LOGIC GATES DISPLAYED IN PLC S7-1500 SIEMENS AND AUTOMATION COMMUNICATION THROUGH AN OPC TO THE COMPUTER.

Authors: David Steven Galvis Arévalo 1802584, Eric Xavier Buitrago Forero 1802410, Johan Andrés Sebastián Gómez Espinosa 1802322

Co-Authors: Olga Lucía Ramos Sandoval, Ph.D.

Abstract: In this document, the result of the implementation of the logic gates AND, OR, NOT and XOR in the different programming languages that the Simatic S7-1500 allows, to become familiar with the configuration of the PLC and the TIA Portal software is carried out. , and then communicate with the Automation Studio simulator. This in order to be able to control a process from a computer, since this is one of the main characteristics of an industrial automation process and also allows expanding knowledge in the connection of the controller with a server. To carry out the entire development of the practice, it was necessary to consult in the bibliography and within the TIA Portal help option the different programming languages and their respective structure, declaration of variables and among other characteristics of each language. Subsequently performing the communication of the OPC server that allows the PLC to share data with the Automation Studio. And finally the programming was carried out in the different languages and they were all simulated and with the expected result, the connection between both is also carried out, which offers the great advantage of communicating remotely from a computer with the control of the process that is being carried out. developing, which is the penultimate step of the pyramid of any automated industry and therefore is one of the important pillars in it.

Abstract: This document shows the result of the implementation of logic gates AND, OR, NOT and XOR in the different programming languages that Simatic S7-1500 allows. This allows to know better the PLC initial configuration and the TIA Portal software and then to communicate with the Automation Studio's simulator. It is with the objective of controlling a process from a computer, this is already one of the main features of an industrial automation process and it allows to expand the knowledge of connecting the controller to a server. To make this development it was necessary to consult in bibliography and the TIAPortal's help option the different programming languages and their structures, declaration of variables and some other features of each language. Later setting connection between PLC and Automation Studio by an OPC server. Finally, a connection developed between both previously mentioned was simulated with a good result. It offers the great advantage of communicating with a computer and to process the control. This is the penultimate step of every automated industry and because of this it's one of the most important pillars of itself.

Introduction

A programmable logic controller is a digital electronic device that uses a programmable memory to store instructions and carry out logical, sequence configuration, synchronization, counting and arithmetic functions, for the control of machinery and processes, in this it is mainly implement logic functions as a first step in learning programming [1].

The PLCs of the SIMATIC S7 family are based on the automation system from Siemens. These are the solution to complex automation processes, their principle is that each PLC consists of a CPU and an input and output module. The CPU contains the program created by the user while the I/O modules allow the communication processes with the field equipment [2].

In principle, the STEP 7 standard software only allowed SIMATIC systems to be programmed in only 3 languages, LAD, FBD and STL, but later another one was added to the list, SCL. The LAD language consists of a contact diagram as well as an electrical diagram, the FUP language is based on logic functions represented in blocks, the STL language is lists of ladder instructions, where the "A" represents a serial connection and the letter “O” represents a parallel connection. And finally, the SCL language is based on conditional statements “if - else”[3].

An OPC server consists of a standard set of interfaces, properties, and methods for use in manufacturing and process control applications. It is based on Microsoft OLE (Object Linking and Embedding) and COM (Component Object Model) technologies. Its main objective is to standardize access to the data that exists in the industry, so that several clients can access the data managed by an OPC server. In this way, the development of several drivers is reduced to a single driver [4]. How to create a new database in an OPC IBH [5].

Objectives

- General:

*Implement logic gates in all the programming languages of the TIA Portal and demonstrate their operation through communication with the PLC and Automation Studio.

- Specifics:

*Know the configuration of the PLC modules depending on its model.

*Design AND,OR,NOT and XOR gates in the TIA Portal languages.

*Communicate the TIA Portal with the PLC and Automation Studio via an OPC.

Development

First, you must know the operation of each logic gate to be implemented and how it works in each of the 4 programming languages of the Simatic S7-1500 PLC. In addition to the use of an OPC as a link between the PLC and Automation Studio and how to configure the Automation server to allow the reading of these variables through the data block.

Materials:

- PLC Siemens S7-1500
- Realtek RTL8188EU Wireless LAN 802.11n USB 2.0 Network Adapter.
- TIA Portal V13
- OPC IBH
- Automation Studio 6.1

Procedure

First a new project is created in TIA Portal V13 and then a device must be added, for this case it is

decided to use the S7-1500 PLC controller in firmware version 1.0, which is the version that have the PLC's that are available in the university laboratory.

Once the TIA Portal is open, we proceed to configure the modules that accompany the CPU, these must be configured in order and they must all be in the same version. The following modules are then configured:

- Power:

PM 70W 120/230VAC

- Communication module:

PROFIBUS CM 1542-5

- Digital input module:

DI 16x24VDC HF

- Digital output module:

DQ 16x24VDC/0.5A ST Analog

- input module:

AI 8xU/ I/RTD/TC ST

- output module:

AQ 4xU/I ST

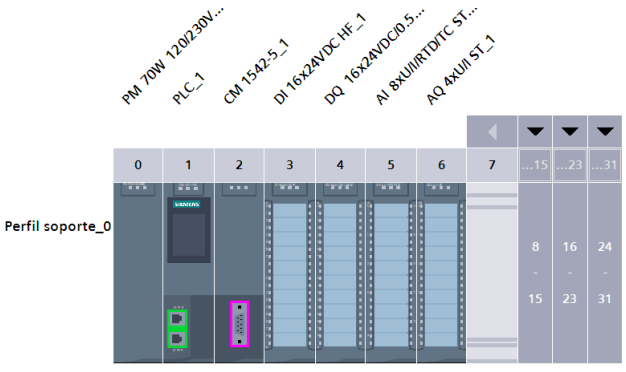


Figure 1: CPU and PLC modules.

The device is configured as shown in figure 1, where all the modules are with their respective firmware version, which in this case was 1.0.

Now the variables that you want to program in the PLC are declared, in the panel on the left of the screen, select the PLC that we are using, the "PLC Variables" tab and then "Show all variables". Just as shown in figure 2.

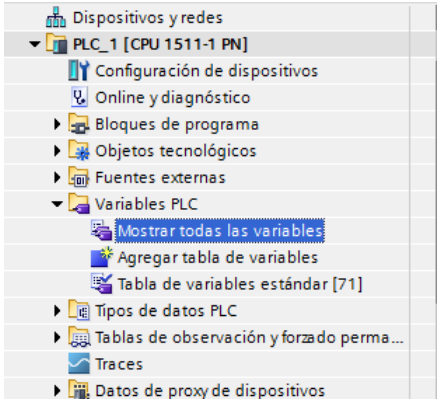


Figure 2: PLC Variables.

A blank variable table appears, in which the variables must be declared and a variable type and address must be selected. Being logic gates what you want to test, you select a Bool type variable and the operand identifier, which depends on whether the variable is input I, output Q, or a space in memory M. Then the address and bit number it refers to to which of the 16 digital inputs or outputs of the S7-1500 you want to assign the newly declared variable. For this case, only 2 inputs and 16 digital outputs are declared to be able to observe the 4 languages and the 4 gates at the same time, as shown in figure 3.

Variables PLC

	Nombre	Tabla de variables	Tipo de datos	Dirección	Rema..	Visibl...	Acces...
1	entrada1	Tabla de variables...	Bool	%I0.0			
2	entrada2	Tabla de variables...	Bool	%I0.1			
3	KOPAND	Tabla de variables...	Bool	%Q0.0			
4	KOPOR	Tabla de variables...	Bool	%Q0.1			
5	KOPNOT	Tabla de variables...	Bool	%Q0.2			
6	KOPXOR	Tabla de variables...	Bool	%Q0.3			
7	FUPAND	Tabla de variables...	Bool	%Q0.4			
8	FUPOR	Tabla de variables...	Bool	%Q0.5			
9	FUPNOT	Tabla de variables...	Bool	%Q0.6			
10	FUPXOR	Tabla de variables...	Bool	%Q0.7			
11	AWLAND	Tabla de variables...	Bool	%Q1.0			
12	AWLOR	Tabla de variables...	Bool	%Q1.1			
13	AWLNOT	Tabla de variables...	Bool	%Q1.2			
14	AWLXOR	Tabla de variables...	Bool	%Q1.3			
15	SCLAND	Tabla de variables...	Bool	%Q1.4			
16	SCLOR	Tabla de variables...	Bool	%Q1.5			
17	SCLNOT	Tabla de variables...	Bool	%Q1.6			
18	SCLXOR	Tabla de variables...	Bool	%Q1.7			

Figure 3: Variables assigned in the PLC.

Having this list of declared variables, we proceed to create the AND, OR, NOT and XOR gates in the main using the 4 programming languages of the S7-1500 PLC which are LAD, FUP, STL and SCL.

First you want to simulate the 4 gates in the 4 languages before being simulated and communicated to Automation Studio.

In the device panel of the TIA Portal located in the left area of the screen, you must add in "Program blocks" you must "Add a new block" for each language you want to use. Where you can also select if the block is an orientation block, a function block, a function block or a data block, as shown in figure 4.

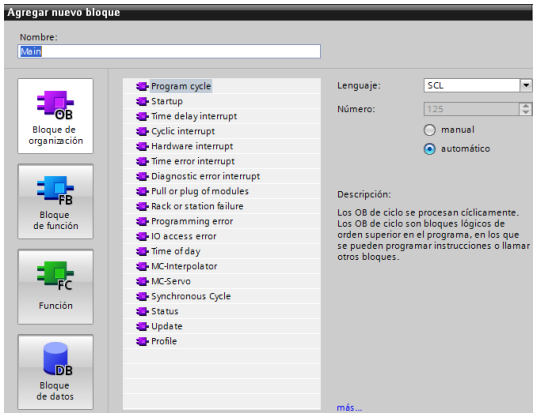


Figure 4: New block menu.

For the first LAD language, the AND gate corresponds to two normally open contacts connected in series, as shown in figure 5:

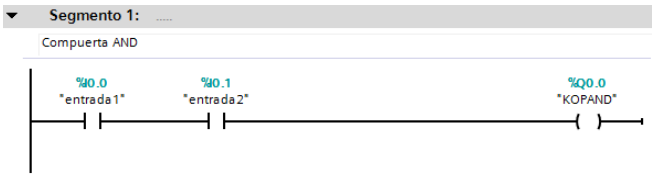


Figure 5: AND gate in LAD language.

In this same language, for the OR gate, two normally open contacts are connected in parallel, as shown in figure 6.

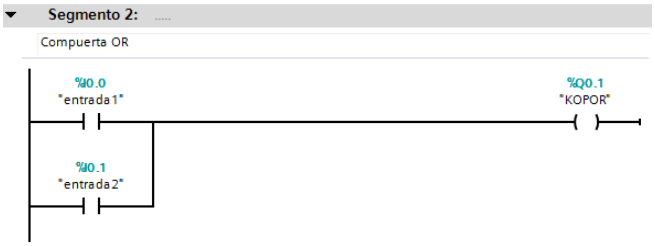


Figure 6: OR gate in LAD language.

Continuing with this programming language, the NOT gate consists of the input associated with a normally closed contact, so that the contact always opens when the digital input is 1, which output will be 0 and will be negated. The diagram is observed in figure 7.

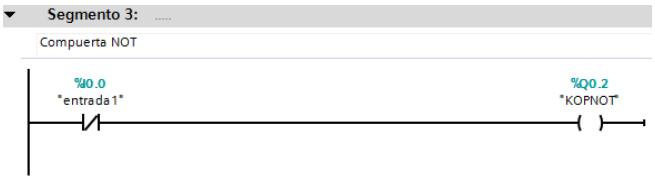


Figure 7: NOT gate in LAD language

And the last gate that corresponds to this language is the XOR, which consists of an input 1 as a normally open contact in series with an input 2 as a contact normally closed, these in parallel to a normally closed contact of input 1, in series with a

normally open contact of input 2. As shown in figure 8.

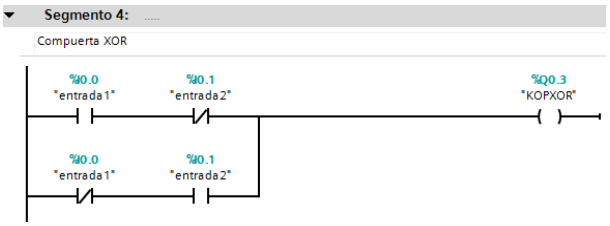


Figure 8: XOR gate in LAD language

Now for FBD language, function blocks are used, AND and OR gates are shown in figure 9.

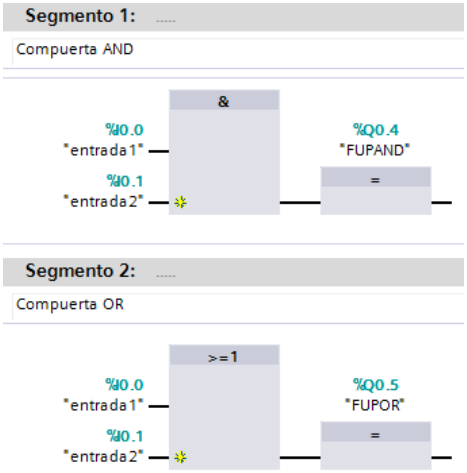


Figure 9: AND and OR gates in FBD language

AND NOT and XOR gates in FBD language are shown in figure 10.

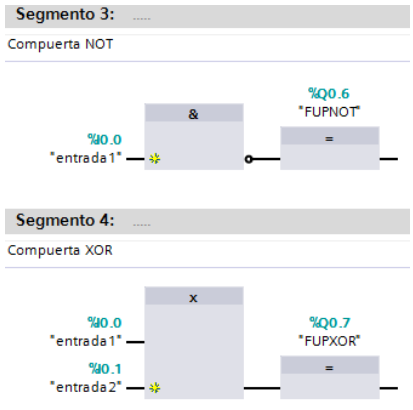


Figure 10: NOT gates and OR in FUP language.

Now for the STL language, which is a list of instructions, remember that the letter A represents a series connection and O a parallel connection. The 4 logic gates are programmed again, then: AND and OR gates are expressed in figure 11.

Figure 11: AND and OR gates in STL language.

And the NOT and XOR gates, where for the first the input 1 is denied by means of the "not" instruction and in the case of the XOR gate the letter X is written, which presents an exclusive OR gate: they are found in figure 12.

Figure 12: NOT and XOR gates in STL language

For the final language, which is SCL, its programming is much simpler, since it only requires a line of code that consists of matching the output to a command between inputs; they are then shown the AND,OR,NOT and XOR gates in this language, in figure 13.

```
1  "SCLAND" := "entrada1" AND "entrada2";
2  "SCLOR" := "entrada1" OR "entrada2";
3 IF "entrada1" THEN
4      "SCLNOT" := 0;
5  ELSE
6      "SCLNOT" := 1;
7  END_IF;
8  "SCLXOR" := "entrada1" XOR "entrada2";
```

Figure 13: Gates in SCL language

Having all the gates programmed, it is compiled and then loaded into the device, which in this

case, being first a simulation, is loaded into the device that allows it to be viewed by PLCSim.

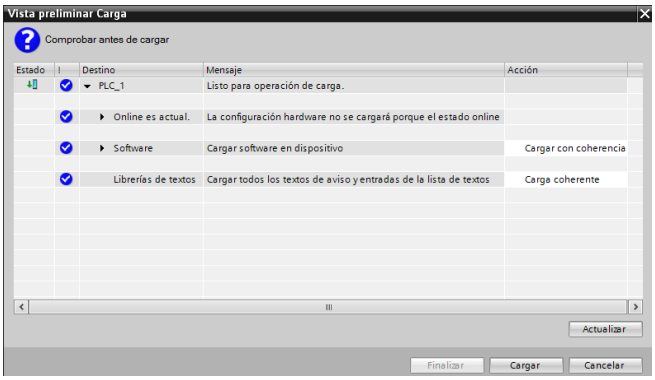


Figure 14: Preview of uploading program to the device.

The online connection is established through the type of interface PN/IE and Interface provided by the PLCSIM, which is PLCSIM S7-1200/S7-1500, and the connection is checked. As seen in figure 15.

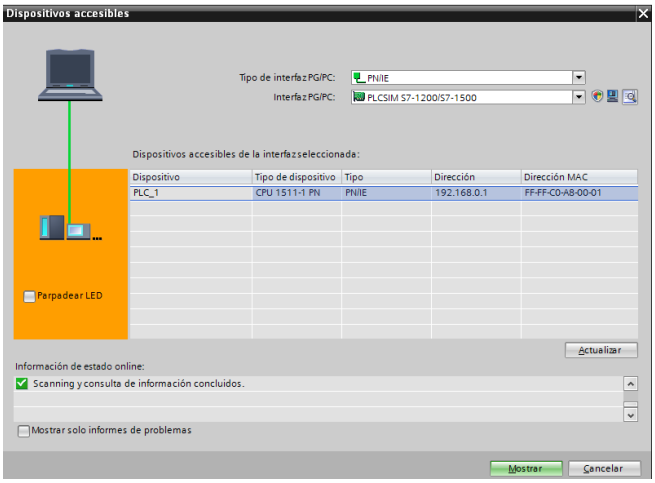


Figure 15: Devices accessible to the TIA Portal

Now you can see that the simulation runs perfectly. Because in the device pane of the TIA Portal all items are in the OK state, as shown in figure 16.

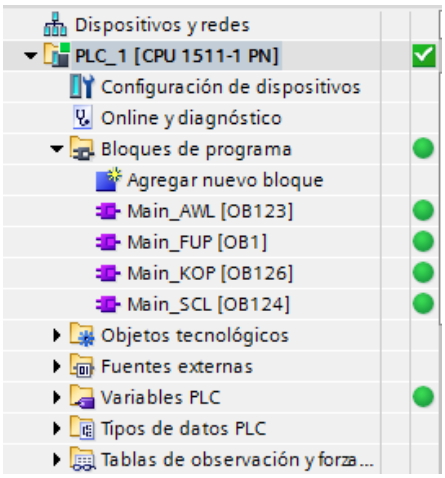


Figure 16: Devices working correctly in the online connection.

In the PLC SIM interface, the variables programmed from the PLC and found in the TIA are imported, in order to observe the full operation of the developed program. Now, all gates are tested using only the 2 inputs created.

When both inputs are in state 0, since only the input is associated with the NOT gate, this output has a state of 1 while the rest are in state 0. As can be seen in figure 17.

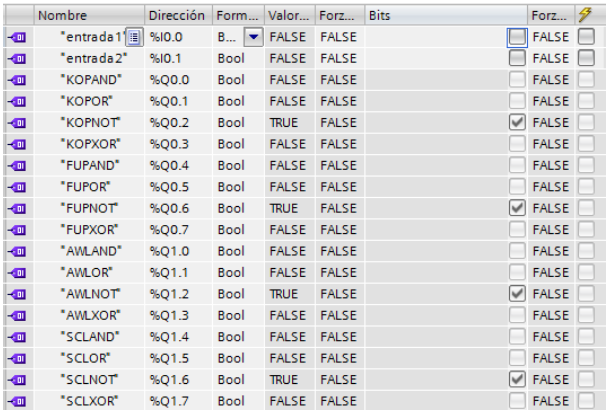
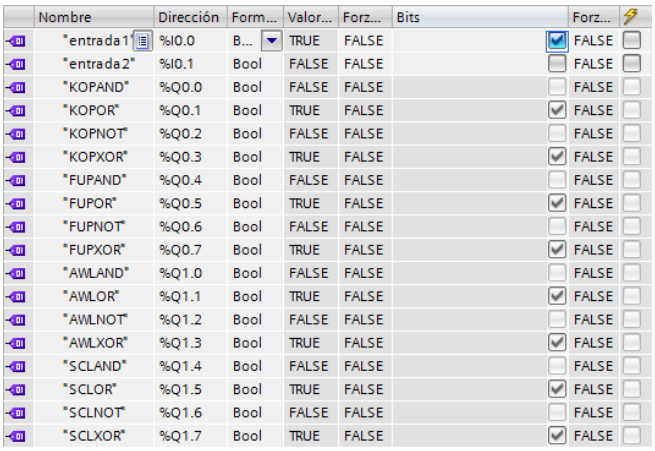


Figure 17: Simulation in PLCSim when both inputs are 0.

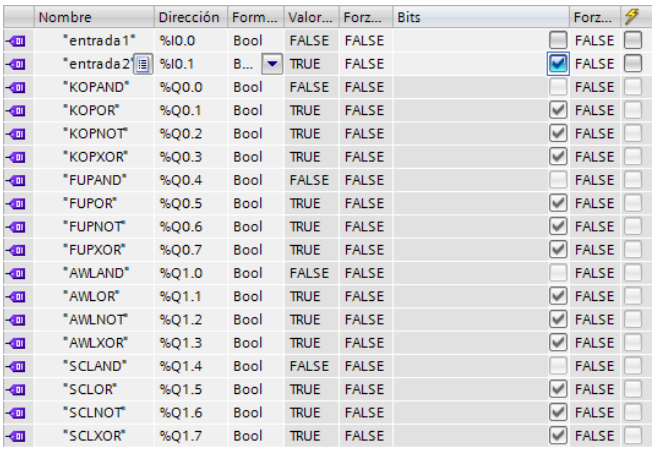
Now, for when “input 1” is in state 1 and “input 2” in state 0, the result is shown in figure 18.



Nombre	Dirección	Form...	Valor...	Forz...	Bits	Forz...
"entrada1"	%I0.0	B...	TRUE	FALSE		FALSE
"entrada2"	%I0.1	Bool	FALSE	FALSE		FALSE
"KOPAND"	%Q0.0	Bool	FALSE	FALSE		FALSE
"KOPOR"	%Q0.1	Bool	TRUE	FALSE		FALSE
"KOPNOT"	%Q0.2	Bool	FALSE	FALSE		FALSE
"KOPXOR"	%Q0.3	Bool	TRUE	FALSE		FALSE
"FUPAND"	%Q0.4	Bool	FALSE	FALSE		FALSE
"FUPOR"	%Q0.5	Bool	TRUE	FALSE		FALSE
"FUPNOT"	%Q0.6	Bool	FALSE	FALSE		FALSE
"FUPXOR"	%Q0.7	Bool	TRUE	FALSE		FALSE
"AWLAND"	%Q1.0	Bool	FALSE	FALSE		FALSE
"AWLOR"	%Q1.1	Bool	TRUE	FALSE		FALSE
"AWLNOT"	%Q1.2	Bool	FALSE	FALSE		FALSE
"AWLXOR"	%Q1.3	Bool	TRUE	FALSE		FALSE
"SCLAND"	%Q1.4	Bool	FALSE	FALSE		FALSE
"SCLOR"	%Q1.5	Bool	TRUE	FALSE		FALSE
"SCLNOT"	%Q1.6	Bool	FALSE	FALSE		FALSE
"SCLXOR"	%Q1.7	Bool	TRUE	FALSE		FALSE

Figure 18:Simulation in PLCSim when input 1 =1 and input 2 =0

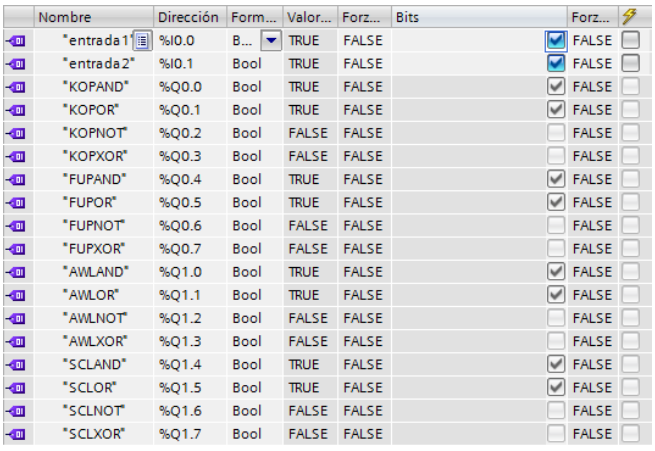
For when "input 1" is in state 0 and "input 2" in state 1. The result is found in figure 19.



Nombre	Dirección	Form...	Valor...	Forz...	Bits	Forz...
"entrada1"	%I0.0	Bool	FALSE	FALSE		FALSE
"entrada2"	%I0.1	B...	TRUE	FALSE		FALSE
"KOPAND"	%Q0.0	Bool	FALSE	FALSE		FALSE
"KOPOR"	%Q0.1	Bool	TRUE	FALSE		FALSE
"KOPNOT"	%Q0.2	Bool	TRUE	FALSE		FALSE
"KOPXOR"	%Q0.3	Bool	TRUE	FALSE		FALSE
"FUPAND"	%Q0.4	Bool	FALSE	FALSE		FALSE
"FUPOR"	%Q0.5	Bool	TRUE	FALSE		FALSE
"FUPNOT"	%Q0.6	Bool	TRUE	FALSE		FALSE
"FUPXOR"	%Q0.7	Bool	TRUE	FALSE		FALSE
"AWLAND"	%Q1.0	Bool	FALSE	FALSE		FALSE
"AWLOR"	%Q1.1	Bool	TRUE	FALSE		FALSE
"AWLNOT"	%Q1.2	Bool	TRUE	FALSE		FALSE
"AWLXOR"	%Q1.3	Bool	TRUE	FALSE		FALSE
"SCLAND"	%Q1.4	Bool	FALSE	FALSE		FALSE
"SCLOR"	%Q1.5	Bool	TRUE	FALSE		FALSE
"SCLNOT"	%Q1.6	Bool	TRUE	FALSE		FALSE
"SCLXOR"	%Q1.7	Bool	TRUE	FALSE		FALSE

Figure 19: Simulation in PLCSim when input 1=0 and input 2 = 1.

And finally, when both inputs are in state 1. It is shown in figure 20.



Nombre	Dirección	Form...	Valor...	Forz...	Bits	Forz...
"entrada1"	%I0.0	B...	TRUE	FALSE		FALSE
"entrada2"	%I0.1	Bool	TRUE	FALSE		FALSE
"KOPAND"	%Q0.0	Bool	TRUE	FALSE		FALSE
"KOPOR"	%Q0.1	Bool	TRUE	FALSE		FALSE
"KOPNOT"	%Q0.2	Bool	FALSE	FALSE		FALSE
"KOPXOR"	%Q0.3	Bool	FALSE	FALSE		FALSE
"FUPAND"	%Q0.4	Bool	TRUE	FALSE		FALSE
"FUPOR"	%Q0.5	Bool	TRUE	FALSE		FALSE
"FUPNOT"	%Q0.6	Bool	FALSE	FALSE		FALSE
"FUPXOR"	%Q0.7	Bool	FALSE	FALSE		FALSE
"AWLAND"	%Q1.0	Bool	TRUE	FALSE		FALSE
"AWLOR"	%Q1.1	Bool	TRUE	FALSE		FALSE
"AWLNOT"	%Q1.2	Bool	FALSE	FALSE		FALSE
"AWLXOR"	%Q1.3	Bool	FALSE	FALSE		FALSE
"SCLAND"	%Q1.4	Bool	TRUE	FALSE		FALSE
"SCLOR"	%Q1.5	Bool	TRUE	FALSE		FALSE
"SCLNOT"	%Q1.6	Bool	FALSE	FALSE		FALSE
"SCLXOR"	%Q1.7	Bool	FALSE	FALSE		FALSE

Figure 20: Simulation in PLCSim when both inputs are equal to 1.

Already after verifying that the programming in all the languages of all the gates was successful, we now proceed to test the communication of the PLC with Automation Studio through an OPC.

Simulation of all gates in all languages with variables in data blocks.

The previous procedure was in order to verify that each language was implemented correctly and without errors in its syntax.

Nowa program is made where the full operation of all the gates and in all the languages in a data block is tested, and that allows connecting with Automation Studio. and in which in turn can be controlled from the TIA Portal or from the PLC directly.

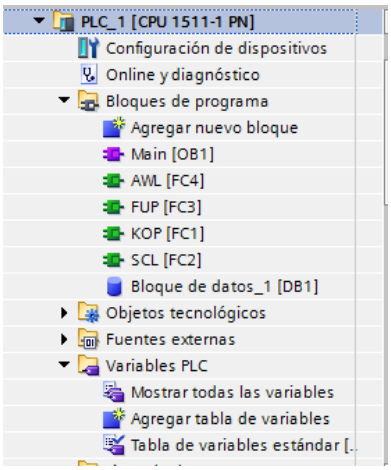


Figure 21: Configuration of 4 functions and data block

In the main, each language is connected as a function, and the variables are declared as shown in figure 22 and the variables of the data block as shown in figure 23.

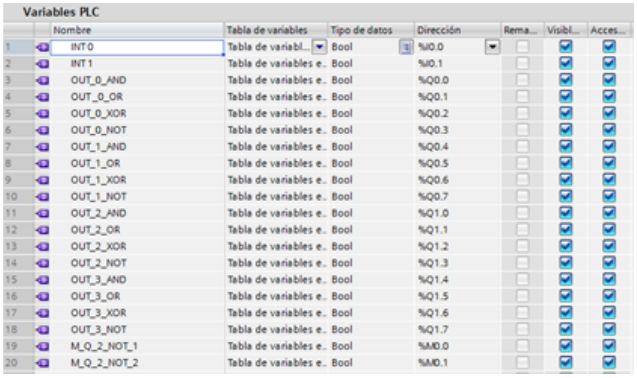


Figure 22: PLC main variables.

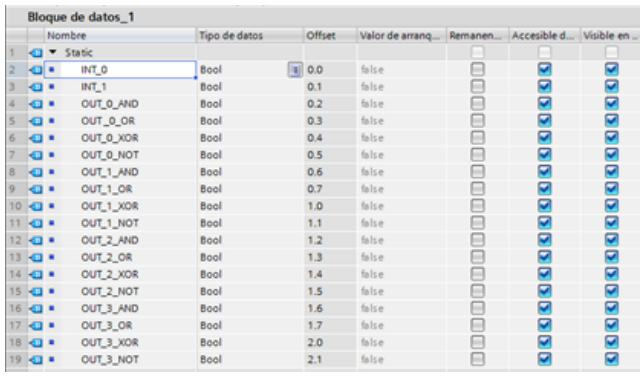


Figure 23: Data block variables.

The main of the program is shown below in figure 24.

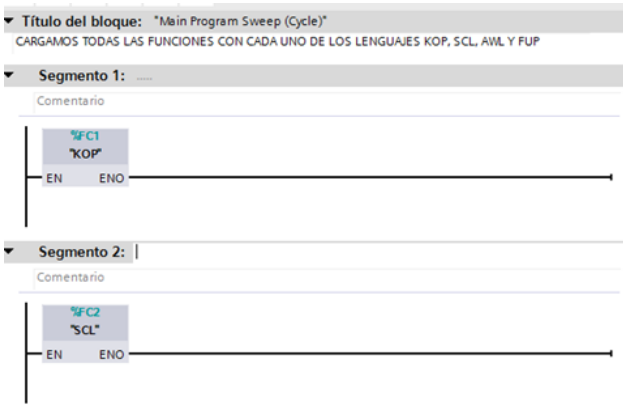


Figure 24: Block functions in main

For the AWL language, the implementation of each gate is given as follows, as expressed in the following figures

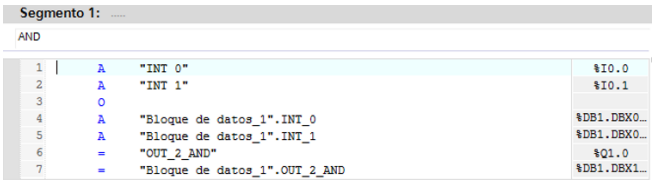


Figure 25: AND gates in STL.

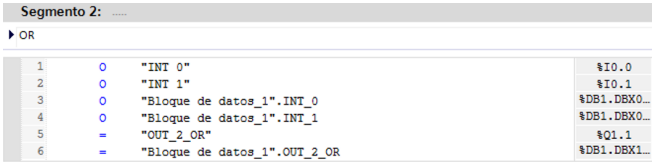


Figure 26: OR gates in STL.

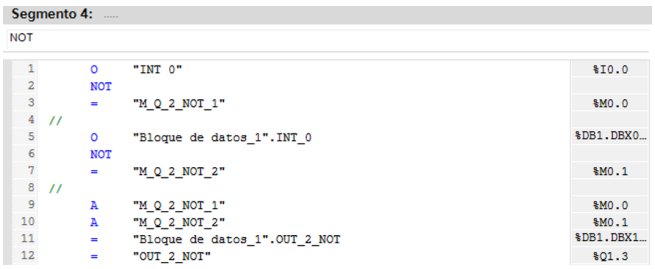


Figure 27: NOT Gates in STL

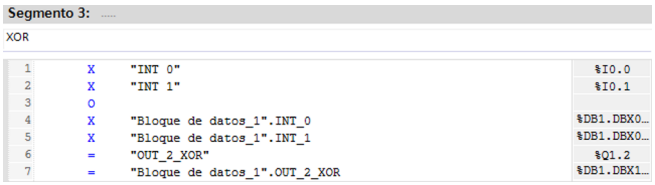


Figure 28: XOR Gates in STL

Then, the implementation of the gates in the FUP language is shown in the following figures.

Figure 29: AND gate in FUP.

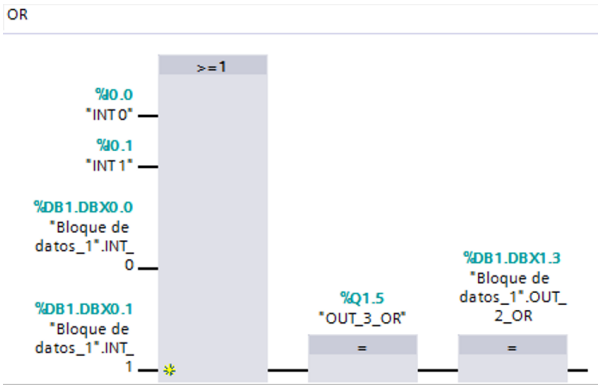


Figure 30: OR gates in FUP.

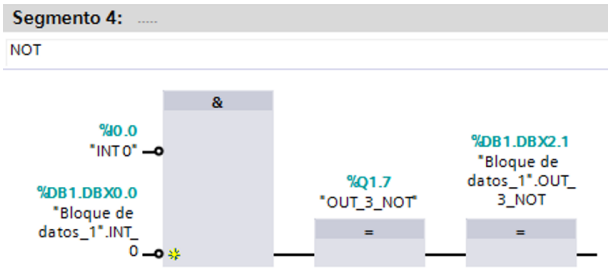


Figure 31: NOT gates in FUP.

Figure 32: XOR gate in FUP

The implementation of the KOP language is shown below.

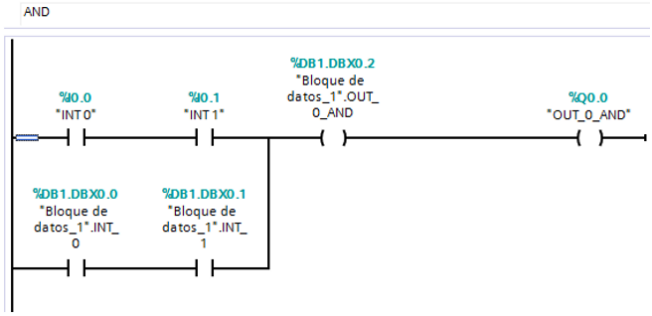


Figure 33: AND gates in LAD.

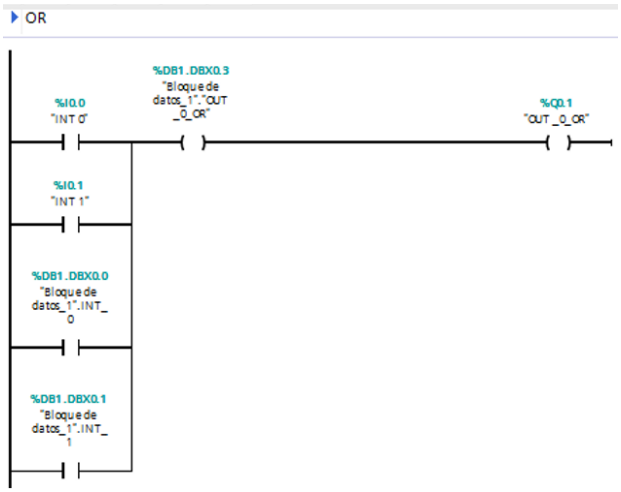


Figure 34: OR gates in LAD.

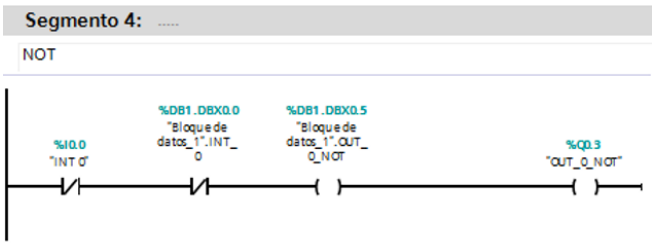


Figure 35: NOT gates in LAD.

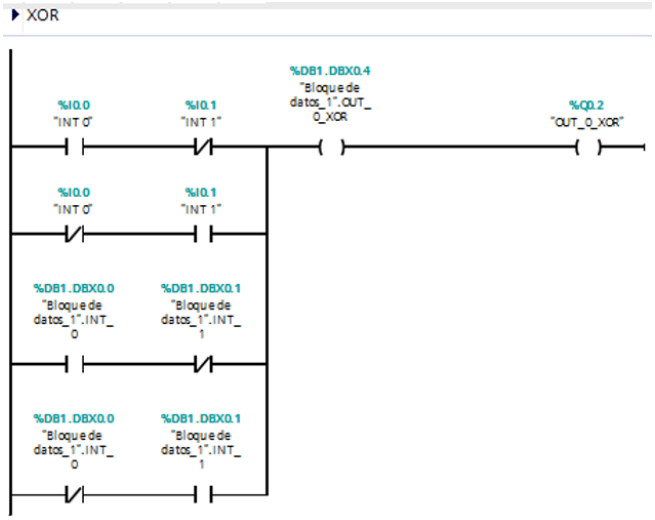


Figure 36: XOR gates in LAD.

And finally, the 4 gates implemented in the SCL language are shown in figure 37.

```
//AND
"OUT_1_AND" :=( "INT 0" AND "INT 1" ) OR ("Bloque de datos_1".INT_0 AND "Bloque de datos_1".INT_1);
"Bloque de datos_1".OUT_1_AND := "OUT_1_AND";
// OR
"OUT_1_OR" := "Bloque de datos_1".INT_0 OR "Bloque de datos_1".INT_1 OR "INT 0" OR "INT 1";
"Bloque de datos_1".OUT_1_OR := "OUT_1_OR";
//XOR
"OUT_1_XOR" := ("INT 0" XOR "INT 1") OR ("Bloque de datos_1".INT_0 XOR "Bloque de datos_1".INT_1);
"Bloque de datos_1".OUT_0_XOR := "OUT_1_XOR";
//NOT
"OUT_1_NOT" := NOT ("INT 0") AND NOT("Bloque de datos_1".INT_0);
"Bloque de datos_1".OUT_1_NOT := "OUT_1_NOT";
```

Figure 37: Gates in SCL.

Communication with Automation Studio.

Now in a new case an AND gate is tested in LAD language, the variables must be created in a data block that will be communicated to Automation Studio. For this communication to be successful, the CPU must be allowed to communicate with external media, disabling the protection. This allows reading and writing on the PLC.

To carry out this procedure, you must go to "Device configuration" and select the CPU, you must move to the protection tab, and at the bottom select and activate the last box. As shown in figure 38.

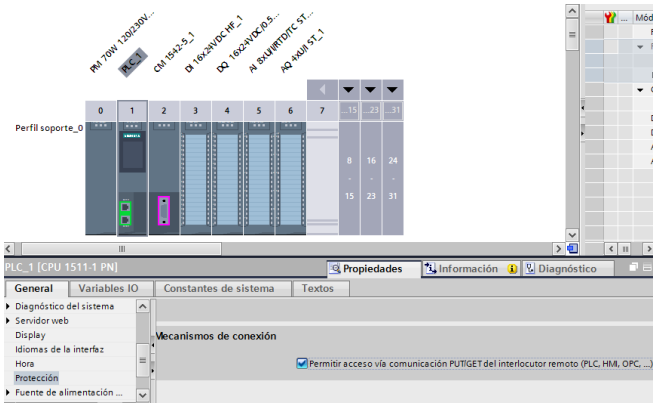


Figure 38: Allow communication access with the OPC.

After this, a data block is created and the variables are configured which interact with the Automation Studio.

Figure 39: Data block variables.

Having the variables declared, a function block is created that will allow us to place it in the "main" block later. This in order to better organize the programming structure of the PLC. Within the function you will find the program in LAD language, this can be seen in figure 40.

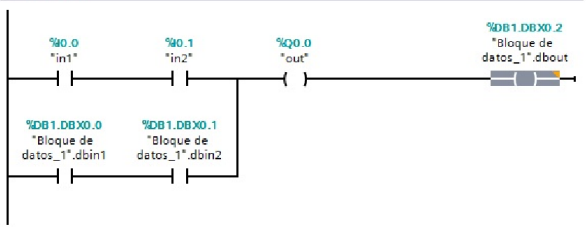


Figure 40: AND gates in LAD

An AND gate is programmed which allows, when activating the two inputs in the Automation, to

can observe a state 1 in the digital output of the PLC, and vice versa, when activating the 2 inputs in the TIA Portal or in the physical module of digital inputs of the PLC, it can be observed in the Automation that the output is in state 1

. properties of the data block, data communication with the OPC must also be allowed. To do this, right click on the "Data block" and select "Properties". In the attributes tab, the "Optimized access to the block" box must be deactivated. As shown in figure 41.

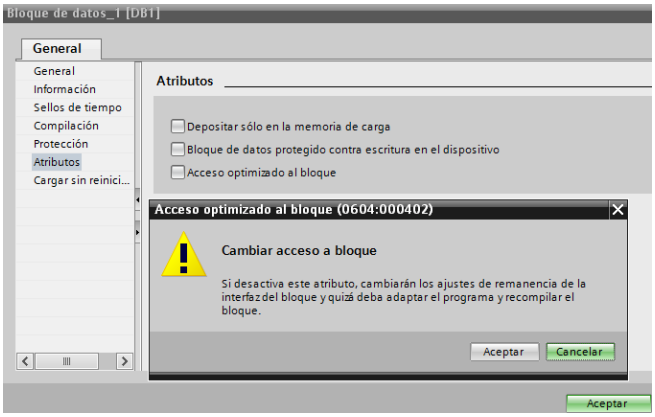


Figure 41: Deactivate data block protection.

Unchecking the box requires you to agree to change access to the block, allowing it to communicate seamlessly.

Now the OPC[6] must be opened, and a new project must be created.

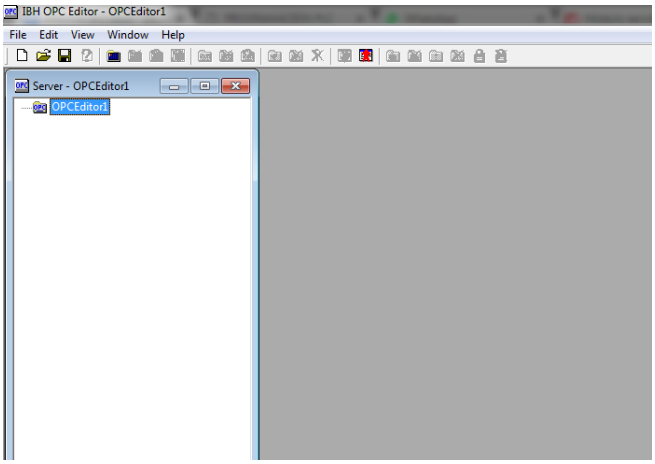


Figure 42: Create new file in OPC.

You must right click on the OPC Editor and insert a new PLC, then the S7-1500 is inserted.

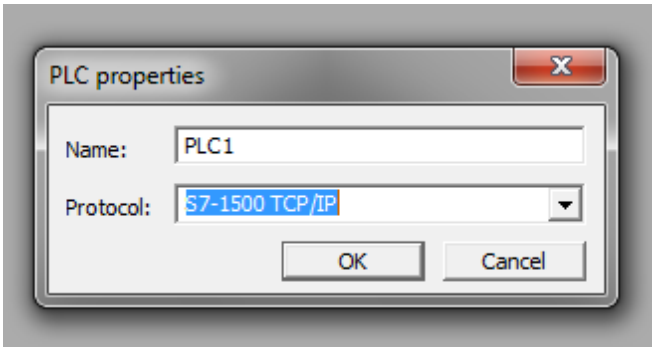


Figure 43: Insert a new PLC.

Now right click on the created PLC and select Connection Settings.

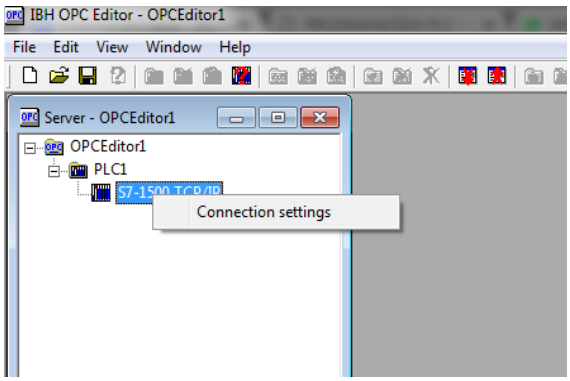


Figure 44: PLC connection settings.

After this, the IP of the PLC is selected in the pop-up window. As it's shown in the following.

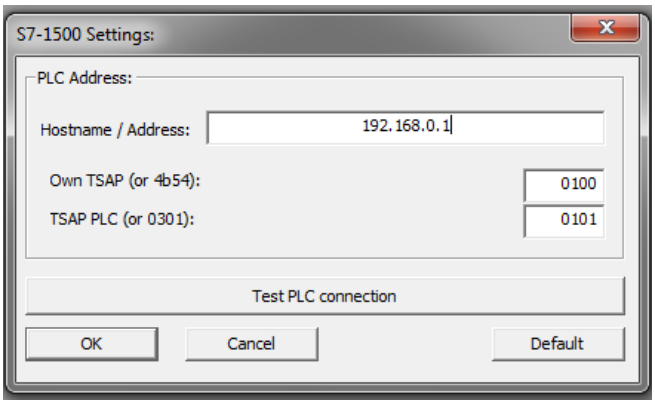


Figure 45: IP selection and connection test to the PLC.

Now the variables to be communicated must be defined, for this the "define variable" option is selected, as shown in figure 46.

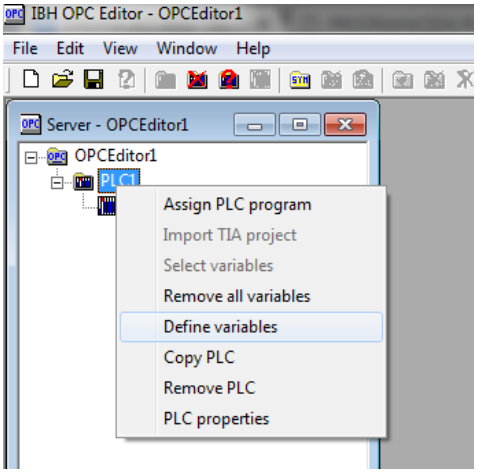


Figure 46: Define variables.

A name is declared for the variable and after this the type of variable is selected, which will be Boolean and then the address and bit to which it corresponds, will depend on the variable of the data block declared in the TIA Portal.

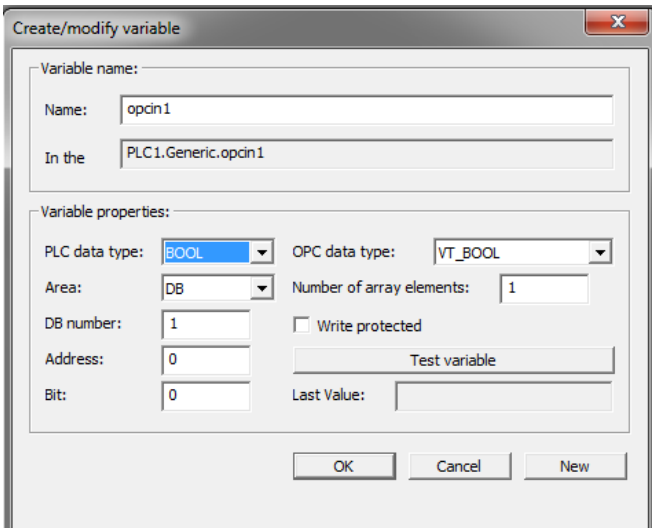


Figure 47: Declare variables and configure address and bit.

After declaring the 3 variables, 2 inputs and 1 output, the OPC server is transferred, in the icon shown in figure 48.

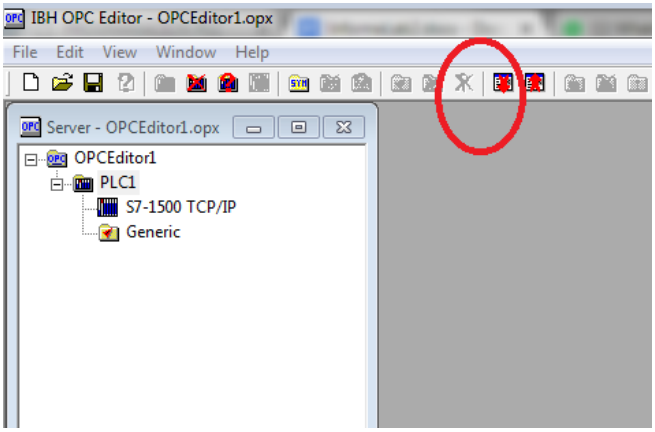


Figure 48: Transfer OPC server.

It is saved and after this, the IBHSoftec is selected.

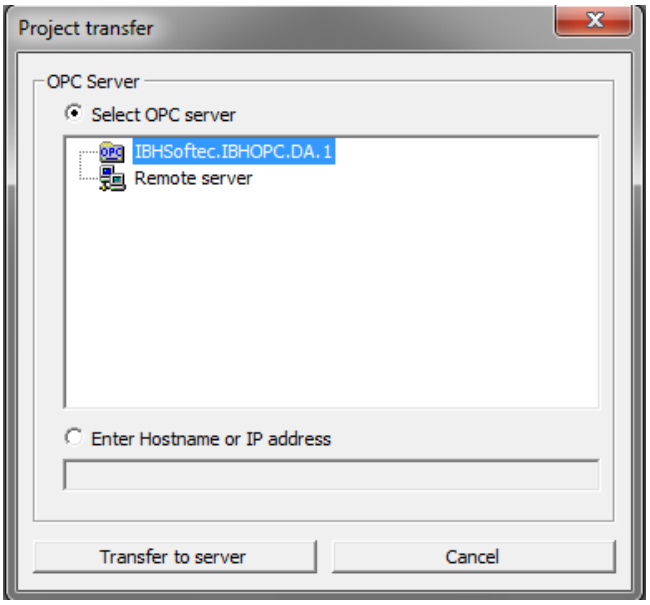


Figure 49: Transfer to IBH.

When this tab appears, it means that the transfer was successful and you are now proceeding to configure the OPC Server in Automation Studio.

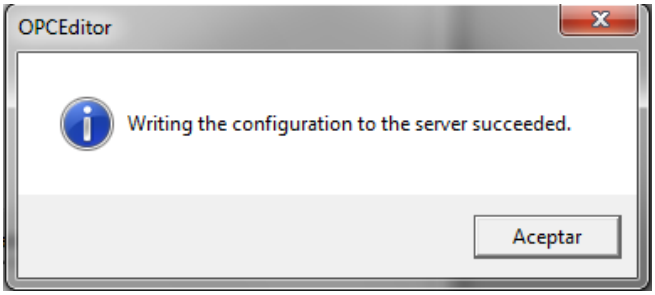


Figure 50: Configuration successful message.

After all the configurations made in the OPC have been completed, Automation Studio opens. For the first input a switch is created in Automation, for this it is selected as shown in the image.

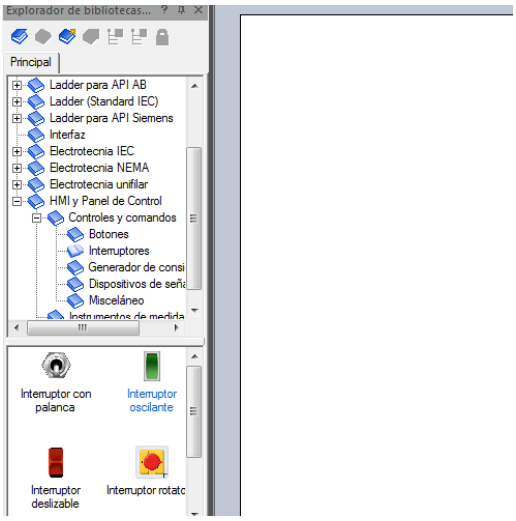


Figure 51: Automation Studio main menu.

In the left panel follow the following direction: HMI and Control Panel -> Controls and Commands -> Switches and select the toggle switch.

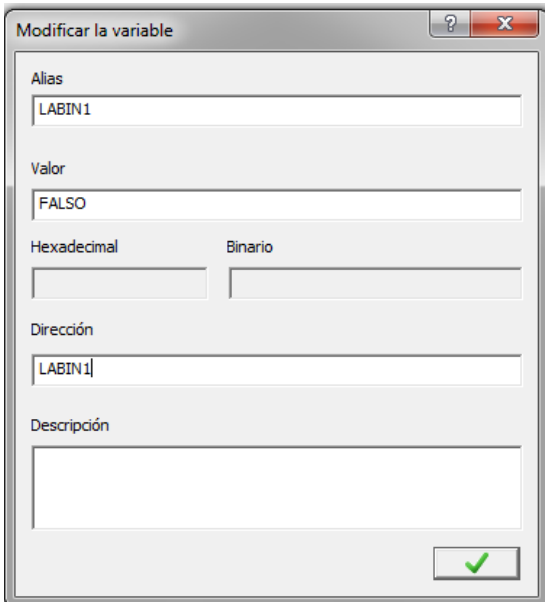


Figure 52: Panel to modify variables.

The previous pop-up window appears, where we name the switch of the graphical interface in the Alias and Address. The name "labin1" is given to switch 1 and "labin2" to switch 2.

When you double click on any of the switches, a window should open, in which you click on External Links as shown in the figure 53.

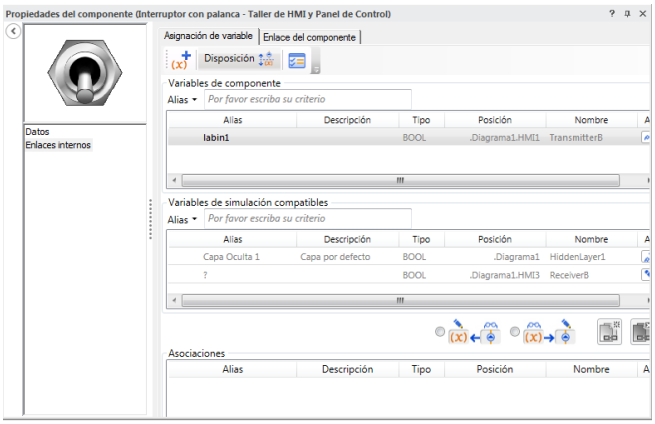


Figure 53: General component configuration panel.

Now an external variable is created by clicking on the "[+ (x)]" button and in the new window the type of variable "bool" is selected and a name is given to the variable, in this case it is called "AUTIN1".

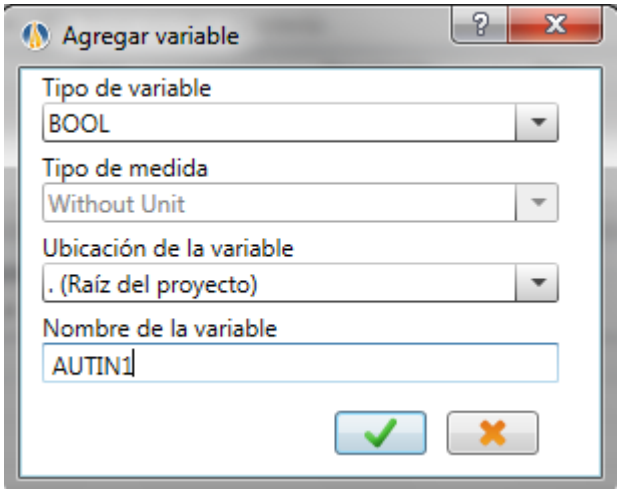


Figure 54: Panel to add variable.

After this, the external variable created is associated with the interface variable (switch or light) and the link is created by performing the following procedure.

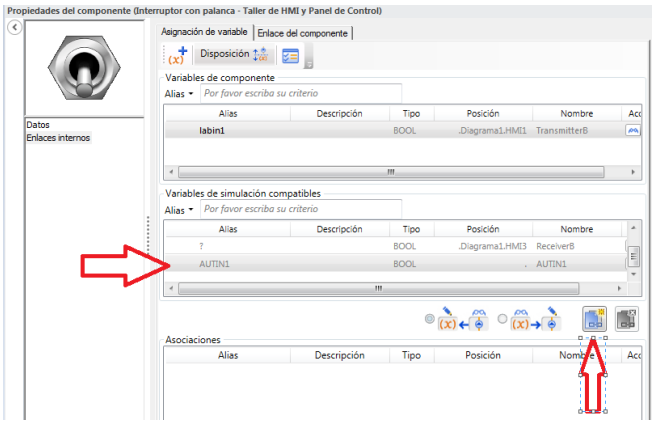


Figure 55: Assign link

It is confirmed that the link has been created when the following appears at the bottom of the window.

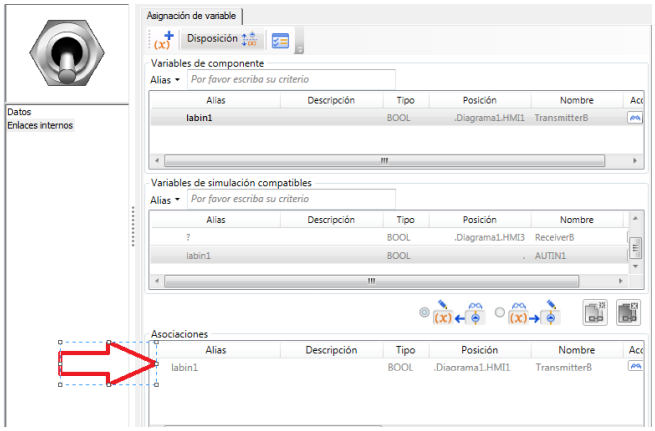


Figure 56: Links assigned.

This same procedure is done with the second switch and with the LED light.

Then the "Variable Manager" option located in the Tools tab is selected, as shown in figure 57.

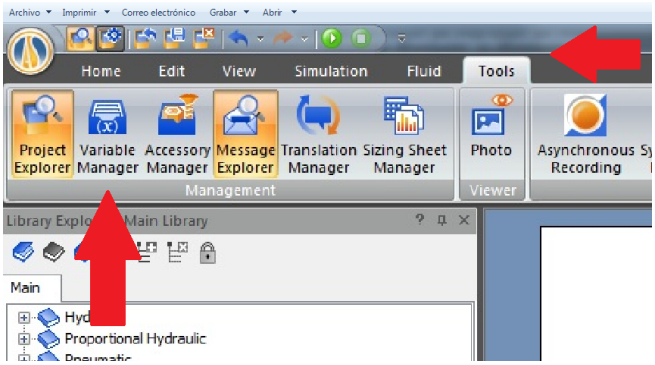


Figure 57: Tools panel.

When the window opens, click on the last option, which is "OPC Client" as shown below in figure 58.

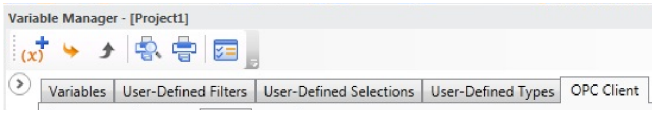


Figure 58: Variable Manager Interface.

Now, in the new window in the OPC Servers tab, a new Server is added, which is the IBH, manufacturer of the OPC used. A group is also added that will be the one that comes from the OPC IBH, then in items the group already created is selected and finally in the Links or links tab the Automation Studio variables are linked with those created in the OPC. They must be linked as read and write variables and create the link, as shown in figure 59.

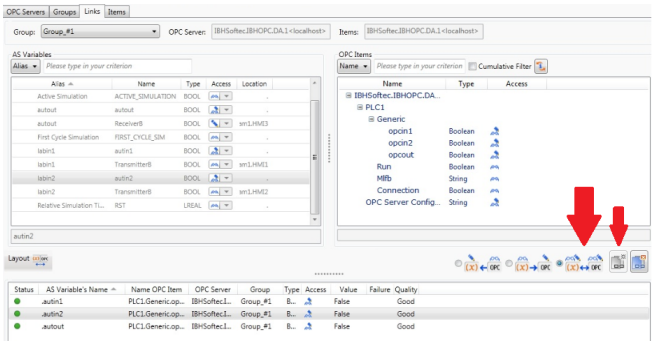


Figure 59: Variable link window.

Finally, when performing this procedure, there is communication between the PLC and Automation Studio, and it interacts to check it. Leaving both inputs in state 0, the bulb does not turn on and then when activating both, it is placed in state 1, as shown in figure 60.

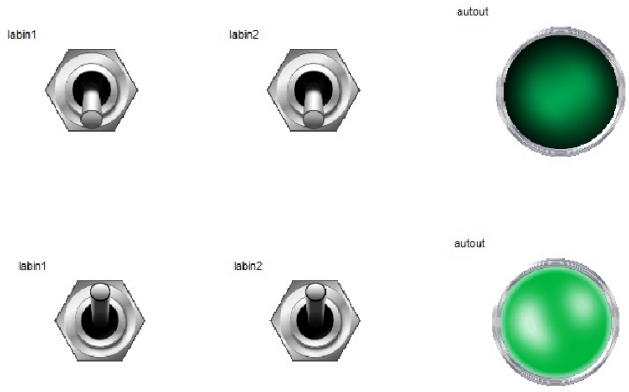


Figure 60: Automation Studio inputs and outputs.

Figures 61 and 62 show that the digital output Q0.0 of the PLC is activated despite the fact that its digital inputs are at 0, this is because they were activated from the Automation Studio.

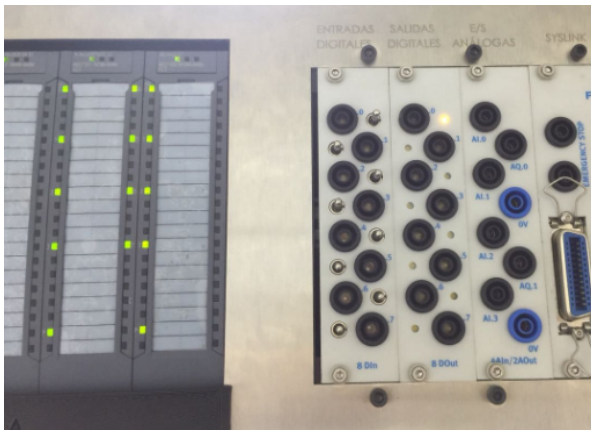


Figure 61: PLC switches.

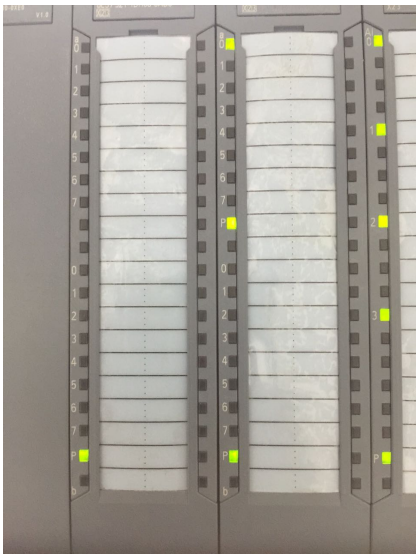


Figure 62: PLC digital inputs and outputs.

Finally showing the communication between the PLC and the Automation Studio.

Conclusions

*It can be concluded that to carry out a complete communication, that is to say with real-time visualization, all the parameters related to the PLC must be included, be it its modules, its communication protocol or its firmware version.

*To know the version of each of the modules, it is necessary to know the version in which the PLC is configured.

*The OPC component within the communication is already standardized which facilitates communication between the MES of a company and its control and field network.

*The implementation of the different logic gates was carried out effortlessly, since each language already has a default command for each of them.

References:

[1]W. Bolton, Mechatronics: electronic control systems in mechanical and electrical engineering, Editorial Alfaomega, 2 ed, pp 423 - 431.

[2]C. Dinis, G. Popa, A. Iagar, “Control of mechanism for pushing trucks using Siemens PLC”, 2016 International conference on applied and theoreticalelectricity (ICATE), 2016.

[3]P. Mengual, STEP 7: An easy way to program Siemens PLC. Editorial Marcombo, pp 72-84.

[4]R. Alves, J. Normey-Rico, A. Merino, C. de Prada, “A SCADA via OPC applied to a pilot plant”, 2nd Brazilian Congress of R&D in Oil & Gas, 2003.

[5]http://ftp.softwaretoolbox.com/support/IBH/SiemensUSBMPI-IBHOPC_0208.pdf

[6]https://www.ibhsoftec.com/epages/63444704.sf/en_GB/?ObjectPath=/Shops/63444704/Products/3204