Lab 1 - Tank Management

GreEn-ER Industrial Control systems Sandbox (G-ICS)
2022 - Wago - Asean-Factori

Table of contents

Introduction

Modern complex Industrial Control System (ICS), which includes SCADA and Distributed Control Systems (DCS), heavily relies on the use of a communication system tailored for the real-time and reliability requests of distributed applications.

In contrast with general communication systems (like Ethernet/TCP/IP networks and other Internet related technologies) industrial communication systems are part of the controller and are able to guarantee that real-time constraints of the application will be met. Despite the fact that some of the protocols are Ethernet embedded (like GOOSE and ProfinetI/O) or even TCP/IP transported (like ModbusTCP and S7) the deployment of the TCP/IP stack requires some modifications in the Internet protocols (like sockets reuse, never-closing TCP connection, Ethernet Vlan frames tagging and priority, etc) that brings the TCP/IP stack close to a deterministic behavior.

Therefore, for a control engineer, the practical knowledge of industrial communication protocols like behavior of the data flows, relationship between PLC programs and network connections, bandwidth use and optimization is an important requirement in order to be able to deploy an industrial control system. The purpose of these labs is to let you acquire a minimal knowledge in the industrial communication field.

During this lab, you will learn how to program a simple application using *Grafcet*, *Ladder*, and *Structured Text*, create a first HMI and simulate the process. We will do simulations because it is the best way to begin with e!Cockpit remotely. Indeed, the difference between the simulation interface and the "*connected to the PLC*" interface is really small. This training is accompanied by detailed videos about each part of the lab. A lot of details about the software are explained in the lab 0. If you are lost with something, try to find the solution in the last lab or feel free to ask the teacher.

Work to do

During this lab, you will work on an example of the management of a liquid tank. You will have to :

- Configure the software
- Declare the variables
- Write a control program for the PLC
- Create an HMI to visualize the process
- Check the functioning with the simulation

Control specifications

The control specification concerns a system based on a tank containing a liquid. There is a sensor for the maximum level (Max_Level), and another for the minimum level (Min_Level). There is also an input button to ask for liquid (Client_Request). Finally, there are two valves to control the liquid distribution (V1) and the tank refill (V2).

To open V1 and distribute the liquid, the level must be higher than Min_Level. To open V2 and refill the tank, the water level must be lower than Max_Level. You can see a diagram of the process in the figure 1 just below.

To simplify the realization of the program, we suppose that at the beginning, the level of liquid is between Min_Level and Max_Level.

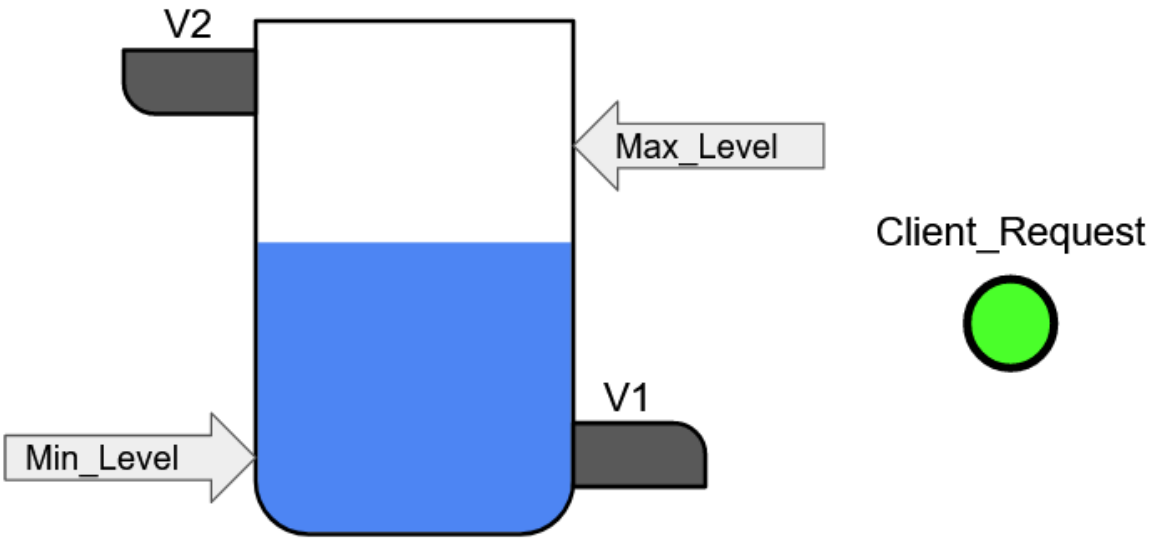


Figure 1 : Diagram of the process

Configuration

PLC Configuration

Beginning a new project

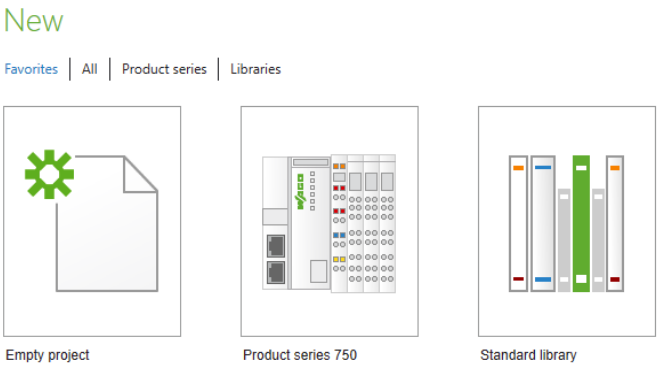


Figure 2 : The project beginning screen

When you launch *e!Cockpit*, you can see the three selection boxes like on the figure 2. In our case, click on "*Empty project*".

Manual Configuration

Firstly, you have to add a controller (a PLC) to your project. In these labs, we will use the ***750-8212*** controller. We chose this one because it is one of the PLC that we use in our laboratory in Grenoble. To do that, type the reference in the research bar to the right, in "*product catalog*". You can also search manually into the tree structure.

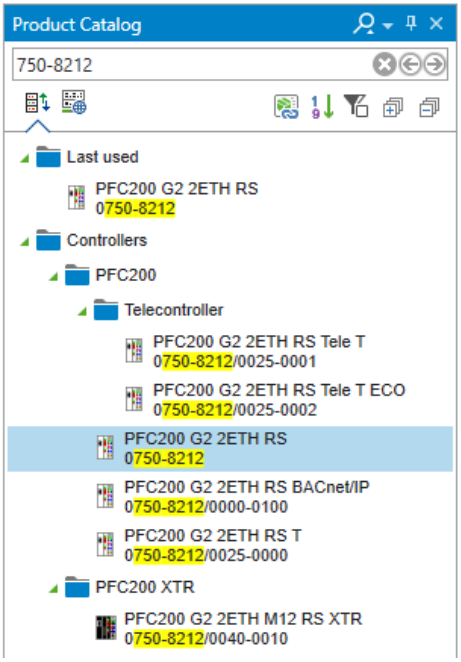


Figure 3 : Product catalog

You can now add the device to the project by drag and drop it in the "*Network/Devices*" window. Double-click on the controller to open a more detailed view (figure 4 below). You can see your PLC represented like the real one, and the end module (reference *750-600*).

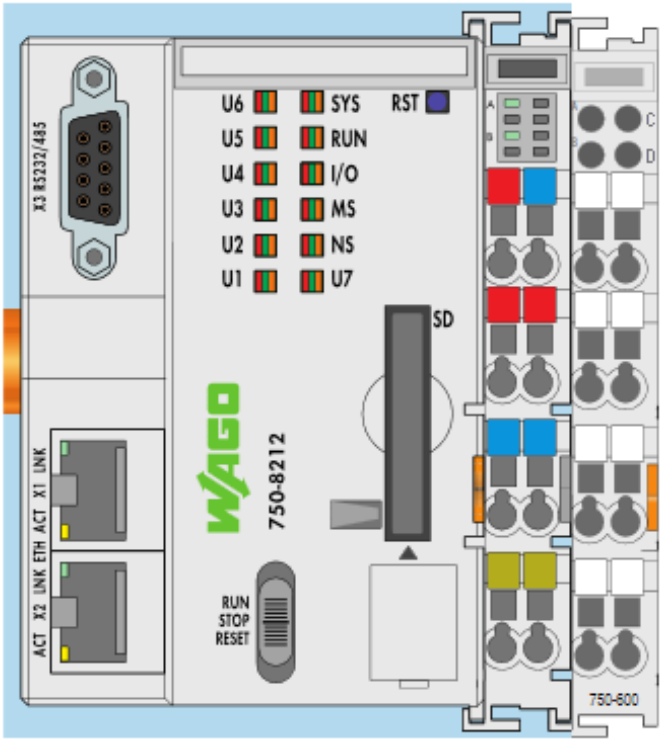


Figure 4: Schematic representation of your PLC, before adding the modules

After that, you have to implement all the inputs, outputs and communication interfaces connected to the controller. Normally, you can find the references by looking at the

PLC (figure 5). In our case, we will use the 16 digital inputs module ***750-1405***, and the 16 digital outputs module ***750-1505***.

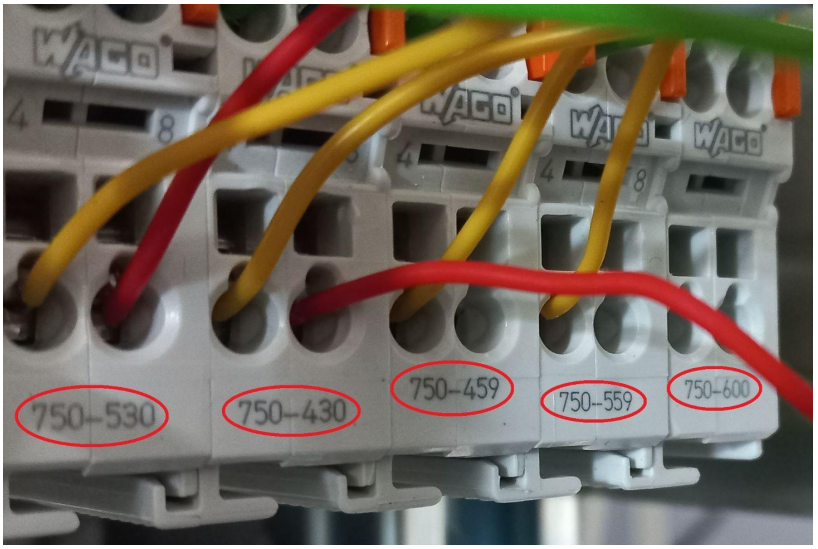


Figure 5 : You can find the references at the bottom of each module

It is possible to type the reference of each module in the research bar as for the controller and drag them to the controller diagram. You should have something like on the figure below.

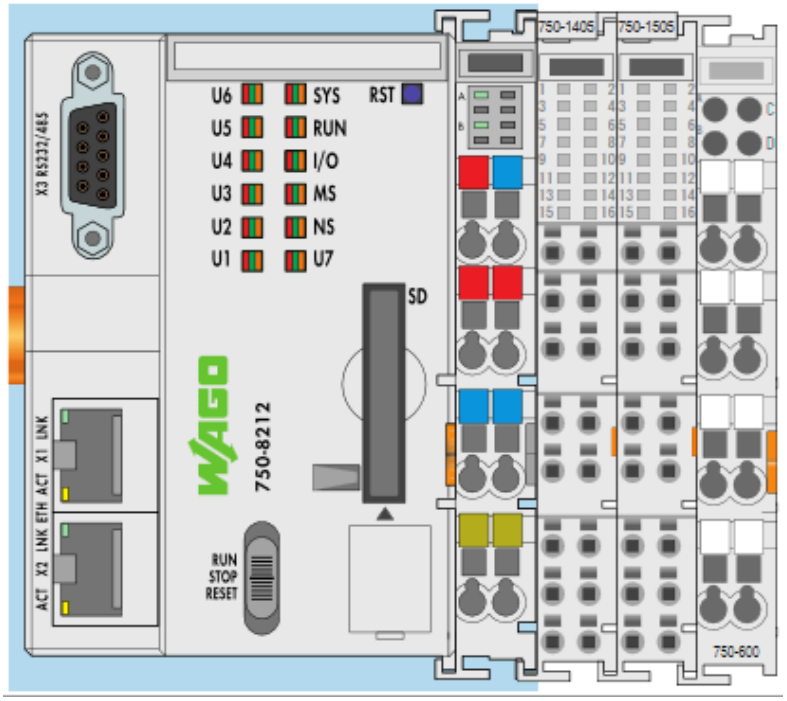


Figure 6 : The final configuration

Variables configuration

The next step is to configure the variables. We need three inputs : *Max_Level*, *Min_Level*, and *Client_Request*, and two outputs : *V1* and *V2*. Double-click on the digital input module. In the window "*Local bus /IO Mapping*", extend the tree structure by clicking on

the "+". You can now see the 16 free inputs in the table. To add an input, double click on a case in the "*Variable*" column, and type its name.

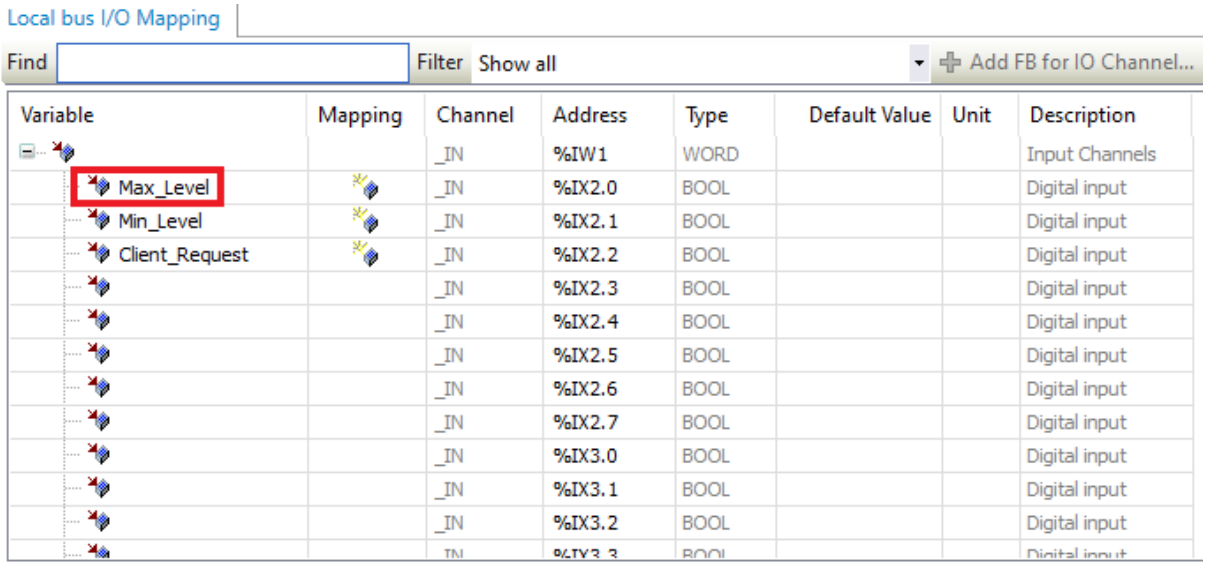


Figure 7 : Double-click on a case in the "Variable" column to create a variable

You can see that the address and the type are automatically created. You can do the same with the output module. At this point, your variables are configured.

⚠ Warning ⚠ : If you need to delete a variable, don't press the "*delete*" key, it will delete the whole module. Instead, you must double-click on the name of the variable, and delete its name with the "*backspace*" key.

Compilation and programming

Program writing

In the "*Program Structure*" window, click on "*Program Structure*" at the bottom right. This tab will show the structure of your program. By default, a ST program is already created, you can right-click on it and delete it.

For our program, we need a Grafcet to command the tank, and a Ladder to activate the valves depending on the Grafcet states. To add a program, right click on "*Application*", then click on "*POU*". In the new window, name your program "*Main_Grafcet*", and select "*Sequential Function Chart*" (in fact this is Grafcet) in the "*Implementation language*" drop-down menu. Click on "*add*".

First of all, we will focus on the Grafcet part of the program. A good advice is to start by drawing a diagram on a piece of paper.

Exercise

Try to imagine how the command Grafcet could be before looking at the solution below. There are a lot of possible solutions. At first, it is not necessary to determine the most optimized. Don't forget the possibility to use "*AND*" or "*OR*" divergences.

Solution

There are at least 4 steps required for this process. The first one (Step0) is the initialization step, followed by a transition to the next step if the water level is between Min_Level and Max_Level. In the second step (Step1), the valve V1 is opened to fill the tank. At this point there are two possibilities. If Max_Level is reached, the program waits for Client_Request in Step2. If Client_Request is true, the valve V2 is opened until Min_Level is not reached or Client_Request is true.

In *e!Cockpit*, we can create the Grafcet and the Ladder transitions shown below :

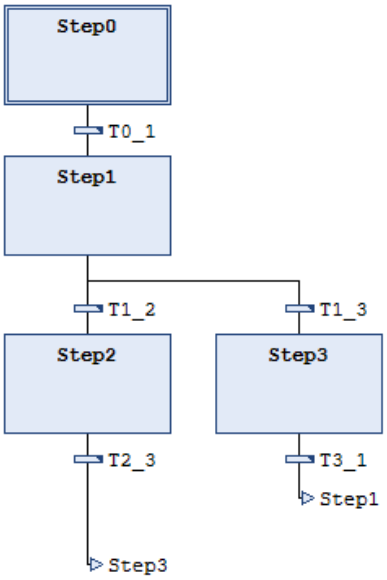
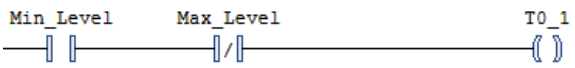
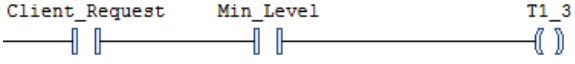
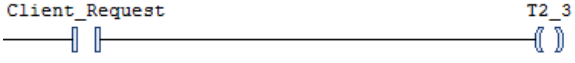


Figure 8 : The Grafcet for this program

The transitions are all visible in the table below :

Boolean Expression	Ladder Expression
T0_1 = Min_Level AND not(Max_Level)	
T1_2 = Max_Level	
T1_3 = Client_Request AND Min_Level	
T2_3 = Client_Request	
T3_1 = not(Min_Level) OR not(Client_Request)	

Now we need to create a Ladder to activate the outputs according to the states of the Grafcet. Right click on "*Application*" and select "*POU*". Choose the Ladder language and name it "*Ladder_Ouputs*". In the programming window, there is already a spot to add a coil. As we are using two outputs, right click in the middle of the window and select "*Insert Network*".

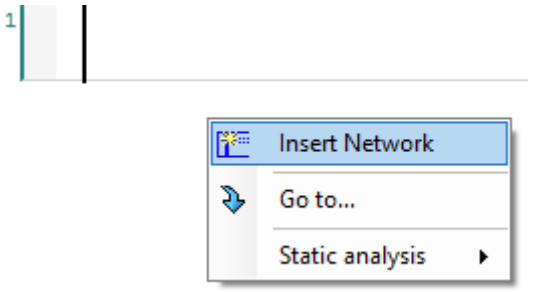


Figure 9 : Click on "Insert Network" to add an output in the ladder diagram window

In our case, V1 is activated when the Grafcet is in state 1, and V2 is activated when the Grafcet is in step 3. To program this, you must add a contact and type "*Program_Name.Step_Name.X*", replacing *Program_Name* by the name of the Grafcet program and *Step_Name* by the name of the state. In our case, we have the output ladder visible in the figure 10 below.

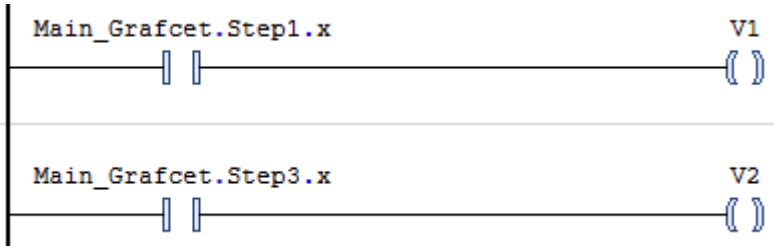


Figure 10 : Output Ladder diagram

Task Configuration

Your program is now finished. Before compiling it, you have to tell the software which programs are used together. To do this, you have to click on "*Task Configuration*" in the program structure window, and then on the task created by default, "*PLC_Task (1)*". There is a "*(1)*" after the name because there is only 1 program associated with this task. In the opened window, there is the default program that we deleted at the beginning. You can select it and click on "*Remove Call*".

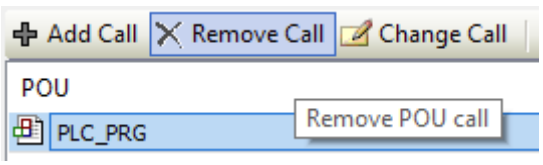


Figure 11 : You should remove the default program

After that, you can click on "*Add Call*" to add the **two** programs in the task configuration.

Compilation

To check for programming errors, you can now click on the "*Program*" tab at the top of the screen, and then on "*Compile*". If there are errors in your programs, messages will appear and you can double click on it to see where the errors are. If there are no errors, you can proceed to the next step.

Scada Interfacing & Simulation

HMI Creation

To simulate the program, the best way is to visualize it. This is why we are going to create an HMI. In e!Cockpit, an HMI screen is called a "*visualization*". To create one, right-click on "*Application*" in the program structure and then on "*visualization*". A new window appears : this is where you can choose the symbol libraries. Select all the available libraries, and click on add.

On the blank screen of the visualization, you can now add the objects you want to create : the tank, the valves, and the inputs. Remember that this is a schematic vision and does not need to be beautiful. There are a lot of possible solutions (try to be inventive !), here is one in the figure 12 below.

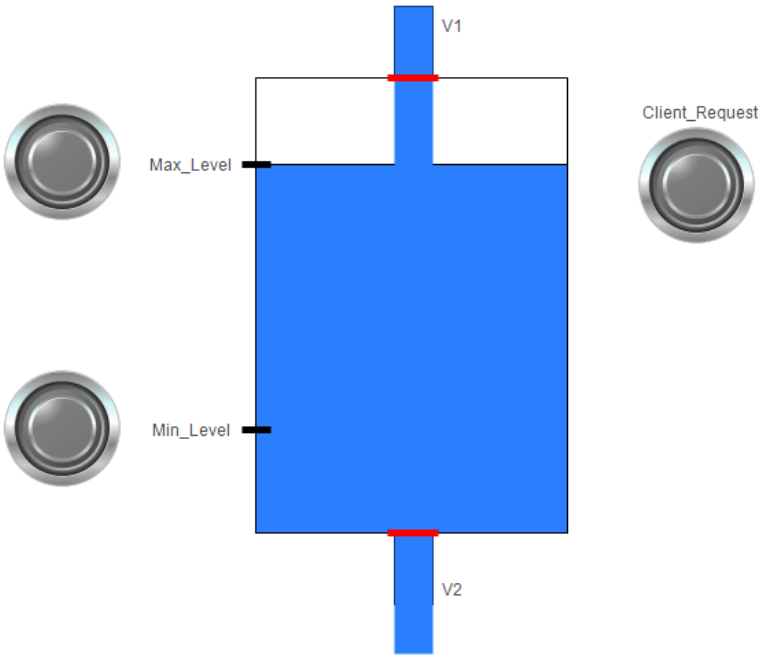


Figure 12 : A solution for the visualization

Do not forget to compile after creating the HMI to verify that there are no errors in the animation expressions for exemple.

Simulation

To simulate and validate the process, click on the "*Program*" tab and then on "*Simulate Application*". If you are on the visualization window, it will become white : it is normal. You must click on "*Run*" to launch the simulation. When it is launched, you can verify your process with the HMI you just created, but you should also verify your Grafcet directly. Indeed, in e!Cockpit you can follow the path of the program in real time. This is really useful to detect a problem. You can also check the values of inputs and outputs in Ladder and Structured Text.

Bonus Exercise

We have seen how to create the program with Grafcet and Ladder, but it is possible to use Ladder only. Create a new project and try to find out how to use only Ladder diagrams. In this case, is it easier or more difficult ? What are the advantages or disadvantages ?

Conclusion

In this lab, you learned how to implement a simple process with Grafcet and Ladder, and how to use simulation in e!Cockpit. In the next lab, we will work on a hydroelectric dam, but with only one language. The difficulty will increase a bit but we will be there to help you if necessary. If this lab was really difficult for you, tell us and try to do the exercises alone before the next lab.