



# XPANGO DRIVER EDITOR USER MANUAL



# Index

---

<b>Introduction.....</b>	<b>4</b>
<b>1 Access to the XPANGO Driver Editor.....</b>	<b>5</b>
<b>2 Modbus and Siemens S7 templates .....</b>	<b>6</b>
2.1 MODBUS Template.....	6
2.1.1 General.....	6
2.1.2 Variables.....	8
2.1.2.1 Variable name .....	8
2.1.2.2 Section name.....	10
2.1.2.3 Area .....	10
2.1.2.4 Address.....	11
2.1.2.5 Bit of word.....	12
2.1.2.6 Data Type .....	13
2.1.2.7 Signed .....	13
2.1.2.8 Access Type .....	14
2.1.2.9 Measurement Unit .....	14
2.1.2.10 Scale.....	14
2.1.2.11 Minimum and Maximum Offset.....	15
2.1.2.12 Alarmed .....	15
2.1.2.13 Decode .....	16
2.1.2.14 Totalizer .....	16
2.1.2.15 Preselected.....	17
2.1.3 Digital Alarms .....	17
2.1.4 Analog Alarms .....	18
2.1.5 Virtual Variables .....	19
2.2 Modbus Template: completion example .....	21
2.3 Siemens S7 Template.....	25
2.3.1 General.....	27
2.3.2 Variables.....	28
2.3.2.1 Area .....	28
2.3.2.2 Data Block Number.....	28
2.3.2.3 Data Type .....	29
2.4 Siemens S7 Template: completion example.....	29
<b>3 XPANGO Driver Editor .....</b>	<b>33</b>
<b>4 Guidelines and suggestions .....</b>	<b>35</b>
<b>5 Contacts.....</b>	<b>36</b>

---

# Introduction

Alleantia's software platform is the unique Industrial Internet of Things solution for communicating with any industrial device, in few seconds, to collect data and send parameters, to use its information in different contexts and applications, transparently from the protocols and configuration of the devices to connect.

Alleantia Industrial IoT platform is based on the 'driver' concept, whereas an XPANGO file is created for every device type, depending on its specifications. XPANGO drivers describe the syntax and the semantics of the device information, sent and received, and are used by Alleantia IoT SCADA Server (ISS) and IoT Gateway Server (IGS) systems for communication via serial line RS232 – 422 – 485 and Ethernet, with different industrial protocols.

Many XPANGO drivers are available and already installed in any out-of-the-factory ISS and IGS systems, used for configuring devices and machines (see *IOT SCADA Server Installation and User Manual - Section 5.2.2.1*). Other drivers are found in Alleantia's *Library of Things* at <http://cloud.alleantia.com/info/products.zul>, where to download updates. Furthermore, it is possible to create proprietary drivers, especially necessary for machines and systems using *Programmable Logic Controllers (PLC)*. The XPANGO Editor supports such requirement.

Tutorial videos for using XPANGO Editor are available online on Vimeo: <https://vimeo.com/alleantia> and Youtube <https://youtube.com/alleantia>.

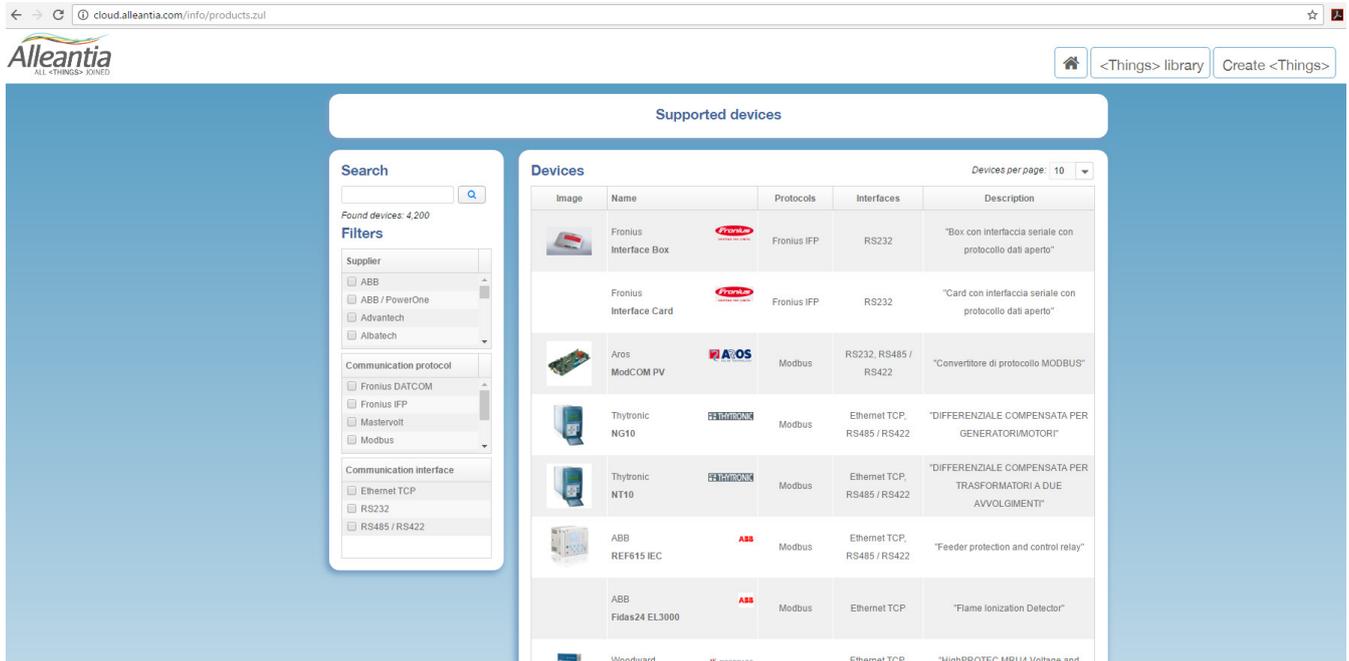


Figure 1 – Alleantia's Library of Things available at <http://cloud.alleantia.com/info/products.zul>

# 1 Access to the XPANGO Driver Editor

To access to the XPANGO Driver editor, go to <http://cloud.alleantia.com> and sign in, using your email and password or social network accounts (Facebook, LinkedIn and Twitter). If you sign in with email and password, fill in all mandatory fields, accept the privacy policy and click **Sign in**. Then, you will receive email with confirmation.

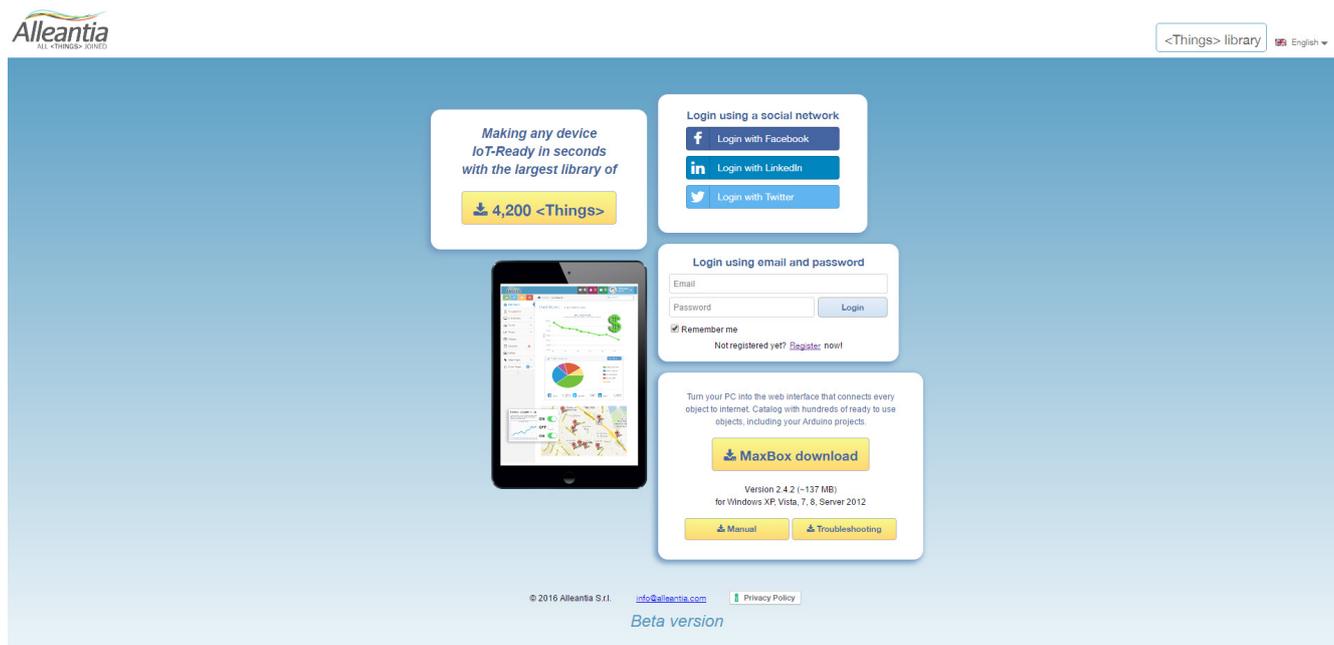


Figure 2 – Alleantia Cloud access page

After that, login to the XPANGO Driver Editor site, go to “Create <Things>” page, where you can create driver for any device. At this link <http://cloud.alleantia.com/xmod/convert.zul> it is possible to download Modbus and Siemens S7 templates in Excel format (check for updates on added support).

# 2 Modbus and Siemens S7 templates

The templates are preset spreadsheets, which simplify the work in creating an XPANGO driver from information taken from the device memory map. Usually, device manufacturers provide them in the configuration manual. PLC programmers shall provide these as part of their activity.

The spreadsheet template structure is simple, intuitive and fit for users with limited experience and education, and do not require programming skills. Once you have understood the characteristics of every data type in the device memory map, all you should do is properly copy information address to the template.

On the Editor page, you can download two types of templates: the Modbus template and the Siemens S7 template.

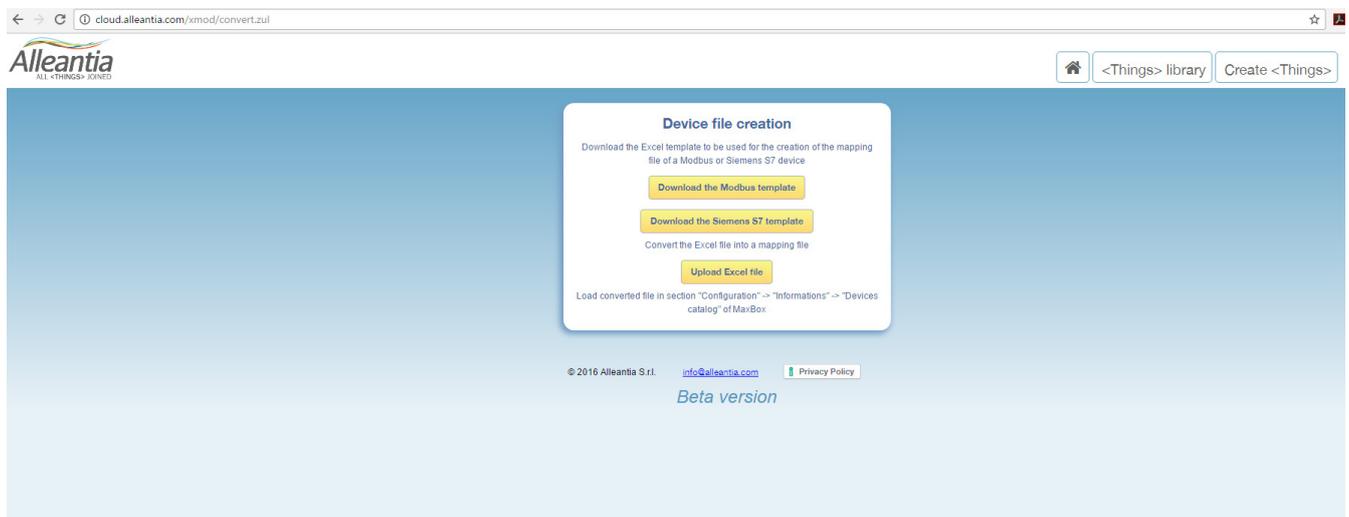


Figure 3 – Alleantia's XPANGO Driver Editor available at <http://cloud.alleantia.com/xmod/convert.zul>

The two templates differ in the way manufacturers provide the communication addresses. The Modbus communication protocol is a de facto standard in the industrial communication and is the most widespread connection language in the world. The Siemens S7 protocol was developed by Siemens specifically for communication with its PLCs.

## 2.1 MODBUS Template

This template should be used to get the XPANGO driver of any device regardless its type or a specific manufacturer.

### 2.1.1 General

First, after opening the file, fill in the “*General*” tab.

# 2 Modbus and Siemens S7 templates

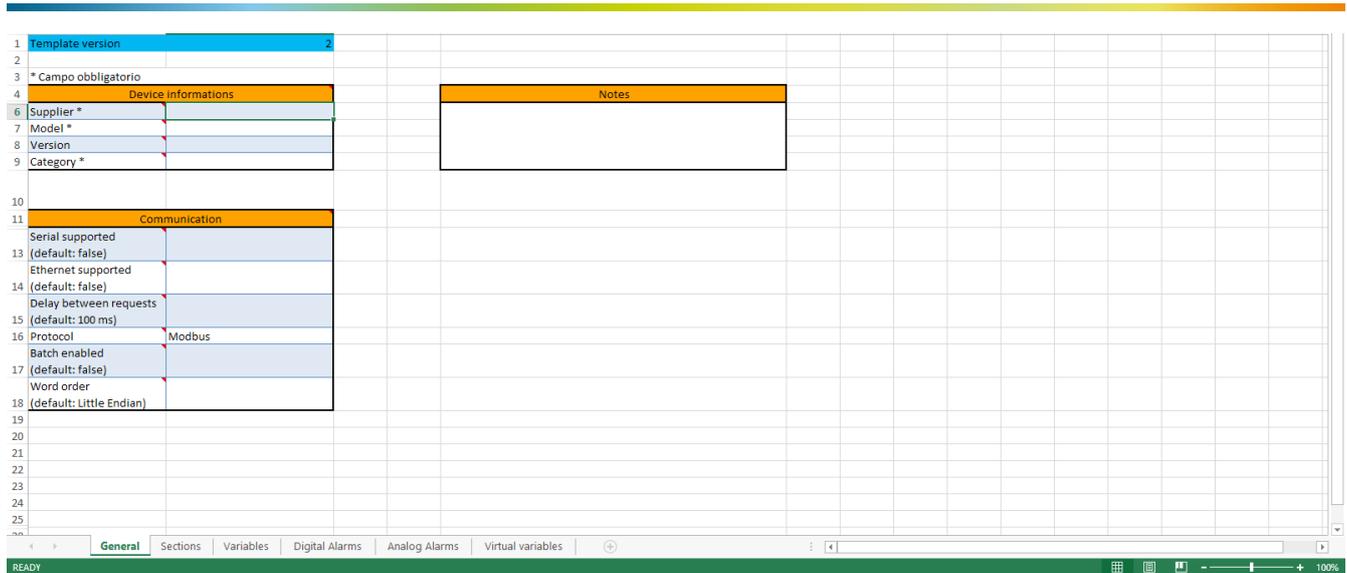


Figure 4 – “General” Modbus template tab

This tab contains 3 tables: *Device Informations*, *Communication* and *Notes*.

In the *Device Informations* table, insert the information of the device you want to get the driver for. Fields marked with \* are mandatory. Insert the manufacturer name, device model and choose a category from dropdown list (B9 cell).

In the *Communication* table insert the device’s type of communication. In the first row specify whether the device is equipped with a serial port RS485, RS232 or not. If it has, select “TRUE” from the dropdown list. If the device is not equipped with a serial port, select “FALSE” or just leave the field empty as “FALSE” is the default value.

**Be sure to select the right communication port for the device as several device models are similar and differ for its communication capability.**

This also applies to the second row of the Communication table. Specify whether the device is equipped with Ethernet port. Select “TRUE” if it has or leave empty if it does not.

In the third row set the delay time between requests to device. Leaving the field empty, you set the default value at 100 ms.

**Non-standard polling frequency should be carefully defined for the device, in consideration of number of variables to collect, variables update and its use, to avoid gateway overload as well as device overload.**

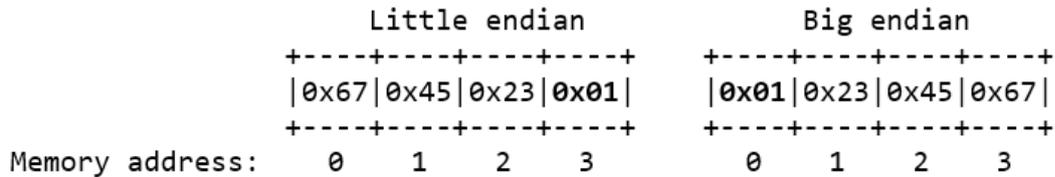
In the last but one row enable or disable the *Bach data Transmission* function, that is transmission of continuous blocks containing more than one data. In this way, it is possible to send more addresses and read or write them in a single transmission. This procedure, so called *Bach Optimization*, increases the speed of communication when sending only one data at a time.

Finally, fill in the table specifying the *Word order*, essential parameter for computers to store data with a size larger than one byte (1 byte = 8 bit). Specifying this field is important if *double word* or *quad word* data type is sent (1 Word = 2 bytes = 16 bit).

# 2 Modbus and Siemens S7 templates

From the dropdown list, you can select either *Big Endian* order or *Little Endian* order. In *Big Endian*, the most significant byte is stored first and the least significant byte is stored at last place. *Little Endian* works exactly on the contrary.

For example, hexadecimal number 0x01234567 in Double Word (32 bit) format will be represented in two following formats (two bold digits represent the most significant byte):



As can be seen in the above picture, *Little Endian* order stores the hexadecimal number with the most significant numbers (0x01) in the higher address (3), meanwhile *Big Endian* order does just the opposite; the most significant numbers are in the address 0, while those least significant numbers (23, 45, 67) will be stored in the lowest memory addresses.

Among these two orders, *Little Endian* is the most used. Verify in the device technical manuals which order has been chosen.

Information about type of the device and its components can be inserted in the table *Notes*, which is optional.

## 2.1.2 Variables

Once the “*General*” tab is filled in, move to “*Variables*” tab. During our communication with the device we can “ask” it about the state of all the variables that we are going to register in the following tab. First, every variable has an identification code represented in the ID column, which cannot be modified.

### 2.1.2.1 Variable name

In the *Variable name* column insert the variable name. First, the name should be unique. Indeed, if two variables in the same section have identical variable name, the mapping of XPANGO driver will not be successful.

# 2 Modbus and Siemens S7 templates

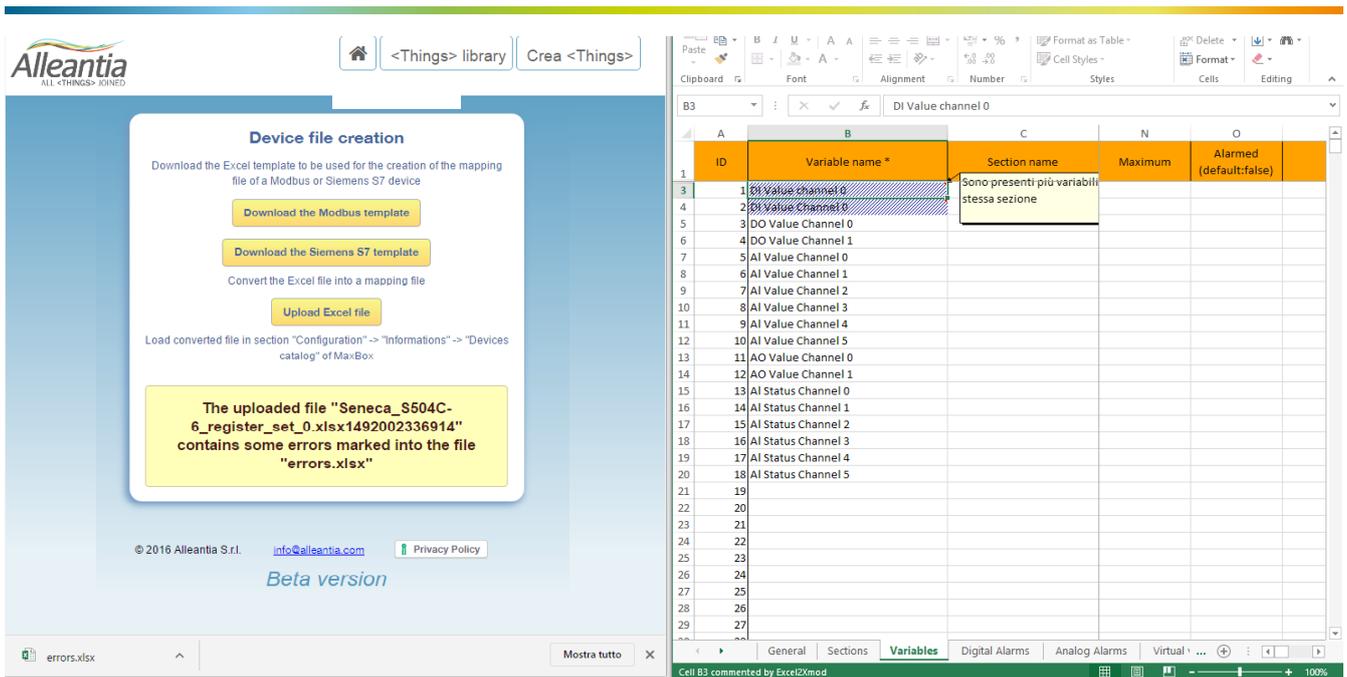


Figure 5 – Variables tab: error generated by multiple variables with the same name

If two variables have the same name, there are two ways to solve the problem.

The first one is to insert two different numbers or letters in the end of their names. The second is to insert two variables with the same name in different sections (see 2.2.2.2).

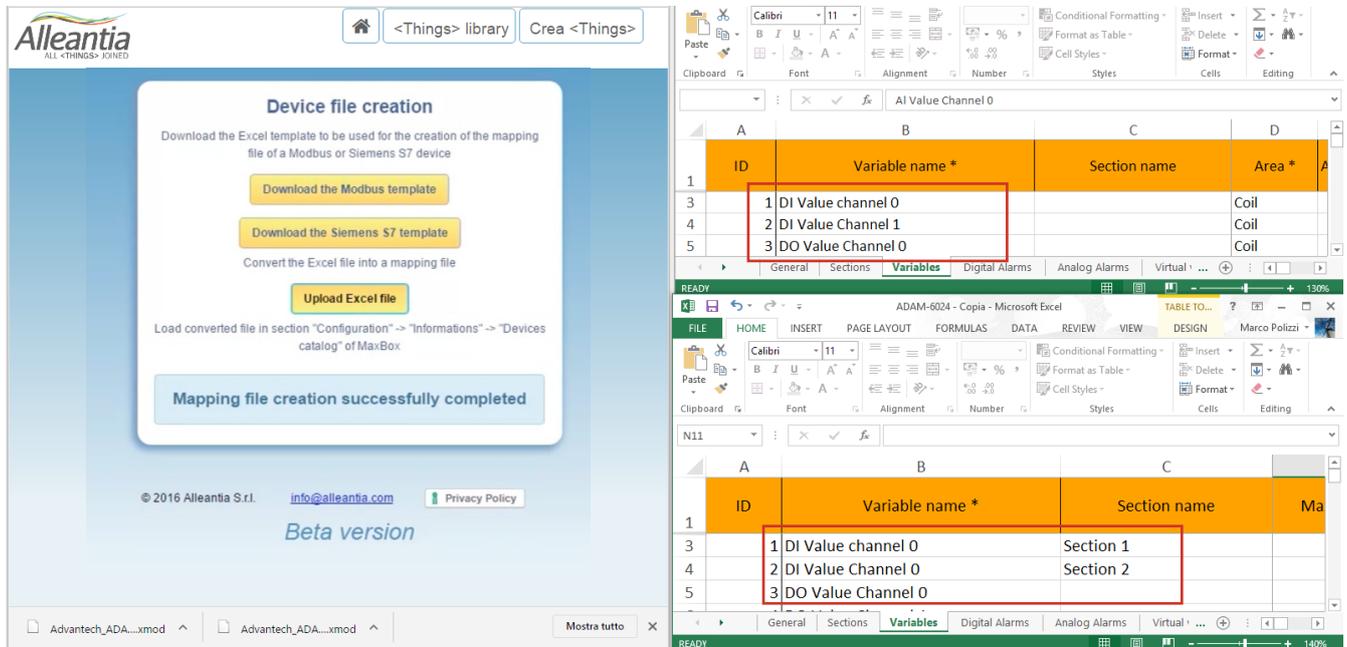


Figure 6 – Variables tab: resolving the error posed by the presence of more variables with the same name

The variable name should also clearly identifiable, as it will be displayed in the IoT-SCADA Server graphic interface once the XPANGO driver is uploaded and the device is configured.

# 2 Modbus and Siemens S7 templates

## 2.1.2.2 Section name

It is possible to group the variables in more sections specifying a *Section name* for each of them. Such subdivision will be displayed in the IoT-SCADA Server graphic interface, once the XPANGO driver is uploaded and the device is configured.

The sections grouping different variables shall not be inserted in the *Section name* column. First, you have to go to the *Sections tab*, which has two columns: ID and Section name. Once the section names are created, in the *Variables tab* a variable can be associated to a section, selecting it from the dropdown list. The dropdown list contains sections from the Sections tab, sorted by their IDs.

The variables association into sections is not mandatory for the mapping of XPANGO file. It may be useful for the user to manage numerous variables.

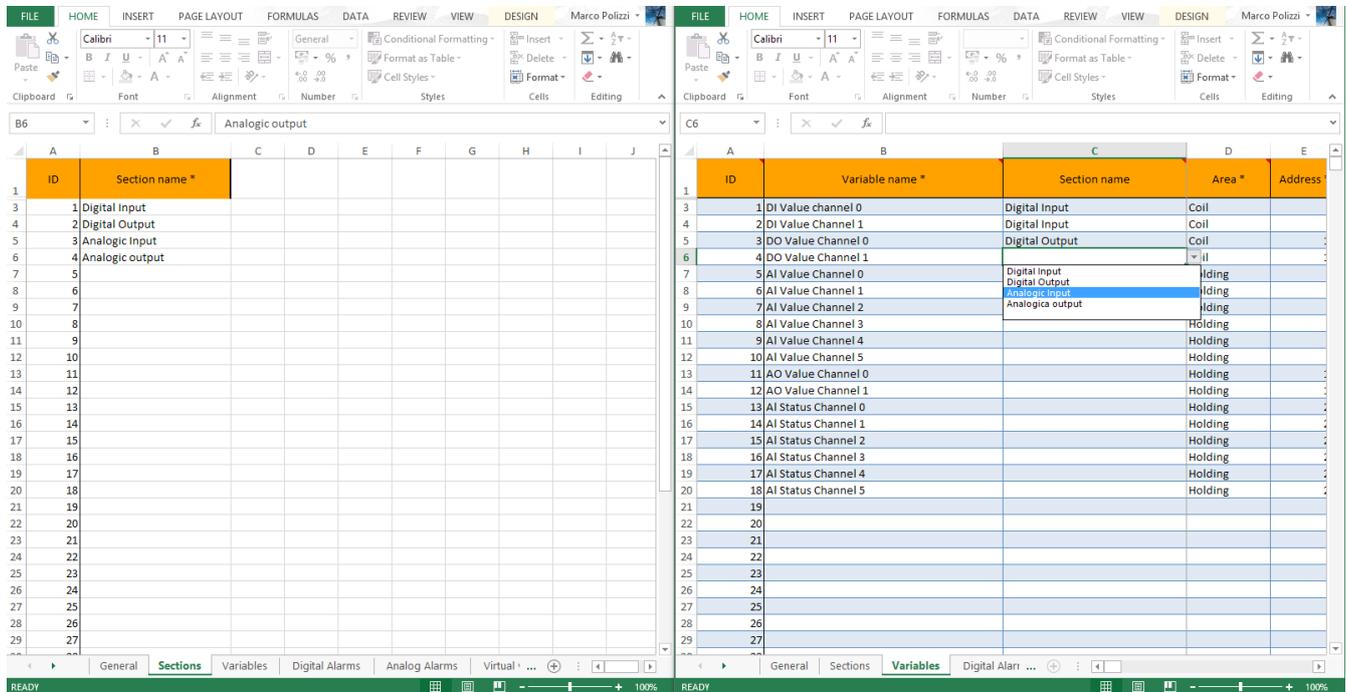


Figure 7 – Sections Tab of Modbus template

## 2.1.2.3 Area

In this column insert the memory area, or register, where the variable are inserted.

The Modbus protocol divides communication data in registers. A register to which a variable belongs depends on the type of information the variable carries.

There are 4 kinds of registers:

1. Coil (or Discrete Output);
2. Discrete Input;
3. Input Register;
4. Holding Register.

# 2 Modbus and Siemens S7 templates

**Coils** are 1 bit registers, used to control discrete outputs, and can operate both in read and write modes.

**Discrete Input** are also 1 bit registers, they store only input data, therefore are read only registers. Being the 1 bit registers, coils and discrete inputs contain only the Boolean data type.

**Input Registers** and **Holding Registers** are both 16 bit registers, thus able to contain data of one word size. The first type manage only inputs data, so are read only registers. The second type are universal registers as they can contain both input and output data, configuration data and many others. In addition, they are Read & Write registers.

Many manufacturers indicate the register data type in the Modbus tables. If this information is not available, by analysing its Modbus address it is possible to derive in what memory area the variable data are saved. Most of time, the manufacturers implement the 'Modicon rule' developed by the inventors of Modbus protocol. Such rules provide a range of addresses for every register type. Thus, by understanding in which addresses intervals the Modbus Address is inserted, you derive the variable what register types.

Modbus Data types e Address Space			
Register	Access	Space Occupied	Address
<i>Coil</i>	Read & Write	1 bit	00001 - 09999
<i>Discrete Input</i>	Read Only	1 bit	10001 - 19999
<i>Input Register</i>	Read Only	16 bits	30001 - 39999
<i>Holding Register</i>	Read & Write	16 bits	40001 - 49999

Table 1 – Addresses range of different Modbus Registers, access mode, type of supported data and their dimensions

**Area field should be filled for every variable in the template, otherwise XPANGO Drive Editor will not complete the mapping.**

## 2.1.2.4 Address

In the *Address* column, insert the Modbus address for the variable that you intend to read or modify. Every variable shall have a different address. There is no universal mode to provide the Modbus addresses. Some manufacturers implement the Modicon rule, some provide the address in hex format, others implement their own modes. Devices' technical manuals usually provide the necessary information.

Modbus Address can have values between 0 and 65535, so between 0 and FFFF in hex format. The address value in the template shall be only in decimal format only. If the manufacturer indicates the addresses in hexadecimal format, it is necessary to convert the address values. By the way, remember that the address first digit indicates the register type to which the variable belongs (0=Coil, 1= Discrete Input, 3=Input, 4=Holding, see Section 2.1.2.3).

Address are represented by numbers from starting from first, with one or more zeros. For example, Figure 8

## 2 Modbus and Siemens S7 templates

shows the Modbus Map for Advantech ADAM-6024 device. The variable *DI Value Channel 0* has address 00001, so is a variable belonging to Coil (0X) register, and has address 1. Meanwhile the variable *AI Value Channel 0* has address 40001; so, it is a variable that belongs to Holding (4X) and has address 1. Select respectively Coil and Holding in the Area column and in the Address column insert 1 for each variable.

### B.2.4 ADAM-6024 12-Ch Universal I/O Module

#### Address (0X):

Address (0X)	Channel	Description	Attribute
00001	0	DI Value	Read
00002	1		Read

00017	0	DO Value	R/W
00018	1		R/W

#### Address (4X):

Address (4X)	Channel	Description	Attribute
40001	0	AI Value	Read
40002	1		Read
40003	2		Read
40004	3		Read
40005	4		Read
40006	5		Read

Figure 8 – Modbus Map of device ADAM-6024 of Advantech

### 2.1.2.5 Bit of word

The field *bit of word* defines the register bit in which data will be stored. This information is necessary if, for example, you want to represent Boolean data type (1 bit) in Holding or Input Registers, or in 16 bit registers. Indeed, there is no sense to specify the stored data's bit of word inside Coil or Discrete Input registers (in this case, the bit of word will be always 0, or the first and only bit!).

As you remember, a Holding register can host a whole word, the first bit is 0, while the last one is 15. In case you want to insert more Boolean variables inside a 16-bit register, these will have the same *Address* but will differ by their *bit of word* value.

The sample extract from Modbus map of Advantech ADAM-6224 is given below. State of digital inputs (*DI Event Status*) will use an address of Holding type for every digital input (Channel from 0 to 3). From the note, provided by the manufacturer, you understand how the first bit of word (bit 0) is responsible for *Unreliable DI Value* boolean type, the second bit of word (bit 1) is responsible for *Safety Value Triggered* Boolean and so on. In Figure 9 you may see an example of how to fill in the template.

## 2 Modbus and Siemens S7 templates

ID	Variable name *	Section name	Area *	Address *	Bit of word	Data type *	Signed (default: false)	Access type (default: Read)
1	DI Event Status Channel 0 - UAFT Timeout		Holding	111	0	Boolean		Read
2	DI Event Status Channel 0 - Safety Value Triggered		Holding	111	1	Boolean		Read
3	DI Event Status Channel 0 - Startup Value Triggered		Holding	111	2	Boolean		Read

### Address (0X):

Address (0X)	Channel	Description	Attribute
<b>Note:</b>			
1. The value definition of DI Event Status			
	Bit	Description	
	0	Unreliable DI value (UART Timeout)	
	1	Safety Value triggered	
	2	Startup Value triggered	
40111	0	DI Event Status	Read
40112	1		Read
40113	2		Read
40114	3		Read

Figure 9 – Modbus Template of ADAM-6024 (on top) and the extract from Modbus map with data to fill it in

### 2.1.2.6 Data Type

*Data Type* column specify variable data type. From the in-cell dropdown list you can select following representation formats: *Boolean*, *Integer*, *Double Integer*, *Quad Integer*, *Float*, *Double Float*, *BCD*, *Double BCD*.

*Coil* or *Discrete Input* registers can collect only one bit, *Data Type* can only be *Boolean* since only one bit is enough for such representation. *Holding* registers can host *Integer* or *Float* data types since they have a 16-bit size. *Double Integer* and *Double Float* data types are allocated into 2 registers of 16-bit memory (e.g. *Holding*) while *Quad Integer* data type on 4 registers (16 bit each).

### 2.1.2.7 Signed

The *Signed* field can take only two values, *TRUE* or *FALSE*, which can be set from the in-cell dropdown list. If the number that will represent the variable is a number with sign, the *Signed* field will take the *TRUE* value, otherwise you may select *FALSE* value or leave it empty (Default: *FALSE*).

Please notice that the field shall be kept *FALSE* for variables in *Coil* or *Discrete Input* registers (variables in such registers cannot be represented with sign since this requires one bit of memory, that would use the whole register). It is necessary to set value to *TRUE* for *Float* variable types since these are signed numbers.

# 2 Modbus and Siemens S7 templates

## 2.1.2.8 Access Type

*Access Type* column define the access type to the variable. There are three types of access permissions: *Write*, *Read* and *Read/Write*. *Read* and *Write* access types grant the ability to read or modify a variable value, respectively. *Read/Write* access grants the ability both to read and modify a value.

First, the *Access Type* is identified by the register of inserted data (see Section 2.1.2.3). If data is inserted in R/W register, it will be for you to decide whether enable or not to read and/or modify a variable value. The access attribute to give to the variable is usually specified in the manufacturers' Modbus map.

## 2.1.2.9 Measurement Unit

This column is optional for a successful XPANGO driver mapping, the measurement unit of the variable concerned. The measurement unit specified here will be then displayed in the IoT-SCADA interface together with the monitored variable value. For example, Volt, Ampere and Watt are used for voltage, current and power (depending on the power). Often the manufacturer indicates measurement units, but it is not necessary to respect them.

For example, the device can provide you with the value of exported active power from an inverter in W. If a device has power of many kW, it might happen that a too large value is displayed, which is not easy to display. If you want the output power from inverter expressed in kW, specify this measurement unit in the *Measurement Unit* column and a multiplication factor equal to 0,001 in the *Scale* column (see Section 2.1.2.9).

## 2.1.2.10 Scale

In this column specify the multiplication factors to apply to the variable value before showing it to the user. It is necessary to fill in this field if you want the measured data from the device to be displayed in a measurement unit set by you and not by the manufacturer.

In Figure 10 you may see an example of extract from Modbus map of Seneca S504C-6-MOD-MID energy meter.

Parametro	Cod. di funzione (Hex)	Segno	INTERO			IEEE		
			Registro (Hex)	Word	U. M.	Registro (Hex)	Word	U. M.
<b>VALORI DEI CONTATORI TOTALI</b>								
+kWh1 • Energia attiva importata fase 1	03 / 04		0100	3	0.1 Wh	1100	2	Wh
+kWh2 • Energia attiva importata fase 2	03 / 04		0103	3	0.1 Wh	1102	2	Wh
+kWh3 • Energia attiva importata fase 3	03 / 04		0106	3	0.1 Wh	1104	2	Wh
+kWhΣ • Energia attiva importata di sistema	03 / 04		0109	3	0.1 Wh	1106	2	Wh

Figure 10 – Modbus map of Seneca S504C-6-MOD-MID energy meter

## 2 Modbus and Siemens S7 templates

Looking at the values of provided addresses according to the IEEE standard, it can be seen how Imported active energy in 3 phases is expressed in Wh. To display the variable in kWh, insert multiplication factor equal to 0,001. To display in MWh insert multiplication factor equal to 1000.

ID	Variable name *	Section name	Measurement unit	Scale
31	+kWh1 • Energia attiva importata fase 1		Wh	kWh 0,001
32	+kWh2 • Energia attiva importata fase 2		Wh	kWh 0,001
33	+kWh3 • energia attiva importata fase 3		Wh	MW 1.000,0

Figure 11 – Modbus template of Seneca S504C-6-MOD-MID energy meter: measurement unit and applied scaling function

### 2.1.2.11 Minimum and Maximum Offset

The factors inserted in the *Offset* column are added to the measured variable value of the device before showing it to the user. Fill in these fields when maximum and minimum values that the variable will take are known. These two values will be displayed in the IoT-SCADA user interface, on the row corresponding to the variable under the *min* and *max* headings.

To get an idea of how it can be useful to set these parameters, here is an example.

Imagine that you can communicate with a pyranometer whose output signal is a current proportional to the irradiation, measured by the device. At night, with irradiation equal to 0 W/m<sup>2</sup> the current output from the pyranometer will be 4 mA, meanwhile the current corresponding to the maximum measurable irradiation (e. g. 1000 W/m<sup>2</sup>) is equal to 20 mA.

To enable the user to read directly the irradiation value in W/m<sup>2</sup> and the output current value of the pyranometer, implement the following relation:

$$irradiation = \frac{(I - 4)}{16} * 1000 \quad \left[ \frac{W}{m^2} \right]$$

In this case, to display the irradiation value set a *Scale* factor equal to 62,5 (1000/16) and *Offset* factor equal to -4. Minimum and maximum values that the variable could take are 0 W/m<sup>2</sup> and 1000 W/m<sup>2</sup>, respectively.

Please note that it is possible to set such parameters directly from the IOT SCADA Server interface (see *IOT SCADA Server Installation and User Manual* - Section 5.2.3).

### 2.1.2.12 Alarmed

The fields of *Alarmed* column are Boolean, having two values TRUE and FALSE. The field will be set as TRUE if the variable concerned has an alarm, otherwise set it as FALSE or leave empty.

After setting TRUE in the *Alarmed* cell of a Boolean data, the variable name will be copied to the *Digital Alarms* tab; for all other data types, it will be copied to the *Analog Alarms* tab. For further details refer to Sections 2.1.2 and 2.1.4 of this manual.

# 2 Modbus and Siemens S7 templates

## 2.1.2.13 Decode

In the *Decode* column it is possible to match the value obtained by the variable and a numerical or alphanumeric value to display. To do it, insert in the cell Variable Name 1 = User Value 1, Variable Name 2 = User Value = 2 and so on. Every field should be separated by a comma. In this way when the variable will obtain the specified value, the corresponding value or message will be displayed.

## 2.1.2.14 Totalizer

The fields in the *Totalizer* column are set TRUE if the variables values will be continuously increasing, and you want to create reports for daily, monthly analysis of such measures. For example, it makes sense to activate such function for monotone growth measures like produced energy, whereas it is useless to apply it to produced power measure, since it can increase or decrease during the machine operations.

Besides the report for a single measure, it is possible to compare the difference between two or more totalizer measures during the reporting period or to compare the same measure with separate reporting periods. For further details refer to Section 6.5 of *IOT SCADA Server Installation and User Manual*.

For example, in the *Totalizer* column set TRUE value to the variables *System Imported energy* and *System Exported energy* in the Modbus template of Seneca S504C-6-MOD-MID energy meter.

ID	Variable name *	Section name	Totalizer (default: false)	Preselected (default: true)
34	+kWhΣ • System imported active energy 1		TRUE	TRUE
35	-kWh1 • Phase 1 exported active energy 1			TRUE
36	-kWh2 • Phase 2 exported active energy 1			TRUE
37	-kWh3 • Phase 3 exported active energy 1			TRUE
38	-kWhΣ • System exported active energy 1		TRUE	TRUE

Figure 12 – Modbus template of S504C-6-MOD-MID energy counter of Seneca: totalizer variables

In the IoT SCADA Server interface, go to Report tab, click  button in *Parameters* table, select the variables of device to make a report (see *IOT SCADA Server Installation and User Manual – Section 6.5*). You can only select the variables with the *Totalizer* field set as TRUE.

# 2 Modbus and Siemens S7 templates

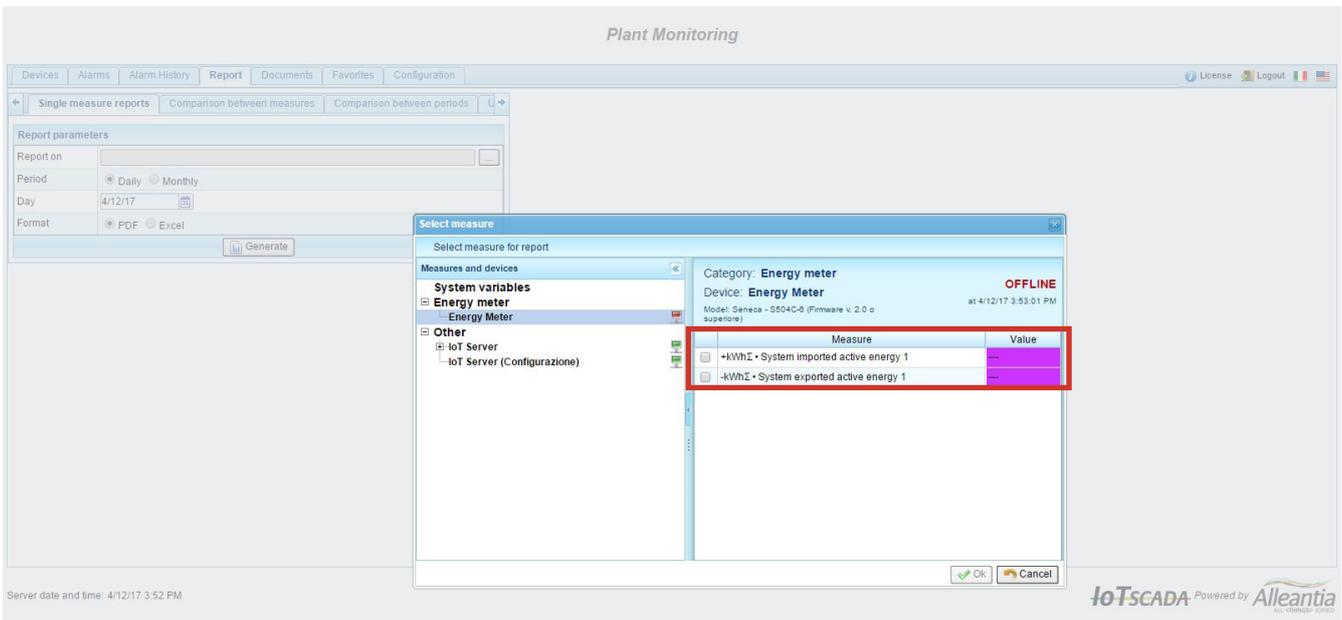


Figure 13 – IoT SCADA Server interface: list of the variables to generate a report

## 2.1.2.15 Preselected

While monitoring the device, it might be useful to hide some variables. To do so, set the *Preselected* cell to FALSE. To show these variables again, set the field to TRUE. This last operation can be done directly from the IoT-SCADA Server interface (see *IOT SCADA Server Installation and User Manual – Section 5.2.3*). Please note that the default value of *Preselected* field is TRUE.

## 2.1.3 Digital Alarms

In *Digital Alarms* tab, for all Boolean variables the field *Alarmed* of *General* tab should be set to TRUE (see Section 2.1.2.12).

ID	Variable name *	Section name	Alarmed (default:false)
1			
3	1 DI Value channel 0	Digital Input	TRUE
4	2 DI Value Channel 1	Digital Input	
5	3 DO Value Channel 0	Digital Output	TRUE
6	4 DO Value Channel 1	Digital Output	
7	5 AI Value Channel 0	Analogic Input	
8	6 AI Value Channel 1	Analogic Input	
9	7 AI Value Channel 2	Analogic Input	
10	8 AI Value Channel 3	Analogic Input	
11	9 AI Value Channel 4	Analogic Input	
12	10 AI Value Channel 5	Analogic Input	
13	11 AO Value Channel 0	Analogic Output	
14	12 AO Value Channel 1	Analogic Output	
15	13 AI Status Channel 0		
16	14 AI Status Channel 1		
17	15 AI Status Channel 2		
18	16 AI Status Channel 3		
19	17 AI Status Channel 4		
20	18 AI Status Channel 5		
21	19		
22	20		
23	21		
24	22		
25	23		
26	24		
27	25		
28	26		
29	27		

ID	Variable name	Section name	Alarm condition *
1			
3	1 DI Value channel 0	Digital Input	
4	2		
5	3 DO Value Channel 0	Digital Output	
6	4		
7	5		
8	6		
9	7		
10	8		
11	9		
12	10		
13	11		
14	12		
15	13		
16	14		
17	15		
18	16		
19	17		
20	18		
21	19		
22	20		
23	21		
24	22		
25	23		
26	24		
27	25		
28	26		
29	27		

Figure 14 –Variables tab (right) and Digital Alarms (left) of the Modbus template

## 2 Modbus and Siemens S7 templates

---

At this point, *ID number* fields will be highlighted in red, while *Variable name* and *Section name* fields of *Digital Alarms* tab will be filled in automatically, in the rows corresponding to the selected variable IDs.

Digital alarms are Boolean type, having only two values TRUE and FALSE. In that regard, set the field of *Alarm Condition* column to TRUE or FALSE.

Giving the example of the Modbus template of ADAM – 6024, Advantech’s device, we report with an alarm the situation when Digital Input 0 is inoperative or inactive. In this case, we should set *Alarm Condition* field of *DI Value Channel 0* to FALSE. In doing so, the user will be warned in case the chosen Boolean variable has a value equal to zero.

Then, fill in *Delay ON* and *Delay OFF* fields.

Insert time interval in *Delay ON* field (in milliseconds), between alarm occurrence and its notification. For example, inserting value 100 in *Delay ON* field, from the moment when the monitored variable value will be assigned the value set in the *Alarm Condition* column and the moment when the alarm is notified, will pass exactly 100 ms. This feature is useful to avoid the “false alarms” notification. If you decide to set this time to 0 ms, the user will receive a warning whenever the variable value is equal to the one set in *Alarm Condition*, even if it is temporary and not a real anomaly. Setting “reasonable” time of *Delay ON*, the device will inform you only about persisting malfunctions.

On the contrary, to be sure that an anomaly is resolved, it is necessary to consider a certain time interval between the moment when the monitored variable value returns to its normal value (different from the one set in *Alarm Condition*) and the moment when the alarm notification is cancelled. Insert this time interval (in milliseconds) within *Delay OFF* column.

Finally, in *Alarm Description* column, it is possible to insert a string to describe the type of alarm notification and the causes that created anomaly. The message in the cell will be displayed in IoT-SCADA Server interface.

### 2.1.4 Analog Alarms

Unlike the alarms described in the previous section, analog alarms have a more sophisticated logic. An analog alarm compares two values through a logic operator (=, !=, >, <, etc.) and is notified if the specified condition is achieved. Since the variables with 0 or 1 values are compared, in *Analog Alarms* tab will appear all non-Boolean variables with *Alarmed* field set to TRUE.

# 2 Modbus and Siemens S7 templates

ID	Variable name *	Section name	Alarmed (default: false)
1	DI Value Channel 0	Digital Input	TRUE
2	DI Value Channel 1	Digital Input	TRUE
3	DO Value Channel 0	Digital Output	TRUE
4	DO Value Channel 1	Digital Output	TRUE
5	AI Value Channel 0	Analogic Input	TRUE
6	AI Value Channel 1	Analogic Input	TRUE
7	AI Value Channel 2	Analogic Input	
8	AI Value Channel 3	Analogic Input	
9	AI Value Channel 4	Analogic Input	
10	AI Value Channel 5	Analogic Input	
11	AO Value Channel 0	Analogic Output	
12	AO Value Channel 1	Analogic Output	
13	AI Status Channel 0		
14	AI Status Channel 1		
15	AI Status Channel 2		
16	AI Status Channel 3		
17	AI Status Channel 4		
18	AI Status Channel 5		

ID	Variable name	Section name	Alarm operator *	Reference value *	Delay ON (default: 0 ms)	Delay OFF (default: 0 ms)	Alarm description *
1							
2							
3							
4							
5	AI Value Channel 0	Analogic Input		10,0			
6	AI Value Channel 1	Analogic Input					
7							
8							
9							
10							
11							
12							
13							

Figure 15 – Variables tab (right) and Analog Alarms (left) of Modbus template

As in the case of Digital Alarms, also here *Variable name* and *Section name* are filled in automatically and IDs concerned are highlighted in red in *General tab*.

In this case, set the alarm condition, modifying fields of *Alarm Operator* and *Reference Value* columns. In first field, select an operator from the in-cell dropdown list, while in second field insert the value that will be compared with the monitored variable.

For example, if you want an alarm notification when *AI Value Channel 0* has a value different from 10, choose the “! =” symbol in *Alarm Operator* column and insert value 10 in *Reference Value* column.

ID	Variable name	Section name	Alarm operator *	Reference value *	Delay ON (default: 0 ms)	Delay OFF (default: 0 ms)	Alarm description *
1							
2							
3							
4							
5							
6							
7	AI Value Channel 0	Analogic Input	<=	10,0			
8	AI Value Channel 1	Analogic Input	! =				
9			>=				
10			>				
11							
12							
13							

Figure 16 – Analog Alarms tab: types of Alarm operators

Likewise, for *Analog Alarms* it is possible to set time of *Delay ON*, *Delay OFF* and display a string with the description of alarm (see Section 2.1.3).

## 2.1.5 Virtual Variables

In *Virtual Variables* tab, it is possible to create customized variables, based on device’s standard measures defined in the *Variables* tab.

Apart from the virtual variable name and the section it belongs to, insert the variable’s data type. From the

## 2 Modbus and Siemens S7 templates

dropdown list of *Data Type* column, it is possible to choose from two formats: Boolean and numerical.

In *Expression* column, insert numerical expression to achieve the virtual variable, based on the device standard variables. Insert the expression, referring to the variables from *Variables* tab with the expression “\$ID variable”. Do not put “=” sign before the formula (different from Excel formulas). The rest of the columns have the same characteristics as the ones in *Variables* tab (see Section 2.1.2).

For example, you want to verify that *System Active Power* variable provided by Seneca S504C-6-MOD-MID energy meter is correctly calculated.

For this, create a virtual variable called *Calculated active power* and give it the following formula:

$$\text{Calculated active power} = I * V * \cos \varphi$$

where *I* is system current, *V* is system voltage and  $\cos \varphi$  is system power factor.

The formula will be implemented in the cell of *Expression* column in the following way:

$$\$7 * \$12 * \$16$$

Looking at the *Variables* tab, you can notice how ID 7, 12 e 16 correspond to the variables ( $V\Sigma$  • System voltage), ( $A\Sigma$  • System current) and ( $PF\Sigma$  • System power factor).

ID	Variable name *	Section name	Maximum
1	V1 • L-N voltage phase 1		
2	V2 • L-N voltage phase 2		
3	V3 • L-N voltage phase 3		
4	V12 • L-L voltage line 12		
5	V23 • L-L voltage line 23		
6	V31 • L-L voltage line 31		
7	VΣ • System voltage		
8	A1 • Phase 1 current		
9	A2 • Phase 2 current		
10	A3 • Phase 3 current		
11	AN • Neutral current		
12	AΣ • System current		
13	PF1 • Phase 1 power factor		
14	PF2 • Phase 2 power factor		
15	PF3 • Phase 3 power factor		
16	PFΣ • System power factor		
17	PI1 • Phase 1 active power		
18	PI2 • Phase 2 active power		
19	PI3 • Phase 3 active power		
20	PIΣ • System active power		
21	SI1 • Phase 1 apparent power		
22	SI2 • Phase 2 apparent power		
23	SI3 • Phase 3 apparent power		
24	SIΣ • System apparent power		
25	Q1 • Phase 1 reactive power		
26	Q2 • Phase 2 reactive power		
27	Q3 • Phase 3 reactive power		

ID	Variable name *	Expression *
10001	Potenza Attiva Calcolata	\$7*\$12*\$16
10002		
10003		
10004		
10005		
10006		
10007		
10008		
10009		
10010		
10011		
10012		
10013		
10014		
10015		
10016		
10017		
10018		
10019		
10020		
10021		
10022		
10023		
10024		
10025		
10026		
10027		

Figure 17 – Virtual Variables tab: implementing of the analytic expression of the virtual variable

If the value of Active calculated power and the one provided by the device correspond, the energy counter is working correctly.

The created virtual variables will be displayed together with the other variables in the IoT SCADA Server interface.

# 2 Modbus and Siemens S7 templates

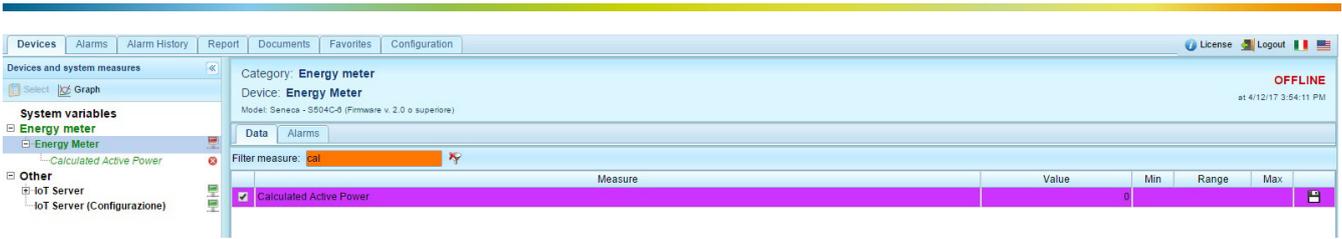


Figure 18 – Created virtual variable in the IoT Scada Server interface

## 2.2 Modbus Template: completion example

We provide in the following a full example of Modbus template completion necessary to create the XPANGO driver of Seneca S504C energy meter.

First, download the device's Modbus map from the company's official website here: <https://www.seneca.it/en/>. Go to the *Downloads* tab and download “**Modbus - LAN communication manual.pdf**”.



Figure 19 - Official website of Seneca where you can download the Modbus map of S504C-6-MOD-MID energy counter

The device's Modbus map, provided by the manufacturer, include several tables.

The manufacturer provides two register sets, numbered with 0 and 1, respectively. The register set 1 is available only for counters with integrated Modbus, counters with integrated Ethernet or RS485 modules with firmware release 2.00, while the Set 0 is available for any device. In the example, we will use the register set 0 as it is “universal”.

# 2 Modbus and Siemens S7 templates

## 6. Register set 0

### 6.1 READING registers (Function code \$01/\$03/\$04)

Parameter	F. code (Hex)	Sign	INTEGER			IEEE		
			Register (Hex)	Words	M.U.	Register (Hex)	Words	M.U.
<b>REAL TIME VALUES</b>								
V1 • L-N voltage phase 1	03/04		0000	2	mV	1000	2	V
V2 • L-N voltage phase 2	03/04		0002	2	mV	1002	2	V
V3 • L-N voltage phase 3	03/04		0004	2	mV	1004	2	V
V12 • L-L voltage line 12	03/04		0006	2	mV	1006	2	V
V23 • L-L voltage line 23	03/04		0008	2	mV	1008	2	V
V31 • L-L voltage line 31	03/04		000A	2	mV	100A	2	V
V $\Sigma$ • System voltage	03/04		000C	2	mV	100C	2	V
A1 • Phase 1 current	03/04	X	000E	2	mA	100E	2	A
A2 • Phase 2 current	03/04	X	0010	2	mA	1010	2	A
A3 • Phase 3 current	03/04	X	0012	2	mA	1012	2	A
AN • Neutral current	03/04	X	0014	2	mA	1014	2	A
A $\Sigma$ • System current	03/04	X	0016	2	mA	1016	2	A
PF1 • Phase 1 power factor	03/04	X	0018	1	-	1018	2	-
PF2 • Phase 2 power factor	03/04	X	0019	1	-	101A	2	-
PF3 • Phase 3 power factor	03/04	X	001A	1	-	101C	2	-
PF $\Sigma$ • System power factor	03/04	X	001B	1	-	101E	2	-

Figure 20 – Modbus map extract of Seneca’s energy counter S504C-6-MOD-MID (Register set 0)

**Parameter:** contains names of the variables to be read.

**F. code (Hex):** function code in hex format. It defines the command type: reading (Function code \$01/\$03/\$04) or writing (Function code \$10).

**Sign:** if this column is checked, the read register value can have negative sign. If not checked, the read register can have only positive sign.

The manufacturer provides two categories of Modbus addresses. The first category represents values in integers, the second uses floating-point representation or 32-bit/sec float according to the IEEE standard. We choose the second category since it is a universal standard representation.

Both categories are divided in 3 sub columns:

**Register (Hex):** register address in hex format.

**Words:** number of word to be read / written for the register (length, 1 word=16 bit).

**M.U.:** measuring unit of parameter.

Referring to S504C-6-MOD-MID energy counter with built-in RS485 port, we will fill in the Modbus template.

In *General* tab, insert the model and communication mode, as in Table 2.

## 2 Modbus and Siemens S7 templates

Device informations	
Supplier *	Seneca
Model *	S504C-6
Version	Firmware v. Precedente a 2.0
Category *	Energy meter
Communication	
Serial supported (default: false)	VERO
Ethernet supported (default: false)	VERO
Delay between requests (default: 100 ms)	
Protocol	Modbus
Batch enabled (default: false)	
Word order (default: Little Endian)	

Table 2 - Completion example of General tab of the Modbus template of S504C-6-MOD-MID Seneca's energy counter

It is possible to communicate with S504C-5-MOD-MID energy counter only via serial port (Serial Supported=TRUE), lacking Ethernet port for LAN communication (Ethernet Supported=FALSE). In our case, *Delay between requests* field was not filled in, however, the default value is 100 ms. Batch function was disabled (Batch enabled=FALSE).

After that we can fill in *Variables* tab, starting from the first variable in the map "V1 • L-N voltage phase 1".

As a variable name, we can copy the one from *Parameter* column of Modbus map.

The register area is not clearly specified, however the variable is stored at a 2-word register (32 bit), so it cannot be Coil or Discrete Input register since their size is 1 bit. Consequently, it may be Holding or Input register.

Recalling that the first register type is Read/Write, while the second contains only Read data type (see Section 2.1.2.3), we can conclude that its Input Register function code is equal to \$03/\$04, encoding for Read register. Nevertheless, it is possible to specify Holding as area, without compromising XPANGO driver, since Holding register can contain any data type: R, W or R/W.

As specified in 2.1.2.3, usually it is possible to understand immediately the register area from the Modbus address of the variable to be read. Converting the address from hex format (HEX=1000) into decimal, we find the value of 4096. According to the 'Modicon Rule', address value between 0 and 9999 is Coil (see Section 2.1.2.4) but it cannot be reconciled with the fact that the variable "V1 • L-N voltage phase 1" requires 32 bit. In this way, you can understand that the addresses are provided according to a different standard, and

# 2 Modbus and Siemens S7 templates

consequently it is impossible to identify the memory area where the register is stored.

Insert the converted value into decimal into *Address* column.

Leave *Bit of word* field empty as it is necessary to specify the bit only in case if, for example, more Boolean data should be stored in 16-bit registers.

*Data Type* is Float, since IEEE standard uses representation in floating point.

In *Signed* column, select TRUE or FALSE according to the indications in *Sign* column of the Modbus map. In this case, we set to FALSE.

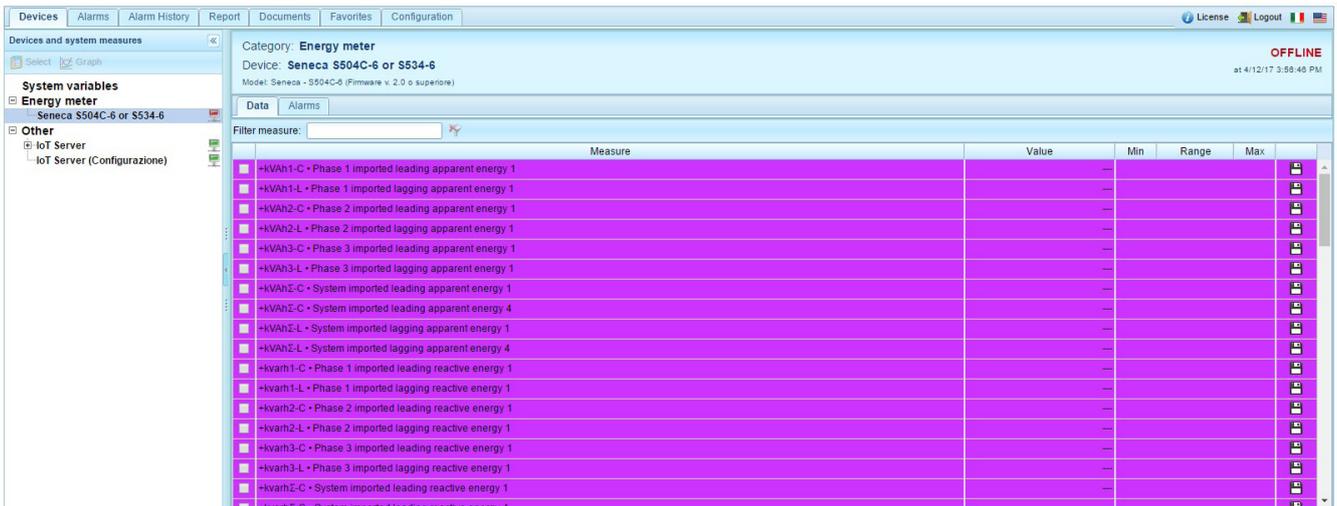
The fields of *Access type* column can be left empty, as they are set to Read by default, and in this case the variables are Read only (cod. \$03/\$04).

Specify the measuring unit, in this case it is Volt (V).

All other columns can be left empty, since no additional information is provided by the manufacturer. Set *Alarmed* column to TRUE if you want to link an alarm to the measure. We leave *Totalizer* and *Preselected* columns set to default (see Sections 2.1.2.12, 2.1.2.14, 2.1.2.15) as this is not a measure represented by monotonic increasing function (measure cannot be combined) and we want its value to be displayed in the IOT SCADA Server interface.

This should be done for all variables in the Modbus map. The list of variables from Excel file will be displayed in the IOT SCADA Server interface, after uploading XPANGO file and configuring the device to communicate with.

Plant Monitoring



Server date and time: 4/12/17 3:56 PM

IoTSCADA Powered by Alleantia

Figure 21 – IoT SCADA Server interface: List of variables of the configured device

# 2 Modbus and Siemens S7 templates

## 2.3 Siemens S7 Template

This template was developed to create the XPANGO driver of PLC Siemens S7.

The data types within the Excel file's columns shall not be described again, since they are the same as in the Modbus template (see Section 2.1). Having its own communication protocol, Siemens S7 template differs from the previous template.

Before connecting Siemens S7 device, check whether the elements of the TIA, listed in the following paragraphs, are enabled:

S7 1200/1500

An external device can access the CPU S7 1200 / 1500, using "base" S7 protocol, working only as HMI, i.e., only basic data transfers are allowed.

To access a DB, some additional PLC settings are required.

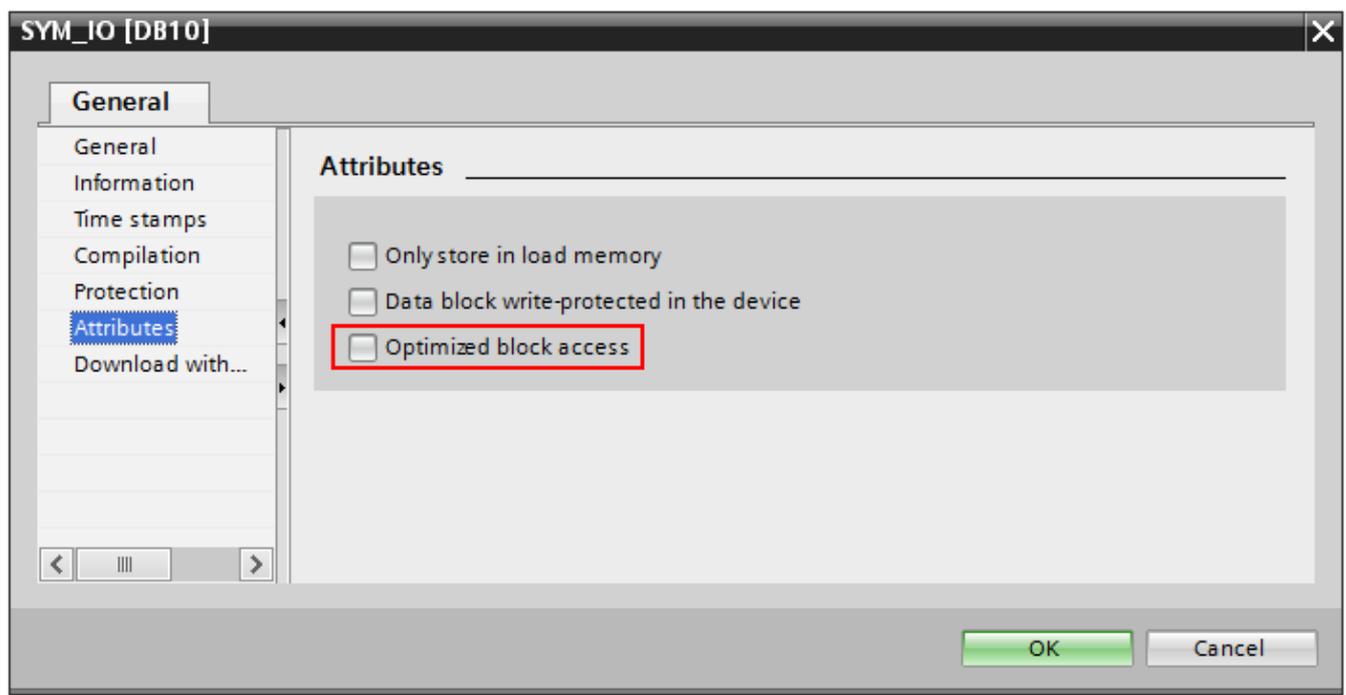
1. It is possible to access global DBs only.
2. Optimized access to the block must be deactivated.
3. The level of access must be "full" and the "connection mechanism" must be GET / PUT.

These settings in TIA Portal are explained below.

### DB properties.

Select the DB on the left side, under "Program blocks" and click Alt-Enter (or select "Properties..." from the menu)

Uncheck "Optimized block access" (it is checked by default).

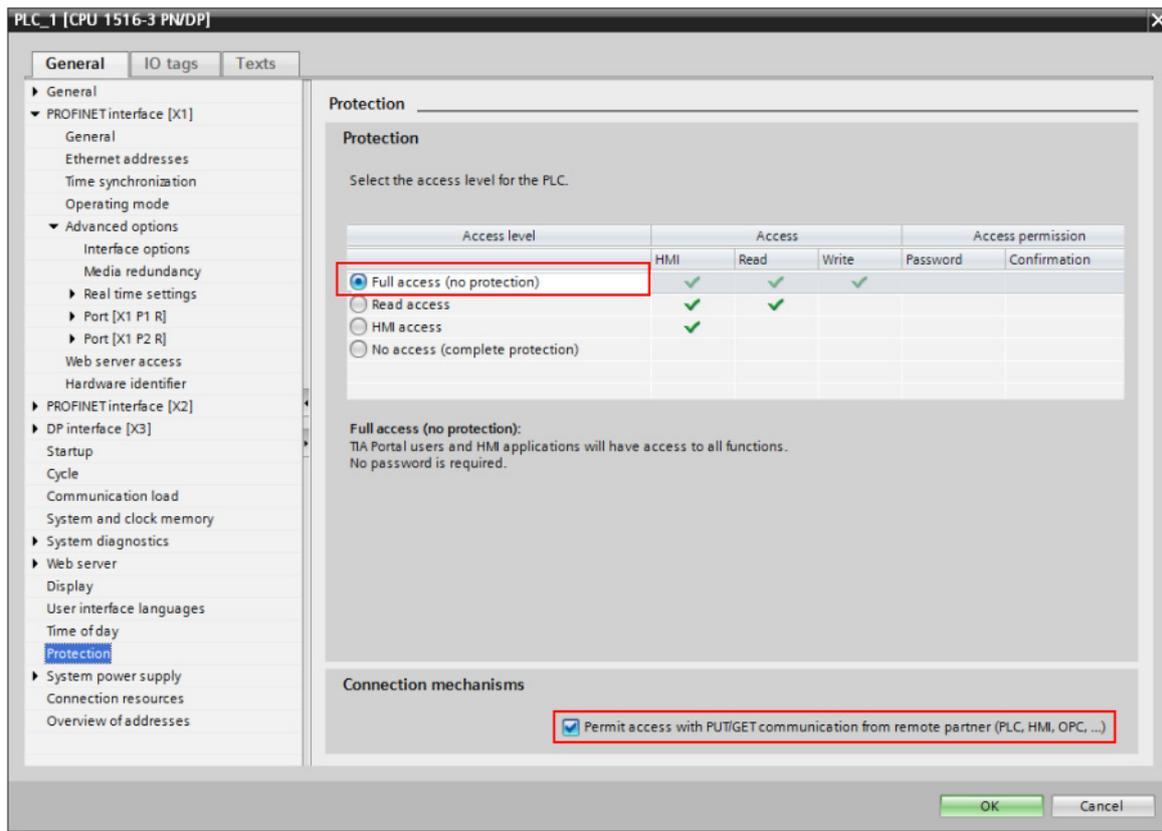


# 2 Modbus and Siemens S7 templates

## Protection

Select the CPU project on the left and click Alt-Enter (or select “Properties...” from the menu).

In *Protection* section, select “Full Access” and check “Permit access with PUT/GET ....”, as shown in figure below.



# 2 Modbus and Siemens S7 templates

## 2.3.1 General

The first difference is in *Category* field of *General* tab. There are 3 categories of devices: *PLC*, *Numerical control* and *Other*. Select first category if it is programmable logic controller (PLC), the second one for Computer numerical control (CNC) and the third category if the device does not fall under two categories.

*Delay Between Request* is set to 10 ms by default, so this the time elapsing from a request and the next one, if the field is empty.

Finally, in S7 template it is not possible neither to enable *Batch data transmission*, nor to choose from *Big* and *Little Endian word orders*, since Siemens S7 PLCs always use *Big Endian* order.

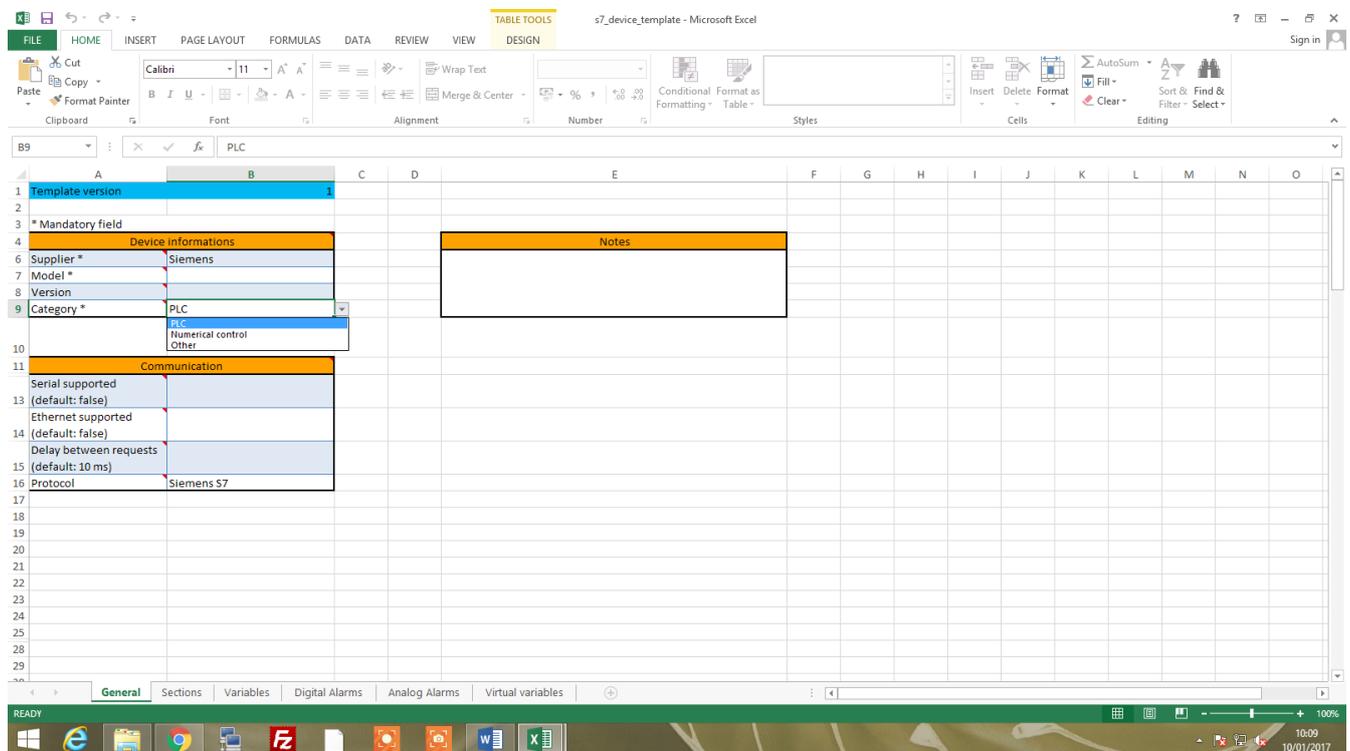


Figure 22 – General tab of the Siemens S7 template

# 2 Modbus and Siemens S7 templates

---

## 2.3.2 Variables

The structure of *Variables* tab is identical to the one in the Modbus template. Every variable has a numerical ID code in ID column which cannot be modified. Furthermore, there are differences in the in-cell dropdown lists of *Area* and *Data Type* columns and there is a new column called *Data Block Number*.

### 2.3.2.1 Area

In this column specify the memory area where the variable will be stored. Siemens PLC's memory is divided in different areas, each contains groups of variables with a specific function. In the dropdown list of Siemens S7 template contains 6 different memory areas:

- Input (IPI);
- Output (IPU);
- Marker;
- Timer;
- Counter;
- Data Block.

*IPI* and *IPU* memory areas stores, respectively values of logical inputs at the beginning of the cycle and values of logical outputs in the end of the cycle.

*Marker* area is for the bit variables and can be programmed.

*Timer* area provide timers, elements able to count time through increases (dt=1ms/10ms/100ms).

In *Counter* area insert counters, that counting events on an external signal. There are up counters, down counters and bidirectional counters.

Finally, there is *Data Block* global area for PLC communication. In this area, there are the variables stated by the PLC programmer, which can be read and monitored after creating the XPANGO driver.

### 2.3.2.2 Data Block Number

*Data Block Global* area of a PLC is divided in *Data Blocks*. *Data Block* is a portion of memory that can collect data of any type and dimension (bit, byte, word, etc.).

The physical dimensions of a *Data Block* are not preset but can be chosen by the programmer. Usually it is 1024 bit for PLC Siemens S7-300 and S7-400. The total dimension is divided in many 8-bit *Data bytes*. For these PLCs every *Data Block* contains 128 *Data bytes*.

## 2 Modbus and Siemens S7 templates

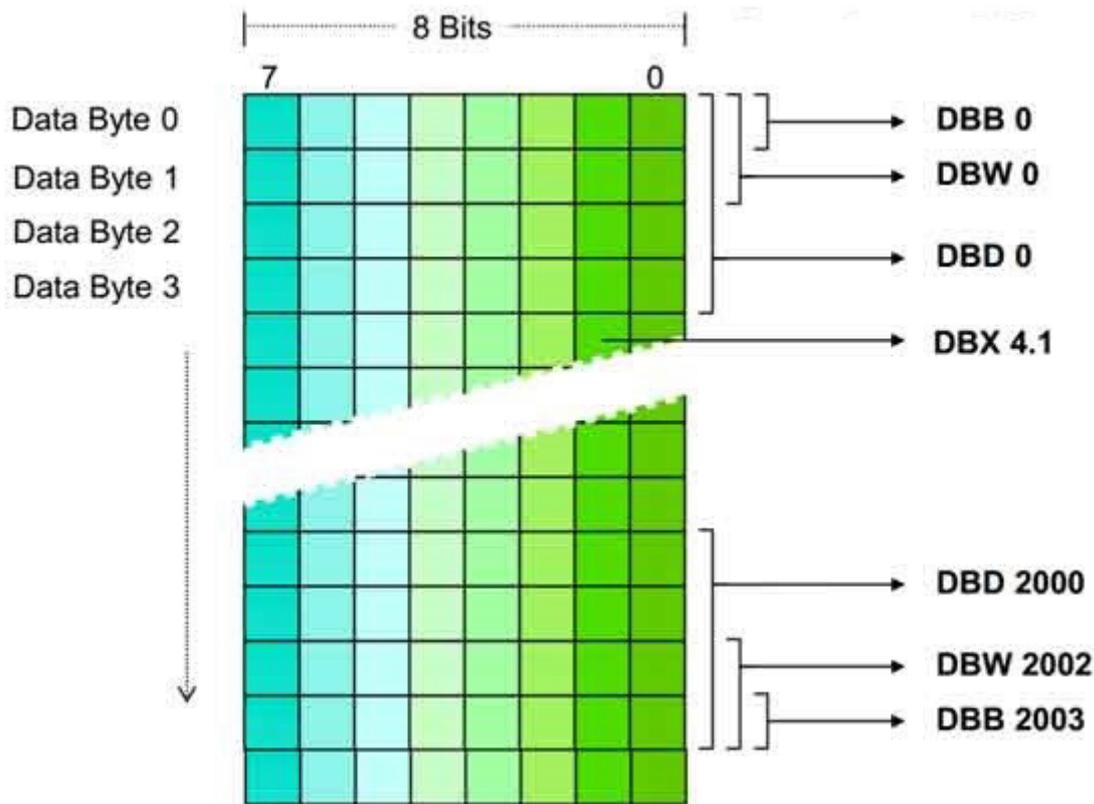


Figure 23 – Memory structure of a PLC

Every portion of memory contains variables data and have an ID number, that is specified in *Data Block Number* column. This field should be left empty if the variable if not in *Data Block Global* memory area.

### 2.3.2.3 Data Type

*Data Type* column values are selected from a dropdown list. The data types *Bool*, *Int* and *Dint* stand for *Boolean*, *Integer* and *Double Integer*. Other types are:

- *Byte*: number in hex format (8 bit);
- *Char*: stores a single character in ASCII format (1 bit);
- *Word*: fixed-sized decimal number (16 bit);
- *Double Word*: fixed-sized decimal number (32 bit)
- *Real*: floating point number in IEEE format (32 bit)
- *String*: sequence of alphanumeric characters;
- *Date*: date in IEC format with one day intervals.

### 2.4 Siemens S7 Template: completion example

In the following we give a complete example of a Siemens S7 Template for Simatic S7-300 PLC. The table, containing different variables to read, was extracted from the Siemens Step 7 software.

## 2 Modbus and Siemens S7 templates

SINOTTICO	SIMBOLO	INDIRIZZO	TIPO	COMMENTO	NOTE
	FAULT_01	DB70.DBW0	INT	WORD ALLARMI 01	
	FAULT_02	DB70.DBW2	INT	WORD ALLARMI 02	
	FAULT_03	DB70.DBW4	INT	WORD ALLARMI 03	
	FAULT_04	DB70.DBW6	INT	WORD ALLARMI 04	
	FAULT_05	DB70.DBW8	INT	WORD ALLARMI 05	
	FAULT_06	DB70.DBW10	INT	WORD ALLARMI 06	
	FAULT_07	DB70.DBW12	INT	WORD ALLARMI 07	
	FAULT_08	DB70.DBW14	INT	WORD ALLARMI 08	
	FAULT_09	DB70.DBW16	INT	WORD ALLARMI 09	
	FAULT_10	DB70.DBW18	INT	WORD ALLARMI 10	
PT101	SEND_A.VCP1	DB70.DBD24	DINT	PRESSIONE VUOTO #1	3 decimali
PT201	SEND_A.VCP2	DB70.DBD28	DINT	PRESSIONE VUOTO #2	3 decimali
PT301	SEND_A.VCP3	DB70.DBD32	DINT	PRESSIONE VUOTO #3	3 decimali
LT101	SEND_A.LEV1	DB70.DBD36	DINT	LIVELLO CAMERA DI FLASH #1	1 decimale
LT201	SEND_A.LEV2	DB70.DBD40	DINT	LIVELLO CAMERA DI FLASH #2	1 decimale
LT301	SEND_A.LEV3	DB70.DBD44	DINT	LIVELLO CAMERA DI FLASH #2	1 decimale
TT101	SEND_A.COT1	DB70.DBD48	DINT	TEMPERATURA CAMERA DI FLASH #1	1 decimale
TT201	SEND_A.COT2	DB70.DBD52	DINT	TEMPERATURA CAMERA DI FLASH #2	1 decimale
TT301	SEND_A.COT3	DB70.DBD56	DINT	TEMPERATURA CAMERA DI FLASH #3	1 decimale
TT102	SEND_A.RCT1	DB70.DBD60	DINT	TEMPERATURA RICIRCOLO #1	1 decimale
TT202	SEND_A.RCT2	DB70.DBD64	DINT	TEMPERATURA RICIRCOLO #2	1 decimale
TT302	SEND_A.RCT3	DB70.DBD68	DINT	TEMPERATURA RICIRCOLO #3	1 decimale
TT103	SEND_A.ST1	DB70.DBD72	DINT	TEMPERATURA VAPORE #1	1 decimale
TT203	SEND_A.ST2	DB70.DBD76	DINT	TEMPERATURA VAPORE #2	1 decimale
TT303	SEND_A.ST3	DB70.DBD80	DINT	TEMPERATURA VAPORE #3	1 decimale
TT104	SEND_A.DT1	DB70.DBD84	DINT	TEMPERATURA DISTILLATO #1	1 decimale
TT204	SEND_A.DT2	DB70.DBD88	DINT	TEMPERATURA DISTILLATO #2	1 decimale
TT304	SEND_A.DT3	DB70.DBD92	DINT	TEMPERATURA DISTILLATO #3	1 decimale
TT105	SEND_A.DTT1	DB70.DBD96	DINT	TEMPERATURA SERBATOIO DISTILLATO #1	1 decimale
TT205	SEND_A.DTT2	DB70.DBD100	DINT	TEMPERATURA SERBATOIO DISTILLATO #2	1 decimale
TT305	SEND_A.DTT3	DB70.DBD104	DINT	TEMPERATURA SERBATOIO DISTILLATO #3	1 decimale
TT106	SEND_A.VCT1	DB70.DBD108	DINT	TEMPERATURA SERBATOIO DEL VUOTO #1	1 decimale
TT206	SEND_A.VCT2	DB70.DBD112	DINT	TEMPERATURA SERBATOIO DEL VUOTO #2	1 decimale
TT306	SEND_A.VCT3	DB70.DBD116	DINT	TEMPERATURA SERBATOIO DEL VUOTO #3	1 decimale

Figure 24 – Sample list of variables addresses for a Simatic S7-300 Siemens PLC, extracted from Step 7 software

Useful information to fill in Excel template are in ADDRESS, TYPE, COMMENT and NOTE columns.

In COMMENT column, there is a variable name that we will monitor so we copy this information to *Variable name* column.

In ADDRESS column, there are communication addresses, in Siemens S7 communication protocol format. To understand it, we analyse every part:

### DB70.DBW0

First 4 digits of the address provide *Data Block Number* where the *WORD ALLARMI 01* variable is stored.

Last 4 digits after the dot, give information on memory area, containing the variable, the size of variable data and the number of *Data byte* from the *Data Block*.

Every area of PLC has its own ID prefix, consisting of 2 letters (in this case DB). The third letter represents data size (see Table 3).

## 2 Modbus and Siemens S7 templates

Area di memoria	Dimensione del dato	Prefisso	Area di memoria	Dimensione del dato	Prefisso
Immagine di processo degli ingressi	Bit	E	Blocchi dati di istanza	Blocco	DI
	Byte	EB		Bit	DIX
	Word	EW		Byte	DIB
	Double	ED		Word	DIW
Immagine di processo delle uscite	Bit	A	Stack dei dati locali	Double	DID
	Byte	AB		Bit	L
	Word	AW		Byte	LB
	Double	AD		Word	LW
Area merker	Bit	M		Double	LD
	Byte	MB	<b>Area di memoria</b>		<b>Prefisso</b>
	Word	MW	Area Timer		T
	Double	MD	Area Contatori		Z
Blocchi dati globali	Blocco	DB			
	Bit	DBX			
	Byte	DBB			
	Word	DBW			
	Double	DBD			

Table 3 – Prefixes used for variables addresses of PLC Siemens S7

This address refers to *WORD ALLARMI 01* variable from *Data Block Global* memory area and represents single-word data in first *Data byte* (DBW0) in *Data Block n° 70* (DB70).

This last information shows that the next variable is not contained in Data byte n° 1 but in n°2. Consequently, *WORD ALLARMI 01* variable uses 0 and 1 data bytes.

Knowing the data size is not enough to identify the data type. For example, both Double Word (DW) and Double Integer (DInt) use 4 bytes. Insert into *Data type* column of Siemens S7 template the information from TYPE column of table with addresses. In this case, this variable is Integer (Int).

ID	Variable name *	Section name	Area *	Data Block Number	Address *	Bit	Data type *	Access type (default: Read)	Measurement unit	Scale
1	WORD ALLARMI 01		Data Block	70	0		Int			

Figure 25 – Example of a 16-bit variable in Siemens S7 template

The NOTE column includes the scale to multiply the actual value of the variable before displaying. For example, “3 decimals” correspond to 0,001 scale (*1 decimal* = 0,1, *2 decimals* = 0,01). As well as for Modbus template, the scale should be inserted in *Scale* column.

## 2 Modbus and Siemens S7 templates

ID	Variable name *	Section name	Area *	Data Block Number	Address *	Bit	Data type *	Access type (default: Read)	Measurement unit	Scale	Offset	Minimum	Maximum
1	WORD ALLARMI 01		Data Block	70	0		Int						
2	WORD ALLARMI 02		Data Block	70	2		Int						
3	WORD ALLARMI 03		Data Block	70	4		Int						
4	WORD ALLARMI 04		Data Block	70	6		Int						
5	WORD ALLARMI 05		Data Block	70	8		Int						
6	WORD ALLARMI 06		Data Block	70	10		Int						
7	WORD ALLARMI 07		Data Block	70	12		Int						
8	WORD ALLARMI 08		Data Block	70	14		Int						
9	WORD ALLARMI 09		Data Block	70	16		Int						
10	WORD ALLARMI 10		Data Block	70	18		Int						
11	PRESSIONE VUOTO #1		Data Block	70	24		DInt			0,001			
12	PRESSIONE VUOTO #2		Data Block	70	28		DInt			0,001			
13	PRESSIONE VUOTO #3		Data Block	70	32		DInt			0,001			
14	LIVELLO CAMERA DI FALSH #1		Data Block	70	36		DInt			0,1			
15	LIVELLO CAMERA DI FALSH #2		Data Block	70	40		DInt			0,1			
16	LIVELLO CAMERA DI FALSH #3		Data Block	70	44		DInt			0,1			
17	TEMPERATURA CAMERA DI FLASH #1		Data Block	70	48		DInt			0,1			
18	TEMPERATURA CAMERA DI FLASH #2		Data Block	70	52		DInt			0,1			
19	TEMPERATURA CAMERA DI FLASH #3		Data Block	70	56		DInt			0,1			
20	TEMPERATURA RICIRCOLO #1		Data Block	70	60		DInt			0,1			
21	TEMPERATURA RICIRCOLO #2		Data Block	70	64		DInt			0,1			
22	TEMPERATURA RICIRCOLO #3		Data Block	70	68		DInt			0,1			
23	TEMPERATURA VAPORE #1		Data Block	70	72		DInt			0,1			
24	TEMPERATURA VAPORE #2		Data Block	70	76		DInt			0,1			
25	TEMPERATURA VAPORE #3		Data Block	70	80		DInt			0,1			
26	TEMPERATURA DISTILLATO #1		Data Block	70	84		DInt			0,1			
27	TEMPERATURA DISTILLATO #2		Data Block	70	88		DInt			0,1			
28	TEMPERATURA DISTILLATO #3		Data Block	70	92		DInt			0,1			
29	TEMPERATURA SERBATOIO DISTILLATO #1		Data Block	70	96		DInt			0,1			
30	TEMPERATURA SERBATOIO DISTILLATO #2		Data Block	70	100		DInt			0,1			
31	TEMPERATURA SERBATOIO DISTILLATO #3		Data Block	70	104		DInt			0,1			
32	TEMPERATURA SERBATOIO DEL VUOTO #1		Data Block	70	108		DInt			0,1			
33	TEMPERATURA SERBATOIO DEL VUOTO #2		Data Block	70	112		DInt			0,1			
34	TEMPERATURA SERBATOIO DEL VUOTO #3		Data Block	70	116		DInt			0,1			

Figure 26 – Example of Siemens S7 template: scale

The address of this Boolean data type is interesting:

DB70, DBX190.1

In this case, to identify position of the data in memory, it is necessary to specify the number of bits that the data contains. This information is after the dot, which can have value from 0 to 7 (also here enumeration starts from 0). Insert this value into Bit column of Siemens S7 template.

ID	Variable name *	Section name	Area *	Data Block Number	Address *	Bit	Data type *	Access type (default: Read)	Measurement unit	Scale
89	LIVELLO MINIMO SERBATOIO DEL VUOTO #1		Data Block	70	190	1	Bool			

Figure 27 – Example of Boolean variable in Siemens S7 template

To sum up, the following address refers to variable called *LIVELLO MINIMO SERBATOIO DEL VUOTO #1* in the *Data Block Global* and is represented by a 1-bit data (DBX) from *Data Block n° 70* (DB70), exactly in 2° bit del 191° *Data byte* (190.1).

Knowing the data size is enough to identify the data type, since among the data types of PLC, only Boolean has 1-bit dimension. In this case, TYPE column contains Bool.

The template variables will be displayed in IOT SCADA Server interface in the same order after uploading the .xmod file and connecting the PLC.

## 2 Modbus and Siemens S7 templates

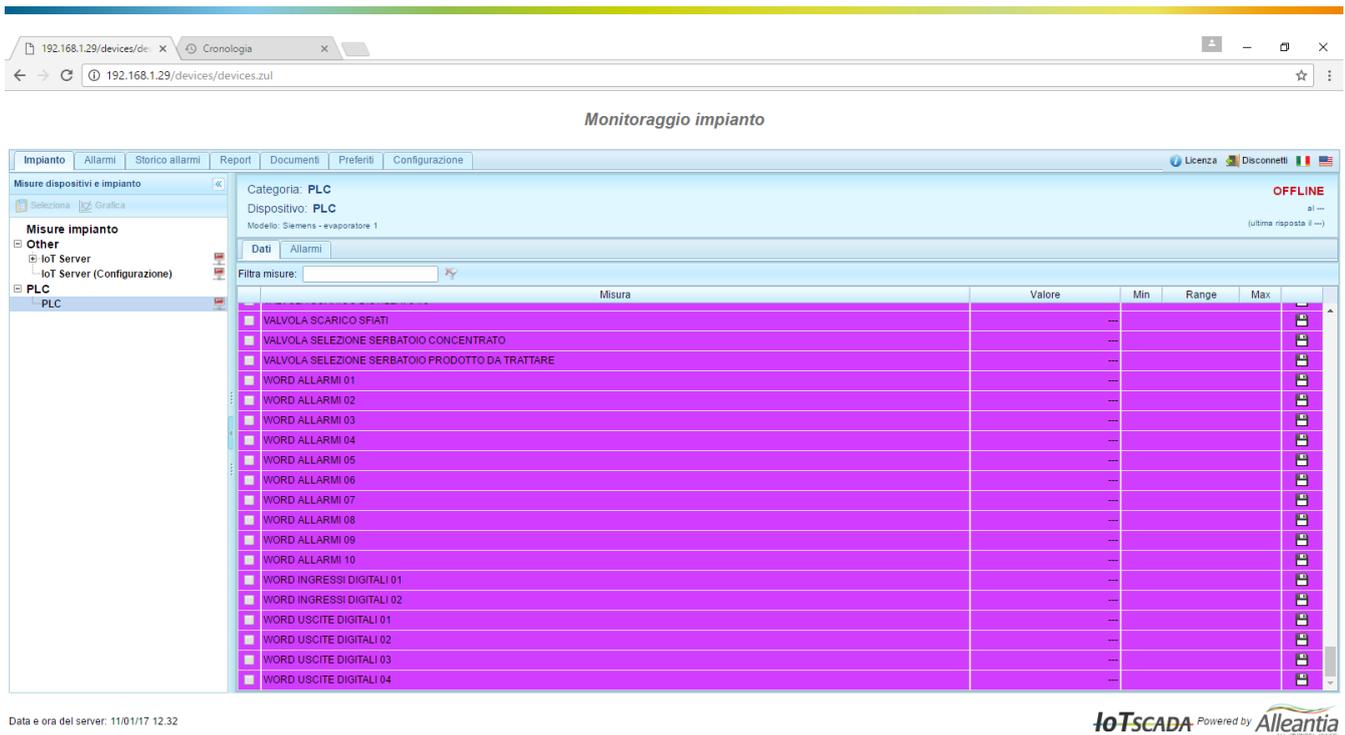


Figure 28 – IoT SCADA Server interface: variables of PLC Siemens displayed after configuration

## 3 XPANGO Driver Editor

Create driver by uploading the excel file here: <http://cloud.alleantia.com/xmod/convert.zul>.

Click *Upload File Excel* button, select the Excel file of Template, then click *Open*.

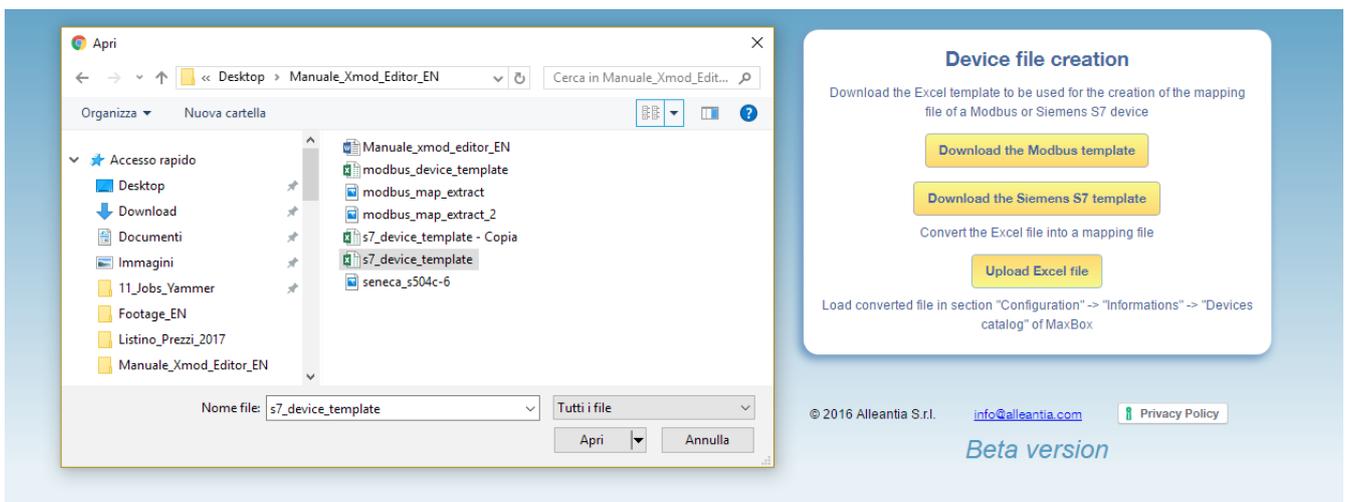


Figure 29 – XPANGO Driver Editor: uploading the template in Excel format to convert it in XPANGO driver

### 3 XPANGO Driver Editor

The XPANGO Driver Editor will convert the .xlxs file into .xmod file. The procedure is the same for both Siemens S7 template and Modbus template.

If mapping file creation is successfully completed, the .xmod file will be downloaded automatically.

For updates on available Editor Tools and mapping of industrial devices, please refer to par. 4.

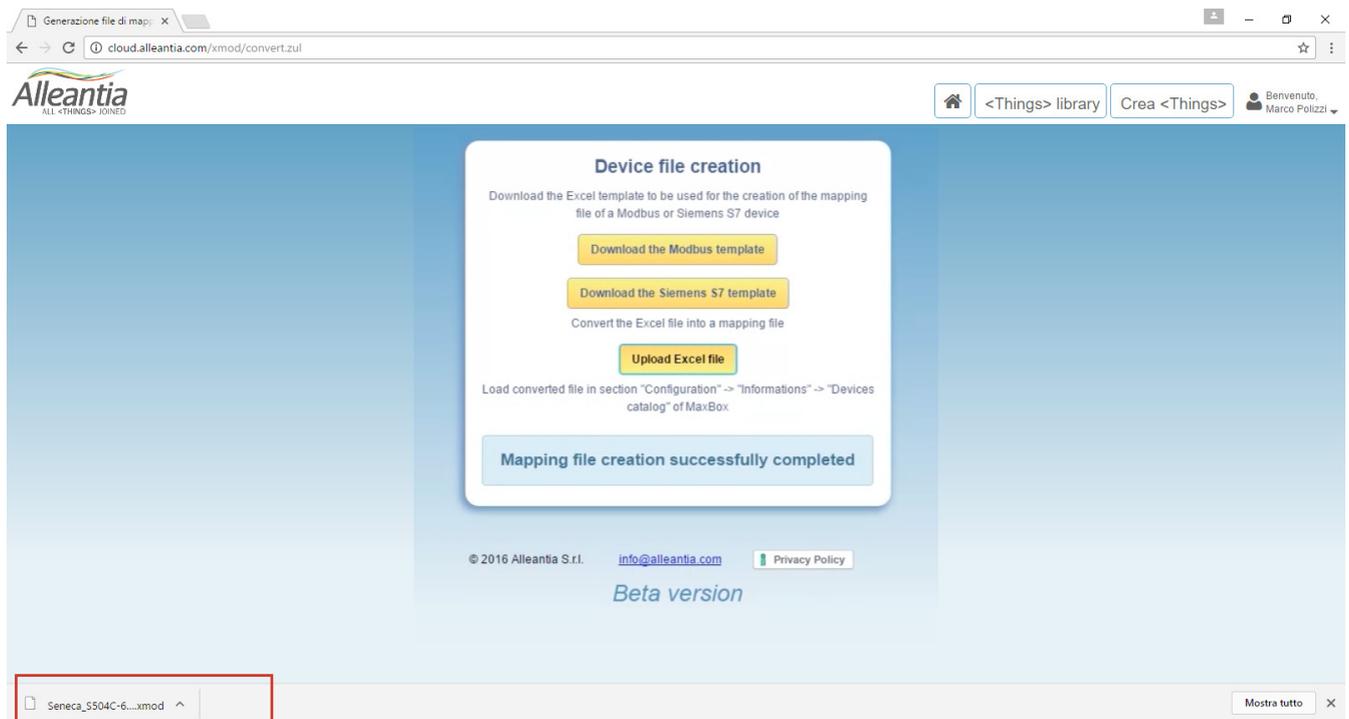


Figure 30 – XPANGO Driver Editor: mapping file creation successfully completed and auto download of XPANGO driver

For a successful communication with the device, its XPANGO driver should be in IoT SCADA Server's *Library of Things*.

Therefore, first upload .xmod file to IoT SCADA Server (see *IOT SCADA Server Installation and User Manual* - Section 5.5.1), then set communication parameters of the device (see *IOT SCADA Server Installation and User Manual* - Section 5.2.2.1).

## 4 Guidelines and suggestions

If the file contains errors, “The uploaded file contains some errors” message will be displayed and XPANGO Driver Editor will auto download errors.xlsx file.

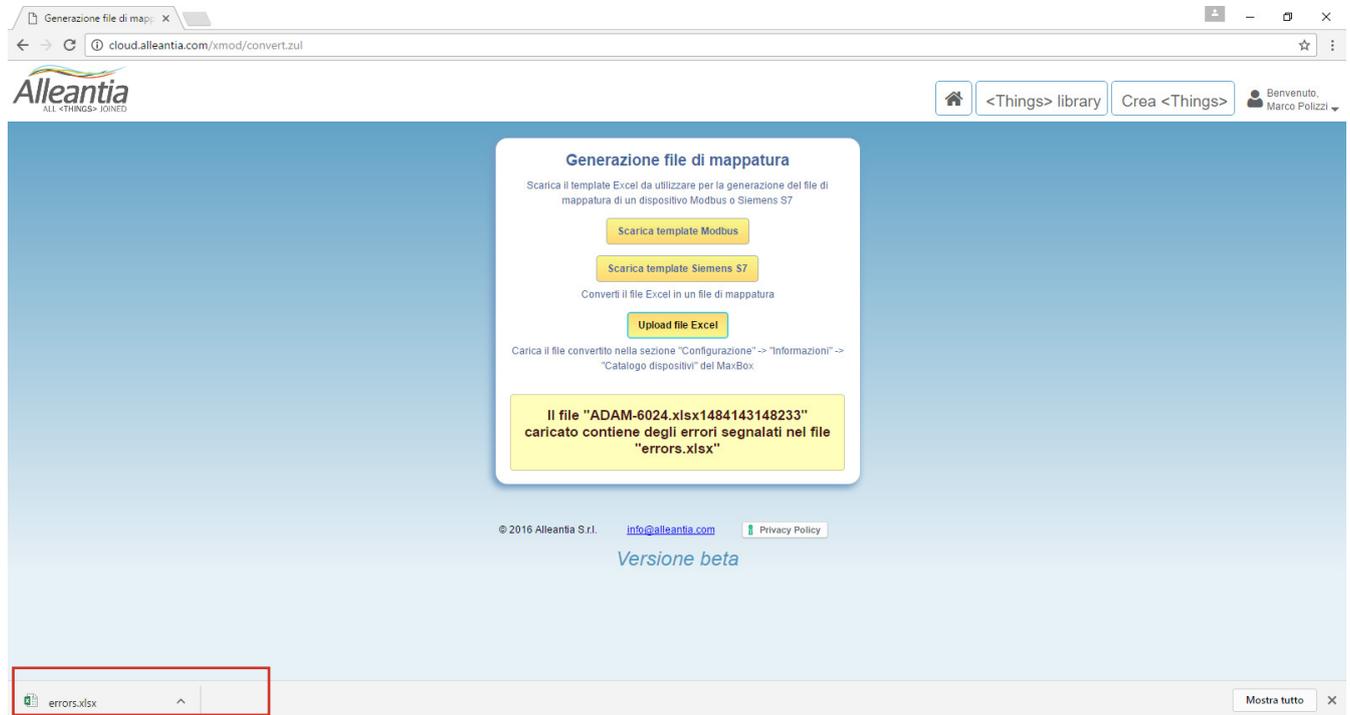


Figure 31 – XPANGO Driver Editor: Errors in Modbus template and auto download errors.xlsx file

In this file, the cells with errors are highlighted and contain useful notes for problem solving. This system helps to find errors quickly (Modbus map has a lot of rows) and provides instructions to use it.

IoT SCADA Server devices can read and show a number of variables, which depends on the license uploaded to the device. It is recommended that you insert only necessary variables or disable the ones you do not need at the moment, setting *Preselected* cell to FALSE.

Organize the variables in sections to make IoT SCADA Server interface more readable, especially if you have a large number of variables.

Use only Microsoft Office programs to fill the templates. For example, if you use Apache OpenOffice, it will be impossible to conclude the mapping because of incompatibility.

XPANGO drivers for Siemens S7 PLCs created by XPANGO Driver Editor might be incompatible and, consequently, impossible to configure, with IoT SCADA server devices with software version lower 3.4.1.

# 5 Contacts

---

**Alleantia s.r.l.**

[www.alleantia.com](http://www.alleantia.com)

**Registered offices:** Via Tosco Romagnola, 136 56025 Pontedera (PI)

**Operating headquarters:** Via Umberto Forti, 24/14 56121 Pisa

**VAT code/Tax code:** IT 02011550502

[info@alleantia.com](mailto:info@alleantia.com)



