NTNU
Norwegian University of
Science and Technology

Automated Aluminum Production Cell

BACHELOR THESIS IE303612 Automation

By

Fredrik Aarset Ryslett (10045), Fredrik Johannes Bakker (10046)
and Sondre Grevle Iveland (10028)

Number of pages: 432

Delivery date: 01.06.2018

Department of ICT and Natural Sciences
Norwegian University of Science and Technology

Supervisor 1: Ottar L. Osen
Supervisor 2: Mikael Tollefsen

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. **Manglende erklæring fritar ikke studentene fra sitt ansvar**.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☑
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☑
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7.**	☑
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert**	☑
5.	**Jeg/vi er kjent med at NTNU vil behandle alle saker hvor det foreligger mistanke om fusk.**	☑
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☑

Publiseringsavtale

Studiepoeng: 20

Veileder: Ottar L. Osen, Mikael Tollefsen

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☑ **ja**	☐ **nei**
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut)	☐ **ja**	☑ **nei**
- Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	☐ **ja**	☐ **nei**
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13	☐ **ja**	☑ **nei**

Dato:

30.05.2018

Preface

This bachelor thesis is written by three automation students at NTNU in Ålesund, and concludes three years of engineering studies during the spring semester of 2018. The thesis originates from earlier work done by one of the graduate students as an internship in RUFO AS during the autumn semester of 2017. This internship led to RUFO contacting Viddal Automation to set the initial conditions for an automated production cell which machines aluminum. The thesis is provided as a collaboration between Viddal Automation and RUFO AS, were RUFO is the customer and Viddal Automation is the thesis principal.

We would like to thank supervisors and contributors for their continuous support and cooperation throughout the thesis work:

- Contributors at RUFO:
  - Knut Steinnes, CEO RUFO AS
  - Arnfinn Velle, Product developer RUFO AS
  - Anders Steinnes, Product designer RUFO AS
  - Bottolf Lødemel, Product designer RUFO AS
  - Jo Sindre Ørstavik, Production manager RUFO AS
- Supervisors:
  - Ottar L. Osen, NTNU Ålesund
  - Mikael Tollefsen, NTNU Ålesund
- Contributors at Viddal Automation:
  - Steffen Viddal, CEO Viddal Automation AS

To fully comprehend the the thesis content, it is recommended that the reader has a technical background.

Enjoy!

Executive Summary

The following bachelor thesis *Automated Aluminum Production Cell* is a project for RUFO AS, provided by Viddal Automation AS, in which the purpose is to develop the electrical- and control system for the production cell. On March 19, 2018 we had a meeting with Viddal Automation discussing the thesis and project progress. It became apparent that neither the physical machine nor the design would be finished in time, as a consequence construction of a prototype was added to the thesis in order to ensure testing of the fully developed control software. Through testing it became apparent that the control software fulfill the requirements completely. However the preliminary model, due to rapid prototyping, has a larger margin of error than specified by RUFO.
In order to illustrate the entire workflow the video presentation was recorded and uploaded to https://www.youtube.com/watch?v=AirGtP0qUXM.

Terminology

Symbols

π	Mathematical constant $\pi = 3.1415$

Abbreviations

PLC	Programmable Logical Controller
ST	Structured text
SFC	Sequential function chart
IDE	Integrated Development Environment
DoF	Degrees of freedom
KRL	KUKA Robot Language
HMI	Human machine interface
GUI	Graphical user interface
BOM	Bill of material
.CSV	Comma separated values
LIN	Command for linear motion
PTP	Command for point to point motion
TCP	Tool center point

List of Figures

Contents

Chapter 1

Introduction

Companies homesourcing their production back to Norway, has been a recurring event mentioned in the media the last 3-4 years. The articles highlights the idea that commitment to industrial 4.0 standard and high degree of automation in manufacturing makes it beneficial to resume Norwegian production. Automation does not only allow Norwegian companies to compete in the international market, but also creates a variety of job prospects which in return provide the employees with better job security and more inventive and ingenious objectives, as well as larger salaries. (Ekornseater, 2016), (Seehusen, 2017)

In virtue of activities performed by local companies, willing to relocate the production back to Norway, automation has become a flourishing sector, that created favorable conditions and demand for the bachelor thesis being presented. Viddal Automation AS is a prime example of a growing business benefiting from such companies by providing custom automated production solutions to manufacturing companies. At the same time Viddal Automation is client and industrial supervisor for this bachelor thesis.

In its turn, the thesis facilitates a bigger ongoing development project, performed by Viddal Automation for a local company named RUFO AS. The thesis covers the design of the electrical and software systems for an isolated Aluminum production cell, meanwhile, Viddal Automation is designing and constructing the production cell.

1.1 RUFO AS

RUFO AS is a production company located in Ørsta, Norway. RUFO was established in 1983 and currently has 19 full-time employees.The company produces flightcases as shown in figure 1.1 often used to transport equipment used in the music industry. RUFO is the biggest supplier of flightcases in Norway, and they want to expand into Europe.

Figure 1.1: RUFO flightcase
Copyright: RUFO.no

As of today most of the flightcase production is done manually, except the plywood plates which are milled using their own milling machine. To expand their market RUFO needs to implement industrial 4.0 and automation to compete on price and demand. Because of this RUFO aims to automate as much as possible of their production over a five year period, starting with the aluminum production cell. This cell will automate the process of cutting and milling holes in the aluminum profiles surrounding the plywood shown in the figure 1.1

1.2 Thesis description

As stated above the thesis is to design the electrical system and develop the software for the aluminum production cell. This production cell shall handle the production of the profiles used to construct the flight case framework, consisting of two profile types. The angle profile shown in figure 3.2 and the guiding profile in figure 3.3.

Figure 1.2: Angle profile (Adam-Hall, 2018a)

Figure 1.3: Guiding profile (Adam-Hall, 2018b)

The production process for an aluminum profile is illustrated in figure 1.4 and includes, cutting the profiles into correct lengths, and drilling holes for pop rivets. There are however some limitations to our thesis compared to the fully operational production cell. The production cell RUFO has ordered from Viddal Automation includes automated profile feeding from a magazine and an automated storage unit for finished profiles. This is not included in the thesis, where as the thesis concerns the control system for the processing unit, which cuts and drills holes in the aluminum. Parallel to our thesis, Viddal Automation will design and construct the production cell.

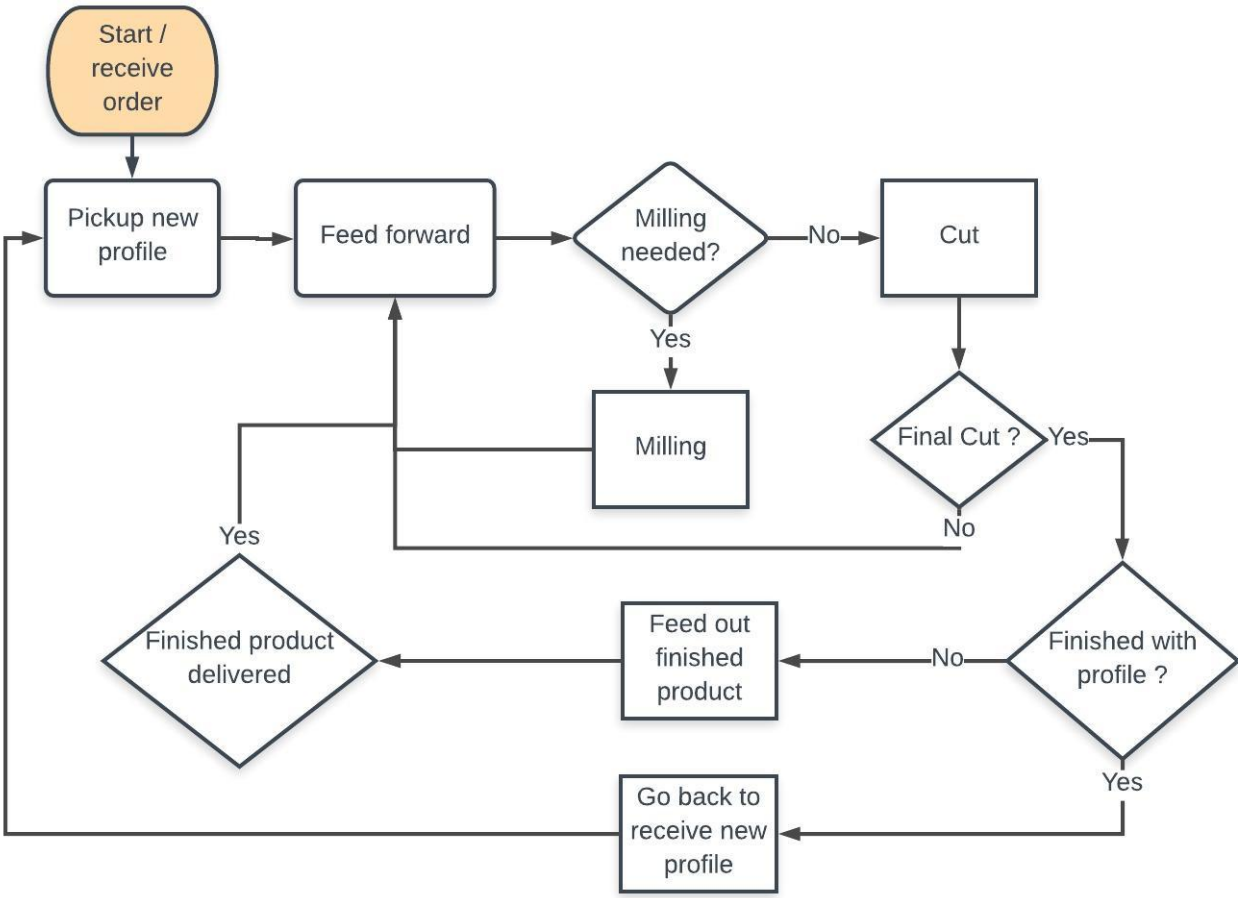


Figure 1.4: Production cell flow chart

1.2.1 Production cell specifications

The project specifications was developed during a meeting that included the bachelor group, supervisors, Viddal Automation and RUFO AS.

1. Precision:
   (a) 1/10 mm precision on cutting and holes.
   (b) Both a cut or hole cannot leave defects on the profiles.
   (c) Circular hole with 5mm diameter
2. Production capacity:
   (a) Production capacity should be maximized to the point were it does not affect quality of the product.

(b) The production cells must allow processing of aluminum profiles up to 6 meters.

3. Requirements for cutting profiles:

   (a) The angle profile should be cut at all angles between -45 to +45 degrees where 0 is 90 degrees, in the vertical axis, as shown in the figure 1.5 below

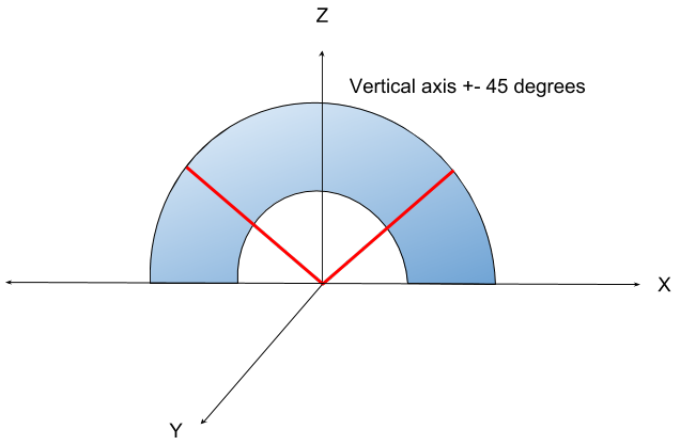


Figure 1.5: Cut in the vertical axis

   (b) Guiding profile should be cut at all angle between -45 to +45 degrees where 0 is 90 degrees, in the horizontal and vertical axis. Meaning that the profile can be cut in the axis figure 1.5 above shows, and as the figure 1.6 illustrates.

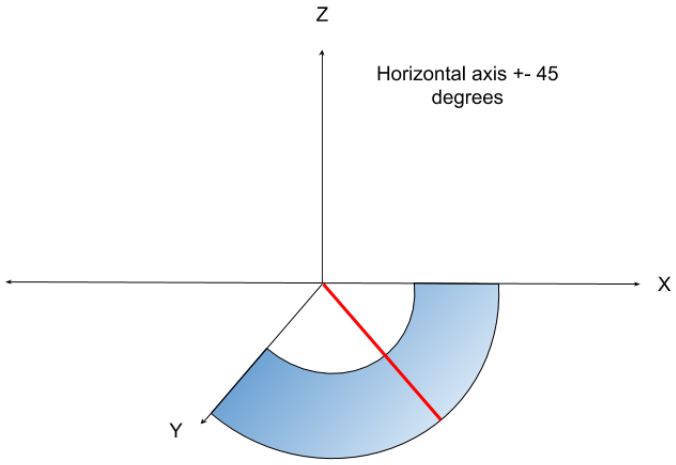


Figure 1.6: Cut in the horizontal axis

   (c) The production cell must have the ability to partially cut through a guiding profile so it can be bent into a 90 degree angle.

(d) The production cell must be able to trim cut both ends of a guiding profile at any angle between -45 to +45 degrees where 0 is 90 degrees, in the vertical axis, as shown in the figure 1.5 above.

4. Reducing wastage:

   (a) The production cell software must optimize production to reduce waste

5. Robot:

   (a) The production cell must have a robot arm used in some part of the production. This robotic arm must be of the type Articulated robot with 6 degrees of freedom.

6. Deadline:

   (a) Mount at RUFO at the beginning of May.

1.2.2 Problem formulation

The information provided above is the fundamentals for the further thesis work. To ensure that the result of the thesis is quantifiable, we developed a problem formulation. This formulation together with the production cell specifications is used to measure the thesis success in section 8 *Testing and results*, and section 9 *Conclusion*. The problem formulation is as follows:

- *Is it possible to develop an automated control system for an aluminum profile production cell, by following the project specifications provided by RUFO AS?*

Chapter 2

Theory

The thesis has multiple problem areas which can be solved using different approaches. Deciding the best solution for each problem, required research on similar automation solutions in the aluminum industry. The problem areas we identified listed below is the foundation for the theoretical research and content in this thesis.

- Machining holes.
- Production optimization.
- HSE - Health Safety Environment:
- Production cell control system:
- Robotic arm:

2.1 Machining holes

Machining holes in aluminum can be done using several methods. The methods we researched were punching, milling and drilling. Milling and drilling are very similar in that the hole quality is decided by the rotational speed, feeding force and hole diameter. Hole punching differs because the quality is decided by the force used to punch a hole through the material. Furthermore this section will look at the mathematical approaches for each method, which are used to determine the best course of action for the production cell in section 4.3.

2.1.1 Punching force

Punching is a technique used to punch a geometrical shaped hole in a material. Since punching can be used to punch different geometrical shapes, the mathematical formulas are different for each shape. (PTD, 2017) The project specifications requires circular 5mm diameter hole, so the calculation formulas for punching holes other than circular are irrelevant. Below are the formula for calculating the punching force shown and explained.(WillsonTool, 2013)

Punching force symbols	
Symbol	Description
D	Hole diameter in mm.
P	The hole perimeter in mm.
t	The material thickness in mm.
σ_s	The material shear strength in MPa

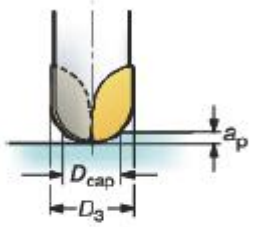
$$P = \pi \cdot D \tag{2.1}$$

$$F = P \cdot t \cdot \sigma_s \quad \text{The punching force in newton.} \tag{2.2}$$

$$m = \frac{F}{g} \quad \text{The punching force in kg} \tag{2.3}$$

2.1.2 Milling

Milling is a technique used to create geometrical components from materials such as wood and metal. Milling has the ability to cut a material in 3 dimensions(Sandvik, 2018). For our thesis we only need to cut in 1 dimension to complete a hole. An overview of how the milling works can be seen in figure 2.1. To mill a hole, one has to take into account three mathematical formulas:



$$D_{cap} = \sqrt{D_3^2 - (D_3 - 2 \cdot A_p)^2} \quad \text{Cutting diameter at cutting depth } A_p \tag{2.4}$$

A_p is the difference between the uncut and the cut surface in axial direction, figure 2.1 however for some milling tools the $D_{cap} = D_3$ because the tool is flat on the end. (Sandvik, 2018)

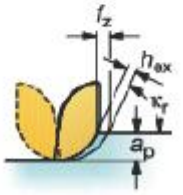


Figure 2.1: Ball nose end mill (Sandvik, 2018)

$$V_c = \frac{\pi \cdot D_{cap} \cdot n}{1000} \quad \text{The Cutting speed (m/min)} \tag{2.5}$$

$$n = \frac{V_c \cdot 1000}{\pi \cdot D_{cap}} \quad \text{Spindle speed, RPM (r/min)} \tag{2.6}$$

2.1.3 Drilling

Drilling and milling is two very similar operations. While milling can be used to create any geometrical shape in a material, the drilling method can only be used to make circular holes in a material. Because of this similarity the math used to decide the cutting speed and spindle speed are almost identical. The only difference is that D_{cap} is no longer cutting diameter at cutting depth A_p but the diameter of the drill D.(Sandvik, 2016)

$$V_c = \frac{\pi \cdot D \cdot n}{1000} \quad \text{The Cutting speed (m/min)} \tag{2.7}$$

$$n = \frac{V_c \cdot 1000}{\pi \cdot D} \quad \text{Spindle speed, RPM (r/min)} \tag{2.8}$$

2.2 Production optimizing

There were two directions to chose from when researching production optimizing. The first direction is linear programming through mathematical algorithms, and the other one artificial intelligence. There were two approaches that appeared in several contexts, the genetic algorithm and the one-dimensional cutting stock problem.

2.2.1 Genetic algorithms

Genetic algorithms is an attempt to simulate the natural evolution in the real world. Were the fittest and strongest animals get to pass on their genes to the next generation. In simpler terms genetic algorithms simulate the term, survival of the fittest. A Genetic algorithm is used to solve or optimize a clearly defined problem using stochastic search algorithms, a common way to explain how such an algorithm operates is explained in detail as a 10 step explanation the book Artificial intelligence by Michael Negnevitsky (Negnevitsky, 2011a).

Genetic algorithms uses chromosomes defined as a binary string to solve a problem, were each chromosome can be the optimal solution to the problem. The algorithm works as an iterative process until the solution satisfies the users termination criterion. During each iteration chromosomes are generated by mating the fittest chromosomes from the previous iteration, to simulate natural evolution as accurate as possible the chromosomes are also exposed to mutations.(Negnevitsky, 2011b)

Such an algorithm can be used to minimize wastage during the production by defining the production optimization as chromosomes with a termination criterion that ends the evolution the optimal solution is found.

2.2.2 The one-dimensional cutting stock problem

The journal *A one-dimensional cutting stock problem in the aluminum industry and its solution* written by *Hartmut STADTLER* Describes a similar optimizing problem as this bachelor thesis. It is a linear optimizing algorithm to optimize the production on 6000mm aluminum profiles. The algorithm calculates the minimum number of profiles needed to produce an order, as result the waste is reduced. (Stadtler, 2000)

2.3 HSE

The finished production cell, both the physical machine and the software must be built to meet the Norwegian health, safety and environment regulations. Such regulations are in order so the risk of harm when operating or standing near the machine is as low as possible.These safety regulations are written down in the machine regulations chapter in the Norwegian law.(Lovdata, 2009)

2.4 Production cell control system

The control system concern the theoretical background used to develop the control software for the production cell.

2.4.1 G-Code

Geometric code or G-Code, is the most commonly used language in computer numerical control programming, commonly known as CNC programming. G-code is used to instruct an automated machine on how and were to move. G-Code is an adaptive language meaning machines developed by different companies or even the same companies may speak different dialects of G-Code. With such an adaptive language, each program can be tailored to a specific task. (Autodesk, 2017)

A line of G-Code can look like this:

- G01 X10 Y54 Z13 F20 T03 M03 S350

The G# defines the motion type, and how to get to the position. X#,Y# and Z# is the target position. F# is the feed rate for the tool, and S# the spindle speed. T# Represents the tool used during the motion. The last bit of information is the M#. The M is also known as M-Code and tells the machine how to perform an action, for example deciding that the spindle shall spin counter clockwise.(Autodesk, 2017)

2.4.2 Communication protocols

Developing a software to control machinery often requires multiple communication protocols. Which protocols you should use, is often decided by the manufacturer of the control components . With this in mind, the protocols used in this thesis are decided by the chosen manufacturer described in chapter.4.4

ADS - Automation Device Specification

Automation Device Specification or ADS is a communication protocol used by Beckhoff in all their hardware and software technology. Figure 2.2 shows how the ADS communication ties an automation system together. (Beckhoff, 2018a)

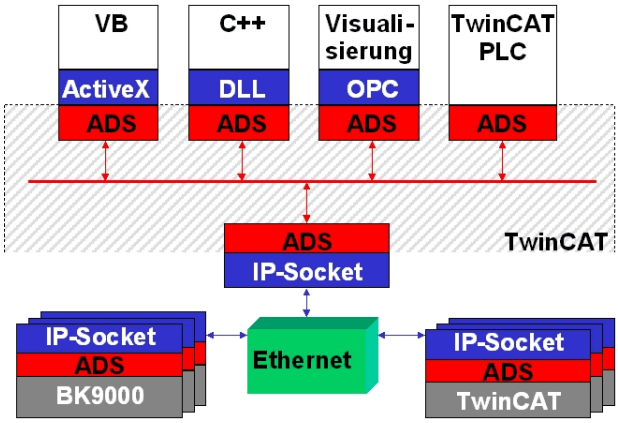


Figure 2.2: Automation Device Specification (Beckhoff, 2018a)

As the figure 2.2 shows the ADS is able connect different devices and software's together. It is able to do this because ADS runs on top of the TCP/IP or UDP/IP protocols. ADS utilizes the same network and addresses as the Internet, but with an added layer to the IP address. A normal IP address has 4 slots *255.255.255.255*. The ADS IP address called AMS Net id, adds two slots two the IP address, *255.255.255.255.255.255*. This requires an additional layer of routing to handle this communication, which is done by Beckhoffs own software called TwinCAT.(Beckhoff, 2018c)

TwinSAFE and Safety over EtherCAT

EtherCAT is a communication protocol for automation technology and uses Ethernet as the physical communication layer. EtherCAT is based on the master, slave principle, were the master sends out a train of information and each slave picks out the information addressed to it selves.(EtherCAT, 2018a)

The TwinSAFE safety system includes both hardware and software solutions, and is developed by Beckhoff. The TwinSAFE hardware part utilizes specialized safety IO modules. These modules are fail safe and therefore legal to use for safety functions according to the Norwegian HSE regulations. The communication is done through the TwinSAFE protocol, fail safe over EtherCAT (FSoE).(Beckhoff, 2017b), (EtherCAT, 2018b)

2.5 Robotic arm

One of RUFO's demands, is that the project must utilize an articulated robotic arm in the production cell, en example of such a robot can be seen in figure 2.4 below. Just implementing such an arm, is a very general demand and can be fulfilled through multiple solutions. A Robotic arm is often implemented in the production were the tasks needed to be performed is either repetitive or in a dangerous work environment. There are two types to mention when talking about robotic arms, industrial and collaborative robots.

An industrial robot is capable to work much faster and handle higher payloads than a human. To implement such a robot, one must take into account the safety regulations. Industrial robots are dangerous and , must be separated from the human work environment. This is usually solved by putting the robot inside a safety cell. (Greenfield, 2018), (Standardization, 2012)

Collaborative robots are safe for humans, so they do not need a safety cell. They are also easier to program and integrate into the work flow of a production line. But because of the safety features that make the robots collaborative, the robot becomes slower and weaker than the industrial type. (Greenfield, 2018), (Standardization, 2012)

Although there are big differences between an industrial and a collaborative robot. What they have in common is that both types operate from coordinate systems and moves in similar patterns.

2.5.1 Coordinate systems

A robotic arm can utilize several coordinate systems to move around in its work station. Since the requirement for our robot is that it can move in 6 degrees of freedom, the common denominations for the coordinate systems is: (x,y,z,A,B,C), were x,y and z represents 3 degrees of freedom and tells the robots were it is in the x,y and z axis. A,B and C is the last 3 degrees, and represents the rotation of the coordinate system. The coordinate systems and rotations are illustrated in the figures 2.3 and 2.4 below.

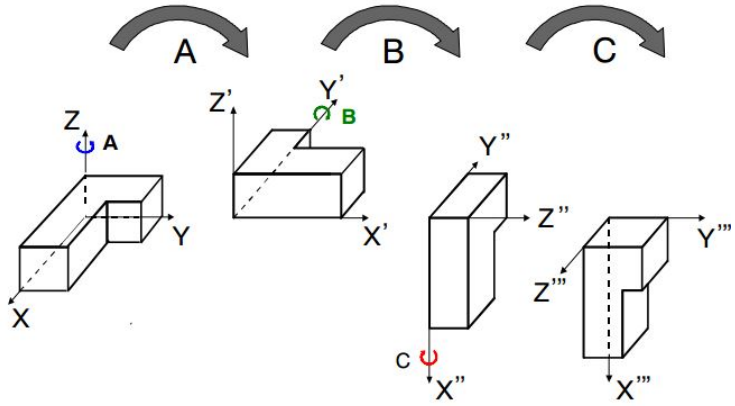


Figure 2.3: Rotation of coordinates (KUKA, 2003)

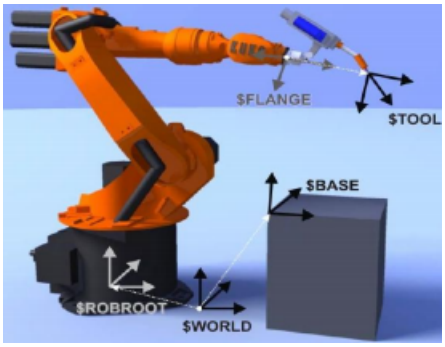


Figure 2.4: Robot coordinate system (Shah, Negal, and Sharma, 2016)

Figure 2.4 above, shows the different coordinate systems of a robotic arm. Below is a simplified explanation of each of the coordinate system, meaning in the real world all the coordinate systems are linked together and influences each other.

Robroot coordinate system

Robroot is the robots internal coordinate system. As shown in figure 2.4 the coordinate system is placed in the base of the robot. The x, y and z axis in robroot is always the origo of the robot, and is a static position which cannot be changed.(Shah, Negal, and Sharma, 2016)

World coordinate system

The world coordinate system can be describes as the room the robot is located in. If the worlds origo is defined in one of the corners in this room, and the robot is placed at a random location in this room. The robroot position at this random location will be in robroot's origo but at position x,y and z with rotation A, B and C in the world system. (Shah, Negal, and Sharma, 2016)

Base coordinate system

The base coordinate system is relative to the worlds system, and a robot can have multiple base coordinate system. A base coordinate system is a position x,y and z in the World system. This means that the origo of a base coordinate is located in position x,y and z axis with rotation A, B and C in the world system. A base system is often used to define workstations around the robot. When you want the robot to move to a certain work station you tell the robot to move to that stations base system. (Shah, Negal, and Sharma, 2016)

Flange coordinate system

The flange coordinate system is located at the tip of the robot, shown in figure 2.4. This system is used as the origo point for any tool mounted to the robot. (Shah, Negal, and Sharma, 2016)

Tool center point

The tool center point known as TCP is a freely definable point in a coordinate system, defined from the flange system. The TCP coordinate is located as a offset from the flange in the x,y and z axis as well as rotation through A,B and C. TCP systems are used to move the robot oriented around the tool. (Shah, Negal, and Sharma, 2016)

Chapter 3

Early concepts

At the early stages of the thesis we had a general idea on how the finished production cell would look. Because one of our thesis members had an internship at RUFO during the autumn semester leading up to this thesis. During the internship he designed a conceptual model of the production cell. It is this concept we used during our early software and electrical prototyping.

3.1 Production cell concept

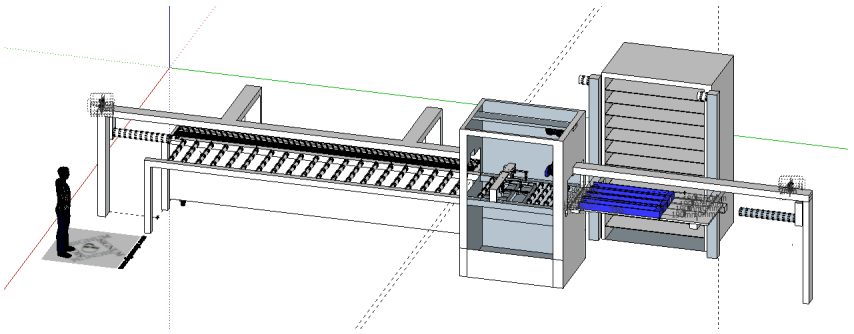


Figure 3.1: Production cell

The concept model shown in figure 3.1 above was developed during the internship by Fredrik Aarset Ryslett. The left side of the model is the station which feeds aluminum profiles into the production cell. The center part is the station which machines the aluminum into correct lengths and places the holes, so the aluminum profile matches the order specifications. On the right side the finished profile is fed out and put into storage.

In the actual machine there will be two production lines. Production line one handles the angled aluminum profiles shown in figure 3.2 below, and production line two handles the guiding profiles shown in figure 3.3 below. This is the concept we used as a foundation for the software development, and as a baseline for the rest of the thesis.

Figure 3.2: Angle profile (Adam-Hall, 2018a)

Figure 3.3: Guiding profile (Adam-Hall, 2018b)

3.2 Software concepts

The thesis main objective is to design and develop the software for the production cell. This naturally mean that the software design went through an iterative process both through discussions with Viddal Automation, RUFO AS and internally through software testing.

First concept

The first design we developed was after the meeting with RUFO and Viddal Automation were we agreed on the specifications surrounding the production cell. Through a meeting with Viddal Automation at later date we discussed and set the parameters for what software to use for the final product, which is discussed later in chapter 7.1. Based on the information gathered from these meetings, the first software concept was designed, illustrated in figure 3.4.

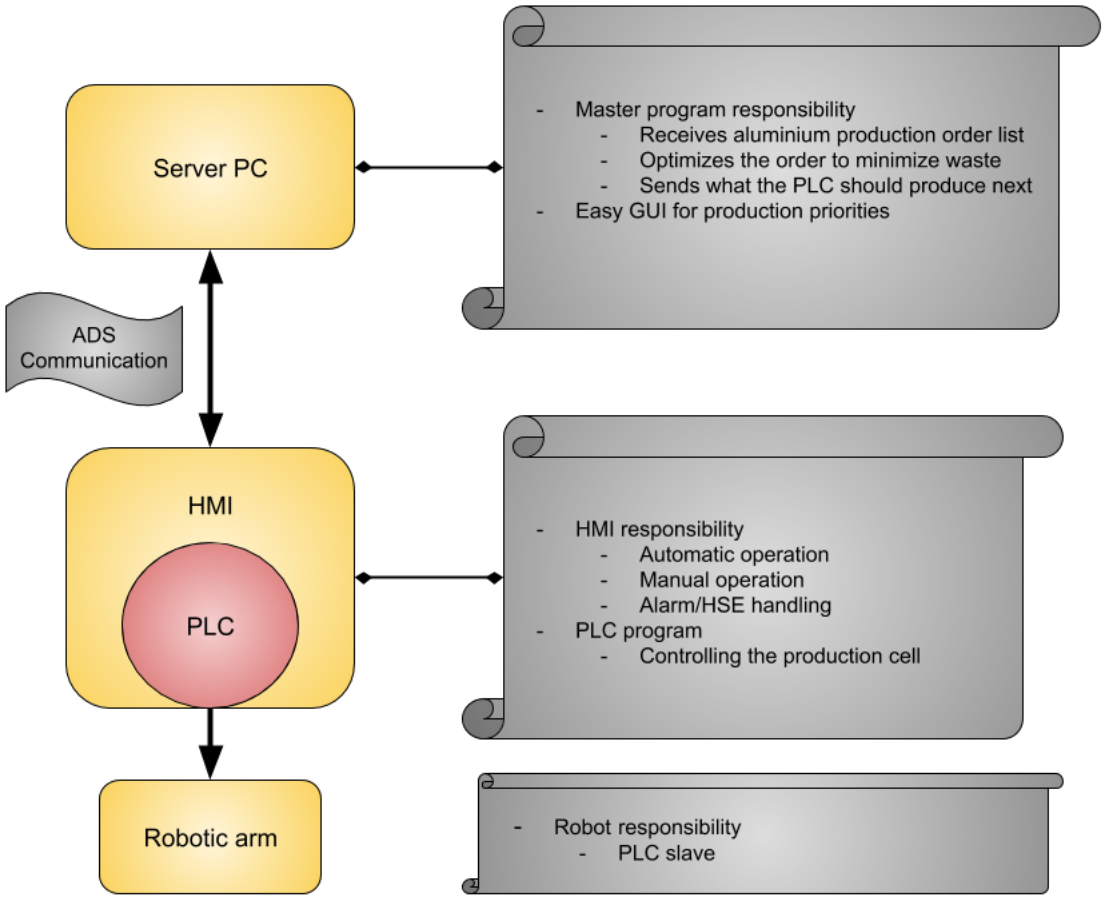


Figure 3.4: First software concept

The idea behind this design is to split up the software into three independent programs illustrated in 3.4 above:

1. Server PC:
   The idea is to develop the software which handles orders, production optimizing and communication with the PLC, and run this software on RUFO's own server separate from the production cell. The server operates as an HTTP server which is the main protocol used in today's web browsers. The reasoning behind designing the server like this, is that it makes it easy for further GUI development. Because for our thesis the main focus should lay on the development of the production cell control, and not a a fancy GUI. The GUI will only make it possible to do simple production order priorities.

2. HMI and PLC:
   The HMI and PLC is the interface and control software which is connected to the production cell. The PLC communicates with the server PC using ADS communication, described in chapter 2.4.2. While the PLC is the entity which communicates and controls the production cell, it is the HMI which is the operators interface. From the HMI the operator shall be able to control the production cell.

3. Robotic arm:
   The robotic arm operates as the PLC's slave. Meaning we develop predefined programs on the robot, which is triggered by events on the PLC.

Second concept

The second concept shown in figure 3.5 below, was developed after we did testing, and discussed the implementation of the first concept with Viddal Automation

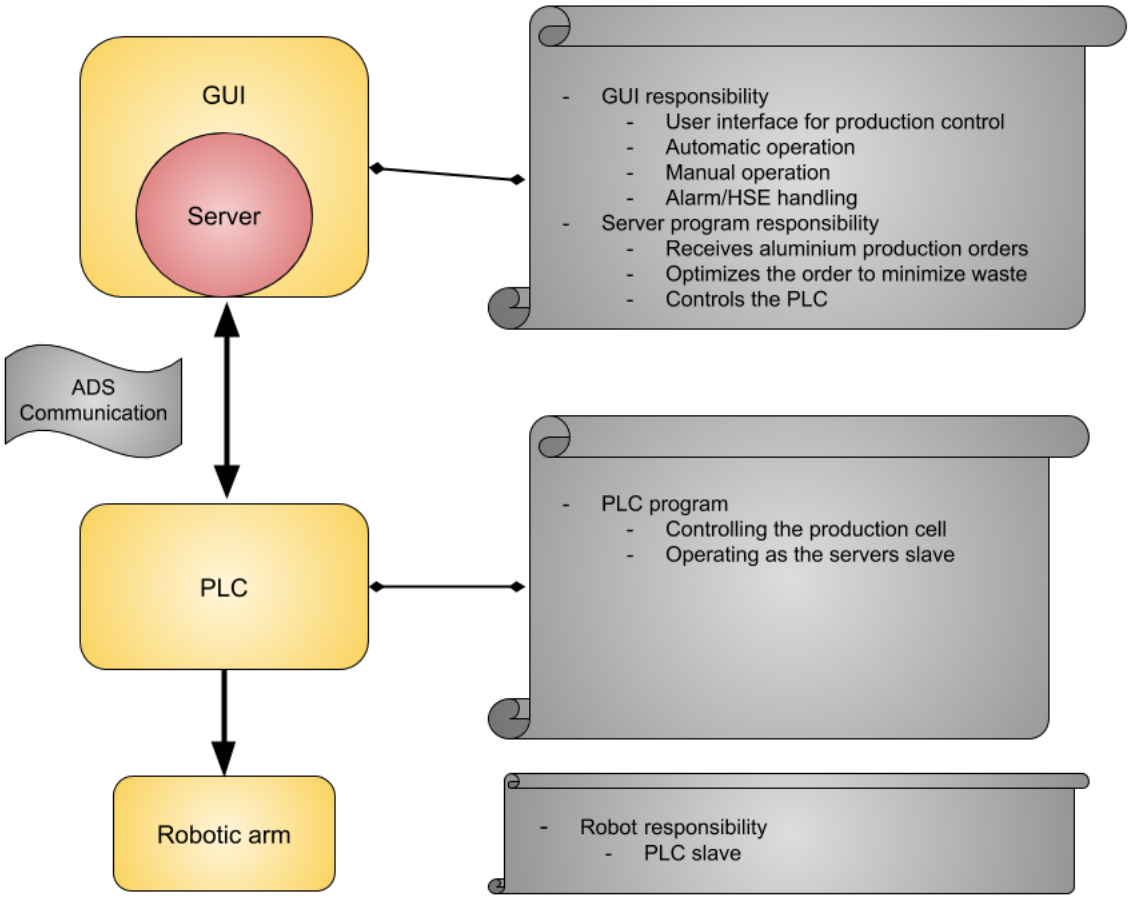


Figure 3.5: Second software concept

Visually the concepts look very similar, but there are some major differences. The server software will now run on its own computer stationed with the production cell. The PLC keeps the original responsibility to control the production cell, but looses the HMI. In this concept there is only one user interface and that is the GUI shown in figure 3.5. The GUI and server in this solution is the same program.

First VS Second concept

We ultimately landed on the decision to abandon the first concept and pursue the second concept based on a few key points.

- Complexity:

  The second concept is easier and less complex that the first concept, based on the fact that the first concept requires four independent programs to run seamlessly together. The server, server GUI, HMI and PLC. Because concept two is less complex, the risk of failure is reduced by pursuing this concept. The implementation of this concept eliminates the need for RUFO's server, by integrating a dedicated computer in the production cell.

- User friendly:

  The second concept is more user friendly, because the final user only has to learn and relay on one HMI/GUI to operate the production cell.

- Time:

  The project needs to be completed in a limited time frame, so a less complex system like the second one will take less time to complete and we can focus on making a reliable and good software.

Chapter 4

Production cell hardware

This chapter includes elements surrounding the design, which is relevant for the control system. Even though it is Viddal Automation's responsibility to design and build the physical machine, some of the vital design elements required joint agreement.

4.1 Robotic arm placement

The production cell specifications from chapter 1.2.1 states that the production cell must implement an Articulated robot with 6 degrees of freedom. The specifications however does not specify how the robot should be implemented. Through discussion and research both online and by visiting companies, we narrowed the robotic implementation into two possible approaches: Using the robot to cut the profiles, or collect the finished profiles from the production cell.

Cutting

When deciding whether or not the robotic arm's optimal implementation is to cut profiles, it is important to consider the cut specifications from chapter 1.2.1 illustrated in the figures 4.1 and 4.2 below

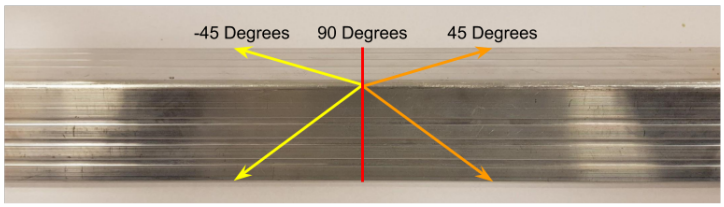


Figure 4.1: Angled profile cut configuration

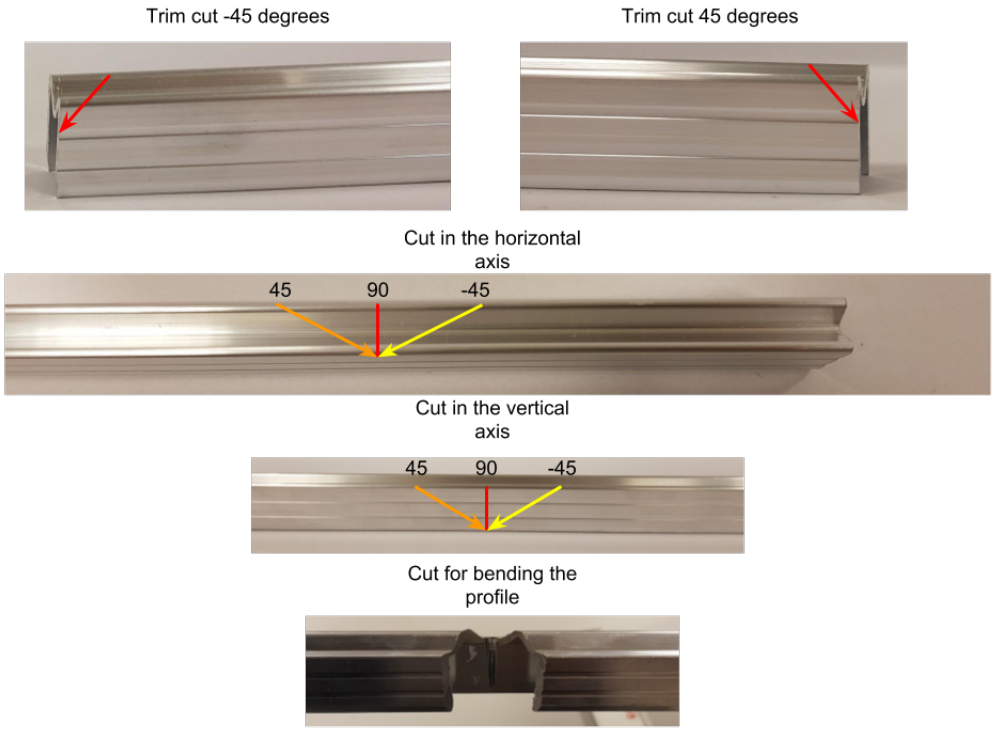


Figure 4.2: Guiding profile cut configuration

To cut the angle profile, a saw with a minimum of 3 DOF is required. The guiding profile cut configuration is more advanced and requires an additional degree of freedom, making it a 4 DOF as a minimum. This means that our 6 DOF robot fulfills the requirements to cut both profile types. Since the production cell requires two production lines, one for the angle an the other for the guiding profile. The robot must be placed in the middle so it reaches both production lines. By doing so we risk that the robot becomes a bottleneck in the production capacity, because while the robot performs a cut on line one, line two must wait until the robot finishes.

Another solution to cut the profiles is to develop dedicated saws for each line. This removes the bottleneck risk because both lines can run independently, which also leads to better lead times. This approach will however increase the complexity of the physical production cell and the control system. When researching for dedicates saws we only found already finished saw systems, most of them with only 3 DOF. So to implement a solution using dedicates saws means that Viddal Automation will have to design and build own dedicates saws.

Profile collector

The second approach is to utilize the robot as a profile collector. In this solution the robot is stationed outside the production cell, and each time a finished profile is machined the robot reaches into the cell and collects the finished profile for delivery. The same bottleneck risk is possible in this solution, since one robot must collect from two production lines.

Another solution her as well, is to implement dedicated stations which feed out the profiles for each production line. Although it is less complex than developing custom dedicated saws. Such a solution can use linear guided gripers which travel along the production line and collects finished profiles, or implementing a conveyor belt.

Robotic arm placement

When deciding were to implement the robot, we discussed the solutions above. We concluded that the logical implementation of the robot would be to utilize it at the cutting station. The reason behind this decision was that developing and constructing two custom dedicated saws would implicitly mean higher production cost and longer production time, than developing a linear gripper or conveyor belt solution. The cuts illustrated in figure 4.1 and 4.2 above are easily performed by the facet that the minimum DOF need is 4, and the robotic arm has 6. The final argument for this solution is that the Robot adds the ability to perform more advanced cuts, should RUFO's flight case production need it in the future.

4.2 KUKA KR6 AGILUS

The choice of robot to use in the cutting station, is the KUKA KR6 AGILUS. The decision for choice of brand and robot type fell on Viddal Automation, since KUKA is the preferred robot brand, this is also were the company expertise lays. It is important to add that all industrial robots matching the KR-6 specifications can be used as a viable alternative in the production cell. The KR-6 robot specifications can be found in this PDF document(KUKA, 2018a).

Figure 4.3: KUKA KR6 AGILUS robot (AET-Labs, 2018)

4.3 Hole station

The most important feature for the hole station is that the machined holes does not leave defects on the aluminum profile. To fulfill this demand we researched several methods as described in section 2.1 in the theory chapter.

Punching

The first idea was to punch holes in the profiles. This method leaves minimal amounts of metal shaving and only one piece of waste, so when designing the machine there is no need to consider cut shaving removal. Punching holes demands a significant amount of force to generate a hole, the amount of force needed is explained in equation 2.2 in section 2.1.1. To calculate the punching force the material shear strength is needed. The shear strength of the profiles used by RUFO is 130MPa (Makeitfrom, 2018), this gives us a minimal punching force of:

$$D = 5mm, \quad t = 2mm, \quad \sigma_s = 130MPa$$

Using the perimeter equation 2.1

$$P = \pi \cdot 5 \quad \rightarrow \quad P = 15,7mm$$

Using the punching force equation 2.2

$$F = 15,7 \cdot 2 \cdot 130 \quad \rightarrow \quad F = 4082N \quad \text{In punching force}$$

The minimal force needed to punch a hole in the profiles is 4082N, and is a viable solution. However when we visited Spilka as, a company producing aluminum profiles for window frames. We were recommended not to punch holes, since the aluminum is a weak metal which bends easily. Spilka had done tests which resulted in that Milling gave better results. Based on this visit and fact that we could not test the result of punching ourselves, we decided on not pursing punching.

Milling and Drilling

Scraping punching as a possible solution leaves us with milling and drilling. When looking at both solutions as methods to machine circular holes in a material, they are practically identical in terms of the physical and theoretical approach. When looking at the equations 2.5, 2.6, 2.7 and 2.8 for milling and drilling in section 2.1.2 and 2.1.3, the only difference is the definition revolving around the tool diameter.

Deciding which method would provide the best result was done through testing on aluminum profiles. The machine used to implement the test could only provide a spindle speed of 2600 RPM. The machines cutting speed was not automated, so it is impossible to know the cutting speed we had during the tests. It is however possible to calculate the maximum cutting speed when the spindle speed is 2600 RPM. For drilling the maximum cutting speed V_c is:

$$n = 2600RPM, \quad D = 5mm$$

Using the drill cutting speed equation 2.7

$$V_c = \frac{\pi \cdot 5 \cdot 500}{1000} \quad \rightarrow \quad V_c = 7,85m/min$$

For milling the maximum V_c is:

$$n = 2600RPM, \quad D_{Cap} = D = 5mm$$

Using the mill cutting speed equation 2.5

$$V_c = \frac{\pi \cdot 5 \cdot 500}{1000} \quad \rightarrow \quad V_c = 7,85 m/min$$

In our test V_c is the same for milling and drilling, because the mill tool we use has a flat head meaning D_{Cap} from equation 2.4 is equal to the diameter of the hole. Even though we could not control the cutting speed on the machine, we know that the speed during test was lower than the calculations above. The result from the test shows that drilling generates a small amount of imperfections on the exit part of the hole. However the milling did not generate any imperfections on either side of the hole, and therefor the chosen method for the production cell.

4.4 Control components

Development of the control system requires an overview off the electrical components. This implies the brand for PLC's, I/O modules, what type of motors to use and how to utilize them.

Beckhoff

The chosen brand for PLC and I/O modules to control the electrical system is Beckhoff. Beckhoff is Viddal Automation's preferred supplier on control components, therefore it was demanded that we implement their products inn our thesis. As explained in section 3.2 the PLC will be used to control the entire production cell. The I/O modules is used to connect all the electrical components like sensors and motors to the PLC.

Motors

As discussed and explained above in this chapter and previous chapters, the production cell has two production lines which requires motors to feed the profiles into the production cell, motors to adjust the height position of the hole station and motors to feed finished profiles out of the production cell. (Lackey, 2018),(Burris, 2018)

While there are an abundance of different motors and solutions. We narrowed the decision down to two types, stepper and servo motors. When choosing between servo and stepper motors, it is multiple factors that needs to be considered. The most important ones are power, speed and accuracy to suit the task. (Lackey, 2018),(Burris, 2018)

For a machine that need high speed and high torque, servo motor is the best choice. A servo motor is stronger at higher speeds than a stepper motor, and with complex drivers and feedback systems delivers more accurate positioning than a stepper. A stepper motor divides a full rotation into steps. Because of this, moving the stepper motor to a precise position is much easier than a servo. In addition to this a stepper motor is cheaper and easier to acquire.(Lackey, 2018),(Burris, 2018)

Truth be told both motor types can be used in the production cell, but there is not any reason to chose the servo over a stepper motor for this application, when a stepper motor delivers the same result at a lower cost and less implementation time.

Chapter 5

Redefining the bachelor thesis

On 19.03.2018 we had a meeting with Viddal Automation discussing the thesis and project progress. It became apparent that the physical machine nor the design would be finished in May, as stated in the project specification in section 1.2.1. The thesis could still be completed without the physical machine, but not without the design. Programming the PLC and KR6 robot is impossible without a design, because the design holds the hardware information. This information is vital because it holds information about sensors, motors, how to lock the profile in position and were and how the robot should perform a cut, to mention some.

To cope with the changes this implies, we agreed with Viddal that the best course of action would be to build a production cell prototype. A prototype facilitates programming of the PLC and KR6, and testing of the fully developed control software. The testing makes it possible to test and control that the thesis achieves the problem formulation in section 1.2.2, and delivers on the project requirements in section 1.2.1.

Adding the design and construction of a prototype to our thesis, requires major changes in labor distribution. We no longer have to design the electrical system for the finished machine, but needed to redistribute that task to the prototype electrical design. Constructing a prototype within the thesis delivery date would consequently mean that one group member had to focus on this task. This decision meant that the rest of the thesis including software and electrical development fell on the remaining two members.

The construction of the prototype happen in the "Tunglab" at NTNU Ålesund, were NTNU has a KR6 robot and safety cage which can hold the prototype. Viddal Automation supplied the electrical components for the prototype and NTNU supplied the construction materials. While these changes redefine the parameters surrounding our thesis, the main objective remains the same:

- *Is it possible to develop an automated control system for an aluminum profile production cell, by following the project specifications provided by RUFO AS?*

Chapter 6

Prototype development

The prototype development and construction is a consequence followed by the thesis redefinition explained in section 5. Constructing the prototype within the time limit and making sure it was economically feasible, demanded fast prototyping and smart solutions. While Viddal Automation supplied the control components, NTNU supplied wiring and materials for construction. Due to the time restrictions we tried to find and implement already existing parts and materials at NTNU to eliminate construction downtime. For our thesis it was more important to make a working prototype which enabled software testing before the delivery date, than a perfect prototype.

The finished production cell will utilize two production lines, one for angle and the other for guiding profiles. For the prototype it is expedient to only include one production line. Since the prototype purpose is to test the control system, it was decided that production line two producing the guiding profiles would give us the best test results. Because the guiding profiles implements the most complex cuts explained in section 4.1 figure 4.2

6.1 Prototype concepts

Before construction we developed and discussed the implementation of two prototype concepts. Where the main argument for prototype solution were what materials we had at NTNU. Both concepts are fairly similar but the differ inn which construction materials to utilize.

6.1.1 Concept one

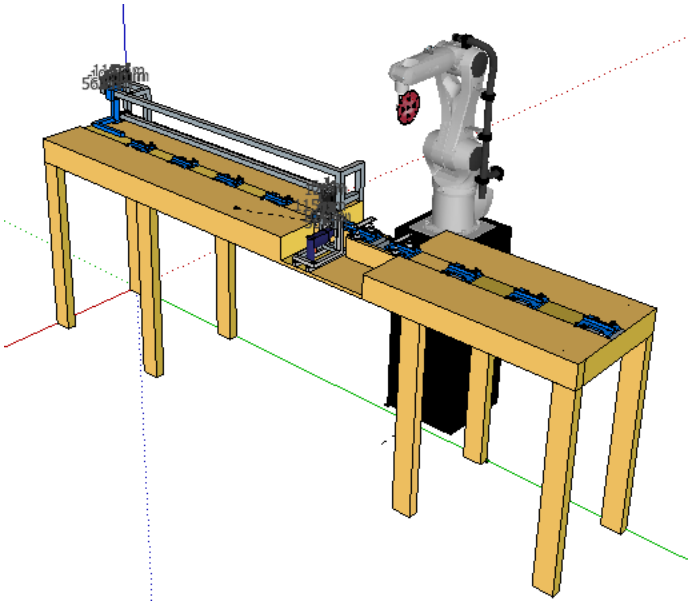


Figure 6.1: Wood framework concept

The first concept illustrated in figure 6.1 above, has a framework consisting of 48x98 mm construction timber wrapped in plywood. Were the control components like, linear guidelines, hole station, grippers and profile holders would be manufactured in metal or 3D printed. The robotic arm is used as intended from section 4.1, as the saw unit.

6.1.2 Concept two

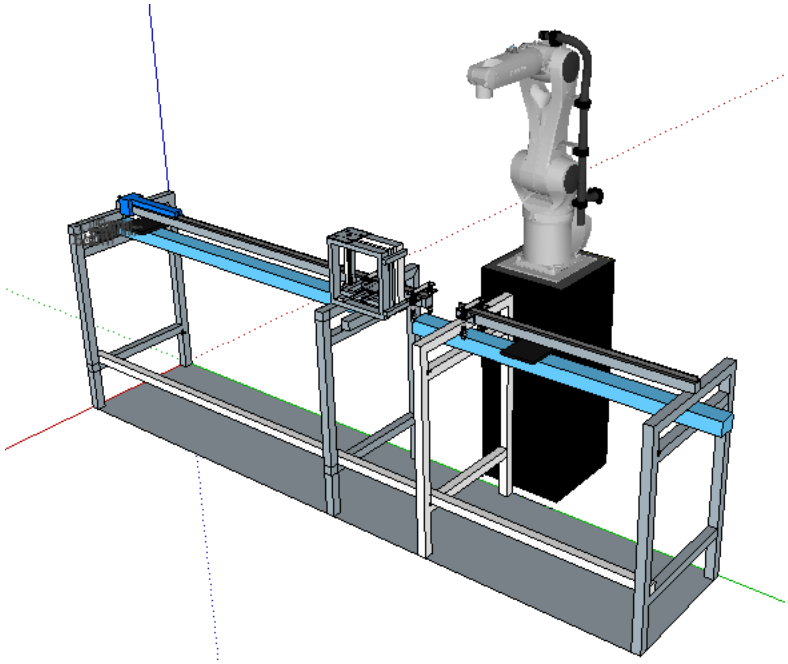


Figure 6.2: Aluminum strut framework concept

The second concept illustrated in figure 6.2 above, is almost identical with concept one. Were the only difference is that concept two is a more refined design, were the framework is constructed using aluminum strut profiles shown in figure 6.3.

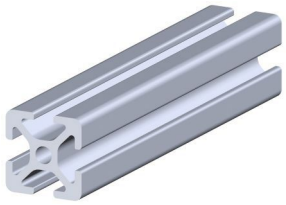


Figure 6.3: Aluminum strut

6.1.3 Prototype

When choosing witch concept to develop into a prototype, the main decision variable was the accessibility of construction materials. We chose concept two because NTNU already had aluminum strut in different dimensions, three finished legs shown in figure 6.2 and mounting hardware for strut profiles. While concept one's wood framework is a cheaper alternative, the aluminum strut is very modular and can be repurposed in other projects. Because NTNU already had aluminum strut and some finished legs, meant that we could start construction while we waited for the remaining materials and equipment.

6.2 Prototype construction

Figure 6.4 below displays the finished prototype. To construct this prototype we used the materials at hand found at NTNU, and Viddal Automation supplied us with the control components. Because the redefinition called for rapid prototyping, we utilized 3D printing for self developed parts. Some parts could not be 3D printed, these parts were made in cooperation with the department of ocean operations and civil engineering. Furthermore this section identifies and explains the key elements of developing and constructing the prototype.

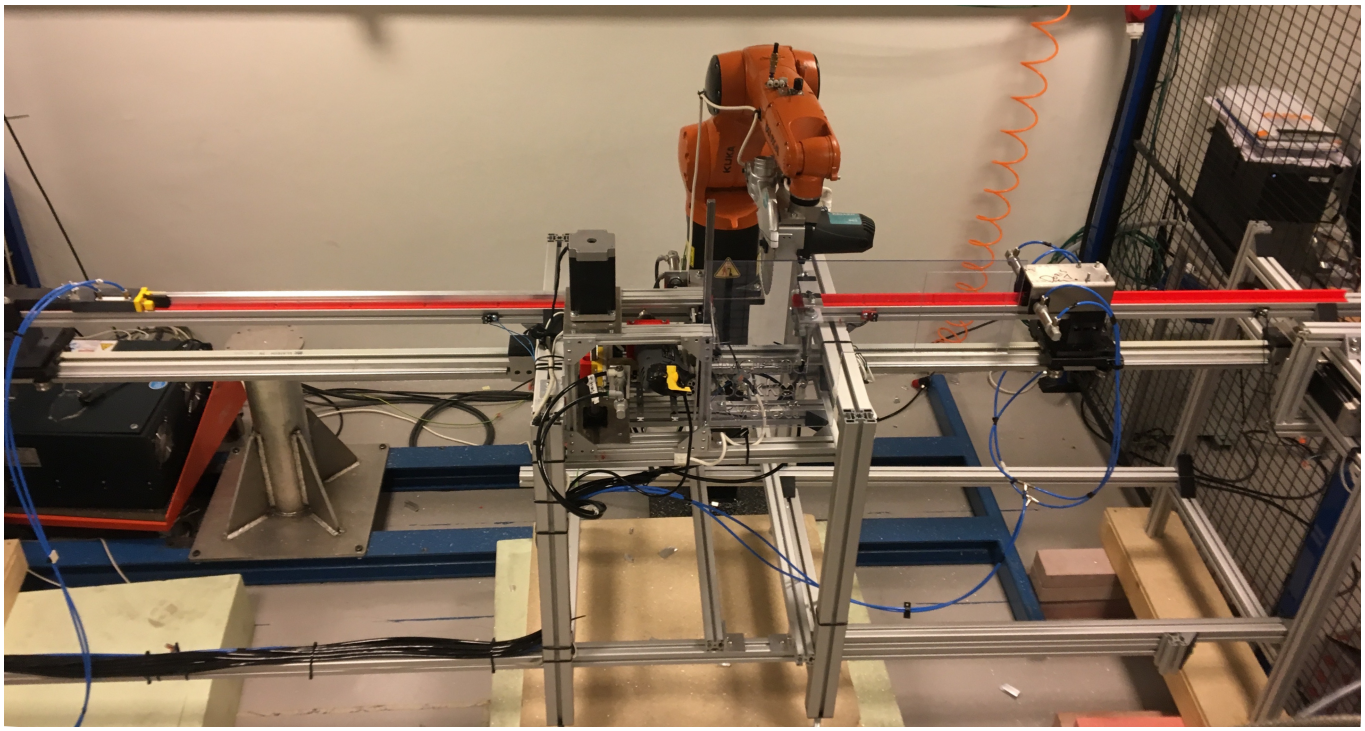


Figure 6.4: Prototype

6.2.1 Production cell input and output

The first element was how to feed profiles in and out of the processing unit, the processing unit is the common denominator for the hole and cut station. Viddal Automation provided two liner guidelines, one for each side of the processing unit controlled by stepper motors.

A 3D printed coupling was developed showed in figure 6.5 to link the stepper motor and guideline together. This particular part was printed with 100% infill to ensure that the part could handle fast acceleration and deceleration, and is used on all the steppers in the prototype.

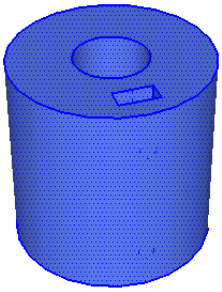


Figure 6.5: Motor coupling

Directly above the guideline is the actual production line shown in figure 6.9. The production line is constructed using aluminum strut with 3D printed guide units showed in figure 6.6. This ensures that the profile is pushed correctly into the processing unit. The guide units is represented as the red part in all figures in this section. An important design detail for this part is the attach mechanism. The bottom part of the guide unit is designed to slide into the aluminum strut, which means quick part assembly and replacement shown in figure 6.7 below. To feed the profiles inn and out of the processing unit two different grippers where developed.

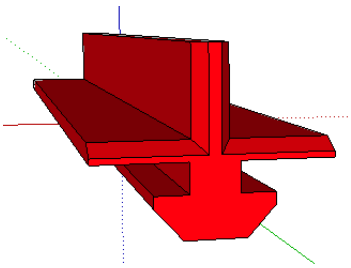


Figure 6.6: 3D printed guide unit

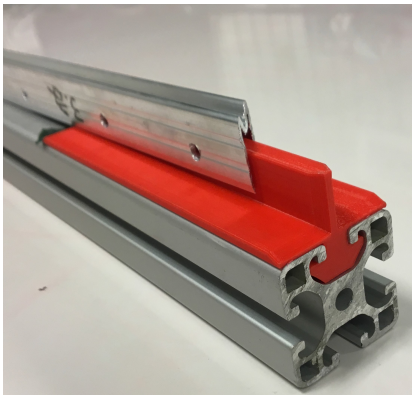


Figure 6.7: Production line example

Input gripper

The gripper for feeding profiles into the processing unit is shown in figure 6.8 and 6.9. The gripper design had three key parameters, it must reach as far as possible into the processing unit, have a firm grip around the profile and glide over the guide units in figure 6.6. To ensure that the gripper holds the profile in position, the design has an integrated pneumatic cylinder. Figure 6.8 shows the 300mm long beak, which ensures that the gripper reaches far into the processing unit.

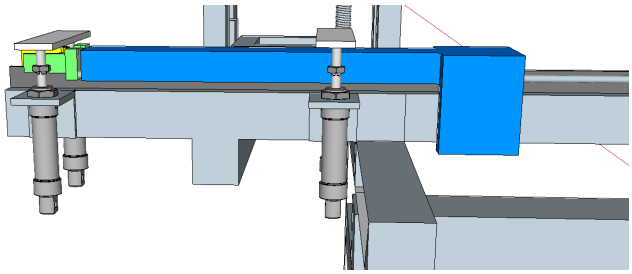


Figure 6.8: Gripper 300mm inside processing station

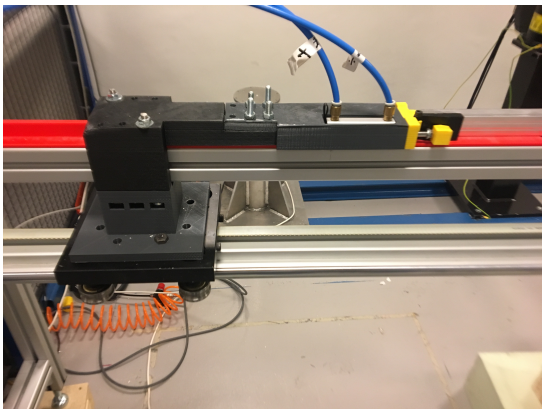


Figure 6.9: Input gripper

Output gripper

The output gripper collects finished profiles and feeds them out. The gripper is integrated with two pneumatic valves shown in figure 6.10, which are configured so the the left valve reacts slower than the right. They are configured this way so profile is pushed into the guide units before a cut. This design also means the gripper acts as a tunnel shown in figure 6.11, letting profiles glide through until a grip command is given.

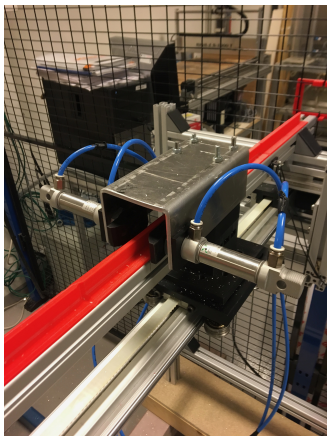


Figure 6.10: Output gripper

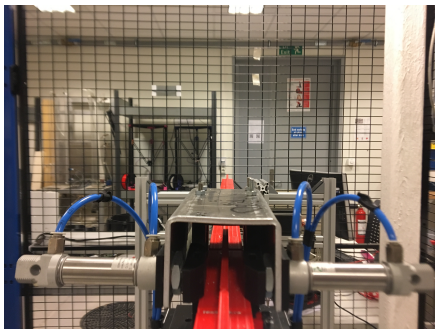


Figure 6.11: Output gripper tunnel

6.2.2 Locking system

To ensure that the profiles are locked into position when performing a cut or hole, we added a pneumatic lock system. The lock system is displayed in figure 6.12, 6.13 and 6.14 below. The locking system is divided into three stations, best shown in figure 6.13. Where station one is at the bottom, two in the middle and three at the top. One and two is used for the hole operation and two, three and output gripper is used for the sawing operation. Station two also includes a pneumatic valve which also pushes the profile into the guide units, like the output gripper.

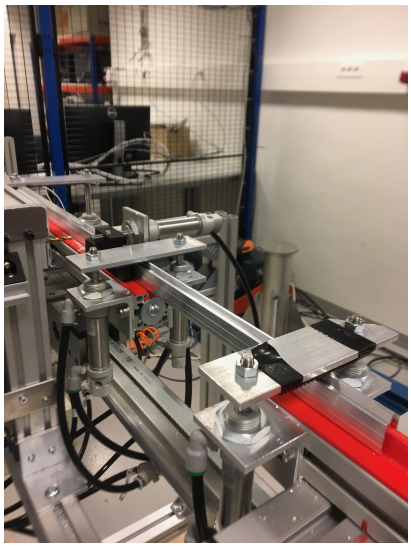


Figure 6.12: Lock system

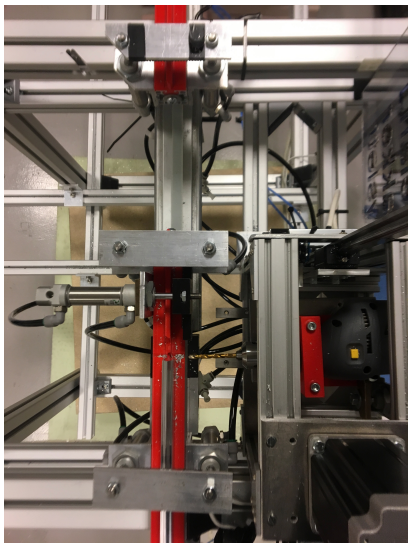


Figure 6.13: Lock system

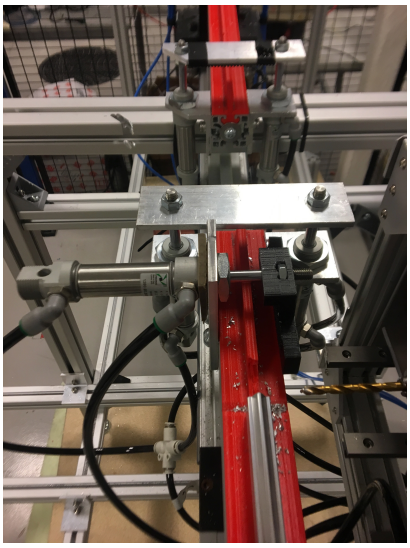


Figure 6.14: Lock system

6.2.3 Hole station

The hole station is constructed using aluminum strut and, self made aluminum mounting hardware and 3D printed parts. Viddal Automation provided a hand held drill, which we customized to fit the hole station. In section 4.3 hole station we argued that milling gave the best hole results. However using a mill tool in the drill provided from Viddal Automation proved impossible. Because the drill only delivers 2500 RPM, and the station uses pneumatic to perform a hole. The use of pneumatic inhibits the possibility to control the cutting speed. A second problem we encountered is that the drills chock is unstable and vibrates, this resulted in that the flat headed milling tool could not latch onto the profile. Therefore the prototype uses a normal drill toll to generate holes shown in figure 6.17. To adjust the hole stations vertical axis we implemented a stepper motor, which utilizes a M16 threaded rod and a M16 nut, shown in figure 6.16.

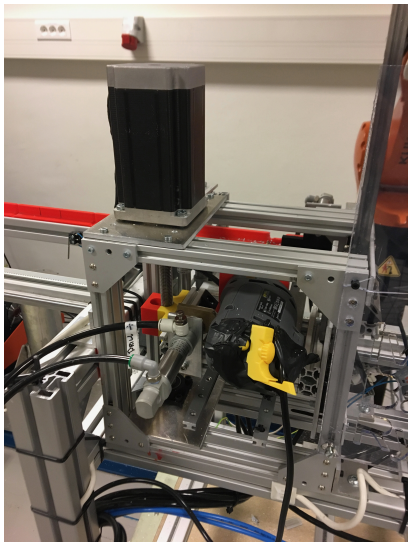


Figure 6.15: Hole station back

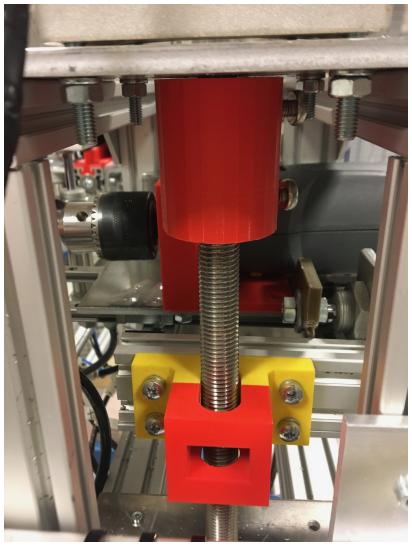


Figure 6.16: Drill adjustment

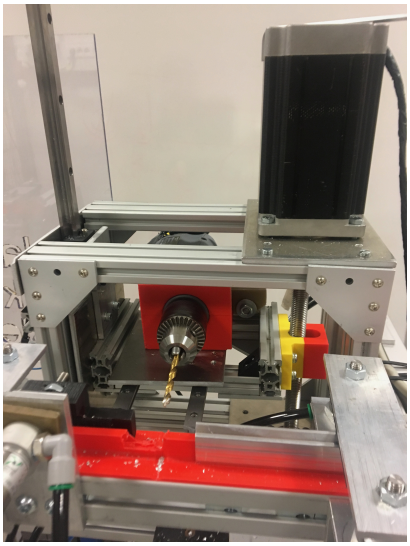


Figure 6.17: Hole station front

6.2.4 Cutting station

The cutting station is located after the hole station, and is defined as the KR-6 robots work space. The station includes the robot, a saw and the cutting area, figure 6.18 and 6.19 below shows all of these components.

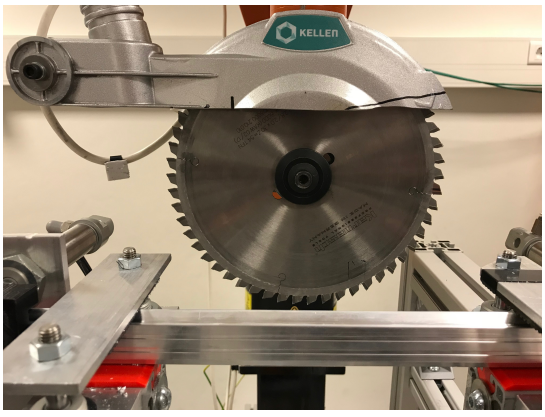


Figure 6.18: Saw and cutting station

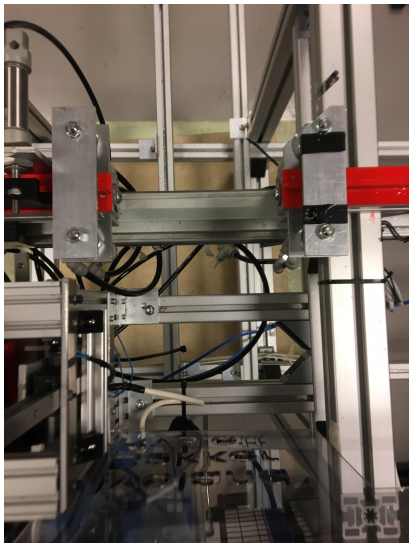


Figure 6.19: Cutting station

Saw fixture

Viddal Automation provided a slide mitre saw which we disassembled and rebuild to fit onto the robot. To mount the saw on the KR-6 we needed to manufacture a saw fixture. The first model was 3D printed and tested, before Viddal Automation had the part manufactured in stainless steel, shown in figure 6.21

Cut area

The cut area is shown in figure 6.19 and is the gap between the lock stations. When testing the room the robot needed to perform each cut, the results were that the cut area needed is 200mm.

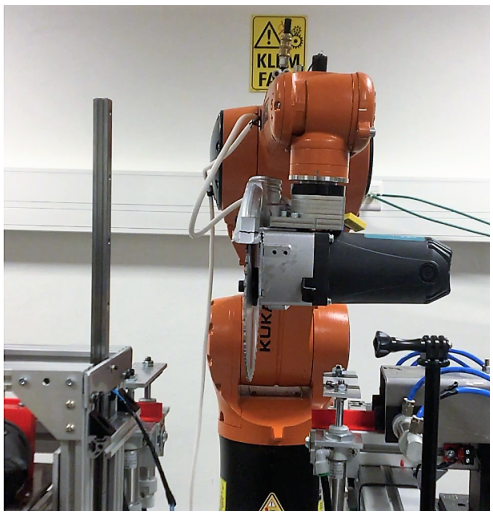


Figure 6.20: Robot cutting station

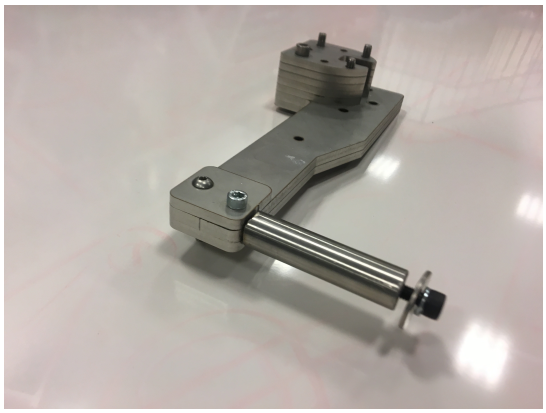


Figure 6.21: Saw fixture

6.2.5 Prototype specifications

Since the prototype was constructed without time to plan every detail and the space restrictions inside the robot safety cage, it lead to the prototype demanded some production specifications to operate correctly:

1. Maximum stock profile length sent to production is 1100mm
2. Minimum length for finished profile 220mm
   Any shorter than this and the profile falls down the cut area.
3. The last length must be 520mm for it to reach the output gripper

6.2.6 Electrical design

The electrical drawings for the prototype were made in PC SCHEMATIC Automation, these drawings are located in appendix H.

6.3 Prototype list of materials

Table 6.1: List of materials

Electrical components:	Brand:	Description:
CP6606	Beckhoff	Industrial PC.
EK1100	Beckhoff	EtherCAT coupler, for connecting IO modules.
EL2008	Beckhoff	Digital output terminal
EL1008	Beckhoff	Digital input terminal
EL2521	Beckhoff	Pulse train terminal for stepper control
EL6692	Beckhoff	Bridge module, for PLC - KUKA communication
Relay 24VDC	Omron	Relay for controlling 24VDC
Relay 24VDC	Omron	Relay for controlling 230V. Control voltage is 24VDC
CW8060	CNC4YOU	Micro stepper controller
NEMA 42		Stepper motor for linear movement
power supply 24VDC, 15A	Wago	Power supply 24VDC
Circular saw	Keller	Saw
Drill	Meeck	Drill
Coils for pneumatic valves	SMC	24VDC coil
KR6 Agilus robot.	KUKA	Industrial robot arm
KUKA KR C4.	KUKA	Robot controller
Safety switch	Eaton	Switch for the safety fence
Wires		
Pneumatic components:	Brand:	Description:
Valves	SMC	5/2 valve consisting of 5 ports and 2 different states (chamber).
Cylinders	SMC	Cylinder for linear movement
Tube and connectors	SMC	
Building material:	Brand:	Description:
Aluminum strut	Bosch	Modular aluminum strut.
linear guidelines		For input and output grippers
linear bearings	Bosch	For moving the drill in Z and Y direction.
Threaded rod M16		For moving the drill in Z direction
mounting equipment		Different screws and nuts for mounting.
Saw fixture		Saw fixture in stainless steel.

6.4 Approximated prototype budget

Table 6.2: Prototype budget

Component	**Quantity**	**Price per pcs**	**Total price in NOK**
Robot with controller	1	300000	300000
CP6606	1	8000	8000
EK1100	3	2000	6000
EL2008	1	1500	1500
EL1008	1	1500	1500
EL2521	3	2500	7500
EL6692	1	2500	2500
Relay 24VDC	8	100	800
CW8060	3	700	2100
NEMA 42	3	700	2100
Power supply 24VDC, 15A	2	2000	4000
Saw blade for aluminum	1	800	800
Circular saw	1	1000	1000
Drill	1	400	400
Pneumatic valves	6	700	4200
Safety switch	1	200	200
Wires and cables	1	2000	2000
Aluminum strut	1	5000	5000
Pneumatic cylinders	9	400	3600
Tubes and connectors	1	500	500
Screws and nuts	1	500	500
Linear guidelines	2	2500	5000
Linear bearings	1	500	500
Threaded rod M16	1	100	100
Saw fixture in stainless steel.	1	1000	1000
		Total:	360800

Chapter 7

Software development

As explained in section 3.2 software concepts, we decided to pursue the second concept. The software is therefore split into three parts. A server containing the GUI, a PLC program to control the production cell, and a third program to control the KR-6 robot.

7.1 Utilized Software

In this sub chapter we will discuss and explain the software used during the software development. This includes support software, IDE's (Integrated Development Environment) and programming languages used to develop our system. With that said, The deciding factor for much of the software used, is decided by Viddal Automation. The reasoning behind this is that Viddal have certain frameworks for their software development and service for already existing systems which we need to follow. By following these frameworks Viddal Automation will be able to continue developing the production cell after we deliver our thesis.

7.1.1 TwinCAT 3

TwinCAT 3 is a software platform for executing and programming Beckhoff hardware, such as Industrial PC's, Embedded PC's and PLC's. TwinCAT runs as an add on in the Visual Studio IDE, and supports multiple programming languages besides the IEC 61131 standard like, C++, Matlab and C#. In addition to supporting multiple languages, Beckhoff has developed multiple libraries which goes with each language. One of these libraries is the ADS-Communication library, mentioned in the theory chapter. Choosing TwinCAT as our main development software is natural, based on the fact that we use Beckhoff PLC and modules as our control hardware. Another important argument is that TwinCAT lets us program all our software except the KUKA robot, in one development environment. (Beckhoff, 2012)

7.1.2 C#

As mentioned in section 3, the system integrates an GUI/server solution. The logical approach to solve this, is by using an objective oriented programming language like Java, C# or Python to mention some. The chapters name already reveals the chosen programming language for the server software, and there is a number of reasons for this decision. The first one is explained in the introduction to this chapter, it is the preferred objective oriented language used in Viddal Automation. However the main and biggest argument to chose C# over its cousins, is that c# is supported by Beckhoff hardware and TwinCAT 3.(Microsoft, 2018a)

We could argue that Java would be a better choice based on the fact that this is the main programming language we have used during our studies. However we had to scrap this choice because TwinCat 3 does not support java. Java and C# are very similar languages with basically the same programming syntax, so choosing C# was the optimum choice.

7.1.3 Visual Studio

Visual studio is an integrated development environment (IDE) from Microsoft. The IDE supports multiple languages like, C#, V#, Python and more. But more importantly it is the main IDE used to develop TwinCat 3 software, and therefore the natural choice for this thesis. (Microsoft, 2018b)

7.1.4 WorkVisual Development Environment

The language used to program the KUKA robots is called KRL and stands for KUKA robot language, WorkVisual is KUKA's own IDE, and it is used to program KUKA products using the KRL language. While there are multiple IDE's for programming KUKA robots, we chose WorkVisual because we have no prior experience programming KUKA robots, and were recommended to use WorkVisual from Viddal Automation. (KUKA, 2018b)

7.2 Server

The brain of the production cell is the server software with an accompanying GUI. The GUI is explained in its own sub chapter further down. In this section the server software development is explained. The server handles the entire system, which includes receiving orders, optimizing orders and sending the production information to the PLC.

It is important to mention that the redefining of the bachelor thesis described in section 5 did not affect the development plan for the server software other than giving the development a stricter time line. So the server software will deliver the same result on the prototype as the finished production cell.

One of the first challenges was to break down the project specifications into code, and making sure the classes we create are within the project guidelines. Figure 7.1 shows the class diagram. The name of each class is named in such a way that they represent production information or a task the server needs to perform.

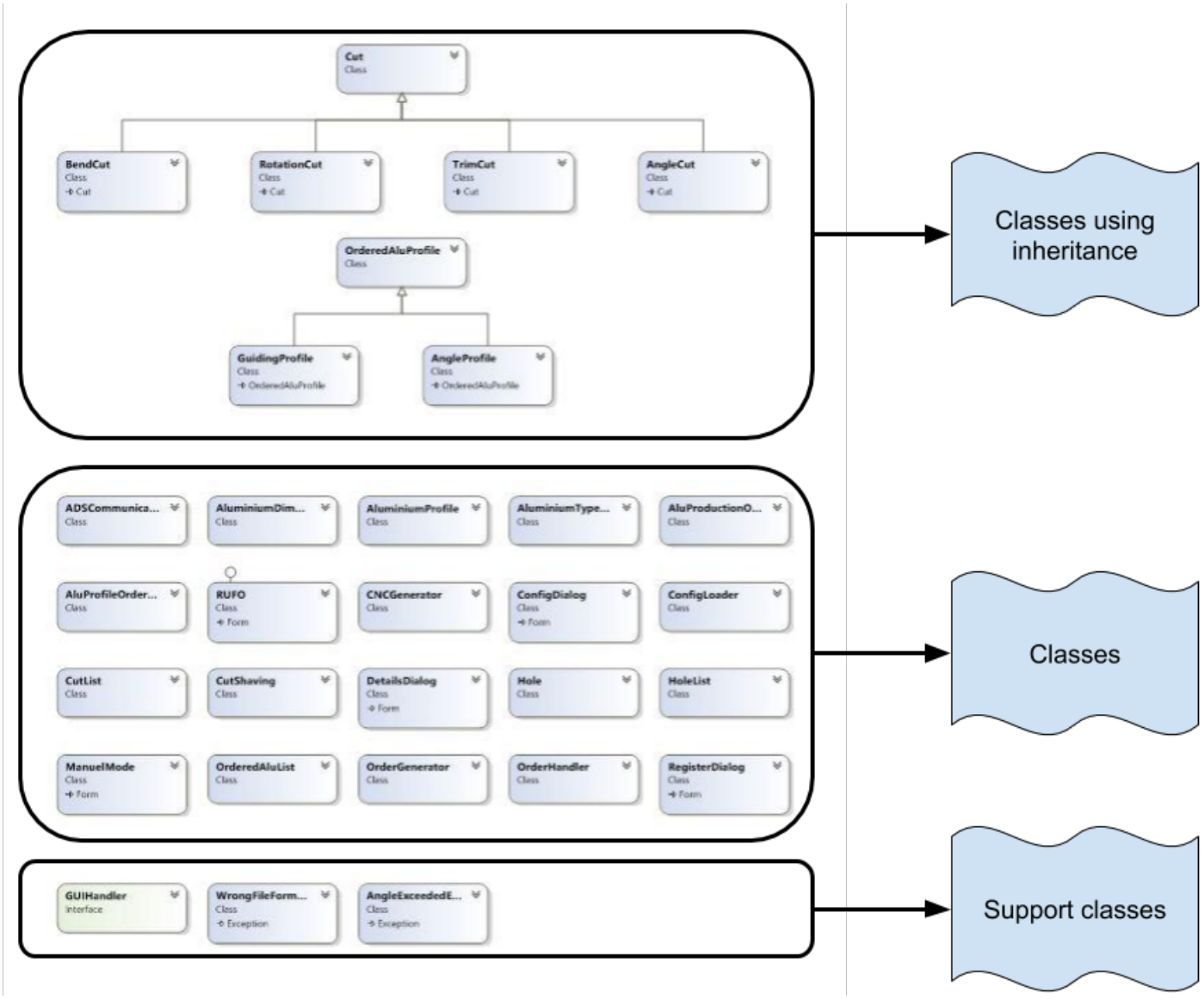


Figure 7.1: Server class diagram

The first task was to create the entity classes which mimics the different types of cuts and holes that can be included in the production of an aluminum profile. We identified 4 different types of cuts and one type of hole from the project specifications. As seen in figure 7.1 above, the class Cut is the super class which four sub classes inherits from, these sub classes are the actual cut's the production cell must be able to execute.

- BendCut: The BendCut represents a cut were the aluminum profile is partially cut so it can be bent into a 90 degree angle.
- TrimCut: The TrimCut represents a cut were the top end of the profile is trimmed of, usually in a +-45 degree angle.

- AngleCut: The AngleCut represents a cut with an angle in the horizontal axis shown, in figure 7.2. Where the angles are 90 degrees +- 45 degrees.

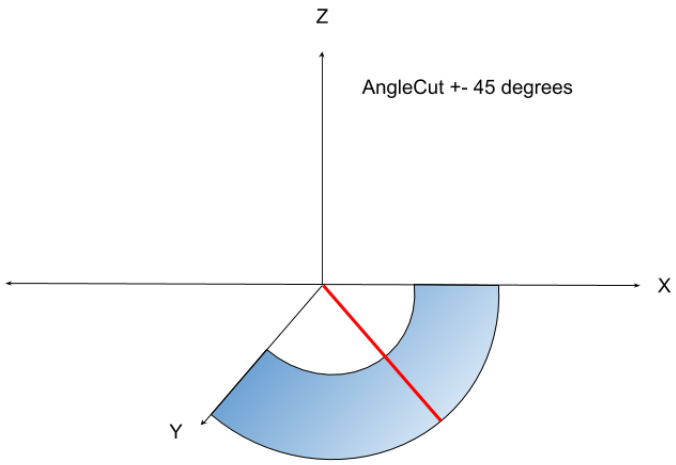


Figure 7.2: AngleCut

- RotationCut: The RotationCut represents a cut with an angle in the vertical axis, shown in figure 7.3. Where the angles are 90 degrees +- 45 degrees.

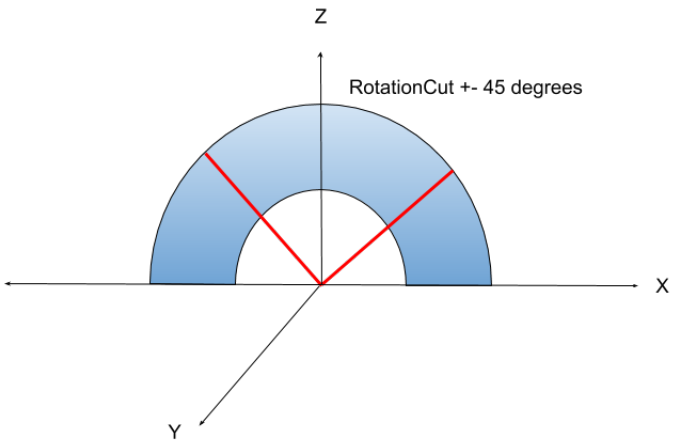


Figure 7.3: RotationCut

- Hole: A hole represents a hole in the profile. We view one side of the profile as a 2D surface meaning the x-axis is along the length of the profile and the y-axis is the height. There is a third parameter called the z-position, this is used if the profile has different holes on different sides of the profile. Meaning the z-position hold the profile side information.

These entity classes represents the business logic's foundation. Figure 7.4 below illustrates a simplified explanation of the server program logic. All the names in the figure can be found as a separate class in the class diagram 7.1 above, except the waste which is described as separate mathematical variable further down.

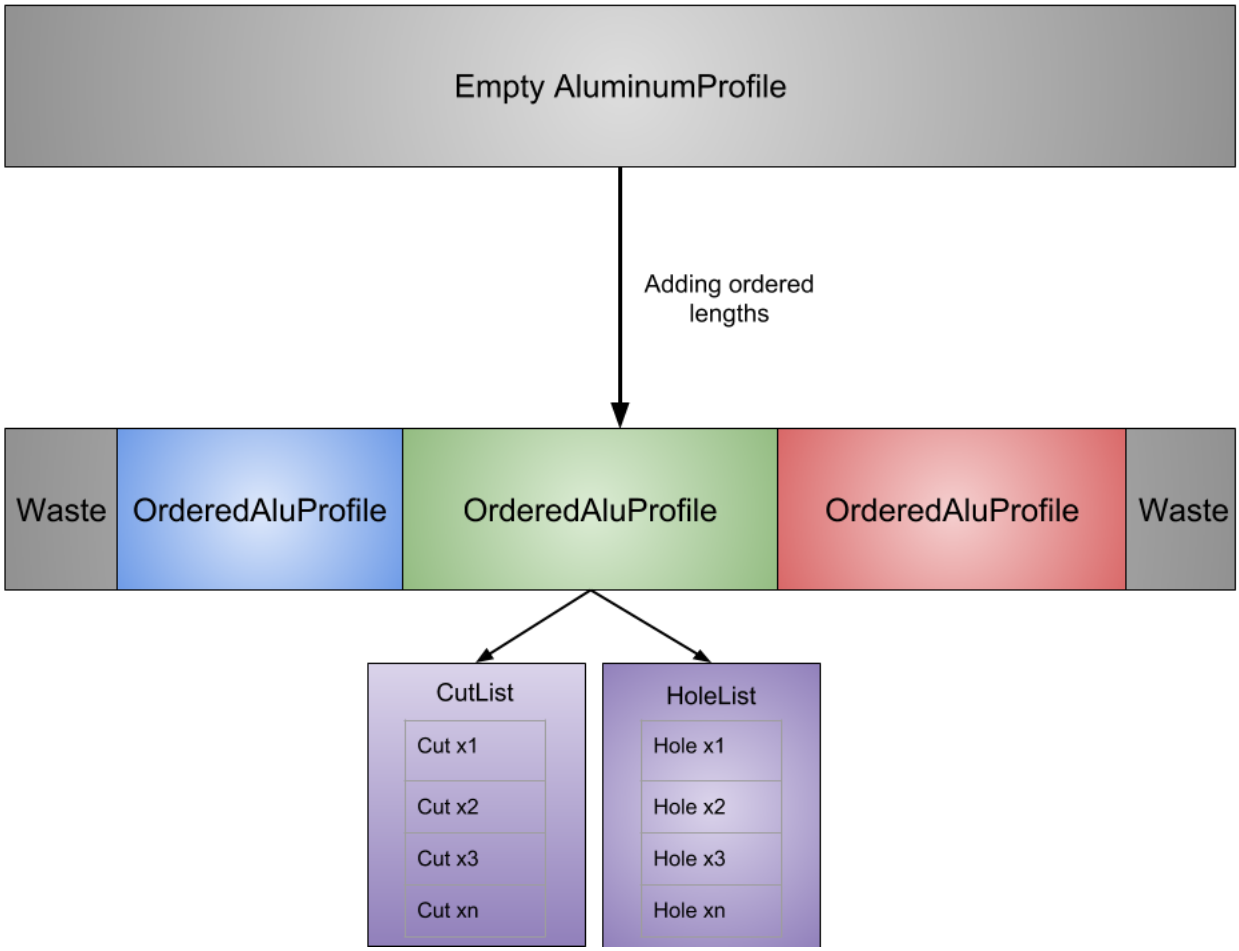


Figure 7.4: Business Logic

The logic represented in figure 7.4 can be explained in four steps:

- AluminumProfile:

  The aluminum profile is a profile from 0 to 6000mm, in RUFO's case from 4000 to 6000mm. Meaning RUFO now uses 4000mm profiles in their production but they want the opportunity to use 6000mm lengths.

- Receiving production orders:

  The server receives production orders, when the order is handled it creates OrderedAluProfiles. An OrderedAluProfile is either an angle profile or a guiding profile. Each OrderedAluProfiles holds two registers, one containing hole specifications and the other register holds cut specifications.

- Optimizing the order for production:

  After the OrderedAluProfile's are created they are added one by one onto the aluminum profile until it is full, by following a set of optimizing rules.

- Relaying the order data: The final step is to relay the order information from the previous steps, to the PLC. This is done by sending a self developed CNC-Code to the PLC.

While this is an oversimplification, it still explains the skeleton of the server logic. With this in mind the next four sub chapters will go into detail about how the server program works, by elaborating the four steps above and hos this is tied together in the GUI

7.2.1 Production orders

To generate orders it was decided, that the best practise would be for the server to receive a file with a specified format which contains all the information needed to produce the aluminum profiles. In the early stages of the project, this file format was supposed to be developed through a joint effort between RUFO and Viddal Automation. RUFO uses AutoCAD to design their flight cases, and have developed a technique to extract the aluminum profile information from the CAD drawing. The information extracted is called a BOM file, commonly known as a Bill of material file. Viddal Automation's task was to transform the BOM file into a readable .CSV file.

While developing the server software, we did not have time to wait for this file format to be developed. Instead we developed our own file format which enabled us to test the software. Initially our format was supposed to be the internal format used in the server software, and Viddal Automation would develop a parser which transformed the BOM.CSV to our format. Later on in the development stage, it became apparent that this solution was redundant. There is no reason to develop three file formats when the solution only requires one. So the natural choice was to scrap our own file format and take Viddal Automation's task to develop the "BOM.CSV" file with RUFO.

File format: BOM file

The development of the comma separated value or .CSV file was defined into two tasks. RUFO's responsibility was to create readable information from the figure 7.5 below. Containing a CAD drawing of a flight case stripped down to just the aluminum profiles. And ours were to decide how to arrange and use the data within the limitations of RUFO's software.

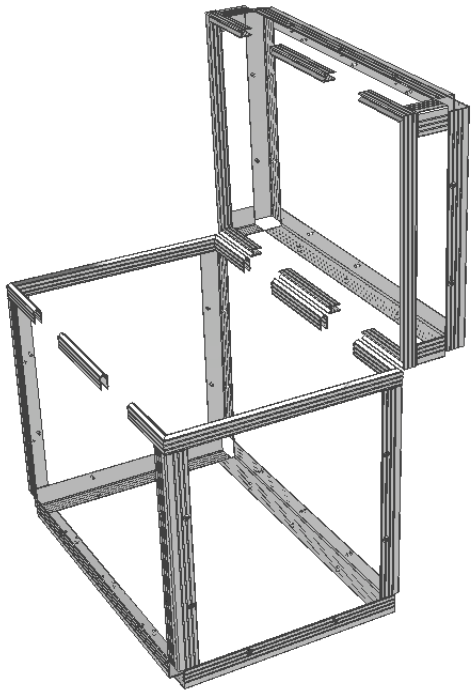


Figure 7.5: Stripped flight case
Copyright: RUFO AS

The BOM.CSV file went through several iterations before it was finished. But through the cooperation with RUFO the final result looks like this:

- BOM file setup: *ProdOrderNmb; ProdNmb; PartNmb; Quantity; ItemNmb; E; ER; EV; E; ER; EV; TX; TH; TR; LX; AX; AY; BX; BY;*
- ProdOrderNmb: Is the number referencing the product to a customer order.
- ProdNmb: Is RUFO's internal product number for a particular flight case.
- PartNmb: Is the profile name/type.
- Quantity: Is the quantity of profiles to be produced.
- ItemNmb: Is the item number the profile has according to the flight case drawings.

- E: Stands for an end cut and the position –> E100. Because each profile has two end cuts E, ER and EV are inputted twice in the .CSV file. Were the first set is for the first end cut and the second set for the last end cut.
- ER: Is the end cut rotation angle in the vertical axis and represents a RotationCut. The information includes the letters ER and a rotation value –> ER45.
- EV: Is the end cut angle in the horizontal axis and represents an AngleCut. The information includes the letters EV and an angle value –> EV90.
- TX: Stands for a TrimCut and the positioning of the cut –> TX150.
- TH: Is the height of the TrimCut meaning at what height the trim should end –> TH15.
- TR: Is the TrimCut's rotation angle in the vertical axis –> TR45.
- LX: Stands for a BendCut and its positioning –> LX275.
- AX: Stands for a Hole on the A side of the profile accompanied with position in the x-axis –> AX50.
- AY: Is the hole on the A sides positioning in the y-axis –> 9.
- BX: Stands for a Hole on the B side of the profile accompanied with position in the x-axis –> AX75.
- BY: Is the hole on the B sides positioning in the y-axis –> 9.

It is important to notice that some of these input variables are generic and some are specific, meaning the size and magnitude of information in the .CSV file will vary from order to order. In the OrderGenerator sub chapter we will give a more detailed description about what information the file contains and how it is parsed.

OrderGenerator

The OrderGenerator class is the part of the program which is responsible to read and parse the order information from the .CSV file. We will not get into the nitty-gritty on how the OrderGenerator does it job, but what the OrderGenerator needs to generate an order.

As mentioned above some of the information needed is essential to generate an order and the rest is generic. The essential information which every order needs is *ProdOrderNmb; ProdNmb; PartNmb; Quantity; ItemNmb; E; ER; EV; E; ER; EV*. Every order RUFO handles today uses *ProdOrderNmb; ProdNmb; PartNmb; Quantity; ItemNmb* as a part of their order system, so it

is logical to include this into our server software. The *E; ER; EV; E; ER; EV* part is also essential because each profile has a length and to achieve this length it needs two end cuts.

It is also important to mention that each order line represents a set amount of OrderedAluProfiles shown in figure 7.4 above. The parameter which decides how many OrderedAluProfiles that should be created within the specifications of the order line is the *Quantity* parameter. So an order line in the .CSV file which only contains the essential information looks like this:

- Essential order: *RUFO123;BL0323;6142M;2;34501;E0;ER45;EV90;E800;ER-45;EV90*

The rest of the information is considered generic, meaning you can have infinite amounts of TrimCut's, BendCut's and holes. The only restriction on this is RUFO's own software and the limits they want to set for each of these parameters. The reasoning behind this decision is that there is difference between the angle profile and guiding profile. An angle profile does not have trim or bend cuts just Rotation or Angle Cut's, represented as end cuts. An angle profile can also have holes on both sides. The guiding profile can have every cut but only holes on the A-side. But what they have in common is the essential information mentioned above. To further illustrate this at the end of this sub chapter we have some different order examples with different amounts of information.

- Guiding profile holes only:
  RUFO123;BL0323;6142M;2;34501;E0;ER45;EV90;E200;ER-45;EV90;
  AX50;AY15;AX100;AY10;AX150;AY15;

- Angle profile:
  RUFO123;BL0323;6142M;2;34501;E0;ER45;EV90;E800;ER-45;EV90;
  AX50;AY15;BX50;AY10;AX150;AY15;BX150;BY10

- Guiding profile one TrimCut and one BendCut:
  RUFO123;BL0323;6142M;2;34501;E0;ER45;EV90;E200;ER-45;EV90;
  TX0;TH20;TR45;LX200;AX50;AY15;AX100;AY10;AX150;AY15;

- Guiding profile two TrimCut and multiple BendCut:
  RUFO123;BL0323;6142M;2;34501;E0;ER45;EV90;E600;ER-45;EV90;
  TX0;TH20;TR45;TX600;TH20;TR45;LX100;LX200;LX350;AX50;AY15;AX100;AY10;AX150;AY15;

7.2.2 Production optimizing

After the order is received and generated the order needs to be optimized on to aluminum profiles to reduce the waste product to a minimum. This optimization is done using three classes in the server program, the CutShaving, AluminiumProfile and the AluProductionOptimizer class. Which will be described in detail in the upcoming three sub chapters.

CutShaving

The CutShaving class represents the part of the material which is removed by the thickness of the saw blade during a cut. As mentioned above every profile consists of two end cuts, meaning that you will by minimum lose one blade thickness in length if you don't implement the CutShaving into the optimizing calculations. The figure 7.6 below illustrates what the CutShaving is and the mathematics behind it, were X is the size of the cut shaving.

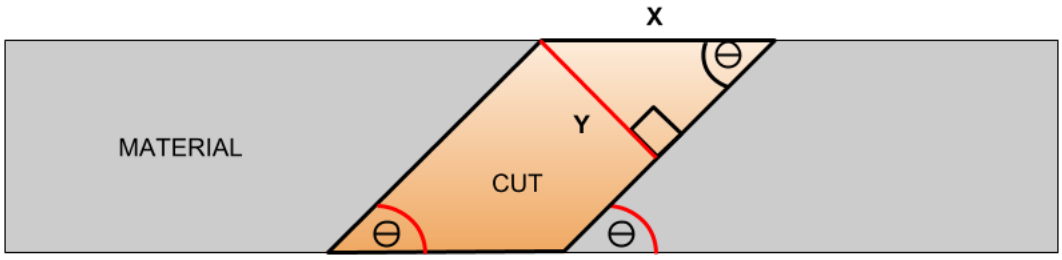


Figure 7.6: Cut shaving illustration

The calculation formula for the cut shaving is:

$$Y = X \cdot \sin\Theta$$
$$X = \frac{Y}{\sin\Theta}$$

This means that the cut shaving size is decided by the angle. Since the angle restrictions for this thesis is 90 degrees +- 45 degrees and our blade thickness is 2.8mm. The minimum cut shaving will be:

$$\frac{2.8}{\sin 90} = 2.8mm$$

And the maximum cut shaving is:

$$\text{-45 degrees: } \Theta = 90 - 45 = 45 \quad \rightarrow \quad \frac{2.8}{\sin 45} = 3.86mm$$
$$OR$$
$$\text{45 degrees: } \Theta = 90 + 45 = 135 \quad \rightarrow \quad \frac{2.8}{\sin 135} = 3.86mm$$

AluminiumProfile

Before describing how the optimization process works we need to explain some of the parameters the AluminiumProfile holds. An empty AluminiumProfile as shown in the figure 7.6 below holds the waste parameters which are essential for the optimization.

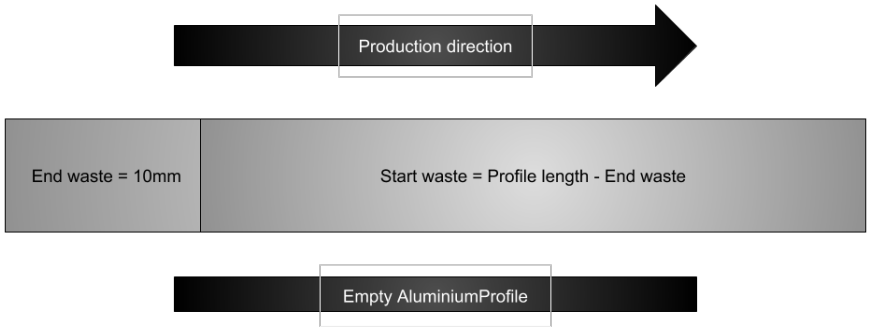


Figure 7.7: Empty AluminiumProfile

An empty AluminiumProfile is a profile that has not yet had OrderedAluProfiles added and optimized to it. The end waste parameter is a constant which always is 10mm in length. This is to makes sure that the profile always has room to do a clean cut at the end of the profile. Start waste is a generic value which is decided by the profile length minus the end waste. The start waste is essentially the amount of room left for OrderedAluProfiles.

AluProductionOptimizer

To optimize the production two different approaches were researched. Artificial intelligence in the form of genetic algorithms. The second approach is the one-dimensional cutting stock problem, both theories are described in the theory chapter under section 7.2.2. Both approaches could be used as a solution to optimize the production in our server software. The downside is that the implementation of both approaches is complex. Actually we concluded that for RUFO's current production capacity the genetic algorithm approach was to complex, and would require more work than it would yield as a viable solution. So we decided to develop a simpler but effective algorithm that would fit RUFO's needs, this algorithm is based on the one-dimensional cutting stock problem, which is to reduce the waste product to its absolute minimum.

The result of the optimizing is illustrated by figure 7.8 below. Were "EW" = end waste and "SW" = start waste. It is also only necessary to implement the CutShaving at the final cut of each profile, to prevent that the profile becomes shorter than its intended length. The AluminiumProfile is filled up until all the OrderedAluProfile's including CutShaving left is longer than the start waste.

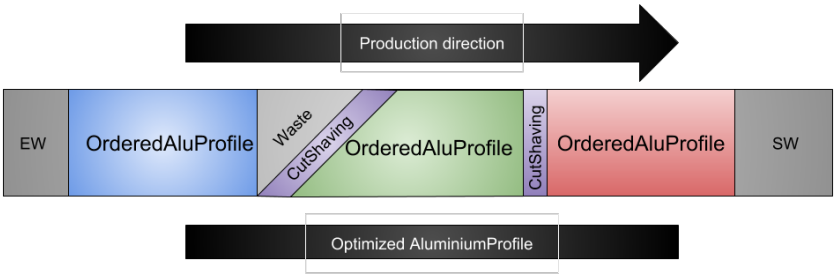


Figure 7.8: Optimized AluminiumProfile

To obtain the result in figure 7.8 we developed an algorithm which implements the factors mentioned above:

Algorithm calculations	
Symbol	Description
EW	The end waste length
CS_n	The CutShaving on the end of each OrderedAluProfile. $$CS_n = \frac{Y}{\sin\Theta_n}$$ Were Y is the blade thickness and Θ_n is the angle of the cut
OL_n	The length of the OrderedAluProfile
OSL_n	The length of the OrderedAluProfile including the cut shaving. $$OSL_n = CS_n + OL_n$$
n	Number of OrderedAluProfile's
TL	The total length of the OrderedAluProfile's. $$TL = \sum_{n=1}^{n} OSL_n$$
SL	The stock length of the stock profiles used in production
NS	Number of stock lengths needed for the production. $$NS = \frac{TL}{SL - EW}$$
NWS	Number of whole stock lengths needed for the production. For example if NS = 2,1 –> NWS = 3. You always round up to the closest integer
W	The minimum amount of waste. $$W = (NWS - NS) \cdot SL$$

Optimizing example:

$$EW = 10mm, \quad n = 5, \quad SL = 3000mm, \quad Y = 2.8mm$$

$$CS_n = \frac{Y}{\sin\Theta_n} \quad \rightarrow$$

$$CS_n = [\frac{2,8}{\sin 90}, \quad \frac{2,8}{\sin 90}, \quad \frac{2,8}{\sin 90}, \quad \frac{2,8}{\sin 90}, \quad \frac{2,8}{\sin 90}] \quad \rightarrow$$

$$CS_n = [2.8, \quad 2.8, \quad 2.8, \quad 2.8, \quad 2.8] \quad \text{mm}$$

$$OL_n = [1000, \quad 1500, \quad 1340, \quad 654, \quad 966] \quad \text{mm}$$

$$OSL_n = CS_n + OL_n \quad \rightarrow \quad OSL_n = [1002.8, \quad 1502.8, \quad 1342.8, \quad 656.8, \quad 968.8]$$

$$TL = \sum_{n=1}^{n} OSL_n \quad \rightarrow \quad TL = 5474mm$$

$$NS = \frac{TL}{SL - EW} \quad \rightarrow \quad NS = \frac{5474}{3000 - 10} = 1,83 \quad \text{Stock lengths needed for the production.}$$

$$NWS = NS \quad \text{Rounded up to closest integer.} \quad \rightarrow \quad NWS = NS = 1,83 = 2$$

$$W = (NWS - NS) \cdot SL \quad \rightarrow \quad W = (2 - 1,83) \cdot 3000 = 510mm \quad \text{Waste from this production.}$$

The example above shows how the AluProductionOptimizer class operates when it is optimizing an order. The first step is calculating how many whole stock length profiles that are needed to produce the order. By doing so, the minimum waste is also calculated. In this example 2 profiles are needed for the production, which together generates 510 millimeters of waste. The last optimization rule is that the OrderedAluProfiles added to an AluminumProfile is sorted in an ascending order, meaning the shortest profile is produced first.

Optimization is performed per order received and generated, meaning already optimized orders in the server software are not optimized together with new orders. The argument for this

approach is that OrderedAluProfile's scheduled for production at a set date might be moved to another AluminumProfile if the order is optimized when a new one is added. This might lead to OrderedAluProfile's needed for a production order in May is delayed to June. You can argue that the re optimization when a new order is received implies better waste reduction. But it also means longer lead times from order to finished product for the customer, which in return will cost RUFO more money than having a slight increase in waste product.

7.2.3 Relaying the order data

After the order is received and optimized, it is time to start the profile production. To start the production, order information has to be relayed from the server to the PLC. This is performed utilizing ADS communication and the CNCGenerator class.

CNCGenerator

The CNCGenerator's responsibility is to generate readable production instructions for the PLC from an optimized AluminumProfile shown in figure 7.8 in section 7.2.2. To give production instructions to the PLC we developed our own dialect/version of the G-Code which we call CNC-Code, as explained in the section 2.4.1 G-Code, most machines speaks their own dialect. To control our production cell we have developed 11 CNC-Code commands, summarized and explained in the list below:

CNC-Code:

- TI = Transport inn:
  Meaning the transport into the production cell.
  The command TI is followed by the length off the transport in millimeters.
  Example → TI200.

- TO = Transport out:
  Meaning the transport out of the production cell.
  This can be done in two ways, by sending TO without any transport value.
  By doing so the machine will transport maximum out.
  Or by sending with a transport value in millimeters.
  Example → TO200, or → TO.

- TC = TrimCut:
  Meaning a TrimCut has to be performed.
  The value following the TC command is the TrimCut height
  this is were the TrimCut ends in millimeters. Example → TC15.

- TR = TrimCut rotation:
  TR is the rotation angle the TrimCut shall be performed with.
  The value following the TR command is the angle in degrees
  Example → TR45.

- TC and TR:
  To perform a TrimCut, both the TC and TR command are needed. So a typical TrimCut command looks like this:
  Example → TC15 TR45.

- BC1 = BendCut:
  Meaning a BendCut has to be performed.
  A BendCut does not have a value. the "1" in BC1 indicates that it is a constant.
  Example → BC1.

- AC = AngleCut:
  Meaning an AngleCut has to be performed.
  The value following the AC command is the angle in degrees
  Example → AC22,5.

- RC = RotationCut:
  Meaning a RotationCut has to be performed.
  The value following the RC command is the angle in degrees
  Example → RC-45.

- M = Milling:
  Meaning a hole has to be milled.
  The M command is just used for production line two which produces the guiding profiles.
  The command is followed by a value describing the holes position in the y-axis.
  Example → M10.

- MA = Milling A side:
  Meaning a hole has to be milled on the A-side of the profile.
  The MA command is just used for production line one which produces the angle profiles.
  Example → MA15.

- MB = Milling B side:
  Meaning a hole has to be milled on the B-side of the profile.
  The MA command is just used for production line one which produces the angle profiles.
  Example → MB15.

- D = Done:
  Meaning the production is completed
  Example → D.

These linguistic commands are used to generate the CNC-Code, and in the next step we will dive into, how the CNC-Code is generated.

CNC-Code generation:

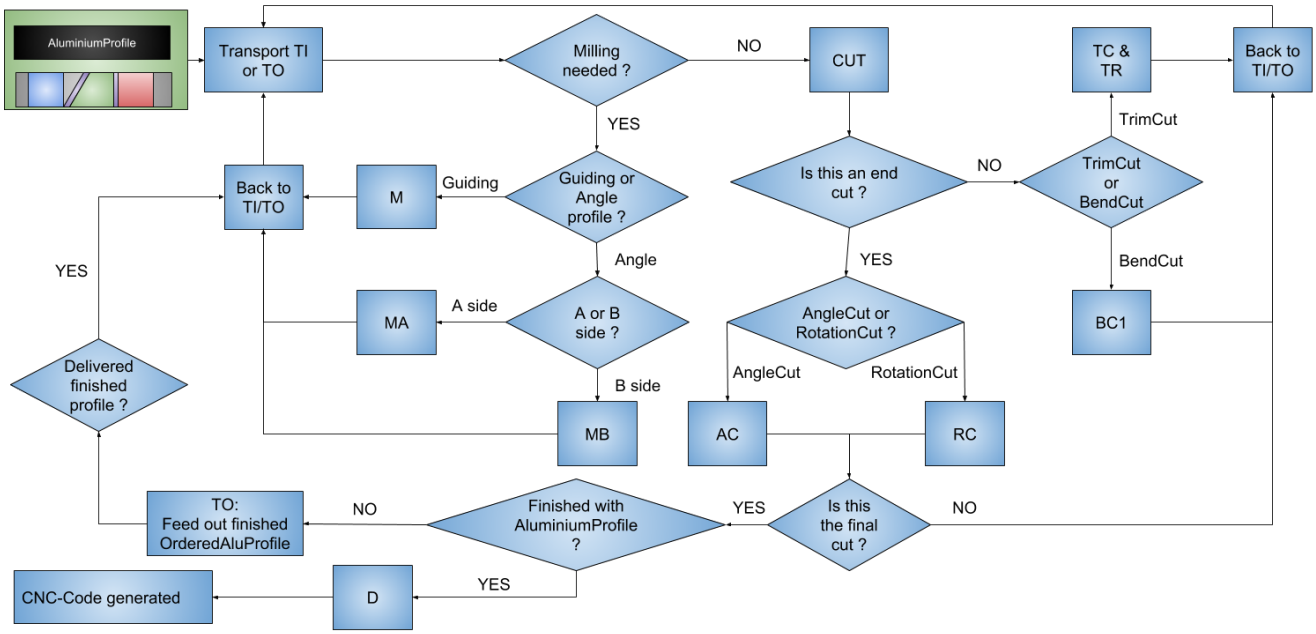


Figure 7.9: CNC-Code generation

Figure 7.9 above illustrates the CNC-Code generation for one AluminiumProfile. The generation process makes sure that the production always moves in a positive direction meaning "TI" and "TO" is followed by a positive value. The transport command in the top left corner is always followed by an action, either milling or cutting. Every time a cut command is reached and it represents a final cut the CNCGenerator checks whether or not this means that the production is finished. If the production is not finished, the final cut commands represents that a finished OrderedAluProfile seen in figure 7.8 needs to be collected and transported out. If the production is finished the last command "D" is generated, and the CNC-Code is finished.

The CNC-Code is generated into a list of instructions shown in figure 7.10. While this list could in theory be infinite, we calculated that the maximum amount of commands required for a 6000mm AluminumProfile was 400 commands. To calculate the list size we estimated that an operation would take place every 50mm. Meaning $\frac{6000}{50} = 120$ transport commands, since an action command is added after each transport command the number of commands is: $120 \cdot 2 = 240$ CNC-Code commands. We rounded the 240 commands up to 400 to ensure that the list is never maxed out.

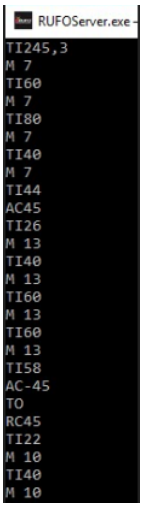


Figure 7.10: Generated CNC-Code

The "Transport TI or TO" block in the top left corner of figure 7.9, means that the production transport is either done by the gripper for feeding inn, represented by command "TI" or the gripper for collecting represented by command "TO" described in section 6.2.1. A variable representing the maximum transport distance allowed by "TI", is used to decide which command to utilize. If the total transport done by "TI" reaches the maximum transport distance allowed, then the "TO" command takes over the rest of the transport until the production is finished.

There are two more variables inn addition to the maximum transport distance allowed, which are essential for the CNC-Code generation. These variables represents the distance from a set origo to the center of the hole tool and the distance to the saw station. Figure 7.11 below illustrates how to measure these variables, where X_h is the hole station offset and X_c is the cut station offset. These variables determines the transport distance which have to be traversed before a cut or mill command.

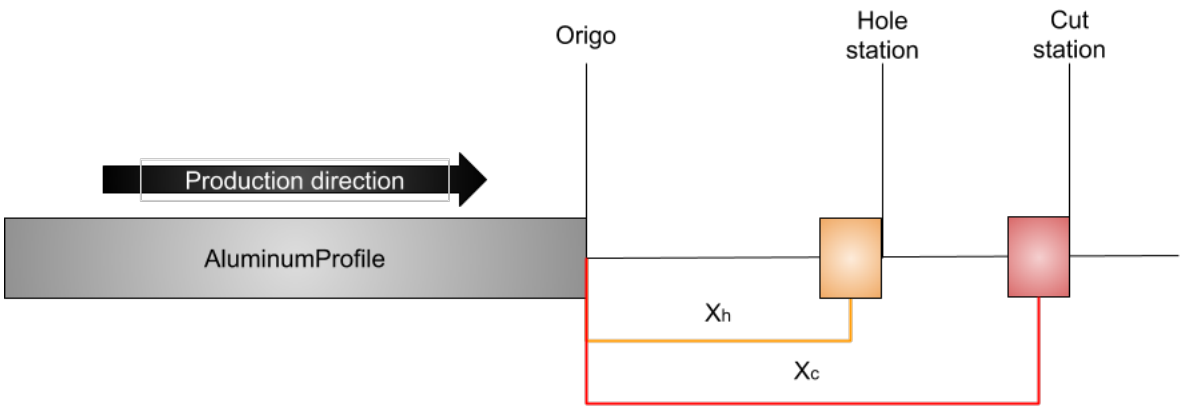


Figure 7.11: CNC generation offsets

The variables explained above determines the production flow, and therefore essential to generate the CNC-Code. These variables are adaptive, meaning the maximum transport distance allowed is decided by the length of the feed lines into the production cell. X_h and X_c are determined by were origo is and the measured distance from origo to the representative stations. Because of this, the variable values can be manipulated in the GUI for easy production cell configuration.

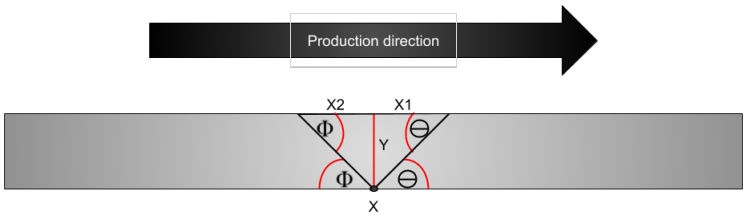


Figure 7.12: Angle offset

When automating cutting with varying angles its important to define the end position for the cut. The end position is constant for all angles, if not implemented it will result inn an error on the profile length in some situations. In figure 7.12 this end position is named X. If the angle is negative at the first end cut or positive on the final end cut as illustrated in figure 7.13, then X1 and X2 from the figure represents the offset which needs to be added to the transport to ensure correct profile lengths.

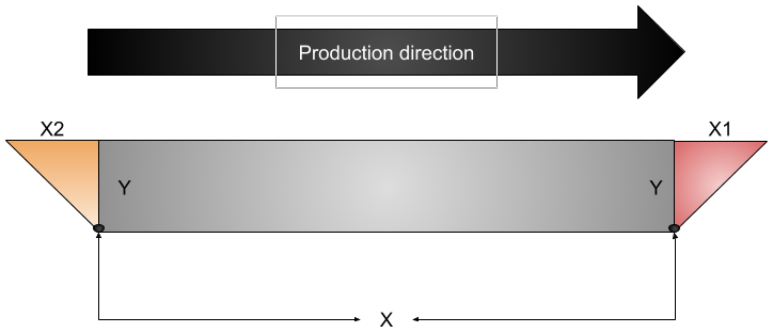


Figure 7.13: Angle offset two

The calculation for these offsets are elaborated from figure 7.12 above:

If the angle θ $\theta < 0$ Then cut offset X1 is

$$X1 = -Y \cdot tan(\theta)$$

If the angle ϕ $\phi > 0$ Then cut offset X2 is

$$X2 = -Y \cdot tan(\phi)$$

ADSCommunication

After the CNC code is generated the final step is to send the array containing the CNC-Code shown in figure 7.10 to the PLC. This is done developing a class utilizing the ADS communication library provided by Beckhoff explained in section 2.4.2. The ADSCommunication class has other responsibilities than just relaying the CNC-Code, its full responsibility is to handle all communication done between the server software and the PLC.

7.2.4 GUI

Everything explained above from receiving an order, optimizing, generating CNC-Code and relaying the instructions to the PLC is handled by the user interface. To do so the GUI is split into several subsections which each has its own specific responsibility.

OrderHandler

The order OrderHandler's responsibility is to handle everything concerning receiving, optimizing and generation of orders. The OrderHandler accomplishes this by listening to a folder called Orders shown in figure 7.14. The listener deletes every file type except .csv files immediately, if added to the folder. If the file is a .csv the OrderHandler controls that it has the correct file format defined in section 7.2.1 Production orders. Should the file contain the correct format, an order is generated, optimized and added to the GUI. This is illustrated in figure 7.15 and 7.16.

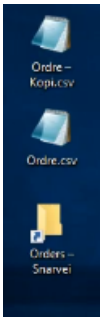


Figure 7.14: Folder listening for orders

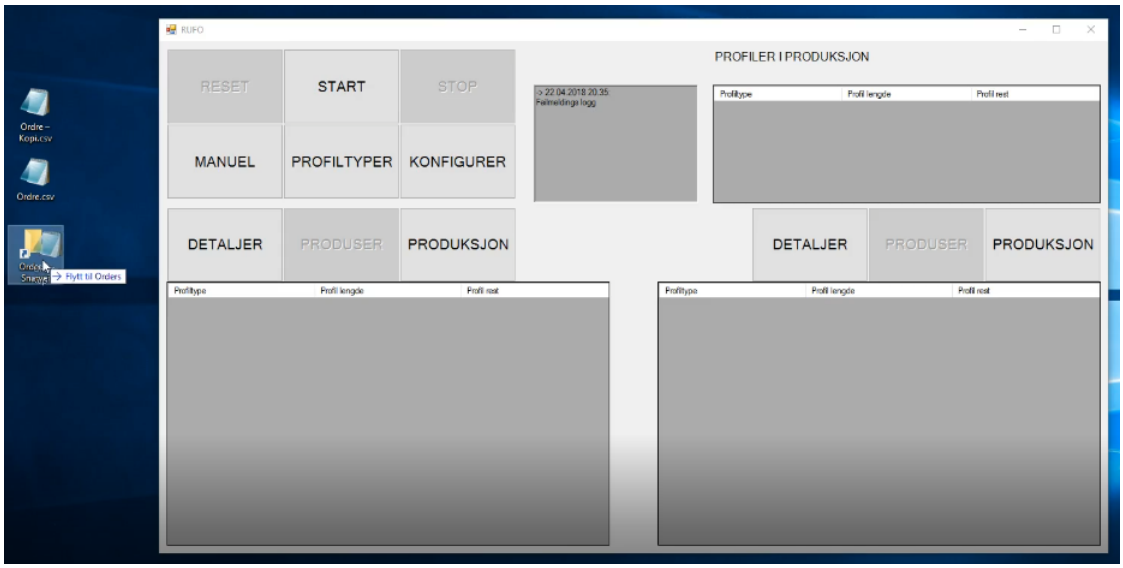


Figure 7.15: Receiving an order

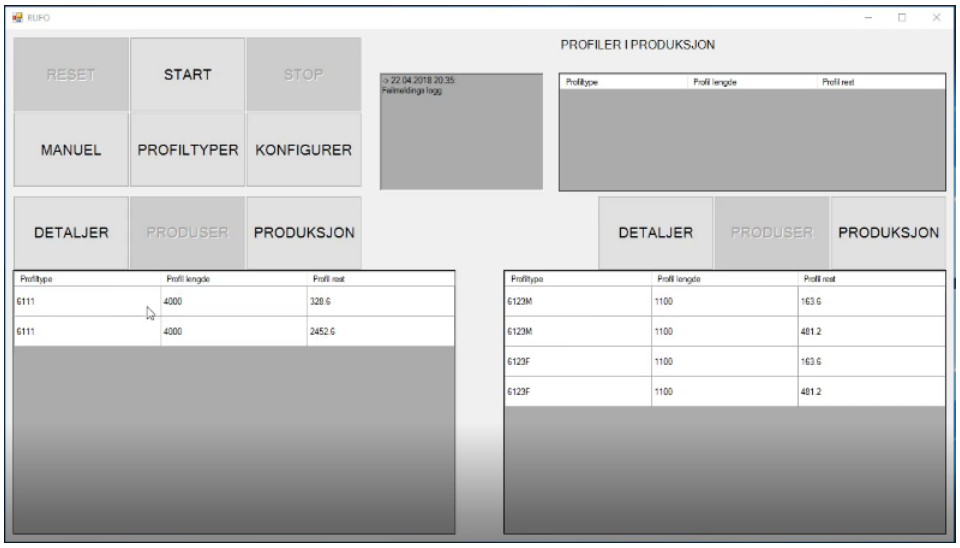


Figure 7.16: Generated and optimized order

Now that we understand the part of the software which handles the receiving and generation of orders, it is time to move on to the GUI's functionality. By referring back to figure 7.16 we can see that the GUI consists of two lists. The one on the left contains the angle profile orders for production line one and the one the right contains the guiding profile order for production line two. Each row in the list represents an AluminiumProfile optimized for production, where you can see the profile type, the length of the stock profile and how much waste each AluminiumProfile will produce. When clicking on the "DETALJER" button a window containing the details about the chosen AluminiumProfile production appears shown in figure 7.17. This window contains information about the OrderedAluProfile's added to the AluminiumProfile.

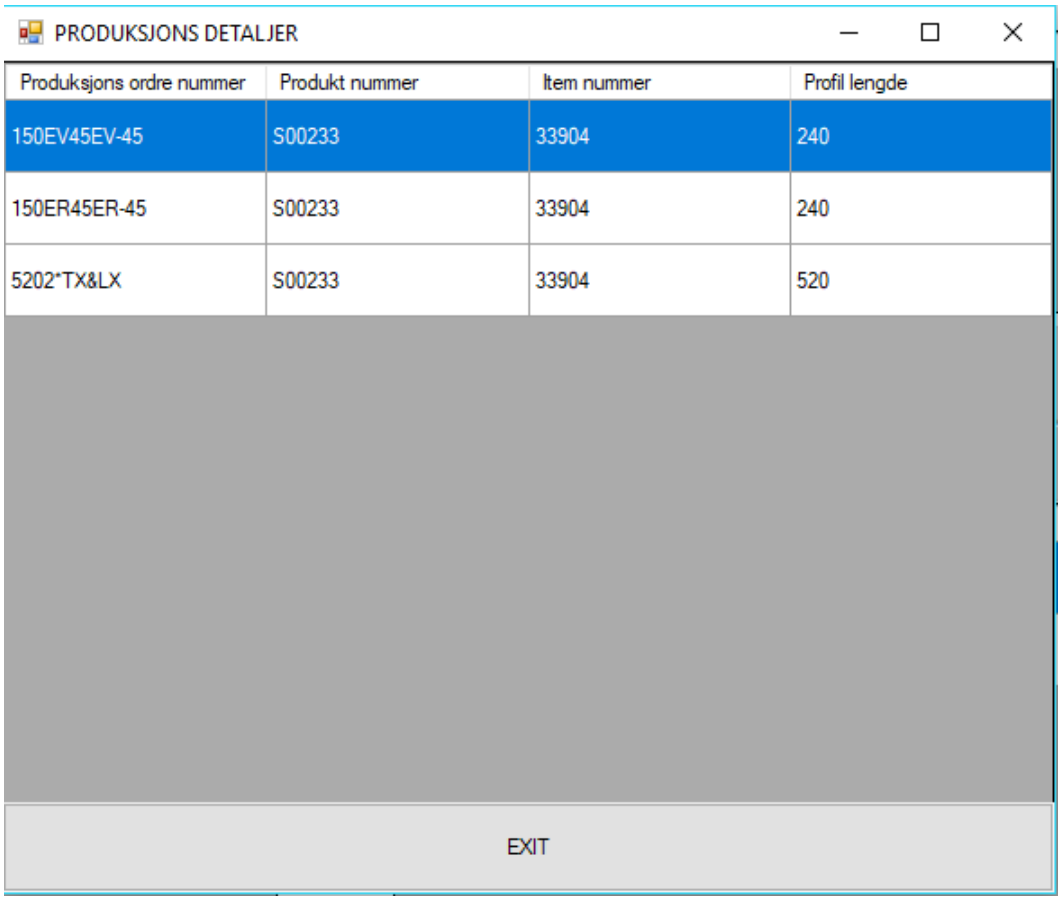


Figure 7.17: AluminiumProfile production details

The three buttons below the "START" button in figure 7.16 is used for testing and configuring of the server and production cell. The buttons "PROFILTYPER" and "KONFIGURER" is explained in the system variables and control sub chapter below. The "MANUEL" button enables manual control of the production cell.

System variables and control

The GUI has accessibility to variables used for order generation, production optimizing and CNC-Code generation, which are crucial for the performance of both server and production cell. The "PROFILTYPER" and "KONFIGURER" enables access to these variables. Both buttons opens their own respective window containing information which can be read, changed and deleted. The windows get their information from .txt, meaning changes to these files are saved even though the application shuts down.

The "PROFILTYPER" button opens a window containing the different types of profiles used in the production, as shown in figure 7.18 below.

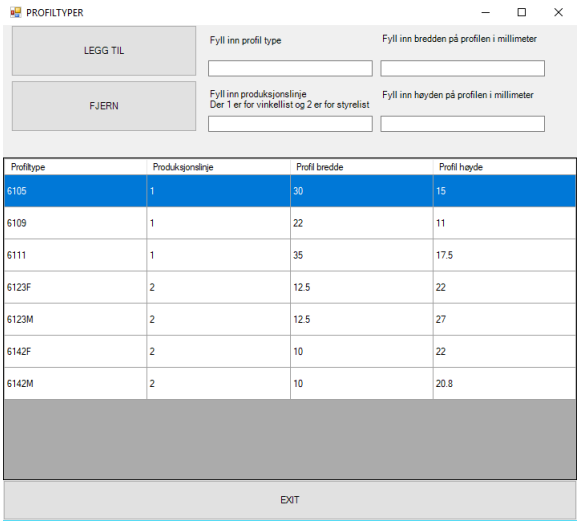


Figure 7.18: Profile types register

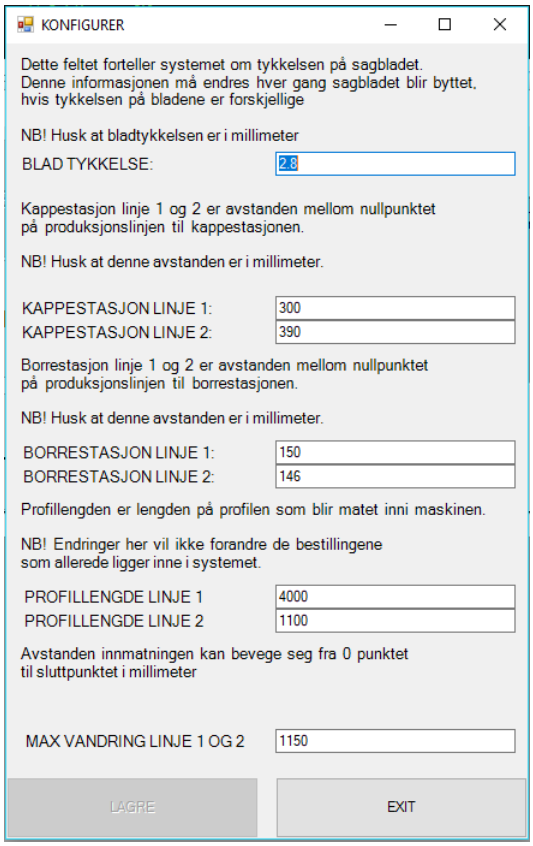


Figure 7.19: Accessing the config file

The "PROFILTYPER" window is used to add and remove profile types used in the production. When generating an order, the OrderGenerator class compares the profile types in the order with the PartNmb parameter in the BOM.CSV file. If the PartNmb in the order does not match any of the profile types in the "PROFILTYPER" register, the order is not generated. By making this information an editable file, RUFO can be much more flexible in their production regarding changes in what profiles they use.

When clicking the "KONFIGURER" button the user get access to the configuration file as shown in the figure 7.19 above. This window gives the operator the ability to read or change the variables used by the CNCGenerator to calculate the transport such as the milling offset, sawing offset and the maximum transport distance variable. The operator can also affect the optimization algorithm with the saw blade thickness used to calculate the CutShaving and the length of the stock AluminiumProfile. This is also added to give RUFO flexibility in the production but also to handle wear and tear in the production cell.

Production

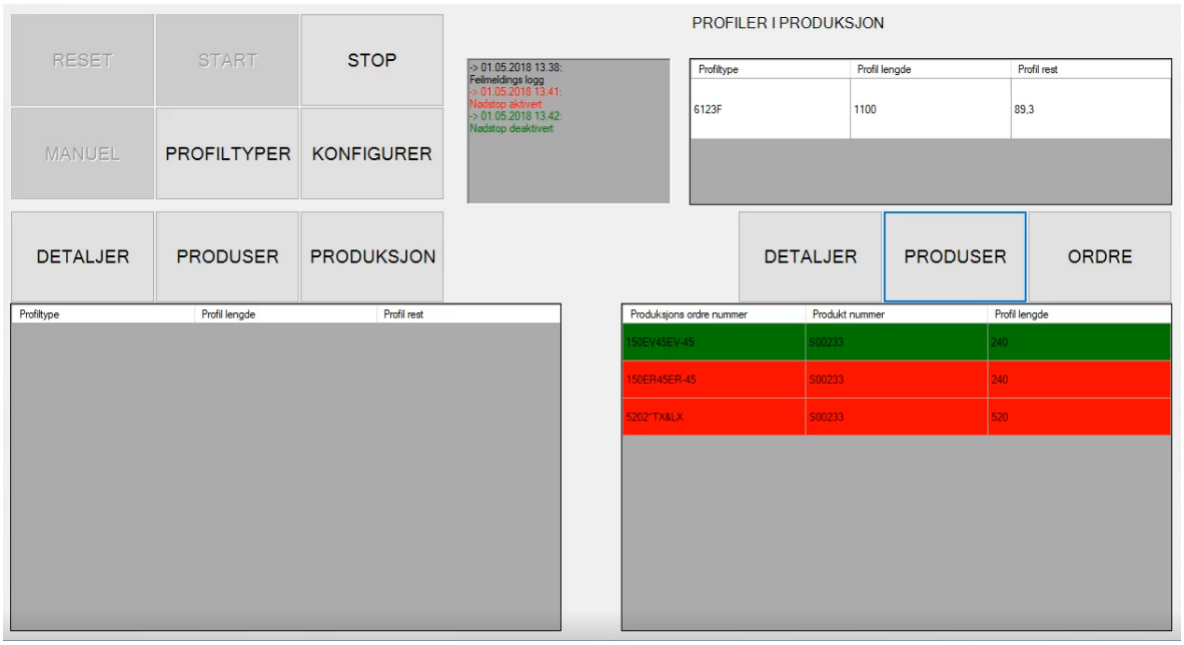


Figure 7.20: GUI in production mode

This section explains how to start the production and how the GUI/Server operates during production. The figure 7.20 above shows how the GUI looks in production mode, meaning an AluminiumProfile has been sent to production. The green row in the list on the right side means that this particular OrderedAluProfile is produced and collected and red rows are still in production. The following steps explains how to start the production:

- 1: Press the "START" button. The production cell will now calibrate itself.
- 2: If emergency stop or the "STOP" is pressed.
  Press "START" to continue" and "RESET" to reset the production cell.
- 3: Chose the profile you want to produce from the list, seen in figure 7.16.
- 4: Press the "PRODUSER" button to start the production.
  Each production line has its own "PRODUSER" button. Meaning the operator can produce angle and guide profiles simultaneously
- 5: Wait until production is complete.
- 6: Repeat.

7.3 PLC

In contrast to the server software, the PLC program was severely affected by the redefinition of the thesis in section 5. The design for the finished production cell is needed to develop the PLC program for it, because the design holds key elements like sensors, pneumatic and other control hardware. These key elements are essential for developing the control software. This means that the PLC program needs to be redeveloped for the actual production cell, however elements from the prototype PLC program can be repurposed.

7.3.1 HSE - Health Safety Environment

The prototype has two safety features which are handled by the PLC program. The emergency stop button is located on the KR-6 robot controller, and when enabled stops the KR-6 robot, PLC program and server. Another layer of security is the cage surrounding the prototype and KR6-robot, if the cage door is open the emergency stop is activated.

7.3.2 PLC - KUKA communication

Communication between a Beckhoff an KUKA robots is done through an bridge module from Beckhoff called EL6692. The module can be added either on the PLC integrated into the KUKA robot cabinet or the standalone PLC. For our project we added the module to the integrated KUKA PLC and connected the PLC's together using EtherCAT communication. Configuration the communication between a PLC and KUKA using EtherCAT is a complicated process. Therefore we have created a step by step guide, this guide can be found inn appendix A..

7.3.3 Stepper calibration

To control and calibrate the stepper motors, we included TwinCAT 3's motion library. This library contains everything needed to control, calibrate and regulate motions (Beckhoff, 2018b). Through using the pulse train module EL2521, the stepper motor can be controlled, were one pulse equals one step. Calibrating stepper motors using EL2521 and the motion library is a complicated process, which required days of work and help from Beckhoff Norway. Therefore we created an appendix explaining how to calibrate the steppers inn appendix B.

7.3.4 Program

The redefinition lead to the construction of prototype, which only includes production line two for the guide profiles. Consequently this means that the program for production line one is not developed. When developing the PLC program, we tried to generalize the process so more of the program is reusable for the finished production cell. Before developing the PLC program we had to decide on the programming approach, where the choice was between ST - structured text using switch case or SFC - Sequential Function chart. We chose using the SFC approach based on a few key points:

- It is a graphical programming approach. Where as ST is text based programming.
- SFC is easier to debug than ST.
- It is easier to follow the production flow using SFC.

Although SFC is a great programming approach, one of its weaknesses is restructuring a finished SFC layout. Restructuring a finished program is much easier using ST and switch case. The figure 7.21 below shows the SFC program layout.

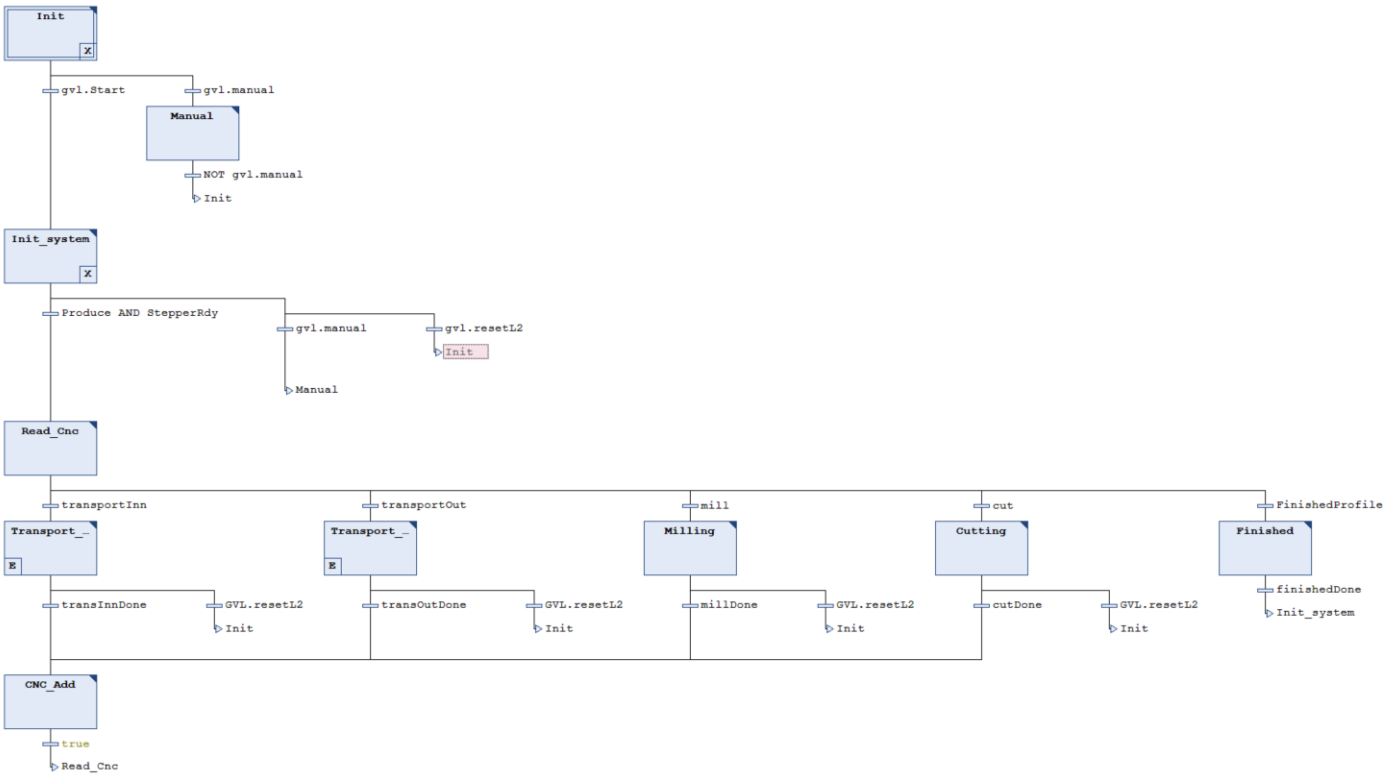


Figure 7.21: SFC of production line 2

When the production cell is inactive the program operates inside the "Init" box in the top left corner of the figure. If the operator presses the "MANUELL" or "START" button in figure 7.16 the operator enables manual control or starts the production cell calibration as explained in section 7.2.4. These commands is carried out by the "Manual" and "Init_system" boxes in figure 7.21. The "Init_system" production calibration means that production cannot start before all stepper motors are moved to their home position, and the pneumatic grippers and holders are set to their default mode.

The program is active in the "Init_system" box until production orders are received from the server, these production orders include the CNC-Code. When the CNC-Code is received the production begins. The first box "Read_Cnc" reads and parses one line of code during each iteration. The iteration process continues until the production is done. Based on the information from the CNC-Code line, one action is performed. The five boxes second to bottom in figure 7.21 represents the linguistic CNC-Code commands explained in section 7.2.3, were from the left.

1. Transport_Inn is triggered by command "TI".
   This box handles the transport inn.

2. Transport_Out is triggered by command "TO".
   This box handles the transport out and finished profile collecting.

3. Milling is triggered by command "M", "MA" and "MB".
   This box handles the hole generation.

4. Cutting is triggered by command "AC", "RC", "TC TR" and "BC1".
   This box handles cutting by relaying the cutting information to the KR-6 robot.

5. Finished is triggered by command "D".
   Terminates the production when it is finished. The program jumps back to "Init_system"

By using a SFC programming solution, the program have been generalized. This SFC structure requires minimal restructuring for the final production cell and to be operational for both production lines. The code however inside each SFC box might demand a more thorough review after the production cell is finished, to identify what changes are needed.

7.4 Robot

As discussed thoroughly in section 4.1 the KR-6 robot is used to cut the profiles. While the prototype only incorporates production line two, the KR-6 program structure includes both production lines and can be used for the finished production cell. The robot will however need some reprogramming for the finished product, which includes the coordinate systems and robot movements. Because these programming elements is dependent on the robots placement.

Before programming, the KR-6 needs calibration. This calibration includes mastering the axis and defining coordinate systems relevant to solve the problem.

7.4.1 Mastering the axis

Mastering the robot axis calibrates each joint's home position. Mastering the axis correctly is the determining factor for the coordinate system and robot accuracy. Each joint has a small pin located at the joint coupling. During the mastering the joint are adjusted until the pin reaches its bottom location, when the bottom is reached the joint is at its home position.

7.4.2 Coordinate systems

In order for the robot to perform every cut configuration, five coordinate systems were required which are illustrated in figure 7.22 and 7.23. Three of the coordinate systems were TCP's and the last two were base systems.

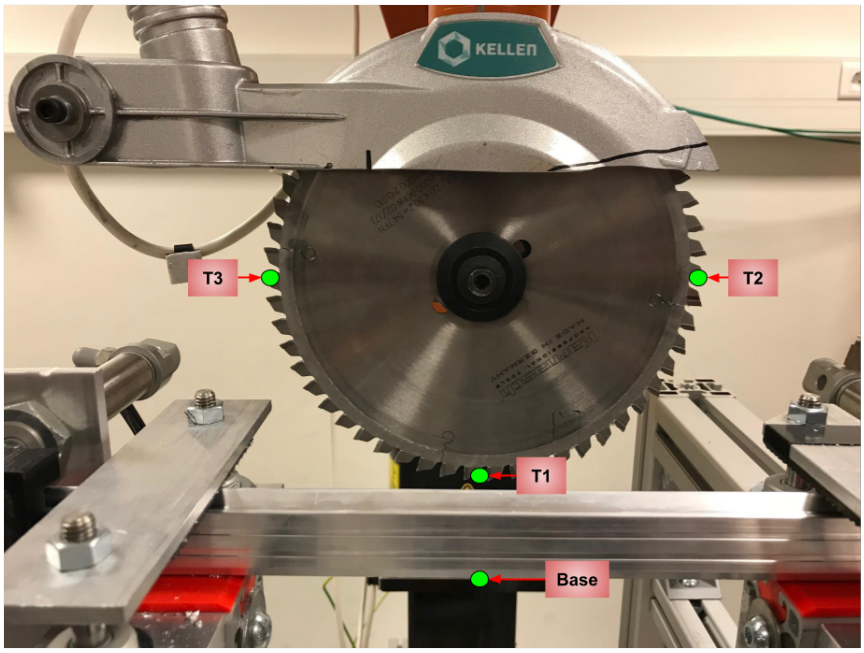


Figure 7.22: Coordinate systems

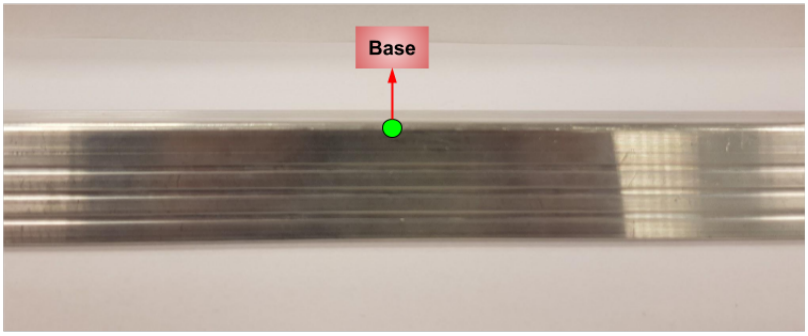


Figure 7.23: Angle profile base coordinate

Base coordinate systems

There are two base coordinates, where the base coordinate in figure 7.22 is the system for production line two, and figure 7.23 for production line one. These base coordinate is used to measure the cut offset used in the CNCGenerator illustrated in figure 7.11 in section 7.2.3. The only difference between these coordinate system is that for the guide profile the base is located at the bottom of the profile, and on the angle profile it is located at the top.

TCP

There are three TCP's calibrated to ensure that the robot can perform all cut configurations. T1 in figure 7.22 is utilized for both production lines and performs the RotationCut, AngleCut and TrimCut. T2 and T3 enables the robot to perform the BendCut.

7.4.3 Motions

Programming the robot to move from one position to another is performed using one of three commands. Which command to use is dependent on the type of movement and task the robot should perform. The commands is as follows:

- PTP motion:
  PTP or Point to point motion is used when the only important parameter is the start and end point. When using this motion, the robot chooses its own path to reach the end point, which makes PTP the fastest robot motion. Programming PTP is done using one of two ways. The first one is telling the robot what angle each joint should have in the end point. The second one utilizes the Cartesian coordinate system (X,Y,Z,A,B,C,S,T) S and T is the saturation and turn. (KUKA, 2003)

- CIRC motion:
  CIRC or Circular motion is used when the task requires a circular motion to reach the end point. To move circular the robot must have a start, endpoint and a point between to decide the circular path. (KUKA, 2003)

- LIN motion:
  LIN or linear motion motion is used when the task requires the robot to move linear relative to the TCP or base. To perform a LIN motion the robot must first perform a PTP motion to ensure that the linear motion always start in the same position. (KUKA, 2003)

7.4.4 Robot program

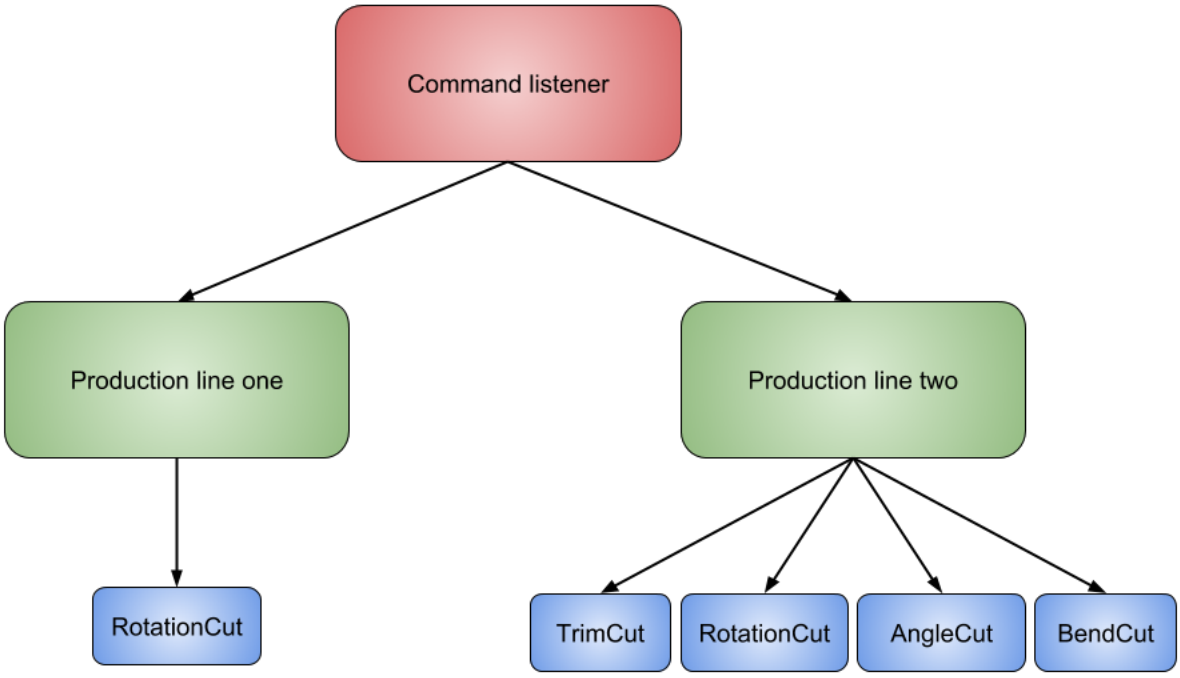


Figure 7.24: Robot program structure

The robot is programmed using a hierarchy structure illustrated in figure 7.24 above. Where the command listener waits for a cut command from the PLC. Production line one and two represents the base coordinate system where the cut is performed and the blue boxes at the bottom represents the cut type.

To perform all cuts a linear motion is used. For the RotationCut and AngleCut the TCP T1 in figure 7.22 has to move through the base location to cut through the profile. Performing a TrimCut is done when T1 ends at a set location above the base location, this ensures that just the top is trimmed off.

All cuts except the BendCut uses TCP T1. The BendCut requires a partial cut through the profile, therefore the TCP T2 and T3 in figure 7.22 is used, T2 and T3 must end a set distance away from the base location when performing the motion.

When programming the robot motion, we had to ensure that the robot could perform the

motion without crashing. The horizontal angles for the AngleCut required no interference to the robot motion. However the vertical angle used in the RotationCut and TrimCut, required that the saw tool rotated 180 degrees in the horizontal axis for angles $\theta < 0$ otherwise the saw engine would crash in the production cell.

The BendCut is the most complicated cut. Performing such a partial cut requires two cuts. These two cuts are illustrated in a birds eye view of a profile in figure 7.25. To bend the profile into a 90 degree angle, it requires two incisions with opposite angles of 45 degrees.

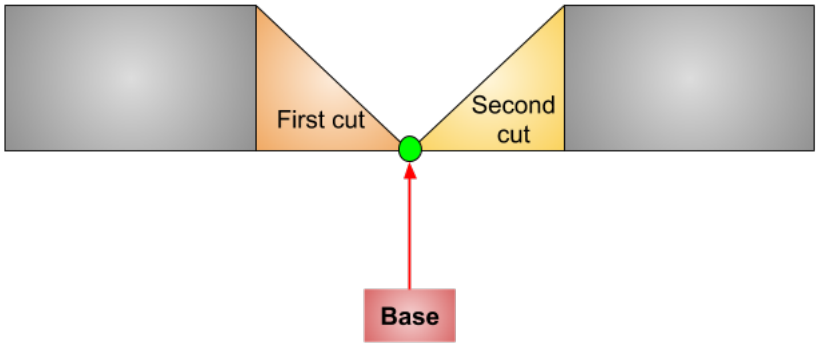


Figure 7.25: BendCut birds eye view

Figure 7.26 below illustrates the BendCut process. The cut required two TCP's T2 and T3, because the second cut required a 180 degree rotation in the horizontal axis to prevent crashing into the production cell.

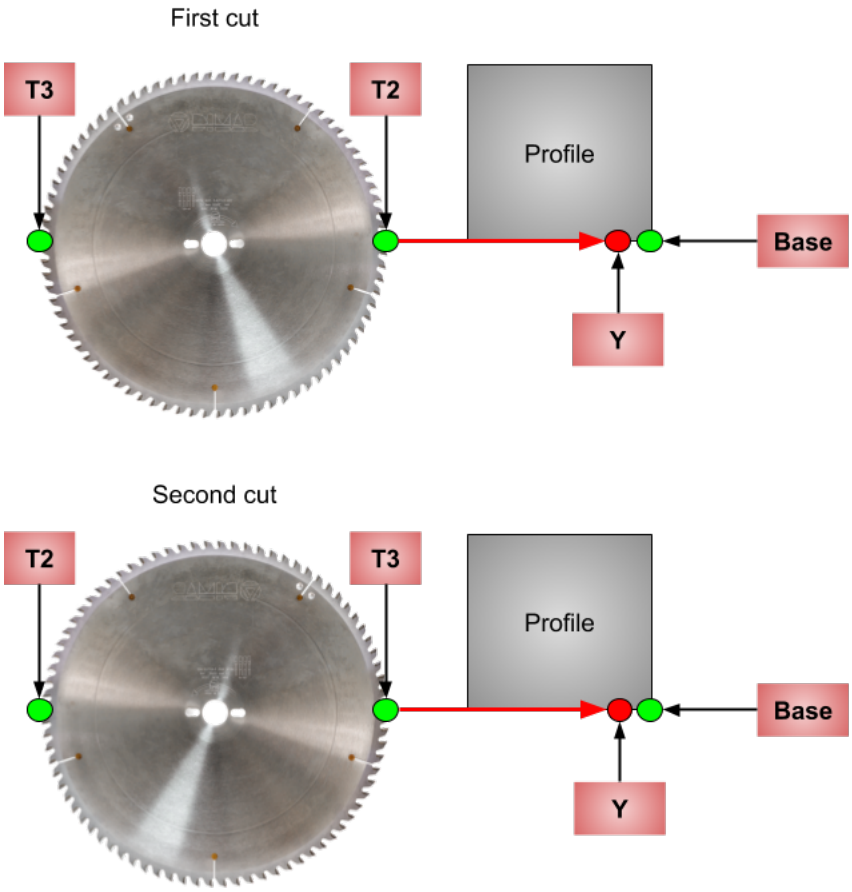


Figure 7.26: Performing a BendCut

Chapter 8

Testing and results

Immediately after the prototype was operational the production testing began. The testing stage of the thesis was used to identify prototype and software related errors and measuring the prototypes accuracy, quality and production lead time.

8.1 Accuracy

Testing the prototypes accuracy was conducted by creating test orders and comparing the order specifications and production result. These tests was used to identify errors in software and hardware, by solving these errors we calibrated the accuracy of the production cell.

8.1.1 Hole station

The first sett of tests were used to check the accuracy of the hole station. Testing the hole station measures two parameters, the holes x and y position. Figure 8.1 below shows the result of the four most important tests. The common denominator for these tests were that the accuracy both in y and x axis were affected greatly by the structural integrity of the hole station.

Figure 8.1: Hole station calibration

Table 8.1 and 8.2 shows how we collected the data after each test and thus identifying areas for improvement and errors. Where test four were the final test before implementing cutting into the tests.

Table 8.1: Hole station test one

Test 1	**Hole**	**x-axis (mm)**	**Actual x-axis (mm)**	**Error (mm)**	**y-axis (mm)**	**Actual y-axis (mm)**	**Error (mm)**
Transport inn	1	10	7	3	9	10	1
Transport inn	2	30	27	3	9	11	2
Transport inn	3	100	95	5	9	9,5	0,5
Transport inn	4	150	147	3	9	12	3
Transport inn	5	200	195	5	9	10,5	1,5
Transport inn	6	620	616	4	9	8,5	0,5
Transport out	7	650	647	3	9	8,5	0,5
Transport out	8	680	676	4	9	8,5	0,5

Table 8.2: Hole station test four

Test 4	**Hole**	**x-axis (mm)**	**Actual x-axis (mm)**	**Error (mm)**	**y-axis (mm)**	**Actual y-axis (mm)**	**Error (mm)**
Transport inn	1	10	10	0	9	9	0
Transport inn	2	30	30	0	9	9,5	0,5
Transport inn	3	100	100	0	11	11,5	0,5
Transport inn	4	150	149,5	0,5	11	11,5	0,5
Transport inn	5	200	201	1	9	10	1
Transport inn	6	620	620	0	9	9,5	0,5
Transport out	7	650	649	1	9	9,5	0,5
Transport out	8	680	678	2	11	11	0

The errors and areas for improvement which we identified and fixed through the tests where:

- The hole stations structural integrity:
  Most of the error in both x and y position shown in table 8.1 and test 2 and 3 not shown in the table was caused by the structural integrity. By reinforcing the hole station structure and locked the entire production cell to the KR-6 robots foundation to eliminate the vibrations caused by the hole station, the error were reduced to the result in table 8.2.

- Backlash on input guideline stepper coupling:
  The motor coupling used to link the stepper motor to the input guideline had backlash. Fixing this issue also affected the x position accuracy shown in table 8.2

- Error in CNCGenerator:
  In section 7.2.3 the maximum travel distance for transport inn is explained, were transport out must take over when this distance is reached. This happens in all the tests done, and is showed in the tables 8.1 and 8.2 above for hole 7 and 8, which are positioned using the output gripper. The error identified for the transition between transport inn and out is shown in figure 8.1 above on the second to top profile, where the transport variables were calculated wrongly. As a result the two last holes were positioned wrongly.

Even though we fixed many errors throughout the hole testing table 8.2 shows a, +-1mm error on the transport inn holes on the x position, +-2mm on transport out and the y position has a +-1mm error. Some of this error can be contributed to the backlash cause by the drill choke explained in section 6.2.3.

Another error was discovered during a thorough test result review. This discovery was done reviewing the full production results shown in figure 8.2 below in section 8.1.2. The holes in profile 2 and 3 have an error in the x position of the hole, because we forgot to take the cut shaving into consideration when adding transport in the CNCGenerator class for hole generation. To solve this we had to add the cut shaving calculation from section 7.2.2 to the holes x position in.

8.1.2 Cut station

The initial cut station tests was done in the same manner as the hole station, were we cut profiles without adding holes. These initial tests identified a lot of errors, the data from the first 5 tests are almost identical with test one shown in table 8.3 below.

Table 8.3: Cut station test one

Test 1:				
Profile length	**Length (mm)**	**Actual length (mm)**	**Error (mm)**	**Cut type**
1	150	150	0	AngleCut +-45 deg
2	147	150	3	RotationCut +-45 deg
3	197	200	3	RotationCut 90 deg
4	520	0	520	RotationCut 90 deg with TrimCut and BendCut
		BendCut failed so the profile broke		
Trim/BendCut	**Position (mm)**	**Actual position (mm)**	**Error (mm)**	
Trim 1	Failed	Not accurate	Failed	
Trim 2	Failed	Not accurate	Failed	
Bend 1	Failed	Did not cut deep enough for bend	Failed	
Bend 2	Failed	Did not cut deep enough for bend	Failed	

The errors and areas for improvement which we identified and fixed through the initial testing where:

- Inaccurate TCP's:
  The TCP's from figure 7.26 had to be re-calibrated.
- Inaccurate Base coordinate system:
  The base system from figure 7.26 had to be re-calibrated
- Error CNCGenerator:
  The testing identified several small bugs which were fixed.
- TrimCut and BendCut:
  TrimCut and BendCut both have errors which we know the cause off, and are explained further down

After implementing the improvements from the initial testing stage, we started the full production testing showed in figure 8.2. These three profiles are all produced from a stock profile which is 1100mm long, and are produced in the numeric order from the figure. The profile lengths and the TrimCut's position have a +-1mm margin, except for profile 3 with the BendCut which have always have a -2 to - 3mm margin.

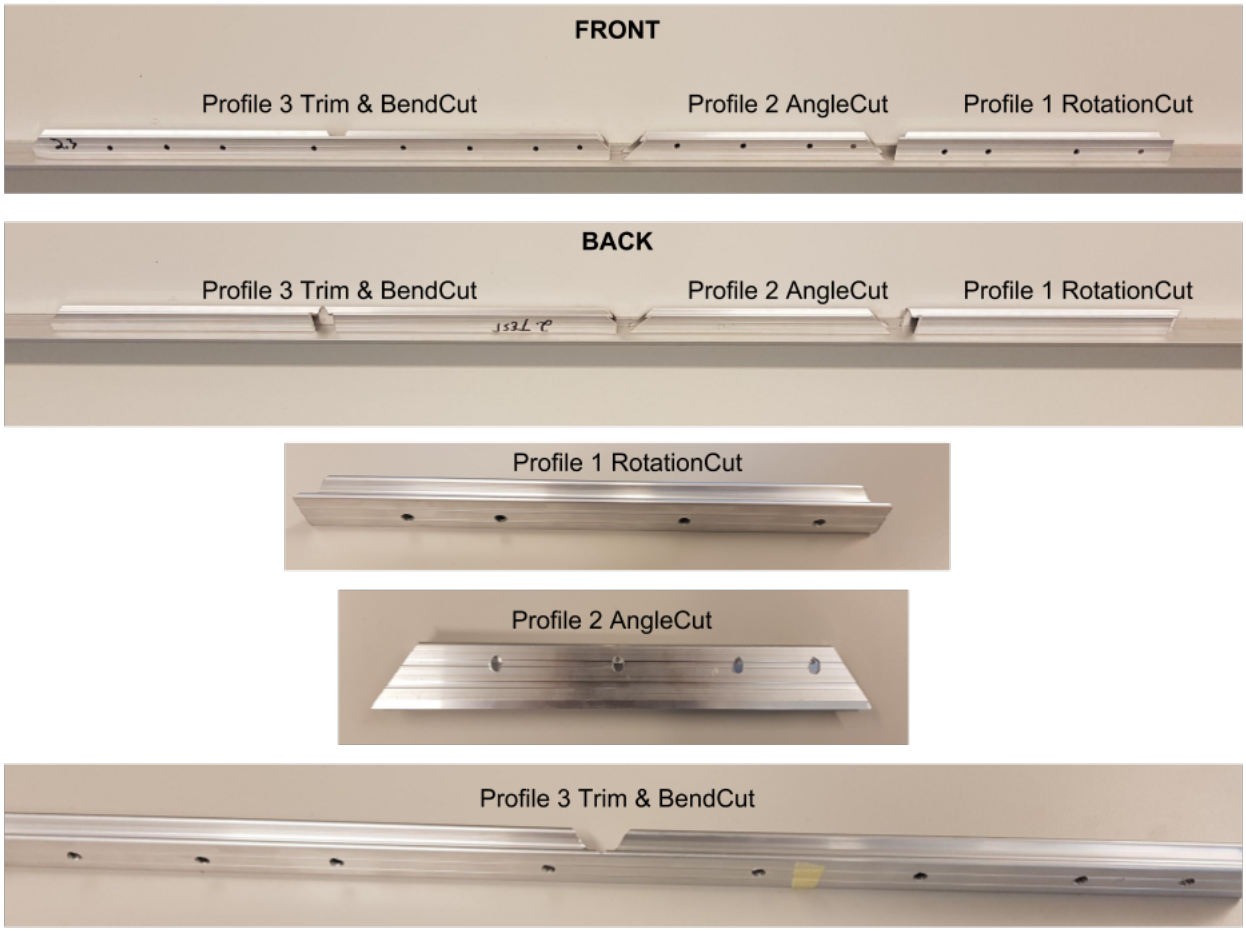


Figure 8.2: Completed production

The table 8.4 shows the statistics from eight profile 3 productions, and as the statistics show, the length is on average 3mm to short and the BendCut misses its position with 3mm. While we are not sure what causes this error, we do know that it is not related to the CNCGenerator in the server software or PLC program or KUKA. However the error is almost a constant 3mm for both the length and cut except for the abnormality in row 6.

Our leading theory is that the output guideline stepper motor is poorly calibrated. Because it is the output gripper which does the final transportation before the BendCut, and final cut of profile three. This theory also explains why the error is larger for hole 7 and 8 in table 8.1 and 8.2 which is also transported by the output gripper. We tried several other approaches which failed, before developing this theory but there were sadly no time to do production testing to verify this theory.

Table 8.4: Profile 3

Profile 3	Length (mm)	Actual length (mm)	Error	BendCut position (mm)	Actual BendCut position (mm)	Error
1	520	517	3	250	247	3
2	520	517	3	250	247	3
3	520	518	2	250	248	2
4	520	517	3	250	247	3
5	520	516.5	3,5	250	247.5	2,5
6	520	515.5	4,5	250	245	5
7	520	517	3	250	247	3
8	520	517	3	250	247	3

BendCut

In section 7.4.4 it is explained how challenging it is to perform a BendCut. This proved to be true during the test stage. The best BendCut produced is shown in figure 8.3 below. When showed to RUFO, they said that it was close but would not be used to produce a flight case, because the crack at the angle is to big.

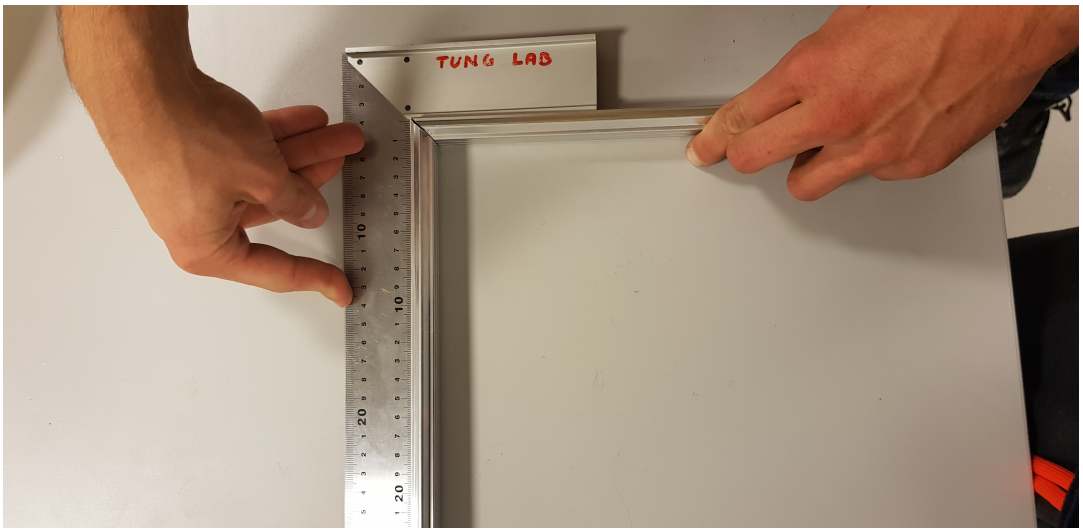


Figure 8.3: BendCut bent into 90 degrees

To perform a perfect BendCut the entry angle for cut one and two from figure 7.25 and 7.26 in section 7.4.4 has to be 45 degrees, and cut 1mm into the 2mm back wall of the profile. Figure 8.4 below shows the huge variations on both the entry angle and back wall penetration.

Figure 8.4: BendCut results

To solve this problem several solutions was implemented and test, while some solutions like re-calibrating the robot and reinforcing the robot-production cell structural integrity gave better result, the inconsistency was still to large. After multiple tests two inconsistency contributors were identified. The first one is the hole station, where figure 8.5 below shows the shaving residue left by the drill. This residue impacts the penetration of the back wall by +-1mm by ending between the red guide unit and the profile, which is enough to explain the inconsistency. The second contributor is vibrations caused by the KUKA robot, which spreads throughout the entire production cell. This contributor was not measurable since we could not reduce the vibrations, but the vibrations were visible while observing the production.

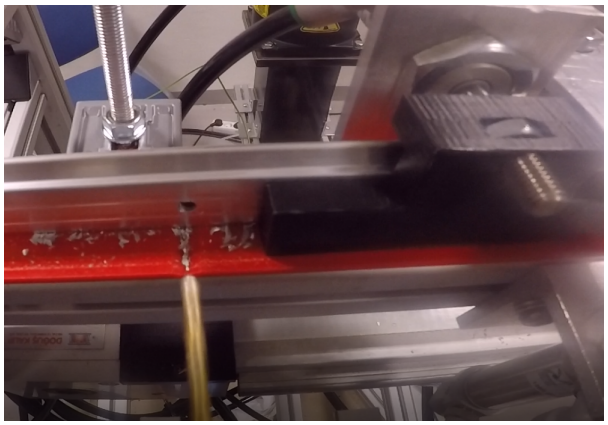


Figure 8.5: Drill shavings

TrimCut

During the production testing we discovered that the first and second TrimCut differed in height even though they had the same specifications. The first TrimCut is the trim cut at the start of the profile, and second TrimCut is at the end of the same profile. Figure 8.7 and 8.6 shows these cuts.

Figure 8.6: First TrimCut

Figure 8.7: Second TrimCut

The table 8.5 below shows the statistics for 8 profile 3's from figure 8.2 and their TrimCuts. The first cut has an height error of +-1mm, and this should also be the case for the second cut. But as the statistics show the second cut error is +-2mm, which makes the difference shown in table 8.6.

Table 8.5: Profile 3 TrimCut

Profile 3	First TrimCut height (mm)	Actual first TrimCut height (mm)	Error	Second TrimCut height (mm)	Actual second TrimCut height (mm)	Error
1	10	9	1	10	12	2
2	10	9.5	0.5	10	12	2
3	10	10	0	10	12	2
4	15	15	0	15	17.3	2.3
5	15	15.5	0.5	15	17.3	2.3
6	17	17	0	17	19	2
7	15	14.3	0.7	15	17	2
8	15	15	0	15	17	2

Table 8.6: Difference first and second TrimCut

Profile 3	Actual first TrimCut height (mm)	Actual second TrimCut height (mm)	Difference (mm)
1	9	12	3
2	9.5	12	2.5
3	10	12	2
4	15	17.3	2.3
5	15.5	17.3	1.8
6	17	19	2
7	14.3	17	2.7
8	15	17	2

The reason for the TrimCut difference is illustrated in figure 8.8, and it is due to inaccuracies in the prototypes construction. There is a small height difference between each side of the cut station, where the right side is slightly higher than the left side. This explains why the average error for cut one and two in table 8.5 above is different.

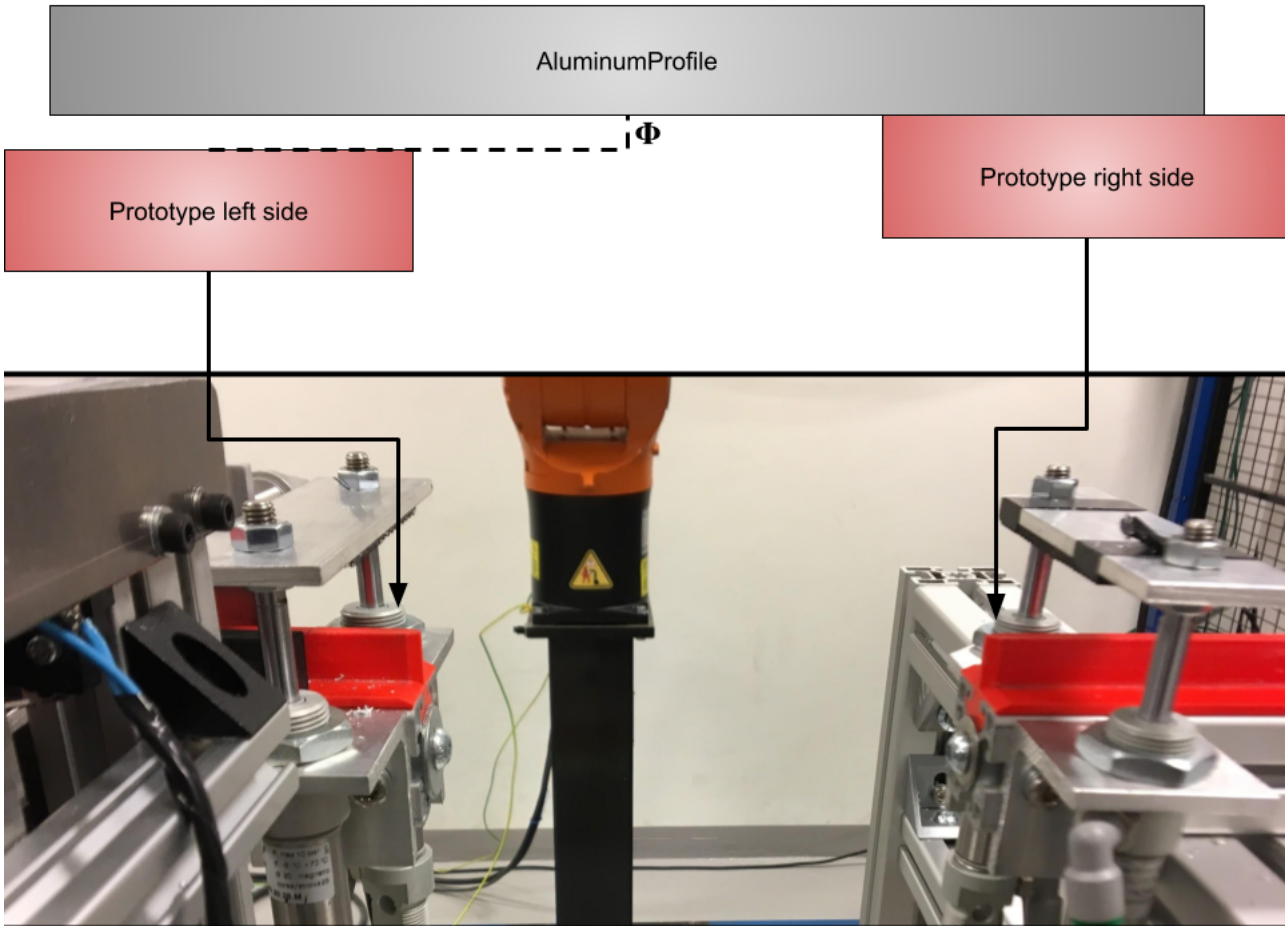


Figure 8.8: First and Second TrimCut difference explained

8.1.3 Accuracy conclusion

Even after identifying problem areas and correcting them, the prototype is still too inaccurate when comparing to the project specifications in section 1.2.1 which require a $\frac{1}{10}mm$ margin. The prototypes margins however is:

- Hole station y-axis margin = $+-1mm$
- Hole station x-axis margin = $+-1mm$ using transport inn,
  and $+-2mm$ using transport out.
- Profile length margin = $+-1mm$ using transport inn,
  and $-2 \quad to \quad -3mm$ using transport out.
- BendCut back wall penetration margin = $+-1mm$.
- TrimCut is positioned correct in the x-axis, but the y-axis or height has a margin of = $+-1mm$

These margins are larger than RUFO's $\frac{1}{10}mm$ demand. While we do not know exactly what causes these larger margins, We do know that they are structural, or stepper calibration related. The reasoning behind this claim is that we have manually calculated the transport from the generated CNC-Code and made sure that all order information is relayed correctly to the PLC, which means that the software has a $\frac{1}{10}mm$ margin. The CNC-Code calculations in table 8.7 below are done after the CNC-Code generated for the production in figure 8.2. The full calculations with explanations can be found in appendix D.

Table 8.7: CNC-Code calculations

Ordered profile	**Millimeter**	**CNC-Code length**	**Millimeter including CutShaving**	**Milimeter excluding CutShaving**	**CutShaving**	
Profile 1	240	Profile 1	244	240,04	3,96	
Profile 2	240	Profile 2	240	240,00	0	
Profile 3	520	Profile 3	522,8	520,00	2,8	
Ordered profile	**First hole**	**CNC-Code position**	**Actual position**	**Calculation**		
Profile 1	20	245,3	20	CNCpos - Drilloffset -Profilewaste	Cut ofsett	390
Profile 2	30	499,3	30	CNCpos - Drilloffset -Profilewaste -Profile1	Drill offset	146
Profile 3	30	743,3	30	CNCpos - Drilloffset -Profilewaste -Profile1 -profile2	Profile waste	79,3

There are several contributors to these larger margins, the structural integrity, vibrations and bad design solutions. The hole station margins is already explained by the backlash in the drill chocs, and the BendCut back wall penetration is greatly affected by the hole station residue. The point is that reaching the $\frac{1}{10}mm$ margin is possible, by having a professional machine designer and constructor to build the machine. Having a professional built machine facilitates for easier stepper and robot calibration as well, which leads to better margins.

8.2 Quality

The production quality that the prototype delivers does not live up to RUFO's standards. Figure 8.9 and 8.10 below shows examples of cuts performed by the prototype. The cuts often leave uneven surfaces with shavings, and sometimes actually bends the profile during the cut. During a prototype presentation for RUFO we approached them with this quality issue, and was told this happens because of two reasons. The first reason is that the velocity the robot uses during the cut is to slow which generates to much heat so the aluminum warps, which can be solved using water cooling or higher cut speeds. The second reason is the lack of profile support during the cut in the cut station.

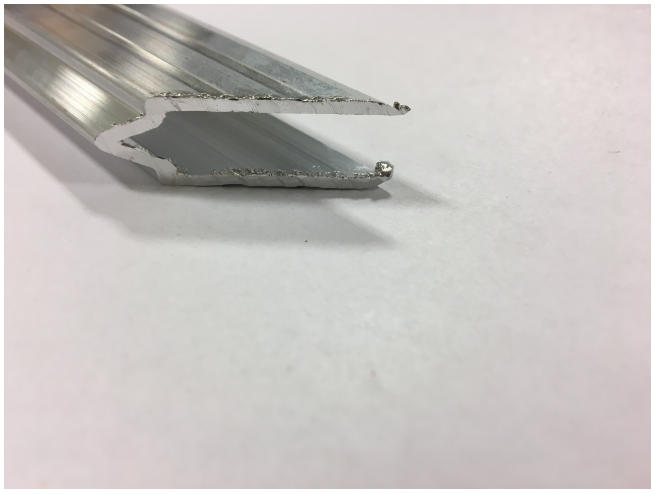


Figure 8.9: Cut shaving

Figure 8.10: Profile bent after cut

Figure 8.11 below shows the hole quality, which according to the project specifications in section 1.2.1 must be without any defects. The prototype does not deliver on these specifications. However this result was expected, because we use a drill which provides a low RPM with backlash on the choke and utilizes a drill tool instead of a milling tool.

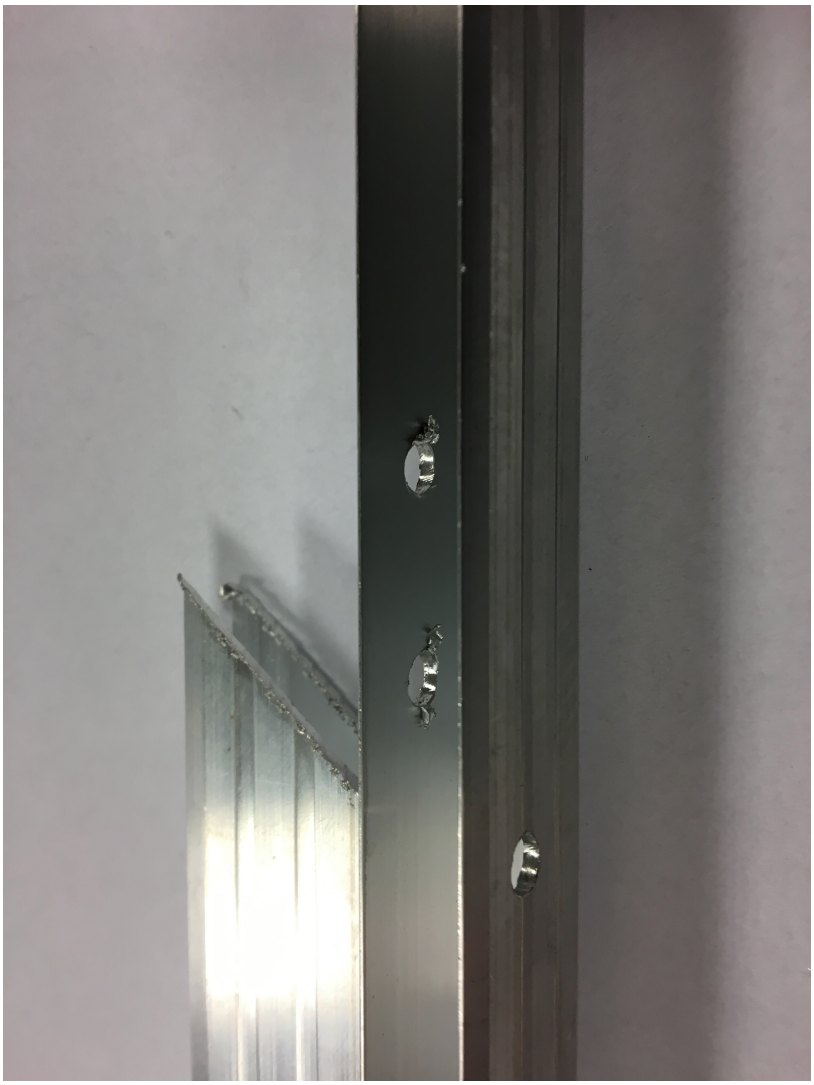


Figure 8.11: Cut and hole quality

The conclusion is somewhat the same as the accuracy chapter above. That constructing a production cell system which delivers quality products is possible by redesigning the production cell, including implementing a dedicated spindle for milling holes and utilizing a spindle with custom cooling to cut the profiles.

8.3 Lead time

By utilizing a captured video of a full production we were able to calculate the lead time and time per hole and cut operation for the prototype seen in table 8.8 and 8.9 below. The production order used to calculate these times is the one showed in figure 8.2 above, and producing three profiles from a 1100mm aluminum profiles takes 4.5 minutes. This is not a particular fast production time but it is important to take into consideration that during testing we focused on quality, accuracy and software debugging. However we have used these times as a baseline to estimate a realistic lead time for a fully developed production cell.

Table 8.8: Prototype lead time

Prototype lead times for 1100mm profile:								
One cycle on the prototype (sec):	**Hole (sec):**	**Cut type:**	**Cut (sec)**	**Transport out (sec):**	**Total with out transport in:**	**Transport in (sec):**	**Total (sec):**	**Total (min):**
270	4	RotationCut 45	14	11				
	3	RotationCut -45	18	9				
	3	RotationCut 90	10	1				
	3	RotationCut 90	10	8				
	5	AngleCut 45	12					
	3	AngleCut -45	12					
	4	TrimCut +45	22					
	4	TrimCut -45	25					
	5	BendCut	37					
	3							
	4							
	3							
	4							
	4							
	3							
	3							
Total time (sec)	**58**		**160**	**29**	**247**	**23**	**270**	**4,5**
Number of actions	**16**		**9**	**4**		**21**	**50**	
Average time	**3,6**		**17,8**	**7,25**		**1,1**	**5,4**	

Table 8.9: Prototype operations

Prototype operations times	
Cut types:	**Time (sec)**
BendCut	37
TrimCut +45	22
TrimCut -45	25
AngleCut 45	12
AngleCut -45	12
RotationCut 45	14
RotationCut -45	18

Improved operation times	
Cut types:	**Time (sec)**
Bend	25
Trim positiv	14
Trim negativ	18
AngleCut 45	8
AngleCut -45	8
RotationCut 45	10
RotationCut -45	14

Operations:	**Average time (s)**
Cut	20,0
Hole	3,6
Transport inn	1,1
Transport out	7,25

Operations:	**Average time (s)**
Cut	13,9
Hole	2,0
Transport inn	1,1
Transport out	5

The left side of table 8.9 prototype operation times are extracted from the video file. By using this information and examining the video file we reduced these times within a reasonable limit, meaning if we had the time to do optimization testing, even shorter operation times should be feasible. These new times are listed in the right side of table 8.9. By utilizing these new operation times the new prototype lead time listed below in table 8.10 is 3,4 minutes, which is over a minute improvement.

Table 8.10: Improved prototype lead time

Improved prototype lead time				
Operations	**Quantity**	**Operation time (sec)**	**Total operation time (sec)**	**Total operation time (min)**
Cut	9	13,9	124,7142857	
Hole	16	2,0	32	
Transport inn	25	1,1	27,5	
Transport out	4	5	20	
Total time:			**204,2142857**	**3,403571429**

Table 8.11: Producing using 6m aluminum

Cutting list for 6m profile		
Profile type:	**Cut type:**	**Profile length (mm):**
6123	45 - 45 AngleCut (Bendcut)	816
6123	45 - 45 AngleCut (Bendcut)	816
6123	45 - 45 AngleCut (Bendcut)	816
6123	45 - 45 AngleCut (Bendcut)	816
6123	45 - 90 AngleCut	116
6123	45 - 90 AngleCut	116
6123	45 - 90 AngleCut	116
6123	45 - 90 AngleCut	116
6123	45 - 90 AngleCut	356
6123	45 - 90 AngleCut	356
6123	45 - 90 AngleCut	356
6123	45 - 90 AngleCut	356
6123	45 - 90 AngleCut	356
6123	45 - 90 AngleCut	356
Tot.		**5864**
Waist:		**136**

RUFO flight case information	
Total length (mm)	
7006	Angle profile
4848	Guide profile
Profiles	
14	Guide profiles
Average number of cuts per profile	
2,285714286	Cuts
Number of holes	
68	Holes
Average number of holes per unit	
Average holes pr mm.	0,014
Average holes pr m.	14,026

Table 8.11 above and table 8.12 below are used to calculate an estimated lead time for production line two in the finished production cell. To do this calculation we calculated the average number of operations from the guide profile aluminum needed for a standard flight case seen in the right side of table 8.11. The left side of this table shows the 6m aluminum optimized with 14 profiles used to calculate the lead time. The lead time for such an profile is estimated to 13.4 minutes shown in table 8.12 below.

However we believe that the 13.4 minute lead time can be improved significantly, by increasing the speed of the stepper motors used to transport profiles inn and out and running operations simultaneously. Production line one for the angle profiles should operate even faster because it only requires AngleCut and RotationCut. But there is however a limit to how much you can optimize each production line, and this is the robotic arm, which has to do cut operations on both production lines. Implementing two robotic arms or dedicated saws would help improving the production cells lead time.

Table 8.12: Production line two lead time

Lead time:				
Operations	**Quantity**	**Operation time (sec)**	**Total operation time (sec)**	**Total operation time (min)**
Cut	32	13,9	443,4285714	
Hole	82	2,0	164,5016502	
Transport inn	114,251	1,1	125,6759076	
Transport out	14	5	70	
Total time:			**803,6061292**	**13,39343549**

8.4 HSE

The human safety environment implementation in the prototype is not representative for the finished production cell, and there is several reasons behind this statement.

- Emergency stop:
  The prototype only utilizes the emergency stop on the robot controller.
- TwinSAFE: The TwinSAFE safety features explained in section 2.4.2 is not implemented in the prototype because the control system does not have safety I/O modules, and only one emergency stop.
- Robot cage: The safety cage's grating is to large. During testing aluminum shavings flew through the cage, forcing us to use goggles.

When constructing the complete production cell enough emergency stop buttons will be added to satisfy the Norwegian machine regulations, which utilizes the TwinSAFE safety features. The processing station which cuts and generates holes in the profiles should be enclosed to eliminate the danger of flying aluminum shavings.

8.5 Price estimate

In section 6.4 the thesis presents an approximate budget for the prototype which has a cost of 360800 NOK. By using this information we have calculated a price estimate shown in table 8.13 for a finished production cell constructed using the same materials and techniques described in section 6 Prototype development. This estimate does not consider the software development, and construction time.

Table 8.13: Price estimate for the production cell

Component	**Quantity**	**Price per pcs**	**Total price in NOK**
Robot with controller	1	300000	300000
PLC with IO modules	1	45000	45000
stepper motor with controller	7	2000	14000
Saw	1	6000	6000
Mill machine	3	5000	15000
Electrical components and wires	1	20000	20000
Metal structure	1	40000	40000
		Total:	440000

It is important to realize that this is a rough price estimate, where the total price may be 440 000 +- 100 000 NOK. But the interesting part is that it is the Robot which constitutes over 50% of the price estimate, so the expansion from one production line as the prototype, to two lines is a small investment. RUFO can use this estimate to calculate the number of full-time equivalent or FTEs needed to turn the production cell investment into profit.

Chapter 9

Conclusion

Although the thesis needed a redefinition, which shifted the goal from just developing the control system to enlarging the thesis, by adding the additional workload of designing and constructing a working prototype. The work throughout the thesis has provided us with the information required to answer the thesis problem:

- *Is it possible to develop an automated control system for an aluminum profile production cell, by following the project specifications provided by RUFO AS?*

By working systematically the thesis work was split into three parts, prototype development 6, software development 7 and results 8. Such composition of activities have provided us with the ability to resolve the thesis problem.

The prototype testing revealed that hardware and construction related problems made the system inadequate to deliver on of RUFO's project specifications 1.2.1 considering accuracy and quality. Where we argument that designing the production cell in cooperation with a professional will solve the issues, and the finished product becomes adequate. However the assertion that the systems problem is not software related is only based on the prototype testing, and software bugs might occur when integrating a production cell with two production lines.

After evaluating the thesis project we consider the work and results satisfactory. We conclude that it is possible to develop an automated system and production cell which correspond to RUFO's demands. Based on the research and development results it is apparent that the server software fulfill the requirements completely. Not only do we think it is possible, but based on the estimated lead times and price we firmly believe that this is an investment which will increase their production capacity and competitiveness.

9.1 Recommendations for Further Work

As recommendations for further work we propose continuing the test stage of the thesis, to clearly identify how to proceed to deliver profiles adequate to RUFO's demands.

- Implementing a hole station using a dedicated spindle for milling.
- Developing and testing different fixtures for the KR-6 robot with spindles and saws utilizing different cooling solutions.
- Reinforcing the structural integrity to reduce vibrations

Bibliography

Adam-Hall (2018a). *Adam Hall Hardware 6105*. Adam Hall. URL: https://www.adamhall.com/shop/je-en/flight-case/extrusions-profiles/894/6105?c=14602 (visited on 03/26/2018).

— (2018b). *Adam Hall Hardware 6123*. Adam Hall. URL: https://www.adamhall.com/shop/je-en/flight-case/extrusions-profiles/911/6123-f?c=14602 (visited on 03/26/2018).

AET-Labs (2018). *AGILUS – KR6 Robots*. AET Labs. URL: http://www.aetlabs.com/product/kuka-agilus-kr6-r700/ (visited on 05/25/2018).

Autodesk (2017). *Getting Started with G-code*. Autodesk. URL: https://www.autodesk.com/industry/manufacturing/resources/manufacturing-engineer/g-code (visited on 02/23/2017).

Beckhoff (2012). *Extended automation*. Beckhoff. URL: https://download.beckhoff.com/download/document/automation/twinsafe/kl1904en.pdf.

— (2017a). *EL252x*. 3.6. Beckhoff. URL: https://download.beckhoff.com/download/document/io/ethercat-terminals/el252xen.pdf.

— (2017b). *TwinSAFE input terminal with 4 fail-safe inputs*. Beckhoff. URL: https://download.beckhoff.com/download/document/automation/twinsafe/kl1904en.pdf.

— (2018a). *ADS-Communication*. Beckhoff. URL: https://infosys.beckhoff.com/english.php?content=../content/1033/cx8090_hw/1610551947.html&id= (visited on 03/07/2018).

— (2018b). *Quick start guide*. 1.1. Beckhoff. URL: https://download.beckhoff.com/download/document/automation/twincat3/TF5000_TC3_NC_PTP_Quick_Starting_Guide_EN.pdf.

— (2018c). *TwinCAT 3 extended Automation*. Beckhoff. URL: https://www.beckhoff.com/english.asp?twincat/twincat-3.htm (visited on 03/22/2018).

Burris, Matthew (2018). "Stepper Motors vs. Servo Motors - Selecting a Motor". In: *Motion Solution*. URL: https://www.lifewire.com/stepper-motor-vs-servo-motors-selecting-a-motor-818841 (visited on 05/12/2018).

Ekornseater, Lars Annfinn (2016). “Samarbeid for fremtidens industri”. In: *TU*. URL: https://www.tu.no/artikler/samarbeid-for-fremtidens-industri/277106 (visited on 05/10/2018).

EtherCAT (2018a). *EtherCAT - the Ethernet Fieldbus*. EtherCAT. URL: https://www.ethercat.org/en/technology.html.

— (2018b). *Safety over EtherCAT*. Beckhoff. URL: https://www.ethercat.org/en/safety.html (visited on 01/26/2018).

Greenfield, David (2018). “Choosing Between Cobots and Industrial Robots”. In: *Automation-World*. URL: https://www.automationworld.com/choosing-between-cobots-and-industrial-robots (visited on 05/12/2018).

KUKA (2003). *KR C Expert Programming*. KUKA. URL: http://control.put.poznan.pl/old/sites/default/files/KUKA_UserProgramming.pdf.

— (2018a). *KR 6 R900 sixx*. KUKA. URL: https://www.kuka.com/-/media/kuka-downloads/imported/6b77eecacfe542d3b736af377562ecaa/0000205456_en.pdf.

— (2018b). *KUKA WorkVisual*. KUKA. URL: https://www.kuka.com/en-at/products/robotics-systems/software/system-software/kuka_systemsoftware/kuka-work-visual.

Lackey, Bill (2018). “Servo Motor vs Stepper Motor: Which is right for your application”. In: *Motion Solution*. URL: https://www.motionsolutions.com/servo-motor-vs-stepper-motor-right-application/ (visited on 05/12/2018).

Lovdata (2009). *Forskrift om maskiner*. Lovdata. URL: https://lovdata.no/dokument/SF/forskrift/2009-05-20-544?q=Forskrift%5C%20om%5C%20maskiner (visited on 02/03/2017).

Makeitfrom (2018). *6060-T66 Aluminum*. Makeitfrom. URL: https://www.makeitfrom.com/material-properties/6060-T66-Aluminum (visited on 04/26/2018).

Microsoft (2018a). *C# Guide*. Microsoft. URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (visited on 04/16/2018).

— (2018b). *Visual Studio*. Microsoft. URL: https://www.visualstudio.com/vs/features/ide/ (visited on 04/26/2018).

Negnevitsky, Michael (2011a). *Artificial intelligence: A guide to intelligent systems*. 3rd ed. Harlow, Edinburgh: Pearson, p. 221.

— (2011b). *Artificial intelligence: A guide to intelligent systems*. 3rd ed. Harlow, Edinburgh: Pearson, pp. 222–229.

PTD (2017). *Punching force calculation*. Project tool and die. URL: https://www.projecttoolanddie.com/site/assets/files/1053/punching-force-calculation.pdf (visited on 06/17/2017).

Sandvik (2016). *Milling Formulas and Definitions*. Sandvik. URL: https://www.sandvik.coromant.com/en-us/knowledge/machining-formulas-definitions/pages/milling.aspx (visited on 04/11/2017).

— (2018). *Milling Formulas and Definitions*. Sandviken Coromant. URL: https://www.sandvik.coromant.com/en-us/knowledge/machining-formulas-definitions/pages/milling.aspx (visited on 04/14/2018).

Seehusen, Joachim (2017). "Her skal industrien utvikle roboter som kan hente produksjon hjem til Norge". In: *TU*. URL: https://www.tu.no/artikler/her-skal-industrien-utvikle-roboter-som-kan-hente-produksjon-hjem-til-norge/397987 (visited on 05/10/2018).

Shah, Bhavin C., Dr. Devendra Negal, and Dr. Swati Sharma (2016). *Coordinat system for industrial robots*. Tech. rep. International Conference on Emerging Technologies in Engineering, Biomedical, Medical and. URL: http://www.ijtre.com/images/scripts/16141.pdf (visited on 04/17/2018).

Stadtler, Hartmut (2000). "A one-dimensional cutting stock problem in the aluminium industry and its solution". In: *ScienceDirect*. URL: https://ac.els-cdn.com/037722179090356G/1-s2.0-037722179090356G-main.pdf?_tid=01bcba19-690d-49a0-ac0b-a5a2509e1732&acdnat=1526122689_6133fd6909d9d79b9967be7f548cb7cb (visited on 02/06/2018).

Standardization, International Organization for (2012). *Robots and robotic device, ISO 8373:2012*. International Organization for Standardization. URL: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en:term:2.10.

WillsonTool (2013). *A closer look at Punching force*. Wilson tool. URL: https://www.wilsontool.com/getattachment/778e090e-786a-47bc-b0d3-bffecc60b25f/Tooling-Around-the-World-e-Newsletter-February-2013 (visited on 03/29/2013).

Appendix A

PLC-KUKA Communication

The communication with the KUKA robot is done through EtherCAT. When coupling the communications setup it is important to know that both the Beckhoff PLC and KUKA robot wants to be the master. To solve this a bridge module called EL6692 is used, which tricks the PLC and KUKA so they both are masters. In this tutorial the EL6692 module is placed inside the KUKA KR-C control cabinet, but it can also be placed as an module for the PLC. The coupling schematics is seen in figure A.1 below.

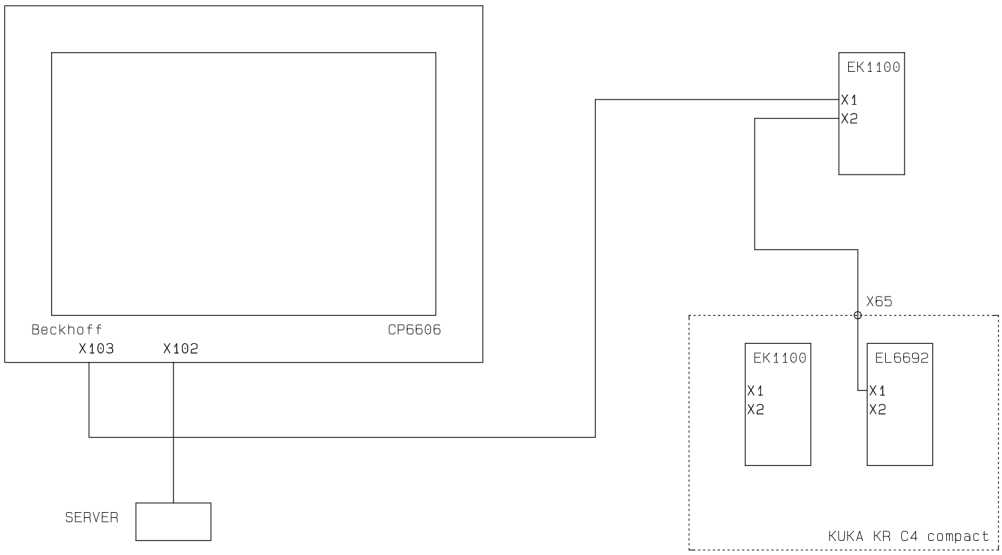


Figure A.1: Connection drawing for the Communication

IO configuration PLC side:

1. Open TwinCAT in Visual Studio. The tutorial is made in Visual studio 2015 however it works the same way in Visual studio 2013. Create a new project of type TwinCAT XAE Project (XML format).

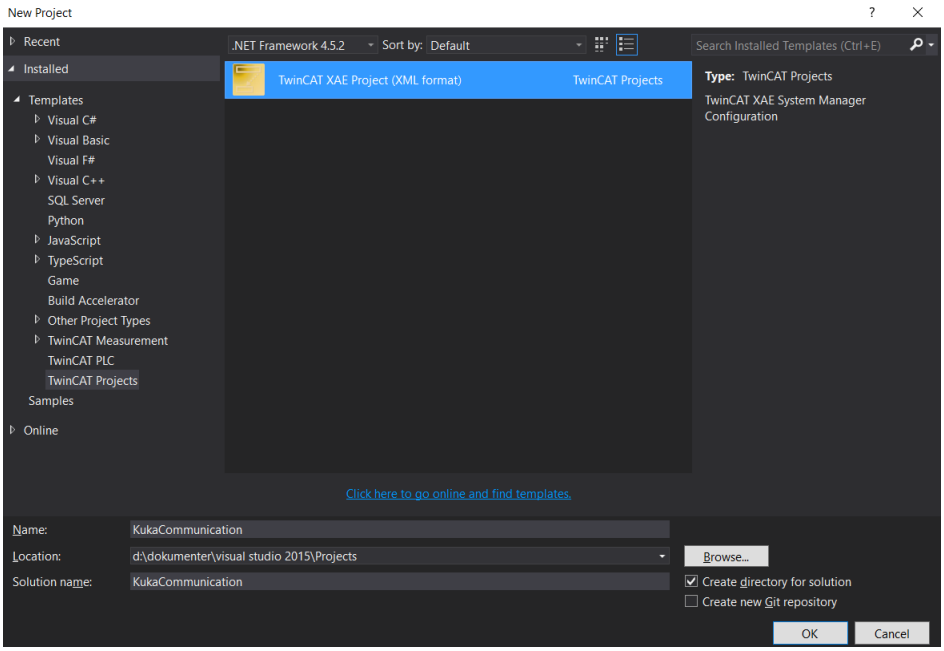


Figure A.2: Create new project

2. Click Choose Target system

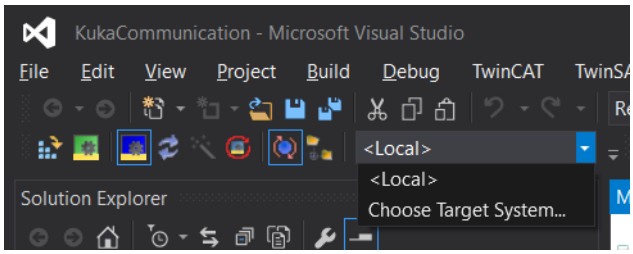


Figure A.3: Connect to the PLC

3. Connect your PC to the network the PLC is on. Search (ethernet)

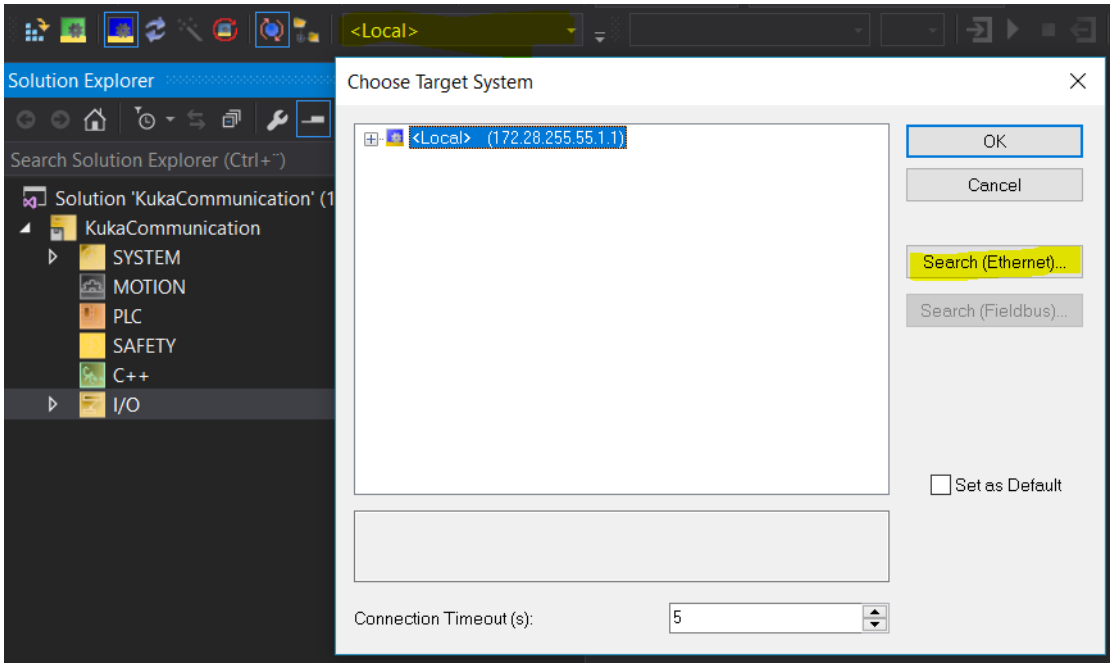


Figure A.4: Create new project

4. Broadcast search, and the PLC and PC will then be listed. Select the PLC and press AddRoute.

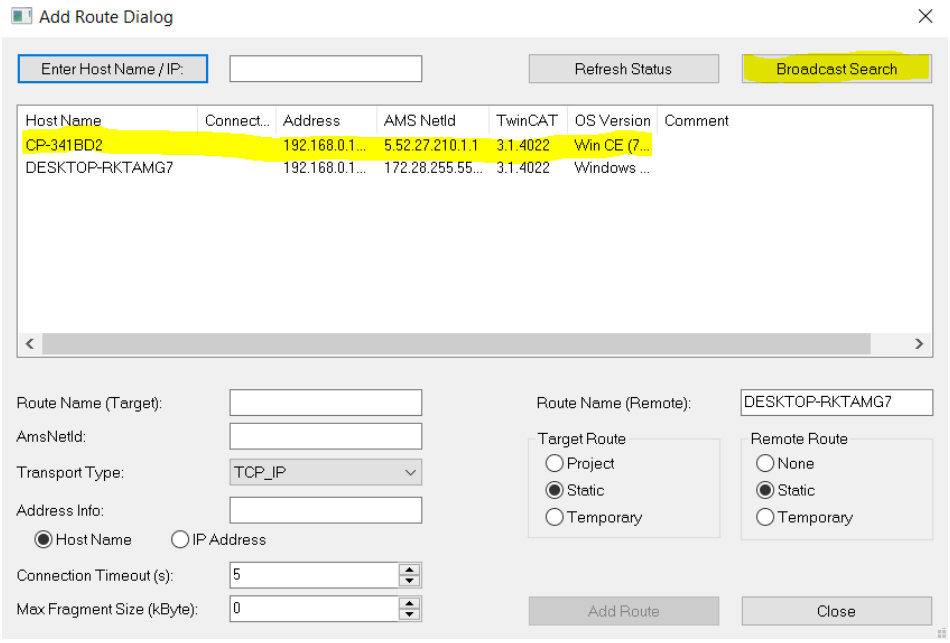


Figure A.5: Create new project

5. Press OK. The default password is blank.

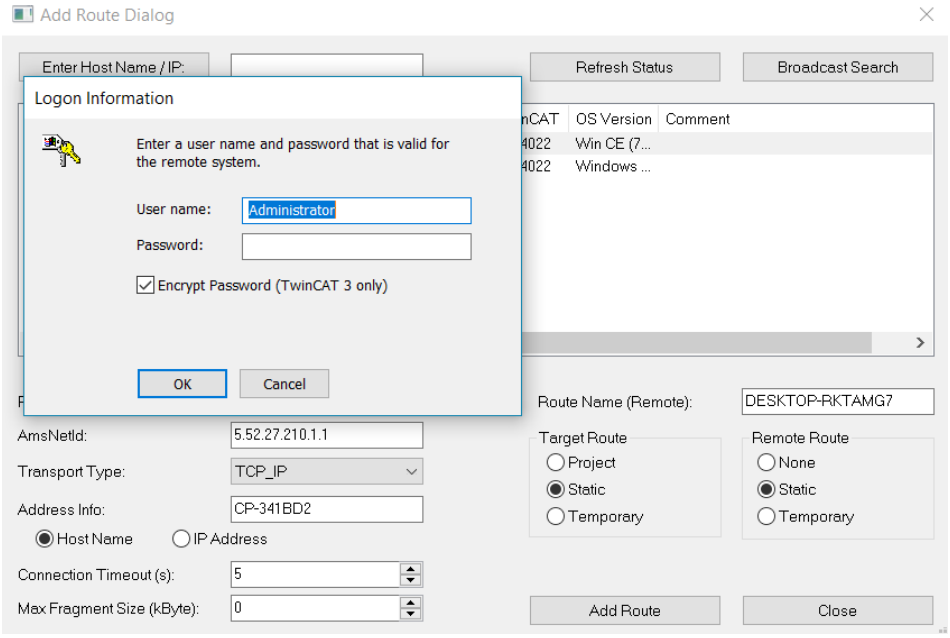


Figure A.6: Create new project

6. The PLC will show an X in the route dialog, meaning it is connected.

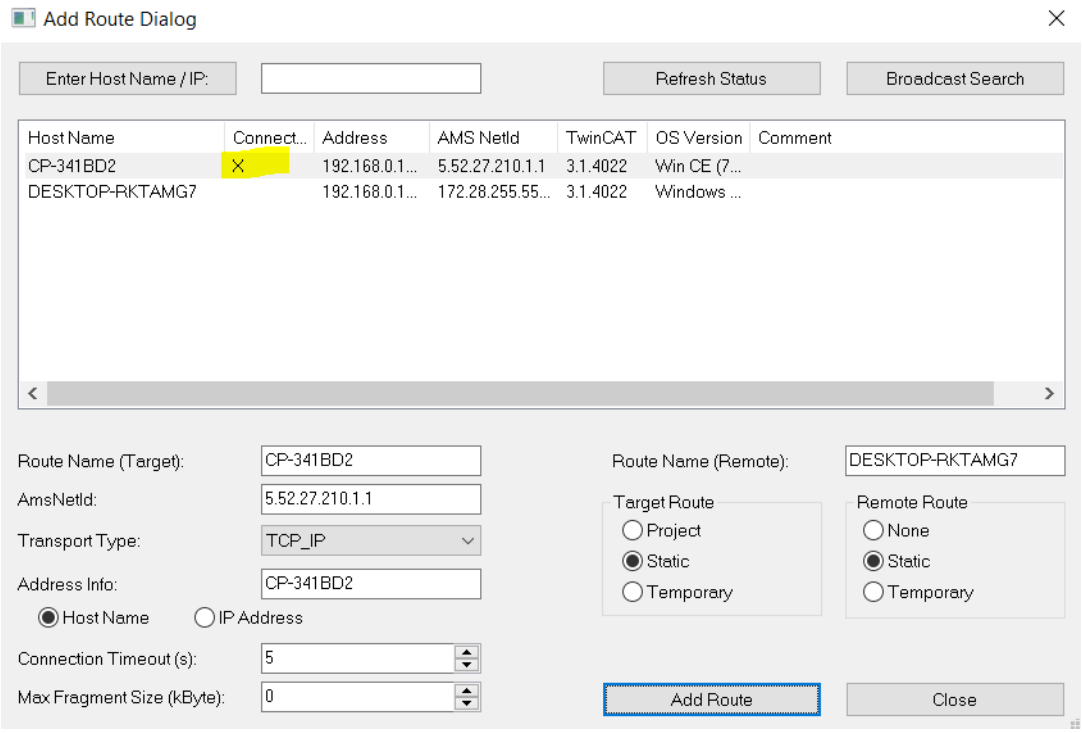


Figure A.7: Create new project

7. Be sure the PLC is selected.

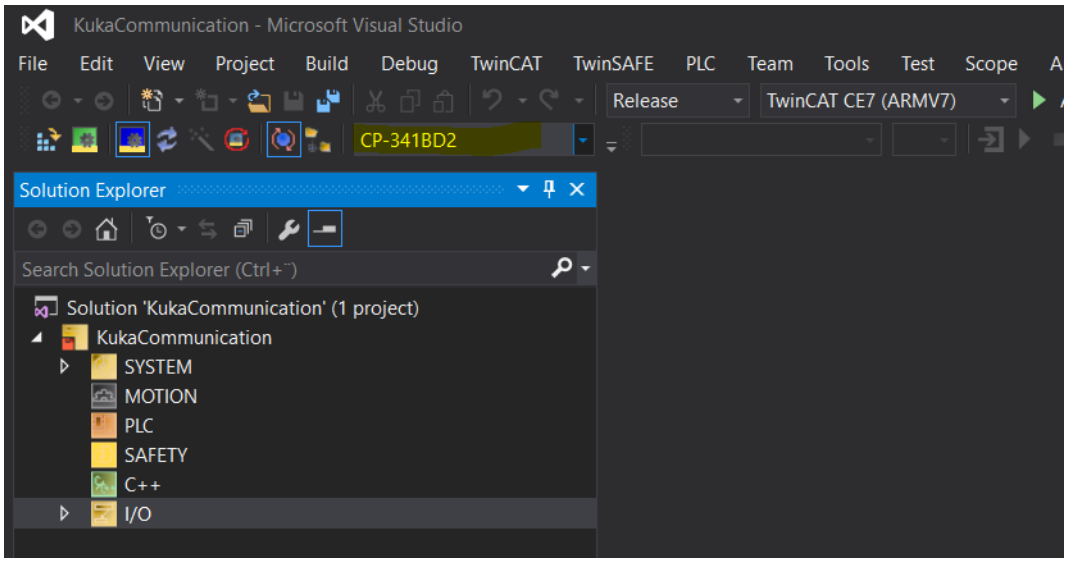


Figure A.8: Create new project

8. Scan the device for IO Modules

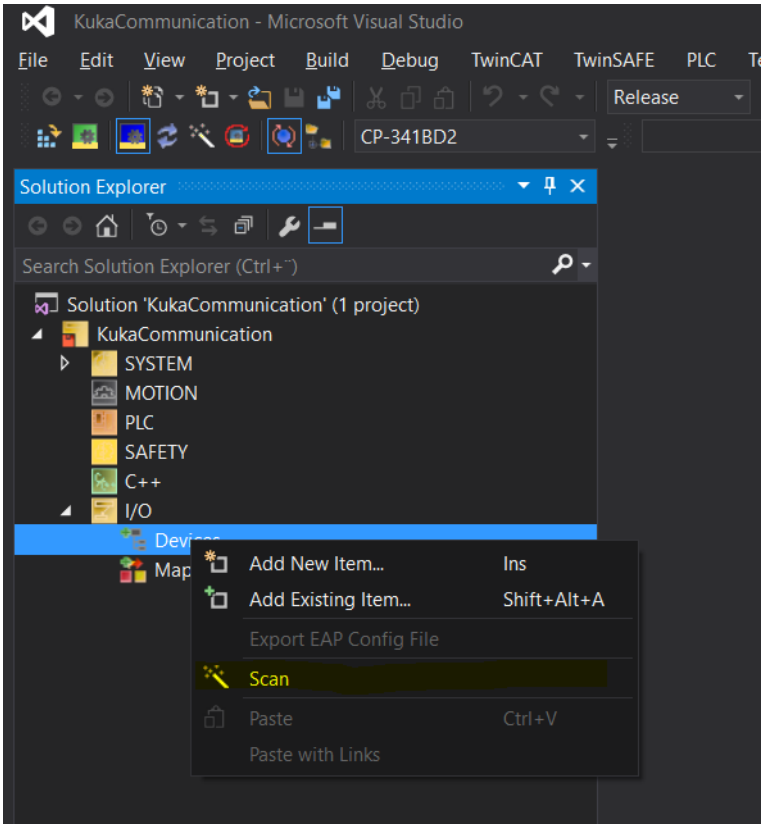


Figure A.9

9. Press OK, the correct EtherCAT device should be pre chosen.

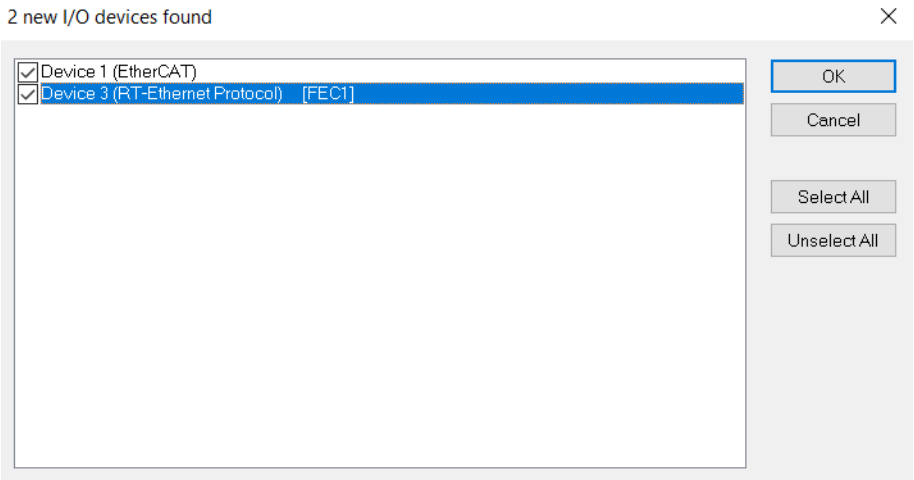


Figure A.10

10. Scan for boxes.

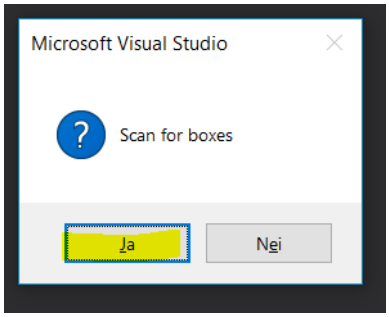


Figure A.11

11. If the EL6692 is placed in the KUKA cabinet, it will show as a separate entitiy i Visual studio

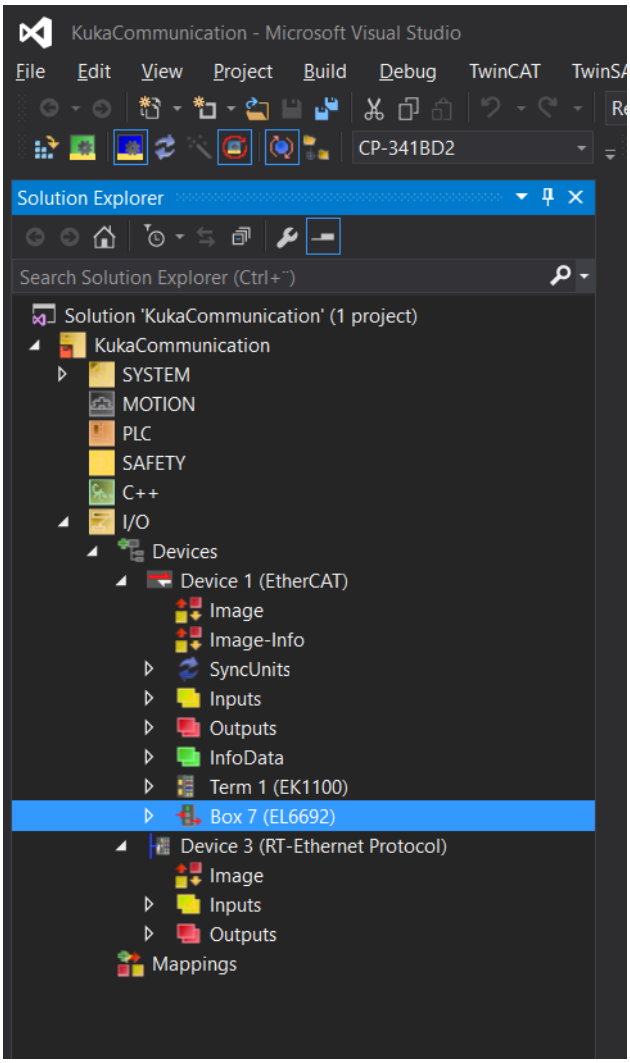


Figure A.12

12. Open the module and add new item in order to add the variables for input and output. The input variable on the PLC is the identical output variable on the KUKA robot and vice versa.

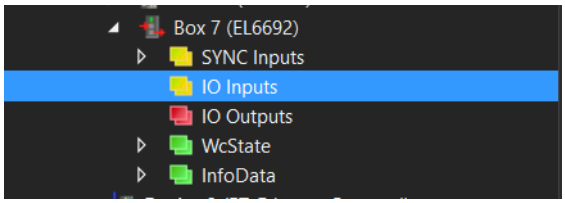


Figure A.13

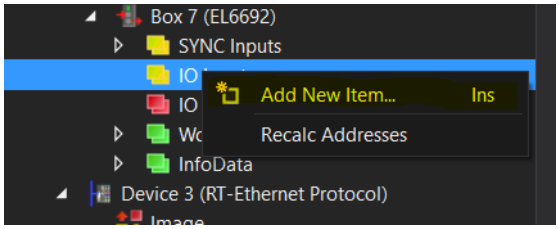


Figure A.14

13. Choose the data type to be sent. The size is in bytes. The number of bytes created here must exactly mach the number of output bytes on the robot side. Choose a name for the variable. A bool on the PLC is 1 byte, but the KUKA handles bools as 1 bit, therefore boolean values is easiest to send as USINT and assign 0= false and 255=true. In this turotrial we send a DINT with bytesize 4, which is used as a signed integer.

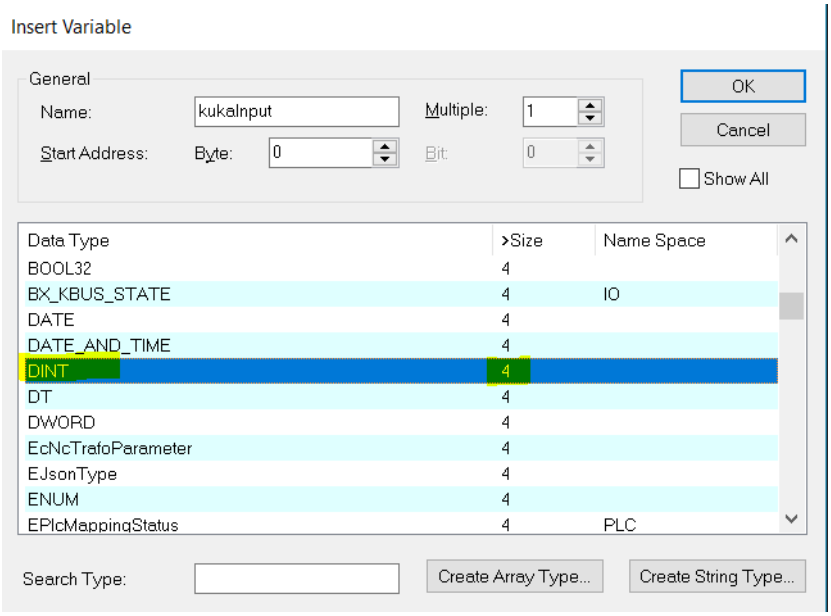


Figure A.15

14. The Input variable is displayed under the IO Inputs under the EL6692 module.

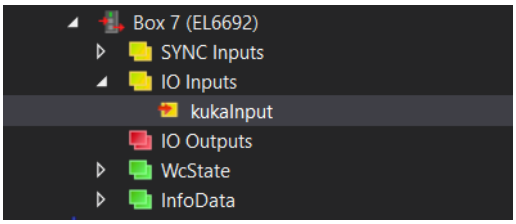


Figure A.16

15. The same operation performed for an output signal to the robot.

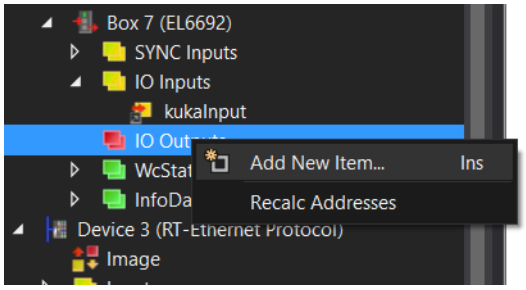


Figure A.17

16. Choose the data type.

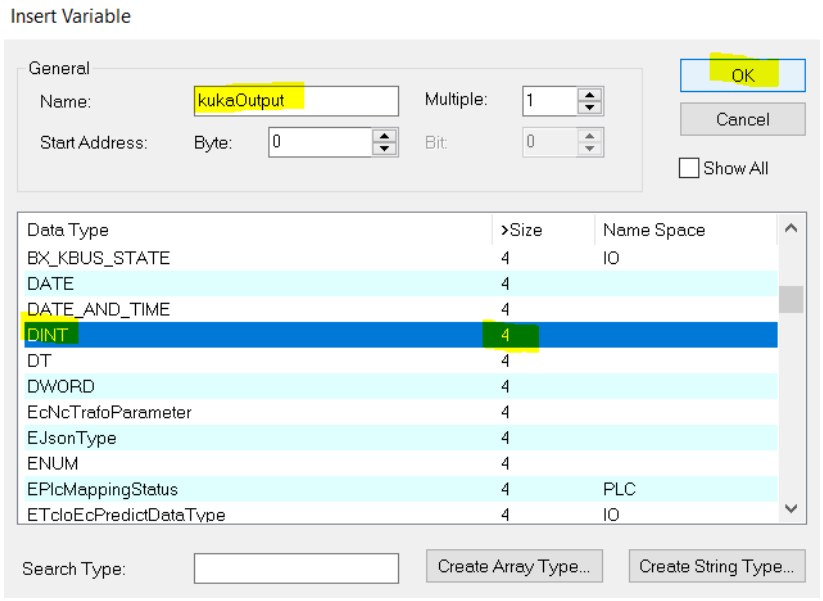


Figure A.18

17. To use the created variables in a program on the PLC, right click and choose add new item on the PLC.

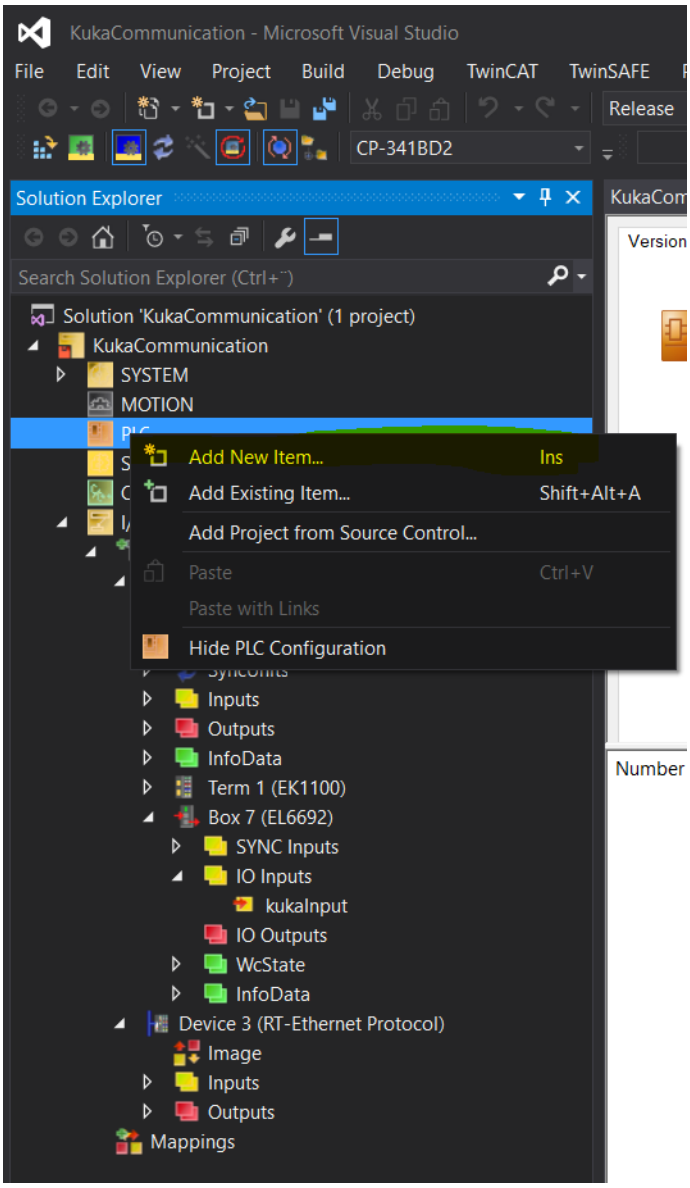

KukaCommunication - Microsoft Visual Studio
File
Edit
View
Project
Build
Debug
TwinCAT
TwinSAFE
Release
CP-341BD2
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'KukaCommunication' (1 project)
KukaCommunication
SYSTEM
MOTION
Add New Item... Ins
Add Existing Item... Shift+Alt+A
Add Project from Source Control...
Paste Ctrl+V
Paste with Links
Hide PLC Configuration
Inputs
Outputs
InfoData
Term 1 (EK1100)
Box 7 (EL6692)
SYNC Inputs
IO Inputs
kukaInput
IO Outputs
WcState
InfoData
Device 3 (RT-Ethernet Protocol)
Image
Inputs
Outputs
Mappings
Version
Number

Figure A.19

18. Choose standard PLC project and click add.

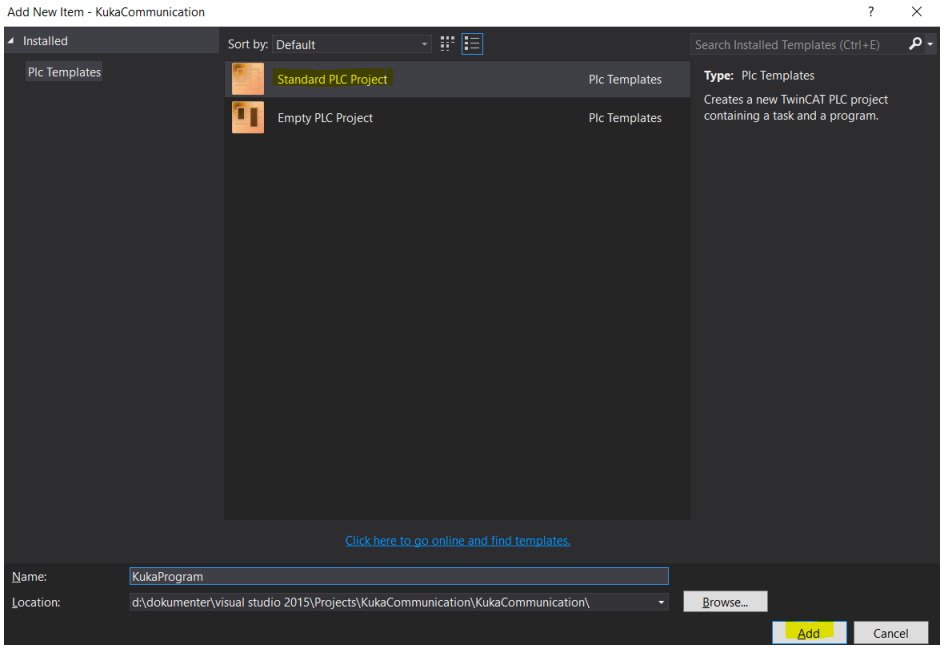


Figure A.20

19. When the program is generated. Open the POUs folder and double click on MAIN to open the program.

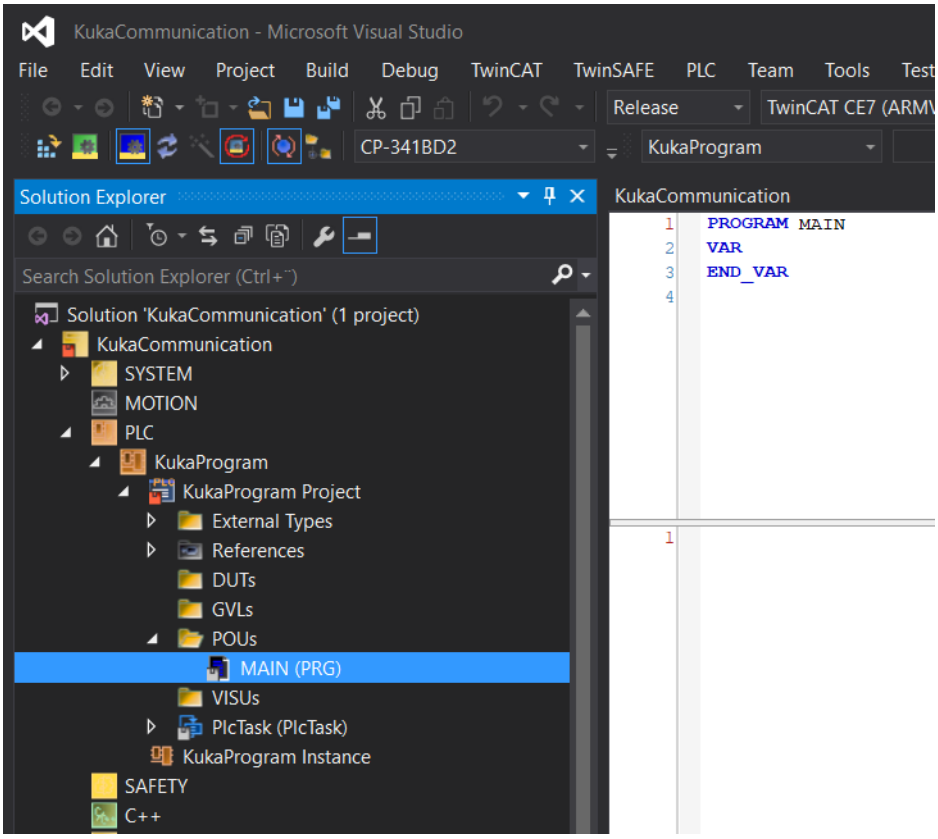


Figure A.21

20. Create the variables in the figure below, these are used to utilize the EL6692 variables created earlier. These variables needs to mach each other.

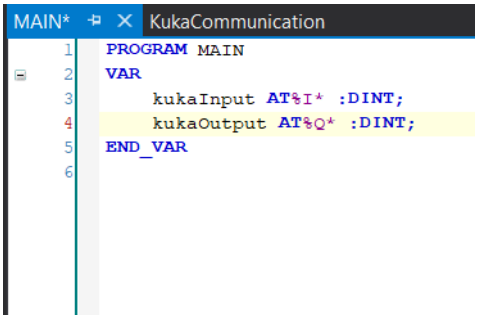


Figure A.22

21. The solution needs be built before we can link the variables together.

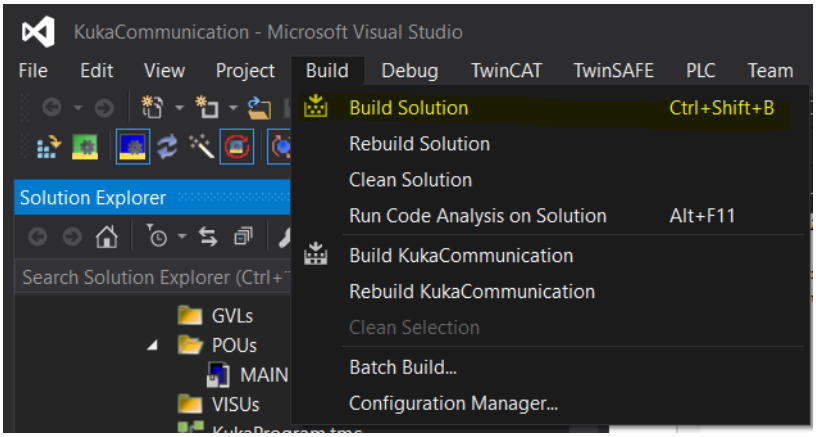


Figure A.23

22. The variables created in the program is now found under Instance.

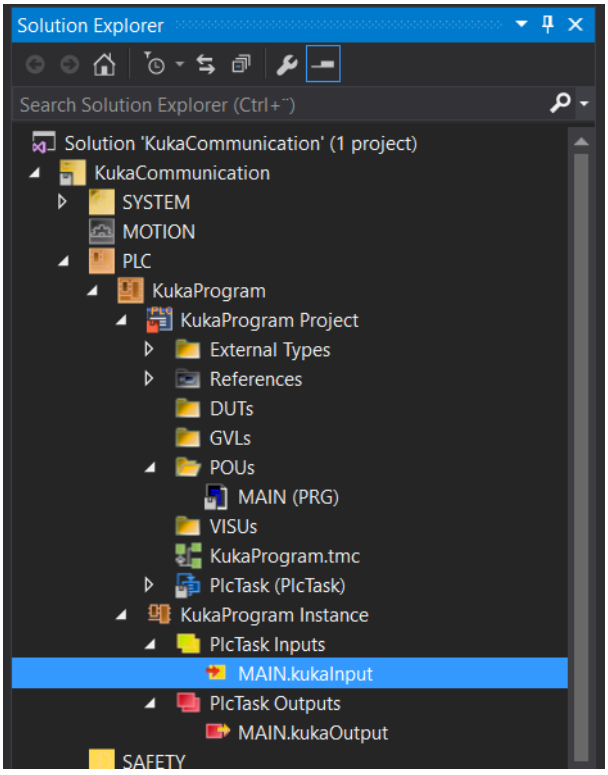


Figure A.24

23. Double click the variable and choose link to.

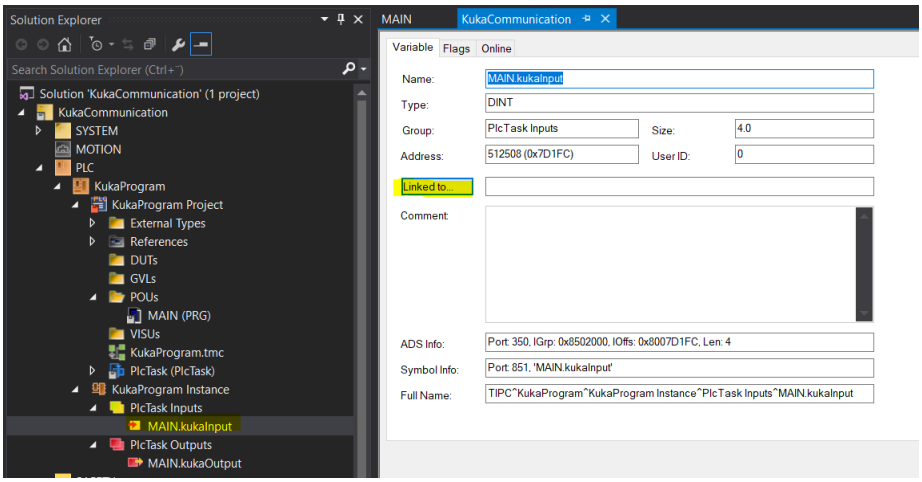


Figure A.25

24. Choose the variable made in the EL6692 module. Only the variable of same data type is showing.

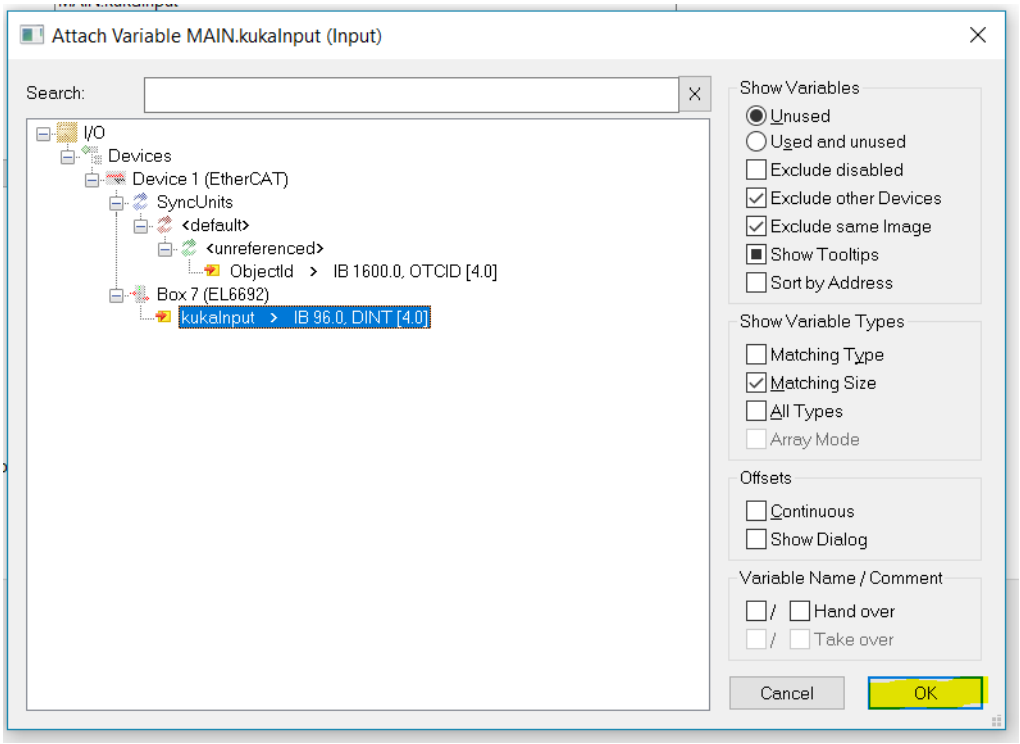


Figure A.26

25. Link the output variable.

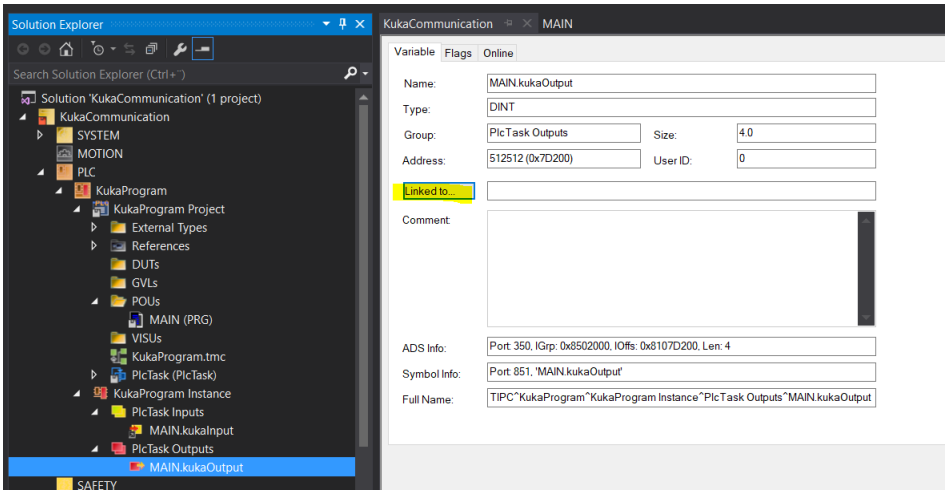


Figure A.27

26. Choose the variable on the EL6692.

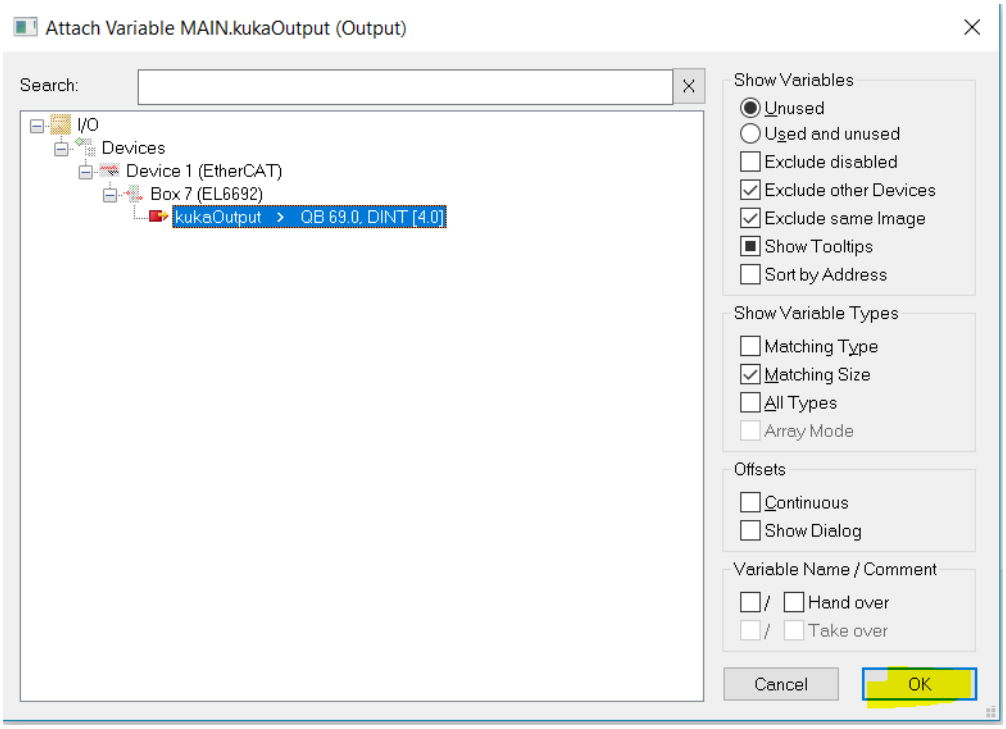


Figure A.28

27. The variables are now connected. Press the button with an Red arrow pointing on it. This button activate the configurations and sends them to the PLC.

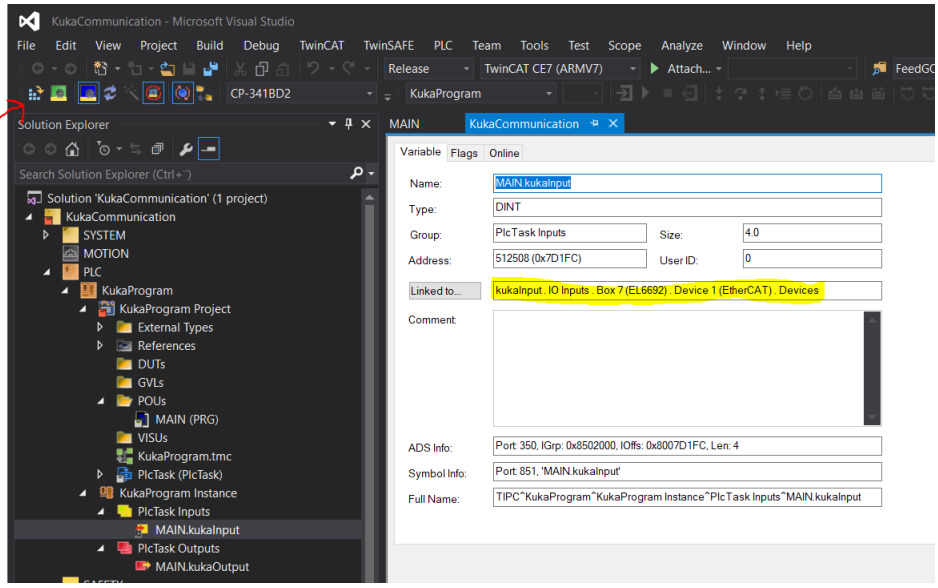


Figure A.29

28. Press OK to load the program to the PLC.

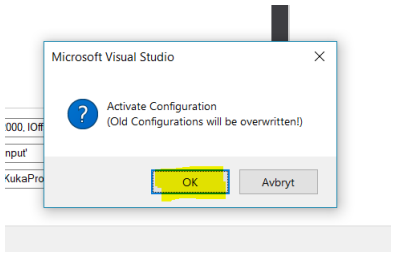


Figure A.30

29. Press the green door to log inn to the PLC.

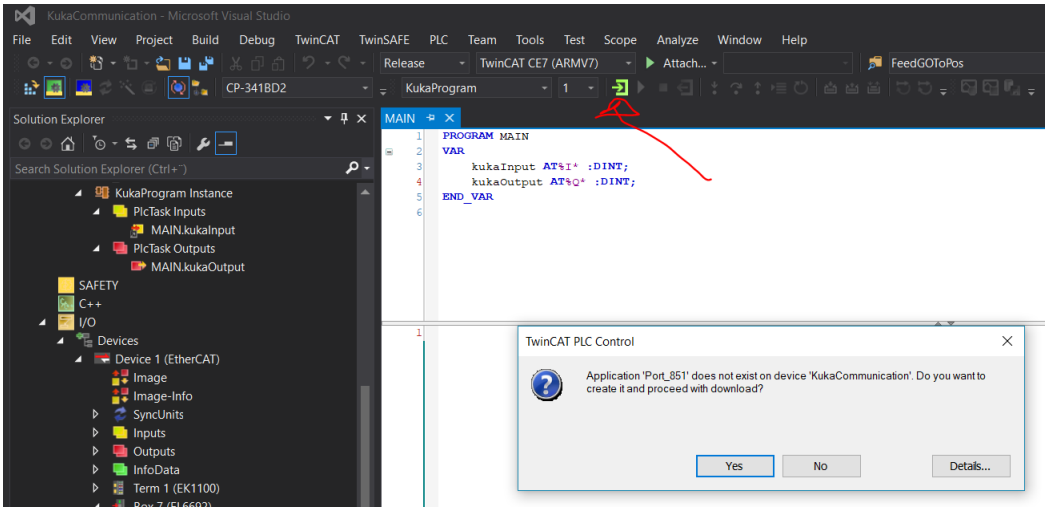


Figure A.31

30. An error will occur on the EL6692 module. This error is because the robot program does not have the same configuration on the robot side. The number of variables both input and output must be identical on both sides.

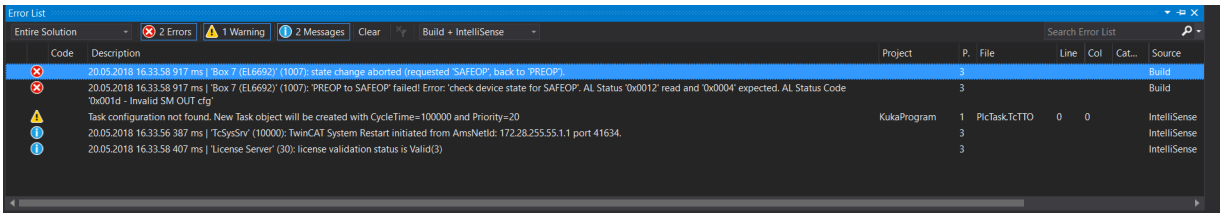


Figure A.32

IO configuration Robot side:

The Robot IO configuration is done in KUKA workvisual.

1. Connect your PC to the port X69 on the robot controller. Open WorkVisual, choose browse and refresh. The different programs on the Controller will show up, choose one.

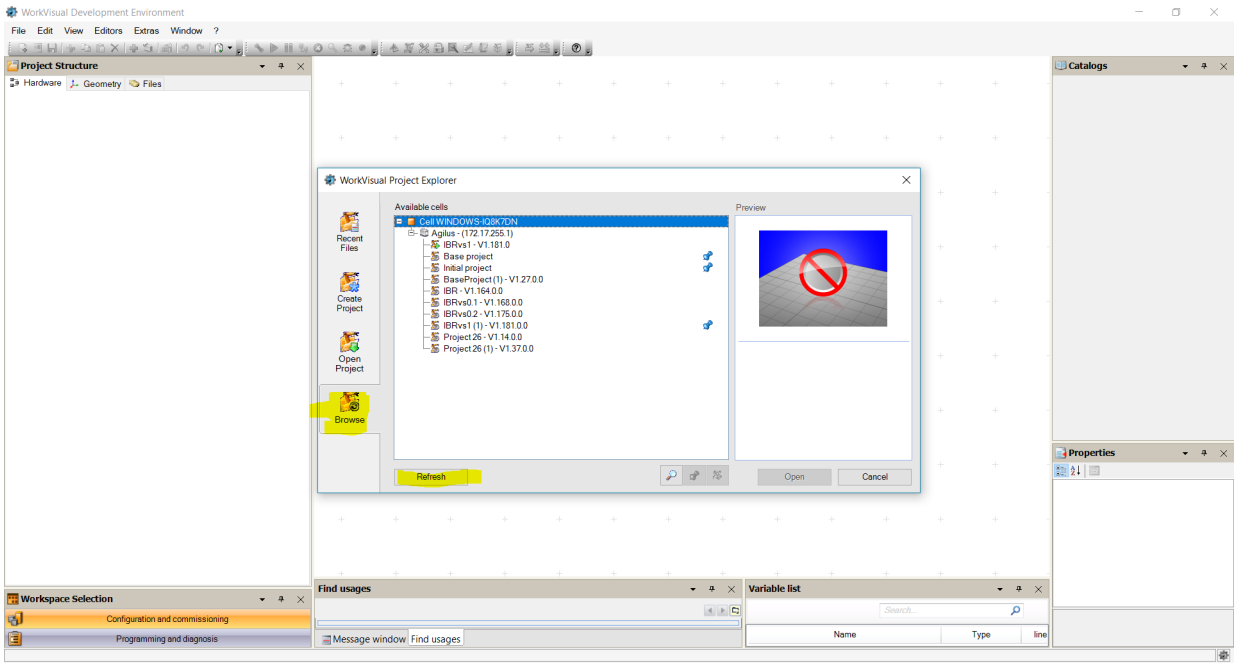


Figure A.33

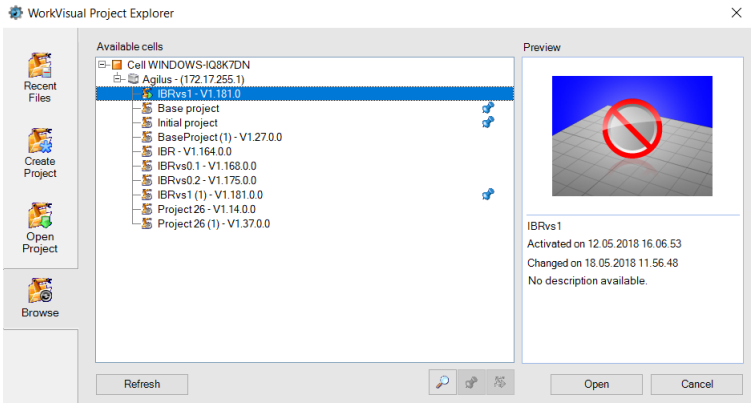


Figure A.34

2. After a while the Controller shows up in the Project structure.

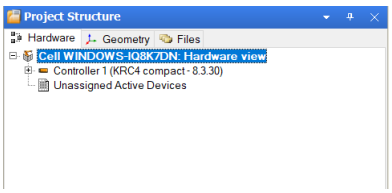


Figure A.35

3. Right click on the controller and set the controller active.

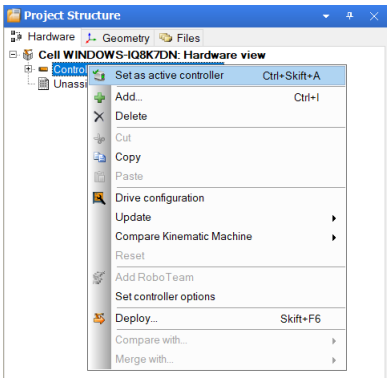


Figure A.36

4. The bridge module EL6692 is found under EK1100. This is how it should be when the bridge module is connected in the robot cabinet. If the bridge module is not found here, add it from the DTM catalog.

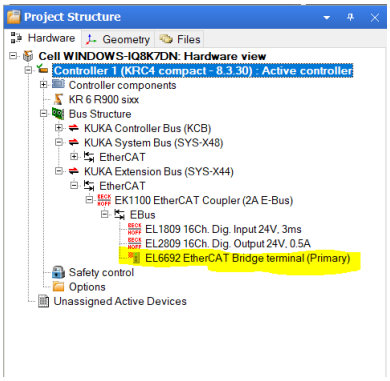


Figure A.37

5. By double clicking the EL6692 the window below shows up. This window is used to create the EL6692 variables. Choose the number of bytes which matches the input and output variable from the PLC. Meaning 4 bytes on both input and output.

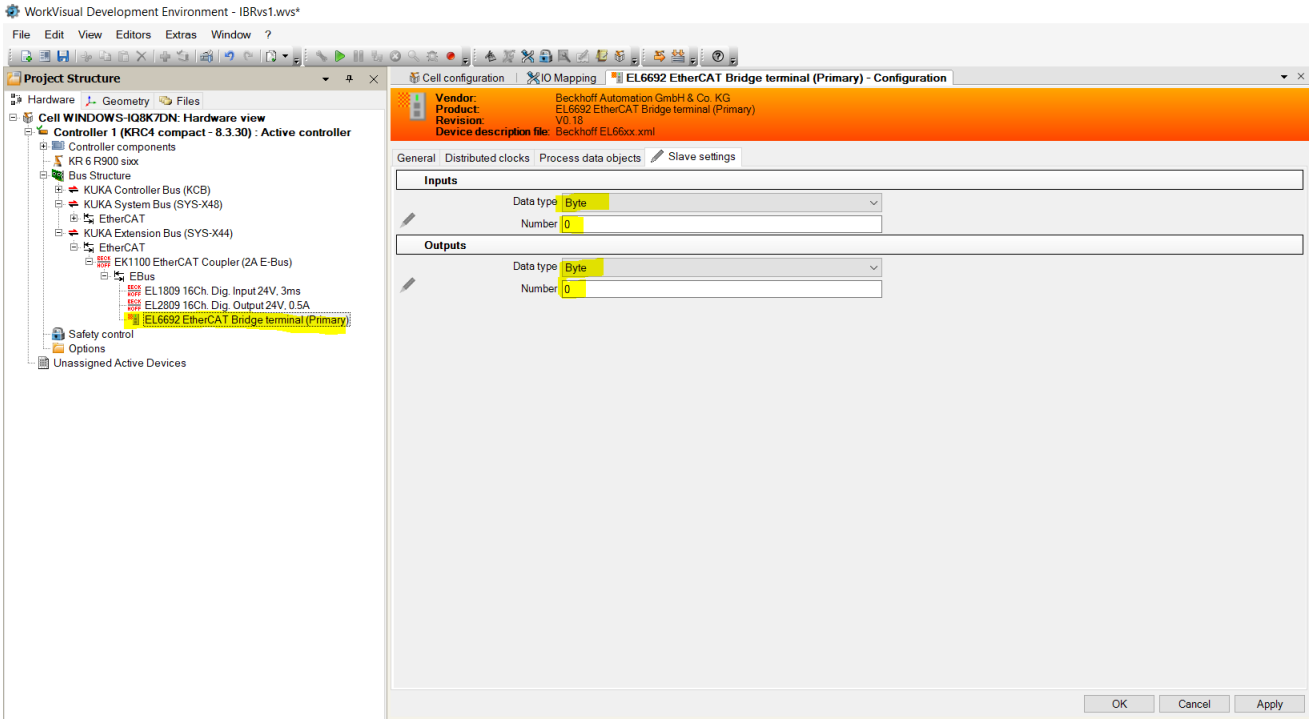


Figure A.38

6. Press apply.

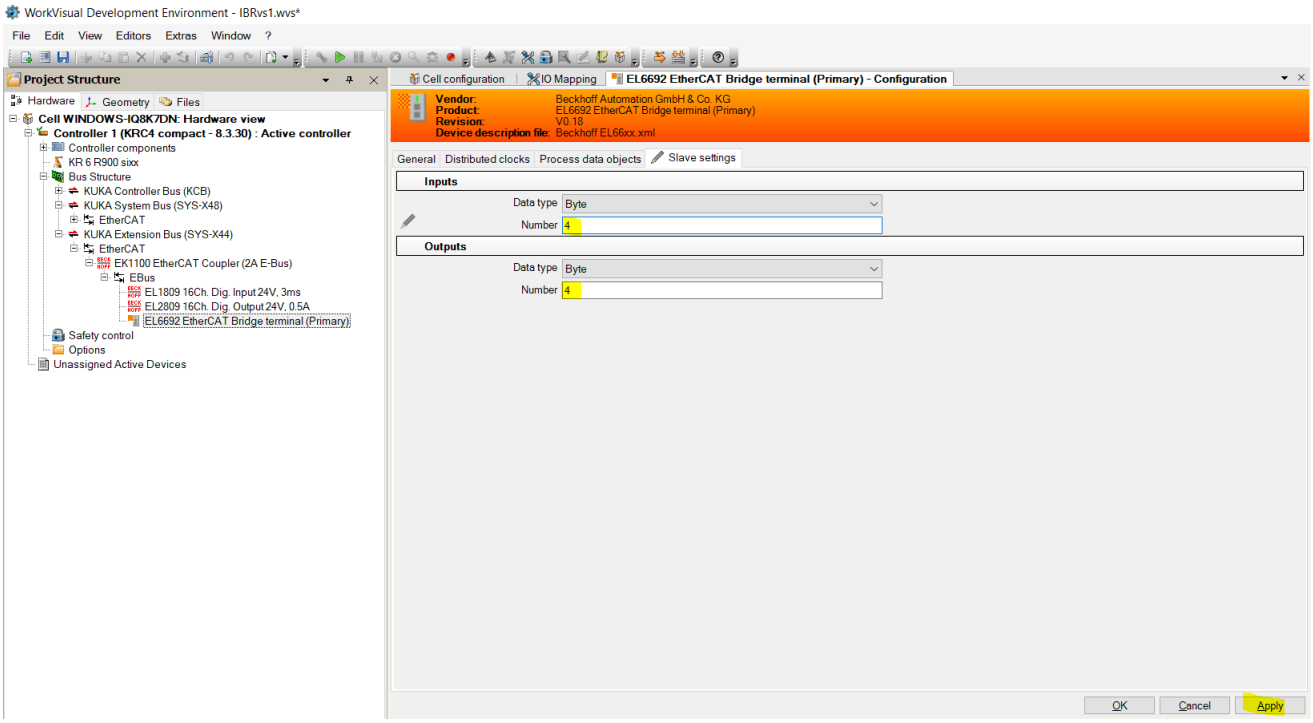


Figure A.39

7. On the bar on top there is a IO mapping button, press this.

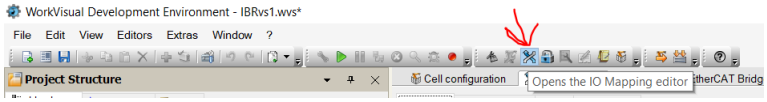


Figure A.40

8. The window for mapping the fieldbus addresses to the internal memory in the robotcontroller opens. choose KR C I/O and fieldbus. In the fieldbus window the Bridge module is choosen.

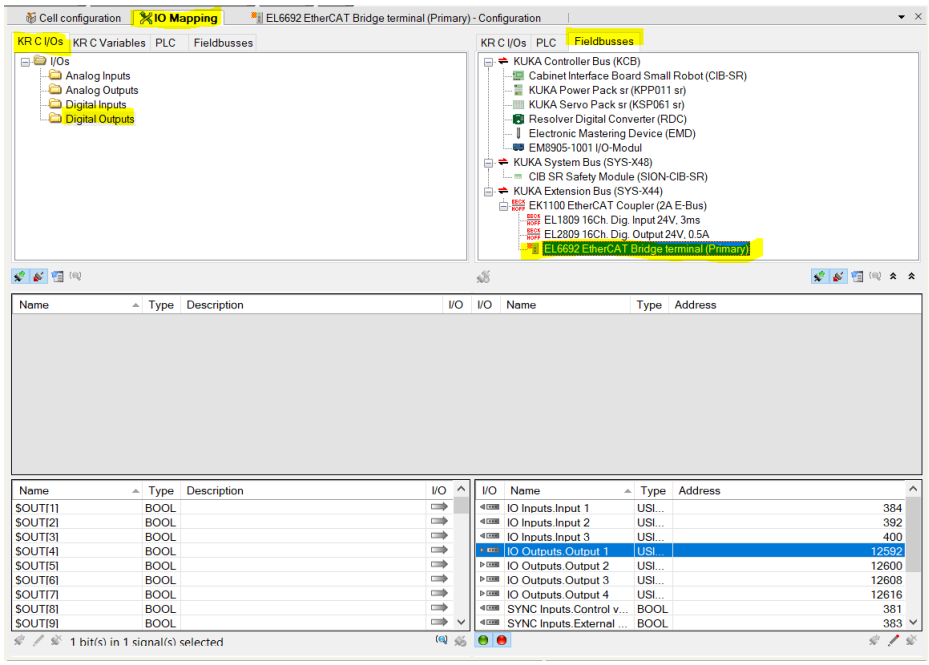


Figure A.41

9. The bottom right in the figure above contains the variables made in the EL6692 window. These variable now has to be linked with internal variables like we did for the PLC. These variables are found in the bottom left of the picture above. Since a KUKA bool = 1 bit, you need to group 8 bools together like the two figures below four times to create 4 bytes

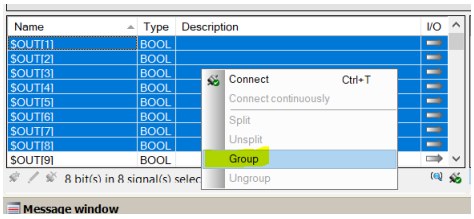


Figure A.42

10. Choose USINT. Make all the outputs needed, in this case this is 4 bytes of USINT.

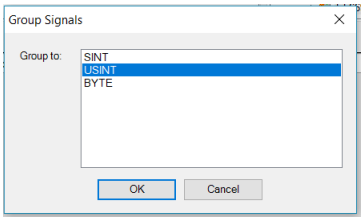


Figure A.43

11. Connect the newly created bytes to the USINT variables from the Bridge module. The output 1 is the first output and this is the first on the PLC side. Meaning that the first 4 output is the DINT made in the PLC program.

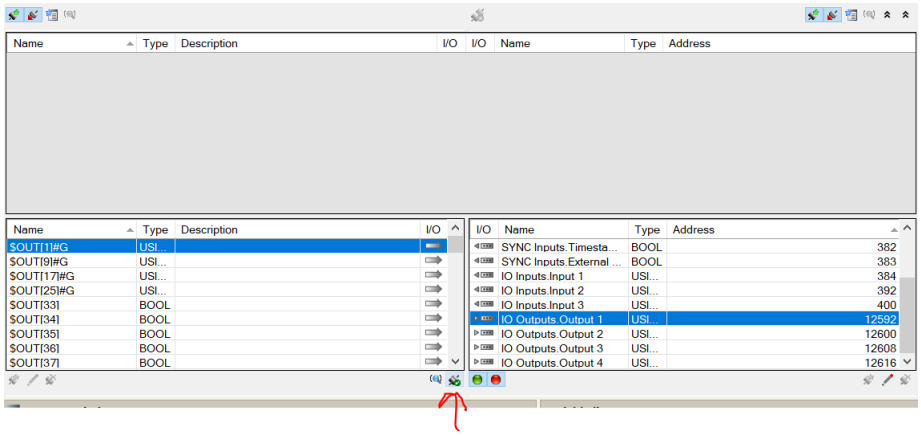


Figure A.44

12. The outputs are connected.

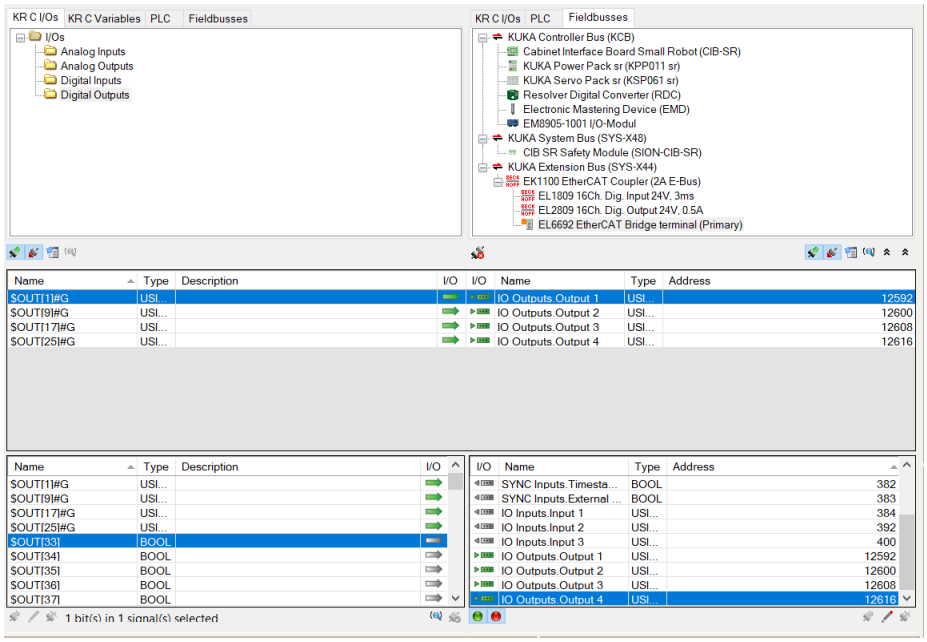


Figure A.45

13. Now connect the input variable from EL6692 the same way.

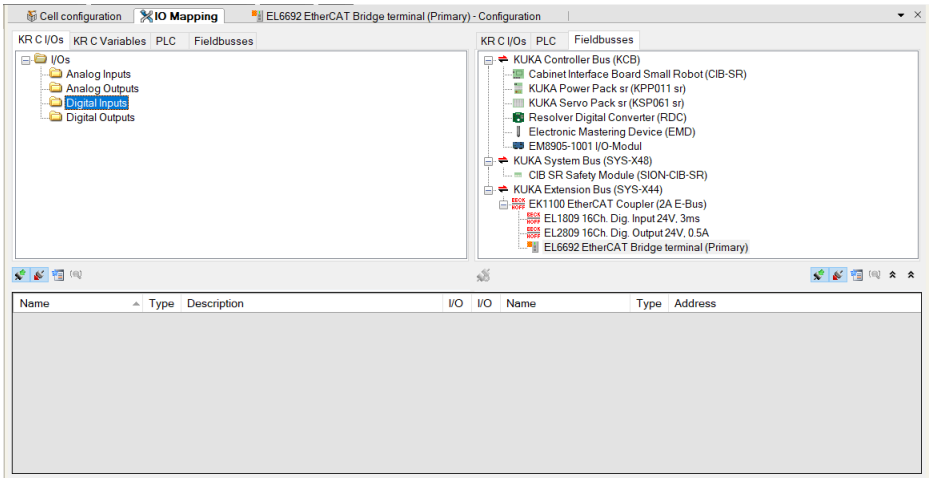


Figure A.46

14. Group 8 bits.

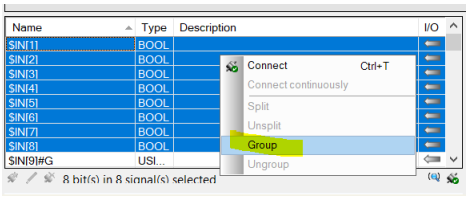


Figure A.47

15. Choose the data type USINT.

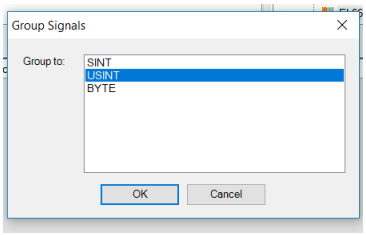


Figure A.48

16. Connect the inputs on the robot side with the inputs on the bridge module. Now the inputs and outputs are linked.

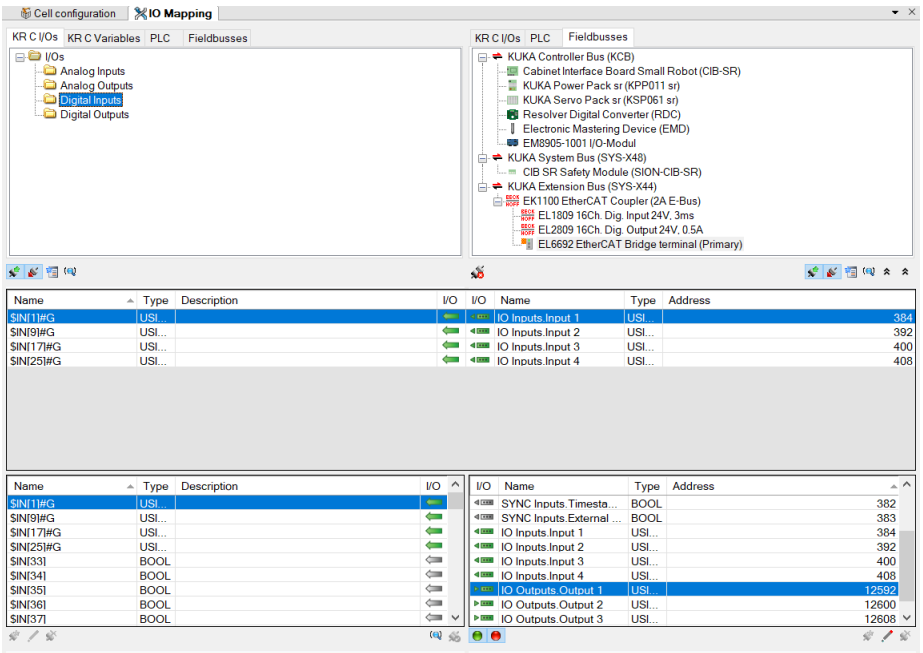


Figure A.49

17. The final step is to link the output bytes and input bytes into one 4byte output and one 4byte input like the PLC. Open the $config.dat file under Files in the Project structure.

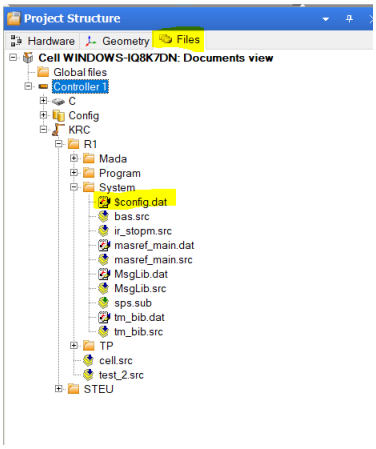


Figure A.50

18. Scroll down to the Userdefined Externals. Make the variable by typing SIGNAL variableFromPLC $IN[1] TO $IN[32] for input from the robot. and SIGNAL variableToPLC $OUT[1] TO $OUT[32] for the output to the PLC. It is the internal KR C address that is written in the $IN[address] and $OUT[address], this must be the addresses that the bytes on the bridge module where mapped to. When connecting 4 bytes together here this variable becomes DINT and will be SIGNED integer. The variabels is now ready to be used in the program.

```
683 ;===================================
684 ; Userdefined Externals
685 ;===================================
686 ; inputs
687 SIGNAL variableFromPLC $IN[1]  TO $IN[32]
688
689 ; outputs
690 SIGNAL variableToPLC $OUT[1]  TO $OUT[32]
691
692
```

Figure A.51

19. Before the program can be deployed login into the KUKA controller as expert or administator

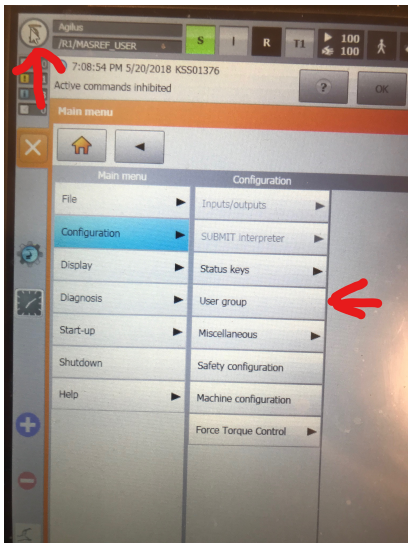


Figure A.52

20. The default password is "kuka" for both users.

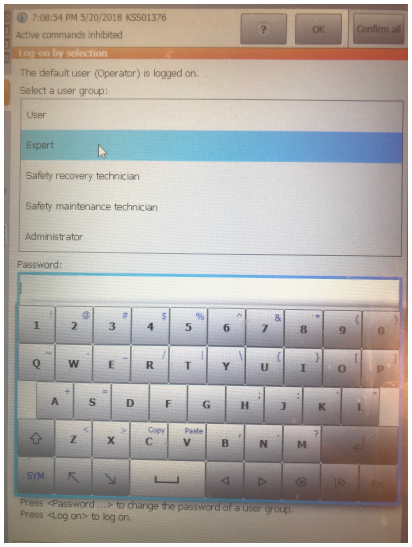


Figure A.53

21. To deploy the changes to the robot controller the button pointed on in the figure A.54.

Figure A.54

22. On the PC this window will show up. press the Finish button, to make everything go automatically or press Next until the deployment is finished.

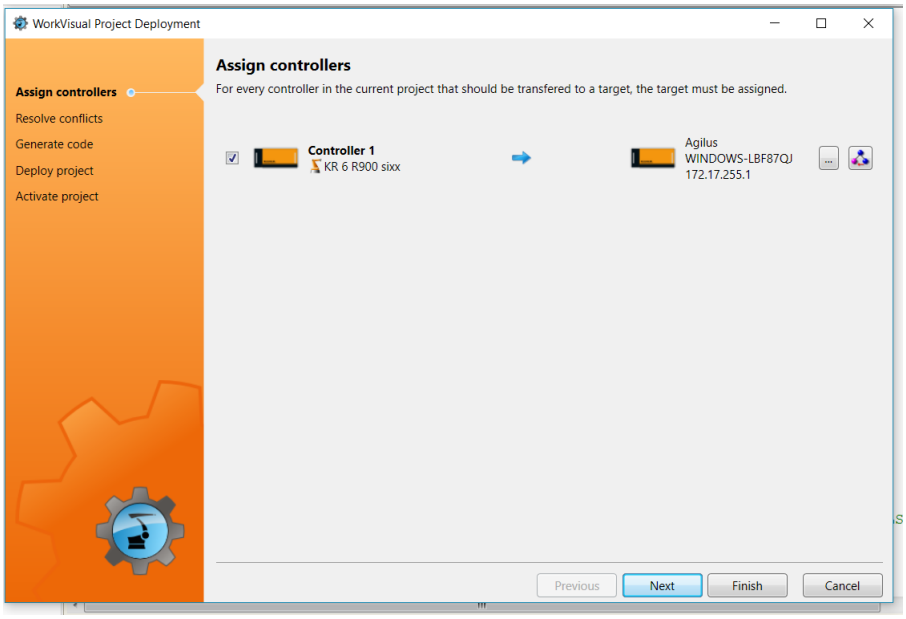


Figure A.55

23. Press yes on the pedant to allow the robot to receive the program.

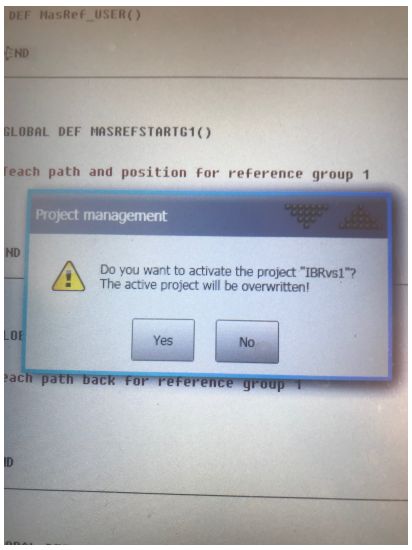


Figure A.56

24. press yes on the pedant to allow the robot to receive the program.

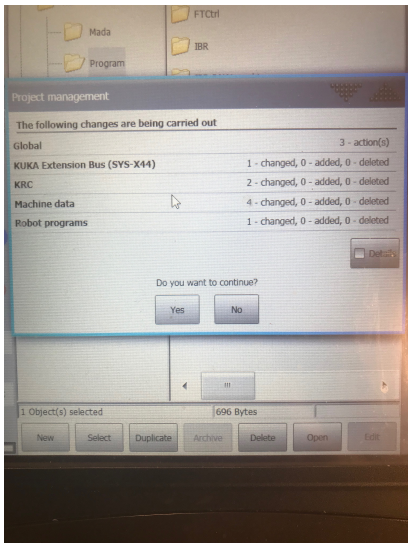


Figure A.57

25. The robot is updated. The error on the PLC is now removed if everything where done correct.

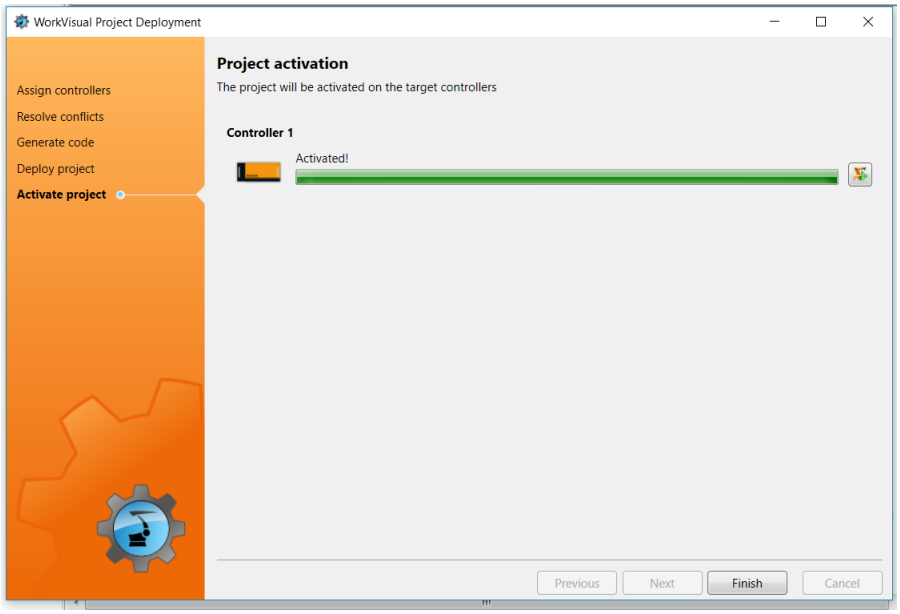


Figure A.58

Appendix B

Stepper calibration

This manual is a short description of how to setup and calibrate a stepper motor, controlled with a EL2521 module from Beckhoff. The stepper calibration is important in order for the distance moved to be correct.

Before starting the stepper calibration is it important to setup the EL2521 pulse train module correctly, without doing so the stepper motor wont move. Figure B.1 and B.2 below shows the two calibration steps which needs to be performed before configuring the stepper. It is important to mention that step one can be done offline and online, but step two has to be done while TwinCAT 3 is connected to the PLC. For more information regarding the EL2521 calibration read the EL2521 documentation (Beckhoff, 2017a).

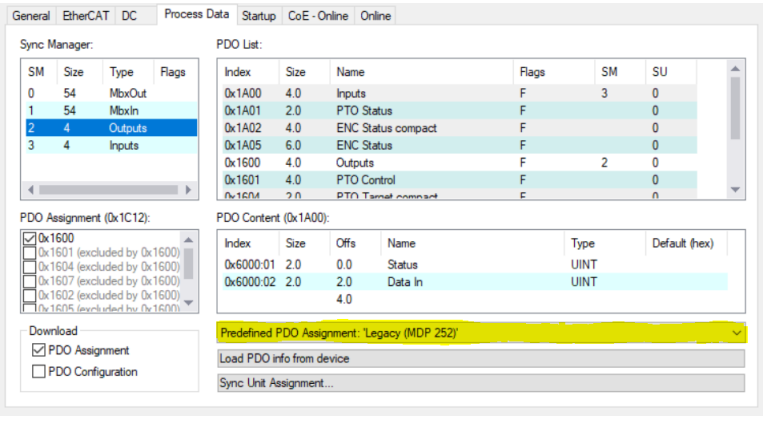


Figure B.1: EL2521 Configuration step one

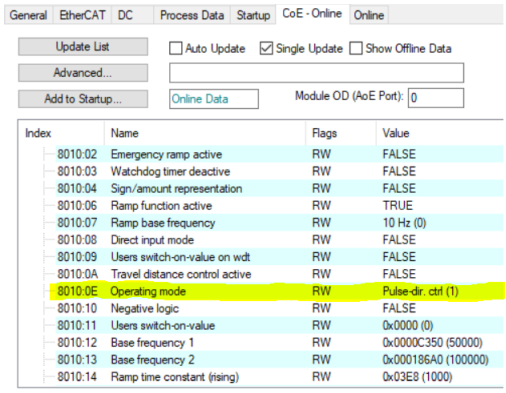


Figure B.2: EL2521 Configuration step two

The axis is connected to the EL2521 module and set up as Axis type: KL2521(Pulse Train Interface), see figure B.3 and the Enc-NC-Encoder is set to Encoder (KL5051/KL2502-30K/KL2521/IP2512/EL2521), see figure B.4. Setting the scaling of the stepper motor, where the numerator is how far the linear movement is on one turn and denominator is the number of steps, see figure B.5.

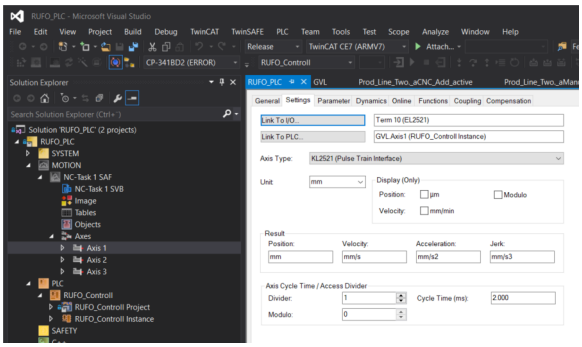


Figure B.3: Setting up the axis configuration

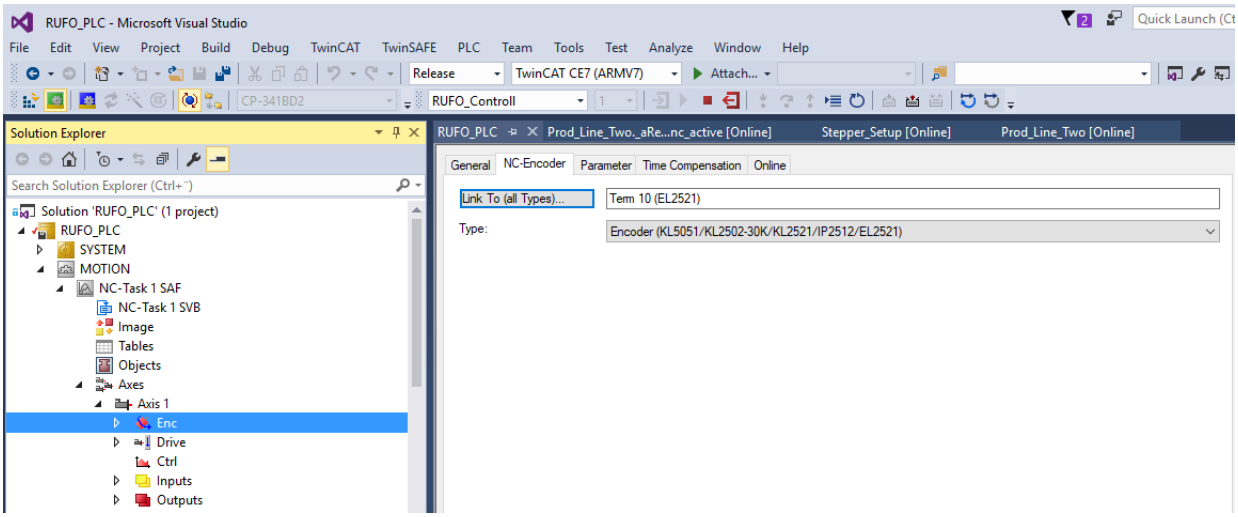


Figure B.4: Setting the encoder type

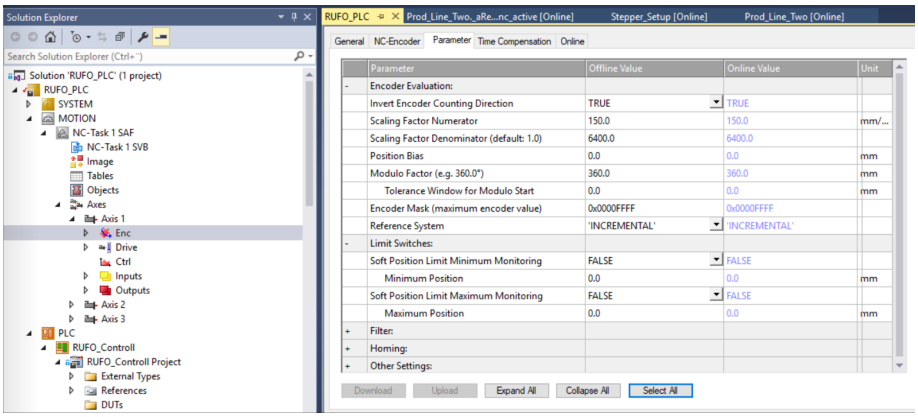


Figure B.5: Overview of the parameters for the encoder

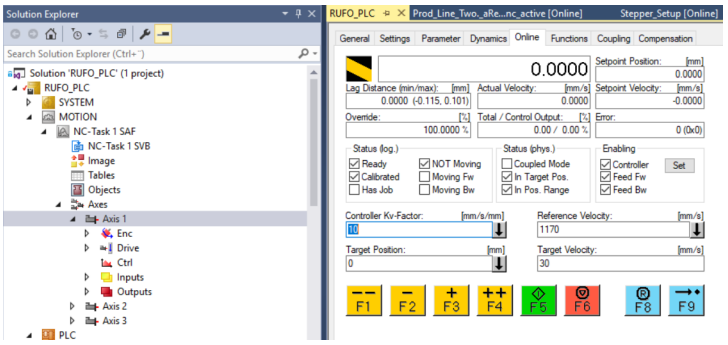


Figure B.6: Online control of the axis

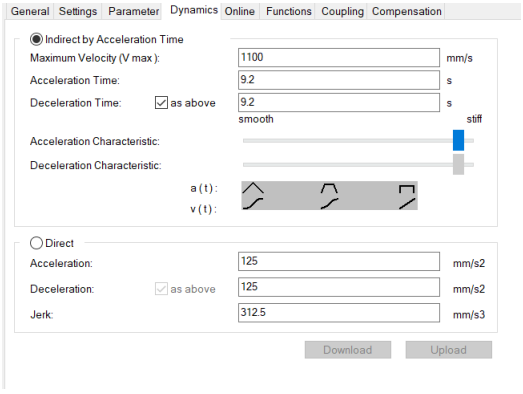


Figure B.7: Calibration parameters

The calibration is performed with adjusting the acceleration, maximum Velocity, Controller Kv-Factor and Reference Velocity. The calibration is done to have minimum overshoot and lag on the motion of the stepper motor and where performed with setting up a scope on the parameters actual position, set position, actual velocity, set velocity, data in, data out. The calibration was performed with taking one parameter at a time until the stepper followed the set point. The final result is displayed in figure B.8.

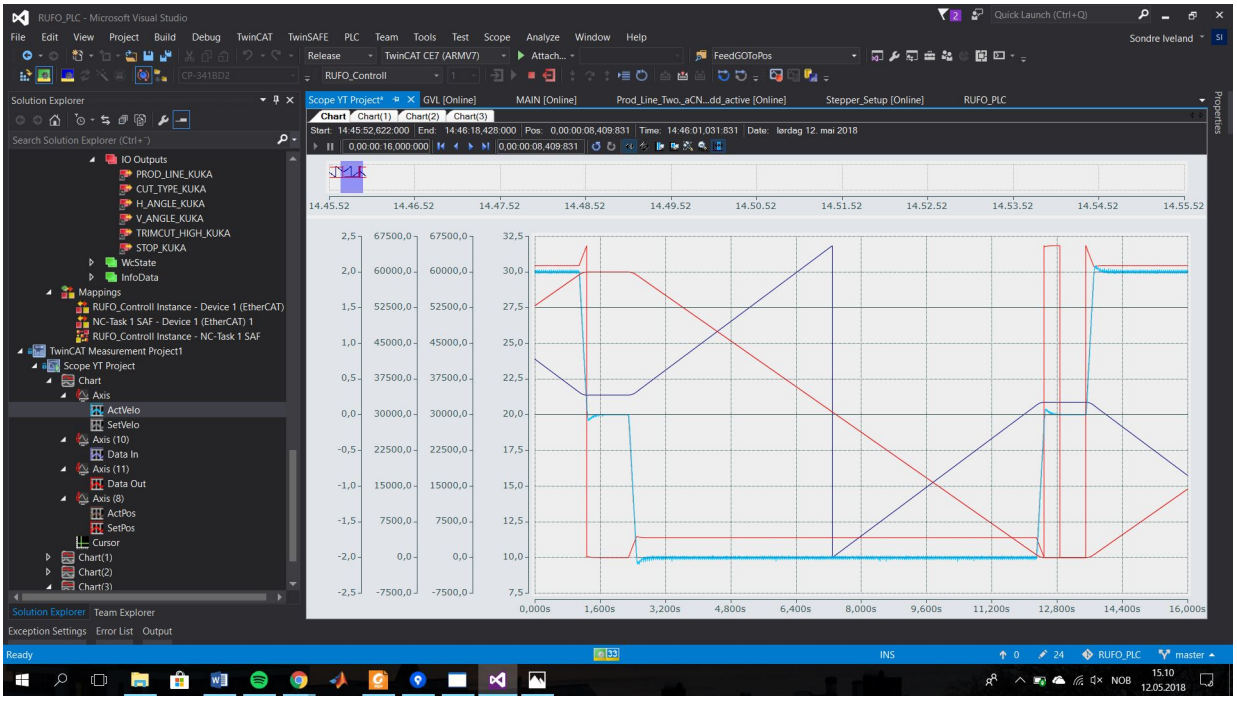


Figure B.8: Overview of the control of the Stepper motor

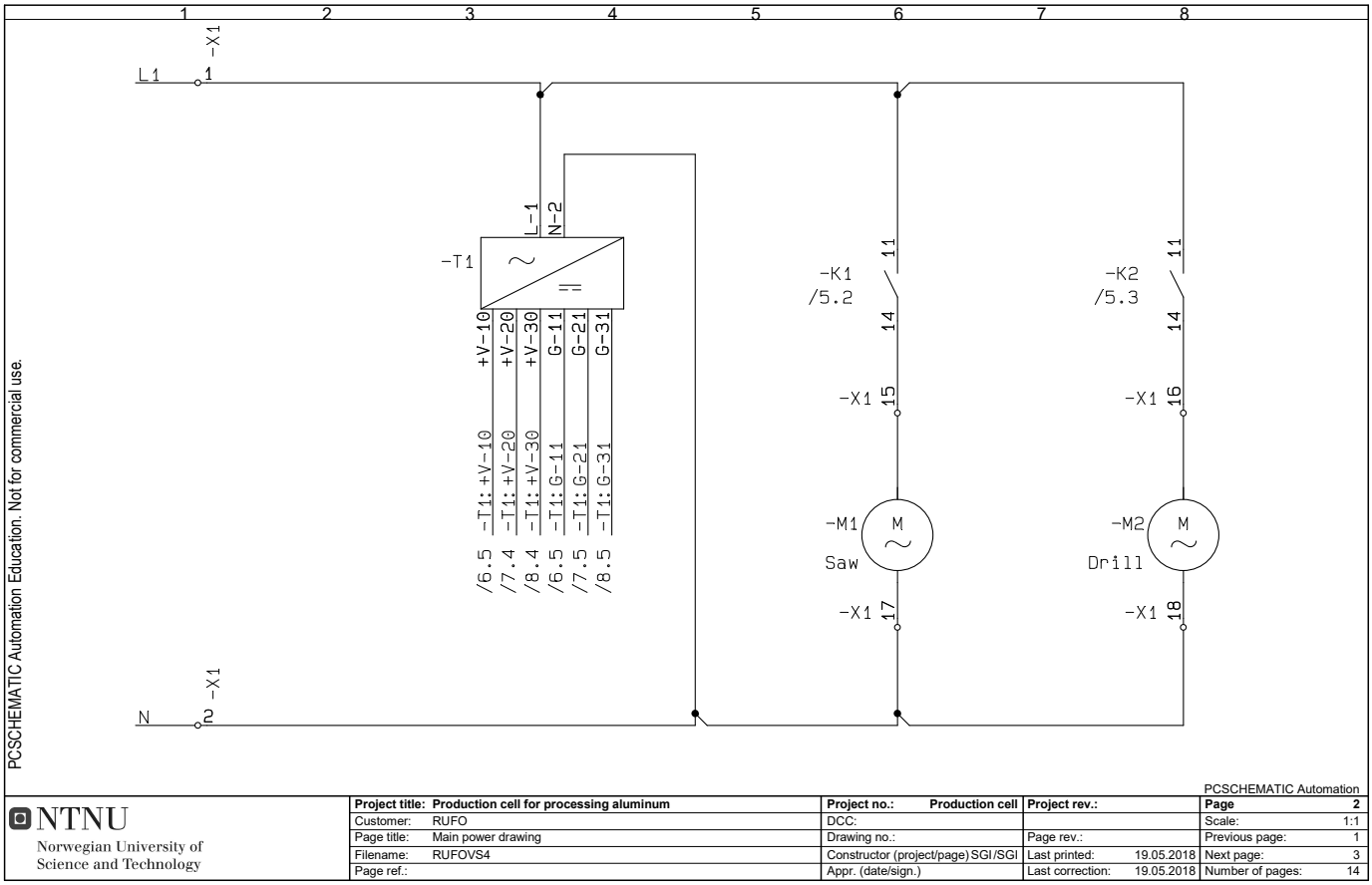
Appendix C

Electrical drawings

Electrical drawings for Production cell for processing aluminum

File name: RUFOVS4 PCSCHEMATIC Automation

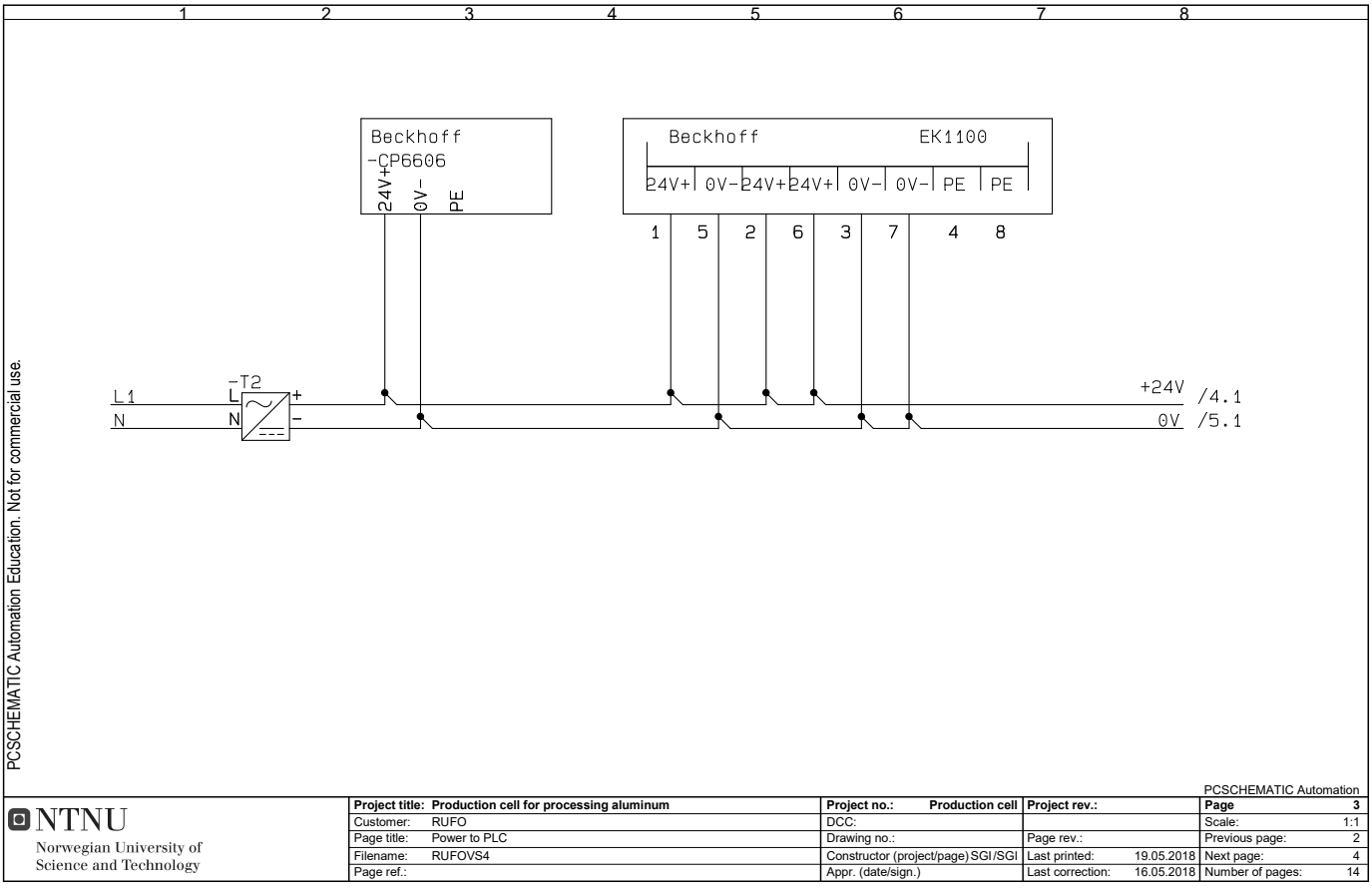
Customer: RUFO	**Project title: Production cell for processing aluminum**		
Page title: Front page	Page ref.:		
NTNU Norwegian University of Science and Technology	**Project no.: Production cell**	**Project rev.:**	**Page: 1**
	DCC:		Scale: 1:1
	Dwg. no.:	Page rev.:	Previous page:
	Eng. (proj/page): SGI SGI	Last print: 19.05.2018	Next page: 2
	Appr. (date/init):	Last edit: 16.05.2018	Total no. of pages: 14



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

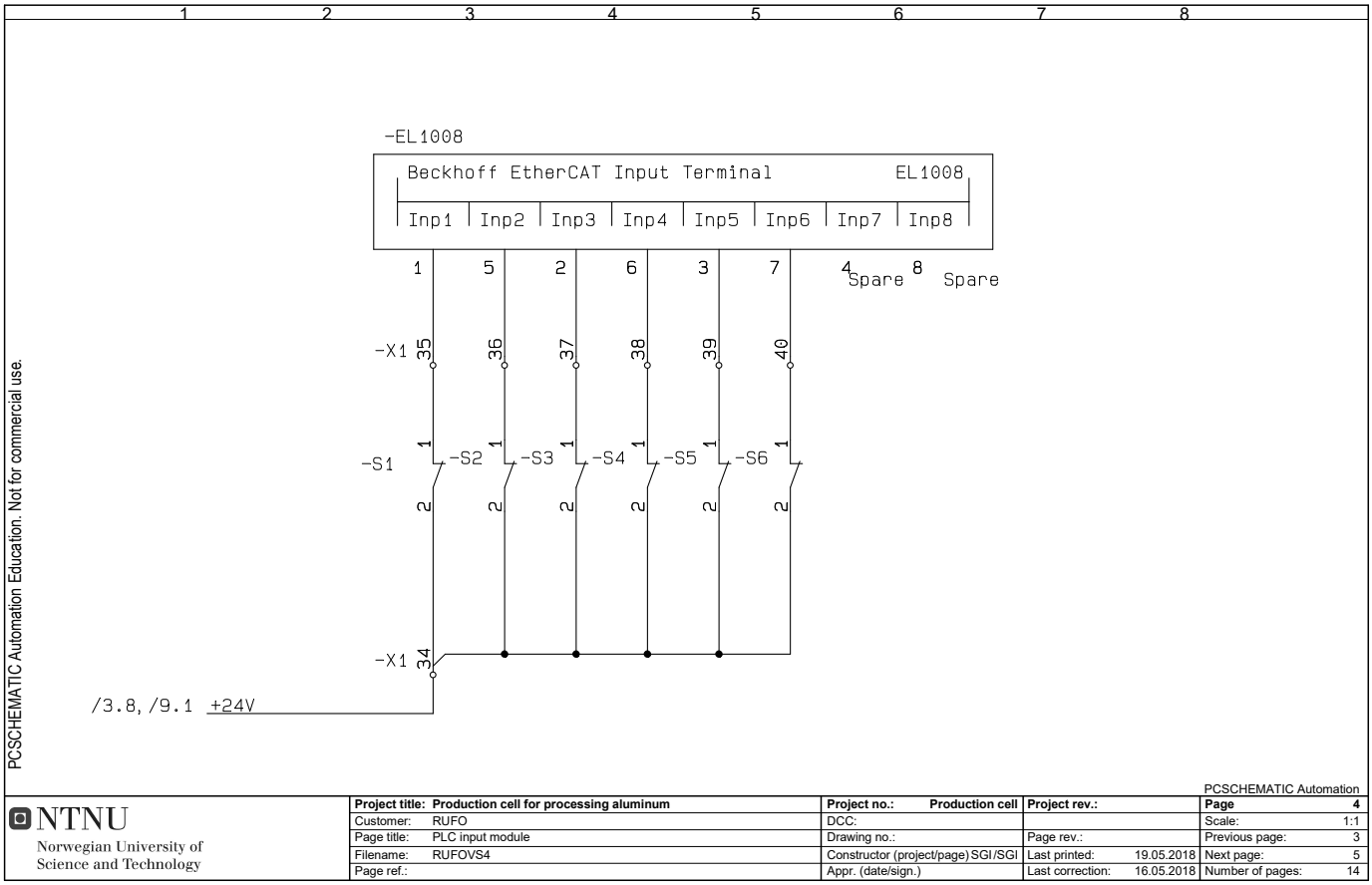
Project title:	**Production cell for processing aluminum**	**Project no.:**	**Production cell**	**Project rev.:**	**Page** **2**
Customer:	RUFO	DCC:			Scale: 1:1
Page title:	Main power drawing	Drawing no.:		Page rev.:	Previous page: 1
Filename:	RUFOVS4	Constructor (project/page)	SGI/SGI	Last printed: 19.05.2018	Next page: 3
Page ref.:		Appr. (date/sign.)		Last correction: 19.05.2018	Number of pages: 14



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

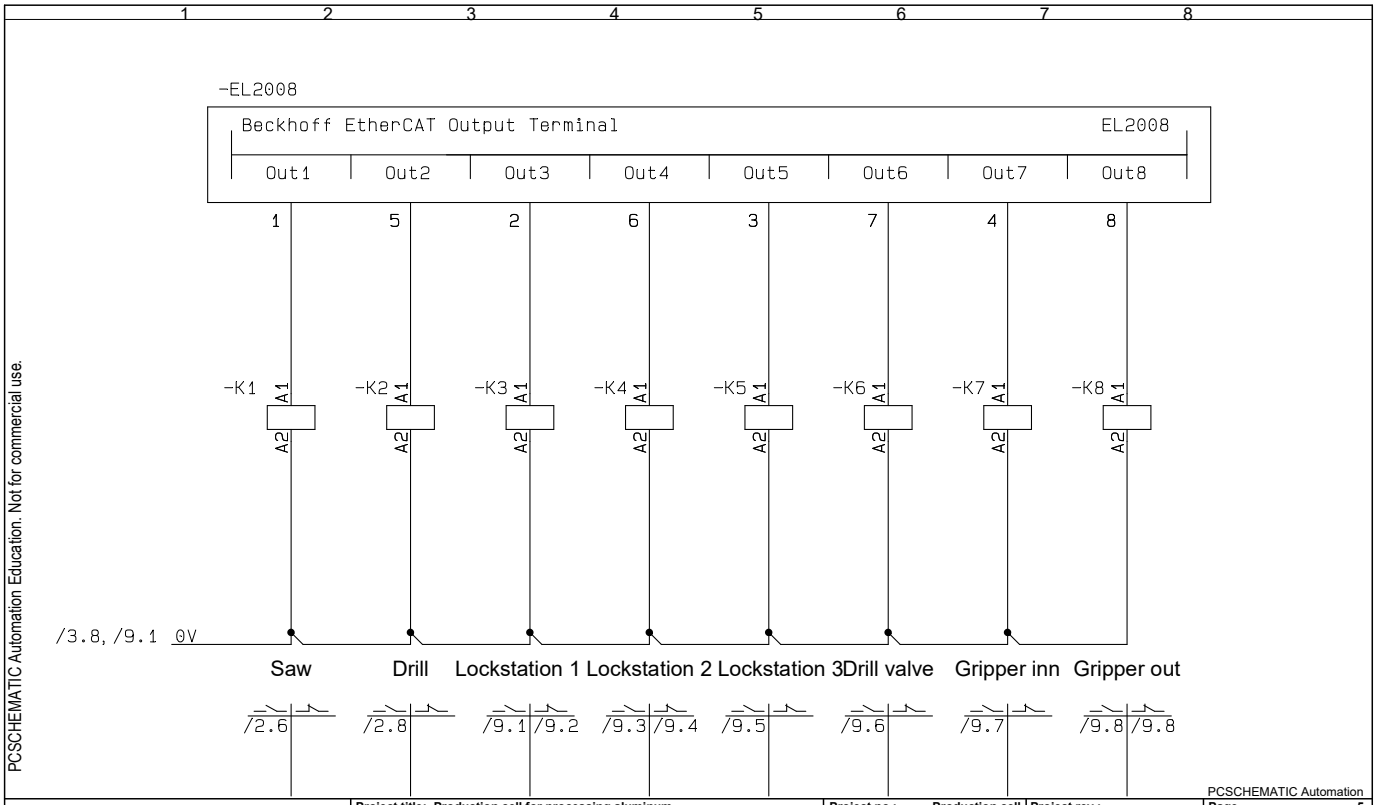
Project title:	**Production cell for processing aluminum**	**Project no.:**	**Production cell**	**Project rev.:**	**Page** 3
Customer:	RUFO	DCC:			Scale: 1:1
Page title:	Power to PLC	Drawing no.:		Page rev.:	Previous page: 2
Filename:	RUFOVS4	Constructor (project/page)	SGI/SGI	Last printed: 19.05.2018	Next page: 4
Page ref.:		Appr. (date/sign.)		Last correction: 16.05.2018	Number of pages: 14



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

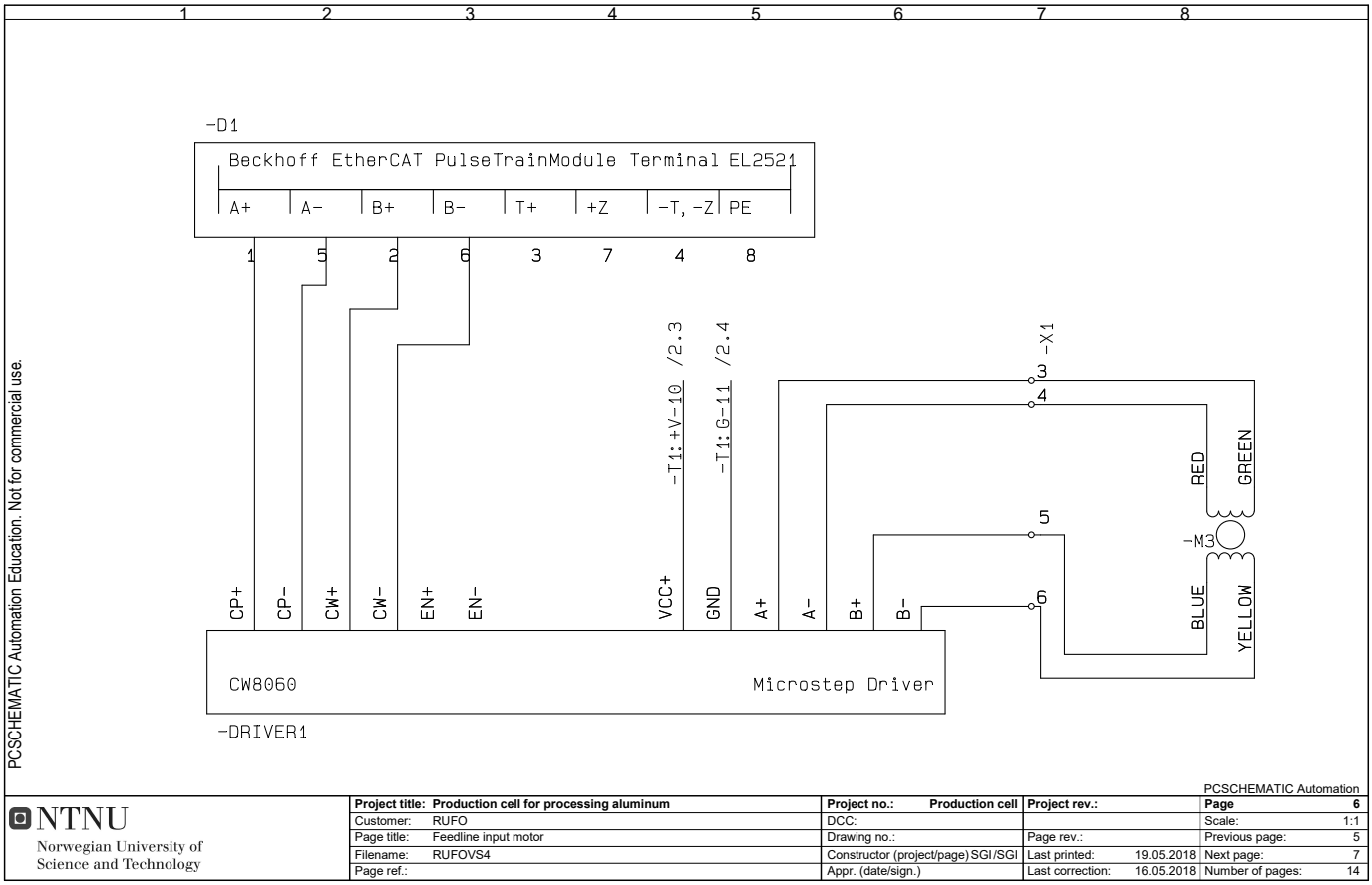
Project title:	**Production cell for processing aluminum**	**Project no.:**	**Production cell**	**Project rev.:**	**Page** 4
Customer:	RUFO	DCC:			Scale: 1:1
Page title:	PLC input module	Drawing no.:		Page rev.:	Previous page: 3
Filename:	RUFOVS4	Constructor (project/page)	SGI/SGI	Last printed: 19.05.2018	Next page: 5
Page ref.:		Appr. (date/sign.)		Last correction: 16.05.2018	Number of pages: 14



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

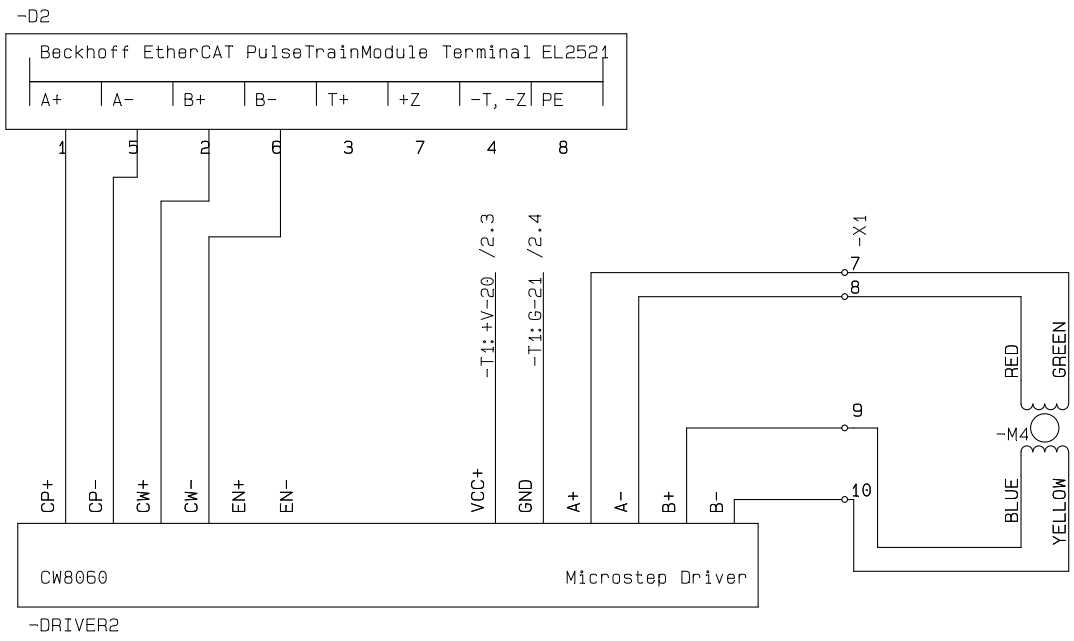
Project title:	**Production cell for processing aluminum**	**Project no.:**	**Production cell**	**Project rev.:**	**Page** 5
Customer:	RUFO	DCC:			Scale: 1:1
Page title:	PLC output module	Drawing no.:		Page rev.:	Previous page: 4
Filename:	RUFOVS4	Constructor (project/page)	SGI/SGI	Last printed: 19.05.2018	Next page: 6
Page ref.:		Appr. (date/sign.)		Last correction: 16.05.2018	Number of pages: 14



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

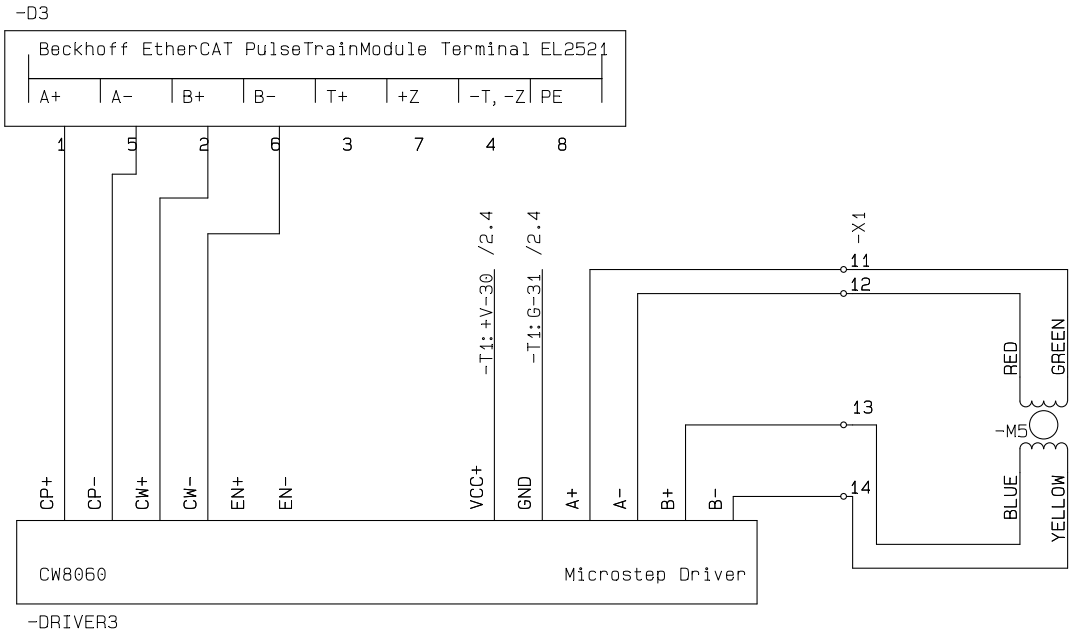
Project title:	**Production cell for processing aluminum**	**Project no.:**	**Production cell**	**Project rev.:**	**Page** **6**
Customer:	RUFO	DCC:			Scale: 1:1
Page title:	Feedline input motor	Drawing no.:		Page rev.:	Previous page: 5
Filename:	RUFOVS4	Constructor (project/page)	SGI/SGI	Last printed: 19.05.2018	Next page: 7
Page ref.:		Appr. (date/sign.)		Last correction: 16.05.2018	Number of pages: 14



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

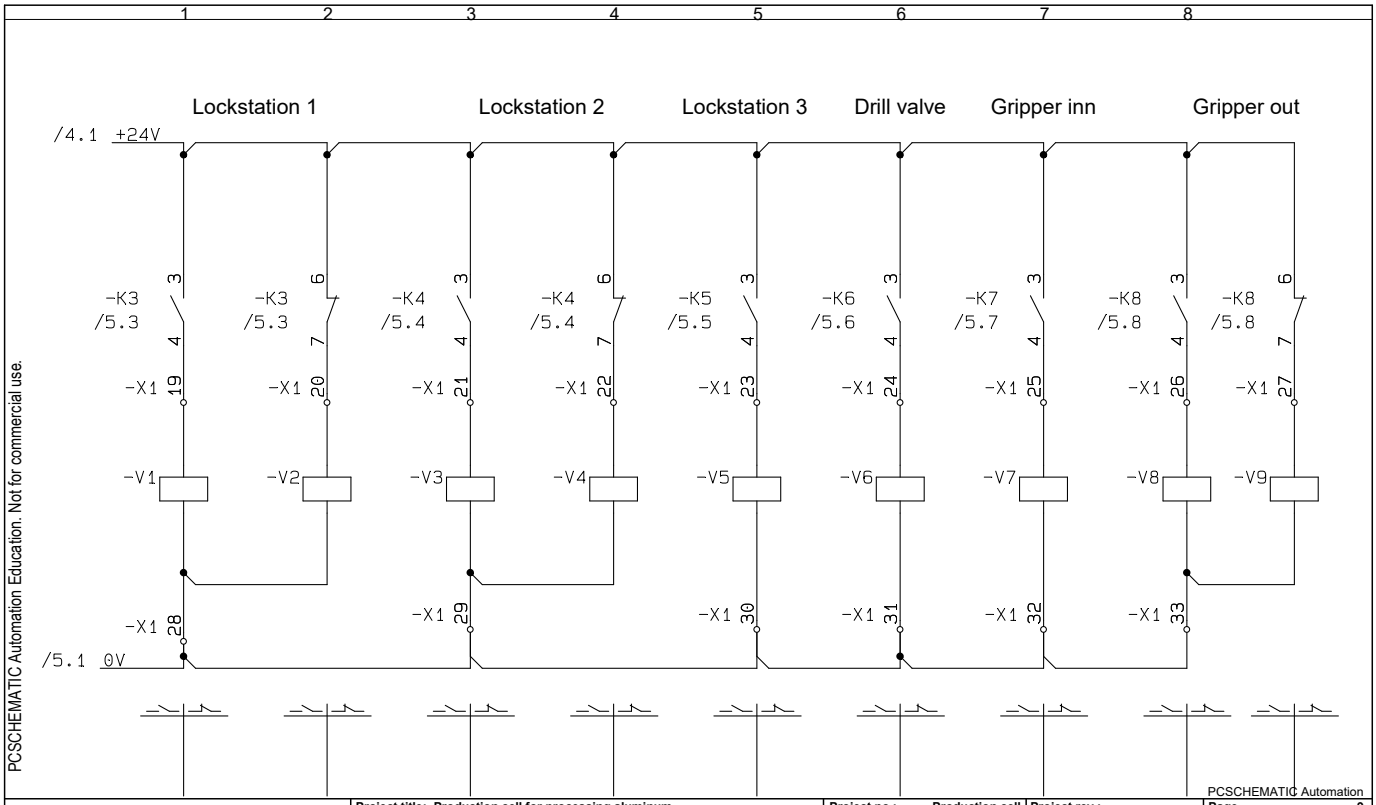
Project title:	**Production cell for processing aluminum**	**Project no.:** **Production cell**	**Project rev.:**	**Page**	**7**
Customer:	RUFO	DCC:		Scale:	1:1
Page title:	Drill height motor	Drawing no.:	Page rev.:	Previous page:	6
Filename:	RUFOVS4	Constructor (project/page) SGI/SGI	Last printed: 19.05.2018	Next page:	8
Page ref.:		Appr. (date/sign.)	Last correction: 16.05.2018	Number of pages:	14



PCSCHEMATIC Automation

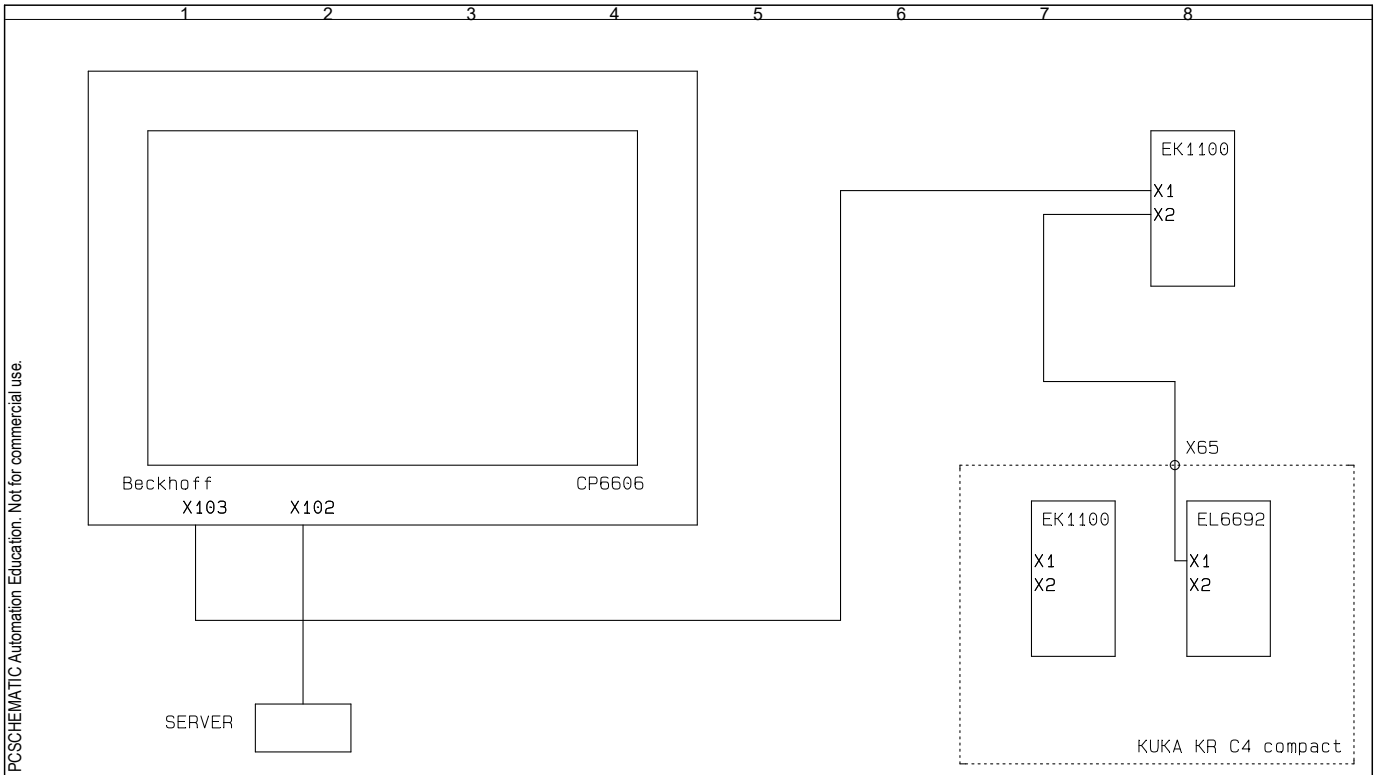
NTNU
Norwegian University of Science and Technology

Project title:	**Production cell for processing aluminum**	**Project no.:** **Production cell**	**Project rev.:**	**Page**	**8**
Customer:	RUFO	DCC:		Scale:	1:1
Page title:	Feedline output motor	Drawing no.:	Page rev.:	Previous page:	7
Filename:	RUFOVS4	Constructor (project/page) SGI/SGI	Last printed: 19.05.2018	Next page:	9
Page ref.:		Appr. (date/sign.)	Last correction: 16.05.2018	Number of pages:	14



NTNU
Norwegian University of Science and Technology

Project title:	**Production cell for processing aluminum**	**Project no.:** **Production cell**	**Project rev.:**	**Page**	**9**
Customer:	RUFO	DCC:		Scale:	1:1
Page title:	Pneumatic valve connections	Drawing no.:	Page rev.:	Previous page:	8
Filename:	RUFOVS4	Constructor (project/page) SGI/SGI	Last printed: 19.05.2018	Next page:	10
Page ref.:		Appr. (date/sign.)	Last correction: 16.05.2018	Number of pages:	14



PCSCHEMATIC Automation

NTNU
Norwegian University of Science and Technology

Project title:	**Production cell for processing aluminum**	**Project no.:** **Production cell**	**Project rev.:**	**Page**	**10**
Customer:	RUFO	DCC:		Scale:	1:1
Page title:	Communication drawing	Drawing no.:	Page rev.:	Previous page:	9
Filename:	RUFOVS4	Constructor (project/page) SGI/SGI	Last printed: 19.05.2018	Next page:	11
Page ref.:		Appr. (date/sign.)	Last correction: 16.05.2018	Number of pages:	14

Terminal		From (int.)	Wire no. (int.)	To (ext.)	Cable (ext.)	Term. type	Pos.	Manufacturer
-X1	1	-T1 L-1		-T2 L			/2.1	
	2	-T1 N-2		-T2 N			/2.1	
	3	-M3 GREEN		-DRIVER1 A+			/6.7	
	4	-M3 RED		-DRIVER1 A-			/6.7	
	5	-M3 BLUE		-DRIVER1 B+			/6.7	
	6	-M3 YELLOW		-DRIVER1 B-			/6.7	
	7	-M4 GREEN		-DRIVER2 A+			/7.7	
	8	-M4 RED		-DRIVER2 A-			/7.7	
	9	-M4 BLUE		-DRIVER2 B+			/7.7	
	10	-M4 YELLOW		-DRIVER2 B-			/7.7	
	11	-M5 GREEN		-DRIVER3 A+			/8.7	
	12	-M5 RED		-DRIVER3 A-			/8.7	
	13	-M5 BLUE		-DRIVER3 B+			/8.7	
	14	-M5 YELLOW		-DRIVER3 B-			/8.7	
	19	-K3 4		-V1 A1			/9.1	
	20	-K3 7		-V2 A1			/9.2	
	21	-K4 4		-V3 A1			/9.3	
	22	-K4 7		-V4 A1			/9.4	
	23	-K5 4		-V5 A1			/9.5	
	24	-K6 4		-V6 A1			/9.6	
	25	-K7 4		-V7 A1			/9.7	
	26	-K8 4		-V8 A1			/9.8	
	27	-K8 7		-V9 A1			/9.8	
	28	-V2 A2					/9.1	
	28	-V1 A2		0V			/9.1	
	29	-V4 A2					/9.3	
	29	-V3 A2		0V			/9.3	
	30	-V5 A2		0V			/9.5	

NTNU
Norwegian University of Science and Technology

Project title:	**Production cell for processing aluminum**	**Project no.:**	**Production cell**	**Project rev.:**		**Page**	**11**
Customer:	RUFO	DCC:				Scale:	1:1
Page title:	Terminal list External / Internal	Dwg. no.:		Page rev.:		Previous page:	10
File name:	RUFOVS4	Eng. (proj/page):	SGI/ SGI	Last print:	19.05.2018	Next page:	12
Page ref.:		Appr. (date/init):		Last edit:	16.05.2018	Total no. of pages:	14

Terminal	From (int.)	Wire no. (int.)	To (ext.)	Cable (ext.)	Term. type	Pos.	Manufacturer
-X1 31	-V6 A2		0V			/9.6	
32	-V7 A2		0V			/9.7	
33	-V9 A2					/9.8	
33	-V8 A2		0V			/9.8	
34	-S2 2					/4.3	
34	-S1 2		+24V			/4.3	
35	-EL1008 1		-S1 1			/4.3	
36	-EL1008 5		-S2 1			/4.3	
37	-EL1008 2		-S3 1			/4.4	
38	-EL1008 6		-S4 1			/4.4	
39	-EL1008 3		-S5 1			/4.5	
40	-EL1008 7		-S6 1			/4.5	
15	-K1 14					/2.6	
16	-K2 14					/2.8	
17			-X1 18			/2.6	
17	-M1 2		-T1 N-2			/2.6	
18	-M2 2		-X1 17			/2.8	

Component	Article no.	Type	Function	Description	Manufacturer	Position
-CP6606			PLC screen			/3.3
-D1			Pulse train module			/6.3
-D2			Pulse train module			/7.3
-D3		EL2521	Pulse train module			/8.3
-DRIVER1			Stepper driver			/6.5
-DRIVER2			Stepper driver			/7.4
-DRIVER3			Stepper driver			/8.4
-EK1100			PLC coupler			/3.6
-EK1100 IN ROBOT			PLC coupler			/10.7
-EL1008			Input terminal			/4.5
-EL2008			Output terminal			/5.5
-EL6692			Bridge module			/10.8
-K1			Saw relay			/5.2
-K2			Drill relay			/5.3
-K3			Lock station 1 relay			/5.3
-K4			Lock station 2 relay			/5.4
-K5			Lock station 3 relay			/5.5
-K6			Drill valve relay			/5.6
-K7			Gripper feedline relay			/5.7
-K8			Gripper Output relay			/5.8
-M1			Saw			/2.6
-M2			Drill			/2.8
-M3			Stepper motor			/6.8
-M4			Stepper motor			/7.8
-M5			Stepper motor			/8.8
-S1			Limit Switch feed home			/4.3
-S2			Limit Switch feed out			/4.3
-S3			Limit Switch drill home			/4.4

NTNU
Norwegian University of Science and Technology

Project title:	**Production cell for processing aluminum**	**Project no.:** **Production cell**	**Project rev.:**	**Page**	**13**
Customer:	RUFO	DCC:		Scale:	1:1
Page title:	Components list	Dwg. no.:	Page rev.:	Previous page:	12
File name:	RUFOVS4	Eng. (proj/page): SGI/ SGI	Last print: 19.05.2018	Next page:	14
Page ref.:		Appr. (date/init):	Last edit: 16.05.2018	Total no. of pages:	14

Component	Article no.	Type	Function	Description	Manufacturer	Position
-S4			Limit Switch drill up			/4.4
-S5			Limit Switch output home			/4.5
-S6			Limit Switch output out			/4.5
-T1			power supply			/2.3
-T2	4046356046640	TRIO-PS/1AC/24DC/5		Power supply unit	PHOENIX CONTACT GmbH & Co. KG	/3.2
-V1			Lockstation1 Down			/9.1
-V2			Lockstation1 Up			/9.2
-V3			Lockstation2 Down			/9.3
-V4			Lockstation2 UP			/9.4
-V5			Lockstation 3			/9.5
-V6			Drill valve			/9.6
-V7			Gripper inn			/9.7
-V8			Gripper out-out			/9.8
-V9			Gripper out-in			/9.8
-X1			Terminal			/2.1
SERVER						/10.2

NTNU
Norwegian University of Science and Technology

Project title:	**Production cell for processing aluminum**	**Project no.:** **Production cell**	**Project rev.:**	**Page**	**14**
Customer:	RUFO	DCC:		Scale:	1:1
Page title:	Components list	Dwg. no.:	Page rev.:	Previous page:	13
File name:	RUFOVS4	Eng. (proj/page): SGI/ SGI	Last print: 19.05.2018	Next page:	
Page ref.:		Appr. (date/init):	Last edit: 19.05.2018	Total no. of pages:	14

Appendix D

CNC code calculations

This is a overview of the generated CNC-Code. The calculation for the different lengths is first and then the calculations for the holes. The Complete CNC-Code generated is showen on the bottom with all values.The reason for the profile number 2 does not contain any cutshawing is that the saw is turned 180 degrees for the cut on the endcut, this results in that the cutshawing ends up in the next profile.

Ordered profile	Millimeter	CNC-Code length	Millimeter including CutShaving	Milimeter excluding CutShaving	CutShaving	
Profile 1	240	Profile 1	244	240,04	3,96	
Profile 2	240	Profile 2	240	240,00	0	
Profile 3	520	Profile 3	522,8	520,00	2,8	
Ordered profile	First hole	CNC-Code position	Actual position calculated from CNC-Code	Calculation		
Profile 1	20	245,3	20	CNCpos - Drilloffset -Profilewaste	Cut ofsett	390
Profile 2	30	499,3	30	CNCpos - Drilloffset -Profilewaste -Profile1	Drill offset	146
Profile 3	30	743,3	30	CNCpos - Drilloffset -Profilewaste -Profile1 -profile2	Profile waste	79,3
CNC-Code	CNC-Value					
TI	245,3					
M	7	Hole profile 1				
TI	60					
M	7	Hole profile 1				
TI	80					
M	7	Hole profile 1				
TI	40					
M	7	Hole profile 1				
TI	44					
AC	45	AngleCut profile 1				
TI	30					
M	13	Hole profile 2				
TI	40					
M	13	Hole profile 2				
TI	60					
M	13	Hole profile 2				
TI	60					
M	13	Hole profile 2				
TI	54					
AC	-45	AngleCut profile 1				
TO						
RC	45	RotationCut profile 2				
TI	30					
M	10	Hole profile 3				
TI	40					
M	10	Hole profile 3				
TI	60					
M	10	Hole profile 3				
TI	60					
M	10	Hole profile 3				
TI	50					
RC	-45	RotationCut profile 2				
TO						
TI	4					
RC	90	RotationCut profile 3				
TC	17	TR	45	TrimCut profile 3		
TI	26					
M	10	Hole profile 3				
TI	80					
M	10	Hole profile 3				
TI	50					
M	10	Hole profile 3				
TI	36,8					
TO	13,2					
M	10	Hole profile 3				
TO	44					
BC	1	BendCut profile 3				
TO	270					
TC	17	TR	-45	TrimCut profile 3		
TO	2,8					
RC	90	RotationCut profile 3				

Appendix E

Pre-project report

PRE PROJECT - REPORT

FOR BACHELOR THESIS

NTNU
Kunnskap for en bedre verden

TITEL:
Production cell for punching and cutting of aluminium profil.

CANDIDATE NUMBER(S):			
Sondre Grevle Iveland, Fredrik Ryslett and Fredrik Bakker.			
DATO: **21.01.2018**	COURSE CODE: **IE303612**	COURSE TITLE: **BACHELOR THESIS**	DOCUMENT ACCESS: - Open
STUDY PROGRAMME: **AUTOMATION ENGINEERING**		NO. PAGES/ATTACHMENT: /	LIB. NO: - Not in use -

CONTRACTOR(S)/SUPERVISOR(S):
Knut Steinnes - RUFO AS Steffen Viddal - Viddal Automation AS. Ottar L. Osen - NTNU Mikael Tollefsen - NTNU

TASK/SUMMARY:

This task is an exam answer performed by students at NTNU in Ålesund.

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Foretaksregisteret
Norway
572 140

Besøksadresse
Larsgårdsvegen 2
Internett
www.ntnu.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
NO 971

INNHOLD

1 INTRODUCTION

For our bachelor thesis we are developing an automated production cell to process aluminum profiles, it will be able to process aluminium up to 6 meters in length. The production cell will process the profiles according to CAD drawings, which gives information about the length, cutting angle, number of punching holes and distance between these holes. When the profile is processed, it will be placed into storage until an operator retrieves it for assembly. Our contractor for the thesis is Viddal Automation AS, which is a company providing automation and programming services. Viddal's client RUFO AS is the company that has hired Viddal to develop the production cell. RUFO AS specialises in developing and producing flight cases , for transportation of equipment in many different industries, such as: offshore, sound, and light etc. RUFO wants to automate its production, in order to be competitive at an international level.

2 THERMS

PLC - Programmable logic controller
CAD - Computer aided design

3 PROJECT ORGANIZATION

3.1 Project Group

Student Number(s)
460049 - Fredrik Aarset Ryslett, project leader
460306 - Fredrik Johannes Bakker, secretary
460001 - Sondre Grevle Iveland, group member

Tabel: Student number (s) for everyone in the group who delivers the task for assessment in the subject ID 302906

3.1.1 Tasks for project leader

- Responsible for the progress, and quality checks of our work
- Planning meetings with the contractor Viddal Automation AS, Client RUFO AS and Supervisors

3.1.2 Tasks for secretary

- Writing meeting reports.
- Writing progress report before bachelor thesis supervisor meetings.

3.1.3 Tasks for other members

- Help the leader and secretary

3.2 Control group

The control group consists of the supervisors: Otta L Osen and Mikael Tollefsen, contractor: Steffen Viddal and client: Knut Steinnes

4 APPOINTMENTS

4.1 Appointments with contractor

21 Nov. 09:00
meeting at the school (NTNU) with Knut from RUFO, Steffen from Viddal automation and Ottar Osen from NTNU

07 Des. 08:00
Meeting with Steffen from Viddal automation

18 Jan. 08:00
meeting at RUFO (Ørsta) with team from RUFO, Steffen from Viddal automation and possibly Ottar Osen or Mikael tollefsen from NTNU.

Since we are doing most of the work at Viddal Automation AS, our group will be in constant contact with the contractor coordinating our work.

4.2 Workplace and resources

During our bachelor work we have access to Viddal Automation's facilities and resources, since Viddal is the contractor they cover all the cost for this project, both material and hardware. In the startup phase we will use NTNU Ålesunds facilities for the report work and planning, but we will gradually move our efforts over to Viddal's facilities. At the end of the bachelor work we will implement our solution at our client RUFO's facilities. We will also have continuous collaboration with the supervisors, contractor and client.

4.3 Group norms – partnership rules – attitudes

1. Working hours
   The bachelor thesis will require a lot of work, so we agreed upon that we start working every day at 09.00 to 17.00 + . The "+" means that we will work every day as late as we need to meet our deadlines.

2. Work ethics
   All members has to have a good work ethics and treat each other with respect. Project members have the right to say their opinion, and in case of disagreement a democratic voting will be performed.

3. Meetings
   Punctuality at meetings with the control group is mandatory. If by any chance a project member cannot attend, he has to leave a notice in advance.

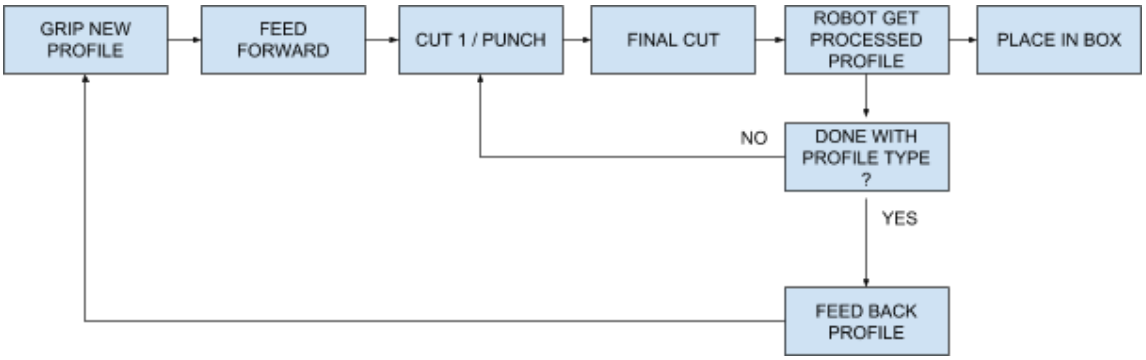
4. Report writing
   We start every Monday with a meeting discussing what we did last week, and the coming weeks plan. This includes writing on the bachelor thesis.

5 PROJECT DESCRIPTION

RUFO AS contacted Viddal Automation to order an automated magazine fed aluminium profile processor production cell. As stated in the introduction, this cell will punch holes and cut the profiles in correct angle and lengths.

Viddal Automation's job is to design an build the production cell, and integrate the magazine feeder. Our part of the project is to design the electrical system and programming the system which punches and cuts the profiles. Figure 1 below simple description about our part of the project.

Figure 1: Production cell flow chart



The desired result would be to prepare an untreated aluminum profile to the feeding table when the operator fill in what flight case he wants to build, the machine will calculate the amount of profile needed by the different types and optimize the lengths to achieve less waist . Then aluminum is fed through the processing module. In this section, all cuts and holes should be placed on the profile. After the processing the robot extracts the finished machined profile and puts it in a box on the outside.

5.1 Problem - goals - purpose

Thesis question:

Is it possible to *develop an automated control system for an aluminium profile processing production cell, by following the project specifications provided by RUFO AS ?*

Impact goal:

Delivering a fully functional product that can be integrated into RUFO's vision, which is to fully automate their production.

Return on sales:

1. Delivering a product which produces the aluminum profiles within RUFO's quality requirements
2. Delivering a product within the budget 2,500,000 kr
3. Delivering a product within the time limit

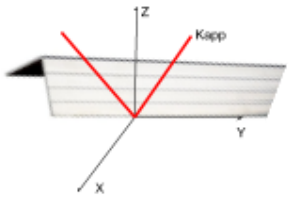
5.2 Requirements for solution - specifications

1. Precision:
    a. Precision of 1/10 mm on cutting and on the position of the holes.
    b. The holes to be machined on aluminum profiles should not have any kind of defects or spoon residue.
    c. It was not cleared how many different types of holes will be machined on the aluminum profiles. RUFO will launch a test period where we only use a size POP rivets. Type 4.8 mm, we will receive feedback as soon as the test period has been completed.

2. Production capacity
    a. Production capacity should go as quickly as possible without affecting the quality of the product.

    b. The production cells will allow processing of aluminum profiles of up to 6 meters.

3. Requirements for cutting profiles
    a. Angle list should be able to cut at any angle, from a degree of freedom as shown in the illustration below



        i. The shortest angle lists used now are 5 cm long. This is shorter than what the production cell can handle. We discussed different methods to solve this and found that RUFO either changes its production design or that these lengths are cut by hand.

    b. Guiding lists should be able to cut at any angle from all degrees of freedom.
        i. Needs radius cutting or milling to fit into corner brackets.

        ii. The guiding list must be cut in two ways
            1. Cut off at different angles.
            2. Cut so that they can be bent.

4. Reducing wastage / urgent orders
    a. The production cell will operate with optimized cutting of profiles that minimize waste product.

5. Sorting of produced profiles
    a. Make use of a KUKA robot that sorts produced profiles according to which box / order they belong to after they come out of the production cell.

    b. It is possible that this is delayed and not delivered in May due to the ordering time on the robot.

6. Deadline
    a. Mount at RUFO at the beginning of May

5.3 Development work - methods

There are several strategies to pick from when working in a project. After some research we decided that the combination of two different strategies would be the best strategy for out bachelor thesis.

The first strategy is the milestone approach. This strategy is based on planning the project based on milestones. A milestone is to split up the project goal into smaller intermediate objectives, this process is not iterative. However the process can have different branches, and change, to reach the milestone as fast as possible. This is different from the waterfall model where all tasks are supposed to be known before starting the project, and the process is streamlined. (Skyttermoen and Vaagaasar 2015)

The second strategy is the agile approach, where the process is iterated into intervals. Using this approach the vision for the finished product changes along the project development. An agile approach is often used in software development projects. When integrating the agile strategy into the project plan, and there is a not clear plan for when tasks need to be completed. (Skyttermoen and Vaagaasar 2015)

Combining these strategies is the best approach for our project because:

1. We need a linear project plan for our project so we can meet our project deadline. Where completing each main activity is a milestone to reach this deadline. This goes against the agile strategy.

2. We know that the programming and testing of the production cell will be an iterative process, which disagrees with the milestone strategy. But this is needed, because of the lack of experience our project group has with producing and industry level production cell.

5.4 Information collection - done and planned

One of our group members had an internship at RUFO in the autumn of 2017, which formed the origin of the bachelor thesis. In this internship, the production process at RUFO has been analyzed, with a lot of focus on aluminum processing.

RUFO has been offered a machine that processes aluminum from a German company specializing in cutting aluminum. Viddal automation has promised that they can deliver a production cell that is at least as effective at the same price. The cell should be developed with a provident view to further automate the production line.

Concept drawings in 3D of the production cell have been developed in collaboration with RUFO and Viddal Automation A/S, which forms the basis of the cell we build in our bachelor assignment. The concept drawings in 3D are developed in sketchup. Documentation of how many meters aluminum RUFO uses in one day, how many angles and type of holes profiles need to be processed and the weight and length of aluminium profiles in a magazine. This documentation is provided in an excel document. RUFO's workers are also given a form

where they noted how much time they spend on drilling and cutting aluminum profiles. We will have access to this information as soon as the results are available.

5.5 Evaluation - Risk analysis

In the process of completing a project there is a lot of different risks involved. These risks can cause the project to be delayed, or worst case scenario the project is not feasible. To minimize the chance for unwanted events happening there will be conducted a risk analysis, and a risk reduction to minimize the risks or eliminate them. Good planning is essential in this process.(Skyttermoen and Vaagaasar 2015)

Figure 2: Before risk reduction

Likelihood	Consequence		
	Low	Medium	High
High	1. Planing		
Medium		1. Sickness	1. Computer failure
Low		1. Communication	1. Collaborating 2. Contract

- Lack of communication within the group.
- Bad planning of work. This will most likely happen.
- Missing collaboration with Viddal and the leading group. To ensure the project is of high quality, the help we get from Viddal is essential.
- Sickness of group members.
- Loss of contract between Viddal and RUFO. If RUFO does not sign the contract with Viddal for making the machine.
- Computer failure.

Figure 3: After risk reduction

Likelihood	Consequence		
	Low	Medium	High
High	Planing		
Medium	Computer failure	Sickness	
Low	Contract	Communication Collaborating	

- Communication is minimized with have a morning meeting, where we discuss the project progress and what to do to complete the next task.
- Bad planning of work. update the gantt diagram and add tasks as they arise.
- Missing collaboration with Viddal and the leading group. To be sure the project is of high quality, the help we get from Viddal is essential. The leader is responsible for this.
- Sickness of group members. will most likely happen, but so long as we have good communication, it shouldn't be a problem. If a group member is sick for more than a week we will have a meeting together with the leading group to see if we need to do any changes.
- Loss of contract, this will be signed if not we do not have a bachelor thesis at all.
- Computer failure. To minimize the consequence of this we are using google disk to store all documents, and using online programs to write. The consequence is therefore minimal.

5.6 Main activities

1. Electrical design:

2. Acquiring electrical parts:

3. Programming:
   - Plan programming setup
   - system programming (PLC)
   - HMI/Master controller

4. Testing

5. Quality control

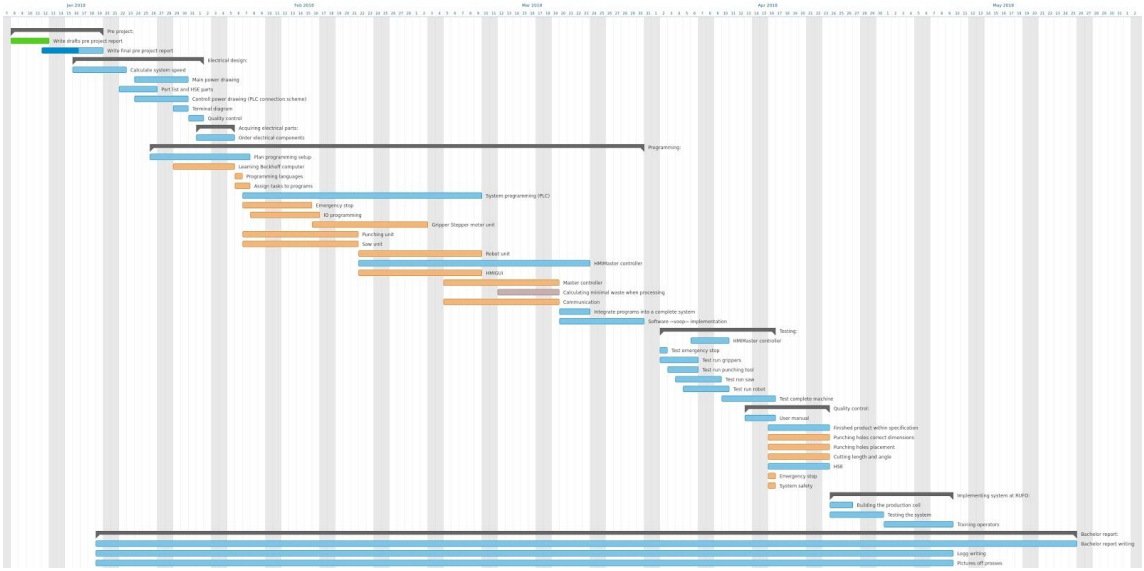
6. Implementing system at RUFO

7. Bachelor report

5.7 Progress plan - management of the project

To plan and manage the project we are using Asana and Instagantt as seen in Figure 4 below. For a more detailed description read 5.7.1 main plan.

Figure 4: Instagant



5.7.1 Main plan

Chapter 5.6 shows the main plan with the main activity/task headlines in the project, all the activities are broken down into smaller sub tasks. By breaking the activities down into smaller tasks it is easier to follow the progress and hit milestones. Even though we have a linear project plan, developed using Asana and Instagantt, the implementation will be an iterative process. This means, that we may circle back to an earlier date and do changes to already finished work because of changes done in another activity. Working in a project with an iterative approach is a often used process to ensure quality in the finished project. This method is backed up and explained in more detail by Torgeir Skyttermoen in the book "Verdiskapende prosjektledelse".
(Skyttermoen and Vaagaasar 2015, p.25)

The first task is to do the electrical design, this includes all the electrical drawings, documentation and parts we need to complete the project. This task is followed up by the next task acquiring electrical parts which is to order and acquire the parts needed.

The next set of activities is the tasks we most likely think are going to be an iterative process. This is the programming, testing and quality control. The programming is broken down into smaller programming tasks, which sums up to all the programming needed for the production cell to operate. To test our program we have the "Testing" task, during this part of the project we most likely have to go back and do changes to the programming to make it operate as desired. When the tests provides desirable results we move on the quality control, for this part we make sure that the production cell delivers finished aluminium profiles which meets

RUFO's quality requirements. During this task we may also need to do subtle changes to the programming, and run tests again until RUFO is satisfied with the quality. The final task is to implement the production cell at RUFO, and to train/instruct the operators on how to operate the system

During our work with the project we will always work on the bachelor thesis report parallel to the other activities listed above, to ensure that we write down our progress while we work.

5.7.2 Management Tools

To manage the project and our progress we have different tools at our disposal.

1. Asana / Instagantt
    a. Is used to monitor our project progress, and to divide tasks among the group members.

2. Google drive
    a. Is used to store all the project information, so it is easily accessible to everyone included in the project.

3. SharelateX
    a. Is used to write our bachelor thesis.

4. Excel/Google Excel
    a. We use Excel for multiple tasks, to calculate and document different parts of our project. But also to document the hours and minutes of each members work.

5.8 Decision-making process

The origin of this bachelor thesis is based on the work one of the group members did during his internship, therefor the thesis is self made but approved by NTNU. This means that at first the project description was very vague and large. To delineate the thesis we had a meeting with the leader group including the supervisor and contractors. Based on this meeting we delineated the project to the project description in chapter 5.

For the decision making process during our thesis work, we will make decisions based on discussions in our group. If the decision is crucial for the design, quality or deadlines, we will arrange meetings and consult with the leader group.

6 DOCUMENTATION

6.1 Reports and technical documents

Documents we need to develop during the bachelor thesis:

1. Electrical design
    a. All the electrical drawings needed to run the production cell
2. Maintenance manual
3. User manual
4. Spare part list

7 PLANNING MEETINGS AND REPORTS

7.1 MEETINGS

Meetings with the control group is planned to be held every other week. This meeting is manly for the supervisors and bachelor group, but RUFO's representative and Steffen Viddal. From Viddal Automation A/S is also invited to these meetings. At these meetings we will focus on the project progress and orientation regarding the bachelor thesis. We will also be in constant communication with Viddal Automation and RUFO, through meetings and informal communication through telefon.

7.2 Periodic reports

During the project we will handle periodic reports with different tools at our disposal. We have a mandatory logg writing after each work day for each group member, we also logg workday hours using an excel diagram. Before we start working each day we have a small morning meeting talking about the project progress. Including what's finished, and whats on the respective days agenda.The milestones are reflected in the Instagantt represented as each main task. When a main task is completed we have reached a milestone in the project.

8 PLANNED HANDLING ANOMALIES

In case the project is not evolving as planned there will be held a meeting with the control group to discuss what can be done to ensure the product will be finished in time. The meeting will discuss how to make the project smaller and then Viddal will need to help with finishing the project.

9 EQUIPMENT / REQUIREMENTS FOR IMPLEMENTATION?

Equipment and software requirements for this project will be provided by Viddal Automation AS.

10 REFERENCES

1. Skyttermoen, T. and Vaagaasar A. (2015). *Verdiskapende Prosjektledelse*. Oslo: Cappelen Damm

Appendix F

Supervisor meeting 23.11.2017

Møtereferat

Møtereferat ansvarlig:
Fredrik Bakker

Dato og Sted:
23.11.17 Møte ble holdt ved NTNU Ålesund på B433

Til stede:
Bachelorgruppe: Fredrik Bakker, Sondre Iveland og Fredrik Ryslett
NTNU veileder: Ottar L. Osem
RUFO: Knut Steinnes
Viddal Automation: Steffen Viddal

Rammeverk:
- XML
- Person lesser på Profil
- Profil blir prosessert i forhold til XML
- Ferdig profiler blir håndtert av personell

Oppdragsgiver:
- Steffen Viddal (Viddal Automation AS)

Lage Spesifikasjoner:
- Budsjett
- Fremdriftsplan
- Prosjekt spesifikasjoner

Tidsplan:

1. Januar:
Konsept, Busjett

1. Februar:
Ferdig Design layout av Viddal Automation AS (smågodt)

Appendix G

Supervisor meeting at RUFO 19.01.2018

Møtereferat

Møtereferat ansvarlig:
Fredrik Bakker

Dato og Sted:
Møtet ble holdt hos RUFO AS den 19.01.2018

Til stede:
Bachelorgruppe: Fredrik Bakker, Sondre Iveland og Fredrik Ryslett
NTNU veileder: Mikael Tollefsen
Viddal Automation AS: Steffen Viddal
RUFO AS: Knut Steinnes og et utvalg med medarbeidere fra ulike avdelinger i RUFO.

Ikke tilstede:
NTNU veileder: Ottar L. Osen

Møte:

Møtes hovedtema var hvilke spesifikasjoner aluminiums produksjons cella skal oppfylle, og hvordan arbeidsfordelingen skal være mellom Viddal Automasjon og Bachelorgruppen. Vi ble enige om at produksjons cellen som leveres til RUFO i starten av Mai ikke inneholder et integrert aluminiums magasin for foring og automatisk lager til produksjons cella. Dette er noe Viddal skal integrere senere i prosessen.

RUFO Kontaktperson

Under møte informerte Knut Steinnes at Arnfinn Velle er vår kontaktperson i RUFO for gjennomføringen av prosjektet.

Arbeidsfordelingen:

Viddal Automation er ansvarlig for design og konstruksjon av produksjons cellen, samt utvikling av softwaren som henter ut aluminiums informasjonen fra CAD tegningene. Bachelorgruppen sitt ansvar er å utvikle styresystemet for produksjons cellen.

Kravspesifikasjon:

Møtets formål var å utvikle kravspesifikasjonen for produksjonscellen Viddal Automasjon skal levere. Dette ble gjort ved at RUFO kom med sine ønsker basert på kompetansen og erfaringene til sine medarbeidere. Nedenfor står kravspesifikasjonen for produksjons cellen slik vi ble enige om iløpet av møte.

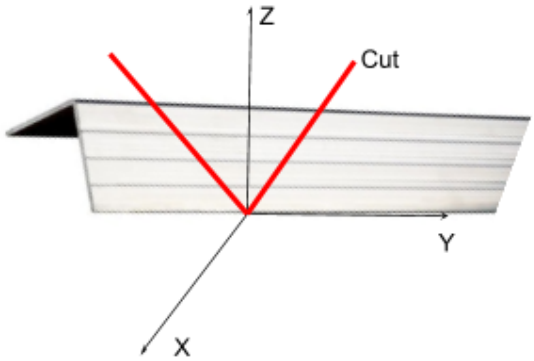
1. Nøyaktighet
    a. Nøyaktigheten på kapping og popnagel hull skal være innenfor $\frac{1}{10}$ mm.
    b. Hvis hullene skal stanses, skal det ikke være graderinger på innsiden av
    c. hullene.
    d. Ved dimensjonering av hullene var de litt uenighet. Så RUFO skal gjennomføre test monteringer ved å benytte seg av bare en type popnagler på 4.8 mm, og gi tilbakemelding på dette.

2. Produksjonskapasitet
    a. Takt tiden på produksjons cellen skal ikke gå utover kvaliteten på produktet. Så kravet er at cellen skal arbeide så fort som mulig, uten at det går på bekostning av kvaliteten.

    b. Produksjonscellen skal kunne prosessere aluminium profiler på opp til 6 meter.

3. Krav til kapping av profiler

a. Vinkellist skal kunne kappes i valgfri vinkel, fra en frihetsgrad som vist i illustrasjonen under.



i. De korteste vinkel listene som brukes nå er 5 cm lange. Dette er kortere enn hva produksjons cellen kan håndtere. Vi diskuterte ulike metoder for å løse dette, og kom frem til at RUFO enten endrer sine produksjonsdesign, eller at disse lengdene blir kappet for hånd.

b. Styrelistene skal kunne kappes i valgfri vinkel fra alle frihetsgradene.
   i. Trenger radius kapping eller fresing slik at de passer inn i hjørne beslag.

   ii. Styrelistene skal kunne kappes på to måter
      1. Kappes rett av i forskjellig vinkler.
      2. Kappes slik at de kan bøyes.

4. Redusering av svinn / hastebestillinger
   a. Produksjons cellen skal operere slik at den optimaliserer kappingen av profilene for at det blir minst mulig svinn.

5. Sortering av produserte profiler
   a. Benytte seg av en kuka robot som sorterer produserte profiler i henhold til hvilken kasse/ordre de hører til etter de kommer ut av produksjons cellen.

    b. Det mulig at dette blir forsinket og ikke levert i Mai på grunn av bestillingstiden på roboten.

6. Deadline
    a. Montere hos RUFO i starten av Mai

Appendix H

Supervisor meeting 16.02.2018

Møtereferat

Møtereferat ansvarlig:
Fredrik Bakker

Dato og Sted:
16.02.18 Møte ble holdt ved NTNU Ålesund på F420

Til stede:
Bachelorgruppe: Fredrik Bakker, Sondre Iveland og Fredrik Ryslett
NTNU veileder: Mikael Tollefsen
NTNU veileder: Ottar L. Osen

Møte:
Møte notatene jeg tok har desverre blitt slettet. Så dette møtereferatet blir skrevet fra hukommelsen.

Tema: Veiledning
Møtet omhandlet hva vi har gjort de siste 2 ukene og hva planen er for de neste 2 ukene. Vi pratet også om ansvarsområdene og hva planen er hvis maskinen ikke blir produsert av viddal. Dette var da avgrense oppgaven til simulering av produksjons cella. Tok også opp stepper motoren og CE godkjenning, fikk også veiledning når dete gjelder kommunikasjonsprotokoller mellom Server, pls og kuka, pls.

Mål for de neste 2 ukene:
Fredrik Bakker:
Ferdigstille server versjon 1 . Få opp kommunikasjon med PLS

Sondre Iveland:
Programmere Kuka roboten. Gjøre kommunikasjonen mellom kuka og PLS ferdig.

Fredrik Ryslett:
Sette opp grunn strukturen i codesys. Sette opp kommunikasjon med Kuka robot.
Kommunikasjon til server fra pls. Stepper motor styring.

Appendix I

Server source code

```
using RUFOServer.Program.Exceptions;
using RUFOServer.Program.Interface;
using RUFOServer.Program.Main;
using RUFOServer.Program.Register;
using RUFOServer.Program.Server;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.Globalization;
using System.Linq;
using System.Threading;
using System.Windows.Forms;
using TwinCAT.Ads;

namespace RUFOServer.Main
{
    /// <summary>
    /// The is the main GUI window,
    /// And is used to launch and control the server software.
    /// </summary>
    public partial class RUFO : Form, GUIHandler
    {
        // Buisness logic objects/variables
        private AluProfileOrderList angleOrderList;
        private AluProfileOrderList guidingOrderList;
        private ADSCommunication ADSCom;
        private OrderHandler orderHandler;
        private CutShaving cutShaving;
        private ConfigLoader configFile;

        // Connection variables
        private string netId = "5.52.27.210.1.1";
        int port = 851;

        // Aluminium profile variables
        private AluminiumProfile angleProfile;
        private AluminiumProfile guidingProfile;
        private AluminiumProfile angleProfileForProduction;
        private AluminiumProfile guidingProfileForProduction;

        // Index variables
        private bool emcyListening = true;
        private bool reset = false;

        // Format variables
        // Ruleset for parsing double values from the string
        private NumberFormatInfo numberFormat;


        /// <summary>
        /// Launching the GUI
        /// </summary>
        public RUFO()
        {
            InitializeComponent();
```

```
57
58              // Load files
59              this.configFile = new ConfigLoader();
60
61              // Generating objects
62              this.cutShaving = new CutShaving(configFile.getValue("Blade"));
63              this.angleOrderList = new AluProfileOrderList();
64              this.guidingOrderList = new AluProfileOrderList();
65
66              this.orderHandler = new OrderHandler(this.angleOrderList,
                  this.guidingOrderList, this.cutShaving, this);
67              // Setting the length of the aluminium profiles
68              // Angle profile
69              this.orderHandler.setProfileLengthLineOne
                  (this.configFile.getValue("ProfileLengthOne"));
70              // Guiding profile
71              this.orderHandler.setProfileLengthLineTwo
                  (this.configFile.getValue("ProfileLengthTwo"));
72              // Start the orderHandler which checks the folder for orders
73              this.orderHandler.start();
74              // Communication
75              this.ADSCom = new ADSCommunication(netId, port);
76
77              // Formating
78              this.numberFormat = new NumberFormatInfo();
79              this.numberFormat.NumberDecimalSeparator = ".";
80              this.numberFormat.NegativeSign = "-";
81
82              /// <summary>
83              // Initializing the GUI elements
84              this.setupDataGrids();
85              this.setUpSwitchBtns();
86              this.setupButtons();
87              this.setupErrorLog();
88
89              if((this.angleOrderList.getOrderList().Any()) ||
                  (this.guidingOrderList.getOrderList().Any()))
90              {
91                  this.guidingOrderProductionList();
92                  this.angleOrderProductionList();
93              }
94
95              // Setup communication
96              connectToPlc();
97              plcErrorMonitor();
98          }
99
100         /// <summary>
101         /// Start error monitor thread
102         /// </summary>
103         private void plcErrorMonitor()
104         {
105             Thread EMCY = new Thread(new ThreadStart(monitor));
106             EMCY.IsBackground = true;
107             EMCY.Start();
108         }
```

```
109
110
111         /// <summary>
112         /// Main method to launch the program
113         /// </summary>
114         public static void Main()
115         {
116             Application.EnableVisualStyles();
117             Application.Run(new RUFO());
118
119         }
120
121         /// <summary>
122         /// Enable connection between server and PLC
123         /// </summary>
124         private void connectToPlc()
125         {
126             try
127             {
128                 //Enable start button
129                 this.startBtn.Enabled = true;
130                 this.ADSCom.connect();
131
132                 if (!this.ADSCom.readADSState().Equals("Run"))
133                 {
134                     this.ADSCom.setState(AdsState.Run);
135                 }
136             }
137             catch (AdsException e)
138             {
139                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
140                 String cpt = "KOMMUNIKASJONSFEIL";
141                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.RetryCancel, MessageBoxIcon.Error);
142
143                 // Try to connect again
144                 if(result == DialogResult.Retry)
145                 {
146                     this.connectToPlc();
147                 }
148
149                 // if cancel, shut down application
150                 else if(result == DialogResult.Cancel)
151                 {
152                     Environment.Exit(0);
153                 }
154             }
155
156             catch (Exception err)
157             {
158                 String txt = "Noe gikk feil" + "\n\n" + err.Message;
159                 String cpt = "NOE GIKK FEIL";
160                 MessageBox.Show(txt, cpt);
161             }
162         }
```

```
163
164         /// <summary>
165         /// Initializing the buttons
166         /// </summary>
167         private void setupButtons()
168         {
169             this.restartBtn.Enabled = false;
170             this.stopBtn.Enabled = false;
171             this.produceAnglebtn.Enabled = false;
172             this.produceGuidingBtn.Enabled = false;
173
174             this.reset_Stpr.Visible = false;
175         }
176
177         /// <summary>
178         /// Setting up the switch buttons over each gridview to it's standard
               setup
179         /// </summary>
180         private void setUpSwitchBtns()
181         {
182             this.switchBtn.Text = "PRODUKSJON";
183             this.switchBtnTwo.Text = "PRODUKSJON";
184         }
185
186         /// <summary>
187         /// Initialize the error log text box
188         /// </summary>
189         private void setupErrorLog()
190         {
191             this.errorLog.ReadOnly = true;
192             this.addErrorToLog( "Feilmeldings logg", Color.Black);
193         }
194
195
196         /// <summary>
197         /// Adding an error to the error logg.
198         /// </summary>
199         /// <param name="error">
200         /// The string containing the error message.
201         /// </param>
202         /// <param name="color">
203         /// what text color the error shall have.
204         /// </param>
205         private void addErrorToLog(string error, Color color)
206         {
207             this.errorLog.SelectionStart = this.errorLog.TextLength;
208             this.errorLog.SelectionLength = 0;
209             this.errorLog.SelectionColor = color;
210
211             string dateAndTime = DateTime.Today.ToShortDateString() + " " +
                  DateTime.UtcNow.ToLocalTime().ToShortTimeString();
212             this.errorLog.AppendText("-> " + dateAndTime + ":\n"  + error +
                  "\n");
213             this.errorLog.SelectionColor = this.errorLog.ForeColor;
214         }
215
```

```
216         /// <summary>
217         /// Setting up all the gridviews to it's standard setup for this GUI
218         /// </summary>
219         private void setupDataGrids()
220         {
221             angleOrderProfilesGrid();
222             guidingOrderProfilesGrid();
223             profilesForProductionGrid();
224         }
225 
226         /// <summary>
227         /// Setting up the top right gridview containing the profiles for
              production to it's standard setup for this GUI
228         /// </summary>
229         private void profilesForProductionGrid()
230         {
231             // Top right gridview containing the profiles for production
232             this.productionGridLabel.Text = "PROFILER I PRODUKSJON";
233             this.productionGrid.ColumnCount = 3;
234             this.productionGrid.ReadOnly = true;
235             this.productionGrid.MultiSelect = false;
236             this.productionGrid.RowHeadersVisible = false;
237             this.productionGrid.Columns[0].Name = "Profiltype";
238             this.productionGrid.Columns[0].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
239             this.productionGrid.Columns[1].Name = "Profil lengde";
240             this.productionGrid.Columns[1].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
241             this.productionGrid.Columns[2].Name = "Profil rest";
242             this.productionGrid.Columns[2].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
243             this.productionGrid.RowTemplate.Height = 65;
244             this.productionGrid.AutoSizeColumnsMode =
                  DataGridViewAutoSizeColumnsMode.Fill;
245             this.productionGrid.ScrollBars = ScrollBars.None;
246         }
247 
248         /// <summary>
249         /// Setting up the bottom left gridview containing the ordered angled
              profiles to it's standard setup for this GUI
250         /// </summary>
251         private void guidingOrderProfilesGrid()
252         {
253             // Right gridview containing the guiding profiles
254             this.guidingOrder.ColumnCount = 3;
255             this.guidingOrder.ReadOnly = true;
256             this.guidingOrder.MultiSelect = false;
257             this.guidingOrder.RowHeadersVisible = false;
258             this.guidingOrder.Columns[0].Name = "Profiltype";
259             this.guidingOrder.Columns[0].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
260             this.guidingOrder.Columns[1].Name = "Profil lengde";
261             this.guidingOrder.Columns[1].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
262             this.guidingOrder.Columns[2].Name = "Profil rest";
263             this.guidingOrder.Columns[2].SortMode =
```

```
                DataGridViewColumnSortMode.NotSortable;
264             this.guidingOrder.RowTemplate.Height = 40;
265             this.guidingOrder.AutoSizeColumnsMode =
                  DataGridViewAutoSizeColumnsMode.Fill;
266         }
267
268         /// <summary>
269         /// Filling the guidingOrder gridview with the orders from the
              OrderHandler class
270         /// </summary>
271         private void guidingOrderProductionList()
272         {
273             this.guidingOrder.Rows.Clear();
274             this.guidingOrder.Columns[0].Name = "Profiltype";
275             this.guidingOrder.Columns[1].Name = "Profil lengde";
276             this.guidingOrder.Columns[2].Name = "Profil rest";
277             this.guidingOrder.DefaultCellStyle.SelectionBackColor =
                  SystemColors.Highlight;
278             this.guidingOrder.DefaultCellStyle.SelectionForeColor =
                  SystemColors.HighlightText;
279
280             foreach (AluminiumProfile p in this.guidingOrderList.getOrderList
                  ())
281             {
282                 String[] data = new String[] { p.getType(), p.getFullLength
                      ().ToString(this.numberFormat), p.getWaste().ToString
                      (this.numberFormat) };
283                 this.guidingOrder.Rows.Add(data);
284             }
285
286             this.guidingOrder.ClearSelection();
287         }
288
289         /// <summary>
290         /// Rearranging the bottom left gridview guidingOrder to show the
              profiles which are being produced from the standard profile
291         /// </summary>
292         private void guidingProductionList()
293         {
294             this.guidingOrder.Rows.Clear();
295
296             this.guidingOrder.Columns[0].Name = "Produksjons ordre nummer";
297             this.guidingOrder.Columns[1].Name = "Produkt nummer";
298             this.guidingOrder.Columns[2].Name = "Profil lengde";
299             this.guidingOrder.DefaultCellStyle.SelectionBackColor =
                  Color.Transparent;
300             this.guidingOrder.DefaultCellStyle.SelectionForeColor =
                  Color.Black;
301
302             int i = 0;
303
304             if (this.guidingProfileForProduction != null)
305             {
306                 foreach (OrderedAluProfile p in
                      this.guidingProfileForProduction.getSortedAluList().getList
                      ())
```

```
307                 {
308                     String[] data = new String[] { p.getOrderNmb(),
                    p.getProdNmb(), p.getLength().ToString
                    (this.numberFormat) };
309                     this.guidingOrder.Rows.Add(data);
310
311                     if (p.isProduced())
312                     {
313                         // Set Bakcground color for a spesific row
314                         this.guidingOrder.Rows.SharedRow
                    (i).DefaultCellStyle.BackColor = Color.Green;
315                     }
316                     else
317                     {
318                         // Set Bakcground color for a spesific row
319                         this.guidingOrder.Rows.SharedRow
                    (i).DefaultCellStyle.BackColor = Color.Red;
320                     }
321
322                     i++;
323                 }
324
325                 this.guidingOrder.Rows[0].Selected = false;
326             }
327         }
328
329         /// <summary>
330         /// Setting up the bottom right gridview containing the ordered
              guiding profiles to it's standard setup for this GUI
331         /// </summary>
332         private void angleOrderProfilesGrid()
333         {
334             // Left gridview containing the angle profiles
335             this.AngleOrder.ColumnCount = 3;
336             this.AngleOrder.ReadOnly = true;
337             this.AngleOrder.MultiSelect = false;
338             this.AngleOrder.RowHeadersVisible = false;
339             this.AngleOrder.Columns[0].Name = "Profiltype";
340             this.AngleOrder.Columns[0].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
341             this.AngleOrder.Columns[1].Name = "Profil lengde";
342             this.AngleOrder.Columns[1].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
343             this.AngleOrder.Columns[2].Name = "Profil rest";
344             this.AngleOrder.Columns[2].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
345             this.AngleOrder.RowTemplate.Height = 40;
346             this.AngleOrder.AutoSizeColumnsMode =
                  DataGridViewAutoSizeColumnsMode.Fill;
347         }
348
349         /// <summary>
350         /// Filling the AngleOrder gridview with the orders from the
              OrderHandler class
351         /// </summary>
352         private void angleOrderProductionList()
```

```
353         {
354             this.AngleOrder.Rows.Clear();
355             this.AngleOrder.Columns[0].Name = "Profiltype";
356             this.AngleOrder.Columns[1].Name = "Profil lengde";
357             this.AngleOrder.Columns[2].Name = "Profil rest";
358             this.AngleOrder.DefaultCellStyle.SelectionBackColor =
                  SystemColors.Highlight;
359             this.AngleOrder.DefaultCellStyle.SelectionForeColor =
                  SystemColors.HighlightText;
360 
361             foreach (AluminiumProfile p in this.angleOrderList.getOrderList
                  ())
362             {
363                 String[] data = new String[] { p.getType(), p.getFullLength
                      ().ToString(this.numberFormat), p.getWaste().ToString
                      (this.numberFormat) };
364                 this.AngleOrder.Rows.Add(data);
365             }
366             this.AngleOrder.ClearSelection();
367         }
368 
369         /// <summary>
370         /// Rearranging the bottom left gridview AngleOrder to show the
              profiles which are being produced from the standard profile
371         /// </summary>
372         private void angleProductionList()
373         {
374             this.AngleOrder.Rows.Clear();
375             this.AngleOrder.Columns[0].Name = "Ordre nummer";
376             this.AngleOrder.Columns[1].Name = "Produkt nummer";
377             this.AngleOrder.Columns[2].Name = "Profil lengde";
378             this.AngleOrder.DefaultCellStyle.SelectionBackColor =
                  Color.Transparent;
379             this.AngleOrder.DefaultCellStyle.SelectionForeColor =
                  Color.Black;
380 
381             if (this.angleProfileForProduction != null)
382             {
383                 int i = 0;
384                 foreach (OrderedAluProfile p in
                      this.angleProfileForProduction.getSortedAluList().getList
                      ())
385                 {
386                     String[] data = new String[] { p.getOrderNmb(),
                          p.getProdNmb(), p.getLength().ToString
                          (this.numberFormat)};
387                     this.AngleOrder.Rows.Add(data);
388 
389                     if(p.isProduced())
390                     {
391                         // Set Bakcground color for a spesific row
392                         this.AngleOrder.Rows.SharedRow
                          (i).DefaultCellStyle.BackColor = Color.Green;
393                     }
394                     else
395                     {
```

```
396                         // Set Bakcground color for a spesific row
397                         this.AngleOrder.Rows.SharedRow
                        (i).DefaultCellStyle.BackColor = Color.Red;
398                         }
399
400                         i++;
401                     }
402                     this.AngleOrder.Rows[0].Selected = false;
403                 }
404
405             }
406
407
408             /// <summary>
409             /// Eventhandler for the angleOrder gridveiw (The bottom left
                  gridview)
410             /// </summary>
411             /// <param name="sender"></param>
412             /// <param name="e"></param>
413             private void AngleOrder_CellContentClick(object sender,
                  DataGridViewCellEventArgs e)
414             {
415                 // Making sure we only select items when wathing the order list
                      and not production list
416                 if(this.switchBtn.Text.Equals("PRODUKSJON"))
417                 {
418                     // Making sure the row clicked is inside the index of the
                          list
419                     if ((e.RowIndex >= 0) && (e.RowIndex <=
                          this.angleOrderList.getOrderList().Count()))
420                     {
421                         this.angleProfile = this.angleOrderList.getOrderList
                          ().ElementAt(e.RowIndex);
422                     }
423                     else
424                     {
425                         this.angleProfile = null;
426                     }
427                 }
428
429             }
430
431             /// <summary>
432             /// Showwing the DetailsDialog with the details about the chosen
                  Angle profile
433             /// </summary>
434             /// <param name="sender"></param>
435             /// <param name="e"></param>
436             private void angleAdvBtn_Click(object sender, EventArgs e)
437             {
438                 if(this.angleProfile != null)
439                 {
440                     DetailsDialog dialog = new DetailsDialog(this.angleProfile);
441                     dialog.Show();
442
443                 } else
```

```
444             {
445                 string msg = "Ingen element er valgt: \nVelg en profil fra
                      listen for å se detaljene";
446                 string caption = "Informasjon";
447                 MessageBox.Show(msg, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Information);
448 
449             }
450 
451         }
452 
453         /// <summary>
454         /// Chosing an element from the guidingOrder GridView
455         /// </summary>
456         /// <param name="sender"></param>
457         /// <param name="e"></param>
458         private void guidingOrder_CellContentClick(object sender,
              DataGridViewCellEventArgs e)
459         {
460 
461             // Making sure we only select items when watching the order list
                  and not production list
462             if (this.switchBtnTwo.Text.Equals("PRODUKSJON"))
463             {
464                 // Making sure the row clicked is inside the index of the
                      list
465                 if ((e.RowIndex >= 0) && (e.RowIndex <=
                      this.guidingOrderList.getOrderList().Count()))
466                 {
467                     this.guidingProfile = this.guidingOrderList.getOrderList
                      ().ElementAt(e.RowIndex);
468                 }
469 
470                 else
471                 {
472                     this.guidingProfile = null;
473                 }
474             }
475 
476         }
477 
478         /// <summary>
479         /// Restarting the production lines.
480         /// </summary>
481         /// <param name="sender"></param>
482         /// <param name="e"></param>
483         private void restartBtn_Click(object sender, EventArgs e)
484         {
485 
486             try
487             {
488 
489                 String txt = "Er du sikker på at du vil starte på nytt ?";
490                 String cpt = "RESET";
491                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.YesNo, MessageBoxIcon.Information);
```

```
492
493                 if(result == DialogResult.Yes)
494                 {
495                     String reset = "GVL.ResetL2";
496                     this.ADSCom.writeToBoolean(reset, true);
497
498                     // Clear production
499                     this.angleProfileForProduction = null;
500                     this.guidingProfileForProduction = null;
501                     this.angleProfile = null;
502                     this.guidingProfileForProduction = null;
503
504                     this.productionGrid.Rows.Clear();
505
506                     this.switchBtnTwo.Text = "PRODUKSJON";
507                     this.guidingOrderProductionList();
508                     this.switchBtn.Text = "PRODUKSJON";
509                     this.angleOrderProductionList();
510
511                     // Enable manuel mode
512                     this.manuelModeBtn.Enabled = true;
513                     // Disable buttons
514                     this.restartBtn.Enabled = false;
515                 }
516
517             }
518             catch (AdsException err)
519             {
520                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + err.Message;
521                 String cpt = "KOMMUNIKASJONSFEIL";
522                 MessageBox.Show(txt, cpt, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
523             }
524
525
526         }
527
528         /// <summary>
529         /// Starting the PLC program
530         /// </summary>
531         /// <param name="sender"></param>
532         /// <param name="e"></param>
533         private void startBtn_Click(object sender, EventArgs e)
534         {
535             this.reset = false;
536
537             String stop = "GVL.StopL2";
538             String start = "GVL.Start";
539             try
540             {
541                 // Enable buttons
542                 this.stopBtn.Enabled = true;
543                 this.produceAnglebtn.Enabled = true;
544                 this.produceGuidingBtn.Enabled = true;
545                 // Disable buttons
```

```
546                 this.startBtn.Enabled = false;
547                 this.restartBtn.Enabled = false;
548 
549                 this.ADSCom.writeToBoolean(start, true);
550                 this.ADSCom.writeToBoolean(stop, false);
551             }
552             catch (AdsException err)
553             {
554                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + err.Message;
555                 String cpt = "KOMMUNIKASJONSFEIL";
556                 MessageBox.Show(txt, cpt, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
557             }
558 
559         }
560 
561         /// <summary>
562         /// Stopping the PLC program
563         /// </summary>
564         /// <param name="sender"></param>
565         /// <param name="e"></param>
566         private void stopBtn_Click(object sender, EventArgs e)
567         {
568             String stopL2 = "GVL.StopL2";
569             String start = "GVL.Start";
570 
571             try
572             {
573                 // Enable buttons
574                 this.startBtn.Enabled = true;
575                 this.restartBtn.Enabled = true;
576                 //Disable buttons
577                 this.stopBtn.Enabled = false;
578                 this.produceAnglebtn.Enabled = false;
579                 this.produceGuidingBtn.Enabled = false;
580 
581                 this.ADSCom.writeToBoolean(start, false);
582                 this.ADSCom.writeToBoolean(stopL2, true);
583 
584             }
585             catch (AdsException err)
586             {
587                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + err.Message;
588                 String cpt = "KOMMUNIKASJONSFEIL";
589                 MessageBox.Show(txt, cpt, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
590             }
591 
592         }
593 
594         /// <summary>
595         /// Handling the "Detaljer" button event
596         /// </summary>
597         /// <param name="sender"></param>
```

```
598         /// <param name="e"></param>
599         private void guidingAdvBtn_Click(object sender, EventArgs e)
600         {
601             if (this.guidingProfile != null)
602             {
603                 DetailsDialog dialog = new DetailsDialog
                      (this.guidingProfile);
604                 dialog.Show();
605             }
606             else
607             {
608                 string msg = "Ingen element er valgt: \nVelg en profil fra
                      listen for å se detaljene";
609                 string caption = "Informasjon";
610                 MessageBox.Show(msg, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Information);
611             }
612         }
613
614         /// <summary>
615         /// Starting the production of the chosen angle profile
616         /// </summary>
617         /// <param name="sender"></param>
618         /// <param name="e"></param>
619         private void produceAnglebtn_Click(object sender, EventArgs e)
620         {
621             if ((this.angleProfile != null) &&
                    (this.angleProfileForProduction == null))
622             {
623
624                 // Disabel manuel mode
625                 this.manuelModeBtn.Enabled = false;
626
627                 this.angleProfileForProduction = this.angleProfile;
628                 this.angleProfile = null;
629                 this.switchBtn.Text = "ORDRE";
630
631                 string msg = "Er profiltype " +
                      this.angleProfileForProduction.getType() + "  av lengde "
632                       + this.angleProfileForProduction.getFullLength() + "mm
                       plassert på produksjonslinje 1 ?";
633                 string caption = "ADVARSEL";
634
635                 DialogResult result = MessageBox.Show(msg, caption,
                      MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
636
637                 if(result == DialogResult.Yes)
638                 {
639                     this.angleProductionList();
640                     this.addAngleProfileToProductionGrid();
641
642                     this.startProductionLineOne();
643                 }
644                 else
645                 {
646                     this.angleProfileForProduction = null;
```

```
647                     this.switchBtn.Text = "PRODUKSJON";
648                 }
649 
650             }
651 
652             // A profile is already in production
653             else if (this.angleProfileForProduction != null)
654             {
655                 string msg = "En profil av type: " +
                      this.angleProfileForProduction.getType() + " Er allerede
                      under produksjon";
656                 string caption = "Informasjon";
657                 MessageBox.Show(msg, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Information);
658             }
659 
660             // No element from the list is chosen
661             else
662             {
663                 string msg = "Ingen element er valgt: \nVelg en profil fra
                      listen før du setter igang produksjonen";
664                 string caption = "Informasjon";
665                 MessageBox.Show(msg, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Information);
666 
667             }
668         }
669 
670         /// <summary>
671         /// Starting the production on production line 1. This is for the
              Angled profiles
672         /// </summary>
673         private void startProductionLineOne()
674         {
675            try
676             {
677                 // The name of the CNCCode array variable on the PLC
678                 String var = "Prod_Line_One.CncCode";
679                 String varT = "Prod_Line_One.Produce";
680 
681                 // Line 1
682                 double cutOffL1 = this.configFile.getValue
                      ("CutOffsetLineOne");
683                 double millOffL1 = this.configFile.getValue
                      ("MillOffsetLineOne");
684                 double maxTravelDistL1 = this.configFile.getValue
                      ("MaxTravelDistInn");
685                 CNCGenerator cncGeneratorL1 = new CNCGenerator(400, cutOffL1,
                       millOffL1, maxTravelDistL1, this.cutShaving);
686 
687                 String[] CNCCode = cncGeneratorL1.generateCNCCode
                      (this.angleProfileForProduction);
688 
689                 this.ADSCom.writeToStringArray(var, CNCCode, 12);
690                 this.ADSCom.writeToBoolean(varT, true);
691             }
```

```
692             catch (AdsException err)
693             {
694                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom server og PLS:" + "\n\n" + err.Message;
695                 String cpt = "KOMMUNIKASJONSFEIL";
696                 MessageBox.Show(txt, cpt, MessageBoxButtons.OK, MessageBoxIcon.Error);
697             }
698 
699 
700             // Starting a monitor thread
701             Thread monitoLineOne = new Thread(new ThreadStart(monitorLineOne));
702             monitoLineOne.IsBackground = true;
703             monitoLineOne.Start();
704 
705         }
706 
707         /// <summary>
708         /// Monitoring Production line 2 during the prosess,
709         /// and updates the produced profile list as they come out of the machine
710         /// </summary>
711         private void monitorLineOne()
712         {
713             String profileDone = "Prod_Line_One.ProfileDone";
714             String finishedProfile = "Prod_Line_One.FinishedProfile";
715             bool lastProfileDone = false;
716             int counter = 0;
717 
718             try
719             {
720                 while ((!this.ADSCom.readBoolean(finishedProfile)) && (!this.reset))
721                 {
722                     if (this.ADSCom.readBoolean(profileDone))
723                     {
724                         // A profile is done, so updating the order grid
725                         this.angleProfileForProduction.getSortedAluList().getList().ElementAt(counter).setProduced();
726 
727                         this.Invoke((MethodInvoker)delegate {
728                             this.angleProductionList(); // runs on UI thread
729                         });
730 
731                         counter++;
732                     }
733 
734                     else if ((!this.ADSCom.readBoolean(profileDone)) && (lastProfileDone))
735                     {
736                         lastProfileDone = false;
737                     }
738                 }
739                 if (this.reset)
740                 {
```

```
741                     this.angleOrderProductionList();
742                     this.clearAngle();
743                     this.angleProfileForProduction = null;
744                     // Enable manuel mode
745                     this.manuelModeBtn.Enabled = true;
746                 }
747                 else
748                 {
749                     // The production sequence is complete
750                     // Remove profile from the list
751                     this.Invoke((MethodInvoker)delegate { // runs on UI thread
752                         this.orderHandler.removeAngleProfile(this.angleProfileForProduction);
753                         this.clearAngle();
754                         // The production sequence is complete
755                         this.switchBtn.Text = "PRODUKSJON";
756                         this.angleOrderProductionList();
757                         this.angleProfileForProduction = null;
758                         this.angleProfile = null;
759                         // Enable manuel mode
760                         this.manuelModeBtn.Enabled = true;
761                     });
762                 }
763 
764             }
765             catch (AdsException err)
766             {
767                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom server og PLS:" + "\n\n" + err.Message;
768                 String cpt = "KOMMUNIKASJONSFEIL";
769                 MessageBox.Show(txt, cpt, MessageBoxButtons.OK, MessageBoxIcon.Error);
770             }
771         }
772 
773         /// <summary>
774         /// Starting the production of the chosen guiding profile
775         /// </summary>
776         /// <param name="sender"></param>
777         /// <param name="e"></param>
778         private void produceGuidingBtn_Click(object sender, EventArgs e)
779         {
780             if ((this.guidingProfile != null) && (this.guidingProfileForProduction == null))
781             {
782 
783                 // Disabel manuel mode
784                 this.manuelModeBtn.Enabled = false;
785 
786                 this.guidingProfileForProduction = this.guidingProfile;
787                 this.switchBtnTwo.Text = "ORDRE";
788 
789                 this.guidingProfile = null;
790 
791                 string msg = "Er profiltype " +
```

```
                    this.guidingProfileForProduction.getType() + "  av lengde "
792                       + this.guidingProfileForProduction.getFullLength() +
                      "mm plassert på produksjonslinje 2 ?";
793                 string caption = "ADVARSEL";
794
795                 DialogResult result = MessageBox.Show(msg, caption,
                      MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
796
797                 if (result == DialogResult.Yes)
798                 {
799                     this.guidingProductionList();
800                     this.addGuidingProfileToProductionGrid();
801                     this.startProductionLineTwo();
802                 }
803                 else
804                 {
805                     this.guidingProfileForProduction = null;
806                     this.switchBtnTwo.Text = "PRODUKSJON";
807                 }
808
809
810             }
811             else if(this.guidingProfileForProduction != null)
812             {
813                 string msg = "En profil av type: " +
                      this.guidingProfileForProduction.getType() + " Er allerede
                      under produksjon";
814                 string caption = "Informasjon";
815                 MessageBox.Show(msg, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Information);
816             }
817
818             else
819             {
820                 string msg = "Ingen element er valgt: \nVelg en profil fra
                      listen før du setter igang produksjonen";
821                 string caption = "Informasjon";
822                 MessageBox.Show(msg, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Information);
823             }
824         }
825
826         /// <summary>
827         /// Starting the production on production line 2. This is for the
              guiding profiles
828         /// </summary>
829         private void startProductionLineTwo()
830         {
831
832            try
833             {
834                 // The name of the CNCCode array variable on the PLC
835                 String plcCncVar = "Prod_Line_Two.CncCode";
836                 String plcProduceVar = "Prod_Line_Two.Produce";
837                 // Line 2
838                 double cutOffL2 = this.configFile.getValue
```

```
                    ("CutOffsetLineTwo");
839                 double millOffL2 = this.configFile.getValue
                      ("MillOffsetLineTwo");
840                 double maxTravelDistL2 = this.configFile.getValue
                      ("MaxTravelDistInn");
841                 CNCGenerator cncGeneratorL2 = new CNCGenerator(400, cutOffL2,
                       millOffL2, maxTravelDistL2, this.cutShaving);
842
843                 String[] CNCCode = cncGeneratorL2.generateCNCCode
                      (this.guidingProfileForProduction);
844
845                 this.ADSCom.writeToStringArray(plcCncVar, CNCCode, 12);
846                 this.ADSCom.writeToBoolean(plcProduceVar, true);
847
848                 foreach(string s in CNCCode)
849                 {
850                     Console.WriteLine(s);
851                 }
852             }
853             catch (AdsException err)
854             {
855                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + err.Message;
856                 String cpt = "KOMMUNIKASJONSFEIL";
857                 MessageBox.Show(txt, cpt, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
858             }
859
860             // Starting a monitor thread
861             Thread monitoLieTwo = new Thread(new ThreadStart
                    (monitorLineTwo));
862             monitoLieTwo.IsBackground = true;
863             monitoLieTwo.Start();
864         }
865
866         /// <summary>
867         /// Monitoring Production line 2 during the prosess,
868         /// and updates the produced profile list as they come out of the
              machine
869         /// </summary>
870         private void monitorLineTwo()
871         {
872             String profileDone = "Prod_Line_Two.ProfileDone";
873             String finishedProfile = "Prod_Line_Two.FinishedProfile";
874             int counter = 0;
875             bool lastProfileDone = false;
876
877             try
878             {
879                 while((!this.ADSCom.readBoolean(finishedProfile)) && (!
                      this.reset))
880                 {
881                     if((this.ADSCom.readBoolean(profileDone)) && (!
                        lastProfileDone))
882                     {
883                         lastProfileDone = true;
```

```
884
885                         // A profile is done, so updating the order grid
886                         this.guidingProfileForProduction.getSortedAluList
                        ().getList().ElementAt(counter).setProduced();
887
888                         this.Invoke((MethodInvoker)delegate {
889                             this.guidingProductionList(); // runs on UI
                        thread
890                         });
891
892                         counter++;
893                     }
894
895                     else if ((!this.ADSCom.readBoolean(profileDone)) &&
                        (lastProfileDone))
896                     {
897                         lastProfileDone = false;
898                     }
899                 }
900
901                 if(this.reset)
902                 {
903                     this.Invoke((MethodInvoker)delegate { // runs on UI
                        thread
904                         this.guidingOrderProductionList();
905                         this.clearGuiding();
906                         this.guidingProfileForProduction = null;
907                         // Enable manuel mode
908                         this.manuelModeBtn.Enabled = true;
909                     });
910                 }
911                 else
912                 {
913                     // Remove profile from the list
914                     this.Invoke((MethodInvoker)delegate { // runs on UI
                        thread
915                         this.orderHandler.removeGuidingProfile
                        (this.guidingProfileForProduction);
916                         this.switchBtnTwo.Text = "PRODUKSJON";
917                         this.guidingOrderProductionList();
918                         this.clearGuiding();
919                         // The production sequence is complete
920                         this.guidingProfileForProduction = null;
921                         this.guidingProfile = null;
922                         // Enable manuel mode
923                         this.manuelModeBtn.Enabled = true;
924                     });
925                 }
926
927             }
928             catch (AdsException err)
929             {
930                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                    server og PLS:" + "\n\n" + err.Message;
931                 String cpt = "KOMMUNIKASJONSFEIL";
932                 MessageBox.Show(txt, cpt, MessageBoxButtons.OK,
```

```
                    MessageBoxIcon.Error);
            }
        }

        /// <summary>
        /// Clear the guiding profile from the profiles in production DataGridView.
        /// </summary>
        private void clearGuiding()
        {
            foreach(DataGridViewRow row in this.productionGrid.Rows)
            {
                if (row.Cells["Profiltype"].Value.ToString().Contains("M") || (row.Cells["Profiltype"].Value.ToString().Contains("F")))
                {
                    this.productionGrid.Rows.Remove(row);
                }
            }
        }

        /// <summary>
        /// Clear the angle profile from the profiles in production DataGridView.
        /// </summary>
        private void clearAngle()
        {
            this.productionGrid.Rows.RemoveAt(0);
        }

        /// <summary>
        /// Add a guiding profile to the profiles in production DataGridView.
        /// </summary>
        private void addGuidingProfileToProductionGrid()
        {
            if ((this.productionGrid.Rows.Count < 2) && (this.productionGrid.Rows.Count > 0))
            {
                String[] data = new String[] { this.guidingProfileForProduction.getType(), this.guidingProfileForProduction.getFullLength().ToString(),
                    this.guidingProfileForProduction.getWaste().ToString() };

                // Inserting in second index
                this.productionGrid.Rows.Insert(1, data);
            }

            else if (this.productionGrid.Rows.Count == 0)
            {
                String[] data = new String[] { this.guidingProfileForProduction.getType(), this.guidingProfileForProduction.getFullLength().ToString(),
                    this.guidingProfileForProduction.getWaste().ToString() };
```

```
976                 this.productionGrid.Rows.Insert(0, data);
977             }
978         }
979 
980         /// <summary>
981         /// Add an angle profile to the profiles in production DataGridView.
982         /// </summary>
983         private void addAngleProfileToProductionGrid()
984         {
985             if((this.productionGrid.Rows.Count == 0) ||
                  (this.productionGrid.Rows.Count == 1))
986             {
987                 String[] data = new String[]
                      { this.angleProfileForProduction.getType(),
                      this.angleProfileForProduction.getFullLength().ToString(),
988                       this.angleProfileForProduction.getWaste().ToString() };
989 
990                 this.productionGrid.Rows.Insert(0, data);
991             }
992 
993         }
994 
995         /// <summary>
996         /// Switching the AngleOrder grid (Bottom left grid) from Order list
              to production list or vise verca
997         /// </summary>
998         /// <param name="sender"></param>
999         /// <param name="e"></param>
1000        private void switchBtn_Click(object sender, EventArgs e)
1001        {
1002            if(this.switchBtn.Text.Equals("PRODUKSJON"))
1003            {
1004                // Changing back to the production list
1005                this.switchBtn.Text = "ORDRE";
1006                this.angleProductionList();
1007            }
1008            else
1009            {
1010                // Changing back to the order list
1011                this.switchBtn.Text = "PRODUKSJON";
1012                this.angleOrderProductionList();
1013            }
1014 
1015        }
1016 
1017        /// <summary>
1018        /// Switching the guidingOrder grid (Bottom right grid) from Order
              list to production list or vise verca
1019        /// </summary>
1020        /// <param name="sender"></param>
1021        /// <param name="e"></param>
1022        private void switchBtnTwo_Click(object sender, EventArgs e)
1023        {
1024            if (this.switchBtnTwo.Text.Equals("PRODUKSJON"))
1025            {
1026                // Changing back to the production list
```

```
1027                this.switchBtnTwo.Text = "ORDRE";
1028                this.guidingProductionList();
1029            }
1030            else
1031            {
1032                // Changing back to the order list
1033                this.switchBtnTwo.Text = "PRODUKSJON";
1034                this.guidingOrderProductionList();
1035            }
1036        }
1037
1038        /// <summary>
1039        /// Opening the mauel operating mode for the production cell
1040        /// </summary>
1041        /// <param name="sender"></param>
1042        /// <param name="e"></param>
1043        private void manualModeBtn_Click(object sender, EventArgs e)
1044        {
1045            String manuelMode = "gvl.manual";
1046            String stopL2 = "GVL.StopL2";
1047            try
1048            {
1049                this.ADSCom.writeToBoolean(manuelMode, true);
1050                this.ADSCom.writeToBoolean(stopL2, true);
1051                ManuelMode manuel = new ManuelMode(this.ADSCom);
1052                manuel.Show();
1053            }
1054            catch (AdsException err)
1055            {
1056                String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + err.Message;
1057                String cpt = "KOMMUNIKASJONSFEIL";
1058                MessageBox.Show(txt, cpt, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
1059            }
1060
1061        }
1062
1063        /// <summary>
1064        /// Opening the RegisterDialog
1065        /// </summary>
1066        /// <param name="sender"></param>
1067        /// <param name="e"></param>
1068        private void profileTypeBtn_Click(object sender, EventArgs e)
1069        {
1070            RegisterDialog register = new RegisterDialog(this.orderHandler);
1071            register.Show();
1072        }
1073
1074        /// <summary>
1075        /// Handling the config button event
1076        /// </summary>
1077        /// <param name="sender"></param>
1078        /// <param name="e"></param>
1079        private void configBtn_Click(object sender, EventArgs e)
1080        {
```

```
1081
1082            ConfigDialog config = new ConfigDialog(this.configFile,
                  this.orderHandler, this.cutShaving);
1083            config.Show();
1084        }
1085
1086        /// <summary>
1087        /// Reset the stepper motors if an error ocurs
1088        /// </summary>
1089        /// <param name="sender"></param>
1090        /// <param name="e"></param>
1091        private void reset_Stpr_Click(object sender, EventArgs e)
1092        {
1093            try
1094            {
1095                string reset = "Stepper_Setup.ResetStpr";
1096                this.ADSCom.writeToBoolean(reset, true);
1097                this.reset_Stpr.Visible = false;
1098            }
1099            catch (AdsException err)
1100            {
1101                String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + err.Message;
1102                String cpt = "KOMMUNIKASJONSFEIL";
1103                MessageBox.Show(txt, cpt, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
1104            }
1105
1106        }
1107
1108        /// <summary>
1109        /// Closing the RUFO application
1110        /// </summary>
1111        /// <param name="sender"></param>
1112        /// <param name="e"></param>
1113        private void RUFO_Closing(object sender, FormClosingEventArgs e)
1114        {
1115            string txt = "Er du sikker på at du vil avslutte
                  applikasjonen ?";
1116            string capt = "ADVARSEL";
1117            DialogResult result = MessageBox.Show(txt, capt,
                  MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
1118
1119            if(result == DialogResult.Yes)
1120            {
1121                this.emcyListening = false;
1122                Environment.Exit(0);
1123            }
1124
1125            else
1126            {
1127                e.Cancel = true;
1128            }
1129        }
1130
1131        /// <summary>
```

```
1132          /// Monitor for error messages from the PLC
1133          /// </summary>
1134          private void monitor()
1135          {
1136              // Variables from the plc
1137              string emcy = "gvl.emcyStop";
1138              string ErrorFeedNmbL2 = "Stepper_Setup.ErrorFeedNmbL2";
1139              string ErrorOutputNmbL2 = "Stepper_Setup.ErrorOutputNmbL2";
1140              string ErrorMilltNmbL2 = "Stepper_Setup.ErrorMillNmbL2";
1141
1142              // Flags
1143              bool emcyAktivatedOnce = false;
1144              bool emcyDeactivatedOnce = false;
1145              bool ErrorFeedNmbL2Shown = false;
1146              bool ErrorOutputNmbL2Shown = false;
1147              bool ErrorMilltNmbL2Shown = false;
1148
1149
1150              while (this.emcyListening)
1151              {
1152                  emcyDeactivatedOnce = false;
1153                  emcyAktivatedOnce = false;
1154
1155
1156                  try
1157                  {
1158                      // If emergency stop is pressed
1159                      while (this.ADSCom.readBoolean(emcy))
1160                      {
1161                          String stopL2 = "GVL.StopL2";
1162
1163                          // Need to stop the plc program if emergency stop
1164                          this.ADSCom.writeToBoolean(stopL2, true);
1165
1166                          if (!emcyAktivatedOnce)
1167                          {
1168                              this.Invoke((MethodInvoker)delegate {
1169
1170                                  emcyDeactivatedOnce = true;
1171
1172                                  this.addErrorToLog("Nødstop aktivert",
                        Color.Red);
1173
1174                                  string capt = "NØDSTROPP";
1175                                  string txt = "Nødstopp aktivert";
1176
1177                                  MessageBox.Show(txt, capt,
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
1178
1179                                  // Disabling btns
1180                                  this.stopBtn.Enabled = false;
1181                                  this.produceAnglebtn.Enabled = false;
1182                                  this.produceGuidingBtn.Enabled = false;
1183
1184                                  // Enabling btns
1185                                  this.restartBtn.Enabled = true;
```

```
1186                                this.startBtn.Enabled = true;
1187
1188                                emcyAktivatedOnce = true;
1189                            });
1190                        }
1191                    }
1192
1193                    // Other errors
1194
1195                    // Stepper errors
1196                    // Stepper for feeding Line 2
1197                    this.Invoke((MethodInvoker)delegate {
1198                        if ((this.ADSCom.readUint(ErrorFeedNmbL2)) != 0 && (!
                    ErrorFeedNmbL2Shown))
1199                        {
1200                            string error = "Error stepper feeding: \n" +
                    this.ADSCom.readUint(ErrorFeedNmbL2);
1201
1202                            this.addErrorToLog(error, Color.Red);
1203                            ErrorFeedNmbL2Shown = true;
1204                            // Enable reset stepper btn
1205                            this.reset_Stpr.Visible = true;
1206                        }
1207                        else if ((this.ADSCom.readUint(ErrorFeedNmbL2)) == 0
                    && (ErrorFeedNmbL2Shown))
1208                        {
1209                            this.addErrorToLog("Steppers reset",
                    Color.Green);
1210                            ErrorFeedNmbL2Shown = false;
1211                        }
1212
1213                        // Stepper for feeding out Line 2
1214                        if ((this.ADSCom.readUint(ErrorOutputNmbL2) != 0) &&
                    (!ErrorOutputNmbL2Shown))
1215                        {
1216                            string error = "Error stepper out: \n" +
                    this.ADSCom.readUint(ErrorOutputNmbL2);
1217
1218                            this.addErrorToLog(error, Color.Red);
1219                            ErrorOutputNmbL2Shown = true;
1220                            // Enable reset stepper btn
1221                            this.reset_Stpr.Visible = true;
1222                        }
1223                        else if ((this.ADSCom.readUint(ErrorOutputNmbL2)) ==
                    0 && (ErrorOutputNmbL2Shown))
1224                        {
1225                            this.addErrorToLog("Steppers reset",
                    Color.Green);
1226                            ErrorOutputNmbL2Shown = false;
1227                        }
1228
1229                        // Stepper for adjusting milling
1230                        if ((this.ADSCom.readUint(ErrorMilltNmbL2) != 0) &&
                    (!ErrorMilltNmbL2Shown))
1231                        {
1232                            string error = "Error stepper milling: \n" +
```

```
                        this.ADSCom.readUint(ErrorMilltNmbL2);
1233
1234                            this.addErrorToLog(error, Color.Red);
1235                            ErrorMilltNmbL2Shown = true;
1236                            // Enable reset stepper btn
1237                            this.reset_Stpr.Visible = true;
1238                        }
1239                        else if ((this.ADSCom.readUint(ErrorMilltNmbL2)) == 0
                         && (ErrorMilltNmbL2Shown))
1240                        {
1241                            this.addErrorToLog("Steppers reset",
                        Color.Green);
1242                            ErrorMilltNmbL2Shown = false;
1243                        }
1244                    });
1245                }
1246                catch (AdsException e)
1247                {
1248                    String txt = "Noe gikk feil med ADS kommunikasjonen
                        mellom server og PLS:" + "\n\n" + e.Message + "\n\n" +
1249                        "Systemet kan dermed ikke registrere nødstoppen så
                        applikasjonen og system må avsluttes";
1250                    String cpt = "KOMMUNIKASJONSFEIL";
1251                    Console.WriteLine(e.StackTrace);
1252                    MessageBox.Show(txt, cpt, MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
1253                    Environment.Exit(0);
1254                }
1255
1256                catch (Exception err)
1257                {
1258                    String txt = "Noe gikk feil" + "\n\n" + err.Message;
1259                    String cpt = "NOE GIKK FEIL";
1260                    MessageBox.Show(txt, cpt);
1261                }
1262
1263                // Emcy deactivated, add to log
1264                if (emcyDeactivatedOnce)
1265                {
1266                    this.Invoke((MethodInvoker)delegate {
1267                        addErrorToLog("Nødstop deaktivert", Color.Green);
1268                    });
1269
1270                }
1271            }
1272
1273        }
1274
1275        /// <summary>
1276        /// Update the orderlists
1277        /// </summary>
1278        void GUIHandler.updateOrders()
1279        {
1280            this.Invoke((MethodInvoker)delegate {
1281                this.guidingOrderProductionList();
1282                this.angleOrderProductionList();// runs on UI thread
```

```
1283            });
1284        }
1285
1286        /// <summary>
1287        /// Handling exceptions by using a MessageBox
1288        /// </summary>
1289        void GUIHandler.exceptionHandling(Exception e)
1290        {
1291            this.Invoke((MethodInvoker)delegate {
1292
1293            if (e is WrongFileFormatException)
1294            {
1295                    String txt = "En av Ordre.csv filene du la til, inneholdt
                        ikke riktig fil format" + "\n" +
1296                    "Orderen ble dermed ikke generert";
1297
1298                    String cap = "NOE GIKK GALT";
1299                    MessageBox.Show(txt, cap, MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
1300
1301
1302            }
1303            else if (e is AngleExceededException)
1304            {
1305                String txt = "En av Ordre.csv filene du la til, inneholdt
                    ikke riktig fil format" + "\n" +
1306                  "Orderen ble dermed ikke generert." +"\n\n" +
1307                  "Feilmeldingen lyder slik:\n"+e.Message;
1308
1309                String cap = "NOE GIKK GALT";
1310                MessageBox.Show(txt, cap, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
1311            }
1312            else if(e is NotFiniteNumberException)
1313            {
1314                String txt = "Noe gikk galt" + "\n" +
1315                    "Orderen ble dermed ikke generert." + "\n\n" +
1316                    "Feilmeldingen lyder slik:\n" + e.Message;
1317
1318                String cap = "NOE GIKK GALT";
1319                MessageBox.Show(txt, cap, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
1320            }
1321
1322            else if(e is FormatException)
1323            {
1324                String txt = "Noe gikk galt" + "\n" +
1325                    "Orderen ble dermed ikke generert." + "\n\n" +
1326                    "Feilmeldingen lyder slik:\n" + e.Message;
1327
1328                String cap = "NOE GIKK GALT";
1329                MessageBox.Show(txt, cap, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
1330            }
1331
1332            });
```

```
1333          }
1334      }
1335  }
1336
```

```
1  using RUFOServer.Program.Server;
2  using System;
3  using System.Collections.Generic;
4  using System.ComponentModel;
5  using System.Data;
6  using System.Drawing;
7  using System.IO;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11 using System.Windows.Forms;
12
13 namespace RUFOServer.Main
14 {
15     /// <summary>
16     ///
17     /// </summary>
18     public partial class RegisterDialog : Form
19     {
20         private String[] profileInfo;
21         private int index;
22         private OrderHandler orderHandler;
23         private String path;
24
25         /// <summary>
26         ///
27         /// </summary>
28         public RegisterDialog(OrderHandler orderHandler)
29         {
30             InitializeComponent();
31
32             this.orderHandler = orderHandler;
33             setUpGrid();
34             getFilePath();
35             setupTxtBoxes();
36             updateGridView();
37
38         }
39
40         /// <summary>
41         /// Finding the filepath for the ProfileInfo.txt file
42         /// </summary>
43         private void getFilePath()
44         {
45             // Finding the path of the program
46             this.path = Path.GetFullPath(Application.StartupPath);
47             // Going to folders back to find the original files
48             this.path = Path.GetFullPath(Path.Combine(this.path, @"..\..\"));
49             // Going into the folder containing the original ProfileInfo.txt
50             this.path = this.path + @"Program\Register\ProfileInfo.txt";
51         }
52
53         /// <summary>
54         /// Initializing the DataGridView showing the ProfileInfo.txt
              information.
55         /// </summary>
```

```
56          private void setUpGrid()
57          {
58              this.profileInfoGrid.ColumnCount = 4;
59              this.profileInfoGrid.ReadOnly = true;
60              this.profileInfoGrid.MultiSelect = false;
61              this.profileInfoGrid.RowHeadersVisible = false;
62              this.profileInfoGrid.Columns[0].Name = "Profiltype";
63              this.profileInfoGrid.Columns[0].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
64              this.profileInfoGrid.Columns[1].Name = "Produksjonslinje";
65              this.profileInfoGrid.Columns[1].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
66              this.profileInfoGrid.Columns[2].Name = "Profil bredde";
67              this.profileInfoGrid.Columns[2].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
68              this.profileInfoGrid.Columns[3].Name = "Profil høyde";
69              this.profileInfoGrid.Columns[3].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
70              this.profileInfoGrid.RowTemplate.Height = 40;
71              this.profileInfoGrid.AutoSizeColumnsMode =
                  DataGridViewAutoSizeColumnsMode.Fill;
72          }
73
74          /// <summary>
75          /// Initialing the text boxes used to add information to the
              ProfileInfo.txt file.
76          /// </summary>
77          private void setupTxtBoxes()
78          {
79              this.profileType.MaxLength = 15;
80              this.profileType.Multiline = false;
81
82              this.prodLine.MaxLength = 1;
83              this.prodLine.Multiline = false;
84
85              this.pWidth.MaxLength = 5;
86              this.pWidth.Multiline = false;
87
88              this.pHeight.MaxLength = 5;
89              this.pHeight.Multiline = false;
90          }
91
92          /// <summary>
93          /// Adding information to the ProfileInfo.txt file.
94          /// </summary>
95          /// <param name="sender"></param>
96          /// <param name="e"></param>
97          private void addBtn_Click(object sender, EventArgs e)
98          {
99              // All the textboxes have inputs
100             if((this.profileType.Text.Any()) && (this.prodLine.Text.Any()) &&
                  (this.pWidth.Text.Any()) && (this.pHeight.Text.Any()))
101             {
102                 String pType = this.profileType.Text;
103                 bool exists = false;
104
```

```
105                 // Check so there are no matching profileTypes
106                 foreach(DataGridViewRow row in this.profileInfoGrid.Rows)
107                 {
108                     for (int i = 0; i < this.profileInfoGrid.Columns.Count; i+
                    +)
109                     {
110                         if(pType.Equals(row.Cells[i].Value.ToString()))
111                         {
112                             exists = true;
113                             break;
114                         }
115                     }
116                 }
117 
118                 // If the profile already exists in the list
119                 if(exists)
120                 {
121                     String text = "Denne profil typen eksisterer allerede i
                    listen";
122                     String caption = "FEIL";
123 
124                     MessageBox.Show(text, caption, MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
125                 }
126 
127                 else
128                 {
129                     String[] line = new string[1];
130                     // In the program production line 1 = 0 and line 2 = 1
131                     int pLine = Int16.Parse(this.prodLine.Text) - 1;
132 
133                     line[0] = this.profileType.Text.ToString() + "," +
                    pLine.ToString() + "," +
134                         this.pWidth.Text.ToString() + "," +
                    this.pHeight.Text.ToString();
135                     Console.WriteLine(pLine);
136                     this.addLine(line);
137                 }
138 
139 
140             }
141 
142             else
143             {
144                 String text = "Alle textfeltene må være fylt ut før du legger
                  til en ny profiltype"+ "\n" +
145                   "Disse tekstefeltene mangler informasjon: "+ "\n\n";
146                 String caption = "FEIL";
147 
148                 if (!this.profileType.Text.Any())
149                 {
150                     text = text + "Profiltype\n";
151                 }
152 
153                 if (!this.prodLine.Text.Any())
154                 {
```

```
155                     text = text + "Produksjonslinje\n";
156                 }
157 
158                 if (!this.pWidth.Text.Any())
159                 {
160                     text = text + "Profil bredde\n";
161                 }
162 
163                 if (!this.pHeight.Text.Any())
164                 {
165                     text = text + "Profil høyde\n";
166                 }
167 
168                 MessageBox.Show(text, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
169             }
170         }
171 
172         /// <summary>
173         /// Production line text box event handler.
174         /// </summary>
175         /// <param name="sender"></param>
176         /// <param name="e"></param>
177         private void prodLine_OnKeyPressed(object sender, KeyPressEventArgs e)
178         {
179             char ch = e.KeyChar;
180 
181             // Is the input is not a number or 8 = backspace
182             if((!Char.IsDigit(ch)) && (ch != 8))
183             {
184                 e.Handled = true;
185             }
186 
187             // Can only type in the number 1 or 2
188             else if((Char.IsDigit(ch)) && ((ch < 49) || (ch > 50)))
189             {
190                 e.Handled = true;
191             }
192 
193         }
194 
195         /// <summary>
196         /// Profile width text box event handler.
197         /// </summary>
198         /// <param name="sender"></param>
199         /// <param name="e"></param>
200         private void pWidth_OnKeyPressed(object sender, KeyPressEventArgs e)
201         {
202             char ch = e.KeyChar;
203 
204             // Is the input is not a number or 8 = backspace
205             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
206             {
207                 e.Handled = true;
208             }
209             // If char = '.', make sure it is the only one
```

```
210             else if(ch == 46)
211             {
212                 if(this.pWidth.Text.Contains('.') || (this.pWidth.TextLength
                      == 0))
213                 {
214                     e.Handled = true;
215                 }
216             }
217 
218         }
219 
220         /// <summary>
221         /// Profile height text box event handler.
222         /// </summary>
223         /// <param name="sender"></param>
224         /// <param name="e"></param>
225         private void pHeight_OnKeyPressed(object sender, KeyPressEventArgs e)
226         {
227             char ch = e.KeyChar;
228 
229             // Is the input is not a number or 8 = backspace
230             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
231             {
232                 e.Handled = true;
233             }
234 
235             // If char = '.', make sure it is the only one
236             else if (ch == 46)
237             {
238                 if (this.pWidth.Text.Contains('.') || (this.pWidth.TextLength
                      == 0))
239                 {
240                     e.Handled = true;
241                 }
242             }
243         }
244 
245         /// <summary>
246         /// Updating the gridview in the GUI
247         /// </summary>
248         private void updateGridView()
249         {
250             if(this.profileInfo !=  null)
251             {
252                 // Clearing the array for the next file
253                 Array.Clear(this.profileInfo, 0, this.profileInfo.Length);
254                 this.profileInfoGrid.Rows.Clear();
255             }
256 
257             // Reading the ProfileInfo.txt file
258             this.profileInfo = File.ReadAllLines(this.path);
259             string[] data;
260             int prodLine = 0;
261 
262             for (int i = 2; i < this.profileInfo.Length; i++)
263             {
```

```
264                 data = this.profileInfo[i].Split(',');
265                 prodLine = Int32.Parse(data[1]);
266                 prodLine = prodLine + 1;
267                 String[] gridData = new String[] { data[0], prodLine.ToString
                      (), data[2], data[3] };
268                 this.profileInfoGrid.Rows.Add(gridData);

269             }
270         }
271 
272         /// <summary>
273         /// Closing the dialog
274         /// </summary>
275         /// <param name="sender"></param>
276         /// <param name="e"></param>
277         private void exitBtn_Click(object sender, EventArgs e)
278         {
279             this.Close();
280         }
281 
282         /// <summary>
283         /// Choosing the element from the gridView
284         /// </summary>
285         /// <param name="sender"></param>
286         /// <param name="e"></param>
287         private void profileInfoGrid_CellClick(object sender,
              DataGridViewCellEventArgs e)
288         {
289             if (e.RowIndex >= 0)
290             {
291                 // Need to make sure the index starts at 2, because the
                      ProfileInfo.txt file starts at line 2
292                 this.index = e.RowIndex + 2;
293             }
294             else
295             {
296                 this.index = 0;
297             }
298 
299         }
300 
301         /// <summary>
302         /// Delete the selected item from the gridview and .txt file
303         /// </summary>
304         /// <param name="sender"></param>
305         /// <param name="e"></param>
306         private void deleteBtn_Click(object sender, EventArgs e)
307         {
308             if(this.index >= 2)
309             {
310                 String text = "Er du helt sikker på at du vil slette denne
                      filen ?\n\n" +
311                       "Sletting av denne filen medfører at det ikke lengre
                        er mulig å produsere denne typen profil";
312                 String caption = "ADVARSEL";
313                 DialogResult result =  MessageBox.Show(text, caption,
```

```
                    MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
314
315                 if (result == DialogResult.Yes)
316                 {
317                     deleteLine();
318                 }
319
320                 else
321                 {
322                     // Reseting the index back to zero, to prevent accidental deletion
323                     this.index = 0;
324                 }
325             }
326
327             else
328             {
329                 String text = "Ingen elemet fra listen er valgt.\n"+
330                     "Venligst velg et element fra listen for å slette det";
331                 String caption = "INFORMASJON";
332                 MessageBox.Show(text, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Information);
333             }
334         }
335
336         /// <summary>
337         /// Deleting the selected row from the gridview when the delete button is pressed
338         /// </summary>
339         /// <param name="sender"></param>
340         /// <param name="e"></param>
341         private void profileInfoGrid_OnDelete(object sender, KeyEventArgs e)
342         {
343             if (e.KeyCode == Keys.Delete)
344             {
345                 if (this.index >= 2)
346                 {
347                     String text = "Er du helt sikker på at du vil slette denne filen ?\n\n" +
348                             "Sletting av denne filen medfører at det ikke lengre er mulig å produsere denne typen profil";
349                     String caption = "ADVARSEL";
350                     DialogResult result = MessageBox.Show(text, caption, MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
351
352                     if (result == DialogResult.Yes)
353                     {
354                         deleteLine();
355                     }
356
357                     else
358                     {
359                         // Reseting the index back to zero, to prevent accidental deletion
360                         this.index = 0;
361                     }
```

```
362                 }
363             }
364         }
365 
366         /// <summary>
367         /// Adding a line to the ProfileInfo.txt file.
368         /// </summary>
369         private void addLine(String[] lines)
370         {
371             File.AppendAllLines(this.path, lines);
372             // Update the list
373             this.updateGridView();
374             // Update the orderHandler with the new registers
375             this.orderHandler.updateAluRegisters();
376         }
377 
378         /// <summary>
379         /// Deleting a line from the ProfileInfo.txt file.
380         /// </summary>
381         private void deleteLine()
382         {
383             this.profileInfoGrid.Rows.RemoveAt(this.index - 2);
384             String[] lines = new String[this.profileInfo.Length - 1];
385 
386             // File explanation
387             lines[0] = "# PROFILETYPE, NUMBER INDICATOR: 0 = ANGLE, 1 =
                GUIDING, PROFILE WIDTH, PROFILE HEIGHT";
388             lines[1] = "";
389             int counter = 2;
390 
391             for (int i = 2; i < lines.Length; i++)
392             {
393                 if(i == this.index)
394                 {
395                     // Do nothing make sure the line does not get written to
                    the file
396                     counter++;
397                     lines[i] = this.profileInfo[counter];
398                 }
399                 else
400                 {
401                     lines[i] = this.profileInfo[counter];
402                 }
403                 counter++;
404             }
405 
406             File.WriteAllLines(this.path, lines);
407             // Update the list
408             this.updateGridView();
409             // Update the orderHandler with the new registers
410             this.orderHandler.updateAluRegisters();
411             this.index = 0;
412 
413         }
414 
415 
```

```
416         }
417     }
418
```

```
 1  using RUFOServer.Program.Server;
 2  using System;
 3  using System.Collections.Generic;
 4  using System.ComponentModel;
 5  using System.Data;
 6  using System.Diagnostics;
 7  using System.Drawing;
 8  using System.Linq;
 9  using System.Text;
10  using System.Threading;
11  using System.Threading.Tasks;
12  using System.Windows.Forms;
13  using TwinCAT.Ads;
14
15  namespace RUFOServer.Main
16  {
17      /// <summary>
18      /// The ManuelMode class enables manuel operation of the production cell.
19      /// </summary>
20      public partial class ManuelMode : Form
21      {
22          private List<Action> lineOneBtns;
23          private List<Action> lineTwoBtns;
24
25          // Line one event handling
26          private int lineOneEvent = -1;
27          private bool mouseDownLineOne;
28
29          // Line to event handling
30          private int lineTwoEvent = -1;
31          private bool mouseDownLineTwo;
32
33          private bool isDone = false;
34
35          // Communication
36          private ADSCommunication ADScom;
37
38          /// <summary>
39          /// Creating an instance of ManuelMode
40          /// </summary>
41          /// <param name="ADScom">
42          /// An ADSCommunication object used to communicate with the production
               cell PLC
43          /// </param>
44          public ManuelMode(ADSCommunication ADScom)
45          {
46              InitializeComponent();
47              this.ADScom = ADScom;
48              this.lineOneBtns = new List<Action>();
49              this.lineTwoBtns = new List<Action>();
50              addEvenhandlers();
51              setupGrids();
52          }
53
54          /// <summary>
55          /// Adding all the buttons event handlers for each production line
```

```
56          /// in their own respective list.
57          /// </summary>
58          private void addEvenhandlers()
59          {
60              // Eventhandlers line one
61              this.lineOneBtns.Add(() => lineOneGripperOne());
62              this.lineOneBtns.Add(() => lineOneGripperTwo());
63              this.lineOneBtns.Add(() => lineOneHolderOne());
64              this.lineOneBtns.Add(() => lineOneHolderTwo());
65              this.lineOneBtns.Add(() => lineOneHolderThree());
66              this.lineOneBtns.Add(() => lineOneTransportInnPluss(this.isDone));
67              this.lineOneBtns.Add(() => lineOneTransportInnMinus(this.isDone));
68              this.lineOneBtns.Add(() => lineOneTransportOutPluss(this.isDone));
69              this.lineOneBtns.Add(() => lineOneTransportOutMinus(this.isDone));
70              this.lineOneBtns.Add(() => lineOneMillOn());
71              this.lineOneBtns.Add(() => lineOneMillOff());
72              this.lineOneBtns.Add(() => lineOneMill());
73              this.lineOneBtns.Add(() => lineOneMillStationUp(this.isDone));
74              this.lineOneBtns.Add(() => lineOneMillStationDown(this.isDone));
75              this.lineOneBtns.Add(() => sawOn());
76              this.lineOneBtns.Add(() => sawOff());
77
78              // Eventhandlers line two
79              this.lineTwoBtns.Add(() => lineTwoGripperOne());
80              this.lineTwoBtns.Add(() => lineTwoGripperTwo());
81              this.lineTwoBtns.Add(() => lineTwoHolderOne());
82              this.lineTwoBtns.Add(() => lineTwoHolderTwo());
83              this.lineTwoBtns.Add(() => lineTwoHolderThree());
84              this.lineTwoBtns.Add(() => lineTwoTransportInnPluss(this.isDone));
85              this.lineTwoBtns.Add(() => lineTwoTransportInnMinus(this.isDone));
86              this.lineTwoBtns.Add(() => lineTwoTransportOutPluss(this.isDone));
87              this.lineTwoBtns.Add(() => lineTwoTransportOutMinus(this.isDone));
88              this.lineTwoBtns.Add(() => lineTwoMillOn());
89              this.lineTwoBtns.Add(() => lineTwoMillOff());
90              this.lineTwoBtns.Add(() => lineTwoMill());
91              this.lineTwoBtns.Add(() => lineTwoMillStationUp(this.isDone));
92              this.lineTwoBtns.Add(() => lineTwoMillStationDown(this.isDone));
93              this.lineTwoBtns.Add(() => sawOn());
94              this.lineTwoBtns.Add(() => sawOff());
95          }
96
97          /// <summary>
98          /// Initializing the DataGridViews,
99          /// used for the buttons to controll production line one and two.
100         /// </summary>
101         private void setupGrids()
102         {
103             // Prod line one
104             this.prodLineOneLabel.Text = "MANUEL KJØRING PRODUKSJONSLINJE 1";
105             this.prodLineOne.ColumnCount = 2;
106             this.prodLineOne.MultiSelect = false;
107             this.prodLineOne.RowHeadersVisible = false;
108
109             this.prodLineOne.Columns[0].ReadOnly = false;
110             DataGridViewButtonColumn manualBtnsOne = new
                  DataGridViewButtonColumn();
```

```
111             manualBtnsOne.Name = "Manuel kjøring";
112             manualBtnsOne.Text = "Manuel kjøring";
113 
114             this.prodLineOne.Columns.Insert(0, manualBtnsOne);
115             this.prodLineOne.Columns[0].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
116             this.prodLineOne.Columns[1].ReadOnly = true;
117             this.prodLineOne.Columns[1].Name = "Beskrivelse";
118             this.prodLineOne.Columns[1].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
119 
120             this.prodLineOne.Columns[2].Visible = false;
121 
122             this.prodLineOne.RowTemplate.Height = 65;
123             this.prodLineOne.AutoSizeColumnsMode =
                  DataGridViewAutoSizeColumnsMode.Fill;
124             this.prodLineOne.ScrollBars = ScrollBars.Vertical;
125 
126             addBtnsProdLineOne();
127 
128             // Prod line two
129             this.prodLineTwoLabel.Text = "MANUEL KJØRING PRODUKSJONSLINJE 2";
130             this.prodLineTwogrid.ColumnCount = 2;
131             this.prodLineTwogrid.MultiSelect = false;
132             this.prodLineTwogrid.RowHeadersVisible = false;
133 
134             this.prodLineTwogrid.Columns[0].ReadOnly = false;
135             DataGridViewButtonColumn manualBtnsTwo = new
                  DataGridViewButtonColumn();
136             manualBtnsTwo.Name = "Manuel kjøring";
137             manualBtnsTwo.Text = "Manuel kjøring";
138 
139             this.prodLineTwogrid.Columns.Insert(0, manualBtnsTwo);
140             this.prodLineTwogrid.Columns[0].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
141             this.prodLineTwogrid.Columns[1].ReadOnly = true;
142             this.prodLineTwogrid.Columns[1].Name = "Beskrivelse";
143             this.prodLineTwogrid.Columns[1].SortMode =
                  DataGridViewColumnSortMode.NotSortable;
144 
145             this.prodLineTwogrid.Columns[2].Visible = false;
146 
147             this.prodLineTwogrid.RowTemplate.Height = 65;
148             this.prodLineTwogrid.AutoSizeColumnsMode =
                  DataGridViewAutoSizeColumnsMode.Fill;
149             this.prodLineTwogrid.ScrollBars = ScrollBars.Vertical;
150 
151             addBtnsProdLineTwo();
152         }
153 
154         /// <summary>
155         /// Adding the control buttons for production line one.
156         /// </summary>
157         private void addBtnsProdLineOne()
158         {
159             String gripperOne = "Description";
```

```
160             this.prodLineOne.Rows.Add("GRIPER 1", gripperOne);
161             String gripperTwo = "Description";
162             this.prodLineOne.Rows.Add("GRIPER 2", gripperTwo);
163             String profileHolderOne = "Description";
164             this.prodLineOne.Rows.Add("PROFIL HOLDER 1", profileHolderOne);
165             String profileHolderTwo = "Description";
166             this.prodLineOne.Rows.Add("PROFIL HOLDER 2", profileHolderTwo);
167             String profileHolderThree = "Description";
168             this.prodLineOne.Rows.Add("PROFIL HOLDER 3", profileHolderThree);
169             String TransportInnPluss = "Description";
170             this.prodLineOne.Rows.Add("TRANSPORT INN +", TransportInnPluss);
171             String TransportInnMinus = "Description";
172             this.prodLineOne.Rows.Add("TRANSPORT INN -", TransportInnMinus);
173             String TransportOutPluss = "Description";
174             this.prodLineOne.Rows.Add("TRANSPORT UT +", TransportOutPluss);
175             String TransportOutMinus = "Description";
176             this.prodLineOne.Rows.Add("TRANSPORT UT -", TransportOutMinus);
177             String millingOn = "Description";
178             this.prodLineOne.Rows.Add("BORRING PÅ", millingOn);
179             String millingOf = "Description";
180             this.prodLineOne.Rows.Add("BORRING AV", millingOf);
181             String mill = "Description";
182             this.prodLineOne.Rows.Add("BOR", mill);
183             String millingUp = "Description";
184             this.prodLineOne.Rows.Add("BOR STASJON OPP", millingUp);
185             String millingDown = "Description";
186             this.prodLineOne.Rows.Add("BOR STASJON NED", millingDown);
187             String sawOn = "Slå på sag montert på robotarmen";
188             this.prodLineOne.Rows.Add("SAG PÅ", sawOn);
189             String sawOff = "Slå av sag montert på robotarmen";
190             this.prodLineOne.Rows.Add("SAG AV", sawOff);
191         }
192
193         /// <summary>
194         /// Adding the control buttons for production line two.
195         /// </summary>
196         private void addBtnsProdLineTwo()
197         {
198             String gripperOne = "Aktiver/deaktiver profil griperen til
                innmatingen";
199             this.prodLineTwogrid.Rows.Add("GRIPER 1", gripperOne);
200
201             String gripperTwo = "Aktiver/deaktiver profil griperen til
                utmatingen";
202             this.prodLineTwogrid.Rows.Add("GRIPER 2", gripperTwo);
203
204             String profileHolderOne = "Aktiver/deaktiver låsestasjon 1";
205             this.prodLineTwogrid.Rows.Add("PROFIL HOLDER 1",
                profileHolderOne);
206
207             String profileHolderTwo = "Aktiver/deaktiver låsestasjon 2";
208             this.prodLineTwogrid.Rows.Add("PROFIL HOLDER 2",
                profileHolderTwo);
209
210             String profileHolderThree = "Aktiver/deaktiver låsestasjon 3";
211             this.prodLineTwogrid.Rows.Add("PROFIL HOLDER 3",
```

```
                profileHolderThree);
212
213             String TransportInnPluss = "Hold knappen inne for å kjøre
                  innmating innover";
214             this.prodLineTwogrid.Rows.Add("TRANSPORT INN +",
                  TransportInnPluss);
215
216             String TransportInnMinus = "Hold knappen inne for å kjøre
                  innmating tilbake";
217             this.prodLineTwogrid.Rows.Add("TRANSPORT INN -",
                  TransportInnMinus);
218
219             String TransportOutPluss = "Hold knappen inne for å kjøre utmating
                   innover";
220             this.prodLineTwogrid.Rows.Add("TRANSPORT UT +",
                  TransportOutPluss);
221
222             String TransportOutMinus = "Hold knappen inne for å kjøre utmating
                   utover";
223             this.prodLineTwogrid.Rows.Add("TRANSPORT UT -",
                  TransportOutMinus);
224
225             String millingOn = "Slå på fres";
226             this.prodLineTwogrid.Rows.Add("BORRING PÅ", millingOn);
227
228             String millingOff = "Slå av fres";
229             this.prodLineTwogrid.Rows.Add("BORRING AV", millingOff);
230
231             String mill = "Kjør fres frem og tilbake";
232             this.prodLineTwogrid.Rows.Add("BOR", mill);
233
234             String millingUp = "Hold knappen inne for å kjøre fres opp";
235             this.prodLineTwogrid.Rows.Add("BOR STASJON OPP", millingUp);
236
237             String millingDown = "Hold knappen inne for å kjøre fres ned";
238             this.prodLineTwogrid.Rows.Add("BOR STASJON NED", millingDown);
239
240             String sawOn = "Slå på sag montert på robotarmen";
241             this.prodLineTwogrid.Rows.Add("SAG PÅ", sawOn);
242
243             String sawOff = "Slå av sag montert på robotarmen";
244             this.prodLineTwogrid.Rows.Add("SAG AV", sawOff);
245         }
246
247
248         /// <summary>
249         /// Handling the buttons on production line one except the transport
              related ones.
250         /// </summary>
251         /// <param name="sender"></param>
252         /// <param name="e"></param>
253         private void prodLineOne_CellContentClick(object sender,
              DataGridViewCellEventArgs e)
254         {
255             DataGridViewButtonCell btn = null;
256
```

```
257             if ((e.ColumnIndex == this.prodLineOne.Columns["Manuel
                  kjøring"].Index) && (e.RowIndex >= 0))
258             {
259                 btn = (DataGridViewButtonCell)(this.prodLineOne.Rows
                      [e.RowIndex].Cells[e.ColumnIndex]);
260
261                 // This index is for the hold button events
262                 if ((btn.Value.ToString().Contains("TRANSPORT")) ||
                      (btn.Value.ToString().Contains("STASJON")))
263                 {
264                     // Do nothing, the other eventhandler handles it
265                 }
266                 else
267                 {
268                     this.lineOneBtns.ElementAt(e.RowIndex).Invoke();
269                 }
270
271             }
272         }
273
274         /// <summary>
275         /// Handling the transport buttons on production line one when button
              is pressed.
276         /// </summary>
277         /// <param name="sender"></param>
278         /// <param name="e"></param>
279         private void prodLineOneGrid_MouseDown(object sender,
              DataGridViewCellMouseEventArgs e)
280         {
281             DataGridViewButtonCell btn = (DataGridViewButtonCell)
                  (this.prodLineTwogrid.Rows[e.RowIndex].Cells[e.ColumnIndex]);
282
283             if ((btn.Value.ToString().Contains("TRANSPORT")) ||
                  (btn.Value.ToString().Contains("STASJON")))
284             {
285                 this.lineOneEvent = e.RowIndex;
286                 this.mouseDownLineOne = true;
287                 Thread eventThreadLineOne = new Thread(new ThreadStart
                      (mouseOnHoldLineOne));
288                 eventThreadLineOne.IsBackground = true;
289                 eventThreadLineOne.Start();
290             }
291         }
292
293
294         /// <summary>
295         /// Handling the transport buttons on production line one when button
              is released.
296         /// </summary>
297         /// <param name="sender"></param>
298         /// <param name="e"></param>
299         private void prodLineOneGrid_MouseUp(object sender,
              DataGridViewCellMouseEventArgs e)
300         {
301             DataGridViewButtonCell btn = (DataGridViewButtonCell)
                  (this.prodLineTwogrid.Rows[e.RowIndex].Cells[e.ColumnIndex]);
```

```
302
303             if ((btn.Value.ToString().Contains("TRANSPORT")) ||
                  (btn.Value.ToString().Contains("STASJON")))
304             {
305                 this.mouseDownLineOne = false;
306             }
307         }
308
309         /// <summary>
310         /// When a transport button for production line one is pressed
311         /// the stepper motor for that button moves.
312         /// </summary>
313         private void mouseOnHoldLineOne()
314         {
315             Stopwatch stopWatch = new Stopwatch();
316             stopWatch.Start();
317
318             while (this.mouseDownLineOne)
319             {
320                 this.isDone = false;
321                 if (stopWatch.ElapsedMilliseconds > 25)
322                 {
323                     this.lineOneBtns.ElementAt(this.lineOneEvent).Invoke();
324                     stopWatch.Reset();
325                     stopWatch.Start();
326                 }
327             }
328
329             this.isDone = true;
330             this.lineOneBtns.ElementAt(this.lineOneEvent).Invoke();
331             stopWatch.Stop();
332             this.lineOneEvent = -1;
333         }
334
335         /// <summary>
336         /// Handling the buttons on production line two except the transport
              related ones.
337         /// </summary>
338         /// <param name="sender"></param>
339         /// <param name="e"></param>
340         private void prodLineTwogrid_CellContentClick(object sender,
              DataGridViewCellEventArgs e)
341         {
342
343             DataGridViewButtonCell btn = null;
344             if ((e.ColumnIndex == this.prodLineTwogrid.Columns["Manuel
                  kjøring"].Index) && (e.RowIndex >= 0))
345             {
346                 btn = (DataGridViewButtonCell)(this.prodLineTwogrid.Rows
                      [e.RowIndex].Cells[e.ColumnIndex]);
347
348                 // This index is for the hold button events
349                 if((btn.Value.ToString().Contains("TRANSPORT")) ||
                      (btn.Value.ToString().Contains("STASJON")))
350                 {
351                     // Do nothing, the other eventhandler handles it
```

```
352                 }
353                 else
354                 {
355                     this.lineTwoBtns.ElementAt(e.RowIndex).Invoke();
356                 }
357 
358             }
359 
360         }
361 
362         /// <summary>
363         /// Handling the transport buttons on production line two when button is pressed.
364         /// </summary>
365         /// <param name="sender"></param>
366         /// <param name="e"></param>
367         private void prodLineTwoGrid_MouseDown(object sender, DataGridViewCellMouseEventArgs e)
368         {
369             DataGridViewButtonCell btn = (DataGridViewButtonCell)(this.prodLineTwogrid.Rows[e.RowIndex].Cells[e.ColumnIndex]);
370 
371             if ((btn.Value.ToString().Contains("TRANSPORT")) || (btn.Value.ToString().Contains("STASJON")))
372             {
373                 this.lineTwoEvent = e.RowIndex;
374                 this.mouseDownLineTwo = true;
375 
376                 Thread eventThreadLineTwo = new Thread(new ThreadStart(mouseOnHoldLineTwo));
377                 eventThreadLineTwo.IsBackground = true;
378                 eventThreadLineTwo.Start();
379             }
380         }
381 
382 
383         /// <summary>
384         /// Handling the transport buttons on production line two when button is released
385         /// </summary>
386         /// <param name="sender"></param>
387         /// <param name="e"></param>
388         private void prodLineTwoGrid_MouseUp(object sender, DataGridViewCellMouseEventArgs e)
389         {
390             DataGridViewButtonCell btn = (DataGridViewButtonCell)(this.prodLineTwogrid.Rows[e.RowIndex].Cells[e.ColumnIndex]);
391 
392             if ((btn.Value.ToString().Contains("TRANSPORT")) || (btn.Value.ToString().Contains("STASJON")))
393             {
394                 this.mouseDownLineTwo = false;
395             }
396         }
397 
398         /// <summary>
```

```
399         /// When a transport button for production line two is pressed
400         /// the stepper motor for that button moves.
401         /// </summary>
402         private void mouseOnHoldLineTwo()
403         {
404             Stopwatch stopWatch = new Stopwatch();
405             stopWatch.Start();
406 
407             while (this.mouseDownLineTwo)
408             {
409                 this.isDone = false;
410                 if(stopWatch.ElapsedMilliseconds > 25 )
411                 {
412                     this.lineTwoBtns.ElementAt(this.lineTwoEvent).Invoke();
413                     stopWatch.Reset();
414                     stopWatch.Start();
415                 }
416             }
417             this.isDone = true;
418 
419             this.lineTwoBtns.ElementAt(this.lineTwoEvent).Invoke();
420             stopWatch.Stop();
421             this.lineTwoEvent = -1;
422         }
423 
424         /// <summary>
425         /// Turn saw on.
426         /// </summary>
427         private void sawOn()
428         {
429             try
430             {
431                 String sawOn = "gvl.sawRun";
432                 this.ADScom.writeToBoolean(sawOn, true);
433             }
434             catch (AdsException e)
435             {
436                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
437                 String cpt = "KOMMUNIKASJONSFEIL";
438                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
439             }
440         }
441 
442         /// <summary>
443         /// Turn saw off.
444         /// </summary>
445         private void sawOff()
446         {
447             try
448             {
449                 String sawOff = "gvl.sawRun";
450                 this.ADScom.writeToBoolean(sawOff, false);
451             }
452             catch (AdsException e)
```

```
453             {
454                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
455                 String cpt = "KOMMUNIKASJONSFEIL";
456                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
457             }
458 
459         }
460 
461         /// <summary>
462         /// Not implemented
463         /// </summary>
464         private void lineOneMillStationDown(bool done)
465         {
466             // Implement functionality
467         }
468 
469         /// <summary>
470         /// Not implemented
471         /// </summary>
472         private void lineOneMillStationUp(bool done)
473         {
474             // Implement functionality
475         }
476 
477         /// <summary>
478         /// Not implemented
479         /// </summary>
480         private void lineOneMill()
481         {
482             // Implement functionality
483         }
484 
485         /// <summary>
486         /// Not implemented
487         /// </summary>
488         private void lineOneMillOff()
489         {
490             // Implement functionality
491         }
492 
493         /// <summary>
494         /// Not implemented
495         /// </summary>
496         private void lineOneMillOn()
497         {
498             // Implement functionality
499         }
500 
501         /// <summary>
502         /// Not implemented
503         /// </summary>
504         private void lineOneTransportOutMinus(bool done)
505         {
506             // Implement functionality
```

```
507         }
508
509         /// <summary>
510         /// Not implemented
511         /// </summary>
512         private void lineOneTransportOutPluss(bool done)
513         {
514             // Implement functionality
515         }
516
517         /// <summary>
518         /// Not implemented
519         /// </summary>
520         private void lineOneTransportInnMinus(bool done)
521         {
522             // Implement functionality
523         }
524
525         /// <summary>
526         /// Not implemented
527         /// </summary>
528         private void lineOneTransportInnPluss(bool done)
529         {
530             // Implement functionality
531         }
532
533         /// <summary>
534         /// Not implemented
535         /// </summary>
536         private void lineOneHolderThree()
537         {
538             // Implement functionality
539         }
540
541         /// <summary>
542         /// Not implemented
543         /// </summary>
544         private void lineOneHolderTwo()
545         {
546             // Implement functionality
547         }
548
549         /// <summary>
550         /// Not implemented
551         /// </summary>
552         private void lineOneHolderOne()
553         {
554             // Implement functionality
555         }
556
557         /// <summary>
558         /// Not implemented
559         /// </summary>
560         private void lineOneGripperTwo()
561         {
562             // Implement functionality
```

```
563         }
564
565         /// <summary>
566         /// Not implemented
567         /// </summary>
568         private void lineOneGripperOne()
569         {
570             // Implement functionality
571         }
572
573         /// <summary>
574         /// Production line two, move hole station down.
575         /// </summary>
576         private void lineTwoMillStationDown(bool done)
577         {
578             String transport = "Prod_Line_Two.millJogDown";
579
580             try
581             {
582                 if (done)
583                 {
584                     this.ADScom.writeToBoolean(transport, false);
585                 }
586                 else
587                 {
588                     this.ADScom.writeToBoolean(transport, true);
589                 }
590             }
591             catch(AdsException e)
592             {
593                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
594                 String cpt = "KOMMUNIKASJONSFEIL";
595                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
596             }
597         }
598
599         /// <summary>
600         /// Production line two, move hole station up.
601         /// </summary>
602         private void lineTwoMillStationUp(bool done)
603         {
604             String transport = "Prod_Line_Two.millJogUp";
605             try
606             {
607                 if (done)
608                 {
609                     this.ADScom.writeToBoolean(transport, false);
610                 }
611                 else
612                 {
613                     this.ADScom.writeToBoolean(transport, true);
614                 }
615             }
616             catch (AdsException e)
```

```
617             {
618                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
619                 String cpt = "KOMMUNIKASJONSFEIL";
620                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
621             }
622 
623         }
624 
625         /// <summary>
626         /// Production line two, drill hole.
627         /// </summary>
628         private void lineTwoMill()
629         {
630             String mill = "gvl.millYMoveL2";
631             try
632             {
633                 if (this.ADScom.readBoolean(mill))
634                 {
635                     this.ADScom.writeToBoolean(mill, false);
636                 }
637                 else
638                 {
639                     this.ADScom.writeToBoolean(mill, true);
640                 }
641             }
642 
643             catch (AdsException e)
644             {
645                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
646                 String cpt = "KOMMUNIKASJONSFEIL";
647                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
648             }
649         }
650 
651         /// <summary>
652         /// Production line two, turn drill off.
653         /// </summary>
654         private void lineTwoMillOff()
655         {
656             String millOff = "gvl.millRunL2";
657 
658             try
659             {
660                 this.ADScom.writeToBoolean(millOff, false);
661             }
662             catch (AdsException e)
663             {
664                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
665                 String cpt = "KOMMUNIKASJONSFEIL";
666                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
667             }
668 
669         }
670 
671         /// <summary>
672         /// Production line two, turn drill on.
673         /// </summary>
674         private void lineTwoMillOn()
675         {
676             try
677             {
678                 String millOn = "gvl.millRunL2";
679                 this.ADScom.writeToBoolean(millOn, true);
680             }
681             catch (AdsException e)
682             {
683                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom server og PLS:" + "\n\n" + e.Message;
684                 String cpt = "KOMMUNIKASJONSFEIL";
685                 DialogResult result = MessageBox.Show(txt, cpt, MessageBoxButtons.OK, MessageBoxIcon.Error);
686             }
687         }
688 
689         /// <summary>
690         /// Production line two, Move the output gripper inn the negative direction.
691         /// </summary>
692         private void lineTwoTransportOutMinus(bool done)
693         {
694             String transport = "Prod_Line_Two.outputJogBack";
695 
696             try
697             {
698                 if (done)
699                 {
700                     this.ADScom.writeToBoolean(transport, false);
701                 }
702                 else
703                 {
704                     this.ADScom.writeToBoolean(transport, true);
705                 }
706             }
707             catch (AdsException e)
708             {
709                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom server og PLS:" + "\n\n" + e.Message;
710                 String cpt = "KOMMUNIKASJONSFEIL";
711                 DialogResult result = MessageBox.Show(txt, cpt, MessageBoxButtons.OK, MessageBoxIcon.Error);
712             }
713         }
714 
715         /// <summary>
716         /// Production line two, Move the output gripper inn the positive direction.
```

```
717         /// </summary>
718         private void lineTwoTransportOutPluss(bool done)
719         {
720             String transport = "Prod_Line_Two.outputJogFwd";
721 
722             try
723             {
724                 if (done)
725                 {
726                     this.ADScom.writeToBoolean(transport, false);
727                 }
728                 else
729                 {
730                     this.ADScom.writeToBoolean(transport, true);
731                 }
732             }
733             catch (AdsException e)
734             {
735                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
736                 String cpt = "KOMMUNIKASJONSFEIL";
737                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
738             }
739 
740 
741         }
742 
743         /// <summary>
744         /// Production line two, Move the input gripper inn the negative
              direction.
745         /// </summary>
746         private void lineTwoTransportInnMinus(bool done)
747         {
748             String transport = "Prod_Line_Two.feedJogBack";
749 
750             try
751             {
752                 if (done)
753                 {
754                     this.ADScom.writeToBoolean(transport, false);
755                 }
756                 else
757                 {
758                     this.ADScom.writeToBoolean(transport, true);
759                 }
760             }
761             catch (AdsException e)
762             {
763                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
764                 String cpt = "KOMMUNIKASJONSFEIL";
765                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
766             }
767         }
```

```
768
769         /// <summary>
770         /// Production line two, Move the input gripper inn the positive
              direction.
771         /// </summary>
772         private void lineTwoTransportInnPluss(bool done)
773         {
774             String transport = "Prod_Line_Two.feedJogFwd";
775
776             try
777             {
778                 if (done)
779                 {
780                     this.ADScom.writeToBoolean(transport, false);
781                 }
782                 else
783                 {
784                     this.ADScom.writeToBoolean(transport, true);
785                 }
786             }
787             catch (AdsException e)
788             {
789                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                    server og PLS:" + "\n\n" + e.Message;
790                 String cpt = "KOMMUNIKASJONSFEIL";
791                 DialogResult result = MessageBox.Show(txt, cpt,
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
792             }
793         }
794
795         /// <summary>
796         /// Production line two, activeate or deactivate profile holder three.
797         /// </summary>
798         private void lineTwoHolderThree()
799         {
800             String holder = "gvl.lockStation3L2";
801
802             try
803             {
804                 if (this.ADScom.readBoolean(holder))
805                 {
806                     this.ADScom.writeToBoolean(holder, false);
807                 }
808                 else
809                 {
810                     this.ADScom.writeToBoolean(holder, true);
811                 }
812             }
813             catch (AdsException e)
814             {
815                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                    server og PLS:" + "\n\n" + e.Message;
816                 String cpt = "KOMMUNIKASJONSFEIL";
817                 DialogResult result = MessageBox.Show(txt, cpt,
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
818             }
```

```
819         }
820
821         /// <summary>
822         /// Production line two, activeate or deactivate profile holder two.
823         /// </summary>
824         private void lineTwoHolderTwo()
825         {
826             String holder = "gvl.lockStation2L2";
827
828             try
829             {
830                 if (this.ADScom.readBoolean(holder))
831                 {
832                     this.ADScom.writeToBoolean(holder, false);
833                 }
834                 else
835                 {
836                     this.ADScom.writeToBoolean(holder, true);
837                 }
838             }
839
840             catch (AdsException e)
841             {
842                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
843                 String cpt = "KOMMUNIKASJONSFEIL";
844                 DialogResult result = MessageBox.Show(txt, cpt,
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
845             }
846
847         }
848
849         /// <summary>
850         /// Production line two, activeate or deactivate profile holder one.
851         /// </summary>
852         private void lineTwoHolderOne()
853         {
854             String holder = "gvl.lockStation1L2";
855
856             try
857             {
858                 if (this.ADScom.readBoolean(holder))
859                 {
860                     this.ADScom.writeToBoolean(holder, false);
861                 }
862                 else
863                 {
864                     this.ADScom.writeToBoolean(holder, true);
865                 }
866             }
867             catch (AdsException e)
868             {
869                 String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                      server og PLS:" + "\n\n" + e.Message;
870                 String cpt = "KOMMUNIKASJONSFEIL";
871                 DialogResult result = MessageBox.Show(txt, cpt,
```

```
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        /// <summary>
        /// Production line two, activate or deactivate output gripper.
        /// </summary>
        private void lineTwoGripperTwo()
        {
            String gripper = "gvl.gripper2L2";

            try
            {
                if (this.ADScom.readBoolean(gripper))
                {
                    this.ADScom.writeToBoolean(gripper, false);
                }
                else
                {
                    this.ADScom.writeToBoolean(gripper, true);
                }
            }
            catch (AdsException e)
            {
                String txt = "Noe gikk feil med ADS kommunikasjonen mellom server og PLS:" + "\n\n" + e.Message;
                String cpt = "KOMMUNIKASJONSFEIL";
                DialogResult result = MessageBox.Show(txt, cpt, MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        /// <summary>
        /// Production line two, activate or deactivate input gripper.
        /// </summary>
        private void lineTwoGripperOne()
        {
            String gripper = "gvl.gripper1L2";

            try
            {
                if (this.ADScom.readBoolean(gripper))
                {
                    this.ADScom.writeToBoolean(gripper, false);
                }
                else
                {
                    this.ADScom.writeToBoolean(gripper, true);
                }
            }
            catch (AdsException e)
            {
                String txt = "Noe gikk feil med ADS kommunikasjonen mellom server og PLS:" + "\n\n" + e.Message;
                String cpt = "KOMMUNIKASJONSFEIL";
                DialogResult result = MessageBox.Show(txt, cpt,
```

```
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        /// <summary>
        /// Close ManuelMode window using the exit button
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void exitBtn_Click(object sender, EventArgs e)
        {
            String manuelMode = "gvl.manual";
            try
            {
                this.ADScom.writeToBoolean(manuelMode, false);
                this.Close();
            }

            catch (AdsException err)
            {
                String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                  server og PLS:" + "\n\n" + err.Message;
                String cpt = "KOMMUNIKASJONSFEIL";
                DialogResult result = MessageBox.Show(txt, cpt,
                  MessageBoxButtons.OK, MessageBoxIcon.Error);
                this.Close();
            }
        }

        /// <summary>
        ///  Close ManuelMode window
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void manuel_Closing(object sender, FormClosingEventArgs e)
        {
            String manuelMode = "gvl.manual";
            try
            {
                this.ADScom.writeToBoolean(manuelMode, false);
            }

            catch (AdsException err)
            {
                String txt = "Noe gikk feil med ADS kommunikasjonen mellom
                  server og PLS:" + "\n\n" + err.Message;
                String cpt = "KOMMUNIKASJONSFEIL";
                DialogResult result = MessageBox.Show(txt, cpt,
                  MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```

```
1  using RUFOServer.Program.Main;
2  using System;
3  using System.Collections.Generic;
4  using System.ComponentModel;
5  using System.Data;
6  using System.Drawing;
7  using System.Globalization;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11 using System.Windows.Forms;
12
13 namespace RUFOServer.Main
14 {
15     /// <summary>
16     /// The DetailsDialog is used to show production details from both
         production lines.
17     /// </summary>
18     public partial class DetailsDialog : Form
19     {
20         private AluminiumProfile profile;
21         private NumberFormatInfo numberFormat;
22         public DetailsDialog(AluminiumProfile profile)
23         {
24             InitializeComponent();
25
26             this.numberFormat = new NumberFormatInfo();
27             this.numberFormat.NumberDecimalSeparator = ".";
28             this.numberFormat.NegativeSign = "-";
29
30             setupDataGrid();
31             this.profile = profile;
32             fillDataGrid();
33
34         }
35
36         /// <summary>
37         /// Initializng the DataGridBView containing the production information
38         /// </summary>
39         private void setupDataGrid()
40         {
41             this.detailsGridView.ColumnCount = 4;
42             this.detailsGridView.ReadOnly = true;
43             this.detailsGridView.MultiSelect = false;
44             this.detailsGridView.RowHeadersVisible = false;
45             this.detailsGridView.Columns[0].Name = "Produksjons ordre nummer";
46             this.detailsGridView.Columns[0].SortMode =
                 DataGridViewColumnSortMode.NotSortable;
47             this.detailsGridView.Columns[1].Name = "Produkt nummer";
48             this.detailsGridView.Columns[1].SortMode =
                 DataGridViewColumnSortMode.NotSortable;
49             this.detailsGridView.Columns[2].Name = "Item nummer";
50             this.detailsGridView.Columns[2].SortMode =
                 DataGridViewColumnSortMode.NotSortable;
51             this.detailsGridView.Columns[3].Name = "Profil lengde";
52             this.detailsGridView.Columns[3].SortMode =
```

```
                DataGridViewColumnSortMode.NotSortable;
53              this.detailsGridView.RowTemplate.Height = 40;
54              this.detailsGridView.AutoSizeColumnsMode =
                  DataGridViewAutoSizeColumnsMode.Fill;
55          }
56
57          /// <summary>
58          /// Filling the DataGridBView with production information
59          /// </summary>
60          private void fillDataGrid()
61          {
62              foreach (OrderedAluProfile p in this.profile.getSortedAluList
                  ().getList())
63              {
64                  String[] data = new String[] { p.getOrderNmb(), p.getProdNmb(),
                       p.getItemNmb(), p.getLength().ToString(this.numberFormat) };
65                  this.detailsGridView.Rows.Add(data);
66              }
67          }
68
69          /// <summary>
70          /// Closing the details dialog
71          /// </summary>
72          /// <param name="sender"></param>
73          /// <param name="e"></param>
74          private void exitBtn_Click(object sender, EventArgs e)
75          {
76              this.Close();
77          }
78      }
79  }
80
```

```
1  using RUFOServer.Program.Main;
2  using RUFOServer.Program.Register;
3  using RUFOServer.Program.Server;
4  using System;
5  using System.Collections.Generic;
6  using System.ComponentModel;
7  using System.Data;
8  using System.Drawing;
9  using System.Globalization;
10 using System.IO;
11 using System.Linq;
12 using System.Text;
13 using System.Threading.Tasks;
14 using System.Windows.Forms;
15
16 namespace RUFOServer.Main
17 {
18     /// <summary>
19     /// This class is the dialog used to manipulate the Config.txt file,
20     /// which contains the production cell and production optimizing
       variables.
21     /// </summary>
22     public partial class ConfigDialog : Form
23     {
24         private string path;
25         private ConfigLoader configFile;
26         private OrderHandler orderHandler;
27         private NumberFormatInfo numberFormat;
28         private CutShaving cutShaving;
29
30         /// <summary>
31         /// Creating an instance of the ConfigDialog.
32         /// </summary>
33         /// <param name="configFile">
34         /// An object containing the information from the Config.txt file
35         /// </param>
36         /// <param name="orderHandler">
37         /// An orderHandler object
38         /// </param>
39         /// <param name="cutShaving">
40         /// An CutShaving object
41         /// </param>
42         public ConfigDialog(ConfigLoader configFile, OrderHandler
           orderHandler, CutShaving cutShaving)
43         {
44             InitializeComponent();
45             this.configFile = configFile;
46             this.orderHandler = orderHandler;
47             this.cutShaving = cutShaving;
48
49             // Ruleset for parsing double values from the string
50             this.numberFormat = new NumberFormatInfo();
51             this.numberFormat.NumberDecimalSeparator = ".";
52             this.numberFormat.NegativeSign = "-";
53
54             setupTxtBoxes();
```

```
55              updateTxtBoxes();
56              setupLabels();
57              getFilePath();
58              setupBtns();
59          }
60
61          /// <summary>
62          /// Setting up the labels
63          /// </summary>
64          private void setupLabels()
65          {
66
67              // Blade thickness
68              this.bladeDescLabel.Text = "Dette feltet forteller systemet om
                  tykkelsen på sagbladet.\n"+
69                  "Denne informasjonen må endres hver gang sagbladet blir
                      byttet,\n" +
70                  "hvis tykkelsen på bladene er forskjellige\n\n"+
71                  "NB! Husk at bladtykkelsen er i millimeter";
72              this.bladeThickLabel.Text = "BLAD TYKKELSE:";
73
74              // Cut Offset
75              this.cutOffsetDescLabel.Text = "Kappestasjon linje 1 og 2 er
                  avstanden mellom nullpunktet\n"+
76                  "på produksjonslinjen til kappestasjonen.\n\n" +
77                  "NB! Husk at denne avstanden er i millimeter.";
78              this.cutOffsetOneLabel.Text = "KAPPESTASJON LINJE 1:";
79              this.cutOffsetTwoLabel.Text = "KAPPESTASJON LINJE 2:";
80
81              // Mill offset
82              this.millOffsetDescLabel.Text = "Borrestasjon linje 1 og 2 er
                  avstanden mellom nullpunktet\n" +
83                  "på produksjonslinjen til borrestasjonen.\n\n" +
84                  "NB! Husk at denne avstanden er i millimeter.";
85              this.millOffsetOneLabel.Text = "BORRESTASJON LINJE 1:";
86              this.millOffsetTwoLabel.Text = "BORRESTASJON LINJE 2:";
87
88              // Profile length
89              this.profileLengthDescLabel.Text = "Profillengden er lengden på
                  profilen som blir matet inn" +
90                  "i maskinen.\n\n" +
91                  "NB! Endringer her vil ikke forandre de bestillingene\n" +
92                  "som allerede ligger inne i systemet.";
93              this.profileLengthOneLabel.Text = "PROFILLENGDE LINJE 1";
94              this.profileLengthTwoLabel.Text = "PROFILLENGDE LINJE 2";
95
96              // Max travel
97              this.maxTravelDescLabel.Text = "Avstanden innmatningen kan bevege
                  seg fra 0 punktet" + "\n" +
98                  "til sluttpunktet i millimeter";
99              this.maxTravelLabel.Text = "MAX VANDRING LINJE 1 OG 2";
100
101         }
102
103         /// <summary>
104         /// Setting up the text boxes
```

```
105         /// </summary>
106         private void setupTxtBoxes()
107         {
108             // Blade thickness
109             this.bladeThickTxt.MaxLength = 5;
110             this.bladeThickTxt.Multiline = false;
111 
112             // Cut Offset
113             this.cutOffsetOneTxt.MaxLength = 5;
114             this.cutOffsetOneTxt.Multiline = false;
115             this.cutOffsetTwoTxt.MaxLength = 5;
116             this.cutOffsetTwoTxt.Multiline = false;
117 
118             // Mill offset
119             this.millOffsetOneTxt.MaxLength = 5;
120             this.millOffsetOneTxt.Multiline = false;
121             this.millOffsetTwoTxt.MaxLength = 5;
122             this.millOffsetTwoTxt.Multiline = false;
123 
124             // Profile length
125             this.profileLengthOneTxt.MaxLength = 5;
126             this.profileLengthOneTxt.Multiline = false;
127             this.profileLengthTwoTxt.MaxLength = 5;
128             this.profileLengthTwoTxt.Multiline = false;
129 
130             // Max travel
131             this.maxTravelDistInnTxt.MaxLength = 5;
132             this.maxTravelDistInnTxt.Multiline = false;
133         }
134 
135         /// <summary>
136         /// Updating the content of the textboxes
137         /// </summary>
138         private void updateTxtBoxes()
139         {
140             // Blade thickness
141             this.bladeThickTxt.Text = this.configFile.getValue
                  ("Blade").ToString(this.numberFormat);
142 
143             // Cut Offset
144             this.cutOffsetOneTxt.Text = this.configFile.getValue
                  ("CutOffsetLineOne").ToString(this.numberFormat);
145             this.cutOffsetTwoTxt.Text = this.configFile.getValue
                  ("CutOffsetLineTwo").ToString(this.numberFormat);
146 
147             // Mill offset
148             this.millOffsetOneTxt.Text = this.configFile.getValue
                  ("MillOffsetLineOne").ToString(this.numberFormat);
149             this.millOffsetTwoTxt.Text = this.configFile.getValue
                  ("MillOffsetLineTwo").ToString(this.numberFormat);
150 
151             // Profile length
152             this.profileLengthOneTxt.Text = this.configFile.getValue
                  ("ProfileLengthOne").ToString(this.numberFormat);
153             this.profileLengthTwoTxt.Text = this.configFile.getValue
                  ("ProfileLengthTwo").ToString(this.numberFormat);
```

```
154
155             // Max travel distance on feeding
156             this.maxTravelDistInnTxt.Text = this.configFile.getValue
                  ("MaxTravelDistInn").ToString(this.numberFormat);
157         }
158
159         /// <summary>
160         /// Default startup values for the buttons in this window
161         /// </summary>
162         private void setupBtns()
163         {
164             this.saveBtn.Enabled = false;
165         }
166
167         /// <summary>
168         /// Finding the filepath for the Config.txt file
169         /// </summary>
170         private void getFilePath()
171         {
172             // Finding the path of the program
173             this.path = Path.GetFullPath(Application.StartupPath);
174             // Going to folders back to find the original files
175             this.path = Path.GetFullPath(Path.Combine(this.path, @"..\..\"));
176             // Goind into the folder containing the original ProfileInfo.txt
177             this.path = this.path + @"Program\Register\Config.txt";
178         }
179
180         /// <summary>
181         /// Closes the window
182         /// </summary>
183         /// <param name="sender"></param>
184         /// <param name="e"></param>
185         private void exitBtn_Click(object sender, EventArgs e)
186         {
187             this.Close();
188         }
189
190         /// <summary>
191         /// Saving the new information to the config.txt file
192         /// and updating the object utilizing the config.txt file information.
193         /// </summary>
194         /// <param name="sender"></param>
195         /// <param name="e"></param>
196         private void saveBtn_Click(object sender, EventArgs e)
197         {
198             // If one of the text boxes are emtpy
199             if ((!this.bladeThickTxt.Text.Any()) || (!
                  this.cutOffsetOneTxt.Text.Any()) || (!
                  this.cutOffsetTwoTxt.Text.Any()) ||
200                 (!this.millOffsetOneTxt.Text.Any()) || (!
                    this.millOffsetTwoTxt.Text.Any()) || (!
                    this.profileLengthOneTxt.Text.Any()) ||
201                 (!this.profileLengthTwoTxt.Text.Any()))
202             {
203                 string txt = "Alle feltene må være fylt ut før du kan lagre";
204                 string caption = "FEIL";
```

```
205
206                 MessageBox.Show(txt, caption, MessageBoxButtons.OK,
                      MessageBoxIcon.Error);
207             }
208
209             else
210             {
211                 string txt = "Viktig! \n" +
212                     "En endring i denne filen,\n" +
213                     "vil kunne drastisk endre ytelsen til maskinen!\n" +
214                     "Er du sikker på at du vil lagre endringene ? ";
215                 string caption = "ADVARSEL";
216
217                 DialogResult result =  MessageBox.Show(txt, caption,
                      MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
218
219                 if(result == DialogResult.Yes)
220                 {
221                     string finalTxt = "Er du helt sikker ?";
222                     string final = "ADVARSEL";
223
224                     DialogResult finalResult = MessageBox.Show(finalTxt,
                        final, MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
225
226                     if (finalResult == DialogResult.Yes)
227                     {
228                         saveConfigFile();
229                         // Update ConfigLoader
230                         this.configFile.updateConfig();
231                         // Update txt box values
232                         this.updateTxtBoxes();
233                         // Update rest of the classes
234                         this.orderHandler.setProfileLengthLineOne
                        (this.configFile.getValue("ProfileLengthOne"));
235                         this.orderHandler.setProfileLengthLineTwo
                        (this.configFile.getValue("ProfileLengthTwo"));
236                         this.cutShaving.setBladeThickness
                        (this.configFile.getValue("Blade"));
237                         // Disable the save button
238                         this.saveBtn.Enabled = false;
239                     }
240                     else
241                     {
242                         // Update txt box values
243                         this.updateTxtBoxes();
244                     }
245                 }
246                 else
247                 {
248                     // Update txt box values
249                     this.updateTxtBoxes();
250                 }
251             }
252
253         }
254
```

```
255         /// <summary>
256         /// Writing to the config file, saving the changes
257         /// </summary>
258         private void saveConfigFile()
259         {
260             String[] lines = new String[this.configFile.getConfigFile
                ().Count];
261 
262             lines[0] = "Blade" + "," + this.bladeThickTxt.Text;
263             lines[1] = "CutOffsetLineOne" + "," + this.cutOffsetOneTxt.Text;
264             lines[2] = "CutOffsetLineTwo" + "," + this.cutOffsetTwoTxt.Text;
265             lines[3] = "MillOffsetLineOne" + "," + this.millOffsetOneTxt.Text;
266             lines[4] = "MillOffsetLineTwo" + "," + this.millOffsetTwoTxt.Text;
267             lines[5] = "ProfileLengthOne" + "," +
                  this.profileLengthOneTxt.Text;
268             lines[6] = "ProfileLengthTwo" + "," +
                  this.profileLengthTwoTxt.Text;
269             lines[7] = "MaxTravelDistInn" + "," +
                  this.maxTravelDistInnTxt.Text;
270 
271             // Write to file
272             File.WriteAllLines(this.path, lines);
273         }
274 
275         /// <summary>
276         /// If one of the textboxes text changes, the saave button is enabled
277         /// </summary>
278         /// <param name="sender"></param>
279         /// <param name="e"></param>
280         private void TextBoxes_TextChanged(object sender, EventArgs e)
281         {
282             this.saveBtn.Enabled = true;
283         }
284 
285         /// <summary>
286         /// The saw blade thickness
287         /// You can only enter digits and decimal with "." in the text boxe
288         /// </summary>
289         /// <param name="sender"></param>
290         /// <param name="e"></param>
291         private void bladeBox_OnKeyPressed(object sender, KeyPressEventArgs e)
292         {
293             char ch = e.KeyChar;
294 
295             // Is the input is not a number or 8 = backspace
296             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
297             {
298                 e.Handled = true;
299             }
300 
301             // If char = '.', make sure it is the only one
302             else if ((ch == 46) || (this.bladeThickTxt.TextLength == 0))
303             {
304                 if(ch == 46 && this.bladeThickTxt.TextLength == 0)
305                 {
306                     e.Handled = true;
```

```
307                 }
308
309                 else if (this.bladeThickTxt.Text.Contains('.'))
310                 {
311                     e.Handled = true;
312                 }
313             }
314         }
315
316         /// <summary>
317         /// The offset from the production cell origo to the cut station on
              production line one
318         /// You can only enter digits and decimal with "." in the text box
319         /// </summary>
320         /// <param name="sender"></param>
321         /// <param name="e"></param>
322         private void cutLineOneBox_OnKeyPressed(object sender,
              KeyPressEventArgs e)
323         {
324             char ch = e.KeyChar;
325
326             // Is the input is not a number or 8 = backspace
327             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
328             {
329                 e.Handled = true;
330             }
331
332             // If char = '.', make sure it is the only one
333             else if ((ch == 46) || (this.cutOffsetOneTxt.TextLength == 0))
334             {
335                 if (ch == 46 && this.cutOffsetOneTxt.TextLength == 0)
336                 {
337                     e.Handled = true;
338                 }
339
340                 else if (this.cutOffsetOneTxt.Text.Contains('.'))
341                 {
342                     e.Handled = true;
343                 }
344             }
345         }
346
347         /// <summary>
348         /// The offset from the production cell origo to the cut station on
              production line two
349         /// You can only enter digits and decimal with "." in the text box
350         /// </summary>
351         /// <param name="sender"></param>
352         /// <param name="e"></param>
353         private void cutLineTwoBox_OnKeyPressed(object sender,
              KeyPressEventArgs e)
354         {
355             char ch = e.KeyChar;
356
357             // Is the input is not a number or 8 = backspace
358             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
```

```
359             {
360                 e.Handled = true;
361             }
362 
363             // If char = '.', make sure it is the only one
364             else if ((ch == 46) || (this.cutOffsetTwoTxt.TextLength == 0))
365             {
366                 if (ch == 46 && this.cutOffsetTwoTxt.TextLength == 0)
367                 {
368                     e.Handled = true;
369                 }
370 
371                 else if (this.cutOffsetTwoTxt.Text.Contains('.'))
372                 {
373                     e.Handled = true;
374                 }
375             }
376         }
377 
378         /// <summary>
379         /// The offset from the production cell origo to the hole station on
              production line one
380         /// You can only enter digits and decimal with "." in the text box
381         /// </summary>
382         /// <param name="sender"></param>
383         /// <param name="e"></param>
384         private void millLineOneBox_OnKeyPressed(object sender,
              KeyPressEventArgs e)
385         {
386             char ch = e.KeyChar;
387 
388             // Is the input is not a number or 8 = backspace
389             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
390             {
391                 e.Handled = true;
392             }
393 
394             // If char = '.', make sure it is the only one
395             else if ((ch == 46) || (this.millOffsetOneTxt.TextLength == 0))
396             {
397                 if (ch == 46 && this.millOffsetOneTxt.TextLength == 0)
398                 {
399                     e.Handled = true;
400                 }
401 
402                 else if (this.millOffsetOneTxt.Text.Contains('.'))
403                 {
404                     e.Handled = true;
405                 }
406             }
407         }
408 
409         /// <summary>
410         /// The offset from the production cell origo to the hole station on
              production line two
411         /// You can only enter digits and decimal with "." in the text box
```

```
412         /// </summary>
413         /// <param name="sender"></param>
414         /// <param name="e"></param>
415         private void millLineTwoBox_OnKeyPressed(object sender,
              KeyPressEventArgs e)
416         {
417             char ch = e.KeyChar;
418 
419             // Is the input is not a number or 8 = backspace
420             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
421             {
422                 e.Handled = true;
423             }
424 
425             // If char = '.', make sure it is the only one
426             else if ((ch == 46) || (this.millOffsetTwoTxt.TextLength == 0))
427             {
428                 if (ch == 46 && this.millOffsetTwoTxt.TextLength == 0)
429                 {
430                     e.Handled = true;
431                 }
432 
433                 else if (this.millOffsetTwoTxt.Text.Contains('.'))
434                 {
435                     e.Handled = true;
436                 }
437             }
438         }
439 
440         /// <summary>
441         /// The max travel distance for the input gripper for both production
              lines
442         /// You can only enter digits and decimal with "." in the text box
443         /// </summary>
444         /// <param name="sender"></param>
445         /// <param name="e"></param>
446         private void maxTravelDistInnTxt_OnKeyPressed(object sender,
              KeyPressEventArgs e)
447         {
448             char ch = e.KeyChar;
449 
450             // Is the input is not a number or 8 = backspace
451             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
452             {
453                 e.Handled = true;
454             }
455 
456             // If char = '.', make sure it is the only one
457             else if ((ch == 46) || (this.maxTravelDistInnTxt.TextLength == 0))
458             {
459                 if (ch == 46 && this.maxTravelDistInnTxt.TextLength == 0)
460                 {
461                     e.Handled = true;
462                 }
463 
464                 else if (this.maxTravelDistInnTxt.Text.Contains('.'))
```

```
465                 {
466                     e.Handled = true;
467                 }
468             }
469         }
470 
471         /// <summary>
472         /// The profile length for production line one
473         /// You can only enter digits and decimal with "." in the text box
474         /// </summary>
475         /// <param name="sender"></param>
476         /// <param name="e"></param>
477         private void profileLengthLineOne_OnKeyPressed(object sender,
              KeyPressEventArgs e)
478         {
479             char ch = e.KeyChar;
480 
481             // Is the input is not a number or 8 = backspace
482             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
483             {
484                 e.Handled = true;
485             }
486 
487             // If char = '.', make sure it is the only one
488             else if ((ch == 46) || (this.profileLengthOneTxt.TextLength == 0))
489             {
490                 if (ch == 46 && this.profileLengthOneTxt.TextLength == 0)
491                 {
492                     e.Handled = true;
493                 }
494 
495                 else if (this.profileLengthOneTxt.Text.Contains('.'))
496                 {
497                     e.Handled = true;
498                 }
499             }
500         }
501 
502         /// <summary>
503         /// The profile length for production line two
504         /// You can only enter digits and decimal with "." in the text box
505         /// </summary>
506         /// <param name="sender"></param>
507         /// <param name="e"></param>
508         private void profileLengthLineTwo_OnKeyPressed(object sender,
              KeyPressEventArgs e)
509         {
510             char ch = e.KeyChar;
511 
512             // Is the input is not a number or 8 = backspace
513             if ((!Char.IsDigit(ch)) && (ch != 8) && (ch != 46))
514             {
515                 e.Handled = true;
516             }
517 
518             // If char = '.', make sure it is the only one
```

```
519             else if ((ch == 46) || (this.profileLengthTwoTxt.TextLength == 0))
520             {
521                 if (ch == 46 && this.profileLengthTwoTxt.TextLength == 0)
522                 {
523                     e.Handled = true;
524                 }
525 
526                 else if (this.profileLengthTwoTxt.Text.Contains('.'))
527                 {
528                     e.Handled = true;
529                 }
530             }
531         }
532     }
533 }
534 
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Entity
8  {
9      /// <summary>
10     /// This class represents a cut from 90 degres with +- 45 degres on that
       axis.
11     /// AngleCut means that the sawblade cuts the material from the side inn.
12     /// </summary>
13     public class AngleCut : Cut
14     {
15         private double angle;
16
17         /// <summary>
18         /// Creating an instance of AngleCut, which needs the position and
           angle of the cut.
19         /// </summary>
20         /// <param name="position">
21         /// The position of the cut in milimeters
22         /// </param>
23         /// <param name="angle">
24         /// The angle in degrees for the cut from the side.
25         /// </param>
26         public AngleCut(double position, double angle) : base(position)
27         {
28             this.angle = angle;
29
30         }
31
32         /// <summary>
33         /// Returning the cut angle in degrees
34         /// </summary>
35         /// <returns>
36         /// The cut angle in degrees
37         /// </returns>
38         public double getCutAngle()
39         {
40             return this.angle;
41         }
42     }
43 }
44
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Entity
8  {
9      /// <summary>
10     /// The BendCut holds the position for the cut, but the cut will not go
       straight through the material.
11     /// Which gives the opurtunity to bend metal into a 90 degree angle.
12     /// For example when cuting aluminium or other metal alloys which are
       bendable.
13     /// </summary>
14     public class BendCut : Cut
15     {
16         /// <summary>
17         /// Creating an instance of BendCut
18         /// </summary>
19         /// <param name="position">
20         /// The poition of the cut in milimeters.
21         /// </param>
22         public BendCut(double position) : base(position)
23         {
24         }
25     }
26 }
27
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program
8  {
9      /// <summary>
10     /// The cut class represents a cut from a saw, laser, scissor etc.
11     /// A cut holds the position for the cut.
12     /// </summary>
13     public class Cut
14     {
15         private double position; // In mm
16         private bool finalCut;
17
18         /// <summary>
19         /// Creating an instance of Cut, with the parameters position and angle
20         /// </summary>
21         ///
22         /// <param name="position">
23         /// The position of the cut in milimeters
24         /// </param>
25         public Cut(double position)
26         {
27             this.position = position;
28             this.finalCut = false;
29         }
30
31         /// <summary>
32         /// Returning the cut position
33         /// </summary>
34         /// <returns>
35         /// The cut position
36         /// </returns>
37         public double getCutPosition()
38         {
39             return this.position;
40         }
41
42         /// <summary>
43         /// Setting the position for the cut in millimeters
44         /// </summary>
45         /// <param name="position">
46         /// The position for the cut in millimeters
47         /// </param>
48         public void setCutPosition(double position)
49         {
50             this.position = position;
51         }
52
53         /// <summary>
54         /// Returning true if this cut is the final cut
55         /// </summary>
56         /// <returns>
```

```
57          /// True if this cut is the final cut
58          /// </returns>
59          public bool isFinalCut()
60          {
61              return this.finalCut;
62          }
63
64          /// <summary>
65          /// Setting if the cut is the final cut or not. True for final and
              false for not final
66          /// </summary>
67          /// <param name="finalCut">
68          /// The boolean value
69          /// </param>
70          public void setFinalCut(bool finalCut)
71          {
72              this.finalCut = finalCut;
73          }
74      }
75  }
76
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Entity
8  {
9      /// <summary>
10     /// Hole class represents a hole in a x&y coordinate system. Which can be
         aplied to any 2D surface.
11     /// The hole can be made with a drill, punching or milling.
12     /// </summary>
13     public class Hole
14     {
15         private double xAxis; // In mm
16         private double yAxis; // In mm
17         private String zPosition;
18
19         /// <summary>
20         /// To create an instance of hole, input the x and y axis in
             millimeter.
21         /// This is were the hole will be created in the 2D surface.
22         /// </summary>
23         /// <param name="xAxis">
24         /// Position of the hole in the x-axis in mm
25         /// </param>
26         /// <param name="yAxis">
27         /// Position of the hole in the y-axis in mm
28         /// </param>
29         public Hole(double xAxis, double yAxis)
30         {
31             this.xAxis = xAxis;
32             this.yAxis = yAxis;
33
34             // zPosition default value
35             this.zPosition = null;
36         }
37
38         /// <summary>
39         /// To create an instance of hole, input the x and y axis in
             millimeter.
40         /// This is were the hole will be created in the 2D surface.
41         /// The zPosition represents a third position of your choice,
42         /// this position could that an angular surface has to sides,
43         /// so zPosition = "B" represents one side and zPosition = "A"
             represents the other.
44         /// </summary>
45         /// <param name="zPosition"></param>
46         /// The third position of the hole. this position could that an
             angular surface has to sides,
47         /// so zPosition = "B" represents one side and zPosition = "A"
             represents the other.
48         /// <param name="xAxis">
49         /// Position of the hole in the x-axis in mm
50         /// </param>
```

```
51          /// <param name="yAxis">
52          /// Position of the hole in the y-axis in mm
53          /// </param>
54          public Hole(String zPosition, double xAxis, double yAxis)
55          {
56              this.xAxis = xAxis;
57              this.yAxis = yAxis;
58              this.zPosition = zPosition;
59          }
60
61          /// <summary>
62          /// Returning the zPosition as a String. If the hole has no zPosition
              null is returned
63          /// </summary>
64          /// <returns>
65          /// The zPosition as a String.
66          /// </returns>
67          public String getZPosition()
68          {
69              return this.zPosition;
70          }
71
72          /// <summary>
73          /// Returning the hole position x-axis in mm
74          /// </summary>
75          /// <returns>
76          /// The hole position x-axis in mm
77          /// </returns>
78          public double getXAxis()
79          {
80              return this.xAxis;
81          }
82
83          /// <summary>
84          /// Returning the hole position y-axis in mm
85          /// </summary>
86          /// <returns>
87          /// The hole position y-axis in mm
88          /// </returns>
89          public double getYAxis()
90          {
91              return this.yAxis;
92          }
93
94          /// <summary>
95          /// Setting the x-axis for the hole in millimeters
96          /// </summary>
97          /// <param name="position">
98          /// The x-axis position in millimeters
99          /// </param>
100         public void setXAxis(double position)
101         {
102             this.xAxis = position;
103         }
104     }
105 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Entity
8  {
9      /// <summary>
10     /// This class represents a cut from 90degres with +- 45 degres on that
       axis.
11     /// RotationCut means that the sawblade cuts the material from the top down
12     /// </summary>
13     public class RotationCut : Cut
14     {
15         private double angle;
16
17         /// <summary>
18         /// Creating an instance of RotationCut, which needs the position and
           angle of the cut.
19         /// </summary>
20         /// <param name="position">
21         /// The position of the cut in milimeters
22         /// </param>
23         /// <param name="angle">
24         /// The angle in degrees for the cut from the top down.
25         /// </param>
26         public RotationCut(double position, double angle) : base(position)
27         {
28             this.angle = angle;
29         }
30
31         /// <summary>
32         /// Returning the cut angle in degrees
33         /// </summary>
34         /// <returns>
35         /// The cut angle in degrees
36         /// </returns>
37         public double getCutAngle()
38         {
39             return this.angle;
40         }
41     }
42 }
43
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Entity
8  {
9      /// <summary>
10     /// The TrimCut class represents a cut performed to trim a part of the material away.
11     /// Usually used to trim away the top or bottom of the material in a +- 45 degree angle after a 90 degree angle cut is perfomred
12     /// </summary>
13     public class TrimCut : Cut
14     {
15         private double heightPos; // IN MM
16         private double rotation; // IN DEGREE
17         /// <summary>
18         /// Creating an instance of TrimCut
19         /// </summary>
20         ///
21         /// <param name="position">
22         /// The position of the trim cut in milimeters
23         /// </param>
24         public TrimCut(double position) : base(position)
25         {
26             this.heightPos = 10;
27             this.rotation = 45;
28         }
29
30         /// <summary>
31         ///  Creating an instance of TrimCut
32         /// </summary>
33         ///
34         /// <param name="position">
35         /// The position of the trim cut in milimeters
36         /// </param>
37         /// <param name="height">
38         /// The height position of the trim cut in milimeters
39         /// </param>
40         /// <param name="rotation">
41         /// The rotation angle of the TrimCut in degrees
42         /// </param>
43         public TrimCut(double position, double height, double rotation) : base(position)
44         {
45             this.heightPos = height;
46             this.rotation = rotation;
47         }
48
49         /// <summary>
50         /// Returning the height position of the cut in millimeters
51         /// </summary>
52         /// <returns>
53         /// The height position of the cut in millimeters
```

```
54          /// </returns>
55          public double getHeightPos()
56          {
57              return this.heightPos;
58          }
59
60          /// <summary>
61          /// Returning the TrimCut rotation angle in degrees
62          /// </summary>
63          /// <returns>
64          /// The TrimCut rotation angle in degrees
65          /// </returns>
66          public double getRotation()
67          {
68              return this.rotation;
69          }
70      }
71  }
72
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Main
8  {
9
10     /// <summary>
11     /// The AluminiumProfile class represents an un processed aluminium
         profile of length 0 to 6000 mm.
12     /// The class holds a register/list of the class OrderedAluList with
         OrderedAluProfile.
13     /// The point of this, is that the AluminiumProfile can hold
         OrderedAluList up to it's own length.
14     /// For example: If the AluminiumProfile is 6000mm the register can hold 3
          objects of OrderedAluProfile with the length 2000mm,
15     /// because 2000mm * 3 = 6000mm. Or you can have many objects with shorter
          lengths.
16     /// </summary>
17     public class AluminiumProfile
18     {
19         private double length; // In mm
20         private double startWaste; // In mm
21         private double width; // In mm
22         private double height; // In mm
23         private const double endWaste = 10; // In mm
24         private String type;
25         private OrderedAluList orderedProfiles;
26
27         /// <summary>
28         /// Creating an instance of AluminiumProfile. The maximum aloved
             length is 6000mm,
29         /// if the input is larger the length will be sett to 6000mm.
30         /// </summary>
31         /// <param name="type">
32         /// Type of aluminium profile
33         /// </param>
34         /// <param name="length">
35         /// Length of the profile in millimeters. Maximum is 6000mm.
36         /// </param>
37         /// <param name="width">
38         /// The width of the profile in millimeters
39         /// </param>
40         /// <param name="height">
41         /// The height of the profile in millimeters
42         /// </param>
43         public AluminiumProfile(string type, double length, double width,
             double height)
44         {
45             if (length > 6000)
46             {
47                 this.length = 6000;
48             }
49
```

```
50              else
51              {
52                  this.length = length;
53              }
54              this.type = type;
55              this.startWaste = this.length - endWaste; // milimeters so we have a rest at the end of the profile
56              this.width = width;
57              this.height = height;
58              this.orderedProfiles = new OrderedAluList();
59          }
60
61          /// <summary>
62          /// Returning the EndWaste in millimeters
63          /// </summary>
64          /// <returns>
65          /// The EndWaste in millimeters
66          /// </returns>
67          public static double getEndWaste()
68          {
69              return endWaste;
70          }
71
72          /// <summary>
73          /// Returning the full length of the profile in millimeters
74          /// </summary>
75          /// <returns>
76          /// The full length of the profile in millimeters
77          /// </returns>
78          public double getFullLength()
79          {
80              return this.length;
81          }
82
83          /// <summary>
84          /// Returning the calculated waste that is left after OrderedAluProfile has been added to the register.
85          /// In millimeters
86          /// </summary>
87          /// <returns>
88          /// The calculated waste that is left after OrderedAluProfile has been added to the register.
89          /// In millimeters
90          /// </returns>
91          public double getWaste()
92          {
93              return Math.Round(this.startWaste, 1, MidpointRounding.AwayFromZero) + endWaste;
94          }
95
96          /// <summary>
97          /// Returning the calculated waste that is left aat the start of the aluminium profile
98          /// </summary>
99          /// <returns>
100         /// The calculated waste that is left at the start of the aluminium
```

```
            profile
101         /// </returns>
102         public double getStartWaste()
103         {
104             // Round start waste to one decimals
105             return Math.Round(this.startWaste, 2,
                  MidpointRounding.AwayFromZero);
106         }
107
108         /// <summary>
109         /// Returning the aluminium profile type.
110         /// </summary>
111         /// <returns>
112         /// The aluminium profile type.
113         /// </returns>
114         public String getType()
115         {
116             return this.type;
117         }
118
119         /// <summary>
120         /// Returning the height of the profile in milimeters
121         /// </summary>
122         /// <returns>
123         /// The height of the profile in milimeters
124         /// </returns>
125         public double getHeight()
126         {
127             return this.height;
128         }
129
130         /// <summary>
131         /// Returning the width of the profile in milimeters
132         /// </summary>
133         /// <returns>
134         /// The width of the profile in milimeters
135         /// </returns>
136         public double getWidth()
137         {
138             return this.width;
139         }
140
141         /// <summary>
142         /// Returning the OrderedAluList containing the ordered profiles
143         /// </summary>
144         /// <returns>
145         /// The OrderedAluList containing the ordered profiles
146         /// </returns>
147         public OrderedAluList getOrderedAluList()
148         {
149             return this.orderedProfiles;
150         }
151
152         /// <summary>
153         /// Adding a OrderedAluProfile to the AluminiumProfile's register.
154         /// But the profile will not be added if the length of the profile you
```

```
            want to add,
155         /// exceeds the lenght that is left on the aluminium profile
              (startWaste).
156         /// </summary>
157         /// <param name="profile">
158         /// An object of the type OrderedAluProfile.
159         /// </param>
160         public void addOrderedAluProfile(OrderedAluProfile profile)
161         {
162             if(this.startWaste > profile.getLength())
163             {
164                 this.orderedProfiles.addProfile(profile);
165             }
166
167             else
168             {
169                 // Do nothing
170             }
171
172             this.startWaste = startWaste - profile.getLength();
173         }
174
175         /// <summary>
176         /// Adding a OrderedAluProfile to the AluminiumProfile's register with
              the calculated cut shaving.
177         /// But the profile will not be added if the length of the profile you
              want to add,
178         /// exceeds the lenght that is left on the aluminium profile
              (startWaste).
179         /// </summary>
180         /// <param name="profile"></param>
181         /// <param name="cutShaving"></param>
182         public void addProfileWithShaving(OrderedAluProfile profile, double
              cutShaving)
183         {
184             if (this.startWaste > profile.getLength())
185             {
186                 this.orderedProfiles.addProfile(profile);
187             }
188
189             else
190             {
191                 // Do nothing
192             }
193             this.startWaste = startWaste - profile.getLength() - cutShaving;
194         }
195
196         /// <summary>
197         /// Sorting the OrderedAluList's list by accending order before
              returning the OrderedAluList object which contains that list
198         /// </summary>
199         /// <returns>
200         /// The OrderedAluList object with its internal list sorted by
              accending order
201         /// </returns>
202         public OrderedAluList getSortedAluList()
```

```
203         {
204             this.orderedProfiles.getList().Sort((pA, pB) =>
205             {
206                 var firstCompare = pA.getLength().CompareTo(pB.getLength());
207                 return firstCompare;
208             });
209 
210             return this.orderedProfiles;
211         }
212     }
213 }
214 
215 
```

```
1  using RUFOServer.Program.Entity;
2  using RUFOServer.Program.Register;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9
10 namespace RUFOServer.Program.Main
11 {
12     /// <summary>
13     /// The AluProductionOptimizer class, represents an cut optimizing
       algorithm.
14     /// Which uses this algorithm to optimize the production order, and splits
       this order into two segments.
15     /// One list for the AngleProfiles and one for the GuidingProfiles
16     /// </summary>
17     public class AluProductionOptimizer
18     {
19         private AluProfileOrderList angleOrderList;
20         private AluProfileOrderList guidingOrderList;
21         private AluminiumTypeRegister typeReg;
22         private AluminiumDimRegister dimReg;
23         private CutShaving cutShaving;
24         private double profileLengthLineOne; // In millimeters
25         private double profileLengthLineTwo; // In millimeters
26
27         /// <summary>
28         /// Creating an instance of AluProductionOptimizer
29         /// With a default value to profileLengthLineOne and
             profileLengthLineTwo set to 4000mm
30         /// </summary>
31         /// <param name="angleOrderList">
32         /// An AluProfileOrderList containing just AngleProfile objects
33         /// </param>
34         /// <param name="guidingOrderList">
35         /// An AluProfileOrderList containing just GuidingProfile objects
36         /// </param>
37         /// <param name="typeReg">
38         /// A list containing all the types of profiles the system can handle
39         /// </param>
40         /// <param name="dimReg">
41         /// A list containing the dimmensions of the profiles the system can
             handle
42         /// </param>
43         /// <param name="cutShaving">
44         /// A cutShaving object
45         /// </param>
46         public AluProductionOptimizer(AluProfileOrderList angleOrderList,
             AluProfileOrderList guidingOrderList, AluminiumTypeRegister typeReg,
              AluminiumDimRegister dimReg, CutShaving cutShaving)
47         {
48             this.angleOrderList = angleOrderList;
49             this.guidingOrderList = guidingOrderList;
50             this.typeReg = typeReg;
```

```
51              this.dimReg = dimReg;
52              this.cutShaving = cutShaving;
53
54              //Default value
55              this.profileLengthLineOne = 4000;
56              this.profileLengthLineTwo = 4000;
57          }
58
59          /// <summary>
60          /// Setting the length of the profiles which can run on production
              line one
61          /// </summary>
62          /// <param name="length">
63          /// The length in millimeters
64          /// </param>
65          public void setProfileLengthLineOne(double length)
66          {
67              this.profileLengthLineOne = length;
68          }
69
70          /// <summary>
71          /// Returning the length of the profiles which can run on production
              line one
72          /// </summary>
73          /// <returns>
74          /// The length of the profiles which can run on production line one
75          /// </returns>
76          public double getProfileLengthLineOne()
77          {
78              return this.profileLengthLineOne;
79          }
80
81          /// <summary>
82          /// Setting the length of the profiles which can run on production
              line two
83          /// </summary>
84          /// <param name="length">
85          /// The length in millimeters
86          /// </param>
87          public void setProfileLengthLineTwo(double length)
88          {
89              this.profileLengthLineTwo = length;
90          }
91
92          /// <summary>
93          /// Returning the length of the profiles which can run on production
              line two
94          /// </summary>
95          /// <returns>
96          /// The length of the profiles which can run on production line two
97          /// </returns>
98          public double getProfileLengthLineTwo()
99          {
100             return this.profileLengthLineTwo;
101         }
102
```

```
103         /// <summary>
104         /// Optimizing the production order and splits the order into two
              seperate lists.
105         /// One for the AngleProfiles and one list for the GuidingProfiles
106         /// </summary>
107         /// <param name="list">
108         /// The list containing the production order
109         /// </param>
110         /// <exception cref="NotFiniteNumberException">description</exception>

111         /// <exception cref="FormatException">description</exception>
112         public void optimize(OrderedAluList list)
113         {
114             List<OrderedAluProfile> typeSorted = new List<OrderedAluProfile>
                  ();
115
116             var typeRegIterator = this.typeReg.getTypeRegister().GetEnumerator
                  ();
117             // Checking the OrderedAluList for each type of profile in the
                  AluminiumTypeRegister
118             String lastType = "";
119             while (typeRegIterator.MoveNext())
120             {
121                 // Clear after each iteration
122                // typeSorted.Clear();
123
124                 var pair = typeRegIterator.Current;
125                 String type = pair.Key;
126                 int value = pair.Value;
127
128                 // If the value is 0, it means that it is an angled profile
129                 if (value == 0)
130                 {
131                     typeSorted = angleProfileSorter(list, typeSorted, type);
132
133                     try
134                     {
135                         // If the list is not empty, optimize it
136                         if (typeSorted.Any())
137                         {
138                             sendListToOptimizing(typeSorted, value);
139
140                         }
141                     }
142                     catch (NotFiniteNumberException ne)
143                     {
144                         throw ne;
145                     }
146                     catch (FormatException fe)
147                     {
148                         throw fe;
149                     }
150
151
152                 }
153
```

```
154                 // If the value is 1, it means that it is a guiding profile
155                 else if ((value == 1) && (!lastType.Trim('M','F').Contains
                      (type.Trim('M', 'F'))))
156                 {
157                     lastType = type;
158                     typeSorted = guidingProfileSorter(list, typeSorted, type);
159 
160 
161                     try
162                     {
163                         // If the list is not empty, optimize it
164                         if (typeSorted.Any())
165                         {
166                             sendListToOptimizing(typeSorted, value);
167                         }
168                     }
169                     catch (NotFiniteNumberException ne)
170                     {
171                         throw ne;
172                     }
173                     catch (FormatException fe)
174                     {
175                         throw fe;
176                     }
177 
178 
179                 }
180             }
181         }
182 
183         /// <summary>
184         /// Adding the GuidingProfiles which has the same type to the same
              list.
185         /// And returning this list
186         /// </summary>
187         /// <param name="list">
188         /// The list containing the production order
189         /// </param>
190         /// <param name="typeSorted">
191         /// The list you want to sort the profiles of the same type into
192         /// </param>
193         /// <param name="type">
194         /// The profile type you want to sort out
195         /// </param>
196         /// <returns>
197         /// Returning the list containing the sorted GuidingProfiles
198         /// </returns>
199         private List<OrderedAluProfile> guidingProfileSorter(OrderedAluList
              list, List<OrderedAluProfile> typeSorted, string type)
200         {
201 
202             foreach (var alu in list.getList())
203             {
204                 if (alu is GuidingProfile)
205                 {
206                     GuidingProfile x = (GuidingProfile) alu;
```

```
207                     // Neede to remove the "M (Male)" or "F(Female)" from the
                        type string since these both indicate that it is a guiding
                         profile
208                     String xtype = x.getType().Trim('M','F');
209                     String newType = type.Trim('M', 'F');
210
211                     if (xtype.Contains(newType))
212                     {
213                         typeSorted.Add(x);
214                     }
215                 }
216             }
217
218             return typeSorted;
219         }
220
221         /// <summary>
222         /// Adding the AngleProfiles which has the same type to the same list.

223         /// And returning this list
224         /// </summary>
225         /// <param name="list">
226         /// The list containing the production order
227         /// </param>
228         /// <param name="typeSorted">
229         /// The list you want to sort the profiles of the same type into
230         /// </param>
231         /// <param name="type">
232         /// The profile type you want to sort out
233         /// </param>
234         /// <returns>
235         /// Returning the list containing the sorted AngleProfiles
236         /// </returns>
237         private List<OrderedAluProfile> angleProfileSorter(OrderedAluList
              list, List<OrderedAluProfile> typeSorted, string type)
238         {
239
240             foreach (var alu in list.getList())
241             {
242                 if (alu is AngleProfile)
243                 {
244                     AngleProfile x = (AngleProfile) alu;
245
246                     if (x.getType().Contains(type))
247                     {
248                         typeSorted.Add(x);
249                     }
250                 }
251             }
252
253             return typeSorted;
254         }
255
256         /// <summary>
257         /// Sending the list containing the sorted profiles to optimizing.
258         /// </summary>
```

```
259         /// <param name="typeSorted">
260         /// The list containing the type sorted profiles
261         /// </param>
262         /// <param name="value">
263         /// The value which indicates if the list contains Guiding og Angle
              profiles
264         /// </param>
265         /// <exception cref="NotFiniteNumberException">description</exception>

266         /// <exception cref="FormatException">description</exception>
267         private void sendListToOptimizing(List<OrderedAluProfile> typeSorted,
              int value)
268         {
269
270             try
271             {
272                 if (value == 0)
273                 {
274                     this.angleOrderList.addProfiles(this.generateAluProfiles
                      (typeSorted));
275                 }
276
277                 else if (value == 1)
278                 {
279                     this.guidingOrderList.addProfiles(this.generateAluProfiles
                      (typeSorted));
280                 }
281             }
282             catch (NotFiniteNumberException ne)
283             {
284                 throw ne;
285             }
286             catch (FormatException fe)
287             {
288                 throw fe;
289             }
290
291         }
292
293         /// <summary>
294         /// Generating the AluminiumProfile objects from the sorted profile
              list
295         /// And returning a list containing the order.
296         /// </summary>
297         /// <param name="profiles"></param>
298         /// <returns>
299         /// Returning the list containing the order.
300         /// </returns>
301         /// <exception cref="NotFiniteNumberException">description</exception>

302         /// <exception cref="FormatException">description</exception>
303         private List<AluminiumProfile> generateAluProfiles
              (List<OrderedAluProfile> profiles)
304         {
305             List<AluminiumProfile> aluProfiles = new List<AluminiumProfile>();
306
```

```
307             // Calculation variables
308             double totalLengthProfile = 0;
309             double totalLengthMale = 0;
310             double totalLengthFemale = 0;
311
312             // Calculating the number of full profiles needed to complete the
                  order
313             foreach (OrderedAluProfile profile in profiles)
314             {
315                 if(profile is GuidingProfile)
316                 {
317                     GuidingProfile p = (GuidingProfile)profile;
318
319                     if(p.getType().Contains("M"))
320                     {
321                         totalLengthMale += profile.getLength() - getCutShaving
                        (p);
322                     }
323                     else
324                     {
325                         totalLengthFemale += profile.getLength() -
                        getCutShaving(p);
326                     }
327                 }
328
329                 else
330                 {
331                     AngleProfile p = (AngleProfile)profile;
332
333                     totalLengthProfile += profile.getLength() - getCutShaving
                    (p);
334                 }
335
336             }
337
338             // For angled profile
339             double nmbOfAluProfiles = totalLengthProfile /
                  (this.profileLengthLineOne - AluminiumProfile.getEndWaste());
340             double nmbOfWholeProfiles = System.Math.Ceiling(nmbOfAluProfiles);
341
342             // For guiding profiles
343             double nmbOfMaleProfiles = totalLengthMale /
                  (this.profileLengthLineTwo - AluminiumProfile.getEndWaste());
344             double nmbOfWholeMale = System.Math.Ceiling(nmbOfMaleProfiles);
345
346             double nmbOfFemaleProfiles = totalLengthFemale /
                  (this.profileLengthLineTwo - AluminiumProfile.getEndWaste());
347             double nmbOfWholeFemale = System.Math.Ceiling
                  (nmbOfFemaleProfiles);
348
349             // For dimmensions
350             double height = 0;
351             double width = 0;
352
353             // Generating guiding profiles
354             if (profiles.FirstOrDefault() is GuidingProfile)
```

```
355             {
356                 GuidingProfile gprofile = (GuidingProfile)
                      profiles.FirstOrDefault();
357
358                 // Getting the dimmensions for the profile type
359                 this.dimReg.getHeightRegister().TryGetValue(gprofile.getType
                      (), out height);
360                 this.dimReg.getWidthRegister().TryGetValue(gprofile.getType(),
                       out width);
361
362                 // Male profiles
363                 for (int i = 0; i < nmbOfWholeMale; i++)
364                 {
365                     aluProfiles.Add(new AluminiumProfile(gprofile.getType
                      ().Trim('F','M') + "M", this.profileLengthLineTwo, width,
                      height));
366                 }
367
368                 // Female profiles
369                 for (int i = 0; i < nmbOfWholeFemale; i++)
370                 {
371                     aluProfiles.Add(new AluminiumProfile(gprofile.getType
                      ().Trim('M','F') + "F", this.profileLengthLineTwo, width,
                      height));
372                 }
373                 try
374                 {
375                     this.addOrderedGuidingProfiles(aluProfiles, profiles);
376                 }
377                 catch (NotFiniteNumberException ne)
378                 {
379                     throw ne;
380                 }
381                 catch (FormatException fe)
382                 {
383                     throw fe;
384                 }
385
386             }
387
388             // Generating angled profiles
389             else
390             {
391                 AngleProfile aprofile = (AngleProfile)profiles.FirstOrDefault
                      ();
392
393                 // Getting the dimmensions for the profile type
394                 this.dimReg.getHeightRegister().TryGetValue(aprofile.getType
                      (), out height);
395                 this.dimReg.getWidthRegister().TryGetValue(aprofile.getType(),
                       out width);
396
397                 for (int i = 0; i<nmbOfWholeProfiles; i++)
398                 {
399                     aluProfiles.Add(new AluminiumProfile(aprofile.getType(),
                      this.profileLengthLineOne, width, height));
```

```
400                 }
401                 try
402                 {
403                     this.addOrderedAluAngleProfiles(aluProfiles, profiles);
404                 }
405                 catch (NotFiniteNumberException ne)
406                 {
407                     throw ne;
408                 }
409                 catch (FormatException fe)
410                 {
411                     throw fe;
412                 }
413             }
414 
415 
416             return aluProfiles;
417         }
418 
419         /// <summary>
420         /// Optimizing each AluminiumProfile object.
421         /// </summary>
422         /// <param name="aluProfiles">
423         /// A list containing the AluminiumProfile objects you want to optimize
424         /// </param>
425         /// <param name="profiles">
426         /// A list containing the order of GuidingProfiles you want to optimize
427         /// </param>
428         /// <exception cref="NotFiniteNumberException">description</exception>
429         /// <exception cref="FormatException">description</exception>
430         private void addOrderedGuidingProfiles(List<AluminiumProfile> aluProfiles, List<OrderedAluProfile> profiles)
431         {
432             try
433             {
434                 double minimumLengthLastProfile = 400; // In millimeters
435                 List<OrderedAluProfile> forRemoval = new List<OrderedAluProfile>();
436 
437                 // Making sure each Aluminium profile gets one piece longer than 400mm
438                 foreach (AluminiumProfile aluProfile in aluProfiles)
439                 {
440                     Boolean added = false;
441                     foreach (OrderedAluProfile profile in profiles)
442                     {
443                         GuidingProfile gp = (GuidingProfile)profile;
444 
445                         if ((profile.getLength() >= minimumLengthLastProfile) && (!added) && (gp.getType().Contains(aluProfile.getType())))
446                         {
447                             aluProfile.addProfileWithShaving(profile,
```

```
                        this.getCutShaving(gp));
448                             forRemoval.Add(profile);
449                             added = true;
450                         }
451                     }
452                     // Removing the profiles added to the previous
                        AluminiumProfile
453                     foreach (OrderedAluProfile profile in forRemoval)
454                     {
455                         profiles.Remove(profile);
456                     }
457                 }
458 
459                 forRemoval.Clear();
460 
461                 // Adding the rest of the profiles
462                 foreach (AluminiumProfile aluProfile in aluProfiles)
463                 {
464                     foreach (OrderedAluProfile profile in profiles)
465                     {
466                         GuidingProfile gp = (GuidingProfile)profile;
467                         if ((profile.getLength() < aluProfile.getStartWaste())
                         && (gp.getType().Contains(aluProfile.getType())))
468                         {
469                             aluProfile.addProfileWithShaving(profile,
                        this.getCutShaving(gp));
470                             forRemoval.Add(profile);
471 
472                         }
473                     }
474 
475                     // Removing the profiles added to the previous
                        AluminiumProfile
476                     foreach (OrderedAluProfile profile in forRemoval)
477                     {
478                         profiles.Remove(profile);
479                     }
480 
481                 }
482             }
483             catch (NotFiniteNumberException ne)
484             {
485                 throw ne;
486             }
487             catch (FormatException fe)
488             {
489                 throw fe;
490             }
491 
492         }
493 
494         /// <summary>
495         /// Optimizing each AluminiumProfile object.
496         /// </summary>
497         /// <param name="aluProfiles">
498         /// A list containing the AluminiumProfile objects you want to
```

```
            optimize
499         /// </param>
500         /// <param name="profiles">
501         /// A list containing the order of AngleProfiles you want to optimize
502         /// </param>
503         /// <exception cref="NotFiniteNumberException">description</exception>

504         /// <exception cref="FormatException">description</exception>
505         private void addOrderedAluAngleProfiles(List<AluminiumProfile>
              aluProfiles, List<OrderedAluProfile> profiles)
506         {
507             try
508             {
509                 double minimumLengthLastProfile = 400; // In millimeters
510                 List<OrderedAluProfile> forRemoval = new
                      List<OrderedAluProfile>();
511
512                 // Making sure each Aluminium profile gets one piece longer
                      than 400mm
513                 foreach (AluminiumProfile aluProfile in aluProfiles)
514                 {
515                     Boolean added = false;
516                     foreach (OrderedAluProfile profile in profiles)
517                     {
518                         AngleProfile ap = (AngleProfile)profile;
519                         if ((profile.getLength() >= minimumLengthLastProfile)
                        && (!added))
520                         {
521                             aluProfile.addProfileWithShaving(profile,
                        this.getCutShaving(ap));
522                             forRemoval.Add(profile);
523                             added = true;
524                         }
525                     }
526
527                     // Removing the profiles added to the previous
                        AluminiumProfile
528                     foreach (OrderedAluProfile profile in forRemoval)
529                     {
530                         profiles.Remove(profile);
531                     }
532                 }
533
534                 forRemoval.Clear();
535
536                 // Adding the rest of the profiles
537                 foreach (AluminiumProfile aluProfile in aluProfiles)
538                 {
539                     foreach (OrderedAluProfile profile in profiles)
540                     {
541                         AngleProfile ap = (AngleProfile)profile;
542                         if (profile.getLength() < aluProfile.getStartWaste())
543                         {
544                             aluProfile.addProfileWithShaving(profile,
                        this.getCutShaving(ap));
545                             forRemoval.Add(profile);
```

```
546                         }
547                     }
548
549                     // Removing the profiles added to the previous
                        AluminiumProfile
550                     foreach (OrderedAluProfile profile in forRemoval)
551                     {
552                         profiles.Remove(profile);
553                     }
554                 }
555             }
556
557             catch (NotFiniteNumberException ne)
558             {
559                 throw ne;
560             }
561             catch (FormatException fe)
562             {
563                 throw fe;
564             }
565
566
567         }
568
569         /// <summary>
570         /// Calculating and returning the Cut Shaving
571         /// </summary>
572         /// <param name="ap">
573         /// An GuidingProfile
574         /// </param>
575         /// <returns>
576         /// Returning the Cut shaving
577         /// </returns>
578         /// <exception cref="NotFiniteNumberException">description</exception>

579         /// <exception cref="FormatException">description</exception>
580         private double getCutShaving(GuidingProfile gp)
581         {
582             double shaving = 0;
583             try
584             {
585                 // Fetching the last cut for the profile
586                 Cut cut = gp.getSortedCutList().Last();
587
588                 if (cut is AngleCut)
589                 {
590                     AngleCut c = (AngleCut)cut;
591                     shaving = this.cutShaving.calculateShaving(Math.Abs
                        (c.getCutAngle()));
592                 }
593
594                 else if (cut is RotationCut)
595                 {
596                     RotationCut c = (RotationCut)cut;
597                     shaving = this.cutShaving.calculateShaving(Math.Abs
                        (c.getCutAngle()));
```

```
598                 }
599
600             }
601             catch(NotFiniteNumberException ne)
602             {
603                 throw ne;
604             }
605             catch (FormatException fe)
606             {
607                 throw fe;
608             }
609
610             return shaving;
611         }
612
613         /// <summary>
614         /// Calculating and returning the Cut Shaving
615         /// </summary>
616         /// <param name="ap">
617         /// An AngleProfile object
618         /// </param>
619         /// <returns>
620         /// Returning the Cut shaving
621         /// </returns>
622         /// <exception cref="NotFiniteNumberException">description</exception>
623         /// <exception cref="FormatException">description</exception>
624         private double getCutShaving(AngleProfile ap)
625         {
626             double shaving = 0;
627
628             try
629             {
630                 // Fetching the last cut for the profile
631                 Cut cut = ap.getSortedCutList().Last();
632
633                 if (cut is AngleCut)
634                 {
635                     AngleCut c = (AngleCut)cut;
636                     shaving = this.cutShaving.calculateShaving(Math.Abs
                        (c.getCutAngle()));
637                 }
638
639                 else if (cut is RotationCut)
640                 {
641                     RotationCut c = (RotationCut)cut;
642                     shaving = this.cutShaving.calculateShaving(Math.Abs
                        (c.getCutAngle()));
643                 }
644             }
645             catch (NotFiniteNumberException ne)
646             {
647                 throw ne;
648             }
649             catch (FormatException fe)
650             {
```

```
                    throw fe;
                }


                return shaving;
            }
        }
    }

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Main
8  {
9      /// <summary>
10     /// The AluProfileOrderList represents a list or register of the class
         OrderedAluProfile.
11     /// It is possible to add and remove AluProfileOrderList objects from the
         list.
12     /// </summary>
13     public class AluProfileOrderList
14     {
15         private List<AluminiumProfile> profiles;
16
17         /// <summary>
18         /// Creating an instance of AluProfileOrderList
19         /// </summary>
20         public AluProfileOrderList()
21         {
22             this.profiles = new List<AluminiumProfile>();
23         }
24
25         /// <summary>
26         /// Removing an object of AluminiumProfile to the OrderedAluList list.
27         /// </summary>
28         /// <param name="profile">
29         /// The object of the type AluminiumProfile you want to add to the list
30         /// </param>
31         public void addProfile(AluminiumProfile profile)
32         {
33             this.profiles.Add(profile);
34         }
35
36         /// <summary>
37         /// Adding a list of AluminiumProfiles to the AluProfileOrderList
             register.
38         /// </summary>
39         /// <param name="profiles">
40         /// An object of the type List<AluminiumProfile>
41         /// </param>
42         public void addProfiles(List<AluminiumProfile> profiles)
43         {
44             this.profiles.AddRange(profiles);
45         }
46
47         /// <summary>
48         /// Removing an object of AluminiumProfile from the OrderedAluList
             list.
49         /// </summary>
50         /// <param name="profile">
51         /// The object of the type AluminiumProfile you want to remove from the
             list
```

```
52          /// </param>
53          public void removeProfile(AluminiumProfile profile)
54          {
55              this.profiles.Remove(profile);
56          }
57
58          /// <summary>
59          /// Returning the list containing the AluminiumProfile objects
60          /// </summary>
61          /// <returns>
62          /// The list containing the AluminiumProfile objects
63          /// </returns>
64          public List<AluminiumProfile> getOrderList()
65          {
66              return this.profiles;
67          }
68      }
69  }
70
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using RUFOServer.Program.Entity;
7
8  namespace RUFOServer.Program.Main
9  {
10     /// <summary>
11     /// The AngleProfile represents an ordered aluminiums angled profile for
         production.
12     /// Such an profile is ordered into a spesific length which requiers x
         amount of cuts.
13     /// In addition to the cutting, such a profile may also need holes for POP
          rivets.
14     /// All this information i stored whithin the class, and can be retracted
         if needed.
15     /// </summary>
16     public class AngleProfile : OrderedAluProfile
17     {
18         private String type;
19         private CutList cuts;
20         private HoleList holes;
21
22         /// <summary>
23         /// Creating an instance of AnglePofile
24         /// </summary>
25         /// <param name="length">
26         /// The length of the profile in millimeters
27         /// </param>
28         /// <param name="orderNmb">
29         /// The order number for the profile
30         /// </param>
31         /// <param name="type">
32         /// The profile type
33         /// </param>
34         /// <param name="prodNmb">
35         /// The product number
36         /// </param>
37         /// <param name="itemNmb">
38         /// The item number in the cad drawing
39         /// </param>
40         public AngleProfile(double length, string orderNmb, String type,
             string prodNmb, string itemNmb) : base(length, orderNmb, prodNmb,
             itemNmb)
41         {
42             this.type = type;
43             this.cuts = new CutList();
44             this.holes = new HoleList();
45         }
46
47         /// <summary>
48         /// Returning the angled alu profile type
49         /// </summary>
50         /// <returns>
```

```
51          /// The angle alu profile type
52          /// </returns>
53          public String getType()
54          {
55              return this.type;
56          }
57
58          /// <summary>
59          /// Adding a Cut to the CuttingList
60          /// </summary>
61          /// <param name="cut">
62          /// The object of the class Cut, you want to add to the list
63          /// </param>
64          public void addCut(Cut cut)
65          {
66              this.cuts.addCut(cut);
67          }
68
69          /// <summary>
70          /// Returning the sorted CuttingList by accending order according to
              the positioning of each cut.
71          /// Meaning the Cut with the lowest position in millimeters is at the
              top of the list.
72          /// </summary>
73          /// <returns>
74          /// The sorted list by accending order using the positioning for each
              Cut
75          /// </returns>
76          public List<Cut> getSortedCutList()
77          {
78              return this.cuts.getSortedList();
79          }
80
81          /// <summary>
82          /// Adding a Hole to the HoleList
83          /// </summary>
84          /// <param name="cut">
85          /// The object of the class Hole, you want to add to the list.
86          /// </param>
87          public void addHole(Hole hole)
88          {
89              this.holes.addHole(hole);
90          }
91
92          /// <summary>
93          /// Returning the sorted HoleList by accending order according to the
              x-axis position of each hole.
94          /// Meaning the Hole with the lowest position in the x-axis in
              millimeters is at the top of the list.
95          /// </summary>
96          /// <returns>
97          /// The sorted list by accending order using the x-axis position for
              each Hole
98          /// </returns>
99          public List<Hole> getSortedHoleList()
100         {
```

```
101             return this.holes.getSortedList();
102         }
103 
104         /// <summary>
105         /// Setting the holeList equal to another holeList
106         /// </summary>
107         /// <param name="list">
108         /// An object of the class HoleList
109         /// </param>
110         public void setHoleList(HoleList list)
111         {
112             this.holes = list;
113         }
114     }
115 }
116 
```

```
using RUFOServer.Program.Entity;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RUFOServer.Program.Main
{
    /// <summary>
    /// The CNCGenerator class generates CNC code from the OrderedAluProfiles
       inside the AluminiumProfiles internal list.
    /// CNC code rules:
    /// TI = Transport Inn
    /// TO = Transport out
    /// TC = TrimCut
    /// TR = TrimCutRotation;
    /// BC = BendCut
    /// AC = Angle cut(From the side inn)
    /// RC = Rotation cut(From the top down)
    /// M = Milling guiding profile
    /// MA = Milling A side angle profile
    /// MB = Milling B side angle profile
    /// D = the profile is produced
    /// </summary>
    public class CNCGenerator
    {
        private String[] cncArray;
        private int length;
        private double cutOffset; // inn millimeters
        private double milOffset; // inn millimeters
        private double maxTransDistInn; // inn millimeters

        private HoleList holes;
        private CutList cuts;
        private CutShaving cutShaving;

        /*CNC code rules:

        TI = Transport Inn
        TO = Transport out
        TC = TrimCut
        TR = TrimCutRotation;
        BC = BendCut
        AC = Angle cut(From the side inn)
        RC = Rotation cut(From the top down)
        M = Milling guiding profile
        MA = Milling A side angle profile
        MB = Milling B side angle profile
        D = the profile is produced*/

        private string transportInn = "TI";
        private string transportOut = "TO";
        private string TrimCut = "TC";
        private string TrimCutRotation = "TR";
```

```
56          private string BendCut = "BC1";
57          private string AngleCut = "AC";
58          private string RotationCut = "RC";
59          private string milling = "M ";
60          private string millingASide = "MA";
61          private string MillingBSide = "MB";
62          private string Done = "D ";
63
64          /// <summary>
65          /// Creating an instance of CNCGenerator.
66          /// Needing the required size of the CNCArray and offset, which is
              the distance between the cut station to the milling station
67          /// </summary>
68          /// <param name="cncArrayLength">
69          /// The required size of the CNCArray
70          /// </param>
71          /// <param name="offset">
72          /// The distance between the cut station to the milling station in
              millimeters
73          /// </param>
74          /// <param name="maxTransDistInn">
75          /// The max travel distance for the feeding in millimeter
76          /// </param>
77          /// <param name="cutShaving">
78          /// An object of the type CutShaving
79          /// </param>
80          public CNCGenerator(int cncArrayLength, double cutOffset, double
              milOffset, double maxTransDistInn, CutShaving cutShaving)
81          {
82              this.length = cncArrayLength;
83              this.cncArray = new String[this.length];
84              this.cutOffset = cutOffset;
85              this.milOffset = milOffset;
86              this.maxTransDistInn = maxTransDistInn;
87              this.holes = new HoleList();
88              this.cuts = new CutList();
89              this.cutShaving = cutShaving;
90              initArray();
91          }
92
93          /// <summary>
94          /// Returning the length of the CNC array as an int
95          /// </summary>
96          /// <returns>
97          /// The length of the CNC array as an int
98          /// </returns>
99          public int getCNCLength()
100         {
101             return this.length;
102         }
103
104         /// <summary>
105         /// Returning the distance between the cut station and origo in
              millimeters
106         /// </summary>
107         /// <returns>
```

```
108         /// The distance between the cut station and origo in millimeters
109         /// </returns>
110         public double getCutOffset()
111         {
112             return this.cutOffset;
113         }
114 
115         /// <summary>
116         /// Returning the distance between the milling station and origo in
              millimeters
117         /// </summary>
118         /// <returns>
119         /// The distance between the milling station and origo in millimeters
120         /// </returns>
121         public double getMilOffset()
122         {
123             return this.milOffset;
124         }
125 
126         /// <summary>
127         /// Setting the distance between the milling station and origo in
              millimeters
128         /// </summary>
129         /// <param name="offset">
130         /// The distance between the milling station and origo in millimeters
131         /// </param>
132         public void setMilOffset(double offset)
133         {
134             this.milOffset = offset;
135         }
136 
137         /// <summary>
138         /// Setting the distance between the cut station and origo in
              millimeters
139         /// </summary>
140         /// <param name="offset">
141         /// The distance between the cut station and origo in millimeters
142         /// </param>
143         public void setCutOffset(double offset)
144         {
145             this.cutOffset = offset;
146         }
147 
148         /// <summary>
149         /// Creating CNCCode which includes transport and actions from the
              OrderedAluProfile
150         /// objects which the AluminiumProfile has in its register. Were
              transport is movement and actions are either holes or cuts.
151         /// And putting the CNCCode into a string array which is returned
152         /// </summary>
153         /// <returns>
154         /// Returning an array containing the CNCCode
155         /// </returns>
156         public String[] generateCNCCode(AluminiumProfile profile)
157         {
158             // Sumarizing all the holes in one list and all the cuts in
```

```
                another
159             generateHolesAndCuts(profile);
160
161             // Building an entire Aluminiumprofile with just cuts and holes
162             buildProfile(profile);
163
164             // Counter for index position for information in the array
165             int counter = 0;
166
167             // Distance between the milling station and the cut station
168             double offset = this.cutOffset - this.milOffset;
169
170             // Transport
171             double transport = 0;
172             double oldTransport = 0;
173
174             // While loop control
175             bool generating = true;
176
177             // Iterator control
178             IEnumerator<Cut> cutIt = this.cuts.getCuttingList().GetEnumerator
                ();
179             IEnumerator<Hole> holeIt = this.holes.getHoleList().GetEnumerator
                ();
180             Cut cut = null;
181             Cut prevCut = null;
182             Hole hole = null;
183
184             // To check if the lists are emtpy or not
185             bool cutIthasNext = cutIt.MoveNext();
186             bool holeItHasNext = holeIt.MoveNext();
187
188                 // Generating CNC code
189                 while (generating)
190                 {
191                 //
                    ************************************************************
                    ************************
192                 // Both lists still have elements
193
194
195                 if (cutIthasNext && holeItHasNext)
196                 {
197                     cut = cutIt.Current;
198                     hole = holeIt.Current;
199
200                     // This is the distance between the first cut and the
                    next hole to be milled
201                     double distCut = profile.getStartWaste() + this.cutOffset
                     + cut.getCutPosition() + getCutAngleOffset(cut, profile);
202                     distCut = Math.Round(distCut, 2,
                    MidpointRounding.AwayFromZero);
203                     double distHole = profile.getStartWaste() + hole.getXAxis
                    () + this.milOffset;
204                     distHole = Math.Round(distHole, 2,
                    MidpointRounding.AwayFromZero);
```

```
205
206                     // Hole first
207                     if (distHole <= distCut)
208                     {
209                         // No transport needed
210                         if (transport == distHole)
211                         {
212                             //Check that there is hole on both sides
213                             if (hole.getZPosition() != null)
214                             {
215
216                                 // A side
217                                 if (hole.getZPosition().Equals("A"))
218                                 {
219                                     this.cncArray[counter] =
                    this.millingASide + hole.getYAxis().ToString();
220                                     counter++;
221                                     holeItHasNext = holeIt.MoveNext();
222                                 }
223                                 // B side
224                                 else
225                                 {
226                                     this.cncArray[counter] =
                    this.MillingBSide + hole.getYAxis().ToString();
227                                     counter++;
228                                     holeItHasNext = holeIt.MoveNext();
229                                 }
230                             }
231                             // Guiding profile
232                             else
233                             {
234                                 // action
235                                 this.cncArray[counter] = this.milling +
                    hole.getYAxis().ToString();
236                                 counter++;
237                                 holeItHasNext = holeIt.MoveNext();
238
239                             }
240                             // cutting if there is a cut in the same pos as
                    milling
241                             if (transport == distCut)
242                             {
243                                 cut = cutIt.Current;
244
245                                 // Finding what type of cut it is
246                                 this.cncArray[counter] = getCNCCutType(cut) +
                     getCNCCutValue(cut);
247                                 counter++;
248
249                                 // if the cut is final cut there is added a
                    transport out.
250                                 if (cut.isFinalCut())
251                                 {
252                                     this.cncArray[counter] =
                    this.transportOut;
253                                     counter++;
```

```
254                                 }
255
256                                 cutIthasNext = cutIt.MoveNext();
257                             }
258                         }
259                         // Transport needed
260                         else
261                         {
262                             // transport
263
264                             double transportToAddNotRounded = -transport +
                        profile.getStartWaste() + hole.getXAxis() +
                        this.milOffset;
265                             // Rounding the number
266                             double transportToAdd = Math.Round
                        (transportToAddNotRounded, 1,
                        MidpointRounding.AwayFromZero);
267                             transport = Math.Round(transport +
                        transportToAddNotRounded, 2,
                        MidpointRounding.AwayFromZero);
268
269                             // Transport inn can still move
270                             if (transport <= this.maxTransDistInn)
271                             {

272                                 this.cncArray[counter] = this.transportInn +
                        transportToAdd.ToString();
273                                 counter++;
274                             }
275                             // Maxout transport inn
276                             else if ((transport > this.maxTransDistInn) &&
                        (oldTransport <= this.maxTransDistInn))
277                             {
278                                 double transNotRounded = this.maxTransDistInn
                         - oldTransport;
279                                 double trans = Math.Round(transNotRounded, 1,
                         MidpointRounding.AwayFromZero);
280
281                                 this.cncArray[counter] = this.transportInn +
                        trans.ToString();
282                                 counter++;
283
284                                 double restTransNotRounded =
                        transportToAddNotRounded - transNotRounded;
285                                 double restTrans = Math.Round
                        (restTransNotRounded, 1, MidpointRounding.AwayFromZero);
286
287                                 this.cncArray[counter] = this.transportOut +
                        restTrans.ToString();
288                                 counter++;
289                             }
290                             // Transport out has to take over
291                            else
292                             {
293                                 this.cncArray[counter] = this.transportOut +
                        transportToAdd.ToString();
```

```
294                                 counter++;
295                             }
296
297                             oldTransport = transport;
298                         }
299                     }
300                     //  Cut first
301                     else
302                     {
303                         cut = cutIt.Current;
304                         double transportToAddNotRounded = cut.getCutPosition
                        () - transport + this.cutOffset + profile.getStartWaste()
                        + getCutAngleOffset(cut, profile);
305                         double transportToAdd = Math.Round
                        (transportToAddNotRounded, 1,
                        MidpointRounding.AwayFromZero); // Rounding the number
306
307                         // If the cuts at the same position dont match in
                        angle make another cut
308                         if ((prevCut != null) && (transportToAdd == 0))
309                         {
310                             // Checking if the previous cut has the same
                            angle and type
311                             if ((getCutAngle(prevCut) != getCutAngle(cut)) ||
                             (!this.getCNCCutType(prevCut).Equals(this.getCNCCutType
                            (cut))))
312                             {
313                                 // Finding what type of cut it is
314                                 this.cncArray[counter] = getCNCCutType(cut) +
                             getCNCCutValue(cut);
315                                 counter++;
316                                 // if the cut is final cut there is added a
                                transport out.
317                                 if (cut.isFinalCut())
318                                 {
319                                     this.cncArray[counter] =
                                    this.transportOut;
320                                     counter++;
321                                 }
322
323
324                             }
325                         }
326                         else
327                         {
328                             if (transportToAdd != 0)
329                             {
330                                 transport = Math.Round(transport +
                                transportToAddNotRounded, 2,
                                MidpointRounding.AwayFromZero);
331
332                                 // Transport inn can still move
333                                 if (transport <= this.maxTransDistInn)
334                                 {
335                                     this.cncArray[counter] =
                                    this.transportInn + transportToAdd.ToString();
```

```
336                                     counter++;
337                                 }
338                                 // Maxout transport inn
339                                 else if ((transport > this.maxTransDistInn)
                        && (oldTransport <= this.maxTransDistInn))
340                                 {
341                                     double transNotRounded =
                        this.maxTransDistInn - oldTransport;
342                                     double trans = Math.Round
                        (transNotRounded, 1, MidpointRounding.AwayFromZero);
343
344                                     this.cncArray[counter] =
                        this.transportInn + trans.ToString();
345                                     counter++;
346
347                                     double restTransNotRounded =
                        transportToAddNotRounded - transNotRounded;
348                                     double restTrans = Math.Round
                        (restTransNotRounded, 1, MidpointRounding.AwayFromZero);
349
350                                     this.cncArray[counter] =
                        this.transportOut + restTrans.ToString();
351                                     counter++;
352                                 }
353                                 // Transport out has to take over
354                                 else
355                                 {
356                                     this.cncArray[counter] =
                        this.transportOut + transportToAdd.ToString();
357                                     counter++;
358                                 }
359
360                                 oldTransport = transport;
361
362                             }
363                             // Finding what type of cut it is
364                             this.cncArray[counter] = getCNCCutType(cut) +
                        getCNCCutValue(cut);
365                             counter++;
366
367                             // if the cut is final cut there is added a
                        transport out.
368                             if (cut.isFinalCut())
369                             {
370                                 this.cncArray[counter] = this.transportOut;
371                                 counter++;
372                             }
373                         }
374                         cutIthasNext = cutIt.MoveNext();
375                     }
376                 }
377
378                 //
                      ************************************************************
                      ************************
379                 // Just the holelist have elements left
```

```
380                 else if (holeItHasNext)
381                 {
382                     hole = holeIt.Current;
383                     double transportToAddNotRounded = hole.getXAxis() -
                    transport + this.milOffset + profile.getStartWaste();
384                     double transportToAdd = Math.Round
                    (transportToAddNotRounded, 1,
                    MidpointRounding.AwayFromZero);
385
386                     if (transportToAdd != 0)
387                     {
388                         transport = Math.Round(transport +
                    transportToAddNotRounded, 2,
                    MidpointRounding.AwayFromZero);
389                         // Transport inn can still move
390                         if (transport <= this.maxTransDistInn)
391                         {
392                             this.cncArray[counter] = this.transportInn +
                    transportToAdd.ToString();
393                             counter++;
394                         }
395                         // Maxout transport inn
396                         else if ((transport > this.maxTransDistInn) &&
                    (oldTransport <= this.maxTransDistInn))
397                         {
398                             double transNotRounded = this.maxTransDistInn -
                    oldTransport;
399                             double trans = Math.Round(transNotRounded, 1,
                    MidpointRounding.AwayFromZero);
400
401                             this.cncArray[counter] = this.transportInn +
                    trans.ToString();
402                             counter++;
403
404                             double restTransNotRounded =
                    transportToAddNotRounded - transNotRounded;
405                             double restTrans = Math.Round
                    (restTransNotRounded, 1, MidpointRounding.AwayFromZero);
406
407                             this.cncArray[counter] = this.transportOut +
                    restTrans.ToString();
408                             counter++;
409                         }
410                         // Transport out has to take over
411                         else
412                         {
413                             this.cncArray[counter] = this.transportOut +
                    transportToAdd.ToString();
414                             counter++;
415                         }
416
417                         oldTransport = transport;
418                     }
419
420                     //check that there is hole on both sides
421                     if (hole.getZPosition() != null)
```

```
422                     {
423                         // A side
424                         if (hole.getZPosition().Equals("A"))
425                         {
426                             this.cncArray[counter] = this.millingASide +
                        hole.getYAxis().ToString();
427                             counter++;
428                             holeItHasNext = holeIt.MoveNext();
429                         }
430                         // B side
431                         else
432                         {
433                             this.cncArray[counter] = this.MillingBSide +
                        hole.getYAxis().ToString();
434                             counter++;
435                             holeItHasNext = holeIt.MoveNext();
436                         }
437                     }
438                     // Guiding profile
439                     else
440                     {
441                         // action
442                         this.cncArray[counter] = this.milling + hole.getYAxis
                        ().ToString();
443                         counter++;
444                         holeItHasNext = holeIt.MoveNext();
445 
446                     }
447                 }
448 
449                 //
                      ************************************************************
                      ************************
450                 // Just the cutlist have elements left
451                 else if (cutIthasNext)
452                 {
453                     cut = cutIt.Current;
454                     double transportToAddNotRounded = cut.getCutPosition() -
                        transport + this.cutOffset + profile.getStartWaste() +
                        getCutAngleOffset(cut, profile);
455                     double transportToAdd = Math.Round
                        (transportToAddNotRounded, 1,
                        MidpointRounding.AwayFromZero);
456 
457                     // If the cuts at the same position dont match in angle
                        make another cut
458                     if ((prevCut != null) && (transportToAdd == 0))
459                     {
460                         if ((this.getCutAngle(prevCut) != this.getCutAngle
                        (cut) || (!this.getCNCCutType(prevCut).Equals
                        (this.getCNCCutType(cut)))))
461                         {
462                             // Finding what type of cut it is
463                             this.cncArray[counter] = getCNCCutType(cut) +
                        getCNCCutValue(cut);
464                             counter++;
```

```
465
466                         // if the cut is final cut there is added a transport out.
467                         if (cut.isFinalCut())
468                         {
469                             this.cncArray[counter] = this.transportOut;
470                             counter++;
471                         }
472                     }
473                 }
474                 else
475                 {
476                     if (transportToAdd != 0)
477                     {
478                         transport = Math.Round(transport + transportToAddNotRounded, 2, MidpointRounding.AwayFromZero);
479
480                         // Transport inn can still move
481                         if (transport <= this.maxTransDistInn)
482                         {
483                             this.cncArray[counter] = this.transportInn + transportToAdd.ToString();
484                             counter++;
485                         }
486                         // Maxout transport inn
487                         else if ((transport > this.maxTransDistInn) && (oldTransport <= this.maxTransDistInn))
488                         {
489                             double transNotRounded = this.maxTransDistInn - oldTransport;
490                             double trans = Math.Round(transNotRounded, 1, MidpointRounding.AwayFromZero);
491
492                             this.cncArray[counter] = this.transportInn + trans.ToString();
493                             counter++;
494
495                             double restTransNotRounded = transportToAddNotRounded - transNotRounded;
496                             double restTrans = Math.Round (restTransNotRounded, 1, MidpointRounding.AwayFromZero);
497
498                             this.cncArray[counter] = this.transportOut + restTrans.ToString();
499                             counter++;
500                         }
501                         // Transport out has to take over
502                         else
503                         {
504                             this.cncArray[counter] = this.transportOut + transportToAdd.ToString();
505                             counter++;
506                         }
507
508                         oldTransport = transport;
```

```
509                         }
510                         // Finding what type of cut it is
511                         this.cncArray[counter] = getCNCCutType(cut) +
                        getCNCCutValue(cut);
512                         counter++;
513
514                         // if the cut is final cut there is added a transport
                         out.
515                         if (cut.isFinalCut())
516                         {
517                             this.cncArray[counter] = this.transportOut;
518                             counter++;
519                         }
520
521                     }
522
523                     cutIthasNext = cutIt.MoveNext();
524                 }
525                 //
                    ************************************************************
                    ************************
526                 // Both lists are empty
527                 else if (!holeItHasNext && !cutIthasNext)
528                 {
529                     this.cncArray[counter] = Done;
530                     generating = false;
531                 }
532                     //
                    ************************************************************
                    ************************
533
534                     // Changes before next iteration
535                     prevCut = cut;
536                     cut = null;
537                     hole = null;
538                 }
539
540                 return this.cncArray;
541
542         }
543
544
545
546         /// <summary>
547         /// Get the cut offset depending on if it is a beginnig cut or final
              cut.
548         /// First cut has a offsett if the angle is negativ sign.
549         /// last cut has a offsett if the angle is positiv.
550         /// This is to make the cuting machine cut to the same position.
551         /// </summary>
552         /// <param name="cut">
553         /// the cut to be checked for offsett.
554         /// </param>
555         ///
556         /// <param name="profile">
557         /// profile type, to adjust for width and length
```

```
558         /// </param>
559         /// <returns></returns>
560         private Double getCutAngleOffset(Cut cut, AluminiumProfile profile )
561         {
562             double cutOffset = 0;
563 
564             if (this.getCutAngle(cut) != 0)
565             {
566 
567                 if ((cut.isFinalCut()) && (this.getCutAngle(cut)>0))
568                 {
569                     if (cut is AngleCut)
570                     {
571                         // If it is 90 degrees cutoffset = 0
572                         if(this.getCutAngle(cut) != 90)
573                         {
574                             cutOffset = -profile.getWidth() * Math.Tan
                        (this.degToRad(this.getCutAngle(cut)));
575                         }
576                     }
577                     else if (cut is RotationCut)
578                     {
579                         // If it is 90 degrees cutoffset = 0
580                         if (this.getCutAngle(cut) != 90)
581                         {
582                             cutOffset = -profile.getHeight() * Math.Tan
                        (this.degToRad(this.getCutAngle(cut)));
583                         }
584                     }
585 
586                 }
587 
588                 else if ((!cut.isFinalCut()) && (this.getCutAngle(cut) < 0))
589                 {
590                     if (cut is AngleCut)
591                     {
592                         // If it is 90 degrees cutoffset = 0
593                         if (this.getCutAngle(cut) != 90)
594                         {
595                             cutOffset = -profile.getWidth() * Math.Tan
                        (this.degToRad(this.getCutAngle(cut)));
596                         }
597                     }
598 
599                     else if (cut is RotationCut)
600                     {
601                         // If it is 90 degrees cutoffset = 0
602                         if (this.getCutAngle(cut) != 90)
603                         {
604                             cutOffset = -profile.getHeight() * Math.Tan
                        (this.degToRad(this.getCutAngle(cut)));
605                         }
606                     }
607                 }
608             }
609 
```

```
610
611                 return cutOffset;
612
613         }
614
615         /// <summary>
616         /// Identifying what type of cut it is and returning the angle value as a double.
617         /// Returning null if the cut is not identified with an angle value
618         /// </summary>
619         /// <param name="cut">
620         /// The cut you want the angle from
621         /// </param>
622         /// <returns>
623         /// Returning the cut angle value as a double.
624         /// Returning null if the cut is not identified with an angle value
625         /// </returns>
626         private double getCutAngle(Cut cut)
627         {
628             double angle = 0;
629
630             if (cut is AngleCut)
631             {
632                 AngleCut c = (AngleCut)cut;
633                 angle = c.getCutAngle();
634             }
635             else if(cut is TrimCut)
636             {
637                 TrimCut c = (TrimCut)cut;
638                 angle = c.getRotation();
639             }
640
641             else if (cut is RotationCut)
642             {
643                 RotationCut c = (RotationCut)cut;
644                 angle = c.getCutAngle();
645             }
646
647             return angle;
648         }
649
650         /// <summary>
651         /// Identifying what type of cut it is and returning a String containing the CNC angle value for that cut
652         /// </summary>
653         /// <param name="cut">
654         /// The Cut you want to generate CNC code for
655         /// </param>
656         /// <returns>
657         /// Returning the CNC value for the cut
658         /// </returns>
659         private string getCNCCutValue(Cut cut)
660         {
661             string CNCCodeValue = "";
662
663             if (cut is AngleCut)
```

```
664             {
665                 AngleCut c = (AngleCut)cut;
666                 CNCCodeValue = c.getCutAngle().ToString();
667             }
668             else if (cut is RotationCut)
669             {
670                 RotationCut c = (RotationCut)cut;
671                 CNCCodeValue = c.getCutAngle().ToString();
672             }
673             else if (cut is TrimCut)
674             {
675                 TrimCut c = (TrimCut)cut;
676                 CNCCodeValue = c.getRotation().ToString();
677             }
678 
679             return CNCCodeValue;
680         }
681 
682         /// <summary>
683         /// Convert angle in degrees to radians
684         /// </summary>
685         /// <param name="degree">
686         /// angel in degrees
687         /// </param>
688         /// <returns></returns>
689         private Double degToRad(Double degree)
690         {
691             return ((degree * Math.PI) / 180);
692         }
693         /// <summary>
694         /// Identifying what type of cut it is and returning a String containing the CNC code for that cut
695         /// </summary>
696         /// <param name="cut">
697         /// The Cut you want to generate CNC code for
698         /// </param>
699         /// <returns>
700         /// Returning the CNC code for the cut
701         /// </returns>
702         private string getCNCCutType(Cut cut)
703         {
704             string CNCCode = "";
705 
706             // Finding what type of cut it is
707             if (cut is TrimCut)
708             {
709                 TrimCut c = (TrimCut)cut;
710                 // The Trimcut needs 4 parts, "TC15 TR45" So 3 of the parts are added here
711                 // "TC15 TR"
712                 CNCCode = this.TrimCut + c.getHeightPos().ToString() + " " + this.TrimCutRotation;
713             }
714             else if (cut is BendCut)
715             {
716                 CNCCode = this.BendCut;
```

```
717             }
718             else if (cut is AngleCut)
719             {
720                 AngleCut c = (AngleCut)cut;
721                 CNCCode = this.AngleCut;
722             }
723             else if (cut is RotationCut)
724             {
725                 RotationCut c = (RotationCut)cut;
726                 CNCCode = this.RotationCut;
727             }
728             return CNCCode;
729         }
730 
731         /// <summary>
732         /// Building the profile as one object. This means summarising all
              the holes an cuts
733         /// </summary>
734         /// <param name="profile">
735         /// Sending in the profile you want to build
736         /// </param>
737         private void buildProfile(AluminiumProfile profile)
738         {
739 
740             Cut previousCut = null;
741             Cut forTrimCut = null;
742             // List containing the length of OrderedAluProfile
743             double[] profileLengths = new double[profile.getSortedAluList
                ().getList().Count];
744             // List containing the previous cut positions
745             double[] previousPos = new double[this.cuts.getCuttingList
                ().Count];
746             // Counters for the logic
747             int counter = 0;
748             int counterC = 0;
749             int index = 0;
750             int startIndex = 0;
751             int indexCounter = 0;
752 
753             // Sumarizing the cuts
754             foreach (OrderedAluProfile p in profile.getSortedAluList
                ().getList())
755             {
756                 // Check what type of profile it is to get the cut list
                    length
757 
758                 if (p is AngleProfile)
759                 {
760                     AngleProfile aP = (AngleProfile)p;
761                     index = aP.getSortedCutList().Count;
762                 }
763                 else
764                 {
765                     GuidingProfile gP = (GuidingProfile)p;
766                     index = gP.getSortedCutList().Count;
767                 }
```

```
768
769                 counter = 0;
770
771                 foreach (Cut cut in this.cuts.getCuttingList().Skip
                      (startIndex))
772                 {
773                     if (counter < index)
774                     {
775                         if ((cut is AngleCut) || (cut is RotationCut))
776                         {
777                             // The first cut is always at position 0 + waste
778                             if ((startIndex == 0) && (counter == 0))
779                             {
780                                 previousPos[indexCounter] =
                        cut.getCutPosition();
781                                 forTrimCut = cut;
782                             }
783
784                             // If the cutposition is 0 and not the first cut,
                         we need to set it equalt to the previous cut
785                             //first cut of profile
786                             else if (cut.getCutPosition() == 0)
787                             {
788                                 if (previousCut is RotationCut)
789                                 {
790                                     RotationCut c = (RotationCut)previousCut;
791                                     //
792                                     if (c.getCutAngle() < 0)
793                                     {
794                                         previousPos[indexCounter] =
                        cut.getCutPosition() + this.getCutShaving(previousCut);
795                                         cut.setCutPosition
                        (previousCut.getCutPosition() + this.getCutShaving
                        (previousCut));
796                                     }
797
798                                     else
799                                     {
800                                         previousPos[indexCounter] =
                        cut.getCutPosition();
801                                         cut.setCutPosition
                        (previousCut.getCutPosition());
802                                     }
803                                 }
804                                 else
805                                 {
806
807                                     previousPos[indexCounter] =
                        cut.getCutPosition();
808                                     cut.setCutPosition
                        (previousCut.getCutPosition());
809                                 }
810                                 forTrimCut = cut;
811                             }
812
813                             // If cut is not zero the position of the cut is
```

```
                    = to the sum(of all the previous cuts)
814                         else
815                         {
816                             if (cut is RotationCut)
817                             {
818                                 RotationCut c = (RotationCut)cut;
819                                 //
820                                 if (c.getCutAngle() < 0)
821                                 {
822                                     previousPos[indexCounter] =
                    cut.getCutPosition();
823                                 }
824                                 else
825                                 {
826                                     previousPos[indexCounter] =
                    cut.getCutPosition() + this.getCutShaving(cut);
827                                 }
828                             }
829                             else
830                             {
831                                 previousPos[indexCounter] =
                    cut.getCutPosition() + this.getCutShaving(cut);
832                             }
833                             cut.setCutPosition(previousPos.Sum());
834                             previousCut = cut;
835                         }
836                     }
837                     // Its TrimCut or BendCut
838                     else
839                     {
840                         if (cut is TrimCut)
841                         {
842                             // Meaning it is the first cut that needs a
                    trim
843                             if (cut.getCutPosition() == 0)
844                             {
845                                 if(forTrimCut != null)
846                                 {
847                                     cut.setCutPosition(cut.getCutPosition
                    () + forTrimCut.getCutPosition());
848                                 }
849
850                             }
851                             // meaning it is the final cut that needs a
                    trim
852                             else
853                             {
854                                 // Trimcut does not need to handle
                    cutShaving
855                                 cut.setCutPosition(cut.getCutPosition() +
                     previousPos.Sum());
856
857                             }
858                         }
859
860                         else if (cut is BendCut)
```

```
861                                 {
862                                     cut.setCutPosition(cut.getCutPosition() +
                                    previousPos.Sum());
863                                 }
864                             }
865                         }
866
867                         // If counter is larger than the index do the last
                            calculation then break the for loop
868                         else
869                         {
870                             break;
871                         }
872
873                         indexCounter++;
874                         counter++;
875                     }
876                     startIndex += index;
877                     profileLengths[counterC] = p.getLength();
878                     counterC++;
879                 }
880
881                 // List containing the length of OrderedAluProfile
882                 profileLengths = new double[profile.getSortedAluList().getList
                      ().Count];
883                 counterC = 0;
884
885                 index = 0;
886                 startIndex = 0;
887                 // Sumarizing the holes
888                 foreach (OrderedAluProfile p in profile.getSortedAluList
                      ().getList())
889                 {
890                     // Check what type of profile it is to get the hole list
                        length
891
892                     AngleProfile aProf = null;
893                     GuidingProfile gProf = null;
894
895                     if (p is AngleProfile)
896                     {
897                         aProf = (AngleProfile)p;
898                         index = aProf.getSortedHoleList().Count;
899                     }
900                     else
901                     {
902                         gProf = (GuidingProfile)p;
903                         index = gProf.getSortedHoleList().Count;
904                     }
905
906                     counter = 0;
907
908                     // Sumarizing the holes
909                     foreach (Hole hole in holes.getHoleList().Skip(startIndex))
910                     {
911                         if(counter < index)
```

```
912                     {
913                         if (hole.getZPosition() != null)
914                         {
915                             // Sumarising the holes with zposition A
916                             if (hole.getZPosition().Equals("A"))
917                             {
918                                 // The holes on the first OrderedAluProfile is always at it's own position
919                                 if (startIndex != 0 )
920                                 {
921                                     hole.setXAxis(hole.getXAxis() + profileLengths.Sum());
922 
923                                 }
924                             }
925                             // Sumarising the holes with zposition B
926                             else
927                             {
928                                 if (startIndex != 0)
929                                 {
930                                     hole.setXAxis(hole.getXAxis() + profileLengths.Sum());
931                                 }
932                             }
933                         }
934 
935                         else
936                         {
937                             // The holes on the first OrderedAluProfile is always at it's own position
938                             if (startIndex != 0)
939                             {
940                                 hole.setXAxis(hole.getXAxis() + profileLengths.Sum());
941 
942                             }
943                         }
944                     }
945 
946                     // If counter is larger than the index do the last calculation then break the for loop
947                     else
948                     {
949                         break;
950                     }
951 
952                     counter++;
953                 }
954 
955                 // Updating counters an adding the OrderedAluProfiles length to the array
956                 if(aProf != null)
957                 {
958                     profileLengths[counterC] = p.getLength() + this.getCutShaving(aProf.getSortedCutList().Last());
959                 }
```

```
960                 else if (gProf != null)
961                 {
962                     profileLengths[counterC] = p.getLength() +
                        this.getCutShaving(gProf.getSortedCutList().Last());
963                 }
964
965                 startIndex += index;
966                 counterC++;
967             }
968         }
969
970         /// <summary>
971         /// Summarising all the holes and cuts fron the OrderedAluProfile
              objects in the AluminiumProfile into two lists
972         /// one containing the holes and another containing the cuts
973         /// </summary>
974         private void generateHolesAndCuts(AluminiumProfile profile)
975         {
976
977
978             foreach (OrderedAluProfile p in profile.getSortedAluList
                  ().getList())
979             {
980                 if (p is AngleProfile)
981                 {
982                     AngleProfile aP = (AngleProfile)p;
983
984                     foreach (Hole hole in aP.getSortedHoleList())
985                     {
986                         // Generating a new hole object with the same values,
                         so a change in this list wont affect the original list
987                         Hole newHole = new Hole(hole.getZPosition(),
                        hole.getXAxis(), hole.getYAxis());
988                         this.holes.addHole(newHole);
989                     }
990
991                     foreach (Cut cut in aP.getSortedCutList())
992                     {
993                         // Generating a new cut object with the same values,
                        so a change in this list wont affect the original list
994                         if (cut is BendCut)
995                         {
996                             BendCut newCut = new BendCut(cut.getCutPosition
                            ());
997                             this.cuts.addCut(newCut);
998                         }
999                         else if (cut is TrimCut)
1000                        {
1001                            TrimCut c = (TrimCut)cut;
1002                            TrimCut newCut = new TrimCut(c.getCutPosition(),
                            c.getHeightPos(), c.getRotation());
1003                            this.cuts.addCut(newCut);
1004                        }
1005
1006                        else if (cut is AngleCut)
1007                        {
```

```
1008                            AngleCut c = (AngleCut)cut;
1009                            AngleCut newCut = new AngleCut(c.getCutPosition
                    (), c.getCutAngle());
1010                            newCut.setFinalCut(c.isFinalCut());
1011                            this.cuts.addCut(newCut);
1012                        }
1013
1014                        else if (cut is RotationCut)
1015                        {
1016                            RotationCut c = (RotationCut)cut;
1017                            RotationCut newCut = new RotationCut
                    (c.getCutPosition(), c.getCutAngle());
1018                            newCut.setFinalCut(c.isFinalCut());
1019                            this.cuts.addCut(newCut);
1020                        }
1021
1022                    }
1023                }
1024                else if(p is GuidingProfile)
1025                {
1026                    GuidingProfile gP = (GuidingProfile)p;
1027
1028                    foreach (Hole hole in gP.getSortedHoleList())
1029                    {
1030                        // Generating a new hole object with the same values,
                     so a change in this list wont affect the original list
1031                        Hole newHole = new Hole(hole.getZPosition(),
                    hole.getXAxis(), hole.getYAxis());
1032                        this.holes.addHole(newHole);
1033                    }
1034
1035                    foreach (Cut cut in gP.getSortedCutList())
1036                    {
1037                        // Generating a new cut object with the same values,
                    so a change in this list wont affect the original list
1038                        if (cut is BendCut)
1039                        {
1040                            BendCut newCut = new BendCut(cut.getCutPosition
                    ());
1041                            this.cuts.addCut(newCut);
1042                        }
1043                        else if (cut is TrimCut)
1044                        {
1045                            TrimCut c = (TrimCut)cut;
1046                            TrimCut newCut = new TrimCut(c.getCutPosition(),
                    c.getHeightPos(), c.getRotation());
1047                            this.cuts.addCut(newCut);
1048                        }
1049
1050                        else if (cut is AngleCut)
1051                        {
1052                            AngleCut c = (AngleCut)cut;
1053                            AngleCut newCut = new AngleCut(c.getCutPosition
                    (), c.getCutAngle());
1054                            newCut.setFinalCut(c.isFinalCut());
1055                            this.cuts.addCut(newCut);
```

```
1056                        }
1057
1058                        else if (cut is RotationCut)
1059                        {
1060                            RotationCut c = (RotationCut)cut;
1061                            RotationCut newCut = new RotationCut
                        (c.getCutPosition(), c.getCutAngle());
1062                            newCut.setFinalCut(c.isFinalCut());
1063                            this.cuts.addCut(newCut);
1064                        }
1065
1066                    }
1067                }
1068            }
1069        }
1070
1071        /// <summary>
1072        /// Calculating and returning the cut shaving
1073        /// </summary>
1074        /// <param name="ap"></param>
1075        /// <returns>
1076        /// The Cut shaving
1077        /// </returns>
1078        private double getCutShaving(Cut cut)
1079        {
1080            double shaving = 0;
1081
1082            if (cut is AngleCut)
1083            {
1084                AngleCut c = (AngleCut)cut;
1085                shaving = this.cutShaving.calculateShaving(Math.Abs
                  (c.getCutAngle()));
1086            }
1087
1088            else if (cut is RotationCut)
1089            {
1090                RotationCut c = (RotationCut)cut;
1091                shaving = this.cutShaving.calculateShaving(Math.Abs
                  (c.getCutAngle()));
1092            }
1093
1094            return Math.Round(shaving,2,MidpointRounding.AwayFromZero);
1095        }
1096
1097        /// <summary>
1098        /// Initializing the cncArray with emtpy strings
1099        /// </summary>
1100        private void initArray()
1101        {
1102            for(int i = 0; i < this.length; i++)
1103            {
1104                this.cncArray[i] = "";
1105            }
1106        }
1107    }
1108
```

```
1109
1110
1111  }
1112
```

```
1  using RUFOServer.Program.Entity;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace RUFOServer.Program.Main
9  {
10     /// <summary>
11     /// The CutList represents a list or register of the class Cut.
12     /// For example this class can be used to store multiple cuts that needs to
          be done on an object or material.
13     /// </summary>
14     public class CutList
15     {
16         private List<Cut> cuts;
17
18         /// <summary>
19         /// Creating an instance of CutList
20         /// </summary>
21         public CutList()
22         {
23             this.cuts = new List<Cut>();
24         }
25
26         /// <summary>
27         /// Adding a Cut to the CutList
28         /// </summary>
29         /// <param name="cut">
30         /// The object of the class Cut, you want to add to the list
31         /// </param>
32         public void addCut(Cut cut)
33         {
34             this.cuts.Add(cut);
35         }
36
37         /// <summary>
38         /// Removing a Cut for the CutList
39         /// </summary>
40         /// <param name="cut">
41         /// The object of the class Cut, you want to remove from the list
42         /// </param>
43         public void removeCut(Cut cut)
44         {
45             this.cuts.Remove(cut);
46         }
47
48         /// <summary>
49         /// Returning the list of cuts. (CutList)
50         /// </summary>
51         /// <returns>
52         /// The list of cuts. (CutList)
53         /// </returns>
54         public List<Cut> getCuttingList()
55         {
```

```
56              return this.cuts;
57          }
58
59          /// <summary>
60          /// Returning the sorted CutList by accending order according to the
              positioning of each cut.
61          /// Meaning the Cut with the lowest position in millimeters is at the
              top of the list.
62          /// </summary>
63          /// <returns>
64          /// The sorted list by accending order using the positioning for each
              Cut
65          /// </returns>
66          public List<Cut> getSortedList()
67          {
68              this.cuts.Sort((cutA, cutB) =>
69               {
70                  var firstCompare = cutA.getCutPosition().CompareTo
                      (cutB.getCutPosition());
71
72                  if(firstCompare == 0)
73                  {
74                      firstCompare = cutA.isFinalCut().CompareTo(cutB.isFinalCut
                        ());
75                  }
76                  return firstCompare;
77               });
78
79              return this.cuts;
80          }
81      }
82  }
83
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Main
8  {
9      /// <summary>
10     /// The CutShaving class calculates the part of the material
11     /// which is removed because of the thickness of the blade cutting the said
           material.
12     /// </summary>
13     public class CutShaving
14     {
15         private double bladeThickness; // In millimeters
16
17         /// <summary>
18         /// Creating an instance of CutShaving
19         /// </summary>
20         /// <param name="bladeThickness">
21         /// The thickness of the blade
22         /// </param>
23         public CutShaving(double bladeThickness)
24         {
25             this.bladeThickness = bladeThickness;
26         }
27
28         /// <summary>
29         /// Returning the thickness of the saw blade.
30         /// The default value is 0
31         /// </summary>
32         /// <returns>
33         /// The thickness of the saw blade. Default value is 0
34         /// </returns>
35         public double getBladeThickness()
36         {
37             return this.bladeThickness;
38         }
39
40         /// <summary>
41         /// Setting the saw blade thickness in millimeters
42         /// </summary>
43         /// <param name="thickness">
44         /// The thickness in millimeters
45         /// </param>
46         public void setBladeThickness(double thickness)
47         {
48             this.bladeThickness = thickness;
49         }
50
51         /// <summary>
52         /// Calculating and returning the cut shaving in millimeters
53         /// </summary>
54         /// <param name="angle">
55         /// The angle of the cut.
```

```
56          /// </param>
57          /// <returns>
58          /// The cut shaving in millimeters
59          /// </returns>
60          /// <exception cref="NotFiniteNumberException">description
61          /// Angle or blade thickness = 0
62          /// </exception>
63          /// <exception cref="FormatException">description
64          /// Blade thickness is a negative number
65          /// </exception>
66          public double calculateShaving(double angle)
67          {
68              if ((angle == 0) || (this.bladeThickness == 0))
69              {
70                  throw new System.NotFiniteNumberException("Angle = 0. Angle has
                       to be a number:  0 > x > 0\n"+
71                       "Or blade thickness = 0. Blade thickness must be > 0");
72              }
73
74              else if(this.bladeThickness < 0)
75              {
76                  throw new System.FormatException("Blade thickness cannot be a
                      negative number");
77              }
78
79              else if (angle == 90)
80              {
81                  // Converting to radians
82                  double radAngle = (Math.PI / 180) * angle;
83                  double cutShaving = (this.bladeThickness / Math.Sin(radAngle));
84                  return cutShaving;
85              }
86
87              else
88              {
89                  // Converting to radians
90                  double radAngle = (Math.PI / 180) * (angle+90);
91                  double cutShaving = (this.bladeThickness / Math.Sin(radAngle));
92                  return cutShaving;
93              }
94
95          }
96      }
97  }
98
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using RUFOServer.Program.Entity;
7
8  namespace RUFOServer.Program.Main
9  {
10
11     /// <summary>
12     /// The AngleProfile represents an ordered aluminiums angled profile for
         production.
13     /// Such an profile is ordered into a spesific length which requiers x
         amount of cuts.
14     /// In addition to the cutting, such a profile may also need holes for POP
          rivets.
15     /// All this information i stored whithin the class, and can be retracted
         if needed.
16     /// </summary>
17     public class GuidingProfile : OrderedAluProfile
18     {
19         private String type;
20         private CutList cuts;
21         private HoleList holes;
22
23         /// <summary>
24         /// Creating an instance of GuidingProfile
25         /// </summary>
26         /// <param name="length">
27         /// The length of the profile in millimeters
28         /// </param>
29         /// <param name="orderNmb">
30         /// The order number for the profile
31         /// </param>
32         /// <param name="type">
33         /// The profile type
34         /// </param>
35         /// <param name="prodNmb">
36         /// The product number
37         /// </param>
38         /// <param name="itemNmb">
39         /// The item number in the cad drawing
40         /// </param>
41         public GuidingProfile(double length, string orderNmb, String type,
             string prodNmb, string itemNmb) : base(length, orderNmb, prodNmb,
             itemNmb)
42         {
43             this.type = type;
44             this.cuts = new CutList();
45             this.holes = new HoleList();
46         }
47
48         /// <summary>
49         /// Returning the guiding alu profile type
50         /// </summary>
```

```
51          /// <returns>
52          /// The guiding alu profile type
53          /// </returns>
54          public String getType()
55          {
56              return this.type;
57          }
58
59          /// <summary>
60          /// Adding a Cut to the CuttingList
61          /// </summary>
62          /// <param name="cut">
63          /// The object of the class Cut, you want to add to the list
64          /// </param>
65          public void addCut(Cut cut)
66          {
67              this.cuts.addCut(cut);
68          }
69
70          /// <summary>
71          /// Returning the sorted CuttingList by accending order according to
              the positioning of each cut.
72          /// Meaning the Cut with the lowest position in millimeters is at the
              top of the list.
73          /// </summary>
74          /// <returns>
75          /// The sorted list by accending order using the positioning for each
              Cut
76          /// </returns>
77          public List<Cut> getSortedCutList()
78          {
79              return this.cuts.getSortedList();
80          }
81
82          /// <summary>
83          /// Adding a Hole to the HoleList
84          /// </summary>
85          /// <param name="cut">
86          /// The object of the class Hole, you want to add to the list.
87          /// </param>
88          public void addHole(Hole hole)
89          {
90              this.holes.addHole(hole);
91          }
92
93          /// <summary>
94          /// Returning the sorted HoleList by accending order according to the
              x-axis position of each hole.
95          /// Meaning the Hole with the lowest position in the x-axis in
              millimeters is at the top of the list.
96          /// </summary>
97          /// <returns>
98          /// The sorted list by accending order using the x-axis position for
              each Hole
99          /// </returns>
100         public List<Hole> getSortedHoleList()
```

```
101         {
102             return this.holes.getSortedList();
103         }
104 
105         /// <summary>
106         /// Setting the holeList equal to another holeList
107         /// </summary>
108         /// <param name="list">
109         /// An object of the class HoleList
110         /// </param>
111         public void setHoleList(HoleList list)
112         {
113             this.holes = list;
114         }
115     }
116 }
117 
```

```
1  using RUFOServer.Program.Entity;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace RUFOServer.Program.Main
9  {
10
11     /// <summary>
12     /// The HoleList represents a list or register of the class Hole.
13     /// For example this class can be used to store multiple holes that needs
         to be drilled into an object or material.
14     /// </summary>
15     public class HoleList
16     {
17         private List<Hole> holes;
18
19         /// <summary>
20         /// Creating an instance of HoleList
21         /// </summary>
22         public HoleList()
23         {
24             this.holes = new List<Hole>();
25         }
26
27         /// <summary>
28         /// Adding a Hole to the HoleList
29         /// </summary>
30         /// <param name="cut">
31         /// The object of the class Hole, you want to add to the list.
32         /// </param>
33         public void addHole(Hole hole)
34         {
35             this.holes.Add(hole);
36         }
37
38         /// <summary>
39         /// Removing a Hole for the HoleList
40         /// </summary>
41         /// <param name="cut">
42         /// The object of the class Hole, you want to remove frome the list
43         /// </param>
44         public void removeHole(Hole hole)
45         {
46             this.holes.Remove(hole);
47         }
48
49         /// <summary>
50         /// Returning the list of holes. (HoleList)
51         /// </summary>
52         /// <returns>
53         /// The list of holes. (HoleList)
54         /// </returns>
55         public List<Hole> getHoleList()
```

```
56          {
57              return this.holes;
58          }
59
60          /// <summary>
61          /// Returning the sorted HoleList by accending order according to the
              x-axis position of each hole.
62          /// Meaning the Hole with the lowest position in the x-axis in
              millimeters is at the top of the list.
63          /// </summary>
64          /// <returns>
65          /// The sorted list by accending order using the x-axis position for
              each Hole
66          /// </returns>
67          public List<Hole> getSortedList()
68          {
69              this.holes.Sort((cutA, cutB) =>
70              {
71                  var firstCompare = cutA.getXAxis().CompareTo(cutB.getXAxis());
72                  return firstCompare;
73              });
74
75              return this.holes;
76          }
77      }
78  }
79
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Main
8  {
9      /// <summary>
10     /// The OrderedAluList represents a list or register of the class
         OrderedAluProfile.
11     /// It is possible to add and remove OrderedAluList objects from the list.
12     /// </summary>
13     public class OrderedAluList
14     {
15
16         private List<OrderedAluProfile> aluProfiles;
17
18         /// <summary>
19         /// Creating an instance of OrderedAluProfile
20         /// </summary>
21         public OrderedAluList()
22         {
23             this.aluProfiles = new List<OrderedAluProfile>();
24         }
25
26         /// <summary>
27         /// Adding an object of OrderedAluProfile to the OrderedAluList list.
28         /// </summary>
29         /// <param name="profile">
30         /// The object of the type OrderedAluProfile you want to add to the
             list
31         /// </param>
32         public void addProfile(OrderedAluProfile profile)
33         {
34             this.aluProfiles.Add(profile);
35         }
36
37         /// <summary>
38         /// Removing an object of OrderedAluProfile from the OrderedAluList
             list.
39         /// </summary>
40         /// <param name="profile">
41         /// The object of the type OrderedAluProfile you want to remove from
             the list
42         /// </param>
43         public void removeProfile(OrderedAluProfile profile)
44         {
45             this.aluProfiles.Remove(profile);
46         }
47
48         /// <summary>
49         /// Returning the list containing the OrderedAluProfile objects
50         /// </summary>
51         /// <returns>
52         /// The list containing the OrderedAluProfile objects
```

```
53          /// </returns>
54          public List<OrderedAluProfile> getList()
55          {
56              return this.aluProfiles;
57          }
58      }
59  }
60
61
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Main
8  {
9      /// <summary>
10     /// The OrderedAluProfile class is ment to be used as an abstract
         superclass.
11     /// Because all processed aluminium profiles has a length, order number,
         product number and item number.
12     /// </summary>
13     public class OrderedAluProfile
14     {
15         private double length; // In millimeters
16         private String orderNmb;
17         private String prodNmb;
18         private String itemNmb;
19         private bool produced = false;
20
21         /// <summary>
22         /// Creating an instance of OrderedAluProfile.
23         /// <param name="length">
24         /// The length of the profile in millimeters
25         /// </param>
26         /// <param name="orderNmb">
27         /// The order number whom the aluminium profile belongs to
28         /// </param>
29         /// <param name="prodNmb">
30         /// The product number.
31         /// </param>
32         /// <param name="itemNmb">
33         /// The item number in the cad drawing
34         /// </param>
35         /// </summary>
36         public OrderedAluProfile(double length, String orderNmb, String
             prodNmb, String itemNmb)
37         {
38             this.length = length;
39             this.orderNmb = orderNmb;
40             this.prodNmb = prodNmb;
41             this.itemNmb = itemNmb;
42         }
43
44         /// <summary>
45         /// Returning the length of the profile in millimeters
46         /// </summary>
47         /// <returns>
48         /// The length of the profile in millimeters
49         /// </returns>
50         public double getLength()
51         {
52             return this.length;
53         }
```

```
54
55          /// <summary>
56          /// Returning a string, which contains the order number of the profile
57          /// </summary>
58          /// <returns>
59          /// A string, which contains the order number of the profile
60          /// </returns>
61          public String getOrderNmb()
62          {
63              return this.orderNmb;
64          }
65
66          /// <summary>
67          /// Returning the product number as a string
68          /// </summary>
69          /// <returns>
70          /// The product number as a string
71          /// </returns>
72          public String getProdNmb()
73          {
74              return this.prodNmb;
75          }
76
77          /// <summary>
78          /// Returning the item number as a string
79          /// </summary>
80          /// <returns>
81          /// The product item as a string
82          /// </returns>
83          public String getItemNmb()
84          {
85              return this.itemNmb;
86          }
87
88          /// <summary>
89          /// Returning true if the aluminium profile is produced
90          /// </summary>
91          /// <returns>
92          /// Is true if the aluminium profile is produced
93          /// </returns>
94          public bool isProduced()
95          {
96              return this.produced;
97          }
98
99          /// <summary>
100         /// Telling the object that it is produced
101         /// </summary>
102         public void setProduced()
103         {
104             this.produced = true;
105         }
106     }
107 }
108
```

```
 1  using RUFOServer.Program.Entity;
 2  using RUFOServer.Program.Exceptions;
 3  using RUFOServer.Program.Register;
 4  using System;
 5  using System.Collections.Generic;
 6  using System.Globalization;
 7  using System.Linq;
 8  using System.Text;
 9  using System.Threading.Tasks;
10
11  namespace RUFOServer.Program.Main
12  {
13      /// <summary>
14      /// The OrderGenerator class represents the process going from a order
          which is received from a .csv file
15      /// to a generated order of OrderedAluProfiles.
16      /// </summary>
17      public class OrderGenerator
18      {
19          private OrderedAluList orderedProfiles;
20          private AluminiumTypeRegister typeReg;
21          private NumberFormatInfo nmbFormat;
22
23          /// <summary>
24          /// Creating an instance of OrderGenerator which needs an
25          /// AluminiumTypeRegister object to compare the .csv file with the
              profile types the system actually can process
26          ///
27          /// </summary>
28          /// <param name="typeReg">
29          /// An object of the type AluminiumTypeRegister
30          /// </param>
31          public OrderGenerator(AluminiumTypeRegister typeReg)
32          {
33              this.orderedProfiles = new OrderedAluList();
34              this.typeReg = typeReg;
35
36              // Ruleset for parsing double values from the string
37              this.nmbFormat = new NumberFormatInfo();
38              this.nmbFormat.NumberDecimalSeparator = ",";
39              this.nmbFormat.NegativeSign = "-";
40          }
41
42          /// <summary>
43          /// Generating an order by parsing an order received as a String[].
44          /// When generating the order each aluminium profile wil be put into a
                OrderedAluList.
45          /// This list is the returned when the order generation is complete
46          /// </summary>
47          /// <param name="order"></param>
48          /// <returns>
49          /// Returning an object of the type OrderedAluList
50          /// </returns>
51          /// <exception cref="WrongFileFormatException">description</exception>
52          ///  <exception cref="AngleExceededException">description</exception>
53          public OrderedAluList generateOrder(string[] order)
```

```
54          {
55              try
56              {
57                  // Clearing the old order from the list
58                  this.orderedProfiles.getList().Clear();
59
60                  // Parsing each line in order seperate to generate objects
61                  // Skipping the first line since it is just the order format
                      description
62                  foreach (String line in order.Skip(1))
63                  {
64                      //Making sure it does not react to empty lines
65                      if(line.Any())
66                      {
67                          this.parseToObjects(line);
68                      }
69                  }
70                  return this.orderedProfiles;
71              }
72
73              catch(WrongFileFormatException e)
74              {
75                  throw e;
76              }
77
78              catch (AngleExceededException ae)
79              {
80                  throw ae;
81              }
82
83          }
84
85          /// <summary>
86          /// Parsing a string line into OrderedAluProfile objects an adding
              them to the
87          /// OrderedAluList register.
88          /// </summary>
89          /// <param name="line">
90          /// The string line you want to parse
91          /// </param>
92          /// <exception cref="WrongFileFormatException">description</exception>
93          /// <exception cref="AngleExceededException">description</exception>
94          private void parseToObjects(string line)
95          {
96
97              try
98              {
99                  // Splitting the String
100                 string[] orderLine = line.Split(';');
101
102                 // The first 11 positions in the line are a constant
103                 string prodOrderNmb = orderLine[0];
104                 string prodNmb = orderLine[1];
105                 string profileType = orderLine[2];
106                 int numberOfItems = Convert.ToInt16(orderLine[3]);
107                 string itemNmb = orderLine[4];
```

```
108
109                 char[] trim = new char[3] {'E','R','V' };
110                 double posFirstCut = Convert.ToDouble(orderLine[5].Trim(trim),
                       this.nmbFormat);
111                 double RotAngleFirstCut = Convert.ToDouble(orderLine[6].Trim
                      (trim), this.nmbFormat);
112                 double AngleFirstCut = Convert.ToDouble(orderLine[7].Trim
                      (trim), this.nmbFormat);
113                 double posLastCut = Convert.ToDouble(orderLine[8].Trim(trim),
                      this.nmbFormat);
114                 double RotAngleLastCut = Convert.ToDouble(orderLine[9].Trim
                      (trim), this.nmbFormat);
115                 double AngleLastCut = Convert.ToDouble(orderLine[10].Trim
                      (trim), this.nmbFormat);
116
117                 // Checking if the profileType is in the AluminiumTypeRegister
118                 int typeNmb = 0;
119                 this.typeReg.getTypeRegister().TryGetValue(profileType, out
                      typeNmb);
120
121                 // Then it is a guiding profile
122                 if (typeNmb == 1)
123                 {
124                     // Number of objects to generate from this orderLine
125                     for(int x = 0; x < numberOfItems; x++)
126                     {
127                         GuidingProfile profile = new GuidingProfile
                        (posLastCut, prodOrderNmb, profileType, prodNmb, itemNmb);
128
129                         // Adding the first and last cut
130                         Cut cut = defineEndCut(RotAngleFirstCut,
                        AngleFirstCut, posFirstCut);
131                         profile.addCut(cut);
132                         Cut cutTwo = defineEndCut(RotAngleLastCut,
                        AngleLastCut, posLastCut);
133
134                         // This is the final cut of this profile
135                         cutTwo.setFinalCut(true);
136                         profile.addCut(cutTwo);
137
138                         // The rest of the orderLine is generic
139                         for (int i = 11; i < orderLine.Length; i++)
140                         {
141                             // TrimCut
142                             if(orderLine[i].StartsWith("TX"))
143                             {
144                                 // Making sure all the information to create
                                the TrimCut is present
145                                 if ((orderLine[i + 1].StartsWith("TH")) &&
                                (orderLine[i + 2].StartsWith("TR")))
146                                 {
147                                     double pos = Convert.ToDouble(orderLine
                                    [i].Trim('T','X'), this.nmbFormat);
148                                     i++;
149                                     double height = Convert.ToDouble(orderLine
                                    [i].Trim('T', 'H'), this.nmbFormat);
```

```
150                                     i++;
151                                     double rotation = Convert.ToDouble
                    (orderLine[i].Trim('T', 'R'), this.nmbFormat);
152
153                                     profile.addCut(new TrimCut(pos, height,
                    rotation));
154                                 }
155                                 // Information missing, Wrong format
156                                 else
157                                 {
158                                     throw new WrongFileFormatException
                    ("Missing information for TrimCut");
159                                 }
160                             }
161
162                             // BendCut
163                             else if(orderLine[i].StartsWith("LX"))
164                             {
165                                 double pos = Convert.ToDouble(orderLine
                    [i].Trim('L', 'X'), this.nmbFormat);
166                                 profile.addCut(new BendCut(pos));
167                             }
168
169                             // Hole
170                             else if (orderLine[i].StartsWith("AX"))
171                             {
172                                 if(orderLine[i + 1].StartsWith("AY"))
173                                 {
174                                     double xAxis = Convert.ToDouble(orderLine
                    [i].Trim('A', 'X'), this.nmbFormat);
175                                     i++;
176                                     double yAxis = Convert.ToDouble(orderLine
                    [i].Trim('A', 'Y'), this.nmbFormat);
177                                     profile.addHole(new Hole(xAxis, yAxis));
178                                 }
179
180                                 // Information missing, Wrong format
181                                 else
182                                 {
183                                     throw new WrongFileFormatException
                    ("Missing information for Hole");
184                                 }
185                             }
186                         }
187
188                         // Adding the generated profile to the list
189                         this.orderedProfiles.addProfile(profile);
190                     }
191
192                 }
193
194                 // Angle profile
195                 else
196                 {
197                     for (int x = 0; x < numberOfItems; x++)
198                     {
```

```
199                         AngleProfile profile = new AngleProfile(posLastCut,
                    prodOrderNmb, profileType, prodNmb, itemNmb);
200
201                         // Adding the first and last cut
202                         Cut cut = defineEndCut(RotAngleFirstCut,
                    AngleFirstCut, posFirstCut);
203                         profile.addCut(cut);
204                         Cut cutTwo = defineEndCut(RotAngleLastCut,
                    AngleLastCut, posLastCut);
205
206                         // This is the final cut of this profile
207                         cutTwo.setFinalCut(true);
208                         profile.addCut(cutTwo);
209
210                         // The rest of the orderLine is generic
211                         for (int i = 11; i < orderLine.Length; i++)
212                         {
213                             // Only holes after the two cuts
214                             if(orderLine[i].StartsWith("AX"))
215                             {
216                                 if (orderLine[i + 1].StartsWith("AY"))
217                                 {
218                                     double xAxis = Convert.ToDouble(orderLine
                    [i].Trim('A', 'X'), this.nmbFormat);
219                                     i++;
220                                     double yAxis = Convert.ToDouble(orderLine
                    [i].Trim('A', 'Y'), this.nmbFormat);
221                                     profile.addHole(new Hole("A", xAxis,
                    yAxis));
222                                 }
223                                 // Information missing, Wrong format
224                                 else
225                                 {
226                                     throw new WrongFileFormatException
                    ("Missing information for Hole");
227                                 }
228                             }
229
230                             else if (orderLine[i].StartsWith("BX"))
231                             {
232                                 if (orderLine[i + 1].StartsWith("BY"))
233                                 {
234                                     double xAxis = Convert.ToDouble(orderLine
                    [i].Trim('B', 'X'), this.nmbFormat);
235                                     i++;
236                                     double yAxis = Convert.ToDouble(orderLine
                    [i].Trim('B', 'Y'), this.nmbFormat);
237                                     profile.addHole(new Hole("B", xAxis,
                    yAxis));
238                                 }
239                                 // Information missing, Wrong format
240                                 else
241                                 {
242                                     throw new WrongFileFormatException
                    ("Missing information for Hole");
243                                 }
```

```
244                                 }
245                             }
246
247                             // Adding the generated profile to the list
248                             this.orderedProfiles.addProfile(profile);
249                         }
250                     }
251                 }
252
253                 catch (IndexOutOfRangeException ie)
254                 {
255                     throw new WrongFileFormatException("Wrong file format", ie);
256                 }
257
258                 catch (FormatException fe)
259                 {
260                     throw new WrongFileFormatException("Wrong file format", fe);
261                 }
262
263                 catch (OverflowException oe)
264                 {
265                     throw new WrongFileFormatException("Wrong file format", oe);
266                 }
267
268                 catch(AngleExceededException ae)
269                 {
270                     throw ae;
271                 }
272
273             }
274
275             /// <summary>
276             /// The end cut is either the last or first cut that seperates the material from each other.
277             /// This method defines if the end cut is either and RotationCut or AngleCut and returns this cut
278             /// </summary>
279             /// <param name="rotAngle">
280             /// The vertical angle of the cut (The rotation angle) in degrees
281             /// </param>
282             /// <param name="horAngle">
283             /// The horizontal angle of the cut in degrees
284             /// </param>
285             /// <param name="pos">
286             /// The position of the cut in millimeters
287             /// </param>
288             /// <returns>
289             /// A Cut object containing the correct cut object
290             /// </returns>
291             ///
292             /// <exception cref="AngleExceededException">description</exception>
293             private Cut defineEndCut(double rotAngle, double horAngle, double pos)
294             {
295                 Cut cut = null;
296
297                 if((rotAngle == 90 && horAngle == 90))
```

```
298             {
299                 cut = new RotationCut(pos, rotAngle);
300             }
301 
302             else if((rotAngle < 90 && rotAngle >= -45 && rotAngle != 0) &&
                  (horAngle == 90))
303             {
304                 cut = new RotationCut(pos, rotAngle);
305             }
306 
307             else if ((rotAngle == 90) && (horAngle < 90 && horAngle >= -45 &&
                  horAngle != 0))
308             {
309                 cut = new AngleCut(pos, horAngle);
310             }
311 
312             // This cannot happen
313             else if((rotAngle < 90 && rotAngle >= -45) && (horAngle < 90 &&
                  horAngle >= -45))
314             {
315                 throw new AngleExceededException("The cutting station cannot
                      rotate in two axes at the same time");
316             }
317 
318             else if((rotAngle == 0) || (horAngle == 0))
319             {
320                 throw new AngleExceededException("The cut angle cannot be 0
                      degrees. Did you mean 90 Degrees ?");
321             }
322 
323             return cut;
324         }
325     }
326 }
327 
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace RUFOServer.Program.Interface
8  {
9      /// <summary>
10     /// The GUIHandler interface is ment to handle events happening in the GUI, which happens outside the GUI class
11     /// </summary>
12     public interface GUIHandler
13     {
14         /// <summary>
15         /// Updating the orders
16         /// </summary>
17         void updateOrders();
18
19         /// <summary>
20         /// Handling exceptions
21         /// </summary>
22         /// <param name="e"></param>
23         void exceptionHandling(Exception e);
24     }
25 }
26
```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RUFOServer.Program.Exceptions
{
    /// <summary>
    /// This exception represents an error when the an angle is not possible.
    /// For example an angle a robotic arm cannot reach.
    /// </summary>
    class AngleExceededException : Exception
    {
        /// <summary>
        /// Creating an empty instance of AngleExceededException
        /// </summary>
        public AngleExceededException()
        {
        }

        /// <summary>
        /// Creating an empty instance of AngleExceededException
        /// With a message
        /// </summary>
        /// <param name="message">
        /// </param>
        public AngleExceededException(string message)
        : base(message)
        {
        }

        /// <summary>
        /// Creating an empty instance of AngleExceededException
        /// With a message and an inner exception
        /// </summary>
        /// <param name="message"></param>
        /// <param name="inner"></param>
        public AngleExceededException(string message, Exception inner)
        : base(message, inner)
        {
        }
    }
}

```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RUFOServer.Program.Exceptions
{
    /// <summary>
    /// This Exception class represents the wrong format inside a file.
    /// For example a .csv or .txt file which have contet which is not formated in the correct way
    /// </summary>
    public class WrongFileFormatException : Exception
    {
        /// <summary>
        /// Creating an empty instance of WrongFileFormatException
        /// </summary>
        public WrongFileFormatException()
        {
        }

        /// <summary>
        /// Creating an empty instance of WrongFileFormatException
        /// With a message
        /// </summary>
        /// <param name="message">
        /// </param>
        public WrongFileFormatException(string message)
        : base(message)
        {
        }

        /// <summary>
        /// Creating an empty instance of WrongFileFormatException
        /// With a message and an inner exception
        /// </summary>
        /// <param name="message"></param>
        /// <param name="inner"></param>
        public WrongFileFormatException(string message, Exception inner)
        : base(message, inner)
        {
        }
    }
}

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Globalization;
4  using System.IO;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Windows.Forms;
9
10 namespace RUFOServer.Program.Register
11 {
12     /// <summary>
13     /// This class is just a register which reads it's input from
       Profileinfo.txt.
14     /// The class holds the dimension information about the different profile
       types.
15     /// To change size and content of this file, you have to change the .txt
       file.
16     /// </summary>
17     public class AluminiumDimRegister
18     {
19         private Dictionary<String, double> heightReg;
20         private Dictionary<String, double> widthReg;
21         private String path;
22         /// <summary>
23         /// Creating an instance of AluminiumDimRegister, which reads the
             ProfileInfo.txt file
24         /// and fills up the register.
25         /// </summary>
26         public AluminiumDimRegister()
27         {
28             this.heightReg = new Dictionary<string, double>();
29             this.widthReg = new Dictionary<string, double>();
30             getFilePath();
31             readAndFillRegisters();
32         }
33
34         /// <summary>
35         /// Finding the filepath for the ProfileInfo.txt file
36         /// </summary>
37         private void getFilePath()
38         {
39             // Finding the path of the program
40             this.path = Path.GetFullPath(Application.StartupPath);
41             // Going to folders back to find the original files
42             this.path = Path.GetFullPath(Path.Combine(this.path, @"..\..\"));
43             // Goind into the folder containing the original ProfileInfo.txt
44             this.path = this.path + @"Program\Register\ProfileInfo.txt";
45         }
46
47         /// <summary>
48         /// Returning the Dictionary containing the type of profile as key
49         /// and the height of the profile as value
50         /// </summary>
51         /// <returns>
52         /// A Dictionary containing type and height
```

```
53          /// </returns>
54          public Dictionary<String, double> getHeightRegister()
55          {
56              return this.heightReg;
57          }
58
59          /// <summary>
60          /// Returning the Dictionary containing the type of profile as key
61          /// and the width of the profile as value
62          /// </summary>
63          /// <returns>
64          /// A Dictionary containing type and width
65          /// </returns>
66          public Dictionary<String, double> getWidthRegister()
67          {
68              return this.widthReg;
69          }
70
71          /// <summary>
72          /// Updating the register from the ProfileInfo.txt file in the
              register folder
73          /// </summary>
74          public void updateRegister()
75          {
76              this.readAndFillRegisters();
77          }
78
79          /// <summary>
80          /// Reading the ProfileTypes.txt file which contains all the profile
              types,
81          /// and parses this information and puts it into the
              AluminiumTypeRegister
82          /// </summary>
83          private void readAndFillRegisters()
84          {
85              //Making sure the registers are empty before readin in a new one
86              this.heightReg.Clear();
87              this.widthReg.Clear();
88
89              string[] lines = System.IO.File.ReadAllLines(this.path);
90              string[] data;
91
92              // Ruleset for parsing double values from the string
93              NumberFormatInfo provider = new NumberFormatInfo();
94
95              provider.NumberDecimalSeparator = ".";
96              provider.NumberGroupSeparator = ",";
97              provider.NegativeSign = "-";
98
99              for (int i = 2; i < lines.Length; i++)
100             {
101                 data = lines[i].Split(',');
102
103                 if (data.Length == 4)
104                 {
105                     this.heightReg.Add(data[0], Convert.ToDouble(data[3],
```

```
                        provider));
106                     this.widthReg.Add(data[0], Convert.ToDouble(data[2],
                        provider));
107                 }
108
109             }
110         }
111     }
112 }
113
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  using System.Windows.Forms;
8
9  namespace RUFOServer.Program.Register
10 {
11     /// <summary>
12     /// This class is just a register which reads it's input from
         Profileinfo.txt.
13     /// And reads and uses the Profile type information.
14     /// To change size and content of this file, you have to change the .txt
         file.
15     /// </summary>
16     public class AluminiumTypeRegister
17     {
18        private Dictionary<String, int> typeRegister;
19        private String path;
20         /// <summary>
21         /// Creating an instance of AluminiumTypeRegister which reads the
             ProfileInfo.txt file
22         /// and fills up the register.
23         /// </summary>
24         public AluminiumTypeRegister()
25         {
26             this.typeRegister = new Dictionary<string, int>();
27             getFilePath();
28             readAndFillRegister();
29         }
30
31         /// <summary>
32         /// Finding the filepath for the ProfileInfo.txt file
33         /// </summary>
34         private void getFilePath()
35         {
36             // Finding the path of the program
37             this.path = Path.GetFullPath(Application.StartupPath);
38             // Going to folders back to find the original files
39             this.path = Path.GetFullPath(Path.Combine(this.path, @"..\..\"));
40             // Goind into the folder containing the original ProfileInfo.txt
41             this.path = this.path + @"Program\Register\ProfileInfo.txt";
42         }
43
44         /// <summary>
45         /// Returning the list / Dictionary containing the different types of
             aluminium profiles.
46         /// The type of aluminium profile is the "Key" and the "Value" is an
             integer which is either 0 or 1.
47         /// 0 = angled profile and 1 = guiding profile.
48         /// </summary>
49         /// <returns>
50         /// The list / Dictionary containing the different types of aluminium
             profiles.
```

```
51          /// </returns>
52          public Dictionary<String, int> getTypeRegister()
53          {
54              return this.typeRegister;
55          }
56
57          /// <summary>
58          /// Updating the register from the ProfileInfo.txt file in the register
               folder
59          /// </summary>
60          public void updateRegister()
61          {
62              this.readAndFillRegister();
63          }
64
65          /// <summary>
66          /// Reading the ProfileTypes.txt file which contains all the profile
              types,
67          /// and parses this information and puts it into the
              AluminiumTypeRegister
68          /// </summary>
69          private void readAndFillRegister()
70          {
71              // Making sure the register is clear before reading inn a new one
72              this.typeRegister.Clear();
73
74              string[] lines = System.IO.File.ReadAllLines(this.path);
75              string[] data;
76
77              for(int i = 2; i < lines.Length; i++)
78              {
79                  data = lines[i].Split(',');
80
81                  if(data.Length == 4)
82                  {
83                      this.typeRegister.Add(data[0], int.Parse(data[1]));
84                  }
85
86              }
87          }
88      }
89  }
90
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Globalization;
4  using System.IO;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Windows.Forms;
9
10 namespace RUFOServer.Program.Register
11 {
12     /// <summary>
13     /// This class loads and reads the Config.txt file into an list
14     /// The config files consists of: "string, double"
15     /// </summary>
16     public class ConfigLoader
17     {
18         private Dictionary<String, double> config;
19         private string path;
20
21         /// <summary>
22         /// Creating an instance of ConfigLoader
23         /// </summary>
24         public ConfigLoader()
25         {
26             this.config = new Dictionary<string, double>();
27             getFilePath();
28             readAndFill();
29         }
30
31         /// <summary>
32         /// Updating the config list.
33         /// </summary>
34         public void updateConfig()
35         {
36             this.readAndFill();
37         }
38
39         /// <summary>
40         /// Returning the config file as a Dictionary list
41         /// </summary>
42         /// <returns>
43         /// The config file as a Dictionary list
44         /// </returns>
45         public Dictionary<String, double> getConfigFile()
46         {
47             return this.config;
48         }
49
50         /// <summary>
51         /// Returning the value associated with the key.
52         /// If they key does not exist the value returned is 0;
53         /// </summary>
54         /// <param name="key">
55         /// The key you want the associated value from
56         /// </param>
```

```
57          /// <returns>
58          /// The value associated with the key.
59          /// </returns>
60          public double getValue(String key)
61          {
62              double value = 0;
63
64              // Always lowercase
65              this.config.TryGetValue(key, out value);
66              return value;
67          }
68
69          /// <summary>
70          /// Finding the filepath for the Config.txt file
71          /// </summary>
72          private void getFilePath()
73          {
74              // Finding the path of the program
75              this.path = Path.GetFullPath(Application.StartupPath);
76              // Going to folders back to find the original files
77              this.path = Path.GetFullPath(Path.Combine(this.path, @"..\..\"));
78              // Goind into the folder containing the original ProfileInfo.txt
79              this.path = this.path + @"Program\Register\Config.txt";
80          }
81
82
83          /// <summary>
84          /// Reading the Config.txt file,
85          /// and parses this information and puts it into the config list
86          /// </summary>
87          private void readAndFill()
88          {
89              // Making sure the register is clear before reading inn a new one
90              this.config.Clear();
91
92              // Ruleset for parsing double values from the string
93              NumberFormatInfo provider = new NumberFormatInfo();
94              provider.NumberDecimalSeparator = ".";
95              provider.NumberGroupSeparator = ",";
96              provider.NegativeSign = "-";
97
98              string[] lines = File.ReadAllLines(this.path);
99              string[] data;
100
101             for (int i = 0; i < lines.Length; i++)
102             {
103                 data = lines[i].Split(',');
104                 if (data.Length == 2)
105                 {
106                     this.config.Add(data[0], Convert.ToDouble(data[1],
                        provider));
107                 }
108
109             }
110
111         }
```

```
112
113         }
114     }
115
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Diagnostics;
4  using System.IO;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using TwinCAT.Ads;
9
10 namespace RUFOServer.Program.Server
11 {
12     /// <summary>
13     /// The ADSCommunication class has the resposibility to connect to an ADS device,
14     /// and read information from the device or send data to the device.
15     /// </summary>
16     public class ADSCommunication
17     {
18         private int adsPort;
19         private TcAdsClient tcClient;
20         private String netID;
21
22         /// <summary>
23         /// Creating an instance of ADSCommunication
24         /// </summary>
25         /// <param name="netID">
26         /// The netid is the ADS address of the device you want to connect to
27         /// </param>
28         /// <param name="adsPort">
29         /// The port you want to commucate with
30         /// </param>
31         /// </summary>
32         public ADSCommunication(String netID,int adsPort)
33         {
34             this.adsPort = adsPort;
35             this.netID = netID;
36             this.tcClient = new TcAdsClient();
37         }
38
39         /// <summary>
40         /// Connecting to the PLC at the chosen port
41         /// <exception cref="AdsException">description</exception>
42         /// </summary>
43         public void connect()
44         {
45             try
46             {
47                 this.tcClient.Connect(this.netID, this.adsPort);
48             }
49             catch(AdsException err)
50             {
51                 throw new AdsException("An error occurred when connecting to: " + adsPort + err.Source);
52             }
53
54         }
```

```
55
56          /// <summary>
57          /// Disconnecting from the PLC
58          /// </summary>
59          public void disconnect()
60          {
61              this.tcClient.Dispose();
62              this.tcClient.Disconnect();
63          }
64
65          /// <summary>
66          /// Setting the ADS state of the connected device
67          /// </summary>
68          /// <param name="state">
69          /// The AdsState you want to change to
70          /// </param>
71          /// <exception cref="AdsException">description</exception>
72          public void setState(AdsState state)
73          {
74              try
75              {
76                  tcClient.WriteControl(new StateInfo(state, tcClient.ReadState
                      ().DeviceState));
77              }
78                catch (AdsException err)
79              {
80                  throw new AdsException("An error occurred when trying to
                      change the device state " + err.Message);
81              }
82          }
83
84          /// <summary>
85          /// Reading and returning the ADS state of the PLC
86          /// </summary>
87          /// <returns>
88          /// Returning the ADS state of the PLC
89          /// </returns>
90          /// <exception cref="AdsException">description</exception>
91          public String readADSState()
92          {
93              try
94              {
95                  StateInfo info = this.tcClient.ReadState();
96                  return info.AdsState.ToString();
97              }
98              catch (AdsException err)
99              {
100                 throw new AdsException(err.Message);
101             }
102
103         }
104
105         /// <summary>
106         /// Accessing and reading an array on the PLC. This can be an array of
              any type.
107         /// After the array is read from the PLC it is returned.
```

```
108         /// </summary>
109         /// <param name="variable">
110         /// The name of the array variable on the PLC.
111         /// </param>
112         /// <returns>
113         /// Returning an List containing the information from the PLC array
114         /// </returns>
115         /// <exception cref="AdsException">description</exception>
116         public List<String> accessAndReadArray(String variable, int
              arrayLength)
117         {
118             int hVar;
119             List<String> varList = new List<string>();
120 
121             try
122             {
123                 hVar = this.tcClient.CreateVariableHandle(variable);
124 
125                 // AdsStream which gets the data
126                 AdsStream dataStream = new AdsStream(arrayLength * 2);
127                 BinaryReader binRead = new BinaryReader(dataStream);
128 
129                 //read comlpete Array
130                 this.tcClient.Read(hVar, dataStream);
131 
132                 varList.Clear();
133                 dataStream.Position = 0;
134 
135                 for (int i = 0; i < arrayLength; i++)
136                 {
137                     varList.Add(binRead.ReadInt16().ToString());
138                 }
139             }
140 
141             catch (AdsException err)
142             {
143                 throw new AdsException(err.Message);
144             }
145 
146             return varList;
147         }
148 
149         /// <summary>
150         /// Writing to a boolean variable on the plc. Using the input
              variables
151         /// "variable" which is the name of the plc variable, and the boolean
              value
152         /// </summary>
153         /// <param name="variable">
154         /// The name of the variable on the PLC.
155         /// </param>
156         /// <param name="value">
157         /// The boolean value you want the variable to change to. "False" or
              "True"
158         /// </param>
159         /// <exception cref="AdsException">description</exception>
```

```
160         public void writeToBoolean(String variable, bool value)
161         {
162             int hVar;
163 
164             try
165             {
166                 hVar = this.tcClient.CreateVariableHandle(variable);
167 
168                 // Write to variable
169                 this.tcClient.WriteAny(hVar, value);
170             }
171 
172             catch (AdsException err)
173             {
174                 throw new AdsException(err.Message);
175             }
176         }
177 
178         /// <summary>
179         /// Reading an boolean value from the PLC, and returning this value
180         /// </summary>
181         /// <param name="variable">
182         /// The name of the variable on the PLC.
183         /// </param>
184         /// <returns>
185         /// The boolean value from the PLC
186         /// </returns>
187         /// <exception cref="AdsException">description</exception>
188         public bool readBoolean(String variable)
189         {
190             int hVar;
191             bool value = false;
192 
193             try
194             {
195                 hVar = this.tcClient.CreateVariableHandle(variable);
196 
197                 // read
198                 value = (Boolean )this.tcClient.ReadAny(hVar, typeof
                      (Boolean));
199             }
200 
201             catch (AdsException err)
202             {
203                 throw new AdsException(err.Message);
204             }
205 
206             return value;
207         }
208 
209         /// <summary>
210         /// Writing a String array to a String array on the PLC.
211         /// The stringLength needs to be equal to the length of the string
              length in the PLC
212         /// </summary>
213         /// <param name="variable">
```

```
214         /// The variable for the String array on the PLC you want to write to
215         /// </param>
216         /// <param name="array">
217         /// The String array you want to send to the PLC
218         /// </param>
219         /// <param name="StringLength">
220         /// The length of each String element in the Array on the PLC
221         /// </param>
222         /// <exception cref="AdsException">description</exception>
223         public void writeToStringArray(String variable, String[] array, int
              stringLength)
224         {
225
226             try
227             {
228                 int hVar;
229                 int arrayPos = 0;
230
231                 for (int i = 0; i < array.Length; i++)
232                 {
233                     arrayPos = i + 1;
234
235                     //Adding the "[i]" to the array variable name
236                     String var = variable + "[" + arrayPos + "]";
237                     hVar = this.tcClient.CreateVariableHandle(var);
238
239                     AdsStream adsStream = new AdsStream(stringLength);
240                     BinaryWriter writer = new BinaryWriter(adsStream,
                        System.Text.Encoding.ASCII);
241
242                     writer.Write(array[i].ToCharArray());
243                     //add terminating zero
244                     writer.Write('\0');
245                     // Write to the ADS device
246                     this.tcClient.Write(hVar, adsStream);
247                 }
248             }
249
250             catch (AdsException err)
251             {
252                 throw new AdsException(err.Message);
253             }
254
255         }
256
257         /// <summary>
258         /// Writing a String array to a String array on the PLC.
259         /// Using a List of integers which contains the hvar for every
              position in the String array on the PLC
260         /// </summary>
261         /// <param name="hVars">
262         /// A list of integers containing the hvar for every position in the
              String array on the PLC
263         /// </param>
264         /// <param name="array">
265         /// The String array you want to send to the PLC
```

```
266         /// <param name="StringLength">
267         /// The length of each String element in the Array on the PLC
268         /// </param>
269         /// </param>
270         /// <exception cref="AdsException">description</exception>
271         public void writeToStringArray(List<Int64> hVars, String[] array, int
              stringLength)
272         {
273             try
274             {
275                 int i = 0;
276                 foreach (int x in hVars)
277                 {
278                     AdsStream adsStream = new AdsStream(stringLength);
279                     BinaryWriter writer = new BinaryWriter(adsStream,
                        System.Text.Encoding.ASCII);
280 
281                     writer.Write(array[i].ToCharArray());
282                     //add terminating zero
283                     writer.Write('\0');
284                     // Write to the ADS device
285                     this.tcClient.Write(x, adsStream);
286                     i++;
287                 }
288             }
289             catch (AdsException err)
290             {
291                 throw new AdsException(err.Message);
292             }
293 
294         }
295 
296 
297         /// <summary>
298         /// Reading values from an readUInt32 variable on the ADS device.
299         /// And returns this uint value
300         /// </summary>
301         /// <param name="variable"></param>
302         /// <returns>
303         /// Returning an uint
304         /// </returns>
305         public uint readUint(string variable)
306         {
307             try
308             {
309                 int hVar = this.tcClient.CreateVariableHandle(variable);
310 
311                 return (uint)this.tcClient.ReadAny(hVar, typeof(uint));
312             }
313 
314             catch (AdsException err)
315             {
316                 throw new AdsException(err.Message);
317             }
318 
319         }
```

```
320
321         /// <summary>
322         /// Generating an unique handle for an ADS variable and returning this
               as a int
323         /// </summary>
324         /// <param name="varriable"></param>
325         /// <returns>
326         /// Generating an unique handle for an ADS variable and returning this
               as a int
327         /// returns>
328         public Int64 getVariable(String variable)
329         {
330             return this.tcClient.CreateVariableHandle(variable);
331         }
332     }
333 }
334
```

```
using RUFOServer.Program.Entity;
using RUFOServer.Program.Exceptions;
using RUFOServer.Program.Interface;
using RUFOServer.Program.Main;
using RUFOServer.Program.Register;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace RUFOServer.Program.Server
{
    /// <summary>
    /// The OrderHandeler class controls and handles all the order information.
    /// The class contains and controls the lists of angle and guiding aluminium profiles for production.
    /// </summary>
    public class OrderHandler
    {
        private AluProfileOrderList angleOrderList;
        private AluProfileOrderList guidingOrderList;
        private AluProductionOptimizer optimizer;
        private AluminiumTypeRegister typeReg;
        private AluminiumDimRegister dimReg;
        private CutShaving cutShaving;
        private OrderGenerator orderGenerator;
        private bool running;
        private string path;

        // Interfaces
        private GUIHandler guiHandler;

        /// <summary>
        /// Creating an instance of OrderHandeler
        /// </summary>
        public OrderHandler(AluProfileOrderList angleOrderList, AluProfileOrderList guidingOrderList, CutShaving cutShaving, GUIHandler guiHandler)
        {
            this.typeReg = new AluminiumTypeRegister();
            this.dimReg = new AluminiumDimRegister();
            this.angleOrderList = angleOrderList;
            this.guidingOrderList = guidingOrderList;
            this.cutShaving = cutShaving;
            this.orderGenerator = new OrderGenerator(this.typeReg);
            this.optimizer = new AluProductionOptimizer(this.angleOrderList, this.guidingOrderList, this.typeReg, this.dimReg, this.cutShaving);
            this.guiHandler = guiHandler;

```

```
51              getFilePath();
52          }
53
54          /// <summary>
55          /// Starting the OrderHandler thread which watches the
56          /// Orders folder for changes and handles theses changes
57          /// </summary>
58          public void start()
59          {
60              Thread thread = new Thread(new ThreadStart(monitorOrders));
61              thread.IsBackground = true;
62              thread.Start();
63          }
64
65          /// <summary>
66          /// Generating and optimizing the received order
67          /// </summary>
68          /// <param name="order">
69          /// The string containing the order
70          /// </param>
71          public void generateOrder(String[] order)
72          {
73              try
74              {
75                  this.optimizer.optimize(this.orderGenerator.generateOrder
                      (order));
76              }
77              catch(WrongFileFormatException e)
78              {
79                  guiHandler.exceptionHandling(e);
80              }
81              catch (AngleExceededException e)
82              {
83                  guiHandler.exceptionHandling(e);
84              }
85              catch (NotFiniteNumberException e)
86              {
87                  guiHandler.exceptionHandling(e);
88              }
89              catch (FormatException e)
90              {
91                  guiHandler.exceptionHandling(e);
92              }
93
94          }
95
96          /// <summary>
97          /// Set the length of the profiles used in production on production
              line one.
98          /// It is important that this length is set in millimeters.
99          /// </summary>
100         /// <param name="length">
101         /// Length of the production line one profiles in millimeters
102         /// </param>
103         public void setProfileLengthLineOne(double length)
104         {
```

```
105             this.optimizer.setProfileLengthLineOne(length);
106         }
107
108         /// <summary>
109         /// Set the length of the profiles used in production on production
              line two.
110         /// It is important that this length is set in millimeters.
111         /// </summary>
112         /// <param name="length">
113         /// Length of the production line two profiles in millimeters
114         /// </param>
115         public void setProfileLengthLineTwo(double length)
116         {
117             this.optimizer.setProfileLengthLineTwo(length);
118         }
119
120         /// <summary>
121         /// Returning the length of the profiles used in production line one
              in millimeters.
122         /// </summary>
123         /// <returns>
124         /// The length of the profiles used in production line one in
              millimeters.
125         /// </returns>
126         public double getProfileLengthLineOne()
127         {
128             return this.optimizer.getProfileLengthLineOne();
129         }
130
131         /// <summary>
132         /// Returning the length of the profiles used in production line two
              in millimeters.
133         /// </summary>
134         /// <returns>
135         /// The length of the profiles used in production line two in
              millimeters.
136         /// </returns>
137         public double getProfileLengthLineTwo()
138         {
139             return this.optimizer.getProfileLengthLineTwo();
140         }
141
142         /// <summary>
143         /// Returning the AluProfileOrderList which contains the list of angle
               profiles for production
144         /// </summary>
145         /// <returns>
146         /// The AluProfileOrderList which contains the list of angle profiles
              for production
147         /// </returns>
148         public AluProfileOrderList getAngleOrderList()
149         {
150             return this.angleOrderList;
151         }
152
153         /// <summary>
```

```
154         /// Returning the AluProfileOrderList which contains the list of guiding profiles for production
155         /// </summary>
156         /// <returns>
157         /// The AluProfileOrderList which contains the list of guiding profiles for production
158         /// </returns>
159         public AluProfileOrderList getGuidingOrderList()
160         {
161             return this.guidingOrderList;
162         }
163 
164         /// <summary>
165         /// Removing an angled AluminiumProfile object from the list containing the angled AluminiumProfile objects
166         /// </summary>
167         /// <param name="profile">
168         /// The angled AluminiumProfile you want to remove
169         /// </param>
170         public void removeAngleProfile(AluminiumProfile profile)
171         {
172             this.angleOrderList.removeProfile(profile);
173         }
174 
175         /// <summary>
176         /// Removing an guiding AluminiumProfile object from the list containing the guiding AluminiumProfile objects
177         /// </summary>
178         /// <param name="profile">
179         /// The guiding AluminiumProfile you want to remove
180         /// </param>
181         public void removeGuidingProfile(AluminiumProfile profile)
182         {
183             this.guidingOrderList.removeProfile(profile);
184         }
185 
186         /// <summary>
187         /// Updating the registers which holds the profile types the
188         /// production cell can machine and what dimensions these profiles are.
189         /// </summary>
190         public void updateAluRegisters()
191         {
192             this.dimReg.updateRegister();
193             this.typeReg.updateRegister();
194         }
195 
196         /// <summary>
197         /// Monitoring the order folder. And when a order is detected which has to be a .csv file:
198         /// The files are read and an order is generated and shown in the GUI
199         /// </summary>
200         private void monitorOrders()
201         {
202             this.running = true;
203             string fileType = ".csv";
```

```
204             DirectoryInfo directory = new DirectoryInfo(this.path);
205
206             // Stopwatch to slow down the read speed
207             Stopwatch stpWatch = new Stopwatch();
208             stpWatch.Start();
209
210             while(stpWatch.ElapsedMilliseconds < 500)
211             {
212                 // Give the system time to start before reading from the
                      folder
213             }
214             stpWatch.Reset();
215             while (this.running)
216             {
217                 FileInfo[] files = directory.GetFiles("*" + fileType);
218                 // Delete all other file types
219                 FileInfo[] xFiles = directory.GetFiles();
220
221                 if (files.Any())
222                 {
223                     foreach (var f in files)
224                     {
225
226                         // Wait 50 ms
227                         Thread.Sleep(50);
228                         // Reading the order
229                         string[] order = System.IO.File.ReadAllLines
                        (f.FullName);
230                         // Generating the new order
231                         this.generateOrder(order);
232                         //Updating the gui
233                         this.guiHandler.updateOrders();
234
235                     }
236
237                     // After the files are read, delete them
238                     foreach (var f in files)
239                     {
240                         f.Delete();
241                     }
242                 }
243
244                 // These files are deleted right away
245                 else if (xFiles.Any())
246                 {
247                     stpWatch.Reset();
248                     stpWatch.Start();
249
250                     while(stpWatch.ElapsedMilliseconds < 50)
251                     {
252                         // Need to wait so the files are not open so they can
                        be deleted
253                     }
254
255                     stpWatch.Stop();
256
```

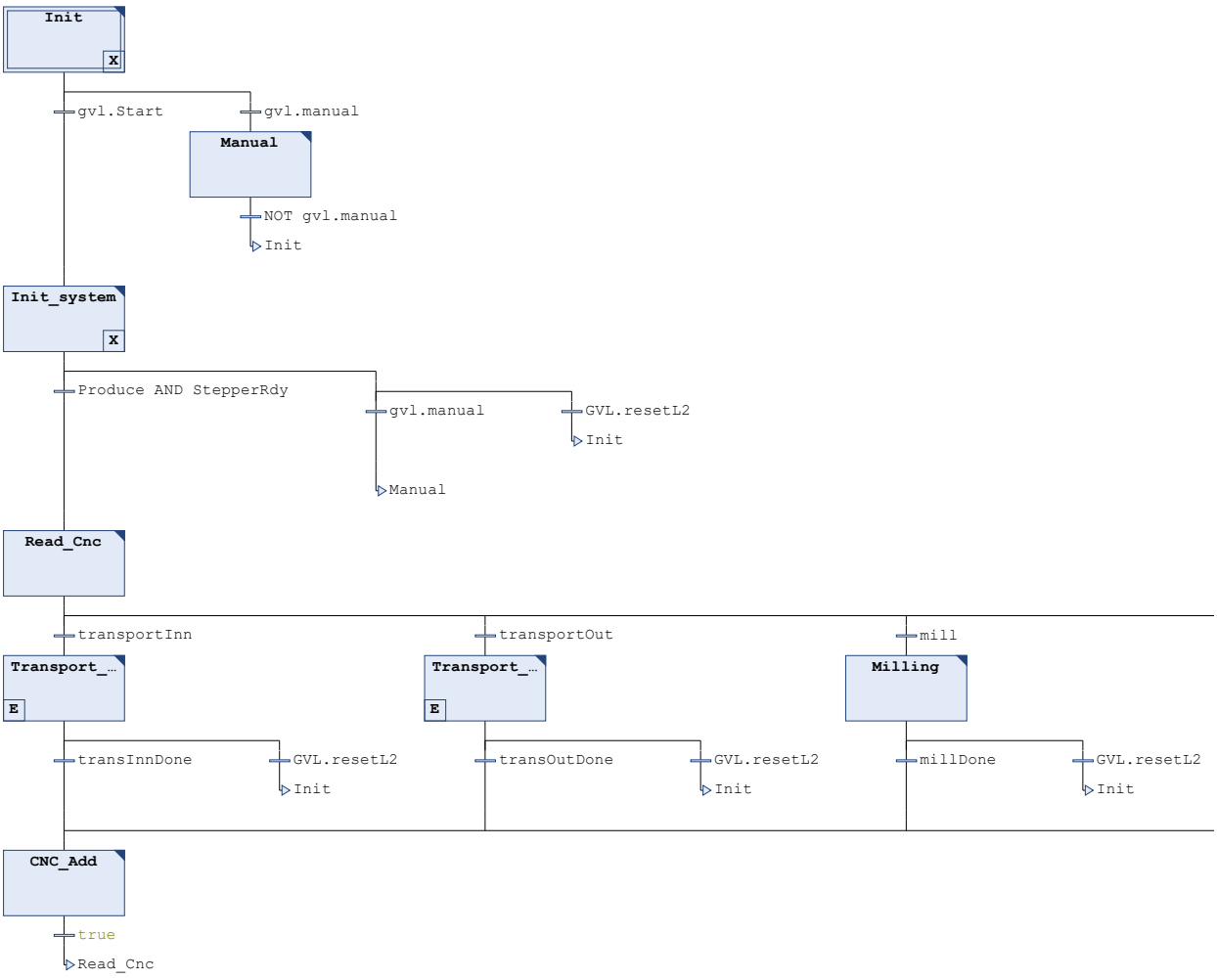
```
257                     foreach (var f in xFiles)
258                     {
259                         if(!f.Name.EndsWith(fileType))
260                         {
261                             f.Delete();
262                         }
263                     }
264                 }
265             }
266         }
267 
268 
269         /// <summary>
270         /// Finding the filepath for the Orders folder file
271         /// </summary>
272         private void getFilePath()
273         {
274             // Finding the path of the program
275             this.path = Path.GetFullPath(Application.StartupPath);
276             // Going to folders back to find the original files
277             this.path = Path.GetFullPath(Path.Combine(this.path, @"..\..\"));
278             // Goind into the folder containing the original ProfileInfo.txt
279             this.path = this.path + @"Orders";
280         }
281     }
282 }
283 
```

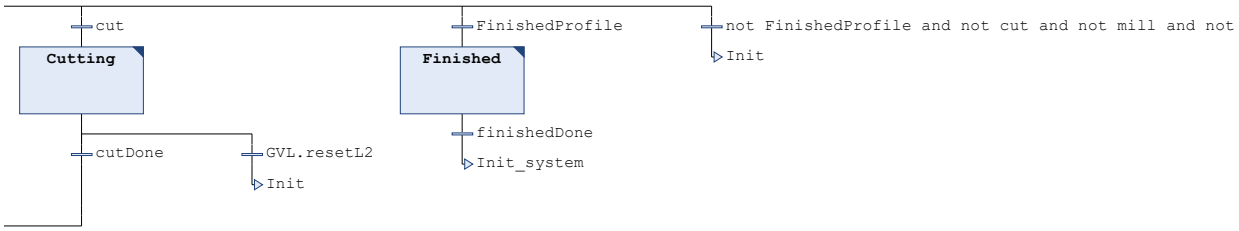
Appendix J

PLC source code

```
PROGRAM Prod_Line_Two
(*
this program is for the guiding profile line.
the program takes a CNC code and bassed on that controll the line or the line can be controlled manualy.
*)

VAR

    // Transitions for the SFC blocks
    transportInn : BOOL ;
    transportOut : BOOL ;
    mill : BOOL ;
    cut : BOOL ;
    transInnDone : BOOL ;
    transOutDone : BOOL ;
    millDone : BOOL ;
    cutDone : BOOL ;
    FinishedProfile : BOOL ;
    finishedDone : bool ;
    Produce : BOOL ;
    StepperRdy : BOOL ;
    ProfileDone : BOOL ;

    //Cnc List:
    CncCode : ARRAY [ 1 .. 1000 ] OF STRING [ 12 ] ;
    CncIndex : INT ;
    Cnc : STRING [ 12 ] ;
    CncCommand : STRING ;
    CncValue : REAL ;


// stepper motor setup for the Feed
    Stpr_Feed_Pwr : MC_POWER ;
    Move_Feed_Stpr : MC_MoveAbsolute ;
    Home_Pos_Feed : LREAL := - 2 ;
    FeedPos : LREAL := 0 ;
    FeedVel : LREAL := 200 ;
    FeedAcc : LREAL := 125.0 ;
    FeedInPOS : BOOL := FALSE ;
    FeedGOToPos : BOOL := FALSE ;
    FeedDec : LREAL := 125.0 ;
    FeedJerk : LREAL := 312.5 ;
    Home_Feed_Stpr : MC_HOME ;
    StprFeedEnable : BOOL ;
    feed_jog : MC_JOG ;
    feedJogFwd : BOOL ;
    feedJogBack : BOOL ;



    // stepper motor setup for the output
    Stpr_Output_Pwr : MC_POWER ;
    Move_Output_Stpr : MC_MoveAbsolute ;
    Home_Pos_Output : LREAL := - 2 ;
    OutputPos : LREAL := 0 ; //mm
    OutputVel : LREAL := 200 ; //mm/s
    OutputAcc : LREAL := 125 ;
    OutputInPOS : BOOL := FALSE ;
    OutputGOToPos : BOOL := FALSE ;
    OutputDec : LREAL := 125 ;
    OutputJerk : LREAL := 312.0 ;
    Home_Output_Stpr : MC_HOME ;
    StprOutputEnable : BOOL ;
    output_jog : MC_JOG ;
    outputJogFwd : BOOL ;
    outputJogBack : BOOL ;

    // stepper motor setup for the drillmovement in Z pos
    Stpr_Mill_Pwr : MC_POWER ;
    Move_Mill_Stpr : MC_MoveAbsolute ;
    Home_Pos_Mill : LREAL := - 2 ;
    MillPos : LREAL := 0 ;
    MillVel : LREAL := 2 ;
    MillAcc : LREAL := 10 ;
    MillInPOS : BOOL := FALSE ;
    MillGOToPos : BOOL := FALSE ;
    MillDec : LREAL := 100 ;
    MillJerk : LREAL := 100 ;
    Home_Mill_Stpr : MC_HOME ;
    StprMillEnable : BOOL ;
    mill_jog : MC_JOG ;
    millJogUp : BOOL ;
    millJogDown : BOOL ;
    millPosOffset : LREAL := 5.3 ; //mm

    Reset_Vel_Stpr : LREAL := 100 ;
    Reset_Acc_Stpr : LREAL := 10 ;
    Reset_Stpr_Done : BOOL := FALSE ;
```

```
91
92          lockTimer  : TON ;
93          lockTimerStart :  BOOL ;
94
95          millTimer :  TON ;
96          millTimerStart :  BOOL  :=  FALSE ;
97
98          // timer for the movement out after drilling
99          millTimer2 :  TON ;
100         // start timer movment out after milling
101         millTimer2Start :  BOOL  :=  FALSE ;
102
103         transportOutTimer  : TON ;
104         transportTimerStart :  BOOL ;
105
106         index :  INT ;
107         CncValue2 :  REAL ;
108
109         // the switch case variable for the motion of the output of the finished profile
110         transportTask :  INT  := 1 ;
111
112         robotReadyRS :  RS ;  // to check that the ready signal for the robot have been present before the done signal.
113     END_VAR
114
```




cut
Cutting
cutDone
GVL.resetL2
Init
FinishedProfile
Finished
finishedDone
Init_system
not FinishedProfile and not cut and not mill and not
Init

```
transportOut and not transportInn
```

```
(*
initialisation of all equipment in production line 2
*)


// Setting all variables back to original values
    transportInn  :=  FALSE ;
    transportOut  :=  FALSE ;
    mill  :=  FALSE ;
    cut  :=  FALSE ;
    transInnDone  :=  FALSE ;
    transOutDone  :=  FALSE ;
    millDone  :=  FALSE ;
    cutDone  :=  FALSE ;
    FinishedProfile  :=  FALSE ;
    Produce  :=  FALSE ;
    StepperRdy :=  FALSE ;
    Gvl . millRunL2  :=  FALSE ;
    ProfileDone  :=  FALSE ;
    GVL . resetL2 := FALSE ;


    transportTask := 1 ;
    GVL . gripper1L2 := FALSE ;
    GVL . gripper2L2 := FALSE ;


    GVL . StprMillEnableL2  :=  TRUE ;
    GVL . StprFeedEnableL2  := TRUE ;
    GVL . StprOutputEnableL2  := TRUE ;


    StprMillEnable  :=  TRUE ;
    StprFeedEnable  := TRUE ;
    StprOutputEnable  := TRUE ;
    //if the robot is in use on this production line, stop it.
IF  NOT  GVL . robotInUseL2  THEN
        gvl . Line  := 0 ;
        gvl . CutType := 0 ;
        gvl . HorAngle := 0 ;
        gvl . VertAngle := 0 ;
        GVL . robotInUseL2  := FALSE ;
END_IF

// sett the calibration flag to perform new calibration.
    GVL . calibrationFlag := TRUE ;
```

```
(*
Manual code for testing the production line in manual.
*)


IF NOT gvl.emcyStop THEN

    // check that not a limitswitch is pressed and stop the stpr if it is pressed.
    IF NOT gvl.limitSwitchMillHome THEN
        millJogDown := FALSE;
    END_IF
    IF NOT gvl.limitSwitchMillUp THEN
        millJogUp := FALSE;
    END_IF
    mill_jog(
    Axis := gvl.Axis2,
    JogForward := millJogUp,
    JogBackwards := millJogDown,
    Mode := E_JogMode.MC_JOGMODE_CONTINOUS,
    Position := MillPos,
    Velocity := MillVel,
    Acceleration := millAcc,
    Deceleration := 1000,
    Jerk := 1000,
    Done => ,
    Busy => ,
    Active => ,
    CommandAborted => ,
    Error => ,
    ErrorID => );

    // check that not a limitswitch is pressed and stop the stpr if it is pressed.
    IF NOT gvl.limitSwitchFeedHome THEN
        feedJogBack := FALSE;
    END_IF
    IF NOT gvl.limitSwitchFeedOut THEN
        feedJogfwd := FALSE;
    END_IF

    feed_jog(
    Axis := gvl.Axis1,
    JogForward := feedJogfwd,
    JogBackwards := feedJogBack,
    Mode := E_JogMode.MC_JOGMODE_CONTINOUS,
    Position := feedPos,
    Velocity := 100,
    Acceleration := feedAcc,
    Deceleration := 1000,
    Jerk := 1000,
    Done => ,
    Busy => ,
    Active => ,
    CommandAborted => ,
    Error => ,
    ErrorID => );
```

```
56
57          // check that not a limitswitch is pressed and stop the stpr if it is
        pressed.
58          IF NOT gvl.limitSwitchOutputHome THEN
59              outputJogBack := FALSE;
60          END_IF
61          IF NOT gvl.limitSwitchOutputOut THEN
62              outputJogfwd := FALSE;
63          END_IF
64          output_jog(
65          Axis := gvl.Axis3,
66          JogForward := outputJogfwd,
67          JogBackwards := outputJogBack,
68          Mode := E_JogMode.MC_JOGMODE_CONTINOUS,
69          Position := outputPos, // not in use in this setting
70          Velocity := 100,
71          Acceleration := outputAcc ,
72          Deceleration := 1000,
73          Jerk := 1000 ,
74          Done => ,
75          Busy => ,
76          Active => ,
77          CommandAborted => ,
78          Error => ,
79          ErrorID => );
80
81
82      END_IF
83
84
85
86
87
88          GVL.calibrationFlag := TRUE;
89
```

```
(*
Initilise the system and calibrate the steppers if they are not calibrated.
Steppers will go to 0 pos.
*)


// stop the drill.
    GVL . millRunL2 := FALSE ;
    GVL . MillYmoveL2  := FALSE ;
    GVL . lockStation1L2 := FALSE ;
    GVL . lockStation2L2 := FALSE ;
    GVL . lockStation3L2 := FALSE ;
    GVL . gripper1L2 := FALSE ;
    GVL . gripper2L2 := FALSE ;
    finishedDone  := FALSE ;
    GVL . robotInUseL2  := FALSE ;
// reset the cnc index
// Set index to 1
    CncIndex  :=  1 ;



IF  NOT  GVL . stopL2  AND  NOT  GVL . emcyStop  THEN

    IF  NOT  GVL . calibrationFlag  THEN
        // power on the feed forward stepper
        Stpr_Feed_Pwr (
        Axis :=  gvl . Axis1  ,
        Enable := StprFeedEnable  ,
        Enable_Positive :=  GVL . limitSwitchFeedOut ,
        Enable_Negative :=  TRUE  ,
        Override :=  ,
        BufferMode :=  ,
        Options :=  ,
        Status =>  ,
        Busy =>  ,
        Active =>  ,
        Error =>  ,
        ErrorID =>  ) ;

        //power on the output stepper motor
        Stpr_Output_Pwr (
        Axis :=  gvl . Axis3  ,
        Enable := StprOutputEnable ,
        Enable_Positive :=  GVL . limitSwitchOutputOut ,
        Enable_Negative :=  TRUE  ,
        Override :=  ,
        BufferMode :=  ,
        Options :=  ,
        Status =>  ,
        Busy =>  ,
        Active =>  ,
        Error =>  ,
        ErrorID =>  ) ;

        //power on the milling hight stepper motor
        Stpr_Mill_Pwr (
```

```
58              Axis := gvl.Axis2 ,
59              Enable := StprMillEnable ,
60              Enable_Positive := GVL.limitSwitchMillUp ,
61              Enable_Negative := TRUE ,
62              Override := ,
63              BufferMode := ,
64              Options := ,
65              Status => ,
66              Busy => ,
67              Active => ,
68              Error => ,
69              ErrorID => ) ;
70
71              //feed stepper motor go home
72              Home_Feed_Stpr (
73              Axis := gvl.Axis1 ,
74              Execute := FeedGOToPos ,
75              Position := 0 ,
76              HomingMode := MC_DefaultHoming ,
77              BufferMode := ,
78              Options := ,
79              bCalibrationCam := NOT gvl.limitSwitchFeedHome ,
80              Done => FeedInPos ,
81              Busy => ,
82              Active => ,
83              CommandAborted => ,
84              Error => ,
85              ErrorID => ) ;
86
87
88
89              // mill stepper home
90              Home_Mill_Stpr (
91              Axis := gvl.Axis2 ,
92              Execute := MillGOToPos ,
93              Position := millPosOffset ,
94              HomingMode := MC_DefaultHoming ,
95              BufferMode := ,
96              Options := ,
97              bCalibrationCam := NOT gvl.limitSwitchMillHome ,
98              Done => MillInPos ,
99              Busy => ,
100             Active => ,
101             CommandAborted => ,
102             Error => ,
103             ErrorID => ) ;
104
105             // output stepper motor home
106             Home_output_Stpr (
107             Axis := gvl.Axis3 ,
108             Execute := OutputGOToPos ,
109             Position := -2 ,
110             HomingMode := MC_DefaultHoming ,
111             BufferMode := ,
112             Options := ,
113             bCalibrationCam := NOT gvl.limitSwitchOutputHome ,
```

```
114             Done => OutputInPOS  ,
115             Busy =>  ,
116             Active =>  ,
117             CommandAborted =>  ,
118             Error =>  ,
119             ErrorID =>  ) ;
120
121
122             // set the motors to do the homing
123             FeedGoToPos := TRUE ;
124             MillGoToPos := TRUE ;
125             OutputGoToPos := TRUE ;
126
127                 // set the calibration flag to true after calibration.
128                 IF  feedINpos  AND  millinpos  AND  outputinpos  THEN
129                 GVL . calibrationFlag  :=  TRUE ;
130                 OutputGoToPos := FALSE ;
131                 MillGoToPos := FALSE ;
132                 FeedGoToPos := FALSE ;
133                 END_IF
134
135         // finished with calibration, sett the feed and output motor to 0
        position
136         ELSE
137             Feedpos  :=  0 ;
138             Move_Feed_Stpr (
139             Axis :=  gvl . axis1  ,
140             Execute :=  FeedGOToPos ,
141             Position :=  FeedPos ,
142             Velocity :=  FeedVel ,
143             Acceleration :=  FeedAcc ,
144             Deceleration := FeedDec  ,
145             Jerk :=  FeedJerk ,
146             BufferMode :=  ,
147             Options :=  ,
148             Done =>  FeedInPos  ,
149             Busy =>  ,
150             Active =>  ,
151             CommandAborted =>  ,
152             Error =>  ,
153             ErrorID =>  ) ;
154             FeedGOToPos := TRUE ;
155
156             OutputPos := 0 ;
157             Move_Output_Stpr (
158             Axis :=  gvl . axis3  ,
159             Execute :=  OutputGOToPos ,
160             Position :=  OutputPos ,
161             Velocity :=  OutputVel ,
162             Acceleration :=  OutputAcc ,
163             Deceleration := OutputDec  ,
164             Jerk :=  OutputJerk ,
165             BufferMode :=  ,
166             Options :=  ,
167             Done =>  OutputInPos  ,
168             Busy =>  ,
```

```
169             Active =>  ,
170             CommandAborted =>  ,
171             Error =>  ,
172             ErrorID =>  ) ;
173             OutputGOToPos := TRUE ;
174 
175                 //ready for production when the motors is calibrated and in 0
    position.
176                 IF  feedINpos  AND  outputinpos  THEN
177                 StepperRdy  :=  TRUE ;
178                 END_IF
179 
180         END_IF
181 
182     // set the signal for moving the motors to false if there is a stop.
183     ELSE
184     FeedGOToPos := FALSE ;
185     outputGOToPos := FALSE ;
186     MillGOToPos := FALSE ;
187 
188     END_IF
189 
190 
191 
192 
193 
194 
195 
196 
197 
```

```
1   (*
2   Read the next command in the CNC list to be done by the machine
3   *)
4   // sett all transitions to false.
5   transInnDone := FALSE ;
6   transOutDone  := FALSE ;
7   millDone  :=  FALSE ;
8   cutDone  := FALSE ;
9   transportInn := FALSE ;
10  transportOut := FALSE ;
11  mill := FALSE ;
12  cut := FALSE ;
13
14  // Load new List
15      Cnc  :=  CncCode [ CncIndex ] ;
16  // Parse String to get command and value
17      CncCommand  :=  LEFT ( Cnc , 2 ) ;
18
19  // get the 2 values from
20  IF  CncCommand  =  'TC'  THEN
21      index  :=  FIND ( STR1 :=  Cnc ,  STR2  :=  'TR' ) ;
22      // get the first value in the cnccode
23      CncValue  :=  STRING_TO_REAL ( MID ( Cnc , index - 3 , 3 ) ) ;
24      // get the seoound value in the cnccode
25      CncValue2  :=  STRING_TO_REAL ( RIGHT ( Cnc , LEN ( Cnc ) - index - 1 ) ) ;  //
    take the length of the string minus the position of the TR. to get out the
    last value in the cnc code.
26
27  ELSE
28      CncValue  :=  STRING_TO_REAL ( RIGHT ( Cnc , LEN ( Cnc ) - 2 ) ) ;
29  END_IF
30
31
32  // Transport inn
33  IF  CncCommand  =  'TI'  THEN
34      transportInn  :=  TRUE ;
35  END_IF
36
37
38  // transport out, finnished with profile piece.
39  IF  CncCommand  =  'TO'  THEN
40      transportOut  :=  TRUE ;
41  END_IF
42
43      // Cut profile
44  IF  CncCommand  =  'TC'  OR  CncCommand  =  'BC'  OR  CncCommand  =  'AC'  OR
    CncCommand  =  'RC'  THEN
45      cut  :=  TRUE ;
46  END_IF
47
48  // milling profile
49  IF  CncCommand  =  'M '  THEN
50      mill := TRUE ;
51  END_IF
52
53
54
```

```
55      // finished with complete profile
56      IF  CncCommand  =  'D '   THEN
57
58          FinishedProfile  :=  TRUE ;
59
60      END_IF
61
62
```

```
(*
transport inn to the machine
*)


// check that the gripper can go to the position.
IF NOT gvl.stopL2 AND NOT GVL.emcyStop THEN
    lockTimer(IN:=lockTimerStart, PT:= T#50MS);
    lockTimerStart := TRUE;
    GVL.lockStation1L2 := FALSE;
    GVL.lockStation2L2 := FALSE;
    GVL.lockStation3L2 := FALSE;
    GVL.gripper2L2 := FALSE;
    GVL.gripper1L2 := TRUE;

// open all locks holding the profile first. //todo change with sensor

    IF (lockTimer.Q) THEN
    //Stepper 1 (Gripper) transport the profile X length
        IF gvl.limitSwitchFeedOut AND gvl.limitSwitchFeedHome THEN
        Move_Feed_Stpr(
        Axis := gvl.axis1,
        Execute := FeedGOToPos,
        Position := FeedPos,
        Velocity := FeedVel,
        Acceleration := FeedAcc,
        Deceleration := FeedDec,
        Jerk := FeedJerk,
        BufferMode := ,
        Options := ,
        Done => FeedInPos,
        Busy => ,
        Active => ,
        CommandAborted => ,
        Error => ,
        ErrorID => );

        feedGOToPos := TRUE;
        ELSE

        FeedGOToPos := FALSE;
        END_IF

        IF FeedInPOS THEN
            FeedInPOS := FALSE;
            FeedGOToPos := FALSE;
            lockTimerStart := FALSE;
            transInnDone := TRUE;
        END_IF
    END_IF

ELSE
FeedGOToPos := FALSE;
END_IF
```

```
(*
transport out of the machine.

*)

// must take the profile before the lockstation3 is released
//release profile then drive out and let go


IF NOT gvl.stopL2 AND NOT GVL.emcyStop THEN
    IF CncValue = 0 THEN

    CASE transportTask OF




    // grip the profile and release all locks
    1:
            GVL.gripper2L2 := TRUE;
        // take of the lock
            gvl.lockStation3L2 := FALSE;
            gvl.lockStation2L2 := FALSE;
            gvl.lockStation1L2 := FALSE;
        //Profile finished
            ProfileDone := TRUE;
            OutputGOToPos := FALSE;
        //must be set back in soulition
            transportOutTimer(in := transportTimerStart, pt := T#50MS);
            transportTimerStart := TRUE;

            IF transportOutTimer.Q THEN
                transportTask := 2;
                transportTimerStart := FALSE;
            END_IF

    // go to the default release pos for the finished profile.
    2:
            OutputPos := 600;
            Move_Output_Stpr(
            Axis := gvl.axis3,
            Execute := OutputGOToPos,
            Position := OutputPos,
            Velocity := OutputVel,
            Acceleration := OutputAcc,
            Deceleration := OutputDec,
            Jerk := OutputJerk,
            BufferMode := ,
            Options := ,
            Done => OutputInPos,
            Busy => ,
            Active => ,
            CommandAborted => ,
            Error => ,
            ErrorID => );
            OutputGOToPos := TRUE;
             IF OutputInPos THEN
```

```
58                      transportTask := 3 ;
59                  END_IF
60
61          //        release the profile
62          3 :
63                  OutputGOToPos := FALSE ;
64                  GVL . gripper2L2 := FALSE ;
65                  transportOutTimer ( in  :=  transportTimerStart , pt :=  T#30MS ) ;
66                  transportTimerStart  :=  TRUE  ;
67                  // when the gripper released move to next case
68                  IF  transportOutTimer . Q  THEN
69                   transportTask := 4 ;
70                   transportTimerStart := FALSE ;
71                  END_IF
72
73          //move motors back to 0 and default position
74          4 :
75                  OutputPos := 0 ;
76                  Move_Output_Stpr (
77                  Axis :=  gvl . axis3  ,
78                  Execute :=  OutputGOToPos ,
79                  Position :=  OutputPos ,
80                  Velocity :=  OutputVel ,
81                  Acceleration :=  OutputAcc ,
82                  Deceleration := OutputDec  ,
83                  Jerk :=  OutputJerk ,
84                  BufferMode :=  ,
85                  Options :=  ,
86                  Done =>  OutputInPos  ,
87                  Busy =>  ,
88                  Active =>  ,
89                  CommandAborted =>  ,
90                  Error =>  ,
91                  ErrorID =>  ) ;
92                  OutputGOToPos := TRUE ;
93                      // make the output stepper go to 0 before going out of the
    block.
94                      IF  OutputInPos  THEN
95                          transportTask := 1 ;
96                          transOutDone := TRUE ;
97                          OutputGOToPos := FALSE ;
98                          transportTimerStart  :=  FALSE ;
99                          GVL . gripper2L2  := FALSE ;
100                     END_IF
101         END_CASE ;
102
103         // if the TO CNC code is not 0, the motor wil go to the position in
    CNCvalue.
104         ELSE
105             gvl . lockStation3L2  :=  FALSE ;
106             gvl . lockStation2L2  :=  FALSE ;
107             gvl . lockStation1L2  :=  FALSE ;
108             gvl . gripper1L2 := FALSE ;
109             gvl . gripper2L2 := TRUE ;
110             transportOutTimer ( in  :=  transportTimerStart , pt :=  T#50MS ) ;
111             transportTimerStart  :=  TRUE  ;
```

```
112             // be sure that the locks is off and the gripper is holding the
        profile.
113             // moving to the setpoint given from the CNCvalue. This is relative
        to the last movement.
114             IF transportOutTimer.Q THEN
115                 Move_Output_Stpr (
116                 Axis := gvl.axis3 ,
117                 Execute := OutputGOToPos ,
118                 Position := OutputPos ,
119                 Velocity := OutputVel ,
120                 Acceleration := OutputAcc ,
121                 Deceleration := OutputDec ,
122                 Jerk := OutputJerk ,
123                 BufferMode := ,
124                 Options := ,
125                 Done => OutputInPos ,
126                 Busy => ,
127                 Active => ,
128                 CommandAborted => ,
129                 Error => ,
130                 ErrorID => ) ;
131                 OutputGOToPos := TRUE ;
132                 //send the feed gripper back to homeposition. ready for new
        profile.
133                 FeedPos := 0 ;
134                 Move_Feed_Stpr (
135                 Axis := gvl.axis1 ,
136                 Execute := FeedGOToPos ,
137                 Position := FeedPos ,
138                 Velocity := FeedVel ,
139                 Acceleration := FeedAcc ,
140                 Deceleration := FeedDec ,
141                 Jerk := FeedJerk ,
142                 BufferMode := ,
143                 Options := ,
144                 Done => FeedInPos ,
145                 Busy => ,
146                 Active => ,
147                 CommandAborted => ,
148                 Error => ,
149                 ErrorID => ) ;
150                 FeedGOToPos := TRUE ;
151                         // output must be in position before going out of the
        block
152                         IF OutputInPos THEN
153                         transOutDone := TRUE ;
154                         OutputGOToPos := FALSE ;
155                         transportTimerStart := FALSE ;
156                         FeedGOToPos := FALSE ;

157                         END_IF
158                 END_IF
159         END_IF
160
161     // if there is a stop signal the motor signal for going is sett to false.
162     ELSE
```

```
163             OutputGOToPos := FALSE ;
164             FeedGOToPos := FALSE ;
165         END_IF
166
167
168
169
170
```

```
1     (*
2     ADD the transport out up to get the profile out.
3     *)
4
5     OutputPos := OutputPos + CncValue;
6
```

```
1      // sett the new distance to be moved for the feed innput.
2      FeedPos  :=  FeedPos  +  CncValue ;
3
```

```
1   (*milling of profile *)
2
3   // check that there is no stop signal.
4   IF NOT gvl.stopL2 AND NOT GVL.emcyStop THEN
5
6   //time for ensuring that the locks is down before milling, to be changed
    with sensor.
7       lockTimer ( IN := lockTimerStart , PT := T#50MS ) ;
8       lockTimerStart := TRUE ;
9       IF gvl.Axis1.NcToPlc.ActPos < 1065 THEN
10              GVL.lockStation1L2 := TRUE ;
11      END_IF
12
13
14  // push the profile in rigth position on the side.
15      GVL.lockStation2L2 := TRUE ;
16
17      //sett the mill position to the value sent from server
18      MillPos := CncValue ;
19
20      //move the mill machine in to position
21      Move_mill_Stpr (
22      Axis := gvl.axis2 ,
23      Execute := MillGOToPos ,
24      Position := MillPos ,
25      Velocity := MillVel ,
26      Acceleration := MillAcc ,
27      Deceleration := MillDec ,
28      Jerk := MillJerk ,
29      BufferMode := ,
30      Options := ,
31      Done => MillInPOS ,
32      Busy => ,
33      Active => ,
34      CommandAborted => ,
35      Error => ,
36      ErrorID => ) ;
37      MillGOToPos := TRUE ;
38
39
40          // check that the milling machine is in right position and that the
    profile is held before milling.
41          IF MillInPOS AND lockTimer.Q THEN
42          GVL.millRunL2 := TRUE ;  //start the drill motor
43          GVL.MillYmoveL2 := TRUE ;  // move the drillmotor in to mill.
44
45          millTimer ( IN := millTimerStart , PT := T#2S ) ;
46          millTimerStart := TRUE ;
47
48              // Check that the milling of the profile is finished and then
    move the milling machine back.
49              IF millTimer.Q THEN
50              GVL.MillYmoveL2 := FALSE ;
51
52              millTimer2 ( IN := millTimer2Start , PT := T#1S ) ;
53              millTimer2Start := TRUE ;
54
```

```
55                      //check that the milling machine is out of the profile
    before relasing the locks.
56                      IF  millTimer2 . Q  THEN
57                      MillGOToPos := false ;
58                      millTimerStart :=  FALSE ;
59                      millTimer2Start := FALSE ;
60                      lockTimerStart := FALSE ;
61                      GVL . millRunL2 := FALSE ;
62                      millDone := TRUE ;
63                      GVL . lockStation1L2 := FALSE ;
64                      GVL . lockStation2L2 := FALSE ;
65                      END_IF
66                  END_IF
67          END_IF
68
69      ELSE
70          //IF there is a stop the mill go to pos must be set to false to be able
    to start the stepper again.
71          MillGOToPos := FALSE ;
72      END_IF
73
74
75
```

```
(*
This is the cutting of the machine. depending on the cnc code the variable
vil change
*)
IF NOT GVL.robotInUseL1 THEN
IF NOT GVL.stopL2 and not GVL.emcyStop THEN
// Hold the profile in position
    GVL.lockStation2L2 := TRUE;
    GVL.lockStation3L2 := TRUE;
    GVL.gripper2L2 := TRUE;
    GVL.robotInUseL2 := TRUE;


// timer for making shure the locks for holding the profile is down
    lockTimer(IN := lockTimerStart, PT := T#30MS);
lockTimerStart := TRUE;
    IF lockTimer.Q THEN
        // check that the robot is ready.
        IF (GVL.KUKA_Ready = 255) THEN
        // set the production line for the robot
        gvl.Line := 2;

            //rotational angle cut
            IF CncCommand = 'RC' THEN
                gvl.cutType := 1;
                gvl.VertAngle := REAL_TO_INT(CncValue * 10);
                gvl.HorAngle := gvl.angle_Offset;
            END_IF

            //vertical angle cut
            IF CncCommand = 'AC' THEN
                gvl.cutType := 1;
                gvl.VertAngle := gvl.angle_Offset;
                gvl.HorAngle := REAL_TO_INT(CncValue * 10);
            END_IF

            // bend cut
            IF CncCommand = 'BC' THEN
                gvl.CutType := 2;
            END_IF

            // trim cut
            IF CncCommand = 'TC' THEN
                gvl.CutType := 3;
                gvl.VertAngle := REAL_TO_INT(CncValue2 * 10);
                gvl.TrimCutHight := REAL_TO_INT(CncValue * 10);
                gvl.HorAngle := gvl.angle_Offset;
            END_IF


        END_IF
        IF gvl.KukaRunSaw = 255 THEN
            gvl.SawRun := TRUE;
        ELSE
            GVL.sawRun := FALSE;
        END_IF
    END_IF
```

```
57
58
59          // set all parameters to 0 when the kuka is finished
60          // check that the kuka is finished and that there have been a ready
        signal before the finish signal.
61          IF  gvl . KUKA_Done  = 255  and  robotReadyRS . Q1  then
62              gvl . Line  := 0 ;
63              gvl . CutType := 0 ;
64              gvl . HorAngle := 0 ;
65              gvl . VertAngle := 0 ;
66              cutDone := TRUE ;
67              lockTimerStart := FALSE ;
68              GVL . robotInUseL2  := FALSE ;
69              GVL . sawRun := FALSE ;
70              // if the cut is a bendcut the lock is released.
71              IF  CncCommand  =  'BC'  THEN
72                  gvl . lockStation2L2 := FALSE ;
73                  gvl . lockStation3L2 := FALSE ;
74              END_IF
75
76
77          END_IF
78          // check that the ready have been before the done signal.
79          robotReadyRS ( SET :=  ( GVL . KUKA_Ready  = 255 ) ,  RESET1 :=  ( gvl .
        KUKA_Done  = 255 ) ,  Q1 =>  ) ;
80
81      END_IF
82      END_IF
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
(*
Finished with the complete profile and set everything back to normal
position.

*)


IF NOT gvl.stopL2 AND NOT GVL.emcyStop THEN
    produce := FALSE;
    FeedPOS := 0;
    OutputPos := 0;

    Move_Feed_Stpr (
    Axis := gvl.axis1 ,
    Execute := FeedGOToPos,
    Position := FeedPos,
    Velocity := FeedVel,
    Acceleration := FeedAcc,
    Deceleration := FeedDec ,
    Jerk := FeedJerk,
    BufferMode := ,
    Options := ,
    Done => FeedInPos ,
    Busy => ,
    Active => ,
    CommandAborted => ,
    Error => ,
    ErrorID => );
    FeedGOToPos := TRUE;

    OutputPos := 0;
    Move_Output_Stpr (
    Axis := gvl.axis3 ,
    Execute := OutputGOToPos,
    Position := OutputPos,
    Velocity := OutputVel,
    Acceleration := OutputAcc,
    Deceleration := OutputDec ,
    Jerk := OutputJerk,
    BufferMode := ,
    Options := ,
    Done => OutputInPos ,
    Busy => ,
    Active => ,
    CommandAborted => ,
    Error => ,
    ErrorID => );
    OutputGOToPos := TRUE;

    IF FeedInPOS AND OutputInPOS THEN
        finishedDone := TRUE;
    END_IF
// sett the move signal to false if there is a stop.
ELSE
        FeedGOToPos := FALSE;
        OutputGOToPos := FALSE;

```

```
57      END_IF
58
```

```
1       // add the CNCindex to the get the next opperation
2       CncIndex  := CncIndex + 1 ;
3       ProfileDone := FALSE ;
4
```

```
PROGRAM MAIN
(*
This is the main program.
 The emergency stop is set here

*)

VAR
    line_one  : BOOL ;
    line_two  : BOOL  := TRUE ;
    stepper_test  : BOOL  := FALSE ;
    stepper_test_final :  BOOL ;
    stepper_test_funk :  BOOL ;


END_VAR
```

```
(*
This program is for functions that need to work for both lines.
*)



// set the stop signal from the robot.
IF  gvl . KUKA_Error  = 0  THEN
    gvl . emcyStop  :=  FALSE ;  //

 ELSE
     GVL . emcyStop  := TRUE ;
 END_IF

    IF  line_one  THEN
    Prod_Line_One ( ) ;
    END_IF

    IF  line_two  THEN
    Prod_Line_Two ( ) ;

    END_IF


IF   Gvl . emcyStop  THEN
    // if there is an error
    gvl . SawRun  := FALSE ;
    gvl . DrilRunLine1A := FALSE ;
    gvl . DrilRunLine1B := FALSE ;
    gvl . millRunL2 :=  FALSE ;
    gvl . KUKA_Stop  := 255 ;


END_IF


    Stepper_Setup ( ) ;
```

```
39
```

```
PROGRAM Stepper_Setup
(*
Set upp all the stepper motors with safety functions.
There is a calibration flag for the power modules because, the homesensor
and the limit switch is the same.
this can be removed if the home sensor is different from the limit switch.
*)

VAR
    StprFeedPwrL2 : MC_POWER ;
    StprOutputPwrL2 : MC_POWER ;
    StprMillPwrL2 : MC_POWER ;

    Stpr_Feed_Reset  : MC_Reset ;
    Stpr_Mill_Reset  : MC_Reset ;
    Stpr_Output_Reset  : MC_Reset ;


    ResetStpr : BOOL ;
    ErrorFeedStprL2 : BOOL ;
    ErrorFeedNmbL2 : UDINT ;


    ErrorOutputStprL2 : BOOL ;
    ErrorOutputNmbL2 : UDINT ;

    ErrorMillStprL2 : BOOL ;
    ErrorMillNmbL2 : UDINT ;

    Feed_Halt_Stpr : MC_Halt ;
    Output_Halt_Stpr : MC_Halt ;
    Mill_Halt_Stpr : MC_Halt ;
END_VAR
```

```

gvl . Axis1 . ReadStatus ( ) ;
gvl . Axis2 . ReadStatus ( ) ;
gvl . Axis3 . ReadStatus ( ) ;


Stpr_Feed_Reset (
    Axis := GVL . Axis1 ,
    Execute := ResetStpr ,
    Done => ,
    Busy => ,
    Error => ,
    ErrorID => ) ;

Stpr_Mill_Reset (
    Axis := GVL . Axis2 ,
    Execute := ResetStpr ,
    Done => ,
    Busy => ,
    Error => ,
```

```
21          ErrorID => ) ;
22
23          Stpr_Output_Reset (
24          Axis := GVL . Axis3 ,
25          Execute := ResetStpr ,
26          Done => ,
27          Busy => ,
28          Error => ,
29          ErrorID => ) ;
30
31          ErrorFeedNmbL2       :=  GVL . Axis1 . NcToPlc . ErrorCode ;
32          ErrorMillNmbL2       :=  GVL . Axis2 . NcToPlc . ErrorCode ;
33          ErrorOutputNmbL2     :=  GVL . Axis3 . NcToPlc . ErrorCode ;
34
35      IF  GVL . calibrationFlag  THEN
36      // power on the feed forward stepper
37          StprFeedPwrL2 (
38          Axis :=  gvl . Axis1  ,
39          Enable := GVL . StprFeedEnableL2  ,
40          Enable_Positive :=  GVL . limitSwitchFeedOut ,
41          Enable_Negative :=  GVL . limitSwitchFeedHome  ,
42          Override :=  ,
43          BufferMode :=  ,
44          Options :=  ,
45          Status =>  ,
46          Busy =>  ,
47          Active =>  ,
48          Error =>  ,
49          ErrorID =>  ) ;
50
51      //        power on the output stepper motor
52          StprOutputPwrL2 (
53          Axis :=  gvl . Axis3  ,
54          Enable := GVL . StprOutputEnableL2 ,
55          Enable_Positive :=  GVL . limitSwitchOutputOut ,
56          Enable_Negative :=  gvl . limitSwitchOutputHome  ,
57          Override :=  ,
58          BufferMode :=  ,
59          Options :=  ,
60          Status =>  ,
61          Busy =>  ,
62          Active =>  ,
63          Error =>  ,
64          ErrorID =>  ) ;
65
66      //        power on the milling hight stepper motor
67          StprMillPwrL2 (
68          Axis :=  gvl . Axis2  ,
69          Enable := GVL . StprMillEnableL2  ,
70          Enable_Positive :=  GVL . limitSwitchMillUp ,
71          Enable_Negative :=  GVL . limitSwitchMillHome  ,
72          Override :=  ,
73          BufferMode :=  ,
74          Options :=  ,
75          Status =>  ,
76          Busy =>  ,
```

```
77          Active =>  ,
78          Error =>  ,
79          ErrorID =>  ) ;
80      END_IF
81
82      Feed_Halt_Stpr (
83          Axis := gvl . Axis1  ,
84          Execute :=  GVL . stopL2  OR  gvl . emcyStop  ,
85          Deceleration := 0  ,
86          Jerk :=  0 ,
87          BufferMode :=  ,
88          Options :=  ,
89          Done =>  ,
90          Busy =>  ,
91          Active =>  ,
92          CommandAborted =>  ,
93          Error =>  ,
94          ErrorID =>  ) ;
95
96
97          Output_Halt_Stpr (
98          Axis := gvl . Axis3  ,
99          Execute :=  GVL . stopL2  OR  GVL . emcyStop ,
100         Deceleration := 0  ,
101         Jerk :=  0 ,
102         BufferMode :=  ,
103         Options :=  ,
104         Done =>  ,
105         Busy =>  ,
106         Active =>  ,
107         CommandAborted =>  ,
108         Error =>  ,
109         ErrorID =>  ) ;
110
111
112         Mill_Halt_Stpr (
113         Axis := gvl . Axis2  ,
114         Execute :=  GVL . stopL2  OR  gvl . emcyStop  ,
115         Deceleration := 0  ,
116         Jerk :=  0 ,
117         BufferMode :=  ,
118         Options :=  ,
119         Done =>  ,
120         Busy =>  ,
121         Active =>  ,
122         CommandAborted =>  ,
123         Error =>  ,
124         ErrorID =>  ) ;
125
126
127     ResetStpr  := FALSE ;
128
```

```
{attribute 'qualified_only'}
VAR_GLOBAL

    Axis1  :  AXIS_REF ;  // feed axis line 2
    Axis2  :  AXIS_REF ;  //mill Z axis line 2
    Axis3  :  AXIS_REF ;  //output movement axis line 2


    limitSwitchFeedHome  AT %I* : BOOL ;
    limitSwitchFeedOut  AT %I* : BOOL ;
    limitSwitchOutputHome  AT %I*  : BOOL ;
    limitSwitchOutputOut  AT %I*  : BOOL ;
    limitSwitchMillHome  AT %I*  : BOOL ;
    limitSwitchMillUp  AT %I*  : BOOL ;
    StprFeedEnableL2  : BOOL ;
    StprOutputEnableL2 : BOOL ;
    StprMillEnableL2 : BOOL ;

    calibrationFlag : BOOL ;


    // MAIN program:
    Stop :  BOOL  :=  FALSE ;
    Start :  BOOL  :=  FALSE ;
    emcyStop  : BOOL ;
    stopL2  : BOOL ;  //the stop flag for line 2
    resetL2  : BOOL ;  // reset the line 2 to initial condition.
    startL2 :  BOOL  :=  FALSE ;  // start the line 2
    manual  : bool ;

    // flag for telling if the produksion line 2 is using the robot. True if
the robot is in use
    robotInUseL2 :  BOOL ;
    // flag for telling if the produksion line 1 is using the robot. True if
the robot is in use
    robotInUseL1 :  BOOL ;



    // physical output
    sawRun  AT %Q*  : BOOL  := FALSE ;  // the start signal for the saw mounted
on robot.
    millRunL2  AT %Q*  : BOOL  := FALSE ;  // the run signal for dril line 2
    DrilRunLine1A   AT %Q*  : BOOL  := FALSE ;  // the run signal for dril line
1 A side
    DrilRunLine1B   AT %Q*  : BOOL  := FALSE ; // the run signal for dril line 1
B side

    // Input and Output KUKA
    Angle_Offset :  INT  :=  900 ;
    //Input
    KUKA_Error  AT %I*  : USINT  ;
    KukaRunSaw  AT %I*  : USINT ;
    KUKA_Done  AT %I*   : USINT ;
    KUKA_Ready  AT %I*  : USINT ;
    // Outputs
    Line  AT %Q* :  USINT ;
```

```
53          CutType  AT %Q* :  USINT ;
54          VertAngle  AT %Q* :  DINT ;
55          HorAngle  AT %Q* :  DINT ;
56          TrimCutHight  AT %Q* :  DINT ;
57          KUKA_Stop  AT %Q*  : USINT ;
58
59
60
61
62          // Output compressed air terminals
63          lockStation1L2  AT %Q* : BOOL ;  // line 2
64          lockStation2L2  AT %Q* : BOOL ;  // line 2
65          lockStation3L2  AT %Q* : BOOL ;  // line 2
66          millYmoveL2  AT %Q*  : BOOL ;  // the movement in y direktion this is the
        drilling.
67          gripper1L2  AT %Q*  : BOOL ;  // gripper for feeder.
68          gripper2L2  AT %Q*  : BOOL ;  // Gripper for the output movement
69
70
71
72          // gamle variabler
73          lockStation_2 :  BOOL ;
74          lockStation_1 :  BOOL ;
75          lockStation_3 :  BOOL ;
76          Homepos_Gripper :  LREAL ;
77          //variabler brukt i linje 1
78          KUKA_Home_Position :  BOOL   :=  FALSE ;
79          KUKA_Rdy :  BOOL  :=  FALSE ;
80          Axis4  :  AXIS_REF ;
81          Homepos  AT %I*     : BOOL  ;
82          Gripper_Sensor :  BOOL ;
83          Timer_Start :  BOOL ;
84
85
86
87
88      END_VAR
89
```

Appendix K

Robot KRL source code

KR C I/Os / I/Os / Digital Inputs
Fieldbusses / KUKA Extension Bus (SYS-X44) / EK1100 EtherCAT Coupler (2A E-Bus) / EL6692 EtherCAT Bridge terminal (Primary)

Name	Type	Description	Name	Description
$IN[1]#G	USINT		IO Inputs.Input 1	
$IN[9]#G	USINT		IO Inputs.Input 2	
$IN[17]#G	DINT		IO Inputs.Input 3	
$IN[49]#G	DINT		IO Inputs.Input 4	
$IN[81]#G	USINT		IO Inputs.Input 11	
$IN[89]#G	USINT		IO Inputs.Input 12	
$IN[97]#G	USINT		IO Inputs.Input 13	
$IN[105]#G	USINT		IO Inputs.Input 14	
$IN[113]#G	USINT		IO Inputs.Input 15	

KR C I/Os / I/Os / Digital Outputs
Fieldbusses / KUKA Extension Bus (SYS-X44) / EK1100 EtherCAT Coupler (2A E-Bus) / EL6692 EtherCAT Bridge terminal (Primary)

Name	Type	Description	Name	Description
$OUT[363]#G	USINT		IO Outputs.Output 1	
$OUT[371]#G	USINT		IO Outputs.Output 2	
$OUT[379]#G	USINT		IO Outputs.Output 3	
$OUT[387]#G	USINT		IO Outputs.Output 4	

WorkVisual project:	IBRvs1 (1) [1.181.0.0]
Safety product option:	Default
Controller:	Controller 1

Safety configuration

Description:	-
Checksum:	B14DDA4E
Last changed:	28.06.2017 15.52
Version:	1

Hardware options

Customer interface	automatic
Input signal for peripheral contactor (US2)	not used
Operator safety acknowledgment	external unit

Global parameters

cartesian monitoring	enabled
Safe monitoring	disabled

By signing, the signatory confirms the correct and complete performance of the safety acceptance procedure in accordance with the operating instructions.

Place, date	Signature
Place, date	Signature
Place, date	Signature

```
&ACCESS RVO1
&COMMENT The motion for the cutting of the angle profile
DEFDAT angleProfile PUBLIC



DECL E6POS XAngleProfileBase={X 0,Y 0,Z 100,A 0,B 0,C 0,S 6,T 59,E1 0.0,E2
0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FAngleProfileBase={TOOL_NO 1,BASE_NO 1,IPO_FRAME #BASE,POINT2[] "
",TQ_STATE FALSE}
DECL PDAT PPDAT0={VEL 100,ACC 100,APO_DIST 100,APO_MODE #CPTP,GEAR_JERK 50}
DECL FDAT FbeginningPosCut={TOOL_NO 1,BASE_NO 1,IPO_FRAME #BASE,POINT2[] "
",TQ_STATE FALSE}
DECL LDAT LCPDAT0={VEL 1,ACC 100,APO_DIST 100,APO_FAC 50,AXIS_VEL 100.0,AXIS_ACC
100.0,ORI_TYP #CONSTANT,CIRC_TYP #BASE,JERK_FAC 50.0,GEAR_JERK 50.0,EXAX_IGN 0}
DECL BASIS_SUGG_T LAST_BASIS={POINT1[] "endPosCut",POINT2[]
"endPosCut",CP_PARAMS[] "CPDAT5",PTP_PARAMS[] "PDAT1",CONT[] " ",CP_VEL[]
"1.0",PTP_VEL[] "100",SYNC_PARAMS[] "SYNCDAT ",SPL_NAME[] "S0 "}
DECL FDAT FendPosCut={TOOL_NO 1,BASE_NO 1,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE
FALSE}
DECL LDAT LCPDAT5={VEL 1.0,ACC 100,APO_DIST 100,APO_FAC 50,AXIS_VEL
100.0,AXIS_ACC 100.0,ORI_TYP #CONSTANT,CIRC_TYP #BASE,JERK_FAC 50.0,GEAR_JERK
50.0,EXAX_IGN 0}
ENDDAT
```

```
&ACCESS RVO1
&COMMENT The motion for the cutting of the angle profile
DEF angleProfile()
;this program is not finished and is only a template to test motion.



E6POS beginnigPosCut
E6POS endPosCut
REAL Zabove
REAL Xabove
REAL Zbelove
REAL Xbelove
Zabove = XAngleProfileBase.z
Xabove =XAngleProfileBase.X
Zbelove = 50 ; length from the top of the profile to the bottom.
Xbelove = 0 ; this is changed based on the angle for the cut
; go to base for angle profile
;

;FOLD PTP AngleProfileBase  Vel=100 % PDAT0 Tool[1] Base[1];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:AngleProfileBase, 3:, 5:100,
7:PDAT0
$BWDSTART=FALSE
PDAT_ACT=PPDAT0
FDAT_ACT=FAngleProfileBase
BAS(#PTP_PARAMS,100)
PTP XAngleProfileBase
;ENDFOLD

; start saw
sawRun = 255;

Xabove =Zabove*(TAN((vAngle/INT_TO_REAL_DIVIDER)-ANGLE_OFFSET))
beginnigPosCut = XAngleProfileBase
beginnigPosCut.X=XAngleProfileBase.x -Xabove
beginnigPosCut.B = XAngleProfileBase.B +
(vAngle/INT_TO_REAL_DIVIDER)-ANGLE_OFFSET

; go to start cut pos
;FOLD LIN beginningPosCut  Vel=2 m/s CPDAT0 Tool[1] Base[1];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:beginningPosCut, 3:, 5:2, 7:CPDAT0
$BWDSTART=FALSE
LDAT_ACT=LCPDAT0
FDAT_ACT=FbeginningPosCut
BAS(#CP_PARAMS,2)
LIN beginnigPosCut
;ENDFOLD

endPosCut = XAngleProfileBase
endPosCut.B=beginnigPosCut.B; beginning angle of tool = end angle of tool
Xbelove =-Zbelove*(TAN((vAngle/INT_TO_REAL_DIVIDER)-ANGLE_OFFSET)); change the
```

```
sign from the above side
endPosCut.z = XAngleProfileBase.z -Zbelove-Zabove
endPosCut.x =XAngleProfileBase.x -Xbelove

; make the cut
;FOLD LIN endPosCut  Vel=1.0 m/s CPDAT5 Tool[1] Base[1];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:endPosCut, 3:, 5:1.0, 7:CPDAT5
$BWDSTART=FALSE
LDAT_ACT=LCPDAT5
FDAT_ACT=FendPosCut
BAS(#CP_PARAMS,1.0)
LIN endPosCut
;ENDFOLD

 ; go back to the beginning point
 ;FOLD LIN beginningPosCut  Vel=1 m/s CPDAT0 Tool[1] Base[1];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:beginningPosCut, 3:, 5:2, 7:CPDAT0
$BWDSTART=FALSE
LDAT_ACT=LCPDAT0
FDAT_ACT=FbeginningPosCut
BAS(#CP_PARAMS,2)
LIN beginnigPosCut
;ENDFOLD



sawRun=0; stop the saw
cutDone=255; sett that the cut is finished for the PLC
END
```

```
&ACCESS RVO1
&REL 1
&COMMENT the motion for the cutting of the guiding profile
&PARAM SUPPRESS = AccessBeforeInitialization
DEFDAT  guidingprofile PUBLIC

DECL POS XguidProfileBase={X 0.0,Y 0.0,Z 50.0000,A 90,B 0,C 0,S 'B010',T
'B100011'} ;,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FTCP={TOOL_NO 6,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL FDAT Fbase={TOOL_NO 6,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE
FALSE}
DECL FDAT FbendCut2={TOOL_NO 5,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE
FALSE}
DECL FDAT FbendCut1={TOOL_NO 7,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE
FALSE}
DECL PDAT PPDAT2={VEL 90.000,ACC 100.000,APO_DIST 100.000,APO_MODE
#CPTP,GEAR_JERK 50.0000}
DECL LDAT setPosDat={VEL 1.00000,ACC 100.000,APO_DIST 100.000,APO_FAC
50.0000,AXIS_VEL 100.000,AXIS_ACC 100.000,ORI_TYP #VAR,CIRC_TYP #BASE,JERK_FAC
50.0000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL LDAT setPosConstOri={VEL 1.00000,ACC 100.000,APO_DIST 100.000,APO_FAC
50.0000,AXIS_VEL 100.000,AXIS_ACC 100.000,ORI_TYP #CONSTANT,CIRC_TYP
#BASE,JERK_FAC 50.0000,GEAR_JERK 50.0000,EXAX_IGN 0}

DECL LDAT cutDat={VEL 0.00500,ACC 0.500,APO_DIST 5.000,APO_FAC 5.0000,AXIS_VEL
1.0,AXIS_ACC 1,ORI_TYP #CONSTANT,CIRC_TYP #BASE,JERK_FAC 50.0000,GEAR_JERK
50.0000,EXAX_IGN 0}


ENDDAT
```

```
&ACCESS RVO1
&REL 1
&COMMENT the motion for the cutting of the guiding profile
&PARAM SUPPRESS = AccessBeforeInitialization
DEF guidingProfile ( )
   POS guidProfileBase
   POS guidProfileEnd
   REAL Zabove; length from base to start heigth
   REAL Xabove; length base to start pos in x direction
   REAL Zbelove; length belove the base.
   real Xbelove; length belove in x axis the robot is moving
   real vCutAngle ; vertical cut angle
   real hCutAngle ;horizontal cut angle

   ; set the right angles for the cut, 0 degrees is sent as 90degrees
       if (vAngle/INT_TO_REAL_DIVIDER)-ANGLE_OFFSET == 0 THEN
       vCutAngle =0;
       else
       vCutAngle =-vAngle/INT_TO_REAL_DIVIDER;
       ENDIF

       ;set the right angles for the cut, 0 degrees is sent as 90degrees
       if (hAngle/INT_TO_REAL_DIVIDER)-ANGLE_OFFSET == 0 THEN
       hCutAngle =0;
       else
       hCutAngle=hAngle/INT_TO_REAL_DIVIDER;
       ENDIF


   Zabove=XguidProfileBase.Z
   Zbelove=-10; mm

; go to the base position 100mm above the profiles base, ready to do the cut
   ;FOLD PTP guidProfileBase  Vel=100 % PDAT2 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:guidProfileBase, 3:, 5:100,
7:PDAT2
      $BWDSTART=FALSE
      PDAT_ACT=PPDAT2
      FDAT_ACT=Fbase
      BAS(#PTP_PARAMS,90)
      PTP XguidProfileBase
   ;ENDFOLD


   guidProfileBase=XguidProfileBase
   guidProfileEnd=XguidProfileBase


   ;regular cut, horisontal or vertical
   if (cutType== 1) then

   ; If the vertical angle is larger then 0 the robot must take a 180 degree
```

```
turn around the A orientation of the tool to not crash.
   if vCutAngle>0 then
    guidProfileBase.A= XguidProfileBase.A + hCutAngle- 180
    guidProfileEnd.A= XguidProfileBase.A + hCutAngle- 180
    Xabove=Zabove*TAN (vCutAngle)
    Xbelove=Zbelove*TAN (vCutAngle)
    guidProfileEnd.C= XguidProfileBase.C -vCutAngle
    guidProfileBase.C= XguidProfileBase.C -vCutAngle

   else
   guidProfileBase.A= XguidProfileBase.A + hCutAngle
   guidProfileEnd.A= XguidProfileBase.A + hCutAngle
   Xabove=Zabove*TAN (vCutAngle)
   Xbelove=Zbelove*TAN (vCutAngle)
   guidProfileEnd.C= XguidProfileBase.C +vCutAngle
   guidProfileBase.C= XguidProfileBase.C +vCutAngle
   ENDIF


    ; rotate the tool to the right angles for the cut.
       ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosDat
         FDAT_ACT=FTCP
        BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD



      guidProfileBase.X = XguidProfileBase.x + Xabove
       ; move to the x pos before the cutting
      ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosConstOri
         FDAT_ACT=Fbase
        BAS(#CP_PARAMS,1)
         LIN guidProfileBase
      ;ENDFOLD







      guidProfileEnd.X = XguidProfileBase.x +Xbelove
      guidProfileEnd.Z= Zbelove
```

```
      ;start the saw
       sawRun=255;
       wait sec 0

      ; do the cut
      ;FOLD LIN guidProfileEnd  Vel=1.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileEnd, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=Fbase
         BAS(#CP_PARAMS,1.0)
         LIN guidProfileEnd
      ;ENDFOLD

      ;go back to where the cut started
      ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=Fbase
         BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD
       cutDone=255
      ; if the angle is larger then 0 the robot is crashing in the saw so move
up to avoid this.
      if vCutAngle >0 then
         guidProfileBase.z =XguidProfileBase.z +100
         guidProfileBase.y=XguidProfileBase.y +100

          ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosDat
         FDAT_ACT=FTCP
        BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD

      ENDIF
     ; move back to the beginnig base of the guid profile
       ;FOLD PTP guidProfileBase  Vel=90 % PDAT2 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:guidProfileBase, 3:, 5:100,
7:PDAT2
      $BWDSTART=FALSE
      PDAT_ACT=PPDAT2
      FDAT_ACT=Fbase
      BAS(#PTP_PARAMS,90)
      PTP XguidProfileBase
```

```
   ;ENDFOLD

   endIF


    ;bend cut
   if (cutType== 2) then
      guidProfileBase =XguidProfileBase

      ; move to the base above the profile
      ;FOLD PTP guidProfileBase  Vel=100 % PDAT2 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:guidProfileBase, 3:, 5:100,
7:PDAT2
      $BWDSTART=FALSE
      PDAT_ACT=PPDAT2
      FDAT_ACT=Fbase
      BAS(#PTP_PARAMS,90)
      PTP XguidProfileBase
   ;ENDFOLD
      guidProfileBase.a=XguidProfileBase.a+45;
      guidProfileBase.y=XguidProfileBase.Y-0; The movement of how much of the
profile is left, to make the wall smaler make smaller number.
      guidProfileBase.Z=guidProfileBase.Z+100

        ; move to the cut position
        ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosDat
         FDAT_ACT=FbendCut1
        BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD

   ;begin saw
   sawRun=255;
   wait sec 0
      ;do the cut
      guidProfileBase.Z=-10
       ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=FbendCut1
        BAS(#CP_PARAMS,0.5)
         LIN guidProfileBase
      ;ENDFOLD
        guidProfileBase.Z=XguidProfileBase.Z+100
        ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
```

```
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=FbendCut1
        BAS(#CP_PARAMS,1.0)
         LIN guidProfileBase
      ;ENDFOLD
      sawRun = 0; Stop the saw
      wait sec 0
      ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosDat
         FDAT_ACT=FTCP
        BAS(#CP_PARAMS,2.0)
         LIN XguidProfileBase
      ;ENDFOLD

      ;First cut of the bend cut finished.
    ; move back to base above profile.
       ;FOLD PTP guidProfileBase  Vel=100 % PDAT2 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:guidProfileBase, 3:, 5:100,
7:PDAT2
      $BWDSTART=FALSE
      PDAT_ACT=PPDAT2
      FDAT_ACT=Fbase
      BAS(#PTP_PARAMS,90)
      PTP XguidProfileBase
   ;ENDFOLD
  guidProfileBase= XguidProfileBase
  guidProfileBase.Z=XguidProfileBase.Z+130
  guidProfileBase.A=XguidProfileBase.a-45;
  guidProfileBase.Y=XguidProfileBase.Y-0; The movement of how much of the
profile is left, to make the wall smaler make smaller number.
; move to the cut position
;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[5] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosDat
         FDAT_ACT=FbendCut2
        BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD

   ;begin saw
   sawRun=255;
   wait sec 0

      guidProfileEnd=guidProfileBase
      guidProfileEnd.z=-4
```

```
      ; do the cut
         ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[5] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=FbendCut2
        BAS(#CP_PARAMS,1.0)
         LIN guidProfileEnd
      ;ENDFOLD

        ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[5] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=FbendCut2
        BAS(#CP_PARAMS,1.0)
         LIN guidProfileBase
      ;ENDFOLD
      ;cut is finished
      sawRun = 0; Stop the saw
      cutDone=255;  cut is finished
      guidProfileBase=XguidProfileBase
      guidProfileBase.Z=XguidProfileBase.Z+30

       ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosDat
         FDAT_ACT=FTCP
         BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD
   ENDIF

   ;trim cut
   if (cutType== 3) then

      ; need hight to end point for this cut. must be sent from plc
      Zabove=XguidProfileBase.Z
      guidProfileEnd= XguidProfileBase

      ;if the trim cut is with positiv angle the saw must be turn 180 degrees on
the A orientation to not crash
   if vCutAngle>0 then
    guidProfileBase.A= XguidProfileBase.A - 180
    guidProfileEnd.A= XguidProfileBase.A - 180
    Xabove=(Zabove-(trimHightZ/INT_TO_REAL_DIVIDER))*TAN (vCutAngle)
    guidProfileEnd.C= XguidProfileBase.C -vCutAngle
    guidProfileBase.C= XguidProfileBase.C -vCutAngle
```

```
   else
    guidProfileBase.A= XguidProfileBase.A + hCutAngle
    guidProfileEnd.A= XguidProfileBase.A + hCutAngle
    Xabove=(Zabove-(trimHightZ/INT_TO_REAL_DIVIDER))*TAN (vCutAngle)
    guidProfileEnd.C= XguidProfileBase.C +vCutAngle
    guidProfileBase.C= XguidProfileBase.C +vCutAngle
   ENDIF
       guidProfileBase.X = XguidProfileBase.x +Xabove

      ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosDat
         FDAT_ACT=Fbase
         BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD

        ;begin saw
   sawRun=255;
   wait sec 0
      guidProfileEnd.Z=(trimHightZ /INT_TO_REAL_DIVIDER)

      ;do the cut motion
      ;FOLD LIN guidProfileEnd  Vel=1.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileEnd, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=Fbase
         BAS(#CP_PARAMS,2.0)
         LIN guidProfileEnd
      ;ENDFOLD
      guidProfileEnd.Y =guidProfileEnd.y + 50

        ;FOLD LIN guidProfileEnd  Vel=1.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileEnd, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=Fbase
         BAS(#CP_PARAMS,2.0)
         LIN guidProfileEnd
      ;ENDFOLD

      ; move a bit in y direction to be shure that both sides of profile is cut
in the same hight.
        guidProfileEnd.Y =guidProfileEnd.y - 50

        ;FOLD LIN guidProfileEnd  Vel=1.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
```

```
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileEnd, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=Fbase
         BAS(#CP_PARAMS,2.0)
         LIN guidProfileEnd
      ;ENDFOLD
      ; move up from the cut position
        ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=cutDat
         FDAT_ACT=Fbase
        BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD
      ; cut finished

       sawRun = 0; Stop the saw
       cutDone=255
      ; finished with cuting.
       if vCutAngle >0 then
         guidProfileBase.z =XguidProfileBase.z +100
         guidProfileBase.y=XguidProfileBase.y +100
          ;FOLD LIN guidProfileBase  Vel=2.0 m/s CPDAT5 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:guidProfileBase, 3:, 5:2.0,
7:CPDAT5
         $BWDSTART=FALSE
         LDAT_ACT=setPosDat
         FDAT_ACT=FTCP
        BAS(#CP_PARAMS,2.0)
         LIN guidProfileBase
      ;ENDFOLD
       ENDIF

       ;FOLD PTP guidProfileBase  Vel=100 % PDAT2 Tool[6] Base[3];%{PE}%R
8.3.22,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:guidProfileBase, 3:, 5:100,
7:PDAT2
      $BWDSTART=FALSE
      PDAT_ACT=PPDAT2
      FDAT_ACT=Fbase
      BAS(#PTP_PARAMS,90)
      PTP XguidProfileBase
   ;ENDFOLD

   endIF

   sawRun = 0; Stop the saw
   cutDone=255
      ;begin saw
```

```
   wait sec 0
   RETURN
END
```

```
&ACCESS RVP
&REL 4
&PARAM DISKPATH = KRC:\R1\Program\IBR-SAW machine
DEFDAT  main PUBLIC
DECL POS XP1={X 0.0,Y 0.0,Z 50.0000,A 90,B 0,C 0,S 'B010',T 'B100011'}
DECL FDAT FP1={TOOL_NO 6,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL PDAT PPDAT1={VEL 100.000,ACC 100.000,APO_DIST 100.000,APO_MODE
#CDIS,GEAR_JERK 50.0000,EXAX_IGN 0}
ENDDAT
```

```
&ACCESS RVP
&REL 4
&PARAM DISKPATH = KRC:\R1\Program\IBR-SAW machine
DEF main( )

   ;DECL section ********************************
   int linevar
   ext BAS(BAS_COMMAND :IN, REAL :IN)
   ;initialization section **********************

 ;interrupt if the Robot is stopping to send signal to PLC, here is all safety
included
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
    INTERRUPT ON 3
   cutDone =0 ;set when the cut is finished, sent to PLC
   linevar =0 ; set the line the robot is producing on


   loop
  ; go to a home position.
;FOLD PTP P1 Vel=100 % PDAT1 Tool[3]:sag 4vpoint Base[3]:prodlinje 2;%{PE}%R
8.3.43,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P1, 3:, 5:100, 7:PDAT1
$BWDSTART=FALSE
PDAT_ACT=PPDAT1
FDAT_ACT=FP1
BAS(#PTP_PARAMS,90)
PTP XP1
;ENDFOLD
;if no command go home

     cutDone =0
     kukaReady=255
      ;check that both line and cuttype is recived
     if (linevar >0) and (cutType >0 ) THEN


          if linevar== 1 THEN ;cutting angle profile
               kukaReady=0
               wait SEC 0
               angleProfile()
               linevar=0;
         ENDIF

          if linevar== 2 THEN  ;cutting guiding profile
               kukaReady=0 ; set the command that the robot is in use
               wait SEC 0
               guidingProfile()
               linevar=0;
           ENDIF


      endif
```

```
      wait SEC 0.1 ; to make the variables be updated.
       linevar = line

   endloop

END
```

```
&ACCESS RV
&PARAM DISKPATH = SYSTEM
DEFDAT $CONFIG
;FOLD BASISTECH GLOBALS
;================================
; Default parameters for movement
; These values shouldn't be changed
;================================

;--------------------------------
; general MOVEMENT - parameters:
;--------------------------------

INT DEF_OV_PRO=100
INT DEF_ADVANCE=3

; PTP - MOVEMENTS
;--------------------------------
INT DEF_VEL_PTP=100; ptp velocity [%]
INT DEF_ACC_PTP=50; ptp acceleration [%]

; CP - MOVEMENTS
;--------------------------------
DECL CIRC_TYPE DEF_CIRC_TYP=#BASE
DECL JERK_STRUC DEF_JERK_STRUC={CP 500.0,ORI 50000.0,AX {A1 1000.0,A2 1000.0,A3
1000.0,A4 1000.0,A5 1000.0,A6 1000.0,E1 1000.0,E2 1000.0,E3 1000.0,E4 1000.0,E5
1000.0,E6 1000.0}}; jerk value for the spline. CP: [m/Sec3], ORI: [[GRAD/Sec3],
AX: [[GRAD/Sec3] (rotatory) resp. [m/Sec3] (linear)
REAL DEF_GEAR_JERK=100.0; gear jerk of axes [%]
REAL DEF_VEL_CP=2.0; path velocity [M/SEC]
REAL DEF_VEL_ORI1=200.0; swivel velocity [GRAD/Sec]
REAL DEF_VEL_ORI2=200.0; rotation velocity [GRAD/Sec]
REAL DEF_VEL_ORIS=200.0; spline swivel velocity [GRAD/Sec]
REAL DEF_ACC_CP=2.3; path acceleration [m/Sec2]
REAL DEF_ACC_ORI1=100.0; swivel acceleration [GRAD/Sec2]
REAL DEF_ACC_ORI2=100.0; rotation acceleration [GRAD/Sec2]
REAL DEF_ACC_ORIS=200.0; spline swivel acceleration [GRAD/Sec2]
REAL DEF_VEL_FACT=1.0
REAL DEF_ACC_SPTP=100.0; ptp spline acceleration [%]

; APO - parameters
;--------------------------------
INT DEF_APO_CPTP=50; PTP-Approximation [%]
INT DEF_APO_CVEL=100; Speed-Approximation [%]
REAL DEF_APO_CDIS=3.0; Distance-Approximation [mm]
REAL DEF_APO_CORI=5.0; Orientation-Approximation [Grad]
REAL DEF_APO_CORIS=80.0; spline Orientation-Approximation [Grad]

;================================
; Structures:
;================================
ENUM BAS_COMMAND
```

```
INITMOV,ACC_CP,ACC_PTP,VEL_CP,VEL_PTP,ACC_GLUE,TOOL,BASE,EX_BASE,PTP_DAT,CP_DAT,
OUT_SYNC,OUT_ASYNC,GROUP,FRAMES,PTP_PARAMS,CP_PARAMS
ENUM OUT_MODETYPE TRIGGER_,CONT_,STOP_
ENUM IPO_M_T NONE,TCP,BASE
ENUM APO_MODE_T CPTP,CDIS
STRUC DIG_OUT_TYPE INT FIRST_BIT, LENGTH, PARITY, CODING
STRUC CTRL_IN_T INT IN_NR, CHAR NAME_NAT[20]
STRUC CTRL_OUT_T INT OUT_NR, BOOL INI, CHAR NAME_NAT[20]
STRUC FCT_OUT_T INT NO, REAL PULS_TIME, BOOL STATE
STRUC FCT_IN_T INT NO, BOOL STATE
STRUC PDAT REAL VEL, ACC, APO_DIST, APO_MODE_T APO_MODE, REAL GEAR_JERK, INT
EXAX_IGN
STRUC LDAT REAL VEL, ACC, APO_DIST, APO_FAC, AXIS_VEL, AXIS_ACC, ORI_TYPE
ORI_TYP, CIRC_TYPE CIRC_TYP, REAL JERK_FAC, GEAR_JERK, INT EXAX_IGN, CIRC_MODE
CB
STRUC FDAT INT TOOL_NO, BASE_NO, IPO_MODE IPO_FRAME, CHAR POINT2[24], BOOL
TQ_STATE
STRUC ODAT INT OUT_NO, BOOL STATE, REAL PULSE_TIME, OUT_MODETYPE OUT_MODE, REAL
TIME_DELAY, OFFSET
STRUC BASIS_SUGG_T CHAR POINT1[24], POINT2[24], CP_PARAMS[24], PTP_PARAMS[24],
CONT[24], CP_VEL[24], PTP_VEL[24], SYNC_PARAMS[24], SPL_NAME[24], A_PARAMS[24]

STRUC OUT_SUGG_T CHAR PARAMS[24]
STRUC MACHINE_DEF_T CHAR NAME[24], INT COOP_KRC_INDEX, CHAR PARENT[24], FRAME
ROOT, ESYS MECH_TYPE, CHAR GEOMETRY[255]
STRUC MACHINE_TOOL_T INT MACH_DEF_INDEX, CHAR PARENT[24], CHAR GEOMETRY[255]
STRUC MACHINE_FRAME_T INT MACH_DEF_INDEX, CHAR PARENT[24], CHAR GEOMETRY[255]

STRUC TRIGGER_PARA INT TriggerPath, BOOL TriggerOnStart, INT TriggerDelay, CHAR
TriggerTask[40]
STRUC CONSTVEL_PARA INT ConstVelTyp, INT ConstVelPath, BOOL ConstVelOnStart
STRUC CONDSTOP_PARA INT CondStopPath, BOOL CondStopOnStart, CHAR
StopCondition[40]
STRUC ADAT TRIGGER_PARA TriggerPara, CONSTVEL_PARA ConstVelPara, CONDSTOP_PARA
CondStopPara

STRUC MODULEPARAM_T CHAR PARAMS[2000]

;==================================
; TORQUE MONITORING
;==================================
SIGNAL sTQM_SPSACTIVE $IN[1026]
SIGNAL sTQM_SPSSTATUS $OUT[1024]

BOOL bTQM_ACTIVE=FALSE
BOOL bTQM_CYC=FALSE
BOOL bTQM_KCPSTATUS=FALSE

STRUC TM_SUGG_T CHAR TM_ID[24]
STRUC TQM_TQDAT_T INT T11, T12, T13, T14, T15, T16, T21, T22, T23, T24, T25,
T26, K1, K2, K3, K4, K5, K6, O1, O2, ID, OVM, REAL TMF
```

```
DECL TQM_TQDAT_T TMDEFAULT={T11 200,T12 200,T13 200,T14 200,T15 200,T16 200,T21
1000,T22 1000,T23 1000,T24 1000,T25 1000,T26 1000,K1 280,K2 280,K3 280,K4 280,K5
280,K6 280,O1 20,O2 30,ID 0,OVM 0,TMF 1.0}
DECL TQM_TQDAT_T TM0={T11 200,T12 200,T13 200,T14 200,T15 200,T16 200,T21
1000,T22 1000,T23 1000,T24 1000,T25 1000,T26 1000,K1 280,K2 280,K3 280,K4 280,K5
280,K6 280,O1 20,O2 30,ID 0,OVM 0,TMF 1.0}

DECL TQM_TQDAT_T TQM_DATA_C[8]
DECL TQM_TQDAT_T TQM_TEMP

;================================
; External declarations:
;================================
EXT BAS (BAS_COMMAND :IN, REAL :IN )
EXT IR_STOPM ( )

;====================
; Signal declarations
; Do not change !!!!!
;====================
SIGNAL CHANNEL_1 $ANOUT[1]
SIGNAL CHANNEL_2 $ANOUT[2]
SIGNAL CHANNEL_3 $ANOUT[3]
SIGNAL CHANNEL_4 $ANOUT[4]
SIGNAL CHANNEL_5 $ANOUT[5]
SIGNAL CHANNEL_6 $ANOUT[6]
SIGNAL CHANNEL_7 $ANOUT[7]
SIGNAL CHANNEL_8 $ANOUT[8]
SIGNAL CHANNEL_9 $ANOUT[9]
SIGNAL CHANNEL_10 $ANOUT[10]
SIGNAL CHANNEL_11 $ANOUT[11]
SIGNAL CHANNEL_12 $ANOUT[12]
SIGNAL CHANNEL_13 $ANOUT[13]
SIGNAL CHANNEL_14 $ANOUT[14]
SIGNAL CHANNEL_15 $ANOUT[15]
SIGNAL CHANNEL_16 $ANOUT[16]
SIGNAL CHANNEL_17 $ANOUT[17]
SIGNAL CHANNEL_18 $ANOUT[18]
SIGNAL CHANNEL_19 $ANOUT[19]
SIGNAL CHANNEL_20 $ANOUT[20]
SIGNAL CHANNEL_21 $ANOUT[21]
SIGNAL CHANNEL_22 $ANOUT[22]
SIGNAL CHANNEL_23 $ANOUT[23]
SIGNAL CHANNEL_24 $ANOUT[24]
SIGNAL CHANNEL_25 $ANOUT[25]
SIGNAL CHANNEL_26 $ANOUT[26]
SIGNAL CHANNEL_27 $ANOUT[27]
SIGNAL CHANNEL_28 $ANOUT[28]
SIGNAL CHANNEL_29 $ANOUT[29]
SIGNAL CHANNEL_30 $ANOUT[30]
SIGNAL CHANNEL_31 $ANOUT[31]
SIGNAL CHANNEL_32 $ANOUT[32]
```

```
;==================================
; ID external user log-on:
;==================================
SIGNAL ExtUserID $IN[1026] TO $IN[1026]
SIGNAL ExtUserIDWatchDog $IN[1026]

;==================================
; Variables:
;==================================
DECL PDAT PDEFAULT={VEL 100.0,ACC 100.0,APO_DIST 100.0,APO_MODE #CDIS,GEAR_JERK
50.0,EXAX_IGN 0}
DECL LDAT LDEFAULT={VEL 2.0,ACC 100.0,APO_DIST 100.0,APO_FAC 50.0,AXIS_VEL
100.0,AXIS_ACC 100.0,ORI_TYP #VAR,CIRC_TYP #BASE,JERK_FAC 50.0,GEAR_JERK
50.0,EXAX_IGN 0}
DECL LDAT CDEFAULT={CB {AUX_PT {ORI #CONSIDER,E1 #CONSIDER,E2 #CONSIDER,E3
#CONSIDER,E4 #CONSIDER,E5 #CONSIDER,E6 #CONSIDER},TARGET_PT {ORI #INTERPOLATE,E1
#INTERPOLATE,E2 #INTERPOLATE,E3 #INTERPOLATE,E4 #INTERPOLATE,E5 #INTERPOLATE,E6
#INTERPOLATE}}}
DECL FDAT FDEFAULT={TOOL_NO 1,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE
FALSE}
DECL FDAT FHOME={TOOL_NO 1,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE
FALSE}
DECL ODAT ODEFAULT={OUT_NO 1,STATE TRUE,PULSE_TIME 0.0,OUT_MODE
#STOP_,TIME_DELAY 0.0,OFFSET 0.0}

DECL ADAT TDEFAULT={TriggerPara {TriggerPath 0,TriggerOnStart FALSE,TriggerDelay
0,TriggerTask[] " "}}
DECL ADAT CVDEFAULT={ConstVelPara {ConstVelTyp 0,ConstVelPath 0,ConstVelOnStart
FALSE}}
DECL ADAT CSDEFAULT={CondStopPara {CondStopPath 0,CondStopOnStart
FALSE,StopCondition[] " "}}

DECL PDAT PDAT_ACT
DECL LDAT LDAT_ACT
DECL FDAT FDAT_ACT
DECL ODAT ODAT_ACT
DECL INT ACT_FILTER
DECL INT ACT_DELAY
DECL INT ACT_DISTANCE

DECL CHAR SPL_NAME[24]

;GROUP-Definitions
;---------------------------------
INT COMPL_GROUP='B0001'

INT DEF_GROUP[10]
DEF_GROUP[1]='B1111'; complete
DEF_GROUP[2]='B0001'; robot only
DEF_GROUP[3]='B0011'; robot and track
DEF_GROUP[4]='B0111'; robot, track and ext. axis2
```

```
DEF_GROUP[5]='B1011'; robot, track and ext. axis3
DEF_GROUP[6]='B1111'; robot, track and ext. axis2+3
DEF_GROUP[7]='B1111'; complete
DEF_GROUP[8]='B1111'; complete
DEF_GROUP[9]='B1111'; complete
DEF_GROUP[10]='B1111'; complete

INT $ACT_GROUP='B0001'

; HOME POSITION
;---------------------------------
E6AXIS XHOME={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1 0.0,E2 0.0,E3
0.0,E4 0.0,E5 0.0,E6 0.0}
E6AXIS XHOME1={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1 0.0,E2 0.0,E3
0.0,E4 0.0,E5 0.0,E6 0.0}
E6AXIS XHOME2={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1 0.0,E2 0.0,E3
0.0,E4 0.0,E5 0.0,E6 0.0}
E6AXIS XHOME3={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1 0.0,E2 0.0,E3
0.0,E4 0.0,E5 0.0,E6 0.0}
E6AXIS XHOME4={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1 0.0,E2 0.0,E3
0.0,E4 0.0,E5 0.0,E6 0.0}
E6AXIS XHOME5={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1 0.0,E2 0.0,E3
0.0,E4 0.0,E5 0.0,E6 0.0}

; REFERENCE POINTS
;---------------------------------
DECL FRAME REF_PT[16]
REF_PT[1]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[2]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[3]={X 690.38031,Y -274.608917,Z 477.984131,A 0.0258575436,B -39.3580132,C
156.430511}
REF_PT[4]={X 731.233215,Y -35.4683571,Z 425.257599,A 172.735428,B 2.32659173,C
1.72396958}
REF_PT[5]={X 466.507385,Y 215.885834,Z 650.944092,A 161.994141,B 28.9299049,C
35.0779495}
REF_PT[6]={X 732.219666,Y -63.6428146,Z 427.069489,A 170.09549,B 24.188652,C
-56.3024216}
REF_PT[7]={X 662.128784,Y -111.026688,Z 450.223907,A 1.41791022,B -21.9487953,C
-124.292786}
REF_PT[8]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[9]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[10]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[11]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[12]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[13]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[14]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[15]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
REF_PT[16]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}

FRAME REF_TOOL={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}

; TOOL and BASE data
```

```
;---------------------------------
BOOL AUTO_IPO_M=FALSE
BOOL STOPM_FLAG=FALSE
BOOL TOOL_CORR_ON=FALSE
BOOL TOOL_CORR_W_ON=FALSE
BOOL BASE_CORR_ON=FALSE
BOOL M_BAS_COR_ON=FALSE
FRAME TOOL_CORR={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
FRAME TOOL_CORR_W={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
FRAME BASE_CORR={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
FRAME M_BASE_CORR={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
INT COR_TOOL_NO=0

INT MAX_TOOL=16
DECL FRAME TOOL_DATA[16]
TOOL_DATA[1]={X 70.0,Y 0.0,Z 190.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[2]={X 49.325,Y -45.614,Z 215.246,A 0.0,B 0.0,C 0.0}
TOOL_DATA[3]={X -43.5111,Y -49.051,Z 220.322205,A 0.0,B 0.0,C 0.0}
TOOL_DATA[4]={X -31.698,Y -54.559,Z 222.048,A -44.1882,B 2.9549,C 179.45}
TOOL_DATA[5]={X 32.31,Y -123.168,Z 114.619,A 135.804596,B -3.0492,C -179.443298}
TOOL_DATA[6]={X -30.047,Y -61.615,Z 218.434,A -44.3851,B 0.1806,C 178.971497}
TOOL_DATA[7]={X -120.117,Y 21.826,Z 123.041,A -44.3851,B 0.1806,C 178.971497};A
45.8049,B 0.922400,C -6.19970
TOOL_DATA[8]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[9]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[10]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[11]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[12]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[13]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[14]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[15]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
TOOL_DATA[16]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}

DECL CHAR TOOL_NAME[16,24]
TOOL_NAME[1,]="Saw"
TOOL_NAME[2,]="sag"
TOOL_NAME[3,]="sag 4vpoint"
TOOL_NAME[4,]="TestTool"
TOOL_NAME[5,]="BendTool2"
TOOL_NAME[6,]="actualTool"
TOOL_NAME[7,]="BendTool1"
TOOL_NAME[8,]=" "
TOOL_NAME[9,]=" "
TOOL_NAME[10,]=" "
TOOL_NAME[11,]=" "
TOOL_NAME[12,]=" "
TOOL_NAME[13,]=" "
TOOL_NAME[14,]=" "
TOOL_NAME[15,]=" "
TOOL_NAME[16,]=" "

DECL LOAD LOAD_DATA[16]
```

```
LOAD_DATA[1]={M 3.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[2]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[3]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[4]={M 6.0,CM {X 60.0,Y 0.0,Z 80.0,A 0.0,B 0.0,C 0.0},J {X 0.045,Y
0.045,Z 0.045}}
LOAD_DATA[5]={M 6.0,CM {X 60.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[6]={M 5.0,CM {X 60.0,Y 0.0,Z 80.0,A 0.0,B 0.0,C 0.0},J {X 0.045,Y
0.045,Z 0.045}}
LOAD_DATA[7]={M 5.0,CM {X 60.0,Y 0.0,Z 80.0,A 0.0,B 0.0,C 0.0},J {X 0.045,Y
0.045,Z 0.045}}
LOAD_DATA[8]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[9]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[10]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[11]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[12]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[13]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[14]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[15]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}
LOAD_DATA[16]={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X 0.0,Y 0.0,Z
0.0}}

DECL IPO_M_T TOOL_TYPE[16]
TOOL_TYPE[1]=#BASE
TOOL_TYPE[2]=#BASE
TOOL_TYPE[3]=#BASE
TOOL_TYPE[4]=#BASE
TOOL_TYPE[5]=#BASE
TOOL_TYPE[6]=#BASE
TOOL_TYPE[7]=#BASE
TOOL_TYPE[8]=#NONE
TOOL_TYPE[9]=#NONE
TOOL_TYPE[10]=#NONE
TOOL_TYPE[11]=#NONE
TOOL_TYPE[12]=#NONE
TOOL_TYPE[13]=#NONE
TOOL_TYPE[14]=#NONE
TOOL_TYPE[15]=#NONE
TOOL_TYPE[16]=#NONE

DECL LOAD LOAD_A1_DATA={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X
```

```
0.0,Y 0.0,Z 0.0}}
DECL LOAD LOAD_A2_DATA={M -1.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X
0.0,Y 0.0,Z 0.0}}
DECL LOAD LOAD_A3_DATA={M 0.0,CM {X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},J {X
0.0,Y 0.0,Z 0.0}}

INT MAX_BASE=32
DECL FRAME BASE_DATA[32]
BASE_DATA[1]={X 525.0,Y 0.0,Z 890.0,A 0.0,B 90.0,C 0.0}
BASE_DATA[2]={X 0.0,Y -525.0,Z 890.0,A 0.0,B 90.0,C 90.0}
BASE_DATA[3]={X 738.172302,Y -4.1488,Z 400.677307,A 89.7035,B 0.2399,C 0.0028}
BASE_DATA[4]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[5]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[6]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[7]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[8]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[9]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[10]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[11]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[12]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[13]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[14]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[15]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[16]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[17]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[18]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[19]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[20]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[21]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[22]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[23]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[24]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[25]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[26]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[27]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[28]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[29]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[30]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[31]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
BASE_DATA[32]={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}

DECL CHAR BASE_NAME[32,24]
BASE_NAME[1,]="AngleProfileBase"
BASE_NAME[2,]="GuideProfileBase"
BASE_NAME[3,]="prodlinje 2"
BASE_NAME[4,]=" "
BASE_NAME[5,]=" "
BASE_NAME[6,]=" "
BASE_NAME[7,]=" "
BASE_NAME[8,]=" "
BASE_NAME[9,]=" "
BASE_NAME[10,]=" "
```

```
BASE_NAME[11,]=" "
BASE_NAME[12,]=" "
BASE_NAME[13,]=" "
BASE_NAME[14,]=" "
BASE_NAME[15,]=" "
BASE_NAME[16,]=" "
BASE_NAME[17,]=" "
BASE_NAME[18,]=" "
BASE_NAME[19,]=" "
BASE_NAME[20,]=" "
BASE_NAME[21,]=" "
BASE_NAME[22,]=" "
BASE_NAME[23,]=" "
BASE_NAME[24,]=" "
BASE_NAME[25,]=" "
BASE_NAME[26,]=" "
BASE_NAME[27,]=" "
BASE_NAME[28,]=" "
BASE_NAME[29,]=" "
BASE_NAME[30,]=" "
BASE_NAME[31,]=" "
BASE_NAME[32,]=" "

DECL IPO_M_T BASE_TYPE[32]
BASE_TYPE[1]=#BASE
BASE_TYPE[2]=#BASE
BASE_TYPE[3]=#BASE
BASE_TYPE[4]=#NONE
BASE_TYPE[5]=#NONE
BASE_TYPE[6]=#NONE
BASE_TYPE[7]=#NONE
BASE_TYPE[8]=#NONE
BASE_TYPE[9]=#NONE
BASE_TYPE[10]=#NONE
BASE_TYPE[11]=#NONE
BASE_TYPE[12]=#NONE
BASE_TYPE[13]=#NONE
BASE_TYPE[14]=#NONE
BASE_TYPE[15]=#NONE
BASE_TYPE[16]=#NONE
BASE_TYPE[17]=#NONE
BASE_TYPE[18]=#NONE
BASE_TYPE[19]=#NONE
BASE_TYPE[20]=#NONE
BASE_TYPE[21]=#NONE
BASE_TYPE[22]=#NONE
BASE_TYPE[23]=#NONE
BASE_TYPE[24]=#NONE
BASE_TYPE[25]=#NONE
BASE_TYPE[26]=#NONE
BASE_TYPE[27]=#NONE
BASE_TYPE[28]=#NONE
```

```
BASE_TYPE[29]=#NONE
BASE_TYPE[30]=#NONE
BASE_TYPE[31]=#NONE
BASE_TYPE[32]=#NONE

;*****************************************
; Variables for CELL DEFINITION
;*****************************************

INT MAX_MACHINES=16
DECL MACHINE_DEF_T MACHINE_DEF[16]
MACHINE_DEF[1]={NAME[] "KR 6 R900 sixx",COOP_KRC_INDEX 1,PARENT[] "WORLD",ROOT
{X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #ROBOT,GEOMETRY[] "ObjectId =
-106603136"}
MACHINE_DEF[2]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[3]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[4]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[5]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[6]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[7]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[8]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[9]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[10]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[11]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[12]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[13]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[14]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[15]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}
MACHINE_DEF[16]={NAME[] " ",COOP_KRC_INDEX 0,PARENT[] " ",ROOT {X 0.0,Y 0.0,Z
0.0,A 0.0,B 0.0,C 0.0},MECH_TYPE #NONE,GEOMETRY[] " "}

DECL MACHINE_TOOL_T MACHINE_TOOL_DAT[16]
MACHINE_TOOL_DAT[1]={MACH_DEF_INDEX 1,PARENT[] "KR 6 R900 sixx",GEOMETRY[] " "}
MACHINE_TOOL_DAT[2]={MACH_DEF_INDEX 1,PARENT[] "KR 6 R900 sixx",GEOMETRY[] " "}
MACHINE_TOOL_DAT[3]={MACH_DEF_INDEX 1,PARENT[] "KR 6 R900 sixx",GEOMETRY[] " "}
MACHINE_TOOL_DAT[4]={MACH_DEF_INDEX 1,PARENT[] "KR 6 R900 sixx",GEOMETRY[] " "}
MACHINE_TOOL_DAT[5]={MACH_DEF_INDEX 1,PARENT[] "KR 6 R900 sixx",GEOMETRY[] " "}
MACHINE_TOOL_DAT[6]={MACH_DEF_INDEX 1,PARENT[] "KR 6 R900 sixx",GEOMETRY[] " "}
```

```
MACHINE_TOOL_DAT[7]={MACH_DEF_INDEX 1,PARENT[] "KR 6 R900 sixx",GEOMETRY[] " "}
MACHINE_TOOL_DAT[8]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_TOOL_DAT[9]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_TOOL_DAT[10]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_TOOL_DAT[11]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_TOOL_DAT[12]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_TOOL_DAT[13]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_TOOL_DAT[14]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_TOOL_DAT[15]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_TOOL_DAT[16]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}

DECL MACHINE_FRAME_T MACHINE_FRAME_DAT[32]
MACHINE_FRAME_DAT[1]={MACH_DEF_INDEX 0,PARENT[] "WORLD",GEOMETRY[] " "}
MACHINE_FRAME_DAT[2]={MACH_DEF_INDEX 0,PARENT[] "WORLD",GEOMETRY[] " "}
MACHINE_FRAME_DAT[3]={MACH_DEF_INDEX 0,PARENT[] "WORLD",GEOMETRY[] " "}
MACHINE_FRAME_DAT[4]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[5]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[6]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[7]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[8]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[9]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[10]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[11]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[12]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[13]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[14]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[15]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[16]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[17]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[18]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[19]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[20]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[21]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[22]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[23]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[24]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[25]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[26]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[27]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[28]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[29]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[30]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[31]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}
MACHINE_FRAME_DAT[32]={MACH_DEF_INDEX 0,PARENT[] " ",GEOMETRY[] " "}

;*****************************************
; Variables for axis CALIBRATING
;*****************************************
DECL INT CAL_AXIS
BOOL CONST_SPEED

;*****************************************
```

```
; Variables for added TechPackages
;****************************************
INT POWER

;****************************************
; General purpose return value of functions
; like INVERSE, FORWARD
;****************************************
INT ERR_STATUS=-1

;****************************************
; Variables for TOUCHUP user limit
;****************************************
BOOL UM_TOUCHUP=FALSE
REAL CARTESIAN_TOL=5.0
REAL ROTATION_ANGLE=10.0
REAL EXTAX_TOL[6]
EXTAX_TOL[1]=10.0
EXTAX_TOL[2]=10.0
EXTAX_TOL[3]=10.0
EXTAX_TOL[4]=10.0
EXTAX_TOL[5]=10.0
EXTAX_TOL[6]=10.0

;****************************************
; Variables for Displaying and Konfiguring
;****************************************
DECL INT I[20]

I[1]=0
I[2]=0
I[3]=0
I[4]=0
I[5]=0
I[6]=0
I[7]=0
I[8]=0
I[9]=0
I[10]=0
I[11]=0
I[12]=0
I[13]=0
I[14]=0
I[15]=0
I[16]=0
I[17]=0
I[18]=0
I[19]=0
I[20]=0

;****************************************
; all for InterBus Mapping on optional segments
```

```
;****************************************

DECL INT IBUS_SEGMENT[16]
IBUS_SEGMENT[1]=0
IBUS_SEGMENT[2]=0
IBUS_SEGMENT[3]=0
IBUS_SEGMENT[4]=0
IBUS_SEGMENT[5]=0
IBUS_SEGMENT[6]=0
IBUS_SEGMENT[7]=0
IBUS_SEGMENT[8]=0
IBUS_SEGMENT[9]=0
IBUS_SEGMENT[10]=0
IBUS_SEGMENT[11]=0
IBUS_SEGMENT[12]=0
IBUS_SEGMENT[13]=0
IBUS_SEGMENT[14]=0
IBUS_SEGMENT[15]=0
IBUS_SEGMENT[16]=0

;****************************************
; Reduction factors
;****************************************
DECL REAL RedVelExAx[6]
RedVelExAx[1]=100.0
RedVelExAx[2]=100.0
RedVelExAx[3]=100.0
RedVelExAx[4]=100.0
RedVelExAx[5]=100.0
RedVelExAx[6]=100.0
DECL REAL RedAccExAx[6]
RedAccExAx[1]=100.0
RedAccExAx[2]=100.0
RedAccExAx[3]=100.0
RedAccExAx[4]=100.0
RedAccExAx[5]=100.0
RedAccExAx[6]=100.0
;ENDFOLD (BASISTECH GLOBALS)
;FOLD AUTOEXT GLOBALS
;=================================
; Structures:
;=================================
ENUM FUNCT_TYPE PGNO_GET,PGNO_ACKN,PGNO_FAULT
ENUM P00_COMMAND INIT_EXT,EXT_PGNO,CHK_HOME,EXT_ERR

STRUC SPS_PROG_TYPE INT PROG_NR, CHAR PROG_NAME[12]
;=================================
; External declarations:
;=================================
EXT P00 (P00_COMMAND :IN, FUNCT_TYPE :IN, CHAR[] :OUT, INT :IN )
;External declaration  for  Submit controlled AutoExt
EXT P00_SUBM (P00_COMMAND :IN, FUNCT_TYPE :IN )
```

```
;==================================
; Variables:
;==================================
; Variables for internal
; Communication:
;----------------------------------
BOOL ERROR_FLAG
BOOL CHECK_HOME=TRUE
BOOL PROG_CONTROL=FALSE
DECL CHAR PRO_NAME1_L[8]
PRO_NAME1_L[]="        "
DECL CHAR PRO_NAME1_A[8]
PRO_NAME1_A[]="        "

INT PGNO=0;copy of ext. pgno
INT PGNO_ERROR=0;transmission error
INT PGNO_TYPE=1;coding type of ext. pgno
INT REFLECT_PROG_NR=0; enable mirroring of program number inputs (1=enabled,
0=disabled)

; Variables for External
; Communication: I/O-Interface
;----------------------------------
INT PGNO_FBIT=33;first bit of ext. pgno input $IN[]
INT PGNO_FBIT_REFL=999;first bit of ext. pgno reflection output $OUT[]
INT PGNO_LENGTH=8;length of ext. pgno (max. 16)
INT PGNO_PARITY=41;parity bit of ext. pgno
INT PGNO_REQ=33;request ext. pgno input
INT PGNO_VALID=42;validate ext. pgno input
INT APPL_RUN=34;application program is running output
INT ERR_TO_PLC=35;error output to PLC
INT P01_STEP
INT CHK_STEP
INT PGNO_FLAG

; Table for assign SPS program number to program name
INT MAX_SPS_PROG=12
DECL SPS_PROG_TYPE SPS_PROG[12]
SPS_PROG[1]={PROG_NR 1,PROG_NAME[] "HP01()    "}
SPS_PROG[2]={PROG_NR 2,PROG_NAME[] "HP02()    "}
SPS_PROG[3]={PROG_NR 3,PROG_NAME[] "HP03()    "}
SPS_PROG[4]={PROG_NR 4,PROG_NAME[] "HP04()    "}
SPS_PROG[5]={PROG_NR 5,PROG_NAME[] "HP05()    "}
SPS_PROG[6]={PROG_NR 6,PROG_NAME[] "HP06()    "}
SPS_PROG[7]={PROG_NR 7,PROG_NAME[] "HP07()    "}
SPS_PROG[8]={PROG_NR 8,PROG_NAME[] "HP08()    "}
SPS_PROG[9]={PROG_NR 9,PROG_NAME[] "HP09()    "}
SPS_PROG[10]={PROG_NR 10,PROG_NAME[] "HP10()    "}
SPS_PROG[11]={PROG_NR 62,PROG_NAME[] "HP62()    "}
SPS_PROG[12]={PROG_NR 63,PROG_NAME[] "HP63()    "}
DECL CHAR TMPNAME[128]
```

```
TMPNAME[]="                                    "
;ENDFOLD (AUTOEXT GLOBALS)
;FOLD BackupManagerConfig
BOOL BM_ENABLED
INT BM_OUTPUTVALUE
SIGNAL BM_OutputSignal $OUT[4095] TO $OUT[4096]
SIGNAL BM_InputSignal $IN[1026] TO $IN[1026]
;ENDFOLD BackupManagerConfig
;FOLD Conveyor
DECL INT CONV_PART_NBR[16]
CONV_PART_NBR[1]=0
CONV_PART_NBR[2]=0
CONV_PART_NBR[3]=0
CONV_PART_NBR[4]=0
CONV_PART_NBR[5]=0
CONV_PART_NBR[6]=0
CONV_PART_NBR[7]=0
CONV_PART_NBR[8]=0
CONV_PART_NBR[9]=0
CONV_PART_NBR[10]=0
CONV_PART_NBR[11]=0
CONV_PART_NBR[12]=0
CONV_PART_NBR[13]=0
CONV_PART_NBR[14]=0
CONV_PART_NBR[15]=0
CONV_PART_NBR[16]=0
;ENDFOLD Conveyor
;FOLD USER GLOBALS
;*****************************************
;Make your modifications -ONLY- here
;*****************************************
;=================================
; Userdefined Types
;=================================

;=================================
; Userdefined Externals
;=================================
; inputs
SIGNAL line $IN[1] TO $IN[8]
SIGNAL cutType $IN[9] TO $IN[16]
SIGNAL hAngle $IN[17] TO $IN[48]
SIGNAL vAngle $IN[49] TO $IN[80]
SIGNAL trimHightZ $IN[81] TO $IN[112]; hight from bottom of profile to and Zpos
for cut
SIGNAL StopRobot $IN[113] TO $IN[120]

; outputs
SIGNAL cutDone $OUT[363] TO $OUT[370]
SIGNAL sawRun $OUT[371] TO $OUT[378]
SIGNAL ERROR $OUT[379] TO $OUT[386]
SIGNAL kukaReady $OUT[387] TO $OUT[394]
```

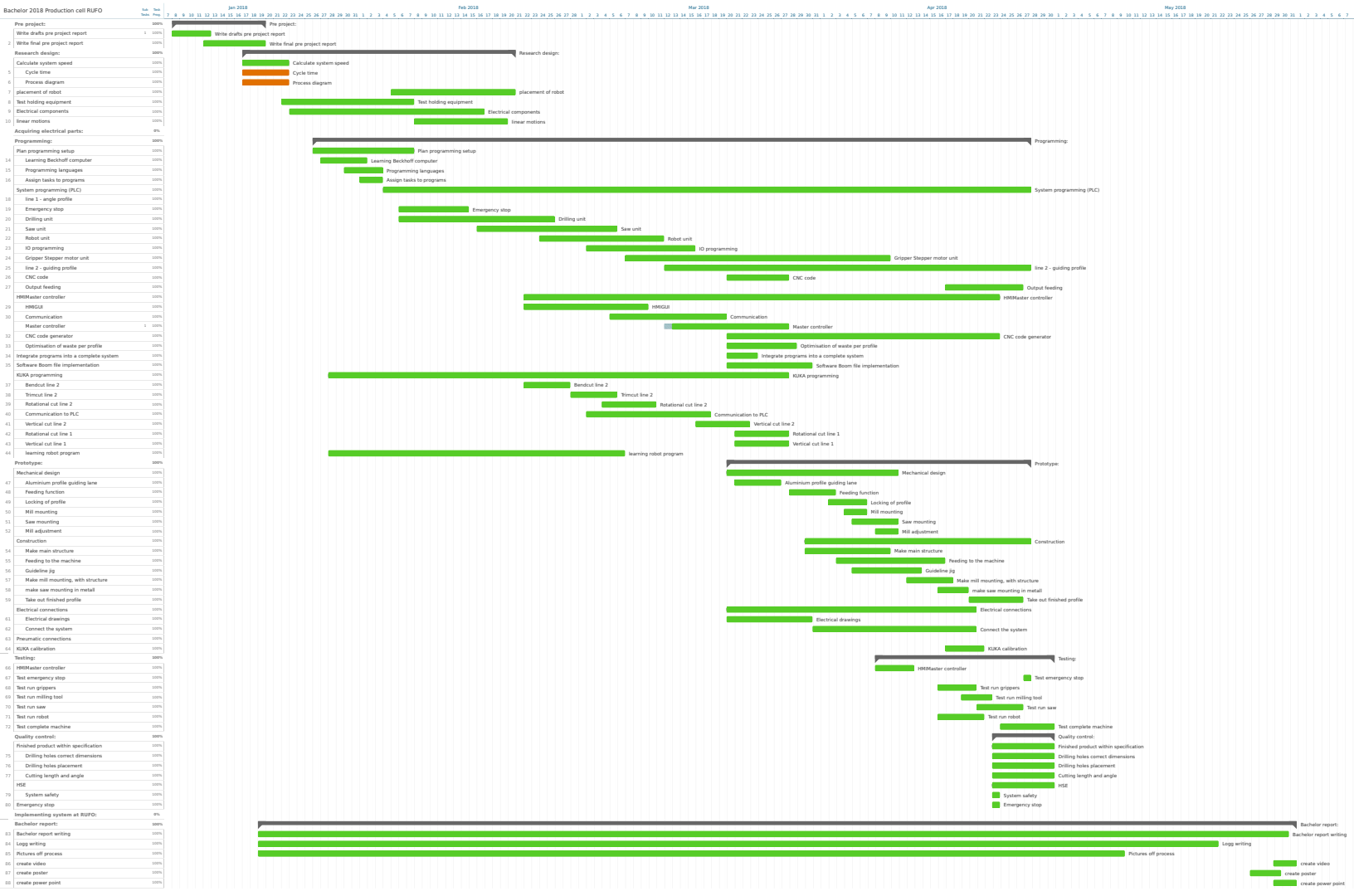
```
;==================================
; Userdefined Variables
;==================================
CONST INT ANGLE_OFFSET=90; the offsett of the value sent in from the plc.
CONST INT INT_TO_REAL_DIVIDER=10; the factor the angel is multiplied with before
PLS is sending
CONST REAL SAW_BLADE_THICKNES=2.8; the thicknes of the saw blade.
;ENDFOLD (USER GLOBALS)
ENDDAT
```

```
&ACCESS R
&COMMENT HandlerOnRobotFault
DEF  IR_STOPM ( )
;-----------------------------------
; Error Handling Robot Controller
; Switch OFF and Switch ON processes
; KRC Version >= V5.5
;-----------------------------------
  ;FOLD DECLARATIONS
  ;FOLD USER DECL
  ; Please insert user defined declarations

  ;ENDFOLD (USER DECL)
  ;FOLD BASISTECH DECL
    BOOL ApplicationRunFlag
    DECL CHAR ID[3]
  ;ENDFOLD (BASISTECH DECL)
 ;ENDFOLD (DECLARATIONS)
  ;FOLD BASISTECH INIT
     INTERRUPT OFF 3
     STOPM_FLAG=TRUE ;Reflects state of interrupt 3 to activate/deactivate
$Stopmess interrupt
     ID[]="CTL"
     If ($STOPMESS==TRUE) THEN
       BRAKE
  ;ENDFOLD (BASISTECH INIT)

  ;FOLD USER STOP
;Make your modifications here
sawRun =0 ; stop the saw
ERROR =255 ;error signal to plc
  ;ENDFOLD (USER STOP)
  ;FOLD BASISTECH STOP
     P00 (#EXT_ERR,#PGNO_GET,ID[],128 )
     ApplicationRunFlag=FALSE
     IF (Appl_Run>0) THEN
       IF $OUT[Appl_Run] THEN
         ApplicationRunFlag=TRUE
         $OUT[Appl_Run]=FALSE
       ENDIF
     ENDIF
     REPEAT
       POWER=SYNC()
       HALT
     UNTIL (($STOPMESS==FALSE) AND ($POWER_FAIL==FALSE))
  ;ENDFOLD (BASISTECH STOP)

  ;FOLD BASISTECH RESTART
     P00 (#EXT_ERR,#PGNO_GET,ID[],0 )
     IF (ApplicationRunFlag==TRUE) THEN
       IF (Appl_Run>0) THEN
         $OUT[Appl_Run]=TRUE
```

```
      ENDIF
    ENDIF
  ;ENDFOLD (BASISTECH RESTART)
  ;FOLD USER RESTART
;Make your modifications here
ERROR =0
  ;ENDFOLD (USER RESTART)

  ;FOLD BASISTECH REACTIVATE
    Endif
        INTERRUPT ON 3
    STOPM_FLAG=FALSE ;Reflects state of interrupt 3 to activate/deactivate
$Stopmess interrupt
  ;ENDFOLD (BASISTECH REACTIVATE)
END
```

Appendix L

Gant diagram


Bachelor 2018 Production cell RUFO
Jan 2018
Feb 2018
Mar 2018
Apr 2018
May 2018
Pre project:
Write drafts pre project report
Write final pre project report
Research design:
Calculate system speed
Cycle time
Process diagram
placement of robot
Test holding equipment
Electrical components
linear motions
Programming:
Plan programming setup
Learning Beckhoff computer
Programming languages
Assign tasks to programs
System programming (PLC)
Emergency stop
Drilling unit
Saw unit
Robot unit
IO programming
Gripper Stepper motor unit
line 2 - guiding profile
CNC code
Output feeding
HMIMaster controller
HMIGUI
Communication
Master controller
CNC code generator
Optimisation of waste per profile
Integrate programs into a complete system
Software Boom file implementation
KUKA programming
Bendcut line 2
Trimcut line 2
Rotational cut line 2
Communication to PLC
Vertical cut line 2
Rotational cut line 1
Vertical cut line 1
learning robot program
Prototype:
Mechanical design
Aluminium profile guiding lane
Feeding function
Locking of profile
Mill mounting
Saw mounting
Mill adjustment
Construction
Make main structure
Feeding to the machine
Guideline jig
Make mill mounting, with structure
make saw mounting in metall
Take out finished profile
Electrical connections
Electrical drawings
Connect the system
KUKA calibration
Testing:
HMIMaster controller
Test emergency stop
Test run grippers
Test run milling tool
Test run saw
Test run robot
Test complete machine
Quality control:
Finished product within specification
Drilling holes correct dimensions
Drilling holes placement
Cutting length and angle
HSE
System safety
Emergency stop
Bachelor report:
Bachelor report writing
Logg writing
Pictures off process
create video
create poster
create power point