TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF ENGINEERING
Department of Electrical Power engineering and Mechatronics

Migration and Integration of FESTO Tripod robot with Beckhoff's CX2042 controller.

FESTO Tripod roboti ümberseadistamine ja integreerimine Beckhoff CX2042 kontrolleriga.

Master Thesis

Student : Erick Suncin

Student Code: 194294 MAHM

Supervisor: Mart Tamre

Tallinn 2020

(*On the reverse side of title page*)

AUTHOR’S DECLARATION

Hereby I declare, that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

“.......” .... 20…..

Author: ....
/signature /

Thesis is in accordance with terms and requirements

“.......” .... 20….

Supervisor: ….......................
/signature/

Accepted for defence

“.......”....................20… .

Chairman of theses defence commission: ....
/name and signature/

Non-exclusive Licence for Publication and Reproduction of GraduationTthesis[1]

I, Erick Suncin (date of birth: 04/14/1989) hereby

1.grant Tallinn University of Technology (TalTech) a non-exclusive license for my thesis: Migration and Integration of FESTO Tripod robot with Beckhoff's CX2042 controller.

supervised by Mart Tamre

1.1reproduced for the purposes of preservation and electronic publication, incl. to be entered in the digital collection of TalTech library until expiry of the term of copyright;

1.2published via the web of TalTech, incl. to be entered in the digital collection of TalTech library until expiry of the term of copyright.

1.3 I am aware that the author also retains the rights specified in clause 1 of this license.

2.I confirm that granting the non-exclusive license does not infringe third persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

[1]Non-exclusive Licence for Publication and Reproduction of Graduation Thesis is not valid during the validity period of restriction on access, except the university's right to reproduce the thesis only for preservation purposes.

_______________ (signature)

_______________ (date)

Department of Electrical Power Engineering and Mechatronics

Student : Erick Rene Suncin Aguilar. 194294MAHM

Study Programme

Main Specialty: MAHM MsC Mechatronics

Supervisor(s): Professor Mart Tamre, Director of MsC Master

Thesis topic:

(In English). Migration and Integration of FESTO Tripod robot with Beckhoff's CX2042 controller.

(In Estonian) FESTO Tripod roboti ümberseadistamine ja integreerimine Beckhoff CX2042 kontrolleriga.

1. Introduction

Nowadays it is pretty common to find obsolete machinery in the industry, meaning that since they were built the same controller or control system has been working in the machine and new applications are not possible, that is why migrating and integrating existing machines to new controller is a must in today's automation applications.

With this Thesis, the goal is to be able to migrate and integrate all the applications of FESTO tripod Delta robot with the new Beckhoff Controller, the CX2042. It is important to note that the FESTO tripod robot is being controlled by its default controller from FESTO (CMXR-C1). By integrating the robot with the new controller, new applications can be done, and also there will be a significant increase in the performance of the robot.

The motivation behind this research and project comes from my background in automation. I believe that migration machines that have a default controller to a new and more capable controller can bring a lot of benefits on the functionality of the machine, also the main goal og migrating a machine to a new controller is to increase performance and have many more applications. Also it is important to note that this project will benefit the department of power electronics and mechatronics in many ways.

Completing this thesis and project will assure the usage of Festo Tripod form many years to come and with increased applications fro this machine, that will also benefit other students in the area. By completing the project we will also be making a better machine and for sure we will have applications that can be tried in such development.

2. Background

The Festo Tripod Delta robot works with its default controller, the Festo PLC CMXR-C1. This controller is a multi-axis controller specially made for kinematics applications and as you may have to guess the FESTO tripod has a lot of kinematics behind its functionality, that is why this controller is the heart of the complete kinematic system. All of this sounds good for the FESTO tripod Robot, so why change it? The answer is simple: The Beckhoff Controller CX2042 is much more powerful than what the FESTO controller offers. This is why the first step is making work the FESTO tripod with the CX2042 and it's current applications. After this is done then we can go further in the applications that are desired for example: integrating MatLab and vision modules to the robot, as well as creating remote access to it. Knowing how to Integrate all this is the main barrier that we will have to pass, integration between two different systems sometimes can be sketchy.

For this project and thesis it will be necessary to read and understand all the corresponding manuals such as the motor driver manuals, the installation manuals, the CX2042 manual and some other articles related to Beckhoff controllers. Also there are a few articles and thesis previously done that had a related topic just like this one. For further applications some IEEE articles will be used.

3. Methodology

Since this project is more of seeing results and less of doing a research the first step on understanding the problematic is understanding how the controller and the robot, drivers etc…can communicate between each other, it is really important to understand every component in the robot. Once everything is understood we can proceed with getting results with the migration of the whole system. Also it is very important to understand how the FESTO tripod works and its current applications so when we replicate everything with the new controller we can understand what the ultimate goal is. For the research part of this project it will be really interesting to have some applications with MatLab and Beckhoff's vision modules.

The results will be measured once the migration, integration of the Festo Tripod robot is made with the new CX2042 controller. If there are good results then the whole system will work perfectly and everything has to be documented, that way we can surely state that a universal approach of integration a machine can be done with the new Beckhoff controller. An excellent result will also let us have new applications added to the Tripod System.

4. Research Schedule

No.	Task Description	Deadline
1.	Understanding of the whole Festo Tripod system Full knowledge of all the components Understanding of the new CX2042 controller	15.11.2020
2.	Integration and Migration of the Tripod system To CX2042 controller Rewiring of the whole cabinet to the new Beckhoff controller	30.01.2021
3.	Documentation and diagrams of the new system Research on new applications and integration to Matlab	01.03.2021
4.	Creation of functional HMI screens	30.04.2021
5.	Final Presentation of the whole system	20.05.2021

Proceeding steps outline

Thinking about what interest you ‹ select a thesis topic proposal ‹ conduct your research ‹ choose your supervisor ‹ completing literature review ‹ planning outlines ‹ conducting your research ‹ writing your thesis ‹ defense presentation ‹ composing journal/conference papers.

Language: English Deadline for submission of Thesis:

Student: Erick Suncin. .... October 12, 2020

(Signature)

Supervisor: Mart Tamre .... October 12, 2020

(Signature)

Head of study programme: Mart Tamre

Company based confidentiality and other terms to formulate on the reverse side.

CONTENTS

PREFACE

This thesis project was proposed by the Professor Mart Tamre, director of Msc in Mechatronics at Taltech university and the author accepted with great enthusiasm the challenge of integration the Beckhoff controller CX2042 to an existing machine and be able to give the department a new possibility in developing tools for upcoming students. The work of this thesis project was conducted in the Mechatronics laboratory where the EXPT tripod robot is located, at Tallinn Technical University. All the components used for this thesis work are property of Taltech university and will open doors to keep using the EXPT robot.

The thesis project is mainly focused on integrating the FESTO EXPT tripod robot to a controller that is capable of having an immense amount of solutions and applications in a single software, that is user friendly since it is based on Microsoft Visual studio and does not require to buy licenses when developing new applications, there are some limits with trial licenses but they are mostly good enough for university researches. Previous researches had been done trying to integrate this robot without success, giving the author extra motivation to accomplish the objective.

The author would like to express gratitude to Professor Mart Tamre for the opportunity of taking this project and helping with the development of the Thesis work. Also the author is thankful with all the Mechatronics department staff that helped with the tools and permissions. Also the deepest gratitude to Bekchoff Estonia and Festo Estonia for giving tips and helping with the development of the Thesis work.

Keywords: Programmable logic controller (PLC), fieldbus, programming, delta robot, Master’s thesis

1. INTRODUCTION

The following thesis project has the main objective of creating and implementing a universal solution to migrate any robot or application to a new a controller, the CX2042 from Beckhoff Automation, and be able to integrate the existing applications and have new applications available as well. This integration is needed because we really want to be able to have new applications with the existing robots and the CX2042 controller, a PC-based controller, is much more powerful than the one the exisitng robots have. For this thesis project we will be focusing the migration and integration of the new CX2042 Beckhoff controller to the existing FESTO Expt Tripod robot, or better called delta robot from FESTO. The FESTO EXPT is a robot fabricated by FESTO that work on the basis of parallel kinematics, it has 3 main servo motors that are controlled by three basic axis controllers and 1 extra servo motor that is also controlled by its own motor drive controller. The tripod robot works with a CMXR-C1 PLC that is also fabricated by FESTO and is a multi axis controller [1] that makes it really easy to program algorithms for picking and placing with a relatively high speed. This controller suits perfectly the main idea of the FESTO tripod robot, that is to have simple picking and placing application and in some industries really small applications.

But as you may have wonder, robotics and automation are developing really fast and new trends and applications are appearing and industry has to keep up with these trends and new applications in order to avoid being an obsolete industry.
For this new controllers, new sensors, new technologies need to be applied to already existing machines to have new opportunities. The main objective of this thesis will be focused in migrating the applications to the new CX2042.

There will be some challenges while integrating this technology since previously there have been researches and projects trying to communicate the actual components to the CX2042 with no success. One of the main reasons there was no success while trying to integrate the new controller was that the communication protocol that the motor drives work on are CANopen, a communication protocol that is based on CAN, and there has to be some configurations and parameters that have to be analysed.

The main idea of migrating this delta robot to a new controller is that the CX2042 is a PC-based controller and Beckhoff has been one of the pioneers of this technology since the first PC-based PLC was fabricated by them in 1986, and as we will see during the first part of this thesis the new industrial revolution (Industry 4.0) has many opportunities in this type of controller, since IoT devices need the integration of IT communication and automation, then keeping with the trend and new technologies will definitely need the use of a new controller that is capable of having new applications and new technologies that are needed in industry and in an university environment as well.

The CX2042 controller is a controller that is part of the CX2000 series and has an Intel® Xeon® CPU with a clock rate of 2.2 GHz (4 cores), if we compare this to the CMXR-C1 there is a huge difference in power between both of them. It also contains a main memory with a size of 8 GB RAM, with expandable option available.The CPU has an internal 128 kB NOVRAM, which acts as a persistent data memory if no UPS is used and it has an operating system (Microsoft Windows 10 IoT Enterprise 2016 or Windows 10 IoT Enterprise 2019 LTSC 64 bit) .

Problem Statement

In this master thesis the main objective will be to achieve a universal solution for the integration on the new controller CX2042 into any machine available, in this case we will be using the Delta Robot as our base. This is necessary in order to be able to migrate any application into a new controller and that way, solve a big issue that many companies have: Having machine with old controllers and not having a solution to migrate and integrate new applications into their machine. In the automation field there is an increased need to keep innovating and applying new processes that improve the manufacturing or industrial capacity, this will help any industry or user to have a better result and be able to keep with the trend of industry 4.0

2. LITERATURE REVIEW

2.1. Automation history and trends

Manufacturing has been one of the main pillars for every country’s economy, especially the use of industrial robots for manufacturing, and will continue to be for many more years to come. If we check on the data of the most powerful countries, China, Germany, and the United States, we clearly see that the contribution of the manufacturing field to the economic field is 27% for China, 23% for Germany, and 12% for the United States [2], but there is a clear decrease in this contribution throughout history.

There have been some major industrial revolutions that have shaped the form and the way industry and manufacturing works. The first major change in the industry happened around the end of the 18th Century and was characterised by converting agricultural and rural areas to urban areas where mass manufacturing of products was made, mainly powered by steam. Afterward the second industrial revolution came between the late 19th century and early 20th century, it was characterised by standardisation and industrialisation of manufacturing technologies at a rapid pace and also more manufacturing inventions were well established in the industrial area. The third industrial revolution happened in the second half of the 20th century and here is where the rise of telecommunications, electronics, and computers, in this time frame two major inventions (that concern this thesis) were brought: PLCs and robots.

And last but not least is where we are standing right now, Industry 4.0 or like some people call this the fourth industrial revolution. Some people do agree that we are experiencing the fourth industrial revolution but some others still disagree, what it is a fact is that we are experiencing new inventions and new technologies and the magnitude is yet unknown.

Industry 4.0 is characterised by the extensive use of the internet [3] and with this a big amount of data is gathered, making it possible for machines and robots to predict certain situations and have self-learning algorithm. Also, a lot of technologies have emerged during this so called “Fourth Industrial revolution” like image processing applications to control robots, artificial intelligence, remote control, Internet of things, 3D printing, autonomous vehicles, etc…

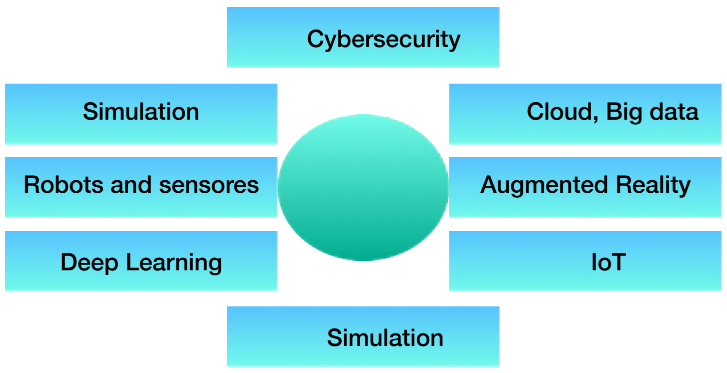


Figure 2.1.1. Industry 4.0 Technologies [5]

This industrial revolution is believed by some optimistic people that this revolution will be the one that will have the most impact on the economy of the world and also improve the quality of the population [4], this means that there is a trend towards these technologies and this means that new machines will come into play in the industry, but what happens with industrial and manufacturing companies that have invested a lot of money in robots and machines and cannot afford a totally new machine or robot? Well this is where integration and migration of new controllers to adapt new technologies to an existing machine enters into scene.

This is something that has taken a lot of importance lately, it is a new trend that consists of transitioning and integrating a new controller to a new robot to make this robot capable of having new technologies, that of course are part of the new industry 4.0 and even though they might need new components like new sensors, vision modules, cameras, more complex algorithms at the end this is much more affordable than a new machine with all this capability.

The new industrial revolution is changing how the industry works and how business operates, the way industry 4.0 is making a fusion between the industrial processes and the daily tasks allows a lot of benefits to society, like: reducing costs, better quality life, quality of processes, improved operations and many more [6]. This is why industry 4.0 can make a huge change on how things are managed in an industrial environment, but even more incredible is that this industrial revolution that we are now living will shape society in a huge way, it will change how business operates, and there will be an increase in incomes benefiting from the industrial change that has to be taken into account if an industry wants to continue a big progress and have a bright future, society 4.0 is being created in a really amazing way.

In the next section we will analize how industry 4.0 and internet of things is shaping industry and how IIoT (Industrial internet of things) is shaping the manufacturing and industry processes.

2.2. Industrial internet of things (IIoT)

It is important to draw a small line between what is internet of things and what is industrial internet of things. To put it in a simple term: IIOT (Industrial Internet of things) is the when IT is integrated to industrial automation devices and control systems such as sensors, technologies, big data, self-learning algorithms, and much more [8]. It is important to note as well that certain parameters have to be fulfilled in order to consider a system part of the current trends in IIOT.

IoT is the technology that connects devices (wired or wireless) to a network, in the other hand IIoT is the kind of technology which deals all the industrial machinery that have intelligent sensors, software and many other components of the current industry 4.0, it also involves the connection of this complex machinery to the network in order to control it through a different service, but it not only involves this connection between machines and sensors it also involves the human being , that is the one in charge of interfacing the unit to have a machine as close as a error-free system, humans will always be involved [9].

The adoption of IIoT can change how industries work, but there is the challenge to have strategies to strengthen efforts to digital transformation while maintaining security in all aspects. There are three areas that need to be focused on with IIoT: [10]

- Availability: The state of machines or processes to be "available" or unoccupied in order to have the maximum use of certain machines or tools.
- Scalability : The characteristic of a company or industrial process to cope and perform really well with increased amount of workload and also have the option to add new components without a need of shutting everything down.
- Security: Since IoT devices are connected to the world wide web then there is mayor concern in cybersecurity, the mayor challenges of this area to keep everything good are:

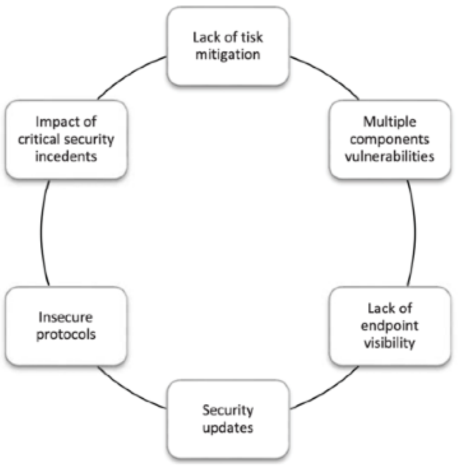

Lack of tisk mitigation
Multiple components vulnerabilities
Lack of endpoint visibility
Security updates
Insecure protocols
Impact of critical security incedents

Figure 2.1.2. Cybersecurity challenges for IIoT [10]

As we reviewed this section we were able to see how there is a small line between IoT devices and IIoT trend in machines. There is a huge amount of information regarding IIoT and industry 4.0. It is very important to keep an eye on this trends and technologies in order to see if it is viable to apply any new technology or application to an industrial process or a machine like the FESTO tripod robot.

2.3. Industrial communication trends

With the introduction of all the new trends and technologies in communication the industrial scenario is having a tremendous change and also having an immense amount of opportunities to shock the world with new processes and applications.
As stated, the main goal of automated systems is the harmonisation of devices beyond networking (just connections) and this is because the core of every distributed system is the smooth exchange of information between devices[11] that will or are connected to the same network.

This smooth exchange of information is possible thanks to the development of communication protocol, creating an opportunity to exchange the information more comprehensive and therefore automation systems began to grow into more complex systems, but this has not always being so easy to state or say since it has been lately where these changes and opportunities have been penetrating the industrial picture [12].

Since the industrial automation world is so heterogenous and the need to communicate all the different devices and the huge amount of variety of devices in mechatronic systems, so this takes the development to a main issue that has to be addressed: "The real time industrial communication" [13]

2.4. Beckhoff Automation

Beckhoff Automation was established around 1980 and now it has more tan 30 years on innovating and giving a large contribution to PC-based automation. It is important to mention that the first PC-based machine controller was created by Beckhoff in 1986 and as we may wonder industry 4.0 needs IT and pc based system to make an all together system, that is why Beckhoff controllers are one of many in the market that will be really useful for future applications and technologies. Beckhoff milestone is as follows:

- 1986 - First Beckhoff PC-XT-based controllers for woodworking machines. This units were first used as operating, computing and memory units, using motorola hardware, but they rapidly realized that the PC- based controller could take over all the system functions and the idea for future inventions was borned.
- 1989 - Lightbus interface card, first optical fieldbus created by Beckhoff, the PC proccersor always assumed the master controller role
- 1990- the all in one PC motherboard created with Intel, now able to communicate with siemens S5, this was known as the "press PC" since the main role and programation was done for metal pressing.
- 1992- communication with mitshubishi PLC
- 2002- Development of motherboard CX100 model, with Pentium processor, this was the return for beckhoff for creating its own motherboards and the first PC embeddded PC with the DIN standard mounting.
- 2003- Ethercat is created, the first real-time field bus and patent for Beckhoff.
- 2004- Intel pentium M processors for controllers
- 2006- Intel core duo processor for controllers
- CX2042-Inttel Xeon CPU with a clock rate of 2.2 GHz, 4 cores in main memory with a size of 8 GB RAM with Microsoft Windows 10 IoT Enterprise as operating system.
- 2010- TwinCat 3 is developed, this is the software to integrate everything in a single automation software.
- 2012- second generation of HMI panels was developed

- 2014- a new axo servo system is developed the AX8000
- 2015- New Ethercat IOT devices are created as well as TwinCat HMIs
- 2017- Ulta compact controllers are created as IPCs
- 2019- Multicore industrial PC with a high grade of protections (IP65/67) are launched into market, and new options for machine learning in Twincat 3 [14]

2.5. CANopen communication protocol

One of the main problems from previous thesis projects was the communication between the motor drives and the CX2042 Beckoff PLC. This happened because the motor drives are made by FESTO and they use the communication protocol called CANpen and they are communicated by the basic axis drive controller from FESTO, as today the interface they are using is the one in which it was commissioned. The next figure shows the control architecture from FESTO EXPT tripod robot.

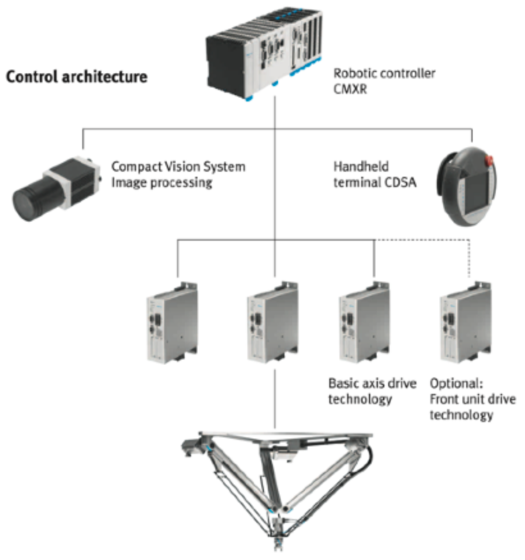

Control architecture
Robotic controller CMXR
Compact Vision System Image processing
Handheld terminal CDSA
Basic axis drive technology
Optional: Front unit drive technology

Figure 2.1.1. Control Arquitecture for FESTO Expt Delta Robot [15]

According to the manual the robot is using 3 motor controllers CMMP-AS-C5-3A for 3 main axes, 1 motor controller CMMP-AS-C2-3A for 1 supplemental axis. These are very old motor drive controllers and apparently they only have the CANopen interface [16].

In order to make a fully functional CANopen network first we need to understand how this works. The CANopen fieldbus is based on the CAN communication protocol but working on higher layer of the OSI network model [17] [18], the OSI is a conceptual model standarizing communication functions across diverse communication devices. Lower level is the basic communication like physical cables, and the upper ones are more complex communication. CANopen was first created for motion-oriented machine controlled applications and now it is extensively used in control of stepper motors and servo motors, as well as some robotic application and industrial machinery applications. There are some important concepts that we need to have into account when configuring a CANopen network, these concepts are:

- Communication models: There are several options to choose from but in for this thesis purpose the master/slave model will be used. Meaning that a master node will be created and several slaves will depend on the master network. [17]
- Communication protocols: as we have explained protocols are used for communicating object in a network
- Device states: a concept used in many other standards and communication protocols that gives information about a devices, e.g. running, emergency state, stop, etc…
- Object dictionary or OD, each device in the network has its own OD and in these parameters for this device are found. These are normally given by the manufacturer.
- Electronic Data sheet (EDS), Standard file format for OD entries [17]
- Device profiles, standards used for running a communication protocol.

All these concepts have to be taken into account in order to make a susccesful communication network. Also to understand a bit better the CANopen the following figure shows the frameworks that are normally used in the CANopen protocol.

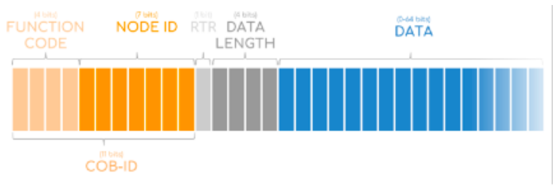


Figure 2.1.2. CANopen frame work [19]

2.6. Integration methods and standards

In the field of industrial automation and robotics there are several standards that need to be followed in order to implement a competitive project and to be in accordance to the industry standards.

The most important standard in PLC programming and implementation us the IEC 61131 and is the "standard for programmable controllers" it is divided in ten parts, each part outlining an important part of PLC programming and all the parts are maintained to keep the industry trend. The first part of this standard is an introductory part mentioning the terms that will be used in subsequent parts. The second part of the standard is for equipment requirements and tests, it describes the normal service conditions requirements, functional requirements, functional type tests and verifications, and electromagnetic compatibility. The third part of this standard (the most widely used and known) is for programming languages, describing each of the standardised programming languages used in industry, like ladder (LD), structured text (ST), functional blocks diagram (FBD), instruction list (IL) and sequential function chart (SFC) [20]. The fourth part is for user guidelines, the fifth part is for communications between PLC and devices and it is important because here there are general rules and standards. The sixth part is for functional safety, the seventh part is a little bit more specialised and is for fuzzy control programming, the eight part are the guidelines for the application and implementation of programming languages, the ninth part is single-drop digital communication interface for small sensors and actuators and the last part is a guide of PLC open XML exchange format for the export and import of IEC 61131-3 projects.

As we see there is plenty of information regarding standards in PLC programming but as we may notice there are complex programming structures and safety measures have to be included at the beginning of the development process. Complex safety functions are normally managed by safety PLCs and this concept is described in another standard, the IEC 61508 functional safety of a functional safety system. This standard is really important to estimate the the risks and the SIL (safety integrity level) of each system [21].

2.7. Similar projects

There are plenty of projects that want to migrate and integrate new controllers to existing machines, this trend is very popular nowadays because there is an increasing need of applying new applications and new automated processes. For example before a normal PLC was able to guarantee a great reaction time but now there is an increased number of field sensors and there are a lot of data driven algorithms that need a device to gather and distribute data to the cloud (Virtual based storage) and many PLCs manufacturers are integrating IoT modules to the PLC, but there is a problem a lot of processes need real time reaction time and cloud based data is not acceptable, so here is PC-based PLC come into play [22].

The use of PLC is a much needed control emphasis that is done because it gives much more operational availability, for example there is a ROV (Remote operated vehicle) called Remora III, designed by Phoenix, that has been operating in search and rescue operations for more than 16 years, and it has been really successful in the activities that it's performed, but there is a need to increase its service capability and operational availability and for this an integration to a PLC was done. With this migration there is a clear increase in application, easy programmable interfaces, world wide availability and much more just by simply integration a PLC control system [23]. This is a simple example on how integrating new PLC systems can help you increase your operational availability and new application can be implemented.

Another good example, that is much closer to us, is the integration of TalTech's Scara robot with the same controller that we will be using for this thesis. This migration and integration was done to prove that replacement and integration of robots with its own basic control system with a customised control system, and much more powerful, is a convenient approach. This project was tested with two experiments, one to measure the accuracy of motion control which resulted to be really accurate and the second one was a performance testing, which the robot performed without any sign of error. This project proved that migrating to a much more powerful controller leaves the scara robot in a better position and with new capabilities for future applications [24].

There is an increase amount of robots that are being integrated and migrated to new PLC systems, specially the PC-based systems, because they can benefit a lot of areas: Education, fun, industries, processes, manufacturing, etc.

2.8. Conclusions

As we have seen from different references taken into account in this literature review we are now aware why migrating a very powerful controller to an existing machine is more than worth it, there are many applications that will be available once this master Thesis project is successful and keeping with the trend on industrial applications and robotic application will just bring benefits to the university.

Industry 4.0 is taking big steps towards development of industrial automation and robotics and it is shaping the society into a more capable society. This does not mean that robots will be doing all the work, as stated humans are necessary for a near free-error automatic system and this will always be needed in order to succeed in a very competitive world.

With the integration of a new controller new applications will be available but we always have to be aware that with new application more security threats can be created as well and security is a very important subject we have to take into account if we want to create a general solution for migrating the CX 2042 controller to any existing machine and we always have to follow the standards in industry in order to be competitive.

This thesis project will benefit the university in an amazing way and it is needed if we want to keep the pace with the evolution of industry 4.0 and the development of automation in industry. This is just a small step that TalTech has to take.

In order to successfully carry out this thesis project we need to solve more specific objectives, such as:

- Be able to successfully migrate and integrate all the current applications of a robot to the new Beckhoff CX2042 controller. To be more specific this thesis will rely in FESTO’s tripod EXPT delta robot, this does not mean that we will only carry out a solution for this robot, we will be creating an universal solution to fully integrate this controller to new applications and robots. If successfully done we will be able to translate this to the industrial field and be able to propose new application to old machines or robots with the new controller with a solution that will be adaptable to any field

- Since there is history while making this solution, we need to be able to create a fully functional communication network that will adapt to the different communication protocols that each machine or robot has, this will benefit the user in a vast way, since one of the main issues while working with in this field is the vast amount of time and effort it has to create a new network to communicate each component

- Once everything is fully functional we want to keep with the pace of industry 4.0 and integrating new applications will be one of the main sub objectives, in this case this thesis will focus in creating a remote access to the machine and integrating some vision modules that will open a wide variety of new application for the machine.

All of these secondary objectives will be attacked once the universal solution of integration of the new controller is available. It is really important to note that making this solution will benefit not only the university, but also many industries that are interested in this application.

This project will be a really interesting challenge since previous work groups have tried to integrate the old-phased out controllers to the CX2042 controller without success and recommending to change the whole motion drive system into one from ABB to have compatibility. Of course this is a good solution to make but it will mean an increase in spending and it will not be worth it if we translate this action to an industry where a machine or a system will cost a lot of money. In industries there is always a demand on spending as less money as possible to make a ROI (return of inversion) worth it.

3. FESTO EXPT TRIPOD ROBOT AND BECKHOFF CX2042 PLC HARDWARE SPECIFICATIONS

3.1 Overview

Festo EXPT tripod robot is a parallel kinematic system, better known in robotics as a delta robot. These robots are widely used for pick and place applications in industries, as well as labelling, gluing, sorting, grouping, and other type of applications. There are different types of delta robots in the market, some of the most notorious delta robots are: prismatic, rotational, linear and cable delta robots, they all have the same principle of functionality, three arms connected to universal joints, but they differ on how movement is transferred to the end effector, for example one of the most common delta robot is the one based on rotational joints which use high torque motors mounted on a fixed platform and connected perpendicular to an arm.

The FESTO tripod robot is based on linear movement, which has a high-speed carriage unit that gives free movement in three dimensions to the effector platform or mobile platform. Normally this type of robots have a really high precision, making this robots very unique and well established in the industry. From factory design the EXPT robot is based on Festo products and also having a FESTO controller (CMXR-C1) which is capable of controlling this type of systems without a problem, made in part specifically for this type of applications.

Mechanically speaking the Festo EXPT tripod robot has three motors that make the main carriage move along the rail with a toothed belt system, and the carriage is attached to the main arm of the system with two universal joints in the connection with the carriage and two other universal joints in the connection to the effector platform, making this system a 3 DOF parallel manipulator. Figure 3.1.1 shows the general diagram of the mechanical parts of the EXPT tripod robot, some measurements may vary depending on the model of the EXPT robot, this project will be based on the EXPT-45 model. One extra motor is exactly at the center of the effector platform to have control of both the suction cups located at the bottom, this can be changed in a future with a gripper or another mechanism.

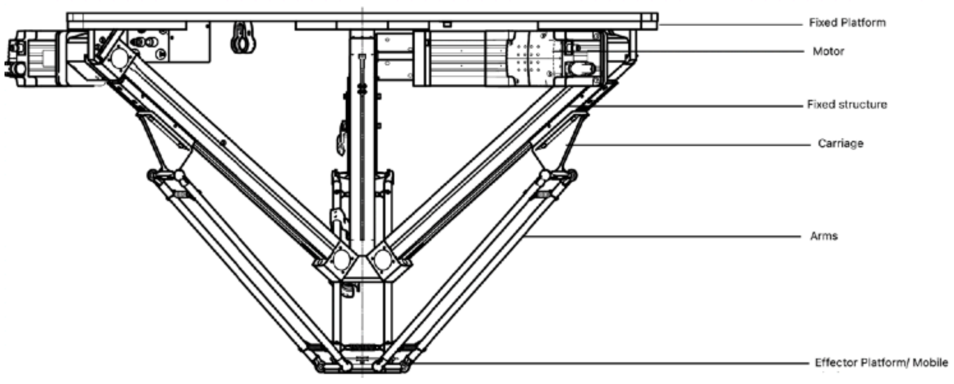


Figure 3.1.1. Festo EXPT tripod Robot diagram [25]

The electrical cabinet of the EXPT robot is located below the robot and it contains all the power stage of the robot meaning that it has the main power line of the motors (240Vac) connected to the motors and the drives, as well as the control stage of the robot (24Vdc) connecting all the electronic components, such as digital inputs, digital outputs, power supply, safety relays and the controller. It is important to note that most of the electric cabinet is from the original Festo design and the only that will be changed will be the controller (PLC). All other components will stay as they are and since there had been two previous works on this robot, some digital inputs and outputs will be changed and connected correctly. This is an important step while migrating any robot or machine to a new controller, compatibility between voltages, between inputs and outputs is a must if there is going to be a migration of the control system. Figure 3.1.2 shows the electrical cabinet from the control (24Vdc) stage viewpoint.

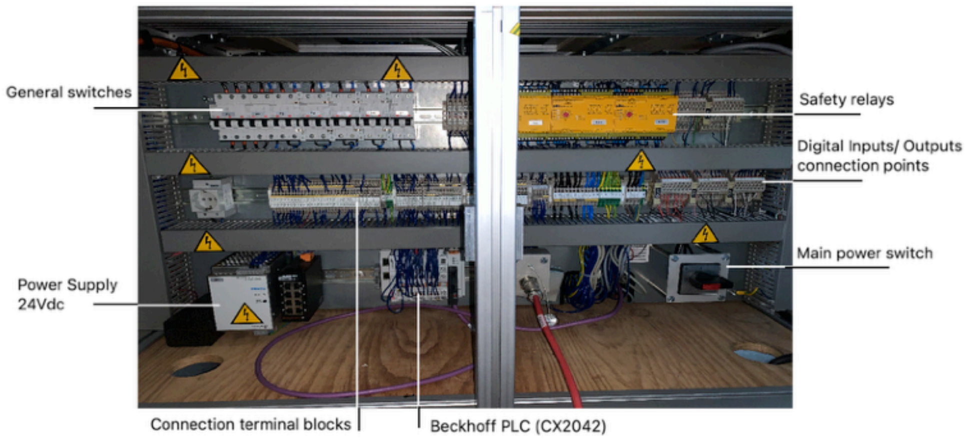


Figure 3.1.2. Festo EXPT tripod Electric cabinet

3.2 Topology and Architecture of the system

One of the most important steps of migrating and integrating a system into a new controller or even designing from scratch a new system is to understand the general topology or automation architecture of the system. The topology of the system shows in a very general way how the components are connected between each other and what type of communication is used in the system to connect each component, also the amount of I/O modules that will be used is shown in the general topology. In this topology just the general control scheme is shown and all the small connections to the terminal blocks are not shown, this connections are shown directly in the fabric handouts from the machine design. The topology will as well help other engineers in future projects and future modifications. the next figure shows the automation architecture of the thesis project, in which the controller CX2042 is shown as the main controller in the working station and all the other connections are part of the electric cabinet

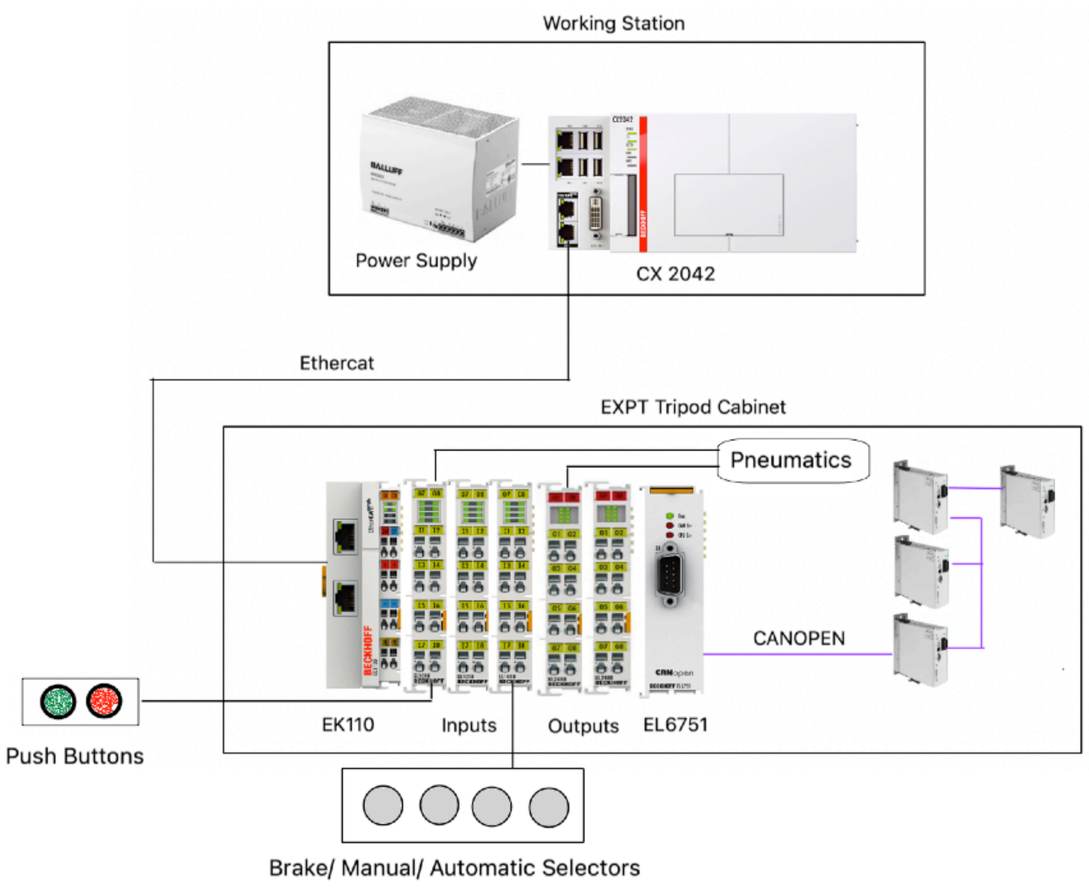


Figure 3.1.2. Automation topology of the system

In the automation topology it is very important to show what type of fieldbus is used to communicate each component, that is why the colour of the cables change according to the communication protocol that is used.

Another important part of the information and a core part of integration of controllers are the inputs and outputs list. Along with the topology knowing the amount of inputs/outputs of the system is extremely important to understand and be aware of what will be done. These two steps will guaranteed that there are high chances of accomplishing the full integration of the system, in fact these two steps are how a project is created and every automation engineer can base it decisions on the information provided by the I/O list and the automation topology. The following tables show the inputs and outputs list.

Table 3.1 Digital Inputs Term 2 module (EL1018)

No	Type of signal	Address	Name (Twincat)	Function	Terminal Block
1	**DI (Term 2)**	1577.0	bEmergencyK20	Emergency button	K20.P5
2	**DI (Term 2)**	1577.1	bSensorTemp	Temperature sensor	DI0.5
3	**DI (Term 2)**	1577.2	bEmergencyK22	Emergency relay K22	K22.P5
4	**DI (Term 2)**	1577.3	bSensorH	Sensor signal	DI0.6
5	**DI (Term 2)**	1577.4	bAutoMode	Automatic mode selection	DI1.2
6	**DI (Term 2)**	1577.5	bSensorW	Sensor signal	DI0.7
7	**DI (Term 2)**	1577.6	bManualMode	Manual mode selection	DI1.5
8	**DI (Term 2)**	1577.7	bDoorSensor	Door sensor Emergency relay K21	K21.P5

Table 3.2 Digital Inputs Term 3 module (EL1018)

No	Type of signal	Address	Name (Twincat)	Function	Terminal Block
1	**DI (Term 3)**	1578.0	bStartButton	Start Push button	DI1.1
2	**DI (Term 3)**	1578.1	Free to use		
3	**DI (Term 3)**	1578.2	bAir	Air pressure (NO)	
4	**DI (Term 3)**	1578.3	Free to use		
5	**DI (Term 3)**	1578.4	Free to use		
6	**DI (Term 3)**	1578.5	Free to use	Free to use	X1.203
7	**DI (Term 3)**	1578.6	bBreakSystem	Breaks are active	
8	**DI (Term 3)**	1578.7	Free to use	Free to use	X1.179

Table 3.3 Digital Inputs Term 4 module (EL1018)

No	Type of signal	Address	Name (Twincat)	Function	Terminal Block
1	**DI (Term 4)**	1579.0	bTopRight	Top right pressure sensor	DI2.1
2	**DI (Term 4)**	1579.1	bTop	Top pressure sensor	DI2.2
3	**DI (Term 4)**	1579.2	bGripTop	Pressure sensor for tool	DI2.3
4	**DI (Term 4)**	1579.3	bGripBot	Pressure sensor for tool	DI2.4
5	**DI (Term 4)**	1579.4	bBottom	Bottom pressure sensor	DI2.5
6	**DI (Term 4)**	1579.5	bBottomLeft	Bottom left pressure sensor	DI2.6
7	**DI (Term 4)**	1579.6	bLeft	Left pressure sensor	DI2.7
8	**DI (Term 4)**	1579.7	bTopLeft	Top left pressure sensor	DI2.8

Table 3.4 Digital Outputs Term 5 module (EL2008)

No	Type of signal	Address	Name (Twincat)	Function	Terminal Block
1	**DO (Term 5)**	1562.0	Free to use		
2	**DO (Term 5)**	1562.1	Free to use		
3	**DO (Term 5)**	1562.2	Free to use		
4	**DO (Term 5)**	1562.3	blightstart	Light for start button	
5	**DO (Term 5)**	1562.4	Free to use	To drives	
6	**DO (Term 5)**	1562.5	Free to use	Free to use	
7	**DO (Term 5)**	1562.6	Free to use	To drives	
8	**DO (Term 5)**	1562.7	blightstop	Light for stop button	

Table 3.5 Digital Outputs Term 6 module (EL2008)

No	Type of signal	Address	Name (Twincat)	Function	Terminal Block
1	**DO (Term 6)**	1563.0	bTopVacuum	Vacuum for top position	DO1.1
2	**DO (Term 6)**	1563.1	bBottomVacuum	Vacuum for bottom pos	DO1.2
3	**DO (Term 6)**	1563.2	bTopRightVacuum	Vacuum for top right pos	DO1.3
4	**DO (Term 6)**	1563.3	bLeftVacuum	Vacuum for left possition	DO1.4
5	**DO (Term 6)**	1563.4	bGrip1Vacuum	Vacuum for tool #1	DO1.5
6	**DO (Term 6)**	1563.5	bGrip2Vacuum	Vacuum for tool #2	DO1.6
7	**DO (Term 6)**	1563.6	bTopLeftVacuum	Vacuum for top left pos	DO1.7
8	**DO (Term 6)**	1563.7	bBottomLeftVacuum	Vacuum for bot left pos	DO1.8

Term 7 (EL2008) digital outputs module is free to use completely, this is just in case there is a need fro a spare digital output module.

3.3 Beckhoff CX2042 PLC overview.

Beckhoff CX2042 is a PLC from the embedded PC family of Beckhoff, where they have combined PC technology along with the modular I/O units all installed in a DIN rail in a cabinet. The CX series family is suitable for all controla tasks that an engineer can come up with, since there is a combination of industrial PC and hardware PLC, as well as a lot of installation space is reduced since there will be added only the units that are required for a specific application. Specifically the CX2042 comes with a Xeon Intel CPU with a clock rate of 2.2 GHz (4 cores). The basic module comes with an internal memory of 8 GB RAM that can be upgraded to 64 GB. Microsoft Windows 10 IoT enterprise is used as operating system and to program the PLC, TwinCat3 is used as software. This PLC will be the main brain of our system and everything will come to this component in order to make the EXPT work correctly.

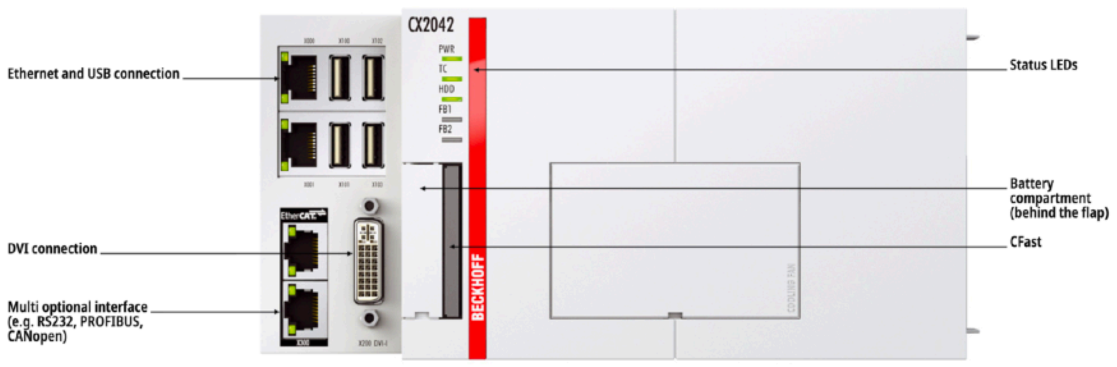

Ethernet and USB connection
DVI connection
Multi optional interface (e.g. RS232, PROFIBUS, CANopen)
CX2042
PWR
TC
HDD
FB1
FB2
BECKHOFF
Status LEDs
Battery compartment (behind the flap)
CFast

Figure 3.1.3. Beckhoff CX2042 PLC basic module overview

The individual components of the CX family come as modules that can be connected in series with standard widths (Beckhoff widths). There are a good amount of modules for I/O Inputs and oputputs, power supplies, communication modules, master/slave modules, interface modules, among others.

For this thesis project and according to the topology the power supply that will be used for the CX2042 will be the BALLUFF BAE0003 and will be installed in the same DIN rail as the CX2042.

In the EXPT cabinet there will be another power supply but this time for 24 VDC and will be the one that the original system has from FESTO, we will power up the EK1100 that will be our Ethercat coupler. This module is used to have a decentralised system, meaning that we can have the I/O modules inside the machine cabinet or in another location and communicate the CPU with the module with only one fieldbus, that in this case will be Ethercat fieldbus. This in the industry is a very common practice, since there is a lot of reduced costs because of cable and commissioning is much easier and faster than having all the components in a single cabinet. In the same rail as the EK1100 we will have our input and output modules (EL1018 for Inputs and EL2008 for outputs).

To communicate the motor drivers we will need a CANOpen module in the system (EL6751), we are using this module since the idea of this thesis project is to migrate every component to the new system and try not to change a component. It is important to remember that Beckhoff has a wide range of communication interfaces modules, we can communicate CanOpen fieldbus, Profibus, Profinet, Modbus, and many other widely known communication protocol.

Knowing the outputs and inputs of the system we can select the inputs and outputs of the Beckhoff PLC. As we can see we now know that we will need three inputs card and 2 output cards. It is important to mention that in Beckhoff PLC systems since they are modular systems we can add and take the supported amount of inputs and outputs of the system (amount)

3.4 Festo Motor drives CMMP-AS

The EXPT tripod robot from FESTO works with 4 motor, three of these motors are the same model (EMMS-AS-100-S-HS-RMB) and they are all connected to a toothed belt, which will be the one in charge of moving the carriage along the rail. These three motors are controlled by FESTO drives, model CMMP-AS-C5-3A. These motor drives are already phased out and one of the original motor controllers was damaged by previous tests, that is the reason why one of the motors is connected to a newer model of these motor drives, the CMMP-AS-C5-3A-M0. The last motor is a rotatory motor which is connected to a CMMP-AS-C2-3A, same model of the other motor drives but with less torque and speed. All four motor drives can be configured through the FCT tool, which is the proprietary software from FESTO to change the factory parameters of the drives, this tool is very important while commissioning these type of drives under a project. Since there are previous studies regarding these motors and their controller/drive, it is recommended to have a look on how the controllers are configured and parametrised. This thesis work will focus on the important parameters to communicate the drives to the CX2042 and since the machine has motor controllers that are phased out then this have caused mayor issues with the integration of the drives with the controller.

The following parameters have to be checked in the FCT tool:

- Application data
- Fieldbus that will be used
- CanOpen Node ID, number for the node
- Type of protocol from the CanOpen fieldbus, FHPP or CiA402 standard protocol
- Factor number, to read correctly the digits
- Components connected to the controller: motor, rail, sensors, etc
- Limit switch by software
- Homing or calibration
- Interpolated position to have a correct function in TwinCat
- Length of the carriage movement, same length as the toothed belt

4. UNIVERSAL SOLUTION FOR MIGRATING AND INTEGRATING CX2042 PLC

4.1 TwinCat3 PLC and modules integration

In order to integrate any system to Beckhoff PLC it is first necessary to be sure the topology is accurate and that there is enough amount of inputs and outputs modules available to integrate the whole system. Once this step is done is is necessary to understand how TwinCat3 works. Twincat 3 is an all-in-one software for Beckhoff automation in which all available applications are integrated into a single software. It is important to note that this software uses different licenses and if the machine in which it is necessary to integrate a Beckhoff PLC is working locally and always attached to a PC it won't be necessary to buy the licenses, but in an industrial environment buying the licenses are completely a must. For this case we will be using the trial/free licenses to work with the EXPT machine, since the machine will only work while the computer is connected to the EXPT. It is important to mention that the following steps mentioned in this project will guide in a very general way on how to create a new project and how to integrate the external or not the same manufacturers components as the Beckhoff PLC or FESTO hardware. It is not intended to follow strictly a guide and the following steps were done following the knowledge of previous programs and getting knowledge from official/published manuals from Beckhoff. Previous work groups had followed strictly some guides, having a negative result while trying to integrate Festo components to Beckhoff controller.

The first step in every software is to create a new project (New Twincat Project XML), in TwinCat 3 it is important to note that the environment is based on Visual Basic, meaning that it will be familiar to anyone that has used any other software based on visual basic, selecting the standard project will give you a general template of the project. The second step in communicating the PLC to the software is to link the PLC IP address to the software in order to create a bridge of communication. To do this, select the local dropdown menu in the upper part and select Choose target system…, this will show a new window where all the devices that are connected to the network will be shown, in this case only one PLC is connected to our network. It is important to check the IP and name of the PLC in order to have correct communication.

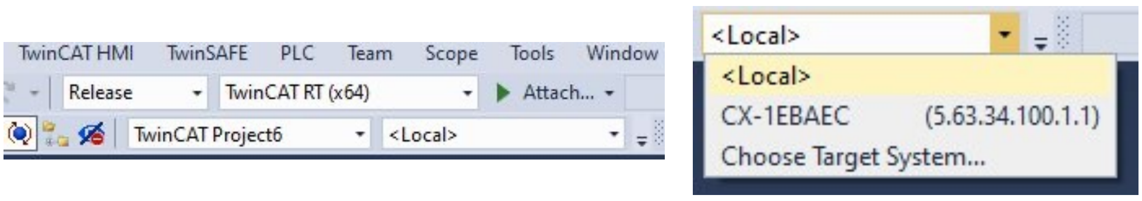
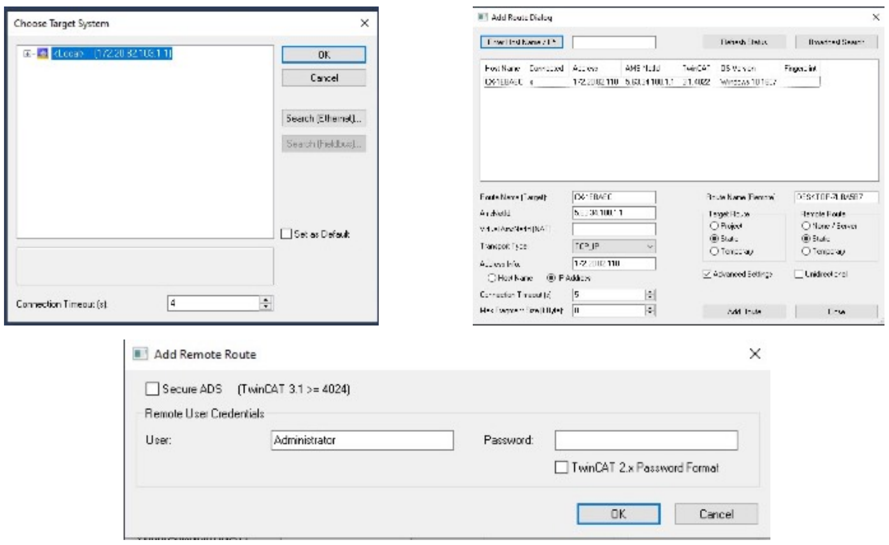


Figure 4.1.1. Choosing the Target system from the dropdown menu

Once in the choose target system new window it is necessary to select the Search (Ethernet) option, this option will work in this case since the PLC is just connected to the LAN network where the machine will be working. A new window will appear and the Broadcast search option will show the controllers available in the LAN network. Selecting the controller, making sure the correct IP is selected and the Host name is correct will let you Add the Route (PLC) to the software. Adding the PLC to TwinCat needs a password in order to successfully connect the controller, the password is the default for any controller, which is the number 1.



<Local> (172.20.82.103.1.1)
Set as Default
Connection Timeout (s): 4

Host Name	Connected	Address	AMS NetId	TwinCAT	OS Version	Fingerprint
CX-1EBAEC	x	172.20.82.110	5.63.34.100.1.1	3.1.4022	Windows 10 1607	

Route Name (Target): CX-1EBAEC
AmsNetId: 5.63.34.100.1.1
Virtual AmsNetId (NAT):
Transport Type: TCP_IP
Address Info: 172.20.82.110
Host Name [✓ IP Address]
Connection Timeout (s): 5
Max Fragment Size (kByte): 0
Route Name (Remote): DESKTOP-7LBA5B7
Target Route: Project [✓ Static] Temporary
Remote Route: None / Server [✓ Static] Temporary
[✓ Advanced Settings]
Unidirectional

Secure ADS (TwinCAT 3.1 >= 4024)
Remote User Credentials
User: Administrator
Password:
TwinCAT 2.x Password Format

Figure 4.1.2. (TOP LEFT) Choosing the Target window, (Top right) Adding route to the software, (bottom) password input to connect the PLC to TwinCat3

Once the PLC is added to the software we can use a very easy way to add all the modules and some of the components connected to the Fieldbus. It is important to note that some components that are made by other companies, like FESTO, SIEMENS, ABB, etc… will not be recognised and some manual addition has to be done. To add the components to the network we select from the project tree, under I/O, Devices option and select the scan option.

This scan option will add all the components in the bus. You have to be clear on which network card of the computer you are connecting to, since this is very important (this can be checked under network setting of windows or any other OS available). Once the scan option is selected and the correct network drive is selected the components will be added to the project tree.

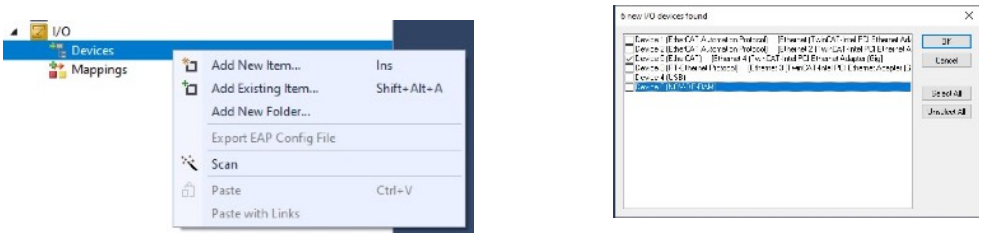


Figure 4.1.3. Selecting the Scan option from the project tree and selecting the correct network adapter

In the project tree it is very clear which components are added, since each one of them has the name of the component. Also it is important to mention that if there is doubt about the components that were added, the topology on how each device is connected can be shown in the device options - topology. This way the engineer will be completely sure that all the components are added accordingly. Also as any other engineering software of automation, any other components can be added manually and avoid the automatic adding of the components through the scan option.

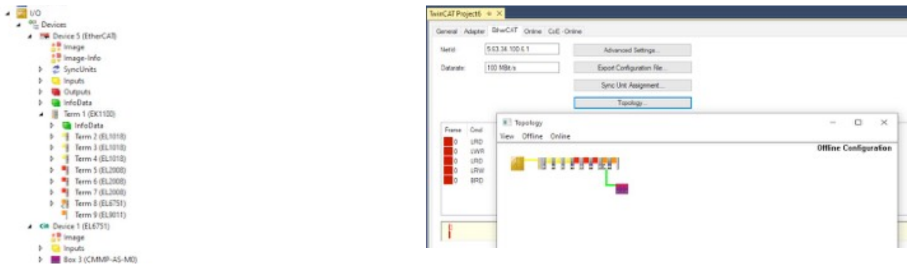


Figure 4.1.4. (Left) Project tree with all the components added. (Right) Topology as shown in TwinCat3

4.2 Integration of components and drives from other manufacturers.

As mentioned before there is the option of adding any components from other companies, as long as the correct modules for having compatibility of the same communication protocol exists, which in this thesis project it will be focused on communicating Canopen devices into Ethercat communication protocol. The first important thing to know is if there is correct module to transfer all the important information into the field bus. The EL6751 will be used in regards of this thesis, according to Beckhoff:

"The Terminal enables within an EtherCAT Terminal network the integration of any CANopen devices and can either be master or slave. In addition, general CAN messages can be sent or received – without having to bother with CAN frames in the applications program."[26]

This small terminal will be the one to communicate the Canopen device to the PLC, but it is really important to understand each communication protocol that will be used in any control system. In this case the Canopen communication protocol works on sending and receiving SDOs and PDOs, as mentioned in the literature review. Each device and each manufacturer has its own data to communicate the device to any controller, meaning that the file needed to integrate any external component is given by the manufacturer, in this specific case FESTO. When working with Canopen devices a .tce or .gsd file extension is needed from the manufacturer and this will change if there is a need to integrate any other communication protocol. For example if there is a Siemens motor drive with Profibus communication, it will be needed a Profibus to Ethercat module, the EL6731 which is the Profibus master/slave terminal and the .gsd extension files, which will integrate the device to the Beckhoff controller.

To add a component from other manufacturer and once the topology is fully understood, adding an existing item in the communication terminal (EL6751) has to be selected and the file with the mentioned extensions has to be selected, this will depend on what component has to be added and where the file has been saved. Once the component is selected it will appear in the project tree under the communication terminal and it can be once again checked in the topology from the device options.

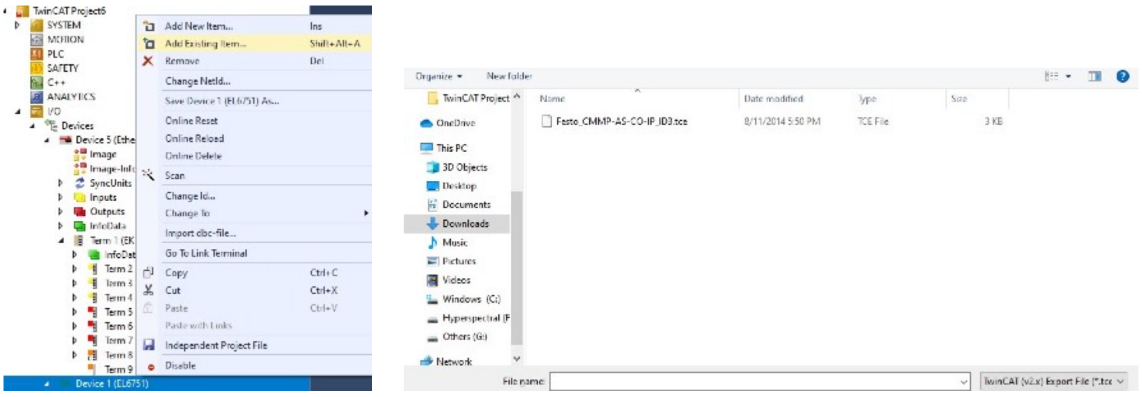


Figure 4.2.1. (Left) Project tree selection from the communication terminal to add an existing item. (Right) Window to select the correct file with the correct file extension

In this specific thesis project the objective is to integrate four motor controllers from FESTO, model CMMP-AS. Two of this controllers are CMMP-AS-C5-A3, one of the controllers is a CMMP-AS-C2-A3 and one new controller CMMP-AS-C5-A3-M0, which is the newer controller from FESTO. This controller was needed because in previous tests one controller was damaged and since the older controllers are already phased out, a newer controller had to be installed, but as mentioned it does not matter if they are old or new, as long as they are integrated into the same communication protocol, it will work correctly.

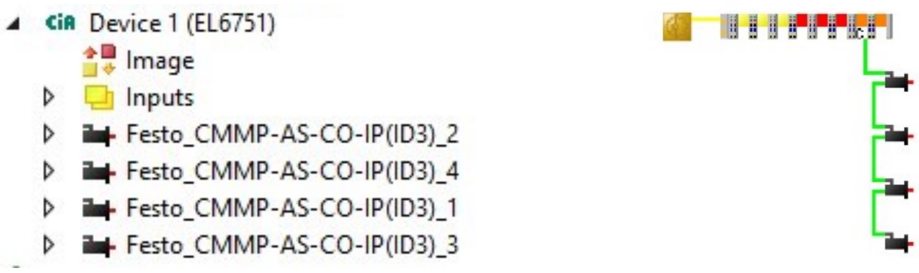
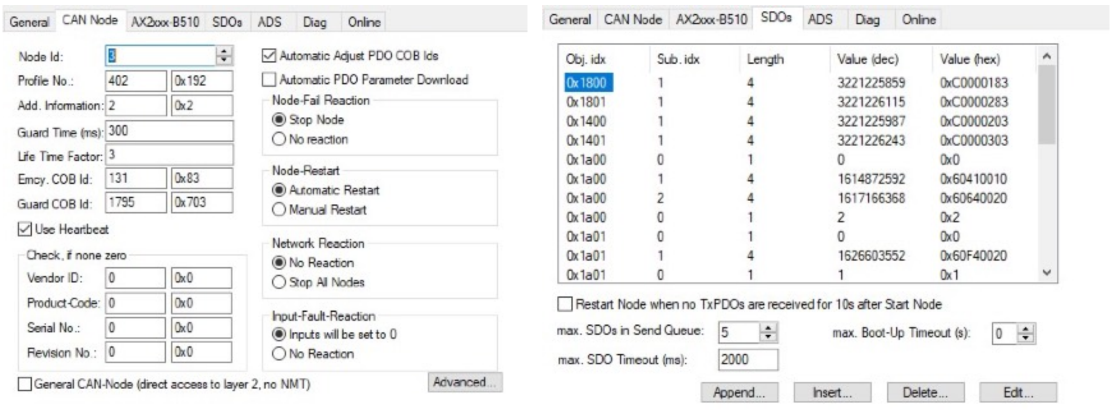

Device 1 (EL6751)
Image
Inputs
Festo_CMMP-AS-CO-IP(ID3)_2
Festo_CMMP-AS-CO-IP(ID3)_4
Festo_CMMP-AS-CO-IP(ID3)_1
Festo_CMMP-AS-CO-IP(ID3)_3

Figure 4.2.2. Four Festo motor drivers integrated into the project tree and the topology after the motor drives are integrated.

An important note regarding any fieldbus communication is that each component in the bus has to have its own unique ID (Profibus) or node (CanOpen). In CanOpen there is a maximum amount of 127 slaves per each master module that is installed. This Can node number can be parametrised in the FCT tool from festo for each driver and in the parameters of each motor drive added to twincat 3, this two numbers have to be the same in both configurations in order to get a successful communication.

But there is another detail regarding the FESTO drivers, since each of this drivers have a unique SDOs there is a need to change the last number from each SDO where it is needed. This can be changed in the SDOs tab while selecting each motor driver and they have to match the CAN node Id. For example if our CAN node Id is three (3), then the SDOs values have to end with the number 3, also is important to note that this is given by each manufacturer and it depends on what is needed for each motor drive.



Node Id:	3	
Profile No.:	402	0x192
Add. Information:	2	0x2
Guard Time (ms):	300	
Life Time Factor:	3	
Emcy. COB Id:	131	0x83
Guard COB Id:	1795	0x703

☑ Use Heartbeat

Check, if none zero

Vendor ID:	0	0x0
Product-Code:	0	0x0
Serial No.:	0	0x0
Revision No.:	0	0x0

☐ General CAN-Node (direct access to layer 2, no NMT)

☑ Automatic Adjust PDO COB Ids
☐ Automatic PDO Parameter Download

Node-Fail Reaction: [✓ Stop Node] No reaction

Node-Restart: [✓ Automatic Restart] Manual Restart

Network Reaction: [✓ No Reaction] Stop All Nodes

Input-Fault-Reaction: [✓ Inputs will be set to 0] No Reaction

Obj. idx	Sub. idx	Length	Value (dec)	Value (hex)
0x1800	1	4	3221225859	0xC0000183
0x1801	1	4	3221226115	0xC0000283
0x1400	1	4	3221225987	0xC0000203
0x1401	1	4	3221226243	0xC0000303
0x1a00	0	1	0	0x0
0x1a00	1	4	1614872592	0x60410010
0x1a00	2	4	1617166368	0x60640020
0x1a00	0	1	2	0x2
0x1a01	0	1	0	0x0
0x1a01	1	4	1626603552	0x60F40020
0x1a01	0	1	1	0x1

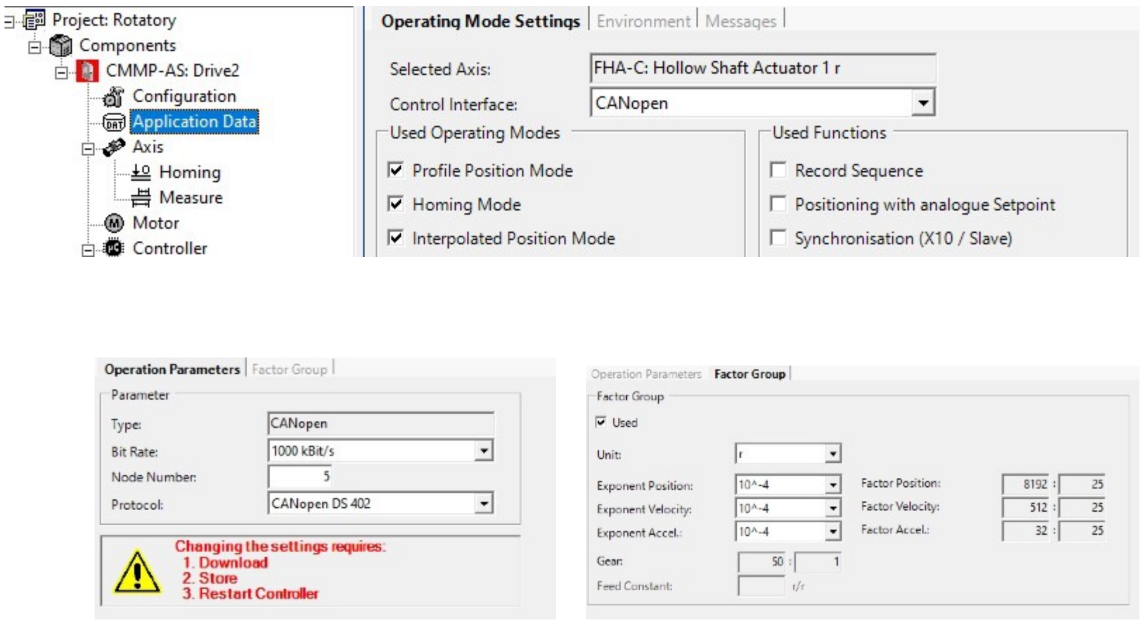
☐ Restart Node when no TxPDOs are received for 10s after Start Node

max. SDOs in Send Queue: 5
max. Boot-Up Timeout (s): 0
max. SDO Timeout (ms): 2000

Figure 4.2.3. Node Id shown in the CAN node tab and changing the SDOs final value to match the CAN node Id

In the FCT tool from Festo this parameters have to match with the ones from Twincat 3 software, therefore in this thesis project just the important parameters will be shown, which are: The CanOpen node id, the factor group escalation (which is how the numbers from the driver are interpreted in the twincat 3 software) and the most important that is the application data configuration so that the driver can communicate through Canopen with the CiA402 protocol, which is a standard in CanOpen communication. The FCT tool can be used with all FESTO drivers and depending on the firmware of each driver the FCT tool will use a different version. For the purposes of the thesis the lastest version (4.xx) was used for the newest drive and the version 2.14.xx was used for the phased out motor controllers.

In the application data option, the way the driver communicates can be changed and there are two options: CiA402 and FHHP protocol from Canopen. Since Beckhoff controllers can only communicate through the CiA402 standard then this option is selected. From the fieldbus menu, the node id can be changed and the factor groupo as well. Due to recommendations from the manufacturers the $x\ 10^{-4}$ factor was used and this value has to be changed in the parameters of the axis.



Selected Axis: FHA-C: Hollow Shaft Actuator 1 r
Control Interface: CANopen

Used Operating Modes
- [✓] Profile Position Mode
- [✓] Homing Mode
- [✓] Interpolated Position Mode

Used Functions
- [ ] Record Sequence
- [ ] Positioning with analogue Setpoint
- [ ] Synchronisation (X10 / Slave)

Parameter
Type: CANopen
Bit Rate: 1000 kBit/s
Node Number: 5
Protocol: CANopen DS 402

Changing the settings requires:
1. Download
2. Store
3. Restart Controller

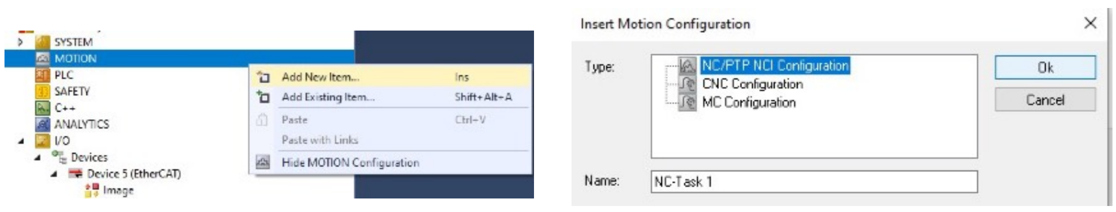
Factor Group
[✓] Used
Unit: r
Exponent Position: 10^-4 Factor Position: 8192 : 25
Exponent Velocity: 10^-4 Factor Velocity: 512 : 25
Exponent Accel.: 10^-4 Factor Accel.: 32 : 25
Gear: 50 : 1
Feed Constant: r/r

Figure 4.2.4. FCT tool important parameters that need to be changed in order to have a successful communication

4.3 Motion control in Twincat 3 Software

Motion control in a PLC is a really important part while working with any motor drive or motor controller, each software has its own pros and cons while working with motor drives and with the motion modules. One of the important aspects and a really great advantage of twincat 3 over other softwares is that it is already integrated inside the specific functions, all that is needed is to download the specific extension of the engineering functions and add the components that are needed. In this project the TF5000 function package was used, This function allows the system to have up to 10 axes added to the motion control section, extendable up to 255 with the TF5020. Without this package there is no option to add axes in the motion section of the software. It is important to remember that licensing in Twincat 3 is free as long as it is used locally on a PC that will always be connected to the PLC which is the case of this project. If there is a system outside in an industrial environment then buying the license with the specific functions will be needed.

To add axes to the motion section, just select add new item from the project tree and once selected a dialog appears asking which type of configuration will be used, there are three options the CNC configuration, the MC configuration and the NC/PTP configuration. CNC configuration stands for Computer Numerical Control system, the MC configuration stands for Motion control for robotics control and NC/PTP stands for Traditional numerical Control system or point-to-point system, the difference between each other is that in the CNC configuration has a local computer that is used to control the positioning and store data, the MC configuration is for robotics control specifically, which is not the case for this thesis. NC/PTP configuration is the point-to-point configuration which control only the position of the components, thus the driver will have the ability to control the velocity and acceleration of each connected motor, which is the case in this thesis project.



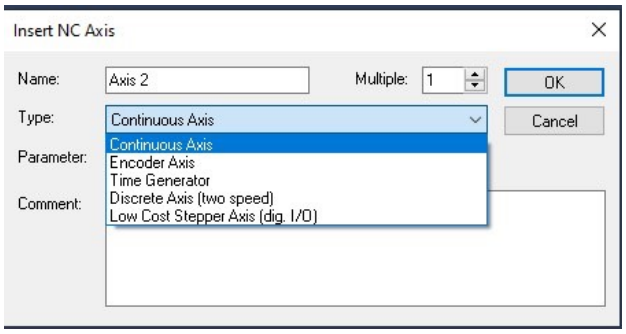
Type: NC/PTP NCI Configuration

Name: NC-Task 1

Figure 4.3.1. Adding the NC/PTP configuration to the motion section of the project

Adding a NC configuration system will add a task in the project tree and very detailed parameters can be changed in the task, the most important of this task parameters is the cycle time in which each component will be sending and receiving information. This cycle time is configured as well in the drivers with specific manufacturer tools, which this case of FESTO is the FCT tool for motor drivers and they both have to be the same in order to make the system work successfully.

Once the task is added, the axes have to be added, there are different options while adding the axes: Continuous Axis, encoder axis, time generator axis, discrete axis (two speed) and los cost stepper axis (dig. I/O). It is clear which is the difference between each other and for this project the continuous axis will be selected for all the motor drives added before


Insert NC Axis

Name: Axis 2

Multiple: 1

Type: Continuous Axis

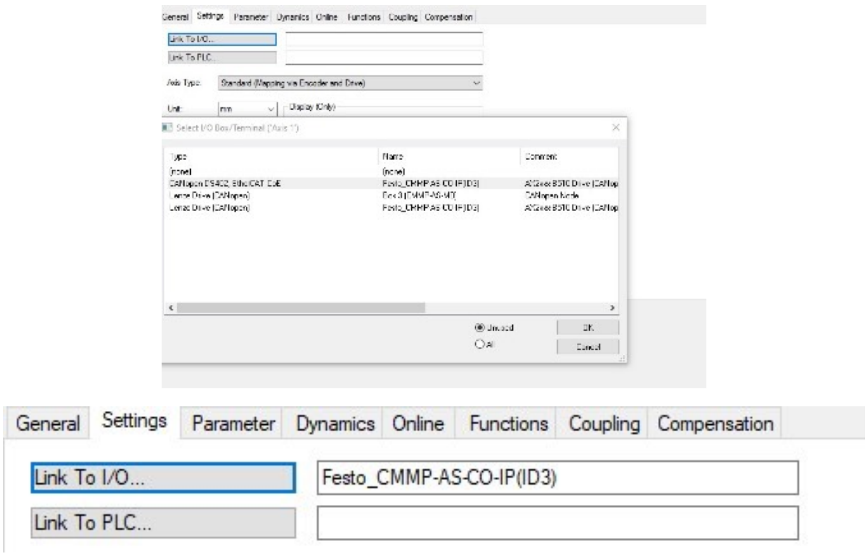
- Continuous Axis
- Encoder Axis
- Time Generator
- Discrete Axis (two speed)
- Low Cost Stepper Axis (dig. I/O)

Parameter:

Comment:

Figure 4.3.2. Adding the Continuous axis to the project tree

Adding the axis has to be done with each motor drive and they have to be linked with each drive that was added previously in the communication terminal (Canopen to Ethercat communication). To link each axis you have to open the axis configuration and select the Setting tab - Link To I/O option. A new dialog window appears and the axis can be linked to the Festo CMMP-AS motor drive that was previously added.



Axis Type: Standard (Mapping via Encoder and Drive)

Unit: mm

Display (Only)

Select I/O Box/Terminal ('Axis 1')

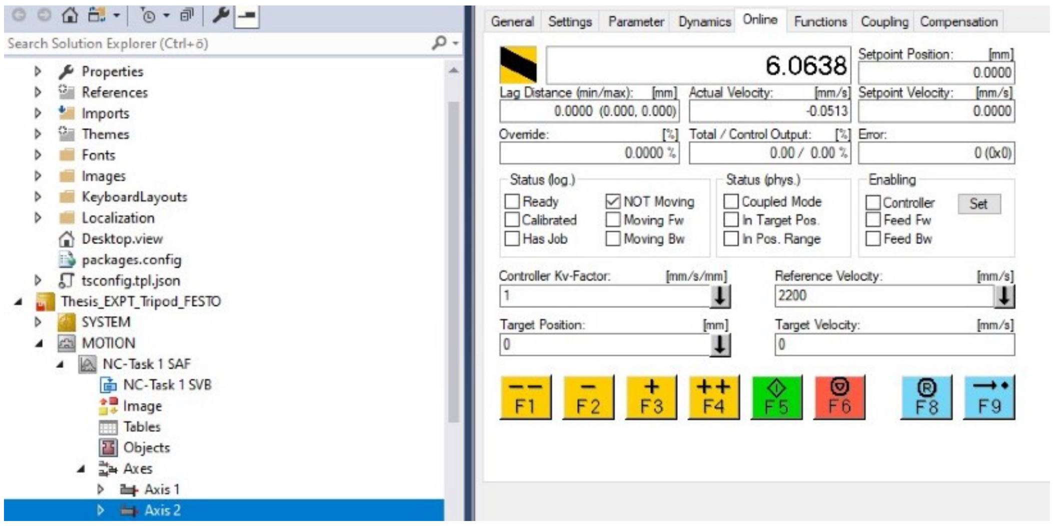
Type	Name	Comment
(none)	(none)	
CANopen DS402, EtherCAT CoE	Festo_CMMP-AS-CO-IP(ID3)	AX2xxx B510 Drive (CANop
Lenze Drive (CANopen)	Box 3 (CMMP-AS-M3)	CANopen Node
Lenze Drive (CANopen)	Festo_CMMP-AS-CO-IP(ID3)	AX2xxx B510 Drive (CANop

(●) Unused ○ All

Link To I/O... Festo_CMMP-AS-CO-IP(ID3)

Figure 4.3.3. Linking the axis to the motor drive in Twincat 3

Once there is a successful link between the motor controller and the Beckhoff CX2042 controller, then there is an online option (which is one of the most important tabs in any automation software) to check if there a value regarding the internal encoder of the drive.


Search Solution Explorer (Ctrl+ö)

- Properties
- References
- Imports
- Themes
- Fonts
- Images
- KeyboardLayouts
- Localization
- Desktop.view
- packages.config
- tsconfig.tpl.json
- Thesis_EXPT_Tripod_FESTO
  - SYSTEM
  - MOTION
    - NC-Task 1 SAF
      - NC-Task 1 SVB
      - Image
      - Tables
      - Objects
      - Axes
        - Axis 1
        - Axis 2

General | Settings | Parameter | Dynamics | Online | Functions | Coupling | Compensation

6.0638

Setpoint Position: [mm] 0.0000

Lag Distance (min/max): [mm] 0.0000 (0.000, 0.000)

Actual Velocity: [mm/s] -0.0513

Setpoint Velocity: [mm/s] 0.0000

Override: [%] 0.0000 %

Total / Control Output: [%] 0.00 / 0.00 %

Error: 0 (0x0)

Status (log.): [ ] Ready [ ] Calibrated [ ] Has Job [✓] NOT Moving [ ] Moving Fw [ ] Moving Bw

Status (phys.): [ ] Coupled Mode [ ] In Target Pos. [ ] In Pos. Range

Enabling: [ ] Controller [ ] Feed Fw [ ] Feed Bw Set

Controller Kv-Factor: [mm/s/mm] 1

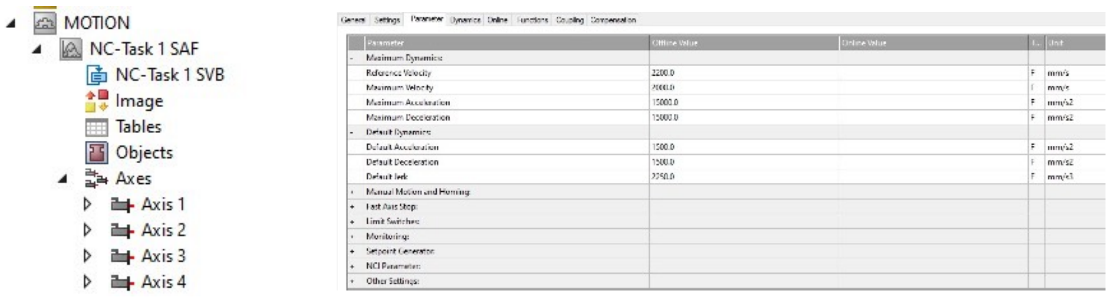
Reference Velocity: [mm/s] 2200

Target Position: [mm] 0

Target Velocity: [mm/s] 0

Figure 4.3.4. Online values of the motor controller shown in twincat 3

The last step to make sure everything is working is to move the axis in the online tab and make sure the physical axis is moving as well. Before making any movement there are important parameters that have to be changed in the axis project tree. These parameters are the velocity that we will be working, the velocity in JOG mode, the velocity in manual movement, acceleration, deceleration, the factor group escalation, and the software limit switches (since we are not using any physical limit switch in the EXPT tripod robot). All of these parameters have to match the original parameters from the manufacturer to make a correct movement of each axis.


- MOTION
  - NC-Task 1 SAF
    - NC-Task 1 SVB
    - Image
    - Tables
    - Objects
    - Axes
      - Axis 1
      - Axis 2
      - Axis 3
      - Axis 4

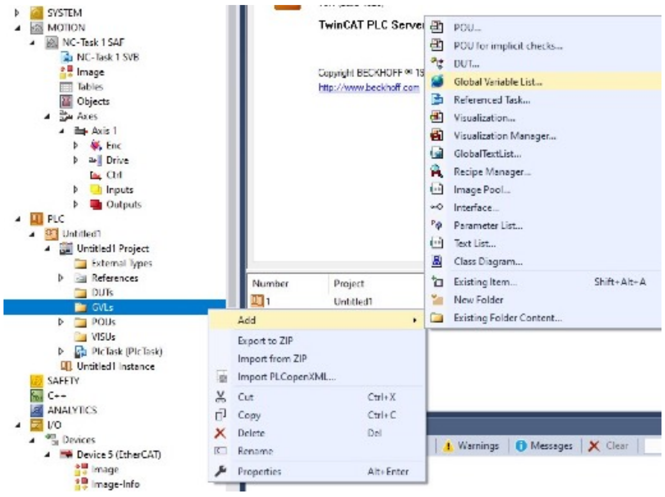
General | Settings | Parameter | Dynamics | Online | Functions | Coupling | Compensation

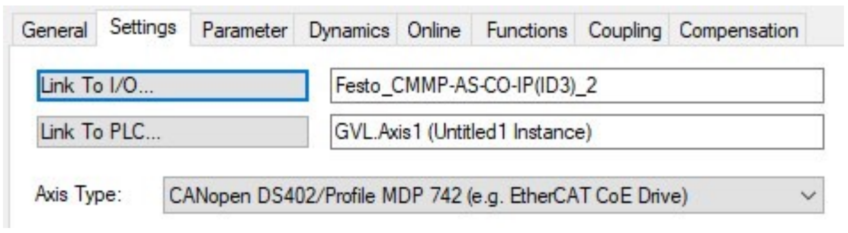
	Parameter	Offline Value	Online Value	T...	Unit
-	Maximum Dynamics:				
	Reference Velocity	2200.0		F	mm/s
	Maximum Velocity	2000.0		F	mm/s
	Maximum Acceleration	15000.0		F	mm/s2
	Maximum Deceleration	15000.0		F	mm/s2
-	Default Dynamics:				
	Default Acceleration	1500.0		F	mm/s2
	Default Deceleration	1500.0		F	mm/s2
	Default Jerk	2250.0		F	mm/s3
+	Manual Motion and Homing:				
+	Fast Axis Stop:				
+	Limit Switches:				
+	Monitoring:				
+	Setpoint Generator:				
+	NCI Parameter:				
+	Other Settings:				

Figure 4.3.5. Parameters for each axis

4.4 Manual movement of each axis using motion control function blocks

Manual movements are always important in any motion control programming, the manual movements can be used to calibrate the axis, to make simple tests and also to make demonstrations in an industrial environment. Beckhoff has the possibility to program all this type of movements thanks to the motion package that include different function blocks, which are already created in twincat 3 to make certain actions with each motor drive. This functions blocks are standard for any motor drive that is integrated to the software, meaning that this solution will work with any type of motor controller from any manufacturer. Before starting to program this function blocks it is really important to link the axis to some variables, using the global variables option in order to declare the axis that will be linked in the software. For this case the default names will be used and we are using the global variable list because it will apply to all the programming that can be done.
Declaring the axis variables is really simple, according to beckhoff it only has to be named as AXIS_REF. This will create a .GVL file extension inside twincat and it will be possible to link to PLC using this file created in twincat. Without linking the axis to the PLC, there will be no option to use the motion function blocks included in Beckhoff motion package.




General | Settings | Parameter | Dynamics | Online | Functions | Coupling | Compensation

Link To I/O... Festo_CMMP-AS-CO-IP(ID3)_2

Link To PLC... GVL.Axis1 (Untitled1 Instance)

Axis Type: CANopen DS402/Profile MDP 742 (e.g. EtherCAT CoE Drive)

Figure 4.4.1. Global variable declaration of the Axis and linking of Axis to the software

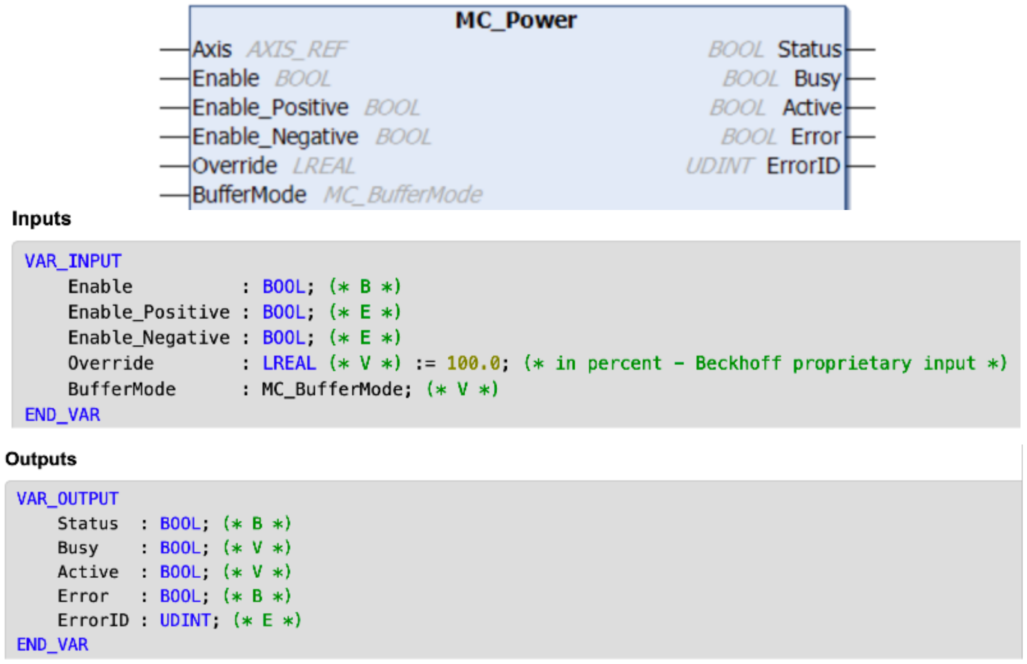
Before programming any of this functions block it is important to know what options are available and that we will be using in this project.

MC_POWER ————————————————— Function block used to enable the drive

MC_RESET ————————————————— Function block to reset any fault

MC_JOG ————————————————— Function block for manual movement

Each of this function blocks have their unique variables, inputs and outputs. It is really important to review all the documentation from each function block. For example for the MC_POWER function block (once declared) the information regarding this function block can be found in official documentation from Beckhoff. The MC_power function block information is as follows:



Inputs

```
VAR_INPUT
    Enable          : BOOL; (* B *)
    Enable_Positive : BOOL; (* E *)
    Enable_Negative : BOOL; (* E *)
    Override        : LREAL (* V *) := 100.0; (* in percent - Beckhoff proprietary input *)
    BufferMode      : MC_BufferMode; (* V *)
END_VAR
```

Outputs

```
VAR_OUTPUT
    Status  : BOOL; (* B *)
    Busy    : BOOL; (* V *)
    Active  : BOOL; (* V *)
    Error   : BOOL; (* B *)
    ErrorID : UDINT; (* E *)
END_VAR
```

Figure 4.4.2. MC_Power function block information [27]

The flow diagram regarding the manual movement of the axis include a certain amount of safety hardware components, such as emergency buttons, door safety sensor, safety relays. The following flow diagram shows the functionality of the manual movements and what decisions and components will be taken into account to perform a simple manual movement of the axis.

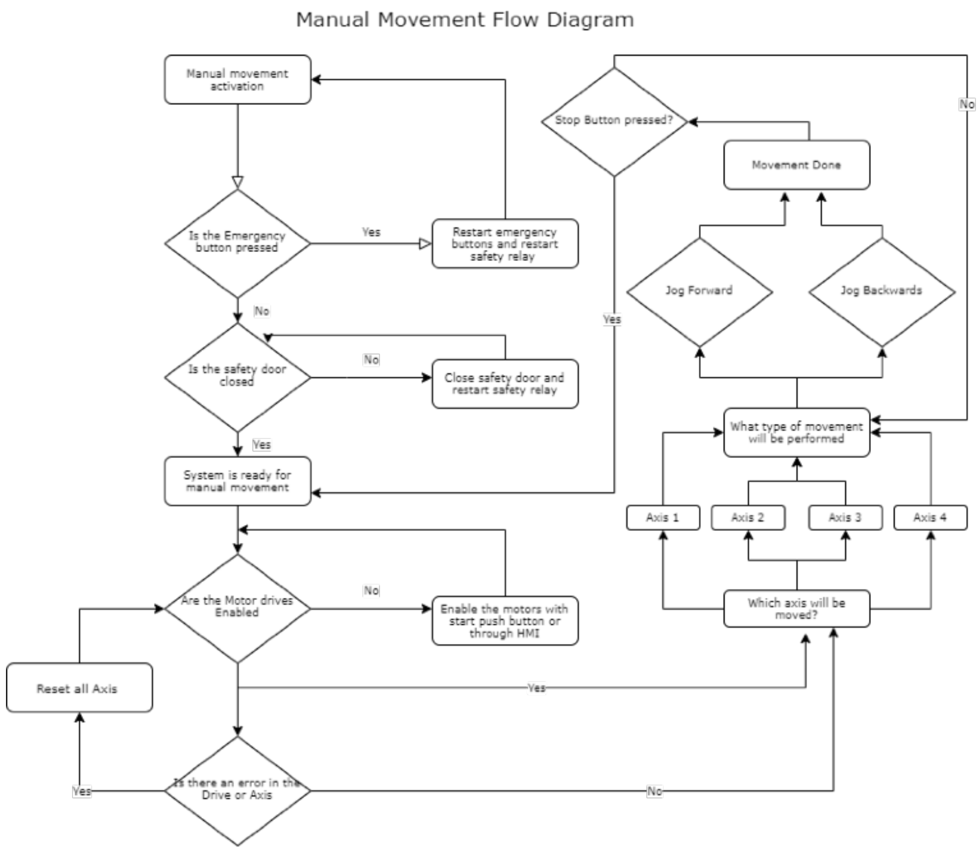

Manual Movement Flow Diagram
Manual movement activation
Is the Emergency button pressed
Yes
Restart emergency buttons and restart safety relay
No
Is the safety door closed
No
Close safety door and restart safety relay
Yes
System is ready for manual movement
Are the Motor drives Enabled
No
Enable the motors with start push button or through HMI
Reset all Axis
Yes
Is there an error in the Drive or Axis
Yes
No
Stop Button pressed?
No
Movement Done
Jog Forward
Jog Backwards
Yes
What type of movement will be performed
Axis 1
Axis 2
Axis 3
Axis 4
Which axis will be moved?

Figure 4.4.3. Flow diagram for Manual movement

The main idea of manual movement is to test the communication between each axis and each drive, as well as the communication between Twincat 3 and Festo drives. The full programming for manual movement can be seen in Appendice section.

4.5 Simple Automatic pick and place programming using Beckhoff CX2042 PLC

The automatic mode of the system will be a simple pick and place application, in which as the name states it will only need a single push of a button to start the sequence of movement. For this application the MC_MoveAbsolute function block was used and it is part of the motion package of Twincat 3. Everytime there is a step a state is changed in the system that works as a memory of the system, to avoid signals crashing and have a sequence movement. The following flow diagram shows the decisions and pre-requisites of the system for the automatic mode. Full programming of the automatic mode can be found in appendix section.

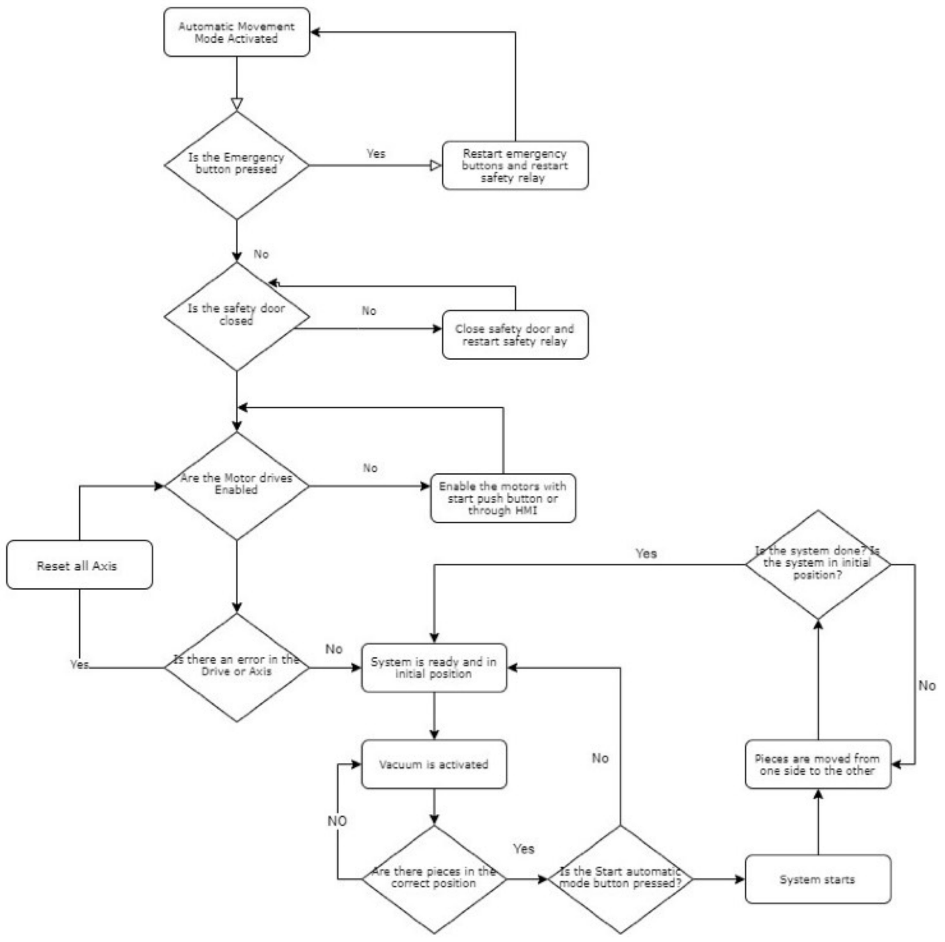

Automatic Movement Mode Activated
Is the Emergency button pressed
Yes
Restart emergency buttons and restart safety relay
No
Is the safety door closed
No
Close safety door and restart safety relay
Are the Motor drives Enabled
No
Enable the motors with start push button or through HMI
Reset all Axis
Yes
Is there an error in the Drive or Axis
No
System is ready and in initial position
Yes
Is the system done? Is the system in initial position?
No
Vacuum is activated
NO
Are there pieces in the correct position
Yes
Is the Start automatic mode button pressed?
No
System starts
Pieces are moved from one side to the other

Figure 4.4.4. Automatic mode, Pick and Place flowchart

4.6 Matlab/Simulink interface and integration

Twincat 3 automation engineering environment (XAE) made by Beckhoff has a very important advantage over other manufacturers software, this advantage is that is based on the widely used Microsoft Visual Studio and added to the applications that come with the engineering solution (motion, connection to ERPs, I/O configuration) this makes Twincat 3 a one of a kind software in the industry. Since is based on visual studio this approaxch allows the possibility to integrate other softwares and other programming tools, that are also based on visual studio such as MATLAB, simulink and C/C++, which are the most used programming softwares in todays industry.

The TC3 target (extension for using Matlab and simulink in Twincat) extends the workflows based on the simulink coder, giving twincat 3 the ability to target from simulink and be able to build them under TcCOM (TwinCat common object model). Allowing users to design their control algorithms and simulations models in simulink and use the simulink coder to compile the files, and later use this files in Twincat 3 software exactly the same way it is used in simulink. Another key feature of this target is that the model of block diagram can be integrated to TwinCat environment to be used as a controller simulation or debugging applications, meaning that it can be fully integrated to the environment.

Another feature is ADS communication in a MatLab script, ADS stands for Automation device specification and is a transport layer within Twincat environment, with this communication data can be exchanged directly to a matlab script or directly from a matlab scripts in real-time, this communication protocol is used only within the TwinCat environment and will be used always when there is a need to communicate with other software tools or to communicate with another PC or device.

Unfortunately for this thesis project the lack of physical sensors makes the full integration of the EXPT robot not so valuable, and just the simple actions to demonstrate the functionality and communication between each software can be shown. Another big issue is the Microsoft Visual Studio 2019 C++, since this version of visual studio is not supported by the link between the interfaces of simulink and Twincat. This is because matlab uses a compiler to generate the files that are readable in twincat 3 integration of the block diagram. Without the proper compiler it is impossible to generate files that are readable for twincat 3.

We will make a small guide on how to create the link between Twincat 3 and simulink, making the supposition that all the previous requirements are correct. To add the target for simulink the first step to be taken is to download TE1400 function target for simulink from the official Beckhoff site. Next is to open Matlab, it is important to mention that the latest version of Matlab which is recommended by Beckhoff is Matlab 2019a (oldest version is Matlab 2010b), other versions outside this range will not function correctly. To add the blocks of Twincat in Matlab you have to add the library from Twincat/functions/TF1400… and run the setup file in Matlab. Once this is added the simulink block diagram can be created. Some changes have to be done in the configuration parameters to create the Twincat target files.

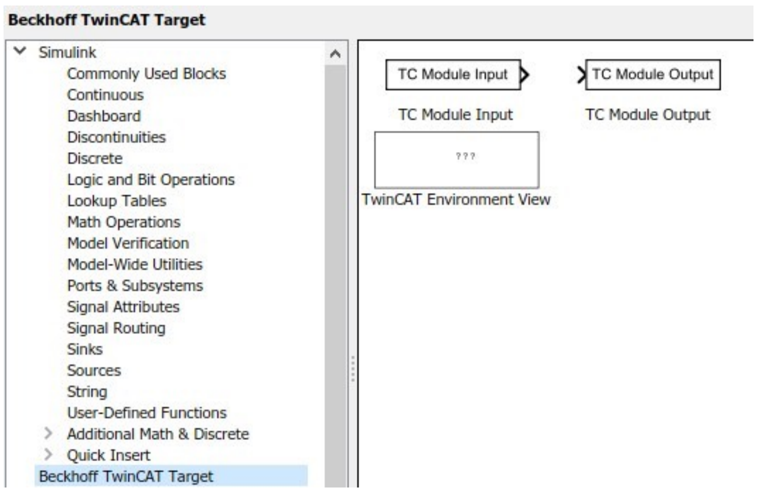
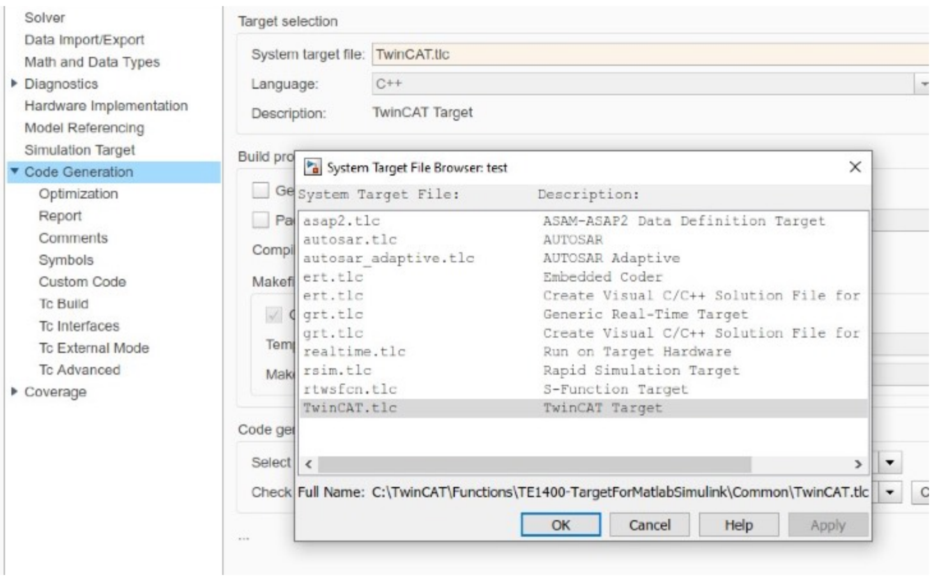


Figure 4.6.1. Adding the Twincat target blocks



Target selection

System target file: TwinCAT.tlc

Language: C++

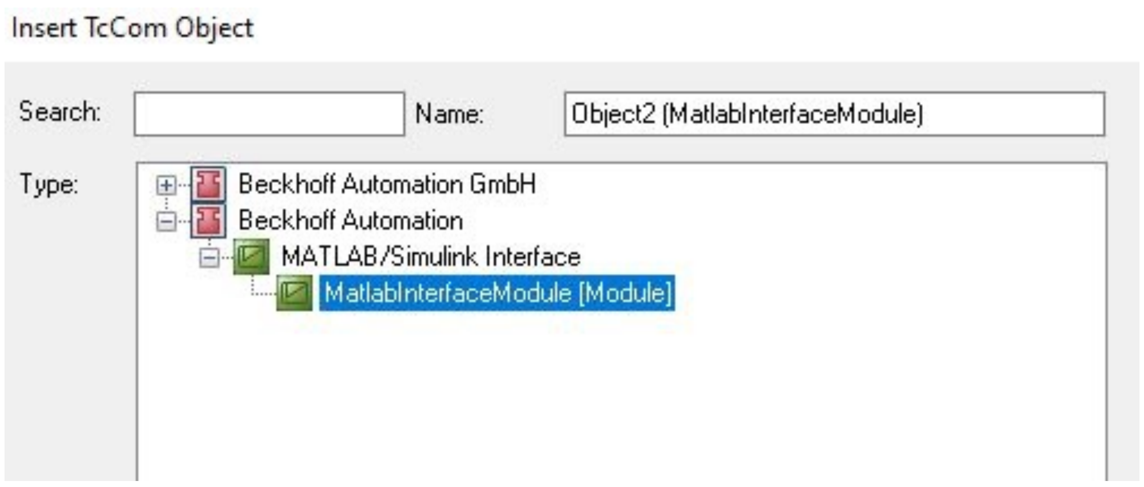
Description: TwinCAT Target

```
System Target File:      Description:
asap2.tlc                ASAM-ASAP2 Data Definition Target
autosar.tlc              AUTOSAR
autosar_adaptive.tlc     AUTOSAR Adaptive
ert.tlc                  Embedded Coder
ert.tlc                  Create Visual C/C++ Solution File for
grt.tlc                  Generic Real-Time Target
grt.tlc                  Create Visual C/C++ Solution File for
realtime.tlc             Run on Target Hardware
rsim.tlc                 Rapid Simulation Target
rtwsfcn.tlc              S-Function Target
TwinCAT.tlc              TwinCAT Target
```

Full Name: C:\TwinCAT\Functions\TE1400-TargetForMatlabSimulink\Common\TwinCAT.tlc

Figure 4.6.1. Changing the system target to generate files

Once the files are created we can use Twincat 3 XAE to add the block diagram that was previously created and link the variables accordingly. From this point on it is all part of programming like if we were using the variables and we can control the robot through simulink function block.


Insert TcCom Object

Search:

Name: Object2 (MatlabInterfaceModule)

Type:
- Beckhoff Automation GmbH
- Beckhoff Automation
  - MATLAB/Simulink Interface
    - MatlabInterfaceModule [Module]

Figure 4.6.3. Adding the block diagram to Twincat3

Unfortunately this is just a theoretical part of the function between matlab and twincat, without the correct compiler this procedure is not possible since there are no files that are generated for twincat 3 to read the files correctly.

5. FUTURE DEVELOPMENTS

Integration this machine with the extraordinary CX2042 PLC from Beckhoff is a major plus for the university, this machine was supposed to be working with this controller, but difficulties integrating different communication protocols and old components of the system made the integration long lasting and not possible for previous researchers. Now that the system is fully integrated there are an amazing amount of options to improve the system:
Integration of a camera and a Twincat vision modules: With this module the image processing applications will be possible and it will just add one more solution to the whole system. With this vision modules there is a significant reduction of engineering, since everything is simplified because configuration, parametrisation and programming takes place in a single software and that is familiar to engineers. Also all image processing applications can be integrated in real time, meaning that latency is eliminated, a major problem while applying image processing solutions. Integrating the vision modules will make the way for the EXPT robot to be more advanced and it will satisfy the demands of todays and tomorrows industry applications. Finally TwinCat vision will give a competitive advantage over other delta robots, will make that application of industry 4.0 look easy to integrate, will increase the production and safety, it will have openness and scalability on new solutions and of course there will be thousands of application in real time that can be programmed to comply with the demand of the industry.

- As mentioned before this machine lacks physical sensors and all the programming and calibration was made only by software limits, for example the carriages have a physical limits but there are limit switches programmed in the software to avoid any physical damage to the machine. Adding sensors to each rail and carriage will improve the effectiveness and precision of the EXPT robot. Calculating the inverse and forward kinematics of the delta robot and having sensors that will limit certain movements will make the robot be a high precision robot.

6. CONCLUSIONS

This thesis work showed that previous researches made with the motor drives and the Beckhoff controller trying to integrate all in a new system were wrong. The main objective of the thesis work was to fully integrate the Festo EXPT tripod robot with a new controller which after successfully integrating the motor drives to the PLC showed that the application was a success. The procedure used during this research and project can be used to integrate any type of hardware from other manufacturers and the steps taken into account for the project can be used from industries that are willing to invest a little bit to get an old machine have a controller that is sticking and evolving along with the current trends of the industry, for example industry 4.0.

The results after working with the controller showed that this type of applications can be used in any industry and the applications and solutions that can be achieved with the controller and with the universal solution of integrating old machines, phased out motor controllers, sensors and hardware from other manufacturers, are really immense and future developments are available once the whole system is integrated to the controller. One great advantage as mentioned several times is the software that this type of PLC is using, Twincat 3 is based in a widely popular and used microsoft visual studio and the develoPments with this interface are growing each day. All the packages and functions that Beckhoff can offer are industry changer and the CX2042 shows exactly the potential with how easy is to migrate all the components to the programmable logic controller.

The thesis project really showed the passion the author has with automation, even thought the project had a bad reputation and other researchers did try to integrate the system and were not successful, this time the steps taken into account and the experience from previous projects helped with the improvement of the system and with the successful steps for a general solution to integrate any machine. Of course it is important to mention that most of the time extra components will be necessary and taking into account the expertise from the manufacturer and official documentation is important to successfully carry out a project of this area.

7. SUMMARY

The thesis project had the main objective of being able to integrate the Beckhoff controller CX2042 into a machine from another manufacturer called EXPT tripod robot from FESTO with a solution that will not only work with this machine, but it will be able to follow the same steps to integrate any other machine that a company or individual might want as an approach. Integrating and migrating old machines into new controllers is something every industry is willing to discover since modern controllers are able to keep the pace with how the world is evolving and how the applications are trying to be in the same line as Industry 4.0. This thesis project was worked on several occasions in the past without success, this was mainly because the communication protocol some components use are different from the one the controller use and this was causing some problems while reading the parameters and data from the motor drives, in this specific occasion. The motor drives are old controllers that were phased out several years ago, causing some misinformation on how to integrate the components to the whole network. During this occasion, the motor drives were fully integrated into the network after some reading and tips from the manufacturers and after these controllers were in the network the solution was pretty straightforward to accomplish. The CX2042 controller works under the software TwinCat3 that is based on the highly popular visual studio from Microsoft, meaning that it is possible to integrate other softwares to twincat 3, making the software really flexible and easy to use. What is even more important is that software like C/C++ and Matlab can be integrated into Twincat 3 using an incorporated interface, this is possible thanks to the free functions packages of licenses available from Beckhoff, since this machine will be connected to a PC then licensing is not necessary.

During this thesis work the results were accomplished making a small guide on how to add the components to the network, it is important to know that there are some steps that have to be taken before integrating the system. These steps include actions that even from the design phase they have to be done, like knowing and understanding the topology, understanding the machine functionality and limits, creating the inputs and outputs list to know the hardware that will be needed. After the steps are understood then the integration can be accomplished without any problems. The major issues regarding this thesis were related to these steps, previous researchers had a bad I/O list and the topology of the system was never created.

Thanks to taking this thesis step by step the integration was a success, leaving the machine ready for more complicated and advanced applications, such as integrating the vision modules to have image processing in the system. The integration of this system made the author create a simple manual movement of the machine and an automatic pick a place simple sequence. This thesis will open the doors to future generations that want to work with this system and add applications that can help any industry.

As a conclusion the main objective was achieved and the machine is ready to be used once again, of course better and more applications could have been done but it was really important to understand how the system worked and how the components could be integrated to the system without any problem and be sure that the problems caused in the past were only bad steps taken towards the final approach. The author considers this thesis as a step forward to future generations and it will open the door to the mechatronics department with this machine, that has not been working for some time.

8. KOKKUVÕTE

Lõputöö peamine eesmärk oli Beckhoff CX2042 kontrolleri integreerimine FESTO tripod EXPT robotiga, kus väljapakutav lahendus mitte ainult töötaks konkreetse robotiga vaid mida oleks võimalik integreerida ka teiste seadmetega vastavalt ettevõtte või kliendi vajadustele. Vanemat tüüpi seadmete integreerimine ja üleviimine uute kontrolleritega kasutamiseks on kõigile ettevõtetele kasulik, kuna kaasaegsed kontrollerid käivad oma võimalustelt tehnika arenguga ja Industry 4.0 nõuete ning võimalustega kaasas. Selle töö teemal on varasemalt pakutud mitmeid mitte edukaid lahendusi, kuna mõningate komponentide kommunikatsiooniprotokoll on kontrolleri kommunikatsiooniprotokollist erinev. See probleem tekitas seni parameetrite ja andmete lugemisel teatud juhtudel vigu mootorite liidestamisel. Mootorite juhtimiseks kasutatakse vanemat tüüpi mootori kontrollereid ja see tekitab segadust, kuidas liidestada kõik komponendid ühtsesse võrku. Antud töö raames integreeriti mootori kontrollerid ühtsesse juhtvõrku komponendi tootjate soovituste abil ja saavutati küllaltki lihtne lahendus, kus kõik kontrollerid on ühtses võrkstruktuuris ühendatud. CX2042 kontroller töötab TwinCat3 tarkvaraga, mis baseerub väga populaarsele MS Visual Studio tarkvarale. See omakorda võimaldab liidestada TwinCat3 tarkvaraga teisi tarkvaralahendusi, mis muudab kogu süsteemi tõeliselt paindlikuks ja lihtsalt kasutatavaks. Veelgi tähtsam on, et C/C++ ja Matlab tarkvara lahendusi saab liidestada TwinCat3-ga, kasutades olemasolevat liidest. Selline liidestamine on võimalik tasuta litsentsiga Beckhoff firmalt.

Lõputöö käigus on valminud ka lühike juhend, kuidas liidestada komponente seadme võrkstruktuuriga. Siinjuures on oluline pidada meeles, et on rida samme, mis tuleb enne süsteemiga liidestamist läbi teha. Niisugusteks sammudeks on toimingud mis tuleks teha juba kavandamise faasis, näiteks kavandatava süsteemi topoloogia määratlemine ja sellest täieliku ülevaate tegemine, seadme funktsionaalsuse ja selle võimaluste määratlemine, sisendite ja väljundite nimekirja koostamine, et määratleda kogu vajalik riistvara. Kui kõik need sammud on läbitud, siis on võimalik kõigi komponentide edukas liidestamine ilma probleemideta. Lõputöö põhifookus on pööratud nende sammude läbiviimise kirjeldamisele. Varasemates analoogsetes töödes on olnud sisend/väljund loend ebatäpne ja topoloogiat ei ole täpselt määratletud..

Tänu samm-sammulisele lähenemisele selles töös oli ka kogu süsteemi liidestamine edukas ja saavutati seadme valmisolek keerukamateks ja väljakutset nõudvamateks ülesanneteks, milleks võiks olla masinnägemise mooduli integreerimine masinnägemise ülesannete lahendamiseks vahetult juhtsüsteemi abil. Töö autori poolt on loodud liidestamiste abil näitena lihtne seadme käsijuhtimise lahendus ja samuti automaatne tõsta ja aseta lihtne lahendus. Antud töö annab võimaluse järgnevateks projektideks selle seadmega, mis võiks pakkuda huvi tööstusele.

Töö kokkuvõttena on saavutatud töö püstitatud eesmärk, kogu seade koos uue juhtsüsteemiga on tööks valmis. Töö annab ülevaate ja juhised kogu süsteemi kasutamiseks edaspidi ja samuti juhised täiendavate komponentide liidestamiseks süsteemiga. Autor on seisukohal, et väljatöötatud lahendus on kasulik ka tulevastele õppuritele..

9. REFERENCES

[1] FESTO tripod Expt: Manual, FESTO, version 1.00

[2] Emerging Technologies: Connecting Millennials and Manufacturing, Joydeep Acharya, Yasutaka Serizawa, Sudhanshu Gaur, 2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)

[3] Industry 4.0: hope, hype or revolution? , Lorenzo Bassi, 2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI), 2017

[4] Plenary: The Challenges of Education in Engineering, Computing and Technology without exclusions: Innovation in the era of the Industrial Revolution 4.0., Claudio R. Brito, Melany M. Ciampi, Maria Feldgen, Osvaldo Clua, Victor A. Barros, IV IEEE World Engineering Education Conference - EDUNINE2020, 2019

[5] Industry 4.0: A review on industrial automation and robotic, June 2016
Jurnal Teknologi 78(6-13), Fauzi Othman, Universiti Teknologi Malaysia

[6] Fourth industrial revolution and its impact on society, Mario Peña-Cabrera, Victor Lomas, Gastón Lefranc, CHILECON 2019, Valparaiso, Chile, 2019

[7] Industrial Internet of Things: A Review, Avish Karmakar, Naiwrita Dey, Tapadyuti Baral, Manojeet Chowdhury, 2019 International Conference on Opto-Electronics and Applied Optics (Optronix), 2019

[8] "What do small and medium-sized enterprises stand to gain from the digitization of their production processes?", Interview with Hans Beckhoff on historical changes in automation technology, Martin Ciupek, PC Control, 2016

[9] C. Gong, "Human-Machine Interface: Design Principles of Visual Information in Human-Machine Interface Design ", International Conference on Intelligent Human -Machine Systems and Cybernetics Year: 2009 , Volume: 2.

[10] Some Security Problems and Aspects of the Industrial Internet of Things, Georgi Tsochev, Technical University of Sofia, 2020

[11] The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0, Martin Wollschlaeger, Thilo Sauter, Juergen Jasperneite, IEEE Industrial Electronics Magazine, 2017

[12] A. J. C. Trappey, C. V. Trappey, U. H. Govinda- rajan, J. J. Sun, and A. C. Chuang, "A review of technology standards and patent portfolios for enabling cyberphysical systems in advanced manufacturing," IEEE Access, Oct. 2016.

[13] Trends in Industrial Communication and OPC UA, Peter Drahoš, Erik Kuera, Oto Haffner, Ivan Klimo, Slovak University of Technology in Bratislava, 018 Cybernetics & Informatics (K&I), 2018, Lazy pod Makytou, Slovakia

[14] 25 years of PC history at Beckhoff, Andreas Thome, Beckhoff Automation, 2014

[15] High-speed handling tripod EXPT, Accurate dynamic response!, FESTO, Product short information, Festo AG & Co. KG, 2011

[16] Festo Tripod robot control with the Beckoff industrial embedded controller CX2042, Master thesis, Artur Pyatkov, Tallinn University of Technology, 2019

[17] CiA 302-1 Standard, CAN in Automation, version 4.1.0, 2009

[18] CiA 306, CAN in Automation, CAN in Automation, version 1.3.0, 2015

[19] CANopen Explained - A Simple Intro, CSS electronics, Soeren Frichs, 2020

[20] International Standard IEC 61131-3 part 3: Programming languages, Reference number IEC 61131-3:2003(E), International Electrotechnical commission, 2013

[21] Using industrial standards on PLC programming learning, Javier Molina, Julio Barbanacho, Carlos Leon, Control & Automation, 2017. MED '17.

[22] Next generation control units simplifying industrial machine learning, Stefano de Blasi, Elmar Engels, 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), 2020

[23] ROV Control System Upgrade, Benjamin Greenspon, OCEANS 2015 - MTS/IEEE Washington, 2015

[24] Integration of SCARA robot with Beckhoff industrial controller, Md Arifur Rahman, Taltech Master thesis projects, 2019

[25] Festo Parallel Kinematic system EXPT, FESTO, Manual, 2020/10 version.

[26] Beckhoff Documentation- EN- EL6751, Master and Slave Terminal for CanOpen, Beckhoff Automation, 2021-02-11 Ver. 3.7

[27] Beckhoff Information system - EN, Twincat 3 PLC Library : Tc2_MC2, MC_Power Function block, Beckhoff Automation, 2020

APPENDICES

Appendix A.1 Declaration of variables and function blocks

```
VAR_GLOBAL
        Axis1 : AXIS_REF;   //Axis one global declaration
        Axis2 : AXIS_REF;   //Axis two global declaration
        Axis3 : AXIS_REF;   //Axis three global declaration
        Axis4 : AXIS_REF;   //Axis four (rotatory) global declaration
END_VAR

PROGRAM MAIN
VAR
//Initial Buttons and Booleans//
bStartButton AT %I*:BOOL;
bStopButton AT %I*: BOOL;
bStart_1 AT %IX54.6 :BOOL ;
bAutoMode AT %I*: BOOL;
bManualMode AT %I*:BOOL;
bDoorSensor AT %I* : BOOL;
bStartScreen AT %I* :BOOL;
bCalibrateAxis AT %M*:BOOL;
bStartAuto AT %I* : BOOL;
bAir AT %I* : BOOL;
rState : LREAL;


//Vacuum DO for each Location
bTopVacuum AT %Q* :BOOL;
bBottomVacuum AT %Q* :BOOL;
bGrip1Vacuum AT %Q* :BOOL;
bGrip2Vacuum AT %Q* :BOOL;


// Memory Flags of correct enabling of Motors
bMotor1Enable AT %M*:BOOL;
```

```
bMotor2Enable AT %M*:BOOL;
bMotor3Enable AT %M*:BOOL;
bMotor4Enable AT %M*:BOOL;

//Booleans to know where the The pieces are located
bTop AT %I* :BOOL;
bBottom AT %I* :BOOL;
bLeft AT %I* :BOOL;
bRight AT %I* :BOOL;
bTopLeft AT %I* :BOOL;
bTopRight AT %I* :BOOL;
bBottomLeft AT %I* :BOOL;
bBottomRight AT %I* :BOOL;
bGripTop AT %I* :BOOL;
bGripBot AT %I*:BOOL;

//Status and Error for each Axis//
bErrorAxis1 AT %I*:  BOOL ;
bStatusAxis1 AT %I*: BOOL ;

bErrorAxis2 AT %I*:  BOOL ;
bStatusAxis2 AT %I*: BOOL ;

bErrorAxis3 AT %I*:  BOOL ;
bStatusAxis3 AT %I*: BOOL ;

bErrorAxis4 AT %I*:  BOOL ;
bStatusAxis4 AT %I*: BOOL ;

//Reset Button//
bResetButton AT %I*: BOOL;

//MC_Power for each Axis to enable each controller//
fbMC_PowerAxis1 : MC_Power ;
fbMC_PowerAxis2 : MC_Power ;
fbMC_PowerAxis3 : MC_Power ;
fbMC_PowerAxis4 : MC_Power ;
```

```
//MC Homing for Calibration of each Axis. Done after Manual and Auto mode
Change if system is not in initial Position
fbMC_HomeAxis1 : MC_Home ;
fbMC_HomeAxis2 : MC_Home ;
fbMC_HomeAxis3 : MC_Home ;
fbMC_HomeAxis4 : MC_Home ;

// Homing Function Block Activation buttons
bHomingExecuteAxis1 AT %I*: BOOL;
bHomingExecuteAxis2 AT %I*: BOOL;
bHomingExecuteAxis3 AT %I*: BOOL;
bHomingExecuteAxis4 AT %I*: BOOL;

//JOG Mode Function Block for each Axis//
fbMC_JogAxis1 : MC_Jog;
fbMC_JogAxis2 : MC_Jog;
fbMC_JogAxis3 : MC_Jog;
fbMC_JogAxis4 : MC_Jog;

// JOG Buttons for controlling each movement
bForwardAxis1 AT %I* : BOOL;
bForwardAxis2 AT %I* : BOOL;
bForwardAxis3 AT %I* : BOOL;
bForwardAxis4 AT %I* : BOOL;

bBackwardsAxis1 AT %I* : BOOL;
bBackwardsAxis2 AT %I* : BOOL;
bBackwardsAxis3 AT %I* : BOOL;
bBackwardsAxis4 AT %I* : BOOL;

//fb RESET for each axis
fbMC_ResetAxis1 : MC_Reset;
fbMC_ResetAxis2 : MC_Reset;
fbMC_ResetAxis3 : MC_Reset;
fbMC_ResetAxis4 : MC_Reset;

//Reset execute Booleans
bReset1 AT %I*:BOOL;
bReset2 AT %I*:BOOL;
```

```
bReset3 AT %I*:BOOL;
bReset4 AT %I*:BOOL;

//Function block for Read status of each Axis
fbMC_ReadStatus1 : Mc_ReadStatus;
fbMC_ReadStatus2 : Mc_ReadStatus;
fbMC_ReadStatus3 : Mc_ReadStatus;
fbMC_ReadStatus4 : Mc_ReadStatus;

//Reading Axis Actual position
fbMC_ReadActualPositionAxis1 : MC_ReadActualPosition;
fbMC_ReadActualPositionAxis2 : MC_ReadActualPosition;
fbMC_ReadActualPositionAxis3 : MC_ReadActualPosition;
fbMC_ReadActualPositionAxis4 : MC_ReadActualPosition;

//Function Blocks for absolute Movement Of the Axis (Known Locations in the
working space)
fbMC_MoveAxis1 : MC_MoveAbsolute;
fbMC_MoveAxis2 : MC_MoveAbsolute;
fbMC_MoveAxis3 : MC_MoveAbsolute;
fbMC_MoveAxis4 : MC_MoveAbsolute;
fbMC_MoveAxis5 : MC_MoveAbsolute;
fbMC_MoveAxis6 : MC_MoveAbsolute;
fbMC_MoveAxis7 : MC_MoveAbsolute;
fbMC_MoveAxis11 : MC_MoveAbsolute;
fbMC_MoveAxis22 : MC_MoveAbsolute;
fbMC_MoveAxis111 : MC_MoveAbsolute;
fbMC_MoveAxis222 : MC_MoveAbsolute;
fbMC_MoveAxis33 : MC_MoveAbsolute;
fbMC_MoveAxis1111 : MC_MoveAbsolute;
fbMC_MoveAxis2222 : MC_MoveAbsolute;
fbMC_MoveAxis333 : MC_MoveAbsolute;
fbMC_MoveAxis11111 : MC_MoveAbsolute;
fbMC_MoveAxis22222 : MC_MoveAbsolute;
fbMC_MoveAxis3333 : MC_MoveAbsolute;
fbMC_MoveAxis111111 : MC_MoveAbsolute;
fbMC_MoveAxis222222 : MC_MoveAbsolute;
fbMC_MoveAxis33333 : MC_MoveAbsolute;
fbMC_MoveAxis_1 : MC_MoveAbsolute;
```

```
fbMC_MoveAxis_2 : MC_MoveAbsolute;
fbMC_MoveAxis_3 : MC_MoveAbsolute;
fbMC_MoveAxis_1_1 : MC_MoveAbsolute;
fbMC_MoveAxis_2_2 : MC_MoveAbsolute;
fbMC_MoveAxis_3_3 : MC_MoveAbsolute;
fbMC_MoveAxis_4 : MC_MoveAbsolute;
fbMC_MoveAxis_1_1_1 : MC_MoveAbsolute;
fbMC_MoveAxis_2_2_2 : MC_MoveAbsolute;
fbMC_MoveAxis_3_3_3 : MC_MoveAbsolute;
fbMC_MoveAxis_1_2 : MC_MoveAbsolute;
fbMC_MoveAxis_2_3 : MC_MoveAbsolute;
fbMC_MoveAxis_3_4 : MC_MoveAbsolute;
fbMC_MoveAxis1_ : MC_MoveAbsolute;
fbMC_MoveAxis2_ : MC_MoveAbsolute;
fbMC_MoveAxis3_ : MC_MoveAbsolute;
fbMC_MoveAxis1_1 : MC_MoveAbsolute;
fbMC_MoveAxis2_2 : MC_MoveAbsolute;
fbMC_MoveAxis3_3 : MC_MoveAbsolute;
fbMC_MoveAxis_4_ : MC_MoveAbsolute;
fbMC_MoveAxis1_1_ : MC_MoveAbsolute;
fbMC_MoveAxis2_2_ : MC_MoveAbsolute;
fbMC_MoveAxis3_3_ : MC_MoveAbsolute;
fbMC_MoveAxis1_1_1 : MC_MoveAbsolute;
fbMC_MoveAxis2_2_2: MC_MoveAbsolute;
fbMC_MoveAxis3_3_3 : MC_MoveAbsolute;
fbMC_MoveAxis1_1_1_ : MC_MoveAbsolute;
fbMC_MoveAxis2_2_2_: MC_MoveAbsolute;
fbMC_MoveAxis3_3_3_ : MC_MoveAbsolute;
fbMC_MoveAxis_4_4 : MC_MoveAbsolute;

//MC relative test
fbMC_Relativemove1 : MC_MoveRelative;
fbMC_Relativemove2 : MC_MoveRelative;
fbMC_Relativemove3 : MC_MoveRelative;

//Error Flags to control Every axis
bError AT %I* :BOOL;

fbMC_StopAxis1 : MC_Stop;
```

```
Timer1 : TON;
Timer2:TON;
Timer3:TON;
Timer4:TON;

END_VAR
```

Appendix A.2 Manual movement programming

```
//Initial Flags and Buttons: Emergency Button, Door Open and Air pressure



//Enabling of all drivers, activated only if Emergency reset is not activated
// And Door is closed. (This is hardware security)
IF bStartButton =1 THEN
        rState:=0000;

END_IF

IF (bStartButton = 1 OR bStartScreen) OR fbMC_PowerAxis1.Enable AND NOT
bStopButton THEN
        fbMC_PowerAxis1(Axis := GVL.Axis1,
                                        Enable := TRUE,
                                        Enable_Positive := TRUE,
                                        Enable_Negative := TRUE);

        fbMC_PowerAxis2(Axis := GVL.Axis2,
                                        Enable := TRUE,
                                        Enable_Positive := TRUE,
                                        Enable_Negative := TRUE);

        fbMC_PowerAxis3(Axis := GVL.Axis3,
                                        Enable := TRUE,
                                        Enable_Positive := TRUE,
                                        Enable_Negative := TRUE);

        fbMC_PowerAxis4(Axis := GVL.Axis4,
                                        Enable := TRUE,
                                        Enable_Positive := TRUE,
                                        Enable_Negative := TRUE);



        fbMC_ResetAxis1(Axis := GVL.Axis1);

ELSE
        fbMC_PowerAxis1(Axis :=GVL.Axis1,
                //Motors are disabled if button Stop is pressed or Emergency push
button is pressed or
                                        Enable := FALSE,
                // Door is opened (Hardware security)
                                        Enable_Positive := FALSE,
                                        Enable_Negative := FALSE);

        fbMC_PowerAxis2(Axis :=GVL.Axis2,
                                        Enable := FALSE,
                                        Enable_Positive := FALSE,
                                        Enable_Negative := FALSE);

        fbMC_PowerAxis3(Axis :=GVL.Axis3,
                                        Enable := FALSE,
                                        Enable_Positive := FALSE,
                                        Enable_Negative := FALSE);

        fbMC_PowerAxis4(Axis :=GVL.Axis4,
                                        Enable := FALSE,
                                        Enable_Positive := FALSE,
                                        Enable_Negative := FALSE);

END_IF

IF fbMC_PowerAxis1.Status =1 THEN
                // Motor/ Driver 1 is activated
        bMotor1Enable := 1;
ELSE
        bMotor1Enable := 0;
```

```
END_IF

IF fbMC_PowerAxis2.Status =1 THEN
                // Motor/ Driver 2 is activated
        bMotor2Enable := 1;
ELSE
        bMotor2Enable := 0;
END_IF

IF fbMC_PowerAxis3.Status =1 THEN
                // Motor/ Driver 3 is activated
        bMotor3Enable := 1;
ELSE
        bMotor3Enable := 0;

END_IF

IF fbMC_PowerAxis4.Status =1 THEN
                // Motor/ Driver 4 is activated
        bMotor4Enable := 1;
ELSE
        bMotor4Enable := 0;

END_IF

//Enabling the read status function block to read the actual status and error in an
axis
IF bStartButton = 1 THEN

        fbMC_ReadStatus1(Axis:= GVL.Axis1,
                                        Enable:=TRUE);
ELSE
        IF bStopButton = 1 THEN
                fbMC_ReadStatus1(Axis:= GVL.Axis1,
                                        Enable:=FALSE);
        END_IF
END_IF
```

```
IF bStartButton = 1 THEN
        fbMC_ReadStatus2(Axis:= GVL.Axis2,
                                        Enable:=TRUE);
ELSE
        IF bStopButton = 1 THEN
                fbMC_ReadStatus2(Axis:= GVL.Axis2,
                                        Enable:=FALSE);
        END_IF
END_IF
IF bStartButton = 1 THEN
        fbMC_ReadStatus3(Axis:= GVL.Axis3,
                                        Enable:=TRUE);
ELSE
        IF bStopButton = 1 THEN
                fbMC_ReadStatus3(Axis:= GVL.Axis3,
                                        Enable:=FALSE);
        END_IF
END_IF
IF bStartButton = 1 THEN
        fbMC_ReadStatus4(Axis:= GVL.Axis4,
                                        Enable:=TRUE);
ELSE
        IF bStopButton = 1 THEN
                fbMC_ReadStatus4(Axis:= GVL.Axis4,
                                        Enable:=FALSE);
        END_IF
END_IF
//Enabling the read actual position function block
IF bMotor1Enable = 1 THEN

        fbMC_ReadActualPositionAxis1(Axis:= GVL.Axis1,
                                        Enable:=TRUE);
ELSE
        IF bStopButton = 1 THEN
                fbMC_ReadActualPositionAxis1(Axis:= GVL.Axis1,
                                        Enable:=FALSE);
        END_IF
END_IF
IF bMotor2Enable = 1 THEN
```

```
        fbMC_ReadActualPositionAxis2(Axis:= GVL.Axis2,
                                        Enable:=TRUE);
ELSE
        IF bStopButton = 1 THEN
                fbMC_ReadActualPositionAxis2(Axis:= GVL.Axis2,
                                        Enable:=FALSE);
        END_IF
END_IF
IF bMotor3Enable = 1 THEN
        fbMC_ReadActualPositionAxis3(Axis:= GVL.Axis3,
                                        Enable:=TRUE);
ELSE
        IF bStopButton = 1 THEN
                fbMC_ReadActualPositionAxis3(Axis:= GVL.Axis3,
                                        Enable:=FALSE);
        END_IF
END_IF
IF bMotor4Enable = 1 THEN

        fbMC_ReadActualPositionAxis4(Axis:= GVL.Axis4,
                                        Enable:=TRUE);
ELSE
        IF bStopButton = 1 THEN
                fbMC_ReadActualPositionAxis4(Axis:= GVL.Axis4,
                                        Enable:=FALSE);
        END_IF
END_IF
//Reset For all axis in case there is an Error. This will be present in all screens
IF bReset1=1 THEN
        fbMC_ResetAxis1(Axis := GVL.Axis1,
                                        Execute := TRUE);
ELSE
        fbMC_ResetAxis1(Axis := GVL.Axis1,
                                        Execute := FALSE);
END_IF
IF bReset1=1 THEN
        fbMC_ResetAxis2(Axis := GVL.Axis2,
                                        Execute := TRUE);
```

```
ELSE
        fbMC_ResetAxis2(Axis := GVL.Axis2,
                                        Execute := FALSE);
END_IF
IF  bReset1=1 THEN
        fbMC_ResetAxis3(Axis := GVL.Axis3,
                                        Execute := TRUE);
ELSE
        fbMC_ResetAxis3(Axis := GVL.Axis3,
                                        Execute := FALSE);
END_IF

IF bReset1=1 THEN


        fbMC_ResetAxis4(Axis := GVL.Axis4,
                                        Execute := TRUE);


ELSE
        fbMC_ResetAxis4(Axis := GVL.Axis4,
                                        Execute := FALSE);
END_IF

// Manual Movement with the Function Block MC_JOg for each axis

IF fbMC_PowerAxis1.Enable =1 THEN //Manual Activation


        fbMC_JogAxis1(Axis := GVL.Axis1,
                JogForward := bForwardAxis1,
                JogBackwards:= bBackwardsAxis1,
                Mode:= MC_JOGMODE_STANDARD_SLOW);

END_IF

IF fbMC_PowerAxis2.Enable =1 THEN //Manual Activation


        fbMC_JogAxis2(Axis := GVL.Axis2,
                JogForward := bForwardAxis2,
                JogBackwards:= bBackwardsAxis2,
                Mode:= MC_JOGMODE_STANDARD_SLOW);

END_IF

IF fbMC_PowerAxis3.Enable =1 THEN //Manual Activation

        fbMC_JogAxis3(Axis := GVL.Axis3,
                JogForward := bForwardAxis3,
                JogBackwards:= bBackwardsAxis3,
                Mode:= MC_JOGMODE_STANDARD_SLOW);

END_IF

IF fbMC_PowerAxis4.Enable =1 THEN //Manual Activation

        fbMC_JogAxis4(Axis := GVL.Axis4,
                JogForward := bForwardAxis4,
                JogBackwards:= bBackwardsAxis4,
                Mode:= MC_JOGMODE_STANDARD_SLOW);

END_IF
```

Appendix A.3 Automatic mode programming

```
//Reset For all axis in case there is an Error. This will be present in all screens
IF bReset1=1 THEN
        fbMC_ResetAxis1(Axis := GVL.Axis1,
                                        Execute := TRUE);
ELSE
        fbMC_ResetAxis1(Axis := GVL.Axis1,
                                        Execute := FALSE);
END_IF
IF bReset1=1 THEN
        fbMC_ResetAxis2(Axis := GVL.Axis2,
                                        Execute := TRUE);
ELSE
        fbMC_ResetAxis2(Axis := GVL.Axis2,
                                        Execute := FALSE);
END_IF
IF  bReset1=1 THEN
        fbMC_ResetAxis3(Axis := GVL.Axis3,
                                        Execute := TRUE);
ELSE
        fbMC_ResetAxis3(Axis := GVL.Axis3,
                                        Execute := FALSE);
END_IF
IF bReset1=1 THEN
        fbMC_ResetAxis4(Axis := GVL.Axis4,
                                        Execute := TRUE);
ELSE
        fbMC_ResetAxis4(Axis := GVL.Axis4,
                                        Execute := FALSE);
END_IF
// Manual Movement with the Function Block MC_JOg for each axis
IF fbMC_PowerAxis1.Enable =1 THEN //Manual Activation
        fbMC_JogAxis1(Axis := GVL.Axis1,
                JogForward := bForwardAxis1,
                JogBackwards:= bBackwardsAxis1,
                Mode:= MC_JOGMODE_STANDARD_SLOW);
        END_IF
```

```
IF fbMC_PowerAxis2.Enable =1 THEN //Manual Activation
        fbMC_JogAxis2(Axis := GVL.Axis2,
                JogForward := bForwardAxis2,
                JogBackwards:= bBackwardsAxis2,
                Mode:= MC_JOGMODE_STANDARD_SLOW);
END_IF
IF fbMC_PowerAxis3.Enable =1 THEN //Manual Activation
        fbMC_JogAxis3(Axis := GVL.Axis3,
                JogForward := bForwardAxis3,
                JogBackwards:= bBackwardsAxis3,
                Mode:= MC_JOGMODE_STANDARD_SLOW);
END_IF
IF fbMC_PowerAxis4.Enable =1 THEN //Manual Activation
        fbMC_JogAxis4(Axis := GVL.Axis4,
                JogForward := bForwardAxis4,
                JogBackwards:= bBackwardsAxis4,
                Mode:= MC_JOGMODE_STANDARD_SLOW);
END_IF
// Absolute Movement for Automatic mode
IF bMotor1Enable =1 AND bMotor2Enable= 1 AND bMotor3Enable= 1 AND
bMotor4Enable =1 AND bAutoMode =1 AND rState=0000 THEN
    fbMC_MoveAxis1(Axis := GVL.Axis1,
                 Execute := TRUE,
                 Position := 0,
                 Velocity:= 4000);

                 fbMC_MoveAxis2(Axis := GVL.Axis2,
                 Execute := TRUE,
                 Position := 5,
                 Velocity:= 4000);

                 fbMC_MoveAxis3(Axis := GVL.Axis3,
                 Execute := TRUE,
                 Position := -2,
                 Velocity:= 4000);

                 fbMC_MoveAxis4(Axis := GVL.Axis4,
                 Execute := TRUE,
                 Position := 0.0,
                    Velocity:= 2);
                    IF fbMC_MoveAxis3.Done =1 THEN
                    bTopVacuum:=1;
                    bBottomVacuum:=1;
                    rState:=1001;
                    END_IF
END_IF
// States For sequences and combinations
//State 100X Corresponds to only Top Vacuum activated if not then is another state
another combination
IF bTop = 1 AND bDoorSensor = 0   AND bAir = 0 AND bLeft=0 AND bBottom=0
AND bTopLeft=0 AND bTopRight=0 AND bBottomLeft=0 AND rState=1001 AND
bStartAuto=1 THEN     // Auto mode  Also
bBottomVacuum:=0;
 fbMC_MoveAxis1(Axis := GVL.Axis1,
                    Execute := FALSE,
                    Position := 0,
                    Velocity:= 4000);

                    fbMC_MoveAxis2(Axis := GVL.Axis2,
                    Execute := FALSE,
                    Position := 5,
                    Velocity:= 4000);

                    fbMC_MoveAxis3(Axis := GVL.Axis3,
                    Execute := FALSE,
                    Position := -2,
                    Velocity:= 4000);

                    fbMC_MoveAxis4(Axis := GVL.Axis4,
                    Execute := FALSE,
                    Position := 0,
                    Velocity:= 2);

rState := 2001;
END_IF
//Programming for each state
// State 1001 Top almost Put center
IF rState= 2001 THEN
```

```
   fbMC_MoveAxis1(Axis := GVL.Axis1,
                  Execute := TRUE,
                  Position := -80.77,
                  Velocity:= 4000);

                  fbMC_MoveAxis2(Axis := GVL.Axis2,
                  Execute := TRUE,
                  Position := 26.5,
                  Velocity:= 4000);

                  fbMC_MoveAxis3(Axis := GVL.Axis3,
                  Execute := TRUE,
                  Position := -176,
                  Velocity:= 4000);
IF fbMC_MoveAxis3.Done =1 AND fbMC_ReadActualPositionAxis3.Position <= -175
THEN
rState:=2010;
END_IF
END_IF
IF rState=2010 THEN
  fbMC_MoveAxis1(Axis := GVL.Axis1,
                  Execute := FALSE,
                  Position := -80.77,
                  Velocity:= 4000);

                  fbMC_MoveAxis2(Axis := GVL.Axis2,
                  Execute := FALSE,
                  Position := 26.5,
                  Velocity:= 4000);

                  fbMC_MoveAxis3(Axis := GVL.Axis3,
                  Execute := FALSE,
                  Position := -176,
                  Velocity:= 4000);
rState:= 2011;
END_IF
//Center Top putting piece
IF rState=2011 THEN
```

```
fbMC_MoveAxis1(Axis := GVL.Axis1,
          Execute := TRUE,
          Position := -97.35,
          Velocity:= 4000);

          fbMC_MoveAxis2(Axis := GVL.Axis2,
          Execute := TRUE,
          Position := 46.20,
          Velocity:= 4000);

          fbMC_MoveAxis3(Axis := GVL.Axis3,
          Execute := TRUE,
          Position := -192.5,
          Velocity:= 4000);

IF fbMC_MoveAxis3.Done =1 AND fbMC_ReadActualPositionAxis3.Position <= -191
THEN
rState:=2030;
END_IF
END_IF
IF rState=2030 THEN
 fbMC_MoveAxis1(Axis := GVL.Axis1,
          Execute := FALSE,
          Position := -97.35,
          Velocity:= 4000);

          fbMC_MoveAxis2(Axis := GVL.Axis2,
          Execute := FALSE,
          Position := 46.20,
          Velocity:= 4000);

          fbMC_MoveAxis3(Axis := GVL.Axis3,
          Execute := FALSE,
          Position := -192.5,
          Velocity:= 4000);
bTopVacuum:=0;
bGrip2Vacuum:=1;
rState:=2040;
```

```
END_IF
IF rState =2040 THEN\
   fbMC_MoveAxis1(Axis := GVL.Axis1,
                  Execute := TRUE,
                  Position := -1,
                  Velocity:= 4000);

                  fbMC_MoveAxis2(Axis := GVL.Axis2,
                  Execute := TRUE,
                  Position := 5,
                  Velocity:= 4000);

                  fbMC_MoveAxis3(Axis := GVL.Axis3,
                  Execute := TRUE,
                  Position := -3,
                  Velocity:= 4000);

IF  fbMC_MoveAxis3.Done  =1  AND  fbMC_ReadActualPositionAxis3.Position  >  -3.5
THEN
                            rState:=2051;

                  END_IF
  END_IF
IF rState=2051 THEN
  fbMC_MoveAxis1(Axis := GVL.Axis1,
                  Execute := FALSE,
                  Position := -1,
                  Velocity:= 4000);

                  fbMC_MoveAxis2(Axis := GVL.Axis2,
                  Execute := FALSE,
                  Position := 5,
                  Velocity:= 4000);

                  fbMC_MoveAxis3(Axis := GVL.Axis3,
                  Execute := FALSE,
                  Position := -3,
                  Velocity:= 4000);
rState:=2060;
```

```
END_IF
//Rotating Axis 4 to change gripper Vacuum
IF rState=2060 THEN

                fbMC_MoveAxis4(Axis := GVL.Axis4,
                Execute := TRUE,
                Position := 0.50,
                Velocity:= 2);

IF fbMC_MoveAxis4.Position > 0.48 THEN
rState:=2061 ;
END_IF
END_IF
IF rState=2061 THEN

                fbMC_MoveAxis4(Axis := GVL.Axis4,
                Execute := FALSE,
                Position := 0.50,
                Velocity:= 2);
rState:=2070;
END_IF
IF rState= 2070 THEN
  fbMC_MoveAxis1(Axis := GVL.Axis1,
                Execute := TRUE,
                Position := -58.35,
                Velocity:= 4000);

                fbMC_MoveAxis2(Axis := GVL.Axis2,
                Execute := TRUE,
                Position := 46.32,
                Velocity:= 4000);

                fbMC_MoveAxis3(Axis := GVL.Axis3,
                Execute := TRUE,
                Position := -187.5,
                Velocity:= 4000);
IF fbMC_ReadActualPositionAxis3.Position < -187 THEN
rState:=2071;
```

```
END_IF
END_IF

IF rState=2072 THEN
  fbMC_MoveAxis1(Axis := GVL.Axis1,
                    Execute := FALSE,
                    Position := -58.35,
                    Velocity:= 4000);

                    fbMC_MoveAxis2(Axis := GVL.Axis2,
                    Execute := FALSE,
                    Position := 46.32,
                    Velocity:= 4000);

                    fbMC_MoveAxis3(Axis := GVL.Axis3,
                    Execute := FALSE,
                    Position := -3,
                    Velocity:= 4000);

rState:=2079;
END_IF

IF rState= 2079 THEN

   fbMC_MoveAxis1(Axis := GVL.Axis1,
                    Execute := TRUE,
                    Position := -75.35,
                    Velocity:= 4000);

                    fbMC_MoveAxis2(Axis := GVL.Axis2,
                    Execute := TRUE,
                    Position := 64.74,
                    Velocity:= 4000);

                    fbMC_MoveAxis5(Axis := GVL.Axis3,
                    Execute := TRUE,
                    Position := -201,
                    Velocity:= 4000);
```

```
IF fbMC_ReadActualPositionAxis3.Position < -200 THEN
bGrip1Vacuum:=1;
rState:=2073;
END_IF
END_IF
IF rState =2074 THEN
  fbMC_MoveAxis11(Axis := GVL.Axis1,
               Execute := TRUE,
               Position := -1,
               Velocity:= 4000);

               fbMC_MoveAxis22(Axis := GVL.Axis2,
               Execute := TRUE,
               Position := 5,
               Velocity:= 4000);

               fbMC_MoveAxis6(Axis := GVL.Axis3,
               Execute := TRUE,
               Position := -2,
               Velocity:= 4000);

               IF fbMC_ReadActualPositionAxis3.Position >= -3 THEN
                       rState:=2075;
//Putting piece a bit up of bottom center
IF rState =2080 THEN

  fbMC_MoveAxis111(Axis := GVL.Axis1,
               Execute := TRUE,
               Position := -85.2,
               Velocity:= 4000);

               fbMC_MoveAxis222(Axis := GVL.Axis2,
               Execute := TRUE,
               Position := 188.7,
               Velocity:= 4000);

               fbMC_MoveAxis33(Axis := GVL.Axis3,
               Execute := TRUE,
               Position := -33.9,
                    Velocity:= 4000);
                    IF fbMC_ReadActualPositionAxis3.Position < -33.8 THEN
                              rState:=2081;
                    END_IF
 END_IF
//Putting Piece in bottom Center
 IF rState =2082 THEN

  fbMC_MoveAxis1111(Axis := GVL.Axis1,
                    Execute := TRUE,
                    Position := -95,
                    Velocity:= 4000);

                    fbMC_MoveAxis2222(Axis := GVL.Axis2,
                    Execute := TRUE,
                    Position := 196.7,
                    Velocity:= 4000);

                    fbMC_MoveAxis333(Axis := GVL.Axis3,
                    Execute := TRUE,
                    Position := -47.2,
                    Velocity:= 4000);

                    IF fbMC_ReadActualPositionAxis3.Position < -47.1 THEN
                              rState:=2084;
                              bBottomVacuum:=1;
                              bGrip2Vacuum:=0;

                    END_IF
 END_IF
  IF rState =2084 THEN
  fbMC_MoveAxis1111(Axis := GVL.Axis1,
                    Execute := FALSE,
                    Position := -91,
                    Velocity:= 4000);

                    fbMC_MoveAxis2222(Axis := GVL.Axis2,
                    Execute := FALSE,
                    Position := 195.86,
                    Velocity:= 4000);

                    fbMC_MoveAxis333(Axis := GVL.Axis3,
                    Execute := FALSE,
                    Position := -47.13,
                    Velocity:= 4000);
   rState:=2085;
   END_IF

   //Returning to Initial position
   IF rState =2085 THEN

      fbMC_MoveAxis11111(Axis := GVL.Axis1,
                    Execute := TRUE,
                    Position := -1,
                    Velocity:= 4000);

                    fbMC_MoveAxis22222(Axis := GVL.Axis2,
                    Execute := TRUE,
                    Position := 5,
                    Velocity:= 4000);

                    fbMC_MoveAxis3333(Axis := GVL.Axis3,
                    Execute := TRUE,
                    Position := -2,
                    Velocity:= 4000);

                    IF fbMC_ReadActualPositionAxis3.Position >= -3 THEN
                              rState:=2086;

   END_IF
   END_IF
   IF rState=2086 THEN



 fbMC_MoveAxis11111(Axis := GVL.Axis1,
                    Execute := FALSE,
                    Position := -1,
                    Velocity:= 4000);
```

```
                    fbMC_MoveAxis22222(Axis := GVL.Axis2,
                    Execute := FALSE,
                    Position := 5,
                    Velocity:= 4000);

                    fbMC_MoveAxis3333(Axis := GVL.Axis3,
                    Execute := FALSE,
                    Position := -2,
                    Velocity:= 4000);
rState:=2087;
                    END_IF

 //Rotating Axis 4 to change gripper Vacuum
IF rState=2087 THEN

                    fbMC_MoveAxis_4(Axis := GVL.Axis4,
                    Execute := TRUE,
                    Position := 0,
                    Velocity:= 2);

IF fbMC_MoveAxis_4.Position < 0.1 THEN
rState:=2088 ;
END_IF

END_IF

IF rState=2088 THEN

                    fbMC_MoveAxis_4(Axis := GVL.Axis4,
                    Execute := FALSE,
                    Position := 0.50,
                    Velocity:= 2);
rState:=2090;
END_IF

//Putting piece a bit up of bottom right
IF rState =2090 THEN
```

```
  fbMC_MoveAxis_1(Axis := GVL.Axis1,
                Execute := TRUE,
                Position := -24.3,
                Velocity:= 4000);

                fbMC_MoveAxis_2(Axis := GVL.Axis2,
                Execute := TRUE,
                Position := 170.5,
                Velocity:= 4000);

                fbMC_MoveAxis_3(Axis := GVL.Axis3,
                Execute := TRUE,
                Position := -14.7,
                Velocity:= 4000);
Timer1(IN:=TRUE,
        PT:=T#4S);
                 IF fbMC_ReadActualPositionAxis3.Position < -14.6 AND
fbMC_MoveAxis_3.Done = 1 AND Timer1.Q=1 THEN
                        rState:=2091;

                END_IF
 END_IF

 IF rState =2091 THEN

  fbMC_MoveAxis_1(Axis := GVL.Axis1,
                Execute := FALSE,
                Position := -70.21,
                Velocity:= 4000);

                fbMC_MoveAxis_2(Axis := GVL.Axis2,
                Execute := FALSE,
                Position := 178.14,
                Velocity:= 4000);

                fbMC_MoveAxis_3(Axis := GVL.Axis3,
                Execute := FALSE,
                Position := -22.67,
                Velocity:= 4000);
```

```
                        rState:=2092;

END_IF

//Putting Piece in bottom Right
IF rState =2092 THEN

 fbMC_MoveAxis_1_1(Axis := GVL.Axis1,
                Execute := TRUE,
                Position := -68.7,
                Velocity:= 4000);

                fbMC_MoveAxis_2_2(Axis := GVL.Axis2,
                Execute := TRUE,
                Position := 207.2,
                Velocity:= 4000);

                fbMC_MoveAxis_3_3(Axis := GVL.Axis3,
                Execute := TRUE,
                Position := -66.5,
                Velocity:= 4000);

                IF fbMC_ReadActualPositionAxis3.Position < -66.4 THEN
                        rState:=2093;

                        bGrip1Vacuum:=0;
                END_IF
END_IF

 IF rState =2093 THEN

 fbMC_MoveAxis_1_1(Axis := GVL.Axis1,
                Execute := FALSE,
                Position := -91,
                Velocity:= 4000);

                fbMC_MoveAxis_2_2(Axis := GVL.Axis2,
                Execute := FALSE,
                  Position := 195.86,
                  Velocity:= 4000);

                  fbMC_MoveAxis_3_3(Axis := GVL.Axis3,
                  Execute := FALSE,
                  Position := -47.13,
                  Velocity:= 4000);
rState:=2094;
END_IF

//Returning to Initial position
IF rState =2094 THEN

   fbMC_MoveAxis_1_1_1(Axis := GVL.Axis1,
                  Execute := TRUE,
                  Position := -1,
                  Velocity:= 4000);

                  fbMC_MoveAxis_2_2_2(Axis := GVL.Axis2,
                  Execute := TRUE,
                  Position := 5,
                  Velocity:= 4000);

                  fbMC_MoveAxis_3_3_3(Axis := GVL.Axis3,
                  Execute := TRUE,
                  Position := -2,
                  Velocity:= 4000);

                  IF fbMC_ReadActualPositionAxis3.Position >= -3 THEN
                          rState:=2096;

END_IF
END_IF
IF rState=2096 THEN
  fbMC_MoveAxis_1_1_1(Axis := GVL.Axis1,
                  Execute := FALSE,
                  Position := -1,
                  Velocity:= 4000);
```

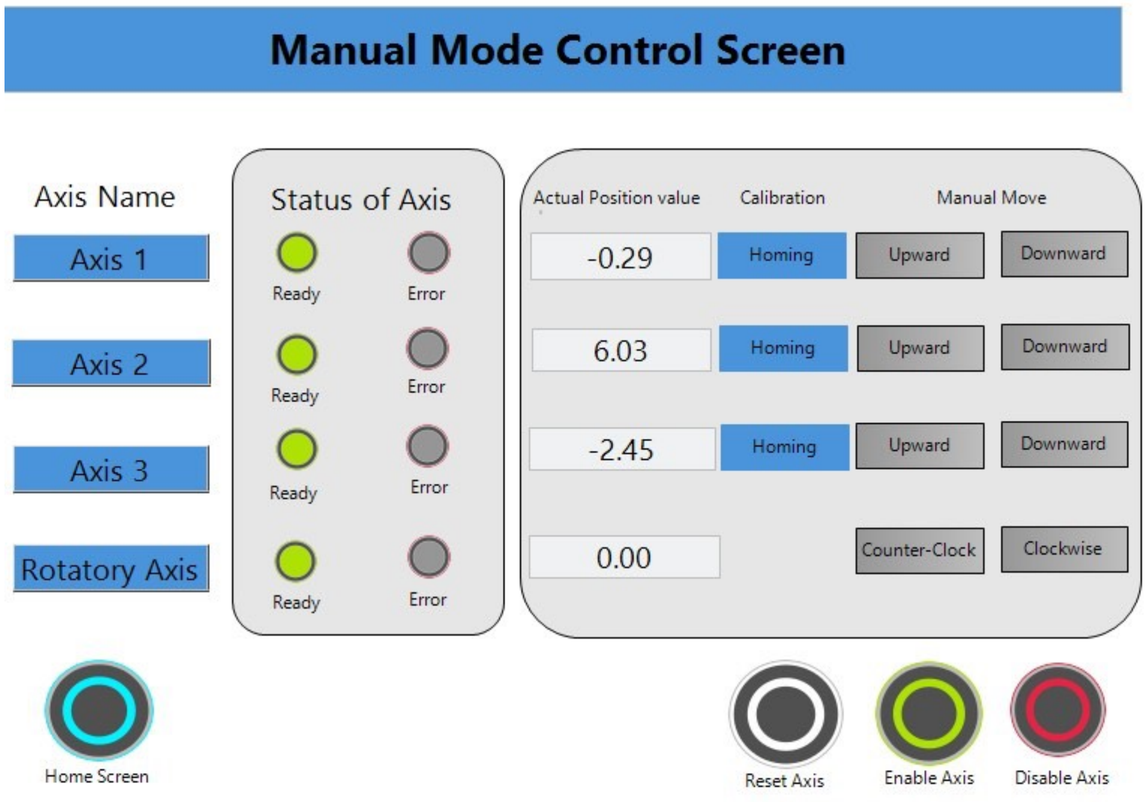
```
                 fbMC_MoveAxis_2_2_2(Axis := GVL.Axis2,
                 Execute := FALSE,
                 Position := 5,
                 Velocity:= 4000);

                 fbMC_MoveAxis_3_3_3(Axis := GVL.Axis3,
                 Execute := FALSE,
                 Position := -2,
                 Velocity:= 4000);
rState:=2097;
                 END_IF

 //Putting piece a bit up top left
IF rState =2097 THEN

  fbMC_MoveAxis_1_2(Axis := GVL.Axis1,
                 Execute := TRUE,
                 Position := -103.5,
                 Velocity:= 4000);

                 fbMC_MoveAxis_2_3(Axis := GVL.Axis2,
                 Execute := TRUE,
                 Position := 11.15,
                 Velocity:= 4000);

                 fbMC_MoveAxis_3_4(Axis := GVL.Axis3,
                 Execute := TRUE,
                 Position := -164.36,
                 Velocity:= 4000);

                 IF fbMC_ReadActualPositionAxis3.Position < -164.2 THEN
                         rState:=2098;

                 END_IF
 END_IF
 //Putting Piece in top left
 IF rState =2098 THEN

  fbMC_MoveAxis1_(Axis := GVL.Axis1,
                 Execute := TRUE,
                 Position := -123.43,
                 Velocity:= 4000);

                 fbMC_MoveAxis2_(Axis := GVL.Axis2,
                 Execute := TRUE,
                 Position := 38.7,
                 Velocity:= 4000);

                 fbMC_MoveAxis3_(Axis := GVL.Axis3,
                 Execute := TRUE,
                 Position := -182,
                 Velocity:= 4000);

                 IF fbMC_ReadActualPositionAxis3.Position < -181.9 THEN
                         rState:=2099;

                         bGrip2Vacuum:=1;

                 END_IF
 END_IF
//Returning to Initial position
IF rState =2099 THEN

   fbMC_MoveAxis1_1(Axis := GVL.Axis1,
                 Execute := TRUE,
                 Position := -1,
                 Velocity:= 4000);

                 fbMC_MoveAxis2_2(Axis := GVL.Axis2,
                 Execute := TRUE,
                 Position := 5,
                 Velocity:= 4000);

                 fbMC_MoveAxis3_3(Axis := GVL.Axis3,
                 Execute := TRUE,
                 Position := -2,
                 Velocity:= 4000);
                 IF fbMC_ReadActualPositionAxis3.Position >= -3 THEN
                         rState:=2100;

END_IF
END_IF
IF rState=2100 THEN



 fbMC_MoveAxis1_1(Axis := GVL.Axis1,
                 Execute := FALSE,
                 Position := -1,
                 Velocity:= 4000);

                 fbMC_MoveAxis2_2(Axis := GVL.Axis2,
                 Execute := FALSE,
                 Position := 5,
                 Velocity:= 4000);

                 fbMC_MoveAxis3_3(Axis := GVL.Axis3,
                 Execute := FALSE,
                 Position := -2,
                 Velocity:= 4000);
rState:=2110;
                 END_IF

  //Rotating Axis 4 to change gripper Vacuum
IF rState=2110 THEN

                 fbMC_MoveAxis_4_(Axis := GVL.Axis4,
                 Execute := TRUE,
                 Position := 0.5,
                 Velocity:= 2);

IF fbMC_MoveAxis_4.Position > 0.49 THEN
rState:=2111 ;
END_IF

END_IF
```

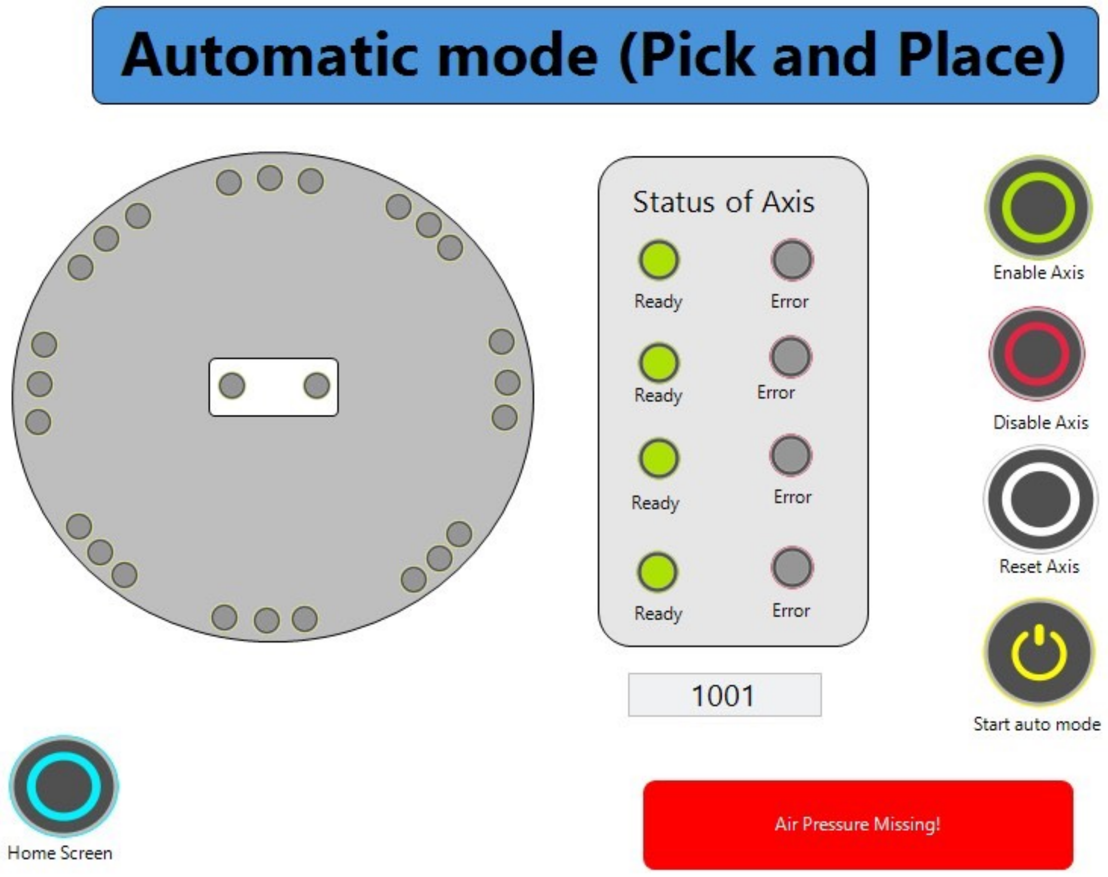
```
IF rState=2111 THEN

                  fbMC_MoveAxis_4_(Axis := GVL.Axis4,
                  Execute := FALSE,
                  Position := 0.50,
                  Velocity:= 2);
rState:=2112;
END_IF

//Putting piece a bit up of bottom left
IF rState =2112 THEN

  fbMC_MoveAxis1_1_(Axis := GVL.Axis1,
                  Execute := TRUE,
                  Position := -97.9,
                  Velocity:= 4000);

                  fbMC_MoveAxis2_2_(Axis := GVL.Axis2,
                  Execute := TRUE,
                  Position := 166.8,
                  Velocity:= 4000);

                  fbMC_MoveAxis3_3_(Axis := GVL.Axis3,
                  Execute := TRUE,
                  Position := -3.27,
                  Velocity:= 4000);
Timer3(IN:=TRUE,
        PT:=T#2S);
                  IF fbMC_ReadActualPositionAxis3.Position < -3.1   AND Timer3.Q=1
THEN
                          rState:=2113;

                  END_IF
 END_IF

 IF rState =2113 THEN

  fbMC_MoveAxis1_1_(Axis := GVL.Axis1,
                  Execute := FALSE,
                  Position := -70.21,
                  Velocity:= 4000);

                  fbMC_MoveAxis2_2_(Axis := GVL.Axis2,
                  Execute := FALSE,
                  Position := 178.14,
                  Velocity:= 4000);

                  fbMC_MoveAxis3_3_(Axis := GVL.Axis3,
                  Execute := FALSE,
                  Position := -22.67,
                  Velocity:= 4000);

                          rState:=2114;

 END_IF

 //Putting Piece in bottom *Lefyt
 IF rState =2114 THEN

  fbMC_MoveAxis1_1_1(Axis := GVL.Axis1,
                  Execute := TRUE,
                  Position := -119,
                  Velocity:= 4000);

                  fbMC_MoveAxis2_2_2(Axis := GVL.Axis2,
                  Execute := TRUE,
                  Position := 187.3,
                  Velocity:= 4000);

                  fbMC_MoveAxis3_3_3(Axis := GVL.Axis3,
                  Execute := TRUE,
                  Position := -30,
                  Velocity:= 4000);

                  IF fbMC_ReadActualPositionAxis3.Position < -29.9 THEN
                          rState:=2115;

                          bGrip2Vacuum:=0;
```

```
//       Timer4(IN:=TRUE,
//PT:=T#2S);
                  END_IF
  END_IF
   IF rState =2115 THEN
   fbMC_MoveAxis1_1_1(Axis := GVL.Axis1,
                  Execute := FALSE,
                  Position := -91,
                  Velocity:= 4000);

                  fbMC_MoveAxis2_2_2(Axis := GVL.Axis2,
                  Execute := FALSE,
                  Position := 195.86,
                  Velocity:= 4000);

                  fbMC_MoveAxis3_3_3(Axis := GVL.Axis3,
                  Execute := FALSE,
                  Position := -47.13,
                  Velocity:= 4000);
rState:=2116;
END_IF
//Returning to Initial position
IF rState =2116 THEN

    fbMC_MoveAxis1_1_1_(Axis := GVL.Axis1,
                  Execute := TRUE,
                  Position := -1,
                  Velocity:= 4000);

                  fbMC_MoveAxis2_2_2_(Axis := GVL.Axis2,
                  Execute := TRUE,
                  Position := 5,
                  Velocity:= 4000);

                  fbMC_MoveAxis3_3_3_(Axis := GVL.Axis3,
                  Execute := TRUE,
                  Position := -2,
                  Velocity:= 4000);
```

```
                    IF fbMC_ReadActualPositionAxis3.Position >= -3 THEN
                              rState:=2117;
END_IF
END_IF
 IF rState=2117 THEN
                    fbMC_MoveAxis_4_4(Axis := GVL.Axis4,
                    Execute := TRUE,
                    Position := 0,
                    Velocity:= 2);
rState:=2200;
END_IF
```

Appendix A.4 HMI Manual and Automatic screens


Manual Mode Control Screen
Axis Name
Axis 1
Axis 2
Axis 3
Rotatory Axis
Status of Axis
Ready
Error
Ready
Error
Ready
Error
Ready
Error
Actual Position value
Calibration
Manual Move
-0.29
Homing
Upward
Downward
6.03
Homing
Upward
Downward
-2.45
Homing
Upward
Downward
0.00
Counter-Clock
Clockwise
Home Screen
Reset Axis
Enable Axis
Disable Axis


Automatic mode (Pick and Place)
Status of Axis
Ready
Error
Ready
Error
Ready
Error
Ready
Error
1001
Enable Axis
Disable Axis
Reset Axis
Start auto mode
Home Screen
Air Pressure Missing!