ELEC-H-516
Programmable Logic Controllers
Basic usage of CoDeSys 3.5

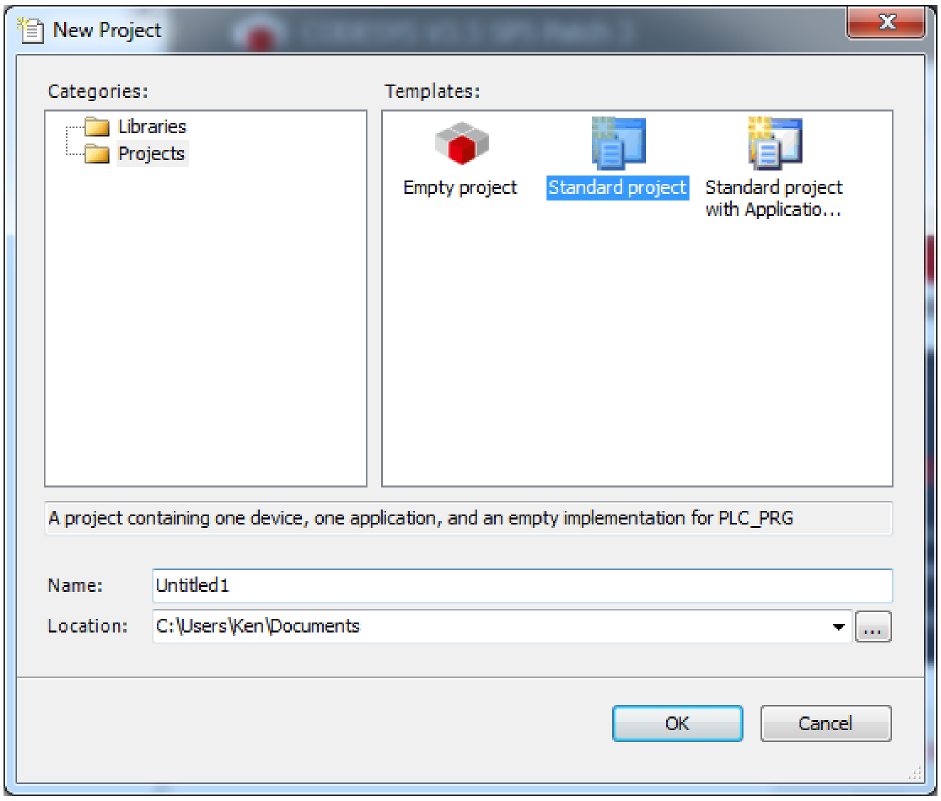
2014–2016

1 Introduction

This document will help you for the basic setup of CoDeSys 3.5 in order to start designing Ladder and/or grafcet properly. It will also present a basic example of simulation for your design.

2 Project setup

First create a new project, choose a standard project.



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

Name: Untitled1

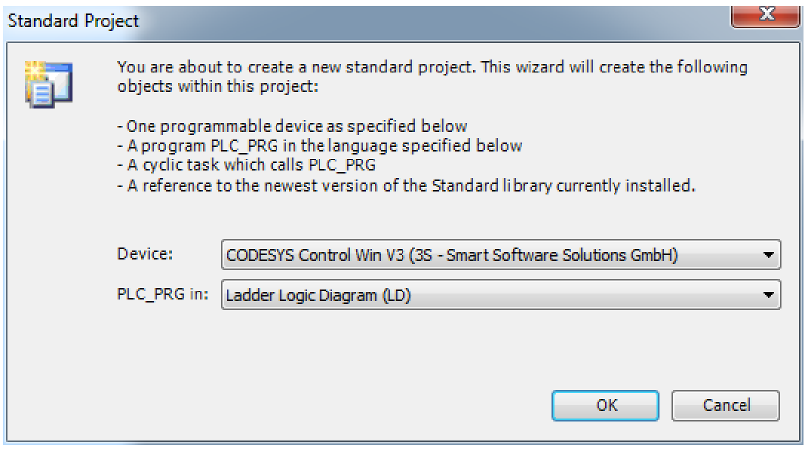
Location: C:\Users\Ken\Documents

Figure 1: Project creation

CoDeSys will then ask you to choose your device, since we are going to use simulation, we let the default value “Control Win V3". Meanwhile, we change the value of the language for PLC_PRG. CoDeSys supports all languages described by the standard IEC-61131:

- Textual languages:
  - Instruction List (IL)
  - Structured Text (ST)
- Graphical languages:
  - Sequential Function Chart (SFC) (Grafcet)
  - Function Block Diagram (FBD)

- The Continuous Function Chart Editor (CFC)
- Ladder Diagram (LD)



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

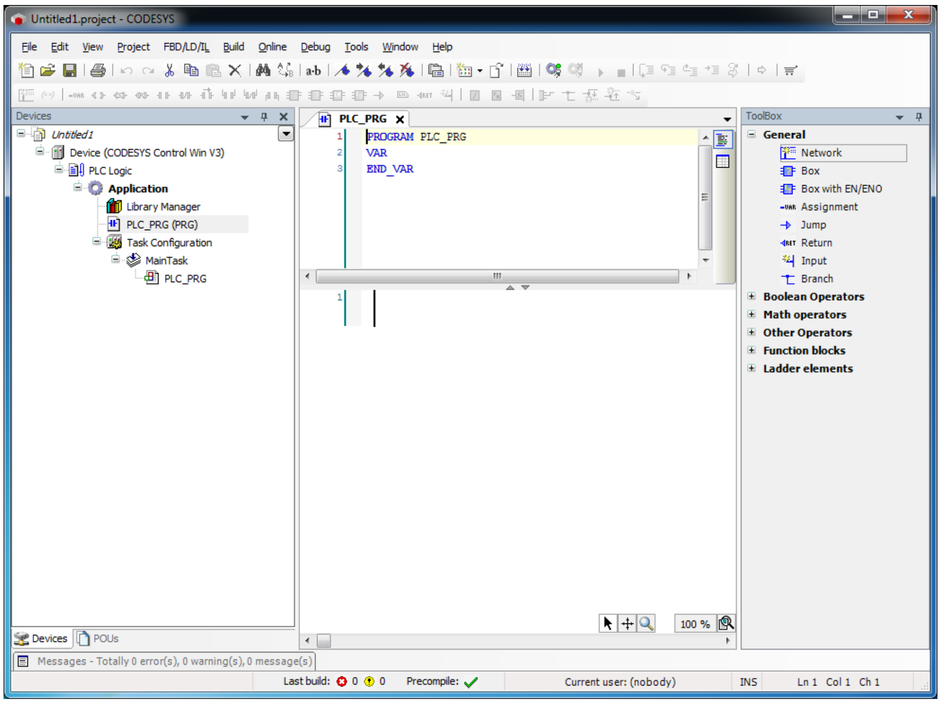
- One programmable device as specified below
- A program PLC_PRG in the language specified below
- A cyclic task which calls PLC_PRG
- A reference to the newest version of the Standard library currently installed.

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

PLC_PRG in: Ladder Logic Diagram (LD)

Figure 2: Language selection

Once it's done, click on the PLC_PRG file on the left, you will get a window similar to the one on figure 3 (we chose ladder for this example).



```
PROGRAM PLC_PRG
VAR
END_VAR
```

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

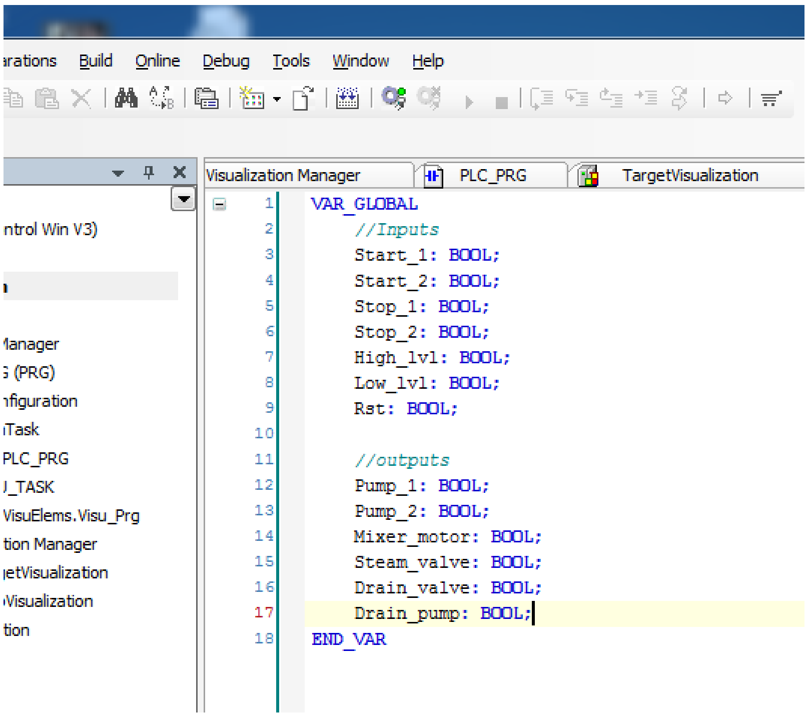
Last build: 0 0 Precompile: ✓ Current user: (nobody) INS Ln 1 Col 1 Ch 1

Figure 3: Main window

Select the `Application` on the left and right click on it or go to Project->Add object, and select `Visualization`. This will create the environment to run your simulation with visual objects like buttons or lights.

We will start by creating a Global Variable List to simulate the inputs and outputs of a real system.
Right click on `Application` then Add object->Global Variable List.

You can now enter your inputs and outputs, the figure 4 gives an example.



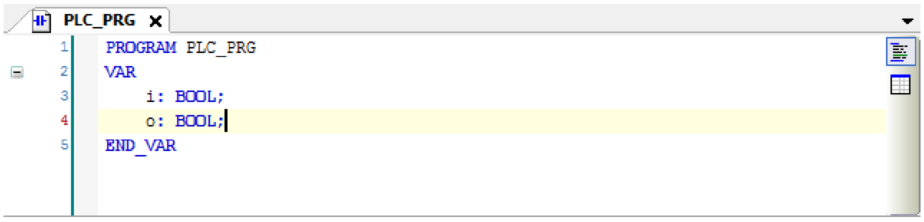
```
VAR_GLOBAL
    //Inputs
    Start_1: BOOL;
    Start_2: BOOL;
    Stop_1: BOOL;
    Stop_2: BOOL;
    High_lvl: BOOL;
    Low_lvl: BOOL;
    Rst: BOOL;

    //outputs
    Pump_1: BOOL;
    Pump_2: BOOL;
    Mixer_motor: BOOL;
    Steam_valve: BOOL;
    Drain_valve: BOOL;
    Drain_pump: BOOL;
END_VAR
```

Figure 4: Global variables (I/Os)

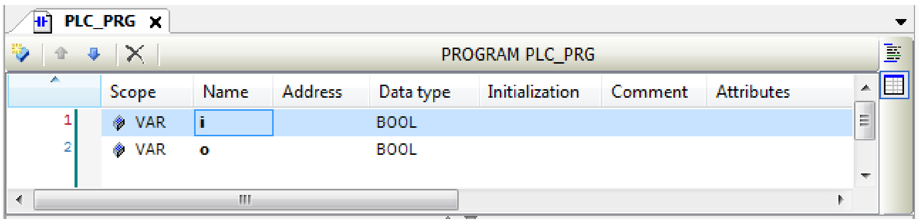
3 Variables

You have two interfaces to declare variables in CodeSys: textually (figure 5) or using a table (figure 6). You can switch from one to the other using the two small buttons and .



```
PROGRAM PLC_PRG
VAR
    i: BOOL;
    o: BOOL;
END_VAR
```

Figure 5: Declare variables as text.



PROGRAM PLC_PRG

	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	i		BOOL			
2	VAR	o		BOOL			

Figure 6: Declare variables in a table.

3.1 Syntax

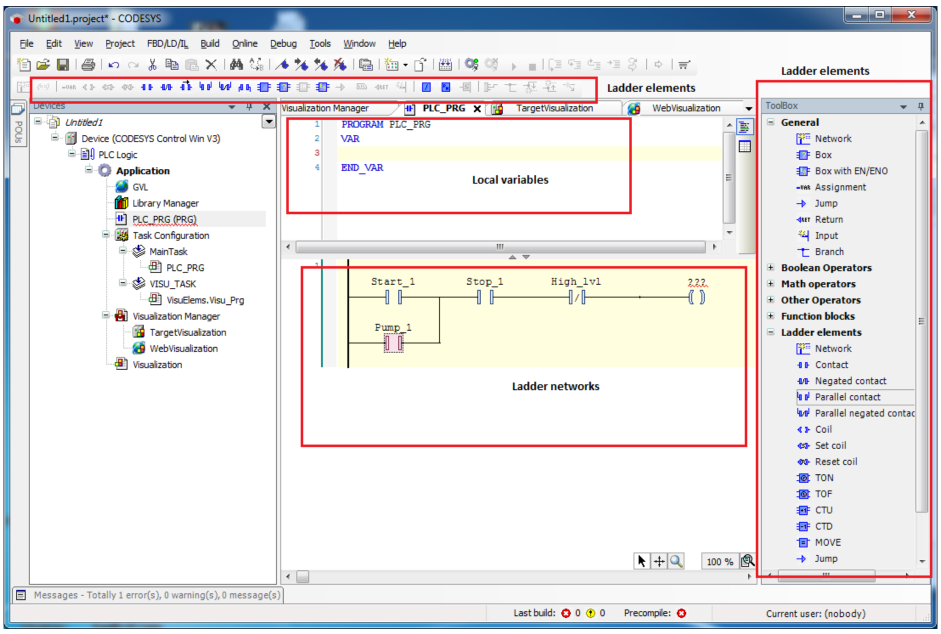
The syntax of the variables is such that their name can not contain the dot "." character. Past this detail, the grammar is as follows: `variable_name: Type := value;`, the `value` part being facultative.

3.2 Types

See “Data type” in the doc.

4 Ladder programming

Elements of ladder programming are available in the right panel and on the top bar, as the figure 7 highlights them. Local variables can be used to add intermediate variables that are not inputs or outputs (they are only usable in the current POU (program organization unit). Here, the POU is the PLC_PRG file in ladder laguage.



```
PROGRAM PLC_PRG
VAR

END_VAR
```

Figure 7: Network

You can select variables to assign to the different elements of your network by clicking on the “...” icon next to the name of the element (which is “???” by default). Then simply assign the variables by selecting them on figure 8.

Edge detection Add a contact to your network, then right-click it and change it to “Edge Detection” to get a rising edge detector. Repeat the operation to have a falling edge detector.

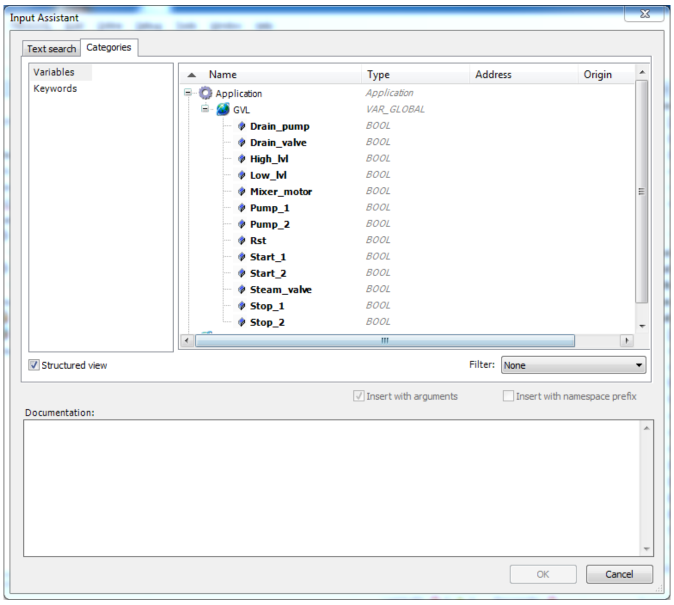
Parallel branch In order to have a parallel branch that comes back up to the same output (like `Start_1` and `Pump_1` on fig 7), you first need an element on the network. Right-click it and choose Insert Contact Parallel.

Variable assignment Ladder diagrams are fundamentally boolean structures, handling boolean objects. When you want to assign a value to a non-boolean variable (*e.g.* an INT), you need to use an extra block allowing you to integrate a different paradigm into LD.

This block is the “Execute” block that you can find in the “General” submenu of the ToolBox. Inside of it, you can type ST instructions, such as “a := 1;” to assign a value to the variable `a`.

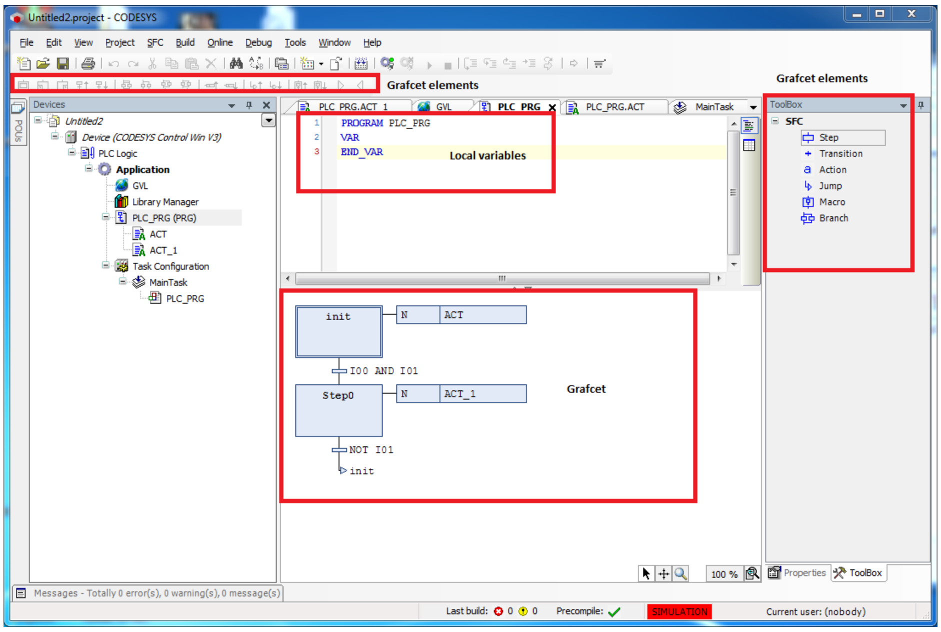
5 Grafcet programming

Elements of grafcet programming are available in the right panel as well as on the top bar. Just like for ladder programming, local variables can be used to add intermediate variables that are not inputs or outputs (they are only usable in the current POU (program organization unit)). Here, the POU is the PLC_PRG file in grafcet language.



Name	Type	Address	Origin
Application	Application		
GVL	VAR_GLOBAL		
Drain_pump	BOOL		
Drain_valve	BOOL		
High_lvl	BOOL		
Low_lvl	BOOL		
Mixer_motor	BOOL		
Pump_1	BOOL		
Pump_2	BOOL		
Rst	BOOL		
Start_1	BOOL		
Start_2	BOOL		
Steam_valve	BOOL		
Stop_1	BOOL		
Stop_2	BOOL		

Figure 8: Selection

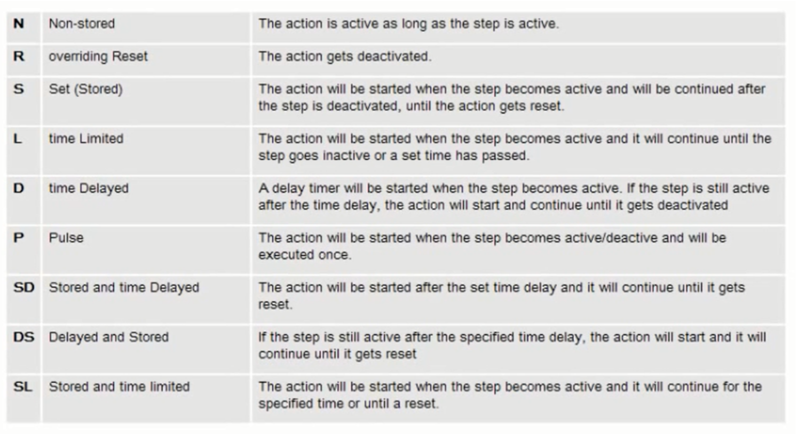


```
PROGRAM PLC_PRG
VAR
END_VAR
```

Figure 9: Grafcet

Once you've added some grafcet elements, you can create transition and actions to attach on it. There are two ways to add actions or transitions on a grafcet network : direct or indirect.
Direct adding means you write structured text logic expressions directly in the boxes (Transitions on figure 9 on the following page).
Indirect adding means you have to create a transition or action file (right click on PLC_PRG->Add object) in any format you want (ladder, structured text...) and detail your expressions for transitions or actions in this file.
For actions, a letter lets you choose whether you want you action the be delayed, time limited, and so on (the default choice is "N" : the action is active as long as the step is active). See figure 10 on the next page for details on the different possibilities.

Note that for the time-dependent actions, CodeSys is expecting a `TIME` variable next to the identifier.



N	Non-stored	The action is active as long as the step is active.
R	overriding Reset	The action gets deactivated.
S	Set (Stored)	The action will be started when the step becomes active and will be continued after the step is deactivated, until the action gets reset.
L	time Limited	The action will be started when the step becomes active and it will continue until the step goes inactive or a set time has passed.
D	time Delayed	A delay timer will be started when the step becomes active. If the step is still active after the time delay, the action will start and continue until it gets deactivated
P	Pulse	The action will be started when the step becomes active/deactive and will be executed once.
SD	Stored and time Delayed	The action will be started after the set time delay and it will continue until it gets reset.
DS	Delayed and Stored	If the step is still active after the specified time delay, the action will start and it will continue until it gets reset
SL	Stored and time limited	The action will be started when the step becomes active and it will continue for the specified time or until a reset.

Figure 10: Action modes (“Qualifiers for Actions in SFC”)

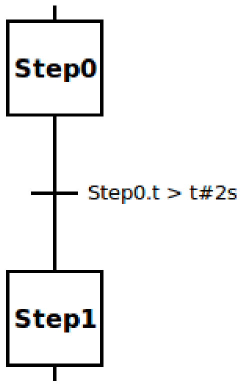


Figure 11: The transition will be fired after 2 seconds have been spent in `Step0`.

Selection of variables is similar to the selection in ladder programming.

5.1 Transition

Several conditions can be concatenated in a single transition expression using `AND` and `OR`.

Timer A transition can be fired according to the time spent in the previous step (see fig 11). To fire the transition after two seconds have been spent in the step `step0`, you would write:

```
step0.t > t#2s
```

5.2 Convergence and divergence

Convergences and divergences are generated using the “branch” tool. To create an AND-divergence, select a step you want to be part of a branch, and create a left or right branch (either *via* right-click or the toolbar). To create an OR-divergence, do the same thing, but selecting a transition.

5.3 Useful modules

Several modules can be quite useful:

CTU Counter[1]

TON Timer

[1] A `WORD` is 16-bit long.

Note that to easily obtain the documentation to a specific module, right-click on it → “browse” → “Go to definition”.

5.4 Troubleshooting

Make sure not to use the same name for the step and for the action attached to it.

6 Simulation and visualization

6.1 Setup

Select simulation from the dropdown menu “Online” so that it ends up checked.

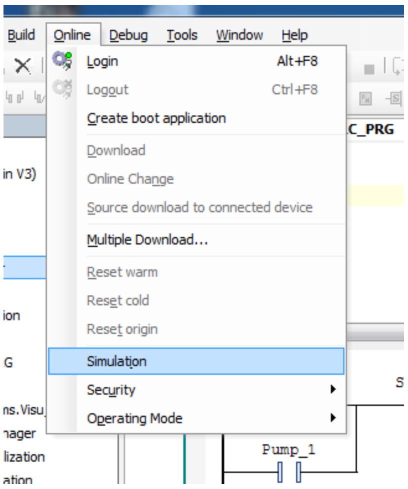


Figure 12: Selection

Then, from the same menu, select login. You are now in simulation mode (fig 13). There it is possible to start or stop the simulation by clicking on the “play” or “stop” button of the top bar. It is also possible to assign (or force) values to variables by simply clicking several times on the variable. When your input variables have the wanted states go to Debug->Write (or Force) Value to assign (or force) the values to the system.
The network should response according to your inputs.

6.2 Visualization

Double click on the Visualization you created during the setup of the project, you should get to the vizualization window.
NB : The project needs to be “Offline” to allow the modification of the visualization (Online->Logout in the top bar).
There you can add objects from the toolbox that can be linked to the different variables you created in your project (fig 15)

The visualization can then be used in sync with the simulation of the system by setting the project back online (Online->login menu in the top bar)

6.3 Troubleshooting

- Should the simulation throw a “License not found” error, it’s probably because of the visualization. Try removing the “visualization manager” from the project tree, keeping only the “visualization”.
- You can import a project from an older version of CodeSys to a newer one, but you will not be able to simulate it. If you want to simulate it, you need to create a new project, not only a new diagram.

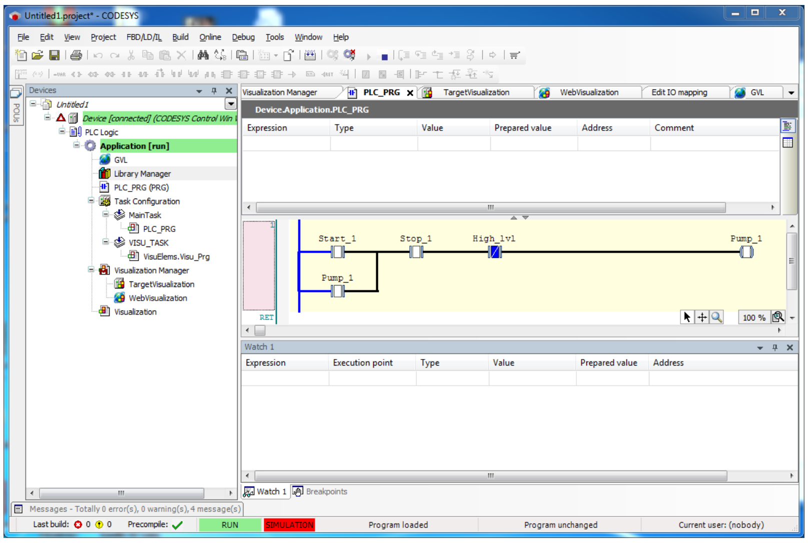


Figure 13: Simulation mode

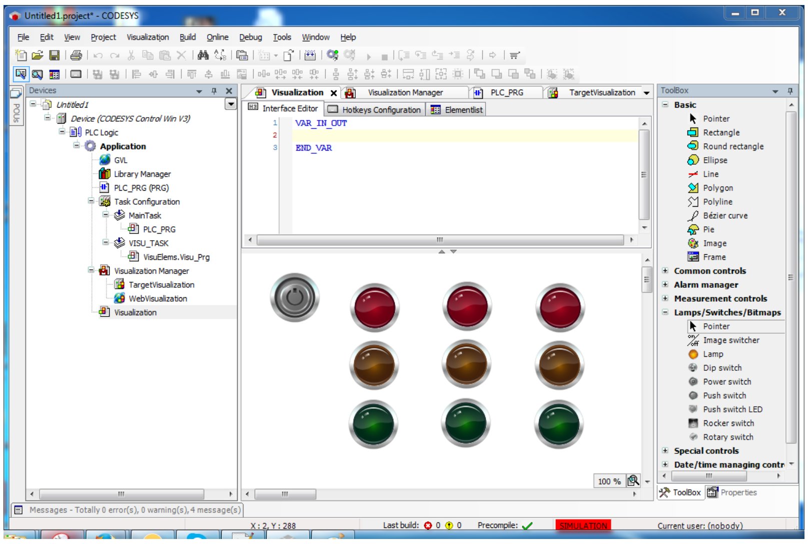


Figure 14: Visualization

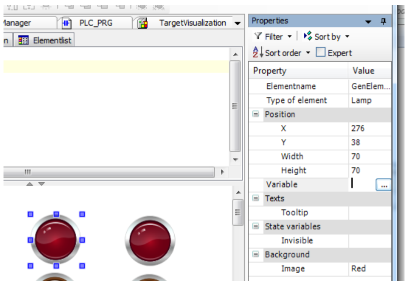


Figure 15: Properties