CODEN:LUTEDX/(TEIE-5263)/1-55/(2009)

Industrial Electrical Engineering and Automation

Automated shifting of a manual sequential transmission in a hybrid vehicle

Axel Bergman

Per Byrhult

Dept. of Industrial Electrical Engineering and Automation
Lund University

Automated shifting of a manual sequential transmission in a hybrid vehicle

Master's Thesis

Axel Bergman and Per Byrhult

September 30, 2009

Supervisor:
Senior Instructor Bengt Simonsson
Examinator:
Professor Mats Alaküla

Lunds universitet
Lunds Tekniska Högskola

Lund University
Dept. of Industrial Electrical Engineering and Automation

Abstract

The increasing environmental awareness in today's society has created a demand for vehicles with less environmental impact. One way to meet the resulting need is to construct hybrid vehicles, hybrid operation which involves a combination of different energy conversion- and storage principles. This project deals with just such a hybrid solution, a parallel hybrid. To construct a competitive solution, it must both be comparable in terms of performance, comfort and price of existing vehicles on the market. Costs can be minimized by using standardized components to the extent possible.

The idea is that a standard internal combustion engine combined with a sequential gearbox and parallel to the combustion engine is coupled an electric motor. The electric motor can be used as the propelling engine, both in combination with internal combustion engine and on its own. In addition, it handles the synchronization of the gearbox at gear change. Together with a shift robot, fully automated and very rapid gear changes can be achieved, without disturbing the driver. This allows more gears to be introduced and the internal combustion engine can be kept at an optimal operating point that reduces emissions and lowers fuel consumption. The electric motor also substitutes the reverse gear, acts as a starter motor and alternator, reducing the number of components.

This thesis work deals with precisely this, namely to prepare and commission an automated sequential gearbox for a parallel hybrid. The project has been carried out at the Department of Industrial Electrical Engineering and Automation (IEA) at Lund University in collaboration with Saab Automobile in Trollhättan.

Sammanfattning

Det ökade miljömedvetandet i dagens samhälle har skapat en efterfrågan på fordon med mindre miljöpåverkan. Ett sätt att möta det uppkomna behovet är att konstruera hybridfordon, där hybriddriften innebär en kombination av olika energiomvandlings- och energilagringsprinciper. Detta projekt behandlar just en sådan hybridlösning, en parallellhybrid. För att konstruera en konkurrenskraftig lösning måste den både vara jämförbar vad gäller prestanda, komfort och pris med befintliga fordon på marknaden. Kostnaderna kan hållas nere genom att använda standardiserade komponenter i den utsträckning det är möjligt.

Tanken är att en standardiserad förbränningsmotor kombineras med en sekventiell växellåda och parallellt med förbränningsmotorn kopplas en elektrisk motor. Den elektriska motorn kan användas som framdrivande motor, både i kombination med förbränningsmotorn och på egen hand. Dessutom används den för synkronisering av växellådan vid växling. Tillsammans med en växlingsrobot kan man uppnå helautomatiserade och mycket snabba växlingar som ej föraren störs av. Detta tillåter att fler växlar introduceras och förbränningsmotorn kan hållas på en optimal arbetspunkt som minskar utsläppen och sänker bränsleförbrukningen. Elmotorn används även som backmotor, startmotor och generator, varvid antalet ingående komponenter reduceras.

Det här examensarbetet behandlar just detta, nämligen att sammanställa och ta i drift en automatiserad sekventiell växellåda för en parallellhybrid. Projektet har utförts på institutionen för Industriell Elektroteknik och Automation (IEA) på Lunds Universitet i samarbete med SAAB Automobile i Trollhättan.

Contents

Abbreviations

A/D	Analog-to-Digital
CAN	Controller Area Network
DSP	Digital Signal Processor
I/O	input/output
ISG	Integrated starter/generator
LED	Light-Emitting Diode
PCB	Printed Circuit Board
TTL	Transistor Transistor Logic

Chapter 1

Introduction

This chapter discusses the frame of this master's thesis and thereby explains the background, the problem statement and the aims of the project.

1.1 Risen need

The increase in awareness regarding environmental issues and rising fuel prices has evoked a need for energy efficient vehicles capable of competing with traditional means of transportation. Consumers expect the same level of performance, comfort and ease of use as in standard cars, to a competitive price.

"En osåld hybrid sparar ingen koldioxid" ("An unsold hybrid saves no carbon dioxide") [4], addresses the main issue and sets the foundation for this master's thesis. To get a competable product, it must be both energy- and cost efficient.

1.2 Background

This master's thesis is part of a PhD thesis project on designing a parallel hybrid drive system for a car. It is carried out at the Department of Industrial Electrical Engineering and Automation (IEA) at Lund University, in cooperation with SAAB Automotive [1]. The concept is a parallel hybrid with a combustion engine and an electric motor, both connected to the drive train. The main component is however the transmission, which in this concept is a sequential manual transmission. It features good efficiency and a low price which makes it a good choice in a car meant for the average car buyer.

To get the most out of a combustion engine, it's important to let it work with the correct load. The pump losses in an Otto engine are significant in most cases due to throttle action. This means that in most driving situations the engine works with even lower efficiency than optimal. An automated transmission can transfer the engine load to a better point without driver interaction. However, ordinary automated transmissions are heavy, complex, slow and features low efficiency. A better choice would be the automated manual transmission. The biggest drawback with available automated manual transmissions is however long shift times. When taking the shift operation from the driver, fast shifting is important to make a pleasant drive. This is where the electrical synchronizing greatly can improve the performance. By substituting the synchronizing discs

in a traditional manual transmission by an external electric motor taking care of the synchronization, shift times can be drastically improved. In addition to this, the same electrical machine is used for propelling the vehicle as well as it is used as starter motor and generator. The combination of a sequential transmission, an electric motor both for synchronization and propulsion, a shift robot conducting the shifting, together constitutes a promising solution of a hybrid vehicle drive train.

1.3 Aims of the study

The main objective of this study is to automate a sequential manual transmission to be utilized in a hybrid vehicle, as discussed in the previous section. This involves setting up a laboratory system including the integration of an electric traction motor taking care of synchronizing as well as propulsion of the vehicle. In addition a shift robot shall be integrated to the system. The end product is a laboratory setup capable of showing the performance of an electrically synchronized automated manual transmission. The work involves the following phases:

- Selection of a suitable temporary traction motor for the laboratory setup
- Design and build of a transmission connecting the electric motor to the gearbox
- Collection of gearbox data for shift robot specification
- Setup of dSpace hardware and software
- Commissioning of the electric drive and gear shifting robot
- Optimization of the shift process
- Evaluation of measured results

A dedicated traction motor is being developed in parallel with this master's thesis and is to replace the temporary motor. This work is not a part of this thesis and is carried out by PhD student Yury Loayza [12]. However this specially developed traction motor is to be integrated in to the laboratory setup at a later stage, improving the shift process even further. This requires preparation of the laboratory equipment for both electrical machines.

Chapter 2

Previous work

To acquire deeper knowledge of the area of research, two previous master's theses were studied. One of them, an entirely theoretical study and the other is a project that also includes practical tests on a laboratory setup.

2.1 Theoretical study

The theoretical pre-study [10], is about combining the price, simplicity and efficiency of a manual transmission with the comfort level of an automatic transmission. Automated manual transmissions have an advantage over automatic transmissions in terms of cost, complexity, efficiency and weight, but the disadvantage of shift comfort [10, p. 21]. The main aim of the thesis was to arrange one or more electrical machines together with a manual transmission in a way that synchronizing of the transmission is accomplished by the electrical machine(s). This has the potential of improved shift times and therefore also shift comfort due to less torque interrupt. Furthermore the arrangement should allow hybrid operation.

The overall guidelines for a manual transmission suitable for this concept is worked out. A main requirement of the study is that the concept should fit to current SAAB engines in current SAAB platform. The SAAB F-35 transmission is used as a base and a series of modifications are suggested to fit the above concept criterias. It is proposed that geometrical gear steps should be used instead of progressive steps used in most passenger vehicles [10, p. 30]. The reverse gear is suggested to be fully electrical and the number of gears increased in order to minimize the rpm ratio between the individual gears. Furthermore a sequential gearbox with racing dog teeth [10, p. 39] as gear engagement mechanism is proposed. The power demands for synchronizing, fully electrical drive and hybrid operation are discussed as well as different options of electrical machine placement. This leads to two options. One of them includes two electrical machines: an ISG for hybrid and electrical drive and a servo motor taking care of synchronizing. The other option involves only one external motor for both hybrid and fully electrical drive as well as for the synchronizing. The two concepts are further developed, evaluated and compared and the conclusion is that the single external electrical machine is the most suited concept.

2.2 Practical concept study

The main objective of this concept study [5] was to verify whether the concept of an electrically shifted sequential gearbox was a feasible way to go. The conclusions from the theoretical study was adopted in this study. With help from SAAB, a test rig was built, including a dog teeth sequential gearbox, a shift servo and a flywheel to simulate accelerations and decelerations. A single electrical machine was used, taking care of both synchronization and electrical drive as described in section 2.1 above.

The results showed that more than 80 % of the shift time lay in the actuation of the shift servo [5, p. 49]. The synchronization of the gearbox took about 15 % of the shift time [5, p. 49], keeping in mind that the tests were done only between gear four and five [5, p. 9]. This beeing a relatively small gear step, thereby the importance of an even faster shift servo and synchronizing electric motor would be preferable to minimize the shift times in between gears with a greater ratio. Also rebuilding the gearbox would be benificial, adding more gears to level out the ratio to get smoother steps in the shifts. The increase in number of gears would not be a problem since the driver is not doing the shifting.

2.3 Pre-study conclusions

The pre-study resulted in these important things to improve:

- Faster shift servo
- Optimized electrical machine to improve acceleration/deceleration times
- Optimized gearbox with more gears at an even ratio. This is to be implemented in the final product but will not contribute to the outcome of this project.

However this project is due to the time limitation of 20 weeks not aiming to improve the gear box and since the optimized electrical machine supposed to be used was in production a temporary machine found in house had to substitute.

Chapter 3

Theoretical outline

The theory behind the automated shift sequence is described, the theoretical performance of the system is calculated and the goal of the project is set.

3.1 Shift sequence

To obtain a smooth shift operation a sequence of actions has to be executed. The sequence begins with setting the transferred torque through the transmission to zero. This enables the teeth connecting gear and shaft to disengage. The next step is moving the dog teeth ring to neutral position. Since two gears cannot be engaged at the same time there is a neutral position between all individual gears where input shaft and output shaft are mechanically disconnected from each other. This mechanically disconnected state is used to synchronize the speed of the shaft and next gear. When the velocity is matched the dog teeth ring is further moved to engage next gear. At this point torque can again be transferred and the sequence is completed. The sequence is illustrated in figure 3.1.

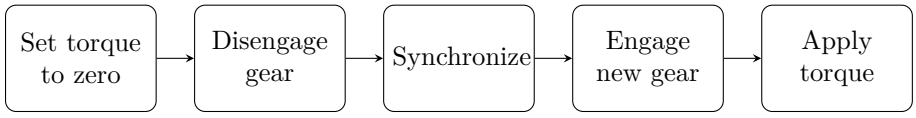


Figure 3.1: The five steps of the shift sequence.

3.2 Theoretical shift times

In order to determine the theoretical limitations of the system, as it would be designed integrated in an car, theoretical shift times were calculated. Primarily the synchronization time were calculated, but also communication delays and shift robot actuation were looked into.

3.2.1 Communication times

Communication times are hard to estimate and calculate without the knowledge of the future system design. But since the solution is going to be optimized for just the purpose of doing a fast and reliable execution of the shifting, it will most certainly be very fast. Probably so fast that it will be negligible compared to the synchronization and shift robot actuation times.

3.2.2 Synchronization time

The following synchronization times are based on the dedicated electrical machine specifications. This machine is designed to be connected to the gearbox input shaft via an external transmission. The ratio of this transmission is set to 2:1 (electrical machine rpm > gearbox input shaft rpm). An interesting notice is that this contradicts the recommendation in [10, p. 70] of a 1:2.2 ratio (electrical machine rpm < gearbox input shaft rpm).

To be able to calculate synchronization times, the total moment of inertia involved in the synchronization process must be estimated. The moment of inertia of a thick-walled cylinder with density (ρ), length (l), outer radius (R) and inner radius (r) is expressed in equation 3.1.

$$J = \frac{1}{2}\,\pi\rho\; l(R^4 - r^4) \tag{3.1}$$

This equation is used to calculate the moment of inertias of the electrical machine, J_{shaft} (equation 3.2), $J_{rotorstack}$ (equation 3.3) and a chain drive sprocket of estimated size connecting the electrical machine to the gearbox, $J_{em\_sprocket}$ (equation 3.4). The density is approximated to $7850\,\mathrm{kg/m^3}$.

$$J_{shaft} = \frac{1}{2}\,\pi \cdot 7850 \cdot 0.46(0.016^4) = 0.37 \cdot 10^{-3} kgm^2 \tag{3.2}$$

$$J_{rotorstack} = \frac{1}{2}\,\pi \cdot 7850 \cdot 0.21(0.043^4 - 0.016^4) = 8.7 \cdot 10^{-3} kgm^2 \tag{3.3}$$

$$J_{em\_sprocket} = \frac{1}{2}\,\pi \cdot 7850 \cdot 0.01(0.03^4) = 0.10 \cdot 10^{-3} kgm^2 \tag{3.4}$$

The total moment of inertia of the electrical machine is according to equation 3.5:

$$\begin{aligned} J_{em} &= J_{shaft} + J_{rotorstack} + J_{em\_sprocket} \\ &= (0.37 + 8.7 + 0.10) \cdot 10^{-3} \\ &= 9.2 \cdot 10^{-3} kgm^2 \end{aligned} \tag{3.5}$$

Since there is no available gearbox data, the moment of inertia is assumed to be $7.9{\cdot}10^{-3}\,\mathrm{kgm^2}$ [10, p. 36]. In addition to the moment of inertia of the gearbox itself, the moment of inertia of another sprocket, connected to the gearbox input shaft, is estimated (equation 3.6).

$$J_{gb\_sprocket} = \frac{1}{2}\,\pi \cdot 7850 \cdot 0.01(0.06^4) = 1.6 \cdot 10^{-3} kgm^2 \tag{3.6}$$

The gearbox total moment of inertia is:

$$J_{gb\_tot} = J_{gb} + J_{gb\_sprocket} = (7.9 + 1.6) \cdot 10^{-3} = 9.5 \cdot 10^{-3} kgm^2 \tag{3.7}$$

As mentioned in [10, p. 36] a transmission alters the effective moment of inertia of the rotating masses after the transmission by a factor $(1/\mathrm{U})^2$ where U is the transmission ratio. The electrical machine is connected to the gearbox via a transmission with a ratio of 2:1. The sum of the moment of inertias involved in the synchronization process is shown in equation 3.8.

$$\begin{aligned} J_{tot} &= J_{em} + \left(\frac{1}{2}\right)^2 J_{gb\_tot} \\ &= 9.2 \cdot 10^{-3} + \left(\frac{1}{2}\right)^2 9.5 \cdot 10^{-3} \\ &= 12 \cdot 10^{-3} kgm^2 \end{aligned} \tag{3.8}$$

The synchronizing work is described in equation 3.9, with work (W), moment of inertia (J), angular velocity before synchronization (ω_1) and angular velocity after synchronization (ω_2).

$$W = \frac{1}{2} J(\omega_1^2 - \omega_2^2) \tag{3.9}$$

Also:

$$P = \frac{W}{t} \iff W = Pt \tag{3.10}$$

By merging equation 3.9 and 3.10, the theoretical synchronization time (t) can be calculated by inserting a known power (P), the calculated total moment of inertia (J_{tot}) and the angular velocities (ω_1) and (ω_2), according to equation 3.11.

$$Pt = \frac{1}{2} J(\omega_1^2 - \omega_2^2) \iff t = \frac{1}{P} \frac{1}{2} J(\omega_1^2 - \omega_2^2) \tag{3.11}$$

The electrical machine developed in parallel to this thesis has a constant torque of 45 Nm up to its base speed of 3000 rpm. Above the base speed it maintains constant power all the way to the maximum angular velocity of 15000 rpm due to the influence of field weakening. This means that the power between 3000 rpm and 15000 rpm in continuous operation is just above 14 kW. However in short periods, the output power can be doubled.

Using equation 3.11 the synchronization times shown in table 3.1 were computed. These times represents the maximum theoretical synchronization times as they are calculated at maximum angular velocity, thus the greatest angular velocity steps.

A full rpm shift from first to second gear results in a synchronization time of 270 ms. This is unacceptable. By reducing the electrical machine shift rpm by a factor two, to 7500 rpm, the synchronization time is reduced to 100 ms.

From equation 3.11 it can be concluded that the angular velocities have a great impact on the synchronization time. By reducing the external transmission

Table 3.1: Theoretical synchronizing times at full rpm gear shifts and an external transmission ratio of 2:1

Gear	**Peak power [kW]**	**Moment of inertia [kgm^2]**	ω_1 **[rpm]**	ω_2 **[rpm]**	**Time [s]**
$1 \rightarrow 2$	28	0.012	15000	10253	0.27
$2 \rightarrow 3$	28	0.012	15000	12500	0.16
$3 \rightarrow 4$	28	0.012	15000	12667	0.15
$4 \rightarrow 5$	28	0.012	15000	11447	0.21

ratio to 1:2, the electrical machine angular velocity would decrease to 3750 rpm at maximum gearbox input shaft angular velocity. With the same electrical machine, the peak power would remain at 28 kW. A drawback of altering the ratio is however that the reflected inertia of the gearbox would increase. The results from just altering the external transmission ratio is presented in table 3.2.

Table 3.2: Theoretical synchronizing times at full rpm gear shifts and an external transmission ratio of 1:2

Gear	**Peak power [kW]**	**Moment of inertia [kgm^2]**	ω_1 **[rpm]**	ω_2 **[rpm]**	**Time [s]**
$1 \rightarrow 2$	28	0.042	3750	2563	0.063
$2 \rightarrow 3$	28	0.042	3750	3125	0.036
$3 \rightarrow 4$	28	0.042	3750	3167	0.034
$4 \rightarrow 5$	28	0.042	3750	2862	0.049

The drastically reduced synchronization times shows that it would be interesting to optimize the external transmission at the same time as optimizing the electrical machine. This is however not the point of this thesis.

Finally it should be noticed that there is a significant error in the power calculations carried out by [10, p. 50-51] resulting in two incorrect tables, since:

$$W \neq \frac{1}{2} J(\Delta\omega)^2 \tag{3.12}$$

3.2.3 Shift robot time

The shift robot used in the project is based on a Lönne (type 7AA71M02) [11] three phase induction motor. The motor is run as a servo with position feedback and its shaft is connected directly to the shift mechanism of the gear box. The motor features a power (P_n) of 0.55 kW, a base speed (n_n) of 2800 rpm and a starting torque factor (T_{start}/T_n) 2.6 .

$$P_n = T_n \omega_n = T_n \frac{2\pi n_n}{60} \iff T_n = \frac{P_n \cdot 60}{2\pi n_n} \tag{3.13}$$

The starting torque is then calculated by:

$$T_{start} = 2.6T_n = 2.6\frac{P_n \cdot 60}{2\pi n_n} \approx 4.9\,\text{Nm} \tag{3.14}$$

Measurements on the gearbox shows that a maximum force of 250 N is required to move the shift mechanism at a static state. To actuate the gearbox a 18 mm long lever 5.2 was fitted to the robot.

This results in a force of 270 N which is enough to move the lever. Due to the unknown dynamics during actuation of the shifting mechanism the time required to do the shifting was not calculated.

3.3 Project goal

The most important feature when automating a transmission is to obtain a pleasant drivers experience. This is obtained by keeping the loss in torque transfer while shifting to a minimum. To get a pleasant drivers experience an approximated time of 100 ms was set, obtained from test drives of vehicles with various automated transmissions [4]. This therefore sets the maximum of the time critical part of the shift sequence in this project.

Chapter 4

Laboratory setup

In this chapter the laboratory setup is described, including rig, electronics, control software and shift robot.

4.1 Laboratory hardware

The laboratory setup used in the previous practical study mentioned in section 2.2 was partly reused in this project. The bench including sequential gearbox and flywheel was reused but shift robot, electrical machine and control software were replaced. A belt drive has been designed and is connecting the ingoing axle and the electrical machine. The belt drive was built to fit both the temporary servo motor and the specially developed motor mentioned in section 1.3.

The complete bench is shown in figure 4.1, with a flywheel in the upper part of the figure, the electrical machine and belt drive in the lower left part and the shift robot covering the gearbox in the lower right part of the figure.

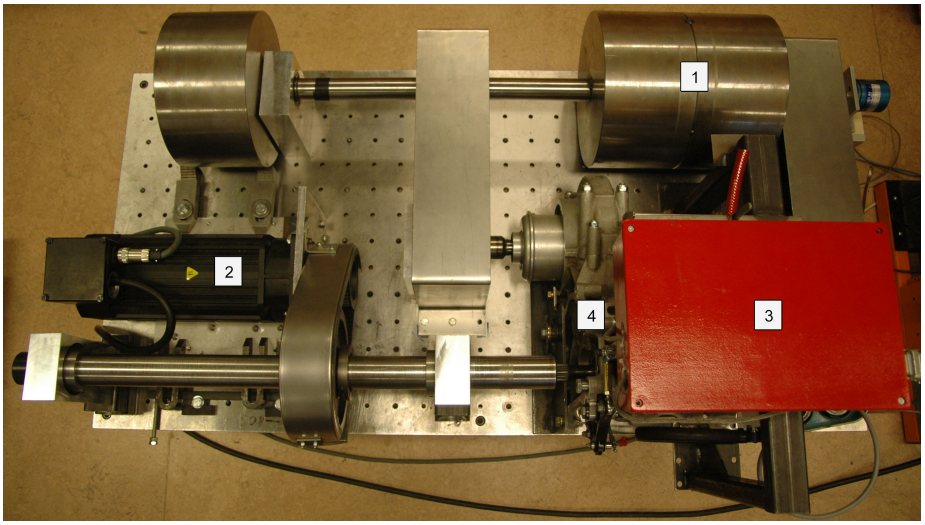


Figure 4.1: The laboratory setup. (1) flywheel, (2) electrical machine, (3) shift robot, (4) gearbox.

4.1.1 Gearbox

The gearbox used in the laboratory setup is built by Xtrack [16]. It features sequential shifting [10, p. 38] and six gears. The gearbox is a racing dog teeth transmission as suggested in the previously studied theoretical study [10]. The transmission gear ratios are presented in table E.1 in appendix E. As can be seen, the steps are neither geometrical nor progressive [10, p. 29-30]. This was however not a problem since the tests only were to be conducted between gear four and five. Running the tests between fourth and fifth gear enables comparison of the results with the results of the previous study [5, p. 45-47].

Gearbox measurements

In order to set the overall requirements of the shift robot some gearbox data was required. Due to the absence of gearbox specifications the data needed was measured. In addition to shift lever travel the force needed for shift lever actuation was measured.

The gear lever travel was measured with a vernier caliper[1] while moving the lever accurately with a turnbuckle. Four positions were measured for every gear: the starting position of the measurement, the beginning of the neutral region (gear disengage point), the end of the neutral region (new gear engage point) and the shift lever end position. Data was measured for all six gears, both while up shifting and down shifting, in two sessions. The data is presented in table C.1 and table C.2 in appendix C. Mid positions for neutral intervals were calculated from the measured data and indicates a lever movement of approximately 5 mm to reach neutral position and an other 5 mm to engage next gear. The distances differs quite a lot between different gears. This lead to a requirement of individually configurable shift robot movement patterns for all five gear steps.

The force needed to actuate the shift lever was measured using a spring gauge. This method does not take gearbox dynamics into consideration but gives a fair hint on the force needed. The force required to move the shift lever was measured to be 250 N. The measured lever travel and force data was used to set the specification of the shift robot. Binar Elektronik AB [3] was involved in the project by SAAB and helped configuring the shift robot used in the project.

4.1.2 Electric motor

A temporary motor had to be picked to substitute the specially developed motor, see chapter 1.3, during its development and production. The main purpose of the temporary motor was to be able to commission the laboratory setup in an early stage and prepare the setup for the specially developed motor. An Atlas Copco servo motor was found capable of doing the job. The maximum angular velocity of this machine is limited to 6000 compared to the 15000 of the specially developed one. This results in lower angular velocity when evaluating the shifting. However, this did not affect the evaluation of shift process since the principals could be tested and the final performance was not of interest at this stage.

[1] Caliper with a vernier scale used for very fine measurements.

4.1.3 Shift robot

The shift robot developed by Binar is based on an asynchronous machine and has built in control- and power-electronics. It can be programmed via an Ethernet connection to do a great variety of tasks, whereas the robot can be set to do the tasks required to shift the gearbox. The Ethernet connection also makes it possible to set up and monitor robot parameters from a PC. The robot was set up by Binar according to the specifications set in section 4.1.1. As for the communication there were two options. Either the robot could be configured to be controlled via CAN or via a set of parallel digital signals. The latter was chosen after discussions with both Binar and Mats Alaküla due to its simplicity. A communication module with Wago [14, 15] input- and output units was added to the robot to enable parallel communication. The signals used to control the shift robot are presented in table 4.1. Multiple control signals for going to neutral position was used so the travel could be individually configured for all neutral positions according to section 4.1.1.

Table 4.1: Table of the digital signals used to communicate with the shift robot.

Signal	Direction	Description
N1	input	Goto neutral position between gear 1 and 2.
N2	input	Goto neutral position between gear 2 and 3.
N3	input	Goto neutral position between gear 3 and 4.
N4	input	Goto neutral position between gear 4 and 5.
N5	input	Goto neutral position between gear 5 and 6.
DIR	input	Sets the direction of the movement.
MAX	input	Goto Max position (engage gear).
HOME	input	Goto Home position (starting position).
ACK	output	Enabled when shift robot completed the last task assigned and is ready to execute a new one.

A fitting for the shift robot was built to attach the shift robot to the gearbox and the robot was equipped with a lever to shift the gearbox as can be seen in figure 4.2.

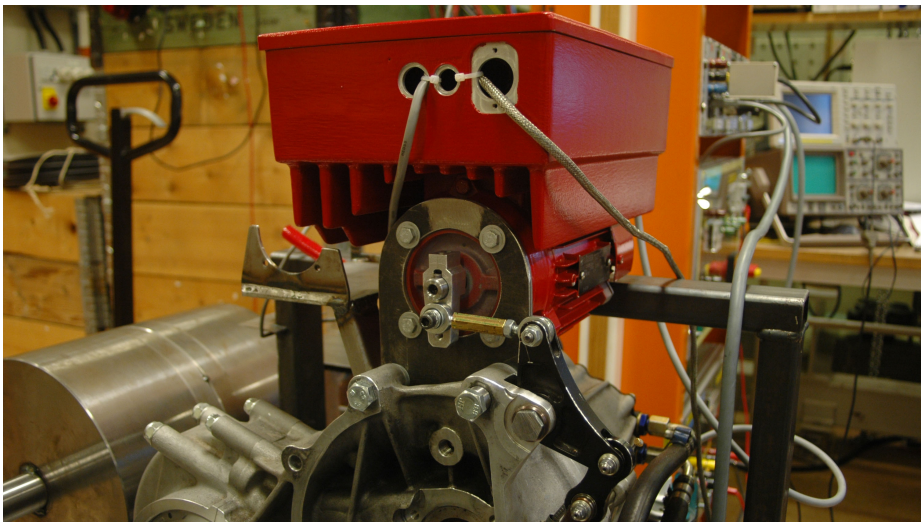


Figure 4.2: The shift robot.

4.2 Electronics

The dSpace system including power electronics is available at IEA as a modular system easily configurable in a laboratory setup. The electronic system was put together using modules as well as some tailor made electronics.

4.2.1 dSpace system

The development environment of this project is based on a dSpace [7] system. This environment incorporates both a hardware PC-card and a software interface well suited for developing mechatronic projects. The system consists of a DS1104 [8] PCI extension card featuring a real-time DSP with a number of inputs and outputs.

The DS1104 card is connected to an interface module used in the IEA laboratory setup via a ribbon cable. The setup can be seen in figure 4.3. This module distributes input and output pins to the back connector of the setup. The connector is used to connect various data collection modules and the power electronics to the dSpace system. However the digital I/O:s used in this project to control the shift robot were not distributed by the interface module. To enable the use of the digital pins, a pin connector was soldered to the experimental area of the board and wires were soldered between this connector and the digital pins at the incoming ribbon connectors.

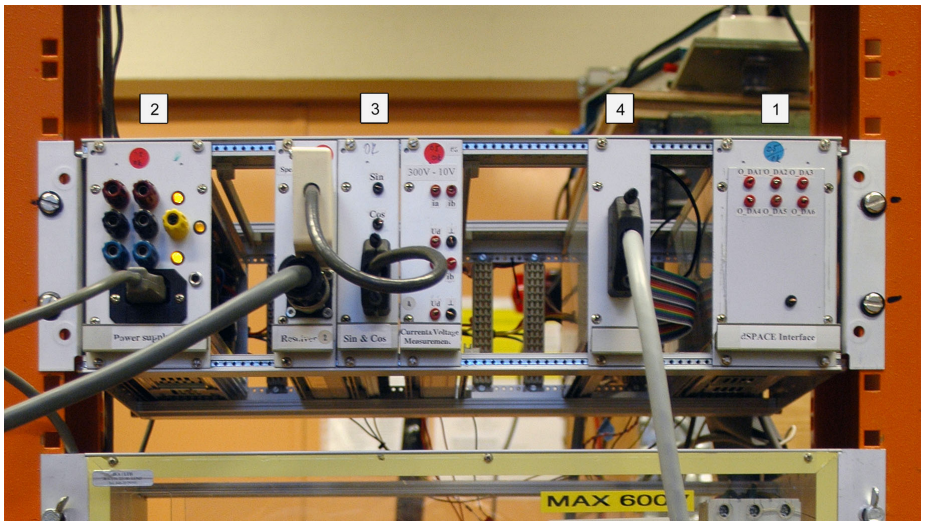


Figure 4.3: The dSPace system. (1) interface module, (2) power supply module, (3) data collection modules, (4) lever shifter.

Level shifter between dSpace and shift robot

The dSpace digital input/output unit operates with standard TTL levels [6]. The Wago input and output modules of the shift robot however operates with 24 V signals [14, 15]. To enable communication between the two a interface unit was designed. It's a universal unit in therms of physical appearance, as it has

the same connector as the other setup modules and is mounted in a dedicated slot-in box fitting the dSpace setup at IEA. Furthermore it features level shifting of all twenty bit I/O signals of the dSpace bit I/O unit. Even if not all signal lines are used in this project it makes the unit adaptable in future projects. For the same reason all inputs and outputs are optocoupled. The inputs and outputs can be reached via a D-sub connector mounted on the front panel, as well as via the back connector. The output signals are buffered via 74HC240 [13] inverting buffers which drives the optocoupler diodes. The load is connected between optocoupler emitter and ground. On the inputs, the external unit is driving the optocoupler LEDs. The 74HC240 buffer inputs are driven high via pull-up resistors or pulled low by the optocoupler transistors. Full schematic and PCB layout are shown in appendix A.2 and appendix A.1 respectively.

4.2.2 Gear drum position feedback

A potentiometer integrated in the gearbox is used to get position feedback of the gear shifting mechanism. The potentiometer is connected to the in section 4.2.1 mentioned dSpace interface module as a voltage divider and the output is A/D-converted in the software to be able to calculate the currently selected gear.

4.2.3 Power electronics

Also the power electronics is a part of the laboratory setups at IEA. Initially a quite modest power converter with a maximum output current capability of 50 A was used. This was more than enough for commissioning the system with the temporary servo motor. Meanwhile, a higher rated version of the power electronics was prepared. This version features water cooling and a maximum current capability of 300 A.

4.3 Software

To control and set up the laboratory environment two programs were used, one to set up and control the laboratory process in real-time (ControlDesk) [9] and another to set up the shift robot (CoDeSys) [2].

4.3.1 ControlDesk

The DS1104 interface card is controlled via a software called ControlDesk. ControlDesk features both Simulink model- and C-code controlled laboratory setups. Simulink models are compiled into C in Matlab and then linked to ControlDesk, while C-code can be written directly in the program and compiled by a built-in compiler. After compiling and configuring the software it is downloaded to the dSpace card and run.

This project requires very fast response times from the board/source code, whereas an entirely optimized C-code controlled setup is required. This since a Matlab generated code will add on a lot of scrap code, slowing down the response of the operation of the board, which is not desirable.

ControlDesk features linkage of variables in the source code to a graphical interface in the program. This interface can contain buttons, graphs etc. which

enables the user to set up a user friendly interface that controls the laboratory process in real-time.

ControlDesk interface

An interface has been designed to control the shift process. It contains buttons controlling the up- and down shifts and gauges to show the angular velocities of the ingoing and outgoing axles of the gearbox. A plot of the different states during the shift process is also shown in the interface, this enables easy overview and debugging of the process in real-time. Figure 4.4 illustrates the ControlDesk interface.

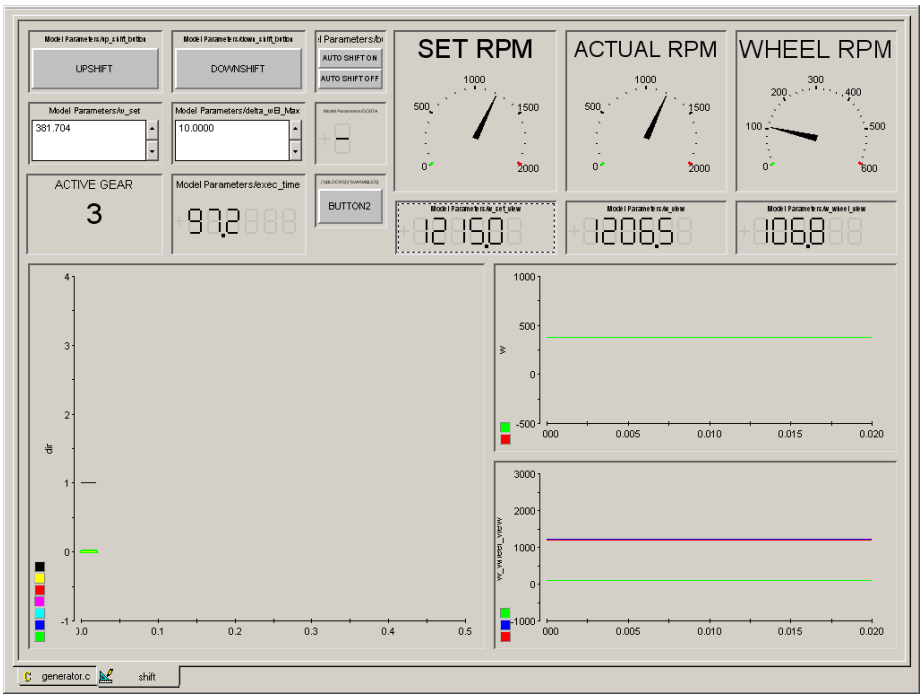


Figure 4.4: The ControlDesk interface.

4.3.2 Source code

The software controlling the shift process is entirely written in C-code and can be studied in deep in the appendix F. The source code is based on four major blocks: two interrupt routines, one block of functions and one main loop.

The core of the shift process is the timer interrupt that is executed every 1E-4 s. It handles the control of the electrical machine, shift robot and errors that might occur. The functions block contains functions for initiating the system, calculating the current gear, calculate the synchronizing rpm during shifts etc. The main loop only contains a non-time critical function checking whether any buttons are pushed in the control program.

4.3.3 CoDeSys

The robot has a setup interface via a software called CoDeSys which communicates with the robot via Ethernet. The CoDeSys interface custom built by Binar offers a number of tunable parameters that easily can be changed and uploaded to the robot.

The robot can be run in two modes, automatic- and manual mode. In automatic mode the robot is controlled by the dSpace system and in manual mode the robot is controlled via CoDeSys by two jog buttons. The robot speed can be adjusted allowing the user jogging in very low speeds when operating in manual mode. The other adjustable parameters concerns robot lever travel. The distance to neutral can be tuned individually between all six gears, there are even a distinction between up- and down shifts. This was suggested because the gear lever travels measured in subsection 4.1.1 showed quite a large distribution between the gears. The neutral range is quite small between two adjacent gears which require precise adjustments. Moreover, the maximum travel for both up- and down shifts can be set as well as the home position. All robot lever travel parameters also have tolerance settings. They are used to individually set a range where the robot sets the ready pin. A picture of the setup screen can be seen in figure 4.5.

Another CoDeSys feature is the ability to plot signals as a function of time. This was used for evaluating parameter changes and shift robot performance.

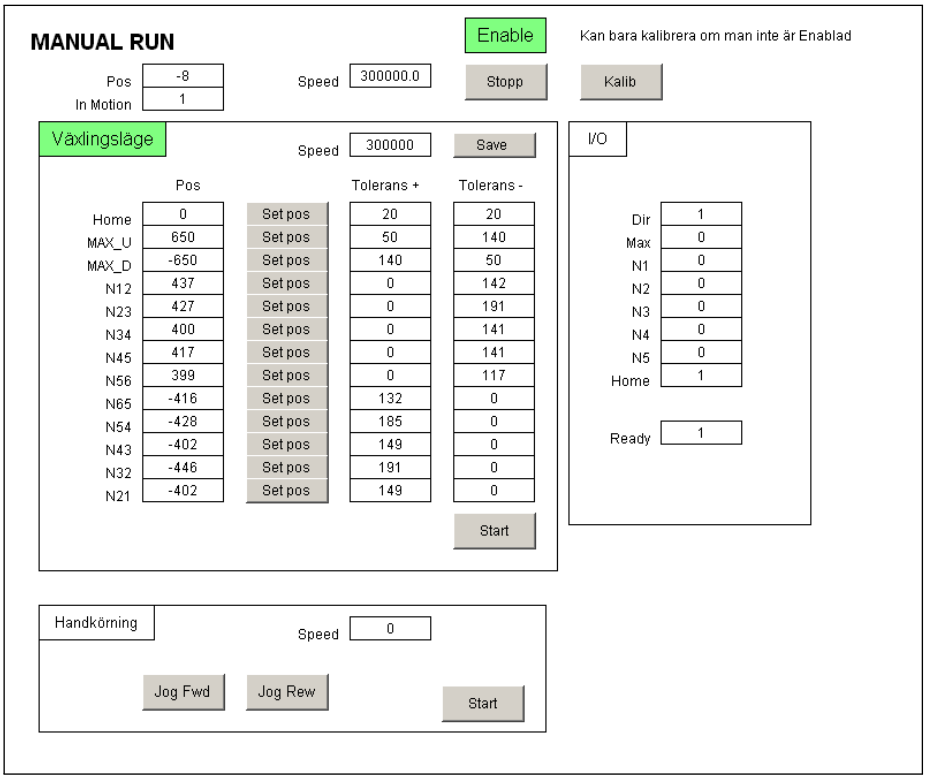


Figure 4.5: The CoDeSys interface.

4.4 Sum up of laboratory equipment

In the figure 4.6 below a sum up of the entire laboratory setup used in the project is presented.

The main controlling unit of the system is the PC with the DS1104 expansion card. It is connected to the dSpace interface module in the laboratory setup which is distributing signals to, and from, the other components in the system. The shift robot communication module is connected via a level shifter to the interface module. The robot is controlled by nine digital signals constituting a parallel bus. The signals are connected according to table D.1 in appendix D. The communication module interprets the parallel signals and communicates with the robot via a CAN-bus. An Ethernet connection between the robot and the computer enables communication with CoDeSys for setup and signal monitoring.

The power electronics are also connected via the interface module. It controls the electrical machine. Velocity signals are fed back to the interface module module both from the electrical machine and the flywheel.

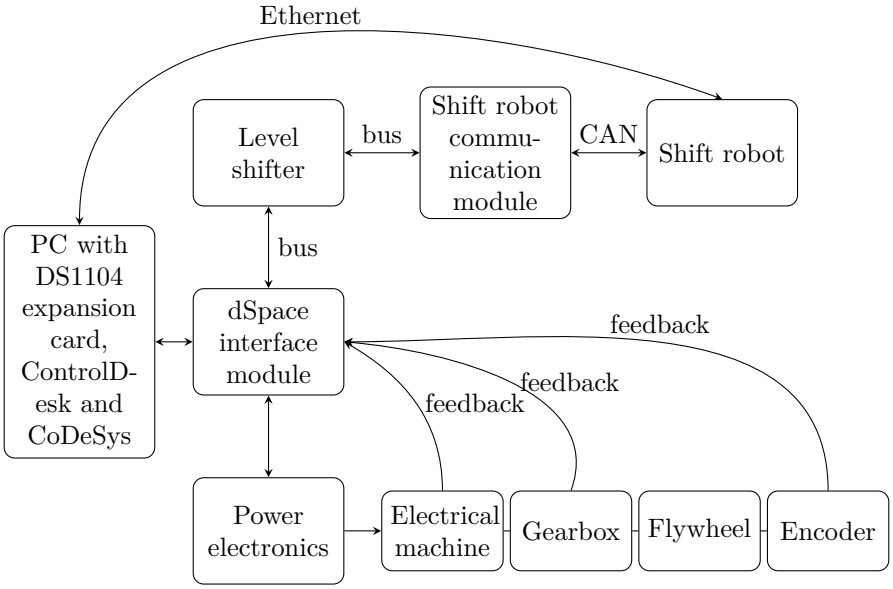


Figure 4.6: Laboratory setup block schematic

Chapter 5

Results

In this chapter the results obtained from a series of test sessions are presented. The results are submitted in chronological order, some performed using only the shift robot, some also including the traction motor to get the full dynamics of the system. In addition to these results, changes to the laboratory equipment are evaluated and discussed. This to find all the bottlenecks and minimize the shift time.

5.1 First static test session

Before the commissioning of the traction motor the performance of the shift robot alone was evaluated in a couple of static tests. These tests were run with the shift robot connected to the gearbox but without the traction motor operating.

The first of these tests aimed to roughly determine the shift robot parameters. This was accomplished by trial and error. The shift robot was repeatedly assigned to move from home position to neutral position. Every time both reference signal and actual rotor position was plotted using CoDeSys. These plots were used for evaluating the changes made in order to iteratively tune the robot. Figure 5.1 and 5.2 represents two typical plots captured during the static test sessions. These were used for evaluation as they show reference position and actual rotor position when going from home position to neutral position.

This tuning session showed that the shortest shift times were obtained by setting a reference value in the most remote region of the tolerance interval. The robot has no overshoot but a small remaining position error which makes it possible to set large reference values without worrying about the lever going to far. The remaining error is also illustrated by figure 5.1 and 5.2.

5.2 Modified shift lever

The first tests showed that the shift robot was to weak and inconsistent when doing the shifting. Binar was contacted and it turned out that they had run their tests and setup of the robot with a 100 N static load on the lever, not the 250 N stated in the specification (see section 4.1.1). To partly solve the problem

the shift lever on the robot were shortened from 30 mm to 18 mm, 18 mm being the shortest possible lever.

As can be seen in table B.1 and B.2 in appendix B, shift times were not improved drastically but a more consistent behavior could be observed. There was still a static positioning error of the lever. Unfortunately Binar was unable to remove this error. However a software update making it possible to increase the current- and voltage limit of the robot power electronics was made. Unfortunately this didn't result in any noticeable improvement in the behavior of the robot.

5.3 Position reference ramping

While running the second test it was discovered that the shift robot position reference signal had the shape of a ramp instead of a sharp step. At the robot speed of 25 000 pulses/s[1] set at the time, which was a recommended starting speed by Binar, the ramping time was quite significant. After some testing it was revealed that the ramping time was dependent on the robot speed setting. Two graphs were plotted with speed set to 10 000 pulses/s and 500 000 pulses/s respectively. They represents two extreme speed settings, clearly showing the connection between speed and ramp time. Figure 5.1 shows the reference signal and actual rotor position with the speed set to 10 000 pulses/s when changing the position reference value from 0 to 420 pulses. This represents going from home position to neutral.

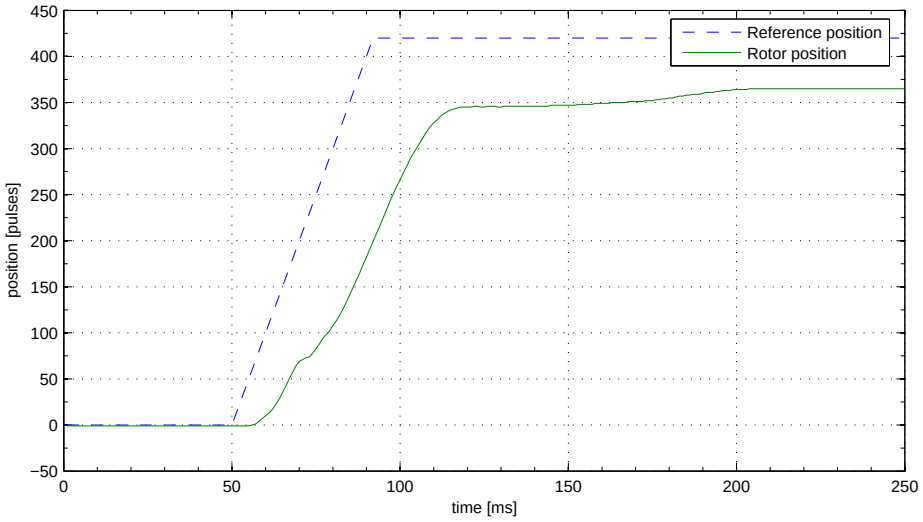


Figure 5.1: Ramping of reference signal with low robot speed (10 000 pulses/s).

The figure indicates a 40 ms ramp time which is unacceptable in comparison to the 100 ms objective for a complete shift sequence. Figure 5.2 is identical to figure 5.1 except that the speed is set to 500 000 pulses/s. This almost eliminates the reference value ramping. By comparing these two plots it is clear that the

[1] Pulses being the pulses registered by the robot position feedback encoder.

ramped reference signal is limiting the robot performance. This resulted in that all further tests were run at a speed of 300 000 pulses/s, being the highest speed giving a noticeable performance improvement.

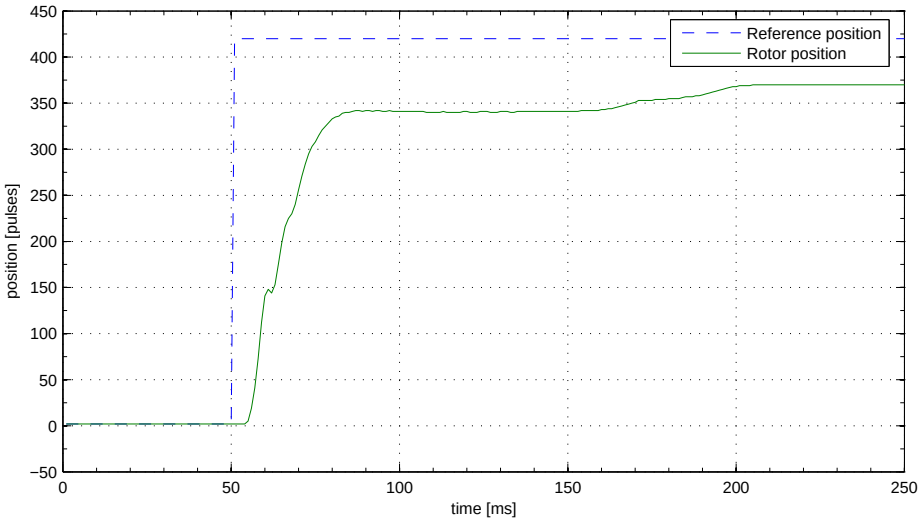


Figure 5.2: Ramping of reference signal with high robot speed (500 000 pulses/s).

A software update eliminating the reference value ramping was discussed with Binar. It was however not possible to change the ramp settings.

5.4 First dynamic test session

The next step was to commission the traction motor in the laboratory setup and evaluate the entire shift process with all dynamics accounted for. The laboratory setup was modified to include the motor and a belt drive connecting it to the gearbox. Now complete shift sequence tests were run, both including actuation of the robot and synchronization by the traction motor according to section 3.1. The test was performed while shifting between fourth and fifth gear as stated in the practical concept study discussed in section 2.2. This giving a reference of how good the shift times were.

To evaluate the time critical part of the shift sequence, the time of every step of it was measured. This was done by assigning a source code variable a new value at every step of the sequence. This variable variation can be seen in figure 5.3. The solid line represents the variable assigned value one through four at each of the sequence steps. The crosshatched signal represents the shift robot ready signal (ACK) and is set by the shift robot when ready.

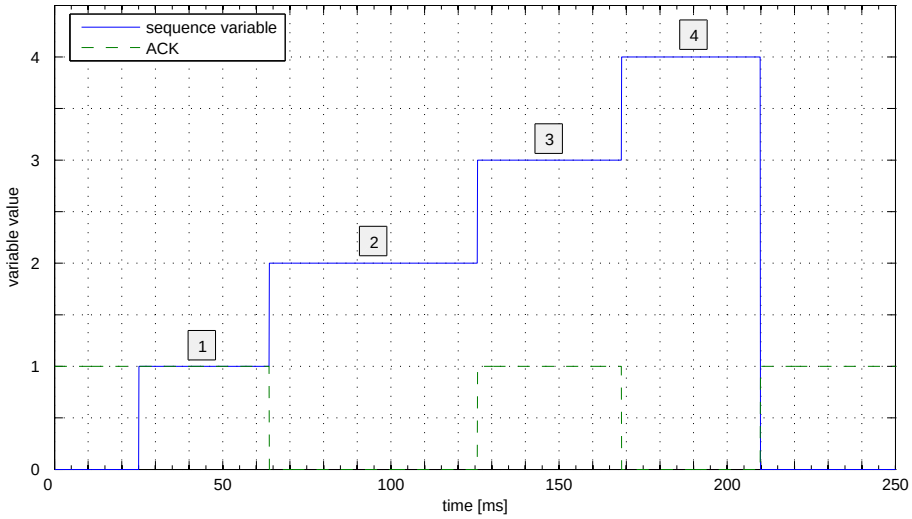


Figure 5.3: Graph captured by ControlDesk showing a complete shift cycle.

The different steps represents:

1. Traction motor torque set to zero, ∼39 ms
2. Shift robot going to the neutral step between the gears, ∼55 ms
3. Synchronization of traction motor and outgoing axle speeds, ∼43 ms
4. Shift robot engaging next gear, ∼41 ms

At the end of the sequence the variable is assigned the value zero, awaiting a new sequence. The entire shift sequence sums up to 178 ms, much higher than the aim of 100 ms.

5.5 Rotary encoder

In the original setup the synchronizing speed was calculated from a sampled value of the traction motor speed just prior to the start of the shift sequence.

This worked out well most times, as the flywheel speed was assumed to be constant during the short time interval of a gear shift. However, when the shift robot for some reason didn't manage to gear up or gear down within the usual time interval of less than 200 ms the flywheel lost a substantial amount of speed. This resulted in badly synchronized input- and output shafts of the gearbox, hence large forces acting on the gearbox when the robot ultimately geared up.

To resolve this problem a rotary encoder was fastened to the flywheel, keeping track of its speed at all times. This speed is used to recalculate the target speed continuously during shifts. So that if the robot gets stuck in between gears for a long time, the traction motor adjusts its sync speed accordingly. This modification didn't cut anything on shift times but errors could be handled in a better way.

5.6 Evaluation of communication times

The dynamic tests resulted in undesirably long shift times, longer than the static tests indicated. It was assumed that extensive communication times between the dSpace system and the robot controller were the main issue. To investigate where time was lost a series of tests were undertaken.

Once again CoDeSys plots showing rotor position as a function of time, like figure 5.1 and 5.2, were captured; this time in a dynamic event. The graphs were compared with the ones from the static tests but no difference in shift robot response could be seen. This even further strengthened the suspicion of communication delays.

Next step was to take a look at the signals at the Wago input- and output modules connecting the shift robot to the dSpace system as described in section 4.2.1. An oscilloscope was connected to the control signal (N4) and the ready signal (ACK) at the input/output modules and plots were captured by oscilloscope and CoDeSys simultaneously. The oscilloscope plot, figure 5.4, shows a 60 ms delay from the control signal positive edge to the ready signal (ACK) positive edge.

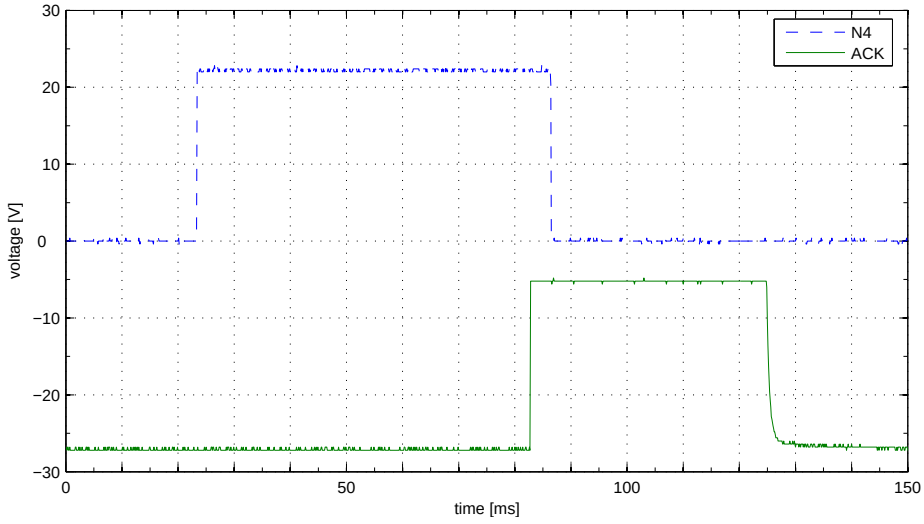


Figure 5.4: Oscilloscope graph showing elapsed time between reference signal (N4) and ready signal (ACK) with 20 ms output module poll time.

Equivalently the CoDeSys plot, figure 5.5, shows a 24 ms delay. This confirmed our suspicion of a delay in the signal transfer between the shift robot and the dSpace system.

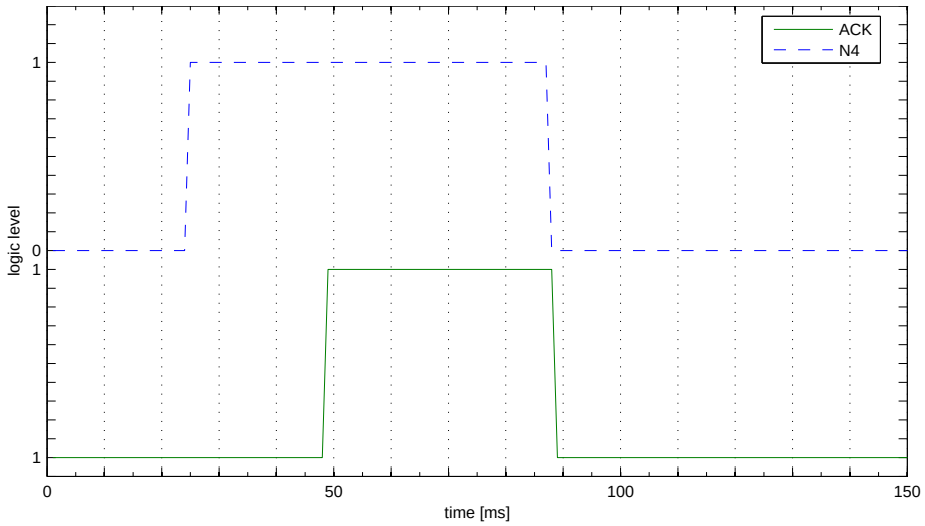


Figure 5.5: CoDeSys graph showing elapsed time between reference signal (N4) and ready signal (ACK) with 20 ms output module poll time.

The reason for this suddenly affecting the shifting was that the shifting previously had been fully independent of the traction motor, thus not affected by communication times involving the ready signal. It was found that the Wago output unit poll time of the shift robot communication module was set to 20 ms. This resulted in as much as 20 ms waiting. The poll time was decreased to 1 ms by changing a software variable which resulted in a response improvement that can be seen in figure 5.6 and 5.7.

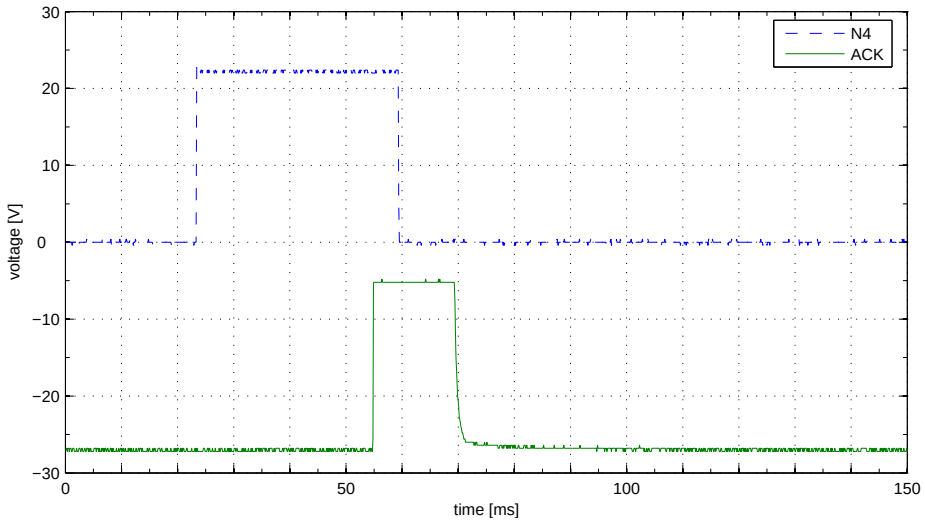


Figure 5.6: Oscilloscope graph showing elapsed time between reference signal (N4) and ready signal (ACK) with 1 ms output module poll time.

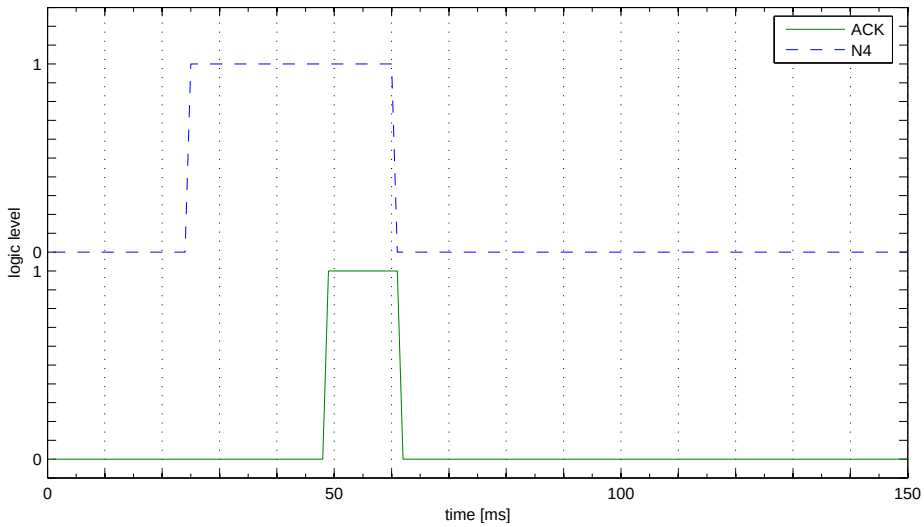


Figure 5.7: CoDeSys graph showing elapsed time between reference signal (N4) and ready signal (ACK) with 1 ms output module poll time.

Since the previous two graphs doesn't isolate the communication delays but also include the execution times of the actual robot movement, two more test were carried out to find out the real communication delays. Both input signal delay and output signal delay were investigated.

To determine the input signal delay, the control signal (N4) and one of the shift robot feedback encoder signals were monitored. The result can be seen in figure 5.8. The plot indicates a delay of 12 ms from control signal change to actual robot movement. This is a delay purely from the shift robot software and hardware.

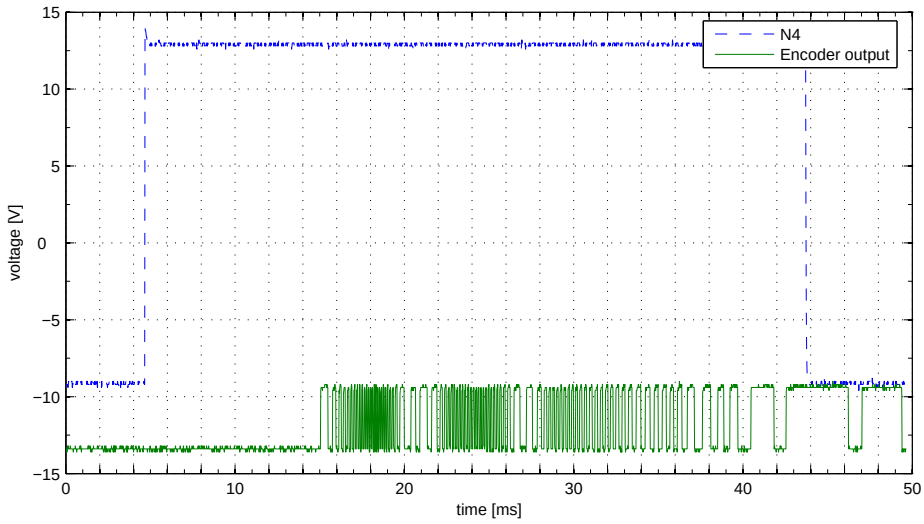


Figure 5.8: Graph from oscilloscope showing time delay between reference signal change and actual rotor movement

The output signal delay, i.e. the ready signal (ACK) delay, was investigated in the same way. That is, encoder signal and ready signal (ACK) was monitored by oscilloscope. Figure 5.9 shows the result. In order to determine the point where the ready signal (ACK) theoretically should go high, the number of pulses to the beginning of the tolerance window, in this case 280 pulses, was counted and the point marked with a vertical line in Matlab. The delay from this point to the ready signal (ACK) positive edge is 7 ms.

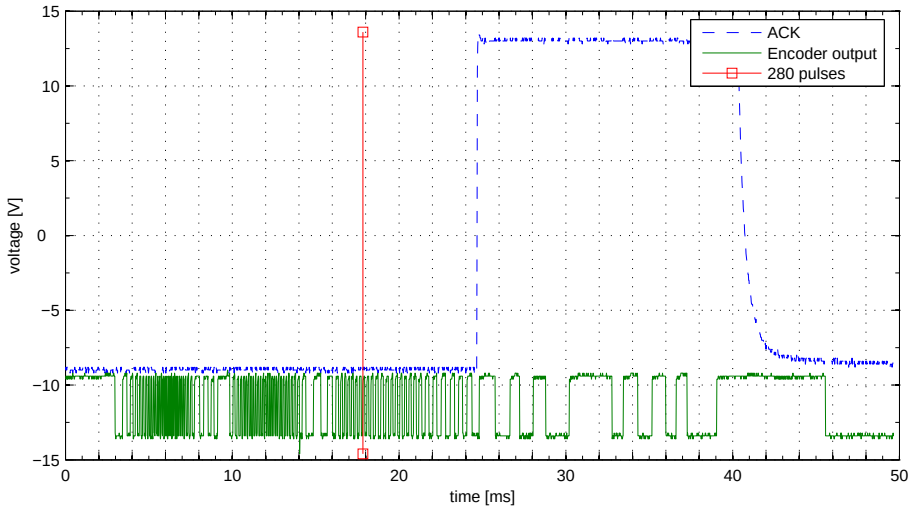
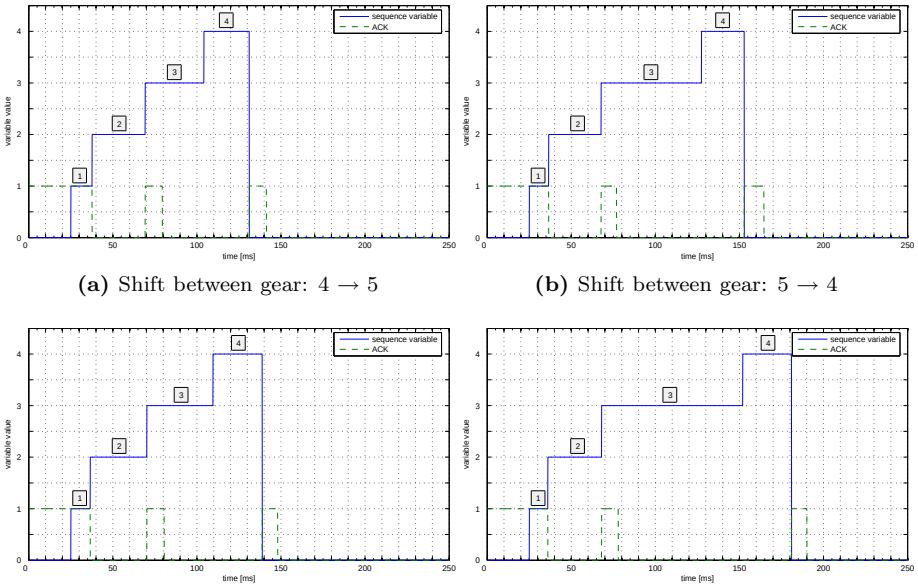


Figure 5.9: Oscilloscope graph illustrating time between setpoint reached and ready signal set

These last two tests marks the end of the communication time investigation. In this setup the delay of the control signals to the system was measured to be 12 ms. The ready signal from the system has a delay of 7 ms.

5.7 Final evaluation and results

To finally evaluate the result of the changes made, a last series of tests were performed. Tests were done shifting from fourth to fifth gear and back again and ditto from first to second, representing the greatest step according to table E.1 in appendix E. The test results are presented in figure 5.10. All tests were run with a DC link voltage of 185 V and an electrical machine speed prior the shift sequence of 1274 rpm. The solid line represents the variable assigned value one through four as the steps of the shift sequence are executed. At the end of the sequence it is assigned the value zero. The crosshatched line represents the shift robot ready signal (ACK).



(a) Shift between gear: 4 → 5

(b) Shift between gear: 5 → 4

(c) Shift between gear: 1 → 2

(d) Shift between gear: 2 → 1

Figure 5.10: ControlDesk graphs of four final shift sequences. (1) Traction motor torque set to zero, (2) Shift robot going to the neutral step between the gears, (3) Synchronization of traction motor and outgoing axle speeds, (4) Shift robot engaging next gear.

Figure 5.11 shows the plots in figure 5.10 merged into one single plot. An interesting observation is that step one (setting EM torque to 0), two (shift robot going to neutral) and four (shift robot gear up/down) are quite uniform regardless of sequence while step three (synchronizing) differs quite a lot.

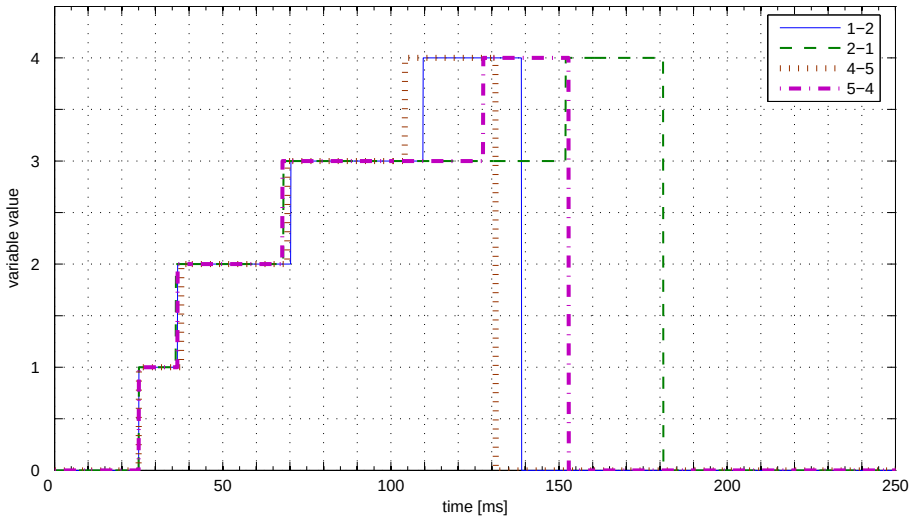


Figure 5.11: ControlDesk graph showing the above plots in one graph.

Compared to the first dynamic test session described in section 5.4 the final time consumption of each of the four steps has decreased to:

1. Traction motor torque set to zero, ∼12 ms
2. Shift robot going to the neutral step between the gears, ∼32 ms
3. Synchronization of traction motor and outgoing axle speeds, ≥40 ms
4. Shift robot engaging gear, ∼27 ms

This sums up to ≥111 ms which still is more than the goal set at 100 ms.

Chapter 6

Discussion and conclusions

A discussion of the results reached in the project is undertaken. Finally some concluding remarks are made regarding the continuation of the project.

6.1 Discussion

As has been presented in chapter 5, this project has not fully reached the goal of a shift time less than 100 ms. However looking at the results and the evaluation of the different time consuming parts of the shifting process, the conclusion that it is possible to do the shifting in a time below 100 ms is indisputable.

In the current setup there is a 12 ms delay on every input control signal to the robot and a 7 ms delay of the ready signal set by the robot when it has completed a task. In every shift sequence the robot is given two commands: goto neutral and goto next gear. This means a total of 38 ms is lost in communication time; every cycle. Parallel communication itself is not slow, but the signal conversions in the current chain makes the communication time consuming. Level shifting, signal polling by the shift robot communication module and signal conversion from parallel to CAN contribute to delays. A better solution would have been to limit the communication to a single CAN-bus.

Remarkable is that the time that it takes to set the electric motor torque to zero is significant, this should not be the case. The time lost here is probably due to the design of the source code, involving an acknowledge from the regulator controlling the electric motor.

A dedicated electric motor replacing the one used so far will be a significant improvement. This will enable much faster acceleration and deceleration times, cutting down in the most time consuming part of the shift process. However, it is important to keep in mind that the tests presented in this report have been conducted at very low speeds due to the laboratory equipment. With the new electric motor, hardware updates and a software with proper error handling, tests at more relevant speeds can be conducted. It is first then the synchronizing will be put up for a real test.

There has been no real analysis of the shift robot. There is a traction motor developed exclusively for this project, and the same should be done with the shift robot. The servo based on an asynchronous machine may not be the best solution for the task. The current robot hardware is by no means optimized

for this task, resulting in a poor shifting actuation. Sometimes the robot is not able to complete the full sequence, resulting in unacceptable long shift times. Probably due to that it is not strong enough. This report doesn't include data and plots representing these unsuccessful shift sequences since they clearly deviated from the normal outcome.

6.2 Concluding remarks

The continuation of the project should involve a complete redesign of the shift robot, the control electronics and preferably the gear box. This to ensure a robust and reliable system, able of running high speed tests with the dedicated electrical machine and finally operating in a prototype hybrid vehicle. The most important thing should be safety when the design is carried out, both mechanically and software oriented. Properly designed software with a failsafe error handling is essential.

Bibliography

[1] Saab automotive, September 2009. http://www.saab.com.

[2] 3S-Smart Software Solutions GmbH. *CoDeSys*, June 2009. http://www.3s-software.com.

[3] Binar Elektronik AB. Website, September 2009. http://www.binarelektronik.se.

[4] Mats Alaküla. Personal conversation, 2008. Lund.

[5] Per Bandrup and Joakin Larsson. Electrically synchronized gear shifting. Master's thesis, Department of Industrial Electrical Engineering and Automation, Lund University, Lund, 2008.

[6] dSpace. *DS1104 R&D Controller Board, Installation and Configuration Guide*, 3.4 edition, May 2002.

[7] dSpace Inc. Website, April 1996. http://www.dspaceinc.com.

[8] dSpace Inc. Website, September 2009. http://www.dspaceinc.com/ww/en/inc/home/products/hw/singbord/ds1104.cfm.

[9] dSpace Inc. *ControlDesk*, June 2009. http://www.dspaceinc.com/ww/en/inc/home/products/sw/expsoft/contrdes.cfm?nv=bbp.

[10] Finnis Leen and Orlando Sanchez. Pre-study of an electrically synchronized sequential manual transmission. Master's thesis, Division of Automotive Engineering, Chalmers University of Technology, Gothenburg, 2006.

[11] Lönne. *Katalog Svensk Elektromotor 2007*, January 2007. http://www.lonne.com/bilder/filer/katalogersvensk/KatalogSvenskElektromotor2007.pdf.

[12] Yury Loayza. *Development of a cost effective drive train for hybrid vehicles*. PhD thesis, Department of Industrial Electrical Engineering and Automation, Lund University, Faculty of Engineering, Lund, Preliminary completed 2010.

[13] NXP. *Octal buffer/line driver*, August 2007. http://www.nxp.com/acrobat_download/datasheets/74HC_HCT240_3.pdf.

[14] WAGO Kontakttechnik GmbH & Co. KG. *4-Channel Digital Input Module DC 24 V*, September 2008. http://www.wago.com/wagoweb/documentation/750/eng_dat/d040200e.pdf.

[15] WAGO Kontakttechnik GmbH & Co. KG. *8-Channel Digital Output Module DC 24 V*, September 2008. `http://www.wago.com/wagoweb/documentation/750/eng_dat/d053000e.pdf`.

[16] Xtrack. Website, April 2008. `http://www.xtrack.com`.

Appendix A

Level shifter

A.1 Layout

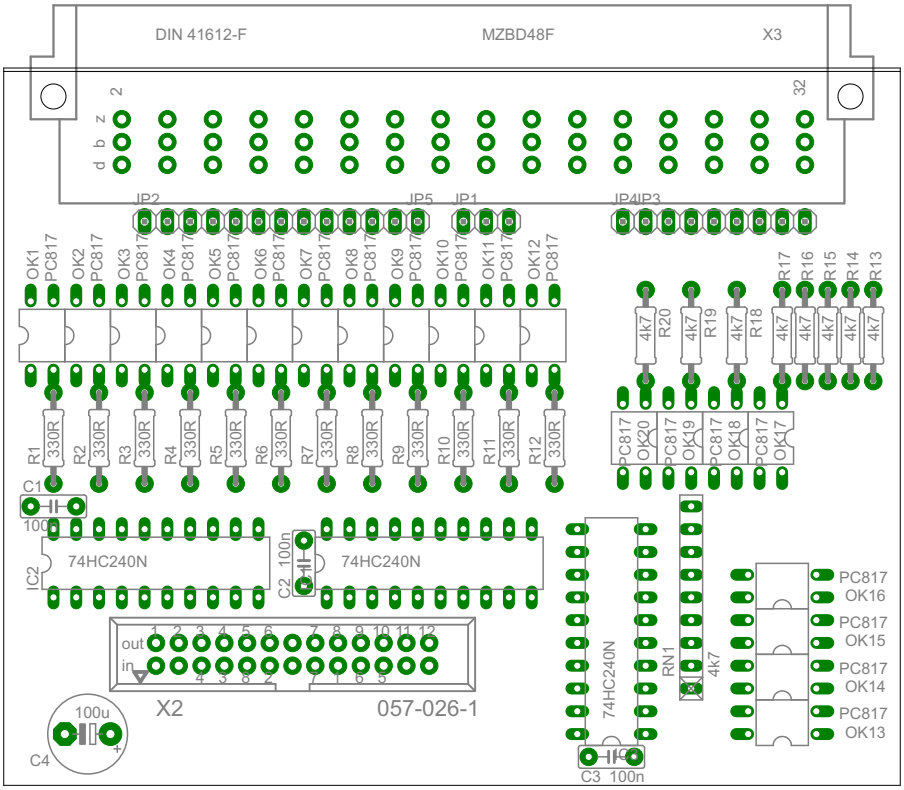


Figure A.1: Interface board layout.

A.2 Schematic

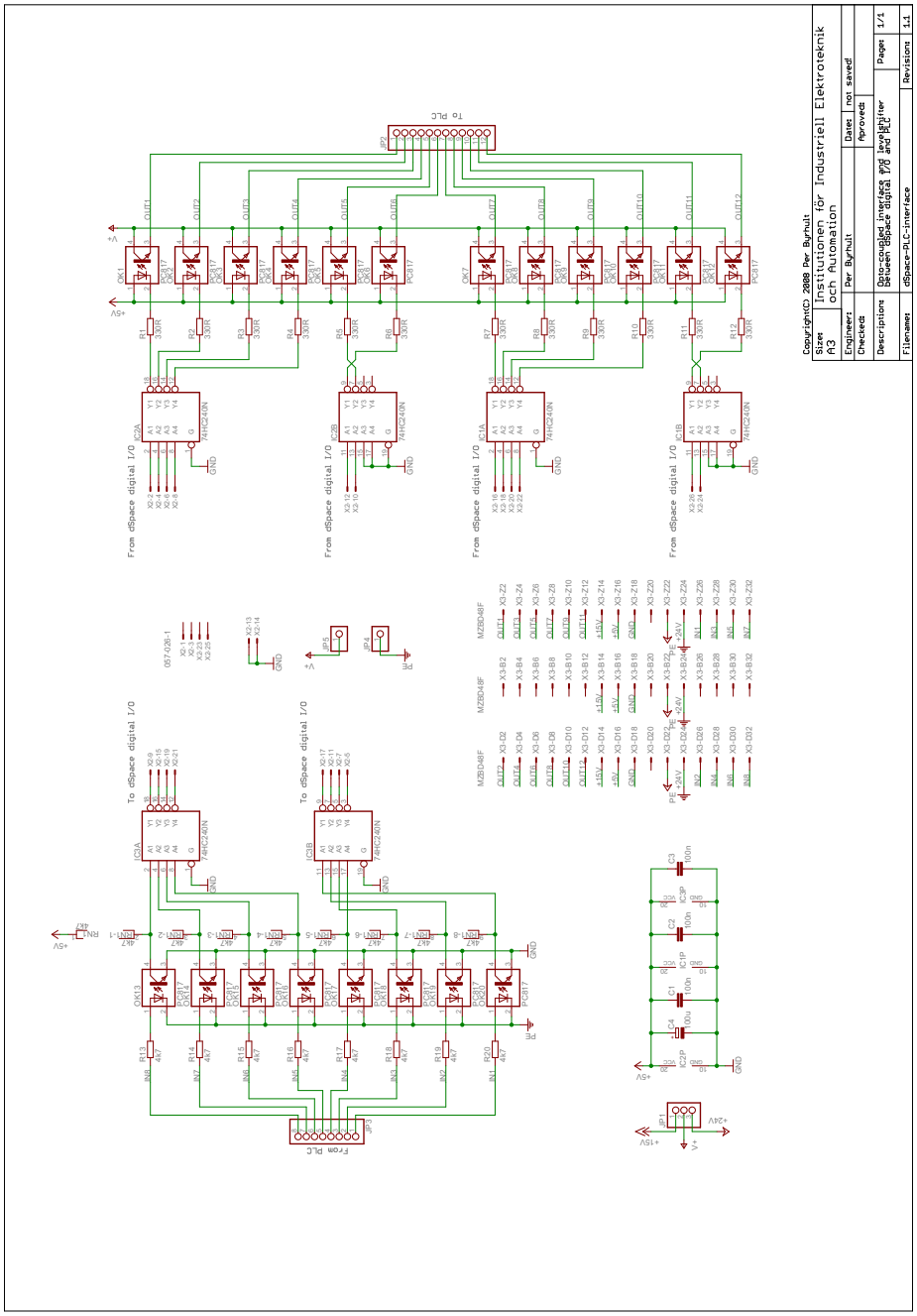


Figure A.2: Interface board schematic.

Appendix B

Shift times

Table B.1: Measured shift times with long lever (30 mm).

Mode	**Speed**	**Position [pulses]**	**Tolerances [pulses]**	**Measured interval [pulses]**	**Measured time [ms]**
N12	25000	0 - 300	-50, +0	0 - 170	36
N12	25000	0 - 300	-50, +0	0 - 170	33
N12	25000	0 - 300	-50, +0	0 - 170	31
N12	25000	0 - 300	-50, +0	0 - 170	36
N12	50000	0 - 300	-50, +0	0 - 170	30
N12	50000	0 - 300	-50, +0	0 - 170	31
N12	50000	0 - 300	-50, +0	0 - 170	31
N12	50000	0 - 300	-50, +0	0 - 170	31
N12	100000	0 - 300	-50, +0	0 - 170	28
N12	100000	0 - 300	-50, +0	0 - 170	29
N12	100000	0 - 300	-50, +0	0 - 170	27
N12	100000	0 - 300	-50, +0	0 - 170	27
N12	300000	0 - 300	-100, +0	0 - 170	20
N12	300000	0 - 300	-100, +0	0 - 170	22
N12	300000	0 - 300	-100, +0	0 - 170	19
N12	300000	0 - 300	-100, +0	0 - 170	20

Table B.2: Measured shift times with short lever (18 mm).

Mode	Speed	Position [pulses]	Tolerances [pulses]	Measured interval [pulses]	Measured time [ms]
N12	25000	0 - 350	-50, +50	0 - 280	33
N12	25000	0 - 350	-50, +50	0 - 280	33
N12	25000	0 - 350	-50, +50	0 - 280	32
N12	25000	0 - 350	-50, +50	0 - 280	32
N12	25000	0 - 350	-50, +50	0 - 280	36
N12	25000	0 - 350	-50, +50	0 - 280	34
N12	25000	0 - 350	-50, +50	0 - 280	33
N12	25000	0 - 350	-50, +50	0 - 280	33
N12	50000	0 - 350	-50, +50	0 - 280	28
N12	50000	0 - 350	-50, +50	0 - 280	27
N12	50000	0 - 350	-50, +50	0 - 280	27
N12	50000	0 - 350	-50, +50	0 - 280	27
N12	100000	0 - 350	-50, +50	0 - 280	26
N12	100000	0 - 350	-50, +50	0 - 280	26
N12	100000	0 - 350	-50, +50	0 - 280	26
N12	100000	0 - 350	-50, +50	0 - 280	25
N12	150000	0 - 350	-50, +50	0 - 280	25
N12	150000	0 - 350	-50, +50	0 - 280	26
N12	150000	0 - 350	-50, +50	0 - 280	25
N12	150000	0 - 350	-50, +50	0 - 280	27
N12	200000	0 - 350	-50, +50	0 - 280	26
N12	200000	0 - 350	-50, +50	0 - 280	25
N12	200000	0 - 350	-50, +50	0 - 280	25
N12	200000	0 - 350	-50, +50	0 - 280	26
N12	300000	0 - 350	-50, +50	0 - 280	26
N12	300000	0 - 350	-50, +50	0 - 280	26
N12	300000	0 - 350	-50, +50	0 - 280	26
N12	300000	0 - 350	-50, +50	0 - 280	25
N12	300000	0 - 420	-140, +0	0 - 280	18
N12	300000	0 - 420	-140, +0	0 - 280	19
N12	300000	0 - 420	-140, +0	0 - 280	18
N12	300000	0 - 420	-140, +0	0 - 280	19
N12	500000	0 - 420	-140, +0	0 - 280	18
N12	500000	0 - 420	-140, +0	0 - 280	18
N12	500000	0 - 420	-140, +0	0 - 280	18
N12	500000	0 - 420	-140, +0	0 - 280	18
N12 - MAX	300000	0 - 650	-100, +50	0 - 550	16
N12 - MAX	300000	0 - 650	-100, +50	0 - 550	16
N12 - MAX	300000	0 - 650	-100, +50	0 - 550	16
N12 - MAX	300000	0 - 650	-100, +50	0 - 550	18

Table B.3: Measured shift times with short lever (18 mm) and increaced current limit.

Mode	**Speed**	**Position [pulses]**	**Tolerances [pulses]**	**Measured interval [pulses]**	**Measured time [ms]**
N12	300000	0 - 420	-140, +0	0 - 280	18
N12	300000	0 - 420	-140, +0	0 - 280	18
N12	300000	0 - 420	-140, +0	0 - 280	18
N12	300000	0 - 420	-140, +0	0 - 280	20
N12	300000	0 - 420	-140, +0	0 - 280	16

Appendix C

Gear lever travel

Table C.1: Measured shift lever travels. Session 1.

Gear	**Start pos. A1 [mm]**	**Neutral begin B1 [mm]**	**Neutral end B2 [mm]**	**End pos. A2 [mm]**	**Delta A [mm]**	**Delta B [mm]**	**Center pos. N [mm]**
6 → 5	254.50	258.20	260.50	264.20	9.70	2.30	4.85
5 → 4	254.90	258.20	261.20	264.10	9.20	3.00	4.80
4 → 3	255.00	258.20	260.70	264.10	9.10	2.50	4.45
3 → 2	255.00	258.00	261.10	264.00	9.00	3.10	4.55
2 → 1	255.10	258.00	260.60	264.20	9.10	2.60	4.20
1 → 2	255.00	251.00	248.50	245.20	-9.80	-2.50	-5.25
2 → 3	254.20	251.40	248.20	245.20	-9.00	-3.20	-4.40
3 → 4	254.20	251.00	248.60	245.20	-9.00	-2.40	-4.40
4 → 5	254.20	251.30	248.20	245.20	-9.00	-3.10	-4.45
5 → 6	254.20	250.60	248.70	245.30	-8.90	-1.90	-4.55

Table C.2: Measured shift lever travels. Session 2.

Gear	**Start pos. A1 [mm]**	**Neutral begin B1 [mm]**	**Neutral end B2 [mm]**	**End pos. A2 [mm]**	**Delta A [mm]**	**Delta B [mm]**	**Center pos. N [mm]**
6 → 5	254.40	258.20	260.40	264.20	9.80	2.20	4.90
5 → 4	255.00	257.90	261.20	264.00	9.00	3.30	4.55
4 → 3	254.80	258.20	260.80	264.10	9.30	2.60	4.70
3 → 2	254.50	258.00	261.40	264.10	9.60	3.40	5.20
2 → 1	254.50	258.20	260.70	264.10	9.60	2.50	4.95
1 → 2	254.70	250.90	248.60	245.20	-9.50	-2.30	-4.95
2 → 3	254.50	251.30	248.00	245.20	-9.30	-3.30	-4.85
3 → 4	254.60	251.00	248.60	245.20	-9.40	-2.40	-4.80
4 → 5	254.40	251.40	248.20	245.20	-9.20	-3.20	-4.60
5 → 6	254.50	250.60	248.50	245.30	-9.20	-2.10	-4.95

Appendix D

Cables and connectors

Table D.1: Table of the digital I/O cable colors and connector pins.

Digital I/O	dSpace	DSUB	Cable	Signal
1	0	1	purple	N12
2	1	14	brown	N23
3	2	2	pink	N34
4	3	15	blue	N45
5	4	3	gray	N56
6	5	16	green	DIR
7	6	4	white	MAX
8	7	17	yellow	HOME
24V		7	red + red-white	
Power earth		20	black + black-white	
9	12	8	red-purple	ACK

Appendix E

Transmission gear ratios

Table E.1: Table of the transmission gear ratios

Gear	Ratio
1st	9.875:1
2nd	6.750:1
3rd	5.625:1
4th	4.750:1
5th	3.625:1
6th	4.125:1

Appendix F

Source code

```
/*************************************************************************
*
* FILE:
*    generator.c
*
* RELATED FILES:
*    Brtenv.h
*
*
*************************************************************************/

#include <brtenv.h>                          /* basic real-time environment */
#include <math.h>
#include <stdio.h>

/*-----------------------------------------------------------------------*/

#define PI          3.141592654
#define sqrt_two    1.414213562
#define sqrt_three  1.732050808
#define sqrt_six    2.449489742783l8

#define MAX_ARR_SIZE 50
#define TRUE  1
#define FALSE 0


/* variables for communication with Slave DSP */
Int16 task_id = 0;                                  /* communication channel */
Int16 index = -1;                                  /* slave DSP command index */

Int16 ch1_index = -1; /* slave DSP command index for ch. 1 */
Int16 ch2_index = -1; /* slave DSP command index for ch. 2 */
Int16 ch3_index = -1; /* slave DSP command index for ch. 3 */
Int16 ch4_index = -1; /* slave DSP command index for ch. 4 */

/* parameters for 3-phase PWM initialization */
Float64 Ts = 1e-4;
Float64 deadband = 0.0;          /* deadband period */
Float64 sync_pos = 0.5;          /* position of the synch. interrupt signal */

/* parameters for 4 channel PWM initialization */
UInt16 mode = SLVDSP1104_PWM_MODE_SYM;       /* PWM mode */
UInt16 pol = SLVDSP1104_PWM_POL_HIGH;        /* PWM polarity */

/* parameters accessed by ControlDesk */
//??? volatile Float64 enable=0;
//??? volatile Float64 sw=0;
volatile Float64 duty3_a=0;
volatile Float64 duty3_b=0;
volatile Float64 duty3_c=0;
/*volatile*/ Float64 AD1_adj=1;
/*volatile*/ Float64 w_adj=0.7;
/*volatile*/ Float64 AD3_adj=1;
/*volatile*/ Float64 Udc_adj=600;//Udc_adj=650;
//???             Float64 ia_adj=50; // conny
//???             Float64 ib_adj=50;
/*volatile*/     Float64 ia_adj=400; // ny omriktare 20090507
/*volatile*/     Float64 ib_adj=400;

/*volatile*/ Float64 Theta_adj=0;
             Float64 AD8_adj=1;
             Float64 DA1;

             Float64 duty1_a=0;
             Float64 duty1_b=0;
             Float64 duty1_c=0;

             Float64 Udc = 0;
```

```
volatile     Float64 w_set = 0;
             Float64 w = 0;
volatile     Float64 isx_set = 0;
             Float64 isx = 0;
volatile     Float64 isy_set = 0;
             Float64 isy = 0;

             Float64 isx_temp = 0;
             Float64 isy_temp = 0;

/*volatile*/ Float64 ia = 0;
/*volatile*/ Float64 ib = 0;
/*volatile*/ Float64 ic = 0;

/*volatile*/ Float64 ia_u_raw = 0; // signal for 'ia' from dspace in [volt]
/*volatile*/ Float64 ib_u_raw = 0;

volatile Float64 ia_u_off = 0.0010; //0.0018; // 0.0117; // offset signal for 'ia'
     from dspace in [volt]
volatile Float64 ib_u_off = 0.0011; //0.0109; //; //; // 0.0017;

Float64 ia_u_filter = 0; // offset signal for 'ia' from dspace in [volt]
Float64 ib_u_filter = 0;



/*volatile*/ Float64 cosTheta = 1;
/*volatile*/ Float64 sinTheta = 0;
/*volatile*/ Float64 Theta = 0;
volatile Float64 usx = 0;
volatile Float64 usy = 0;
const Float64 usxymax = 1/sqrt_two; // power elctronics book, M.A. P.K page 65
     figure 2.20


volatile Float64 Imax = 30;
Float64 omega = 0;
Float64 theta_old = 0;
volatile Float64 delta_wEl_Max = 10;


Float64 omegaFiltred = 0;
Float64 torque = 0;

Float64 w_analog = 0 ;

/*-------------------------------------------------------------------------*/
/*-- BEGINNING OF VARIABLES BY PER & AXEL ---------------------------------*/
/*-------------------------------------------------------------------------*/

const UInt32 TIME_OUT1 = 30000;                          //Shift robot goto N time out
     constant
const UInt32 TIME_OUT2 = 30000;                          //Shift robot goto MAX time out
     constant

UInt32 time_out1 = 0;                                    //time out counter
UInt32 time_out2 = 0;                                    //time out counter

const Float64 gear_position_table[5] = {0.37, 0.49, 0.62, 0.74, 0.87};
     //Table of gear position sensor readings
const Float64 gear_ratio_table[6] = {9.875, 6.750, 5.625, 4.750, 3.625, 4.125};
     //Measured values {9+7/8 6+6/8 5+5/8 4+6/8 3+5/8 4+1/8}

volatile UInt32 DI = 0;                                  //Digital input port
volatile UInt32 ACK = 0;                                 //Shift robot acknowledge signal
volatile UInt32 DIR = 0;                                 //Shift direction, (up
     shift/down shift)
volatile UInt32 shift_sequence = 0;                      //Current shift sequence state

volatile UInt32 up_shift_button_flag = 0;                //Up shift button flag
volatile UInt32 up_shift_button = 0;                     //Up shift interface button
volatile UInt32 last_up_shift_button = 0;                //Up shift button last state
volatile UInt32 down_shift_button_flag = 0;              //Down shift button flag
volatile UInt32 down_shift_button = 0;                   //Down shift interface button
volatile UInt32 last_down_shift_button = 0;              //Down shift button last state

volatile UInt32 HOME = 0;
volatile UInt32 MAX = 0;
volatile UInt32 N12 = 0;
volatile UInt32 N23 = 0;
volatile UInt32 N34 = 0;
volatile UInt32 N45 = 0;
volatile UInt32 N56 = 0;

volatile UInt32 current_gear = 0;                        //Currently active gear
volatile UInt32 next_gear = 0;                           //Next gear to switch to

/*volatile*/ Float64 reference_speed = 0;                //Reference speed of EM

/*volatile*/ UInt32 acc = 1;                             //Acceleration flag
/*volatile*/ Float64 delta_w_acc = 0.00015;               //Acceleration step size
     constant
/*volatile*/ Float64 delta_w_dec = 0.00015;               //Decelaration step size
     constant
UInt32 auto_up_shift = 0;                                //Automatic up shift flag
UInt32 auto_down_shift = 0;                              //Automatic down shift flag
/*volatile*/ Float64 shift_speed = 400;                  //Autoamtic up shift speed
///*volatile*/ Float64 down_shift_speed = 450;            //Automatic down shift speed
```

```
volatile Float64 encoder_counter = 0;                //Counts number of encoder
    flancs
/*volatile*/ Float64 w_wheel = 0;                    //Wheel speed (rad/sec)
/*volatile*/ Float64 temp_w_wheel = 0;               //
/*volatile*/ Float64 filtered_w_wheel = 0;
/*volatile*/ Float64 interrupt_counter = 0;
volatile Float64 w_set_view = 0;
volatile Float64 w_view = 0;
volatile Float64 w_wheel_view = 0;

volatile UInt32 button = 0;

volatile Float64 GOSTA = 0;

volatile UInt32 ENC2 = 0;
volatile UInt32 ENC1 = 0;
volatile UInt32 last_ENC1 = 0;

/*-------------------------------------------------------------------------*/
/*-- END OF VARIABLES BY PER & AXEL -------------------------------------*/
/*-------------------------------------------------------------------------*/


/*Variabler*/
Float64 AD1, AD2, AD3, AD4, AD5, AD6, AD7, AD8;      //AD1 reads the current active
    gear
Float64 stopp ;
UInt16 scantable[4] = {1, 2, 3, 4};
Float64 u[4];
Float64 w_temp = 0;



// Float64 L_sx = 0.85e-3; // mätningar på stator induktans 20090324
// Float64 L_sy = 0.85e-3; //
volatile Float64 L_sx = 1.6e-3; // 20090328 gradient på ström stigning delta_isx =
    4A under Ts=1e-4 vid U_L=64V
volatile Float64 L_sy = 1.6e-3; //2.6e-3; // 20090409 approx
// Float64 Rs = 0.2;          // mätningar på stator resistans med multimeter
    20090324 r_ab=r_bc=r_ca=0.4 ohm
Float64 Rs = 0.19881;          // 20090326 gradient på kurvan usx vs isx : usx =
    0.19881* isx + 7.5077

Float64 Psi_m = 0.1504; // [V*s] 20090829 // 0.15122; //0.14871; // beräknad Psi_m
    från tomgångs mätningar av usy-w 20090324 // 0.5188;// 0.1588

Float64 J_el_machine = 2.34; //  20090929 calculated from the change in w when isy
    10A is applied



Float64 view_isx_2_max = 0;
Float64 view_isx_2_min = 0;
Float64 view_usx_L_12_max = 0;
Float64 view_usx_L_12_min = 0;


Float64 view_usx_R_12 = 0;
Float64 view_usx_L_12 = 0;
Float64 view_usx_e_12 = 0;
Float64 view_usx_T_12 = 0;

Float64    view_00 = 0;
Float64    view_01 = 0;


Float64 total_time              =  0;
volatile Float64 test_omega     = 50;
volatile Float64 test_amplitude  = 0;
volatile Float64 test_dc        = 0;
Float64 test_torque_ref         = 0;


Float64 i_alpha = 0;
Float64 i_beta = 0;
Float64 Kt = 0.02;
// Float64 pol_adj = 1; // defined in function calcTheta
Float64 rpm_axel = 0;   // axel till växellåda
volatile Float64 exec_time;
Float64 u_a_set, u_b_set, u_c_set, u_alpha_set, u_beta_set;

Float64 AD1_filtred = 0;

Float64 delta_theta = 0;
static Float64 omega_temp=0;
static Float64 AD1_temp=0;

const Float64 Npp = 3;                  // polpar i motorn
const Float64 R_am = 2;                 // ratio motor axel till växellådaxel

// ensure that moment is low to be set the gearbox to neutral
const Float64 epsTorque = 1;
const Float64 psi_s_x = 1;              // ??? preliminär
const Float64 psi_s_y = 1;

// control mode variables
const UInt32 ControlModeSpeed   = 0;
const UInt32 ControlModeTorque  = 1;
```

```
volatile UInt32 controlMode = 0;

//
#include "i_w_control.c"


/*-------------------------------------------------------------------------*/
/*-- BEGINNING OF FUNCTIONS BY PER AND AXEL ---------------------------*/
/*-------------------------------------------------------------------------*/

void initShiftSystem() {
    /* Digital I/O initiation */
    ds1104_bit_io_init(DS1104_DIO11_OUT | DS1104_DIO0_OUT |
                       DS1104_DIO1_OUT | DS1104_DIO2_OUT |
                       DS1104_DIO3_OUT | DS1104_DIO4_OUT |
                       DS1104_DIO5_OUT | DS1104_DIO6_OUT |
                       DS1104_DIO7_OUT | DS1104_DIO12_IN |
                       DS1104_DIO18_IN | DS1104_DIO19_IN);      ///bit_IO channel
                            11,0,1,2,3,4,5,6,7   out
                                                                //        channel 12
                                                                    in

    ds1104_bit_io_set(DS1104_DIO7);                             //Set HOME pin
    HOME = 1;
}

void updateWheelSpeed() {
    if(interrupt_counter > 200) {
        w_wheel = encoder_counter*PI/10;
        filtered_w_wheel = w_wheel*0.10 + 0.90*temp_w_wheel ;           //filtered
             speed
        temp_w_wheel = filtered_w_wheel;
        encoder_counter = 0;
        interrupt_counter = 0;
    }
    interrupt_counter = interrupt_counter + 1;
}

Float64 calcRPM( Float64 rad_per_sec ) {
    Float64 rpm;
    rpm = rad_per_sec / (2 * PI) * 60;
    return rpm;
}

/* Returns the current gear [1..6] */
UInt32 currentGear(){
    UInt32 x = 1;
    if(AD1 <= gear_position_table[0]){                          //Check if current
         gear = 1
        x = 1;
    }else if(AD1 <= gear_position_table[1]){                    //Check if current
         gear = 2
        x = 2;
    }else if(AD1 <= gear_position_table[2]){                    //Check if current
         gear = 3
        x = 3;
    }else if(AD1 <= gear_position_table[3]){                    //Check if current
         gear = 4
        x = 4;
    }else if(AD1 <= gear_position_table[4]){                    //Check if current
         gear = 5
        x = 5;
    }else{                                                      //Current gear = 6
        x = 6;
    }
    return x;                                                   //Return the
         current gear
}

/* Returns the traction motor reference speed */
Float64 referenceSpeed(Float64 current_speed, UInt32 gear, UInt32 f_dir){
    if(f_dir == 1){                             //Up shift, dependent on
         current/next gear and the ratio in between
        switch(gear){
            case 1:
                reference_speed = current_speed * gear_ratio_table[1];
                break;
            case 2:
                reference_speed = current_speed * gear_ratio_table[2];
                break;
            case 3:
                reference_speed = current_speed * gear_ratio_table[3];
                break;
            case 4:
                reference_speed = current_speed * gear_ratio_table[4];
                break;
            case 5:
                reference_speed = current_speed * gear_ratio_table[5];
                break;
        }
    }else if(f_dir == 0){                       //Down shift, dependent on
         current/next gear and the ratio in between
        switch(gear){
            case 2:
                reference_speed = current_speed * gear_ratio_table[0];
                break;
            case 3:
                reference_speed = current_speed * gear_ratio_table[1];
                break;
```

```
            case 4:
                reference_speed = current_speed * gear_ratio_table[2];
                break;
            case 5:
                reference_speed = current_speed * gear_ratio_table[3];
                break;
            case 6:
                reference_speed = current_speed * gear_ratio_table[4];
                break;
        }
    }
    return reference_speed;
}

/* Shift robot commander */
void shiftRobotCom(UInt32 c_gear, UInt32 n_gear){
    ds1104_bit_io_clear(DS1104_DIO7);                   //Clear HOME pin
    HOME = 0;
    if((c_gear == 1) || (n_gear == 1)){                 //If 1->2 or 2->1
        ds1104_bit_io_set(DS1104_DIO0);                 //Goto neutral 1-2
        N12 = 1;
    }else if((c_gear == 2) || (n_gear == 2)){           //If 2->3 or 3->2
        ds1104_bit_io_set(DS1104_DIO1);                 //Goto neutral 2-3
        N23 = 1;
    }else if((c_gear == 3) || (n_gear == 3)){           //If 3->4 or 4->3
        ds1104_bit_io_set(DS1104_DIO2);                 //Goto neutral 3-4
        N34 = 1;
    }else if((c_gear == 4) || (n_gear == 4)){           //If 4->5 or 5->4
        ds1104_bit_io_set(DS1104_DIO3);                 //Goto neutral 4-5
        N45 = 1;
    }else if((c_gear == 5) || (n_gear == 5)){           //If 5->6 or 6->5
        ds1104_bit_io_set(DS1104_DIO4);                 //Goto neutral 5-6
        N56 = 1;
    }
}

/* Automated shift loop */
void shiftLoop() {
    Float64 w_down_shift=0;
    switch(acc) {
        case 1:                                 //accelerate
            w_set = w_set + delta_w_acc * gear_ratio_table[currentGear() - 1];
            if(fabs(w_set) > shift_speed) {
                if(currentGear() < 5) {
                    auto_up_shift = 1;
                    acc = 2;
                } else {
                    acc = 3;
                }
            }
        break;

        case 2:                                 //upshift
            auto_up_shift = 0;
            if(!shift_sequence) {
                acc = 1;
            }
        break;

        case 3:                                 //decelerate
            w_set = w_set - delta_w_dec * gear_ratio_table[currentGear() - 1];

            if (currentGear() > 1) {
                w_down_shift = shift_speed * gear_ratio_table[currentGear() -
                    1]/gear_ratio_table[currentGear() - 2];
            } else {
                w_down_shift = 100;
            }

//???           if(fabs(w_set) < (down_shift_speed - (8 - currentGear()) *
     down_shift_speed/11)) {
            if(fabs(w_set) <  w_down_shift ) {
                if(currentGear() > 1) {
                    auto_down_shift = 1;
                    acc = 4;
                } else {
                    acc = 1;
                }
            }
        break;

        case 4:                                 //downshift
            auto_down_shift = 0;
            if(!shift_sequence) {
                acc = 3;
            }
        break;
    }
}
/*-------------------------------------------------------------------------*/
/*-- END OF FUNCTIONS BY PER AND AXEL -------------------------------------*/
/*-------------------------------------------------------------------------*/


/* interrupt service routine for external interrupt */
void ext_int_3( void ) {
    //encoder_counter = encoder_counter + 1;
    ENC2 = ((ds1104_bit_io_read() & 0x40000) != 0);
    ENC1 = ((ds1104_bit_io_read() & 0x80000) != 0);
```

```
    if(ENC1 ^ ENC2) {
        encoder_counter = encoder_counter + 1;
    } else {
        encoder_counter = encoder_counter - 1;
    }
}


/* interrupt service routine for PWM sync interrupt */
void PWM_sync_interrupt( void ) {

    int counter=0;
    static int initiated= FALSE;
    static Float64 arr_time  [MAX_ARR_SIZE];
    static Float64 arr_theta [MAX_ARR_SIZE];
    static Float64 arr_w_lin [MAX_ARR_SIZE];
    Float64 arr_theta_cont[MAX_ARR_SIZE];
    Float64 theta_raw = 0;
    Float64 out_coef_arr[2]={0,0}; // {a,b} contains the coeffs 'a' and 'b' in
        y=a*x+b
    Float64 w_lin = 0;

    const Float64 w_filter_needed = 150; // for w < 100 a stronger filter is needed
    Float64 filter = 100;

    //RTLIB_TIC_START();                              /* start time measurement */

    if (initiated == FALSE) {
        for(counter=0; counter<MAX_ARR_SIZE; counter++) {
            arr_time        [counter]=counter;
            arr_theta       [counter]=0;
            arr_theta_cont  [counter]=0;
            arr_w_lin       [counter]=0;
        }
        initiated=TRUE;
    }
    // ----

    // ----




    host_service(1, 0);                             /* Data Acquisition service */

    /*-------------------------------------------------------------------------*/
    /*-- ISR ROUTINES BY PER AND AXEL BEGINS ----------------------------------*/
    /*-------------------------------------------------------------------------*/

    DI = ds1104_bit_io_read();

    ACK = ((DI & 0x1000) != 0);

    updateWheelSpeed();

    if(w_set > 100 && (shift_sequence == 0) && ACK) {
        if(up_shift_button_flag || auto_up_shift){
            up_shift_button_flag = 0;
            current_gear = currentGear();
            if(current_gear == 6){
                //ABORT
            }else{
                RTLIB_TIC_START();                          /* start time
                    measurement */
                setControlMode(ControlModeTorque);
                set_torque(0);                                          //Set motor
                    torque to zero, M=0
                ds1104_bit_io_set(DS1104_DIO5);                         //Set DIR
                    pin (Up shift)
                DIR=1;
                ds1104_bit_io_clear(DS1104_DIO7);                       //Clear
                    HOME pin
                HOME = 0;
                next_gear = current_gear + 1;
                shift_sequence = 1;
            }
        }else if(down_shift_button_flag || auto_down_shift){
            down_shift_button_flag = 0;
            current_gear = currentGear();
            if(current_gear == 1){
                //ABORT
            }else{
                RTLIB_TIC_START();                          /* start time
                    measurement */
                setControlMode(ControlModeTorque);
                set_torque(0);                                          //Set motor
                    torque to zero, M=0
                ds1104_bit_io_clear(DS1104_DIO5);                       //Clear DIR
                    pin (Down shift)
                DIR=0;
                ds1104_bit_io_clear(DS1104_DIO7);                       //Clear
                    HOME pin
                HOME = 0;
                next_gear = current_gear - 1;
                shift_sequence = 1;
            }
        }
```

```
}

switch(shift_sequence){
    case 1:
        if((fabs(get_torque()) < epsTorque) && !ACK){          //Acknowledge
             M=0 and acknowledge HOME cleared?
            shiftRobotCom(current_gear, next_gear);            //Set N pin
            shift_sequence = 2;
        }
        break;
    case 2:
        time_out1++;
        if(ACK){                                               //Acknowledge
             neutral?
            setControlMode(ControlModeSpeed);
            w_set = referenceSpeed(w_wheel*3*2, current_gear, DIR);   //Set
                 speed according to current gear and speed, v=x
            ds1104_bit_io_clear(DS1104_DIO0|DS1104_DIO1
                                |DS1104_DIO2|DS1104_DIO3
                                |DS1104_DIO4);                 //Clear N pins
            N12 = N23 = N34 = N45 = N56 = 0;
            time_out1 = 0;
            shift_sequence = 3;
        }else if(time_out1 >= TIME_OUT1){                      //Timeout if N
             not reached
            setControlMode(ControlModeSpeed);
            /*ds1104_bit_io_clear(DS1104_DIO0|DS1104_DIO1
                                |DS1104_DIO2|DS1104_DIO3
                                |DS1104_DIO4); */                //Clear N pins
            //ds1104_bit_io_set(DS1104_DIO7);                  //Set HOME pin
            time_out1 = 0;
            shift_sequence = 0;
        }
        break;
    case 3:
        w_set = referenceSpeed(w_wheel*3*2, current_gear, DIR); //Set speed
             according to current gear and speed, v=x
        if((fabs(w_set - w) < delta_wEl_Max) && !ACK){          //Acknowledge
             v=x?
            ds1104_bit_io_set(DS1104_DIO6);                    //Set MAX pin
            MAX = 1;
            shift_sequence = 4;
        }
        break;
    case 4:
        time_out2++;
        if(ACK){                                               //Acknowledge
             MAX?
            exec_time  = RTLIB_TIC_READ() * 1000;
            ds1104_bit_io_clear(DS1104_DIO6);                  //Clear MAX pin
            ds1104_bit_io_set(DS1104_DIO7);                    //Set HOME pin
            MAX = 0;
            HOME = 1;
            time_out2 = 0;
            shift_sequence = 0;
        }else if(time_out2 >= TIME_OUT2){                      //Timeout if
             MAX not reached
            if(controlMode == ControlModeSpeed) {
                setControlMode(ControlModeTorque);
                set_torque(0);                                 //Set motor
                     torque to zero, M=0
                ds1104_bit_io_clear(DS1104_DIO6);              //Clear MAX pin
                shiftRobotCom(current_gear, next_gear);        //Set N pin
                MAX = 0;
                shift_sequence = 0;
            }
            //ds1104_bit_io_set(DS1104_DIO7);                  //Set HOME pin
            //time_out2 = 0;
            //shift_sequence = 0;
        }
        break;
}

/*-------------------------------------------------------------------------*/
/*-- ISR ROUTINES BY PER AND AXEL ENDS ------------------------------------*/
/*-------------------------------------------------------------------------*/

/* write PWM Duty cycle to slave DSP and test for error */
ds1104_slave_dsp_pwm3_duty_write(task_id, index, duty3_a, duty3_b, duty3_c);

/* activate the previously written DAC values synchronously */
ds1104_dac_strobe();

/*---------------------------------Read from AD converter---------------*/
ds1104_adc_start(DS1104_ADC2|DS1104_ADC3|DS1104_ADC4|DS1104_ADC5);

ds1104_adc_read_mux(scantable, 4, u);
AD1 = u[0];                                  //Current active gear (Potentiometer
     reading)
w_analog    = 1000*u[1];                                  // AD2   // w is calculated
     from delta_theta instead
AD3 = u[2];                                  // AD3
Udc = u[3];                                  // AD4
ia_u_raw  = ds1104_adc_read_conv(2);           // AD5
ib_u_raw  = ds1104_adc_read_conv(3);           // AD6
cosTheta = ds1104_adc_read_conv(4);          // AD7
sinTheta = ds1104_adc_read_conv(5);          // AD8
```

```
    //omegaFiltred = (omega*0.01 + 0.99*omega_temp);     // läser in w och filtrerar
        denna
    //omega_temp = omegaFiltred;

    // --- filters

    AD1_filtred = AD1*0.01 + 0.99*AD1_temp ;              // filtrerar AD1 : växel
    AD1_temp = AD1_filtred;


    /*----------------------------------Switchning on/off--------------------*/

    //Available only for some of the interfaces

/*???     if(sw==1) {
        ds1104_bit_io_set(DS1104_DIO11);
    } else if(sw==0) {
        ds1104_bit_io_clear(DS1104_DIO11);
    }
*/
    /*----------------------------------DAC----------------------------------*/
    ds1104_dac_write(1, DA1);


    /*Put all your controller code here*/

    /*----offset eliminering, filter and Skalning-----*/

    ia_u_filter = (ia_u_raw + ia_u_off) * 0.1    + (1 - 0.1) * ia_u_filter;
    ib_u_filter = (ib_u_raw + ib_u_off) * 0.1    + (1 - 0.1) * ib_u_filter;

    ia = ia_u_filter * ia_adj;
    ib = ib_u_filter * ib_adj;
    ic = -ia - ib;


    Udc = Udc * Udc_adj;

    Udc_raw_test = Udc;
    Udc=udcFilter(Udc);

                                        //calc. current c

//???     w = (-w*854*0.01 + 0.99*w_temp);                    // läser in w och
      filtrerar denna
//???     w_temp = w;

    /*----calc theta and new cosTheta and sinTheta-------*/


    theta_raw       = calcTheta(sinTheta, cosTheta, Theta_adj);

    //??? array_push_back (arr_time , total_time) ;
    array_push_back (arr_theta, theta_raw) ;
    array_theta_continuous(arr_theta, arr_theta_cont);

    theta_old   = Theta;


    Theta = linear_approx( arr_time,  arr_theta_cont,  arr_time[MAX_ARR_SIZE-1],
        out_coef_arr);
    cosTheta = cos(Theta);
    sinTheta = sin(Theta);

    //Theta    = mean(arr_theta_cont);
    view_00 = theta_raw;


    /*----calculate omega from rotor position-------*/
    w_lin = sign_ind *out_coef_arr[0] / Ts;
    array_push_back (arr_w_lin, w_lin) ;
    view_01 = mean(arr_w_lin);


    omega_temp = omegaFiltred;
    //omega = sign_ind * calc_omega(Theta, theta_old, Ts);
    //omega = w_lin;
    omega = mean(arr_w_lin);
    filter=max(w_filter_needed-fabs(omegaFiltred), 1);
    omegaFiltred = (omega*1/filter + (1-1/filter)*omega_temp);
                                // läser in w och filtrerar denna

    w = omegaFiltred; // strange ??? for some reason the angular velocity need to
        be multiplied by -1


    rpm_axel = w*30/(Npp*R_am*PI);


    /*------beräknade parrametrar------- */
    isy_temp = isy;
    isx_temp = isx;
```

```
    i_alpha = sqrt_three * ia / sqrt_two;                                   //3phase ->
          2phase transformation
    i_beta = (ib - ic)/sqrt_two;                                            //
          Powerinvariant conversion

    isx = i_alpha*cosTheta + i_beta*sinTheta;                               //ab -> xy
          coordinate transformation
    isy = i_beta*cosTheta - i_alpha*sinTheta;


    if (controlMode == ControlModeSpeed)
    {
        set_torque( J_el_machine / (100000*Ts) * ( w_set - w )); // ?
    }



    calculateUsxUsy (); // calculate the required stator voltages usx,usy
                        // to acchieve referens currents isx_set isy_set at the
                             current w

    //??? usx=0;
    //??? usy=test_dc ;
    //usy=(test_dc + test_amplitude * (sin(test_omega * total_time)));

    //--- from xy to alpha_beta coordinates
    theta_12     = Theta + sign_ind * w * 3/2*Ts;    // predicted mean theta at
          control interval 1-2 ;
                                              // 0:begin of this interval 1: now+Ts
                                                   2: now+2*Ts
                                              // factor sign_ind = -1  because w and
                                                   theta have different sign
    u_alpha_set = usx*cos(theta_12) - usy*sin(theta_12);          //xy -> alpha-beta
          coordinates
    u_beta_set  = usx*sin(theta_12) + usy*cos(theta_12);
    //u_alpha_set = usx*cosTheta - usy*sinTheta;                  //xy -> alpha-beta
          coordinates
    //u_beta_set = usx*sinTheta + usy*cosTheta;

    u_a_set = sqrt_two*u_alpha_set/sqrt_three;                   //Twophase to
          Threephase coordinates
    u_b_set = u_beta_set/sqrt_two - u_alpha_set/sqrt_six;
    u_c_set = -u_alpha_set/sqrt_six - u_beta_set/sqrt_two;
    //a = u_a_set - u_b_set;


    /*-----------------------------------Duty cycle for 3-phase motor PWM-----------*/


//???     duty1_a = 0.5;
//???     duty1_b = 0.5;
//???     duty1_c = 0.5;

duty1_a = u_a_set/Udc + 0.5;
duty1_b = u_b_set/Udc + 0.5;
duty1_c = u_c_set/Udc + 0.5;



    /*-----------------------------------Duty cycle for 4 channel PWM---------------*/

    /* write PWM Duty cycle to slave DSP and test for error */


    // channel 1 not in use
    ds1104_slave_dsp_pwm_duty_write(task_id, ch1_index, 0.1);

    // channel 2
    ds1104_slave_dsp_pwm_duty_write(task_id, ch2_index, duty1_a);

    // channel 3
    ds1104_slave_dsp_pwm_duty_write(task_id, ch3_index, duty1_b);

    // channel 4
    ds1104_slave_dsp_pwm_duty_write(task_id, ch4_index, duty1_c);


    /************************************************************************
    *
    * PWM connections on the interface (from top):
    *     _____________
    *     | 3_b |     | 3_a |
    *     _____________
    *     |     | 3_c |     |
    *     _____________
    *     | 1_c | 1_b | 1_a |
    *     _____________
    *     |     |     | ch1 |
    *     _____________
    *
    * Channel 1 on the 4 channel PWM is not in use
    * Channel 2 = 1_a
    * Channel 3 = 1_b
    * Channel 4 = 1_c
```

```
    ********************************************************************/

    //exec_time  = RTLIB_TIC_READ() / Ts;
    total_time = total_time + Ts;

    arr_counter++;
    arr_counter= fmod(arr_counter,MAX_ARR_SIZE)   ;

}

/*-----------------------------------Main-----------------------------------*/
void main(void)
{


    init();                             //DS1104 and RTLib1104 initialization
    initShiftSystem();                  //Shift system initiation

    /* initialization of slave DSP communication */
    ds1104_slave_dsp_communication_init();

    /* initialization of 3-phase PWM generation on slave DSP */
    ds1104_slave_dsp_pwm3_init(task_id, Ts, duty3_a, duty3_b, duty3_c,
                               deadband, sync_pos);

    /* registration of PWM duty cycle update command */
    ds1104_slave_dsp_pwm3_duty_write_register(task_id, &index);

    /* init D/A converter in latched mode */
    ds1104_dac_init(DS1104_DACMODE_LATCHED);

    /* initialization of PWM sync interrupt */
    ds1104_set_interrupt_vector(DS1104_INT_SLAVE_DSP_PWM,
                                (DS1104_Int_Handler_Type) &PWM_sync_interrupt,
                                SAVE_REGS_ON);

    /* initialization of 4-chanal PWM generation on slave DSP */
    ds1104_slave_dsp_pwm_init(task_id, Ts, duty1_a, mode, pol,
                              SLVDSP1104_PWM_CH1_MSK |
                              SLVDSP1104_PWM_CH2_MSK |
                              SLVDSP1104_PWM_CH3_MSK |
                              SLVDSP1104_PWM_CH4_MSK);

    /* registration of PWM duty cycle update commands */
    /* channel 1 */
    ds1104_slave_dsp_pwm_duty_write_register(task_id, &ch1_index, 1);
    /* channel 2 */
    ds1104_slave_dsp_pwm_duty_write_register(task_id, &ch2_index, 2);
    /* channel 3 */
    ds1104_slave_dsp_pwm_duty_write_register(task_id, &ch3_index, 3);
    /* channel 4 */
    ds1104_slave_dsp_pwm_duty_write_register(task_id, &ch4_index, 4);

    ds1104_enable_hardware_int(DS1104_INT_SLAVE_DSP_PWM);

    ds1104_set_interrupt_vector(DS1104_INT_EXTERNAL_3,
                                (DS1104_Int_Handler_Type) &ext_int_3,
                                SAVE_REGS_ON);

    ds1104_enable_hardware_int(DS1104_INT_EXTERNAL_3);


    RTLIB_INT_ENABLE();

    //start PWM interupt
    ds1104_slave_dsp_pwm3_start(task_id);


    ds1104_slave_dsp_pwm_start(task_id, SLVDSP1104_PWM_CH1_MSK |
                                        SLVDSP1104_PWM_CH2_MSK |
                                        SLVDSP1104_PWM_CH3_MSK |
                                        SLVDSP1104_PWM_CH4_MSK);

    /* Background tasks */
    while(1) {
        RTLIB_BACKGROUND_SERVICE();                     /* background service */

        if(up_shift_button != last_up_shift_button) {           //Any change to
             button?
            if(up_shift_button > last_up_shift_button) {        //Rising edge
                up_shift_button_flag = 1;
                last_up_shift_button = 1;
            } else {                                            //Falling edge
                up_shift_button_flag = 0;
                last_up_shift_button = 0;
            }
        }
        if(down_shift_button != last_down_shift_button) {       //Any change to
             button?
            if(down_shift_button > last_down_shift_button) {    //Rising edge
                down_shift_button_flag = 1;
                last_down_shift_button = 1;
            } else {                                            //Falling edge
                down_shift_button_flag = 0;
                last_down_shift_button = 0;
            }
        }

        if(button){
```

```
            shiftLoop();
        }

        w_set_view = calcRPM(w_set/3);
        w_view = calcRPM(w/3);
        w_wheel_view = calcRPM(filtered_w_wheel);
    }

}
```