Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2014•2015
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: energie

Masterproef

Upgrade of the motor test unit

Promotor :
ing. Eric CLAESEN

Promotor :
ing. TIMO VAISANEN

Kevin Geutjens
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie

2014•2015
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: energie

Masterproef

Upgrade of the motor test unit

Promotor :
ing. Eric CLAESEN

Promotor :
ing. TIMO VAISANEN

Kevin Geutjens

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie

Upgrade of the motor test unit

Hardware/software upgrade and supplying communication.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Master's thesis

Automation Engineering

Spring 2015

Kevin Geutjens

Clarification of signature

ABSTRACT

HAMK Valkeakoski
Automation Engineering
Exchange studie

Author Kevin Geutjens **Year** 2015

Subject of Master's thesis Upgrade of motor test unit

ABSTRACT

The purpose of this thesis was to complete an already started upgrade process of the local didactical motor test unit. The new PC program has already been written by a previous thesis, but lacks communication and is not completely finished. Also new hardware components have to be chosen and implemented. Finally all these components have to be implemented and commissioned.

In order to achieve the communication to the PC, the old I/O panels will be replaced by a new Beckhoff PLC, fitted with an Ethernet connection, PROFINET connection and the necessary I/O terminals. In order to send the necessary data to the PC program, the connection between the PC and the PLC will be established with an Ethernet cable by using the TwinCAT ADS Ethernet protocol. The old ABB frequency converter will also be replaced by a new ABB frequency converter, equipped with a PROFINET interface. The communication between this frequency converter and PLC will happen with a PROFINET connection.

The old hardware and software, as well as the new PC program made by the previous thesis are well documented so these are very useful sources. Also Beckhoff has a big online information system that I will use very often.

I have improved the previously made PC program and added the communication with the PLC to it. Also the PLC program itself is programmed in a way it makes the setup much safer in use. The motor test unit is now again fully operational and can be used by the students.

Keywords Motor, Beckhoff, ADS, PROFINET, C#.

Pages **59** p. + appendices 16 p.

ACKNOWLEDGEMENT

First of all I would like to thank a few people from HAMK university: Osmo Leiniainen for helping me find components and showing me around in the automation lab the first day I arrived, as well for showing me how the motor test unit initially worked. Timo Väisänen as my supervisor, for all the help I received. Juha Sarkulla for giving initial help to get my TwinCAT project started. Also a big thanks for Jan-Peter Novak, for doing a great effort of ordering all the necessary Beckhoff components I needed for my thesis. Further, many thanks go to all the other teachers and student in the Automation department who made this thesis possible. A special thanks also goes to Annina Herala for making this Erasmus+ project possible, and providing constant help in times of need.

Next I would also like to thank my friends in Belgium Jori Winderickx and Steffen Triekels, as well as my roommate Jan-Lennert Geuns for helping me figure out problems in the programming. I would like to thank all my family and friends (both in Belgium and Finland) for supporting me.

Finally, a big thanks also goes to Petja Jantunen from the Beckhoff technical support for providing quick and good support.

TABLE OF FIGURES

Figure 1 A Beckhoff PLC with the controller module and I/O modules [1]. 2
Figure 2 The PLC scan cycle 3
Figure 3 Ladder diagram with A, B and C as inputs and X as an output. 4
Figure 4 SFC program 5
Figure 5 Left: basic construction of a three-phase induction motor. [2] ; Right: more efficient flat windings. [3] 6
Figure 6 Basic representation of a squirrel cage rotor[4] 6
Figure 7 The stator magnetic field (top) caused by the 3-phase AC voltage (bottom) 7
Figure 8 Current-speed characteristic[5] and Torque-speed characteristic[5] 8
Figure 9 Motor connections. Left: Star (Y) ; Right: Delta (Δ) 9
Figure 10 Characteristics of the start-delta start. 10
Figure 11 Frequency converter with diode rectifier and IGBT inverter [6] 11
Figure 12 Voltage waveforms in different stages of the drive [6] 11
Figure 13 IGBT inverter 12
Figure 14 Steady-state equivalent circuit of the induction motor [7] 12
Figure 15 Progress of the M-n characteristics curve by using V/f control. 13
Figure 16 Stator voltage versus frequency profile using scalar control [7] 13
Figure 17 Magnetic particle brake. [8] 14
Figure 18 Simplified structure of an incremental encoder.[9] 14
Figure 19 Determination of the direction in quadrature output signals [10] 15
Figure 20 RS422 differential line output for the A-channel. [10] 15
Figure 21 Wheatstone bridge of the strain gauges inside the load cell. [11] 16
Figure 22 PROFINET communication channels 17
Figure 23 Complete setup 19
Figure 24 Motor unit with motor, mass, brake and encoder 20
Figure 25 Motor nameplate 20
Figure 26 Left: Brake with load cell. Right: Occurring forces 21
Figure 27 Encoder 21
Figure 28 Control unit 22
Figure 29 IO-panels 22
Figure 30 ABB ACS600 Frequency converter 23
Figure 31 The old windows 95 PC 24
Figure 32 Original software “SIMO” 25
Figure 33 Top: old hardware layout. Bottom: new hardware layout. 29
Figure 34 Complete PLC configuration 30
Figure 35 Beckhoff CX9020 controller[15] 31
Figure 36 EL5101 channels and electrical layout. 32
Figure 37 BK1250 terminal. 32
Figure 38 KL1408 channels and electrical layout 33
Figure 39 KL2408 channels and electrical layout 33
Figure 40 KL3064 terminal and electrical layout 34
Figure 41 Terminal KL3112 and KL3102 35
Figure 42 KL4002 output terminal 35
Figure 43 ABB ACS355 36
Figure 44 Complete device configuration. 38
Figure 45 GVLs used in the PLC program 39
Figure 46 PPO type messages available for ACS355 drives 40
Figure 47 The separate bits in the status word and the control word. 41
Figure 48 PLC program layout 42

Figure 49 I/O handling in the PLC program 43
Figure 50 Link between the flag variable and the linked I_O variable..................... 44
Figure 51 Connection monitoring 45
Figure 52 SFC used for the direct starting mode 46
Figure 53 SFC for the star-delta starting mode 47
Figure 54 Function block to control the inverter. 48
Figure 55 ADS structure 49
Figure 56 ADS message 50
Figure 57 Class structure in the C# program 51
Figure 58 Layout of the main program 54
Figure 59 Inverter start tab page 55
Figure 60 Manual control screen 56
Figure 61 Start-up chart 56
Figure 62 Acceleration chart 57
Figure 63 Characteristic chart 57
Figure 64 Status bar 57
Figure 65 Completed motor test unit 58

CONTENTS

1 ASSIGNMENT 1

2 THEORETICAL BACKGROUND 2

2.1 PLC........ 2
2.1.1 Basics........ 2
2.1.2 Scan cycle........ 2
2.1.3 Programming........ 4
2.2 Asynchronous electrical motor........ 6
2.2.1 Construction 6
2.2.2 Basic working principles 7
2.2.3 Characteristics 8
2.2.4 Connection types 9
2.2.5 Direct online start (DOL) 9
2.2.6 Star-delta start........ 10
2.2.7 Soft starter start........ 10
2.3 Frequency converter........ 10
2.3.1 Structure 10
2.3.2 Scalar control........ 12
2.4 Magnetic particle brake........ 14
2.5 Incremental encoder........ 14
2.5.1 Working principal........ 14
2.5.2 RS422 TTL output........ 15
2.6 Load cell and amplifier 16
2.7 PROFINET........ 16
2.7.1 Network 16
2.7.2 PROFIdrive........ 17

3 INITIAL SITUATION 19

3.1 Setup........ 19
3.2 Hardware 20
3.2.1 Motor unit........ 20
3.2.2 Control unit........ 22
3.2.3 PC 24
3.3 Software 25
3.4 Functions........ 26
3.5 Components requiring an upgrade 26
3.5.1 IO-Panels 26
3.5.2 Frequency converter 27
3.5.3 Communication 27

4 NEW HARDWARE........ 29

4.1 Layout........ 29
4.2 PLC........ 29
4.2.1 Replacing the old I/O panels 29
4.2.2 Configuration overview........ 30
4.2.3 Controller........ 31
4.2.4 Encoder terminal EL5101........ 32
4.2.5 Bus coupler terminal BK1250........ 32

4.2.6 Digital input terminal KL1408 33
4.2.7 Digital output terminal KL2408 33
4.2.8 Analog input terminal 0..10V KL3064.... 34
4.2.9 Analog input module 0..20 mA KL3112.... 34
4.2.10 Analog input module -10V…+10V KL3102 35
4.2.11 Analog output module 0..10V KL4002.... 35
4.3 Frequency converter.... 36
4.3.1 Usage 36
4.3.2 Replacement 36
4.3.3 New drive 36
4.3.4 PROFINET communication 37
4.3.5 Modified parameters.... 37
4.4 PC.... 37

5 PLC PROGRAM.... 37

5.1 TwinCAT 3 37
5.1.1 eXtended Automation Engineering 37
5.1.2 eXtended Automation Runtime.... 37
5.2 Device configuration.... 38
5.2.1 Beckhoff PLC terminals.... 38
5.2.2 PROFINET devices.... 39
5.3 Global Variables Lists GVL.... 39
5.4 PROFINET communication in TwinCAT 3 40
5.4.1 PPO messages.... 40
5.4.2 Handling the status word and the control word..... 41
5.5 Program layout.... 42
5.5.1 Main layout.... 42
5.5.2 Input and output handling.... 43
5.6 Connection- and E-stop monitoring.... 44
5.6.1 E-stop monitoring.... 44
5.6.2 Connection monitoring.... 45
5.7 Datalogger.... 45
5.8 Program selection.... 46
5.8.1 Direct start 46
5.8.2 Star-delta start.... 47
5.8.3 Soft starter start.... 47
5.8.4 Frequency converter start 47
5.8.5 Motor characteristic.... 48
5.8.6 Drive control (manual mode) 48

6 PC PROGRAM 49

6.1 Existing.... 49
6.2 PLC communication.... 49
6.2.1 TwinCAT ADS.... 49
6.2.2 Class structure 51
6.2.3 Connecting to the PLC 51
6.2.4 Variable handles 51
6.2.5 PLC-connection monitoring 52
6.2.6 Retrieving measured data 52
6.2.7 Updating gauges 53
6.3 Layout.... 54

6.3.1 Starting modes ... 55
6.3.2 Manual control ... 55
6.3.3 Charts ... 56
6.3.4 Start-up chart ... 56
6.3.5 Acceleration chart ... 57
6.3.6 Characteristics chart ... 57
6.3.7 Status bar ... 57

7 CONCLUSION ... 58

SOURCES ... 59

Appendix 1 Hardware list
Appendix 2 PLC IO table
Appendix 3 Electrical schematics
Appendix 4 Modified parameters ACS355

1 ASSIGNMENT

The goal of this thesis is to finish an upgrade process of the motor test unit located in HAMK Valkeakoski. This machine is a didactical setup so students can learn the characteristics of an engine depending on which way the engine is started.

In 2012, two students already had this assignment for their thesis. One student was responsible for upgrading the software on the PC, the other student was responsible for communication. The software on the PC was finished, but because the communication part was not made at all, the test unit was far from finished. Also no new hardware has been installed so the system was still equipped with the original hardware.

The test unit must be able to function with a new windows 7 PC at the end of this thesis, and should keep functioning for at least 20 years. This means some hardware components have to be upgraded. Secondly it also has to be safer to use, which means the way an emergency stop is handled has to be changed completely. At last it has to be student proof.

2 THEORETICAL BACKGROUND

2.1 PLC

2.1.1 Basics

A Programmable Logic Controller (PLC) is a programmable controlling device used for automation of (industrial) processes, such as factory assembly lines, chemical processes or even amusement rides. The advantage of using a PLC instead of a normal computer, is that a PLC can handle many in- and outputs, and controls the system in real-time (chapter 2.1.2). Also both the hardware and software of a PLC are optimized to operate in various conditions. For instance, a PLC is usually immune to electrical noise, resistant to vibration and impact, and can work in extended temperature ranges. Furthermore, a PLC has a modular structure, with a “controller” module that stores the program, and a wide range of in- and output modules that are chosen depending on the process (Figure 1). The controller module is fitted with non-volatile memory to stores the program for the PLC. This way the program will still be kept even after performing a power cycle.

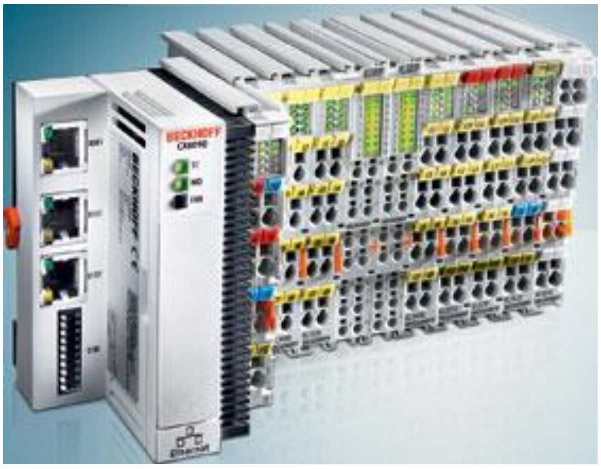


Figure 1 A Beckhoff PLC with the controller module and I/O modules [1].

2.1.2 Scan cycle

The main feature of a PLC is that it has real time control. This means when an input value changes, the desired output responds within a desired period so a process can be controlled. This reaction time can vary, for example a temperature control requires a reaction time in the unit of seconds, while a high speed process requires response within milliseconds.

In order to achieve this, a PLC uses the “scan cycle” (Figure 2). This cycle continuously runs and takes only a few milliseconds depending on the program. It consists of 3 basic steps: reading the inputs (A), executing the program (B) and changing the outputs (C).

The first step is to read all the input values (also called periphery inputs) of the modules that are connected to the PLC (A). These values are then stored in the input memory of the controller, so the program can access these values. In order to make this process faster, the controller only updates the changed input values. Input values that are the same as the previous cycle are not written in the input memory again, since they are still in it. After this the main program is executed once (B). By executing it once, it prevents the program of being stuck in an infinite loop. The program reads the necessary values from the input memory, and will change the desired output values if needed. These output values are written in the output memory of the controller. Finally all the changed output values in the output memory are transferred to the physical outputs (C) (periphery output). After this the controller repeats this cycle.

The time it take to execute this complete cycle is called the “cycle time”, and is a very important factor in automation processes. Because this cycle is executed many times per second, usually only 1 or 2 input values will change in one cycle. This results in a very low load on the program. In order to keep a process real-time, a maximum cycle-time is determined. If the cycle time exceeds this (e.g. program too long), damage can be caused to the process since the controller cannot respond fast enough anymore to changes. A watchdog can be implemented in the controller, checking this time.

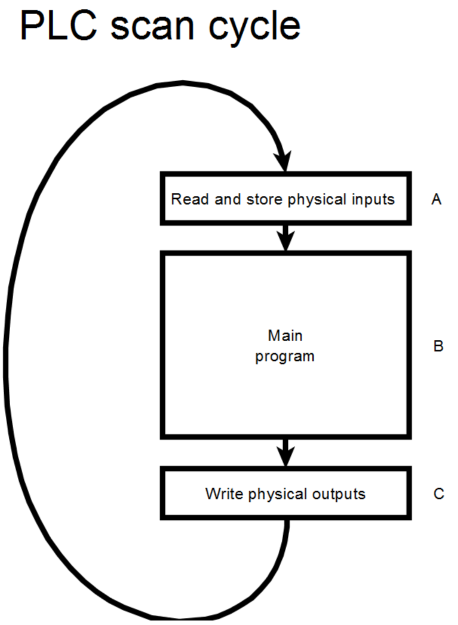


Figure 2 The PLC scan cycle

2.1.3 Programming

The PLC is programmed so it will perform the desired actions for the process. It is done by writing the program on a PC, and sending this program to the PLC. This program uses standard programming languages under the IEC 61131-3 standard:

- Ladder diagram (LD)
- Structured text (ST)
- Function block diagram (FBD)
- Instruction list (IL)
- Sequential function charts (SFC)

In this thesis LD, ST and SFC will be used.

The ladder diagram (Figure 3) is a method that can be compared to the old relay racks. Each relay is represented by a symbol on the ladder diagram with connections between devices. Figure 3 shows an example program. In the upper network output X will be activated when both A and B are active. The lower network shows an OR configuration of B and C, and an AND configuration with A. Now X will only be activated when A will be active in combination with B or C also being active.

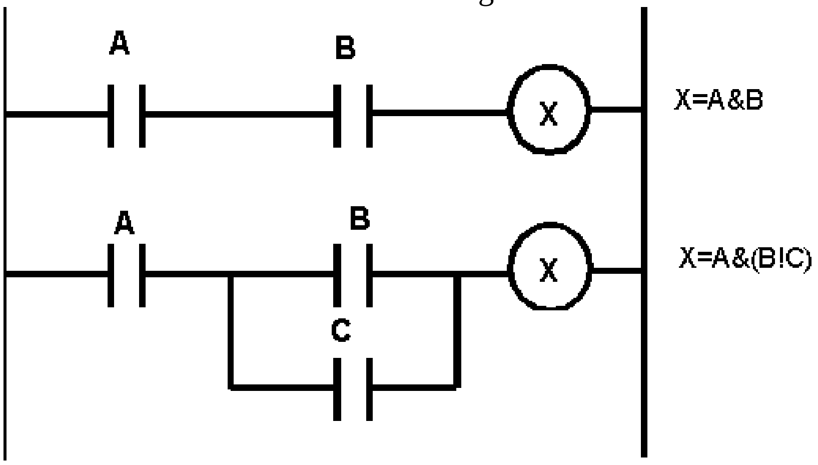


Figure 3 Ladder diagram with A, B and C as inputs and X as an output.

Structured text is a textual programming language with the same syntax as Pascal. An example of code is shown below, performing the same action as Figure 3.

```
VAR
           A : BOOL;
           B : BOOL;
           C : BOOL;
           X : BOOL;
END_VAR
X := A AND B;
X := A AND (B OR C);
```

Sequential function chart (Figure 4) is a graphical programming language mostly used for sequential behaviour. It consists of 3 major components: Steps, conditions and actions. A step contains several actions to be executed. In order to go to the next step in the SFC, specified conditions must be met. When the program advances to the next step, the current step is deactivated and the next step is activated.

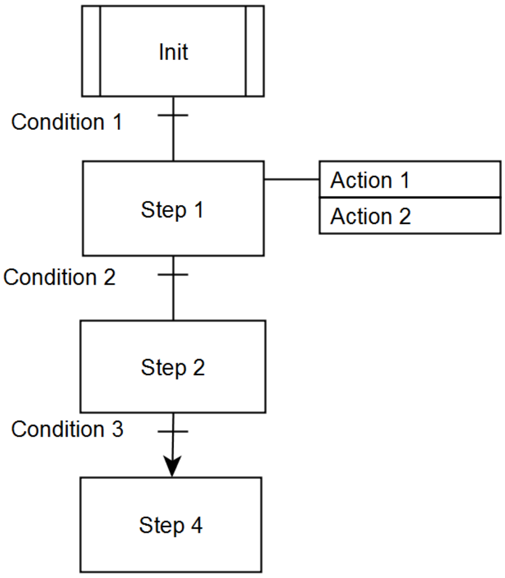


Figure 4 SFC program

2.2 Asynchronous electrical motor

2.2.1 Construction

An asynchronous induction motor is an electrical motor that uses electromagnetic induction to create torque on the output shaft. It consists of two major parts: the stator (stationary) and the rotor (rotating). The stator is connected to the three-phase AC power supply, and has no moving mechanical parts. Compared to a DC-motor, this type of motor requires very little maintenance since it requires no mechanical commutation.

The stator is the non-moving part of the motor and it has three pairs of windings, each pair rotated 60° relative to the previous pair (Figure 5). Every pair is connected to one phase of the incoming three-phase AC voltage. Connected, these windings will create a rotating magnetic field inside the stator. In order to make this magnetic field more efficient flat winding are used. These windings are protected by an aluminium or cast-iron housing fitted with cooling fins on the outside.

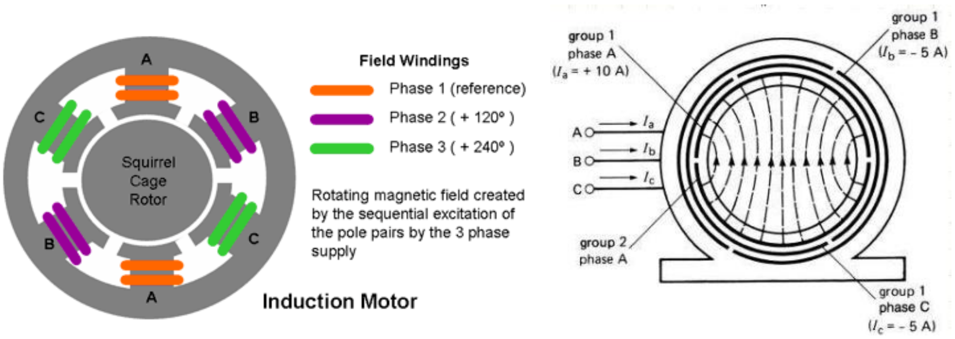

A
B
C
Squirrel Cage Rotor
C
B
A
Field Windings
Phase 1 (reference)
Phase 2 (+ 120°)
Phase 3 (+ 240°)
Rotating magnetic field created by the sequential excitation of the pole pairs by the 3 phase supply
Induction Motor
group 1 phase A (I_a = + 10 A)
group 1 phase B (I_b = − 5 A)
A
B
C
I_a
I_b
I_c
group 2 phase A
group 1 phase C (I_c = − 5 A)

Figure 5 Left: basic construction of a three-phase induction motor. [2] ; Right: more efficient flat windings. [3]

Located in the inside of the stator, the rotor is responsible for transferring the electrical energy of the stator to mechanical energy. The most common type of rotor is the squirrel cage rotor (Figure 6), as it is very rugged and reliable. It consists of longitudinal conductive bars that are connected to a conducting “shorting” ring on each side. The bars are offset on a slight angle in order to reduce the noise and vibrations.

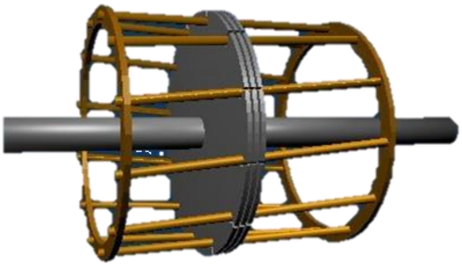


Figure 6 Basic representation of a squirrel cage rotor[4]

2.2.2 Basic working principles

In order to create torque, the induction motor relies on the principle of electromagnetic induction. A rotating electromagnetic field is generated in the stator (Figure 7), and passes through the rotor. Because the electromagnetic field is moving, and the rotor is still not moving yet, the flux lines are intersecting with the bars of the rotor. According to the law of Faraday-Lenz (1), this will create induced voltage inside the rotor bars.

$$E = B * l * v \tag{1}$$

This voltage will result in a flowing current following from the law of Ohm (2).

$$I = \frac{E}{R_{rotor}} \tag{2}$$

There is now a current flowing through the squirrel cage. If a conductor with a flowing current is positioned in a magnetic field, this conductor will experience a force called the Lorentz force. The magnitude of this force can be calculated using equation 3.

$$F = B * I * l \tag{3}$$

This force will cause a torque on the rotor and it will start to rotate. The resulting rotational speed of the rotor is lower than the rotation speed of the torque if the stator magnetic field crosses the rotor bars with a speed greater than zero. This is the reason this kind of motor is called an asynchronous motor. The difference in speed between the rotor and the stator magnetic field is called the “slip”, and can be calculated by formula 4.

$$s = \frac{n_s - n_r}{n_s} \tag{4}$$

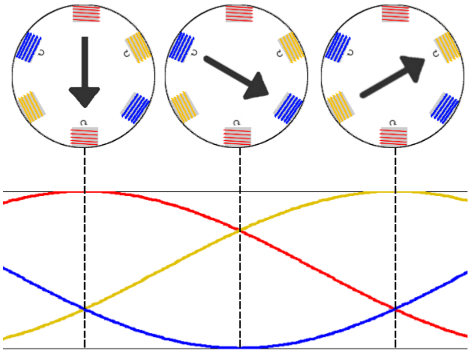


Figure 7 The stator magnetic field (top) caused by the 3-phase AC voltage (bottom)

2.2.3 Characteristics

A squirrel cage induction motor has several characteristics. Shown in Figure 8 is the current-speed characteristic. When the rotor is stationary, the current flowing through the motor is maximal. Once the motor starts accelerating, the current will decelerate. This can be explained by formula 5.

$$I_{stator} = \frac{U - BEMF}{R_{Stator}} \tag{5}$$

As explained in 2.2.2, a current is induced in the rotor bars when the motor is rotating. This current induces a separate magnetic field in the rotor, which causes a new (inversed) voltage in the stator. This inversed voltage is called the back-EMF (BEMF). The BEMF will be 0V when the rotor is stationary, since the magnetic field of the rotor does not cross the stator windings yet. Once the rotor starts rotating, the BEMF will increase and the total voltage over the stator windings will decrease. The decrease in voltage also means a decrease in current according to Ohm's law.

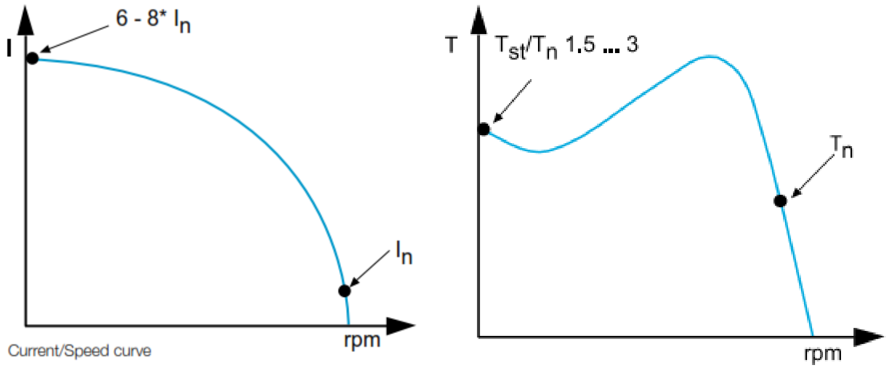


Figure 8 Current-speed characteristic[5] and Torque-speed characteristic[5]

Figure 8 shows the torque-speed (or torque-slip) characteristic of a 3-phase squirrel cage induction motor. The nominal torque can be calculated using formula 6. When the motor is turned on, a starting torque T_{st} is present. It is about 1.5 times the nominal torque, and it is important that the load torque on the motor is lower than the starting torque. If the load is higher than the starting torque, the motor cannot start, and the high starting current will keep flowing through the engine heating it up fast. Once the motor accelerates, it will reach a maximal torque T_{max} which is around 3 times the nominal torque. This is also called the breakdown-torque and is located at around 80% of the synchronous speed. After this point the torque will decrease to 0 as the speed nears the synchronous speed. The nominal torque T_n is located a little bit to the left of the synchronous speed. When the load on the motor increases, the rotationspeed will decrease. This causes an increase of the torque created by the motor axis an prevents it from stalling. When the load torque exceeds the breakdown-torque T_{max} , the motor will stall and stop.

$$M_n = \frac{9.55\ x\ P_{out}}{n_n} \tag{6}$$

2.2.4 Connection types

As explained in 2.2.1, the motor has three pairs of windings. Each pair has its own letter: U, V or W. Each end of the windings is numbered one or two. These windings can be connected to the 3-phase grid in 2 ways: star or delta (Figure 9).

If a motor is connected in star (Figure 9, left), the line voltage U_L will be standing over two pairs of windings. Because of this, the voltage over 1 pair of windings will be the phase voltage U_f (7). On the other hand, a motor connected in delta configuration (Figure 9) has the full line voltage U_L over each pair of windings.

$$U_f = \frac{U_L}{\sqrt{3}} \tag{7}$$

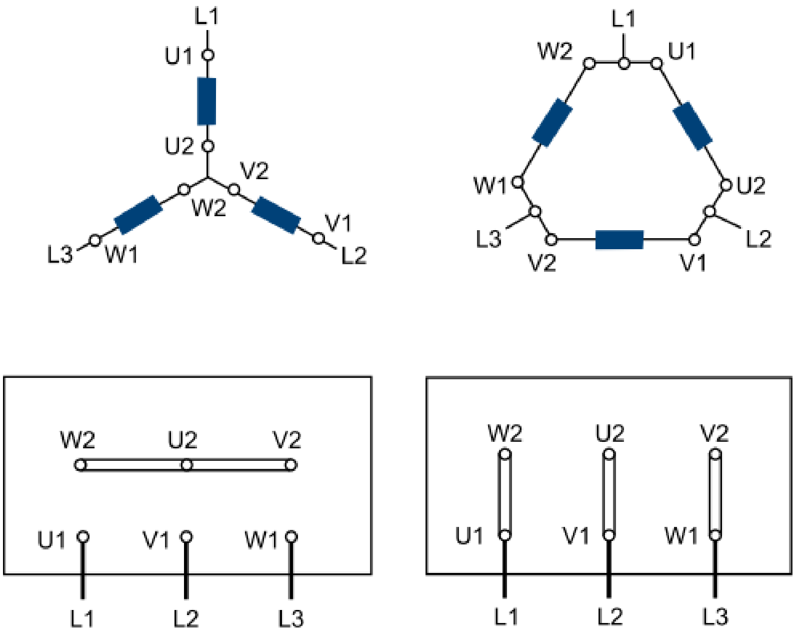


Figure 9 Motor connections. Left: Star (Y) ; Right: Delta (Δ)

2.2.5 Direct online start (DOL)

When the motor is started using the DOL method, the motor is placed in the desired configuration (star or delta) and directly connected to the grid. It is the most common and simple method, but the starting current is very high. It has the same value as the short circuit current which is about 7-14 times the rated motor current. Because of this high current, this method does have a high starting torque. Another disadvantage is that this puts a high mechanical stress on the motor.

2.2.6 Star-delta start

This kind of start reduces the starting current and starting torque. The motor is started using the star configuration. This reduces the voltage over the windings, resulting in a lower starting current (33% of the DOL current) and a lower starting torque (33%). When the motor is at full speed, the motor is rearranged to a delta configuration, providing full torque availability. Figure 10 shows the characteristics of the star-delta start. The transition from star to delta happens at 95% of the synchronous speed.

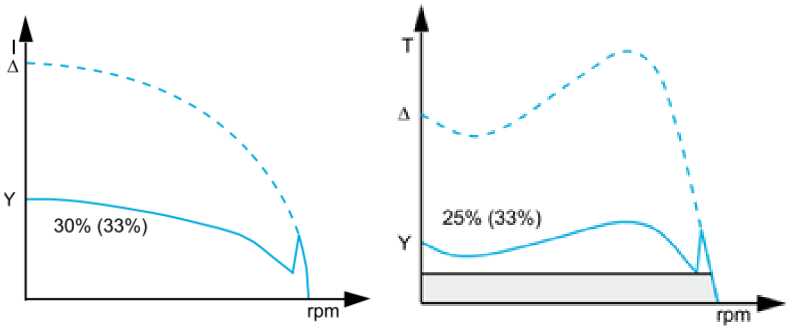


Figure 10 Characteristics of the start-delta start.

2.2.7 Soft starter start

A soft starter regulates the voltage over the motor to reduce the starting current. It ramps up the voltage from initial voltage to max voltage. First, the voltage is kept very low, in order to prevent sudden jerks. After that, the voltage and the torque increases so the motor starts accelerating. This starting type reduces the starting torque and mechanical stress on the motor.

2.3 Frequency converter

A frequency converter (“drive”) is a device that is used to control motors. It converts the AC-input power to an AC-output power with frequency that can be varied. The drive will change the output frequency to influence the motor speed and –torque, in order to get the optimal characteristics for the application it is used in.

2.3.1 Structure

Figure 11 shows the basic electrical structure of a frequency converter. The schematic can be divided in 3 main parts: Rectifier circuit, DC link and inverter circuit. The input power (U1, V1, W1) will first be rectified to a DC voltage. Next this DC-voltage will be converted back to AC power with the desired (variable) frequency using an inverter circuit. The motor will be connected to terminals U2, V2 and W2.

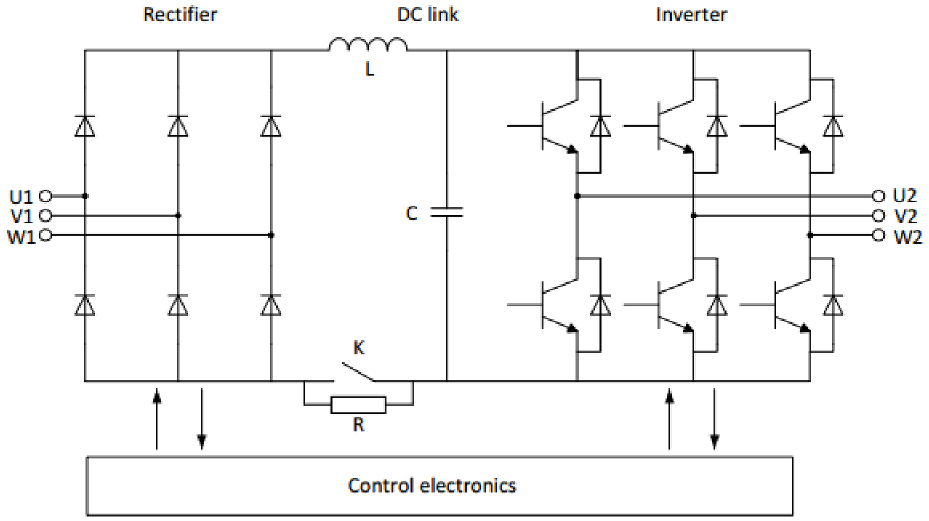


Figure 11 Frequency converter with diode rectifier and IGBT inverter [6]

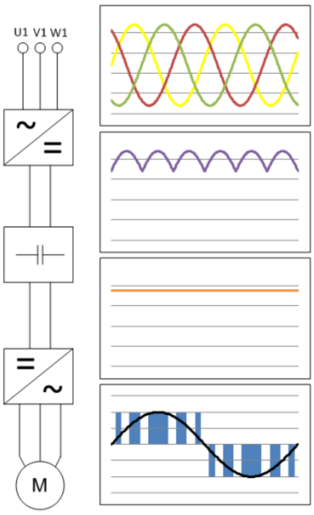


Figure 12 Voltage waveforms in different stages of the drive [6]

The rectifier circuit is a three-phase bridge rectifier, which converts the three-phase 380VAC input voltage to 520V DC-link voltage. It consists of six diodes (two per phase) that will let the current pass in only one direction. Next in the drive is the DC-link with a storage capacitor. This capacitor is big enough so it can keep the voltage coming from the rectifier as constant as possible. This allows the inverter to work very precise.

The next and final module in the structure is the inverter, the construction of the IGBT inverter is show in Figure 13.

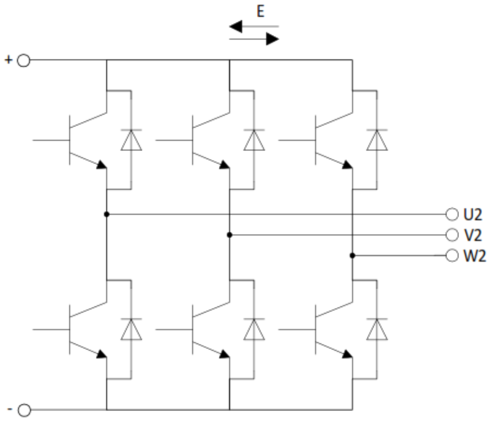


Figure 13 IGBT inverter

The IGBT inverter converts the DC voltage from the DC-link back into an AC voltage with a variable frequency. It usually contains six IGBTs , from which the top three chop the DC-voltage to the desired AC-voltage, and the bottom three close the circuit so the current can flow.

2.3.2 Scalar control

A frequency converter has many options of controlling the speed and torque from the motor. The simplest method – and also the one active in this thesis- is the scalar control. Scalar control or open-loop control does not consider the position of the rotor and the direction of the speed. It is most commonly used in application where exact rotor precision is not needed and also does not need help from an external position sensor on the rotor. The control is purely done by changing the voltage and the frequency. As long as the motor does not exceed nominal speed, a fixed V/f ratio is maintained. For this reason the scalar control is also called V/f control. A constant V/f ratio also means that a constant flux in the stator is maintained, and can be found by using the simplified steady-state equivalent circuit of the induction motor (Figure 14). In this circuit we can assume that the stator resistance R_s is zero, and that L_l is the combination of the leakage inductance of the stator and rotor. L_m are the windings generating the stator flux.

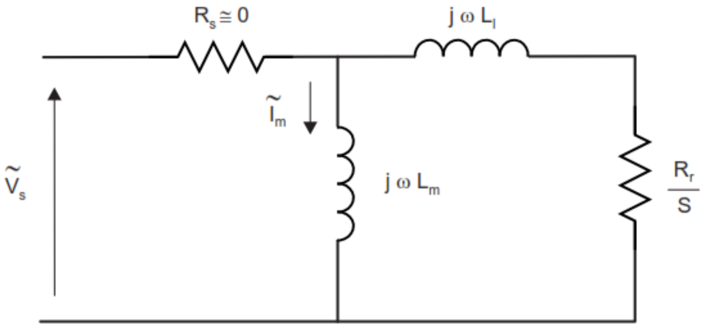


Figure 14 Steady-state equivalent circuit of the induction motor [7]

Resulting from this equivalent circuit, we can retrieve equation 8.

$$I_m = \frac{V_s}{2\pi f L_m} \qquad (8)$$

I_m is responsible for flux in the stator. Resulting from equation 8, keeping a constant V/f ratio keeps the value of I_m constant resulting in a constant stator flux.

Figure 15 shows the progress of the characteristics curve. By using a constant V/f, the nominal torque moves to the right if the frequency increases.

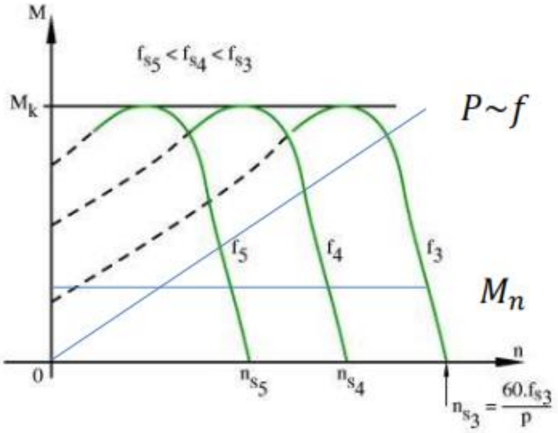


Figure 15 Progress of the M-n characteristics curve by using V/f control.

However, at low frequencies (0-f_c) a problem arises (Figure 16). Low frequencies result in a low impedance of the stator windings, meaning that the stator resistance can't be neglected anymore. Because of this, the flux (and resulting torque) would drop faster than wanted in V/f control. In order to compensate this, an extra voltage compensation is added for low frequencies. This compensation is called the I/R compensation, and can be easily modified in the drive parameters. The region from f_C until f_{rated} is the linear V/f part. In this range a constant V/f ratio is maintained. When the speed set point is set on the rated frequency f_{rated}, the voltage reaches its maximal amplitude. Consequently, when the motor has to spin faster than the rated speed, the voltage cannot increase. This causes the available torque to decrease (8).

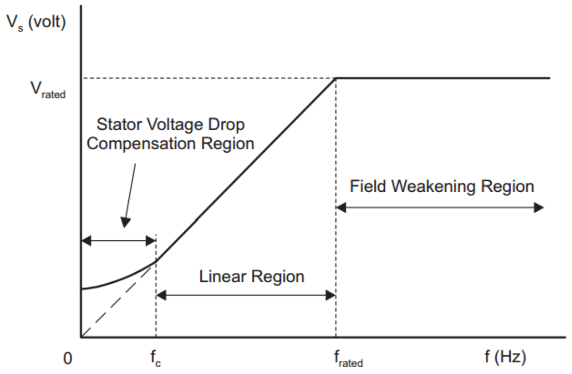


Figure 16 Stator voltage versus frequency profile using scalar control [7]

2.4 Magnetic particle brake

In order to be able to apply an extra load to the motor axis, a brake is installed in the setup. The magnetic particle brake consists of a rotor connected to the motor and a fixed housing (Figure 17). Between these components a cavity is provided filled with magnetic particles. In the housing are coils that can provide a magnetic flux through the brake. When there is no electrical current, no magnetic field is created so the magnetic particles remain in the cavity. As a result there is no braking force. However when an electrical current flows through the coils, a magnetic flux is created which tries to bind the particles together [8]. The rotor tries to pass through these bound particles, creating a resisting force on the rotor. The more current is supplied, the stronger the particles will bind and the higher the braking force will be. The advantage of this type of brakes is that the torque can be controller very accurately, with an almost linear voltage to torque ratio.

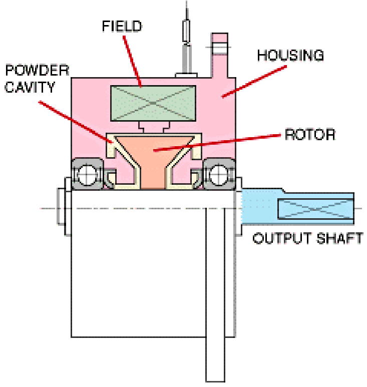


Figure 17 Magnetic particle brake. [8]

2.5 Incremental encoder

2.5.1 Working principal

An incremental encoder generates pulses at a frequency proportional to the rotational speed of the axis it is connected to. A spinning disk with radial slits is connected to the input shaft and fixed slits are mounted to the housing (Figure 18). On one side of the rotating disk LED’s shine light on the disk, while on the other side photo transistors are mounted behind the fixed slits. Every time the light can pass through a slit, a pulse is generated.

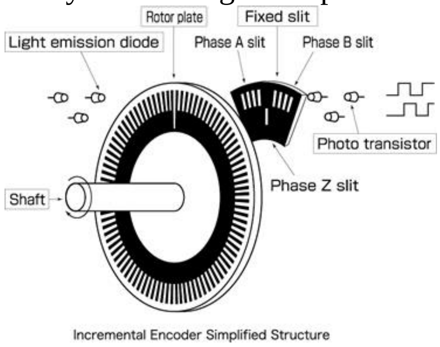


Figure 18 Simplified structure of an incremental encoder.[9]

Technically, only one LED/phototransistor pair is enough to measure the rotational speed of the shaft. However, if also the direction of the rotation is needed, an extra pair of LED/phototransistor is added, which is shifted half a slit (90° electrical). Now one pair is the “A” pair, and the other is the “B” pair. This configuration is called the “quadrature” configuration. If the A pulse rises followed by the B pulse, the axis is rotating CW, while is the B pulse rises before the A pulse the axis rotates CCW. This way both the speed and direction can be determined. Usually, a third pair is added that only gives 1 short pulse each full rotation. This is called the “0”, “C” or “Z” pair.

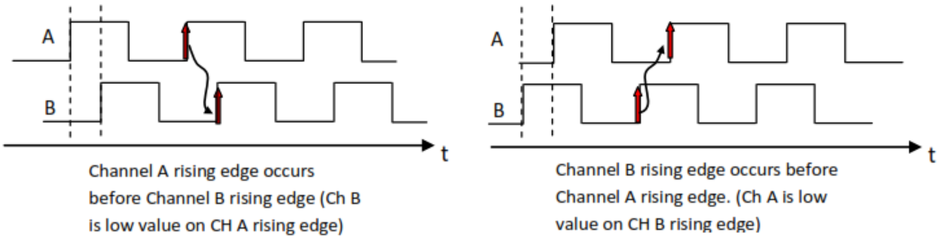

A
B
t
Channel A rising edge occurs before Channel B rising edge (Ch B is low value on CH A rising edge)
A
B
t
Channel B rising edge occurs before Channel A rising edge. (Ch A is low value on CH B rising edge)

Figure 19 Determination of the direction in quadrature output signals [10]

2.5.2 RS422 TTL output

The encoder used in this thesis has an RS422 TTL output signal. RS422 is called a balanced differential line. Figure 20 shows this configuration for the A-channel. A similar configuration is used for the B and 0 channels. The differential line configuration creates a certain noise immunity, since the receiver reads the difference between the A and /A signal. Common mode noise will be actively cancelled out this way. The TTL outputs means that the signal levels are generated using TTL logic. Resulting from this means that the “high” level is a signal greater than 2.5V, and a “low” level is a signal lower than 0.5V.

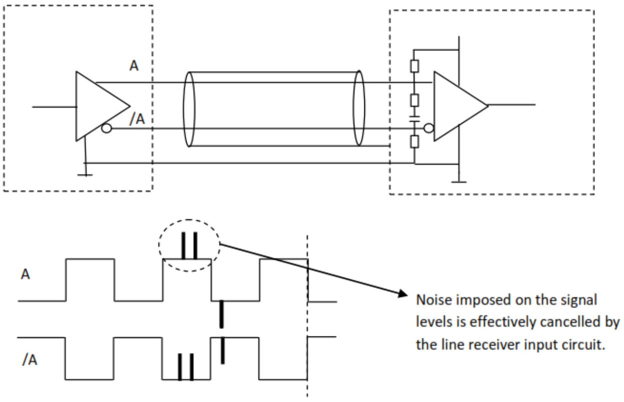

A
/A
A
/A
Noise imposed on the signal levels is effectively cancelled by the line receiver input circuit.

Figure 20 RS422 differential line output for the A-channel. [10]

2.6 Load cell and amplifier

A load cell is used to measure forces. It is a transducer whose electrical signal is directly proportional to the measured force. The load cell in this thesis is a strain gauge load cell.

The strain gauge load cell uses 4 strain gauges configured in a Wheatstone bridge (Figure 21). An input (excitation) voltage is placed over two sides of the bridge. When a force is applied, the resistance of the strain gauges will change (two will increase, other two will decrease), resulting in a voltage between the “signal +” and “signal –“ wires. However, the amplitude of this signal is very low. The FSO of the used load cell is 2mV/V, which means that when 10V excitation is applied, the signal difference between no load and full load is only 20mV. This is not high enough, so before connecting to the PLC, the signal needs to be amplified by an amplifier with the same FSO as the load cell.

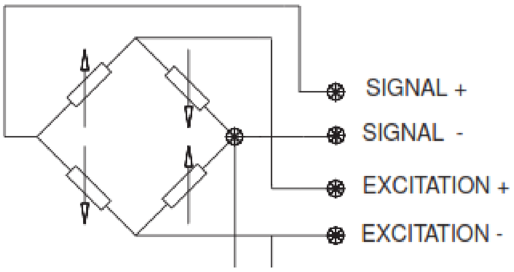


Figure 21 Wheatstone bridge of the strain gauges inside the load cell. [11]

2.7 PROFINET

2.7.1 Network

PROFINET is an Ethernet based Industrial Ethernet communication network. It differs from normal Ethernet connections in that it can communicate much faster and in real time. In order to do this PROFINET uses three protocol levels (Figure 22):

- TCP/IP level for normal IT services. Cycle is time around 100ms.
- RT (Real Time) level. This is used for standard I/O systems and simple drive solutions. Cycle time is around 10ms up to 1ms.
- IRT (Isochronous real time). This is used for high speed applications such as fast I/O systems or motion control. The cycle time here is up to 31.25 µs.

PROFINET RT uses cyclic data exchange, which means that data is exchanged between controller and device in a predefined cycle time. In order to achieve this cycle time, RT data is prioritized over standard Ethernet data.

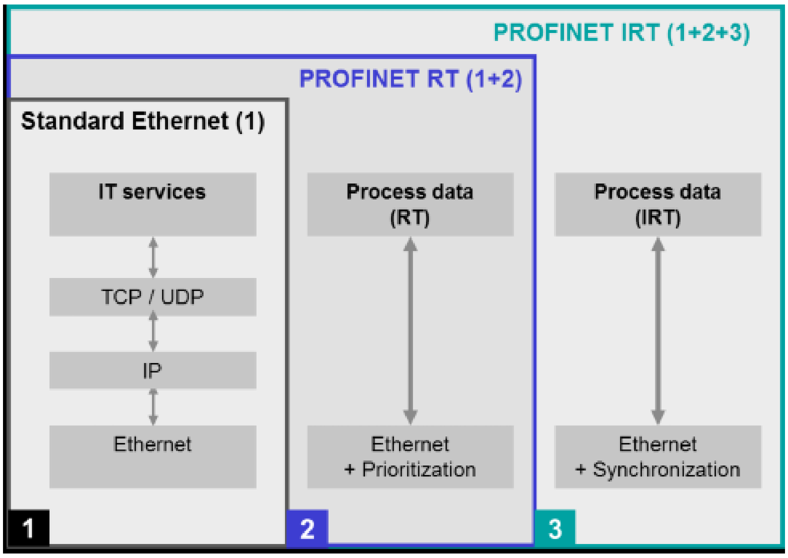


Figure 22 PROFINET communication channels

2.7.2 PROFIdrive

PROFINET exchanges data transparently by default. The received or sent data has to be interpreted by the user in the PLC or PC solution. In order to make communication easier, application profiles for PROFINET were developed. These are joint specifications concerning certain properties, performance characteristics and behaviour of devices and systems that are developed by manufacturers and users [12]. Generally, there are two different types of profiles:

- General application profiles
  These can be used for different examples, such as PROFIsafe for implementation of functional safety, or PROFIenergy for low energy applications.
- Specific application profiles
  These are developed for specific types of applications, such as PROFIdrive, which is a profile specifically made for communication with drives.

As mentioned above, communication between the PLC and the drive will happen using the PROFIdrive application profile. This way the communication happens in the most efficient way, with minimal cycle times and easy control of the drive.

3 INITIAL SITUATION

3.1 Setup

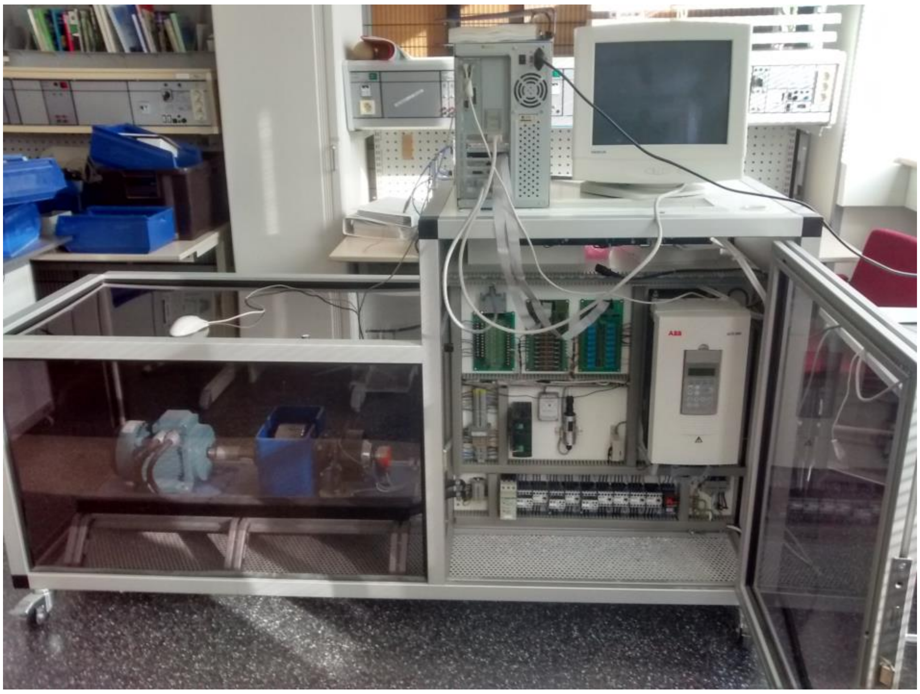


Figure 23 Complete setup

The test unit contains three groups of components: motor unit, control unit and PC with communication. The motor and brake are both controlled by the main control unit. This control unit in its turn communicates with the PC through old parallel connections.

Depending on what the user clicks on the PC, the control unit will execute the demanded tasks and send back all measured data necessary for the PC. Chapter 3.4 contains more detailed information about the functions.

The complete setup was made as a thesis in the year 2000 by a Finnish student called Osmo Vuotila. His thesis has been written completely in Finnish, which made it very hard to get information about the working principles, since this thesis was made by an exchange student who cannot speak Finnish.

3.2 Hardware

3.2.1 Motor unit

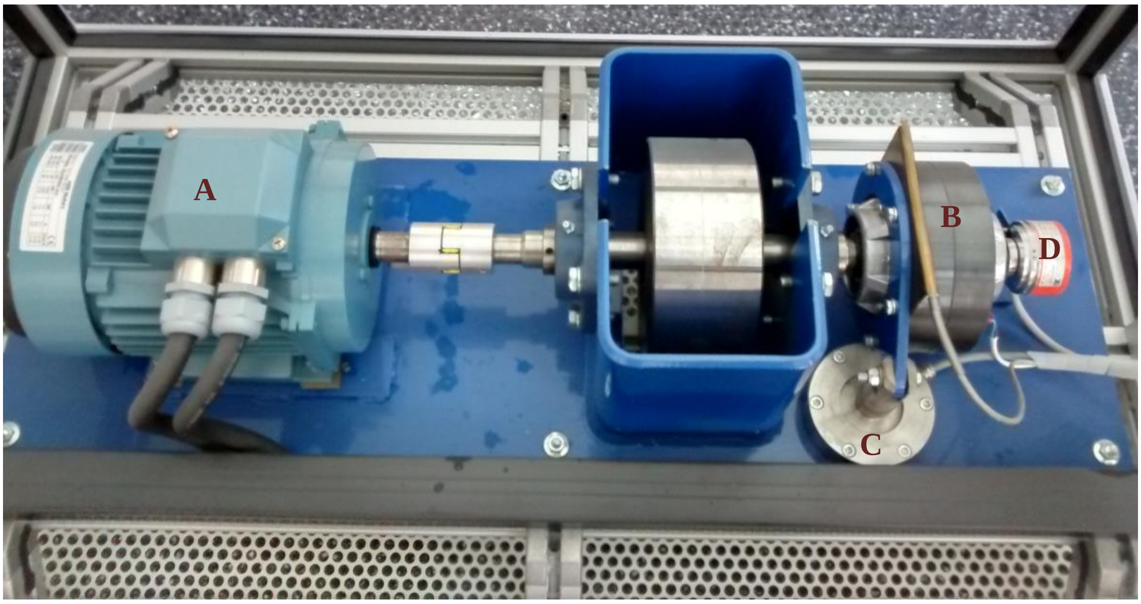


Figure 24 Motor unit with motor, mass, brake and encoder

Motor (A)
The biggest and main part of the motor unit is the motor itself. It is a 1.75kW three-phase asynchronous induction motor from ABB. It is controlled by the ABB ACS600 frequency converter in the control unit, and is mechanically connected to a mass that acts as the load.

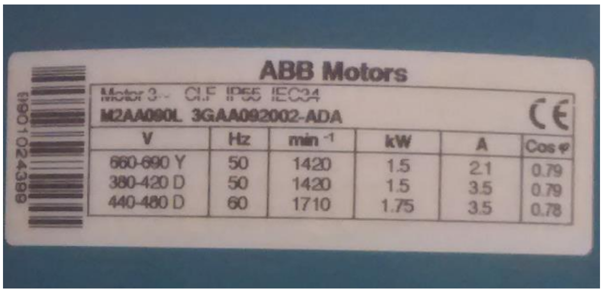


ABB Motors

Motor 3~ Cl.F IP55 IEC34
M2AA090L 3GAA092002-ADA

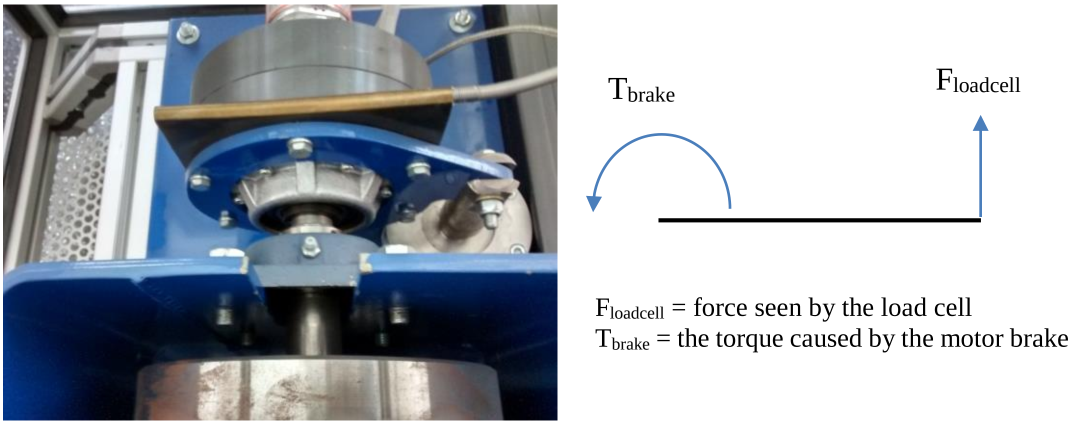
V	Hz	min -1	kW	A	Cos φ
660-690 Y	50	1420	1.5	2.1	0.79
380-420 D	50	1420	1.5	3.5	0.79
440-480 D	60	1710	1.75	3.5	0.78

Figure 25 Motor nameplate

Brake (B)
On the other end of the mass a magnetic particle brake is connected. The brake is a Lenze type 04 and has a maximum braking torque of 40Nm. It is used primarily for testing the overall power-to-slip characteristic from the motor, but can also be manually operated with the PC. It is controlled by a 0-10VDC analogue signal coming from the control panel, where 0V is no braking power and 10V is maximum braking power. The temperature of the brake is measured by a PT-100 resistor inserted into the brake.[13]

Load cell (C)

The braking force is measured by a load cell, mounted as shown below. If the brake starts applying more force, it will put more pressure on the load cell, so the force on the load cell is a unit for the applied braking force.



$F_{loadcell}$ = force seen by the load cell
T_{brake} = the torque caused by the motor brake

Figure 26 Left: Brake with load cell. Right: Occurring forces

Encoder (D)

Speed measurement is done with the incremental encoder. It is made by Kübler (type 5820-1541-5000) and has a pulse rate of 5000 pulses/revolution. Communication is established by an RS-422 connection to the control unit. Here the I/O panel measures the time between 2 pulses to measure the rotation speed of the motor. Direction is not measured.

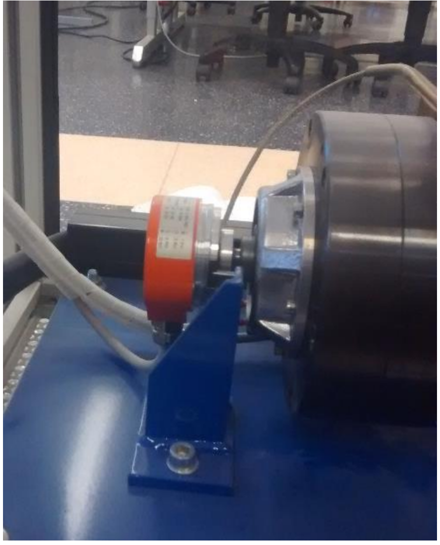


Figure 27 Encoder

3.2.2 Control unit

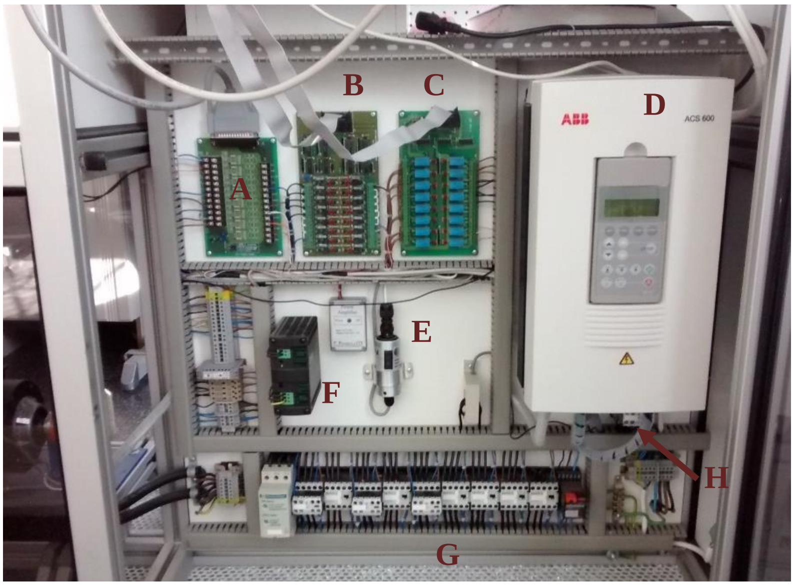


Figure 28 Control unit

The control unit is the heart of the setup. In this part all the components of the motor unit are controlled, information about current, power and rotationspeed is gathered and send to the PC.

Analog IO-panel (A), digital input panel (B), digital output panel(C)
All of the signals coming from and going to the components in the control unit and motor unit are connected to these panels. Each panel is connected to the PC with his own parallel interface, and because of this it will be impossible to connect it to a modern PC. Therefore the panels have to be replaced by new components so a connection to a PC can be established.

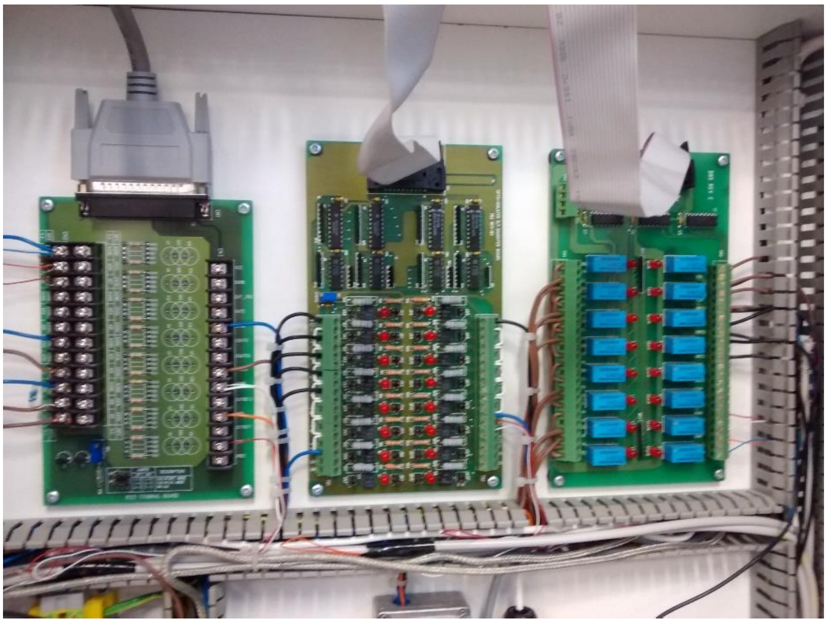


Figure 29 IO-panels

Frequency converter (D)
The frequency converter is an ACS600 from ABB, and is also equipped with digital and analog inputs and outputs. These IO's are used to control the motor when the manual frequency converter mode is active, and is the only way of communication. The frequency converter also returns its own measured current as well as power consumed by the motor. There are no possibilities to use any Ethernet or bus-connection to communicate with it. Because of the lack of communication possibilities and the overall age of the frequency converter, it has been decided to replace it by a new one in this thesis.

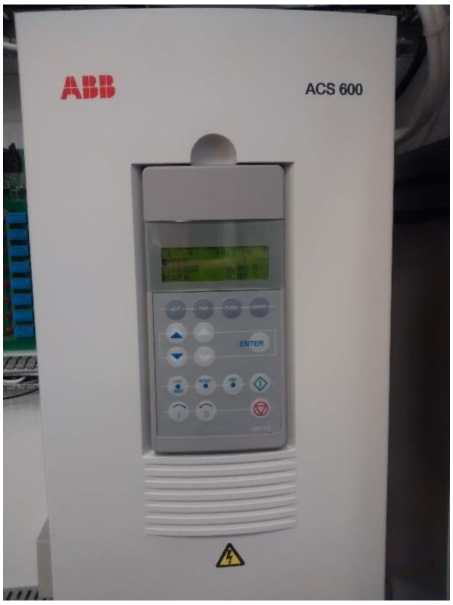


Figure 30 ABB ACS600 Frequency converter

Power amplifier and transducer (E)
The output current of the IO-panels is not sufficient to power the brake, so an amplifier is placed between the brake and the IO-panel. The amplifier has an input voltage of 0-10VDC and output voltage of 0-24VDC/1A.

The transducer converts the small voltages coming from the load cell to a measurable voltage for the IO-panel.

24VDC power supply (F)
This 24VDC power supply provides 24VDC to all the components that need a 24V supply/signal. It has a maximum output current of 5A.

Contactors (G)
Because the outputs of the IO-panels do not provide enough voltage or current, contactors are used to connect various parts of the control unit and motor unit to the 230/400V lines. They are controlled by the digital output panel.

The soft starter is located on the far left of the contactors, and is used when the program is in “soft start mode”.

Main contactor
This contactor supplies the main voltage to the whole system. It is connected to the emergency stop, so when the emergency stop is activated, the complete system loses power.

3.2.3 PC

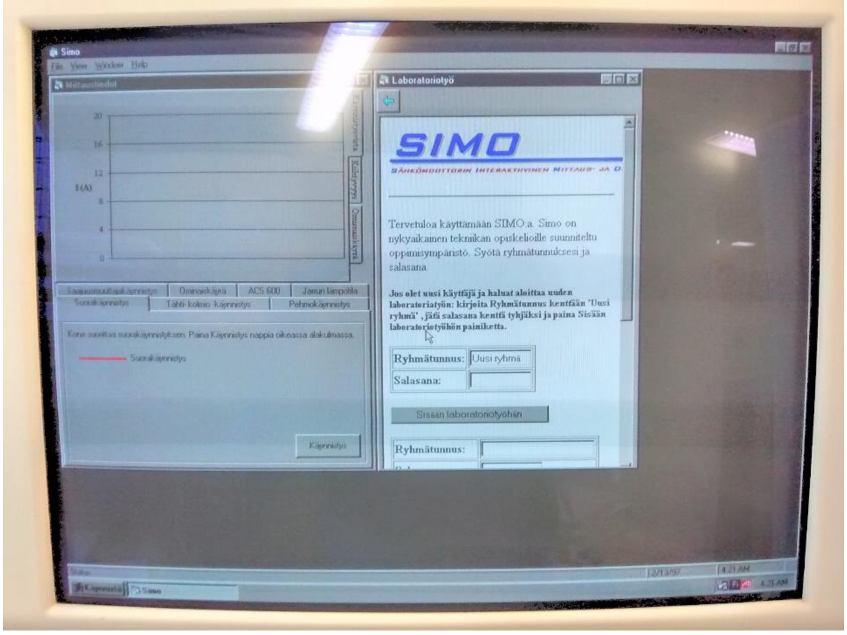
Figure 31 The old windows 95 PC

The PC is an outdated Windows 95 desktop with 192 MB of RAM and a 400MHz Intel Pentium II processor. It communicates with the control unit by 3 separate parallel interfaces. It will be replaced by a new PC with the Windows 7 operating system.

3.3 Software

There are 2 versions of the software: the original (working) software running on the Windows 95 PC, and the new version made by Vincent Peerlinck in 2012. [14]

The original software is called "SIMO" and is currently still being used by the students. The main problem with this software is that it only works on the old computer, and will not be able to communicate with the new hardware that will be installed. For this reason new software has already been developed.



Kone suorittaa suorakäynnistyksen. Paina Käynnistys nappia oikeassa alakulmassa.

Tervetuloa käyttämään SIMO:a. Simo on nykyaikainen tekniikan opiskelioille suunniteltu oppimisympäristö. Syötä ryhmätunnuksesi ja salasana.

Jos olet uusi käyttäjä ja haluat aloittaa uuden laboratoriatyön: kirjoita Ryhmätunnus kenttään 'Uusi ryhmä' , jätä salasana kenttä tyhjäksi ja paina Sisään laboratoriotyöhön painiketta.

Ryhmätunnus: Uusi ryhmä

Salasana:

Ryhmätunnus:

Figure 32 Original software "SIMO"

This new version of the software is fully working, but has no ways of communicating with the hardware. It has been developed in Microsoft Visual Studio 2010 using C# and works on Windows 7 and newer. An important part of this thesis is to complete this software by implementing communication between the PC and the hardware.

3.4 Functions

The current software has seven different modes:

- Direct start
- Star-delta start
- Start with soft starter
- Start with frequency converter
- Power-to-slip characteristic
- Manual mode
- Brake temperature gauge

By clicking the start button in the first four modes, the motor will be started accordingly to the selected mode. Meanwhile the current through the engine is being measured and stored. After a few seconds the motor is disconnected again and the program will show the current that has been flowing through the engine in function of the time. This way students can compare the difference in the starting modes, and the difference in start-up current.

When the power-to-slip characteristic mode is activated, the motor starts and the brake will gradually increase torque until the motor stops. After that the brake releases and the motor is disconnected. Students can then see the power/slip characteristic on the screen.

In manual mode the students can control the motor speed, direction and brake torque. The speed and direction are both controlled by the frequency converter. The temperature of the brake gauge can be seen on the screen when the last mode is activated.

3.5 Components requiring an upgrade

3.5.1 IO-Panels

Firstly, all of the IO-panels are connected to an old parallel interface to the PC. The digital input and output panels are connected directly to the motherboard of the old PC. If the PC will be upgraded, it will be impossible to connect these. Secondly, all of the hardware components are directly operated by the program of the PC. This means that the PC sends the signals to all of the hardware components separately. The disadvantage of this is that when the PC crashes there is no control of the hardware components anymore, which can sometimes lead to dangerous situations. Resulting from all these arguments, it has been decided to upgrade/replace the IO-panels.

3.5.2 Frequency converter

The currently installed frequency converter is made in 1999. It is very bulky in size and has no options of communication installed. Because of this, parameters of the frequency converter could only be modified by using the operator panel located on the frequency converter. In order to gain more flexibility, and to make sure this test unit will still work after 20 years, it has been decided to replace the frequency converter as well.

3.5.3 Communication

Following from the decisions of upgrading the IO-panels and frequency converter, a new way of communication is also needed. One of the goals is to make the system more flexible, so instead of using separate hardwired connections between the PC, PLC and frequency converter, a communication protocol will be used. This way they will be easily replaceable in case of a failure, and it will be very easy to expand the setup as well. Which types of communication is used will be explained later in the thesis.

4 NEW HARDWARE

4.1 Layout

Since the old system used very old parallel connections to the PC and analog signals to control the frequency converter, it has been decided to change these connections and make them easier and more flexible. In Figure 33 the top diagram represents the old layout. The bottom diagram is the new layout, which uses an independent PLC. This PLC communicates with the new PC using an Ethernet connection and communicates with the new inverter by using a PROFINET connection. It is already clear that the new layout is much easier to implement and maintain.

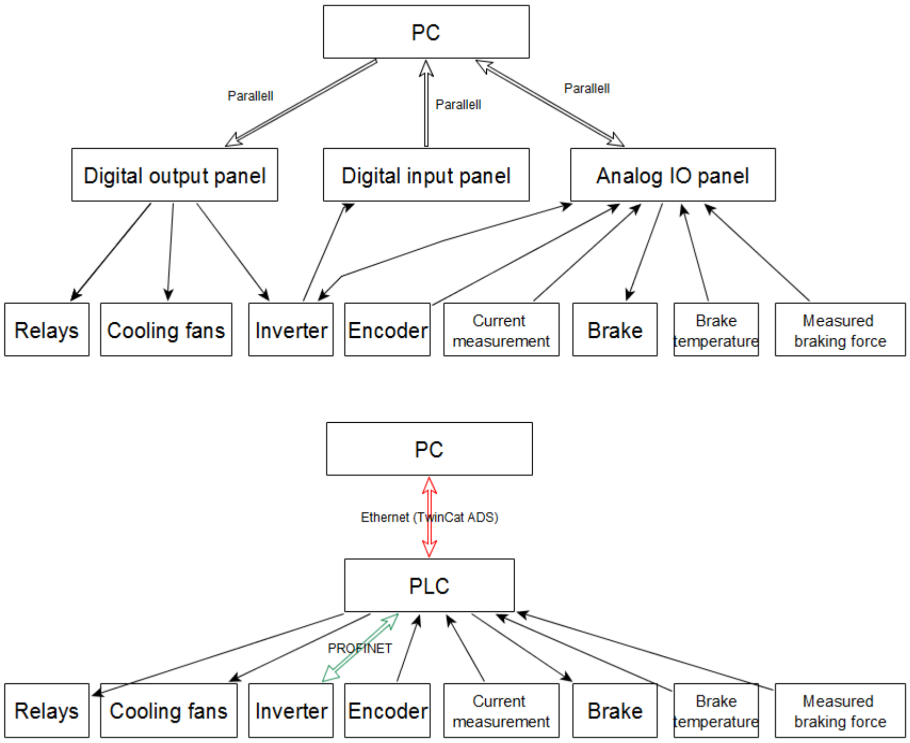


Figure 33 Top: old hardware layout. Bottom: new hardware layout.

4.2 PLC

4.2.1 Replacing the old I/O panels

As mentioned in “3.5.1 IO-Panels”, the IO-panels were connected to the PC using old parallel connections. Also, the panels only converted signals to and from the PC. There was no intelligence present in the panels, which means that all of the hardware components were directly operated by the program of the PC. The disadvantage of this was that when the PC crashed there was no control of the hardware components anymore, which could sometimes lead to dangerous situations. In order to achieve a safer and more reliable system, a form of intelligence needs to be added between the PC and the hardware components.

In order to gain this intelligence, a PLC will be added between the PC and the hardware, replacing the IO-panels. It will be in control of all the hardware components, and retrieve all data from the mounted sensors. Also, communication with the new frequency converter will be managed by this PLC through PROFINET.

4.2.2 Configuration overview

Beckhoff has a modular I/O terminal system, which makes it easy to replace, add and remove I/O terminals. Every bus terminal can communicate with the controller through a build-in bus that is directly integrated in the bus terminal. Every terminal that is added is automatically connected to this bus. Depending on the name of the bus terminal the terminal has the K-bus (KLxxxx) or the Ethercat-bus (ELxxxx). Both types are incompatible with each other, and need a coupler to transfer data from one bus to another. It is only possible to transfer from the E-bus to the K-bus. Once the coupler from E- to K-bus is inserted, you cannot put a coupler after it to transfer to the E-bus. Thus all the E-bus terminals must be placed, followed by an E- to K-bus coupler and then all the K-bus terminals must be placed.

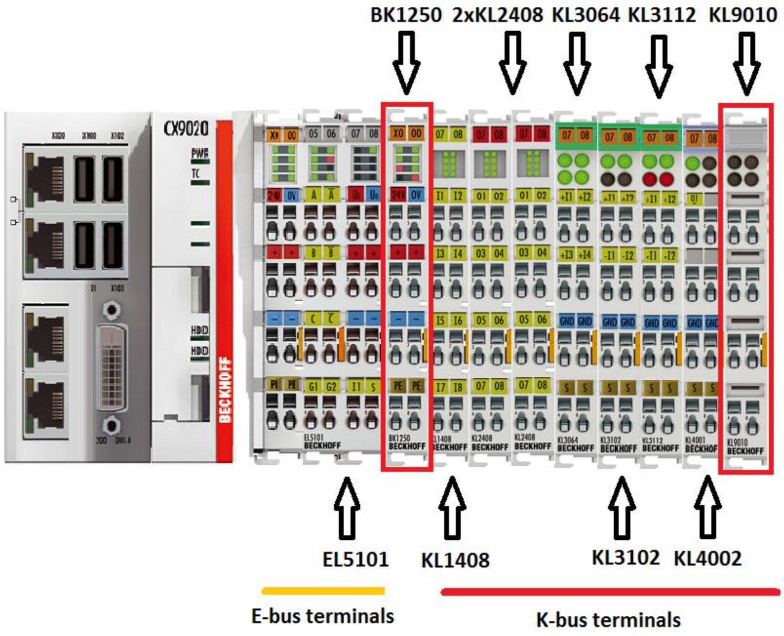


Figure 34 Complete PLC configuration

The complete PLC configuration is shown in Figure 34. The first element is the controller with the Ethernet and PROFIBUS ports. Following is the EL5101 encoder interface. This is the only E-bus terminal, so this is followed by a BK1250 coupler terminal so all the K-bus terminals can be connected. The next element is the KL1408 digital input terminal, followed by two KL2408 digital output terminals KL2408. The four next terminals are dedicated to analog signals: 0-10V input (KL3064), 0..20mA input (KL3112), -10..+10V analogue input (KL3102) and 0..10V analogue output (KL4002). The final terminal is the KL9010 end terminal.

4.2.3 Controller

Many projects in HAMK use the local PC to both program and run the PLC program. This PC is used as the controller and is connected to a bus coupler to communicate with the I/O Beckhoff terminals. The main problem with this method is that when the PC is turned off or crashes, the control over the I/O terminals is also lost, as the connection between the PC and terminals is lost. In order to make sure the control over the motor will be maintained at any time, a dedicated controller will be used instead of a PC. This way the controller can still maintain control over all the hardware components when the PC is turned off or crashes.

The controller is a Beckhoff CX9020-0110-M930. This controller is equipped with its own CPU and integrated TwinCAT 3 runtime, so it does not need another PC to operate. It is also equipped with two RJ45 Ethernet interfaces, from which one will be used to connect the PLC to the PC and send the necessary data. This will also be used to program the PLC. Next, there are also 2 RJ45 PROFINET RT controller interfaces (option M930). One of these interfaces will be used to establish PROFINET communication between the ABB frequency converter and the PLC. The controller also has 4 USB 2.0 interfaces and a DVI connection (screen). These will not be used, but can be used for later projects or when this test unit will need future upgrades. Right next to the CPU module is the power supply terminal for the electrical connections.[15] This module requires 24V DC power supply, which is already present in the current electrical setup of the motor test unit. Figure 35 shows a CX9020 controller. The controller in the project has two PROFINET interfaces instead of the RS485 interface shown in the picture.

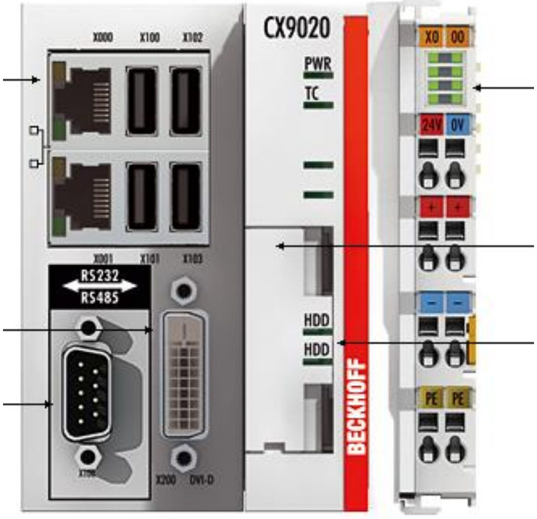


Figure 35 Beckhoff CX9020 controller[15]

On the right side of the controller different I/O busterminals of Beckhoff can be added. The CX9020 controller has an E-bus, so first all the ELxxxx components must be inserted.

4.2.4 Encoder terminal EL5101

The encoder used in this thesis has RS422 TTL output signals. In order to be able to read these signals, this specific encoder interface terminal is required. It is able to read all the input signals, and provide 24V power input to the encoder. The E-bus version (EL5101) of this interface terminal has been chosen, since the delivery time was only a few days, compared to 2 weeks with the K-bus version (KL5101). As explained in chapter 4.2.2, the first elements in the system must be the Ethercat busterminals, so this terminal is mounted first next to the controller.

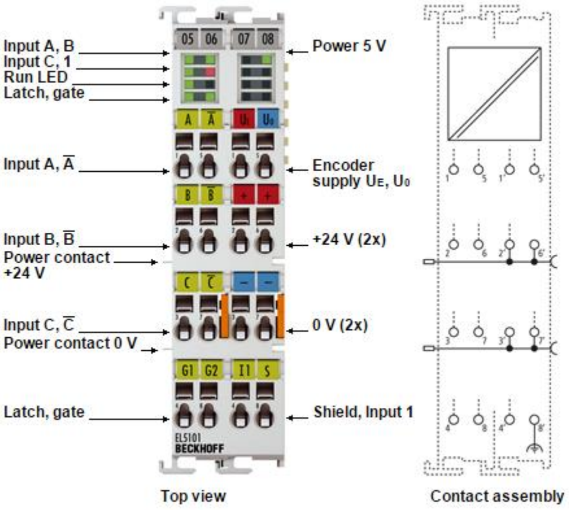


Figure 36 EL5101 channels and electrical layout.

4.2.5 Bus coupler terminal BK1250

All of the next terminals in the configuration will be K-bus terminals, since HAMK Valkeakoski has a big stock of these. In order to transfer from the E-bus to the K-bus, this buscoupler is required (Figure 37). This terminal connects the E-bus to the K-bus terminals. It also needs its own power source (24VDC) in order to be able to operate. After this terminal up to 64 extra terminals (K-bus) can be added to the system.

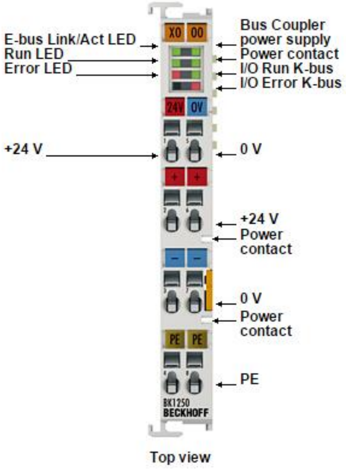


Figure 37 BK1250 terminal.

4.2.6 Digital input terminal KL1408

Because the testing unit has three fuses and an emergency stop, digital inputs are required. The KL1408 digital input terminal acquires binary data from the system and sends it to the controller. It has eight input channels which is double of what is necessary. This is done to be able to expand, and because these were already available in HAMK. Each input channel is equipped with an indication LED so it is easy to see if a channel gets an input or not. Also, every input signal is electrically isolated from the controller, since it has an opto-coupler for each channel.

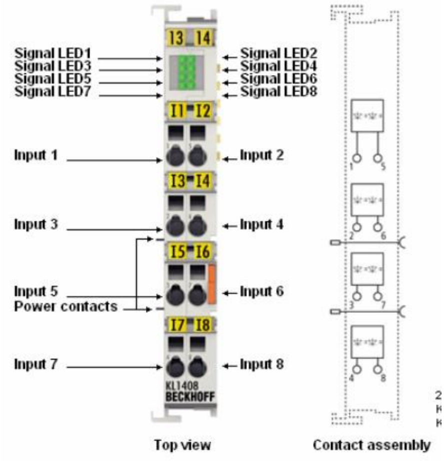


Figure 38 KL1408 channels and electrical layout

4.2.7 Digital output terminal KL2408

There are 2 digital output terminals present in the system. Each KL2408 has 8 digital outputs that are electrically isolated from the controller. All the relays that are in the system, are controlled by these terminals.

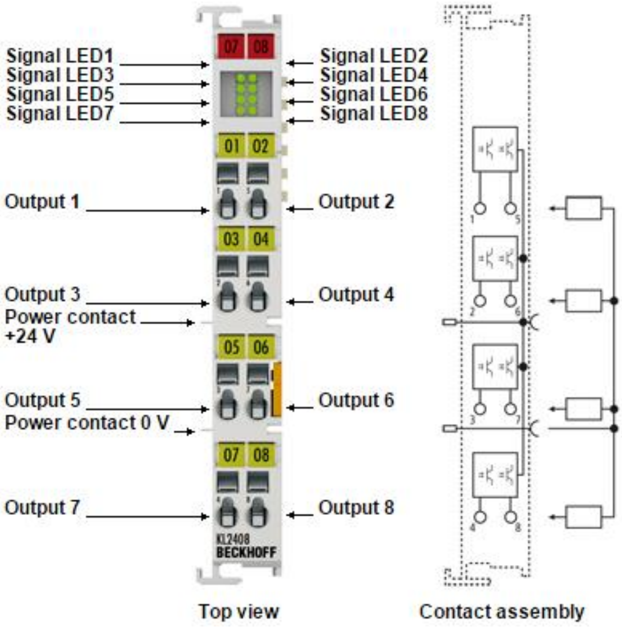


Figure 39 KL2408 channels and electrical layout

4.2.8 Analog input terminal 0..10V KL3064

This analogue input terminal measures an input value ranging from 0V to 10V DC. The voltage gets digitized to a resolution of 12 bits and is transmitted to the controller. It takes approximately 4ms to convert the value, which is by far fast enough for the application it has to measure. It is connected to the pt100 converter module from Dataxel, which gives 0V and 0°C, and 10V and 300°C.

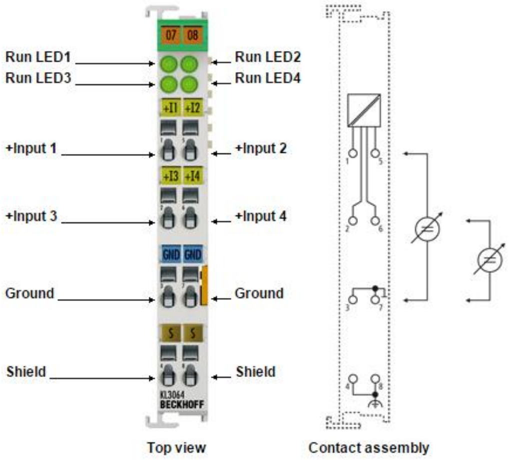


Figure 40 KL3064 terminal and electrical layout

4.2.9 Analog input module 0..20 mA KL3112

The KL3112 analogue input module measures signals in the range of 0 to 20mA and converts it into a 15bit number. It uses a differential input, which means the circuit of the signal is electrically floating with respect to a ground reference. This means the sensor to connect usually has 2 signal cables (+ and -) and 1 ground cable.

However, a number of problems occurred while using this analogue input together with the sensor:

- First of all the ammeter it is connected with has a 0..20mA single ended output (2-wire), while the analogue module uses differential inputs (3-wire). This problem has been solved by connecting the signal wire of the sensor into the “+input 1” side of the terminal (Figure 41). In order to now mimic a differential signal, the “-input 1“ side got connected to the “Ground” side, which in its turn is connected to the 0V signal of the sensor. This way the single ended output signal is connected to a differential input.
- Secondly, this terminal takes 140ms to convert an analogue input value to a number. This is way too slow, since the motor only needs 400ms to go from 0 to nominal speed when directly started. This would result in only 3 samples, which is insufficient. In order to make this terminal faster, it had to be configured in TwinCAT using the build in registers. It has now been configured to only take 10ms to convert an input value. Resulting in much more samples when the motor started, this gave a much better and smoother graphic.

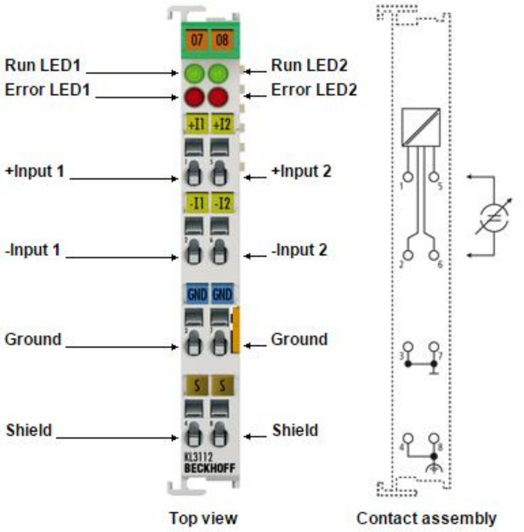


Figure 41 Terminal KL3112 and KL3102

4.2.10 Analog input module -10V…+10V KL3102

Because the force sensor in this setup has an output signal of -10V to +10V, this terminal is necessary to measure the braking force. Just like the KL3112 module, this module also has differential inputs (Figure 41). It has a resolution of 16 bit, and needs 140ms to sample. This sample rate is not fast enough, so it has been configured to be 10ms. Also, the output signals of the sensor are single ended, so the "-Input 1" terminal got connected to the "Ground" terminal in order to mimic a differential signal.

4.2.11 Analog output module 0..10V KL4002

The final I/O terminal is used to control the magnetic particle brake in the unit. It has an output voltage of 0V to 10V, with a 12bit resolution.

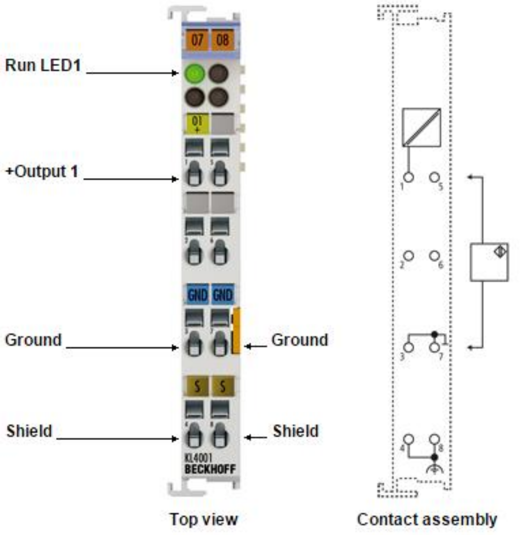


Figure 42 KL4002 output terminal

4.3 Frequency converter

4.3.1 Usage

The frequency converter (or “drive”) is used to give students more control over the motor, and to show the influence on the starting current when starting using this drive. When the manual program mode is selected, students can manually control the speed and direction of the motor.

4.3.2 Replacement

In the old system an ABB ACS600 drive was used. It was controlled by electrically connecting analog and digital inputs of the drive to analog and digital outputs on the old IO cards. Also speed and current of the motor was transmitted from the drive to the IO cards by electrical connections. In other words, the communication between drive and PC was very complex.

In order to simplify the communication, it was decided to use communication through some kind of bus or ethernet protocol. The old drive had none of these capabilities. Adding that the drive was already 15 years old lead to the decision to replace it by a new drive fitted with communication possibilities.

4.3.3 New drive

The new drive had to be cheap, yet student proof and had to be able to use PROFINET communication. According to T.Väisänen, ABB drives of the ACS355 series were very cheap yet good. HAMK already had more than 10 of those drive for students to learn with. Also, this drive can fit an extra module for PROFINET communication. Resulting from this was an ABB ACS355 drive for motors with 2.2kW nominal power. This is higher than the 1.75kW motor that is installed in the system, in order to make it more “student proof”. An optional FENA-11 PROFINET module is fitted to the drive, so it can use PROFINET communication with the controller.

Figure 43   ABB ACS355

4.3.4 PROFINET communication

In order for the drive to be able to communicate by PROFInet, an additional adapter had to be ordered. In this case FENA-11 adapter for ABB ACS355 drives has been chosen. It has 1 RJ-45 Ethernet port, and supports PROFINET, Modbus TCP and normal Ethernet/IP.

4.3.5 Modified parameters

For the drive to be able to work with the PLC, a number of parameters had to be changed. The list (and order) of the parameters to be modified can be found in appendix 3.

4.4 PC

The main starting point of the thesis was that the computer had to be upgraded. The new PC is a standard desktop with 1 RJ-45 Ethernet port that will be used to connect to the PLC. It has a 3.3 GHz quad-core processor and 16GB of RAM. The operating system is Windows 7.

5 PLC PROGRAM

5.1 TwinCAT 3

5.1.1 eXtended Automation Engineering

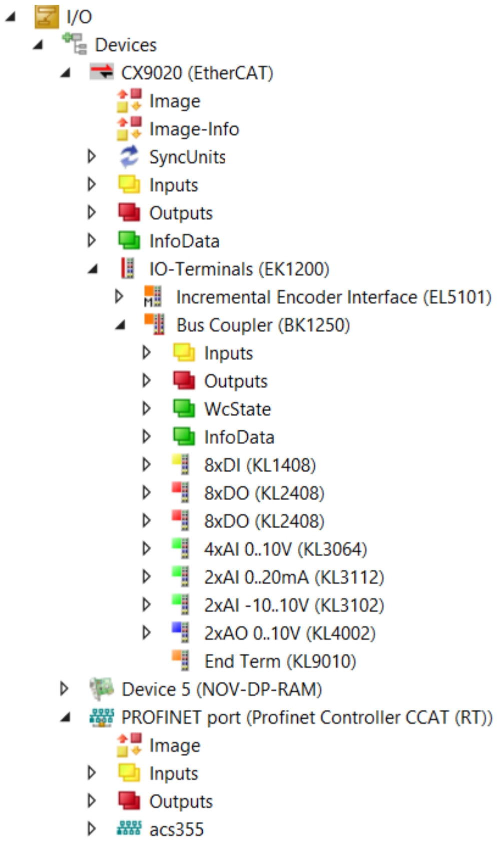
TwinCAT 3 XAE is PC based programming software for Beckhoff products. It integrates in the Microsoft Visual Studio 2013 environment in order to provide the user with an expandable and future-proof environment.

5.1.2 eXtended Automation Runtime

The TwinCAT 3 Runtime is an environment where the TwinCAT modules can be loaded, executed and administrated. The CX9020 controller in the system has this runtime installed, so this controller is able to run the program and configure all the IO-terminals that are connected to it.

5.2 Device configuration

In order for the controller to be able exchange data with the IO-terminals, a device configuration has to be generated (Figure 44). This device configuration describes all the hardware that is connected to the controller, including the build-in memory of the controller. Also, all the terminals can be configured here and variables can be connected to physical inputs and outputs on the terminals.



```
I/O
  Devices
    CX9020 (EtherCAT)
      Image
      Image-Info
      SyncUnits
      Inputs
      Outputs
      InfoData
      IO-Terminals (EK1200)
        Incremental Encoder Interface (EL5101)
        Bus Coupler (BK1250)
          Inputs
          Outputs
          WcState
          InfoData
          8xDI (KL1408)
          8xDO (KL2408)
          8xDO (KL2408)
          4xAI 0..10V (KL3064)
          2xAI 0..20mA (KL3112)
          2xAI -10..10V (KL3102)
          2xAO 0..10V (KL4002)
          End Term (KL9010)
    Device 5 (NOV-DP-RAM)
    PROFINET port (Profinet Controller CCAT (RT))
      Image
      Inputs
      Outputs
      acs355
```

Figure 44 Complete device configuration.

5.2.1 Beckhoff PLC terminals

The first device in Figure 44 is the CX9020 controller. It shows “inputs” and “outputs”, which are used for getting basic information about the controller. The first device under the CX9020 is the EL5101 terminal, which is connected to the Ethercat bus of the controller (EK1200).The next terminal is the bus coupler, which has all the K-bus terminals connected to it. Now, every terminal can be configured separately and the variables used in the program are assigned to the correct in- or outputs channels from the correct terminal.

5.2.2 PROFINET devices

Because the CX9020 is fitted with 2 PROFINET ports, the devices connected to PROFINET should also be added in the configuration. These devices are shown under “PROFINET port” (Figure 44), and can be found automatically by letting the controller scan the network. In this case the ABB ACS355 drive is connected to the network and added in the device configuration. Now all the PROFINET parameters can be configured. 5.4 will explain the configuration and handling of the PROFINET communication in TwinCAT more detailed.

5.3 Global Variables Lists GVL

A global variables list (GVL) is a list of variables that are accessible regardless of the part of the program. All of the variables that are used in the program are put in global variable lists in order to make the program as flexible as possible. In order to gain clear overview of the variables, 4 different GVL’s are used in this thesis (Figure 45):

- **I_O:** This list only contains variables that are connected to physical in- or outputs. This is done in order to get a clear overview of what goes out and what stays inside the PLC.
- **Flags:** All of the variables that only stay inside of the program and are not connected to any output signal (both physical and Ethernet) are located in this list. Also, when any binary (contactor) output value needs to be changed, the program changes values in this GVL. Only the last step in the program transfers the variables from this GVL to the corresponding flags in the I_O GVL. This way full control and overview over the outputs is constantly maintained.
- **PC_Variables**: The variables that are exchanged with the PC program through the Ethernet connection can be found here.
- **PROFINET:** Since the PROFINET communication requires a lot of variables, a separate GVL is made containing these.

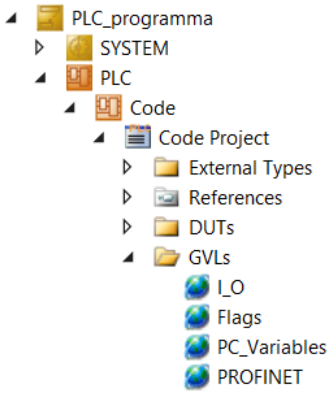


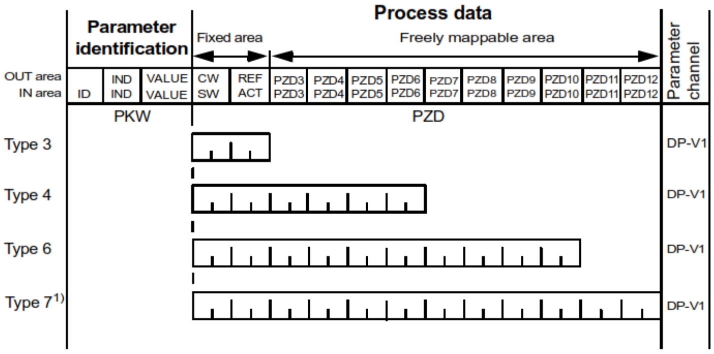
Figure 45 GVLs used in the PLC program

5.4 PROFINET communication in TwinCAT 3

5.4.1 PPO messages

As mentioned in chapter 3.6.2, we will use the PROFIdrive profile to communicate with the drive. In order to accomplish this, a “PPO” message type has to be chosen. PPO is a “Parameter/Process data object”, and determines how data will be exchanged between the drive and the controller.

Because PROFIdrive uses PROFINET RT, every cycle a PPO message will be sent from the drive to the controller and vice versa. The speed at which this happens is determined by the chosen cycle time. A PPO message consists of two fixed data words, followed by a number of data words containing process data (PZD) (Figure 46). The amount of PZD words depend in the PPO type that has been chosen. In this thesis PPO type 4 will be used, meaning that two fixed data words and four PZD words will be exchanged between controller and drive.



OUT area – Data sent from master to slave (control data)
IN area – Data sent from slave to master (actual data)
Parameter identification:
ID – Parameter identification
IND – Index for arrays
VALUE – Parameter value (Max. 4 bytes)
PKW – Parameter ID/value
Process data:
CW – Control word
SW – Status word
REF – Reference
ACT – Actual value
PZD – Process data (application-specific)
DW – Data word

[1] Not supported by ACS355 drives

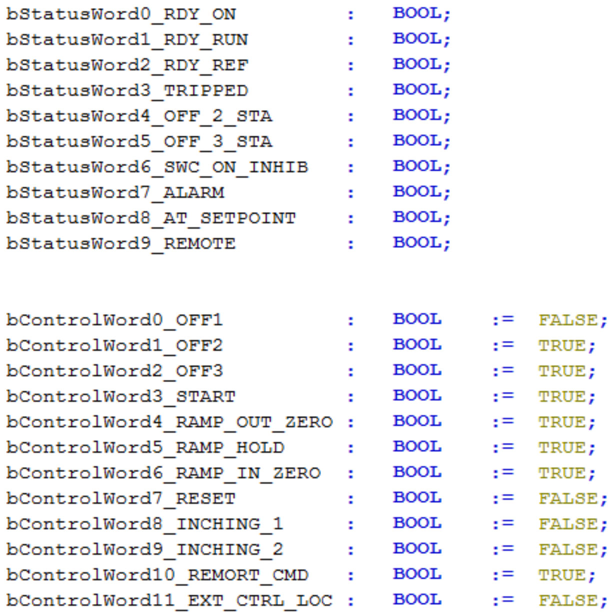
Figure 46 PPO type messages available for ACS355 drives

The two fixed data words in the message sent from the controller to the drive consist of a “Control Word” (CW) and a “Reference Value” (REF). The control word contains various bits that can control the status of the motor e.g. you can start and stop the motor by changing the corresponding bit in the CW. The REF is used to control the speed of the motor. The way this REF value is handled can be parametrised in the drive. The four PZD values contain data that has to be written to certain parameters. Which parameters are affected are chosen in the drive parameters.

In the message sent from the controller to the drive, the 2 fixed data words now contains a “Status Word” (STW) and an “Actual Value” (ACT). The status word contains bits that give information about the state of the drive e.g. if the drive is active or stopped. The ACT word contains the current rotational speed of the motor. This value is used in the program to show the motor speed on the screen. Next, the four PZD values can contain information about the current speed of the motor, the current flowing through the motor, drive temperature… . The information that is transmitted through these words can be parametrised.

5.4.2 Handling the status word and the control word.

As mentioned in 5.4.1, the status word and control word consist of separate bits with each its own function. Inside of the PROFINET GVL, the variables for the status word and the control word are located, as well as their separate bits (Figure 47). By doing this, every bit is easily accessible, making it very easy to control the motor and get the status of the drive.



```
bStatusWord0_RDY_ON           :    BOOL;
bStatusWord1_RDY_RUN          :    BOOL;
bStatusWord2_RDY_REF          :    BOOL;
bStatusWord3_TRIPPED          :    BOOL;
bStatusWord4_OFF_2_STA        :    BOOL;
bStatusWord5_OFF_3_STA        :    BOOL;
bStatusWord6_SWC_ON_INHIB     :    BOOL;
bStatusWord7_ALARM            :    BOOL;
bStatusWord8_AT_SETPOINT      :    BOOL;
bStatusWord9_REMOTE           :    BOOL;


bControlWord0_OFF1            :    BOOL     :=   FALSE;
bControlWord1_OFF2            :    BOOL     :=   TRUE;
bControlWord2_OFF3            :    BOOL     :=   TRUE;
bControlWord3_START           :    BOOL     :=   TRUE;
bControlWord4_RAMP_OUT_ZERO   :    BOOL     :=   TRUE;
bControlWord5_RAMP_HOLD       :    BOOL     :=   TRUE;
bControlWord6_RAMP_IN_ZERO    :    BOOL     :=   TRUE;
bControlWord7_RESET           :    BOOL     :=   FALSE;
bControlWord8_INCHING_1       :    BOOL     :=   FALSE;
bControlWord9_INCHING_2       :    BOOL     :=   FALSE;
bControlWord10_REMORT_CMD     :    BOOL     :=   TRUE;
bControlWord11_EXT_CTRL_LOC   :    BOOL     :=   FALSE;
```

Figure 47 The separate bits in the status word and the control word.

5.5 Program layout

5.5.1 Main layout

The main layout of the PLC program is shown in Figure 48. The PC sends information about the selected starting mode to the PLC. A matching function block is loaded depending on the selected mode. Next when the user gives the command to start the measurement, the datalogger is started to measure all necessary data, and the motor starts according to the selected mode by controlling the needed contactors. After the measurement is done, the datalogger sends an array back to the PC containing all the measured data during the measurement. When manual mode is selected, full drive control is possible by the user, and the PC will constantly receive updates about the input values of the PLC. Another key feature is the constant connection monitoring. The PLC constantly gets polling signals from the PC, and when these disappear (e.g. cable unplugged) the E-stop is triggered. When this happens all of the outputs are overridden in such a way the machine goes to a safe state. When the E-stop is not active, all outputs are just normally connected.

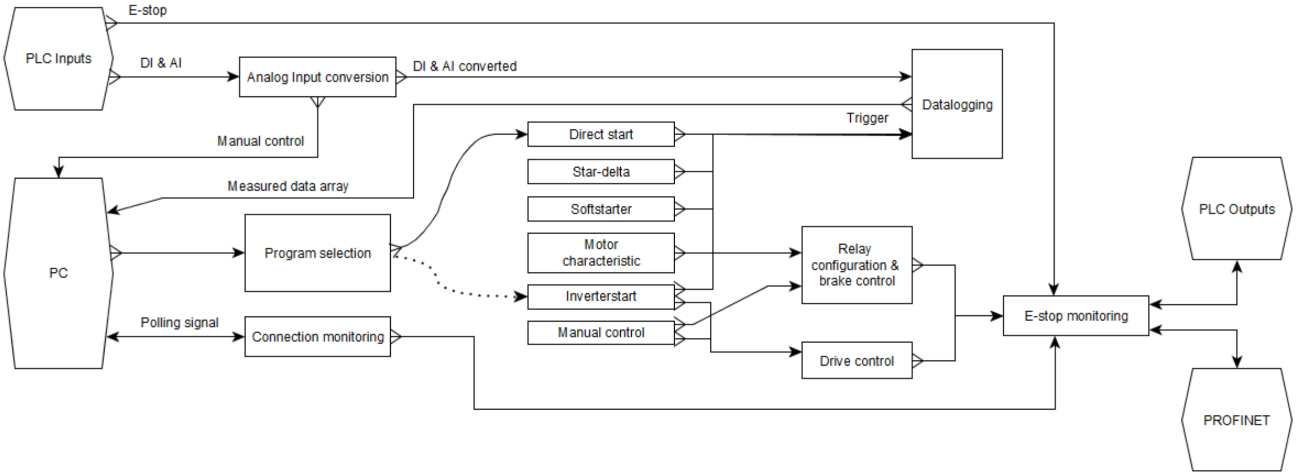


Figure 48 PLC program layout

5.5.2 Input and output handling

All the analogue inputs get converted to metric values before they get used in the program. This is done by running a conversion function block every time after the inputs are read and before the main program gets activated (Figure 49). All these converted values are then stored in the "PC_Variables" GVL, since all of these values are read by the PC. The conversion values are as follows:

- **Brake temperature:**
    - Raw input values: 0 – 65535
    - Sensor range: 0°C – 300°C
    - Conversion: 1 bit = 0.004578°C
- **Measured speed:**
    - Raw input values: 1 increment = 0.01 Hz
    - Slits / rotation : 5000
    - Conversion: 1 bit = 1/8333.33 rpm
- **Measured current:**
    - Raw input values: 0 – 65535
    - Calibration: 1.2 A = 4245 raw input value
    - Conversion: 1 bit = 0.00028267 A
- **Brake power**
- **Current (inverter)**
    - Raw input values: 1 increment = 0.01A
    - Conversion: 1bit = 0.01 A

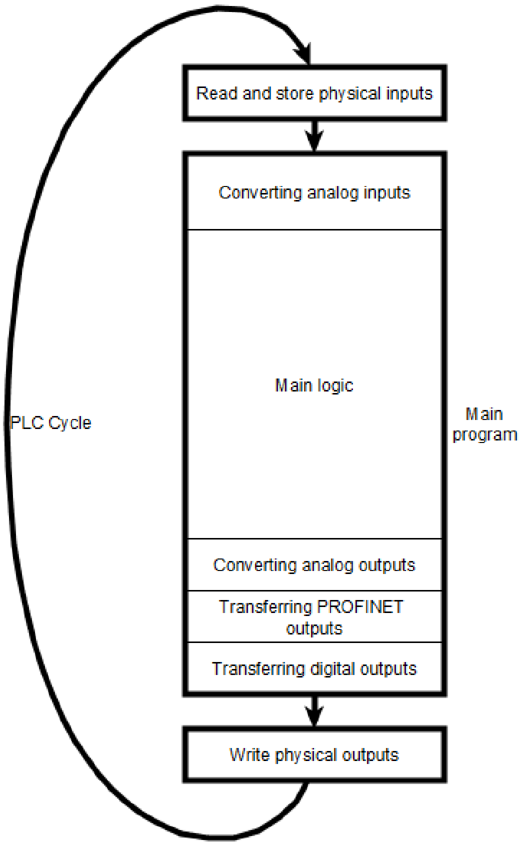

Read and store physical inputs
Converting analog inputs
Main logic
Converting analog outputs
Transferring PROFINET outputs
Transferring digital outputs
Write physical outputs
PLC Cycle
Main program

Figure 49 I/O handling in the PLC program

All of the variables that are linked to the physical outputs, are stored in the "I_O" GVL. However, these variables do not get used during the main program. For each output in the "I_O" GVL, there is an identical variable in the "flags" GVL. The variables in this GVL are not linked with physical outputs, though these are accessed during the main program. Only in the last step of the main program, the values from the "flags" variables are transferred to the corresponding values of the "I_O" GVL, with extra safe logic added. This step is shown in Figure 49 at the "transferring digital outputs" step. The reason for this is to be able to have full control over the outputs, and to be able to force them to the safe state at any given moment. Figure 50 shows an example used in the program. The variables "boRotateCW_K31" and "boRotateCCW_K32" are read from the "flags" GVL. Because of the electrical layout, contactors K31 and K32 should never be active at the same time since this would cause a major short-circuit. In order to prevent this, extra safety measures are added in transferring the "flag" value to the "I_O" value. By adding a negating contact of the other contactor, one contact can never be active at the same time as the other. After this the emergency stop variable is also added, so the emergency stop always has the highest priority. This way, when the emergency stop is pressed, "bNotEmergencyStop" will block the circuit (Figure 50) so bo_K31 and bo_K32 can never get a Boolean "1". By using this kind of structure

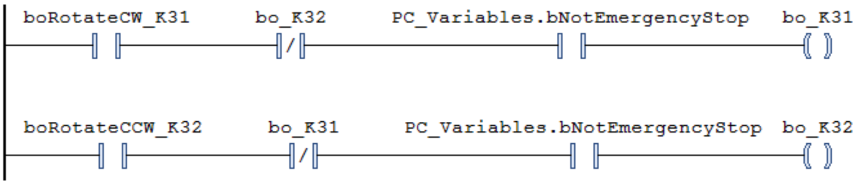

boRotateCW_K31
bo_K32
PC_Variables.bNotEmergencyStop
bo_K31
boRotateCCW_K32
bo_K31
PC_Variables.bNotEmergencyStop
bo_K32

Figure 50 Link between the flag variable and the linked I_O variable.

5.6 Connection- and E-stop monitoring

5.6.1 E-stop monitoring

The main goal of adding the PLC was to improve the safety for when the connection between the PC and motor was lost. An emergency stop bit has been added in the program. When this bit is Boolean 0 the emergency stop is active. The emergency stop is activated when at least one of the next events occur:

- The emergency stop is pressed
- The current measured by the sensor or the drive exceeds 6A during five seconds.
- The connection between the PC and the PLC is lost
- The brake temperature is over 45°C

When the emergency stop is active and none of the above event occur, it takes 3 seconds for the emergency stop to reset. This is done to prevent the emergency stop to activate too frequently.

5.6.2 Connection monitoring

In order to be able to detect if the connection is lost, every 50 ms the PC writes a Boolean 1 in variable “bPoll”. This bit will reset an off-delay timer so the variable “bConnectionActive” will stay active. The very next step in the code is that “bPoll” is reset. This way, when the connection is lost, the timer will not be reset, so “bConnectionActive” will be reset. Figure 51 shows this process.

When the connection is lost, the power to the drive and main 3-phase power supply will cut off after 5 minutes. This delay has been added so the drive will not lose power every time the PC crashes or the Ethernet cable is accidently removed for a few seconds.

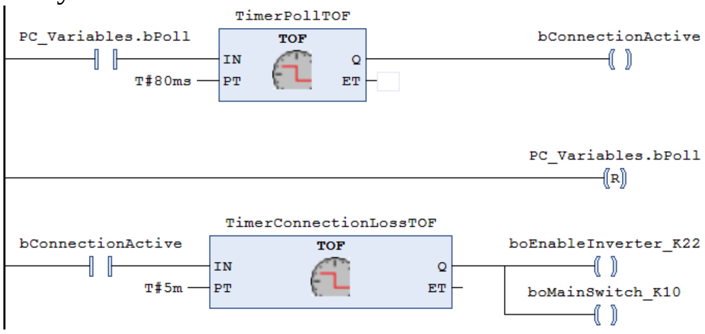


Figure 51 Connection monitoring

5.7 Datalogger

When a measurement is started, all the measured data has to be stored. The datalogger saves all the measured data in an array (Table 1). This array has one column for each variable to be measured, and an array containing the timestamp of the sample. Every sample, an extra row is made and data is stored in this row. In order to maintain a constant sample rate, the datalogger runs completely separate from the main program, by using a separate PLC task. This task is forced to run faster and given a higher priority than the main program. A timer is added in the datalogger, giving pulses on at constant rate. Every time a pulse is generated all the analog inputs are read, and the values are stored in the array. The speed of these pulses (sample rate) can be modified.

	Speed (rpm)	**Current (A)**	**Inverter current (A)**	**Brake temperature (°C)**	**Brake power (N)**	**Timestamp (ms)**
Sample 1						
Sample 2						
....						
Sample n						

Table 1 Array used for storing the measured data

5.8 Program selection

The selection of the starting mode is done by a separate function block ("Program selector"). A number is sent by the PC, representing a selected starting mode (e.g. 1 = direct start). Corresponding to this number, the function block for the selected starting mode is loaded.

Every function block for a starting mode has an input for starting the measurement, and returns whether or not the measurement is busy. If any of the modes is doing the measurement, none of the other modes can be started. Also, the emergency stop is added in the logic, so that when the emergency stop is active, the motor can never be started. The motor can only be started when all the other modes signal that they are completed, the emergency stop is not active and the start button is clicked on the PC.

5.8.1 Direct start

The direct start mode is executed by using a sequential flow chart function block. When this function block is loaded, the program waits in the "Init" step. Here all the contactors are reset and the function block signals that it is completed and ready to start. Once the start signal is given (5.6), the contactors are prepared, which means the motor gets connected in delta, the direction gets chosen and only one contactor is left to start the motor. When this is done, the datalogger is started and the final contactor is closed starting the motor. After 3 seconds the contactor is opened again, and the motor is stopped by using the brake. When the motor is stopped all the contactors are opened again and the "Init" step is activated again.

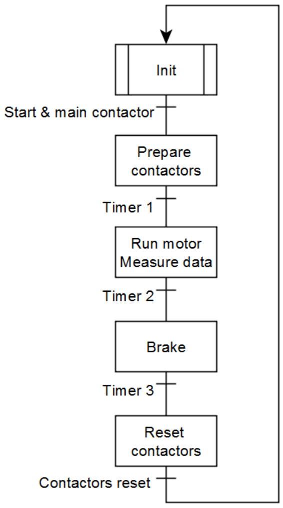


Figure 52 SFC used for the direct starting mode

5.8.2 Star-delta start

Just as for the direct start, the star-delta mode also uses the sequential flow chart. It works in the same way as the direct start, only now the motor is first placed in star configuration and is then started. When the motor is at nominal speed the star configuration is opened and the motor is placed in delta configuration. A short delay is placed during this transfer in order to prevent short circuits. When the measurement is done, the motor is uncoupled and braked.

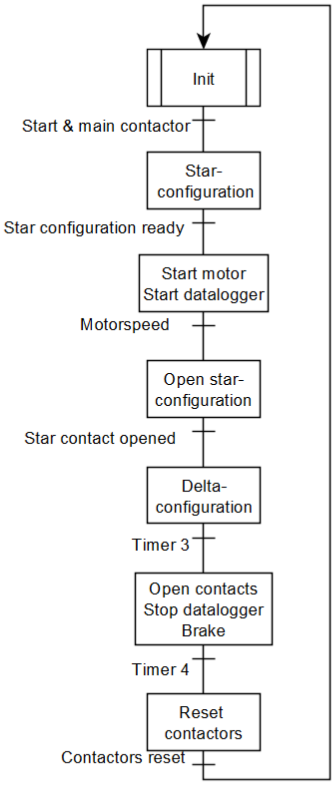


Figure 53 SFC for the star-delta starting mode

5.8.3 Soft starter start

This starting mode uses the same SFC as the direct start (Figure 52). It takes the exact same steps, only now the motor is not connected to directly to the power, but to the soft starter.

5.8.4 Frequency converter start

The last starting mode is the start by using the frequency converter. When this mode is selected the drive is connected to the motor. Just like the previous starting modes this also uses an SFC. The motor is started by using a function block made for controlling the motor (5.8.6). This function block is activated in the SFC, and given start – and stop commands. Note that the brake is not used in this SFC, as the drive can slow down the motor itself.

5.8.5 Motor characteristic

This mode will be used to measure the torque-slip characteristic of the motor (2.2.3). In order to accomplish this, the motor is started in delta configuration with the soft starter. When the motor is running, the soft starter is bridged so the motor is now directly coupled to the net. From this moment the datalogger is started, and the brake applies full force to the motor. Because of this the motor will gradually slow down until it is completely stopped. When the motor stopped, the motor is disconnected, the brake releases and the datalogger stops.

5.8.6 Drive control (manual mode)

Because the drive uses PROFINET to communicate, a separate function block (Figure 54) is made. Inside, various tasks are executed:

- The emergency stop is connected to bit 2 of the control word. When this bit is reset, the drive will make the motor stop very fast (0.5s).
- All the contactors needed for the drive to operated are activated
- The start and stop are connected to bit 0 of the control word. The code that handles the stop is executed after the code for the start. This way the stop always has priority to the start.
- In manual mode, this block allows the speed and the direction of the motor to be changed.
- The “active” output is active when bit 1 of the status word is Boolean 1. This bit tells whether or not the motor is active.
- The “complete” output is active when bit 1 of the status word is 0.

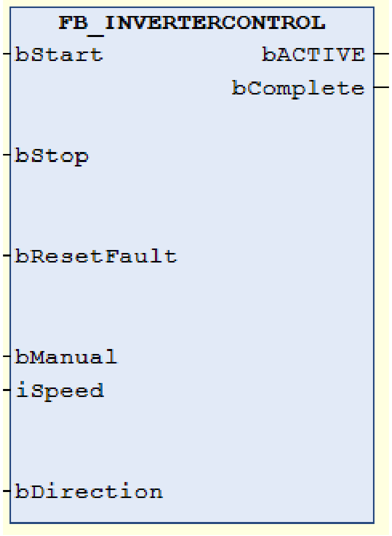


Figure 54 Function block to control the inverter.

6 PC PROGRAM

6.1 Existing

In 2012 Vincent Peerlinck had already written a major part of the program, including the “simoclasses”[14]. He also prepared an interface for the measurements, but it was lacking communication and still needed some improvement.

6.2 PLC communication

To be able to send the data from the PLC to the PC, some way of communication has to be provided. Since the new system has to function for another 20 years, this has to be a uniform and simple. Because every PC now has at least standard 1 Ethernet port, and the CX9020 controller has it as well, the communication will happen through an Ethernet connection. This way the connection remains very simple by using just 1 Ethernet cable.

6.2.1 TwinCAT ADS

To allow the user program (C#) to communicate with the TwinCAT program in the CX9020 controller, the TwinCAT ADS interface must be used. The Automation Device Specification (ADS) is a device- and fieldbus-independent interface [16], which in this case will be used with the TCP/IP protocol.

The .NET component of ADS is used in C# (figure 51). It’s done by importing the TwinCAT.Ads namespace into visual studio. This allows communication with the TwinCAT message router that has to be installed on the PC. This router is necessary for translating the ADS messages and sending it to the right devices on the network through the TCP/IP (Ethernet) port of the PC. On the other side of the Ethernet cable is the CX9020 controller, which is also fitted with the message router so the messages can access the PLC program in the TwinCAT 3 runtime.

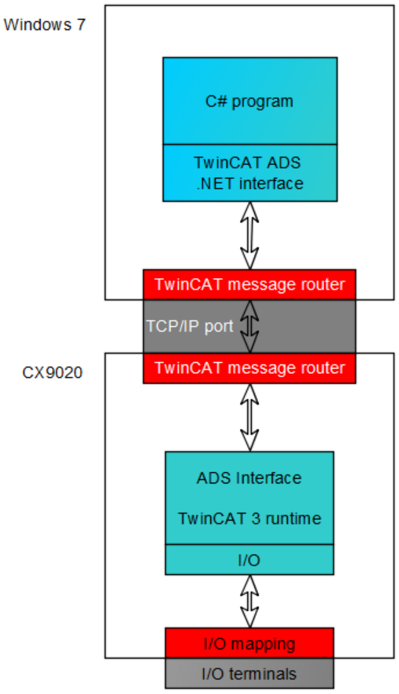


Figure 55 ADS structure

TwinCAT ADS uses the client/server-principle on TCP, with the PC as the client and the CX9020 controller as the server. This means that the PC always accesses the controller and not vice versa. The structure of the messages sent by the ADS interface is shown in Figure 52. This ADS data packet consists of various parts:

- **AMS Net ID**: This is the unique device address in the ADS network. It is an address apart from the IP-address. The ADS routers will create a link between these two addresses
- **ADS Port:** The port that will be used for the data to pass through. This is standard port 851.
- **Index group:** This contains the location of the group of variables (e.g. GVL) that has the desired variables to be accessed.
- **Index offset:** The location of the variable in the group.
- **ADS Data:** Contains the type of service (read, write, connect, ..) and the data to be transmitted between the devices.

All this data is preceded by the standard TCP header and the IP-header. Lastly there are the acknowledge bits (SA, DA) and the CRC code ("cyclic redundancy check").

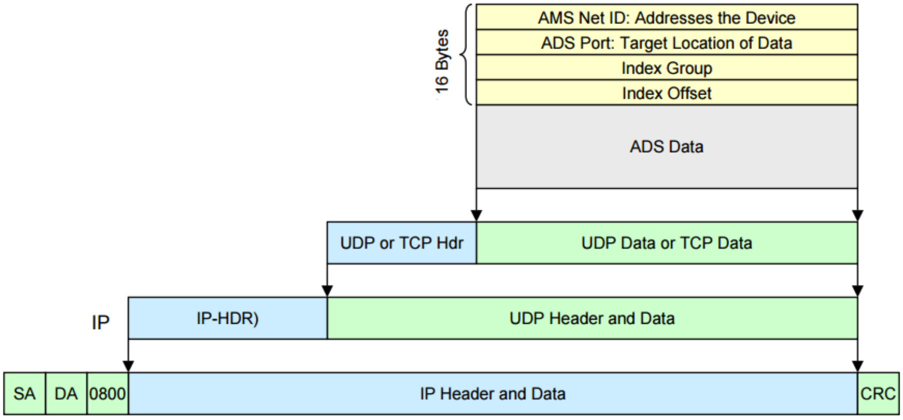
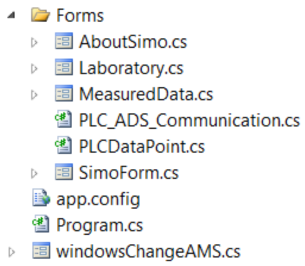


Figure 56   ADS message

6.2.2 Class structure

Figure 57 shows the class structure used in the PC program. A separate class is made that handles the communication with the PLC ("PLC_ADS_Communication.cs"). This is done so that higher classes (forms) can communicate easily with the PLC by simply calling methods from this class. The class "MeasuredData.cs" is the class for the user form containing all the data, charts and controls for the PLC.



```
◢ Forms
    ▷ AboutSimo.cs
    ▷ Laboratory.cs
    ▷ MeasuredData.cs
      PLC_ADS_Communication.cs
      PLCDataPoint.cs
    ▷ SimoForm.cs
  app.config
  Program.cs
▷ windowsChangeAMS.cs
```

Figure 57 Class structure in the C# program

6.2.3 Connecting to the PLC

The first thing to happen is to connect the PC to the PLC. This is done by the code shown below, which is run by the PLC_ADS_Communication class. The PC is assigned as an ADS client, and connect to the device with a specified address and port. This address and port can be changed in the program, and is stored so it keeps this the next time the program starts.

```
public void connectAdsClient()
        {
            adsClient = new TcAdsClient();
            try
            {
                AmsNetId address = new AmsNetId(amsNetID);
                adsClient.Connect(address, port);
            }
            catch (Exception e)
            {}
        }
```

6.2.4 Variable handles

To access variables in the PLC, the C# program needs to use "variable handles". These variable handles are variables that contain information about the location of the variables handle in the PLC. As an example the declaration of the variable for the program mode in the PLC is shown below. It is important to note that the variable "hModeProgram" does not directly contain the data of the PLC variables. It is only used for accessing the variable.

```
private int hModeProgram;
hModeProgram=adsClient.CreateVariableHandle("PC_Variables.
intModeProgram");
```

Next is shown how the value in the PLC is changed by using this variable handle. The parameter “mode” has a number representing the number of the selected starting mode.

```
public void ChangeProgramMode(Int16 mode)
        {
            adsClient.WriteAny(hModeProgram, mode);
        }
```

Reading a variables is done in almost the same way. This is an example where the speed of the motor is read from the PLC. The variable “hSpeed” is the variable handle for the variable in the PLC.

```
public UInt16 getSpeed()
        {
           Return  Convert.ToUInt16(adsClient.ReadAny(hSpeed
           ,  typeof(double)));
        }
```

6.2.5 PLC-connection monitoring

As explained in 5.6.2, the connection is monitored by constantly setting a bit in the PLC. This is done by running a separate thread in the class of the measured data form. This thread is started when the program opens, and constantly writes the necessary bit in the PLC. It also reads information about the emergency stop and PROFINET connection state. If it can’t read this info, it means that the connection between the PC and the PLC is lost. Whenever this happens, an error message is shown and the complete user interface gets disabled. Now the thread will try to connect to the PC again every 3 seconds. Also the bottom status bar will show that the PLC is disconnected.

6.2.6 Retrieving measured data

When the user starts a measurement, the program freezes while the motor is running to prevent the user from clicking wrong things during the measurement. When the PLC finishes the measurement, the PC gets notified and reads the array containing all the measured data (chapter 5.7). This array is sent through ADS interface as a data stream. The PLC communication class reads this class and restores it back into an array of a “struct”. This struct is created by making a new variable of the class “PLCDataPoint” shown below

```
public class PLCDataPoint
    {
        public double nSpeed;
        public double nCurrent;
        public double nCurrentInverter;
        public double nBrakeTemp;
        public double nBrakePower;
        public double nAcceleration;
        public double dSlip;
        public double nTimeStamp;
    }
```

This class contains a variable for each measured parameter by the PLC, in addition to the extra variables for calculation the slip and acceleration. Before reading the stream, the length of the stream has to be provided. This is calculated by following equation.

$$Length = N * 8 * 5 \tag{9}$$

Here, N = the amount of measurements taken by the PLC. Each measurement has five values, with each value being an 8-byte double. Multiplying all these numbers results in the expected length of the stream in bytes. Next step is to read the stream. The stream starts by reading the first 8 bytes (1st value of the 1st measurement) and writing the resulting value in corresponding variable of the PLCDataPoint. After these 8 bytes, the next 8 bytes are read and again written in the next variable. After 5x8 bytes, the variable of the PLCDataPoint contains all the data from the 1st measurement and is stored as the first element in an array of PLCDataPoints.

The acceleration is calculated by subtracting the current measured speed from the speed of the previous sample. To reduce the noise of this calculation, it averaged by adding four previous acceleration to each other and dividing it by four (equation 10). The same noise reduction is done for the measured braking power.

$$\alpha_{av} = \frac{\alpha_s + \alpha_{s-1} + \alpha_{s-2} + \alpha_{s-3}}{4} \tag{10}$$

At the same time of reading the value of the speed, the slip is calculated by equation 11)

$$s = \frac{n_{synchr} - n_{meas}}{n_{synchr}} \tag{11}$$

After all the measurements are read and stored, the array of PLCDataPoints is returned to the “measured data” form, where it will be used to put in charts.

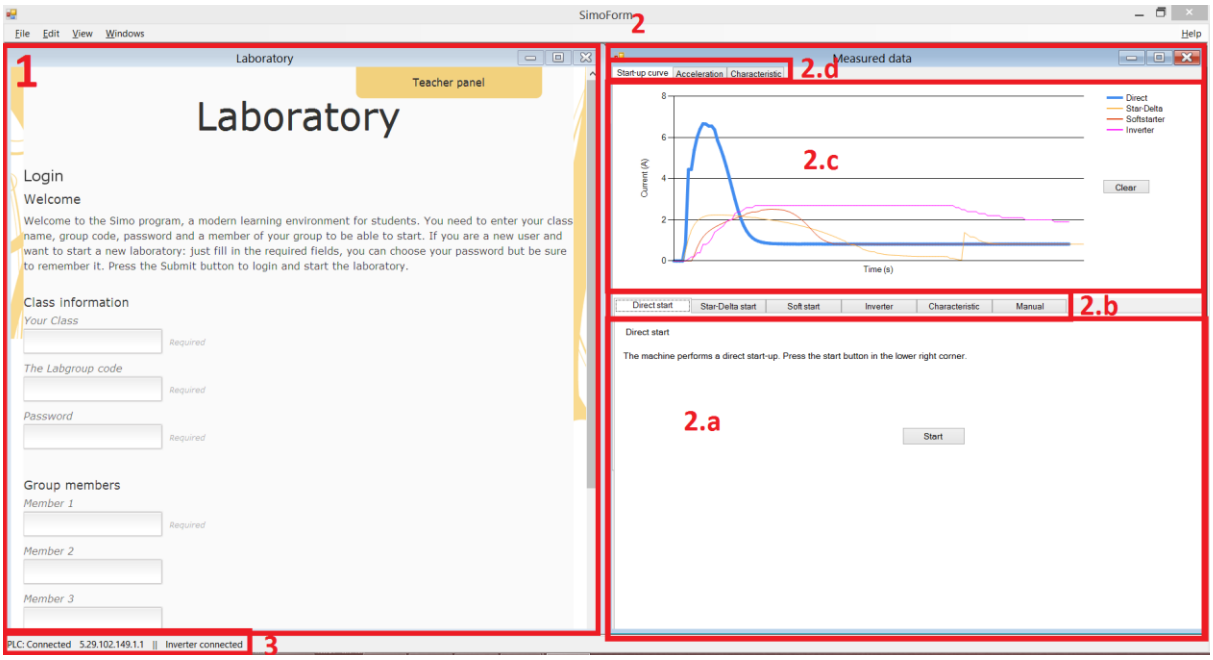
6.2.7 Updating gauges

When the user is in manual control mode, a real time feedback from variables to the user is required. To achieve this an extra thread is used, which constantly reads the values from the PLC and runs separately so the main program doesn’t freeze.

6.3 Layout

The main layout of the program is shown in Figure 58. It consists of different parts:

- **1**: The window for the laboratory. This has not been part of this thesis. For more information please refer to the thesis of Vincent Peerlinck [14]
- **2**: This window is used for all the measurements and control of the motor.
    - **a:** This part contains the control for the selected starting mode. Depending on which starting mode settings can be made here.
    - **b:** Here the student can choose which starting mode has to be used.
    - **c:** After measurements are made, they are shown in this chart area.
    - **d:** Because various parameters are measure, the user can choose which chart needs to be shown here.
- **3**: Here the status of the PLC, inverter and brake are shown. When an error occurs it is shown here which kind of error this is.



Laboratory

Login

Welcome

Welcome to the Simo program, a modern learning environment for students. You need to enter your class name, group code, password and a member of your group to be able to start. If you are a new user and want to start a new laboratory: just fill in the required fields, you can choose your password but be sure to remember it. Press the Submit button to login and start the laboratory.

Class information

Your Class — Required

The Labgroup code — Required

Password — Required

Group members

Member 1 — Required

Member 2

Member 3

Direct start

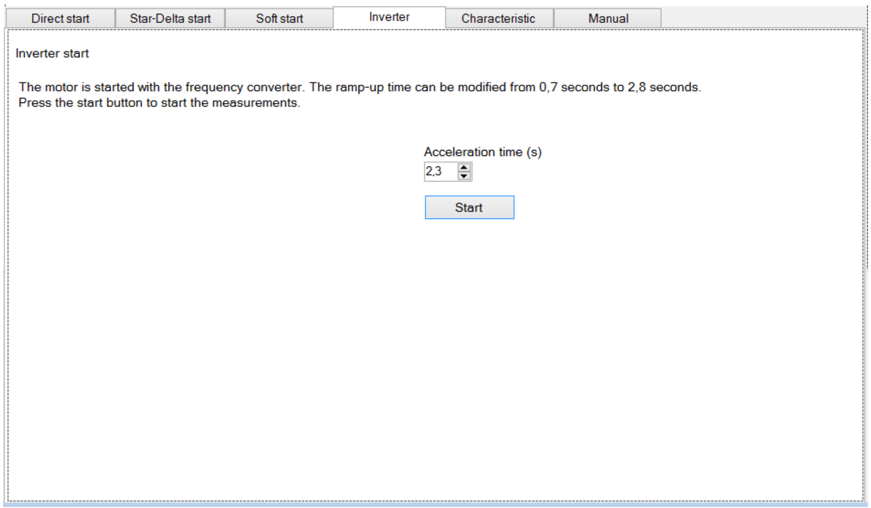
The machine performs a direct start-up. Press the start button in the lower right corner.

PLC: Connected 5.29.102.149.1.1 || Inverter connected

Figure 58 Layout of the main program

6.3.1 Starting modes

The 4 different starting modes (direct start, start-delta start, soft starter, and inverter) and the characteristic mode all have the same layout. By clicking on the corresponding tab page, a starting mode is selected. To start the measurement the user just has to press the “start” button. Only for the inverter start an extra parameter is added (Figure 59). The user can modify the ramp-up time from the motor ranging from 0.7 seconds to 2.8 seconds.



Inverter start

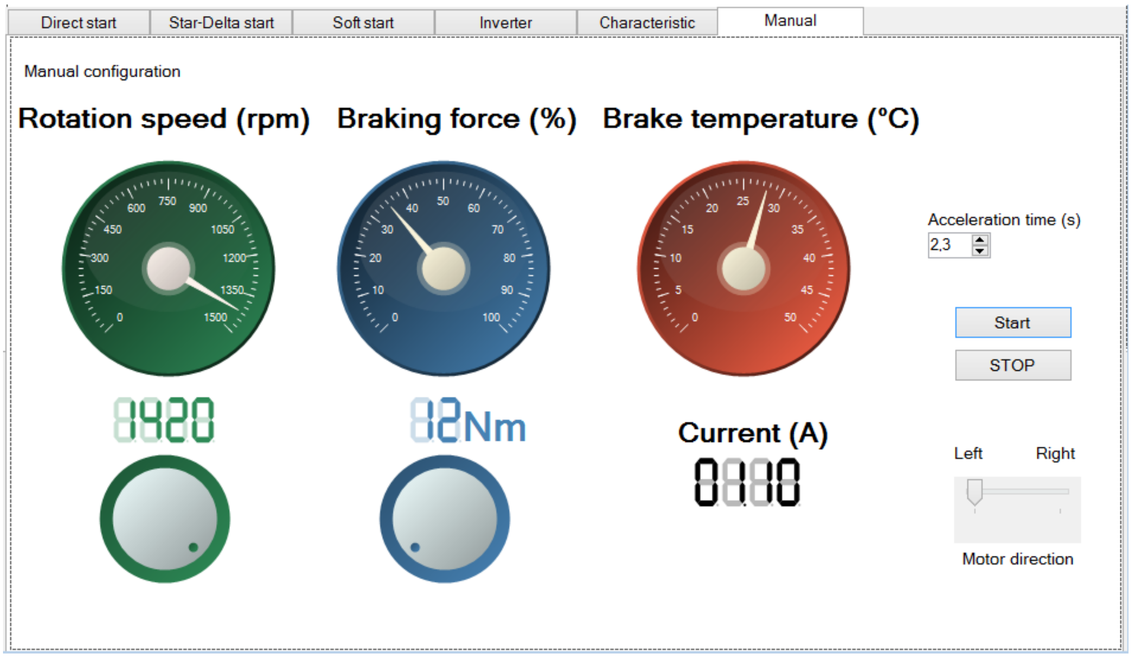
The motor is started with the frequency converter. The ramp-up time can be modified from 0,7 seconds to 2,8 seconds.
Press the start button to start the measurements.

Acceleration time (s): 2,3

Figure 59 Inverter start tab page

6.3.2 Manual control

If the user selects the manual control, he gets control over various parameters in the setup. Every parameter has received its own colour in order to maintain a better overview (Figure 60). First of all the user can control the rotation speed (green) of the motor by rotating the green knob. The user then gets feedback of the motor speed by both a digital and an analogue gauge. Secondly, the user also can change the braking force of the brake (blue) by rotating the blue knob. The braking force is shown in Nm and percentage of the maximum force. Furthermore, the brake temperature is shown (red). When this value exceeds 50°C, the brake will be disabled. Next, the user also get feedback from the current that is flowing through the engine, measured by the drive. Lastly the rotation direction can be changed and the ramp-up time can be modified.



Acceleration time (s) 2,3

Figure 60 Manual control screen

6.3.3 Charts

In order to be able to show the user the data that has been measured, charts are used. Three different charts are used: the “start-up chart”, “Acceleration chart” and the “characteristic chart”.

6.3.4 Start-up chart

The start-up chart shows the user the progression of the start-up current. It will draw current of all 4 different starting types, with each its own unique colour. These colours are the same in the other charts as well. Figure 61 shows this chart, with the current of the direct start highlighted. The program will highlight the line of the starting mode that is currently selected. When the user goes to the “star-delta” starting mode, the yellow line of start-delta will be highlighted etc. . Also, the user has the option to clear all the starting modes so the chart will be empty.

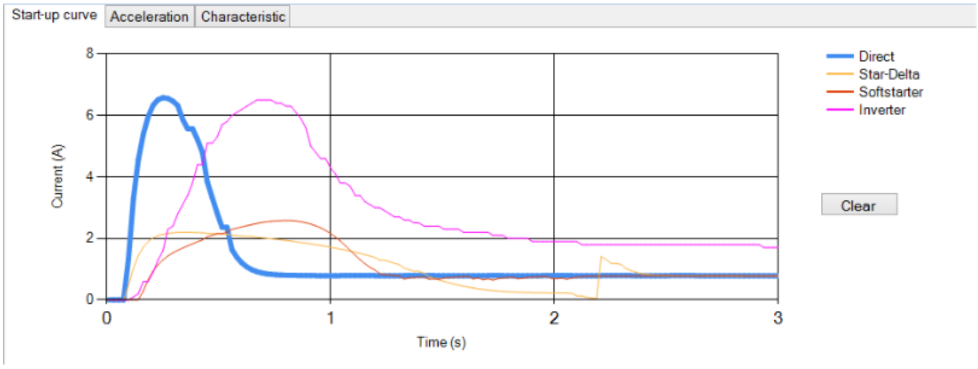


Figure 61 Start-up chart

6.3.5 Acceleration chart

This chart shows both the acceleration and the speed of the motor. The same colours as in the start-up chart are used in order to make it easier. The speed is always darker then the acceleration, but still has the same colour. Depending on the selected starting mode, the corresponding curves are highlighted. Furthermore, the user can also choose to only show the acceleration or only the speed chart. Figure 62 shows the chart for the inverter start mode.

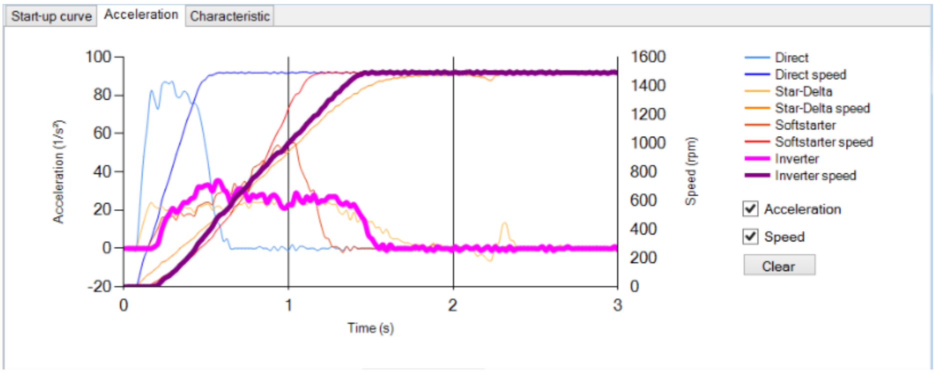


Figure 62 Acceleration chart

6.3.6 Characteristics chart

When the user starts the motor in motor characteristics mode, the torque-speed characteristic is measured and shown (Figure 63).

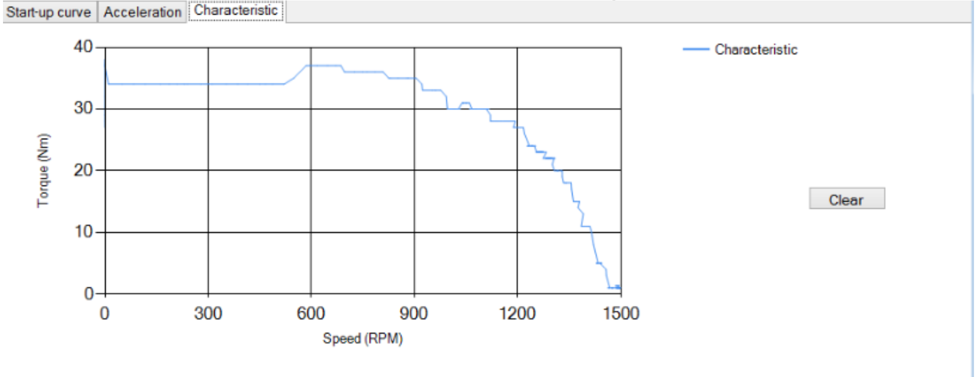
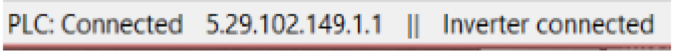


Figure 63 Characteristic chart

6.3.7 Status bar

The status bar is located on the bottom of the screen, and gives a small overview of the status of the setup (Figure 64). It shows whether or not the PLC is connected, and the current AMS Net ID that is being used. It also shows if the inverter is disconnecting, connecting and connected. Lastly it also gives a sign when the brake is overheated.



PLC: Connected 5.29.102.149.1.1 || Inverter connected

Figure 64 Status bar

7 CONCLUSION

It can be concluded that the program works and is ready for students to use. The old computer and program are replaced by a brand new PC and program. This program has now been improved and finished as well. The hardware located on the system has also been upgraded so the setup can go on for another 20 years.

The new setup is equipped with a brand new drive, and Beckhoff CX9020 PLC. This PLC is fitted with all the necessary I/O terminals so the old I/O cards of the old system are removed. Next the old ACS600 drive has been removed and replaced by a new ABB ACS 355 drive. These new components made it possible to upgrade the communication system. Now the communication between the PC and PLC is realized by using one simple Ethernet connection by the use of the TwinCAT ADS interface. Furthermore, the communication between the PLC and the drive now also happens through an Ethernet connection by using PROFINET. Resulting from all of this is a simpler, yet safer system. Thanks to the new controller, the system can always come to a safe stop when the PC crashes or any of the connections is lost. This has resulted in a more student proof setup.

The software on the PC is now compatible with Windows 7, since it is a completely new program. A big part of the program has already been made previously, but has now been completed by improving the layout and adding communication with the PLC. This communication has been realized by using the TwinCAT ADS interface.

With the completed setup the students are now able to get a better vision of the difference between the starting modes of an ASM. By adding new technology in the hardware, students can also get a first touch of PROFINET communication and PLC’s.

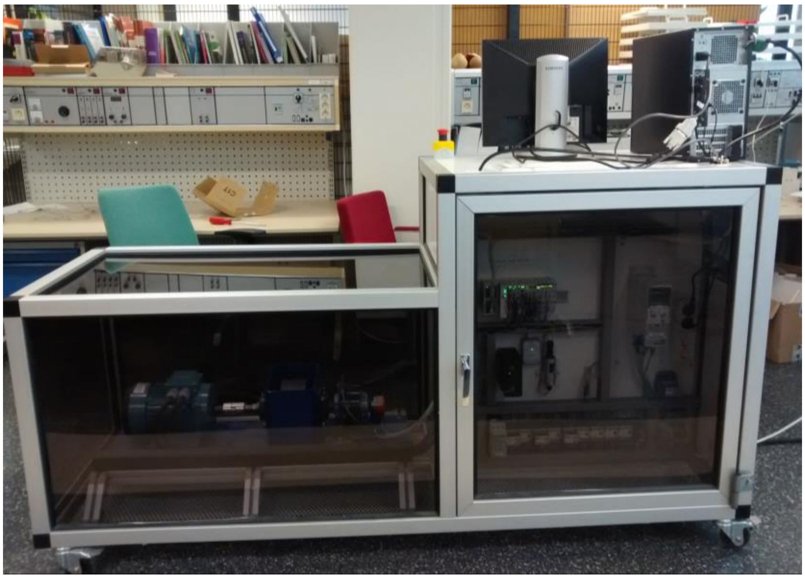


Figure 65 Completed motor test unit

SOURCES

[1] “Embedded PC for building automation,” 2013. [Online]. Available: http://www.instrumentation.co.za/45356n.

[2] MpowerUK, “Electric Drives - AC Motors (Description and Applications).” [Online]. Available: http://www.mpoweruk.com/motorsac.htm.

[3] G. Vandensande, “Elektrische machines.” .

[4] Meggar, “Squirrel cage rotor.” [Online]. Available: http://en.wikipedia.org/wiki/Induction_motor#/media/File:Squirrel_cage.jpg.

[5] J. Rees, M. Kjellberg, and S. Kling, “Softstarter Handbook,” p. 92, 2010.

[6] R. Naukkarinen, “Drive Module Testing Method Comparison for Drive Service Workshop,” 2012.

[7] B. Akin and N. Garg, “Scalar (V / f) Control of 3-Phase Induction Motors,” 2013.

[8] OGURA Industrial corp., “Electromagnetic clutches and brakes.” [Online]. Available: http://www.ogura-clutch.com/products/industrial/howtheywork/electromagnetic-particle-brake.html.

[9] Mandee Liberty & Vikram Phadke, “Encoders Basic Training.” .

[10] Pepperl & Fuchs, “Incremental Encoder Output Signal Overview,” pp. 1–3.

[11] E. a G. L, “Technical bulletin,” vol. 1, no. 909, p. 2, 2003.

[12] PROFIBUS & PROFINET International (PI), “PROFINET The Leading Industrial Ethernet Standard.” [Online]. Available: http://www.profibus.com/technology/profinet/.

[13] Osmo vuotila, “Physical and Electrical Planning of 3-phase AC-Engine Measuring Device,” HAMK, 2000.

[14] V. Peerlinck, “Upgrade computer program for engine,” HAMK University Of Applied Sciences, 2012.

[15] Beckhoff, “CX 9020 Manual.” pp. 1–48, 2015.

[16] Beckhoff, “TwinCAT ADS.” [Online]. Available: infosys.beckhoff.com/english.php?content=../content/1033/tcadsdeviceplc/html/tcadsdeviceplc_intro.htm&id=.

Appendix 1

HARDWARE LIST

Devicename	**Manufacturer**	**Type**	**Properties**
Motor	ABB	M2AA 90 L ; 3GAA 092 002	3-phase , 4 poles, 1,5 kW
Encoder	Kübler	8.5820.1541.5000	5000 pulses/revolution
Brake	Lenze	04	Magnetic powderbrake, 40Nm
Brake amplifier	P.Poukkula Oy		0-10VDC to 0-24V DC 1A
PT-100 sensor			0-300°C
PT-100 converter	Datexel	DAT 201 3W-0-10V-PT100-0-300	0-300°C , 0-10VDC output
Current meter	Carlo Gavazzi	A 82-10-50	Maximum 50A AC
Frequency converter	ABB	ACS355-03E-05A6-4+k466(+acs-cp-A)	With FENA11 adapter and controlpanel
Power supply	Murrelekronik	MCS5-230/24	24VDC, 5A, regulated
Loadcell	Gefran	TC-K1C-F-S	0-100kg
Loadcell amplifier	Gefran	CIR-N-2-N-O	3mV/V
Relay	Moeller	DIL EM-01-G	3-phase, 24V DC signal
Relay	Schrack	MT321024	3 x switches, 24V DC signal
Relay socket	Schrack	MR78750	
PLC controller	Beckhoff	CX9020-0110-M930	+ 2 x RJ-45 PROFINET CONTROLLER
PLC Encoder interface	Beckhoff	EL5101	RS422 inputs
PLC Bus coupler	Beckhoff	BK1250	E-bus to K-bus
PLC DI	Beckhoff	KL1408	8xDI
PLC DO	Beckhoff	KL2408	8xDO
PLC AI	Beckhoff	KL3064	4 x AI, 0-10VDC
PLC AI	Beckhoff	KL3102	4 x AI, 0-20mA
PLC AI	Beckhoff	KL3112	2 x AI, -10-10V DC
PLC AO	Beckhoff	KL4002	4 x AO, 0-10VDC
PLC End terminal	Beckhoff	KL9010	

Appendix 2

PLC IO TABLE

Controller				
Beckhoff CX9020-0110-M930				
Type	**Address**	**Description**	**Device**	**Signal**
EN	Port 1	Ethernet	PC	
EN	Port 2	Ethernet		
PN	Port 1	PROFInet RT	Inverter	
PN	Port 2	PROFInet RT		

Encoder interface				
Beckhoff EL5101				
Type	**Address**	**Description**	**Device**	**Signal**
DI		A	Kübler	0-5VDC
DI		A/	Kübler	0-5VDC
DI		B	Kübler	0-5VDC
DI		B/	Kübler	0-5VDC
DI		C	Kübler	0-5VDC
DI		C/	Kübler	0-5VDC
Out		24V	Kübler	24VDC
Out		0V	Kübler	0VDC

Binary Inputs				
Beckhoff KL1408				
Type	**Address**	**Description**	**Device**	**Signal**
DI	1	Fuse	Fused terminal block	0/24 VDC
DI	2	Fuse	Fused terminal block	0/24 VDC
DI	3	Fuse	Fused terminal block	0/24 VDC
DI	4	-		
DI	5	-		
DI	6	-		
DI	7	-		
DI	8	E-stop	Button	0/24VDC

Binary Outputs				
Beckhoff KL2408				
Type	**Address**	**Description**	**Device**	**Signal**
DO	1	K21	DILEM-01-G	0/24 VDC
DO	2	K22	DILEM-01-G	0/24 VDC
DO	3	K23	DILEM-01-G	0/24 VDC
DO	4	K24	DILEM-01-G	0/24 VDC
DO	5	K25	DILEM-01-G	0/24 VDC
DO	6	K31	DILEM-01-G	0/24 VDC
DO	7	K32	DILEM-01-G	0/24 VDC
DO	8	K41	DILEM-01-G	0/24 VDC
DO	9	K42	DILEM-01-G	0/24 VDC
DO	10	K90	DILEM-01-G	0/24 VDC

DO	11	K10	DILEM-01-G	0/24 VDC
DO	12			
DO	13			
DO	14			
DO	15			
DO	16			

Analog Inputs				
Beckhoff KL3064				
Type	**Address**	**Description**	**Device**	**Signal**
AI	0	Brake temp.	PT-100	0-10 VDC
AI	2			0-10 VDC
AI	4	Brake power	Gefran	0-10 VDC
AI	6			0-10 VDC

Analog Inputs				
Beckhoff KL3112				
Type	**Address**	**Description**	**Device**	**Signal**
AI	8			
AI	10			
AI	12	Current	Carlo Gavazzi	0-20mA
AI	14			

Analog Inputs				
Beckhoff KL3102				
Type	**Address**	**Description**	**Device**	**Signal**
AI	16			
AI	18			
AI	20	Current	Carlo Gavazzi	-10-10V
AI	22			

Analog Outputs				
Beckhoff KL4002				
Type	**Address**	**Description**	**Device**	**Signal**
AO	0			
AO	2			0-10 VDC
AO	4	Brake	Amplifier	0-10 VDC
AO	6			0-10 VDC

Appendix 3

ACS355 MODIFIED PARAMETERS

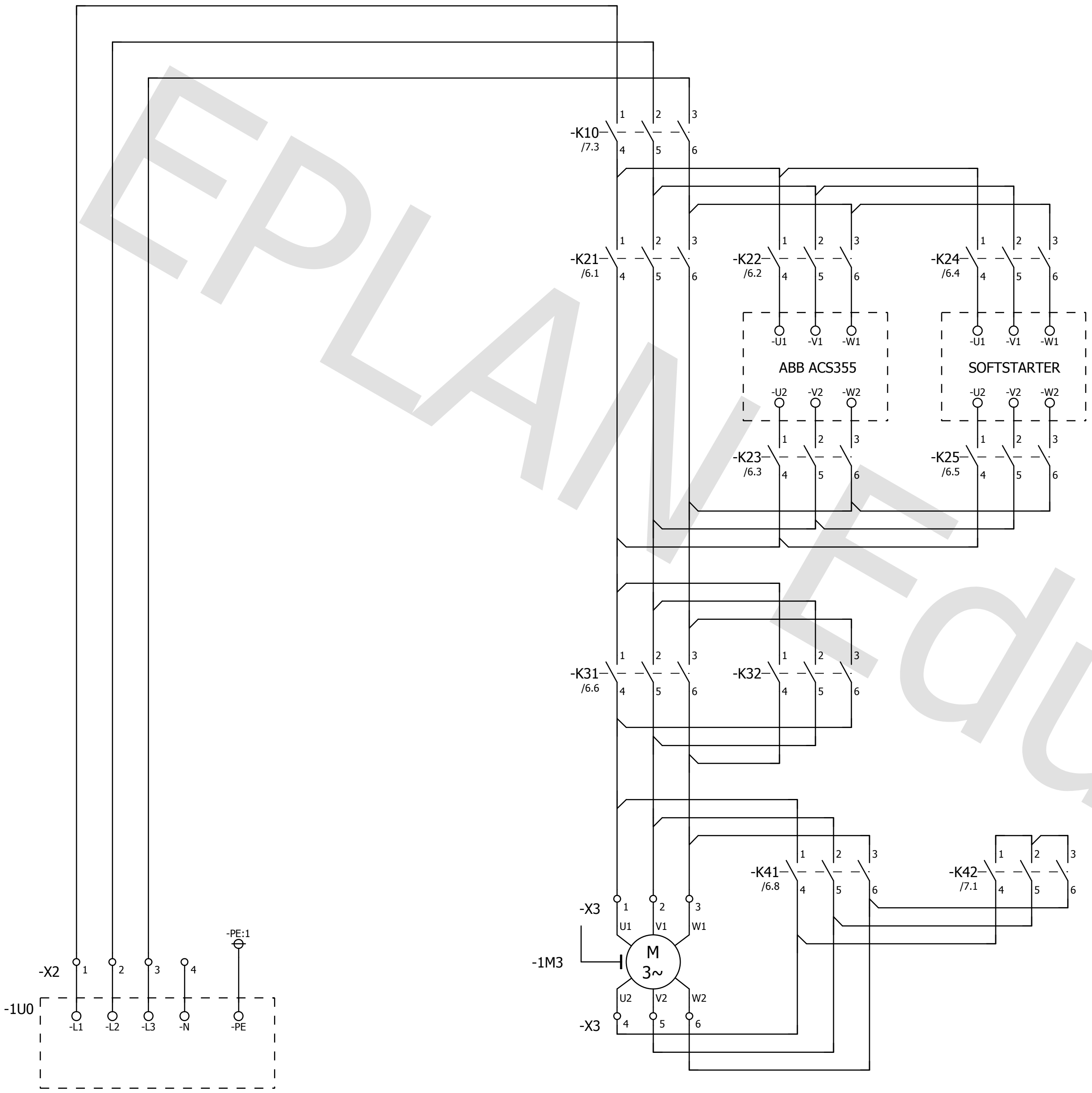
Number	Name	Value	Description
9909	Motor nom. Power	1.8 kW	
9908	Motor nom. Speed	1420 rpm	
9906	Motor nom. Curr	3.5A	
9802	Comm. Prot. Sel.	EXT FBA	Communication through FENA 01
5102	Protocol	10	ProfiDrive
5104	IP Configuration	0	Static IP
5105		192	
5106		168	
5107		1	
5108		4	
5109	Subnet CIDR	24	255.255.255.0
5110	GW Address 1	192	
5111	GW Address 2	168	
5112	GW Address 3	1	
5113	GW Address 4	1	
5501	FBA Data out 1	1	Control word
5502	FBA Data out 2	2	REF1
5503	FBA Data out 3	2202	Parameter 2202 ”Acc. Time 1” is remotely modified this way.
5401	FBA Data in 1	4	Status word
5402	FBA Data in 2	5	Actual Value 1
5403	FBA Data in 3	104	Current
5404	FBA Data in 4	102	Speed
5127	FBA Par Refresh	REFRESH	Restarts the FENA 01 module to apply params.
3401	Signal 1 Param	SPEED	The speed gets shown on the display
2601	Flux opt. Enable	ON	Activates the flux optimization function
2602	Flux braking	FULL	Activates the flux braking function
2606	Switching Freq	16kHz	
2609	Noise Smoothing	Enable	
2208	Emergency Dec. Time	0.5s	
2203	Deceleration time 1	7.0s	
2102	Stop function	RAMP	
1610	Display alarms	YES	
1609	Start Enable 2	COMM	
1608	Start Enable 1	COMM	
1604	Fault reset select	Start/Stop	
1601	Run enable	COMM	
1103	REF1 Select	COMM	
1002	EXT2 Commands	COMM	
1001	EXT1 Commands	COMM	
5127	FBA Par Refresh	REFRESH	Restarts the FENA 01 module to apply params.

!!! Remember to put the drive in **REMOTE control mode** by pressing the ”Loc/rem” button. The display mentions ”REM” in the top left corner when it is in remote control mode.

Appendix 4

ELECTRICAL SCHEMATICS

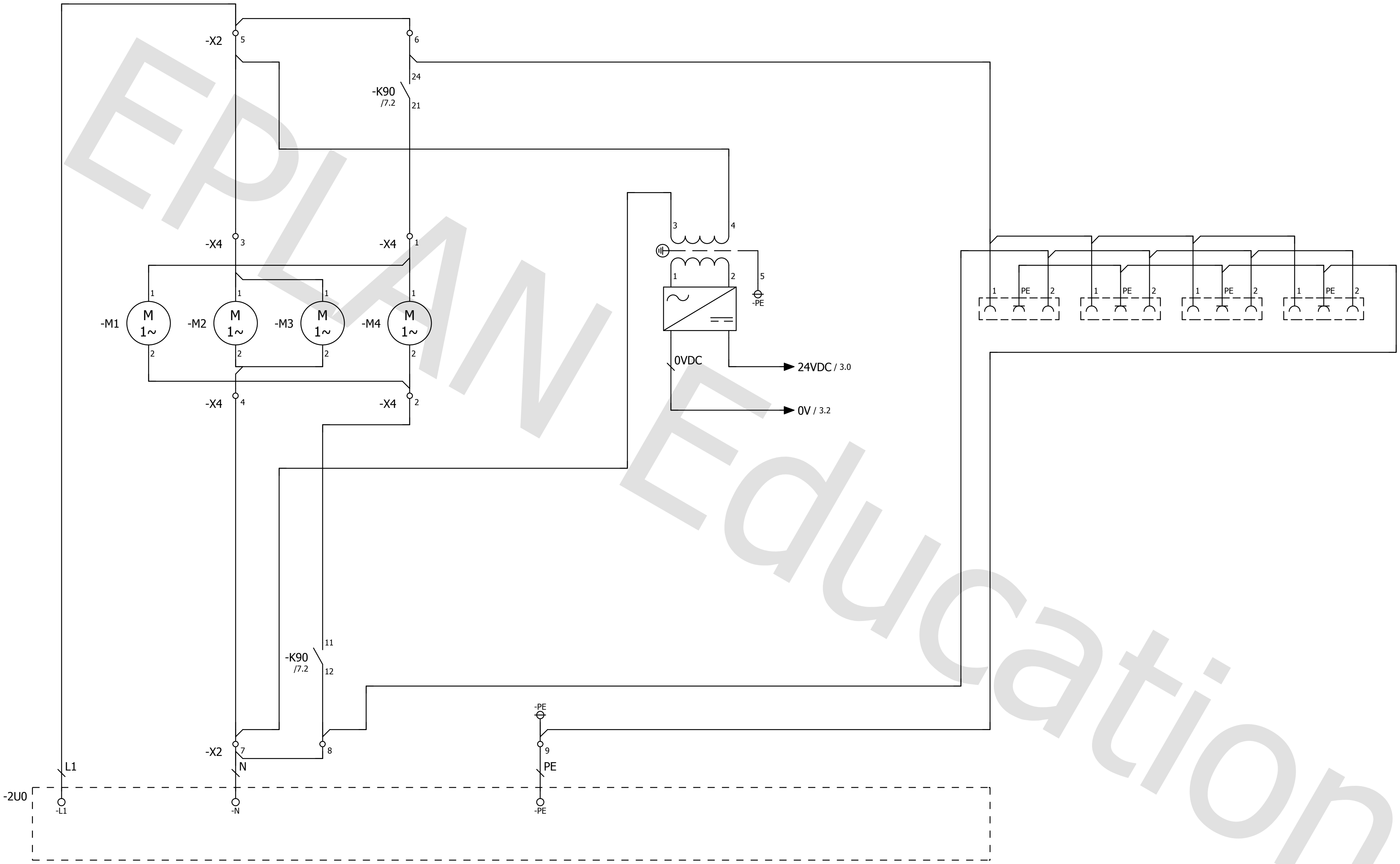
0 1 2 3 4 5 6 7 8 9



EPLAN Education

			Datum	13/05/2015	HAMK		HAMK University of Applied sciences	3F DISTRIBUTION		= ELEC
			Bew.	KEVIN						+
			Gecontr		Projectsjabloon met IEC-coderingsstructuur				IEC_tpl001	Blad 1
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door				Blad 11

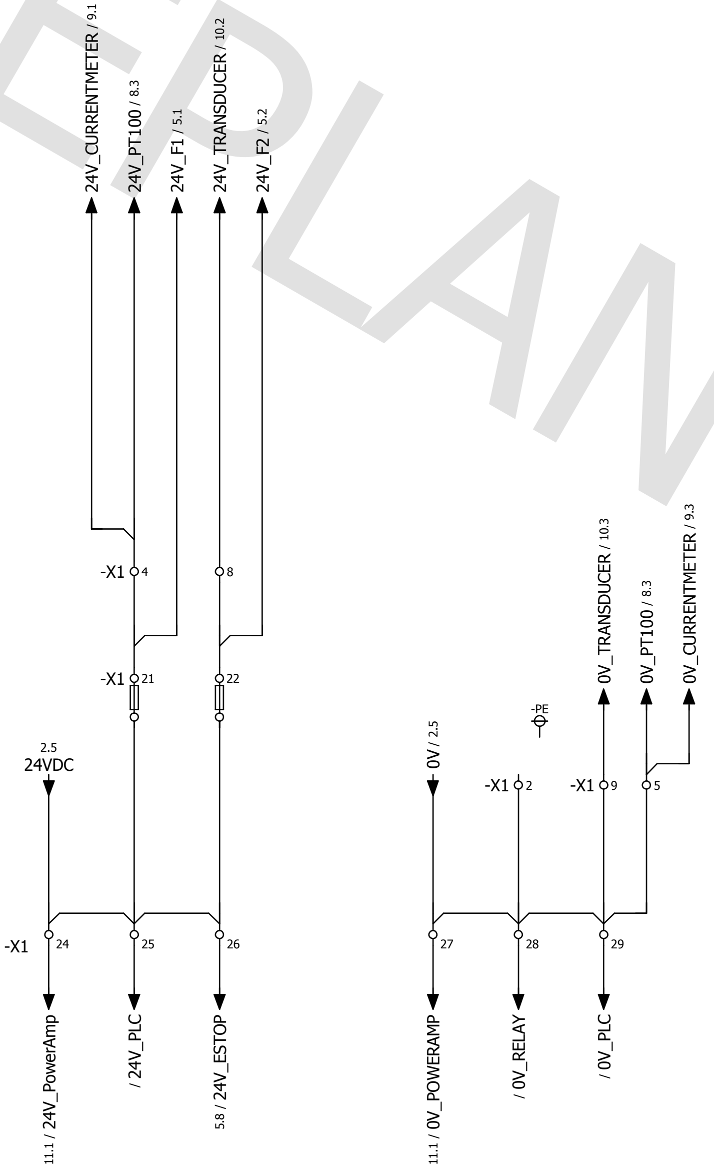
EPLAN Education



			Datum	4/05/2015	HAMK		HAMK University of Applied sciences	1F DISTRIBUTION		= ELEC
			Bew.	KEVIN						+
			Gecontr		Projectsjabloon met IEC-coderingsstructuur				IEC_tpl001	Blad 2
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door				Blad 11

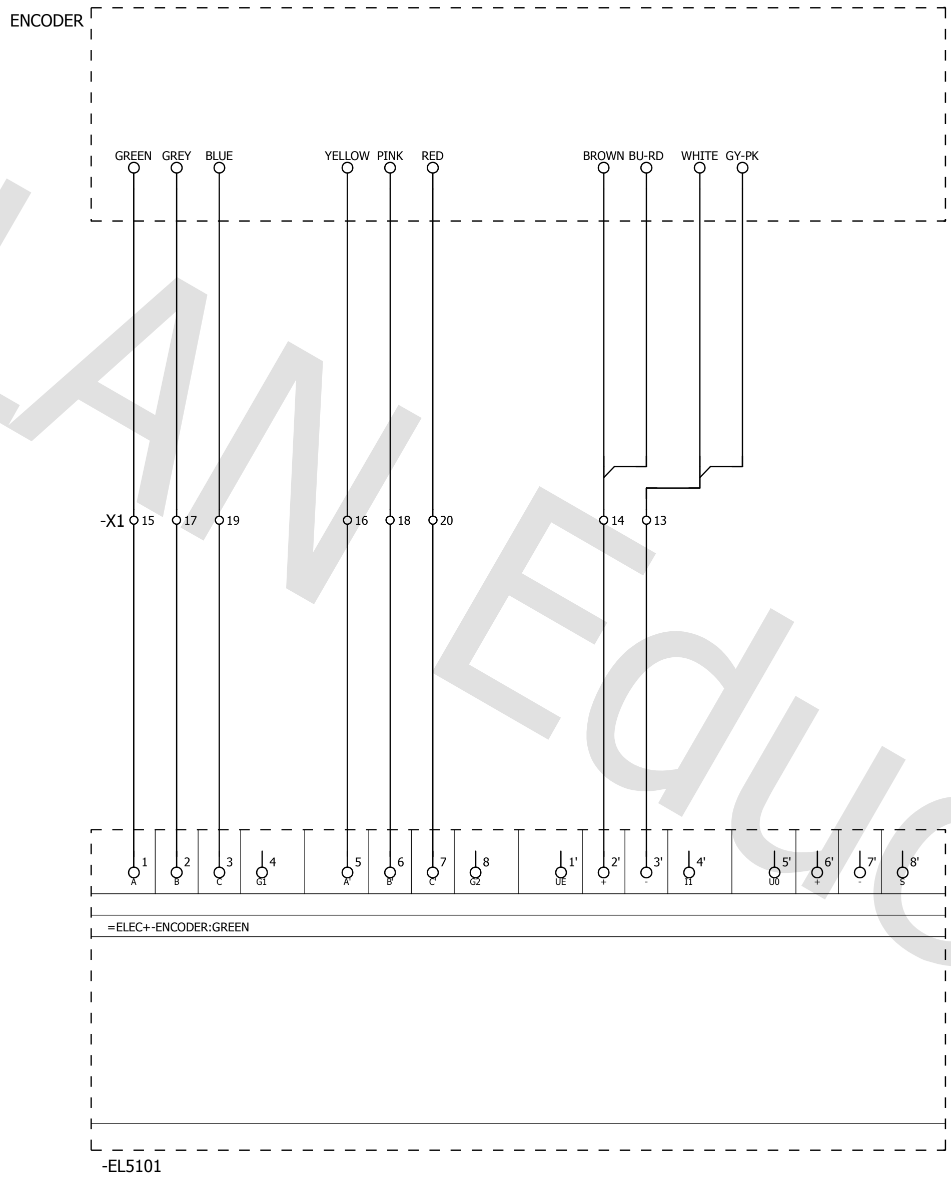
0 1 2 3 4 5 6 7 8 9

EPLAN Education



			Datum	13/05/2015	HAMK		HAMK University of Applied sciences	24VDC		= ELEC
			Bew.	KEVIN						+
			Gecontr		Projectsjabloon met IEC-coderingsstructuur				IEC_tpl001	Blad 3
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door				Blad 11

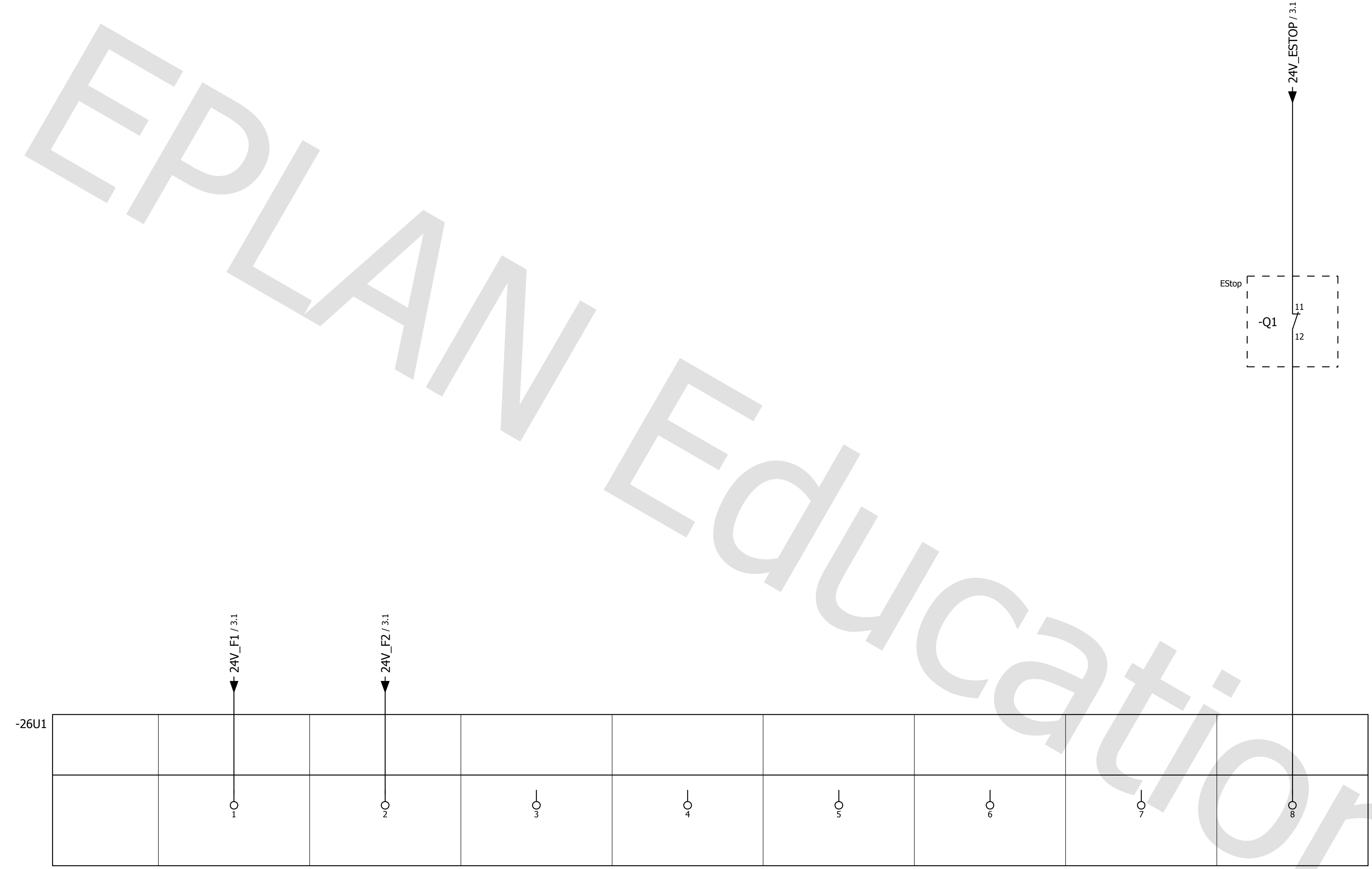
EPLAN Education



3

5

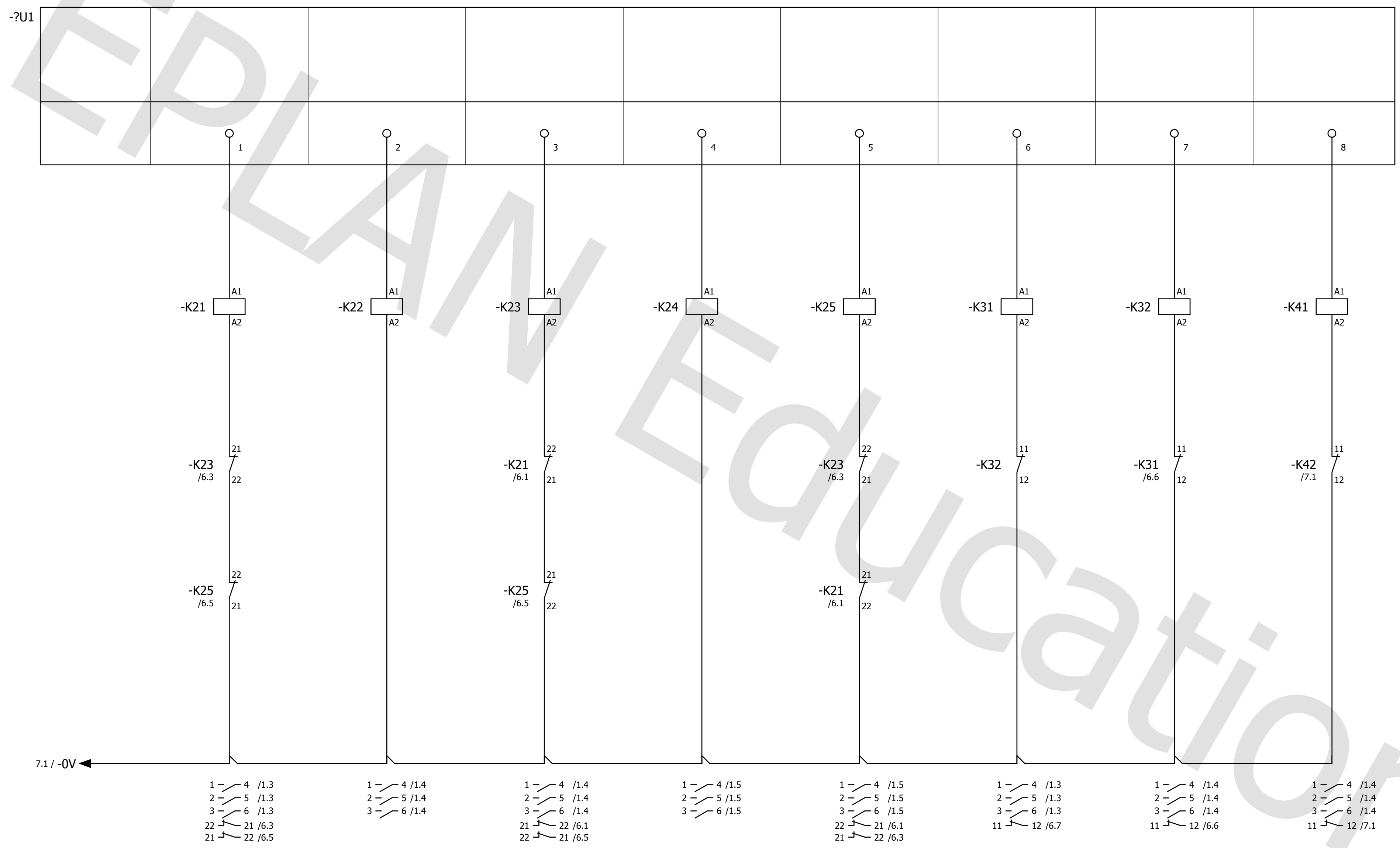
			Datum	4/05/2015	HAMK		HAMK University of Applied sciences	PLC_KL5101		= ELEC
			Bew.	KEVIN						+
			Gecontr		Projectsjabloon met IEC-coderingsstructuur				IEC_tpl001	Blad 4
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door				Blad 11



EPLAN Education

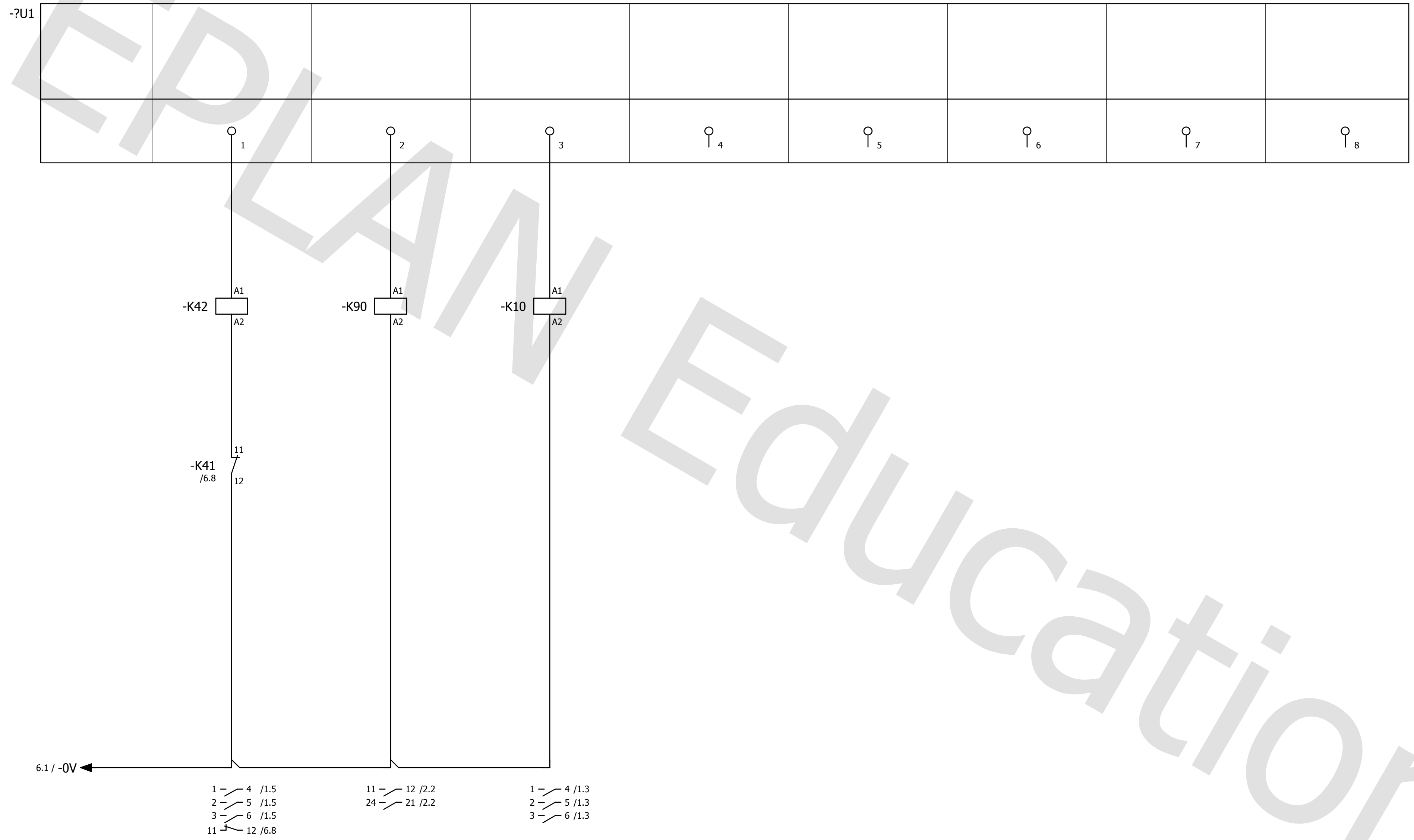
			Datum	4/05/2015	HAMK	HAMK University of Applied sciences	PLC_DI_KL1408		= ELEC
			Bew.	KEVIN					+
			Gecontr		Projectsjabloon met IEC-coderingsstructuur			IEC_tpl001	Blad 5
Wijziging	Datum	Naam	Oorspr		Vervanging van / Vervangen door				Blad 11

EPLAN Education



5

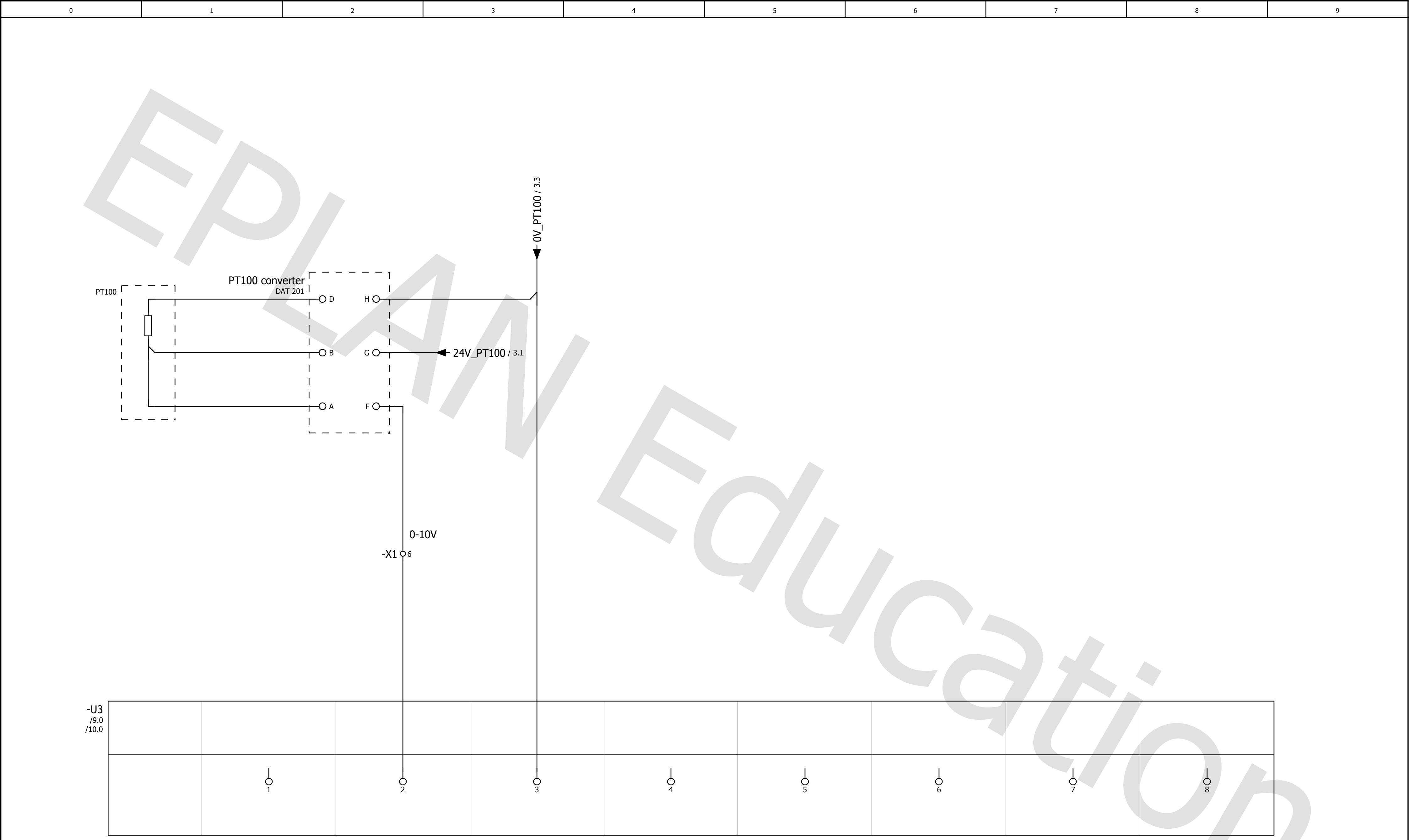
7



EPLAN Education

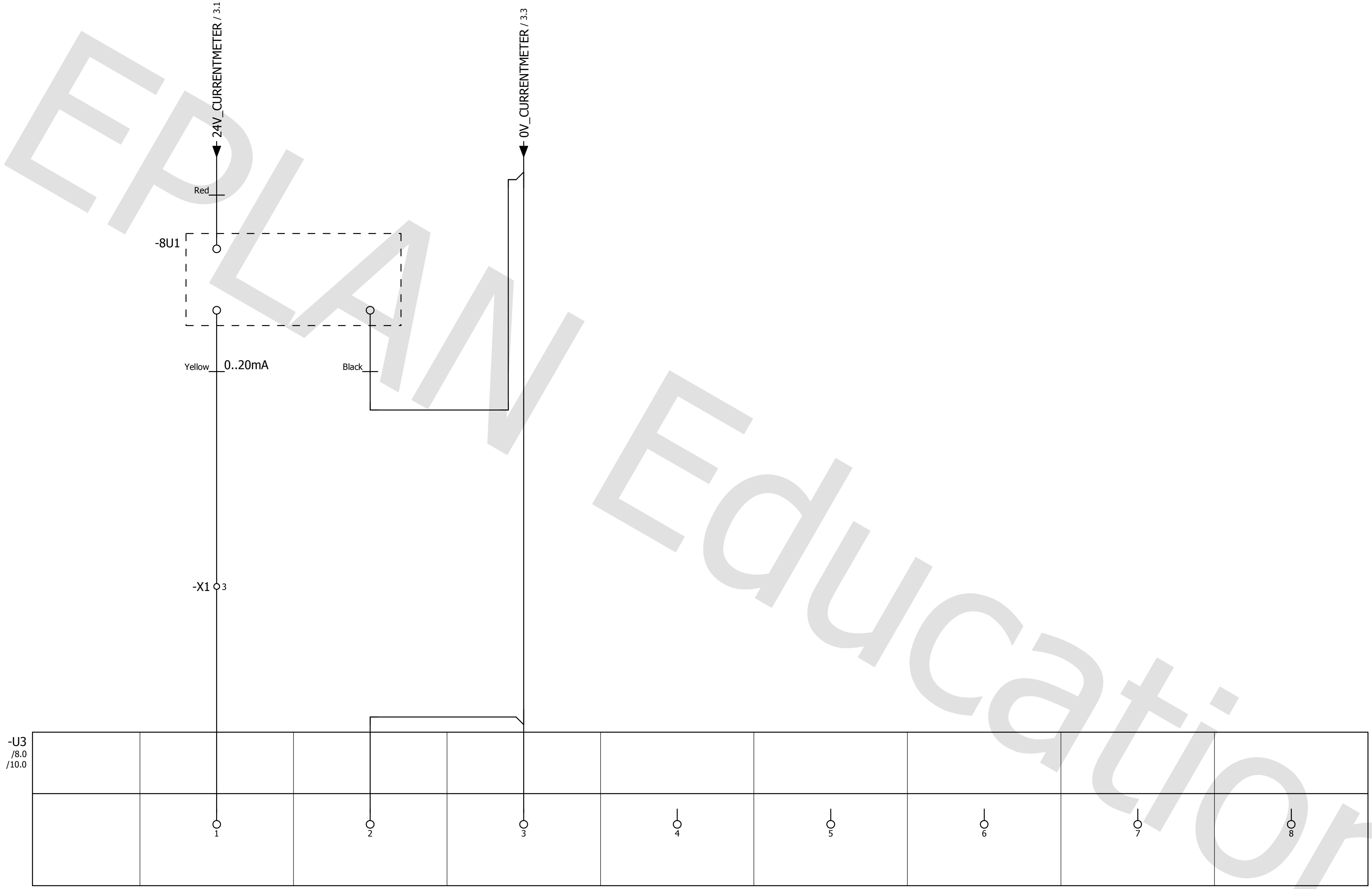
			Datum	4/05/2015	HAMK		HAMK University of Applied sciences	PLC_DO2_KL2408		= ELEC
			Bew.	KEVIN						+
			Gecontr		Projectsjabloon met IEC-coderingsstructuur				IEC_tpl001	Blad 7
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door				Blad 11

EPLAN Education



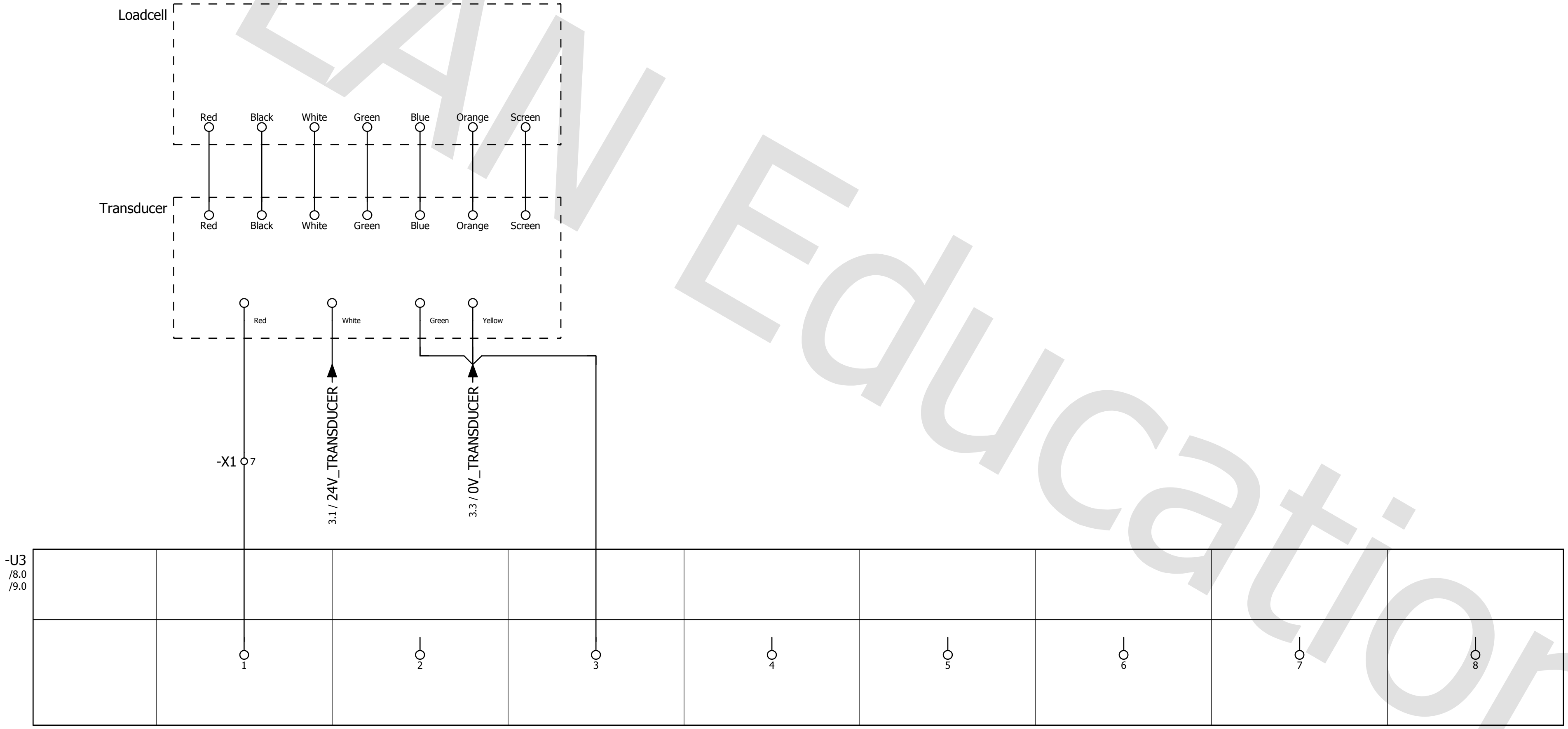
			Datum	13/05/2015	HAMK		HAMK University of Applied sciences	PLC_AI_KL3064		= ELEC	
			Bew.	KEVIN						+	
			Gecontr		Projectsjabloon met IEC-coderingsstructuur				IEC_tpl001	Blad	8
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door				Blad	11

EPLAN Education



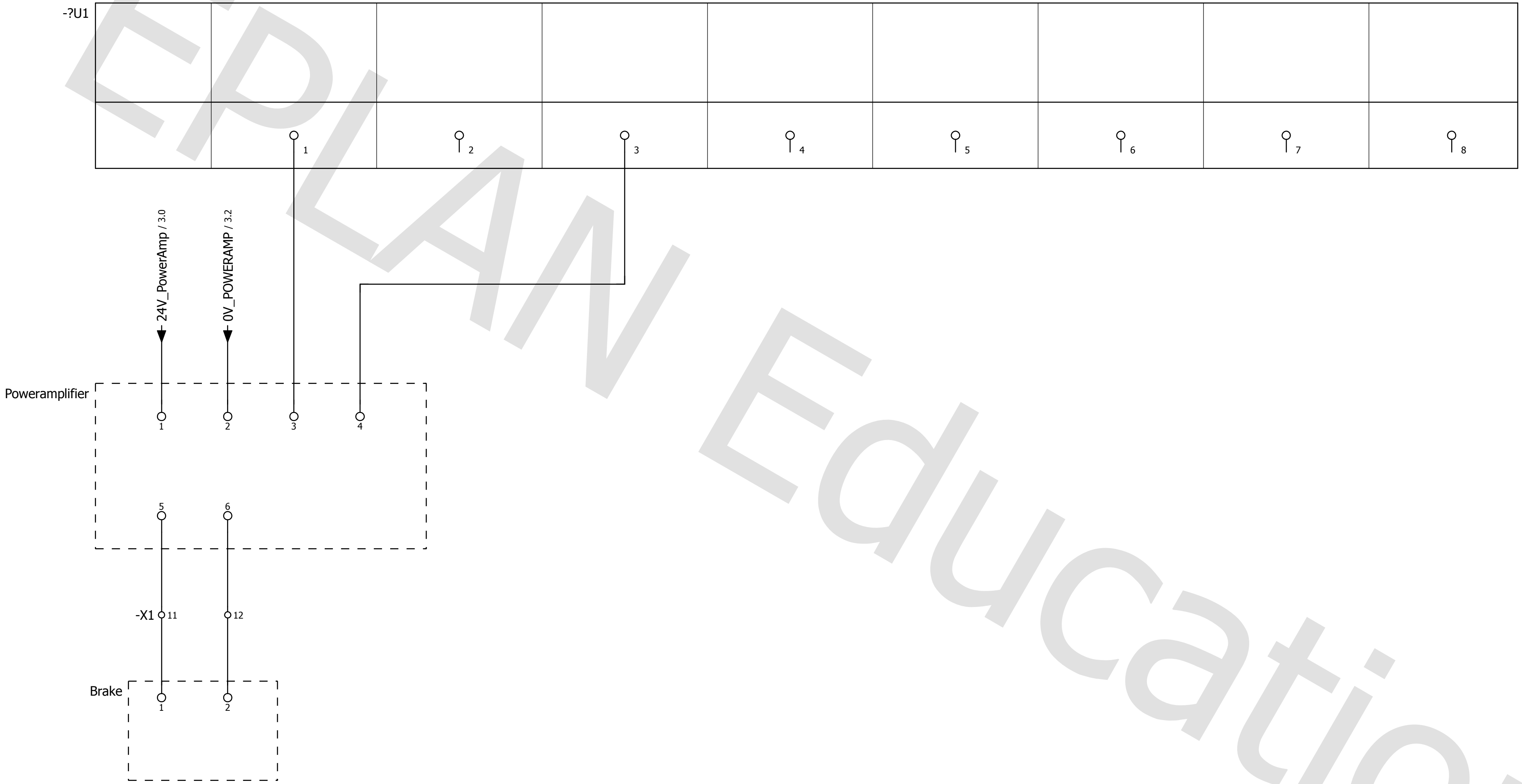
			Datum	4/05/2015	HAMK		HAMK University of Applied sciences	PLC_AI_KL3112		= ELEC
			Bew.	KEVIN						+
			Gecontr		Projectsjabloon met IEC-coderingsstructuur				IEC_tpl001	Blad 9
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door				Blad 11

EPLAN Education



			Datum	13/05/2015	HAMK		HAMK University of Applied sciences	PLC_AI_KL3102	= ELEC
			Bew.	KEVIN					+
			Gecontr		Projectsjabloon met IEC-coderingsstructuur			IEC_tpl001	Blad 10
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door			Blad 11

EPLAN Education



10

			Datum	13/05/2015	HAMK		HAMK University of Applied sciences	PLC_AO_KL4002		= ELEC	
			Bew.	KEVIN						+	
			Gecontr		Projectsjabloon met IEC-coderingsstructuur				IEC_tpl001	Blad	11
Wijziging	Datum	Naam	Oorspr		Vervanging van	Vervangen door				Blad	11

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Upgrade of the motor test unit

Richting: **master in de industriële wetenschappen: energie-automatisering**
Jaar: **2015**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Geutjens, Kevin

Datum: **10/06/2015**