CODEN:LUTEDX/(TEIE-5281)/1-68(2011)

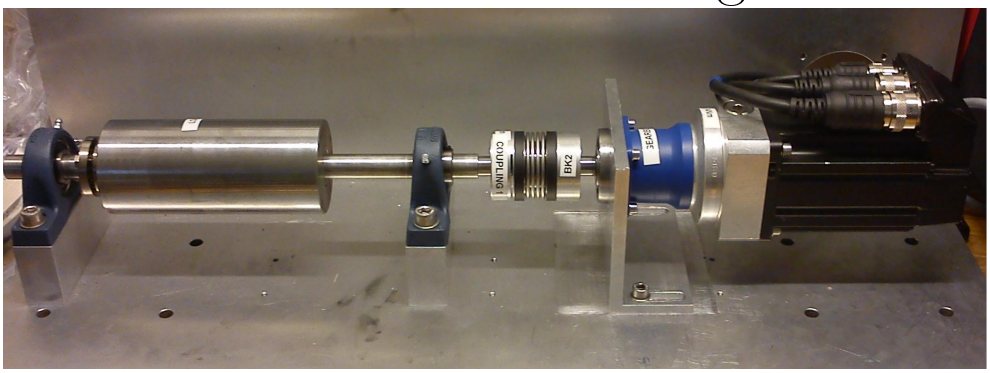
Industrial Electrical Engineering and Automation

Mechatronics Demo Rig

Björn Lineke

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Mechatronics Demonstration Rig



Master's Thesis

Björn Lineke
IEA, LTH
March 18, 2011

Abstract

This masters thesis presents a development of a software to analyse and evaluate a rotary mechatronic demonstration rig. Most of the hardware existed and the task was to develop software in Excel and PLC Ladder to detect differences between the different setups in a mechatronic system. The software is supposed to be used for educational purposes on Tetra Pak and should thereby be easy to use and give the user control over the necessary functions on the PLC.

With this software users should get a hands-on experience on how to configure and tune a mechatronic system. Also the user should have the opportunity to analyse data collected between different tests to see what has changed when changing the mechanics in the system.

A user has the option to change gearboxes, loads, couplings, shafts and the CAM-profiles the rig uses and save the data into an Excel workbook of choice. Once the values has been saved analyses can be done by different plots and frequency analysis and from here the user can make conclusions on the changes.

This report will explain the software and how to use it to detect differences between different setups and hopefully give a better understanding of mechatronic systems and how to adept them well.

The tests results presented in this report are just samples of what can be done with the software and how to use it.

I also would like to acknowledge all the help I have received from my mentor Thorbjörn Bengtsson at Tetra Pak, where he has been helping me out whenever I got stuck and keeping my work realistic when I was going crazy.

Contents

1 Background and Implementation

As more electronics is introduced in many mechanical systems when developing new machines and upgrading old equipment, the need for education and understanding of the new mechatronic systems is necessary. This rig is supposed to give developers a chance to get hands-on experience of how to tune and develop a mechatronic system without any danger of destroying or damaging expensive or advanced systems. Almost all hardware for the rig was ordered and ready when the project started and the task was to make software for the PLC and an interface in Excel to control the rig and analyse collected data.

To figure out what tests to run and how to analyse data from different these, the specifications for the hardware was analysed to see what properties that were interesting for each part and then make the necessary implementation in the Ladder programming for running a test. Once that was working the control should be made from Excel where the data acquisition and analyses are also handled with VBA macros.

Using VBA and Ladder programming gives a lot of room for further development as these languages a widely used on Tetra Pak.

The idea behind this demonstration rig is to give courses in mechatronic development on Tetra Pak. Where instead of just having theoretical education the people attending should also try out their own thoughts and ideas and get a hands-on experience of how everything interacts.

To achieve this, the interface should be easy to use and understand, contain tools for analyses and save data for comparison with different runs.

The rig is configurable with six different loads, two different bellow couplings, one elastomer coupling with three different properties, three different gearboxes and the ability to run it without gearbox. Later in the project new shafts where also added with different properties. The configuration is simple and it is possible add new equipment without any major changes, as part of the idea is that more functions are to be added in the future.

2 Presentation

2.1 Hardware

In this section all hardware and some of the parameters that affect the tests are presented. In figure 1 a general setup of the rig is shown. All parts can be changed and the rig contains six different loads, three couplings, three gearboxes, four shafts and one servo motor. Figure 1 shows how the rig is set up with the hardware.

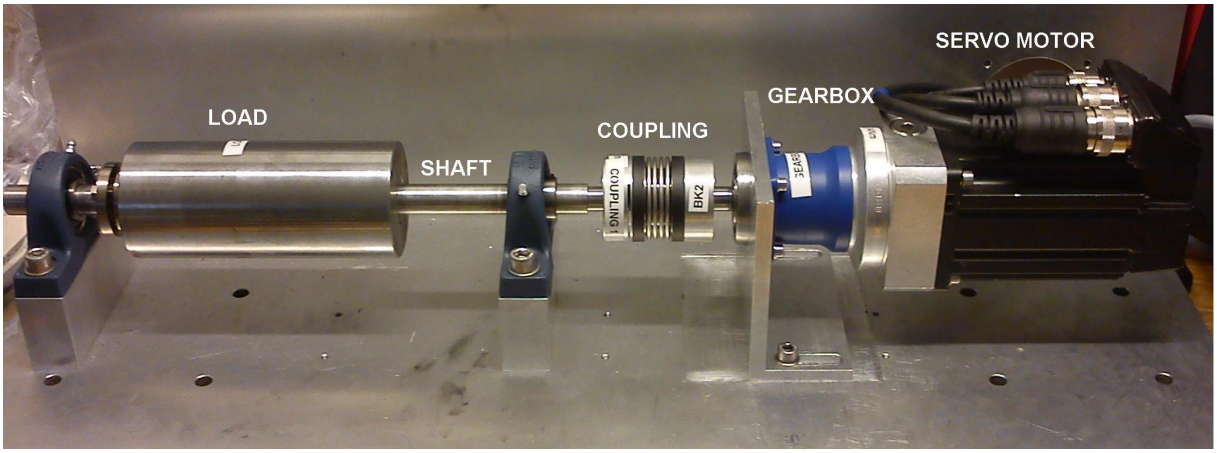


Figure 1: The Mechatronics Rig

2.1.1 Servo motor, MPL-310P

The servo motor used on this mechatronics rig is a Rockwell MPL-A310P brushless motor with a SinCos encoder. The drive and servo uses 230 VAC one phase and does not require a 400 VAC supply. This makes the rig easier to use wherever it is taken. Attached to the servo is a SinCos encoder which enables a very high accuracy of the current position. During one revolution the encoder generates 4096 sine and cosine waves and this is converted with an A/D converter with a resolution of 10-bits. This gives a resolution of 2 097 152 discrete points per revolution. Figure 2 is taken from the description by the manufacture SickStegmann and shows how the encoder works. This code track returns a series of digital position points as well as sine and cosine periods. With this the position of the servo motor is calculated.

The servo can generate a torque of 1.58 Nm continuously and peak torque of 3.61 Nm. The gearboxes with a ratio of 5:1 give some limitations on the torque using the EL coupling so it will not break. For these configurations the torque is limited to 75% continuously and 150% peak.

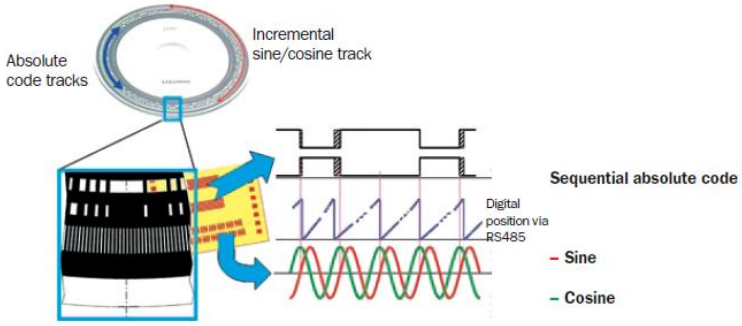


Figure 2: SinCos encoder taken from Description of the Hiperface by SickStegmann

2.1.2 Electrical cabinet

The electrical cabinet for the mechatronics demo rig holds all the electronics. Connected to the backplane of the system is the CPU, a network adapter and a SERCOS (SErial Real-time COmmunication System) card. The system's IO ports are connected to the ethernet card and managed by a Ethernet/IP (Industrial Protocol) bus, while the servo drive is connected with a SERCOS interface. The SERCOS interface is an optical communication bus and a standard used for motion control.

To the servo three cables from the electrical cabinet are connected, break, power and feedback. The mechanical brake is directly connected to the 24 volt power supply and is disengaged as soon as the electrical cabinet is powered up. The power comes from the Servo Drive and controls the current and frequency to the servo and the feedback is from the SinCos encoder.

2.1.3 Clamping element

The clamping elements are used to fasten the loads to the shafts and only holds the load on side. This gives an option to fasten the load close to the servo or, to get a less stiff configuration, far from the servo. The inertia of the clamping element is $65 * 10^{-6} kgm^2$.

2.1.4 Loads

The loads for the system are basically just different sized fly wheels, ranging from an inertia of $70 * 10^{-6} kgm^2$ to $27300 * 10^{-6} kgm^2$. This gives the user an option to choose to have an over, normal or under sized inertia ratio for the servo to see how this affects the system. The user also has to take in to account that inertia from the clamping elements, gearbox, shaft and couplings has contribution to the total inertia. These are, however, almost constant no matter what configuration is used.

(a) ELC (ELastomer Coupling)

(b) EC2 coupling

(c) BC2 coupling

Figure 3: Couplings

2.1.5 Couplings

Currently for the mechatronics rig there are three different couplings to choose from. Two bellow couplings, EC2 and BC2, with high performance, see figure 3(c) and 3(b), and one elastomer coupling, see figure 3(a), with the option to insert three different elastomer with different properties. The bellow couplings can handle 30 Nm torque while the elastomer coupling (ELC) can handle 6, 17, 21 Nm depending on the properties of the elastomer. These are labelled as Coupling 1 (ELC), Coupling 2 (EC2) and Coupling 3 (BC2), where Coupling 3 is the least expensive and Coupling 1 is the most expensive.

2.1.6 Gearboxes

The configuration of the rig includes three different gearboxes, ranging from the cheaper basic Alphira, see figure 4(a), gearbox to the more expensive LP+ 70, see figure 4(b), and the most expensive SP+ 60, see figure 4(c). The differences between the gearboxes are mainly the backlash, noise level and the no load torque. Even if the rated noise levels with no load are very close, the differences when run with loads are very clear. Basically more backlash produces more noise. The inertia of the gearboxes are basically the same to make it easier to configure different test, see table 1 for values. The gearboxes are labelled Gearbox 1 (SP+ 60) for the most expensive with best performance and Gearbox 2 (LP+ 70) for the middle one and for the least expensive Gearbox 3 (Alphira).

Table 1: Gearbox specifications

	Alphira	**LP+ 70**	**SP+ 60**
Inertia ($10^{-6}kgm^2$)	54	52	52
No load torque (Nm)	0.05	0.05	0.9
Backlash (arcmin)	20	12	6
No load noise (dB at 3000 rpm)	66	69	59

(a) Alphira (b) Alpha LP+ 70 (c) Alpha SP+ 60

Figure 4: Gearboxes

2.1.7 Shafts

There are four different shafts to the rig that can be used. One entirely made from stainless steel, two that have a smaller diameter on the middle to make them less stiff and one made from aluminium. All these give different properties to the system. Unfortunately tests with these shafts have not been run as they where added late in the project. The project's aim is to make a test area for developers so that they can perform their own tests and give them some ideas and hints where to look and how to perform some tests.

2.2 Software

The software used to control the mechatronics rig is written in Visual Basic in Excel and is using Direct Data Exchange (DDE) to communicate with a local OPC server on the computer. The OPC server acts as an Application Programming Interface (API) to the PLC where it is possible to up and download values and settings. In Excel it is possible to analyse uploaded values, download new profiles, change settings and tune the servo.

2.2.1 PLC ladder

The programming for the PLC is made in Ladder, the programming language mostly used in Tetra Pak for PLCs. The structure of the program handles different types of errors, runs CAM profiles (for more information see appendix C), simulates big programs and/or systems with delays and also simple motion commands like move and home. The move command tells the servo to move to a specific position with a given speed, acceleration and jerk. The home command takes the servo to it is home position, which is zero degrees.

The CAM profiles downloaded to the PLC is calculated with a function called Motion Calculate CAM Profile (MCCP) and put in a CAM profile variable that contains the master (time) and slave (degrees) positions and also how the profile should move between points - two options, cubic or linear. The motion is then executed with a Motion Axis Time CAM (MATC) that sends the instructions to the servo drive to perform.

When a motion instruction is started on the actual axis the software also starts a virtual axis as a reference measurement when uploaded to Excel. With this virtual axis it is possible to see how delays in the system affects the real axis and with this information set the tolerances large enough to avoid any errors. The delays are simulated by two periodic tasks, one which is executed every 10th millisecond and one every 25th milliseconds. In these tasks, when chosen, the servo starts while the reference axis starts as soon as the command is received. The time when the servo is initiated is then 0-10 or 0-25 milliseconds delayed and the difference can be studied in different runs.

When a CAM profile is started the software starts sampling values and saves these to a matrix with the dimensions 6 columns and 5000 rows. The values saved are position, velocity, acceleration, position error, torque and reference position which is the position of the virtual axis started at the same time as the actual axis. The values are saved every second millisecond which gives a sample rate of 500 Hz. This is controlled by a motion event as an interrupt that is executed at each coarse update rate set to two milliseconds. The user is not able to change this sample rate due to restrictions in the PLC.

Once started the sampling continues for ten seconds and can not be stopped until it is done. This gives some restrictions when using the program as the user should not start a new CAM profile or other motion event within the ten second time window. If started before, the sampled series might be invalid.

The complete Ladder code can be found in appendix H.

2.2.2 Excel interface

The interface used to communicate with the program in the PLC is written in VBA in Excel. From the Control sheet the controls for plotting, motion instructions, uploading of data and downloading of new CAM profiles are located. The second sheet, Tuning, has the controls for tuning the servo, where the user is able to upload and download settings to the PLC. From here it's also possible to plot a desired number of samples to see how the settings affects the performance right after the runs.

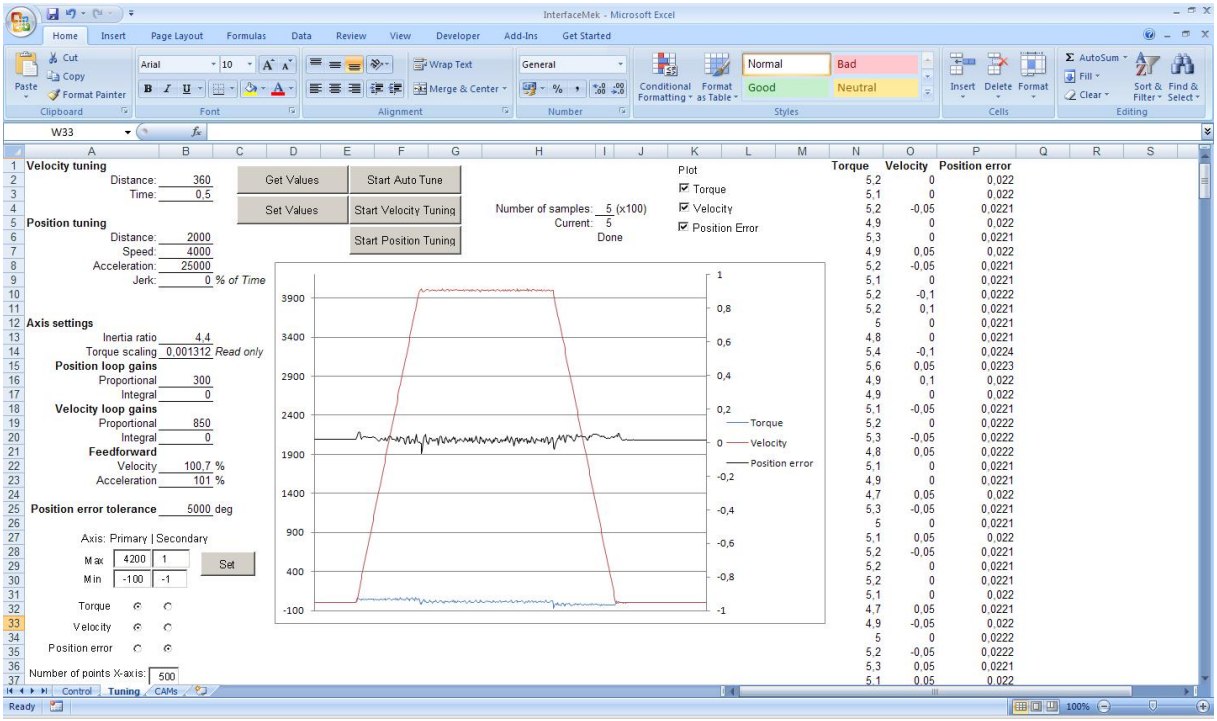


Figure 5: Tuning interface page

Tuning

The servo is controlled by two cascade coupled PI controllers, the inner loop is a PI controller for the position and the outer loop is a PI controller for the velocity. Only the proportional gain is used on these two control loops, so the PI controllers are used as plain P controllers, setting the integral part to zero.

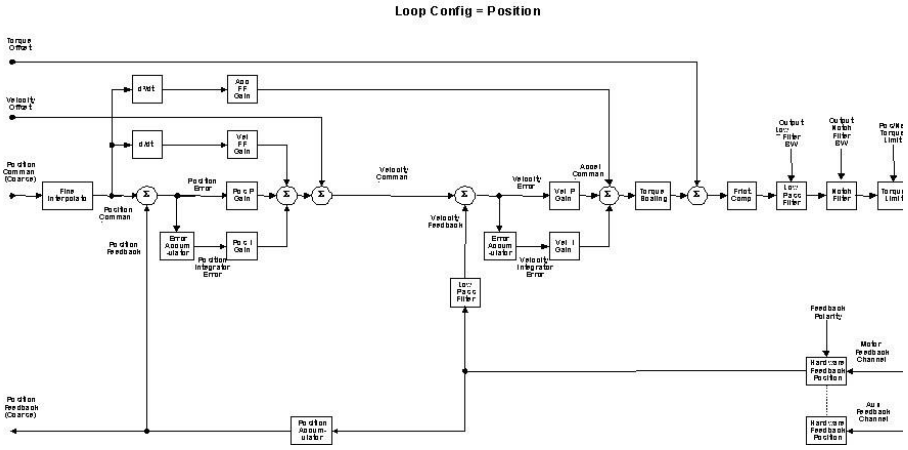


Figure 6: Block diagram over the position loop

The P-part, proportional gain, is multiplied with the current offset for either the position or the velocity, depending on the loop. It is hard to get a good understanding of these two loops due to the lack of information and low resolution pictures, which are hardly readable. But figures 6 and figure 7 shows the available documentation of these control loops.

Tuning the servo is done in three steps. The first step is to acquire the inertia ratio of the load versus the servo. This is done by running a function called Motion Run Axis Tuning (MRAT), where the PLC runs a single triangular velocity profile where acceleration time, deceleration time and position servo bandwidth are measured. From these parameters the acceleration, deceleration and inertia are calculated. After the MRAT function the measured and calculated values are sent to the function Motion Apply Axis Tuning (MAAT) which from these parameters calculates position proportional and integral gain, velocity proportional and integral gain, velocity and acceleration feedforward, maximum speed, acceleration and deceleration, output filter bandwidth, output scaling and position error tolerance. These two functions are called with the "Auto Tune" button from the Tuning sheet. The most important part that are kept are the inertia ratio. If the inertia ratio already is calculated or saved from a previous run, the inertia ratio can be entered and downloaded to the drive. The control variable calculated from the inertia ratio is the torque scaling, see equation 1. Once this is acquired the next step is to tune the velocity loop.

$$TorqueScaling = \frac{100 * J_{motor}(1 + InertiaRatio)}{Torque_{Rated} * \frac{DriveResolution}{ConversionConstant} * \frac{1}{2*pi}} \quad (1)$$

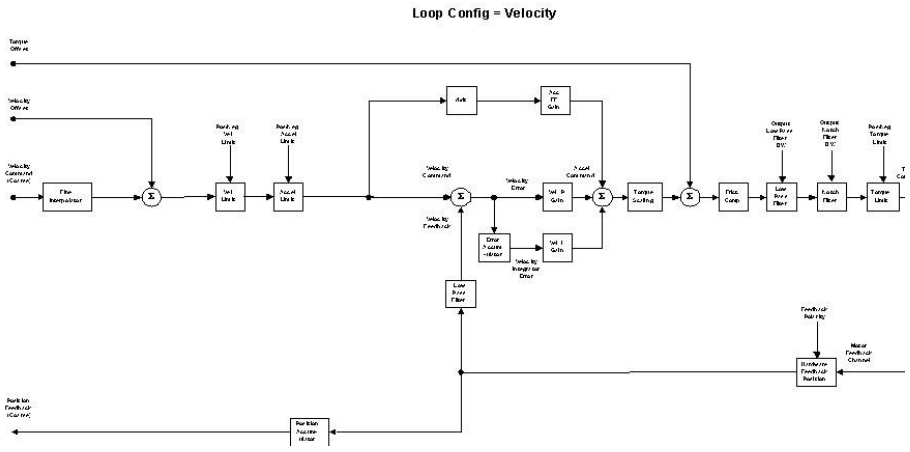


Figure 7: Block diagram over the velocity loop

The servo drive has two control loops one velocity loop and one position loop. The first loop to tune is the velocity loop and then when the velocity loop is tuned the tuning of the position loop is started.

Tuning the velocity loop is initiated by setting all the values in the position loop to zero and the velocity feed forward to 100% and then run "Start Velocity Tuning" in the Excel sheet. Once running the velocity gain is increased until the system goes unstable. No plots are made while tuning the velocity loop. The user should clearly hear when the system becomes unstable. When the system is unstable the velocity gain is reduced by 50%. Now when the velocity loop is tuned, the velocity tuning is stopped and the velocity feed forward is set to zero and the tuning of the position loop can be started.

When tuning the position loop the gain is slowly increased and each time the "Start Position Tuning" is pushed, one session is run and chosen data is downloaded to the sheet. The position gain is set as high as possible with no overshoots in the position error.

The last part of the tuning is to set the feed-forward parameters. If possible this should be done with the actual CAM that are going to be run, else with the position tuning. Here the velocity feed forward is set so the position error during the constant speed part is as low as required. The acceleration feed forward is set to bring the position error during the acceleration to a minimum. When these parameters are set the tuning is complete.

To make the tuning easier and make the charts easier to read, controls for changing the chart properties are added for the users' controllability, see figure 8.

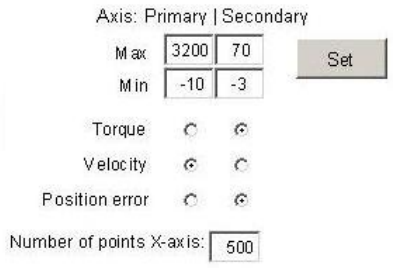


Figure 8: Tuning parameters

Data acquisition

Once the tuning of the servo is done, testing of the current setup can begin. Most of the tests are run with a CAM profile downloaded from the "CAMs" sheet where some different CAM's are predefined. A user can add their own CAM's to the sheet as long as they maintain the same structure. The structure is taken from "camTOOLinx Ver 3.12.1", an Excel tool to design CAM profiles developed by Tetra Pak. Once the CAM's are placed on the "CAMs"-page and the CAM-list is updated they can be chosen and downloaded to the PLC. When downloading a CAM to the PLC the type of coupling, time scaling and distance scaling needs to be set and the CAM can not be longer than 200 points.

When the CAM has been downloaded the drive and servo should be powered up by pushing "Enable Drive", the contactor for the drive and Motion Servo On (MSO) is turned on, and the servo is ready to start.

The CAM is run once the "Start Servo" is pushed and continues to run until "Stop all motion" is pressed. When the servo is started data acquisition is started and saved to arrays in the PLC CPU for ten seconds with a resolution of 500 Hz. No new motion instructions should be sent until the sampling is done.

When the test has been run data can be uploaded from the PLC and put in to a workbook of the users' choice. If the user does not enter a workbook that can be found a new one will be created with the name the user chooses. The uploaded data from the PLC are position, velocity, acceleration, position error, torque and reference position. The user can choose to download 100-5000 values (0.2 second - 10 seconds) of samples. Included in the upload is also the current tuning settings.

When a user has a workbook with uploaded data it can be used at any time to analyse the downloaded data. When analysing the data it can be plotted with a choice of any number of data points and the chosen data series can be plotted against any other series as x-axis. It is also possible to plot the series as a FFT function to see different frequencies of the system. This requires that the user has enabled the analysis toolpak for Excel (see appendix A). This function can be found under Excel options, Add-Ins,

then manage Excel Add-Ins and enable Analysis ToolPak. All plots made are inserted last is the workbook with the data.

The complete VBA code can be found in appendix G.

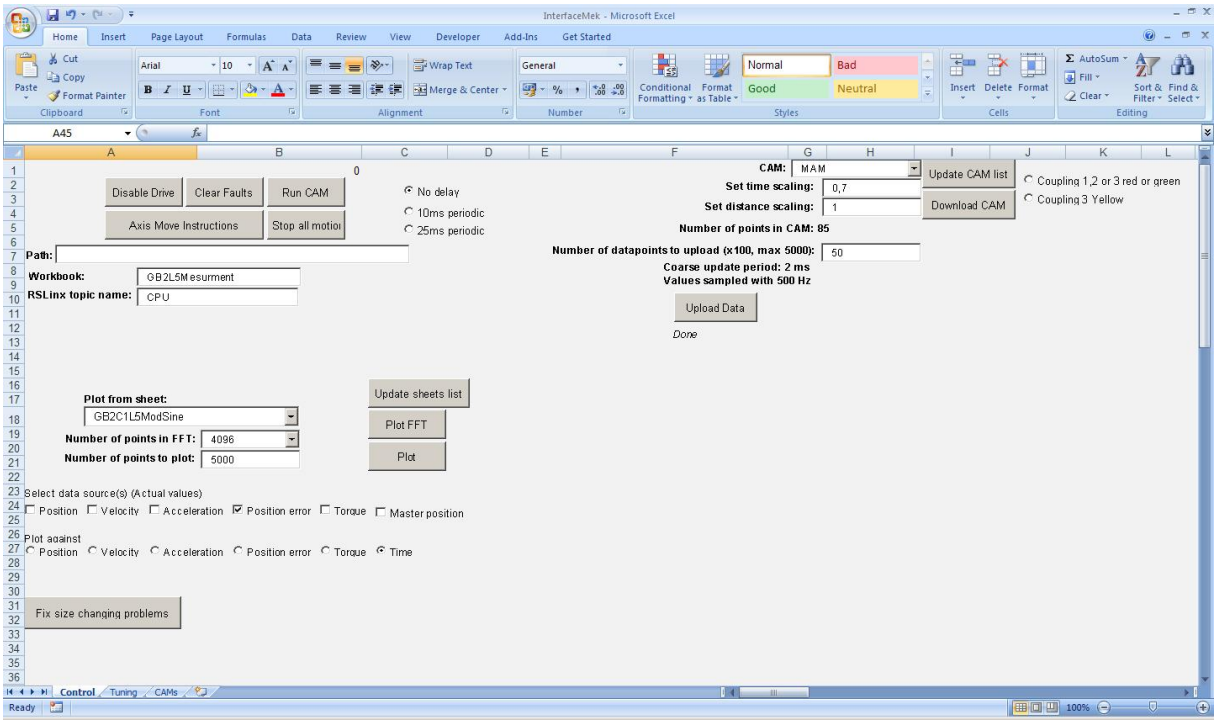


Figure 9: Control page. Data acquisition and control

3 Tests and results

All tests and graphs are generated with the Mechatronics Demo rig Excel software. These tests are made to give some hints to the users and give them an idea of how to use the software and how to detect different properties. As there are very many combinations that can be defined, the tests analysed here are based on the following eight hardware configurations.

Gearbox 1	Coupling 3 Type A	Load 3
		Load 5
	Coupling 1	Load 3
		Load 5
Gearbox 2	Coupling 3 Type A	Load 3
		Load 5
	Coupling 1	Load 3
		Load 5

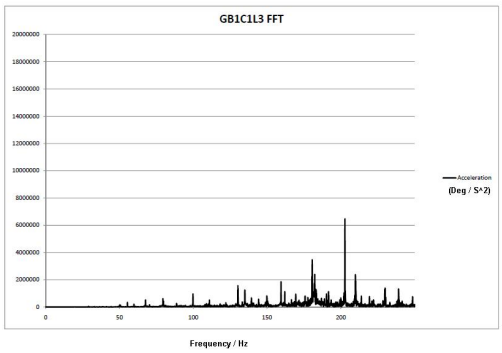
Each of these configurations were run with a trapezoidal, a S-Curve, a ModSine and a ModTrap CAM profile as well as a JOG motion. For more information on CAM profiles see appendix C.

3.1 Gearbox testing

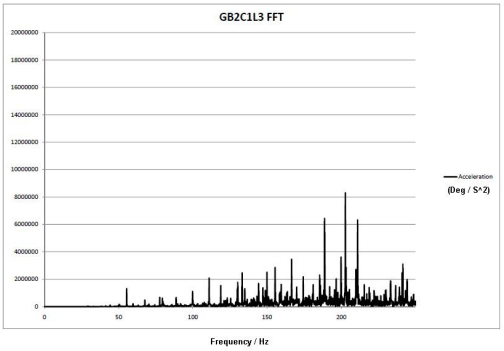
Gearboxes comes in a variety of different types and quality. These tests are going to show the differences in backlash for the different gearboxes. The easiest difference to notice is the sound level emitting from the gearboxes. The three different ones used in this thesis have a rated noise level of (ranging from best to worst) 58 (SP+60), 70 (LP+70) and 68 dB (Alphira). These ratings are taken from the manufactures data sheet and are measured at 3000 rpm with no load and differs from when the gearbox is under load.

To run the test choose the configuration that are going to be used. Choose load, coupling and two gearboxes. Set up the system with the first gearbox and tune it. Once the tuning is done it is time to collect a range of values. The easiest way to get a good series of samples is to run the servo at a constant speed, also called MAJ (Motion Axis Jog). This can be done under Axis Move Instructions where jog speed, acceleration and jerk can be set. Once the system is running at a constant speed the sampling can begin by pushing "Start Data Collect". The label underneath will change to "Collect complete" when done, then close the form.

Type in a workbook where you would like to save the data and upload the sampled values from the PLC. To get best results all five thousand values should be uploaded.



(a) FFT of a MAJ 4000 rpm GB1



(b) FFT of a MAJ 4000 rpm GB2

Figure 10: FFT of MAJ runs at 4000 rpm with load 3

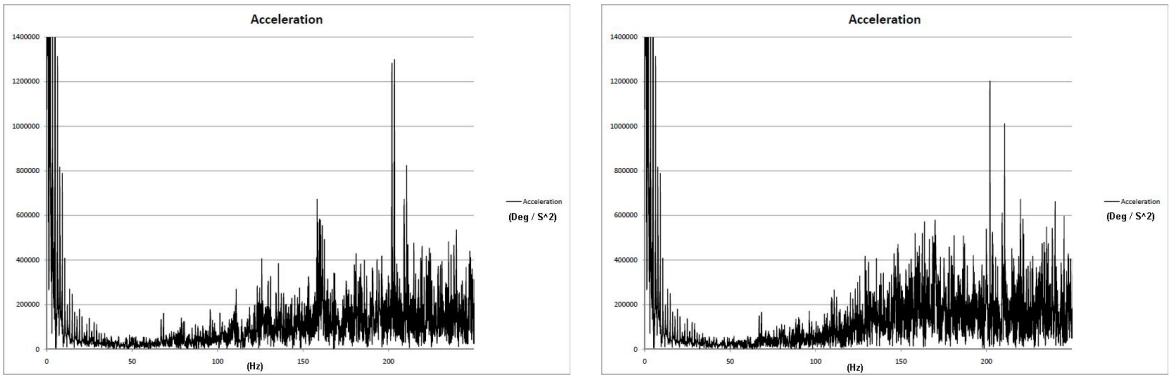
Check that the downloaded values are the correct ones, i.e. check that the speed is correct. Then change gearboxes and redo the process from the tuning part.

When both data series have been collected the analysing can begin. A good way to see the differences are by making a FFT plot and analyse the frequency contents of the two series. Analysing the figures in figure 10 it is easy to see that the right figure, 10(b), contains more noise than the left figure, 10(a), which is the run with Gearbox 1. This difference in vibrations are generated by the backlash in the gearboxes, as Gearbox 2 has a bigger backlash it generates more noise. It is also possible to just listen to the runs as the gearboxes with more backlash are a lot louder when under load. As the figures in table 1, gearbox specifications, show, the noise levels should be about the same at runs with no load but this does not comply when the gearbox is used with loads.

3.2 Coupling testing

Differences between the couplings are mainly the dampening they have. The easiest way to detect differences is in systems with a lot of vibrations. Like a system with a small load, which gives high gains when tuning, and a gearbox with some backlash. Figure 11 shows two runs with the only difference that figure 11(a) is a FFT graph for a ModSine profile with a coupling with high dampening and 11(b) is a FFT run with a ModSine with a very stiff coupling.

With the higher dampening the frequencies around 150 Hz and above 210 Hz are less but the two spikes at 165 Hz and 210 Hz are a little bit amplified.



(a) Coupling 3, high dampening

(b) Coupling 1

Figure 11: FFT plots for the acceleration of a ModSine CAM

3.3 Load testing

Sizing the servo to the load is an important task when designing a system in order to know that the servo is going to perform as expected. To be able to do this in a good way it is crucial to know how the size of the load affects the system. The main thing when the load increases is that the gains get lower when tuning. With this rig the system is very stable and it is very difficult to make it unstable when the load is small, but not impossible. With the hardware setup Gearbox 1, Coupling 1 and Load 3 and 5 the following tuning parameters are achieved:

Table 2: Tuning parameters

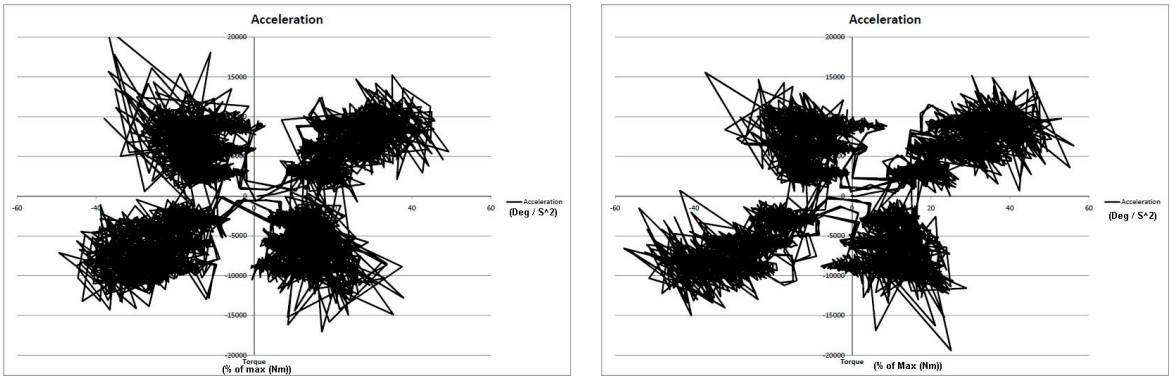
	Inertia ratio	Pos. prop. gain	Vel. prop. gain	Vel. ff	Acc. ff
Load 3	2.3	500	1300	100.7%	101%
Load 5	4.4	300	850	100.7%	101%

As seen in table 2 the gains get lower and thus less aggressive when the loads get bigger if tuned correct. It is important to keep the system less aggressive when the loads get higher to avoid that the system goes unstable. This also means that it may not be able to follow small sudden changes in a CAM profile with enough accuracy or speed.

What the user has to keep in mind is that during heavy load with high acceleration and deceleration a lot of heat is generated by the servo. Avoiding this can be hard under some conditions. Trying to increase the acceleration and deceleration phase can be one option to lower the generated heat. A good tool to theoretically examine some of these properties are "Motion Analyzer" by Rockwell Automation. On this rig the maximum torque is set to 75% continuous and 150-200% peak of maximum to not damage the

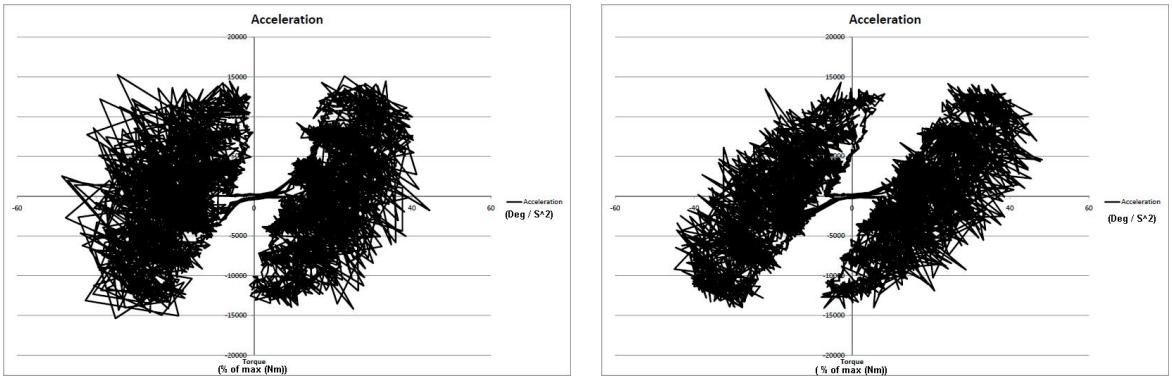
couplings.

One interesting aspect of the different loads is that the only CAM that gave a big difference in torque was the trapezoidal CAM. Even if the inertia ratio was almost twice as high with the larger load, as seen in figure 13, the torque is kept to about 45-50% with both Load 3 and 5 and the acceleration is the same with the ModSine CAM profile. While in figure 12, with the trapezoidal CAM profile, the difference is more apparent as the torque goes up from 50% to about 60% and the acceleration stays almost the same.



(a) Load 3 (b) Load 5

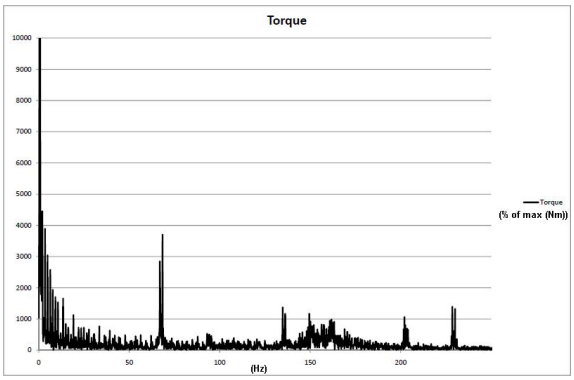
Figure 12: Acceleration against Torque Trapezoidal CAM



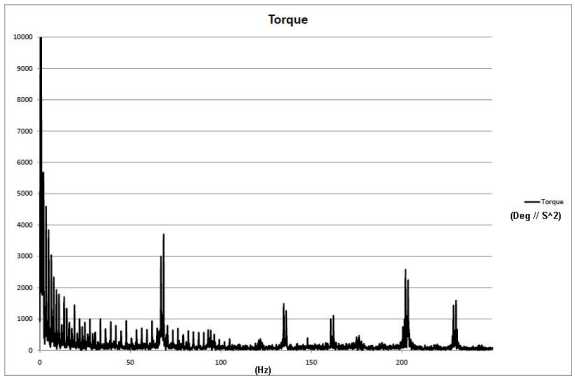
(a) Load 3 (b) Load 5

Figure 13: Acceleration against Torque ModSine CAM

When using Load 3 there is some strange behaviour around 160-170 Hz as seen in figures 14 and 15 that are not present for Load 5, no matter if a trapezoidal or ModSine CAM is used. But then if figures 14(b) and 15(b) are compared it is possible to see that

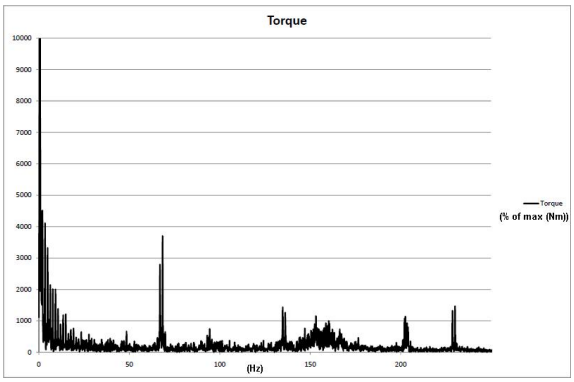


(a) Load 3

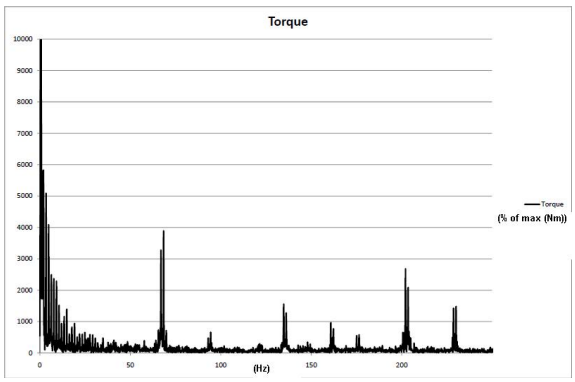


(b) Load 5

Figure 14: FFT plots for the torque with trapezoidal CAM



(a) Load 3



(b) Load 5

Figure 15: FFT plots for the torque with ModSine CAM

the bigger load has some problems handling the trapezoidal CAM and dampening the lower oscillations that appears while a smaller load, compare figures 14(a) and 15(a), can cope with the oscillations and dampen them out.

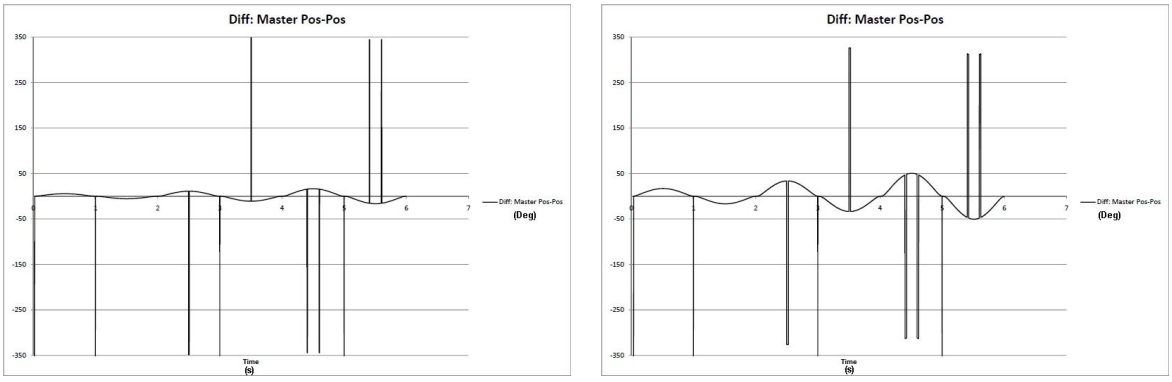
The conclusion is that when using a bigger load it is more important to design your CAM profile in a better way to minimize oscillations and vibrations in the system to spare the mechanics.

3.4 Delays

When handling big programs with time critical events it is important to know how the given commands actually are performed. This rig only has one axis to perform tests on. But there is a virtual axis running in parallel with the actual axis -when the actual axis is stopped the virtual axis is stopped, when performing a homing the virtual axis also performs a homing.

When downloaded to a spreadsheet the value of the virtual axis is called "Master Position". The difference between the real axis and the virtual axis can be plotted by selecting "Diff: Master Pos-Pos". As the simulated delays for starting the CAM profile are made by a periodic task, of for 10 ms and of for 25 ms, the delays are not always the same even if the same settings are used. In figure 16, both runs are made with the same settings in the PLC but as the tasks are periodic they might not start at the same time, as seen in the left figure, 16(a), compared to the right figure, 16(b), where the actual axis started a short time later.

As seen the maximum difference is 50 degrees in figure 16(b) and in figure 16(a) it is about 10 degrees. In time critical systems with a lot of code this can prove to be very important. There are a few ways to get round this problem in RSLogix. One good option is to have a virtual axis running in the background and synchronize all other axes' with this. This means that it is possible to tell an axis to start when another axis reaches a set value. Then starting the axis is managed by the drive rather than the CPU which will lead to higher accuracy and shorter delays.

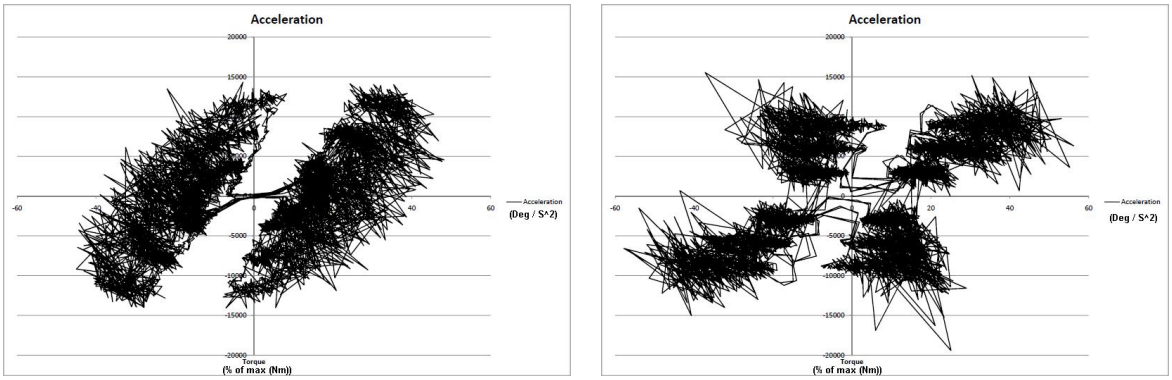


(a) Short delay

(b) Long delay

Figure 16: Two independent runs with the same CAM profile and settings

With higher speeds larger errors will appear, the runs in figure 16 was maximum 1800 rpm. The spikes in the graphs appear in the zero crossing when the virtual axis passes zero degrees until the actual axis passes zero degrees.



(a) ModSine CAM (b) Trapezoidal CAM

Figure 17: Acceleration against torque

3.5 CAMs

When designing different CAM profiles it is important to understand the differences between the CAM's and how this affects the performance. Using a low performance CAM can use up to 30-40 percent more torque than a high performance CAM and still have the same acceleration and perform the same task. As seen in figure 17 where the left figure, 17(a), is a plot showing acceleration versus torque where the max torque is about 45-50 percent and in the right figure, 17(b), the torque reaches almost 60 percent. Both motions perform the same action, where the load is rotated 360 degrees and back to 0 then 720 degrees and back to 0 and the last movement is 1080 degrees and back to 0.

When designing the movements and CAM profiles it is important to consider how the torque is affected by the choices made. If made properly the designer can keep the wear of the equipment down and maybe keep the size of the servo small to save weight and money. This an important task when designing the CAM profiles. There are a lot of choices to consider when the design is done. For example, if the friction in the system is high it can be used in the breaking phase of the CAM, this is helpful as the acceleration phase can be longer and requires less torque, keeping the heat and torque of the servo down.

4 Result and Discussion

When using the software and analysing data from the rig, especially with the FFT spectrum, vibrations that are harmful for the mechanics can easily be spotted. As the rig is going to be used as a training platform for development engineers the software provides an easy way for the persons using it to see the differences of how they design and control their systems.

The software will hopefully give a new approach for developers to improve their knowledge about mechatronic systems. As developers have a solid experience on either mechanical systems or electronic systems, but unfortunately experience of both are not as common. Giving people the opportunity to sit down and have a hands-on experience with the design and tune of a mechatronic system will hopefully increase this knowledge.

When reading about different CAM-profiles, an understanding for what actually happens with the system and the performance between the profiles are not apparent. Analysing tests with different profiles shows what differences that affects the properties.

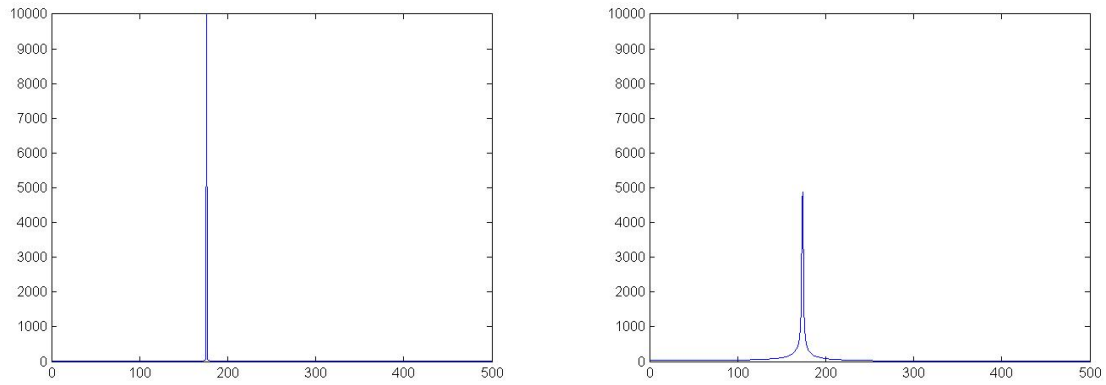
Overall it is a well working software with the ability to tune and control the rig. Download and run CAM-profiles after the users choice, use basic motion function. Some basic tests have been developed for the user to try out, but hopefully own ideas will be tried out and new results will be found. The ability to save data into a separate workbook for later use or to send to a college for an opinion.

5 Future work

This mechatronics project is prepared for belt drive tests as well as the current hardware configuration with shafts. Parts and necessary configurations are made on the hardware side of the project for the assembly to be possible.

On the software side additional PLC programming are required and new tests need to be developed in the Excel interface. The analysis of data and communication with the PLC is still the same and can be reused without any alterations.

When calculation the FFT of the series some leakage is affecting the calculations. Leakage appears in FFT graphs when the finite series does not start and end at zero. As seen in figure 18 (left figure) the frequency of 175 Hz appears as a tall spike in the diagram, but in the right figure, 18(b), the spike is somewhat stretched and does not show as high amplitude as the figure with no leakage. This might make a FFT graph somewhat harder to interpret. This leakage problem can be solved by an implementation of windowing on the series. Using a Hanning window would most likely be sufficient, but preferable the user should have some options on what kind of window he or she would prefer.



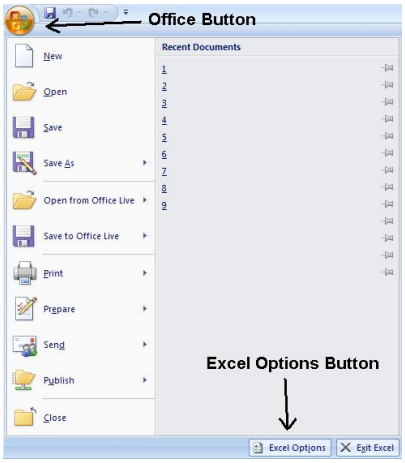
(a) FFT 175 Hz sinus no leakage

(b) FFT 175Hz sinus leakage

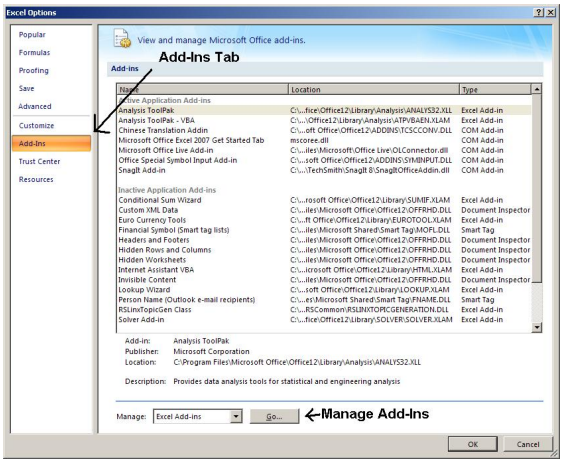
Figure 18: FFT leakage of a 175 Hz sinus

A Setting up Excel

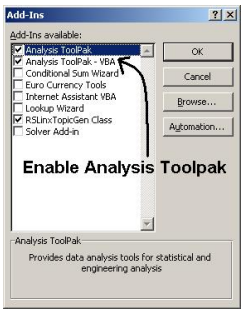
To be able to use all functions in this program, Excel analysis toolpak has to be activated. To activate this function first push the Office button, down in the right corner of the menu is a button named Excel options, see figure 1(a), push it. Sometimes it is necessary to close the interface to open Excel options. Once the Excel options are open, go to the tab "Add-Ins", figure 1(b) and click "Go". A new window opens where you should have the options to check the "Analysis ToolPak" and "Analysis ToolPak VBA", figure 1(c). Check both these options and click "Ok". Reopen the interface and all should be good to go.



(a) Office menu



(b) Excel Options



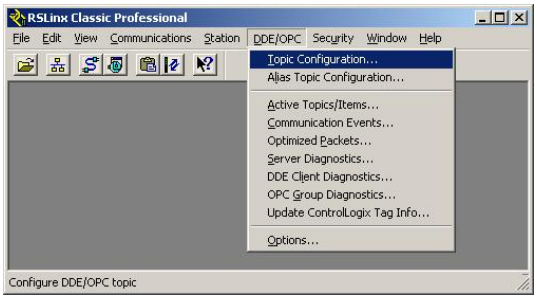
(c) Manage add-ins

Figure A.1: Setup Microsoft Office

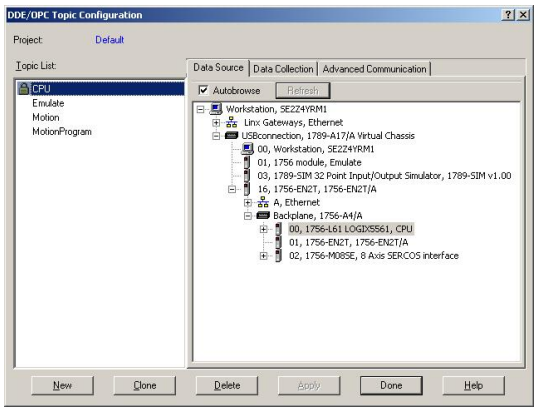
B Setting up RSLinx

To connect to the PLC with the computer, RSLinx needs to be configured and given a topic name.

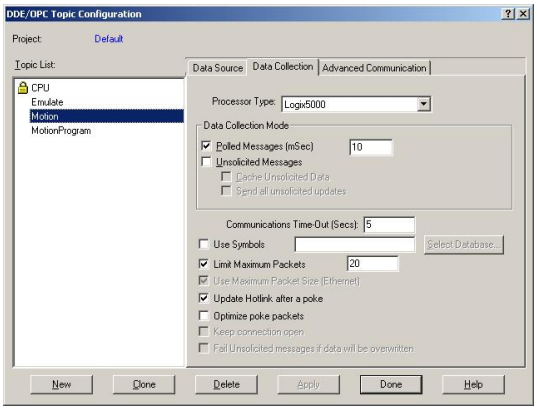
First open RSLinx topic configuration, see figure 1(a). Create a new topic and give it a name you would like to use, see figure 1(b). Under the data source tab find the logix CPU (LOGIX5561) and select it. Open the data collection tab, figure 1(c), and choose Logix5000 as "Processor Type". And under the last tab "Advanced Communication" choose "Communication Driver", figure 1(d), to be "USBconnection". The topic name should now work with the interface in Excel.



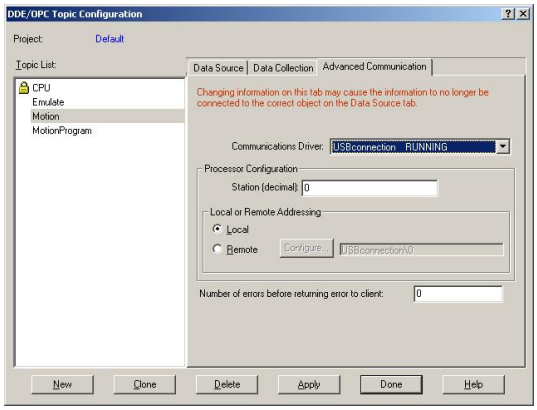
(a) RSLinx main window



(b) RSLinx topic configuration



(c) RSLinx Data Collection



(d) RSLinx Advanced Communication

Figure B.1: Setup RSLinx

C CAM profiles

CAM profiles can be designed in many different ways, this in many cases can change the performance to the better or worse if performing the same motion. Following information and pictures are taken from "CAMTool Different Approaches Feb 09" written by Thorbjörn Bengtsson.

One of the easiest CAM profiles to design is the trapezoidal profile. This is however a CAM profile that are very harsh to the mechanics due to infinite jerk and thus also gives a lower acceleration. In figure C.1 it's possible to see how the jerk has three spikes, in the beginning of the acceleration, when the deceleration motion begins and when the servo stops. These three events are very harsh to the mechanics.

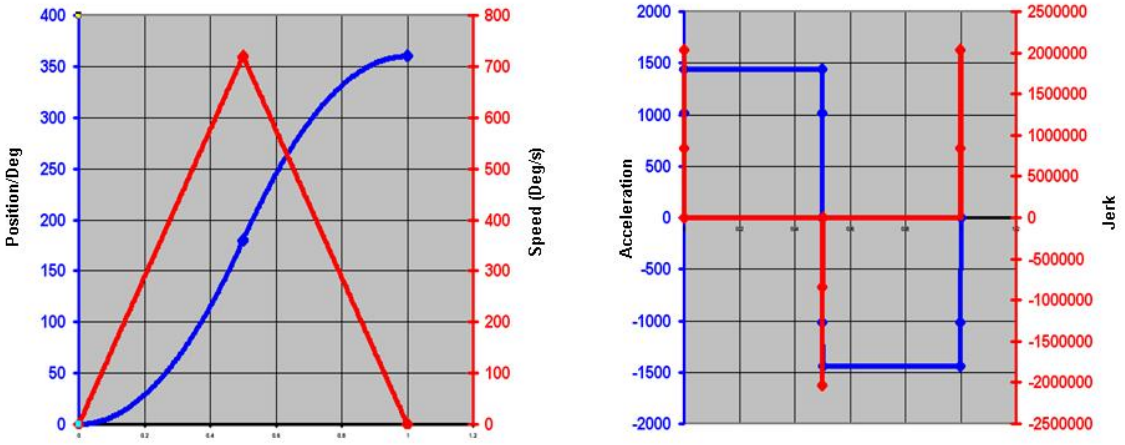


Figure C.1: CAM profile: Trapezoidal

The RA S-Curve is also very harsh to the mechanics but has a higher acceleration. Using this kind of curve, figure C.2, with 100% jerk is not much better than using the trapezoidal curve. But with some design modifications the RA S-Curve can be used with 50% jerk, figure C.3, which gives a smother motion and spares the mechanics of high jerk. This limits the acceleration as well as the jerk and gives a higher maximum speed to the motion.

Adding some more points to a S-Curve with 50% jerk gives a design called ModTrap profile, figure C.4, where the more reference points are used in the acceleration and deceleration phases and thus gives a smoother acceleration curve which takes down the jerk. This profile is usually a good choice when initial designing a CAM profile.

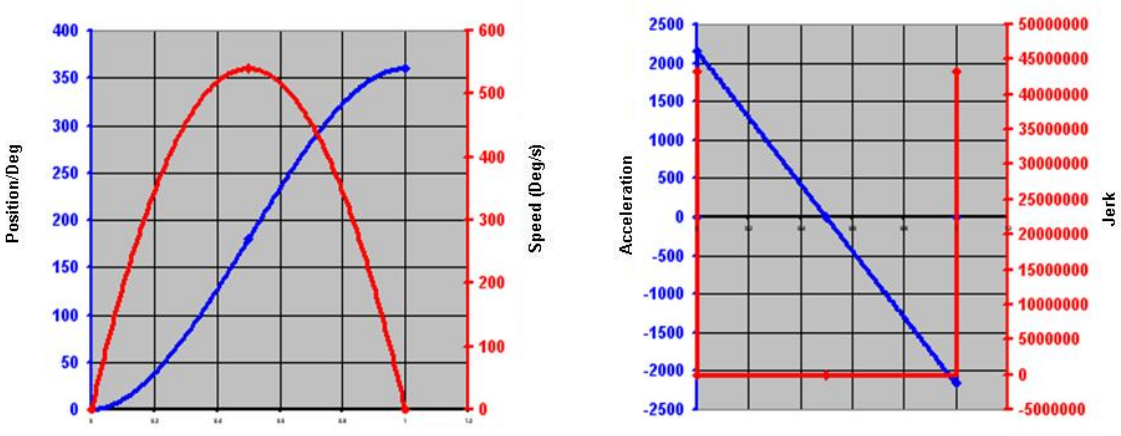


Figure C.2: CAM profile: SCurve 100% jerk

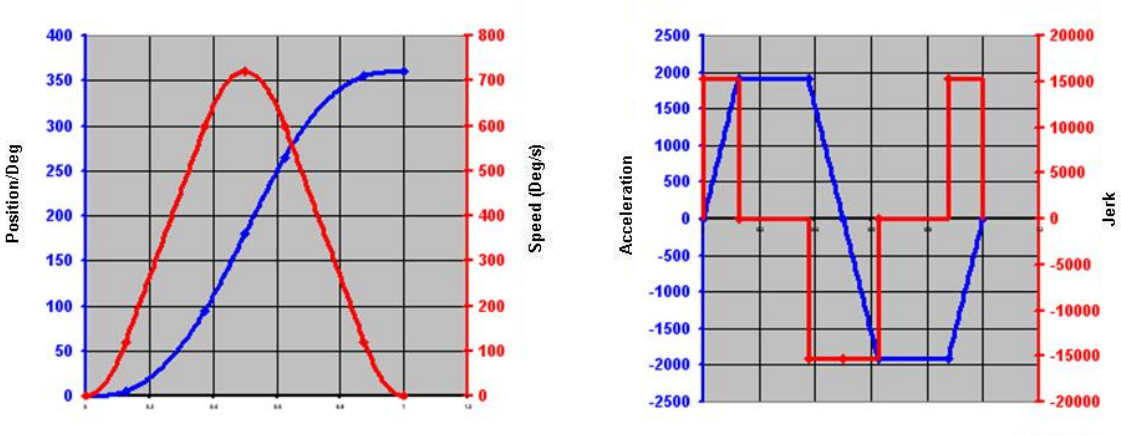


Figure C.3: CAM profile: SCurve 50% jerk

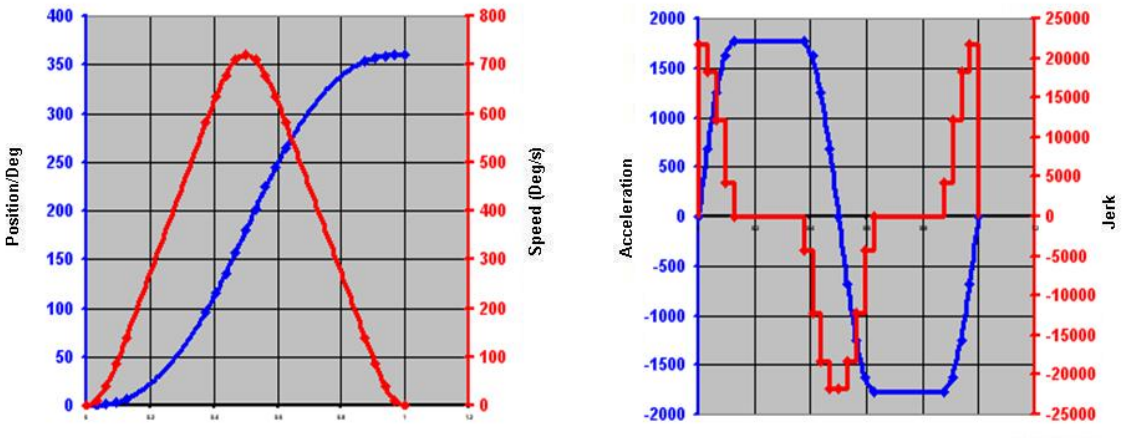


Figure C.4: CAM profile: ModTrap

A smoother way of a designing is a ModSine profile, figure C.5. This type profile does not have a constant acceleration phase and uses the entire time to accelerate or decelerate. This profile is nice to the servo due to good accordance with the motor envelope as it has high acceleration at low speeds.

Combining the acceleration phase of the ModSine and the deceleration phase of the ModTrap profile, figure C.6, gives a very good movement for the servo in accordance with the motor envelope.

If a system has high friction this can be used to increase the time spent on acceleration and decrease the time of deceleration as the friction can absorb a lot of energy instead of the servo. If this can be done the servo can have lower acceleration over a longer time which would lead to lower torque and maybe even take down the size of the servo drive and the servo in some applications.

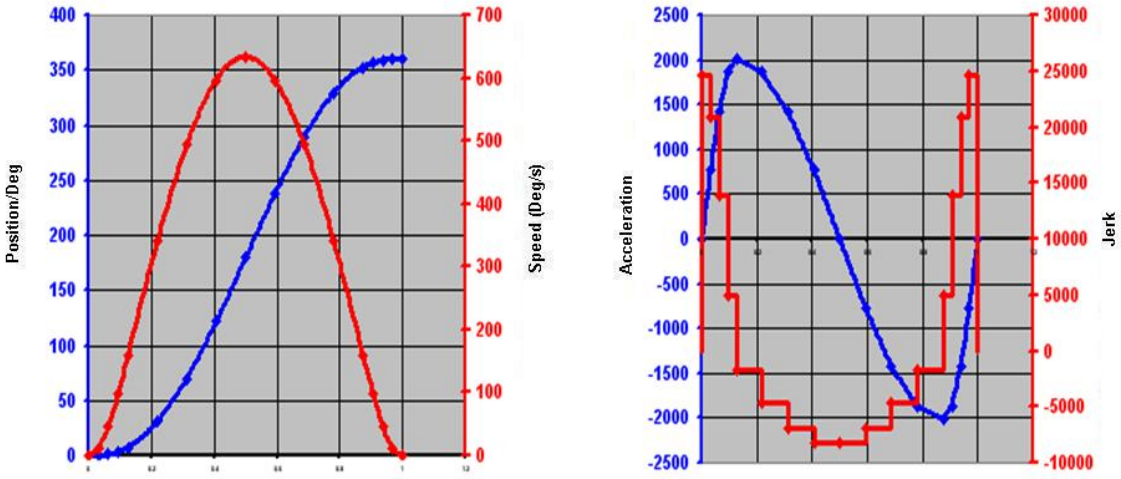


Figure C.5: CAM profile: ModSine

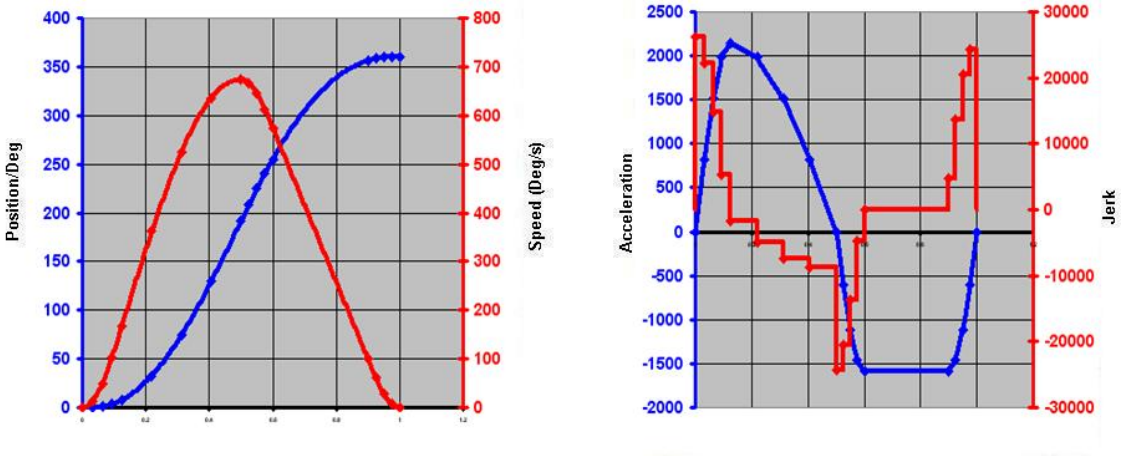


Figure C.6: CAM profile: Mixed ModSine ModTrap

D Nomenclature

API - Application Programming Interface
CAM - Motion profile for the servo to follow
DDE - Direct Data Exchange
ELC - Elastomer Coupling
GSV - Get System Value
HMI - Human Machine Interface
MAAT - Motion Applay Axis Tuning
MAFR - Motion Axis Fault Reset
MAH - Motion Axis Home
MAJ - Motion Axis Jog
MAM - Motion Axis Move
MAS - Motion Axis Stop
MASR - Motion Axis Shutdown Reset
MATC - Motion Axis Time Cam
MCCP - Motion Calculate Cam Profile
MRAT - Motion Run Axis Tuning
MSF - Motion Servo Off
MSO - Motion Servo On
NEQ - Not Equal
OPC - OLE for Process Control
SERCOS - SErial Real-time COmmunication System
SSV - Set System Value
VBA - Visual Basic for Applications

E Couplings

EC2 bellow coupling
Rated torque: 30 Nm

BC2 bellow coupling
Rated torque: 30 Nm

Elastomer coupling
Type A (Red): Shore hardness 98 Sh A (High damping)
Type B (Green): Shore hardness 64 Sh D (High torsional stiffness)
Type C (Yellow): Shore hardness 80 Sh A (Very high damping)
Rated torque:
Type A: 17 Nm. Type B: 21 Nm. Type C: 6 Nm.
Max torque:
Type A: 34 Nm. Type B: 42 Nm. Type C: 12 Nm.

F Tables

Table F.1: Inertia hardware

	Motor	**Alphira**	**LP+ 70**	**SP+ 60**	**Shaft**	**ELC**	**BC2**	**EC2**
Inertia ($10^{-6}kgm^2$)	44	54	52	52	50	20	120	120

Table F.2: Inertia loads

	Load 1	**Load 2**	**Load 3**	**Load 4**	**Load 5**	**Load 6**
Inertia ($10^{-6}kgm^2$)	70	180	870	1970	4130	27300

G Code: Visual Basic

G.1 Control

```
        'Sets global variables
Public Cancel As Boolean 'Boolean to cancel data upload
Public interface As Workbook

Private Sub Axis_Button_Click()
    'Open form to set Axis settings
    Axis_Form.Show
End Sub

Private Sub Bugfix_Button_Click()
    'Shows a dialog and asks if the user would like to add information to their Windows register, if
        Yes the script runs else it exits
    If MsgBox("This will add a value to the windows register. Do you wish to continue?", vbYesNo, "
        Bugfix") = vbNo Then Exit Sub
    Dim myWS As Object
    'access Windows scripting
    Set myWS = CreateObject("WScript.Shell")
    'write registry key, 11.0 if Office 2003, 12.0 if Office 2007
    If Application.Version = "12.0" Then myWS.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Office
        \12.0\Common\Draw\UpdateDeviceInfoForEmf", "1", "REG_DWORD"
    If Application.Version = "11.0" Then myWS.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Office
        \11.0\Common\Draw\UpdateDeviceInfoForEmf", "1", "REG_DWORD"
End Sub

'Checks sheet cams for available CAMs and puts them in the combobox
Private Sub CAM_List_Button_Click()
    'Declares variables that are going to be used
    Set interface = ThisWorkbook
    Dim counter As Integer
    Dim setFirst As Boolean

    'Resets the parameters that are going to be used
    setFirst = True
    CAM_Box.Clear
    counter = 1

    'Scans top row of sheet CAMs to find the name of the CAMs and adds them to the list. Checks 1000
        colums
    Do While counter < 1000
        'Checks the cell to see if's not empty
        If Sheets("CAMs").Cells(1, counter) <> "" Then
            'If the cell contains text, it's added to the combobox
            CAM_Box.AddItem Sheets("CAMs").Cells(1, counter).Text
            'Sets the default value to the first found
            If setFirst Then
                CAM_Box.Text = Sheets("CAMs").Cells(1, counter).Text
                setFirst = False
            End If
        End If
        'Increase the counter
        counter = counter + 1
    Loop

End Sub


'Resets axis status
Private Sub Clear_Faults_Button_Click()
    'Stops all motion
    Call Stop_Button_Click
    'Sends command to reset axis faults
    ThisWorkbook.SendData "1", "Program:MainProgram.bClearFaults,L1,C1"
End Sub

'Function to upload data from PLC
Private Sub Data_Button_Click()
    'Sets variables to use
    Set interface = ThisWorkbook
    Dim DDEchan As Long
    Dim counter As Integer
    Dim vSheet As Variant
    Dim path As String
    Set wBook = Sheets("Control")
```

```
'Sets the path for the workbook, if nothing is written in the path field, path sets to same as
     current workbook
If Path_Box = "" Then
    path = ThisWorkbook.path
Else
    path = Path_Box.Text
End If

'Resets variables
counter = 0
Cancel = False

'Opens connection to OPC-server
DDEchan = Application.DDEInitiate("RSLinx", Topic_Box.Text)

'Sets the workbook where data is going to be stored
If Dir(path & "\" & Workbook_Box.Text & ".xlsx") <> "" Then 'checks if workbook exists in
     directory
    Dim i As Long
    For i = Workbooks.Count To 1 Step -1 'Loop to check if workbook is open
        If Workbooks(i).Name = Workbook_Box.Text & ".xlsx" Then
            Set databook = Workbooks(i) 'Sets workbook if open
            Exit For
        End If
    Next
    If i = 0 Then
        Set databook = Workbooks.Open(Filename:=path & "\" & Workbook_Box.Text, UpdateLinks:=0) '
             Opens workbook if not open
    End If
'Same as above, but if "Show file extensions" is turned if in explorer, this loop is used istead
ElseIf Dir(path & "\" & Workbook_Box.Text) <> "" Then
    Dim n As Long
    For n = Workbooks.Count To 1 Step -1
        If Workbooks(n).Name = Workbook_Box.Text Then
            Set databook = Workbooks(n)
            Exit For
        End If
    Next
    If n = 0 Then
        Set databook = Workbooks.Open(Filename:=path & "\" & Workbook_Box.Text, UpdateLinks:=0)
    End If
'If no workbook is found, a new is created and saved
Else
    'Adds workbook
    Workbooks.Add

    'Saves workbook
    ActiveWorkbook.SaveAs path & "\" & Workbook_Box.Text

    'Creates temporary sheet needed to delete the unwanted ones
    Sheets.Add(After:=Sheets(Sheets.Count)).Name = "TemporarySheet"

    'Turns off alerts when deleting sheets
    Application.DisplayAlerts = False
    'Deleting unwanted sheets
    Sheets("Sheet1").Delete
    Sheets("Sheet2").Delete
    Sheets("Sheet3").Delete
    'Turns alerts of again
    Application.DisplayAlerts = True

    ActiveWorkbook.Save

    Set databook = ActiveWorkbook 'Sets workbook where data is going to be saved

    'Informs the user that a new workbook has been created
    MsgBox "Book not found, book created!"
End If

'Jumps to workbook where data is going to be saved
Application.Goto databook.Sheets(1).Range("A1")
'Creates new worksheet and sets the name. If name already exists the error is skipped and the name
     is set to generic
On Error Resume Next

'Creates a new worksheet to store data
databook.Sheets.Add(After:=databook.Sheets(Sheets.Count)).Name = "Data" & CStr(Sheets.Count)

'Turns of alerts
Application.DisplayAlerts = False

'Checks for temporarysheets in the workbook and deletes it
For Each vSheet In databook.Sheets
```

```
    If vSheet.Name = "TemporarySheet" Then databook.Sheets("TemporarySheet").Delete
Next

'Turns on alerts
Application.DisplayAlerts = True
On Error GoTo 0



'Sets labels of the uploaded data
Sheets(Sheets.Count).Range("A1") = "Angle"
Sheets(Sheets.Count).Range("B1") = "Velocity"
Sheets(Sheets.Count).Range("C1") = "Acceleration"
Sheets(Sheets.Count).Range("D1") = "Position Error"
Sheets(Sheets.Count).Range("E1") = "Torque"
Sheets(Sheets.Count).Range("F1") = "Time"
Sheets(Sheets.Count).Range("G1") = "Reference Position"

'Creates the "Cancel button"
databook.Sheets(Sheets.Count).Range("I7") = "Cancel"
databook.Sheets(Sheets.Count).Range("I7").Font.Bold = True

'Loop to upload data
Do While counter < Data_Box.value
    'Prints where the loop currently is
    Upload_Label.Caption = "Getting value " & CStr(counter * 100 + 1) & " to " & CStr(counter *
        100 + 100) & " of " & CStr(Data_Box.value * 100)
    databook.Sheets(Sheets.Count).Range("I6") = Upload_Label.Caption

    'Upload 100 values at once from first array
    databook.Sheets(Sheets.Count).Range("A" & CStr(counter * 100 + 2) & ":A" & CStr(counter * 100
        + 101)) = Application.DDERequest(DDEchan, "arrData[0," & CStr(counter * 100) & "],L100,C1
        ")

    'Checks if cancel button has been pressed
    DoEvents
    If ActiveCell.Text = "Cancel" Then Cancel = True
    If Cancel Then
        Exit Do
    End If

    'Upload 100 values at once from second array
    databook.Sheets(Sheets.Count).Range("B" & CStr(counter * 100 + 2) & ":B" & CStr(counter * 100
        + 101)) = Application.DDERequest(DDEchan, "arrData[1," & CStr(counter * 100) & "],L100,C1
        ")

    'Checks if cancel button has been pressed
    DoEvents
    If ActiveCell.Text = "Cancel" Then Cancel = True
    If Cancel Then
        Exit Do
    End If

    'Upload 100 values at once from third array
    databook.Sheets(Sheets.Count).Range("C" & CStr(counter * 100 + 2) & ":C" & CStr(counter * 100
        + 101)) = Application.DDERequest(DDEchan, "arrData[2," & CStr(counter * 100) & "],L100,C1
        ")

    'Checks if cancel button has been pressed
    DoEvents
    If ActiveCell.Text = "Cancel" Then Cancel = True
    If Cancel Then
        Exit Do
    End If

    'Upload 100 values at once from fourth array
    databook.Sheets(Sheets.Count).Range("D" & CStr(counter * 100 + 2) & ":D" & CStr(counter * 100
        + 101)) = Application.DDERequest(DDEchan, "arrData[3," & CStr(counter * 100) & "],L100,C1
        ")

    'Checks if cancel button has been pressed
    DoEvents
    If ActiveCell.Text = "Cancel" Then Cancel = True
    If Cancel Then
        Exit Do
    End If

    'Upload 100 values at once from fifth array
    databook.Sheets(Sheets.Count).Range("E" & CStr(counter * 100 + 2) & ":E" & CStr(counter * 100
        + 101)) = Application.DDERequest(DDEchan, "arrData[4," & CStr(counter * 100) & "],L100,C1
        ")

    'Checks if cancel button has been pressed
```

```
        DoEvents
        If ActiveCell.Text = "Cancel" Then Cancel = True
        If Cancel Then
            Exit Do
        End If

        'Upload 100 values at once from sixth array
       databook.Sheets(Sheets.Count).Range("G" & CStr(counter * 100 + 2) & ":G" & CStr(counter * 100 +
            101)) = Application.DDERequest(DDEchan, "arrData[5," & CStr(counter * 100) & "],L100,C1")

        'Checks if cancel button has been pressed
        DoEvents
        If ActiveCell.Text = "Cancel" Then Cancel = True
        If Cancel Then
            Exit Do
        End If

        counter = counter + 1
    Loop

    'Empties the cells where the "Cancel button" and loop information was
    databook.Sheets(Sheets.Count).Range("I6") = ""
    databook.Sheets(Sheets.Count).Range("I7") = ""

    'Goto sheet where data was downloaded
    Application.Goto databook.Sheets(Sheets.Count).Range("F2")

    'Resets counter
    counter = 0

    'Saves the tuning parameters to the sheet

    databook.Sheets(Sheets.Count).Range("T1") = "Tuning parameters"
    databook.Sheets(Sheets.Count).Range("T2") = "Inertia ratio"
    databook.Sheets(Sheets.Count).Range("T3") = "Torque scaling"
    databook.Sheets(Sheets.Count).Range("T4") = "Position Gain Proportional"
    databook.Sheets(Sheets.Count).Range("T5") = "Position Gain Integral"
    databook.Sheets(Sheets.Count).Range("T6") = "Velocity Gain Proportional"
    databook.Sheets(Sheets.Count).Range("T7") = "Velocity Gain Integral"
    databook.Sheets(Sheets.Count).Range("T8") = "Velocity feedforward"
    databook.Sheets(Sheets.Count).Range("T9") = "Acceleration feedforward"

    'Doesn't download data if cancel has been pushed
    If Cancel = False Then
        ThisWorkbook.SendData "1", "Program:SetAxisProperties.GetGains,L1,C1"
        databook.Sheets(Sheets.Count).Range("U2") = ThisWorkbook.GetData("Program:SetAxisProperties.
            rLoadInertiaRatio,L1,C1")
        databook.Sheets(Sheets.Count).Range("U3") = ThisWorkbook.GetData("Program:SetAxisProperties.
            rTorqueScaling,L1,C1")
        databook.Sheets(Sheets.Count).Range("U4") = ThisWorkbook.GetData("Program:SetAxisProperties.
            rPositionProportionalGain,L1,C1")
        databook.Sheets(Sheets.Count).Range("U5") = ThisWorkbook.GetData("Program:SetAxisProperties.
            rPositionIntegralGain,L1,C1")
        databook.Sheets(Sheets.Count).Range("U6") = ThisWorkbook.GetData("Program:SetAxisProperties.
            rVelocityProportionalGain,L1,C1")
        databook.Sheets(Sheets.Count).Range("U7") = ThisWorkbook.GetData("Program:SetAxisProperties.
            rVelocityIntegralGain,L1,C1")
        databook.Sheets(Sheets.Count).Range("U8") = ThisWorkbook.GetData("Program:SetAxisProperties.
            rVelocityFeedforwardGain,L1,C1")
        databook.Sheets(Sheets.Count).Range("U9") = ThisWorkbook.GetData("Program:SetAxisProperties.
            rAccelerationFeedforwardGain,L1,C1")
    End If
    'Prints timestamps for data
    Do While ActiveCell.Offset(counter, -1) <> ""
        ActiveCell.Offset(counter, 0) = counter / 500
        counter = counter + 1
    Loop

    'Prints if cancel button has been used or operation was compleated
    If Cancel Then
        Upload_Label.Caption = "Upload canceled by user"
    Else
        Upload_Label.Caption = "Done"
    End If

    'Shutsdown connection with server
    Application.DDETerminate (DDEchan)

    databook.Sheets(Sheets.Count).Range("A:X").EntireColumn.AutoFit

    Exit Sub

End Sub
```

```
'Activates or deactivates drive
Private Sub Enable_Drive_Button_Click()
    'checks if drive is going to be diabled or enabled
    If Enable_Drive_Button.Caption = "Enable Drive" Then
        'Turns on the contactor
        ThisWorkbook.SendData "1", "Program:MainProgram.PowerOn,L1,C1"
        'Activates the servo
        ThisWorkbook.SendData "1", "Program:MainProgram.EnableMSO,L1,C1"
        Enable_Drive_Button.Caption = "Disable Drive"
    Else
        'Disables the servo
        ThisWorkbook.SendData "0", "Program:MainProgram.EnableMSO,L1,C1"
        'Turns off the contactor
        ThisWorkbook.SendData "1", "Program:MainProgram.PowerOff,L1,C1"
        Enable_Drive_Button.Caption = "Enable Drive"
    End If
End Sub


'Plots fft from data
Private Sub FFT_Button_Click()
    'creates variables
    Set interface = ThisWorkbook
    Dim counter As Integer
    Dim wBook As Variant
    Set wBook = Sheets("Control")
    Set databook = Workbooks(Workbook_Box.Text)

    'Control to make sure enough data is represented to make the fft
    If databook.Sheets(wBook.Pages_Box.Text).Range("A" & CStr(FFT_Box.value + 1)) = "" Then
        MsgBox ("Not enough values, change number of values or add more")
        Exit Sub
    End If

    'Activates sheet where fft is going to be calculated
    Application.Goto databook.Sheets(wBook.Pages_Box.Text).Range("A1")

    'Checks with ffts are going to be calculated and calculates them
    If Position_Box Then
        ActiveSheet.Range("H2:H5000") = ""
        Application.Run "ATPVBAEN.XLAM!Fourier", ActiveSheet.Range("$A$2:$A" & CStr(FFT_Box.value + 1)
            ), ActiveSheet.Range("$H$2:$H$" & CStr(FFT_Box.value + 1)), False, False
    End If
    If Velocity_Box Then
        ActiveSheet.Range("I2:I5000") = ""
        Application.Run "ATPVBAEN.XLAM!Fourier", ActiveSheet.Range("$B$2:$B" & CStr(FFT_Box.value + 1)
            ), ActiveSheet.Range("$I$2:$I$" & CStr(FFT_Box.value + 1)), False, False
    End If
    If Acceleration_Box Then
        ActiveSheet.Range("J2:J5000") = ""
        Application.Run "ATPVBAEN.XLAM!Fourier", ActiveSheet.Range("$C$2:$C" & CStr(FFT_Box.value + 1)
            ), ActiveSheet.Range("$J$2:$J$" & CStr(FFT_Box.value + 1)), False, False
    End If
    If Error_Box Then
        ActiveSheet.Range("K2:K5000") = ""
        Application.Run "ATPVBAEN.XLAM!Fourier", ActiveSheet.Range("$D$2:$D" & CStr(FFT_Box.value + 1)
            ), ActiveSheet.Range("$K$2:$K$" & CStr(FFT_Box.value + 1)), False, False
    End If
    If Torque_Box Then
        ActiveSheet.Range("L2:L5000") = ""
        Application.Run "ATPVBAEN.XLAM!Fourier", ActiveSheet.Range("$E$2:$E" & CStr(FFT_Box.value + 1)
            ), ActiveSheet.Range("$L$2:$L$" & CStr(FFT_Box.value + 1)), False, False
    End If

    'Activates cells where data is going to be presented
    Application.Goto databook.Sheets(wBook.Pages_Box.Text).Range("N2")

    'Sets labels for the data
    ActiveCell.Offset(-1, 0) = "Angle"
    ActiveCell.Offset(-1, 1) = "Velocity"
    ActiveCell.Offset(-1, 2) = "Acceleration"
    ActiveCell.Offset(-1, 3) = "Position Error"
    ActiveCell.Offset(-1, 4) = "Torque"

    'Resets counters
    counter = 0

    'Sets values to calculate X-axis frequencies
    Dim freq As Double
    Dim sampFreq As Double
```

```
'Calculates sample freq
sampFreq = FFT_Box.value / databook.Sheets(wBook.Pages_Box.Text).Range("F" & CStr(FFT_Box.value +
    1)).value
freq = 0

'Sets formula to sheet to calculate the absolute value of the fft
Do While counter < FFT_Box.value
    If Position_Box Then ActiveCell.Offset(counter, 0) = "=IMABS(H" & CStr(counter + 2) & ")"
    If Velocity_Box Then ActiveCell.Offset(counter, 1) = "=IMABS(I" & CStr(counter + 2) & ")"
    If Acceleration_Box Then ActiveCell.Offset(counter, 2) = "=IMABS(J" & CStr(counter + 2) & ")"
    If Error_Box Then ActiveCell.Offset(counter, 3) = "=IMABS(K" & CStr(counter + 2) & ")"
    If Torque_Box Then ActiveCell.Offset(counter, 4) = "=IMABS(L" & CStr(counter + 2) & ")"
    'Calculates frequencies for X-axis
    Sheets(wBook.Pages_Box.Text).Range("M" & CStr(counter + 2)) = freq
    freq = freq + sampFreq / FFT_Box.value
    counter = counter + 1
    DoEvents
Loop

'Adds new chart for the fft data
Charts.Add
ActiveChart.ChartType = xlXYScatterLines
ActiveChart.Move After:=Sheets(Sheets.Count)
counter = 1
'Adds series for the chart
If Position_Box Then 'Checks if position is going to be plotted
    If counter = 1 Then
        ActiveChart.SetSourceData Source:=Sheets(wBook.Pages_Box.Text).Range("N2:N2001"), PlotBy:=
            xlColumns
    Else
        ActiveChart.SeriesCollection.NewSeries
    End If
    ActiveChart.SeriesCollection(counter).values = "=" & CStr(wBook.Pages_Box.Text) & "!$N$2:$N$"
        & CStr(FFT_Box.value + 1)
    ActiveChart.SeriesCollection(counter).XValues = "=" & CStr(wBook.Pages_Box.Text) & "!$M:$M"
    ActiveChart.SeriesCollection(counter).Name = Sheets(wBook.Pages_Box.Text).Range("A1")

    ActiveChart.SeriesCollection(counter).Select
    With Selection.Border
        .Weight = xlThin
        .LineStyle = xlAutomatic
    End With
    With Selection
        .MarkerBackgroundColorIndex = xlAutomatic
        .MarkerForegroundColorIndex = xlAutomatic
        .MarkerStyle = xlNone
        .Smooth = False
        .MarkerSize = 5
        .Shadow = False
    End With
    counter = counter + 1
End If

If Velocity_Box Then 'Checks if velocity is going to be plotted
    If counter = 1 Then
        ActiveChart.SetSourceData Source:=Sheets(wBook.Pages_Box.Text).Range("N2:N2001"), PlotBy:=
            xlColumns
    Else
        ActiveChart.SeriesCollection.NewSeries
    End If

    ActiveChart.SeriesCollection(counter).values = "=" & CStr(wBook.Pages_Box.Text) & "!$O$2:$O$"
        & CStr(FFT_Box.value + 1)
    ActiveChart.SeriesCollection(counter).XValues = "=" & CStr(wBook.Pages_Box.Text) & "!$M:$M"
    ActiveChart.SeriesCollection(counter).Name = Sheets(wBook.Pages_Box.Text).Range("B1")

    ActiveChart.SeriesCollection(counter).Select
    With Selection.Border
        .Weight = xlThin
        .LineStyle = xlAutomatic
    End With
    With Selection
        .MarkerBackgroundColorIndex = xlAutomatic
        .MarkerForegroundColorIndex = xlAutomatic
        .MarkerStyle = xlNone
        .Smooth = False
        .MarkerSize = 5
        .Shadow = False
    End With
    counter = counter + 1
End If

If Acceleration_Box Then 'Checks if acceleration is going to be plotted
```

```
    If counter = 1 Then
        ActiveChart.SetSourceData Source:=Sheets(wBook.Pages_Box.Text).Range("N2:N2001"), PlotBy:=
            xlColumns
    Else
        ActiveChart.SeriesCollection.NewSeries
    End If

    ActiveChart.SeriesCollection(counter).values = "=" & CStr(wBook.Pages_Box.Text) & "!$P$2:$P$"
        & CStr(FFT_Box.value + 1)
    ActiveChart.SeriesCollection(counter).XValues = "=" & CStr(wBook.Pages_Box.Text) & "!$M:$M"
    ActiveChart.SeriesCollection(counter).Name = Sheets(wBook.Pages_Box.Text).Range("C1")

    ActiveChart.SeriesCollection(counter).Select
    With Selection.Border
        .Weight = xlThin
        .LineStyle = xlAutomatic
    End With
    With Selection
        .MarkerBackgroundColorIndex = xlAutomatic
        .MarkerForegroundColorIndex = xlAutomatic
        .MarkerStyle = xlNone
        .Smooth = False
        .MarkerSize = 5
        .Shadow = False
    End With
    counter = counter + 1
End If

If Error_Box Then 'Checks if position error is going to be plotted
    If counter = 1 Then
        ActiveChart.SetSourceData Source:=Sheets(wBook.Pages_Box.Text).Range("N2:N2001"), PlotBy:=
            xlColumns
    Else
        ActiveChart.SeriesCollection.NewSeries
    End If

    ActiveChart.SeriesCollection(counter).values = "=" & CStr(wBook.Pages_Box.Text) & "!$Q$2:$Q$"
        & CStr(FFT_Box.value + 1)
    ActiveChart.SeriesCollection(counter).XValues = "=" & CStr(wBook.Pages_Box.Text) & "!$M:$M"
    ActiveChart.SeriesCollection(counter).Name = Sheets(wBook.Pages_Box.Text).Range("D1")

    ActiveChart.SeriesCollection(counter).Select
    With Selection.Border
        .Weight = xlThin
        .LineStyle = xlAutomatic
    End With
    With Selection
        .MarkerBackgroundColorIndex = xlAutomatic
        .MarkerForegroundColorIndex = xlAutomatic
        .MarkerStyle = xlNone
        .Smooth = False
        .MarkerSize = 5
        .Shadow = False
    End With
    counter = counter + 1
End If

If Torque_Box Then 'Checks if torque is going to be plotted
    If counter = 1 Then
        ActiveChart.SetSourceData Source:=Sheets(wBook.Pages_Box.Text).Range("N2:N2001"), PlotBy:=
            xlColumns
    Else
        ActiveChart.SeriesCollection.NewSeries
    End If

    ActiveChart.SeriesCollection(counter).values = "=" & CStr(wBook.Pages_Box.Text) & "!$R$2:$R$"
        & CStr(FFT_Box.value + 1)
    ActiveChart.SeriesCollection(counter).XValues = "=" & CStr(wBook.Pages_Box.Text) & "!$M:$M"
    ActiveChart.SeriesCollection(counter).Name = Sheets(wBook.Pages_Box.Text).Range("E1")

    ActiveChart.SeriesCollection(counter).Select
    With Selection.Border
        .Weight = xlThin
        .LineStyle = xlAutomatic
    End With
    With Selection
        .MarkerBackgroundColorIndex = xlAutomatic
        .MarkerForegroundColorIndex = xlAutomatic
        .MarkerStyle = xlNone
        .Smooth = False
        .MarkerSize = 5
        .Shadow = False
    End With
```

```
            counter = counter + 1
        End If

        'Sets the properties for the axis
        ActiveChart.Axes(xlCategory).Select
        With ActiveChart.Axes(xlCategory)
            .MinimumScale = 0
            .MaximumScale = databook.Sheets(wBook.Pages_Box.Text).Range("M" & CStr(FFT_Box.value)).value /
                2
            .MinorUnitIsAuto = True
            .MajorUnitIsAuto = True
            .Crosses = xlAutomatic
            .ReversePlotOrder = False
            .ScaleType = xlLinear
            .DisplayUnit = xlNone
        End With

        ActiveChart.Axes(xlValue).Select
        With ActiveChart.Axes(xlValue)
            .MinimumScale = 0
            .MaximumScale = 500000
            .MinorUnitIsAuto = True
            .MajorUnitIsAuto = True
            .Crosses = xlAutomatic
            .ReversePlotOrder = False
            .ScaleType = xlLinear
            .DisplayUnit = xlNone
        End With

    End Sub

    'Performs a MAH on the Axis bringing it to 0 deg
    Private Sub Home_Button_Click()
        ThisWorkbook.SendData "1", "Program:MainProgram.EnableMAH,L1,C1"
    End Sub

    'Opens the form for simple motion commands
    Private Sub Move_Button_Click()
        Controls.Show
    End Sub

    'Updates the list with worksheets from specified workbook
    Private Sub Pages_Button_Click()
        'Sets parameters that are used
        Set interface = ThisWorkbook
        Dim counter As Integer
        Dim setFirst As Boolean
        Dim X As Variant
        Dim path As String

        If Path_Box = "" Then
            path = ThisWorkbook.path
        Else
            path = Path_Box.Text
        End If

        'Resets parameters
        setFirst = True
        counter = 1
        Pages_Box.Clear

        If Workbook_Box.Text = "" Then
            MsgBox "Workbook field empty"
            Exit Sub
        End If

        If Dir(path & "\" & Workbook_Box.Text & ".xlsx") <> "" Then
            Dim i As Long
            For i = Workbooks.Count To 1 Step -1
                If Workbooks(i).Name = Workbook_Box.Text & ".xlsx" Then
                    Set wBook = Workbooks(i)
                    Exit For
                End If
            Next
            If i = 0 Then
                Set wBook = Workbooks.Open(Filename:=path & "\" & Workbook_Box.Text, UpdateLinks:=0)
            End If

        ElseIf Dir(path & "\" & Workbook_Box.Text) <> "" Then
            Dim n As Long
            For n = Workbooks.Count To 1 Step -1
                If Workbooks(n).Name = Workbook_Box.Text Then
                    Set wBook = Workbooks(n)
```

```
                Exit For
            End If
        Next
        If n = 0 Then
            Set wBook = Workbooks.Open(Filename:=path & "\" & Workbook_Box.Text, UpdateLinks:=0)
        End If

    Else
        MsgBox "Workbook not found."
        Exit Sub
    End If

    Application.Goto interface.Sheets("Control").Cells(1, 1)

    'Searches the specified workbook for worksheets, if found it's added to the combobox
    For Each X In wBook.Sheets
        If TypeName(X) = "Worksheet" Then
            On Error GoTo sheetsHasNoName
            Pages_Box.AddItem X.Name
            On Error GoTo 0
            'Sets the first sheet found to default
            If setFirst Then
                Pages_Box.Text = X.Name
                setFirst = False
            End If
        End If

    Next
    Application.Goto interface.Sheets("Control").Cells(1, 1)
    Exit Sub

sheetsHasNoName:
    MsgBox "Sheets does not exist"
    On Error GoTo 0
    Exit Sub

End Sub



'Plots graf for selected data
Private Sub Plot_Button_Click()
    Set interface = ThisWorkbook
    Set wBook = Sheets("Control")
    'Sets variables for the function
    Dim counter As Integer
    Dim nbrOfData As Integer

    'Sets the number of datapoints to be plotted
    nbrOfData = Plot_Number_Box.value
    counter = 0


    'Checks if selected workbook and workssheet exists. If not code jumps to corresponding error
        handler at the end of this function
    On Error Resume Next
    Set databook = Workbooks(Workbook_Box.Text)
    If databook Is Nothing Then
        On Error GoTo errBook
        Workbooks.Open Workbook_Box.Text
    End If
    Set databook = Workbooks(Workbook_Box.Text)
    On Error GoTo errHandler
    Application.Goto databook.Sheets(wBook.Pages_Box.Text).Range("A1")
    On Error GoTo 0


    'Adds new chart and type and puts it last in workbook
    Charts.Add
    ActiveChart.Move After:=Sheets(Sheets.Count)
    ActiveChart.ChartType = xlXYScatterLines

    'Code to delete all series in the graf that Office 2007 so kindly puts there without permission
    Dim vGraf1 As Variant
    For Each vGraf1 In ActiveChart.SeriesCollection
        ActiveChart.SeriesCollection(1).Delete
    Next

    'Checks which boxes that are choosen and plots the data accordingly

    'Checks if position is going to be used
    If wBook.Position_Box Then
    counter = counter + 1
```

```
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(counter).Name = "=""Position"""
ActiveChart.SeriesCollection(counter).XValues = "=" & wBook.Pages_Box & "!R2C" & GetXValues() & ":
    R" & CStr(nbrOfData) & "C" & GetXValues()
ActiveChart.SeriesCollection(counter).values = "=" & wBook.Pages_Box & "!R2C1:R" & CStr(nbrOfData)
    & "C1"
ActiveChart.SeriesCollection(counter).MarkerStyle = xlNone
End If

'Checks if velocity is going to be used
If wBook.Velocity_Box Then
counter = counter + 1
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(counter).Name = "=""Velocity"""
ActiveChart.SeriesCollection(counter).XValues = "=" & wBook.Pages_Box & "!R2C" & GetXValues() & ":
    R" & CStr(nbrOfData) & "C" & GetXValues()
ActiveChart.SeriesCollection(counter).values = "=" & wBook.Pages_Box & "!R2C2:R" & CStr(nbrOfData)
    & "C2"
ActiveChart.SeriesCollection(counter).MarkerStyle = xlNone
End If

'Checks if Acceleration is going to be used
If wBook.Acceleration_Box Then
counter = counter + 1
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(counter).Name = "=""Acceleration"""
ActiveChart.SeriesCollection(counter).XValues = "=" & wBook.Pages_Box & "!R2C" & GetXValues() & ":
    R" & CStr(nbrOfData) & "C" & GetXValues()
ActiveChart.SeriesCollection(counter).values = "=" & wBook.Pages_Box & "!R2C3:R" & CStr(nbrOfData)
    & "C3"
ActiveChart.SeriesCollection(counter).MarkerStyle = xlNone
End If

'Checks if position error is going to be used
If wBook.Error_Box Then
counter = counter + 1
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(counter).Name = "=""Position Error"""
ActiveChart.SeriesCollection(counter).XValues = "=" & wBook.Pages_Box & "!R2C" & GetXValues() & ":
    R" & CStr(nbrOfData) & "C" & GetXValues()
ActiveChart.SeriesCollection(counter).values = "=" & wBook.Pages_Box & "!R2C4:R" & CStr(nbrOfData)
    & "C4"
ActiveChart.SeriesCollection(counter).MarkerStyle = xlNone
End If

'Checks if torque is going to be used
If wBook.Torque_Box Then
counter = counter + 1
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(counter).Name = "=""Torque"""
ActiveChart.SeriesCollection(counter).XValues = "=" & wBook.Pages_Box & "!R2C" & GetXValues() & ":
    R" & CStr(nbrOfData) & "C" & GetXValues()
ActiveChart.SeriesCollection(counter).values = "=" & wBook.Pages_Box & "!R2C5:R" & CStr(nbrOfData)
    & "C5"
ActiveChart.SeriesCollection(counter).MarkerStyle = xlNone
End If

'Checks if position reference is going to be used
If wBook.Reference_Box Then
counter = counter + 1
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(counter).Name = "=""Reference Position"""
ActiveChart.SeriesCollection(counter).XValues = "=" & wBook.Pages_Box & "!R2C" & GetXValues() & ":
    R" & CStr(nbrOfData) & "C" & GetXValues()
ActiveChart.SeriesCollection(counter).values = "=" & wBook.Pages_Box & "!R2C7:R" & CStr(nbrOfData)
    & "C7"
ActiveChart.SeriesCollection(counter).MarkerStyle = xlNone
End If

'Sets a label to the graf, if not empty
If counter > 0 Then
    With ActiveChart
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = GetXLabel()
    End With
End If

'Sometimes an additional series is created that contains a 1, this function delets that series
Dim vGraf As Variant
For Each vGraf In ActiveChart.SeriesCollection
    If vGraf.Name = "Series" & CStr(counter + 1) Then ActiveChart.SeriesCollection("Series" & CStr
        (counter + 1)).Delete
Next
```

```
    Exit Sub

    'Errorhandler if book dosn't exist
errBook:
    MsgBox "Book not found"
    Exit Sub
    'Errorhandler if sheet dosn't exist
errHandler:
    MsgBox "Sheet not found"
    Exit Sub

End Sub

'Returns what data that is going to be used on the X-axis. Called by Plot_Button_Click()
Function GetXValues() As String

    Set wBook = interface.Sheets("Control")
    If (wBook.Position_Plot.value) Then
        GetXValues = "1"
    ElseIf wBook.Velocity_Plot Then
        GetXValues = "2"
    ElseIf wBook.Acceleration_Plot Then
        GetXValues = "3"
    ElseIf wBook.Error_Plot Then
        GetXValues = "4"
    ElseIf wBook.Torque_Plot Then
        GetXValues = "5"
    ElseIf wBook.Time_Plot Then
        GetXValues = "6"
    End If
End Function

'Returns the label of the X-axis. Called by Plot_Button_Click()
Function GetXLabel() As String
    Set wBook = interface.Sheets("Control")
    If (wBook.Position_Plot.value) Then
        GetXLabel = Position_Plot.Caption
    ElseIf wBook.Velocity_Plot Then
        GetXLabel = Velocity_Plot.Caption
    ElseIf wBook.Acceleration_Plot Then
        GetXLabel = Acceleration_Plot.Caption
    ElseIf wBook.Error_Plot Then
        GetXLabel = Error_Plot.Caption
    ElseIf wBook.Torque_Plot Then
        GetXLabel = Torque_Plot.Caption
    ElseIf wBook.Time_Plot Then
        GetXLabel = Time_Plot.Caption
    End If
End Function

'Starts the cam and data collection
Private Sub Start_Button_Click()
    'Starts cam with no delay, max 10ms delay, or max 25ms delay. Simulates large programs
    If Noperiod_Option Then ThisWorkbook.SendData "1", "Program:MainProgram.bCam1,L1,C1"
    If Shortperiod_Option Then ThisWorkbook.SendData "1", "Program:MainProgram.bCam10ms,L1,C1"
    If Longperiod_Option Then ThisWorkbook.SendData "1", "Program:MainProgram.bCam25ms,L1,C1"
End Sub

'Downloads the cam to the PLC
Private Sub Download_CAM_Button_Click()
    'Declares varibels to be used
    Set interface = ThisWorkbook
    Dim chosenCam As Integer
    Dim camRowCount As Integer
    Dim counter As Integer
    Dim DDEchan As Long
    Dim value As Variant

    If Coupling1 Then
        ThisWorkbook.SendData "200", "Program:MainProgram.rPeakTorque,L1,C1"
        ThisWorkbook.SendData "1", "Program:MainProgram.bSetTorque,L1,C1"
    ElseIf Coupling2 Then
        ThisWorkbook.SendData "150", "Program:MainProgram.rPeakTorque,L1,C1"
        ThisWorkbook.SendData "1", "Program:MainProgram.bSetTorque,L1,C1"
    Else
        MsgBox "Choose coupling"
        Exit Sub
    End If

    Coupling1.value = False
    Coupling2.value = False
```

```
    'Empty the cam in the plc
    counter = EmptyCAM()

    'resets counter
    counter = 0

    'Returns colum for chosen cam
    chosenCam = GetCAM()

    'Returns number of values of cam
    camRowCount = GetCAMRows(chosenCam)
    CAM_Points_Label.Caption = "Number of points in CAM: " & CStr(camRowCount)

    'Opens link to OPC server
    DDEchan = Application.DDEInitiate("RSLinx", Topic_Box.Text)

    'Activates cam sheet
    Application.Goto ActiveWorkbook.Sheets("CAMs").Cells(3, chosenCam)

    'Download cam to PLC
    Do While counter < camRowCount
        Application.DDEPoke DDEchan, "Program:MainProgram.cam1[" & CStr(counter) & "].Master,L1,C1",
            ActiveCell.Offset(counter, 1)
        Application.DDEPoke DDEchan, "Program:MainProgram.cam1[" & CStr(counter) & "].Slave,L1,C1",
            ActiveCell.Offset(counter, 2)
        If ActiveCell.Offset(counter, 3).Text = "Cubic" Then
            Range("AA99") = "1"
            Application.DDEPoke DDEchan, "Program:MainProgram.cam1[" & CStr(counter) & "].SegmentType,
                L1,C1", Range("AA99")
        Else
            Range("AA99") = "0"
            Application.DDEPoke DDEchan, "Program:MainProgram.cam1[" & CStr(counter) & "].SegmentType,
                L1,C1", Range("AA99")
        End If
        counter = counter + 1
    Loop
    'Activates first sheet in workbook
    Application.Goto ActiveWorkbook.Sheets("Control").Range("A1")

    'Sets length of cam in PLC
    Range("AA99") = camRowCount
    Application.DDEPoke DDEchan, "Program:MainProgram.dCam1Length1,L1,C1", Range("AA99")

    'Sets timescale in PLC
    Range("AA99") = Time_Scale_Box.value
    Application.DDEPoke DDEchan, "Program:MainProgram.TimeScaleCam1,L1,C1", Range("AA99")

    'Sets distancescale in PLC
    Range("AA99") = Distance_Scale_Box.value
    Application.DDEPoke DDEchan, "Program:MainProgram.DistScaleCam1,L1,C1", Range("AA99")

    'Calculates the cam
    Range("AA99") = "1"
    Application.DDEPoke DDEchan, "Program:MainProgram.bCalcCam,L1,C1", Range("AA99")

    'Terminates DDE connection
    Application.DDETerminate DDEchan

    If camRowCount > 200 Then
        MsgBox "CAM is to long, maximum 200, current " & CStr(camRowCount)
    End If

    Range("B1") = "=RSLinx|" & Topic_Box.Text & "!'Axis1.DriveFault,L1,C1'"

End Sub


'Returns the column for the chosen cam
Function GetCAM() As Integer
    'Creates a counter
    Dim counter As Integer
    'Sets teh counter
    counter = 1

    'Checks the first 1000 colums in the CAMs sheet for the selected CAM
    Do While counter < 1000
        If CAM_Box.Text = Sheets("CAMs").Cells(1, counter) Then
            Exit Do
        End If
        counter = counter + 1

    Loop
```

```
    'Returns the colum the CAM was found in
    GetCAM = counter
    If counter > 999 Then
        MsgBox ("CAM not found")
    End If

End Function

'Returns number of values in chosen cam
Function GetCAMRows(cam As Integer) As Integer
    Dim counter As Integer
    counter = 0

    'Checks number of CAM points
    Application.Goto ActiveWorkbook.Sheets("CAMs").Cells(3, cam)
    Do While ActiveCell.Offset(counter, 0) <> ""
        counter = counter + 1
    Loop
    GetCAMRows = counter
End Function

'Empties the cam in PLC
Function EmptyCAM() As Integer
    Dim counter As Integer
    Dim DDEchan As Long
    Dim value As Variant

    'Sets value to 0
    Sheets("Control").Range("AA99") = "0"
    Set value = Sheets("Control").Range("AA99")
    counter = 0

    'Opens a channel to RSlinx
    DDEchan = Application.DDEInitiate("RSLinx", Topic_Box.Text)

    'Sets the all 200 values in the cam to 0
    Do While counter < 200
        Application.DDEPoke DDEchan, "Program:MainProgram.cam1[" & CStr(counter) & "].Slave,L1,C1",
            value
        Application.DDEPoke DDEchan, "Program:MainProgram.cam1[" & CStr(counter) & "].Master,L1,C1",
            value
        Application.DDEPoke DDEchan, "Program:MainProgram.cam1[" & CStr(counter) & "].SegmentType,L1,
            C1", value
        counter = counter + 1
    Loop

    'Closes the DDE connection
    Application.DDETerminate DDEchan

    'Returns 0 when done
    EmptyCAM = 0
End Function

'Stops all motion
Private Sub Stop_Button_Click()
    'Stops Axis1 and AxisVirtual
    ThisWorkbook.SendData "1", "Program:MainProgram.EnableMAS,L1,C1"
End Sub
```

G.2 Tuning

```
        'Performs an auto tune
Private Sub Auto_Tune_Button_Click()
    ThisWorkbook.SendData "1", "Program:MainProgram.EnableAutoTune,L1,C1"
End Sub

'Uploads current tuning settings
Private Sub Get_Values_Button_Click()
    Sheets("Tuning").Range("B2") = ThisWorkbook.GetData("Program:MainProgram.rDistanceScale,L1,C1")
    Sheets("Tuning").Range("B3") = ThisWorkbook.GetData("Program:MainProgram.rTimeScale,L1,C1")
    Sheets("Tuning").Range("B6") = ThisWorkbook.GetData("Program:MainProgram.TuneDistFWD,L1,C1")
    Sheets("Tuning").Range("B7") = ThisWorkbook.GetData("Program:MainProgram.TuneSpeed,L1,C1")
    Sheets("Tuning").Range("B8") = ThisWorkbook.GetData("Program:MainProgram.TuneACC,L1,C1")
    Sheets("Tuning").Range("B9") = ThisWorkbook.GetData("Program:MainProgram.TuneJerk,L1,C1")
    ThisWorkbook.SendData "1", "Program:SetAxisProperties.GetGains,L1,C1"
    Sheets("Tuning").Range("B13") = ThisWorkbook.GetData("Program:SetAxisProperties.rLoadInertiaRatio,
        L1,C1")
    Sheets("Tuning").Range("B14") = ThisWorkbook.GetData("Program:SetAxisProperties.rTorqueScaling,L1,
        C1")
    Sheets("Tuning").Range("B16") = ThisWorkbook.GetData("Program:SetAxisProperties.
        rPositionProportionalGain,L1,C1")
    Sheets("Tuning").Range("B17") = ThisWorkbook.GetData("Program:SetAxisProperties.
        rPositionIntegralGain,L1,C1")
    Sheets("Tuning").Range("B19") = ThisWorkbook.GetData("Program:SetAxisProperties.
        rVelocityProportionalGain,L1,C1")
    Sheets("Tuning").Range("B20") = ThisWorkbook.GetData("Program:SetAxisProperties.
        rVelocityIntegralGain,L1,C1")
    Sheets("Tuning").Range("B22") = ThisWorkbook.GetData("Program:SetAxisProperties.
        rVelocityFeedforwardGain,L1,C1")
    Sheets("Tuning").Range("B23") = ThisWorkbook.GetData("Program:SetAxisProperties.
        rAccelerationFeedforwardGain,L1,C1")
    Sheets("Tuning").Range("B25") = ThisWorkbook.GetData("Program:SetAxisProperties.
        rPositionErrorTolerance,L1,C1")
End Sub


'Sets the chartproperties
Private Sub Graf_Button_Click()

    ActiveSheet.ChartObjects("Chart 1").Activate
    ActiveChart.SeriesCollection("Torque").values = "='Tuning'!$N$2:$N$" & CStr(X_Axis_Box)
    ActiveChart.SeriesCollection("Velocity").values = "='Tuning'!$O$2:$O$" & CStr(X_Axis_Box)
    ActiveChart.SeriesCollection("Position error").values = "='Tuning'!$P$2:$P$" & CStr(X_Axis_Box)
    If Torque_Primary Then
        ActiveChart.SeriesCollection("Torque").AxisGroup = 1
    ElseIf Torque_Secondary Then
        ActiveChart.SeriesCollection("Torque").AxisGroup = 2
    End If
    If Velocity_Primary Then
        ActiveChart.SeriesCollection("Velocity").AxisGroup = 1
    ElseIf Velocity_Secondary Then
        ActiveChart.SeriesCollection("Velocity").AxisGroup = 2
    End If
    If Error_Primary Then
        ActiveChart.SeriesCollection("Position Error").AxisGroup = 1
    ElseIf Error_Secondary Then
        ActiveChart.SeriesCollection("Position Error").AxisGroup = 2
    End If

    ActiveChart.Axes(xlValue, xlPrimary).MaximumScale = Primary_Max_Box.value
    ActiveChart.Axes(xlValue, xlPrimary).MinimumScale = Primary_Min_Box.value
    On Error Resume Next
    ActiveChart.Axes(xlValue, xlSecondary).MaximumScale = Secondary_Max_Box.value
    ActiveChart.Axes(xlValue, xlSecondary).MinimumScale = Secondary_Min_Box.value
    On Error GoTo 0


End Sub

'Downloads the tuning values to the PLC
Private Sub Set_Values_Button_Click()
    ThisWorkbook.SendData Sheets("Tuning").Range("B2"), "Program:MainProgram.rDistanceScale,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B3"), "Program:MainProgram.rTimeScale,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B6"), "Program:MainProgram.TuneDistFWD,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B7"), "Program:MainProgram.TuneSpeed,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B8"), "Program:MainProgram.TuneACC,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B9"), "Program:MainProgram.TuneJerk,L1,C1"

    Range("B14") = TorqueScaling(Sheets("Tuning").Range("B13").value)
    ThisWorkbook.SendData Sheets("Tuning").Range("B13"), "Program:SetAxisProperties.rLoadInertiaRatio,
        L1,C1"
```

```
    ThisWorkbook.SendData Sheets("Tuning").Range("B14"), "Program:SetAxisProperties.rTorqueScaling,L1,
        C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B16"), "Program:SetAxisProperties.
        rPositionProportionalGain,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B17"), "Program:SetAxisProperties.
        rPositionIntegralGain,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B19"), "Program:SetAxisProperties.
        rVelocityProportionalGain,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B20"), "Program:SetAxisProperties.
        rVelocityIntegralGain,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B22"), "Program:SetAxisProperties.
        rVelocityFeedforwardGain,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B23"), "Program:SetAxisProperties.
        rAccelerationFeedforwardGain,L1,C1"
    ThisWorkbook.SendData Sheets("Tuning").Range("B25"), "Program:SetAxisProperties.
        rPositionErrorTolerance,L1,C1"
    ThisWorkbook.SendData "1", "Program:SetAxisProperties.SetGains,L1,C1"
End Sub

'Downloads the settings and runs the tuning then uploads the data
Private Sub Start_Position_Tuning_Button_Click()
    Call Set_Values_Button_Click
    ThisWorkbook.SendData "1", "Program:MainProgram.EnableTunePos,L1,C1"
    Application.OnTime Now + TimeValue("00:00:03"), "Upload_Values" 'Calls the Upload Values from
        module Upload with one second delay
End Sub

'Starts and stops the velocity tuning
Private Sub Start_Velocity_Tuning_Button_Click()
    If Start_Velocity_Tuning_Button.Caption = "Start Velocity Tuning" Then
        ThisWorkbook.SendData "1", "Program:MainProgram.EnableTuneVel,L1,C1"
        Start_Velocity_Tuning_Button.Caption = "Stop Velocity Tuning"
    Else
        ThisWorkbook.SendData "0", "Program:MainProgram.EnableTuneVel,L1,C1"
        Start_Velocity_Tuning_Button.Caption = "Start Velocity Tuning"
    End If
End Sub

'Calculates the torqueScaling from inertia ratio
Function TorqueScaling(inertiaRatio As Double) As Double
    'Torque scaling with gearbox
    TorqueScaling = 0.000243 * (1 + inertiaRatio)
    'Torque scaling without gearbox
    'TorqueScaling = 5*0.000243 *(1+inertiaRatio)
End Function
```

G.3 Form

```
        'Populates the comboboxes when the form is opened
Private Sub Userform_Initialize()
    MAM_Box.Text = "Abs"
    MAM_Box.AddItem "Abs"
    MAM_Box.AddItem "Inc"
    MAM_Box.AddItem "Rot shortest"
    MAM_Box.AddItem "Pos rot"
    MAM_Box.AddItem "Neg rot"
    MAM_Box.AddItem "Abs master offset"
    MAM_Box.AddItem "Inc master offest"

    MAJ_Box.Text = "Forward"
    MAJ_Box.AddItem "Forward"
    MAJ_Box.AddItem "Reverse"

End Sub

'Closes the form
Private Sub Close_Button_Click()
    Unload Me
End Sub

'Starts the sampling
Private Sub Collect_Button_Click()
    SendData 1, "Program:MainProgram.CollectMain,L1,C1"
    Collect_Label.Caption = "Collecting" 'Changes the caption of the label
    Application.OnTime Now + TimeValue("00:00:11"), "CollectComplete" 'Sets up a deley when the label
        should change
End Sub

'Downloads given settings to the PLC
Private Sub Download_Button_Click()
    If MAM_Box.Text = "Abs" Then SendData "0", "Program:MainProgram.dMAMmovetype,L1,C1"
    If MAM_Box.Text = "Inc" Then SendData "1", "Program:MainProgram.dMAMmovetype,L1,C1"
    If MAM_Box.Text = "Rot shortest" Then SendData "2", "Program:MainProgram.dMAMmovetype,L1,C1"
    If MAM_Box.Text = "Pos rot" Then SendData "3", "Program:MainProgram.dMAMmovetype,L1,C1"
    If MAM_Box.Text = "Neg rot" Then SendData "4", "Program:MainProgram.dMAMmovetype,L1,C1"
    If MAM_Box.Text = "Abs master offset" Then SendData "5", "Program:MainProgram.dMAMmovetype,L1,C1"
    If MAM_Box.Text = "Inc master offest" Then SendData "6", "Program:MainProgram.dMAMmovetype,L1,C1"

    SendData MAM_Position_Box, "Program:MainProgram.rMAMposition,L1,C1"
    SendData MAM_Speed_Box, "Program:MainProgram.rMAMspeed,L1,C1"
    SendData MAM_Acc_Box, "Program:MainProgram.rMAMacc,L1,C1"
    SendData MAM_Jerk_Box, "Program:MainProgram.rMAMjerk,L1,C1"

    If MAJ_Box.Text = "Forward" Then SendData "0", "Program:MainProgram.dMAJdirection,L1,C1"
    If MAJ_Box.Text = "Reverse" Then SendData "1", "Program:MainProgram.dMAJdirection,L1,C1"

    SendData MAJ_Speed_Box, "Program:MainProgram.rMAJspeed,L1,C1"
    SendData MAJ_Acc_Box, "Program:MainProgram.rMAJacc,L1,C1"
    SendData MAJ_Jerk_Box, "Program:MainProgram.rMAJjerk,L1,C1"

End Sub

'Preforms a homing
Private Sub MAH_Button_Click()
    SendData 1, "Program:MainProgram.EnableMAH,L1,C1"
End Sub

'Performs a JOG
Private Sub MAJ_Button_Click()
    SendData 1, "Program:MainProgram.bEnableMAJ,L1,C1"
End Sub

'Performs a move
Private Sub MAM_Button_Click()
    SendData 1, "Program:MainProgram.bEnableMAM,L1,C1"
End Sub

'Stop all motion
Private Sub MAS_Button_Click()
    SendData 1, "Program:MainProgram.EnableMAS,L1,C1"
End Sub

'Powers the servo
Private Sub Power_Button_Click()
    If Power_Button.Caption = "Power ON" Then
        'Turns on the contactor
        ThisWorkbook.SendData "1", "Program:MainProgram.PowerOn,L1,C1"
        'Activates the servo
        ThisWorkbook.SendData "1", "Program:MainProgram.EnableMSO,L1,C1"
```

```
        Power_Button.Caption = "Power OFF"
    Else
        'Disables the servo
        ThisWorkbook.SendData "0", "Program:MainProgram.EnableMSO,L1,C1"
        'Turns off the contactor
        ThisWorkbook.SendData "1", "Program:MainProgram.PowerOff,L1,C1"
        Power_Button.Caption = "Power ON"
    End If
End Sub

'Sends the data to the PLC
Function SendData(data As Variant, Item As String)
    Dim DDEchan As Long
    'Fix data to be the correct format
    Sheets("Control").Range("AA99") = data
    Set data = Sheets("Control").Range("AA99")
    'Open DDE channel to RSLinx, Topic Emulate
    DDEchan = Application.DDEInitiate("RSLinx", TopicName.Text)
    'Send data
    Application.DDEPoke DDEchan, Item, data
    'Close DDE connection
    Application.DDETerminate (DDEchan)
    Exit Function
End Function

'Gets data from the PLC
Function GetData(Item As String) As Variant
    Dim DDEchan As Long
    Dim value As Variant
    'Open DDE channel
    DDEchan = Application.DDEInitiate("RSLinx", TopicName.Text)
    'Get data
    GetData = Application.DDERequest(DDEchan, Item)
    'Close Connection
    Application.DDETerminate (DDEchan)
    Exit Function
End Function
```

G.4 Module

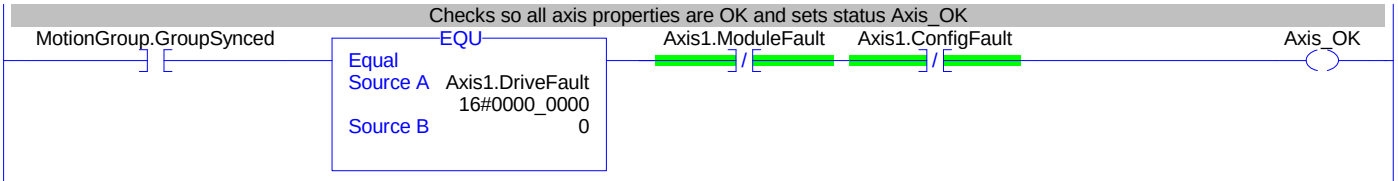
```
        'Function called by tuning position loop
Function Upload_Values() As Integer
    Dim counter As Integer
    counter = 0
    Range("N2:P5001") = ""
    Range("I6") = "Cancel"
    'Opens connection to OPC-server
    DDEchan = Application.DDEInitiate("RSLinx", Sheet1.Topic_Box.Text)
    Do While counter < Sheets("Tuning").Range("I4").value
        Sheets("Tuning").Range("I5") = counter + 1
        'Upload 100 values at once from first array
        If Sheets("Tuning").Torque_Box Then Sheets("Tuning").Range("N" & CStr(counter * 100 + 2) & ":N
            " & CStr(counter * 100 + 101)) = Application.DDERequest(DDEchan, "arrData[4," & CStr(
            counter * 100) & "],L100,C1")
        DoEvents
        If ActiveCell.Text = "Cancel" Then
            Exit Function
        End If
        If Sheets("Tuning").Velocity_Box Then Sheets("Tuning").Range("O" & CStr(counter * 100 + 2) &
            ":O" & CStr(counter * 100 + 101)) = Application.DDERequest(DDEchan, "arrData[1," & CStr(
            counter * 100) & "],L100,C1")
        DoEvents
        If ActiveCell.Text = "Cancel" Then
            Exit Function
        End If
        If Sheets("Tuning").Position_Box Then Sheets("Tuning").Range("P" & CStr(counter * 100 + 2) &
            ":P" & CStr(counter * 100 + 101)) = Application.DDERequest(DDEchan, "arrData[3," & CStr(
            counter * 100) & "],L100,C1")
        DoEvents
        If ActiveCell.Text = "Cancel" Then
            Exit Function
        End If
        counter = counter + 1

    Loop
    Range("I6") = "Done"
    'Shutsdown connection with server
    Application.DDETerminate (DDEchan)
End Function

'Function called by start data collect in axis move form
Function CollectComplete() As Integer
    Controls.Collect_Label.Caption = "Collect complete"
End Function
```

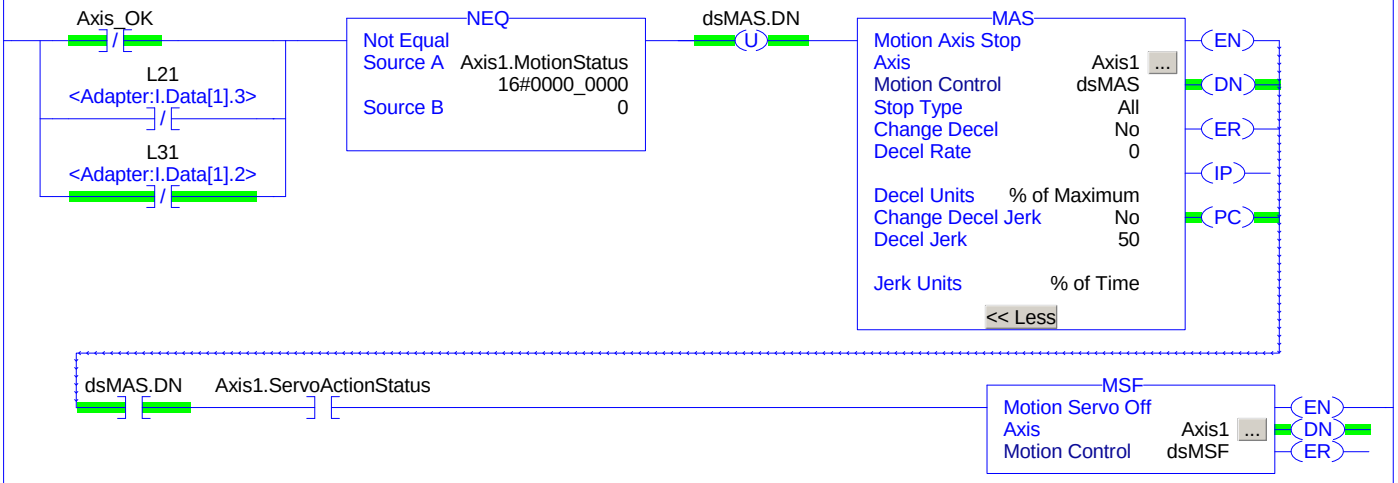
Checks so all axis properties are OK and sets status Axis_OK

0



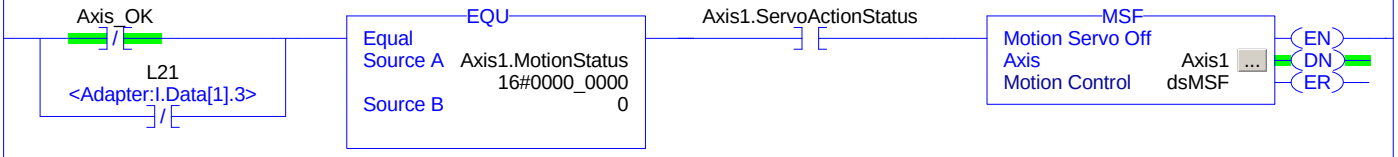
If Axis is not OK or the power goes down on L21 or L31 all motion is stopped

1



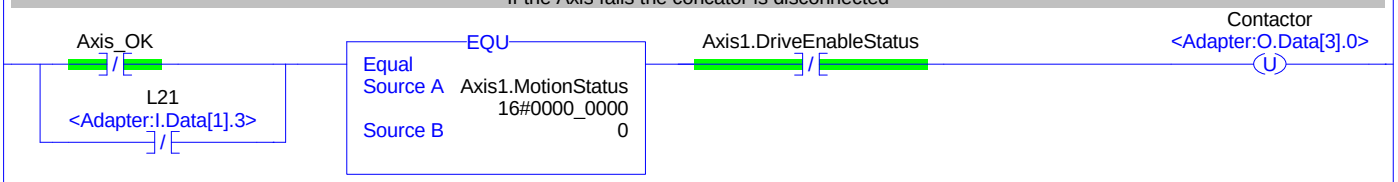
If the Axis fails the servodrive shuts down

2



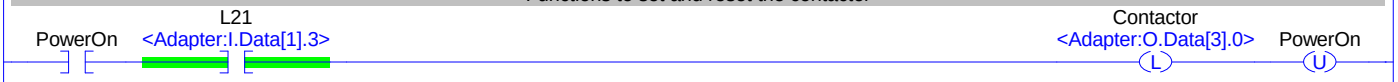
If the Axis fails the concator is disconnected

3



Functions to set and reset the contactor

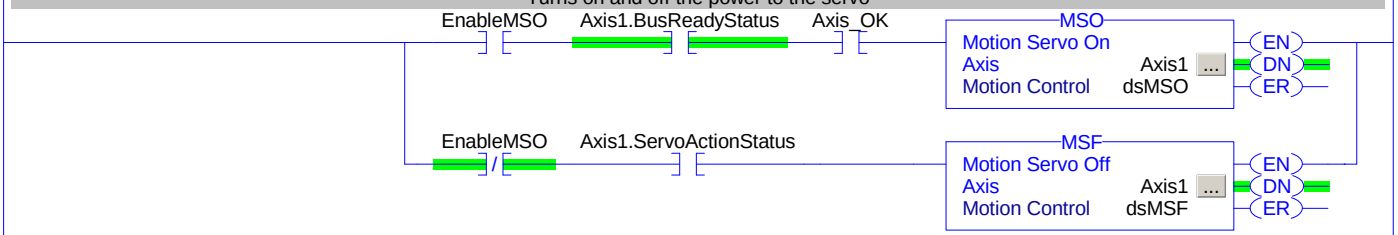
4



5

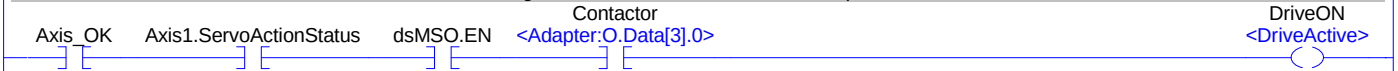
Turns on and off the power to the servo

6



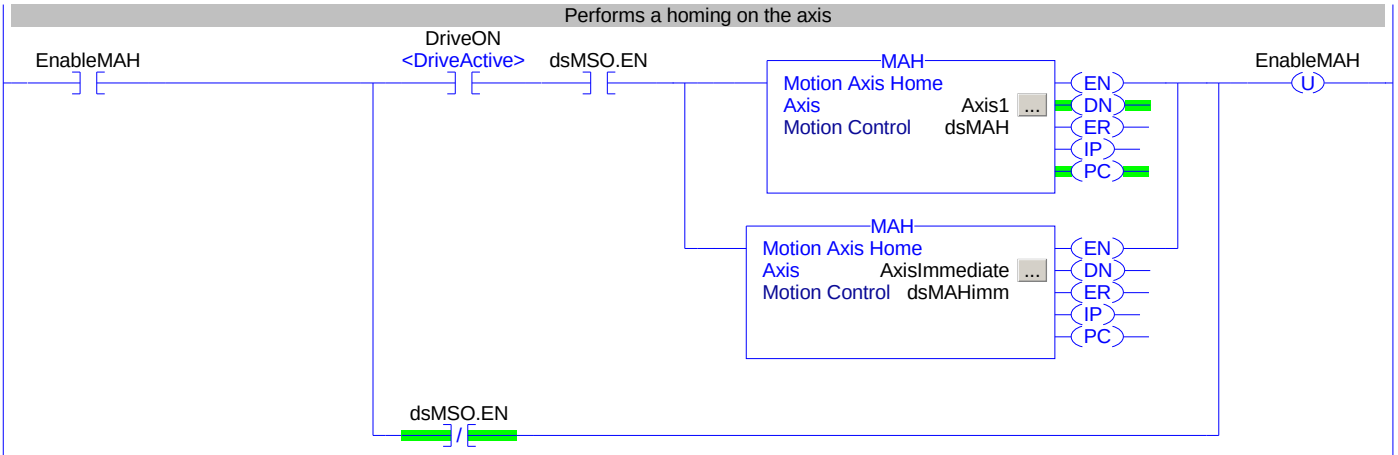
Sets the DriveON flag if Axis is OK and the contactor and power to the servo is on

7



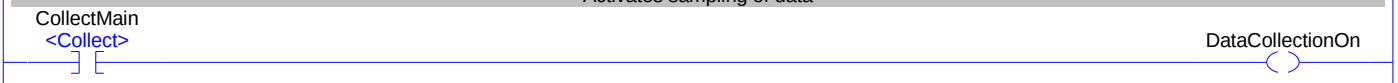
Performs a homing on the axis

8



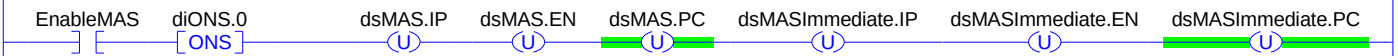
Activates sampling of data

9

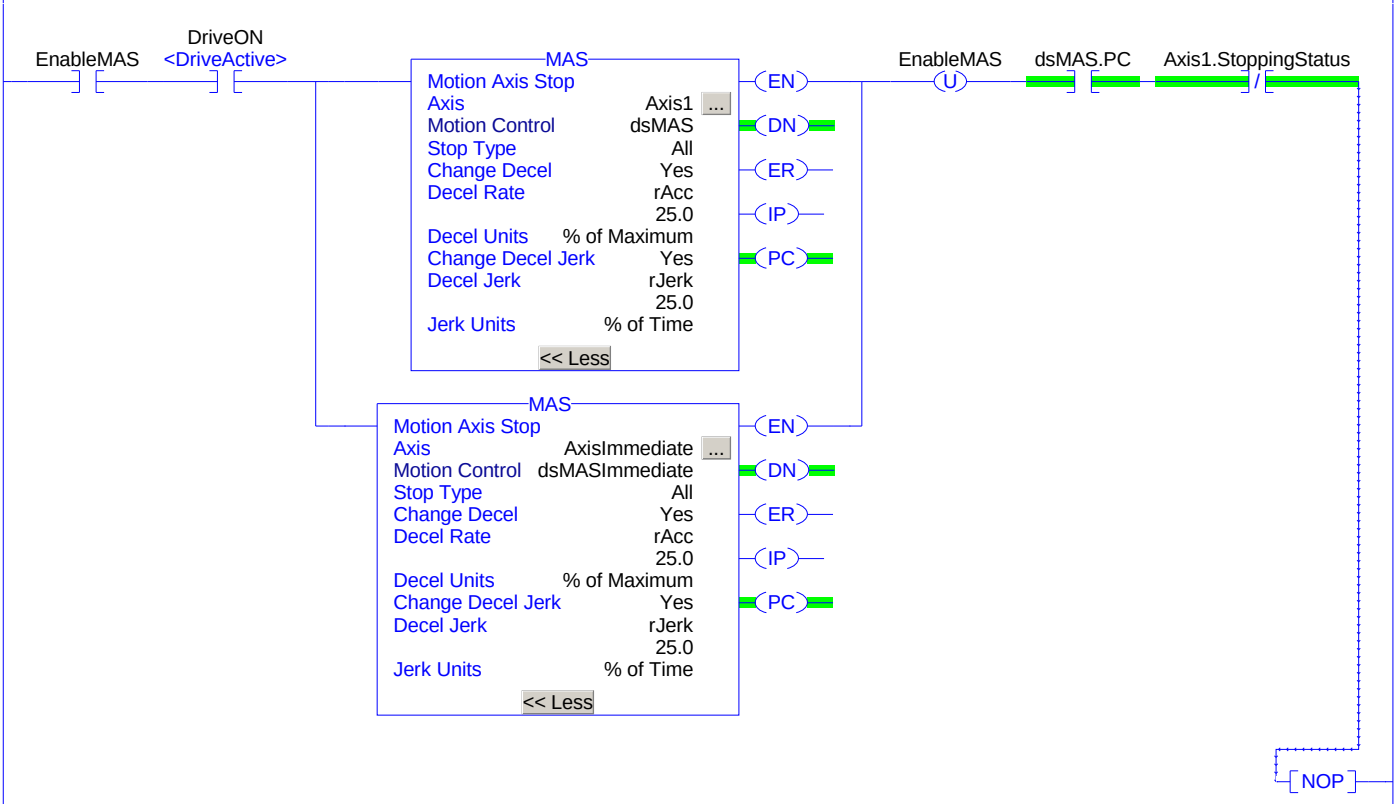


Stops all motion

10

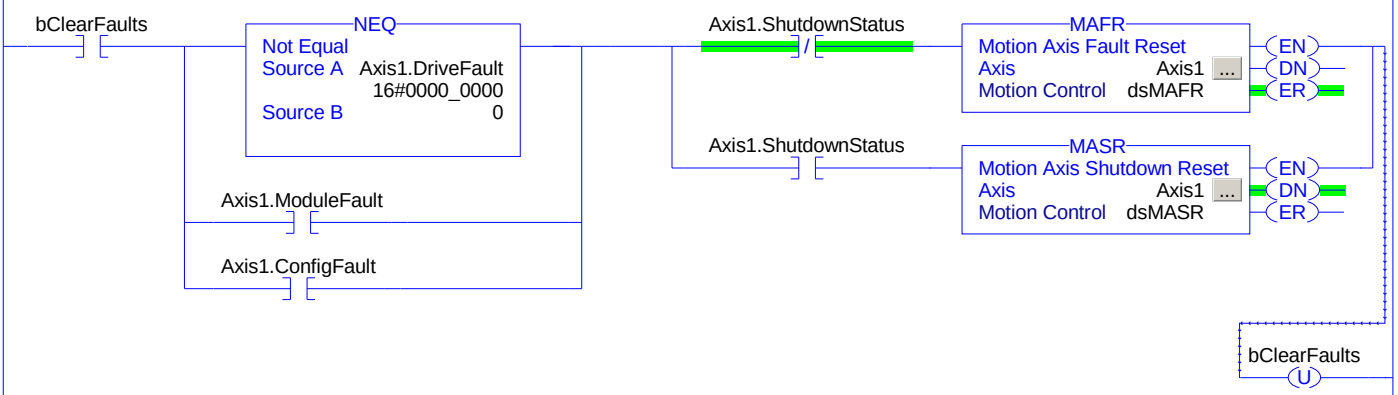


11



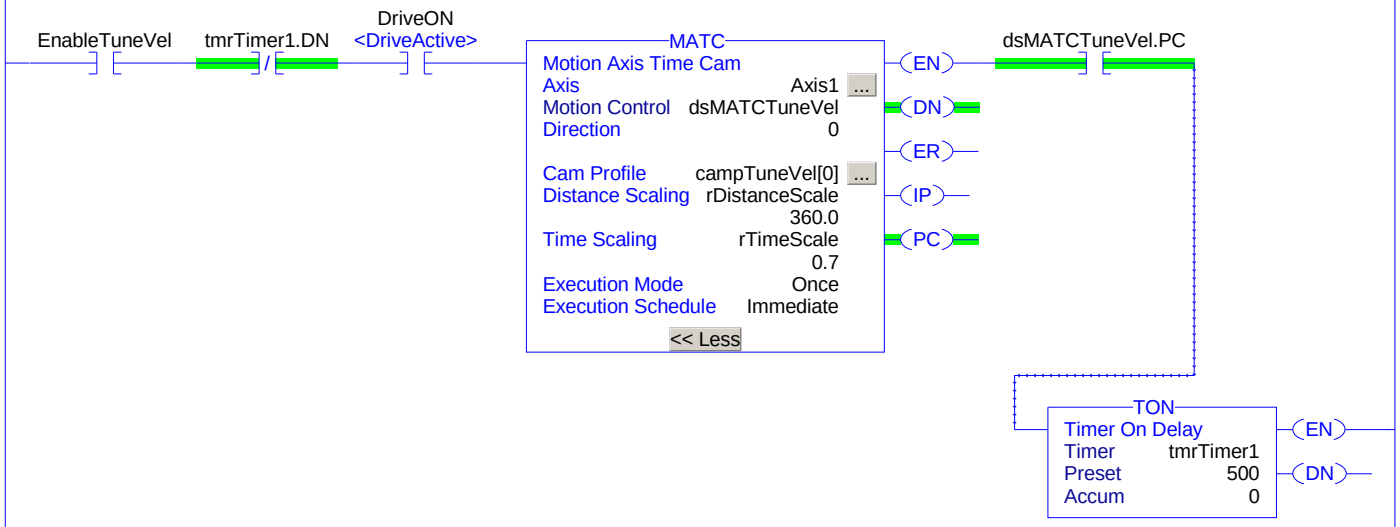
Function to reset drive faults

12



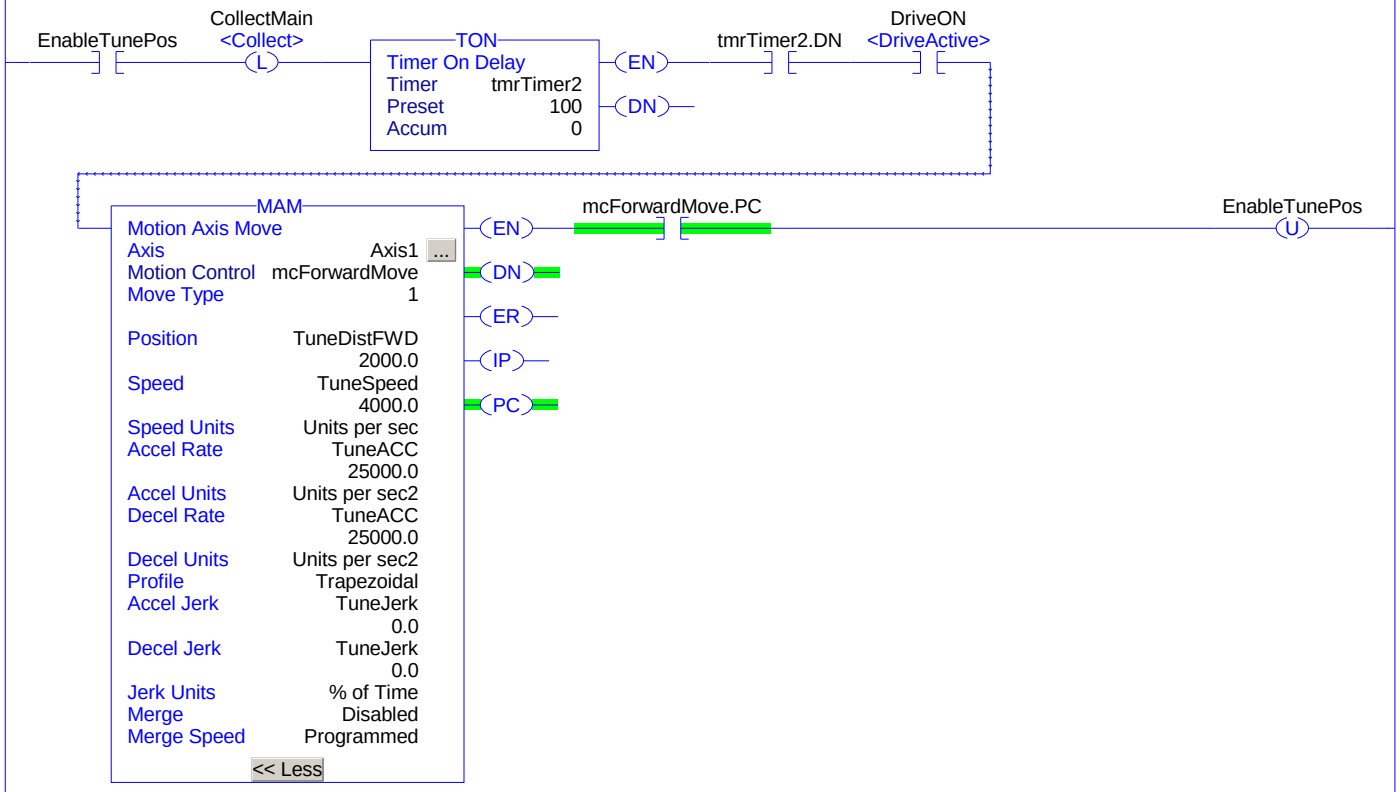
Cam used for tuning the velocity loop

13



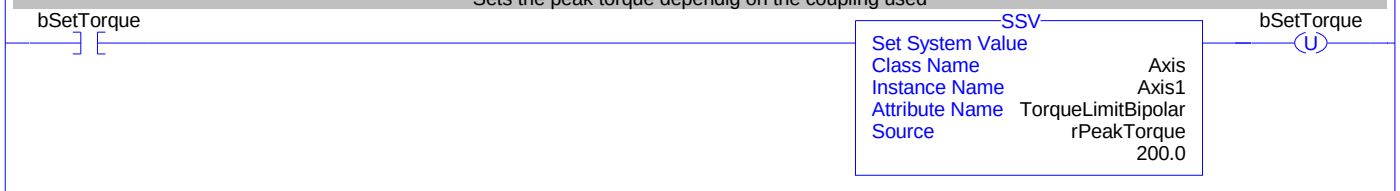
Instruction for tuning the position loop

14



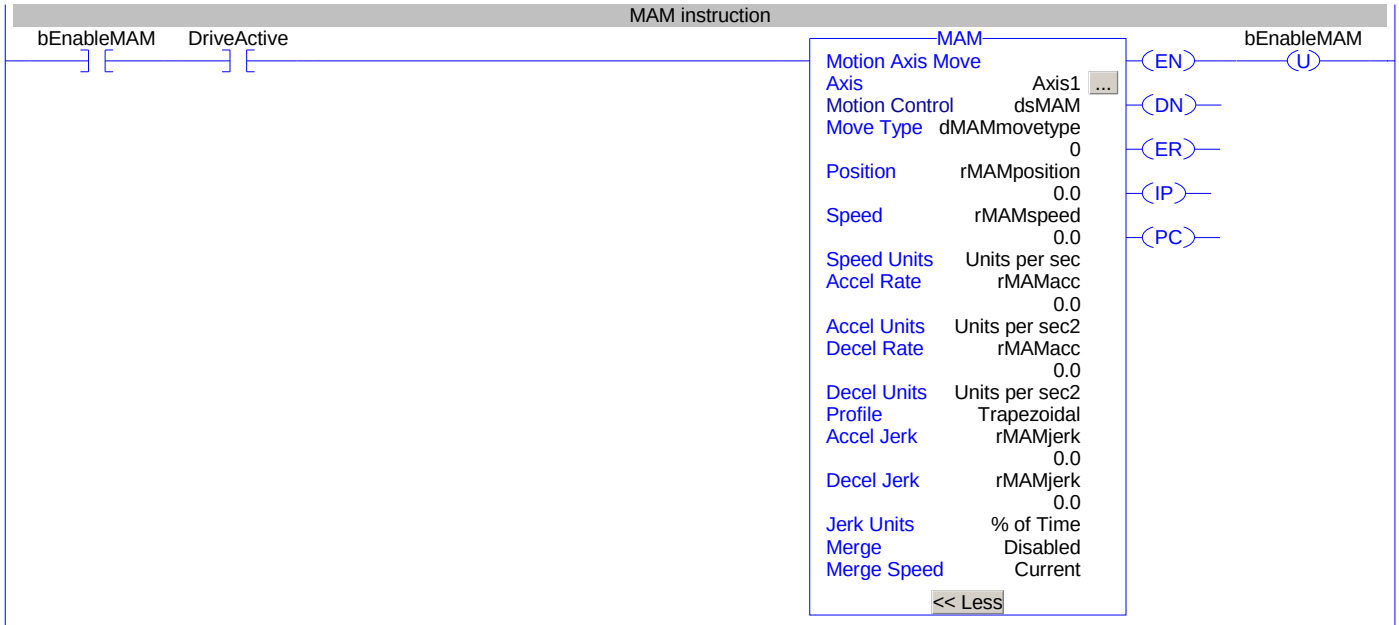
Sets the peak torque dependig on the coupling used

15



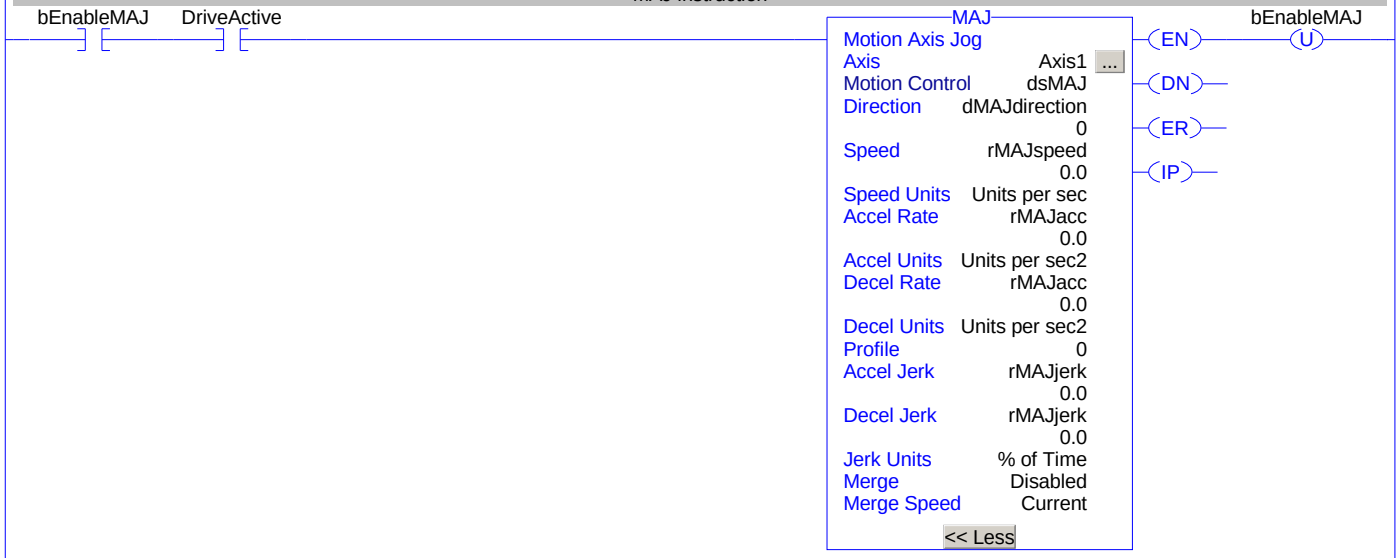
MAM instruction

16



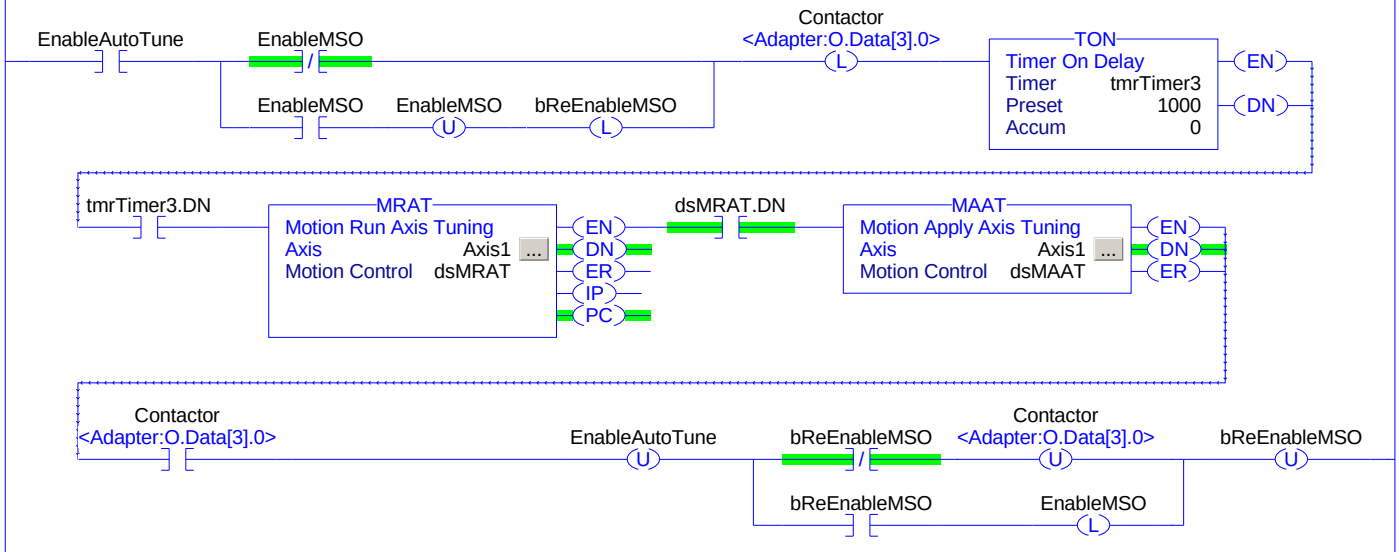
MAJ instruction

17



Auto tune and sets axis parameters

18



Instructions to run a masters axis for MAPC purposes

19 bStopMaster

MAS
Motion Axis Stop
Axis AxisVirtual
Motion Control dsMASmaster
Stop Type All
Change Decel No
Decel Rate rDecMaster
100
Decel Units % of Maximum
Change Decel Jerk No
Decel Jerk rDecJerkMaster
100
Jerk Units % of Maximum
<< Less
EN DN ER IP PC

20 bStopMaster /

MAJ
Motion Axis Jog
Axis AxisVirtual
Motion Control dsJOGvirtual
Direction 0
Speed dVirtualSpeed
10000
Speed Units Units per sec
Accel Rate dVirtualAcc
100
Accel Units % of Maximum
Decel Rate dVirtualAcc
100
Decel Units % of Maximum
Profile Trapezoidal
Accel Jerk dVirtualJerk
0
Decel Jerk dVirtualJerk
0
Jerk Units % of Time
Merge Disabled
Merge Speed Current
<< Less
EN DN ER IP

Calculates the downloaded CAM

21 bCalcCam dsMCCPcalccam.EN(U) bCamCalculated(U)

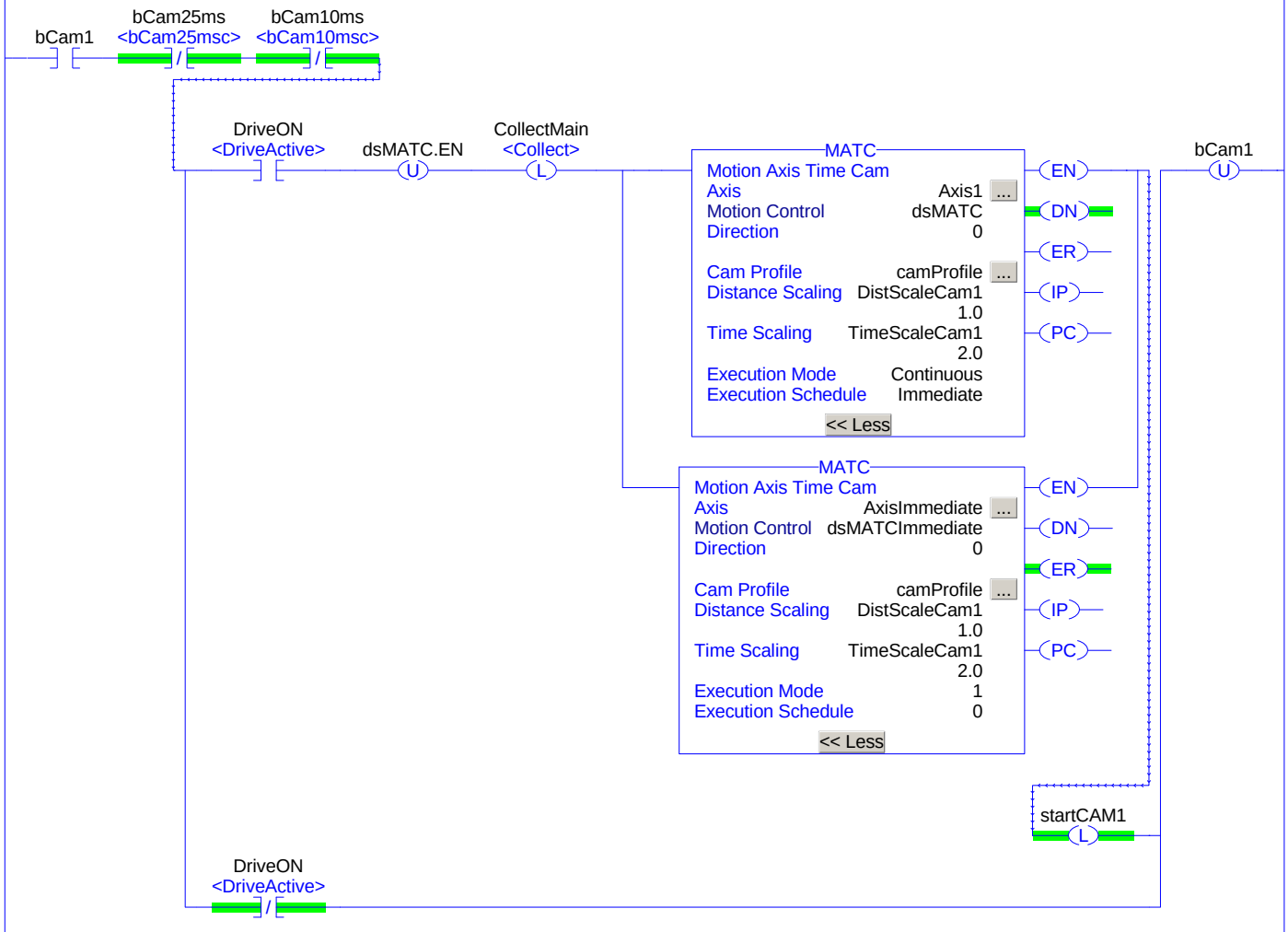
MCCP
Motion Calculate Cam Profile
Motion Control dsMCCPcalccam
Cam cam1
Length dCam1Length1
61
Start Slope rStartSlope
0
End Slope rEndSlope
0
Cam Profile camProfile
EN DN ER

dsMCCPcalccam.DN

bCamCalculated(L) bCalcCam(U)

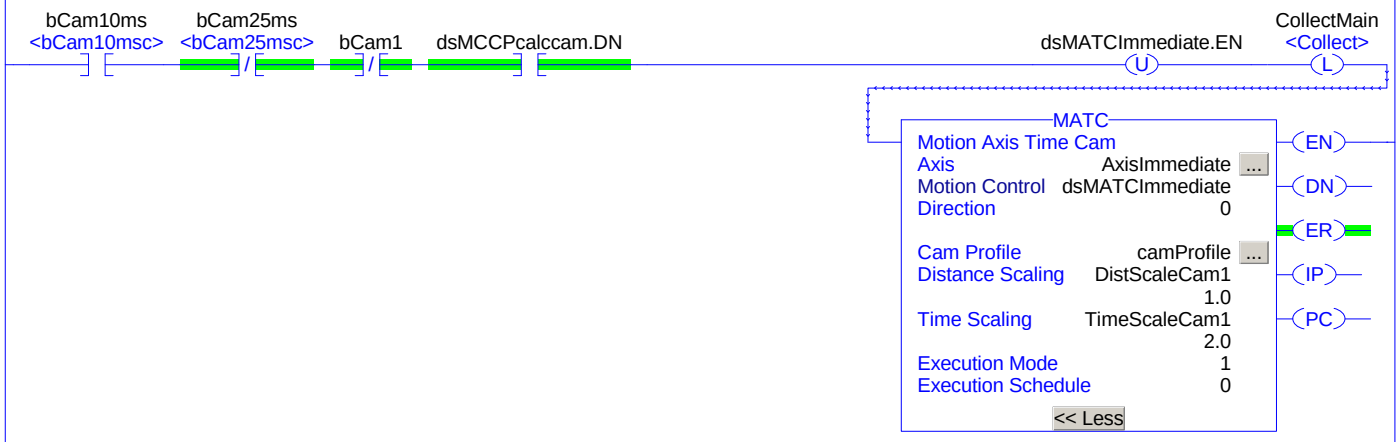
Instruction for running the cam with no delay

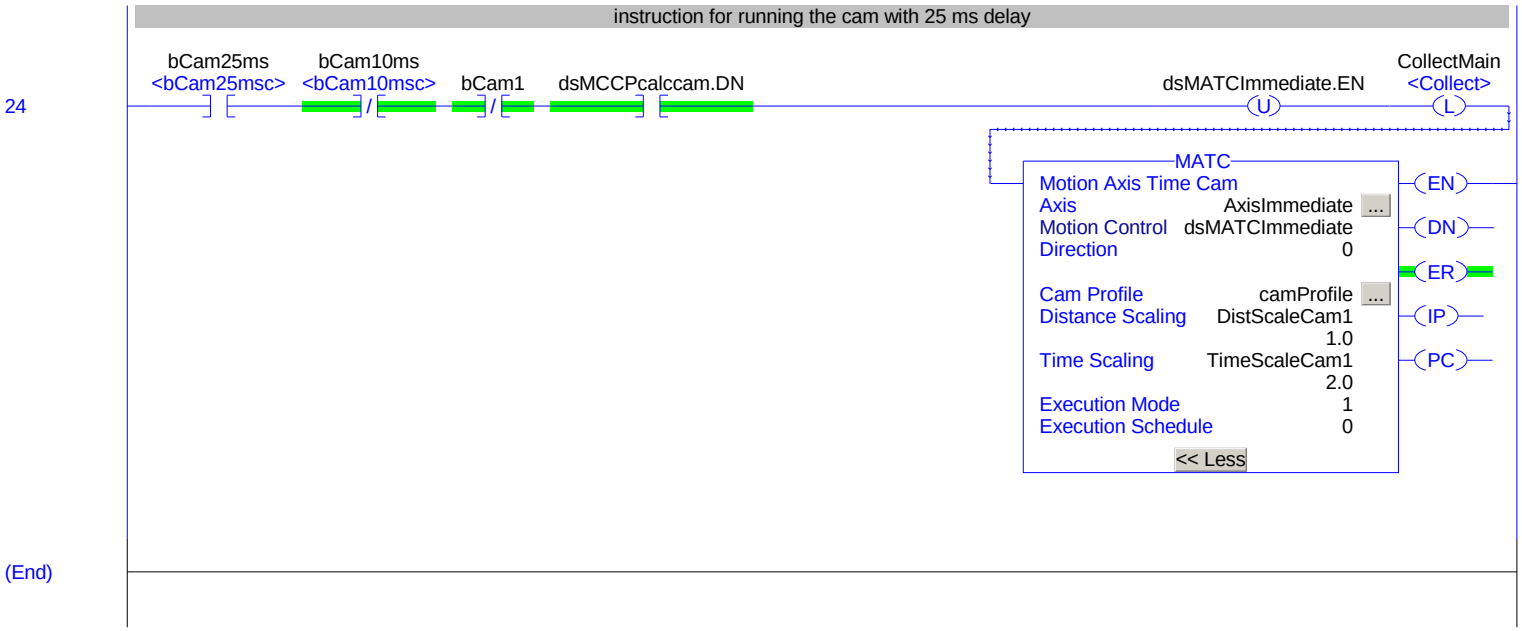
22

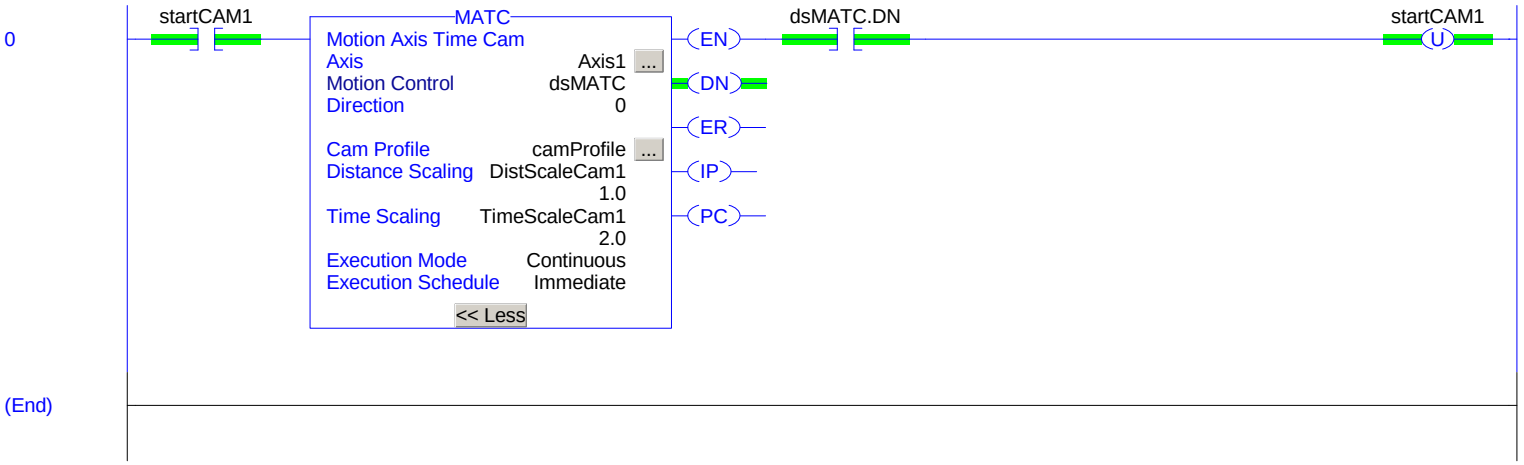


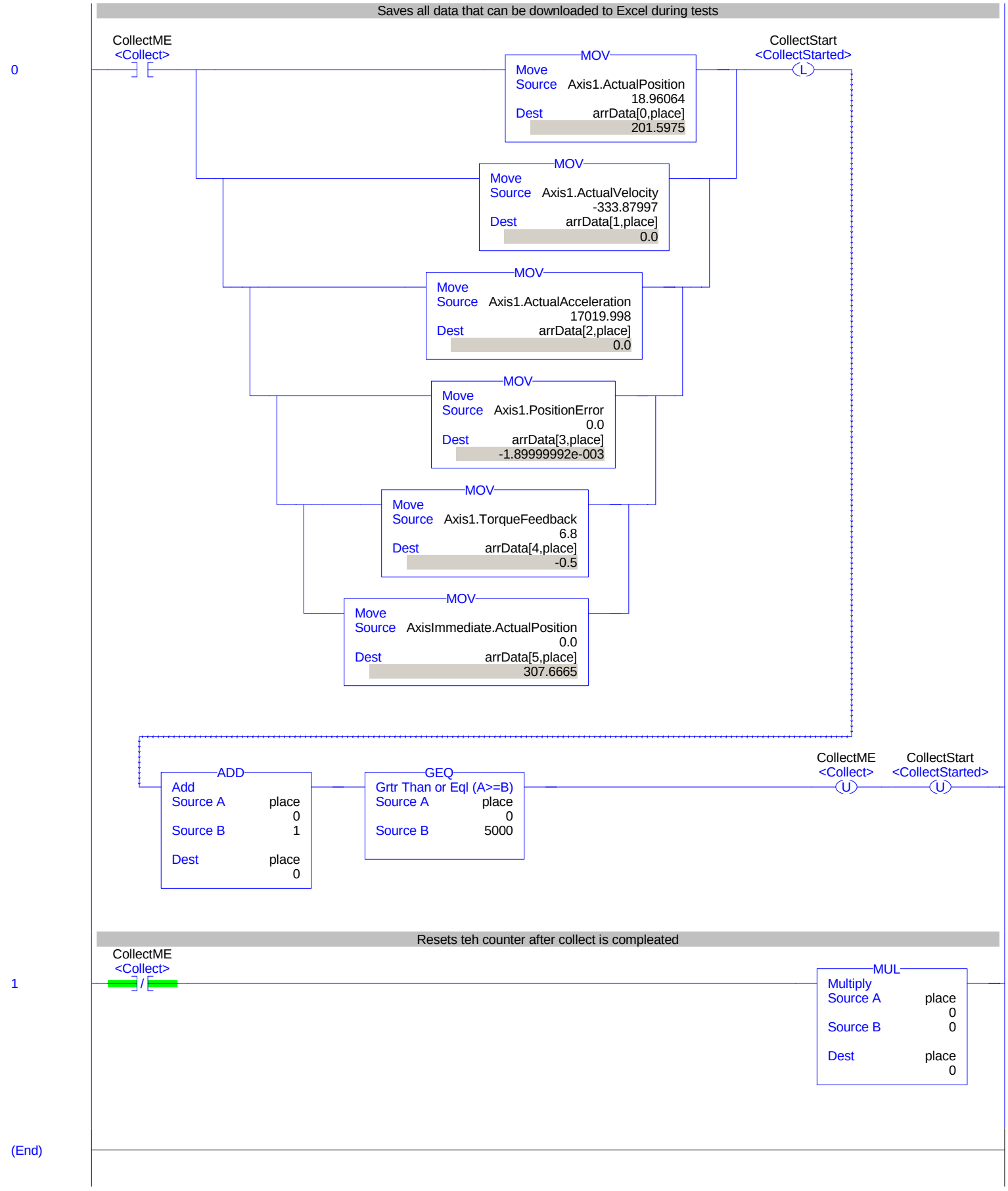
instruction for running the cam with 10ms delay

23




instruction for running the cam with 25 ms delay
bCam25ms
<bCam25msc>
bCam10ms
<bCam10msc>
bCam1
dsMCCPcalccam.DN
dsMATCImmediate.EN
U
CollectMain
<Collect>
L
24
MATC
Motion Axis Time Cam
Axis AxisImmediate
Motion Control dsMATCImmediate
Direction 0
Cam Profile camProfile
Distance Scaling DistScaleCam1
1.0
Time Scaling TimeScaleCam1
2.0
Execution Mode 1
Execution Schedule 0
<< Less
EN
DN
ER
IP
PC
(End)


0
startCAM1
MATC
Motion Axis Time Cam
Axis Axis1
Motion Control dsMATC
Direction 0
Cam Profile camProfile
Distance Scaling DistScaleCam1
1.0
Time Scaling TimeScaleCam1
2.0
Execution Mode Continuous
Execution Schedule Immediate
<< Less
EN
DN
ER
IP
PC
dsMATC.DN
startCAM1
U
(End)


Saves all data that can be downloaded to Excel during tests
CollectME
<Collect>
CollectStart
<CollectStarted>
0
MOV
Move
Source Axis1.ActualPosition
18.96064
Dest arrData[0,place]
201.5975
MOV
Move
Source Axis1.ActualVelocity
-333.87997
Dest arrData[1,place]
0.0
MOV
Move
Source Axis1.ActualAcceleration
17019.998
Dest arrData[2,place]
0.0
MOV
Move
Source Axis1.PositionError
0.0
Dest arrData[3,place]
-1.89999992e-003
MOV
Move
Source Axis1.TorqueFeedback
6.8
Dest arrData[4,place]
-0.5
MOV
Move
Source AxisImmediate.ActualPosition
0.0
Dest arrData[5,place]
307.6665
ADD
Add
Source A place
0
Source B 1
Dest place
0
GEQ
Grtr Than or Eql (A>=B)
Source A place
0
Source B 5000
CollectME
<Collect>
U
CollectStart
<CollectStarted>
U
Resets teh counter after collect is compleated
CollectME
<Collect>
1
MUL
Multiply
Source A place
0
Source B 0
Dest place
0
(End)

Sets and gets systemproperties

0

SetProperties

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name PositionUnwind
Source dPositionUnwind
0

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name LoadInertiaRatio
Source rLoadInertiaRatio
4.397838

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name ConversionConstant
Source rConversionConstant
0.0

SetProperties
(U)

1

GetProperties

GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name PositionUnwind
Dest dPositionUnwind
0

GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name LoadInertiaRatio
Dest rLoadInertiaRatio
4.397838

GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name ConversionConstant
Dest rConversionConstant
0.0

GetProperties
(U)

2

SetGains

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name TorqueScaling
Source rTorqueScaling
1.31167460e-003

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name LoadInertiaRatio
Source rLoadInertiaRatio
4.397838

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name VelocityFeedforwardGain
Source rVelocityFeedforwardGain
100.7

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name AccelerationFeedforwardGain
Source rAccelerationFeedforwardGain
101.0

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name VelocityIntegralGain
Source rVelocityIntegralGain
0.0

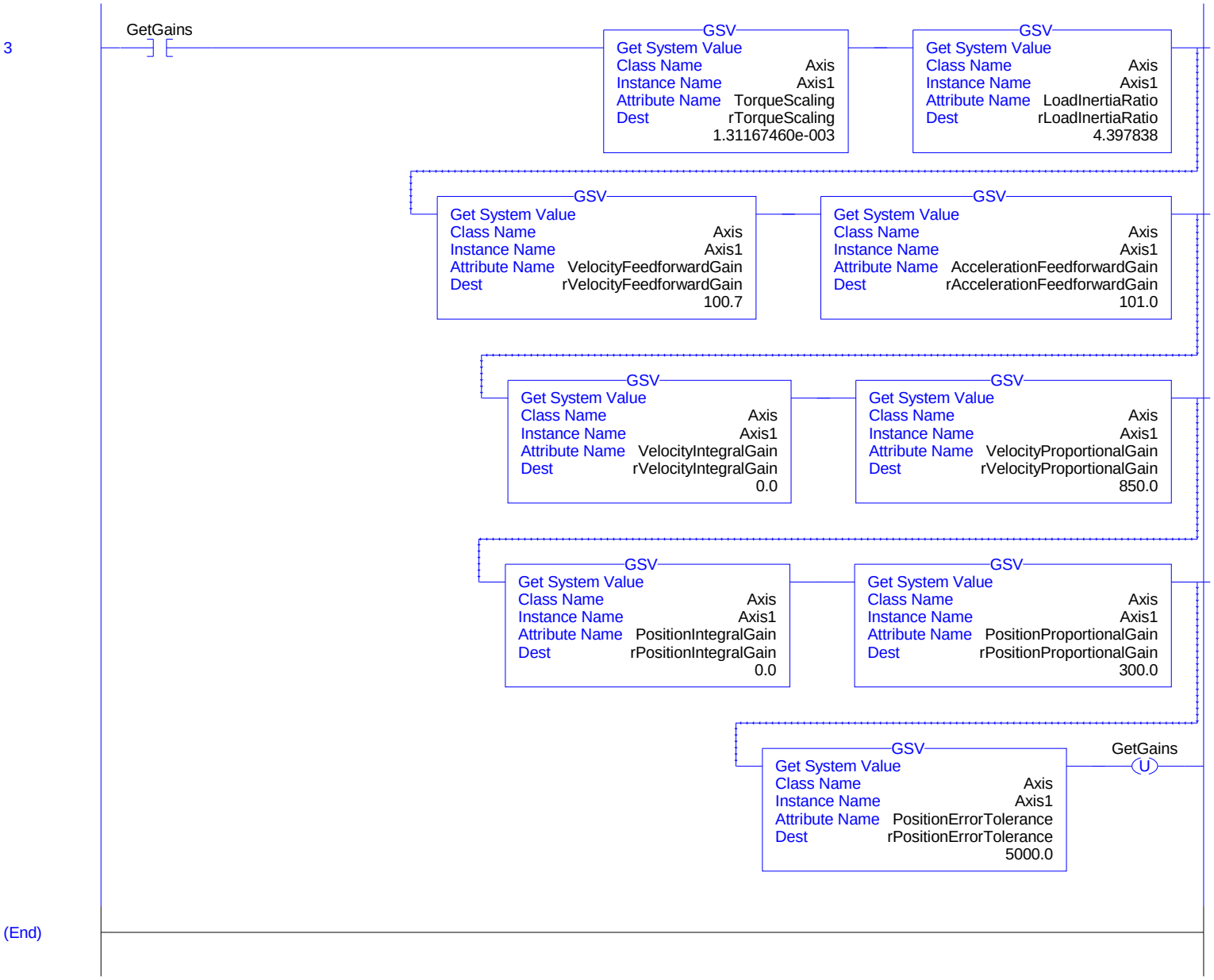
SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name VelocityProportionalGain
Source rVelocityProportionalGain
850.0

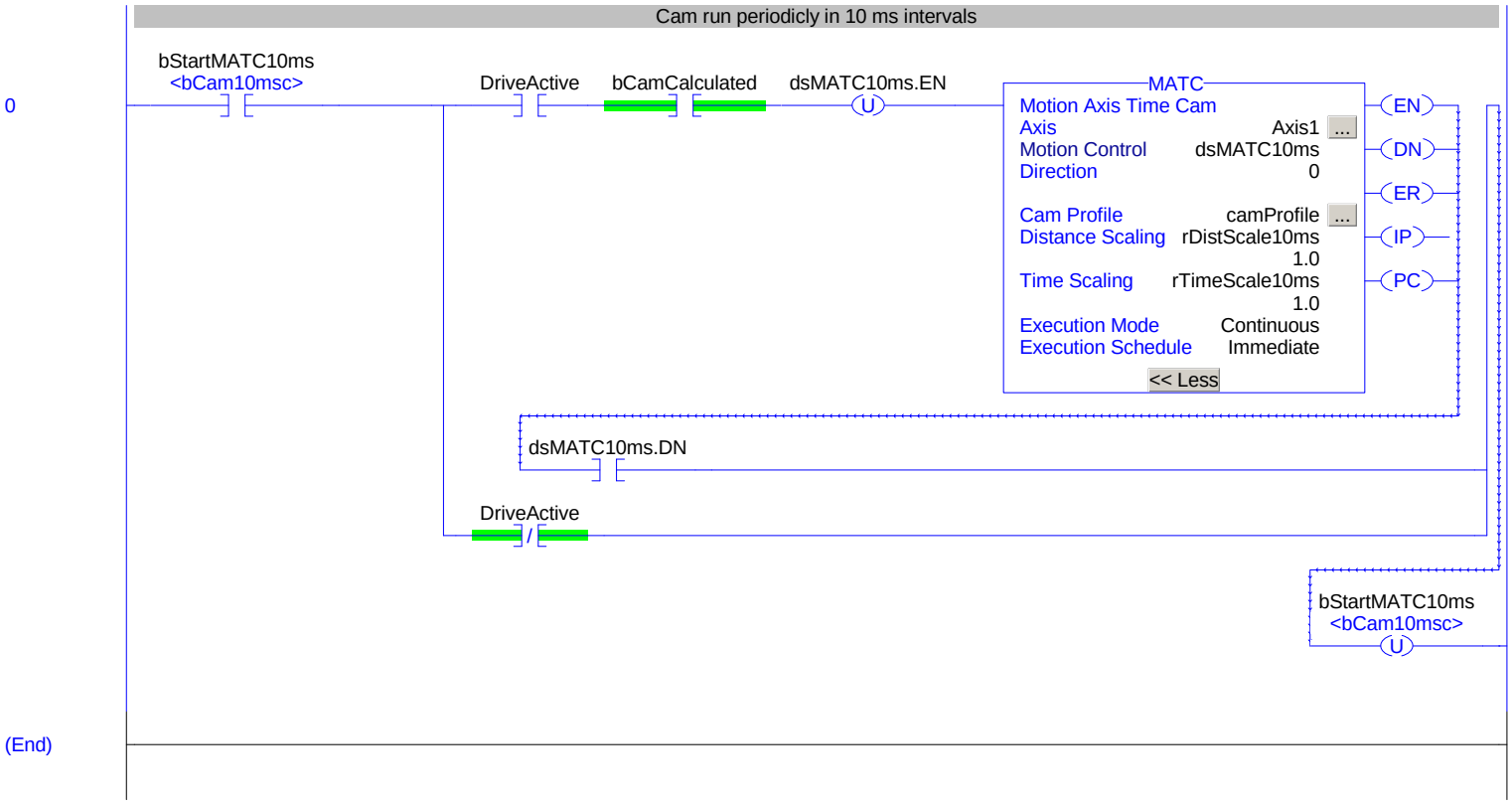
SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name PositionIntegralGain
Source rPositionIntegralGain
0.0

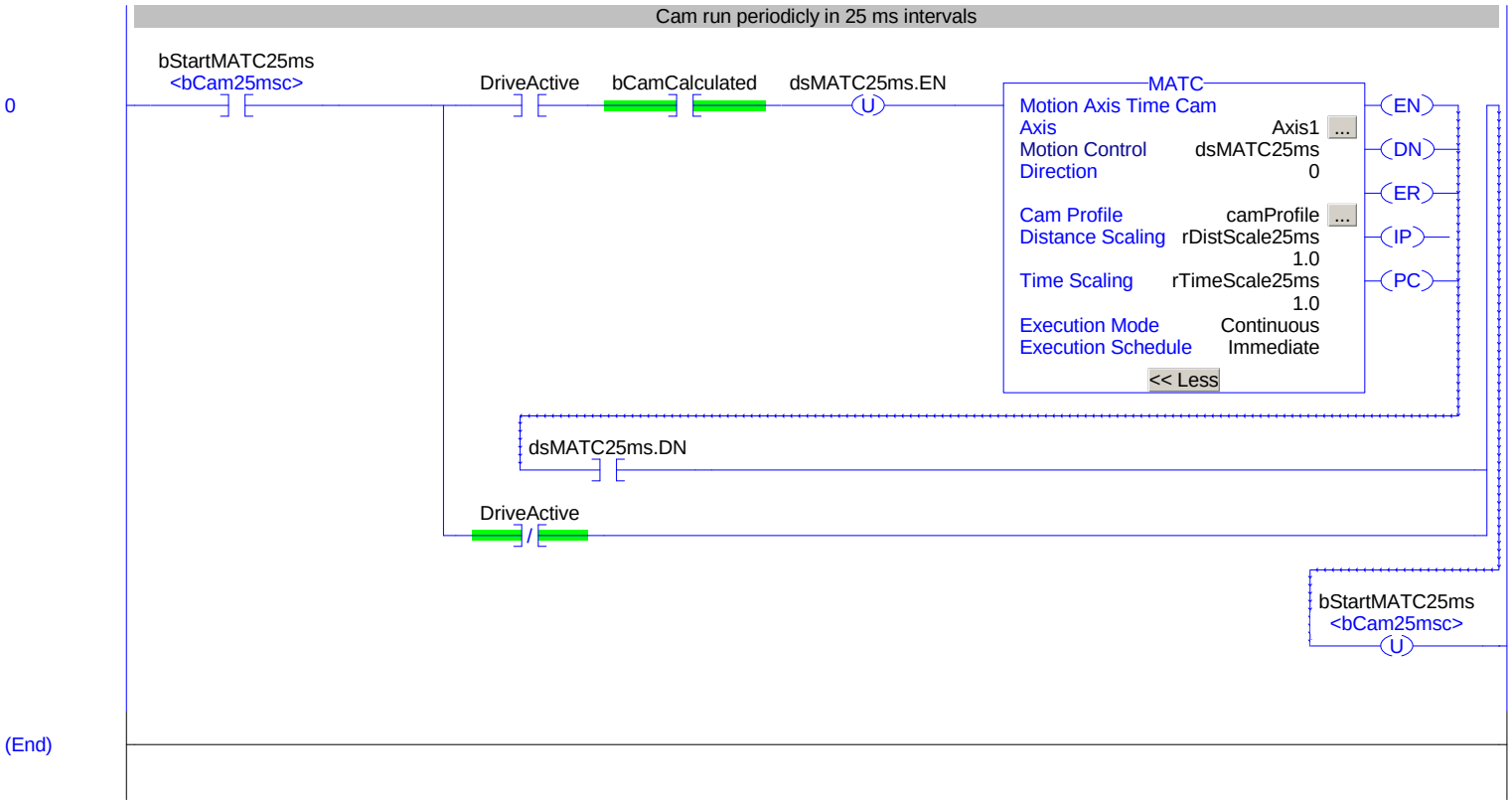
SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name PositionProportionalGain
Source rPositionProportionalGain
300.0

SSV
Set System Value
Class Name Axis
Instance Name Axis1
Attribute Name PositionErrorTolerance
Source rPositionErrorTolerance
5000.0

SetGains
(U)


3
GetGains
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name TorqueScaling
Dest rTorqueScaling
1.31167460e-003
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name LoadInertiaRatio
Dest rLoadInertiaRatio
4.397838
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name VelocityFeedforwardGain
Dest rVelocityFeedforwardGain
100.7
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name AccelerationFeedforwardGain
Dest rAccelerationFeedforwardGain
101.0
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name VelocityIntegralGain
Dest rVelocityIntegralGain
0.0
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name VelocityProportionalGain
Dest rVelocityProportionalGain
850.0
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name PositionIntegralGain
Dest rPositionIntegralGain
0.0
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name PositionProportionalGain
Dest rPositionProportionalGain
300.0
GSV
Get System Value
Class Name Axis
Instance Name Axis1
Attribute Name PositionErrorTolerance
Dest rPositionErrorTolerance
5000.0
GetGains
U
(End)


Cam run periodicly in 10 ms intervals
bStartMATC10ms
<bCam10msc>
DriveActive
bCamCalculated
dsMATC10ms.EN
U
MATC
Motion Axis Time Cam
Axis Axis1
Motion Control dsMATC10ms
Direction 0
Cam Profile camProfile
Distance Scaling rDistScale10ms
1.0
Time Scaling rTimeScale10ms
1.0
Execution Mode Continuous
Execution Schedule Immediate
<< Less
EN
DN
ER
IP
PC
dsMATC10ms.DN
DriveActive
/
bStartMATC10ms
<bCam10msc>
U
0
(End)


Cam run periodicly in 25 ms intervals
bStartMATC25ms
<bCam25msc>
DriveActive
bCamCalculated
dsMATC25ms.EN
U
MATC
Motion Axis Time Cam
Axis Axis1
Motion Control dsMATC25ms
Direction 0
Cam Profile camProfile
Distance Scaling rDistScale25ms
1.0
Time Scaling rTimeScale25ms
1.0
Execution Mode Continuous
Execution Schedule Immediate
<< Less
EN
DN
ER
IP
PC
dsMATC25ms.DN
DriveActive
/
bStartMATC25ms
<bCam25msc>
U
0
(End)

I Buttons list

Sheet: Control
Enable/Disable Drive: Turns on/off the contactor and MSO
Clear Faults: Clears faults on the drive
Run CAM: Starts the downloaded CAM
Axis Move Instructions: Opens a form where it's possible to run simple move instructions **Stop all motion:** Stops the servo and virtual axis.
Update CAM list: Updates the drop down list with the CAM's
Download CAM: Downloads the selected CAM and the time and distance scaling
Upload Data: Uploads the requested number of data points from the PLC to the specified workbook on a new sheet
Update sheets list: Updates the list with sheets from the specified workbook, ignores sheets that are not of the type Worksheet
Plot FFT: Plots a FFT graph from the selected sheet with the specified number of points and source data
Plot: Plots a XY graph with the selected values

Axis Move Instructions
MAH: Performs a homing of the servo, takes it to 0 deg
MAM: Performs a move instruction
MAJ: Performs a constant speed instruction
MAS: Stops the servo
Power ON/OFF: Same as Enable/Disable Drive
Start Data Collect: Starts sampling data to the vectors that can be downloaded from the Control sheet
Download Settings: Downloads MAM and JOG settings to the PLC
End: Closes the form

Sheet: Tuning
Get Values: Uploads the current tuning values
Set Values: Downloads the tuning values
Start Auto Tune: Performs an auto tune and sets the tuning parameters
Start/Stop Velocity Tuning: Start or stops the velocity tuning sequence
Start Position Tuning: Downloads tuning parameters, runs a position tuning sequence and plots the selected parameters in the graph
Set: Sets parameters for the graph to get the visibility needed

J Specifications

Specifications for the
Mechatronics Demo Rig Development
Master thesis.

Supervisor: Thorbjörn Bengtsson, +46 46 36 1265, thorbjorn.bengtsson@tetrapak.com
Student: Björn Lineke, +46 46 36 4607 Bjorn.lineke@tetrapak.com

Scope
The thesis is for two students but could be adjusted to fit one student. The student's background should be in the Automation area in the end of the Master Programme. Preferable the thesis should take place in autumn 2010.

Abstract
The Mechatronic Demo Rig is built for constituting the bases for Mechatronics training aimed to increase the understanding of the degree of influence different parameters has on performance of a Mechatronic system. These parameters can be compliance (e.g. play in gearboxes, torsional stiffness of a transmission element), inertia ratio motor vs. load, profile design etc. The way this shall be done is by letting the "students" setting up different cases/experiments and compare the results of these cases.

The rig shall be very flexible in terms of being able to run the mechanics in all possible configurations. This shall be achieved through parameterization on a HMI. The HMI shall be realized in Excel-VBA due to that this software is situated in basically all PC's.

The results of the tests shall be presented in tables and graphs and shall be able to be saved for documentation purposes.

Points to cover in the Master thesis
General
- Learn the RSLogix 5000 programming software in general
- Understand the Motion Control functions
- Set up a number of general test cases
- Familiarise with Excel-VBA

Specifics
- Hardware configuration according to system set up
- Write PLC logic for the general test cases
- Write instructions for the different test cases
- Write PLC logic for data collection
- HMI programming in Excel–VBA for intuitive set up of experiments
- HMI programming in Excel–VBA for presenting experiments results
- HMI programming in Excel–VBA for saving the results in a separate file

Additional Specifications
- Write PLC logic for self tuning
- Exploring the possibilities of using .NET instead of DDE

Misc
- The entire work shall be made in the English language
- All logic shall be made in the Ladder language
- Parameterization might be made in ST

K Known bugs

Tuning page: When setting all data series to secondary axis the primary and secondary axis switches places in the graph.

Plotting: Excel sometimes has a hard time plotting some plots. My experience is that plotting Acceleration against Torque from a MAJ run basically overloads Excel. Others series are fine with Acceleration against Torque.

L References

VBA Microsoft MSDN: *http://msdn.microsoft.com/en-us/library/*

Excel Tips: *http://excel.tips.net/*

Microsoft Excel 2003 and 2007 help files

Rockwell RSLogix 5000 v18 Help files

Rockwell: *www.rockwell.com*

Carton Bottle Motion Training - SW config and functions, By Thorbjörn Bengtsson, 2008

Allen-Bradley, Motion selection guide March 2010

Tetra Pak, Motion control systems, TP-CS 107, May 2008

Rockwell Automation, MPL-A310P, data sheet

Wittenstein produktkatalog 2010

My mentor: Thorbjörn Bengtsson, Tetra Pak