UNIVERSITÉ TOULOUSE III - PAUL SABATIER
IUP of Intelligent Systems

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics and Interdisciplinar Studies

LABORATOIRE D'ANALYSE ET D'ARCHITECTURE DES SYSTÈMES
Robotics - Action - Perception Team

International Master of Interactive Systems

DIPLOMA THESIS

Liberec 2014

Bc. Ivo Knejp

UNIVERSITÉ TOULOUSE III - PAUL SABATIER
IUP of Intelligent Systems

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics and Interdisciplinary Studies

Laboratoire d'Analyse et d'Architecture des Systèmes, RAP team

Study Programme: N2612 – Electrical Engineering and Informatics
Study Field: 2612T071 – Engineering of Interactive Systems

Implementation of a control system on a 6-axes robot

Implémentation d'un système de commande sur un robot 6 axes

Implementace řídícího systému 6-osého robota

Author: **Ivo Knejp**
Advisor: Patrick Danès
Co-advisor: Viviane Cadenat
Tutor: Alexandre Nketsa

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL, UPS a LAAS-CNRS mají právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, UPS a LAAS-CNRS, kteří mají právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 6/01/2014

Podpis:

Declaration

I am aware of the fact that my thesis is fully covered by the Act No. 121/2000 on copyright, particularly § 60 (school work).

I note that the TUL, UPS and LAAS-CNRS have the right to enter into a license agreement on the use of my thesis and I declare that I agree with the possible use of my thesis (sale, rent, etc.).

I am aware that the use of my work or license itself, I can only with the consent of TUL, UPS and LAAS-CNRS, who have the right to demand from me a reasonable contribution to the costs incurred by the University for the creation of the work (until their actual amount).

The thesis I developed alone, using the mentioned literature, and in consultation with the head of the thesis and consultant.

Date: 6/01/2014

Signature:

Acknowledgements

I would like to thank my parents and all the others who supported and motivated me during working on the thesis and during my studies of the international studding program that took place mostly in Toulouse, France.

Also I would like to thank to RAP team, which gave me possibility to work on really interesting project with FESTO robot and to all people that helped and advised me during my internship in LAAS laboratory.

UNIVERSITÉ TOULOUSE III - PAUL SABATIER

IUP of Intelligent Systems

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics and Interdisciplinary Studies

Laboratoire d'Analyse et d'Architecture des Systèmes, RAP team

---

Annotation

This work is about the implementation of a control system on a 6-joint industrial robot produced by the FESTO company. The control system is developed with the National Instruments software LabView. First part of this work deals with the communication between the control unit which runs a real-time operating system and the robot via the CAN bus. The CANopen protocol is used to implement the communication. Second part of the work deals with forward differential kinematic and inverse kinematic models of the robot. This work also discusses the programming of the control interface, which allows the user to control the robot under various operating modes. It also implements the designed inverse kinematic model and other developed algorithms for reference movements of the robot.

Key words

Industrial robot, Cartesian robot, control system, direct differential kinematic model, inverse kinematic model, modified Denavit-Hartenberg convention, CANOpen, FESTO, LabView, CompactRio

Annotation

Ce stage a eu pour but d'implémenter un système de contrôle sur un robot industriel 6 axes fabriqué par FESTO. Ce système de contrôle est développé avec le logiciel LabView de National Instruments. La première partie de ce travail a été de mettre en place la communication entre l'unité de contrôle qui exécute un système temps réel et le robot via le bus CAN avec le protocole CANopen. La seconde partie de ce travail a été de déterminer le modèle géométrique inverse et le modèle différentiel direct du robot et de programmer l'interface de contrôle du robot. Cette interface permet à l'utilisateur de contrôler le robot dans différents modes. Elle implémente aussi les modèles géométrique et différentiel ainsi que d'autres algorithmes pour définir les mouvements de référence du robot.

Mots clés :

Robot industriel, robot Cartésien, système de contrôle, modèle géométrique inverse, modèle différentiel direct, convention des paramètres de Denavit-Hartenberg modifiés, CANOpen, FESTO, LabView, CompactRio

Anotace

Tato práce se zabývá návrhem a vývojem řídícího systému pro 6-kloubého průmyslového robota vyrobeného firmou FESTO. Řídící systém je vyvíjen v programu LABview od společnosti National Instruments. První část práce se zabývá komunikací mezi řídícím počítačem CompactRio s operačním systémem reálného času a robotem po sběrnici CAN za pomoci protokolu CANopen. Druhá část práce se věnuje problematice kinematických modelů v robotice. Jsou zde popsány postupy pro výpočet přímého diferenciálního a inverzního kinematického modelu. V této práci se také hovoří o programování rozhraní, které umožňuje ovládat robota v různých operačních módech. Vytvořený program také implementuje navržený inverzní kinematický model a další algoritmy pro plánování pohybu robota.

Klíčová slova

Průmyslový robot, Kartézský robot, řídící systém, přímý diferenciální kinematický model, inverzní kinematický model, modifikovaná Denavit-Hartenbergova konvence, vizuální zpětná vazba, CANOpen, FESTO, LabView, CompactRio

Contents

List of Figures

List of Tables

List of acronyms

CAN - Controller Area Network

DDKM - Direct Differential Kinematic Model

DH - Denavit-Hartenberg

DKM - Direct Kinematic Model

EMCY - Emergency Message

IKM - Inverse Kinematic Model

LabView - Laboratory Virtual Instrument Engineering Workbench

NI - National Instruments

NMT - Network Management

OSI - Open Systems Interconnections

PDO - Process Data Object

SDO - Service Data Object

SYNC - Synchronisation Message

RAP - Robotique, Action et Perception

RPDO - Receive Process Data Object

TPDO - Transmit Process Data Object

VI - Virtual Instrument

1 Introduction

Industrial robots are one of the most essential parts of automation in industry. They offer a great precision along with very high speed and repeatability. Thanks to their positive impact on quality of final product, volume of production and cost of the whole production process, they spread all over the world in all kinds of factories and industry facilities. The sales of industry robots keep increasing every year and so does their part in industry.

This report deals with the development of a control system for an industry robot. The aim is to substitute the existing system which has been found to be incomplete and unreliable. Designing a real-time control system from scratch for a 6-axis robot is a very complex task with many different problems to solve. First of all a combination of suitable hardware and software must be chosen. Both of those must cope with real-time constraints in order to get maximum security and performance from the robot. For this purpose a commercial solution from National Instruments, was chosen. Between the PXI platform and upgrading of the old industry PC, the CompactRio controller and corresponding module for communication has been the best choice. To develop software a National Instruments tool called LabView is used. Once the suitable hardware and software have been set, the communication with the robot must be done. All the fundaments of this communication are described in next chapter. When the communication is established and fundamental communication functions have been implemented, the robot kinematics and control takes place. To control the robot and define its tasks in Cartesian space the inverse kinematic model of robot has to be designed and implemented. To do so we have chosen the modified Denavit-Hartenberg convention and an analytical computation method. The whole procedure of designing robot kinematics is described further in this work. Once the kinematic model has been implemented, the basic reference trajectory generation can be done. Because the control system for a robot is a huge system with high complexity, a solid design must be prepared before putting everything together. All the links between various parts must be precisely defined and all the functions must be accurately described. Doing so, all this the risk of facing unexpected errors and troubles is reduced.

1.1 Robot Control

Robot control is a very large field, which covers a lot of different subjects. The main underlying topics are: control theory, robot kinematics, trajectory generation, vision based control, localization and mapping, robot locomotion and robot navigation, the last three topics being mainly related to mobile robotics. In this work, we cover only a subset of control theory, robot kinematics and motion planning.

1.2 The FESTO robot

FESTO is a well known German global manufacturer and supplier of pneumatic and electromechanical systems, components for process control and factory automation solution. Through the new technologies FESTO increase the performance of their robots. The robot from FESTO bought by LAAS is a custom Cartesian robot, which has six electrically powered joints.

1.2.1 Robot Structure

A Cartesian robot is a special type of robot whose three main axes of motion are linear and are at right angles to each other. Three prismatic joints simplify the robot control (kinematic model) and trajectory generation. In Figure 1 we can see the first three prismatic joints each on perpendicular axis. These three joints are designated by black rectangles with numbers 1, 2 and 3. The robot has a wrist made with the three revolute joints 4, 5 and 6.

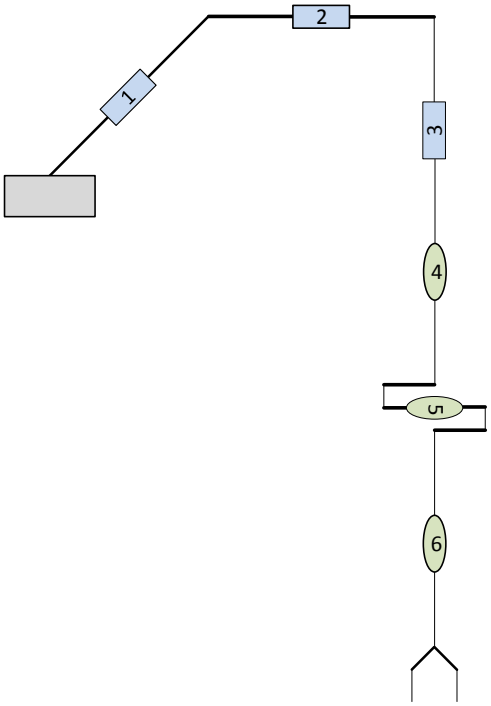

Figure 1 Robot schematics

Figure 2 shows a photo of the FESTO Cartesian robot to be controlled. It has a solid gantry to which is fixed the first prismatic joint. On this first joint, the second prismatic joint is mounted. Then, the third joint is mounted on an orthogonal axis. On the third vertical joint, the first revolute joint is mounted. This joint turns around the axis of the joint onto which it is mounted. Two other revolute joints are mounted. The last joint has a little platform enabling to mount an end effector. The gantry of the robot is enclosed in a Plexiglas box, which is necessary for safety of persons around. The doors of this box have to be closed every time, before the robot is used. Whenever they are opened, the voltage to motors is switched off. The robot also has another set of two emergency buttons. Pressing any of these buttons has the same effect as the opening the doors.

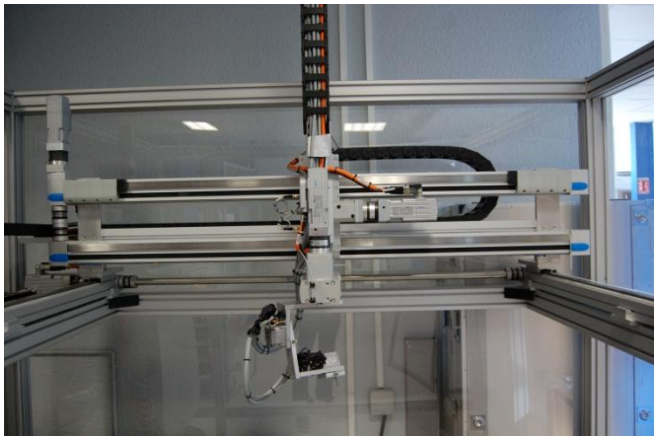

Figure 2 Cartesian robot developed by FESTO

1.2.2 Joints and Actuators

Each joint of the robot is related with one motor through which it is put into motion. Each joint has different type of motor and gear. All the joints are powered electrically and have different working strokes. All the characteristics of the motors can be accessed by the special software developed by FESTO. To do so it is necessary to connect the computer with the motor controller by a serial cable. The software is called Festo Configuration Tool. Table 1 displays a brief description of all the joints.

Label	Joint	Type of motor	Gear	Working stroke	Description
X	Prismatic	EMMS-AS-70-S-RM	5:1	1180mm	Provides linear translation along its two toothed belts.
Y	Prismatic	EMMS-AS-55-S-TM	5:1	1180mm	Provides linear translation along its toothed belt.
Z	Prismatic	EMMS-AS-70-S-RMB	3:1	800mm	Provides linear translation along its Cantilever axis.
ROTZ	Revolute	EMMS-55-S	3:1	270°	Provides rotation around axis Z.
ROTY	Revolute	EMMS-AS-40M	-	180°	Provides rotation around axis Y.
ROTX	Revolute	EMMS-AS-40M	-	360°	Provides rotation around axis Z.

Table 1 Robot joint's properties

After the manual examination of the robot body and wires, an admissible stroke of the revolute joints ROTY and ROTZ is deduced which avoids the risk of damaging the hardware. It is done as a software protection. It does not allow set the drive target position beyond the safe boundaries.

1.2.3 The motor controllers

Each motor of the robot is connected to one FESTO motor controller. Used type of the motor controllers is CMMP. The CMMP is a code of motor controller type. It provides various communication interfaces: PROFIBUS, CAN bus, DeviceNet, etc. It also has a seven-segment display to indicate the error states of motor controller. Each motor controller keeps the info about the joint it controls. It keeps the coefficients of the feedback control law, conversion factors or values of encoders. It can deliver information (position, velocity, etc.) on the joints from the associated sensors.

Figure 3 The robot motor controllers

1.3 The National Instruments Software and Hardware

National Instruments (NI) is an American producer of automated test equipment and virtual instrumentation software. The goal is to provide a software and hardware platform that accelerate the design and implementation of measuring and control systems. This is why we have chosen such a commercial solution for our control system. A software developed in the LabView framework can run in real time on a dedicated computing unit called CompactRio. This can support a module for

CANopen communication. NI also offers a special CANopen library, where all elementary APIs of CANopen protocol are implemented.

1.3.1 LabView

LabView is a dynamical system design and development environment. It relies on a visual programming language called "G", developed by NI. This platform is used by engineers and scientists in all kind of manufacturing processes, from design to production in multiple industries, advanced research, and academia. LabView offers wide a spectrum of functionalities, which are used to build all kind of dynamical systems. Graphical programming enables engineers to quickly learn how to master this tool in order to easily build a system while using advanced measuring technologies and control hardware, to analyze critical data, and to share their results.

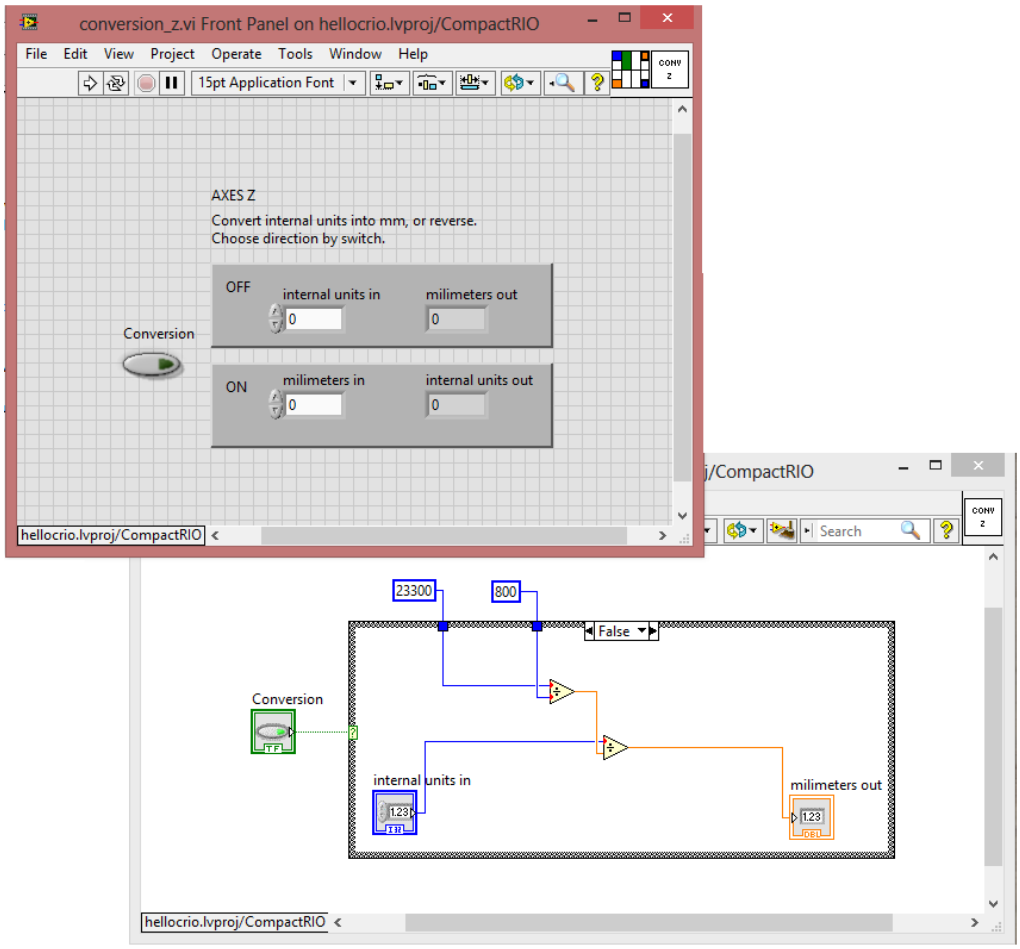

Figure 4 Example of a simple LabView program

A program made in LabView is called a virtual instrument (VI). Every virtual instrument has three parts. The first part is called front panel. This panel is used as the interface for anyone who is using the program. It can contain all types of input (controls and data references) and output (indicator) components. The second part is called back panel. This panel is a block diagram. It defines all the logic which connects the components from the front panel. The last part of every program is called connector panel. It serves to define input arguments to the program as well as output variables, which the program returns. This enables to easily embed any VI as a subroutine of other VIs in order to create a larger programs. Figure 4 shows a window with a front panel on the top and a block diagram of back panel on the bottom.

1.3.2 The CompactRio Computing Unit

CompactRio combines a real-time controller, reconfigurable I/O modules, an FPGA module and an Ethernet extension chassis. It is an easy-to-extend platform, in that new interfaces can easily be added. For our purpose, we use the following configuration of CompactRio in order to control the robot via CANopen protocol:

- cRIO-9024, Real-Time PowerPC Controller for cRIO;
- cRIO-9113, 4-slot Virtex-5 LX 50 Reconfigurable chassis for cRIO;
- NI 9881, C series CANopen interface.

Figure 5 A CompactRio platform with several modules

The specific "LabView RT" real time operating system is installed on CompactRio. This unit is then connected to a desktop PC by an Ethernet cable, so it can be controlled through the NI LabView software. Programmed applications can be deployed and executed, and are interfaced with the user through the desktop PC. Some cooperating applications can be run simultaneously on the user's PC. These can cooperate and share data in order to reduce the load on the side of CompactRio. However only the applications that do not require real-time performance can be executed on the side of the desktop PC. The communication through the Ethernet cable uses a specific real-time protocol developed by NI, where all the real-time data go through the queue with their time order preserved. This allows to have a LabView real-time control interface.

2 Communication

To ensure the communication between the control unit and the robot, the CANopen protocol is used. In this chapter all the primitives of the CANopen protocol are described. CANopen is based on the Controller Area Network (CAN) bus. The CAN bus was designed by German firm Robert Bosch GmbH in late eighties. The main attributes of this bus are a relatively high speed of transmission, high reliability, high endurance to extreme conditions and low price of wires. These attributes caused its high popularity in control systems.

The CAN protocol has been an internationally standardized in ISO 11898-1. It comprises the data link layer of the 7-layer ISO/OSI reference model [3]. Two communication services are provided by CAN bus: data frame transmission (sending of a message) and remote transmission request (requesting of a message). The CAN controller chip automatically performs all the others services like error signaling and automatic re-transmission of erroneous frames.

2.1 The CANopen Protocol

The CANopen protocol is internationally standardized (EN 50325-4) protocol for embedded control system. It uses the higher layer of the CAN bus. Specification of this protocol ensures that CANopen devices can communicate correctly among themselves. Thanks to this, it is used in many different application fields. It grants a direct access to the internal data of devices and it allows to transfer time critical process data.

Each motor of the robot has its own motor controller (see 1.2.3). Each controller is declared as a node on the CAN bus. The CompactRio is declared as a master device. It controls the motor controllers, which are declared as slaves.

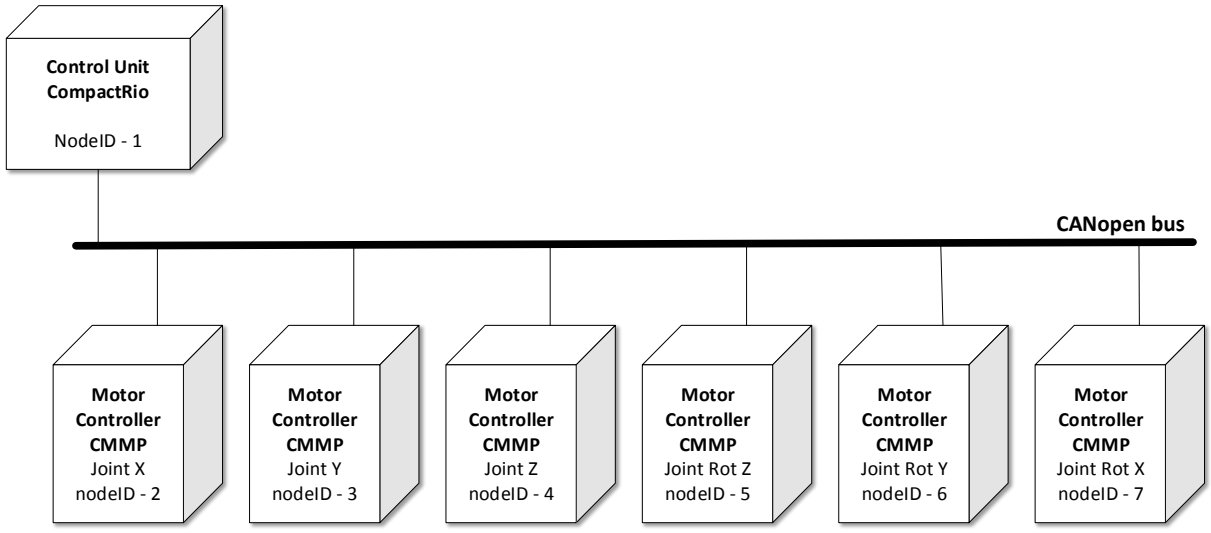

Figure 6 Devices connected to the CANopen bus

Messages are used to transfer data from a control unit, in this case the CompactRio, to any motor controller. Figure 7 displays two methods to access motor controller data via the CANopen bus. The first method is with access acknowledgement. It uses Service Data Objects (SDOs): both reading and writing by the control unit into controller are followed by an acknowledgement. The second method is without access acknowledgement. It uses Process Data Objects (PDOs). Two different types of PDOs are available: Transmit Process Data Object (TPDO) to read from a controller and Receive

Process Data Object (RPDO) to write in a controller. Both access methods are described further in this chapter.

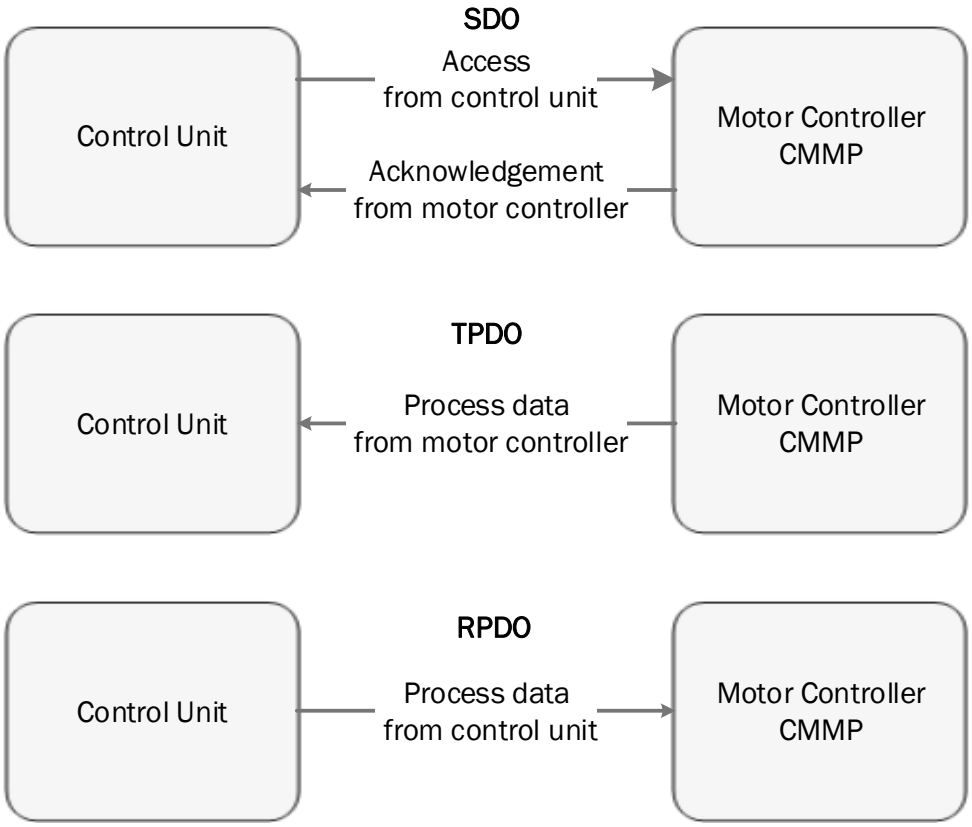

Figure 7 Object access with and without acknowledgement via CANopen [4]

Every type of message has its own unique identifier. The lower the identifier the higher the message priority. Table 2 shows the general form of a CANopen message.

Specific Identifier	Number of data bytes (0-8)	Data bytes (0 - 7 bytes)

Table 2 General form of a CANopen message

SDO, PDO and other types of messages such as synchronization, emergency protocol and network management, which are defined for special application cases, are described further in this chapter.

2.1.1 Messages, Node Identifiers

Data in a CANopen network are transferred by messages. CANopen messages are based on CAN bus communication objects. A communication object carries a CANopen message and a Communication Object ID (COB-ID)(Table 3). It is composed from a Node Identifier (Node-ID) and a 4-bit function code.

Every device interconnected through the CANopen protocol must have a Node-ID. The Node-ID can be any number between 1 and 127, provided that every device has an unique Node-ID. Each message contains the Node-ID of its recipient. A message sent to Node-ID 0 is broadcasted to all present devices but its producer.

Bit number	10	9	8	7	6	5	4	3	2	1	0

Meaning	Function Code	Node-ID

Table 3 Communication Object ID

2.1.2 The CANopen Network Limits

Every CANopen network has limitations based on the baud rate used by network's devices. The used baud rate influences the maximum length of the bus and the maximum stub length. The bus length is the distance between its first and last nodes. The cable stub length is the distance between the node and its connection to the bus. The sum of all the stubs in the network is called accumulated stub length. Every CANopen bus must support at least one of the following baud rates.

Baud Rate	Bus length	Max. stub length	Max. accumulated stub length
1 Mbit/s	25 m	1,5 m	7,5 m
800 kbit/s	50 m	2,5 m	12,5 m
500 kbit/s	100 m	5,5 m	27,5 m
250 kbit/s	250 m	11 m	55 m
125 kbit/s	500 m	22 m	110 m
50 kbit/s	1000 m	55 m	275 m
20 kbit/s	2500 m	137,5 m	687,5 m
10 kbit/s	5000 m	275 m	1375 m

Table 4 The CANopen network limitations by chosen baud rate.

The baud rate value for communication with the robot is 500kbit/s. This value was preset and stored by FESTO in the motor controller.

2.1.3 Object Dictionary

Variables in the CANopen devices are called objects. These objects are stored in a structure of variables called object dictionary. The object dictionary of any FESTO motor controller contains these elements:

- **Index** - It is a 2-byte address of the object in the object dictionary.
- **Sub-index** - It is a 1-byte address of the object in a sub-array with certain index.
- **Object name** - It is a symbolic type of the object. It can be an array, simple variable or a structure.
- **Name** - It is a string describing the object.
- **Type** - It is the data type of the object.
- **Attribute** - It is an information on the access rights for the object (read-only, write-only, read/write).

2.1.4 Service Data Object

The Service Data Object (SDO) protocol allows the control unit to access any object from the object dictionary of the CANopen device. Using the SDO does not require any further setting operation. This makes the SDO good for parameterization of the motor controller. The drawback of the SDO is its lower speed due to access acknowledgement. This makes the SDO unsuitable for real-time control.

Using the SDO allows the control unit to read or write objects which are in the dictionary of a motor controller. The SDO allows to transfer data with any length. Messages with specific structure are used. Every message has one or more segments. The first segment of the message contains bits that are necessary for communication and for handling of errors of SDO frame. Next three bytes contain

index and sub-index of the object. Last four bytes of the first segments carry data. The structure of the first segment is shown in Table 5.

Byte 0	Bytes 1 - 2	Byte 3	Bytes 4-7
600_h or 580_h + node-ID	Object Index	Object Sub-Index	up to 4 bytes of data

Table 5 First segment of the SDO message

If the transferred data are longer than four bytes, then they are divided into multiple segments. The first byte of each next segment contains again bits that are necessary for communication and for handling errors in a SDO frame. After the first byte, up to seven bytes of data follow (Table 6).

Byte 0	Bytes 1-7
600_h or 580_h + node-ID	up to 7 bytes of data (segment transfer)

Table 6 Next segments of the SDO message

SDO always starts from the control unit. The consumer of a message signals its reception by sending an acknowledgement, in the case of a writing command, or read-out value, in the case of a reading command (Figure 7). Specific identifier for SDO message consists of the base 600_h + node-ID of the motor controller to which the message is sent. Specific identifier of response message is 580_h + node-ID.

All the data types supported by the motor controller CMMP are described in Table 7.

Data type	Meaning	Min. value	Max. value
UINT8	8 bit value without algebraic sign	0	255
INT8	8 bit value with algebraic sign	-128	127
UINT16	16 bit value without algebraic sign	0	65535
INT16	16 bit value with algebraic sign	-32768	32767
UINT32	32 bit value without algebraic sign	0	$(2^{32} - 1)$
INT32	32 bit value with algebraic sign	$-(2^{32})$	$(2^{31} - 1)$

Table 7 Supported data types for SDO by motor controller CMMP [4]

2.1.5 Process Data Object

A Process Data Object (PDO) is basically a short message with high priority with a pre-defined data structure. The PDO is the best way to transfer data in real time: status and control data, sensor values and other I/O devices information. Unlike SDO, there is no acknowledgement that the message was received by any specific node.

There are two types of PDOs, the Transmit PDO (TPDO) and the Receive PDO (RPDO). The TPDO is used to transfer information from the motor controller to the control unit. In contrast the RPDO is used to transfer from the control unit to the motor controller. Every device which sends or receives any PDO must have properly set information how the data in PDO are structured and when they should be sent or received. The PDO can be sent or received periodically, or only when a synchronization message arrives. Table 8 shows an example of PDO message, with three mapped objects. A total length of the three objects is 8 bytes.

The configuration process of PDO parameters is called PDO linking. This linking must be done while the controller is in pre-operational state (more about states of device can be found in chapter 2.1.7). All changes of parameters must be done in a specific way.

181_h	8	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Specific Identifier	Number of data bytes	First mapped object				Second mapped object			Third mapped object

Table 8 Structure of PDO message

The motor controller CMMP allows only PDO, which carries from one to four objects with a maximum total length of 64 bits. The controller can link four RPDO and four TPDO.

Every PDO must be properly set before it is used. Once a PDO message is set, the communication is very fast, because there is no access acknowledgement. This makes PDO messages suitable for real-time control. In this case, TPDOs are set so that the motor controller receives and processes data (controlword, position to go, etc.) sent by the control unit. RPDOs are set to send important data (statusword, current position, current velocity, etc.) to the control unit.

2.1.6 SYNC Message

SYNC messages is used when several devices have to be synchronized with each other. A SYNC message is cyclically sent by the control unit. All nodes which are connected to the bus receive these SYNC messages and use them to start application-specific behavior. The SYNC messages are usually used to signal other devices, that they should receive or send out certain PDO.

Specific identifier of SYNC message is always 80_h. As can be seen in Table 9, this message doesn't carry any data.

80_h	0	-	-	-	-	-	-	-	-

Table 9 SYNC message

2.1.7 Network Management

Every CANopen device implements the Network Management (NMT) state machine. The state machine has different states and transitions between them. To modify the device state, the control unit must send a NMT message (Table 11). Only the control unit can modify the states of other devices.

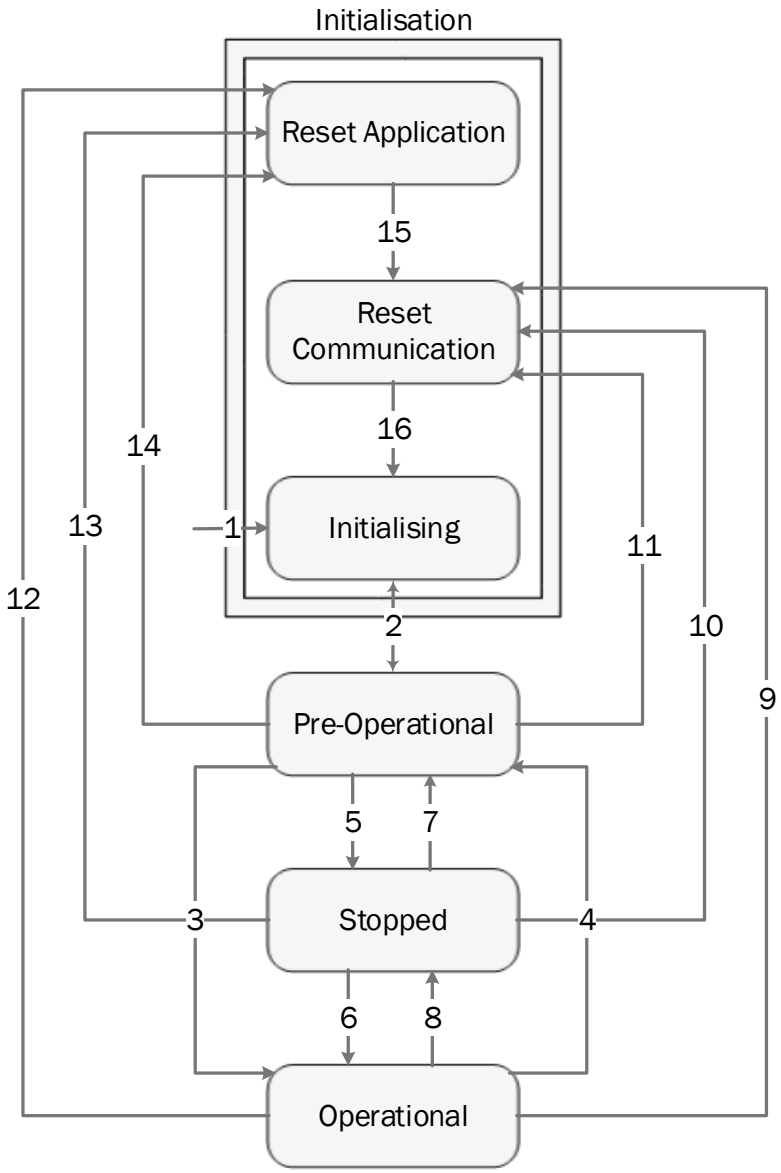

Figure 8 Network management state machine of a CANopen device [4]

The NMT state machine of a CANopen device can be seen in Figure 8. States of the device are represented by rounded rectangles. The transitions between the states are shown as numbered arrows. Each transition is coded in a message in Command Specifier (CS) byte. CS bytes are described in Table 10. The three states Reset Application, Reset Communication and Initializing, are in the Initialization block. The transition between those states (numbers 15, 16 and 2) are done automatically.

	Meaning	CS byte	Target State	
2	Boot Up	--	Pre-Operational	7Fh
3	Start Remote Node	01h	Operational	05h
4	Entre Pre-Operational	80h	Pre-Operational	7Fh
5	Stop Remote Node	02h	Stopped	04h
6	Start Remote Node	01h	Operational	05h
7	Entre Pre-Operational	80h	Pre-Operational	7Fh
8	Stop Remote Node	02h	Stopped	04h
9	Reset Communication	82h	Reset Communication*	--
10	Reset Communication	82h	Reset Communication*	--
11	Reset Communication	82h	Reset Communication*	--
12	Reset Application	81h	Reset Application*	--
13	Reset Application	81h	Reset Application*	--
14	Reset Application	81h	Reset Application*	--

Table 10 Transitions in Network Management state machine (* Final target state is pre-operational) [4]

To modify state, a NMT message is sent. The specific identifier of a NMT message is 000_h. This means it has the highest priority among all messages. Each NMT message consists of two bytes. The first byte contains the command specifier of certain transition and the second byte contains the target device node-ID. If the second byte equals zero, then all nodes in the network are addressed. No acknowledgement is generated in response to the NMT message.

000_h	2	CS byte	node-ID	-	-	-	-	-	-

Table 11 NMT message

Some of the messages cannot be used if the device is in a specific state. The state must be changed if the application requires messages that are not currently available. In Table 12 the availability of the PDO, SDO and NMT messages is mentioned. Available communication is marked with '**X**'.

Name	Meaning	SDO	PDO	NMT
Reset Application	No communication. All CAN object are reset to their rest values (application parameters set)	-	-	-
Reset Communication	No communication. The CAN controlled is newly initialized.	-	-	-
Initializing	Condition after hardware reset. Resetting of the CAN node, sending of the boot-up message.	-	-	-
Pre-Operational	Communication via SDOs possible. All PDOs are not active (no sending / evaluation).	**X**	-	**X**
Operational	Communication via SDOs possible. All PDOs are active (sending / evaluation).	**X**	**X**	**X**
Stopped	No communication except for NMT and heartbeat messages.	-	-	**X**

Table 12 Accessibility of messages in different states

2.1.8 Emergency Message

When an internal error of the CANopen device occurs, a emergency message is sent. It contains eight data bytes. The first two bytes contain an error code. The third byte contains an additional error code. The last five bytes contain zeros. Specific identifier for an emergency message is 80_h + node-ID (see Table 13). [4]

COB-ID	Data Length	Byte 0-1	Byte 2	Byte 3-7
80_h + Node ID	8	Emergency Error Code	Error Register	0

Table 13 Emergency message

2.1.9 Heartbeat Message

Heartbeat is an error control message, which is used to monitor the communication between a slave and master devices. The slave device periodically sends a heartbeat message to the master device. If the master device does not receive such message on a regular basis, then it can take appropriate measures.

The identifier of a heartbeat message is 700_h + nodeID. A heartbeat message contains one reference data byte (see Table 14). It is a byte which indicates in which state the device is found. The byte values for possible states can be seen in the last column of Table 10.

700_h + nodeID	1	State	-	-	-	-	-	-	-

Table 14 Heartbeat message

2.1.10 Bootup Message

A bootup message is sent during the booting of any CANopen device. It has the same structure as a heartbeat message, which is described further. A reference data byte is set to zero (see Table 15).

700_h + nodeID	1	0	-	-	-	-	-	-	-

Table 15 Bootup message

2.1.11 Nodeguarding Message

Nodeguarding is also an error control message, which is used to monitor the communication between a slave and a master. In this case the master and slave nodes monitor each other. The master cyclically asks (sends a requests) the slave about its NMT status. If such requests are not received by the slave within a certain period, then the slave triggers a related error message. On the other hand, the master checks if the slave responds within a certain period.

The master device has to send a request in the form of a remote frame. It is a simple message with a specific identifier and one bit set. The specific identifier of a the remote frame is 700_h + nodeID. This remote frame carries no data (Table 16).

700_h + nodeID	Remote bit	-	-	-	-	-	-	-

Table 16 Nodeguarding request - remote frame

The slave node (e.g. motor controller) responds with a message, which is almost identical to the heartbeat message. The message contains one byte of reference data. This byte is composed of one toggle bit and the NMT status of the motor controller. The toggle bit is the 8th bit of reference data byte. Its value is inverted with each NMT message so the control unit can verify there is no message lost (Table 17).

700_h + nodeID	1	Toggle bit / NMT status	-	-	-	-	-	-	-	-

Table 17 Nodeguarding message

The structure of the reference data byte is described in Table 18.

Bit	Value	Name	Meaning
7	80_h	toggle bit	Change with each message
0 - 6	$7F_h$	NMT status	04_h Stopped 05_h Operational $7F_h$ Pre-Operational

Table 18 Nodeguarding - structure of reference data byte

A nodeguarding message shares the specific identifier with a heartbeat message. Thanks to that, only one type of messages can be used to detect errors on the controllers at one time.

2.1.12 Specific Identifiers

In Table 19 are all the specific identifiers of possible messages. All the identifiers of PDOs can be changed. The table reposts their standard values, which are usually used.

Object type	Identifier	Object type	Identifier
SDO (Master to slave)	600_h + nodeID	RPDO3	401_h
SDO (Slave to master)	580_h + nodeID	RPDO4	501_h
TPDO1	181_h	SYNC	080_h
TPDO2	281_h	EMCY	080_h + nodeID
TPDO3	381_h	HEARTBEAT	700_h + nodeID
TPDO4	481_h	NODEGUARDING	700_h + nodeID
RPDO1	201_h	BOOTUP	700_h + nodeID
RPDO2	301_h	NMT	000_h

Table 19 Specific identifiers of all messages [4]

2.2 Device Control via CANopen

The CANopen communication allows the host (e.g. control unit) to dialog with the motor controller via two special objects - statusword and controlword. These two objects are essential for a real-time motor control. The statusword provides the basic information about the current operation and the motor controller state. To read out the statusword from the motor controller the TPDO is used. The controlword is used to send commands to the controller. It can be used to modify the motor controller state, or to control the drive within the current operating mode (2.3). The controlword is linked into the RPDO, so the motor controller can receive commands in real time. Both of these objects are described further in this chapter.

Figure 9 shows the state diagram of the FESTO motor controller. The diagram is divided into three general areas: Power disabled, Power enabled and Fault. The first area, Power disabled, includes all states, where the final stage is off. The second area, Power enabled, includes all states, where the final stage is on. The last area, Fault, includes all the states, where the handling of errors is needed. States of the motor controller are represented by rectangles with state names written inside and circles with numbers represent transitions.

When the final stage is on, so is the voltage to the motor. Only then it is possible to move the drive. The final stage should be on only for the time necessary to perform any task. When the task is done, the final stage should be switched off to prevent damage of the motor.

All the possible states are described in Table 20. A transition can be trigged through a command sent by the host or internally through the motor controller, when an error occurs. The host can send a command through a controlword with a specific bit combination (Table 21).

When the motor controller is switched on, it administers the self-test. When the self-test is successfully finished, the motor controller is in the *Switch On Disabled* state. At this point the CAN communication is available and the parameters of motor controller can be set (PDO linking, etc.). At this point the power supply is off and the shaft is thus free. When all the parameters are properly set, the power supply can be switched on. Switching on is done by the transitions 2, 3 and 4. After those transitions the motor controller reaches the *Operation Enabled* state. The voltage to the motor is on. Also the PDO, which are essential for motor command, are now available. The motor is now controlled according to the current operation mode. When the robot work is finished, the final stage should be switched off through the transitions 5, 6 and 7. The transition 9 has to be used with caution: if this transition is trigged while the motor is still running, then the motor fizzles out unregulated. This can lead to a damage of hardware.

Name	Meaning
NOT READY TO SWITCH ON	Self-test of motor controller is done. The CAN communication is not activated. The final stage is off.
SWITCH ON DISABLED	Self-test was completed. The CAN communication is activated. The final stage is off.
READY TO SWITCH ON	Digital inputs "final stage" and "controller enable" are monitored. The motor controller waits until they are at 24V. The final stage is off.
SWITCHED ON	The final stage is switched on.
OPERATION ENABLE	Voltage to the motor is on. The motor is controlled according to the currently chosen operation mode. The final stage is switched on.
QUICKSTOP ACTIVE	The quick stop function was initiated. The voltage to the motor is still on and the motor is controlled according to the setting of quick stop function.
FAULT REACTION ACTIVE	An error has occurred. If the error is critical, the systems immediately switches into fault state. Otherwise, voltage to the motor is still on and the motor is controlled according to the fault reaction function.
FAULT	An error has occurred. Voltage to the motor is off.

Table 20 States of the motor controller [4]

If any error occurs, the motor controller is instantly branched into *Fault* state. In this case certain actions, as emergency breaking, can be still performed. This depends on the severity of the error.

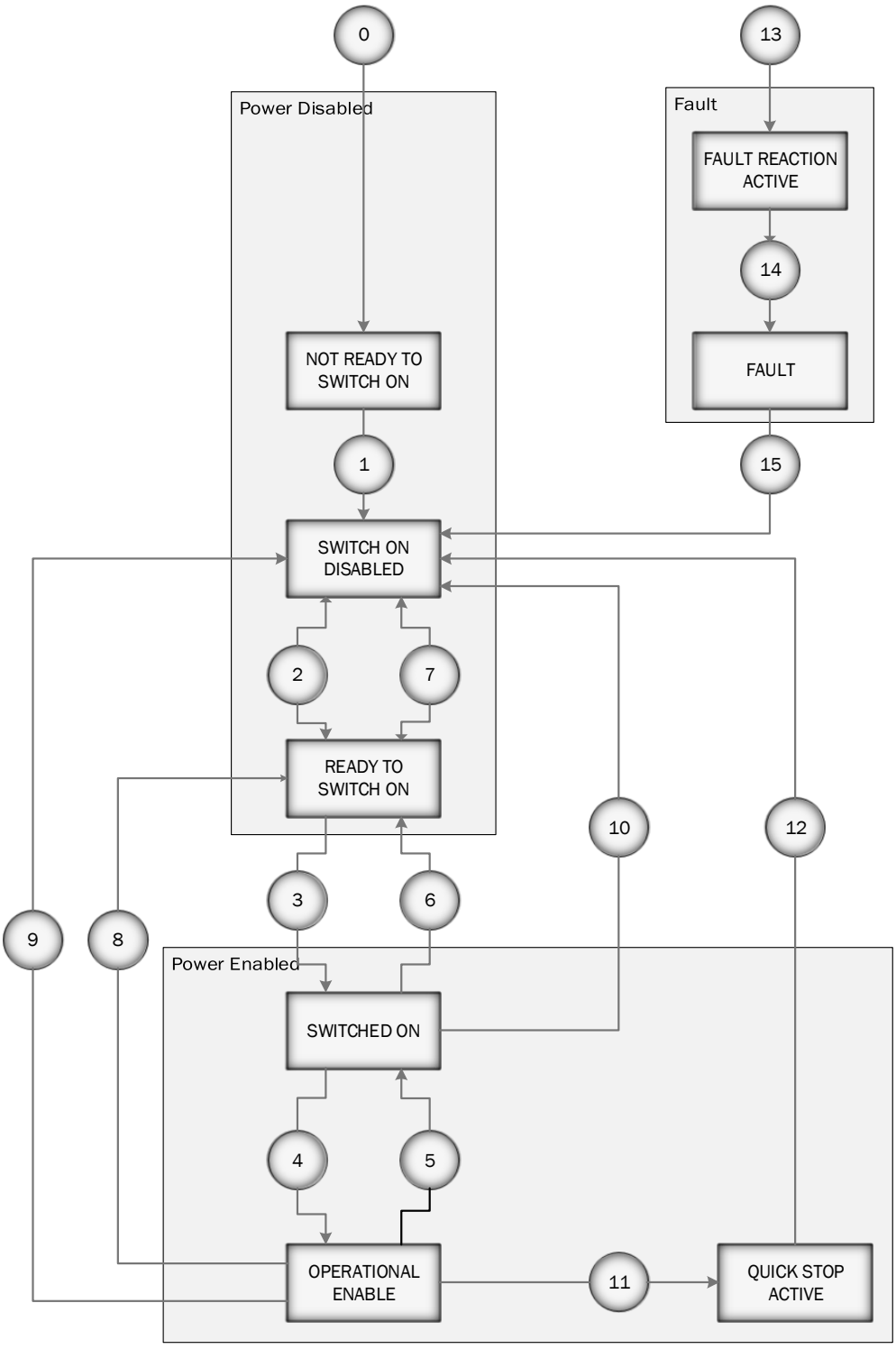

Figure 9 The state diagram of the FESTO motor controller [4]

All the non-internal transitions can be trigged by certain commands. The commands are run by setting the transition code accordingly into the controlword. The lowest four bits are jointly evaluated in order to trigger a state transition. All the different bit combinations can be seen in Table 21, in the third column ('X' means that the value of the bit is not important). Every command has its name enclosed in quotation marks. For example if the motor controller is in the state *Switched On*, then value $000F_h$ (command "Enable Operation") can be set in the controlword in order to trigger a

transition. After this command, the motor controller is set into the state *Operation Enable*. This example assumes that no other bits are set in the controlword.

	Transition performed if	**Bit Combination**	**Bit combination of commands**				**Action**
			3	2	1	0	
0	Switched on or reset occurred	Internal transition					Execute self-test
1	Self-test is successful	Internal transition					Activation of CAN communication
2	Final stage and regulator activated + command "Shutdown"	"Shutdown"	X	1	1	0	-
3	Command "Switch On"	"Switch on"	X	1	1	1	Switching on the final stage
4	Command "Enable Operation"	"Enable Operation"	1	1	1	1	Control according to the set operation mode
5	Command "Disable Operation"	"Disable Operation"	0	1	1	1	Final stage is blocked, motor rotates freely
6	Command "Shutdown"	"Shutdown"	X	1	1	0	Final stage is blocked, motor rotates freely
7	Command "Quickstop"	"Quickstop"	X	0	1	X	-
8	Command "Shutdown"	"Shutdown"	X	1	1	0	Final stage is blocked, motor rotates freely
9	Command "Disable Voltage"	"Disable Voltage"	X	X	0	X	Final stage is blocked, motor rotates freely
10	Command "Disable Voltage"	"Disable Voltage"	X	X	0	X	Final stage is blocked, motor rotates freely
11	Command "Quickstop"	"Quickstop"	X	0	1	X	Motor breaks according to the quick stop function
12	Breaking has ended or command "Disabled Voltage"	"Disable Voltage"	X	X	0	X	Final stage is blocked, motor rotates freely
13	Error occurred	Internal transition					If error is critical, transition 14 occurs, else the fault reaction function is activated
14	Error resolution is ended.	internal transition					Final stage is blocked, motor rotates freely
15	Error resolved + command "Fault Reset"	"Fault Reset"	Raising edge on bit 7				Acknowledge error (with rising edge)

Table 21 Transitions of the FESTO motor controller [4]

2.2.1 Statusword

A statusword is a 16-bit unsigned integer object that displays several information about the motor controller. The statusword has the read-only access rights. The meaning of all statusword bits is described in Table 22.

Bit No.	Name	Description
0	State of the motor controller	Coding of state of the motor controller in bits 0, 1, 2, 3, 5 and 6 is described in Table 23.
1		
2		
3		
4	Voltage enabled	This bit is set when the final stage transistors are switched on.
5	State of the motor controller	Coding of state of the motor controller in bits 0, 1, 2, 3, 5 and 6 is described in Table 23.
6		
7	Warning	This bit shows that a direction of rotation is blocked because one of the limit switches has been trigged.
8	Drive is moving	This bit is set when the actual velocity is outside the related tolerance window.
9	Remote	This bit is set when the final stage of motor controller can be switched on via CANopen.
10	Target reached	Depends on the operational mode: ❖ Profile Position Mode - The bit is set when the target position is reached. That means that the current position is in the parameterized position window. ❖ Profile Velocity Mode - The bit is set when the current velocity of the drive is within the parameterized velocity tolerance window.
11	Internal limit active	This bit is set when the internal limitation is active.
12	Operation mode specific bit	Depends on the operation mode: ❖ Profile Position Mode - The bit is set when new target position was recognized. It is deleted by setting *New set point* bit in controlword. ❖ Profile Velocity Mode - The bit is set when the current velocity of the drive is within the parameterized velocity tolerance window. ❖ Homing Mode - The bit is set when the home position is reached with no error.
13	Error	Depends on the operation mode: ❖ Profile Position Mode - The bit is set when the current position is lying outside the parameterized position window while the drive run is finished. ❖ Homing Mode - This bit is set when the reference travel is interrupted.
14	Manufacturer status bit	The meaning of this bit is configurable.
15	Drive referenced	The meaning of this bit is configurable.

Table 22 Statusword [4]

Through the statusword it is possible to read out the current state of the motor controller. The states are coded in bits 0, 1, 2, 3, 5 and 6. In Table 23 the possible states of the state diagram are listed along with the corresponding bit combinations.

The seventh and the eighth bit of the statusword are so called manufacturer-specific. They must be configured in a specific way. For our application these bits are not important. The configuration details of those two bits can be found in the manufacturer documentation [4].

State	Bit 6	Bit 5	Bit 3	Bit 2	Bit 1	Bit 0
Not Ready To Switch On	0	X	0	0	0	0
Switched On Disabled	1	X	0	0	0	0
Ready To Switched On	0	1	0	0	0	1
Switched On	0	1	0	0	1	1
Operation Enable	0	1	0	1	1	1
Quick Stop Active	0	0	0	1	1	1
Fault Reaction Active	0	X	1	1	1	1
Fault	0	X	1	1	1	1
FAULT (in accordance with DS402)	0	X	1	0	0	0

Table 23 States of the FESTO motor controller representation in the statusword

2.2.2 Controlword

A controlword is a 16-bit unsigned integer. It is used to trigger the transition or to perform some other actions. The actions depend on the current operation mode.

Bit No.	Name	Description
0	Command bits	Control of the state transitions (see Table 21, column bit combination of commands).
1		
2		
3		
4	Start motion	Depends on the operation mode: ❖ Profile Position Mode - The drive starts the motion on raising edge of this bit. ❖ Homing Mode - The drive starts the parameterized reference travel on raising edge and interrupts this travel on falling edge.
5	Change set immediately	When this bit is set, any new positioning task will be executed immediately by interrupting any previous ongoing task. When this bit is not set the new task will not be executed before the old one is finished (This bit is evaluated only if the profile position mode is set).
6	Relative	When this bit is set, the motor controller refers target position of current position task to the nominal position of the position controller (This bit is evaluated only if the profile position mode is set).
7	Reset Fault	On the raising edge of this bit, the controller tries to acknowledge the existing errors. It is successful only if the causes of the errors were resolved.
8	Halt/Stop	Depends on the operation mode: ❖ Profile Position Mode - When this bit is set the ongoing positioning task is interrupted and deceleration of the drive is initiated. ❖ Profile Velocity Mode - When this bit is set the velocity is reduced to zero. Deleting the bit afterwards cause the drive the accelerate again. ❖ Homing Mode - When this bit is set the parameterized reference run is interrupted.
9	Reserved	This bit is always zero.
10		
11		
12		
13		

14 15		

Table 24 Controlword description [4]

As Table 24 shows, bits 0, 1, 2 and 3 are used to cause the command to trigger some state transition (Table 21). Also the bit 7 is used to trigger the transition "Fault Reset". The bit 4 is basically used to start the reference travel related to the some operation modes. The bit 8 is used to stop the travel. The second byte of this object is reserved and it always equals to zero.

The controlword is used to switch on the motor controller final stage, start the reference travel, interrupt the travel if it is necessary and switch off the final stage when the task is done.

2.3 Operation Modes

Several operation modes are available. For each of them, the bits of controlword and statusword can be interpreted differently. Only the modes in bold are compatible with a control through the CANopen. All operation modes are listed below:

- **profile torque mode**
- **profile velocity mode**
- **homing mode**
- **profile position mode**
- **interpolated position mode**
- torque-controlled mode
- velocity-controlled mode
- reference run (homing)
- position mode
- synchronous position specification

The developed control system uses only three of these modes: homing mode, profile position mode and profile velocity mode. These three operating modes are described further in this chapter.

Each mode has its own set of parameters, which must be set. All the parameters related with a position, velocity and acceleration can be converted from internal units in the motor controller to a real units. This conversion is done by the conversion factors and it is also influenced by the current setting of encoders. Both the conversion factors and the values of encoders must be set correctly in order to get the right values. Currently the values of the conversion factors are set so that the real units for position, velocity and acceleration are millimeters, millimeters per seconds and millimeters per square seconds in the case of prismatic joint, and degrees, degrees per second and degrees per square second in the case of revolute joint.

2.3.1 Homing Mode

The homing mode is the basic mode used to drive the robot to an initial position. The few following parameters must be set before the homing mode is initiated: homing speed, homing acceleration and homing offset. When everything is set, the homing function can be activated by the corresponding command through controlword. Figure 10 shows on the left side all the input objects to homing mode, on the right side all the output objects which are influenced by homing profile. In the statusword, the 13th bit is set once the drive has reached its home position with no error.

Figure 10 Simple homing profile representation

In Figure 11, is shown the displacement of the zero position from the home position. The home position is determined by the physical properties of each joint.

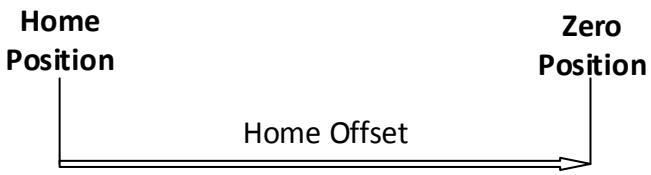

Figure 11 Homing mode - displacement of zero position [4]

There are several methods of homing. Homing methods use four different signals to drive the robot: a positive and negative limit switch, homing switch and zero pulse of angle encoder. Every method has a specific direction of reference travel and a specific type of evaluation of zero pulse.

In our case for the three prismatic joints, the method with reference travel to the negative limit switch is selected. This means that each joint moves quickly toward its negative limit switch, then moves back slowly when this switch has been reached. While it is going back, it searches for the exact home position. For the three revolute joints the method with use of a homing switch is chosen. These joints can move quickly in positive or negative direction. So, when the homing switch is passed, the direction is inverted and speed is reduced, in order to search for the exact position of the homing switch.

2.3.2 Profile Position Mode

This mode is used to move a joint to any desired position. A velocity trapezoid profile is defined. The parameters of the profile position mode such as maximum velocity, acceleration, deceleration and quick stop deceleration can be set through the corresponding objects. The trajectory generator generates the position demand value from the target position. Then the position demand value goes to the position control function (i.e. to a low-level position servo) where all the limits of the profile are applied. Finally, the motor controller moves the drive accordingly to the control effort which is generated by the curve generator (Figure 12). Both functions must be properly parameterized before the profile position mode is used. Position control function is a closed loop control function.

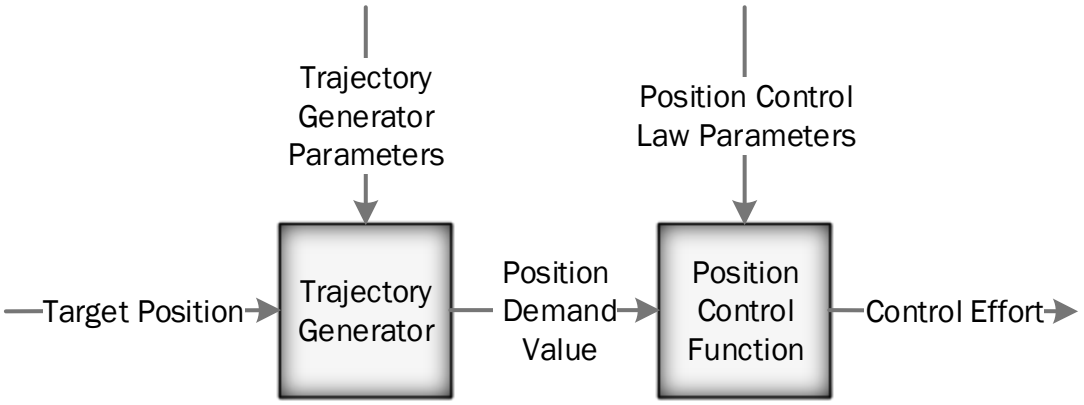

Figure 12 The curve generator

Figure 12 displays the schematics of the Position Control Function. It is a function that checks whether the joint has reached the desired position, or whether an error has occurred. The target position is reached whenever the current position of the drive remains within a specified range for a specific time. If so, the bit 10 in the statusword is set to 1. Difference between the position feedback value and the position demand value constitute the tracking error. If the value of the tracking error exceeds a specified value for a specific time, then the motion of the drive is considered as incorrect and the drive is stopped. The bit 13 in the statusword is set to 1 in order to indicate the error state of the drive.

The position profile is represented in Figure 13. The joint starts at time T1 with a given acceleration. It accelerates until it reaches its maximum velocity V1. The drive then moves at constant velocity until it starts to decelerate. So as to stop at the target position at time T2. All the profile parameters must be set before the run is initiated. The second profile which starts at time T3 and ends at the time T4, displays other reference travel with a different maximum velocity V2. Naturally the acceleration and deceleration profiles are faster than the profiles of the first profile.

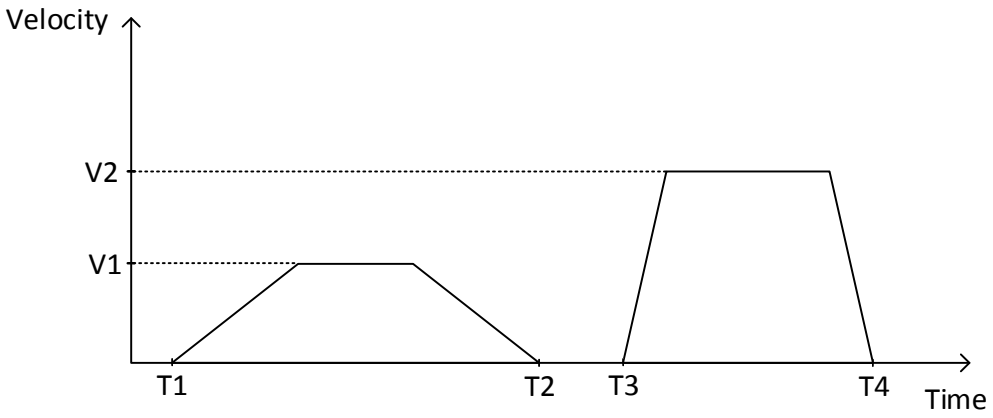

Figure 13 The position profile

The parameters of the profile position mode can be modified during the run of the drive. To do so, the bit 5 of the controlword must be set to 1. With the new profile parameters, the new trajectory is generated and the drive immediately initiates the new positioning task. Figure 14 shows the case when at time T2 new parameters were processed and new a trajectory was generated. The profile parameters can be modified after they are stored in the motor controller buffer. They are stored automatically,

shortly after the beginning of the reference travel. Once they are stored this is indicated in the statusword.

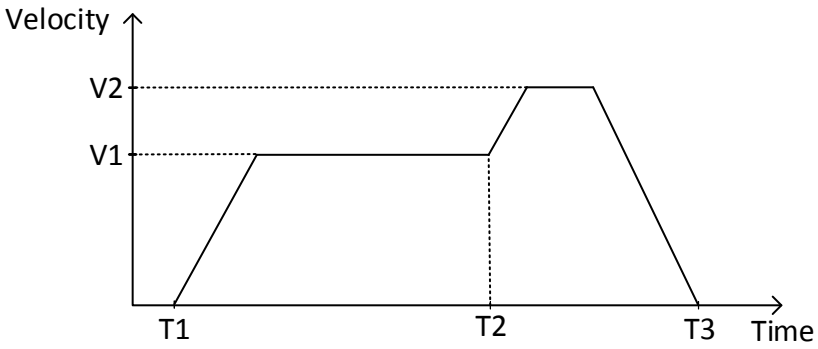

Figure 14 Positioning profile with change of parameters

More information about all of the objects which can be parameterized, in order to set the position profile mode, can be found in the manufacturer documentation [4].

2.3.3 Profile Velocity Mode

The profile velocity mode is based on a low-level velocity servo of the motor. In this mode the target velocity that the drive should reach is set. A limiting function is applied in order to get a smooth velocity demand value, which is safe for the motor. This velocity goes to the velocity controller along with the current velocity of the drive and velocity control parameter set. The velocity controller then generates the control effort used to move the drive.

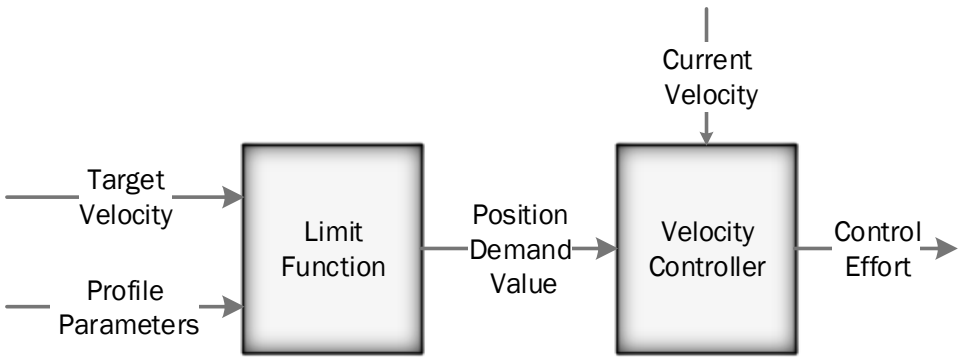

Figure 15 Simplified structure of the profile velocity mode

Figure 16 displays the position and velocity on an experiment run. Both the position and velocity are in an internal units. The bottom chart shows, how the drive accelerates until it reaches the target velocity. It remains at the constant speed until it has to decelerate so as to stop smoothly. The top chart shows how the position is changing over time.

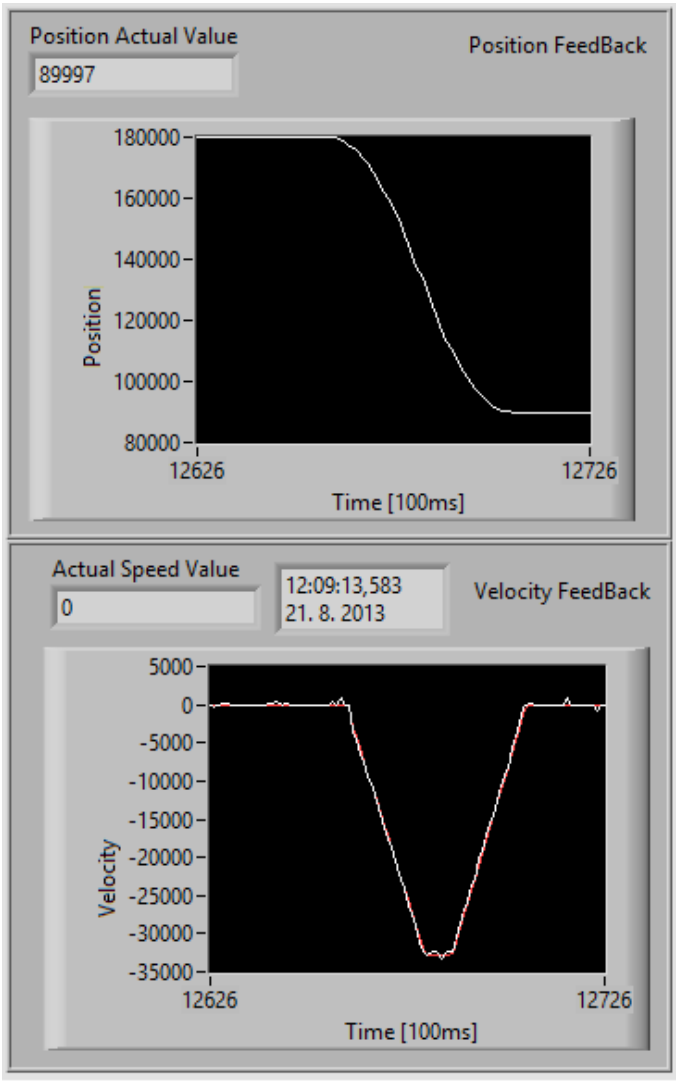

Figure 16 The position and velocity feedback when using the profile velocity mode

As can be seen on the bottom chart, there is a noise interfering in the velocity feedback. The unfiltered value of the velocity is drawn with white color. The red line represents the filtered current velocity. However the filtered value is not used for the control, but only for the turn-through protection of the motor. The current velocity value is computed as a differential of the current position value. It means that the noise comes from the position sensor.

With the profile velocity mode, two other sub-functions are activated. These two functions check if the drive is either going by its target velocity or if it is standing still. Both functions work with parameterized windows and time periods. If the current velocity stays within a threshold window for a specific time, then the drive is considered as standing still and the bit 12 of the statusword is set to 0. If the current velocity stays within a velocity window for a specific time, then the target velocity of the drive is considered as reached.

In order to reach a target position with profile velocity mode, it is necessary to compute the exact time when the drive must start deceleration (4.1.7).

3 Robot Kinematics

Every robot can be represented as a kinematic chain. The chain is composed from the links and the joints. The links represent the rigid elements of the robot. The joints either revolute or prismatic are connected to the actuators and endow robot with mobility. The revolute joint permits a relative rotation around a single axis and the prismatic joint allows a linear motion along a single axis. The current configuration of each joint can be expressed with a single real number: an angle of rotation or distance of translation.

There are two main approaches to characterize robot kinematics: forward kinematics and inverse kinematics. Forward kinematics maps of the joint configuration variables to the end-effector position and orientation in Cartesian space. The inverse kinematics does the exact opposite. With the inverse kinematics it is possible to derive the set of configurations of all the joints leading to given end-effector position and orientation in Cartesian space. This is used for example to program pick and place operation, where we know the coordinates of the object and coordinates of the place where it should be put.

3.1 Modified Denavit-Hartenberg Convention

In order to compute both kinematic models, it is necessary to attach a frame to each link and then compute transformation matrices between the frames. To make the choice of the frames more systematic, the modified Denavit-Hartenberg convention is used. Each frame associated with one i-th link can be designated as $R_i = (O_i, x_i, y_i, z_i)$, where the O_i stands for its origin. The rules below describe how to assign frames with $i = 2, \dots, n$ to each link.

- O_{i-1} is the base of the common perpendicular of the axis of the joints L_{i-1} and L_i and it is situated on axis L_{i-1}. If those axis are parallel, the perpendicular has to be chosen arbitrarily.
- x_{i-1} is the unit vector of the common perpendicular of the axis of the joints L_{i-1} and L_i in the direction from L_{i-1} toward L_i. If these axis are convergent the orientation of this unit vector is chosen arbitrarily.
- z_{i-1} is the unit vector of the axis L_{i-1} with arbitrary orientation.
- y_{i-1} is chosen so the frame represented by axis x_{i-1}, y_{i-1} and z_{i-1} is orthogonal.

The axes of R_0 can be chosen arbitrarily. It must be rigidly linked to the robot. Essentially it is chosen in a way to make further computations as easy as possible.

By applying these rules on the FESTO robot, the frames can be easily assigned to the robot links. The result is displayed on Figure 17. The frames are drawn with red lines. As can be seen, the origins of the frames o_3, o_4, o_5 and o_6 are the same. [1]

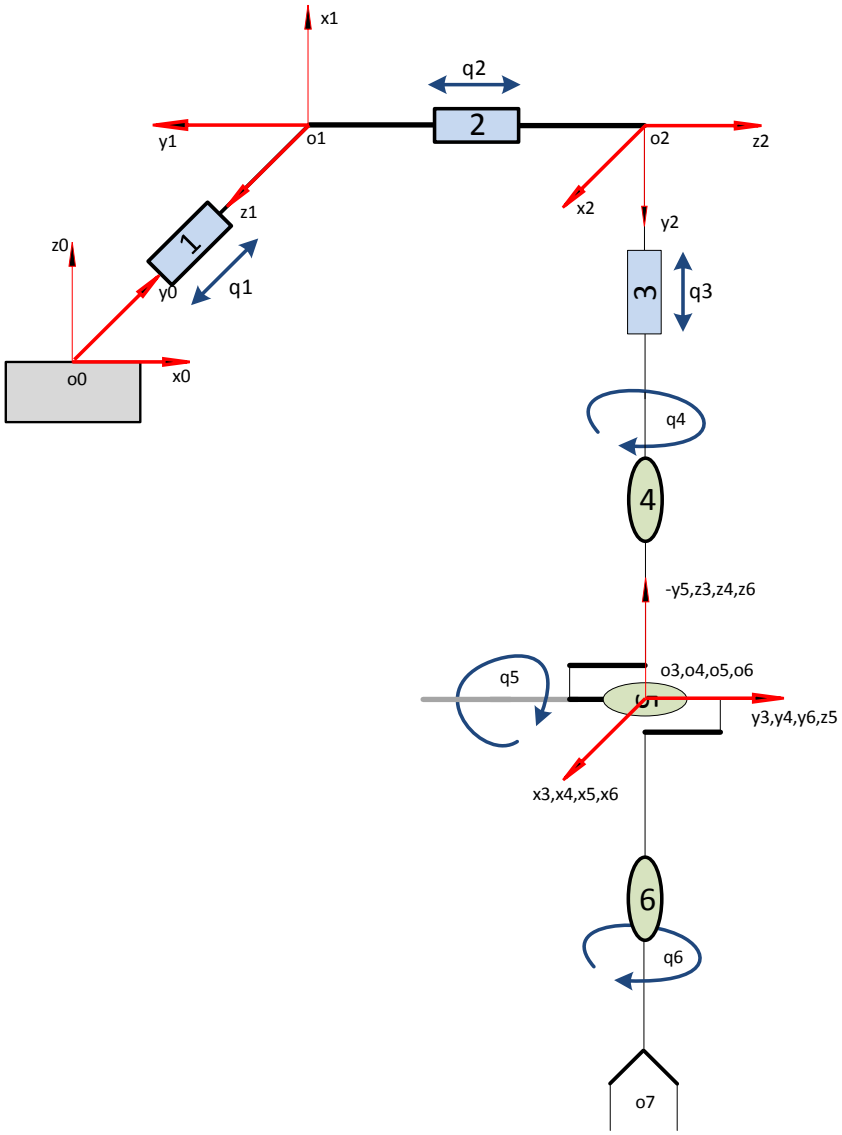

Figure 17 Robot schematics with assigned frames

The parameters α_{i-1}, a_{i-1}, θ_i and r_i, with $i = 1, 2, \dots, n$, where n is the number of joints of robot are called modified Denavit-Hartenberg parameters. These parameters can be derived by following rules [1]:

- σ_i = 1 if the joint type is prismatic; 0 if the joint type is revolute.
- α_{i-1} = algebraic angle between axis z_{i-1} and z_i, measured around axis x_{i-1}.
- a_{i-1} = arithmetic distance between the joint axis O_{i-1} and O, measured along axis x_{i-1}.
- θ_{i-1} = algebraic angle between axes x_{i-1} and x_i, measured around axis z_i.
- r_i = arithmetic distance from the point O_{i-1} to the point O_i, measured along axis z_i.

By applying these listed rules to Figure 17, the modified Denavit-Hartenberg parameters can be derived. They are displayed in Table 25.

	1	2	3	4	5	6
σ_i	1	1	1	0	0	0
a_{i-1}	0	0	0	0	0	0
α_{i-1}	π/2	π/2	π/2	0	- π/2	π/2
r_i	q_1	q_2	q_3	0	0	0
θ_i	π/2	π/2	0	q_4	q_5	q_6

Table 25 Modified Denavit-Hartenberg's parameters of the robot

3.1.1 Set of Transformation Matrices

The general form of an homogenous transformation matrix between two frames based on the modified Denavit-Hartenberg parameters is as follows:

$$T_{i-1,i} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \cos\alpha_{i-1}\cdot\sin\theta_i & \cos\alpha_{i-1}\cdot\cos\theta_i & -\sin\alpha_{i-1} & -r_i\cdot\sin\alpha_{i-1} \\ \sin\alpha_{i-1}\cdot\sin\theta_i & \sin\alpha_{i-1}\cdot\cos\theta_i & \cos\alpha_{i-1} & r_i\cdot\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

This matrix can be also represented in this simple form:

$$T_{i-1,i} = \begin{pmatrix} R_{i-1,i} & p_{i-1,i} \\ 0 & 1 \end{pmatrix} \quad (2)$$

In (2) $p_{i-1,1}$ and $R_{i-1,i}$ represent terms the translation vector and the rotation matrix between frames R_{i-1} and R_i.

The values from Table 25 can be substituted into the general form of transformation matrices (1). The result is the set of transformation matrices between the frames. These matrices are used for computing both direct and inverse kinematic models. They are as follows:

$$T_{0,1} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -q_1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_{1,2} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -q_2 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{2,3} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -q_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_{3,4} = \begin{pmatrix} \cos q_4 & -\sin q_4 & 0 & 0 \\ \sin q_4 & \cos q_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

$$T_{4,5} = \begin{pmatrix} \cos q_5 & -\sin q_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin q_5 & -\cos q_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_{5,6} = \begin{pmatrix} \cos q_6 & -\sin q_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin q_6 & \cos q_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3.2 Inverse Kinematic Model

The inverse kinematic model is used to derive the configuration parameters for given end-effector position and orientation. To solve this problem there are two possible solutions: analytical and

numerical. The analytical solution is chosen, because it is suitable for Cartesian robots as well as for real-time control.

3.2.1 Computation of IKM

First step is to identify the transformation from the 6th frame to the 0th frame in Cartesian and joints space. When we do so, we finally get the system to solve (4).

$$T_{3,2}(q_3) \cdot T_{2,1}(q_2) \cdot T_{1,0}(q_1) \cdot T_{0,6}(x) = T_{3,2}(q_3) \cdot T_{2,1}(q_2) \cdot T_{1,0}(q_1) \cdot T_{0,6}(q) \tag{4}$$

The complexity of the system is reduced by expressing the right site of (4) as a function of q_4 , q_5 and q_6.

$$T_{3,2}(q_3) \cdot T_{2,1}(q_2) \cdot T_{1,0}(q_1) \cdot T_{0,6}(x) = T_{3,4}(q_4) \cdot T_{4,5}(q_5) \cdot T_{5,6}(q_6) \tag{5}$$

The matrix $T_{0,6}(x)$ represents the end-effector position and orientation in Cartesian coordinates. The matrix $T_{0,6}(x)$ is known.

$$T_{0,6}(x) = \begin{pmatrix} t_{1,1} & x & t_{1,3} & t_{1,4} \\ t_{2,1} & x & t_{2,3} & t_{2,4} \\ t_{3,1} & x & t_{3,3} & t_{3,4} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

To solve the system it is better to first compute the right-hand side. The matrix products are computed in order to get the transformation matrix from the 6th frame to the 4th frame (9).

$$\begin{aligned} T_{3,5} = T_{3,4} \cdot T_{4,5} &= \begin{pmatrix} \cos q_4 & -\sin q_4 & 0 & 0 \\ \sin q_4 & \cos q_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos q_5 & -\sin q_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin q_5 & -\cos q_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos q_4 \cdot \cos q_5 & -\cos q_4 \cdot \sin q_5 & -\sin q_4 & 0 \\ \sin q_4 \cdot \cos q_5 & -\sin q_4 \cdot \sin q_5 & \cos q_4 & 0 \\ -\sin q_5 & -\cos q_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{7}$$

$$\begin{aligned} T_{3,6} = T_{3,5} \cdot T_{5,6} &= T_{3,5} \cdot \begin{pmatrix} \cos q_6 & -\sin q_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin q_6 & \cos q_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos q_4 \cdot \cos q_5 \cdot \cos q_6 - \sin q_4 \cdot \sin q_6 & x & \cos q_4 \cdot \sin q_5 & 0 \\ \sin q_4 \cdot \cos q_5 \cdot \cos q_6 + \cos q_4 \cdot \sin q_6 & x & \sin q_4 \cdot \sin q_5 & 0 \\ -\sin q_5 \cdot \cos q_6 & x & \cos q_5 & 0 \\ 0 & x & 0 & 1 \end{pmatrix} \end{aligned} \tag{8}$$

$$T_{4,6} = \begin{pmatrix} \cos q_5 \cdot \cos q_6 & x & \sin q_5 & 0 \\ \sin q_6 & x & 0 & 0 \\ -\sin q_5 \cdot \cos q_6 & x & \cos q_5 & 0 \\ 0 & x & 0 & 1 \end{pmatrix} \tag{9}$$

Once the right-hand-side has been processed, the computation of the left-hand side follows. The direction of the transformation can be inverted as done in (10).

$$
\begin{aligned}
T_{i-1,i} &= \begin{pmatrix} R_{i-1,i} & P_{i-1,i} \\ 0_{1x3} & 1 \end{pmatrix} \rightarrow T_{i,i-1} = \begin{pmatrix} R_{i-1,i}^T & -R_{i-1,i}^T \cdot P_{i-1,i} \\ 0_{1x3} & 1 \end{pmatrix} \\
T_{0,1} &= \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -q_1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow T_{1,0} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -q_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
T_{1,2} &= \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -q_2 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow T_{2,1} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -q_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
T_{2,3} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -q_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow T_{3,2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & -q_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
\tag{10}
$$

The goal is to find the relation between the joint space parameters and end-effector position and orientation in Cartesian space. Because the second column of a rotation matrix is redundant once its first and third columns are given it does not need to be developed.

In order to get the final expression of $T_{3,6}(q_1, q_2, q_3)$ following products are computed:

$$
T_{1,6} = T_{1,0}(q_1) \cdot T_{0,6}(x) = \begin{pmatrix} t_{3,1} & x & t_{3,3} & t_{3,4} \\ -t_{1,1} & x & -t_{1,3} & -t_{1,4} \\ -t_{21} & x & -t_{2,3} & (-t_{2,4} - q_1) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}
$$

$$
T_{2,6} = T_{2,1} \cdot T_{1,6} = \begin{pmatrix} -t_{2,1} & x & -t_{2,3} & (-t_{2,4} - q_1) \\ -t_{3,1} & x & -t_{3,3} & -t_{3,4} \\ t_{1,1} & x & t_{1,3} & (t_{1,4} - q_2) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}
$$

$$
T_{3,6} = T_{3,2} \cdot T_{2,6} = \begin{pmatrix} -t_{2,1} & x & -t_{2,3} & (-t_{2,4} - q_1) \\ t_{1,1} & x & t_{1,3} & (t_{1,4} - q_2) \\ t_{3,1} & x & t_{3,3} & (t_{3,4} - q_3) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{13}
$$

Finally the final expression is derived:

$$
\begin{aligned}
&T_{3,6}(q_4, q_5, q_6) \\
&= \begin{pmatrix} \cos q_4 \cdot \cos q_5 \cdot \cos q_6 - \sin q_4 \cdot \sin q_6 & x & \cos q_4 \cdot \sin q_5 & -t_{2,4} - q_1 \\ \sin q_4 \cdot \cos q_5 \cdot \cos q_6 + \cos q_4 \cdot \sin q_6 & x & \sin q_4 \cdot \sin q_5 & t_{1,4} - q_2 \\ -\sin q_5 \cdot \cos q_6 & x & \cos q_5 & t_{3,4} - q_3 \\ 0 & x & 0 & 1 \end{pmatrix}
\end{aligned}
\tag{14}
$$

From (14), the system of equations to be solved can be deduced. It is shown in Table 26.

$-\boldsymbol{t_{2,1}}$	=	$\cos q_4 \cdot \cos q_5 \cdot \cos q_6 - \sin q_4 \cdot \sin q_6$
$-\boldsymbol{t_{2,3}}$	=	$\cos q_4 \cdot \sin q_5$
$\boldsymbol{t_{1,1}}$	=	$\sin q_4 \cdot \cos q_5 \cdot \cos q_6 + \cos q_4 \cdot \sin q_6$
$\boldsymbol{t_{1,3}}$	=	$\sin q_4 \cdot \sin q_5$
$\boldsymbol{t_{3,1}}$	=	$-\sin q_5 \cdot \cos q_6$
$\boldsymbol{t_{3,3}}$	=	$\cos q_5$
$-\boldsymbol{t_{2,4}} - \boldsymbol{q_1}$	=	0
$\boldsymbol{t_{1,4}} - \boldsymbol{q_2}$	=	0
$\boldsymbol{t_{3,4}} - \boldsymbol{q_3}$	=	0

Table 26 Relation between the joint configurations and the end-effector parameters.

From the above system, it is possible to compute the relation between the joints configuration and end-effector position and orientation parameters. All lines from Table 26 must be used to get this relation. Because the end-effector can reach one position with different joint configuration more than one solution can be found.

Solution for the first three joints is simple. To express q_1, q_2 and q_3, the last three lines of the table are used.

$$\begin{aligned} \boldsymbol{q_1} &= -\boldsymbol{t_{2,4}} \\ \boldsymbol{q_2} &= \boldsymbol{t_{1,4}} \\ \boldsymbol{q_3} &= \boldsymbol{t_{3,4}} \end{aligned} \tag{15}$$

From the 6th line, equation (16) can be derived.

$$\begin{aligned} \cos q_5 &= t_{3,3} \\ \sin q_5 &= \pm\sqrt{1 - \cos^2 q_5} = \pm\sqrt{1 - t_{3,3}^2} \end{aligned} \tag{16}$$

From (16) the value of q_5 can be derived by using function $atan2(\sin(x), \cos(x))$. There are two solutions:

$$\boldsymbol{q_5 = atan2(\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}, t_{3,3})}, \text{ with } \boldsymbol{\varepsilon_5 = \pm 1} \tag{17}$$

The 2nd and the 4th lines are necessary to express q_4:

$$\begin{aligned} -t_{2,3} &= \cos q_4 \cdot \sin q_5 \\ t_{1,3} &= \sin q_4 \cdot \sin q_5 \end{aligned} \tag{18}$$

From (18) it is possible to derive values for $\sin q_4$ and $\cos q_4$:

$$\sin q_4 = \frac{t_{1,3}}{\sin q_5} = \frac{t_{1,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}} \tag{19}$$

$$\cos q_4 = \frac{-t_{2,3}}{\sin q_5} = \frac{-t_{2,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}}$$

It is again possible to use function $atan2(\sin(x), \cos(x))$ in order to express the solution of q_4. This solution is possible if and only if the $\sin q_5 \neq 0$.

$$\boldsymbol{q_4 = atan2\left(\frac{t_{1,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}}, \frac{-t_{2,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}}\right)} \quad (20)$$

To express q_6 the 1st and 3rd lines are used. The 1st line is multiplied by $(-\sin q_4)$ and the 3rd line by $\cos q_4$:

$$\begin{aligned} \sin q_4 \cdot t_{2,1} &= -\sin q_4 \cdot \cos q_4 \cdot \cos q_5 \cdot \cos q_6 + \sin^2 q_4 \cdot \sin q_6 \\ \cos q_4 \cdot t_{1,1} &= \cos q_4 \cdot \sin q_4 \cdot \cos q_5 \cdot \cos q_6 + \cos^2 q_4 \cdot \sin q_6 \end{aligned} \quad (21)$$

By using classical trigonometric properties it is possible to derive the value for $\sin q_6$:

$$\sin q_6 = \sin q_4 \cdot t_{2,1} + \cos q_4 \cdot t_{1,1} = \frac{t_{2,1} \cdot t_{1,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}} - \frac{t_{1,1} \cdot t_{2,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}} \quad (22)$$

Then from the 5th line it is possible to express $\cos q_6$. The result is in (24).

$$t_{3,1} = -\sin q_5 \cdot \cos q_6 \quad (23)$$

$$\cos q_6 = -\frac{t_{3,1}}{\sin q_5} = -\frac{t_{3,1}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}} \quad (24)$$

It is finally possible to derive value for q_6. By using the function $atan2(\sin(x), \cos(x))$.

$$\boldsymbol{q_6 = atan2\left(\frac{t_{2,1} \cdot t_{1,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}} - \frac{t_{1,1} \cdot t_{2,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}}, -\frac{t_{3,1}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}}\right)} \quad (25)$$

The solution (25) is possible only if $\sin q_5 \neq 0$. There is an another solution to be computed for $\sin q_5 \neq 0$.

If $\sin q_5 = 0$ then $\cos q_5 = \pm 1$. In such a case there is an infinite connected set of values for q_4 and q_6 (Figure 18).

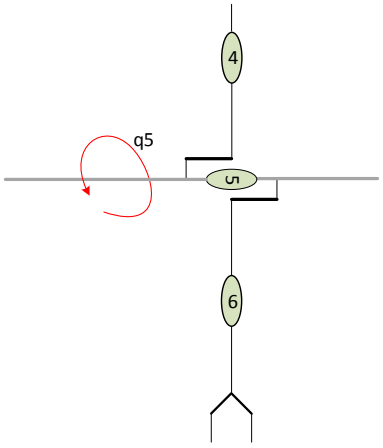

Figure 18 The configuration of the robot if $q_5 = k \cdot \pi$, with k = {0, 1}

In order to simplify the expression of q_4 and q_6, the substitution $\cos q_5 = \varepsilon_5'$ is used:

$$\begin{aligned} -t_{21} &= \varepsilon_5' \cdot \cos q_4 \cdot \cos q_6 - \sin q_4 \cdot \sin q_6 \\ t_{1,1} &= \varepsilon_5' \cdot \sin q_4 \cdot \cos q_6 + \cos q_4 \cdot \sin q_6 \end{aligned} \tag{26}$$

ε_5' has two possible values: $\varepsilon_5' = 1$ if $q_5 = 0°$ or $\varepsilon_5' = -1$ if $q_5 = \pi$.

The development of the 1st line:

$$\begin{aligned} -t_{21} \cdot \varepsilon_5' &= \cos q_4 \cdot \cos q_6 - \varepsilon_5' \cdot \sin q_4 \cdot \sin q_6 \\ &= \cos(\varepsilon_5' \cdot q_4) \cdot \cos q_6 - \sin(\varepsilon_5' \cdot q_4) \cdot \sin q_6 = \cos(\varepsilon_5' \cdot q_4 + q_6) \end{aligned} \tag{27}$$

The development of the 3rd line:

$$\begin{aligned} t_{1,1} &= \varepsilon_5' \cdot \sin q_4 \cdot \cos q_6 + \cos q_4 \cdot \sin q_6 \\ &= \sin(\varepsilon_5' \cdot q_4) \cdot \cos q_6 + \cos(\varepsilon_5' \cdot q_4) \cdot \sin q_6 = \sin(\varepsilon_5' \cdot q_4 + q_6) \end{aligned} \tag{28}$$

Finally from (27) and (28) can be derived that:

$$\boldsymbol{\varepsilon_5' \cdot q_4 + q_6 = atan2(t_{1,1}, -t_{21} \cdot \varepsilon_5')} \tag{29}$$

Based on the value of ε_5' this equation has two general solutions. The second solution with $\varepsilon_5' = -1$ is only theoretical, because the fifth robot joint cannot reach position $q_5 = \pi$ due to its physical constrains. There is only one admissible solution:

$$\boldsymbol{q_4 + q_6 = atan2(t_{1,1}, -t_{21})} \tag{30}$$

This solution corresponds to an infinite number of (q_4, q_6) pairs.

This close the computation of the IKM

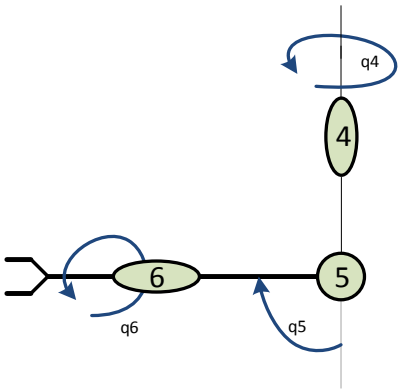

Figure 19 The configuration of the robot if $q_5 = \frac{\pi}{2}$

To sum up, if $\sin q_5 \neq 0$ then the IKM has two different solutions for $\varepsilon_5 = \pm 1$.

Joint	IKM solutions
1	$q_1 = -t_{2,4}$
2	$q_2 = t_{1,4}$
3	$q_3 = t_{3,4}$
4	$q_4 = atan2(\frac{t_{1,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}}, \frac{-t_{2,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}})$
5	$q_5 = atan2(\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}, t_{3,3})$
6	$q_6 = atan2(\frac{t_{2,1} \cdot t_{1,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}} - \frac{t_{1,1} \cdot t_{2,3}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}}, -\frac{t_{3,1}}{\varepsilon_5 \cdot \sqrt{1 - t_{3,3}^2}})$

Figure 20 The IKM solution for $\sin q_5 \neq 0$

Two solutions for one given end-effector configuration can be seen in Figure 21. The second solution is drawn in red. The first solution is: $q_4 = 0°$, $q_5 = \frac{\pi}{2}$ and $q_6 = \frac{\pi}{2}$. The second solution for this case is: $q_4' = \pi$, $q_5' = -\frac{\pi}{2}$ and $q_6' = -\frac{\pi}{2}$.

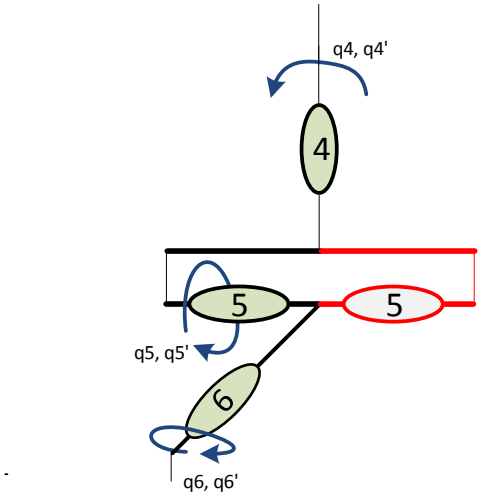

Figure 21 The configuration of last three joints with two possible solutions

If the first solution for $\varepsilon_5 = 1$. can be expressed as a function of six joint values: $S(q_1, q_2, q_3, q_4, q_5, q_6)$ and the second solution for $\varepsilon_5 = -1$ as a function: $S'(q_1', q_2', q_3', q_4', q_5', q_6')$. The relation between the solutions is as follows:

$$\boldsymbol{q_1 = q_1'} \qquad \boldsymbol{q_4 = q_4' - \pi}$$
$$\boldsymbol{q_2 = q_2'} \qquad \boldsymbol{q_5 = q_5' + \pi}$$
$$\boldsymbol{q_3 = q_3'} \qquad \boldsymbol{q_5 = q_6' + \pi}$$

3.3 Direct Differential Kinematic Model

The direct differential kinematic model is used to compute the change of position and orientation of the robot in time in respect to the change of configuration of the joint set $(q_1, \dots, q_6)$. It is differential version of a basic method of forward kinematics.

3.3.1 Jacobian Matrix

One of the most important quantities used for the analysis and control of robot motion is so called Jacobain Matrix. This matrix is commonly use for many tasks in robot engineering from motion planning, through determination of singular configurations to torque and forces transformation from end-effector to the manipulator joints. In our case the Jacobian matrix is used to derive the direct differential kinematic model of the robot. It is better to use the matrix $J_{(i)_j}$, instead of the general $J_{(n)_0}$. It means that the frame i is considered as a body and that the projection is made into the frame j instead of frame 0. This reduces the complexity of DDKM computation.

3.3.2 Computation steps of DDKM

1. Determination of the preferential Jacobian matrix
    a) Compute p, i, j and express $J_{i_{(j)}}$
    b) Determine $z_{k_{(j)}}$
    c) Determine $p_{ki_{(j)}}$
    d) Compute the cross products $z_{k_{(j)}} \times p_{ki_{(j)}}$
2. Determination of the skew anti-symmetric matrix $\hat{P}_{in_{(j)}}$ associated to vector $p_{in_{(j)}}$
3. Determination of $dp_{(0)}$ and $d\varphi_{(0)}$
    a) Multiply the matrices from right to left

b) Introduce a variable E_i whenever an element includes more than one arithmetical operation

4. Determination of *dx*

3.3.3 Computation of DDKM

The first step of computation of direct differential kinematic model is to compute the Jacobian matrix $J_{(i)_j}$. To choose suitable *i* and *j* the basic relation $i = int(n/2)$ and $j = i + 1$, where *n* is equal to the number of robot joints is used. In this case it is necessary to compute the Jacobian matrix $J_{(4)_3}$:

$$J_{4_{(3)}} = \begin{pmatrix} Z_{1_{(3)}} & Z_{2_{(3)}} & Z_{3_{(3)}} & Z_{4_{(3)}} \times p_{44_{(3)}} & Z_{5_{(3)}} \times p_{54_{(3)}} & Z_{6_{(3)}} \times p_{64_{(3)}} \\ 0 & 0 & 0 & Z_{4_{(3)}} & Z_{5_{(3)}} & Z_{6_{(3)}} \end{pmatrix} \tag{31}$$

The rotational matrices $R_{1,3}$, $R_{3,5}$ and $R_{3,6}$ are computed from transformation matrices (3).

$$R_{1,3} = R_{1,2} \cdot R_{2,3} = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{32}$$

$$\begin{aligned} R_{3,5} = R_{3,4} \cdot R_{4,5} &= \begin{pmatrix} \cos q_4 & -\sin q_4 & 0 \\ \sin q_4 & \cos q_4 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos q_5 & -\sin q_5 & 0 \\ 0 & 0 & 1 \\ -\sin q_5 & -\cos q_5 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \cos q_4 \cdot \cos q_5 & -\cos q_4 \cdot \sin q_5 & -\mathbf{\sin q_4} \\ \sin q_4 \cdot \cos q_5 & -\sin q_4 \cdot \sin q_5 & \mathbf{\cos q_4} \\ -\sin q_5 & -\cos q_5 & \mathbf{0} \end{pmatrix} \end{aligned} \tag{33}$$

$$\begin{aligned} R_{3,6} &= R_{3,5} \cdot R_{5,6} \\ &= \begin{pmatrix} \cos q_4 \cdot \cos q_5 & -\cos q_4 \cdot \sin q_5 & -\sin q_4 \\ \sin q_4 \cdot \cos q_5 & -\sin q_4 \cdot \sin q_5 & \cos q_4 \\ -\sin q_5 & -\cos q_5 & 0 \end{pmatrix} \cdot \begin{pmatrix} \cos q_6 & -\sin q_6 & 0 \\ 0 & 0 & -1 \\ \sin q_6 & \cos q_6 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \cos q_4 \cdot \cos q_5 \cdot \cos q_6 - \sin q_4 \cdot \sin q_6 & \cdots & \mathbf{\cos q_4 \cdot \sin q_5} \\ \sin q_4 \cdot \cos q_5 \cdot \cos q_6 + \cos q_4 \cdot \sin q_6 & \cdots & \mathbf{\sin q_4 \cdot \sin q_5} \\ -\sin q_5 \cdot \cos q_6 & \cdots & \mathbf{\cos q_5} \end{pmatrix} \end{aligned} \tag{34}$$

The values $Z_{1_{(3)}}$, $Z_{2_{(3)}}$ and $Z_{3_{(3)}}$ can be represented by unit matrix of size 3x3 and values of $Z_{4_{(3)}} \times p_{44_{(3)}}$, $Z_{5_{(3)}} \times p_{54_{(3)}}$ and $Z_{6_{(3)}} \times p_{64_{(3)}}$ equal to $(0,0,0)^T$. Values $Z_{4_{(3)}}$, $Z_{5_{(3)}}$ and $Z_{6_{(3)}}$ are vectors in bold of product matrices (32), (33) and (34). Their values can be seen in (35):

$$\begin{aligned} Z_{4_{(3)}} &= (1,0,0)^T \\ Z_{5_{(3)}} &= (-\sin q_4, \cos q_4, 0)^{\mathrm{T}} \\ Z_{6_{(3)}} &= (\cos q_4 \cdot \sin q_5, \sin q_4 \cdot \sin q_5, \cos q_5)^T \end{aligned} \tag{35}$$

The resulting Jacobian matrix $J_{4_{(3)}}$ is:

$$J_{4_{(3)}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\sin q_4 & \cos q_4 \cdot \sin q_5 \\ 0 & 0 & 0 & 0 & \cos q_4 & \sin q_4 \cdot \sin q_5 \\ 0 & 0 & 0 & 1 & 0 & \cos q_5 \end{pmatrix} \tag{36}$$

Because our Jacobian matrix projects from the body 3 into the frame 4, it is necessary to compute the final projection from the 3rd frame to the 0th frame.

$$\begin{pmatrix} dp_0 \\ d\varphi_0 \end{pmatrix} = \begin{pmatrix} R_{0,3} & 0_{3x3} \\ 0_{3x3} & R_{0,3} \end{pmatrix} \cdot \begin{pmatrix} I_{3x3} & -\hat{P}_{46_{(3)}} \\ 0_{3x3} & I_{3x3} \end{pmatrix} \cdot J_{4_{(3)}} \cdot dq \tag{37}$$

In (37) the $R_{0,3} = \prod_{i=1}^{j} R_{i-1,i}$ and it can be represented as product of three matrices (38).

$$\begin{pmatrix} dp_0 \\ d\varphi_0 \end{pmatrix} = \begin{pmatrix} R_{0,1} & 0_{3x3} \\ 0_{3x3} & R_{0,1} \end{pmatrix} \cdot \begin{pmatrix} R_{1,2} & 0_{3x3} \\ 0_{3x3} & R_{1,2} \end{pmatrix} \cdot \begin{pmatrix} R_{2,3} & 0_{3x3} \\ 0_{3x3} & R_{2,3} \end{pmatrix} \cdot J_{4_{(3)}} \cdot dq \tag{38}$$

To simplify the whole operation we start computation from right to left. The first two components from the left side of (38) can be taken in order to compute their product designated as x_1

$$x_1 = J_{4_{(3)}} \cdot dq = \begin{pmatrix} dq_1 \\ dq_2 \\ dq_3 \\ -\sin q_5 \cdot dq_5 + \cos q_4 \cdot \sin q_5 \cdot dq_6 \\ \cos q_4 \cdot dq_5 + \sin q_4 \cdot \sin q_5 \cdot dq_6 \\ dq_4 + \cos q_5 \cdot dq_6 \end{pmatrix} \tag{39}$$

It is convenient moment to substation (40) on result (39).

$$\begin{gathered} E_1 = -\sin q_5 \cdot dq_5 + \cos q_4 \cdot \sin q_5 \cdot dq_6 \\ E_2 = \cos q_4 \cdot dq_5 + \sin q_4 \cdot \sin q_5 \cdot dq_6 \\ E_3 = dq_4 + \cos q_5 \cdot dq_6 \end{gathered} \tag{40}$$

The computation continues with the same step. By that x_2 is computed. It is the product of next component of (39) and x_1 (41).

$$x_2 = \begin{pmatrix} R_{2,3} & 0_{3x3} \\ 0_{3x3} & R_{2,3} \end{pmatrix} \cdot x_1 = \begin{pmatrix} dq_1 \\ -dq_3 \\ dq_2 \\ E_1 \\ -E_3 \\ E_2 \end{pmatrix} \tag{41}$$

Computation continues in the same way in order to get final projection.

$$x_3 = \begin{pmatrix} R_{1,2} & 0_{3x3} \\ 0_{3x3} & R_{1,2} \end{pmatrix} \cdot x_2 = \begin{pmatrix} dq_3 \\ -dq_2 \\ dq_1 \\ E_3 \\ -E_2 \\ E_1 \end{pmatrix} \tag{42}$$

The final projection (43) is computed when all the components are multiplied into one product. The final transformation expresses the dependency of the joint configuration change on the change of the end-effector position and orientation.

$$\begin{pmatrix} dp_0 \\ d\varphi_0 \end{pmatrix} = \begin{pmatrix} R_{0,1} & 0_{3x3} \\ 0_{3x3} & R_{0,1} \end{pmatrix} \cdot x_3 = \begin{pmatrix} dq_2 \\ -dq_1 \\ dq_3 \\ E_2 \\ -E_1 \\ E_3 \end{pmatrix} = \begin{pmatrix} dq_2 \\ -dq_1 \\ dq_3 \\ \cos q_4 \cdot dq_5 + \sin q_4 \cdot \sin q_5 \cdot dq_6 \\ \sin q_5 \cdot dq_5 - \cos q_4 \cdot \sin q_5 \cdot dq_6 \\ dq_4 + \cos q_5 \cdot dq_6 \end{pmatrix} \tag{43}$$

4 Control Program

The main aim is to develop a control program which substitutes the existing one. The program must allow the user to control the robot with a user-friendly interface. It must support several modes of operation. Also it is necessary to build a platform which is easy to extend and prepared for future work.

The main objectives which should the program implement:

- Homing Mode
- Profile Position Mode
- Profile Velocity Mode
- Feedback Control
- Vision-Based Navigation

4.1 Design

A development under the LabView requires a specific approach. All the programming done by block diagrams is directly connected with the interface on a front panel. Therefore the conception cannot be compared to any common pattern.

The application can be seen as a hierarchy of many VIs. On the top of this hierarchy are the VIs which are used by users to control the program. They are the most complex ones and they envelope all the other VIs. On the other side of the hierarchy are the most simple VIs. Each of them represents a single fundamental function.

4.1.1 VI Hierarchy

The control program is built from 102 different VIs. They are divided in several groups with regard to their complexity and purpose. There are 7 different groups: Main, Advanced, Fundamental, Sub, PDO and Computation. To distinguish VIs from different groups, a name prefix of each VI refers to the group in which the IV belongs.

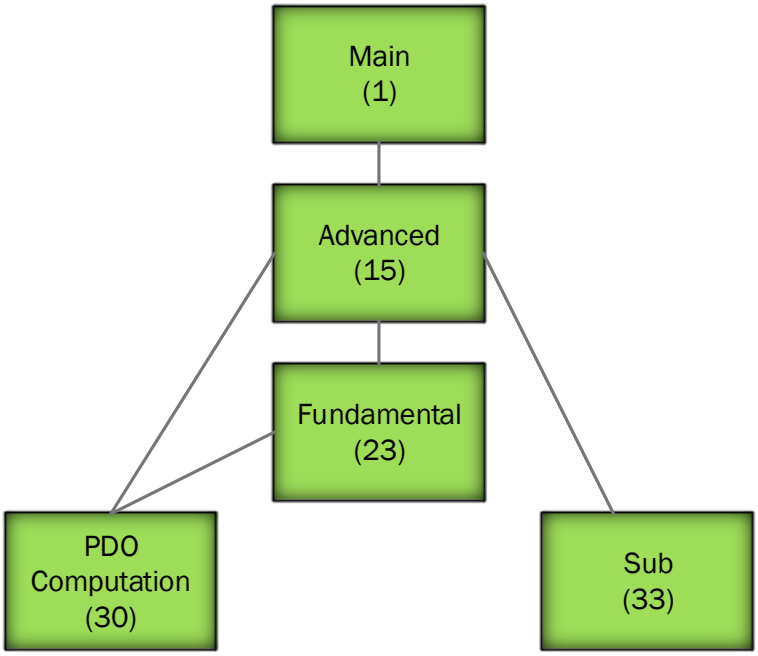

Figure 22 The hierarchy of used VIs. Numbers of VIs in each group are in the parenthesis.

Figure 22 shows the hierarchy of the VIs used to build the control program. The top VI belong to the Main group. There is only one main VI. This VI is used to execute the program. The second group is Advanced. This group contains the VIs which are seen and controlled by the user. They have user-friendly interviews, various buttons, charts and indicators. The Advanced group VIs are mainly composed from the Fundamental and Sub group VIs. The Fundamental group VIs are simple functions and subroutines. The VIs from the Sub group are mainly used to display information about the motor controllers. Handling of PDOs is done by the VIs in PDO group. Other support computations are done by VIs from the last group.

4.1.2 Program Workflow

The main program is composed from several interfaces which are used by user. Figure 23 displays the basic workflow of the application.

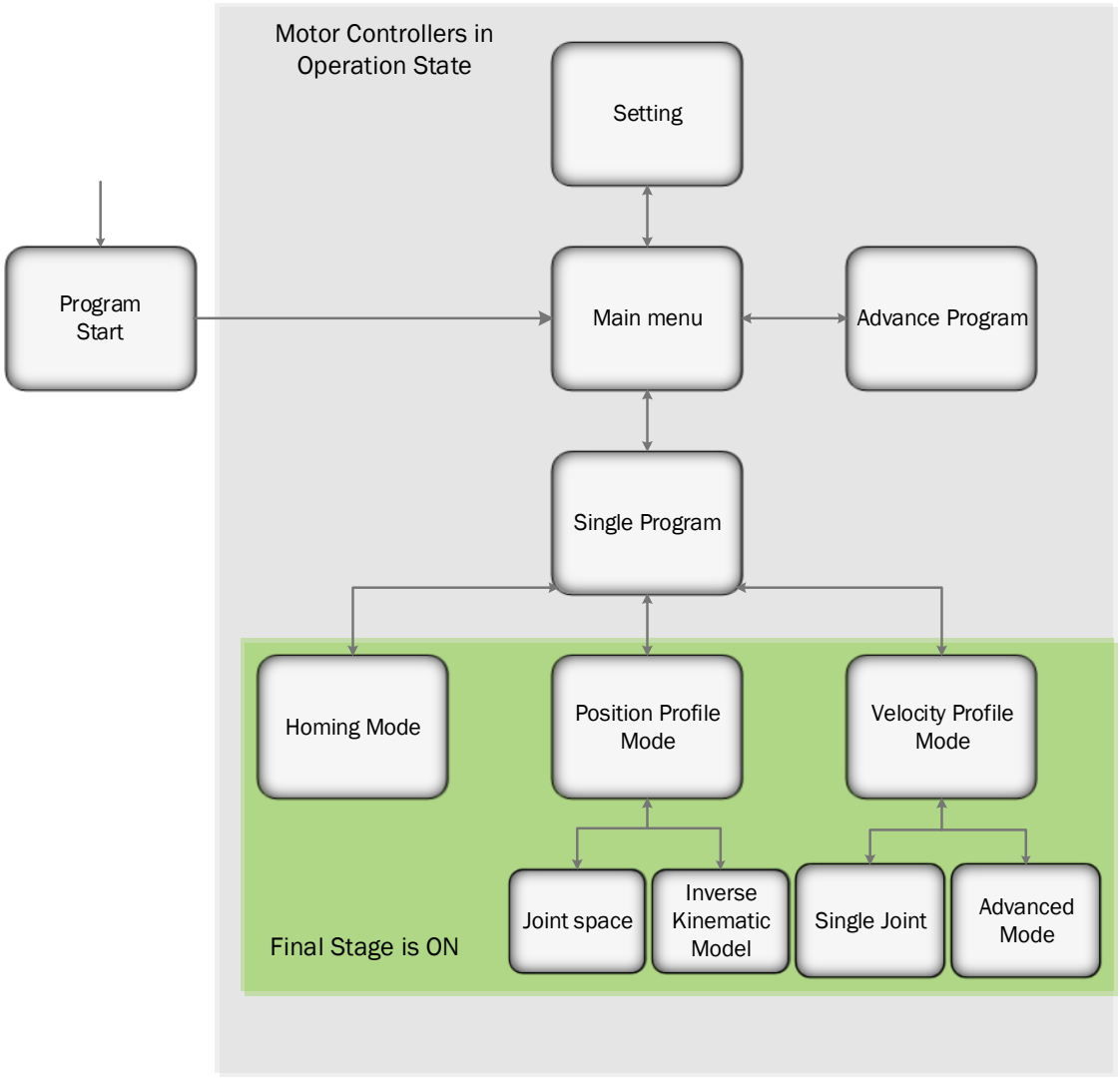

Figure 23 Program Workflow

When the program is executed the initiation phase starts. It is an automatic phase, where instances of CANopen interface and PDOs are created. When everything is prepared the program sets all the controllers in the operational state. Then the application shows the main menu (Figure 28) where the user takes the initiative.

The user can either access the setting menu, single program menu or advance program menu. The advance menu will enable user to program robot to do more difficult tasks. For now no advance program is implemented. Within the setting menu it is possible to review all the parameters of each motor controller and modify them by SDO messages. The setting menu also offers to user possibility of the simple PDO control, the heartbeat protocol and the tool for mapping PDOs. This menu can be easily extended with other functions. Last option leads user to the single program menu (Figure 29). From there user can choose between different operating modes of the motor controllers. A first thing to do when the user choose one of the operating modes is switching on the voltage. Then the user can choose the operating mode. The voltage is switched off automatically whenever the user goes back to the single program menu. Particular options of the single program menu are discussed further in this chapter.

4.1.3 Used PDOs

In order to control the robot in real-time, some of the objects must be read and modified by PDO to avoid having a any delay. In total five PDOs are used: two TPDOs and three RPDOs. Both TPDOs are set, so the controller sends them periodically every 40ms. The motor controllers are set so they processed all three RPDOs whenever they arrive. Table 27 show what objects are mapped to each PDO. All the motor controllers have mapped the same objects.

PDO	Mapped Objects
TPDO1	Status word, actual position value
TPDO2	Velocity actual value, velocity actual value filtered
RPDO1	Control word
RPDO2	Target Position, End Velocity
RPDO3	Target Velocity

Table 27 Used PDOs in the program

4.1.4 Setting Menu

The setting menu is a very important part of the program, because it allows the user to see all the parameters of each motor controller. The setting menu is displayed in Figure 24. In the left side of the menu are the node selector and the schema of the robot. A cluster menu with many buttons is in the middle and a heartbeat protocol interface is on the right side of the window. This interface gives the user an overview about the current NMT states of the controllers.

From the menu, the user can negotiate all the different parameters of the controllers and modify them with SDO write function. Such actions require deeper knowledge of the system. Therefore any change of the parameters must be done only with the FESTO documentation. It is also possible to modify the mapped objects of any PDO. The user can also modify the NMT state of any controller or modify the current operating mode. This is useful especially with the PDO simple control while testing new functions.

Some important facts must be mentioned. For example it is not possible to set a device into pre-operational state and then run the PDO simple control, because PDO is not available in pre-operational state. On the other hand it is not possible to link PDOs while the controller is in the operational state. For this task the controllers have to be set in pre-operational state. If any error occurs it is displayed in the error boxes. These error boxes shows only non-critical errors, which does not relate with the robot. Such errors mostly occur when a data type is misplaced. The object indexes, sub-indexes and data types can be found in FESTO documentation along with the explanation of all the parameters.

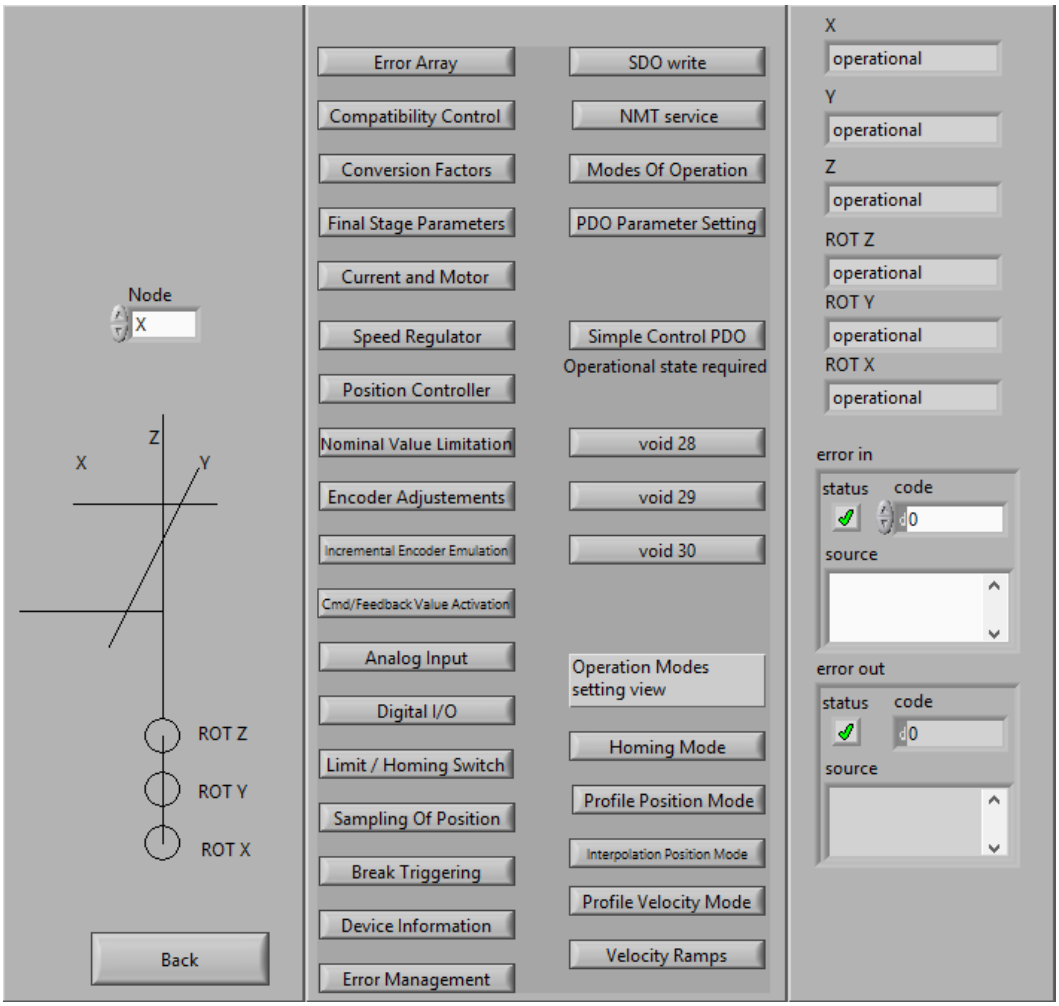

Figure 24 The setting menu of the control program

4.1.5 Homing Mode Control

The homing mode is a simple but very frequently used function. It is necessary to set the robot joints to their home positions after each reboot. The homing mode control allows the user to switch on the final stage and then choose homing of one or all the joints. The first RPDO and TPDO is used in this VI.

There are four buttons which allow the user to: switch on the final stage, activate homing for only one selected joint, activate homing for all joints and go back to previous menu. In total four led-diodes indicate if the homing mode was set properly, if the final stage was activated, if any drive is moving and if any error occurred while executing the homing function. If the user switches on the voltage then it is switched off again while the user goes back to the previous menu.

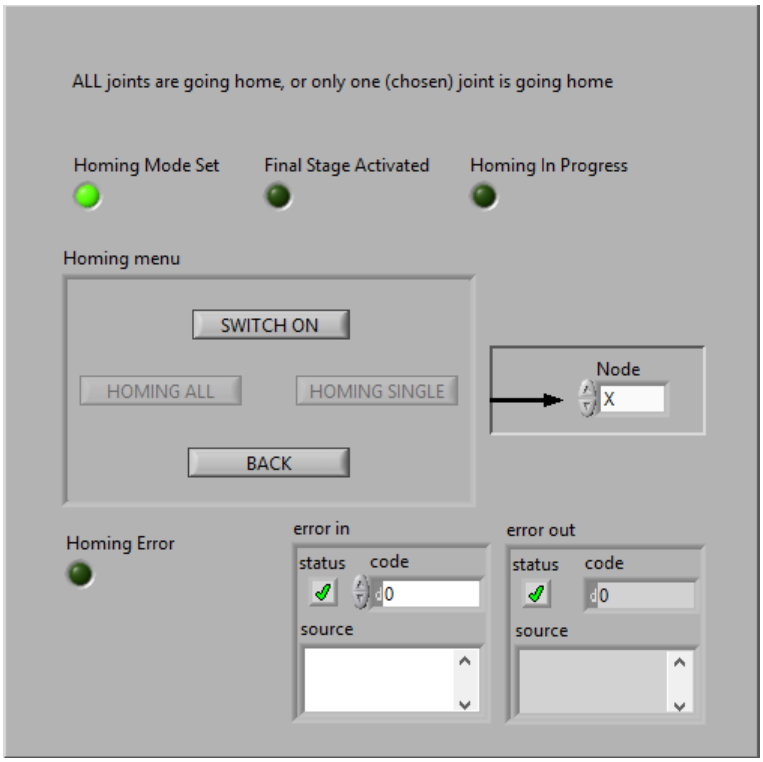

Figure 25 The homing mode control interface

4.1.6 Profile Position Mode Control

The functionality of the profile position mode was described in the chapter concerning the operating modes. This mode allows to move the drive to the desired position. All the positions of the controllers are given in the real units. For the prismatic joints, position is given in millimeters and for revolute joints in degrees. Again the user can choose if only one joint or all joints will be driven to the target position.

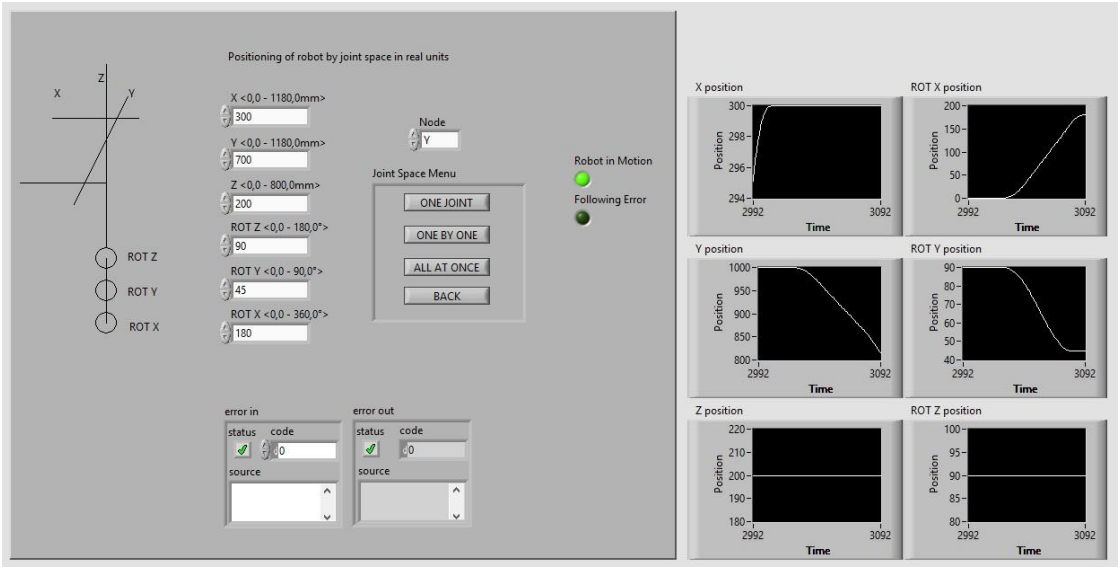

Figure 26 The position profile control interface of the control program

As can be seen in Figure 26, the position profile control interface is divided into two parts. The left part provides the control of the robot. It contains six input fields for entering the target position for each joint. There are three buttons for executing the reference travel of one or all joints

and one button for going back to the previous menu. Then there are two led diodes to signal if the robot is moving or if an error has occurred. The right part displays the position feedback of each joint in real-time with 100ms period.

Another interface will be implemented to position the end-effector in the task space (Cartesian coordinates).

4.1.7 Profile Velocity Mode Control

The last single program control deals with the profile velocity mode. So far it is the most advanced control. From the velocity profile menu it is possible to control only one joint or all the joints together. Using of this mode is very important for future work, where the close loop control will be implemented. It is required for more complex tasks which will the robot do.

In order to move each drive with a smooth velocity trapezoid profile it is necessary to compute the time when the drive must start deceleration. If the user wants to move all the joints at once, then it is necessary to compute a separate target speed for each joint, so all the joints start and stop the motion together.

Assume that q_1^* is the target position, q_0^* is the current position, T is the total time of travel, Δ is the time in which the drive accelerates from rest to maximum velocity v_{max}, or decelerate from maximum velocity v_{max} to zero velocity.

$$q_1^* - q_0^* = (T - \Delta) \cdot v_{max} \tag{44}$$

If the drive accelerates with a linear acceleration and decelerates with a linear then the deceleration value of Δ can be easily computed:

$$\Delta = \frac{v_{max}}{a} \tag{45}$$

From equations (44) and (45) it is possible to express the value of the total time of travel T.

$$T = \frac{q_1^* - q_0^* + \frac{v_{max}^2}{a}}{v_{max}} \tag{46}$$

The time when the drive has to start deceleration is evident:

$$T_{stop} = T - \Delta \tag{47}$$

In order to plan the motion of all the joints, so they all travel within the same time, the maximum velocity of the travel for each joint must be adjusted. First it is necessary to compute the time of travel for each joint (47). Then the maximum velocity is computed for all the joints but the one with the maximum time(48). By doing this it is possible to achieve that all the joints start and finish their motion in the same time.

$$v_{max} = \frac{T - \sqrt{T^2 - \frac{4}{a} \cdot |q_1^* - q_0^*|}}{\frac{2}{a}} \tag{48}$$

In Figure 27 the control interface for one joint under the profile velocity mode is displayed. This interface can be divided into two parts. The left part deals with control while in the right part the velocity and position feedback are plotted in real-time.

In this interface there are in total three buttons. One button is used to process the input target position and compute the time to stop and total time of travel. Second is used to start the motion and the third allow user to go back to the previous menu. There are several different values displayed and two led diodes indicating if the drive is moving and the direction of the motion. On the right side of the window there are two charts. The top chart provides the actual position feedback. The second chart provides the actual velocity feedback.

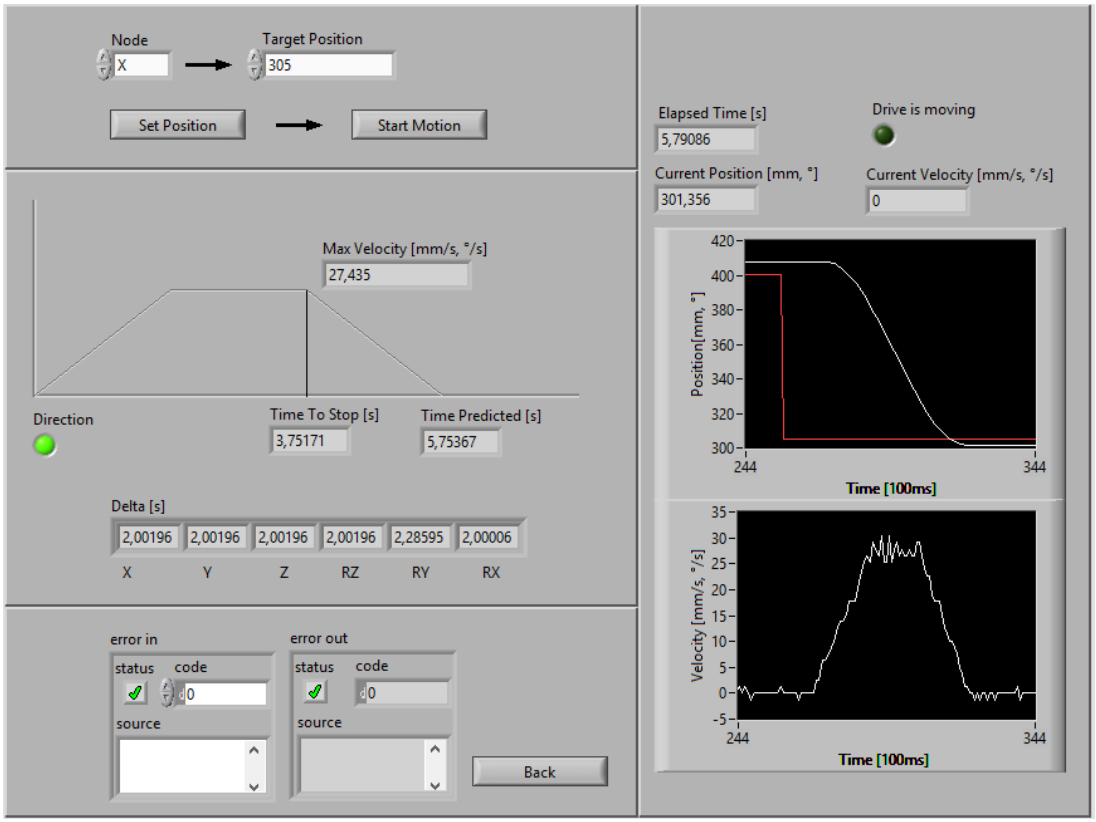

Figure 27 Profile velocity mode control interface for one joint

The control menu for positioning all the joints is only in a working version. It has six input fields in order to entre target position of all the joints. Once the position is set, it computes the adjusted velocities for all the joints but the slowest.

There is also an interface prepared for feedback proportional control. It samples the trajectory of the drive and it should compute the errors between the demand position values and the current position values. However this module is not ready and it is prepared the future work.

4.2 Control Program Use

To give an idea how to work with the program, the three general use cases are described. Each use case regards one operating mode. The first step is the same for all three cases.

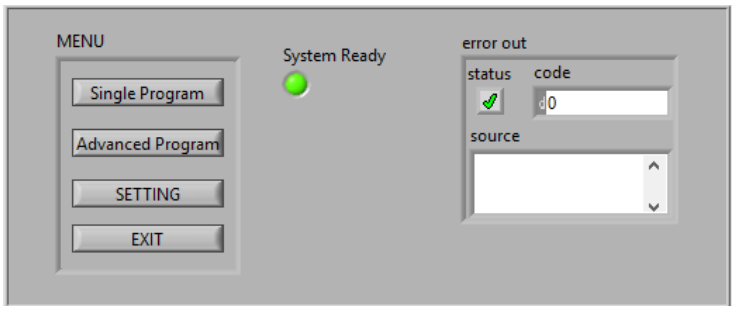

Figure 28 Control program main menu

When the control program starts the main menu appears (Figure 28). The program establishes the communication and the led diode "System ready" signals that system is ready. Only then it is possible to continue by clicking on the button "Single Program".

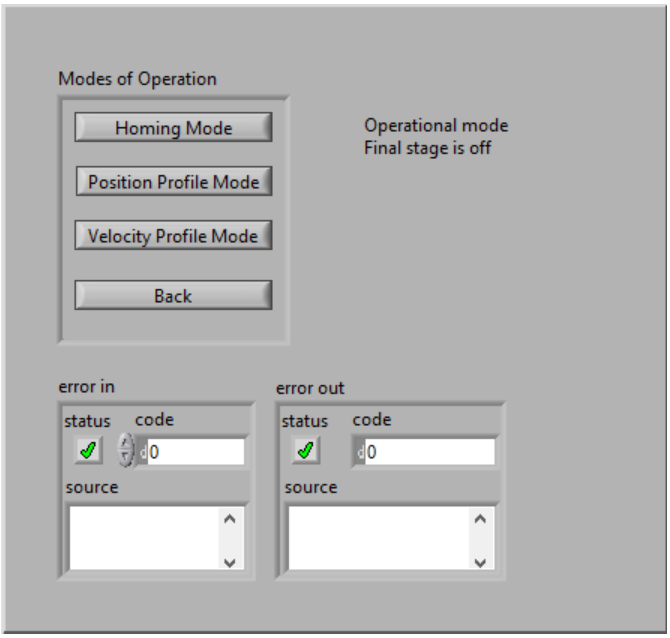

Figure 29 Single program menu

From the single program menu (Figure 29) it is possible to access the control menus of all the three operation modes. Use cases of each mode are described further.

4.2.1 Use of Homing Control

Set the robot to its home position is a very common task. It must be done after each reboot of the robot. This use case describes how to set the joint "X" to its home position.

When the single program menu appears (Figure 29) the homing mode menu is accessed by clicking on the button "Homing Mode".

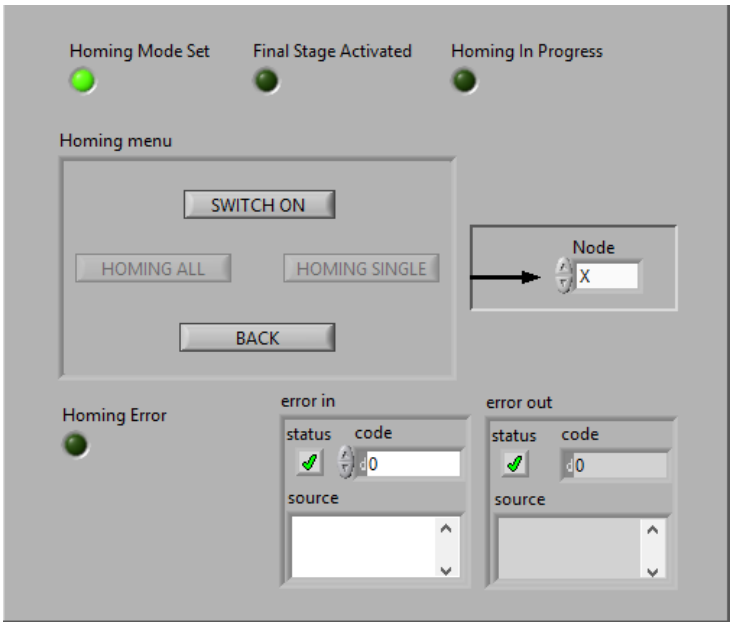

Figure 30 Homing mode control - voltage switched off

When the homing mode menu (Figure 30) appears, the program signals that the homing mode was successfully set by lighting up the led diode "Homing Mode Set". The first thing to do, before the homing function can be executed, is to switch on the voltage in the motors. This is done by clicking on the button "SWITCH ON".

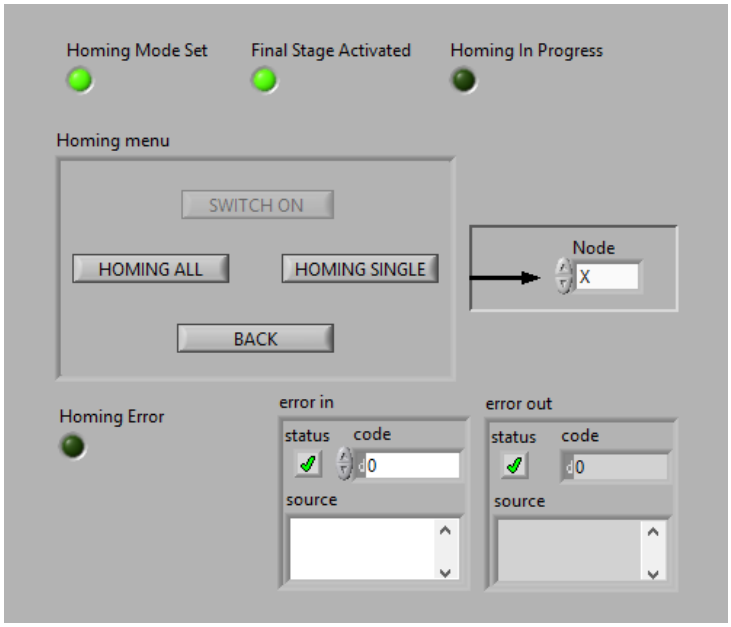

Figure 31 Homing mode control - voltage switched on

When the voltage is on the led diode "Final Stage Activated" is switched on. The joint "X" is chosen on the right side of the control window (Figure 31). Then the button "HOMING SINGLE" is pressed in order to set the joint to its home position. While the drive is moving toward its home position the led diode "Homing in Progress" is switched on. It is switched off when the drive arrives home. If any error occurs the led diode "Homing error" is lighted up and the program switches off the voltage.

4.2.2 Use of Profile Position Control

The profile position mode is a basic way how to set a drive to the target position. This use case describes how to set the joint "X" to the position 400 millimeters by using the position profile menu.

When the single program menu appears (Figure 29) the user position profile menu is accessed by clicking on the button "Position Profile Mode".

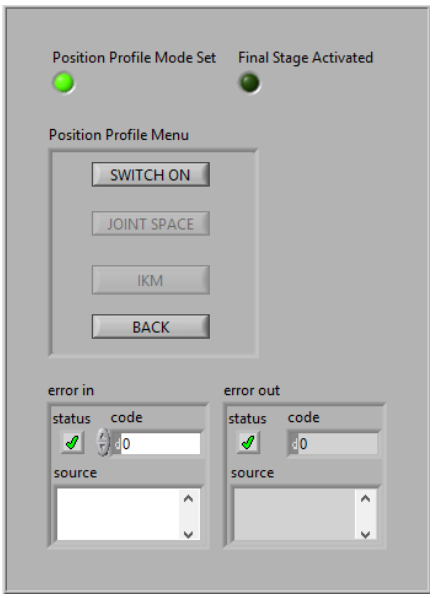

Figure 32 Position profile menu - voltage switched off

When the position profile menu (Figure 32) is opened, the profile position mode is set. It is indicated by lighting up the led diode "Position Profile Mode Set". The next step is to switch on the voltage in motors. It is done by clicking on the button "SWITCH ON".

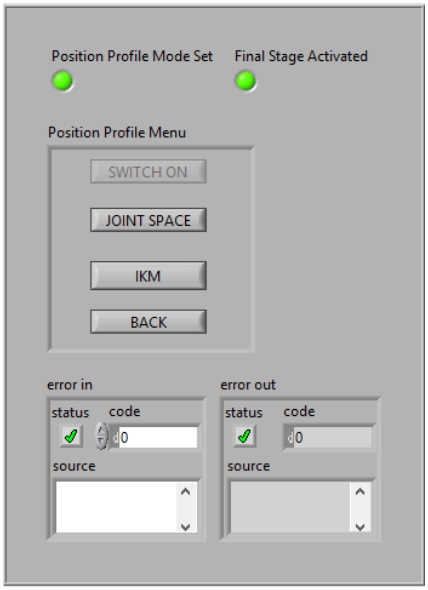

Figure 33 Position profile menu - voltage switched on

When the final stage is switched on the led diode "Final Stage Activated" is also switched on. Then it is possible to click on the button "JOINT SPACE" (Figure 33).

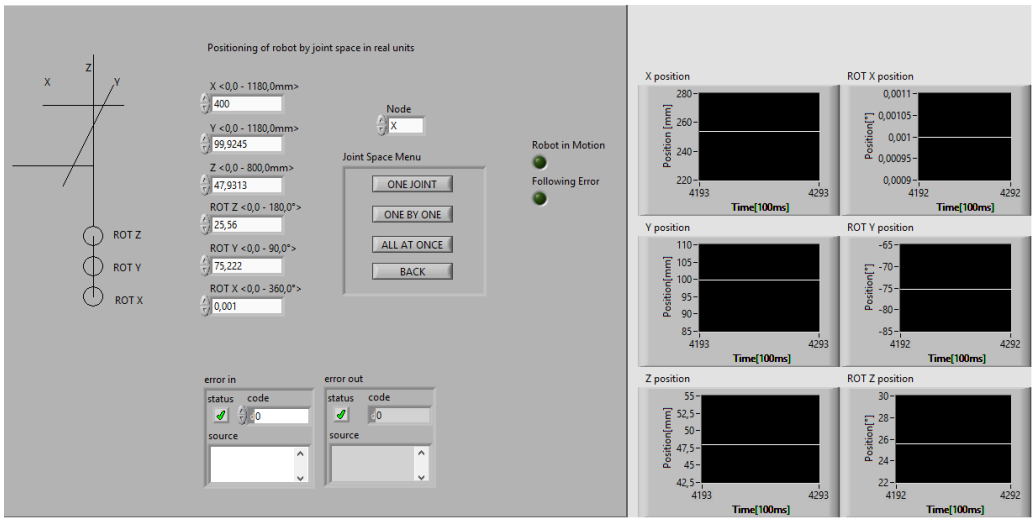

Figure 34 Position profile control - before travel

When the position profile menu appears (Figure 34) it is possible to set the target position for each joint. The target position of the joint "X" is set to 400 millimeters. After setting the target position and the joint "X" it is possible to start the motion by clicking on the button "ONE JOINT".

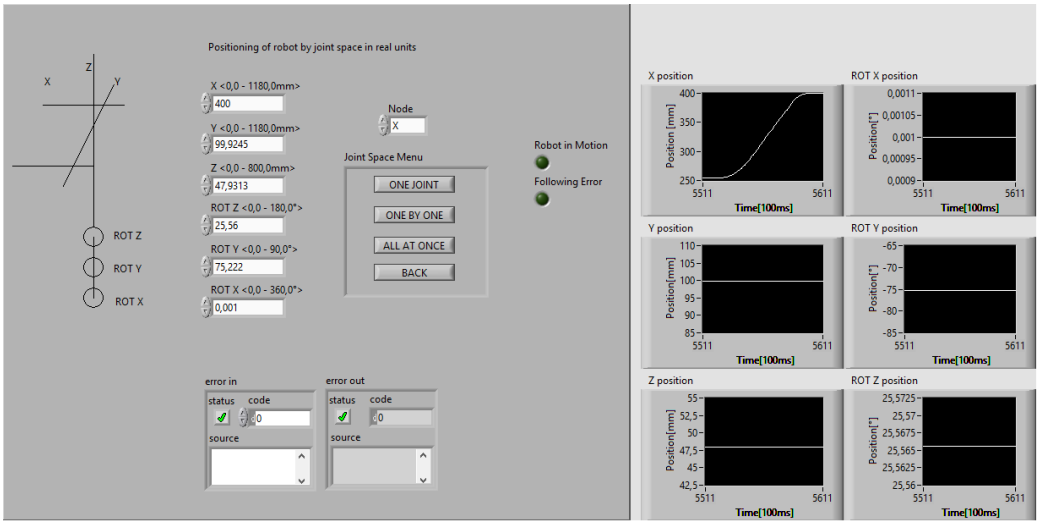

Figure 35 Position profile control - after travel

During the reference travel of the drive, the led diode "Robot in Motion" is switched on. It is again switched off once the travel is finished. On the right side of the window there is a position feedback of each joint. The chart of the joint "X" position displays the trajectory of the joint.

4.2.3 Use of Velocity Control

There are a lot of different application, where the profile velocity mode can be used. This use case describes how to use the velocity profile menu to set the joint "X" to the position 250 millimeters.

When the single program menu appears (Figure 29) the user clicks on the button "Velocity Profile Mode".

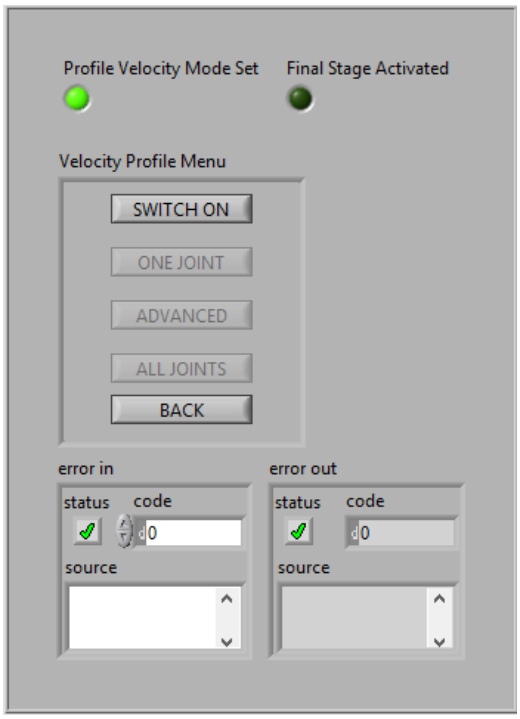

Figure 36 Velocity profile menu - voltage switched off

When the velocity profile menu (Figure 36) is opened, the profile velocity mode is set. It is indicated by lighting up the led diode " Profile Velocity Mode Set". The next step is to switch on the voltage in motors. It is done by clicking on the button "SWITCH ON".

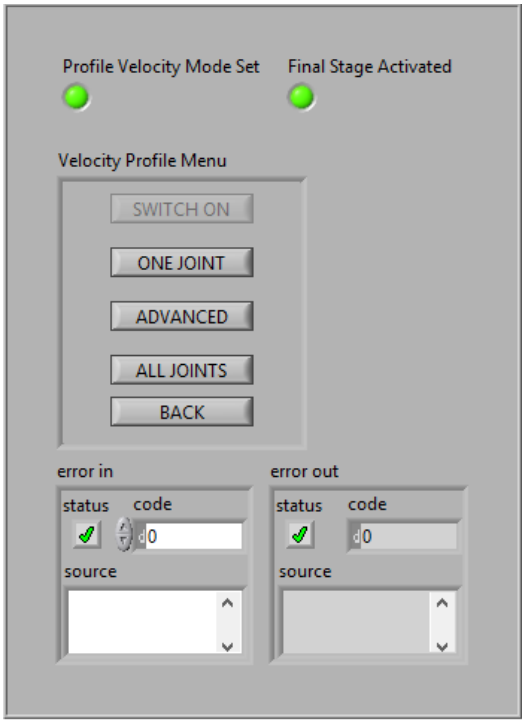

Figure 37 Velocity profile menu - voltage switched on

When the final stage is switched on the led diode "Final Stage Activated" is also switched on (Figure 37). Then it is possible to click on the button "ONE JOINT".

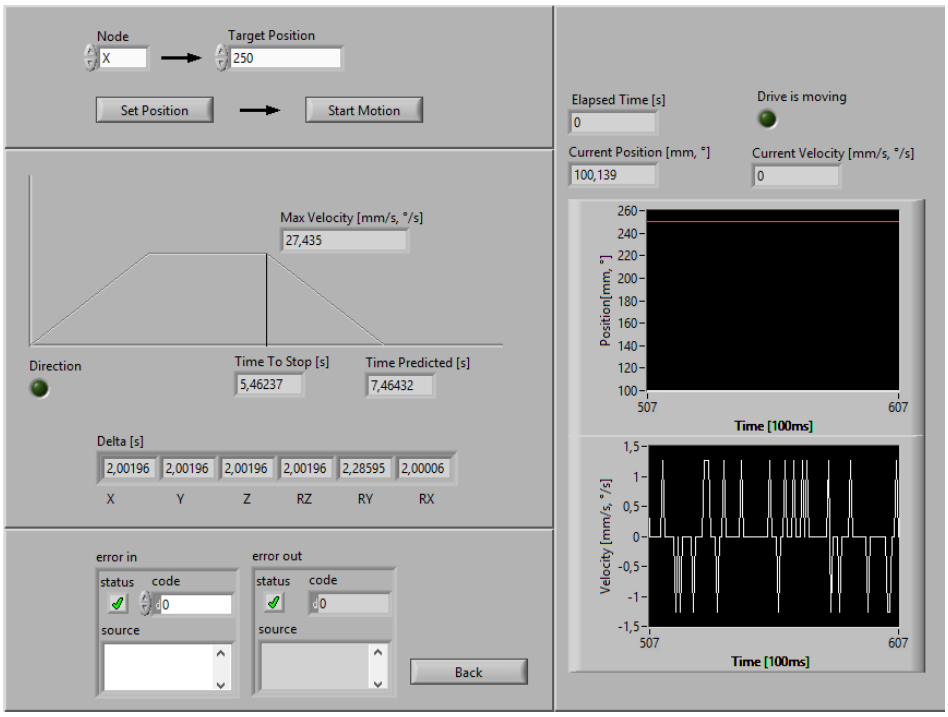

Figure 38 Velocity profile control - before travel

When the velocity profile control appears (Figure 38) it is possible to select the joint and set the target position. It this case joint "X" is selected and target position is set to 250 millimeters. The parameters of the reference travel must be computed. This is done by pressing the button "Set Position". When the button is pressed the time to stop and the predicted time of the travel are computed. Also the direction of the travel is indicated by the led diode "Direction". Everything is prepared for the reference travel. The travel is launched by clicking on the button "Start Motion".

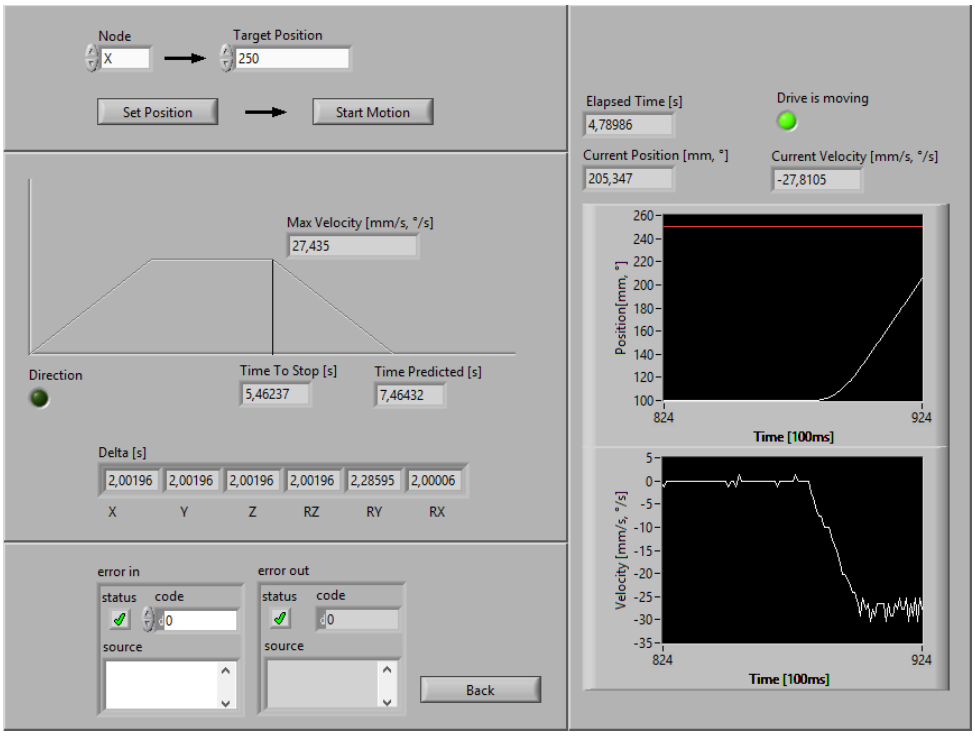

Figure 39 Velocity profile control - during travel

When the reference travel starts the led diode "Drive is moving" is switched on. The drive accelerates to its maximum velocity. Then it keeps going with the constant velocity until the time of the travel is equal to the time to decelerate. On the position feedback chart on the right side of the window (Figure 39), is the target position drawn by the red line and the current position by the white line.

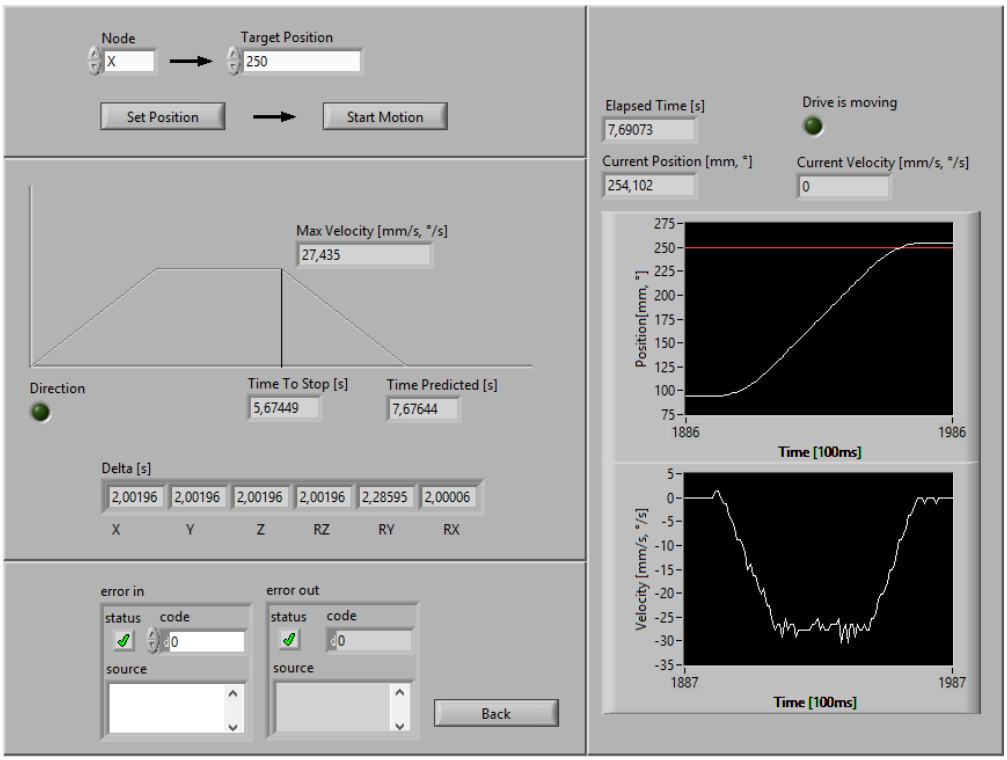

Figure 40 Velocity profile control - after travel

After the reference drive is finished the led diode "Drive is moving" is switched back off. It is possible to see the trapezoid profile on the velocity feedback chart and the trajectory made on the position feedback chart. It is also possible to see that there is a 4,102 millimeters error.

Conclusion

All the work was done during the six month long internship. The implementation of the control system on a 6-joint robot is a very complex task with many different problems to solve. It requires a lot of preparations. First of all it was necessary to decide which hardware and software will be used for the development. It was decided to use commercial solution from company National Instruments. The controller Compact Rio with the corresponding module was bought. The software is developed in the National Instruments environment LabView. This choice of National Instruments as a supplier can be considered as very good. The development in LabView is easy-to-learn and the community around NI products is very strong. The NI CANopen library is also very good. The first communication fundaments were established shortly after delivery of all the purchased articles.

Before the delivery the direct differential and inverse kinematic models were designed. The direct differential model is good for the future work with a mounted camera on the end-effector of the robot. The inverse kinematic model was made in order to allow the user of the system to program basic robot tasks in Cartesian space. Both models were computed by using modified Denavit-Hartenberg parameters and analytical computation methods.

In order to have safe and reliable program it was necessary to examine all of the motor controller parameters. The right values for each parameters were found and stored. After a long process, which required a lot of testing and consultations with the manufacturer of the robot, the right parameters were found. After the tuning of the parameters it was possible to start with the safe testing of all three used modes of operation. It was also possible to start work on the control program for each mode.

When everything was ready to build a control application, the architecture was designed and all the necessary functions for the application were prepared. The program enveloped all the prepared functions using both fundamental and advanced functions.

First version of the control program is made and prepared for further expansion. The program allows the user to control the robot in real-time. Currently the three different modes are available: homing, profile position and profile velocity. The module with control of profile velocity mode is made so the feedback control can be implement.

Future Work

There are many different things to implement in the future. The program must be developed further along with the implementation of the other parts.

The first part to implement is the position feedback control. It will use first the proportional controller, later the proportional-integral-derivative controller. The next part to implement is the inverse kinematic model. This will allow the user to give the tasks to the robot in the Cartesian space. Another part to implement is an interface which will allow the user to program more complex robot trajectory.

In the future a camera will be mounted on the end-effector. When there is a module for image processing and the camera is mounted, the vision-based navigation can be implemented. It will be another large part of the system.

References

[1] BAYLE, Bernard. UNIVERSITÉ LOUIS PASTEUR DE STRASBOURG. Introduction à la Robotique. Strasbourg, 2005. Dostupné z: http://eavr.u-strasbg.fr/~bernard/education/master_gsb/poly_master_gsb.pdf

[2] BOIMOND, Patrice, Antoine DUSTON, Nicolas PETIT a Mathieu SALAMON. LAAS-CNRS. Compte Rendu de TER: Implémentation d'un système de commande générique sur un robot 6 axes. Toulouse, 2010.

[3] CiA. [online]. [cit. 2013-05-16]. Available on: http://www.can-cia.org/

[4] FESTO. Manual: CANopen for Motor Controller CMMP. Esslingen: FESTO, 2008.

[5] GROSSMANN, Jérémy. LAAS-CRNS. Implémentation d'un système de commande générique d'un robot six axes. Toulouse, 2010.

[6] LaValle, Steven. Planning Algorithms. Cambridge University Press, 2006.

[7] SPONG, Mark W. Robot modeling and control. Hoboken: John Wiley, 2006, 478 s. ISBN 04-716-4990-2