Codesys

Paolo Burgio
paolo.burgio@unimore.it

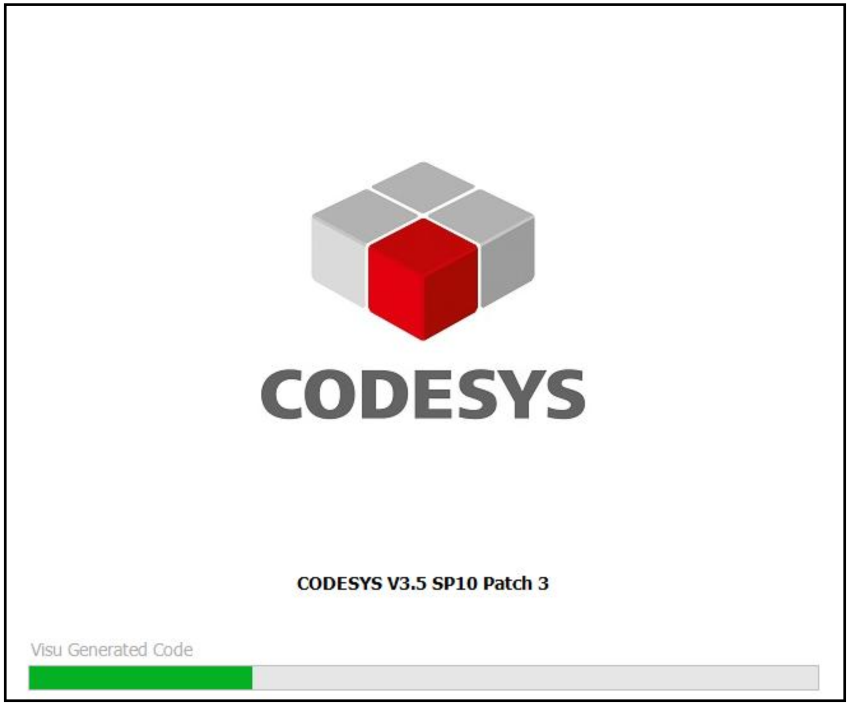
> Programming is a skill best acquired by practice and example rather than from books.

ALAN TURING

Load the main program interface

What is it?

- An IDE to create PLC programs, and **simulate them**
- In any of the five main languages
- I use V3.5 SP1 patch 3, recommended version (for compatibility with the examples I'll give you)

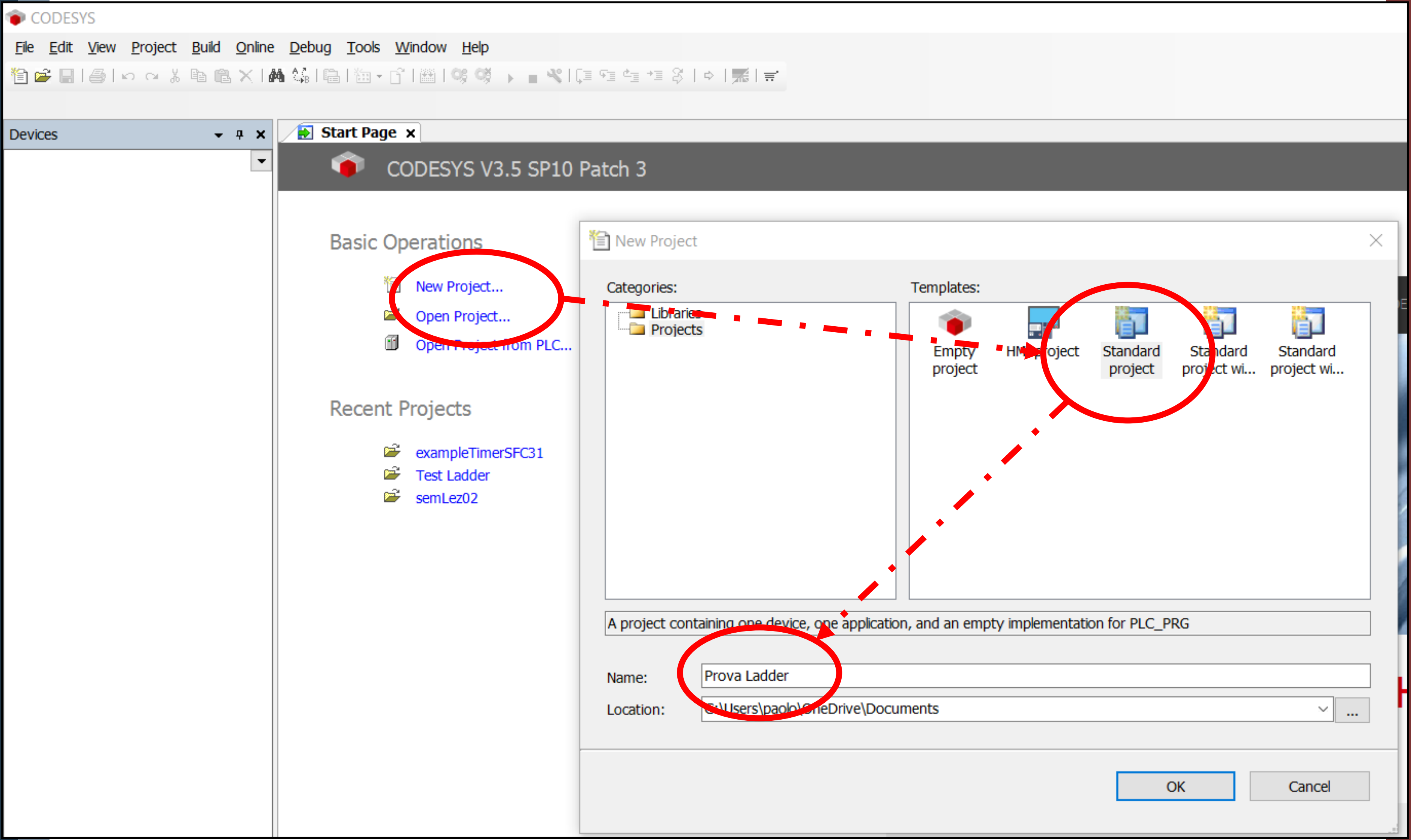


CODESYS

CODESYS V3.5 SP10 Patch 3

Visu Generated Code

Create a project

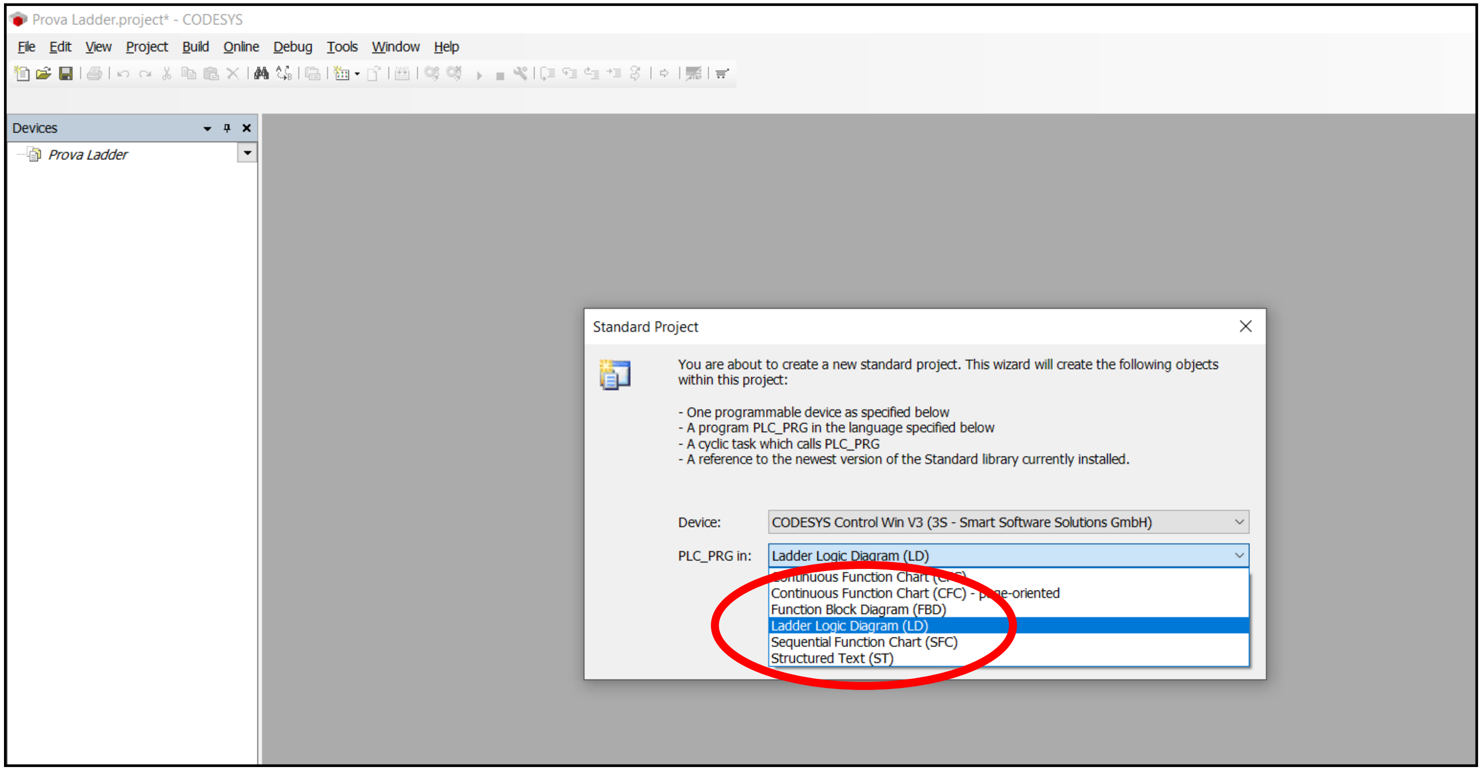


A project containing one device, one application, and an empty implementation for PLC_PRG

Name: Prova Ladder

Location: C:\Users\paolo\OneDrive\Documents

Select the language


Prova Ladder.project* - CODESYS
File Edit View Project Build Online Debug Tools Window Help
Devices
Prova Ladder

Standard Project

You are about to create a new standard project. This wizard will create the following objects within this project:

- One programmable device as specified below
- A program PLC_PRG in the language specified below
- A cyclic task which calls PLC_PRG
- A reference to the newest version of the Standard library currently installed.

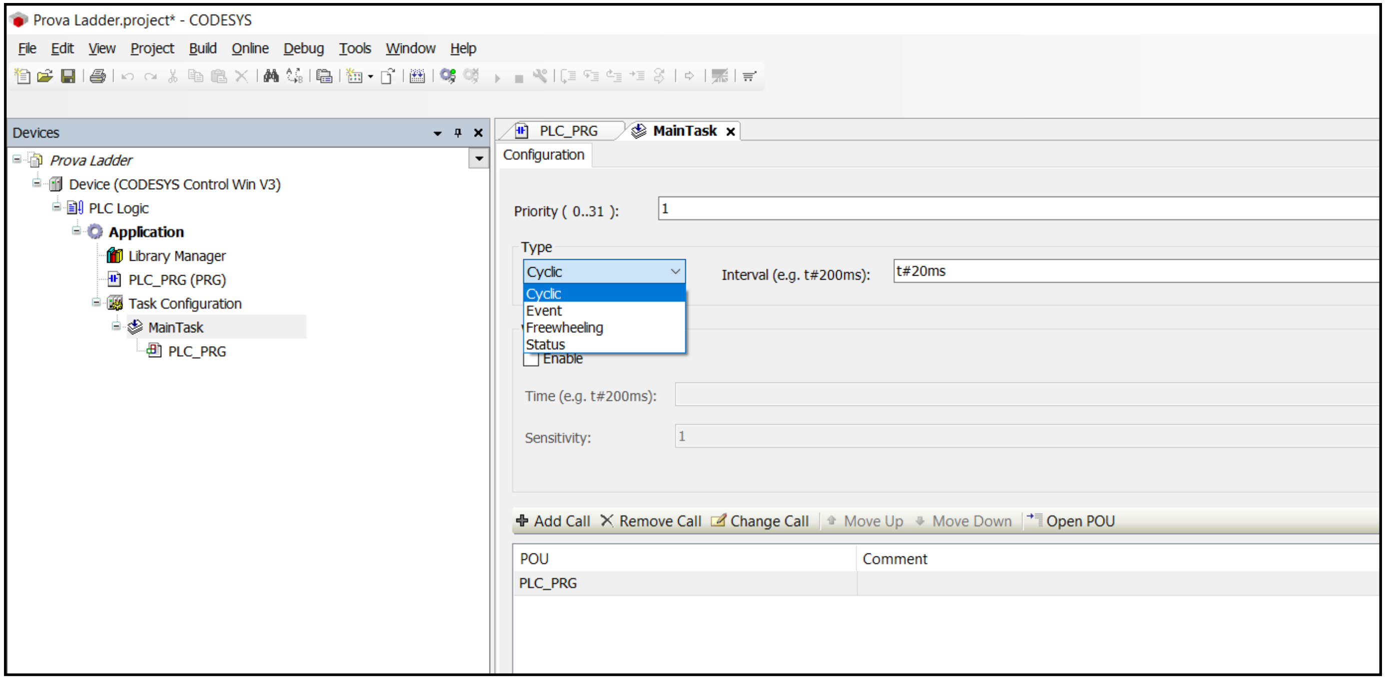
Device: CODESYS Control Win V3 (3S - Smart Software Solutions GmbH)

PLC_PRG in: Ladder Logic Diagram (LD)

Continuous Function Chart (CFC)
Continuous Function Chart (CFC) - page-oriented
Function Block Diagram (FBD)
Ladder Logic Diagram (LD)
Sequential Function Chart (SFC)
Structured Text (ST)

Project workbench

› Your application has a Main task, that (here) runs cyclically



- *Prova Ladder*
  - Device (CODESYS Control Win V3)
    - PLC Logic
      - **Application**
        - Library Manager
        - PLC_PRG (PRG)
        - Task Configuration
          - MainTask
            - PLC_PRG

Priority (0..31): 1

Type: Cyclic

Interval (e.g. t#200ms): t#20ms

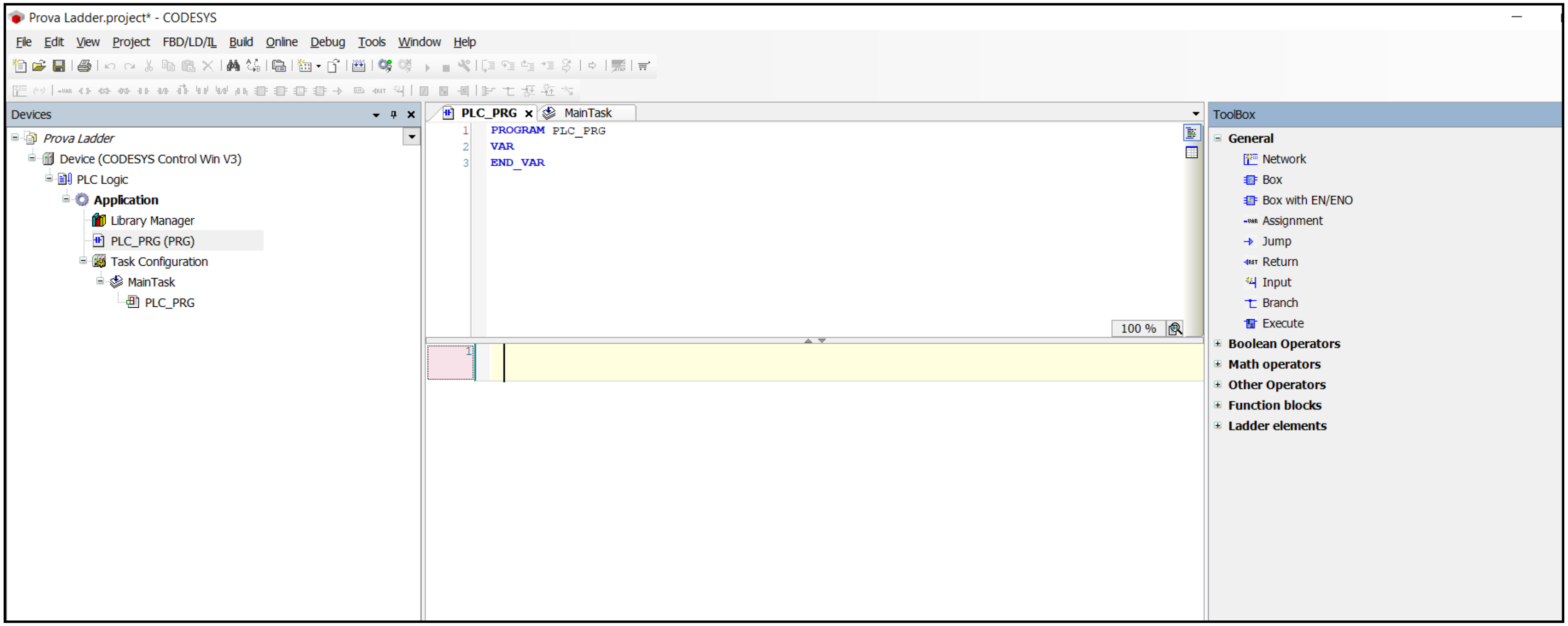
Time (e.g. t#200ms):

Sensitivity: 1

POU	Comment
PLC_PRG	

Project workbench - Ladder

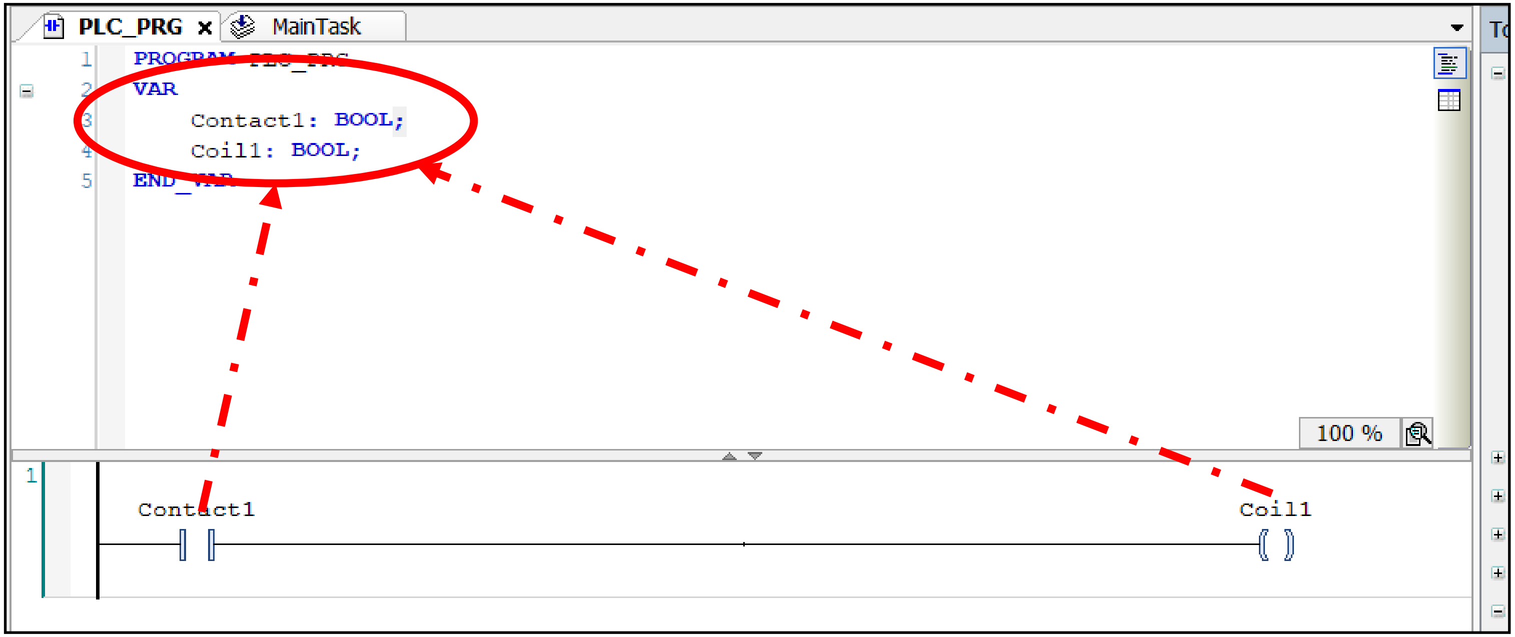
- You can create Ladder diagrams using drag/drop from the toolbox



```
PROGRAM PLC_PRG
VAR
END_VAR
```

Adding a contact + coil

- Two global variables are automatically created in the variable definition window always in ST lang), both of `bool` type, as specified by us
- Here, we want a switch that turns on a lamp, hence we need a NO contact and a coil
- PS here you don't see the right power rail as it's implicit

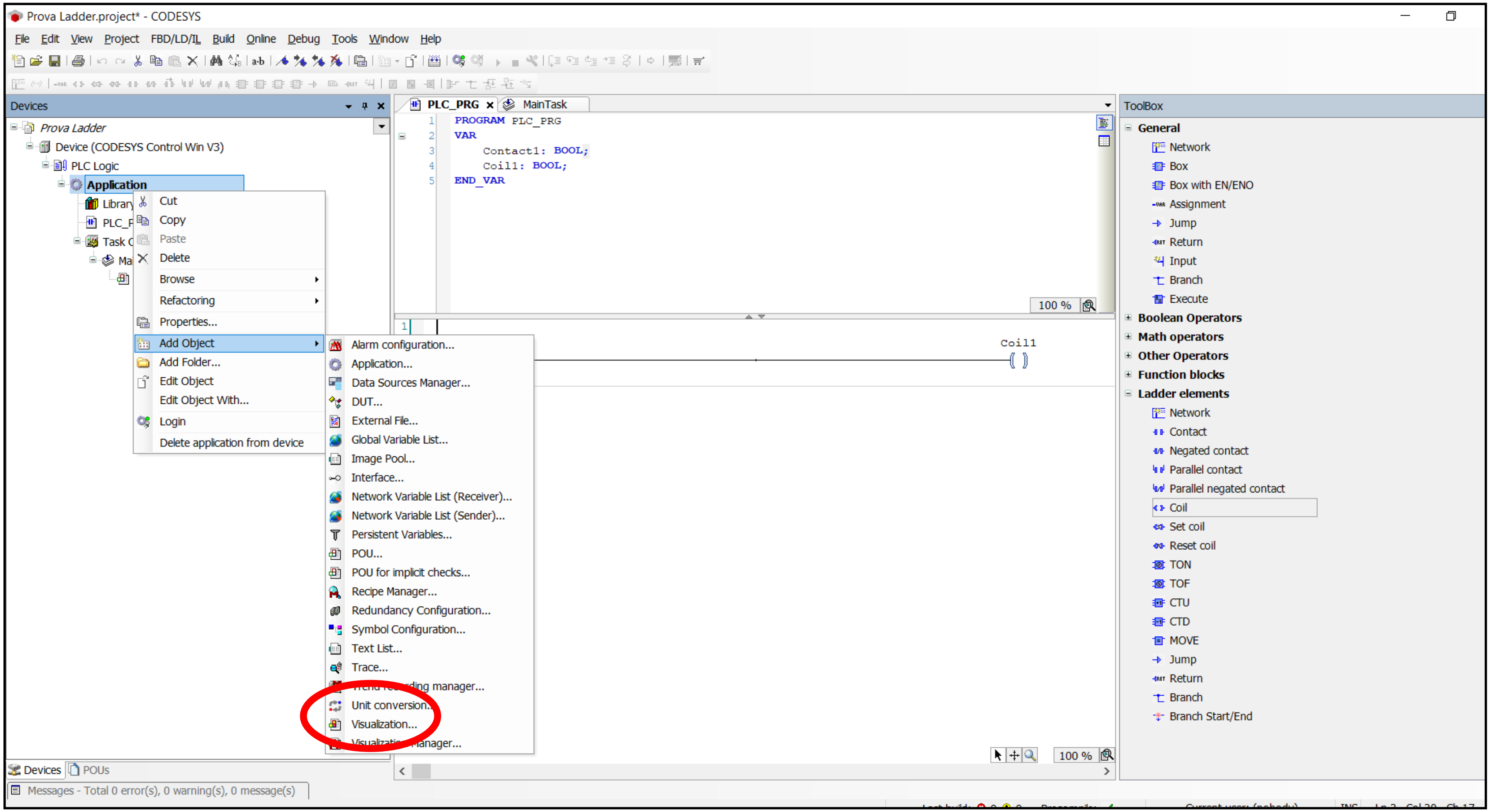


```
PROGRAM PLC_PRG
VAR
    Contact1: BOOL;
    Coil1: BOOL;
END_VAR
```

View the simulated system

Add a Visualization object

- Application -> Add Object -> Visualization

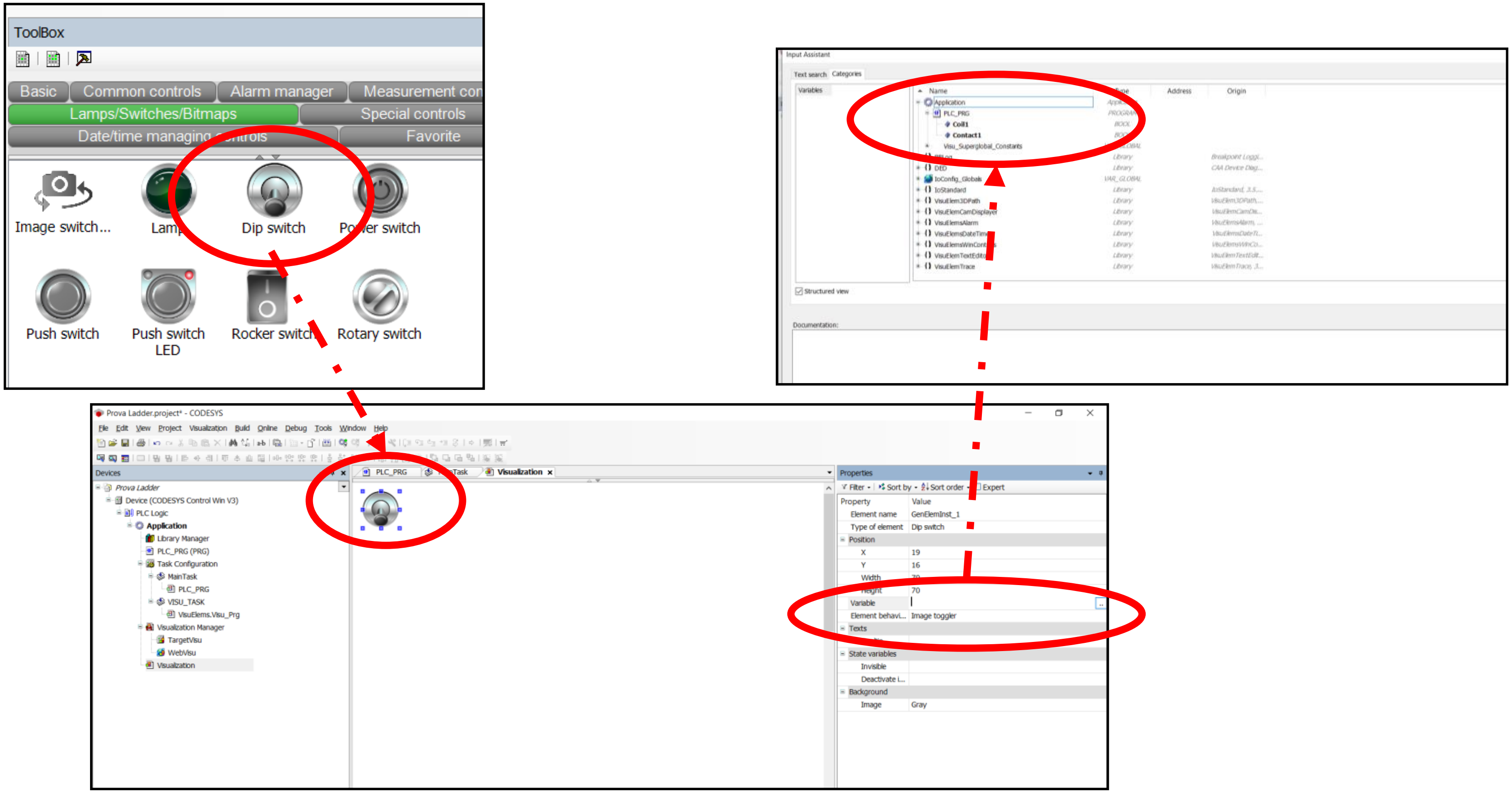


```
PROGRAM PLC_PRG
VAR
    Contact1: BOOL;
    Coil1: BOOL;
END_VAR
```

Messages - Total 0 error(s), 0 warning(s), 0 message(s)

Add elements, and link to variables

- Here, we added a dip switch from the toolbox, and we select the `Contact1` var from the Properties window
- Now, add a lamp and bind it to `Coil1`

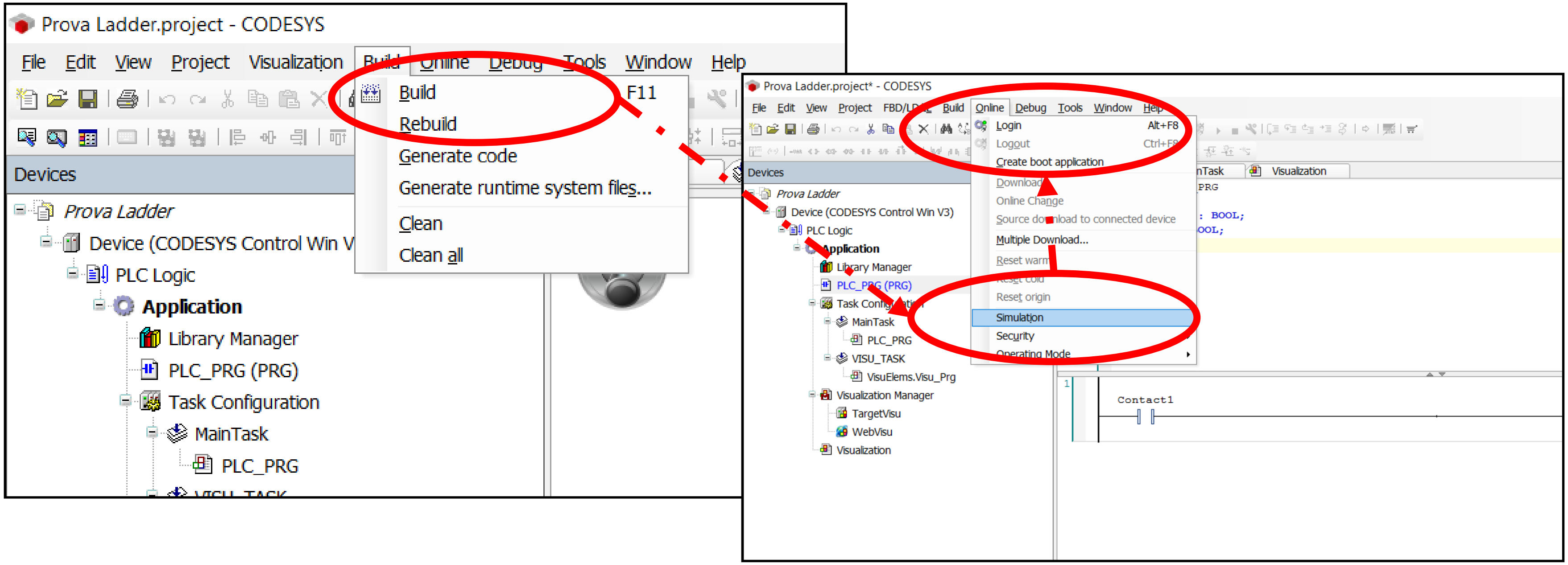


Name	Type	Address	Origin
Application	Application		
PLC_PRG	PROGRAM		
Coil1	BOOL		
Contact1	BOOL		
Visu_Superglobal_Constants	VAR_GLOBAL		
BPLog	Library		Breakpoint Loggi...
DED	Library		CAA Device Diag...
IoConfig_Globals	VAR_GLOBAL		
IoStandard	Library		IoStandard, 3.5,...
VisuElem3DPath	Library		VisuElem3DPath,...
VisuElemCamDisplayer	Library		VisuElemCamDis...
VisuElemsAlarm	Library		VisuElemsAlarm, ...
VisuElemsDateTime	Library		VisuElemsDateTi...
VisuElemsWinControls	Library		VisuElemsWinCo...
VisuElemTextEditor	Library		VisuElemTextEdit...
VisuElemTrace	Library		VisuElemTrace, 3...

Property	Value
Element name	GenElemInst_1
Type of element	Dip switch
Position	
X	19
Y	16
Width	70
Height	70
Variable	
Element behavi...	Image toggler
Texts	
State variables	
Invisible	
Deactivate i...	
Background	
Image	Gray

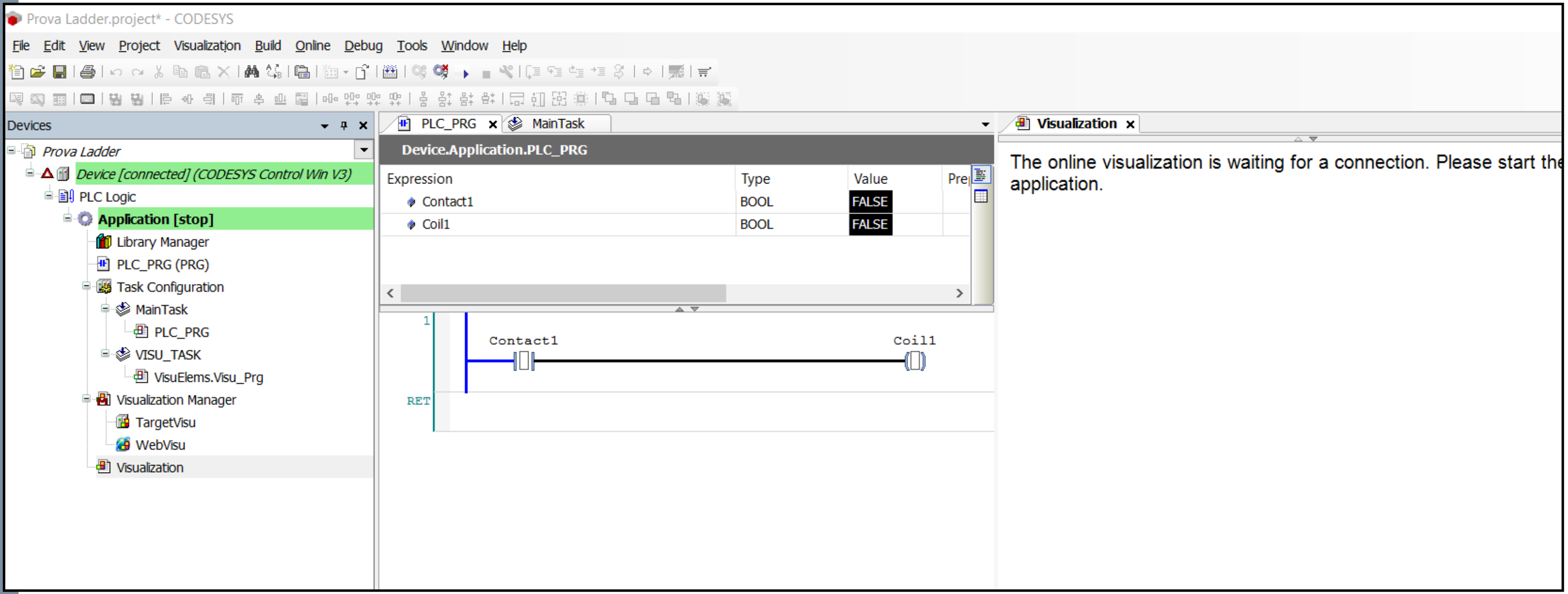
Compile and set up simulator

- Build the system, from the menu or with F11
- Login from the Online menu to download the required run libs
  - Before..make sure you ticked “Simulation”!
- Now, we're ready to go



Run workbench

- After a while, simulator/simulation is set up
- Click on Debug -> Start to go
- Nothing happens

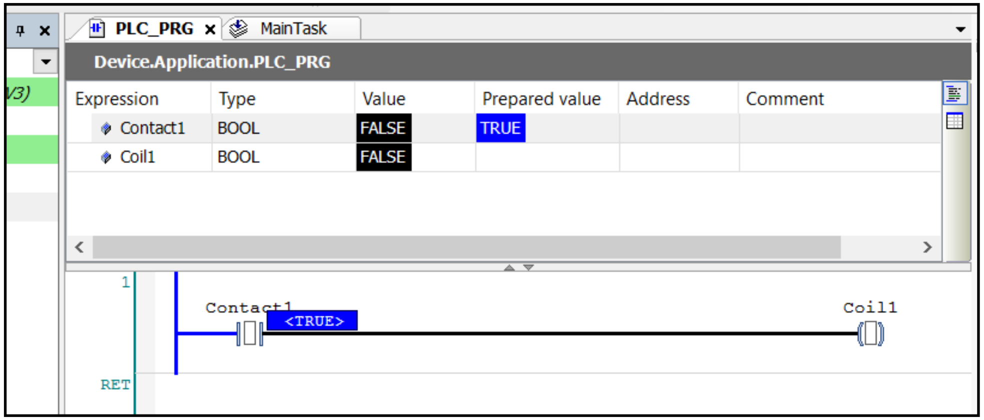


Expression	Type	Value	Pre
Contact1	BOOL	FALSE	
Coil1	BOOL	FALSE	

The online visualization is waiting for a connection. Please start the application.

Modify values

- Via the “watch expression” window, use the “Prepared value”
- Then, apply the value with the Debug -> Write value menu item (or CTRL+F7)



Expression	Type	Value	Prepared value	Address	Comment
Contact1	BOOL	FALSE	TRUE		
Coil1	BOOL	FALSE			

- In this case, in our example, we can also manually acting on the switch

Remember to log out after you’re done! ☺

Sequential contacts vs. parallel contacts

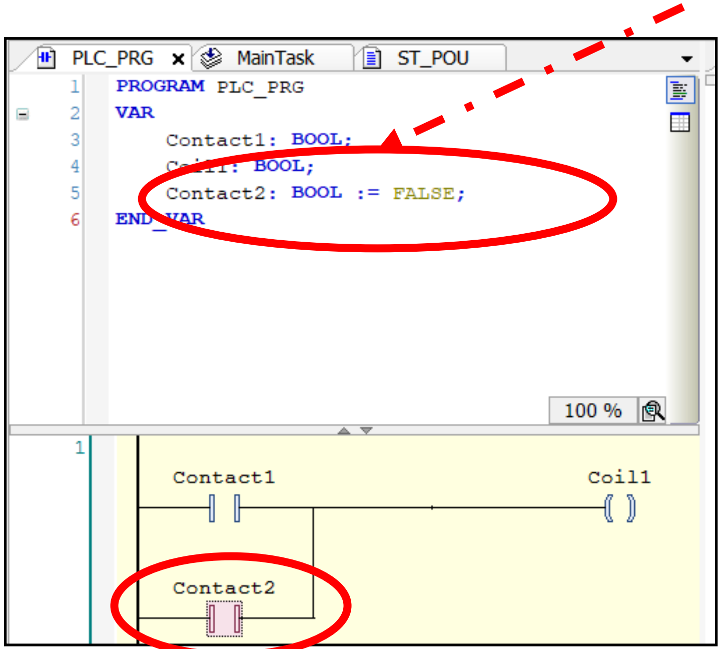
Logical “AND”

- ..easy, simply drag&drop

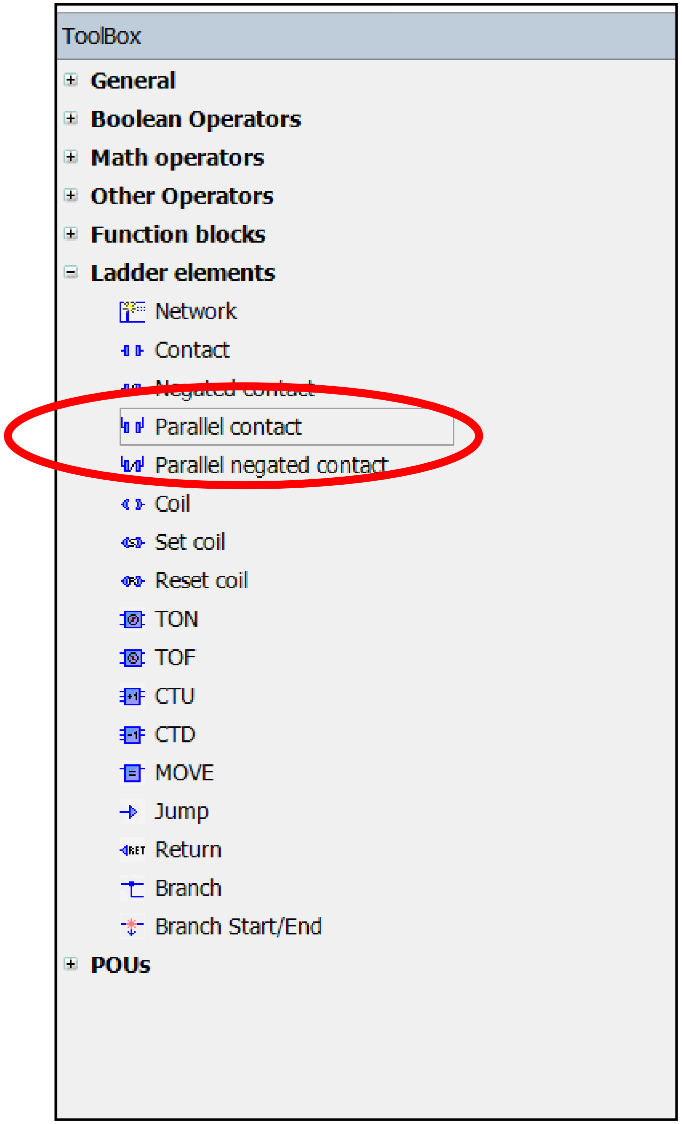
Logical “OR”

- “Parallel contact” components from toolbox
- IDE helps us to insert it…

PS good programmers remember to initialize vars ;)



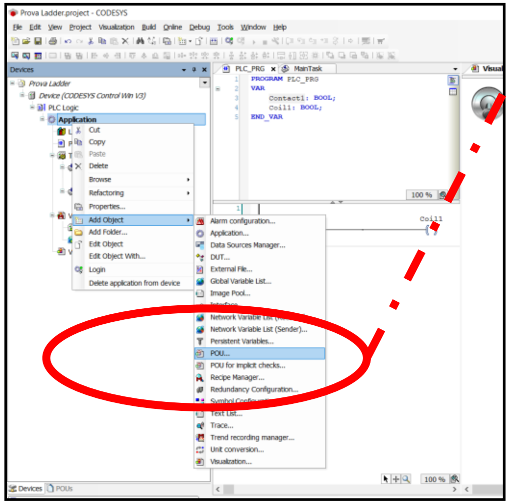
```
PROGRAM PLC_PRG
VAR
    Contact1: BOOL;
    Coil1: BOOL;
    Contact2: BOOL := FALSE;
END_VAR
```



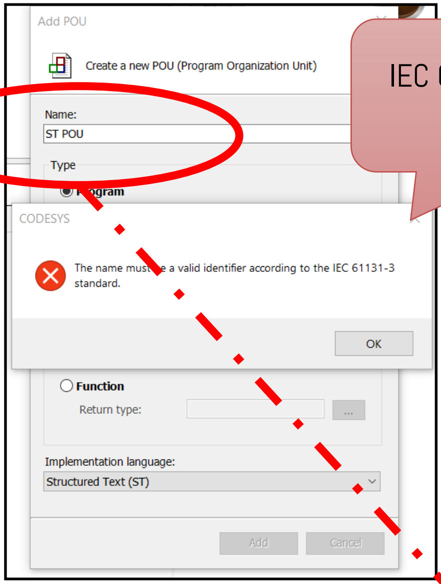
Structured Text

Add new ST POU

- Program Organization Unit let you add logics in the same application, using different languages
- We now add a **Program POU**



```
PROGRAM PLC_PRG
VAR
    Contact1: BOOL;
    Coil1: BOOL;
END_VAR
```



Name: ST POU

● Program

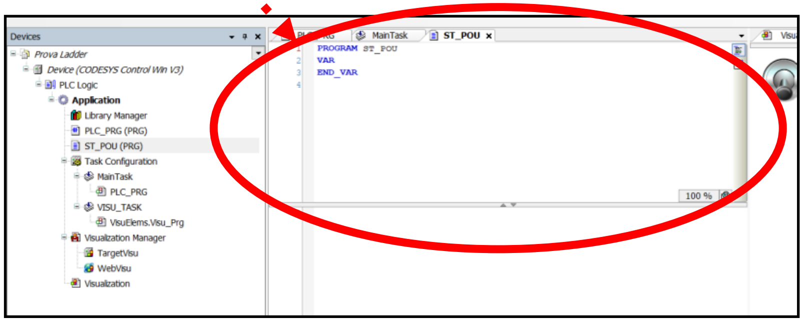
The name must be a valid identifier according to the IEC 61131-3 standard.

○ Function

Return type:

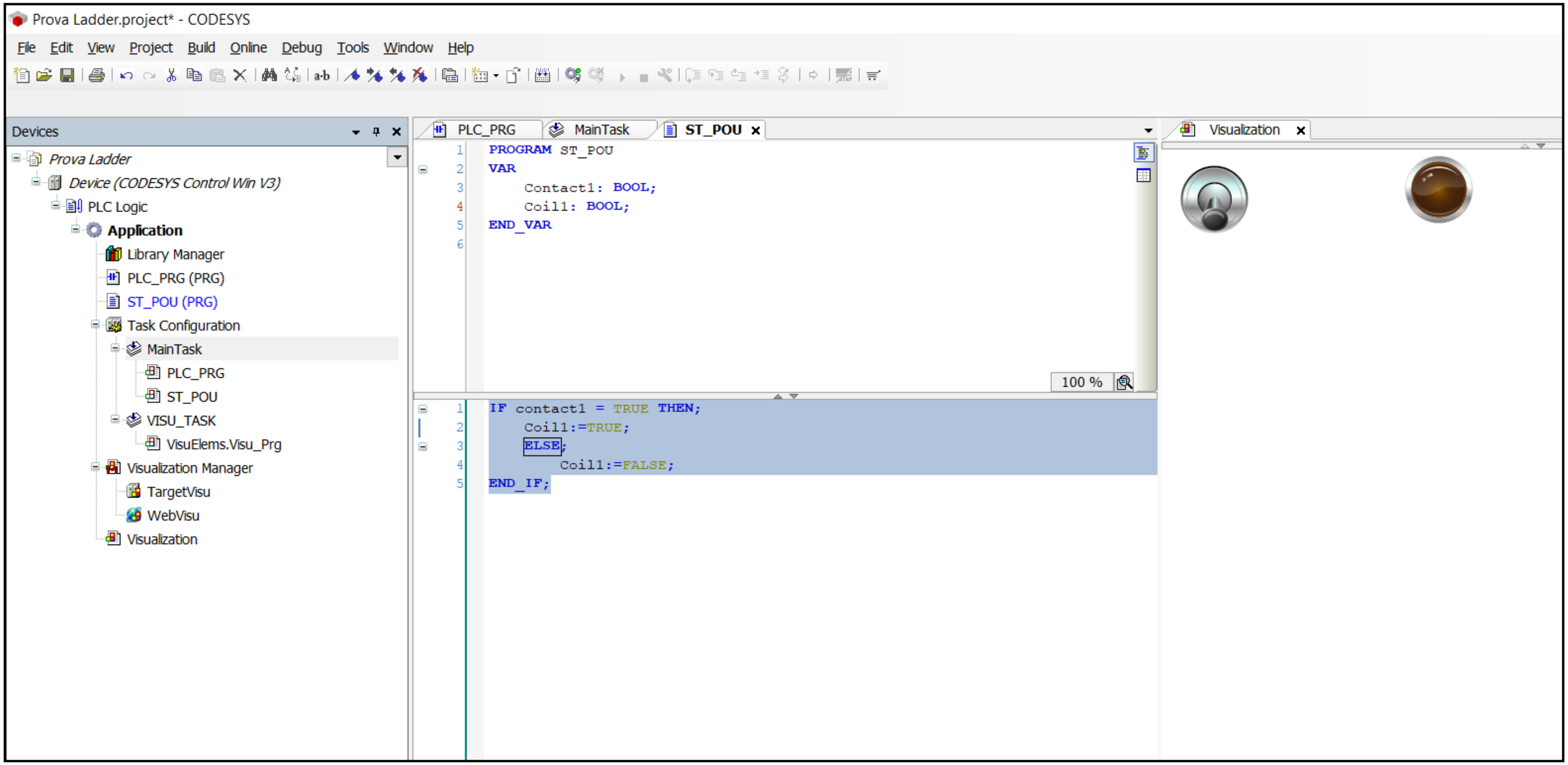
Implementation language: Structured Text (ST)

IEC 61131 does not allow spaces in names



```
PROGRAM ST_POU
VAR
END_VAR
```

Write the ST code

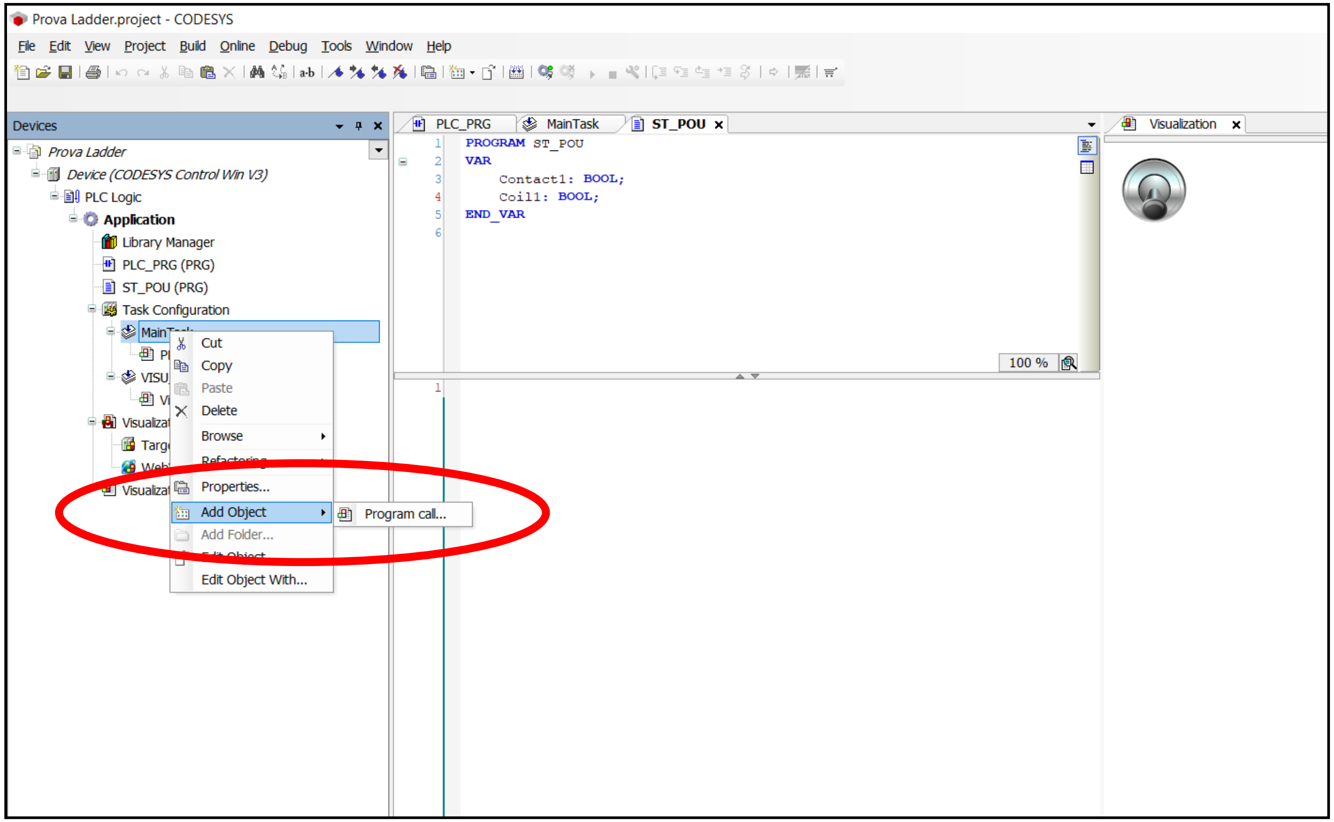


```
PROGRAM ST_POU
VAR
    Contact1: BOOL;
    Coil1: BOOL;
END_VAR
```

```
IF contact1 = TRUE THEN;
    Coil1:=TRUE;
    ELSE;
        Coil1:=FALSE;
END_IF;
```

Are we done? Not yet...

› We created a POU Program, but we haven't called it yet from within the `MainTask`...



```
PROGRAM ST_POU
VAR
    Contact1: BOOL;
    Coil1: BOOL;
END_VAR
```

Run and set values

- If you set `Contact1` to `TRUE`, then `Coil1` goes to `TRUE`
- ..but the simulated Light & Switch don't turn on!

Why?

- Because they are **not** attached to **those** `Contact1` and `Coil1` vars you think...
- Look out when you write names...

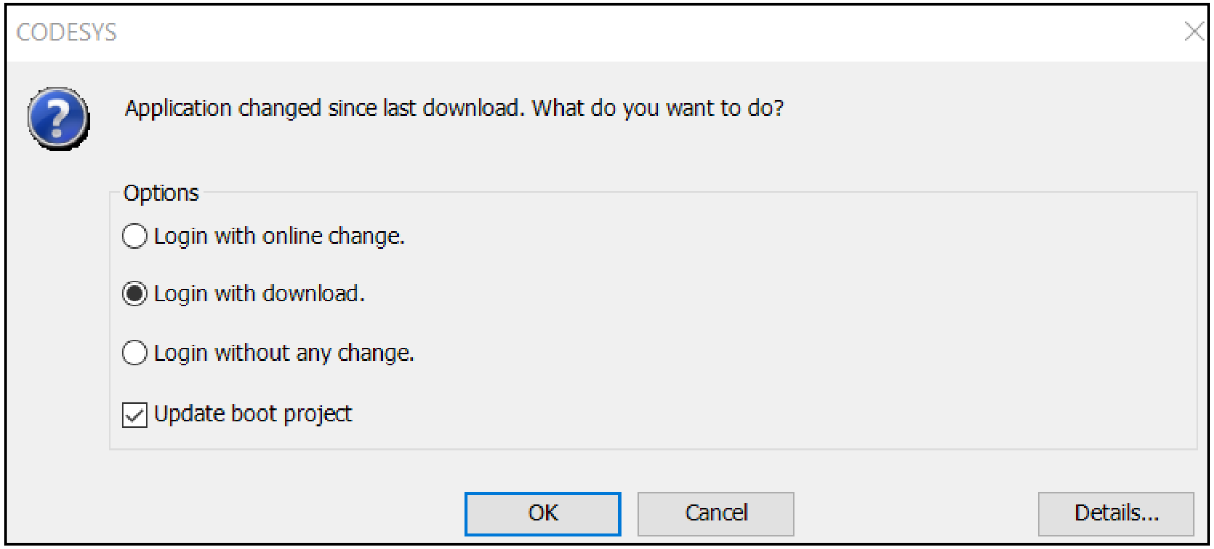
Should we attach those vars to the two simulated objects?

- (recommendation) Only if requested by the application specs
- In this case, I use them for debugging/teaching purposes, so my specs say "no" ☺

Compile & Login again

We added a ST block, so the simulation engine might require some components

- Codesys will prompt us



CODESYS

Application changed since last download. What do you want to do?

Options

○ Login with online change.

● Login with download.

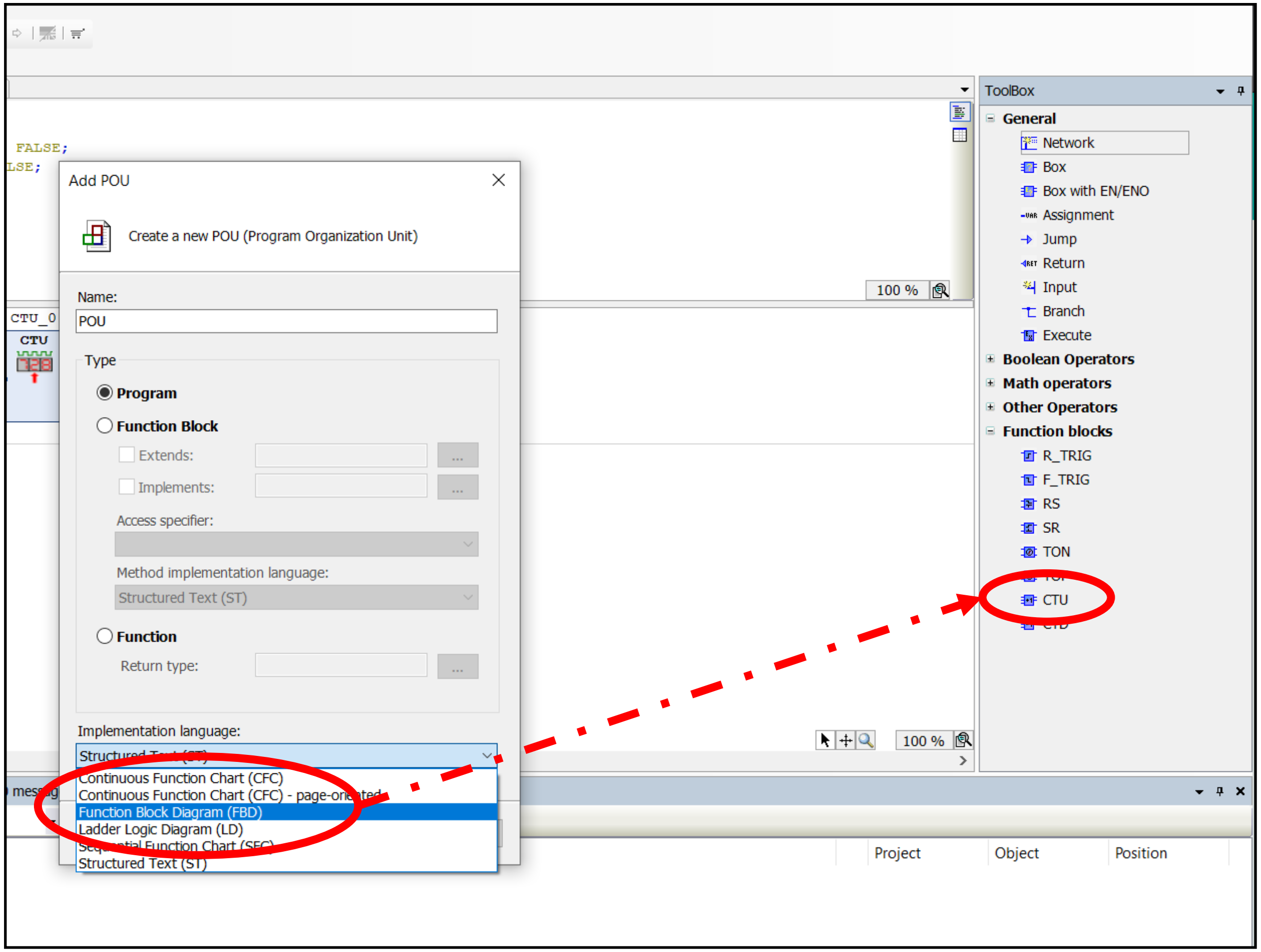
○ Login without any change.

☑ Update boot project

Programming with Function Blocks

Counters - CTU

- Create a program, or add a POU of type "FBD"
- Then, drag a Counter (CTU) in the workbench



Add POU

Create a new POU (Program Organization Unit)

Name: POU

Type

● Program

○ Function Block

□ Extends:

□ Implements:

Access specifier:

Method implementation language: Structured Text (ST)

○ Function

Return type:

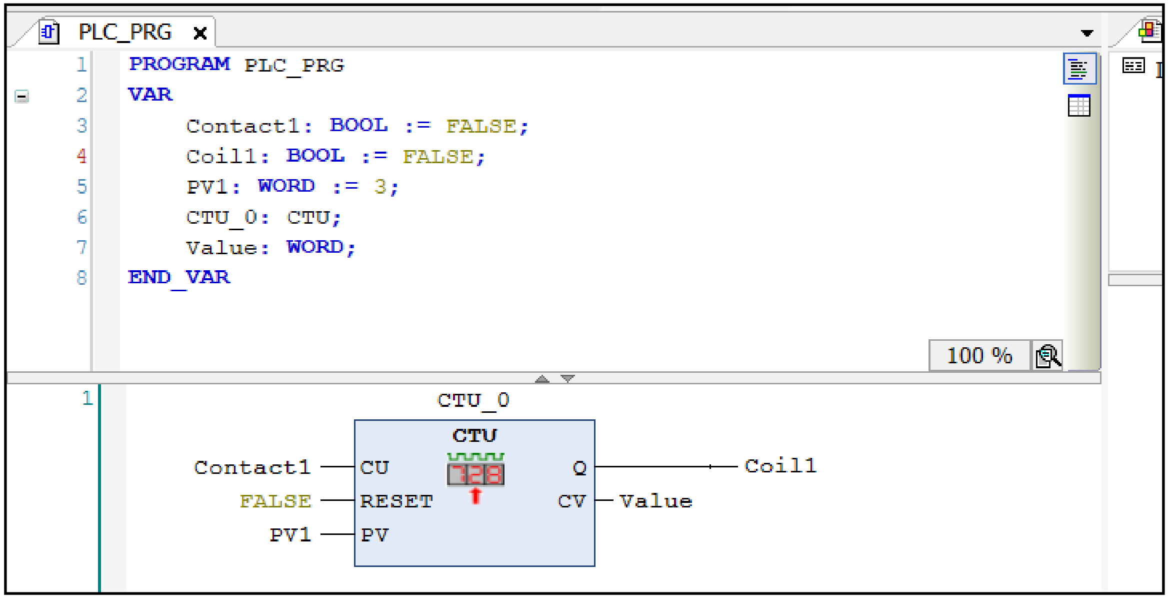
Implementation language: Structured Text (ST)

- Continuous Function Chart (CFC)
- Continuous Function Chart (CFC) - page-oriented
- Function Block Diagram (FBD)
- Ladder Logic Diagram (LD)
- Sequential Function Chart (SFC)
- Structured Text (ST)

Bind the CTU

Connect CTU in&outs to vars

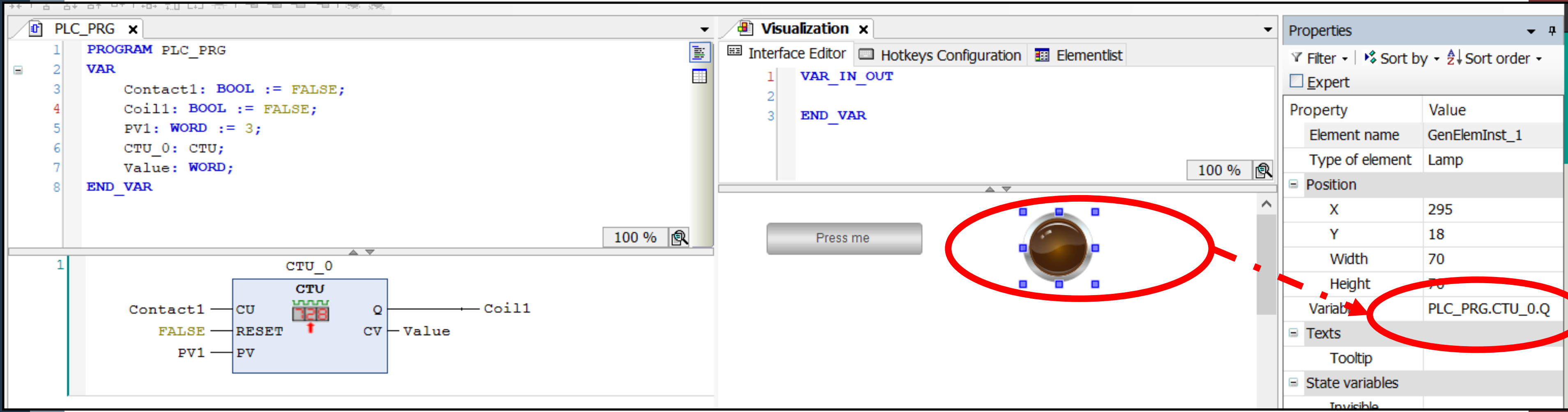
- CU to a contact (press&hold button)
- Here, RESET is false (just for this example)..might use a button/contact?
- PV1 is a WORD
- Q to a coil (we want to turn on a lamp)
- CV to a WORD variable, to monitor the status



```
PROGRAM PLC_PRG
VAR
    Contact1: BOOL := FALSE;
    Coil1: BOOL := FALSE;
    PV1: WORD := 3;
    CTU_0: CTU;
    Value: WORD;
END_VAR
```

Add visualization

- Attach a “simple” press&hold button to `Contact1`
- Add a lamp and attach to `Coil1`
- ..or, can also attach it directly to CTU out `PLC_PRG.CTU_0.Q`



PLC_PRG

```
PROGRAM PLC_PRG
VAR
    Contact1: BOOL := FALSE;
    Coil1: BOOL := FALSE;
    PV1: WORD := 3;
    CTU_0: CTU;
    Value: WORD;
END_VAR
```

Visualization

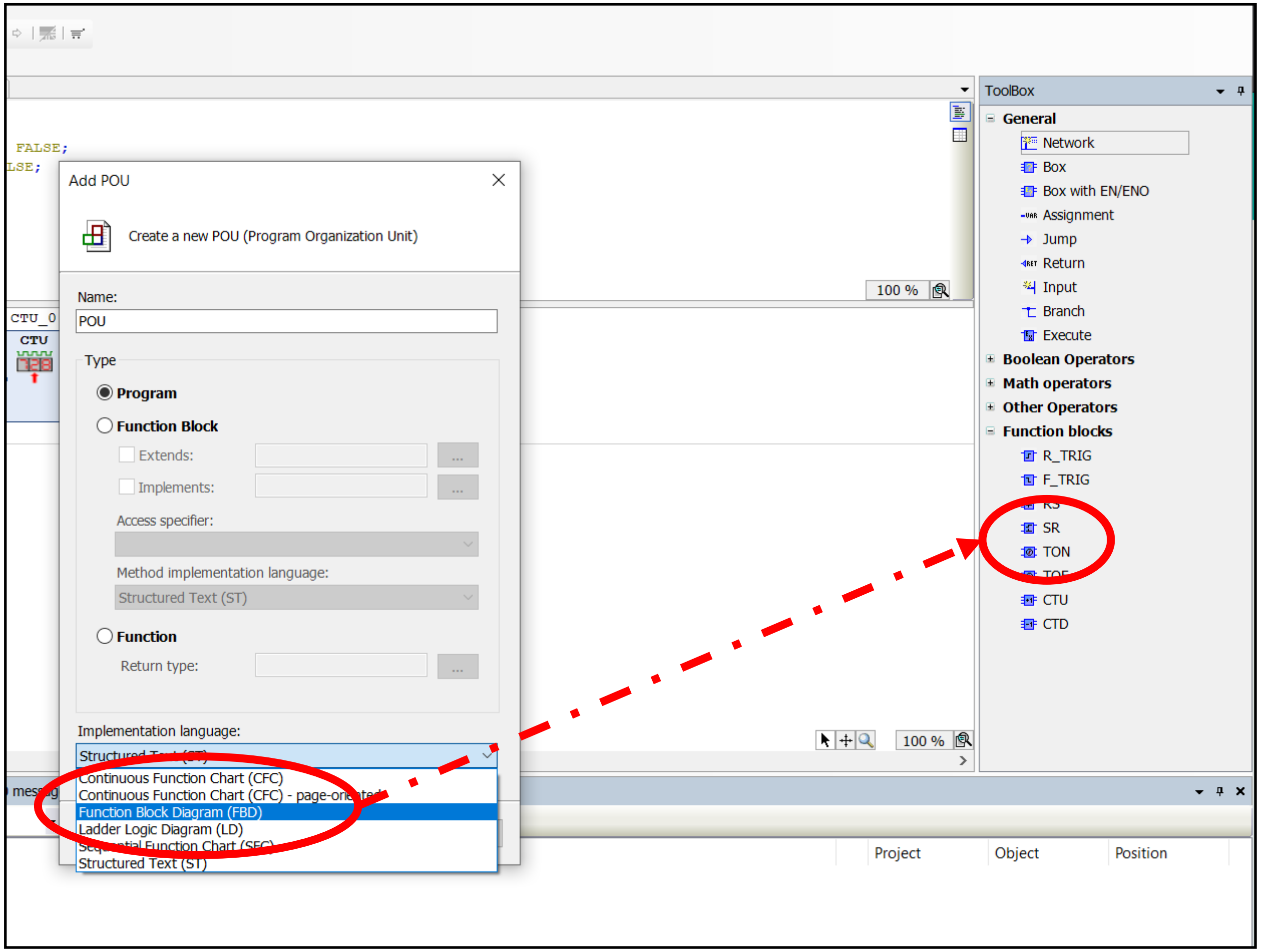
```
VAR_IN_OUT

END_VAR
```

Properties

Property	Value
Element name	GenElemInst_1
Type of element	Lamp
Position	
X	295
Y	18
Width	70
Height	70
Variab	PLC_PRG.CTU_0.Q
Texts	
Tooltip	
State variables	
Invisible	

Timers – TON

- Create a program, or add a POU of type “FBD”
- Then, drag a Timer On (TON) in the workbench



FALSE;
LSE;

CTU_0
CTU

Add POU

Create a new POU (Program Organization Unit)

Name: POU

Type

● Program

○ Function Block

□ Extends:

□ Implements:

Access specifier:

Method implementation language: Structured Text (ST)

○ Function

Return type:

Implementation language: Structured Text (ST)

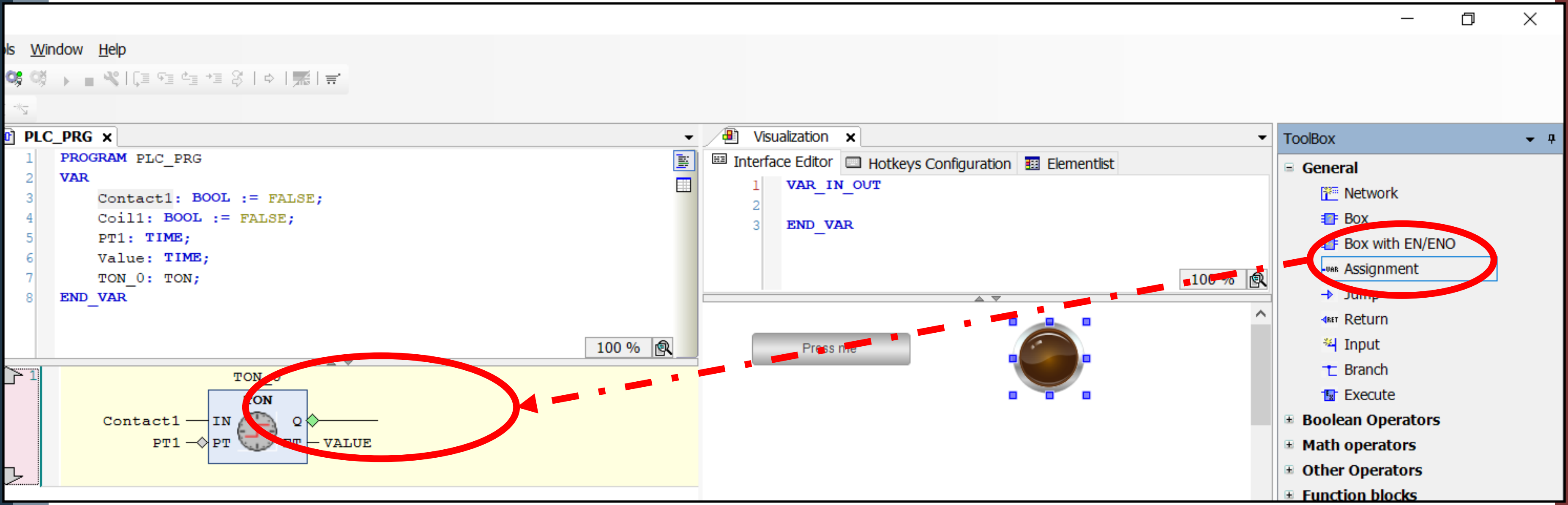
- Continuous Function Chart (CFC)
- Continuous Function Chart (CFC) - page-oriented
- Function Block Diagram (FBD)
- Ladder Logic Diagram (LD)
- Sequential Function Chart (SFC)
- Structured Text (ST)

ToolBox

- General
  - Network
  - Box
  - Box with EN/ENO
  - Assignment
  - Jump
  - Return
  - Input
  - Branch
  - Execute
- Boolean Operators
- Math operators
- Other Operators
- Function blocks
  - R_TRIG
  - F_TRIG
  - RS
  - SR
  - TON
  - TOF
  - CTU
  - CTD

Bind the TON

- IN to a contact (press&hold/standard button)
- Remember, IN starts timer @its rising edge, and resets @its falling edge
- PT1 is a TIME
- Q to a coil (we want to turn on a lamp) using the **assignment** operator
- ET to a TIME variable, to monitor the status



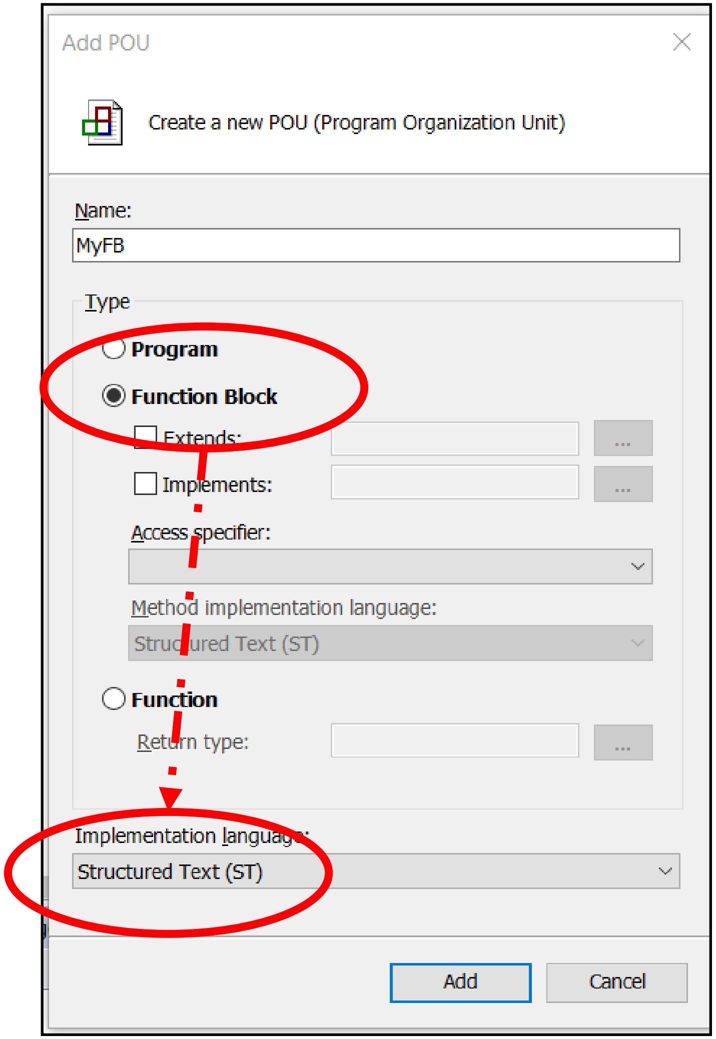
```
PROGRAM PLC_PRG
VAR
    Contact1: BOOL := FALSE;
    Coil1: BOOL := FALSE;
    PT1: TIME;
    Value: TIME;
    TON_0: TON;
END_VAR
```

```
VAR_IN_OUT

END_VAR
```

Defining Function Blocks

Create a new POU of type Function Blocks

- (of course, in ST)



Add POU

Create a new POU (Program Organization Unit)

Name: MyFB

Type

○ Program

● Function Block

□ Extends:

□ Implements:

Access specifier:

Method implementation language: Structured Text (ST)

○ Function

Return type:

Implementation language: Structured Text (ST)

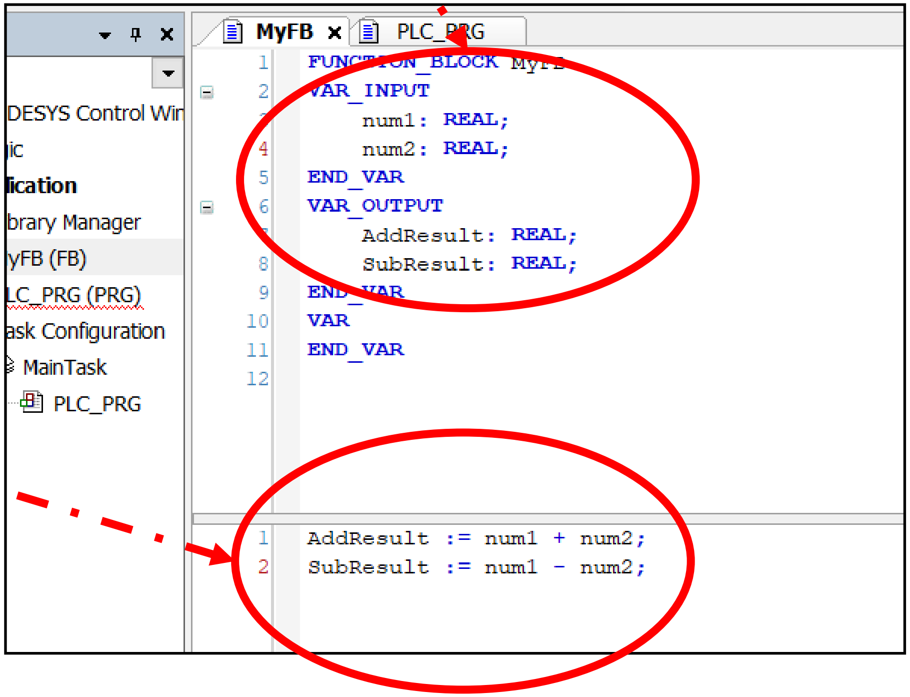
Implement the Function Block

Add in/out vars

- `num1`, `num2` as ins, `SumResult` and `SubResult` as outs, all `REAL` numbers

Add FB logics

- In ST workbench



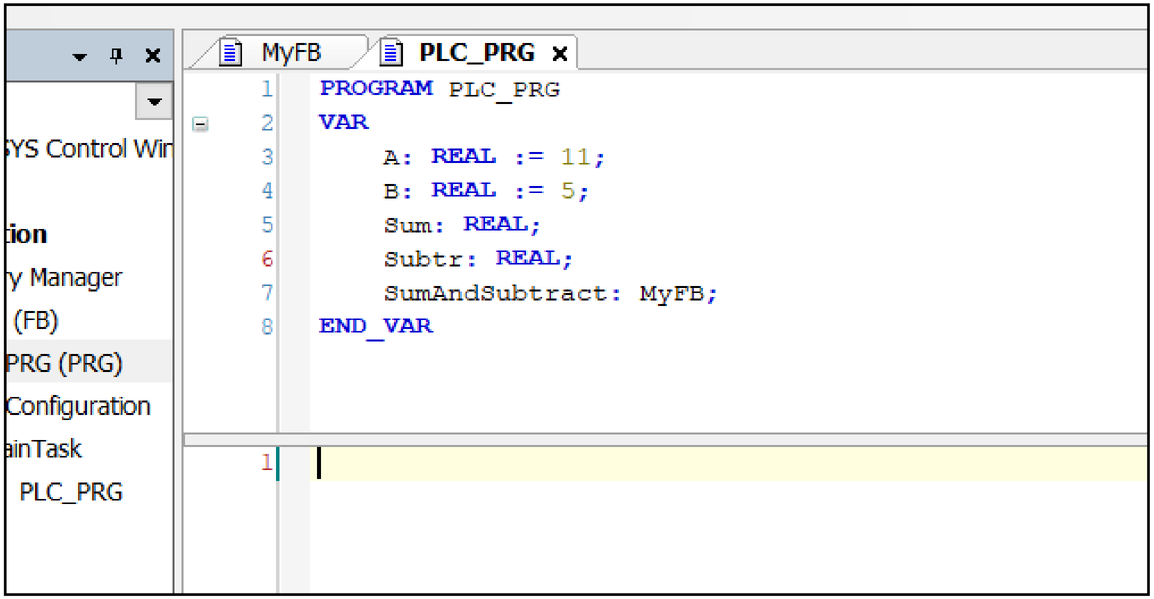
```
FUNCTION_BLOCK MyFB
VAR_INPUT
    num1: REAL;
    num2: REAL;
END_VAR
VAR_OUTPUT
    AddResult: REAL;
    SubResult: REAL;
END_VAR
VAR
END_VAR
```

```
AddResult := num1 + num2;
SubResult := num1 - num2;
```

Use it in the Application

In the main POU, create vars

- A and B are assigned, respectively, 11 and 5
- Also, instantiate the FB



```
PROGRAM PLC_PRG
VAR
    A: REAL := 11;
    B: REAL := 5;
    Sum: REAL;
    Subtr: REAL;
    SumAndSubtract: MyFB;
END_VAR
```

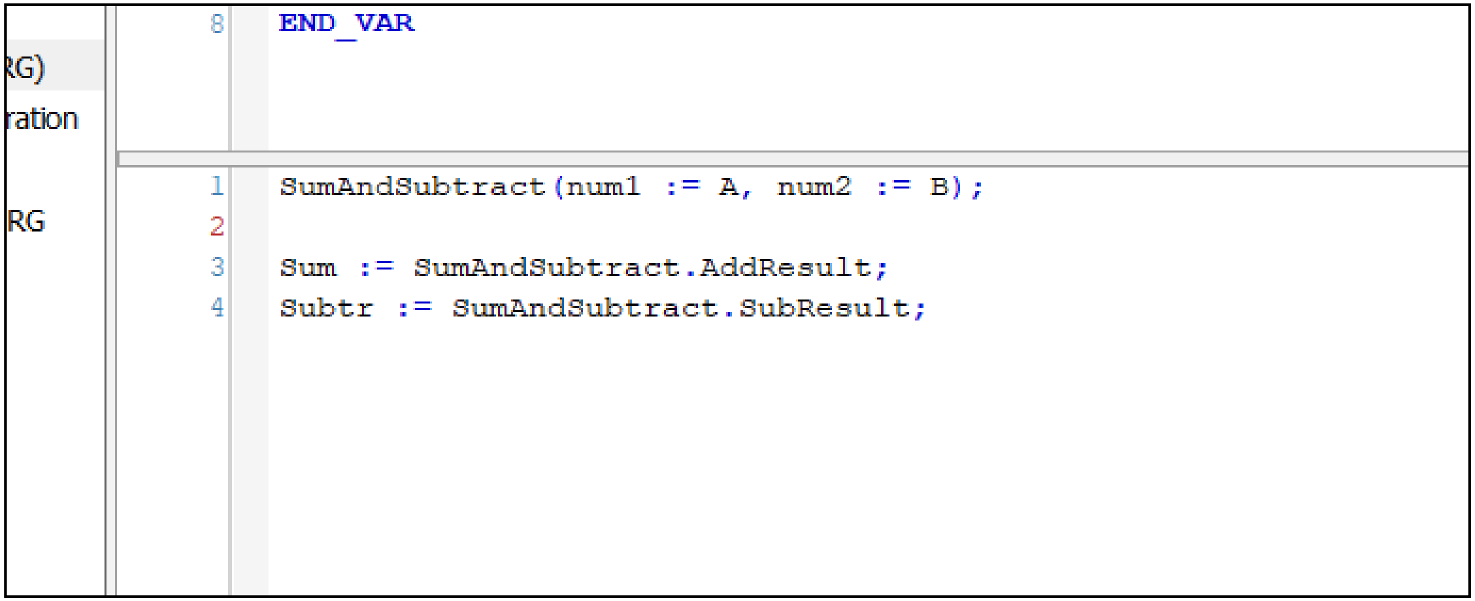
Call the FB from application

Need to explicitly bind FB input vars to main POU the vars

- E.g., A to num1
- Can use dot notation to fetch output values, after calling

Slightly different than in “traditional” programming languages

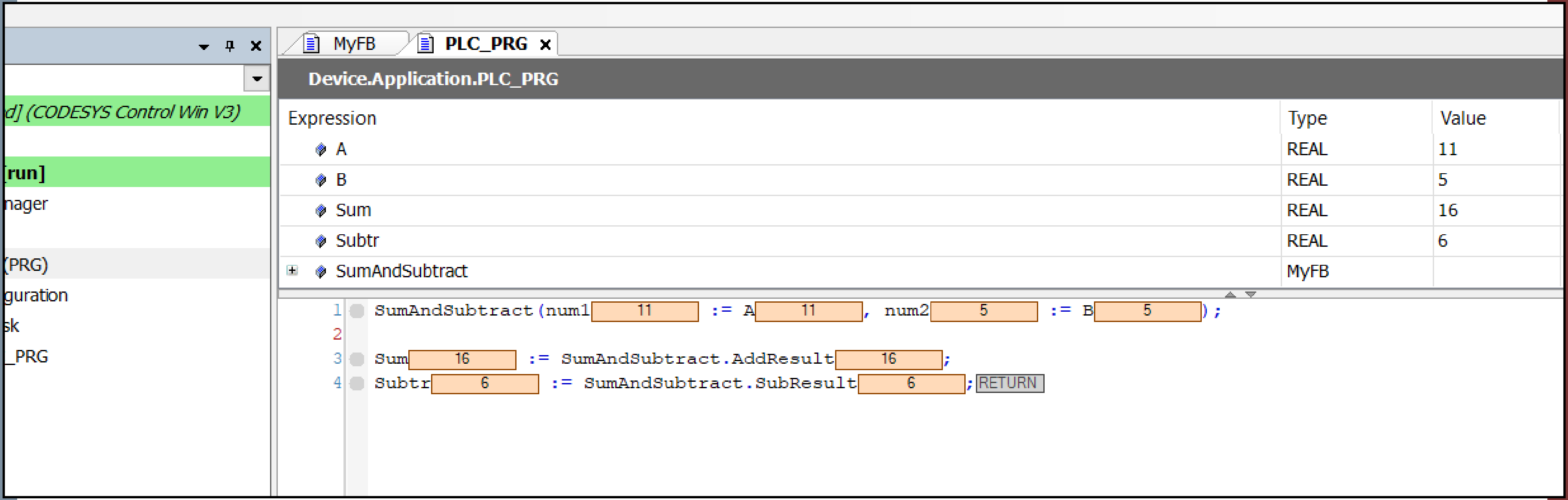
- Why?



```
END_VAR
```

```
SumAndSubtract(num1 := A, num2 := B);

Sum := SumAndSubtract.AddResult;
Subtr := SumAndSubtract.SubResult;
```

Simulate..and enjoy! ☺



Device.Application.PLC_PRG

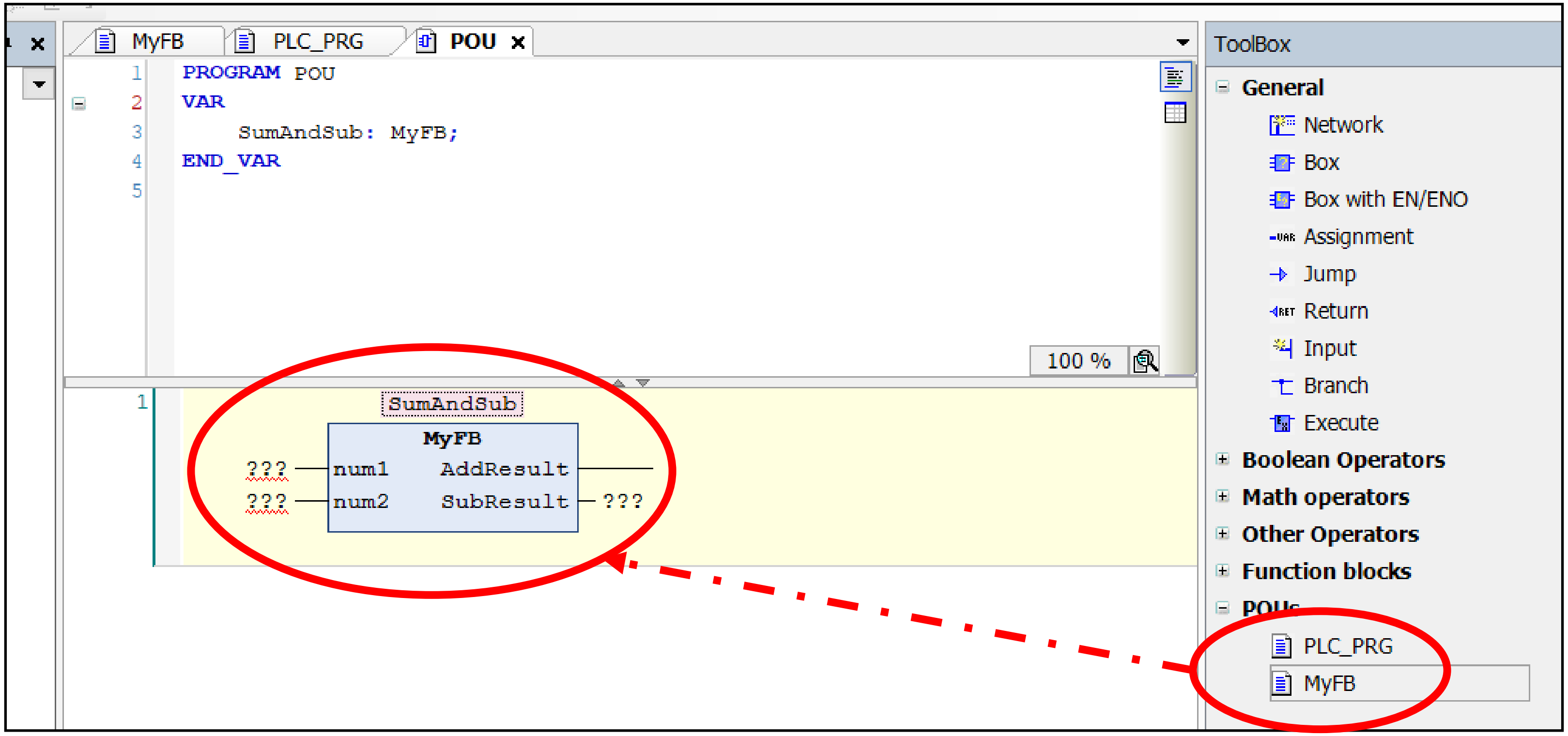
Expression	Type	Value
A	REAL	11
B	REAL	5
Sum	REAL	16
Subtr	REAL	6
⊞ SumAndSubtract	MyFB	

```
SumAndSubtract(num1 11 := A 11 , num2 5 := B 5 );

Sum 16 := SumAndSubtract.AddResult 16 ;
Subtr 6 := SumAndSubtract.SubResult 6 ; RETURN
```

Call from another POU

Now, your amazing FB can be used in another POU!

- Instantiate a FBD POU
- Find `MyFB` in the toolbox
- Instantiate and try it!



```
PROGRAM POU
VAR
    SumAndSub: MyFB;
END_VAR
```

Exercise

Let's code!

Implement any of the automatas that we saw so far using an FSM written using ST `CASE-SWITCH`

- Base automata

"Identify even sequences of a (even empty),

followed by one, or more, or no, b, ended by c"

- The traffic light

- Whatever you want!

You might want to use Function blocks to separate and test different functionalities using different POUs

How to run the examples

Let's code!

- Find them in `Code/` folder from the course website

To download Codesys, ask the teacher, or open an issue in our GitHub page

References

Course website

- http://hipert.unimore.it/people/paolob/pub/Industrial_Informatics/index.html

My contacts

- paolo.burgio@unimore.it
- http://hipert.mat.unimore.it/people/paolob/

Resources

- Brian Hobby, Codesys tutorials (a must to learn the tool in 5 mins)
- A small blog
  - www.google.com