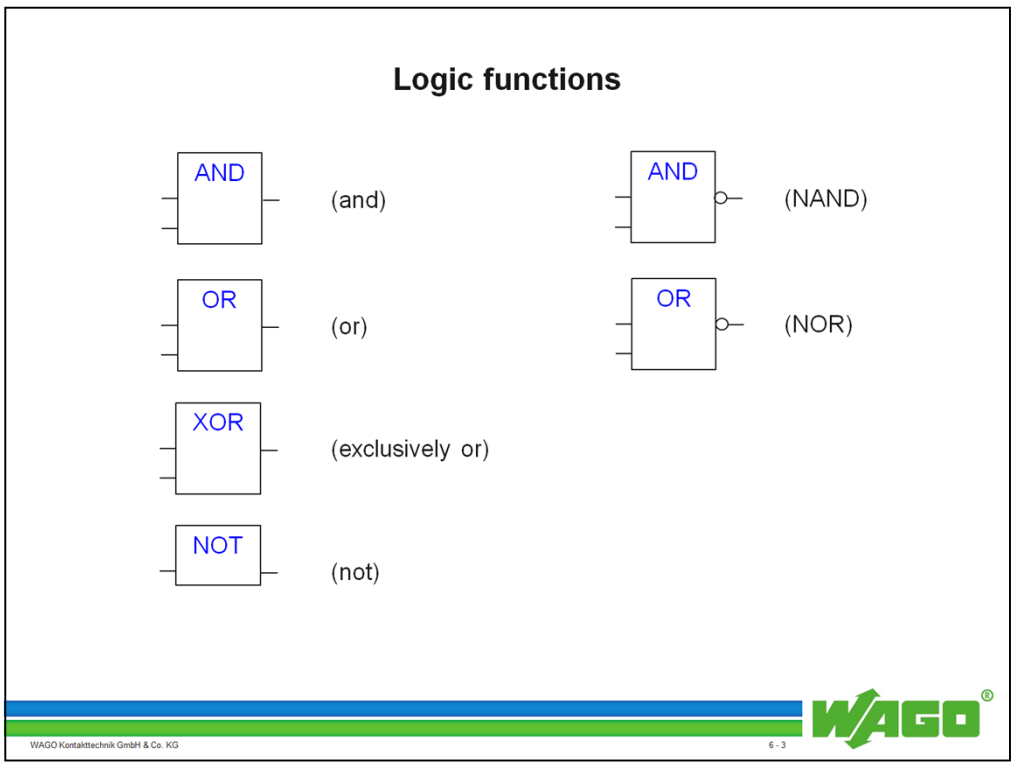
6

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

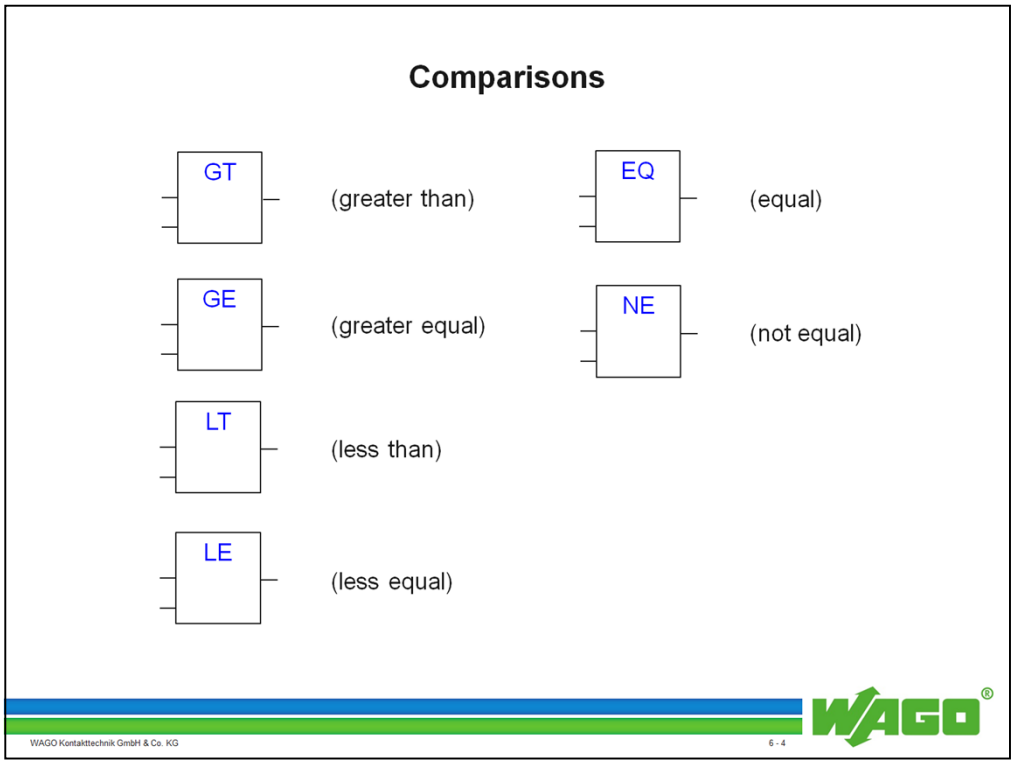


AND: The output is TRUE, if all inputs are TRUE.
(For BOOL, BYTE, WORD, DWORD)

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)

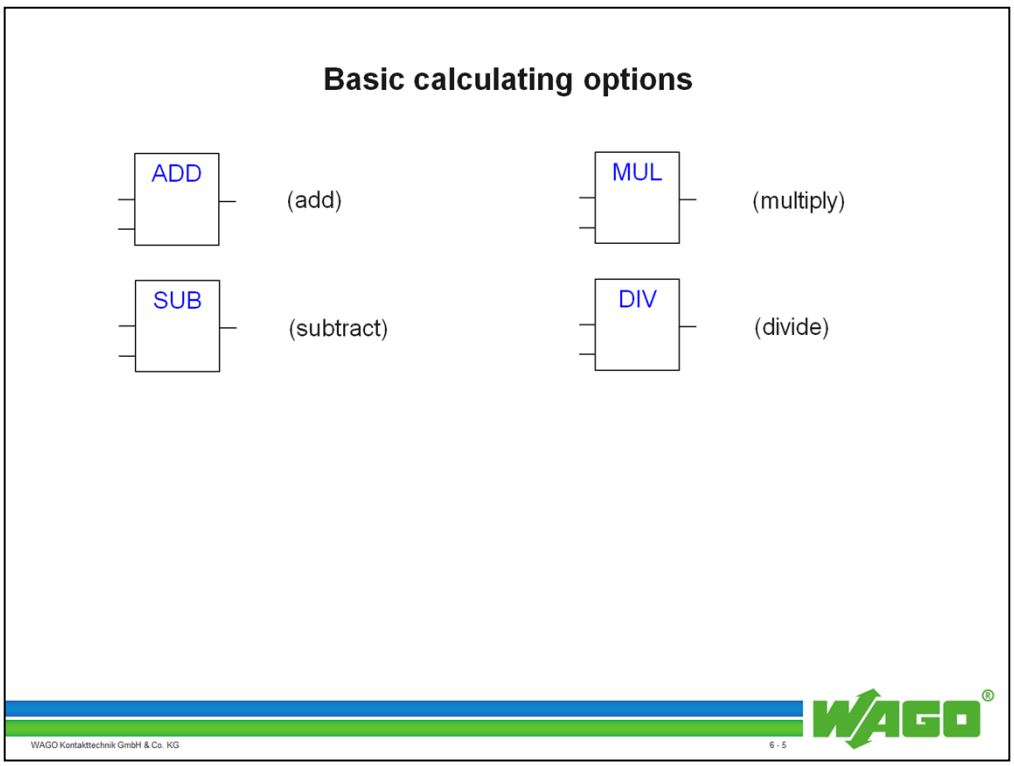
GE: The output is TRUE, if input 1 is greater than or equal to input 2.
(For all data types)

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)



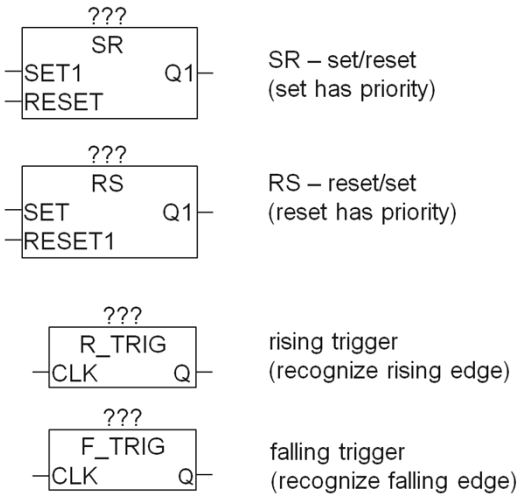
ADD: The output is the result of adding the inputs.
(For all data types, except BOOL, STRING, DATE, TOD)

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)



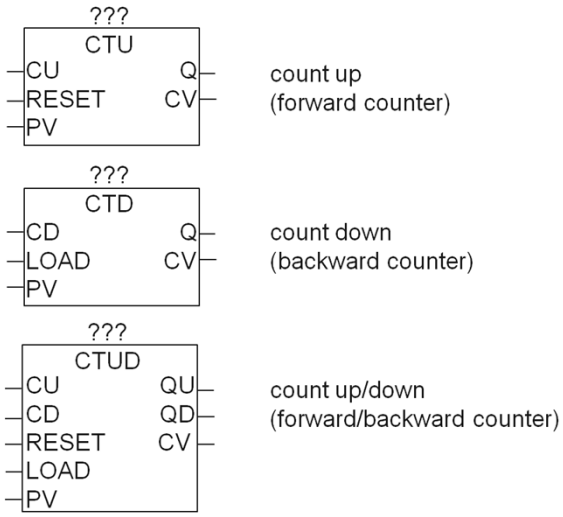
SR: A holding element. If the input SET1 were TRUE once, then the output remains TRUE, until the RESET input is TRUE. (only for BOOL)

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)

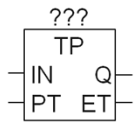


CTU: At a rising edge at the CU input, the Counter status is increased by 1 at the CV output. If PV is equal to CV, then the output is TRUE. Using the RESET input, the counter can be reset to 0.

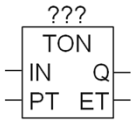
CTD: At a rising edge at the LOAD input, CV is equal to PV. At a rising edge at the CD input, the counter status is decreased by 1 at the CV output. The output Q is TRUE if CV equals 0. Output Q becomes TRUE, if CV equals 0.

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.

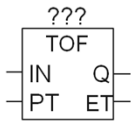
Standard Function Blocks (3)



timer pulse
(pulse encoder)



timer on
(timer delay)



timer off
(off delay)

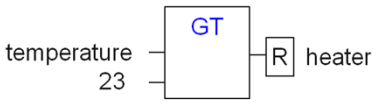
TP: A rising edge at the Boolean input IN sets the t put Q for the time that is set at the input PT (preset time).

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.

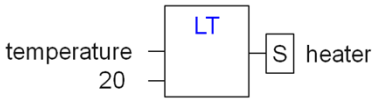
TOF: A rising edge at the Boolean input IN sets the output Q. A falling edge at IN starts the Timer, and after expiration of the time at input PT, Q is again FALSE. Output ET is the elapsed time for the timer. Output ET is the elapsed time for the timer.

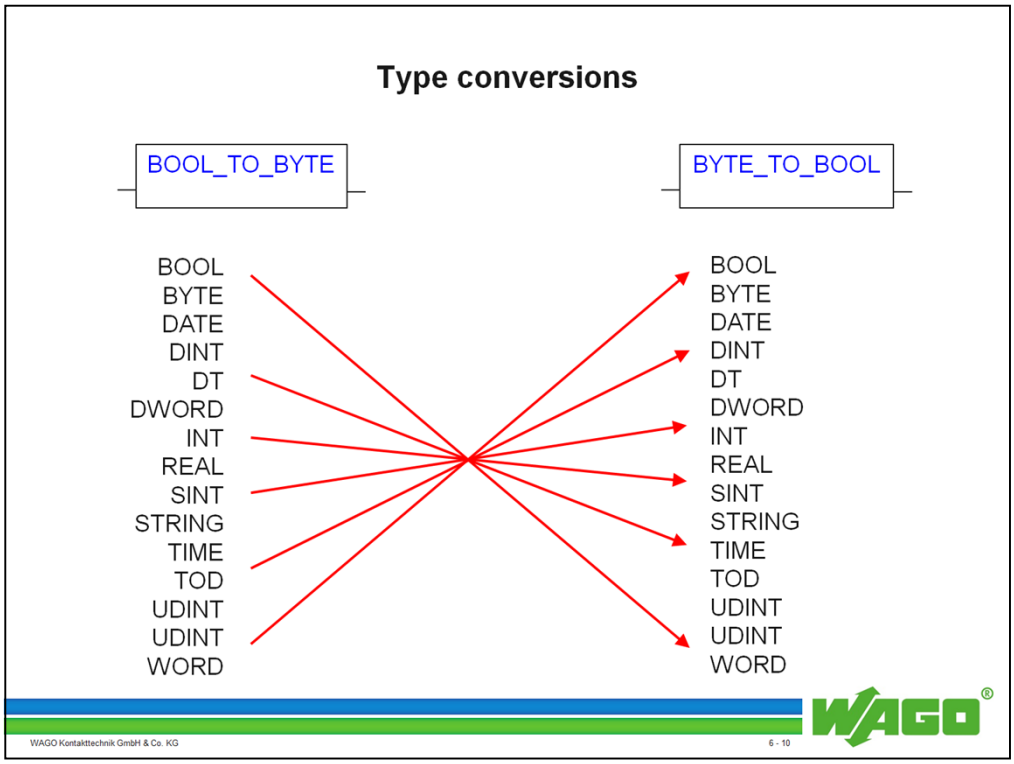
Set/ Reset (variants)

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.

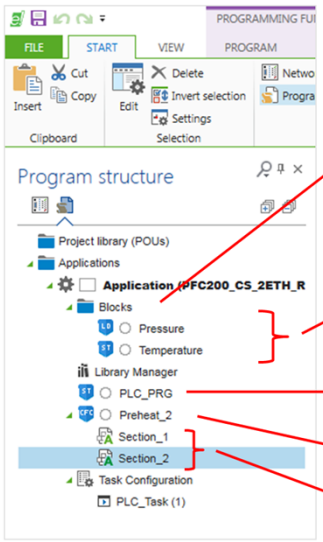
Variables and power off

	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)

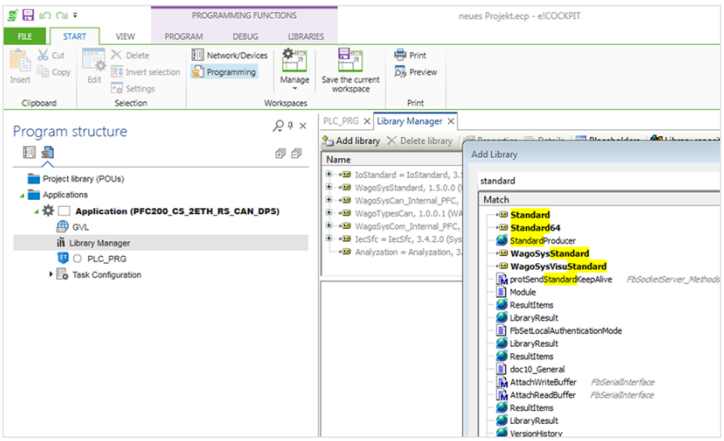
Different Function blocks

The main program
(Subroutines are called from here)

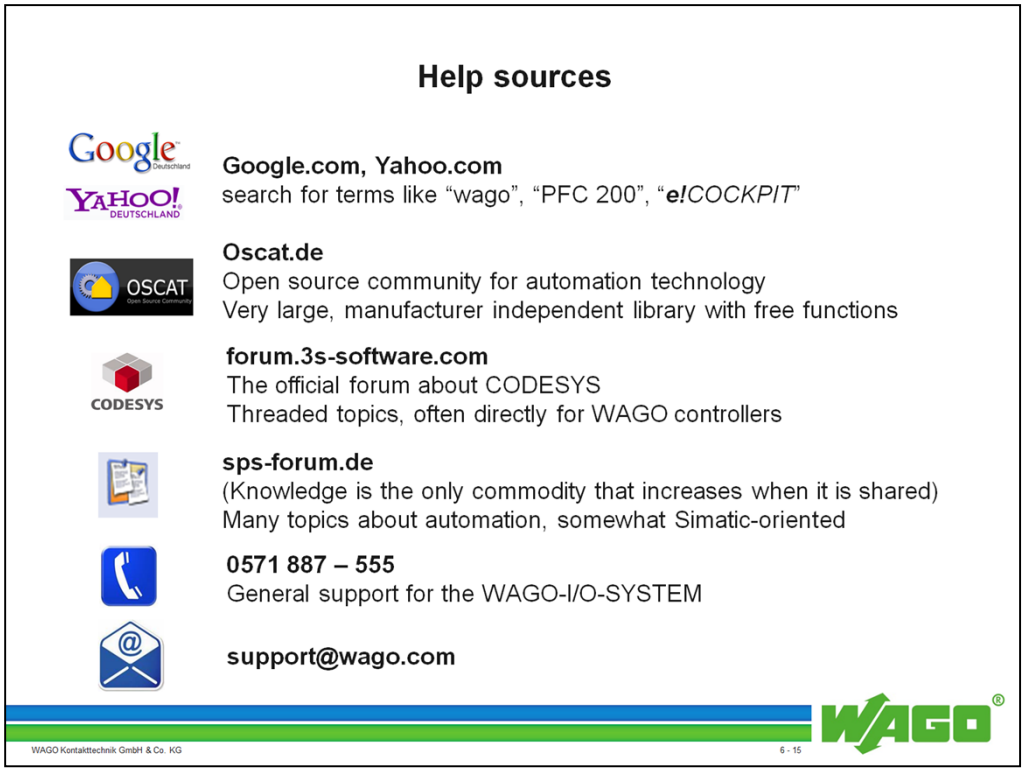
A subroutine

Action within a subroutine

Adding libraries



Using the search box makes it easier 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