Industrial Automation MKT4152

Introduction to

Programmable Logic Controllers

(PLC)

IEC 61131-3

STRUCTURED TEXT

Structured Text

- In ST, a program will start with “PROGRAM”, followed by its name, and end with “END_PROGRAM”
- Everything in between is your program code
- Re-execution after execution
  When program reaches end, and **PLC scan cycle** starts over again, this program will be executed again.
- ST is NOT CASE SENSITIVE:
  - Program
  - PROGRAM
  - PrOgRaM
- PROGRAM / END PROGRAM construct often not necessary with PLC programming software, as these are handled automatically

```
PROGRAM stexample
VAR
x : BOOL;
END_VAR
x := TRUE;
REPEAT
x := FALSE;
UNTIL x := FALSE;
END_REPEAT;
END_PROGRAM;
```

Syntax, Comments and Documentation

- Comments are marked with /*   bla bla */
  or with (* bla bla *)
- You should always document. A lot. Please
- Each line, i.e. each statement, has to be ended with a semicolon

  ;

  This is important, as ST uses this to know when a statement ends.
- Again, ST is
  NOT CASE SENSITIVE
- Space has no function. Use as many as you wish, especially to make code readable.

- When you upload your program to a PLC, it is compiled by the compiler, and therefore translated into machine code, that can be run on the PLC
- The compiler uses the syntax of the programming language to understand your program

```
PROGRAM stexample
VAR
x : BOOL;/* initialize variable */
END_VAR
x := TRUE;
/* you can also do comments
over multiple lines.
If you want, you can write the
story of your lifetime, copy and
paste FAUST or whatever */
REPEAT
x := FALSE;
UNTIL x := FALSE;
END_REPEAT;
END_PROGRAM;
```

Statements

- Statements tell what to do
- Ends with semicolon
- Statement is an assignment

- Example on the right hand side is telling the PLC to create a variable called X and that the variable should be a BOOL type.

- Variable: place, or address, which is used to store data, with a name easily readable for human beings. Ideally (Use names that make somehow sense, not:
  - var1
  - var2
  - varnew

- Variables are always of defined **data type**

```
X : BOOL;
x : bool;
x : bOOl;
```

Variables and their definitions

- Variables have to be defined in the header
- Variable definitions are made by

```
VAR
statement1;
statement2;
…
END_VAR
```

- Outside of this construct, no new variables can be declared!!!
- Different variable types exist, but will be defined later
- Some basic data types:
- BOOL: TRUE or FALSE, or 1 and 0
- INT: integer

- Variables can be named differently:
  Siemens: **symbols**
  Allen Bradley: **tags**
  IEC61131-3: **variables**

```
PROGRAM stexample
VAR
x : BOOL;
END_VAR
x := TRUE;
REPEAT
x := FALSE;
UNTIL x := FALSE;
END_REPEAT;
END_PROGRAM;
```

Data Types

Elementary data types:

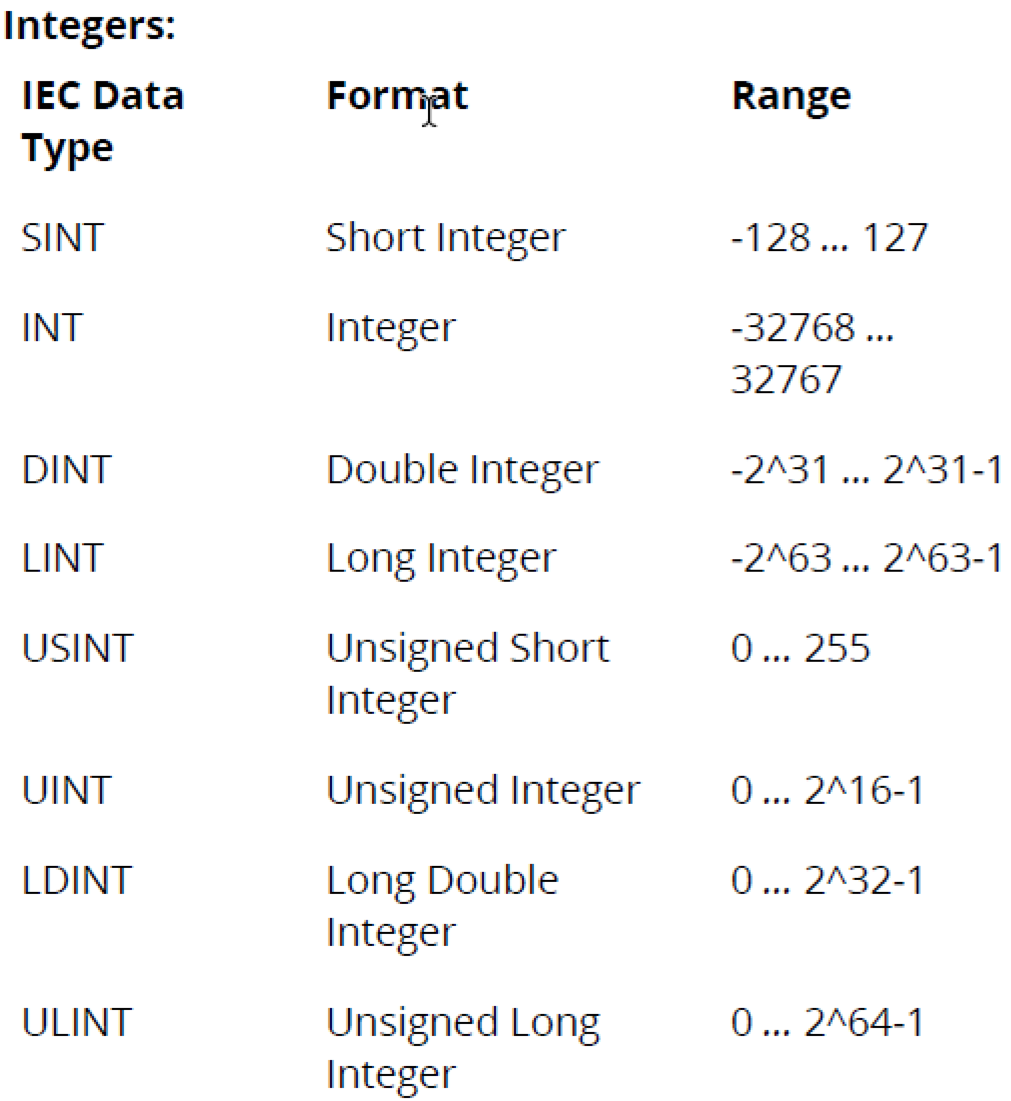
- Integers
- Floating points
- Time
- Bit strings

Derived Data Types

- User defined data types made of elementary data types
- Structured data types
- Enumerated data types
- Sub-ranges data types
- Array data types
- All the derived data types are built by making a construction of the keywords **TYPE** and **END_TYPE** .
  In between the keywords is the kind of derived data type you want to declare.

```
VAR
  a : bool;
  b : int;
  c : real;
END_VAR
```

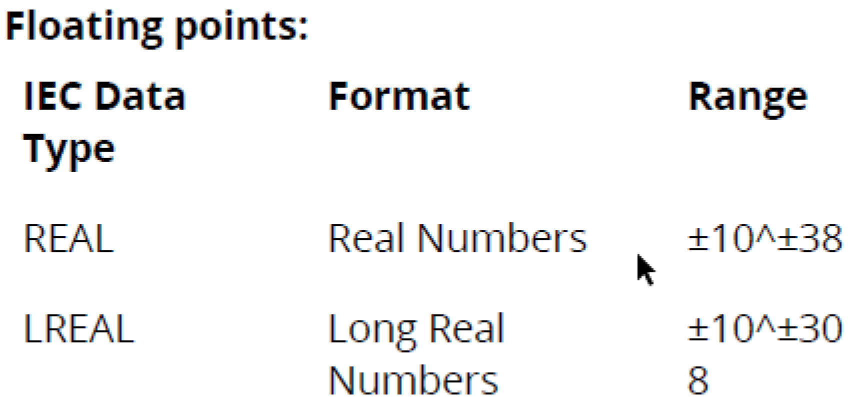
Data Types - Integers



Integers:

IEC Data Type	Format	Range
SINT	Short Integer	-128 ... 127
INT	Integer	-32768 ... 32767
DINT	Double Integer	-2^31 ... 2^31-1
LINT	Long Integer	-2^63 ... 2^63-1
USINT	Unsigned Short Integer	0 ... 255
UINT	Unsigned Integer	0 ... 2^16-1
LDINT	Long Double Integer	0 ... 2^32-1
ULINT	Unsigned Long Integer	0 ... 2^64-1

Data Types – Floating Points



Floating points:

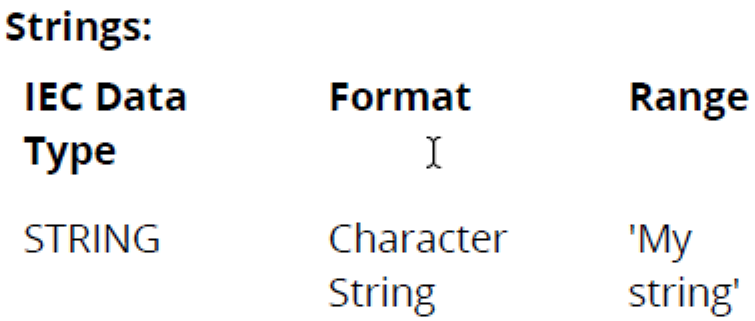
IEC Data Type	Format	Range
REAL	Real Numbers	±10^±38
LREAL	Long Real Numbers	±10^±308

Data Types – Time

Time:

IEC Data Type	**Format**	**Use**
TIME	Duration of time after an event	T#10d4h38m57s12ms TIME#10d4h38m
DATE	Calendar date	D#1989-05-22 DATE#1989-05-22
TIME_OF_DAY	Time of day	TOD#14:32:07 TIME_OF_DAY#14:32:07.77
DATE_AND_TIME	Date and time of day	DT#1989-06-15-13:56:14.77 DATE_AND_TIME#1989-06-15-13:56:14.77

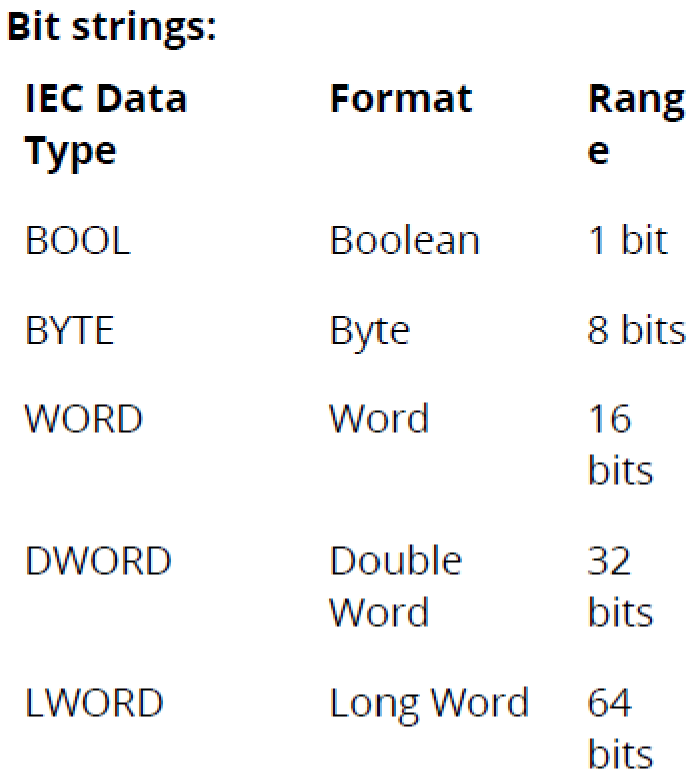
Data Types – Strings



Strings:

IEC Data Type	Format	Range
STRING	Character String	'My string'

Data Types – Bit Strings



Bit strings:

IEC Data Type	Format	Range
BOOL	Boolean	1 bit
BYTE	Byte	8 bits
WORD	Word	16 bits
DWORD	Double Word	32 bits
LWORD	Long Word	64 bits

Operators and Expressions I

- Operators are used to manipulate data
- Expression: construct that, when evaluated, yields a value
- When the compiler compiles an expression, it will evaluate the expression and replace the expression with the result.
- Expressions are composed of **operands** and **operators**
- Example RHS:
  A and B are operands
  + is the operator

```
X := 8;
Y := 16;

A+B;
```

Operators and Expressions II

Operation	Symbol	Binding strength	Execution ranking
Put in parentheses	(expression)	Strongest binding	1
Function call	Function name (parameter list)		2
Exponentiation	**		3
Negate (sign)	-		4
Building of complements	NOT		
Multiply	*		5
Divide	/		6
Modulo	MOD		7
Add	+		8
Subtract	-		9
Compare	<,>,<=,>=		10
Equal to	=		11
Not Equal to	<>		12
Boolean AND	AND		13
Boolean XOR	XOR		14
Boolean OR	OR	Weakest binding	15

Operators and Expressions III

A + B * MAX(C , D)

- Operator with the highest precedence is parenthesis
- Therefore, MAX(C,D) will be evaluated first.

A + B * D

- Assuming D > C
- B * D will be executed next, according to previously shown ranking table
- Last, A will be added to the results of (B * D)

Types of Operators & Types of Expressions

Four different types of operators

1. Arithmetic Operators

2. Relational Operators

3. Logical Operators

4. Bitwise Operators

Arithmetic Operators

- + (add)
- - (substract or negate)
- * (multiply)
- ** (exponent)
- / (divide)
- MOD (modulo divide)

- These represent math, and will result in math of specific data type
- Example:
- 15 MOD 4
- result:
- 3

Four different types of operators

1. Arithmetic Operators

2. Relational Operators

3. Logical Operators

4. Bitwise Operators

Relational Operators

To compare or find a relation between two values you can use one of the relational operators. They are used for comparison and the **result will be a boolean value (BOOL type), either TRUE or FALSE**.

= (equal)

< (less than)

<= (less than or equal)

> (greater than)

>= (greater than or equal)

<> (not equal)

Example:

```
TEMPERATURE := 93.9;
TEMPERATURE >= 100.0
```

Result:

FALSE

Types of Operators & Types of Expressions

Four different types of operators

1. Arithmetic Operators

2. Relational Operators

3. Logical Operators

4. Bitwise Operators

Logical Operators

If you want to compare boolean values (BOOL) and make some logic out of it, you have to use **logical operators** . These operators also yield a boolean value **of TRUE or FALSE** as a result of the expression.

Operands: boolean
Output: boolean

Operators:

- AND or &
- OR
- XOR
- NOT

Example:

LIMIT_SWITCH1 := TRUE;

LIMIT_SWITCH2 := FALSE;

LIMIT_SWITCH1 ***OR*** *LIMIT_SWITCH2*

Result:

TRUE

Types of Operators & Types of Expressions

Four different types of operators

1. Arithmetic Operators

2. Relational Operators

3. Logical Operators

4. Bitwise Operators

Bitwise Operators

Operations are performed bitwise.
A logic function is performed for each bit of two numbers. The result is a new number, the result of the bitwise operations.

Operators: AND, OR, XOR, NOT

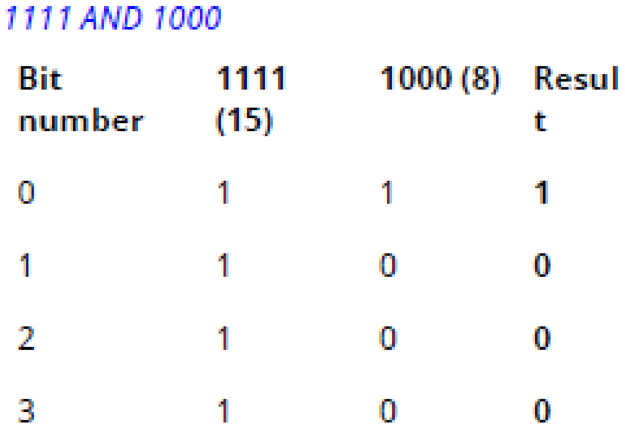
Example: 15 & 8

Result : 8

To understand what is going on, we convert the numbers into binary:

15 = 1111

8 = 1000



1111 AND 1000

Bit number	1111 (15)	1000 (8)	Result
0	1	1	1
1	1	0	0
2	1	0	0
3	1	0	0

Assignment Statement and Operator

- Assignment: most fundamental statement

- Common mistake: confuse assignment with relational operator
- Example: A=B
  This is not an assignment, but a relational expression. The output is either TRUE or FALSE
- Corrected Example: A:=B
  A is assigned the value of B

```
A := B;
/* A is assigned the current
value of B */

A:=10;
/* A is assigned the value of
10 */

B := A + 2;
/* B is assigned the value of
A plus 2, resulting in 12 */
```

Conditional Statements

- Conditional statement (IF statement)

```
IF [boolean expression] THEN
<statement>;
ELSIF [boolean expression] THEN
<statement>;
ELSE
<statement>;
END_IF;
```

Conditional Statements

- Multiple expressions can be combined

```
IF (INPUT1) AND (INPUT2)
THEN
OUTPUT1 := TRUE;
END_IF;
```

CASE Statements

Handling of numeric cases of an expression

```
CASE [numeric expression] OF
result1: <statement>;
resultN: <statemtent>;
ELSE
<statement>;
END_CASE;
```

```
PROGRAM_STEP := 3;

CASE PROGRAM_STEP OF
1: PROGRAM_STEP :=PROGRAM_STEP+1;
2: PROGRAM_STEP :=PROGRAM_STEP+2;
3: PROGRAM_STEP := PROGRAM_STEP+3;
ELSE
PROGRAM_STEP := PROGRAM_STEP+10;
END_CASE;
```

Iteration with Repeating Loops: FOR LOOPS

These are used to execute a statement multiple times

Syntax:

```
FOR count := initial_value TO final_value BY increment DO
<statement>;
END_FOR;
```

The loop is executed a specified number of times.

However, it is possible to exit the loop with „exit“.
Example:

```
FOR count := initial_value TO final_value BY increment DO
<statement>;
IF [boolean expression] THEN
EXIT;
END_IF;
END_FOR;
```

```
A := 0;
FOR count:= 0 TO 20 BY 5 DO
A := A + count;
END_FOR;

/* result: A = 50 */
```

Iteration with Repeating Loops: WHILE LOOPS

Syntax:

```
WHILE [boolean expression] DO
<statement>;
END_WHILE;
```

Keeps repeating as long as expression remains TRUE

Best practise:
Insert an if statement with EXIT into loop to prevent running infinitely:

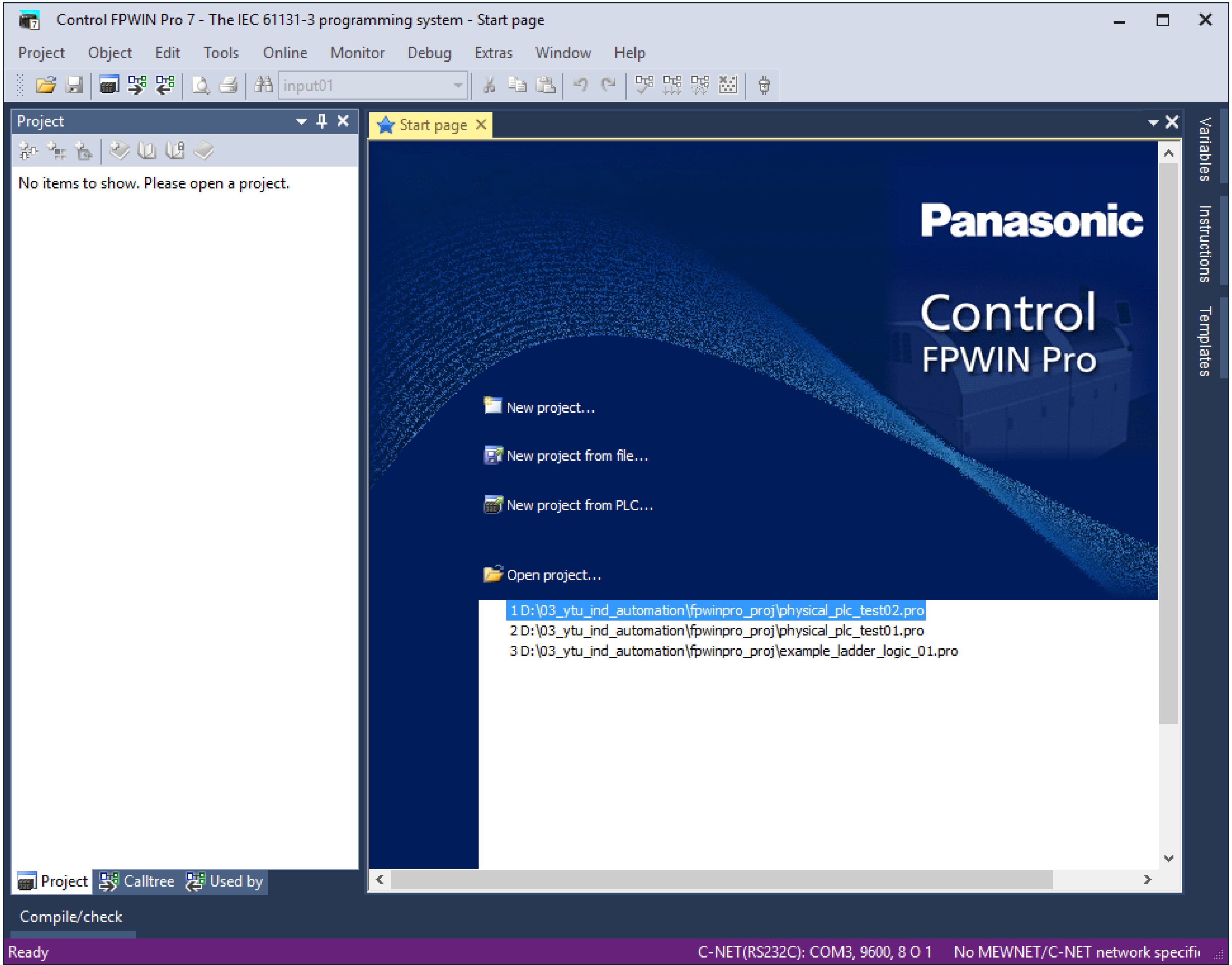
```
IF [boolean expression] THEN
EXIT;
END_IF;
```

```
counter := 0;
WHILE counter < 10 DO
counter := counter + 1;
machine_status := counter * 10;
END_WHILE;
```

IEC61131-3 Application Example

PANASONIC CONTROL FP WIN PRO

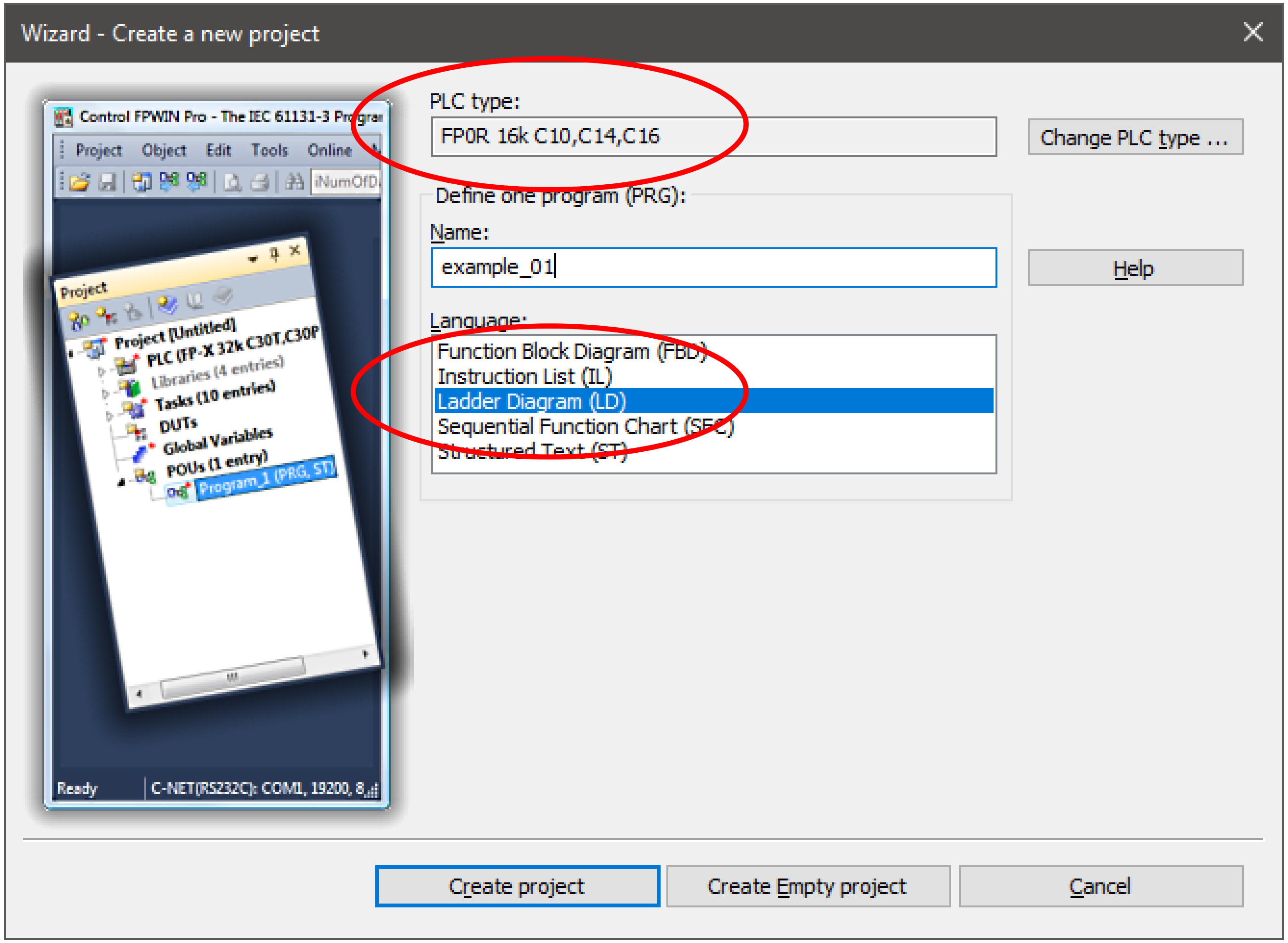
Start Screen



1 D:\03_ytu_ind_automation\fpwinpro_proj\physical_plc_test02.pro
2 D:\03_ytu_ind_automation\fpwinpro_proj\physical_plc_test01.pro
3 D:\03_ytu_ind_automation\fpwinpro_proj\example_ladder_logic_01.pro

Ready C-NET(RS232C): COM3, 9600, 8 O 1 No MEWNET/C-NET network specifi

Creating a project



Wizard - Create a new project

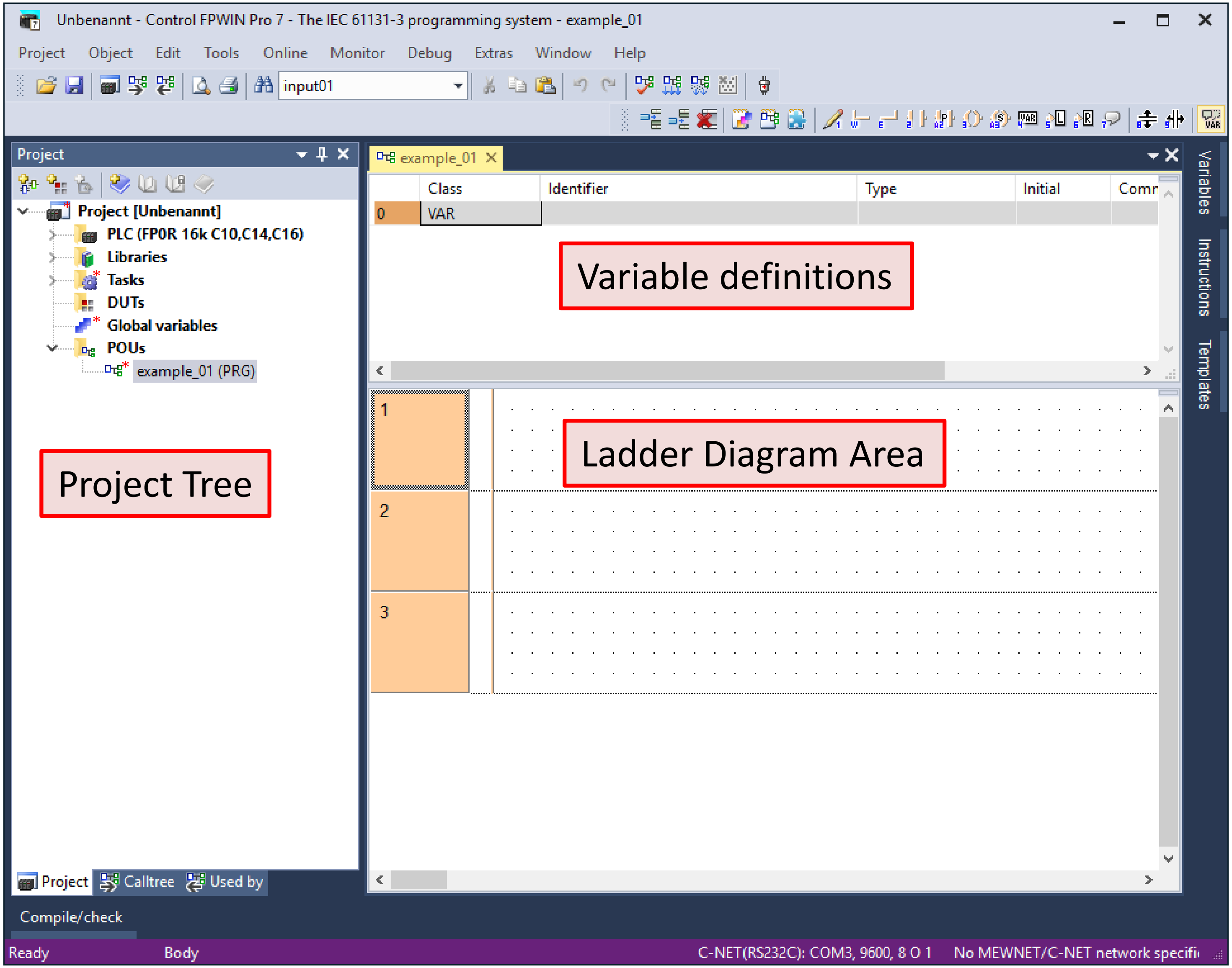
PLC type: FP0R 16k C10,C14,C16

Define one program (PRG):

Name: example_01

Language:

- Function Block Diagram (FBD)
- Instruction List (IL)
- Ladder Diagram (LD)
- Sequential Function Chart (SFC)
- Structured Text (ST)



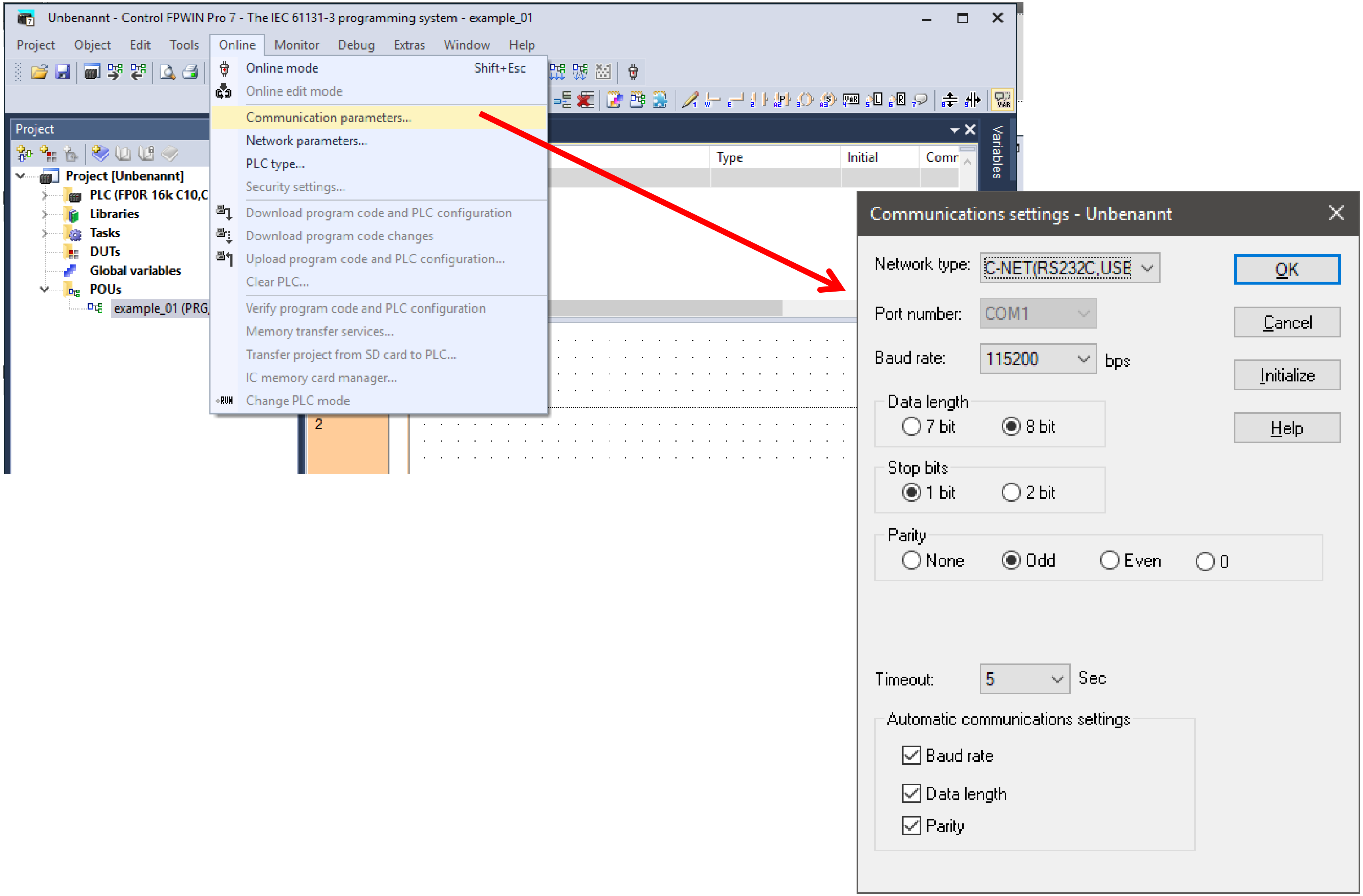
	Class	Identifier	Type	Initial	Comr
0	VAR				

Variable definitions

Ladder Diagram Area

Project Tree

Setting Communication Parameters


Unbenannt - Control FPWIN Pro 7 - The IEC 61131-3 programming system - example_01
Project
Object
Edit
Tools
Online
Monitor
Debug
Extras
Window
Help
Online mode Shift+Esc
Online edit mode
Communication parameters...
Network parameters...
PLC type...
Security settings...
Download program code and PLC configuration
Download program code changes
Upload program code and PLC configuration...
Clear PLC...
Verify program code and PLC configuration
Memory transfer services...
Transfer project from SD card to PLC...
IC memory card manager...
Change PLC mode
Project
Project [Unbenannt]
PLC (FP0R 16k C10,C
Libraries
Tasks
DUTs
Global variables
POUs
example_01 (PRG
Type
Initial
Comr
Variables
2
Communications settings - Unbenannt
OK
Cancel
Initialize
Help

Network type: C-NET(RS232C,USE

Port number: COM1

Baud rate: 115200 bps

Data length: ○ 7 bit ● 8 bit

Stop bits: ● 1 bit ○ 2 bit

Parity: ○ None ● Odd ○ Even ○ 0

Timeout: 5 Sec

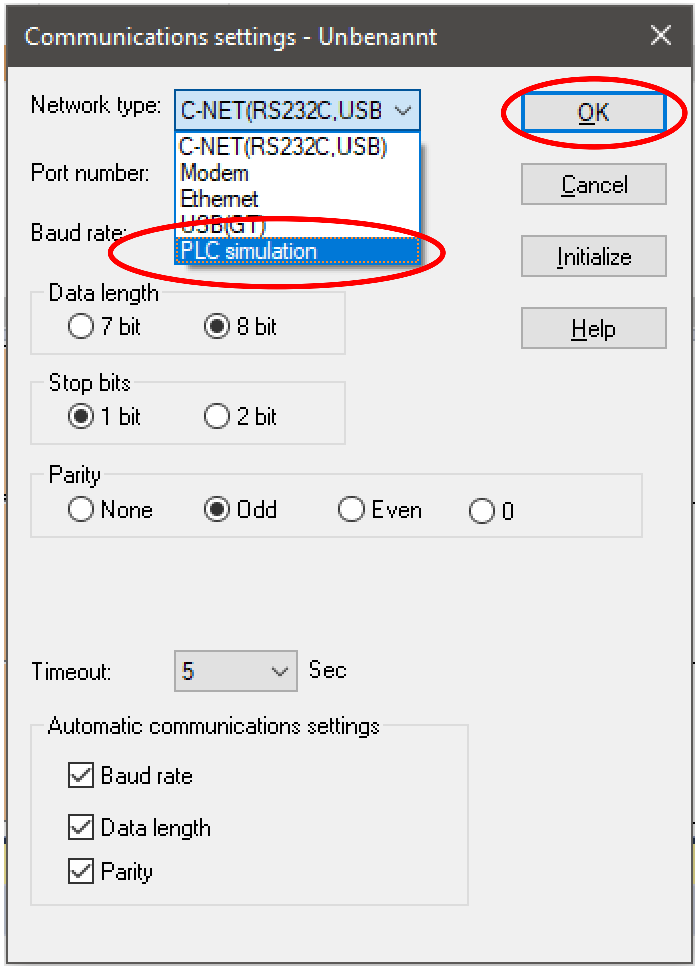
Automatic communications settings

☑ Baud rate

☑ Data length

☑ Parity

Communication Parameters: PLC Simulation



Communications settings - Unbenannt

Network type: C-NET(RS232C,USB

C-NET(RS232C,USB)
Modem
Ethernet
USB(GT)
PLC simulation

Port number:

Baud rate:

Data length: ○ 7 bit ● 8 bit

Stop bits: ● 1 bit ○ 2 bit

Parity: ○ None ● Odd ○ Even ○ 0

Timeout: 5 Sec

Automatic communications settings

☑ Baud rate

☑ Data length

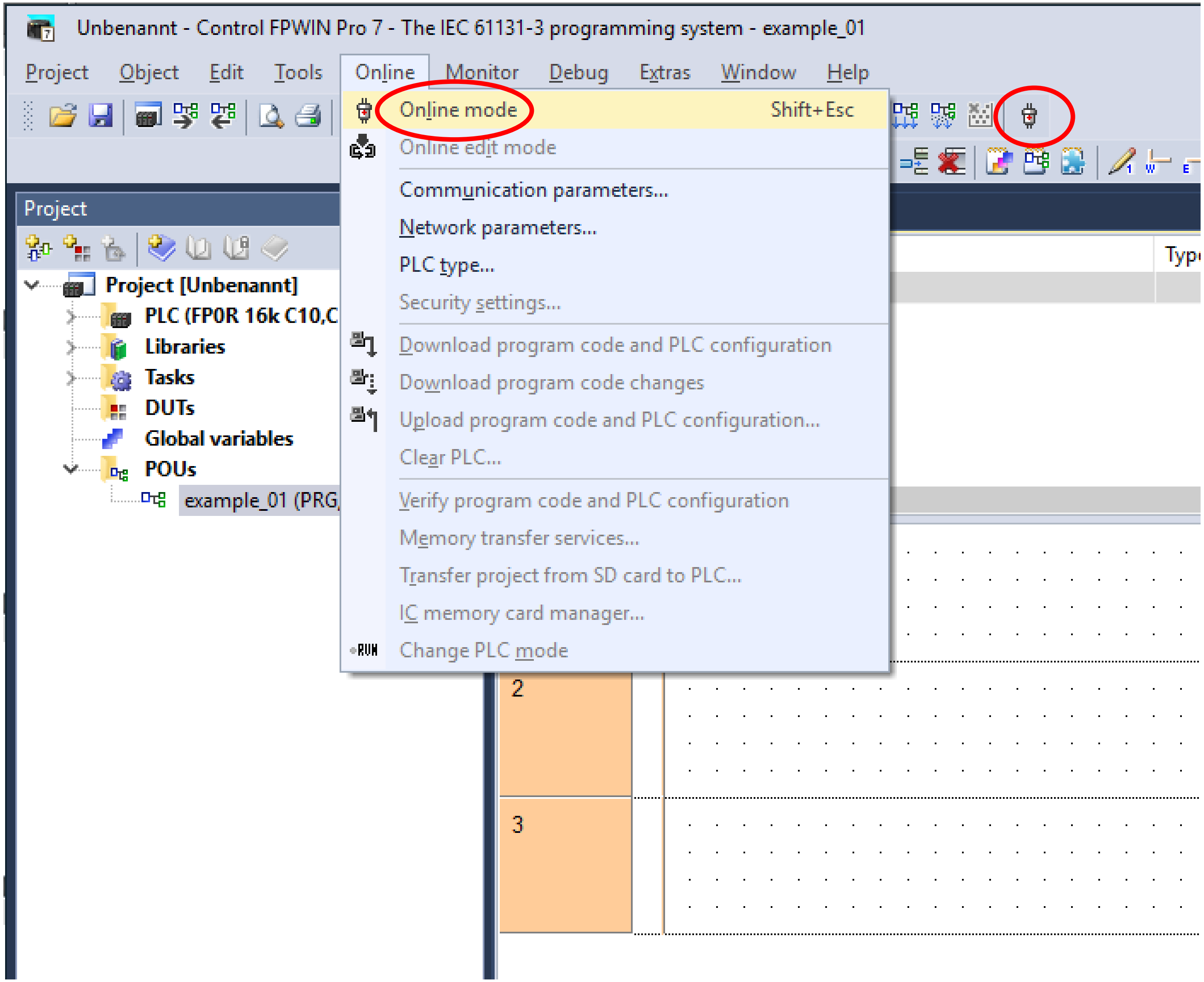
☑ Parity

Operation Modes of PLC Software (General)

- Offline Mode
  - Used for project management, programming, setting parameters

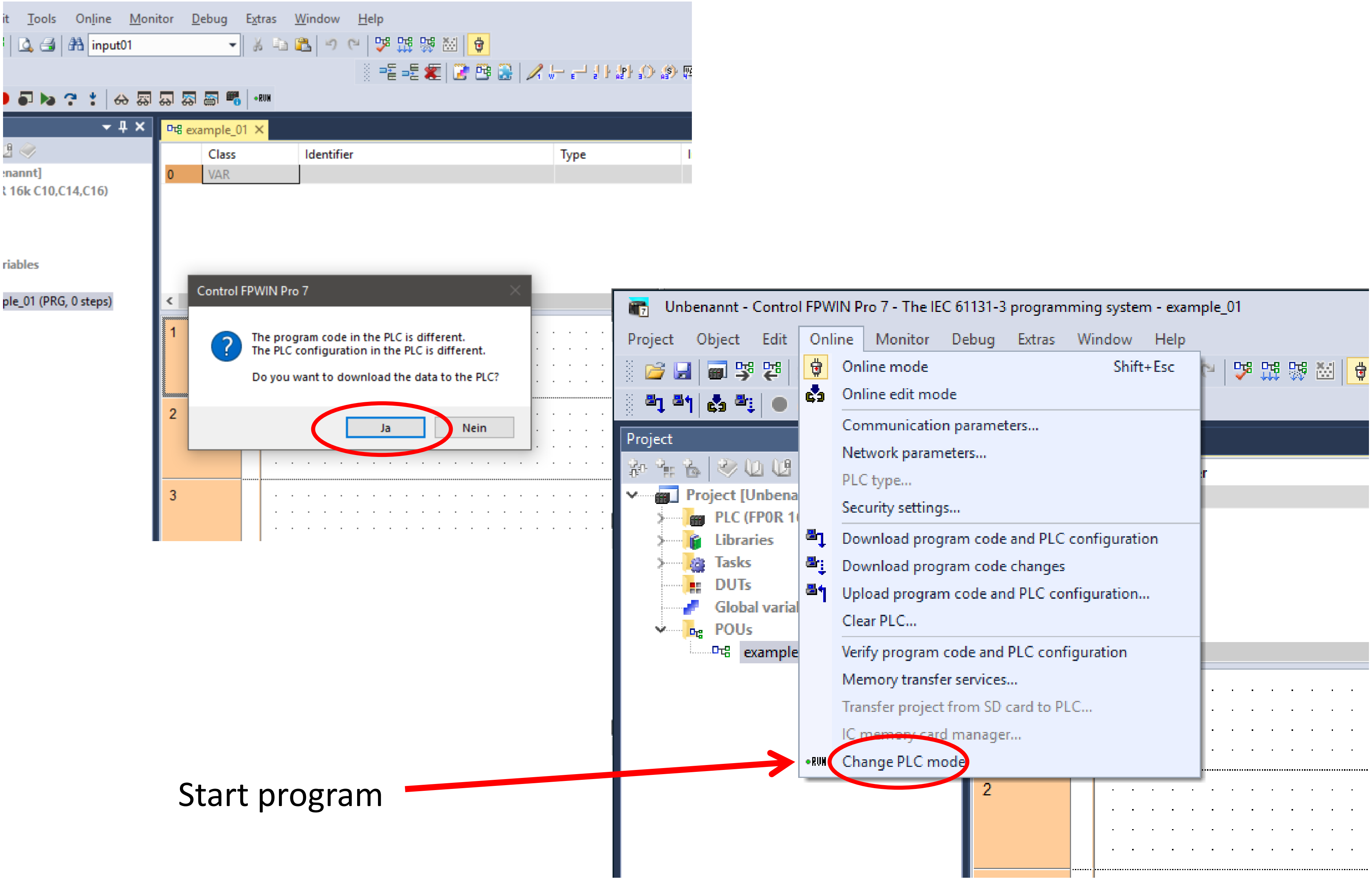
- Online Mode
  - Connects User interface to PLC target (Simulated or physical)
  - Start/stop downloaded PLC program
  - Watch parameters
  - Change logic state of variables

Operation Modes: Changing to online mode



Unbenannt - Control FPWIN Pro 7 - The IEC 61131-3 programming system - example_01

Operation Modes: Changing to online mode



Control FPWIN Pro 7

The program code in the PLC is different.
The PLC configuration in the PLC is different.

Do you want to download the data to the PLC?

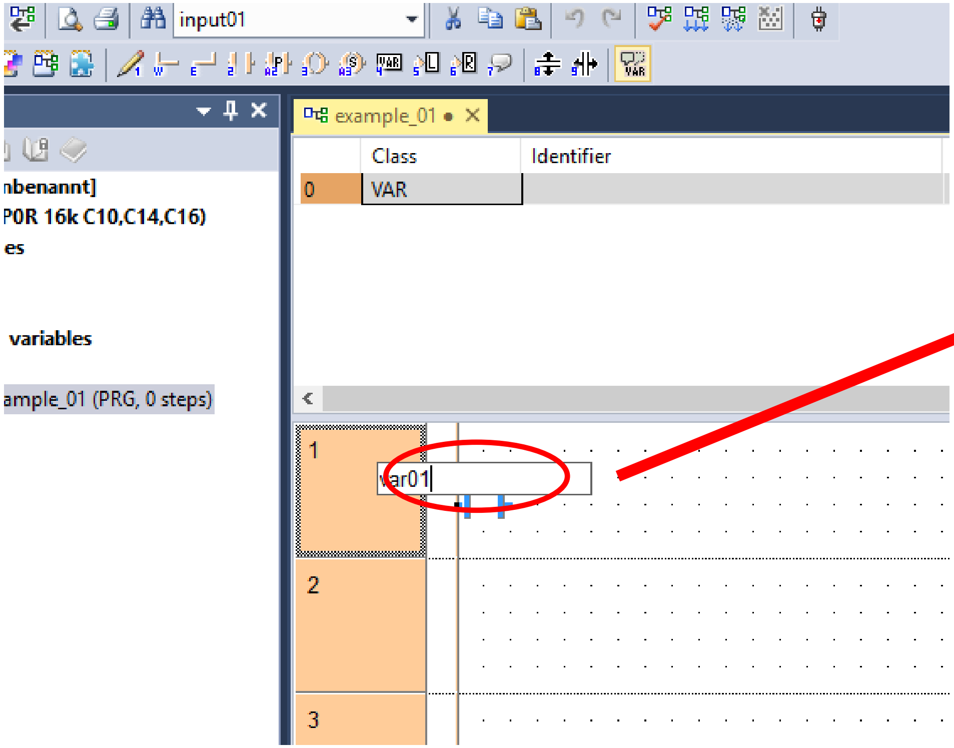
Unbenannt - Control FPWIN Pro 7 - The IEC 61131-3 programming system - example_01

Project
Object
Edit
Online
Monitor
Debug
Extras
Window
Help
Online mode Shift+Esc
Online edit mode
Communication parameters...
Network parameters...
PLC type...
Security settings...
Download program code and PLC configuration
Download program code changes
Upload program code and PLC configuration...
Clear PLC...
Verify program code and PLC configuration
Memory transfer services...
Transfer project from SD card to PLC...
IC memory card manager...
Change PLC mode

Start program

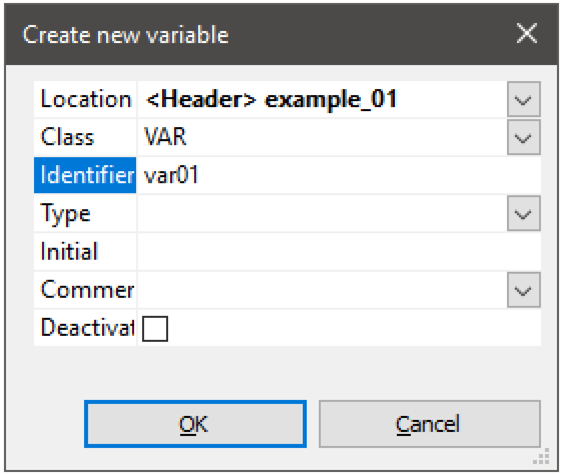
Example Program

Programming (Making changes) requires OFF-LINE MODE!!!!



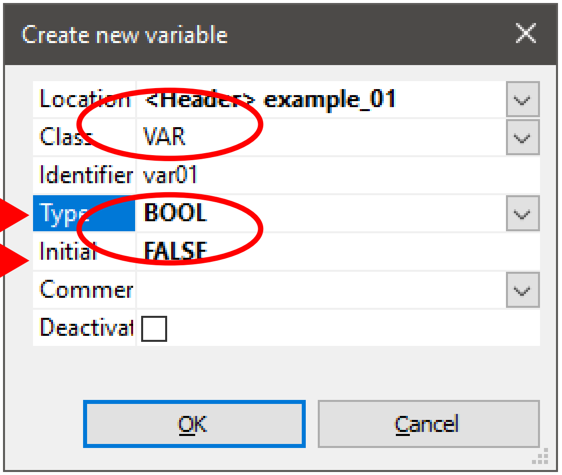
	Class	Identifier
0	VAR	

var01



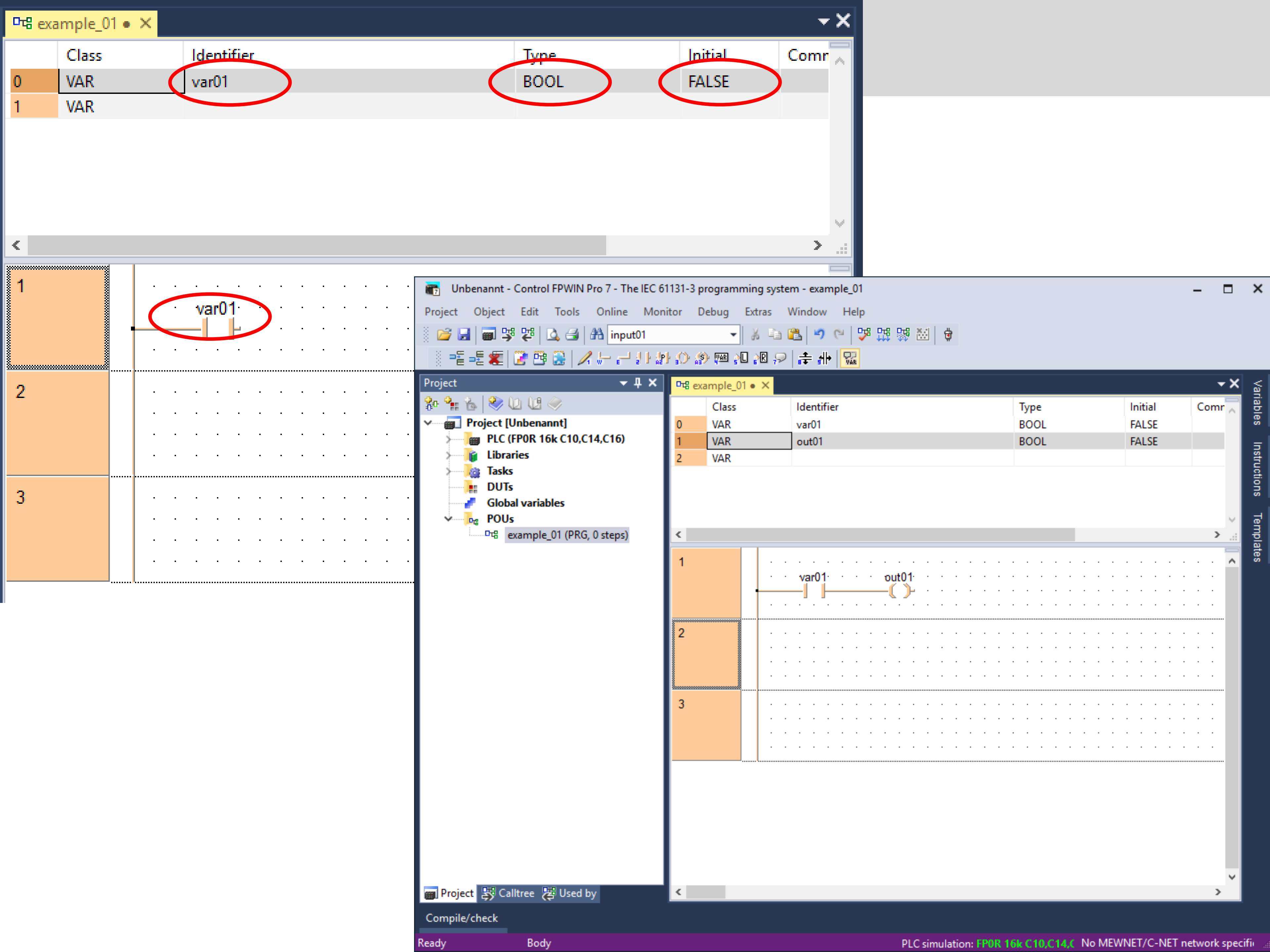
Create new variable

Location	<Header> example_01
Class	VAR
Identifier	var01
Type	
Initial	
Commer	
Deactivat	☐



Create new variable

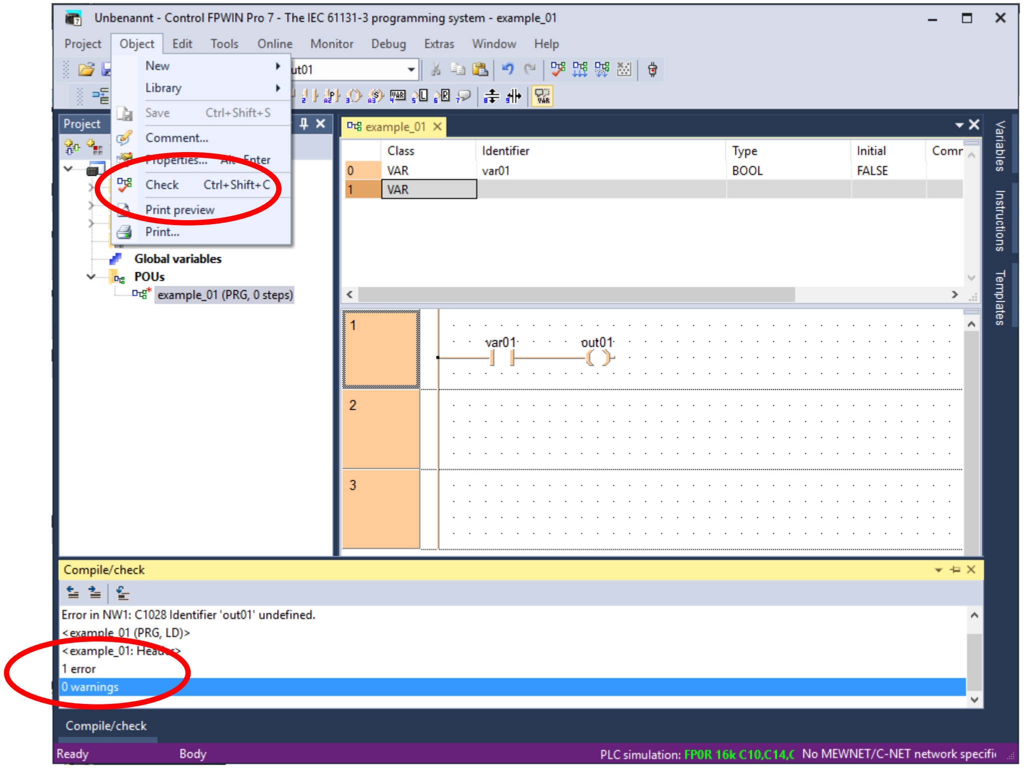
Location	<Header> example_01
Class	VAR
Identifier	var01
Type	BOOL
Initial	FALSE
Commer	
Deactivat	☐



	Class	Identifier	Type	Initial	Comr
0	VAR	var01	BOOL	FALSE	
1	VAR				

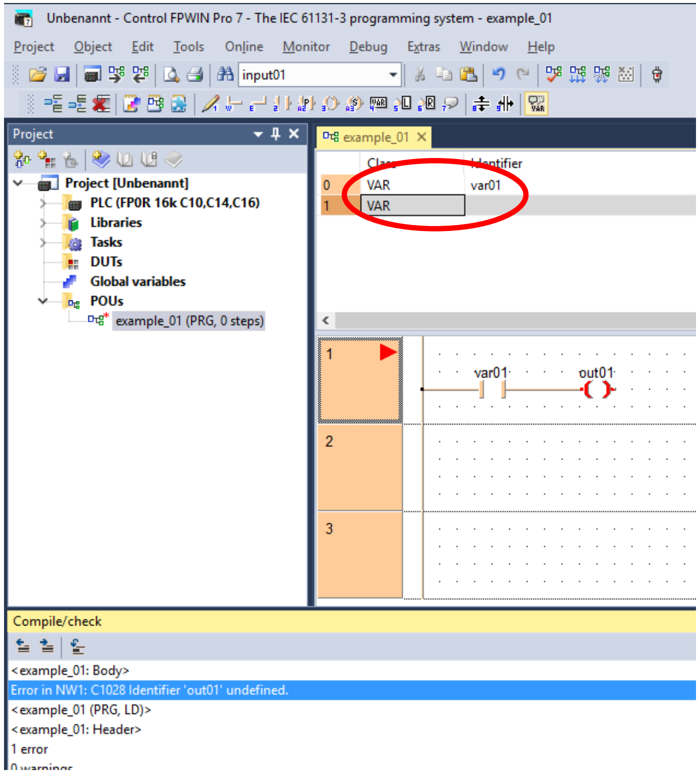
	Class	Identifier	Type	Initial	Comr
0	VAR	var01	BOOL	FALSE	
1	VAR	out01	BOOL	FALSE	
2	VAR				

Code Verification



	Class	Identifier	Type	Initial	Comr
0	VAR	var01	BOOL	FALSE	
1	VAR				

```
Error in NW1: C1028 Identifier 'out01' undefined.
<example_01 (PRG, LD)>
<example_01: Header>
1 error
0 warnings
```

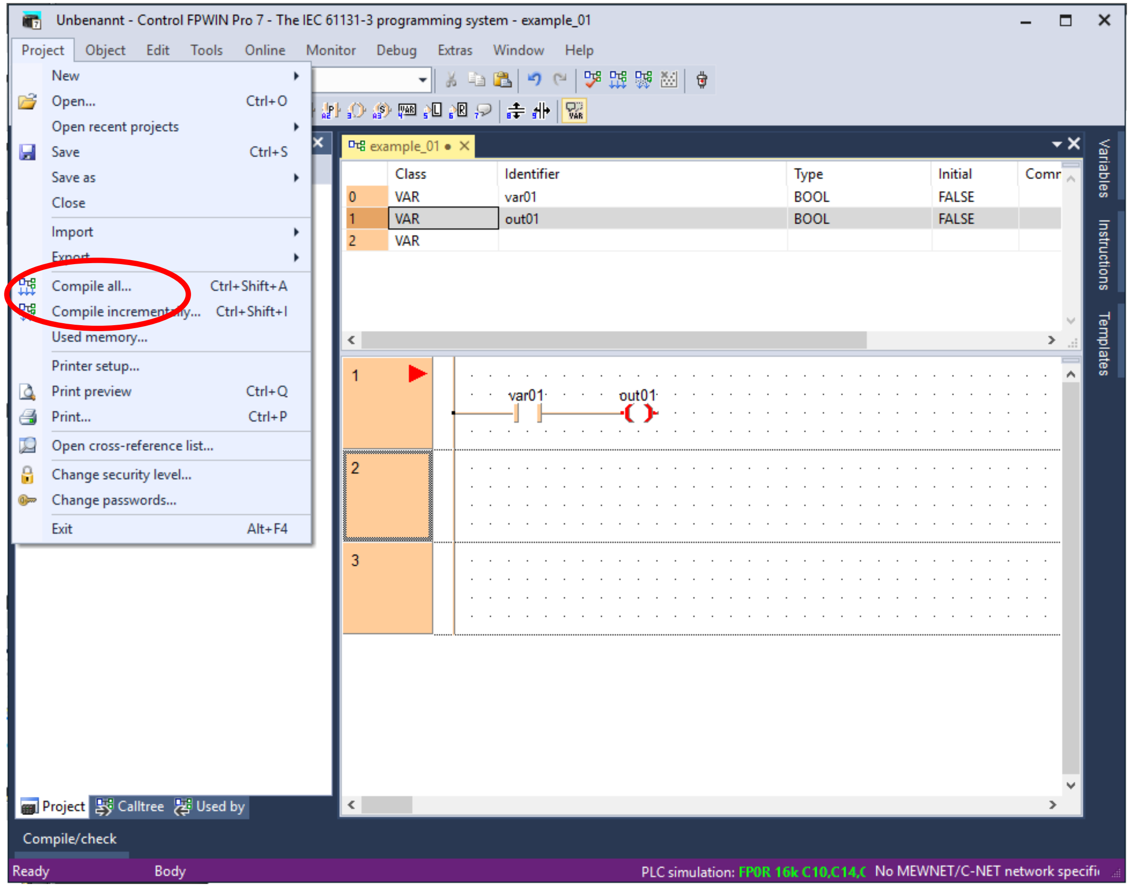


	Class	Identifier
0	VAR	var01
1	VAR	

```
<example_01: Body>
Error in NW1: C1028 Identifier 'out01' undefined.
<example_01 (PRG, LD)>
<example_01: Header>
1 error
0 warnings
```

Error: variable out01 undefined!!!!

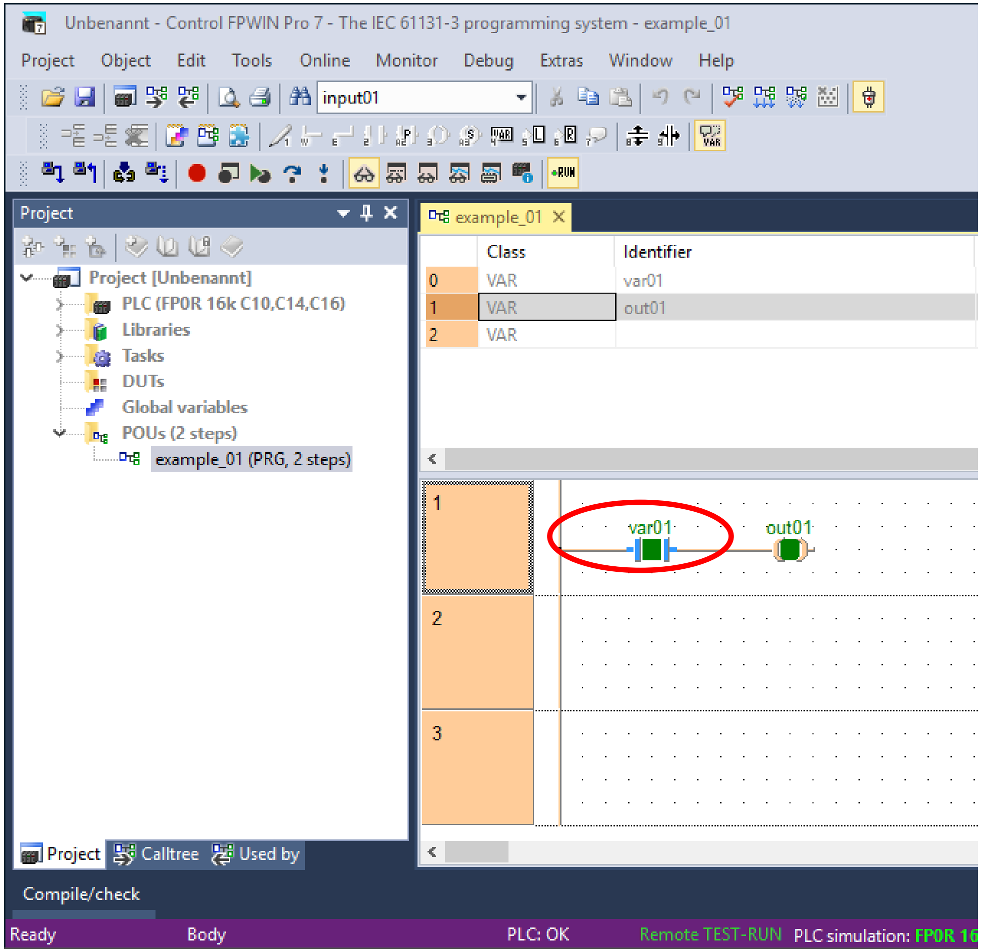
Compilation



	Class	Identifier	Type	Initial	Comr
0	VAR	var01	BOOL	FALSE	
1	VAR	out01	BOOL	FALSE	
2	VAR				

Ready Body PLC simulation: FP0R 16k C10,C14,C No MEWNET/C-NET network specifi

Testing Program on PLC Simulation



Unbenannt - Control FPWIN Pro 7 - The IEC 61131-3 programming system - example_01

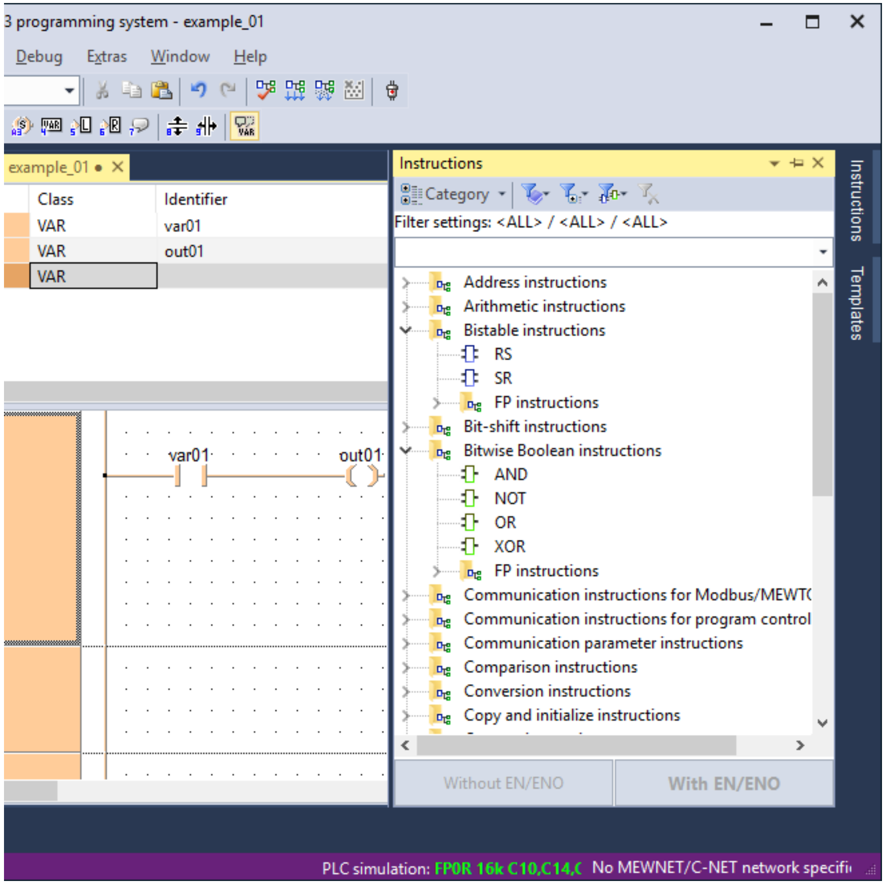
	Class	Identifier
0	VAR	var01
1	VAR	out01
2	VAR	

Double-clicking on contacts changes variable state.

Values of variable sare displayed in code visualization

Instructions

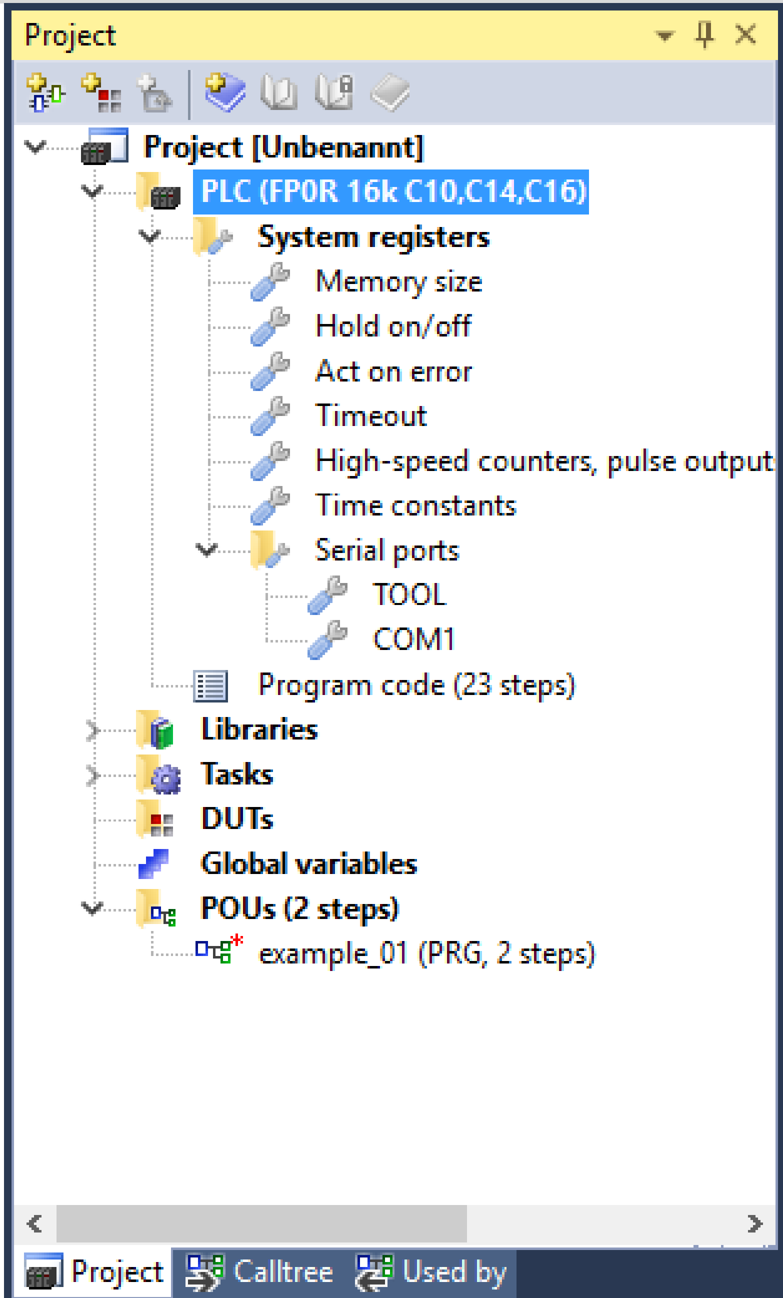
- Instructions are function blocks in general
- Names are conforming with IEC61131-3
- Example blocks
  - AND
  - OR
  - XOR
  - RS
  - SR
  - TOF
  - TON


3 programming system - example_01
Debug
Extras
Window
Help
example_01 ●
Instructions
Category
Filter settings: <ALL> / <ALL> / <ALL>
Address instructions
Arithmetic instructions
Bistable instructions
RS
SR
FP instructions
Bit-shift instructions
Bitwise Boolean instructions
AND
NOT
OR
XOR
FP instructions
Communication instructions for Modbus/MEWT(
Communication instructions for program control
Communication parameter instructions
Comparison instructions
Conversion instructions
Copy and initialize instructions
Without EN/ENO
With EN/ENO
var01
out01
Instructions
Templates
PLC simulation: FP0R 16k C10,C14,C
No MEWNET/C-NET network specifi

Class	Identifier
VAR	var01
VAR	out01
VAR	

Project Tree Categories – PLC target

- General information on target device
- Specific info, counters, programming interfaces

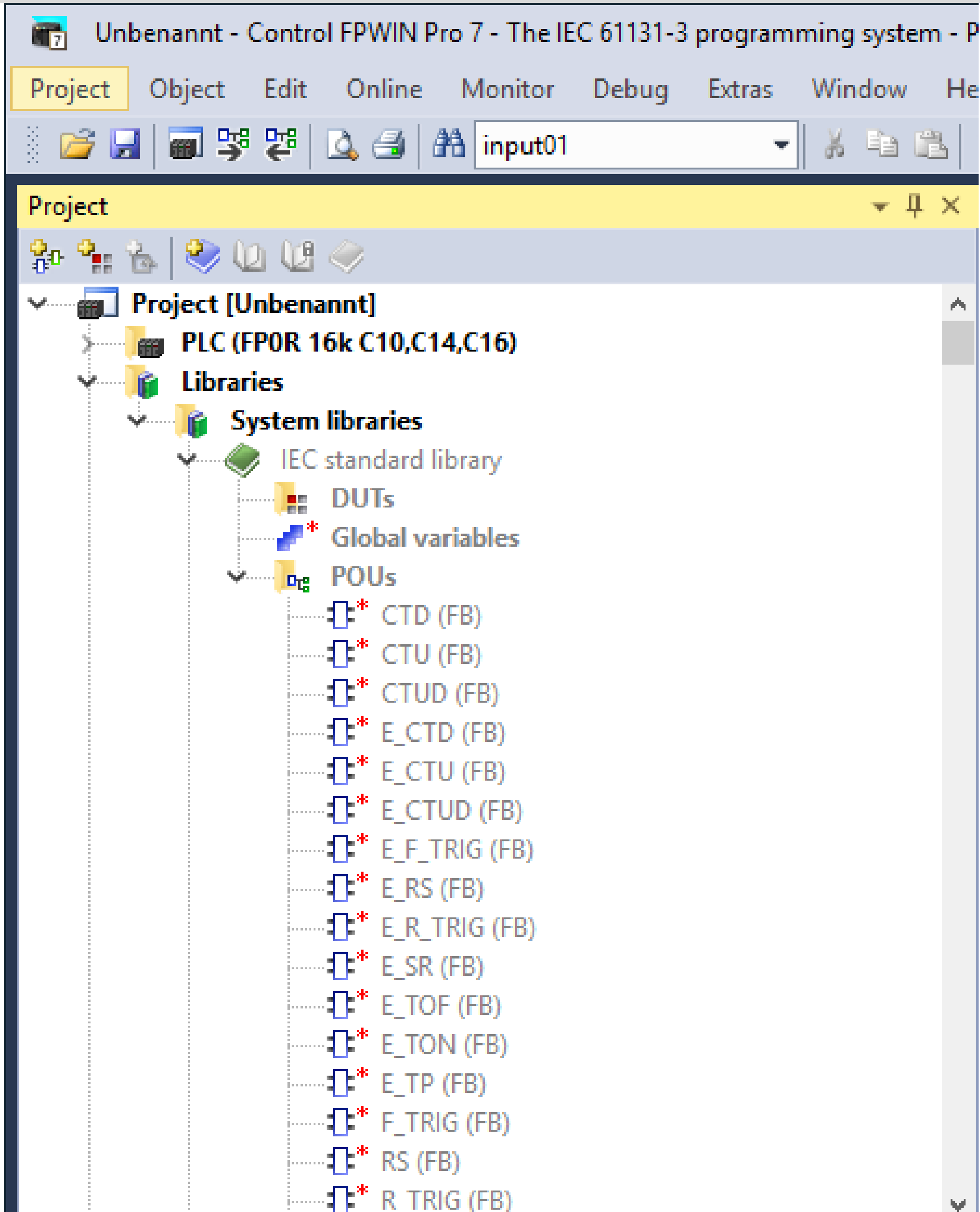


Project

- Project [Unbenannt]
  - PLC (FP0R 16k C10,C14,C16)
    - System registers
      - Memory size
      - Hold on/off
      - Act on error
      - Timeout
      - High-speed counters, pulse output
      - Time constants
      - Serial ports
        - TOOL
        - COM1
    - Program code (23 steps)
  - Libraries
  - Tasks
  - DUTs
  - Global variables
  - POUs (2 steps)
    - example_01 (PRG, 2 steps)

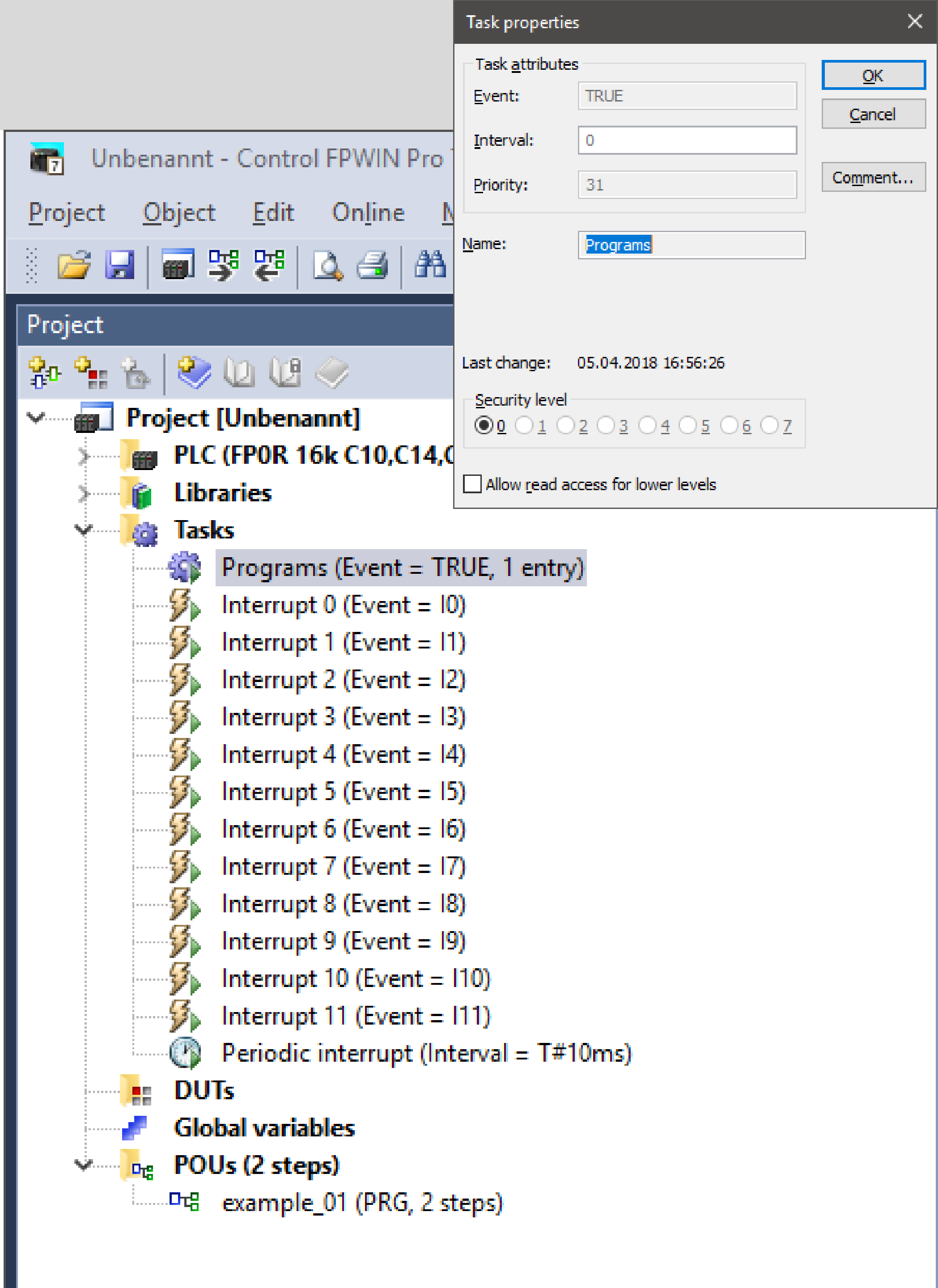
Project Tree Categories - Libraries

- Shows installed available libraries
- Example: IEC standard library contains common blocks, such as CTU CTD, RS, TON, ….
- Custom libraries can be made from custom POU‘s
- Additional libraries from Panasonic can be installed



Project Tree Categories - Tasks

- Tasks shows exectuted POU‘s, interrupts and periodic interrupts
- „Programs“ entry
  assigned POU‘s are continuously re-executed as fast as possible
- „Periodic interrupt“
  assigned POU‘s are executed cyclic, with specified interval, here 10ms.
- Each task is assigned a priority
- Interrupts have higher priority than „programs“ or „periodic interrupts“



Task properties

Task attributes

Event:	TRUE
Interval:	0
Priority:	31
Name:	Programs
Last change:	05.04.2018 16:56:26

Security level
● 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

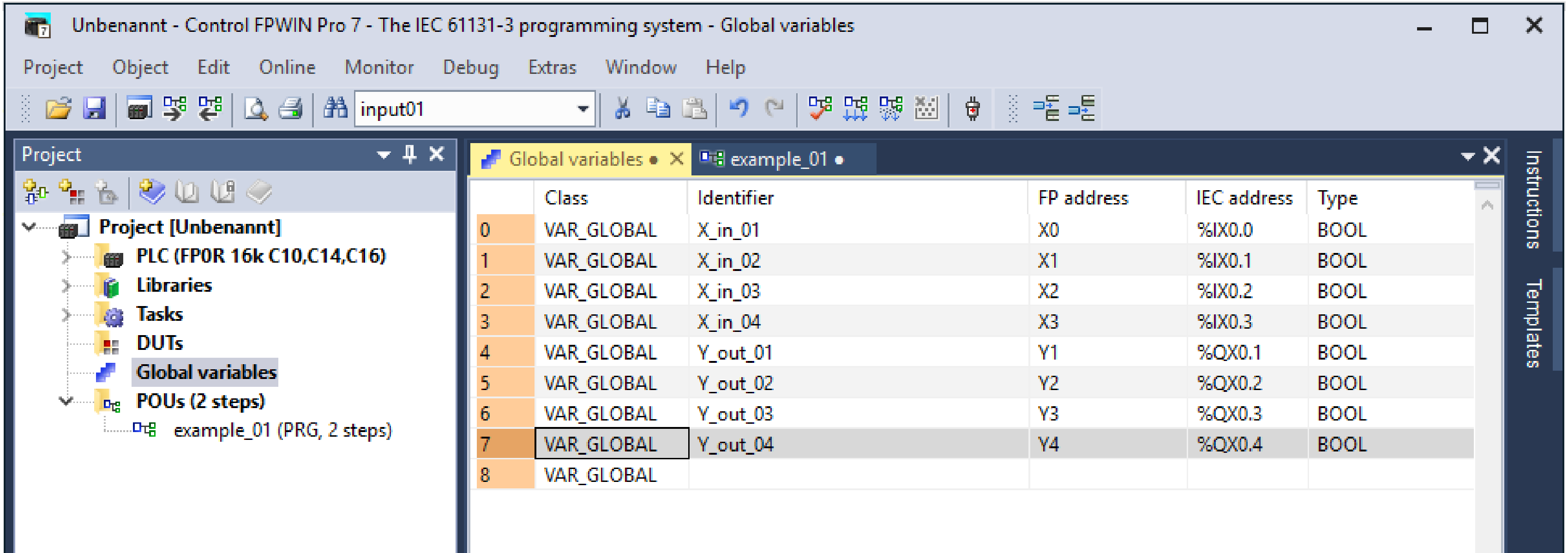
□ Allow read access for lower levels

Project

- Project [Unbenannt]
  - PLC (FP0R 16k C10,C14,C
  - Libraries
  - Tasks
    - Programs (Event = TRUE, 1 entry)
    - Interrupt 0 (Event = I0)
    - Interrupt 1 (Event = I1)
    - Interrupt 2 (Event = I2)
    - Interrupt 3 (Event = I3)
    - Interrupt 4 (Event = I4)
    - Interrupt 5 (Event = I5)
    - Interrupt 6 (Event = I6)
    - Interrupt 7 (Event = I7)
    - Interrupt 8 (Event = I8)
    - Interrupt 9 (Event = I9)
    - Interrupt 10 (Event = I10)
    - Interrupt 11 (Event = I11)
    - Periodic interrupt (Interval = T#10ms)
  - DUTs
  - Global variables
  - POUs (2 steps)
    - example_01 (PRG, 2 steps)

Project Tree Categories - Global Variable List

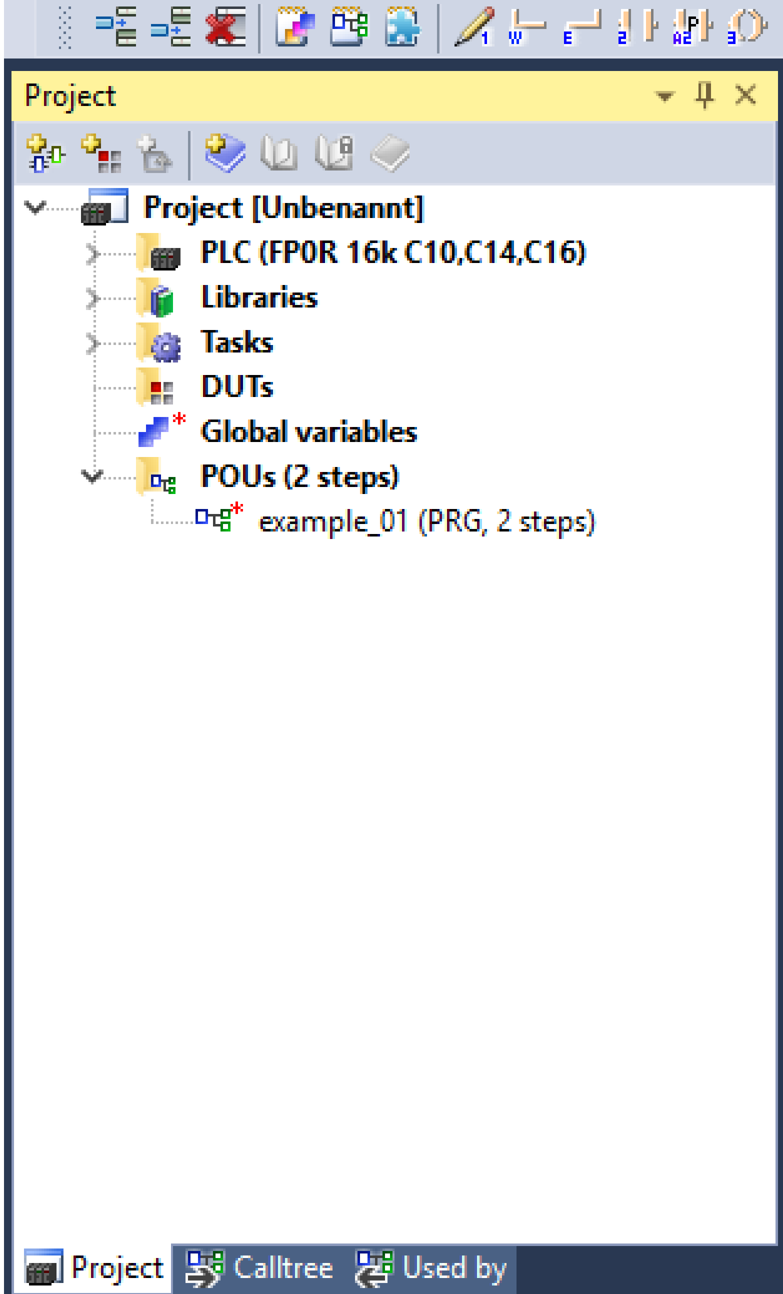
- Global variables can be accessed by any POU (program or function)
- Hardware addresses are advised to specify as global variables
- FP address refers to panasonic nomenclature
- IEC address is the same adress, but in IEC notation
- Type defines datatype, BOOL for boolean



	Class	Identifier	FP address	IEC address	Type
0	VAR_GLOBAL	X_in_01	X0	%IX0.0	BOOL
1	VAR_GLOBAL	X_in_02	X1	%IX0.1	BOOL
2	VAR_GLOBAL	X_in_03	X2	%IX0.2	BOOL
3	VAR_GLOBAL	X_in_04	X3	%IX0.3	BOOL
4	VAR_GLOBAL	Y_out_01	Y1	%QX0.1	BOOL
5	VAR_GLOBAL	Y_out_02	Y2	%QX0.2	BOOL
6	VAR_GLOBAL	Y_out_03	Y3	%QX0.3	BOOL
7	VAR_GLOBAL	Y_out_04	Y4	%QX0.4	BOOL
8	VAR_GLOBAL				

Project Tree Categories – POU‘s

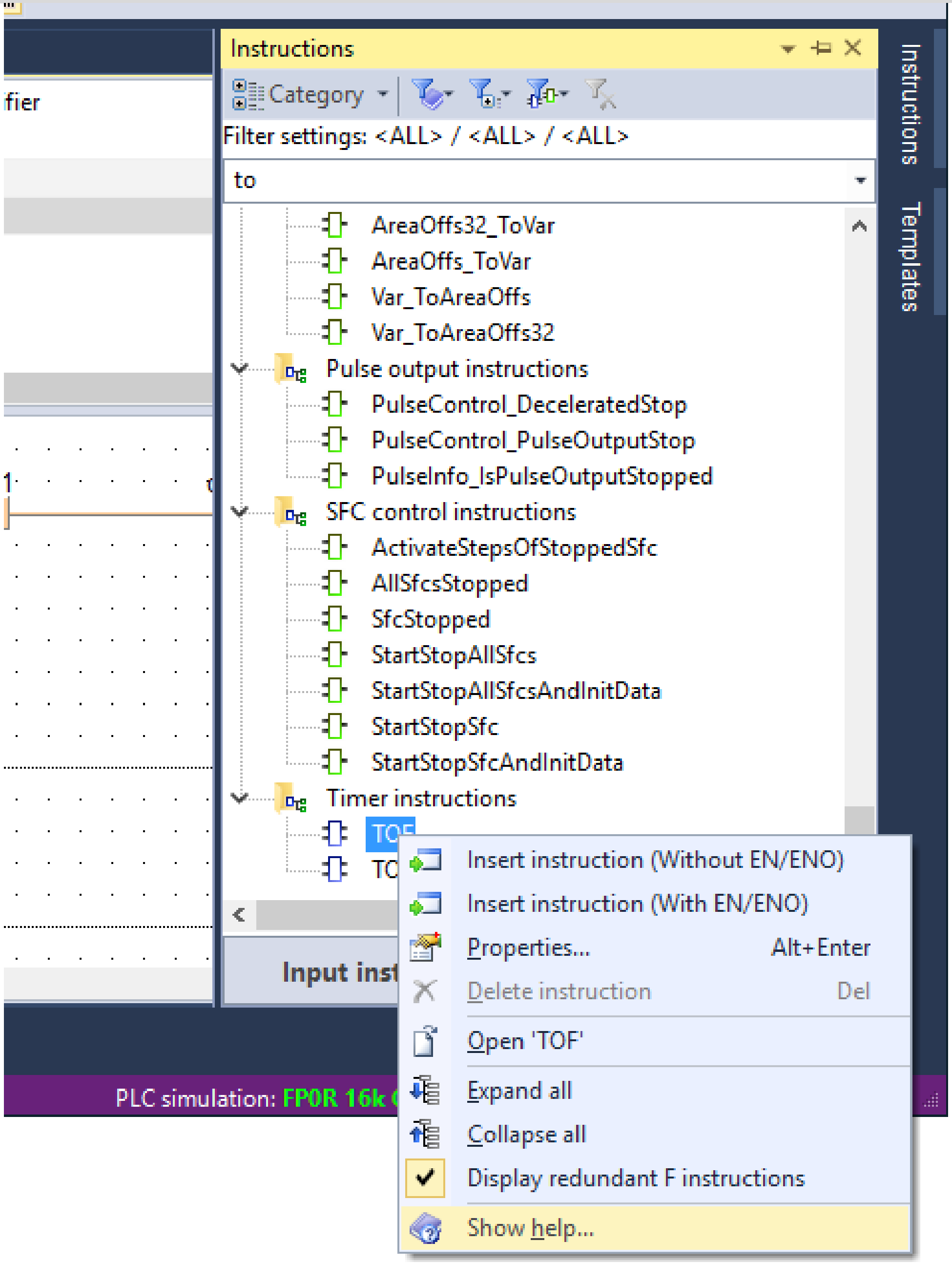
- POU – Program organization unit
- Custom user programs, functions, etc.
- Unlimited number of programs.
- Programs need to be listed in tasks to be executed



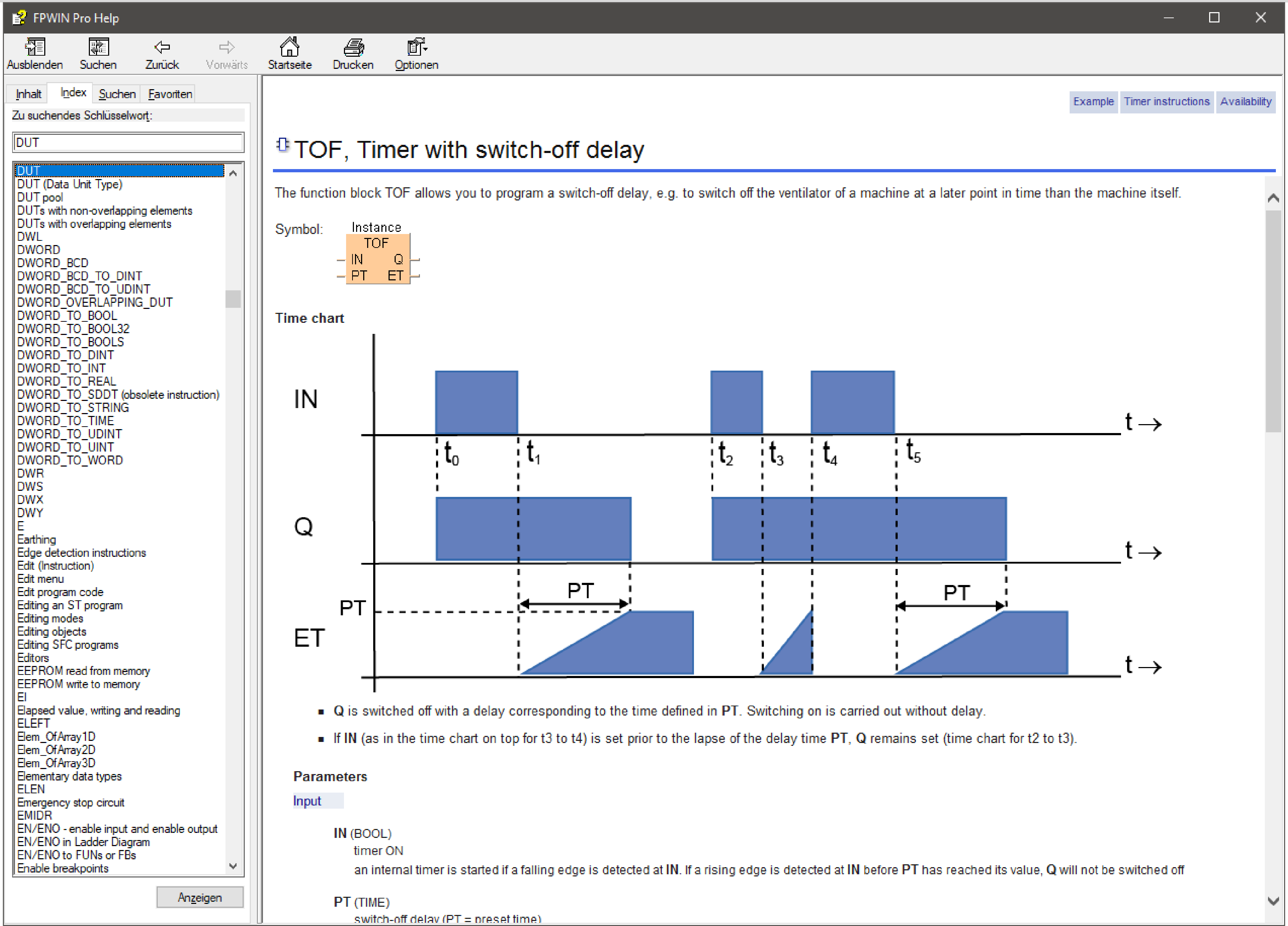
Help Function

- Help can be accessed directly from the menu
- Help on FB‘s of installed libraries is accessed by right-click on a FB from within the istructions list.

- **Good Practise:**
  First refer to help before using an unkown FB to learn on required inputs and outputs.



Help Function - Example: TOF, Timer with switch off delay



DUT
DUT (Data Unit Type)
DUT pool
DUTs with non-overlapping elements
DUTs with overlapping elements
DWL
DWORD
DWORD_BCD
DWORD_BCD_TO_DINT
DWORD_BCD_TO_UDINT
DWORD_OVERLAPPING_DUT
DWORD_TO_BOOL
DWORD_TO_BOOL32
DWORD_TO_BOOLS
DWORD_TO_DINT
DWORD_TO_INT
DWORD_TO_REAL
DWORD_TO_SDDT (obsolete instruction)
DWORD_TO_STRING
DWORD_TO_TIME
DWORD_TO_UDINT
DWORD_TO_UINT
DWORD_TO_WORD
DWR
DWS
DWX
DWY
E
Earthing
Edge detection instructions
Edit (Instruction)
Edit menu
Edit program code
Editing an ST program
Editing modes
Editing objects
Editing SFC programs
Editors
EEPROM read from memory
EEPROM write to memory
EI
Elapsed value, writing and reading
ELEFT
Elem_OfArray1D
Elem_OfArray2D
Elem_OfArray3D
Elementary data types
ELEN
Emergency stop circuit
EMIDR
EN/ENO - enable input and enable output
EN/ENO in Ladder Diagram
EN/ENO to FUNs or FBs
Enable breakpoints

TOF, Timer with switch-off delay

The function block TOF allows you to program a switch-off delay, e.g. to switch off the ventilator of a machine at a later point in time than the machine itself.

Symbol:

Time chart

- **Q** is switched off with a delay corresponding to the time defined in **PT**. Switching on is carried out without delay.
- If **IN** (as in the time chart on top for t3 to t4) is set prior to the lapse of the delay time **PT**, **Q** remains set (time chart for t2 to t3).

Parameters

Input

IN (BOOL)
timer ON
an internal timer is started if a falling edge is detected at **IN**. If a rising edge is detected at **IN** before **PT** has reached its value, **Q** will not be switched off

PT (TIME)
switch-off delay (PT = preset time)

Comparison with other Software Packages

- As FP WIN PRO is an implementation of IEC61131-3, main features, FB names, basic structure will be very similar to other IEC implementations, such as Codesys
- Code can be imported and exported
- However, FP WIN PRO is not compatible with Codesys, or similar, directly.