Codesys

Paolo Burgio
paolo.burgio@unimore.it

UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA
High Performance
Real Time Lab

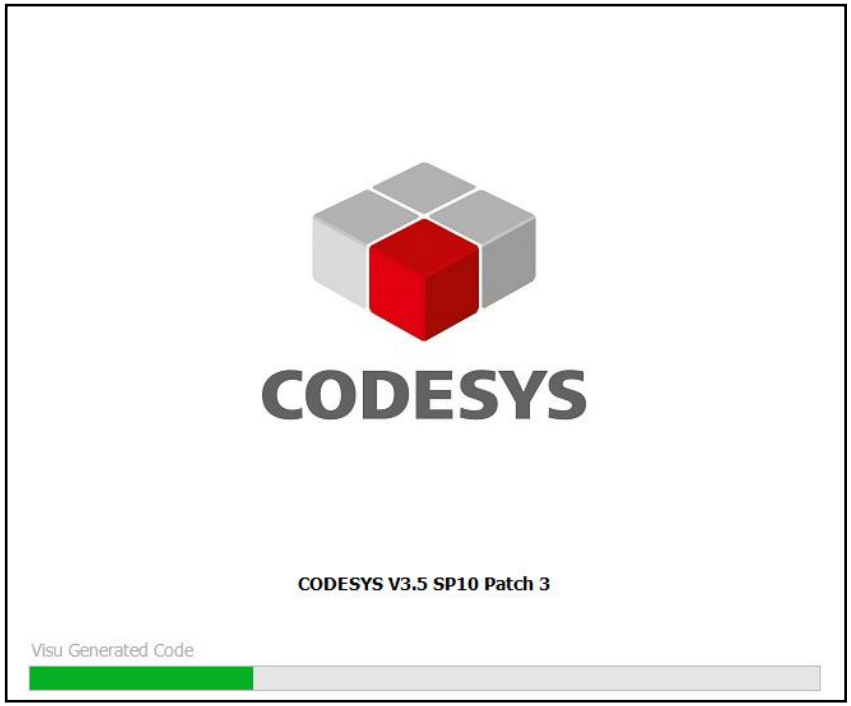
Programming is a skill best acquired by practice and example rather than from books.

ALAN TURING

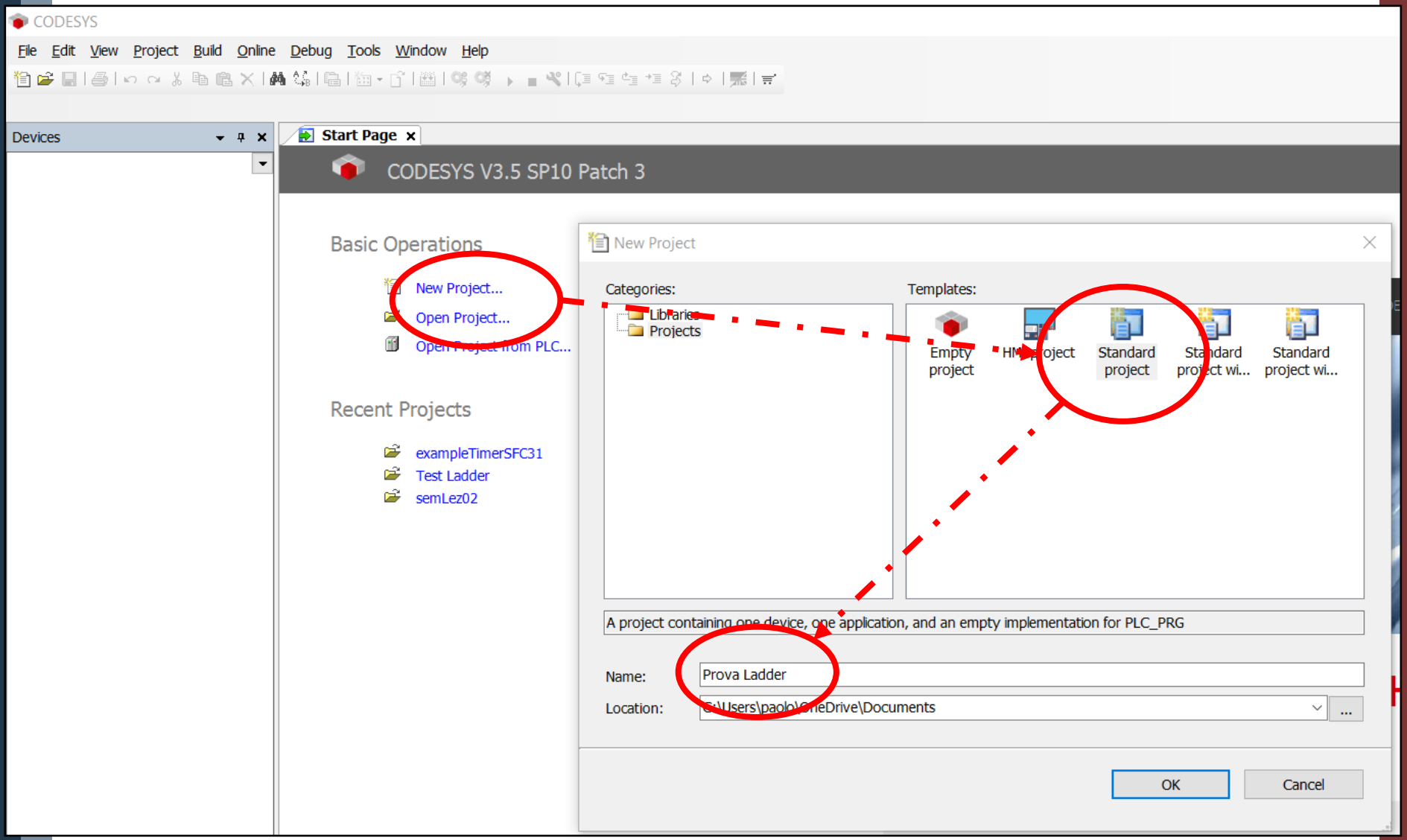
Load the main program interface

What is it?

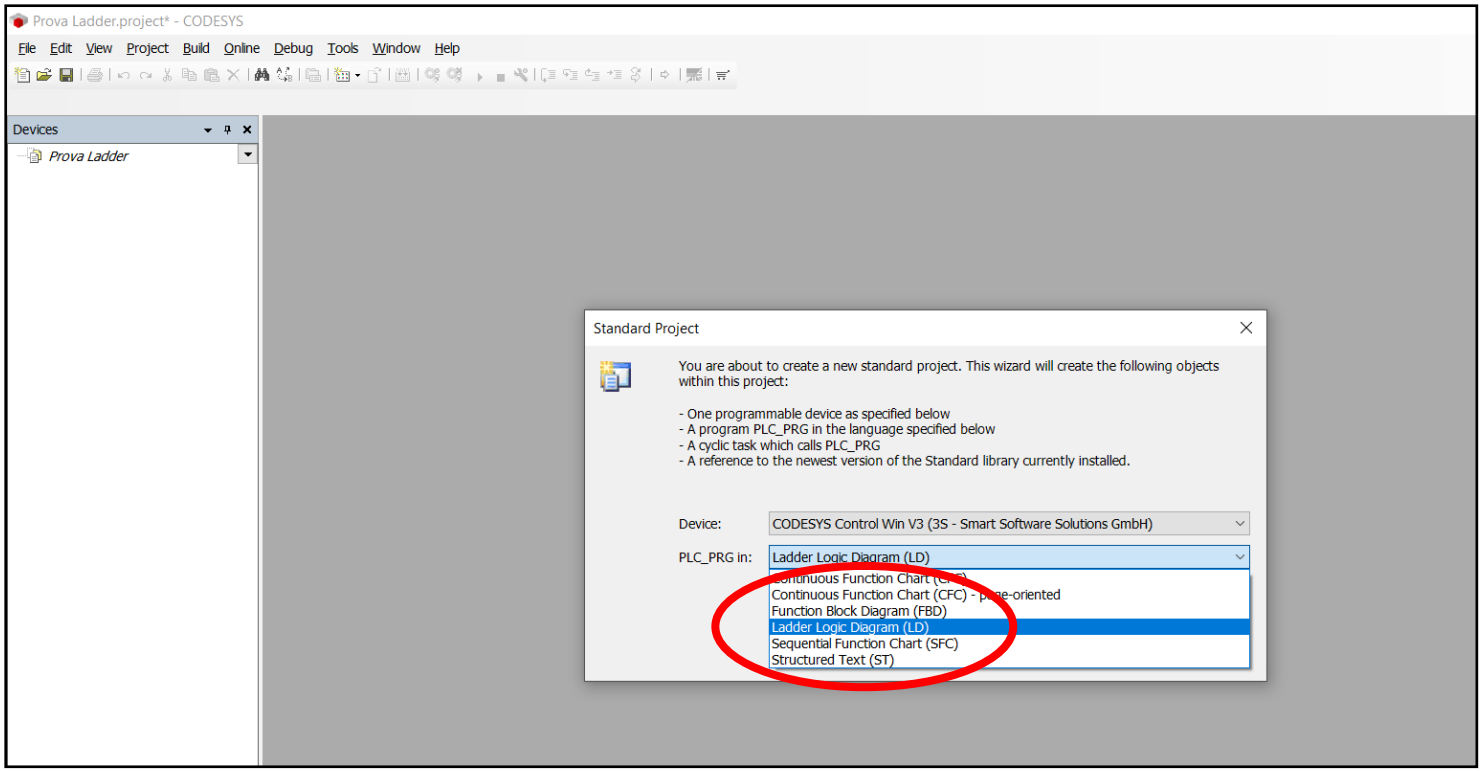
- An IDE to create PLC programs, and **simulate them**
- In any of the five main languages
- I use V3.5 SP1 patch 3, recommended version (for compatibility with the examples I'll give you)



Create a project

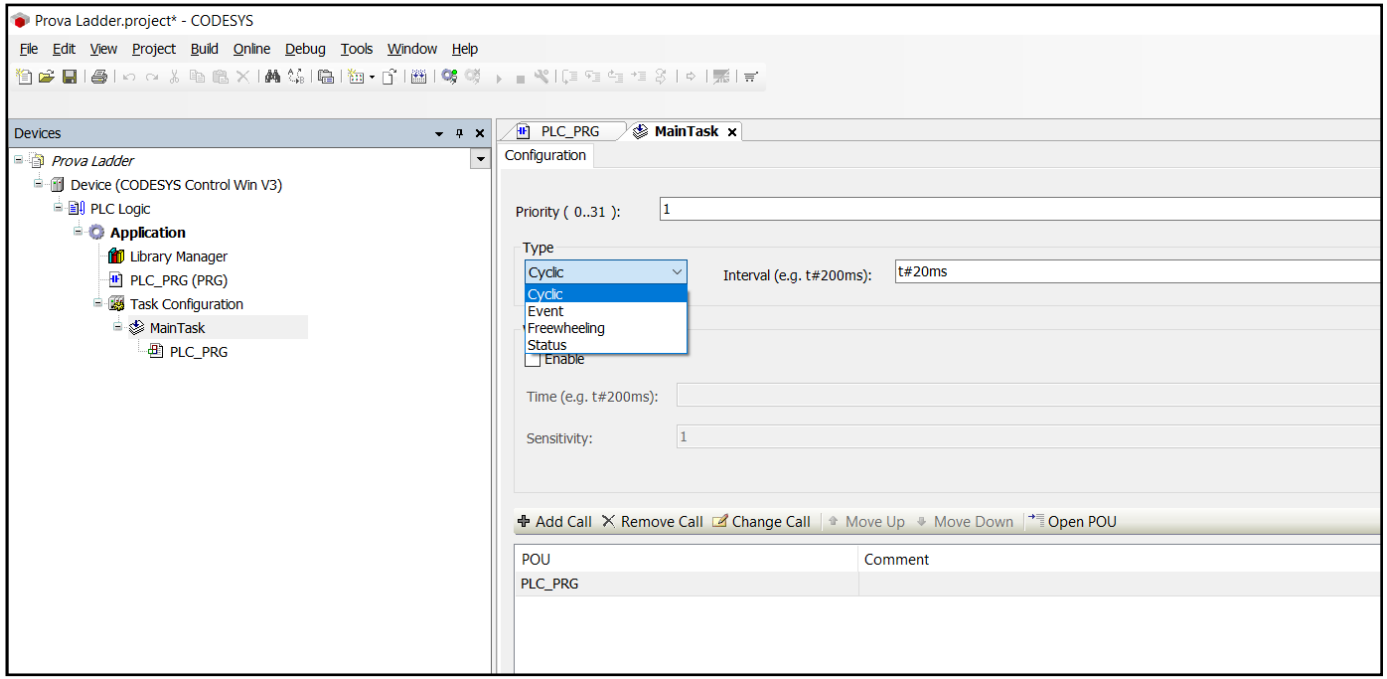


Select the language



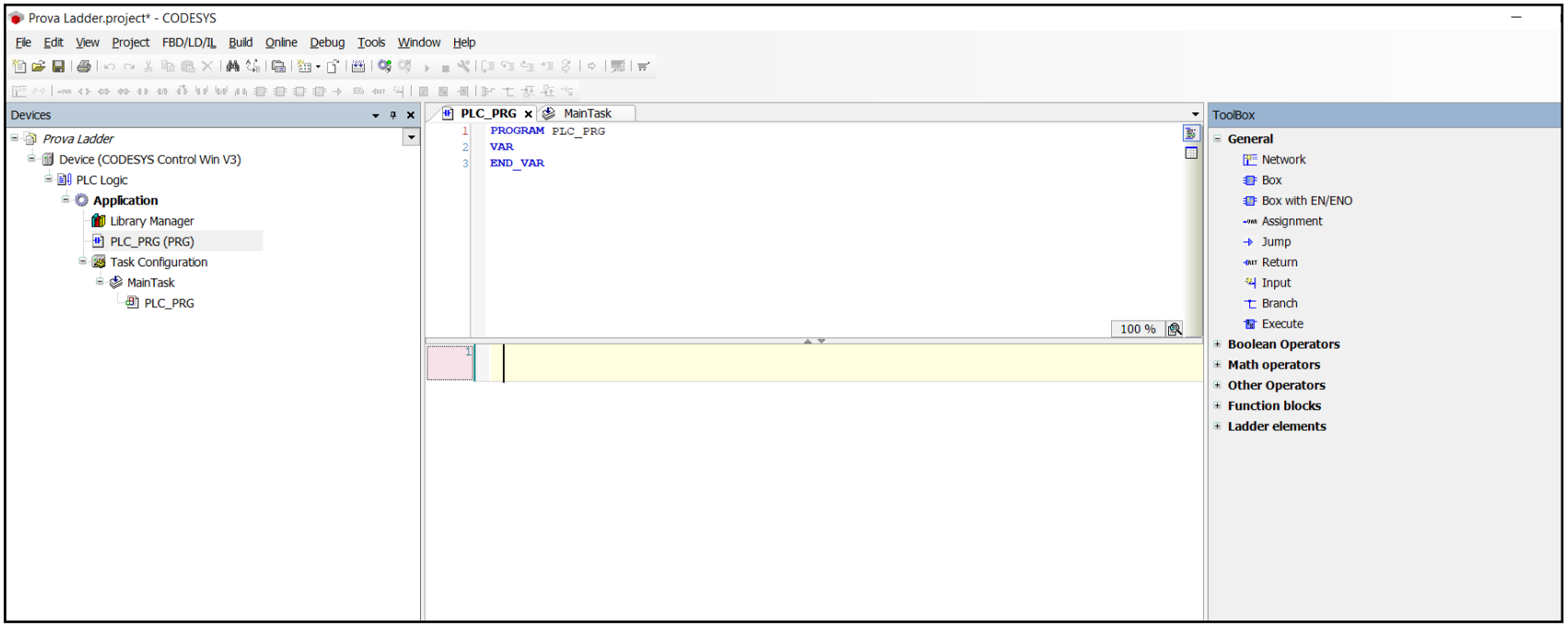
Project workbench

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



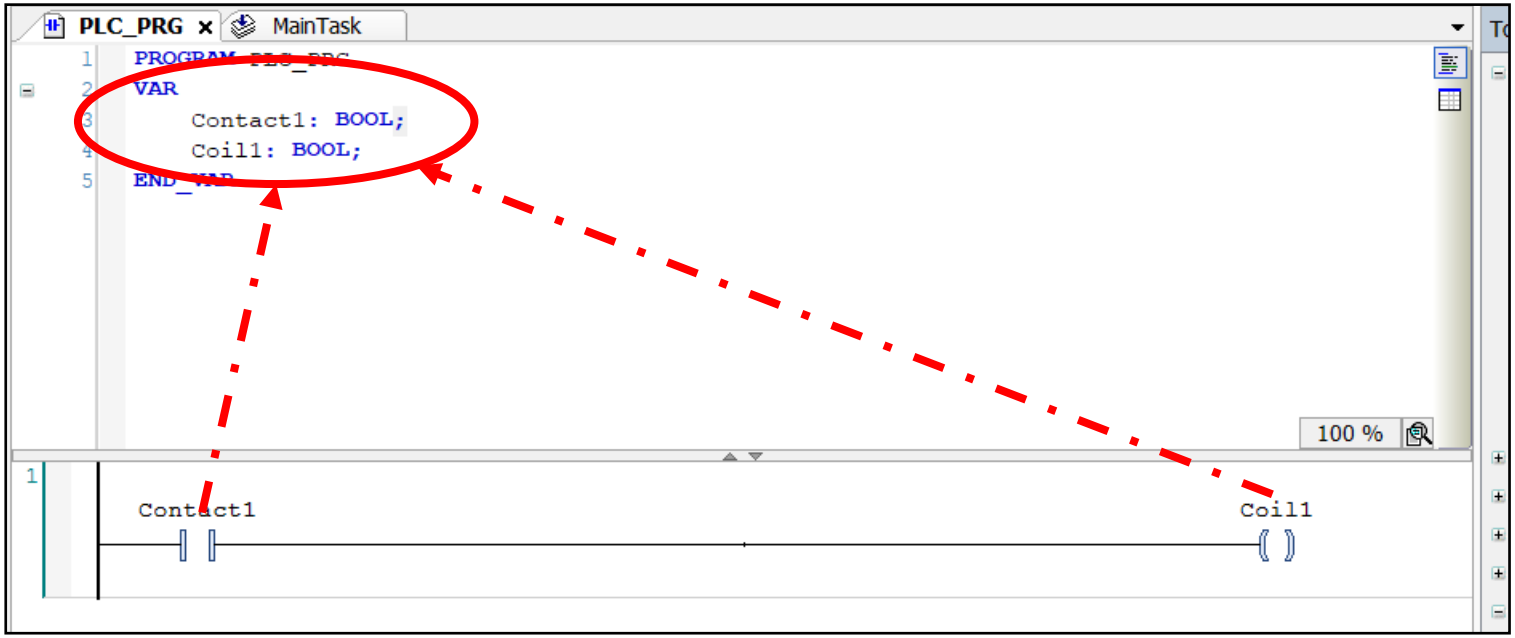
Project workbench - Ladder

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



Adding a contact + coil

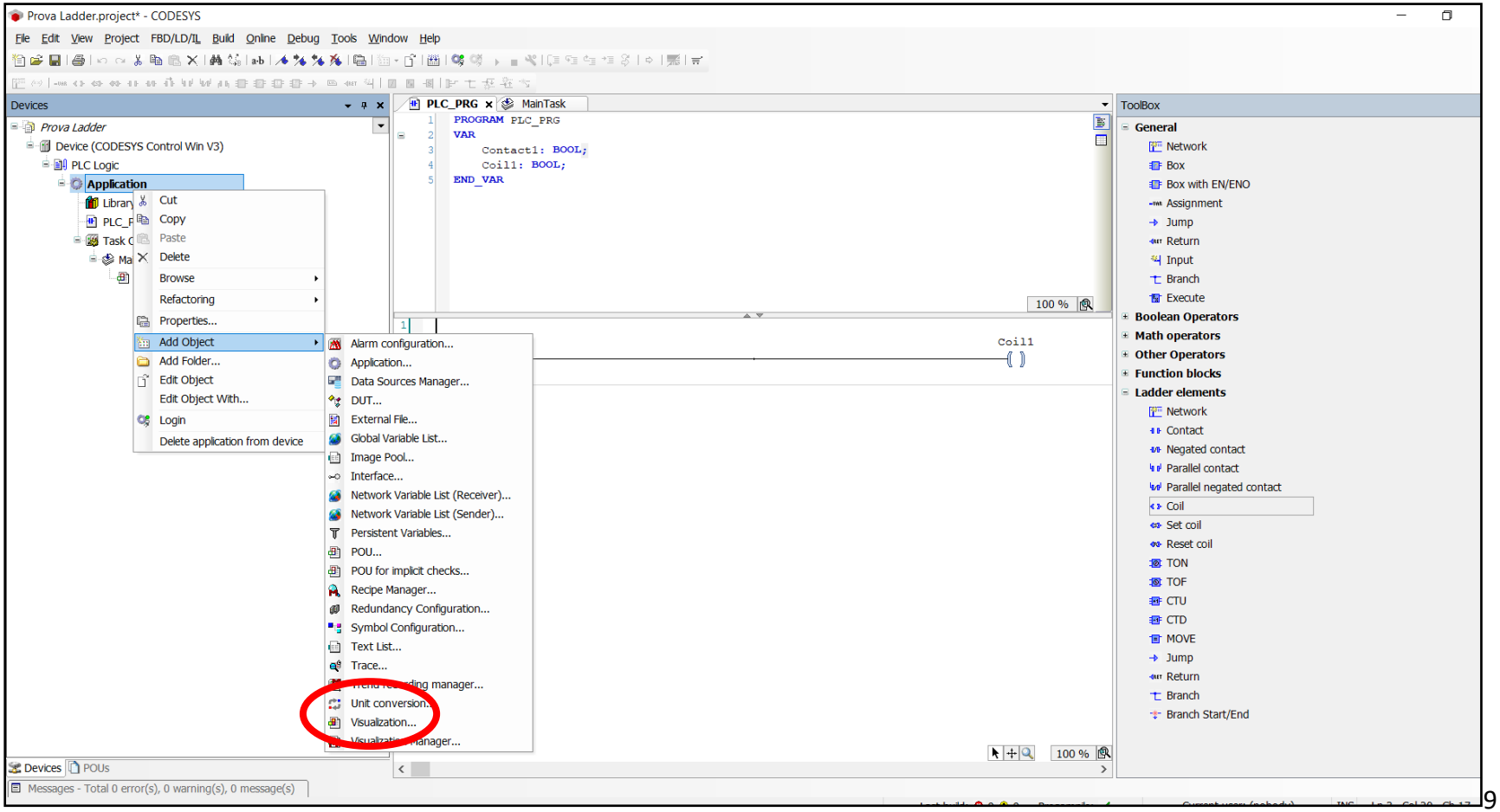
- Two global variables are automatically created in the variable definition window always in ST lang), both of `bool` type, as specified by us
- Here, we want a switch that turns on a lamp, hence we need a NO contact and a coil
- PS here you don't see the right power rail as it's implicit



View the simulated system

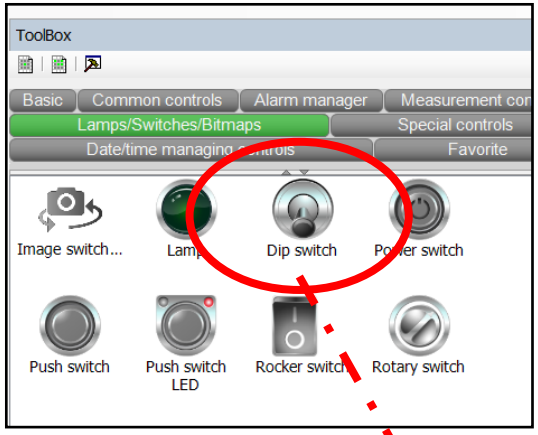
Add a Visualization object

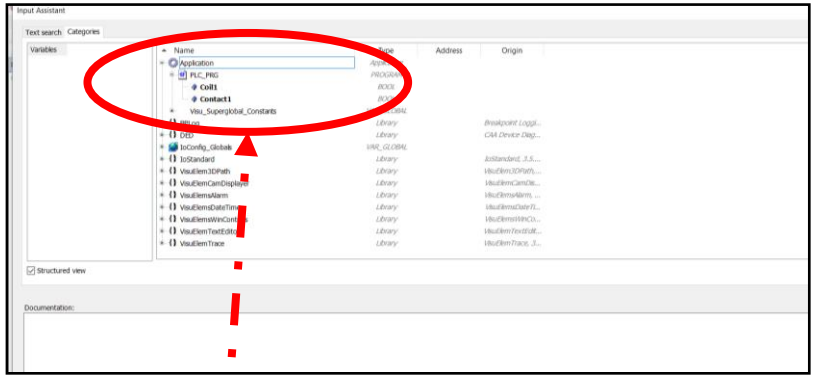
› Application -> Add Object -> Visualization

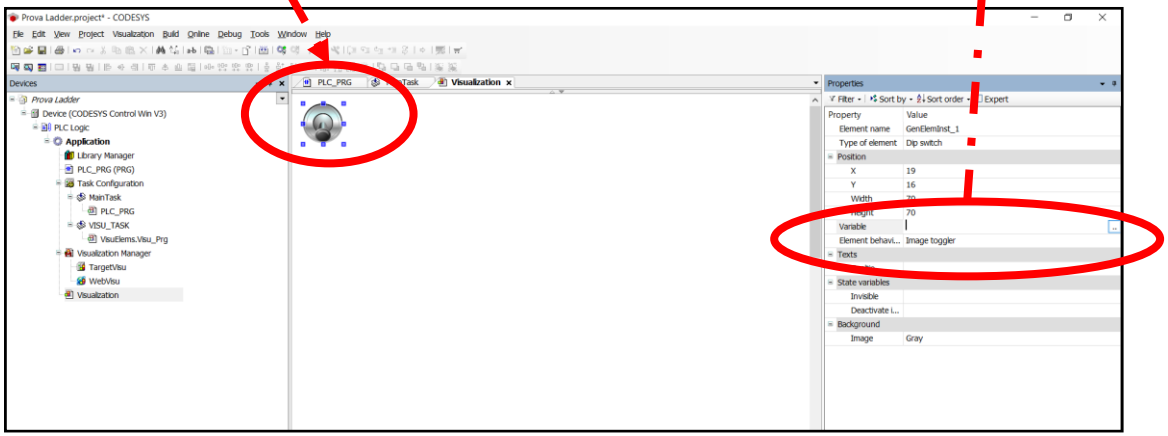


Add elements, and link to variables

- 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`

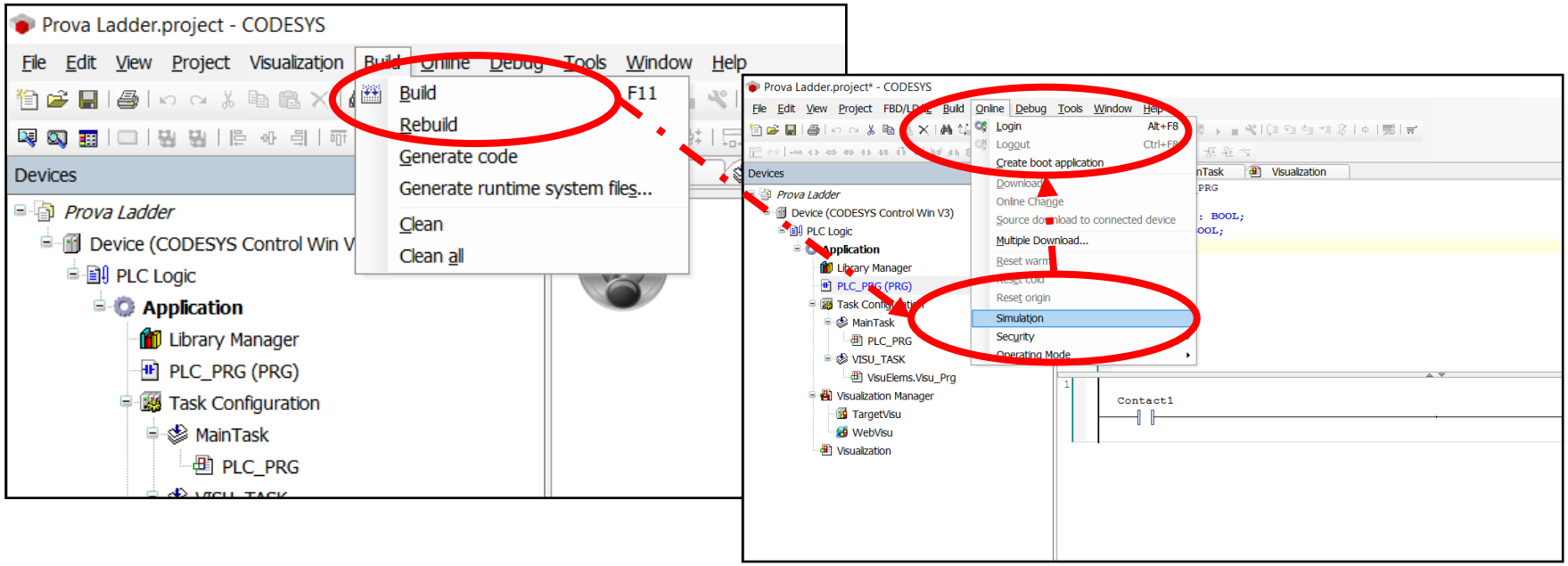






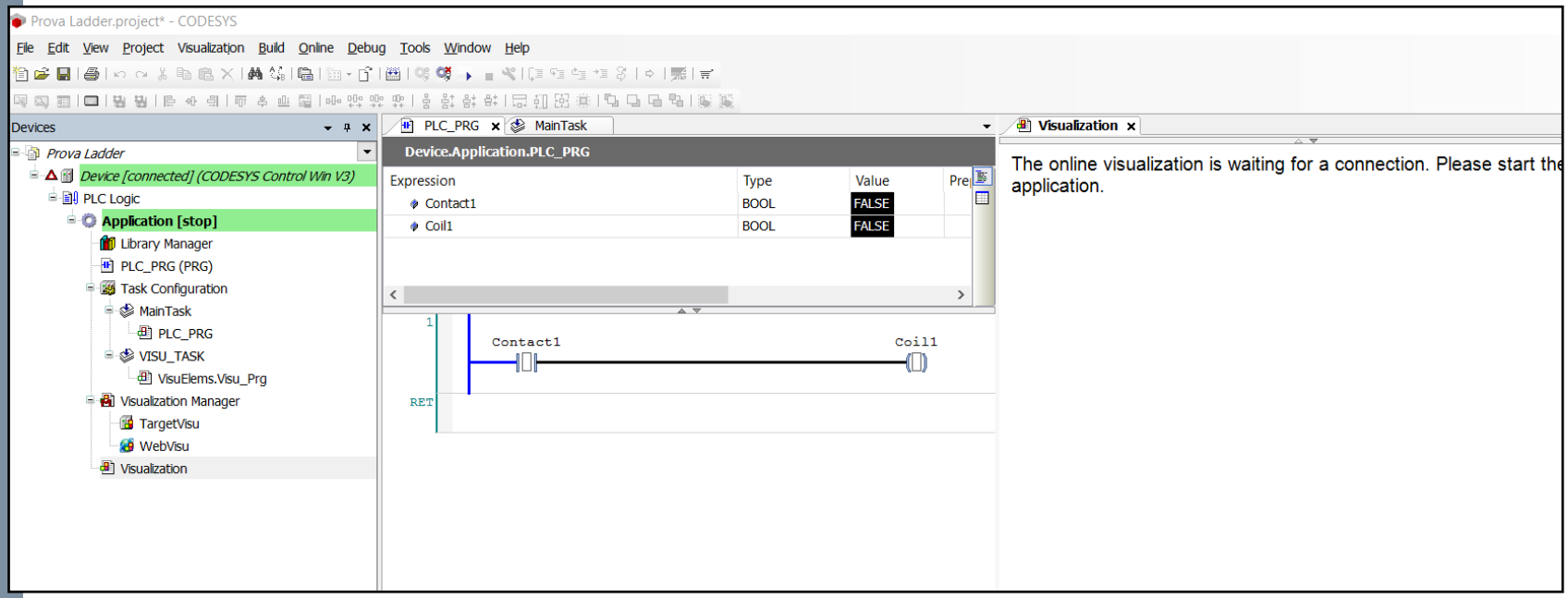
Compile and set up simulator

- 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



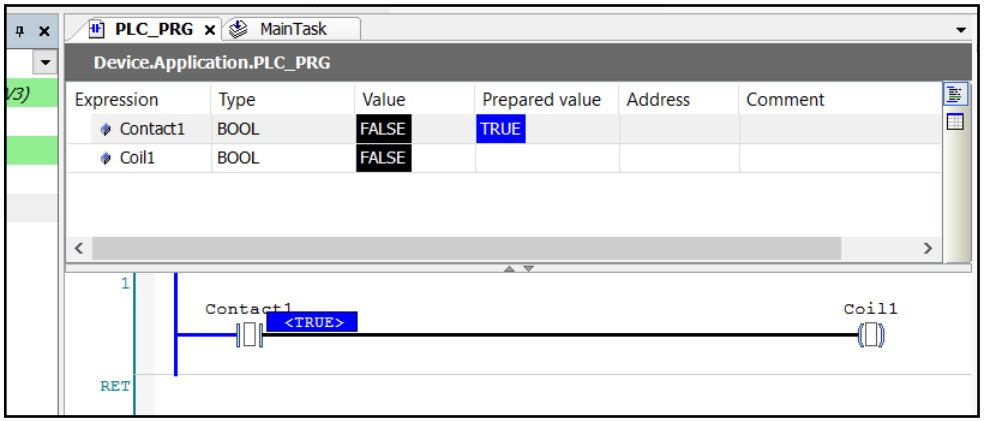
Run workbench

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



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)



- 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"

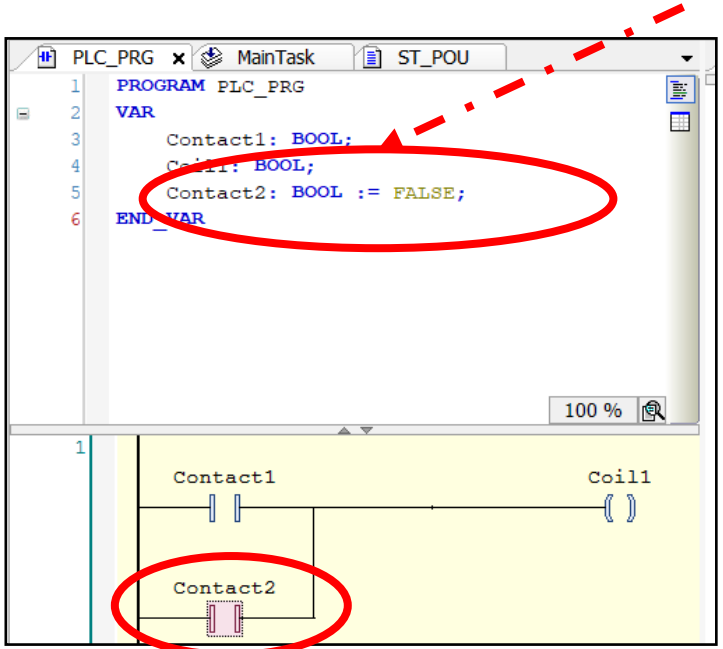
› ..easy, simply drag&drop

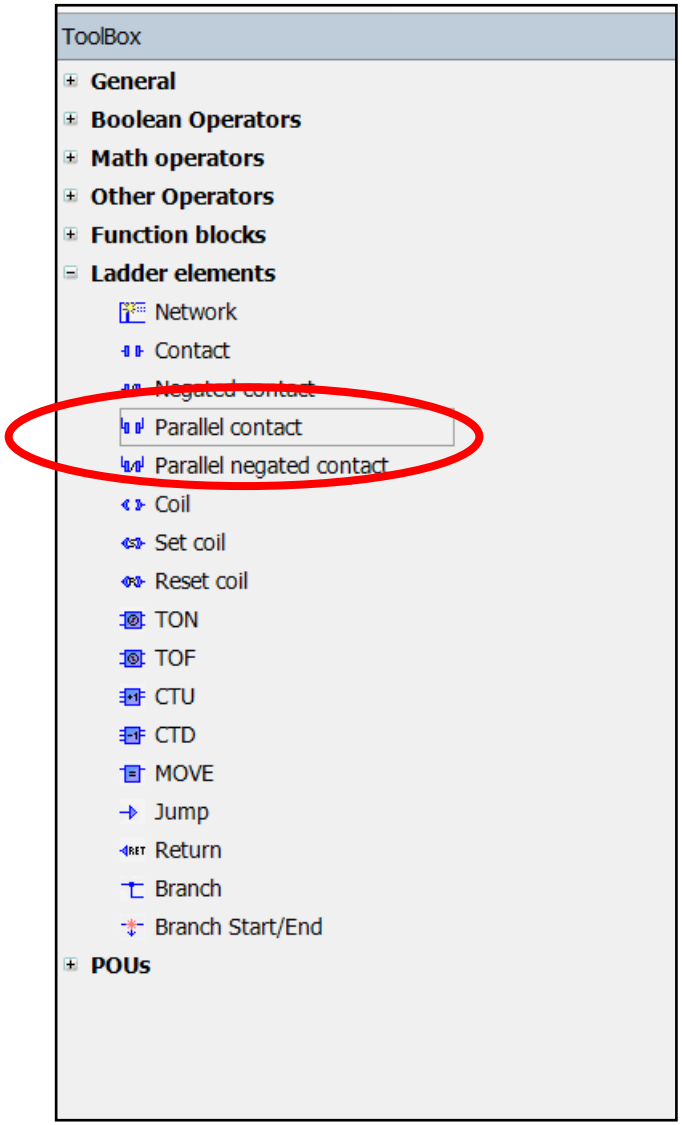
Logical "OR"

› "Parallel contact" components from toolbox

› IDE helps us to insert it...

PS good programmers remember to initialize vars ;)

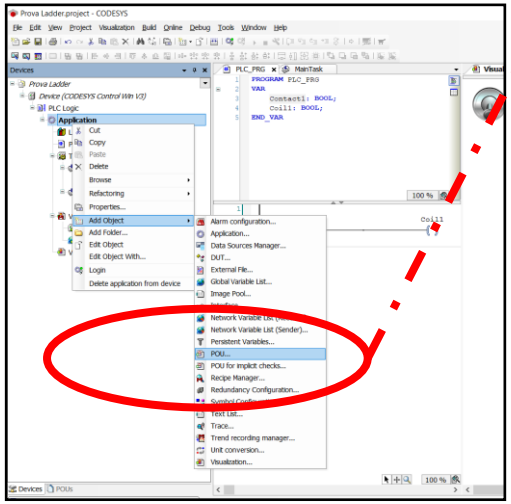


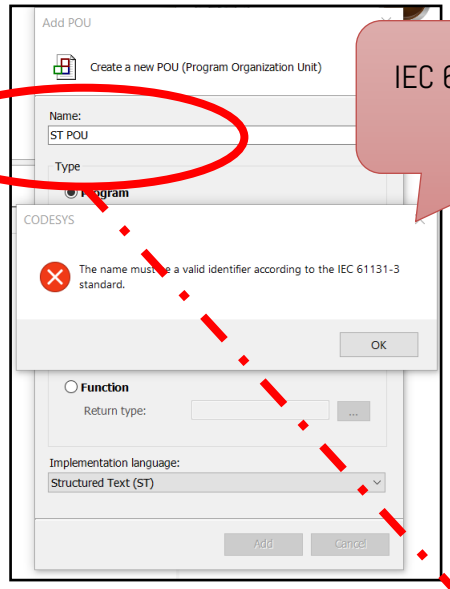


Structured Text

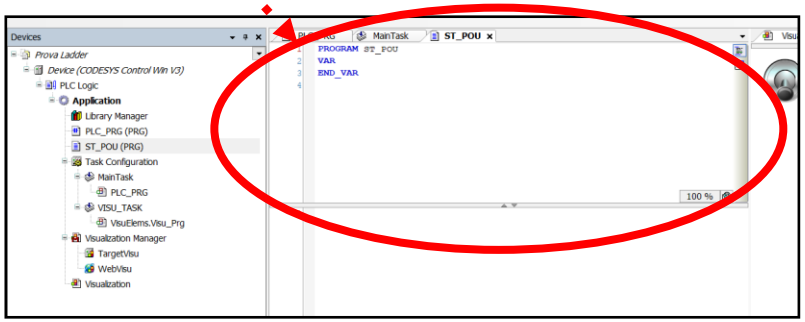
Add new ST POU

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

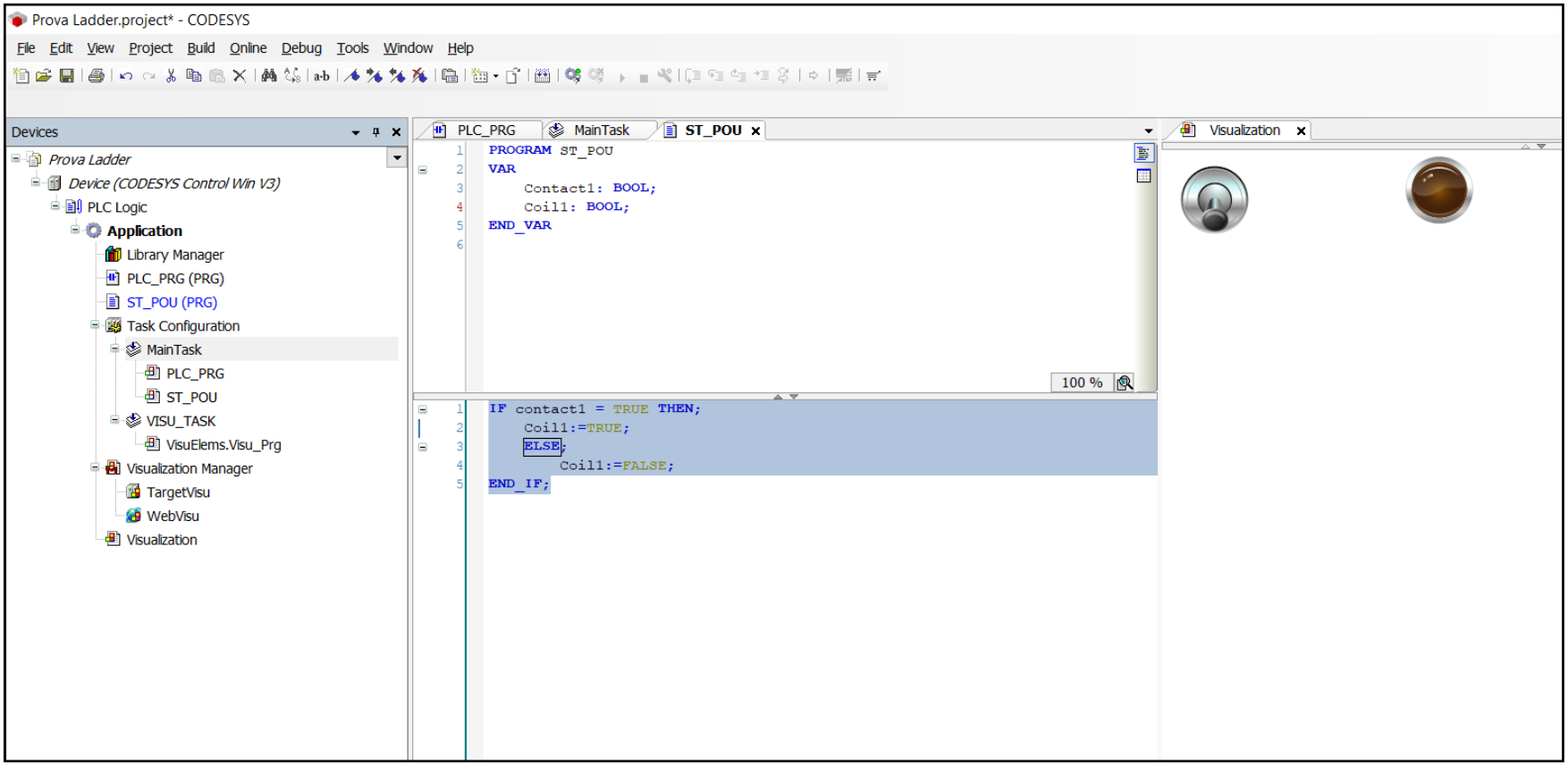




IEC 61131 does not allow spaces in names

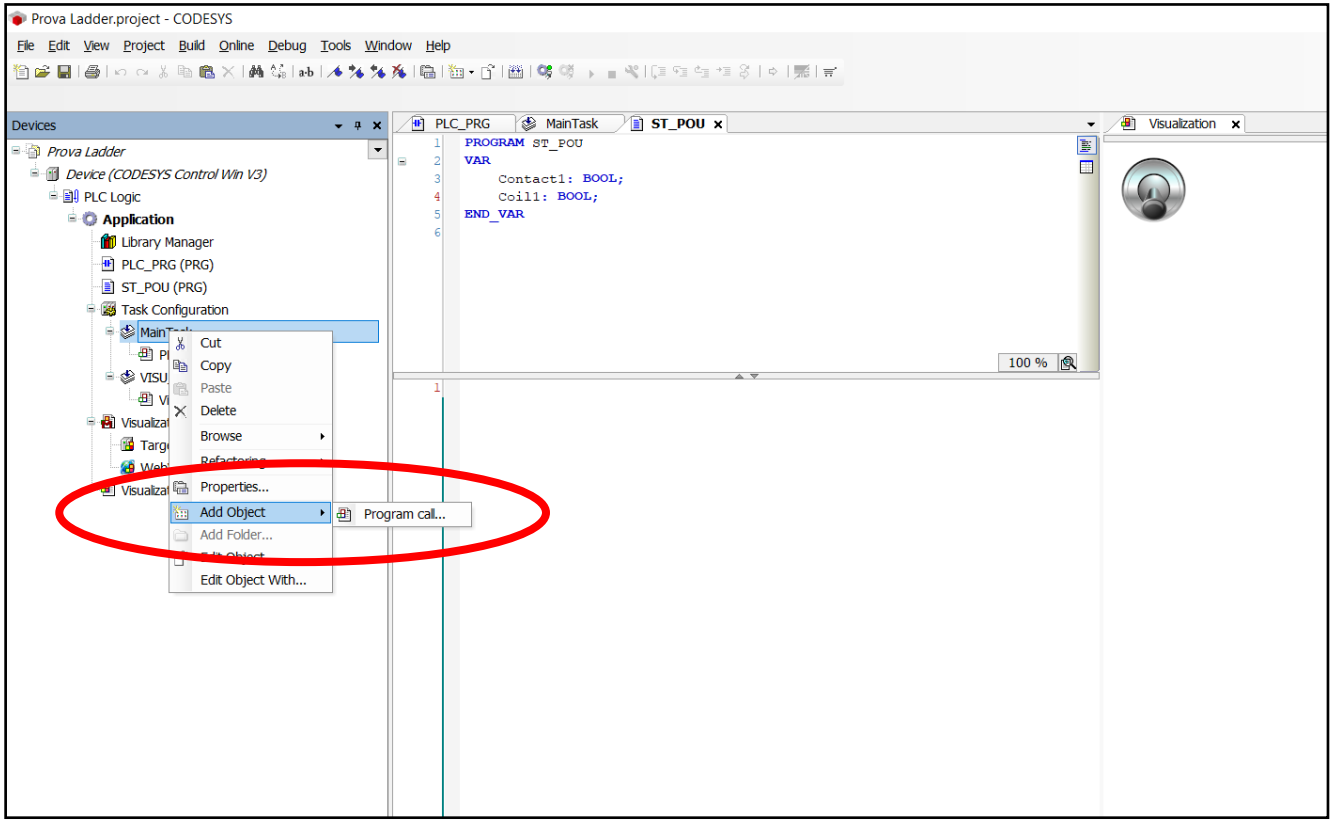


Write the ST code



Are we done? Not yet…

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



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…

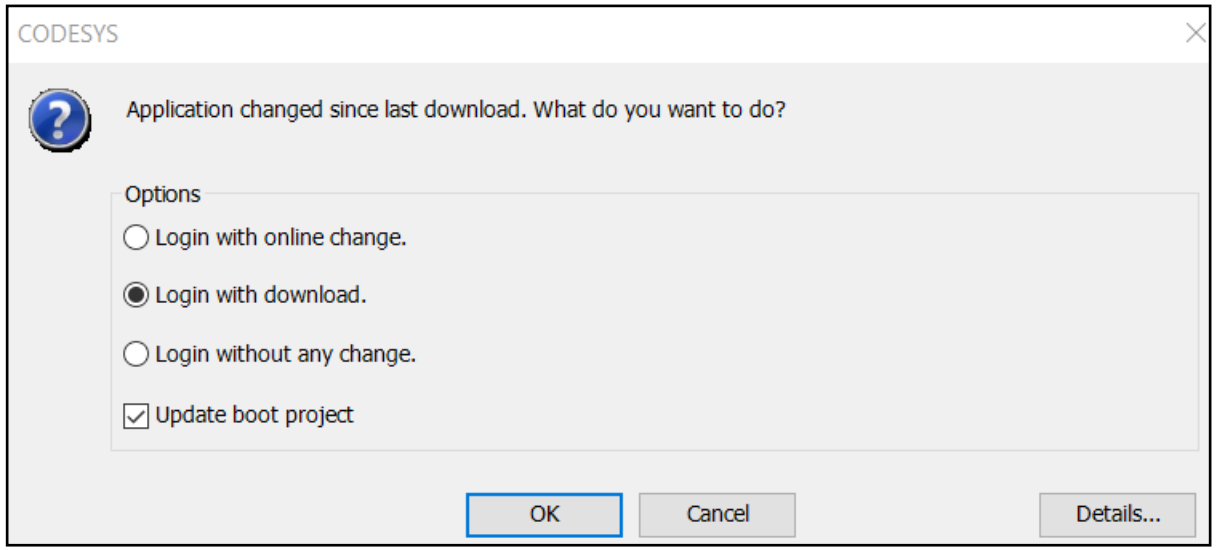
Should we attach those vars to the two simulated objects?

- (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

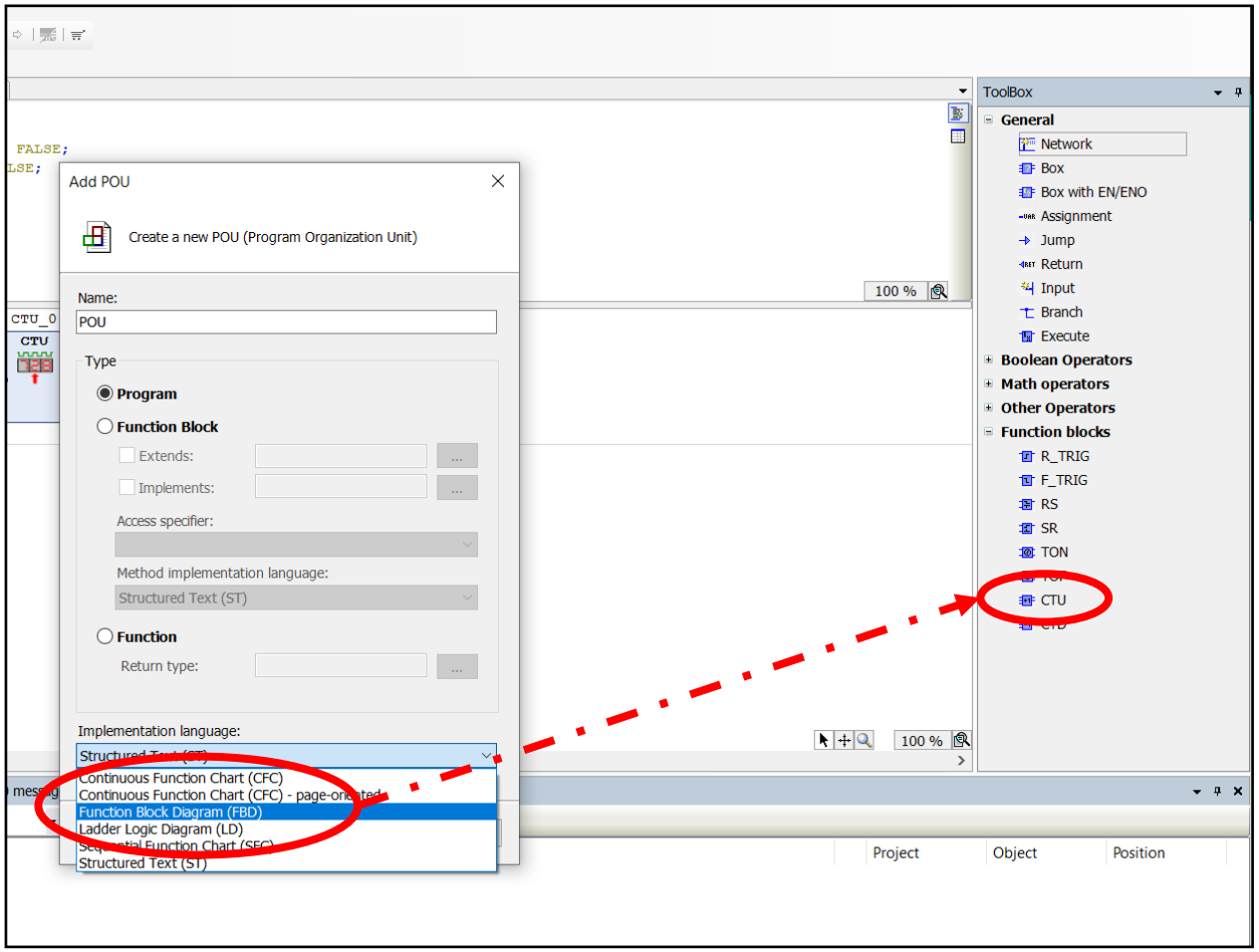
› Codesys will prompt us



Programming with Function Blocks

Counters - CTU

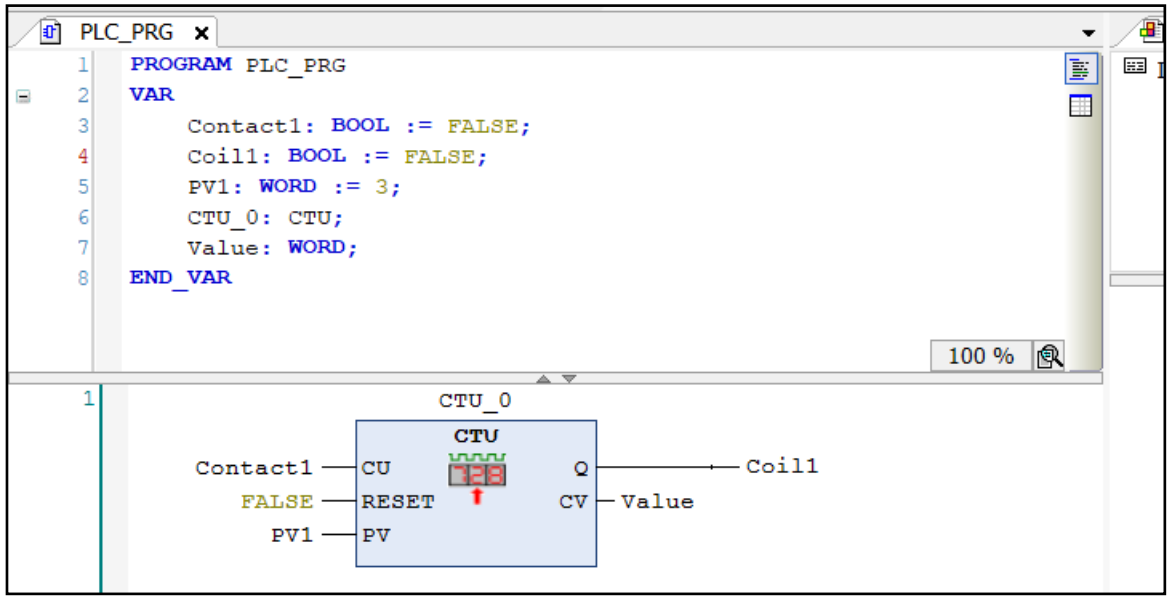
- Create a program, or add a POU of type “FBD”
- Then, drag a Counter (CTU) in the workbench



Bind the CTU

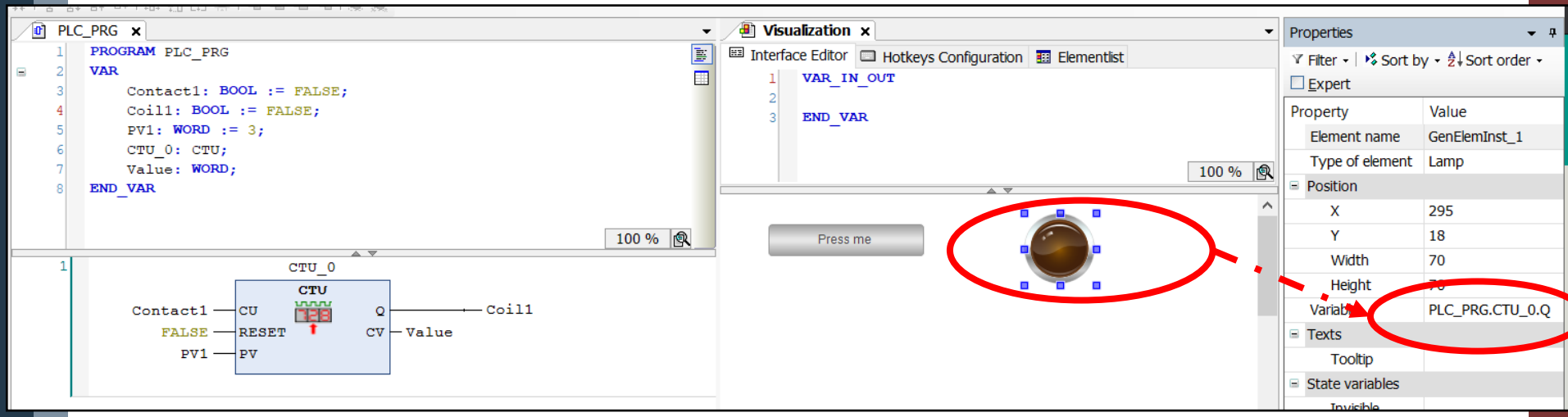
Connect CTU in&outs to vars

- `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



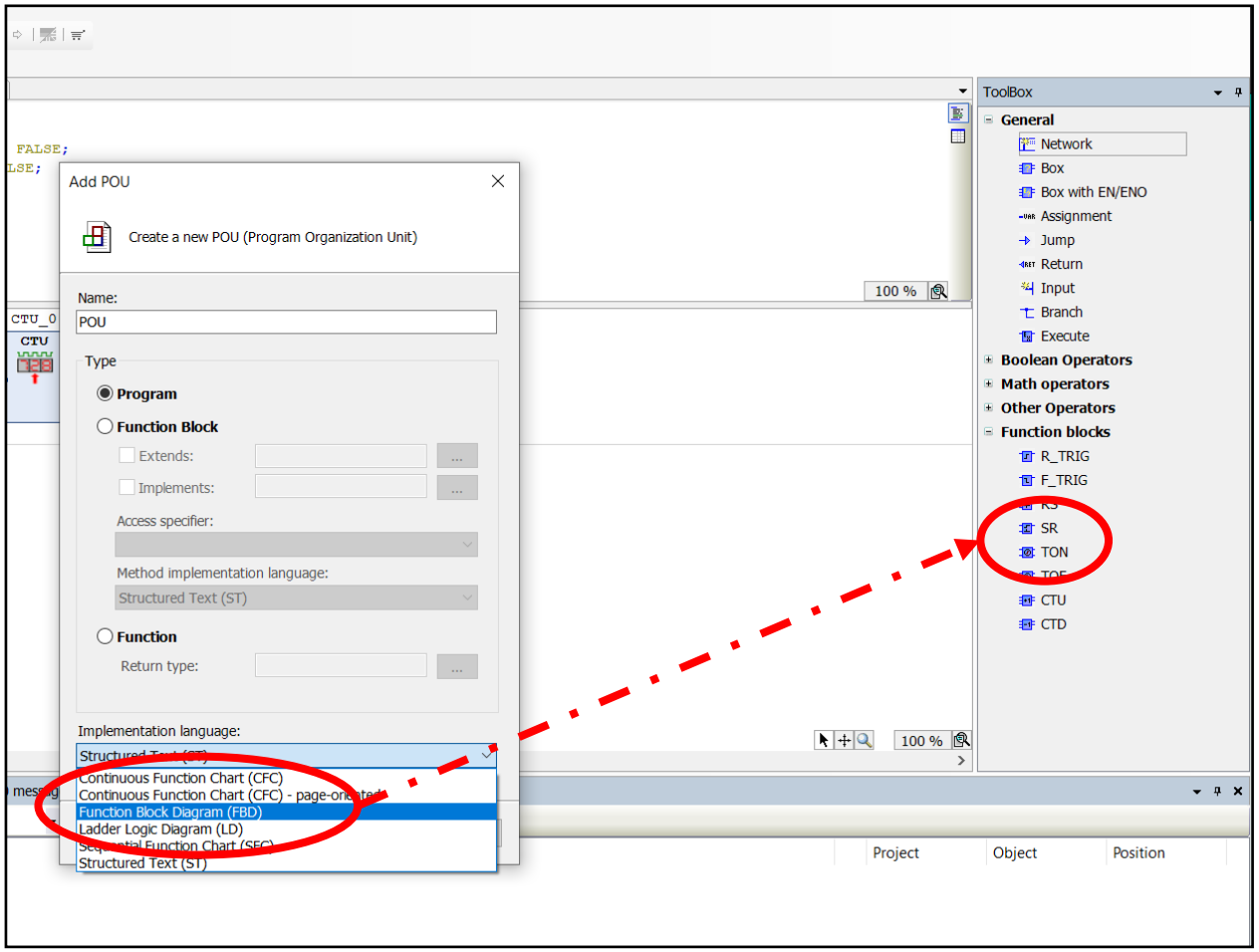
Add visualization

- 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`



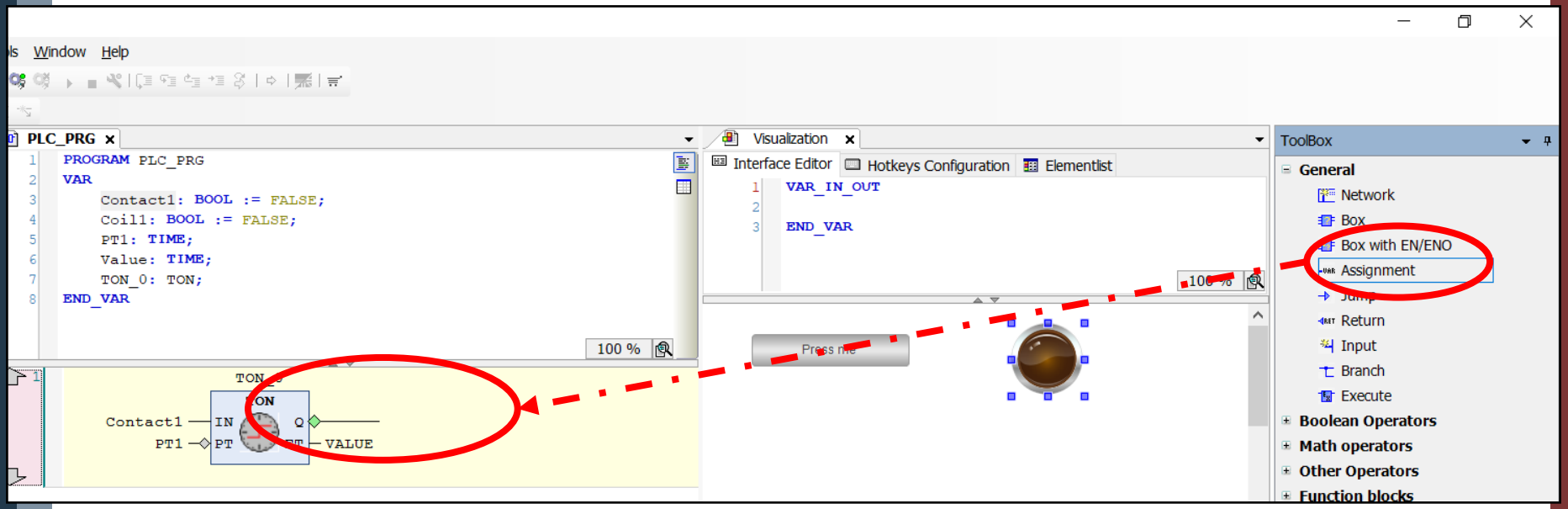
Timers – TON

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



Bind the TON

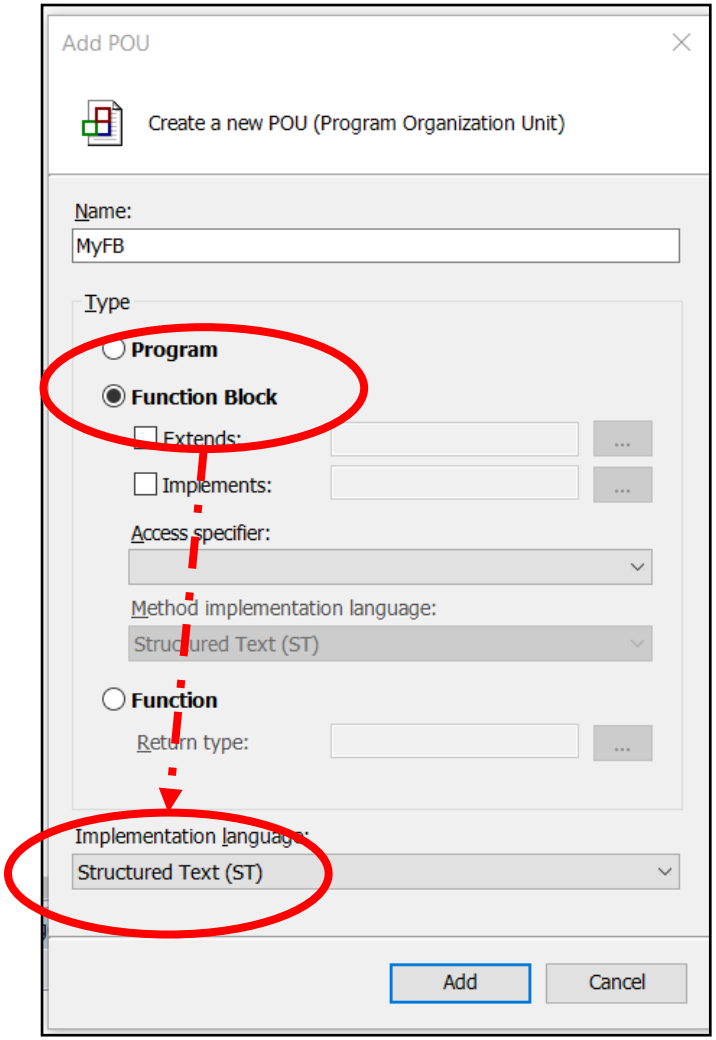
- `IN` to a contact (press&hold/standard button)
- Remember, IN starts timer @its rising edge, and resets @its falling edge
- `PT1` is a `TIME`
- `Q` to a coil (we want to turn on a lamp) using the **assignment** operator
- `ET` to a `TIME` variable, to monitor the status



Defining Function Blocks

Create a new POU of type Function Blocks

› (of course, in ST)



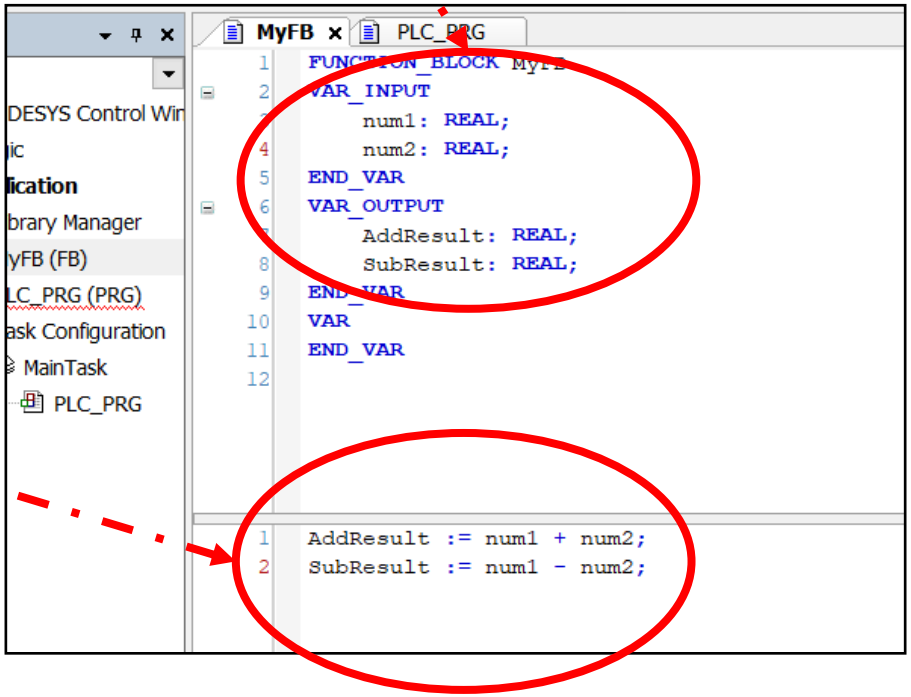
Implement the Function Block

Add in/out vars

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

Add FB logics

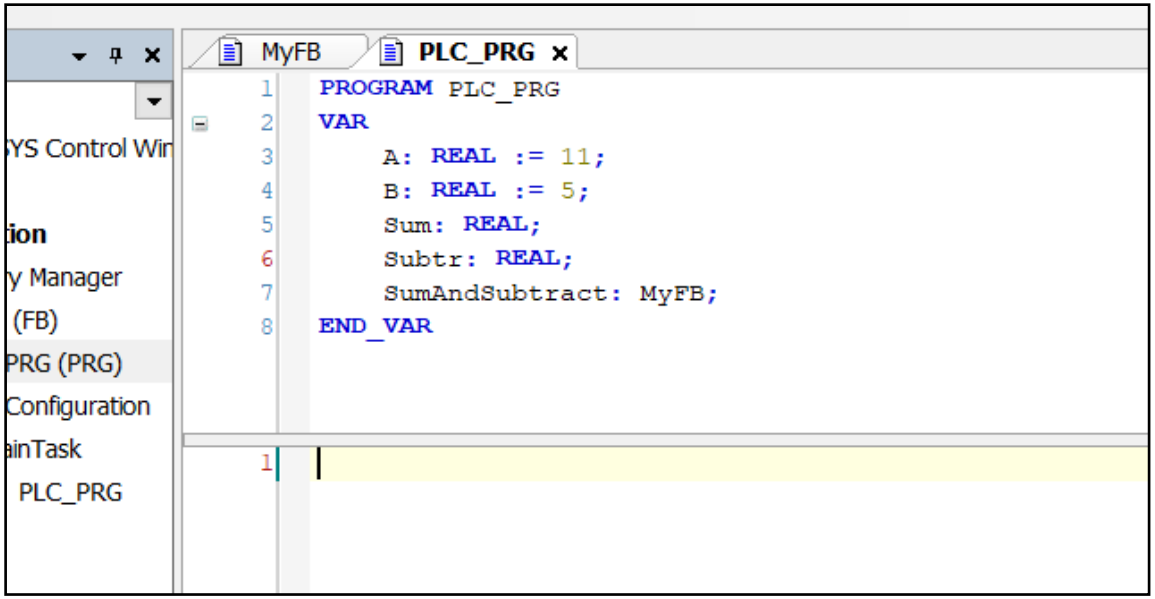
› In ST workbench



Use it in the Application

In the main POU, create vars

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



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

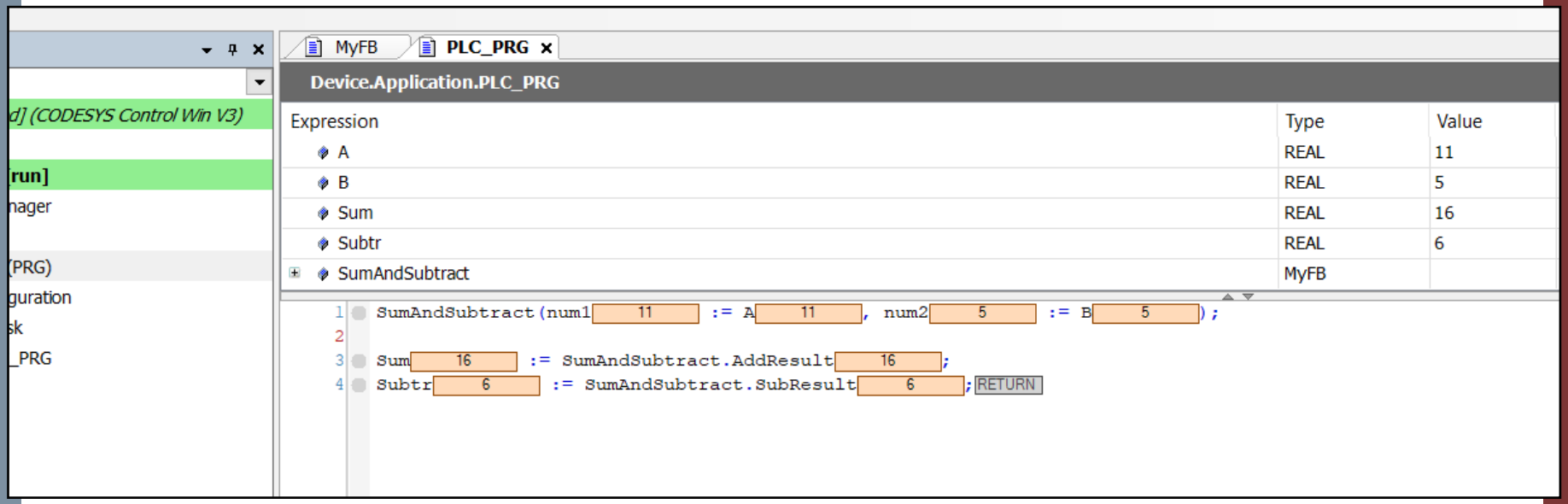
Slightly different than in "traditional" programming languages

- Why?

```
8   END_VAR
```

```
1   SumAndSubtract(num1 := A, num2 := B);
2
3   Sum := SumAndSubtract.AddResult;
4   Subtr := SumAndSubtract.SubResult;
```

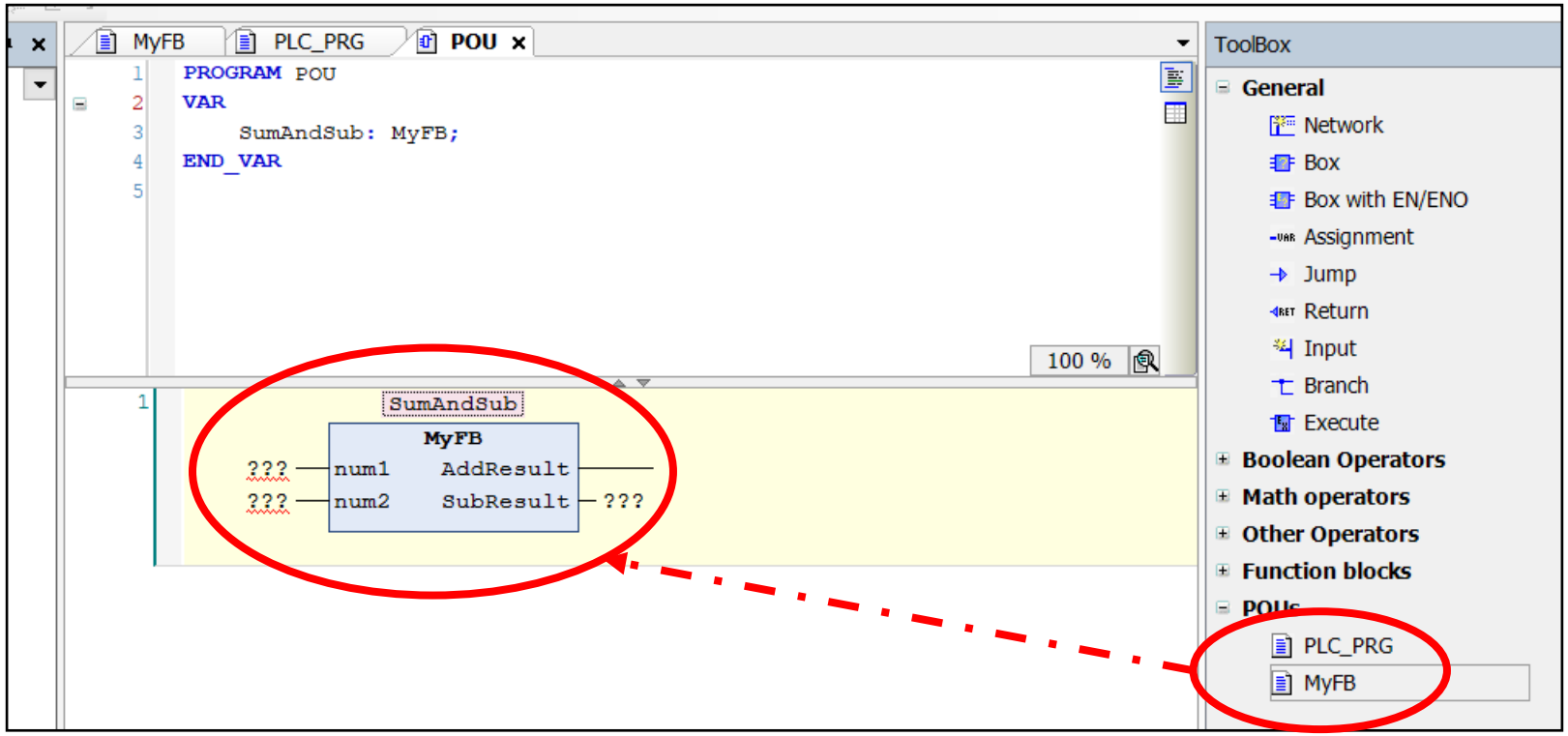
Simulate..and enjoy! ☺



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!



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