NTNU

Norwegian University of
Science and Technology

Design and Testing of a Race Car Inverter

Anders Holter Bjørkto

Simen August Tinderholt

Master of Science in Cybernetics and Robotics
Submission date: July 2016
Supervisor: Bjørn B. Larsen, IET
Co-supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem description

In 2016, Revolve NTNU's fifth car will represent the team in Formula Student. Formula Student is a competition for engineering students, where teams design and build racing cars. The cars are evaluated in a series of tests evaluating performance and engineering design decisions.

For this season, Revolve wants to develop a self-made inverter, with the goal of increasing the performance of the four-wheel driven car. The power systems and motor control shall be implemented and integrated with the other electronic and tractive systems of the car.

During the fall of 2015, the candidates compiled a prestudy on the design of such a system. This work shall be continued by:

1. Revising solutions from the prestudy
2. Develop further requirements for the system
3. Design and manufacture hardware design prototypes of the system modules
4. Develop the necessary software of basic system functionality
5. If time permits, extend the performance of the system
6. Perform testing of the implemented system
7. Evaluate the final system with regards to suitability, quality and performance

Abstract

Electric drivetrains have become more prevalent in high performance motorsport communities. With high power to weight ratios and the ability to recover energy when braking, they are able to increase performance for racing cars. Such a machine needs to be powered and controlled. An inverter is the connecting interface between the power source and the motor performing this task. With the amount of energy and power needed to drive a racing car, safety and reliable operation is paramount.

This Master's Thesis will present the design, implementation and testing of a two-level voltage source inverter and motor control system. The target vehicle is driven on all four wheels, so a four channel inverter system is needed. The target motor of the car is a three-phase permanent magnet synchronous motor. The VSI presented is based on Silicon-Carbide Metal Oxide Semiconductor Field Effect Transistors (SiC MOSFETs).

The motor control system will be targeted at Atmel microcontrollers, for physical implementation in the car. Furthermore, base development for a System-on-Chip(SoC) will be done, for further development at later stages. Switching the motor control system onto a more powerful SoC will enable more precise control of the motor.

Sammendrag

Elektriske drivsystemer har gjennom de siste årene blitt vanlige i racing-miljøet. Høye kraft til vekt-rater samt muligheten for regenerering av energi ved bremsing, gjør at elektriske motorer kan brukes til å øke ytelsen til racerbiler. En slik maskin trenger energi og må kontrolleres, omformeren er koblingsleddet mellom energikilden og motoren som utfører dette. Med nok energi og effekt til å drive en racerbil, er sikkerhet og pålitelighet viktig.

Denne Masteroppgaven vil presentere designarbeid, implementasjon, og testing av et to-nivås spenningskildeomformer- og motorkontrollsystem. Systemet er tiltenkt brukt på en bil med firehjulstrekk via fire uavhengige motorer, så en fire-kanals omformer er ønsket. Motorene som skal drives er tre-fase permanentmagnet synkronmotorer, og inverteren vil være basert på Silisiumkarbid MOSFET-transistorer.

Motorkontrollsystemet vil i hovedsak bli utviklet for en Atmel mikrokontroller, for den praktiske implementasjonen i bilen. Grunnleggende arbeid vil også bli gjennomført på et enbrkkesystem (SoC) for videre arbeid i senere steg av utviklingen av systemet. Bruk av et enbrikkesystem vil tillate bruk av mer avanserte kontrollalgoritmer for mer korrekt styring av motoren.

Acknowledgements

We want to thank Revolve NTNU for trusting us with this project. Without financial support and know-how, this project would not have been possible, we therefore want to thank Revolve's sponsors for supplying us with components, PCB production and experience. For helping structuring our report, and keeping us focused, we want to thank our supervisors, Bjørn B. Larsen and Amund Skavhaug.

Contents

List of Figures

List of Tables

Abbreviations

4WD	:	Four-wheel drive
AC	:	Alternating current
ADC	:	Analog-to-digital converter
AIR	:	ccumulator isolation relay
ASIC	:	Application specific integrated circuit
BMS	:	Battery managament system
CAN	:	Controller area network
DC	:	Direct current
ECU	:	Engine control unit
EV	:	Electric vehicle
FPGA	:	Field-programmable gate array
FS	:	Formula Student
FSAE	:	Formula SAE
GLV	:	Grounded low voltage
HW	:	Hardware
IC	:	Internal combustion
IGBT	:	Isolated gate bipolar transistor
MOSFET	:	Metal oxide semiconductor field effect transistor
NTC	:	Negative temperature coefficient
PCB	:	Printed circuit board
PL	:	Programmable logic
PM	:	Permanent magnet
PWM	:	Pulse-width modulation
SAE	:	Society of Automotive Engineers
SiC	:	Silicon-carbide
SoC	:	System-on-chip
SoM	:	System-on-module
SPI	:	Serial peripheral interface
SW	:	Software
TV	:	Torque vectoring
VSI	:	Voltage source inverter

1. Introduction

1.1 Formula Student

Formula Student is a series of annual competitions where engineering students from all over the world compete with one-seat racing cars. The competitions started as SAE (Society of Automotive Engineers) Mini Indy at the University of Houston in 1979, but today they include a number of spin-off events all over the world. Fully electric vehicles (EVs) were introduced for the first time in Formula Student in 2010, and competes in a class separate from the internal combustion (IC) vehicles. However, in Formula Student UK, both EVs and IC vehicles compete in the same class, resulting in EV teams winning the event the last years.

Figure 1.1: Contestants at the 2015 FS Austria competition

The events are divided into eight different challenges, where a total of 1000 points can be obtained by each team. The team with the highest overall score wins the event. The total score is distributed, as shown in Table 1.1, between dynamic and static events. The static events are presentation, engineering design and cost analysis, and the goal of these events is to differentiate teams on the basis of planning, conceptual work, design, cost and general engineering practice.

Static events	Business presentation	75
	Cost and sustainability	100
	Engineering Design	150
Dynamic events	Skid-pad	50
	Acceleration	75
	Fuel Economy	100
	Autocross	150
	Endurance	300
Total score		1000

Table 1.1: Formula Student competition evaluation

The dynamic events are acceleration, skid-pad, autocross, fuel economy and endurance. The goal of the acceleration event is to drive 75 meters as fast as possible, from a standing start. The skid-pad event is a four-lap figure-of-eight course, testing the car's cornering abilities. In autocross, the vehicle's maneuverability and handling are tested at a tight track with sharp turns, to be completed as fast as possible. The endurance event evaluates the overall performance, and tests durability and reliability of the vehicle. It is a single 22km race, with driver change half way. This is one of the most important events both because it gives the most points, but also because it is the hardest test for both the car and the drivers. Fuel efficiency score is given in the same event, based on readings from an energy meter mounted in the car.

1.2 Revolve NTNU

Revolve NTNU is a racing team representing the Norwegian University of Science and Technology in the Formula Student competition. The team is an independent student organization, who works on the project of building a new racing car from scratch every year.

Revolve was founded in 2010, and the team's first entry to the Formula Student competition was in 2012 with the internal combustion car KA Borealis R. The team was awarded the "Best newcomer award" and finished at 17^{th} place overall in the UK. KA Aquilo R, the following year's car, made it to 16^{th} place in the same

competition. Revolve's first fully electric vehicle was designed for the 2014 season, KOG Arctos R. With this car, the team finished at 8th place in the UK. Revolve's 2015 electric car, Vilje, achieved a 3rd place in engineering design in the UK and a 4th overall in Formula Student Austria.

The 2016 car is called Gnist, and will participate in the Formula Student events at Silverstone in the UK, the Red Bull Ring in Austria and at Hockenheim in Germany. Gnist is the teams first four-wheel driven car, and with this new powerhouse and great results from last year, the goals are driven high, aiming for the top.

Figure 1.2: Revolve 2016 team after the reveal of Gnist

1.3 Motivation

A fully electric drivetrain has a distinct advantage to IC engines in the Formula Student competitions. The high torque and fast response of an EV fits well with the short tracks with tight and fast turns of the competition. The advantages does come at some cost, however. The battery accumulator of the electrical drivetrain is heavy, and there is added weight and complexity in the motor control system.

Revolve NTNU has been steadily improving their result every year since the first entry in 2012. This is a trend we want to continue, and the 2016 car therefore sports four-wheel drive (4WD). Each wheel is attached to a motor, each requiring a voltage source inverter to operate. In such a system, the control systems are more complex, and require more from the engine control unit (ECU). A 4WD solution adds considerable weight to the vehicle, due to multiple motors, power electronics and cooling circuitry. However, a 4WD system maximizes the total traction of the vehicle, and better utilizes regenerative braking. Individual control of four motors also allow for the implementation of torque vectoring (TV). A good TV algorithm distributes the torque between the four wheels, to give optimal traction and regenerative braking abilities. It also eliminates the need for a mechanical differential, reducing weight and increasing the level of control the driver has over the vehicle.

AMK has an inverter system that is widely used in the Formula Student competition. This system has also been considered by Revolve for the 2016 car, and it has been found that it is one of the best off the shelf solutions available. However, this inverter solution is large and heavy. In an acceleration and turning-focused competition like FSAE, weight is an important success parameter for any vehicle. Early estimates show that the inverter from AMK will weigh more than 7kg. In addition, the large form factor of the AMK inverter is difficult to position in the monocoque, and will be difficult to access when mounted in the car, should it be needed. The goal is to design an inverter that is an improvement over AMK's off the shelf solution. The main parameters that will be sought to be improved are weight, power efficiency and response time.

1.4 The electrical drivetrain

The teams competing in Formula Student are allowed to choose between internal combustion vehicles and fully electrical vehicles, as long as the vehicle complies with the corresponding rule set in the FSAE rules[16]. Both combustion and electric cars are power limited to ensure a safe competition. For electric vehicles, the limitation is set to a maximum voltage of 600V and a maximum power draw of 80kW from the battery accumulator.

The electrical drivetrain is the core of any electric vehicle, and consists of three main parts: The battery accumulator, the voltage source inverter and the motor.

In Revolve's 2016 car, a three-phase permanent magnet (PM) motor is mounted on the hub of each wheel, for a four-wheel drive.

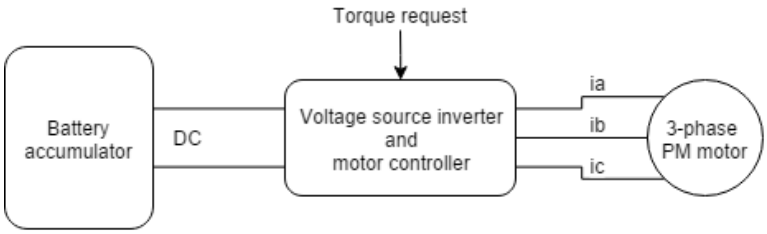


Figure 1.3: The electrical drivetrain[15]

1.4.1 Battery accumulator

The battery accumulator stores the energy for driving the vehicle. It consists of 288 battery cells , in 2 cells in parallel, and 144 144 in series, giving a nominal accumulator voltage of 550 V and a capacity of about 7.45 kWh. While each single cell is lightweight, the battery accumulator is the single heaviest part of the vehicle, weighing more than 45 kg.

The battery cells needs to be monitored to make sure the individual cell voltages and temperatures are within safe operating ranges. The battery management system (BMS) is responsible for this. If the BMS detects a cell voltage outside the safe range, it shuts down the tractive system through the accumulator isolation relays (AIRs), two DC relays, one connected in series with each battery pole. These open on command from the BMS, or if the shutdown circuit is opened. The shutdown circuit traverses most of the car, and is activated if any of the safety systems detect an error or a fault.

1.4.2 Voltage source inverter

The voltage source inverter (VSI, or simply inverter) connects the battery accumulator to the three-phase PM motor. The VSI modulates the three-phase current driving the PM motors. The amplitude and frequency of the currents are determined by the motor controller, based on a torque request from the ECU.

The electromagnetic torque of a permanent magnet motor is determined from the permanent-magnet flux, the number of pole pairs and the stator current, as in

Equation 1.2.

$$\begin{bmatrix} i_a(t) \\ i_b(t) \\ i_c(t) \end{bmatrix} = I_s \cdot \begin{bmatrix} \sin(\omega \cdot t) \\ \sin(\omega \cdot t - \frac{2\pi}{3}) \\ \sin(\omega \cdot t - \frac{4\pi}{3}) \end{bmatrix} \tag{1.1}$$

$$T_e = \frac{3}{2} P \cdot \varphi_m \cdot I_s \tag{1.2}$$

We can see From Equation 1.1 that a three-phase PM motor produces torque from three phases of sinusoidal current waveforms such as in Figure 1.4. The torque provided by the PM motor is controlled by changing the amplitude and frequency of the stator current waveforms. This is done by high frequency switching of the transistors in each phase leg in Figure 1.5. The amplitude determines the motor power, while the frequency is proportional to the speed of the motor. The task of the motor controller is to control the switches of the VSI in such a way that the amplitude and frequency corresponds to the driver's requested torque.

A two-level VSI such as the one in Figure 1.5 is the most widely used structure for smaller scale systems. It is compact, lightweight and have relatively good performance. Higher performance can be achieved with multilevel VSIs, where the DC voltage is divided into multiple levels. This will reduce losses in the motor, but the drawback is the rapidly increasing size: a three-level VSI requires 12 transistors, instead of only 6 for a two-level variant. Thus, it was early decided on a two-level system, due to easier implementation of the control system, and low weight being one of the main focuses for this car.

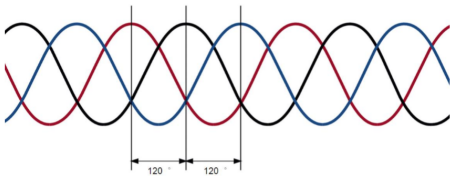


Figure 1.4: Three-phase alternating current waveforms[15]

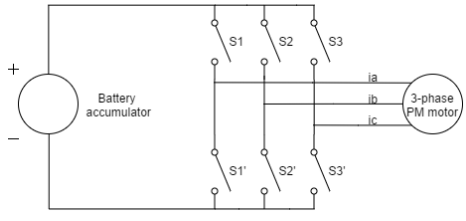


Figure 1.5: Two-level VSI structure[15]

1.4.3 Permanent-magnet motor

The permanent magnet (PM) synchronous motor is the electric machine with the highest torque-to-weight ratio, and therefore the most suitable for a low weight, high power race car. While expensive, the additional cost compared to simpler, weaker motor solutions can easily be justified by the higher performance of the PM motor.

This season, AMK's DD5-14-10-POW is the chosen motor. Its most important characteristics are listed in Table 1.2. The high power and low weight makes it ideal for a FS four-wheel drive car.

Continous motor power	P_n	12.3 kW
Maximum torque	M_{max}	21 Nm
Maximum current	I_{max}	100 A_{RMS}
Rated speed	N_n	12000 rpm
No-Load speed	N_0	18617 rpm
Mechanical speed limit	N_{max}	20000 rpm
Pole number	P	10
Stator inductances (dq axis)	L_d/L_q	0.44 mH / 0.54 mH
Weight	m	3.55 kg

Table 1.2: AMK motor characteristics[1]

2. Motor control theory

In this chapter, the mathematical basis for the implemented control system will be presented. Field Oriented Control theory and the relevant expansions will be explored, and a simple simulink model designed for testing the effect of different systems will be presented.

2.1 Basic equations

The controlled system is a Permanent Magnet Synchronous Motor from AMK. The desired controlled variables are torque and speed. The motor output speed can be expressed as a result of the developed electrical torque T_e through Equation 2.1 where ω is the rotational speed of the rotor in rads/s, T_e is the torque developed by the motor, and T_l is the load torque. In our analysis, we will first model the motor currents as a vector I_s in the three-axis coordinate system shown in Figure 2.1. The current vector can then be represented as in Equation 2.3. T_e can be expressed as Equation 2.2, where P is the number of poles in the motor, and λ_r is the flux coupling between the stator and rotor.

$$\frac{d\omega}{dt} = \frac{T_e - T_L}{J_e q} \tag{2.1}$$

$$T_e = \frac{3P}{2}\lambda_r I_s \tag{2.2}$$

$$\vec{I_s} = \sqrt{\frac{2}{3}}(i_a e^{j0} + i_a e^{j2\pi/3} + i_a e^{j4\pi/3}) \tag{2.3}$$

$$\vec{V_s} = \vec{R_s}\vec{I_s}(t) + \frac{d}{dt}L_s\vec{I_s}(t) + \lambda_r\omega \tag{2.4}$$

While this model of the motor is useful for analysis, and has been used in motor

control, it does not lend itself to dynamic control. This is partially because it offers little insight into the workings of the motor, and partly because it is overly complex, but primarily because it presents a nonlinear time variant system. To be able to apply control theory to the motor control system we will need to modify the model.

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ 0 \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} cos(\theta) & cos(\theta - \frac{2\pi}{3}) & cos(\theta - \frac{4\pi}{3}) \\ -sin(\theta) & -sin(\theta - \frac{2\pi}{3}) & -sin(\theta - \frac{4\pi}{3}) \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} \tag{2.5}$$

$$i_a + i_b + i_c = 0 \tag{2.6}$$

$$i_c = -i_a - i_b \tag{2.7}$$

The first modification to be done is to realize that while there are three input currents, we only need two variables to parametrise the two-dimensional current vector I_s. Because the three phases are connected internally, the currents must sum to zero, as in Equation 2.6. We can therefore eliminate one variable from our system, and represent our current vector in a Cartesian coordinate system. The system chosen has it's primary axis (α) on the A-axis of ABC system, and secondary (β) 90 degrees counter clockwise. The transformation from ABC to the $\alpha\beta$ system is called Clarke's transformation, and is seen in Equation 2.5.

$$\vec{i_{dq}} = \sqrt{\frac{2}{3}}(i_a e^{j0} + i_a e^{j2\pi/3} + i_a e^{j4\pi/3})e^{j\theta} \tag{2.8}$$

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} cos(\theta) & cos(\theta - \frac{2\pi}{3}) & cos(\theta - \frac{4\pi}{3}) \\ -sin(\theta) & -sin(\theta - \frac{2\pi}{3}) & -sin(\theta - \frac{4\pi}{3}) \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{2.9}$$

While Clarke's transform reduces the amount of variables to work with, it does not solve the primary problem of the ABC model. A direct $\alpha\beta$-based model would still have to output a sine wave response for a constant torque output. To solve this we let our coordinate system rotate with the rotor, affixing the α axis to the rotor's north-south axis. The transformation from the ABC reference frame

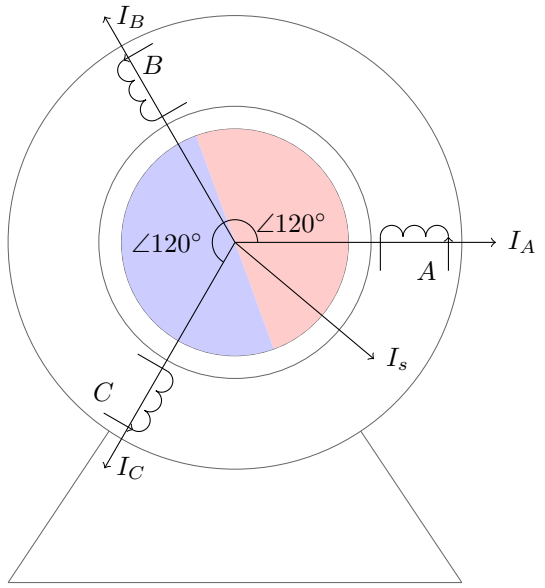


Figure 2.1: ABC model of a Permanent Magnet Synchronous Motor

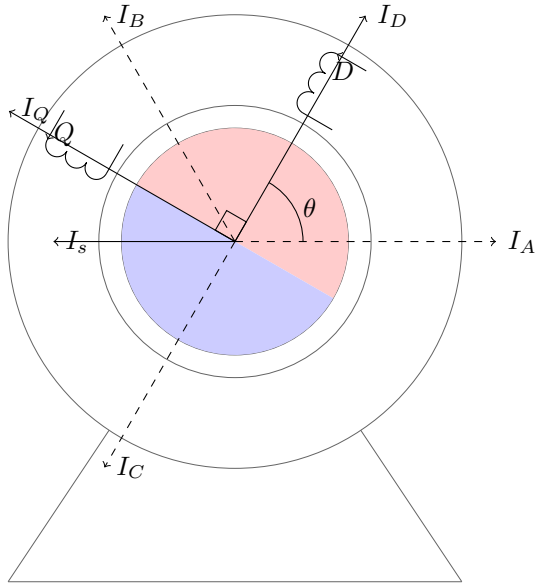


Figure 2.2: DQ model of a Permanent Magnet Synchronous Motor

to this rotating reference frame is called Park's transformation, and is found in Equation 2.9. The axes in this reference frame is refered to as the direct (d) axis, and the quaternion (q) axis, after it's orientation relative to the rotor flux's primary axis.

2.2 Field-Oriented Control

$$T_e = \frac{3}{2} P i_q (\lambda_r - (L_q - L_d) i_d) \tag{2.10}$$

$$T_e = \frac{3}{2} P i_q \lambda_r \tag{2.11}$$

When visualizing the motor currents in the DQ reference frame as in Figure 2.2, sine wave currents are reduced to two dc currents. In general, the d axis current controls rotor field strength, and the q current generates torque on the rotor. The torque developed is given by Equation 2.10.

The most basic control loop implemented in this reference frame is shown in Figure 2.3. It is based on the assumption that when operating below the rated speed of the motor (18600 RPM for the AMK), it can be assumed for an ideal motor that controlling i_d to zero will yield an optimal control algorithm, and Equation 2.10 reduces to Equation 2.11.

This is the basis for Field Oriented Control. Feedback control is provided through measuring the phase currents, and using an encoder to measure the rotor position and perform Clarke's transformation to calculate I_{dq}. PID regulators can then be used to calculate the required output voltage V_{dq}. $|V_{dq}|$ must allways be kept less than the DC motor supply. v_q should therefore be reduced to ensure this limitiation. Multiplying V_{dq} with the inverse Park's transform matrix yields the vector of output voltages V_{ABC} that is fed to the PWM generator driving the power transistors generating an output voltage. By setting $i_{d_{ref}}$ to 0, and calulating $i_{q_{ref}}$ with Equation 2.11, fast and correct torque response can be achieved up to the motor rated speed. If the motor is driven further, v_d will dominate v_q, degrading the inverter's ability to push i_q, decreasing the generated torque.

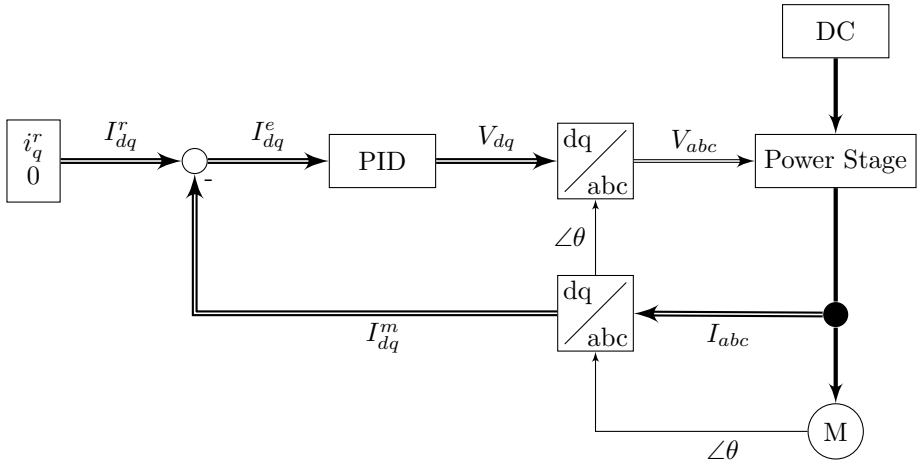


Figure 2.3: Flow diagram of a simple FOC inverter

2.3 Advanced Control Methods

$$v_d = R_s i_d + L_d \frac{di_d}{dt} - \omega L_q i_q \tag{2.12}$$

$$v_q = R_s i_q + L_q \frac{di_q}{dt} - \omega L_q i_d + \lambda\omega \tag{2.13}$$

The FOC model of the motor can be expressed as an optimization problem. Limited by the current rating of the motor, and the voltage of the DC supply, we want to minimize the current used to generate the requested amount of torque.

$$min \qquad |I_{dq}(T_e)| \tag{2.14}$$

$$s.t \qquad T_e = T_{e_{ref}} + \epsilon \tag{2.15}$$

$$i_q^2 + i_d^2 \leq i_{s_{max}}^2 \tag{2.16}$$

$$v_q^2 + v_d^2 \leq v_{dc}^2 \tag{2.17}$$

If we insert Equation 2.12 and Equation 2.13 in Equation 2.17 the voltage constraint

can be restated as Equation 2.18.

$$\left[(R_s i_q + L_d i_d \omega + \lambda\omega)^2 + (R_s i_d - L_d i_q \omega)^2\right] \leq v_{dc}^2 \tag{2.18}$$

2.3.1 Maximum Torque Per Ampere

While the assumption that controlling i_d to 0 will result in an optimal control scheme is not completely incorrect, we can see from Equation 2.10 that it is not the whole truth. If we want to achieve optimal torque control while minimizing the applied current, we must make use of reluctance torque, the torque generated by the second term of the equation. As we increase the amount of applied torque current, the optimal setpoint for i_d is actually slightly negative. Equation 2.19 from [18] shows the equation calculating an optimal i_d setpoint when disregarding any disturbances and parameter errors. It is derived from the Figure 2.4a, showing the path of this algorithm in black and a constant torque of 10Nm in blue. The algorithm is called the Maximum Torque Per Ampere path, and minimizes current draw for a given torque value.

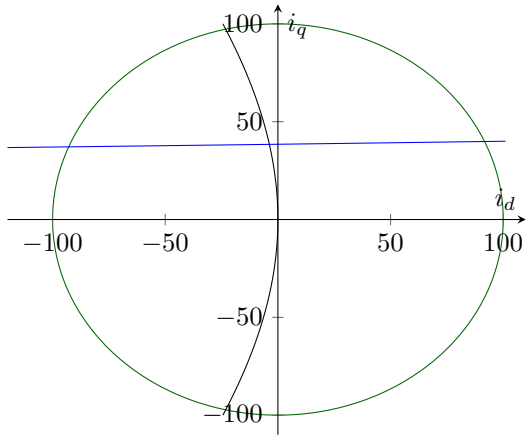
$$i_{d_{MTPA}} = \frac{\lambda}{2(L_q - L - d)} - \sqrt{\frac{\lambda^2}{4(L_q - L_d)^2} + i_q^2} \tag{2.19}$$

The limits of the AMK motor system are displayed in Figure 2.4b. The green line shows the 100A current limit, while the red and orange lines represent the voltage constraint when $V_{dc} = 550V$ at $19000RPM$ and $16000RPM$, respectively.

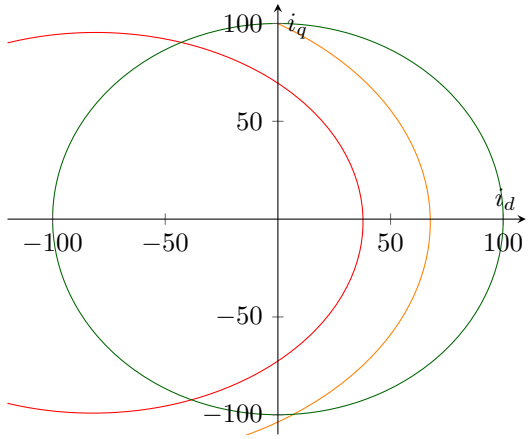
2.3.2 Field Weakening Control

As can be seen from Figure 2.4b, as the speed increases, the voltage limit ellipse constricts. In ordinary FOC, as outlined earlier, this causes the motor to lose torque and stabilize at some constant speed less than the no-load speed of the motor if 0 i_d is applied. As can be intuited from Figure 2.4b, it is possible to reach higher speeds than the rated speed of the motor. If one were to apply some negative i_d current, the current vector would be moved left in the plane, allowing the controller to follow the voltage limit curve upwards, generating more torque. This technique is called field weakening, as it works by weakening the permanent

magnet field generated by the rotor. Care should be taken to limit the duration and strength of field weakening, as this will heat up the stator magnets, and can permanently weaken them.



(a) The MTPA curve



(b) System limits at a speed of 16000 and 20000RPM

Figure 2.4: FOC parameter curves

Being able to operate above the maximum rated speed of the motor is beneficial, as it allows us to increase the gearbox reduction of the car, yielding higher output torque, and faster acceleration, while retaining the car's top speed. Gnist is designed for a top speed of $115 km/h$ with a gear reduction of approximately $15.5 : 1$, yielding a motor top speed of about $20\ 000 RPM$. As shown in Figure 2.5 the motor loses power at about 18 600 RPM. This necessitates field weakening action.

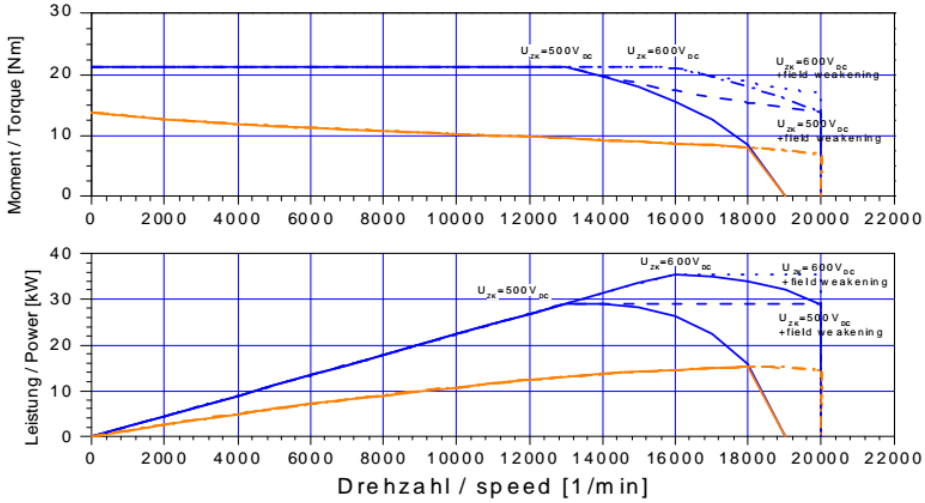


Figure 2.5: The torque vs speed plot of an AMK DD5 motor [1]

Field weakening control can be implemented in several ways. The algorithms can be grouped based on their working principle into four groups[13]:

Feed-forward These algorithms estimate the back-emf and required i_d current based on speed and/or current measurements. While the back-emf is estimated, no feedback is used, as the voltage is used to directly apply the folllowing current. These methods are generally fast and stable, but are sensitive to parameter deviations, temperature and inductor saturation.

Feedback These algorithms estimate the back-emf based on speed and/or current measurements. A feedback controller is used to controll the applied i_d. These methods are not as fast as feed-forward, and offer poor transient response. They are however more effective, as they utilize the DC-link voltage better by eliminating steady-state error. They are also less sensitive to parameter deviations, temperature and inductor saturation.

Hybrid In an attempt to get the best of both worlds, these methods use a combination of the preceding methods. For example, a feed-forward estimate can be modified by a feedback regulator, yielding faster step response while eliminating steady state-errors.

Advanced techniques Other, more advanced control-theory based algorithms may be employed. They generally offer further benefits over the hybrid

method, while requiring significantly more computing power.

As seen in Table 2.1, all algorithms have some redeeming attributes. Since this is a prototype project, the advanced algorithms were discarded early on. However, the hybrid field weakening algorithm outlined in [18] was considered possible to implement.

	Feed-forward	Feedback	Hybrid	Advanced
Parameter sensitivity	–	+	+	++
Transient response	++	-	+	++
Steady state error	–	+	+	++
Tuning complexity	++	+	-	–
Computational complexity	++	+	-	–
Design complexity	++	+	-	–

Table 2.1: Choice matrix for field weakening calculation algorithms.

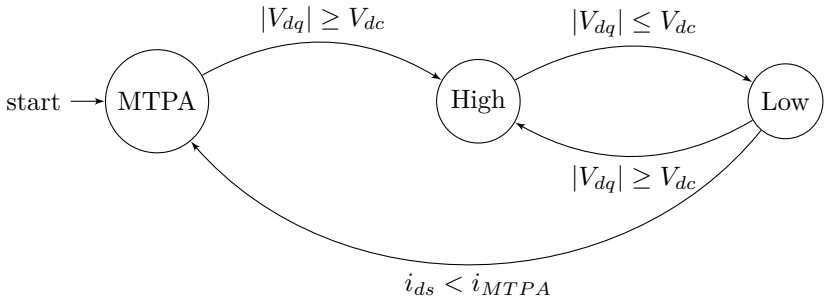


Figure 2.6: State diagram of the proposed voltage-follower FW algorithm.

The proposed algorithm tracks the output voltage of the PID regulator in Figure 2.3, comparing it with the DC supply voltage. The governing state machine is shown in Figure 2.6. In normal operation, the MTPA i_d setpoint is passed on, allowing optimal control when inside the operating limits. When the voltage limitation is reached and $|V_{dq}| >= V_{dc}$, field weakening operation is started. In the "High" state, the output voltage magnitude is too high, and d current is applied to reduce it. When in this state, the next i_d is decided by Equation 2.20. This is essentially an integration only PID controller, attempting to reduce the speed error. The proposed FW algorithm therefore requires the torque input to be governed by a speed controller. This is done to solve an issue that makes simpler FW approaches have problems winding down from FW mode. In the "Low" state, the motor is operating below the voltage limit. The I_{dq} vector is moved left toward

MTPA curve by Equation 2.21. When the MTPA curve is hit, and $i_{d_t} = i_{d_{MTPA}}$, control is returned to the MTPA algorithm. This algorithm allows the inverter to effectively track the system limits.

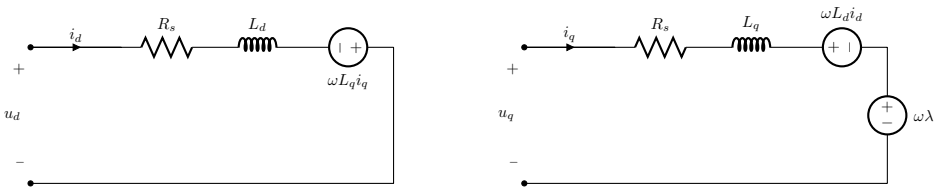
$$i_{d_t} = i_{d_{t-1}} - \alpha(\omega_r - \omega) \tag{2.20}$$

$$i_{d_t} = i_{d_{t-1}} + \beta(\omega_r - \omega) \tag{2.21}$$

2.4 System modelling

Several attempts have been made at constructing a comprehensive model of the motor controller system. For his 2015 master's thesis, Lars Helge Opsahl developed a Simscape Power Systems-based model. This model computed AC waveforms, switching stage performance and motor response. This system was tried adapted to the new motors, but due to the power stage modelling, simulation performance was extremely poor. The source of the poor simulation performance, the power stage PWM signals, forces the simulation step size to grow extremely small to be able to catch the system dynamics. A 1 second long simulation could easily take 2 minutes to compute. While 2 minutes for 1 second is not that bad, it is often required to do longer runs of 10s or more, increasing simulation time to 20 minutes. As work had to be focused on the development of the system hardware rather than simulation, it was decided not to pursue this model further.

The motor simulation work was later returned to, with the aim of finding an accurate model with higher simulation performance. On the assumption that the only way to alleviate the performance issue is to avoid modelling the power stage modulation, or implement some way of simulating it's effects, a new model was designed. By abstracting the power stage output as an ideal, bounded, voltage source, the total simulation time for a 1s simulation is reduced from 2 minutes to 15 seconds, a speed boost of 800%. This new model represents the motor dynamics in the DQ reference frame, rather than the ABC frame, as this reduces the amount of computations required.



(a) D-axis motor model

(b) Q-axis motor model

Figure 2.7: D and Q axis models, from page 328 of [13]

2.4.1 Motor Model

Figure 2.8 shows the motor model. Based on the equivalent circuit in Figure 2.7, the motor phase coils in the DQ plane are modeled as a first order RL filter. The back emf voltage is estimated based on Equation 2.13 and Equation 2.12, and subtracted from the input voltage. While it is in reality generated over the length of the coil, the lack of any capacative elements in the circuit allows us to make this simplification. Equation 2.10 is used to calculate developed torque, and integrated to calculate output speed. The instantaneous motor output power is calculated by Equation 2.24, with T_e given in Nm, and ω in rad/s. The speed value is then scaled from rad/s to RPM before output. As the motor phases are modelled as a filter, it is possible to input both idealized DC voltages, and realistic pulse width modulated signals. If desired this model could therefore be used to model other vector control systems using more advanced output modulation techniques than PWM.

$$\omega = \int \alpha \tag{2.22}$$

$$\omega = \int T_e/J \tag{2.23}$$

$$P_{mech} = \omega T_e \tag{2.24}$$

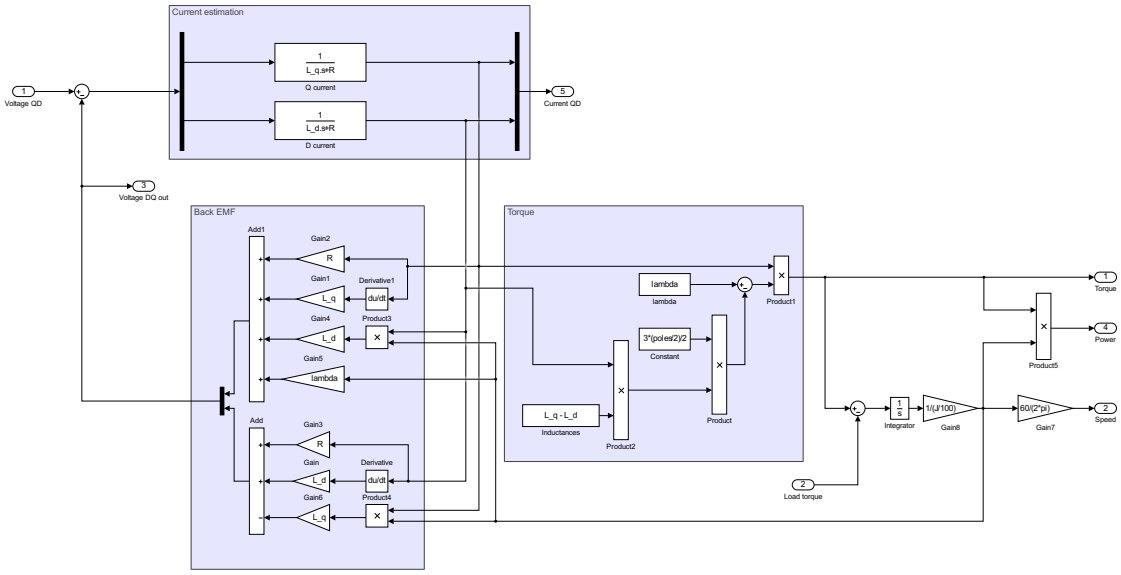


Figure 2.8: Motor model

2.4.2 Controller

The modelled control system uses a PI controller to regulate the phase voltages. The PI references are generated by the a PI speed controller, and the MTPA and Field Weakening algorithms. Before calculating the error vector, the i_q reference is corrected to ensure a request with magnitude lower than $100A$. The PI current regulator is set up with integral clamping.

2.4.3 Field Weakening

The field weakening algorithm described in subsection 2.3.2 is not trivial to implement in Simulink because of the state storage implied. However, a rough equivalent using switches and relay blocks to control the error input of a PID regulator has been developed for testing purposes. During testing this has allowed the simulated motor speed to exceed it's 18600 RPM limitation.

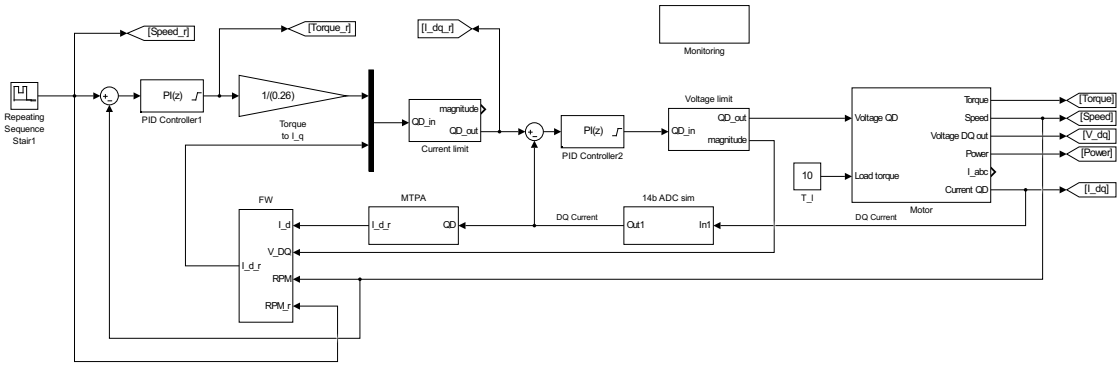


Figure 2.9: Simulink controller model overview

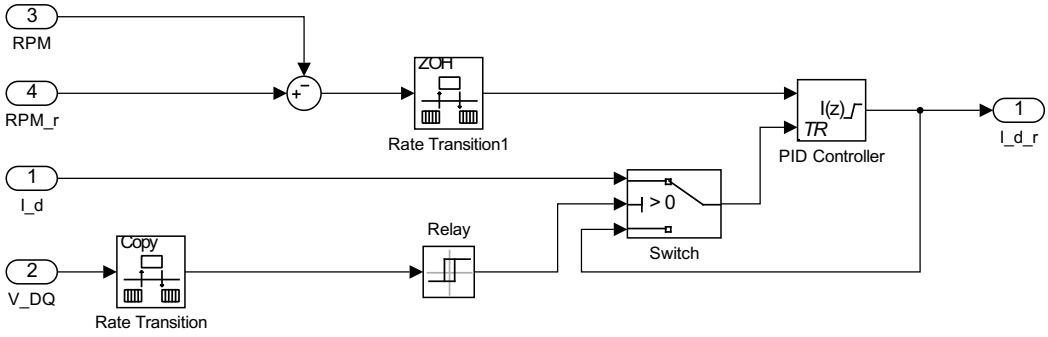


Figure 2.10: Field weakening approximation

3. System Requirements

This chapter will represent the requirements and functionality deemed necessary, based on the information in chapter 2, and experiences testing and tuning the 2015 inverter system.

3.1 Power system

The power system requirements were explored further in the authors' prestudy [20].

Functionality The power system must be able to output a variable voltage controlled by the central controller system. Assembly time should be considered when designing these systems.

Load The finished system should be capable of handling both the continuous and peak current loads of the motors.

Bandwidth The switching stage must be able to switch at a high enough frequency that stable control can be achieved.

Safety All Power systems should be designed to safely withstand it's working voltage.

All components of the inverter designed to carry phase power should be designed to carry the motor's rated current continuously, and its peak current for 3 seconds (the maximum time the car will use at peak torque).

When designing the high voltage systems of the car, all PCBs populated by both high and low voltage systems should have at least 4mm isolation zones around all borders, as well as coating the PCB with a conformal coating to eliminate creepage.

3.2 Digital Interface

The inverter needs to be able to receieve setpoints and commands from the rest of the car and the test crew via CAN. In addition to ordinary setpoint commands, status monitoring and test functionality should be in place. As the inverter produces a lot of data, a separate high bandwidth interface should be in place for testing.

Functionality The inverter interface should be able to adjust parameters, setpoints and state settings. A simple interface should be available to change regulator parameters and monitor state data.

Load Data transmission and reception should not interfere with normal operation.

Bandwidth The high-bandwidth interface should be able to send all relevant data at full speed. The CAN interface should not overload the rest of the car.

Safety Requests that may jeopardize safe operation of the inverter should not be fulfilled.

Revolve designs it's own analysis software that permits streaming and on-the-fly innterpretation of data from the car's CAN network. An inverter plugin should be designed for this software to allow easy access to state and parameter data. In adition, it's line graph plugin can be utilized for display of state data. A potential high bandwidth interface should be designed so as to be compatible with Analyze.

3.3 Physical Interface

The inverter must be able to interface with the motor encoders and temperature sensors. To simplify the ECU design, the inverter physical interface should mirror that of the AMK inverter. During testing season it will be useful to have programming ports available for each inverter in the casing.

Functionality The inverter must interface correctly with the motors, and CAN bus. External connectors should be available for a separate high speed interface, and programming. Connectors should be compatible with AMK alternative.

Load The external connectors should be made with high-quality connectors and shielded wiring to be noise-resistant.

Safety All internal connections should be done with high-quality connectors and wiring. Loose wiring inside the inverter can be a safety hazard.

All low-voltage outside connectors should be made with the same Deutcsh autosport connectors as the AMK inverter casing, to allow quick replacement should there be a need to. High quality connectors will also reduce clutter, problems and wear and tear during testing, saving precious hours of debugging.

3.4 Control system

The inverter control system should be able to regulate the motor output torque. Changes should be made to last years control system to ensure operation within the motor limits. Implementation of MTPA and field weakening algorithms should also be attempted.

Functionality The motor controller should regulate torque within the system limits.

Bandwidth The controller should have a high enough bandwidth to allow stable, efficient control, but not so fast that computing time jitter, interrupts or slower tasks causes the system to miss deadlines.

Safety The controller should allow safety shut downs, and guarantee safe operation in every plausible situation.

To reduce ECU coding complexity, and allow torque derating based on speed, the controller interface should mirror the AMK interface. Accordingly, the torque setpoint should be set by a PI speed regulator within limits dictated by the ECU. While this adds some complexity, it acts as a safety-feature, and allows hotswapping the two inverter systems.

3.5 Safety features

As the inverter directly controls the current in to and out of the battery, it is critical that errors are detected and acted upon. Overcurrent or overvoltage errors

could cause battery fires or damage the motor beyond repair. The inverter should act appropriately as fast as possible once an error is detected

Functionality The inverter should be able to detect overcurrent, overvoltage and overtemperature errors. Any errors should be signaled to the test crew, and appropriate action should be taken to avert damage.

Bandwidth The detection time constant should be chosen according to the failure cause dynamic.

Safety Error handling action should be taken before the end of the next control period after detection. Any action taken should not jeopardize the system or driver.

When an error state is discovered, the inverter should either reduce torque, cut torque to 0, or shut off the power stage. The phase currents are the fastest changing states in the system. Considering the phase current time constant of $3.5ms$, an error checking algorithm running more often than 1kHz should be fast enough to catch any state threshold violation fast enough to perform reparative action.

3.6 System Architecture

Mirroring the overall design of the 2015 inverter, the R16 is divided into a Control Card, a Gate Driver Board, and a Power stage. This allowed the design work to be segmented into a simulation-heavy power system, innovative gate driver system, and complex computing platform. The gate driver board also serves as the isolation barrier between the two others. This reduces the restrictions and design constraints on the control card. In addition, the inverter system should include the additional safety circuits required by the FSAE rules.

Control Card The control card contains all computing power in the inverter. It should be able to read all necessary sensors, and output a control signal to the power stage via the gate driver card. To allow switching between SoC and microcontroller solutions, a slot-in solution allows swapping the computing section of the board.

Gate Driver The 2015 inverter used 3 Semikron gate driver boards controlling one phase each. To shorten assembly time, the R16 will feature a single three-channel gate driver for each inverter. For the same reason, the gate driver should slot on top of the Power Board with headers. To reduce cable clutter all power stage temperature and current sensors are connected to the gate driver PCB and wired to the Control Card through a single cable bundle.

Power Board The power board constitutes the power stage of the R16. It connects power transistor legs, DC link capacitor, and motor phases through a PCB, and presents a detachable interface through the Gate driver. This board is designed to tolerate the high currents passing from the battery to the motor phases.

Additional Systems In addition to internal control safety, a system allowing the monitoring and safe charging or discharging of the inverter DC link capacitors is required. In the inverter, this system consists of the Voltage Indicator Circuit (VIC), discharge circuit, and shutdown circuit interlocks.

4. Hardware design

4.1 Current sensors

The motor controller requires feedback from all motor currents. The electromagnets in the motors are connected in a wye-configuration, as seen in Figure 4.1. Therefore only two currents need to be measured, the last phase current can be computed by Kirchhoff's current law, as in Equation 4.1.

$$\begin{aligned} I_a + I_b + I_c &= 0 \\ I_c &= -(I_a + I_b) \end{aligned} \tag{4.1}$$

The pedal sensors controlling the torque output of earlier cars have had less than 1000 steps from 0 to full torque without the driver mentioning any sensitivity problems. In addition, the motor connected to these cars were a 250Nm motor, more than 10 times more powerful than the current motor. In light of this, a torque ripple of $1/100th$ of the maximum output, or $0.21Nm$, should have acceptably low impact on driver performance. With a maximum phase current of $100A_{RMS}$, the maximum acceptable current ripple becomes $1.5A_{pk-pk}$. Therefore a current sensor system with a precision lower than $1.5A/LSB$ should be sufficient. While the maximum current through the motor should not exceed $100A_{RMS}$, experience from 2015 indicates that momentary spikes in the current due to fast switching transients may exceed twice that value. The maximum nominal current rating of the sensor should therefore be chosen higher than the peak value of the motor rating of $141A$. The peak current rating of the

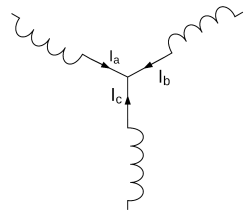


Figure 4.1: Motor's internal wye connection

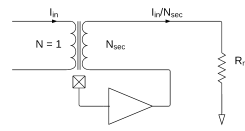


Figure 4.2: A closed loop hall effect sensor

Nominal current	>141A
Peak current	>300A
Bandwidth	>25kHz
Accuracy	$< 1.5A/LSB$
Noise	$1.5A_{pk-pk}$

Table 4.1: Current sensor requirements

sensor must be higher than $300A$. The sensor bandwidth needs to be higher than the fastest switching frequency planned for the inverter.

Current measurements can typically be done in two ways, current shunt measurement, or current transducers. In a current shunt measurement, current is passed through a high power, low resistance resistor, and the resulting voltage is amplified and measured. In a current transducer the magnetic field generated by the flowing current is measured and transformed by a hall-effect sensor into a voltage signal that is read directly, or a smaller current signal that may be measured via a current shunt. Using a current shunt is lighter and smaller, but complicates PCB layout. By applying current shunt measurements directly one also has to account for the considerable common mode voltages on the signal, while transducers typically are completely isolated from the high voltage system.

	HO-S 150	LA 150-P
Nominal current	150A	150A
Current range	$+/-375A$	$+/-212A$
Bandwidth	100kHz	150kHz
Noise	$+/-0.5\%$	$< +/-0.4\%$
Supply voltage	$5V$	$+/-15V$
Weight	32g	35g

Table 4.2: Current sensor core data, LA 150-Pweight with adaptor PCB

Because it is critical that these sensors work and show correct results, current transducers were preferred over shunts. In the project report, two designs were considered, and the Lem LA 150-P chosen. LA 150-P is a compensation sensor with a bipolar voltage supply. It was later discovered that the LA sensor does not satisfy the measurement range specification. It is possible, but expensive to supply the required $+/-15V$ power source. After reevaluating the sensitivity requirements and supply complexity, it was decided to replace the LA 150-P sensors with a non-compensated sensor. These sensors can be powered by the already present 5V supply. With proper calibration these sensors achieve better than 1% precision. With a 400A range, and 12 bit ADC, like the one in the Atsam, this

results in a sensor system with better than $1A/LSB$ precision, and $4A$ max error. In cooperation with a LEM automotive sales representative, several sensor solutions were considered. Finally LEM proposed the HO-S 150 sensor. Its most important data can be seen in Table 4.2.

4.2 Encoder

To measure the motor angle, each motor comes mounted with a Heidenhain ECI1118 encoder. This encoder features built-in self checks, and communicates via the advanced EnDat 2.2 protocol. The following information is gathered from the Heidenhain EnDat primer[7].

4.2.1 Endat

"The EnDat interface is a digital, bidirectional interface for encoders. It is capable both of transmitting position values as well as transmitting or updating information stored in the encoder, or saving new information. Thanks to the serial transmission method, only four signal lines are required. The data is transmitted in synchronism with the clock signal from the subsequent electronics. The type of transmission (position values, parameters, diagnostics, etc.) is selected through mode commands that the subsequent electronics send to the encoder."

The Encoder Data (EnDat) system specifies a voltage supply, standard pinout, communications protocol and certain encoder specifications. Most EnDat encoders are developed and sold by DR. JOHANNES HEIDENHAIN GmbH. The subsequent electronics - the user electronics - is responsible for supplying the required 5V voltage and RS485 transcievers, but there are FPGA IP's available to implement the communication protocol.

EnDat communicates via a dual RS485 hardware link. By using time skew compensation, to offset signal transmission time through the link, data rates up to 16MHz can be achieved by an EnDat 2.2 master.

The EnDat protocol allows serial communication with an absolute encoder, in addition, most EnDat encoders allow storing a position offset in the internal memories. In this way, the encoder 0 position can be aligned with the rotor magnetic field.

These features makes it ideal for motor control applications in an EV. The alternative absolute encoder interfaces are typically slower, and incremental encoders requires the car to be rolled before working, which is not ideal.

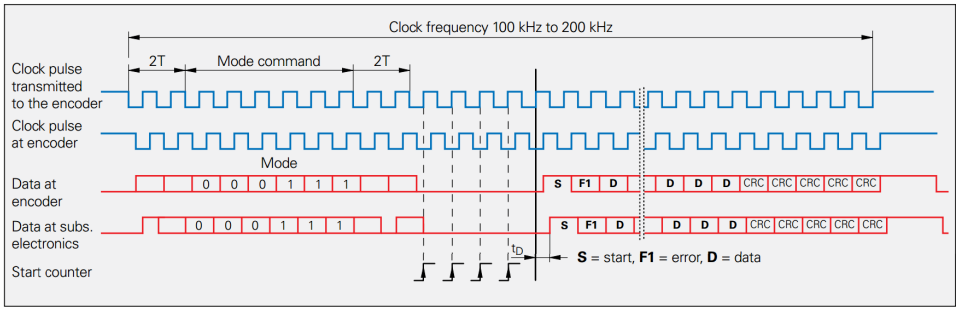


Figure 4.3: The EnDat serial protocol format. From page 6 in [7]

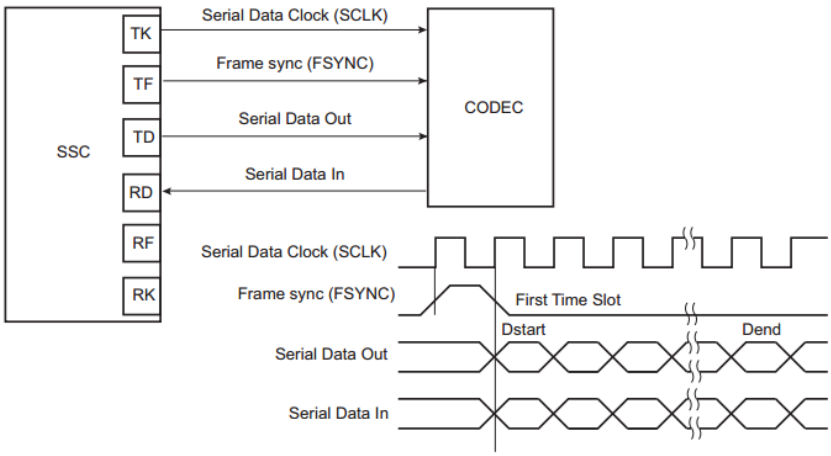


Figure 4.4: The SSC emulating a bidirectional serial protocol.
From page 1065 of the Atsam E70N21 datasheet [2]

As a result of it becoming clear that the inverter had to communicate with an EnDat encoder came late in the development process, an attempt had to be made to emulate the protocol. Communicating with the encoder requires sending two clock cycles with 0-bit data, then transmitting a three bit mode command, and its inverse, before a new two-bit cycle of 0-bits. The controller then waits for an unspecified amount of time, while clocking the encoder, until the encoder responds with the command result, some status bits, and a checksum value. The time delay before the response varies with the command excecuted. The command sequence not aligning two an 8 or 9 bit structure makes emulating this communication interface difficult,

an issue that is compounded by the variable delay, and continued clocking. This is not a form of communication supported in most serial communication modules.

The protocol was attempted emulated by the Synchronous Serial Controller in the Atsam. This module is made to be as flexible as possible, and allows the user to program his or her own serial interface. As seen in Figure 4.4, the module features one receive and transmit module, with both modules sporting a data line, clock line, and select line. The select lines can be programmed to signal various states in the reception and transfer buffer. When simulating the EnDat protocol, the transmit flag was used to assert the RS485 bus, enabling the bus driver when transmitting, and releasing it when receiving. The transmit clock was used to generate the clock bus signal, and to drive the internal receive clock, synchronizing the two buffers. The transmit message format is highly customizable. The length can be adjusted from 2 to 32 bits, and appended and prepended with various bit patterns. By transmitting a 6-bit message with two 0-bits before and after transmission, the command transmission could be successfully emulated. No good solution for the bus waiting was found, but an interrupt may be used to trigger reception.

While an output message that looked correct was transmitted to the encoder, no coherent answer message was ever received. As the EnDat protocol is extensive, and there is strict timing and correctness restraints, the reason for this was probably some error in the way data was transmitted, either in the RS485 transciever setup, or the setup of the SSC module. It was clear that an alternate method of implementing the EnDat controller was needed.

Texas Instruments has a line of ARM9-based processors containing a small programmable logic arera called the Industrial Communication SubSystem. This allows the processor to implement more involved communication protocols like Ethernet or EnDat without locking it into the hardware. These also come with premade libraries for motor control, allowing implementation of FOC for AC motors. This would have required learning another microcontroller platform, and reimplementing the nearly finished controller source code on this new platform, however. It was therefore not considered further.

Another solution was provided by Heidenhain themselves. Mazet, a german electronics and software company has developed several FPGA IPs for use with EnDat encoders. These act as master controllers, and communicate externally with SPI or paralell bus interfaces. As we had a compatible FPGA development board, and the atsam has an abundance of modules supporting SPI communication, this was

seen as a much more viable solution.

4.2.2 Altera MAX10

The FPGA IP required to communicate with the encoder was aquired by a sponsorship deal with Mazet and Heidenhain. The chosen FPGA on which to implement was an Altera MAX10, as there were a development kit for this FPGA available to the authors. While the MAX10 series does not feature a large number of logic elements, the entire series is available in QFP packaging, making PCB manufacturing possible within the team's production capabilities. The EnDat 2.2 master IP does not require a very large amount of logic elements, so even with two instances of the IP, all the MAX10 variants have a sufficiently large amount of elements to do the job.

The IP contained the HDL design of the communication protocol in enchrypted HDL, which was translated into a netlist and usable code using Synplify Premier. The resulting HDL code was then imported to Quartus and implemented in the block diagram shown in Figure 4.5. As each control card handles two inverters, two instances of the communication protocol IP is required on a single chip, keeping the hardware count low. The IP prefers a running frequency of 64 MHz, so a PLL was inserted to scale the clock frequencies provided from the 16 MHz clock on the control card. In addition to the encoder IPs and the PLL, a counter has been added to blink some LEDs on the board, so that a quick glance is enough to verify that the chip is alive. The results of the compilation of the full MAX10 system on a 10M16 variant of the MAX10 series is shown in Table 4.3.

	Used	Available
Total logic elements	4649	15840
Total registers	1957	
Number of pins	25	101

Table 4.3: Compilation restults of the MAX10

The encoder communication IP is provided with a SPI interface, making the connection to external controllers simple. It also includes the option of a parallel-bus interface, for a faster communication protocol when used with on-chip controllers inside an FPGA or a SoC. This is something that will be desired when the entire control system is implemented on a SoC, so that external systems can be reduced to a minimum.

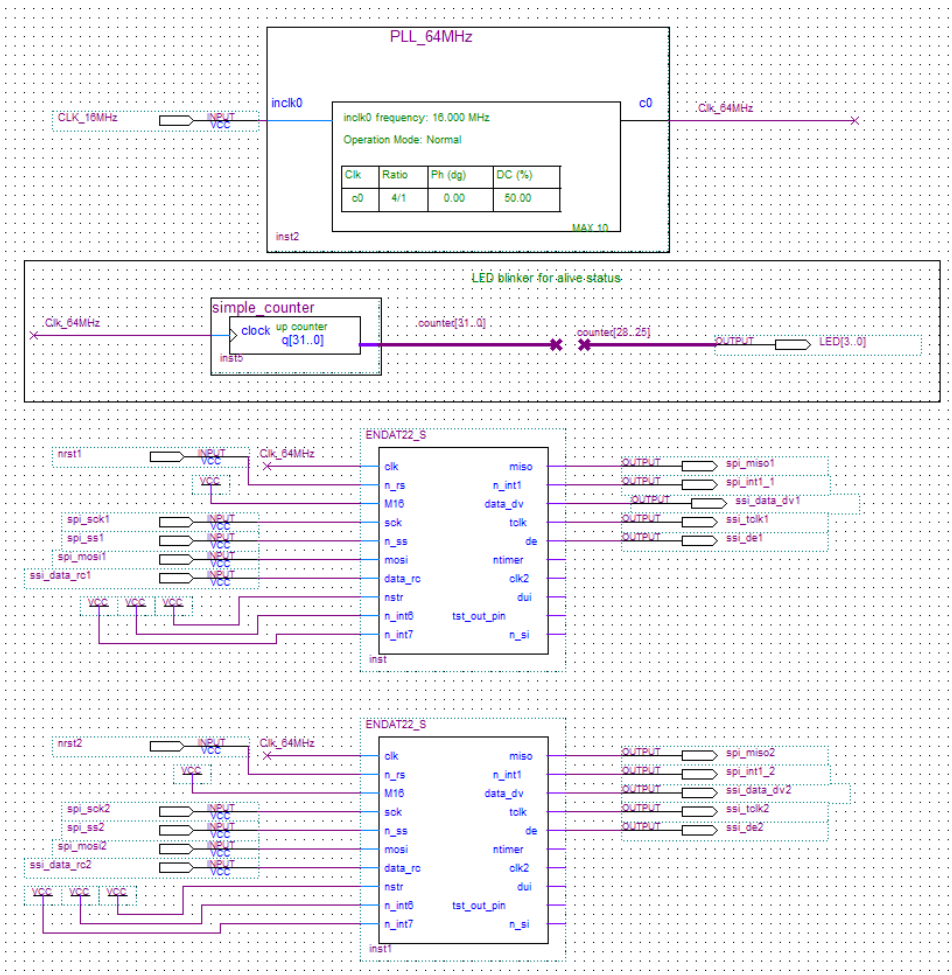


Figure 4.5: The final block diagram of the encoder communication system

4.3 Control Card and Insert

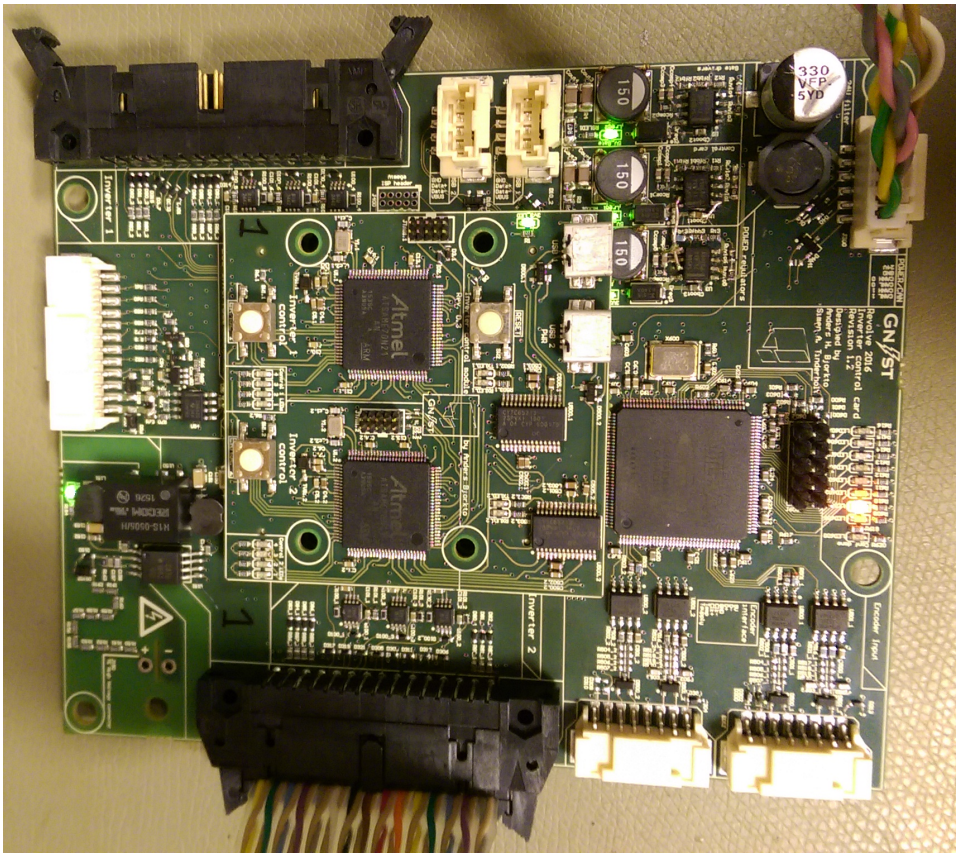


Figure 4.6: Inverter control card with Atsam insert

4.3.1 Specifications

The control board is the main processing node in the inverter, making it the most complex PCB in the system. To complete all necessary tasks, it should provide the following features:

- Header for Atsam insert and Enclustra Mercury ZX5.
- Power supply
- Car communication

- USB connectivity
- Isolation
- Encoder interface
- Safety system
- Voltage measurement

4.3.2 Design

The control board PCB was built on the work done in the previous semester [20], but due to changes in the specification, a redesign was necessary. The design of the Control card is largely dominated by overall design decisions made by the team over the last three years. This simplifies the design process for all members, and minimizes the necessary parts stockpile. This also facilitates debugging and repairs, as it quells the tendency of electronics members to become gurus of their own system, allowing them to quickly recognise modules in others' designs. The main control module on the board is based on the footprint of Enclustra's Mercury ZX5, and the two 168-pin headers in the centre of the board are the mounting for the insert. An overview of the board is given in Figure 4.7.

Power supply

The power supply is needed to drive the board, as well as supplying power to the gate drivers. The voltage regulators are based on Revolve NTNU reference designs, using a TI Webench-design incorporating the TPS54560 buck-regulator, using the GLV 24V as input. There are two 5V regulators: one supplying the control board itself, and one dedicated to supplying the two gate drivers the board is interfacing. To supply the board's 3.3V needs, a similar buck-regulator is implemented, but with a modified feedback bridge through the Rfbb3 and Rfbt3 in Figure 4.8, altering the output voltage, keeping all other parameters identical. These three voltage regulators, backed up by a second 3.3V regulator on the insert provide ample power to the low-voltage elements of the inverter.

Communication with the car

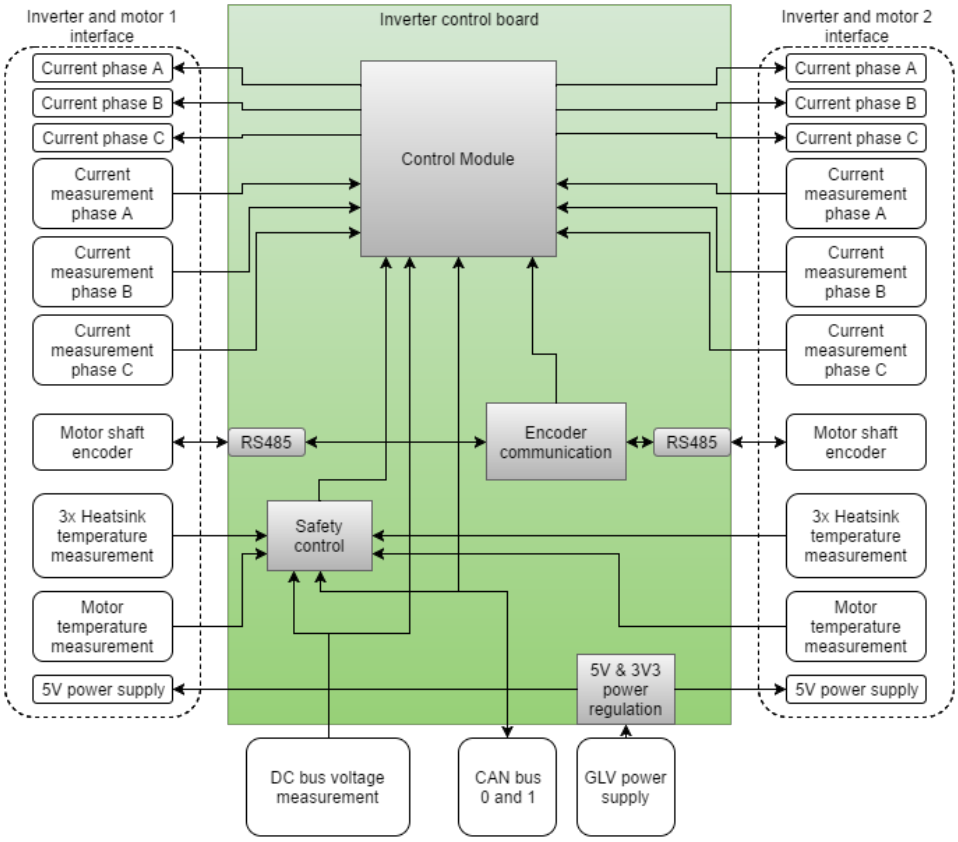


Figure 4.7: Inverter control card overview

The inverter interfaces the car in two ways: Can buses and a few signals directly from the ECU. Both the ZX5 module and the Atsam controllers have built-in CAN controllers, but neither have physical transceivers. Due to their high current consumption, and to reduce CAN bus stub lengths, these must be placed outside the controller, and as close to the outside connector as possible.

The transceiver used is the Texas Instuments' ISO1050 transceiver[12], the standard for all the systems in the car. There are two CAN buses in the car, and the control card must support up to two controllers on each PCB. There are therefore four chips on the board, with an additional transceiver to connect the Atmega safety

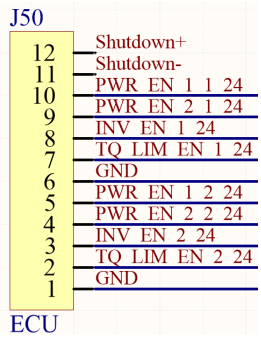


Figure 4.9: ECU and shutdown header

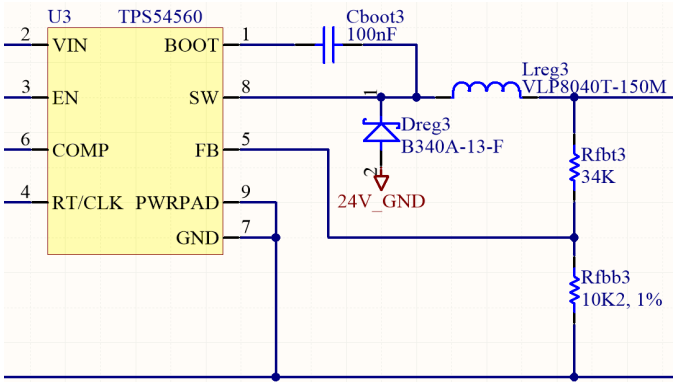


Figure 4.8: 3.3V regulator

controller to one of the buses. The transceivers are placed as close to the header as possible, so that the total stub length in the car is kept short, without forcing placement of termination. If the stub lengths are too long, reflections at the end of the stub may occur, corrupting data.

The AMK inverter requires four enable signals to function properly. These have been implemented on the R16 inverter as well, so that the ECU and torque vectoring systems have an identical interface to the inverter, regardless of which is currently situated in the car. The signals are transmitted at 24V, so they are scaled down to the motor controller's 3.3V input level through an optocoupler. In addition, the car's shutdown circuit is taken in to the board through the same header.

Isolation

According to the FSAE rules[16], EV4.1.7 all high-voltage areas must be sufficiently separated from low-voltage areas. As the control board includes the measuring circuit of the DC bus, this section of the board has been separated with a $\tilde{6}$mm separation barrier in all layers of the PCB. In addition, high frequency optocouplers have been used to isolate the gate driver from the control board.

Encoder interface

The controller needs to know the position of the rotor in order to do correct calculations. The encoder mounted in the motors communicates on RS485 with differential lines, so a set of SP3485 half-duplex Transcievers[9] are placed between the header

and the MAX10 containing the encoder interface IP. This is teh required physical interface of the encoders, and the required interface software implemented on the MAX10 is further described in section 4.2.

Safety systems

There are several sensor measurings and safety features required for the inverter to operate properly. The control card thus needs to provide a well-functioning interface for these. The connection to the sensors themselves are done via the gate driver header, but some manipulation of the signals are needed on the board. The temperature sensors are NTC or PTC sensors, and the second resistor in the voltage divider is placed on the control board.

The current sensors work at 5V, but the Zynq and Atsam have a maximum ADC reading value of 0.5V and 3.3V, respectively. The output signal from the sensors are scaled through an INA337[10], so that the sensor voltage level does not go higher than what the control unit can handle. The formula for this conversion is $G = \frac{2R_2}{R_1}$. The sensors output a maximum of 5V, and with $R_2 = 127\text{k}\Omega$ and $R_1 = 400\text{k}\Omega$, the level does not go beyond the Atsam's limit. By changing the values to $R_2 = 20\text{k}\Omega$ and $R_1 = 400\text{k}\Omega$, the gain will be 0.1, keeping within the Zynq's limits. The overcurrent signal from the sensors is input directly to the Atmega (running at 5V), and put through a voltage divider to 3.3V before reaching the control insert. An Atmega processor is implemented on the board to offload some processing tasks from the Atsam processors on the insert. It reads temperature sensors, DC voltage and overcurrent indicators from the sensors, and reports logging data over the CAN bus. The circuitry around this chip is fairly simple, and the code is presented in section 5.2

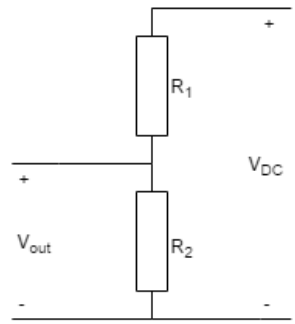


Figure 4.10: Voltage divider

Voltage measurement

The voltage measurement is done with a voltage divider over the DC bus, scaling it down to an acceptable level. As the input limit on the Atsam is 3.3V and the accumulator delivers voltages in the 600V range, a divider such as the one in Figure 4.10 with $R_1 = 1200k\Omega$ and $R_1 = 3.9k\Omega$ gives the ability to measure spikes

up to about 1kV, which should cover the irregularities from the accumulator with a good margin.

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2} \tag{4.2}$$

$$V_{in} = V_{out} \cdot \frac{R_1 + R_2}{R_2} = 3.3\text{V} \cdot \frac{1203900}{3900} \approx 1031V \tag{4.3}$$

After the voltage divider, an optical isolation amplifier from Avago, ACPL-C87[19] is used. This outputs a differential voltage signal, so an INA826[11] is used to transform it to a single-ended voltage level. Since scaling happens in the voltage divider, no further gain is needed on the instrumentation amplifier.

4.3.3 Insert

In addition to the ZX5 module, a backup solution was required, should the development of the Zynq code be slower than expected or fail. This backup came in the form of an in-house designed insert based on the Atmel SAMe70 microcontroller, used in systems troughout the car.

In order for the control board to be compatible with both the ZX5 module and the insert, the latter was designed around the same headers and footprint as the ZX5. The ZX5 features several physical interfaces that is not required for the inverter to work, such as SDRAM and ethernet transceivers, so these were left out of the Atsam insert, in order to keep it as simple as possible. The insert contains two microcontrollers and all circuitry required for these to work properly. USB-USART bridges are added to mirror the USB physical on the ZX5, and Mini-USB headers placed to be able to use the insert without the carrier card, for development purposes. This feature is further backed by a 3.3V regulator connected to one of the USB headers and a set of four indicator LEDs per microcontroller.

4.3.4 Issues and experiences

During the development of the control board and the insert some issues surfaced that required redesign of the card. The authors had misinterpreted which type of encoder was mounted in the motors, taking for granted that it was the same type

as in the 2015 car, and copying those schematics. As a closer reading of the (rather cryptic) motor documentation, it became clear that the EnDat IP was needed, resulting in a redesign from scratch, implementing the MAX10.

Since the insert is designed to replace the Mercury ZX5, and custom made for this carrier card, any change made in the carrier card will most likely force a change in the insert as well. In its current state, inserting the ZX5 module will only allow the Zynq to interact with one inverter, due to the pins used on the header. The traces that connects to the first encoder IP placed on the MAX10 is placed on the ethernet pins on the ZX5 header, which is not directly connected to the Zynq chip, but rather to the ethernet physial on the module. However, when the ZX5 is developed, the MAX10 will not be necessary on the carrier, and the board will need a redesign. In any redesign of the carrier, more care should be taken with regards to choosing ZX5-compatible pins used when connecting to the insert, and the insert designed from there.

4.4 Power card

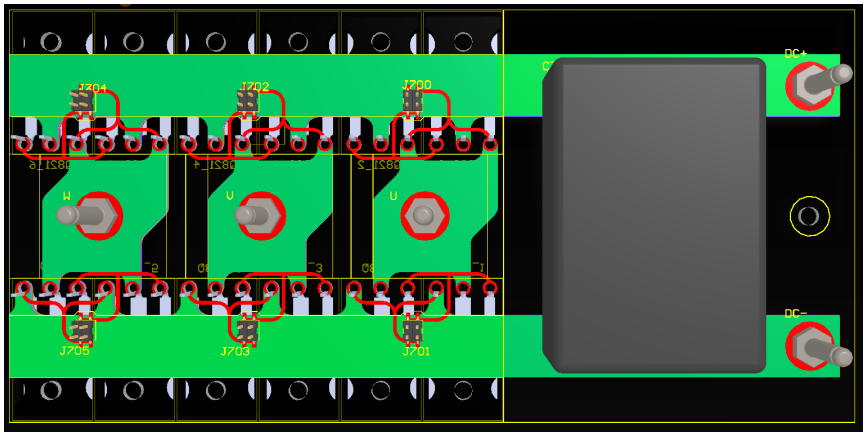


Figure 4.11: The final Power PCB design

The power PCB carries all high power components in the inverter. It is responsible for connecting switching transistors, decoupling capacitors and gate driver signals together and carrying the high AC and DC currents required by the motor. This is where the power and current requirements of the inverter is fulfilled.

Stackup	Voltage drop [mV]	Dissipated power [W]
4l 35μm	49.7	0.997
4l 70μm	35.2	0.704
6l 35μm	36.5	0.730
6l 70μm	28.5	0.570
1l 350μm	20.5	0.410

Table 4.4: Voltage drop and power loss in DC+ rail at 20A

4.4.1 Requirements

When designing for a given current draw, trace width and copper thickness are design parameters. Increasing conductor cross section will lower conductor resistance according to Equation 4.4. Copper thickness is usually denoted by weight per area in $\frac{\text{Oz}}{\text{mil}^2}$ abbreviated Oz, or thickness in μm. Ordinarily, circuit boards use 35 μm copper weight on inner layers, and 17.5μm on outer layers. In the final inverter, the Power board and transistors should be able to withstand both the continuous and peak load without failing.

4.4.2 Simulation

In the prestudy[20], the Power PCB design was simulated and evaluated in Mentor Graphics Hyperlynx. This program presents a 2D map of voltage drops and current density throughout the PCB. It was decided that while all tested board designs would satisfy the design constraints, the 6 layer 70μm would be the final design choice, as it was the most robust design available from Revolve's regular manufacturer.

$$R = \frac{\rho * Length}{Width * Thickness} * (1 + (\alpha * (T - 25\,^{\circ}\text{C})) \tag{4.4}$$

$\rho =$ Resistivity

$\alpha =$ Thermal coefficient of resistivity

The results from the prestudy were as follows. Different PCB stackups were simulated at 11kW load to find an acceptable copper thickness and layer count. The simulation is done only for the DC+ rail, and as such the power dissipation must be multiplied by 3 to get the dissipation of the whole PCB.

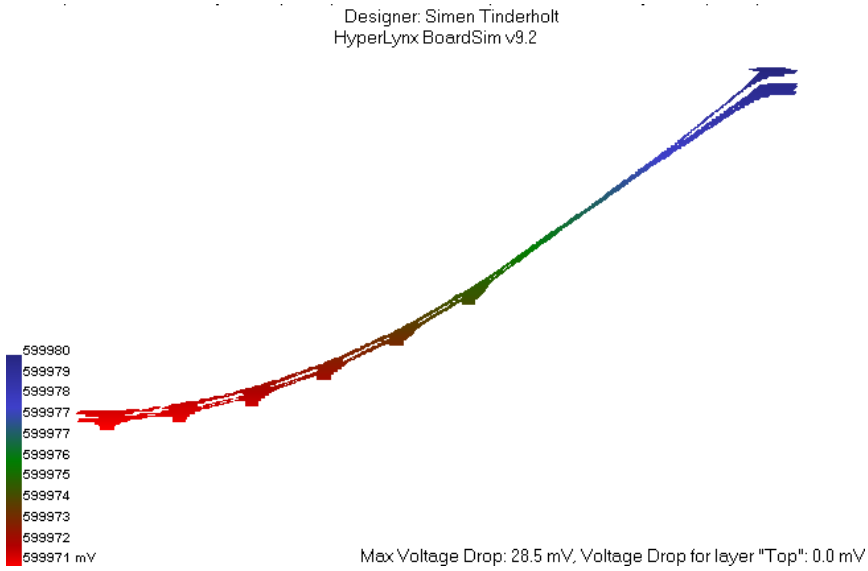


Figure 4.12: Voltage drop in DC+ bus with six layers of 2Oz copper weight.

Because of the cost of producing non-standard PCB thicknesses our PCB sponsors NCAB and Simpro could provide PCBs with at most 70μm inner layer thickness. Multilayer solutions were therefore simulated to test thicker copper stackups. Every stackup is tested at 20A DC, or 11kW at the nominal 550V battery voltage, the simulation results are listed in Table 4.4.

Usually when using multiple layers to transfer power, via stitching is used to distribute the power evenly between the layers and components. However, all components in the high current path were through-hole and had equal access to the different layers. It was therefore considered unnecessary. One alternative applying one 350μm layer was also tested, performing well in simulations. Unfortunately, this production method set a maximum trace thickness of 12mm, which made connecting the parallel AC switches difficult. While all copper weights have acceptable copper losses at 11kW, 6l 70μm was chosen, to allow some head room. This gives a total copper loss in the power board of about 1.71W.

Since the inverters should sit on both sides of the cooler block, the power PCB is produced as two complementary PCBs, were one has the DC connection and capacitor moved to the opposite side of the AC output stage. This way, the inverter systems is mirrored about the cooling block, which makes the cooling block shorter and casing smaller while simplifying system assembly.

To assure that no wires disconnect during driving due to loose nuts or connectors, positive locking connectors were used to connect AC and DC systems. In addition, the power PCBs will be bolted to the cooling block and secured with locking wire. To minimize inductance and board complexity, the phase connectors are placed in the middle of the board.

4.4.3 Experiences and issues

Because of the massive amount of copper in the PCB traces, soldering the power PCB's through-hole components has proven trying. Ordinary soldering irons do not have power output to heat the PCB to the required temperatures. To solve this problem a reflow oven was used. If preheated to $180°C$ the transistors and capacitor can be hand-soldered by an ordinary soldering iron. This could also be solved by using a heat-plate. When producing more than one PCB a wave-soldering machine should be considered, as the process of preheating and soldering works, but still has some quaity issues. During testing the Power PCB has worked well.

Through working with the system, and discussing the system with Kjell Ljøkelsøy, some points of concern have appeared. In any further work on the inverter, these should be adressed. Even though it is our belief that the Power PCB should function well in an in-car application.

Decoupling Using a single decoupling capacitor, and placing it on one end of the PCB may impair performance. The long leads from capacitor to switching elements, and the varying length to each of the switching stages will worsen the effect of the capacitor as a low-latency source of energy. Using several smaller capacitors placed interspersed on the PCB would be better.

Transistor orientation Bending the transistor legs to align them with the cooling block will allow current transients to create significant voltage drops across the length of the leg. These spikes increase power dissipation, and impair performance. In addition they may overcome the dielectric strength of the transistor and destroy it. Allowing the transistors to be fully seated will shorten the lead length, and present a wider lead, mitigating the problem.

Transistor hole placement To simplify the design of the PCB, the transistors are connected through copper peninsulas, and not directly to the main conductors. This allows one conductor to occupy all board layers. However it presents a higher inductance path to the transistor.

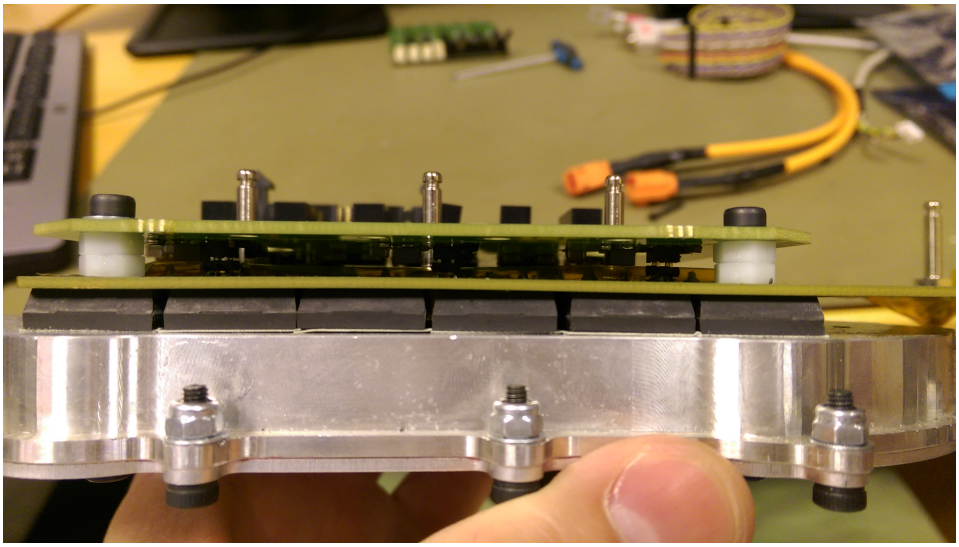


Figure 4.13: Power Board mounted on the heat sink, transistors in between.

Board length The unsupported length of the PCB allows it to flex more than anticipated. Especially when supporting the rather heavy capacitor.

Transistor mounting pressure To allow higher water flow through the cooling block, only two of the 6 transistors on each side are retained against the block, with the PCB exerting pressure on the rest of the transistors. This does not guarantee sufficient pressure on the switch to ensure a good thermal interface.

4.4.4 Future work

Many of these design flaws, specifically the board length, peninsula and orientation issues, arose from the early design freeze of the board and cooler system. If both were redesigned the issues could be solved. By placing the power board on the side of the cooling block the transistors could be stood upright on the PCB. The simulation results also suggests that at $70\mu m$ a full 6-layer PCB may not be neccessary. The highest power dissipation Table 4.4 is a total of $3W$. Compared to the predicted power loss of $500W$ in the transistors, this will provide a negligible heating of the PCB. If the design is rearranged, two layers can be used for DC conductors and three for AC phases, leaving one for control signal routing. If $70\mu m$ PCBs are used, power loss comparable to or better than the $4L35\mu m$ de-

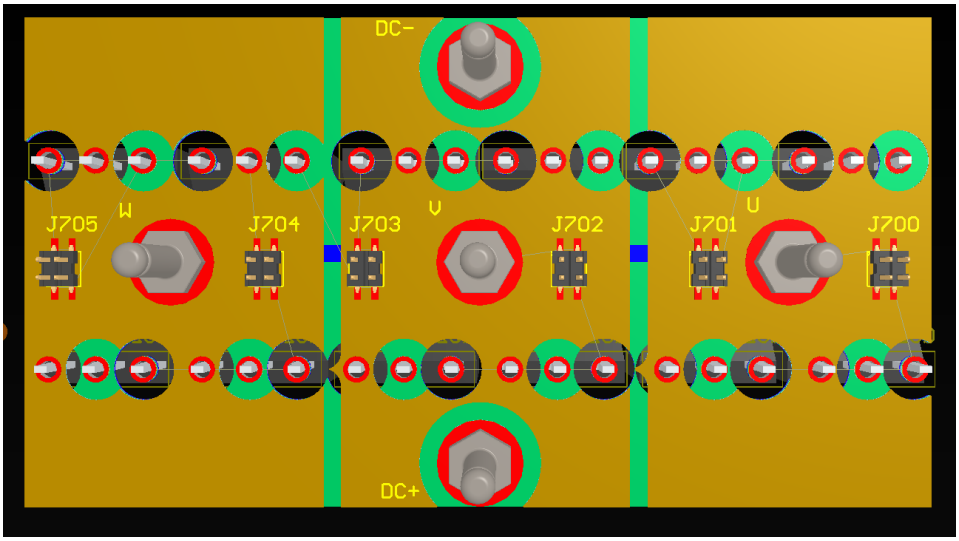


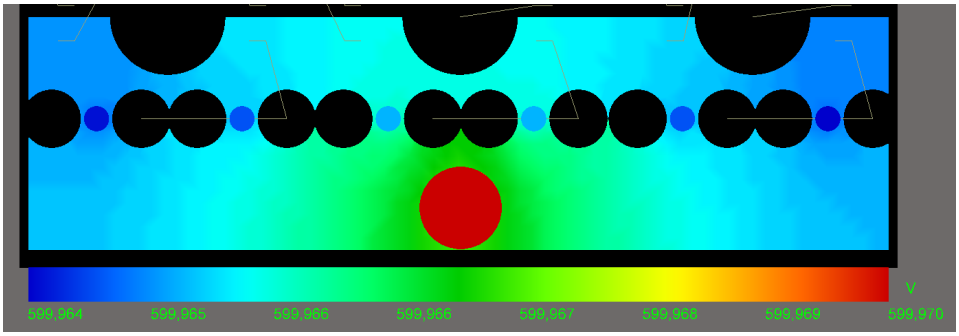
Figure 4.14: 3D image of revised Power PCB showing three phase conductors on top, and lower layers with DC conductors beneath.

sign should be possible. A rough mockup of the revised design, using the already existing schematic can be seen in Figure 4.14.

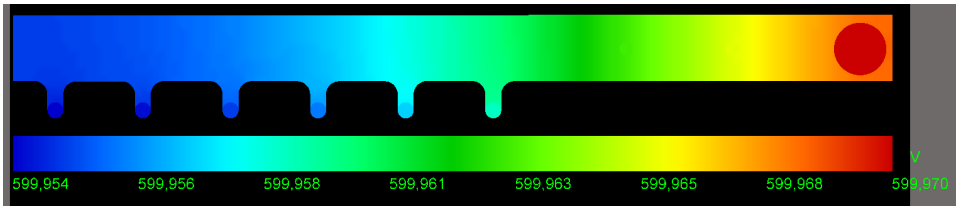
To verify the new design, new simulations were done. The recently released Altium Power Distribution Network (PDN) analysis tool was used for these new simulations. Hyperlynx is a good software suite, but importing a PCB design file from Altium Designer can be tedious, so the closer and simpler alternative was chosen. Simulation results with the new design show promising results, with significantly lower voltage drops. The new simulations are done with a more realistic 18kW load, as the 11kW load simulated earlier is lower than the desired continuous rating. As the simulations are done in a new software, an additional 11kW load simulation was done to allow comparison to the Hyperlynx results.

The reference simulation in Figure 4.15c shows a maximum drop of $31mV$. This simulation result is worse, but close to the original $28.5mV$ drop, so it can be assumed that the Altium PDN package will produce comparable results to Hyperlynx.

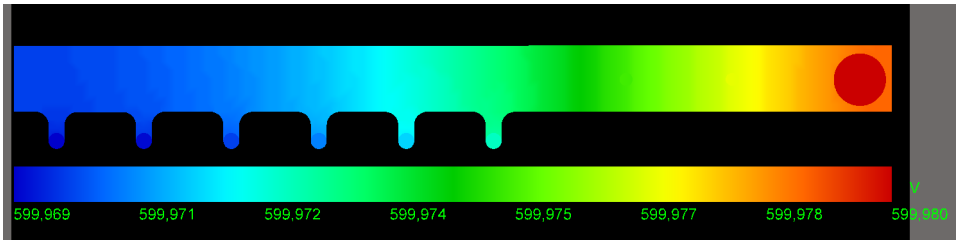
As seen when comparing the lowest scale value in Figure 4.15a and Figure 4.15b the new design is more effective than the old, with $10mV$ less voltage drop to the farthest transistor, and a much more even distribution of the switch-to-DC conductor length.



(a) New design at 18kW load evenly distributed over all phases.
Lowest value on scale is lowest value overall.



(b) Original design at 18kW load evenly distributed over all phases.
Lowest value on scale is lowest value overall.



(c) Original design at 11kW load evenly distributed over all phases.
Lowest value on scale is lowest value overall.

Figure 4.15: Simulation results from Altium PDN analysis tool

This new design also connects the conductors directly to the transistor leads, reducing the path inductance. By redesigning the cooling block with separate mounting bolts for each transistor, proper mounting pressure can be achieved for each transistor. Some thought has to be put into capacitor placement, but it should be possible to place film capacitors with sufficient voltage rating in between the phase connectors.

4.5 Gate driver card

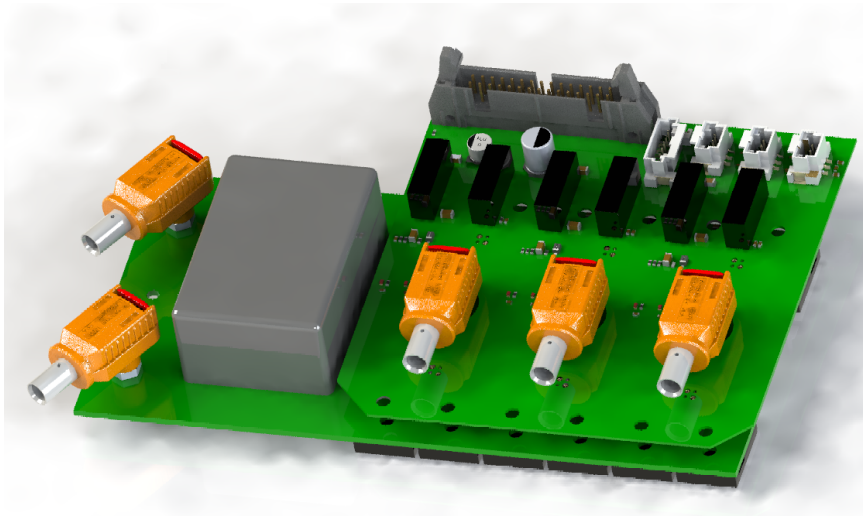


Figure 4.16: Gate driver and Power PCB assembly in Solidworks.

4.5.1 Specifications

The switching stage consist of six transistor pairs, connected in a three-phase two level inverter-setup. According to the transistor data sheet, the appropriate gate control voltage levels are $+15V/-3V$ in reference to the transistor source pin. As seen in Figure 4.17, the amount of current passing through the transistor increases the amount of energy required to turn it off. An ordinary processor output pin or optocoupler will ordinarily be able to sink or source less than $30mA$ continuously. To allow fast switching of the transistors, and avoid transient states where the power stage is halfway on, a gate buffer must be implemented. These systems are called gate drivers. Designing a single gate driver board for each inverter will

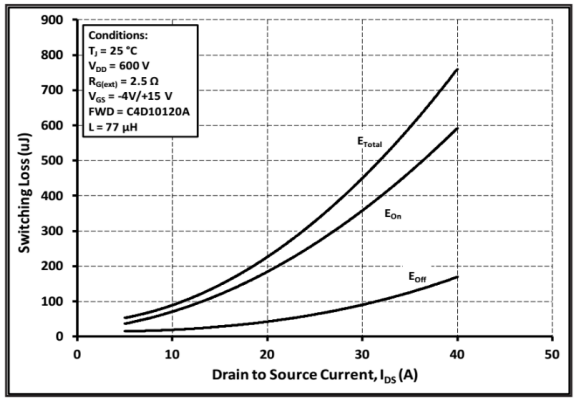


Figure 4.17: Clamped Inductive Switching Energy vs. Drain Current ($V_{DD} = 600V$) from [22]

simplify assembly and design work. One board should therefore control 6 transistor pairs.

To further ease assembly and reduced cable clutter, each driver board should also have connections for temperature sensors and current sensors. This allows all signals for a single inverter to be passed through a single cable harness without additional connectors. The gate driver board is designed to stack on top of the power PCB. It must therefore accommodate holes for the AC phase connectors that poke through it. This represents a design challenge, as it is crucial that any holes in the gate driver perfectly matches the power PCB. However this will facilitate ease of assembly and ensure easy access to the phase connectors.

When designing the gate driver PCB, it is crucial to be conscious of voltage ratings, as this is the dividing line between the low voltage controller section and the high voltage power section. Care has therefore been taken to maintain spacing between separate voltage sections of the card, and the card will be conformally coated after manufacturing. In the design, the FSAE clearance rules stated in EV4.1.7, will be guiding. Additionally, when choosing component for the isolation barriers, such as DC/DC converters and gate driver ICs, the voltage rating should be based on continues tests, and not HI-pot tests. In any single component a voltage rating of at least 3000V is recommended.

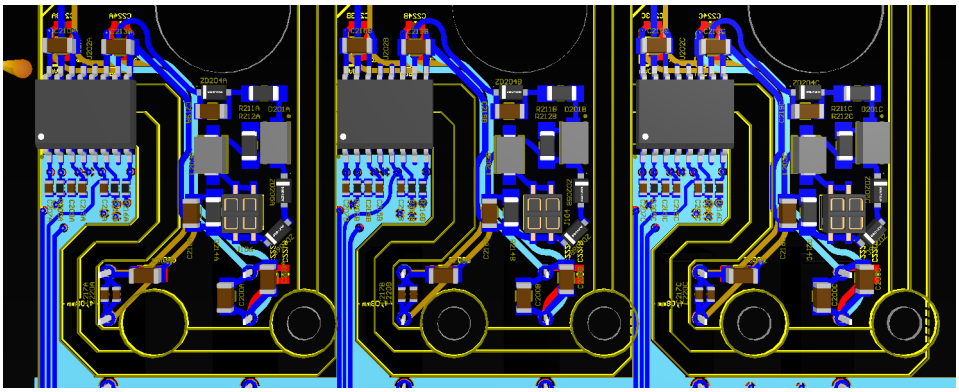


Figure 4.18: Gate driver 4mm isolation zones marked in silk.

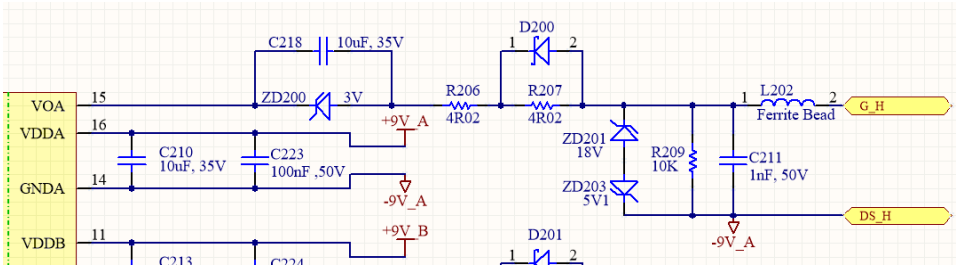


Figure 4.19: Gate driver output filter

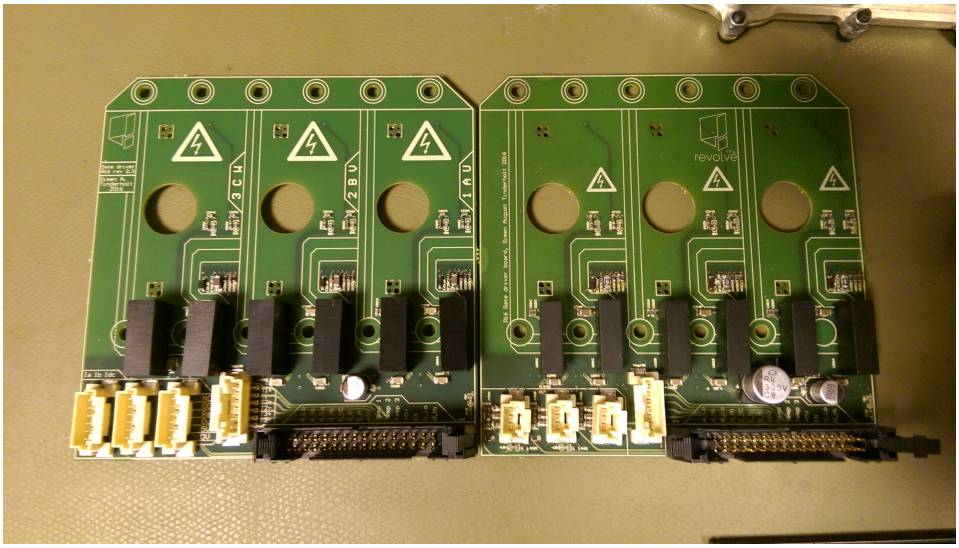


Figure 4.20: Gate driver card, first (right) and last (left) revision.

4.5.2 Design

Since we have never before designed a gate driver circuit it was decided to base the schematic on a manufacturer application note[5] with modifications made to accomodate the newer 900V-rated transistors[6]. This application note, together with its addendum, describes a gate driver for a Cree third generation silicon carbide MOSFET, based on a Si8233 driver IC. This chip features 3000V isolation variable gate power supply, and 15 MHz bandwidth, making it ideal for this application. The application note suggests the filter circuit shown in Figure 4.19. When supplied with an 18V supply across VDDA-GNDA, the output VOA swings from 0-18V. The zener diode ZD200 forces a -3V drop across capacitor C218 resulting in a 15V output voltage. When the output voltage drops to 0V, the C218 maintains the -3V drop, pulling energy from the transistor Gate pin. To suppress any voltage or current spikes due to the fast switching waveforms, a weak LC-filter is applied near the end of the circuit by L202 and C211.

While the instantaneous gate current of max 4A is more than can be expected to sustained from any DC/DC converter, the current is only sustained for less than $10ns$ every time the transistor switches state. At a 25kHz switching frequency, the continuous current spply required, given appropriate decoupling, is $2 * 4A * 10ns * 25kHz = 2mA$, as such, a 1W converter is more than enough, with $55mA$ continuous rating at 18V.

Several DC/DC converters were considered and both the Murata and Recom solutions were tested. The Traco Power solution provides a higher isolation value, but the two others are cheaper, and have sufficient isolation. Care was taken to choose devices based on 1 minute isolation tests rather than Hi-Pot tests, as the devices will be subject to the rated voltage over a continuous period. Because we got a sponsorship agreement with Recom to supply the RH-0509D, this unit was chosen

	Murata MEV1D0509SC	Recom RH-0509D	Traco Power TMV 0509DHI
Isolation (1 min)	Flash tested	1500Vac	5700Vdc
Isolation (1 sec)	3000Vdc	3000Vdc	5200Vdc
Voltage Input	5V	5V	5V
Voltage Output	+/−9V	+/−9V	+/−9V
Power rating	1W	1W	1W
Package	SIP7	SIP7	SIP7

Table 4.5: DC/DC Converter data

for the final product.

To ease design work, and insure proper isolation, the PCB has been divided to four separate sections. Three identical high voltage sections and one low voltage section. A four mm keep out zone is maintained between each section to ensure high voltage isolation. Each high voltage zone services two transistor pairs. To ensure correct timing when switching the transistors, the trace length of channels has been matched and the filter circuit has been designed as compact as possible while maintaining manufacturability.

4.5.3 Issues and Experiences

Some issues have appeared during testing ad development of this card. When designing the first prototypes of the card, the zener clamping circuit (ZD201 and ZD203) was drawn with 15V and 3V zener diodes. These therefore had a very high chance of burning up during testing, as the voltage slightly exceeded +15V or -3V. This lead to short across the filter, leading to a very high current running through the gate resistors and zener regulator (ZD200 through R207). At one point the gate 1206-size resistors desoldered themselves. The error was isolated to the diode clamper circuit during testing, as replacement of these parts would momentarily solve the issue, but the fact that the replaced parts had the wrong zener voltage value would not be detected until the final redesign. The final schematic revision has the correct zener diodes listed. Subsequent testing has been without issues.

To limit the current loop area of the high frequency signals optocouplers were used. An error in the schematics component for the part lead to the voltage supplies pins being crossed on one end of the chip, shorting the 5V rail. In addition, the output stage was open collector, not open emitter as first assumed, leaving the phase controls floating. These errors were corrected on the prototype by cutting the crossed voltage supply (as seen in Figure 4.21, the optocouplers has a double set of voltage supply pins), and adding pull-ups to the output. Because a fault in the Control Card to Gate Driver wiring could leave the switches closed, the pull-up resistors were seen as a security liability. In the final revision, the optocouplers on the gate driver board were scrapped in favor of optocouplers and 10k pull-ups on the Control Card, and weak 47k pull-downs on the Gate Driver. This results in a 4.1V input to the gate driver when a 5V signal source is used. As the Si8233 is designed for 3.3V-5V signaling, the voltage drop is no problem. To supply the

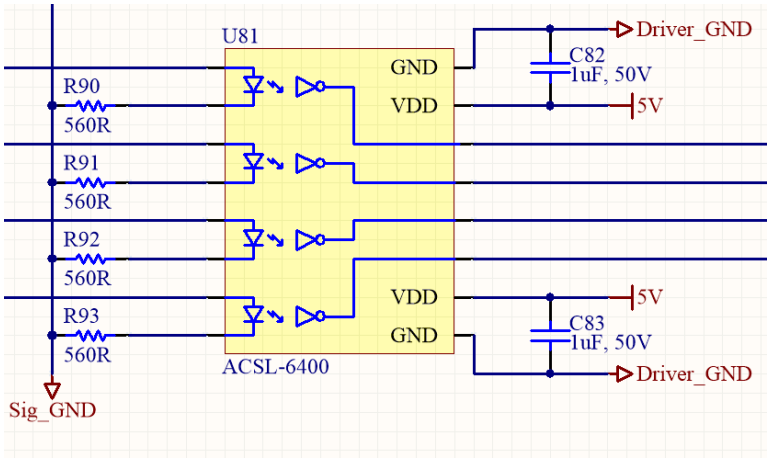


Figure 4.21: Optocoupler schematic drawing

first choice of current sensors (LEM LA 150-p) a 5V to +/−15V DC/DC converter was included in the first revision. To save space on the card and remove a possible error source, a new current sensor with 5V single-ended supply was chosen.

4.5.4 Future work

The gate driver circuit is designed to closely match the power PCB form factor. As such, any change in the power PCB must also trigger a change in the gate driver circuit design. As it is, the gate driver works and allows easy assembly of the inverter. However, it would be possible to compress it significantly if a vertical design would be implemented. This would however come at the cost of assembly time and accessibility. For a prototype system this would not be recommended, but once a working system is perfected, it might be viable.

4.5.5 Casing design

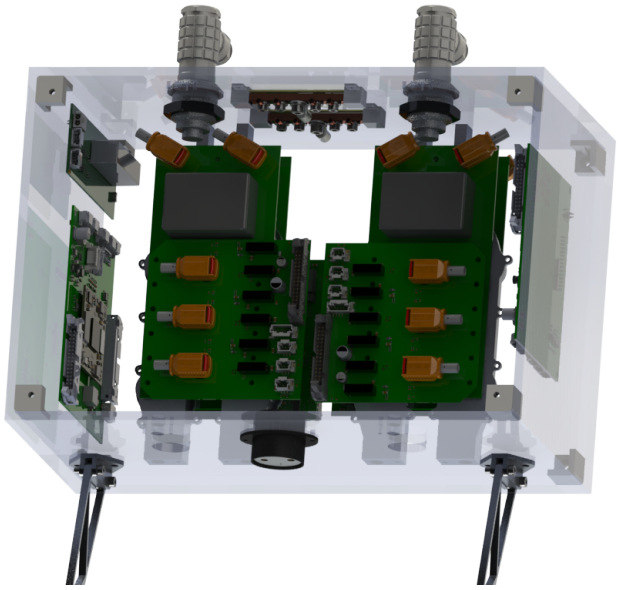


Figure 4.22: The VSIs mounted inside the casing

Four inverters take up a lot of space and weight if placed separately around the car. There are however several ways to make this compact and lightweight. Mounting the inverters on the side of the vehicle was considered early in the design process. This would allow the car to separate the cooling of the left and right side power systems, and save some weight in cooling tubes. However,the rules demand that the sides of the car be crash zones. Making a side impact structure outside the inverters would have added a lot of unnecessary weight. The chosen concept was one were all four inverters shared the same enclosure. This has the benefit of being the concept with the least amount of extra DC wiring, and allows the two pairs of inverters to share some systems and connectors.

The final design is a carbon fibre box with a volume of 15.8dm^3, and a weight including connectors of 1,350g. To save weight, the system has two independent heat sinks. The power stages are mounted on both sides of the heat sinks, with each cooler accommodating two inverters. Each control card regulates two power stages, they are therefore placed on the side walls, controlling one heat sink each. To make system maintenance easy, component access has been key. The lids both

on the top and bottom of the inverter are detachable, making the hardware easily accessible. To ease disassembly all high voltage wires have snap-lock connectors to the power boards, and control cards are easily disconnected with a single cable.

4.6 System on Chip

A System on Chip - or a SoC - is a chip which contains both a processor, programmable logic and other elements such as ADCs and memory. A SoC is usually divided in two "sides": processing system (PS) side and programmable logic (PL) side. This combination of technologies enable the user to program the chip both with regular application code on the PS side, as well as hardware programming on the PL side, getting the best from both worlds. The ability to utilize both the processor and logic also makes for a very powerful system with fast data processing.

In this project a ZynQ 7000-series[23] from Xilinx has been used, featuring a dual-core ARM Cortex A9 processor and several peripherals. Implementing the motor control system on a SoC will enable parts of the control to happen in parallel, running through the control loop, reading sensor data and performing safety checks all at the same time. In addition, using a combined system such as this will enable the chip to run control system for two motors at the same time, reducing the amount of hardware needed.

4.6.1 System on Modules

As most SoCs come in ball-grid array packages, which are very complex to design with and solder on a PCB, the design of a control system PCB with a single SoC would be too complicated. This is where a system on module (SoM) comes in. A System-on-Module is a type of single-board computer, extending the one-chip principles of a SoC to a board. They typically include a processing unit, power regulators, communication interfaces and memory, so that the system works without anything else. These finished modules usually comes with board-to-board connectors, so that they can be inserted into a carrier card. This enables us to create the carrier card, and not worry about the complexities of ball-grid array soldering.

There is a fairly large amount of SoMs using the Zynq chip as their core. In this

project Avnet's MicroZed[3] and PicoZed[4] were in focus, and have been used for development. Unfortunately, SoMs are fairly expensive, and a sponsorship deal with Avnet did not come through. The focus was then shifted towards Enclustra's Mercury ZX5 module[8], and this footprint was used for the final implementation.

4.6.2 Using a SoC in the control system

Each element of the motor control system can be implemented on each side of the SoC. Using the programmable logic can be beneficial for a hard real-time implementation of the control algorithm, resulting in a more stable execution. However, the control loop would have to be written in a hardware description language (HDL), which can be a challenge if one are not familiar with such. Xilinx provides a program with Vivado, called Vivado HLS, which lets the user to write the code in C or C++, and automatically translate into RTL code targeted on the specific device. Implementing the control loop on the PL side frees the PS side up to handle communication with the car, and possibly safety measures. The Zynq system is powerful enough to run a Linux operating system in the background of the PS, so that real-time capabilities easily can be added to the application.

An alternative is to use the PS side for the control application and communication, and implement safety systems in the PL. The safety systems can either be implemented as a hardware-module, or in a soft-core processor running on the PL side of the chip. Having the control system in the PS would result in an easier implementation of the control loop, but possibly a lower performance due to timing issues, especially at high running frequencies.

4.6.3 Manufacturer choice

During the concept phase of the project, various control system hardware was weighed. The choice ended on a System on Chip, with regular microcontrollers as a backup system. In the FPGA world, Xilinx and Altera are the two largest proucers of FPGAs and SoCs. Both deliver a great variety of products, with several tools for developing application and hardare platforms. The choice initially fell on using Xilinx' due to more familiarity with the tools. Avnet Silica is a company supplying several development boards with Xilinx chips, and after a meeting with their local engineer, the choice fell on Xilinx' Zynq 7000-series SoC and the MicroZed

development board. Xilinx provides a powerful tool for programming applications on their platforms called SDSoC, which includes the opportunity to automatically generate hardware accelerators from one or more functions of the application.

In addition to the Zynq-chip, Altera's MAX10 FPGA have been used to implement the Encoder communication system. The choice for this system fell on Altera because of a sposorship deal with Arrow, a company supplying development boards based on Altera chips. This choice was also made after some work with Xilinx' development tools, and bad experience with the possibility to get help on problems for these tools. It became clear that Altera's suppport system is easier to work with, and their tools are more intuitive. Our experience with the different tools are discussed later in this paragraph.

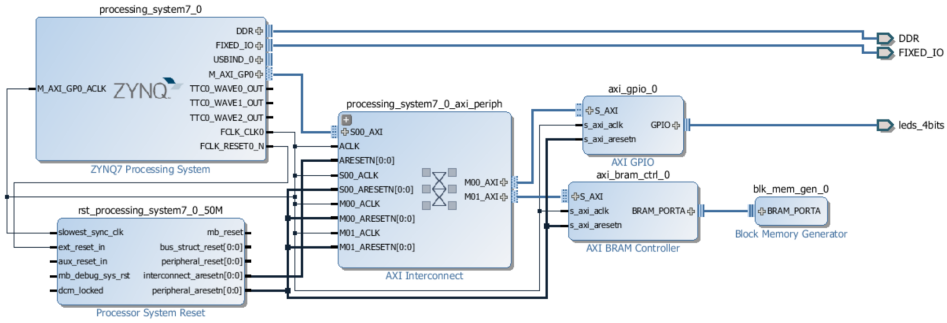
4.6.4 A comparison of tools

Suring the project, both Xilinx' and Altera's development tools have been used. While they have been targeted at different architectures, the work flow of the systems are fairly similar. You start by putting together your hardware in Quartus(Altera) or Vivado(Xilinx), incluing any IP-blocks, interconnects and HDL code you need. This can be done in several ways, wither by writing HDL code or creating a block diagram, or in the case of Quartus, a connection diagram called Qsys can be utilized. When creating the necessary hardware platforms for our systems, the chosen method was block diagrams. Figure Figure 4.23 shows the block diagram style in each program, in Vivado and Quartus.

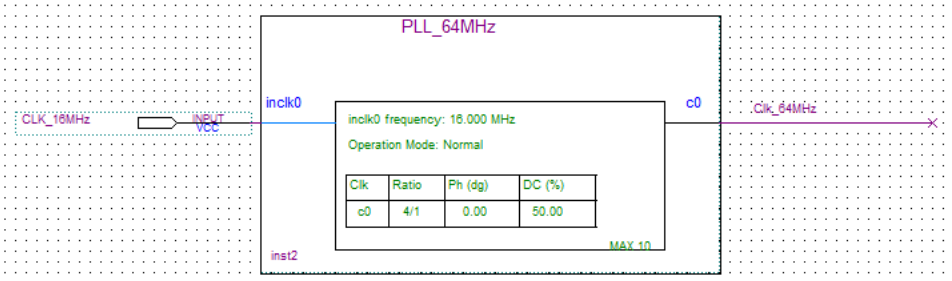
Once the hardware platform is complete, it is loaded onto the chip. If the system is only running on the programmable logic of the chip, the process is done, but if the system contains a processor (either hard- or soft-core), a board support package(BSP) will be needed to tell the processor what hardware elements are available to it, and in what way it may communicate and utilize them. The BSP contains drivers and header files for the IPs included in the hardware, so that they can be used in a straight-forward fashion in your application. Once the application is written, it can be uploaded with SDK/SDSoC (Xilinx) or Nios IDE (Altera).

Xilinx' SDSoC tool is potentially very powerful, with the ability to analyse your application code and recommend functions and parts of the code to implement as HW accelerators, and automatically create these accelerators for the parts you want. However, the learning curve is very steep when you don't have a lot of

experience with their tools, and starting with such an advanced tool is intimidating. While there are guides on using SDSoC and Vivado with a Zynq system, they are hard to come by, and not very easy to follow, as they are cross-referencing each other a lot, making it hard to keep track of the steps. It might have been easier to start off using SDK, as it is a less complex tool with more usage throughout the community.



(a)



(b)

Figure 4.23: Block diagram style in Vivado (a) and Quartus (b)

Altera's tool Quartus works mostly the same way as Xilinx', but with access to better written and more detailed guides it is easier to learn. The workshop lab guides we got from Arrow does a great job explaining what happens and why it is done in each step, making them easy to follow. While we have used Quartus for a shorter time than SDSoC, it has been a faster development cycle for the MAX10 than the Zynq.

In retrospect, the decision to go for Xilinx' system may have been done rather rash, with too much emphasis on the author's experience with somewhat outdated tools and platforms. Over the course of this project, the main problem the authors have experienced with development is accessibility of support. That, coupled with

the lack of guides getting to know SDSoC, proved a challenging experience. SDK may have been a better start point, as it is more widely used, and thus easier to troubleshoot. Altera's products have been easier to work with, partially because the MAX10 platform is a simpler product, but in no small part due to great availability and quick responses from our contacts in Arrow.

5. Software Design

5.1 Atsam code

The Atsam controller code is the most extensive code written during this project. It leans heavily on the code and experiences made during the work on last years inverter. Care has been taken to develop it according to good code practices. Circumstances and time have however conspired to force some shortcuts and bad choices to be made.

Each processor should control one inverter. This generalizes code design, and minimizes testing time. Each processor then serves four purposes: speed control, torque control, error handling and status transmission. Error handling of slow signals, such as temperature and voltage, is handled by an external processor. Therefore the Atsam only needs to handle internal control failures, such as overcurrents and supply overvoltage. The following requirements are imposed on the controller code base:

- The program must control the inverter power stage with FOC to develop the torque requested through CAN.
- Security checks should be done fast enough to catch and remedy any threshold violations before damage is done.
- In addition to essential control tasks, the inverter software should allow controller parameter adjustments and direct test bench control.
- It is crucial that any additional features operate in a manner that does not interfere with the control loop.
- Any drivers or modules designed for the inverter should be easily readable.
- Any driver or module written for the inverter should have a single, well defined purpose, and remain clearly constrained to that purpose.

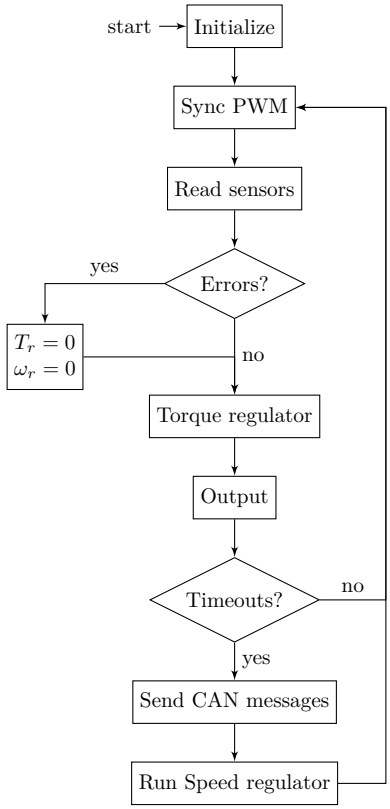


Figure 5.1: Flow chart of the Atsam processor program

Task Name	Priority	Frequency
Torque Control	4	>8kHz
Speed Control	3	100Hz
Data transmission	2	300Hz
Alive Transmission	1	1Hz
Status Transmission	1	10Hz

Table 5.1: Inverter Control Tasks, sorted by decreasing priority.

5.1.1 Flow Control and OS

In any real time system it is important to consider how the system choses which tasks are performed next. Bad handling of task selecion and timing requirements can lead to several real time problems, most notably priority inversion and timing constraint violations. When low priority jobs like communication and low-frequency controllers are run instead of a more important duty, the system suffers from priority inversion. This can lead to issues with control flow timing, or in bad cases cause time constraint violations. If a task that needs to run at a specified frequency misses a period, or is delayed so much that its output is not valid, even though it comes before its next run time, a timing constraint violation has occured. The inverter system has five tasks with varying priorities and time frames, listed in Table 5.1.

Based on last year's experiences, an operating system was never considered for the Atsam solution. To be able to control the motor currents effectively, the control loop would have to run faster than 8kHz. As this will push the processing power of the processor, little to no processing power would be available to run the scheduler. As an example, the maximum scheduling period of the FreeRTOS scheduler, used in other systems of the car, is 1ms. This means that if any other tasks were running than the controller, the controller would fail to compute at least 8 control steps every time it lost arbitration.

As implementing a premade OS in this system is not possible, the remaining architeture options are big-while, and interrupt-based code. In big-while programs, code execution is governed by a big while loop, either running continuously, running code based on events, or the whole loop can be synchronized to some external signal. The free-running big while loop is perfect and simple when the processor is significantly more powerfull than required, ensuring that the time the whole loop takes to execute is short enough that the timing of the executed tasks is not influ-

enced. Based on experiences from the Atsam4E16E, this is not the case. The 2015 inverter was rigt on the edge of what the processor could handle, with an additional floating point operation added to allow power limiting leading to the control loop violating its timing constraints.

The more efficient way of handling flow control is the synchronized big while loop. In this system, the big-while runs one loop, and then waits for some process to finish before running the next round. It is still assumed that the processor has some amount of processing overhead, but as long as all delays are accounted for, this overhead can be quite small. This system is perfect for small control systems with little outside interaction. As long as the variation in the execution time of the while loop is limited, this method can be more efficient than the interrupt based scheduling. If there are significant variations in execution time however, for example due to slower-running regulators, communication or other actions done less frequently than the primary action, the synchronizing process period must be longer than the longest possible loop length, making this method inefficient.

In an interrupt-based system, the main loop contains non-time-critical code, with all time-sensitive behaviour governed by an interrupt routine. This programming method can guarantee correct control timing, and little period swing, but in cases where the processor is very pressed may be slower than a big while loop because of the context switch required when executing an interrupt routine. This switch operation can take more than 20 clock cycles. At control frequencies closing on 25kHz or higher, when one cycle is at most 12 000 cycles, this can be the difference between a working or non-working program. According to an ARM guide on the NVIC interrupt handler in Cortex-M processors, on the Cortex-M7 or Cortex-M4 architectures however, the context switch is hardwired, and given that the interrupt routine uses less than 5 registers, interrupt latency can be as low as 12 clock cycles[24]. In systems where some high-priority task needs to coexist with one or several low-priority tasks, this method allows interleaving of the low-priority tasks between the high-priority ones. This prevents low-priority tasks from determining the high priority execution frequency. It is also possible to prevent priority inversion if care is taken during development, priority inversions are nearly unavoidable in a big-while solution.

Working on last year's code, the synchronized big while method was used in the main controller. As seen in Listing 1, synchronization is performed on the PWM period, with each new loop starting on the beginning of a PWM period. Initially,

```
90      while (1)
91      {
92            if(phase_voltage_is_new_period()){
93                 status_signal(LOOP_LOST_SYNC);
94            }
95            else{
96                 while(!phase_voltage_is_new_period()){/*counter++;*/}
97            }
```

Listing 1: PWM synchronization in main.c

this would have worked well, as there were few extra tasks to be performed. However the speed controller and CAN transmission tasks have proved to be hard to integrate in an elegant manner. As seen in Figure 5.1 these task are tacked on to the end of the control loop, and run every time a timeout of the timer counters is signalled. Listing 2 shows the CAN and speed controller code. There are separate timers and signals for each transmission frequency, and the code is kept as short as possible. This alleviates the performance issues to some degree. However, the code is left in a rather unreadable state. Tests done by toggling an output pin each period confirm a strong variation of round-trip times. The current code achieves acceptable performance with an 8-16kHz controller frequency, but becomes unstable if going much faster. If an interrupt-based platform had been developed instead, the controller period would have been undisturbed by low-priority tasks like speed control and CAN transmission. This would also have the added benefit of more elegant code, with the big while loop becoming a rather short while loop, and some of the hacks done to ensure transmission of messages being unnecessary. While possible, timer interrupts should be avoided in favor of active flag checks for the lower priority tasks. This lowers the risk of priority inversion.

```
148 if(tc_overflowOccurred(TC_TIMER_DATA)){
149     if(status_check(READY_TO_DRIVE)){
150         state.min_torque = can_wrapper_get_min_torque_request();
151         state.max_torque = can_wrapper_get_max_torque_request();
152         state.speed_request = can_wrapper_get_speed_request();
153         if(status_check(OVERRIDE_SPEED_CONTROLLER)){
154             state.torque_request = state.max_torque;
155         }
156         else{
157             state.torque_request =
 ↪  do_pid(&(state.regulator_data.PID_reg_speed), state.speed_request -
 ↪  state.speed );
158         }
159     }
160     else{
161         state.torque_request = 0.0;
162     }
163     encoder_calc_speed(&state, NUM_DATA_MESSAGES * 100);
164     //Send state data @100hz
165     send_data_message = true;
166 }
167 if(send_data_message){
168     send_data_message = !can_wrapper_send_data_msg(&state);
169 }
170 if(tc_overflowOccurred(TC_TIMER_STATUS) || resend_status_message){
171     //Send status message @10hz
172     resend_status_message = !can_wrapper_send_state_msg();
173     counter = 0;
174     status_clear(LOOP_LOST_SYNC);
175 }
```

Listing 2: Speed control and CAN transmission triggered by timer counter signals at end of control loop in main.c

5.1.2 Drivers and modules

To achieve as high performance as possible hardware and software design focused on taking advantage of the many hardware modules of the Atsam microcontroller. In many cases, this allows tasks that would ordinarily be handled in software to be offloaded to hardware modules. The following are the most important drivers and modules written for the Atsam controller program.

```
127 typedef struct{
128     adc_channel_num_t channel_number;
129     bool differential_channel;
130     bool dual_sample_hold;
131     uint16_t offset;
132     adc_gain_t gain;
133 }adc_channel_opts_t;
134
135 typedef struct {
136     adc_resolution_t resolution;
137     adc_trigger_t conversion_trigger;
138     adc_signMode_t sign_mode;
139 }adc_opts_t;
140
141
142
143 uint32_t     adc_init(Afec *const afec, adc_opts_t *config, adc_channel_opts_t
     ↪  enabled_channels[], uint8_t num_channels);
144 void          adc_change_sequence(Afec *const afec, bool
     ↪  enable_conversion_sequencing, adc_channel_num_t sequence[AFEC_NB_CHANNELS]);
145 void          adc_init_comparator(Afec *const afec, adc_comp_opts_t *const config,
     ↪  void (*callback)(void));
146
147 void          adc_start_conversion(Afec *const afec);
148 bool          adc_is_conversion_done(Afec *const afec, adc_channel_num_t channel);
149 int16_t          adc_get_result(Afec *afec, adc_channel_num_t channel);
150
151 const volatile uint32_t *adc_get_result_register_address(Afec *afec);
```

Listing 3: The ADC driver interface

ADC

The ASF code base has been simplified and condensed into the interface seen in Listing 3. `struct adc_opts_t` contains the common ADC settings; resolution, conversion trigger and signedness. For each channel initialized, `struct adc_channel_opts_t` contains the per channel information such as channel number, offset and gain. `bool dual_sample_hold` decides whether or not the channel should be sampled simultaneously with its sister channel. If software trigger is selected, `adc_start_conversion` triggers a conversion sequence. The driver will automatically select single trigger mode, so that a single trigger will start a conversion of all enabled channels. `adc_get_result` will return the last recorded value of the given channel. The ADC interface also contains functionality to enable an unused comparator function in the ADC, allowing interrupts or internal event lines to be activated when a measurement satisfies the programmed requirements.

```
57      uint32_t pwm_period_ticks;
58      bool center_aligned;
59      bool active_high_out;
60      bool enable_period_start_event;
61      bool deadtime_enabled;
62      uint16_t deadtime_pwmh_ticks;
63      uint16_t deadtime_pwml_ticks;
64  } pwm_opt_t;
65
66  uint16_t pwm_calc_deadtime_ticks(uint32_t deadtime_ns);
67  uint32_t pwm_calc_period_ticks(uint32_t pwm_frequency, bool center_aligned);
68
69  uint32_t pwm_init(Pwm *p_pwm, pwm_opt_t *settings, uint8_t num_channels);
70  bool pwm_is_next_period(Pwm *p_pwm);
71  void pwm_update_channel_data(Pwm *p_pwm, pwm_opt_t *p_channel, uint8_t channel,
     ↪  uint32_t duty);
72  void pwm_channel_enable(Pwm *p_pwm, uint8_t channel);
73  void pwm_channel_disable(Pwm *p_pwm, uint8_t channel);
74
75  void pwm_enable_override(Pwm *p_pwm, uint8_t channel);
76  void pwm_disable_override(Pwm *p_pwm, uint8_t channel);
```

Listing 4: The PWM driver interface

PWM

Listing 4 shows the PWM module interface. A single initializer function takes the number of channels to be initialized, in addition to `struct pwm_opt_t` containing the settings for all channels. The driver assumes that all channels share all settings except duty cycle. The `uint16_t deadtime_pwmh_ticks`, `uint16_t deadtime_pwml_ticks` and `uint32_t pwm_period_ticks` variables can be calculated from SI units with `pwm_calc_deadtime_ticks` and `pwm_calc_period_ticks`. `pwm_is_next_period` allow synchronizing the processor loop with the PWM period. `pwm_update_channel_data` updates the channel duty cycle.

```
71  typedef struct{
72       uint32_t                channel_number;
73       Xdmac_ch_Sz_t           element_size;
74       uint32_t                array_size;
75       uint32_t                num_transfers;
76       Xdmac_ch_dir_t           peripheral_transfer_dir;
77       volatile uint32_t*      volatile source_adress;
78       volatile uint32_t*      volatile destination_adress;
79       Xdma_event_num_t      trigger;
80       uint32_t                interrupt_mask;
81       uint32_t                interrupt_priority;
82       void                     (*callback)(uint32_t interrupt_status);
83       Xdmac_bus      source_bus;
84       Xdmac_bus      destination_bus;
85  }Xdma_transfer_opts_t;
86
87
88  void xdma_setup_peripheral_transfer(Xdmac *xdmac, Xdma_transfer_opts_t* opts);
89  void xdma_setup_memory_transfer(Xdmac *xdmac, Xdma_transfer_opts_t* opts);
90  void xdma_start_transfer(Xdmac *xdmac, Xdma_transfer_opts_t* opts);
91  void xdma_stop_transfer(Xdmac *xdmac, Xdma_transfer_opts_t* opts);
```

Listing 5: The DMA driver interface

DMA

The Direct Memory Access module allows the processor to perform background access to internal and peripheral memory areas. By providing the DMA with two pointers and a data size, data can be moved from one memory area to another, without direct processor intervention. This allows large contiguous memories, such as character strings or SPI commands to be efficiently transmitted without busy-waiting. `Xdma_transfer_opts_t` contains the settings for one DMA channel, or transfer. DMA transfers transfer a number of arrays, or microblocks, of elements, as illustrated in Figure 5.2. The size of both element, array and transfer can be adjusted. Transfer setup is done once with `dma_setup_peripheral_transfer`, or `dma_setup_memory_transfer`. After setup, the transfer can be triggered at any time with `xdma_start_transfer`. To ensure that all available data has been transfered when fetching a result, `xdma_stop_transfer` should be called to flush the DMA buffers before reading the output pointer.

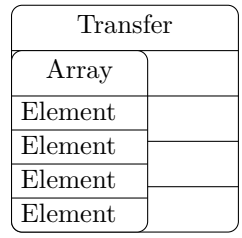


Figure 5.2: DMA datastructure

```c
#define PID_AW_CLAMPING
//#define PID_AW_BACKCOUNTING

typedef struct{
     float32_t K_p;
     float32_t K_i;
     float32_t K_d;
     float32_t K_antiwindup;

     float32_t T_s;
}Pid_settings_t;

typedef struct{
     Pid_settings_t settings;

     float32_t integrator;
     float32_t last_input;
     float32_t last_output;

     float32_t saturation_max;
     float32_t saturation_min;
}Pid_data_t;

float32_t do_pid(Pid_data_t *data, float32_t error);
```

```c
float32_t do_pid(Pid_data_t *data, float32_t error){
#ifdef PID_AW_CLAMPING
     if(!pid_saturated(data->last_output, data->saturation_max,
 ↪   data->saturation_min)){
          data->integrator += error;
     }
#else
     data->integrator += error;
#endif


     float32_t derivate = error - data->last_input;

     float32_t result = ((data->settings.K_p * (error)) + (data->settings.K_i *
 ↪   (data->integrator)) + (data->settings.K_d * (derivate)))/data->settings.T_s;

     float32_t out;
     out = fminf(result, data->saturation_max);
     out = fmaxf(out, data->saturation_min);

#ifdef PID_AW_BACKCOUNTING
     data->integrator -= data->settings.K_antiwindup * (out - result);
#endif
     return out;
}
```

Listing 7: The PID module

PID

Last year's inverter used the `arm_math` PID library to implement PID controllers. While they worked satisfactorally, there was no explicit integrator variable, making it hard to implement anti-wind up measures. It was therefore decided to implement a custom PID module. The new PID regulators allow both back counting and integrator clamping, and have internal saturation functionality. No initialization is necessary, as all information is stored in structures.

```
27 typedef union{
28     uint32_t     u32;
29     int32_t          i32;
30     uint16_t     u16[2];
31     int16_t          i16[2];
32     uint8_t          u8[4];
33     int8_t          i8[4];
34 }flash_data_t;
35
36 uint32_t flash_read_user_signature(flash_data_t *p_data, size_t size);
37 uint32_t flash_write_user_signature(const flash_data_t *p_buffer, size_t size);
38 uint32_t flash_erase_user_signature(void);
```

Listing 8: The Flash storage Interface

Flash Storage

The inverter should be able to store tuning parameters between power-ons. While no external memory has been added to allow this, the Atsam processors feature a reserved user-writeable flash memory of 512 bytes. This memory can be used to store any data required. The interface in Listing 8 allows access to the flash userpage, as this memory segment is called. While the interface is fairly straight forward, the user must be aware that accessing the user page memories will lock the flash memory during read or write operations. The user page accessors are loaded to RAM, and are not dependent on flash access to be run. All other processor activity will however be suspended, as trying to access the flash in this period will lock the processor. Before writing to the flash memory, it is recommended to wipe the stored data to all ones with `flash_erase_user_signature`, as `flash_write_user_signature` will only write zero values, not ones. In the inverter, the user page module is only accessible when the power stage is completely inactive for safety reasons.

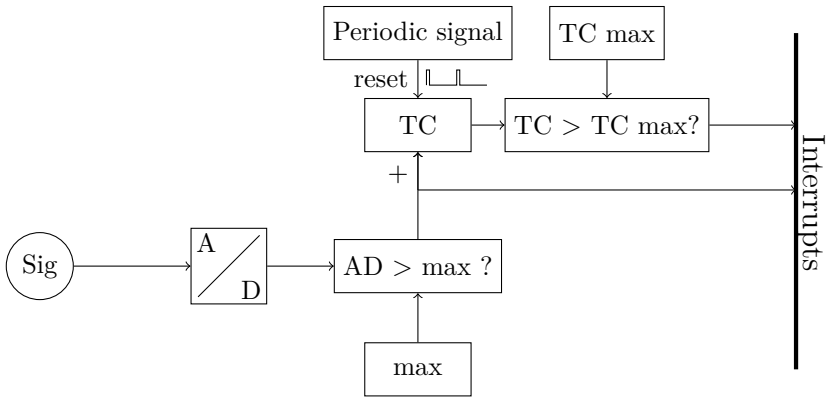


Figure 5.3: Proposed hardware threshold guard using ADC comparators and timer counters

5.1.3 Future Work

Completing the Atsam code is as far as the authors are aware the last puzzle piece missing for a working inverter. Simplifying and rewriting it after delivery may well yield a functional prototype if time allows. If the inverter code is to be rewritten, changing the scheduler code to an interrupt-based implementation would be fairly simple, and should be considered. In addition the ADC comparator circuit and timer counters could be combined as in Figure 5.3 to allow hardware status checking both for slow-moving and fast signals. This would greatly off-load the processor, and enable all error-checking algorithms to run parallel with the control loop.

5.2 Atmega SW

5.2.1 Safety checks

As the control system needs several safety checks to perform as intended, these have been offloaded from the Atsam processor to an external Atmega. The Atmega reads the temperature three places on each power PCB, and on each motor. In addition, the processor reads the DC voltage and the overcurrent signals from two current sensors. The interface to the control insert consists of the three signal lines listed in Table 5.2.

Signal name	Indicating
E_GEN	Overvoltage
E_T	High temperature
E_C	Overcurrent

Table 5.2: Error lines from Atmega to each Atsam

5.2.2 Program

Initializing

Setting up the processor and the peripherals is done in four steps:

```
97  int main(void){
98      setupIO();
99      adc_init();
100     timer1_init();
101     can_init(clkph_16MHZ, CAN_BPS_1000k, rx_info, NUM_CAN_RX_BUFFERS);
102     sei();
103
104     // Kick off adc continous adc reading.
105     adc_start_continous();
```

Listing 9: Atmega Initialization

The last thing done before the main loop starts is calling `adc_start_continous()`, which starts the first conversion. All subsequent conversions are started from the ISR.

Main loop

The main loop performs two functions: translating raw values and setting the correct error pins.

```
107       while (1) {
108            for (int i = 0; i<ADC_CHANNELS_NUM; i++){
109                 translateValue(adc_raw[i], i);
110            }
111            setErrorPins();
112       }
113  }
```

Listing 10: Atmega Main loop

`translateValue` loops through the raw data array, and translates the raw values to either temperature in centidegrees (1/100th degree) celcius or centivolts, based on the position in the array currently being read. As there are two different types of temperature sensors - one for the inverter temperatures and one for the motor temperature - the function also differentiates between two formulas. The formulas were extracted from the temperature sensor's datasheets by doing linear regression on the given resistance values per temperature.

```
21   int16_t sensor_adc_to_temp(int16_t adc_value, bool motorTemp){
22        int16_t dVin = sensor_adc_to_mVin(adc_value);
23        if(motorTemp){
24             return 41703 - 190*dVin;
25        }else{
26             return 46*dVin - 452;
27        }
28   }
```

Listing 11: Converting ADC data to temperature

```
38   int16_t sensor_adc_to_vdc(int16_t adc_value){
39        int16_t dVin = sensor_adc_to_mVin(adc_value);
40        return (dVin*((R1+R2)/R2))/100;
41   }
```

Listing 12: Converting ADC data to voltage

Setting the error pins is done separately for each inverter system, with hysteresis. If any of the four temperatures are above the limit, the error pin is set, and it is not turned off until the temperature is below a lower limit. The overcurrent (OC) checks are done in a similar fashion: if either sensor is triggered, the E_C signal goes high, and it does not go low again until both OC signals are inactive.

ADC	Sensor
ADC2	Inverter 2, temperature 3
ADC3	Inverter 2, temperature 2
ADC4	Inverter 1, temperature 3
ADC5	DC voltage
ADC6	Inverter 1, temperature 1
ADC7	Inverter 1, temperature 2
ADC8	Inverter 2, motor temperature
ADC9	Inverter 2, motor temperature
ADC10	Inverter 2, temperature 1

Table 5.3: ADC and sensor mapping

ADC interrupt

The ADC module triggers an interrupt when a conversion is done. The ISR reads the ADC value, and stores it into the current place of the raw data array. Subsequently, the index is incremented by one, and the ADC channel to be read is updated. The sensors are mapped to the ADCs as shown in Table 5.3. Since the ADCs used start at ADC2, the channel index is incremented by 2 before being loaded into the channel register of the ADC module. Then the next conversion is started.

```
232 ISR(ADC_vect){
233     static int8_t adc_index = 0;
234
235     // Read ADC conversion result
236     adc_raw[adc_index] = ADC;
237
238     // Increment channel to be read
239     adc_index ++;
240     adc_index = adc_index % ADC_CHANNELS_NUM;
241     adc_set_source(adc_index + 2); // +2: want to read ADC2..10, not 0 and 1
242
243     // Start new conversion
244     ADCSRA |= (1<<ADSC);
245     _delay_us(10);
246 }
```

Listing 13: ADC Interrupt Service Routine

```
ISR(TIMER1_COMPA_vect){
     // Timer 1 @ 10Hz
     // Update message data
     for(int i = 0; i<4; i++){
          temperature_inv1_msg_data.i16[i] = sys_1_temp[i].temp;
          temperature_inv2_msg_data.i16[i] = sys_2_temp[i].temp;
     }


     /***** MESSAGE TESTING ****/
     /*temperature_inv2_msg_data.i16[0] = adc_raw[0]; // test switch temp formula
     temperature_inv2_msg_data.i16[1] = sys_2_temp[T3].temp;
     temperature_inv2_msg_data.i16[2] = adc_raw[1];
     temperature_inv2_msg_data.i16[3] = sys_2_temp[TM].temp;*/
     can_send_message(&temperature_inv1_msg);
     _delay_us(1000);
     can_send_message(&temperature_inv2_msg);
     _delay_us(1000);

     static int8_t alive_timer = 0;
     // Executes @1 Hz.
     if(alive_timer == 9){
          alive_msg_data.u8[0] = 0x00;
          can_send_message(&alive_msg);
          alive_timer = 0;
     }
     alive_timer ++;
}
```

Listing 14: Timer Interrupt Service Routine

Timer interrupt

The timer counter is set up with interrupt and a prescaler of 256. Using a 16MHz crystal, this results in $\frac{16\text{MHz}}{256} = 62500\text{Hz}$ in the counter. For a 10Hz interrupt frequency, the timer is loaded with $\frac{62499\text{Hz}}{10Hz} = 6249$. This results in a slightly faster than 0.1s interrupt period. This interrupt launches data logging CAN messages, sending two messages, each containing temperature log data for one motor and inverter. In addition, a timer counts up, and when reaching 10 sends an alive signal message.

5.3 Zynq software

During the development of the control system, the initial plan was to implement it on a Zynq SoC. However, several obstacles prevented this plan from coming to life. Using the SDSoC and Vivado software from Xilinx proved to be a steeper learning curve than expected, resulting in a very slow development. While there were a fairly good plan on which hardware elements were needed on the platform, the development of this platform countered several problems.

After the decision to go away from the Zynq as the control system chip, the work had to be targeted a little differently, towards making the system as good a starting point as possible for next year's team to continue working on the inverters. The work done on the Zynq has been focused on the basics for a control system - communication with the car, sensor readings and PWM readings. These parts of the system have all been implemented as hardware elements in the platform, and the majority of work required to get them working is understanding the supplied drivers, and altering them or creating new where necessary.

The work that has been done in this project on the Zynq system will be a good starting point for next year's team to build on. The focus has not been so much on implementing the actual inverter control loop, as it has been on understanding the underlying structure of the program and acquiring experience with the tools that can be passed on to those working with the SoC later. Putting together the complete hardware platform is up to future iterations of the project, but the experiences drawn here should make the system relatively easy to pick up.

5.3.1 Hardware platform

The hardware platform contains all the desired elements of the system, and is the base on which the entire application runs. For Xilinx' devices it is generated in Vivado, a block diagram style editor. The main component of the hardware platform is the Zynq processing block shown in Figure 5.4. Having this in the platform enables the processing system, and most of the Zynq peripherals (I^2C, USB, etc) are activated through the settings of this block. Without this block, it will not be possible to program any software onto the chip, and it will act as a regular FPGA only using the programmable logic side. There is also the possiblity to implement a soft-core MicroBlaze processor on the platform instead of the main

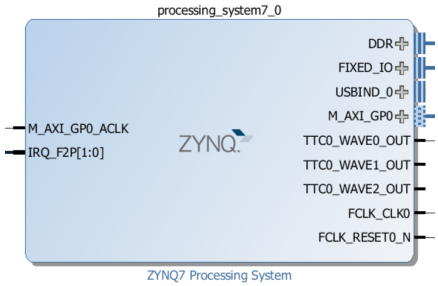


Figure 5.4: Zynq processor system block

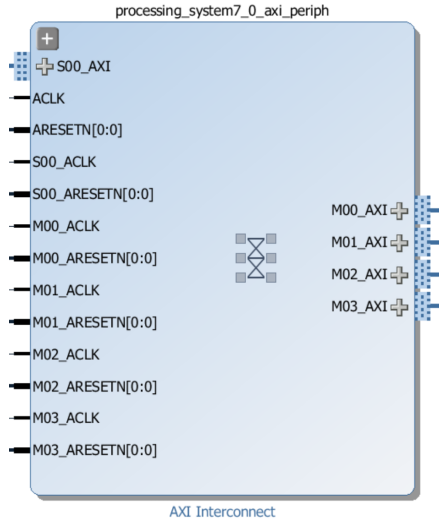


Figure 5.5: AXI interconnect block

processing block, or both.

The processing system communicates with the rest of the hardware platform through AXI buses, with a special interconnect used to connect up to 64 slaves to the AXI master port on the processing block. With two AXI master ports in the processor, there is room for a plethora of modules in the hardware platform.

GPIO

The chip connects with the outside world through different types of general purpose IO ports, activated in different ways. The PL side has IO ports connected through a GPIO block, which in turn communicates with the rest of the system on the AXI bus. Each of these GPIO blocks can have two channels active, each controlling up to 32 pins.

The PS side of the chip has its own dedicated IO bank, called MIO. On most of the SoMs produced, the MIO pins are dedicated to direct communication with module chips such as DDR RAM, Flash memory and such, but some are left to the user, and used to connect to the communication peripherals. In addition to the

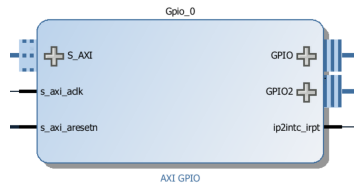


Figure 5.6: AXI GPIO block

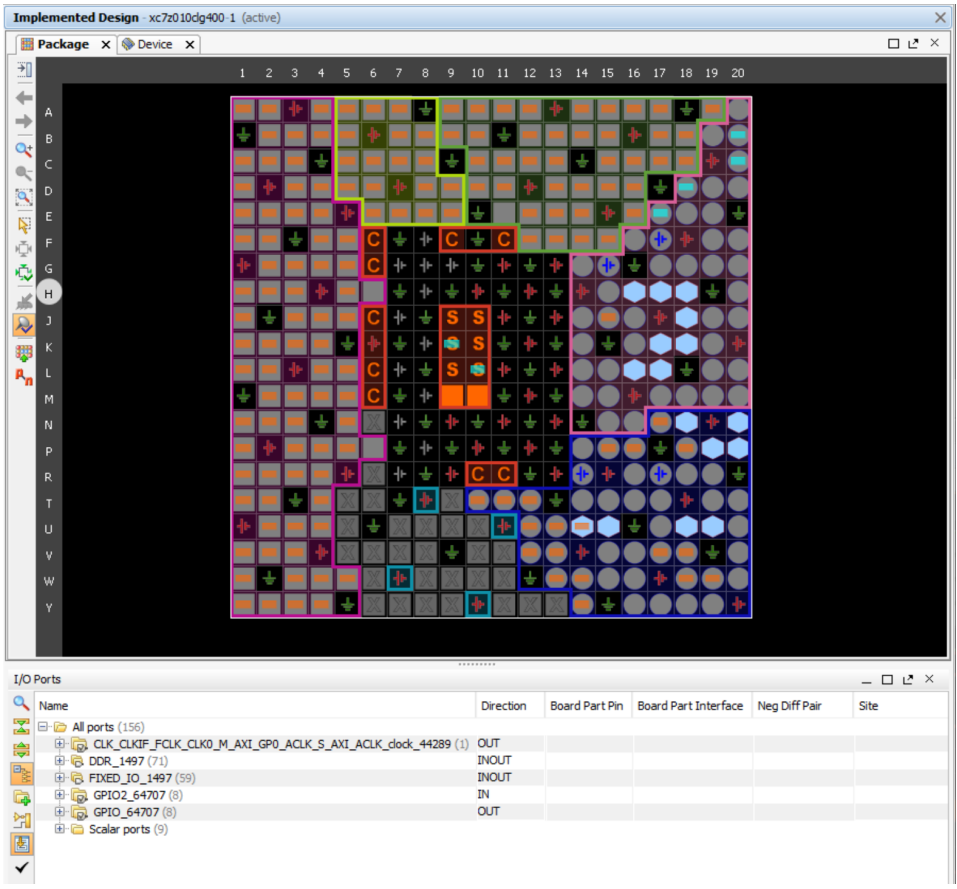


Figure 5.7: The package view in Vivado

dedicated MIO pins, the PS can connect with the package pins via EMIO, which provides direct PS connection through the PL. All the preipherals in the PS can be set to use EMIO instead of a set of MIO pins, and then the EMIO connections can be given to a specific package pin.

Connecting the different IO pins in the block diagram is done through the package view in Vivado, shown in Figure 5.7. All signals set to be external from the block diagram are listed, along with a view of each physical pin on the package. The package pins are divided into banks used for different purposes - for example, bank 34 and 35 (outlined in blue and pink, respectively) are used for PL GPIO pins, and each signal are assigned to a specific pin through this view.

Board support package

Once the hardware platform block diagram is complete, a bitstream can be generated for transfer to the chip. If the platform includes a processing system, there will be need of a board support package(BSP) to tell the processor which hardware blocks it has access to and how to communicate with them. The hardware code is exported from Vivado before the BSP is generated in SDK or SDSoC, and an application project uses the BSP as a foundation before the code itself is written. Each block included in the hardware platform includes a written C/C++ driver, and the BSP consolidates the drivers for all the elements of the platform, and produces a file with platform specific parameters such as each element's address on the AXI bus etc. In the application project, the drivers can be included from the BSP and the different block specific parameters used to work with the hardware platform.

5.3.2 Development hardware platform

Despite the aforementioned problems, some parts of the inverter system has been implemented with working prototypes. The hardware platform the test system is shown in Figure 5.8.

GPIO The first obstacle getting to know the system was getting a working GPIO interface, with blinking LEDs and making the processor detect button pushes. This will provide the basics of reading the ECU enable signals and overcurrent signals, handling enable signals to the gate drivers and quick status indicators via LEDs on the control board.

During the initial work with the system, a simple hardware platform with a GPIO block was implemented, but did not work properly. It appeared communication over the AXI bus somehow was restricted, due to some access violations in the underlying AXI bus protocols on the chip. The authors searched for solutions to this online, but solutions from similar cases did not solve the problem. A request was made to Xilinx' support forums, but with no response from their staff, the problem persisted. This time delay was some of the reason the SoC solution was dropped as main focus, and implementation on the familiar Atsam processor began.

Once the Atsam-based solution was underway, the Zynq platform was picked up

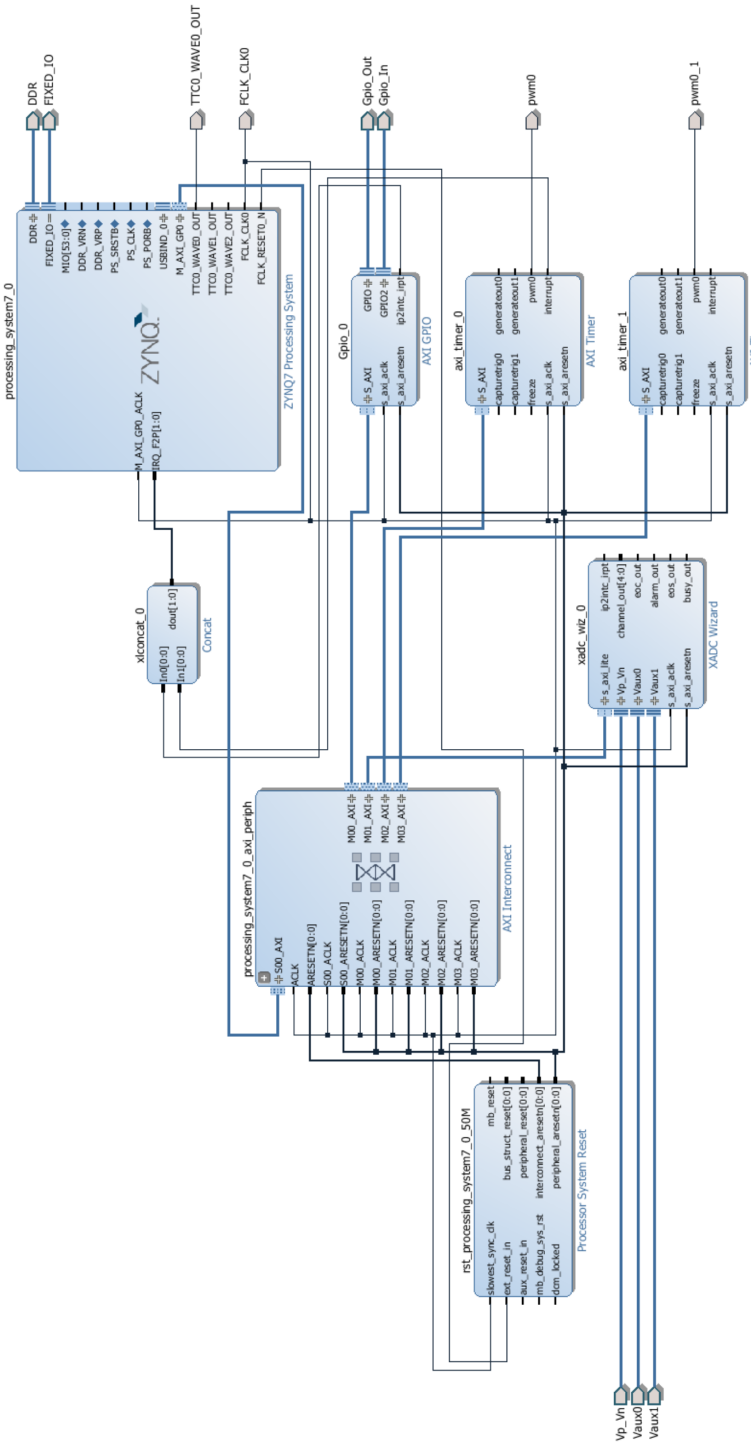


Figure 5.8: Test hardware platform

with a fresh hardware platform project, and this time the problem had dissappeared. In the first case, the project was a SDSoC project, which includes the ability to automatically create hardware accelerators. In the second case, the project was based off a simpler SDK-style application project, which may have a different underlying project structure, solving the problem. The problem may also have been solved by updated drivers, since the second attempt was done several weeks later.

CAN A simple working CAN project was tested to prove the function of the CAN peripheral. The MicroZed was conencted from the MIO pins through transceivers on an STK500 board, and the CAN bus interface on the STK500 to a development board for Atsam4e programmed to print whatever received on the CAN bus to a terminal. Using the provided drivers was sufficient, and although no Revolve-related IDs or data structures were sent, the test data was successfully written from the Atsam4e.

ADC The on-chip ADC XADC is instantiated in the hardware platform, and can be configured with any of the 17 differential signal pins activated. As these have dedicated package pins, no pin assignments are needed for these. The XADC smales 12 bits at up to 1MSPS. This is enough channels to measure all signals that is needed for two inverters - three temperatures and three currents per inverter, and the DC voltage. Measuring all these signals yields $\frac{1MSPS}{13} \approx 77KSPS$ for each signal, if the XADC is allowed to run in continous mode. In testing and getting to know the XADC, the internal temperature and voltage level measurements was tested. These proved to function well, and the provided functions for translating raw values to temperature for the sensor was used.

PWM PWM is created by an AXI timer block with two internal counters, using both to create the PWM. Counter 0 is used to decide the PWM frequency, and counter 1 is used to decide the duty cycle. Equation 5.1 describes how the two registers provide the desired PWM output, based on the clock frequency input to the IP.

$$PWM\_PERIOD = (MAX\_COUNT - TLR0 + 2) * AXI\_CLOCK\_PERIOD$$
$$PWM\_HIGH\_TIME = (MAX\_COUNT - TLR1 + 2) * AXI\_CLOCK\_PERIOD \tag{5.1}$$

The PWM test have proved not to work entirely as expected. The initialization and loading of load registers seemingly works well, but the measured output is inverse - with the registers loaded to an expected period of 1/10 s, the output period is instead 10 seconds, and the same with the duty cycle. The reason for the error probably lies in the transformation of Equation 5.1 to provide a formula for what value to load into TLR, but the authors has not yet been able to verify this.

5.3.3 Future work

There is naturally a great deal of work to be done before a functioning system is completed, but the work presented here should provide a head start.

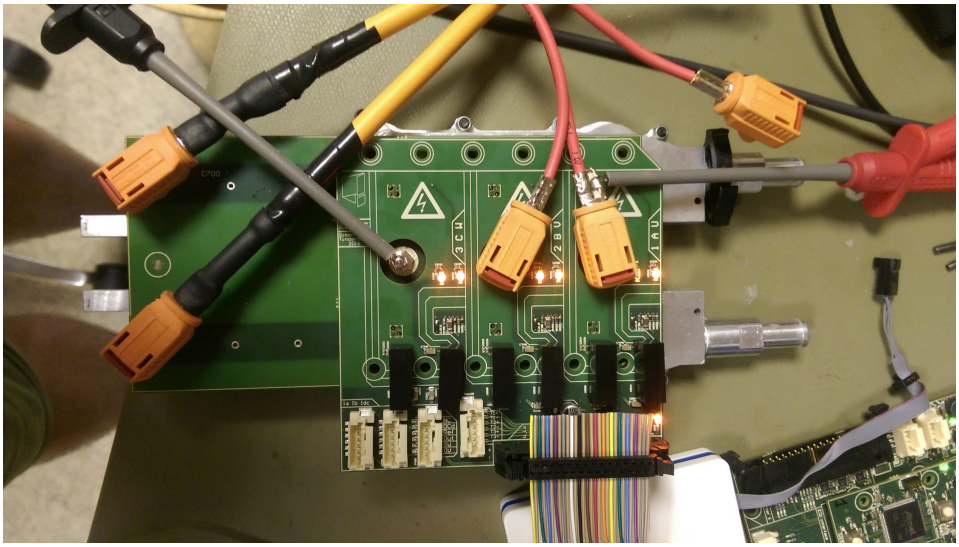
- The hardware platform will need completion. The full extent of the hardware platform will depend on the chosen solution with regards to PS/PL usage, and deciding which side of the SoC should complete which tasks.
- Implementing the control loop on the ZX5 modules and using them in the completed system will require a redesign of the control board, as both the MAX10 and the Atmega subsystems can be implemented directly into the Zynq platform.
- Proper wrapper functions to encapsulate the drivers in the BSP to better suit the project will make the process of writing the application code that much easier.

6. Unit Tests

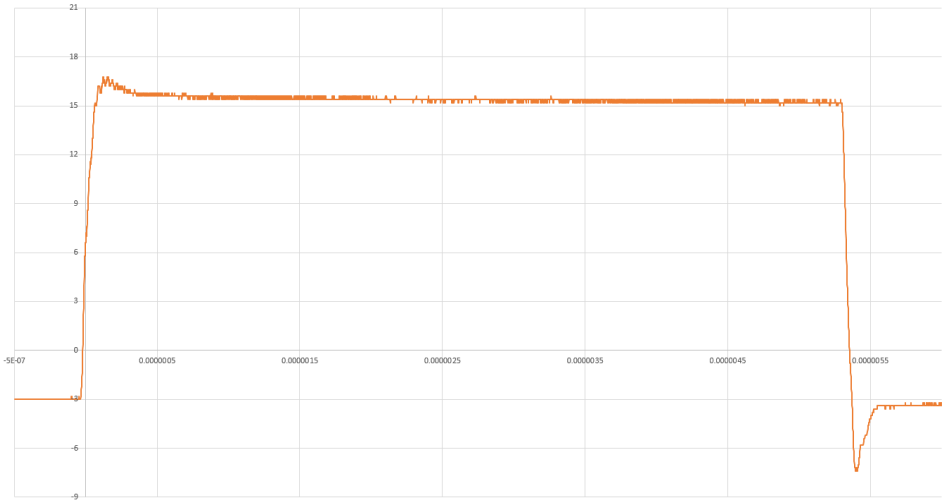
To verify the system hardware, unit tests have been performed on the most important systems. The ECU interface and shutdown circuit measurement were not tested because they were not considered important until the system should be mounted in the car. The shutdown curuit measurement is based on a standard circuit used with success both in the discharge, precharge, and BMS modules. USB communication was not tested, as the CAN interface is sufficient until controller tuning is started.

6.1 Gate Drivers

The gate driver output filter was tested by applying a constant square wave with 50% duty cycle and 400ns dead time to the gate driver inputs. The resulting waveform was recorded with an oscilloscope. Test were done both with and witout connected transistors. In addition, the waveforms were measured with a 3A load current through the power stage. As can be seen in Figure 6.1b, the voltage rises to 15V, with a slight 3V overshoot, and returns to -3V upon deactivation.



(a) Gate voltage test setup with phase voltage measurement.

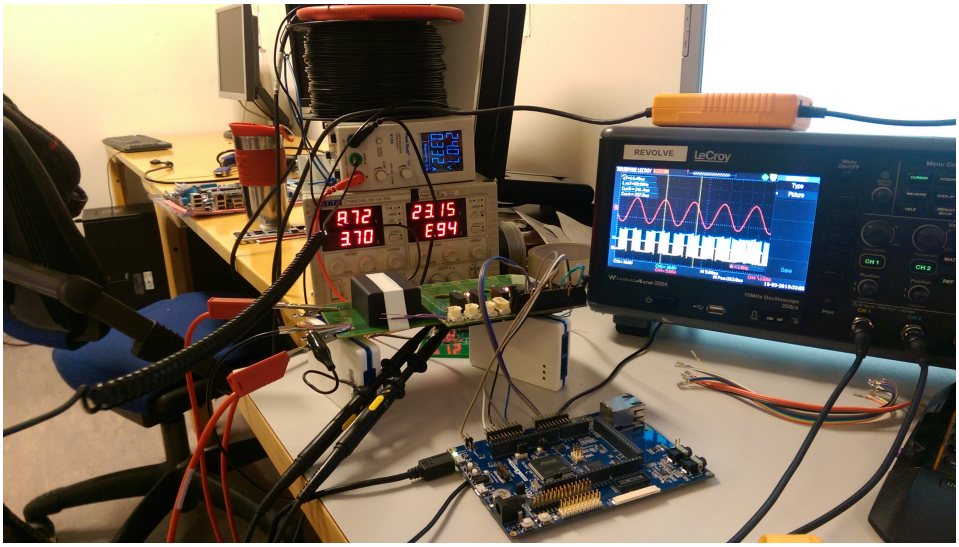


(b) Gate terminal voltage under 3A load. Plotted in Excel.

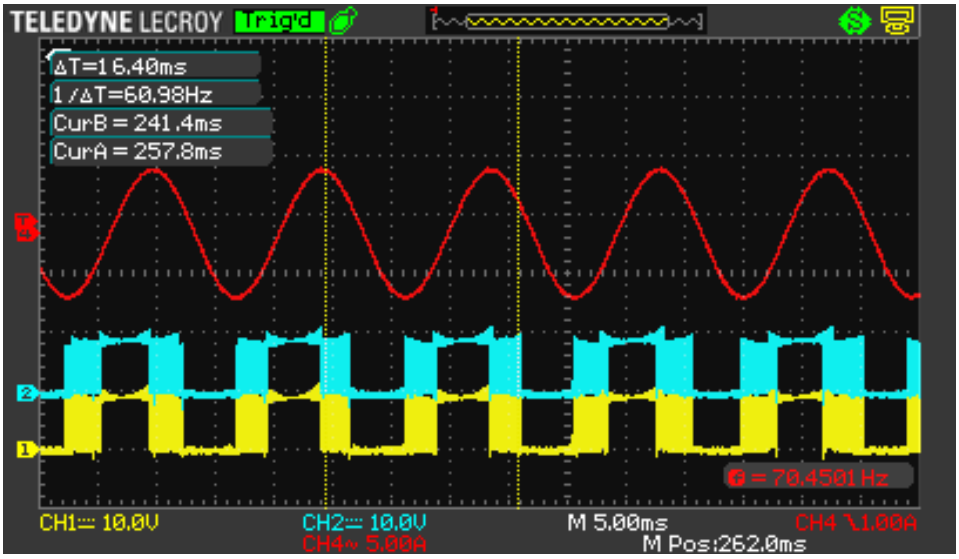
Figure 6.1: Gate driver test setup

6.2 Power Stage

To test the power stage of the inverter, a 30V sine wave was applied to a test inductor using pulse width modulation. The test inductor consists of roughly 150m of coiled 1.5mm^2 wire. The current through the coil was measured with a Fluke iFlex 3000 AC current probe. In Figure 6.2a the test setup can be seen. An atsam devkit is used to provide power and PWM signals to the gate driver. The inverter power stage is being driven from the 10V lab supply in the background. On top of the lab supply stack, the load inductor and red current probe can be seen. This test was done in early february on the prototype inverter. When attempting to reproduce it to get better data, it was discovered that two of three phases had undiscovered issues due to the earlier overcurrent incident. While the data was not exported in a format suitable for plotting, the oscilloscope screen dump shows the results well enough. The power stage performed perfectly in the initial test, replacing the faulty transistors on the power PCB will resolve the current issues.



(a) Current measurement test setup

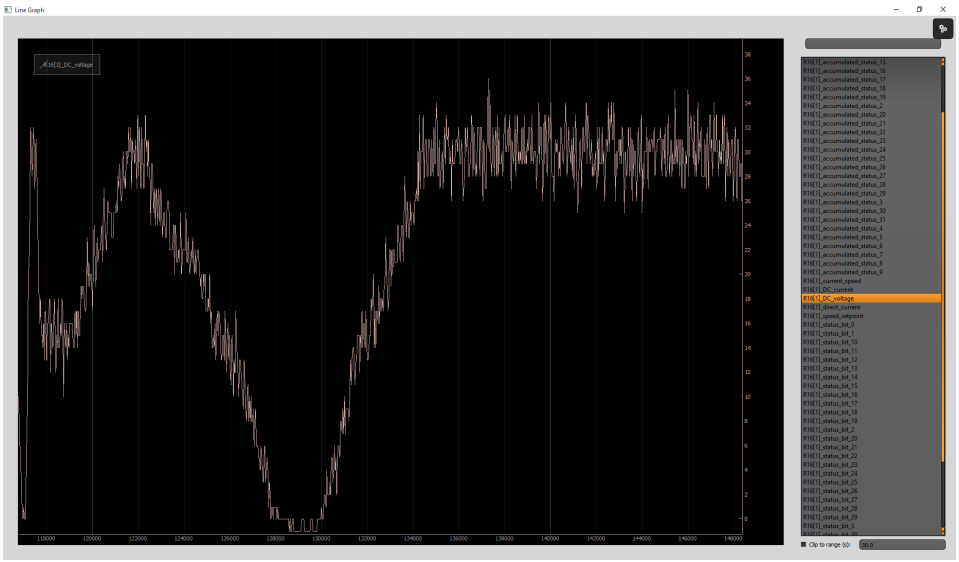


(b) Gate voltage signals (channels 1, and 2) Phase current (channel 3).

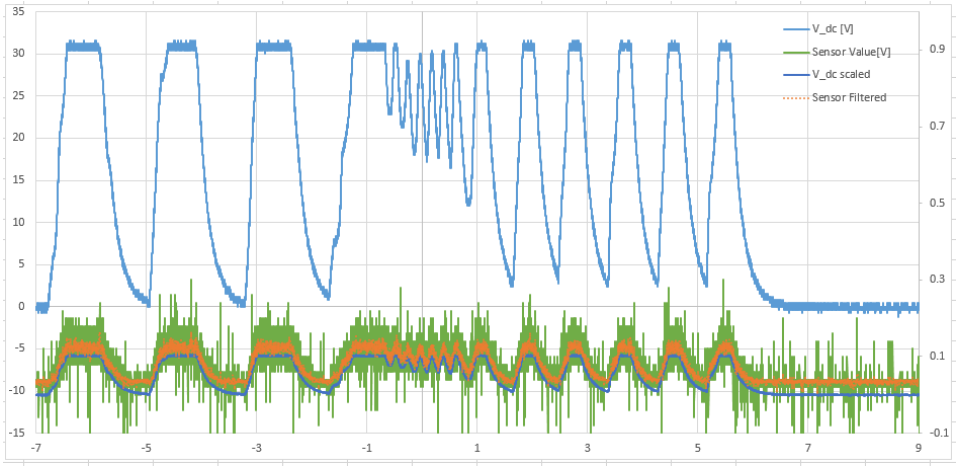
Figure 6.2: Power stage test setup

6.3 DC Voltage Measurement

The DC voltage measurement was tested by applying a varying voltage to the measurement points. The measured voltage was filtered with a 10 sample running average at 8kHz, and transmitted through CAN to Revolve Analyze. Measurements were also taken with an oscilloscope to verify that the results were not just lucky noise-pickup. The DC link voltage was measured before and after the isolation circuit. The scaled voltage was filtered with a running average trend filter in Excel, and the DC link voltage was multiplied by the circuit scaling factor (0.0033). This allows verification of the scaling circuit.Figure 6.3a shows the resulting output voltage in Revolve Analyze. The measurement is precise to within 2V, more than adequate when considering the total range of more than 600V, making it precise within 1%. A heavier filter may also be applied to acquire a more precise value, but this may introduce a measurement offset if the noise filtered is not pure white noise. A seen in Figure 6.3b, the scaling circuit introduces a sizeable amount of noise on the noise. This is probably due to the isolation circuit, and the low output voltages measured.



(a) Voltage measurement result in Revolve Analyze



(b) Voltages measured at measurement input (V_dc, left axis) and after isolation circuit (Sensor value, right axis)

Figure 6.3: DC voltage measurement tests

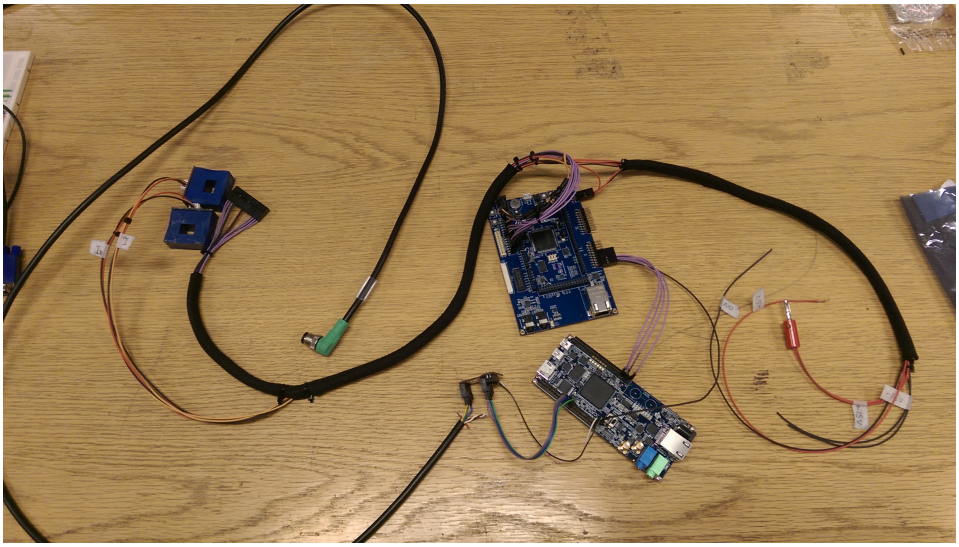
6.4 Current measurement

When testing the current measurement circuit, a design error was found in the schematics for the INA337 amplifier used to scale the sensor output voltage from 0-5V to 0-3.3V. Because the input polarity was switched around, the amplifier always outputs 0V. Aside from this error, the sensors and connectors worked perfectly, and the ADC measurements were already verified with the DC voltage test.

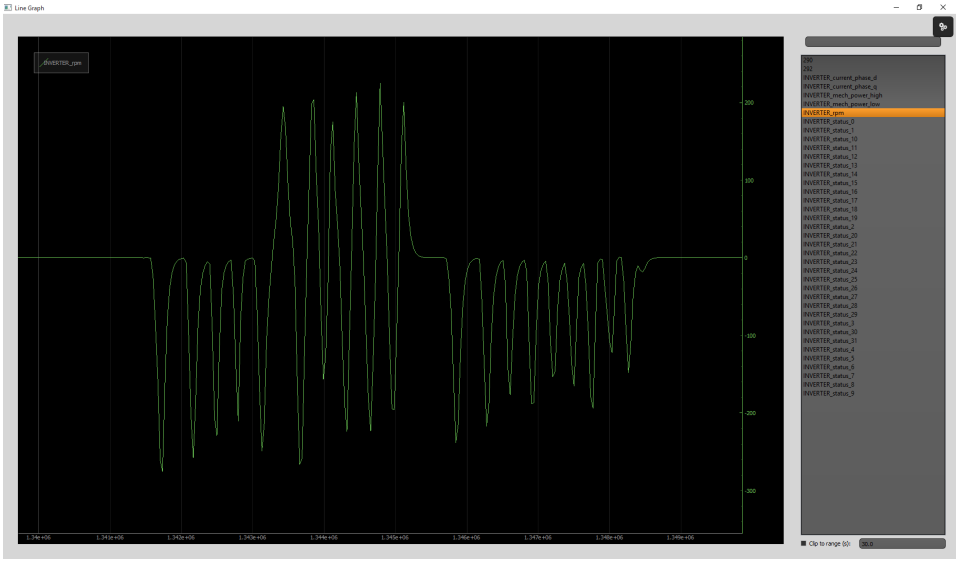
6.5 Motor Drive tests

To test the control loop before the final hardware revision was back from manufacturing, a testbed was designed from development boards and the working prototypes of the Power Board and Gate driver. This setup consisted of two developer's kits, and two milled circuit board to adopt the current measurements and encoder signals. An ATSMV71Q21 Xplained Ultra development board acted as a controller board, the Atsam V71Q21 microcontroller posessing the same processor core, and containing a superset of the Atsam E70N21's features. All drivers written for the E71 also works for the V71Q21 if written with some care, which allows this board to be used as a testbed without modifying any code when switching to the finished boards. An Arrow DECA Max10 developers kit with a milled RS485 transciever board allowed measuring the encoder position. To connect the system cleanly without a giant wire bundle, a harness connecting the current sensor board, Gate Driver, Control Board and lab supply was made.

At the time this setup was finally in place, there was only two weeks testing time left before the system had to be finished, or left off the car. In the end, no successfull tests of the motor control algorithm was performed, but several successfull encoder tests, and prototype current sensor measurements were performed. Figure 6.4b shows speed data from the motor axle being rotated by hand. Due to the hurried nature of the project at this point there are no current sensing data retained.



(a) Testing harness



(b) Motor speed measurement

Figure 6.4: Motor drive test setup

6.6 Testing Accident

While testing the motor controller, a combination of an encoder failure, current measurement error and lack of sleep lead to the Power Board sustaining damage to the second phase. One transistor pair blew up, and damage was sustained, but not initially detected in the two other phases. There was significant scorching of the PCB as seen in Figure 6.5, ut as far as can be ascertained, no damage was done to the traces themselves. The damaged transistor pair was replaced, and resistance tests were done across the drain-source connection of rest of the phases. The rest of the power stage was assumed unharmed. Later testing revealed that the the lower sections of phases V and W also were damaged, and had short circuits from gate to source terminals. The cause of the failure is assumed to be an encoder or current measurement failure, causing the DQ-reference plane to be out of alignment with the rotor. This would cause large stationary currents as the inverter will be unable to properly regulate the output voltage.

While this posed no risk to people, or other equipment, the inverter testing routines were amended to prevent a new incident. Up until this point, only one person had been present at the lab during low voltage testing. Later testing always had two people present, to ensure that operating the equipment while tired, or otherwise reckless operation would be prevented. In addition, new routines for verifying the functionality of all systems before engaging the motor control was implemented. As it was recognized that the user error was due to exhaustion, this was considered appropriate action. No further events of the kind occured.

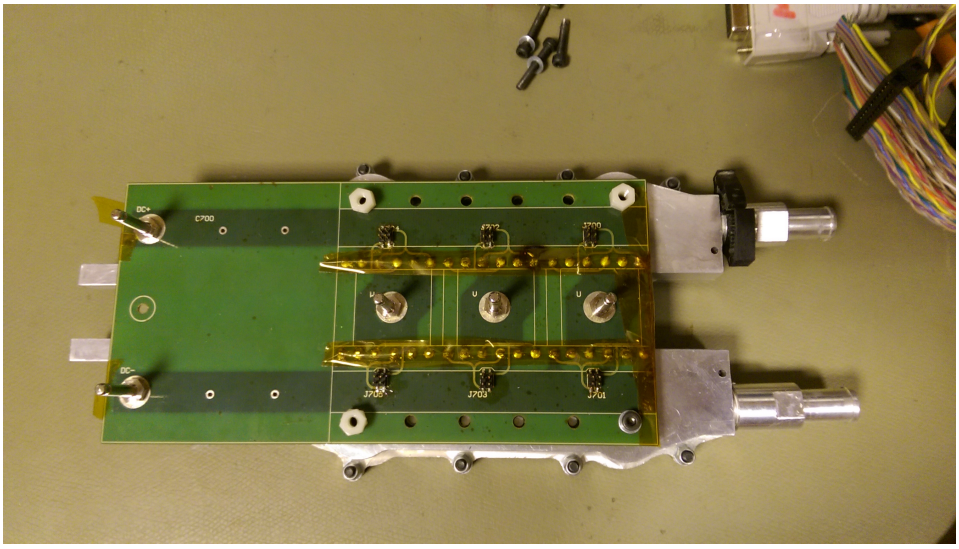


Figure 6.5: Damages after overcurrent incident. Scorching can be seen near J702, above phase V.

7. Future Work

As the inverter is not at the moment functional, there is some work to be done if it is to be used next year. It is the authors' belief that the current hardware components are functional, and that only software changes should be necessary to make the system functional.

For the software side of the project further work should focus on simplifying the existing code base, improving stability, and most important, making it work. A simple software that spins a motor inefficiently is light years ahead of an advanced software that does nothing.

- The Atsam control software should be rewritten as an interrupt-timed control loop. This will remove some insecurities in the scheduler as implemented, and make the code easier to work with. There are allready drivers designed for this purpose on the ECU module, so this should not be a very hard task.
- The phase overcurrent, and DC overvoltage checks, among others, should be implemented as hardware guards using the ADC comparators. This will offload large parts of the rather taxing error handling algorithm to hardware. Regardless of how the comparator signal is handled, through flag checking, or interrupts, this will also allow faster response for the affected systems.
- Focus should be kept on testing simple control systems before more advanced ones. This is especially important as simulations show that the AMK motor requires little to no field weakening action to reach maximum speeds. MTPA and field weakening algorithms should therefore be taken out of the code base until it is proven that the system can rotate the motor.

Further work on the hardware platform should be kept off until the existing one is verified with working software. With that said, there are several ways to improve packaging, cost and performance.

- The most important change that can be made to hardware platform is a redesign of the power board. The layout outlined in subsection 4.4.4 would solve most of the problems stated about the power board, but still has some

logistical problems regarding gate drivers, DC link placement, and connector design.

- While the chosen Amphenol Radlok connectors simplify inverter assembly greatly, they are not really small enough for use with the $4mm^2$ wires used in the phase connectors. Ordinary cable shoes should be considered as a replacement. This will also reduce the system cost.
- As an extreme space saving measure, a redesign of the power boards, gatedrivers and control card might allow the controller card to stack atop the others. This would shrink the inverter by up to 25%, and make the system more immune to noise. This would however also very negatively affect the maintainability of the completed system.

To improve computing performance, an effort should be to further develop the Zynq SoC solution. A working Zynq controller would allow more efficient control strategies, more flexible logging, and compress the control card considerably.

- The safety system developed for the Atmega and the encoder IP in this project may be integrated onto the same chip, greatly reducing hardware cost and PCB design complexity. This integration can either be done directly into the main control loop running on the ARM core, or as a separate soft-core processor running in parallel, or as a HDL-written hardware IP implemented on the PL side.
- Tests should be run on the efficiency of the system, and different ways of implementing the system with regards to how much work is done on the PL side vs. the PS side, and which parts of the PS application could benefit from being turned into hardware accelerators.
- If a working Zynq prototype is achieved, the control card should be redesigned, as the Zynq SoC would render both Atmega and Max10 coprocessing systems unnecessary. This redesign has the potential of greatly reducing the size of the board, and if coupled with the previously proposed PCB stack, reduce cable cluttering in the inverter.

8. Conclusion

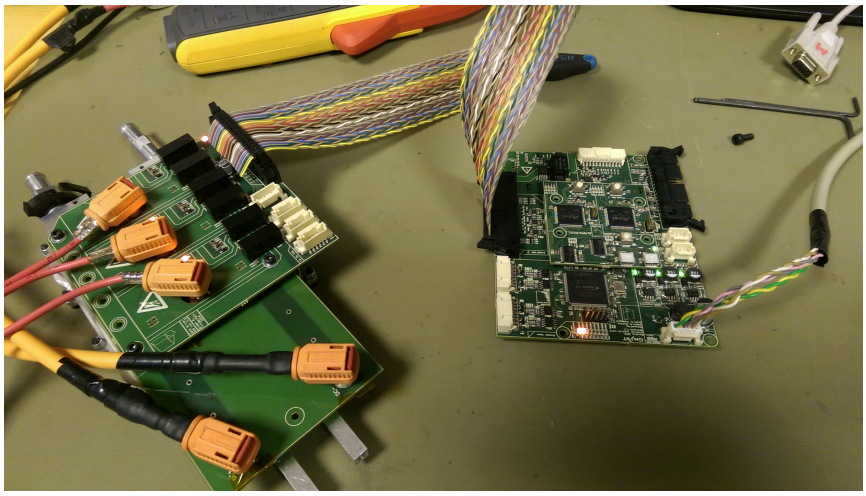


Figure 8.1: The finished inverter system during testing

Based on experiences from the 2015 inverter design, research and input from outside sources, a SiC-MOSFET based inverter hardware system has been designed, manufactured, and tested.

The design estimates from the prestudy are presented in Table 8.1. We see that the R16 concept is estimated to be 10 litres smaller, and more energy efficient while operating at a higher switching frequency than the AMK inverter. Most of these improvements are because of smaller, more efficient transistors allowing smaller capacitors and cooling blocks. Furthermore, being able to program the inverter control ourselves allows us to more seamlessly integrate the inverter into the rest

Stat	**R16**	**AMK**
Switching frequency	16kHz	8kHz
Casing volume	15.8L	24.33L
Power loss	<194W	300W
Total weight	4,350g	11,500g

Table 8.1: Comparative specifications of the AMK and R16 inverters

of the car's systems. This will for example allow wireless tuning and monitoring via the telemetry system.

Care has been taken in the design work to simplify assembly and reparation work, while also keeping the casing size as small as possible. To minimize casing size and weight, gate drivers capable of stacking on the power systems have been designed in-house. The average assembly time of the inverter has been reduced from last year by keeping all fasteners accessible, and enforcing the use of plastic fasteners wherever possible, to eliminate grounding requirements.

The final inverter system is estimated to weigh a total of 4.5kg, nearly the same as last year's one motor inverter, and 7kg lighter than the alternative 11.5kg AMK inverter. If it had been implemented in the car, this would easily have been the biggest weight cut in the car.

The SoC controller concept was abandoned as the main implementation focus, due to slow development and hardware issues. A working base software is however ready for further development in later iterations of the system if desired. The main control software, implemented on an Atmel microcontroller is close to completion, but was not functional when this report was written.

Basic unit testing of the system has been done to verify functionality. The time available has reduced the amount of testing done, especially on the fully functional control system. The current sensor circuit will require somem modifications all other sensors are functioning. The power stage has been shown to work well.

In conclusion, a stable hardware system capable of fulfilling the required task has been developed, and suggestions have been made for further development. If the main controller software is rewritten and built with a focus on making basic functionality work before extending further, achieving a working motor control system should be possible with the present hardware.

Bibliography

[1] AMK. *Motor data sheet, DD5-14-10-POW Formula Student - 18600-B5*, 04 2015.

[2] *Atmel SAME70N Datasheet*. Atmel, 2016. URL http://www.atmel.com/Images/Atmel-11296-32-bit-Cortex-M7-Microcontroller-SAM-E70Q-SAM-E70N-SAM-E70Datasheet.pdf.

[3] *MicroZed Evaluation Kit*. Avnet, 2015. URL http://microzed.org/sites/default/files/product_briefs/pb-microzed-eval-v2a.pdf.

[4] *Avnet product brief: PicoZed*. Avnet, 2015. URL http://microzed.org/sites/default/files/product_briefs/PB-AES-Z7PZ-SOM-G-v7.pdf.

[5] *KIT8020-CRD-8FF1217P-1 CREE MOSFET Evaluation Kit User's Manual*. Cree, 10 2014. URL http://go.pardot.com/l/101562/2015-08-10/9z1/101562/854/KIT8020_CRD_8FF1217_1.pdf. Rev. A.

[6] *Converting Cree Half Bridge Evaluation Board to Use New 900VGenMOSFET*. Cree, 8 2015. URL http://www.wolfspeed.com/downloads/dl/file/id/148/product/0/converting_cree_half_bridge_evaluation_board_to_use_new_900vgenmosfet.pdf. Rev. 0.

[7] *Technical Information - EnDat 2.2 – Bidirectional Interface for Position Encoders*. Dr. Johannes Heidenhain Gmbh, 2015. URL http://www.heidenhain.de/fileadmin/pdb/media/img/383_942-27_EnDat_2-2_en.pdf.

[8] *Mercury ZX5 SoC Module product brief*. Enclustra, 2015. URL http://www.enclustra.com/assets/files/products/soc_modules/mercury_zx5/ProductBrief_MercuryZX5.pdf.

[9] Exar. *SP3485: +3.3V Low Power Half-Duplex RS-485 Transceiver with 10Mbps Data Rate*, 06 2012. URL https://www.exar.com/content/document.ashx?id=639.

[10] Texas Instruments. *Wide-Temperature, Precision INSTRUMENTATION*

AMPLIFIER, 06 2002. URL http://www.ti.com.cn/cn/lit/ds/symlink/ina338.pdf.

[11] Texas Instruments. *Precision, 200-µA Supply Current, 2.7-V to 36-V Supply Instrumentation Amplifier with Rail-to-Rail Output*, 04 2013. URL http://www.ti.com.cn/cn/lit/ds/symlink/ina826.pdf.

[12] Texas Instruments. *ISO1050 Isolated CAN Transceiver*, 01 2015. URL http://www.ti.com/lit/ds/symlink/iso1050.pdf.

[13] Piet Vanassche Joris Lemmens and Johan Driesen. Pmsm drive current and voltage limiting as a constraint optimal control problem. *IEEE Journal of Emerging and Selected Topics in Power Electronic*, 2:326 – 338, 2015. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6808523.

[14] Sungmin Kim, Young-Doo Yoon, Seung-Ki Sul, and K. Ide. Maximum torque per ampere (mtpa) control of an ipm machine based on signal injection considering inductance saturation. *Power Electronics, IEEE Transactions on*, 28: 488–497, 2013.

[15] Lars H. Opsahl. Design and testing of voltage source inverter and motor control system for electric vehicle. Master's thesis, Norwegian University of Technology and Science, 7 2015.

[16] *2016 Formula SAE Rules*. SAE International, 2015. URL http://www.fsaeonline.com/content/2016_FSAE_Rules.pdf.

[17] M. Salcone and J. Bond. Selecting film bus link capacitors for high performance inverter applications. *Electric Machines and Drives Conference, 2009. IEMDC '09. IEEE International*, pages 1692–1699, May 2009.

[18] Shinn-Ming Sue and Ching-Tsai Pan. Voltage-constraint-tracking-based field-weakening control of ipm synchronous motor drives. *Industrial Electronics, IEEE Transactions on*, 55:340–347, 2008. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4401187.

[19] Avago Technologies. *ACPL-C87B, ACPL-C87A, ACPL-C870 Precision Optically Isolated Voltage Sensor*, 06 2016. URL http://www.avagotech.com/docs/AV02-3563EN.

[20] Simen A. Tinderholt and Anders Holter Bjørkto. Design and testing of in-

verter. Master's thesis, Norwegian University of Technology and Science, 12 2015.

[21] Gangyao Wang, J. Mookken, J. Rice, and M. Schupbach. Dynamic and static behavior of packaged silicon carbide mosfets in paralleled applications. *Applied Power Electronics Conference and Exposition (APEC), 2014 Twenty-Ninth Annual IEEE*, pages 1478–1483, March 2014.

[22] *SiC-MOSFET transistor*. Wolfspeed, 8 2015. Prelim.

[23] *Zynq-7000 All Programmable SoC Overview*. Xilinx Inc., 2015. Rev. 1.8.

[24] Joseph Yiu and Carl Williamson. *A Beginner's Guide on Interrupt Latency - and Interrupt Latency of the ARM® Cortex®-M processors*. ARM, 2013. URL https://community.arm.com/docs/DOC-2607.

Appendix A. Source Code

As there are tens of thousands lines of code, multiple complex schematics attached to this project, they will not be added to the thesis itself, but rather as a digital download. Below follows a list of the content.

A.1 MAX10 encoder project

The MAX10 encoder project is attached in the folder `MAX10`. The file `ENDAT22_S.qfp` is a Quartus Prime project, and the entire hardware platform is opened when opening this file. Quartus Prime Lite edition 15.1 was used during thisproject, but Quartus II 15.X should also be able to open this project.

A.2 Atsam code

The Atsam code is in the folder `Atsam`. Opening the file `Inverter_Control_16.atsln` accesses the project.

A.3 Atmega code

The Atmega project is in the folder `Atmega`. It is accessed by opening the file `Inverter_Atmega_16.atsln` with Atmel studio 7.0.

A.4 Zynq project

The Zynq project is divided in two:

Vivado hardware platform files These are in the folder `Zynq/Vivado`. They are made with Vivado 2015.4, but later versions should also be able to open the project. Opening the file `uZed_inv1_120416` to access the project.

SDSoC project The SDSoC project was made in SDSoC 2015.4, which is a licensed program. In order to open this project, a license is required, but the source files are contained in the folders of the projects.

The SDSoC project is made up of three sub-projects:

- The hardware platform project `ledDriver_hw_platform_0`
- The board support package project `standalone_bsp_0`
- The application project `led_blinker`

To open these files as projects, open SDSoC, set the `SDSoC` folder as workspace. On the start screen, `Import project`, and set the workspace folder as root directory. All three projects should automatically be selected. Click `Finish`, and the projects are available in the Project Explorer.

Appendix B. Board Design

The board designs are attached as .pdf files created from Altium. These files include all schematic drawings, layer-view of each signal layer on the PCB, and a 3D-view of the final PCB. the file is interactive, so nets, traces and layers can be highlighted in the pdf. The files are located in the `Schematics` folder.

In addition to the manufactured PCBs, a schematic file of the proposed changed power PCB is attached in the same folder.