

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.

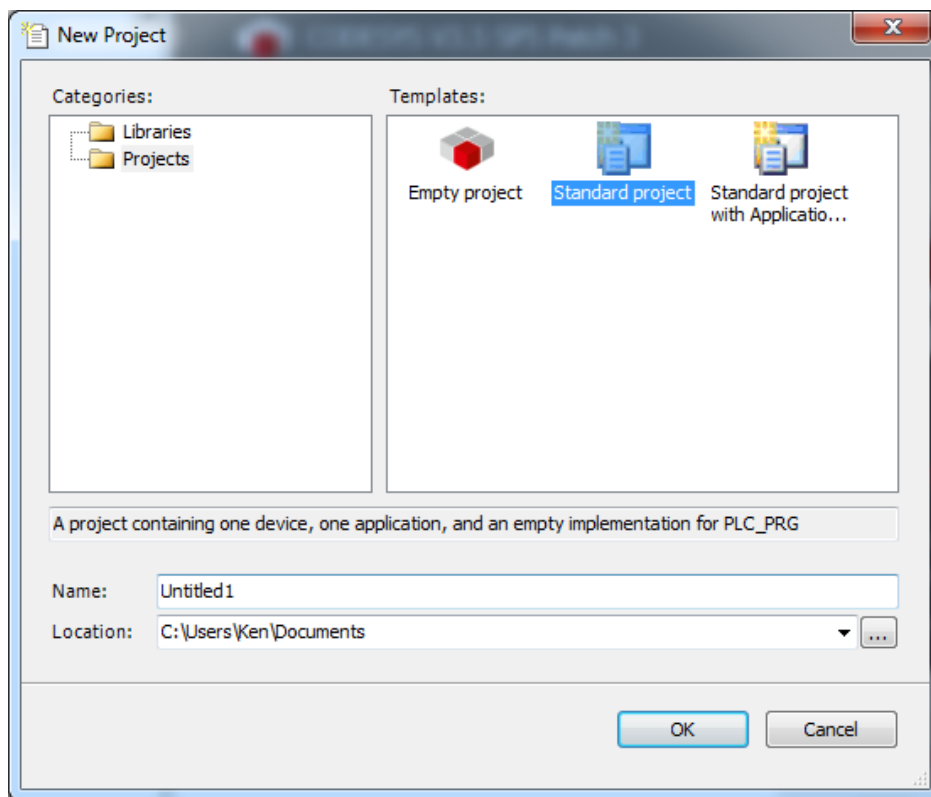


Figure 1: Project creation

CoDeSys will then ask you to choose your device, since we are going to use simulation, we let the default value of “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)
- Graphic Languages:
 - Sequential Function Chart (SFC) (Grafcet)
 - Function Block Diagram (FBD)

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

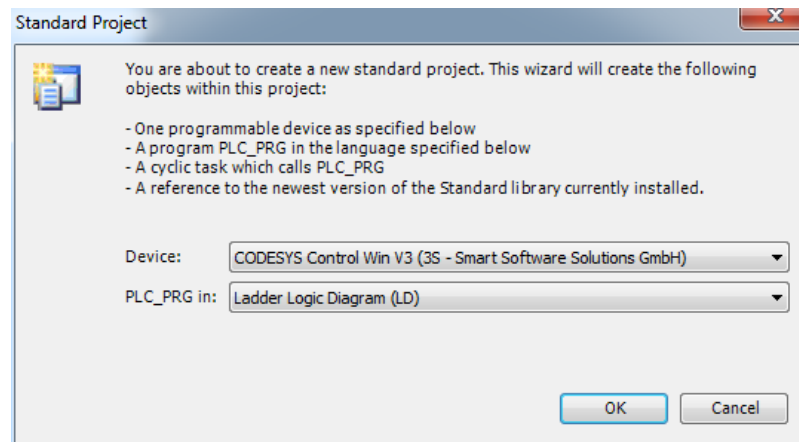


Figure 2: Language selection

Once done, click on the PLC_PRG file on the left, you will get a window similar to the one below I chose ladder for this example).

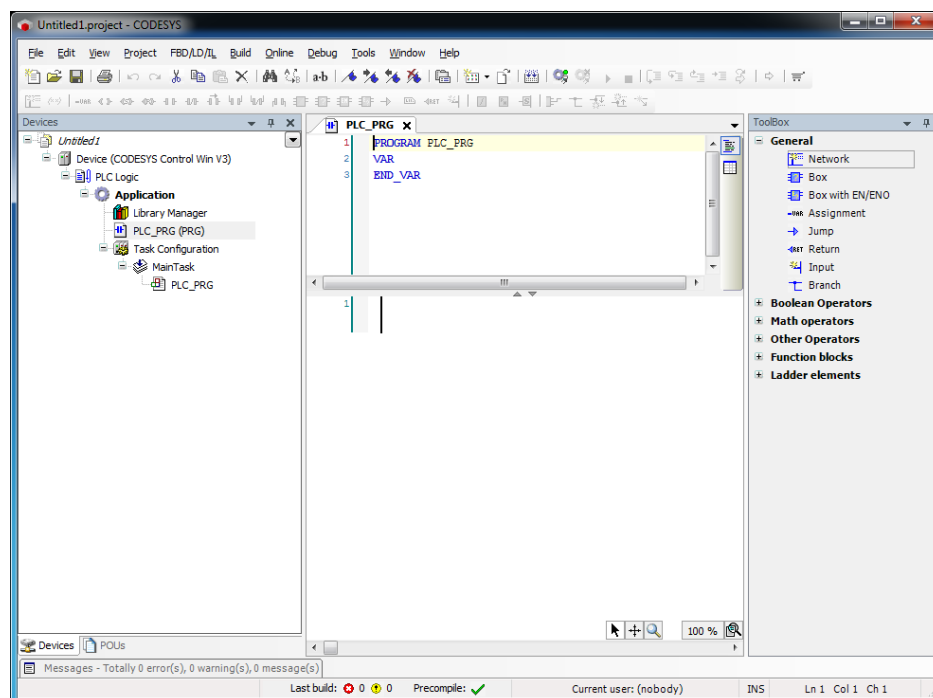


Figure 3: Main window

Select the Application 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 (as we don't actually have a PLC). Right click on Application then Add object->Global Variable List.

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

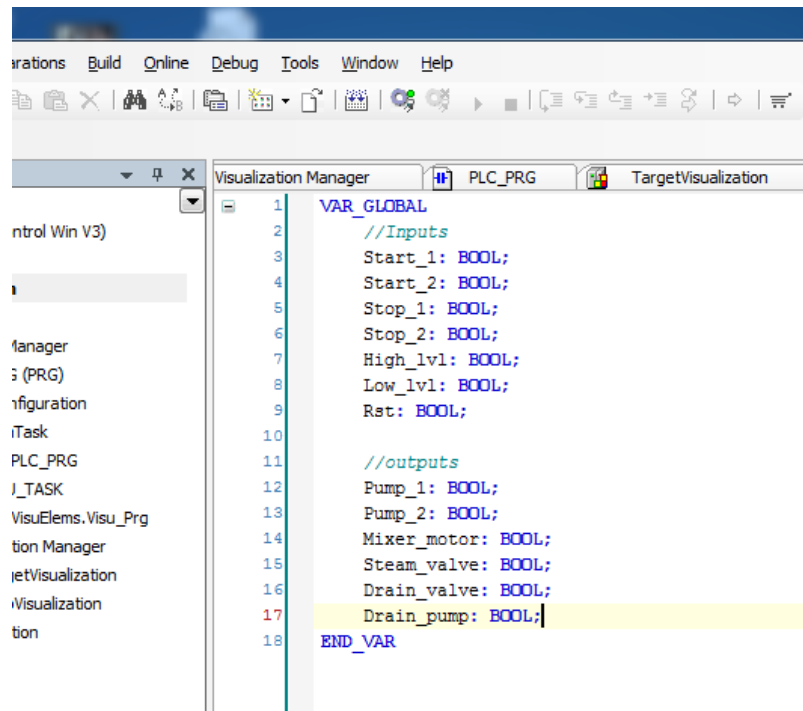


Figure 4: Global variables (I/Os)

3 Variables

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 for ladder programming are available in the right panel and on the top bar. 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 language.

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. The same goes with local variables, both are available in this menu.

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 (see fig ??), you first need an element on the network. Right-click it and choose Insert Contact Parallel.

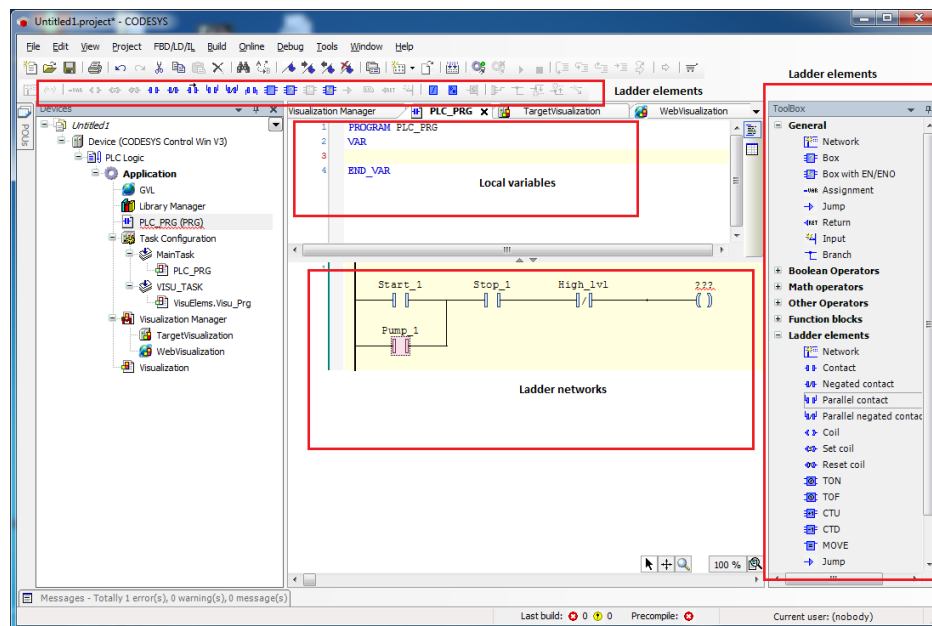


Figure 5: Network

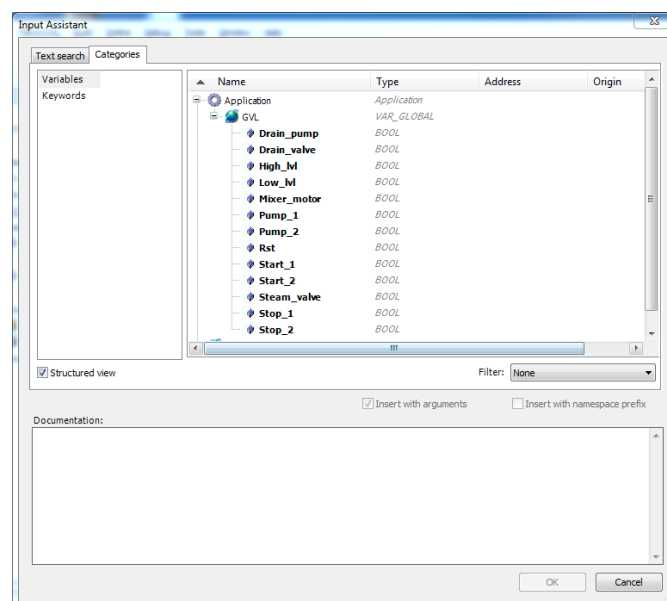


Figure 6: Selection

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.

Once you've added some grafcet elements, you can create transition and actions to attach on it. There are two ways of adding actions of transitions on a grafcet network : direct or indirect.

Direct adding means you write structured text logic expressions directly in the boxes (Transitions on 7 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

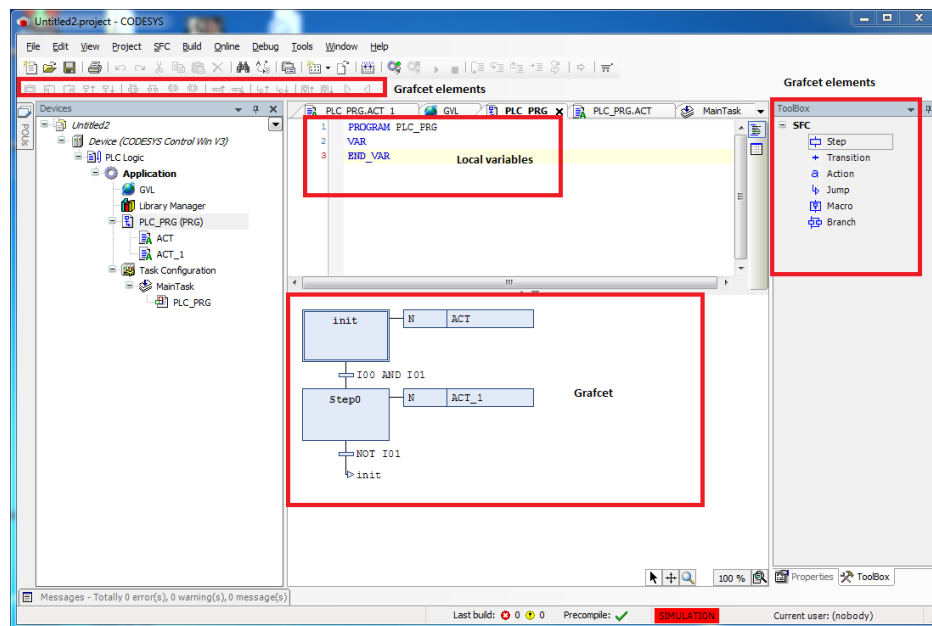


Figure 7: Grafcet

transitions or actions in this file.

For actions, a letter lets you choose whether you want your action to be delayed, time limited, etc... (the default choice is "N" : the action is active as long as the step is active). See figure 8 for details on the different possibilities.

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 8: Action modes ("Qualifiers for Actions in SFC")

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

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 9). To fire the transition after two seconds have been spent in the step **step0**, you would write:

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

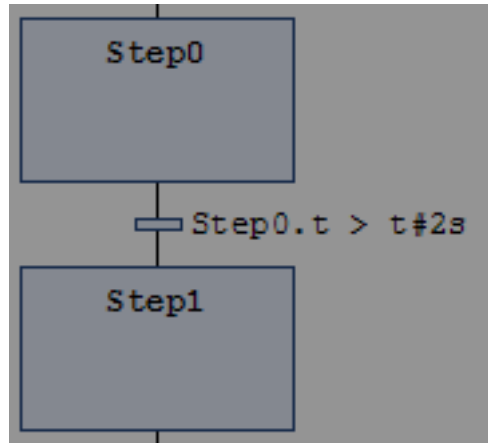


Figure 9: Using a timer on a transition

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 Counter¹

TON Timer

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

6 Simulation and visualization

6.1 Setup

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

Then, from the same menu, select login. You are now in simulation mode (fig 11 on the following page) 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 windows.

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 13 on page 8)

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

¹A WORD is 16-bit long.

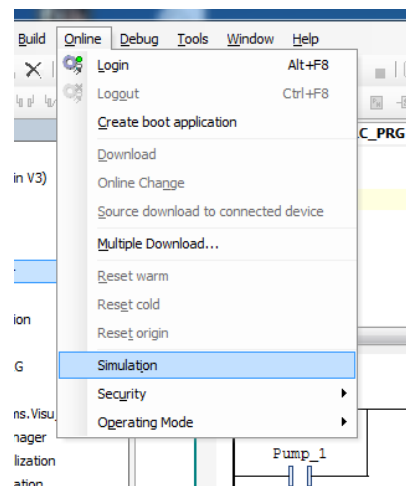


Figure 10: Selection

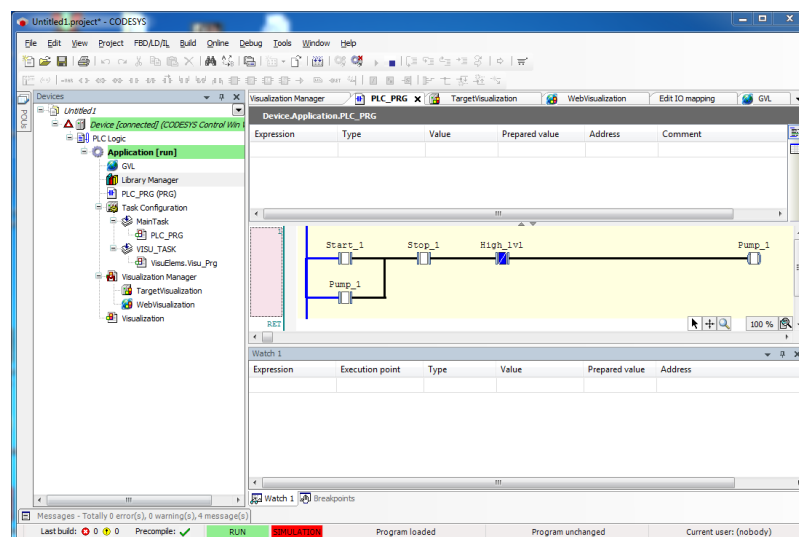


Figure 11: Simulation mode

6.2.1 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".

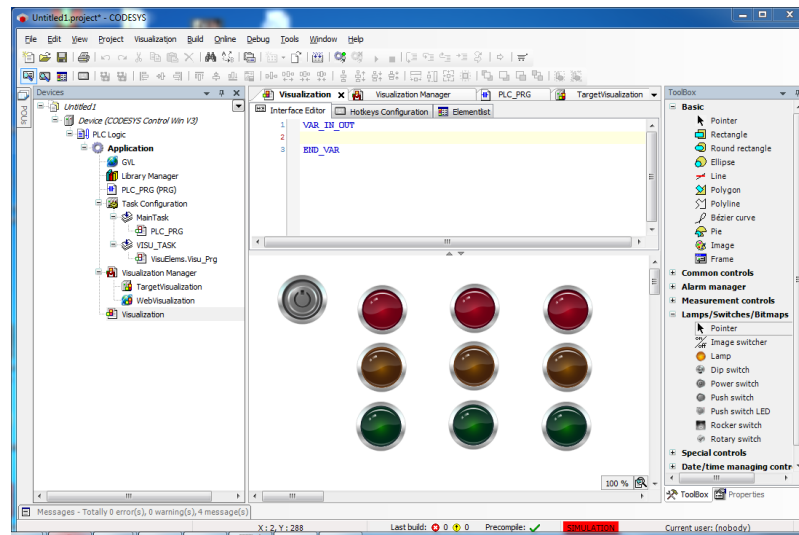


Figure 12: Visualization

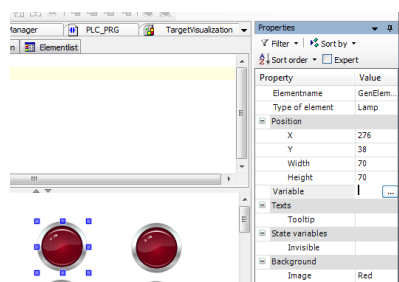


Figure 13: Properties