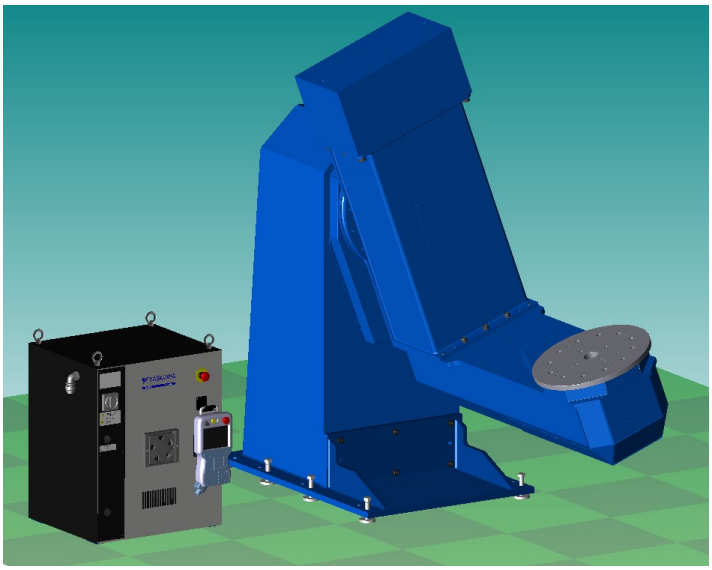
CHALMERS
UNIVERSITY OF TECHNOLOGY



Simplification and method description for implementing positioners in MotoSimEG-VRC

Bachelor thesis in Mechanical engineering

JOHAN LENÉ
FREDRIK STRÖM

Institute of Industrial and Materials Science
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Bachelor thesis 2019

Simplification and method description for implementing positioners in MotoSimEG-VRC

JOHAN LENÉ
FREDRIK STRÖM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institute of Industrial and Materials Science
Mechanical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Simplification and method description for
implementing positioners in MotoSimEG-VRC

Supervisor: Robin Lindor, Yaskawa
Examiner: Åsa Fast-Berglund, Chalmers University of Technology

Bachelor Thesis 2019
Institute of Industrial and Materials Science
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Positioner and controller in a simulated environment
Gothenburg, Sweden 2019

Preface

After three years of studying Mechanical engineering, 180 credits, at Chalmers University of Technology, this report is a result of Johan Lenés and Fredrik Ströms bachelor thesis containing 15 credits. Contact with Yaskawa was made through Thomas Sabel, the Human Resource Manager at Yaskawa, whom the students met at a Chalmers work fair. The current project has been carried out in the Industry and Material science department of Chalmers University of Technology. With this, the students would like to acknowledge:

Sven Ekered, our supervisor at Chalmers University, for a great support during the whole project.
Robin Lindor, our supervisor at Yaskawa, for making this project possible and for giving us the tools to achieve success with the project.
Åsa Fast-Berglund, our examiner at Chalmers University, for providing knowledge for this project.
Kristoffer Hedram and **Åsa Einarsson** at Lin Education, for providing their knowledge about teaching material and digital learning.

We also want to thank everyone at Yaskawa who has helped us with this project and everyone at Stena Industry Innovation Laboratory for their help, kindness and for letting us be in your premises.

Abstract

The robotic simulation software MotoSimEG-VRC from the robot manufacturer Yaskawa is a powerful simulation tool for simulating robot programs. The software enables the user to present layout solutions, program robots and positioners and to examine the reachability of the robot. Implementation of robots in the software works effortlessly but to successfully set up a positioner with working kinematics has proved to be more difficult.

To simplify this process, the students will examine an alternative method of implementation using information-files known as robotinf-files. By pedagogically listing the essential parameters of the script that need to be defined and how to find the data, manuals will be created of how to create the files and how to implement positioners using them.

The report is structured with four main chapters describing the process of how the work progressed. These are the results of what has been discovered during the project. The methodology of the work is mainly learning the software thoroughly and investigate the script to develop the working process of creating and using robotinf-files. By analyzing users and collecting information regarding how to best present information manuals are presented in Appendix A.3 and A.4 as the result of this work.

Besides the manuals, this project also explains the relevant parts of the software for this project. The explanation of variables and construction of the script could be used for further development and usage of the software.

Contents

1
INTRODUCTION

The introduction aims to give background information, including the aim and limitations of the project.

1.1 Background

The future of manufacturing is automated, and in order to increase the efficiency of production and the introduction of robot-based automation, simulation software is more often used. The simulation software MotoSimEG-VRC from Yaskawa is a powerful simulation tool for simulating robot programs. The program allows the user to plan, simulate and program the robot system. With MotoSim, a user can simulate a working model of the real robotic system.

Yaskawa offers several different robot solutions, from small robots to big spot welding systems. They sell and develop the systems, the controllers and all sorts of implements. Yaskawa is far gone in their development with welding robots, who often are combined with a positioner. To have a precise angle is crucial when welding. The robot cannot always reach every spot in the right angle, but with the help of a rotatable positioner, the object could be oriented to the best position for a welding operation. The more axes, the easier it is to reach the unapproachable places. The positioners, which could be single- or multi-axis, are used in robot-based automation to orient and manipulate objects in robot cells.

Today, the positioners are too complicated to set up in the program and is far too time-consuming, even for an experienced user. The robot kinematics has an easier implementation. In the setup, there are predefined robot models with correct kinematics and positions. The robot models are imported with an information-file that defines the rotation axes and the following kinematics. Yaskawa is interested in a similar approach for their positioners. This project will investigate methods of how to work more efficiently and help users internally och externally to more easily implement positioners in the simulation software.

1.2 Aim

This work aims to evaluate and simplify the working process of how to program positioners in MotoSimEG-VRC. This will be done by creating a detailed description of how to create and import information-files, similar to how robots are implemented. The goal is

to have a description that could be used by both experienced and inexperienced users.

By pedagogically listing the essential parameters that need to be defined and how the data is found, the user will not need to be as experienced, and the users could be more independent in their use of MotoSimEG-VRC. This means customers faster could use their robots and save work-hours for both Yaskawa and their customers.

If the positioners could be imported using information-files containing correct information of axis position and rotation the same way as the robots, the implementation will be much easier and faster. The description of how to attach the coordinates and kinematics will consist of steps of how to create the information-files, where to find the data and how to import the files. If possible, it would also be desirable to present improvement proposals of how the software is designed.

1.3 Limitations

This work will only consider one and two-axis positioners in MotoSimEG-VRC. The description of how to define the axes and coordinates are the same, and the results could later on be used on positioners with more axes.

Due to time constraints, the work will only take place in the software. No tests will be attempted on physical robots or positioners. No further simulations other than making sure that the kinematics works will be created.

Design solutions in the software will only be theoretical, not developed. The same applies to design changes in the CAD-files. Since the primary aim is to establish simplifications of how to implement positioners, this will be the main task. Further improvements will only be discussed theoretically.

1.4 Specification of issue under investigation

- Which are the essential parameters for the implementation and how are they found?
- How are the information files built, and how are they manipulated?
- What type of information is needed for the users? Will this depend on the user experience?
- In which form will the detailed description best be presented to users?

2

THEORETICAL BACKGROUND

In this chapter, the technical elements of the project are described. The components, knowledge about the software and the tools required to understand the project are specified.

2.1 Automation

Automation can be described as a process that works more or less without human interaction. By using logical programming commands and mechanized equipment, this technology has been highly adapted in production and manufacturing. The purpose is to relieve human labour and raise safety, but also raise efficiency and quality in the manufacturing processes (Lamb, 2013).

2.2 Industrial robots

Industrial Robots gets more and more common in the manufacturing industry, due to an ongoing trend towards higher automation in production. Especially in the metal and the electronics industries (International Federation of Robotics, 2018). As Miller (1989) mentions, there are several benefits with robots in the manufacturing industry, for example:

- Reduced labor cost
- Increased output rate
- Improved product quality
- Elimination of dangerous jobs

These benefits have been essential to develop robots further, to be even safer, accurate and faster. Today there are many different types of applications that industrial robots can handle, but often a robot is designed to complete a specific task, such as spot welding, material handling, pick and place, spray painting, and so forth.

2.3 Positioner

A positioner is a sort of robotic fixture which could manipulate the object and consequently integrate with the robot. With different sorts of positioners, one could be found who meets the requirement for the purpose. The positioners come with different numbers of stations regarding how many objects could be supplied and with a different number of rotation axes regarding how much the object could be manipulated.

The benefits of a robotic positioner are that the robot could manipulate the object more quickly and accurately (Yaskawa, n.d.). In welding operations, this has many advantages. With an integrated robot and positioner, a more favourable weld position could be found, which means higher quality and weld integrity. An orbital weld operation could, for example, be made in a single operation rotating the object instead of moving the robot.

This project only consists of one and two-axis positioners. Figure 2.1 shows a two-axis positioner to the left and a one-axis positioner to the right. The main parts of a positioner are listed below and how the parts will be named henceforth in this report.

- Base: The foundation of the positioner that is placed on the ground. (A in Figure 2.1)
- Arm: In this example, shaped like an L and rotates around the tilt-axis connected to the base. (B in Figure 2.1)
- Faceplate: This part is where the fixture holding the workpiece is placed. It rotates around the rotating-axis connected to the arm for two-axis positioners, or the base for one axis positioners. (C in Figure 2.1)

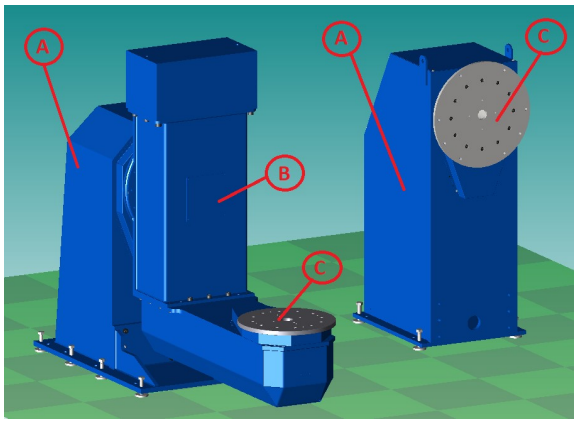


Figure 2.1: *Positioner. The authors own picture.*

2.4 Controller

A controller can shortly be described as a computer that's connected to the robot, that controls the servomotors in the joints of the robot for it to move correctly. It functions as a master to the robot and can control a network of robots and external axes so they may work together, where the controller restricts a maximum amount of axes. For the robot to move, the controller uses coded commands or robot programs, which differ between brands in how they are programmed (Nubiola, 2015). An industrial robot can be programmed either online with a device called a pendant attached to the controller that is used to control the robot's movement, or offline through simulation software on a computer. Figure 2.2 below shows the correlation between the controller, the robot and the positioner.

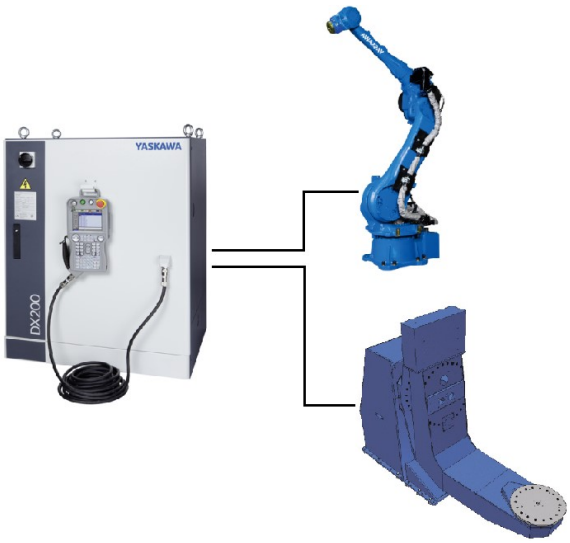


Figure 2.2: *Controller. The authors own picture.*

2.5 Simulation software

Robot simulations started in the early 1990s and been developed ever since (Gales, 2017). Simulation is defined as the imitation of the operation of a real-world process or system over time. It involves the generation of an artificial history of the system and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system being represented (Pichappan, Ahmadi & Ariwa, 2011).

Creating a complete virtual model of a robot or system by simulating components and control programs can significantly impact the general efficiency of a project (Smashingrobotics, 2012). If the kinematic, reachability, cycle time and layout of the system could be tested in advance, errors could be eliminated, and better decisions could be made. The later in the project, the more expensive it gets to make changes and simulations enables tests to be made in an early stage.

2.5.1 MotoSimEG-VRC

MotoSimEG-VRC is the robotic simulation software developed by Yaskawa. It is used for offline programming (explained in chapter 2.6.2), but also for layout planning or to control the robot reachability.

2.6 Robot programming

Today, there are two methods to program industrial robots systems, online- and offline programming.

2.6.1 Online programming

Online programming is usually done with the handheld pendant. The robot is controlled by manipulating its positions manually through the pendant, and desired locations are stored to build a program of the robot's movement later. The downside with this method is that the robot has to be out of production while programming but the reachability of the robot concerning the surroundings are visible.

2.6.2 Offline programming

This method is based on using simulation software on a computer to program the robot systems in a virtual environment. The software is using CAD-files that resembles a robot cell, where the user could simulate a work task for the robot system. This simulation program can later be translated to robot code and transferred to the physical controller.

The significant benefit with this method is that it has almost no impact on the robot systems productivity. However, the simulation software will never be completely accurate to represent the real world, so programs need to be controlled with the real system (Owen-Hill, 2016).

2.7 Coordinate systems

In three-dimensional space, the coordinate system is based on three perpendicular axes, x, y and z, which gives length, width and height. This is what is known as the Cartesian coordinate system and enables positions to be determined for a three-dimensional space (Nykamp, n.d.). The intersection of the axes is called the origin and has the value of zero on each axis.

An object could be moved and rotated around the x, y or z-axis. In a virtual environment, a coordinate system is needed to locate the objects. The object imported to the virtual environment is built with its own coordinate system, also known as OCS (Object Coordinate System) (Umeå Universitet, n.d.). The virtual environment also has a coordinate system located in the origin of the virtual environment known as WCS (World Coordinate system). From this, the objects could be moved or rotated relative to the world. This is also known as a transformation.

When an object is moved, known as translation, it could shift in all directions. A positive value is in the direction of the coordinate-axis and vice-versa. For rotations, there are two different ways to think, as fixed angles or Euler angles. For Euler angles, the coordinate system rotates along with the object. That means that a rotation around the x-axis moves the y- and z-axis and a sequent rotation is around the y- or z-axis is done with their current axes (Umeå Universitet, n.d.). For fixed values, the coordinate system orients from the original orientation. A rotation around the x-axis moves the y- and z-axis, but a sequent rotation around the y- or z-axis is done with the original axis. Summarized, fixed angles rotate around the original OCS, and Euler angles rotate around the new axes. The rotations are however done in different orders. Rotations around x-, then z-axis with fixed angles gives the same coordinates as with Euler angles for rotations around z-, then x-axis.

2.8 Right-hand rule

The right-hand rule is used to get correct directions of coordinate axes and rotation in the Cartesian Coordinate System. The thumb, index finger and middle finger of the right hand should be held, so they form three angles 90 degrees from each other. The thumb point the positive x-axis, the index finger the positive y-axis and the middle finger represents the positive z-axis. The correct rotation around each axis is as imagining wrapping the right hand around any axis, with the thumb indicates the positive direction, then the positive rotation follows the direction of the four fingers (Dogra & Randhawa, 2017).

2.9 HTA

Hierarchical Task Analysis (HTA) is an established method used in industrial learning. It is being used to create a clear structure and workflow for the tasks being performed (Fast-Berglund & Mattsson, 2017). The HTA-analysis is displayed with a tree structure where its depth and width are analyzed. The form of the diagram shows how complex the task is. The greater the depth and width of the diagram, the more complex the task is. In the diagram, the workflow can be followed by hierarchical and chronological order. The sequence chronologically is from top to bottom, then left to right.

3
METHODS

In this chapter, the approach and arrangement of the project are described.

3.1 Learning MotoSim EG-VRC

The students were offered education of the software at the beginning of the project, mainly to get an understanding of how the coordinates and rotations axes are configured and manipulated. During two days of training in SII-Lab with the supervisor from Yaskawa, Robin Lindor, the basic understanding of the program was achieved.

Once the basic understanding of the software existed, the work of defining the different aspects of the project was initiated. Several tries and experiments led to the comprehension needed for the development of the script and working methods. A crucial part of this project was to get a deep understanding of the software. To achieve this, many hours were spent learning. With increased knowledge, different working methods and structures of the script could be developed and evaluated. There is no learning material on the software for this purpose, so learning by doing and taking notes is how progress was made. To understand the logic, the purpose of the controller and the different features several attempts were made in the software to understand fully.

Accordingly to the theory of the learning pyramid, that is also how progress is made, see Figure 3.1. The theory, developed by the National Training Laboratory, suggests that most students retain nearly 90% of what they learn through teaching others (Education corner, n.d.). The learning process of the software consisted nearly exclusively of active learning, where both students participated and discussed the work.

Figure 3.1: *Learning Pyramid. (Kuropatwa, 2007). CC-BY*

3.2 HTA

HTA was used to evaluate each step and to simplify the process. When the present way of working was defined and listed, the work steps could be seen and evaluated from their meaning and approach. With analyses of how the work previously was done and how the future method is made, regarding the work of creating and using robotinf-files, the different approaches could be compared and evaluated. Through comparisons regarding the width and depth of the analysis, the complexity and time of the work could be evaluated and developed. The HTA also functioned as a first guide since the steps from start to finish were listed even though the analysis did not contain descriptions of how.

3.3 Interviews

By interviewing users of the program, both experienced and inexperienced, the students got a hold of what type of problems were encountered when installing a positioner. Questions were asked regarding how the software was used today and how to elaborate the methods. The data collection consisted of qualitative interviews since the software is relatively small. Interviews were primarily made on users of the software to develop the methods and manuals but also experts in different fields linked to this subject.

The objective of all interviews was to get a discussion regarding the subject, which enabled a more detailed understanding of the user experience of the software. A broad spectrum of ideas and thinking outside the box were essential for creating a description for everyone. That is why a broad variety of people were interviewed in this project.

3.4 Observations

By observing people using the manuals and work in MotoSim, the students got a more extensive knowledge of the situation. The students could then consequently see how smoothly the implementation went. With the different manuals, from the preliminary to the final ones, users were observed to see which parts worked and which ones needed improvement.

Observations were a complement to the interviews and led to further discussions afterwards. While the interviews gave the students opinions from the users, some things had to be noticed. How the user interpreted the manual and where they got stuck is something not all people evaluate. Some sequences that were easy and obvious for the students first became known as a problem after observations. The observations also led to some new comprehension. Experienced users of MotoSim, yet not of the script, had some alternative ways to work which were valuable knowledge for the students.

3.5 Evaluation and validation

When manuals were created, there was a need for feedback to analyze the capability and user experience. A variety of different people, from Yaskawa employees to Chalmers professors and students were interviewed and observed to see the pedagogy of the descriptions. The validations led to a deeper understanding from the experienced users and things not thought of from the inexperienced users, which led to a complete list of information needed.

The validations happened in different steps and helped the students to evolve both the working methods and refine the design of the manuals. It was obtained through discussions from the interviews and observations. The meaning of this was to verify that the users at Yaskawa accepted the student's choices. The feedback was analyzed and further led to the best possible result.

3.6 Developing the manuals

The work of developing the manuals occurred in many different steps. From the HTA, the steps in chronological order were listed, and during the work of developing the working process, the important steps and critical parts were noticed. Preliminary manuals were compared and discussed with MotoSim users from a pedagogical and user experience viewpoint.

Once the process of how to create and implement the robotinf-files was decided, the work of developing the final manual started. In order to create a well functioning and good-looking manual theory regarding manuals was studied. A visit was also made at a company working with digital learning to get a deeper understanding of how to best present information for users.

4

EXPLANATION OF MOTOSIM

This chapter mentions the relevant information regarding what has been taught in MotoSim. In order to provide clarity for the following chapters this chapter explains the most common topics needed for the project.

4.1 Controller

In MotoSim, the type of controller-system is chosen during the initialization. The controller this work has focused on is the Yaskawa DX200, this because this model has the most positioners available. It can control up to 72 axes, so it is often used in networks with several robots and positioners that can collaborate (Yaskawa, 2017).

With this controller, a handheld pendant is included to program the robot. During offline programming in MotoSim, the pendant connected to DX200 is a replica of the physical pendant, seen in Figure 4.1. There is a slight difference between the pendants for different controllers, which results in a different working process for different controllers. Therefore the manuals created will only cover said controller model.

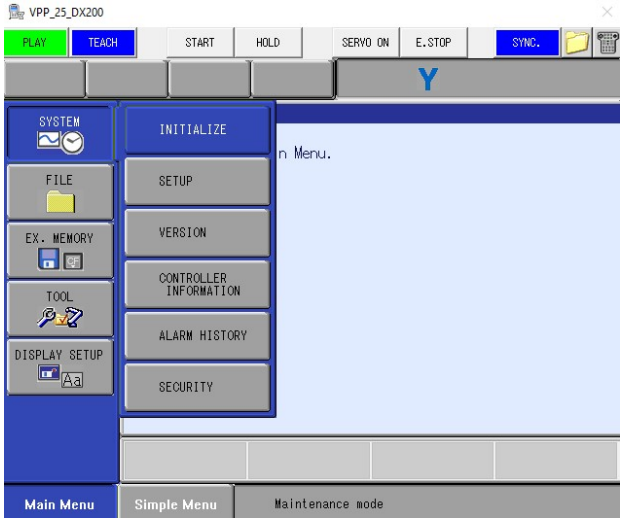


Figure 4.1: *DX200 pendant. The authors own picture.*

4.2 Rotation axes

For each positioner installed in MotoSim, the rotation axes have to be defined for the simulation to work the same way as in reality. The most natural position to define is the intersection between the tilt axis and the rotating axis, see Figure 4.2, where the tilt axis is the rotation axis for the arm, and the rotating axis is the rotating axis for the faceplate.

The positioner has a direction of rotation, which is considered the positive direction and vice versa. For practical applications, this has to be entered correctly. The positive direction is found in the construction drawing sheets, and by using the right-hand rule regarding the direction of the tilt and rotating axis, the positive direction could be defined.

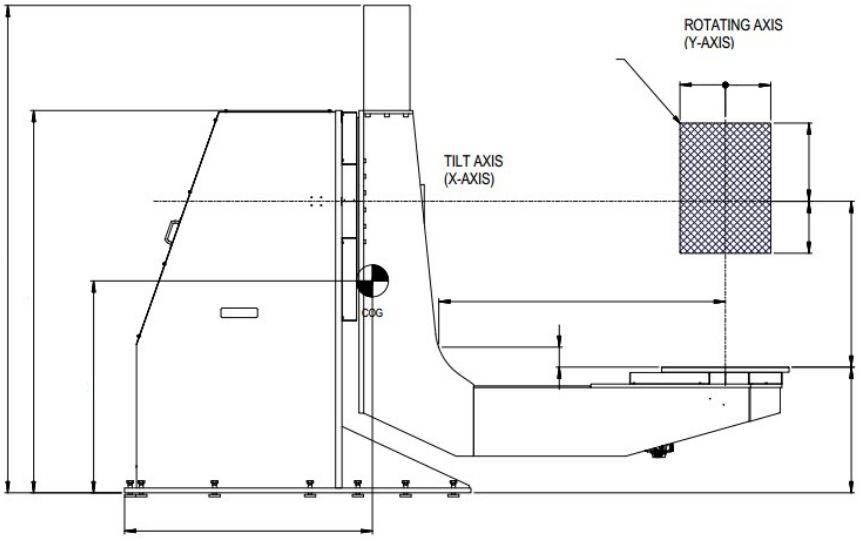


Figure 4.2: *Construction drawing sheet. The authors own picture.*

4.3 Hierarchy

For the models in MotoSim to move correctly, there is a need for a hierarchical build-up. The system in MotoSim builds on a parent/child relation. These relations can be edited and selected when configuring a positioner. The action of moving or rotating a parent will lead to the children following. However, when moving a child connected to a parent, only the child can move or rotate without any influence on the parent.

4.4 Kinematics

The kinematic models in MotoSim are generated when initialising a controller and are placed in the world origin but can be moved and oriented to the desired position. In MotoSim, the kinematic models are visualised as a frame. These models are the rotation centre and rotate around their positive z-axis, which is manipulated in Position panel. The degrees of rotations are chosen during the Controller setup. When initialising a

positioner in the controller, they are categorised as external axes and defined by the number of rotation-axes. Currently in MotoSim, there is only the possibility of inserting two external axes. Turn-1 is chosen for one-axis positioners and Turn-2 for two-axis positioners. The kinematic model inserted will for a one-axis positioner be named ex1, and for a two-axis positioner, ex2 will also be added. The kinematic models are attached to the rotation axes, which is called EX1-POS and EX2-POS. The rotation axes are just coordinates to which the kinematics are attached.

4.5 Position panel

This function is used to rotate joints manually in MotoSim. These joints are kinematic models and can be manipulated separately and can be rotated both in positive and negative direction accordingly to the axis orientation. For each axis, there is a maximum and minimum value of rotation and is set during the controller-setup.

4.6 CadTree

The CadTree is a toolbox in MotoSim that shows all models, axes, frames etc. used in a saved cell, displayed as a column in a tree structure. From this structure, it is easy to understand parent/child relations of the models. In the toolbox, it is possible to add, delete and manipulate all parts in the tree, and also read the coordinates for all parts.

4.7 Model library

This feature in MotoSim is used to import CAD-files from a specified folder located on the hard drive. In the Model Library window, a preview of the model is shown. The models imported to the cell might consist of one or many CAD-files, which relation, position and axes can be found in the CadTree. The CAD-files used for MotoSim are saved in .hsf format.

4.8 Robotinf-file

A robotinf-file is a text-based script containing information of rotation axes, model files and hierarchy. Instead of manually insert this data to MotoSim, the robotinf-file places CAD-files, inserts axes and sets wanted hierarchy relation. This file can be opened in Notepad or any similar text editor, where the content can be edited for a specific use. In chapter 6.3, the content of the robotinf-file will be further explained and clarified.

5
WORKING PROCESS OLD WAY

In order to understand the progress of how the manual was made and how the new method to work has been structured, the previous process has been listed and explained. This process was taught at first and served as a foundation for the subsequent development.

5.1 Generally

The previous working process creating a functional positioner without a robot-inf file contains many steps. The user needs to be familiar with how MotoSim and the kinematic works, and a broad understanding of the software is needed. The Figure 5.1 below shows the main steps in the right order from the Hierarchical task analysis. The analysis, which served as the first disposition of the steps, helped the students to learn and further develop the method. The entire Hierarchical task analysis containing all steps is found in Appendix A.1. In the following subchapters, the different steps are described more thoroughly.

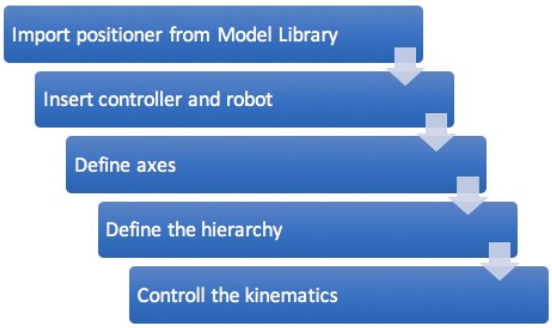


Figure 5.1: *Main tasks old working process. The authors own picture.*

5.2 Import positioner from Model library

The first step of implementing a positioner is to add the CAD-files from Model library. All users of the software have a library containing the models which easily could be added to the world by drag and drop or a double-click. Engineers at Yaskawa has different folders in the Model Library containing different categories of models. The students were, however, only given one and two-axis positioners due to the limitations of the project.

The CAD-files in the .hsf-format are “dead files”, meaning they do not have functioning kinematics. The axes of the rotating parts are defined as positions of the intersection

between the tilt and rotating axis. They are defined as centre lines and are given as positions. To import these models containing functioning kinematics is the goal of this project. All the following steps in this method is to get the kinematics to work correctly.

5.3 Insert controller and robot

Next step is to insert and choose the controller with the right parameters for the simulation. Different parameters such as offset, motion range, reduction ratio and rotations are added when the controller is implemented. The only parameter relevant to this project is the rotations of the axes. The other parameters are relevant for off-line programming and need to be correct in order for the positioners to work in practice, but since this project is limited to only trying the kinematics, they are irrelevant. During visits at Yaskawa, it was noticed that a lot of the work in MotoSim is layout planning. Making sure positioners and robots do not clash and fit in the robot cell is essential steps for the sales department.

In MotoSim a virtual pendant opens when initialising a controller, the same as the handheld pendant attached to the physical controller. This pendant has to be operated with the keyboard and cannot be operated with a mouse. The arrow keys are used to orient, the space bar to choose an option and the enter key to move on to the next stage. This pendant took a while to get used to and felt a bit cumbersome, but once accustomed, the controller could be initialised quite quickly. The first step is to choose a language and then to choose a robot and what sort of positioner.

In this stage, the actual positioner is not chosen, only how many axes it has. It is the axes that are implemented, which later on, connects to the model file of the positioner. Rotation, the only relevant parameter, will give the model files the ability to rotate around these axes. Once all parameters have been entered, the robot kinematics is automatically loaded with a robot-inf file which gives it working kinematics. Once initialised the robot with working kinematics is submitted to the world. For the positioner, MotoSim generates the same amount of kinematic models as chosen axes during the setup. The models generated are called ex1, and if two axes are chosen, ex2 is added.

5.4 Define axes

For each positioner installed in MotoSim, the rotation axes have to be defined for the simulation to work the same way as in reality. The most straightforward position to define is the intersection between the tilt axis and the rotating axis, as explained in chapter 4.2. The kinematic models then ties to the positioners axes, called centre-lines.

By moving EX1-POS to the first tilt axis and EX2-POS to the second rotation axis, the rotation centre is defined. This is done by left-clicking on the kinematic models in the CadTree and choosing to which centre-line the kinematic models are being attached.

5.5 Define the hierarchy

Once the model files and axes have been inserted to MotoSim, the next step is to build the hierarchy in a way so that they have the correct relationship. For the kinematics to work correctly, the hierarchy can be set up in many different ways, but some combinations might be problematic if the positioner has to be moved. Therefore the most straightforward and most logical solution is that if the base is moved, then both the arm and the faceplate follows. To change the parent/child relations, the SetParent function, found when right-clicking on a part in the CadTree is used.

In order to achieve the correct hierarchy, the body is set as a parent to the first rotation axis, which further is set as a parent to the arm. When the rotation axis is affected, the arm will rotate as well. With the logic above, the arm is given a parent relationship to the second rotation axis, and further this axis as a parent to the faceplate. This gives the hierarchy a logic and simple structure that results in a correct kinematic movement and can easily be moved in the simulation environment.

In Figure 5.2 the importance of the correct hierarchy can be seen. The positioner to the left shows a correct build of the hierarchical structure. However, with wrong parent/child relations, the kinematics will not function properly as the positioner to the right shows. The hierarchal structure has previously not been defined and therefore this step is very error-prone.

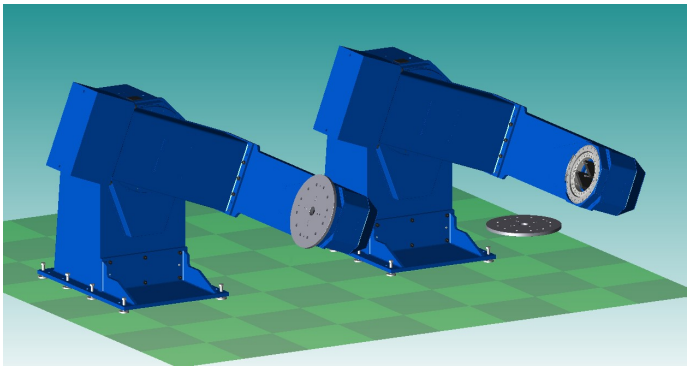


Figure 5.2: *Faulty hierarchy. The authors own picture.*

5.6 Control the kinematics

The last step, once the positioner-setup is finished, is to control that everything works properly. Even though this project does not involve simulations actually to be made, the kinematics still has to be tested. This could be done by rotating the axes independently to make sure the hierarchy and the parameters have been entered correctly. Under the toolbar, the function Position Panel is found, which is used for this purpose. When selecting the positioner, the axes could rotate separately. If the axes work properly and have a positive direction according to the construction-sheets, the implementation is completed.

6
WORKING PROCESS NEW WAY

This chapter explains the development of the new method. The content and learning process of the robotinf-files, as well as the methods of creating and using them, is presented. This is the foundation that leads to the final result of the manuals.

6.1 Generally

The knowledge of the software was acquired by experimenting with both work-methods. To understand the different components of the robotinf-file, an understanding of the old work-method was necessary. A comprehension of the hierarchy of the components, as well as a perception of coordinates, orientation and positioning, had to be obtained. By making several attempts in different ways, a conception of the logic behind the software was procured. If the hierarchy was not appropriately built the kinematics would not work, and if the rotation of the axes was wrongly implemented, the positioning would be wrong. Little by little advances were made, and different ways to build the hierarchy was discovered. As long as there is a logic behind how the different components and their axes are linked to each other, there are different approaches to a successful implementation.

This comprehension of these different features led to the proceeded work of developing the robotinf-file. Yaskawa gave a functioning file but merely as a suggestion, and the task was to eliminate unnecessary parts and slim it down to make it more understandable. Several attempts of eliminating different parts and trying different constructions of the script, as well as discussions with employees at Yaskawa and Per Nyqvist, an expert of robot simulations at Chalmers, led to a broader understanding of the features in the robotinf-file.

The new working process consists of working with the information-files, also known as robotinf-files. When the files have been created the experience needed to build the simulation models are minimized. Also, with ready-to-use information-files, the time to implement the positioners are greatly reduced. This will enable Yaskawa to offer their customers the files which will make the simulation more user-friendly and save working hours of both Yaskawa and their customers. Therefore two different manuals will be developed to have a standardized work of how to create and implement the information-files.

The first manual will be about how to create these files. It will contain a detailed description and a step-by-step guide of how the files are created. Even though it is relatively time-consuming to create these files, it is a one time job. Once the information-file has been created, it saves much time each time that positioner type will be needed in another simulation. In the long-run, a library of information-files could make it a lot easier and faster to build new simulation models. To create a new robotinf-file takes about the same

time as to implement a positioner the old way, as explained in chapter 5. This process has some advantages in a logical sense, but mainly the time-spending aspect for the following time simulation models are made. This manual will mainly be used internally at Yaskawa since they are creating the information-files.

The second manual will be a shorter description of how to implement the files once they are created. The latter one will be simpler to use and should be able to be used by anyone familiar with MotoSim. Here is where Yaskawa could offer their customers the information-files accordingly to what positioners they have acquired.

6.2 Method of how to use the new robotinf-file

This method to work by will result in a manual that will both be used as a complement to the first manual and as a more accessible guide for people using ready robotinf-files. Regarding the complement to the first guide, there is no point to explain the steps of creating the controller and verify the kinematics twice. If the first manual includes the steps of creating the file, the second manual could be referred to how to do the last steps and verify the kinematics. The complexity of this manual is not as complicated as for the first one. These steps will be more pedagogically listed since this should be able to be used by less experienced users of MotoSim.

The implementation of an existing robotinf-file is relative to creating the file, less technically demanding. The files are implemented in the final step of initialising the controller, and the most significant difficulties are how to orient through the pendant and what parameters are implemented. This project was limited to just making sure the kinematics works, and thus, the only parameter needed is the positive and negative rotation degrees. A lot of others parameters, such as motor type, I/O modules, reduction ratio etc. are added in this stage but are not relevant for the project cause. As long as the axes are added when defying if it is one or two axes and the rotation degrees are chosen during the initialisation, the kinematics could be controlled.

A preliminary manual was then made of how to create a controller and further to control the kinematics from the HTA. The steps of implementation comparing the old method and the new method are much shorter. When comparing the analyses in Appendix A.1 and Appendix A.2, the picture becomes clear. The steps removed are the ones replaced by the script.

The manual, with a description of how to orient through the pendant and what parameters need to be inserted, was made. A first validation was made on the student's supervisor. The first feedback received was that it was not profound enough. Both the texts and pictures could be more describing, and notes were taken. Later Yaskawa engineers tried the manual as well, and the same observations were made. Even though some clarifications on the manual was needed, with some highlights and better descriptions, the method of work seemed fine. The entire HTA containing all steps is found in Appendix A.2, Figure 6.1 below shows the two main steps.

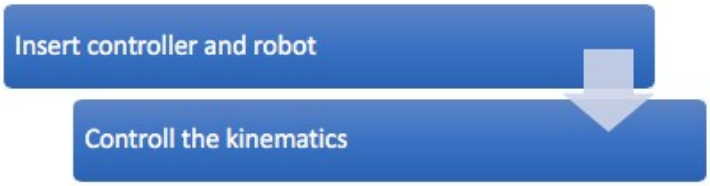


Figure 6.1: *Main tasks new working process. The authors own picture.*

6.3 The content of the robotinf-file

During the first education with Yaskawa, the students used the current robotinf-template to import a positioner. All variables of the file were not known during the education, therefore an essential part of this project was to define all the contents of the robotinf-file. By manipulating the script, the group could develop the working process and change the outcome in many different ways. The essential contents of the robotinf-file that have been clarified during this project are:

- ADD This function is used to import a model file to an already existing unit. The relationship between the parts will be that the imported file is going to have a child relation to the already existing unit.
- AXIS6 Gives the model the wanted position and orientation by typing the coordinates in mm and direction in degrees (x, y, z, Rx, Ry, Rz). The position and orientation is relative to the parent position and orientation.
- MOV This function moves an existing unit to another existing unit and sets the hierarchy with a child relationship.
- Dummy - The meaning of Dummy is usually used as a temporary unit that needs to be manipulated afterwards. In this case, if not creating a positioner as a dummy, it will instead be created as a fixed unit and the ability to change the positioners name will be lost. It can be made when initializing the controller once, but once it saved, the name is set.
- %RBPATH% points to the directory path containing .hsf files. The path is in the same folder as from where the robotinf-file is currently placed.
- %RBNAME% returns the same variable typed as the positioners name during the controller set-up.

6.4 Method of how to create the robotinf-file

The step-by-step guide of how to create the file once the robotinf-files structure was defined had two different techniques. The first one was to use the data of the construction drawing sheets and define the position between the axes out of the distances in the sheets. This would be preferable as the user directly could create the robotinf-files without even opening MotoSim. However, the problem emerged when different CAD-files had different origins, and there is no way to comprehend this from the drawings, as seen in see Figure 4.2. That means the positioners CAD-parts in MotoSim cannot be oriented since there is no way to define these origins in relation to the MotoSims origin. Also, if the CAD-files

are defined with different coordinates (x, y, z) than the coordinates in MotoSim, they ended up wrong.

The problem could be fixed with a standardization where the people creating the CAD-files always constructed the positioner from the same centre and with the same coordinates. Even though that could, relatively easily, be established for future work, the problem mentioned above would still remain for all the existing positioners. Making these CAD-changes for all positioners would be very time-consuming with no significant benefits.

The second technique is first to import a positioner from Model Library to MotoSim. Then the coordinates and positions between the CAD-parts are correct and based on that the user could study how the axes are rotated and the distances between different origins. This ended up being more flexible since no form of standardization was needed. The second technique in that sense seemed the most logical and a preliminary manual was created and were discussed with Yaskawa.

First of a preliminary manual of how to create a script accordingly to the hierarchy was made, and was then verified with Yaskawa engineers. This meant that each part had to be measured accordingly to their parent. The measuring-tool, as seen in Figure 6.2, was easy to use in MotoSim and made sense to the engineers. However, it seemed hard to grasp between what points the measuring should be made. The hierarchy of models and axes in MotoSim that first seemed logical became more problematic when inserting them into the script. Even when using a manual, it was hard to know where to measure. The base was located in relation to the world, the first axis in relation to the base, the second axis in relation to the first axis, the arm in relation to the first axis and the faceplate in relation to the second axis. This was hard to grasp for users.

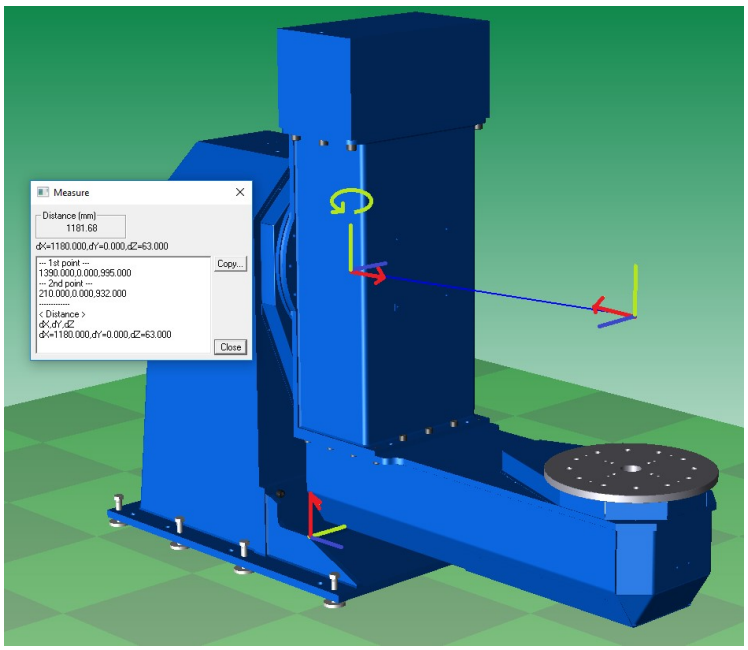


Figure 6.2: *Measuring in MotoSim. The authors own picture.*

The rotation of all the models and axes was even more complex. The base rotation could be read in a panel below the CadTree, but for all the other models and the two axes, the only way to understand the rotation was through the right-hand rule. As seen in Figure 6.2, the arm is placed from the first rotation axis, and the coordinates are relative to the rotation axis orientation and placement. In this case the arm needs to be placed in the positive z-direction and rotated 180 degrees around the y-axis. The distance-tool in MotoSim gives the distance according to the worlds orientation (the bottom coordinate system in Figure 6.2), which means that the coordinates given by the tool is not the same as the coordinates that need to be entered into the script. Even though experienced CAD- and CAM-users are familiar with coordinate systems, this is an error source. If a coordinate system is rotated two times, it is easy to make mistakes. Since MotoSim using fixed coordinate systems and not Euler angles as many other systems, a simpler way to work had to be found.

The engineers who first tested the preliminary manual asked for a more standardized way of work. Mainly it was the rotation of axes which was troublesome but also the measuring. An idea that the user could read the values below the CadTree for both the coordinates and rotations for all the models and axes, the same way as for the base was wanted. This idea had earlier been discussed but rejected because of a structure of hierarchy the same way as before seemed like the best choice. A different idea of how to build the script was introduced.

When the script was structured, so everything was tied to the world, the execution of creating the script became much easier. As desired, all positions and rotations could be read below the CadTree, shown in Figure 6.3, and implemented in the script. The script still built the hierarchy correctly, but the data is simpler to interpret. When the positioner is added to the world, and all the origins of the models and axes are relative to the world, it is easier to understand the data. A new template of the script was made with a few changes to the manual regarding how to structure the positioner and how to read the data.

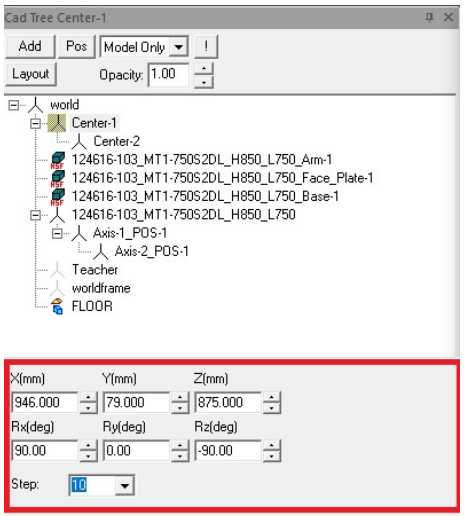


Figure 6.3: *Coordinates in CadTree. The authors own picture.*

The new manual was later verified with Yaskawa engineers. As expected, the new course of action went much smoother. It became more difficult to make mistakes, and the implementation went faster when the user directly could read the values without using the measuring tool. Both the students and Yaskawa engineers were convinced this was the right method. Even though the preliminary manual was not enough descriptive the method became a basis for further work with the manual.

One thing that was discovered during the observations of the engineers was their experience with MotoSim. The students assumed the software was widely used, but it turned out only a few were familiar with the software. This was discussed with a project manager who was familiar with the problem. At term, this is something they want to change with these manuals as a start. The original idea that the manual used by Yaskawa employees could be a bit less detailed had to be changed. The further work of creating this manual included to make it as detailed as the second one.

6.5 The development of the robotinf-file

During the introduction and education in MotoSim with supervisor Robin, the students were handed the robotinf-file Yaskawa currently is using for the positioners. During the education and by using the script, the group discovered some problems that needed to be investigated:

- If needed to import the same positioner model twice, there had to be separate folders containing both robotinf-files and model-files.
- The meaning and use of a Dummy-model had to be looked into.
- The script is very error-prone.
- The script is unstructured and long.

In order to create an easy-to-follow manual and a standardized working process, these parts had to be investigated. A more accurate description of the robotinf-file, and discoveries of the script in chronological order is explained below.

First, the script creates a Dummy model to the world called S1, and the positioner created from the controller is moved as a child to the Dummy model with the MOV function. The following steps in the script add the model files for each part with a fixed name, and the position and orientation relative to its parent. When a positioner is created, it automatically creates the position for the kinematic, seen in Figure 6.4 as EX1-POS and EX2-POS, with the second rotation axis connected to the first rotation axis as a child. The Faceplate and an eventual Arm is consequently connected to the kinematic models as a child relative to them. The hierarchy for the positioner with this script will result as Figure 6.4 below.

```
ROBOT
{
TYPE=MT1-5000S2D-S1
}

MODELMACRO
{
ADD world S1 dummy
AXIS6 S1 = 0,0,0,0,0,0

MOV %RBNAME% S1

ADD S1 124621-100_MT1-5000S2D-1_Base-1 "%RBPATH%\124621-100_MT1-5000S2D-1_Base-1.hsf"
AXIS6 124621-100_MT1-5000S2D-1_Base-1 = 0,0,25,0,0,0

AXIS6 %RBNAME%_EX1-POS = 2339.5,0,1856,-90,0,-90
ADD %RBNAME%_ex1 124621-100_MT1-5000S2D-1_Arm-1 "%RBPATH%\124621-100_MT1-5000S2D-1_Arm-1.hsf"
AXIS6 124621-100_MT1-5000S2D-1_Arm-1 = 0,0,-1960,180,0,180


AXIS6 %RBNAME%_EX2-POS = 0,0,0,90,-90,0
ADD %RBNAME%_ex2 124621-100_MT1-5000S2D-1_Face_Plate-1 "%RBPATH%\124621-100_MT1-5000S2D-1_Face_Plate.hsf"
AXIS6 124621-100_MT1-5000S2D-1_Face_Plate-1 = 0,0,-797,0,0,0
}
```

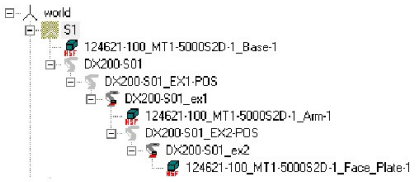


Figure 6.4: *First script and following hierarchy. The authors own picture.*

The way this script is built, if needed to use the same positioner type more than once to the controller, there has to be the same amount of separate robotinf-files in separate folders. This because the names of the parts have to be unique in the controller.

When the definition of %RBNAME% was discovered, the group could eliminate two problems identified with the first edition of robotinf-file. When inserting the variable for the model-files, MotoSim names the parts the name given during the initialization. The name that is typed after the variable, is the name that will be shown in the CAD-tree.

The need to have separate folders for the same positioner was then eliminated, and the same robotinf-file could be used several times for the same controller. When %RBNAME% is used, unique names will appear in the controller as long as any chosen name is not used twice.

The dummy, which just was a temporary unit, was eliminated at this point. The script got slimmer and more flexible than the first edition, see Figure 6.5. The coordinates and orientation are still based on the parent/child relation.

```
ROBOT
{
TYPE=MT1-1000S2D
}

MODELMACRO
{

AXIS6 %RBNAME%_EX1-POS = 1390,0,995,90,0,-90
AXIS6 %RBNAME%_EX2-POS = 0,0,0,90,0,0

ADD %RBNAME% %RBNAME%-Base "%RBPATH%\124617-100_MT1-1000S2D_H970_LR850_Base-1
AXIS6 %RBNAME%-Base = 198,0,25,0,0,-90

ADD %RBNAME%_ex1 %RBNAME%-Arm "%RBPATH%\124617-100_MT1-1000S2D_H970_LR850_Arm.hsf"
AXIS6 %RBNAME%-Arm = 0,-63,1180,0,180,0

ADD %RBNAME%_ex2 %RBNAME%-Faceplate "%RBPATH%\124617-100_MT1-1000S2D_H970_LR850_Face_plate.hsf"
AXIS6 %RBNAME%-Faceplate = 0,0,450,0,90,0
}
```

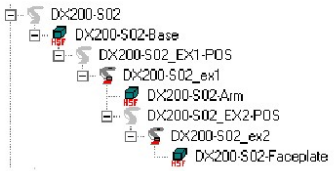


Figure 6.5: *Second script and following hierarchy. The authors own picture.*

After the first verification of the manuals, a few problems emerged that needed further

investigation. The problems regarding the implementation of coordinates and orientation, mentioned in the previous chapter led to some changes in the script.

The parent/child method was difficult to understand for users, and the relation between the axes and model-files demanded a deep understanding of coordinate systems. To make the implementation easier, it was chosen that all parts should be relative to the world. A problem however occurred when all parts became relative to the world. When installed in MotoSim, the parts positioned wrongly.

This problem got addressed by adjusting the hierarchy with the script using the MOV function to set the correct relations. The students later fixed the script to adjust the hierarchy by moving the parts to the correct relations by using the MOV function. A user also suggested that the parts article-number and model name is desired when the positioner is installed. The script now generates the same name for each part as the .hsf file name. In this case, the name has to be added at least three times in the script, but the possibility for name changing still exists.

Yaskawa employees also desired the possibility to change the name of the entire positioner. Therefore a dummy-model with the same name as typed during the initialization of the controller is added with the prefix "Positioner", see Figure 6.6.

```
ROBOT
{
TYPE=MT1-1000S2D
}

MODELMACRO
{

ADD world Positioner-%RBNAME% dummy
MOV %RBNAME% Positioner-%RBNAME%

AXIS6 %RBNAME%_EX1-POS = 1390,0,995,90,0,-90
AXIS6 %RBNAME%_EX2-POS = 0,0,0,90,0,0

ADD %RBNAME% %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Base-1 "%RBPATH%\124617-100_MT1-1000S2D_H970_LR850_Base-1.hsf"
AXIS6 %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Base-1 = 198,0,25,0,0,-90

ADD %RBNAME% %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Arm.hsf "%RBPATH%\124617-100_MT1-1000S2D_H970_LR850_Arm.hsf"
AXIS6 %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Arm.hsf = 210,0,932,90,0,90

ADD %RBNAME% %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Face_plate.hsf "%RBPATH%\124617-100_MT1-1000S2D_H970_LR850_Face_plate.hsf"
AXIS6 %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Face_plate.hsf = 1390,0,545,90,-90,0

MOV %RBNAME%_EX1-POS %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Base-1
MOV %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Face_plate.hsf %RBNAME%_ex2
MOV %RBNAME%-124617-100_MT1-1000S2D_H970_LR850_Arm.hsf %RBNAME%_ex1
}
```

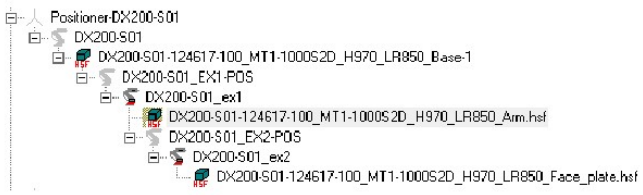


Figure 6.6: *Third script and following hierarchy. The authors own picture.*

After several tries, the group noticed that there was a high fault-ratio when naming the parts in the script. For each part, it is needed to copy and paste at least three times,

and since the script is space-sensitive, there is a high risk to make errors. Therefore the students decided that the article number should be typed as the name for the positioner during the installation for the controller.

By the final template, if given their article number when initializing the positioner in the controller, each model file was named their article number and then which part. Both the model-files and the dummy, which also gives a name to the entire positioners, could however be changed afterwards. This worked as the best template both regarding how short it is, the simplicity of it and the possibilities to make changes regarding the names. More about how to make name changes is described in Manual 1, found in Appendix A.3.

The final templates for the scripts, both for one and two-axis positioners, which is used for the manual is presented in Figure 6.7.

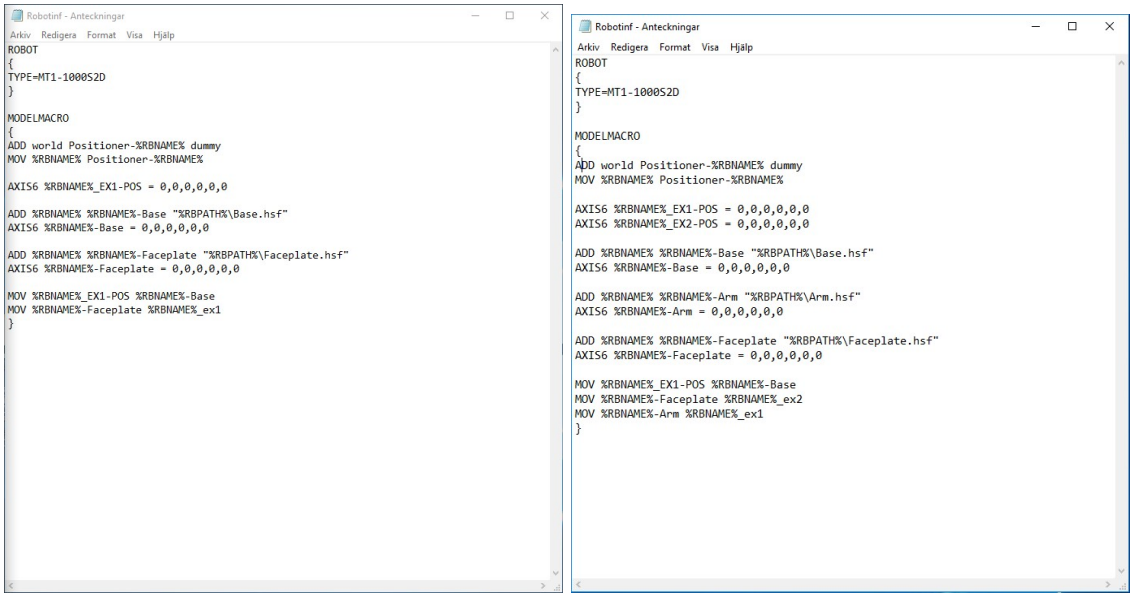


Figure 6.7: *Final scripts. The authors own picture.*

7
WRITING A MANUAL

This chapter describes the work of how the final manuals were made, including some theories regarding the design aspect and guidelines of what a thorough manual should include.

7.1 How to create a manual

The aim of the project was to create relevant manuals for the implementation of positioners in MotoSim. It was discussed to present the information to users in alternative ways, such as movies, games or audios. The experience of the software was as mentioned less than expected, and all steps thoroughly had to be explained. Movies could have been used but would have been harder to follow with that many steps. A lot of open tabs would also have been needed to complete the task. MotoSim, the template and then a movie player would have been too troublesome to use. Therefore a text-graphics manual pedagogically listing the steps seemed like the easiest and most informative way to explain the work.

After defining how to create the information-files preliminary manuals were made. The observations and validation steps from Yaskawa employees, the academic supervisor and fellow students led to further development. Feedback was taken into consideration, and once the methods were defined, it was time to create the final manuals. Guidelines regarding how to create content specifications were found. Hackos (1994) listed some major steps of how to direct the work for documents.

- Understanding the purpose of the specification process
- Determining goals and objectives
- Analyzing audience and environment
- Developing a detailed task analysis
- Analyzing organizational strategies

The first three steps were mainly done in the early step of the project. During these steps where the software was learned, the students understood which steps were complicated. Further interviews, observations and validations led to even more information of what needed to be included in the manuals. For the fourth step Hackos (1994) mentions that in the design of documents that will be used to operate, administer, or maintain a product, in this case, a manual, a task-oriented approach to information is essential. Task-oriented information will:

- Reduce training time
- Enhance productivity

- Decrease customer-support costs
- Increase audience satisfaction with information

How to establish what needed to be included in the manual was also done in earlier steps. The author mentions that it is highly important the document must be constructed from the viewpoint of the audience. The last step is about the final stage of development. This step is mostly about the publication of the document which the students are not involved with. The author does, however, mentions that the meaning is to shape the product as rational as possible by correctly analyzing the information collected.

When it had been chosen to create a text-graphics manual, the work of how to best present it regarding the ratio between the text and pictures began. Graphics play an important role in making instructions user-friendly (Inaba, Parsons & Smillie, 2004). The role of graphics is to show the item of interest for each step and how it looks. Uses of locators orient the user, and detailed views highlight the essential steps. A common mistake in software instructions is to use too much graphic, which makes it cumbersome to use accordingly to the authors. They further listed some ground rules for the use of graphics in instructions, the most relevant to this project are:

- Use a locator and detailed view to show a detail difficult to see on the bigger view
- Avoid unnecessary details or graphics
- Include each item referenced in the text in the detailed view

A visit was also made to Lin Education, at their diversion Loops. It is a company based in Gothenburg who works with digital learning and teaching material. Their mission is to unite pedagogy and technology in order to make learning more efficient and more meaningful with the power of digitization. Their specialization is teaching larger groups in a subject, everything between kindergarten to larger corporations. From the target group, they work with finding the best tool for learning and how to involve everyone.

Even though this project aim is pretty specific and narrow, they helped the students with some expertise in how to think with teaching material. This project is mainly about finding the best method to work and to create manuals for that, not to teach the user an entire software. The manuals take the user through the steps of implementation, but the more explanatory the manuals are, the more the user could learn the subject. However, this is a hard balance, a manual should be kept as short as possible but still be explanatory.

The manuals are mainly instruction manuals, more than a teaching material. However, if these manuals could be used as a learning tool for further knowledge in MotoSim and still be simple to use that would be optimal. This balance between the content and usability were discussed with Lin. The most important thing to have in mind, according to them, were knowing the audience and users of the product. A few keynotes should be included in the structure, according to Lin:

- The aim of the manual
- The purpose of the manual
- The steps of what is happening, a description and a timeframe
- Content that the users keep with them

- A perspective of how to move on and evolve in the future

They further mentioned the four knowledge expressions essential to learning: fact, understanding, skill and familiarity. Accordingly to a learning designer at Lin, it is important that all of them are included. For technical instructions like this project, many of them are to fact-oriented and forgets to involve the other features.

Fact and understanding are about how the steps are explained and understood with the texts and graphics. The familiarity aspect is regarding recognition and how the product feels. It is associated with physical experiences, something that is felt and something that the recipient knows (Björklund, 2016). Skill, how experienced the users are with the application, is something that needs to be defined.

7.2 The final creation of the manuals

Once the methods was defined, and the preliminary manuals was tested, the final manuals were made. The feedback from the preliminary drafts consisted mainly of the complexity. Some steps were not explained thoroughly enough, while others were too much. It was also decided that the new manuals should have some introduction explaining the meaning and purpose of the methods. After the discussions with Lin, the students realized the importance of an explanatory introduction describing the aim and why this method and manual should be used. An introduction long enough that the users could grasp the meaning but still be short enough that it is being read.

Decisions were also made whether the manual should be divided into chapters or just by steps. Division of chapters looks very structured, but for manuals this short it would feel rather excessive. It is also hard to divide the steps into chapters, and it would not help the flow of the manual in any way. A step-by-step manual with general descriptions of the different sections instead of chapters gave the best flow and still a good design.

After comparing with a couple of Yaskawa manuals, a certain design was elected, and a design proposal was sent to the students as an idea of how it could look. This was considered more as a guideline than demand but still something to follow. The functionality of the manuals, regarding the pedagogy and usability, was considered more important than the design. However, a clean document with a design that could relate to Yaskawa's brand was desired. Figure 7.1 below shows the chosen design for the manuals.

Figure 7.1: *Design of the manuals. The authors own picture.*

The program chosen for making the manuals was Microsoft Publisher. Since the student's knowledge of design work was limited, a program that was easy to use but still gives a professional outcome was desired. Microsoft Publisher is an easy program to use, similar to other office programs which the students were familiar with. The choice therefore fell on Publisher to use for creating the final manuals.

7.2.1 Manual 1: Create a robotinf-file

Since this project includes how to create a script for both one- and two-axis positioners, it led to a few problems since the structure differ slightly between these two types. The students, however, decided that one manual could be used for both courses of action. With different templates for the one- and two-axis positioners the steps of how to gather the data for coordinates are orientation are the same. The differences are that the one-axis positioners have fewer axes and model files. The name of axes also differed between the positioners, but otherwise, the logic is similar. In the steps that differed explanatory notes were added.

To distinctly note the differences with explanatory texts worked great. Validations with users led to clarifications that led to a manual who functioned for all one- and two-axis positioners. The manual, which is intended for Yaskawa-employees only, got suited for their corporate language and knowledge. Final touches on the design and layout led to the final result, which is presented in Appendix A.3.

7.2.2 Manual 2: Insert positioner with robotinf-file

When initializing a controller, there is a couple of steps during the setup that is out of this project's limitations. Most of the parameters which are implemented during the initialization are not relevant for controlling the kinematics. It was first considered to leave these part out of the manual, but to make a comprehensible manual the group decided to include these steps. It would look slightly negligent if the manual ignored several steps, even though that is how the implementation is made. Some of the parameters cannot even be entered through the software but still has to be worked through. However, since it is intended to be in practical use for both Yaskawa employees and their customers, it was decided to include all the steps.

To increase the knowledge and to prevent users from getting stuck, explanations of all parameters were included, but explanatory notes stated the fact that all parameters except rotations of the axes could be skipped. Final touches on the design and layout led to the final result, which is presented in Appendix A.4.

8
DISCUSSION

A problem with this project was the lack of theories and well-established methods to work by, both concerning how to simplify an existing working process and the later work of developing the manuals.

Regarding the work of developing the working process, a problem occurred when it was noticed that the knowledge from Yaskawa employees regarding the aim of this project was less than expected. Help concerning how to operate in the software existed, but about the content of the script, no help was to be found. The upsides of this were that the students obtained a deep understanding of the script, which led to a variety of possibilities regarding the structure. Learning by doing is as mentioned the best way to learn a subject. The lack of structured methods to work by, do however, from an academic viewpoint, seem a bit inadequate.

About the design of the final manuals, the theories used were helpful, but still many decisions had to be determined by the students. The amount of descriptions and the ratio between pictures versus text is difficult to balance. Even though interviews and observations from a broad variety of people were collected the criticism and opinion from these varies, especially regarding the design of the manuals. The results which were presented, manuals regarding this new method is structured by the students and therefore, somewhat subjective.

Concerning the data collection through interviews, observations and validations, some opinions could also be mentioned. These three were often intertwined. To look at a person using the manual helped the students see the flaws, both regarding the design and the content. The interviews which followed were discussions regarding the user's opinions and what was noticed from the students. This was something that was hard to plan ahead and prepare questions for. The discussions varied between different users and different steps of the process. Some people were observed and questioned over the method, others of the different parts of manual and some people just assessed the design. The validations towards the final step of the process were done through interviews and observations. This was a clarification that the manuals presented reached the goals set at the beginning of the project, and later could be used by Yaskawa and their customers in the future.

The students had an opportunity to try to implement a positioner in RobotStudio, a similar software developed by another robotics manufacturer, with the help of Per Nykvist. The reason for this was to study the differences and other possibilities of how to implement external axes. This software and the method of implementing positioners differed a lot, therefore the group were not able to find a better way of working in MotoSim using this method. Since learning a new simulation-software is very time consuming, particularly

with peculiar units as positioners, the lack of time for this project made it impossible to investigate other software further. Instead, the time was spent to understand MotoSim fully.

9

CONCLUSIONS AND RECOMMENDATIONS

To summarize, the progress of this work has been separated into two parallel steps that later merged to a final result in the form of two manuals. First off, learning MotoSim and developing the script, which could be seen as the working process. Secondly, the process of developing and making the manuals, which could be seen as the design process.

The progress that later resulted in the working method presented in the manuals originated from the understanding of how the hierarchy was defined in the script. When the comprehension of the hierarchy and the different variables was obtained, different scripts could be presented. Discussions with Yaskawa and verification with users enabled the students to a decision of how the files are created with a template that was sent to Yaskawa. The final manuals are presented in Appendix A.3 and Appendix A.4

For the future, the manuals could now be used in the further process of education and increasing the usage of MotoSim. Standardization of this implementation, which both simplified and significantly reduced the time, is valuable for both project leaders, salespeople and robot designers at Yaskawa. If there were more time for this project, the students would have been interested in more experiments with scripts for positioners containing more axes and perhaps even conveyors. However, this project has enabled future work for this since the variables and the structure of the script have been defined.

The method of working with robot-inf files has resulted in a more quality assured, standardized and simplified working method. Reducing time has been one of the main targets. Times of an inexperienced user was taken comparing the old and new working process. Creating a robotinf-file using Manual 1 took less than 10 minutes, and the implementation using Manual 2 took about 5.5 minutes. The implementation using the old working process took about 9.5 minutes, with active help from the students. Once the files have been created every implementation has great time savings. The 4 minutes saved is the time of implementing the model files, define the axes and structure the hierarchy which the robot-inf file does automatically. The old process is furthermore very error-prone, which most often results in even more time. The times of implementation is also shortened with a more experienced user.

After the results were presented to Yaskawa, the students were given the opportunity to test their skills and knowledge on the program and script on a five-axis positioner. Although this is outside the limitations of the project, it was something the students wanted to test to see if the presented results enabled further work for multi-axis positioners. Currently, it is only possible to insert two external axles in MotoSim, as mentioned in chapter

4.4. This makes the work of robotinf-files difficult for more than two external axes. The five-axis positioner on which the work was carried out can most easily be explained as two separate two-axis positioners on either side of a glare protection, which is placed on a rotatable base. This is often used for welding operations, where an operator handles the object on one side, and the robot operates on the other side of the protection.

The students tried adding a Turn-1 for the rotatable base which operates the same as for a one-axis positioner. Furthermore, two Turn-2 were added for the positioners on each side. The students tried their method of tying all files and axes to the world, similar to the manuals presented. The differences however occurred when multiple scripts in different folders were used, and the hierarchy became more difficult to structure. Although, MOV and %RBNAME% could still structure the hierarchy by adding both two-axis positioners as children to the base. The name of the first axis, in this case the base, is where the other two rotation centres are connected. This name must be initialized in the controller setup and be referred to in the scripts of the two-axis positioners. Thus, a five-axis positioner could be implemented using three separate robot-inf files.
This clearly verified that the results of this project, containing the explanations and working method through the robotinf-file is fully functioning with multi-axis positioners. Although, when more than one script is being used, it is hard to create a standardized working process. An experienced MotoSim user could, however, with the manuals and results regarding the script, continue to experiment and standardize the work with multi-axis positioners.

Some recommendations and ideas for future work are further standardization with working methods and processes in MotoSim. While interviewing Yaskawa employees, the students noticed that different software and methods were used. Standardized methods of further implementations and robot-programming with manuals could reduce the time and consequently, the labour costs. Some involvement in making more employees familiar with MotoSim is recommended. A few thoughts regarding the software also led to some improvement proposals for MotoSim. During the many hours spent on the software, the students listed a few notes on the software that could be fixed:

- If the viewpoint is changed, this is saved as an action. When trying to undo a certain action, the software only undoes the last view change. The ability to undo a viewpoint is unnecessary, consider to change this feature.
- To right-click on a specific model or axis in the CadTree, the object first has to be highlighted by a left-click. This differs from other software products, consider to change this feature.
- In the pendant, a mouse cannot be used and has to be operated through the keyboard. Many users found this disturbing and to operate the mouse in this would have further simplified the initialization of the controller, consider to change this feature.
- The students find no reason why no more than two external axes could be implemented during controller initiation. This would enable a single robot-inf file for multi-axis positioners.

Bibliography

Björklund, L. (2016). *Att medvetandegöra det omedvetna: de fyra F-n och andra kunskapsbegrepp i skola och i forskning om lärande.* Linköpings universitet.
From: https://liu.se/cetis/konferenser/documents-tis2016/kalmar-lars-b-kunskap.pdf

Dogra, S.K. Randhawa, H. (2017). *Symmetry and Group Theory in Chemistry (2nd Edition).*
From: https://app.knovel.com/web/view/khtml/show.v/rcid:kpSGTCE002/cid:kt011QKMC1/viewerType:khtml//root_slug symmetry-group-theory/url_slug:cartesian-coordinate?kpromoter= federation%23%3F&kpromoter=federation

Education Corner. (n.d.). *The Learning Pyramid.*
From: https://www.educationcorner.com/the-learning-pyramid.html

Fast-Berglund, Å., Mattsson, S. (2017), *Smart Automation: Metoder för slutmontering.* Lund: Studentlitteratur.

Gales, C.C. (2017). *Robot Simulation: A Tool For Project Success.* Automate, 2017, Detroit.
From: https://www.automateshow.com/search-results.cfm?q=Gales&sa=SEARCH

Hackos, J.T. (1994). *Managing Your Documentation Projects.* New York: John Wiley & Sons, Inc., (Chapter 10)

Inaba, K. Parsons, S.O. Smillie, R.M. (2004). *Guidelines for developing instructions.* Boca Raton: CRC Press.

International Federation of Robotics. (2018). *Executive Summary World Robotics 2018 Industrial Robots: Robot Sales 2017: Impressive growth.*
From: https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf

Lamb, F. (2013). *Industrial Automation: Hands-On.* New York: McGraw-Hill Education.

Miller, R.K. (1989). *Industrial Robot Handbook.*
From: https://link.springer.com/content/pdf/10.1007%2F978-1-4684-6608-9.pdf

Nubiola, A. (2015). *The future of robot off-line programming.*
From: http://coro.etsmtl.ca/blog/?p=529

Nykamp D.Q. (n.d.). *Cartesian coordinates.*
From: https://mathinsight.org/cartesian_coordinates

Owen-Hill, A. (2016). *What Are the Different Programming Methods for Robots?.*
From: https://blog.robotiq.com/what-are-the-different-programming-methods-for-robots

Pichappan, P, Ahmadi, H, Ariwa, E. (2011). *Innovative Computing Technology: First International Conference, INCT 2011, Tehran, Iran, December 13-15, 2011. Proceedings.*

Smashingrobotics. (2016). *Most Advanced Robotics Simulation Software Overview.*
From: https://www.smashingrobotics.com/most-advanced-and-used-robotics-simulation-software/

Umeå University. (n.d.). *Geometriska transformationer.*
From: http://www8.tfe.umu.se/courses/systemteknik/Multimed2/mm2_99/Bok99/4_5/Kapitel45.htm

Yaskawa. (2017). *Motoman DX200: Industrial robot controller* [Product sheet]. Yaskawa.
From: https://www.yaskawa.co.uk/index.php?eID=dumpFile&t=f&f=11192&token=efe3c920a0b9c9af3992ac23ce1c5b8b7fc68d89

Yaskawa. (n.d.). *Positioners.*
From: http://www.motoman.lt/products/positioners/?no_cache=1

Pictures

Kuropatwa, D. (2007). *Learning Pyramid.* [Electronic Picture]
From: https://www.flickr.com/photos/dkuropatwa/2097911609

A

Appendix

A.1 HTA, old working process

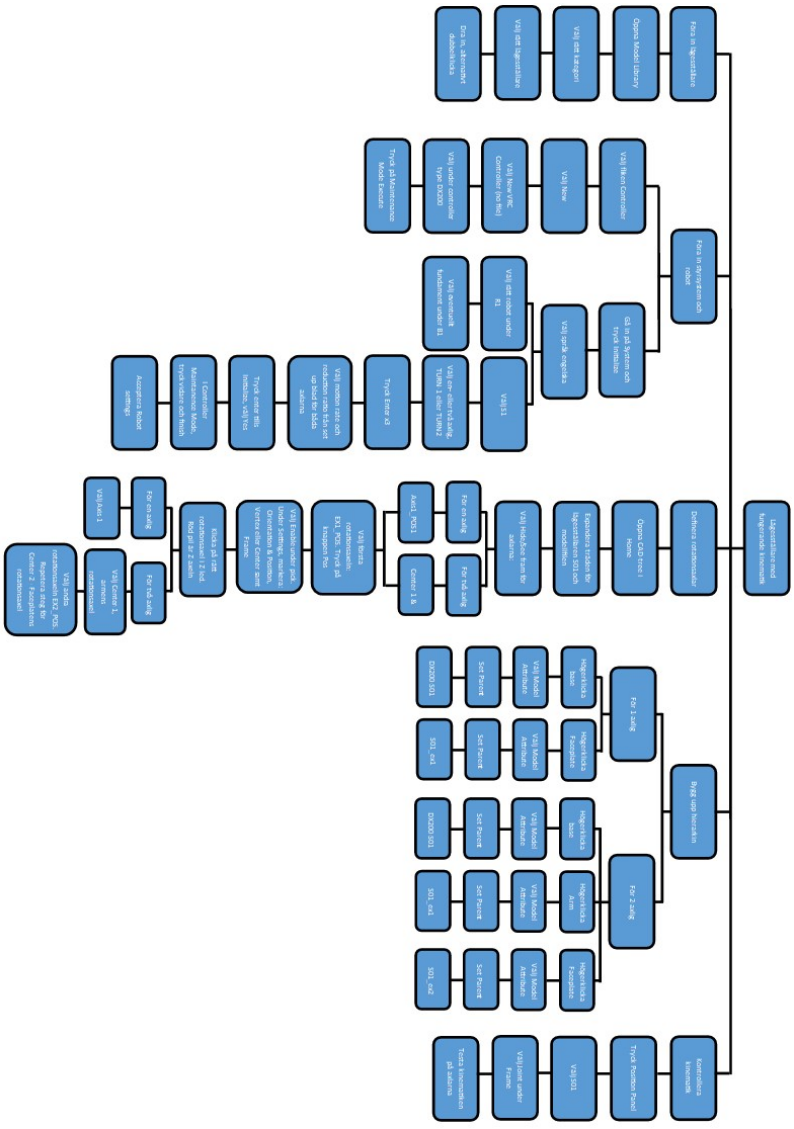


Figure A.1: *HTA, old working process. The authors own picture.*

A.2 HTA, new working process

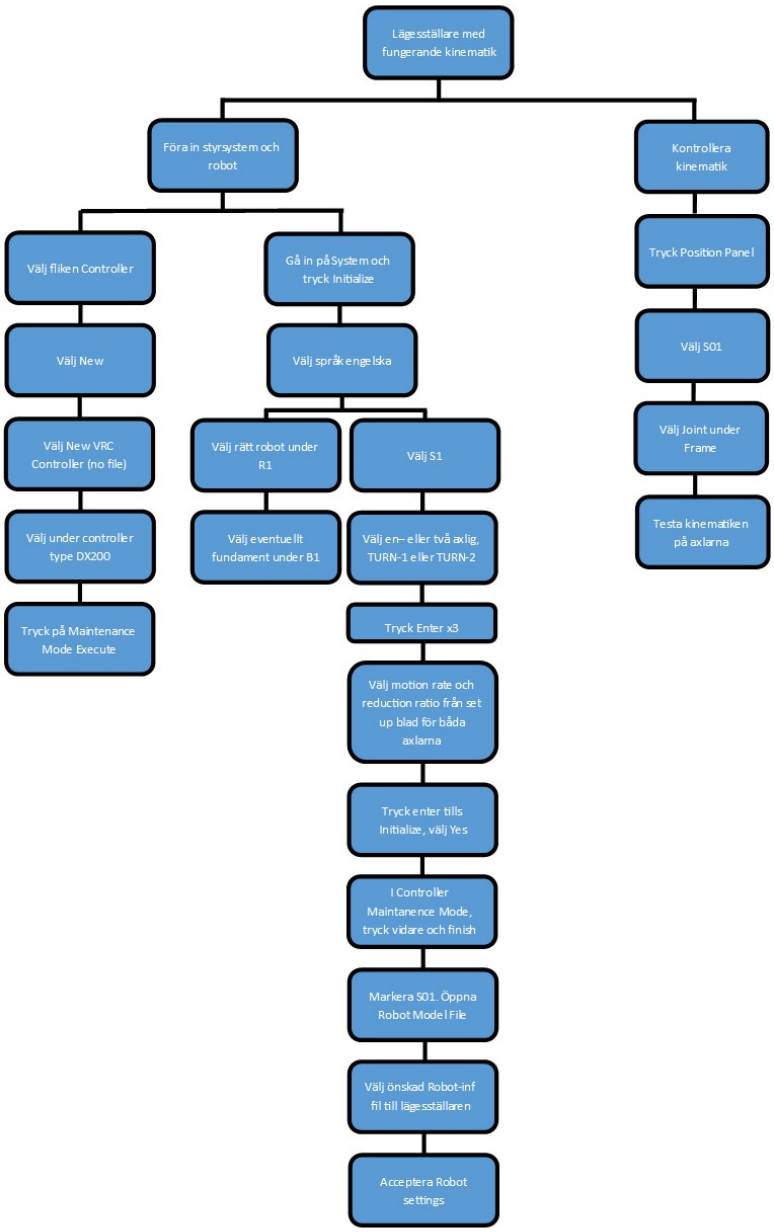


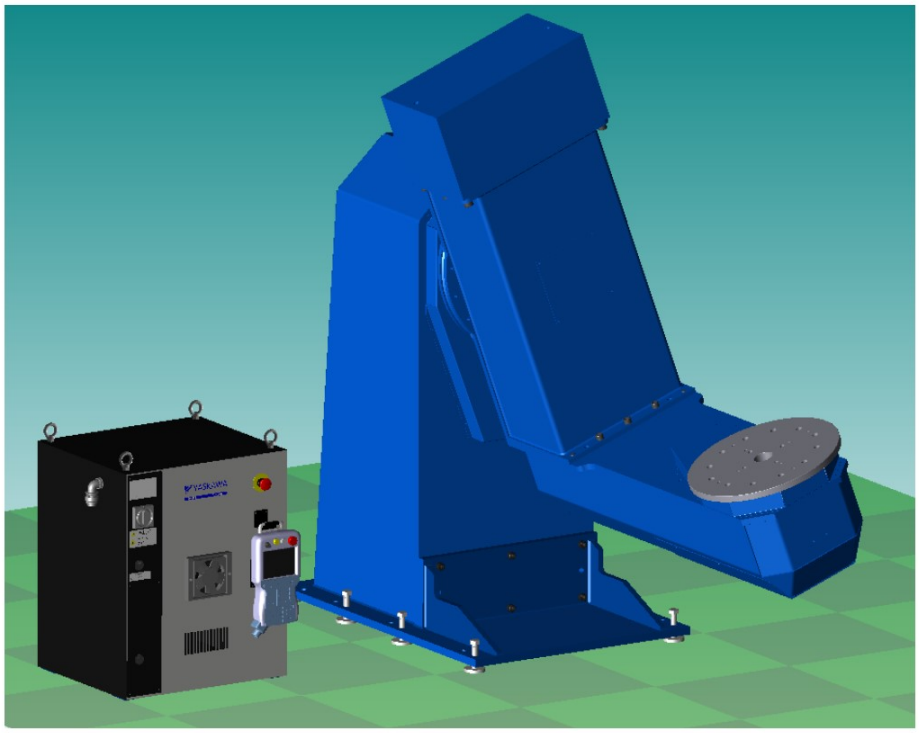
Figure A.2: *HTA, new working process. The authors own picture.*

A.3 Manual 1

YASKAWA

Create a robotinf-file

Instruction Manual



Page 1

YASKAWA

Introduction

This manual will take you through the steps of creating a robotinf-file for MotoSim EG-VRC. These files will simplify the process of implementing positioners and consequently reduce the time for setting them up. It is a text-graphic manual following all steps and will be able to be used by someone with minimal experience of MotoSim and moderate computer knowledge. As a complement, the manual “Insert positioner with robotinf“ could be used to verify that the file has been created correctly.

This method functions for one and two-axis positioners by working with the following templates created for this implementation. The templates are structured so that the user only needs to insert the coordinates and names of the hsf-files for the desired positioner. Two separate templates have been created, for one and two-axis positioners. The manual is a step-by-step guide of how to find and implement relevant data that needs to be entered into the templates.

YASKAWA

In these first steps we relocate the robotinf template to the correct folder and change the searchways so correct modelfiles are loaded.

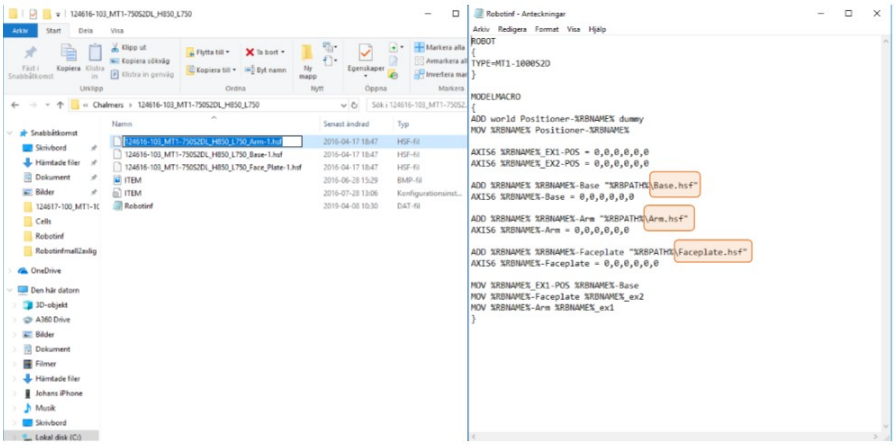
Step 1: Locate the corresponding robotinf-template regarding the number och axes. Copy the file to the disired positioner folder in **Model Library** containing the .hsf files. Open the template with notepad or similar text editor.

The filename of the template cannot be changed.

Step 2: For each hsf file, copy the entire file name including the .hsf ending.

Paste the filenames after ”%RBPATH%\ in the template marked below and replace the current names. Do not erase the citation marks.

Note: Make sure that the selected file name is pasted to the corresponding part into the



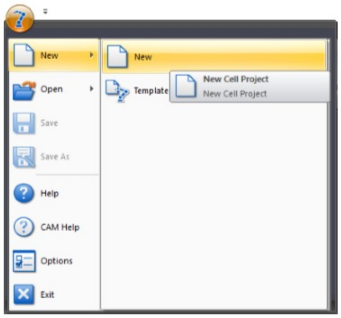
For two-axis positioners, three file names have to be entered, and for one-axis positioners, two file names have to be entered.

YASKAWA

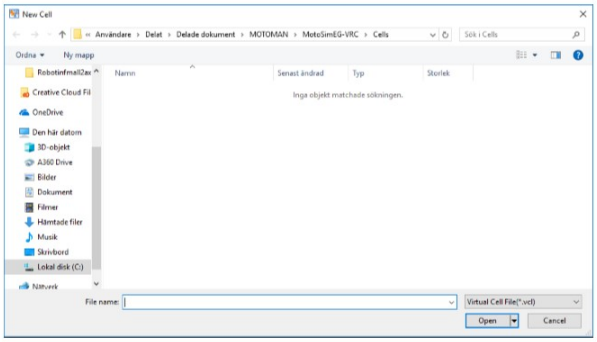
In the following steps we start a new project in MotoSim and insert a posistioner

Step 3: Open MotoSim EG VRC.

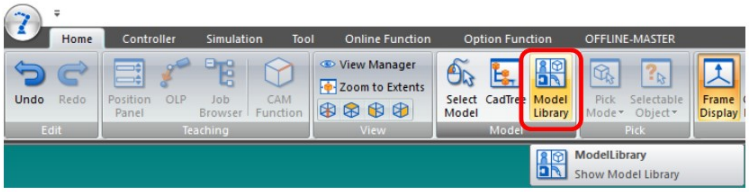
Step 4: Start a new cell project by clicking on the blue robot in the top left corner, select **New** and then **New** again.



Step 5: Write a desired name for the cell project after file name. Press **Open** to create the project



Step 6: Under the Home tab, left-click on the icon **Model Library**

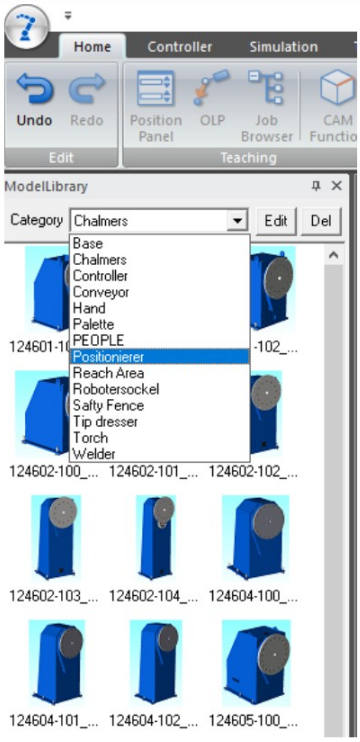


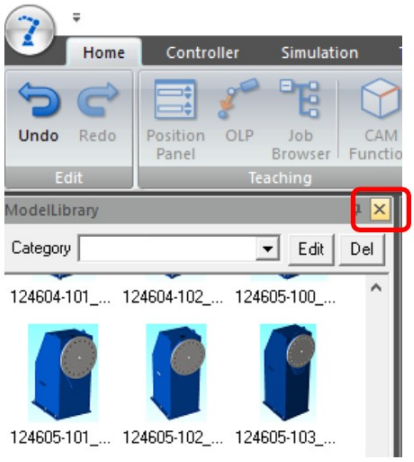
YASKAWA

Step 7: In the **Category** drop-down list, choose the category containing the positioners.

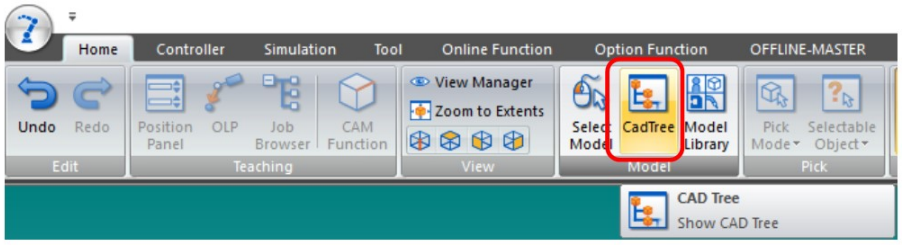
Step 8: When the preferred positioner is found in the list, simply double click the model or drag and drop the model in the open space.

The positioner will now be placed in world origin. Close the model library tab by clicking x in the top right corner



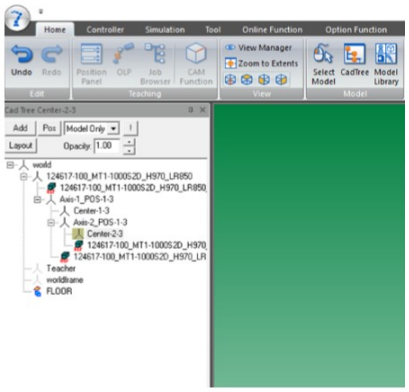


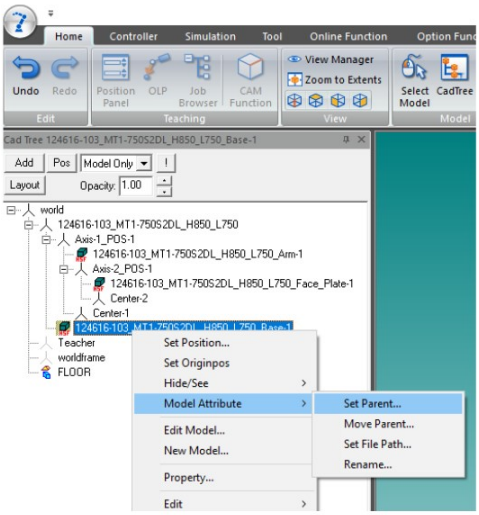
Step 9: Select **CadTree** in the Home tab.



In the following steps we define the hierarchy temporarily. This make it easier to read the coordinates for each part later on.

Step 10: Expand the tree structure in the left toolbox by pressing the ⊞ sign. This will enable to show all the components of the positioner.





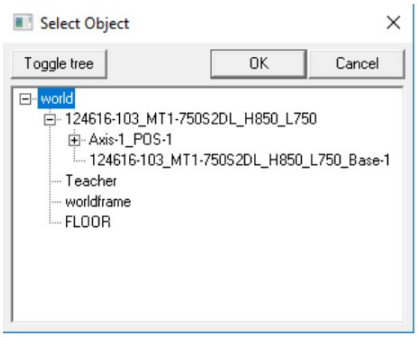
Step 11: Now right click on a model file in the CAD-tree marked as

Select **Model Attribute** and click on **Set Parent**.

Step 12: In the pop-up window Select Object, select **world** and press **OK** to set it as parent for the model file.

Repeat step 11 and step 12 for all model files and also for Center 1.

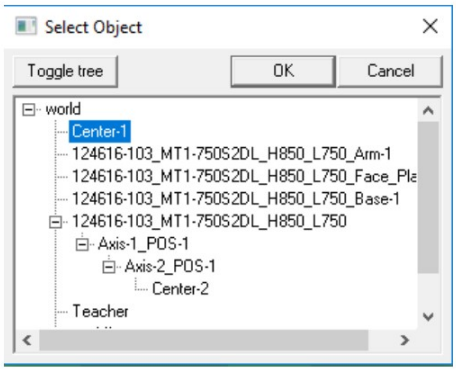
Note: If it is a one-axis positioner, the axis is named Axis-1_POS-1.



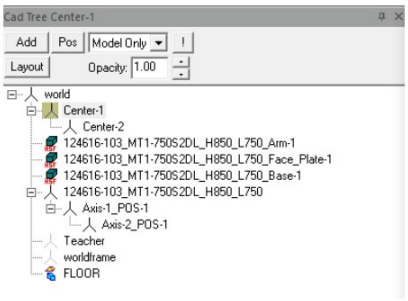
YASKAWA

Step 11: If it's a 2-axis positioner, repeat the last step and set Center 1 as parent for Center 2.

Chose Center 2, set parent in "Select Object", mark Center 1 and press **OK**.



Once the hierarchy is defined, it is time to update the coordinates in the robotinf file. When all the models is tied this way, the coordinates of them are relative to the world origin.



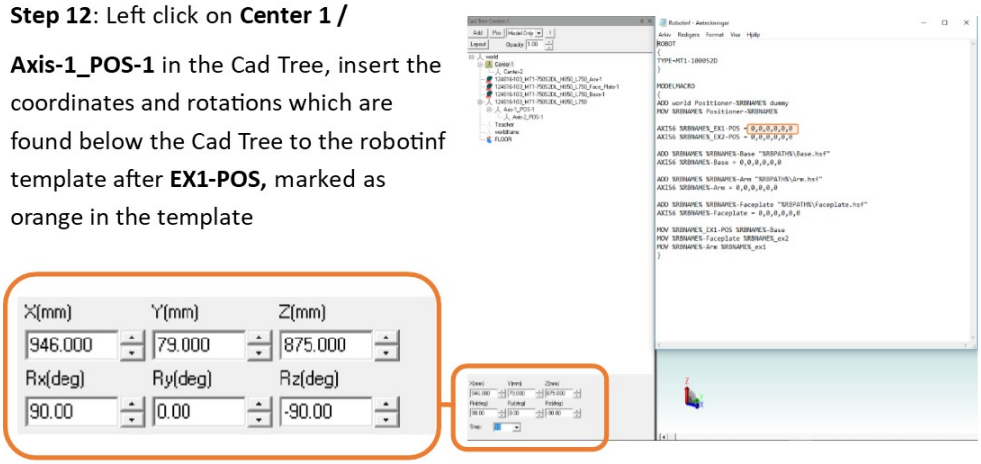
YASKAWA

In the following steps we update the robotinf template with the coordinates of all necessary parts for the positioner.

***Note**: In the template all coordinates and rotations are set as zero, which is the world origin. With the hierarchy defined this mean that all models and axes could be defined by reading the data from MotoSim.*

The values are x, y and z and the rotation around theese axes; Rx, Ry and Rz. The coordinates entered in the template comes in this order; x, y, z, Rx, Ry, Rz. These values have to be separated by a comma.

Step 12: Left click on **Center 1 / Axis-1_POS-1** in the Cad Tree, insert the coordinates and rotations which are found below the Cad Tree to the robotinf template after **EX1-POS,** marked as orange in the template

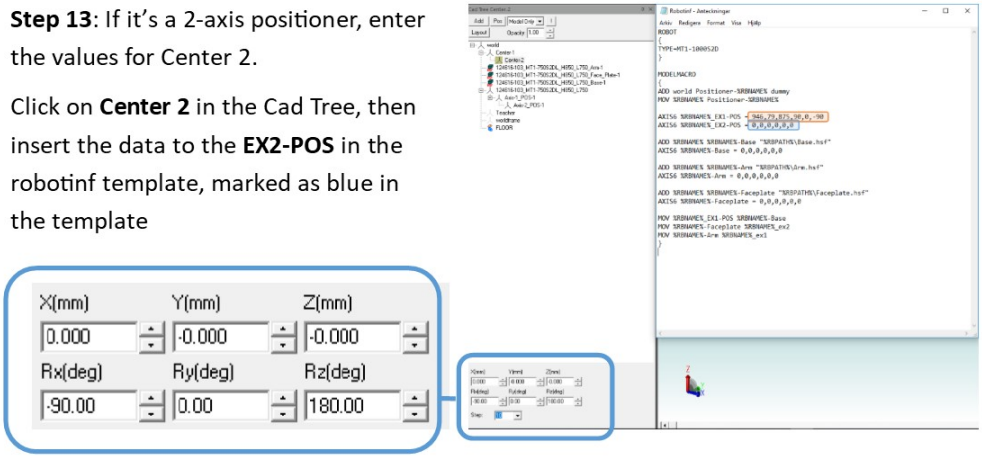


***Note**: If the coordinates or rotations have a decimal point, this has to be entered into the template with a dot.*

YASKAWA

Step 13: If it's a 2-axis positioner, enter the values for Center 2.

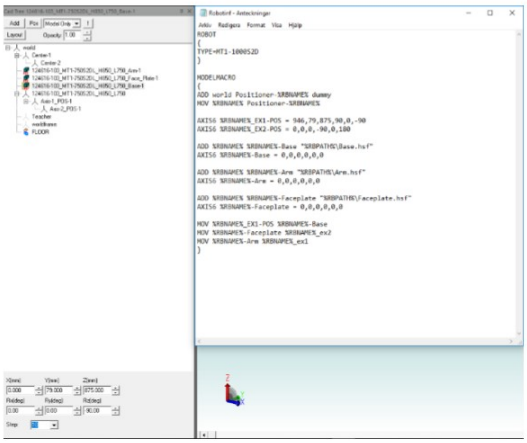
Click on **Center 2** in the Cad Tree, then insert the data to the **EX2-POS** in the robotinf template, marked as blue in the template



Step 14: Now we define the origin of the modelfiles

Select a HSF file  in the Cad Tree. As in the previous step, read the position data below the Cad Tree and enter in the robotinf template. The orientation of the model file must be entered on its right place to the coresponding HSF file in the template.

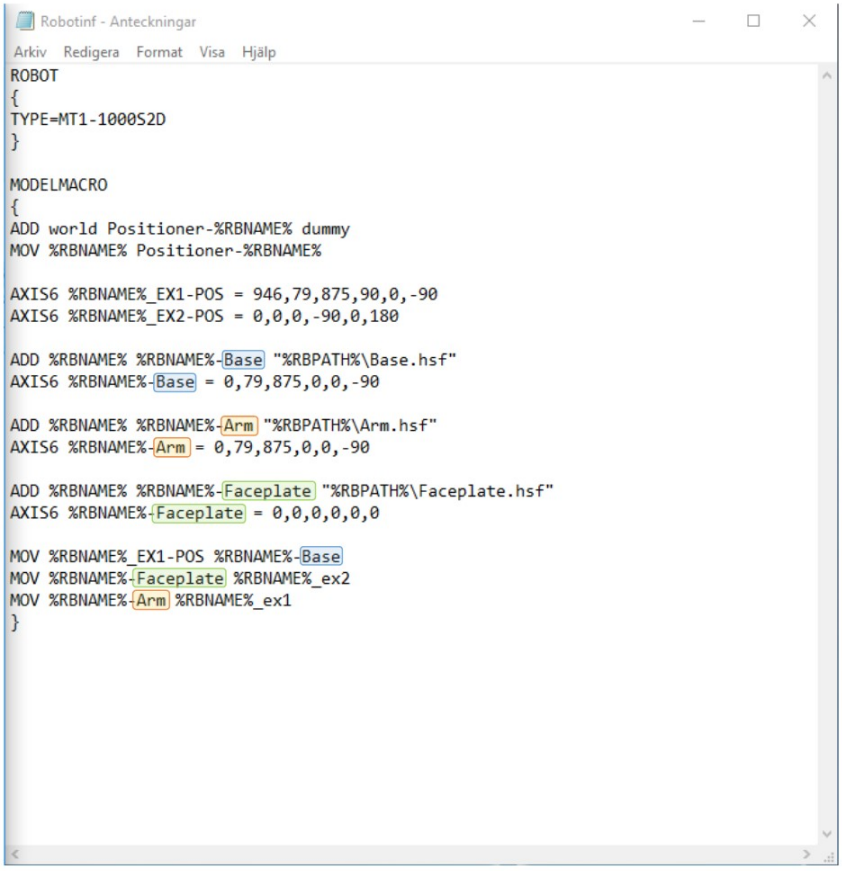
Repeat for all models until all coordinates has been entered to the



YASKAWA

If it is desired to change the names of the model files in MotoSim, it is possible but not necessary for this script to function. If not desired, move on to the next page.

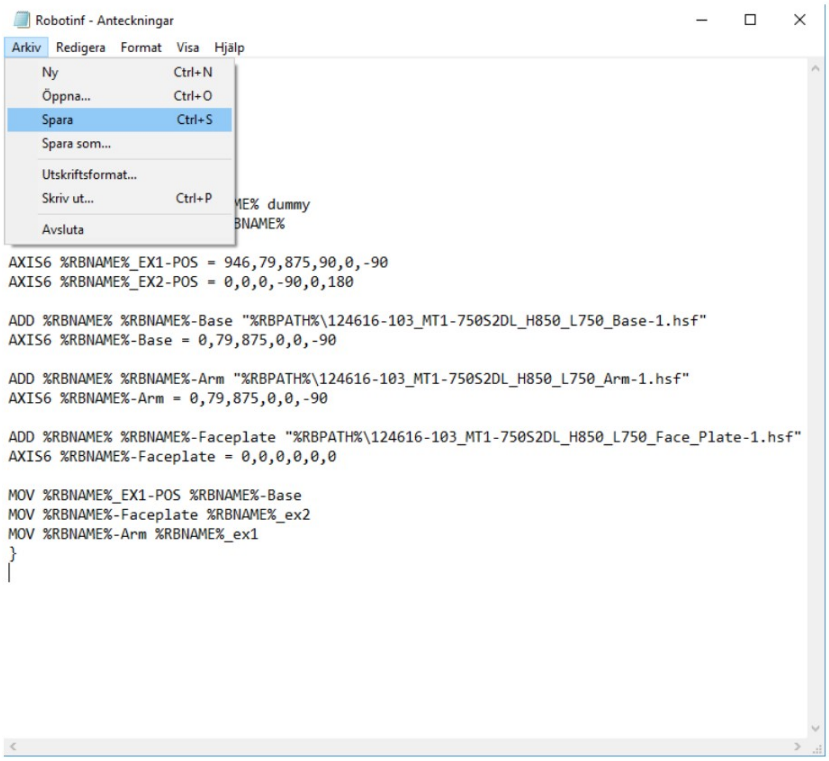
Step 15: Changing the name of a model needs to be done in three different places, marked as blue, orange and green below. The name need to be exactly the same in all three places. This enables the script to load the model file with the right coordinates and hierarchy.



YASKAWA

Step 16: When all the coordinates and rotations are entered, save the robotinf template by clicking File and Save.

Note that the template need to be saved in the same folder as the HSF files



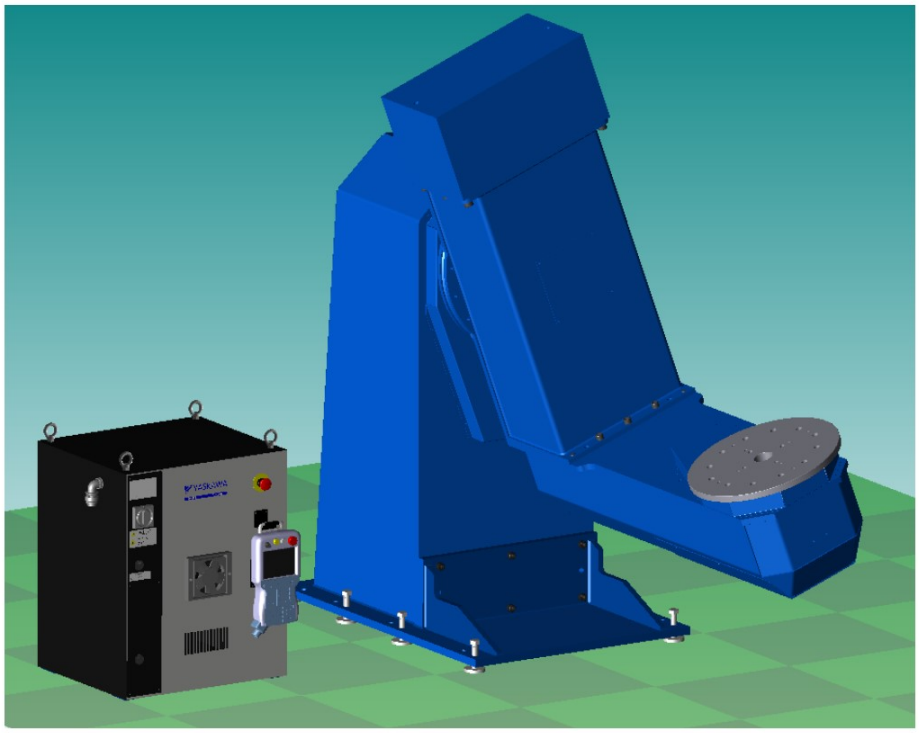
You're now finished, use the manual "Insert positioner with robotinf" to varify that the file is fully functioning

A.4 Manual 2

YASKAWA

Insert positioner with robotinf

Instruction Manual



Page 1

YASKAWA

Introduction

This manual will take you through the steps of using a robotinf-file for MotoSim EG-VRC. The file simplifies the process of implementing positioners and consequently reduce the time for set up. With existing robotinf-files, a positioner with functioning kinematics and hierarchy is added.

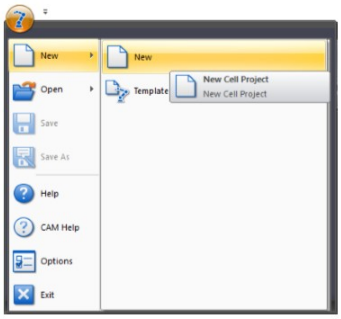
It is a text-graphic manual following all necessary steps and will be able to be used by someone with minimal experience of MotoSim and moderate computer knowledge. This manual is also used as a complement for the manual “Create a robotinf-file”, a manual describing how the files are created as a verification that the files are functioning. This method functions for one and two-axis positioners. The manual is a step-by-step guide of how to operate through the controller initialization and which parameters that need to be inserted.

YASKAWA

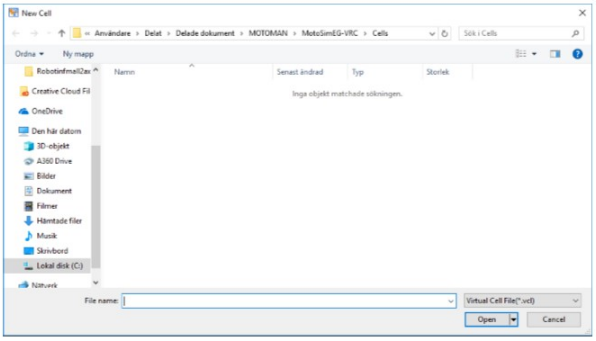
In the following steps we start a new project in Motosim

Step 1: Open MotoSim EG-VRC.

Step 2: Start a new cell project by clicking on the blue robot in the top left corner, select **New** and then **New** again.



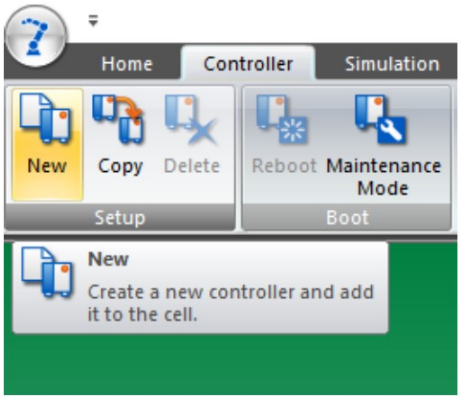
Step 3: Write a desired name for the cell project after file name. Press **Open** to create the project

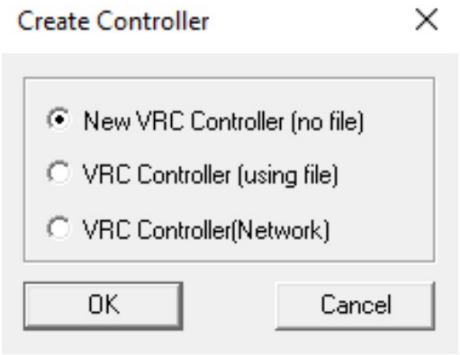


YASKAWA

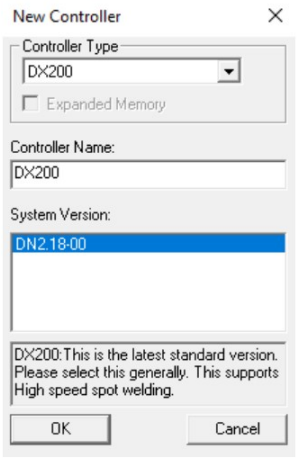
In the following steps we create a new controller in MotoSim

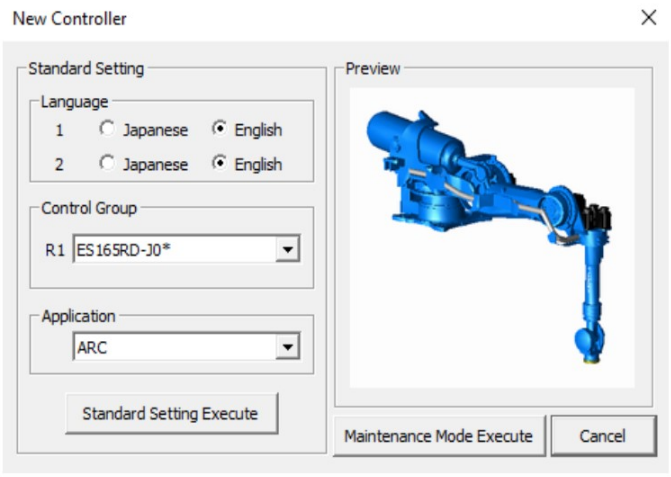
Step 4: Enter the tab Controller. Create a new controller and then choose **New VRC Controller (no file)**. Accept with **OK**.





Step 5: The tab opens where the Controller Type is selected. Choose the **DX200** as Controller type and press **OK**. A new tab opens where language is choosed, robot and application. Once selected press **Maintenance Mode Execute**.

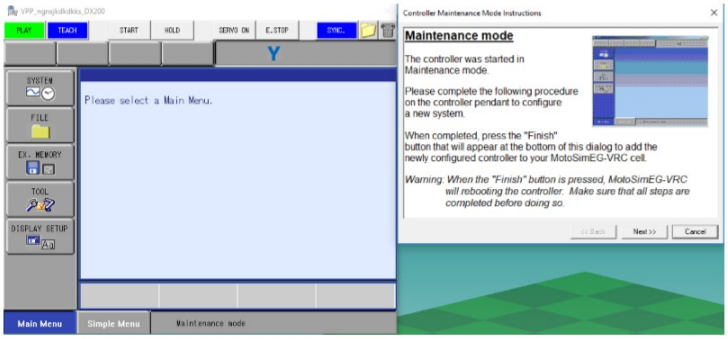




YASKAWA

In the following steps the controller is initialized and positioners are added. Also how to orient through the pendant is explained.

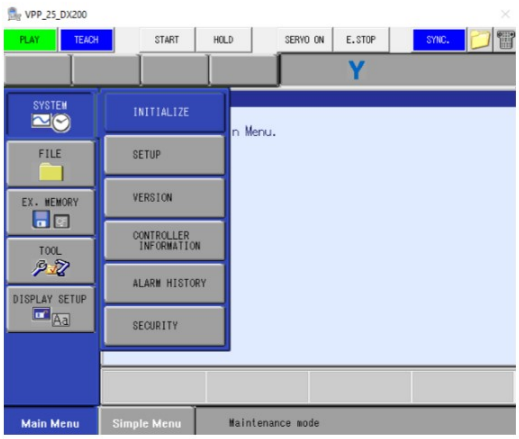
Step 6: Following windows will show up, the left one is the virtual pendant and the one to the right is the installation window. Do not close any of these during the installation

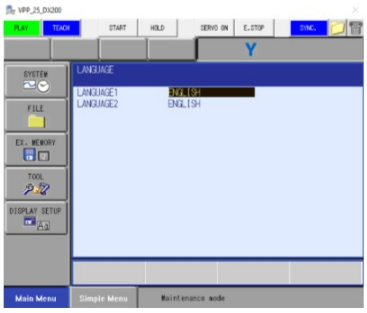


Step 7: Click on **System** in the pendant, then choose **Initialize**.

Chose the prefered language using the keyboard, explained in the note window, and move to the next stage.

Once initialized, the pendant has to be operated with the keyboard and can't be operated with a mouse. The *arrow keys* are used to orient, the *spacebar* to choose an option and the *enter key* to move on to the next stage. Press *Esc* to return to previous step

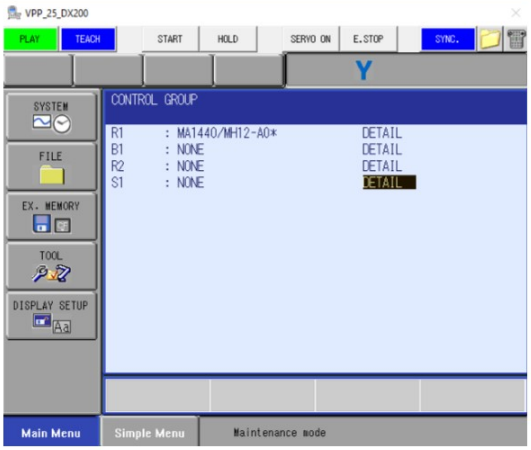




YASKAWA

Step 8: In the Control Group window, the type of robot, conveyors and positioners can be added.

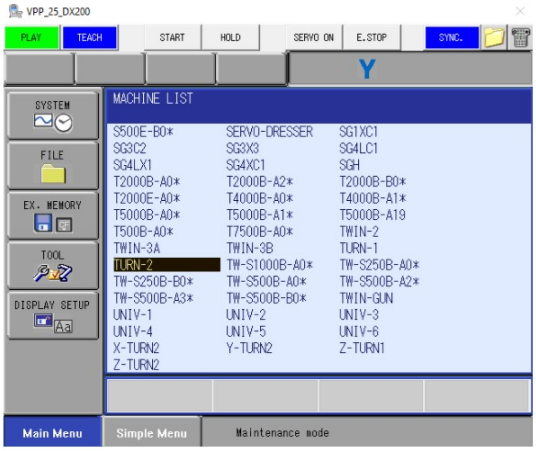
Go to **S1** and press space to choose the type of positioner regarding the number of axes.



Step 9: For 1-axis positioners choose **TURN-1** and for 2-axis positioners choose **TURN-2**.

Note: If an additional positioners is to be added repeat step 8 and step 9 for S2, S3 and so forth.

When the desired amount of positioners are added, continue to the next window



YASKAWA

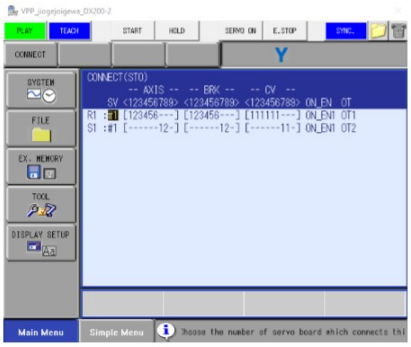
In the following steps the data of the positioners is added. If the aim of the implemtation is layout planning the only relevant data is the rotation of the axes. This is set in Step 12 and it is necessary to add rotation for the kinematics to function.

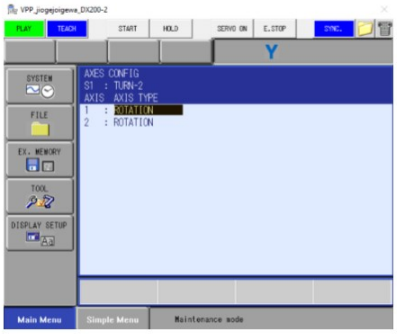
Data as offset, reduction ratio, motor type etc plays no essential role for the simulation. If the aim is offline simulation with further practical usage theese has to be added correctly. The data are found in the setup manuals for the positioners.

Step 10: Insert data for the servo-settings accordingly to the setup manual

Inställning hur man frånkopplar axlar i systemet

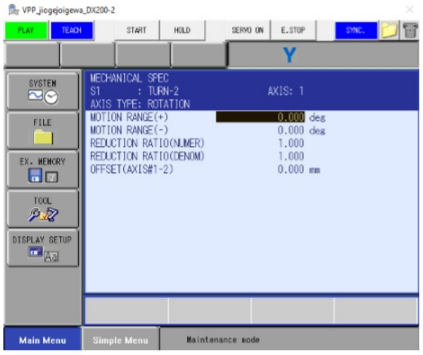
STO=Safe torque off





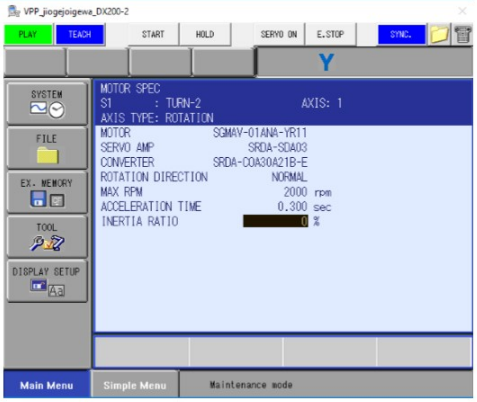
Step 11: Set the axis-type for each positioner axis accordingly to the setup manual. One for each positioner added.

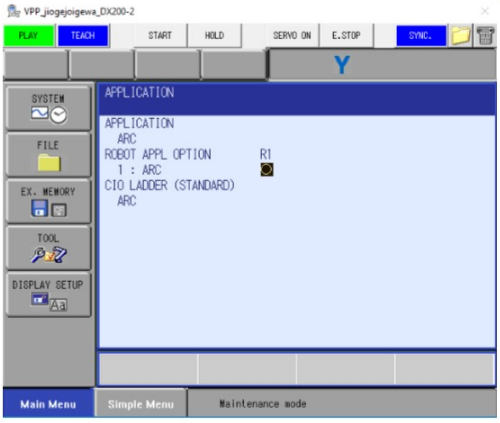
Step 12: Type the Mechanical Specification for each axis to the positioner accordingly to the setup manual. The Motion Range sets the limits of rotation of the axes in degrees



YASKAWA

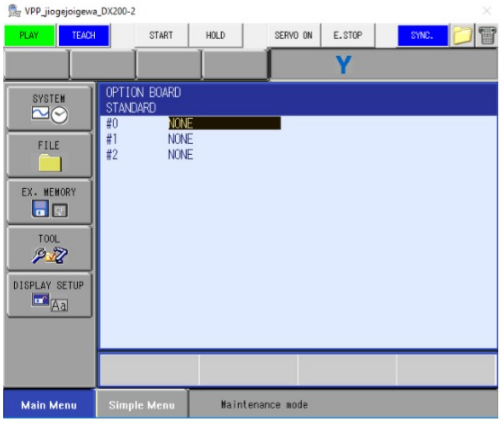
Step 13: Insert the motor-specification for each axis accordingly to the setup manual





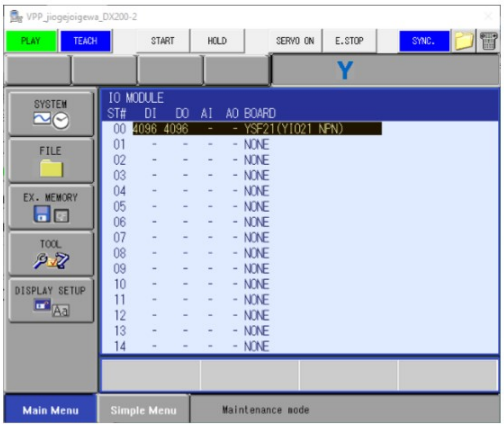
Step 14: Choose the specific application for the robot(s)

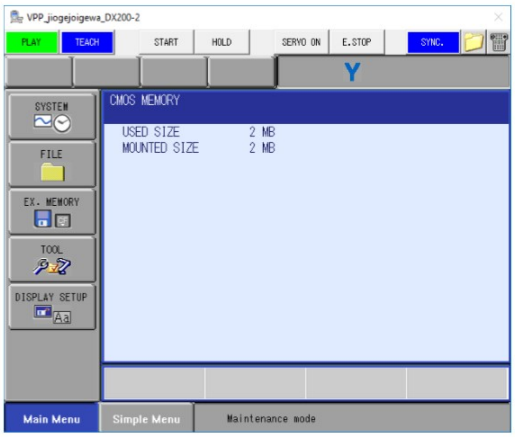
Step 15: This is an information page of the Option Board, continue to the next step.



YASKAWA

Step 16: This is an information page of the I/O module, continue to the next step





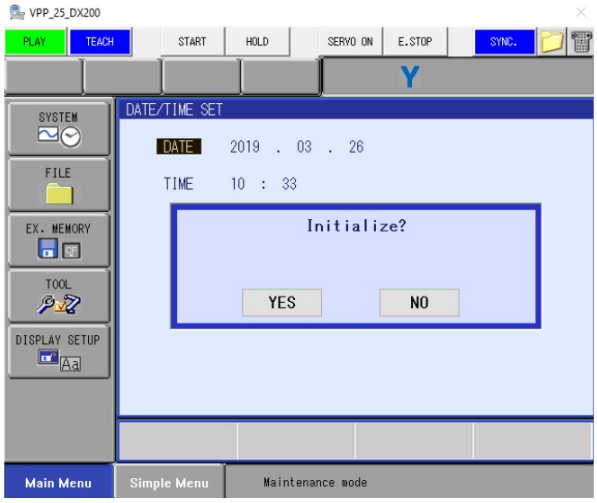
Step 17: This is an information page of the internal memory of the controller, continue to the next step

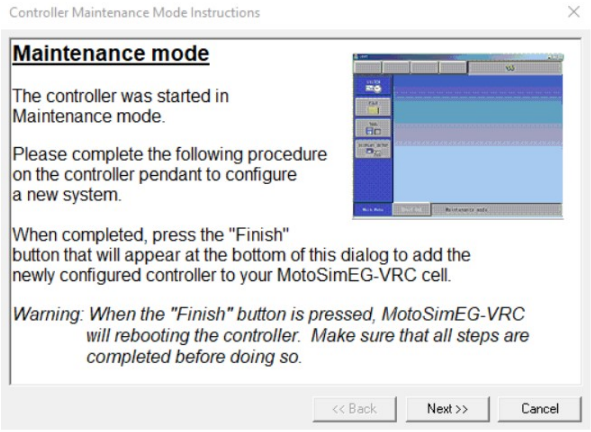
YASKAWA

In the following steps we finish the installation and select the robotinf-file

Step 18: Once the data has been implemented the window Date/Time Set is opened. Accept and then initialize the Controller.

Wait a few seconds until the pendant says "Please select a Main Menu" in the bottom line



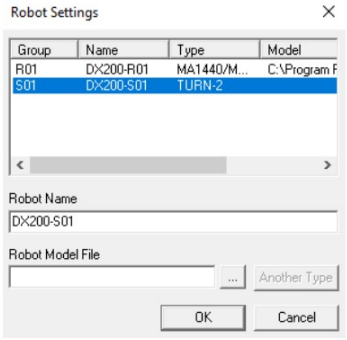


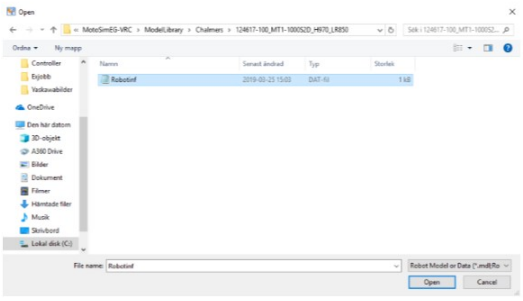
Step 19: In the window **Controller Maintenance Mode Instructions**, click **Next** two times and then **Finish**.

Once the finish button is pressed the controller will reboot.

YASKAWA

Step 20: In the next pop-up window, the positioners are added. Choose the desired positioner and press the ... button.

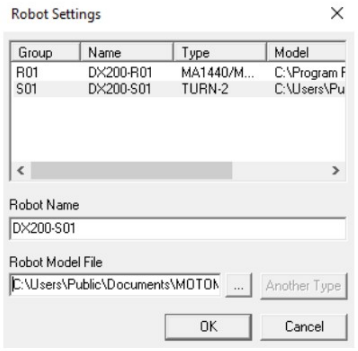




Step 21: Go to the wanted positioner folder containing the robotinf-file, select the file and press **Open**

Step 22: Once the desired positioners have a model file selected, press **OK**

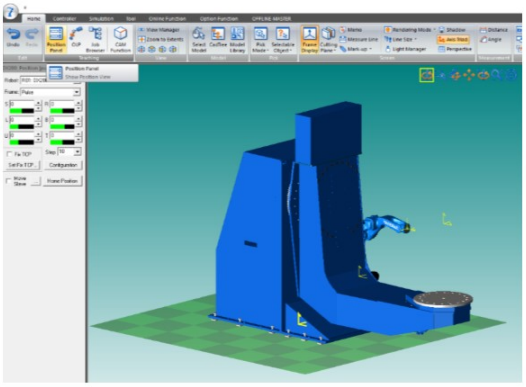
In the textbox Robot Name, you can change the name of the unit and its parts later seen in CadTree



YASKAWA

In the final steps, the kinematic of the positioner is controlled.

Step 23: Now the positioner and the robot is visible in MotoSim, the pendant is also shown, which can be closed for now. To rotate the positioner click **Position Panel** under the Home tab.



Step 24: Now choose the positioner in the **Robot** drop-down menu, then choose **Joint** in the **Frame** drop-down menu. Check the rotation of the axes by clicking on the up and down arrow after the axis number