Elina Bisi

Development and Demonstration of an Electric Powertrain Integration Testbench Including PC-based Control System

Metropolia University of Applied Sciences

Bachelor of Engineering

Automotive Engineering Degree Programme

Bachelor's Thesis

18.11.2024

Abstract

Author(s):	Elina Bisi
Title:	Development and Demonstration of an Electric Powertrain Integration Testbench Including PC-based Control System
Number of Pages:	29 pages + 0 appendices
Date:	18.11.2024
Degree:	Bachelor of Engineering
Degree Programme:	Automotive Engineering Degree Programme
Specialisation option:	Automotive Electronics Engineering
Instructors:	Juuso Kelkka, senior manager, technology development, IONCOR Oy Pasi Kovanen, senior lecturer

---

This thesis covers an electric powertrain testbench development with PC-based control system for battery system testing commissioned by IONCOR Oy. This thesis report focuses on the PC-based control system, which was done using Vector CANoe on CAN bus. The objective of this project is to produce a working testbench with a control system ready for battery system testing.

First, the batteries, inverters and motors used in the testbench are introduced. After that, CAN bus, CANopen protocol, and CANoe are explained before presenting the development process of the control system. Lastly, the testbench development and testing during this project is demonstrated.

The system parts were tested independently constantly throughout the project before the whole system was assembled. Interference and compatibility problems were found during the project, which were a good to find and solve. The functional safety diagnostics in the battery pack sensed the noise level induced by the inverters when motors were idling and intermittently triggered a spurious fault condition. The testbench will be used for battery testing with present and future IONCOR battery products.

The project in whole was a success and produced a working testbench with a control system ready for testing.

Keywords: batteries, testing, control systems

Tiivistelmä

Tekijä:	Elina Bisi
Otsikko:	Sähköisen voimansiirron testipenkki-integraation kehitys ja demonstraatio sisältäen tietokonepohjaisen ohjausjärjestelmän
Sivumäärä:	29 sivua + 0 liitettä
Aika:	18.11.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Ajoneuvotekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Autosähkötekniikka
Ohjaajat:	Senior manager, technology development Juuso Kelkka, IONCOR Oy Lehtori Pasi Kovanen

---

Tämä insinöörityö käsittelee sähköisen voimansiirron testipenkin ja sen tietokonepohjaisen ohjausjärjestelmän kehitystyötä. Työn tilaaja on IONCOR Oy. Tämä raportti keskittyy tietokonepohjaisen ohjausjärjestelmän kehitykseen. Työn tavoitteena oli tuottaa toimiva testipenkkijärjestelmä akkutestaukseen.

Projekti alkoi tutustumisella alkuperäiseen testipenkkiin ja sen ohjausjärjestelmään, jonka pohjalta aloitettiin uuden ohjausjärjestelmän luonti. Alkuperäiseen testipenkkiin tehtiin tarvittavat muutokset ja uusien komponenttien integrointi. Ohjausjärjestelmä luotiin Vector CANoe -tietokoneohjelmistolla CAN-väylällä käyttäen CANopen-protokollaa.

Järjestelmän komponentteja testattiin koko projektin ajan ennen järjestelmän lopullista kokoamista. Häiriöitä ja yhteensopivuusongelmia todettiin jo projektin aikana ja löydöt olivat hyödyllisiä. Akkupaketin toiminnallisen turvallisuuden diagnostiikka havaitsi häiriön, kun moottori oli päällä mutta ei pyörinyt, mikä laukaisi harhaanjohtavan virheilmoituksen.

Kokonaisuutena projekti onnistui ja tuotti toimivan akkutestaukseen tarkoitetun sähköisen voimansiirron testipenkin. Testipenkkiä tullaan käyttämään IONCOR:n akkutuotteiden testaukseen.

Avainsanat: akut, testaus, ohjausjärjestelmät

Contents

Abbreviations

AC	Alternating current
BMS	Battery management system
CAN	Controller area network
CAPL	Communications access programming language
DC	Direct current
ECU	Electronic control unit
HV	High voltage
LV	Low voltage
NMT	Network management, a CANopen protocol
PDO	Process data object, a CANopen protocol
PDU	Power distribution unit
RPM	Revolutions per minute
SDO	Service data object, a CANopen protocols
SOC	State of charge
VAC	Volts alternating current
VDC	Volts direct current
Vector	Vector Informatik GmbH

1 Introduction

The objective for this thesis is to develop and demonstrate an electric powertrain testbench for battery testing. The main objective was to create a PC-based control system for the testbench. This thesis will examine in detail the making of the PC-based control system using CAN bus and Vector CANoe and generally cover the components and changes made to the testbench.

The purpose for this testbench is to simulate actual powertrain of a vehicle in battery pack testing. Through this type of testing, different faults can be found that other methods would not be able to find.

This thesis was commissioned by IONCOR Oy (known before as Valmet Automotive EV Power Oy). Established in 2018, IONCOR is the subsidiary of Valmet Automotive Oy that was established in 1968. IONCOR is a Tier 1 -class battery system supplier and contract manufacturer based in Salo and Uusikaupunki, Finland, and Kirchardt, Germany.

2 Components

There are two battery packs of IONCOR's Modular Power Pack modules, two inverters and two electric motors used in this project. The motors will be connected to each other using an axle when the other motor can be used as a load and generator. As a generator the motor will charge the battery it is connected to. The whole bench configuration is found in chapter 5.

2.1 Battery pack

IONCOR's Modular Power Pack (Figure 1) in high voltage application is designed for easy integration or as a so-called "plug-and-play" system. This means that the customer can design their product around the Power Pack system and choose the number of modules. The Power Pack uses LTO or lithium titanite oxide

chemistry which is known to be safe, and to have good performance and life span (BU-205: Types of Lithium-ion 2023). The maximum number of modules for one PDU or power distribution unit is 15. The modules are in series. The nominal voltage is 48 V, capacity 2.2 kWh and energy 46 Ah of one module (Modular Power Pack, High Voltage 2022).

Figure 1. Modular Power Pack (right) and PDU (left). (Modular Battery Platform 2023.)

The two battery packs in this project consist in total of 15 Modular Power Pack modules divided to packs of 7 and 8 which means nominal voltage of 336 V and 384 V. Both packs have their own PDU with a BMS and are connected to their respective inverters.

2.2 Inverters

Inverter converts direct current or DC to alternating current or AC or the other way around. The battery packs supply DC power and are connected to the inverter and the inverter is connected to the electric motor which requires AC power. The inverter can change the motor speed by changing the frequency or torque by adjusting the amplitude of the AC power (How electric vehicles work 2024). Inverters are very common in automotive motoring applications.

The first inverter used is HES880-104-0602A-5 from ABB Oy. HES880 has rated input voltage of 300 to 750 VDC and output 0 to 500 VAC (Traction Inverters n.d.). HES880 is designed for example for off-highway applications and other environments that require high shock and vibration durability (Traction Inverters n.d.). HES880 can be controlled using CAN bus or its original handheld controller. The second inverter is EC-C1200-450-L from Danfoss Oy. For EC-C1200 rated nominal voltage is 750 VDC and 500 DAC and current limit is rated to 300 A. EC-C inverters are designed to stand high number of load cycles and to be used in heavy duty machines or buses for example (High voltage electric inverters n.d.).

2.3 AC motors

AC motor is a type of an electric motor that converts electric energy into mechanical motion. AC motors have a wide variety of uses and sizes and they are very common. Stationary stator and rotating rotor are the main parts of an AC motor. AC motor technology is based on two magnetic fields interacting and thus making the rotating motion. Two general types of AC motors are synchronous and induction motor that have different subtypes. (AC Motor 2024.)

The first motor, connected to the HES880 inverter, is M3LK 160ML 4 B5 permanent magnet synchronous motor or PMSM rated at 140 kW. The second motor, connected to the EC-C1200 inverter, is EM-PMI375-T1100-2900 synchronous reluctance (assisted) permanent magnet motor or SRPM rated at 296 kW. The EM-PMI375-T1100 motor is designed for heavy machinery, bus, and marine vessel applications (High voltage multipurpose motors n.d.).

3 CAN bus and Vector CANoe

3.1 CAN

CAN bus is the bus type used in this project and its function and protocols are explained in this chapter. Understanding the function of CAN bus is important so one can understand how the control system for the testbench was developed.

CAN bus system was the first bus system in mass production motor vehicle in 1991 and has since been established as a standard in automotive and automation industries. CAN bus provides great benefits, for example, in diagnostics. Control units on the bus receive the needed information for fault diagnostics immediately. (Automotive Electrics and Automotive Electronics 2007: 92.)

CAN bus conveys messages by changing the voltage levels. CAN bus can be high speed (from 125kbit/s to 1 Mbit/s, CAN-C) or low speed (from 5 to 125 kbit/s, CAN-B). High speed CAN bus is used when faster communication is required, for example safety critical systems such as drivetrain and chassis controls. Similarly, when slower communication is required, slow speed bus is used, for example, air-conditioning, mirror adjuster and navigation control systems. One bus can have up to 30 electronic control units. (Automotive Electrics and Automotive Electronics 2007: 92.)

Different buses can be connected to one or more gateway modules. There can be different kinds of buses on one gateway module because the gateway is a translator between them. Gateway processes the information from one bus and, if needed, sends it to another. For example, the diagnostic bus depends on the gateway to forward it the information needed. (Automotive Electrics and Automotive Electronics 2007: 88-92.)

Networks nodes (Figure 2) are the part in electronic control units that process and send information from and to the bus. The node consists of CAN transceiver, CAN controller and a microcontroller. The CAN controller generates a bit stream from the binary data and forwards it to the transceiver on “TxD” line seen in the Figure 2. The CAN transceiver amplifies signals, generates voltage, and transmits it to the bus on high and low lines. Messages received from the bus are first processed by the transceiver and then sent to the controller on “RxD” line. The microcontroller runs an application program and analyses the received data and prepares outbound data. (Automotive Electrics and Automotive Electronics 2007: 93.)

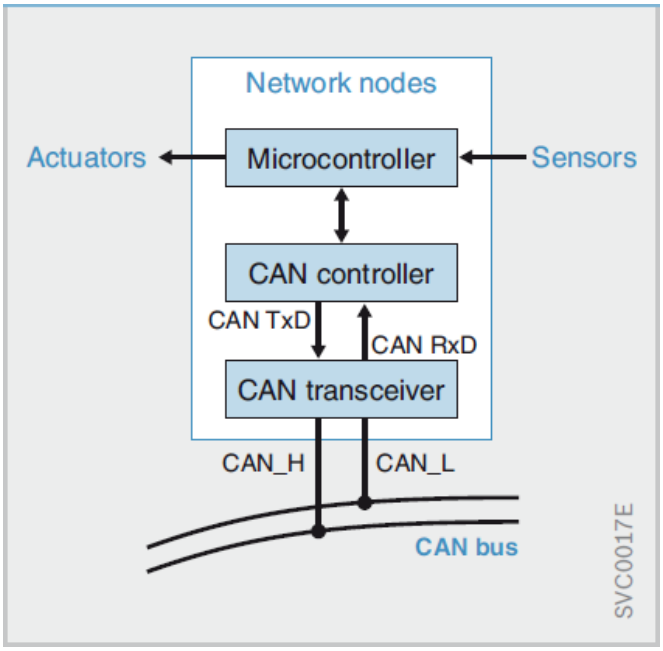

Figure 2. Network nodes in the CAN. (Automotive Electrics and Automotive Electronics 2007: 93.)

Most commonly CAN bus works on a twisted two-wire line. Untwisted pair might also be used, and single-wire lines have been used to reduce manufacturing costs. On the two-wire line, the data is sent on both lines but on different voltage levels to filter out interference. Figure 3 visualizes the voltage levels for both high and low speed buses where the dominant state represents “0” in binary and the recessive “1” in binary. (Automotive Electrics and Automotive Electronics 2007: 94-95.)

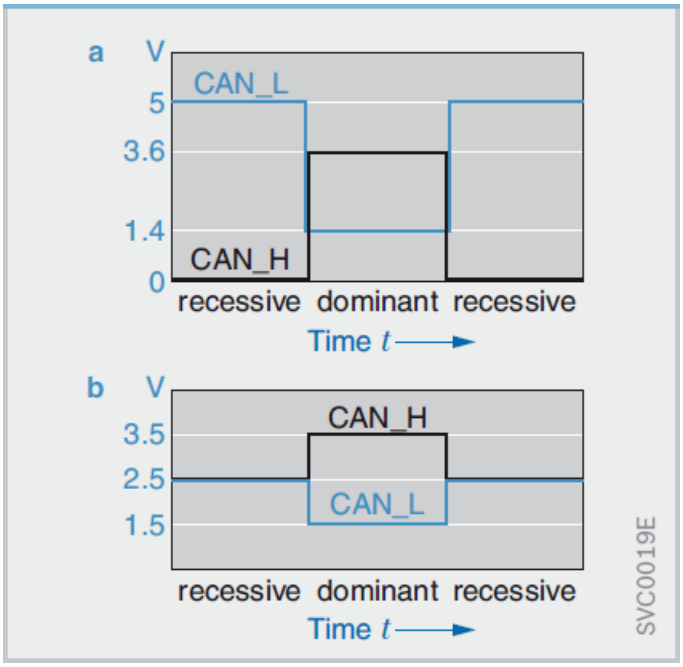

Figure 3. Voltage level of low-speed (a) and low-speed (b) CAN. (Automotive Electrics and Automotive Electronics 2007: 95.)

The benefit of using two-wire lines is filtering out the interference. The two lines are usually named CAN high (CAN_H) and CAN low (CAN_L), because of the difference in the voltage levels. The receiving node subtracts the low-level voltage from the high lever voltage and because the voltage levels are symmetrical, when interference occurs, it can be filtered out as it affects both lines equally. (Automotive Electrics and Automotive Electronics 2007: 94-95.)

3.1.1 Frame format

The data on CAN bus moves in messages. The information is arranged in a predetermined sequence that has different parts in the frame assigned in the standard. CAN bus supports two message formats, base format or CAN 2.0 A, and extended format or CAN 2.0 B. The most important difference between these is the length of the identifier, which is 11 bits for base format and 29 bits divided into 11 and 18 bits for extended format. One bit has the binary value of “1” or “0”. (Automotive Electrics and Automotive Electronics 2007: 98.)

In this thesis, the CAN base frame format will be explained. The CAN bus frame format is divided into sections that have different functions as can be seen in Figure 4. The frame starts with a dominant bit called start-of-frame bit. This is followed by arbitration field which includes the 11-bit identifier and remote transmission request bit or RTR-bit totalling 12 bits. The RTR-bit tells if the frame contains data or is requesting data. (Automotive Electrics and Automotive Electronics 2007: 98.)

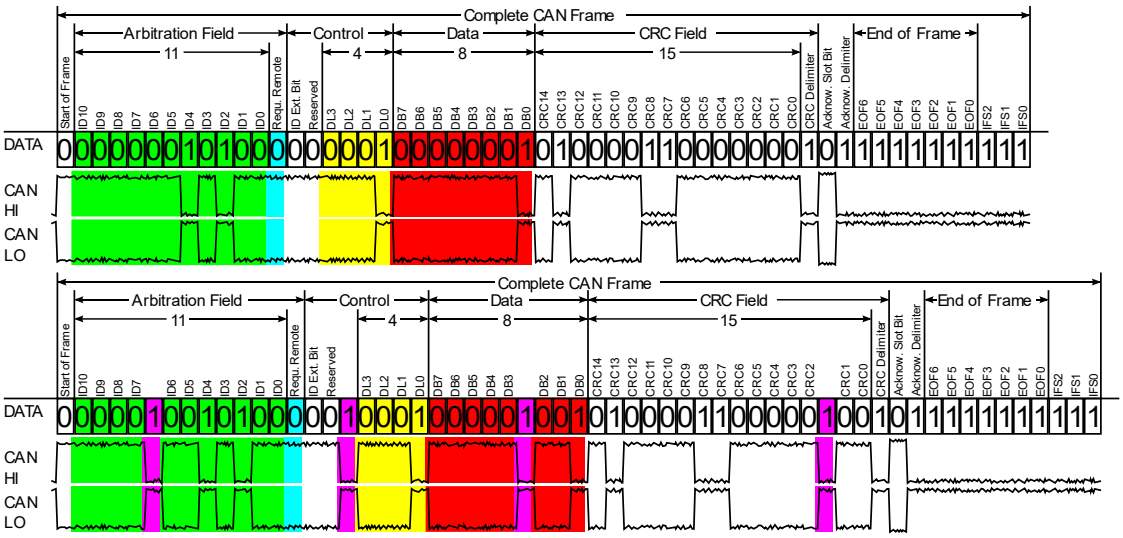

Figure 4. CAN bus base frame format, above without stuff bits and below with. (File:CAN-Frame mit Pegeln mit Stuffbits.svg 2014.)

The next field after the arbitration is control. First bit out of six bits in the control field is a dominant identifier extension or IDE bit, and after that is a recessively sent bit reserved for future extensions. The last four bits of the control field tell the length of the following data field. The data field, which contains the actual information that the message is conveying, can be up to 64 bits or 8 bytes long. (Automotive Electrics and Automotive Electronics 2007: 99.)

The cyclic redundancy checksum or CRC-field follows the data field. The CRC-field is one method to check that the message was conveyed successfully and if errors occurred. The CRC-field is a calculated 15-bit check sequence from start-of-frame, arbitration, control, and data fields by the sending node. The receiving node will do a calculation based on the frame to see if it matches to the CRC-

field. The CRC delimiter, which is the 16th bit, is recessive and ends the CRC field. (Automotive Electrics and Automotive Electronics 2007: 99.)

Before end-of-frame are two acknowledgment bits. The first bit, ACK slot bit, is sent as recessive by the transmitting node and overwritten by the receiver as dominant. The second bit of acknowledgment is for the sender to verify if the message was transferred correctly. If there is no second bit in the ACK field, then an error has been detected. (Automotive Electrics and Automotive Electronics 2007: 99-100.)

In the second frame example of Figure 4, stuff bits marked with purple can be seen. The stuff bits are a form of fault checking. Through start-of-frame to end of CRC-field after every five consecutive bits is a bit of opposite value. The receiving node clears the stuffed bits after receiving. (Automotive Electrics and Automotive Electronics 2007: 101.)

3.1.2 CANopen

In this project CANopen is used as the communications protocol, because all the devices used in the project support it. CANopen is a high-level communications protocol and device profile specification that bases on CAN. CANopen is standardized and widely used, for example, in off-highway machinery, railway applications and automation (CANopen CC – The standardized embedded network n.d.).

The device that uses CANopen has three logical parts. CANopen protocol handles the communication using CAN bus, application software handles internal controls and object dictionary interfaces the two. For example, a motor speed is collected from a sensor, stored in a parameter in object dictionary and then send on bus if mapped to a message. CANopen protocols needed in the project are SDO (service data object), PDO (process data object), NMT (network management) and heartbeat protocol. (CANopen CC – The standardized embedded network n.d.).

The object dictionary (Figure 5) is a standardized listing of communication and application parameters. This means that the object dictionary includes all data that is needed for communication. Every data has a 16-bit index and an 8-bit sub-index with what the data can be found. Different dictionary index ranges have different purposes. (CANopen internal device architecture n.d.).

Index	Description
0000h	reserved
0001h - 025Fh	Data types
0260h - 0FFFh	reserved
1000h - 1FFFh	Communication object area
2000h - 5FFFh	Manufacturer specific area
6000h - 9FFFh	Device profile specific area
A000h - BFFFh	Interface profile specific area
C000h -FFFFh	reserved

Figure 5. Object dictionary indexes. (CANopen internal device architecture n.d.).

SDO protocol or service data objects enable access to all entries of the object dictionary. Communication between two CANopen devices can be established on CAN bus with an SDO. The owner of the object dictionary, which is being accessed, is the SDO server and the device accessing the object dictionary on the other device is the SDO client. (Service data object (SDO) n.d.).

PDO protocol or process data objects are used for high-priority control and status information. PDO message consist of one CAN frame. One bus can have multiple PDO messages, because one message can only contain 8 bytes or 64 bits of pure data. PDO messages have their own communication and mapping parameter sets in object dictionary, using which the messages can be brought to use. The communication parameters of a PDO indicate the identifier of the frame and the triggering of the message, for example, a periodic timer. The mapping parameters tell what data is being transmitted and where it is to be stored. (Process data object (PDO) n.d.).

3.2 Vector CANoe

The control system in this project functions on multiple CAN buses, which control the components with CANoe-software and its hardware. The control system is needed to configure and power on the components, drive the bench, and analyse in case of faults or errors. Clear and easy-to-use control system is also important for the safety of the testing process.

CANoe Family from Vector Informatic GmbH is a PC software developed for automotive and related industries. CANoe is a software for development, analysis and testing from software components and subsystems up to ECUs and entire networks. CANoe can be used on real hardware, prototypes, or real products, and in virtual environments or machines. Application areas are analysis, diagnostics, simulation, stimulation, and testing. (CANoe Family n.d.).

For the CANoe software to send messages to buses, a network interface is needed. In this project VN8911 base unit is used (Figure 6). VN8911 is part of the Vector's VN8900 modular network interface family (VN8900 n.d.). VN8911 has possibility for eight channels, but only six are used during this project. VN8911 receives the messages on a bus and translates it to the CANoe on a computer and translates and transmits the messages to the bus sent from the computer.

Figure 6. Vector VN8911 base unit. (VN8900 n.d.).

CANoe has multiple different configuration templates, such as CANopen in Figure 7, from which user can choose from. The CANopen template has simulation and measurement setup ready from where user can continue creating the system. Simulation setup is important because there the networks are created for each bus. The network is what contains all information required for communication on its bus. To receive data on a bus, the network in CANoe is only required to be connected to the correct channel with correct bus speed and dbc-file.

A network is set to the selected bus type and the nodes are filtered out to work on that bus. Under network nodes, interactive generators, replay blocks and databases can be assigned. There are different node options that have different functions. A different node might be needed to use some specified functions in CAPL. With interactive generator, no code is needed. It creates a window from which messages can be sent to bus. Logged data can be replayed using a replay block.

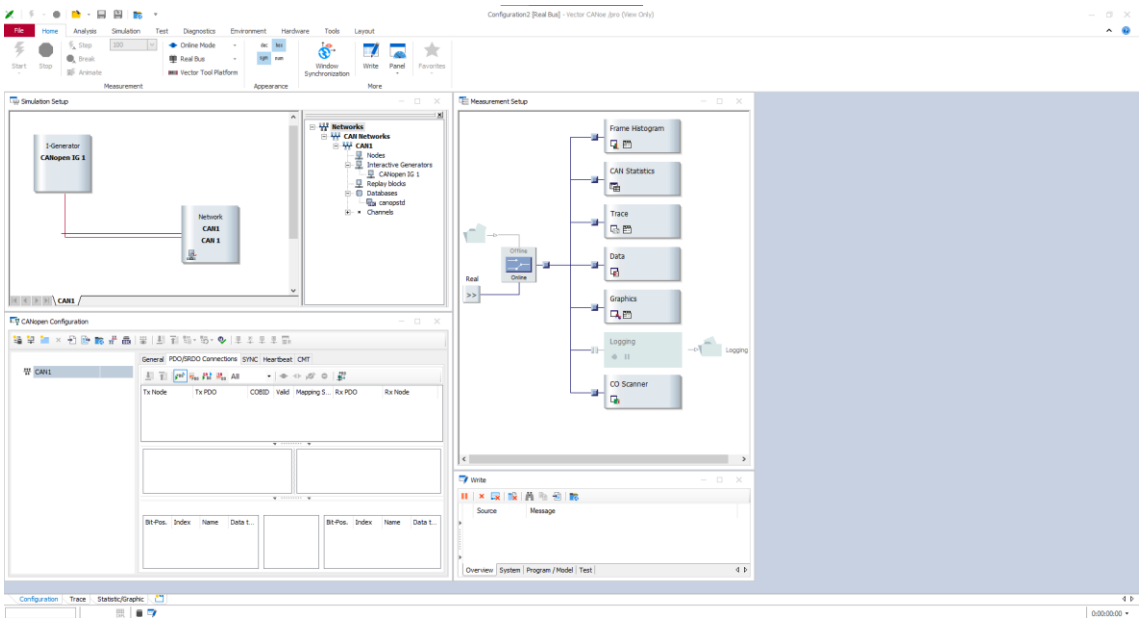

Figure 7. CANoe CANopen template.

For the message transmission and receiving to be successful, the bus speed and network need to be correctly set in “Network Hardware” and “Channel Mapping”. In “Channel Mapping”, the network created before is set to the correct bus channel that it is connected physically on the network hardware which is the VN8911 in this project. In “Network Hardware” the bus speed can be set and scanned if it is not known.

Database or dbc-file is the part that CANoe uses to translate the message data into more understandable terms, because by default the data in the frame is interpreted from binary to hexadecimal form. For example, if a message contains the rotation speed of the motor, that data is written as a signal into the dbc file and with that CANoe knows what and how many signals each message contains. The signals need to be in the correct order in the dbc file so that correct data is translated correctly. Signals and messages can be named any way the user wants. One message or frame can have multiple signals, depending on the signal length. The dbc-files are created and edited in CANdb++ editor (Figure 8).

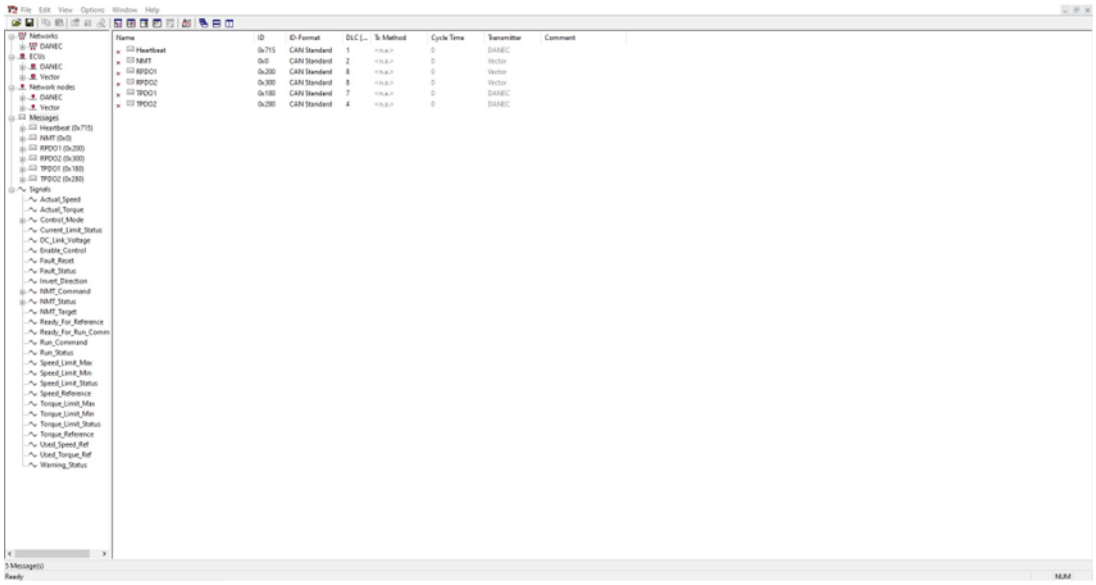

Figure 8. EC-C1200 dbc-file opened in CANdb++ editor.

One important part of this project was the creation of the control panels. Panel creation in CANoe is simple using the panel designer (Figure 9). Panels are important part of the clarity of the control system. Panels can have different gauges, sliders, and buttons, for example. In addition, pictures and objects can be brought to the panel designer.

The different objects on the panel relate with system variables to the CAPL code, which gives the objects function, for example to start or stop something. Gauges and other visual meters can take their data directly from the signal. For example, a speed gauge can take the data directly from a motor speed signal that is being sent from the inverter in this case.

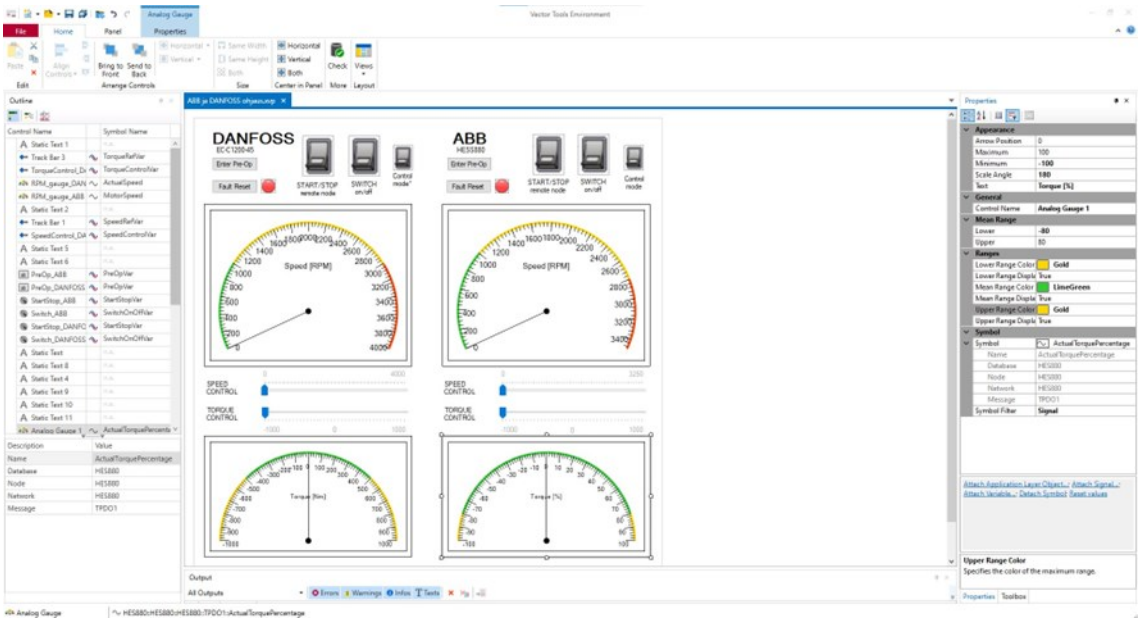

Figure 9. A control panel of this project open in Panel Designer.

CAPL is an event-driven programming language developed by Vector Informatik. CAPL is similar to C programming language. In CANoe, CAPL programs are written and compiled in a dedicated browser, CAPL Browser (Figure 10). Typical tasks for CAPL programs might be reacting to received messages or signal changes and sending messages. (CAPL Used With CANoe And CANalyzer n.d.)

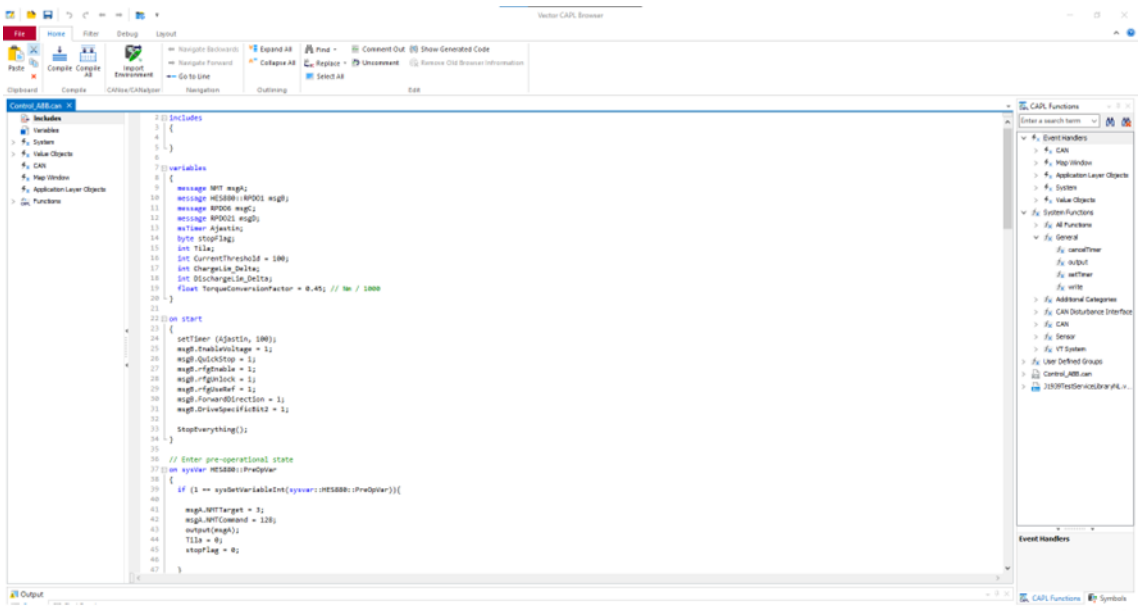

Figure 10. A code from this project open in CAPL Browser.

In addition, the aforementioned system variables are important if panels are to be connected to a code. Every button, switch, text box and slider need a system variable. A system variable stores the current state of the object it is connected to and stores it up to the time the value changes. For example, a two-state switch with values of “1” for on and “0” for off.

CAPL code is saved as a file on the computer and the file needs to be connected to a node block in configuration. One code file needs at least one network to be connected for it to work, but it can be connected to multiple networks. The code needs to be connected to all the networks it needs data from. For example, if the code needs to monitor signals from network 1 and send data to network 2, it needs to be connected to both.

4 PC-based control system

The creation of the control system in CANoe started with an analysis of the requirements. The main control panel, that would be used for driving the motors, needed to be simple but also have as much information as possible. The two battery packs would also need a panel each to turn the packs on. The final requirement was a drive loop panel, so that the loop would run automatically with the set limits.

The whole control system is based on the battery packs and the control system of the original testbench, which had the databases and networks for the battery and HES880. The battery pack database and network had to be duplicated and named differently, because the original system required only one battery. Starting with the original control system meant that the battery controls and diagnostic tools were already present. In this projects case there is six buses and networks which are HES880, EC-C1200 and two for both battery packs.

4.1 Configuration

The information needed to create the databases and to configure the inverters, was in the inverters' communications manuals and PC software. The communication through CAN bus needed to be configured. HES880 was configured using SDO that had an existing panel and code in CANoe. HES880 database needed to be edited. EC-C1200 was configured using its PC software. The database was created according to the configuration of the EC-C1200.

EC-C1200 communication configuration was done in Danfoss' PC software PowerUSER. The message configuration was easy to do in the software. The message was turned on and signals were chosen from a listing, which was written to the device, EC-C1200. Creating the dbc-file at the same time was convenient because the PowerUSER presented all information needed at the same time.

Even though EC-C1200 configuration was easy and quickly done using the software, a problem with the control mode change was found. The inverter refused to change the control mode using bus commands. This was solved after a parameter was found that determined the source for the control mode, which had to be set to "255" meaning network.

EC-C1200 transmits heartbeat, PDO1, and PDO2 and receives NMT, PDO1, and PDO2. The heartbeat message only contains NMT status signal, which tells the status of the device, for example pre-operational, running or stopped. For example, PDO1 received by EC-C1200 includes control mode, speed and torque reference, control enable, run command, fault reset and invert direction signals (Figure 11).

Name	Message	Multiplexing/...	Startbit	Leng...	Byte Order	Value Type	Initial Value	Factor	Offset	Mini...	Maxi...	Unit	Value Table
Control_Mode	RPDO1	-	0	16	Intel	Unsigned	0	1	0	0	0		VtSig_Control_...
Speed_Reference	RPDO1	-	16	16	Intel	Signed	0	1	0	0	0	RPM	<none>
Torque_Reference	RPDO1	-	32	16	Intel	Signed	0	1	0	0	0	Nm	<none>
Enable_Control	RPDO1	-	48	1	Intel	Unsigned	0	1	0	0	0		<none>
Run_Command	RPDO1	-	49	1	Intel	Unsigned	0	1	0	0	0		<none>
Fault_Reset	RPDO1	-	50	1	Intel	Unsigned	0	1	0	0	0		<none>
Invert_Direction	RPDO1	-	51	1	Intel	Unsigned	0	1	0	0	0		<none>

Figure 11. EC-C1200 receive PDO1 signals as seen in CANdb++.

In the communication documents for HES880 a good example of configuration through SDO was found. Some of the configuration was already done in the previous project HES880 was part of but the mapped parameters had not been saved so the configuration had to be done again. HES880 transmits PDO1 and receives NMT, PDO1, PDO6, and PDO21 from the PC.

In HES880 the parameters for receive PDO1 message are in index 1400. The index 1400 has 5 subindexes. The message values, such as transmission type and communication objects identifier, and enabling and disabling the message are determined in the 1400 index.

The data of the receive PDO1 message is determined in the index 1600, which has 5 subindexes. One of these subindexes is the number of mapped objects and the rest are the possible data objects. The wanted data objects are searched from the library in the objects dictionary in the document and written to the subindexes of the index 1600. The length of the data must be considered, because if the message has two objects the length of 32 bits mapped, then the message is full.

The message mapping for HES880 took more time than expected because, even though the example in the documentation was useful, in this project more signals were needed and finding the correct ones from the parameters and object dictionary in the documents turned out to be difficult. The main issues were the torque mode and limit changes from bus.

Using the parameter document, found in the ABB’s Drive Composer PC-software, the control mode parameter was found, and it was mapped to a message. This parameter changes the control mode between two pre-set modes which are speed and torque in this case. The pre-set modes are in two other parameters, from which the first parameter acquires the values.

4.2 Main control panel

The main control panel window can be seen in Figure 12. The panel is split into two sides for both inverters, which are named accordingly at the top. Both sides needed the same panel objects, so the panel was to be symmetrical and easy to follow. The panel developed through testing, even though nothing was removed but objects were added.

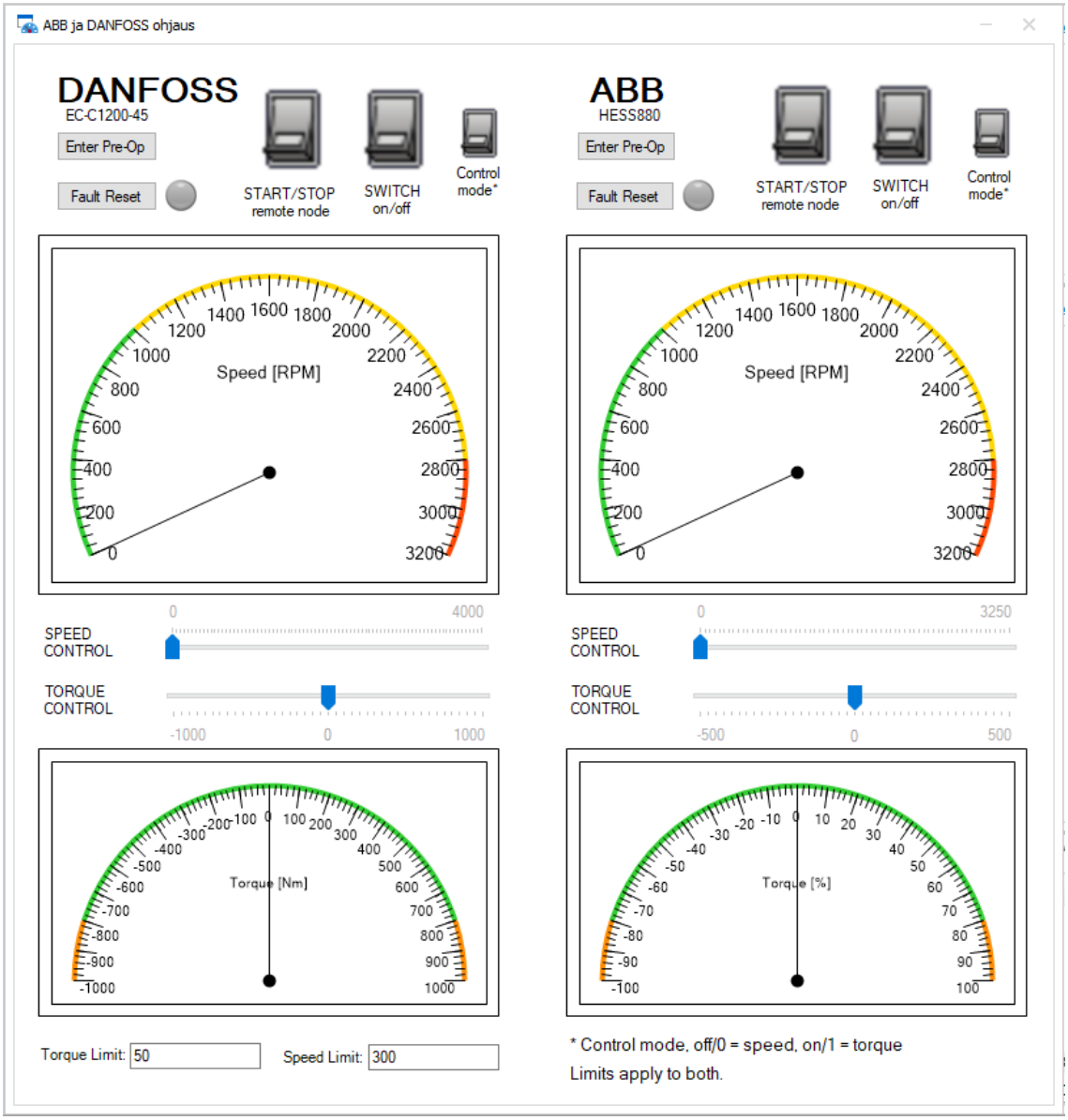

Figure 12. The main control panel.

At the top is the communication start and run start functions for both inverters and drive mode selector. Those are done by using two step switches, which has an indicator led for “on” position. In addition, a button for pre-operational state is included because it is needed for inverter configuration. Below, the buttons are the fault reset button and fault indicator light. Both motors have speed and torque gauges and sliders in the panel for drive control. At the bottom of the panel there are text boxes for limits and specifications for the control mode buttons.

As mentioned before, the objects in the panel require system variables because the panel is connected to a code. The system variables were created under the name of the inverter to tell the variables apart in the code. For example, the “Enter Pre-Op” button (Figure 12) has a system variable named “PreOpVar” for each but under different namespaces so the variables can be distinguished.

The code was created one button or switch at a time. In Figure 13 the part of the code for the pre-operational button can be seen. The “on sysvar” function, first seen on line 31, is triggered every time the system variable it is set to monitor changes value. In this example, it is connected to the HES880’s “PreOpVar”. When the button is pressed, the value changes from “0”, which is the initial value in this case, to “1” which is the wanted value.

```
35
36  // Enter pre-operational state
37  on sysVar HES880::PreOpVar
38  {
39    if (1 == sysGetVariableInt(sysvar::HES880::PreOpVar)){
40
41      msgA.NMTTarget = 3;
42      msgA.NMTCommand = 128;
43      output(msgA);
44      Tila = 0;
45      stopFlag = 0;
46
47    }
48  }
49
```

Figure 13. Lines 35 to 49 of HES880’s control code.

When the value changes, the code in the “on sysvar” function is run. The “if” function on line 33 is one of the basic functions in coding and used frequently in CAPL too. If the value of the button is “1”, then everything in the “if” function is run, otherwise nothing is done. First in the “if” function values for two signals are set in a message.

The “msgA” is the NMT message the signals are in, and it is defined in the beginning of the code (Figure 13). The two signals in the NMT message that need to be defined are “NMTTarget” and “NMTCommand”. The “NMTTarget” is the id of the receiving node, in this case HES880. The “NMTCommand” is the command number for the mode the inverter is set. For the pre-operational mode, the command value is 128.

After the signal values are set, the message can be sent. The “output” function on line 37 transmits the chosen message to the bus. The message is sent on the correct bus because the message is predefined in the code even if the CAPL node is connected to multiple networks.

The last two variables in the “if” function are “Tila” and “stopflag”. “Tila” is a variable that changes value according to which button is pressed. In this button it receives the value of “0” which means pre-operational mode in this code. The “stopflag” is a variable that receives the value “1” when stopping is requested, which is not with this button.

The functionality of the panel was tested throughout the project. Before the motors were attached to each other with the axle, both motors were configured and tested to ensure the function and safety. The code was also developed and reworked during testing when faults or problems were found.

4.3 Battery pack control panels

The battery pack control panels were made based on the battery pack control system, which has much more options and controls than were needed in this project. The main controls are the “CAN Tx” and “Start Request” (Figure 14). In

addition, an indicator light for general error was added so user knows if a KL15 reset is needed. KL15 is a physical switch in the electrical harness of the battery that simulates the ignition switch of a vehicle.

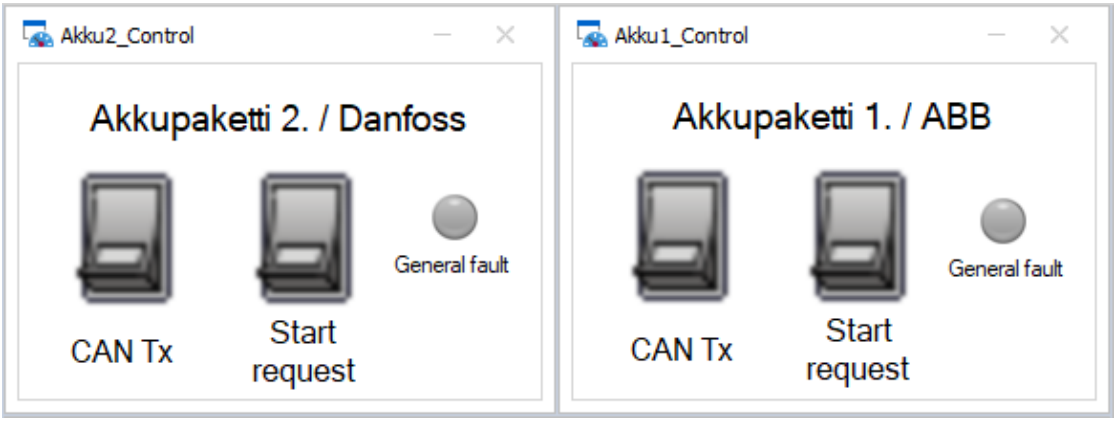

Figure 14. Battery pack 1 and 2 control panels.

The “CAN Tx” button starts the sending of the start message. This message is sent periodically, because if the message is not sent the battery will not stay on. This message contains a signal called “Start Request”. When the switch is turned on the signal gets the value of “1” and the battery will close its contactors. Then there is voltage in the DC link connector, also called the battery positive and negative.

4.4 Drive loop panel

The drive loop panel (Figure 15) was the last panel created because the functionality of the testbench was tested first with the main control panel. The purpose of the drive loop is to run a loop where, using the other motor as a motor and the other as a generator, one battery pack discharges and the other one charges. The panel allows the user to set the speed and torque values and voltage and SOC limits. The drive loop monitors the limits and when a limit is reached, the direction of the power changes but the physical turning direction does not change.

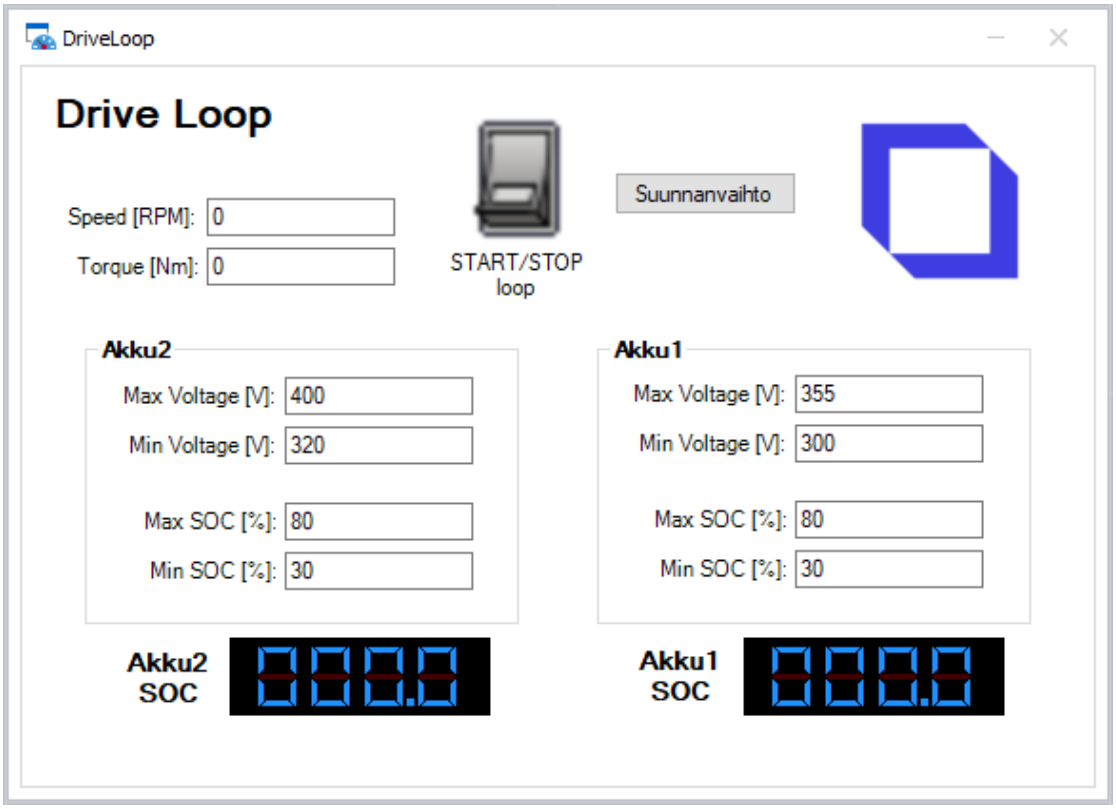

Figure 15. The drive loop panel.

For example, if battery pack 1 has a SOC of 40 and battery pack 2 has a SOC of 60, the drive loop will start to discharge the battery with higher SOC which would be pack 2. The generating motor is in torque control mode and the discharging motor is in speed control mode.

The code is written so that when the loop is started, it sets every speed and torque request to zero first in case the motors are running. Then the loop code checks that the battery pack values are between the limit values that the user has set. After that, the voltage and SOC level checks are done to determine which battery charges and discharges.

After all this is checked and decided, the speed and torque values, that the user has set, are set to the system variables of the speed and torque controls in the main panel. Then the motors are run until a limit is reached and the direction of power is changed if no errors occur. If a limit is reached, the program will set

speed and torque to zero and monitor the speed ramp down. When the speed is low enough, the speed and torque modes and references are set again.

5 Testbench

5.1 Design and rework

This project begun by removing the unnecessary and old parts from the testbench body. The ABB inverter and motor were attached to a rear axle from a vehicle for a previous project on the bench. The ABB components were moved to create space for new components. When Danfoss' motor and inverter arrived, the real mechanical designing could start.

Most of the mechanical design was made by a colleague with extensive knowledge, because the 3D designing was not a focus of this thesis. The 3D designing (Figure 16) was done on CATIA V5. One of the most important parts of the designing was to add sufficient support for the heavy Danfoss motor. The support design of the Danfoss' side was inspired by the ABB's side of the bench. In this way the bench also looks symmetrical but is sturdy enough to withstand the forces of the motors driving against each other.

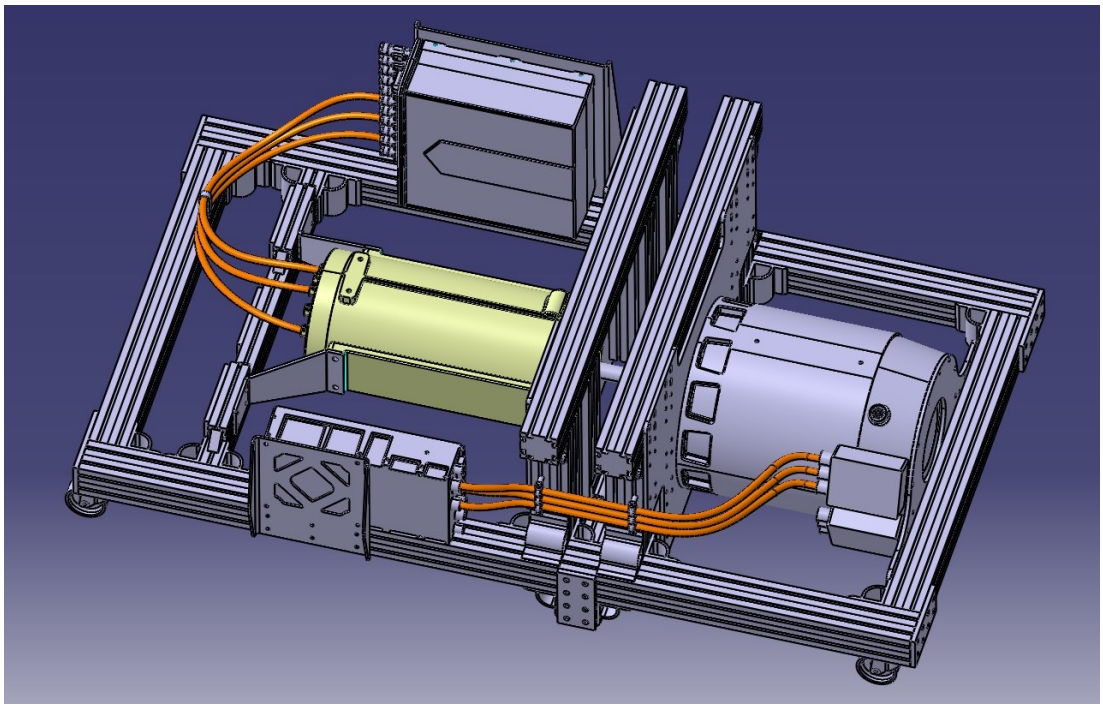

Figure 16. The testbench in CATIA V5.

The new Danfoss inverter needed a support plate. After all these, aluminium profile, support plates and most of the fasteners, were planned, the parts needed to be ordered. For the orders the suppliers needed to be chosen and contacted for a quote. After the parts arrived, the bench needed to be assembled according to the plans.

During the assembly of the testbench an overhead crane and a jack were used because only the Danfoss' motor weighs 300 kg. The overhead crane was used mainly when the Danfoss' motor needed to be moved and when it was put into place. The jack was utilized alongside the crane when the bench needed to be lifted from two spots.

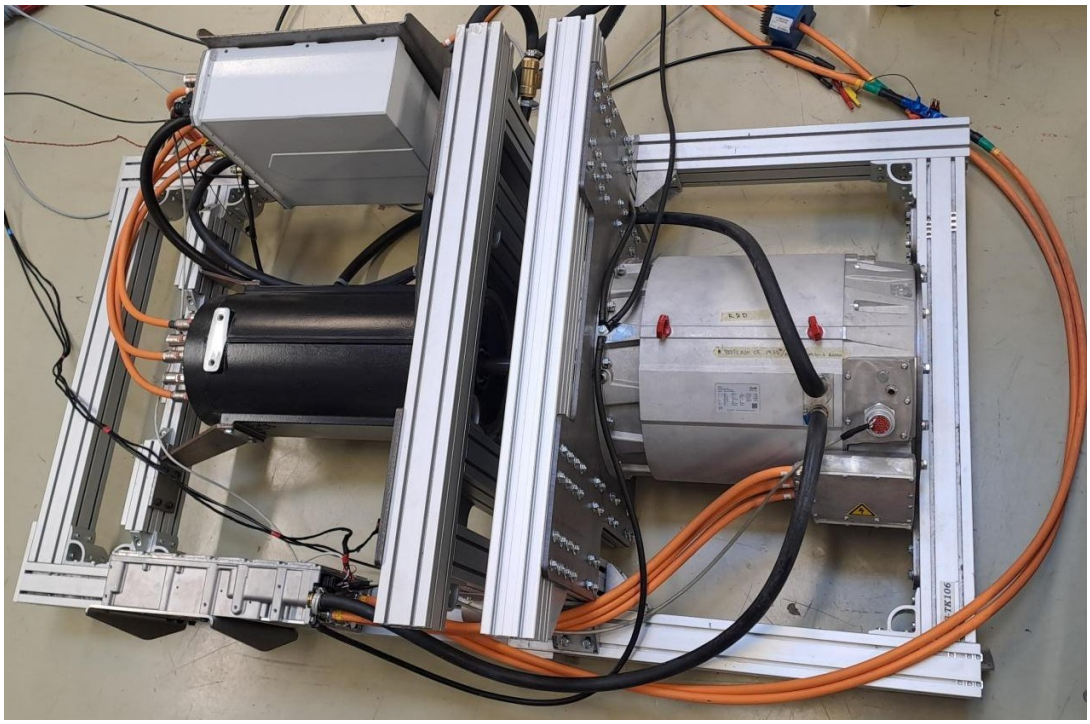

Figure 17. The testbench.

The whole motor inverter testbench required cooling, which was demonstrated during one test drive when EC-C1200 gave an overheating error. The cooling system was designed by the aforementioned colleague, because the cooling was not within the scope of this thesis. The cooling system uses LAUDA Variocool VC 5000 as the cooler and pump with 50/50 mixture of water and coolant. The coolant lines can be seen in Figure 17.

5.2 Electrical connections

In Figure 18. the topology can be seen. Battery packs and inverters are connected by high voltage DC link which consist of battery positive and battery negative. Both sets of invert and motor are connected by three phases. Commonly these three phases are named “U”, “V” and “W”. Every component body is connected with a ground cable to each other to prevent interference.

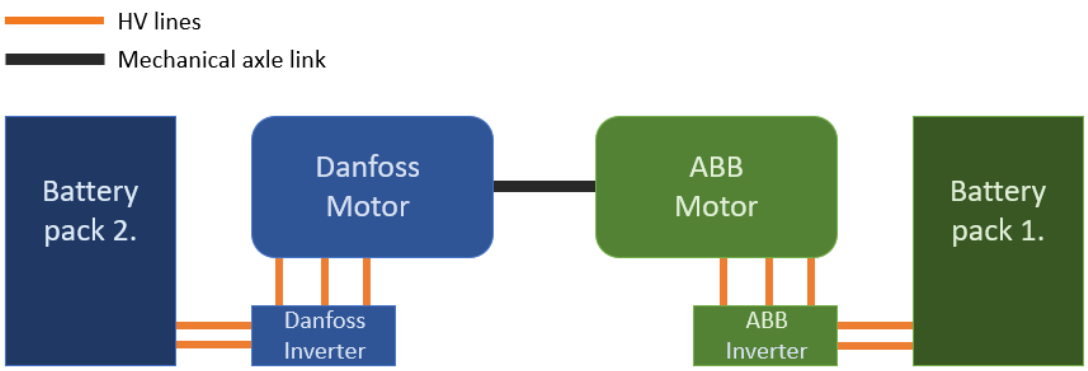

Figure 18. System architecture.

The ABB side of the HV and LV connections was already made, so only the DC link cables needed to be extended and a connector for the battery positive and negative was added. The Danfoss side of the HV connections were done according to Danfoss' requirements and instructions.

HES880 already had the required communication and operating lines to function, but an emergency stop button was added. EC-C1200 needed all communication and operating line to be added. PDU's, EC-C1200 and HES880 have operating voltage of 24 V.

Electrical connectors and pins for EC-C1200 and motor were ordered from a supplier. The CAN bus communication for EC-C1200 was made with a twisted cable pair, as recommended, and ground cable. The resolver of the motor was connected to the corresponding pins on the inverter. The resolver determines the position of the motor's shaft (Ellis 2012: 288).

6 Development and testing

The testing of the system was done throughout the project in safe environment with personal safety equipment. Small testing was done after every change to see the changes in action and if it was wanted and needed. First the motors were tested separately before integrating to the testbench to see and understand the controls and actions. The first test was done on both motors mainly to get them running on speed control mode through CAN bus. Then the other signals and messages needed for other control started to be implemented.

During testing, multiple mistakes in the code were found and quickly solved. For example, in a “if” function a system variable was compared instead of a signal, that was supposed to be compared to a set value. This was noticed when the drive loop behaved unexpectedly when switching control modes. During the inverter configuration problems rose and were solved. There were several problems with the control modes, but those problems were also overcome.

Some problems with the battery were noticed at the beginning of the project, but finding problems was one the goals of this project. One of the battery packs functional safety diagnostics gave a spurious error every time the motors would be at idle. Colleagues were enquired about the error code and a proper testing day was set up to uncover the problem.

The problem was interference in the system during idle and it triggered some safety parameters in the BMS. This was found and resolved during the testing day, when the battery pack received a new software where some voltage levels could be examined closely. After measuring the actual noise level and determining that operation was normal and expected, the system parameters were adapted to assure system compatibility.

Some unpleasant noises were discovered after attaching the mechanical axle link but occasional problems with ABB’s motor were already present before that. When driving on low RPM or stopping a very loud rattling noises started. These

rattling noises were traced back to ABB motor's encoder. After consulting ABB contacts, the rattling noises were minimized, and encoder problems solved. Some testing with moderate speed and torque requests have been done during the final weeks of the project.

7 Summary

The objective for this thesis was to produce a working testbench with PC based control system (Figure 19). During this project, the testbench was designed and reworked, the PC based control system was created and the whole system was configured to be used. From the view of functionality, the project was a success and the testbench works as intended.

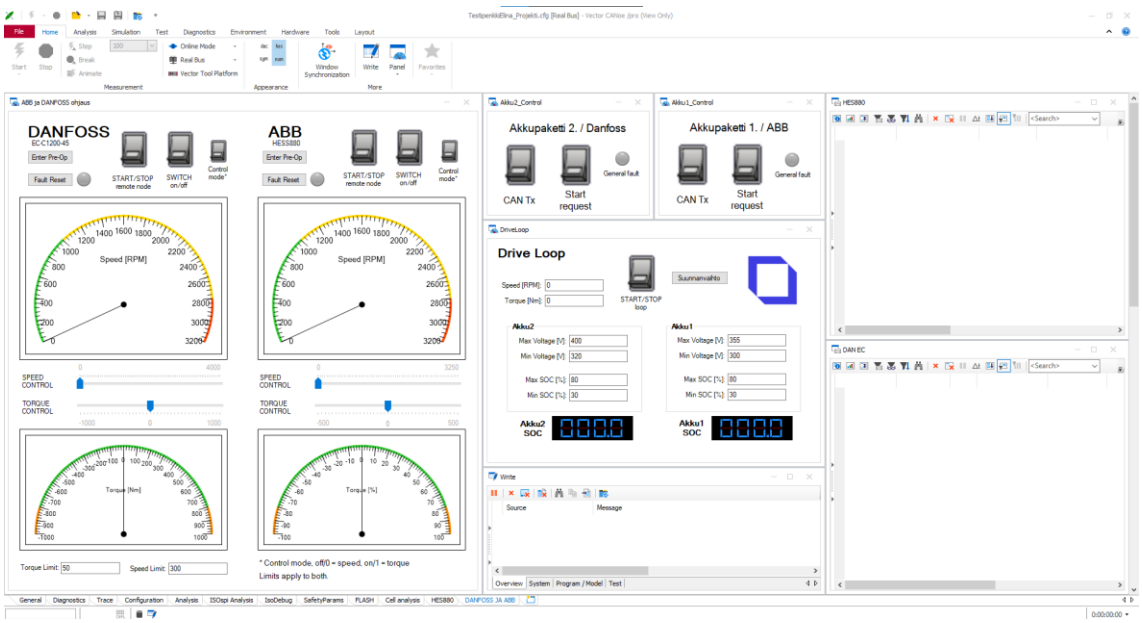

Figure 19. The final PC based control system for the testbench.

The project began with an introduction to the battery pack controls and CANopen and starting the reworking the testbench. After the components were confirmed the testbench design was finalised, parts were ordered and designing of the main control panel could start. With the first versions of the main control panel the function of the components was tested throughout the project and changes to

configurations was done. After all parts arrived and the testbench was built, the last weeks were spent on finalizing the control system and inverter configuration.

During the project there were multiple problems and difficulties as are likely in every project. Most noticeable problems were the delivery times of some packages. The inverter configuration problems also proved to be difficult and slow problems to resolve. All the problems slowed down the project timetable, even though the timetable was flexible.

Even though the testbench is working as planned it still requires some work, mainly on the visual side. The cable and coolant line managements are planned to be implemented soon. The splinter protection around the axle link is also planned and waiting for material and implementation. Some component changes are in planning as well as longer tests.

The testbench is going to be used in product testing with present and future battery products as planned. The testing with an actual powertrain is proven to be needed and useful to produce results because the environment is so different from other types of testing. The possible problems found in this testing can be then solved and are less likely to be present when in actual use.

Faults that could be found with this testing have mainly to do with interference in HV line and electromagnetic interference. In addition, the effect that the grounding and cable management could have, for example, on the communication lines can be studied. The safety parameters set for the batteries can be realistically tested with the testbench too.

The project came to an end successfully. With the system functionality proven in use, the target of the work was achieved.

References

AC Motor. 2024. Online material. Geeks for Geeks. <https://www.geeksforgeeks.org/ac-motor/>. Updated 1.5.2024. Accessed 13.6.2024.

Automotive Electrics and Automotive Electronics. 2007. E-book. 5th edition. Robert Bosch GmbH.

BU-205: Types of Lithium-ion. 2024. Online material. Battery University™. <https://batteryuniversity.com/article/bu-205-types-of-lithium-ion>. Updated 8.12.2023. Accessed 12.6.2024.

CANoe Famliy. N.d. Online material. Vector Informatik GmbH. <https://www.vector.com/int/en/products/products-a-z/software/canoe/#>. Accessed 12.6.

CANopen CC – The standardized embedded network. N.d. Online material. CAN in Automation. <https://www.can-cia.org/can-knowledge/canopen>. Accessed 5.8.2024.

CANopen internal device architecture. N.d. Online material. CAN in Automation. <https://www.can-cia.org/can-knowledge/canopen-internal-device-architecture>. Accessed 5.8.2024.

CAPL Used With CANoe And CANalyzer. N.d. Vector Informatik GmbH. <https://www.vector.com/us/en/know-how/capl/#>. Accessed 9.8.2024.

Ellis, George. 2012. Control System Design Guide. 4th Edition. Butterworth-Heinemann.

File:CAN-Frame mit Pegeln mit Stuffbits.svg. 2014. SVG-file. Wikimedia Commons. <https://commons.wikimedia.org/wiki/File:CAN-Frame_mit_Pegeln_mit_Stuffbits.svg>. Updated 12.3.2014. Accessed 15.7.2024.

High voltage electric inverters. N.d. Online material. Danfoss A/S. <https://www.danfoss.com/en/products/dps/electric-converters-motors-and-systems/electric-converters/high-voltage-electric-inverters/#tab-support-and-service>. Accessed 10.10.2024.

High voltage multipurpose motors. N.d. Online material. Danfoss A/S. <https://www.danfoss.com/en/products/dps/electric-converters-motors-and-systems/electric-motors/high-voltage-multipurpose-motors/#tab-overview>. Accessed 10.10.2024.

How electric vehicles work. 2024. Online material. Sustainable Energy Authority of Ireland. <https://www.seai.ie/plan-your-energy-journey/for-your-home/electric-vehicles/about-evs/how-electric-vehicles-work>. Accessed 12.6.2024.

Modular Battery Platform. 2023. Online material. Valmet Automotive. <https://www.valmet-automotive.com/electrifying/va-modular-power-pack/>. Accessed 10.10.2024.

Modular Power Pack, High Voltage. 2022. Online material. Valmet Automotive. <https://www.valmet-automotive.com/wp-content/uploads/2022/05/va-modular-power-pack-hv-data-sheet_2022_05.pdf>. Updated 1.5.2022. Accessed 13.6.2024.

Process data object (PDO). N.d. Online material. CAN in Automation. <https://www.can-cia.org/can-knowledge/pdo-protocol>. Accessed 5.8.2024.

Service data object (SDO). N.d. Online material. CAN in Automation. <https://www.can-cia.org/can-knowledge/sdo-protocol>. Accessed 5.8.2024.

Traction Inverters. N.d. Online material. ABB. <https://new.abb.com/electric-drivetrains/traction-inverters>. Accessed 10.10.2024.

VN8900. N.d. Online material. Vector Informatik GmbH. <https://www.vector.com/int/en/products/products-a-z/hardware/network-interfaces/vn89xx/>. Accessed 12.6.