

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

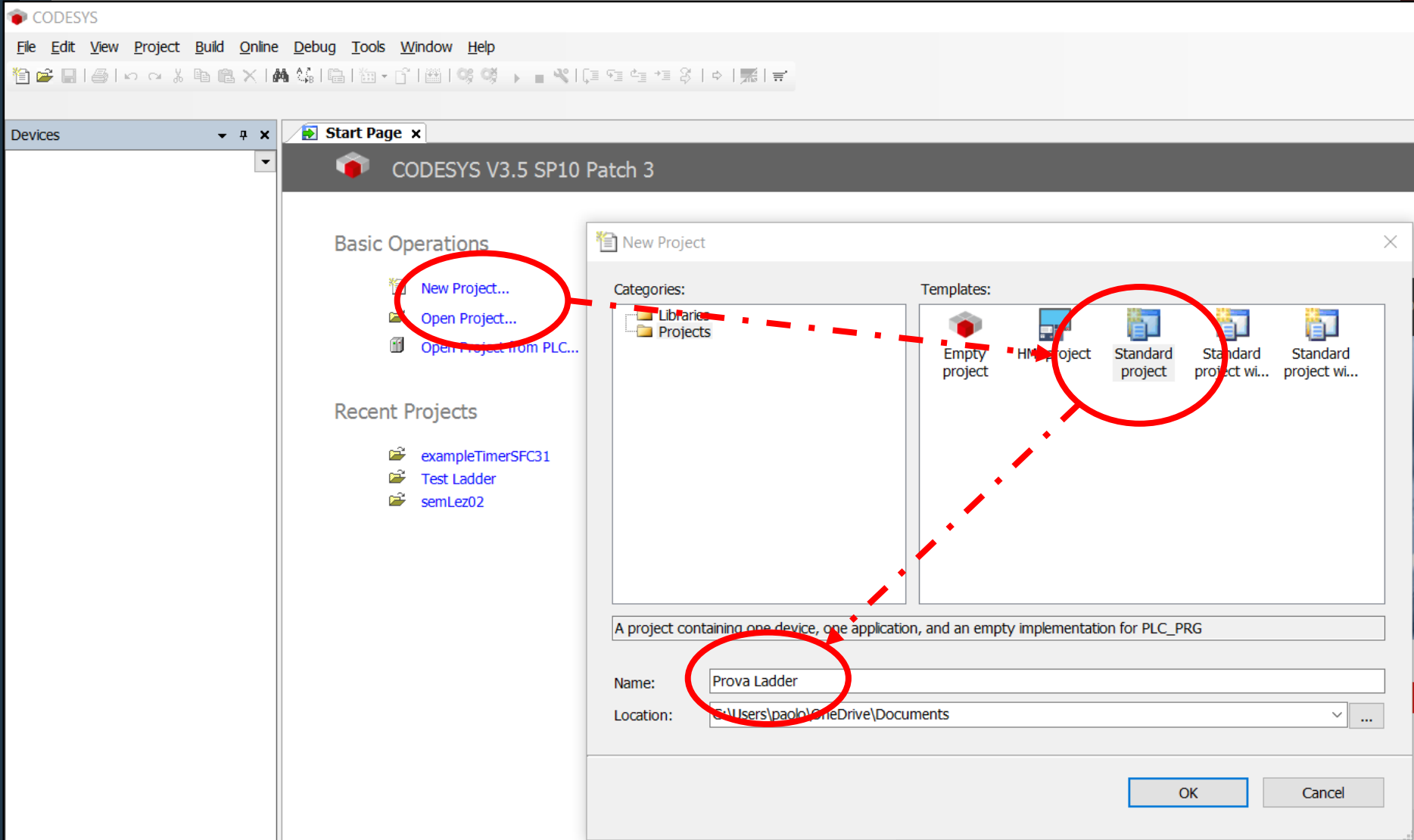
What is it?

- › 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)



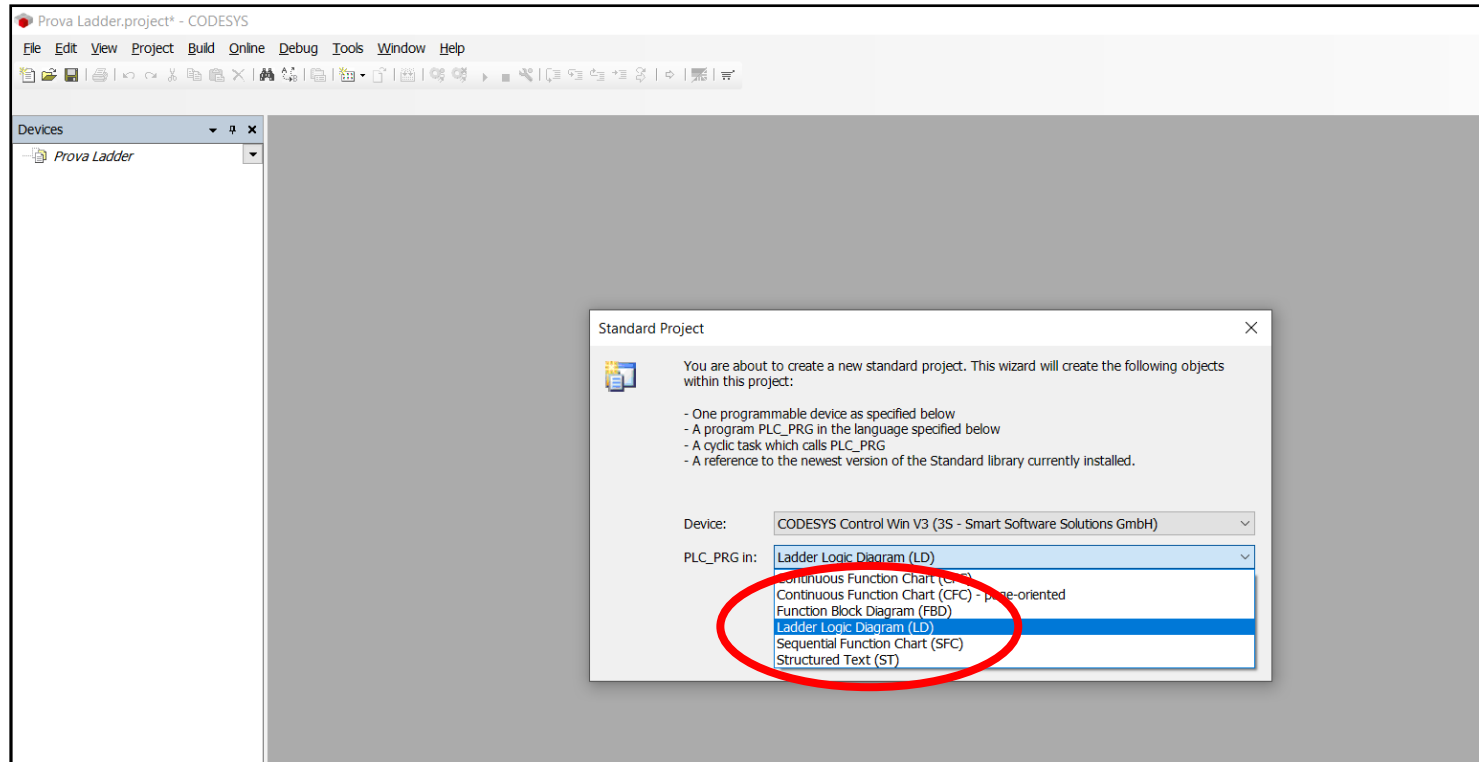


Create a project





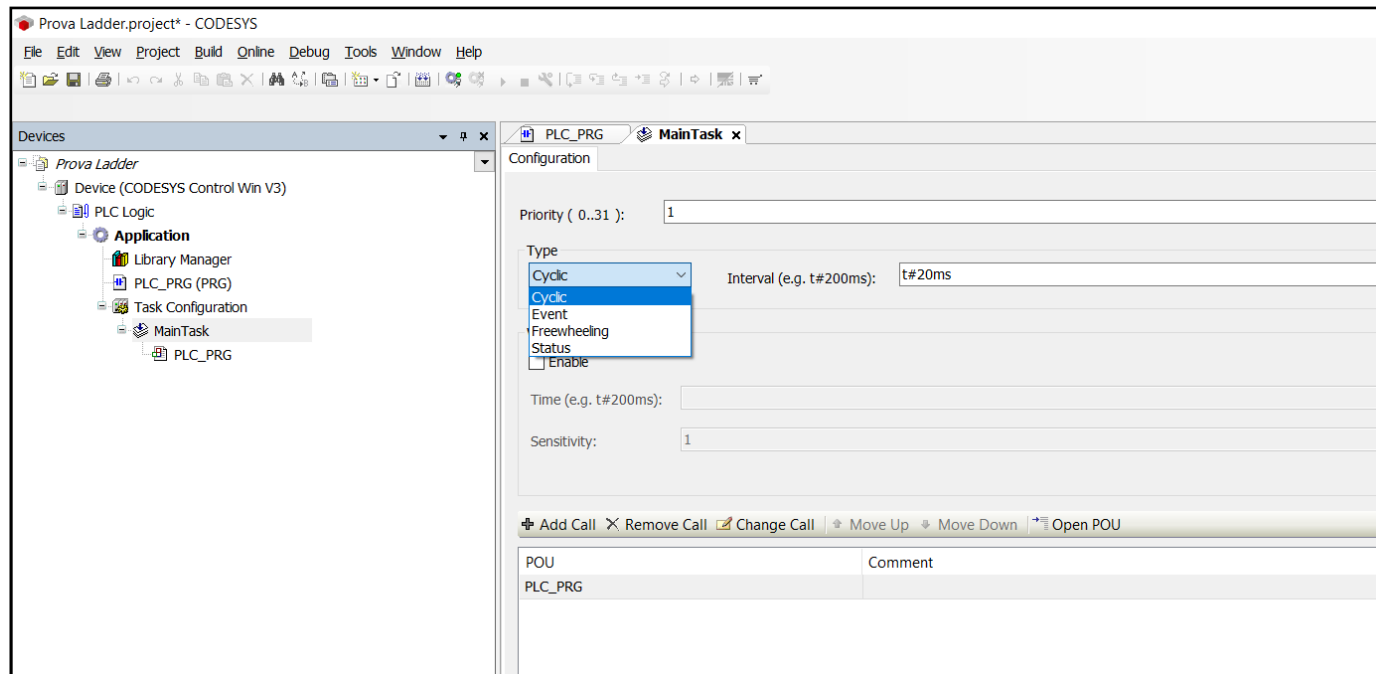
Select the language





Project workbench

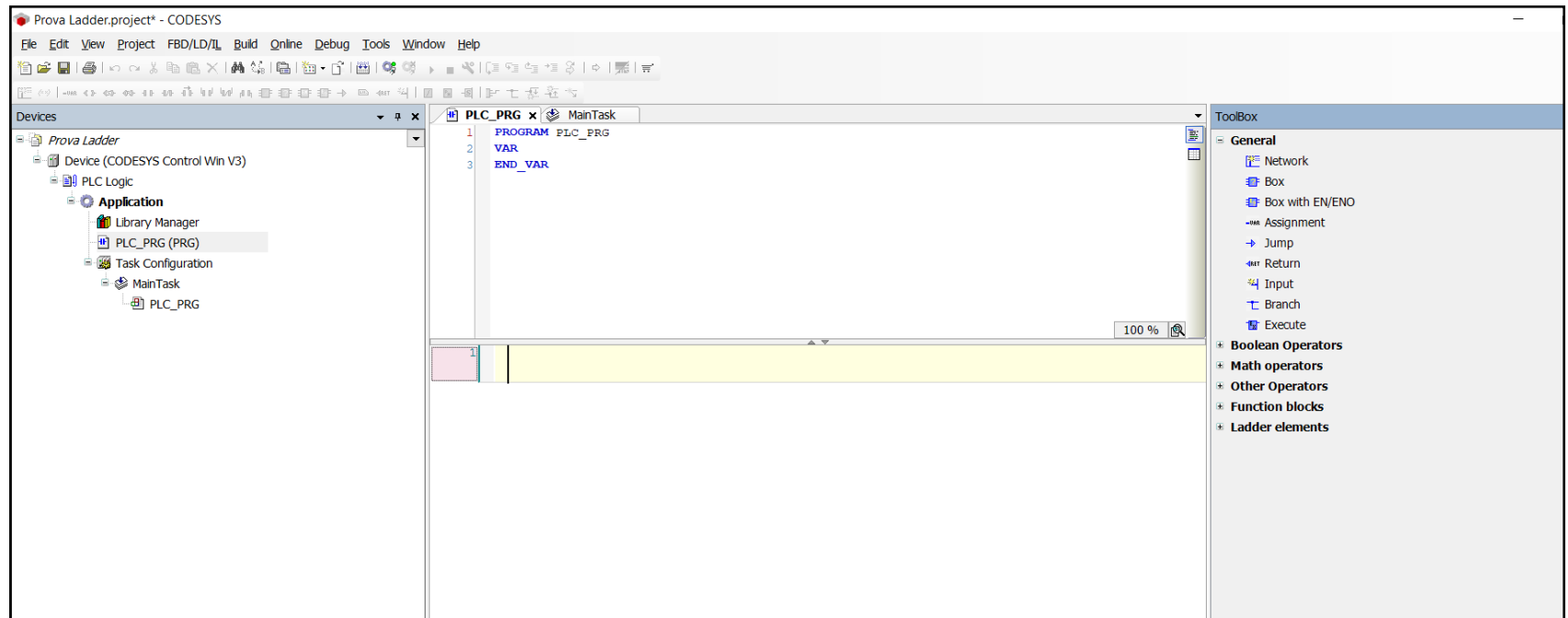
- › Your application has a Main task, that (here) runs cyclically





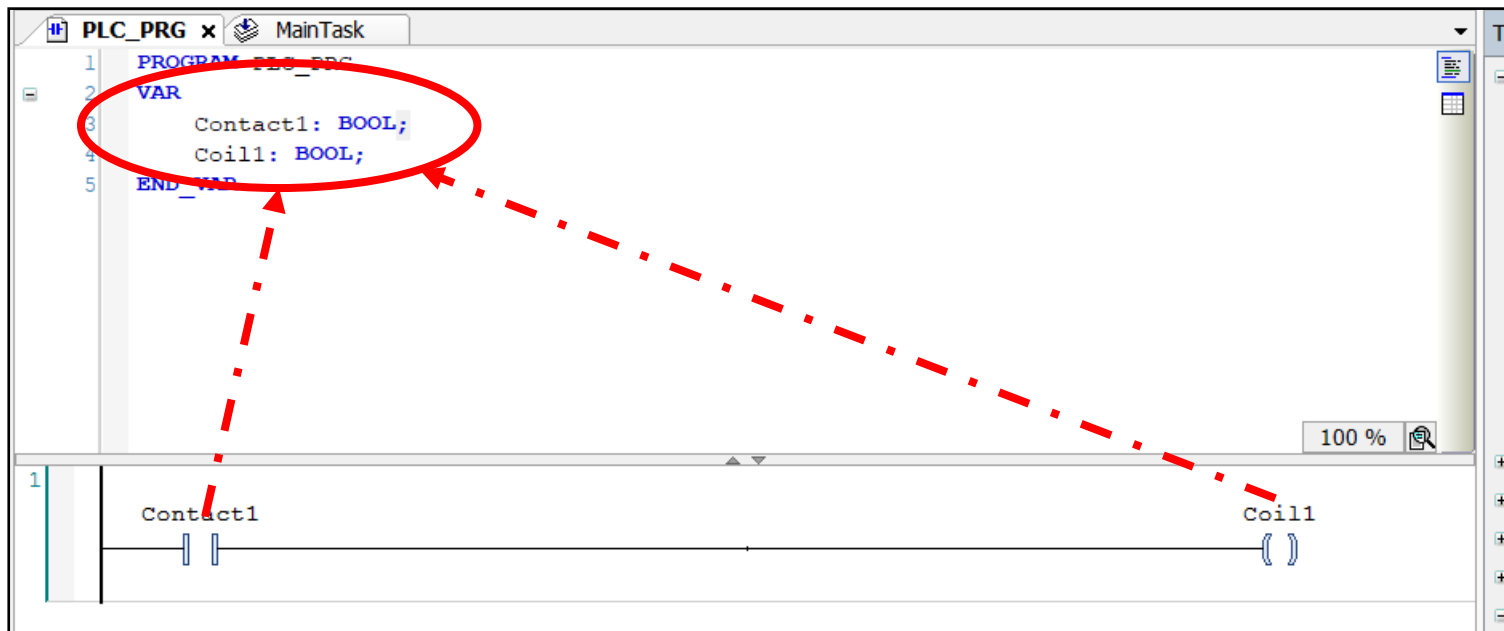
Project workbench - Ladder

- › 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

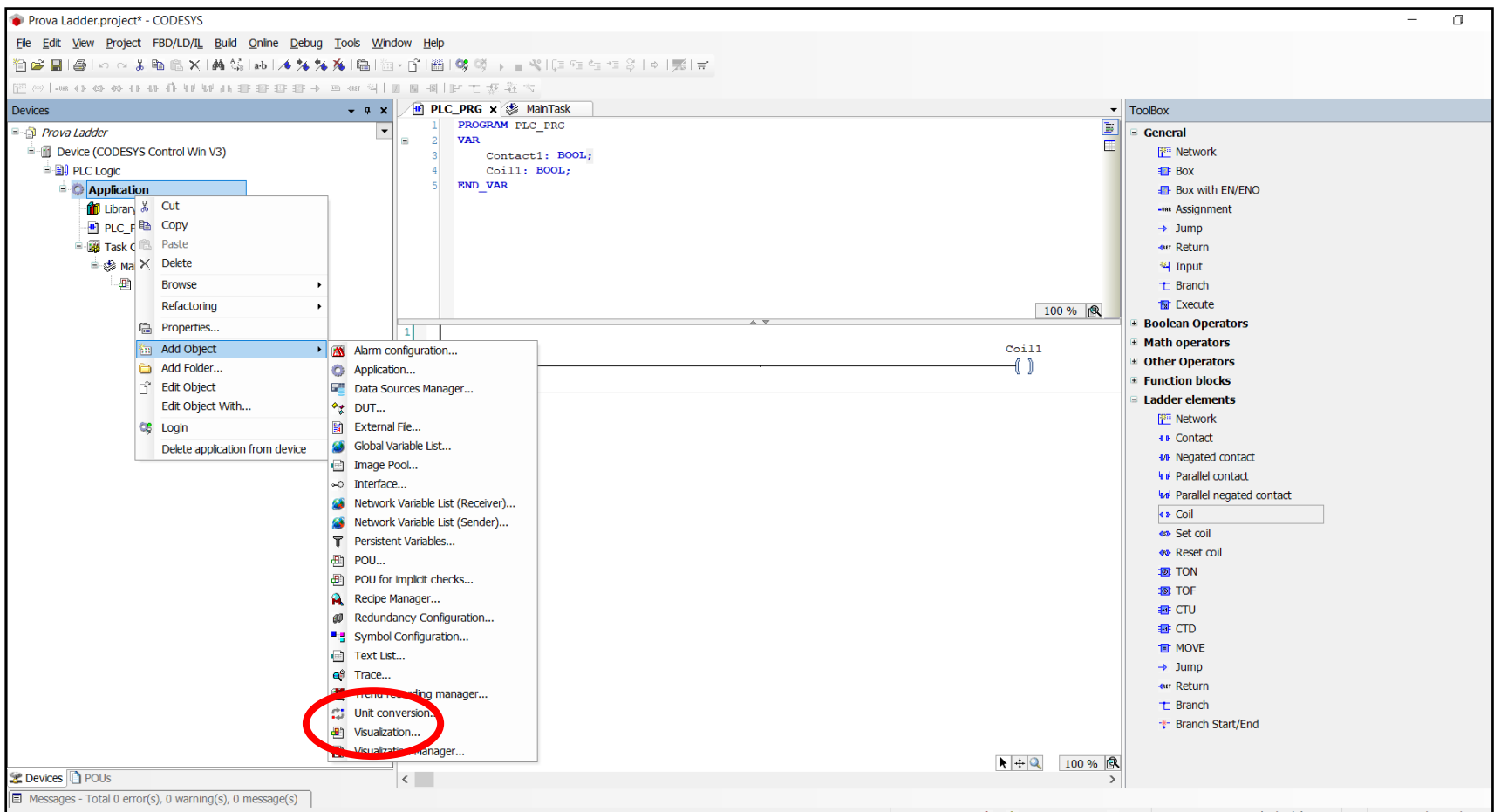




View the simulated system

Add a Visualization object

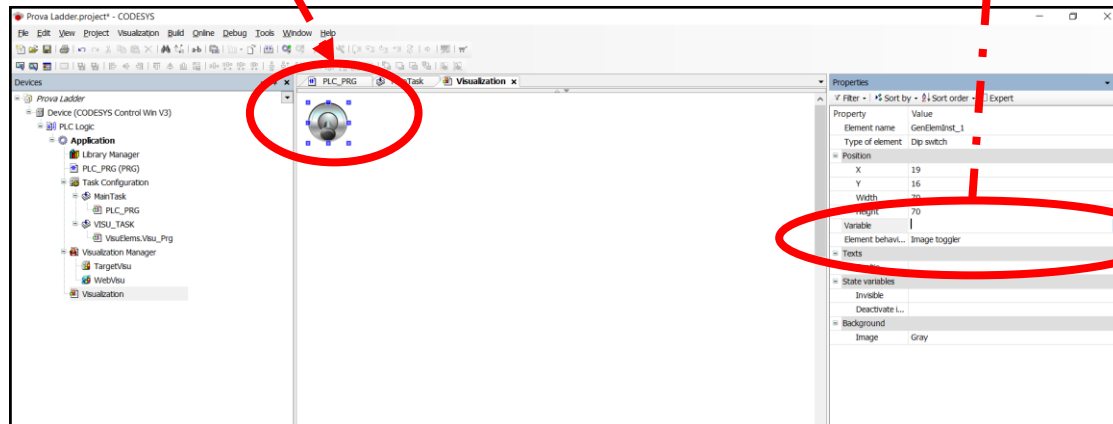
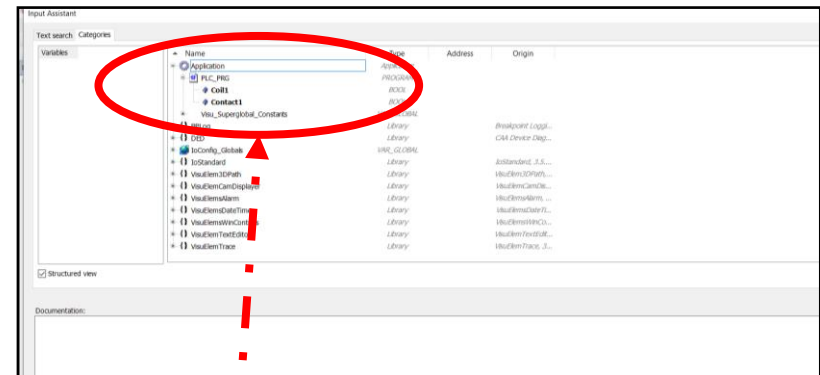
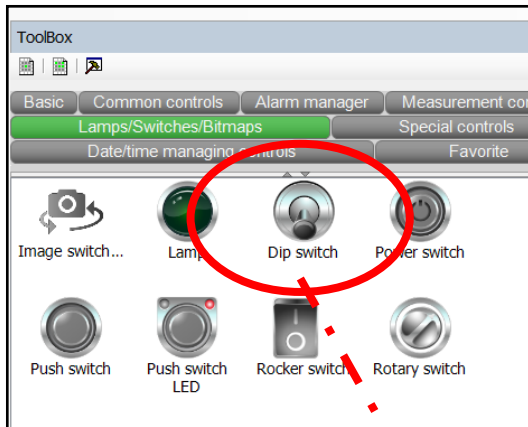
› Application -> Add Object -> Visualization





Add elements, and link to variables

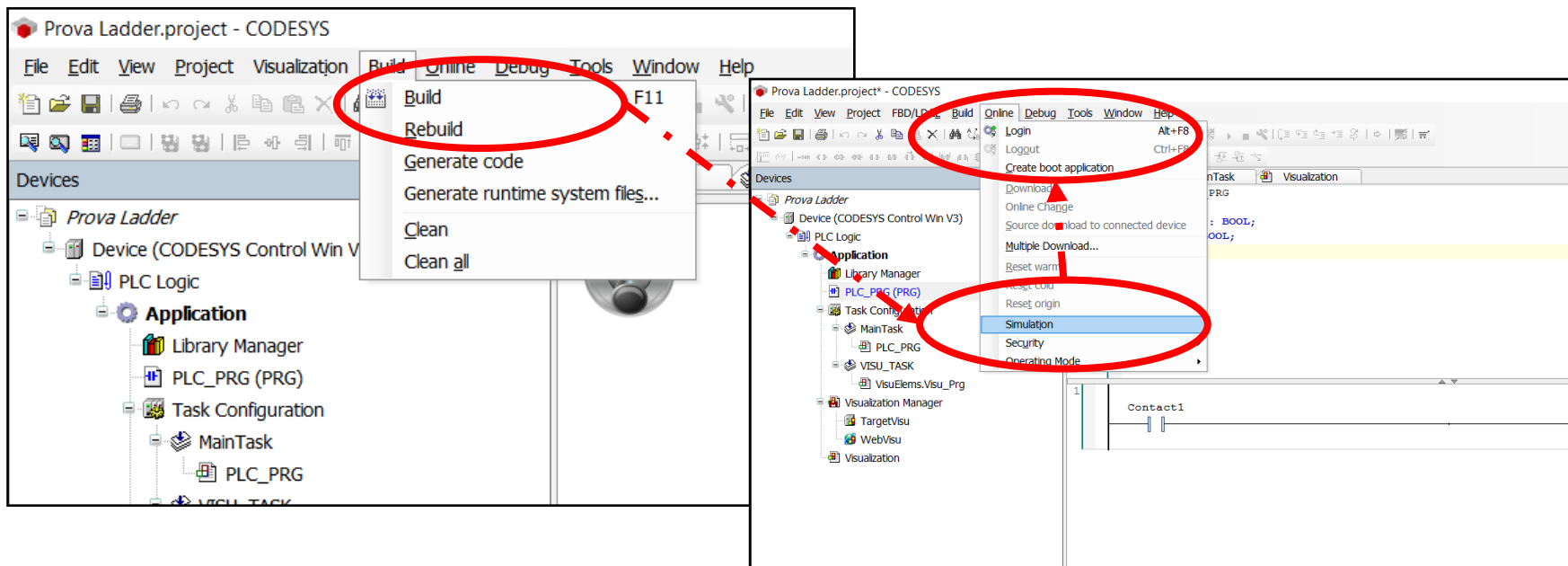
- › Here, we added a dip switch from the toolbox, and we select the `Contact1` var from the Properties window
- › Now, add a lamp and bind it to `Coil1`





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

Prova Ladder.project* - CODESYS

File Edit View Project Visualization Build Online Debug Tools Window Help

Devices

Prova Ladder

- Device [connected] (CODESYS Control Win V3)
 - PLC Logic
 - Application [stop]
 - Library Manager
 - PLC_PRG (PRG)
 - Task Configuration
 - MainTask
 - PLC_PRG
 - VISU_TASK
 - VisuElems.Visu_Prg
 - Visualization Manager
 - TargetVisu
 - WebVisu
 - Visualization

Device.Application.PLC_PRG

Expression	Type	Value	Pre
Contact1	BOOL	FALSE	
Coil1	BOOL	FALSE	

1

RET

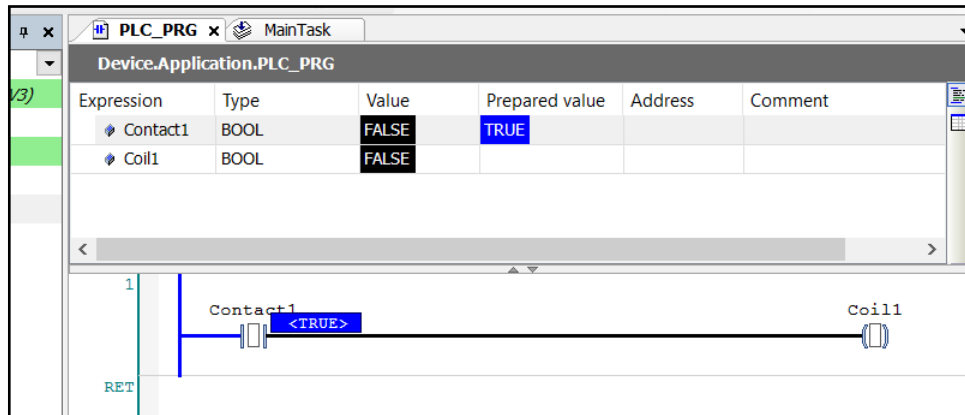
Contact1

Coil1

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)



- › 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"

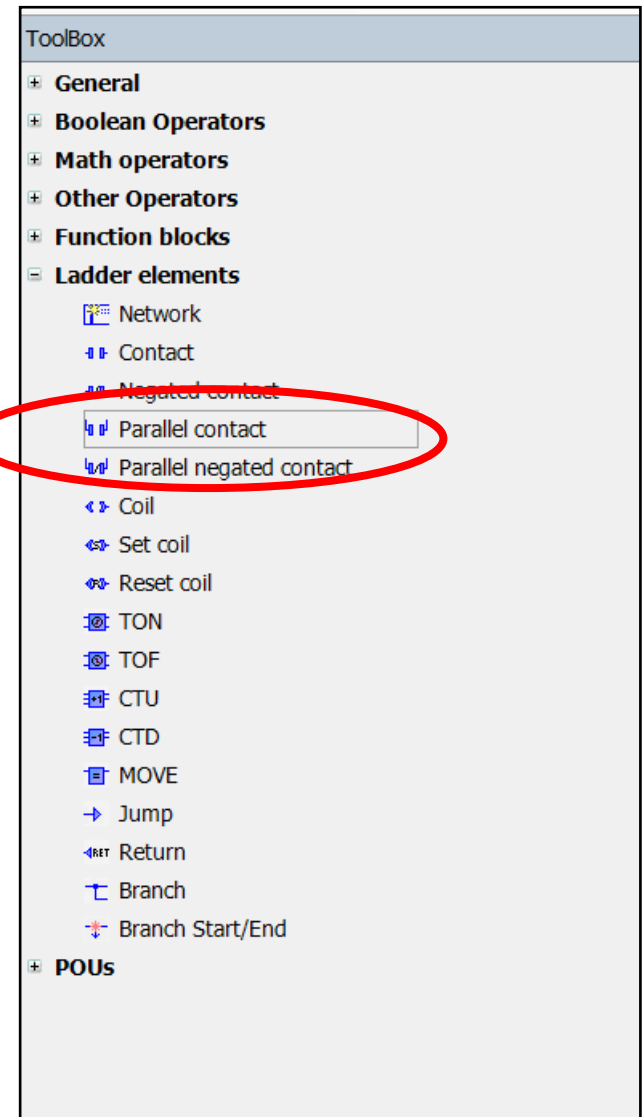
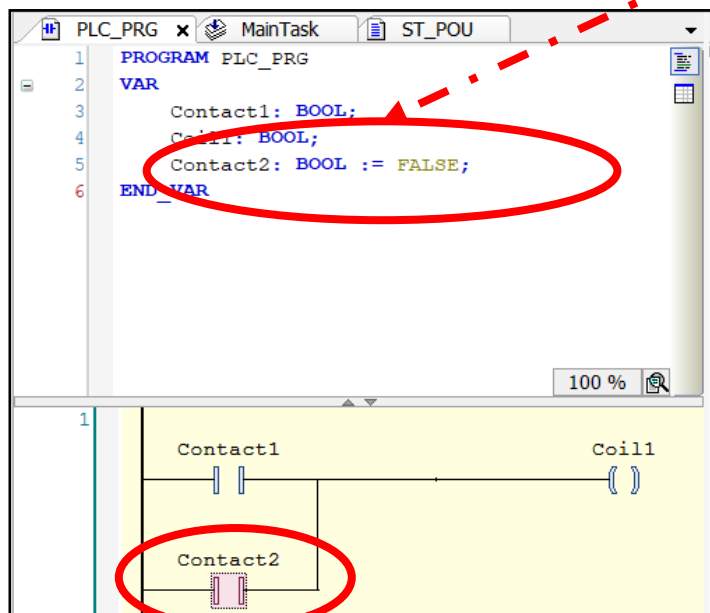
› ..easy, simply drag&drop


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

Prova Ladder.project* - CODESYS

File Edit View Project Build Online Debug Tools Window Help

Devices

- Prova Ladder
 - Device (CODESYS Control Win V3)
 - PLC Logic
 - Application
 - Library Manager
 - PLC_PRG (PRG)
 - ST_POU (PRG)
 - Task Configuration
 - MainTask
 - PLC_PRG
 - ST_POU
 - VISU_TASK
 - VisuElems.Visu_Prg
 - Visualization Manager
 - TargetVisu
 - WebVisu
 - Visualization

PLC_PRG MainTask ST_POU x

```
1 PROGRAM ST_POU
2 VAR
3     Contact1: BOOL;
4     Coil1: BOOL;
5 END_VAR
6
```

100 %

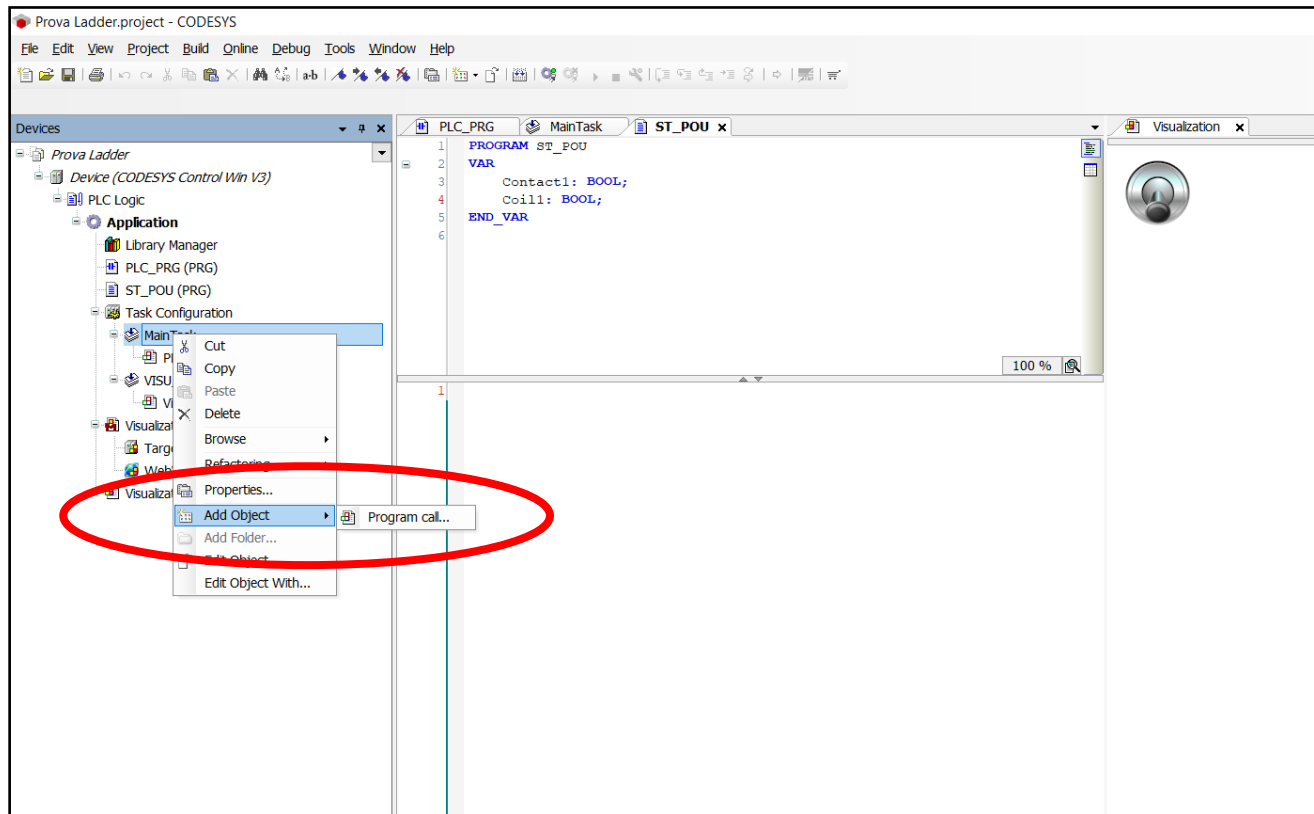
```
1 IF contact1 = TRUE THEN;
2     Coil1:=TRUE;
3 ELSE;
4     Coil1:=FALSE;
5 END_IF;
```

Visualization x



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?

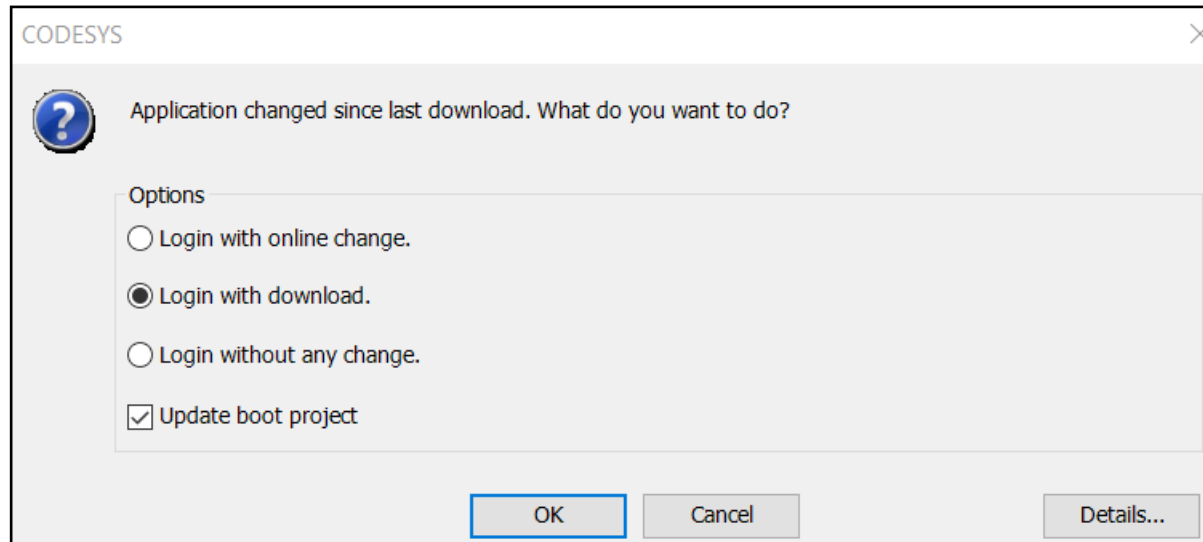
- › (recommendation) Only if requested by the application specs
- › In this case, I use them for debugging/teaching purposes, so my specs say "no" 😊





Compile & Login again

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

› Codesys will prompt us

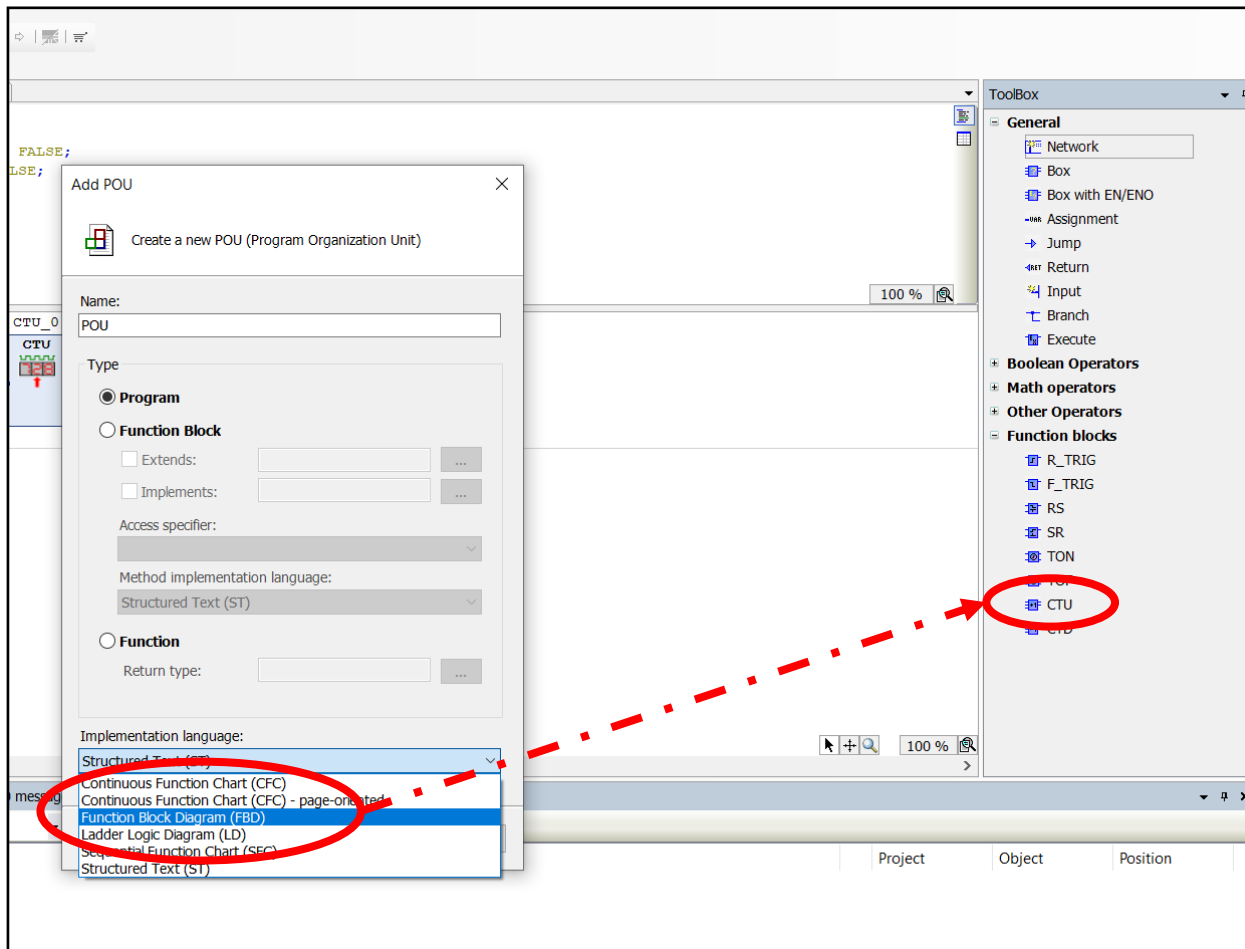




Programming with Function Blocks

Counters - CTU

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

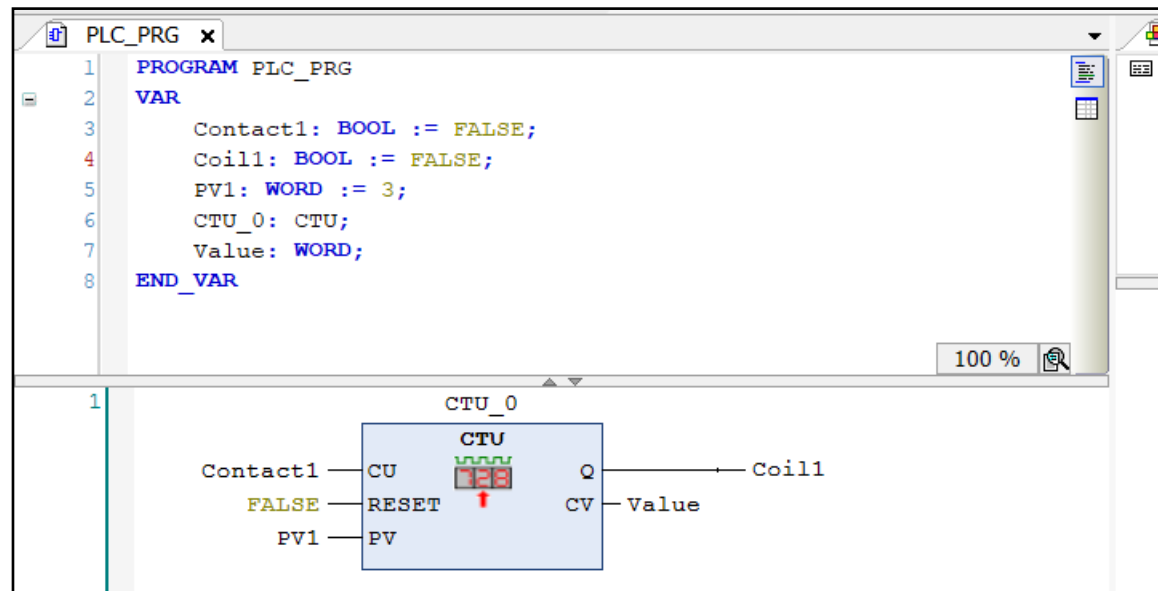




Bind the CTU

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`

The screenshot displays a PLC programming environment with three main windows:

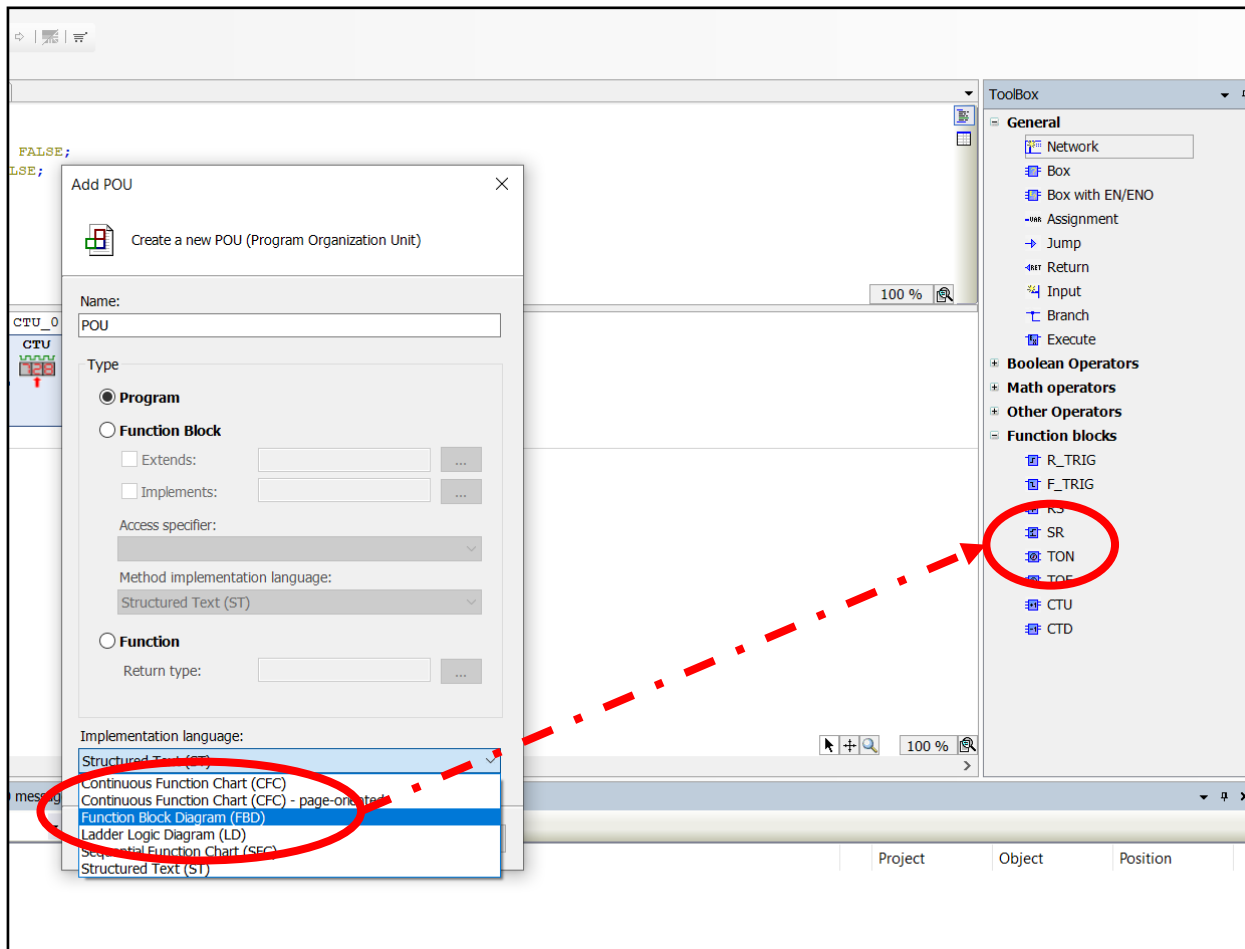
- PLC_PRG x**: Shows the ladder logic program. The first rung contains a CTU (Counter Up) block. The `CU` (Current Value) input is connected to `Contact1`, the `RESET` input is connected to `FALSE`, and the `PV` (Presets Value) input is connected to `PV1`. The `Q` (Output) is connected to `Coil1`, and the `CV` (Current Value) is connected to `Value`.
- Visualization x**: Shows the visualization editor. It contains a button labeled "Press me" and a lamp icon. The lamp icon is circled in red, and a red arrow points from it to the properties panel.
- Properties**: Shows the properties of the selected lamp element. The `Variable` property is set to `PLC_PRG.CTU_0.Q`, which is also circled in red.

The Properties panel details are as follows:

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

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

The screenshot displays a PLC programming environment with the following components:

- PLC_PRG x**: A text editor window showing the following code:

```
1 PROGRAM PLC_PRG
2 VAR
3   Contact1: BOOL := FALSE;
4   Coill: BOOL := FALSE;
5   PT1: TIME;
6   Value: TIME;
7   TON_0: TON;
8 END_VAR
```
- Visualization x**: A window showing a ladder logic diagram. The first network contains a normally open contact labeled 'Contact1' connected to a timer coil labeled 'TON_0'. The timer coil has a preset time 'PT1' and a value 'VALUE'. A red circle highlights the 'TON_0' coil. A red dashed arrow points from this coil to the 'Assignment' option in the toolbox.
- ToolBox**: A sidebar on the right containing various function blocks. The 'Assignment' option is circled in red.



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) ▾

Add Cancel



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

```
1 FUNCTION_BLOCK MyFB
2 VAR_INPUT
3     num1: REAL;
4     num2: REAL;
5 END_VAR
6 VAR_OUTPUT
7     AddResult: REAL;
8     SubResult: REAL;
9 END_VAR
10 VAR
11 END_VAR
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



Use it in the Application

In the main POU, create vars

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

```
1  PROGRAM PLC_PRG
2  VAR
3      A: REAL := 11;
4      B: REAL := 5;
5      Sum: REAL;
6      Subtr: REAL;
7      SumAndSubtract: MyFB;
8  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?

RG)	8	END_VAR
ration		
RG	1	SumAndSubtract(num1 := A, num2 := B);
	2	
	3	Sum := SumAndSubtract.AddResult;
	4	Subtr := SumAndSubtract.SubResult;





Simulate..and enjoy! 😊

MyFB PLC_PRG x

Device.Application.PLC_PRG

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

```
1 SumAndSubtract (num1 11 := A 11, num2 5 := B 5);  
2  
3 Sum 16 := SumAndSubtract.AddResult 16;  
4 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!

The screenshot shows a PLC programming environment with a POU (Program Organizational Unit) being edited. The main window displays a ladder logic network with a function block call to 'MyFB'. The function block has two inputs labeled 'num1' and 'num2', both with red wavy lines indicating they are not yet defined. It has two outputs labeled 'AddResult' and 'SubResult', both with red wavy lines. A red oval highlights the function block call. A dashed red arrow points from the 'MyFB' entry in the 'ToolBox' on the right to the function block call. The 'ToolBox' on the right shows a hierarchy of components: General, Boolean Operators, Math operators, Other Operators, Function blocks, and POU. The 'POU' section is expanded, showing 'PLC_PRG' and 'MyFB'. The 'MyFB' entry is circled in red.

```
1 PROGRAM POU
2 VAR
3     SumAndSub: MyFB;
4 END_VAR
5
```

100 %

1

SumAndSub

MyFB

??? num1 AddResult

??? num2 SubResult ???

ToolBox

- General
 - Network
 - Box
 - Box with EN/ENO
 - VAR Assignment
 - Jump
 - Return
 - Input
 - Branch
 - Execute
- Boolean Operators
- Math operators
- Other Operators
- Function blocks
- POU
 - PLC_PRG
 - MyFB



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



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