

Tampere University of Applied Sciences, Master of engineering  
Department of electrical engineering  
Crivelli Manuel

Final thesis

# **Design and creation of an HMI and a PROFIBUS network configuration with a PLC for a coiler drive**

Thesis supervisor: M.Sc. Lauri Hietalahti

TAMK

Tampere

June 2010

**Writer:** Crivelli Manuel

**Thesis:** Design and creation of an HMI and a PROFIBUS network configuration with a PLC for a coiler drive

**Number of pages:** 102

**Graduation time:** June 2010

**Thesis supervisor:** Lauri Hietalahti

**Commissioned by:** Tampere University of Applied Sciences

---

## **Abstract**

This work is about the design of a user's Human-Machine Interface (HMI) that is used to manage a Coiler Drive. The user has to be able to set settings and check data in this HMI. The user can for example change the process speed and check the tension of the nylon web of a paper machine.

Data and parameters set are sending through an industrial field bus (PROFIBUS) to a Programmable Logic Controller (PLC).

The purpose of this thesis is to explain the process to create an HMI and to configure a PROFIBUS network. Different software has been used and they are explained summarily.

The creation of HMI is very important for the user because it allows the easy entry of settings and furthermore the simple control of a process. Today, almost every process contains HMI.

With the HMI, the Coiler Drive becomes a complete configurable industrial machine.

Future steps will be the analysis of the settings for modification, deletion or addition.

---

**Keywords:** Final thesis, Coiler Drive, Human-Machine Interface, PROFIBUS, Programmable Logic Controller, Frequency converter, Siemens, Vacon.

**Tekijä:** Crivelli Manuel

**Opinnäytetyö:** HMI:n suunnittelu ja toteutus sekä Profibus-verkon konfigurointi  
kelainkäyttöä varten

**Sivumäärä:** 102

**Valmistumisaika:** Kesäkuu 2010

**Työn ohjaaja:** Lauri Hietalahti

**Työn tilaaja:** Tampereen ammattikorkeakoulu

---

## Tiivistelmä

Tässä työssä suunnitellaan ja toteutetaan ihminen-kone-rajapintaa (Human-Machine Interface, HMI), jota käytetään kelainkäytön hallintaan. Käyttäjän on voitava tehdä asetuksia ja tarkistaa tietoja tämän HMI:n avulla. Käyttäjä voi esimerkiksi muuttaa prosessin nopeutta ja tarkistaa paperikoneen nylonverkon kireyttä.

Tiedot ja parametrit lähetetään teollisen kenttäväylän Profibusen kautta ohjelmoitavan logiikan ohjaimeen (Programmable Logic Controller, PLC).

Tämän opinnäytetyön tarkoituksena on selvittää HMI:n suunnitteluprosessia ja Profibus-verkon konfigurointia. Työssä on käytetty erilaisia ohjelmistoja, jotka on kuvattu lyhyesti.

HMI:n suunnittelu on sängen tärkeää käyttäjän kannalta, koska se mahdollistaa helpot asetukset ja lisäksi helpottaa prosessin valvontaa. Nykyisin melkein kaikissa prosesseissa onkin käytössä HMI.

HMI:n avulla kelainkäytöstä saadaan täysin konfiguroitava teollisuuskone. Jatkossa on tarkasteltava muokkausten, poistojen ja lisäysten asetuksia.

---

Hakusanat: kelainkäyttö, ihmisen ja koneen rajapinta, HMI, Profibus, ohjelmoitavan logiikan ohjain, taajuusmuuttaja, Siemens, Vacon

## Foreword

This work has been carried out at Tampere University of Applied Science in Finland during the second semester 2010 academic course.

Working on this subject was a big challenge for me because I learned notions during my training in the “Institut Supérieur Industriel de Bruxelles” but I never really practice it.

I progressed step by step and of course I had problems but it was very interesting to try each different approaches and to finally figure out a solution.

Luckily I was not alone to do this work, Mr. Sc. Lauri Hietalahti helped me a lot and that is why I would like to offer my most profound thanks to him, for his help and for the opportunity to work on this project.

Also, a very special gratitude to Mrs Kirsti Kallio for her help with the translation of my abstract to Finnish, to Mrs Ruth Crivelli and Mrs Isabelle Cartier to take the time to correct my text and finally to my teacher Mr. Sc. Olivier Debia for his help during my PROFIBUS addresses problem.

Tampere, Finland

June 2010

---

Crivelli Manuel

## Table of Contents

1	Introduction .....	1
2	Web industries .....	2
2.1	Coil history .....	3
2.2	Coil fabrication .....	3
2.3	Winder machine .....	4
3	Coiler drive presentation.....	6
4	Electric motor .....	7
4.1	Construction .....	7
4.2	Functional principle.....	8
4.3	General specifications.....	9
4.3.1	Slip.....	9
4.3.2	Speed control .....	9
4.4	Electric motor in the Coiler Drive.....	10
5	Frequency converter .....	11
5.1	Electrical diagram .....	11
5.1.1	Rectifier.....	12
5.1.2	Intermediate circuit.....	12
5.1.3	Inverter .....	12
5.1.4	Control and regulation circuit.....	12
5.2	Frequency converter drives .....	13
5.3	Frequency converter in the Coiler Drive .....	14

6	Programmable logic controller.....	15
6.1	Introduction.....	15
6.2	Design of a PLC.....	16
6.3	Processor.....	19
6.4	Input/Output module (I/O).....	19
6.5	Storage elements.....	19
6.6	Auxiliary.....	20
6.7	Programming.....	20
6.8	Language.....	21
6.8.1	Instruction List (IL).....	21
6.8.2	Structural Text (STL).....	21
6.8.3	Ladder Diagram (LAD).....	22
6.8.4	Function Bloc Diagram (FBD).....	22
6.9	PLC in the coiler drive.....	23
7	Sensors in the coiler drive.....	24
7.1	Tension sensor.....	24
7.2	Speed sensor.....	25
8	Software for the frequency converter.....	26
8.1	NC Load.....	26
8.2	NC Drive.....	27

9	Software for the PLC.....	28
9.1	STEP-7.....	28
9.1.1	Presentation of STEP-7 .....	28
9.1.2	Hardware configuration' .....	31
9.1.3	The program .....	32
9.1.4	Transfer to the PLC .....	33
9.1.5	More Helps.....	34
9.2	WinCC .....	35
9.2.1	HMI introduction.....	35
9.2.2	Presentation of WinCC.....	36
9.2.3	The Screen .....	37
10	HMI created for the coiler drive.....	39
11	Field bus solution .....	43
11.1	Introduction .....	43
11.1.1	PROFIBUS-FMS (Field Message Specification) .....	43
11.1.2	PROFIBUS-DP (Decentralized Peripheral).....	44
11.1.3	PROFIBUS-PA (Process Automation).....	44
11.2	PROFIBUS-DP Layer .....	45
11.2.1	Physical Layer (layer 1) .....	45
11.2.2	Bus connection.....	46
11.2.3	Field bus Data Link (Layer 2) .....	47
11.2.4	Bus Access Control.....	48
11.3	PROFIBUS-DP in SIMATIC S7 system .....	49
11.3.1	Generic Station Description (GSD) files.....	50

11.4	PROFIBUS network STEP-7 program .....	51
11.4.1	Reading data of a DP slave device with SFC 14 "DPRD_DAT" .....	51
11.4.2	Writing data of a DP slave device with SFC 15 "DPWR_DAT" .....	54
11.5	Cable and connector configuration .....	57
11.6	Frequency converter Vacon NXP configuration.....	57
11.7	Reading and writing communication .....	60
11.7.1	Parameter field .....	61
11.7.2	Process field .....	62
11.8	Example of data exchange through the PROFIBUS network .....	63
12	Final conclusion.....	65
12.1	The work with STEP-7.....	65
12.2	The work with WinCC .....	65
12.3	The work on the PROFIBUS network .....	66
12.4	What is next?.....	67
13	Bibliography.....	68
14	Appendices.....	72

# 1 Introduction

The aim of this thesis is to introduce the work done for the coiler drive. In the past, the coiler drive received customizations. The first one was to manage the process with two frequency converters, it was done by Jesús Macarro Galindo in 2004 (final thesis: "Frequency converters controlled by PC in coiler drive process"). The second one was to add a regulation by feedback system. That work was done in 2005 by Bravo Zarza Jacinto and García García José Luis (final thesis: "Design of coiler drive process and their regulation by feedback system").

Since, more customizations were done, for example, old frequency converters (Vacon CX) were replaced by two more recent (Vacon NXP) and a programmable logic controller (Siemens Simatic S7-300) was added. This new installation more sophisticated allows to manage the process with an human-machine interface (HMI) through a PROFIBUS network.

A HMI allows to have a view on every data in the process. It is useful to do practical experiences, to change the speed, the torque or the tension in the nylon web for example. With this installation, students are able to take some measurements and to see how works a process managed by an HMI through a PROFIBUS network.

Therefore, tasks were to do the design and the creation of an HMI and to configure all the settings for the PROFIBUS network. To explain all the work done, this structure is used: after a brief introduction to explain winder process, the presentation of the coiler with its different elements and the software used is introduced. Then, the main steps to create a program for programmable logical controller with the STEP-7 software and the creation of the HMI with WinCC software are explained. The last part contains the procedure to create and to use a PROFIBUS network with the PLC Simatic S7-300 and the Vacon NXP frequency converter.

## 2 Web industries

Paper is the oldest of the web industries with more than a century of culture and history. A web is a long, flexible and thin material, the paper industry holds many web records such as manufacturing speed (2 300 m\min) converting speed (3 050 m\min) and width (10 m). (Roisum 1998)

The demands of these high speeds and wide widths also necessarily mean that the paper industry tends to have more precise and robust machinery. The converting industry (drying, laminating, packing, etc) is bigger in terms of numbers than any other web manufacturing industries. There are 10 000 converters in North America but many of the converting companies and machine suppliers are smaller than paper industry companies. (Roisum 1998)

The largest web industry in terms of numbers is the printing industry, about 50 000 in North America. While most printing companies are small, several are large enough to do serious research and development. The Table 1, shows that the web industry is very large and diverse. In fact, web industry may very well be the largest industry. Paper and textiles are the first materials produced as a web, steel and plastic followed. (Roisum 1998)

**Table 1: Web material and product examples (Roisum 1998, 2)**

Film	Bags	Wrap, food
	Photographic	Wrap, shrink
	Tape, adhesive	Wrap, stretch
	Tape, magnetic	
Foil	Aluminium food wrap	
Nonwovens	Filters	Medical garment and covers
	Insulation	Personal hygiene products
Paper	Bags	Tape
	Boxes	Tissue
	Currency	Towelling
	Labels	Waxed
	Magazine	Wrapping
Textiles	Newspaper	Writing
	Carpet	Upholstery
	Clothing	

## 2.1 Coil history

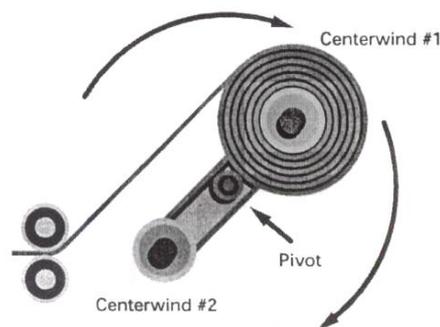
For 5 000 years thin materials as for example papyrus had been made in web. It is an economic advantage to make a product long because it can be manufactured continuously. In 1798, Louis-Nicolas Robert made for the first time webs continuously with the invention of the Fourdrinier paper machine. (*Roisum 1998*)

It took until the early of 19th century to see paper makers use coils of paper and not traditional sheets of paper. Over the years, the coils built became more and more longer and wider. Today, a coil for a newspaper machine is ten meter wide, ten kilometer long and can weight ten tons! (*Roisum 1998*)

## 2.2 Coil fabrication

The actual fabrication is made by a continuous way. The paper is winded without stop and when the coil has its maximal size, it is evacuated and a new coil support is loaded. The operation has to be made without deceleration of the machine and mostly without breakage paper. (*Roisum 1998*)

As seen in Figure 1, the solution is showed to wind a coil continuously. Two or more center winds are placed on a rotating axis that allows the other winder to be ready to start a new roll on the fly. (*Roisum 1998*)

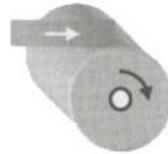


**Figure 1: Turret winder (*Galindo 2004, 50*)**

Such coils require a very careful handling. Indeed, a coil doesn't behave like a solid, for example if a coil is placed on the ground; the lower part is going to have a lot of stress and will grovel. This can cause some deformations, can make unusable because there will be dolt problem and this created suddenly and breakage. (*Roisum 1998*)

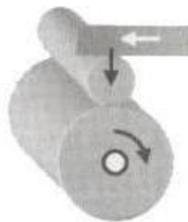
### 2.3 Winder machine

There are four kinds of class of winder. The simplest class is a center wind, as seen in Figure 2, where the wound roll is driven through its core. The drive is usually an electric motor who establishes incoming web tension between the wound roll and upstream driven element. (Good & Roisum 2008)



**Figure 2: Center winder (Good & Roisum 2008, 6)**

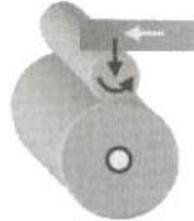
More complex is a center winder with layon roller, as seen in Figure 3. The layon roller also referred to as a pack roller or a ride roller serves several possible needs, depending on the application. (Good & Roisum 2008)



**Figure 3: Center winder with layon roller (Good & Roisum 2008, 6)**

First, it reduces air entrainment into the winding roll. Second, a layon roller serves an important role in the control of winding wrinkles and finally, a roller of some sort may be required to support the weight of the winding roll to reduce deflection in the system. (Good & Roisum 2008)

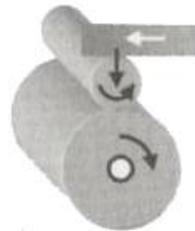
In many winders a drive motor is connected to the layon roller rather than winding roll. These winders are called surface winder as seen in Figure 4. (Good & Roisum 2008)



**Figure 4: Surface winder (Good & Roisum 2008, 6)**

Surface winders are the norm in paper mills but are founded in many other industries. The only one limitation with surface winders is that at low nip loading, the winder can unhook when the friction forces between the drums and the outer layer are insufficient to wind a roll. (Good & Roisum 2008)

The last common class of winder is the center-surface winder as shown in Figure 5.



**Figure 5: Center-surface winder (Good & Roisum 2008, 6)**

Two independently controlled motors are in the windup section. In one arrangement, one motor is connected to the core of the roll and the other to the layon roller. Center surface winders are complicated, costly and until recently fairly rare. (Good & Roisum 2008)

### 3 Coiler drive presentation

Coiler drive is a kind of winder machine, used in different industries such as paper, textile and even metal industry. This machine enables to roll different materials, Figure 6:



Figure 6: Material rollable by a coiler drive

The coiler drive used in Figure 7 is a didactic machine. It winds the nylon web from one coil to the other. The purpose of this machine is to familiarize the student with the process, however industrialist coiler drives are really more sophisticated.



Figure 7: The coiler drive

The different elements are showed in the Figure 8:

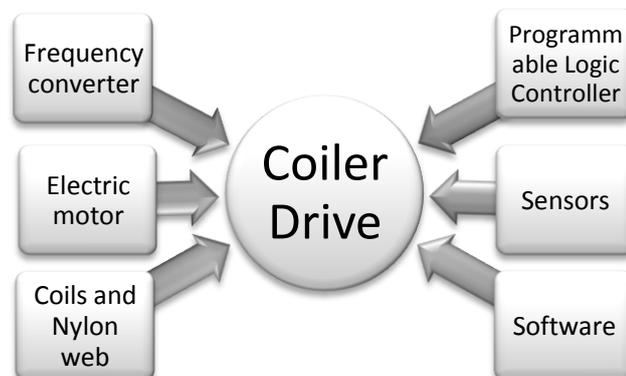


Figure 8: The different elements in the coiler drive

## 4 Electric motor

The electric motor used in the coiler drive is an asynchronous squirrel-cage geared motor. Today, this type of motor is the cheapest and the most used in the industry. The motor's brand is SEW-EURODRIVE, a German manufacturing company. Founded in 1937, it is now an international organization that manufactures geared motors, frequency inverters and servo drives. It employs over 11 000 people worldwide with annual sales in excess of 2 billions € (2008). (*EURODRIVE 2010*)

### 4.1 Construction

The basic structure of an asynchronous motor is schematically illustrated in Figure 9. There are two mechanically independent parts: the rotor and the stator.



**Figure 9: Asynchronous squirrel-cage motor** (*Amin 2001, 6*)

Both of them are separated from each other by a narrow air gap that allows the former to rotate freely inside the stator. The magnetic circuit of the rotor and stator consists of iron laminations. The stator and rotor yokes close the motor's magnetic circuit. Windings are fixed in slots distributed regularly around the air gap in both the rotor and the stator. (*Amin 2001*)

In the stator, there are pre-fabricated sections which contained several turns of elementary conductors (copper) lodge within the various slots regularly situated around the stator. Sections are linked together in order to constitute a number of independent circuit called phases. Most industrial asynchronous motor have three identical, symmetrical phases spaced by an angle of  $2\pi/3$  radian. Squirrel-cage is made of aluminium or copper bars. (*Amin 2001*)

The bars lodge in the rotor slots and are short-circuited at their two ends by the end-rings as seen on Figure 10.



Figure 10: Squirrel-cage (Amin 2001, 8)

## 4.2 Functional principle

The stator's coils are fed by an alternative three-phase current and each one produces a magnetic flux whose resultant is a rotating magnetic field. This flux cuts the rotor conductors and it induces a current which in turn generate a magnetic field induced in the squirrel cage. The rotor's field tends to oppose the flux change (Lenz's law) and a dynamic effect is created between the two fluxes and causes the rotation of the rotor. (Amin 2001)

The frequency of rotating field is imposed by the frequency of stator current. This is called the synchronous speed. If you change the frequency of the stator current, you change the speed of rotating field and thus the frequency of the rotor. The formula for the synchronous speed is:

$$n_s = \frac{f}{p}$$

$n_s$  in Hz, is the frequency of rotation of the rotating fields, called the synchronous speed.

$f$  in Hz is the frequency of stator current (AC power frequency)

$p$  is the number of phase

The rotor can never reach the synchronous speed as there would be no change in rotor flux in the conductors. This explains the name of this kind of motor: "asynchronous". There are two inconveniences when the motor is started: the current is very important in the stator and the torque is low. (Amin 2001)

Today, with the power electronic progress, frequency converters are used to change the speed, but also to manage a good start, typically with an enough torque and without a too high current in the stator. Squirrel-cage motors are used in the new TGV and in lots of subway around the world. (*Amin 2001*)

### 4.3 General specifications

#### 4.3.1 *Slip*

The Slip is a quantity that reflects the difference in speed of an induction machine compared to a hypothetical synchronous machine built with the same stator. In other words, it's the difference between the speed of the rotor and the speed of the rotating magnetic field in the stator. (*Amin 2001*)

The slip should stand around 2% for big motors and around 6–7% for small motors.

$$s = \frac{n_s - n_r}{n_s}$$

$s$  the slip in %

$n_s$  in Hz, is the frequency of rotation of the rotating fields, called the synchronous speed.

$n_r$  in Hz, is the frequency of rotation of the rotor

#### 4.3.2 *Speed control*

As seen above, to change the rotor's speed, we need to change the frequency of the rotating field in the stator. In other words, the synchronous speed has to be changed. In the synchronous speed's formula, there are only two variables:  $f$  for frequency and  $p$  for phase number. If the number of phase is changed, the frequency of synchronization will be still a sub-multiple of the mains frequency. With 50 Hz, for example, sub-multiple of 3 000 rpm will be, 1 500, 1 000, 750, etc. (*Amin 2001*)

Another way to change the speed is to vary the frequency  $f$  of the stator's current. As a result, the frequency converters would change not only the frequency  $f$  but the tension as well. The advantage is to keep the motor in the same magnetic state regardless of the frequency of the stator's current.

This way allows to keep the same curve as a function of frequency torque for any frequency of stator current, as seen in the Figure 11 and 12. (Amin 2001)

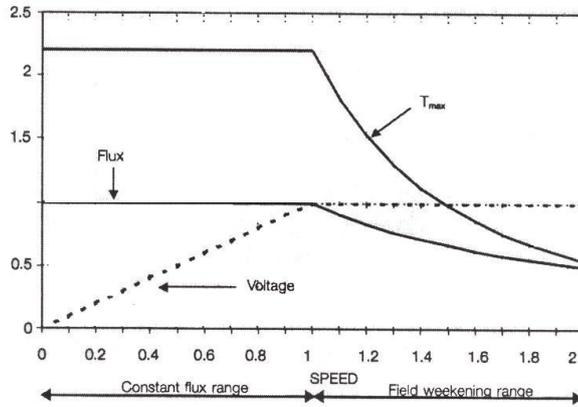


Figure 11: Maximum torque (Galindo 2004)

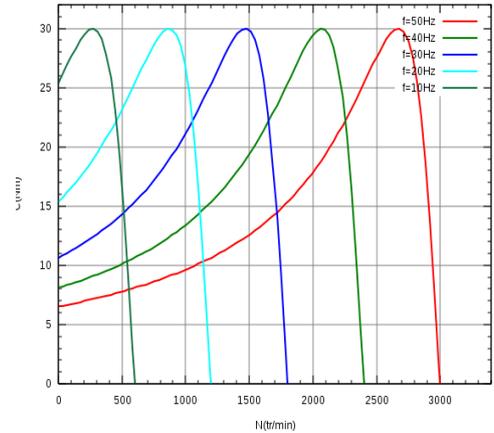


Figure 12: Torque trend (Amin 2001, 15)

#### 4.4 Electric motor in the Coiler Drive



Figure 13: Unwinder motor



Figure 14: Winder motor

**Designation:** R27 DT90S4 / BMG / EV1A

**Nominal power :** 1.1kW

**Nominal Torque:** 7.5 N m

**Nominal speed:** 1400 rpm

**Nominal current:** 2.8 A

**Nominal voltage :** 400 V

**Power factory :**  $\cos \varphi$  0.77

**Efficiency :** 77.5 %

## 5 Frequency converter

A frequency converter is an electronic device that converts alternating current (AC) of one frequency, usually 50 or 60Hz to an alternating current of another frequency.

The frequency converter used in this study comes from Vacon brand. It is a manufacturer of variable speed AC drives and comes from Finland. Founded in 1993 with 13 key employees, it is now one of a leading global supplier of AC drive around the world. Since, 1 200 more people have joined manufacture, serving a network of customers and partners all over the world. The revenues in 2008 were 293 millions. (VACON 2010)

### 5.1 Electrical diagram

There are four main components in modern frequency converters, Figure 15:

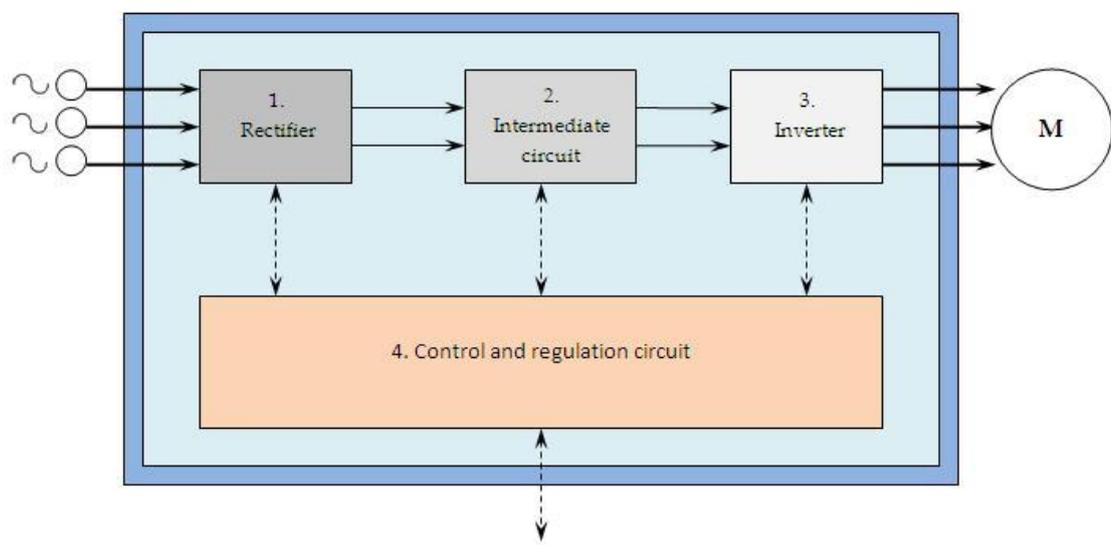


Figure 15: four main components in modern frequency converter (Galindo 2004)

Recent frequency converters also contain a stage power factor correction to respect the electromagnetic compatibility standard.

### **5.1.1 Rectifier**

The rectifier stage produces direct current which will then be inverted to produce AC of the desired frequency. Its purpose is to convert the three-phase AC voltage to a pulsating DC voltage. (*Amin 2001*)

### **5.1.2 Intermediate circuit**

There are many kinds of intermediate circuits but three kinds are the most important. One kind converts the voltage of the rectifier into a DC current. A second type stabilizes the pulsating DC voltage and sends it into the inverter. The last type, converts a constant DC voltage from the rectifier into a variable value. (*Amin 2001*)

### **5.1.3 Inverter**

The inverter's role is to convert constant DC voltage into an AC voltage and to control the frequency. It may use thyristors, IGCTs or IGBTs. (*Amin 2001*)

### **5.1.4 Control and regulation circuit**

The electronic control unit manages circuit transmit signals to the rectifier, the intermediate circuit and the inverter. It carries out control and subjugation of the machine through the static inverter so that; the user can directly control the speed. Its design depends primarily on the control strategy chosen, Vector control or Scalar control. (*Amin 2001*)

## 5.2 Frequency converter drives

As seen in the squirrel-cage motor's chapter, the electric motor speed is related to the frequency of the stator's currents.

This link allows a mathematical control of the rotor's speed by controlling the frequency of the stator's current. Thus, there is a direct relationship between the driving frequency current to the stator and the rotor mechanical's speed, which allows for any desired mechanical's speed to set the corresponding stator frequency.

This is the frequency converter's principle, controlling a mechanical speed by controlling the frequency of the stator current. There are many advantages to use a frequency converter, such as shown in the Figure 16:

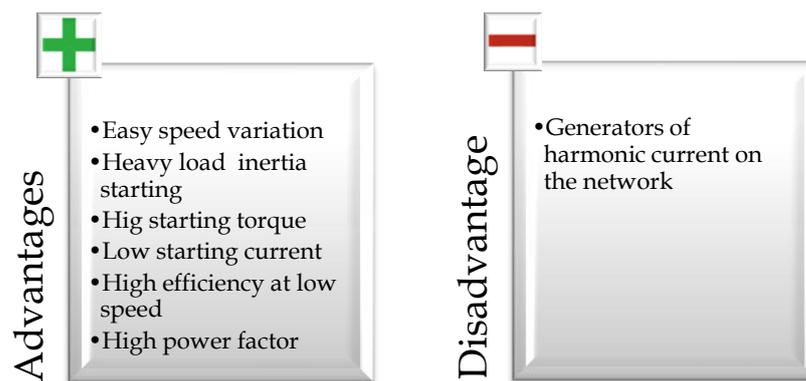


Figure 16: Advantages vs. disadvantage

The advance of powers electronics technology and the cost of these kinds of equipments have made a current utilization of frequency converter in the industry.

But frequency converters aren't used only with squirrel-cage motors. For example one another application is in the aerospace and airline industries. Often, airplanes use 400Hz power. Frequency converter 50Hz or 60Hz / 400Hz is needed to use in the ground power unit. These power units are used to power the airplane while it is on the ground. Other application in renewable energy systems, frequency converters are an essential component of double fed induction generators as used in modern multi-megawatt class wind turbines. (*e-document [15]*)

### 5.3 Frequency converter in the Coiler Drive

Brand: **VACON**  
DRIVEN BY DRIVES



**Figure 17: The two frequency converters Vacon NXP used**

The designation and its meaning of the frequency converter is in appendix 1

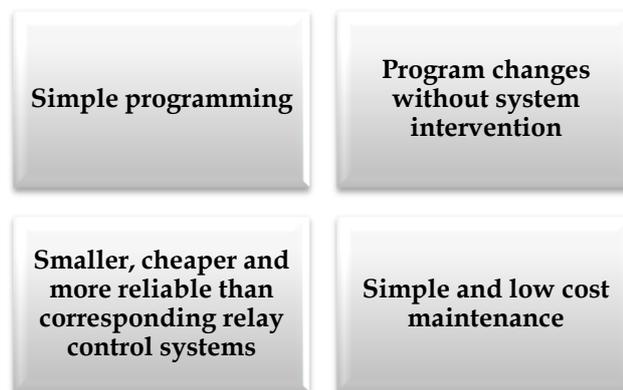
## 6 Programmable logic controller

Now, Programmable Logic Controllers (PLCs) represent a key factor in industrial automation. Its use allows a flexible adaptation to varying processes also as rapid fault finding and error elimination. (*e-document [16]*)

In this chapter, the design of a programmable logic controller and its interaction with peripherals will be explained.

### 6.1 Introduction

In 1968, a group of engineers at General Motors developed the first PLC. The company looked for a new device able to replace complex relay control systems. The different requirements were shown on the Figure 18:



**Figure 18: Requirements for the first PLC**

The development resulted in a system; which enabled the simple connections of binary signals. With this kind of system, it is possible for the first time to plot signals on a screen and to file these in electronic memories. (*e-document [16]*)

The task of a PLC is the interconnection of input signals according to a specified program and, if the value is “true”, to switch the corresponding output. Boolean algebra is the base of all the mathematical operations. For example, a motor could therefore be switched on or off. This I/O behaviour is similar to that of a pneumatic switching valve controller or an electromagnetic relay. Since the first PLC, enormous progresses are made in the development of microelectronics and during the seventies; binary I/O were finally expanded with the addition of analogue I/O.

Since, many of applications require analogue processing, Figure 19, as such:



**Figure 19: Applications with analogue processing**

The program is stored in an electronic memory and now the tasks of a PLC have rapidly multiplied. For example, mathematical computing operations, timer and counter functions, memory setting, etc. (*e-document [16]*)

The PLCs currently on offer in the market place have been adapted to customer requirements. It's really possible to buy a perfect adapted PLC for any applications. Miniature PLCs are now available with a little number of I/O, also available larger PLCs until 256 I/O. Additional I/O, analogue, positioning and communication modules can be installed to extend possibilities of PLCs. (*e-document [16]*)

## 6.2 Design of a PLC

The term "programmable logic controller" is defined as follows by EN 61131-1 norm:

"A digitally operating electronic system, designed for use in an industrial environment, which uses a programmable memory for the internal storage of user-oriented instructions for implementing specific functions such as logic, sequencing, timing, counting and arithmetic, to control, through digital or analogue inputs and outputs, various types of machines or processes. Both the PC and its associated peripherals are designed so that they can be easily integrated into an industrial control system and easily used in all their intended functions." (EN 61131-1 norm)

A programmable logic controller is therefore nothing more than a computer, made specifically for certain control tasks. The Figure 20 shows system components of a PLC.

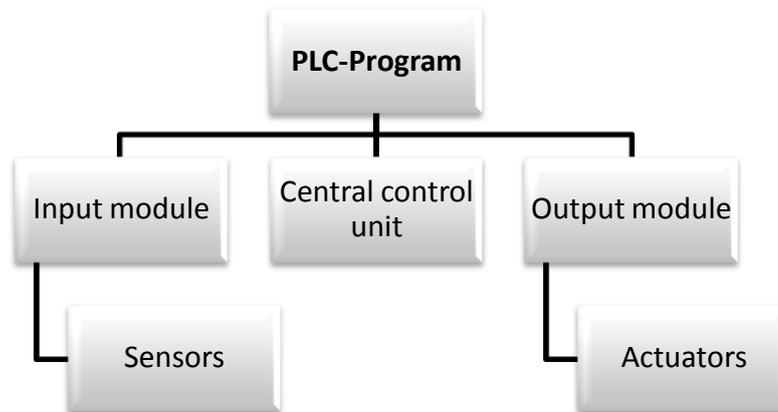


Figure 20: System components of a PLC (e-document [16])

**The input module** converts incoming signals into signals, which can be processed by the PLC. It passes these to the central control unit.

**The output module** performs the reverse task; this converts the PLC signals into signals suitable for the actuators.

**The central control unit** process signals in accordance with the program stored in the memory. The program of a PLC can be created in various languages, Figure 21, as such:



Figure 21: Languages

Other special languages are also available, like Grafset language for example.

Most frequently, in industry modular PLCs are used. They may be configured individually and the modules required for practical applications are inserted in a rack. Individual modules are linked via a bus system.

Three examples of modular PLCs are shown in the Figures 22,23 and 24:



Figure 22: Simatic S7-300  
(Siemens)



Figure 23: SLC 500  
(Rockwell)



Figure 24: Modicon M340  
(Schneider)

The intern structure of a PLC is shown in the Figure 25:

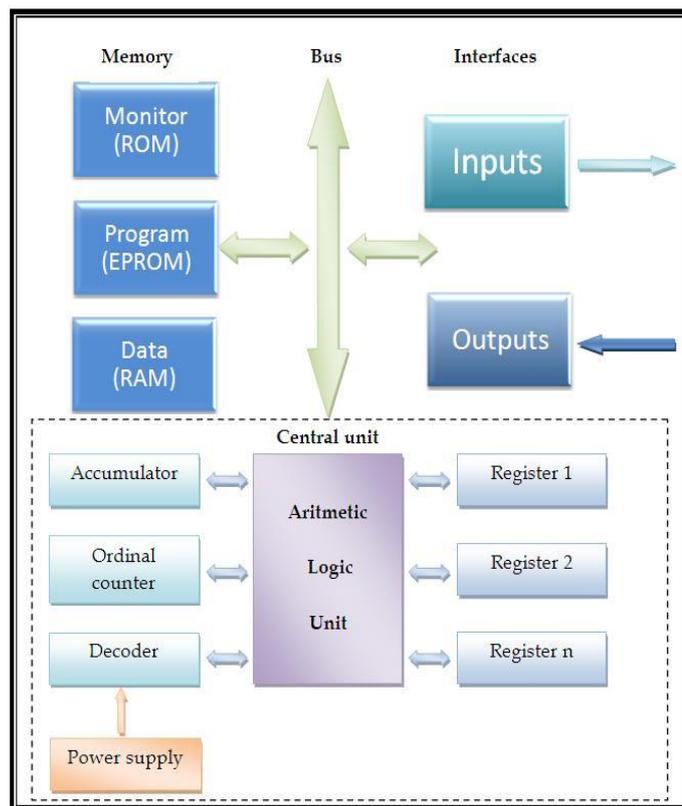


Figure 25: The intern structure of a PLC (e-document [16])

### 6.3 Processor

It is like a heart for the PLC, it is in the central unit (CU) and its performances depend on the processor's powerfulness. The central unit is an electronic card builds around processor ship (s) and provides this kind of functions:

- Logical operations on bits or on words
- Temporisation and counting

There are three different types of realization:

- **Wire technology**, the fastest but the most expensive too, only for particular purpose.
- **Microprocessor technology**, the cheapest, because uses microprocessor produced in large number
- **Mixed technology**, some operations are with wire, to raise the speed. To provide the connection between the CU and the I/O cards it must have a coupler. (*e-document [16]*)

### 6.4 Input/Output module (I/O)

They provide the interface role of the command unit. There are a command part and an operative part. The command part recovers information about process status and the operative part is made by actuators that act physically on the process.

The I/O module collects all the data and provides a good communication with the UC. A lot of PLCs use removable module, with specially ways. (*e-document [16]*)

### 6.5 Storage elements

To store data and program, PLCs use different kind of memories.

The **Random Access Memory** (RAM) is volatile but there is a battery to save all data. The **Read-Only Memory** (ROM), contains the operating system and monitor, PROM and **EPROM** are sometimes used to specific applications or the save programs.

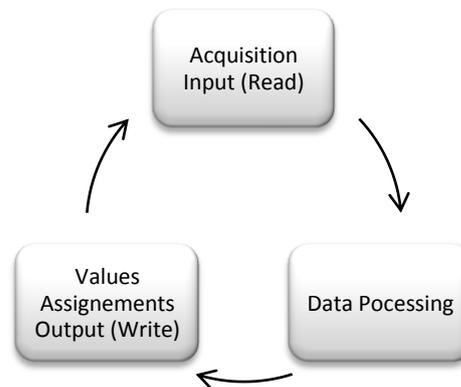
(*e-document [16]*)

## 6.6 Auxiliary

Mainly there is the power supply, which aims to provide right continuous tensions (5V,12V,...) taking into account, the little network disconnections. There is a fan, very important for frame with a lot of modules or in case where the ambience can become warm. The rack, whose supports the entire PLC. An indicator states that show the presence of a tension, a program execution, etc (*e-document [16]*)

## 6.7 Programming

The program has to provide all the time an operative cycle that has three kinds of tasks, as shown in Figure 26:



**Figure 26: Operative cycle** (*e-document [16]*)

The instruction execution is done in order of their storage in program memory, incrementing a counter-computer ordinal initially containing the start address of memory. It allows the transition to the next instruction, unless a "skip forward" can moved further away, at an address higher, are breaching not a set of instructions. (*e-document [16]*)

## 6.8 Language

By default, four languages can be used. The choice depends of the application and also preferences programmer. The four languages are:

### 6.8.1 *Instruction List (IL)*

It is a textual language, a kind of assembly language used to programme microprocessor. An instruction starts a line and in this line, there is an operator with one or more operand. For example in Table 2, the operator LD is used to load a value, ST to save this value. (*e-document [4],[5]*)

**Table 2: Example of IL language (*e-document [4]*)**

Label	Operator	Operand	Comment
FAB2	LD	%IX5	start
	ST	%MX11	stage 2

This language is not convenient because it is difficult to have a graphic analyse and the programming is complex. This language is not often used. (*e-document [4],[5]*)

### 6.8.2 *Structural Text (STL)*

The Structural Text language is an informatics language. It presents some analogies with the Pascal language. It uses expressions, ordered assembly of operator and operand with priority, as such:

$$E:=(A>B) \text{ AND } (C>D)$$

It allows to evaluate the right part and to assign the value in E. This language is used when there are a lot of complex algorithms with numerical treatment. For Boolean function it is never used. (*e-document [4],[5]*)

### 6.8.3 Ladder Diagram (LAD)

This language uses a graphic shape, with contacts. The application is introduced by one or more network (s) and a network is formed by graphical elements or function blocs connected between them by lines, Figure 27. Lines start on the left (alimentation) and ended on the right (Output).



Figure 27: Example of LAD (e-document [4],[5])

This language is very efficient for combinatorial systems and it is used a lot in industries. But LAD is difficult to program with time functions or for independent outputs. (e-document [4],[5])

### 6.8.4 Function Bloc Diagram (FBD)

FBD is a language very close than LAD language. In this language contacts are replaced by boxes, Figure 28. Inputs are on the left and outputs on the right.

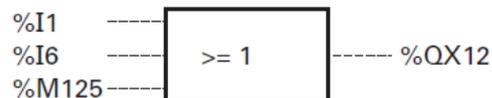


Figure 28: Example of FBD (e-document [4],[5])

Blocs are founded in a library or are created by the users. A very interesting thing is that with the use of blocs, mistakes are decreased. Restrictions are the same than LAD, efficient for combinatorial systems but difficult to program with time functions or independent outputs. (e-document [4],[5])

## 6.9 PLC in the coiler drive

Brand:

**SIEMENS**



Figure 29: Simatic S7-300

Series: SIMATIC S7-300

DC 24V	<b>PS307</b> 5A	<b>CPU315-2 DP</b>  SIMATIC S7-300	<b>SM321</b> DI 32xDC 24V	<b>SM322</b> DO 32xDC 24V/0.5A
307-1EA00-OAAO		315-2AF03-OABO	321-1BL00-OAAO	322-1BL00-OAAO

**Power Supply (PS):** 307 5A DC 24V

**Central processing unit: (CPU):** 315-2 DP

**Signal modules (SM):** 321 DI 32xDC 24V and 322 DO 32xDC 24V/0.5A

## 7 Sensors in the coiler drive

Two main sensors are present in the coiler drive process, a position sensor to measure the tension of the nylon web and infrared sensor to measure the speed of the nylon web.

### 7.1 Tension sensor

Its task is to measure the tension in the nylon web. For that, this sensor is divided in two parts. The first one is the inductive linear displacement transducer. The inductive displacement transducer is made by two coils which are encapsulated in a stainless steel cylinder and it contains a metal plunger core causes opposing changes of inductance when it is displaced through the center of the coils.

The second part is the transducer module. It has been designed to demodulate the measuring signal to obtain a DC-output signal proportional to the measuring stroke of the inductive linear displacement transducer. (*Zarza & Garcia 2005*)

On the Figure 30, the tension sensor in the coiler drive is shown.

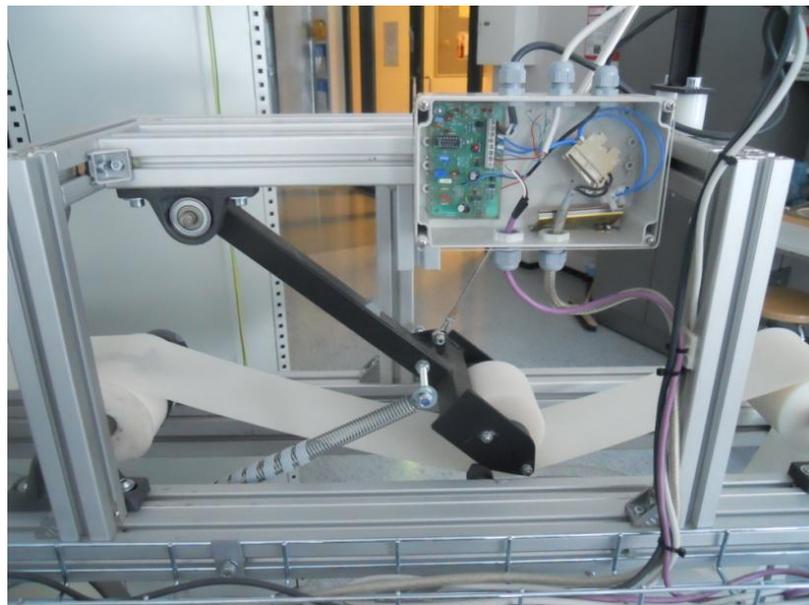


Figure 30: Tension sensor in the coiler drive

## 7.2 Speed sensor

A speed sensor is necessary to measure and to control the speed of the nylon web. A system of infrared measuring has been chosen. The design of a plate with holes has been fixed to the auxiliary cylindrical coil. This method is cheap and it is also used in industry. Three parts compose this sensor.

The first one is the infrared light encoder (OMRON E3JK-DS30M1), it is put in the front of the speed rotating plate and sends a signal to the plate which is reflected and it returns to the infrared light sensor. The second one is the speed rotating plate fixed to the auxiliary coil. It was designed in order to achieve fifty pulses per revolution. The last part is the frequency to voltage converter (AXIS 10 ATP2/3) that obtains a frequency signal from the infrared light encoder. The frequency converter needs a voltage signal, so it is required to change the signal by means of the converter. Therefore, the converter transforms the frequency signal in a voltage signal for the frequency converter. (*Zarza & Garcia 2005*)

On the Figure 31, the speed sensor in the coiler drive is shown.



**Figure 31: Speed sensor in the coiler drive**

## 8 Software for the frequency converter

In this chapter, the two different software employed for the frequency converter will be introduced. The purpose is to show you with which software is used with which elements and what are their functions. Basic functions will be introduced, more information about software can be found in the online-manual, on the constructor's website<sup>1</sup>.

### 8.1 NC Load

Two software from Vacon are used to manage the frequency converter, NC Load and NC Drive. The Vacon NC Load, Figure 32, is a basic tool for downloading.

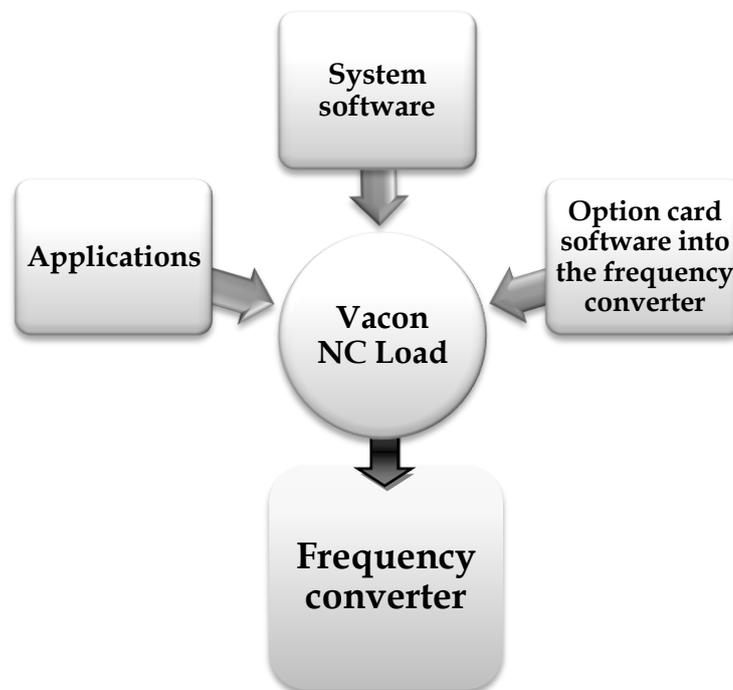


Figure 32: Vacon NC Load software (VACON 2010)

---

<sup>1</sup> Vacon website: [www.vacon.com](http://www.vacon.com)

The graphical user interface, Figure 33, provides an easy point-and-click selection of applications to be downloaded.

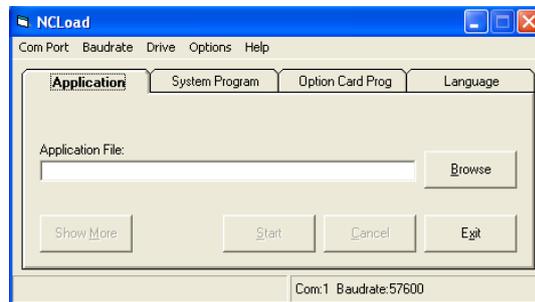


Figure 33: NC Load main window

## 8.2 NC Drive

The Vacon NC Drive, Figure 34, is the commissioning and monitoring tool. It allows to:

- Download and upload parameters set between the drive and the PC
- Compare parameters set
- Change the active application
- Save and print parameters and service reports to file or paper
- Control the drive
- Set references
- Operate the NXP data logger, and more.

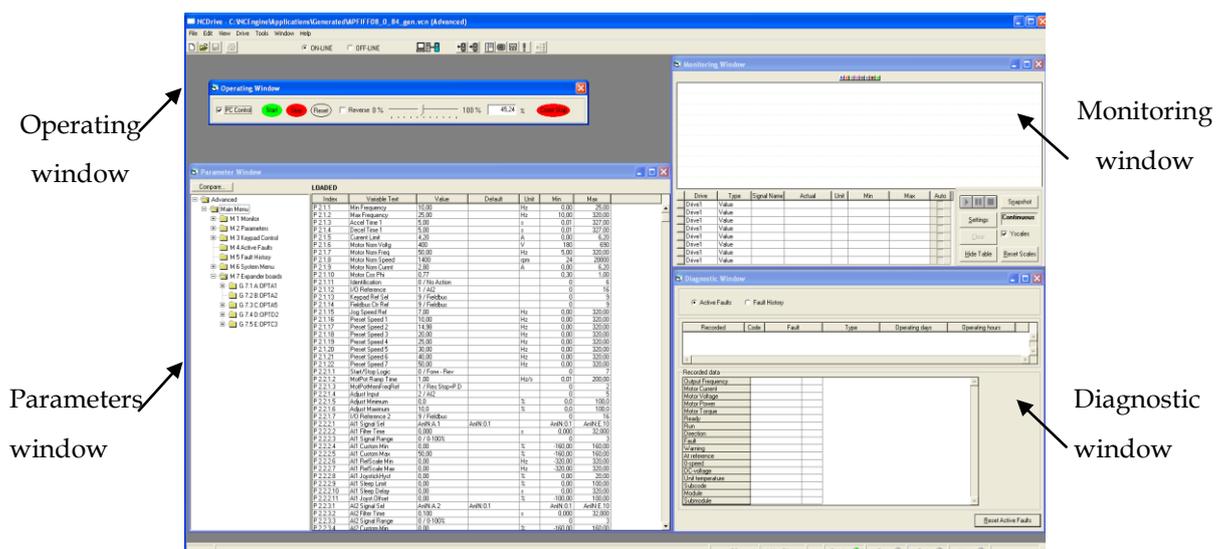


Figure 34 : FC Drive main window

The Vacon NC Drive also allows to monitor up to eight user-specified variables simultaneously on a graphical trend screen. In the Vacon NXP, it can also operate the data logger and communicate via a CAN network with up to 254 drives. (VACON 2010)

## 9 Software for the PLC

Two different software are also used to manage correctly the PLC. The first one is STEP-7 for the PLC program and the second one is WinCC for the human-machine interface. Both are made by Siemens and they are in the SIMATIC suite.

### 9.1 STEP-7

This chapter explains the notions about the STEP-7 software in a general case. All the steps to create a STEP-7 project for the coiler drive (creation of a new project, hardware configuration, programming and downloading to the PLC) are explained in the “*User manual - Coiler drive project creation with STEP-7*” (appendix 3).

#### 9.1.1 *Presentation of STEP-7*

This software allows having an access to Siemens PLC, in our case, the SIMATIC S7-300. The standard STEP-7 packages comprise a number of applications that you can see on the Figure 35.

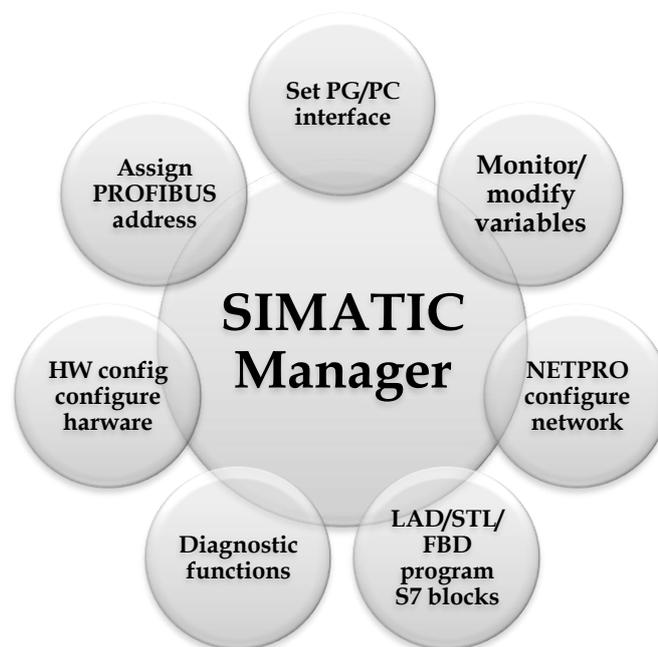


Figure 35 : Standard STEP-7 packages (*e-document [3]*)

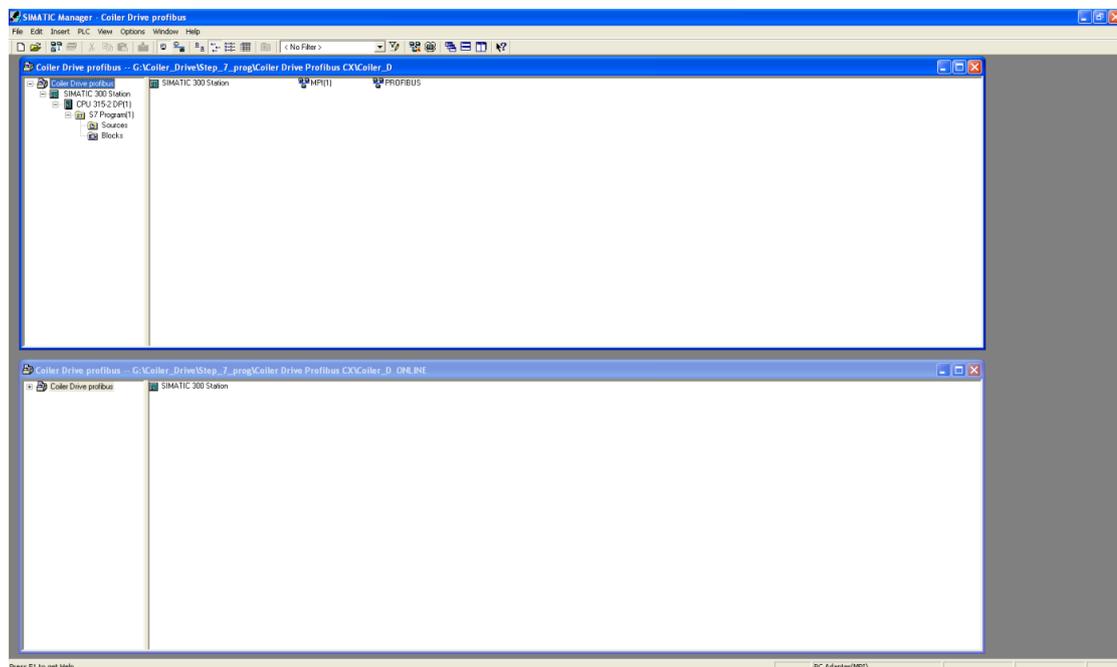
Each of which does a specific job when programming an automation task, such as:

- Configuring the hardware and settings its parameters
- Configuring networks, connections and interfaces
- Creating and debugging user programs

Additional optional software tools are available to expand the standard STEP-7 package for particular applications. These include programming language packages such as SCL, S7GRAPH or HiGraph. (*e-document [3],[4]*)

SIMATIC Manager collects all the data and settings necessary for an automation task and combines this information into a project. Within this project, all data provides comprehensive online help including context-sensitive help for selected folders, objects and error messages. (*e-document [3],[4]*)

STEP-7 project is divided into folders and objects, Figure 36. Folders are objects which can contain other folders and objects.



**Figure 36: STEP-7 - Main window**

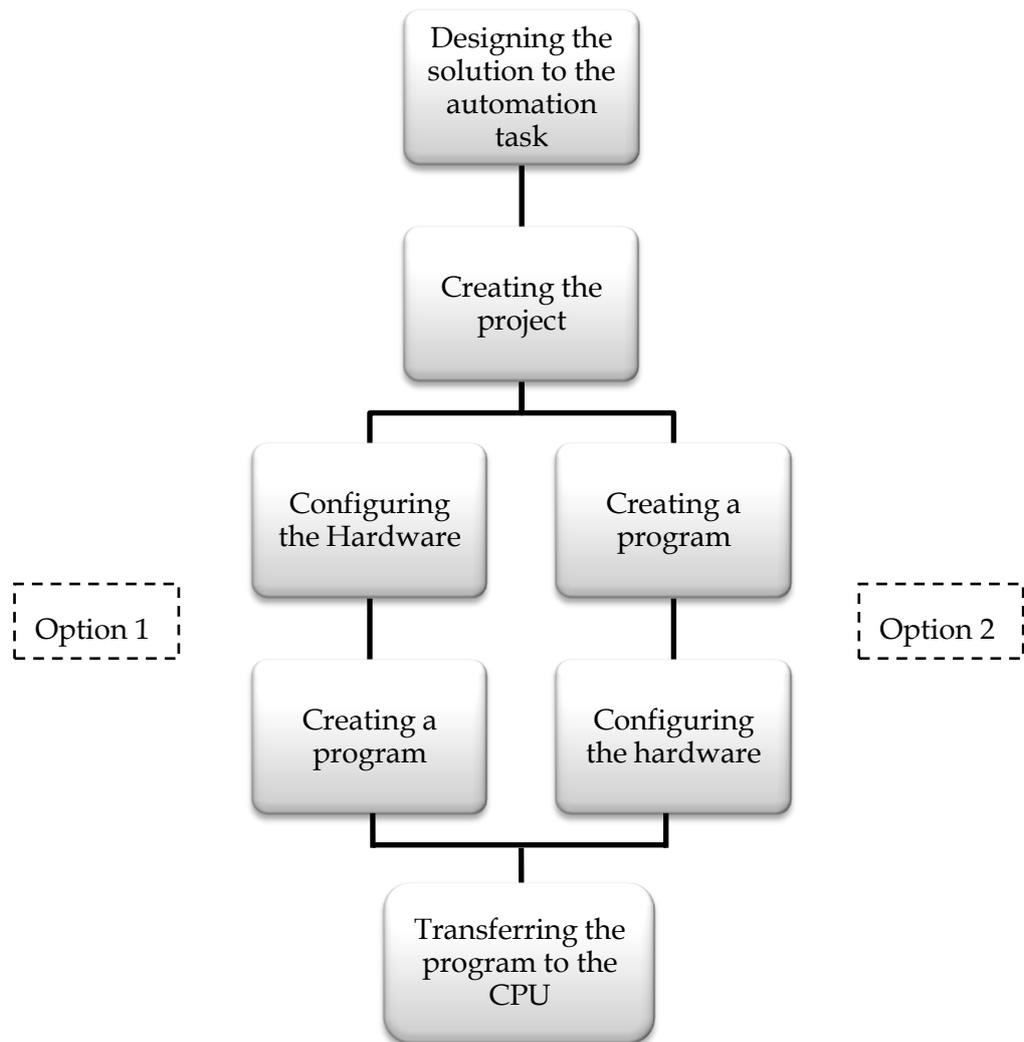
It can program an individual controller in different languages and takes into account the network of PLC, which allows access to any network controller for programming and the possibly to automatic send messages between them. (*e-document [3],[4]*)

The first step in STEP-7 is to create a project. In a project, there are the full descriptions of your automation. It consists of two main parts:

- The description of the material
- The functional description (the program)

A wizard is there to help you in the project's configuration, you can choose to use it or not. Sometimes it is better to not use the wizard; it configures default poorly on the controller. (*e-document [3],[4]*)

Then you can choose two different ways: configuring the hardware or creating a program, as seen on Figure 37:



**Figure 37: Option 1 and option 2 (*e-document [5]*)**

If you are creating programs with a lot of inputs and outputs, it is recommended to configure the hardware first. The advantage of this is that STEP-7 displays the possible addresses in the Hardware Configuration Editor (HW Config).

If you choose the second option, you have to determine each address yourself, depending on your selected components and you cannot call these addresses via STEP-7. (e-document [3],[4])

### 9.1.2 Hardware configuration'

Let's take the first option and add a station, here S7-300 to our project. After that, we need to define the entire hardware configuration, Figure 38:

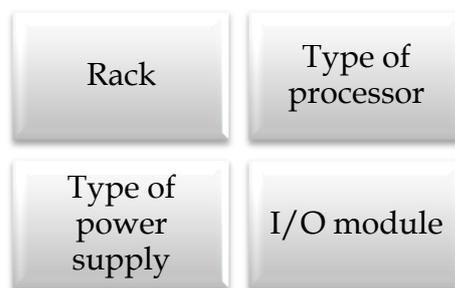


Figure 38: Hardware configuration (e-document [5])

The last step in the hardware's configuration is to define all the symbols in the symbols table. It's a very important part because symbols will be employed for programming and they are directly connected to an address in the I/O module.

As seen in Figure 39, the I/O module has different addresses.

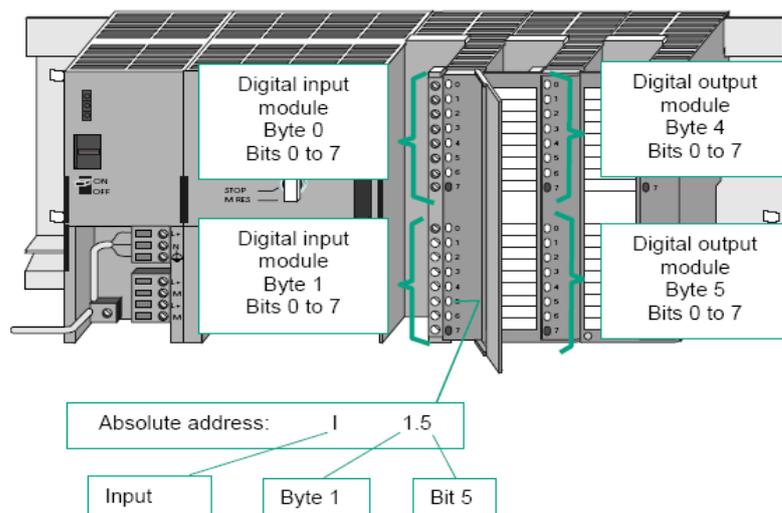


Figure 39: Addresses in the PLC (e-document [5])

The bytes 0 and 1 are used for the input addresses and the bytes 4 and 5 for the output addresses.

The letter I is used for Input addresses (I like Input) but we don't use the letter O for output, it's too close than 0 (zero) so the letter Q is used, Q 4.3 for example for an output address. (*e-document [5]*)

### 9.1.3 The program

The program will be placed in the PLC, its name is OB for Organisation Block. The main program will be in the OB1 and in this block we can call other functions like FB or FT for example. To program in the OB, we need to choose between three main languages (LAD, FBD or STL). By default, there is no IL language in STEP-7.

(*e-document [4]*)

#### LAD language

As seen in the chapter 7.8.3 the LAD language is a series of networks that will be covered sequentially. The inputs are represented by switches or inverted input, release by coils or flip flops. There are also unary operations (I/O) inverter pending a face amount or down. The outputs are mandatory right of the network must be clearly defined our I/O, either directly by their code, or with their name defined in the symbol table (enter name in quotation marks). It connects the elements in series to the AND function and in parallel to the OR, as seen in the Figure 40.

#### Ladder Logic (LAD)

Suitable for users from the electrical engineering industry, for example.

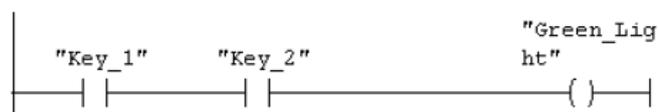


Figure 40: STEP-7 - LAD language (*e-document [4]*)

We may use internal bits (can be used in coils and switches), as used in a calculator memory for example to store an intermediate result. We can also introduce more complex elements, in particular operations on bits such as a flip-flop SR (Priority release) or RS (priority engagement), POS and NEG for the detection of fronts. In the manual "CONT S7" other useful functions like meters, tempos, etc can be founded. Generally, the program is broken down into several networks and the networks are executed sequentially. (*e-document [4]*)

## FBD language

Exactly the same than LAD language but the presentation is different, as seen in the Figure 41. (*e-document [4]*)

### Function Block Diagram (FBD)

Suitable for users from the world of circuit engineering, for example.

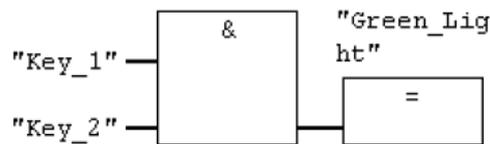


Figure 41: STEP-7 - FBD language (*e-document [4]*)

## STL language

It is a textual language which is closest to the internal behaviour of the PLC (corresponding roughly to the assembler in a computer). The system always knows translate LAD, FBD or LT, but not vice versa. The program consists of a sequence of lines, Figure 42, each specifying an operation code followed by an operand, only one. The operand can be an absolute address (I 0.3) or a symbol in quotes (if symbols were defined!). (*e-document [4]*)

### Statement List (STL)

Suitable for users from the world of computer technology, for example.

```
A    "Key_1"  
A    "Key_2"  
=    "Green_Light"
```

Figure 42: STEP-7 - STL language (*e-document [4]*)

### 9.1.4 Transfer to the PLC

After programming and saving the project, it must be transferred to the PLC. It is better to put the controller in STOP statue. When the program is loaded, put the PLC in RUN-P statue and in the ONLINE window, the state variables can be directly visualized in the program. Some tests are possible with the debug mode.

In LAD, Figure 43, the patterns become dotted with places where the powers fails and the patterns become green where there is powers.

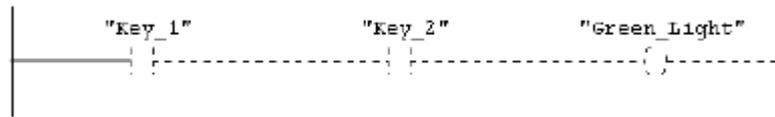


Figure 43: Debugging with LAD (e-document [5])

In FBD, Figure 44, the signal state is indicated by "0" and "1." The dotted line means that there is no result of logic operation.



Figure 44: Debugging with FBD (e-document [5])

In STL, Figure 45 a table is displayed next to the program, specifying the values (0 or 1) of the operands. For Statement List the following is displayed in tabular form:

- Result of logic operation (RLO)
- Status bit (STA)
- Standard status (STANDARD)

		RLO	STA	Standard
A	"Key_1"	0	0	0
A	"Key_2"	0	0	0
=	"Green_Light"	0	0	0

Figure 45: Debugging with ST (e-document [5])

We can also list the status of all variables, or monitor them. (e-document [5])

### 9.1.5 More Helps

It is documented in the software (menu or F1), but also in the start menu under SIMATIC → Documentation → English. The online help (F1 on a component if it has forgotten the details of its entries, for example) is really useful.

## 9.2 WinCC

After a short introduction of Human-Machine Interface, it will be explained in general the different notions about the WinCC software. For more information about the HMI in the coiler drive, the file: “

### 9.2.1 *HMI introduction*

Before to explain what exactly WinCC is, let's see what mean an HMI (Human - Machine Interface). A HMI system represents the interface between man (operator) and process (machine/plant).

On the Figure 46, the device OP 77b is used with an HMI program.

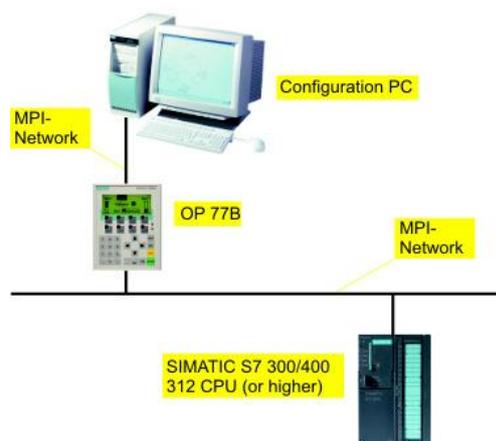


Figure 46: Process with an HMI device (*e-document [6]*)

The PLC is the actual unit which controls the process. Hence, there is an interface between the operator and WinCC (at the HMI device) and an interface between WinCC and the PLC. An HMI system assumes the following tasks. (*e-document [6]*)

#### **Process visualization**

The process is visualized on the HMI device. The screen on the HMI device is updated very often, almost dynamically.

#### **Operator control of the process**

The operator can control the process by means of the graphical user interface GUI. The operator can preset reference values for the controls or start a motor for example.

### **Displaying alarms**

Critical process states automatically trigger an alarm, when the set point value is exceeded for example.

### **Archiving process values and alarms**

The HMI system can log alarms and process values. This feature allows you to log process sequences and to retrieve previous production data.

### **Process values and alarms logging**

The HMI system can output alarms and process value reports. This allows you to print out production data at the end of a shift, for example.

### **Process and machine parameter management**

The HMI system can store the parameters of processes and machines in recipes. For example, you can download these parameters in one pass from the HMI device to the PLC to change over the product version for production. Now, let's see more information about WinCC. (*e-document [6]*)

#### **9.2.2 Presentation of WinCC**

It is a software made by Siemens to create HMI interface between the person and the machine. WinCC allows the operation and observance of the process that run in a machine. The communication between WinCC and the machine takes place via an automation system. Different objects are necessary to be configured to monitor and operate the system, such as:

- Screens to show and operate the processes on the control device
- Tags to transfer data between the operating device and the installation
- Archive to store the process data
- Alarms to indicate the operating status of the system on the operating device

In order that the operator can perform tasks, the HMI device must be configured. The configuration steps necessary to do this, Figure 47, are explained here. You can find also the practical example for the coiler drive in the manual “User manual – Coiler drive HMI configuration with WinCC” (Appendix 4).

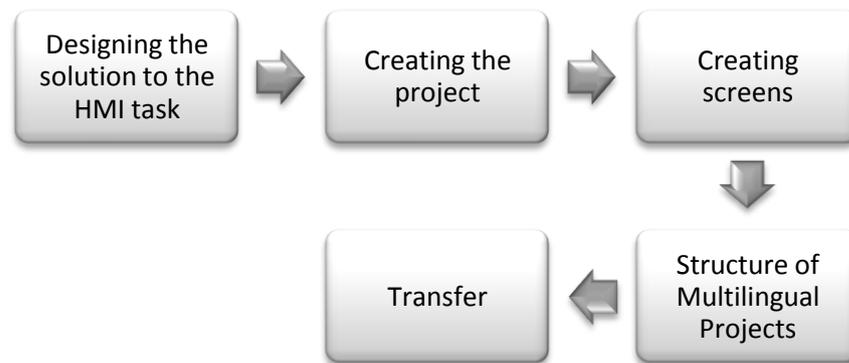


Figure 47: Configuration steps in WinCC (e-document [6])

A project is the basis for configuring the user interface; all the objects will be created and configured in it. By objects it means .

- Screen
- Tags
- Alarms

### 9.2.3 The Screen

The main screen, Figure 48, contains different parts. In the **Project View**, there is the tree structure that contains all the configurable elements. Projects are edited in the **Work Area**. All elements are arranged on the borders of the work area. The **Property View** is used to edit object properties, e.g. the colour of text.

The **Toolbox** contains a selection of objects that you can add to your screens, for example, pictures, images objects or operate control elements.

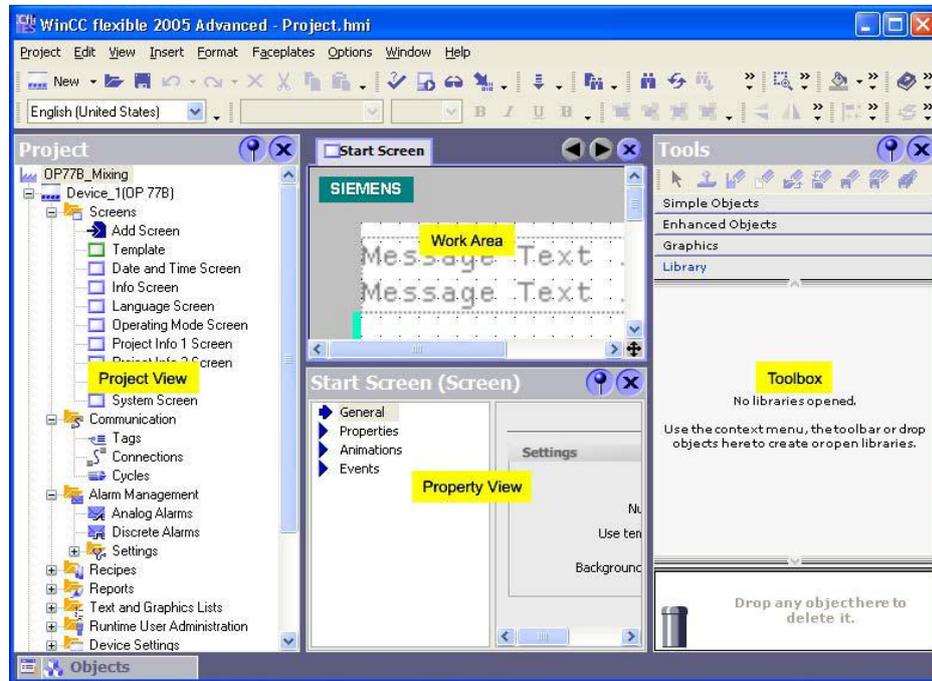


Figure 48: WinCC - Main screen

## 10 HMI created for the coiler drive

The first screen in the human-machine interface is the start screen, Figure 49.



Figure 49: HMI - Start screen

It allows to choose the language and gives a view of the coiler drive. Choose the language with a left click on the flag and then the main screen will be opened, Figure 50.



Figure 50: HMI - Main screen

First, click on the button to energize the coiler drive. Make sure that the coiler drive is properly connected with the power supply. If everything is right, the red LED control becomes green, both frequency converter turn on and both electric engines are energized.

After that, set the parameters in the Step 1, Figure 51, (Speed web, acc/deacc and Tension web). Click on the 0 and enter the correct values.



Figure 51: HMI - Step 1



Figure 52: HMI - Step 2

Then, the second step is to choose the wind mode (wind or unwind), click on the selector to change the mode, Figure 52.

When the step 1 and 2 are done, the PLC has all the settings and parameters to manage the coiler drive properly. Therefore, in step 3, motor brakes can be turn off, by clicking switch 1 and 2, Figure 53. Then the motor control LEDs become green, Figure 54.

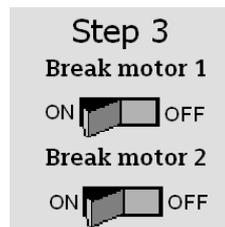


Figure 53: HMI - Step 3

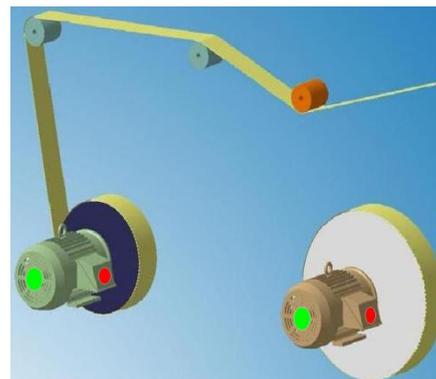


Figure 54: HMI - Motor control LED

Before clicking on the green button in Step 4, check if the coils are really free to turn and if everything is ready. The coiler drive has to be free of tools and other kind of objects that could create some damages. After this checking, click on the green button to start the process.

Arrows appears on the web to show the web direction, Figure 55. Data from the two frequency converters (n, T and P) are collected through the PROFIBUS network. The

same thing for the speed sensor and the tension sensor data. This print screen is done with the simulator and that is why these parameters show the default value, 0.

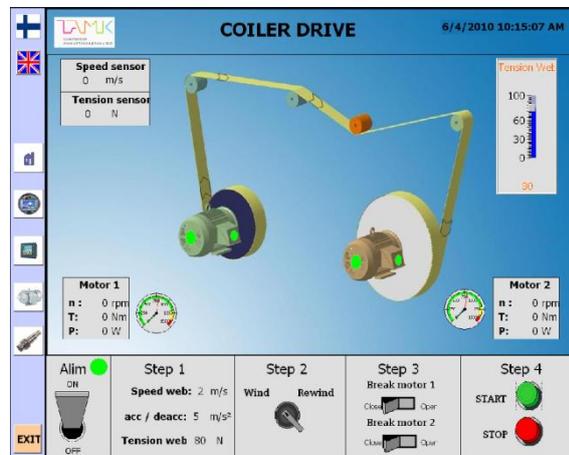


Figure 55: HMI - Process running

It is possible to change parameters even if the coiler drive process is on. For the tension web, for example, the button can be drag to rise or to decrease the tension in the web. Other screens are done to show more information. On the left side, in the navigation menu, click on the third icon to activate the trend screen, Figure 56.

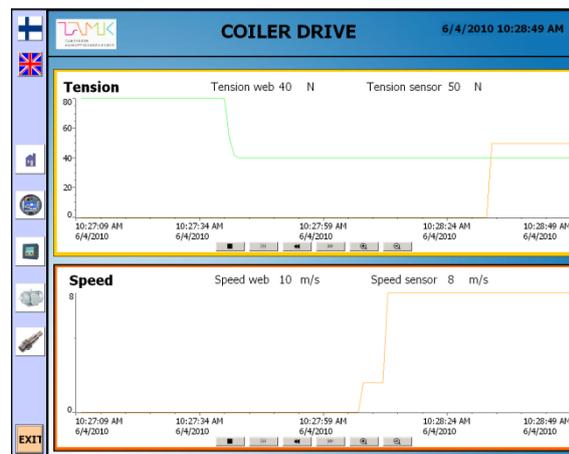


Figure 56: HMI - Trend screen

This screen shows four different trends, as:

- Tension web (in green)
- Tension sensor (yellow)
- Speed web (orange)
- Speed sensor (red)

It is possible to stop the trends to do some measurements.

The last screen is done to show all the available information, through the PROFIBUS network, about both electric motors, Figure 57.

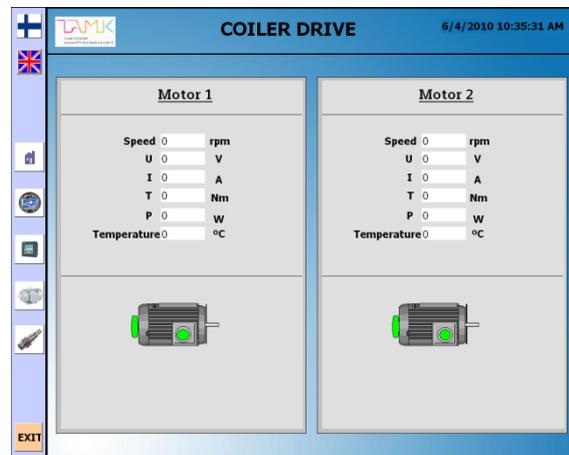


Figure 57: HMI - Electric motor screen

To stop the coiler drive process, push on the red button and close the two brakes with the switches in step 3.

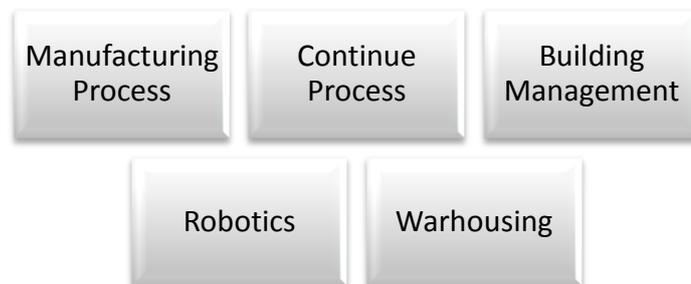
For every screen it is possible to change the language between English and Finnish by clicking on the flag on the top left of the navigation menu.

## 11 Field bus solution

The choice was to use a PROFIBUS network. PROFIBUS means PROcess FIEld BUS and it is a field bus based on a shielded twisted pair or an optical network using fiber optic cable. Using industrial field bus technology, considerable savings can be made particularly in the mechanical installation, fitting and wiring of the plant equipment due to reduce cabling for distributed I/O devices. (*e-document [16]*)

### 11.1 Introduction

In 1987, the German industry has therefore initiated the PROFIBUS cooperative project. The regulation and norms developed by this body were documented in the German norm DIN 19245. Since 1996, national standard became the international EN 50170 standard in Europe, since 1999. PROFIBUS is even in the international CEI 61158 standard and it takes an important place in the global market of field bus. There are more than 15 millions devices installed in 1 millions of network and allows to answer to a large type of applications as such: on the Figure 58.



**Figure 58: Applications with PROFIBUS network (*e-document [16]*)**

PROFIBUS is composed of three variants called:

#### **11.1.1 PROFIBUS-FMS (*Field Message Specification*)**

First PROFIBUS application used, it is an application with an exchange between the masters to synchronize activities, use the MMS (Manufacture Message Specification) technology. Now it is not really used because the "Ethernet technology" replaced it. (*e-document [16]*)

### 11.1.2 PROFIBUS-DP (Decentralized Peripheral)

Used for applications type master-slave, with device management and a very short access time. Speeds are until 12 Mb/s and more than 90% of PROFIBUS applications are DP type. (e-document [16])

### 11.1.3 PROFIBUS-PA (Process Automation)

PROFIBUS-PA is used to control applications of process with a communication with actuators or sensors. The alimention and the signal are on only one wire. (e-document [16])

The Figure 59 shows the three PROFIBUS variant in a process:

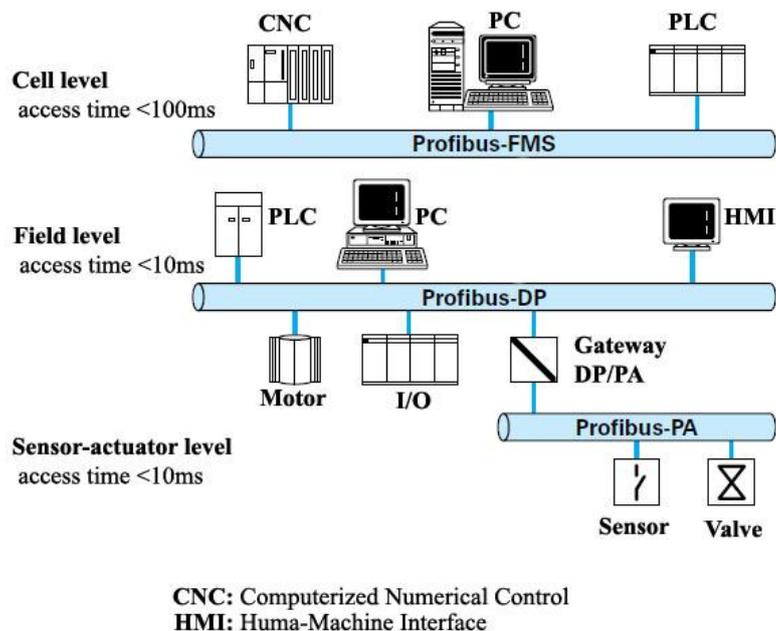


Figure 59 : Classification of PROFIBUS Networks (e-document [16])

The PROFIBUS network allows to have:

- Direct control of the frequency converter (Run, Stop, Speed reference, Fault, reset, ...)
- Full access to all parameters
- Monitor status (Output frequency, Output current, ...)

As the coiler drive PROFIBUS network used is the DP variant, only this variant will be introduce here. (e-document [16])

## 11.2 PROFIBUS-DP Layer

The specification of the PROFIBUS-DP standard cover layers 1 and 2 in the Model OSI (Open System Interconnection). The layer 1 is the physical layer, characterizes supports of transmission, three standard are used, RS485, optic fibber (OF) and CEI 61159-2.

The layer 2 is the data link layer, it calls FDL for Field bus Data Link layer, and it characterizes access procedures, transmission services and telegram structures.

PROBIFUS-DP is based on a semi-duplex, asynchronous, gap-free synchronization. The connection is on only one wire; positives point is to decrease the length, easy to add stations and works even if a station is out of order. Negative point, only one station can emit at a time. (*e-document [16]*)

### 11.2.1 Physical Layer (layer 1)

The coiler drive PROFIBUS network is equipped with a RS 485 transmission. The wire is a bi-axial cable, its colour is purple, Figure 60.

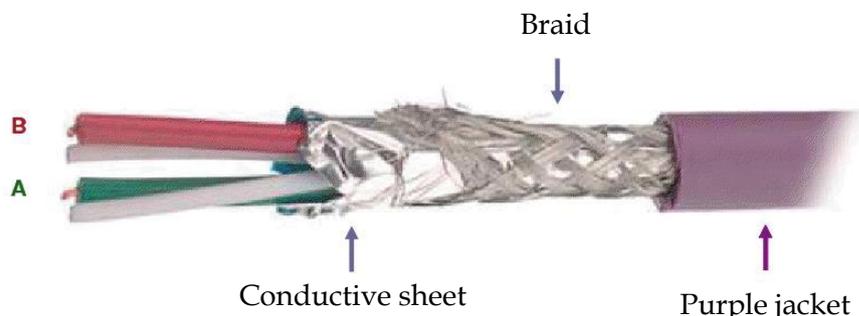


Figure 60 : PROFIBUS cable structure (*e-document [16]*)

Data are transmit in an 11 bit character frame, see Figure 61 in Non Return to Zero (NRZ).

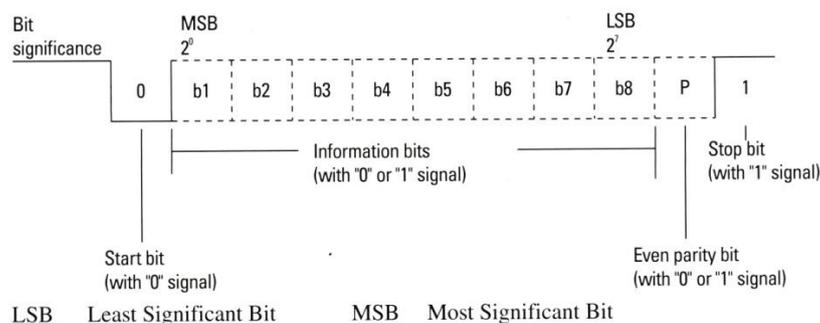


Figure 61 : PROFIBUS UART character frame (*Decentralization with PROFIBUS-DP 2000, 17*)

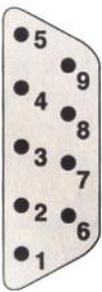
During the transmission, a binary "1" corresponds to a positive level on line RxD/TXD-P (Receive/Transmit-Data-P) as opposed to RxD/TxD-N (Receive/Transmit-Data-N). In specialized literature, the two PROFIBUS data line are also referred to as the A line and the B line. A line corresponds to the RxD/TxD-N signal and the B line to the RxD/TXD-P signal.

*(Decentralization with PROFIBUS-DP 2000)*

### 11.2.2 Bus connection

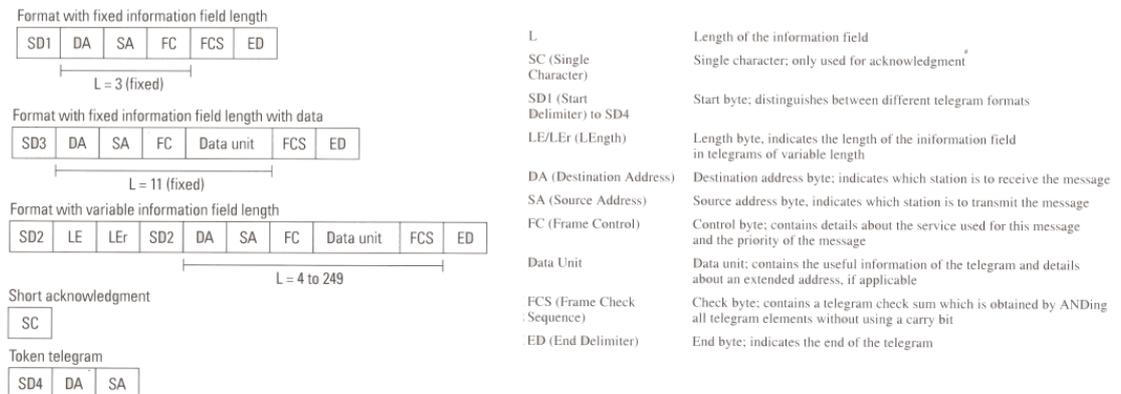
The standard EN 50 170 recommends a 9-pin sub D plug connector, details in the Table 3.

**Table 3 : Pin assignment of the 9-pin sub D plug connector** *(Decentralization with PROFIBUS-DP 2000, 18)*

View	Pin No.	Signal Name	Designation
	1	SHIELD	Shield or function ground
	2	M24	Ground of the 24V output voltage
	3	RxD7TxD-P	Receiving/sending data-plus B line
	4	CNTR-P	Signal for direction control P
	5	DGND	Data reference potential (ground)
	6	VP	Supply voltage-Plus 24V Plus of the output voltage
	7	P24	(auxiliary power)
	8	RxD7TxD-N	Receiving/sending-data-Minus A line
	9	CNTR-N	Signal for direction control N

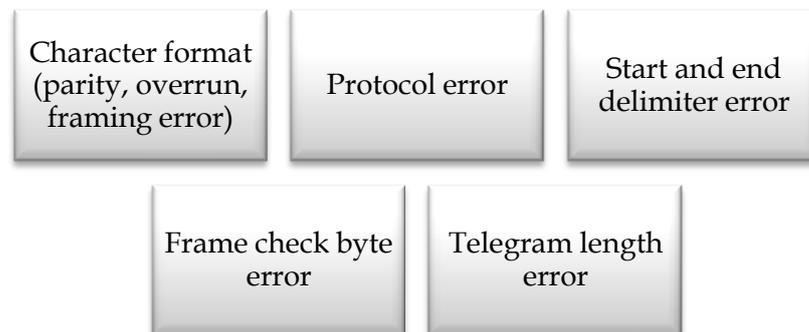
### 11.2.3 Field bus Data Link (Layer 2)

Telegram formats, as seen on Figure 62, provide a high degree of transmission security.



**Figure 62: PROFIBUS telegram formats (Decentralization with PROFIBUS-DP 2000, 24)**

The following types of errors can be detected, Figure 63.



**Figure 63: Errors who can be detected (e-document [16])**

The addresses to 0 from 127 can be used; there are therefore 128 addresses usable. Normally the address 0 is used by diagnostic tools, address 126 is for devices who receive their address by the bus and 127 is the distribution address. Addresses from 1 to 125 can be freely employed for masters and slaves. (e-document [16])

### 11.2.4 Bus Access Control

PROFIBUS use a hybrid bus access control mechanism. It consists of a decentralized token passing procedure for communication between the active nodes (master) and the centralized master-slave procedure for communication between the active and the passive nodes. When an active node (bus station) has the token, it takes over the master function on the bus to communicate with both passive and active nodes.

The exchange of messages on the bus is organized by means of node addressing. Each PROFIBUS node is given an address which must be unique throughout the entire bus system. (*Decentralization with PROFIBUS-DP 2000*)

However in the coiler drive PROFIBUS network, there is only one master and only one slave therefore, there is not really this kind of notion. There is still a token even in DP mono master structure, as seen in the figure 64.

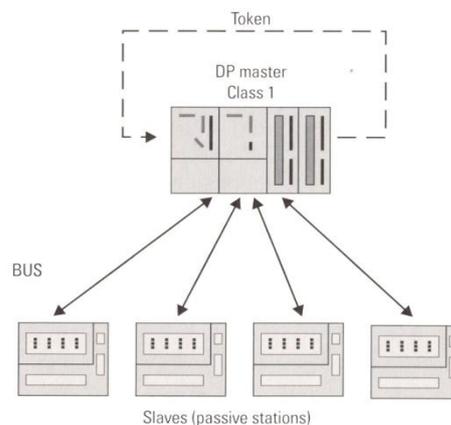


Figure 64 : DP mono-master structure (*Decentralization with PROFIBUS-DP 2000, 34*)

### 11.3 PROFIBUS-DP in SIMATIC S7 system

PROFIBUS is an integral part of SIMATIC S7 systems. The I/O peripherals are totally integrated in the system by the STEP-7 configuration tool. SIMATIC S7 provides integrated or plug-in PROFIBUS-DP interfaces for the connection of field devices with more complex technical functions. There are two types of PROFIBUS-DP interfaces in SIMATIC S7-300. The first type is to have DP interfaces integrated on the CPUs. This is that configuration that is chosen for the coiler drive, the CPU used is the CPU 315-2DP. The other type of interfaces is to use a Plug-in DP interfaces through an Interface Module (IM) or a Communication Processor (CP). (*Decentralization with PROFIBUS-DP 2000*)

With the CPU 315-2 DP it is possible to choose between master or slave function. It can be operate both, as DP master or as DP slave. In the coiler drive process, the PLC is of course the master. In the Table 4, it is shown the technical data of the integrated PROFIBUS-DP CPU 315-2 DP interface. (*Decentralization with PROFIBUS-DP 2000*)

**Table 4: Technical data of the integrated PROFIBUS-DP interfaces (*Decentralization with PROFIBUS-DP, 2000, 48*)**

Module	CPU 315-2 DP	
MILFB	6ES7 315-2AF03-0AB0	
Number of interfaces	2 (1 <sup>st</sup> interface for MPI only)	
Operating mode	DP master	DP slave
Baud rates Kbit/sec	9,6 to 12 000	9,6 to 12 000
Max. number of DP slaves	64	-
Max. number of modules	512 total	32
Input bytes per slave	244 max	-
Output bytes per slaves	244 max	-
Input bytes as slave	-	244 max
Output bytes as slave	-	244 max
Consistent data modules	32 bytes max.	32 bytes max.
Usable input area	1 Kbyte	
Usable output area	1 Kbyte	
Max. parameter data per slave	244 bytes	
Max. config. data per slave	244 bytes	

Max. diagnostic data per slave	240 bytes	
Cross communication support	Yes	Yes
Constant bus cycle time	Yes	-
SYNC/FREEZE	Yes	No

### ***11.3.1 Generic Station Description (GSD) files***

GSD files are ASCII files for DP slave contain characteristic device properties of these DP components. The characteristics are standardized and the syntax is fixed, it allows to check PROFIBUS device data. (*Decentralization with PROFIBUS-DP 2000*)

Characteristics are for example:

- Name of the manufacturer
- Software and hardware version
- Designation devices
- Transmission rates allowed

It is used in our case to be able to install the Vacon frequency converter in the PROFIBUS Network. The appendix 2 gives more information about the PROFIBUS network and how to install and use the Vacon GSD file.

## 11.4 PROFIBUS network STEP-7 program

To be able to read and write data through the PROFIBUS network, some special communication functions have to be used. In STEP-7 there are many communication functions but only two are the most used. The first one allows reading data and the second one is to write data into the slave, the specifications for these functions can be found in the appendix 5.

### 11.4.1 *Reading data of a DP slave device with SFC 14 "DPRD\_DAT"*

This function has to be used because it is only possible to read a maximum of four continuous bytes using load instructions. With the SFC 14 function data can be read with the maximum length fixed for each specific CPU. In our case, we need to read twelve bytes. When the reading starts, two possibilities can happen. The first one is if no error occurred during the data transfer, the data that have been read are entered in the destination area identified by RECORD. The second one is if an error occurred during the data transfer, the type of error is sent to RET\_VAL. In the documentation you can find the error information with the error code in the appendix 5. Now, let's see how the function has to be configured in STEP-7. (*e-document [12]*)

The Vacon frequency converter is in PPO1 mode. It means that eight bytes are for parameter data and four for the process data. Therefore it should have two reading function and also two writing functions.

To create the first reading function for the parameter data, in STEP-7 program, a new network has to be created in the main block (OB1) and in STL language the function SFC 14 has to be called, like on the Figure 65.

```
CALL "DPRD_DAT"          SFC14          -- Read Consistent Data of a Standard DP Slave
LADDR :=
RET_VAL:=
RECORD :=
```

Figure 65: LAD/STL/FBD - CALL SFC 14 function

Three parameters have to be defined: *LADDR*, *RET\_VAL* and *RECORD*. In the Table 5 it is explain the description for each parameter.

**Table 5: SFC 14 function description**

Parameter	Declaration	Data Type	Memory Area	Description
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured start address from the I area of the module from which the data will be read. <b>Note:</b> Addresses have to be entered in exadecimal format. For example, diagnostic address 100 means: LADDR:=W#16#64.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Destination area for the user data that were read. This must be exactly as long as you configured for the selected module with STEP 7. Only the data type BYTE is permitted.

The start address *LADDR* is a word in hexadecimal format to say which is the byte where data have to be read. In the GSD file of our Vacon frequency converter it is written that the first byte is in the address 256 (100 in hexadecimal format). Therefore this value will be entering in the *LADDR* parameter.

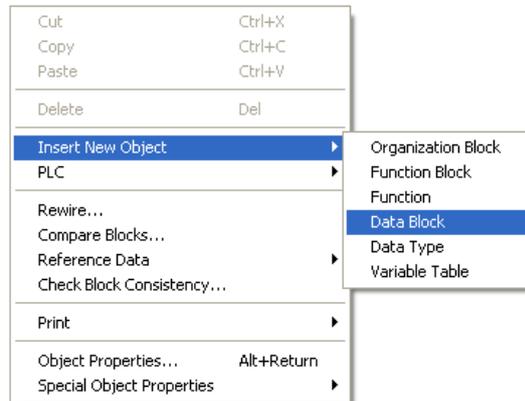
***LADDR :=W#16#100***

Second parameter is *RET\_VAL*, it is used like a diagnostic parameter. If an error occurs, the error code will be return to the address specified. It is important to choose an address not used. In our case, the address 200 in the memory will be chosen. MW means Memory Word.

***RET\_VAL:=MW200***

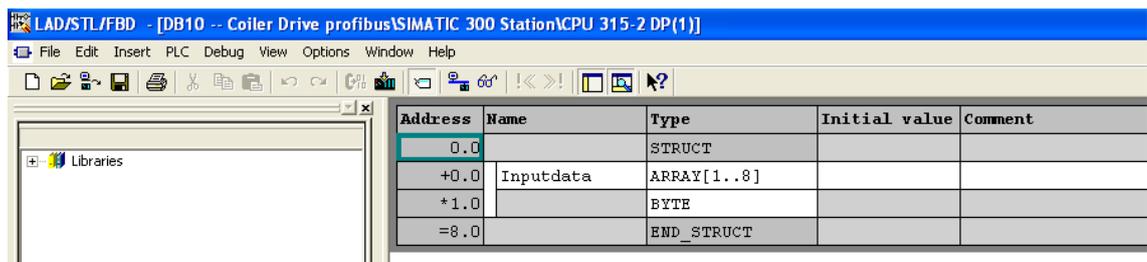
The last parameter *RECORD* specifies where data have to be saved. The best way is to save data in a table. Therefore, we have to create a table with eight bytes; a table is called Data Block in STEP-7.

To create a Data Block, in SIMATIC Manager, in the Blocks tab, a right click in the middle of the screen then “Insert New Object” and “Data Block” has to be chosen, as seen on Figure 66.



**Figure 66: STEP-7 - Insert Data Block**

The name has to be changed in DB10 and the then the LAD/STL/FBD editor (double click on the DB10 icon) has to be opened. Here the data block can be created, for that specify a name like in our case “Inputdata” and the type is “ARRAY[1..8], BYTE. It means that a table with eight squares is created and the type of the data is BYTE as in the Figure 67.



**Figure 67: LAD/STL/FBD - DB10 configuration**

The data block can be saved and the window closed. The last step is to connect this table (DB10) with the reading function. Therefore, in RECORD the name of our data block has to be set:

***RECORD:=DB10.Inpudata***

Finally here the function, Figure 68, with the parameters:

```

PROFIBUS - parameter data

CALL "DPRD_DAT"                SFC14          -- Read Consistent Data of a Standard DP Slave
LADDR :=W#16#100
RET_VAL:=MW200
RECORD :=DB10.Inputdata       P#DB10.DBX0.0
NOP 0
    
```

Figure 68: SFC 14 parameter data function configured

For the second reading function, create a new network and in the same way, create the second reading function for the process data and the data block 31 with only four bytes, as shown on Figure 69 and 70.

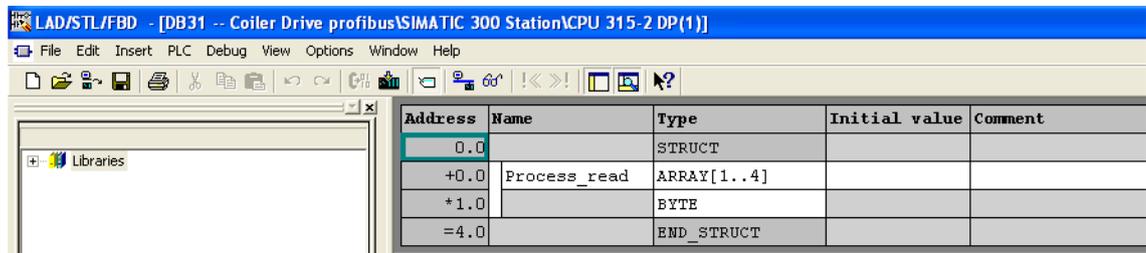


Figure 69: LAD/STL/FBD - DB31 configuration

```

CALL "DPRD_DAT"                SFC14          -- Read Consistent Data of a Standard DP Slave
LADDR :=W#16#108
RET_VAL:=MW204
RECORD :=DB31.Process_read    P#DB31.DBX0.0
NOP 0
    
```

Figure 70: SFC 14 process data function configured

#### 11.4.2 Writing data of a DP slave device with SFC 15 "DPWR\_DAT"

This function is used to write data into the DP device. The step configuration of this function it is the same than the SFC 14. First the function SFC 15 has to be called, Figure 71.

```

CALL "DPWR_DAT"                SFC15          -- Write Consistent Data to a Standard DP Slave
LADDR :=
RECORD :=
RET_VAL:=
    
```

Figure 71: LAD/STL/FBD - CALL SFC 15 function

Three parameters have to be defined: *LADDR*, *RET\_VAL* and *RECORD*. In the Table 6 it is explain the description for each parameter.

**Table 6: SFC 15 function description**

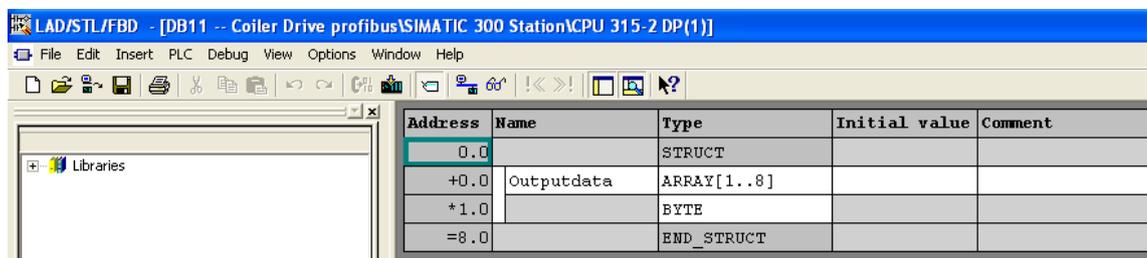
Parameter	Declaration	Data Type	Memory Area	Description
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured start address from the process image output area of the module to which the data will be written. <b>Note:</b> Addresses have to be entered in hexadecimal format. For example, diagnostic address 100 means: LADDR:=W#16#64.
RECORD	INPUT	ANY	I, Q, M, D, L	Source area for the user data to be written. This must be exactly as long as you configured for the selected module with STEP 7. Only the BYTE data type is permitted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.

The start address *LADDR* to write data into the frequency converter, is at the address 256 (100 in hexadecimal format). Therefore this value will be entering in the *LADDR* parameter:

***LADDR :=W#16#100***

In the *RECORD* parameter, the source area for the user data to be written has to be specified. The best way is like for the reading function is to create a new data block.

The procedure is exactly the same than for SFC 14 function, only two things are different, the name of the bloc (DB11) and the name of the table (Outputdata), as seen on the Figure 72.



**Figure 72: LAD/STL/FBD - DB11 configuration**

This new data block will be send trough the PROFIBUS network and data will be written in the frequency converter.

**RECORD:=DB11.Outputdata**

Last step is to set a memory word for the RET\_VAL parameter. In our case we choose the MW202.

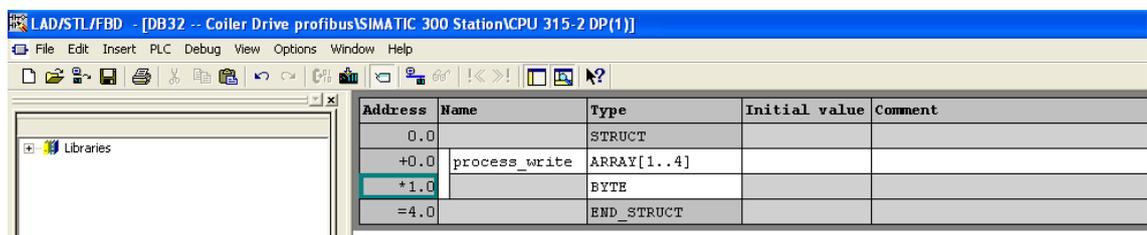
**RET\_VAL:=MW202**

Finally here the function, Figure 73, with the right parameters:

```
CALL "DPWR_DAT"          SFC15          -- Write Consistent Data to a Standard DP Slave
LADDR :=W#16#100
RECORD :=DB11.Outputdata  P#DB11.DBX0.0
RET_VAL:=MW202
NOP 0
```

**Figure 73: SFC15 parameter data function configured**

For the second writing function, create a new network and in the same way, create the second writing function for the process data and the data block 32, as shown on Figure 74 and 75.



**Figure 74: LAD/STL/FBD - DB32 configuration**

```
CALL "DPWR_DAT"          SFC15          -- Write Consistent Data to a Standard DP Slave
LADDR :=W#16#108
RECORD :=DB32.process_write  P#DB32.DBX0.0  -- Temporary placeholder variable
RET_VAL:=MW206
NOP 0
```

**Figure 75: SFC 15 process data function configured**

After all these steps, the reading and writing function is configured. Save the project and upload it into the PLC, as seen in the chapter 10.

## 11.5 Cable and connector configuration

To have a connection between the PLC and the frequency converter, one PROFIBUS cable, as on the Figure 60, has to be used. The cable links the CPU DP port in the PLC and the PROFIBUS card port in the frequency converter.

## 11.6 Frequency converter Vacon NXP configuration

Two types of PROFIBUS field bus boards exist, the OPT-C3 and the OPT-C5. It will be explained the configuration only for the first field bus boards, Figure 76, because it is this one that is present in the frequency converter.

Profibus NXOPTC3 option board

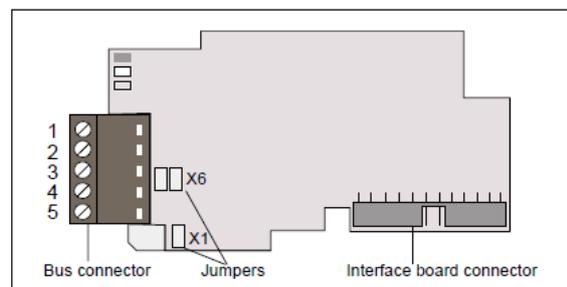


Figure 4-1. Vacon Profibus option board NXOPTC3

Signal	Connector	Description
Shield	1	Cable shield
VP	2	Supply voltage - plus (5V)
RxD/TxD -P	3	Receive/Transmit data - plus (B)
RxD/TxD -N	4	Receive/Transmit data - minus (A)
DGND	5	Data ground (reference potential for VP)

Figure 76: PROFIBUS NXOPT-C3 option board (e-document [16])

Vacon Profibus field bus Board OPT-C3 is connected to the field bus through a 5-pin pluggable bus connector.

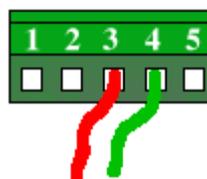


Figure 77: OPT-C3 terminal (e-document [16])

As seen in the Figure 77, the Receive/Transmit data - minus (A) has to be plugged in the slot 4 and the Receive/Transmit data - plus (B) in the slot number 3.

But the grounding has to be prepared first. There are three manners of grounding but the most effective and recommended is grounding by clamping the cable to the converter frame. Strip about 5 cm of the PROFIBUS cable as shown in Figure 78.



Figure 78: PROFIBUS cable stripped

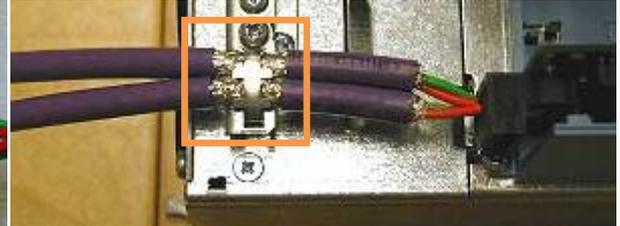


Figure 79: 360 connection

When it is done, insert the red and green data cables into terminal 3 for the red and terminal 4 for the green. Use a 360 connection to ground the cable shield, as shown in the Figure 79 after this step, the cables configuration is done. Now, let's see the configuration of the frequency converter interface.

First, in the "Expander Board" Menu find the NXOPTC3 Parameters and in the Slave address settings (G7.5.1.1.) put the number 3, as shown in the Figure 80.

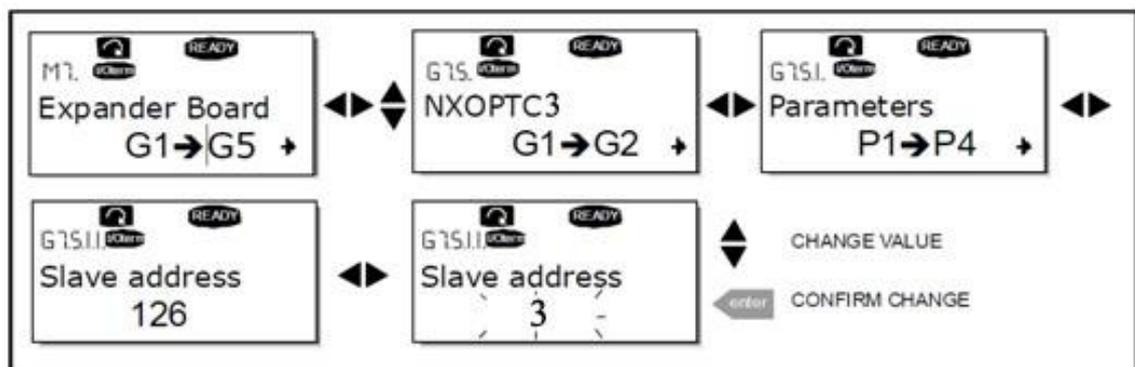


Figure 80: Frequency converter settings (e-document [16])

After this step, the network should be work. To check it, two ways are possible. The first way is to control the three LED on the PROFIBUS field board. The first one shows the PROFIBUS status (red), the second one the PROFIBUS board status (yellow) and the last one the Field bus module (green), more details in the Table 7.

Table 7: PROFIBUS status LED (*e-document [16]*)

<b>Profibus status LED (PS) RED</b>	
LED is:	Meaning:
OFF	Profibus communicates normally. <ul style="list-style-type: none"> <li>Data exchange between Master and Slave</li> </ul>
ON	Profibus communication is broken or not started. <ul style="list-style-type: none"> <li>Bus cable broken or incorrectly connected</li> <li>Wrong configuration or parametrization data of Master</li> <li>Master is off line or shut down</li> </ul>

<b>Profibus board status LED (BS) YELLOW</b>	
LED is:	Meaning:
OFF	Option board not activated
ON	Option board in initialisation state waiting for activation command from the frequency converter
Blinking fast (once/sec)	Option board is activated and in RUN state <ul style="list-style-type: none"> <li>Option board is ready for external communication</li> </ul>
Blinking slow (once/5 secs)	Option board is activated and in FAULT state <ul style="list-style-type: none"> <li>Internal fault of option board</li> </ul>

<b>Fieldbus status LED (FS) GREEN</b>	
LED is:	Meaning:
OFF	Fieldbus module is waiting for parameters from the frequency converter <ul style="list-style-type: none"> <li>No external communication</li> </ul>
ON	Fieldbus module is activated <ul style="list-style-type: none"> <li>Parameters received and module activated</li> <li>Module is waiting for messages from the bus</li> </ul>
Blinking fast (once/sec)	Module is activated and receiving messages from the bus
Blinking slow (once/5 secs)	Module is in FAULT state <ul style="list-style-type: none"> <li>No messages from Master within the watchdog time</li> <li>Bus broken, cable loose or Master off line</li> </ul>

The second way is to use the monitor to see the present status of the PROFIBUS Field bus, as shown on Figure 81.

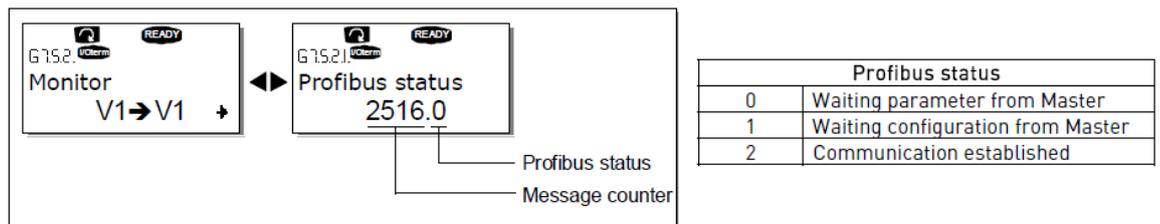
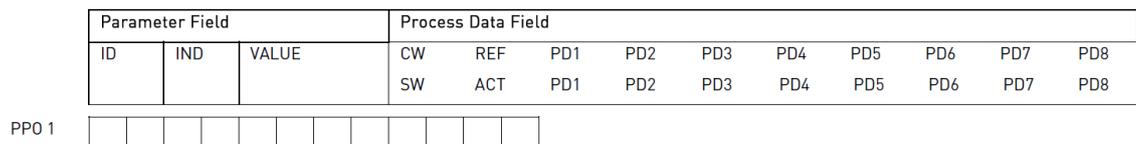


Figure 81: Profibus status

The last step is to change in the Menu to say that the frequency converter is managed by a field bus and not by the control panel.

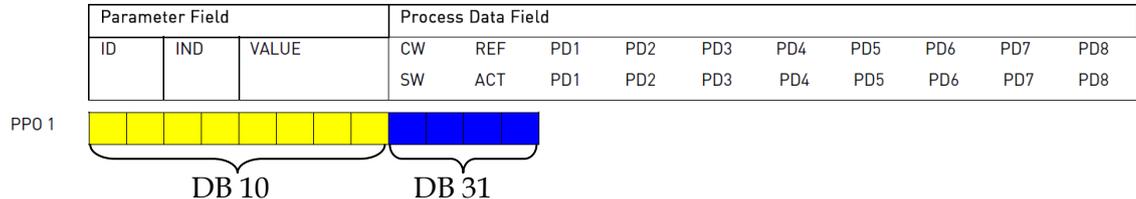
## 11.7 Reading and writing communication

As seen in the chapter 11.4, to read and to write data into the frequency converter, special functions and data blocks have to be used. There are two different functions and four different data blocks created in our case because we use the PPO1 mode, Figure 82. In this mode, eight bytes are used like “Parameter Field” and four bytes are used like “Process Data Field”.



**Figure 82: PPO 1 in Vacon NXP (e-document [16])**

The DB10 data block used for reading contains the eight bytes for the “Parameter Field” and the DB31 data block used for reading contains the four bytes for the “Process Data Field”, Figure 83.



**Figure 83: PPO1 with data blocks configuration (e-document [16])**

For the writing is the same principle. The DB11 data block contains the eight bytes for the “Parameter Field” and the DB32 data block contains the four bytes for the “Process Data Field”.

### 11.7.1 Parameter field

The size of the parameter field is eight bytes and it is divided into three parts: ID, Index (IND) and VALUE, Figure 84.

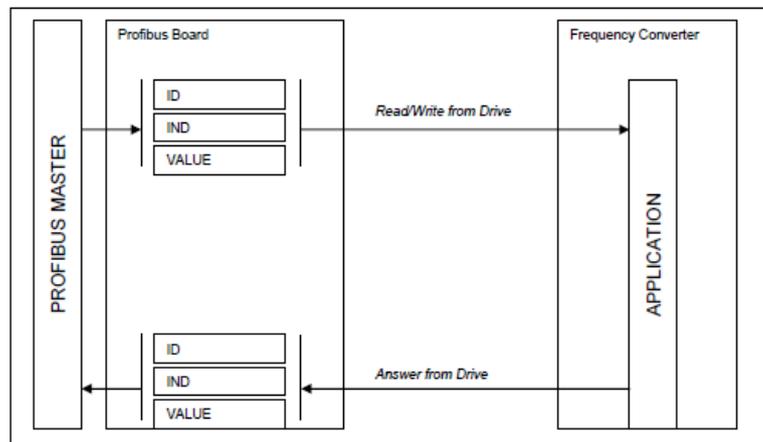


Figure 84: Transfer of parameter field (e-document [16])

All the details can be found in the documentation (e-document [16]), here it will be explain only the most important parameters. For the ID parameter, on the Figure 85, it is shown the two possibilities for reading or for writing.

Read parameter value		Write parameter value	
ID		ID	
10	XX	20	XX

Figure 85: ID parameter (e-document [16])

In the first byte, reading (10) or writing (20) has to be chosen and in the second byte, the ID parameter has to be set in hexadecimal format. The ID number of each parameter can be found in the Vacon documentation or with NC Drive.

The **Index** (IND) parameter is not in use for our configuration.

The **Value** parameter shows the value of the data chosen (speed, temperature, etc).

Two bytes are used for the value, as shown on Figure 86.

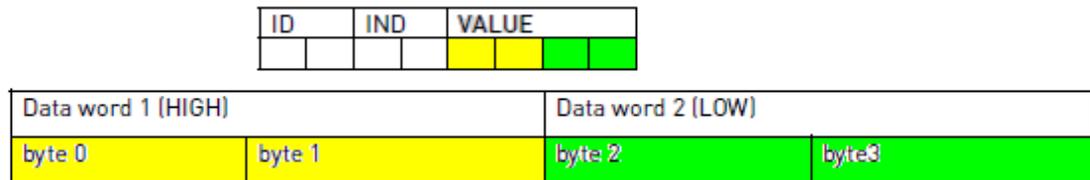


Figure 86: Data value (e-document [16])

In writing mode the data to be written is placed in the field "Data word 2" and in reading mode the answer is in the field "Data word 2". "Data word 1" is normally zero.

### 11.7.2 Process field

The process data field is used to control, in writing, the motor (run, stop, reset, etc) two bytes CW (CW for Control Word). In reading it used to read quick actual values (output frequency, output current, etc), Figure 87.

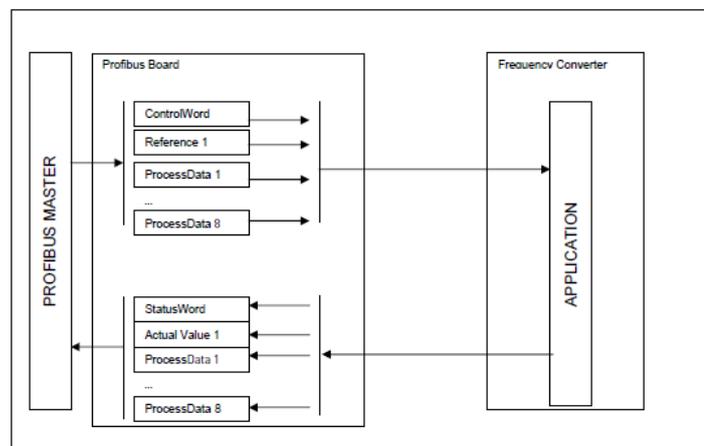


Figure 87: Transfer of process field (e-document [16])

The size of the field is four bytes in PPO1 mode. In our case, the use of process data field is to start and stop via the Control Word written by the Master and to set the rotating speed with Reference (REF). All the details for the CW and the REF parameter can be found in the Vacon manual (e-document [16]).

## 11.8 Example of data exchange through the PROFIBUS network

Take a practical example where, the motor has to be started with 50% of its speed reference and the parameter “maximum frequency” has to be read.

As seen in the previous chapter in the parameter field, the ID has to be set in reading mode 10 and with the ID parameter 66 hex for our example (66 hex= 102 dec = maximum frequency). The Index parameter is not used so it will have 00 00 and the value is also set with 00 00.

For the process field, to start the motor the code 04 7F has to be used. To start the motor with 50% of its speed, the REF parameter has to contain the value 50.00. In hexadecimal format it is 13 88 hex. The Figure 88 is a summary of the frame to send.

ID	1066 hex	1 - Read parameter value 066 - Parameter 102 (= e.g. maximum frequency )
IND	0000 hex	0000 - No meaning
VALUE	0000 0000 hex	0000 0000 - No meaning
CW	047F hex	04 7F- Start command (see chapter control word and state machine)
REF	1388 hex	Speed ref. 50,00% (= 25,00 Hz if parameter min. frequency 0 Hz and max. frequency 50 Hz)

PP01 frame

10	66	00	00	00	00	00	00	04	7F	13	88
----	----	----	----	----	----	----	----	----	----	----	----

Figure 88: PPO 1 frame (*e-document [16]*)

To send these different values into the frequency converter the different data blocks have to be used.

The first step is to put the first eight bytes in the data block 11 and then the last four bytes in the data block 32.

If everything works properly, when the frame is sent on the PROFIBUS network, the motor starts and the maximum frequency can be read into the data block 10, as seen on the Figure 89.

		Writing data block - SFC 15 function				Reading data block - SFC 14			
		Address	Display format	Status value	Modify value	Address	Display format	Status value	Modify value
<b>Parameter field</b>	1	DB11.DBB 0	HEX	B#16#10	B#16#10	1	DB10.DBB 0	HEX	B#16#10
	2	DB11.DBB 1	HEX	B#16#66	B#16#66	2	DB10.DBB 1	HEX	B#16#66
	3	DB11.DBB 2	HEX	B#16#00	B#16#00	3	DB10.DBB 2	HEX	B#16#00
	4	DB11.DBB 3	HEX	B#16#00	B#16#00	4	DB10.DBB 3	HEX	B#16#00
	5	DB11.DBB 4	HEX	B#16#00	B#16#00	5	DB10.DBB 4	HEX	B#16#00
	6	DB11.DBB 5	HEX	B#16#00	B#16#00	6	DB10.DBB 5	HEX	B#16#00
	7	DB11.DBB 6	HEX	B#16#09	B#16#00	7	DB10.DBB 6	HEX	B#16#13
	8	DB11.DBB 7	HEX	B#16#C4	B#16#00	8	DB10.DBB 7	HEX	B#16#88
<b>Process field</b>	1	DB32.DBB 0	HEX	B#16#04	B#16#04	1	DB31.DBB 0	HEX	B#16#00
	2	DB32.DBB 1	HEX	B#16#7F	B#16#7F	2	DB31.DBB 1	HEX	B#16#00
	3	DB32.DBB 2	HEX	B#16#13	B#16#13	3	DB31.DBB 2	HEX	B#16#00
	4	DB32.DBB 3	HEX	B#16#88	B#16#88	4	DB31.DBB 3	HEX	B#16#00

Figure 89: View in STEP-7

We can see that in the data block 10, in the VALUE area, there is 13 88 hex, in decimal we obtain 50 00, that is correct because the maximum frequency is 50,00Hz.

## 12 Final conclusion

To begin let's see in detail the work that has been realized and the problems encountered. After that we will see all the improvement that can be done in the future for the coiler drive process.

### 12.1 The work with STEP-7

In a previous work, a program was already made with STEP-7 for the coiler drive process. It is a small program but enough to start and stop the energy into the coiler drive. First of all the task was to learn how this software works. Manuals from Siemens were very useful particularly the manuals "*SIMATIC Working with STEP-7 V5.1*" (*e-document [2]*) that allows a fast understanding of STEP-7 by showing examples.

There were no big problems to work with STEP-7. The connection between the PC and the PLC worked properly. Some mistakes were done eventually but it was easy to fix these with help. In STEP-7 the help is very important; it should be used because there is a lot of useful information.

### 12.2 The work with WinCC

It was harder to understand this software because there were no examples already made. Luckily there are two very useful manuals "*SIMATIC HMI WinCC flexible Getting Started First Time User*" and "*SIMATIC HMI WinCC flexible Getting Started Experts*" (*e-document [7] and [8]*) that give through one example a lot of information like how to create a project or an interface multilingual. At the end of these two manuals, it is possible to create the first interface. But it is still very difficult to really know the straight way to do some specific task. A lot of practice is needed to really be able to work efficiently.

Some communication problems appeared when some tests were done with the PLC. Indeed, there are two possibilities to test the interface; with the simulator or with the launcher. If the way with simulator is chosen, the PC plays the role of the PLC and of course it is not possible to change parameters with the real PLC. So, to test with the PLC, the way with the launcher has to be taken.

### 12.3 The work on the PROFIBUS network

This task was the most difficult part for this thesis. First of all Siemens gives some basic information about PROFIBUS but it is very difficult to really find very useful information. Siemens Company organizes many training sessions for PROFIBUS, so maybe this is why they do not publish a lot of information.

Luckily, the book "*Decentralization with PROFIBUS-DP*" found in the TAMK library describes all the steps to configure the PROFIBUS network with STEP-7 properly. It was a very useful book, with many examples and a lot of details. The PLC configuration for the PROFIBUS network was done without big problems and properly with the help of this book. The big problem was about the communication configuration. It depends on what kind of devices are used, for the coiler drive, a frequency converter from Vacon is used. On the Vacon website there is one manual about PROFIBUS but it describes the steps for the installation of the field card and the configurations but not for the communication.

Many tries (addresses modified, direct communication, etc) were done to send data through the PROFIBUS into the frequency converter. But without success and there was the error code number 80B1 (see appendix 5) in the reading and writing function. The configuration was right because the LED control on the PROFIBUS field card in the frequency converter and on the PLC show that everything worked. The problem came therefore from the communication.

Some e-mails were sent to Vacon company to identify the exact location of the problem. The field bus team from Vacon checked the coiler drive STEP-7 project. The mistake was to send a frame of twelve bytes in one data block. The right way was to send two frames, the first frame with eight bytes (Parameter Field) and the second frame with four bytes (Process field). Otherwise everything was right in the configuration and in the program. With this information, some runs were tried and finally it was possible to read and write data through the PROFIBUS network. Nevertheless one problem remained with the four bytes of process data field because it was impossible to start the motor for example. It means that the Parameter field works properly but the Process data field does not work. One reason can be the error code (8093) in the RET\_VAL for the two functions in the process data field.

## 12.4 What is next?

There is still much work to do before that the coiler drive process is operational.

First of all, the PROFIBUS network has to be fixed and tested. The process data field problem has to be figured out with the help of Vacon. When the PROFIBUS network is operational other tasks can be done.

The main task will be to integrate the interface with STEP-7; the different symbol addresses in WinCC are already made but not in STEP-7 because the PROFIBUS network is needed to test every symbol address. When the integration will be done, every button and parameter has to be tested under real conditions. Some parameters should be analysed for modification, deletion or addition.

Another task will be to do a safety program with STEP-7. Its task will be to check every parameter set by the user in the control panel. If the user sets the rotating speed value too in the control panel for example; the program has to send an error message and block the start of the process.

If the process works properly, it is possible to imagine a function on STEP-7, that allows the change of the winding direction. A button is already configured in the interface for this purpose.

Finally, it can be interesting, when everything works properly to use a HMI device with a tactile screen (SIMATIC HMI KTP400 for example) instead of a PC and a kind of "User Manual" for the students could be made.

## 13 Bibliography

### Books:

Amin, B. (2001). *Induction motors: analysis and torque control*. Berlin: Springer.

K.Good, J., & R. Roisum, D. (2008). *Winding: Machines, Mechanics and measurements*. Lancaster: DEStech Publications, Inc.

R. Roisum, D. (1998). *The Mechanics of Web Handling*. Atlanta: TAPPI PRESS.

Weigmann, J., & Gerhard, K. (2000). *Decentralization with PROFIBUS-DP*. Erlangen: SIEMENS.

### Final Thesis:

Bravo Zarza, J., & García García, J. L. (2005). *Design of coiler drive process and their regulation by feedback system*. Tampere (FI): Tampere Polytechnic.

Galindo, J. M. (2004). Frequency converters controlled by PC in coiler drive process. Tampere (FI): Tampere Polytechnic.

### Websites:

EURODRIVE, S. (2010). *SEW-EURODRIVE: Gearmotors, Frequency Inverters and Decentralized Drive Engineer*. Retrieved April 23, 2010, from SEW EURODRIVE: <http://www.sew-eurodrive.com/>

Trau, P. (2006, Novembre). *Utilisation de STEP-7*. Retrieved April 15, 2010, from UFR Physique et Ingénierie Campus Meinau: <http://www-ipst.u-strasbg.fr/pat/autom/siemens/step7.htm>

VACON. (2010). *Vacon - a leading supplier of variable speed AC drives*. Retrieved June 01, 2010, from VACON driven by drives: <http://www.vacon.com/>

## E-documents:

### **Manual Siemens**

[1] Overview to *Configuring Hardware and Communication Connections STEP-7 V5.1* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number 6ES7 810-4CA05-8BA0

[2] Overview to *SIMATIC Working with STEP-7 V5.1* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number 6ES7 810-4CA05-8BA0

[3] Overview to *System Software for S7-300/400 System and Standard Functions* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number 6ES7810-4CA05-8BR0

[4] Overview to *SIMATIC Programming with STEP-7 V5.1* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number 6ES7810-4CA05-8BR0

[5] Overview to *SIMATIC WinCC V7.0 Getting started* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number A5E02211557-01

[6] Overview to *SIMATIC HMI WinCC flexible 2007 Compact / Standard / Advanced* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number 6AV6691-1AB01-2AB0

[7] Overview to *SIMATIC HMI WinCC flexible Getting Started First Time User* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number 6ZB5370-1CM03-0BA2

[8] Overview to *SIMATIC HMI WinCC flexible Getting Started Experts* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number 6ZB5370-1CM03-0BA2

### **Manual Vacon**

[9] Overview to *user's manual nxs/p frequency converters* [pdf-file]. [referred to 4.06.2010]

Available: <http://www.vacon.com/Default.aspx?id=450403&FileView=462792>

[10] Overview to Frequency Converter User's manual "Five in One +"- application manual [pdf-file]. [referred to 4.06.2010]

Available: <http://www.vacon.com/Default.aspx?id=450403&FileView=462794>

### **Manual PROFIBUS**

[11] Overview to User's manual Nx frequency converters PROFIBUS DP OPTION BOARD [pdf-file]. [referred to 4.06.2010]

Available: <http://www.vacon.com/Default.aspx?id=450403&FileView=477110>

[12] Overview to *Configuration of S7-300 with CPU315-2 DP as PROFIBUS-DP Master* [pdf-file]. [referred to 4.06.2010] Available: <http://www.hilscher.com/>

[13] Overview to *SIMATIC NET S7-CPs for PROFIBUS Manual* [pdf-file]. [referred to 4.06.2010] Available: documentation package order number C79000-G8976-C154

[14] Overview to *Réseau Profibus* [pdf-file]. [referred to 4.06.2010] Available: <http://www.techniques-ingenieur.fr/book/s8160/reseau-profibus.html>

### Others

[15] Overview to *Alimentation des machines asynchrones* [pdf-file]. [referred to 10.06.2010] Available: <http://www.techniques-ingenieur.fr/book/d3620/>

[16] Overview to *Automates programmable industriels* [pdf-file]. [referred to 10.06.2010] Available: <http://www.techniques-ingenieur.fr/book/s8015/>

## **14 Appendices**

**Appendix 1:** Designation Vacon NXP frequency converter

**Appendix 2:** PROFIBUS GSD file

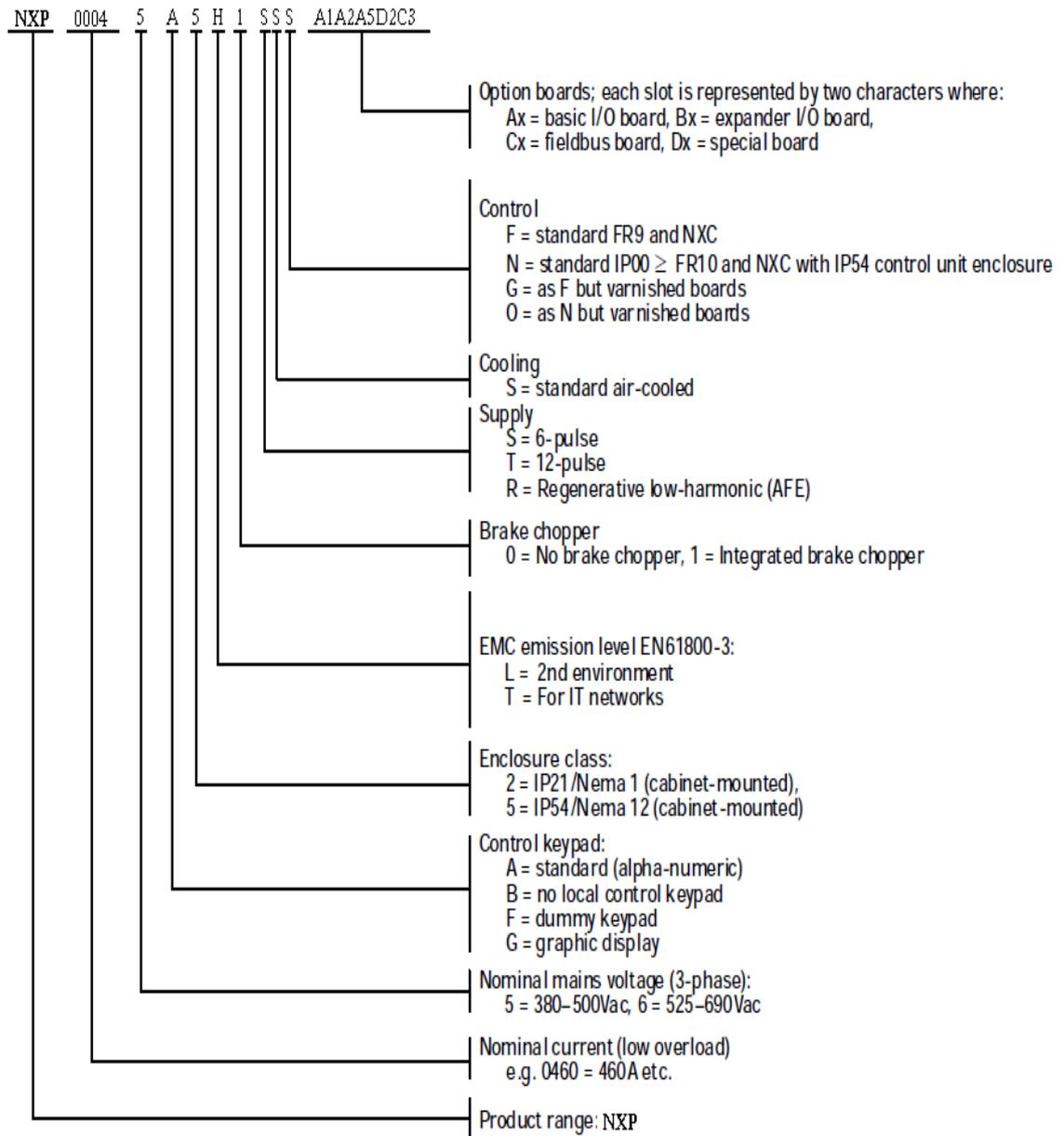
**Appendix 3:** User manual - Coiler drive project creation with STEP-7

**Appendix 4:** User manual – Coiler drive HMI configuration with WinCC

**Appendix 5:** Programming and configuring the PROFIBUS-DP network

**Appendix 6:** Specifications for function SFC 14 and SFC 15

## Appendix 1: Designation Vacon NXP frequency converter



## Appendix 2: PROFIBUS GSD file

```
#Profibus_DP
GSD_Revision = 1
Vendor_Name = "Vaasa Control"
Model_Name = "Vacon NX"
Revision = "1.1"
Ident_Number = 0x9500
Protocol_Ident = 0
Station_Type = 0
FMS_supp = 0
Hardware_Release = "HW1.0"
Software_Release = "SW1.0"
9.6_supp = 1
19.2_supp = 1
93.75_supp = 1
187.5_supp = 1
500_supp = 1
1.5M_supp = 1
3M_supp = 1
6M_supp = 1
12M_supp = 1
MaxTsdr_9.6 = 60
MaxTsdr_19.2 = 60
MaxTsdr_93.75 = 60
MaxTsdr_187.5 = 60
MaxTsdr_500 = 100
MaxTsdr_1.5M = 150
MaxTsdr_3M = 250
MaxTsdr_6M = 450
MaxTsdr_12M = 800
Redundancy = 0
Repeater_Ctrl_Sig = 0
24V_Pins = 0
Implementation_Type = "SPC3"
Freeze_Mode_supp = 1
Sync_Mode_supp = 1
Auto_Baud_supp = 1
Set_Slave_Add_supp = 0
Min_Slave_Intervall = 6
Modular_Station = 1
Max_Module = 5
Max_Input_Len = 28
Max_Output_Len = 28
Max_Data_Len = 56
Modul_Offset = 0
Slave_Family = 1
Fail_Safe = 1
Max_Diag_Data_Len = 6
Module = "VACON PPO 1" 0xF3, 0xF1
EndModule;
Module = "VACON PPO 2" 0xF3, 0xF5
EndModule;
Module = "VACON PPO 3" 0xF1
EndModule;
Module = "VACON PPO 4" 0xF5
EndModule;
Module = "VACON PPO 5" 0xF3, 0xF9
EndModule;
Module =
"_____special_____" 0x00
EndModule
Module = "PPO 2" 0xF3, 0xF1, 0xF1, 0xF1
EndModule
Module = "PPO 4" 0xF1, 0xF1, 0xF1
EndModule
Module = "PPO 5" 0xF3, 0xF1, 0xF1, 0xF1,
0xF1, 0xF1
EndModule
```

## Appendix 3: User manual - Coiler drive project creation with STEP-7

### Creating a project

In this chapter, it will be introduce what are the steps to create and use a project in STEP-7, every step will be explained.

To start the application, double click on SIMATIC Manager shortcut:



When SIMATIC Manager is opened, select “New project”, the New Project window is opened like on the Figure 1.

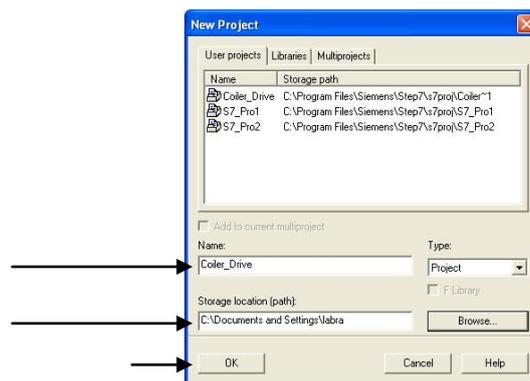


Figure 1: SIMATIC - New Project

Set the “Storage location (path)”, enter a name for the new project, confirm and quit with the OK button.

Now, you are in the main menu of SIMATIC Manager. The object folder MPI has automatically generated which you can see in the right-hand half of the project screen. It is each time automatically generated because MPI is the standard programming and communication interface of the CPU.

Now new objects have to be inserted. In the left-hand side of the project screen, select the project, open the shortcut menu with the right mouse button and select the command “Insert new object”.

Insert SIMATIC 300 station as seen on the Figure 2; the newly inserted objects appears in the right-hand half of the project screen.

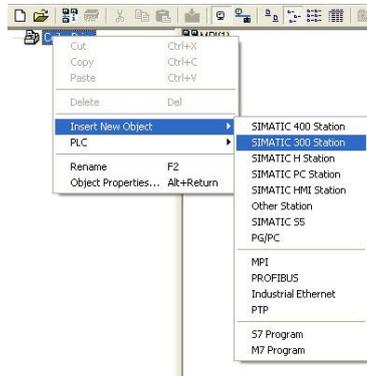


Figure 2: SIMATIC Manager - Insert Station

As seen in the chapter 9.1, two ways are possible, in our case we choose to configure the hardware first.

### Hardware configuration

Next step is the configuration of the hardware used on the S7-300 programmable controller. Call the HW Config  module either by opening the shortcut menu with a right click on the SIMATIC 300 (1) icon and then select "Open Object", Figure 3.

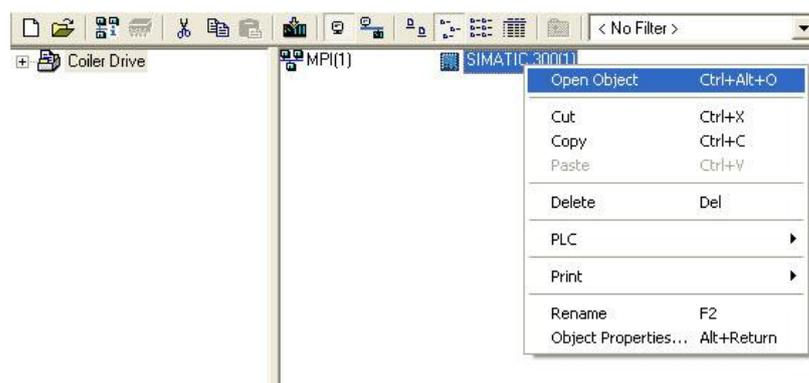


Figure 3: HW Config - Open Object

The HW Config module is automatically started and a screen appears which is divided into two horizontal sections. At this stage it is still empty because the next step is to configure the hardware for the SIMATIC 300 station. The first step is the rack configuration.

For that, in the toolbar, click the catalogue button  to open the hardware catalogue.

In this catalogue open the SIMATIC 300 folder and under “RACK-300” select “Rail”. Drag and drop the selected rack to the upper left section of the screen, Figure 4.

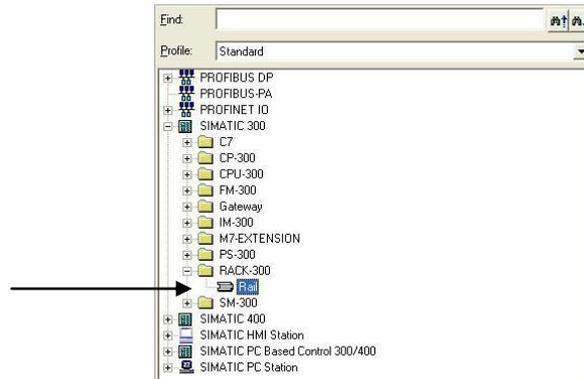


Figure 4: HW Config - Rail

Now, we have to select a power supply (PS), open the folder PS-300, and select the PS 307 5A. Drag and drop the selected PS in the section number 1, Figure 5.

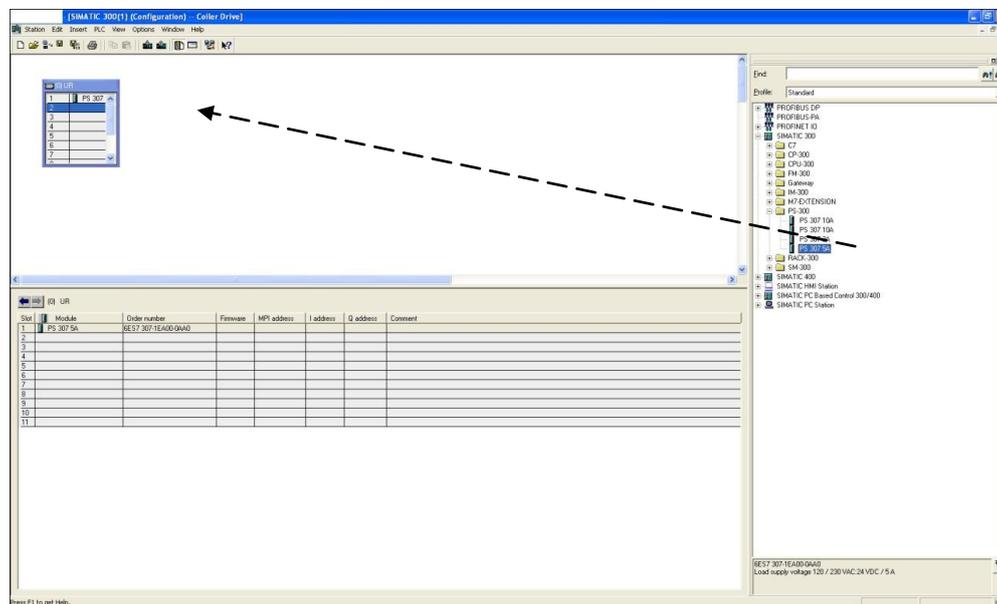
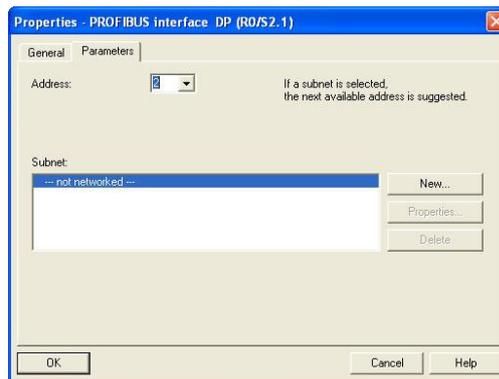


Figure 5: HW Config - PS

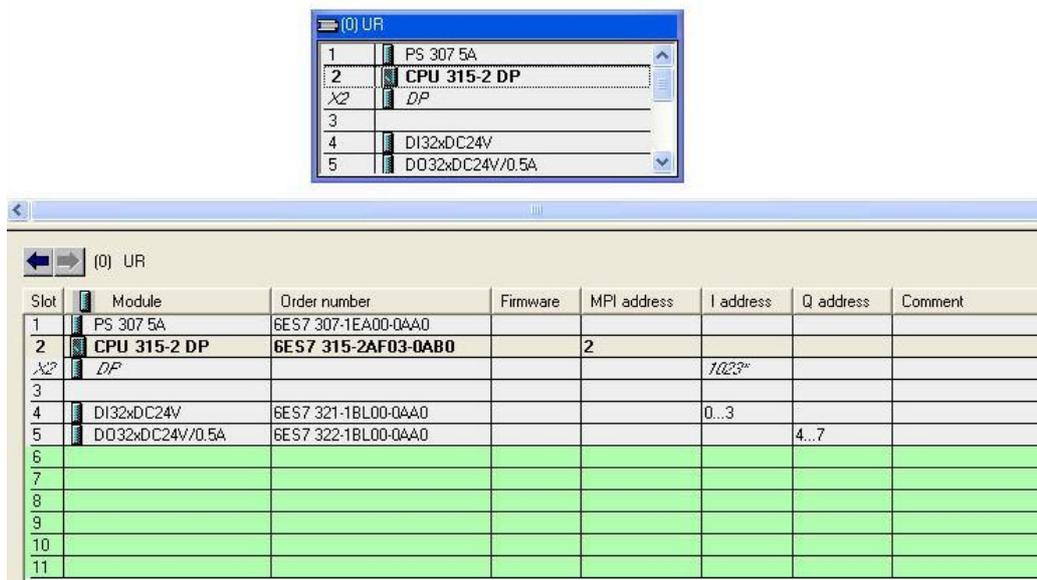
Next, open the CPU-300 folder, open the CPU 315-2DP folder then select the 6ES7 “315-2AF03-0AB0” V1.0 and drag it to the section number 2. The parameter tab of the properties – PROFIBUS Node DP Master dialog box, Figure 6, opens automatically.



**Figure 6: Properties - PROFIBUS interface DP**

Here you can set the parameters of the DP master interface integrated on the CPU. PROFIBUS network configuration will be done in the chapter 14, for the moment click only on the OK button to quit the dialog box. Last step is to select the right I/O modules. Open the SM-300 folder, then open the DI-300 folder and drag and drop the SM 321 DI 32xDC 24V in the section 4. The same for the section 5 open the DO-300 folder and select the SM 322 DO 32xDC 24V/0.5V.

The Figure 7 shows the final configuration:



**Figure 7: HW Config - Final Configuration**

## Programming with STEP-7 software

In the Chapter 9.3.1 about STEP-7 software, we have seen that there are many different languages. First of all, only one language has to be chosen, it is not obligatory but it is really easier for the user because he will not have too many different languages to read. In our case, we have chosen the STL language because the SFC functions can be called only with the STL code. The program in the PLC is really short; indeed there are not many of I/O. Most of the function will be done through the PROFIBUS network (speed settings, torque settings, etc).

In Step-7 the main program is in the OB1 block. This block is subdivided by networks. In our case, we only need two different networks. The first network contains the code to manage the alimantation of the coiler drive through the control panel.

STL instructions used are basic:

English Mnemonics	Description
A	And
O	Or
=	Assign

Here the code, Figure 8, it allows when the "Start" button is pushed and that the "Stop" button is not pushed to start the process alimantation in the coiler drive. And of course to stop the alimantation when the "Stop" button is pushed.

```
A(  
O    "Start"           I2.0      -- start process  
O    "Alim_process"   Q4.1      -- process alimantation  
)  
A    "Stop"           I2.2      -- stop process  
=    "Alim_process"   Q4.1      -- process alimantation
```

**Figure 8: LAD/STL/FBD - Main program**

The Start button, for example, has the address I 2.0. This variable or symbol is configured in the Symbols table, in that table every symbol receive an address, a data type, as seen on the Figure 9. Therefore the first step is to fill this symbol table with all variables and to give for every variable one address cautiously.

	Status	Symbol	Address	Data type	Comment
1		Alim_process	Q 4.1	BOOL	process alimentation
2		Break_1	Q 4.5	BOOL	break motor 1
3		Break_2	Q 4.6	BOOL	break motor 2
4		Cycle Execution	OB 1	OB 1	
5		DPRD_DAT	SFC 14	SFC 14	Read Consistent Data of a Standard DP Slave
6		DPWR_DAT	SFC 15	SFC 15	Write Consistent Data to a Standard DP Slave
7		I/O_FLT1	OB 82	OB 82	I/O Point Fault 1
8		Start	I 2.0	BOOL	start process
9		Stop	I 2.2	BOOL	stop process
10		VAT_1	VAT 1		
11		VAT_read	VAT 2		
12		VAT_write	VAT 3		
13					

**Figure 9: STEP-7 - Symbols table**

When the symbols table is filled, double click on the OB1 block to call the LAD/STL/FBD module. Write the code like in the figure 8, in the first network. In the second network there is the code to call the reading and writing function for the PROFIBUS network. The programming of this network is explained in the chapter 14.

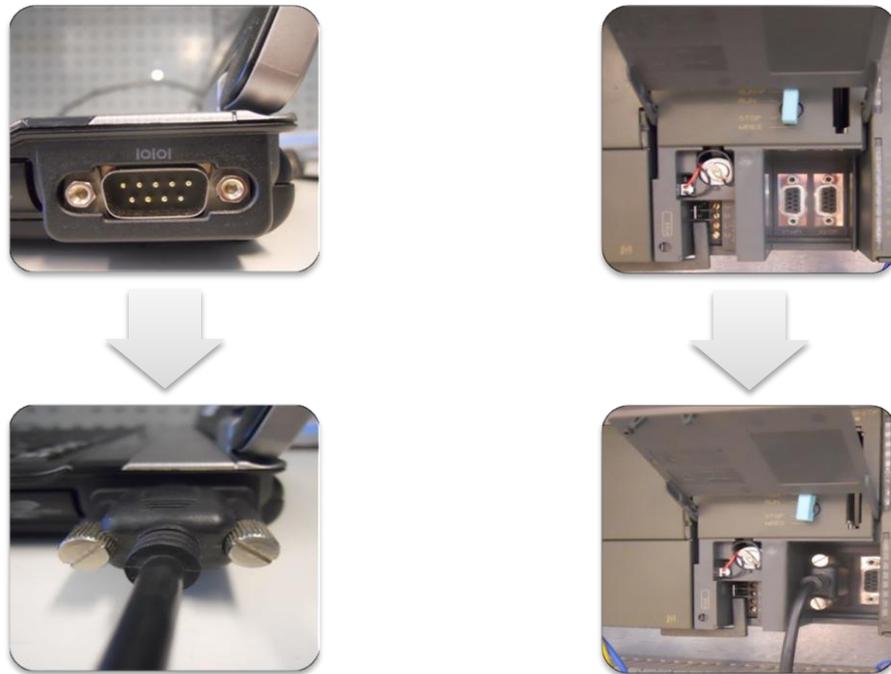
### Downloading the Program to the Programmable Controller

When the hardware configuration and the programming are done all the data have to be sent in the PLC memory, into the CPU. The first step is to save the project. Saving the project allows to check if there are errors. Next, connect the MPI cable, Figure 10, from the PC to the PLC.



**Figure 10: MPI cable**

Plug it as on the Figure 11 and turn on the PLC and then the PC.



**Figure 11: PC and PLC connection**

Click on the online icon:  or in the "View" menu click on "Online". After this step two windows are opened, the offline windows already opened shows the situation on the programming device and the online windows shows the situation on the CPU. Make sure that the connection is done trying to open the Blocks file in the online window. The online or offline status is indicated by the different colour in the headers, as seen on the Figure 12.



**Figure 12: STEP-7 - Online window mode**

Select the Block folder in the offline window and then download the program to the CPU using the download icon:  or the menu command PLC >Download and then check if everything is downloaded in the online window.

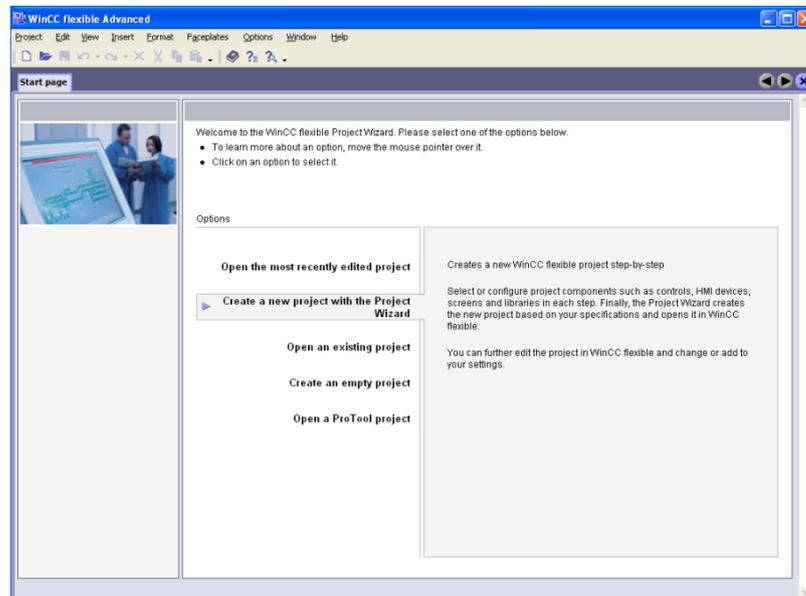
## Appendix 4: User manual – Coiler drive HMI configuration with WinCC

### Creating a project

To start the application, double click on the shortcut:



The WinCC flexible project is opened and it is possible to be guided step by step through the configuration settings, for that, select “Create a new project with the Project Wizard”, as seen on Figure 1:



**Figure 1: WinCC - Create a new project with the Project Wizard**

Next step is to choose the configuration. To operate the coiler drive, one computer with an HMI interface and one PLC is needed.

Therefore “Small Machine” has to be selected then the Next button can be clicked, Figure 2.

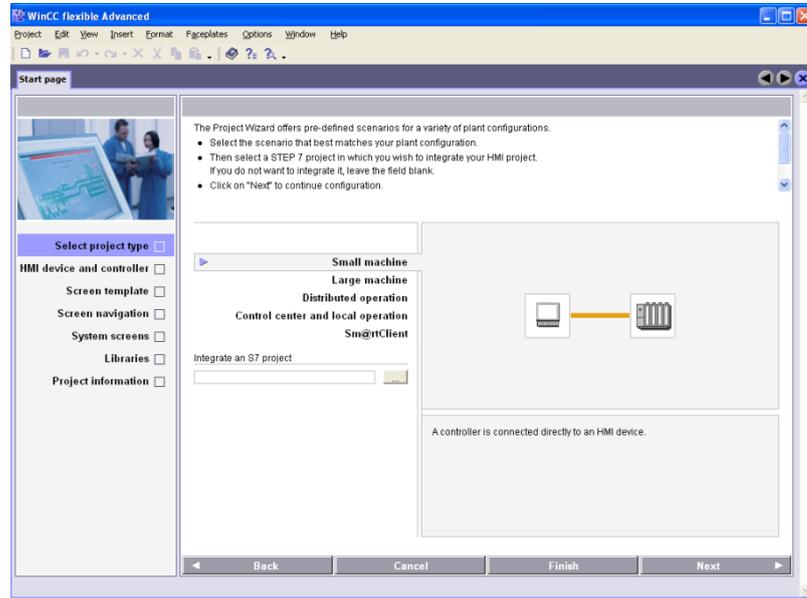


Figure 2: WinCC - Select project type

In the next screen, the HMI device has to be selected. For this, click on the picture of the device and select in the PC folder WinCC flexible Runtime, Figure 3. This option means that the PC will be used like a HMI device.

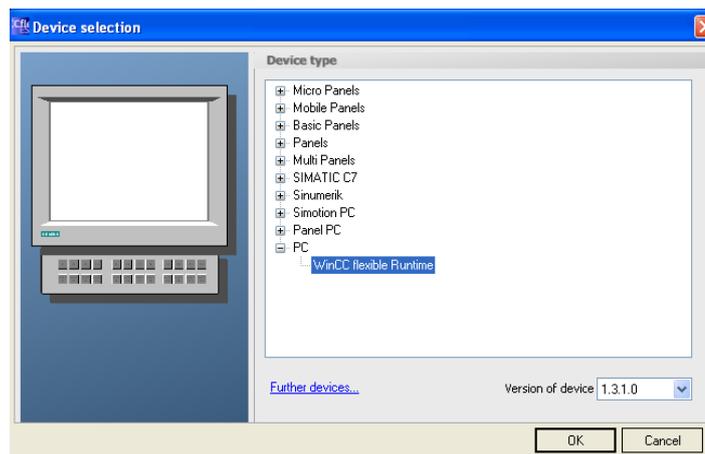


Figure 3: WinCC - Device selection

Then, choose the MPI/ DP connection and select the right PLC, for the coiler drive SIMATIC S7 300/400, Figure 4.

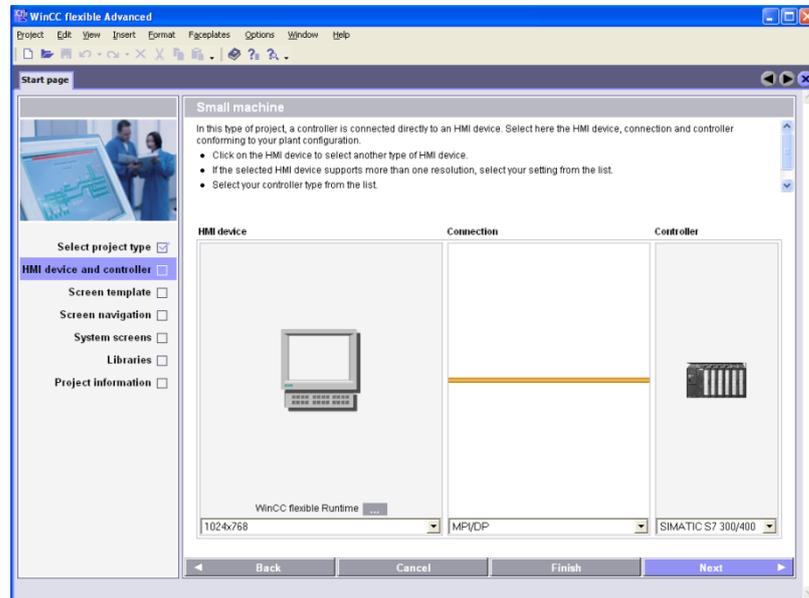


Figure 4: WinCC - HMI device and controller

Next step is to choose what kind of template will be used for the interface. If something really basics is needed, it's good to use this wizard but in our case, the interface will be done by our self so it's better not to use it.

Therefore, unmark all the fields, to have an empty screen, like on Figure 5.

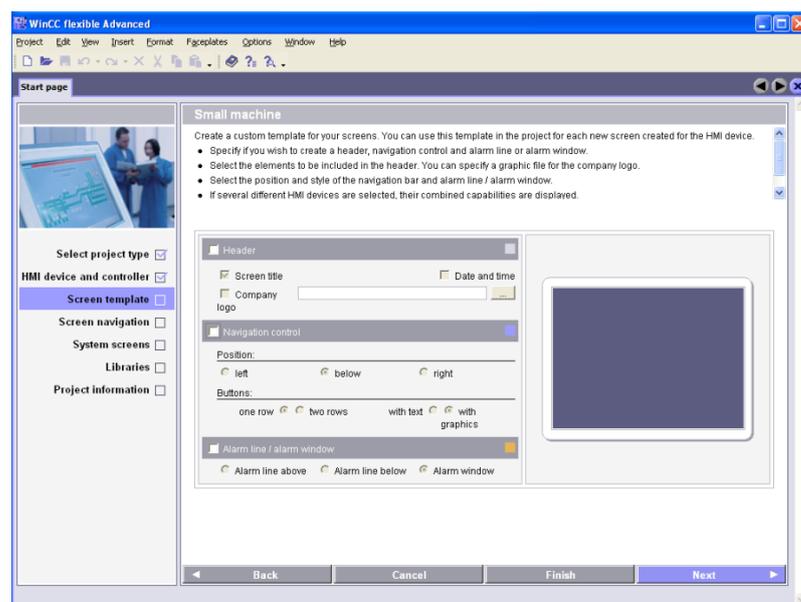


Figure 5: WinCC - Screen template

Click on “Next” to apply the standard settings provided on the page “Screen navigation”, same for the pages “System screens” and “Libraries”.

Then the last step in this wizard is to enter information on the project and to click on the Finish button, Figure 6.

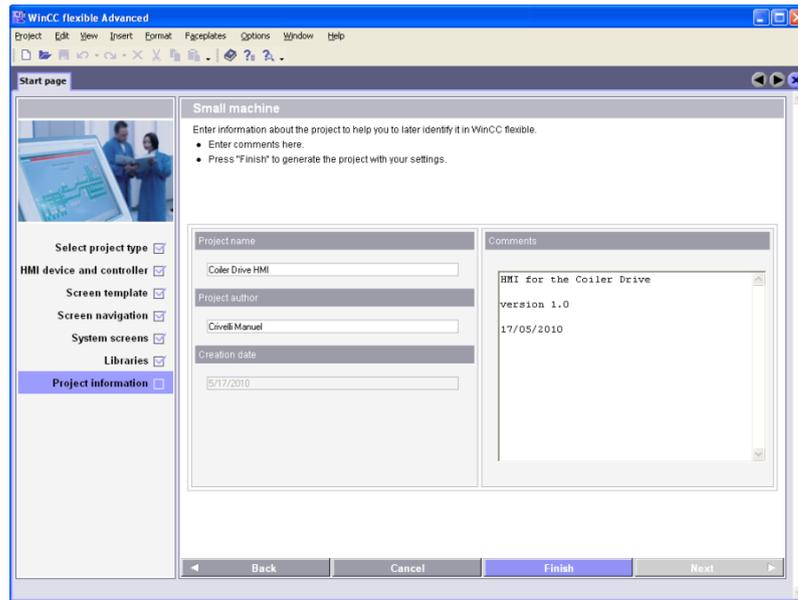


Figure 6: WinCC - Project information

The project wizard has already created some elements, as shown on Figure 7.

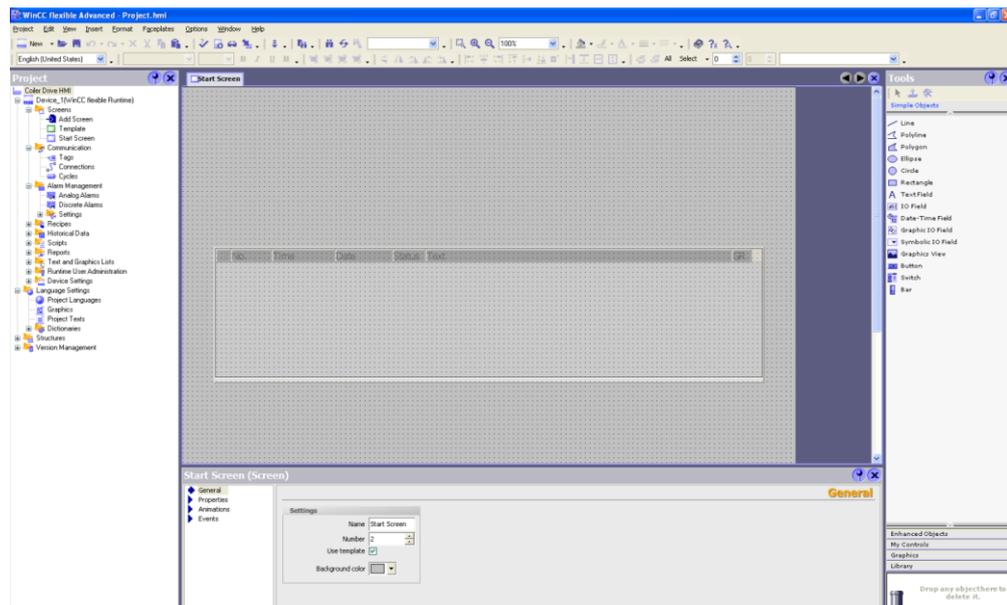


Figure 7: WinCC - Start screen

**Screens:** some pre-configured screen, the start screen, the template and all the future screens are stored in the “Screens area”.

**Communication:** the connection settings between the HMI (PC for our case) and the controller have been already defined, Figure 8. Normally default settings are right and it is not needed to change it.

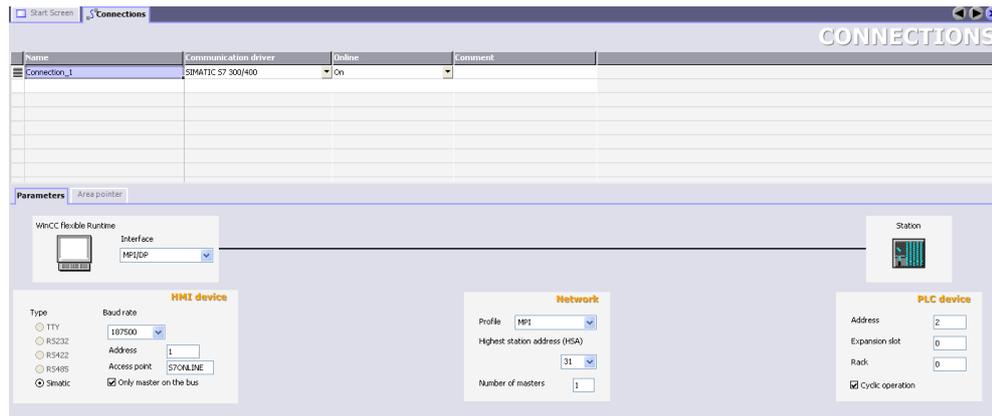


Figure 8: WinCC - Connections

The last step is to go in the “Tag” screen. In this screen, Figure 9, all the symbols used by the HMI have to be defined. It’s better to use the same name that for those that are in the symbol table, in STEP-7. Here examples of “Tags” in the coiler drive.

At the end, it is important that addresses are the same addresses that in the PLC.

Name	Connection	Data type	Address	Comment
Alim_process	Connection_1	Bool	Q 4.1	process alimentation
Break_1	Connection_1	Bool	Q 4.5	break motor 1
Break_2	Connection_1	Bool	Q 4.6	break motor 2
Current_Language	<Internal tag>	Int	<No address>	
motor_1_speed	Connection_1	Int	IW 4	speed motor1
motor_2_speed	Connection_1	Int	IW 6	speed motor 2
programme	Connection_1	Bool	I 14.0	choose between wind or rewind
Speed_sensor	Connection_1	Int	IW 12	speed from the speed sensor
Speed_web	Connection_1	Int	IW 10	speed's setting
Start	Connection_1	Bool	I 2.0	start the process
Stop	Connection_1	Bool	I 2.2	stop the process
Tension_sensor	Connection_1	Int	IW 8	value from the tension sensor
Tension_web	Connection_1	Int	IW 2	

Figure 9: WinCC - Tags

After all these steps, the configuration is done. Now it is possible to start the creation of the interface.

## Interface creation

### Template

First the template has to be defined. Template is the basic screen, in every new screen done, the template will be applied. In the Template there are these kinds of object:

- Navigation menu
- Logo
- Project Title
- Background
- Date and time
- Other

Here the template for the Coiler Drive on Figure 10.



Figure 10: WinCC - Template coiler drive

## Insert images

In this chapter, the way to insert the TAMK logo will be described. It is the same principle to insert pictures in the template or in an another screen. Click on the Graphics tab, Figure 11.

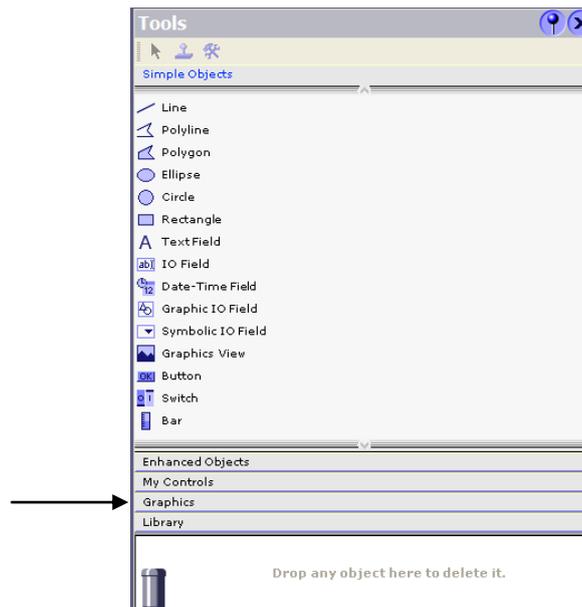


Figure 11: WinCC - Tool > Graphics tab

Right click on the "New link", select "New folder link..." Figure 12.

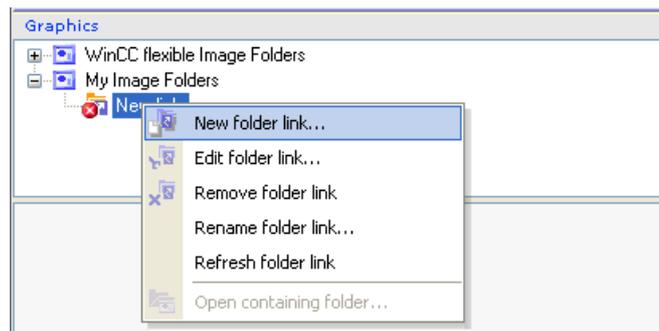


Figure 12: WinCC - Graphics > "New link"

Enter the folder's path in the dialog box, Figure 13.

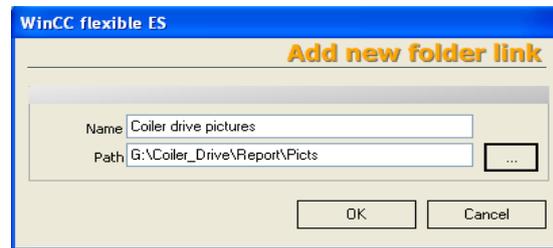


Figure 13: WinCC - Add new folder link

Close the dialog box with the OK button. A new link is created and you can find in it all our pictures.

Drag and drop the TAMK logo in the work area, Figure 14. To set properties, double click on the logo and the Appearance windows will be open. In properties tab you can modify the size and other settings.

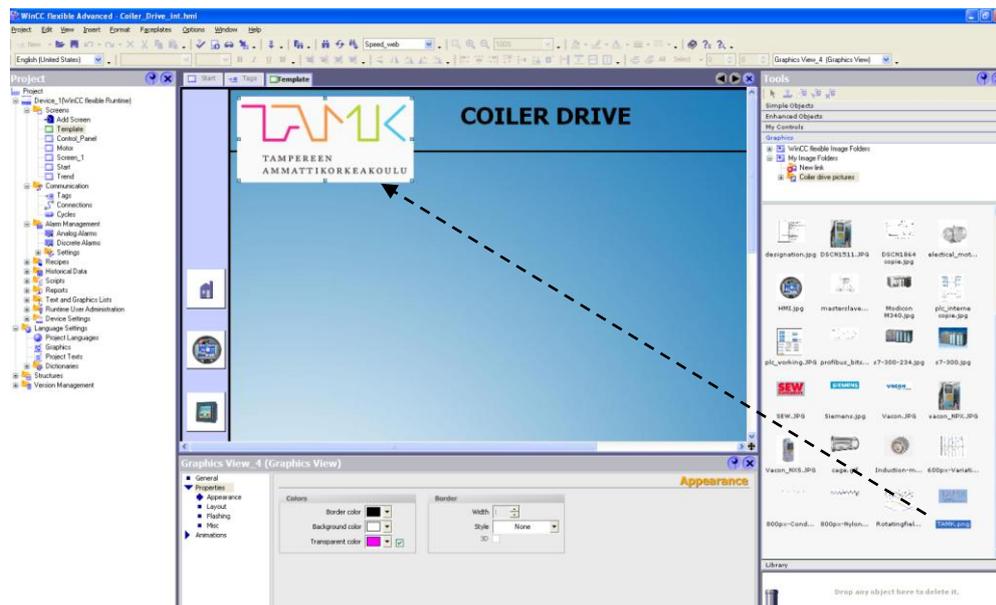


Figure 14: WinCC - drag and drop the picture

## Button

In the template, it is useful to create a navigation menu. It allows to change screen automatically when a button is clicked. To explain what are the steps to create a button, the trend button will be choose as an example. First click on the “Simple Objects” menu, Figure 15. Choose the object Button and drag and drop in the work area:

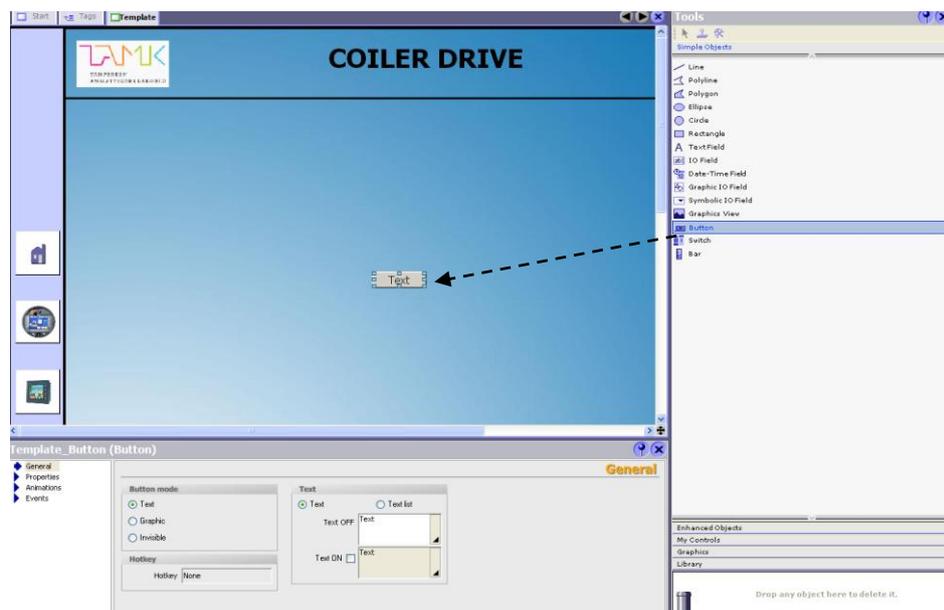


Figure 15: WinCC - Drag and drop button

Now, we have to insert a picture on the button. In WinCC a picture is a graphic element, therefore choose “Graphic”, Figure 16, in the Button mode and in the Graphic OFF field choose the right picture then valid with the set button.

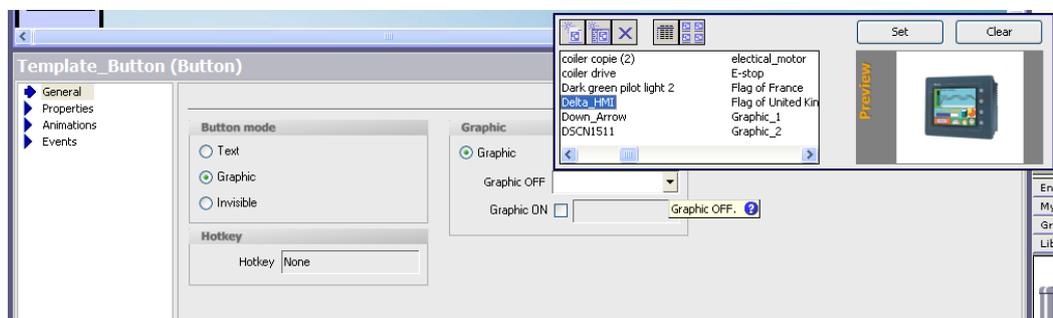


Figure 16: WinCC - General button settings

The last step is to configure the button to show the right screen. In our case the trend screen. Click on the Events tab and then select "Click. Then selection the arrow on the right and choose in the Screens menu the Activate Screen function, Figure 17.

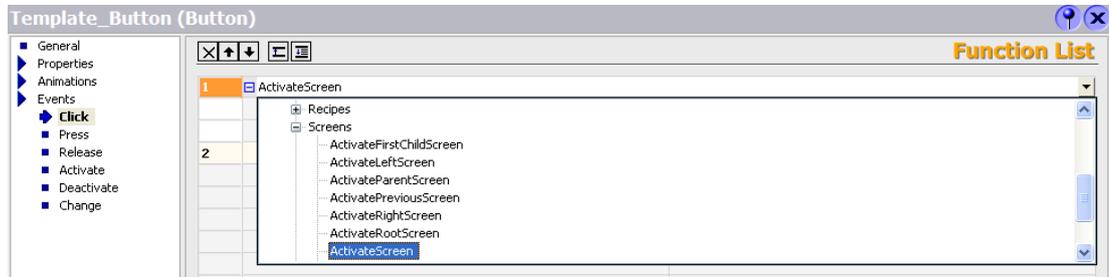


Figure 17: WinCC - Events configuration

Configure the function with the screen name and valid, Figure 18.

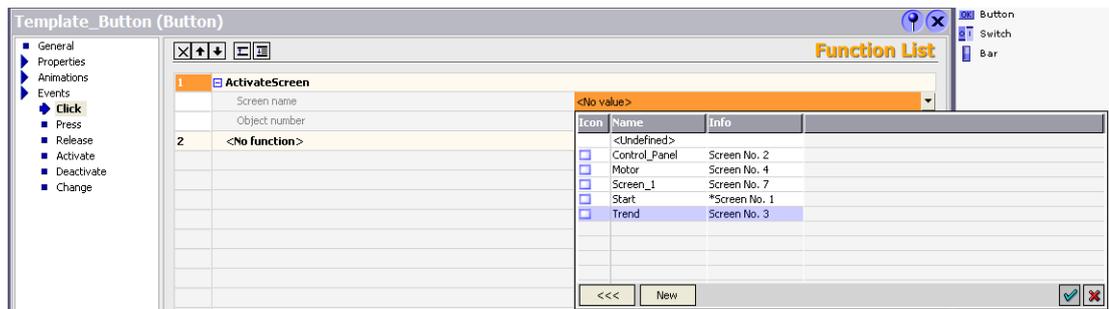


Figure 18: WinCC - Function configuration

After this step, the button is correctly configured and it can be inserted in the navigation menu.

## Appendix 5: Programming and configuring the PROFIBUS-DP network

First the STEP-7 configuration and programming for a PROFIBUS network will be introduced. Secondly the cable installation and connections will be seen and finally the configuration for the frequency converter will be explained.

### PROFIBUS Network STEP-7 Settings

We assume that the STEP-7 project and all the others specifications like hardware configuration are already done.

We start in the main project screen. To add a PROFIBUS network, right click to open the shortcut menu and select "Insert New Object", Figure 1.

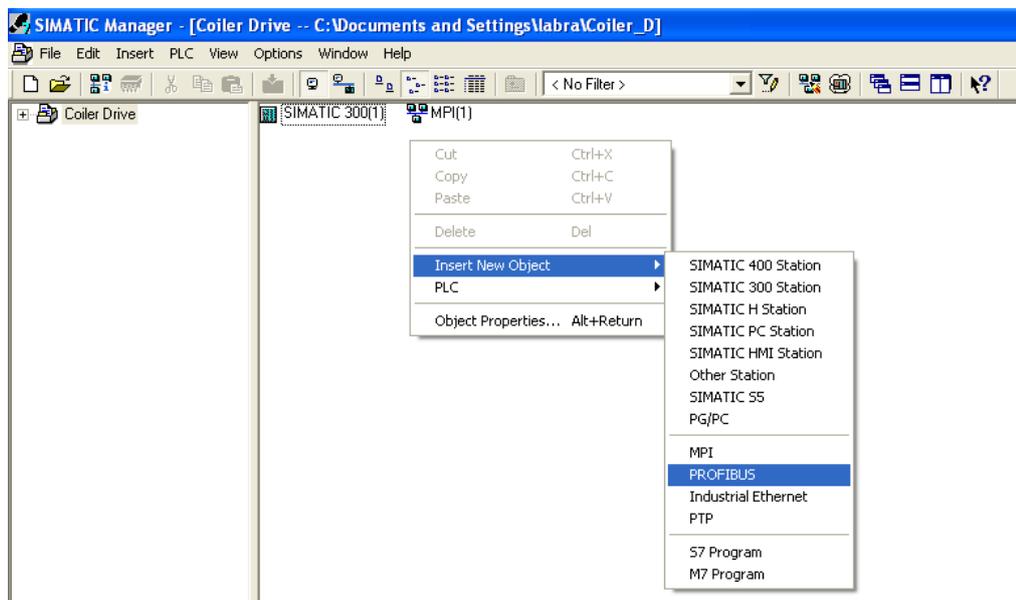


Figure 1: SIMATIC Manager - Insert New Object

Next step select the object PROFIBUS and right click to open the shortcut menu and then select "Open Object", Figure 2, to call the graphic configuration tool NetPro.

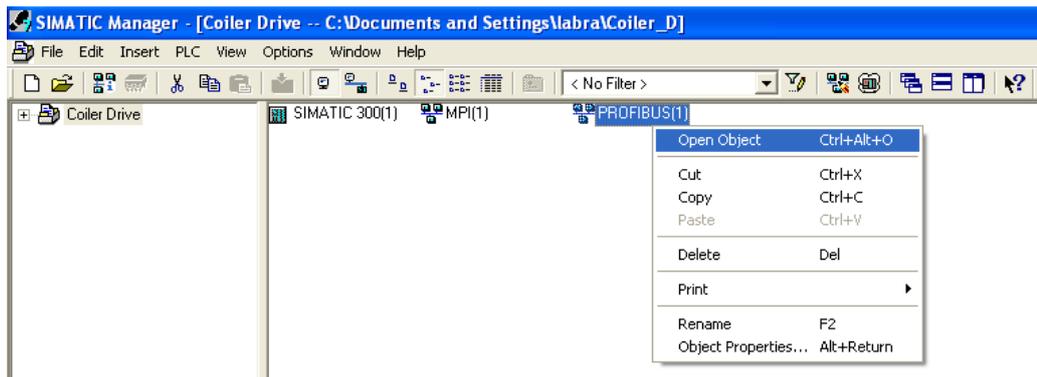


Figure 2: SIMATIC Manager - Open Object

Select the PROFIBUS subnet in violet (PROFIBUS (1)), and right-click to open the shortcut menu. Select the command “Object Properties”, Figure 3.

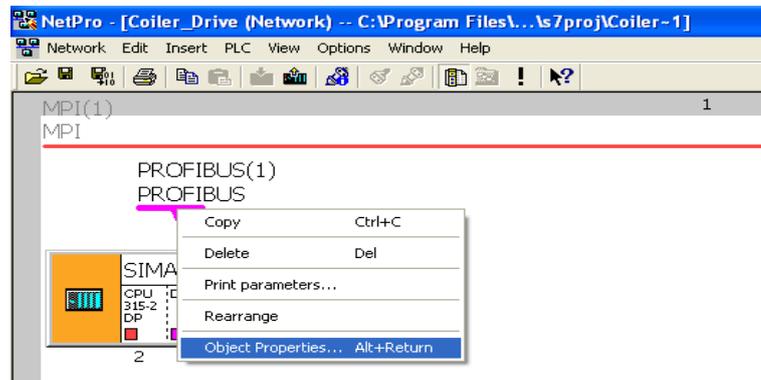


Figure 3: NetPro - Object Properties

In the dialog box “Properties - PROFIBUS “open the “Network Settings”. Here you can set all relevant network parameters for the PROFIBUS subnet, Figure 4.

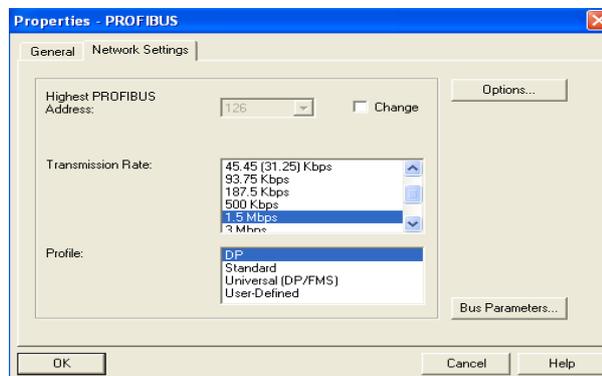


Figure 4: NetPro - Properties PROFIBUS

**Highest PROFIBUS address:** This parameter is used to optimize bus access control (token management) for multi-master bus configuration. With a mono-master PROFIBUS-DP configuration, don't change the default setting of 126 for this parameter.

**Transmission Rate:** The transmission speed you select here, will apply to the entire PROFIBUS subnet. This means that all stations (also called nodes) which are used must support the selected baud rate.

**Profile:** Each bus profile contains a set of PROFIBUS bus parameters. In our case, we are in a mono-master configuration, so we have to choose DP profile.

**Option:** button opens a box where you can run PROFIBUS-DP in the Constant Bus Cycle mode.

**Bus parameter:** button provides access to the bus parameters calculated by STEP-7.

Confirm the suggest settings (default settings) for the example project with OK. The next step is to configure the DP Slaves in the *HW Config*. For this, select Simatic 300 station and right-click to open the shortcut menu. Select the command "Open Object", Figure 5.

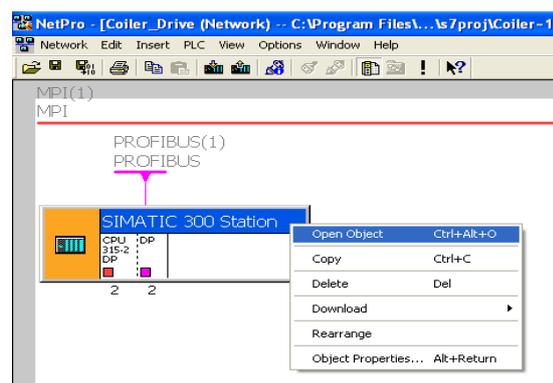


Figure 5: NetPro - Open Object

Vacon frequency converters are not in the Siemens data base. If we want to be able to use it like a slave, we need to install a GSD file in the Siemens data base. GSD file can be found on the website manufacturer, more details in chapter 11.3.1. Open your internet browser and go to [www.vacon.com](http://www.vacon.com), click on “Support & Downloads” and after that on Software.

Finally click on Field bus conf files and download the Vacon NX Profibus GSD (vac29500.gsd)<sup>2</sup>. The GSD file can be found in the appendix 1.

When the file is downloaded, cut it and paste it in these two folders:

- C:\Program Files\...\Siemens\Step7\S7DATA\GSD
- C:\Program Files\Siemens\...\Step7\S7DATA\NSMET

The right way is to paste only in the first folder but if the version 5.4 is used, it doesn't work. To be sure, it's better to paste the gsd file in the both folder. To install it in STEP-7, in HW Config, click on the Option menu and “Install GSD File...” Figure 6.

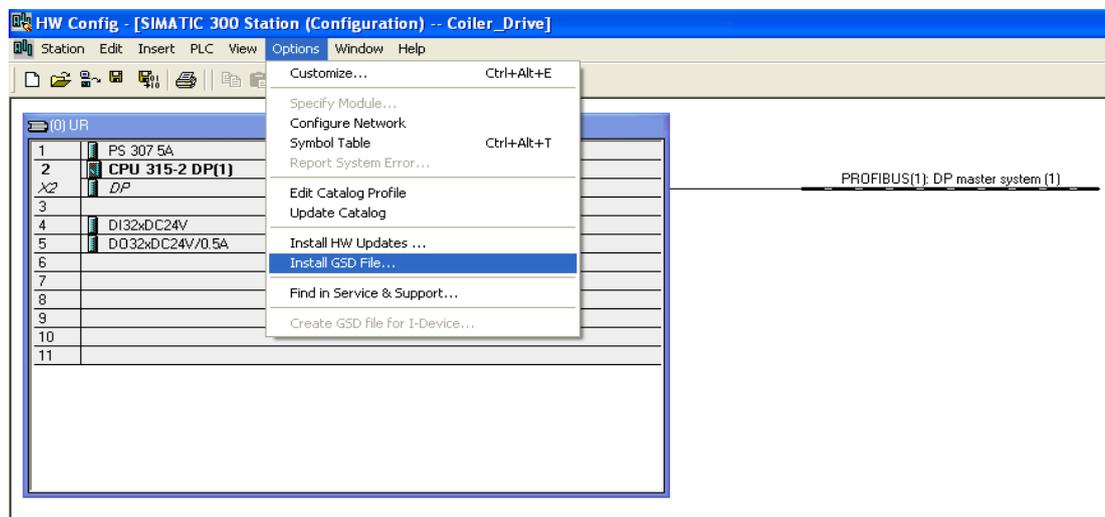


Figure 6: HW Config - Install GSD File

<sup>2</sup> The direct link is: <http://www.vacon.com/Default.aspx?id=463715> (visited 05.05.2010).

Click on the “Browse...” button and check if the GSD file can be open from the GSD folder or from the NSMET folder. In this example, it is open from the NSMET folder. Select the vaco29500.gsd file and click on Install button, Figure 7.

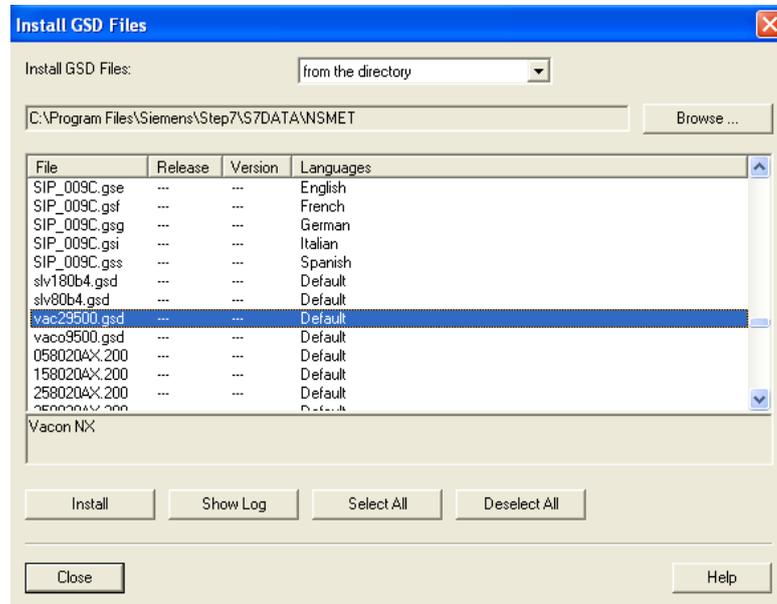


Figure 7: HW Config - vac29500 GSD file installation

Now, the GSD file is installed in the Siemens database and the frequency converter NXP can be selected. For this, click on the upper right-hand side of the screen and select the “PROFIBUS-DP” folder > “Additional Field Devices” > “Drives” and drag and drop the “Vacon NX device”, Figure 8.

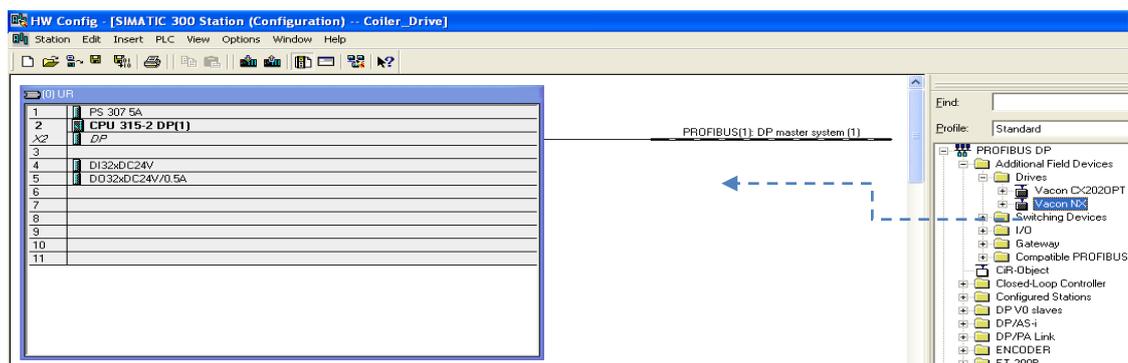
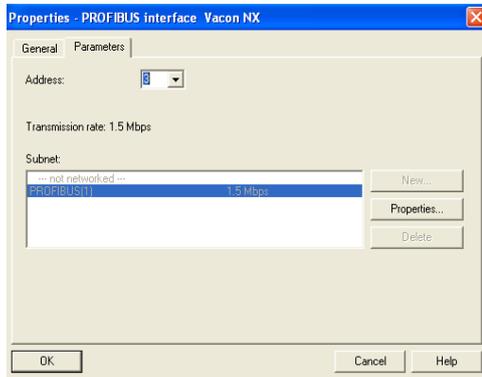
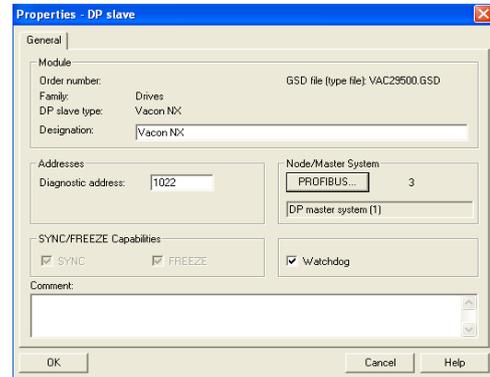


Figure 8: HW Config - Drag and drop Vacon NX device

The Properties – PROFIBUS Node interface Vacon NX dialog box automatically opens, Figure 9. Here set the PROFIBUS address for this DP slave to “3” and click Ok to return the HW Config station screen.



**Figure 9: Properties - PROFIBUS interface Vacon NX**

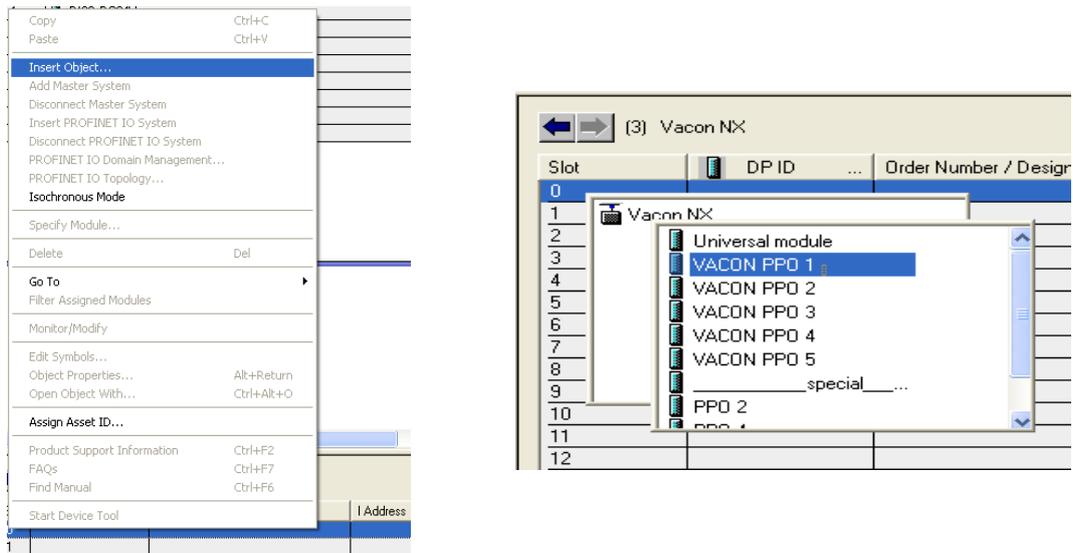


**Figure 10: Properties - DP slave**

Now, double-click on the Vacon NX box. This opens the DP Slave Properties dialog box, Figure 10. It is shown characteristic like order number, device family, type and description.

The diagnostic address is used by the CPU to indicate a DP Slave failure with organization block OB86.

Finally to know which address is used for the first byte, select the Vacon NX box and on the bottom right-click on the rack 0, select “Insert Object” and then click on NX PPO1, Figure 11.



Slot	DP ID	Order Number / Designation	I Address	Q Address
0	4AX	VACON PPO 1	256...263	256...263
1	2AX	--> VACON PPO 1	264...267	264...267
2				

Figure 11: HW Config - Insert Object DP-slave

HW Config can be closed, save the changes and control if in NetPro the PROFIBUS network is done as shown in Figure 12.

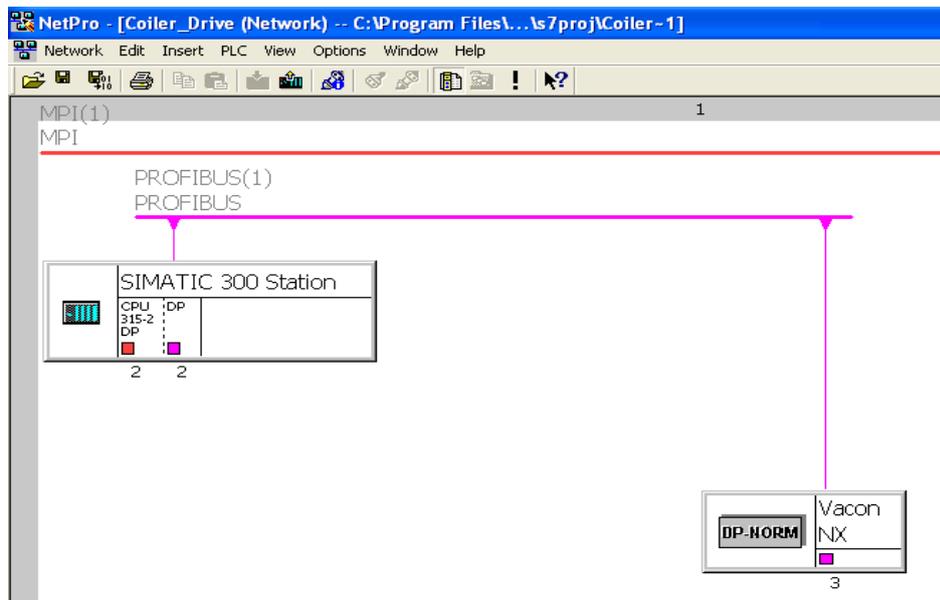


Figure 12: NetPro - PROFIBUS network check

## Appendix 6: Specifications for function SFC 14 and SFC15

SFCs for Distributed I/Os

### 15.5 Reading Consistent Data of a DP Standard Slave with SFC 14 "DPRD\_DAT"

#### Data Consistency

Refer to the section Overview of S7 Communication and S7 Basic Communication – Data consistency.

#### Purpose of SFC 14

You require SFC 14 "DPRD\_DAT" because you can only read out a maximum of four continuous bytes using load instructions that access the I/Os or the process image input table.

#### Description

With SFC 14 "DPRD\_DAT" (read consistent data of a DP standard slave), you read the consistent data of a DP standard slave, with the maximum length being fixed for each specific CPU. You will find the maximum length in the technical data of your CPU. If no error occurred during the data transfer, the data that have been read are entered in the destination area identified by RECORD.

The destination area must have the same length as configured for the selected module with STEP 7.

If you read from a DP standard slave with a modular design or with several DP identifiers, you can only access the data of one module/DP identifier per SFC 14 call specifying the configured start address.

Parameter	Declaration	Data Type	Memory Area	Description
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured start address from the I area of the module from which the data will be read. <b>Note:</b> Addresses have to be entered in hexadecimal format. For example, diagnostic address 100 means: LADDR:=W#16#64.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Destination area for the user data that were read. This must be exactly as long as you configured for the selected module with STEP 7. Only the data type BYTE is permitted.

*SFCs for Distributed I/Os*

**Error Information**

<b>Error Code (W#16#...)</b>	<b>Explanation</b>
0000	No error occurred.
8090	<ul style="list-style-type: none"> <li>• You have not configured a module for the specified logical base address or</li> <li>• you have ignored the restriction concerning the length of consistent data or</li> <li>• you have not entered the start address in the LADDR parameter in hexadecimal format.</li> </ul>
8092	A type other than BYTE is specified in the ANY reference.
8093	No DP module from which you can read consistent data exists at the logical address specified in LADDR.
80A0	Access error detected while the I/O devices were being accessed.
80B0	Slave failure on external DP interface module.
80B1	The length of the specified destination area is not identical to the user data length configured with STEP 7.
80B2	System error with external DP interface module.
80B3	System error with external DP interface module.
80C0	The data haven't yet been read by the module.
80C2	System error with external DP interface module.
80Fx	System error with external DP interface module.
87xy	System error with external DP interface module.
808x	System error with external DP interface module.

## 15.6 Writing Consistent Data to a DP Standard Slave with SFC 15 "DPWR\_DAT"

### Data Consistency

Refer to the section: Overview of S7 Communication and S7 Basic Communication – Data consistency.

### Purpose of SFC 15

You require SFC 15 "DPWR\_DAT" because you can only write a maximum of four continuous bytes using the transfer instructions that access the I/Os or the process image input table.

### Description

With SFC 15 "DPWR\_DAT" (write consistent data to a DP standard slave), you transfer the data in RECORD consistently to the addressed DP standard slave. The maximum length of the data to be transferred is fixed for each specific CPU. You will find this information in the technical data for your CPU. The data is transferred synchronously, in other words, on completion of the SFC, the write job is also completed.

The source area must have the same length as you configured for the selected module with STEP 7.

If the DP standard slave has a modular design, you can only access one module of the DP slave.

Parameter	Declaration	Data Type	Memory Area	Description
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured start address from the process image output area of the module to which the data will be written. <b>Note:</b> Addresses have to be entered in hexadecimal format. For example, diagnostic address 100 means: LADDR:=W#16#64.
RECORD	INPUT	ANY	I, Q, M, D, L	Source area for the user data to be written. This must be exactly as long as you configured for the selected module with STEP 7. Only the BYTE data type is permitted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.

*SFCs for Distributed I/Os*

---

**Error Information**

Error Code (W#16#...)	Explanation
0000	No error occurred.
808x	System error with external DP interface module.
8090	<ul style="list-style-type: none"><li>You have not configured a module for the specified logical base address or</li><li>you have ignored the restriction concerning the length of consistent data or</li><li>you have not entered the start address in the LADDR parameter in hexadecimal format.</li></ul>
8092	A type other than BYTE is specified in the ANY reference.
8093	No DP module to which you can write consistent data exists at the logical address specified in LADDR.
80A1	Access error detected while I/O devices were being accessed.
80B0	Slave failure on external DP interface module.
80B1	The length of the specified source area is not identical to the user data length configured with STEP 7.
80B2	System error with external DP interface module.
80B3	System error with external DP interface module.
80C1	The data of the previous write job on the module have not yet been processed by the module.
80C2	System error with external DP interface module.
80Fx	System error with external DP interface module.
85xy	System error with external DP interface module.