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

INTEGRATION OF SCARA ROBOT WITH BECKHOFF INDUSTRIAL CONTROLLER

SCARA ROBOTI INTEGREERIMINE BECKHOFFI TÖÖSTUSKONTROLLERIGA

MASTER THESIS

Student: Md Arifur Rahman

Student code: 165594MAHM

Supervisor: Mart Tamre, Professor and
Programme Director (Mechatronics)

Tallinn, 2019

(*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/

Department of Electrical Power Engineering and Mechatronics

THESIS TASK

Student: Md Arifur Rahman, 165594MAHM

Study programme: MAHM02/13 - Mechatronics

Main speciality: Mechatronics

Supervisor(s): Professor, Mart Tamre, +372 620 3202

Consultants: None

Thesis topic:

(in English) Integration of Scara Robot with Beckhoff Industrial Controller

(in Estonian) Scara Roboti Integreerimine Beckhoffi Tööstuskontrolleriga

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Study of available Literatures and evaluating the suitability of Proposed idea	
2.	Getting familiarized of the Beckhoff controller and TwinCAT system	
3.	Get detail knowledge of the robot's parameters	
4.	Building Demo setup for safe testing proposed method	
5.	Finalizing the control algorithm	
6.	Executing test run with the main robot	
7.	Analyzing the outcome and making improvement as required	
8.	Compiling the thesis report	

Language: English

Deadline for submission of thesis: 03.01.2020

Student: Md Arifur Rahman ..... "....."....2020

Supervisor: Prof. Mart Tamre ..... "....."....2020

Head of study programme: Prof. Mart Tamre ..... "....."....2020

CONTENTS

PREFACE

This thesis and the experiments are done in industrial robotics laboratory at mechatronics and autonomous systems centre of Tallinn University of Technology. This thesis idea was initially originated by Prof. Mart Tamre, and later on the proper reasoning of this research was found out by thorough study of research articles journals. Mr. Tamre deserves special gratitude because of his cooperation and support for completing this research. In addition, I am also thankful to all members of mechatronic engineering department, who indirectly facilitated my study and research during the whole duration. I also like to thank Tallinn University of Technology for giving me the opportunity to conduct my masters study. I am also thankful to the ABB company, from where I got important technical knowledge related to the field and came to understand the technological needs from industrial perspective. I also appreciate my friends and colleagues for their cooperation and advice during my stay in Estonia. Finally I am grateful to my family who was always at my side, gave me financial and moral support and inspiration for accomplishing greater goals.

List of abbreviations and symbols

ANN	Artificial Neural Networks
EtherCAT	Ethernet for Control Automation Technology
FBD	Function Block Diagram
HMI	Human Machine Interface
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IL	Instruction List
IoT	Internet of Things
LD	Ladder Diagram
OEM	Original Equipment Manufacturer
OROCOS	Open Robot Control Software
PC-ORC	PC-based Open Robot Control
PLC	Programmable Logic Controller
PMSM	Permanent Magnet Synchronous Motor
SCARA	Selective Compliance Assembly Robot Arm
ST	Structured Text

1 INTRODUCTION

Starting from the mid-twentieth century Robotics is one of the major topics in the field of industrial automation and automation research. The first industrial robot was developed in the year 1959 [1] and ever since the technologies behind robotics are being modernized. The latest motivation in robotics research is concentrated on improved vision systems, artificial intelligence and machine learnings. Despite that as always speed, connectivity and programmability are still among the top concerns in industrial automation. Especially in the recent concept of industry 4.0, connectivity and decentralized decision making are the factors that are going to get more importance in future industrial processes.

The trend in the manufacturing industries is shifting toward flexibility and connectivity. Because of extensive modernization in information technology and communication across the globe, nowadays it is practical to execute the ideas that were once impossible. In recent time customer's expectation is changing faster than ever. And as a result, it is getting more challenging to upgrade the manufacturing process for making sure the product's diversity. The idea of mass production is turning to be obsolete and instead customized production is making its place. In this context industrial automation and more specifically industrial robotics going through a technology revolution, when it is a priority for automation engineers to deliver according to the demand from consumers and industrial planners.

The question going to be addressed in this thesis is, how effectively an old industrial robot can be integrated into a modern production facility when the technology is rapidly shifting and the demand is under constant change. A description of the industrial robot's evolution is given in chapter 2 of this thesis. But in short, we are in the transition period toward the fourth industrial revolution or industry 4.0 and this revolution is expected to be a combination of several major innovations in digital technology. The first and foremost factor of industry 4.0 is the interconnection between machines, devices, sensors and peoples. It gives us a clear indication that in the coming years, interconnected machines will play a big role in industrial automated process. In this thesis, the problem of integrating old generation machines into a modern industry is addressed in the context of current trends and requirements.

This thesis is intended to give a solution for upgrading the control system of older industrial robots. Primarily this research is supposed to benefit small and medium size industries who operate a comparatively small fleet of robots. As industrial robots are

rather expensive machines they cannot be procured on a frequent basis. And for smaller industries, it is a burden to finance the purchasing money for a new robot. In that situation, just updating or replacing the robot's control system is a much economic decision for small industries.

This thesis will act as a sample case study for smaller industries to evaluate the idea of replacing an old robot controller with newer control systems. Moreover this thesis will also work as a ground for future academic research on automation and industrial robotics. As technology is under continuous change, the priorities of industrial automation are also expected to change over time. In that alternating scenario, this thesis will provide important information about industrial robots and control systems, for conducting further research in this field.

2 HISTORICAL BACKGROUND AND LITERATURE REVIEW

In this chapter, a chronological description is given about the evolution of industrial robotics and technological changes. Afterward, a brief overview of existing industrial robot control systems is given and recent researches done in this field are discussed.

2.1 Problem statement

Typical industrial robots used in factories usually have specific controllers for their operations. The majority of these controllers are suitable for a single mode of operation with a specific robot model. But when it comes to versatility and flexibility they are not the best option to work with. Because almost every OEM (Original Equipment Manufacturer) robot controller works with only a few robot models. And many of these controllers do not have enough options to integrate the machine with other equipment in a factory. When connectivity is a major concern for factory automation, these OEM controllers bring a huge drawback to the industrial planners. It makes the users dependant on a few hardware and as a result, reduces the flexibility for future changes.

With proper maintenance, the hardware of an industrial robot can provide the desired service over a decade; but the control system cannot do the same for more than few years. Because of the rapid modernization of technologies, they need to be updated quite often. But control system upgrade is not always supported by the companies. Besides that, purchasing a new OEM controller is also an expensive decision. In this scenario, taking the approach of developing a universal control system for industrial robotics is a much reliable option, since it is less expensive compared to OEM controllers and more flexible for upgrades and modifications.

2.2 Objectives

The main objective of this thesis is to verify the concept of replacing the OEM controller of an older generation mechanically viable robot with a modern industrial control system. The background research of relevant literature and experiment is done during the thesis to support this proposal. The goal of the experiment is to replace an existing controller of a third-generation SCARA robot with a newer

generation Beckhoff industrial controller. The tasks include interfacing the robot's hardware with the control system devices and making a properly functional control interface to operate the system directly from a computer . The experiment is evaluated by running motion control programs in the new controller and the robot should demonstrate an exact point to point movements as it is commanded by the user. In short, there are three stages of this experiment.

1) Establishing a functional interface between robot and control hardware;
2) Executing robotic motion with structured text program;
3) Configuring communication protocol for remote operation.

2.3 Progress of industrial robotics

The age of industrial robots started from the mid of twentieth century approximately during the same time of the third industrial revolution. Although the concept of robotics was not entirely new at that time, because the idea of artificial mechanical manipulator and machine's intelligence was present long before that mostly in forms of science fictions and stories. The first known use of the word 'Robot' was in a science fiction play by a Czech writer Karel Čapek in 1920 [2]. The term 'Robotics' first mentioned between the years 1940 and 1950 by another science fiction writer Isaac Asimov in science fiction stories under his novel called 'I, Robot'. In this novel he also brought up the widely known 'Three Laws of Robotics' that is also called as 'Asimov's Law' [3].

The advancement of robotics is generally categorized in four generations. The first generation of robotics began from 1950 and it lasted till 1967. In 1954 a scientist name George Devol designed the first industrial robot [2]. George Devol together with an entrepreneur Joseph Engelberger founded a company in 1956 to manufacture robots for industrial applications. The company was named 'Unimation' and their first development in the year 1959 was called UNIMATE [1]. That robot was installed in General Motor's production line in New Jersey, USA in 1962. It was programmed for doing single task like extracting materials from casting machine [2].

The significance of robotics in industrial sector influenced many other companies to devote in robotics development, and as a result by 1960s several new robot manufacturers emerged. One of these companies was AMF (American Machine and Foundry) Corporation, who was the first company to brought robotics in Japanese industries as well by their 'Versatran' robot in 1967 [2]. The first robot in Europe was

installed in the same year 1967 at 'Svenska Metallverken', a company in Sweden. Like most of the first generation robots it was programmed for doing simple monotonous pick and place work [4].

The second generation robotics are described by the years of development between 1968 and 1977. During this time sensors were introduced in robotics that helped to develop close loop systems for motion control. At this similar time PLCs were also invented, that leads to the ability to control robotic systems from digital equipments [5]. In 1973 German automation company KUKA developed a robot that had six electromechanically driven axes, it was named 'Famulus'. One year after that in 1974 the first microcomputer controlled robot called T3 (The Tomorrow Tool) was made by 'Cincinnati Milacron' [1]. The major users of these earlier industrial robots were automotive industries where they were mainly operated for sensitive jobs like welding. Because of high reliability and accuracy, those robots gained immense popularity. It is considered that the era of robotics started from 1980s, because during this time many scientific and technological improvements contributed to robotics research [2] [5].

With the rise of digitalization and automation in industries, and the developments of programmable controllers and digital equipments started a new era in industrial robotics [6]. The third generation robotics are defined by the developing year from 1978 to 1999. In 1978 a new design was proposed by a Japanese scientist Hiroshi Makino from the University of Yamanashi [2]. It was named SCARA (Selective Compliance Assembly Robot Arm) because of its limited maneuverability in horizontal direction and rigidity in vertical direction. In 1992 another new robotic kinematic structure was proposed by a Swiss scholar Reymond Clavel from the Swiss Federal Institute of Technology in Lausanne [2]. It was known as the Delta robot, and the design was based on parallel kinematics instead of serial kinematics. It had three translational axes and one rotational axis and was able to operate at a much faster speed because of its lightweight and relatively simpler kinematic chain.

From the 1980s robot manufacturers also commenced developing robot programming languages for their machines. For instance, Unimation created a language called VAL (Variable Assembly Language), Fanuc started to use Karel programming language from 1988 and ABB developed their own language called Rapid in 1994 [7]. In 1994, synchronization of robots was attained by a company called Motoman, which is a subsidiary of Yaskawa Corporation. Motoman made the robot control system MRC that was able to synchronize control of two robots with up to 21 axes [2]. In 2004 they

released their improved control system NX100 that was capable to synchronize control of 38 axes from four robots [1].

The begin of fourth generation robotic is considered from the year 2000 and that is still underway. Significant additions in this time are the inclusion of superior computing ability, sophisticated sensors, artificial intelligence and machine learning ability. In addition, standardization of communication protocols and development of common platform for robot programming is also initiated in this period [5]. The high reliability in control systems and sensors leads the robots for non-commercial applications, for instance robotic vacuum cleaner and robotic loan mowers are being used by private owners. In addition collaborative robot or cobot has also been introduced by number of companies.

2.4 Industry 4.0 and current trends in industries

The term Industry 4.0 that refers to the fourth industrial revolution is believed to be underway in recent years. The fourth industrial revolution is triggered by the advancement of Information and Communication Technology (ICT) and it is a combined outcome of innovations in digital technology. The concept of Industry 4.0 appeared in 2011 when the idea was presented by Henning Kagermann on behalf of the German federal government at Hannover fair [8]. The idea behind industry 4.0 is to establish cyber-physical systems and smart factories, where every machine and sensor will be connected under one system, which will lead to smart automation and the establishment of distributed control and the Internet of things (IoT).

In industry 4.0 concept, the technical process and the business process of a company are proposed to merge in one system to improve the flexibility in decision making, optimization in production quantity and in mass custom production. Industry 4.0 is comprised of several design principles, that technically serve as a guideline to understand and implement this concept in a company. There are often six design principles as explained by experts; however, in 2016, it has been depicted through a study that there are four main design principles for industry 4.0 [9]. These design principles are namely Interconnection, Information transparency, Decentralized decisions and Technical assistance.

From these design principles, it is understandable that industry 4.0 is not only about technical innovation, but it has to be an improvement of technical and business

process combined. And at the core of this concept is cyber-physical systems and Internet of things, where all of the process and machines need to be connected in one platform. This brings the issue of optimizing previously existing processes, when it becomes necessary to reengineer old equipments and configure both hardware and software platforms to make sure the interconnectivity. At the same time standardizations of technologies are important because of diversity in equipments and processes to ensure flexibility in automation planning.

2.5 Methods and standards in industrial robotics

The field of industrial robot is quite heterogeneous in terms of hardware, communication protocols and languages used for programming. Since industrial robot's market is dominated by major robot manufacturers, it is also challenging for open source developers to implement new technologies unless they are approved by bigger organizations. As most robot manufacturers and control system builders are intense to provide complete solution for their robot, there is lack of common benchmark for evaluating the performance of one robot controller with another. As a result, technological improvements in industrial robotics are comparatively sloth. While currently researches are taking place on artificial neural networks (ANN), machine learning and big data; technologies used industrial robotics are relatively old, that includes PID control loop, forward/inverse kinematics and so on.

There are distinct standardizations exist for PLC programming, motion control and communication protocols. For instance, IEC 61131-3 is an open international standard that defines a guideline for software architecture and programming languages for programmable controllers. The current edition of this standard has published in 2013, and this version specifies the syntax and semantics of two textual languages: structured text (ST) and instruction list (IL); and two graphical languages: ladder diagram (LD) and function block diagram (FBD). 'PLCopen' is another common standards that creates vendor independent libraries for different PLC applications. Especially for motion control applications PLCopen function blocks are widely accepted to automation system designers.

Particularly for robotic applications there are few standards, that is recently developed or under development. 'ISO/TC 299' (Standardization in the field of robotics, excluding toys and military applications) is one of these standards that deals to ensure proper safety, standardized interfaces and performance criteria [10]. IEEE

1872 (Ontologies for Robotics and Automation) is another standard developed by IEEE that provides a unified way of representing knowledge and a common set of term definitions from robotics and automation domain [11].

The robot controllers currently being used in industries are mostly modular type, they are built to operate with specific types of robot in mind. Some of current generation controllers are ABB's IRC5 controller and KUKA KR C4 controller. IRC5 is ABB's fifth generation controller, that is designed to operate mainly with ABB robots. IRC5 has different variants suitable for specific applications areas. IRC5 is programmed in ABB's 'RAPID' programming language [12]. On the other hand, KUKA KR C4 controller has five variants, each has a different size factor and different numbers of drive axes. KR C4 supports high level PLC programming that is useful for I/O interfacing and utilization of motion control function blocks [13].

There are different existing approaches in industrial robot control development; namely, model-based development, cost/performance driven development, automation technology driven development and application driven robot control development. Each of these approaches has its own advantage, especially in recent time model based development is the fundamental one used in industrial robotic developments [14]. Researches has been done in the past for developing open robot control systems such as OROCOS (Open Robot Control Software) and PC-ORC (PC-based open robot control system) [15], [16]. Robot Operating System (ROS) was created in 2007 with the intention of making an open platform for robotic developments. Especially the development of 'ROS-Industrial' in 2012 opened possibilities for vendor-independent open-source drivers and prospects of interchangeable hardware components through standardized interfaces.

2.6 Related researches and outcomes

Numerous researches has been conducted on industrial robotics for improving performance and acceptance criteria. Many of these researches addressed the issue of reengineering and reusing industrial robots with new generation equipments and standardised protocols.

In 2018 a conference paper was presented in IFAC PapersOnLine [17], that demonstrates the result of an experiment to re-engineer an industrial robot for performing new set of tasks. The objective of that study was to make modification of

an industrial KUKA robot and to reuse it with new control system in a different industrial setting. The objective was accomplished and outlined in the form of simulation and execution with real machine. In that study communication method was partly covered and the new system was prepared with unconventional hardware and software.

Alternatively, several researches was done to evaluate the performance and reliability of different industrial protocols. For instance in 2012 a research paper was published by two researchers from Korea Electronic Technology Institute [18], who proposed a centralised soft robot control approach based on EtherCAT protocol. They executed an experiment to control a 6 joints robot from a software based master device with 10 KHz main data transfer frequency and 1 KHz position control frequency. The experiment demonstrated a short response time and real time performance while the delay among the slave devices was less than 50 μs.

In 2017 another study was published by a group of researchers from Slovak University of Technology in Bratislava [19]. They experimented standard Ethernet based TCP/IP protocol for controlling a robotic cell with three robots. The aim was to find a reliable solution for robot control in a non-Real-time network of communication. Although different industrial protocols were also discussed in the paper, the experiment was concentrated on TCP/IP and UDP protocols. According to the researchers, the concept is applicable for equipments that is programmed for autonomous behaviour up to certain extant and not dependable on precise timing of control instruction.

2.7 Thesis structure

This thesis is comprised of seven chapters. Chapter 1 is about an introduction of the thesis topic. Chapter 2 gives a historical overview and researches on industrial robotics. Chapter 3 is about the technical description of the hardware and software used in this thesis. In chapter 4 detail explanation is given about the hardware integration process. Chapter 5 is about describing the process of software commissioning and device parameterizations. In chapter 6 test executions have been described and performance is analyzed based on test results. In chapter 7, an overall summary of the entire thesis is given.

3 DESCRIPTION OF HARDWARE AND SOFTWARE

In this chapter, detail explanation is given about the hardware and software tools used in this thesis including the robot and control system devices. As the motive of the thesis is to demonstrate an optimum solution with available equipments; the selection of the tools was made primarily based on availability and compatibility. In addition technical specifications were analysed in order to maximize the performance of the new development.

3.1 Hirata AR-S350 SCARA robot

This SCARA robot from industrial robotics laboratory in mechatronics and autonomous systems centre was taken into account for the experiment. This robot was made by Hirata company in Japan and it was required to replace previous OEM controller with a new control system. A brief specification of the OEM controller is given in appendix. This model (AR-S350-4-200) of the robot was primarily designed in 1997 [20] and this unit was manufactured in 1999. A picture of this robot and manufacturer's label is shown in figure 3.1.

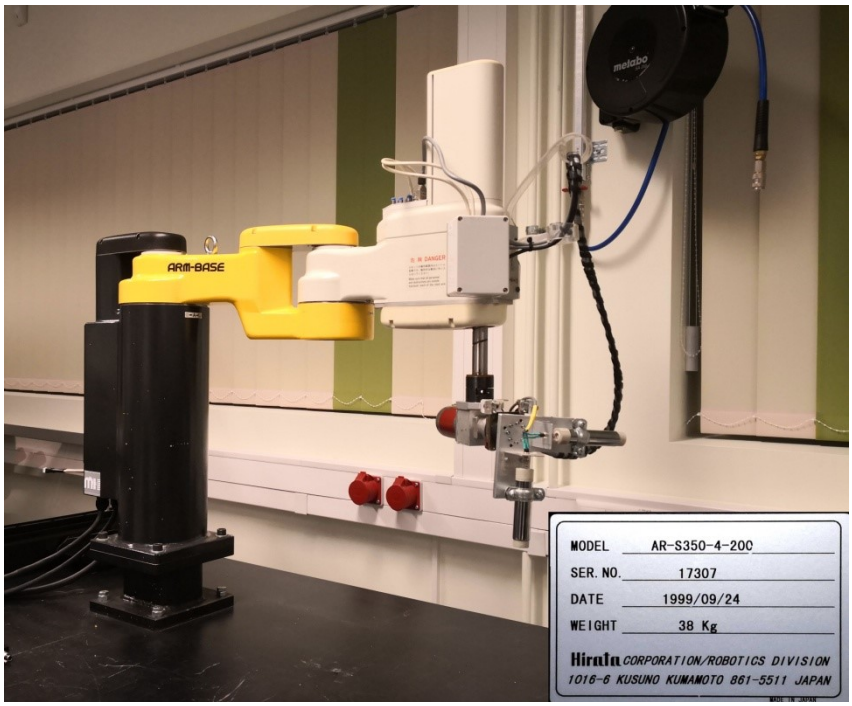


Figure 3.1 Hirata AR-S350 robot used in this thesis.

Like a typical SCARA robot, this is a 4-axis robot. Three of these axes are rotational axis and one is translational axis. Where A and B axes are responsible for horizontal movements, Z-axis adjusts the height and W-axis determines the orientation of the end effector. There is one AC servo motor connected with each axis and all of these motors are attached with encoders. These encoders are used for getting position feedback about motor's shaft. In addition there is one sensor connected with every axis, that indicates the limit of robot's working area for each axis. Detail technical parameters and physical construction of this robot is given in appendix 1.

3.1.1 Motors

There are four AC servo motors in this robot, that performs all physical movements through gears connected with each axis. All of these motors are AC 3-phase synchronous motor. Motor parameters are given in table 3.1. Since these motors are enclosed inside the robot, it was not possible to access the motor's nameplate. The only available approach was to acquire these parameters from third party sources. Motor's of the similar model has been searched online, and the values were taken from nameplate pictures of corresponding motor models. Pictures are given in appendix 3.

Table 3.1 Hirata AR-S350 robot's basic motor parameters [21]

	A-Axis motor	B-Axis motor	Z-Axis motor	W-Axis motor
Motor's Model	Panasonic MSM042A2UE	Panasonic MSM022A2UE	Panasonic MSM022A2UE	Panasonic MQMZ012A2U
Motor Type	3 Ø AC Servo	3 Ø AC Servo	3 Ø AC Servo	3 Ø AC Servo
Power Output	400 W	200 W	200 W	100 W
Rated Input Voltage	106 V	92 V	92 V	63 V
Rated Input Current	2,5 A	1,6 A	1,6 A	1,0 A
Rated Frequency	200 Hz	200 Hz	200 Hz	200 Hz
Rated Speed	3000 rpm	3000 rpm	3000 rpm	3000 rpm
Connection	Y	Y	Y	Y
Rated Torque	1,3 Nm	0,64 Nm	0,64 Nm	0,32 Nm

By construction all of these motors are permanent magnet synchronous motor or PMSM; where the rotor is installed with permanent magnets and the stator has electromagnets connected with three phase AC power supply. When the stator is energised with AC current, a rotating magnetic field is produced. The rotor takes turns because of interactions between the permanent magnet's magnetic field and the stator's rotating magnetic field. Internal construction of a PMSM is shown in figure 3.2.

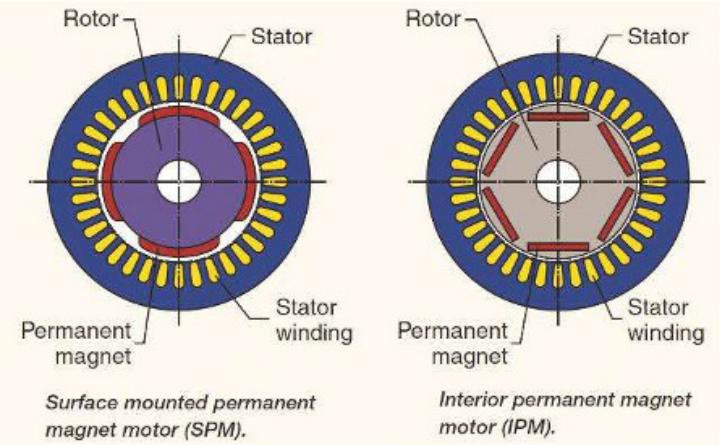


Figure 3.2 Internal construction of permanent magnet synchronous motor [22]

As this is synchronous motor, slip is always zero, and the rotor always rotates at the synchronised speed with the rotating magnetic field. Speed of rotating magnetic field is directed by the frequency of the electric current in stator winding. As a result, by changing the current's frequency in stator winding, speed of synchronous motor is controlled. This speed can be calculated using equation 3.1.

$$N_S = \frac{120f}{p} \tag{3.1}$$

where N_S – synchronous speed, rpm,
f – frequency of AC supply, Hz,
P – number of poles.

3.1.2 Encoders

As mentioned earlier, encoders are used for getting feedback about the motor's shaft position. There are primarily two types of encoders; incremental type and absolute type. In this robot, all of these four encoders are incremental type encoder. Since these encoders give data about the angular position, they are categorized as rotary encoder and a rotary encoder works following the movement of a disks connected with shaft. In figure3.3, structure and working principle of an incremental rotary encoder is shown.

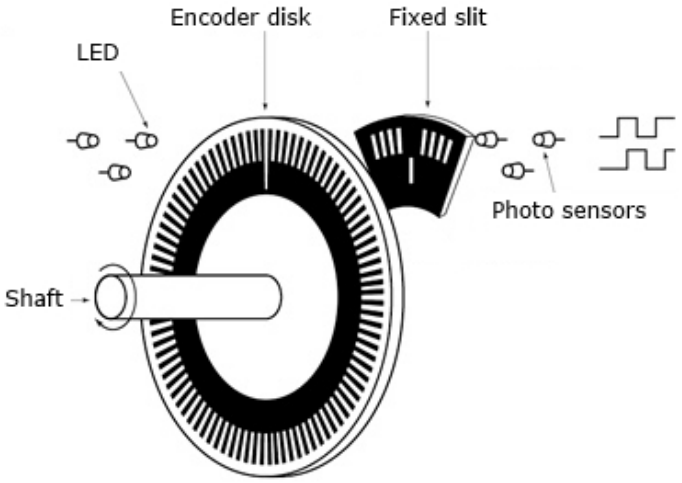


Figure 3.3 Incremental rotary encoder structure [23]

An incremental encoder gives the output as electrical pulses. These pulses are generated by sensors connected next to the encoder disk. For a known pattern in the encoder disk, number of pulses are counted to calculate the angular rotation, velocity and acceleration. To identify the direction of rotation a second sensor is required in addition to extra slits in the disk. Alongside these two sensors a third sensor is used to detect full revolutions of the shaft. In figure 3.4, output signal patterns are shown for an incremental encoder.

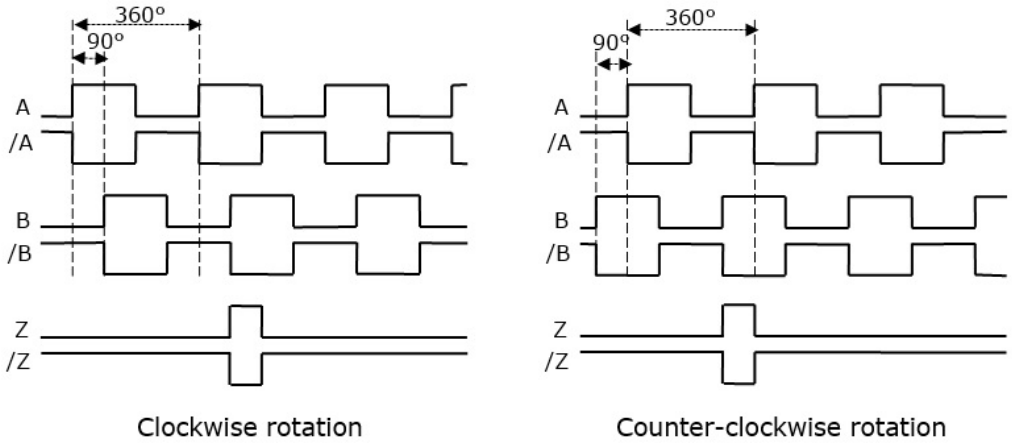


Figure 3.4 Output pulses of TTL incremental encoder

From the figure it is seen that, channel A and channel B sensors give output square wave that is 90° phase shifted. The direction of rotation can be detected by identifying the leading and lagging phases of these channels. The encoders used in this robot are TTL-2500-5V encoder. The speciality of TTL encoder is that, they output complementary signal for each channel. Complementary signals are useful for

eliminating the noise induced in transmission wire because of electromagnetic interference.

This encoder gives 2500 pulses in one revolution. But because of the channel's phase offset, it outputs one rising edge or falling edge in every 90° either in channel A or in channel B. As a result, this encoder takes four count in one pulse, that makes the entire number of counts 10000 in one revolution. By observing this counts, it is possible to calculate exact angle of rotation, angular velocity and acceleration of the motor's shaft.

$$\phi = \frac{n * 360}{10000} \tag{3.2}$$

where ϕ – motor's angle of rotation, °,

n – number of encoder counts.

Unlike an absolute encoder, an incremental encoder does not give original position data by default. Therefore it is required to perform homing operation to identify the real shaft position.

3.1.3 Sensors

There are four proximity sensors installed in this robot. These are digital type sensors, so that they do not give information about analogue distance. These sensors can only be turned on or off and they are used as limit switch for each axis. The list of installed sensors is given in table 3.2.

Table 3.2 List of sensors in AR-S350 robot [21], [24]

	Model no.	Sensor type	Output type	Output operation	Axis operational limit
A-Axis sensor	GXL-8FB-R	Inductive	NPN	Normally closed	-110° to 110°
B-Axis sensor	GXL-8FB-R	Inductive	NPN	Normally closed	-135° to 135°
Z-Axis sensor	GXL-8HB-R	Inductive	NPN	Normally closed	
W-Axis sensor	GXL-8HB-R	Inductive	NPN	Normally closed	

As these are inductive sensors, they only detect metals targets. Since that there are metal sensor tabs mounted with each axis in the robot. NPN type sensors are also known as sinking sensor as they sink the output to the ground. Sample wiring diagram and internal construction of a NPN type sensor is shown in figure 3.5.

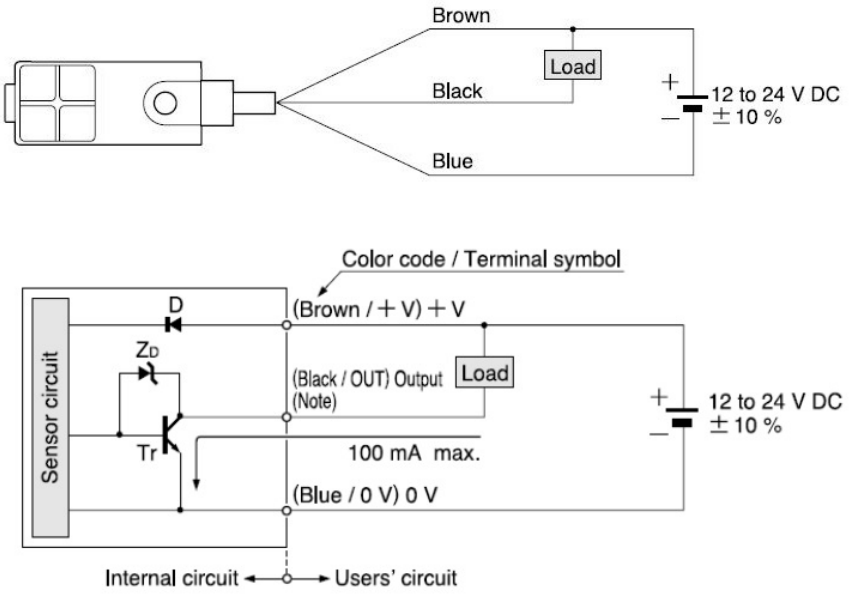


Figure 3.5 Sample wiring diagram of GXL-series npn type sensor [24]

In the case of these sensors, outputs are NC or normally closed. This indicates they output high signal when there is no metal object detected; and when the sensor detects metal, the outputs turn low.

3.2 Beckhoff CX2030 controller

Beckhoff controller has been selected for this work because of several advantages over other controllers. The foremost is the ease of integration and programming using TwinCAT system. As Twin CAT supports all of IEC 61131-3 programming languages, Beckhoff controllers are programmable with IEC 61131-3 recognised graphical and textual languages; In addition, TwinCAT can also compiles C/C++ programs, that makes it easier to write high level programs for automation tasks.

In addition, Beckhoff controllers are quite modular in type, because of their easily detachable and configurable I/O terminals. And almost all of Beckhoff controllers are configured for EtherCAT protocols, and with additional modules they can also work with other industrial protocols. Beckhoff also has specific libraries for robotic motion control, that includes SCARA robot, delta robot and 6dof robot. Utilizing this pre built libraries, robotic motion control can be performed in more effective and easy manner.

CX2030-0120 is an embedded PC type programmable controller made by Beckhoff. This was one of the newest type of controllers available in the industrial robotics laboratory at the time of this thesis. This controller has Intel core i7 dual core processor with 1,5 GHz of processing speed and it has a memory (RAM) of 2 GB. This controller was installed with Microsoft windows embedded standard 7P, 32 bit operating system. In figure 3.6, a simple combination of this module with power supply unit is shown.

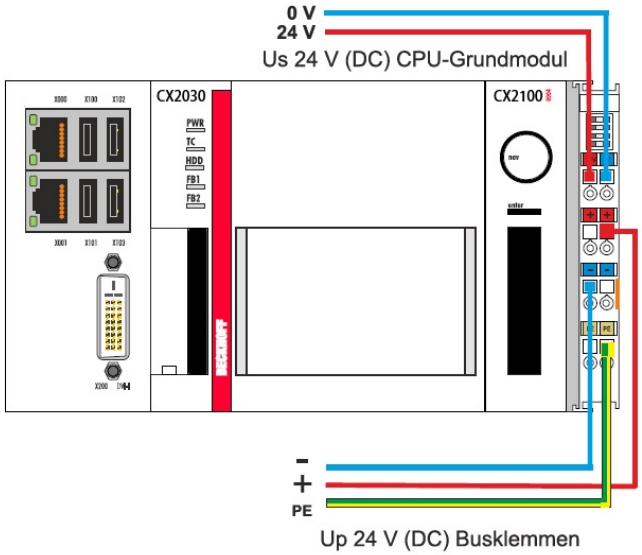


Figure 3.6 Simplest combination of CX2030 controller with power supply module [25]

The basic CX2030 controller has four USB 2.0 interface, two RJ45 Ethernet interface for connecting to LAN or EtherCAT, one DVI-I interface to connect with monitor or panel and one CFast card slot. Typically this controller is used in conjunction with additional modules. Following are the some of the modules used in this work.

CX2100-0914 power supply unit is used to power up the basic CX2030 module. It takes 24 V DC as power source and serves as the primary mean to connect the controller with power source. In addition it connects the terminal bus (K-bus or E-bus) with the main controller, and also provide power supply to bus terminals via power contacts.

EL2008 is an EtherCAT terminal (E-bus) used for digital outputs. It has 8 output channels with maximum output current of 0,5 A per channel. The outputs are single wire connection and they use 24 V DC. So the other end of connected load device has to be at 0 V for proper uses.

EL1018 is a 8 channel EtherCAT terminal (E-bus) for digital inputs. This is also a 24 V DC module with 10 µs input filters. These inputs are single wire connections and they are internally grounded at 0 V. So, it requires logical high (15 V to 30 V) potential to turn on the inputs.

EK1100 module is used for coupling EtherCAT terminals (E-bus terminals) with EtherCAT network. It has two RJ45 port, for EtherCAT in and EtherCAT out connections. This module needs 24 V DC power supply for the power contacts and powering up E-bus modules.

EK1110 module is used at the end of E-bus terminal blocks to further extend the EtherCAT network. It has one RJ45 connector that is used for EtherCAT out connection. This module converts E-bus signals into 100BASE-TX Ethernet signals.

3.3 Beckhoff AX5203 servo drive

This is an AX5000 series servo drive from Beckhoff company and its specific model number is AX5203-0000-0210. In addition to the CX2030 controller, this servo drive was used as part of the control system for running high power circuits in the systems. Essentially it worked as a power amplifier in servo control loop, and it performed the commutation task for the connected synchronous motors.

Figure 3.7 Beckhoff AX5203 servo drive [26]

This servo drive is able run up to two motors at same time with its channel A and channel B. There are separate encoder and resolver interfaces for position feedback in each channel. This drive needs two separate power source for its operation. 24 VDC

power is needed for running control and communication related tasks. And high voltage single phase or three phase AC is required for powering the motors. 24 V system can work independent of the high voltage side, so it is possible to configure the drive in case of uncontrolled situation by turning the AC power off. Important electrical data for output channels are given in table 3.3.

Table 3.3 AX5203 electrical data for single phase 230 V power connection [27]

Electrical parameters	Approximate values
Rated output current / channel	3 A
Maximum rated current / channel	4,5 A
Maximum output current /channel	10 A
Total rated output current	4,5 A
Total maximum output current	20 A
Rated apparent power	2,4 kVA
Power dissipation	85 W

This drive has two RJ45 communication ports. These ports are configured for EtherCAT protocol, one is for incoming and the other is for outgoing line. AX5203 drive's Physical configurations and electrical connection example are shown in appendix 2.

3.4 TwinCAT

TwinCAT is a software platform used in PC based automation projects. This was used in this thesis for configuring and programming the PLC and servo drive. This platform is developed by Beckhoff company. TwinCAT uses different software tools to make the entire system work.

TwinCAT XAR (eXtended Automation Runtime) is one of the tools used by TwinCAT. IT creates a real-time kernel inside windows operating system and dedicates one or more cores inside a multi-core processor for TwinCAT specific tasks. As a result, other applications in OS do not share processing memory with TwinCAT. So it ensures uninterrupted use of processing memory for controlling automation equipments directly from PC in real-time speed.

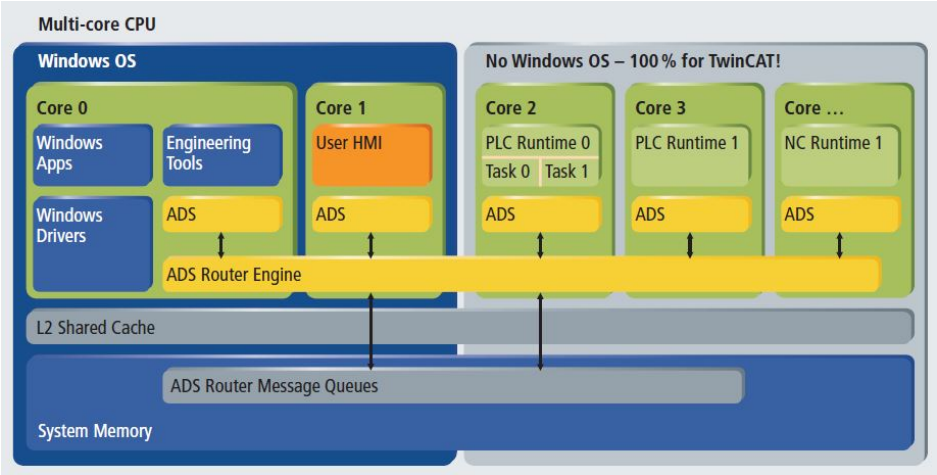


Figure 3.8 Processor's core distribution in TwinCAT XAR [28]

TwinCAT XAE (eXtended Automation Engineering) is the development environment for automation projects. It works based on Visual Studio IDE (Integrated Development Environment). TwinCAT projects are created in XML file format. In this thesis TwinCAT version 3.1 was used that was installed on top of Visual studio 2017. TwinCAT XAE is required only in the development PC where projects are created; on the other hand, TwinCAT XAR is required both in development PC and in the remote embedded PC (controller in this case) where the projects are activated.

In TwinCAT it is possible to write programs in five IEC 61131-3 standardized programming languages. These languages are: ladder diagram (LD), function block diagram (FBD), structured text (ST), instruction list (IL) and sequential function chart (SFC). In this thesis mainly structured text format was used for programming.

3.4.1 TwinCAT drive manager

TwinCAT Drive Manager (TCDM) is an application inside TwinCAT system, that is used for configuration and parameterization of servo drives. Drive manager is automatically added in TwinCAT projects when the system finds a servo drive after device scan. In connection to every channels of the drive, it is required to create one NC/CNC axis configuration. Axis configurations work as an abstraction layer inside TwinCAT system that links hardware configurations with controller's main program. TwinCAT Drive Manager user interface is shown in figure 3.9.

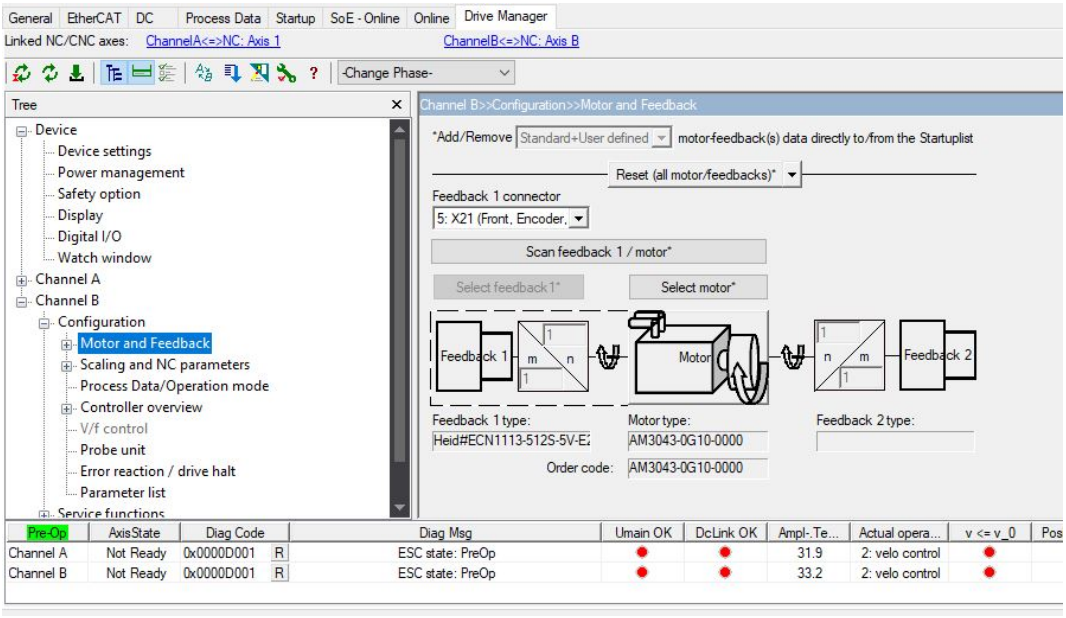


Figure 3.9 TwinCAT drive manager user interface

Few parameters are essential to specify inside drive manager. Device power management is one that required to match with the power connection type. In this case '230 V|1 phase|50 Hz| (Europe AC)' was selected as power setting. For a multi channel drive, motor and feedback data are added individually for each channel. Drive manager loads motor data from XML files stored in a folder name 'MotorPool' inside TwinCAT directory. There is one data file for each available motor types and for missing motor types this file needs to be created manually. These files are created using 'TC Motor Data File Generator' software that was provided by Beckhoff.

3.4.2 TC Motor data file generator

As mentioned above this software is used for making motor data file in XML format. It is a supplementary software provided by Beckhoff. For making the data files it is required to know certain motor parameters and the type of the motor. In following table the list of required parameters are given.

Table 3.4 Required parameters for generating motor data file

Motor Parameters
Motor's construction (rotary/linear)
Functional principle (motor type)
Maximum motor speed
Number of pole pairs
Motor Back EMF constant
Motor peak torque
Motor continuous stall torque
Rotor moment of inertia
Motor peak current
Motor continuous stall current
winding resistance: phase to phase
winding inductance: phase to phase
Motor feedback type
Motor temperature sensor type
Motor brake type

3.5 EtherCAT communication protocol

In industries, communication protocols are used to maintain connectivity between machines and workstations. In this thesis it was used to make real-time communication between the control system devices. There are few widely popular communication protocols used in industries nowadays. EtherCAT (Ethernet for Control Automation Technology) is one of those, that was first developed in 2003 by Beckhoff Automation. From 2004 the rights of EtherCAT are belongs to an independent organization called 'EtherCAT Technology Group' (ETG). Currently ETG is responsible for maintenance and standardization of EtherCAT protocol. EtherCAT is currently standardised under fieldbus standard IEC 61158 and IEC 61784 [29].

3.5.1 EtherCAT communication layers

EtherCAT is an application layer protocol. It works based on IEEE 802.3 Ethernet protocol. EtherCAT uses two types of data transfer. Standard data utilizes layer 1 up to layer 4, just like ordinary Ethernet protocol. But the real time data follows only

physical layer (layer 1) and data link layer (layer 2), and bypasses the other layers. It reduces the cycle time to ensure real-time communication. In figure 3.10, layers of EtherCAT communication protocol are shown.

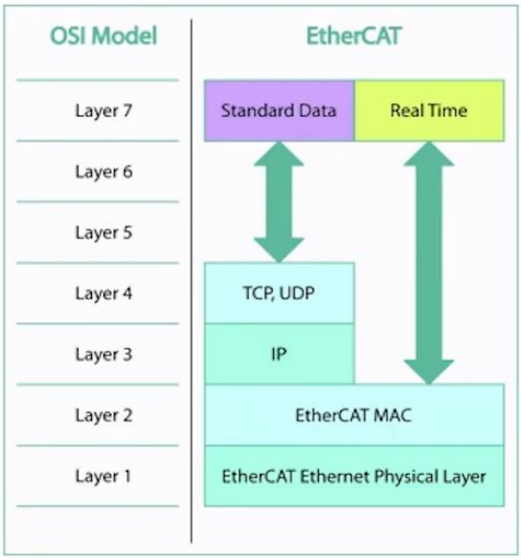


Figure 3.10 OSI model of EtherCAT communication [30]

3.5.2 EtherCAT frame structure

EtherCAT network follows Master-Slave configuration. EtherCAT master is normally a controller that sends data frames through every nodes in network. Each slave device connected to the nodes process the relevant data to that node from the data frame. Each slave device reads and adds to only particular bits into the data frame. EtherCAT master uses only standard Ethernet MAC address. It makes it possible to implement the master in any hardware platform that has RJ45 ports with Ethernet functionality. On the other hand EtherCAT slave devices need to have EtherCAT Slave Controller (ECS) chip in their hardware to process the data frames on the fly. The size of data frames targeted at each node can vary between a 1 bit to 60 Kbytes.

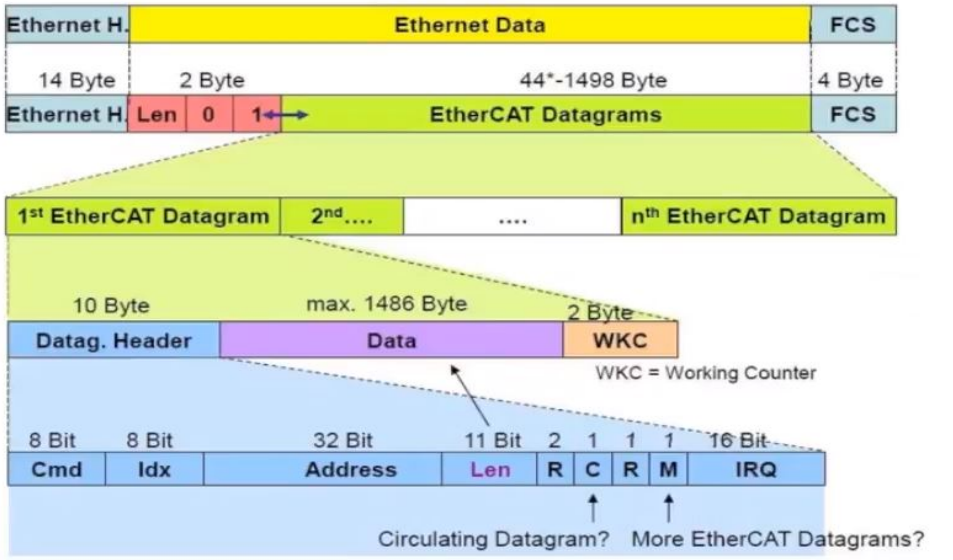


Figure 3.11 EtherCAT data in a standard Ethernet frame [31]

Within one standard Ethernet frame there can be several EtherCAT datagram. Each datagram consists of a datagram header, data itself and the working counter (WKC). The datagram header contains the command, address, length and various check bits. The data holds the message that has to transfer to the slave. The working counter counts the number of actions in the datagram. When the datagram pass through several nodes, each nodes that addressed by that datagram increment the working counter. If the WKC value in the returned data frame does not match with the expected value, it is identified as an transmission error in network. EtherCAT follows a distributed clock mechanism. The I/O functionality of the slave devices can be triggered from master's clock cycle or from the local clock in the slave. The local clock is synchronised with a reference clock in the system.

3.5.3 EtherCAT network topologies

EtherCAT protocol is quite flexible in terms of network topologies. Star, tree, line, bus or combination of these topologies can be used in EtherCAT network. Because of the cable redundancy feature, ring topology is the most appropriate for EtherCAT network. In case of cable break in a ring topology, EtherCAT master can keep the communication on by implementing separate loops using full-duplex operation.

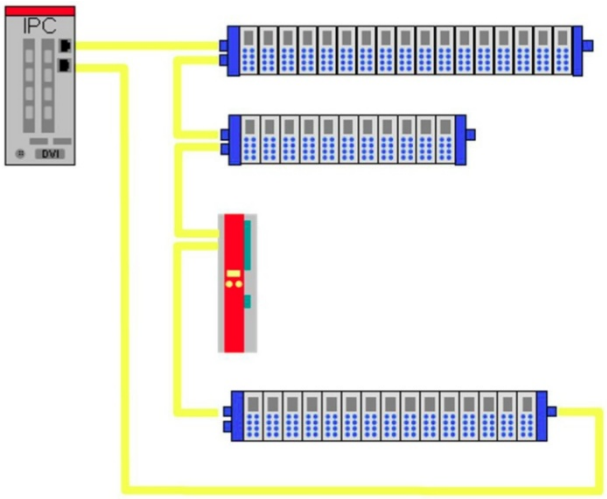


Figure 3.12 An EtherCAT network with ring topology [32]

Every EtherCAT device normally has two ports. One port is connected to receive data frame from previous device and the other port is used for sending the data frame to the next device in network. EtherCAT nodes have self terminating feature. That suggests if a node is disconnected from the next node, the network detects this as open connection and terminate the network at that point. Because of this reason, EtherCAT network does not need any extra module or resistor for bus ending. In an EtherCAT network up to 65535 slaves can be connected regardless of their network topology [29].

4 HARDWARE INTEGRATION

4.1 Test development

Considering the level of risks and complexity of working with the real machine, it was decided to go through a testing procedure with a simplified hardware setup. It was found to be more acceptable and helpful to break down the complexity of the integration process. There were few approaches to conduct the testing procedures. It could be done either in software-in-the-loop method, hardware-in-the-loop testing or with simplified hardware setup. Considering the phase of this work, either hardware-in-the-loop testing or testing with real hardware were the appropriate testing approach.

Hardware-in-the-loop (HIL) testing is the testing method, where real control devices are used to run a process in virtual environment. In this case hardware-in-the-loop testing was omitted, because it requires extra hardware modules for interfacing controller with virtual environment. And making a virtual model would cost unreasonable time. As the control hardware have already been chosen, decision was made to conduct testing with real hardware by making a simplified physical setup of the system, that can resemble the process but in small scale.

4.1.1 Testing model

The testing model was created to resemble a simple servo control loop using real control system equipments chosen for the robot. In addition to controller and servo drive, one spare servo motor was added in this system. This was a Panasonic servo motor and it belongs to the same series that is installed inside the robot. Besides that, one inductive proximity sensor and EtherCAT digital input module were used in the model. In figure 4.1, block diagram of this model and in figure 4.2 picture of this setup is shown.

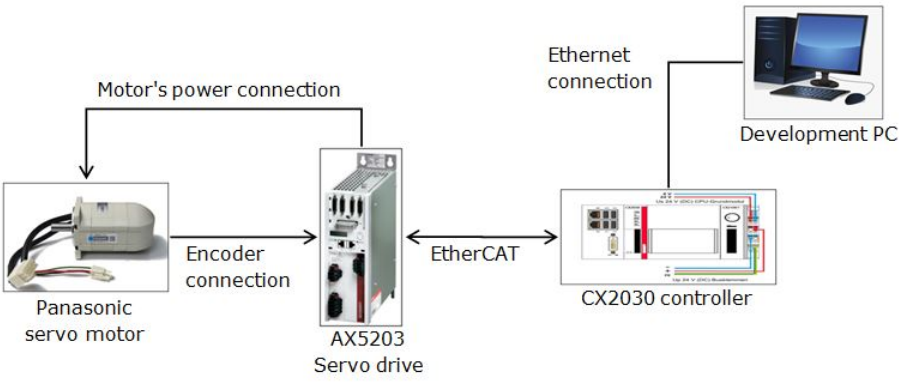


Figure 4.1 Block diagram of the test setup

Figure 4.2 Hardware test setup.

4.1.2 Test outcomes

There were few objectives of this testing that include, finding any flaws in system configuration, sorting out malfunctionality of any hardware modules, applicability of the current drive with the Panasonic servo motors and verifying the usability of TTL encoder's feedback for position calculation. This testing was incredibly helpful to understand the working methods of TwinCAT system and different Beckhoff modules.

Moreover, it lead to find out the problem in the servo drive that was primarily use in the system. The original servo drive (AX5203-0000-0011) installed in this setup was found malfunctional. Steps had been taken to find solution of this problems, that include firmware updating and running the drive with different motor; but they were ineffective to solve the issue. So, finally decision was taken to move on experimenting with another version (AX5203-0000-0210) of the servo drive.

With the new drive, similar problem was not encountered and motion control was found working as it was programmed in the controller. Encoder's feedback was also as expected; after parameterization of the scaling factors it was giving exact position data following the shaft movements. After getting a reliable outcome from this test, it was time to proceed to the integration of the robot with new control system.

4.2 Interfacing the robot

Robot's control hardware combination had basic similarity with the model used for testing. But in this case there were multiple motors to drive and in addition it was also required to interface the robot's sensors with the I/O modules. In figure 4.3, a block diagram of the new control system hardware is shown.

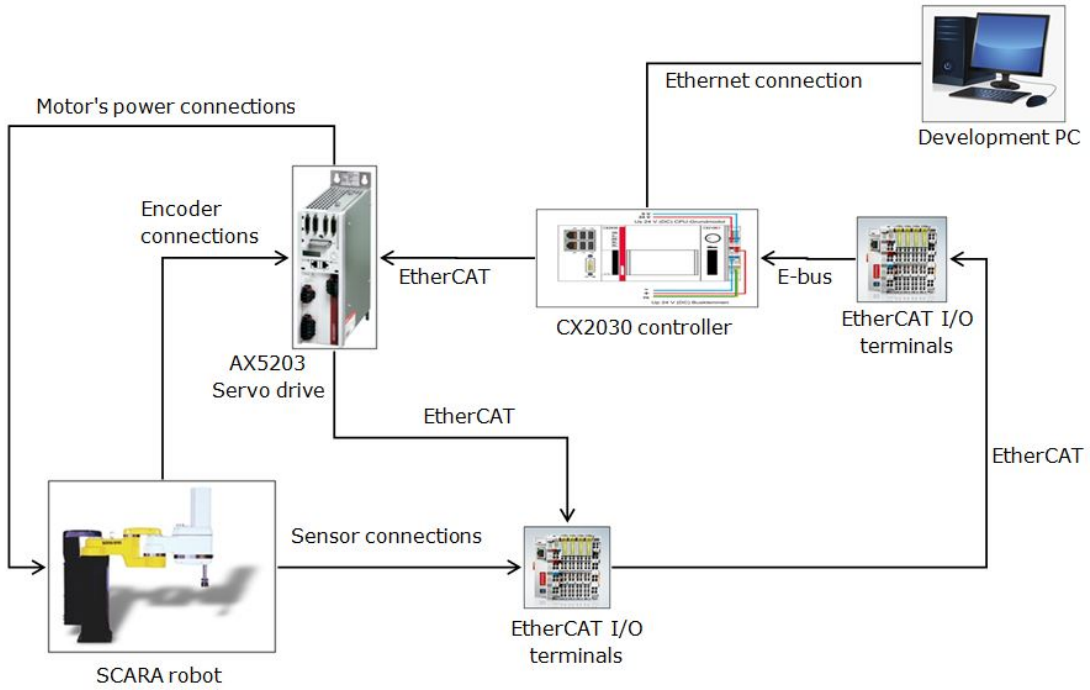


Figure 4.3 Block diagram of the robot's control system

For this new control system, ring network topology was chosen, because of its redundancy advantage, as it was discussed in the chapter 3. Considering an industrial scenario, it was assumed that control hardware and the robot placed at separate location in a factory. As a result of that I/O terminal blocks were divided into two sections, one section was mounted next to the controller itself and the other section was installed near the robot for interfacing the sensors.

Since there was one servo drive available with two drive channels, it was only possible to run two axes of the robot. Axis-A and axis-B was chosen to operate in this case, because of their greater significance in end effector positioning. Axis-A and axis-B motors and encoders were connected with the channel-A and channel-B of the drive respectively. For the power connections X13 and X23 connection ports were used (figure A.2.1). These ports utilize four connecting wires (U, V, W, and PE) for three phase motors. The respective connecting wires in robot's motor line are marked correspondingly as A, B, C, and D for axis-A motor and E, F, G and H for axis-B motor (figure A.1.3).

4.2.1 Encoders connections

The encoder's connection with the servo drive was relatively critical to ensure, as for TTL type encoders, wire break detection is not supported, there was no way to identify wrong combination or disconnections of wires. This connections utilize D-sub 15 pin (DA-15) connector. It was required to manually match and solder each wire from motor's encoder cable to the D-sub plug. As described earlier in chapter 3 (3.1.2) about encoders, TTL incremental type encoders give the output from three sensors with six connecting wires. In addition to these six wires there are four more wires used for U_S (+5 V) and GND (0 V). The wiring configuration of D-sub connector and motor's encoder wires along with TTL encode functions are given in table 4.1.

Table 4.1 Axis-A and axis-B encoder's wiring chart

TTL encoder function	Channel-A encoder connection		Channel-B encoder connection	
	DA-15 connector pin number (X11)	Wire in robot's A/B-axis encoder line	DA-15 connector pin number (X21)	Wire in robot's A/B-axis encoder line
n.c.	1	–	1	–
Gnd 0V	2	X	2	Y
n.c.	3	–	3	–
U_S 5V	4	Z	4	a
B+	5	C	5	N
n.c.	6	–	6	–
Ref Z	7	F	7	S
A+	8	A	8	L
n.c.	9	–	9	–
Gnd 0V (sense)	10	X	10	Y
n.c.	11	–	11	–
U_S 5V (sense)	12	Z	12	a
B-	13	D	13	P
Z	14	E	14	R
A-	15	B	15	M

4.2.2 Sensors connections

As discussed in previous chapter, the sensors installed in the robot are NPN type inductive proximity sensors with normally closed output. This type of sensors sink output to the ground, so the other end of the output need to be positively biased for proper application. On the other hand, digital input terminals (EL1018) used in this work has single wire input channels that is internally grounded. That made it impossible to directly connect the sensors with these input terminals.

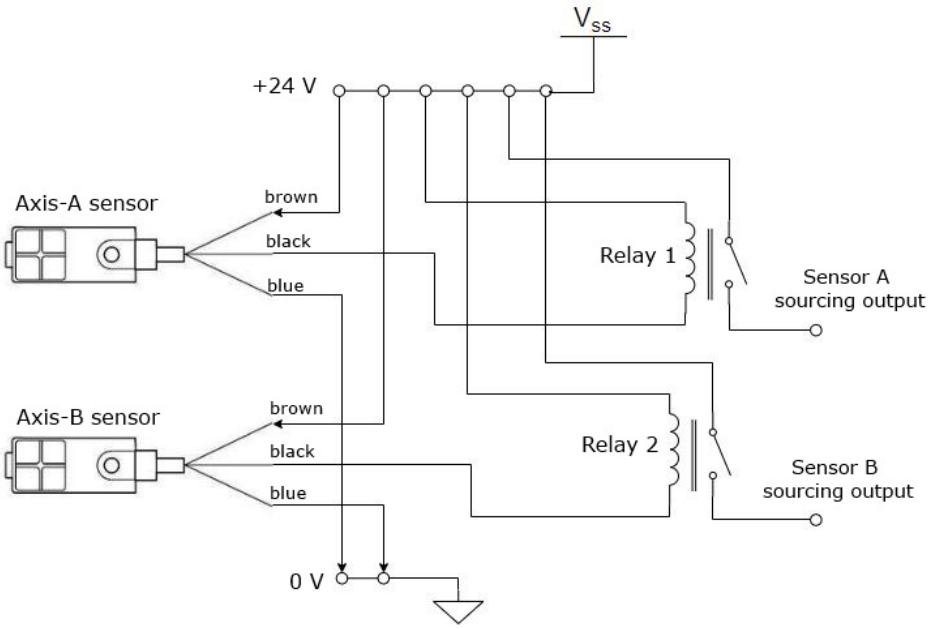


Figure 4.4 Relay circuit used for axis-A and axis-B sensor output conversion

This problem was solved by adding an extra layer of interposing relays between the sensors and the input terminal blocks. In figure 4.4 the circuit of sensors connection is shown. This circuit converts the sensor's sinking outputs into sourcing outputs, which is applicable with EL1018 input terminals. The connections of a three wire sensor are typically marked with colours; brown (+v), blue (-v) and black (output). In this case the corresponding wires in robot's cable are marked respectively as d, b, and G for axis-A sensor and e, c, and T for axis-B sensor (figure A.1.4).

4.3 Motor's parameter identification

Motor parameter identification was an important part in the experiment. As the robot's motors are third party motor, their data files were not readily available. For generating the data files it was required to know certain motor parameters as mentioned in table 3.4. The motors used in the robot are belongs to Panasonic MINAS X series. The original motor's datasheet [33] was useful to identify some of the required parameters. In following table, motor data acquired from the datasheet are shown.

Table 4.2 Motor's Parameters taken from datasheet [33]

Motor Parameter	A-axis motor (MSM042A2UE)	B-axis motor (MSM022A2UE)
Motor's construction (rotary/linear)	Rotary	Rotary
Functional principle (motor type)	Synchronous	Synchronous
Maximum motor speed	5000 rpm	5000 rpm
Number of pole pairs	4	4
Motor peak torque	3,36 Nm	1,91 Nm
Motor continuous stall torque	1,3 Nm	0,64 Nm
Rotor moment of inertia	0,37 * 10^{-4} Kg.m^2	0,17 * 10^{-4} Kg.m^2
Motor continuous stall current	2,5 A	1,6 A
Motor feedback type	Incremental 2500 P/r	Incremental 2500 P/r
Motor temperature sensor type	No temperature sensor	No temperature sensor
Motor brake type	No brake	No brake

Number of pole pairs is calculated using equation 3.1 and the values from table 3.1.

$$\text{Number of poles, } P = \frac{120f}{N_S} = \frac{120*200}{3000} = 8$$

So, the number of pole pairs is 4, that is same for both motors.

Data about winding resistance, winding inductance, Motor's peak current and Back EMF constant were missing in datasheet. These parameters either had to measure manually or to find out from alternative source.

4.3.1 Resistance and inductance measurement

Winding resistance was measures manually by a multimeter. Since the required resistance value was between phases, and as the motor's internal construction is Y connected, this value can be found by measuring resistance from the end of one phase to the next phase. In figure 4.5 the connection of resistance measurement is shown. This measurement was taken between each phases and the average value was used in the motor data. In ideal case, these three values should be equal or very close.

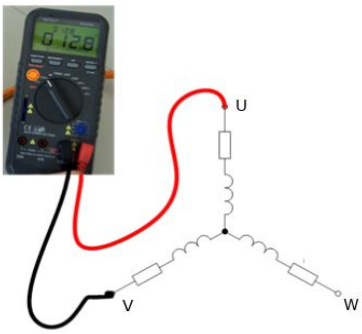


Figure 4.5 Motor's phase to phase resistance measurement [34]

The next required parameter was winding inductance. The most reliable way of measuring inductance is by using LCR meter. In absence of that, it was required to calculate it from electrical measurement values. Following formulas were applied for calculation.

Ohm's law,
$$Z = \frac{V}{I} \tag{4.1}$$

where Z – winding impedance, Ω,
V – phase to phase voltage, V,
I – phase to phase current, A.

Eddy current inspection formula,
$$Z = \sqrt{R^2 + X_L{}^2}$$
$$X_L = \sqrt{Z^2 - R^2} \tag{4.2}$$

where R – phase to phase resistance, Ω,
X_L – inductive reactance, Ω.

Reactance formula,
$$X_L = 2\Pi f L$$
$$L = \frac{X_L}{2\Pi f} \tag{4.3}$$

where f – signal frequency, Hz,
L – winding inductance, H.

In this method it was required to know four parameters; voltage (V), current (I), resistance (R) and frequency (f). Winding resistance was already been measured as described above. Rest of measurement values were taken from the circuit shown in figure 4.6.

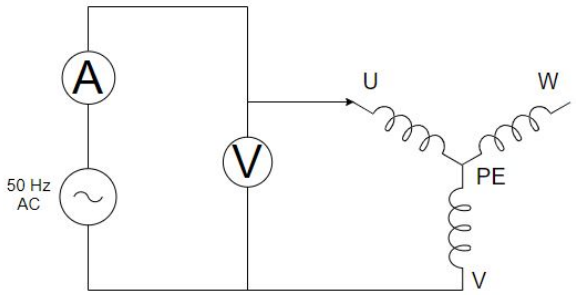


Figure 4.6 Winding inductance measurement circuit

Here the circuit was powered from 50 HZ AC source and the voltage and the current values were taken from the voltmeter and ammeter respectively. This measurements were taken separately for both axis-A motor and axis-B motors. In table 4.3, values taken from the measurements are shown.

Table 4.3 Motor's electrical measurement values

A-axis Motor (MSM042A2UE)				
Phases	Phase to phase resistance (R)	Phase to phase voltage (V)	Frequency (f)	Phase to phase current (I)
U-V	2,56 Ω	4,57 V	50 Hz	0,84 A
V-W	2,56 Ω	4,42 V	50 Hz	1,20 A
W-U	2,53 Ω	4,45 V	50 Hz	1,02 A
Average	2,55 Ω	4,48 V	50 Hz	1,02 A
B-axis Motor (MSM022A2UE)				
Phases	Phase to phase resistance (R)	Phase to phase voltage (V)	Frequency (f)	Phase to phase current (I)
U-V	4,48 Ω	4,77 V	50 Hz	0,55 A
V-W	4,49 Ω	4,81 V	50 Hz	0,59 A
W-U	4,47 Ω	4,66 V	50 Hz	0,80 A
Average	4,48 Ω	4,75 V	50 Hz	0,65 A

A-axis motor inductance calculation:

$$Z = \frac{4,48}{1,02}\,\Omega = 4,39\,\Omega$$

$$X_L = \sqrt{4,39^2 - 2,55^2}\;\Omega = 3,57\,\Omega$$

$$L = \frac{3,57}{2 * \Pi * 50}\,\mathrm{H} = 0,01136\,\mathrm{H} = 11,36\,\mathrm{mH}$$

B-axis motor inductance calculation:

$$Z = \frac{4,75}{0,65}\,\Omega = 7,31\,\Omega$$

$$X_L = \sqrt{7,31^2 - 4,48^2}\ \Omega = 5,78\,\Omega$$

$$L = \frac{5,78}{2 * \Pi * 50}\,\mathrm{H} = 0,01839\,\mathrm{H} = 18,39\,\mathrm{mH}$$

4.3.2 Complete parameter list

Two parameters; maximum current and back EMF constant were still missing for making the motor data files. Motor's maximum current is the parameter that is normally measured by mean of destructive testing. It is done in industries under safety precaution. It was not practical to execute it in this situation. On the other hand, for measuring motor's back EMF constant, it is required a special setup, where the testing motor is driven by another motor at constant speed. This test was also impractical in during the thesis because, the motors were already installed inside the robot, and it is not possible to manually turn the robot's arm in constant speed.

The only way to get these parameters was to find the values from another motor's datasheet with same type of construction and closely similar values. Hiwin AC servo motor datasheet was found useful in this case. The required values for axis-A and axis-B were taken respectively from Hiwin FRLS402 and Hiwin FRLS202 servo motor parameters.

Table 4.4 Motor's complete parameter list [33], [35]

Motor Parameters	A-axis Motor (MSM042A2UE)	B-axis Motor (MSM022A2UE)
Motor's construction (rotary/linear)	Rotary	Rotary
Functional principle (motor type)	Synchronous	Synchronous
Maximum motor speed	5000 rpm	5000 rpm
Number of pole pairs	4	4
Motor peak torque	3,36 Nm	1,91 Nm
Motor continuous stall torque	1,3 Nm	0,64 Nm
Rotor moment of inertia	0,37 * 10^{-4} Kg.m^2	0,17 * 10^{-4} Kg.m^2
Motor continuous stall current	2,5 A	1,6 A
Motor feedback type	Incremental 2500 P/r	Incremental 2500 P/r
Motor temperature sensor type	No temperature sensor	No temperature sensor
Motor brake type	No brake	No brake
winding resistance: phase to phase	2,55 Ω	4,48 Ω
winding inductance: phase to phase	11,36 mH	18,39 mH
Motor peak current	7 A	5,1 A
Motor Back EMF constant	30 mV/rpm	23 mV/rpm

4.3.3 Motor data file creation

From the acquired values shown in table 4.4, the motor data files were created both for axis-A and axis-B motors. 'TC Motor Data File Generator' software was used for making the data files. The desired file format 'AxisInfo(.xml)' was chosen from menu>>Schema. Then after defining the motor types, respective parameters were added in the data files. The data files were then saved in default directory (C:\TwinCAT\3.1\Components\Base\Addins\TcDriveManager\MotorPool). In following figure, parameters of the data files are shown.

Motor source data and calculated data for 'MSM042A2UE_Hirata-AR-S350_A'

Text	Value	Unit
Maximum motor speed (S-0-0113)	5000	rpm
Number of pole pairs (P-0-0051)	4	
Motor EMF (P-0-0055)	30.0	mV/rpm
Motor peak torque (P-0-0073)	3.36	Nm
Motor continuous stall torque (P-0-0070)	1.30	Nm
Mechanical motor data (P-0-0071): Rotor moment of inertia	0.37	*0.0001 kgm^2
Motor peak current (S-0-0109)	7.000	A
Motor continuous stall current (S-0-0111)	2.500	A
Electric motor model (P-0-0066): Winding resistance (phase to phase)	2.55	Ohm
Electric motor model (P-0-0066): Winding inductance (phase to phase)	11.00	mH
Motor warning temperature (S-0-0201)	80.0	°C
Motor shut down temperature (S-0-0204)	140.0	°C
Electrical commutation offset (P-0-0057)	90.00	deg
Thermal motor model (P-0-0062): Time constant 1	1020	s
Thermal motor model (P-0-0062): Warning limit	80	%
Thermal motor model (P-0-0062): Error limit	100	%
Thermal motor model (P-0-0062): Error reaction	1: Shut down on error level	
Motor temperature sensor type (P-0-0061)	4: No motor temperature switch or s...	
Feedback 1 gear numerator (P-0-0152)	1	
Feedback 1 gear denominator (P-0-0153)	1	
Feedback 1 type (P-0-0150): Feedback type string	Unkn#TTL-2500I-5V	

Motor source data and calculated data for 'MSM022A2UE_Hirata-AR-S350_B'

Text	Value	Unit
Maximum motor speed (S-0-0113)	5000	rpm
Number of pole pairs (P-0-0051)	4	
Motor EMF (P-0-0055)	23.0	mV/rpm
Motor peak torque (P-0-0073)	1.91	Nm
Motor continuous stall torque (P-0-0070)	0.64	Nm
Mechanical motor data (P-0-0071): Rotor moment of inertia	0.17	*0.0001 kgm^2
Motor peak current (S-0-0109)	5.100	A
Motor continuous stall current (S-0-0111)	1.600	A
Electric motor model (P-0-0066): Winding resistance (phase to phase)	4.48	Ohm
Electric motor model (P-0-0066): Winding inductance (phase to phase)	18.39	mH
Motor warning temperature (S-0-0201)	80.0	°C
Motor shut down temperature (S-0-0204)	140.0	°C
Electrical commutation offset (P-0-0057)	90.00	deg
Thermal motor model (P-0-0062): Time constant 1	1020	s
Thermal motor model (P-0-0062): Warning limit	80	%
Thermal motor model (P-0-0062): Error limit	100	%
Thermal motor model (P-0-0062): Error reaction	1: Shut down on error level	
Motor temperature sensor type (P-0-0061)	4: No motor temperature switch or sensor (off)	
Feedback 1 gear numerator (P-0-0152)	1	
Feedback 1 gear denominator (P-0-0153)	1	
Feedback 1 type (P-0-0150): Feedback type string	Unkn#TTL-2500I-5V	

Figure 4.7 Motor data file parameters for axis-A motor (up) and axis-B motor (down)

5 COMMISSIONING AND PARAMETERIZATION

In this chapter the steps regarding software commissioning and drive parameterization is described in a chronological order. Later on a homing application is developed and controller is programmed for operation in automatic and manual mode.

5.1 Commissioning

The commissioning process begins with opening a new TwinCAT XAE project in visual studio IDE. TwinCAT project requires to specify the target system, where a communication link is created between the controller and the TwinCAT project. The target controller was added in the project using 'Add Route Dialog' box that appeared after clicking 'Choose Target System'. By pressing broadcast search, TwinCAT shows the list of all devices physically connected with the computer. From this list preferred controller was selected and the route was created by means of device's IP address through Ethernet protocol.

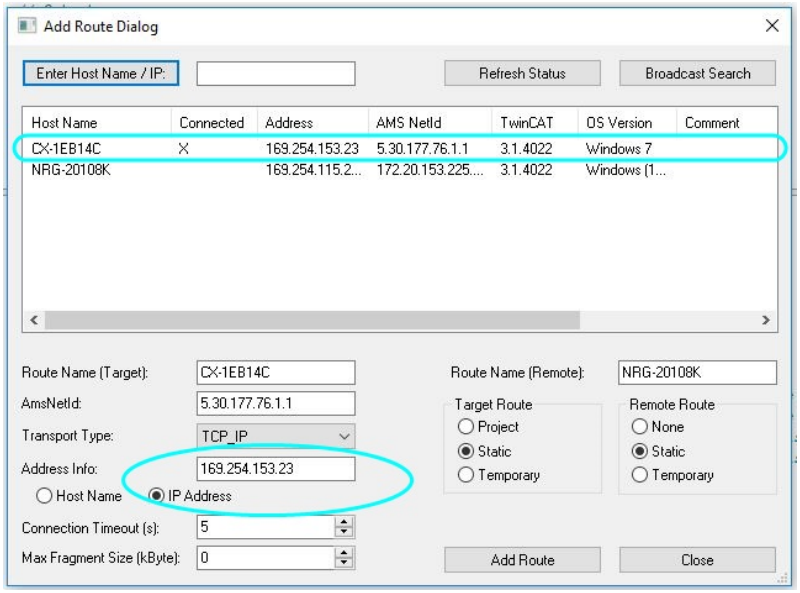


Figure 5.1 Adding route to target system from TwinCAT XAE project

As in this case the target device (CX2030) is a 32 bit system, the solution platform was also required to change to 32 bit. Before adding devices in the project it was also necessary to turn the TwinCAT into 'Config' mode. It was changed and verified from 'TwinCAT XAE Base' toolbar at top of visual studio screen. After that a device scan was

performed from Solution Explorer window. This scan function brings a list of all devices and adapters that is physically connected with the controller. Only the devices related to this work were added in the project.

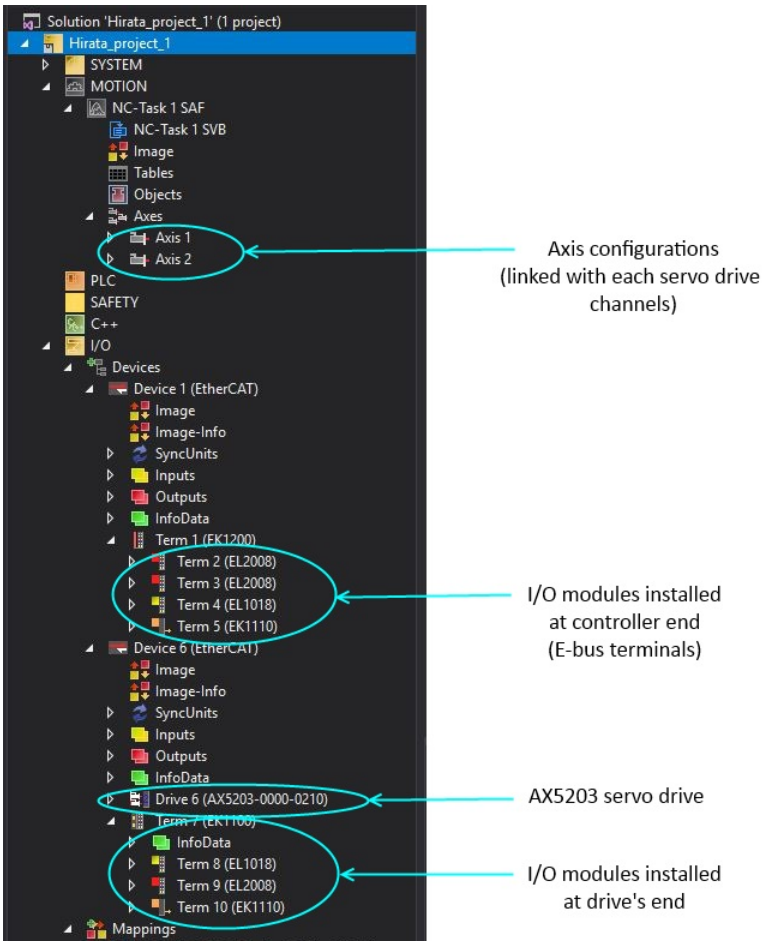


Figure 5.2 TwinCAT system manager tree after performing device scan and creating NC axes

Since, automatic scan also identified the servo drive in the device list, TwinCAT could also scan for motors connected with the drive. This step was skipped, since the motor used in this work are third party motors, they did not have electronic name plates. But TwinCAT needs electronic name plate to identify motors. After the device scan, two NC axis configurations were created, linked with the servo drive's each output channel.

5.1.1 Drive configuration setting

After adding the required devices and NC configurations, the devices were configured. For drive parameterization, the drive manager was opened from solution explorer>>'Drive 6 (AX5203-0000-0210)'. Since the automatic motor scan was skipped, the motor data had to be entered manually. The XML motor data files have already been created explained in chapter 4. Before loading the motor data it was

important to correct the drive's power setting. The power supply setting was chosen '230 V|1 phase|50 Hz (Europe AC)' from device>>power management in drive manager tree.

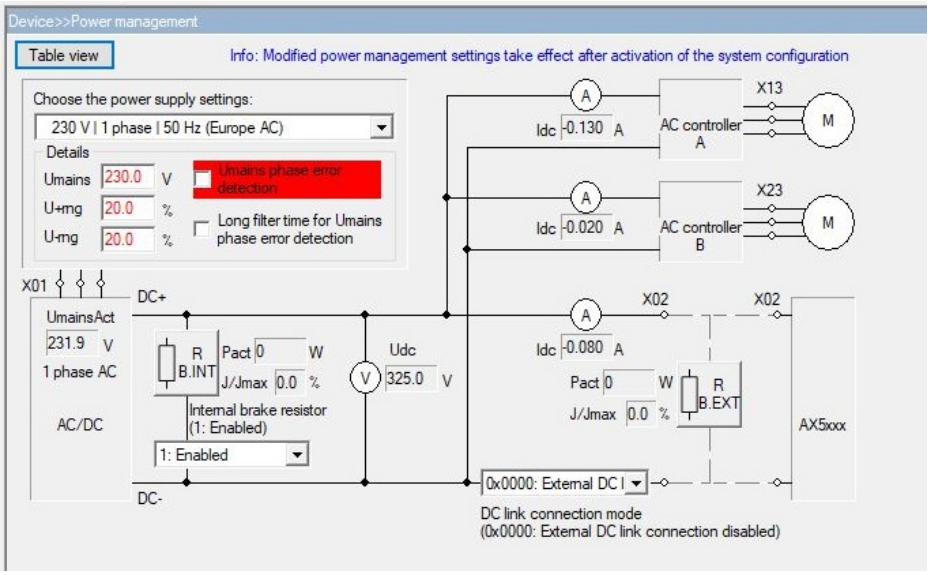


Figure 5.3 Drive manager power configuration screen

After setting the power configurations, 'Motor and Feedback' tab was opened from drive manager tree consecutively for both channel A and channel B. After pressing the 'Select motor' button following screen appeared.

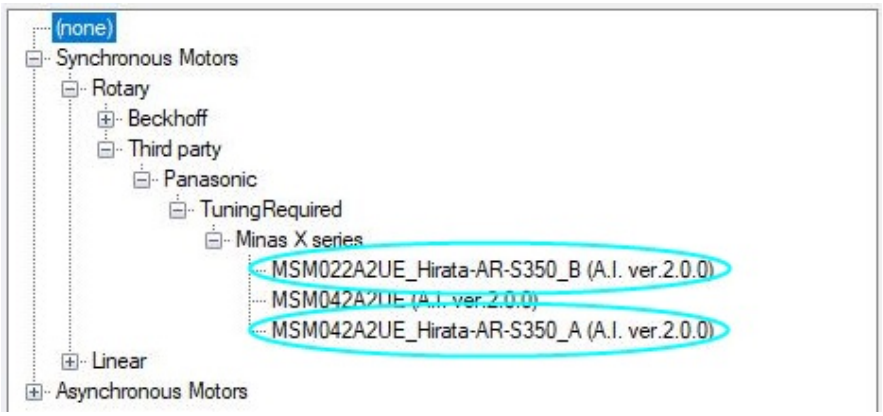


Figure 5.4 TwinCAT's motor selection window

This screen shows all of the motors types that is available in the default motorpool directory. The created motor data files for this project are highlighted in the figure. These two files were added in channel A and channel B in according order. Selecting the motors also added the feedback 1 type in drive manager, since this data was also given during motor data file generation.

5.1.2 NC parameters scaling

Before adjusting the NC scaling parameters, it was required to define the unit of movement in each axis. As both of the robot's operating axes (axis-A and axis-B) give rotational movement, the unit of movement was selected 'Degree' from axis configuration settings. After a thorough analysis of TwinCAT's NC scaling it was found that TwinCAT primarily needs two values called 'feed constant' and 'position resolution' for adjusting scaling parameters. These two numbers are used for calculating the value of 'NC scaling factor'.

$$\text{NC scaling factor} = \frac{\text{Feed constant}}{\text{Position resolution}} \tag{5.1}$$

Position resolution is the bitwise resolution used in controller side to identify the movement position. This number should always be higher than maximum resolution of the encoder, that is the number of total counts in one motor revolution. In this case the resolutions of the encoders are 10000 as explained in chapter 3 (3.1.2). So, position resolution should be higher than 10000. In drive manager the lowest option was 2^20 or 1048576 that is much higher than the required value. So, 2^20 was selected as position resolution.

On the other hand feed constant is an application related parameter that describes the amount of physical movement in one motor revolution. To identify the desired value it was required to examine the robot's physical construction. It was found out that the robot has a gear ratio 1:80 in axis-A and 1:50 in axis-B (table A.1.1 in appendix 1). So, one motor revolution (360 degree) was divided with the gear ratio numbers to find the feed constant values. These resultant values were 4,5 and 7,2 respectively for axis-A and in axis-B.

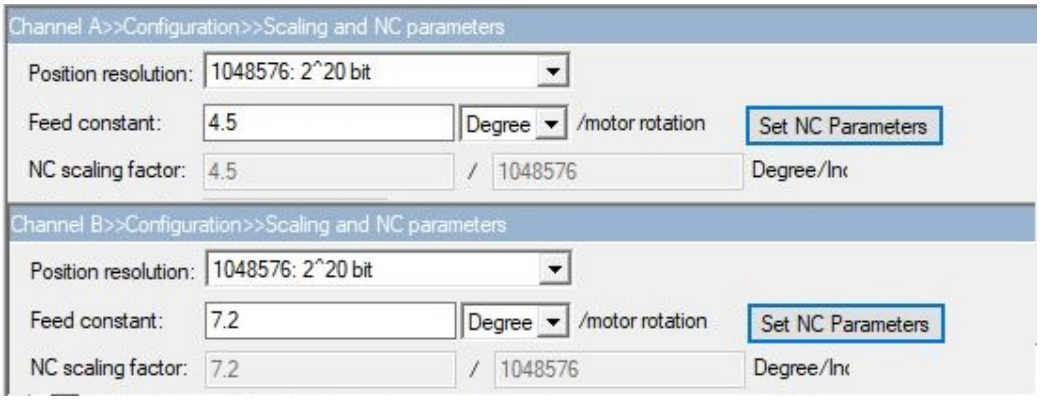


Figure 5.5 Scaling and NC parameter settings for axis-A (up) and axis-B (down)

The values were entered from 'Scaling and NC parameters' setting inside drive manager. It was also important to recheck the values are same in encoder's parameter setting in axis configurations. Both of drive configuration and axis configuration need similar values of scaling factor, since without that encoder's count would give wrong position data. The values of axes' velocity, acceleration and jerk are dependent on scaling factor. So, these values were also updated after defining scaling factor for each axes. Once device configurations and scaling was done, it was time to transfer the settings to the drive. It was done by pressing 'Activate Configuration' button in toolbar. The system took few moments to upload all configurations and then the TwinCAT restarted in run mode.

5.1.3 Phase sequence check and commutation search

Before moving the axes in manual mode it was required to execute few drive commands from drive manager's service functions. These commands were executed in run mode, after enabling the axes; so it caused certain movement and vibration of the axes.

'P-0-0166: Motor and feedback connection check' command was executed for comparing the motor's phase and the encoder's counting directions. Because by default, the drive cannot know whether motor phases are connected in the right sequence or not. In case of a wrong sequence in phase connection, encoder could identify the reverse direction as forward and that would ultimately lead to incorrect counting by the controller. This is avoided by running 'P-0-0166: Motor and feedback connection check' command in drive commands window. Picture shown in figure 5.6.

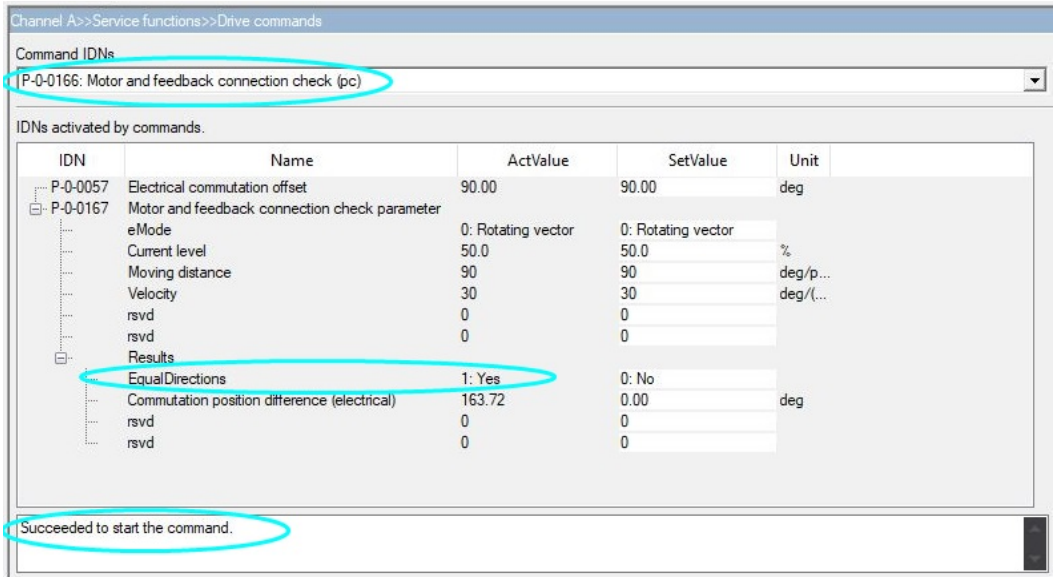


Figure 5.6 Motor and feedback connection check drive command

After execution of the command, the text ' Succeeded to start the command' appeared in screen and values under P-0-0167 were changed. It was important that the 'Equal Directions' active value was '1:Yes'. That ensures motor phases and encoder counting are in same direction. In case of '0:No' in 'Equal Directions' field, it was necessary to swap the connection of any two phases in motor's power line.

'P-0-0160: Calibrate commutation offset' command was used for executing the commutation search for the drive to identify the rotor's exact position. It is required because the drive has to know the correct phase to excite at the time of starting the motors. As in this case the motors are permanent magnet motor, by construction they are brushless type. And it is known, brushless motors uses electronic commutation instead of mechanical brushes. So, it is necessary to find the corresponding stator windings to excite, to keep an electrical commutation offset of 90 for shaft rotation.

This is done with the help of feedback systems in this case the encoders. When executing this command, motor's winding is excited for a while and the rotor makes a bit of movements. Encoder registered that movement data, and the drive calculates rotor' exact position in the magnetic field. This process is useful to avoid unexpected axis alignments when operation begins. This procedure need to follow at least once while commissioning. In case of using absolute encoders position data are stored in the system and it is not necessary to check every time when drive restarts. But in this case, because of using incremental encoders position data gets lost with power off, and commutation search need to be done every time drive restarts.

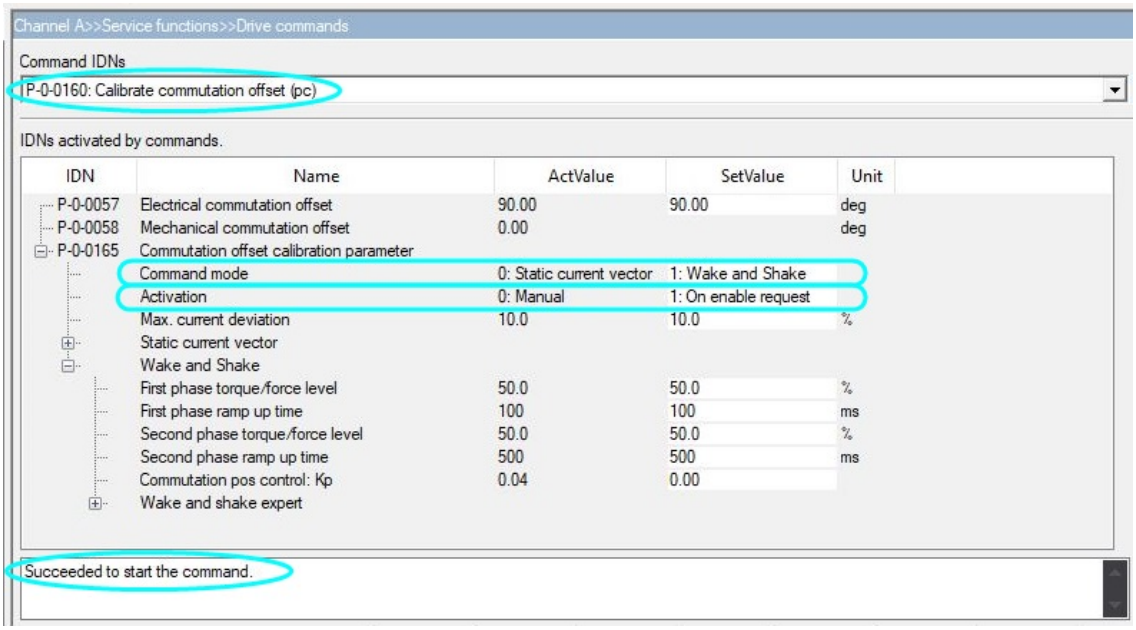


Figure 5.7 Commutation search drive command

P-0-0160 was selected from drive commands list. Under P-0-0165 command mode was selected '1:Wake and Shake', since this mode causes less movement of the axis. Activation was selected '1:On enable request' to automatically run this check after every restart. Executing the command took few seconds and after execution the message 'Succeeded to start the command' appeared at the bottom of screen. The drive was then ready to run in manual mode.

5.2 Homing

Homing refers to the process of axis movements that helps the controller to identify the absolute axis position. Homing is done with the help of a reference signal, that used to be at a known position. When the axis takes certain movements and detects the reference signals, that particular location is registered with the known position value. And following this process the controller identifies the axis actual position in the workplace. In the case of using incremental type encoders, homing is a mandatory procedure when machine starts up, since incremental encoders does not store position values, there is no other way to identify the actual position by the controller.

In this development homing has been done with the help of proximity sensors that is installed with every axes. As explained in chapter 3, these sensors are inductive proximity sensors, with a normally closed output. They give high output when there is no detection, but when they detect the metal tabs connected with axes the state changed to low. By following this detection controller registers that position with a

known position value. From robot's manual it is known that each axes has a maximum working range and that is detectable by the sensors. As a result, in this case the sensors serve as reference signals and the maximum working ranges are the known positions for homing.

In TwinCAT there are several modes of homing procedure, they are categorised by mean of compatibility with position measuring systems or encoders. The TTL-incremental encoders, used in this robot fall under the relative position category. For this category the applicable homing mode is the 'homing based on reference cam' or 'Plc Cam' called in TwinCAT. This mode is relatable with the basic method of homing, which is discussed above. In this mode a digital signal is used as reference at a defined point in travel path. The controller detects the signal edge for allocating the reference position to that position. For other homing modes absolute encoder or part-absolute encoder are required.

5.2.1 Homing parameterization

Homing parameterization was required inside NC configurations>>Encoder under Parameter settings. Reference System was selected 'incremental' as incremental encoders were used. 'Encoder mask' value was kept the original values as 0xFFFFFFFF. This number refers to bit width of encoder counting position and used in range overflow. Since the robots axes works in certain range that falls much below this number, there was almost no chance of range overflow. 'Invert encoder counting direction' was kept false since encoder counting direction complied with direction of axis rotation. 'Reference mode' was selected 'Plc CAM' as explained above it is the most appropriate one for incremental encoders.

Parameter	Offline Value
Encoder Evaluation:	
Invert Encoder Counting Direction	FALSE
Scaling Factor Numerator	4.5
Scaling Factor Denominator (default: 1.0)	1048576.0
Position Bias	0.0
Modulo Factor (e.g. 360.0°)	360.0
Tolerance Window for Modulo Start	0.0
Encoder Mask (maximum encoder value)	0xFFFFFFFF
Encoder Sub Mask (absolute range maximum value)	0x000FFFFF
Reference System	'INCREMENTAL'
Limit Switches:	
Filter:	
Homing:	
Invert Direction for Calibration Cam Search	FALSE
Invert Direction for Sync Impuls Search	TRUE
Calibration Value	-110.0
Reference Mode	'Plc CAM'
Other Settings:	

Figure 5.8 Axis-A homing parameter settings

'Invert direction for calibration cam search' determines the direction of axis rotation for searching reference signal. Standard movement is the negative direction. It was selected false that ensures the axes moves toward negative direction during homing. Calibration values were chosen according to the data about maximum working range of the robot axes (Appendix 1). These values were -110 and -135 respectively for axis-A and Axis-B. Negative values were given, because the axes moves toward negative direction during homing and it refers that, sensor's activation point will be on the negative side.

5.2.2 Program for automatic homing

Since in TwinCAT, homing procedure does not operate in manual control mode it was required to make program for homing application. The program was designed to run in automatic mode by the controller for executing two different type of tasks. In addition to automatic homing, it is also useful for point to point move by jogging function.

This program was written in structured text format using Beckhoff's 'Tc2_MC2' and 'Tc2_NC' libraries in addition to default libraries. These libraries comply with PLCopen standard motion control functions. The copy of this program is added in appendix 4. This program was structure as a state machine diagram. In figure 5.9 the state diagram of this program is shown.

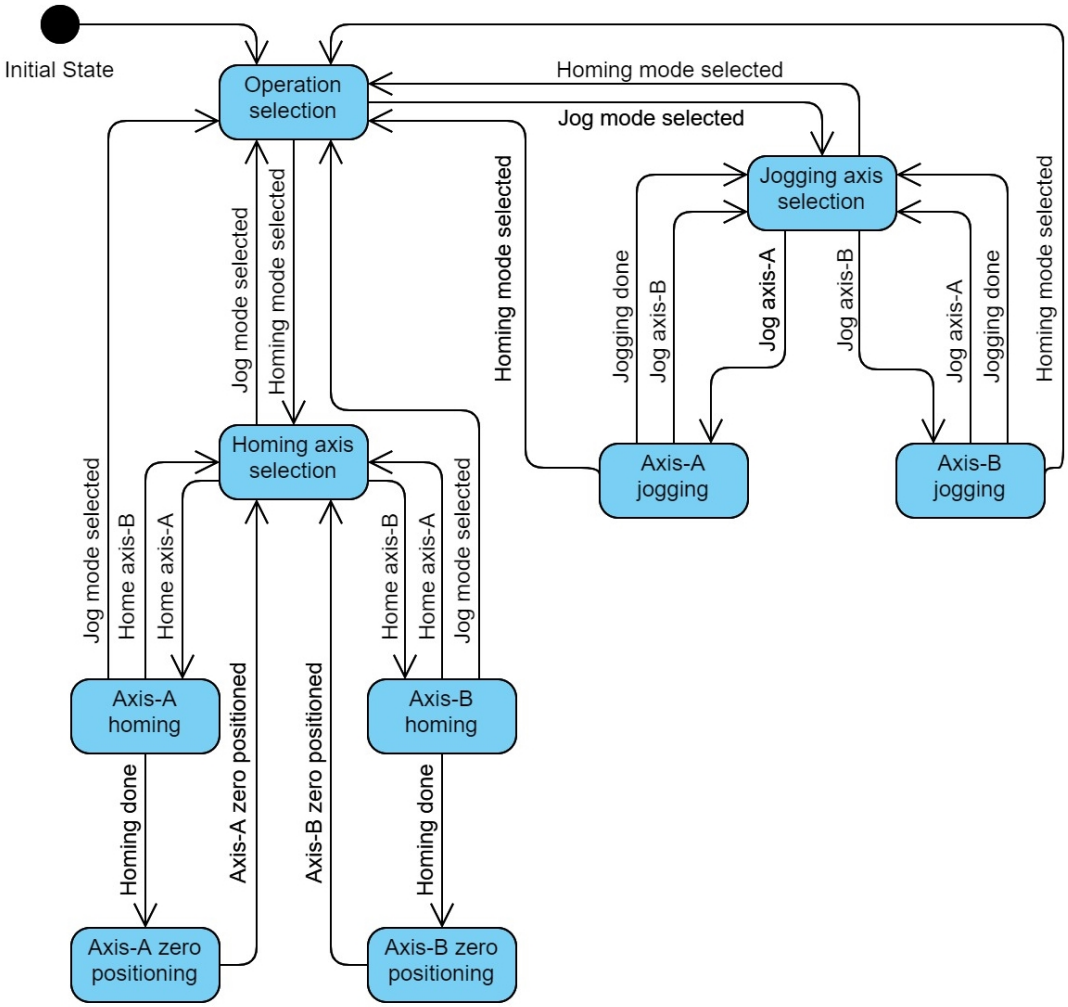


Figure 5.9 State machine diagram of homing and jogging operation

There are ten states in this state machine including the initial state. Three of these states are used for operation selections and axis selections. These selections are made by operator from HMI input. The rest of the states execute the movements by motion functions. Jogging operation is more similar to manual control, that is directly manoeuvred by operator from HMI buttons. And the homing function is entirely autonomous that results automatic movement for axis homing. After completing homing, axes take another automatic move to be placed at zero position. The transitions in this state machine are controlled by input buttons from HMI. In figure 5.10, the HMI of this program is shown.

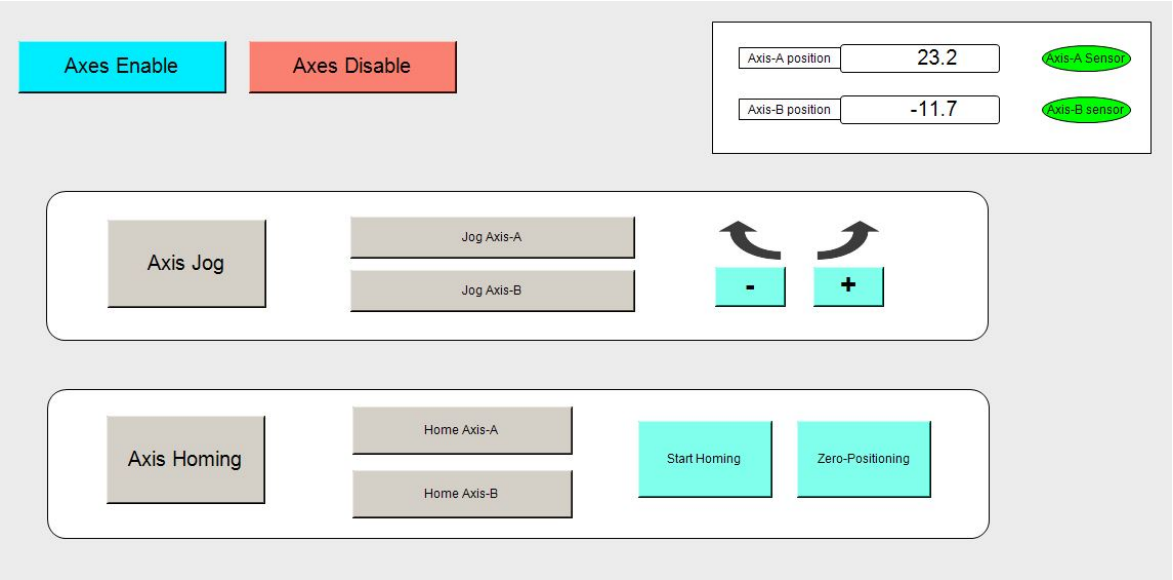


Figure 5.10 HMI for axes homing and axes jogging

In this HMI, the two buttons at upper left corner are dedicated for axis enabling and disabling at any point of operation. At upper right corner there are four parameters to observe; axis-A and axis-B current positions and sensor status. Green colour of sensor indicators imply that sensors are on. In lower part of HMI two separated tabs indicate two different applications. Here the gray buttons are responsible for operation mode selection and axis selection. Aqua coloured buttons cause axis movement either in jog mode or by confirming autonomous movement.

6 TESTING AND ANALYSIS

In this chapter, few tests has been executed and results are analysed for evaluating the robot's movements and the controller's performance. Brief explanation is given about the test parameters and the result are shown in graphs.

6.1 Motion analysis

Robot's axes Motion is an important criteria to test, since almost all of robotic applications are directly related to the accuracy of relevant axes positioning. In this case, few test has been done from the new controller by executing point to point movements. The results of these test are analysed by observing the encoder's position feedback and the received position information is compared with the given commands.

6.1.1 Axis-A motion tests

Three tests has been executed with axis-A in manual mode. Six motion parameters were varied during these tests, to ensure back and forth axis movement with different velocity and acceleration. The testing parameters and the test results are shown in table 6.1 and in figure 6.1 to 6.3 respectively.

Table 6.1 Axis-A motion tests settings

Motion parameters	Test-1	Test-2	Test-3
Initial position	0 degree	10 degree	-10 degree
Target position	10 degree	-10 degree	35 degree
Target velocity	5 degree/s	10 degree/s	25 degree/s
Acceleration	10 degree/s^2	20 degree/s^2	20 degree/s^2
Deceleration	10 degree/s^2	20 degree/s^2	20 degree/s^2
Jerk	200 degree/s^3	40 degree/s^3	50 degree/s^3

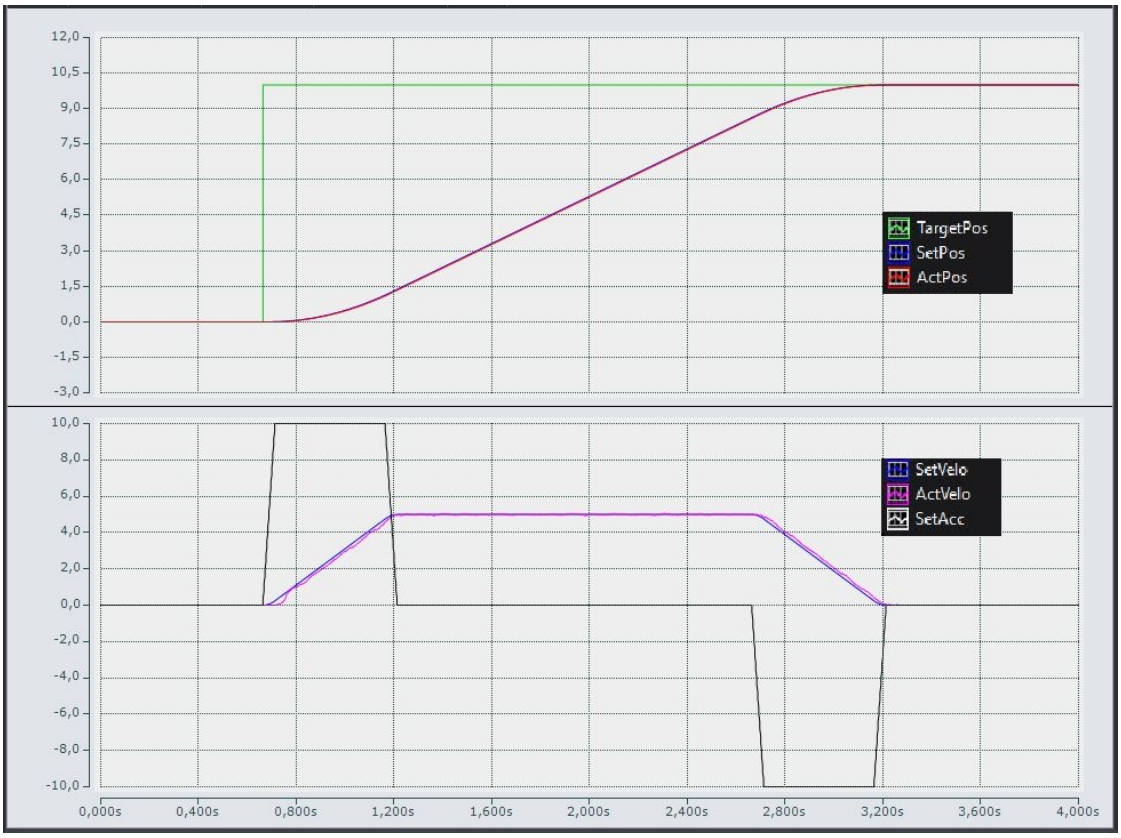


Figure 6.1 Axis-A motion test-1 result

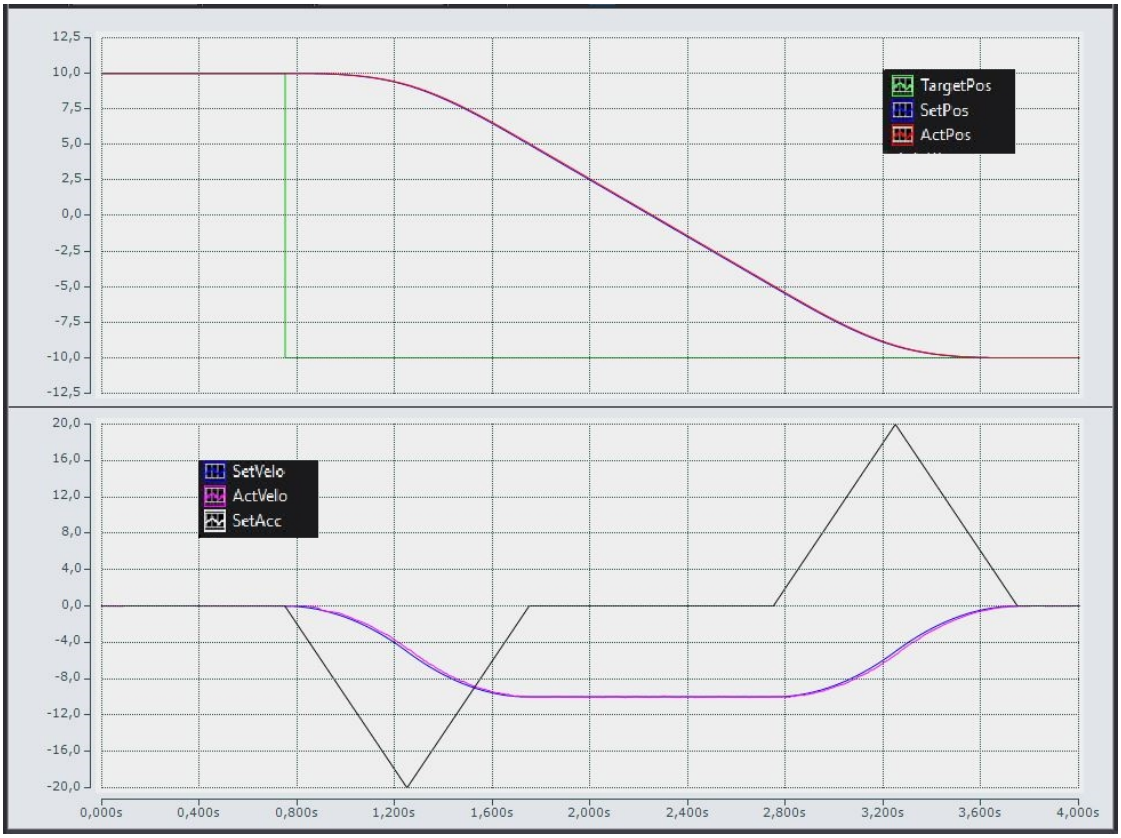


Figure 6.2 Axis-A motion test-2 result

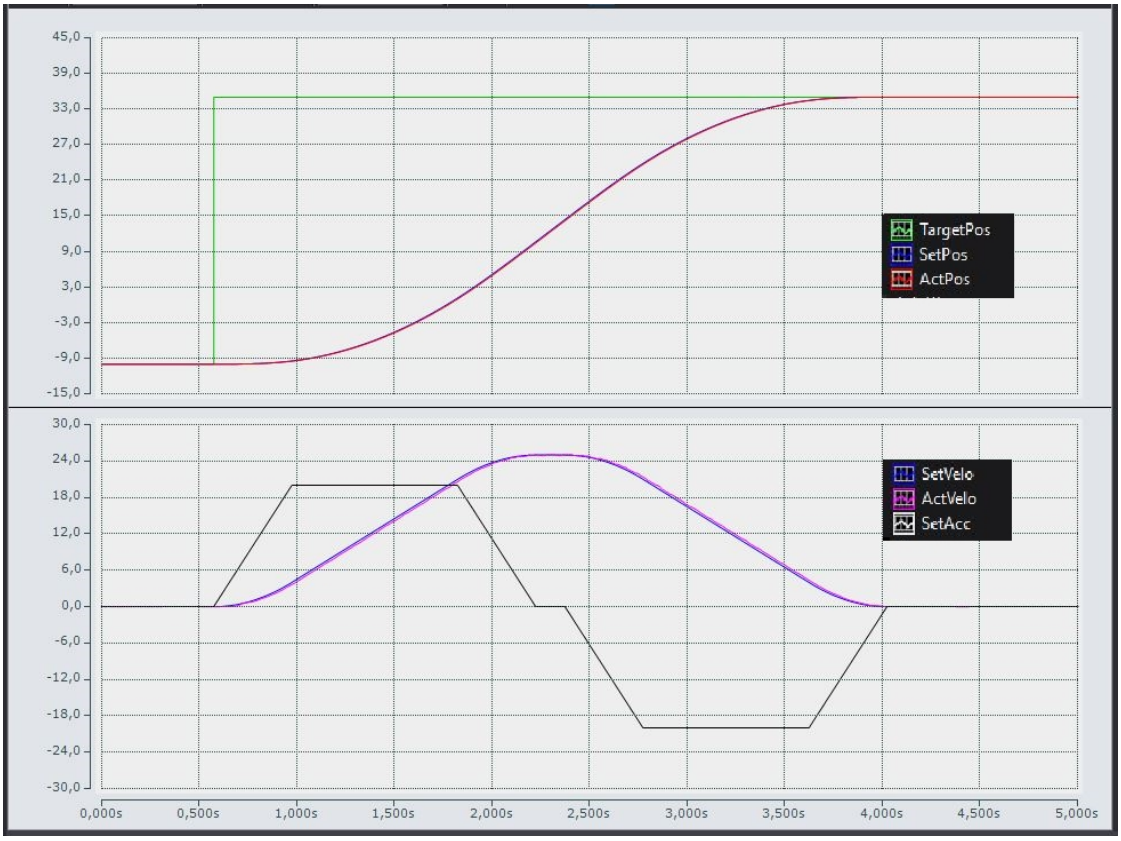


Figure 6.3 Axis-A motion test-3 result

Analysis

From these tests it is visible that the axis reached the target position every time without any overshoot. Here it is significant to observe that, despite of a point to point move, axis is controlled in a continuous mean by generating a new set position in each controller's cycle time. And the set position is always followed by the actual axis position. In the graphs, they are hardly distinguishable as they mostly overlapped.

The velocity and acceleration also followed the commanded values, that it comprehensible from the peak values in the graphs. In test-3 actual velocity did not reach the target value because, the deceleration started at corresponding moment for stabilizing the axis at target position. In these tests, the actual velocity lags the set velocity with an approximate value between 0,2 s and 0,4 s. The possible reason can be the mechanical inertia of the motor and gear assembly.

6.1.2 Axis-B motion tests

Similar to axis-A, three tests has been done for axis-B by varying parameters in manual mode. Test parameters are shown in table 6.2 and results are in figure 6.4 to figure 6.6.

Table 6.2 Axis-B motion tests settings

Motion parameters	Test-1	Test-2	Test-3
Initial position	0 degree	-15 degree	10 degree
Target position	-15 degree	10 degree	60 degree
Target velocity	5 degree/s	15 degree/s	25 degree/s
Acceleration	20 degree/s^2	30 degree/s^2	30 degree/s^2
Deceleration	20 degree/s^2	30 degree/s^2	30 degree/s^2
Jerk	100 degree/s^3	50 degree/s^3	40 degree/s^3

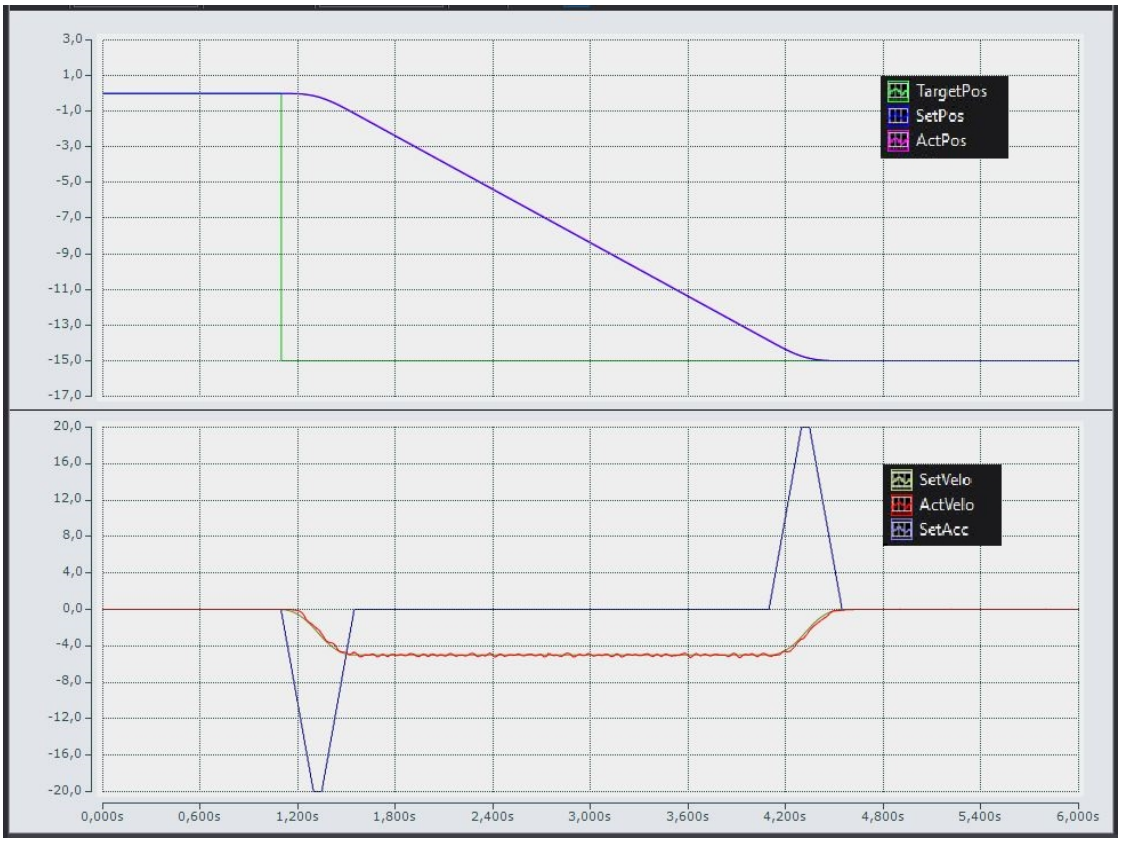


Figure 6.4 Axis-B motion test-1 result

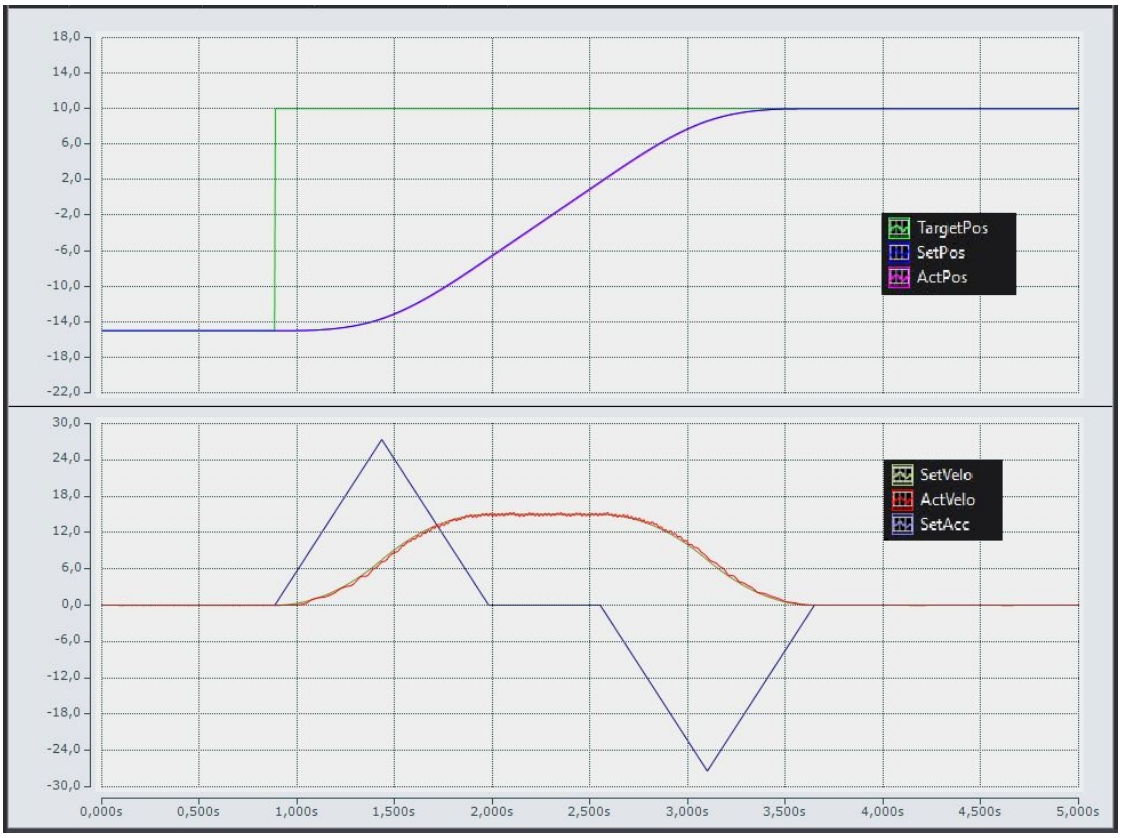


Figure 6.5 Axis-B motion test-2 result

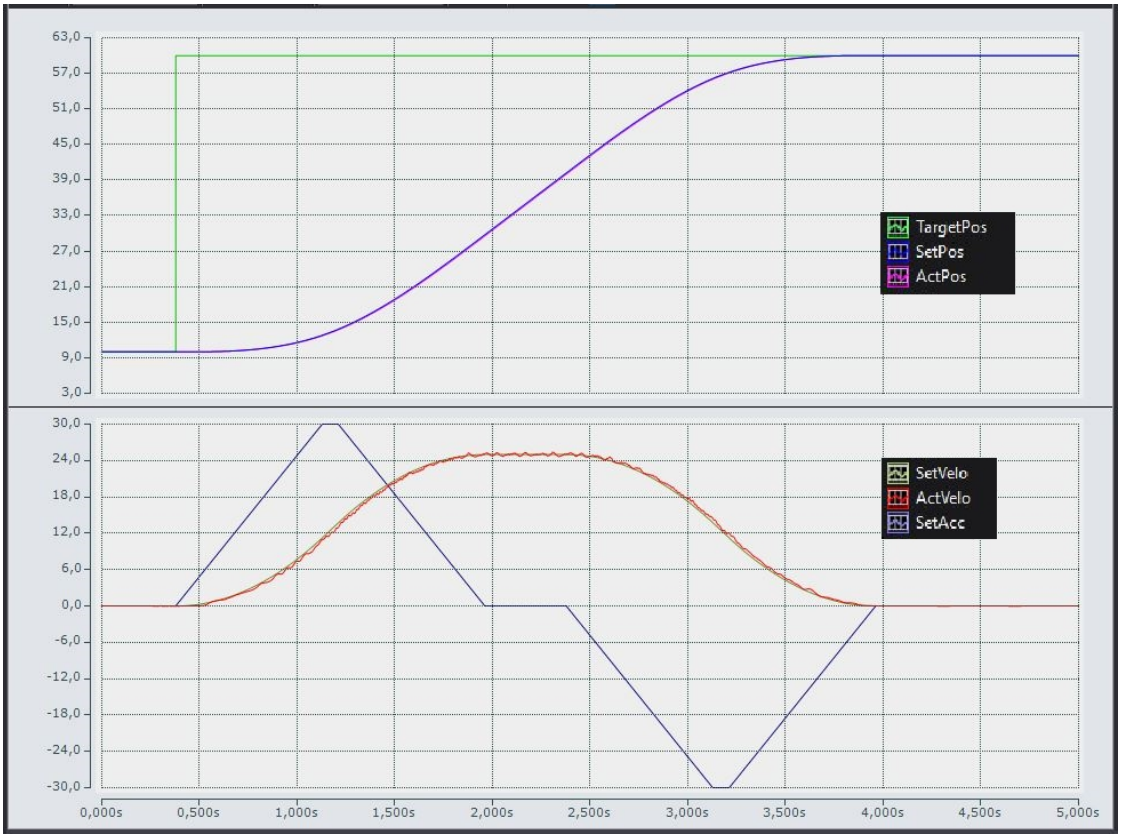


Figure 6.6 Axis-B motion test-3 result

Analysis

In all of these three tests axis attained the target positions without any noticeable overshoot or destabilizations. Actual velocity was in accordance with the target velocity. From the graphs it is visible that, velocity was not totally stable throughout the movement in comparison to axis-A. This might happened because of uneven torque distribution, or any mechanical flaws belongs to motor. In test-2 acceleration did not reach maximum point because, the target velocity was low and the jerk limited the rate of acceleration rise before getting to the maximum point.

6.2 Controller's performance analysis

It is really hard to represent a control system's performance in numerical form. And since the controller used in this thesis is able to run much complicated tasks, this robotic application had very less impact on the controller's overall performance. For analyzing the controller's performance, few parameters have been monitored, while executing the axes motions. In figure 6.7, online monitored data of the controller's system latency is shown.

Figure 6.7 Controller's system latency during task execution

From graph it is seen that, the default system latency was about 1 µs. The peaks in the graph indicates the moments while controller executed any active or passive tasks. And while executing the axes motions there were no significance rise of system latency. As it is observed from the graph, latency was always below 8 µs. In figure 6.8, controller's task execution time is for different tasks shown in time domain.

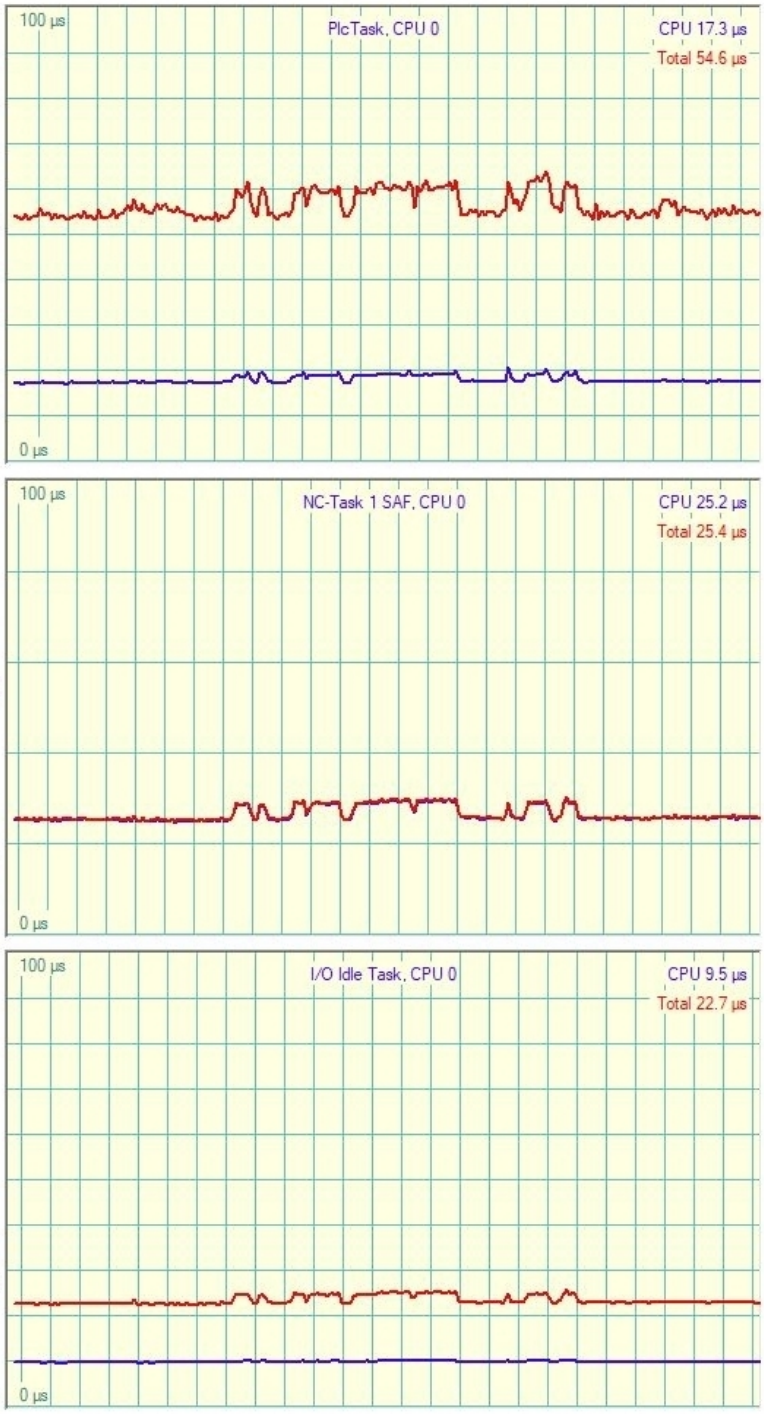


Figure 6.8 Controller's task execution times

'CPU' (blue) graph shows the time required by the CPU for task execution. The 'Total' (red) graph displays the total time spent from starting of the task till the end. The difference of these two lines indicates the waiting time, while CPU was executing higher priority tasks.

In the figure three types of tasks are monitored. 'NC-Task 1 SAF' refers to block execution tasks for NC operations. This tasks include setpoint generations and feeding the fieldbus I/O to NC. The rises in this graph indicates the time when the axes were in motion. In the third there are rises at similar times, since these motion commands were given from manual I/O, they were handeled as I/O tasks. 'Plc Task' graph follows the similar pattern and the maximum task execution time during axis motion was between 60 to 65 µs. And at no motion time it was about 55 µs. NC Task has almost zero waiting time, since it is a higher priority task for CPU.

6.3 Recommendations for future work

Despite being a fully functional system, there is much space of continuing this work for improvements. Since, the inverse and forward kinematics is a primary mean of robotic motion, it is advised to develop this in the future. The currently utilized structured text or function blocks can be used for the kinematic control as well. In addition C/C++ can be utilized for programming, which lot more flexible and object oriented programming language. Alternatively, 'TwinCAT Kinematic Transformation' can be added in future developments. In kinematic transformation there are specific function blocks, that is useful for robotic motion control and SCARA robot's kinematic is also included in this library. For using this feature an additional library name, 'Tc2_NcKinematic Transformation' need to add in solution project.

In addition, for ensuring the full utilization of the robot, it is required to run all four axis of the robot axes. Currently, because of limited number of drive channels, only two axes are used. For full scale movement, this setup needs another similar servo drive or different servo drive with two channels. This is not a technically challenging task, since the entire process is already described in this thesis. Full scale use of this robot will be very useful for making application that is more focused on specific task by running all axes,.

Furthermore, Machine vision can be added in this system by installing cameras to the robot's workstation and using TwinCAT machine vision feature. As, machine vision is extremely important in industrial robotics nowadays, it will be a significant addition to this robot and potentially lead to conduct more comprehensive research on machine vision for SCARA robot controlling.

There are few mechanical issues that need to overlook for this robot's performance improvement. As it is seen from the motion tests, axis-B velocity is not as smooth as axis-A; this issue need to be investigated. Few additional test can be done to this axis with different test parameters. If this symptom exist, then motor data file for the corresponding motor may need to recreate after taking new electrical measurements. Specially for motor's inductance measurement a better method can be followed. It can be either tested with an LCR meter; otherwise the similar method can be followed but using a variable frequency signal generator for powering the circuit at a different frequency.

It has been observed during the experiments that, robot's sensor tabs are not perfectly aligned. As a result, the axes did not take position exactly at centre after homing; even though, according to the NC values they were at zero positions. This can be fixed by physically moving the sensor tabs and placing at right position. It might require to use some manual measurement tools for accurate angule measurement. The another way to solve this, is by assigning manipulated axis position value for homing.

Finally, there is scope of improvement in the safety features of this control system. Although, because of careful operations during experiments there was no unexpected movement occurred. And in case of autonomous operation, correct axis programming and accurate NC parameterization ensured the safe movement within operational range. But it was always important to take extra caution during axis movement and it is advised to developed an extra layer in motion safety. In current setup, the robot can only be immobilized by turning the main power off, for the sake of stopping uncontrolled movement. This can be improved by integrating safe torque off (STO) feature in this system. STO is a quite accepted technique as industrial motion safety measures. It can be added to the system by installing TwinSAFE drive option card to the AX5203 servo drive's safety slot.

7 CONCLUSIONS

This thesis was started with the goal of integrating a new generation control system with a comparatively older generation less utilized robot. The aim was to prove the concept that, replacement and integration of industrial robots with customised control system is a convenient approach. The concept has been proven through experiments with an industrial SCARA robot and control hardware from a different vendor, by developing a common interface for fully functional control system.

The significance of this research topic is justified by a thorough literature search related to the history and evolution of industrial robots. In a sequence of that current trends and international standards about industrial automation and robotics has been explored during this thesis. Later during development stage these knowledge has created a positive influence that ensured the developed control system complies with the current industrial requirements.

This thesis has been done to solve a particular problem in industries. That is flexibility of industrial automation processes specifically about industrial robotics. The new controller developed in this thesis much flexible in terms of integration and modifications compared to the original controller. Beckhoff controller, in addition to TwinCAT automation platform and EtherCAT communication protocol has been used in this development. The advantages these systems has been described in the thesis and verified by experiments.

For making the proper control system the robot itself and the control hardware had been thoroughly investigated for specifications and relevant parameters. Especially, it was challenging to find all necessary parameters belongs to the robot's motors. They have been acquired and calculated from various sources and by following different methods. In addition, necessary electrical circuitry was prepared for proper communication between devices and modules including, sensors, encoders, servo drive, and controller.

After ensuring all necessary steps of proper hardware combination, parameter identifications, and device configurations, it was possible to generate automation tasks from software environment. Program has been written form TwinCAT system using structured text language, to testify basic movements of the robot axes. Finally, the system performance has been observed and analysed from the results found from test runs.

The performance was analyzed under two test categories. One was about motion control accuracy, where robot's actual position, velocity and acceleration were compared with set values. The received results in this test was quite satisfactory, because in almost every test actual parameters have matched with target parameters. The other category was about the controller's performance testing. The result found from this test was relative to conclude, although the controller performed all tasks without any sign of error. In reality, this controller is able to execute much-complicated tasks. For analyzing the true performance level, more complex applications have to be developed in the future based on the current solution.

JÄRELDUS

Selle lõputöö eesmärk oli integreerida uue põlvkonna juhtimissüsteem vanema põlvkonna vähemkasutatud robotiga. Eesmärk oli tõestada kontseptsiooni, et tööstusrobotite asendamine ja integreerimine kohandatud juhtimissüsteemiga on mugav lähenemisviis. Kontseptsioon on tõestatud katsetega tööstusliku SCARA roboti ja juhtimisriistvaraga, mis pärineb teiselt tootjalt, töötades välja ühise liidese täielikult funktsionaalseks juhtimissüsteemiks.

Selle uurimisteema olulisust õigustab põhjalik kirjandusotsing, mis on seotud tööstusrobotite ajaloo ja arenguga. Selle lõputöö käigus on uuritud praeguseid trende ja rahvusvahelisi standardeid tööstusautomaatika ja robootika kohta. Hilisemas arendusetapis on need teadmised loonud positiivse mõju, mis tagas väljatöötatud juhtimissüsteemi vastavuse praegustele tööstusnõuetele.

See lõputöö on tehtud konkreetse probleemi lahendamiseks tööstuses. See on tööstusautomaatika protsesside paindlikkus, eriti seoses tööstusliku robootikaga. Selles väitekirjas on uus kontroller välja töötatud on võrreldes algse kontrolleriga integreerimise ja modifikatsioonide osas palju paindlikum. Selles arenduses on lisaks TwinCAT automaatikaplatvormile ja EtherCAT-i kommunikatsiooniprotokollile kasutatud ka Beckhoffi kontrollerit. Nende süsteemide eeliseid on lõputöös kirjeldatud ja katsetega kontrollitud.

Nõuetekohase juhtimissüsteemi loomiseks oli robot ise ja juhtimisriistvara tehniliste kirjelduste ja asjakohaste parameetrite osas põhjalikult läbi uuritud. Eriti keeruline oli leida kõik vajalikud parameetrid roboti mootoritele. Need on saadud ja arvutatud erinevatest allikatest ja erinevaid meetodeid järgides. Lisaks valmistati ette vajalik elektriskeem nõuetekohaseks suhtluseks seadmete ja moodulite vahel, kaasa arvatud andurid, kooderid, servoajam ja kontroller.

Pärast kõigi riistvara korrektsete kombinatsioonide, parameetrite tuvastamise ja seadme konfiguratsioonide vajalike sammude tagamist oli tarkvarakeskkonnast võimalik genereerida automatiseerimisülesandeid. Programm on kirjutatud TwinCAT-i süsteemist, kasutades struktureeritud tekstikeelt, et tõendada robotitelgede põhilisi liikumisi. Lõpuks on süsteemi jõudlust jälgitud ja analüüsitud katsete tulemuste põhjal.

Tulemusi analüüsiti kahes katsekategoorias. Üks käsitles liikumise juhtimise täpsust, kus võrreldi roboti tegelikku asukohta, kiirust ja kiirendust seatud väärtustega. Selle testi tulemused olid üsna rahuldavad, kuna peaaegu igas testis olid tegelikud parameetrid vastavusse seatud parameetritega. Teine kategooria puudutas kontrolleri jõudluskontrolli. Selle testi tulemus oli lõplik, kuigi kontroller täitis kõiki ülesandeid ilma veamärkideta. Tegelikult on see kontroller võimeline täitma palju keerukaid ülesandeid. Tegeliku jõudluse taseme analüüsimiseks tuleb tulevikus praeguse lahenduse põhjal välja töötada keerukamad rakendused.

LIST OF REFERENCES

[1] B. Singh, N. Sellappan, and P. Kumaradhas, "Evolution of Industrial Robots and their Applications," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 3, no. 5, pp. 763–768, 2013.

[2] A. Gasparetto and L. Scalera, "A Brief History of Industrial Robotics in the 20th Century," *Sci. Res. Publ.*, pp. 24–35, 2019.

[3] M. Boden *et al.*, "Principles of robotics : regulating robots in the real world," *Conn. Sci.*, vol. 29, no. 2, pp. 124–129, 2017.

[4] J. Wallén, "The history of the industrial robot," 2008.

[5] I. Zamalloa, R. Kojcev, and A. Hern, "Dissecting Robotics — historical overview and future perspectives," *Acutronic Robot.*, 2017.

[6] A. Rojko, "Industry 4 . 0 Concept : Background and Overview," *Spec. Focus Pap.*, vol. 11, no. 5, pp. 77–90, 2017.

[7] J. Lapham, "RobotScript ™ : the introduction of a universal robot programming language," *Ind. Rob.*, vol. 26, no. 1, pp. 17–25, 1999.

[8] S. Vaidya, P. Ambad, and S. Bhosle, "Industry 4 . 0 – A Glimpse," in *2nd International Conference on Materials Manufacturing and Design Engineering*, 2018, pp. 233–238.

[9] M. Hermann, T. Pentek, and B. Otto, "Design Principles for Industrie 4 . 0 Scenarios," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 2016, pp. 3928–3937.

[10] ISO, *About ISO/TC 299 Robotics*. 2018.

[11] IEEE, *IEEE 1872-2015 - IEEE Standard Ontologies for Robotics and Automation*. 2015.

[12] ABB, "IRC5 Industrial Robot Controller." [Online]. Available: https://new.abb.com/products/robotics/controllers/irc5. [Accessed: 22-Dec-2019].

[13] KUKA, "KUKA KR C4." [Online]. Available: https://www.kuka.com/en-de/products/robot-systems/robot-controllers/kr c4. [Accessed: 22-Dec-2019].

[14] T. Brogardh, "Robot Control Overview : An Industrial Perspective," *Model. Identif. Control*, vol. 30, no. 3, pp. 167–180, 2009.

[15] H. Bruyninckx, "Open Robot Control Software: the OROCOS project," in *International Conference on Robotics 8 Automation*, pp. 2523–2538.

[16] K.-S. Hong, K.-H. Choi, J.-G. Kim, and S. Lee, "A PC-based open robot control system: PC-ORC," *Robot. Comput. Integr. Manuf.*, pp. 355–365.

[17] D. Karastoyanov and S. Karastanev, "Reuse Industrial Robots," in *IFAC (International Federation of Automatic Control) PapersOnLine*, 2018, pp. 44–47.

[18] I.-K. Jung and S. Lim, "An EtherCAT based Real-time Centralized Soft Robot Motion Controller," *2012 Int. Symp. Instrum. Meas. Sens. Netw. Autom.*, vol. 1, pp. 117–120, 2012.

[19] J. Bohuslava, J. Martin, and H. Igor, "TCP / IP Protocol Utilisation in Process of Dynamic Control of Robotic Cell According Industry 4 . 0 Concept," *2017 IEEE 15th Int. Symp. Appl. Mach. Intell. Informatics*, pp. 217–222, 2017.

[20] "Hirata History - Hirata Engineering Europe." [Online]. Available: http://uk.hirata.de/unternehmen_historie.php. [Accessed: 31-Dec-2019].

[21] Hirata, "BASE ROBOT USER ' S GUIDE AR-S series." Tokyo.

[22] C. JASZCZOLT, "Understanding permanent magnet motors," *Applied Automation supplement for Control Engineering and Plant Engineering*.

[23] "Position Sensors | Types, LVDT, Rotary Encoder," 2019. [Online]. Available: https://www.electronicshub.org/position-sensors/. [Accessed: 22-Nov-2019].

[24] "Micro-size Inductive Proximity Sensor." [Online]. Available: https://elcodis.com/parts/5201212/GXL-8FB.html.

[25] "Manual CX2020, CX2030, CX2040 Embedded PC." Beckhoff.

[26] Beckhoff, "System manual Servo Drives AX5000." 2018.

[27] Beckhoff, "Startup Servo Drive AX5000," vol. 5.9. p. 20, 2018.

[28] Beckhoff, "TwinCAT 3 | eXtended Automation." [Online]. Available: https://www.beckhoff.com/twincat/. [Accessed: 26-Nov-2019].

[29] "EtherCAT Technology Group (ETG)." [Online]. Available: https://www.ethercat.org/en/tech_group.html. [Accessed: 28-Nov-2019].

[30] G. Johnson, "Determinism in industrial ethernet: A technology overview — Part 2." [Online]. Available: https://www.processonline.com.au/content/industrial-networks-buses/article/determinism-in-industrial-ethernet-br-a-technology-overview-part-2-966929628. [Accessed: 28-Nov-2019].

[31] I. Jung and S. Lim, "An EtherCAT based Control System for Human-Robot Cooperation," in *2011 16th International Conference on Methods & Models in Automation & Robotics*, 2011, pp. 341–344.

[32] F. Essler, "EtherCAT topology variations drive system performance." [Online]. Available: https://iebmedia.com/index.php?id=11375&parentid=63&themeid=255&hft=92&showdetail=true&bb=1. [Accessed: 28-Nov-2019].

[33] Panasonic, "MINAS Xseries Digital AC Servo Motor & Driver." .

[34] V. Bobek, "PMSM Electrical Parameters Measurement," 2013. [Online]. Available: https://docplayer.net/6038811-Pmsm-electrical-parameters-measurement.html. [Accessed: 01-Dec-2019].

[35] HIWIN, "AC Servo Motor & D2 Drive Technical Information." Taiwan, 2017.

[32] A. Kruglov, “HIRATA ROBOTI JUHTIMINE LÄBI BECKHOFF PLC,” Tallinn University of Technology, 2017.

APPENDICES

Appendix 1 : Hirata AR-S350 Robot Technical Specifications

Table A.1.1 Hirata AR-S350 robot's physical parameters

Axis	A-Axis	B-Axis	Z-Axis	W-Axis	W-Axis with high inertia (Option)
Motor output	AC400W	AC200W	AC200W	AC100W	
Operational area	220°	270°	200, 350mm	540°	
Maximum speed	294° /sec	384° /sec	1,125mm/sec	1,200° /sec	540° /sec
Gear ratio	1:80	1:50	15mm/rev	1:21	1:46.6666
Repeatability	±0.03mm		±0.03mm	±0.03°	
Maximum compounded speed	5,346mm/sec				
Arm length	First arm: 350mm Second arm: 300mm				
Rated payload	Acceleration/Deceleration 100%: 1kgf ～ 45%: 10kgf (max.)				
Weight	38kgf				

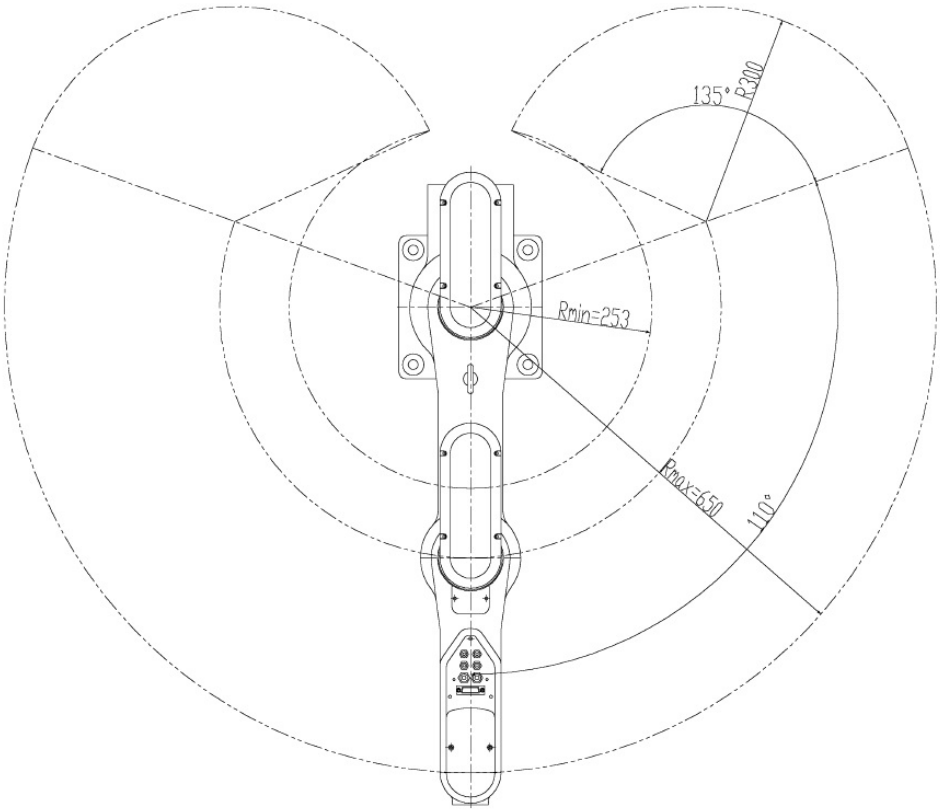


Figure A.1.1 AR-S350 robot's operational area

Table A.1.2 Parts list of AR-S350 robot

01	A-Axis base	14	B-Axis reduction gear	27	Tube cap
02	A-Axis arm	15	B-Axis sensor	28	Tool flange
03	A-Axis motor	16	Z-Axis sensor tab	29	Z-Axis sensor
04	A-Axis reduction gear	17	Z-Axis motor	30	W-Axis motor
05	A-Axis stopper	18	Ball screw spline	31	W-Axis reduction gear
06	Back panel	19	Z-Axis timing pulley	32	W-Axis timing pulley
07	External cable	20	Z-Axis timing pulley	33	W-Axis timing pulley
08	Multi cover	21	Z-Axis timing belt	34	W-Axis timing belt
09	A-Axis sensor	22	Z-Axis brake	35	W-Axis tension bolt
10	A-Axis sensor tab	23	Z-Axis tension bolt	36	W-Axis sensor
11	Transportation eye bolt	24	Z-Axis sensor tab Upper limit stopper	37	W-Axis sensor tab
12	B-Axis arm	25	Z-Axis lower stopper		
13	B-Axis motor	26	Tube		

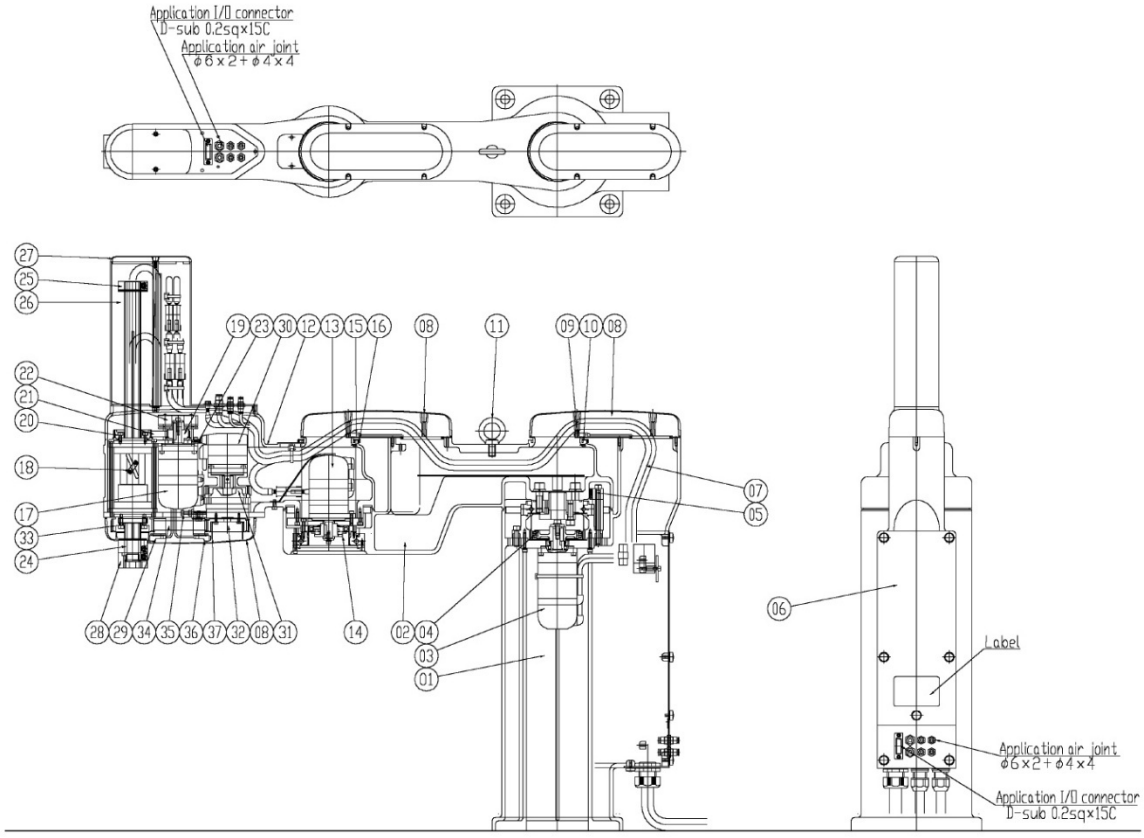


Figure A.1.2 AR-S350 robot's structure

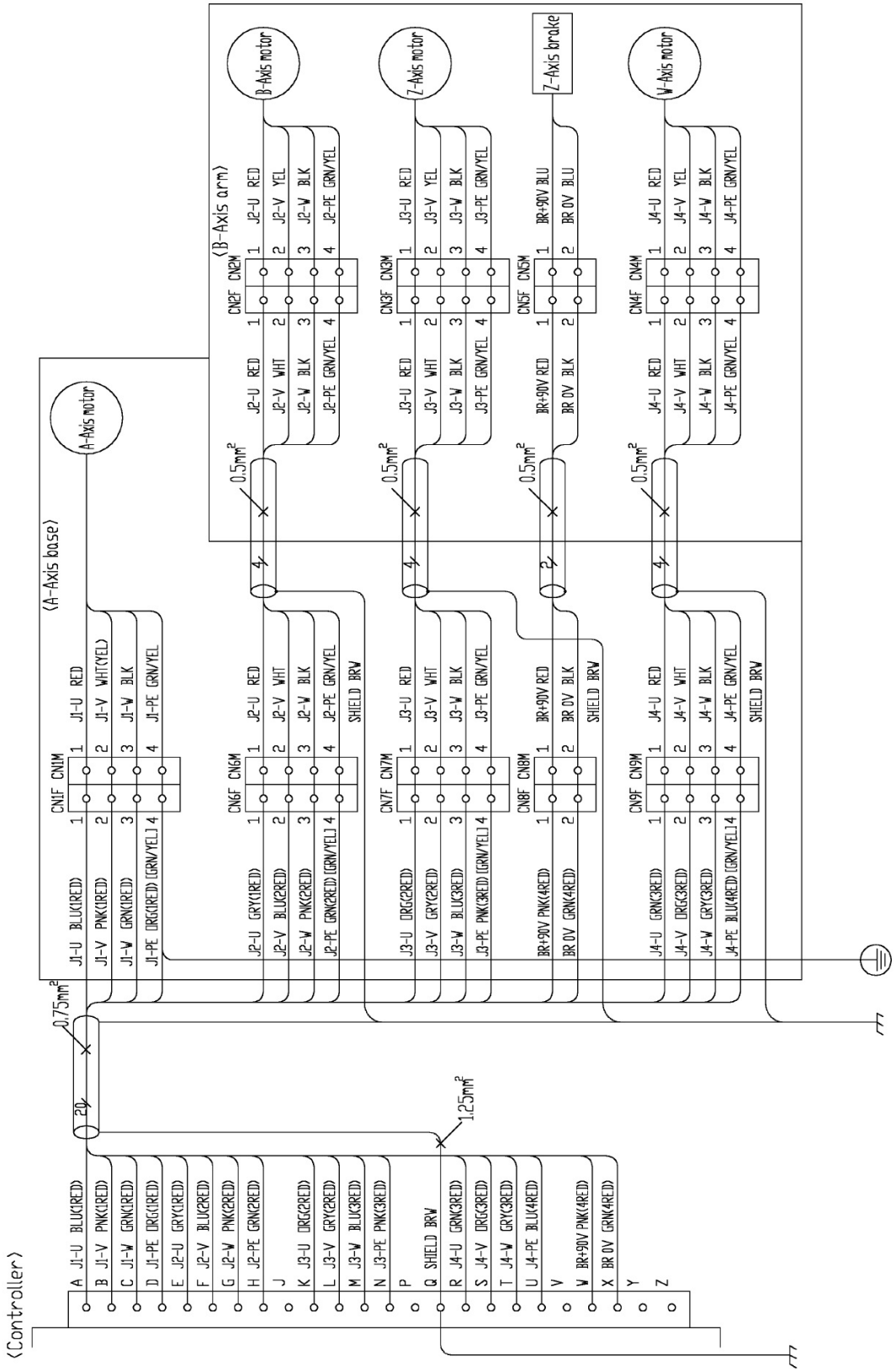


Figure A.1.3 Motor's wiring diagram for AR-S350 robot

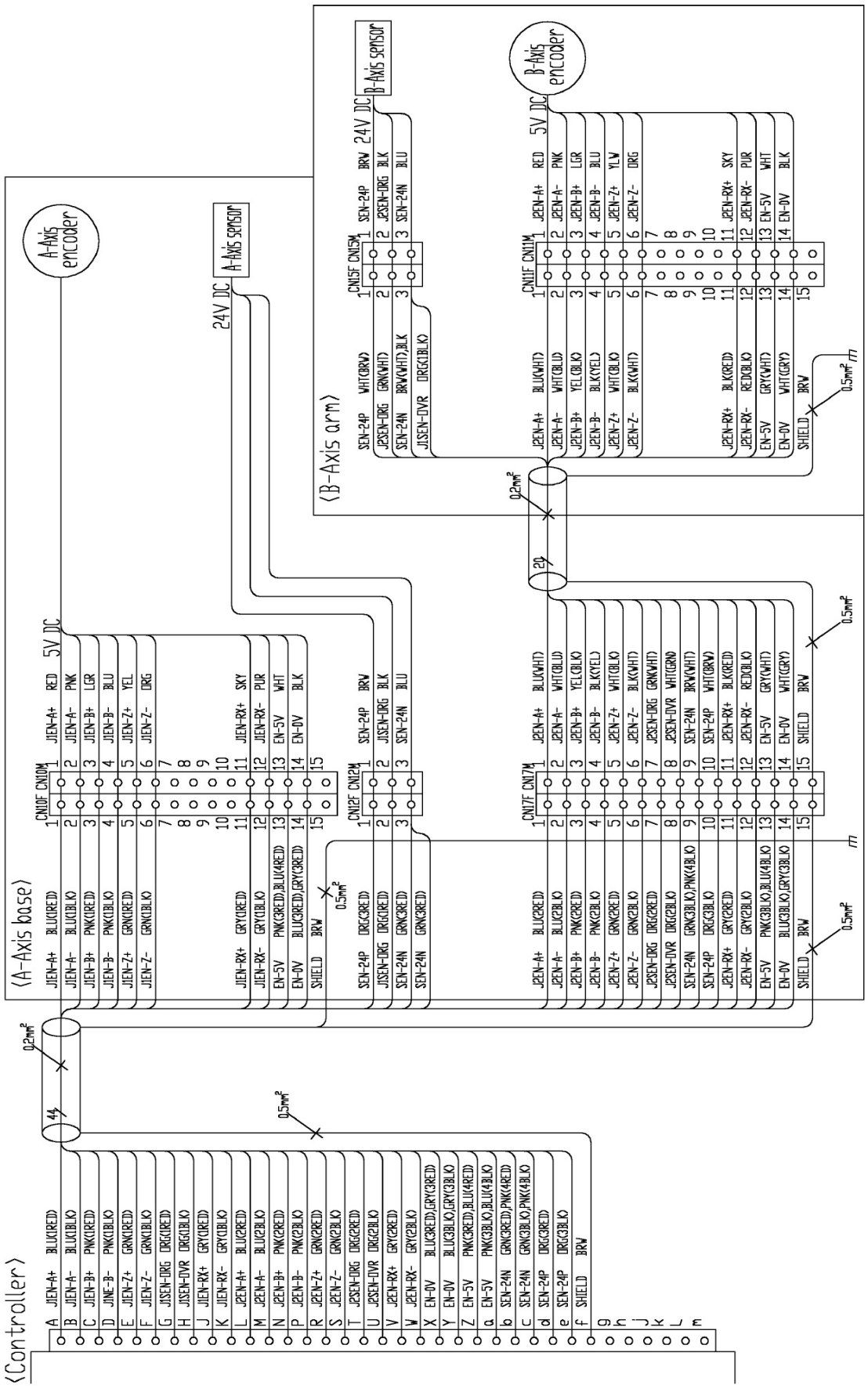


Figure A.1.4 Wiring diagram for A/B axis encoders

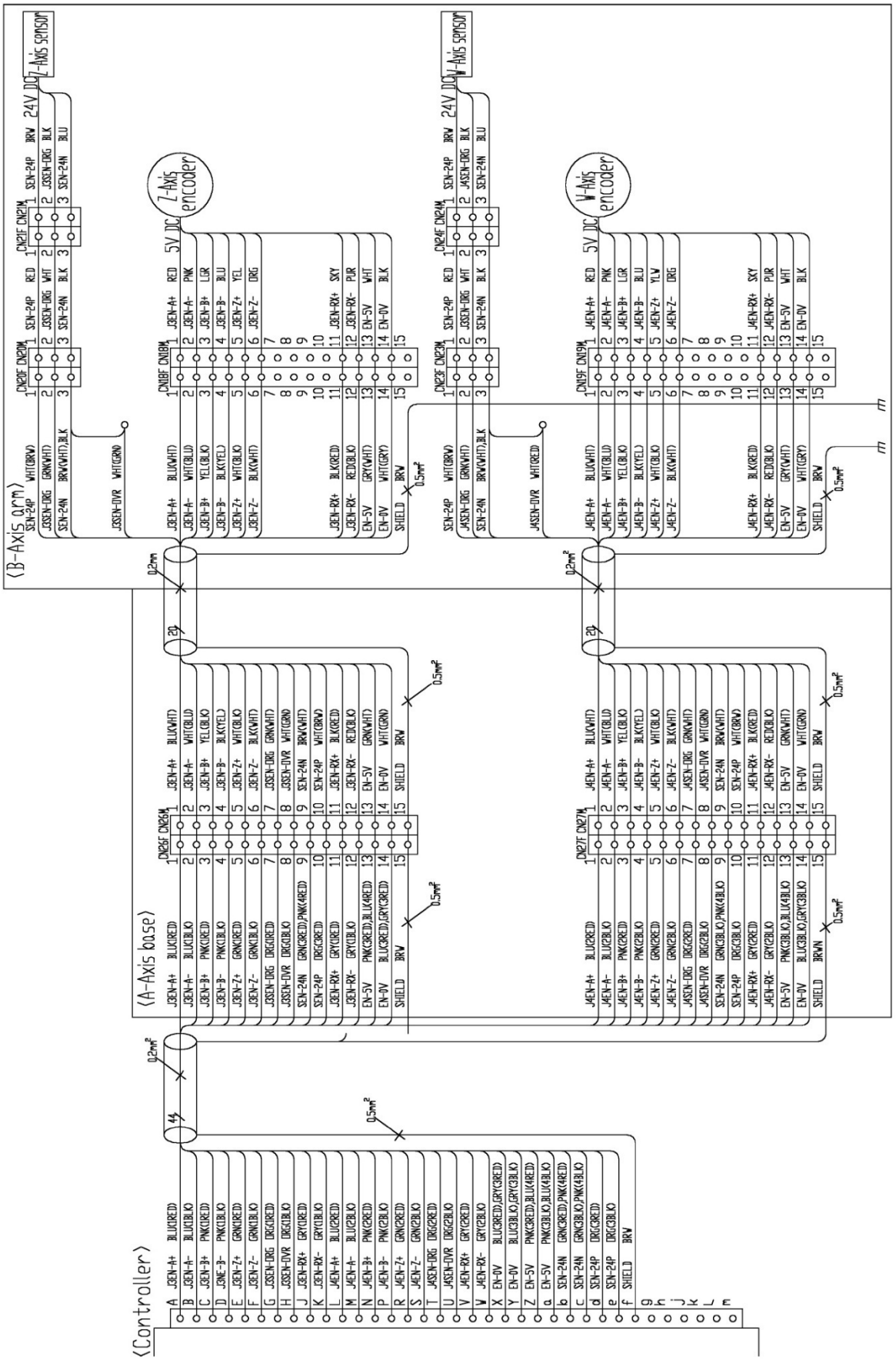


Figure A.1.5 Wiring diagram for Z/W axis encoders

No.	Name	
1	X11 - feedback connection, encoder	
2	X12 - feedback connection, resolver	
3	X21 - feedback connection, encoder channel B (only for two-channel unit)	
4	X22 - feedback connection, resolver channel B (only for two-channel unit)	
5	X3x - optional slot for safety card X4x - optional slot for expansion cards	
6	Navigation rocker	
7	Status LED for EtherCAT output	
8	Labelling field	
9	X05 - socket for EtherCAT output	
10	X03 - power supply 24 V DC input	
11	X14 – sensor for motor temperature, brake and OCT	
12	X24 – sensor for motor temperature, brake and OCT channel B (only for two-channel unit)	
13	X23 - motor connection (U, V, W, PE) channel B (only for two-channel unit)	
14	X13 - motor connection (U, V, W, PE)	
15	X01 - mains supply 100 - 480 V	
16	X02 - DC link output (max. voltage 875 V DC) Connection for the external brake resistor	
17	DANGER	Max. voltage 875 V DC at the DC link terminal points (X02). Once the device has been switched off dangerous voltage will still be present for a further 5 minutes. The device is safe once the voltage has fallen below 50 V.
18	X04 - socket for EtherCAT input	
19	Labelling field	
20	Status LED for EtherCAT input	
21	Display	
22	X06 - connection for digital inputs and outputs	

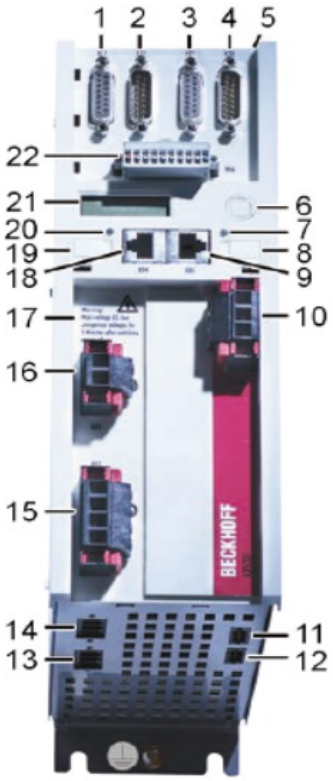


Figure A.2.1 AX5203 visual description [26]

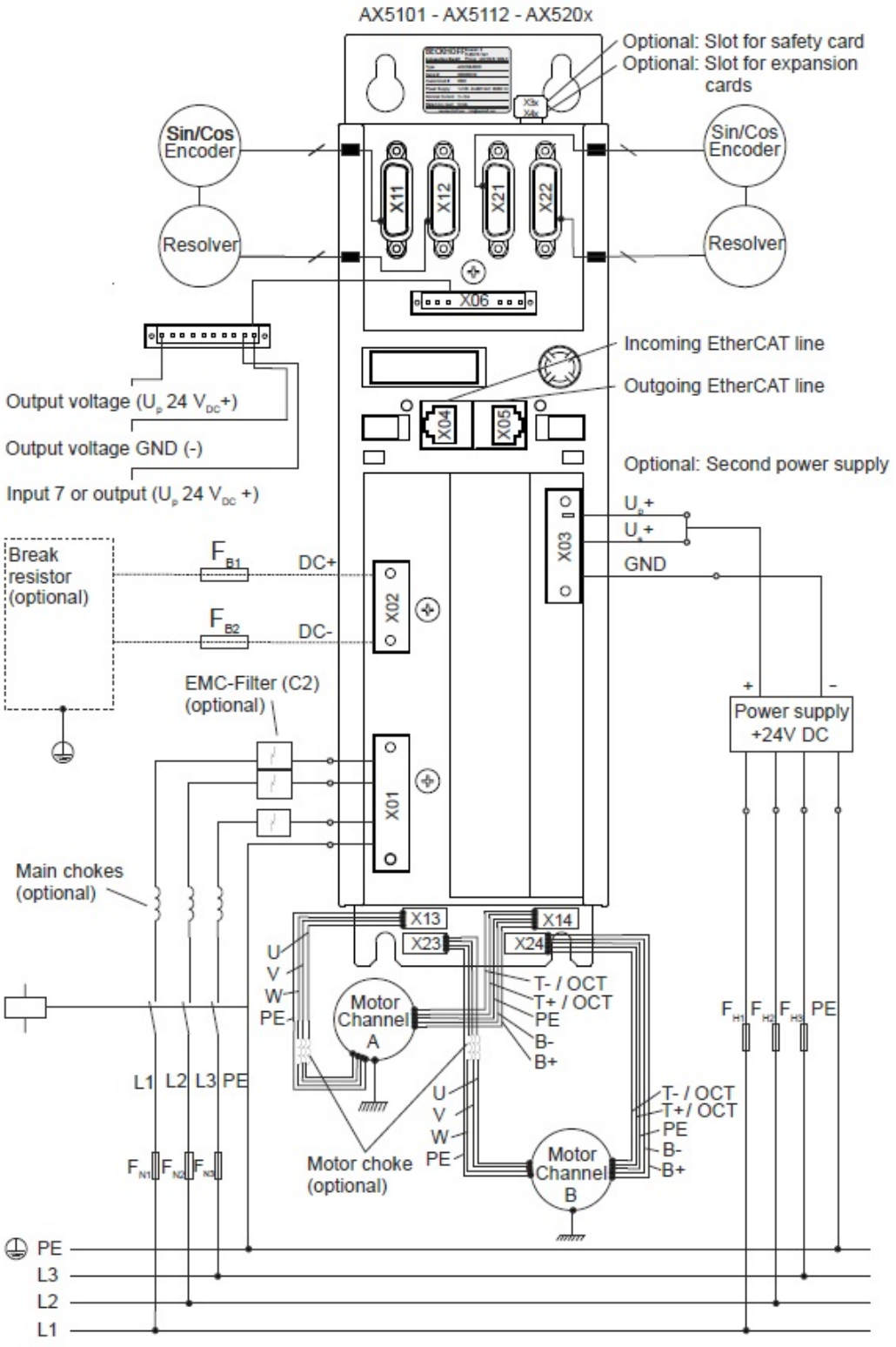
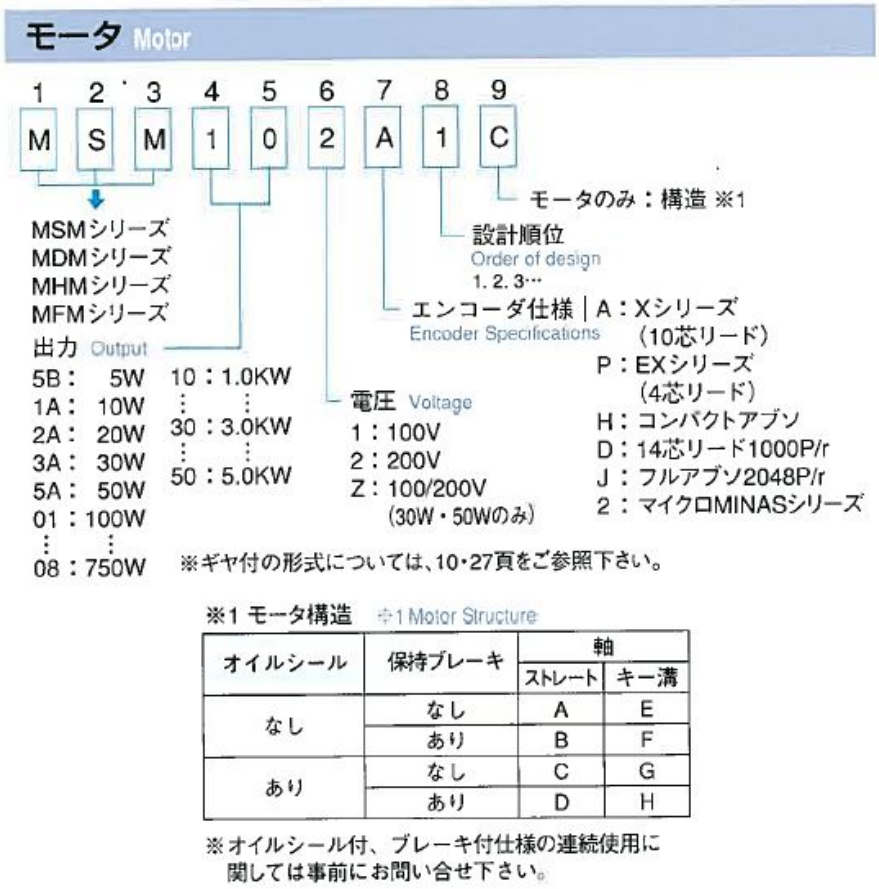


Figure A.2.2 AX5203 servo drive connection example [26]



モータ Motor

1	2	3	4	5	6	7	8	9
M	S	M	1	0	2	A	1	C

MSMシリーズ
MDMシリーズ
MHMシリーズ
MFMシリーズ

出力 Output
5B： 5W
1A： 10W
2A： 20W
3A： 30W
5A： 50W
01：100W
⋮
08：750W

10：1.0KW
⋮
30：3.0KW
⋮
50：5.0KW

電圧 Voltage
1：100V
2：200V
Z：100/200V
(30W・50Wのみ)

エンコーダ仕様 Encoder Specifications
A：Xシリーズ (10芯リード)
P：EXシリーズ (4芯リード)
H：コンパクトアブソ
D：14芯リード1000P/r
J：フルアブソ2048P/r
2：マイクロMINASシリーズ

設計順位 Order of design
1. 2. 3…

モータのみ：構造 ※1

※ギヤ付の形式については、10・27頁をご参照下さい。

※1 モータ構造 ※1 Motor Structure

オイルシール	保持ブレーキ	軸	
		ストレート	キー溝
なし	なし	A	E
	あり	B	F
あり	なし	C	G
	あり	D	H

※オイルシール付、ブレーキ付仕様の連続使用に関しては事前にお問い合せ下さい。

Figure A.3.1 Panasonic MINAS X series motor classifier [33]

		MSM シリーズ(ローイナーシャ)					
定格出力(KW) Rated output		0.03	0.05	0.1	0.2	0.4	0.75
トルク Torque (N・m)	定 格 Rated	0.095	0.16	0.32	0.64	1.3	2.4
	最 大 Max.	0.28	0.48	0.95	1.91 (1.70)注1	3.36	6.9
ロータイナーシャ Rotor inertia ($\times10^{-4}$kg・m^2)	ブレーキなし W/0 brake	0.018	0.027	0.063	0.17	0.37	1.33
	ブレーキ付 W/brake	0.022	0.032	0.067	0.20	0.4	1.41
回転速度 Speed (r/min)	定 格 Rated	3000					3000
	最 高 Max.	5000					4500
保持ブレーキ Brake		DC24V電源(別途ご準備下さい) 仕様についてはP19頁をご参照下さい Power source of DC24V. (prepared by customer) Refer to P19 for specification.					
エンコーダ Encoder		インクリメンタル2500P/r、アブソリュート2048P/r Incremental 2500P/r, Absolute 2048P/r					
使用周囲条件 Ambient Condition		温度(使用時)：0～40℃ Working temperature 0～40℃		湿度(使用、保存時)：85%RH以下(結露なきこと) Working & Storage humidity 85% or less (free from dew)			振動：49m/s^{2}以下 Vibration：4.9m/s^2 or less

Figure A.3.2 Panasonic MINAS X series motor data [33]

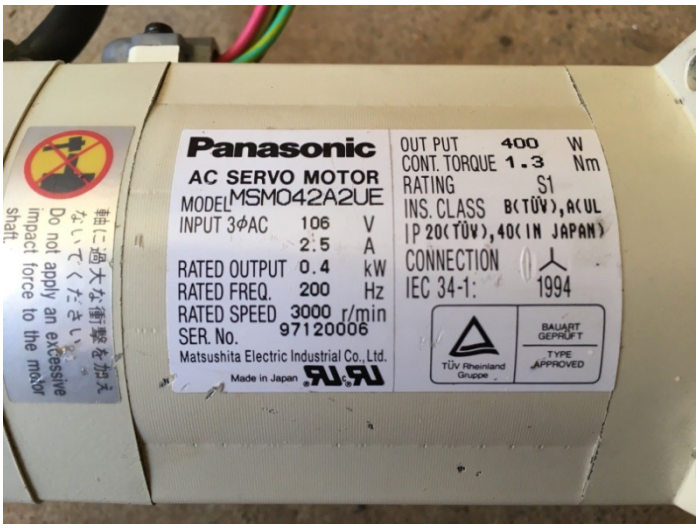


Figure A.3.3 Panasonic MSM042A2UE motor's nameplate

Figure A.3.4 Panasonic MSM022A2UE motor's nameplate

Figure A.3.5 Panasonic MQMZ012A2U motor's nameplate

Appendix 4 : Structured Text Program for Axes Homing

```
PROGRAM MAIN
VAR
        power_button AT %I* : BOOL;
        stop_button AT %I* : BOOL;
        button AT %I* : BOOL;

        jog_mode_btn AT %I* : BOOL;
        hom_mode_btn AT %I* : BOOL;
        opr_mode_select AT %I* : INT;

        jog_axis_A_btn AT %I* : BOOL;
        jog_axis_B_btn AT %I* : BOOL;
        jogging_axis_select AT %I* : INT;

        hom_axis_A_btn AT %I* : BOOL;
        hom_axis_B_btn AT %I* : BOOL;
        homing_axis_select AT %I* : INT;

        axis_move_forward AT %I* : BOOL;
        axis_move_backward AT %I* : BOOL;

        homing_start_btn AT %I* : BOOL;
        Zero_pos_btn AT %I* : BOOL;

        sens_A AT %I* : BOOL;
        sens_B AT %I* : BOOL;

        Axis_A : AXIS_REF;
        Axis_B : AXIS_REF;

        Power : MC_Power;
        Reset : MC_Reset;
        Jogging : MC_Jog;
        McVelocity : MC_MoveVelocity;
        Stop : MC_Stop;
        Homing : MC_Home;
        AbsoluteMove : MC_MoveAbsolute;

        State : DINT;
```

```
END_VAR

Axis_A.ReadStatus();
Axis_B.ReadStatus();

IF power_button THEN
Power(Axis:=Axis_A,
                Enable:=TRUE,
                Enable_Positive:=TRUE,
                Enable_Negative:=TRUE);
Power(Axis:=Axis_B,
                Enable:=TRUE,
                Enable_Positive:=TRUE,
                Enable_Negative:=TRUE);
Reset(Axis:=Axis_A);
Reset(Axis:=Axis_B);
END_IF

IF stop_button THEN
Stop(Axis:=Axis_A,
                Execute:=TRUE);
Stop(Axis:=Axis_B,
                Execute:=TRUE);
END_IF

////Mode selection buttons////
        IF jog_mode_btn THEN
                opr_mode_select := 1;
        ELSIF hom_mode_btn THEN
                opr_mode_select := 2;
        END_IF

        IF jog_axis_A_btn THEN
                jogging_axis_select := 1;
        ELSIF jog_axis_B_btn THEN
                jogging_axis_select := 2;
        END_IF

        IF hom_axis_A_btn THEN
                homing_axis_select := 1;
        ELSIF hom_axis_B_btn THEN
                homing_axis_select := 2;
        END_IF
```

```
CASE State OF

    0:              //State initial
        State := 50;
    50:             //State Operation mode select
        IF opr_mode_select=1 THEN
            State := 300;
        END_IF
        IF opr_mode_select=2 THEN
            State := 100;
        END_IF
    100:            //State Homing Axis select
        IF homing_axis_select=1 THEN
            State := 110;
        ELSIF homing_axis_select=2 THEN
            State := 120;
        ELSIF opr_mode_select=1 THEN
            State := 50;
        END_IF
    110:            //State Axis-A Homing
        Homing(Axis:=Axis_A,
                bCalibrationCam:=sens_A,
                Execute:=homing_start_btn);
        IF Homing.Done THEN
            State := 210;
        ELSIF homing_axis_select=2 THEN
            State := 100;
        ELSIF opr_mode_select=1 THEN
            State := 50;
        END_IF
    120:            //State Axis-B Homing
        Homing(Axis:=Axis_B,
                bCalibrationCam:=sens_B,
                Execute:=homing_start_btn);
        IF Homing.Done THEN
            State := 220;
        ELSIF homing_axis_select=1 THEN
            State := 100;
        ELSIF opr_mode_select=1 THEN
            State := 50;
        END_IF
    210:            //State Axis-A zero position move
```

```
        AbsoluteMove   (Axis:=Axis_A,
                                Position:=0,
                                Velocity:=10,
                                Acceleration:=20,
                                Deceleration:=20,
                                Jerk:=50,
                                Execute:=Zero_pos_btn);
        IF AbsoluteMove.Done THEN
                State := 100;
        END_IF
220:                    //State Axis-B zero position move
        AbsoluteMove   (Axis:=Axis_B,
                                Position:=0,
                                Velocity:=10,
                                Acceleration:=20,
                                Deceleration:=20,
                                Jerk:=50,
                                Execute:=Zero_pos_btn);
        IF AbsoluteMove.Done THEN
                State := 100;
        END_IF
300:                    //State Jogging Axis select
        IF jogging_axis_select=1 THEN
                State := 310;
        ELSIF jogging_axis_select=2 THEN
                State := 320;
        ELSIF opr_mode_select=2 THEN
                State := 50;
        END_IF
310:                    //State Axis-A Jogging
        Jogging(Axis:=Axis_A,
                        JogForward:=axis_move_forward,
                        JogBackwards:=axis_move_backward,
                        Mode:=MC_JOGMODE_STANDARD_SLOW);
        IF Jogging.Done THEN
                State := 300;
        ELSIF jogging_axis_select=2 THEN
                State := 300;
        ELSIF opr_mode_select=2 THEN
                State := 50;
        END_IF
320:                    //State Axis-B Jogging
        Jogging(Axis:=Axis_B,
```

```
                    JogForward:=axis_move_forward,
                    JogBackwards:=axis_move_backward,
                    Mode:=MC_JOGMODE_STANDARD_SLOW);
            IF Jogging.Done THEN
                State := 300;
            ELSIF jogging_axis_select=1 THEN
                State := 300;
            ELSIF opr_mode_select=2 THEN
                State := 50;
            END_IF
END_CASE
```