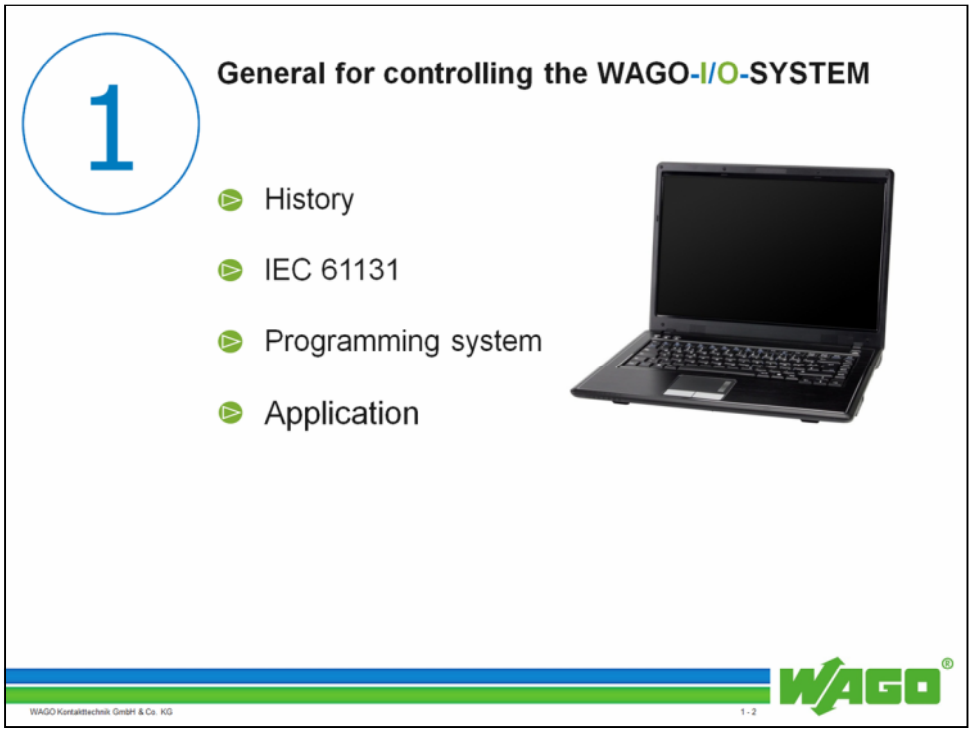

WAGO
e!COCKPIT
Fundamentals
WE
INNOVATE!

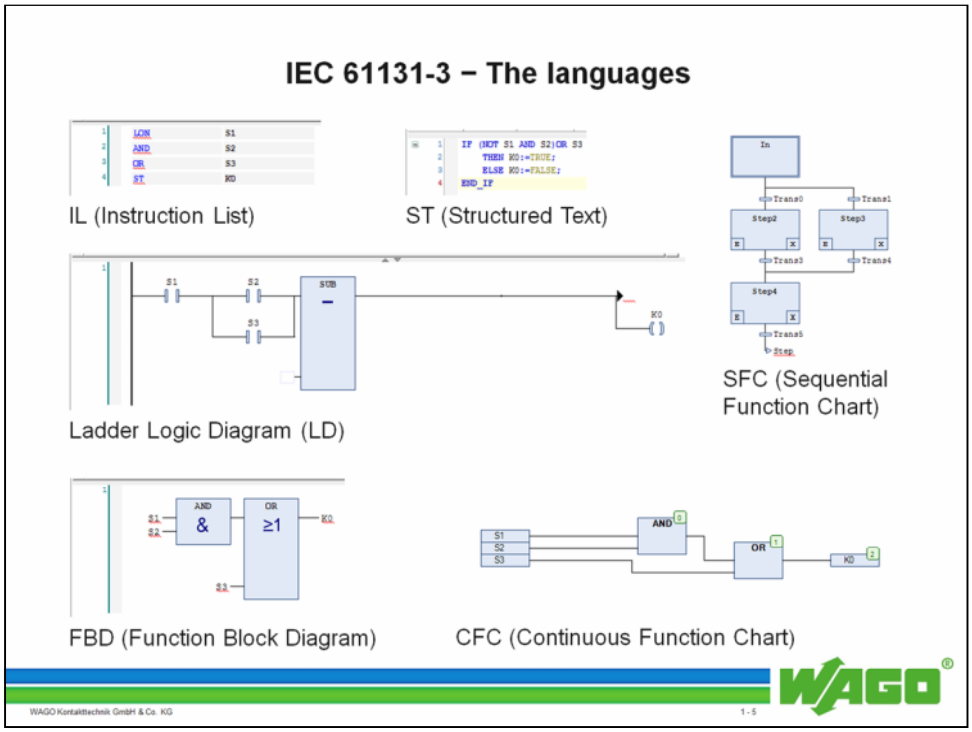


Contents

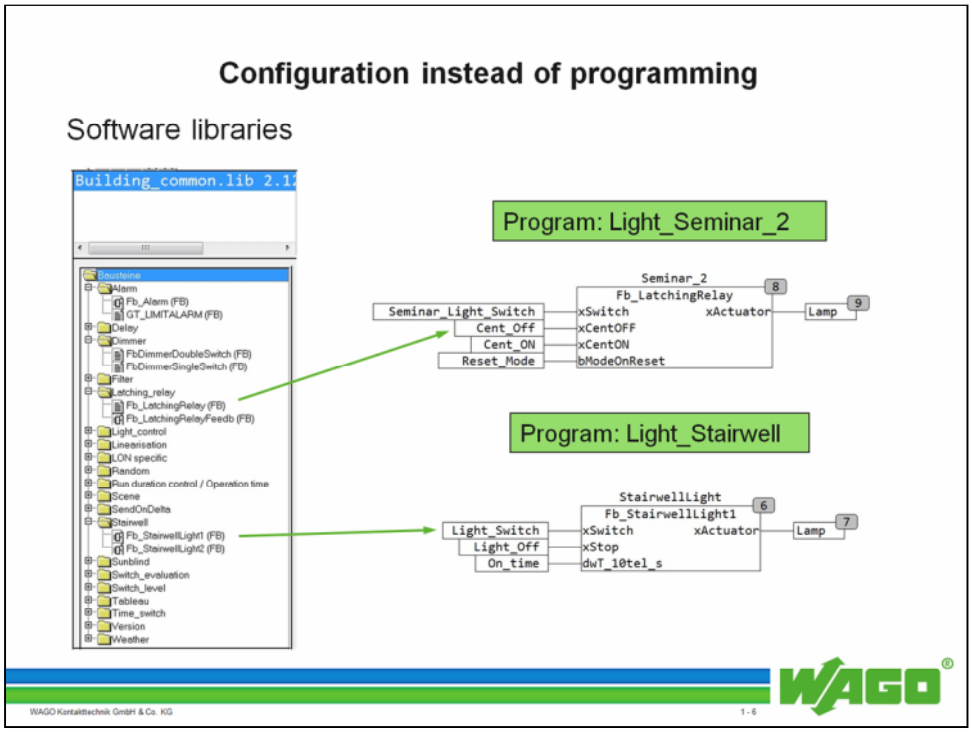
History

Programmable Logic Controllers

• 1965 ... 70	In the US, the automotive industry encouraged the development of a new type of control for converting transfer lines (PLC-Programmable Logic Controllers)
• Beginning in 1974	These controllers have been used in Germany as well (PLC - Programmable Logic Controller)
• 1983	DIN 19239
• 1993	DIN IEC 61131-3
• Today	Large selection of manufacturers. IEC 61131 contains definitions of requirements for modern PLC systems in order to counteract language elements and provide a uniform program structure independent of programming systems.



English		German		
abbr.	Designation	abbr.	Designation	Remarks
IL	Instruction List	**AWL**	Instruction list	Comparable to the assembler
LD	Ladder Diagram	**KOP**	Ladder diagram	Comparable to a circuit diagram (in exploded view) that has been rotated 90°.
FBD	Function Block Diagram	**FBS (FUP)**	Function Block-Language	In part (in particular for Siemens STEP 7) also known as LD (Logic Diagram).
SFC	Sequential Function Chart	**AS**	Sequential function chart	a type of status diagram in STEP 7 known as S7 GRAPH. The IEC 61131-3: 2003 sees the SFC as an evolution of Grafcet according to EN 60848.
ST	Structured Text	**ST**	Structured text	Derivation of the high level language PASCAL, designated as SCL (Structured Control Language) for STEP 7.
CFC	Continuous Function Chart (Only mentioned in IEC 61131-3)	**CFC**	Continuous function chart	In CFC, the function blocks (which are identical with those in FBD) are freely positioned.



There are a number of libraries in different areas, the various components which are already finished.

Controllers

e!*COCKPIT* currently supports the following controllers

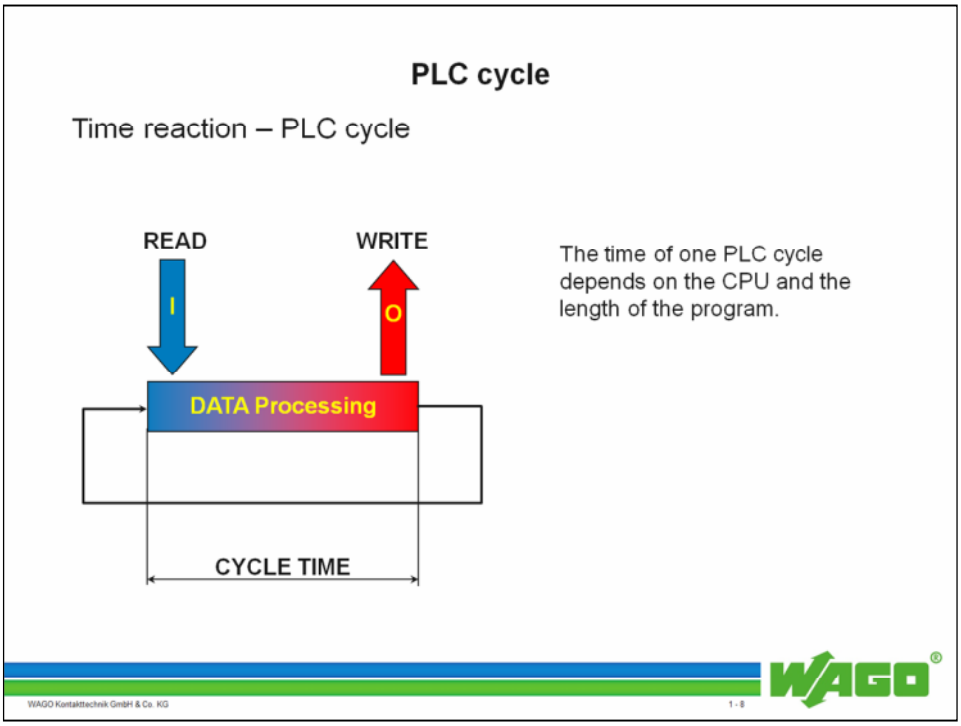
PFC200 750-8202

PFC200 750-8203

PFC200 750-8204

PFC200 750-8206

Programming according to IEC 61131-3



Like most PLCs, a WAGO controller also functions using the IPO principle. It has an input part, a processing part, and an output part.

The data from connected sensors and actuators are connected to the controller using Input/output devices.

The controller functions cyclically: It reads the values of all the inputs at the beginning of a cycle (“read process image” is also used in conjunction with this).

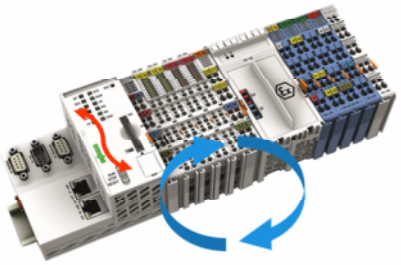
Afterwards, all program instructions are (sequentially) processed and the assigned outputs are written at the end of the cycle into the output components.

To optimize processing, reading and writing the process data is carried out in one step at the beginning of the cycle.

Application

Stand alone vs. networked controllers

- PLC can control autonomous units
- Thanks to the networking features many PLC can be connected to create larger systems



As a system partner, WAGO solves all automation tasks using centralized and decentralized control architecture in all sectors of industrial, process, and building automation.

The IEC 61131 programming standard is thus an important guideline for modular automation components.

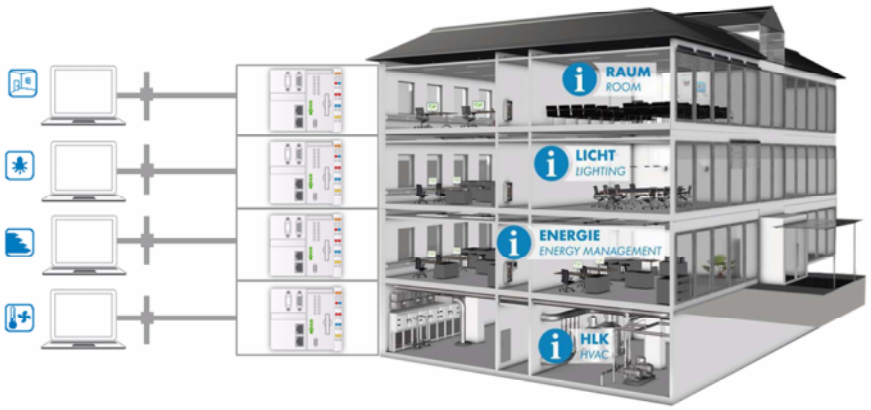
Integrated Web pages and Web-based visualization provide IT applications with real-time process data.

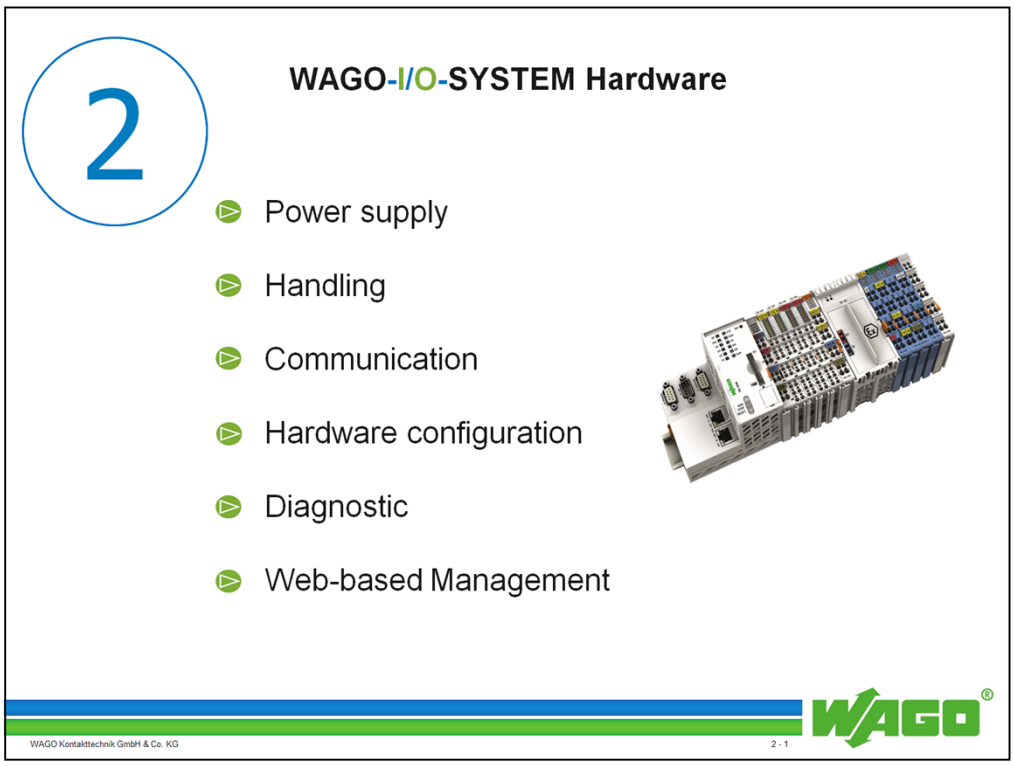
Both large memory and an integrated multitasking system readily meet stringent automation requirements.

A large number of library functions support both software/hardware interfaces and integrated file system.

Application

Decentralized intelligence with networked controllers





WAGO-I/O-SYSTEM 750, 753, 758 for scalable automation solutions

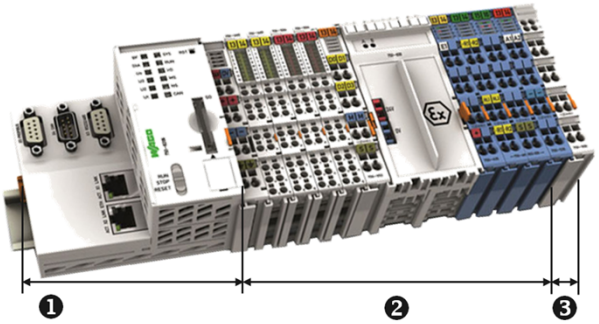
Fine modularity and fieldbus-independence are hallmarks of the WAGO-I/O-SYSTEM, which boasts worldwide approvals for a diverse range of applications. During development, great care was taken to ensure the system could account for all the requirements placed on decentralized fieldbus systems.

Designed to Meet Practical Requirements

- I/O modules with pluggable connectors (753 Series)
- Fine granularity: I/O module accommodates 1, 2, 4, 8 or 16 channels
- Fieldbus-independent: Fieldbus couplers and controllers for the most common fieldbus protocols and industrial ETHERNET standards
- A sound investment: Fieldbus-independent node design easily accommodates new bus standards while retaining the I/O modules
- Clear Identification: Color-coded group marker carriers and WAGO WSB markers for clamping units
- Scalable performance: With economy and standard couplers as well as programmable controllers on through to comprehensive IPCs
- High-performance: Controllers for distributed control networks according to IEC 61131-3

Fielbus nodes

The configuration



1. Controller
2. I/O modules (max. 64)
3. End module

Total extension

The length of the module assembly that can be connected to the controller is 780 mm. The width of the end module is 12 mm. When assembled, the I/O modules have a maximum length of 768 mm.

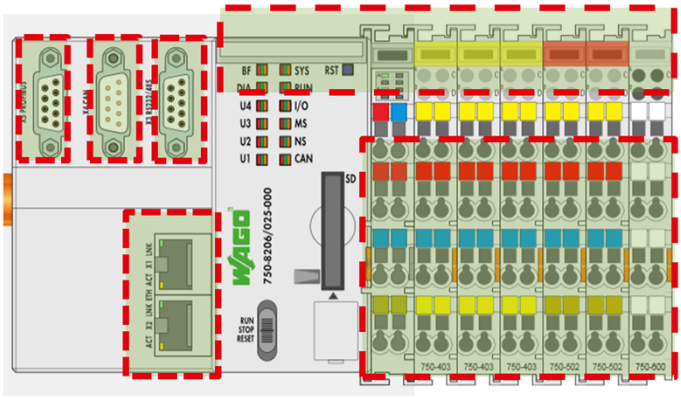
Examples:

64 I/O modules of 12 mm width can be connected to a controller.
32 I/O modules of 24 mm width can be connected to a controller.

.

Power Supply

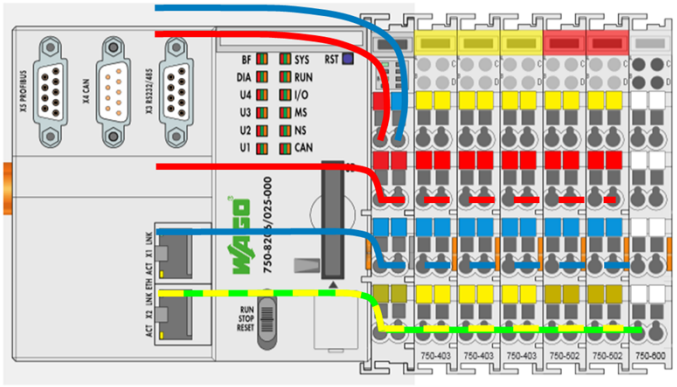
The potential levels



- Fieldbus, external interfaces
- Module electronics
- External peripheries

Power Supply

Supply with electrical isolation



Bus coupler and Bus modules electronics:

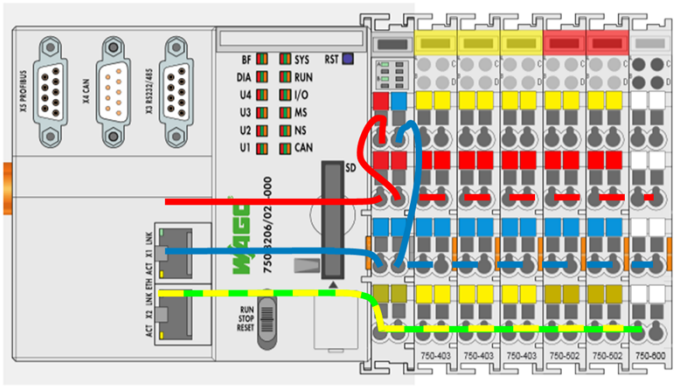
- external: 24 VDC, max. 500 mA
- internal: 5 VDC, 2000 mA

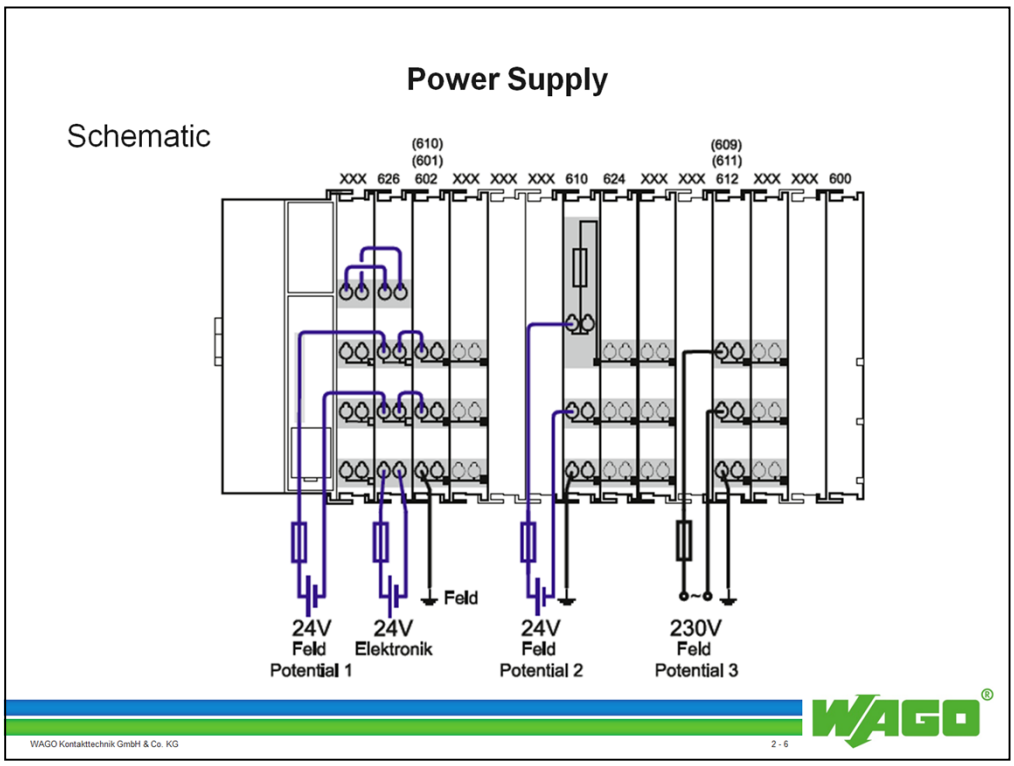
External peripheries:

- 24 VDC, max. 10 A

Power Supply

Supply without electrical isolation





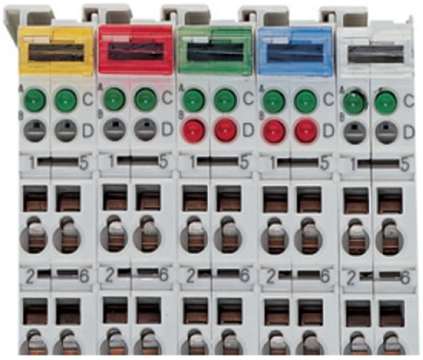
3.6.5 Supply example

System and field supply must be isolated!
The system supply and field supply must be isolated to ensure bus operation in the event of short circuits on the actuator side.

Additional information for designing a ring feeding
In order to increase system safety, a ring feeding of the ground potential is recommended. Thus, the ground potential is maintained, in the event that a bus terminal is pulled from the potential group.
In ring feeding, the grounding conductor is connected to the beginning and end of a potential group.
Please see the additional information for designing a ring feeding in chapter "Grounding"> "protective earth" ring feeding.

Color-coding

yellow	Digital inputs
red	Digital outputs
green	Analog inputs
blue	Analog outputs
	Specialty functions

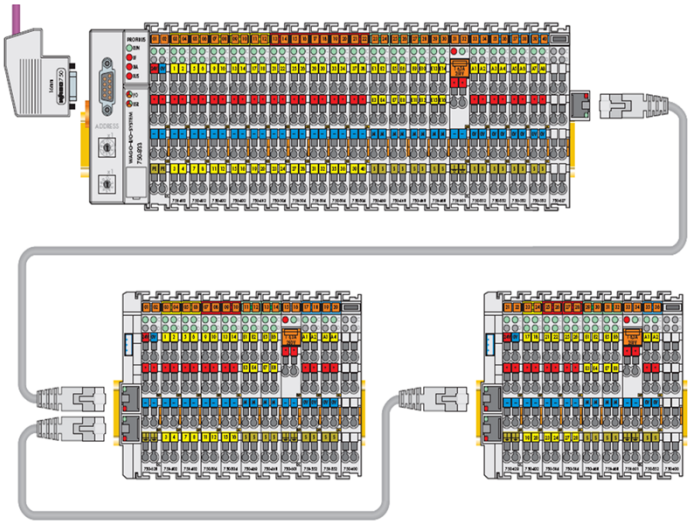


750 – 4xx inputs

750 – 5xx outputs

750 – 6xx Special and system modules

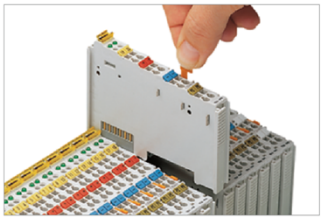
Internal Data Bus Extension



Handling the I/O modules

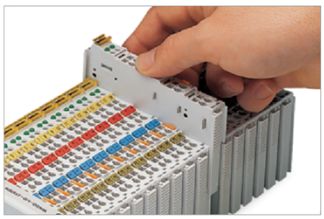
Assembly/Removal

- Assembly: Insert until the module engages noticeably
- Removal: Pull on the orange actuating element
- Mount terminals if free of voltage



Practical tips

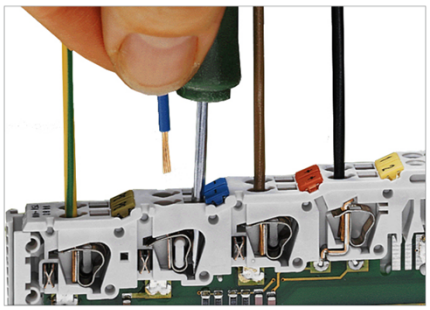
- Be careful with the blade contacts when removing the terminal (risk of injury)
- Keep gold contacts clean and do not touch with fingers
- Deposit terminal with the printed side up



Wiring the I/O modules

CAGE CLAMP® Connection

- A universal system
- Suitable for all copper wires from 28 to 2 AWG (0.08 mm² to 2.5 mm²)
- One conductor per clamping unit
- Clamping of the wire without damage through unique design
- Gastight contact area between conductor and current bar
- Vibration and shock resistant
- Simple, easy-to-use design
  Fast and maintenance-free

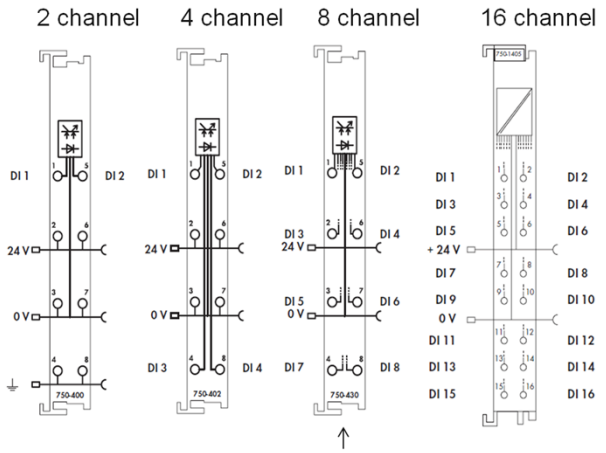


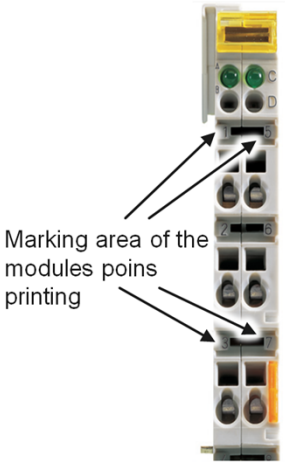
c UL us	c UL us	E175199, UL 508
ABS	ABS (American Bureau of Shipping)	03-HG374860/2-PDA; 05-ES578983-X
BV	BV (Bureau Veritas)	13453/B0 BV
DNV	DNV (Det Norske Veritas)	A-12260; Cl. B
GL	GL (Germanischer Lloyd)	11 631-10 HH; 26 624-05 HH; 26 898-05 HH; 59 627-08 HH; 60 241-09 HH; Cat. A, B, C, D (EMC 1)
KR	KR (Korean Register of Shipping)	HMB05880-EL004 ff
LR	LR (Lloyd's Register)	02/20026 (E2); Env. 1, 2, 3, 4
NKK	NKK (Nippon Kaiji Kyokai)	TA06190M
PRS	Polski Rejestr Statkow	TE/1720/880590/08
RINA	RINA (Registro Italiano Navale)	ELE153207CS 001
Ex	c UL us	E198726, ANSI/ISA 12.12.01
	DEMKO, PTB	08ATEX142851 X; IECEx PTB 07.0064 X
	Brazilian Ex	MC,AEX-7538-X (OCP 0004)
	TÜV	07ATEX554086 X; IECEx TUN 09.0001 X

	Bestellnr.	Artikelbezeichnung	Ex		Schiffszulassungen										UL
Digitale Eingangsklemmen	750-400	2 DI 24 V DC, 3,0 ms			x	x*	x	x	x	x	x	x*	x*	x	x*
	750-401	2 DI 24 V DC, 0,2 ms			x	x*	x	x	x	x	x	x*	x*	x	x*
	750-402	4 DI 24 V DC, 3,0 ms			x	x*	x	x	x	x	x	x*	x*	x	x*
	750-403	4 DI 24 V DC, 0,2 ms	x		x	x*	x	x	x	x	x	x*	x*	x	x*
	750-405	2 DI 230 V AC			x	x*	x	x	x	x	x	x*	x*	x	x*
	750-406	2 DI 120 V AC			x	x*	x	x	x	x	x	x*	x*	x	x*
	750-408	4 DI 24 V DC, 3,0 ms, negativschaltend			x	x*									x*

Connect the peripherals

Special features of the digital modules

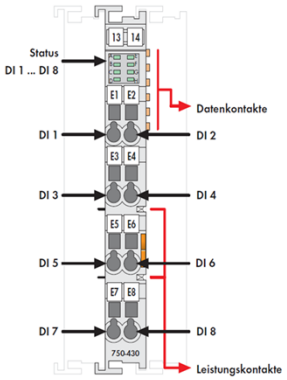


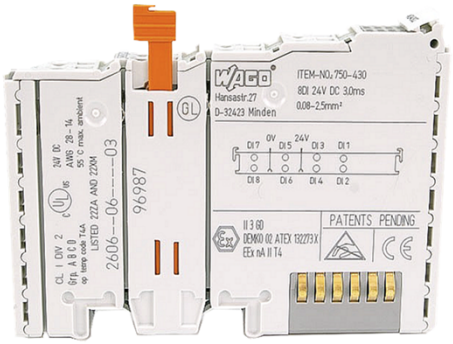


The marking area of the modules points printing
1-8 ***fits usually not to*** the channel occupancy

Connect the peripherals

Wiring information from the data sheet





Online or catalog or on the board

Configuration of the Controller

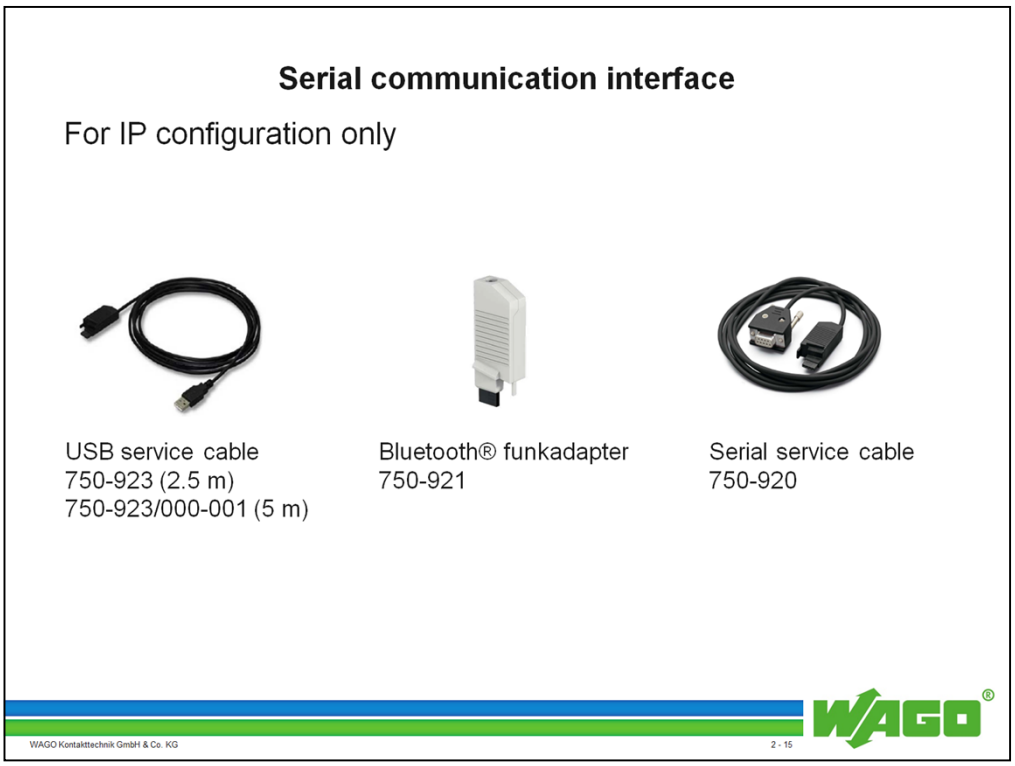
- TCP IP settings
- Runtime system
- Switch/ LED
- Further settings
- Technichal data

Communication

Connection between *e!COCKPIT* PC and Controller

- Network cable
  - Configuration
  - Program download
- USB service cable
  - Configuration

The communication between PC and controller takes place using standard network interface cards via Ethernet.



(From the WAGO-I/O-*CHECK* manual)

5.1 Configuring the communication connection

The following work steps are required to set up the communication connection:

- Switch off the power supply of the fieldbus controller.
- Open the configuration interface of the fieldbus controller.

Warning!
Do NOT touch the interface contacts of the fieldbus controller with your fingers or any conductive objects!

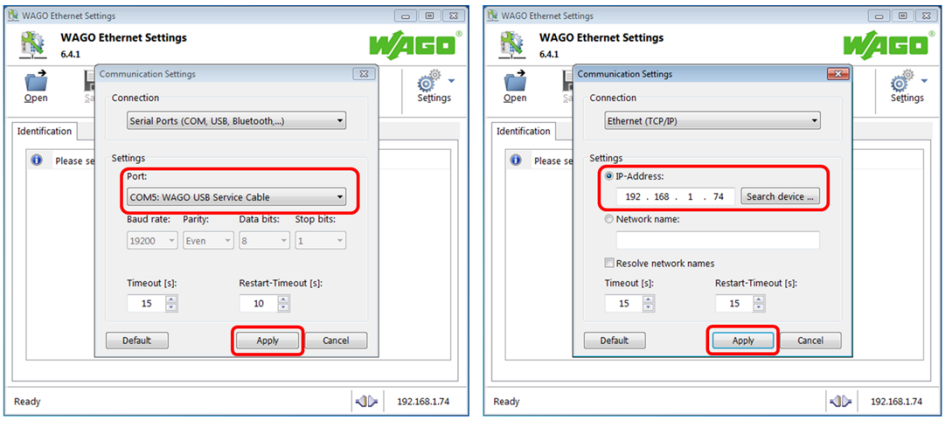
- Connect the configuration interface to the appropriate socket of the communication cable 750-920.
- Connect the Sub-D socket of the communication cable with a free serial interface of the computer.

WAGO-I/O-*CHECK* is now able to communicate with the node.

Hardware configuration – IP address

WAGO ETHERNET Settings

Two connection options

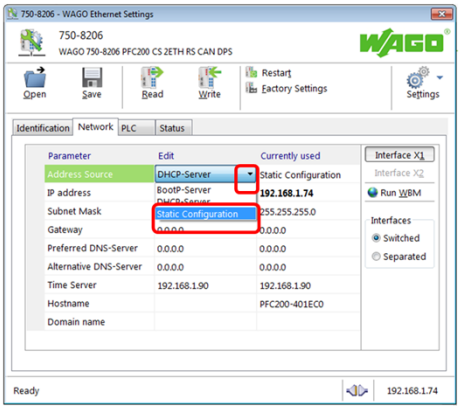


Via a USB service cable

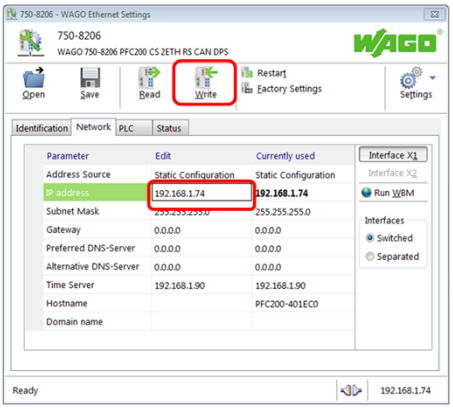
Over the network

Hardware configuration – IP address

WAGO ETHERNET Settings



Source: static configuration

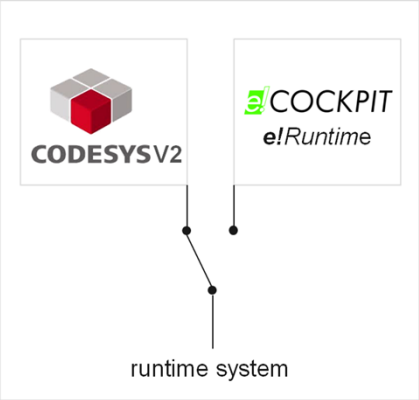


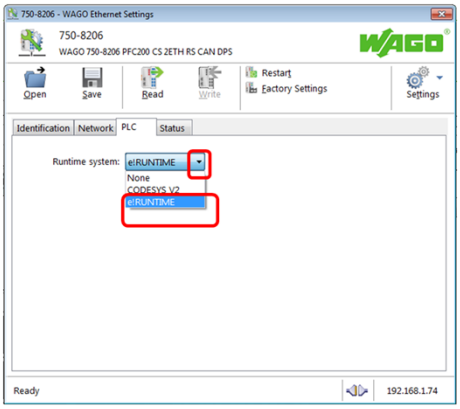
To set IP settings => to write

Hardware configuration – runtime system

WAGO ETHERNET Settings

! Don't forget
Switch the runtime system – from CODESYS V2 to ***e!****Runtime*





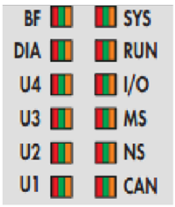
Operating mode switch

Position	Activation	Function
RUN	Switch RUN	**Application Start**
STOP	Switch STOP	**Application Stop**
RESET	Switch RESET > 2s Switch RESET > 7s	**Reset warm** **Reset cold**

! If the operating mode switch is in STOP position, the Controller can not be started from ***e!****COCKPIT*.

Diagnostic

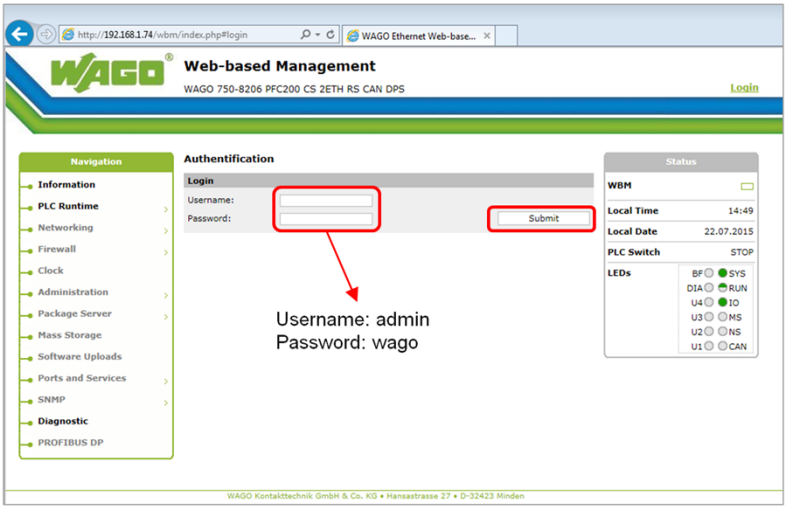
Diagnostic LEDs of PFC 200



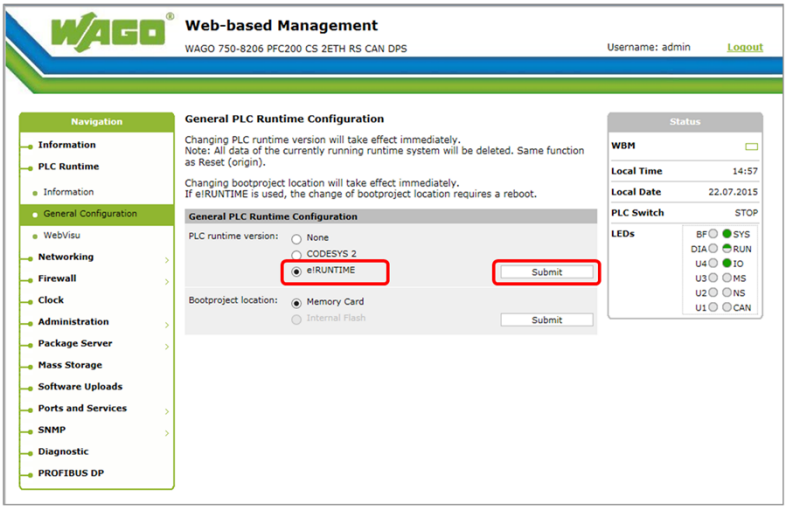
BF/DIA	Bus diagnostic PROFIBUS
CAN	Bus diagnostic CANopen
MS/NS	Bus diagnostic Ethernet
I/O	Diagnostic K-Bus
U1…U4	User diagnostic

LNK ACT Active network connection

Web-based Management

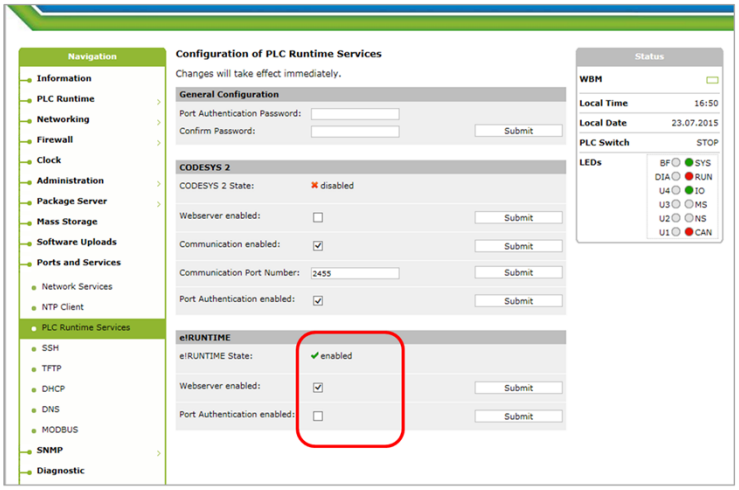


Web-based Management



Web-based Management – additional settings

Webserver enabled: for the using ***e!****COCKPIT* web visualization port authentication switches ***e!****COCKPIT* password on/ off (Default: admin/ wago)



Technical data – 8202 … 8206

ETHERNET type:	100 Mbit/s
CPU:	32 bit
Program memory:	16 Mbyte
Data memory:	64 Mbyte
Retain memory:	128 kbyte
Filesystem:	256 Mbyte

Data protocols: Modbus TCP/IP, Netzwerkvariable, CANopen, ProfibusDP
Service protocols : HTTP, BootP, DHCP, DNS, SNTP, FTP, SNMP

3

Software, Tools & Docs WAGO-I/O-SYSTEM

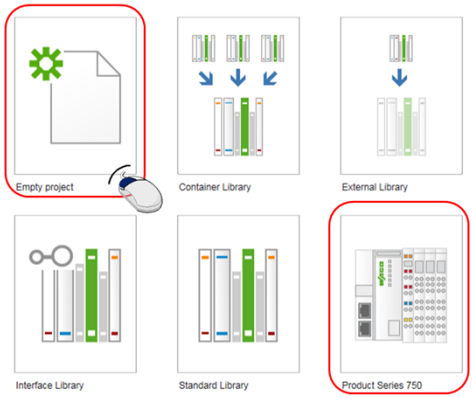
- *e!COCKPIT* functionality
- Create a project
- Software overview
- Configuration
- I/O-CHECK

e!COCKPIT functionality

Configuration	„I/O-*CHECK*“
Programming	Visualization

Create a project

You can start a project either at
Empty project or
Product Series 750.



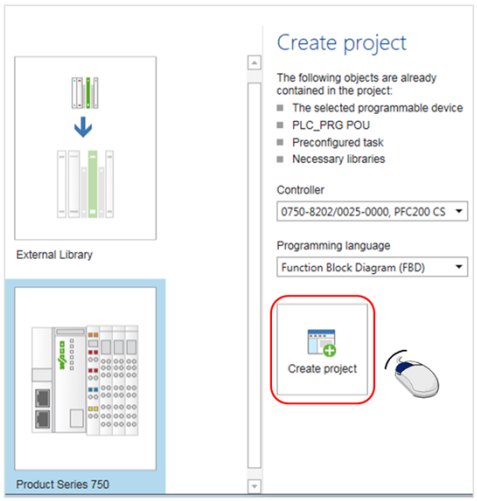
Create a project

Empty project:

Preferably, in online configuration or more controllers in the project.

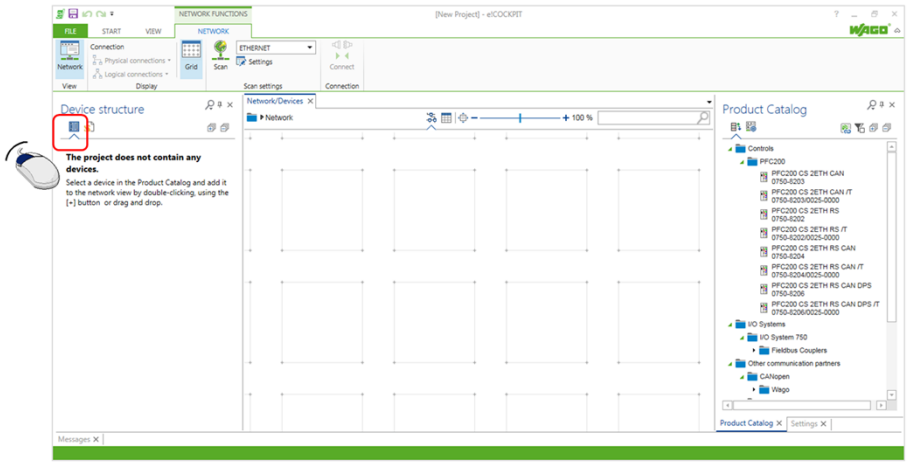
Product Series 750:

Preferably, in offline configuration with 1 controller/ project.



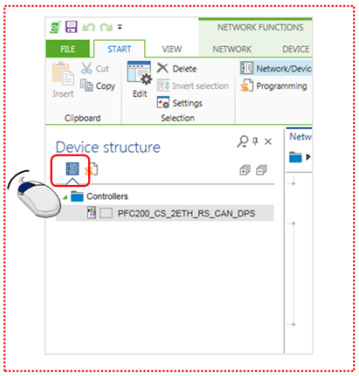
Software overview

After creating an empty project ***e!****COCKPIT* starts in the graphical Device structure.

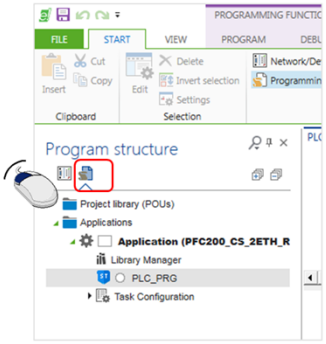


Device structure – Program structure

You can switch between the Device structure and Program structure with symbols on the left side of the screen

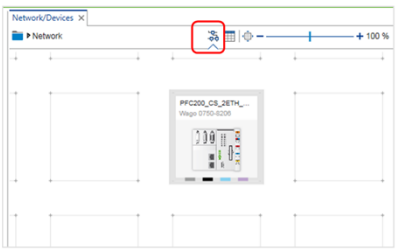


Device structure



Program structure

Alternative Device structure views



Graphical view

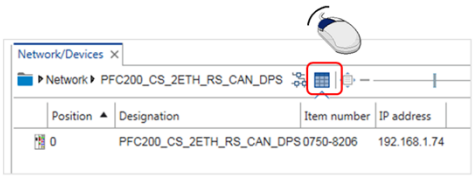


Table view

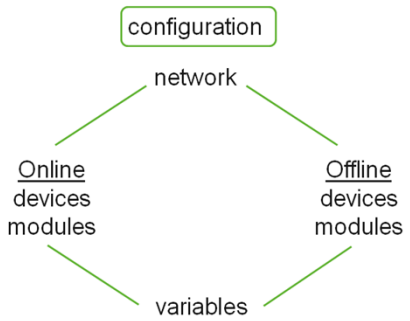
Configuration steps in *e!COCKPIT*

- Configure network
- Configure network nodes

Online

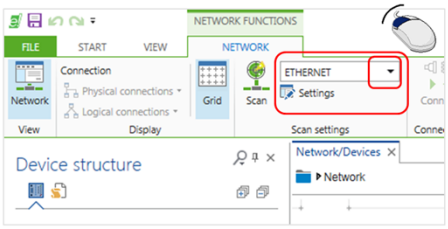
Offline

- Configure node modules (if necessary)
- Configure variables

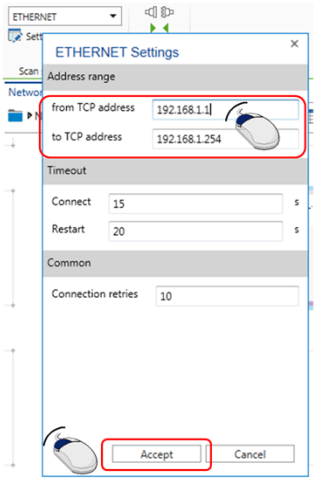


Configure network

Adjust the ETHERNET network settings according to your IP address range

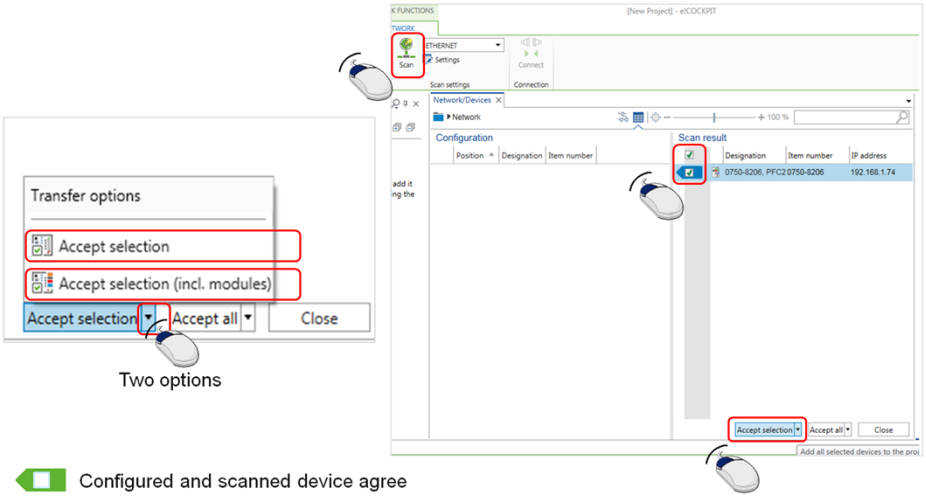


IP network settings



Online node configuration

Scanning the devices

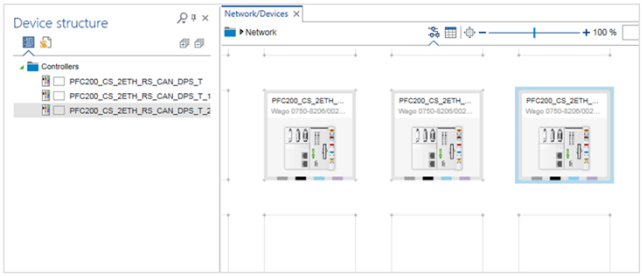


Configured and scanned device agree

New device is detected

Online node configuration

Configuration with controllers and modules



Graphical view

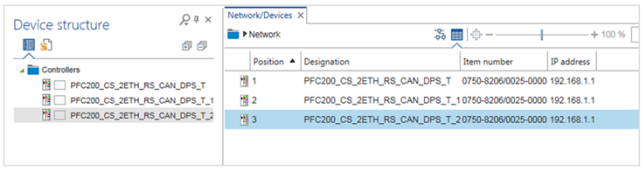
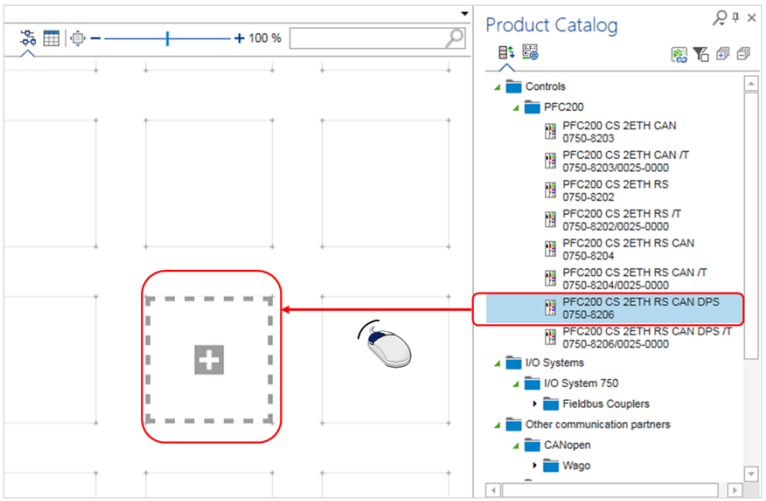


Table view

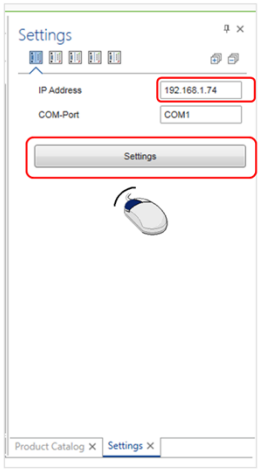
Offline node configuration

Insert devices from product catalog

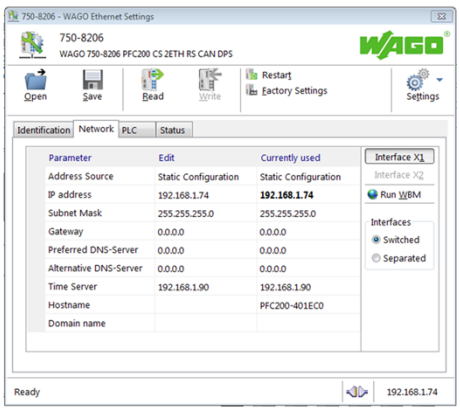


Offline node configuration

1st option: Set the communication parameters



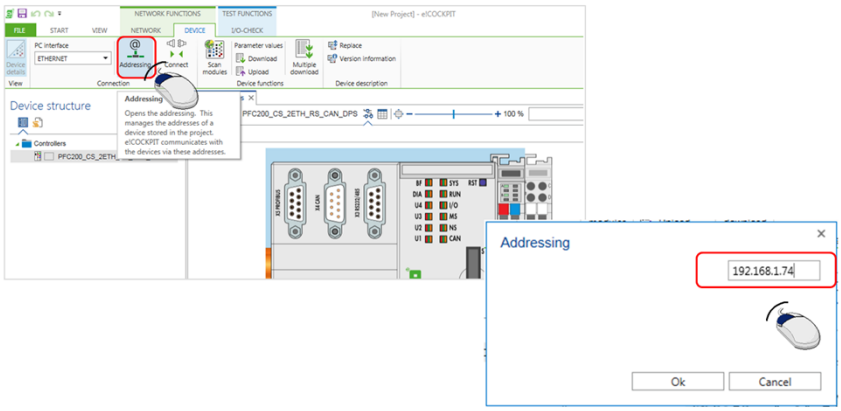
If Communication parameters are set correctly, you can read controller settings via Ethernet Settings



Offline node configuration

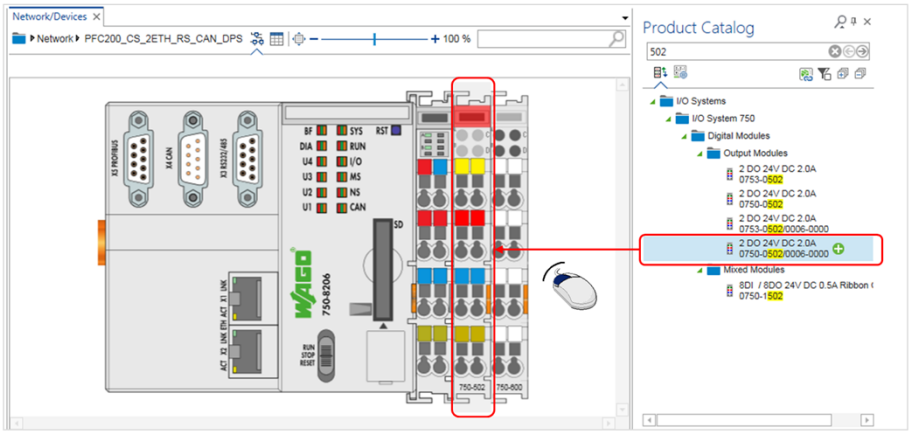
2nd option: Set the communication parameters

In Menu =>Device =>Addressing



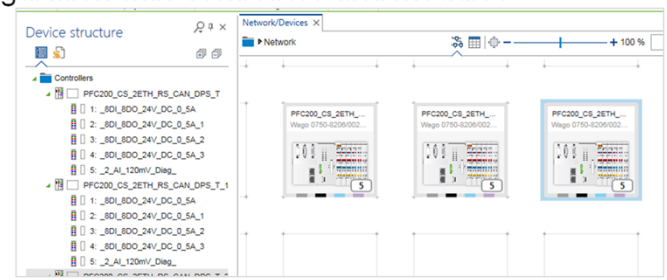
Offline node configuration

Insert the modules from the product catalog



Offline node configuration

Configuration with controllers and modules



Graphical view

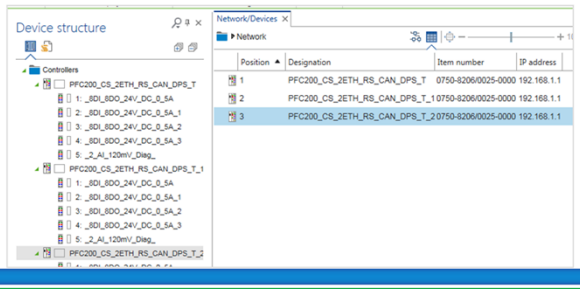
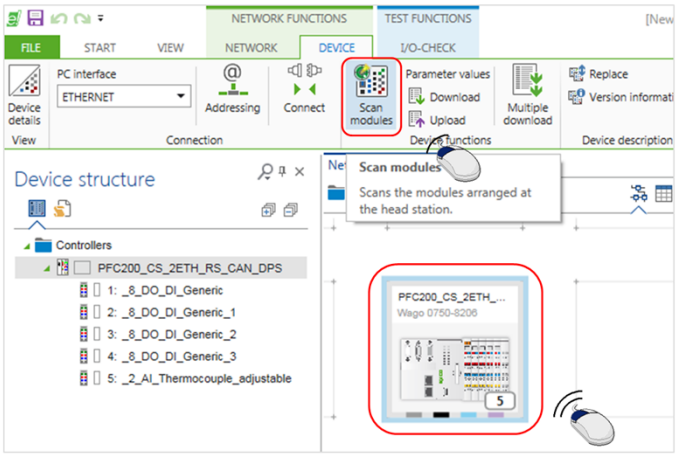


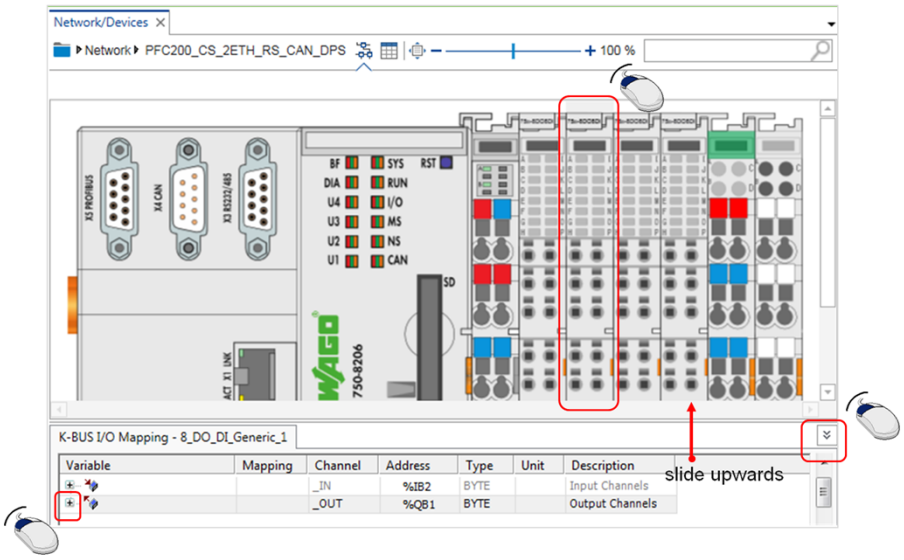
Table view

Configure variables

After Scan devices => Scan modules

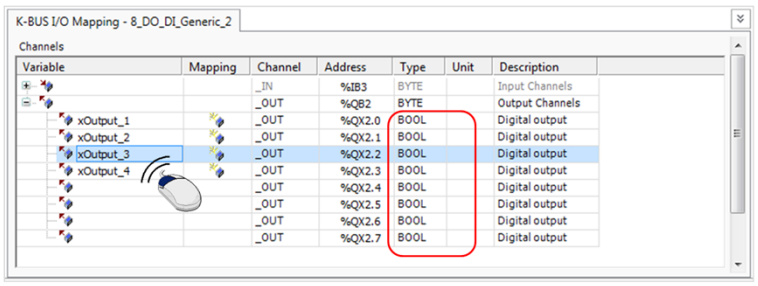


Configure variables



Configure variables

Type in the variables name of inputs and outputs

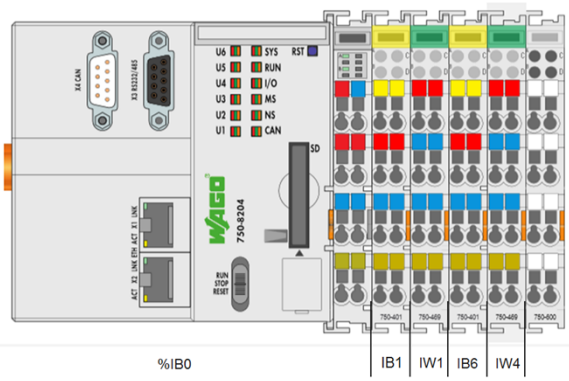


Digital modules can be addressed in word/ byte or in bit.

For outputs only alternatively and not at the same time.

Configure variables

%I, %Q process image of addressing scheme (example)

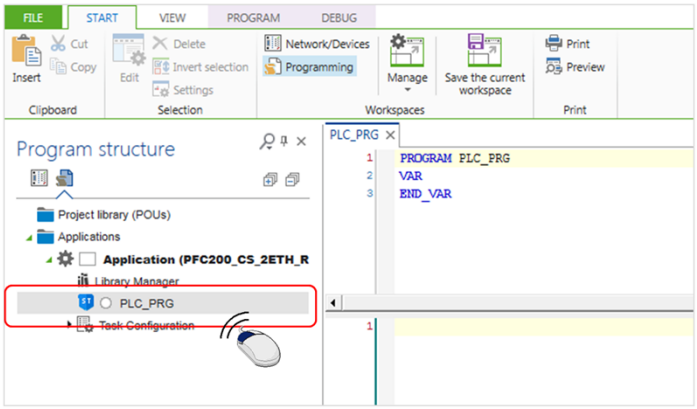


Byte	Word
%IB0 couple %IW0	
%IB1 digital module1	
%IB2	%IW1analog module1_1
%IB3	
%IB4	%IW2 analog module1_2
%IB5	
%IB6 digital module2	%IW3
%IB7 (empty)	
%IB8	%IW4 analog module2_1
%IB9	
%IB10	%IW5 analog module2_1
%IB11	

Due to this overlap of %IB and %IW areas, using this addressing is not recommended. Additionally, are changed the I%,%Q addresses for subsequent in the nodes configuration.

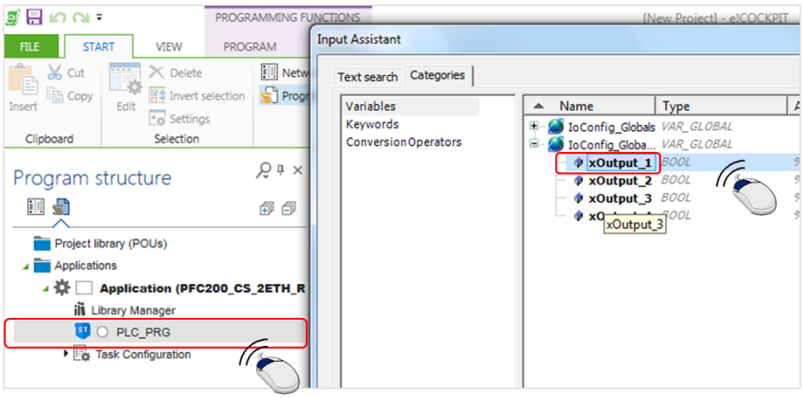
Write a simple program

Open Program structure



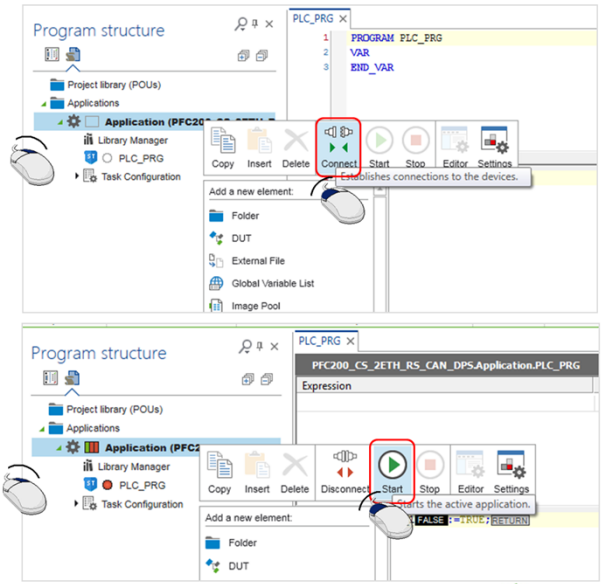
Write a simple program

Write a simple program (click in PLC_PRG and press the F2 button to get the configured variables

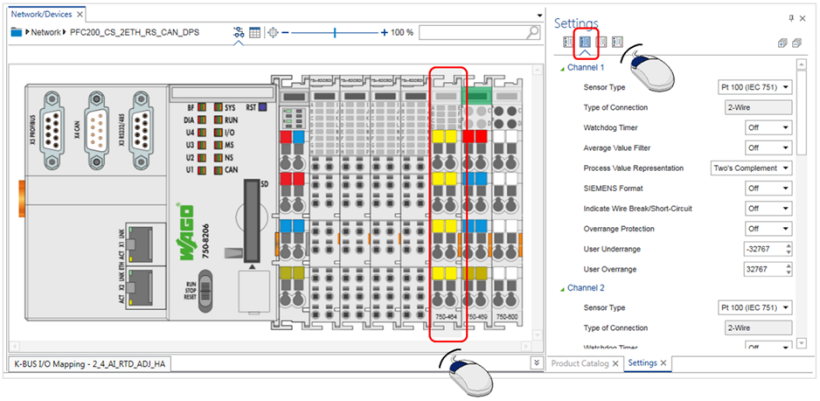


Write a simple program

Connect and Start

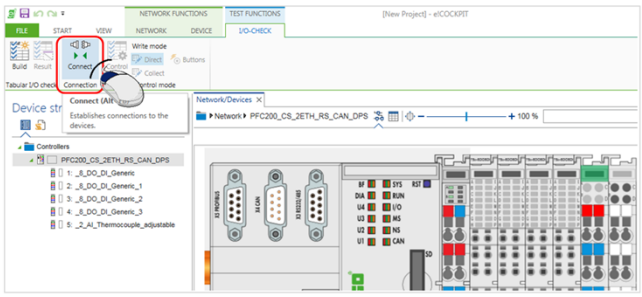


Module configuration



Additionally, some modules can be configured at settings (eg. 750-464).

I/O-CHECK

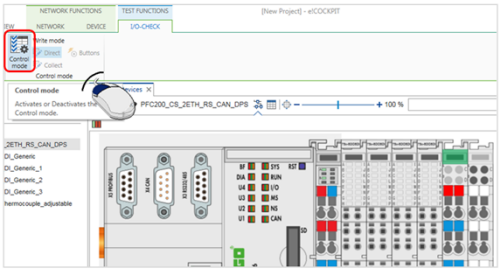


I/O-CHECK	control mode disabled:	Inputs can be read
I/O-CHECK	control mode enabled:	Inputs can be read, outputs can be set

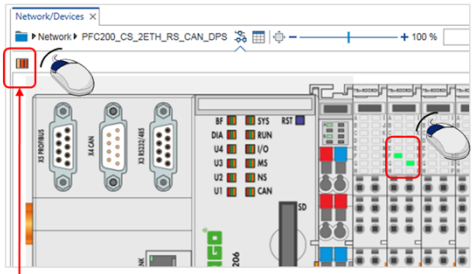
Control mode is only available, if PLC in STOP.

I/O-CHECK

How to read inputs and to set outputs



I/O-CHECK in control mode



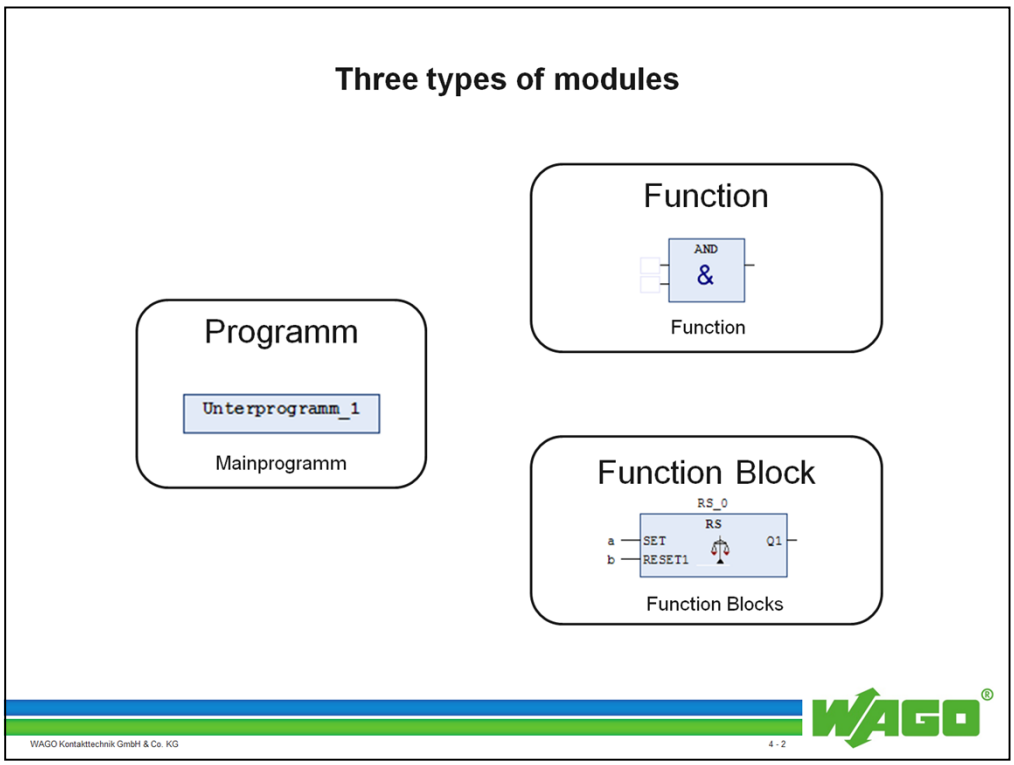
To set outputs in I/O-CHECK
click on the LED symbol

Control mode indicating color

4

Programming

- Function Block types
- Exercises



A project contains different types of objects:

- Modules
- Definitions of data types,
- Display elements (visualization), and
- Resources

The first module that is created in a project is automatically named
PLC_PRG

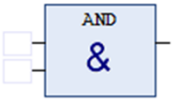
This starts the execution (corresponds to the Main function in a C program), and from here out additional components can be called like

- Programs
- Function Blocks (also called function modules)
- Functions

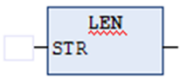
“Function” module type

Function

- A Function is a parameterizable module without memory, i.e. it always delivers the same results from the same input parameters
- A Function delivers its results back to the accumulator
- A Function has a data type
- This means that a function can only deliver a data element as a result



Operator



Function

The function in the POU

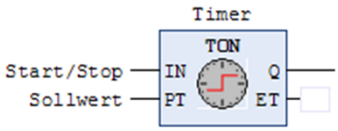
The Function is abbreviated 'FUN' and the key word is 'Function'. The function can be programmed as a parameterizable module in order to thus replace standard functions.

According to IEC 61131, it has been determined that the function may be assigned any number of input values. However, only one function value is returned. Thus, the module is used similar to an IL Instruction. Correspondingly, no output values can be assigned. Also, no Function Blocks can be called by a Function.

“Function Block” module type

Function Block

- A Function Block is a module that delivers one or more values when executed
- It can buffer values
- There are IEC FBs, Manufacturer FBs, and User FBs



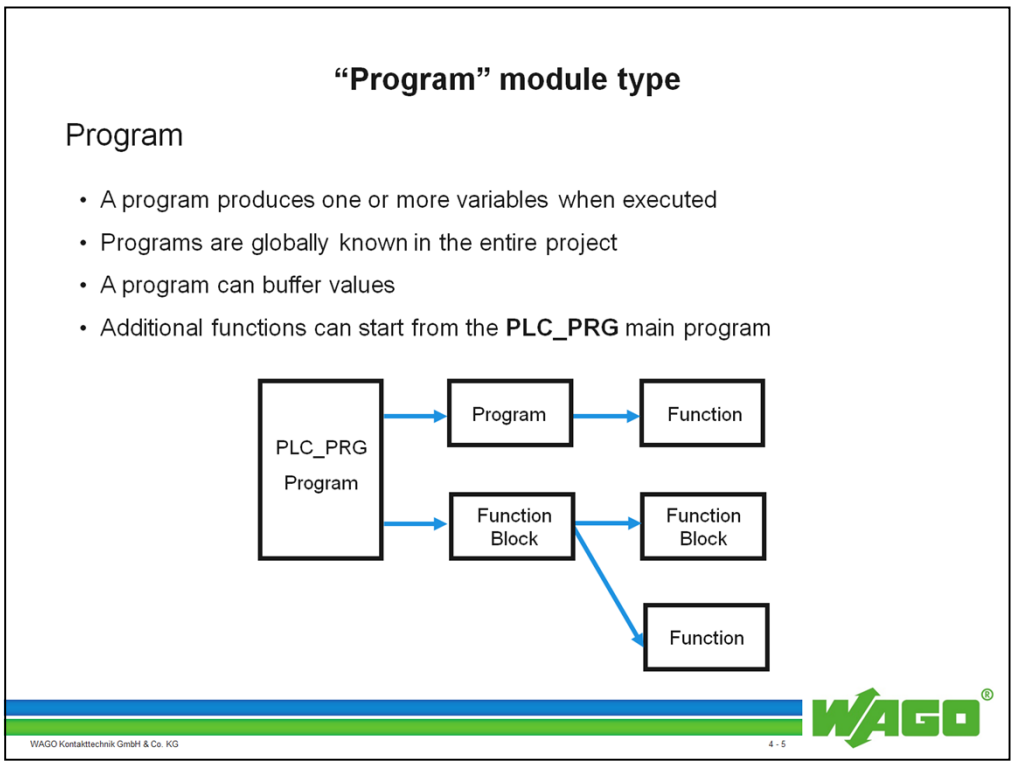
The Function Block in the POU

The Function Block is abbreviated FB. As a key word, the designation 'FUNCTION_BLOCK' is used.

In contrast to *Functions*, Function Blocks can transmit several output and in-out parameters. The parameters must be not "switched" when called (with the exception of the data instance).

Inputs (INPUT), outputs (OUTPUT), and input/output (IN/OUTPUT) signals and values can be transmitted using the function block. A memory, the instance, is assigned to the function block, which is defined according to the module start.

When calling a FB, a data instance must be assigned to it.
The data type of the data instance thereby always corresponds to the name of the FB.



The IEC 61131 modules for programming structure

Three modules types are standard in IEC 61131. The following module types are subordinate to the programming structure, which is designated as the 61131 "Program Organization Unit" (POU) in the IEC. These module types are designated as Program (PROG), Function (FUN), and Function Block (FB). A short description of the module types follows.

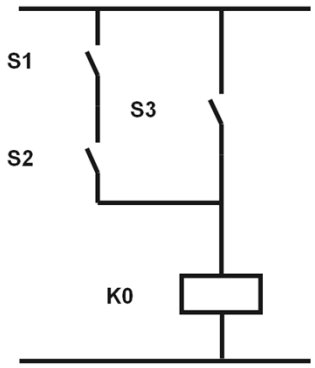
The Program

The most important part of a PLC program according to the IEC is the Program, designated by the abbreviation PROG. The key word has the designation 'PROGRAM'. This module type represents the main program, that was OB1 according to DIN 19239. Functions & Function Blocks are called in the PROG module. At the beginning of the module, the arrangement of inputs/outputs and global data are defined, which the remaining program then accesses.

Exercise 4.1

First program in LD

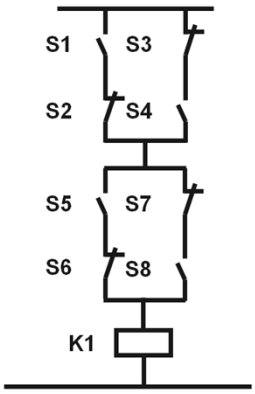
Write a program with the following function



Exercise 4.2

Program in CFC

Write the program in CFC

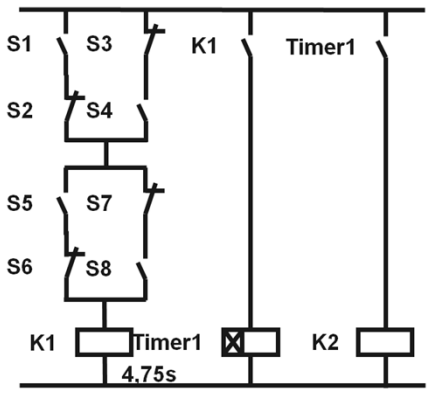


You will find an explanation of the modules "AND" and "OR" in the appendix on page 6-3.

Exercise 4.3

Program with a timer

Add an ON delay for output K2 with a 4.75 s delay time to the program from exercise 4.2.



You will find an explanation of the Timer modules in the appendix on page 6-8.

Exercise 4.4

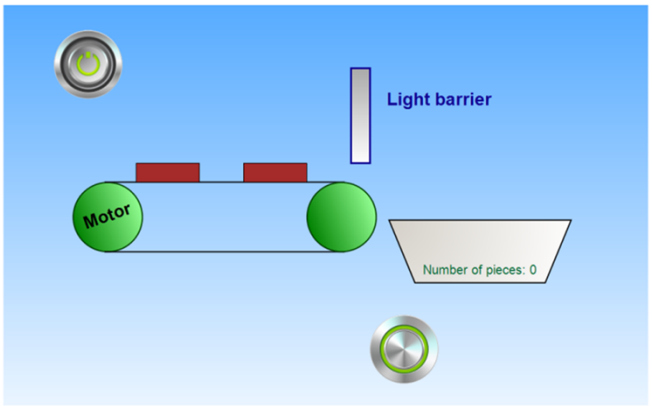
Generate a 'PLC_VISU' with the elements depicted.

The system is switched using 'S1' to "RUN" and using 'S2' to "STOP".

Using 'S3' simulates the "light sensor" that is used for counting the parts.

The "motor" runs as long as the counter has not reached the value 5 and the system is set to "RUN".

The counter is reset by using "S4" (pallet change).



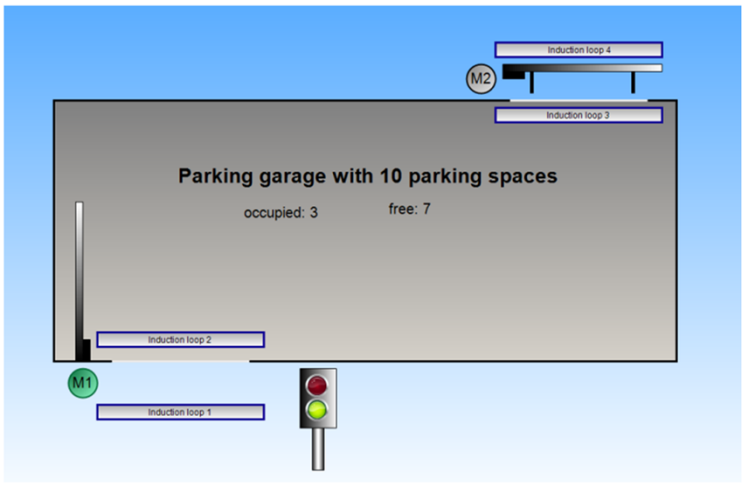
WAGO Kontakttechnik GmbH & Co. KG 4 - 9

Generate a project for controlling a conveyor belt with the following characteristics:

- The system is activated via the RUN button (switch).
- Activating the switch starts the motor for the conveyor belt (if the quantity is below the maximum value).
- The conveyor belt transports packets to a palette, which can accept a maximum of 5 packets.
  The number of packets is determined by means of a light barrier.
- If the maximum number of 5 packets is reached, then the conveyor belt stops.
- The palette is then exchanged by the system monitor for an empty one, and by actuating the palette exchange button (switch), the quantity counter is reset to zero and the conveyor belt is restarted.
- There is a STOP button to shut off the system, which shuts off the conveyor belt and deactivates the packet counter unit (The quantity status is not reset). The RUN button must be activated for restart. The RUN button must be activated for restart.

As an addition to the SPS program, generate a ***e!****COCKPIT* visualization, via which the system can be monitored or alternately operated.

Exercise 4.5 (Optional)



Add to the project for controlling a parking garage with the following characteristics:

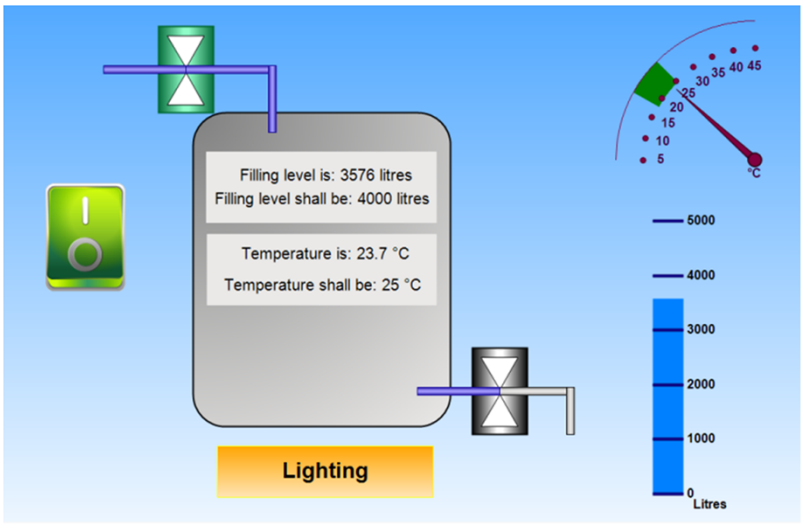
- The first barrier (M1) is opened by actuation of "Induction loop 1". By actuating "Induction loop 2", a counter (number of vehicles) is increased by one and the barrier (M1) is closed again.
- The second barrier (M2) is opened by actuation of "Induction loop 3". By actuating "Induction loop 4", a counter (number of vehicles), is reduced by one and the barrier (M2) is closed again.
- Counting may only function when the barriers are open.
- If the maximum number of 10 vehicles is reached, the traffic light at the entrance to the parking garage switches from green to red and the entrance to the parking garage is blocked.

Additional task:

- Expand the project by a variable specification (input field in VISU) of the maximum number (CAR_MAX) to be between 10-50 vehicles.
- Using the program, determine the number of available parking places and display this at the entrance to the parking garage.

Exercise 4.6

Boiler



Exercise 4.6 (description)

Boiler

Implement a boiler control, by means of which a water boiler can be filled and heated.
The system should be started/stopped using a Start/Stop button.

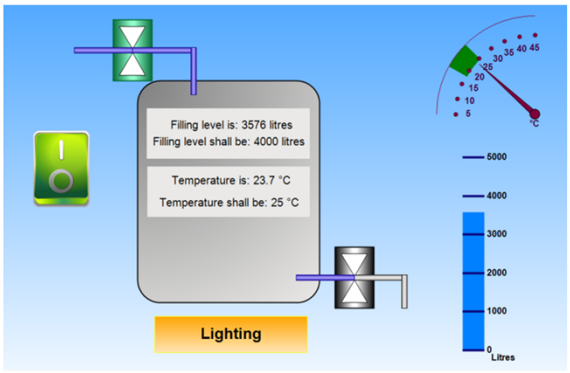
At a filling level of 4000 liters, the supply should be closed. The fill level is determined using a 0-10 V module..

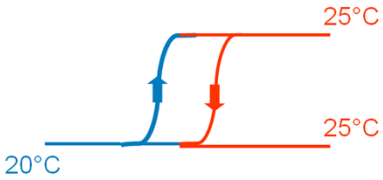
At a temperature above 25°C, boiler firing should be stopped. The temperature is determined by means of a thermocouple.

Exercise 4.7

Addition

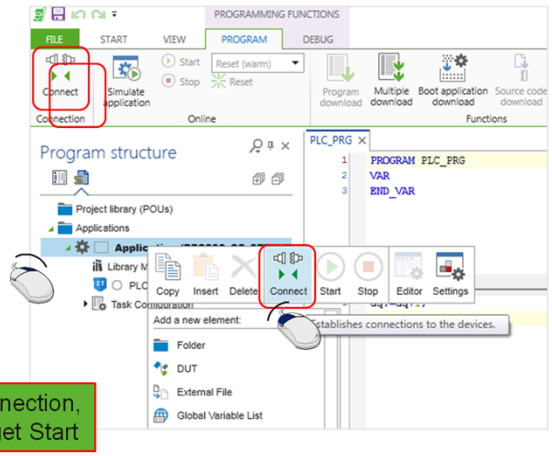
In order to protect the burner, it should be switched via a hysteresis function.





Download the program in device (RAM)

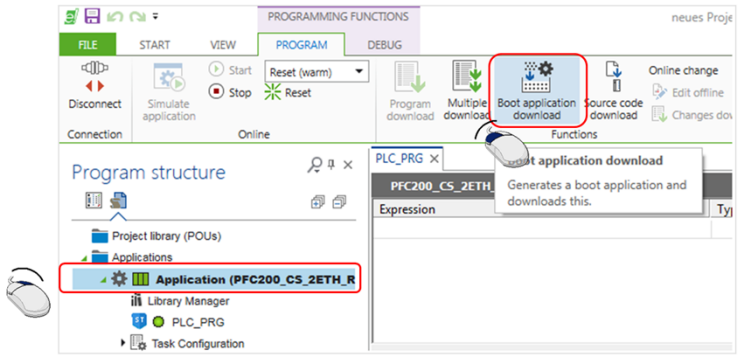
By click of the right mouse button on application, or connection under the program ribbon



After connection, don't forget Start

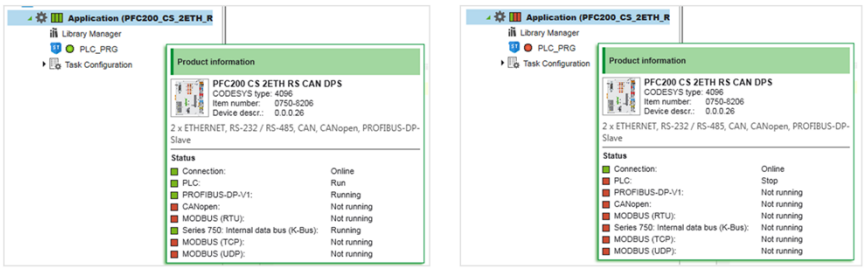
Download the program in device (Flash)

For permanent saving, boot application- download under the program ribbon



Diagnostics

The state of the controller is indicated by the status field



Left: Connection state
Center: Run/ Stop
Right: Fieldbus/ K bus state

5

Data types & variables

- Data types
- Variables

Elementary data types according IEC 61131-3

Data type	Lower limit	Upper limit	Data width
BOOL	**FALSE**	**TRUE**	**1 Bit**
BYTE	**0**	**255**	**8 Bit**
WORD	**0**	**65535**	**16 Bit**
DWORD	**0**	**4´294´967´295**	**32 Bit**
LWORD	0	2^{64}-1	64 Bit
SINT	-128	127	8 Bit
USINT	0	255	8 Bit
INT	**-32768**	**32767**	**16 Bit**
UINT	0	65535	16 Bit
DINT	-2´147´483´648	2´147´483´647	32 Bit
UDINT	0	4´294´967´295	32 Bit
LINT	2^{63}	2^{63}-1	64 Bit
ULINT	0	2^{64}-1	64 Bit
REAL	**Exp. 3.141592654**		**32 Bit**
LREAL			64 Bit
STRING	variable (255 Byte max.)		
TIME	**Exp. T# 5d4h5m10s123ms**		**32 Bit**
TOD	Exp. TOD#15:36:10.340	(TIME_OF_DAY)	32 Bit
DATE	Exp. D#2001-03-28		32 Bit
DT	Exp. DT#2001-03-28-11:26:01	(DATE_AND_TIME)	32 Bit

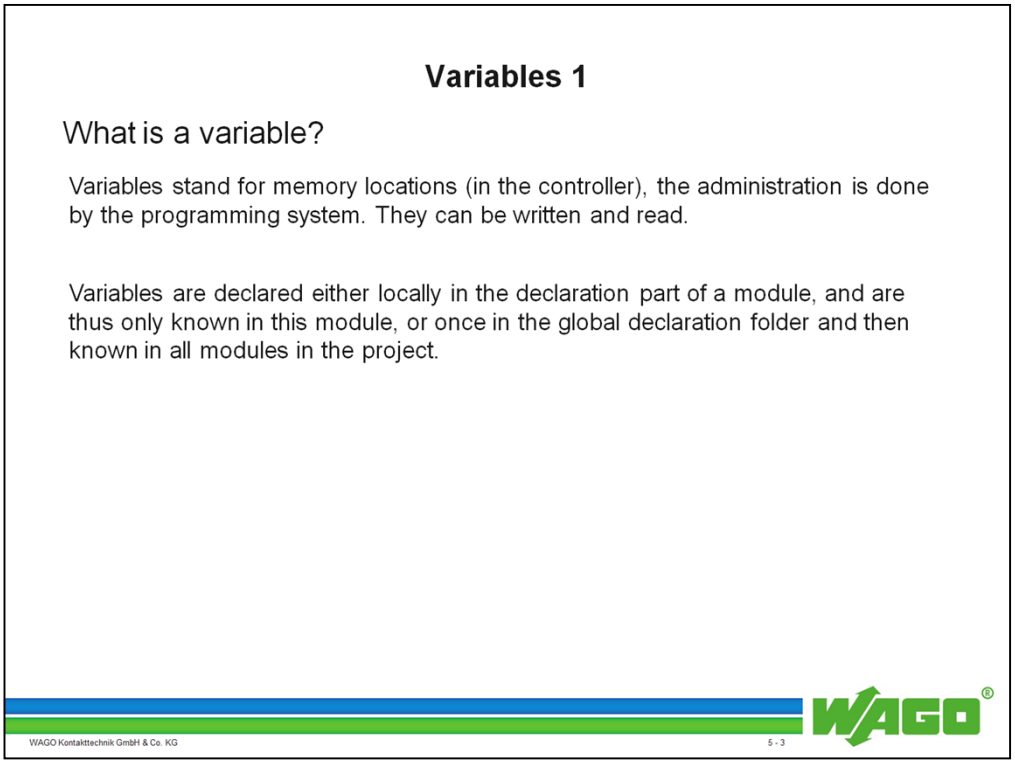
There has to be a standardization of data types (data formats) for the uniform presentation of process data beyond system boundaries of different manufacturers.

At this point, IEC 61131-3 introduces specific requirements for PLC systems.

The most commonly used elementary data types

- BOOL
- BYTE
- WORD
- INT
- DWORD
- REAL
- TIME

are marked in the table (above) in **bold**.



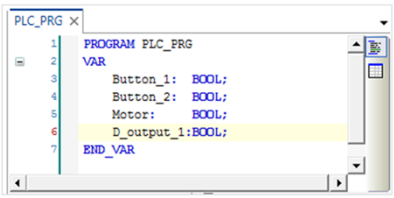
A local variable is only known in the module in which it is declared. In the standard case, most variables are local. In case values need to be exchanged with other modules/programs, they can be declared as global variables.

It is permissible to call a local variable and a global variable by the same name. In this case, the variable is used that is “closer” to the module, thus a local variable.

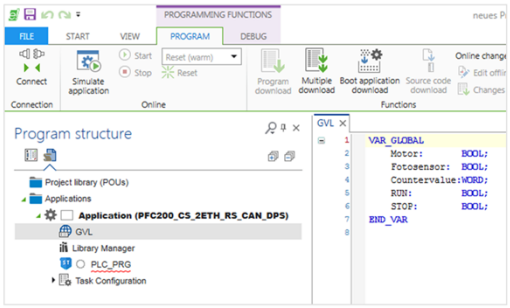
When considering identically named variables, the following sequence arises:

Local variable → Global variable → Variable from a library

Variables 1



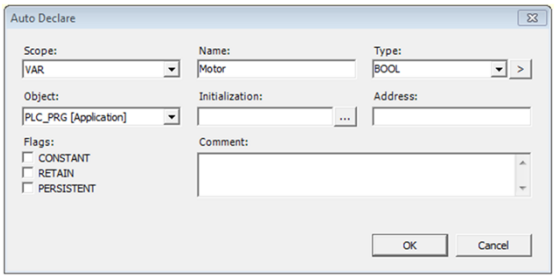
Local declaration



Global declaration

Variables 2

Declaration of variable class



Class	Description
VAR	Local variable in a POU (Standard)
VAR_INOUT	Input variable
VAR_OUTPUT	Output variable
VAR_IN_OUT	Input/output variable
VAR_GLOBAL	Global variable, known in all POUs

WAGO Kontakttechnik GmbH & Co. KG 5 - 5

All variables that are only used in this module are declared in the declaration part of a module.

These can be

- Local variables (VAR = used for temporary data, that is only valid for processing the respective module)
- Input variables (VAR_INPUT = used for data that is transmitted into the module)
- Output variables (VAR_OUTPUT = used for output data that is generated in the module and output to other program parts)
- Input/output variables (VAR_IN_OUT = used for data that is transmitted into the module, processed there, and then output to additional program parts.

The declaration syntax follows the standard of IEC 61331-3

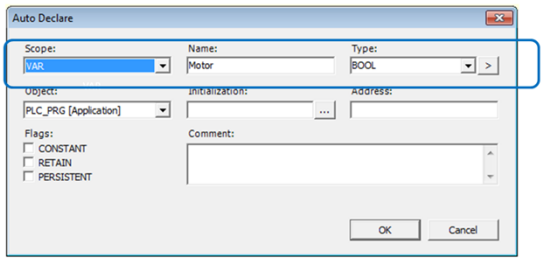
The declaration VAR_GLOBAL moves the variable out of the local declaration part into the list of Global_variables (to be found in the “Resources” tab 4).

All variables, constants, or residual variables can be declared global variables that should be known in the entire project.

Variables 3

Declaration of a variable

Required entries



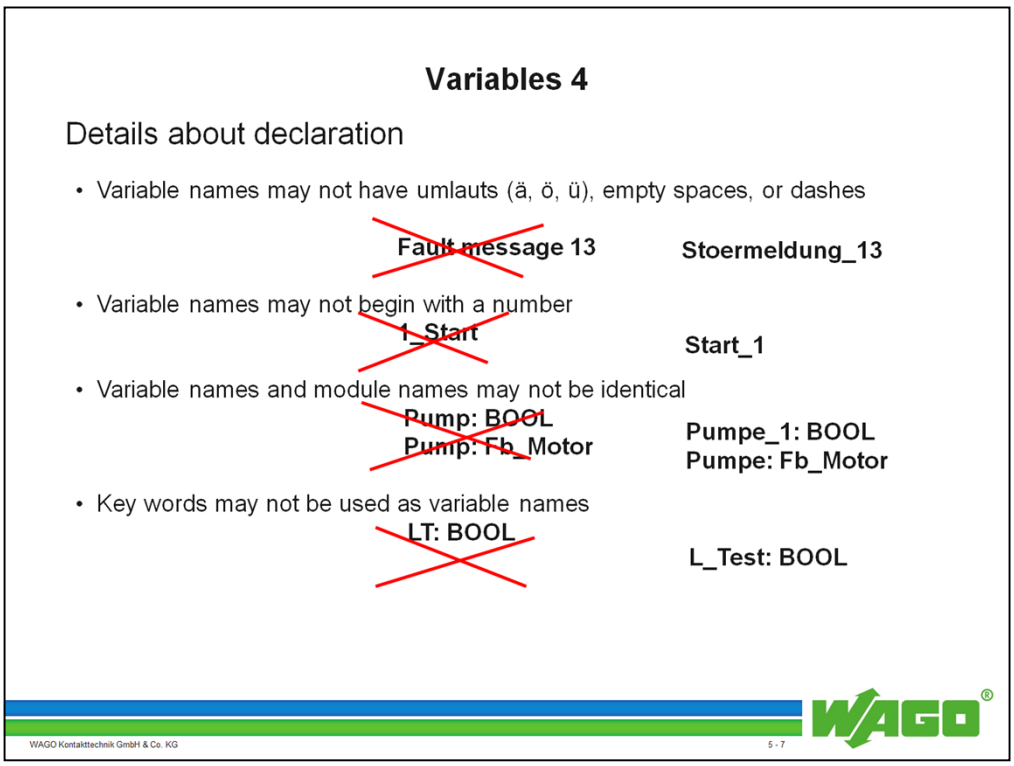
- Name
- Type
- Initial value
- Address
- Comment
- CONSTANT — Constants; the value is not changed during a program execution (only read)
- RETAIN — These variables retain their value after an uncontrolled termination of the control or a reset
- PERSISTENT — In contrast to Retain variables, these variables retain their value even after a cold start, i.e. when the control is terminated normally and has been restarted, or after a download. Persistent variables are not automatically Retain variables!

WAGO Kontakttechnik GmbH & Co. KG 5 - 6

Retain/ Persistant variables

Retain variables can keep their values even after warm reset or power off..
The behaviour is shown in the table below.

	Reset warm (supply failure)	Reset cold	Reset original
Standard	0	0	0
VAR RETAIN PERSISTENT	X	X	0



For the designations, that is the naming of variables, it should be taken into consideration that variables cannot have empty spaces or umlauts, they cannot be declared twice, or be identical to key words.

No distinction is made between upper and lower case spelling of variable names, meaning that VAR1, Var1 and var1 are all the same variable.

Underscores are significant in designation, e.g. "A_BCD" and "AB_CD" are interpreted as different designations.

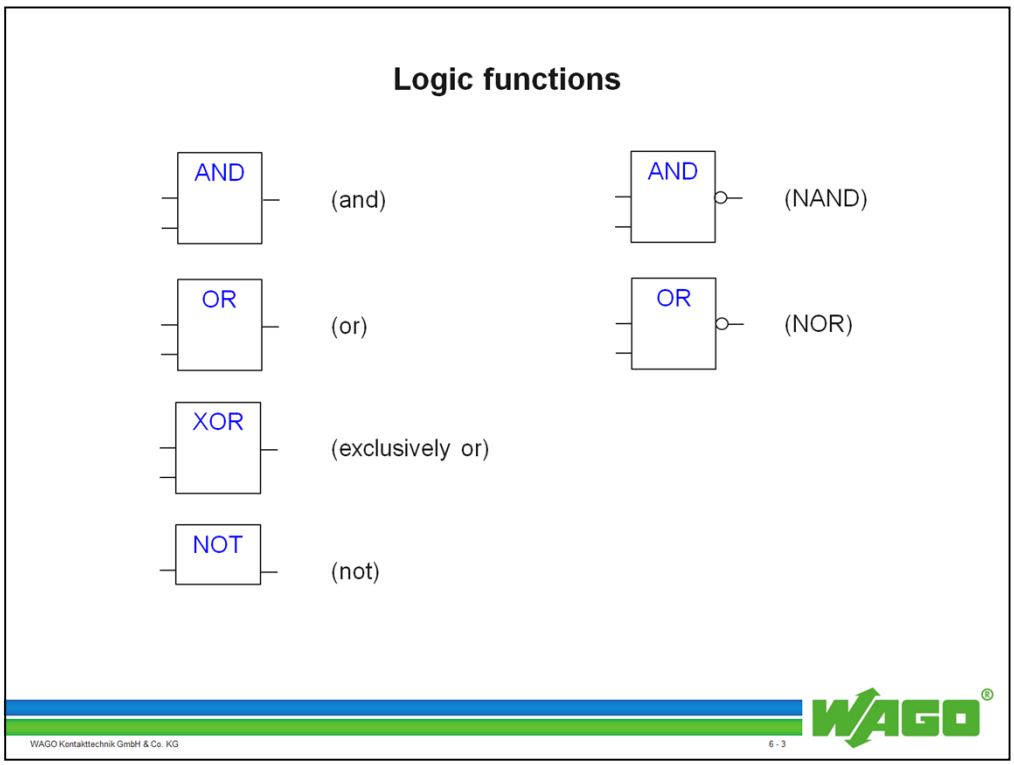
Consecutive underscores are prohibited at the beginning of a designation or within a designation. The designation length as well as the significant range is unlimited (max. 255 characters) for all practical purposes.

6

Appendix

Useful key combinations

F1	Help
F2	Input help
F5	Start
<Strg>+F7	Write value
F7	Force value
<Strg>+<Umschalt>+F7	Cancel force
<Umschalt>+F2	Variable declaration
<Alt>+F8	Log in
<Strg>+F8	Log out
F4	Next error

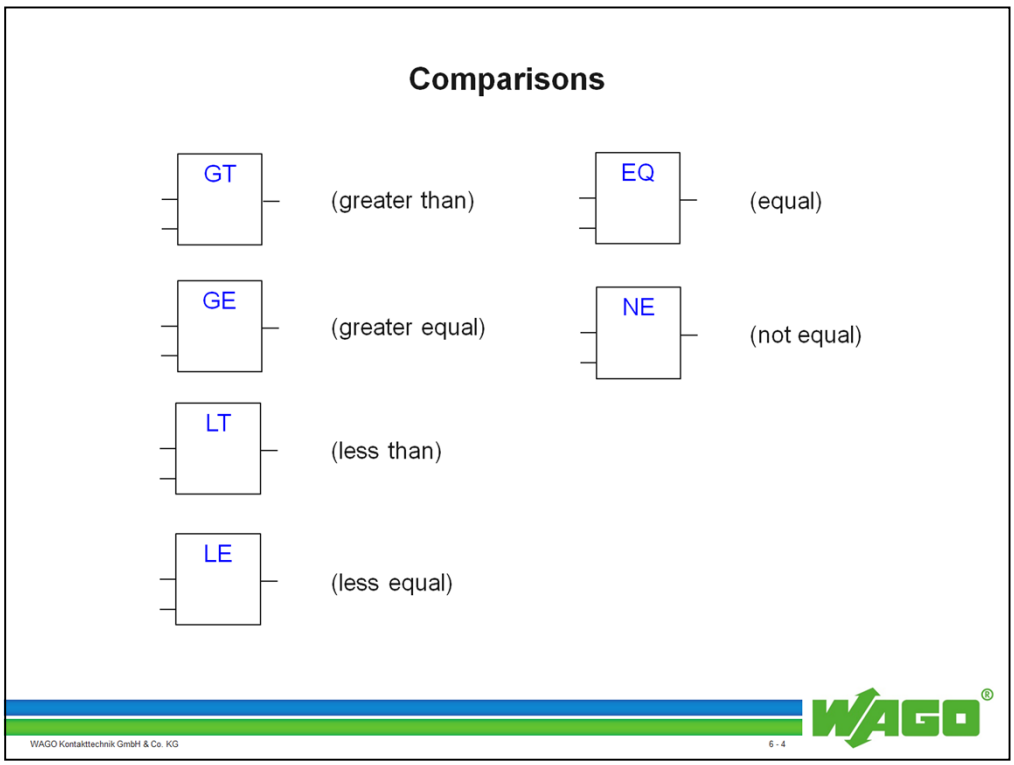


AND: The output is TRUE, if all inputs are TRUE.
(For BOOL, BYTE, WORD, DWORD)

OR: The output is TRUE, if at least one input is TRUE.
(For BOOL, BYTE, WORD, DWORD)

XOR: The output is TRUE, it exactly one input is TRUE.
(For BOOL, BYTE, WORD, DWORD)

NOT: The output is the opposite of the input.
(For BOOL, BYTE, WORD, DWORD)



GT: The output is TRUE, if input 1 is greater than input 2.
(For all data types)

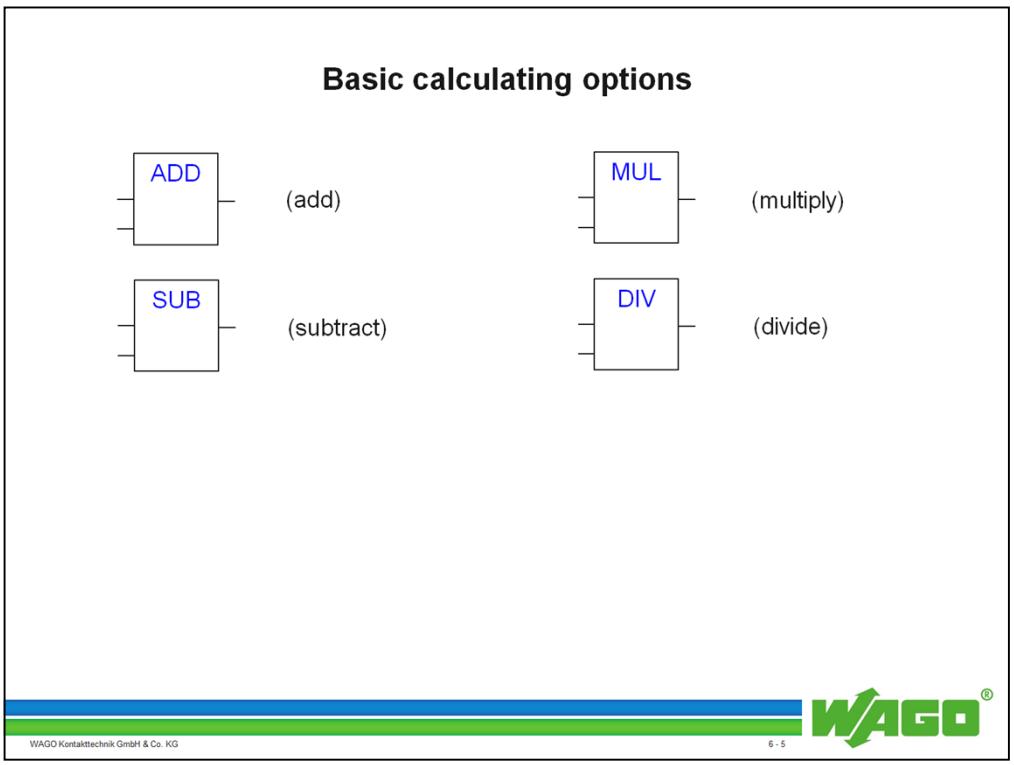
GE: The output is TRUE, if input 1 is greater than or equal to input 2.
(For all data types)

LT: The output is TRUE, if input 1 is less than input 2.
(For all data types)

LE: The output is TRUE, if input 1 is less than or equal to input 2.
(For all data types)

EQ: The output is TRUE, if both inputs are equal.
(For all data types)

NE: The output is TRUE, if both inputs are not equal.
(For all data types)



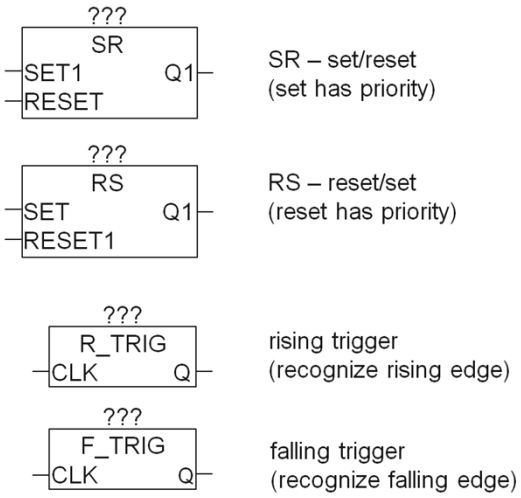
ADD: The output is the result of adding the inputs.
(For all data types, except BOOL, STRING, DATE, TOD)

SUB: The output is input 1 minus input 2.
(For all data types, except BOOL, STRING, DATE, TOD)

MUL: The output is the result of multiplying the inputs.
(For all data types, except BOOL, STRING, DATE, TOD)

DIV: The output is input 1 divided by input 2.
(For all data types, except BOOL, STRING, DATE, TOD)

Standard Function Blocks (1)



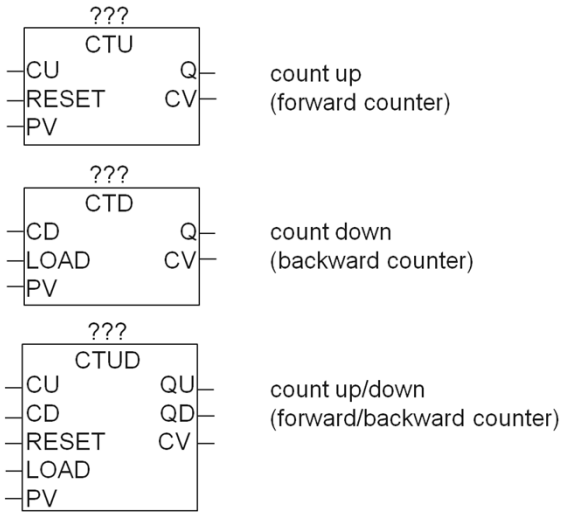
SR: A holding element. If the input SET1 were TRUE once, then the output remains TRUE, until the RESET input is TRUE. (only for BOOL)

RS: A holding element. If the input SET1 were TRUE once, then the output remains TRUE, until the RESET input is TRUE. (only for BOOL)

R_TRIG: Reacts to a rising edge of the input CLK and sets the output Q for a program cycle to TRUE. (only for BOOL)

F_TRIG: Reacts to a falling edge of the input CLK and sets the output Q for a program cycle to TRUE. (only for BOOL)

Standard Function Blocks (2)

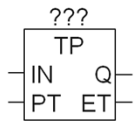


CTU: At a rising edge at the CU input, the Counter status is increased by 1 at the CV output. If PV is equal to CV, then the output is TRUE. Using the RESET input, the counter can be reset to 0.

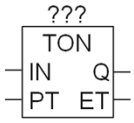
CTD: At a rising edge at the LOAD input, CV is equal to PV. At a rising edge at the CD input, the counter status is decreased by 1 at the CV output. The output Q is TRUE if CV equals 0. Output Q becomes TRUE, if CV equals 0.

CTUD: At a rising edge at input LOAD, CV equals PV. At a rising edge at input CD, the counter status on output CV is reduced by 1. At a rising edge on input CU the counter status CV is increased by 1. Using input RESET the counter can be reset to 0. If PV equals CV Output QU becomes TRUE. Output QD becomes TRUE, if CV equals 0.

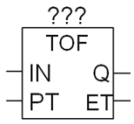
Standard Function Blocks (3)



timer pulse
(pulse encoder)



timer on
(timer delay)



timer off
(off delay)

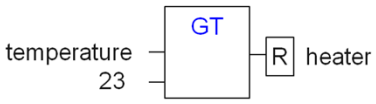
TP: A rising edge at the Boolean input IN sets the t put Q for the time that is set at the input PT (preset time).

TON: A rising edge at the Boolean input IN starts the Timer and after expiration of the time at input PT, the output Q is TRUE. Output ET is the elapsed time for the timer.

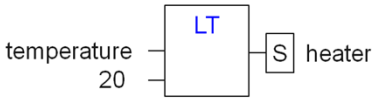
TOF: A rising edge at the Boolean input IN sets the output Q. A falling edge at IN starts the Timer, and after expiration of the time at input PT, Q is again FALSE. Output ET is the elapsed time for the timer. Output ET is the elapsed time for the timer.

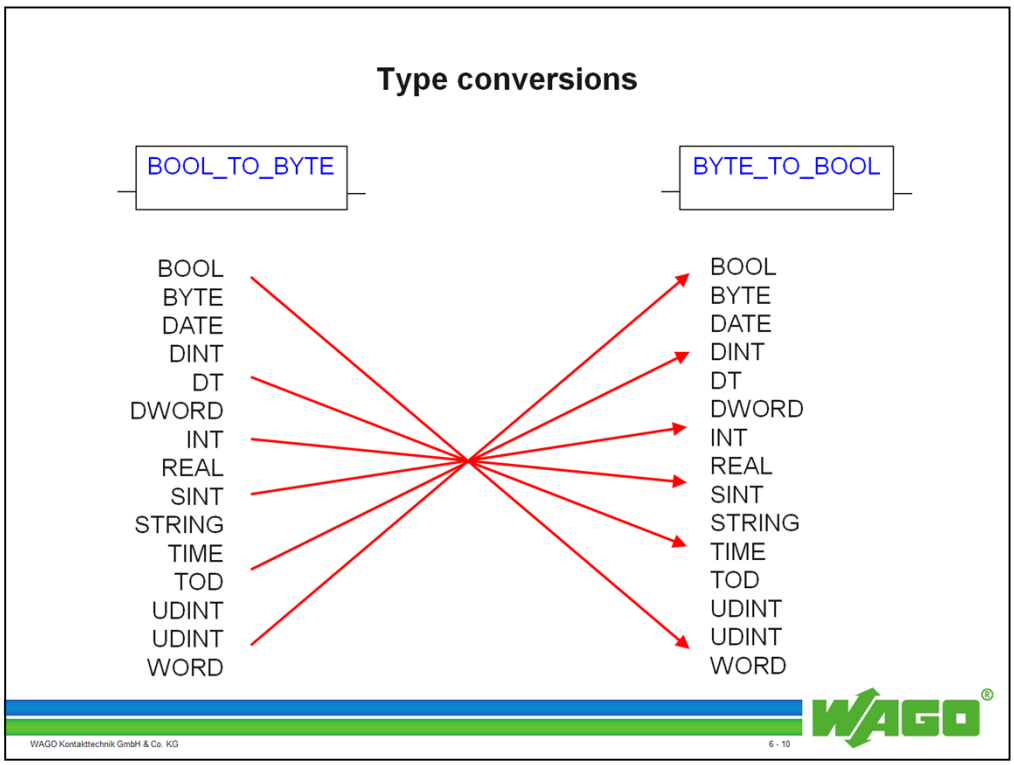
Set/ Reset (variants)

A reset output is set to FALSE, if the associated gate returns TRUE. The output retains its value, even if the gate jumps back to FALSE.



A set output is set to TRUE, if the associated gate returns TRUE. The output retains this value, even if the gate jumps back to FALSE.





In general, each data type can be converted into another.
However, data losses can occur in some type conversions.

Variables and power off

Persistent variables (key word PERSISTENT) maintain their values only after a new download ("online (download)") as they are not stored in the "retain area". If persistent variables are also supposed to maintain their previous values following an uncontrolled loss of power, then they must additionally be declared as VAR_RETAIN.
Application example: An operating hours counter, which should resume counting following a power failure.

Retain variables (key word RETAIN) keep their values after an uncontrolled termination as well as after normally switching the PLC on or off (according to the "online Reset)" command). Retain variables are, however, re-initialized for "Reset (cold)", "Reset (original)" and for a new program download.
Application example: An item counter in a production line, which continues counting where it left off after a power failure.

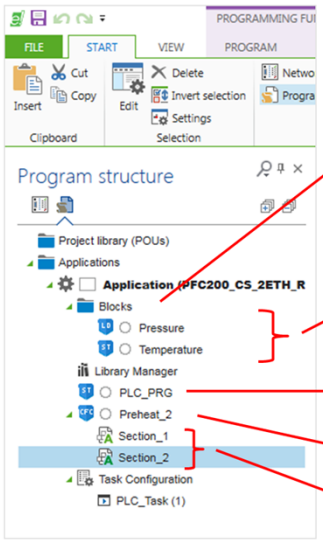
Variables and power off

	Reset warm (power off)	Reset cold	Reset original
Standard	0	0	0
VAR RETAIN PERSISTENT	X	X	0

X = value retained 0 = value is re-initialized

Reset original deletes a program from RAM and Flash memory

Project structures



Folder (compilation of modules)

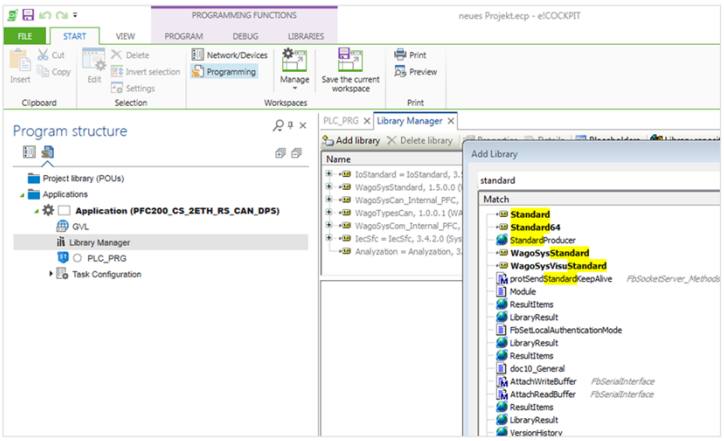
Different Function blocks

The main program
(Subroutines are called from here)

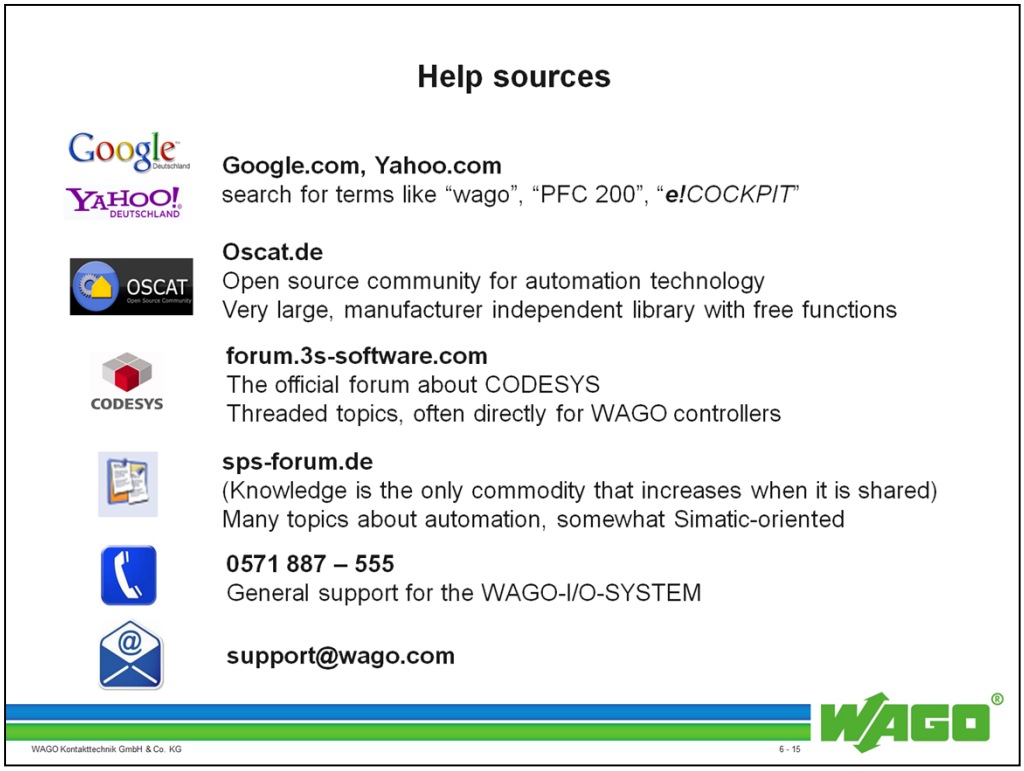
A subroutine

Action within a subroutine

Adding libraries



Using the search box <u>makes it easier</u> to find the correct library



In addition to the substantial teaching offerings by WAGO Kontakttechnik GmbH & Co. KG, there are also numerous online offerings as well as books, which enable further training in a self-directed study in use of the ***e!****COCKPIT* software package according to IEC 61131.

In the **OSCAT-Forum**, examples for the following PLC systems can be found:

Discussions and announcements about the core library, oscat.lib for CODESYS 3.

Literature

IEC 6113/ Programming

SPS-Programmierung gemäß IEC 61131-3
Mit Beispielen für CODESYS und STEP 7, by Heinrich Lepers
(German Language)
2nd edition, dated 05.2007
Franzis Verlag GmbH, ISBN: 3-7723-5805-5

Programming Industrial Automation Systems:
Concepts and Programming Languages, Requirements for Programming Systems, Decision Making Aids, by Karl Heinz John and Michael Tiegelkamp
3rd revised edition 2000
ISBN: 978-3-540-66445-1

PLC Software Development according to IEC 61131.
by Jens von Aspern, published May 2000
452 pages, including illustrations and tables, paperback
Publisher Hüthig Jehle Rehm GmbH
ISBN: 3778526812

The book PLC programming according to IEC 61131-3, with examples of CODESYS and STEP 7 by Heinrich Lepers, 4th Edition, of 08.2011, Franzis Verlag GmbH, ISBN: 3 7723-5805-5, is especially highly recommended for migrators from Siemens S7.

Summary (acc. amazon.de)

“The aim of this book is to convince PLC programmers and programming for beginners that programming in higher level languages of the IEC standards (FBS / CFC, ST / SCL and AS / graph) is much more effective, easier to maintain and more cost-efficient than the still widespread programming according to the statement list (STL / LAD / FBD). The book is aimed at both S5 / S7 practitioners who have programmed in STL / LAD / FBD and learn about the benefits of higher IEC-compliant languages CFC, SCL, SFC and S7-Graph and wish to use it, as well as to all who want to enter as a beginner in programming according to IEC 61131-3. These are trainees in vocational schools, chambers, etc.. and students, but also engineers, technicians, foremen and skilled workers.”


WE
INNOVATE!
WAGO