University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2023

Electrical Power Engineering

Development of a HIL system for simulating FMUs

Benjamin Haugnes

Faculty of Technology, Natural Sciences and Maritime Sciences

Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2023

Title: Development of a HIL system for simulating FMUs

Number of pages: 63

Keywords: servo machine, HIL, FMI, FMU

Student: Benjamin Haugnes

Supervisor: Dietmar Winkler

Summary:

Several years ago, USN purchased a servo test stand for education purposes, to demonstrate small-scale load handling, consisting of two induction machines connected to each other, and controlled by its own driver. The servo test stand has a limited range of application, so a real-time processing unit was acquired to remote control the test stand with more opportunities.

The purpose of this thesis is to analyses the possibility of controlling the servo test stand with the use of a real-time processing unit, and the possibility of controlling the servo test stand with the use of models based on the function mock-up interface architecture. In this study a hydropower model was included to simulate how the varying flow in the penstock is acting on the turbine and generator, using both OpenModelica and Dymola to generate FMUs.

It was found to be possible to create simple FMUs in OpenModelica for controlling the servo test stand, and when creating hydropower FMUs the modelling tool Dymola had to be used to make it possible to control the servo test stand, due to interface challenges between OpenModelica and the hydropower library OpenHPL.

Preface

This master's thesis was written as the final part of the two-year mast program in Electrical Power Engineering, at the University of South-Eastern Norway (USN).

I would like to use this time to thank my supervisor, Dietmar Winkler, for giving excellent support and guidance through this project. I would also like to express my gratitude to the people at OpenModelica for their help with FMU. The task description can be found in Appendix A.

Porsgrunn, 15.05.2023

Benjamin Haugnes

Contents

Nomenclature

AC	Alternating Current
CPU	Central Processing Unit
DC	Direct Current
FB	Function Block
FMI	Functional mock-up interface
FMU	Functional mock-up unit
FPGA	Field Programmable Gate Array
HIL	Hardware-in-the-loop
I/O	Input/Output
LAN	Local Area Network
PID	Proportional, Integral, and Derivative
RMF	Rotating magnetic field
RTC	Real-Time processing unit
VFD	Variable Frequency Drive
WLAN	Wireless Local Area Network

1 Introduction

This introduction consists of background and motivation, previous work done related to the servo test stand, objective, and scope of this report.

1.1 Background and motivation

Through the last year (2022), the focus on energy has been emphasized heavily due to the Russia-Ukraine war. This has led to skyrocketing energy prices in Europe and led to the focus on being energy independent [1].

The world is also in a transition going from fossil-based energy sources towards renewable energy sources at the same time.

With ongoing high energy prices and the global electrification has led to a focus on optimizing the already existing system to handle more energy.

Several years ago, USN acquired the servo test stand for education purposes, to demonstrate small-scale load handling. The servo test stand consists of two induction machines connected to each other, where each of the machines is controlled by its own driver. Due to the limitation in the application range that this servo test stand could be used for, a Real-time processing unit (RTC) from the company dSPACE was purchased by the university to use it for remote controlling the servo test stand.

1.2 Previous work

There is only one previous work done related to the servo test stand conducted by USN.

A master's thesis was done in the spring semester of 2022 [2]. In this master's thesis, the main objective was to remote control the servo test stand by a computer running a model. It was concluded that the servo test stand could be remotely controlled by using a Python script, that was sending signal through an Arduino, that was connected to the I/O ports of the servo test stand.

1.3 Objective

The intended purpose of this report is to replicate the work done in the previous master's thesis [2], Development of an open control interface for a servo machine test stand, using an RTC. The test stand is intended to be used for HIL simulation. To complete the main objective a study of the process of loading a hydropower model as an FMU to the RTC is needed, to prove the feasibility of this setup.

1.4 Methods

Information will be gathered from data sheets of the hardware and software provided by the vendors and from relevant online literature. The components and software have been analysed to find limitations. For modelling the programs Dymola [3] and OpenModelica [4] have been used.

1.5 Scope

The scope of this thesis is to analyse the components of the HIL setup and interface capabilities and restrictions. Implementing hardware input and output connection for the servo test stand and MicroAutoBox III. Investigating the process of exporting FMUs and controlling the HIL setup with an FMU.

1.6 Outline of this report

In Chapter 2 the concept of the setup is presented.

In Chapter 3 the hardware and software used for this setup are presented.

In Chapter 4 the process of setting up the system and running a test is presented.

In Chapter 5 the process of building and testing a hydropower FMU is presented.

In Chapter 6 the discussion finding and observations from chapter 4 and 5.

In Chapter 7 the conclusion is presented.

2 Concept

The concept is to implement a HIL system that can be controlled by a simulation of a digital twin which is representing a complete hydropower system. The purpose of this HIL system is to use it to simulate how varying water flow in the penstock acts on the turbine and the generator. The block diagram of HIL system can be seen in Figure 2.1, and this system consists of a model, real-time processing unit (RTC), and a servo test stand.

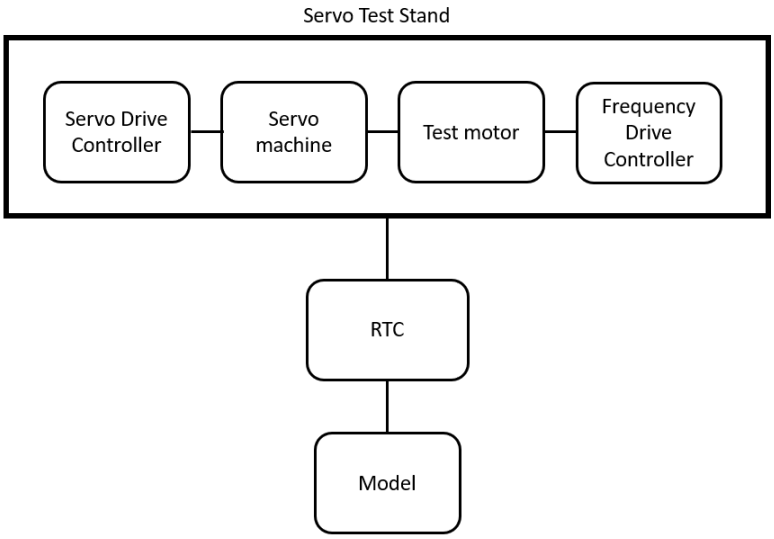

Figure 2.1: The block diagram shows an overview of the HIL system.

The servo test stand consists of a servo machine controlled by a servo drive controller and a test motor controlled by a variable frequency drive (VFD) controller. For the hydropower purpose the servo machine will act as the generator running at constant speed while the test motor will act as the turbine applying different load to the generator. The servo machine and the test motor are described more in chapter 2.1, while the VFD and servo drive is described in chapter 2.2 and 2.3 respectively.

The hydropower model will be made as a functional mock-up unit (FMU) [5], which allows for models in different modelling programs to interexchange between. The FMU is described in chapter 2.4 and hydropower model in chapter 2.5.

The model will be loaded on to the RTC which will communicate between the model and hardware of the system. The RTC is a device that receive data form both the model and the system, which it is processing and sending back out as an output to the system. Laying in the name real-time, this unit must operate as instantaneously as possible.

2.1 Induction machines

Induction machines are also called asynchronous machines and is commonly used. About 90% of all electrical motor are induction machines, because they are reliable and robust in operation [6].

Induction machines can be run as both motor and generator. They are often only referred to as induction motors, due to when they are operating as a generator the output voltage can vary widely when the load is changing. For smaller installation such as windmills they are favoured, because they are usually working in parallel with a larger grid and only providing a fraction of the power provided to the customer. In this case the varying output voltage from the generator will not be a problem due to the larger grid is providing voltage and frequency control [7].

There are two main types of three phase induction machines, the squirrel-cage and the wound-rotor (slip ring rotor) motor, where the squirrel cage is the most used one due to higher cost and maintenance concerned with the slip ring rotor [7].

The squirrel-cage stator consists of a highly permeable steel laminations, where a three-phase winding circuit is passed through slots in the steel lamination. The rotor consists of laminated core with slots for carrying conductors that is shorted in both ends of the rotor, to create a closed circuit. The conductors on the rotor is a bit skewed, to minimize vibration. Lamination is used in both the stator and rotor to reduce eddy currents, that is generated by the changing magnetic field in the windings [8].

The mechanical separation of the stator winding of 120°, produces a rotating magnetic field (RMF). Due to this RMF an electromotive force (EMF) is induced in the rotor, that will produce an induced current going through the conductors on the rotor. This induced current has a changing magnetic field that turns the rotor [9].

The synchronous speed of the machine (n_{sync}), is determined by the applied frequency (f) and the number of poles (p) on the machine, as seen in Equation (2.1) [7].

$$n_{sync} = \frac{120f}{p} \tag{2.1}$$

The induced current in the rotor is lagging the main field current, making it difficult to achieve synchronous speed [10].
The slip speed (n_{slip}) is the difference between the synchronous and rotor (n_m) speed, given in Equation (2.2). This can be used further to find the relative speed of the motor (slip), given in Equation (2.3) [7].

$$n_{slip} = n_{sync} - n_m \tag{2.2}$$

$$S = \frac{n_{slip}}{n_{sync}} \bullet 100\% \tag{2.3}$$

The induction machine can operate in four different ways, forward motoring, reverse motoring, forward braking and reverse braking. When the machine is braking, it is acting as a generator, the torque and speed is going in opposite directions of each other. When the machine is

motoring, it is acting as a motor, the torque and speed is going in the same directions, as can be seen from Figure 2.2 [11].

By Reducing the RMF to be lower than the rotor speed, giving a higher rotor speed than synchronous speed, the machine while then act as a generator. The slip will in this case be negative.

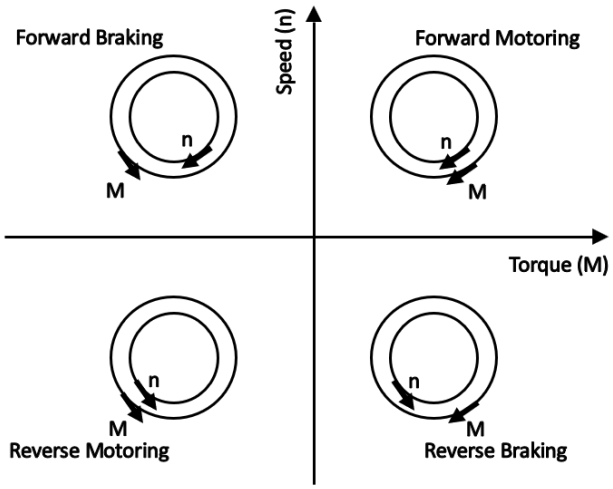

Figure 2.2: Showing the four different quartiles an AC motor can work in.

The relation between revolutions per minute (rpm) and the angular velocity (radians per second), can be seen in Equation (2.4), and the relationship between torque (τ), power (P) and angular velocity (ω) can be seen in (2.5) [7].

$$\omega = 2\pi \bullet \frac{rpm}{60} \tag{2.4}$$

$$\tau = \frac{P}{\omega} \tag{2.5}$$

2.2 Frequency drive controller

Variable frequency drive (VFD) is a controller that is used for driving electrical motors, by varying the supplied voltage and frequency to the motor. VFDs are used to improve the efficiency, and to reduce the mechanical stress on the motor [12].

The VFD is installed between the power supply and the motor to control the flow of energy from the supply to the motor. Inside the drive the AC power is converted to DC power, and then feed though capacitors before the power is then inverted from DC power back to AC power and deliver to the motor. The driver is rectifying the power to make it possible to adjust

the voltage and frequency that is fed to the motor, which makes it possible to run the motor at torque or speed as required. The capacitors, inside the drive is used to remove the ripple currents created when AC power is converted to DC [13].

In a VFD the output frequency and voltage are usually controlled by a pulse-with-modulation (PWM) technique, where the PWM retrieves a digital signal that tells it to switch the DC power on and off to simulate the AC oscillations [13].

2.3 Servo drive controller

The servo drive controller works much like the frequency drive controller in the way it controls the flow of energy to adjust the torque and speed of the servo machine. The main difference is that the servo drive controller is installed with a feedback loop. The feedback loop will send back information about the speed and position of the machine's rotor to the servo drive, which will then adjust the voltage and frequency applied to the machine, to minimize the difference between the applied and measured speed. The feedback loop consists of a sensor that is connected to the machine's rotor and measures the speed of the rotor and sending this information back to the servo drive controller [14].

The most common ways to control the speed is either by using a PID controller or using a PWM controller. As mentioned in 2.2 the VFD use the PWM controller method, while the servo drive controller uses the PID controller method. PID is a control loop system that uses the feedback loop to eliminate the difference between the applied and measured speed in the system. [14].

The servo drive used in this setup uses a resolver as speed and position sensor. The resolver consists of two main parts, a stator and a rotor which is attached to the shaft of the servo machine.

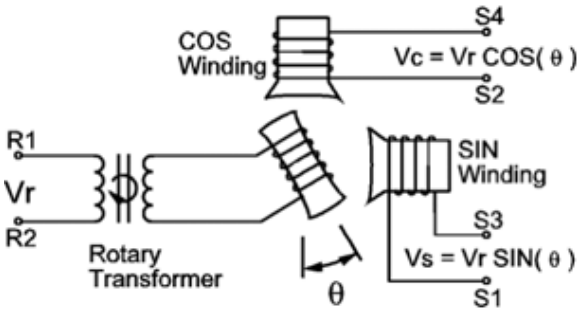

Figure 2.3: Illustration of the resolvers windings [15].

The stator of the resolver consists of three windings, primary, sin and cosin winding. The sin and cosin windings are mounted 90 degrees apart from each other. It works in the way that primary windings of the stator are sending a sin AC input which gets transformed by the rotor. The rotor has rotating windings which will induce voltages in the cosin and sin windings. This induced voltage in the sin and cosin winding is measured and sent to the resolver digital converter (RDC) to determine the speed and position of the machine [15].

2.4 FMU

The FMU is based on the functional mock-up interface (FMI) [16] architecture, which is an open standard used for exchanging dynamic simulation models as a standardized format between different modelling tools. The FMI comes in three standardized versions 1.0, 2.0 and 3.0. The standard versions are neither forward nor backward compatible. Which makes it not possible to use an FMI 2.0 based model in a tool that only supports FMI 1.0 [17].

The FMU is distributed as a ZIP file that contains a model description in the form of an XML file, either a binary code or C source code, or both to execute the FMI functions[17].

The structure of the binary code is dependent on the operative system where the model is created, meaning that a binary code made for the model in a Windows operating system cannot be run on a Linux operating system, unless it is created on the Windows system with Linux binaries, and vice versa through a Docker or a virtual machine (VM) [17].

C source code often also only referred to as source code, is independent on the operating system where it is created and executed, meaning that the model can be created on a windows computer and then be transported to a Linux operating system for simulation, making it possible to generate a FMU without knowing the end target processor of the importer [17].

Model description XML is a text file that can be read by both machines and humans, and the file stores information about parameters, variables and etc of the model. This makes the FMU having a smaller footprint and takes up less memory space of the processing unit [17].

The FMI has three different interface types:

- Co-Simulation.
- Model Exchange.
- Scheduled Execution.

All these three interface types can be used for exporting models from one tool to another and to simulate together with several other models. The main difference lays in the way the interface work. Scheduled Execution is only a part of the FMI 3.0, and not the previous versions. Due to FMI 3.0 being quite new, there are few modelling programs that are supporting this architecture, compared to the FMI 2.0 that was released in 2014 [5].

The RTC in this thesis is using the FMI 2.0 architecture as can be seen in chapter 3.1.4, due to differences in the features of Co-Simulation and Model Exchange from FMI 2.0 to FMI 3.0. The following listing is for the FMI 2.0 (except for the Schedule Execution as this can only be found in FMI 3.0).

- Co-Simulation comes with its own fixed solver, where the parameters of the solver can be change, but not the solver itself. Due to the inclusion of a solver, this causes a delay to the information that is traveling through the FMU. The internal solver is setting the initial states of the model, and the time steps are defined in the model [17].
- Model Exchange comes without a solver, and the functions of the model can be directly accessed by the importing tool, making it possible to manipulate the model's equations. The external solver is setting the initial states and time steps for this model [17].
- Scheduled Execution: makes it possible to run the model at scheduled time and not at real-time. The scheduler is not part of the FMU and must be provided by the importer. It has the possibility of being exported with or without an internal solver [17].

2.5 Hydropower model

Depending on the type of hydropower plant, the components of the system can difference quite a lot, but the main principle of a hydropower system is that they have a penstock (pressure shaft) leading water to a turbine which is rotating a generator. The generator is producing electricity that is delivered to the grid. The turbine has an outlet for removal of the water that has been used for energy production as can be seen in Figure 2.4.

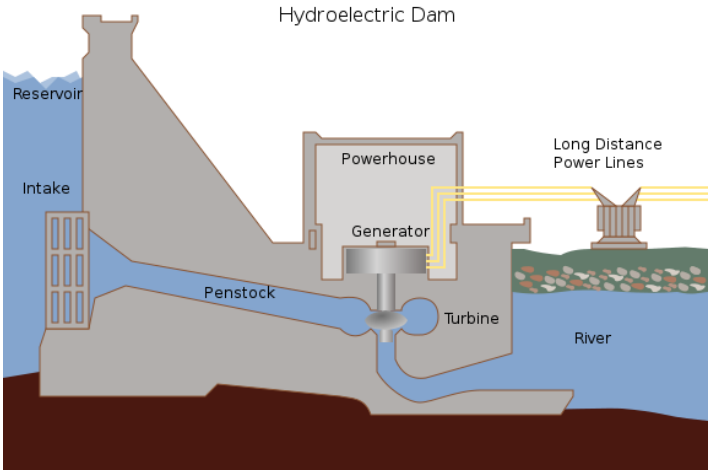

Figure 2.4: The illustration is showing the main parts of a hydropower plant[18].

As mentioned in chapter 2, the intention is to use the setup to simulate varying flow in the penstock, and how it acts on the turbine and generator. Varying flow can be caused by several factor, such as reduction of the guide vane opening. When the guide vanes opening is reduced, this leads to pressure building up in the penstock, which causes a varying flow of water to the turbine and hence effects the generators power production.

The guide vanes are flaps that are used to guide the water to the turbine as can be seen in Figure 2.5. They are typically used in combination with the either the Kaplan or the Francis turbine.

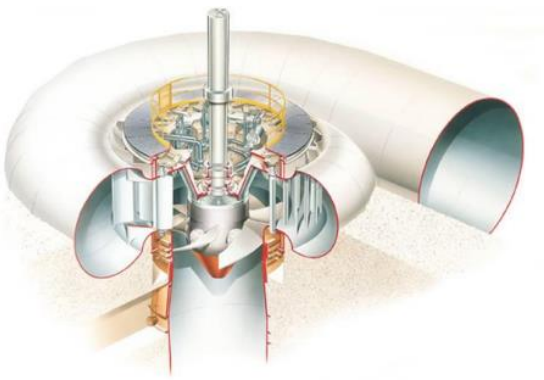

Figure 2.5: Kaplan turbine with guide vanes. Illustration from Kvaerner.

The hydropower system can be expressed as one unit through a transfer function, as seen in Figure 2.6, where mark 1 represents the PI controller, mark 2 the waterway (including the turbine) and mark 3 the mechanical part of the system. The transfer function is a simplification of the system, that can be used for designing control systems. The values for Tw and Ta can be seen in Equation (2.6) and Equation (2.7), where Tw is the time constant of the waterway and Ta is the time constant of the mechanical acceleration.

The transfer function can be derived by using the formula for head change (pressure change), the valve equation (for the turbine) and acceleration of a swinging mass with per-unit quantities and Laplace transformation [19].

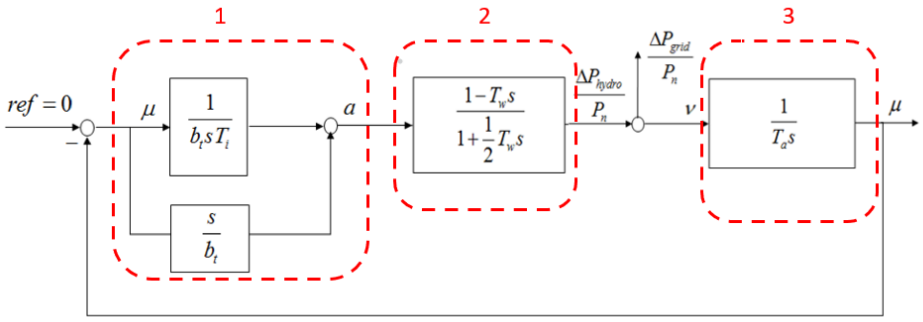

Figure 2.6: Transfer function representing the hydropower power plant. Where 1 is the PI controller, 2 is the waterway and 3 is representing the mechanical part of the system [19].

$$T_W = \frac{Q_n}{g * H_n} \bullet (\frac{L}{A}) \tag{2.6}$$

$$T_a = \frac{J \bullet \omega_n^2}{P_n} \tag{2.7}$$

Where:

- Q_n = Nominal flow rate (m^3/s).
- H_n = Nominal water head (m).
- g = Gravity acceleration (m/s^2).
- L = Total length of the waterway (m).
- A = Average pipe areal (m^2).
- P_n = Nominal turbine power (W).
- J = Turbine and generator inertia ($kg \bullet m^2$).
- ω_n = Nominal angular velocity (rad/s).

3 HIL setup

In this chapter the hardware, communication path and software used in this HIL setup is described. The chapter also includes the modelling tool used to create tests models for the HIL system in chapter 4 and 5.

3.1 Hardware overview

The servo test stand was provided by the company Lucas-Nulle [20], and it consisted of two induction motors, one frequency drive and a servo drive controller with a servo drive controller interface shown in Figure 3.1. In addition, the RTC unit MicroAutoBox III [21] was acquired from the company dSPACE.

Figure 3.1: The servo test stand, where the striped box shows the servo test stand provided from Lucas-Nulle.

3.1.1 Induction machines

The servo machine is an MCA four-pole model from Lenze rated at 390V. The machine has a power factor of 0.76 (cos φ), and delivers a power of 1.7 kW at 140 Hz, with a current of 4.4A and a nominal speed of 4050 rpm [22].

The test motor is a M2VABBA_3 six-pole model from ABB, rated at 220-240V. The motor has a power factor of 72 (cos phi). The power consumed by the motor is 0.37 kW at 50 Hz and with a current of 2.25 A. The nominal speed of this motor is 910 rpm [23].

3.1.2 Frequency drive

The VFD used in this servo test stand is the ABB ACS350 as seen in Figure 3.2. It is installed with a control panel that can be used to control the drive locally, but there are some limitations for the control options for the drive such as cascade control is not an option. Therefore, it is installed with analog and digital I/O ports and has the possibility to be installed with adapters to communicate through CAN and several other communication protocols. The VFD have seven different macro settings that it can be programmed with [24].

Figure 3.2: The ABB ACS350 VFD [24].

From the display on the control panel the speed, current, torque load, time and control type of the motor can be seen. The menu option of this controller can be seen in Table 3.1.

Table 3.1: Shows the menu options of the VFD.

Parameters
Assistants
Changed parameters
Fault logger
Time and date
Parameter backup
I/O settings

3.1.3 Servo drive controller

The servo test stand uses the 9400 HighLine [25] servo inverter from Lenze, seen in Figure 3.3, as the servo drive controller. The servo drive is installed with a brake resistor to dissipate the energy generated when the servo machine is braking, to eliminate over voltages faults. The braker resistor that the servo drive is installed with can withstand 2000W. The servo drive is installed with three different communication paths, CANopen, I/O ports and Diagnostics port.

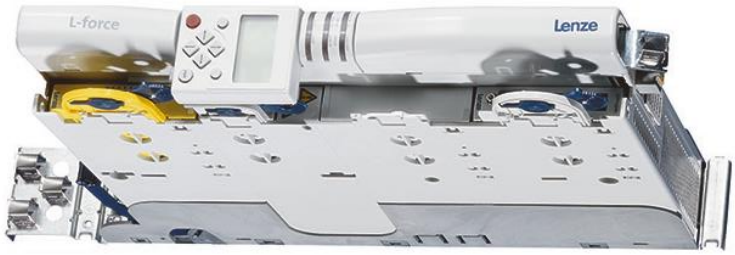

Figure 3.3. The Lenze 9400 HighLine servo inverter [26].

The servo drive controller was also provided with a control panel from Lucas-Nulle. The control panel comes with its own software called ActiveServo [20] which can be used to configure the control panel seen in Figure 3.4.

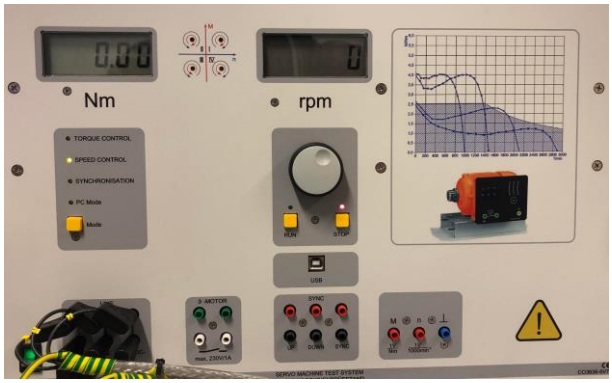

Figure 3.4: The servo drive control panel.

3.1.4 MicroAutoBox

The MicroAutoBox III is the RTC used in this setup, and it consist of three main modules, a DS1403 Processing board, a DS1513 multi I/O board and a DS1514 FPGA based board. The MicroAutoBox III has an ARM processor architecture and is running on a Linux based operation system. The MicroAutoBox comes equipped with its own web interface that can be accessed through the IP address of the device [27].

The processing board consists of a processor that is executing the real time processing and enables the MicroAutoBox to communicate with a host computer, and it also contains two ethernet, one automotive ethernet and one USB port.
There are two ways to communicate between the host computer and the MicroAutoBox, through LAN directly or WLAN. The DS1513 and DS1514 boards provides additional I/O channels for both analog and digital signals, and a CAN interface for the processing board [27].

DS1513 is a multi I/O board that can handle both analog and digital signals, and can also handle CAN, LIN and UART signals [27].

The DS1514 FPGA board can be configured to meet other hardware specifications, such as the DS1552 Multi-I/O board which is similar to the DS1513 board [27].

The MicroAutoBox setup consist of three separate components, a MicroAutoBox III, a Break-Out Box and a power supply. The Break-Out Box and power supply is described in chapter 3.1.4.1 and 3.1.4.2 respectively [27].

Figure 3.5. The dSPACE MicroAutoBox III [21].

3.1.4.1 Break-Out Box

The MicroAutoBox Break-Out Box DS1541 is a desktop connection panel used for flexible signal connections for the MicroAutoBox series. The Break-Out Box consists of 156 terminal blocks which each has 6 I/O connection points and is accessed by the MicroAutoBox through a 156-signal pin connection cable. The Break-Out Box is an extension to the different DS boards such as the DS1513 and the DS1514 that is installed on this version of the MicroAutoBox [27].

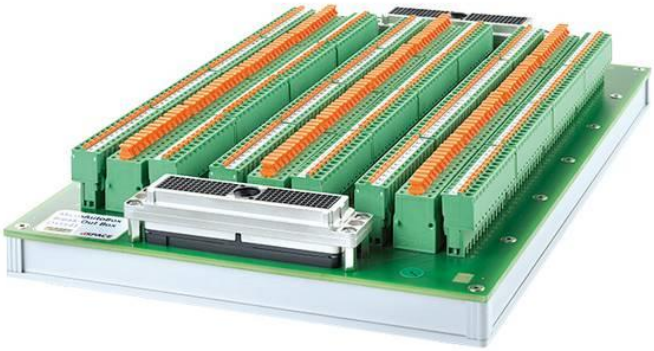

Figure 3.6. The dSPACE MicroAutoBox Break-Out Box DS1541 [28].

3.1.4.2 Power supply

The power supply used is the model TSP 600-124 from Traco Power. It takes an input voltage of 110-120/220-240V AC at 50/60 HZ and converts it to 24-28V DC, that is delivered to the MicroAutoBox. Input current is 5-10A, and output current is 25A at 24V DC. Output power is 600W at 40°C. For safety reasons for the MicroAutoBox, there is installed a fuse rated at 25A/80V. The power supply has knob that can be used to adjust the delivered DC voltage within its given range [29].

Figure 3.7. Traco Power TSP 600-124 power supply [30].

3.2 Communication path

The MicroAutoBox, servo drive and VFD have several different ways to communicate with other hardware. A common communication for those three is CAN as well as analog and digital I/O signals. As described in chapter 4.5 the system is using analog and digital signal to communicate between each other.

Digital I/O signals are signals used to communicate between electronic devices. They are sending bits, which are either 0 or 1, that is typically generated by a switch or a sensor sending a voltage signal that is interpreted as either logic high (1) or low (0), representing a true or false statement. A voltage of 0V is interpreted as a low logical state while a voltage of 24V is interpreted as a high logical state [31].

Analog I/O signals are sent/received as a continuous signal. Typically, the signal can vary between 0-10V and 4-20 mA, where a 0V or 4 mA represents 0% scaling and 10V or 20 mA represents 100% scaling [32].

3.3 Software

The MicroAutoBox uses the programs ConfigurationDesk [33] and ControlDesk [34] from dSPACE and the servo drive uses Engineer and Easy from Lenze [35].

When it comes to which modelling software that can be used, there are several options such as Dymola [3], Simulink [36], OpenModelica [4], SimulationX [37] to give some examples. Almost every one of them comes with a license that must be purchased. OpenModelica is a promising open-source modelling software based on the Modelica modelling language and is operating in similar manner as the popular Dymola software.

3.3.1 ConfigurationDesk

The data program used to create real-time applications for the dSPACE RTCs is called ConfigurationDesk. The ConfigurationDesk uses three main parts to create a fully functionable application, which are model topology, device topology and hardware topology [38].

Where the hardware topology represents the functionality of the MicroAutoBox, model topology represents the functionality of the model and the device topology represents the functionality of the external device [38].

The ConfigurationDesk can create runnable applications for models based on Simulink implementation container (SCI), FMU and bus simulation container (BSC). The ConfigurationDesk is embedded with MATLAB Simulink which makes it quite easy to work with Simulink. Python is also embedded for creating scripts for the application [39].

For FMU models, the ConfigurationDesk fully supports FMUs made in Dymola, Simulink and SimulationX, for FMI version 2.0 and the interface type Co-Simulation [38].

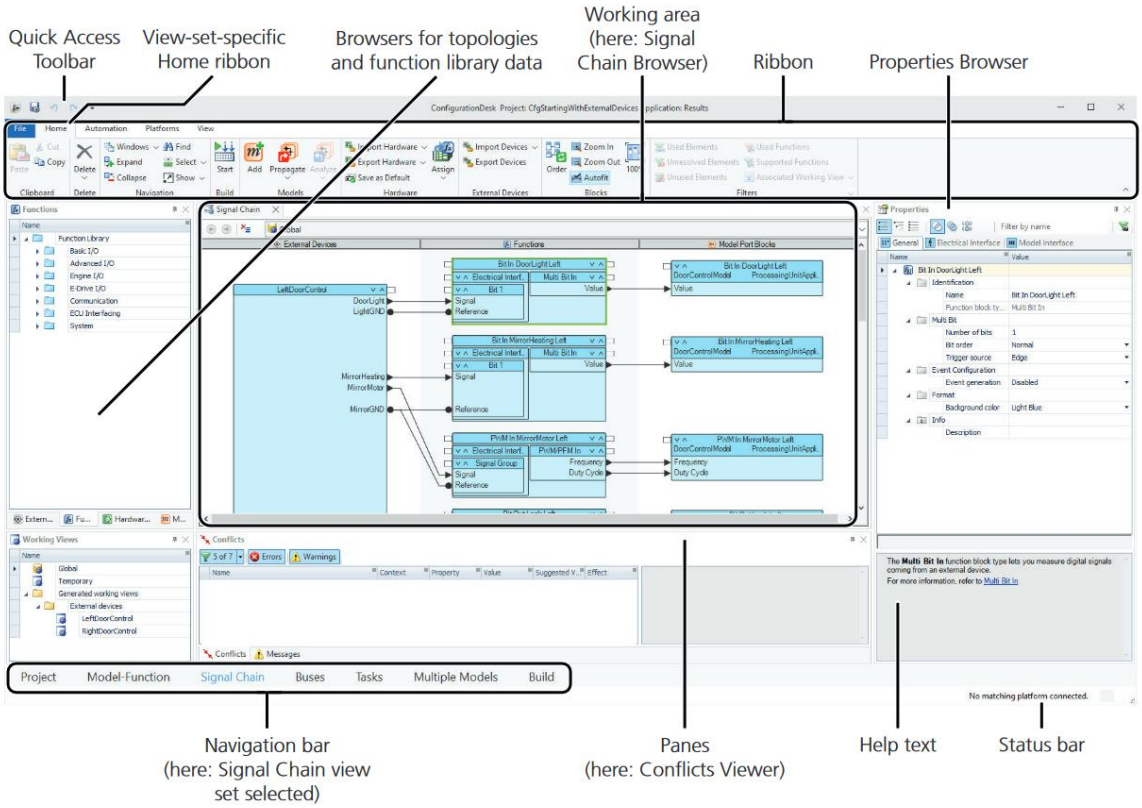

Figure 3.8: Shows the different properties and functionalities of the ConfigurationDesk [38].

The process of building a simple application based on a FMU model and connecting the MicroAutoBox to the ConfigurationDesk is described in Appendix B.

A disadvantage of the ConfigurationDesk is that there are no possibilities to change out the original parameters of the model directly in the software itself. This problem can be bypassed by the following three methods:

- **Method 1 (XML Cache):** by opening AppData on the computer and going to this address, \Local\Temp\dSPACE\CfgDesk\Cache\"*cache number*"*"application name"* \Components*" model name"*. The model description file can be found in this location and be changed [40].
- **Method 2 (reloading model):** by updating the model in the modelling software with new values and reload it to the ConfigurationDesk.
- **Method 3 (ControlDesk):** The ControlDesk is another software provided by the dSPACE, where it is possible to create several data sets that can be loaded to the

platform. The process of making data sets and how the ControlDesk is working is described in chapter 3.3.2.

A disadvantage with method one is when storing the parameters in the cache of ConfigurationDesk, if the model is reloaded in ConfigurationDesk, the new values will then be replaced with the old values.

3.3.2 ControlDesk

ControlDesk is a simulation testing tool, used for testing the real-time application made in the ConfigurationDesk. A limiting factor of the ConfigurationDesk is that there are no ways to change the original parameters of the model without editing the model description and recompiling the application, the ControlDesk is a solution to this problem [41].

By loading the spatial data file (SDF) produced by the ConfigurationDesk when compiling an application on to the ControlDesk, it is possible to create several new data sets with new parameter values that can be loaded on to the MicroAutoBox III. SDF is a file that contains the system description of the real-time application, including the CPU name of the unit and parameters of the model [42].

The ConfigurationDesk layout, which can be seen in Figure 3.9, has the possibility to control the application running on the MicroAutoBox, take measurement, record and change the parameters [43]. It is also embedded with Python for creating scripts.

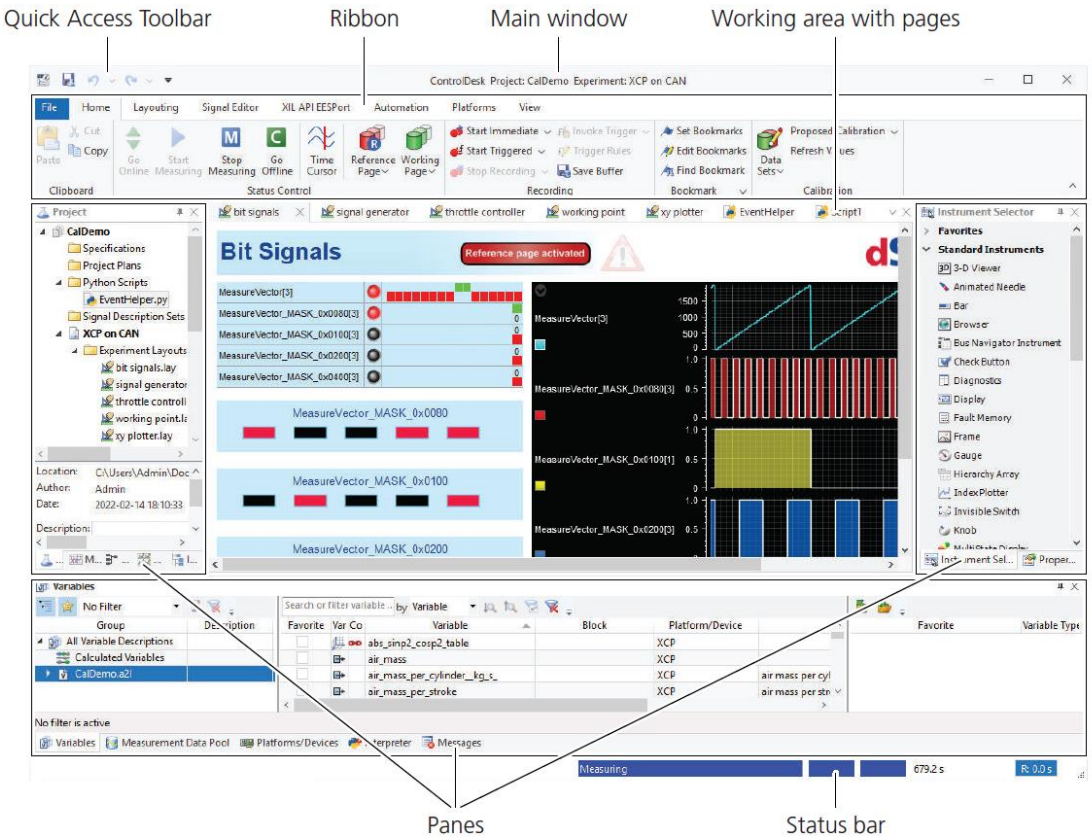

Figure 3.9: Shows an overview of the ControlDesk [41].

The process of building a simple test environment is described in Appendix B.

3.3.3 Lenze Engineer

The servo drive uses the Lenze Engineer software to configure parameters and functionalities of the servo drive, which can be seen in Figure 3.10. Through the Lenze Engineer software the Easy Navigator software can be accessed. This makes it possible to use the Function Block (FB) editor tab.

The Easy Navigator gives access to add ports and build the communication path for the system. By using the diagnostics adapter, the servo drive can be accessed.

The Lenze Engineer software gives the opportunity to upload data from the system or to create a new project. For creating a new project, the Lenze package manger can be uses to install libraries containing the specific parameters for the different Lenze products.

When the Lenze Engineer software is in online mode (connected to servo drive), changing parameters in Engineering will be implemented live [44].

On the bottom of the Figure 3.10 the live speed, speed setpoint, torque setpoint and motor current of the servo machine is displayed when the servo drive is active.

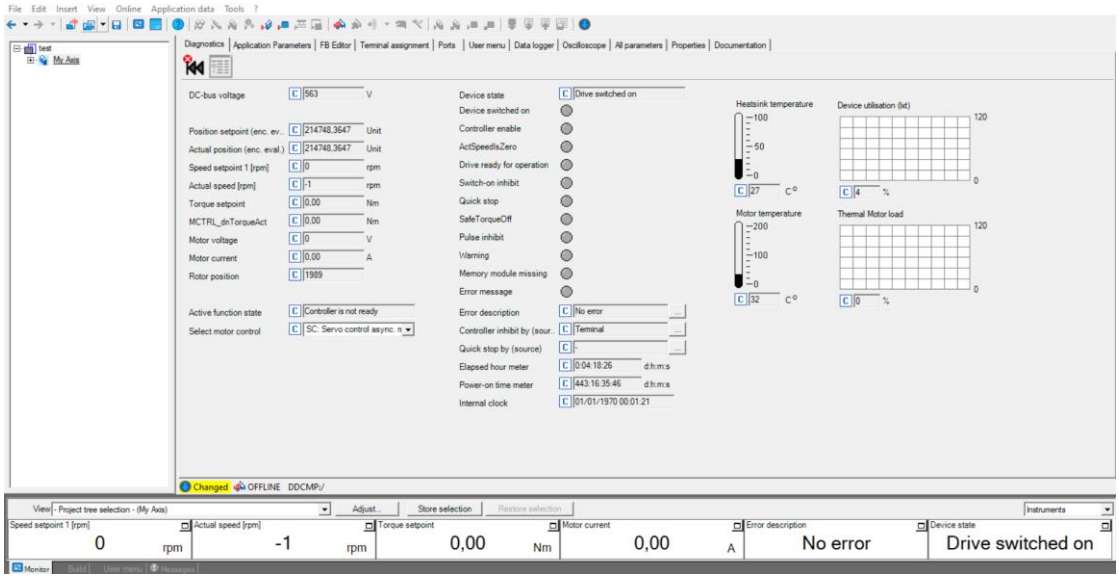

Figure 3.10: Showing the interface of the Lenze Engineer software.

3.3.4 OpenModelica

OpenModelica comes in several versions, the standard, the beta and the nightly version. The version used here is the nightly 1.22, due to this version is updated every night through the work week, and because this version allows for both exporting binary code trough docker and comes with a prefixed modelIdentifier in the source code. This functionality is not available in the standard 1.21 and the beta version.

OpenModelica supports the FMI standard version 1.0 and 2.0 for both Model Exchange and Co-Simulation, and it can also import Model Exchange and Co-Simulation FMUs created in other modelling tools. OpenModelica can generate both source code and binary code for the FMU. When it comes to the binary FMU code, it can be exported in Native or for another

operating system using a docker. To use the docker method a docker desktop [45] must be installed on the host computer.

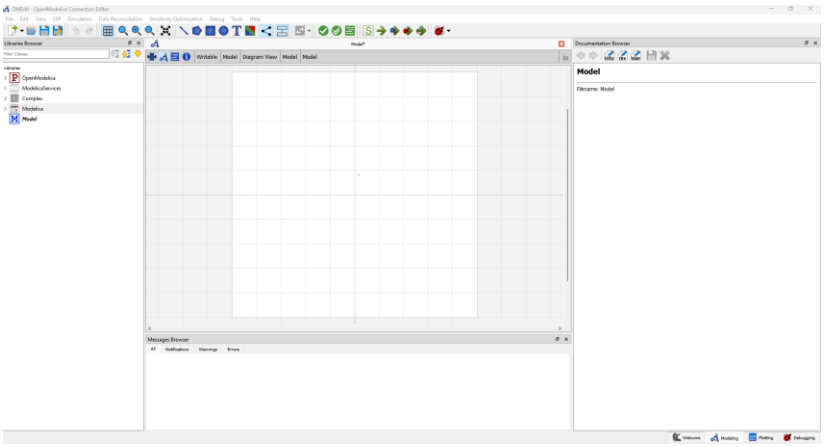

Figure 3.11: Shows an overview of the OpenModelica software.

OpenModelica comes with the ability to use libraries provided by third parties such as OpenHPL [46], which is an open-source hydropower library made by the USN. This library contains waterway components like reservoir, turbine, generator, pipes, etc. and it is linked with the OpenIPSL [47] which is an open-source Power System Library.
Several other libraries can also be installed for different purposes. The OpenHPL was used for the model described in chapter 5.

4 Setting up the HIL system

This chapter contains the description of the process of setting up the HIL system. To test that the HIL system is working as intended, a simple model of a turbine-generator combination is created in OpenModelica, where the guide vane reduces the waterflow to the turbine. The model is then exported as an FMU to the MicroAutoBox for testing the system. The complete model and FMU can be found at GitHub [48].

4.1 Connecting the hardware

Mentioned in chapter 3.1, the MicroAutoBox, VFD and servo drive have options for both communicating through digital and analog I/O connections as well as CAN. Since the MicroAutoBox must configure the CAN as an I/O connection through the Break-Out Box, the most viable method is to use the standard digital and analog I/O ports.

Both the MicroAutoBox and the servo drive has programmable I/O ports, see Table 4.1 and Table 4.2. While The VFD has fixed functions for the different macros, Table 4.3 shows the I/O functions for the torque control macro. This macro was chosen for the VFD, because it allows for both speed and torque control.

Table 4.1: The digital and analog I/O ports for the servo drive.

Terminal	Range	Alternative Range	Function
AI 1	±10V	±20mA	Programmable
AI 2	±10V	-	Programmable
AO 1	±10V	-	Programmable
AO 2	±10V	-	Programmable
DI 1-8	24V	-	Programmable
DO 1-4	24V	-	Programmable

Table 4.2: The digital and analog I/O ports of the MicroAutoBox DS1513 I/O board.

Terminal	Range	Alternative Range	Function
AI 1-16	±10V	-	Programmable
AI 1-16(1)	±10V	-	Programmable
AO 1-8	±10V	-	Programmable
DI 1-24	24V	5V	Forward / Reverse
DO 1-24	24V	5V	Programmable

Table 4.3: The digital and analog I/O ports of the VFD for Torque control macro.

Terminal	Range	Alternative Range	Function
AI 1	0-10V	±10V	Speed setpoint
AI 2	0-10V	0-20mA	Torque Setpoint
AO	4-20mA	-	Speed
DI 1	24V	-	Stop / Start
DI 2	24V	-	Forward / Reverse
DI 3	24V	-	Speed / Torque control

4.2 Setting up the variable frequency drive

In chapter 3.1.2 it was described that the VFD has a control panel, that can be used to set the parameters for the test motor and to configure the different macro settings.

The control panel of the VFD was set to the settings specified for torque control macro in [24], with specifications given for the test motor. The switching frequency of the VFD was changed from 8 kHz to 16 kHz to give the motor a quicker response time.

The VFD comes applied with a 24V DC reference port for the digital signal. This reference will not be used since the MicroAutoBox has the possibility for providing this voltage through its battery supply.

A limiting factor of the MicroAutoBox is that it only allows for receiving or sending analog voltage signals. This is a problem when it comes to the VFD which only supports analog current signals from the AO port in the range 4-20mA. A solution was to bridge the analog out and

[1] Using a trigger to activate the analog input signal.

analog out ground port of the VFD with a 100Ω resistance. This gives a voltage in the range of 0.4-2V, that the MicroAutoBox can read.

4.3 Setting up the servo drive

The servo drive was originally set up with parameters and a FB diagram provided by Lucas-Nuelle. From the data sheet of the machine, it was found to be a difference between the parameters applied in the Lenze Engineer program loaded on the servo drive. Since the machine used was also a Lenze product, the parameters could easily be changed out by updating the machine settings of the program with a data set containing the exact parameters of the machine from the EASY package manager.

The FB diagram loaded on the servo drive contained the CAN connection to the control panel. The use of the control panel was not needed for this setup. The connection was disconnected, except for the CANopen out signal as this provides the screens of the control panel with measured speed and torque and did not interfere with the rest of the setup. The CANopen in port was providing the driver with speed and torque setting and was rewired to the AI1 and AI2 respectively for remote control.

The output torque and speed were wired from the CANopen out port to the AO1 and AO2, where AO1 is sending out measured speed and AO2 is sending out measured torque, as seen in Appendix C.

The DI2 port was activated to provide the driver with a remote start/stop signal. Due to some hardwiring in the system, it was not possible to remote control the start/stop button on the control panel. The new FB can be seen in Appendix C, while the edits done for the torque and speed signal in can be seen in Figure 4.1.

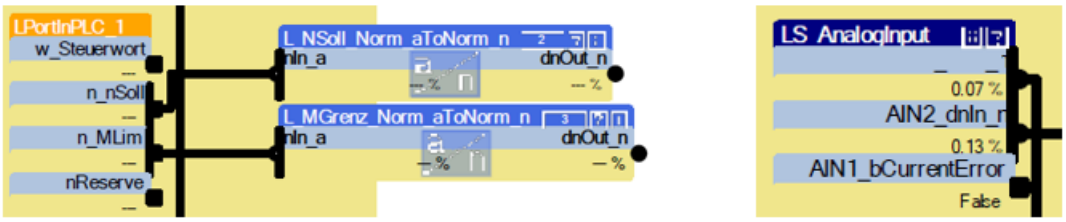

Figure 4.1: Showing a CAN in port on the left and the analog in ports on the right, in the FB diagram.

4.4 Turbine-Generator model

In the turbine-generator model, the test motor will be forward or reverse motoring, while servo machine as generator in forward or reverse braking. The synchronous speed of the test motor is 1000 RPM, which also will be the setpoint for the generator. Making the generator operating in forward braking and the turbine in forward motoring as can be seen in Figure 2.2. Where the

shaft between the two machines is spinning in the same direction and the torque in opposite, giving an increased torque load to the generator when the torque is increasing on the motor.

The biggest difference between VFD and servo drive is that the settings are fixed for the VFD meaning that a signal must be sent to the controller to define direction of the test motor and the control type (speed/torque control), while servo drive uses ± 10V for the analog ports that defines the direction of the servo machine.

From Figure 4.2 an overview of the test model can be seen. This model is setting the direction of the servo machine to forward braking and the test motor to forward motoring.

The model works in the way that it ramps up the servo machine first to 1000 rpm before the test motor gets loaded. After a while the torque load on the test motor is dropping to illustrate how the reduce water into the turbine. The input and output signal of this model is labelled SM for servo machine and TM for test motor, to make it easier to configure in ConfigurationDesk.

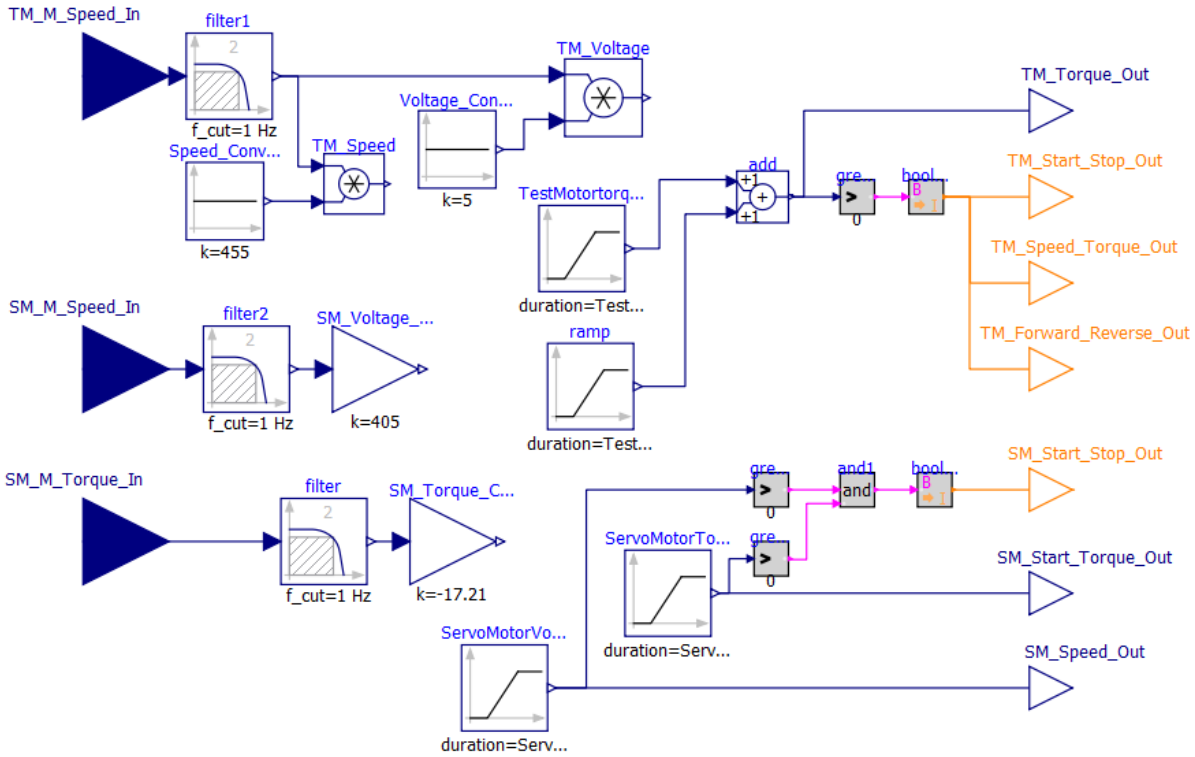

Figure 4.2: Showing the simple test model, created for the turbine-generator configuration.

4.4.1 Input signals

The model consists of three input signals, measured speed from the test motor and measured speed and torque from the servo machine. They are all connected to a filter, to remove measurement noise.

The nominal speed of the test motor is 910 rpm, and the voltage range goes from 0.4-2V, making a scaling factor of 455 for the measured speed.

For the original Lucas-Nuelle setup, the torque was taken out in the opposite direction as the measured and as the tenth of its value, while the speed was taken out the right direction without

any down scaling. With a voltage range of 0-10V, this gave a scaling factor of 405 for the speed and -17.21 for the measured torque of the servo machine.

4.4.2 Output signals

A voltage signal of 2.5V is sent to the AI1 port of the servo drive to give it a speed setpoint of 1000 rpm. The servo drive is also sent a 0.05V signal to the AI2 as a constant torque signal. The constant torque is needed to ramp up the speed to the setpoint, and to keep the speed at setpoint. The start/stop signal is staying high (1) when both a torque and speed signal is sent to the servo drive, and low (0) when one or both signals are sending zero voltage out.

The torque signal out to the VFD consists of two ramps, one ramping up the voltage from 0 to 8V, giving a torque of 80% (3 Nm) on the test motor. The ramp is starting 5 seconds after simulation and has a duration of 3 seconds. The second ramp is ramping down the torque from 8V to 4V, starting 20 seconds after simulation has started and has a duration of 3 seconds. In Figure 4.3 the torque output can be seen simulated in OpenModelica, with a timeframe of 300 seconds. The digital signal sent to the high (1) if the torque output is higher than zero, and low (0) when the torque output is zero or below.

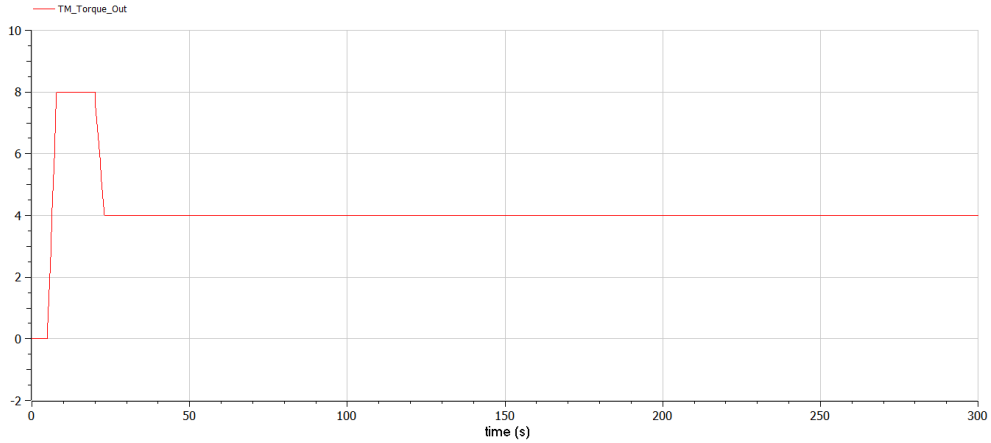

Figure 4.3: Shows the input voltage signa representing the torque setting of the motor that is sent to the VFD, simulated in OpenModelica.

The signals sent to the digital port on both the servo drive and VFD, consists of integer signals to ensure that there are not sent signals containing decimals. Digital signals are sent as 1 or 0. To ensure that the integer signal sends a signal of either 1 or 0, a greater block is used. The greater block is setting a Boolean value of true or false. This signal is then converted to integer, where true equals 1 and false equals 0.

4.5 Setting up the MicroAutoBox III

The MicroAutoBox was connected to the host computer and the ConfigurationDesk as specified in Appendix B [49]. To enable the MicroAutoBox to automatically start when plugging the power cable to the outlet, the remote-control cable was connected to the positive voltage of the power supply [27].

The Break-Out Box was connected to the DS1513 Multi I/O on the MicroAutoBox with a 156 -signal pin connection cable.

The ConfigurationDesk was used to manage the connections between the model, servo drive and VFD. By connecting the model signals to the I/O ports of Break-Out Box. The signal chain made can be seen in Figure 4.4 and Figure 4.5, while the ports used can be seen in Table 4.4.

Table 4.4: Showing the servo drive and the VFD ports connected to the MicroAutoBox.

MicroAutoBox:	Signal Port	Reference port
VFD:		
AI1	Z2	a2
AI2	Y2	a2
AO	V6	a2
DCOM	C2	N1 (High)
DI1	D2	A1
DI2	E2	A1
DI3	F2	A1
Servo Drive:		
AI1	S2	a2
AI2	T2	a2
AO1	U3	a2
AO2	V3	a2
DI2	D6	N1 (High)

The MicroAutoBox can deliver 24V DC from the power supply to the digital signals. The 24V DC supply is enabled by connecting a jumper cable between the N1 and M1 port on the Break-out Box. The ConfigurationDesk has as default set all the digital port to provide 5V, and the 24V DC supply must be activated on the bocks where this is wanted. This was done by changing the electrical interface of the digital block to high-side switch, enabling the high reference point (N1). For the VFD the DCOM circuit was set to active high-side switch, while for the servo drive the start/stop signal was set to active high-side switch [50].

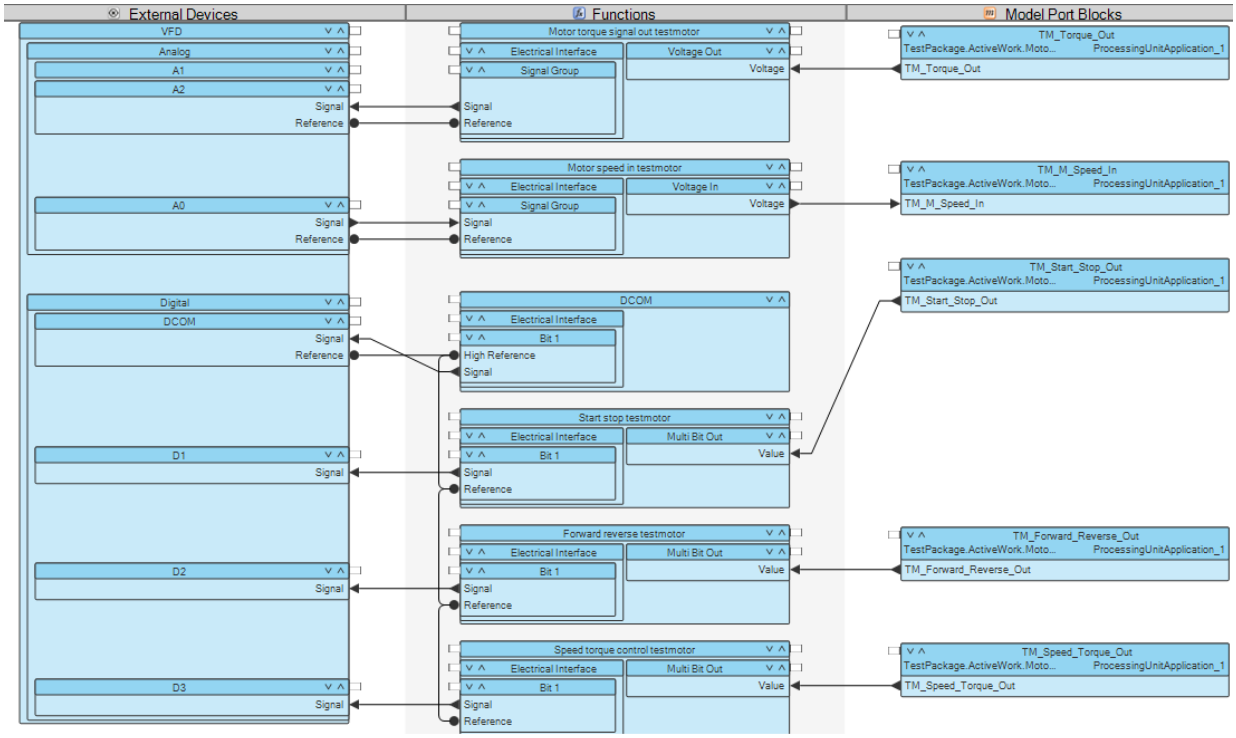

Figure 4.4: Shows the connection from the model to the I/O port of the VFD through the I/O ports the MicroAutoBox, in ConfigurationDesk.

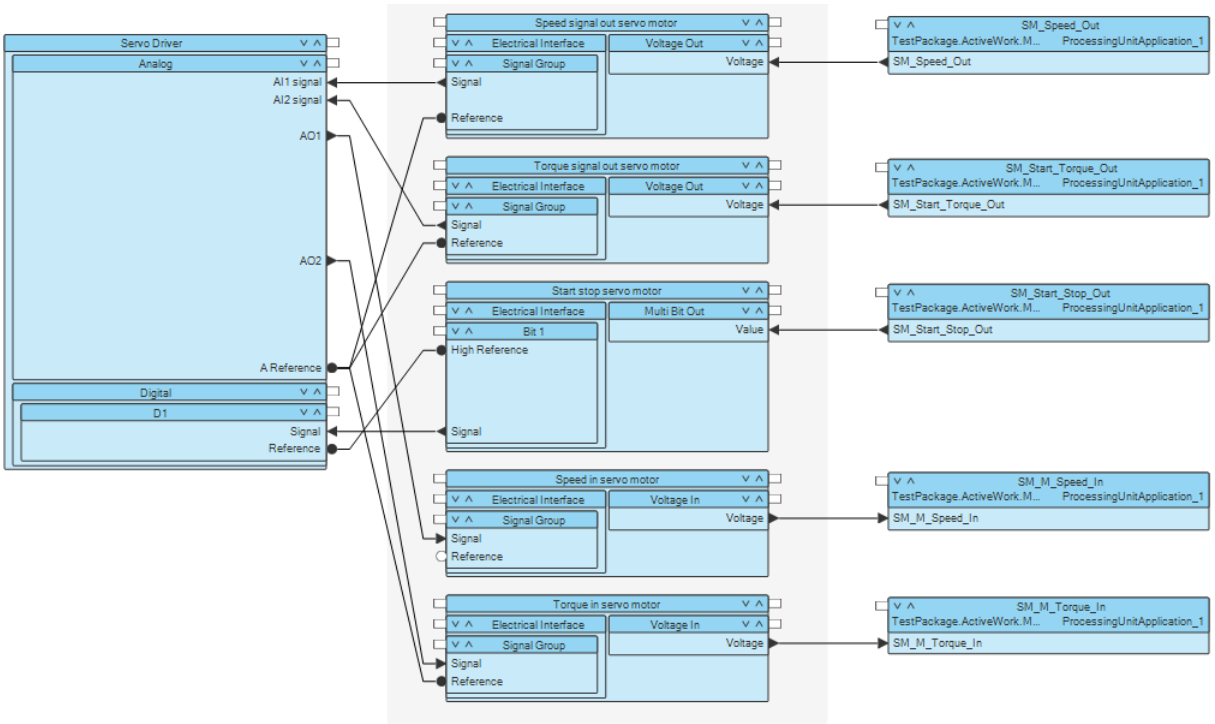

Figure 4.5: Shows the connection from the model to the I/O port of the Servo drive through the I/O ports the MicroAutoBox, in ConfigurationDesk.

4.6 Testing the HIL system

After the application was compiled, the SDF file created by ConfigurationDesk was uploaded to the ControlDesk for recording the HIL simulation. The simulation of the setup was recorded for a 60 second time frame.

From Figure 4.6, the measured torque and speed of the servo machine can be seen. The torque is having a high spike of negative 0.8 Nm in the start of the period when the speed of the servo machine is ramping up to nominal speed, before it falls to approximately negative 0.3 Nm. In the time frame six to nine seconds, the torque load on the servo machine was increased from negative 0.3 Nm to approximately negative 0.6 Nm, when the torque produced by the test motor was increased from 0 to 80%. At the twenty second mark the torque on the test motor is reduced to 40%, this results in a measured negative torque of 0.3 Nm on the servo machine.

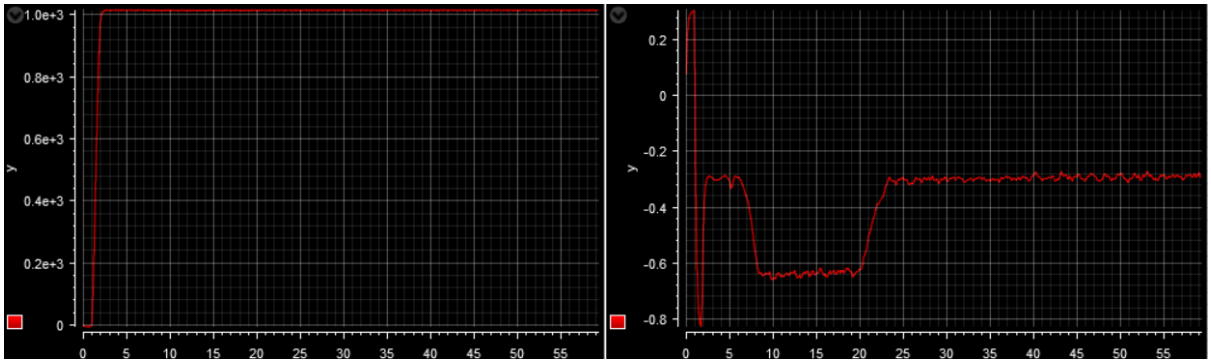

Figure 4.6: Show the measured speed (rpm) and torque (Nm) of the servo machine. Speed curve on the left and torque curve on the right.

The measured speed of the servo machine seen on the left side of Figure 4.6, has been zoomed in on in Figure 4.7.
After the speed reaches 1000 rpm, it stays approximately the same through the simulation, reducing only 2 to 3 rpm at the most, when the torque of the test motor is loaded at 80%.

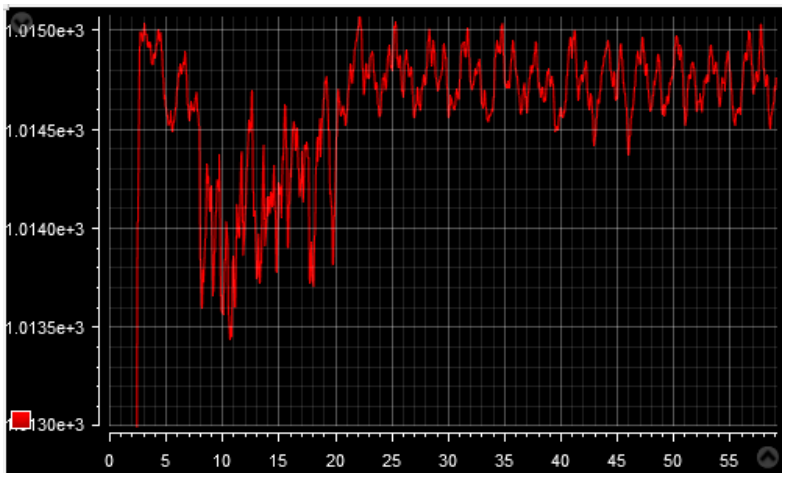

Figure 4.7: Shows the servo machine (rpm), zoomed in on the flat line as seen in Figure 4.6.

When scaling down to the first measured second for the torque and speed curve of the servo machine as seen in Figure 4.8, the measured speed of the servo machine is decreasing from zero to approximately a negative rpm of 6, before the speed setpoint of the servo machine signal is sent to the servo machine, while the measured torque is increasing to about 0.3 Nm, before it drops to negative 0.8 as seen in Figure 4.6.

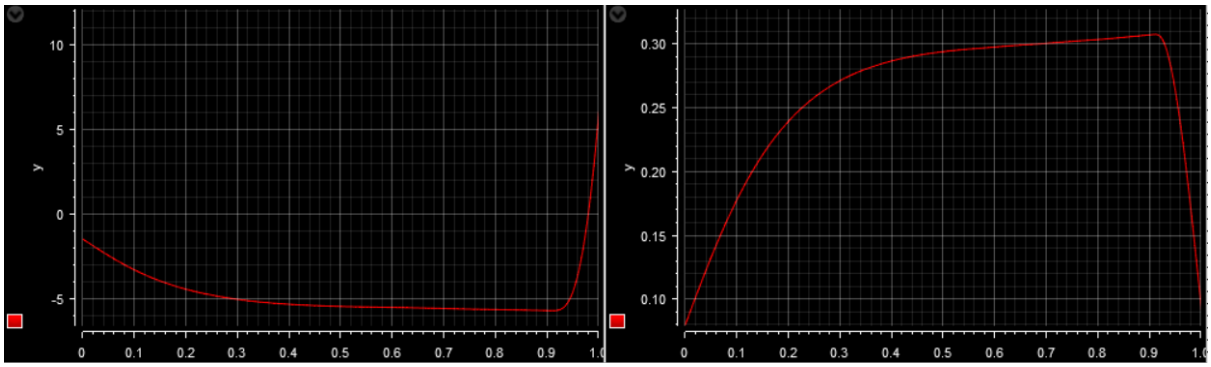

Figure 4.8: Showing the measurement in the time frame 0 to 1 second for both torque and speed. Speed curve seen on the left and torque to the right side.

In the Figure 4.9 the measured speed of the test motor can be seen. The speed is increasing from 0 to approximately 5 rpm, from time zero to approximately 0.8 seconds, and stays at 5 rpm until the torque is increased to 80%. The speed of the test motor is then increasing to approximately 19 rpm. After the time mark 20 seconds, the torque load is then decreased to 40% and the measured speed is also decreasing to be approximately the same as when there was no torque applied to the test motor. When observing at the first recoded second, it has similar shape (but inverted) as the curve of the servo machine seen in Figure 4.8.

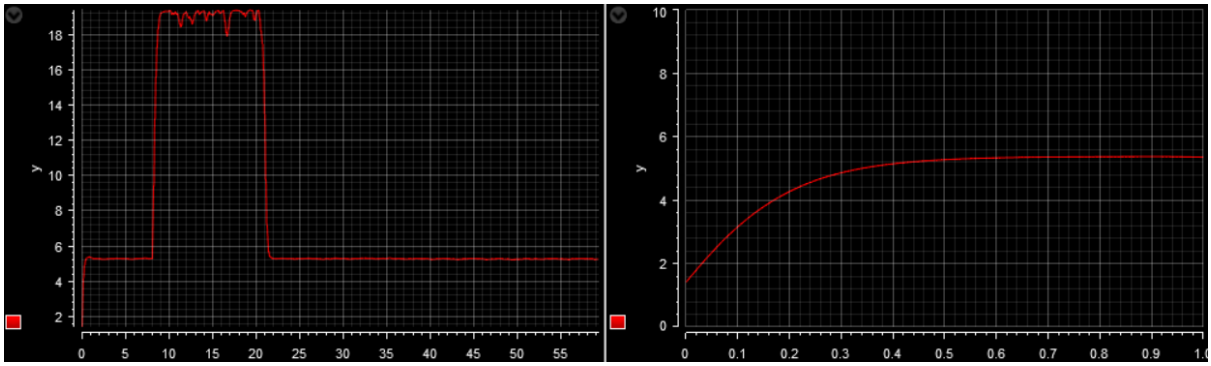

Figure 4.9: Showing the measured speed of the test motor for the whole simulation to the left and the first recorded second to the left.

5 Hydropower Simulation

For building the hydropower model, the OpenHPL library was used in OpenModelica. In this library there are several hydropower example models. For the hydropower model created in this chapter, the Simple example model from OpenHPL was used as foundation. The complete model and FMU can be found at GitHub [48].

The setup in this chapter is using the hardware connections made for the MicroAutoBox, the VFD and the servo drive described in chapter 4.5, and is also including the ConfigurationDesk setup.

5.1 Model

In Figure 5.1 an overview of the complete model can be seen, consisting of the three sub-models, Servo machine model (SMM), hydropower (HPM) and a Signal Checker (SC).
The total the model has two inputs and seven outputs. Measured speed and torque from the servo machine is the input sent into the model, while the model is sending out the direction and calculated torque load to the test motor, and the speed and initial torque is sent to the servo machine.

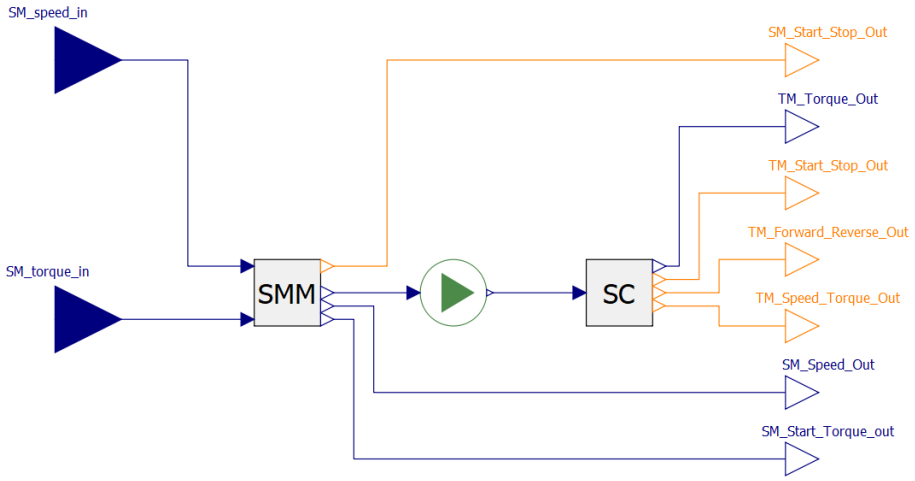

Figure 5.1: The complete hydropower model.

5.1.1 Hydro model

The Simple example model in the OpenHPL seen in Figure 5.2, consist of a reservoir, intake, surge tank, penstock, turbine, discharge and tail. A controller consisting of ramp signal is connected to the turbine. The ramp signal is decreasing the guide vane opening and the waterflow to the turbine. The model also includes a data set with initial parameters for the different parts of this system.

The modifications done to the Simple example model consist of two constant blocks to keep the reservoir and tail water at steady level, and that is also keeping the water flow constant through the model. An input for the measured angular velocity (W_in) from the servo machine is added. The power generated by the turbine is then divided by the W_in, which is giving a

torque signal that will be sent out to the test motor. A division by zero protector is also added to the W_in signal for preventing an error caused by the power being divided by zero. Due to size differences of model and the test motor, the torque signal is scaled down to a fitting level before it gets sent out to the motor. This scaling also includes a conversion from torque to voltage signal. The modifications can be seen in Figure 5.3.

The controller start time have reduced from 500 seconds, as initially set in OpenHPL Simple example model, to 50 seconds.

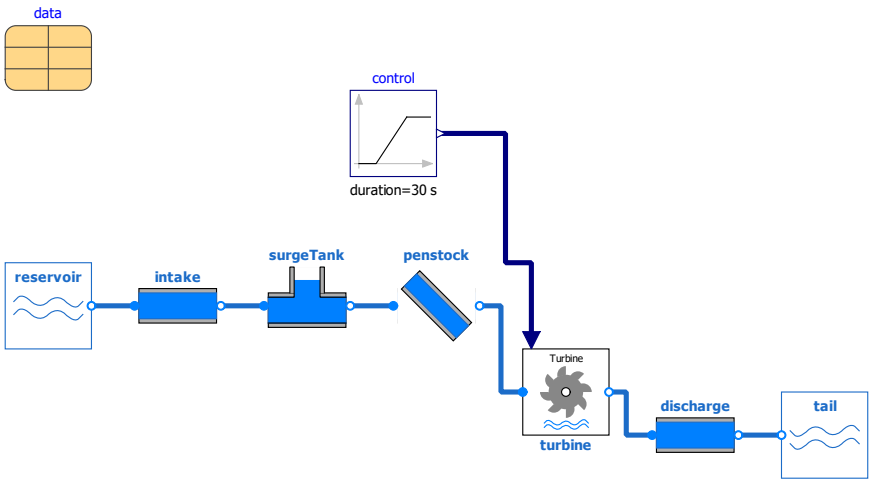

Figure 5.2: Shows the Simple example model from the OpenHPL.

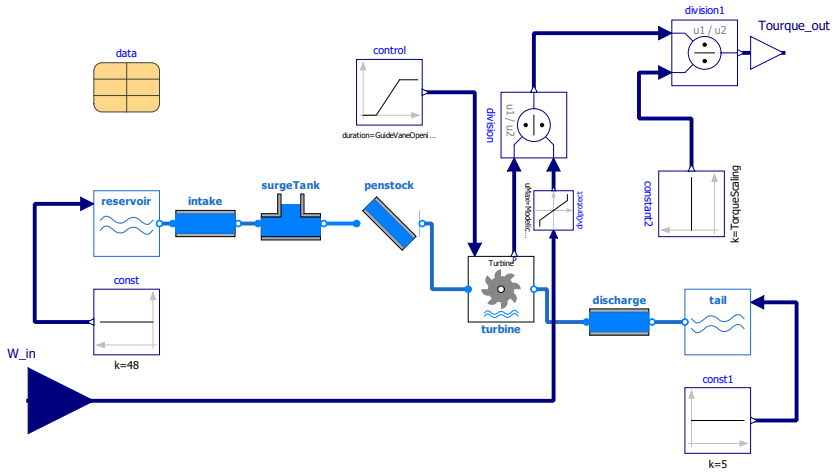

Figure 5.3: Shows the Simple example model from the OpenHPL, with modifications.

In Figure 5.4 the simulation of the waterflow in the penstock can be seen. When the guide vane opening is reduced it causes oscillations in the penstock.

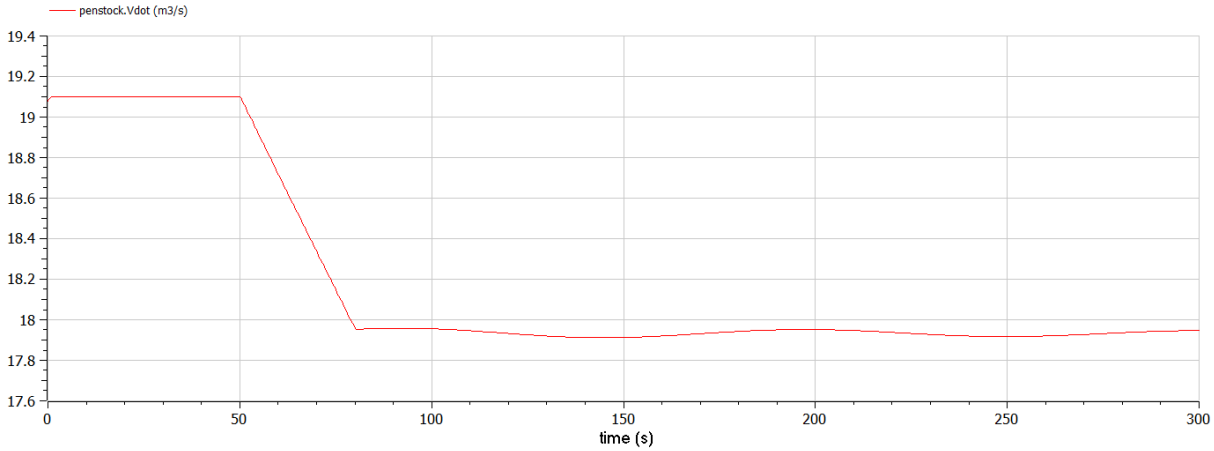

Figure 5.4: Shows the simulation of the water flow in the penstock, simulated in OpenModelica.

5.1.2 Servo machine model

The servo machine model is based on the model described in chapter 4.4, with the same parameters for the speed and start torque. The start time of the torque and speed signals was reduced from 1 to 0.1 seconds. The model also consists of two inputs, where AO1 is the measured speed of the servo machine and the AO2 is the measured torque. A filter of the same type as used in chapter 4.4 was added to the measured torque signal to remove measurement noise. The torque signal is scaling by a factor of negative 17.21. For the speed signal in, it has a conversion from rpm to angular velocity and a scaling of 405, before the signal is sent out to the hydropower model.

A delay is added to the measured angular velocity, setting the first 0.1 seconds of the measurement to zero.

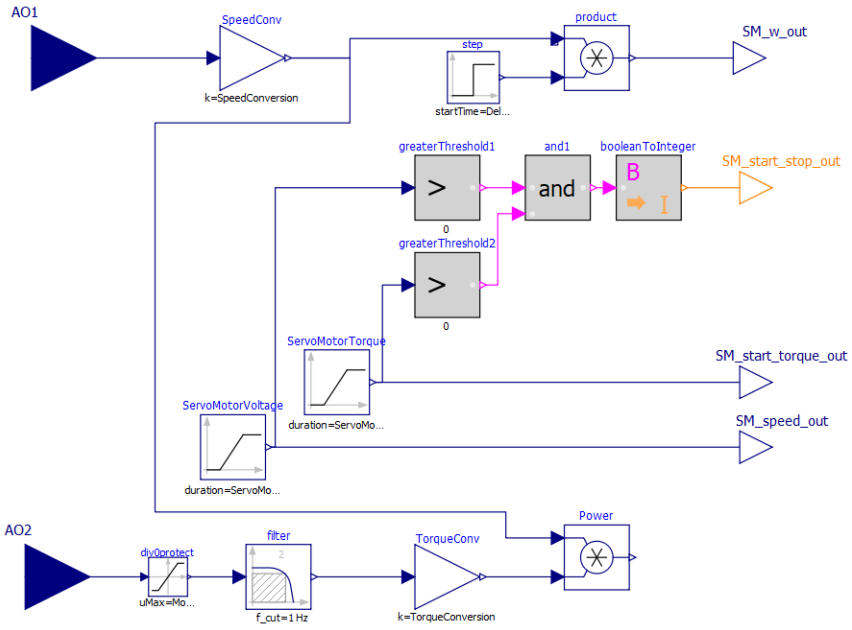

Figure 5.5: Showing an overview of the Servo machine model.

5.1.3 Signal checker

The signal checker is used for checking that the voltage signal send from the hydropower model is within the range of what the VFD and MicroAutoBox is capable of withstand. The model also includes the same principle for digital signals as described in chapter 4.4.2.
An overview of the model can be seen in Figure 5.6.

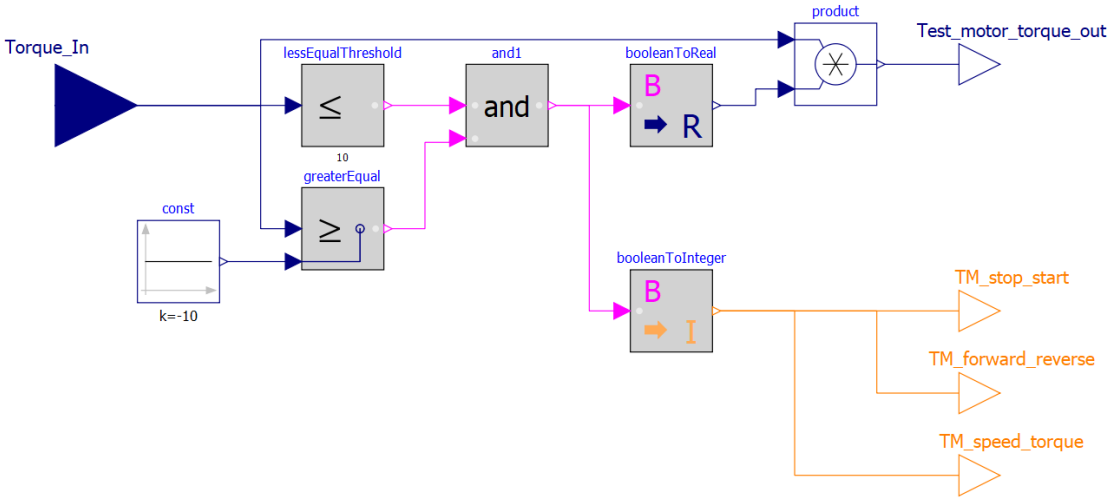

Figure 5.6: Showing an overview of the signal checker model.

By using the hydropower model, with a ramp signal representing the measured servo machine speed, the output signal from the signal checker can be simulated, as seen in Figure 5.7.
The signal checker is blocking the high start signal as seen in the beginning, with the first registered start value of 10, before it quickly drops to a bit below 8.

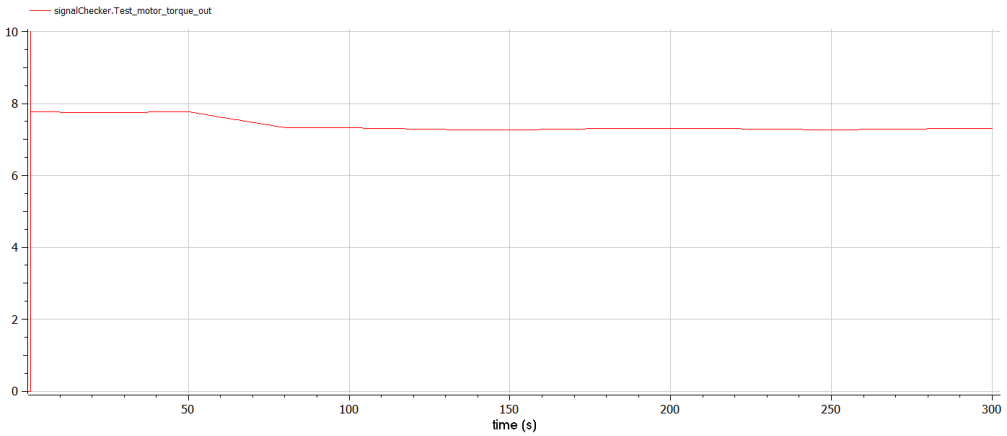

Figure 5.7: Showing the simulated torque signal from the signal checker model in OpenModelica.

5.2 Simulation

With a simulation time of 300 seconds, the model was uploaded to the MicroAutoBox and was recorded by the ControlDesk.

The recorded angular velocity and torque of the servo machine can be observed in Figure 5.8. Once the machine reaches the nominal speed of 1000 rpm, it maintains a steady angular velocity of 106 radians/seconds. By examining the graph in Figure 5.9, which has been zoomed in to display the interval between zero and nominal speed, it can be seen that it takes approximately 0.9 seconds for the machine to reach its nominal speed, when having a starting time of 0.1 second. Furthermore, there is a decrease in the recorded angular velocity from zero to a negative value of 0.6, which occurs during the timeframe of 0.05 to 0.07 seconds.

It can be observed that the recorded torque of the servo machine reaches approximately negative 0.8 Nm around time 0.8 seconds, as depicted in Figure 5.10. After the guide vane opening decreases, the torque falls to above negative 0.5 Nm, where it stays out the simulation time, as observed in Figure 5.8.

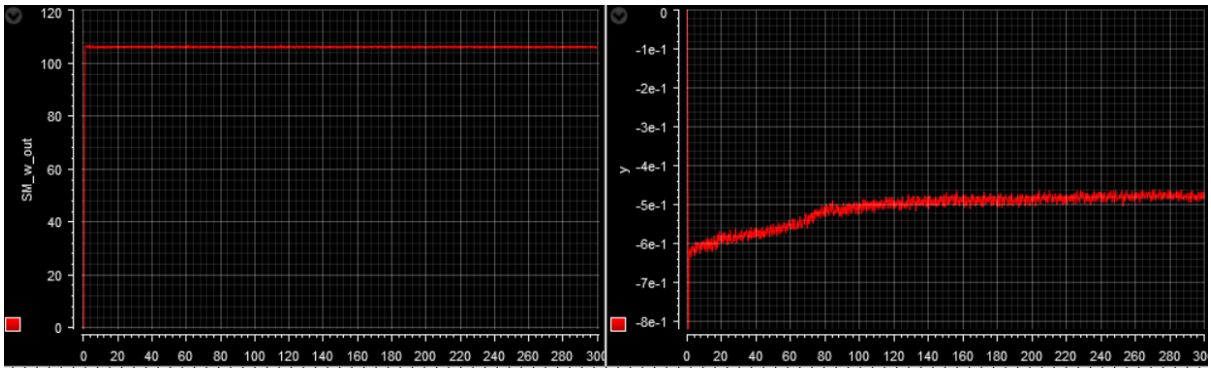

Figure 5.8: Shows the measured angular velocity and torque of the servo machine. Angular velocity on the left and torque on the right side.

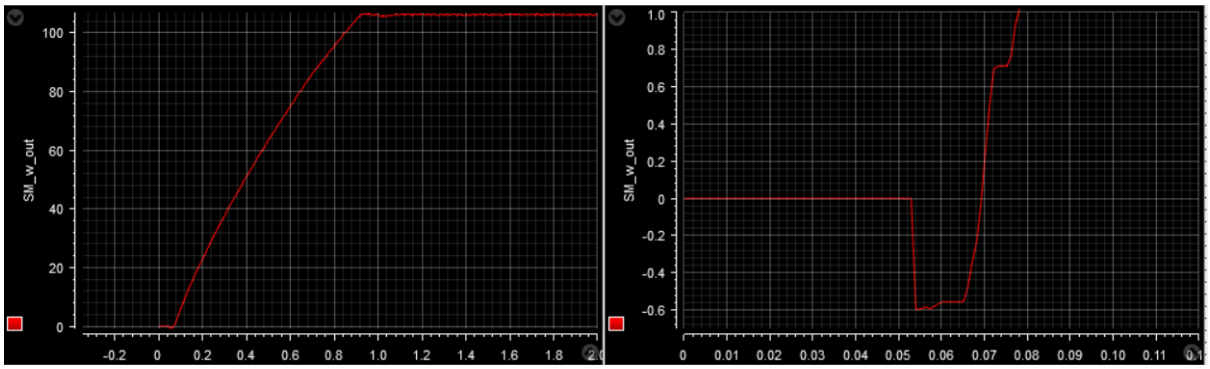

Figure 5.9: Shows the measured angular velocity of the servo machine, for the time frame 0 to 2 and 0 to 0.08 seconds, in radians/seconds.

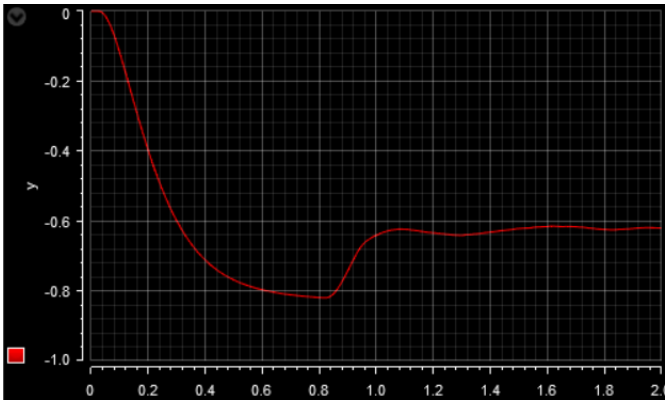

Figure 5.10: Shows the first two seconds of the measured torque in Nm.

Figure 5.11 exhibits both the input and output of the signal checker. It can be observed that the input signal for a short time frame is reaching $8*10^{62}$, while the output signal is reaching 10 at the most.

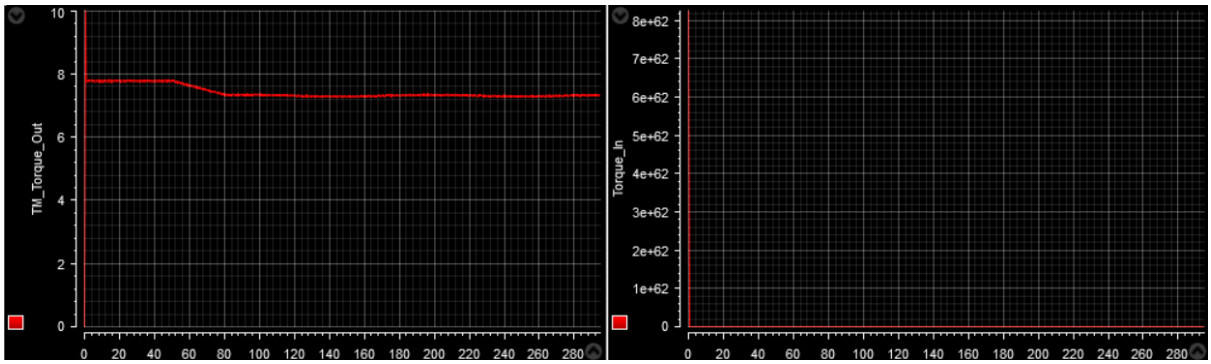

Figure 5.11: Shows the torque signal in and out of the signal checker. Torque out to the left and torque in on the right side.

The calculated power of the servo machine can be seen in Figure 5.12. Since the calculated power is negative, this means that the servo machine act as a generator and is generating power.

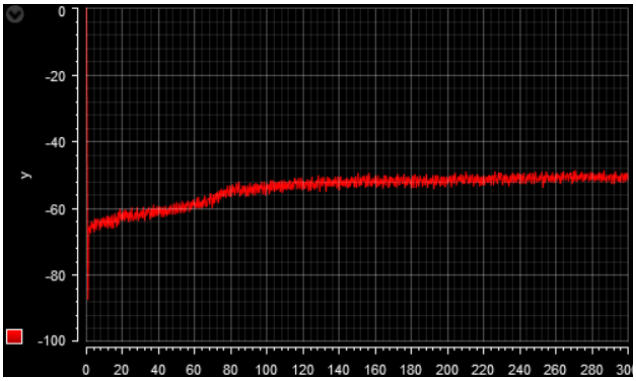

Figure 5.12: Shows the power generated by the servo machine.

6 Discussion

A prefix for the model identifier is not included in FMUs generated by the 1.21.0 version of OpenModelica. The prefix needs therefore to be specified in ConfigurationDesk in order to use the FMUs generated by the 1.21.0 version of OpenModelica. Appendix B provides information on how to specify the prefix in ConfigurationDesk. Additionally, the multiarch and crossbuild are not updated for the 1.21.0 version of OpenModelica, meaning that it is not possible to generate binary code using a Docker desktop.

In the case of generating an FMU that is stored inside a package in the modelling tool, the FMU will by default take the name “package.model.fmu”. This naming convention works fine in ConfigurationDesk, but when the model is inside multiple packages, the resulting FMU may be named as “package.package.packag.model.fmu”. This can cause errors when compiling the application in ConfigurationDesk. However, changing the task name in ConfigurationDesk may resolve the issue in some cases. Similar issue can occur in OpenModelica, but this only occurs with more complex names. A solution could have been to change the name of the FMU when creating it in OpenModelica, but this functionality is not working properly and is only resulting in changing the model identifier and model name inside the FMU, and not the file name. An alternative solution that works fine is to extend the model to a higher level of the package or outside the package and creating an FMU from there. This method also solves the issue in ConfigurationDesk.

In ConfigurationDesk and ControlDesk, when uploading an application to the MicroAutoBox that is set to start instantaneous. The time step of the application is moved forward, making the signals of the application being activated before they are supposed to. By rather starting the application after it is uploaded, the timestep will be at the right seps.

Instead of using a switch, an add block was utilized for the model in chapter 4. This was done because of in certain instances, switching between two signals caused a momentary drop to zero for a period of one to two milliseconds.

As the torque is increased on the test motor, the motor starts to rotate. This is because when torque increases, the angular acceleration is increasing and hence increases the angular velocity. The direction of the measured speed is showed positive when it is supposed to be negative, this is because of the VFD only sending out a current in the range 4-20 mA, only showing the speed and not the direction of the speed.

To accelerate the servo machine to the desired speed, it was necessary to provide a steady torque signal to the servo machine, otherwise it would only accelerate to roughly 300 rpm. In the setup provided by Lucas-Nulle, when running the servo machine on speed control, a constant torque signal was sent to the motor. A value of 0.05V was chosen because it was sufficient to get the servo machine up to setpoint without jerking.

From both chapter 4 and 5 it was measured speed and torque of the test motor and servo machine when no initiate signal is send to the servo driver and VFD. An interesting observation made, when a signal sent to activate the servo drive without sending any signals for torque and speed, the machine rotated at roughly the same magnitude as the measured under simulation. When investigating this further, it was found to be a voltage of approximately 0.011 V using a Multimeter, staying active on the circuit that was causing this fault measurements. This voltage was probably a result of a leakage current in the system.

The OpenHPL library uses short-cut references, which is not supported by OpenModelica. This caused problems when exporting FMUs containing certain items in the OpenHPL library. To address this problem, the model used in chapter 5 was exported as and FMU through Dymola, where short-cut references is supported.

OpenModelica is an open-source modelling tool, which comes with the marks of being open-source, such as bugs, crashing etc. however, it is a free tool that works fine for the most when using the Modelica library for generating FMUs. When it comes to Dymola where everything works properly, it comes with its drawbacks in the form of high license cost and that there is needed for a separate license for generating FMUs. Depending on the usage case and budget of the user, the evaluation of which modelling tool to use must be done.

7 Conclusion

In conclusion, the VFD and servo drive of the servo test stand can be effectively controlled by a real-time processing unit, like the MicroAutoBox III from dSPACE, through the analog and digital I/O ports.

It has been proved that by creating FMUs of both a simple model and an existing hydropower model, such as the Simple example model from OpenHPL, it is feasible to utilize them to control the servo test stand. However, when using the 1.21.0 standard version of OpenModelica, it was found to be quite tricky to generate FMUs that the ConfigurationDesk would compile since the prefix of the model identifier was not included in the FMUs generated. By changing to the newest nightly version 1.22.0, the FMUs could be generated as binary and source code that could be compiled by ConfigurationDesk.

Both versions of OpenModelica had the same problem with not supporting short-cut references, as used in the OpenHPL hydropower library. This limed OpenModelica to be used to generated hydropower FMU, and had to be switched to Dymola, which is supporting short-cut references, to generate FMUs based on the OpenHPL library.

Additional effort for this setup may be exploring the integration of a hydropower station with a grid, either as a single FMU or as multiple FMUs. Creating a standardized storage system for the measured data could also be a further development step.

References

[1] IEA, ‘World Energy Outlook 2022 – Analysis’. https://www.iea.org/reports/world-energy-outlook-2022 (accessed May 03, 2023).

[2] A. S. Kvalsund, ‘Development of an open control interface for a servo machine test stand’, Master thesis, University of South-Eastern Norway, 2022. Accessed: Apr. 28, 2023. [Online]. Available: https://openarchive.usn.no/usn-xmlui/handle/11250/3011347

[3] Dassult Systems, ‘Dymola - Dassault Systèmes®’. https://www.3ds.com/products-services/catia/products/dymola/ (accessed May 08, 2023).

[4] OpenModelica, ‘Introduction’. https://openmodelica.org/ (accessed May 08, 2023).

[5] modelonms, ‘What is FMI?’, Sep. 12, 2019. https://modelon.com/blog/functional-mock-up-interface-fmi/ (accessed Apr. 30, 2023).

[6] P. Schavemaker and L. Van der Sluis, *Electrical power system essentials*, Second edition. Chichester, West Sussex: John Wiley & Sons, Ltd, 2017.

[7] S. J. Chapman, *Electric machinery fundamentals*, 5th ed. New York: McGraw-Hill, 2012.

[8] Lesics, ‘How does an induction motor work?’ https://lesics.com/how-does-an-induction-motor-work.html (accessed Apr. 30, 2023).

[9] BYJUS, ‘Faraday’s Laws of Electromagnetic Induction - Lenz’s Law, Formula, Derivation, Video, and FAQs’, Feb. 26, 2020. https://byjus.com/physics/faradays-law/ (accessed May 03, 2023).

[10] Electrical Deck, ‘What is Slip in Induction Motor? - Effect of Slip on Induction Motor’, *Electrical Deck - All about Electrical & Electronics*, Nov. 20, 2020. https://www.electricaldeck.com/2020/11/slip-in-induction-motor-and-effect-of-slip-on-induction-motor.html (accessed May 03, 2023).

[11] Electrical Concepts, ‘Torque Slip Characteristics of Induction Machine’, *Electrical Concepts*, Nov. 09, 2017. https://electricalbaba.com/torque-slip-characteristics-of-induction-machine/ (accessed May 03, 2023).

[12] Danfoss, ‘What is a variable frequency drive?’ https://www.danfoss.com/en/about-danfoss/our-businesses/drives/what-is-a-variable-frequency-drive/ (accessed Jan. 17, 2023).

[13] Thomas Publishing Company, ‘All About AC Motor Controllers - What They Are and How They Work’. https://www.thomasnet.com/articles/instruments-controls/ac-motor-controllers/ (accessed Jan. 26, 2023).

[14] Thomas Publishing Company, ‘All About Servo Motor Controllers - What They Are and How They Work’. https://www.thomasnet.com/articles/instruments-controls/servo-motor-controllers/ (accessed Jan. 26, 2023).

[15] Advanced Micro Controler, ‘Tutorial: What Is A Resolver?’ https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-resolver/ (accessed Apr. 28, 2023).

[16] fmi-standard, ‘Functional Mock-up Interface’, *The leading standard to exchange dynamic simulation models*. https://fmi-standard.org/ (accessed May 08, 2023).

[17] fmi-standard, ‘Functional Mock-up Interface Specification’. https://fmi-standard.org/docs/3.0/ (accessed Feb. 20, 2023).

[18] Tennessee Valley Authority, ‘File:Hydroelectric dam.svg - Wikipedia’, Aug. 18, 2000. https://commons.wikimedia.org/wiki/File:Hydroelectric_dam.svg (accessed Apr. 23, 2023).

[19] D. Winkler, ‘Lecture Transfer function based model’. Apr. 25, 2023.

[20] Lucas-Nuelle, ‘Dynamic servo machine test system for 0.3kW machines incl. software ActiveServo’. https://www.lucas-nuelle.us/2776/pid/22637/apg/11298/Dynamic-servo-machine-test-system-for-03kW-machines-incl-software-ActiveServo.htm (accessed May 08, 2023).

[21] dSPACE, ‘MicroAutoBox III’, *Compact and robust in-vehicle prototyping system*. https://www.dspace.com/en/inc/home/products/hw/micautob/microautobox3.cfm (accessed Jan. 11, 2023).

[22] Lenze, ‘MCA asynchronous servo motors__v4-0__EN.pdf’, Accessed: Apr. 16, 2023. [Online]. Available: https://download.lenze.com/TD/MCA%20asynchronous%20servo%20motors%20%20__v4-0__EN.pdf

[23] ABB, ‘3GVA083001-ASB | motor manual’. http://new.abb.com/products/3GVA083001-ASB/3gva083001-asb (accessed Apr. 24, 2023).

[24] ABB, ‘EN_ACS350 UM_D.pdf Variable frequency drive manual’. [Online]. Available: https://library.e.abb.com/public/2cf5b5aabb5777a9c125733d00407394/EN_ACS350%20UM_D.pdf

[25] Lenze, ‘Lenze 9400 Servo Driver Reference Manual’, Accessed: Mar. 22, 2023. [Online]. Available: https://download.lenze.com/TD/E94AxHE__Servo%20Drives%209400%20HighLine%20(from%20Firmware%2001-50)__v15-0__EN.pdf

[26] Lenze, ‘9400 HighLine servo inverter’. https://www.lenze.com/en-de/products/inverters/servo-inverters/9400-highline-servo-inverter (accessed Jan. 18, 2023).

[27] dSPACE, ‘User Guide’, *MicroAutoBoxIIIHardwareInstallationandConfiguration.pdf*, 2020.

[28] dSPACE, ‘MicroAutoBox Break-Out Box’. https://www.dspace.com/en/inc/home/products/hw/micautob/microautobox2/mabx_breakoutbox.cfm (accessed Jan. 18, 2023).

[29] Traco Power, ‘Certificate of Conformity’, Accessed: Jan. 11, 2023. [Online]. Available: https://www.tracopower.com/int/series/tsp

[30] Traco Power, ‘TSP 600-124 power supply user manual’. https://www.tracopower.com/int/model/tsp-600-124 (accessed Jan. 11, 2023).

[31] OMEGA Engineering, ‘Digital I/O Functionality’. https://www.omega.co.uk/literature/transactions/volume2/digitalio.html# (accessed May 09, 2023).

[32] Electrical Classroom, ‘Difference between DI, DO, AI, AO| Digital IO and Analog IO’, Nov. 24, 2019. https://www.electricalclassroom.com/digital-i-o-and-analog-i-o/ (accessed May 10, 2023).

[33] dSPACE, ‘ConfigurationDesk’, *Configuration and implementation software for dSPACE real-time hardware*. https://www.dspace.com/en/inc/home/products/sw/impsw/configurationdesk.cfm (accessed May 08, 2023).

[34] dSPACE, ‘ControlDesk’, *Universal experiment software for ECU development*. https://www.dspace.com/en/inc/home/products/sw/experimentandvisualization/controldesk.cfm (accessed May 08, 2023).

[35] Lenze, ‘EASY Engineering’, *EASYEngineeringToolsv3-0ENpdf*, Accessed: May 08, 2023. [Online]. Available: https://www.lenze.com/fileadmin/lenze/documents/en/flyer/EASY_Engineering_Tools__v3-0__EN.pdf

[36] MathWorks, ‘Simulink - Simulation and Model-Based Design’. https://www.mathworks.com/products/simulink.html (accessed May 08, 2023).

[37] ESI Group, ‘SimulationX | System Simulation Software’. https://www.esi-group.com/products/simulationx (accessed May 08, 2023).

[38] dSPACE, ‘User Guide’, *ConfigurationDeskIntroductionandOverview.pdf*, no. b, 2022.

[39] dSPACE, ‘User Guide’, *ConfigurationDeskRealTimeImplementationGuide.pdf*, no. b, 2022.

[40] dSPACE, ‘User Guide’, *ConfigurationDeskUserInterfaceReference.pdf*, no. b, 2022.

[41] dSPACE, ‘User Guide’, *ControlDeskIntroductionandOverview.pdf*, no. b, 2022.

[42] dSPACE, ‘User Guide’, *ControlDeskCalibrationandDataSetManagement.pdf*, no. b, 2022.

[43] dSPACE, ‘User Guide’, *ControlDeskMeasurementandRecording.pdf*, no. b, 2022.

[44] Lenze, ‘L-force 9400 Servo Driver parameter setting and configuration.’, Accessed: Mar. 20, 2023. [Online]. Available: https://download.lenze.com/TD/E94AxHE__Servo%20Drives%209400%20HighLine%20(Firmware%2001-37)__v1-6__EN.pdf

[45] Docker Inc, ‘Docker Desktop’, May 08, 2023. https://docs.docker.com/desktop/ (accessed May 08, 2023).

[46] D. Winkler, ‘OpenHPL’. USN. Accessed: Apr. 20, 2023. [Online]. Available: https://openhpl.opensimhub.org/

[47] M. De Castro *et al.*, ‘Version [OpenIPSL 2.0.0] - [iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations]’, *SoftwareX*, vol. 21, p. 101277, Feb. 2023, doi: 10.1016/j.softx.2022.101277.

[48] B. Haugnes, ‘GitHub model repository’. https://github.com/haugnesb/Development-of-a-HIL-system-for-simulating-FMUs-models (accessed May 09, 2023).

[49] dSPACE, ‘User Guide’, *MicroAutoBoxIIGettingStarted.pdf*, no. b, 2022.

[50] dSPACE, ‘User Guide’, *ConfigurationDeskFunctionBlockProperties.pdf*, no. b, 2022.

Appendices

Appendix A Task description.

Appendix B Working with dSPACE.

Appendix C FB diagram.

Appendix A: Task description

University of South-Eastern Norway
Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Development of HiL system for hydropower simulations based on FMI technology

USN supervisor: Dietmar Winkler

External partner: Electrical Power Systems research group (USN)

Task background:
A servo machine test stand (depicted below) is coupled to a small, frequency converter controlled, asynchronous machine. This test stand shall be used as an example implementation of a Hardware-In-the-Loop (HiL) system that can be controlled by a simulation outputs of a digital twin representing a complete hydro power system. The digital twin model shall be exported as a functional mock-up unit (FMU) that can be loaded into a MicroAutoBox III real-time PC unit and which then controls the test motor and servo brake the test stand.

Task description:
The following tasks should be carried out:

1. Analyse the components used in the servo machine test stand
2. Analyse the interface capabilities and restrictions of installed hardware of both the servo and the load machine.
3. Analyse and setup the dSPACE MicroAutoBox III system and its ConfigurationDesk software interface.
4. Implement the necessary hardware input and output connections for the servo drive and dSPACE MicroAutoBox III
5. Investigate the process of exporting an FMU from an existing hydropower system model and importing this into the dSPACE MicroAutoBox III real time PC in order to run a Hardware-in-the-Loop simulation
6. Evaluate the possibility of controlling the servo test stand with a PC by either using a simulation tool directly or via FMUs
7. Document the work and in a report.

It should be a goal for the candidate to publish the results of the work, e.g., at a conference or journal.

Student category: EPE

Is the task suitable for online students (not present at the campus)? no

Practical arrangements:
The work needs to be carried out on campus since access to the hardware is essential.

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature):

2023-01-24 D. Dunker

Student (write clearly in all capitalized letters):

BENJAMIN HAUGNES

Student (date and signature):

2023-01-24 Benjamin Haugnes

Appendix B: Working with dSPACE.

Working With dSPACE

This document contains the following:

- Setting up the MicroAutoBox and connecting it to the ConfigurationDesk.
- Building a simple real-time application for the MicroAutoBox
- Setting up a test environment in ControlDesk for a real time application.

The guide is just briefly scratching the surface of what the ConfigurationDesk and ControlDesk is providing and is just meant to be a quick guide for understanding how the program works and how to create a simple working environment. For more complex tasks see the dSPACE help or the links referred to in this document.

Installation of dSPACE software.

The easiest way to install the dSPACE programs is either by merging the two CDs delivered with the dSPACE product or by downloading the merged folder from their web side. To get a successful installation, external firewalls must be deactivated, and the 8dot3 file name creation must be disabled. The 8dot3 file name creation can be changes by following these steps:

1. Run regedit.exe.
2. Go to \HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem.
3. Right click on the NtfsDisable8dot3NameCreation and set it to 0 for disabling it.
4. Reboot the computer for implementing the change.

When the dSPACE programs are installed, the licences must be activated in the dSPACE licence manager with the CmStick connected to the computer.

Setting up the MicroAutoBox and connecting it to the ConfigurationDesk.

The simplest way to connect to the MicroAutoBox is to connect it directly to the host pc through an ethernet cable connected to the host port on the MicroAutoBox. To access the

MicroAutoBox directly, the configuration of the host pc ethernet port must be set to a static IPaddress. From the manufacturer the MicroAutoBox has the following default IP-address 192.168.140.010. The IPv4 on the ethernet port must be set to a static IP-address in the range 192.168.140.001 and 192.168.140.254, and with the sub network 255.255.255.0. Important to remember not to set the static IP-address to the same as the MicroAutoBox, then they will not communicate [1].

The method using LAN and DHCP server, see [2] "*chapter setting up the connection to the host pc"*. The MicroAutoBox can also be accessed through WLAN, but not for the first run time, due to this functionality is disabled as default from the manufacture. The WLAN has to be enabled in the web interface of the MicroAutoBox or through the dSPACE command

prompt (cmd), while the MicroAutoBox is connected to the host pc through an ethernet connection [2].

When the static IP-address is set and the MicroAutoBox is turned on, it can be accessed through the web interface by using the IP-address of the MicroAutoBox for changing configurations seen on Figure 1.

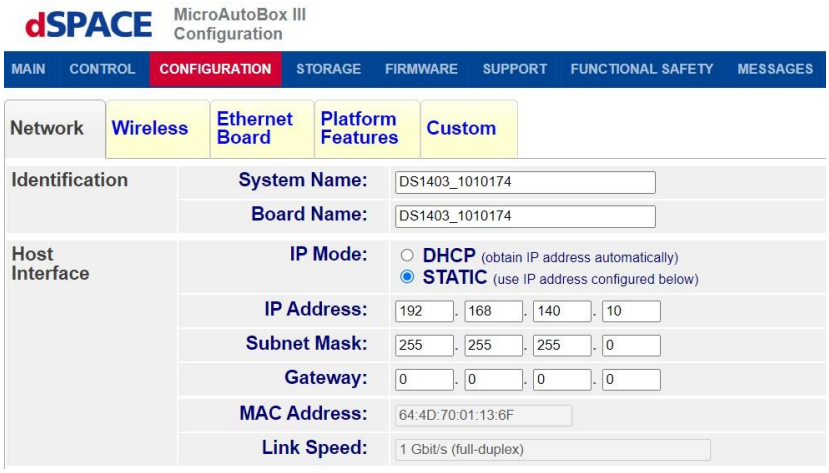

Figure 1: Showing the configuration window in the web interface for the MicroAutoBox.

To connect the MicroAutoBox to the ConfigurationDesk, is done by opening the register platform in the ConfigurationDesk and typing in the IP-address, serial number or the MACaddress of the MicroAutoBox seen in Figure 2.

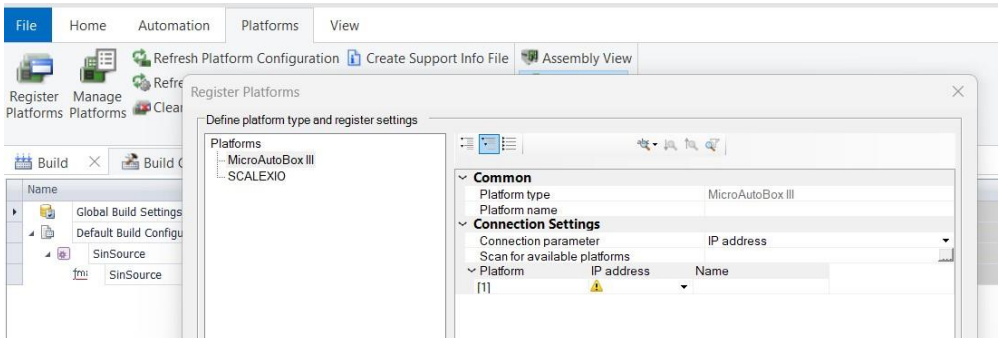

Figure 2: Showing how to add the MicroAutoBox to the ConfigurationDesk.

The next step is to build a real-time application to run on the MicroAutoBox.

Building a simple real-time application for the MicroAutoBox.

This document contains the building process of a real-time application based on an FMU model with the 8 steps as seen in Figure 3.

Tips: by clicking on the help button, all the documentation for the MicroAutoBox, ConfigurationDesk and ControlDesk can be retrieved as PDFs.

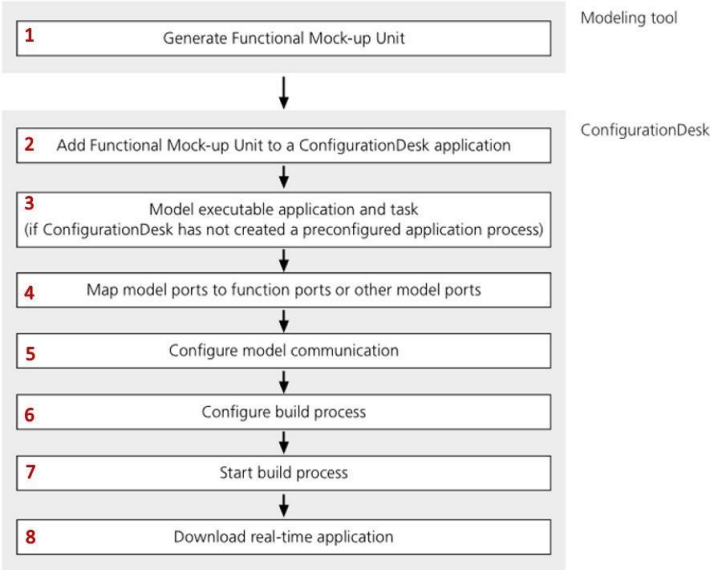

Figure 3: Shows the step-by-step approach to build a real-time application [3].

Before starting the building process, the Figure 4 shows how to navigate the ConfigurationDesk program. It might make it a bit easier to follow the step-by-step approach described later.

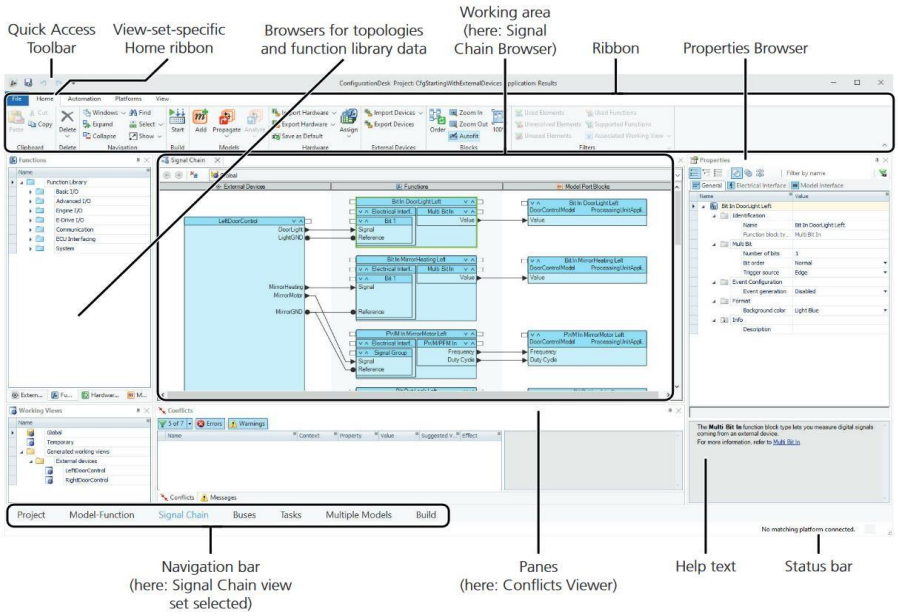

Figure 4: Shows the different properties and functionality of the ConfigurationDesk [3].

Step 1: Creating an FMU for ConfigurationDesk

The ConfigurationDesk supports FMU type 2.0 running on the FMI 2.0.1 architecture, and it can only guarantee support for the following modelling tools:

- Dymola
- SimulationX
- MapleSim

The ConfigurationDesk will regardless of the FMU is containing only source code, binary code or both, it will under the build process of the application compile it to a Linux arm based 32 binary code that the MicroAutoBox can read. It is recommended to at least support the FMU with source code (C code) due to this being independent of the operations system it is created by and can be read by all the different operation system architecture [4].

In theory the ConfigurationDesk should be able to use FMUs made in other modelling tools that supports the same FMI architecture, such as OpenModelica. Thera are two ways to get the OpenModelica generated FMU to work, either by defining a prefix as described in step 6 or by downloading the latest nightly version 1.22, where this prefix is fixed.

Step 2: Adding the topologies to ConfigurationDesk.

There are three main ways to add an FMU model to the ConfigurationDesk:

1. When creating a new project or application, the model can be added here.
2. Adding a model through the home ribbon add button.
3. Adding a model through the project bar by replacing model topology.

The Device topology can be added through the project bar or by creating a new device under the external device window in the signal chain bar see Figure 5. When the device is created, ports can be added as seen in Figure 5. The ports created must be configured in the property browser, to indicated what type of signal is retrieved or sent from the device.

Note: the device topology is not needed for the building process, but it is nifty to include for getting a better overview of the system.

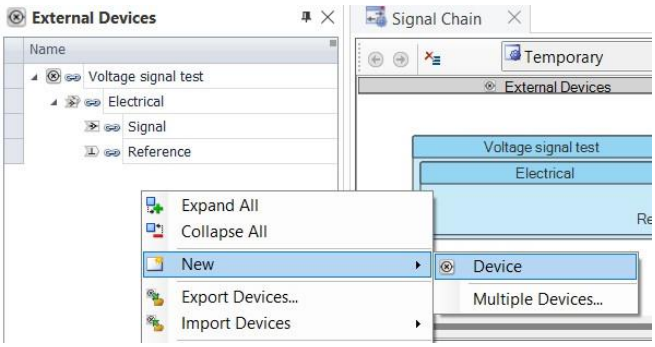

Figure 5: Creating a new device in the signal chain bar.

Hardware topology can be added either under the project bar or when creating a new application in the same way as for the model. If the MicroAutoBox is not connected when

adding the topology, a predefined version of the same MicroAutoBox model can be added see Figure 6.

When using a predefined version of the MicroAutoBox, the application will not be automatically uploaded to a connected unit after the build process of the application is finished. This is due to the ConfigurationDesk not having an exact end location, it only has the hardware specification of that platform. ConfigurationDesk contains predefined hardware for several different models of the MicroAutoBox III and other dSPACE products.

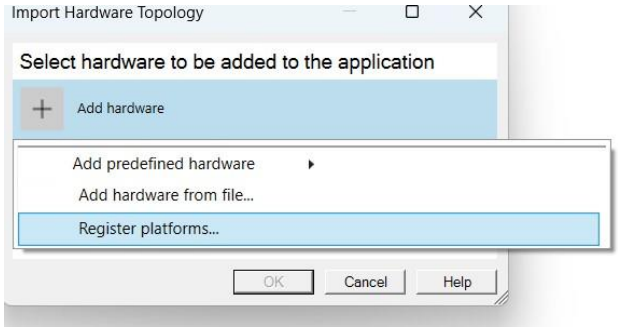

Figure 6: Adding hardware topology.

Tips: in the case of changing out a model with a new one. Then the old model must then be removed before building the application. Figure 7 shows three places where the old model must be removed from. In the case of updating the old model, it can be done by right clicking on the old model and press reload, keeping alle the connections.

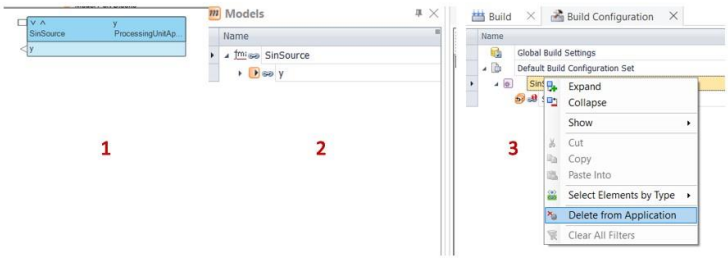

Figure 7: Point 1 and 2 is the in the signal-chain and 3 is under build. Point one can be bypassed by clicking delete completely.

Step 3: Model executable application and task.

As of default the ConfigurationDesk is creating a preconfigured application process as seen in Figure 8. If it is not set as a default, a preconfigured application process must be created either in the task bar or when adding the model as seen Figure 9.

When working with several models there is need for defining priority of the different tasks. This is due to the real-time operating system (RTOS) of the MicroAutoBox can only execute one task at the time. The time step of the task can also be edited in the task window.

Figure 8: Shows the preconfigured application process that can be seen under the task bar.

Note: Long and complicated names can lead to the ConfigurationDesk will not be able to compile. If the FMU has a long and complicated name, and will not compile in ConfigurationDesk, change the task name to something simpler such as test1 etc. In the case that this is not working, the naming must be changed where the FMU is generated.

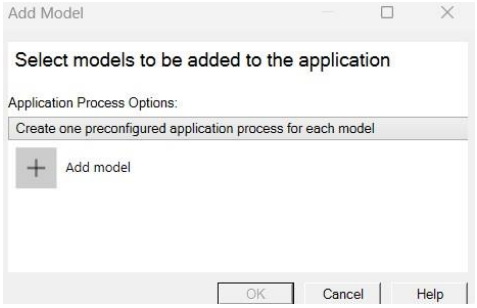

Figure 9: Creating a preconfigured application process when adding a model.

Step 4: Mapping the connections.

When all the topologies are added, they can then be dragged out from the topology browser into the signal chain window and linked together by dragging an output port to an input port as seen in Figure 10. If mapping several FMUs together it is recommended to use the Multiple Models window, as this makes it easier to map them together as seen in Figure 11.

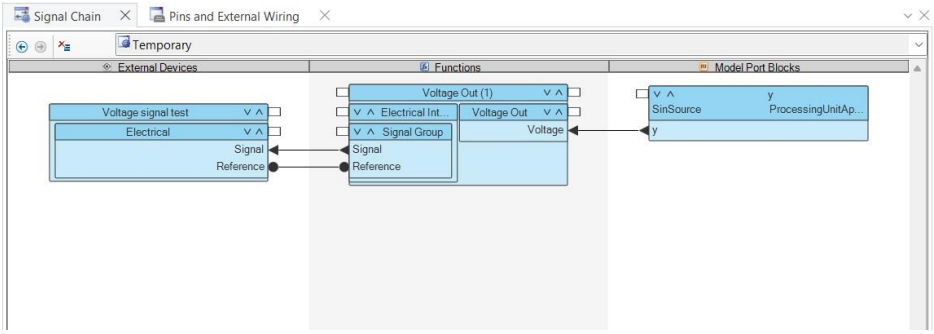

Figure 10: Showing the signal chain for a simple sinusoidal source applied to a device.

Note: A function block can be linked to several model bocks and not vice versa. A model block can be linked to another model block, while the function block cannot be connected to another function block.

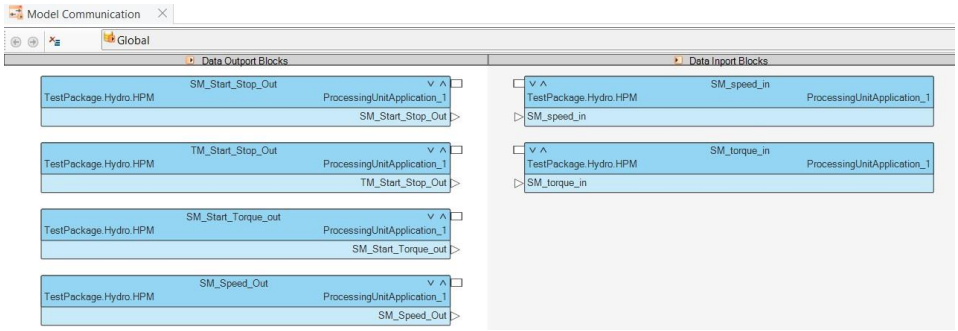

Figure 11: Shows the Multiple Models window.

The MicroAutoBox comes equipped with the possibility of delivering 24V from its power supply to the digital input/output (I/O) signals. To use this function, it must be enabled on the digital function block in the ConfigurationDesk.

The electrical interface of the Multi Bit Out must be changed from Low to High-side switch to enable the 24V DC supply as seen in Figure 12. As a default the polarity of this switch is set to active-high, in the case that the activation should started when sending int signal of 1. The polarity must be changed from active-high to active-low [5].

Note: when for instance using an VFD that has an DCOM circuit, it is only needed to set this connection to High-side switch, as the DCOM is interconnected to the other digital ports.

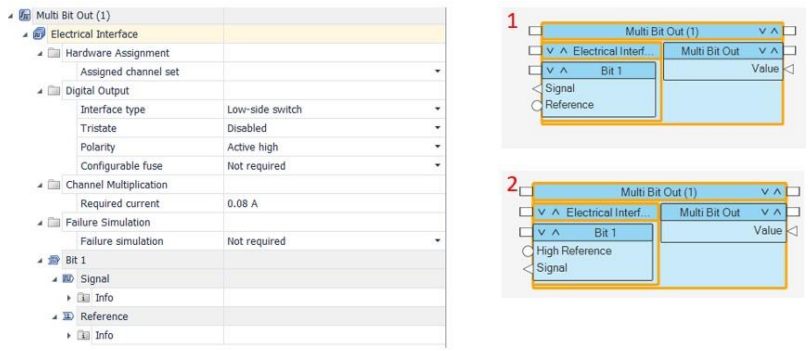

Figure 12: Showing the electrical interface of the Multi Bit Out function block on the left side. On the right side the Multi Bit Out function block can be seen, where 1 shows the Low-side switch is active while 2 shows the High-side switch active.

Step 5: Configure model communication.

This step is only required when using multiple models that communicate with each other.

Information about Blocking and nonblocking can be found in ConfigurationDesk real-time implementation on page 396.

Step 6: Configure building process.

In Figure 13 the build window can be seen. In this window the building process can be configured.

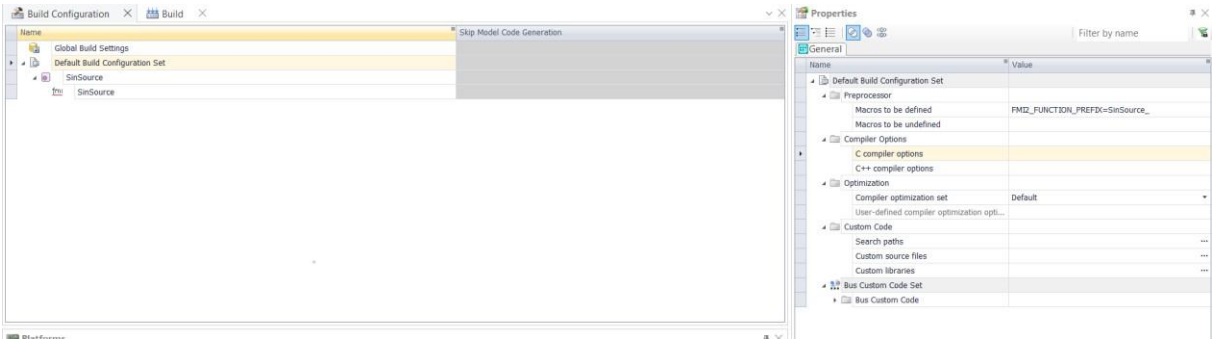

Figure 13: Showing the build window, and where to define macros.

For FMUs generated in OpenModelica standard 1.21 version the build configuration must be edited to make them runnable.

By heading to the *Default Build Configuration Set* → *Preprocessor* → *Macros to be defined* in the Build window seen in Figure 13 and entering the command **FMI2_FUNCTION_PREFIX=XX_** where the XX must be changed out with the model identifier. name.

This name can be found in the model description of the FMU. Usually, it is the same as the file name unless the settings for move FMU and FMU is not set to default. Another thing that can cause a difference from the model name and the model identifier name is when exporting a model from OpenModelica that is inside a package. The model name will then be *PackageName.ModelName*, the model identifier name will then be *PackageName_ModelName*.

Step 7 and 8: Start building process and download application.

The next step is then to is to build the application and the upload the application to the MicroAutoBox. In the ConfigurationDesk the building prosses can be started through the start button on the home ribbon. If the MicroAutoBox is connected to the host pc and turn on when the application is built, it will be automatically uploaded to the MicroAutoBox and initiated as a default setting. If this is not wanted situation it can be changed under the Global Build Settings window seen in Figure 13.

If the MicroAutoBox is not connecter nor turned on when the application is built, the application must then be uploaded manually either through the ConfigurationDesk or by the web interface of the unit. In Figure 14 the load to platform button in Configuration desk will load the build to the MicroAutoBox, if the MicroAutoBox is not detected the load to matching platform button is greyed out.

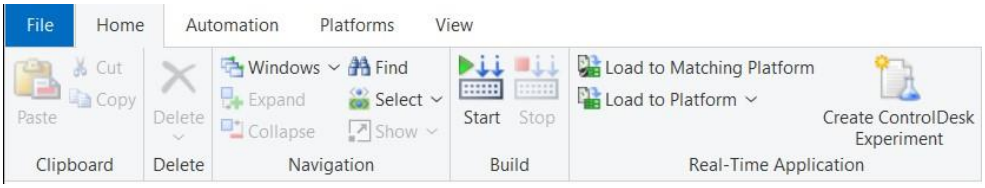

Figure 14: Shows the home page ribbon.

From the both the ConfigurationDesk and the web interface the application can be started, stopped and unloaded. In the ConfigurationDesk this prosses is done in the platforms window on the bottom of the build window page, while for the web interface this can be done in the control window see Figure 15.

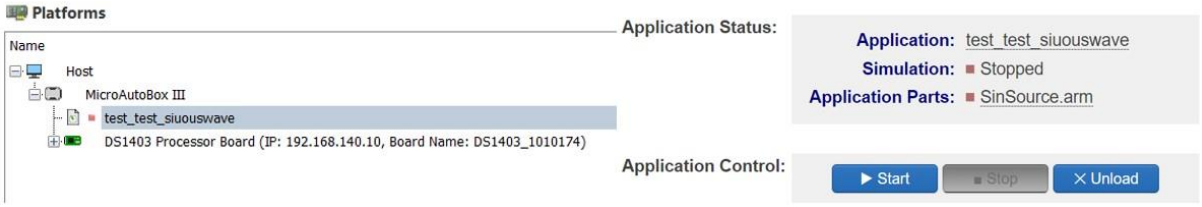

Figure 15: Shows the platform manager to the left and the web interface to the right.

After the building process is finished, it will create the files SDF, RTA, TRC and MAP. The files contain:

- RTA file: is the real-time application containing the executable object file.
- SDF file: is the system description and contains the system components names and parameters.
- TRC file: contains all the variables for the system.
- MAP file: is the file that maps the physical addresses to the symbolic names.

The RTA file is the only one need for running the application on the MicroAutoBox.

Setting up a test environment in ControlDesk for a real time application

This chapter shows how to import an application to ControlDesk, changing parameters with data set, measuring and recording.

As done earlier the MicroAutoBox has been connected to the ConfigurationDesk, as a standard from the dSPACE it will be automatically connected to all the other dSPACE product installed on the host computer. In case the ControlDesk are not showing the MicroAutoBox, repeat the same steps for connecting it in ControlDesk as done in ConfigurationDesk. In Figure 16 an overview of the ControlDesk can be seen [6].

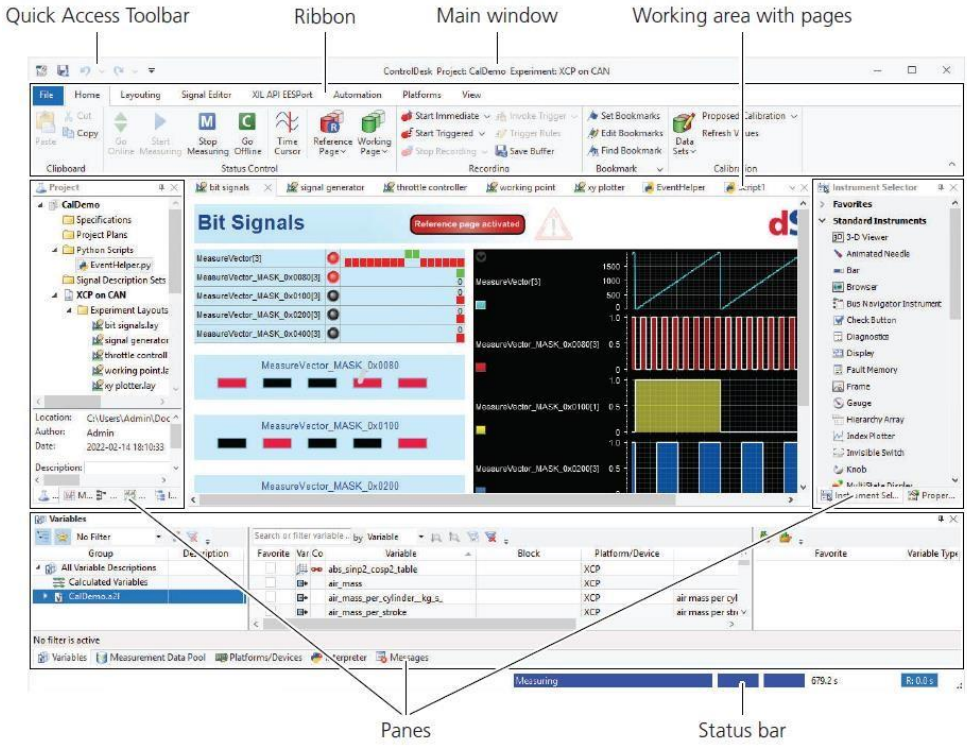

Figure 16: Shows an overview of the ControlDesk.

The ControlDesk uses the SDF file generated by the build process to make it possible to take measurements and changing out the parameters. The SDF file can be found under the file location of the build in ConfigurationDesk. The SDF file can be added when creating a new project of experiment, with the platform that it is supposed to be loaded on to.

Note: when the button go online on the home ribbon is pressed. The application cannot be unloaded, unless the go offline button is pressed. Going offline does not mean that the application is stopped.

Creating new parameters:

A simple way to generate new values for the parameters, is to create a new layout, and dragging the parameters out on to a layout, changing the values and generating and creating a new data set.

A layout will be automatically generated when creating a new experiment, they can also be created by right clicking on the experiment layouts under the task bar.

The parameters of the model can be found under the model root, in the variable tab as seen in Figure 17. The parameters are marked with a P. When dragging the parameter out on to a layout, there are several options to choose from as seen in Figure 18. The numerical input is one of the easier to use [7].

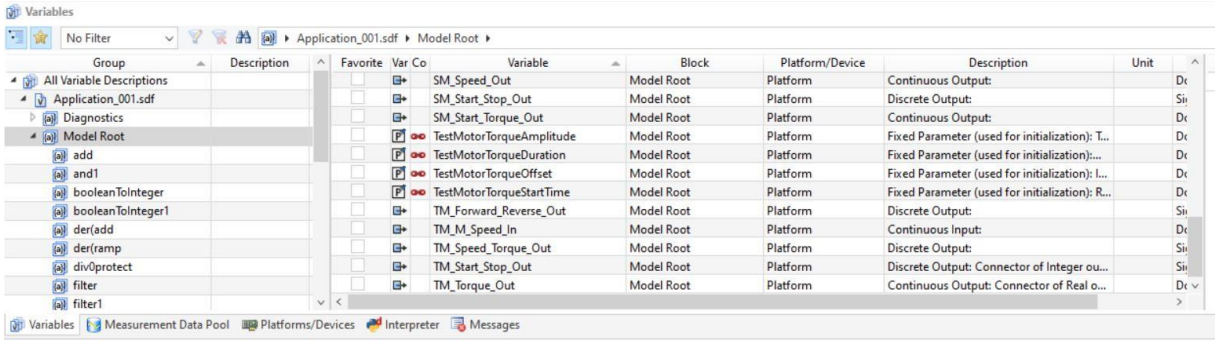

Figure 17: Showing variable tab, and where to fine the parameters of the application.

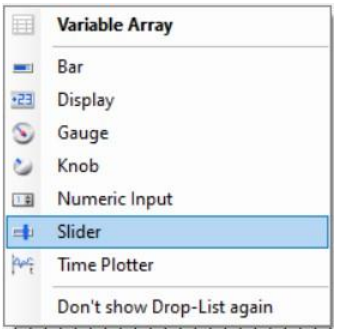

Figure 18: Shows the options for the parameter dragged out on to the layout.

The numerical input has a writable area, which will not store the values written, only working as display. To change the values, it can be done either by using the up and down buttons connected to the display or by right clicking on the display, which is giving an option for adjusting the values.

By going to the hardware configuration under the project tab, the application SDF file can be found. Right clicking on the SDF file, the option of creating a data set from the open layouts appear. The generated data set can be seen in Figure 20, where activating the download on online calibration starts must be activated for uploading new values, an arrow pointing downwards will emerge on the data set icon. When starting the application from the ControlDesk the old values will be sent to the RTC first, by stop and restarting the application loaded on the RTC, the new parameters will be activated.

Note: only one data set can be set to upload values on at the time. To deactivate a data se,t it can be done by pressing the close button as seen in Figure 20.

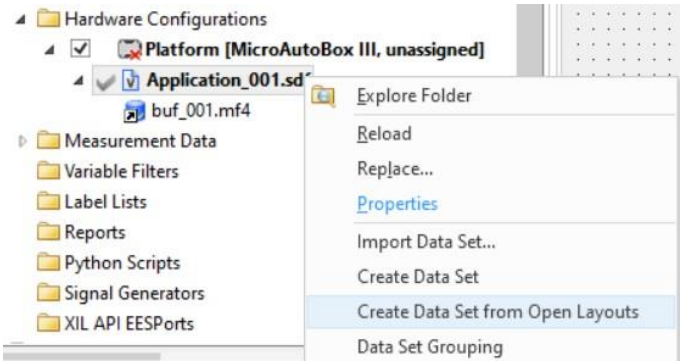

Figure 19: Shows how to create a data set from an open layout.

Note: as seen in Figure 20, the icon of the data set is half green. This is because it is not a complete data set, only storing the parameters that has been changed.

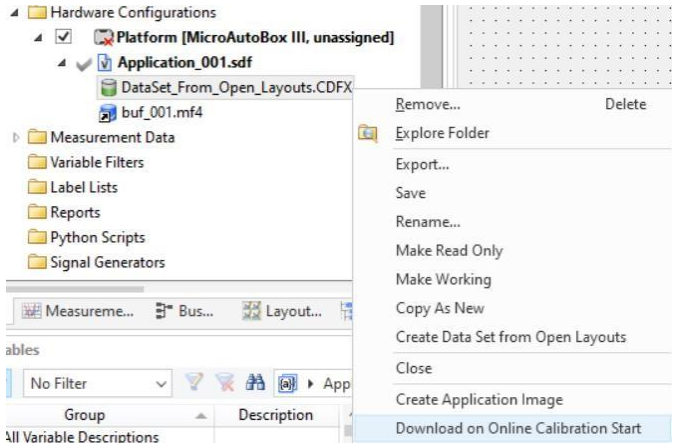

Figure 20: Showing the created data set and how to upload these changes to the RTC.

Note: there are several other ways to work with data sets described in the PDF ControlDesk Calibration and Data Set Management [8].

Adding measurements and recordings:

By going the Measurement configuration tab as seen in Figure 21, measurements can be added under the recorder. As of default Recorder 1 is added under the creation of the project. Measurement can be added by either dragging a time plotter into an open layout as seen in Figure 18, or by dragging it into the recorder [9].

Note: the red chain is indicating that there is a time plot attached to it, while the yellow indicates that it is activated by the ControlDesk, in a layout or in the recorder for measurements.

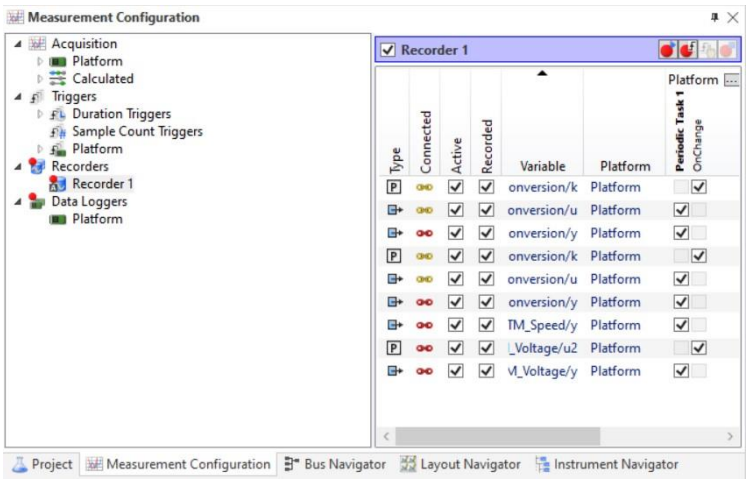

Figure 21: Shows the measurement configuration tab.

When running the application, the ControlDesk is not measuring unless it is specified. This can either be done with adding a trigger to the measurement or by pressing the start

measuring button on the home ribbon. Pressing the Start measuring button will start the application if it is not activated.

Note: information about triggers can be found in ControlDesk Instrument Handling [7].

Note: in the case the user wants to store the measurements and has not recorded them, the save buffer found on the home ribbon. This generates a buffer file that is stored under the measurement data in the project window. The buffer memory is only storing the lates 60 seconds of the measured data.

When it comes to the plotting of the measurements, the graph can be adjusted by right clicking on the graph and opening the Axis/signal properties, as seen in Figure 22. This also gives the opportunity of changing label names, background colour, etc.

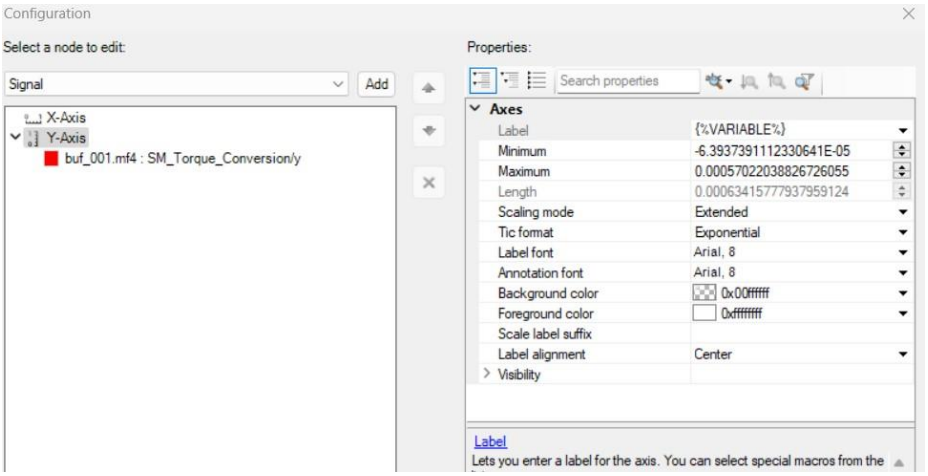

Figure 22: Shows the Axis/signal property window, for the graph.

In the measurement and configuration window, triggers can be added to record and star/ stop measurements. Recordings can also be started by pressing the start immediate button on the home ribbon. The recorder is only recording the measurements activated in the recorder as seen in Figure 21.

Note: go online can also be activated with either pressing the start measuring or start immediate recording button.

References

[1] dSPACE, ‘User Guide’, *MicroAutoBoxIIGettingStarted.pdf*, no. b, 2022.
[2] dSPACE, ‘User Guide’, *MicroAutoBoxIIIHardwareInstallationandConfiguration.pdf*, 2020.
[3] dSPACE, ‘User Guide’, *ConfigurationDeskIntroductionandOverview.pdf*, no. b, 2022.
[4] dSPACE, ‘User Guide’, *ConfigurationDeskRealTimeImplementationGuide.pdf*, no. b, 2022.
[5] dSPACE, ‘User Guide’, *ConfigurationDeskFunctionBlockProperties.pdf*, no. b, 2022.
[6] dSPACE, ‘User Guide’, *ControlDeskIntroductionandOverview.pdf*, no. b, 2022.
[7] dSPACE, ‘User Guide’, *ControlDeskInstrumentHandling.pdf*, no. b, 2022.
[8] dSPACE, ‘User Guide’, *ControlDeskCalibrationandDataSetManagement.pdf*, no. b, 2022.
[9] dSPACE, ‘User Guide’, *ControlDeskMeasurementandRecording.pdf*, no. b, 2022.

Appendix C: FB diagram.

24.04.2023 Stellantrieb - Drehzahl