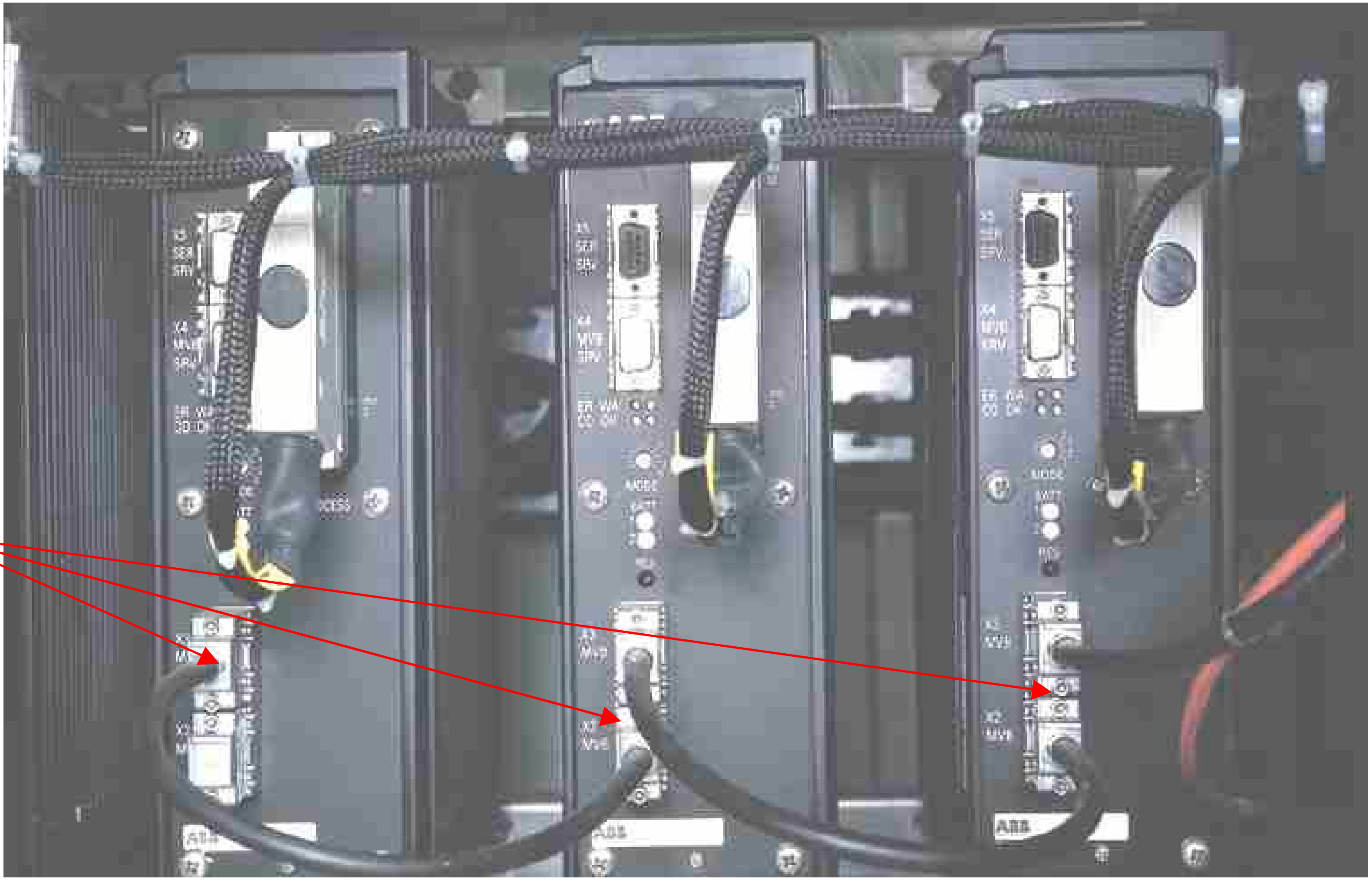


courtesy Backmann

mounted on DIN-rail, 24V supply
cheaper (€5000)
not water-proof,
no ventilator
extensible by a parallel bus (flat cable or rail)

Specific controller (railways)

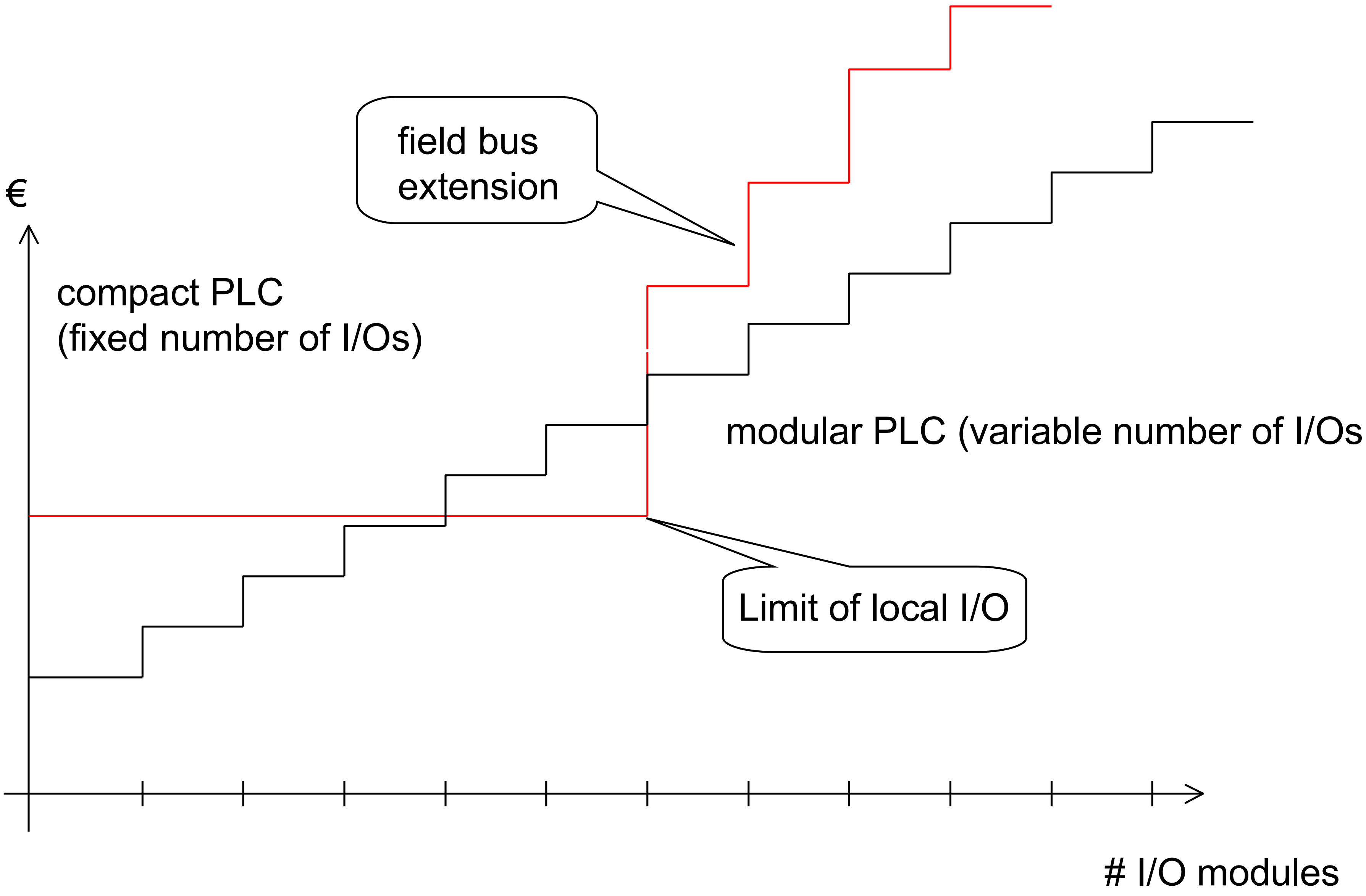
data bus



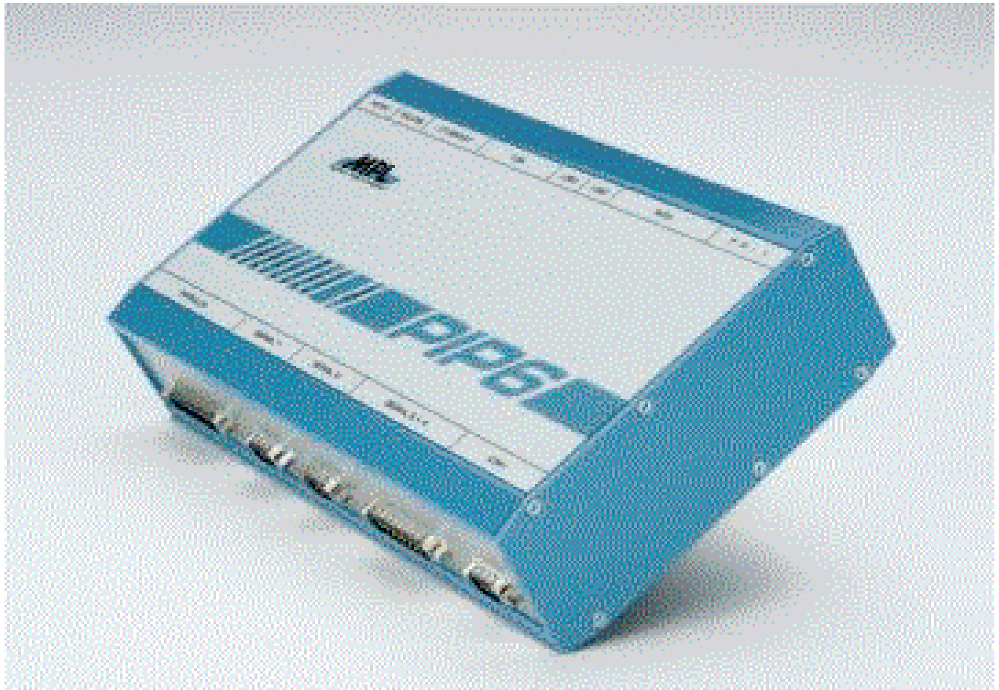
three PLCs networked by a data bus.

special construction: no fans, large temperature range, vibrations

Compact or modular ?



Industry- PC

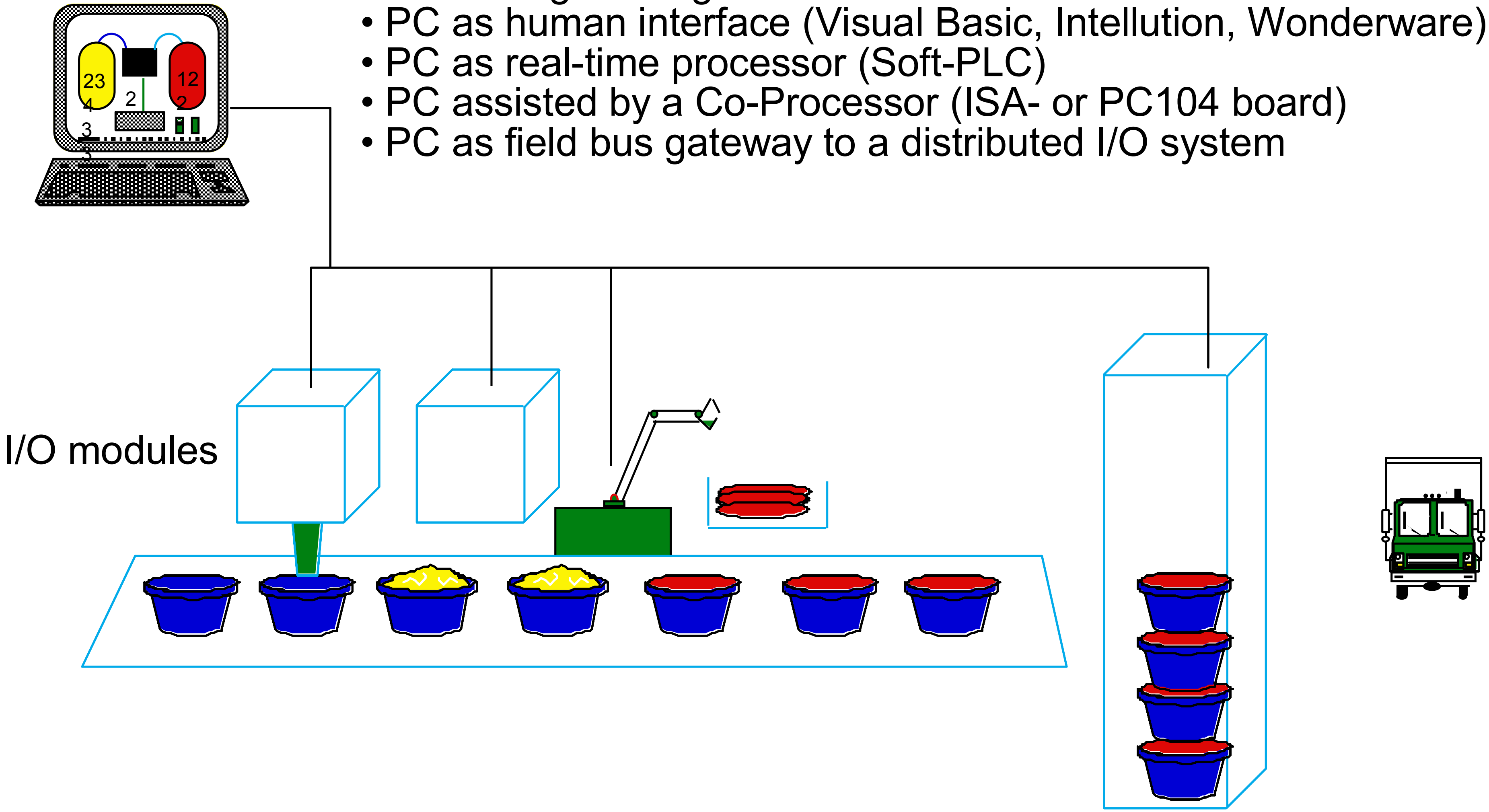


Wintel architecture
(but also: Motorola, PowerPC),
MMI offered (LCD..)
Limited modularity through mezzanine boards
(PC104, PC-Cards, IndustryPack)
Backplane-mounted versions with PCI or Compact-PCI

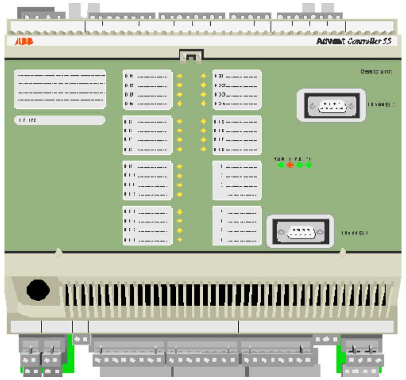
Competes with modular PLC
no local I/O,
fieldbus connection instead,
costs: € 2000.-

Soft-PLC (PC as PLC)

- PC as engineering workstation
- PC as human interface (Visual Basic, Intellution, Wonderware)
- PC as real-time processor (Soft-PLC)
- PC assisted by a Co-Processor (ISA- or PC104 board)
- PC as field bus gateway to a distributed I/O system

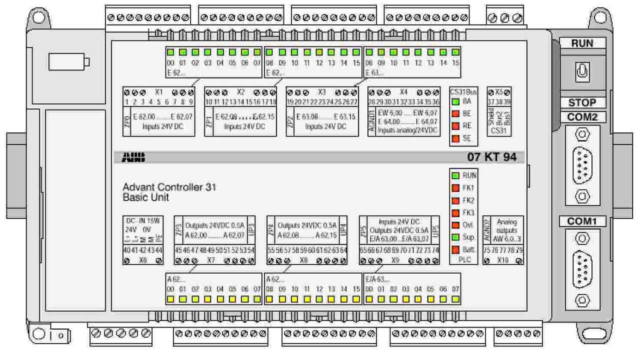


Compact PLC



courtesy ABB

courtesy ABB



courtesy ABB

Monolithic (one-piece) construction
Fixed casing
Fixed number of I/O (most of them binary)
No process computer capabilities (no MMC)
Can be extended and networked by an extension (field) bus
Sometimes LAN connection (Ethernet, Arcnet)
Monoprocessor

Typical product: Mitsubishi MELSEC F, ABB AC31, SIMATIC S7

costs: € 2000

Specific Controller (example: Turbine)

tailored for a specific application, produced in large series

Programming port

Relays and fuses

Thermocouple

inputs

binary I/Os,

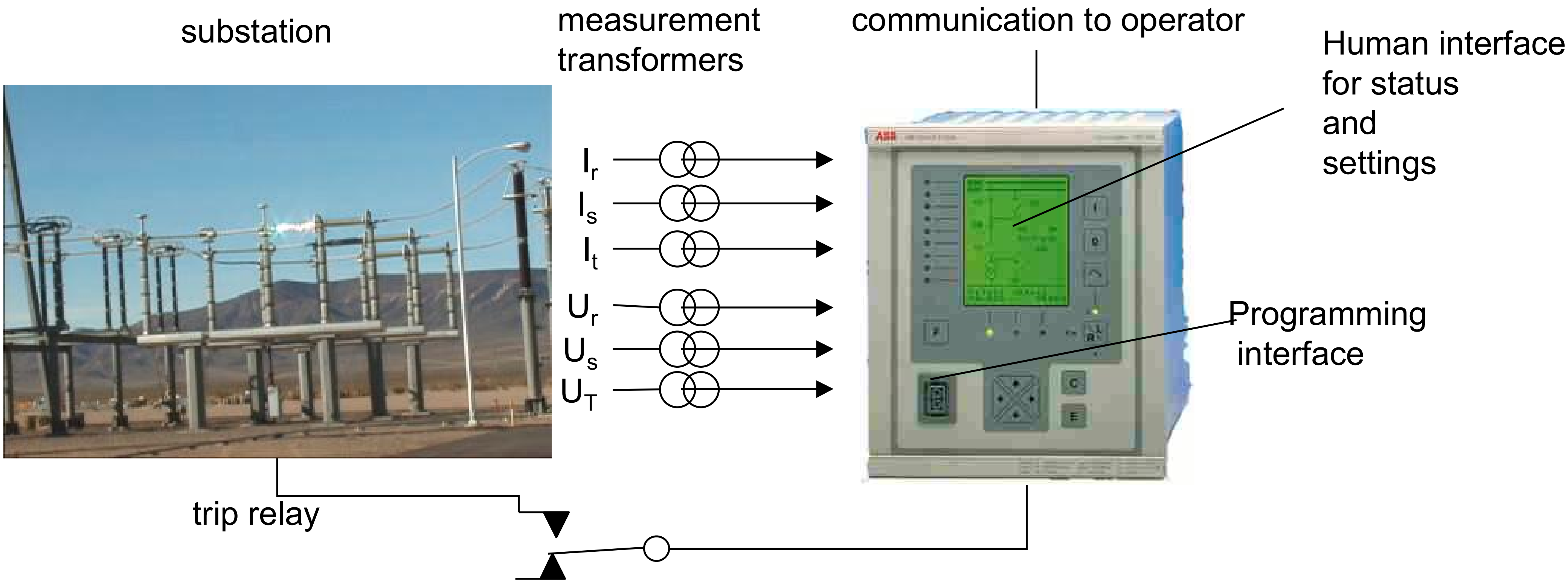
CAN field bus

RS232 to HMI

courtesy Turbec

cost: € 1000.-

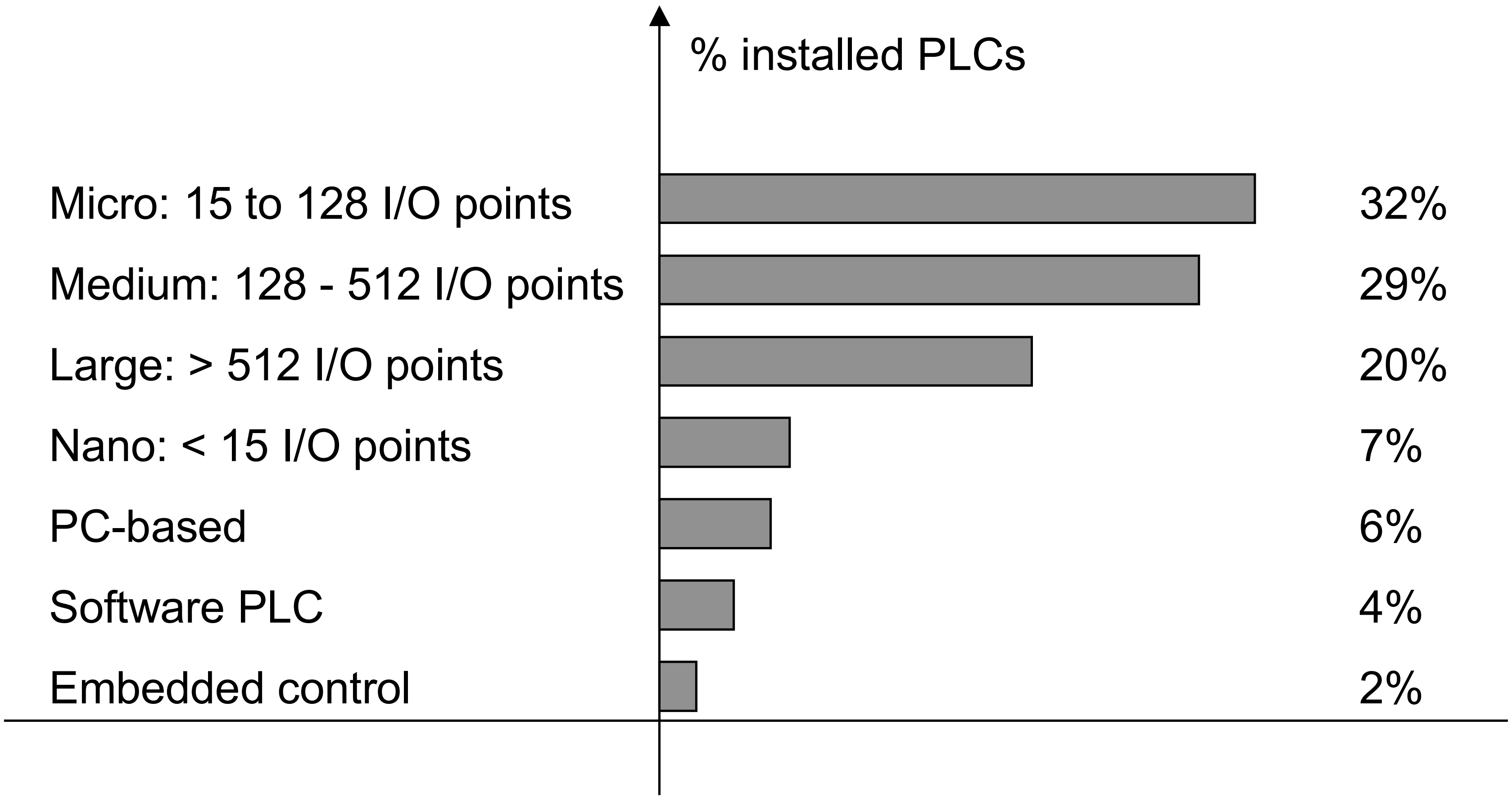
Protection devices



Protection devices are highly specialized PLCs that measure the current and voltages in an electrical substation, along with other status (position of the switch) to detect situations that could endanger the equipment (over-current, short circuit, overheat) and triggers the circuit breaker (“trip”) to protect the substation.
In addition, it records disturbances and sends the reports to the substation’s SCADA.
Sampling: 4.8 kHz, reaction time: < 5 ms.

costs: € 5000

Market share



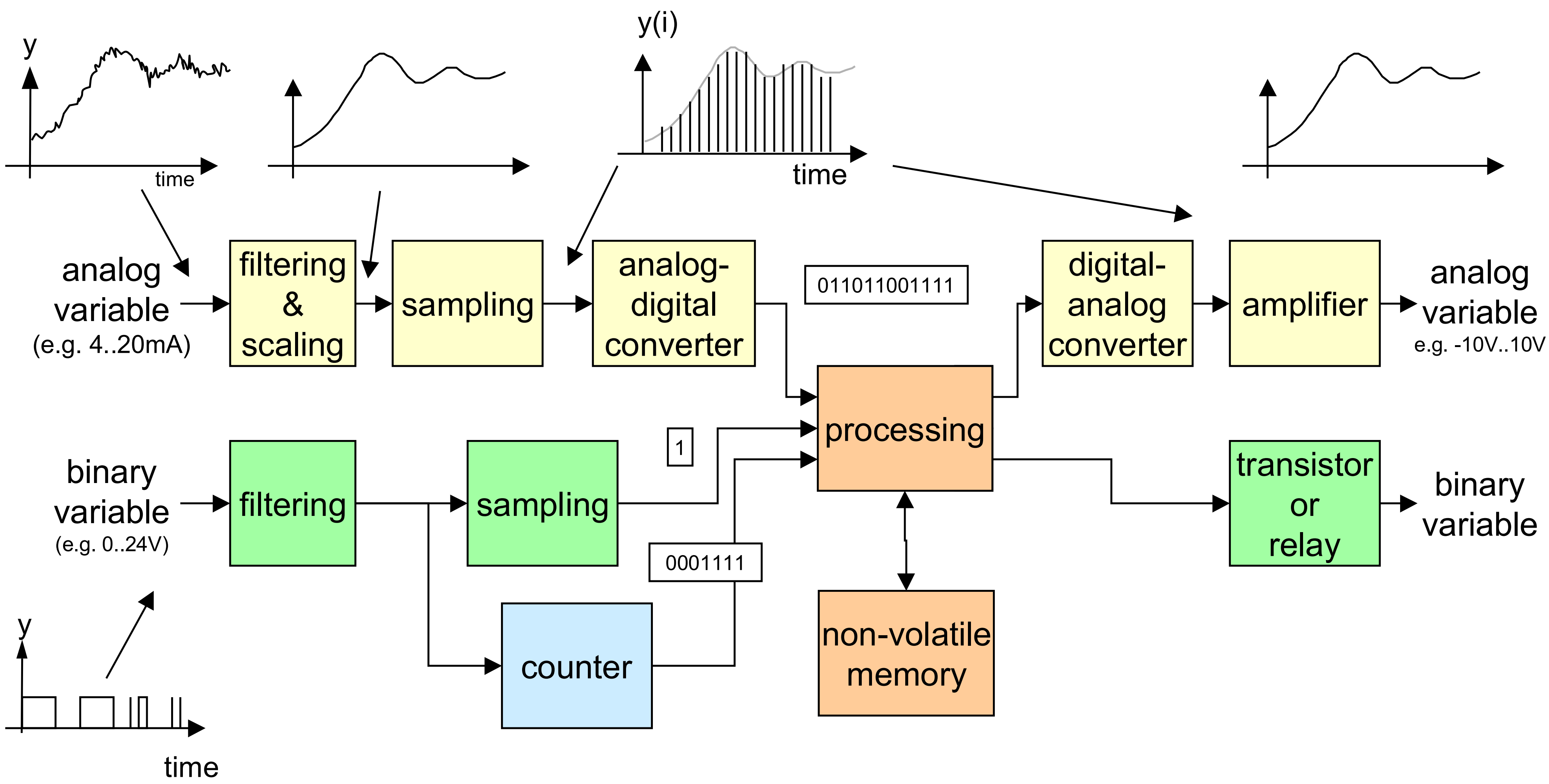
Source: Control Engineering, Reed Research, 2002-09

Comparison Criteria – Example

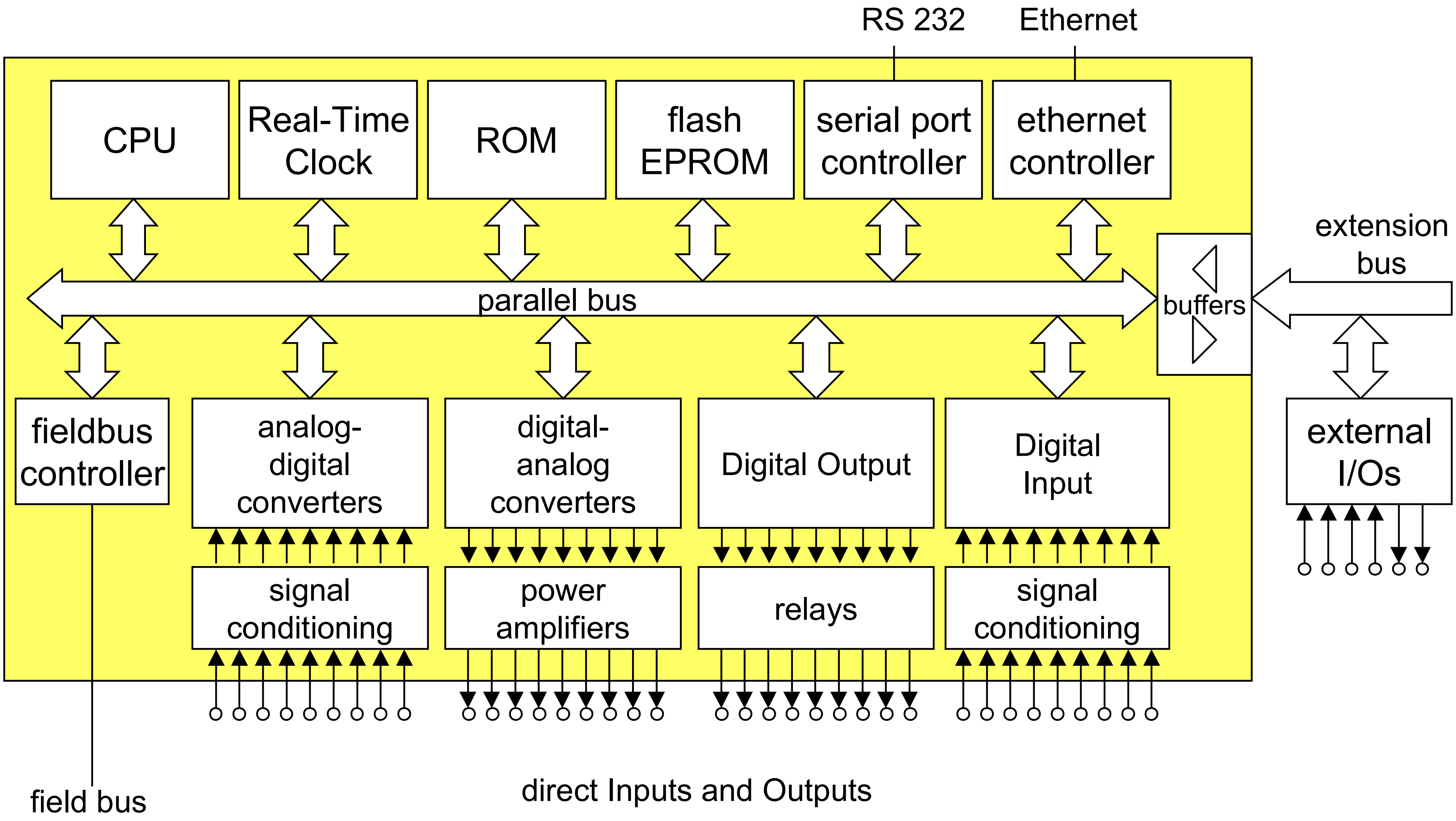
Brand	Siemens	Hitachi
Number of Points	1024	640
Memory	10 KB	16 KB
Programming Language	• Ladder logic • Instructions • Logic symbols • Hand-terminal	• Ladder Logic • Instructions • Logic symbols • Basic • Hand-terminal
Programming Tools	• Graphic on PC	• Graphic on PC
Download	no	yes
Real estate per 250 I/O	2678 cm2	1000 cm2
Label surface per line/point	5.3 mm2 7 characters	6 mm2 6 characters
Network	10 Mbit/s	19.2 kbit/s
Mounting	DIN rail	cabinet

2.3.3 PLCs: Function and construction

The signal chain



General PLC architecture



2.3.4 Continuous and discrete control

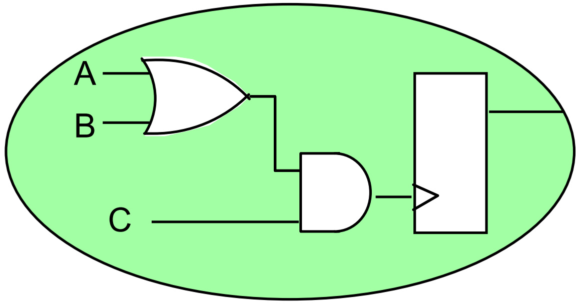
Matching the analog and binary world

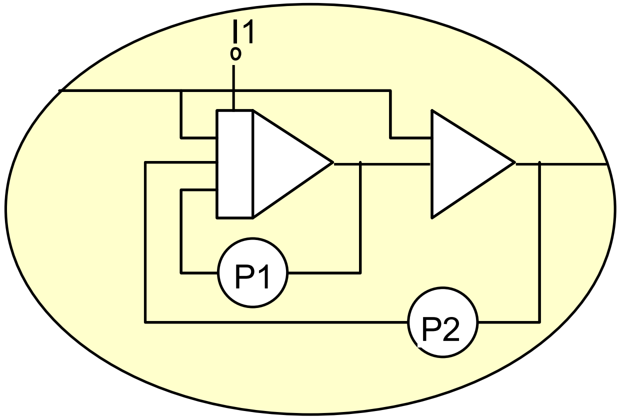
discrete control

analog regulation

PLC evolution

Binary World	Analog World
relay controls, Relay control pneumatic sequencer	Pneumatic and electromechanical controllers





combinatorial   sequential

Regulation, controllers

discrete processes

continuous processes

Programmable Logic Controllers

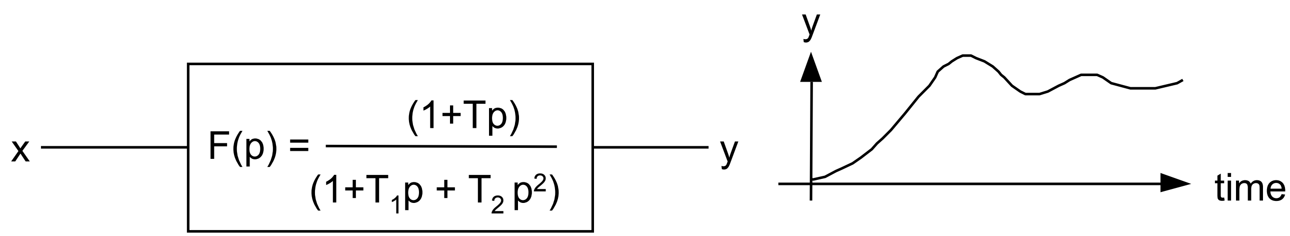
(Speicherprogrammierbare Steuerungen, Automates Programmables)

Continuous Plant (reminder)

Example: traction motors, ovens, pressure vessel,...

The state of continuous plants is described by continuous (analog) state variables like temperature, voltage, speed, etc.

There exist a fixed relationship between input and output,described by a continuous model in form of a transfer function F.
This transfer function can be expressed by a set of differential equations.
If equations are linear, the transfer function may be given as Laplace or Z-transform.



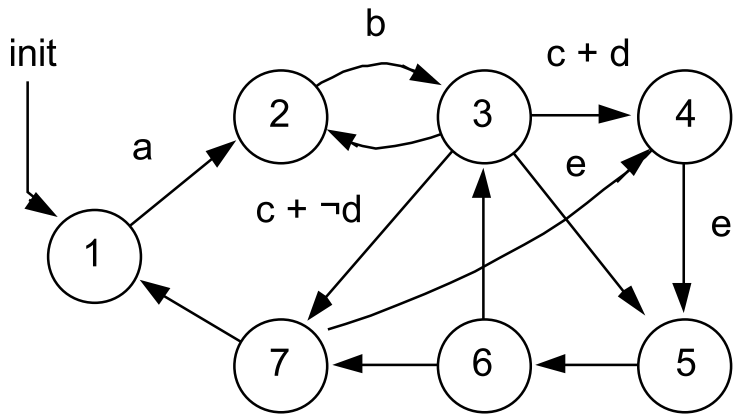
Continuous plants are normally reversible and monotone.
This is the condition to allow their regulation.

The time constant of the control system must be at least one order of magnitude smaller than the smallest time constant of the plant.

the principal task of the control system for a continuous plant is its regulation.

Discrete Plant (reminder)

Examples: Elevators, traffic signaling, warehouses, etc.



The plant is described by variables which take well-defined, non-overlapping values.
The transition from one state to another is abrupt, it is caused by an external event.
Discrete plants are normally reversible, but not monotone, i.e. negating the event which caused a transition will not revert the plant to the previous state.

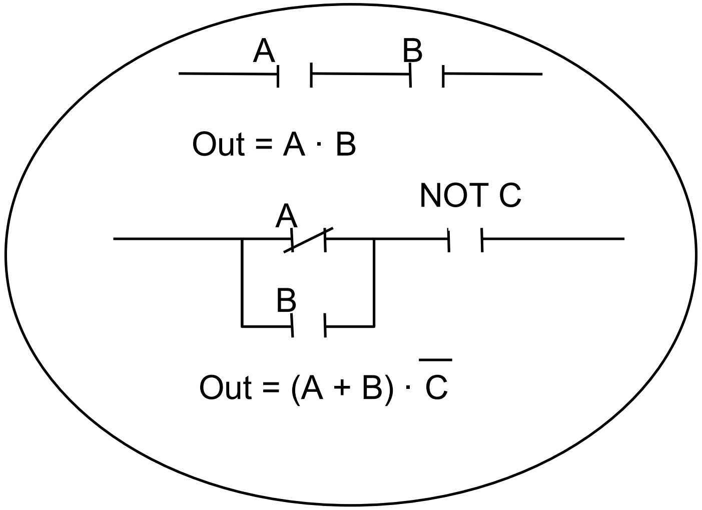
Example: an elevator doesn't return to the previous floor when the button is released.

Discrete plants are described e.g. by finite state machines or Petri nets.

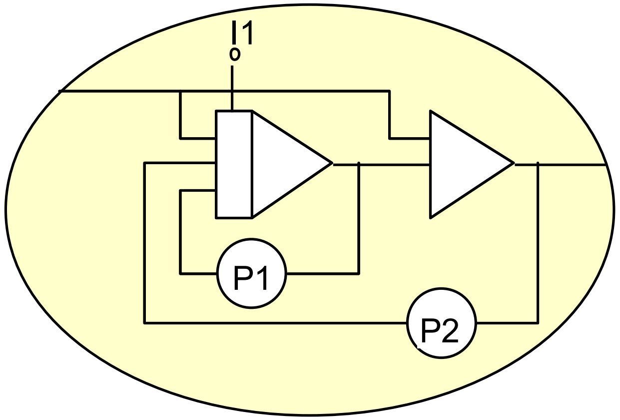
the main task of a control system with discrete plants is its sequential control.

Continuous and Discrete Control (comparison)

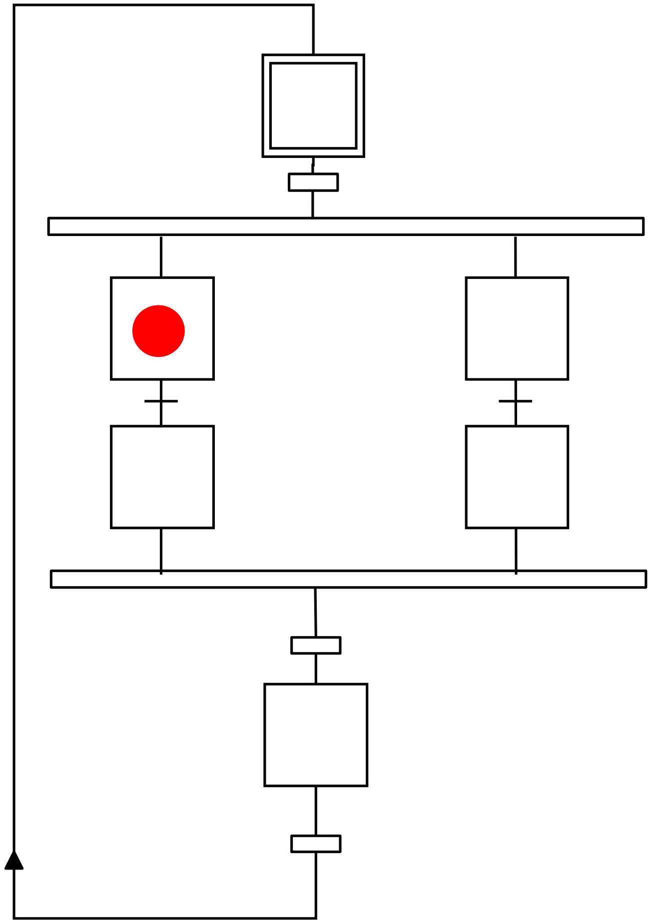
"combinatorial"[1]	"sequential"
e.g. ladder logic, CMOS logic	e.g. GRAFCET, Petri Nets



ladder logic



analog building blocs



1) not really combinatorial: blocs may have memory

2.3.5 Programming languages

"Real-Time" languages

Extend procedural languages to express time

("introduce programming constructs to influence scheduling and control flow")

- ADA
- Real-Time Java
- MARS (TU Wien)
- Forth
- "C" with real-time features
- etc…

could not impose themselves

languages developed for cyclic execution and real-time

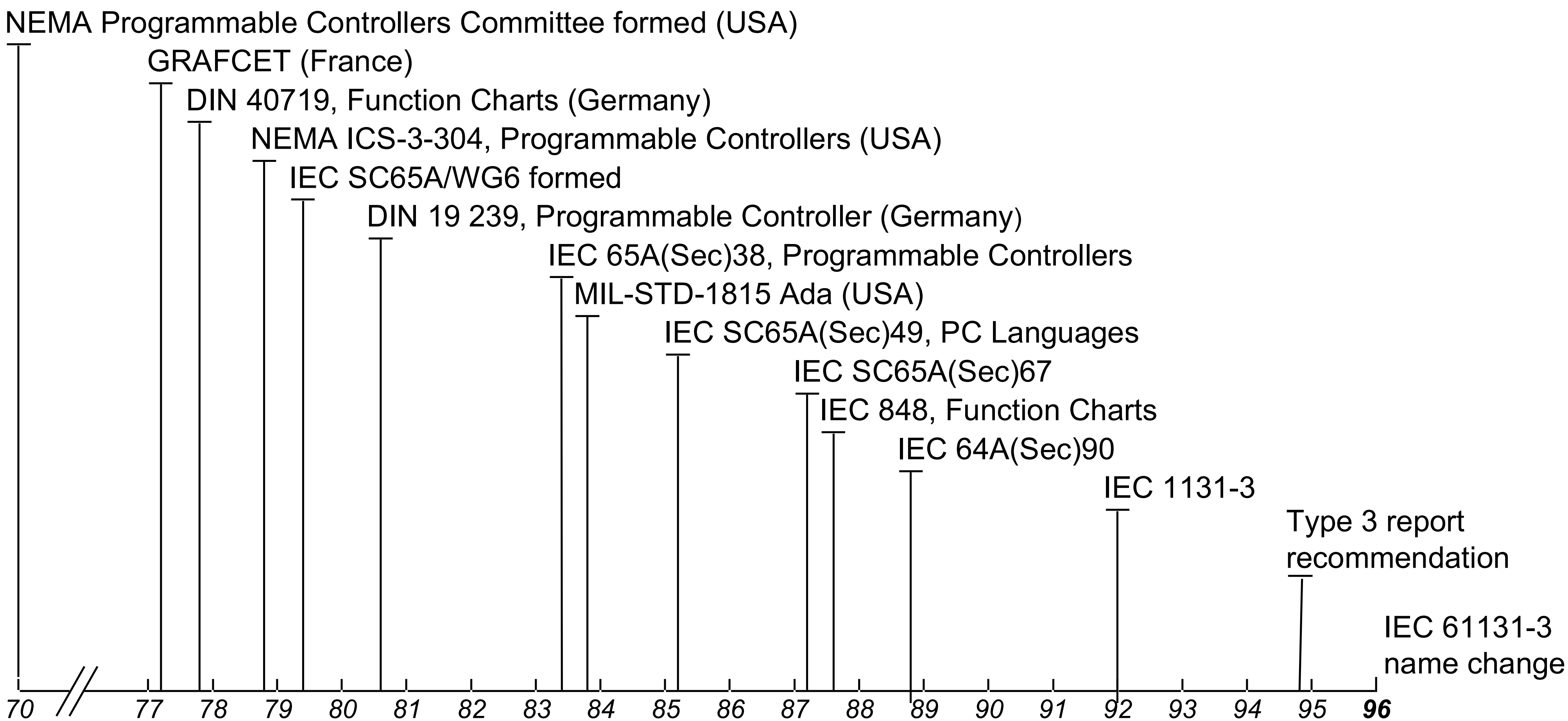
("application-oriented languages")

- ladder logic
- function block language
- instruction lists
- GRAFCET
- SDL

etc...

wide-spread in the control industry. Now standardized as IEC 61131

The long march to IEC 61131



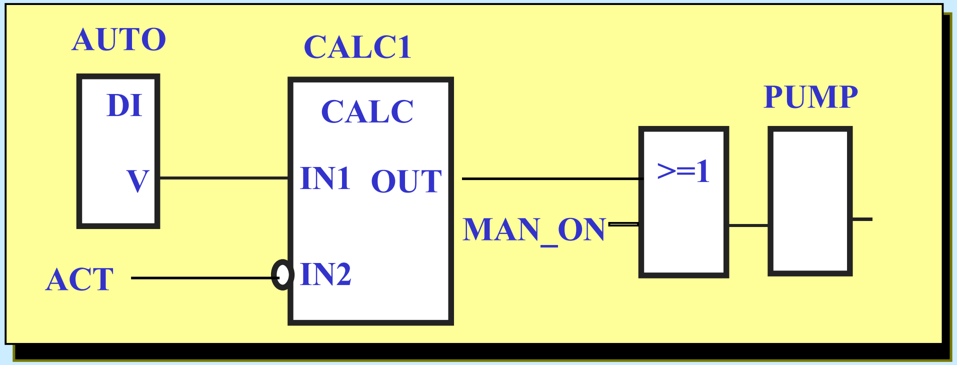
Source: Dr. J. Christensen

The five IEC 61131-3 Programming languages

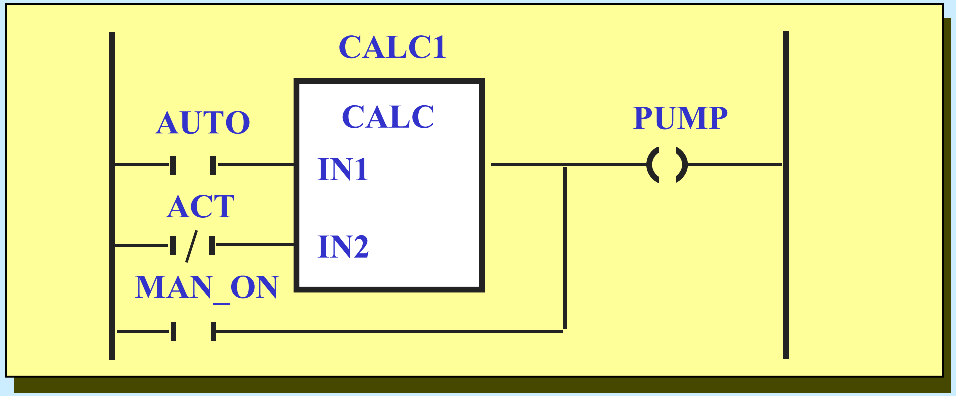
http://www.isagraf.com

graphical languages

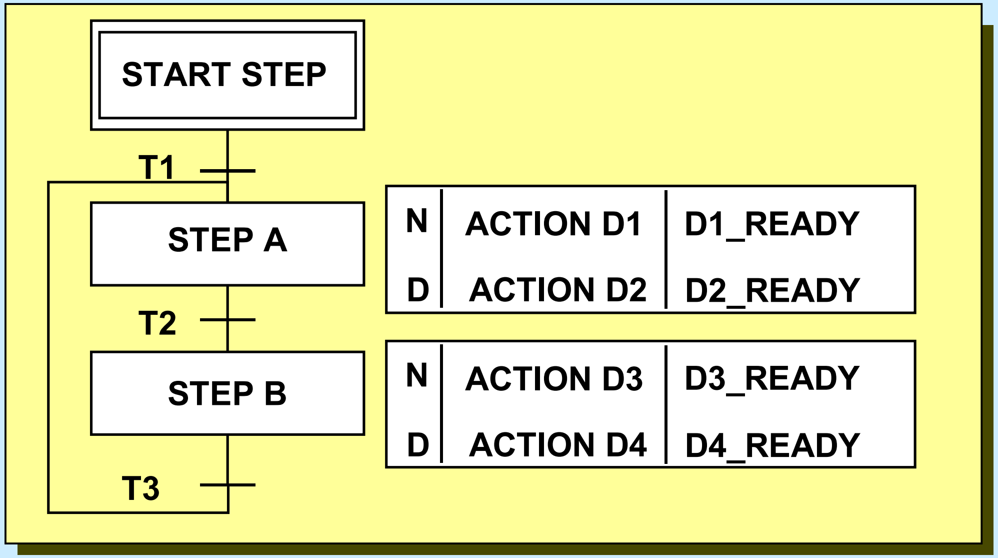
Function Block Diagram (FBD)



Ladder Diagram (LD)



Sequential Flow Chart (SFC)



textual languages

Instruction List (IL)

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

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
```

Importance of IEC 61131

IEC 61131-3 is the most important automation language in industry.

80% of all PLCs support it, all new developments base on it.

Depending on the country, some languages are more popular.

2.4.2.1 Function Blocks Language

Function Block Languages

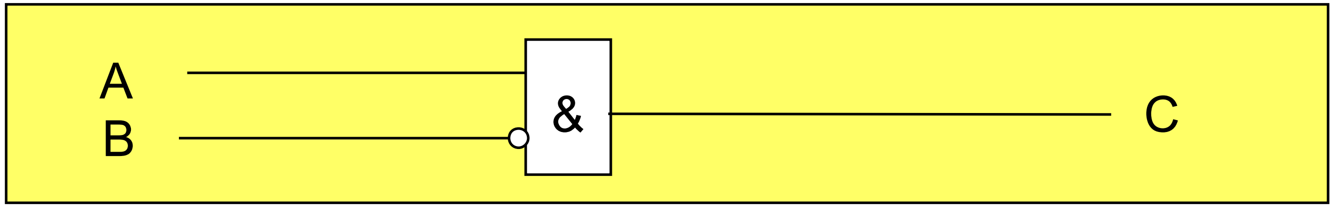
(Funktionsblocksprache, langage de blocs de fonctions)
(Also called "Function Chart" or "Function Plan" - FuPla)

The function block languages express "combinatorial" programs in a way similar to electronic circuits.
They draw on a large variety of predefined and custom functions

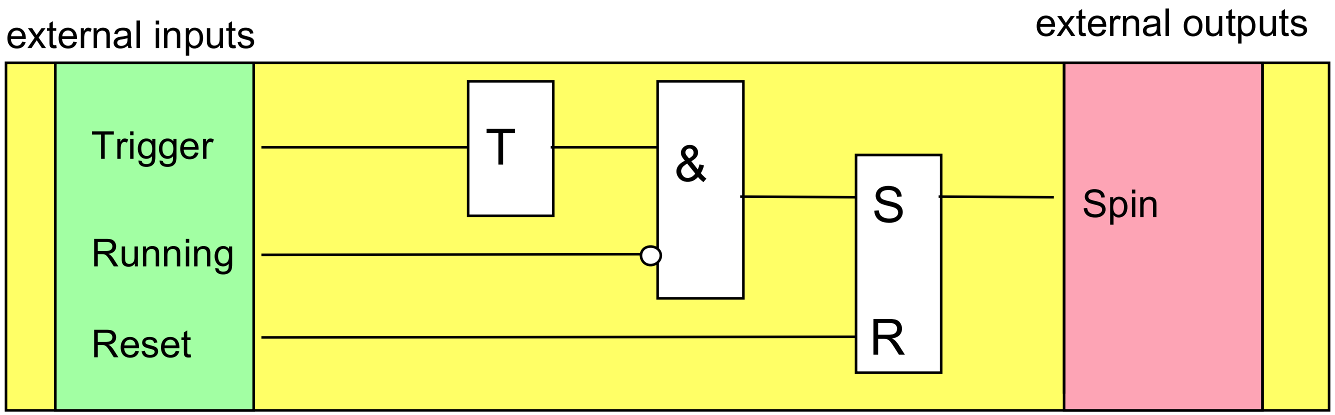
This language is similar to the Matlab / Simulink language used in simulations

Function Block Examples

Example 1:



Example 2:



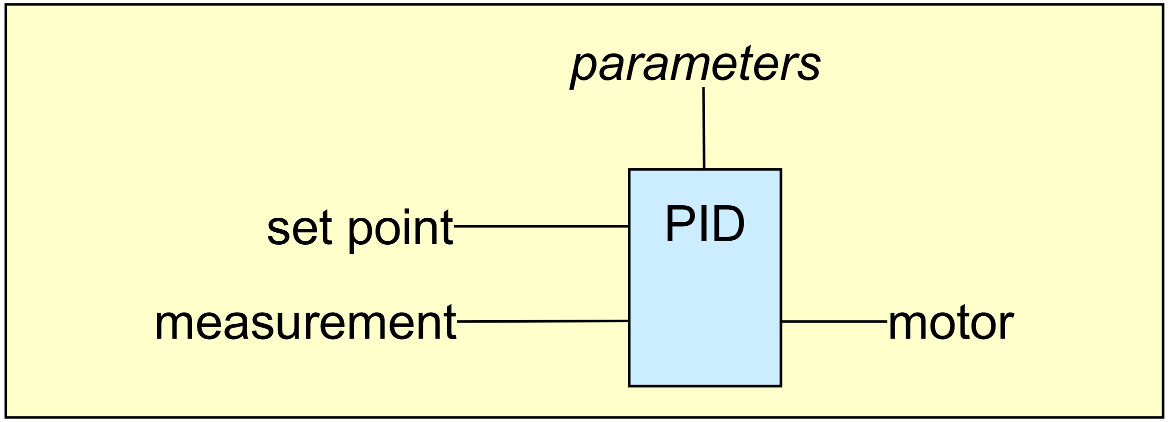
Function blocks is a graphical programming language, which is akin to the electrical and block diagrams of the analog and digital technique.

It mostly expresses combinatorial logic, but its blocks may have a memory (e.g. flip-flops).

Function Block Elements

Function block

Example



"continuously" executing block, independent, *no side effects*

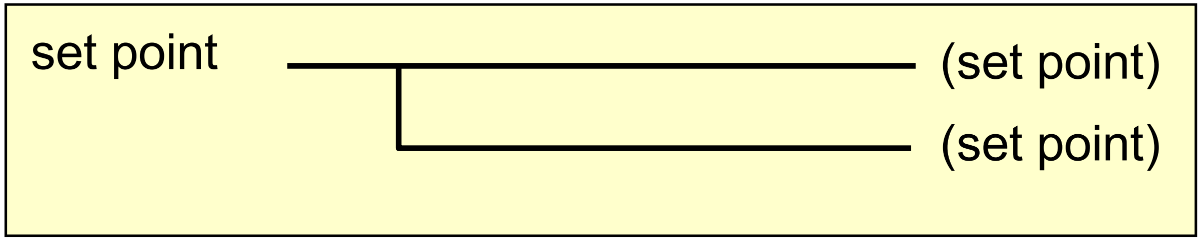
The block is defined by its:

- Data flow interface (number and type of input/output signals)
- Black-Box-Behavior (functional semantic, e.g. in textual form).

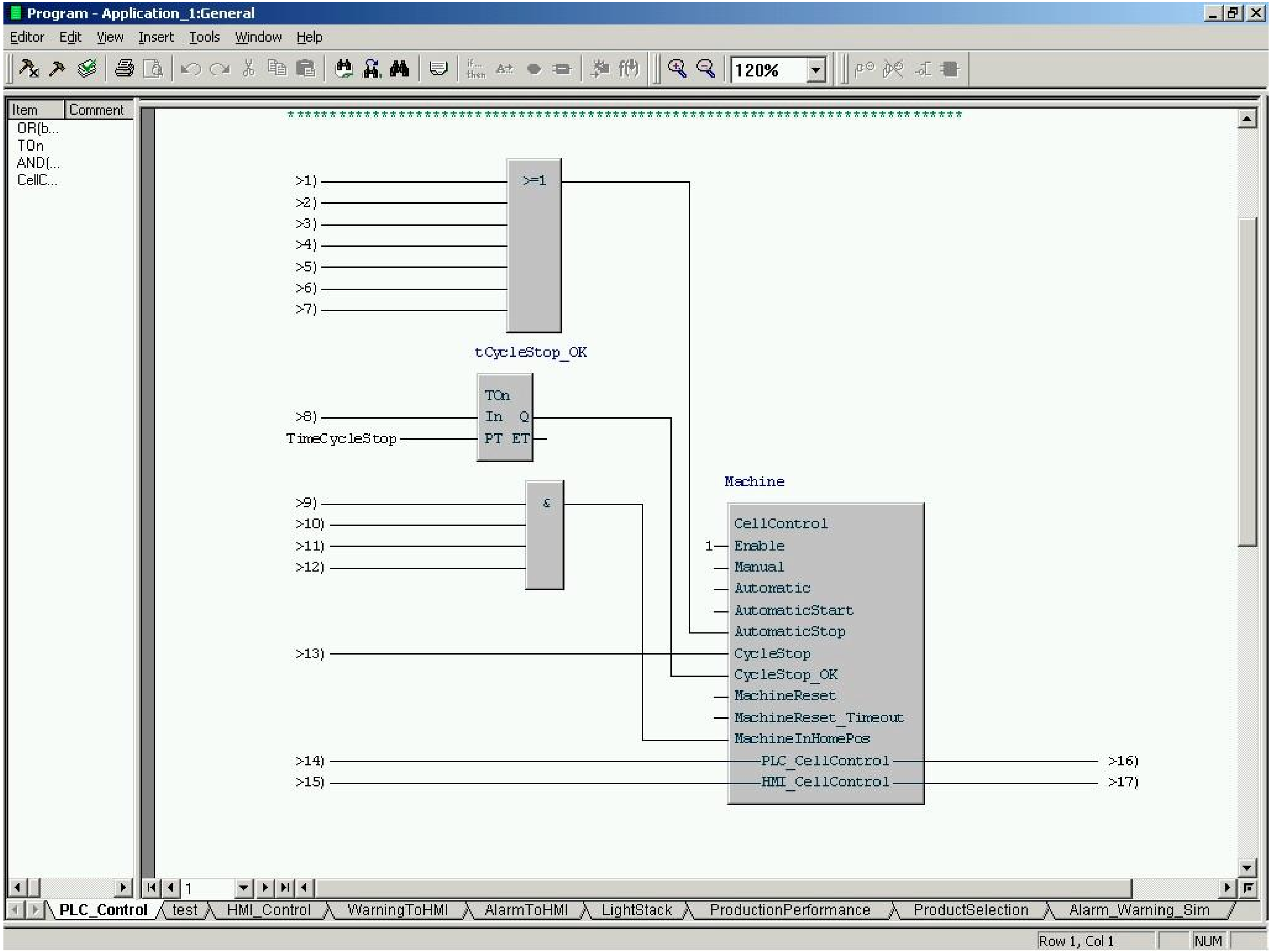
Signals

Connections that carry a pseudo-continuous data flow.
Connects the function blocks.

Example



Function Block Example



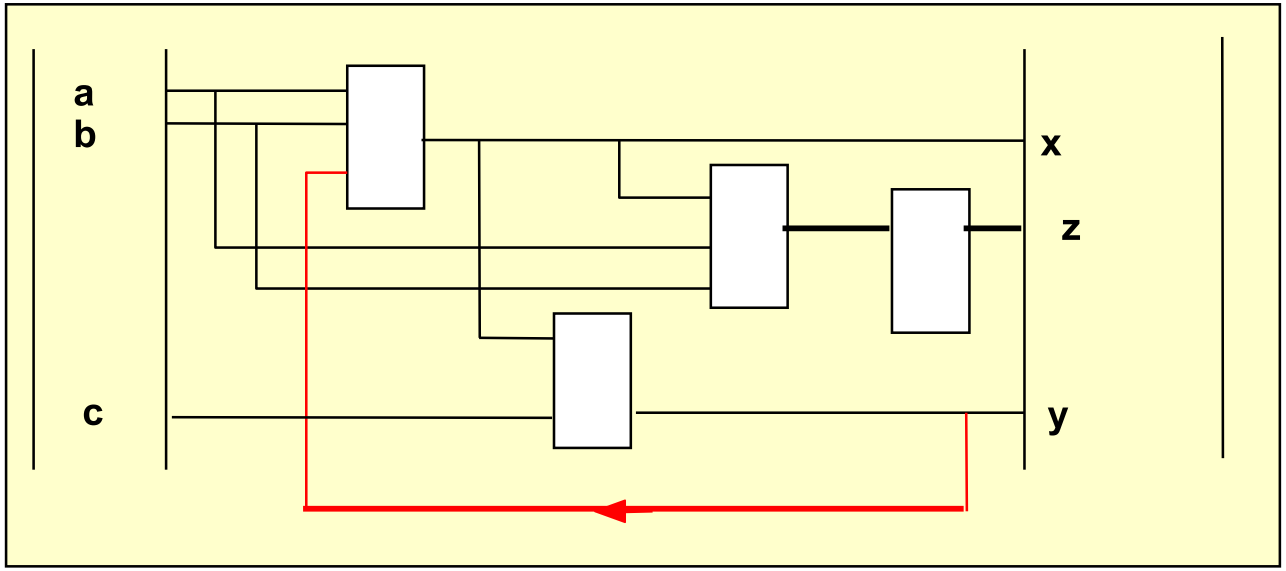
Function Block Rules

There exist exactly two rules for connecting function blocks by signals (this is the actual programming):

- Each signal is connected to exactly one source.
  This source can be the output of a function block or a plant signal.
- The type of the output pin, the type of the input pin and the signal type must be identical.

For convenience, the function plan should be drawn so the signals flow from left to right and from top to bottom. Some editors impose additional rules.

Retroactions are exception to this rule. In this case, the signal direction is identified by an arrow. (Some editors forbid retroactions - use duplicates instead).



Types of Programming Organisation Units (POUs)

1) Functions

- are part of the base library.
- have no memory.

Example are: adder, multiplier, selector,....

2) Elementary Function Blocks (EFB)

- are part of the base library
- have an individual memory ("static" data).
- may access global variables (side-effects!)

Examples: counter, filter, integrator,.....

3) Programs (Compound blocks)

- user-defined or application-specific blocks
- may have a memory
- may be configurable (control flow not visible in the FBD

Examples: PID controller, Overcurrent protection, Motor sequence
(a library of compound blocks may be found in IEC 61804-1)

Function Block library

The programmer chooses the blocks in a block library, similarly to the hardware engineer who chooses integrated circuits out of the catalogue.

This library indicates the pinning of each block, its semantics and the execution time.

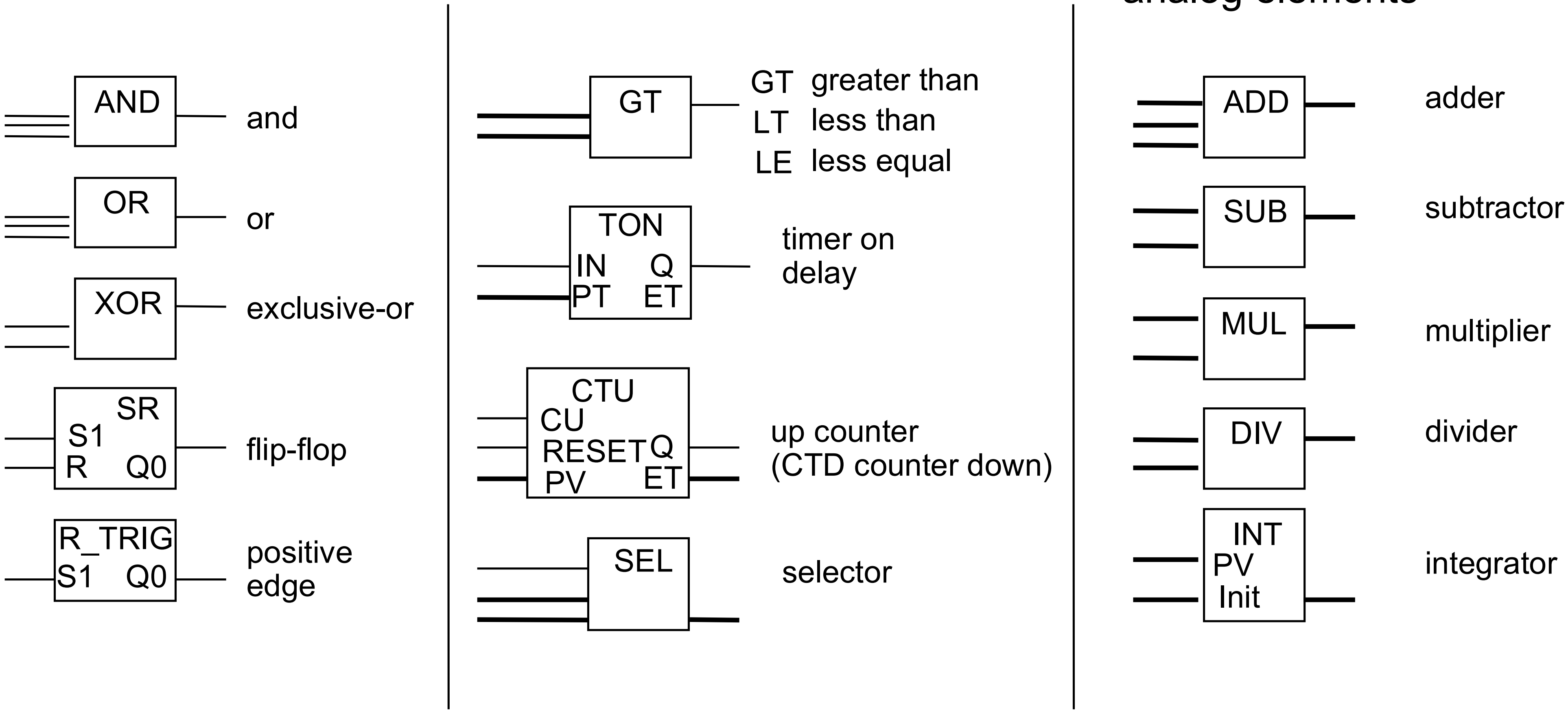
The programmer may extend the library by defining function block macros out of library elements.

If some blocks are often used, they will be programmed in an external language (e.g. “C”, micro-code) following strict rules.

IEC 61131-3 library (extract)

binary elements

analog elements

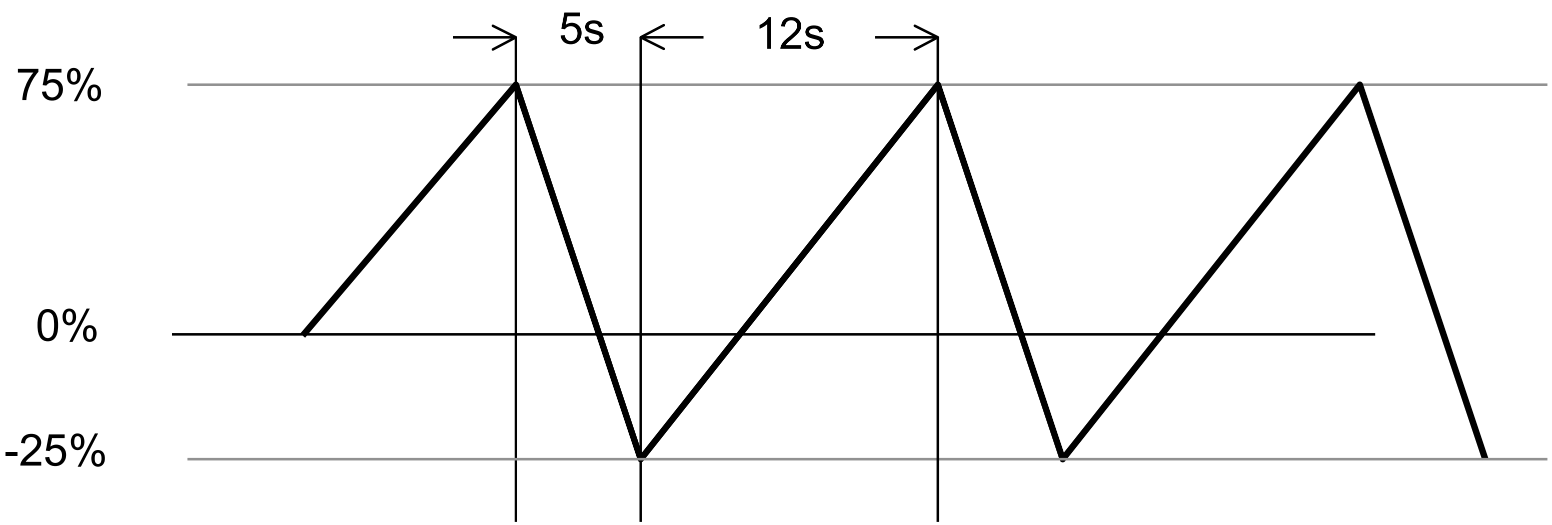


The number of inputs or outputs and their type is restricted.

The execution time of each block depends on the number of inputs and on the processor.

Exercise: Tooth saw generator

exercise: build a tooth-saw (asymmetric) generator with the IEC 61131 elements of the preceding page



Library functions for discrete plants

Basic blocks

- logical combinations (AND, OR, NOT, EXOR)
- Flip-flop
- Selector m-out-of-n
- Multiplexer m-to-n
- Timer
- Counter
- Memory
- Sequencing

Compound blocks

- Display
- Manual input, touch-screen
- Safety blocks (interlocking)
- Alarm signaling
- Logging

Analog function blocks for continuous control

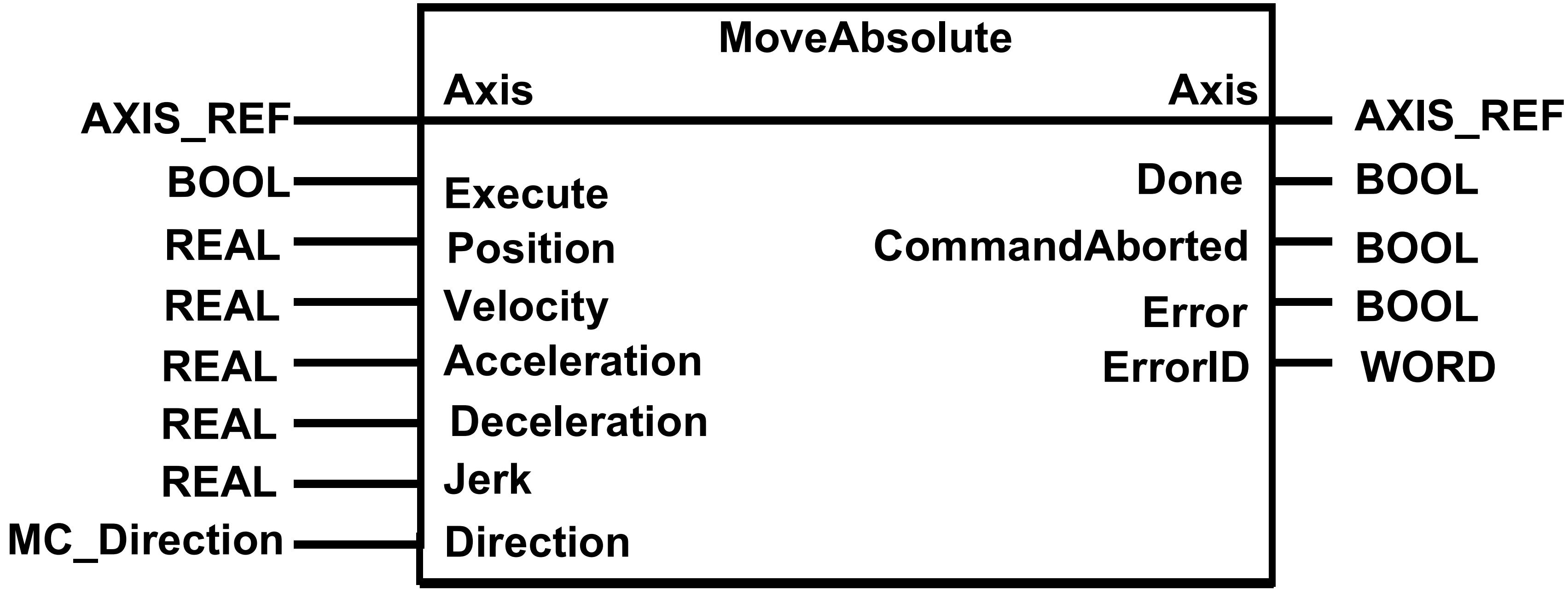
Basic blocks

Summator / Subtractor
Multiplier / Divider
Integrator / Differentiator
Filter
Minimal value, Maximum value
Radix
Function generator

Regulation Functions

P, PI, PID, PDT2 controller
Fixed set-point
Ratio and multi-component regulation
Parameter variation / setting
2-point regulation
3-point regulation
Output value limitation
Ramp generator
Adaptive regulation
Drive Control

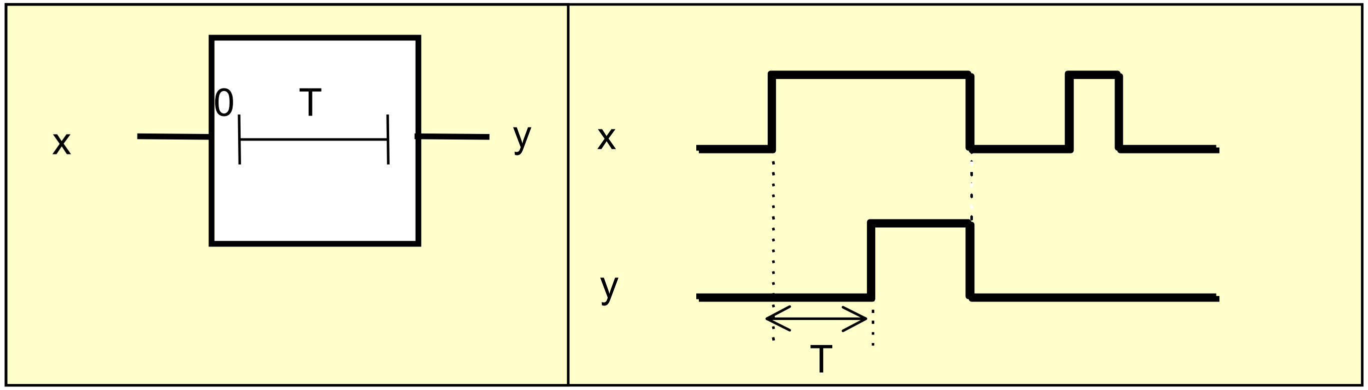
Function Block library for specialized applications



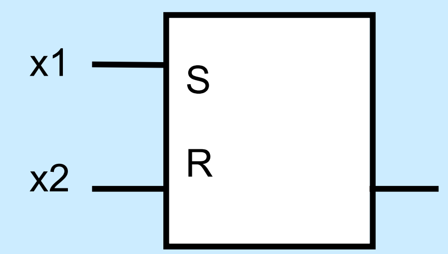
Example: FB for motion control

Specifying the behaviour of Function Block

Time Diagram:



Truth Table:



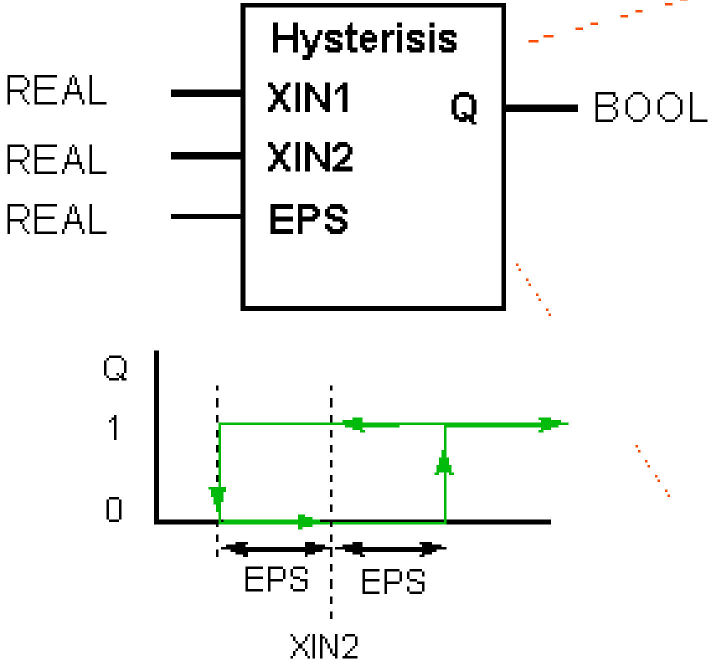
x1	x2	y
0	0	previous state
0	1	0
1	0	1
1	1	1

Mathematical Formula:

x → $K_p x + K_d \frac{dx}{dt} + K_i \int_0^t x d\tau$ → y

Textual Description: Calculates the root mean square of the input with a filtering constant Equal

Function Block specification in Structured Text

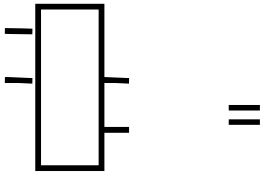



```
FUNCTION_BLOCK HYSTERISIS
 VAR_INPUT
   XIN1, XIN2 : REAL;
   EPS : REAL;  (* Hysteris:
 END_VAR
 VAR_OUTPUT
   Q : BOOL := 0
 END_VAR
 IF Q THEN
   IF XIN1 < (XIN2-EPS) THEI
     Q := 0 (* XIN1 decreas:
   END_IF;
 ELSIF XIN1 > (XIN2 + EPS )
   Q := 1; (* XIN1 increasi1
 END_IF;
END_FUNCTION_BLOCK
```

Function Block decomposition

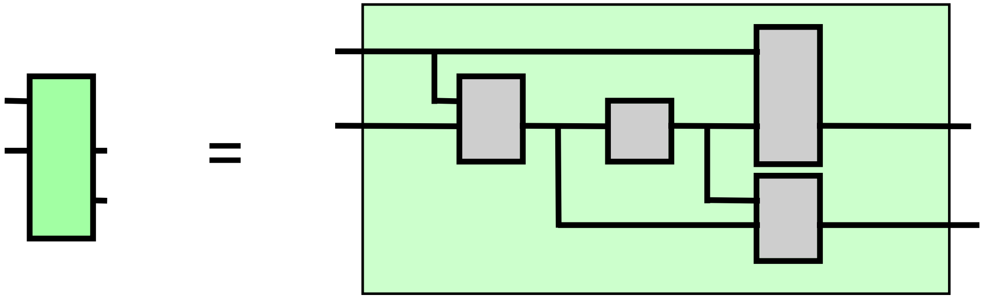
A function block describes a *data flow interface.*
Its *body* can be implemented differently:

Elementary block The body is implemented in an *external language* (micro-code, assembler, java, IEC 61131 ST):



```
procedure xy(a,b:BOOLEAN; VAR b,c: BOOLEAN);
begin
  ......
  ....
end xy;
```

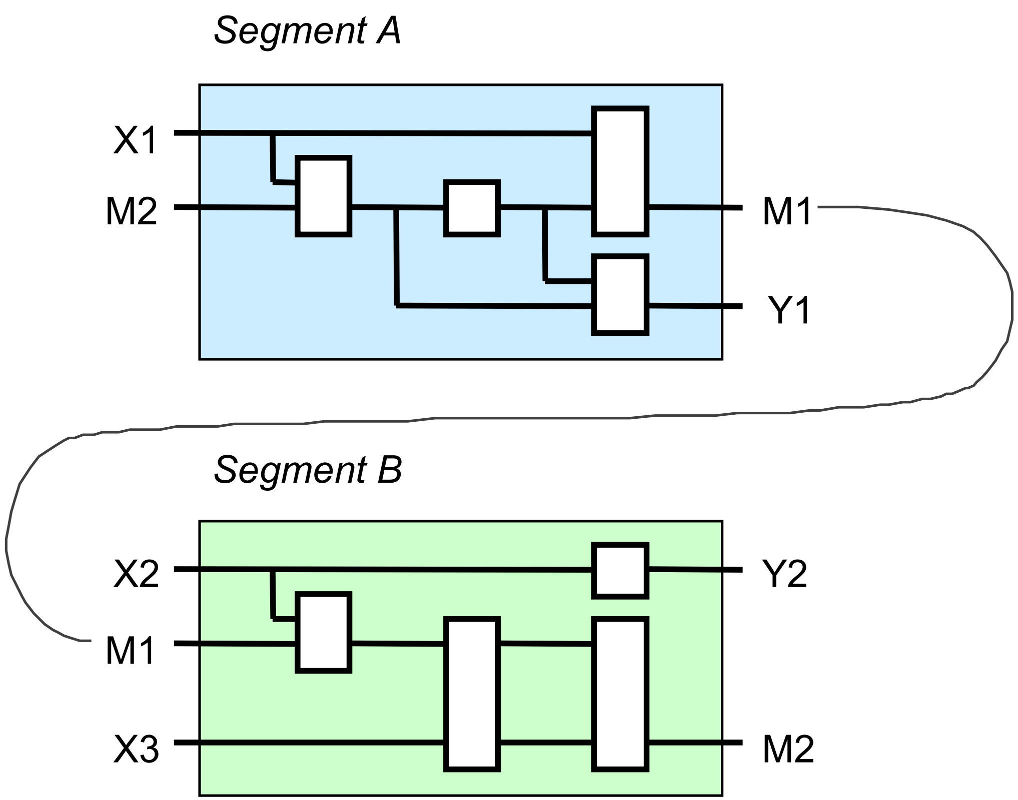
Compound block The body is realized as a *function block program*.
Each input (output) pin of the interface is implemented as exactly one input (output) of the function block.
All signals must appear at the interface to guarantee freedom from *side effects.*



Function Block segmentation

An application program is decomposed into segments ("Programs") for easier reading, each segment being represented on one (A4) printed page.

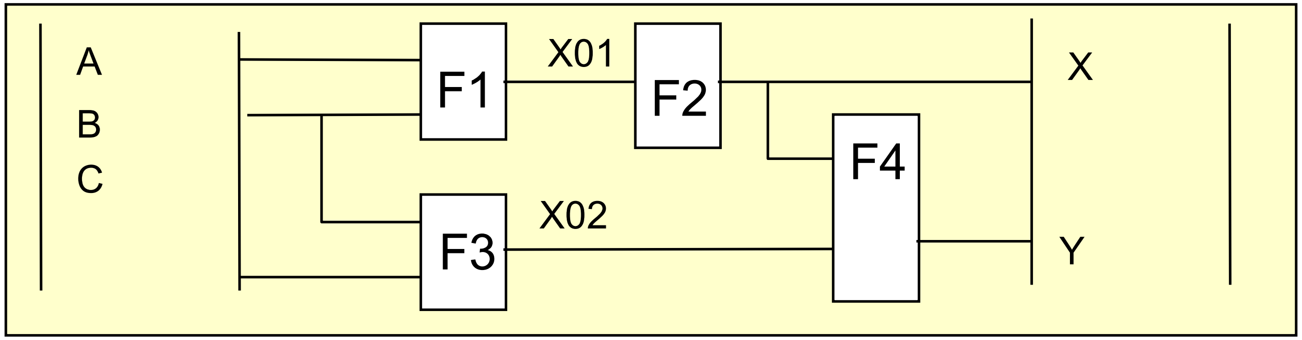
- Within a segment, the connections are represented *graphically*
- Between the segments, the connections are expressed by *signal names*



2.3.5.3 Program execution

Execution of Function Blocks

Segment or POU
(program organization unit)

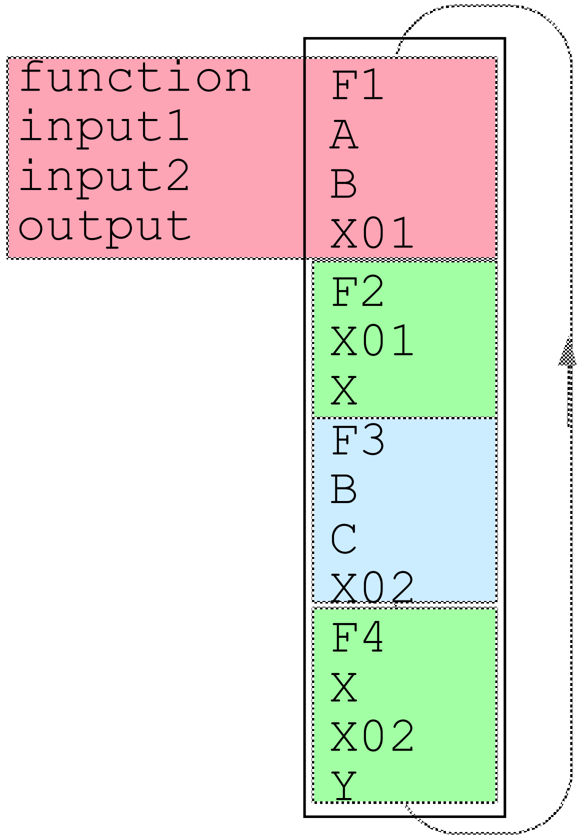


Machine Code:

The function blocks are translated to machine language (intermediate code, IL), that is either interpreted or compiled to assembly language

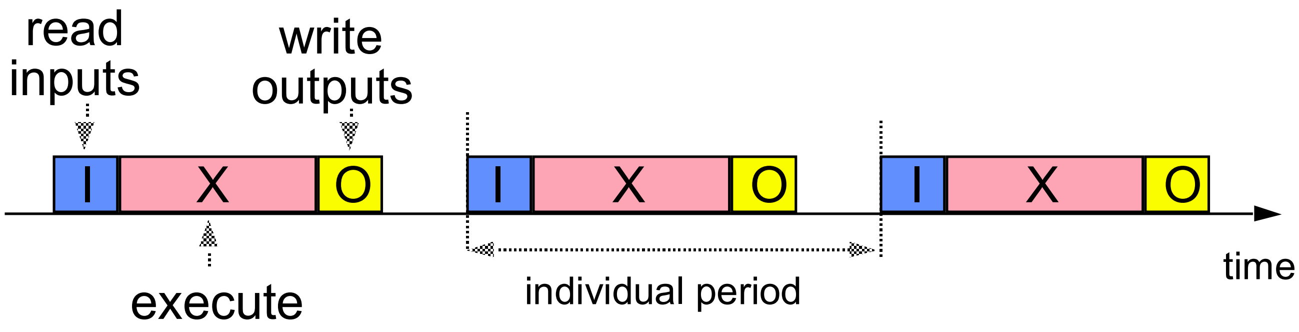
Blocks are executed in sequence, normally from upper left to lower right

The sequence is repeated every x ms.



Input-Output of Function Blocks

Run-time:



The function blocks are executed cyclically.
- all inputs are read from memory or from the plant (possibly cached)
- the segment is executed
- the results are written into memory or to the plant (possibly to a cache)

The order of execution of the blocks generally does not matter.
To speed up algorithms and avoid cascading, it is helpful to impose an execution order to the blocks.

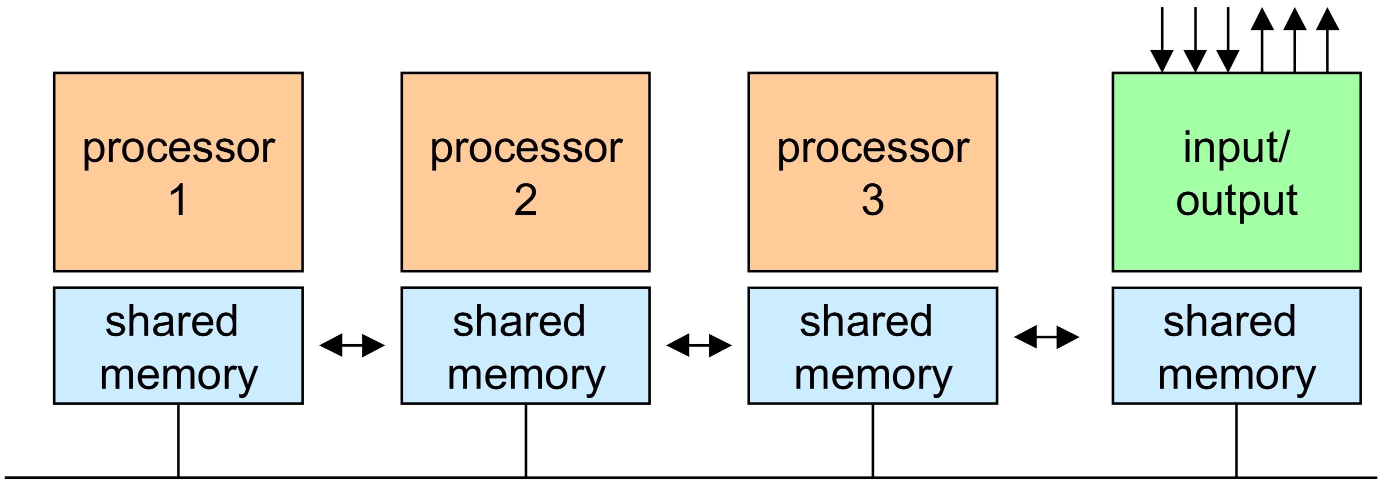
The different segments may be assigned a different individual period.

Parallel execution

Function blocks are particularly well suited for true multiprocessing (parallel processors).

The performance limit is given by the needed exchange of signals by means of a shared memories.

Semaphores are not used since they could block an execution and make the concerned processes non-deterministic.



Program configuration

The programmer divides the program into tasks (sometimes called pages or segments), which may be executed each with a different period.

The programmer assigns each task (each page) an execution period.

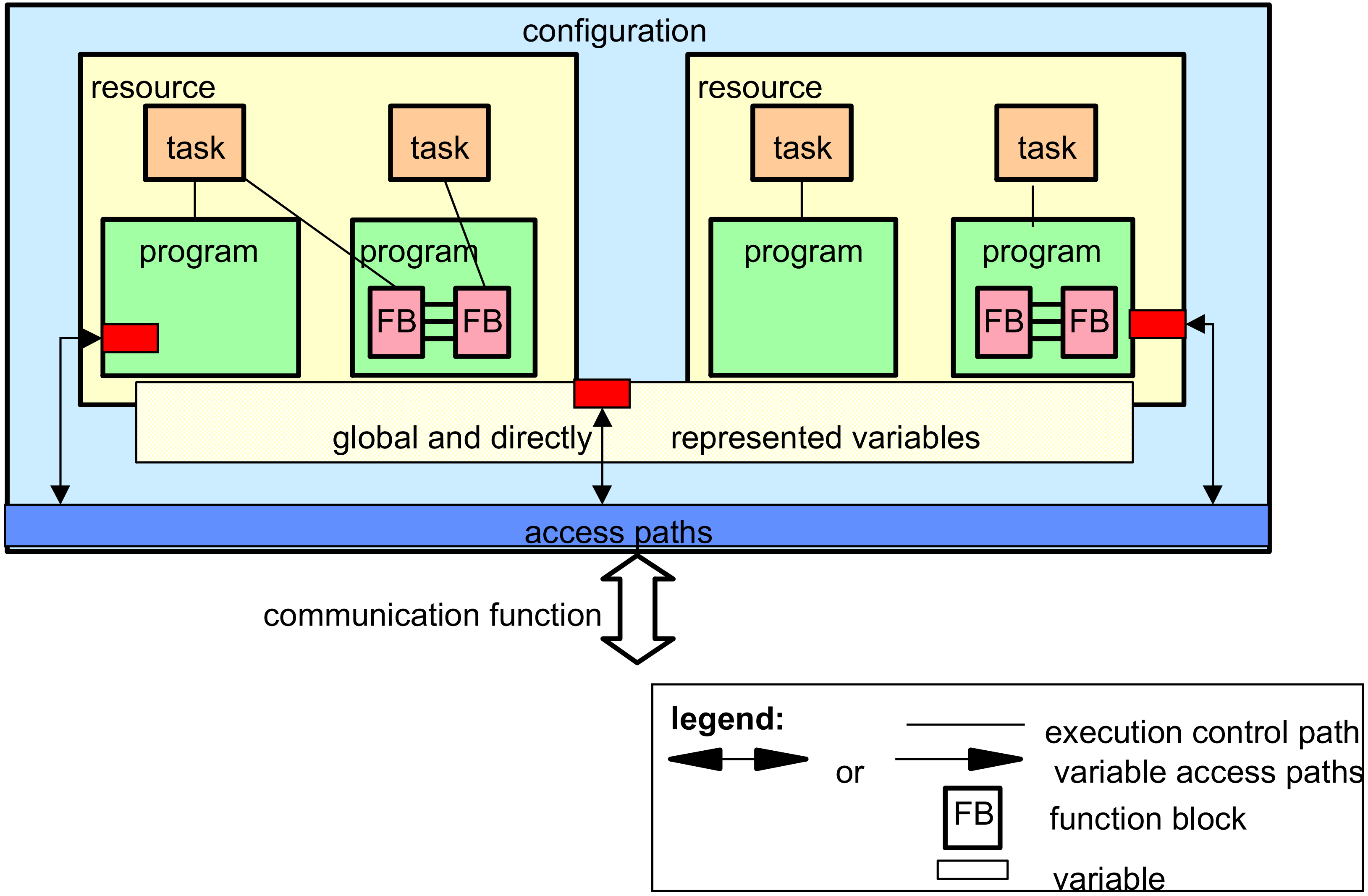
Since the execution time of each block in a task is fixed, the execution time is fixed.

Event-driven operations are encapsulated into blocks, e.g. for transmitting messages.

If the execution time of these tasks cannot be bound, they are executed in background.

The periodic execution always has the highest priority.

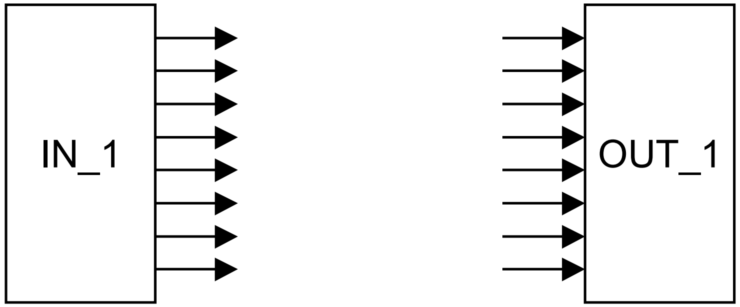
IEC 61131 - Execution engine



2.3.5.4 Input and Output

Connecting to Input/Output, Method 1: dedicated I/O blocks

The Inputs and Outputs of the PLC must be connected to (typed) variables



The I/O blocks are configured to be attached to the corresponding I/O groups.

Connecting to Input / Output, Method 2: Variables configuration

All program variables must be declared with name and type, initial value and volatility.
A variable may be connected to an input or an output, giving it an I/O address.
Several properties can be set: default value, fall-back value, store at power fail,…
These variables may not be connected as input, resp. output to a function block.

	Name	Data Type	Attributes	Initial Value	I/O Address	Access Va
3	HK_Float	real	retain	0.0	Controller_1.0.11.3.1	Controller_
4	HK_Min	real	retain	-100.0		
5	HK_Max	real	retain	+100.0		
6	HK_OnOff	bool	retain		Controller_1.0.11.1.1	
7	DZ_Input	bool	retain		Controller_1.0.11.1.2	
8	HK_DInput1	bool	retain		Controller_1.0.11.2.1	
9	HK_Dinput2	bool	retain		Controller_1.0.11.2.2	
10	HK_DInput3	bool	retain		Controller_1.0.11.2.3	
11	HK_DInput4	bool	retain		Controller_1.0.11.2.4	
12	HK_DInput5	bool	retain		Controller_1.0.11.2.5	
13	HK_DInput6	bool	retain		Controller_1.0.11.2.6	
14	HK_DInput7	bool	retain		Controller_1.0.11.2.7	
15	HK_DInput8	bool	retain		Controller_1.0.11.2.8	
16	Right2LeftCnt	int	retain	0		
17	HK_MaxTime	time	retain	5s23ms		

Variables | Function Blocks

predefined addresses

2.3.5.5 Structured Text

Structured Text

(*Strukturierte Textsprache*, langage littéral structuré)

language similar to Pascal (If, While, etc..)

The variables defined in ST can be used in other languages.

It is used to do complex data manipulation and write blocs

Caution: writing programs in structured text can breach the real-time rules !

Data Types

Since Function Blocks are typed, the types of connection, input and output must match.

•Elementary Types are defined either in Structured Text or in the FB configuration.

binary types:		analog types:	
BOOL	1	REAL	(Real32)
BYTE	8	LREAL	(Real64)
WORD	16		
DWORD	32		

•Derived Types are user-defined and must be declared in ***Structured Text***

subrange,
enumerated,
arrays,
structured types
(e.g. AntivalentBoolean2)

variable can receive initial values and be declared as non-volatile (RETAIN)

61131 Elementary Data Types

No.	Keyword	Data Type	Bits
1	BOOL	Boolean	1
2	SINT	Short integer	8
3	INT	Integer	16
4	DINT	Double integer	32
5	LINT	Long integer	64
6	USINT	Unsigned short integer	8
7	UINT	Unsigned integer	16
8	UDINT	Unsigned double integer	32
9	ULINT	Unsigned long integer	64
10	REAL	Real numbers	32
11	LREAL	Long reals	64
12	TIME	Duration	depends
13	DATE	Date (only)	depends
14	TIME_OF_DAY or TOD	Time of day (only)	depends
15	DATE_AND_TIME or DT	Date and time of day	depends
16	STRING	Character string	
17	BYTE	Bit string of length 8	8
18	WORD	Bit string of length 16	16
19	DWORD	Bit string of length 32	32
20	LWORD	Bit string of length 64	64
21		variable length double-byte string	

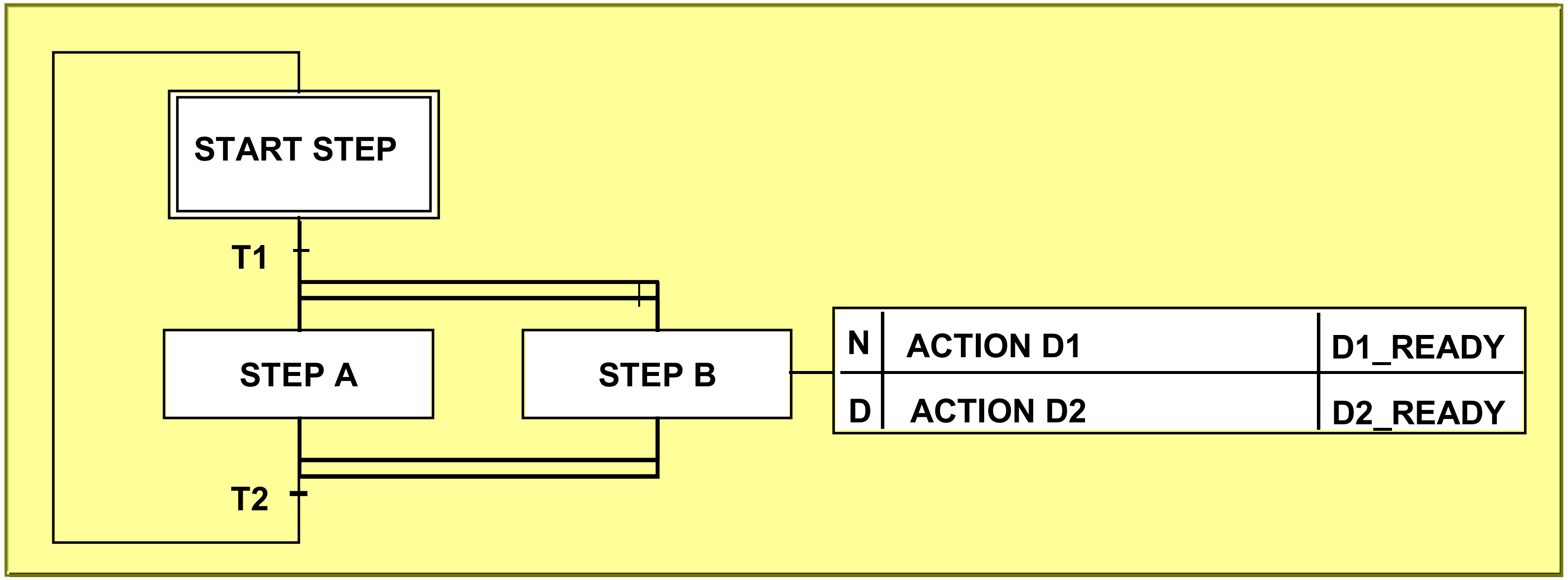
Example of Derived Types

```
TYPE
  ANALOG_CHANNEL_CONFIGURATION
    STRUCT
      RANGE: ANALOG_SIGNAL_RANGE;
      MIN_SCALE : ANALOG_DATA ;
      MAX_SCALE : ANALOG_DATA ;
    END_STRUCT;
  ANALOG_16_INPUT_CONFIGURATION :
    STRUCT
      SIGNAL_TYPE : ANALOG_SIGNAL_TYPE;
      FILTER_CHARACTERISTIC : SINT (0.99)
      CHANNEL: ARRAY [1..16] OF ANALOG_CHANNEL_CONFIGURATION;
    END_STRUCT ;
END_TYPE
```

2.3.5.6 Sequential Function Charts

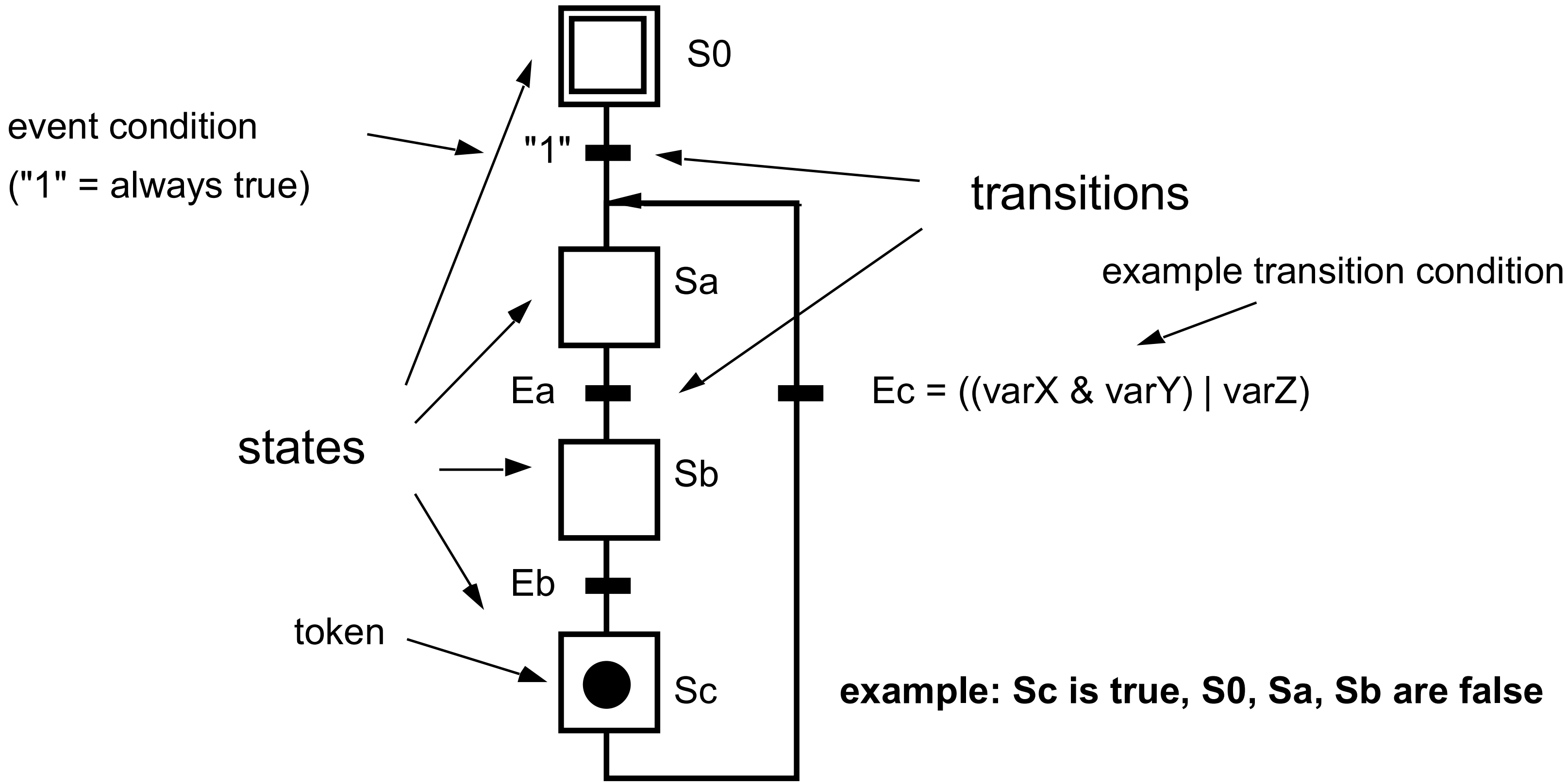
SFC (Sequential Flow Chart)

(*Ablaufdiagramme*, **diagrammes de flux en séquence - grafcet)**



SFC describes sequences of operations and interactions between parallel processes.
It is derived from the languages Grafcet and SDL (used for communication protocols),
its mathematical foundation lies in Petri Nets.

SFC: Elements



The sequential program consists of states connected by transitions.

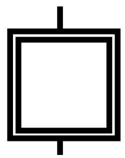
A state is activated by the presence of a token (the corresponding variable becomes TRUE).

The token leaves the state when the transition condition (event) on the state output is true.

Only one transition takes place at a time, the execution period is a configuration parameter

SFC: Initial state

State which come into existence with a token are called *initial states.*



All initial states receive exactly one token, the other states receive none.

Initialization takes place explicitly at start-up.

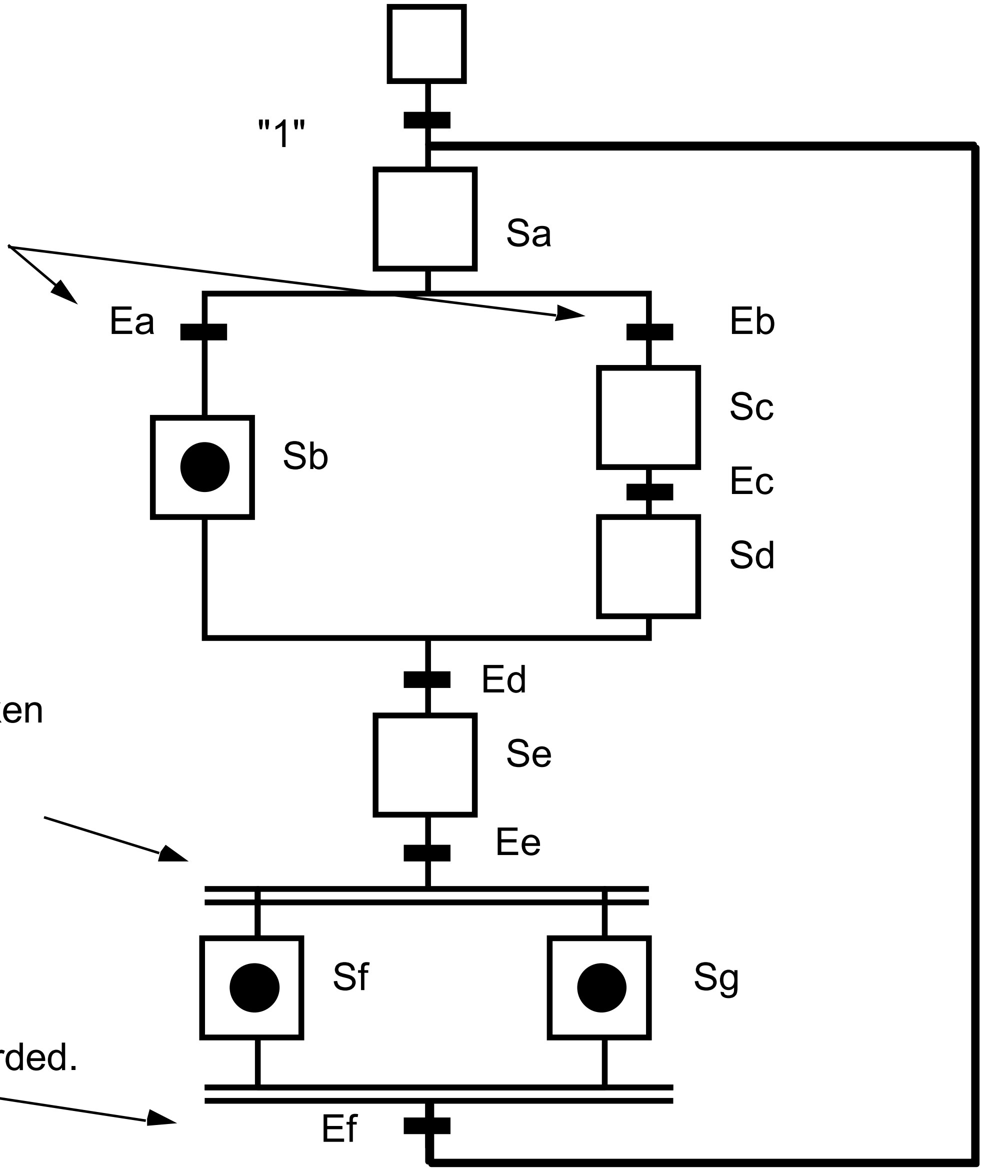
In some systems, initialization may be triggered in a user program (initialization pin in a function block).

SFC: Switch and parallel execution

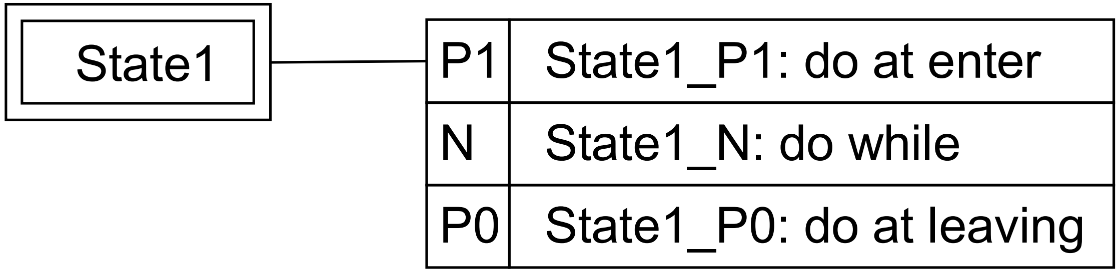
token switch : the token crosses the first active transition (at random if both Ea and Eb are true)

token forking : when the transition Ee is true, the token is replicated to all connected states

token join : when all tokens are present, and the transition Ef is true, one single token is forwarded.



SFC: P1, N and P0 actions



P1 (pulse raise) action is executed once when the state is entered
P0 (pulse fall) action is executed once when the state is left
N (non-stored) action is executed continuously while the token is in the state

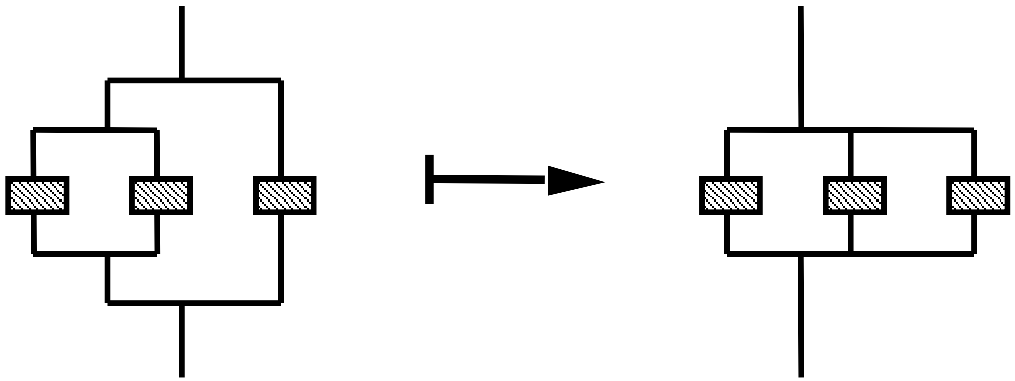
P1 and P0 actions could be replaced by additional states.

The actions are described by a code block written e.g. in Structured Text.

SFC: graphic rules

The input and output flow of a state are always in the same vertical line (simplifies structure)

Alternative paths are drawn such that no path is placed in the vertical flow
(otherwise would mean this is a preferential path)



intentional displacement to avoid optical preference of a path.

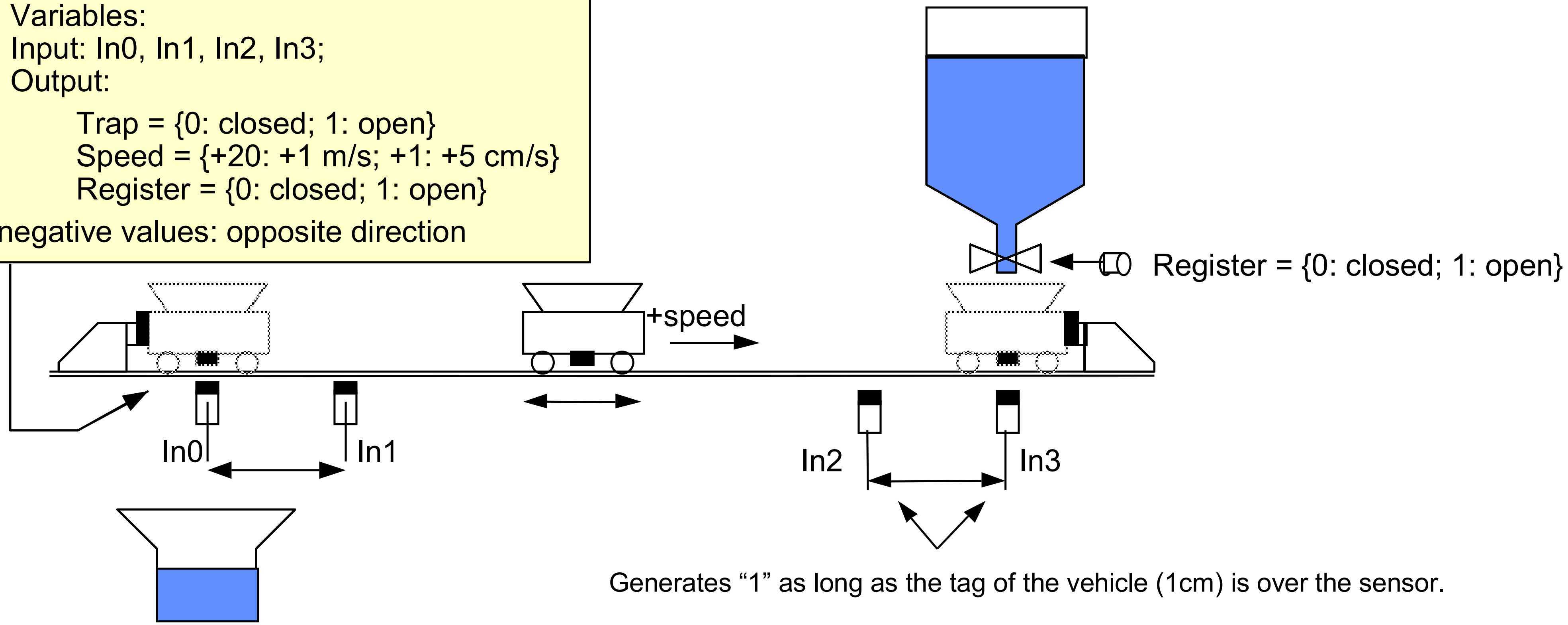
Priority:

- The alternative path most to the left has the highest priority, priority decreases towards the right.

- Loop: exit has a higher priority than loopback.

SFC: Exercise

Variables:
Input: In0, In1, In2, In3;
Output:
Trap = {0: closed; 1: open}
Speed = {+20: +1 m/s; +1: +5 cm/s}
Register = {0: closed; 1: open}
negative values: opposite direction

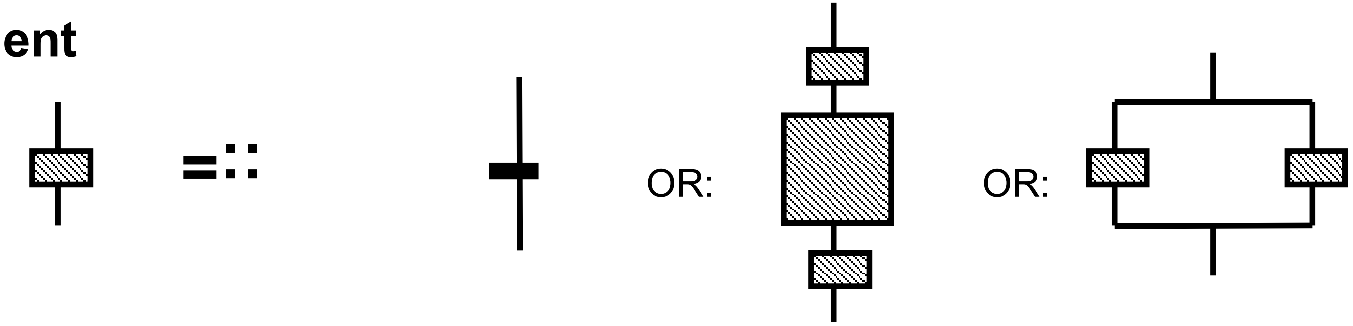


initially: let vehicle until it touches I0 at reduced speed and open the trap for 5s (empty the vehicle).
Speed = 5 cm/s between I0 and I1 or between I2 and I3, speed = 1 m/s between I1 and I2.

1 - Let the vehicle move from I0 to I3
2 - Stop the vehicle when it reaches I3.
3 - Open the tank during 5s.
4- Go back to I0
5 - Open the trap and wait 5s.
repeat above steps indefinitely

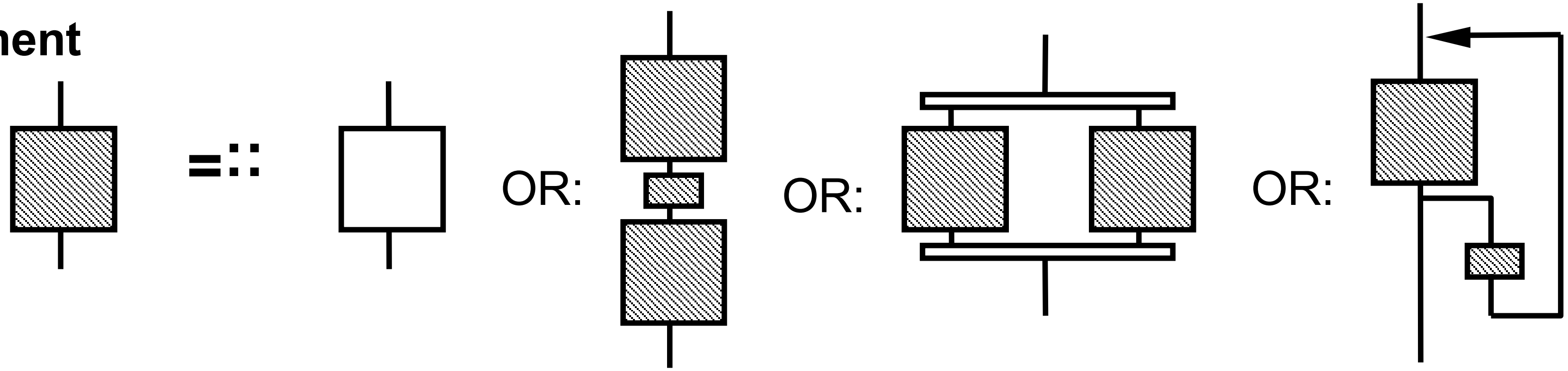
SFC: Building subprograms

T-element



transition T-sequence alternative paths

S-element



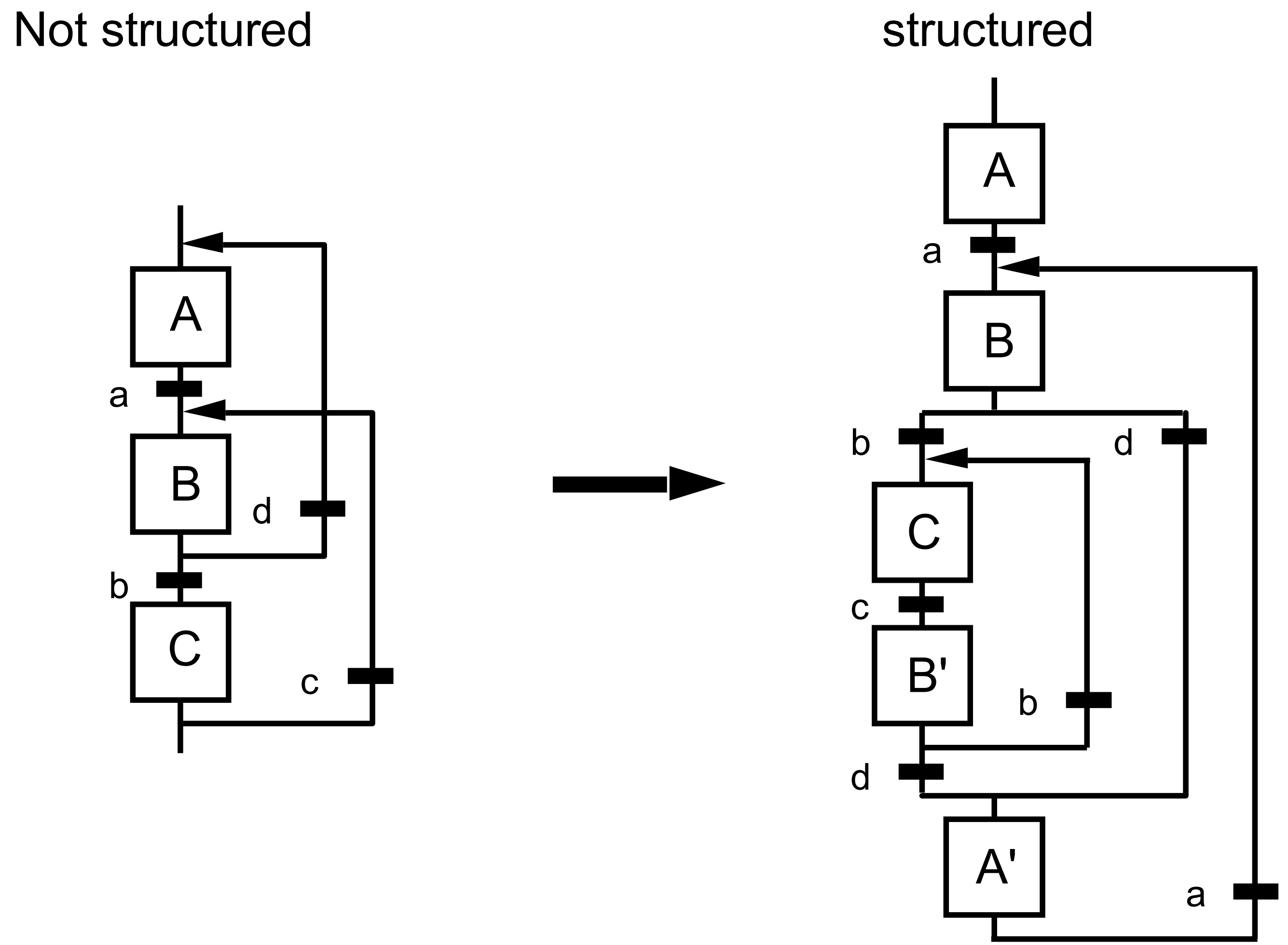
state S-sequence parallel paths loop

The meta-symbols T and S define structures - they may not appear as elements in the flow chart.

A flow chart may only contain the terminal symbols: state and transition

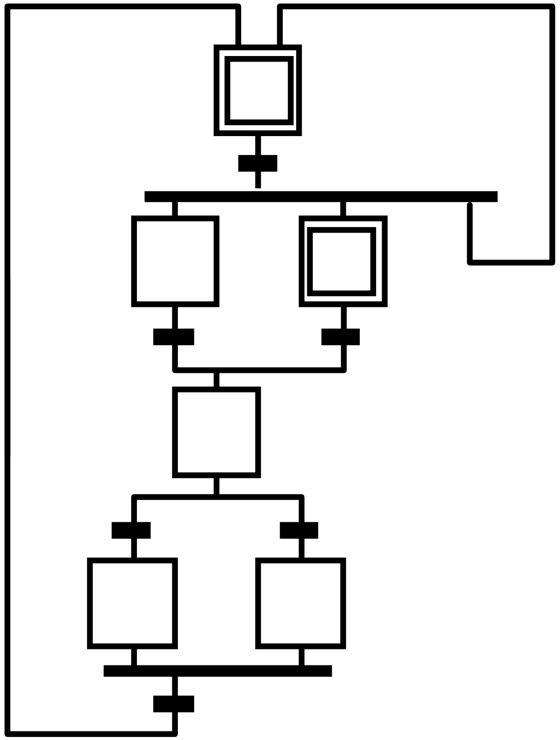
SFC: Structuring

Every flow chart without a token generator may be redrawn as a structured flow chart (by possibly duplicating program parts)



SCF: Complex structures

These general rules serve to build networks, termed by DIN and IEC as *flow charts*



Problems with general networks:
deadlocks
uncontrolled token multiplication

Solution:
assistance through the flow chart editor.

Function blocks And Flow Chart

Function Blocks:

Continuous (time) control

Sequential Flow Charts:

Discrete (time) Control

Many PLC applications mix continuous and discrete control.

A PLC may execute alternatively function blocks and flow charts.

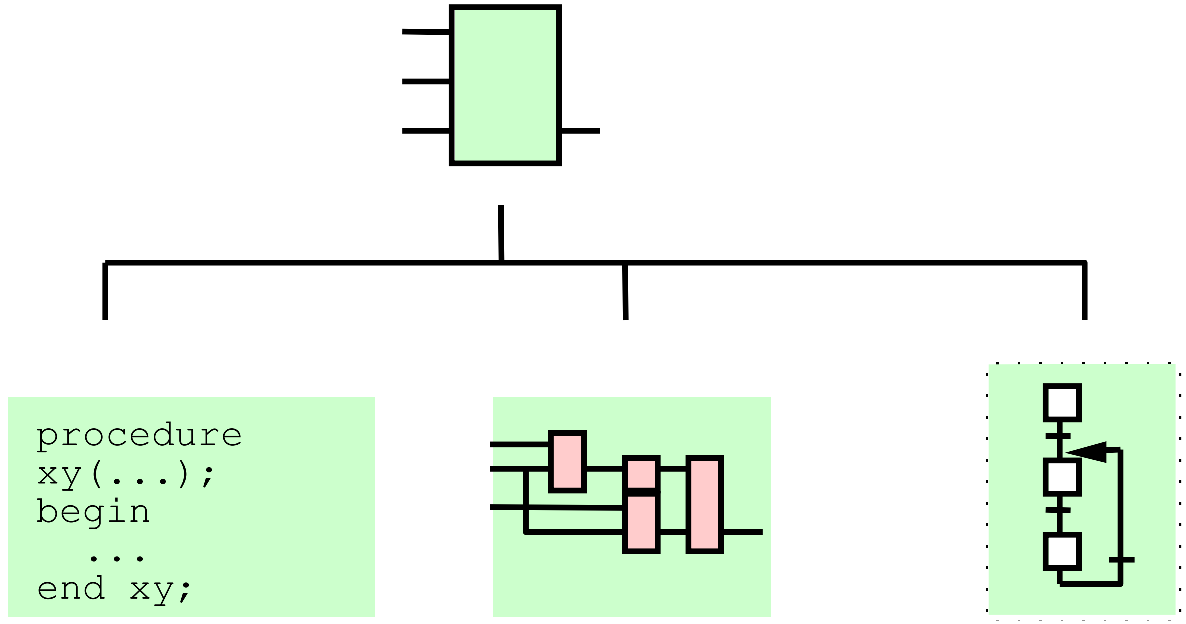
A communication between these program parts must be possible.

Principle:

> The flow chart taken as a whole can be considered a function block with binary inputs (transitions) and binary outputs (states).

Executing Flow Charts As blocks

A function block may be implemented in three different ways:



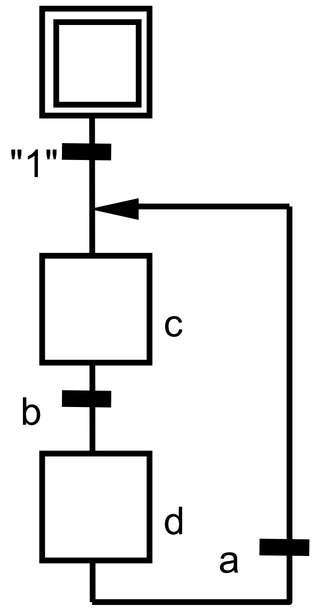
extern (ST) function blocks flow chart

Function blocks and flow chart communicate over binary signals.

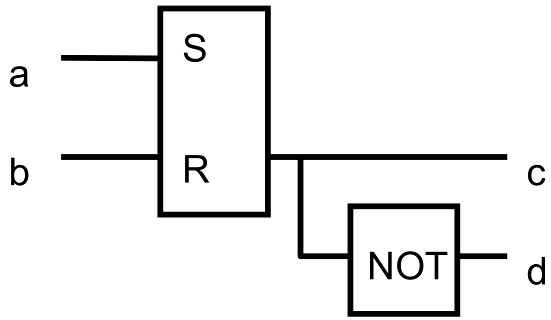
Flow Charts Or Function blocks ?

A task can sometimes be written indifferently as function blocks or as flow chart.
The application may decide which representation is more appropriate:

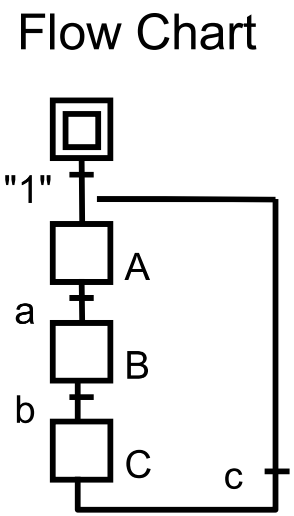
Flow Chart

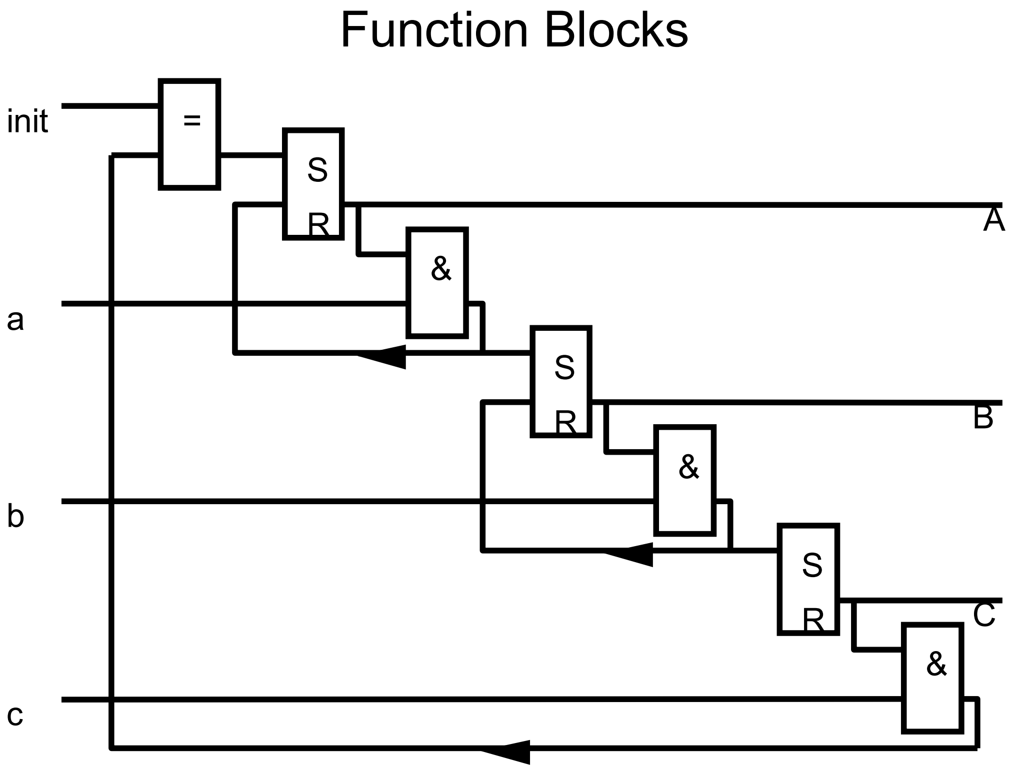


Function Block



Flow Charts Or Blocks ? (2)





In this example, flow chart seems to be more appropriate:

2.3.5.7 Ladder Logic

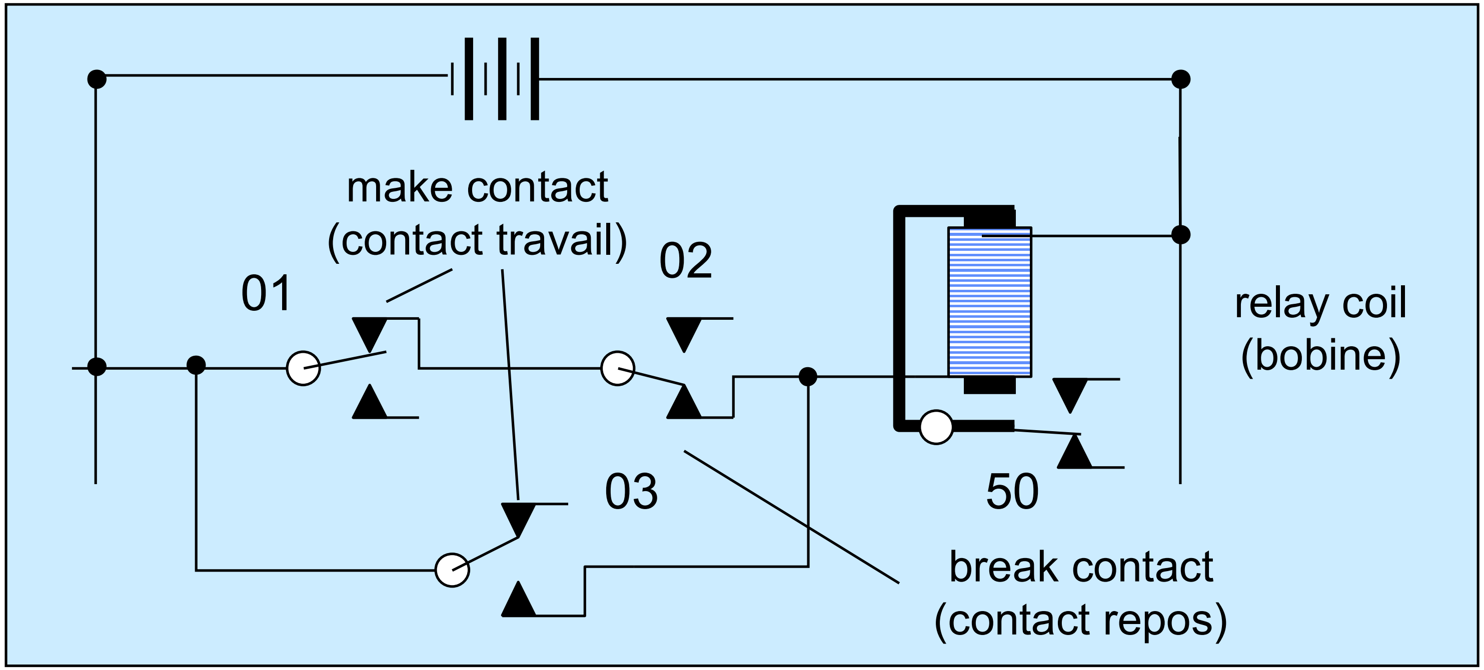
Ladder logic (1)

(*Kontaktplansprache*, langage à contacts)

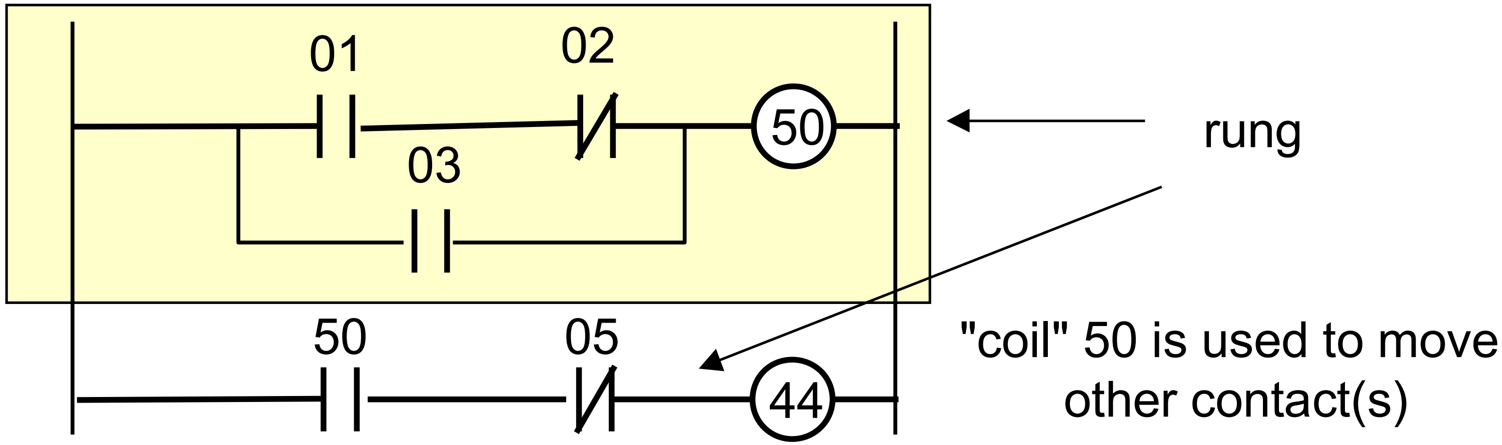
The ladder logic is the oldest programming language for PLC
it bases directly on the relay intuition of the electricians.
it is widely in use outside Europe.
It is described here but not recommended for new projects.

Ladder Logic (2)

origin:
electrical
circuit



corresponding
ladder diagram



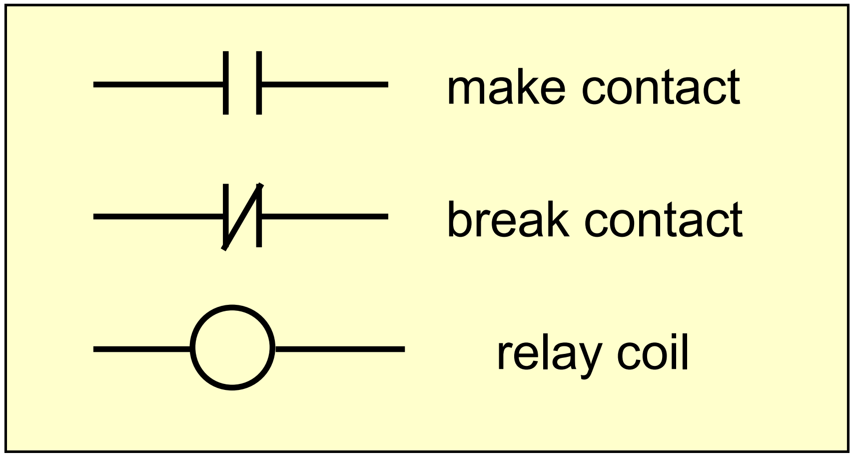
Ladder logic (3)

The contact plan or "ladder logic" language allows an easy transition from the traditional relay logic diagrams to the programming of binary functions.

It is well suited to express combinational logic

It is not suited for process control programming (there are no analog elements).

The main ladder logic symbols represent the elements:



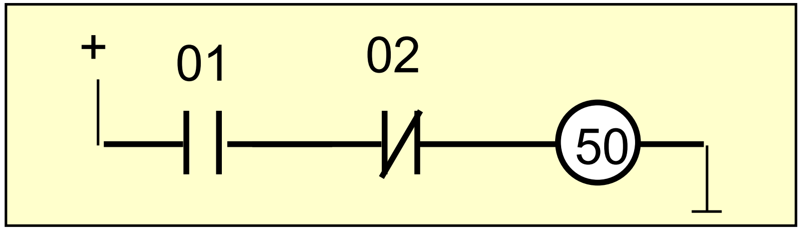
Ladder logic (4)

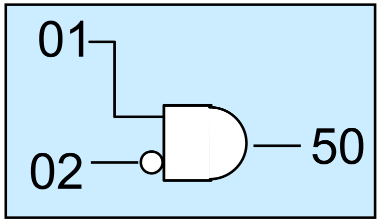
Binary combinations are expressed by series and parallel relay contact:

ladder logic representation

"CMOS" equivalent

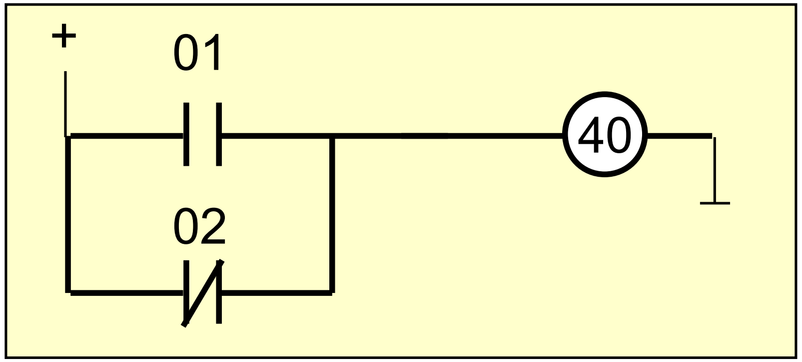
Series

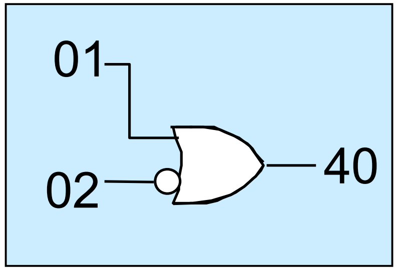




Coil 50 is active (current flows) when 01 is active and 02 is not.

Parallel

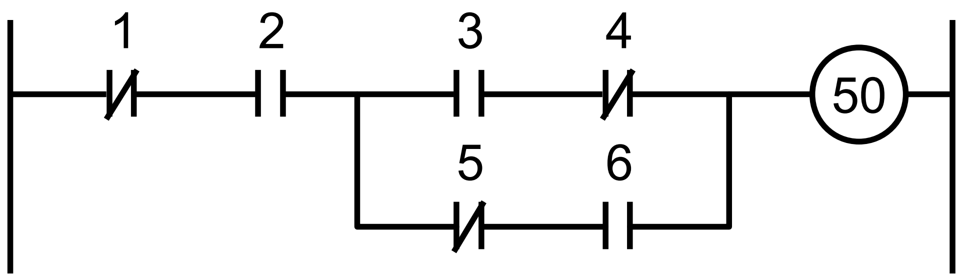




Coil 40 is active (current flows) when 01 is active or 02 is not.

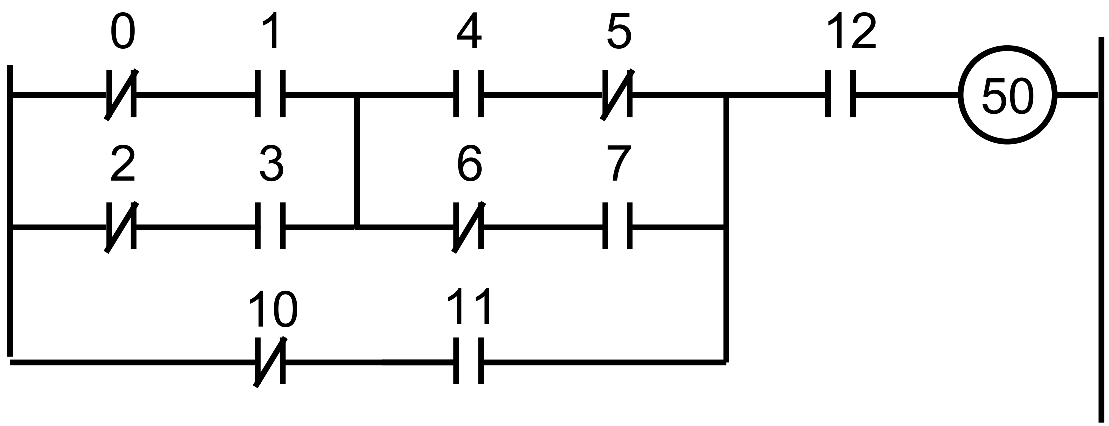
Ladder logic (5)

The ladder logic is more intuitive for complex binary expressions than literal languages



textual expression

!N 1 & 2 STR 3 & N 4 STR N 5 & 6 / STR & STR = 50



!0 & 1 STR 2 & 3 / STR STR 4 & 5 STR N 6 & 7

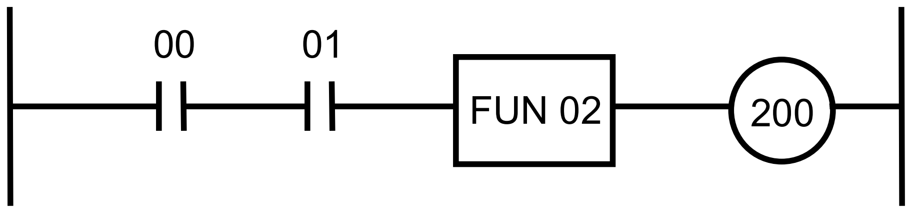
/ STR & STR STR 10 & 11 / STR & 12 = 50

Ladder logic (6)

Ladder logic stems from the time of the relay technology.

As PLCs replaced relays, their new possibilities could not be expressed any more in relay terms.

The contact plan language was extended to express functions:



literal expression:

!00 & 01 FUN 02 = 200

The intuition of contacts and coil gets lost.

The introduction of «functions» that influence the control flow itself, is problematic.

The contact plan is - mathematically - a functional representation.

The introduction of a more or less hidden control of the flow destroys the freedom of side effects and makes programs difficult to read.

Ladder logic (7)

Ladder logic provides neither:
• sub-programs (blocks), nor
• data encapsulation nor
• structured data types.

It is not suited to make reusable modules.

IEC 61131 does not prescribe the minimum requirements for a compiler / interpreter such as number of rungs per page nor does it specifies the minimum subset to be implemented.

Therefore, it should not be used for large programs made by different persons

It is very limited when considering analog values (it has only counters)

→ used in manufacturing, not process control

2.3.6 Instruction Lists

Instruction Lists (1)

(*Instruktionsliste*, liste d'instructions)

Instruction lists is the machine language of PLC programming
It has 21 instructions (see table)

Three modifiers are defined:
"N" negates the result
"C" makes it conditional and
"(" delays it.

All operations relate to one result register (RR) or accumulator.

Operator	Modifier	Description
LD	N	Loads operand in RR
ST	N	Stores current result from RR
S		Sets the operand
R		Resets the operand
AND	N, (	Boolean AND
OR	N, (	Boolean OR
XOR	N, (	Exclusive OR
ADD	(	Arithmetic addition
SUB	(	Arithmetic subtraction
MUL	(	Arithmetic multiplication
DIV	(	Arithmetic division
GT	(	Comparison greater than
GE	(	Comparison greater than or equal to
EQ	(	Comparison equal
LE	(	Comparison less than
LT	(	Comparison less than or equal to
NE	(	Comparison not equal
)		Executes delayed operation
CAL	C, N	Calls a function block
JMP	C, N	Jumps to label
RET	C, N	Returns from called function

Instruction Lists Example (2)

Label	Operator	Operand	Comment
	LD	temp1	(*Load temp1 and*)
	GT	temp2	(*Test if temp1 > temp2*)
	JMPCN	Greater	(*Jump if not true to Greater*)
	LD	speed1	(*Load speed1*)
	ADD	200	(*Add constant 200*)
	JMP	End	(*Jump unconditional to End*)
Greater:	LD	speed2	(*Load speed2*)

Instructions Lists is the most efficient way to write code, but only for specialists.

Otherwise, IL should not be used, because this language:
- provides no code structuring
- has weak semantics
- is machine dependent

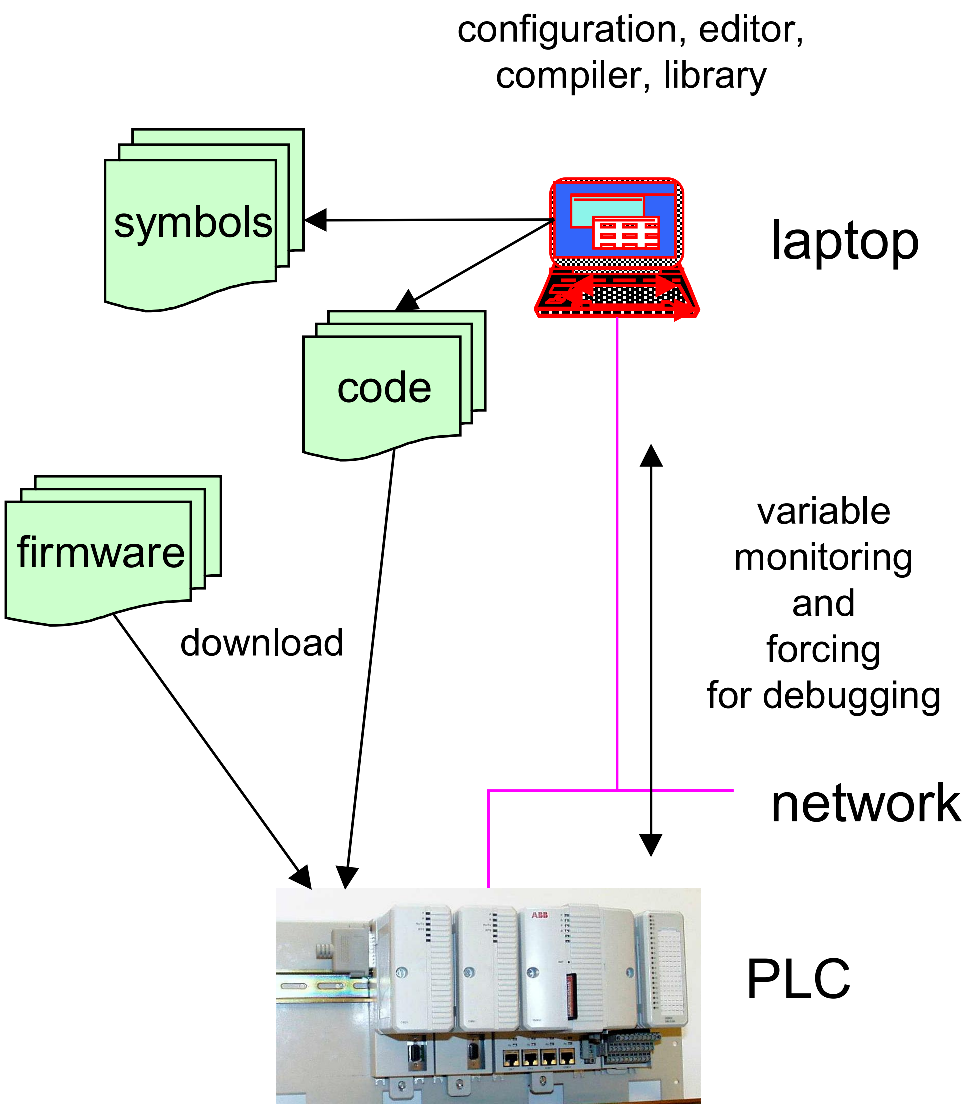
2.3.5.9 Programming environment

Programming environment capabilities

A PLC programming environment (ABB, Siemens, CoDeSys,...) allows:

- programming of the PLC in one of the IEC 61131 languages
- defining the variables (name and type)
- binding of the variables to the input/output (binary, analog)
- simulating
- downloading to the PLC of programs and firmware
- uploading of the PLC (seldom provided)
- monitoring of the PLC
- documenting and printing.

61131 Programming environment



Program maintenance

The source of the PLC program is generally on the laptop of the technician.

This copy is frequently modified, it is difficult to track the original in a process database, especially if several persons work on the same machine.

Therefore, it would be convenient to be able to reconstruct the source programs out of the PLC's memory (called back-tracking, *Rückdokumentation*, reconstitution).

This supposes that the instruction lists in the PLC can be mapped directly to graphic representations -> set of rules how to display the information.

Names of variables, blocks and comments must be kept in clear text, otherwise the code, although correct, would not be readable.

For cost reasons, this is seldom implemented.

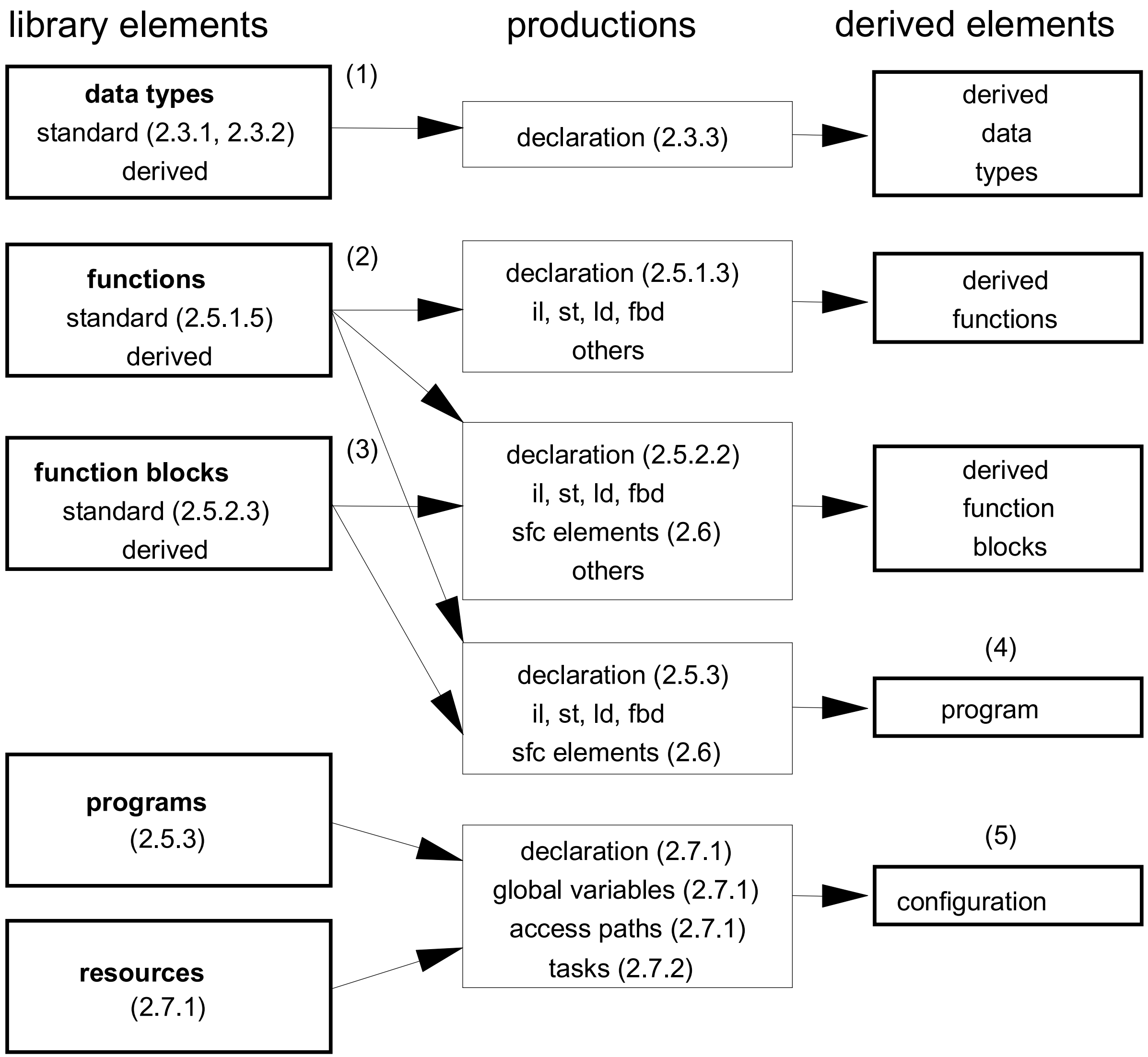
Is IEC 61131 FB an object-oriented language ?

Not really: it does not support inheritance.

Blocks are not recursive.

But it supports interface definition (typed signals), instantiation, encapsulation, some form of polymorphism.

IEC 61131-3 elements



Assessment

Which are programming languages defined in IEC 61131 and for what are they used ?

In a function block language, which are the two elements of programming ?

How is a PLC program executed and why is it that way ?

Draw a ladder diagram and the corresponding function chart.

Draw a sequential chart implementing a 2-bit counter

Program a saw tooth waveform generator with function blocks

How are inputs and outputs to the process treated in a function chart language ?

Program a sequencer for a simple chewing-gum coin machine

Program a ramp generator for a ventilator speed control (soft start and stop in 5s)

Limitations of IEC 61131

- it is not foreseen to distribute execution of programs over several devices

- event-driven execution is not foreseen. Blocks may be triggered by a Boolean variable,
  (but this is good so).

EPFL