ISSN 0280-5316
ISRN LUTFD2/TFRT--5911--SE

Stiffness Compensation and External Control of Gantry-Tau Robots

Martin Holmstrand
Kristina Silverbåge

Lund University
Department of Automatic Control
January 2013

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
January 2013

Document Number
ISRN LUTFD2/TFRT--5911--SE

Author(s)
Martin Holmstrand
Kristina Silverbåge

Supervisor
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Klas Nilsson, Dept. of Computer Science, Lund University, Sweden
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle
Stiffness Compensation and External Control of Gantry-Tau Robots

Abstract
This master thesis describes how to compensate for the compliance of the Gantry-Tau robot. The main reason for this is to improve the performance of the robot. The compliance is modeled as nine springs, one for each cart and each link. The model seems promising but has only been tested in homing position of the robot but shows a positive result in decreasing the compliance. External control of the CNC software ISG and its drivers for a Gantry-Tau robot F1 is also investigated. A way to add external measurement equipment and adding external position references was implemented.

Keywords

Classification system and/ or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-62

Recipient's notes

Security classification

Stiffness compensation and external control of Gantry-Tau robots

Kristina Silverbåge, Martin Holmstrand

January 11, 2013

Abstract

This master thesis describes how to compensate for the compliance of the Gantry-Tau robot. The main reason for this is to improve the performance of the robot. The compliance is modeled as nine springs, one for each cart and each link. The model seems promising but has only been tested in homing position of the robot but shows a positive result in decreasing the compliance. External control of the CNC software ISG and its drivers for a Gantry-Tau robot F1 is also investigated. A way to add external measurement equipment and adding external position references was implemented.

Acknowledgment

We owe our supervisor Klas Nilsson and Anders Robertsson a great thanks.

We want to thank our German friends, Thomas Dietz, Ulrich Schneider and Manuel Durst, for all help and supervision of our work with the ISG-kernel at Fraunhofer IPA in Stuttgart.

We want to thank Simon Barth and Dominique Schaer at Güdel for the stay in Langenthal helping us getting the F1 CAD files.

Lastly we want to thank the people at the Department of Automatic Control for tips and support. Especially, Anders Blomdell, Björn Olofsson, Olof Sörnmo, August Bjälemark and Andrea Åkerström.

The research leading to these results has received funding from the EU-project MONROE and the EU-project SMErobot.

Contents

1 Introduction

The purpose of this thesis is to study stiffness of the Gantry-Tau robots at LTH and to use the identified stiffness to improve stiffness behavior. A way to implement the external commanding of the individual motor was found for the drive system of the F1 robot.

1.1 Background

The goal of the EU-project SMErobot was to construct a lightweight, flexible and cheap robot [15]. The goal was achieved with the Gantry-Tau robot D1. The robot was mainly built and developed by LTH and Güdel in cooperation with ABB Robotics. The Gantry-Tau robot is a parallel robot whose basic principles are patented by ABB in Västerås, Sweden. In the following project MONROE within EU-project Echord the robot was further developed together with Güdel and Fraunhofer IPA. The robots are currently in need of evaluation and further improvement from a control and application perspective.

The purpose of the developed robot is to fill the gap between the traditional industrial robot and CNC-machines. Traditional CNC machines are often very stiff and slow while industrial robots which often are less stiff but can achieve higher precision and speed.

This thesis is an add-on to the SMErobot project, the MONROE-project and Isolde Dresler's doctoral thesis [7].

1.2 Gantry-Tau Parallel Kinematic Machine

The Gantry-Tau robot is, as opposed to the common serial industrial robot, a parallel kinematic machine, PKM. This means that the arms are connected to the end effector in parallel, instead of in series, see Figure 1.1.

The Gantry-Tau robot can be set up in many different ways. The tool can be positioned in several different ways, different number of carts could be used, and the tool could be positioned on either sides of the carts.

All manipulators at the Robotics Lab at LTH use the same basic setup, with three linear guide ways and three carts. In the Gantry-Tau design the end-effector is coupled to the carts with 6 links placed in a 3-2-1 configuration, as seen in Figure 1.2. The links are coupled with spherical joints both to the carts and to the end-effectors. The robots used in this thesis are denoted F1 and L2, respectively.

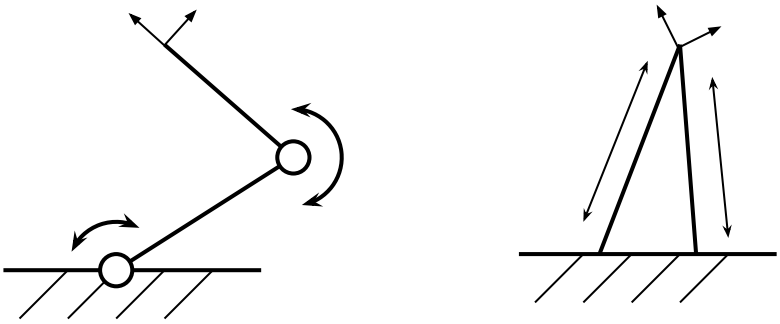

Figure 1.1: Difference between serial and parallel manipulator.

1.2.1 The L2 Gantry-Tau robot

Figure 1.2: The L2 robot.

The L2 Gantry-Tau robot is a horizontal design with 5 degrees of freedom, DOF, horizontal in the sense that the carts move horizontally. Within the scope of this thesis only 3 degrees of freedom will be considered. The reduction of the scope is made since the parallel kinematic part of the robot has 3 DOF. L2 is equipped with an ABB control system, the ABB IRC5 control system augmented with an open robot control architecture interface, ExtCtrl [7]. It is possible to con-

nect sensors or external measuring equipment, to implement custom algorithms and to connect to other equipment systems through the ExtCtrl interface. The robot currently resides at the Robotics Lab of LTH, a research and educational lab shared by the Department of Automatic Control and Department of Computer Science.

1.2.2 The F1 Gantry-Tau robot

The F1 is a vertical robot with 6 degrees of freedom. This is obtained through the possibility to rotate the arm clusters fastening points. To reduce backlash the F1 robot has been equipped with double motor control on each cart.

The F1 robot has a Beckhoff AX2000 driver system connected to a real time PC which is running an ISG NC kernel.

It is the largest robot that LTH designed, and it currently resides in Ingvar Kamprad Design Center at Lund University.

Figure 1.3: The F1 robot at the Robotics Lab of LTH, Lund.

2 Problem Formulation

2.1 Goal

The goal of this thesis is to reduce the compliance of the Gantry-Tau robot. The approach taken was to model the stiffness of a Gantry-Tau robot and use it to actively compensate for the compliance. This could also be done through improving the mechanical stiffness of both the carts and the link system.

To compensate for compliance the displacement has to be measured and fed back to the robot. An external interface for adding external sources and sensors to the system could therefore be very useful not only for stiffness measurement but for all applications involving external sensors.
To get a more clear view of the project it has been divided into the following sub-goals.

- Add velocity and torque references to the F1 motor drivers.
- Develop an external interface for the F1 control system.
- Conduct stiffness measurements on the L2 robot.
- Improve the compliance of the L2 robot.

In the beginning of this thesis the intention was to implement stiffness compensation for the F1 robot. This goal were abandoned due to the lack of the time and the lack of the software key the ISG-NC kernel controlling the robot. It was decided that the compensation should be implemented on the L2 robot, because this is the robot most similar to the F1 robot.

2.2 Demarcation

A master thesis is limited to 20 weeks. The main part of the thesis was performed at the Department of Automatic Control at LTH. Five weeks were spent at Fraunhofer Institute for Manufacturing Engineering and Automation IPA in Stuttgart, since their knowledge about the Beckhoff drivers and the ISG and TwinCAT is superior to that of LTH.

Neither of the students had any experience of robotics before the start of this thesis and this was taken into account.

There was also a short visit to Güdel in Langenthal Switzerland, where CAD files for future improvement of the decoupling between the physical and virtual

version of F1 were retrieved to Lund.
The following items will not be a part of this master thesis.

- Design and mounting of hardware on the Gantry-tau robots. Small adjustments may be made but the robots are already constructed.
- Kinematic calibrations and implementation of robot kinematics.

2.3 Individual Contributions

Kristina was main responsible for the part of understanding the kinematics, the Simulink implementation of the compensation of the L2 robot compliance and investigating the different drivers.

Martin was main responsible for the force modeling of the carts, the Heidenhain setup and implementing the external control of the F1 robot.

All measurements stiffness modeling, and the report were done in collaboration.

2.4 Outline

In Chapter 3 a short description of the kinematics and the calibration of the robots is described. The modeling of the stiffness is presented.

Chapter 4 first presents all the soft- and hardware related to implementing the external communication on the F1 robot, found in Section 4.1. Section 4.2 describes the soft- and hardware used to make the stiffness measurement on L2 and F1. Section 4.2.9 describes how the measurement was set up.

The results of this thesis are presented in Chapter 5 and are discussed in Chapter 6. Finally some future work is suggested in Chapter 7.

3 Theory

The Gantry-Tau robot differs from the traditional industrial robot mainly as it is a parallel machine and not a traditional serial robot. This chapter is intended to give the reader a basic understanding of the kinematics, found in Section 3.1, and kinematic calibration, see Section 3.2, of the parallel robot. The modeling of the stiffness of the Gantry-Tau robots can be found in Section 3.3.

3.1 Kinematics

The L2 and F1 are a 5-DOF and 6-DOF robot, respectively. The L2 robot's serial wrist was not considered in this thesis, therefore it is seen as having 3-DOF. Section 3.1.1 refers to the L2 robot kinematics and Section 3.1.2 refers to the F1 robot kinematics.

The kinematics models and calibration routines have been implemented by Dr. Isolde Dressler.

3.1.1 Kinematics of the 3-DOF Gantry-Tau

The kinematics describes the relation between the position of actuators, q_i, the tool position, $T = (T_x, T_y, T_z)^{\mathrm{T}}$, and the orientation R_T. The kinematics describes how the robot is configured, c, and depend on the physical dimensions, s, of the robot. Several solutions can exist, s and c both needs to be specified to distinguish which solution is used. The forward kinematics were the joint position is given and the position of the end-effector need to be determined is

$$(T, R_T) = f_{fk}(q, s, c) \tag{3.1}$$

For the inverse kinematics the end-effector position is given and the joint position is to be determined.

$$q = f_{ik}(T, R_T, s, c) \tag{3.2}$$

The inverse kinematics is in general easier to solve than the forward kinematics for PKM machines. The kinematic chain, i, is the connection between the actuator's attachment point, A_i, and the end effector attachment point, B_i, in Figure 3.1. The kinematic constraint for the each kinematic chain can be written as

$$L_i u_i = A_i(q_i) - B_i(T, R_T) \tag{3.3}$$

where u_i is the unitary vector along link i with length L_i or as a scalar equation

$$L_i^2 = ||A_i(q_i) - B_i(T, R_T)||^2 \tag{3.4}$$

The carts and the cluster of links are numbered according to the number of links in the kinematic chain, see Figure 3.1. There are two different frames used, the end effector frame and the global frame. The ball joint position A_{ij} and B_{ij} in the kinematic constraint, Equation 3.3 or Equation 3.4, can be written as

$$A_{ij} = A_{ij}^0 + q_i v \tag{3.5}$$

$$B_{ij} = T + R_T B_{ij}^0 \tag{3.6}$$

where A_{ij}^0 is the ball joint associated with $q_i = 0$, i is the arm, j is the link and v is the direction of the guideways.

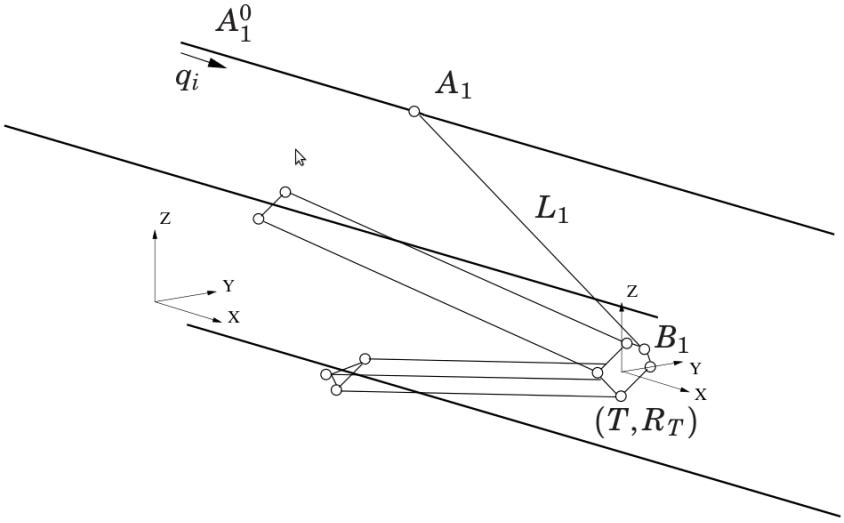

Figure 3.1: Schematics of the Gantry-Tau PKM.

It is possible to consider each cluster of links as one link, thanks to the 3-2-1 configuration of the Gantry-Tau, as seen in Figure 3.2. This is called the nominal kinematic. The joint position for the nominal case can be described as

$$A_i = (A_{ij}^0 - R_T B_{ij}^0) + q_i v = A_s^0 + q_i v \tag{3.7}$$

$$B_i = T \tag{3.8}$$

where A_i^s is the ball joint associated with $q_i = 0$ for the simplified robot. Each i is now associated with $i = \{1, 2, 3\}$ and the ball joint is associated with $q_i = 0$

The kinematic constraint for link i is therefore

$$L_i^2 = ||A_i^s + q_i v - T||^2 \tag{3.9}$$

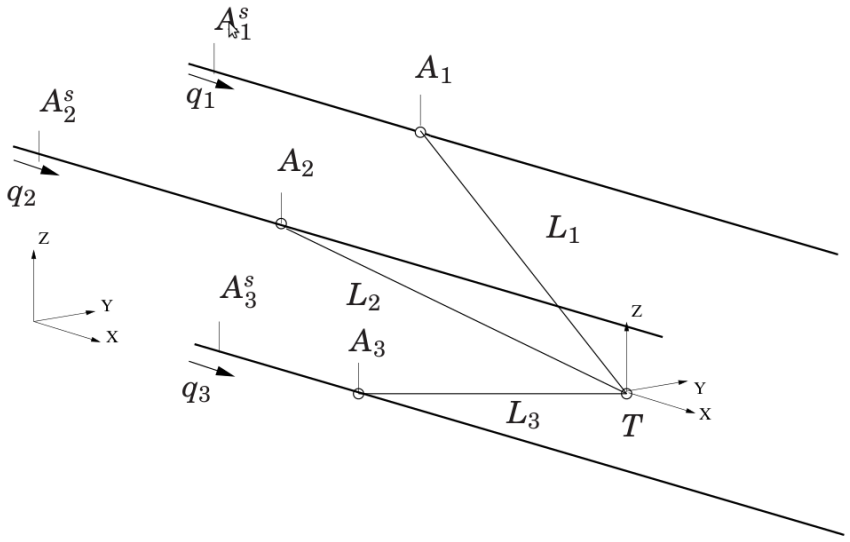

Figure 3.2: Simplification of the Gantry-Tau [7].

3.1.2 Kinematics of the 6-DOF Gantry-Tau

The carts in this configuration are numbered by the number of links there are in the kinematic chain. The difference to the L2 configuration is that the ball joint fastening plate of cart 2 can rotate due to a ball screw and the 3:rd carts plate can rotate and tilt through ball screws, see Figure 3.3. Due to this the F1 has 6 degrees of freedom. The rotational axes q_4 and q_5 are parallel to each other if those two axes are synchronized ($q_4 = -q_5$) the orientation and position of the end effector could be treated separately and an analytic solution could be found, this, however, would lead to the F1 robot only having 5 DOF.

The z-axis in this configuration is vertical and parallel to the actuators, the x-axis is horizontal and the y-axis is orthogonal to the linear actuators. Note that the F1 robot compared to the L2's, which is a horizontal configured robot, is vertically configured.

Because of the six actuators of the configuration the kinematic chains is far more complex than for the L2 case. The A_{ij}^{bj} are not considered in the global frame but in moving plates frames. Because of this arm 1 has an analytical solution for the inverse kinematic which can be found like in the 3 DOF case, but for arm cluster 2 and 3 a standard Newton-Raphson iterative algorithm was applied. The forward kinematics are also solved with the Newton-Raphson method but based on a general Stewart platform kinematics. The kinematics for the F1 robot were developed by Adam Nilsson and implemented by Isolde Dressler [7].

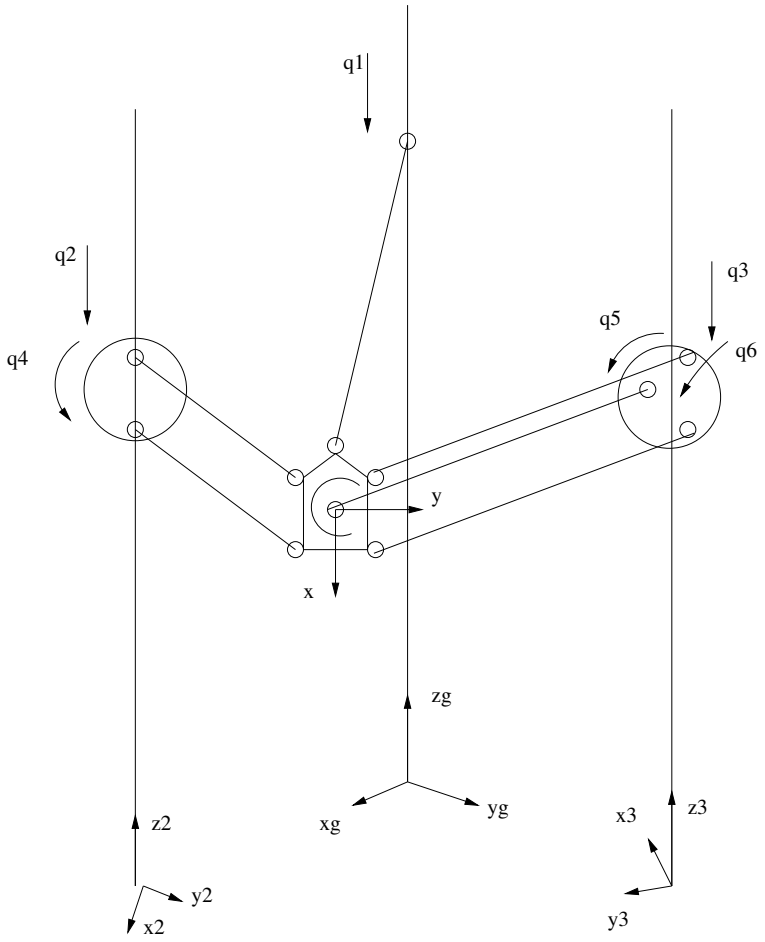

Figure 3.3: Kinematics for F1 [7].

3.2 Kinematic Calibration

The purpose of the kinematic calibration is to improve the robot accuracy.

This section is a brief overview of the calibration routines. The Gantry-Tau robots do not have any built in sensor for self calibration. Therefore an external measuring device has to be used.

3.2.1 Calibration of the 3-DOF Gantry-Tau

The calibration for the L2 robot was done with a Nikon K600 camera tracking the end effector pose of the robot. For the nominal kinematic model 3 dimensional measurements are adequate. For a set of N joint positions q_k and the corresponding pose of the end effector $(T_{m,k}, R_{T,m,k})$ one can identify the kinematic parameter set s by minimizing the cost function V. The cost function for the nominal case, which is used to calibrate the L2 robot is

$$V = \sum_{j=1}^{3} \sum_{k=1}^{N} (||A_j^s + q_{j,k} v - T_{m,k}||^2 - L_j^2)^2 \tag{3.10}$$

where L is the link length and A is the ball joint position.

The end effector pose $(T_{m,k}, R_{T,m,k})$ is given in the camera frame but the nominal kinematic is in the robot frame. The optimal parameters are determined in the measurement frame. To transform those to the robot frame for the 3 dimensional case one can use

$$T_{m,k} \approx H_{rb}^{mb} f_{fk,tr}(q_k, s_0, c) \tag{3.11}$$

where H_{rb}^{mb} is the transformation between the robot base and the measurement base with the translational part $f_{fk,tr}$ of the forward kinematic.

If nonparallel guideways are assumed the kinematic model of the L2 gives 21 scalar parameters.

3.2.2 Calibration of the 6-DOF Gantry-Tau

The fact that calibration of the F1 robot is more complicated than the L2 robot is not hard to imagine. Beside the rotational joints the robot has double motors to minimize the backlash. Because of the F1 kinematic 6D measurements of the end effector position and orientation are necessary. The calibration is discussed in detail by Isolde Dressler [7].

3.3 Stiffness model of the Gantry-Tau

The straightforward way to model the Gantry-Tau stiffness is to model each link as a spring.

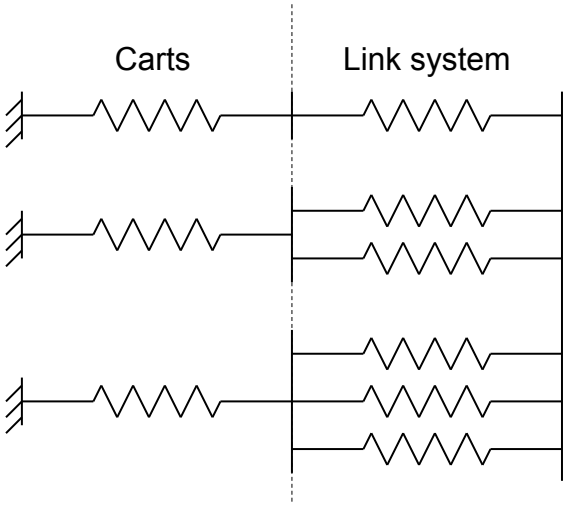

Figure 3.4: Model of the arm system and the carts in relation to their rails.

To include the stiffness of the carts in relation to their rails, the compliance is modeled as springs but only including the compliance in the direction of the guideways, see Figure 3.4.

The compliance of the link system and the carts can be investigated separately since the compensations can be made separately. A force will give rise to a displacement that could be divided into two. One for the carts and one for the arm system. The displacement of carts can be compensated for directly. The displacement of the link system is a Cartesian displacement and the reference can for the carts can be obtained with the inverse kinematics, see Figure 3.5.

The model will not take into account any other deformation of the carts than in the guideway direction, since the compliance in other directions is likely to be negligible.

The resulting force on the carts can be calculated from the force exerted to the tool center point, TCP. This can be calculated in two ways, by using the Jacobian matrix, or by decomposing the forces in the link system.

All the variables in this Section are collected in the Table 3.1.

$\delta\theta$	Infinitesimal rotation.
$\overline{F}_{links}$	Column vector containing the magnitude of each link force.
$\overline{F}_{tcp}$	Cartesian force in global frame of the end effector.
$\overline{F}_{cart}$	Force placed on the carts.
J	The Jacobian matrix.
K	The stiffness matrix.
k_i	Spring constant i.
$\overline{\ell}_i$	Distance from the end effector to spherical joint i center, in end effector coordinate frame.
$\overline{M}_{tcp}$	Cartesian torque in global frame of the end effector.
q	Position of cart along the carts.
$\overline{r}_{cart}$	Displacement of cart.
dr	Infinitesimal translation.
$\overline{s}_i$	The direction of an individual link.
S_{links}	Matrix with the individual link direction as columns.
T	Matrix with the cross product of $\overline{\ell}_i$ and $\overline{s}_i$ as columns.
$\overline{\tau}_i$	Torque in end effector frame.
$\hat{\tau}$	Matrix with torques as its columns.

Table 3.1: Variables found in Section 3.3.

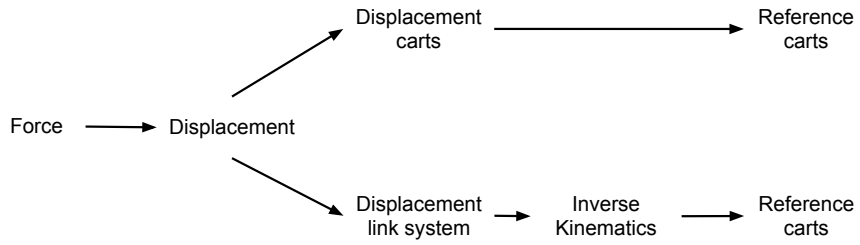

Figure 3.5: Compensation of the carts and the the links system and how to obtain the references to the motors.

3.3.1 Stiffness of the link system

The total stiffness of the link system of the Gantry-Tau robot can be defined as in [22]

$$d\overline{F} = K\,\delta\overline{q} \tag{3.12}$$

The forces and torques affecting the end effector are represented by a vector $\overline{F}$, where both components are represented in the global Cartesian coordinate system.

$$\overline{F} = \begin{pmatrix}\overline{F}_{tcp} & \overline{M}_{tcp}\end{pmatrix}^{\mathrm{T}}$$

$$with\ \overline{F}_{tcp} = \begin{pmatrix}f_x & f_y & f_z\end{pmatrix}^{\mathrm{T}},\ \overline{M}_{tcp} = \begin{pmatrix}m_x & m_y & m_z\end{pmatrix}^{\mathrm{T}}$$

$$and\ \delta\overline{q} = \begin{pmatrix}dr\\ \delta\theta\end{pmatrix} \tag{3.13}$$

where the pair $(dr, \delta\theta)^{\mathrm{T}}$ represents the infinitesimal translation and infinitesimal rotation, respectively. The load is composed by the Cartesian force and the torque around each axis, defined in the global frame. The infinitesimal rotation is assumed to be zero, ($\delta\theta = 0$), since there is no way of compensating for this with a 3-DOF link system.
The stiffness-matrix K is given by

$$K = \sum_i k_i \overline{S}_i \overline{S}_i^{\mathrm{T}} \tag{3.14}$$

where k_i is the spring constant for link i with S_i, defined as

$$\overline{S}_i = \begin{pmatrix}\overline{s}_i\\ \overrightarrow{OA_i} \times \overline{s}_i\end{pmatrix}^{\mathrm{T}} \tag{3.15}$$

A_i is the point where the links are attached to the carts and O is the origin,with $\bar{s}_i$ as the direction of each link defined in global Cartesian coordinate system [21]

$$\bar{s}_i = \begin{pmatrix} x_i & y_i & z_i \end{pmatrix}^{\mathrm{T}} , \; \bar{s}_i \cdot \bar{s}_i^{\mathrm{T}} = 1 \tag{3.16}$$

3.3.2 Forces exerted to the Carts

By arranging all the link directions in one matrix

$$S_{links} = \begin{pmatrix} \bar{s}_1 & \bar{s}_2 & \cdots & \bar{s}_6 \end{pmatrix} \tag{3.17}$$

and the force magnitude of each link arranged in a vector

$$\overline{F}_{links} = \begin{pmatrix} f_1 & f_2 & \cdots & f_6 \end{pmatrix}^{\mathrm{T}} \tag{3.18}$$

The sum of the forces on the end effector in matrix form is simply

$$S_{links}\,\overline{F}_{links} + \overline{F}_{tcp} = 0 \tag{3.19}$$

The torque exerted by one link can be expressed as

$$\overline{\tau}_i = \overline{\ell}_i \times (f_i\,\bar{s}_i) = f_i(\overline{\ell}_i \times \bar{s}_i) \tag{3.20}$$

With each ℓ defined as the distance from the end effector center point to each joint, see Figure 3.6, as

$$\overline{\ell}_i = \begin{pmatrix} \ell_{xi} & \ell_{yi} & \ell_{zi} \end{pmatrix}^{\mathrm{T}} \tag{3.21}$$

Then arranging the torque into one matrix can be expressed as

$$\hat{\tau} = \begin{pmatrix} \overline{\ell}_1 \times \bar{s}_1 & \overline{\ell}_2 \times \bar{s}_2 & \ldots & \overline{\ell}_6 \times \bar{s}_6 \end{pmatrix} \overline{F}_{links} = T\,\overline{F}_{links} \tag{3.22}$$

With the torques of the end effector balanced, one gets

$$T\,\overline{F}_{links} + \overline{M}_{tcp} = 0 \tag{3.23}$$

The Cartesian force and the rotational forces on the end effector combined into one matrix equation can be written as

$$\begin{pmatrix} S_{links} \\ T \end{pmatrix} \overline{F}_{links} + \begin{pmatrix} \overline{F}_{tcp} \\ \overline{M}_{tcp} \end{pmatrix} = 0 \tag{3.24}$$

Then the forces in each link can be calculated as

$$\overline{F}_{links} = -\begin{pmatrix} S_{links} \\ T \end{pmatrix}^{-1} \begin{pmatrix} \overline{F}_{tcp} \\ \overline{M}_{tcp} \end{pmatrix} \tag{3.25}$$

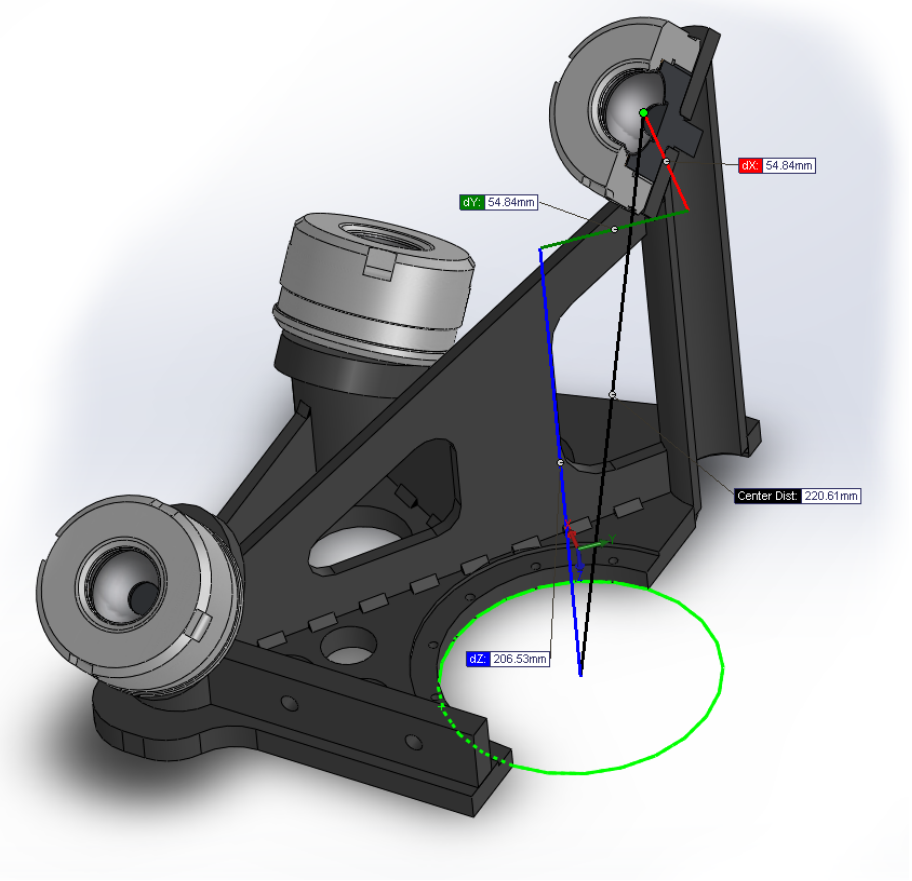

Figure 3.6: The offsets of an individual joint displayed in the CAD coordinate system.

From the forces in each individual link the forces on the modeled spring associated with each cart can be calculated as

$$|F_{cart_i}| = \sum^{links} F_{link} \cdot d_i \tag{3.26}$$

where d_i is the direction in which the cart travels.

From the force of each individual cart the displacement of the cart in Cartesian coordinates can be calculated.

$$\overline{F}_{cart_i} = -k_i \overline{r}_{cart_i} \tag{3.27}$$

3.3.3 Force exerted to the carts using the Jacobian

The force applied on the carts in the direction of the guideways can be obtained by using the Jacobian F_{cart_i} [23] since the carts are linear actuators. The relation

between F_{cart_i} and F is:

$$\overline{F}_{cart} = J^{\mathrm{T}}(q)\overline{F} \tag{3.28}$$

where q is the position of the carts along the guideway.

The spring constant of each cart is then given by Hooke's law:

$$\overline{F}_{cart_i} = -k_i\, \overline{r}_{cart_i} \tag{3.29}$$

4 Method

Section 4.1 describes the control system architecture used on the F1 robot. First a description of the ISG-NC kernel and how it is connected to the PLC and the drives.

Section 4.1.4, 4.1.5, and 4.1.6 describes tools used for developing the server and client implemented for the external communication of the F1 control system.

Section 4.1.7 and 4.1.7 contains a description of the AX2000 and the AX5000 drivers. This is because when the F1 robot was designed the idea was to equip it with AX5000 drives but because of the long delivery time it was decided to use the older generation AX2000 instead.

Section 4.2 involves the soft- and hardware including the measuring equipment to make the stiffness measurements on the L2 PKM and the carts of the F1 PKM.

4.1 External Communication for the F1 PKM

4.1.1 The ISG-NC kernel

The ISG-NC kernel software is a CNC-PC solution [8]. The software uses the RTX-environment which enables it to run realtime tasks on a Windows PC. The software can also be used with a PLC with custom hardware to extend the functionality of the machine. The ISG NC-kernel is used with the F1 robot and is developed and owned by ISG in Germany.

The common input of a CNC-program is so called G-code. The NC-kernel is capable of parsing the G-code into information that can be used to plan a path for the tool.

The path generated by the G-code is simply a linear or circular interpolation of the points or the circles defined by the G-code. However to control a machine in a good manner it is not enough to feed this directly to the drivers of the motors. The physical setup of the machine has to be taken into account. With correct data on the gear-ratios, maximum and minimum torque etc, a path can be planned. The path is sent to the drivers via a real time communication link.

The path generated is not only a position reference in time, but also a desired velocity and torque. All these values are not required by the drivers but will increase the performance of the control.

A CNC-machine's mechanical setup needs to be described in some way from the desired tool position to the position of each individual motor axis. In some more simpler setups of CNC-machines this translation, or transformation of coor-

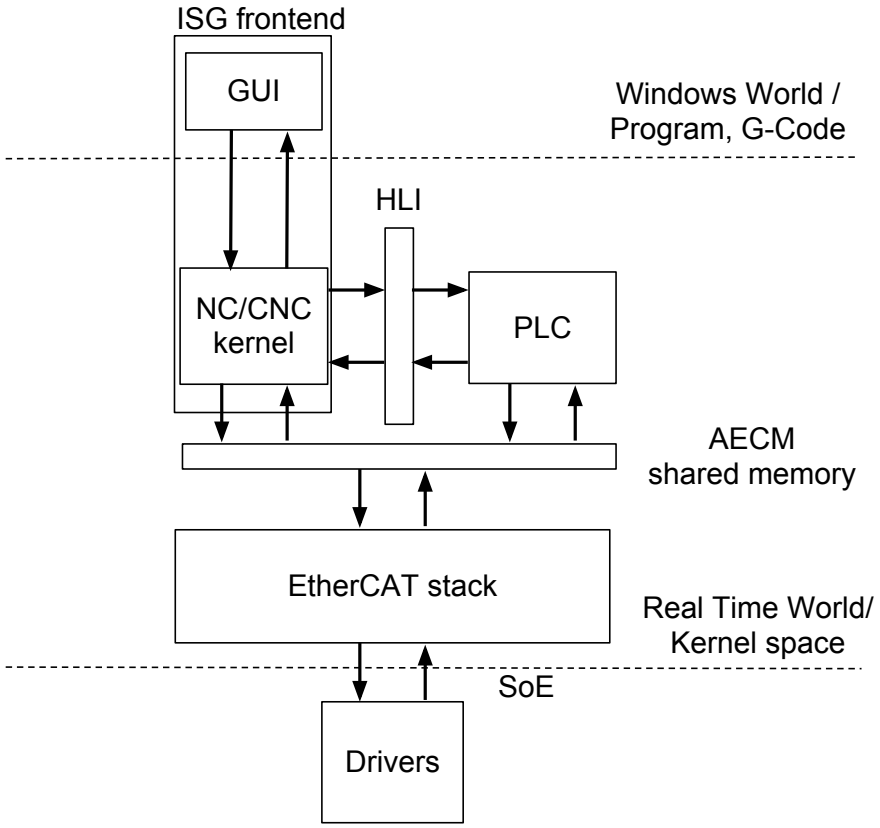

Figure 4.1: Overview of the ISG kernel and how it is connected to PLC and drives.

dinates as it is usually called, is simple to understand, as a constant per axis that represents the conversion between rotational to linear movement. In the case of the F1 robot this transformation is more complex, see Section 3.1.1, and implemented as a realtime DLL used by the ISG-kernel.

HLI memory

To extend the usability of the kernel and the the software, ISG has added a shared memory, the HLI, High Level Interface [17]. This interface is used to create an interface to a PLC, Programmable Logic Controller. A PLC is traditionally a stand alone computer using a programming language like Ladder logic or Sequential Function Chart.

The HLI has been very important in the development of the server and client in this thesis, since it enables the access to many of the internal variables of the NC-kernel. The interface is sectioned logically in similar way as the kernel itself is sectioned. The interpolator and path planning can be accessed through the

interface. This is what has been used to implement the external control of the axes.

External Variables

The HLI is not the only way of sending or receiving data to or from the kernel. There is also a possibility to use the External Variables feature of the kernel [16]. These variables are defined by an ASCII text file in the ISG-NC folder. The external variables could be used with a GUI or another application to exchange information.

The kernel needs a way to communicate with the drivers of the machine. In the case of the F1 robot this is done with the help of an external EtherCAT stack developed by Beckhoff. The communication between the kernel and the stack is done with the aid of another shared memory, the AECM shared memory. This memory has to be mapped by the kernel to ensure that the correct data is input and output.

4.1.2 PLC

PLC stands for Programmable Logic Controller, and is a digital computer often used in automation or electromechanical processes. One difference between a common PC and a hardware PLC is that a PLC is designed for a large number of in/outputs, sometimes directly coupled to the CPU. Often the PLC-hardware is also designed for a more demanding physical environment than a common PC with respect to for example vibrations, higher temperatures and electrical immunity. A PLC is a hard realtime system.

One international standard for PLC's is the IEC61131 standard, where the third part, IEC61131-3, regulates the common languages used in a PLC. There are five languages in the standard: Ladder logic diagram, Function block diagram, Structured text, Instruction list and Sequential function chart.

Of these languages two are textual: Instruction list and Structured text. Instruction list is a low level language that resembles assembly programming. Structured text is high level language that resembles Pascal. It is the language used in this thesis for testing.

The PLC can be coded to perform many tasks, for example tool changing. The software PLC of the F1 takes care of the homing of the robot. It is coded in an IDE, integrated developing environment, called CoDeSys, and was used during investigation and evaluation of the type of communication needed. In this thesis the implementation of external communication with the ISG-NC kernel could be viewed as a software PLC.

4.1.3 RTX

RTX is a realtime extension for Windows. RTX is implemented as a realtime subsystem, RTSS, a Windows HAL, hardware abstraction layer, extension and some libraries. RTSS runs in parallel with Windows. Therefore it is possible to program applications that are independent from Windows. RTX provides realtime scheduling of threads and inter-process communication and synchronization. It is also possible to share data between Windows and RTSS applications [9] [18]. The ISG kernel is running in the RTX environment as well as the server of the external communication of F1. In this thesis the implementation of external communication with the ISG-NC kernel PC could be viewed as a PLC.

4.1.4 Code verification tools

When safe code is one of the design priorities of a program some kind of testing and verification of the code is necessary. While writing a program it is not only important for the code to be working and being well commented but also to be simple to understand. Even a short program can be hard to interpret and a short example is taken from the international obfuscated `C` code contest [19]:

```
main() { printf(&unix["\021%six\012\0"],
         (unix)["have"]+"fun"-0x60);}
```

The writing of better code can be aided in several ways. Some ways could involve additional programs to define test cases and verifying the operation of the program. Testing of the program could also be made by analyzing the code before the compilation by another program. Another possibility is to analyze the behavior of the program by running it in a controlled environment and analyzing the reading and writing of the memory used by the program.

4.1.5 Splint

Splint is an acronym for Secure Programming Lint. Splint is a free GNU licensed static error checking tool used to check for security vulnerabilities and general coding mistakes [10].

The program was used to aid error-checking and enforce good programming practice when developing the `C`-programs used in the thesis.

The tool is a static analyzer, meaning that it does not perform any analysis on a running program. It is capable of checking for a large number of potential errors, such as `NULL` references, accessing undefined values or uninitialized variables and

extended type checking. Splint also has the ability to check for memory management errors. This could also be extended by using dynamic checking using a tool like Valgrind.

Splint was used to check the code of the server, to increase the readability of the code and reduce the risk of coding errors.

To demonstrate the functionality of the programs splint was used to analyze a short RPN calculator program. Output and comments can be found in Appendix A.

4.1.6 Valgrind

Valgrind is free GPL program used to dynamically detect memory errors [11]. It was originally used to detect memory errors on x86 Linux systems but has evolved to a complete tool-chain for many platforms.

In contrast to a static tool, the analysis is performed at run time, this is both to its advantage and disadvantage. A static tool could try to interpret the programmers intentions while a dynamic analysis can check for different kind of errors, as well as performing other types of analysis. For example is Valgrind capable of detecting race-conditions, analyze heap usage and cache analysis amongst other things.

One of the large drawbacks is the difficulty of finding the relevant test cases. The process of finding all possible inputs in most real applications is impossible.

In this thesis it was used to detect memory errors in the server and client but the tool can be used for many other purposes.

4.1.7 Beckhoff Drivers

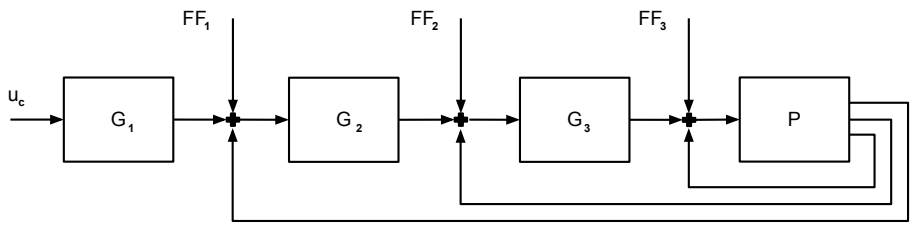

Figure 4.2: Common controller structure for industrial motor controllers.

A common control structure for position control of motors can be seen Figure 4.2. The controllers are connected in series and the structure is often called cascaded. Often the controllers are standard PI controllers, and the feed forward the derivative of the reference.

Figure 4.3: Beckhoff AX2000.

Figure 4.4: Beckhoff AX5000.

To be able to control the current of a large AC-motor some kind of amplifier has to be used. Modern motor drivers are mostly digital PWM-controlled amplifiers with a digital interface. The drivers will also in most robotic applications need to be fed with realtime data through a hard realtime connection, in this thesis all Beckhoff drivers use EtherCAT. The list of industrial communication protocols can be made very long.

The configuration of the drivers are made in TwinCAT. More about TwinCAT can be found in Section 4.2.8. The drivers considered in this part of the project will be the Beckhoff AX2000 and AX5000. The AX2000 divers are currently controlling the F1 robot.

Beckhoff AX2000

The AX2000 drivers, seen in Figure 4.3, can set speed, position or torque references. It is possible to add external references to the torque only [2] [1]. The position, velocity and current controller unit are PI controllers [3]. In the AX2000 there is an internal velocity feed forward gain, but no internal torque feed forward, an overview of the controller structure can be seen in Figure 4.5.

The drivers can use many different communication protocols, by installing add-in cards with different hardware. The drivers for the F1 use the Beckhoff AX2000-B110 EtherCAT interface card, to connect the drivers to the PC running the ISG kernel.

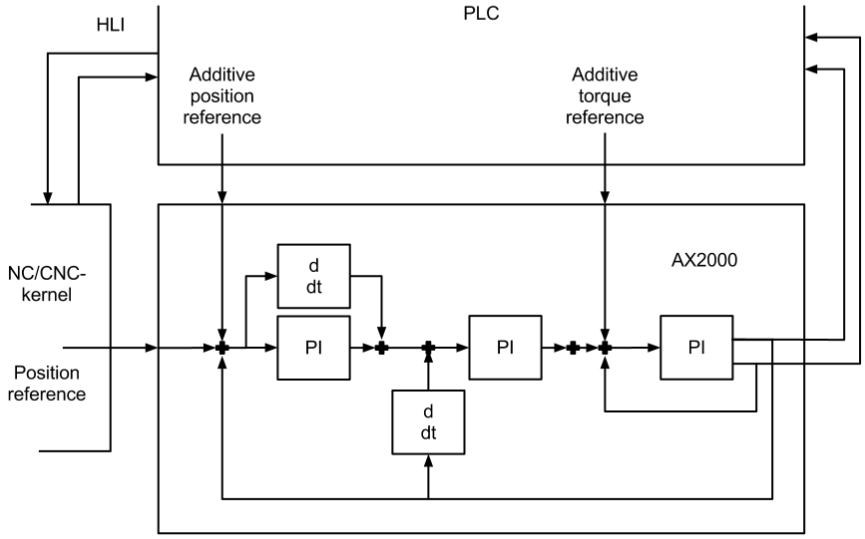

Figure 4.5: AX2000 driver internal structure.

Beckhoff AX5000

The references sent to the AX5000 drives, seen in Figure 4.4, can be speed or positions references, with the possibility to add position, velocity and torque as external references [20], the controller structure can be seen in Figure 4.6. The position controller unit is a P controller, while the velocity and current controller units are PI. The configuration of the drivers is done through TwinCAT 2 [4]. The AX5000 has an internal feed forward for the velocity and torque loop in the drivers. These drivers have an EtherCAT interface which is used to receive hard realtime data.

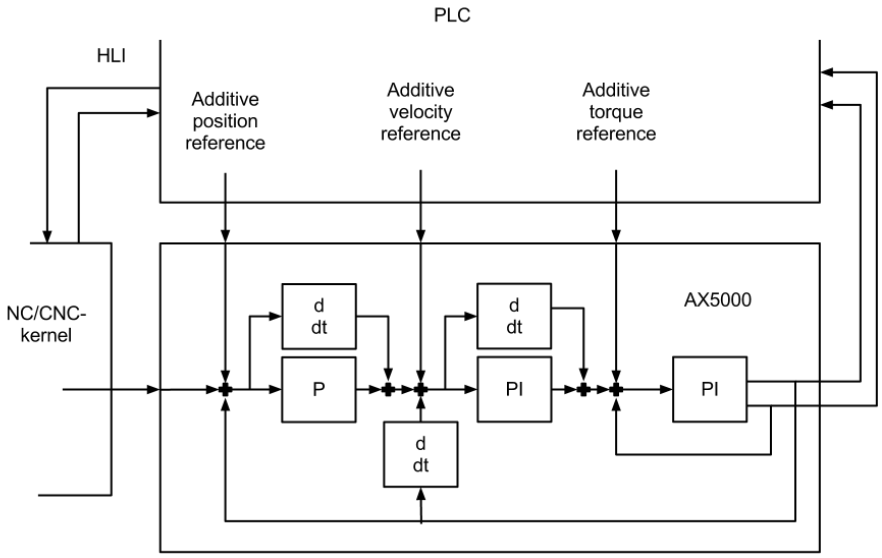

Figure 4.6: AX5000 driver internal structure.

4.2 Stiffness measurement of the L2 PKM

The system setup consists of the PKM robot itself, the ABB IRC5 controller and a force/torque sensor mounted on the TCP. The ABB controller is connected to an ExtCtrl PC with the possibility to log and add references. An overview can be seen in Figure 4.7. To be able to measure the displacement of the carts and the TCP a measurement PC handling the Heidenhain gauges has to be connected to the already existing set up.

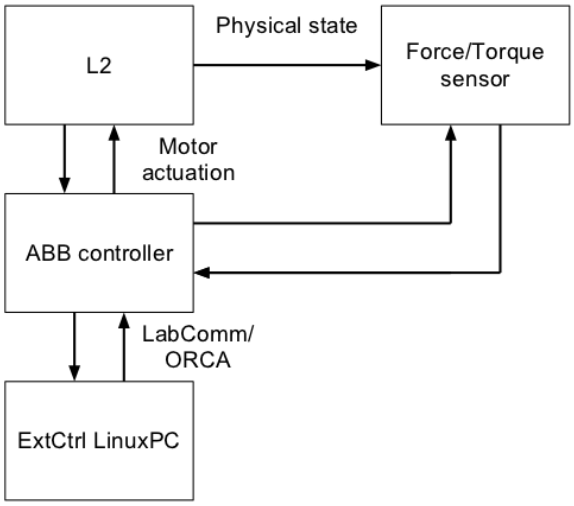

Figure 4.7: L2 initial setup.

4.2.1 ABB IRC5 controller

The drive system is developed by ABB and it is the fifth generation of ABB robotic controller. The driver is more than just a single motor driver and also contains a path-planner and is able to execute RAPID code.

Inside the cabinet there is a central computer calculating the paths of the tool. The axes are equipped with a computer controlling the motor, the position, and velocity references. It is between this link between the central computer and the axis computer that the ExtCtrl “cuts” to extract references and actual values [5].

4.2.2 ExtCtrl

The ExtCtrl software consists of a ABB controller and a computer running a "ExtCtrl server" with a client connected. The program that is to be run with the

software is a Matlab Simulink program compiled with the Real Time Workshop, then uploaded to the computer connected to the cabinet. This enables the system to be connected to any type of sensor or external measuring equipment, and interfaced with the control cabinet.

The client connects via a Python interface and can enable and disable the external control software.

ExtCtrl was developed within the Robotics Lab, LTH [6].

4.2.3 LabComm

LabComm is a binary protocol suitable for transmission of samples of process data [12]. It is designed to keep the communication overhead at a minimum, and is capable of one and two way communication. The protocol is designed to take care of the order of data and little or big endian problems. The usage of LabComm is simple, the specification and identifier is placed in a shared file, for example the file in this thesis:

```
sample int data[3]
sample float forcesensor[6]
```

LabComm was used to communicate both with the ExtCtrl computer and the external measuring computer.

4.2.4 OpCom

The OpCom software is written in Python and is used to control the loading and unloading of the program generated by Simulink to the ExtCtrl computer, a screenshoot of the interface can be seen in Figure 4.8. The program connects to another PC in the lab that is connected to the IRC5 controller via a RS232 connection.

After connecting to the control cabinet OpCom is used to load the program to the realtime ExtCtrl PC.

Once the program is loaded onto the computer the controller can be set to a "Submit" state, seen in Figure 4.9. In this state data is sent to the computer but not back to the controller. This can be very useful since the program can be tested and debugged to ensure that the program is safe and will not harm the robot itself or a person in the vicinity of the robot.

When the program is tested and appropriate measures are made to ensure the program is safe, the "Obtain" state can be entered.

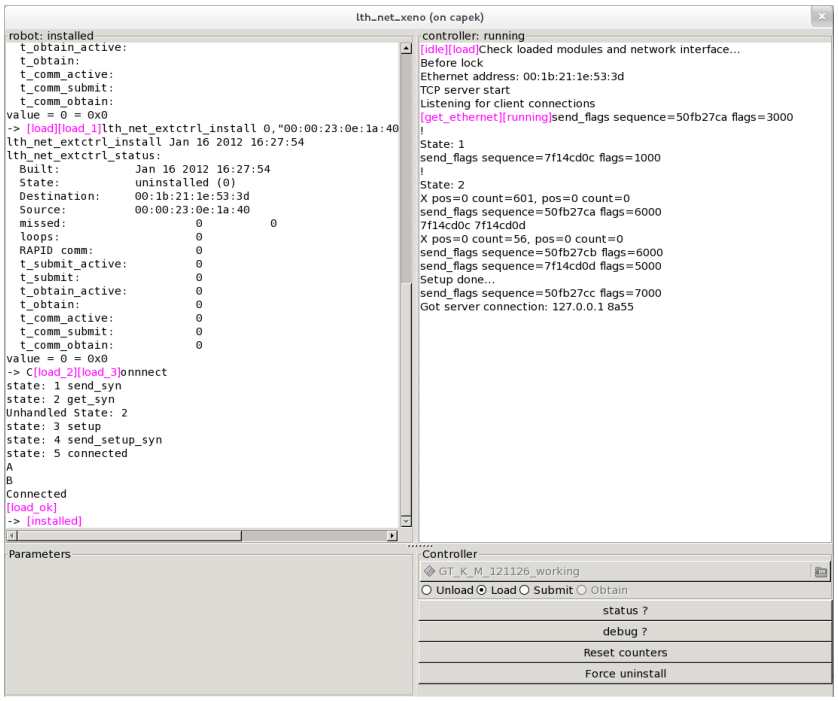

Figure 4.8: Screen shot of OpCom.

4.2.5 ATI Mini85 Force and Torque sensor

The force sensor connected to the end effector of the robot is connected directly to the IRC5 controller. The sensor, seen in Figure 4.10, is connected to the 5-DOF extension of the robot and uses its own coordinate system [13]. Therefore the force and torque must be transformed into the global frame before any calculations can be made, as well as filtering of the noisy signal. This is done in the Simulink program.

This force sensor is a 6-DOF sensor, meaning it can sense Cartesian force and the torque of each axis. This means there is no way of separating the force due to acceleration and deceleration. If this would be needed it would have to be included in the model or a 12-DOF sensor would have to be used which can separate the external force and the force due to acceleration.

4.2.6 Heidenhain ST3078

The Heidenhain ST3000 TTLx10 is a high-precision length-gauge that can be used to measure short distances. The gauges can be seen in Figure 4.11. The resolution of the gauges is $0.5\mu m$. The length is measured at the tip and a spring ensures that it is extended. Three Heidenhain gauges were used to measure dis-

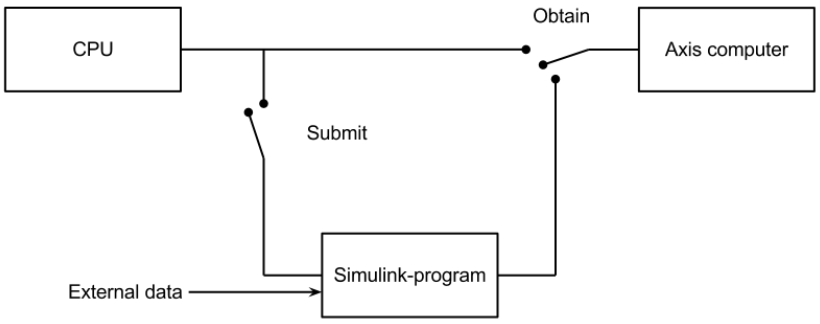

Figure 4.9: OpCom functionality.

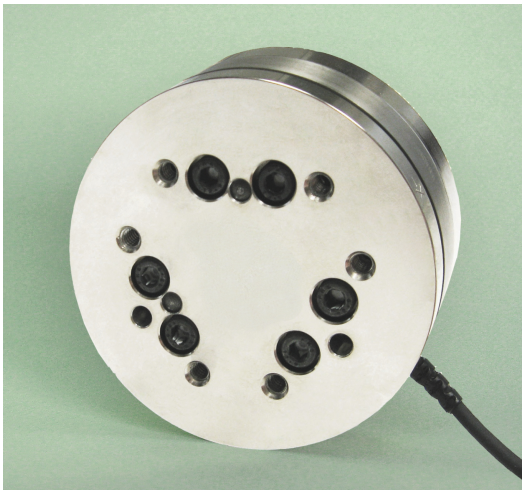

Figure 4.10: Mini85 Force and torque sensor.

placement in three dimensions. Each gauge was connected to a I/O Beckhoff terminal. To be able to use the gauges in three dimensions a holder for the gauges was machined.

4.2.7 Beckhoff terminals

Beckhoff has developed a modular I/O terminal system. The system is modular in the way that a large variety of bus couplers can be used with a large number of I/O modules. In our work we used the Beckhoff EK 1100 terminal, which has an EtherCAT interface with an EL5101 incremental encoder interface, the terminals can be seen in Figure 4.12

Figure 4.11: Heidenhain ST 3000 length-gauge.

4.2.8 TwinCAT

TwinCAT is a Beckhoff software tool to enable the use of a PC as EtherCAT master. TwinCAT is capable of using a long list of both realtime and non realtime communication protocols. In this thesis TwinCAT is using the EtherCAT protocol to communicate with both drivers and IO-units. TwinCAT is guaranteed to be a real-time application since it runs under RTX-environment in Windows.

TwinCAT 2

TwinCAT 2 is the standard version of TwinCAT used in the industry. The

Figure 4.12: EtherCAT terminal and incremental interface.

software is mature and has been fairly well documented [14]. Since the software has been in development since 1996, many add-on modules are available. Most notable from a robotics point of view is the CNC/NC modules.

The modules are capable of interpreting G-code, do path planning and have the capability to use kinematic transformation. The transformations are standard kinematic transformations for common CNC-machines and standard industrial robots. Custom kinematics is not available.

The CNC software itself is not made by Beckhoff, but by the ISG-NC kernel. Because of this the features of TwinCAT CNC and the ISG-NC are basically the same. The GUI has been remade by Beckhoff and does not look anything like the ISG GUI.

TwinCAT was used to connect the Heidenhain length gauges to the measuring PC enabling the values to be read and sent to the ExtCtrl computer via a LabComm connection.

4.2.9 L2 measuring setup

The spring constant of the carts were measured with the Heidenhain length gauges. The gauges were fastened on the rails with a clamp as seen in Figure 4.13.

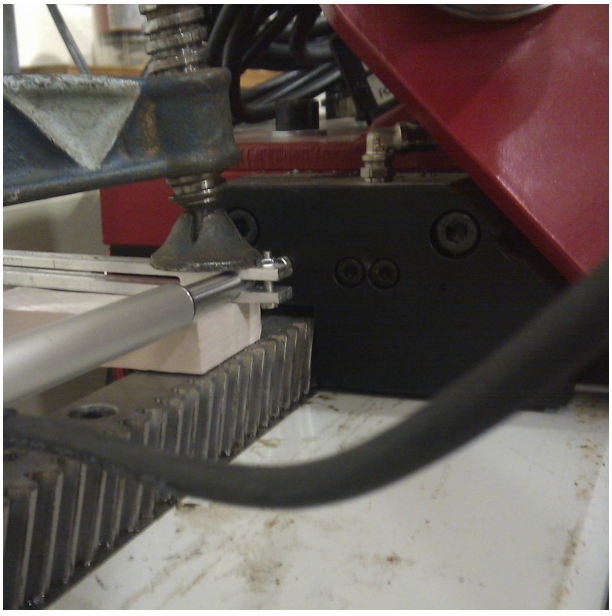

Figure 4.13: L2 cart measurements.

To apply a force to the end effector a ratchet tie down was attached to the force sensor. The force measure by the end of the force sensor was then translated to the

center of each ball joint on the end effector and the force placed on the carts was obtained as described in Section 3.3. By using three gauges the measurements could be made on all three carts simultaneously.

During the measurement the force was applied in steps. The robot was given several seconds of settling time to assure no dynamics were taken into account. In every measurement step values were recorded for two seconds. To remove noise and reduce the risk of recording unwanted dynamics when determining the spring constant the mean value was used.

The measurements made of the link system was conducted in a similar way as the measurements made on the carts. A holder for the gauges was milled to be able to measure the Cartesian displacement. A large block of aluminum was fastened to the link system end effector to provide flat surfaces for the gauges, see the setup in Figure 4.14.

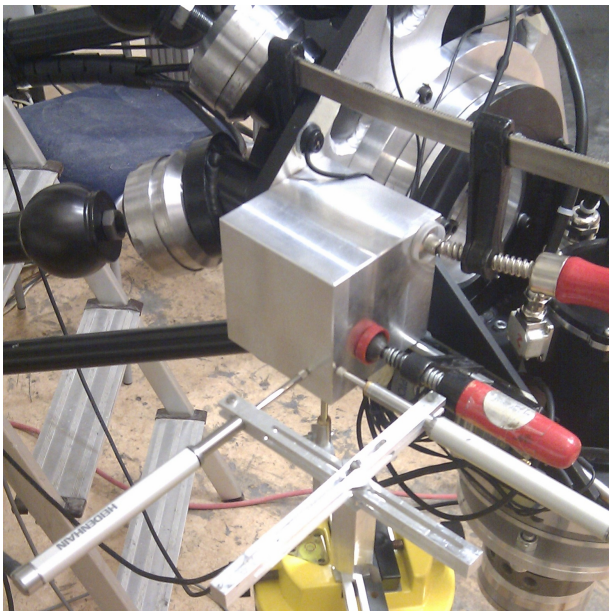

Figure 4.14: L2 3-DOF measurements.

While performing these measurements the compensation for the cart stiffness was active, therefore the movement of the carts was assumed to be negligible. The force was applied as several steps and the mean was used in the calculations which were done in Matlab.

4.2.10 F1 measuring setup

The measurements of the carts were obtained using a Heidenhain gauge together with the Beckhoff EK 1100 and using TwinCAT 2. The gauge was attached to the

robot with a magnetic foot stand. A ratchet tie was fastened horizontally between two of the frame legs. An additional ratchet tie was strapped vertically around one of the carts and attached to the scale which was fixed to the horizontal ratchet tie. When the vertical ratchet tie attached to the cart was fastened a downward force was placed by tightening the vertical ratchet tie, the resulting force and displacement was recorded. The data was collected with TwinCAT and saved as ASCII files. The files were then parsed and plotted in a MATLAB script.

5 Results

External communication for the F1 PKM was implemented and briefly tested on the the double motor cart and the rotating plate of cart 2, this can be found in Section 5.1.

Sections 5.2 and 5.3 describe how the stiffness measurements were set up and how the resulting spring constants were calculated. In Section 5.2.3 the implementation of the compensation is described and the resulting performance is shown.

5.1 Implementation of external communication

The external commanding of an axis was implemented by using a realtime TCP/IP connection with a rate of 1kHz. The resulting control of the axes was logged and can be seen in Figure 5.1 and 5.2.

An overview of the implemented software can be seen in Figure 5.3. The server was implemented as a "soft PLC" and the external source could be any device capable of realtime TCP/IP communication.

5.1.1 Server

The software on the realtime PC controlling the F1 was extended by a program accessing the ISG-NC kernel shared memory. This enabled external control of each axis controlled by the kernel. The program was written in `C` during the visit to Fraunhofer IPA, Stuttgart.

The server must be able to handle the communication with clients in such a manner that the operation of the robot is safe, both for the operator and the hardware. In normal operation of the ISG-NC kernel this is thoroughly checked, but this check is not made on any external references. Checks if the set point are eligible are not made in the server and is completely left to the connecting application.

If, for some reason, the communication is disrupted, a step change of the reference could occur. To prevent this the reference slowly returns to the uncompensated value at a speed of $0.1\mu m$ per millisecond before accepting new connections. If a package is lost the server will allow one package to be lost before entering the "safe disconnect" state. A simple graphical representation can be seen in Figure 5.4.

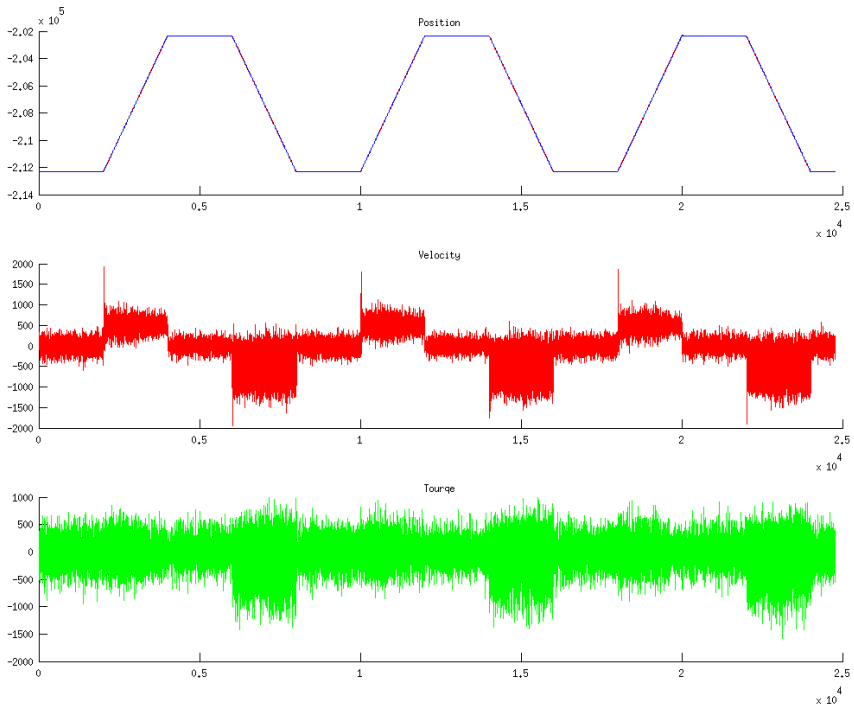

Figure 5.1: Position, velocity and torque for the ball screw on cart 2. The position reference and the actual position is hard to tell apart since the position error is small.

5.1.2 Client

The client was implemented both as one Unix and one Windows version. The main difference between the Windows and the Linux client is that the Windows implementation use winsock2 as its communication API. The client was intended to measure the force of the end effector and compensate for the compliance of the robot. The software was used to verify the external reference setting and log the references and the actual values of the control system.

A client communicating with the server running on the realtime PC can be implemented on any system capable of realtime TCP/IP.

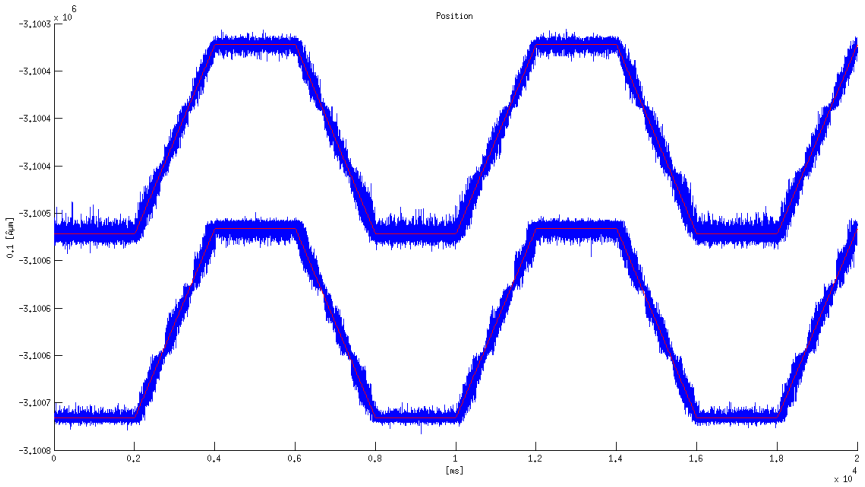

Figure 5.2: The position of the double motor on cart 2 and the reference sent to the motors.

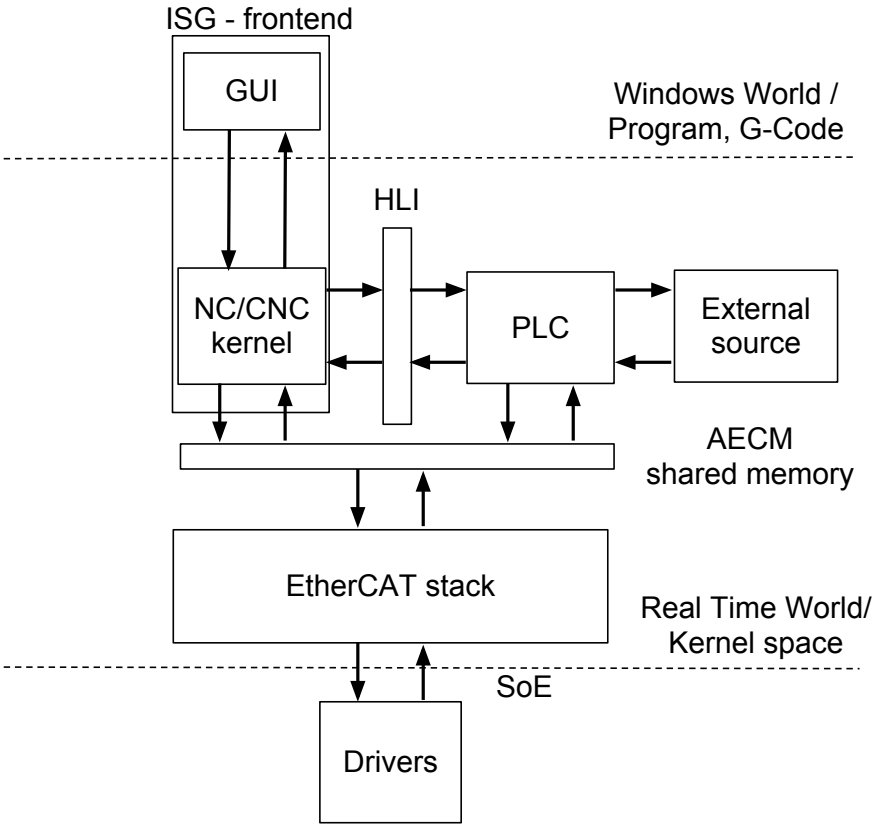

Figure 5.3: Communication with external source.

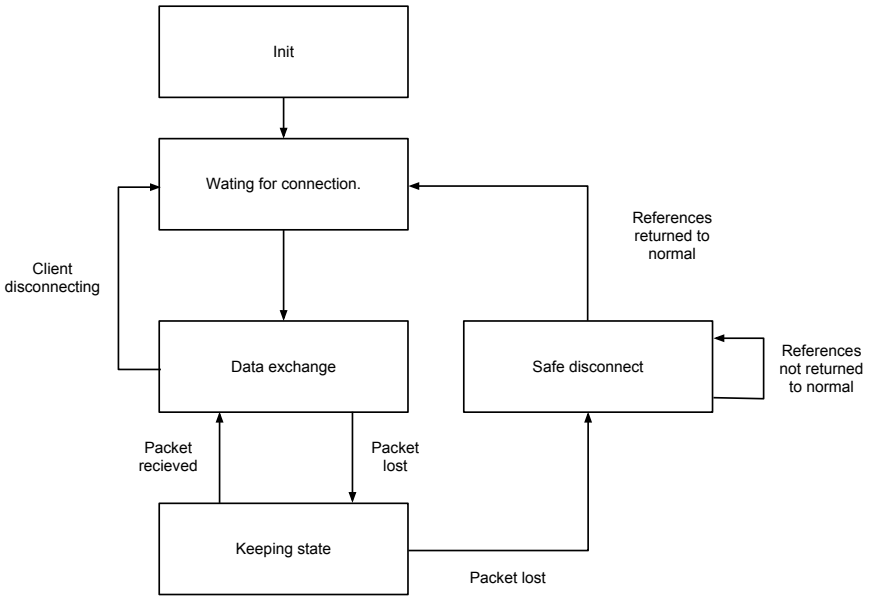

Figure 5.4: The states of the server.

5.2 Stiffness measurements and compensation L2

The stiffness of the L2 PKM was measured, both for the carts relative to their tracks and the stiffness of the link system. The main tool for measuring the displacement was the Heidenhain gauges.

The Heidenhain gauges values needed to be accessible in ExtCtrl. The gauges were connected to the Beckhoff terminals, which sent the values with EtherCAT to a PC. The EtherCAT network was a separate network with only the PC and the terminals as connected devices. The data was sent to the the ExtCtrl system with a LabComm connection, a overview of the system can be seen in Figure 5.5. The program was implemented in `C` under Windows XP and TwinCAT 2.

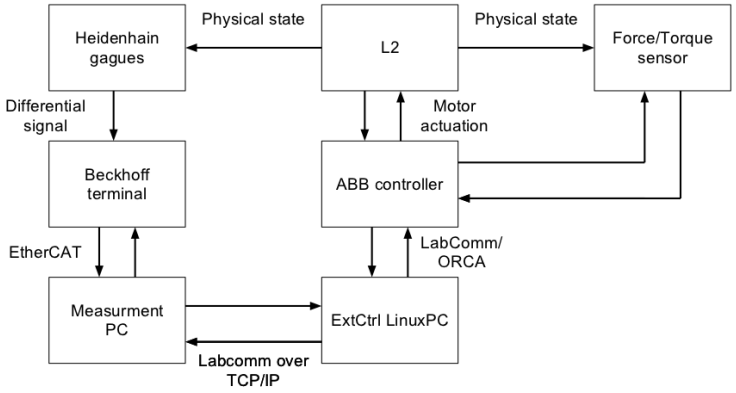

Figure 5.5: L2 with extended measuring equipment.

5.2.1 Carts

The estimated spring constants of the carts can be seen in Table 5.1 and the measured and the estimated spring constant for cart 2 can be seen in Figure 5.6.

Cart nr.	Spring constant
cart 1	0.0406 $\mu m/N$
cart 2	0.0509 $\mu m/N$
cart 3	0.0600 $\mu m/N$

Table 5.1: Spring constants of carts in relation to their rails.

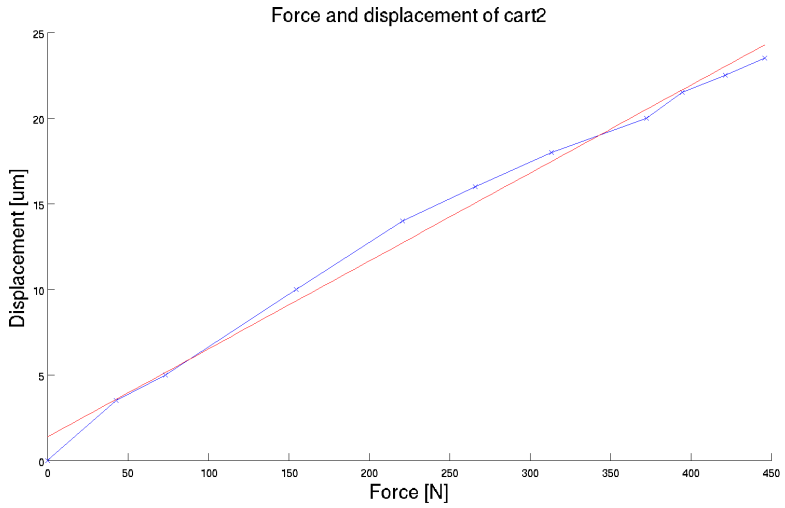

Figure 5.6: Estimated and measured spring constant for cart 2 of L2.

5.2.2 Links

The spring constant for the links can be seen in Table 5.2, and the measurement points and estimated spring constants for arm 21 in Figure 5.7. Both in Figure 5.8 and in Figure 5.7 one can see a non linearity, it appears in the experiments at approximately 360N in the x axis. This will be further discussed in Section 6.1 The spring constants used in the compensation is the second row on Table 5.2.

Link nr.	Spring constant, used	Spring constant
Link 11	-0.6351 $\mu m/N$	-0.3271 $\mu m/N$
Link 21	1.8024 $\mu m/N$	0.7700 $\mu m/N$
Link 22	-0.5799 $\mu m/N$	-0.2478 $\mu m/N$
Link 31	-12.0260 $\mu m/N$	-7.4292 $\mu m/N$
Link 32	8.7635 $\mu m/N$	5.3801 $\mu m/N$
Link 33	4.4191 $\mu m/N$	2.7638 $\mu m/N$

Table 5.2: Measured spring constants from links. The left column is before the non linearity and is used in the compensation. The right column is after the non linearity is passed.

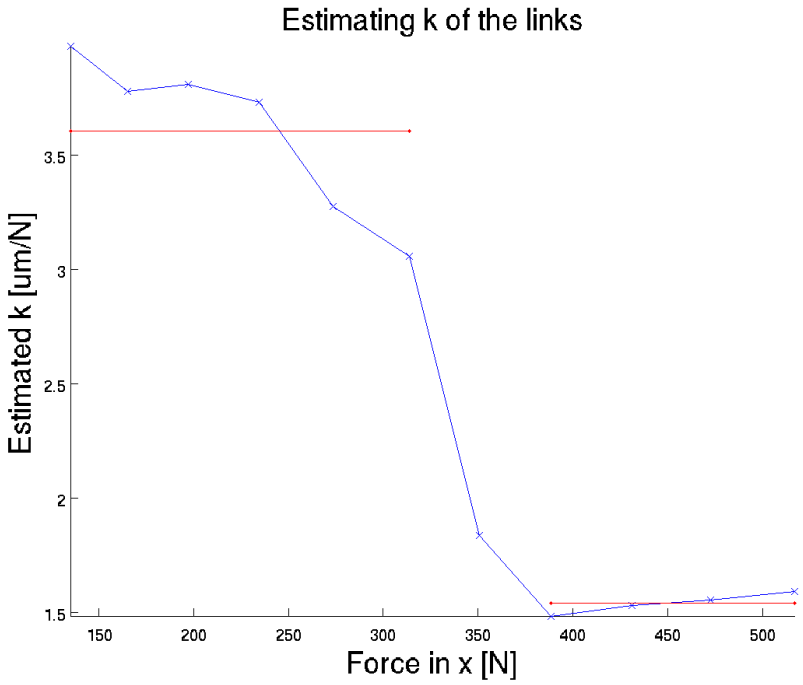

Figure 5.7: Estimated and measured spring constant on L2 of arm 21.

5.2.3 Compensation

Compensation was made by using the determined stiffness and spring constants and calculating the displacement of each part and simply adding the calculated displacement to the reference. The displacement of the carts was added directly to the reference, while the displacement of the arm system was transformed with the inverse kinematics to obtain the references of the carts.

The compensation was implemented in the Simulink program using ExtCtrl, see Figure 5.9.

In Figure 5.10 the displacement and force in the x-axis of the 3-DOF end effector can be seen. The displacement of the TCP behaves as expected, apart from the non linearity the the displacement can be seen to be proportional to the applied force.

In Figure 5.11 a force is applied with the compensation active and the displacement can again be seen. The displacement of the parallel end effector in the x direction is as intended lower. The compensation after the non linearity is not decreasing the compliance since the spring constant after the backlash is lower, see Figure 5.7. The non linearity can be seen in both figures, since it was never compensated for.

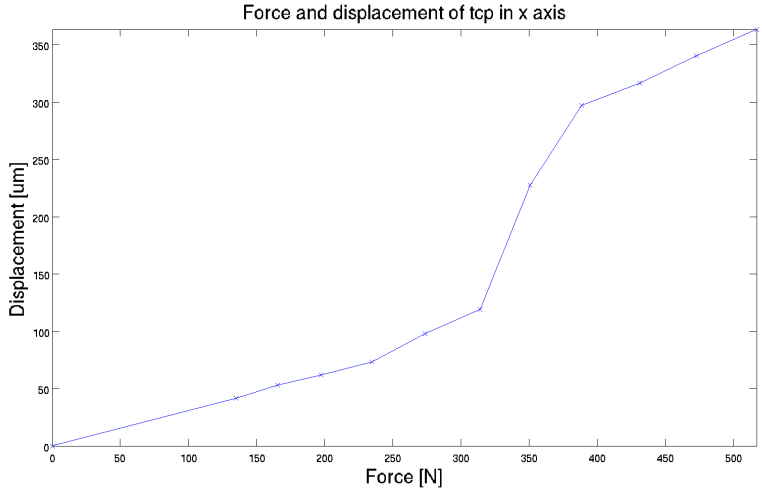

Figure 5.8: L2 TCP displacement due to applied force.

5.3 Stiffness measurements for F1

The stiffness measurements of the F1 robot are not complete enough to enable implementation of any useful compensation. The algorithms used with the L2 could be implemented for the F1 control system if more time was given.

The arms were not mounted during the stiffness measurement of the carts.

5.3.1 Carts

Data from measurement on the cart 2 can be seen in Figure 5.12 and the spring constant of each carts can be seen in Table 5.3.

cart 1	4.2081 $\mu m/N$
cart 2	7.5215 $\mu m/N$
cart 3	5.5578 $\mu m/N$

Table 5.3: Spring constants of carts relative their rails of the F1 robot.

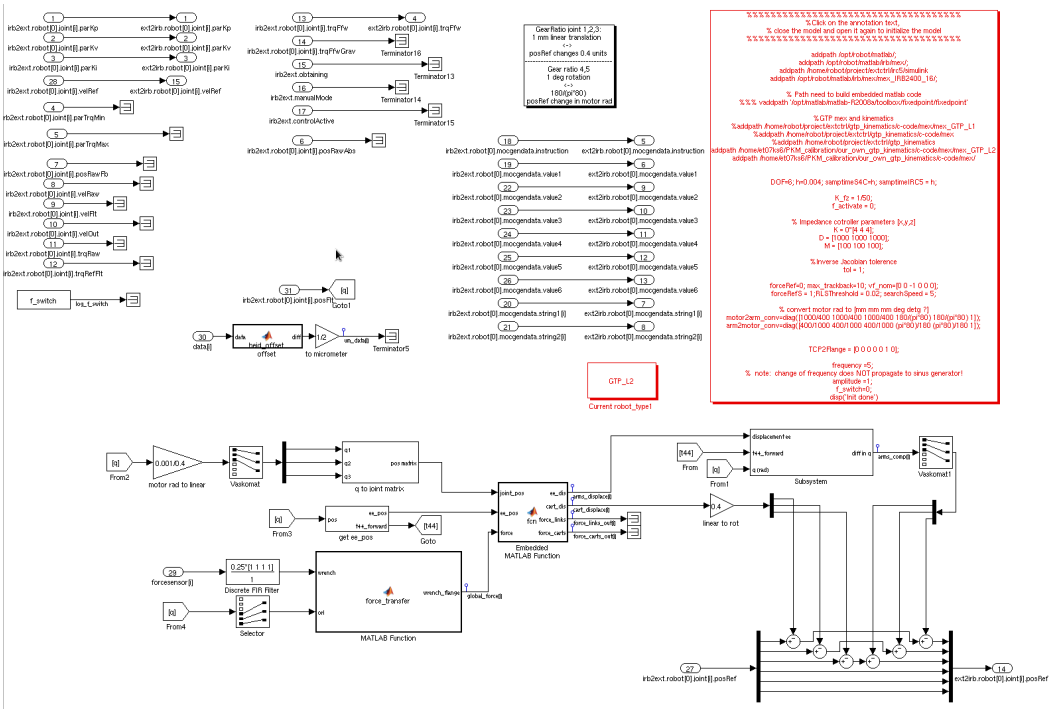

Figure 5.9: Compensation in Simulink.

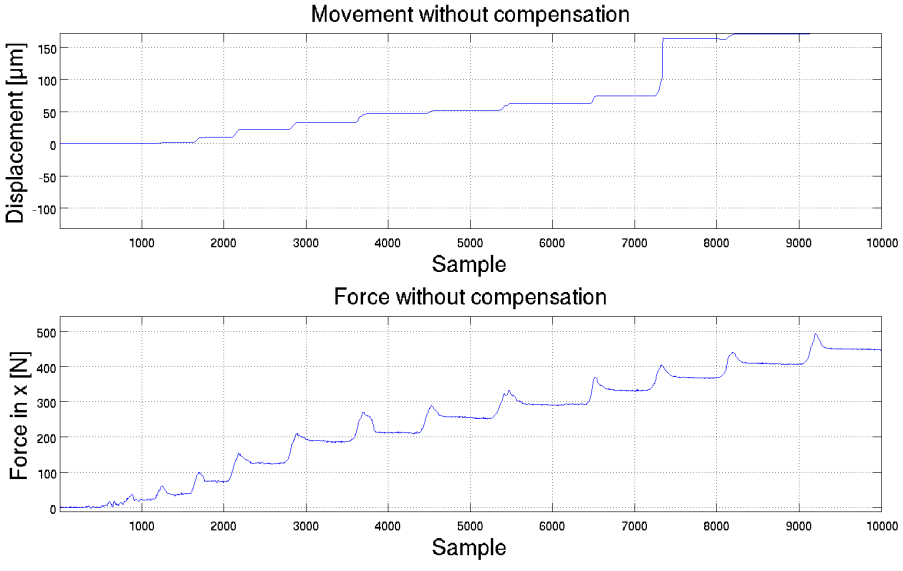

Figure 5.10: Displacement of L2 end effector in x without compensation.

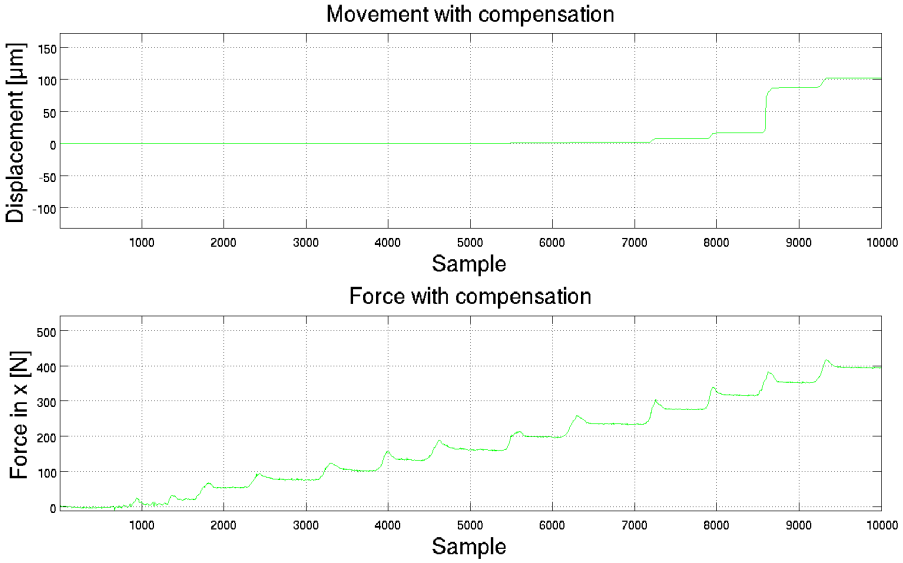

Figure 5.11: Displacement of L2 end effector in x with compensation.

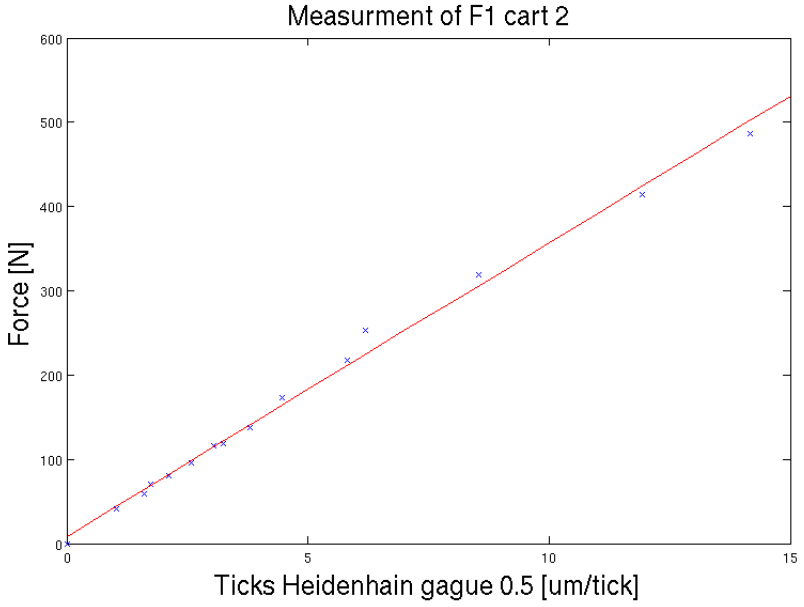

Figure 5.12: Force and displacement measurements of cart 2 F1.

6 Discussion and Conclusion

At the beginning of this master thesis we knew that more practical projects are time consuming, often to a larger degree than expected. Despite this we never thought that some parts of the project would take as much time as it did.

One of the largest parts of this project have been to understand the hardware and software and finding the best way or any way to make them work. Another erudition is how important it is to have the right licenses for the software one is using. The implementation of the external communication for the F1 could not be continued since the licenses were never acquired.

6.1 L2

The compliance of the carts compared to the compliance of the TCP was found to be considerably smaller. After implementing the compensation for the compliance the performance was considerably improved, see Section 5.2.3.

The non-linearity of the displacement was very carefully measured and confirmed not to be a measuring error. The "jump" is large enough to be felt by touching the end effector and applying a force to it. The explanation to the unexpected non-linearity is that the end effector is "resting" on the third cart. When a large force is exerted by the link system, the sign of the total force of the links attached to the third cart changes, in this case the force in the global positive z direction.

Since the weight of the link system and the end effector is unknown the expected resting force is unknown. The non-linearity was never compensated for since this could introduce significant mechanical wear.

All measurements were done without taking any dynamics into account. One way to improve the model would be include the dynamics of this as well. Depending on the application this could improve the performance and in some cases it may be necessary.

When the measurements on L2 were made, the placement of the Heidenhain gauges may have affected the result. While measuring the displacement of the carts the initial measuring position was on the back of the cart. The values obtained were discarded because after changing the measuring point to the point seen in Figure 4.13 the new values were more linear and closer to what was expected to find.

During the measurements of the end effector displacement the placement of the gauges will affect the result to a larger degree since the rotation is not taken into account. The assumption that there is no rotation of the end effector is proba-

bly not true, and by changing the point measured on the aluminum block, as seen in Figure 4.14, the measured displacement will change. One way of estimating the rotation would be to fasten two gauges on different places on the end effector and measure the rotation around one axis.

It is impossible to have negative spring constants in this kind of model. The lack of measured rotation could be the cause of the negative spring constants of the links. It was tested to set the rotation to a small value and the sign of the spring constants changed to positive. However the compensation is working with the measured spring constants, the reason for this may be that the measurements are made in one pose and with the gauges only connected to one point.

The measurements of the L2 could be extended and verified to a greater degree. More measurements could be made, for example extending the number of poses of the robot.

Adding the last two links would be necessary for any real application since all tools will be connected to the 5-DOF TCP.

The Jacobian could have been used to obtain the force on the carts but since the code for the translation and theory were developed within this thesis it was decided to not use the Jacobian.

6.2 F1

The possibilities for further improvement of the F1 robot are still large. External connection and the possibility to add external position references were implemented. This, however, is only a foundation to improve the control system of the F1.

The spring constant of the F1 carts were determined. This was done because the intention at the beginning of the thesis was to make compliance compensation also for the F1 robot.

6.2.1 External communication with F1 NC-kernel

It would be desirable to access the control variables of the F1 robot as well. This is unfortunately not possible since the controller is within the drivers.

The protocol used to communicate was TCP/IP. The reason to use TCP instead of UDP is to ensure that no packets are lost.

The use of TCP or any kind of IP protocol for the real-time part of the communication is questionable. A much simpler approach with raw Ethernet frames could have been used to minimize overhead, with the drawback of being forced to stay within the local sub-net and, possibly, a more static setup.

As a middle way UDP could be used together with good handling of packet loss.

One way of implementing the communication, in this case over TCP/IP, would have been through the PLC in CoDeSys. The reason this was not done was because the library for IP-communication is not guaranteed to be real-time.

6.2.2 Moving the position loops of the F1 control system

Currently the position loops are located within the drivers, but they could be moved to the ISG kernel PC and be written inside the PLC. To be able to move the position loops one has to send velocity references to the drivers. It is possible to configure the drivers through the EtherCAT Configurator so that one can send position, velocity and torque. It is, however, not possible to configure the ISG kernel to send anything other than position reference. By moving the position loops the possibilities to extend the control system is larger, controller values could be set during operation and it would be possible to set velocity feed forward with drivers normally not capable.

The position have not been moved due to two reasons. First, the lack of the software key to the ISG kernel. It was only possible to borrow this for a limited time from Fraunhofer IPA. Secondly it would have been desirable to test sending velocity and torque to the drivers on a separate driver and motor setup and not on the F1 robot directly because of the risk of damaging the equipment.

6.2.3 Control system for CNC-machines

Since the F1 robot is researched "from scratch" by LTH together with Güdel and Fraunhofer IPA, any control system could be used. The ISG-kernel was chosen because Fraunhofer IPA was familiar with the software. Another path with open software would be one way of continuing. The Linux-CNC is a free open source PC system that could be investigated in the future. It provides most of the features required to control the F1 robot. This would require an EtherCAT stack for Linux, if the same drivers had to be used.

6.2.4 Comparison of Beckhoff drivers

Both the AX5000 and the AX2000 drivers can be configured with TwinCAT 2. The Beckhoff AX5000 has more possibilities to add external references to the controller while the AX2000 is only able to add an additive torque reference. The AX5000 has internal feed forward for both velocity and torque loops, while the AX2000 only has the possibility to set gain of the torque feed forward.

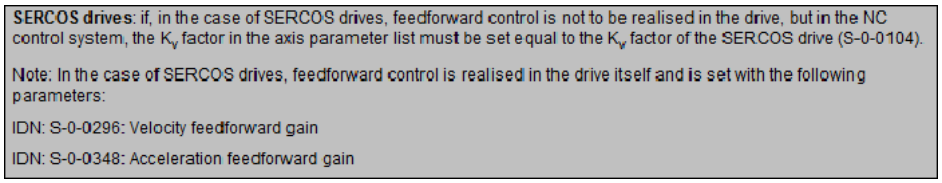

Figure 6.1: An example of the contradictions in the AX2000 documentation.

Feed forward is necessary if high performance is needed by the application. In some applications the possibility to add torque references could be very important.

One of the more time consuming problems with the Beckhoff drivers is the lack of documentation and the slow support. Beckhoff has grown exponentially the last years and the support has not kept up with the expansion of the company. There is no major difference in the quality of the documentation of the drivers. A small example of how inconsistent the documentation can be found in Figure 6.1 taken from the AX2000 documentation.

7 Future Work

7.1 L2

The two serial wrists on the robot are likely to have lower stiffness than the link system and carts. Compensation for the compliance of last two links would be required to benefit fully from the compensation implemented in this thesis.

The stiffness test has only been performed in the homing position of the robot. One should test the compensation implemented in the whole workspace of the robot. It would be interesting to do a milling test or any other practical test to see how well the compensation works.

7.2 F1

As mentioned before there is a lot of work to be done to improve this robot's compliance performance. The next step is to implement the position loops on a external computer with added feed forward.

It could be a good idea to test the additive velocity reference on an external setup of a driver and a motor to make sure not harming the F1 robot when running it with a velocity control loop.

TwinCAT 3 was released during the fall 2012. It could be a replacement for the ISG kernel. It has support for plugin programs in `C` and `C++`. Since the code written and compiled is run in the kernel space, there are still some limitations. The code will not be able to use any `C++` Run-time-DLL:s, or use code which handles `C++` exceptions. The `new()` operation, is not allowed since the creation of new objects may influence deterministic behavior.

Perhaps the most notable feature is the Matlab and Simulink interfaces added. TwinCAT 3 can run Matlab Simulink modules in real-time. The interface can also be used as a block in Simulink to interface with TwinCAT.

It has not been tested within the scope of this thesis but it could be further investigated.

7.3 Other related applications

The CAD files collected at Güdel will be used to a make a virtual model of the F1 and L2 robot in ABB RobotStudio. The model is intended to be coupled with the real robot to the virtual for running and simulations. The work has already begun, and will be continued by the staff at the department and other students.

Bibliography

[1] BECKHOFF. *Digital Servo amplifier AX2000*, 07 edition, 2007. Chapter: Setup 9.3.2.

[2] BECKHOFF Drive Technology. *Additional documentation for the AX2000-B110 servo drive EtherCAT interface for the AX2000*, 2.1 edition, October 2007. Chapter: Linking into the System Manager.

[3] BECKHOFF Drive Technology. *IDN Reference for AX2000*, 2.0 edition, October 2007. Chapter: IDN list.

[4] BECKHOFF Drive Technology. *Operating instructions Servo Drives AX5000*, 3.3 edition, July 2011. Chapter: Operating modes.

[5] Anders Blomdell, Isolde Dressler, Nilsson Klas, and Anders Robertsson. Flexible application development and high-preformance motion control based on external sensing and reconfiguration of ABB industrial robot controllers. In *Proc. ICRA 2010 Workshop on Innovative Robeot Control Archiecture for Demanding (Research) Aplications*, September 2005.

[6] Anders Blomdell, Isolde Dressler, Klas Nilsson, and Anders Robertsson. Flexible application development and high-performance motion control based on external sensing and reconfiguration of ABB industrial robot controllers. In *Proc. ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*, Anchorage, AK, June 2010.

[7] Isolde Dressler. *Modeling and Control of Stiff Robots for Flexible Manufacturing*. PhD thesis, Department of Automatic Control, Lund University, Sweden, September 2012.

[8] `http://www.isg-stuttgart.de/kernel.html?&L=1`, 2012.

[9] `http://www.intervalzero.com/rtx-faq`, 2012.

[10] `http://www.splint.org/`, 2012.

[11] `http://valgrind.org/`, 2012.

[12] `http://wiki.cs.lth.se/moin/LabComm`, 2012.

[13] `http://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini85`, 2012.

[14] `www.beckhoff.de/default.asp?twincat/default.htm`, 2012.

[15] `www.smerobot.org`, 2013.

[16] Industielle Steuerungstechnik GmbH. *External Variables within the NC*, January 2010.

[17] Industielle Steuerungstechnik GmbH. *Description of CNC / PLC -interface (HLI-manual)*, August 2012.

[18] IntervalZero. *Hard Real-Time with IntervalZero RTX on the Windows Platform*, January 2010.

[19] David Korn. `http://www.ioccc.org/1987/korn.c`, 1987.

[20] Rudolf W. Meier. AX5000 - motion control for high dynamic position. `http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCAQFjAA&url=ftp%3A%2F%2Fftp.beckhoff.com%2Fbelgium%2FAX5000%2Factuell_presentation_GB_AX5000.ppt&ei=bOJaUPDrMsrf4QSEz4DABQ&usg=AFQjCNEcgsGtDkue7POjEJO2KBr-KQjTuQ&sig2=IlqAeNju5-PRqjyYuk8zbA`, October 2007.

[21] H. Lipkin N. Ciblak. *Synthesis of Stiffnesses by Springs*, volume DETC'98, pages 1–10. School of Mechanical Engineering Georgia Institut of Technology Atlanta, September 1998.

[22] Harvey Lipkin Namik Cibiak. Assymetric cartesian stiffness for the modeling of compliant robotic systems. *Robotics, Kinematics, Dynamics and Control.*, 1994.

[23] Vidyasagar Spong, Hutchinson. *Robot Modeling and Control*. John Wiley and Sons, Inc., 2006.

A Splint Example

The original code was a programming assignment in the C-programming course at LTH. The code presented was the first task in the course. The resulting program is a classic rpn-calculator. When the code is compiled with gcc flags “-Wall” and “-pedantic” no errors or warnings are outputted by gcc.

```
1    if(!stackpointer)
```

Original rpn.c

```
rpn.c:11:9: Operand of ! is non-boolean (unsigned int): !stackpointer
The operand of a boolean operator is not a boolean. Use +ptrnegate
to allow ! to be used on pointers. (Use -boolops to inhibit warning)
```

```
    if(stackpointer != 0)
```

Corrected rpn.c

The warnings issued by splint is rather large number for the small amount of code. The first warning can be seen as a stylistic and readability warning. Some programmers may feel that using the boolean statement is unnecessary and ugly but using it increases readability.

```
        exit(1);
```

Original rpn.c

```
rpn.c:14:14: Argument to exit has implementation defined behavior:
1 The argument to exit should be 0, EXIT_SUCCESS or EXIT_FAILURE
(Use -exitarg to inhibit warning)
```

```
1       exit(EXIT_FAILURE);
```

Corrected rpn.c

The warning that is given by Splint speaks for itself, the argument should be defined by the defined macros EXIT_SUCCESS or EXIT_FAILURE.

```
                push(pop() + pop());
```

Original rpn.c

```
rpn.c:43:17: Return value (type int) ignored: push(pop() + pop())
Result returned by function call is not used. If this is intended,
can cast result to (void) to eliminate message. (Use -retvalint to
inhibit warning)
```

```
static void push(int number)
```

Corrected rpn.c

This warning shows a potential weakness in the program since the old code ignored any potential overflows. The warning was removed altering the function call pop(), by moving the error handling to the pop function and exiting the program in case of overflow.

```
    exit(0);
```

Original rpn.c

```
rpn.c:86:5: Unreachable code: exit(0) This code will never be reached
on any possible execution. (Use -unreachable to inhibit warning)
```

```

```

Corrected rpn.c

This warning may seem pointless, and may even be so in this cause, but in a more complex program this type of error detection could be very useful.

```
int main(int argc, char** argv)
```

Original rpn.c

```
rpn.c:30:14: Parameter argc not used A function parameter is not
used in the body of the function. If the argument is needed for type
compatibility or future plans, use /*@unused@*/ in the argument declaration.
(Use -paramuse to inhibit warning)
```

```
int main(/*@unused@*/int argc, /*@unused@*/char** argv)
```

Corrected rpn.c

The warning here is mostly self explanatory. This type of warning is a good sanity check for code since a forgotten argument may cause serous problems and an ignored one could case unnecessary confusion.

B Code

All code for the external communication, the commucation with the Heidenhan gauges and simulik models developed in this project can be found at LTH Control Department's svn repository https://www.control.lth.se/svn/M008-PKM.