



Automated Control System

Selection, Design and Programming of Automated Control
System for a Conveyor Belt in Fishing Industry

Guðmundur Arnar Grétarsson

**Final thesis for B.Sc. degree
Keilir Institute of Technology
University of Iceland
School of Engineering and Natural Sciences**



Keilir
Institute of
Technology



UNIVERSITY OF ICELAND

Automated Control System

Guðmundur Arnar Grétarsson

24 ECTS thesis submitted in partial fulfillment of a
Baccalaureus Scientiarum degree in Mechatronic Engineering Technology

Advisors

Sverrir Guðmundsson
Úrsúla Ögn Guðnadóttir

Keilir Institute of Technology
University of Iceland
School of Engineering and Natural Sciences
Reykjanesbær, May 2014

Automated Control System

Selection, design and programming of automated control system for a conveyor belt in fishing industry

24 ECTS thesis submitted in partial fulfillment of a *Baccalaureus Scientiarum* degree in Mechatronic Engineering Technology

Copyright © 2014 Guðmundur Arnar Grétarsson
All rights reserved

Keilir Institute of Technology
School of Engineering and Natural Sciences
University of Iceland
Grænásbraut 910
235 Reykjaneshæð

Sími: 578 4000

Bibliographic information:

Guðmundur Arnar Grétarsson, 2014, *Automated Control System*, BSc thesis, Keilir Institute of Technology, University of Iceland, pp. 67.

Printing: Háskólaprent
Reykjaneshæð, June 2014

Abstract

A simulated model of an automated control system for a processing line, under design for the fish processing company Háteigur was implemented and programmed. The processing line consists of conveyor belts and other moving parts. The simulated model demonstrating different sections of the automated control system on the processing line was set up to see whether or not all the requirements proposed by the company Háteigur could be accomplished. The project was broken down into smaller sections, since the real processing line has not yet been constructed, and a smaller conveyor belt used for testing each individual part.

Útdráttur

Hermilíkan af sjálfvirku stýrikerfi fyrir vinnslulínu, sem verið er að hanna fyrir fiskverkunarfyrirtækið Háteigur, var framkvæmd og forritað. Vinnslulínan samanstendur af færiböndum og öðrum hreyfanlegum hlutum. Hermilíkanið sýnir fram á virkni hinna mismunandi þátta af sjálfvirka stýrikerfi vinnslulínunnar, með það fyrir augun að athuga hvort hægt sé að uppfylla allar kröfur sem lagðar eru fram af fyrirtækinu Háteig. Verkefninu var skipt upp í minni hluta, þar sem ekki var búið að smíða vinnslulínuna og prófanir framvæmdar á litlu tilraunafæribandi.

Table of Contents

List of figures	vii
List of Tables.....	xi
Abbreviations.....	xiii
Definitions	xiii
Acknowledgements	xv
1 Introduction.....	1
2 Background – General information	3
3 Processing line functions	5
3.1 Processing line breakdown	5
3.1.1 Short description of desired functionality.....	5
4 Components description.....	11
4.1 Electrical motors.....	11
4.1.1 DC motors – Brush and brushless.....	11
4.1.2 AC motors – Single-phase and polyphase	11
4.1.3 Stepper motors	12
4.1.4 Servo motors	12
4.2 Motor drives	12
4.2.1 Variable speed drive	12
4.2.2 Stepper motor drive.....	12
4.2.3 Servo motor drive	12
4.3 Power supply/converter.....	13
4.4 Programmable logic controller.....	13
4.5 HMI interface	13
4.6 SoMachine software suite	14
4.7 Sensors.....	14
4.7.1 Mechanical sensors – Limit switches	15
4.7.2 Optical sensors	15
4.7.3 Induction sensors	15
4.8 I/O Splitter boxes.....	15
4.9 Communication system	16
4.9.1 I/O direct connections	16
4.9.2 CAN bus.....	16
4.10 Connectors and wiring diagrams.....	17
4.10.1 7/8 power supply to splitter box connector wiring	17
4.10.2 M12 5 pin CAN bus connector	17
4.11 Safety measures.....	18
4.11.1 Emergency stop button	18

4.11.2 Circuit breakers/ Thermal overload breakers	18
5 Programming	19
6 Project components	21
7 Sensor position and motor types	23
8 Results and discussion	25
9 Conclusions	35
References	37
Appendix A – Programming code.....	39
A1 Function Blocks.....	41
_A1.1 ILX Function Blocks	41
_A1.2 ATV Function Blocks.....	46
_A1.3 Other Function Blocks.....	48
A2 Programs	51
_A2.1 Lexium drives reset and set position	51
_A2.2 Altivar drive.....	53
_A2.3 Stacker program.....	54
_A2.4 Testing function blocks for ILX drive	57
Appendix B – Project plan.....	59
Appendix C – Datasheets web links.....	63

List of figures

Figure 1.1 - Processing line (see also tbl. 1.1).....	1
Figure 3.1 - Processing line flowchart.....	5
Figure 3.2 - Conveyor before Stacker 1	5
Figure 3.3 - Stacker 1	6
Figure 3.4 - Conveyor with Scraper and Washer	7
Figure 3.5 - Stacker 2, original design, now been modified.....	8
Figure 3.6 - Conveyor after Stacker 2 (unit 7 in fig. 1.1).....	9
Figure 4.1 - HMISTU touchscreens	13
Figure 4.2 - Machine Struxure pyramid	14
Figure 4.3 - Advantys FTB CANopen IP 67 monobloc I/O splitter boxes	15
Figure 4.4 - 7/8 connector.....	17
Figure 4.5 - M12 5 pin CAN bus connector	17
Figure 4.6 - Circuit breaker and thermal overload relay	18
Figure 7.1 - Suggested sensor position.....	23
Figure 8.1 - Picture of the final equipment setup (see also tbl. 8.1).....	25
Figure 8.2 - Limit switch with metal end plunger	27
Figure 8.3 - Limit switch with steel roller	27
Figure 8.4 - Sensor test platform	27
Figure 8.5 - Lexium drive, CAN settings	28
Figure 8.6 – Splitter box, CAN settings	28
Figure 8.7 - CAN baud rate for project configured (SoMachine)	29
Figure 8.8 - CAN node setting for PLC (SoMachine).....	30
Figure 8.9 - CAN node setting for component in project (SoMachine).....	30
Figure 8.10 - CANopen I/O mapping for sensor variable	31

Figure 8.11 - Sensor 0, sensor 2 and sensor 5 triggered.....	31
Figure A.1.1 - MC_Power_ILX	41
Figure A.1.2 - ConfigureIO_ILX	41
Figure A.1.3 - MC_Readdigitalinput_ILX.....	42
Figure A.1.4 - MC_Movevelocity_ILX	43
Figure A.1.5 - MC_Stop_ILX	43
Figure A.1.6 - MC_Setposition_ILX	44
Figure A.1.7 - MC_Reset_ILX	44
Figure A.1.8 - MC_Moveabsolute_ILX.....	45
Figure A.1.9 - MC_Power_ATV.....	46
Figure A.1.10 - Movevelocity_ATV	46
Figure A.1.11 - MC_Jog_ATV	47
Figure A.1.12 - MC_Stop_ATV	48
Figure A.1.13 - RS	48
Figure A.1.14 - CTU	49
Figure A.1.15 - TON.....	49
Figure A.2.1 - Lexium drives reset and set position	51
Figure A.2. 2 - Lexium drives reset and set position, part 1	51
Figure A.2.3 - Lexium drives reset and set position, part 2	52
Figure A.2.4 - Lexium drives reset and set position, part 3	52
Figure A.2.5 - Lexium drives reset and set position, part 4	52
Figure A.2.6 - Lexium drives reset and set position, part 5	53
Figure A.2.7 - Lexium drives reset and set position, part 6	53
Figure A.2.8 - Altivar drive.....	53
Figure A.2.9 - Stacker program.....	54
Figure A.2.10 - Stacker program, part 1	54
Figure A.2.11 - Stacker program, part 2	54

Figure A.2.12 - Stacker program, part 3.....	55
Figure A.2.13 - Stacker program, part 4.....	55
Figure A.2.14 - Stacker program, part 6.....	56
Figure A.2.15 - Stacker program, part 6.....	56
Figure A.2.16 - Testing function blocks for ILX drive	57
Figure A.2.17 - Testing function blocks for ILX drive, part 1	57
Figure A.2.18 - Testing function blocks for ILX drive, part 2	58
Figure A.2.19 - Testing function blocks for ILX drive, part 3	58

List of Tables

Table 1.1 - Processing line components	1
Table 6.1 - Project components	21
Table 8.1 - Components in fig. 8.1	26
Table 8.2 - Lexium drive, CAN settings	28
Table 8.3 - Splitter box, CAN settings	28
Table 8.4 - CAN bus settings for project.....	29
Table A.1.1 - MC_Power_ILX parameters	41
Table A.1.2 - ConfigureIO_ILX parameters	42
Table A.1.3 - MC_Readdigitalinput_ILX parameters.....	42
Table A.1.4 - MC_Movevelocity_ILX parameters	43
Table A.1.5 - MC_Stop_ILX parameters	43
Table A.1.6 - MC_Setposition_ILX parameters	44
Table A.1.7 - MC_Reset_ILX parameters.....	44
Table A.1.8 - MC_Moveabsolute_ILX parameters.....	45
Table A.1.9 - MC_Power_ATV parameters.....	46
Table A.1.10 - Movevelocity_ATV parameters.....	46
Table A.1.11 - MC_Jog_ATV parameters	47
Table A.1.12 - MC_Stop_ATV parameters.....	48
Table A.1.13 - RS parameters	48
Table A.1.14 - CTU parameters	49
Table A.1.15 - TON parameters	49

Abbreviations

PLC	Programmable Logic Controller
AC	Alternating Current
DC	Direct Current
I/O	Input / Output
X movements	Vertical movement
Y movements	Horizontal movement
HMI	Human Machine Interface
CAN	Controller Area Network
ECU	Electronic Control Units

Definitions

Stacker	Mechanical part of the system that has X and Y movements, it handles one tray at a time and either stacks it or takes it off a stack
Scraper	Stationary metallic plate with grooves in it for trays to pass through, while materials on the trays is pushed off as they pass it.
Washer	Mechanical part of the system, it works like a washing machine, cleaning the trays passing through.

Acknowledgements

The design of the mechanical parts of the processing line was done by Gylfi Þór Guðlaugsson. I want to thank Gylfi for his good advisement, assistance and for always being ready to answer my questions when needed.

1 Introduction

The project is about selecting, designing and programming an automated control system for a new processing line (fig. 1.1) under design for a fish processing company. The processing line will take trays from a stack (approximately eighteen trays in one stack), one at a time, scrape off the contents that are on it, wash it and then restack them again.

The mechanical part of the processing line is designed by Gylfa Þór Guðlaugsson, to accomplish the desired functionality set by the company Háteigur. The design of the automated control system of the processing line, presented in this report, is a collaborative effort between Gylfi, Háteigur and Keilir Institute of Technology.

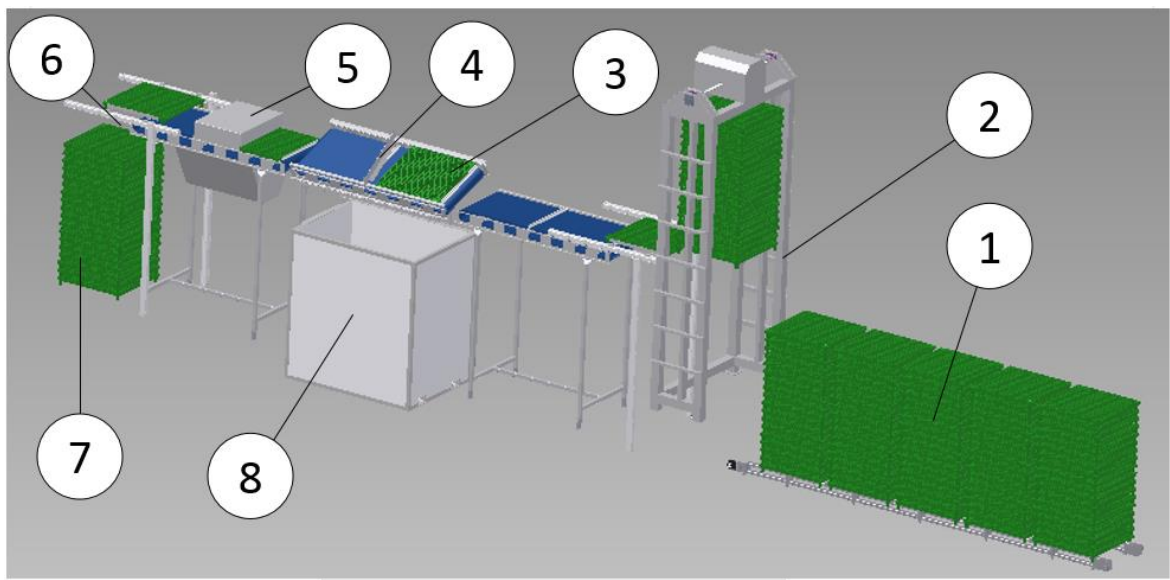


Figure 1.1 - Processing line (see also tbl. 1.1)

Table 1.1 - Processing line components

1.	Conveyor before Stacker 1. (Loading Area)	5.	Washer
2.	Stacker 1	6.	Stacker 2, original design, now been modified
3.	80 x 80 tray	7.	Conveyor after Stacker 2. (Buffer Area)
4.	Scraper	8.	Storage unit

Part of the reason why Háteigur is interested in testing out this idea, rather than just using a robotic arm to handle the stacking of the trays, is from their previous experience in using a robotic arm, it was not performing up to their expectations of speed and needed too much maintenance (Matthías Magnússon, Háteigur, personal comment, Jan. 2014).

Each stack loaded to the starting point of the processing line will consist of approximately eighteen 80 x 80 cm trays. Each tray is expected to weight around eighteen kilograms, when loaded with materials. At the starting point, there will be a loading area that can handle up to five stacks of trays; it can be loaded occasionally every fifteen minutes. This will save labor, as the first stacker unit needs not to be constantly monitored from running empty. The process line is required to be fast enough to manage up to 360 trays in one hour, which amounts to about twenty stacks each hour. At the end of the process line there will be a need for a buffer zone to temporally hold the stacks; i.e. where stacks can wait until they are removed.

2 Background – General information

Conveyor belts have been around for a long time. First they were mainly used to transport coals and timber. Henry Ford was the first person to utilize a conveyor belt as an assembly line for his Ford T-model in the year 1913¹. The purpose of the automation implemented by a conveyor belt system is to reduce the need for labor and thereby lower production cost.

Since then, the technology behind conveyor belts has come a long way, especially after the advent of computerized control system; even though the mechanical part of the conveyors have not changed much, the control systems around them have taken great steps forward.

With automated control system, conveyor belts are able to accomplish much more than just moving material between places. The modern day sensors and mechanical controlling parts, allows for the design of complex multitasking conveyor belt systems, capable of multiple tasks in much shorter time and with less personal than during the early days.

The fishing industry makes use of processing lines for different tasks, utilizing the benefits and efficiency that conveyor belts can bring to their production line. The machinery and capabilities of conveyors continuously evolves, and hence, there is always room for new improved solutions for production lines.

Every year there are some new innovations in the field of automated control systems. Hence, it is very progressive and offers many technical solutions for solving similar tasks. The newest addition is probably the capability of accessing and controlling remotely through computers or smart phones². The systems are able to monitor themselves and notify the personal supervising the equipment immediately if problem arises.

¹ (2014, May) The History Channel website. [Online].
<http://www.history.com/this-day-in-history/fords-assembly-line-starts-rolling>

² (2014, May) Control Engineering. [Online]
<http://www.controleng.com/single-article/machine-control-on-the-water-via-ipad/99508d0c64ddc1495c5006e38337d5af.html>

3 Processing line functions

The processing line is estimated to be over 10 meters, and will have 8 sensors and 9 motors, with room to add more if needed. It needs to be able to control its tasks automatically. The processing line was broken down into smaller sections (fig. 3.1) for this project. Numbers in fig. 3.1 correspond to the functions described in the different sections.

3.1 Processing line breakdown

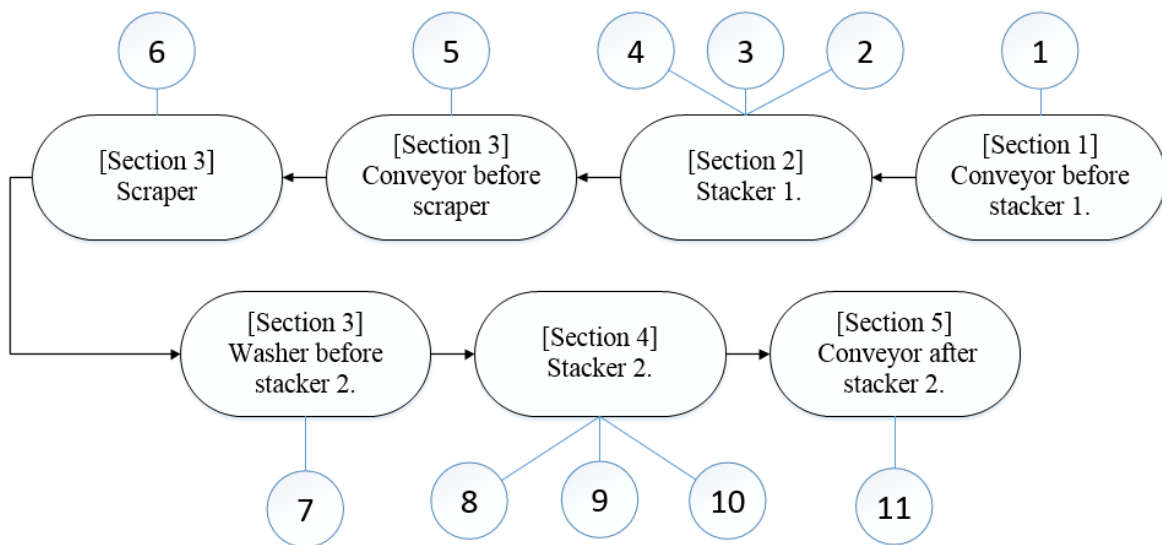


Figure 3.1 - Processing line flowchart

3.1.1 Short description of desired functionality

Section 1 - Conveyor before Stacker 1.

The conveyor belt before Stacker 1 (fig. 3.2) can hold up to five stacks of trays, so that an employee only has to reload new stacks every fifteen minutes or so.

- (1) Is in waiting position while Stacker 1 is in use, moves stacks one step forward when Stacker 1 signals it is empty.

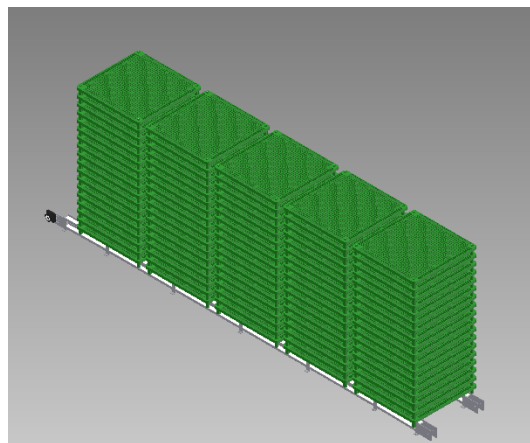


Figure 3.2 - Conveyor before Stacker 1

Section 2 - Stacker 1.

Stacker 1 (fig. 3.3) has X and Y movement. It takes one tray at a time from the stack and puts it on the upper conveyor following Stacker 1. Height of stacks can be different when they are loaded into Stacker 1.

- (2) Has to be able to identify if Stacker 1 is empty or not, and if empty, sends a signal to conveyor in front which responds by bringing in a new stack of trays.
- (3) Has to be able to position trays into correct height, so that the Y movement part can grab the trays one at a time and move them onto the upper conveyor belt.
- (4) Puts the tray on the upper conveyor belt and needs to be able to identify if there is space available on it, or if it has to stop because Section 3 is fully loaded by trays.

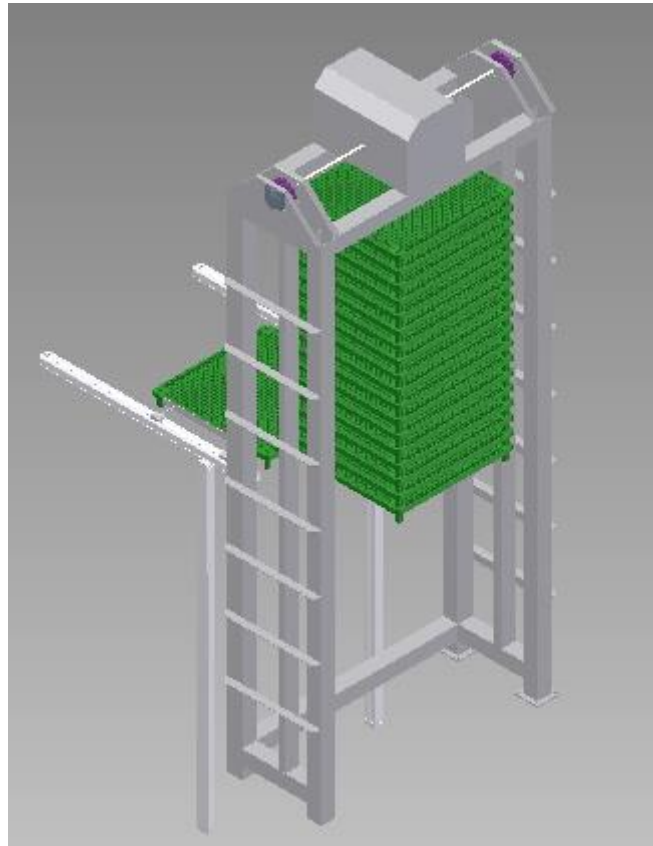


Figure 3.3 - Stacker 1

Section 3 - Conveyor with Scraper and Washer

Trays in this part (fig. 3.4) move forward towards the stationary Scraper, which will remove the material on the trays off them into a storage unit to the side of the conveyor belt. Once that has been done, the tray is transported into a Washer unit, which will rinse all material and dirt that is still stuck on the trays. After the Washer there will be buffer zone for trays from which Stacker 2 picks them up and restacks them back into a stack of trays. This part of the processing line has undergone modifications from the original drawing (fig. 3.4), a modification put forward by Gylfi and Háteigur, which are not represented there. The planned design for this part is to have the starting area on a horizontal plane, from there the conveyor belt will be on a rising incline, ending in a 45° at the part where the Scraper is. After the Scraper the conveyor belt will be on a lowering incline, ending again on a horizontal plane before the Washer unit. It will no longer be three separate conveyor belts as represented in fig. 3.4, but one whole unit.

- (5) Detects when trays are loaded on it by stacker 1 and moves them forward towards the Scraper and Washer. Needs to be able to stop Stacker 1 if belt is not moving and there is not enough room for a new tray.
- (6) Stationary metal plate with grooves in it to let trays pass through but not the material on them.
- (7) Needs to be able to let stacker 1 know if it is full or if there is room available for more trays, also needs to be able to let stacker 2 know if there are trays waiting to be stacked.

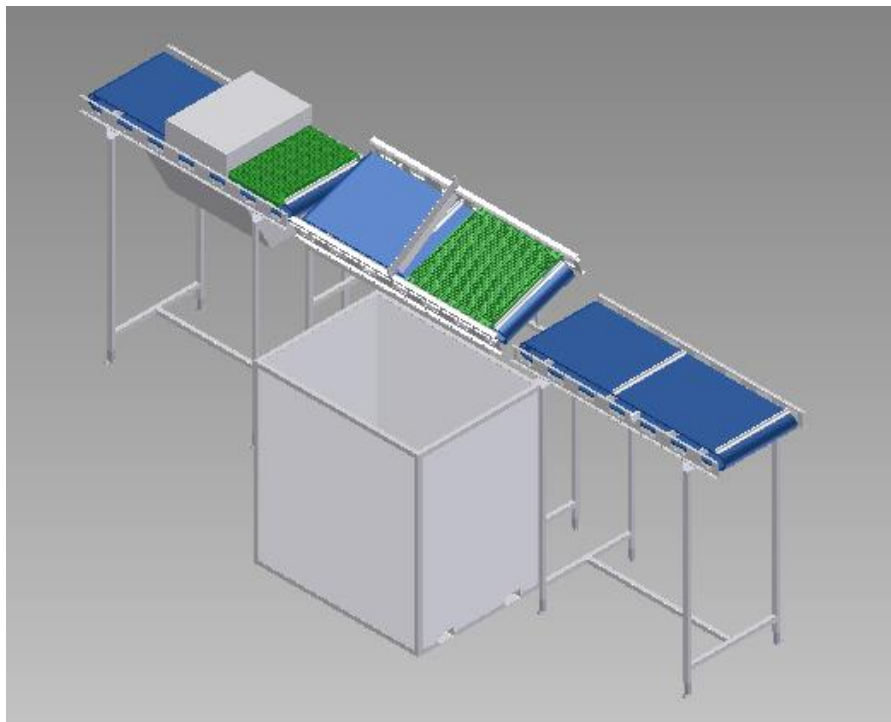


Figure 3.4 - Conveyor with Scraper and Washer

Section 4 - Stacker 2.

Stacker 2 (fig. 3.5) has X and Y movements. It takes one tray at a time and restacks them on a conveyor belt (Section 5). Once the stack reaches a height of eighteen trays, the conveyor belt (Section 5) is started, Stacker 2 is emptied and can then begin stacking a new stack. If the conveyor belt (Section 5) does not have enough space for another stack, Stacker 2 pauses until enough space becomes available.

- (8) Detects if there are trays waiting on the conveyor before Stacker 2, and if so, fetches them and stacks them.
- (9) Needs to be able detect the height of the stack so that the lifting arm (stacking arm) can position itself correctly when stacking. Also needs to be able to determine the height of stack so it knows to stop when a height of eighteen trays is reached.
- (10) Starts the conveyor behind (Section 5) Stacker 2 once stack reaches height of eighteen trays.

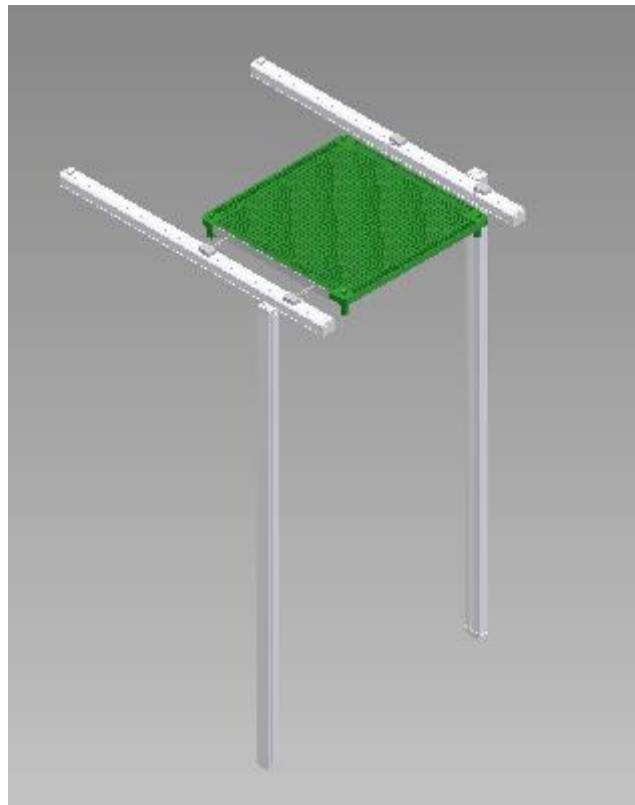


Figure 3.5 - Stacker 2, original design, now been modified

Section 5 - Conveyor after Stacker 2.

The conveyor behind Stacker 2 (fig. 3.6) can hold up to five stacks of trays, so that an employee only has to empty every fifteen minutes or so.

- (11) Needs to be able to detect if there is room for more stacks on the belt, or if it should pause.

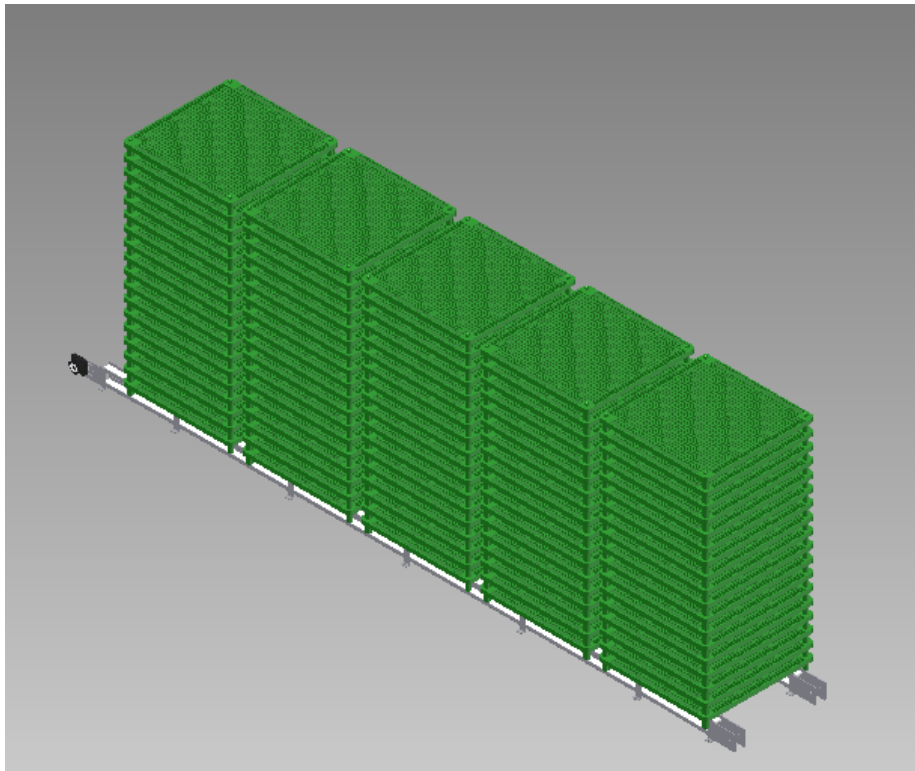


Figure 3.6 - Conveyor after Stacker 2 (unit 7 in fig. 1.1)

4 Components description

Descriptions of components that are usable in conveyor belt systems, the functionality they offer and the differences between them.

4.1 Electrical motors

Even though there are different variations of electrical motors available, they are based on the same principle, i.e. to convert electrical energy into mechanical energy, used to move mechanical parts.

Yet each type of them is differently suited for each task that is needed. Below follows descriptions of types available.

4.1.1 DC motors – Brush and brushless

DC motors can be split into two groups, those that have brushes that are in direct contact with the moving parts of the motor and those that are brushless. The ones with brushes are cheaper to manufacture but require more maintenance compared to the brushless counterparts. DC motors are frequently used in appliances that run on low power or with battery driven source; e.g. copy machines, printer systems and remote controlled models [1]³.

4.1.2 AC motors – Single-phase and polyphase

AC motors can be divided into two groups, single-phase motors and polyphase. Single-phase motors can be found where there are low-power requirements; e.g. general household appliances or power tools like blenders or garage door openers. Polyphase/Multiphase motors are more likely to be used for higher power, like in the industrial sector where more mechanical power is needed from the motors, e.g. conveyor belt systems.

AC motors can be further divided into two types, i.e. induction motors and synchronous motors. As with the DC brush motors, the AC induction motors tend to be a lot cheaper to manufacture and hence much more popular than the synchronous motors. AC induction motors have sometimes been referred to as the workhorse of the industry.

AC motors are generally cheaper than DC motors, more robust, less maintenance cost and more reliable. They do however suffer from that controlling speed is often more complex than for with DC motors, more expensive and involve variable speed drives for accomplishing that part [1]⁴.

³ Chapter 9.5, pp 217-225

⁴ Chapter 9.6, pp 225-227

4.1.3 Stepper motors

Stepper motors are specialized versions of electrical motors providing more accurate control on the motors rotational circle, each rotation (360°) of the motor can be split up into equal steps/angles, where a pulse driven signal can make the motor move one step/angle at a time. Smaller DC motor often come with a set number of steps/angles per rotation, like 200 steps in one rotation, while some AC motors with stepper motor drives can be programmed for different steps in one rotation, depending on the level of control needed [1]⁵.

4.1.4 Servo motors

Servo motors are similar to stepper motors, i.e. they are used in applications where precision is necessary. The difference between them is that while stepper motors are split into steps/angles per rotation and driven by pulse signals, servo motors respond to pulse driving signals by moving into a pre-ordained motor position depending on the frequency of the pulse signal. Servo motors offer the possibility of being easier to program for precise controlling. Stepper motors have traditionally been more popular choice in precise machine controls, like industrial robots, but recent developments have lowered the price of servo motors and made them viable option in the industry.

4.2 Motor drives

Motor drives are used when there is need to affect the driven motor in a more specific manner than just on and off, i.e. when there is need for a finesse in motor control for the application. Motor drives are available in different types for the different variations of motors (Variable speed drive for AC motors, stepper motor drives for stepper motors and servo motor drives for servo motors).

4.2.1 Variable speed drive

Used for the controlling the speed of motors.

4.2.2 Stepper motor drive

Used for controlling a rotation speed of motor, direction of rotation and programming of steps per rotation where available.

4.2.3 Servo motor drive

Often already incorporated into the motor house itself, continuously monitors position of motor and corrects if necessary. Controls the position of motor in accordance to the frequency of the pulse signal received.

⁵ Chapter 9.5, pp 227-234

4.3 Power supply/converter

Systems using PLC, motor drives and sensors, often require the use of power supply/converter modules. If the main power source for the system is AC current, it can affect the performance of computer driven equipment, (PLC). The control circuit boards of motor drives are often powered by 24 volt DC (Altivar 312).

4.4 Programmable logic controller

Programmable logic controllers are digital electronic devices, which use programmable memory to store instructions to implement functions such as logic, sequencing, timing, counting and arithmetic used to control machines and their processes.

They are made robust and often with specific tasks in mind, by tailoring them to meet certain operational needs. By making them more specialized, they run less risk of encountering errors while running under continuous operation. This is often essential on industrial systems.

PLC run their programs continuously and update response and outputs based on the received input signals [1]⁶.

4.5 HMI interface

The human machine interface, the part of the machine that handles the interaction between the operator and the machine; e.g. on/off buttons, computers, touchscreens (fig. 4.1) [2], dials, emergency stop buttons and etc. Every part that can affect the machine operation by usage of operator can be considered as part of the HMI interface on that machine.



Figure 4.1 - HMISTU touchscreens⁷

⁶ Chapter 21, p. 491

⁷ (2014, May) Schneider Electric. [Online]. <http://www.schneider-electric.com/>

4.6 SoMachine software suite

SoMachine software suite is developed by Schneider Electric for use with programming of PLC's, manufactured by them. SoMachine allows you to program and commission all the elements in Schneider's Electric Flexible and Scalable Control platforms (fig. 4.2).

By having all the different machine platforms united into one software package, both programming and general overview of the program is made more visible and easier to manage. The SoMachine program interface is also highly visual and intuitive [3].

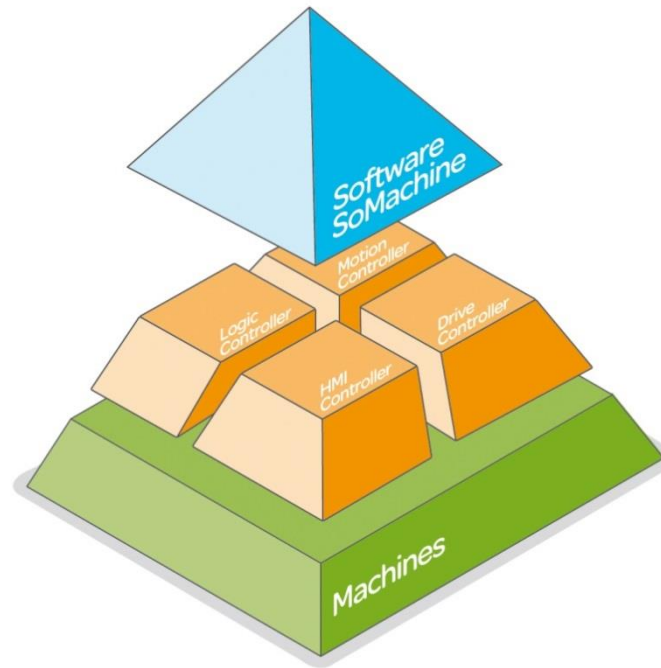


Figure 4.2 - Machine Struxure pyramid⁸

4.7 Sensors

Sensors are essential parts of the automation solutions. They generate the input signals, which the PLC reacts upon.

“The term sensor is used for an element which produces a signal relating to the quantity being measured [1]⁹”.

The selection of sensors for automated applications depends on the parameters to be observed, e.g. counting of objects, volume of liquid in containers, temperature, color, material type or other possible variations for measurement.

⁸ (2014, May) Schneider Electric. [Online]. <http://www.schneider-electric.com/>

⁹ Chapter 2, p. 29

4.7.1 Mechanical sensors – Limit switches

Sensors that trigger on contact. They generate signals by passing current through them, they are interpreted either as 1 and 0 (on or off). The PLC can then be programmed to respond to their signal depending on the function needed, counting or reacting when the sensors triggers.

4.7.2 Optical sensors

Optical sensors do not use physical contact mechanism to trigger the measurement, they generate a beam of light, which is either received by sensor on the opposite side or bounced back by reflective surface. When that beam is broken, the sensor triggers.

4.7.3 Induction sensors

Induction sensors are like mechanical sensor and optical sensors, but do not use a physical contact mechanism or a beam of light to trigger. They measure the change in inductance of the sensor coil when objects pass nearby their sensory surface. Used for the detection of metallic objects.

4.8 I/O Splitter boxes

Just like extension modules can be added to the PLC for increased I/O operations, there is also the option of using I/O splitter boxes (fig. 4.3) which operate on the fieldbus. Since they can make use of the fieldbus, the CANopen protocol can be used in its setup of nodes and programming. By use of I/O splitter boxes, a centralized automation system can be partly decentralized. The I/O splitter boxes can be connected to each other, making cable connections more simplified and the replacements of parts easier on large automated systems, instead of running cables from each component to the control cabinet where the PLC is located [4].

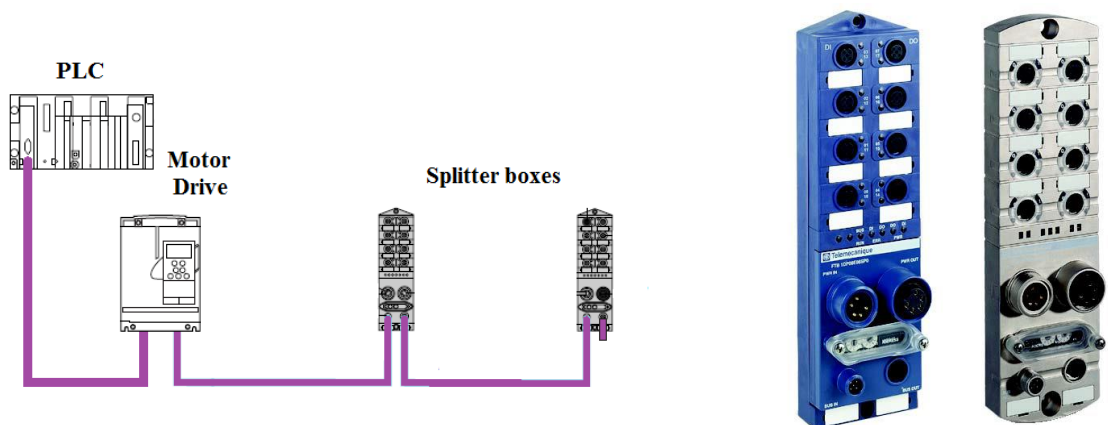


Figure 4.3 - Advantys FTB CANopen IP 67 monobloc I/O splitter boxes

4.9 Communication system

Parts of the automated control system need to communicate with each other so that the interaction between different machine components does not interfere with each other or disrupt the machine process.

This is done by using sensors or other monitoring equipment that conveys information on the machine status to the PLC, which it then can react upon, depending on its programming.

This can be implemented by direct cables from sensors to PLC using the I/O connectors on the PLC or by using the serial bus interface and/or CAN bus interface when available.

4.9.1 I/O direct connections

Connection cables that lie directly from the monitoring equipment towards the PLC, using the existing I/O connectors that are available on the PLC or additional I/O extension modules. They are easy to setup and connect, but have the disadvantage of requiring separate cable from each equipment to the PLC. On larger automated machine installations; with multiple monitoring equipment, this can lead to overcrowding within the control cabinet from cables.

4.9.2 CAN bus

“A modern automobile may have as many as seventy electronic control units (ECUs) for various subsystems, e.g. engine management systems, anti-lock brakes, traction control, active suspension, airbags, cruise control, windows, etc. This could involve a lot of wiring. However, an alternative approach is use a common data bus with data transmitted along it and made available to all parts of the car. Bosch has thus developed a protocol known as CAN or Controller Area Network. The Can bus is now also used as a fieldbus in other automation systems.

Can is a multi-master serial bus standard for connecting ECUs. Each node in the system is able to both send and receive messages [1]¹⁰.”

CANopen is a higher level communication protocol that makes use of the CAN bus.

¹⁰ Chapter 22, p. 507

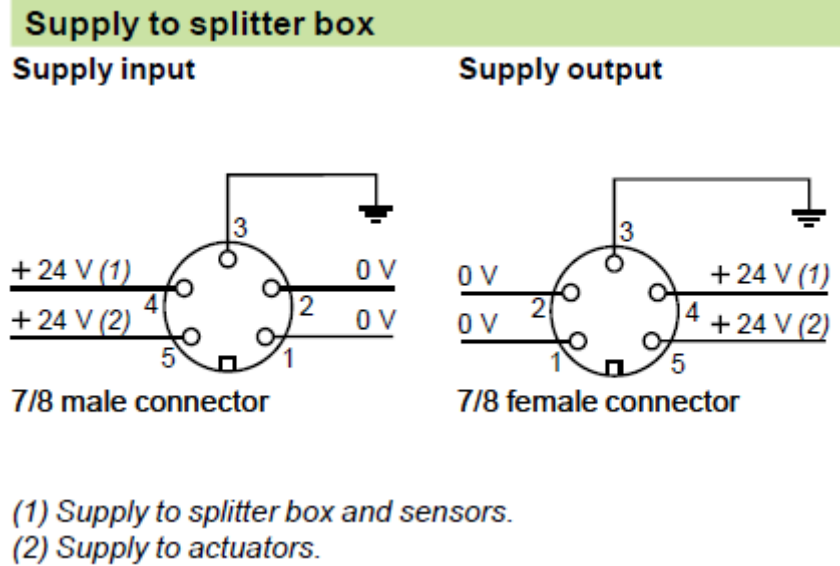


Figure 4.4 - 7/8 connector¹¹

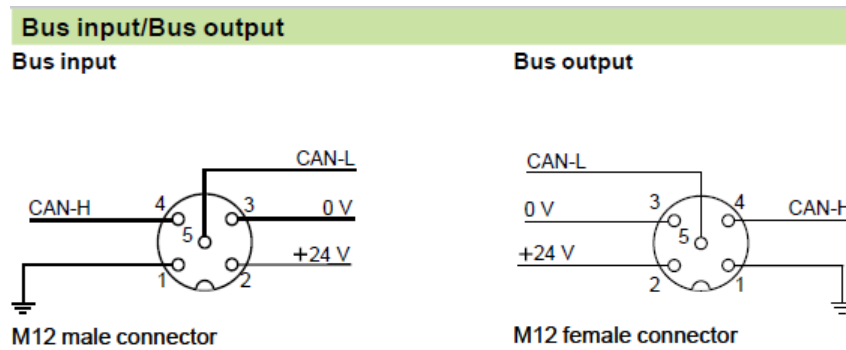


Figure 4.5 - M12 5 pin CAN bus connector¹²

4.10 Connectors and wiring diagrams

4.10.1 7/8 power supply to splitter box connector wiring

Used to supply power to the I/O splitter boxes (fig. 4.4).

4.10.2 M12 5 pin CAN bus connector

Used in the communications between I/O splitter boxes and other nodes of the CAN bus system (fig. 4.5).

¹¹ See Appendix C, IP 67 monobloc I/O splitter boxes for fieldbuses Advantys, FTB splitter boxes

¹² See Appendix C, IP 67 monobloc I/O splitter boxes for fieldbuses Advantys, FTB splitter boxes

4.11 Safety measures

As with any mechanical applications, it is important to implement safety measures, both to increase the safety of personnel that make use of the equipment and to prevent mechanical parts if something goes wrong.

4.11.1 Emergency stop button

An emergency button intended to shut down all operations by cutting off power, in some cases it will trigger the braking on motors, so that they will stop quickly.

4.11.2 Circuit breakers/ Thermal overload breakers

Motors and other parts can quickly heat up if the system starts to draw too much current, e.g. when it is running under more stress than intended. Circuit breakers, thermal overload relays (fig. 4.6) and temperature sensors located on motors can be installed to prevent equipment failure before it occurs.



Figure 4.6 - Circuit breaker and thermal overload relay

5 Programming

The programming for this project is all handled through the SoMachine software suite from Schneider Electric. As stated in chapter 3, the project is split into smaller parts, each part pertaining to a specific section (fig. 3.1) of the processing line. By doing so, the programming blocks for the processing line, can be designed so that they are interchangeable with each other, making it simpler to add or change parts of the project afterwards.

SoMachine offers the following six choices in programming language:

- Continuous Function Chart (CFC)
- Function Block Diagram (FBD)
- Instruction List (IL)
- Ladder Logic Diagram (LD)
- Sequential Function Chart (SFC)
- Structured Text (ST)

Programming was done in Function Block Diagram (FBD), which complies with the international standard for programmable controllers programming languages IEC 1131-3.

6 Project components

Compiled list of the electrical components and software used in the testing of this project (tbl. 6.1), wiring, cables, and small conveyor belt not included.

Table 6.1 - Project components

Parts number	Quantity	Description
K21R71K4	1	AC motor, 3Ph, 400V, 0,25kW, Asynchron motor
ILA1F572PCA0	2	Lexium integrated drive, AC synchronous servo motor
PAS42BRM0600	2	Lexium Linear Motion, 600 mm.
ATV312H037N4	1	Variable speed drives for asynchronous motors
ABL8REM24030	1	Power supply, 24V 3A
ABL8RPS24100	1	Power supply, 24V 10A
TM238LFDC24DT	1	Logic Controller
SoMachine Suite v.3.1.10.1	1	Software
VMWorkstation v.10	1	Software
CoDeSys gateway program v.3	1	Software
ZCP21M12	8	Limit switch body
ZCE02	4	Steel roller plunger
ZCE11	4	Metal end plunger with nitrile boot
HMISTU855	1	Touch panel screen
FTB1CN16CPO	1	Advantys FTB CANopen IP67 Monobloc I/O Splitter box
VW3CANTAP2	1	CANopen junction box
LC1D09	1	Thermal overload relay
GV2-P06 / 1-1.6A	1	Circuit breaker, 1-1.6A

7 Sensor position and motor types

Suggested position for sensors (fig. 7.1) on the processing line and the selection of motors for each section. (See fig. 3.1 in chapter 3.1 for section breakdown).

Section 1 - Conveyor before Stacker 1.

1. Limit switch. When triggered informs the PLC that a stack of tray is waiting to be loaded into Stacker 1.
- 1 AC motor. With this part there is no need for speed control and therefore a circuit breaker and a thermal overload relay will be enough for motor control.

Section 2 - Stacker 1.

2. Limit switch. When triggered informs the PLC that a stack of trays has been loaded into Stacker 1, ready to be unstacked upon Section 3.
3. Limit switch. When triggered informs the PLC that the top tray is in position for being picked up and moved onto Section 3.
- 2 servo motors. One for vertical movement of lifting the stack up, and the other for horizontal movement of picking trays and placing them onto Section 3. They offer position feedback at all times, enabling movement control for correct positioning of trays.
- 1 DC motor. Small motor for controlling the holding unit which grabs the trays and holds them while they are transported from stack onto Section 3.

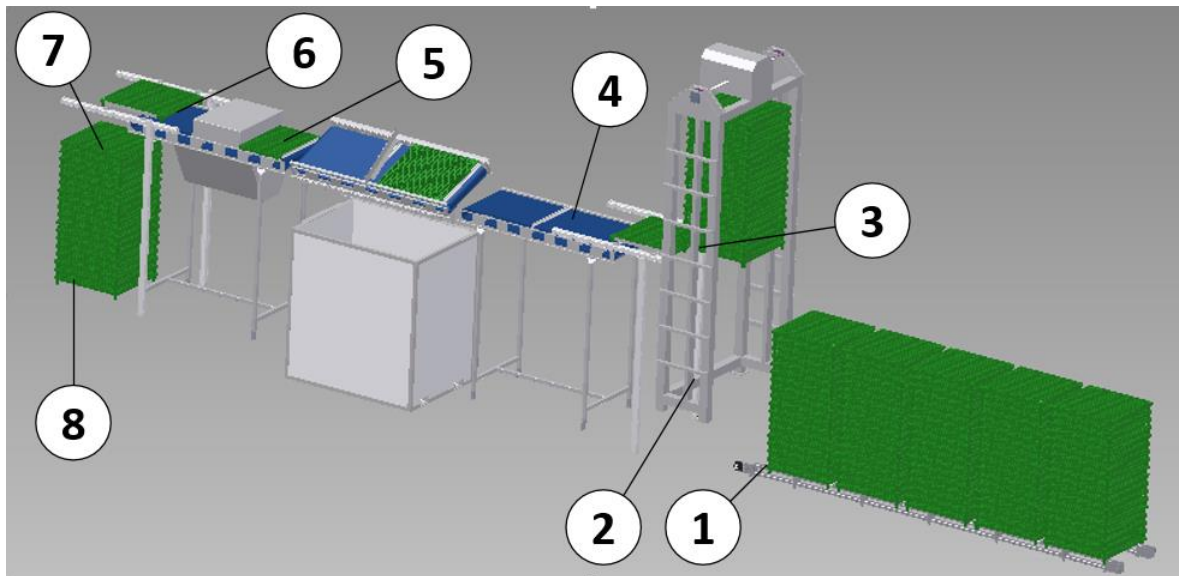


Figure 7.1 - Suggested sensor position

Section 3 - Conveyor with Scraper and Washer

4. Limit switch. When continuously triggered informs the PLC that Section 3 is fully loaded by trays. Making it necessary for Stacker 1 to go into waiting position until this sensor goes off and there is room for another tray.
5. Limit switch. Informs the PLC that a tray is about to pass into the Washer unit and that it should start up. Can also be used as a counting trigger for number of trays going through the processing line.
6. Limit switch. When triggered it informs the PLC that a tray is ready to be picked up by Stacker 2.
 - 1 AC motor. AC motor combined with a variable speed drive, making it possible for speed control on this Section. Some materials that go through the processing line and are to be scraped off might offer different resistance (wet material sticks more to trays compared to dry material), making it a viable option to control the speed of the conveyor belt at this section by an operator. Lowering the speed for the hard to scrape off material and raising it again for the easier material.

Section 4 - Stacker 2.

7. Limit switch. When triggered informs the PLC that the stack of trays has reached its intended height (18 trays), and can be moved onto Section 5.
 - 2 servo motors. One for vertical movement of lowering the stack down, and the other for horizontal movement of picking trays up from Section 3 and stacking them. They offer position feedback at all times, enabling movement control for correct positioning of trays.
 - 1 DC motor. Small motor for controlling the holding unit which grabs the trays and holds them while they are transported from Section 3 onto a stack.

Section 5 - Conveyor after Stacker 2.

8. Limit switch. When triggered informs the PLC that Section 5 is fully loaded, and therefore Stacker 2 will need to go into waiting position until this Section has been unloaded and there is room again for new stacks.
 - 1 AC motor. With this part there is no need for speed control and therefore a circuit breaker and a thermal overload relay will be enough for motor control.

8 Results and discussion

It was decided to use mostly equipment from Schneider Electric for this project. The reason behind that was not only that they are an accomplished manufacturer of industrial solutions for automated control systems, but also because Keilir Institute of Technology already had much of the electronic equipment needed to make the mini-simulator of the belt conveyor (fig. 8.1).

By using the equipment available on stock at Keilir Institute of Technology, testing of different components for the project could be carried. This made it possible to test and change out components as the project progressed.

A virtual machine software, VMware workstation 10, was used for this project. It was needed for running a virtual machine provided by Keilir Institute of Technology. That virtual machine had Windows 7 operating system with the SoMachine software suite installed along with the CoDeSys gateway program. These programs (SoMachine and CodeSys) were necessary for the configuration between electrical components and the programming part of this project.

Once the necessary programs had been procured and installed, it was decided to use an existing conveyor belt system at school. It already had a variable speed drive (ATV312H037N4), PLC (TM238LFDC24DT), HMI touchscreen (HMISTU855) and an AC motor. On that setup additional components (see chapter 6 for more detailed list) would then be added later and tested.

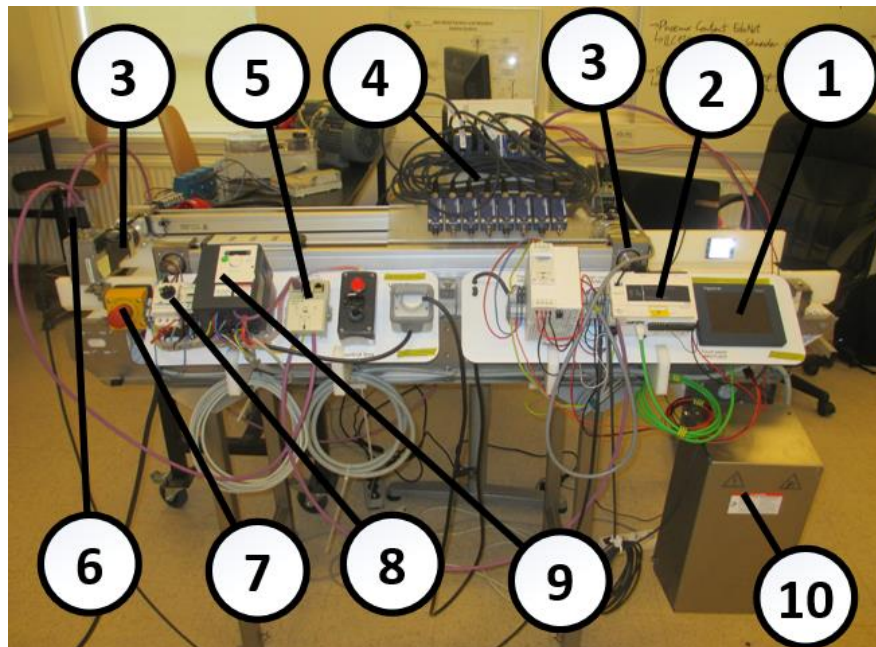


Figure 8.1 - Picture of the final equipment setup (see also tbl. 8.1)

Table 8.1 - Components in fig. 8.1

1	Touch panel screen - (HMISTU855)	6	CAN bus connectors
2	Logic controller - (TM238LFDC24DT)	7	Emergency stop button
3	Lexium integrated drive - (ILA1F572PCA0)	8	Circuit and Thermal overload breakers
4	Sensor test platform	9	Variable speed drive - (ATV312H037N4)
5	CANopen junction box - (VW3CANTAP2)	10	AC motor - (K21R71K4)

The variable speed drive was used to test and simulate the parts of the project that would require the use of a conveyor belt. Two Lexium integrated drives (ILA1F572PCA0) were selected and used to test and simulate the Stacker 1 and 2 parts of the project, one used for the X movement while the other would handle the Y movements.

When the two Lexium integrated drives were added, it was decided to change out the existing power supply (ABL8REM24030) on the conveyor belt system, and replace it with a more powerful one (ABL8RPS24100) ensuring that the Lexium drives would be able to draw enough current when needed.

Part of the project was also checking into if using I/O splitter boxes would be a feasible solution. The Advantys FTB CANopen IP67 Monobloc I/O Splitter box (FTB1CN16CPO) was selected because it makes use of the CANopen protocol. The CAN bus was already being used for other electronic components of the project, e.g. Altivar 312 variable speed drive, Lexium integrated drives. The CANopen junction box (VW3CANTAP2) is used in this project so that the variable speed drive is able to connect to the CAN bus.

By setting the system up in such a way, components could be connected in series, and in case of malfunctions or replacements, such a setup should make it easier in the long run, to find and switch out faulty components.

First it was necessary to test and wire up existing components separately, starting by testing if the conveyor belt was running as intended with the variable speed drive, and if all the cables and connectors were wired correctly. The original setup had a three phase power connector for powering the variable speed drive and the AC motor. A power supply was also wired to that connector, in order for the necessary 24 volt required to run the PLC and the HMI could be provided.

There were no readymade cables for the CAN bus ports of neither the I/O splitter boxes nor the Lexium drives. Therefore they had to be made on the spot with the necessary M12 5 pin connectors and cable in order for connecting those components to the CAN bus. Where no connectors ends were available (7/8 connector was not available), connections and cable were setup in a temporarily fashion (see chapter 4.10.1 for wiring diagram), so that working principles of the project could be proven.

A clear plastic board was procured and drilled with holes, so that it could be possible to fasten 8 sensors; 4 with metal end plungers (fig. 8.2), 4 with steel roller plunger (fig. 8.3) and a splitter box onto one platform.

The sensor test platform (fig. 8.4) was then connected to the PLC with a CAN bus cable, and a temporary testing program was created on the SoMachine, to test out if the communications were working between the splitter box and the PLC as intended, at the same time, the variable speed drive was disconnected from the CAN bus so that it would not interfere with the testing of new components. Once the sensor test platform had been tested out, it was disconnected and the Lexium integrated drives were wired up and connected to the CAN bus and tested in a similar fashion.



Figure 8.2 - Limit switch with metal end plunger



Figure 8.3 - Limit switch with steel roller

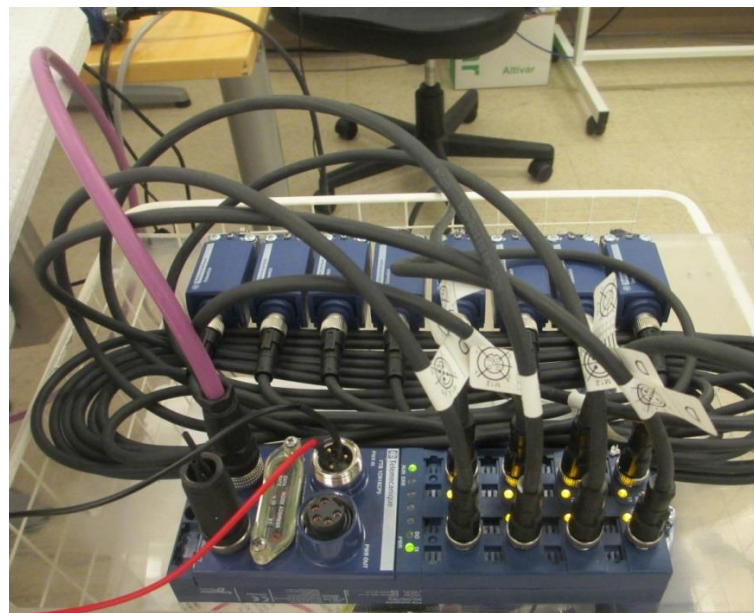


Figure 8.4 - Sensor test platform

Once these different components had been tested out separately, it was time to connect them serially, line to line, and get them to work together as a whole. First attempts were not successful, as the CAN bus system ran into errors on the PLC. Connecting cables were checked for faulty wiring and the CAN bus node configurations were looked at, problem was traced down to being incorrect CAN bus node manual configuration on components and setting of baud rate. When that was done, the components were able to communicate between each other using the CAN bus.

To configure the CAN node settings for the Lexium integrated drives, the motor housing needs to be opened and the switches (fig. 8.5 and tbl. 8.2) inside manually set to the desired settings. For the Altivar 312 drive, CAN bus settings are configured through the inbuilt HMI interface. By selecting [RDY-CON-ADCO] the CAN node number can be set and through [RDY-CON-BDCO] the baud rate can be configured. The Altivar 312 drive also has to have [RDY-CTL-FR1-NET] set, in order for it to accept remote control through the CAN bus. The splitter box is configured through switches (fig. 8.6 and tbl. 8.3) located on the front side.

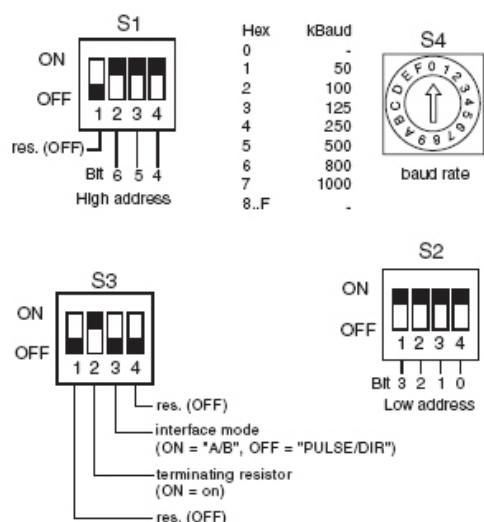


Figure 8.5 - Lexium drive, CAN settings

Table 8.2 - Lexium drive, CAN settings

Switch settings S1 and S2:	S1.2	S1.3	S1.4	S2.1	S2.2	S2.3	S2.4
Address bit:	6	5	4	3	2	1	0
Fieldbus address 127 (default)	1	1	1	1	1	1	1
Fieldbus address 25 (example)	0	0	1	1	0	0	1

Switch setting S4	Baud rate (Kbaud)
1	50
2	100
3	125
4	250
5	500
6	800
7	1000

Table 8.3 - Splitter box, CAN settings

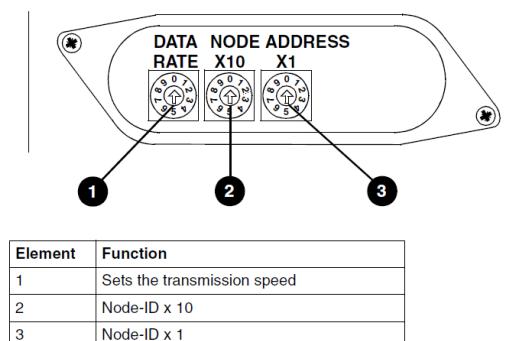


Figure 8.6 – Splitter box, CAN settings

Switch position	Transmission speed
0	Automatic recognition
1	10 Kbits/s
2	20 Kbits/s
3	50 Kbits/s
4	100 Kbits/s
5	125 Kbits/s
6	250 Kbits/s
7	500 Kbits/s
8	800 Kbits/s
9	1 Mbits/s

Each component using the CAN bus needs to have a unique node number and the baud rate must be configured to be same on all components (tbl. 8.4).

The CAN bus settings also need to be configured through the SoMachine software suit (fig. 8.7 – 8.9).

Table 8.4 - CAN bus settings for project

CAN node number	Component
3	Altivar_312
4	Lexium_ILA_1
5	Lexium_ILA_2
6	FTB1CN16CPO
127	TM238LFDC24DT
Baud Rate:	500.000

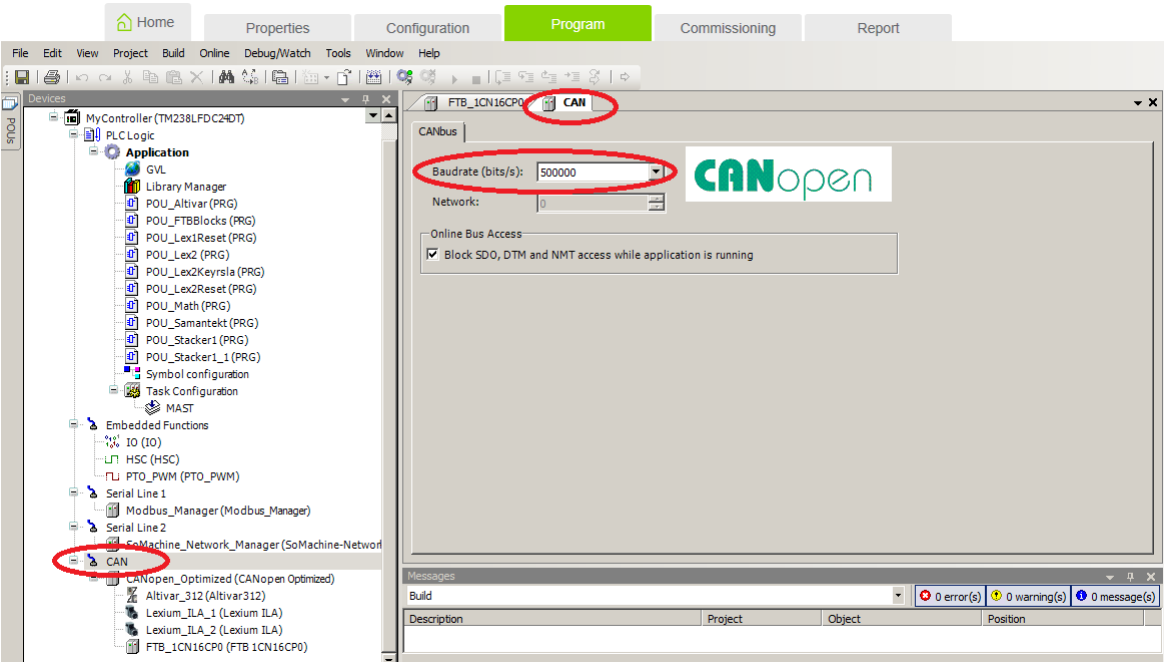


Figure 8.7 - CAN baud rate for project configured (SoMachine)

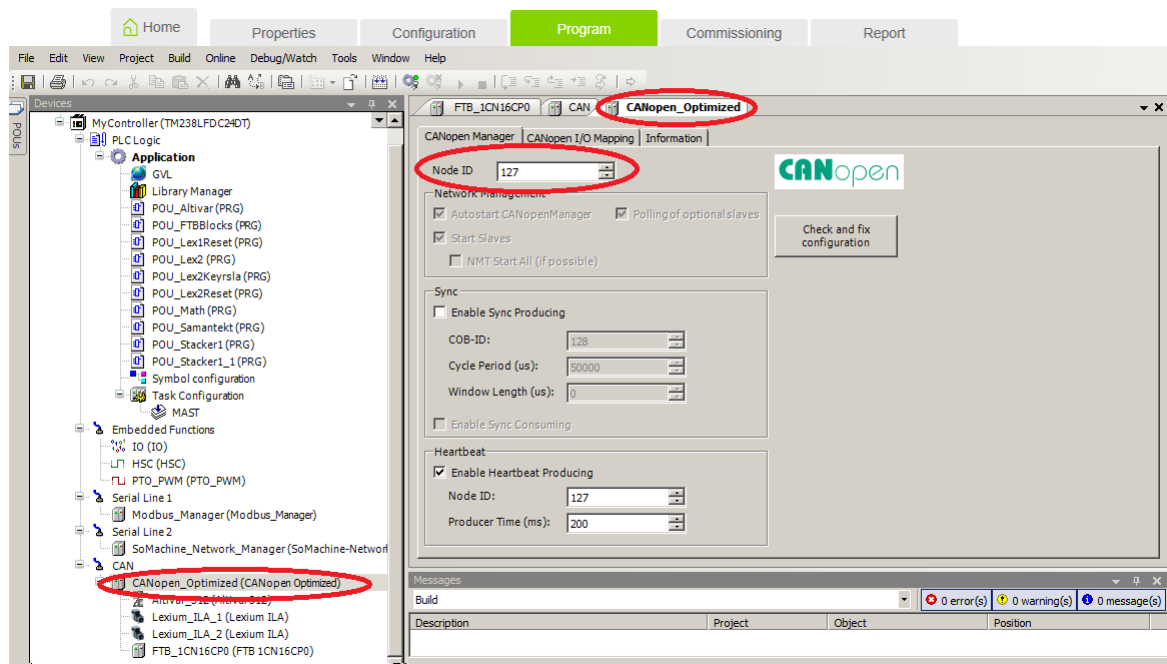


Figure 8.8 - CAN node setting for PLC (SoMachine)

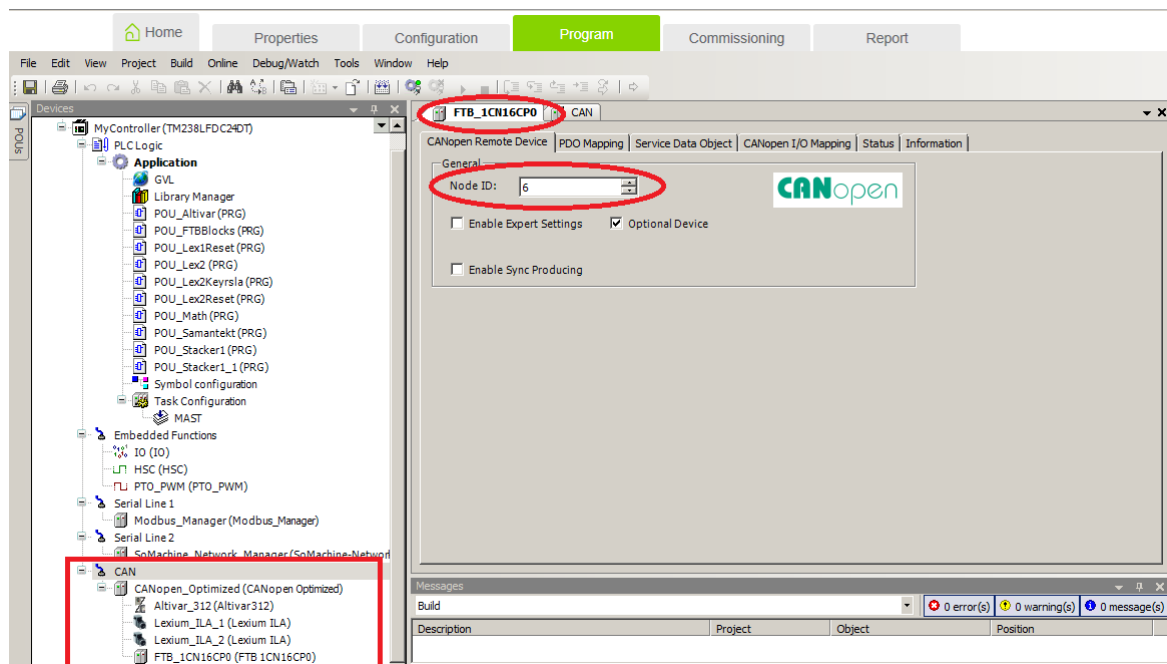


Figure 8.9 - CAN node setting for component in project (SoMachine)

With the CAN node and baud rate setting configured, both on components and in SoMachine, the CANopen protocol is able to run without errors, bringing all components used in the project online and therefore capable of communicating and interacting with the PLC.

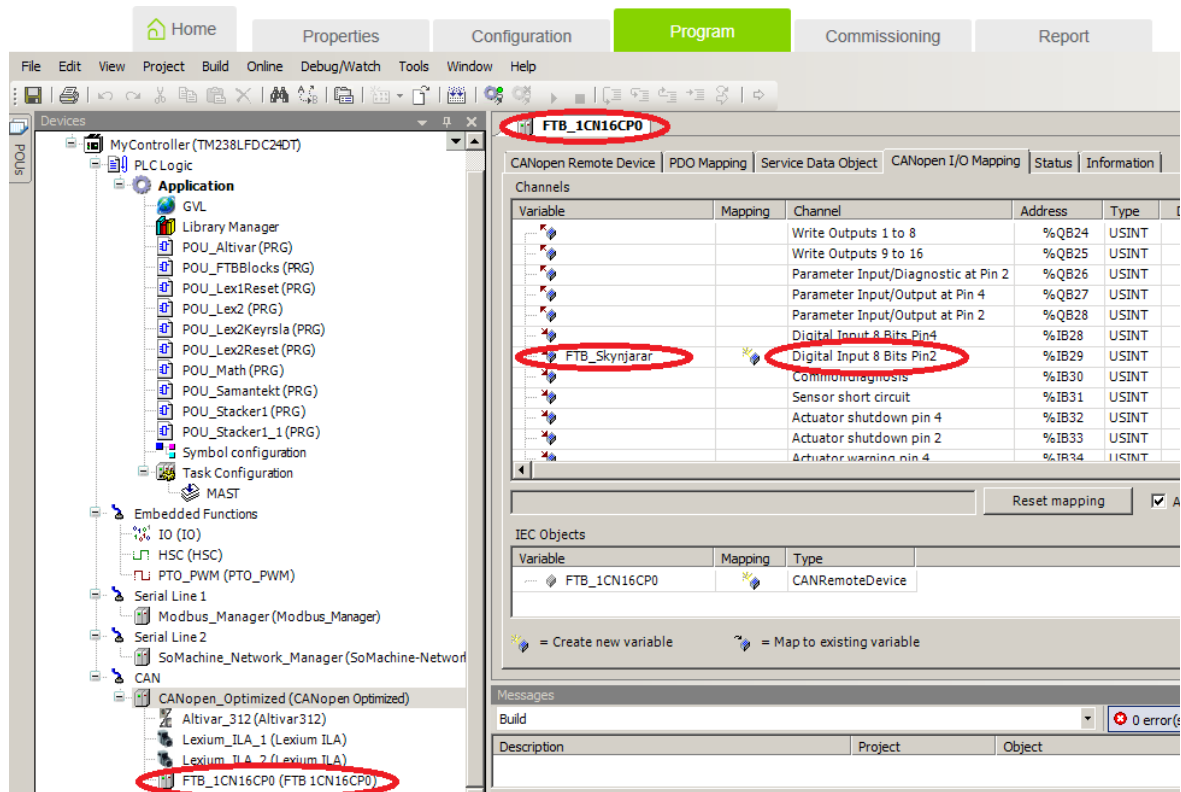


Figure 8.10 - CANopen I/O mapping for sensor variable

Before sensor inputs can be used in any of the programs for this project, they need to be located under the CANopen I/O mapping (fig. 8.10) for the splitter box (FTB1CN16CP0). They are registered under a channel named Digital Input 8 Bits Pin2. Creating a new variable connected to the channel, makes it accessible for use in programs created for this project.

The output of the channel is given as a byte (00000000 - 11111111), where each digit seat corresponds to a different sensor input (8 different connector for sensor inputs on FTB1CN16CP0), but the signal is interpreted as a single number (0-255). As all the inputs are registered 1 (on) when sensors are not triggered, the variable connected to the channel will return the value of 255. If sensor 0, sensor 2 and sensor 5 were to be triggered (fig. 8.11), the variable would return the value 218 (11011010). This makes it possible to implement different responses into the programs of the project, depending on the sensors triggered and the action required upon it.

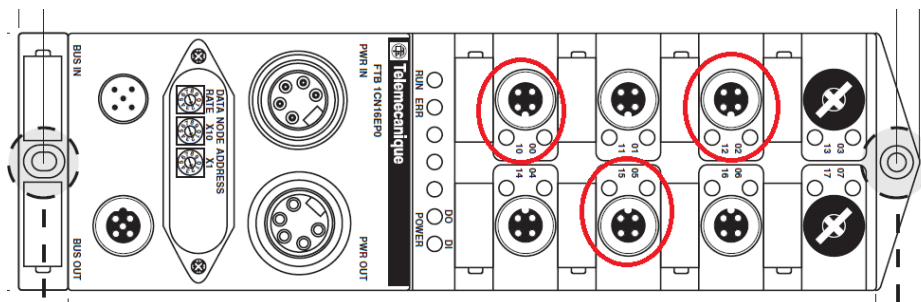


Figure 8.11 - Sensor 0, sensor 2 and sensor 5 triggered

Before any of the programs needed for the processing line were created, it was necessary to create temporary programs (see Appendix A, chapter A2.4) to test out the program blocks available, finding out their functions and limitations. Testing out and locating the necessary function blocks for the final programs is time consuming but returns its investment in the long run.

When the Lexium integrated drives start up, they do not know in which position they are, so in order for making the best use of their precise position capabilities, a program was made (see Appendix A, chapter A2.1) which makes the drive move till it hits the edge of its movements (in-build sensor trigger a kill switch for the drive), once it has located that position, it resets itself and sets that position as 0, therefore making it possible for later programs to correctly position the moving arm of the Lexium integrated drive into preordained positions. Thus it is possible to move in a set distance all the time, just like the distance of moving one tray up would require.

Once the Lexium integrated drives are in position 0, they are ready for the next program. The sensors on the splitter box trigger its activation. The 600 mm distance of the Lexium Linear Motion rail was split up into 6 different positions, (for testing purpose the height of each tray was set at 100 mm) made to simulate the off-stacking of 6 trays in a whole run of its length. A longer Linear Motion rail would have been needed for the simulation of 18 trays therefore it was tested by running it 3 times on these settings for timing measurements.

The program for the Stacker (see Appendix A, chapter A2.3) starts by lifting the stack high enough for the Y-movement unit to get under the topmost tray. Once the Y-movement unit is in position, the stack is lowered, leaving the top tray sitting on the Y-movement unit, it then moves with the tray towards its opposite end and deposits the tray there. The same movement set repeats 6 times at which point the X-movement arm will reach its top position. At that point the whole program will return back to its starting positions and request a new stack of trays for it to unstack. The program is reversed for the opposite effect (stacking of trays back into stacks).

The program (see Appendix A, chapter A2.2) for Section 3 of the processing line (see fig. 3.1) makes use of the Altivar variable speed drive to handle control. It is continuously on while Stacker 1 is putting trays on it, moving trays through the Scraper and Washer, towards Stacker 2. The speed settings for the conveyor are configurable through a physical turn dial located on the processing line, making it easy to change speed if required. It has triggering inputs from sensor to react upon if detects that the belt has no room for new trays. It also triggers the sensor input to start the Washer. After the Washer it notifies Stacker 2. that there are trays ready to be picked up and restacked.

The program for the conveyor belt that make use of relays for motor control, is in waiting position till it gets a signal from the Stacker, notifying that it is empty and requires a new stack of trays. If the program detects from its sensor input that there is a stack waiting, it will send it forward and stop once the Stacker unit tells it has received the stack and is now loaded again. The program is reversed for the opposite effect (Stacker 2 needs to be emptied and stacks are moved into buffer zone).

For this project, it had been planned to make use of a HMI touch panel screen, but due to an error that could not be resolved within the limited time of the project, it had to be disregarded.

9 Conclusions

The project was successful in demonstrating that electronic components selected for the project were capable of using the CANopen system as communication between themselves, therefore using fewer cables towards the control cabinet with the PLC. Also by using the same type of cable (CAN cable) between different component, it should make it easier to replace or add components later on.

With the equipment setup (fig. 8.1 and tbl. 8.1) it was possible to test out components and programs, simulating the functions (chapter 3.) for the processing line (fig. 1.1).

The testing and programming for the splitter box (FTB1CN16CPO) was successful, showing that it was possible to make use of it for the processing line in handling the sensory inputs.

The timing of the X and Y movement for the stacker was measured. The result when using the equipment at hand (fig. 8.1 and tbl. 8.1), showed that the stacker unit would be capable of processing 18 trays in 2 minutes and 15 seconds. This fulfills the requirement of processing 20 stacks of trays in 1 hour. However the final result is different, if the loading of new stacks into the stacker unit is taken into account and that the Y movement for the processing line is a longer distance than 600 mm (the Lexium Linear motion used in project was only 600 mm long). If the Y distance to travel would be increased to 1000 mm and the loading of a new stack of trays would take 15 seconds, it was calculated that it would take the stacker unit 4 minutes to process one stack of 18 trays, resulting in 80 minutes total to process 20 stacks of trays at current speed, which was 20 minutes over the requirement for this section.

The Lexium integrated drives (ILA1F572PCA0) selected in this project were not able to deliver the necessary speed with the settings used. Motors can be easily changed for more powerful and faster ones, and incorporated into existing machine structure and programming, as long as the same type of motor (Servo, AC, DC) with similar communication connections (CAN) are used in the recommended positions, preferably motors from the same manufacturer as used in project (Schneider Electric).

The electrical components, handling the controlling of inputs and responses (PLC, motor drives, splitter box and sensors), are capable of delivering the functions set out by project.

The use of a touchscreen panel for project was not accomplished due to an error that could not be resolved within the limited time of the project.

References

- [1] William Bolton, *Mechatronics*, 5th ed.: Pearson Education Limited, 2011.
- [2] Schneider Electronic. (2013, July) Magelis HMISTU655/855 User Manual. Pdf. [Online].
<http://www.schneider-electric.com/download/WW/EN/file/163031101-EIO0000000614.04.pdf/?showAsIframe=true&fileName=EIO0000000614.04.pdf&reference=EIO0000000614&docType=User-guide>
- [3] Schneider Electric. (2014, May) OEM machine programming software - SoMachine - A single software environment. [Online].
<http://www.schneider-electric.com/products/ww/en/5100-software/5140-pac-plc-programming-software/2226-oem-machine-programming-software-somachine/?XTMC=somachine&XTCR=2>
- [4] Schneider Electric. (2009, September) Advantys FTB CANopen IP67 Monobloc I/O Splitter box User guide. Pdf. [Online].
http://www.schneider-electric.com/download/WW/EN/file/27463201-1606218_02A55.pdf/?showAsIframe=true&fileName=1606218_02A55.pdf&reference=1606218_02A55&docType=User-guide

All the data and design about the mechanical side of the processing line was obtained from Gylfi Þór Guðlaugsson

All images, except for the photos (taken personally by author) were obtained from the home webpage of Schneider Electric (<http://www.schneider-electric.com>) or from datasheets and user manuals pertaining to components used in project which are published by Schneider Electric.

All images of function blocks and tables in Appendix A are obtained from SoMachine software suite programming library, SoMachine is a software program made by Schneider Electric.

Appendix A – Programming code

A1 Function Blocks

A1.1 ILX Function Blocks

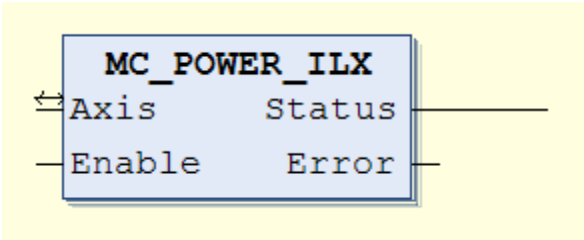


Figure A.1.1 - MC_Power_ILX

Table A.1.1 - MC_Power_ILX parameters

FUNCTION_BLOCK MC_Power_ILX			
Axis	Axis_Ref_ILX	VAR_IN_OUT	axis structure
Enable	BOOL	VAR_INPUT	FALSE: switch off, TRUE: switch on
Status	BOOL	VAR_OUTPUT	State of power amplifier, FALSE: switched off, TRUE: switched on
Error	BOOL	VAR_OUTPUT	error occured

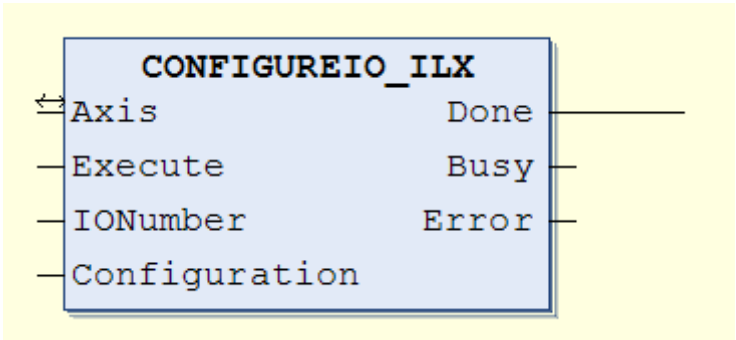


Figure A.1.2 - ConfigureIO_ILX

Table A.1.2 - ConfigureIO_ILX parameters

FUNCTION_BLOCK ConfigureIO_ILX			
Axis	Axis_Ref_ILX	VAR_IN_OUT	axis structure
Execute	BOOL	VAR_INPUT	rising edge starts writing configuration parameters
IONumber	UINT	VAR_INPUT	number of IO to configure (0=IO0, 1=IO1, 2=IO2, 3=IO3) 0 = input freely usable 1 = LIMP input (only configurable with IO0) 2 = LIMN input (only configurable with IO1) 3 = STOP input 4 = REF input 5 = input programmable 128 = output freely usable 129 = index pulse output (only configurable with IO0) 130 = output programmable
Configuration	UINT	VAR_INPUT	
Done	BOOL	VAR_OUTPUT	IO configured without error
Busy	BOOL	VAR_OUTPUT	busy
Error	BOOL	VAR_OUTPUT	error occurred

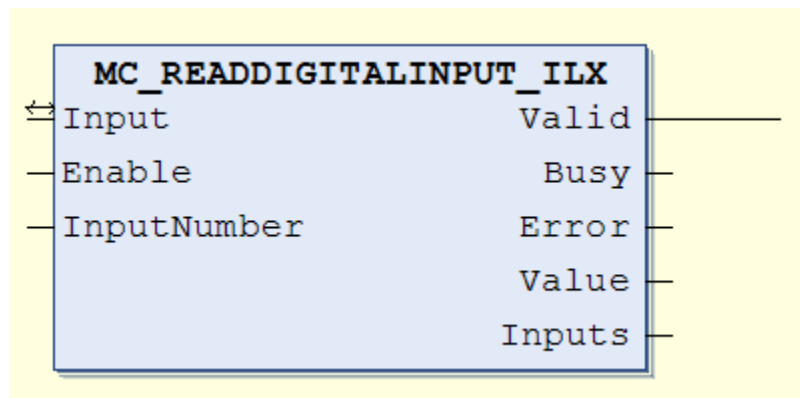


Figure A.1.3 - MC_Readdigitalinput_ILX

Table A.1.3 - MC_Readdigitalinput_ILX parameters

FUNCTION_BLOCK MC_ReadDigitalInput_ILX			
Input	Input_Ref_ILX	VAR_IN_OUT	axis structure
Enable	BOOL	VAR_INPUT	TRUE: returns the status of the inputs continously
InputNumber	INT	VAR_INPUT	selects the input (IO0=0, IO1=1, IO2=2, IO3=3)
Valid	BOOL	VAR_OUTPUT	input signal is available
Busy	BOOL	VAR_OUTPUT	busy
Error	BOOL	VAR_OUTPUT	error occurred
Value	BOOL	VAR_OUTPUT	value of the selected input signal
Inputs	WORD	VAR_OUTPUT	value of all inputs (Inputs.0 = IO0, ..., Inputs.3 = IO3)

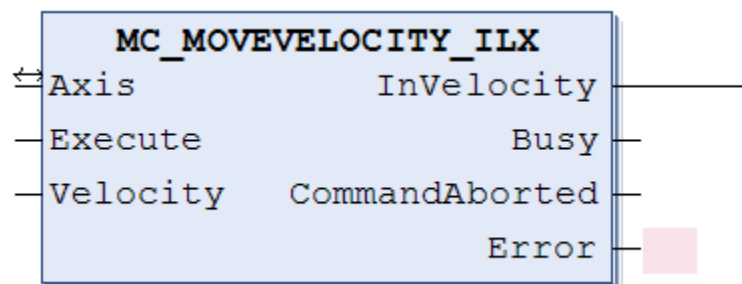


Figure A.1.4 - MC_Movevelocity_ILX

Table A.1.4 - MC_Movevelocity_ILX parameters

FUNCTION_BLOCK MC_MoveVelocity_ILX			
Axis	Axis_Ref_ILX	VAR_IN_OUT	axis structure
Execute	BOOL	VAR_INPUT	rising edge starts motion
Velocity	INT	VAR_INPUT	target velocity of the motion [rpm]
InVelocity	BOOL	VAR_OUTPUT	commanded velocity reached
Busy	BOOL	VAR_OUTPUT	busy
CommandAborted	BOOL	VAR_OUTPUT	FB was aborted by another command
Error	BOOL	VAR_OUTPUT	error occurred

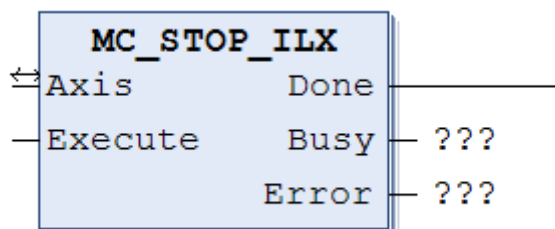


Figure A.1.5 - MC_Stop_ILX

Table A.1.5 - MC_Stop_ILX parameters

FUNCTION_BLOCK MC_Stop_ILX			
Axis	Axis_Ref_ILX	VAR_IN_OUT	axis structure
Execute	BOOL	VAR_INPUT	rising edge starts execution
Done	BOOL	VAR_OUTPUT	done without error
Busy	BOOL	VAR_OUTPUT	busy
Error	BOOL	VAR_OUTPUT	error occurred

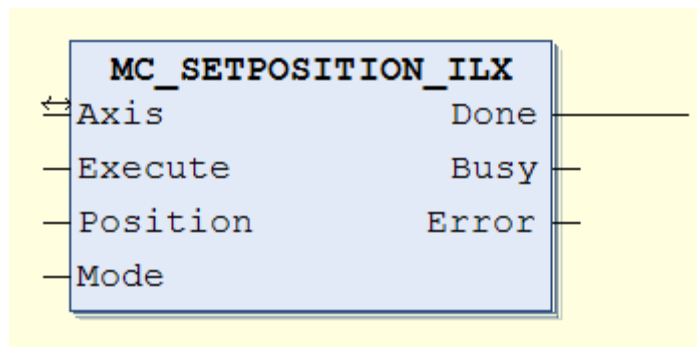


Figure A.1.6 - MC_Setposition_ILX

Table A.1.6 - MC_Setposition_ILX parameters

FUNCTION_BLOCK MC_SetPosition_ILX			
Axis	Axis_Ref_ILX	VAR_IN_OUT	axis structure
Execute	BOOL	VAR_INPUT	rising edge starts setting new position
Position	DINT	VAR_INPUT	position [Inc]
Mode	BOOL	VAR_INPUT	FALSE: absolute position, TRUE: relative to actual motor position
Done	BOOL	VAR_OUTPUT	set new position
Busy	BOOL	VAR_OUTPUT	busy
Error	BOOL	VAR_OUTPUT	error occurred

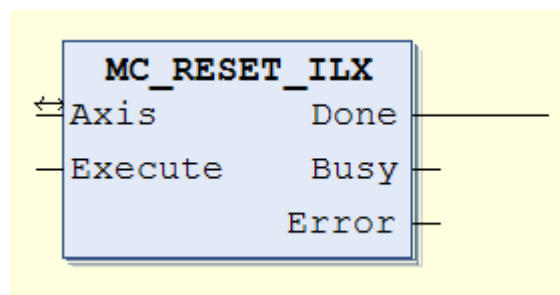


Figure A.1.7 - MC_Reset_ILX

Table A.1.7 - MC_Reset_ILX parameters

FUNCTION_BLOCK MC_Reset_ILX			
Axis	Axis_Ref_ILX	VAR_IN_OUT	axis structure
Execute	BOOL	VAR_INPUT	rising edge starts execution
Done	BOOL	VAR_OUTPUT	done without error
Busy	BOOL	VAR_OUTPUT	busy
Error	BOOL	VAR_OUTPUT	error occurred

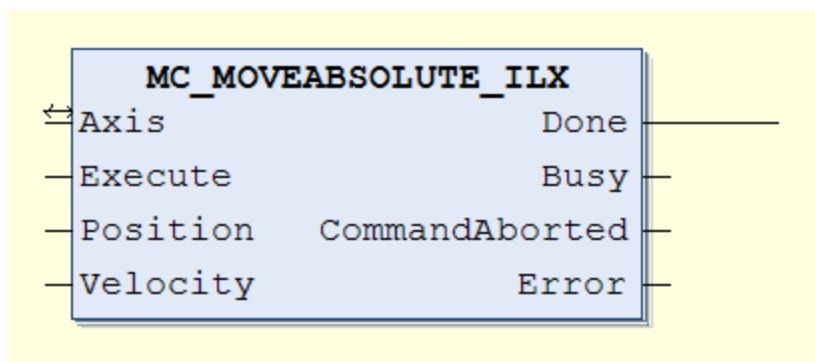


Figure A.1.8 - MC_Moveabsolute_ILX

Table A.1.8 - MC_Moveabsolute_ILX parameters

FUNCTION_BLOCK MC_MoveAbsolute_ILX			
Axis	Axis_Ref_ILX	VAR_IN_OUT	axis structure
Execute	BOOL	VAR_INPUT	rising edge starts motion
Position	DINT	VAR_INPUT	target position of movement [Inc]
Velocity	INT	VAR_INPUT	target velocity of movement [rpm]
Done	BOOL	VAR_OUTPUT	commanded position reached
Busy	BOOL	VAR_OUTPUT	busy
CommandAborted	BOOL	VAR_OUTPUT	FB was aborted by another command
Error	BOOL	VAR_OUTPUT	error occurred

A1.2 ATV Function Blocks

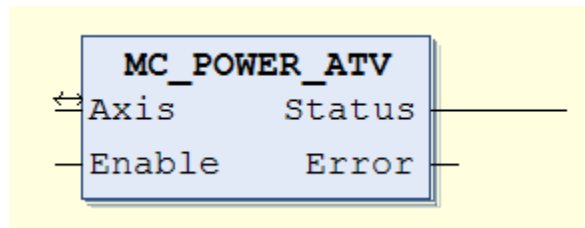


Figure A.1.9 - MC_Power_ATV

Table A.1.9 - MC_Power_ATV parameters

FUNCTION_BLOCK MC_Power_ATV			
Axis	Axis_Ref_ATV	VAR_IN_OUT	axis structure
Enable	BOOL	VAR_INPUT	FALSE: switch off, TRUE: switch on
Status	BOOL	VAR_OUTPUT	State of power amplifier, FALSE: switched off, TRUE: switched on
Error	BOOL	VAR_OUTPUT	error occurred

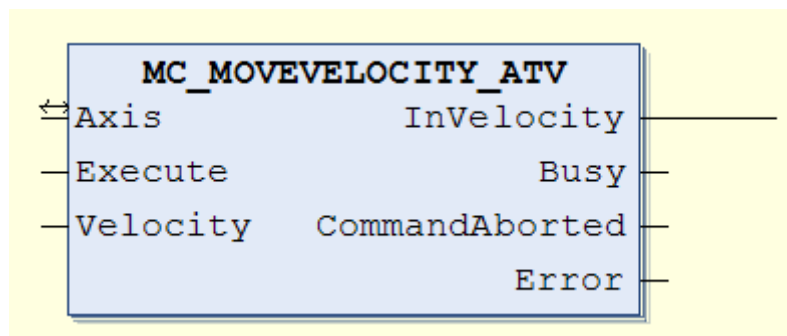


Figure A.1.10 - Movevelocity_ATV

Table A.1.10 - Movevelocity_ATV parameters

FUNCTION_BLOCK MC_MoveVelocity_ATV			
Axis	Axis_Ref_ATV	VAR_IN_OUT	axis structure
Execute	BOOL	VAR_INPUT	rising edge starts motion
Velocity	INT	VAR_INPUT	target velocity of movement: frequency (-5000 ... 5000) [0.1 Hz]
InVelocity	BOOL	VAR_OUTPUT	commanded velocity reached
Busy	BOOL	VAR_OUTPUT	busy

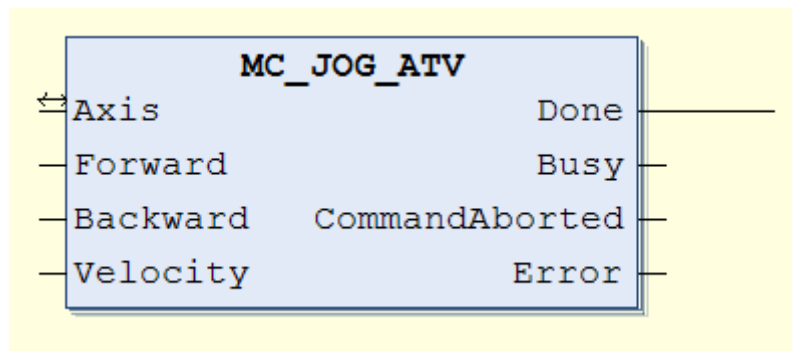


Figure A.1.11 - MC_Jog_ATV

Table A.1.11 - MC_Jog_ATV parameters

FUNCTION_BLOCK MC_Jog_ATV			
Axis	Axis_Ref_ATV	VAR_IN_OUT	axis structure
Forward	BOOL	VAR_INPUT	start the jogged motion in positive direction (clockwise)
Backward	BOOL	VAR_INPUT	start the jogged motion in negative direction (counterclockwise)
Velocity	INT	VAR_INPUT	target velocity of movement: frequency (-5000 ... 5000) [0.1 Hz]
Done	BOOL	VAR_OUTPUT	done without error
Busy	BOOL	VAR_OUTPUT	busy
CommandAborted	BOOL	VAR_OUTPUT	FB was aborted by another command
Error	BOOL	VAR_OUTPUT	error occurred

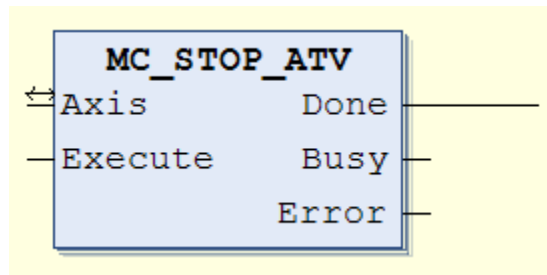


Figure A.1.12 - MC_Stop_ATV

Table A.1.12 - MC_Stop_ATV parameters

FUNCTION_BLOCK MC_Stop_ATV			
Axis	Axis_Ref_ATV	VAR_IN_OUT	axis structure
Execute	BOOL	VAR_INPUT	rising edge starts execution
Done	BOOL	VAR_OUTPUT	done without error
Busy	BOOL	VAR_OUTPUT	busy
Error	BOOL	VAR_OUTPUT	error occurred

A1.3 Other Function Blocks

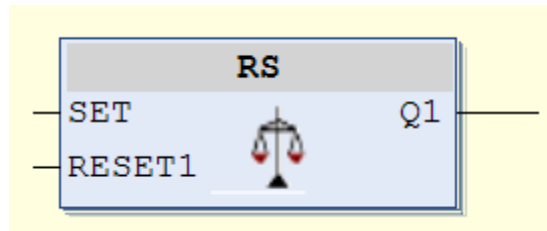


Figure A.1.13 - RS

Table A.1.13 - RS parameters

FUNCTION_BLOCK RS			
SET	BOOL	VAR_INPUT	Input to set Q1
RESET1	BOOL	VAR_INPUT	Input to reset Q1 (reset dominant)
Q1	BOOL	VAR_OUTPUT	

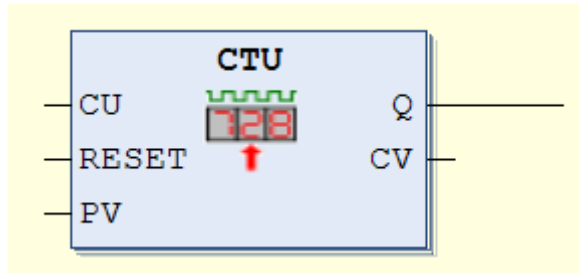


Figure A.1.14 - CTU

Table A.1.14 - CTU parameters

FUNCTION_BLOCK CTU			
CU	BOOL	VAR_INPUT	Count Up
RESET	BOOL	VAR_INPUT	Reset Counter to 0
PV	WORD	VAR_INPUT	Counter Limit
Q	BOOL	VAR_OUTPUT	Counter reached the Limit
CV	WORD	VAR_OUTPUT	Current Counter Value

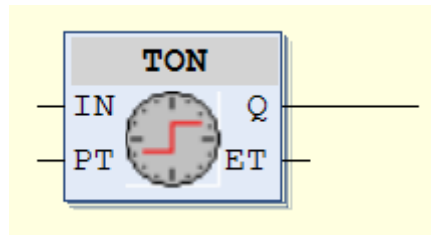


Figure A.1.15 - TON

Table A.1.15 - TON parameters

FUNCTION_BLOCK TON			
IN	BOOL	VAR_INPUT	starts timer with rising edge, resets timer with falling edge
PT	TIME	VAR_INPUT	time to pass, before Q is set
Q	BOOL	VAR_OUTPUT	gets TRUE delay time after a rising edge at IN
ET	TIME	VAR_OUTPUT	elapsed time since rising edge at IN

A2 Programs

A2.1 Lexium drives reset and set position

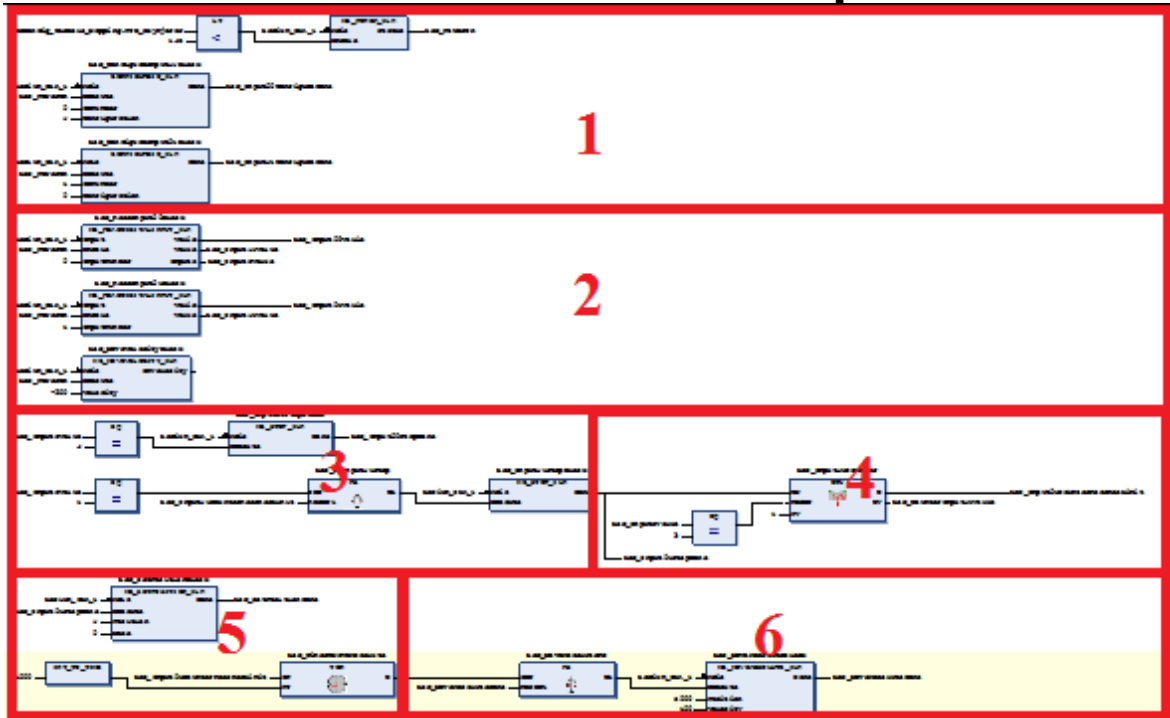


Figure A.2.1 - Lexium drives reset and set position

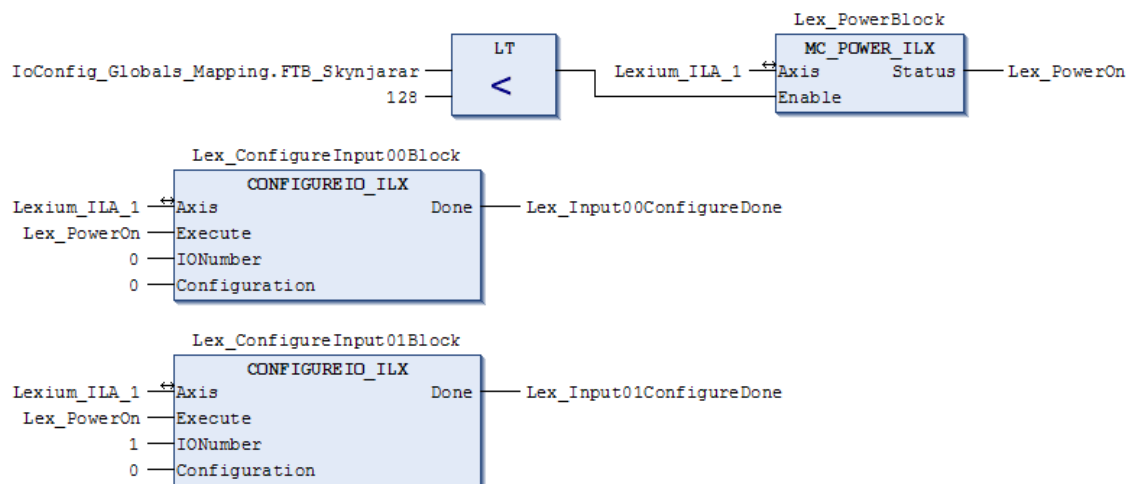


Figure A.2. 2 - Lexium drives reset and set position, part 1

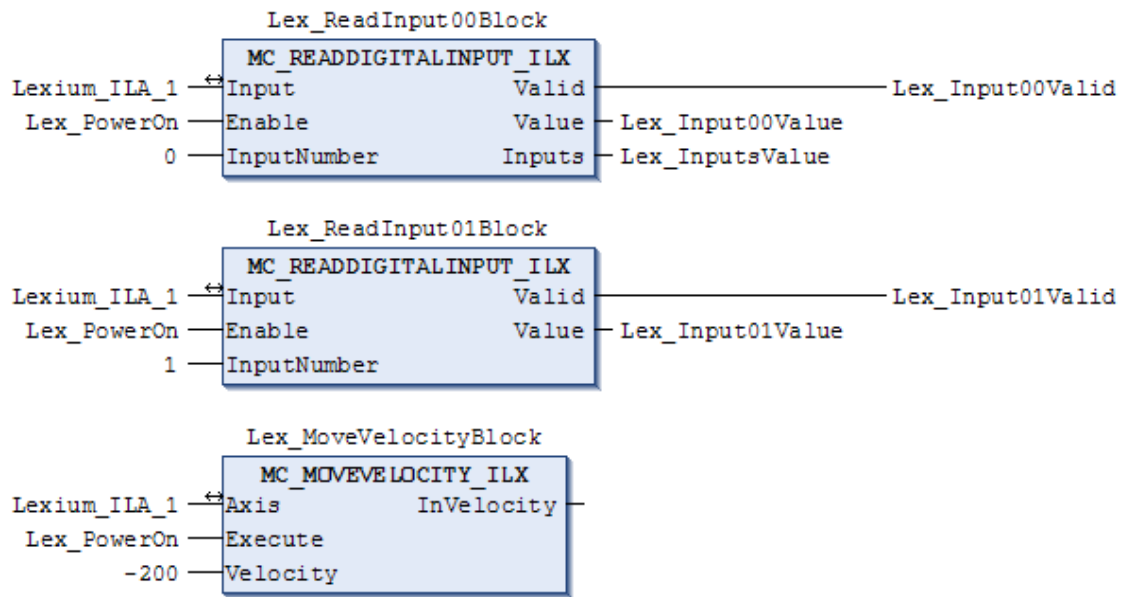


Figure A.2.3 - Lexium drives reset and set position, part 2

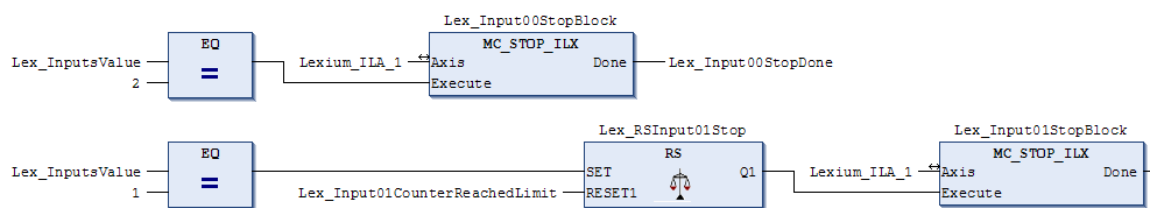


Figure A.2.4 - Lexium drives reset and set position, part 3

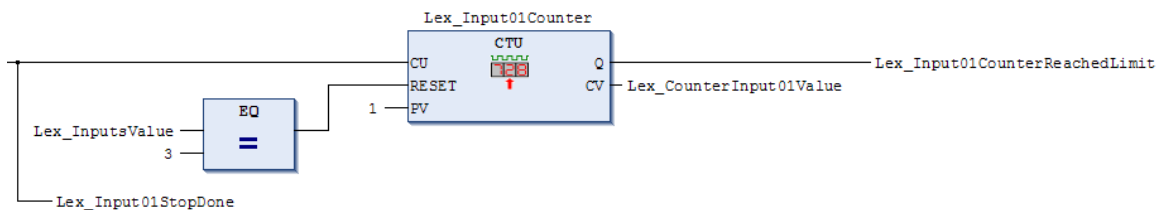


Figure A.2.5 - Lexium drives reset and set position, part 4

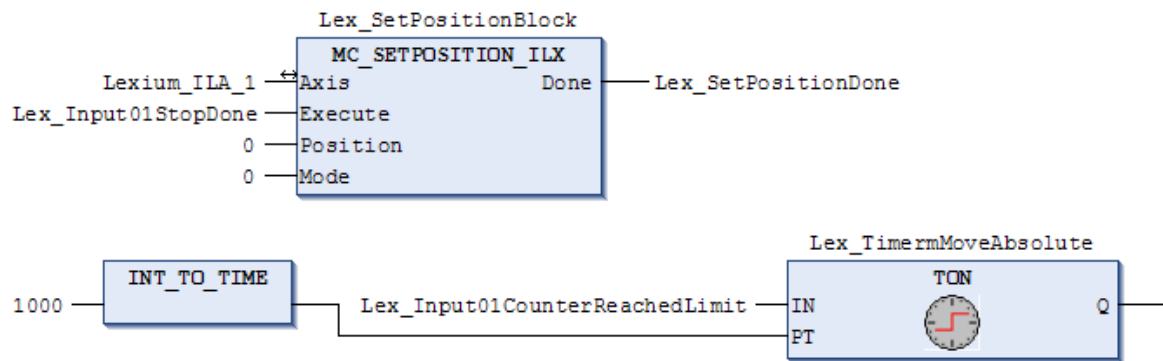


Figure A.2.6 - Lexium drives reset and set position, part 5

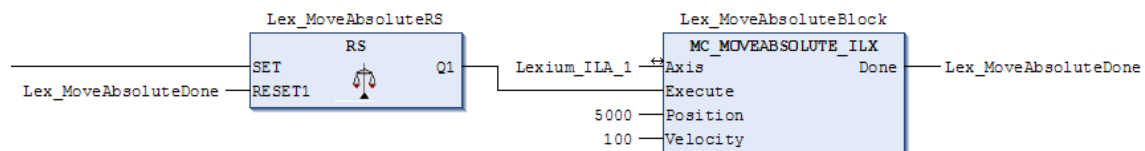


Figure A.2.7 - Lexium drives reset and set position, part 6

A2.2 Altivar drive

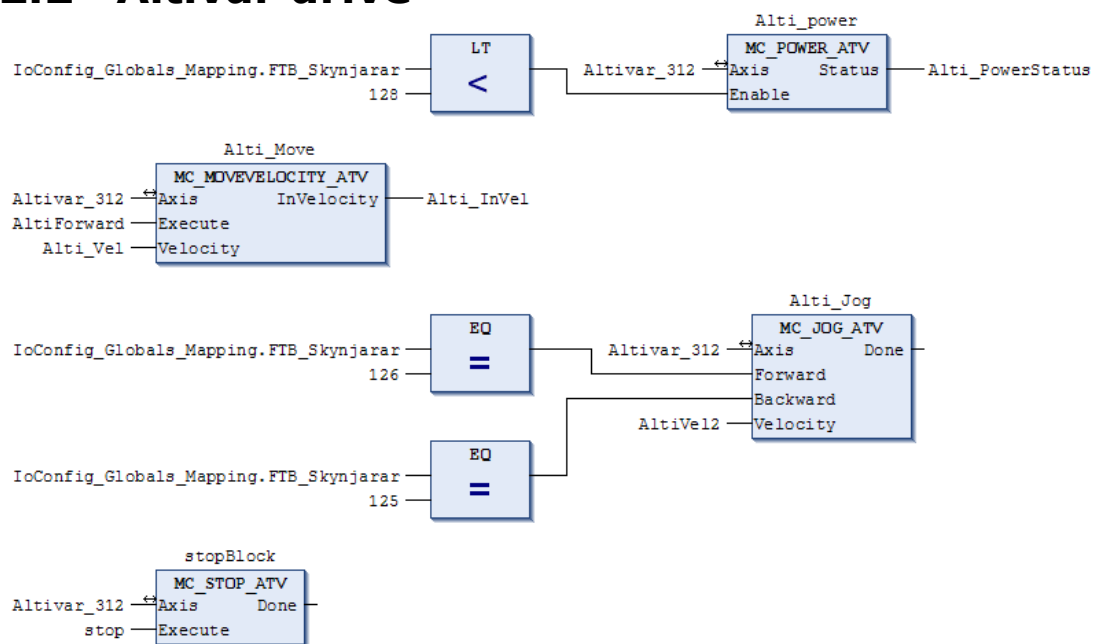


Figure A.2.8 - Altivar drive

A2.3 Stacker program

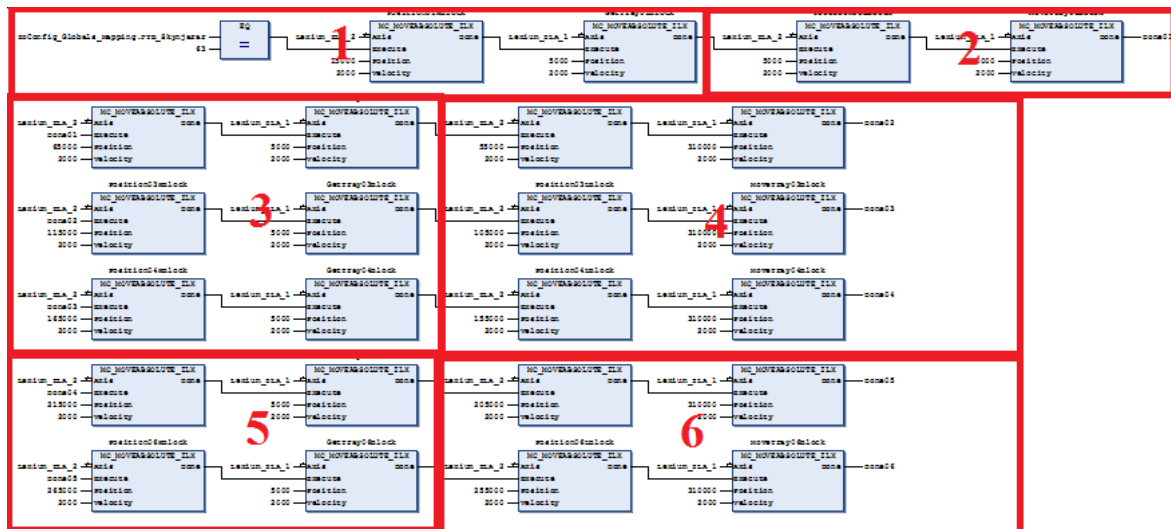


Figure A.2.9 - Stacker program

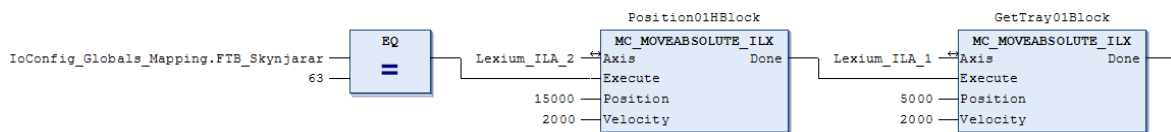


Figure A.2.10 - Stacker program, part 1

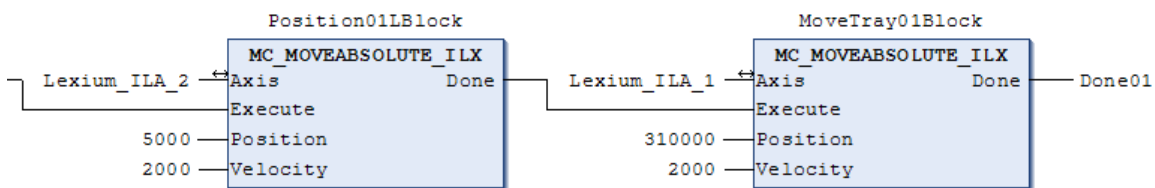


Figure A.2.11 - Stacker program, part 2

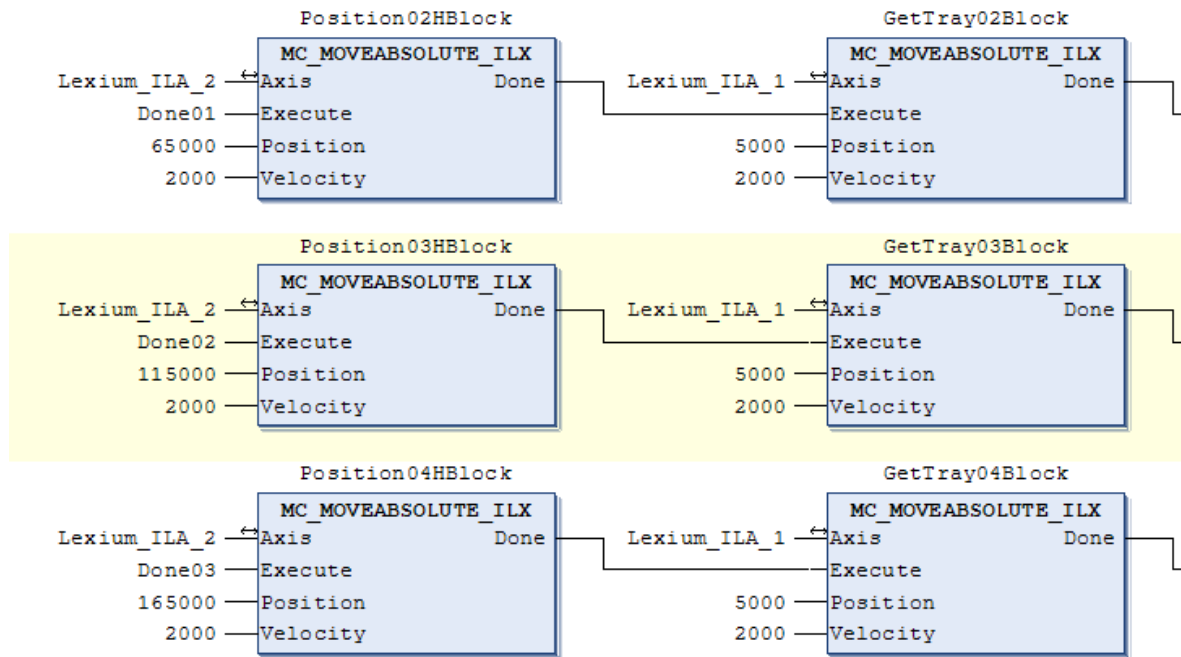


Figure A.2.12 - Stacker program, part 3

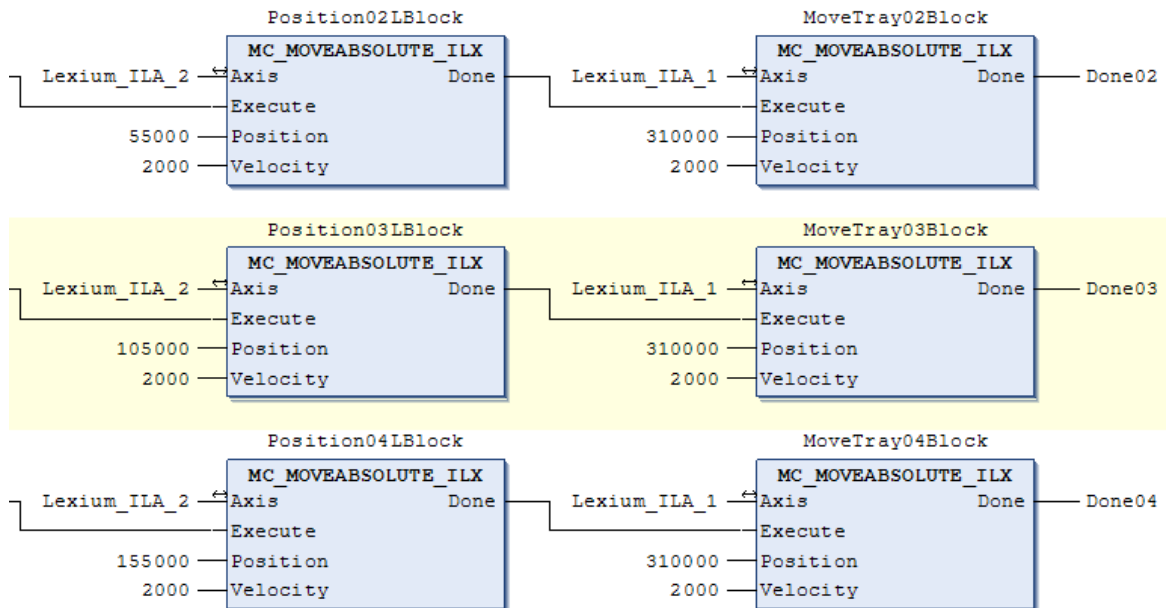


Figure A.2.13 - Stacker program, part 4

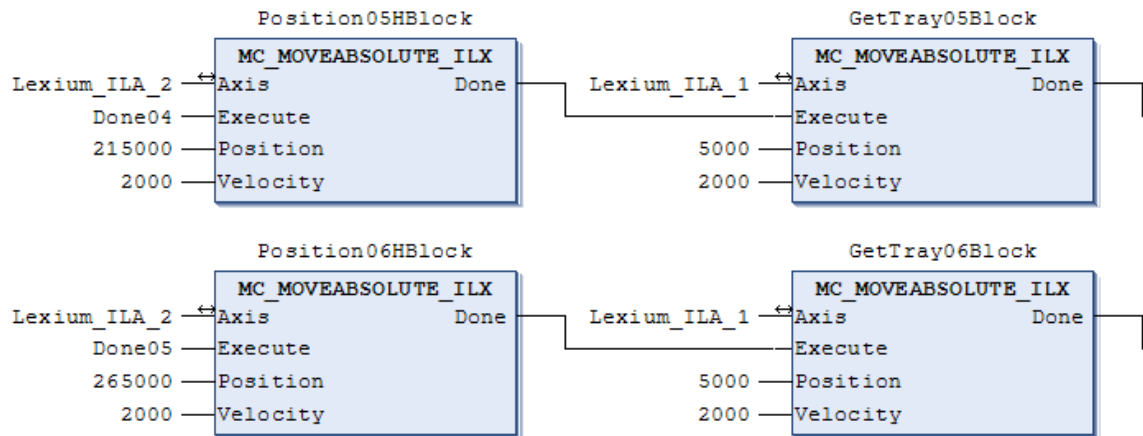


Figure A.2.14 - Stacker program, part 6

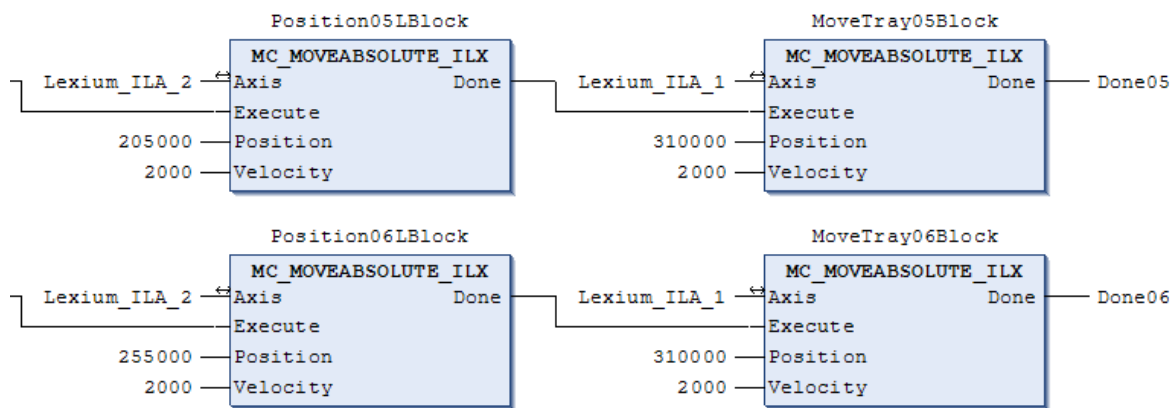


Figure A.2.15 - Stacker program, part 6

A2.4 Testing function blocks for ILX drive

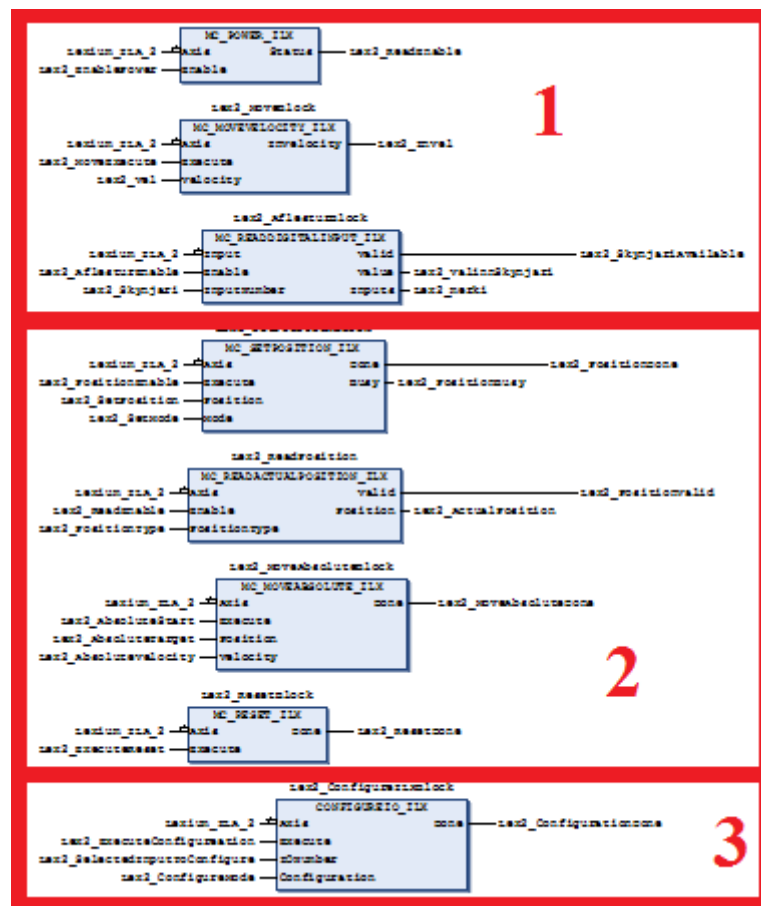


Figure A.2.16 - Testing function blocks for ILX drive

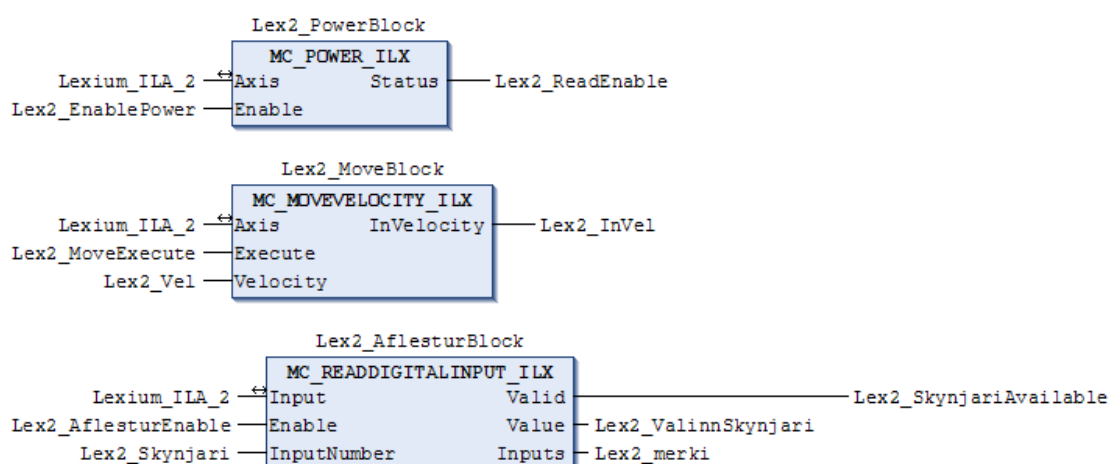


Figure A.2.17 - Testing function blocks for ILX drive, part 1

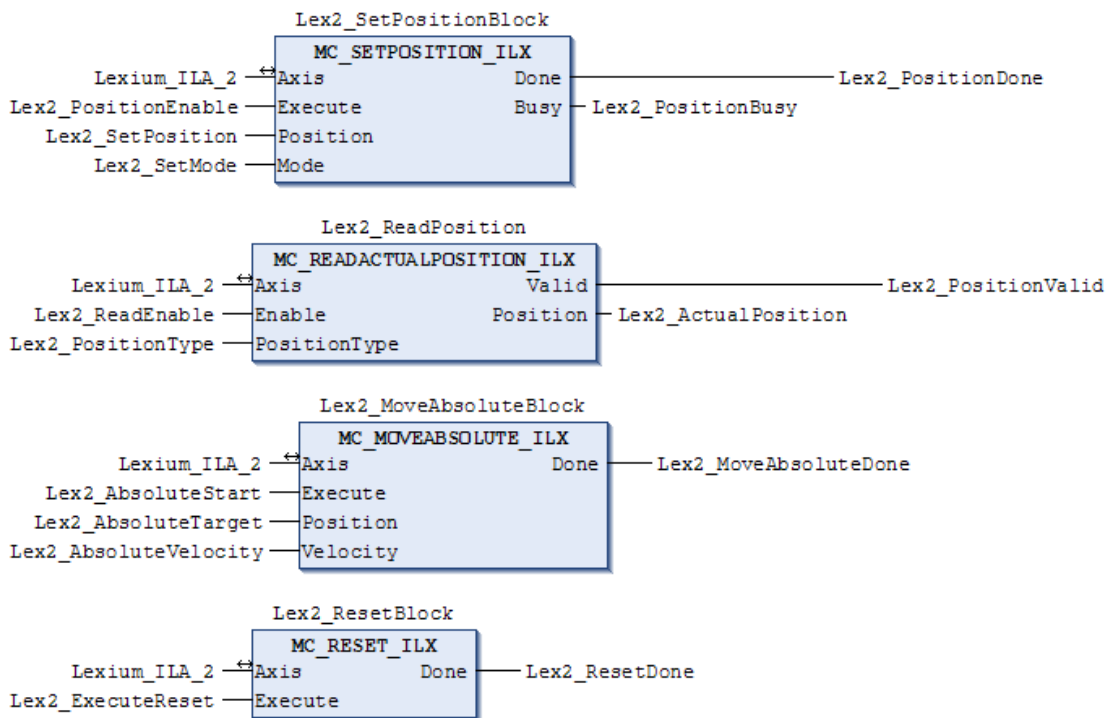


Figure A.2.18 - Testing function blocks for ILX drive, part 2

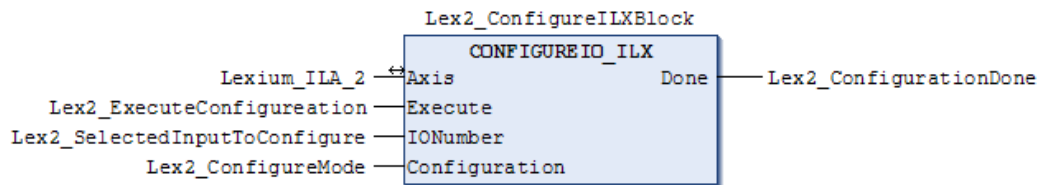
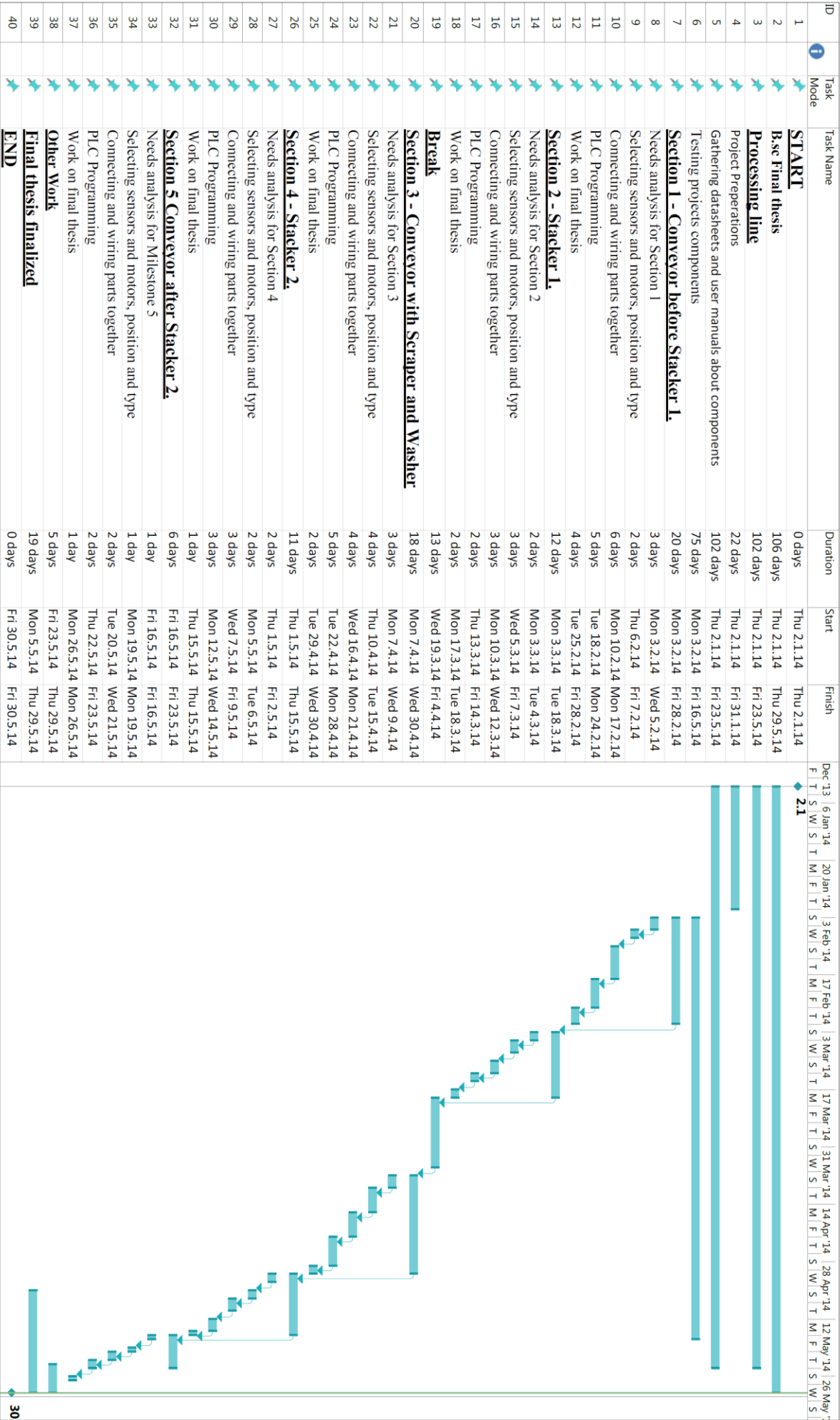


Figure A.2.19 - Testing function blocks for ILX drive, part 3

Appendix B – Project plan



Appendix C – Datasheets web links

Web links to user guides, manuals, datasheets and information concerning the components used in project.

Web links accessed and online in May 2014

SoMachine Programming Guide

Schneider Electronic. (2014, May) SoMachine Programming Guide. Pdf [Online]
https://stevenengineering.com/tech_support/PDFs/45MANUAL_SOMACHINE-PROGRAM.pdf

SoMachine Introduction

Schneider Electronic. (2014, May) SoMachine Introduction. Pdf [Online]
http://stevenengineering.com/tech_support/PDFs/45SOFTWARE_SOMACHINE.pdf

Adventys FTB CANopen IP67 Monobloc I/O Splitter box User guide

Schneider Electronic. (2014, May) Adventys FTB CANopen IP67 Monobloc I/O Splitter box User guide. Pdf [Online]
http://www.schneider-electric.com/download/WW/EN/file/27463201-1606218_02A55.pdf/?showAsIframe=true&fileName=1606218_02A55.pdf&reference=1606218_02A55&docType=User-guide

IP 67 monobloc I/O splitter boxes for fieldbuses Advantys, FTB splitter boxes

Schneider Electronic. (2014, May) IP 67 monobloc I/O splitter boxes for fieldbuses Advantys, FTB splitter boxes. Pdf [Online]
http://msavtomatika.com.ua/sites/default/files/doc/schneider_electric/2_promyshlennye_kontrollery_i_paneli_operatora/promyshlennye_kontrollery/moduli_udalennogo_vvoda_vyvoda_advantys/ftb/advantys_ftb_katalog_eng.pdf

ATV312 – Programming manual

Schneider Electronic. (2014, May) ATV312 – Programming manual. Pdf [Online]
http://www.schneider-electric.com/download/WW/EN/file/27530066-ATV312_programming_manual_EN_BBV46385_02.pdf/?showAsIframe=true&fileName=ATV312_programming_manual_EN_BBV46385_02.pdf&reference=BBV46385&docType=User-guide

ATV312 – CANopen manual

Schneider Electronic. (2014, May) ATV312 – CANopen manual. Pdf [Online]
http://www.schneider-electric.com/download/WW/EN/file/27502303-ATV312_CANopen_manual_BBV52819_02.pdf/?showAsIframe=true&fileName=ATV312_CANopen_manual_BBV52819_02.pdf&reference=BBV52819&docType=User-guide

ATV312 – Installation manual

Schneider Electronic. (2014, May) ATV312 – Installation manual. Pdf [Online]
http://www.schneider-electric.com/download/WW/EN/file/27501445-ATV312_Installation_manual_EN_BBV46391_01.pdf?showAsIframe=true&fileName=ATV312_Installation_manual_EN_BBV46391_01.pdf&reference=BBV46391&docType=User-guide

Magelis HMISTU655/855 - User Manual

Schneider Electronic. (2014, May) Magelis HMISTU655/855 - User Manual. Pdf [Online]
<http://www.schneider-electric.com/download/WW/EN/file/163031101-EIO0000000614.04.pdf?showAsIframe=true&fileName=EIO0000000614.04.pdf&reference=EIO0000000614&docType=User-guide>

Modicon M238 Logic Controller - Programming Guide

Schneider Electronic. (2014, May) Modicon M238 Logic Controller - Programming Guide. Pdf [Online]
http://download.schneider-electric.com/files?p_Reference=EIO0000000384&p_EnDocType=User%20guide&p_FileId=455436187&p_FileName=EIO0000000384.06.pdf

M238 Logic Controller - Hardware Guide

Schneider Electronic. (2014, May) M238 Logic Controller - Hardware Guide. Pdf [Online]
<http://www.schneider-electric.com/download/WW/EN/file/455436167-EIO0000000016.07.pdf?showAsIframe=true&fileName=EIO0000000016.07.pdf&reference=EIO0000000016&docType=User-guide>

Lexium ILx Field Bus Manual - CANopen DS301

Schneider Electronic. (2014, May) Lexium ILx Field Bus Manual - CANopen DS301. Pdf [Online]
http://download.schneider-electric.com/files?p_Reference=ILx1F_CANopenDS301_manual_V201_EN&p_EnDocType=User%20guide&p_FileId=27549306&p_FileName=ILx1F_CANopenDS301_manual_V201_EN.pdf

ILX Library Function blocks - Software manual

Schneider Electronic. (2014, May) ILX Library Function blocks - Software manual. Pdf [Online]
http://www.schneider-electric.com/download/WW/EN/file/27639191-ILX_FB_Manual_V209_EN.pdf?showAsIframe=true&fileName=ILX_FB_Manual_V209_EN.pdf&reference=0198441113886-EN&docType=User-guide

ILA1B, ILA1F, ILA1R Lexium Integrated Drive - Product manual

Schneider Electronic. (2014, May) ILA1B, ILA1F, ILA1R Lexium Integrated Drive - Product manual. Pdf [Online]

[http://www.schneider-electric.com/download/WW/EN/file/27534461-](http://www.schneider-electric.com/download/WW/EN/file/27534461-ILA1B_ILA1F_ILA1R_manual_V200_logo-EN.pdf/?showAsIframe=true&fileName=ILA1B_ILA1F_ILA1R_manual_V200_logo-EN.pdf&reference=0198441113562&docType=User-guide)

[ILA1B_ILA1F_ILA1R_manual_V200_logo-](http://www.schneider-electric.com/download/WW/EN/file/27534461-ILA1B_ILA1F_ILA1R_manual_V200_logo-EN.pdf/?showAsIframe=true&fileName=ILA1B_ILA1F_ILA1R_manual_V200_logo-EN.pdf&reference=0198441113562&docType=User-guide)

[EN.pdf/?showAsIframe=true&fileName=ILA1B_ILA1F_ILA1R_manual_V200_logo-](http://www.schneider-electric.com/download/WW/EN/file/27534461-ILA1B_ILA1F_ILA1R_manual_V200_logo-EN.pdf/?showAsIframe=true&fileName=ILA1B_ILA1F_ILA1R_manual_V200_logo-EN.pdf&reference=0198441113562&docType=User-guide)

[EN.pdf&reference=0198441113562&docType=User-guide](http://www.schneider-electric.com/download/WW/EN/file/27534461-ILA1B_ILA1F_ILA1R_manual_V200_logo-EN.pdf/?showAsIframe=true&fileName=ILA1B_ILA1F_ILA1R_manual_V200_logo-EN.pdf&reference=0198441113562&docType=User-guide)