UNIVERSITY OF OSLO
Department of informatics

Brushless Direct Current Motor Control Using Sensorless Single-signal Feedback

Cand. Scient Thesis

Kyrre Aalerud

7 August 2006

Abstract

This report is a study into the use of a novel Compound back Electromotive Force (CBEMF) method for driving brushless direct current (BLDC) motors. The study was conducted in an attempt to find a simpler drive-method that would allow fast motor rotation with less powerful microcontrollers. The results show this method to be viable, and that further study may be beneficial. CBEMF signal is a composite signal containing feedback information from all 3 motor phases.

By utilizing the CBEMF principle one can design a controller to run very high speed motors in closed-loop control without the need for sensors. The CBEMF principle will also reduce the processing needs in the controller by a significant amount.

Preface

This work has been done in cooperation with Atmel Norway AS. They have been of great help for me getting to know the Atmel AVR microcontroller series better as well as providing me with a professional development platform with debugger. Without this I can say for certain that this work would never have been completed. A big thanks to everybody at the AVR lab in Trondheim, you are a great bunch and I am grateful of having had the chance to meet you. Special thanks to Jacob for being such a great guy. I think I may owe you a beer or two.

I'd like to thank my friend Fredrik who have been used for proofing. Without you this document would be in a much sadder state. How you can see one missing character in this sea of words I will probably never know.

I also would like to thank my advisor Jim Tørresen for accepting me as his student and for his guidance and patience.

Oslo, August 2006

Kyrre Aalerud

1 Table of contents

2 Introduction

The idea for this work came from a fascination of the physical simplicity of a brushless motor. Few moving parts to wear out together with high efficiency made it appealing to investigate further.

Most controller systems, like the one depicted in [4], use a specialized chip or a very high performance microcontroller to handle the demanding digital signal processing (DSP). The simplified driving scheme was devised after inspiration from the work of Jianwen Shao [1].

To drive a brushless motor using a low priced controller should be beneficial for many actors on the market today. Lowering prices of final product, or increasing profit margins is common business sense. For very small and fast rotating motors, a controller based on a slower micro controller unit (MCU) would also draw less power and be a good option for mobile solutions where the motor driven may be very small.

In this chapter I first go through the background and inspiration for the work as well as the basics of the Brushless DC motor (BLDC) [2.3 Introduction to brushless motors], and the microcontroller [2.4 Introduction to microcontrollers], in [2 Introduction].
Then in [3 Previous works] a quick summary of previous works is done.
I then present the CBEMF principle in [4 “Composite BEMF” (CBEMF)].
In [5 Implementation] I present the toolkits used before covering the software implementation in [5.2 The Software]. This is followed up with the first driver-stage and results from this in [5.3 First driver-stage] before covering the second driver-stage in [5.4 Second driver-stage].
This is followed by the conclusions in [6 Conclusions] before I discuss further research in [7 Suggested further research].

2.1 Problem description

To avoid the need for a fast DSP and other special devices the goal is to get the motor running in a controlled manner using a compound feedback signal and a single analog comparator. This comparator happens to be built into the microcontroller. This reduces the number of external components. Further goals would be to implement common functions such as current-monitoring for over-current control. This can also be used for power-regulation in a constant-current (torque) type motor driving.

Research has not found any experiments or literature describing this single signal feedback proposed for commutation control.

The main goal of this work is to explore the CBEMF [Chapter 2.3.4 and 4] control approach to determine if it is viable and, what benefits or drawbacks it may have. The method is tested by using the already known and accepted method of driving a motor in locked-loop [ref!], combined with the CBEMF method.

2.2 Method

Due to the nature of the main theory of this report, the CBEMF principle and the question of whether or not it will work at all, the natural choice of method is the practical approach. Implementing the CBEMF principle and getting a proof of concept should thus be first stage. Analysis of the results may then lead to improvements and possible new theories.

The alternative to implementation would have been simulation of the CBEMF principle with appropriate software tools. This method was rejected mostly because implementing the system in real life seemed more interesting. In addition, the proper tools for such a simulation did not seem readily available.

2.3 Introduction to brushless motors

Brushless motors utilize the rotating field principle. An applied rotating field transfers energy to a permanent field in the rotor via magnetic interaction. This transforms the field rotation into physical motion. The process of rotating the applied field can be subdivided into discrete steps and is commonly referred to as commutation. A direct correlation to brushed motors here is a slide-assembly that will switch the drive power between the phases using brushes. This slide is called a commutator. The external rotating field is created by static coils with fields pulsing out of phase from each other, making the rotor the only moving part in the system.

The rotating magnetic field principle, though commonly credited to Nikola Tesla in 1882 or thereabouts, was productively employed by mainstream scientists such as Michael Faraday and James Clerk Maxwell in the 1820s. Tesla, however, exploited the principle to design a unique two-phase induction motor in 1883. Michael von Dolivo-Dobrowlsky invented the first modern three-phase "cage-rotor" in 1890. Introduction of the motor from 1888 onwards initiated what is known as the Second Industrial Revolution, making possible the efficient generation and long distance distribution of electrical energy using the alternating current transmission system, also of Tesla's invention (1888) [6]

All benefits of brushless motors stem from being brushless. With a lack of brushes comes the lack of sparks commonly created between the brushes and commutator in a regular brushed motor. This allows the motors to be used in areas where sparks would be harmful or even dangerous. The lack of brushes also reduces friction in the motor allowing for higher efficiency. Adding further to the benefits the lack of brushes being prone to bouncing or overheating at higher rpm allows the motor to run continuously at very high rpm. The practical rpm limit of a brushless motor can be set by its bearings, the strength of which magnets are held onto the rotating shaft or the speed at which the controller can commutate the motor. This limitation is then the lowest value of said limits with an applied satisfactory safety-margin.

The benefit of simple construction also comes from the lack of brushes. Neither the brushes nor the mount have to be included in the motor. This fact reduces both the size and cost of the motor by that of said brush-commutator assembly.

Sensor less motors can also avoid the sensors which otherwise would be the alternative to brushes, thus the name sensor less. The sensors are replaced by resistor/capacitor networks and op-amps, all having much higher reliability and longer service life. The alternative components are placed in the controller instead of in the motor and connected to the motors drive-signals. The controller is usually easily replaceable.

2.3.1 Three-phase BLDC

Mechanically the three-phase Brushless Direct Current motors, commonly called BLDC motors, are very simple. They consist of a stator on which copper wire is wound to create coils and a rotor with permanent magnets. There is at least one coil for each phase, but the number of magnets can vary depending on the design.

The BLDC motor is best driven by a square wave drive signal which is easily made with 3 half-bridges, one for each phase. A half-bridge is an electrical circuit with two transistors, one connected between upper supply rail and load. The other connected between lower supply rail and load. Activating one at a time one can connect the load to either rail. Care must be taken not to activate both at once as this would effectively create a short circuit between the high and low supply rail.

Benefits	**Caveats**
Can handle high rpm	Some torque ripple
Simple drive circuitry for optimal driving	Some audible noise from the commutation
High power to weight ratio	Use magnets (more expensive raw material than copper windings.)
High torque	

Table 1 Benefits of BLDC

2.3.2 Internal design examples

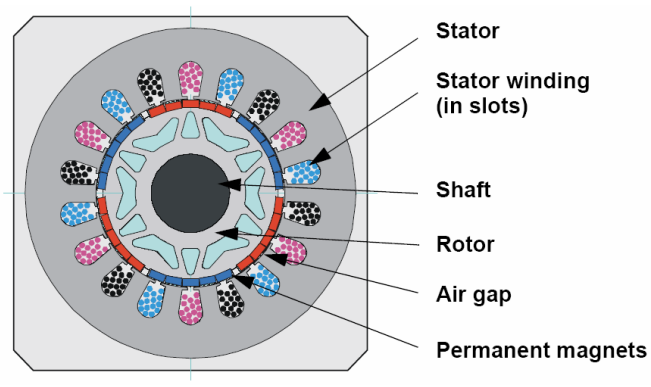


Figure 1 PMSM cross section

In a cross section of a typical Permanent Magnet Synchronous Motor (PMSM) [Figure 1] the windings overlap. We see the stator from above with colored windings in between the teeth of the stator. The windings should ideally fill the slots between the stators. Passing down one slot, skipping 2 slots then up the next, the windings make up coils. Magnets covering the same angular span are found in the rotor. Since more than one of the windings are energized at once they will be reinforcing and cancelling out each other at various places around the stator. In the illustration, each phase has 3 coils wound on the stator with 120° displacement and each coil covers 3 stator teeth, (and also skips 2 slots which are filled by windings from other phases), making up 60°. The overlap is one tooth making up 20° or 33% for each coil to the neighboring two coils. Notice the width of the magnet poles on the rotor in relation to the 60°

angular span of the coils. The PMSM is akin to the BLDC motor in that only the winding changes along with the ratio of magnet width.

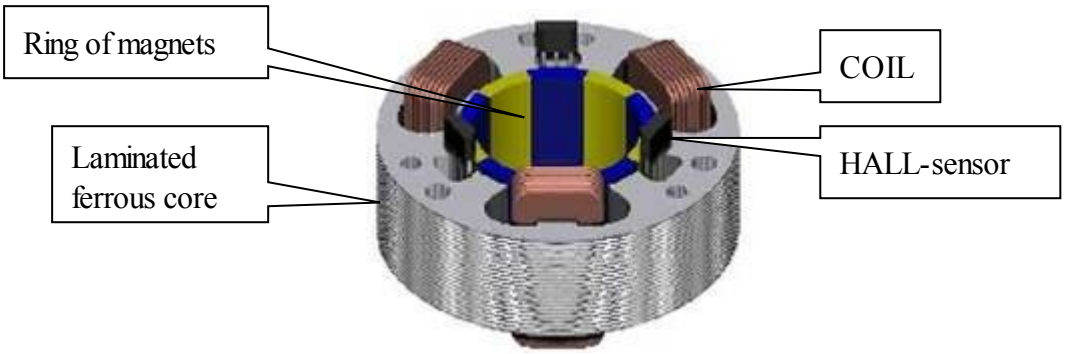


Figure 2 BLDC motor. Modified figure from [10]

In a BLDC motor one would use narrower magnets and usually non-overlapping windings (winding each tooth as one coil), to create distinct coils around the stator. (See [Figure 2]) The magnets influence approximately one pole at a time and thus create sharp distinct BEMF-transitions that when masked by the drive-voltage 2/3 of the time appears as trapezoidal BEMF. The figure depicts a motor with 3 hall-sensors but for sensor less driving they would not be needed and consequently not included.

The motor in [Figure 2] has one coil for each phase and 10 magnets forming 5 pairs. Notice the width of the stator teeth in relation to the magnets. However, this is not a symmetrical design and thus, this motor will be subject to some extra mechanical noise as the forces in the motor will be pulling the shaft to the side. Normally, 6 coils is a minimum to create a symmetrical motor without such issues. The coils would then form pairs with coils on opposite sides of the rotor. Two and two opposing coils would be connected in parallel or series to be driven by the same phase. When energized the sideways pull on the rotor would be cancelled out and only the intended rotational force would be applied to the rotor. This also reduces strain on the bearings. Readers interested in the intricate workings and interactions between the number of coils and magnets that make up a brushless motor should read Dr. Duane Hanselmans book [3] where these matters are discussed fully.

2.3.3How to drive a BLDC motor

Driving a BLDC motor is done using a trapezoid waveform and requires three half-bridges [Chapter 2.3.1] capable of fully discrete control. Single input half bridges will not work for normal trapezoid type driving. One must be able to turn both high and low side off, allowing a phase to float in order to drive the motor correctly. By energizing two phases at a time, in a certain pattern, the rotor is driven around. The next step must be timed correctly to keep the motor turning the right direction. Too slow switching will brake and possibly even pull the rotor backwards causing it to stutter.

The reason for the driving to be called trapezoid is the shape of the driven phases in conjunction with the motors BEMF signal during the two non driven parts of the commutation cycle. In the delay following the low drive periods the signal will move from low to high while after the high drive period the signal will move from high to low, thus completing a trapezoidal waveform.

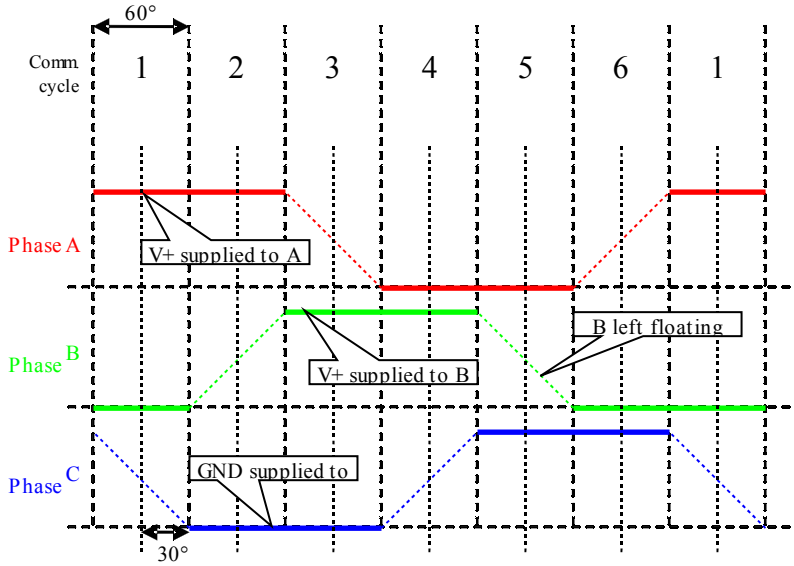


Figure 3 Commutation steps for BLDC visualized

If one designates the phases A, B and C, a way to drive the motor a full commutation cycle would be to start by applying positive voltage to A, and negative to B as seen on first row of [Table 2] and [Table 3]. (Starting point just selected by preference.) The next step, one would keep positive voltage to A, but switch the negative supply to B off, and at same time start supplying negative polarity to C. The next step is to shift the positive supply from A to B. Next comes shifting the negative from C to A. Final two steps is first shifting the positive to C and then at last shifting the negative to B. This completes the cycle making the next step equal the first step again. The entire cycle is visualized in [Figure 3]. Notice how the 3 waveforms are offset 120° from each other. For clarity an extra step is shown in [Figure 3] equal to step 1. The cycle repeats for as long as the motor is run.

To reverse direction of the motor, the controller simply runs the commutation steps backwards. There are no electrical connections that need changing.

In a table, the sequence would look like this:

Commutation step	Phase A	Phase B	Phase C
1	+	-	
2	+		-
3		+	-
4	-	+	
5	-		+
6		-	+

Table 2 Commutation steps as polarity pr. phase

Another way of annotating the steps:

Commutation step	+	-
1	A	B
2	A	C
3	B	C
4	B	A
5	C	A
6	C	B

Table 3 Commutation steps as phase's pr. polarity

Only one of the motors phases will be changing at a time and one of the phases will always be left floating. This is essential for reading the BEMF signal, it being available on the non-driven phase.

It follows from these 6 steps that one electrical revolution needs all 6 commutations. An interesting fact about these motors is that they do not have to rotate a full 360/6 degrees between each of the steps. The actual rotation of the motors output shaft is determined by the number of magnetic pole-pairs in the motor. The coils in the motor will be "working against" the magnets and in a complete commutation cycle they will have moved an N and an S polarized magnet through their field of influence. If there are multiple pairs of magnets, multiple cycles must be performed to cycle through all of them. The net effect of this is a smaller movement pr cycle allowing motor to deliver higher torque at less speed. It creates the ratio between the electrical rpm (e-rpm) and the actual rpm of the motor equal to the number of magnet pairs. This can be looked upon as a sort of magnetic gearing.

The actual number of magnets and their widths in relation to the number of coils in the motors stator is a complex design problem I will not delve into. Suffice to say the primary concern for driving the motor sensor less is to have a magnet interacting with each of the energized poles at any given time. This means that a motor having 6 coils (2 coils for each phase) will benefit from 4 magnetic poles over 2 so that one of the poles in each coil will not waste energy creating a magnetic field without any magnet to interact with.
The 6 poled motor with 4 magnets would have a rotational factor of 2:1 making two electrical cycles (12 commutations) equal one actual revolution of the shaft. Similarly a 9-pole motor can use 12 magnets (6 pairs) to gain a ratio of 6:1 or 6 magnets (3 pairs) to gain a ratio of 3:1.

The benefit of multi-pole motors is as mentioned increased torque. Another benefit is the reduction of the torque ripple. The downside is that to drive the motor at high rpm one will need a faster controller. A motor with 12 magnetic poles, 6:1 ratio and a required actual rpm of 10.000 would need to be commutated 6.000 times pr second. This follows from 10.000 rpm equaling 60.000 commutations pr minute (cpm) and thus converted to commutations pr second (cps) becomes 1.000, which in turn becomes 6.000 when the 6:1 ratio is taken into account. Using the above example of 12 magnets we see that each commutation lasts for 166.66µs when running at the required 10.000 rpm. The previously mentioned 6 magnet motor would have 333.33µs long commutation steps at the same 10.000 rpm.

Speed control is done by regulating the voltage supplied to motor. To do this, the drive signals are pulsed with a high frequency and the ratio between on and off time is adjusted according to required power level. The frequency remains the same, but the pulse width changes. This is called Pulse Width Modulation. (PWM) Using the full timer resolution available (8 bit), and full system clock as timing source (8 MHz) the PWM frequency becomes 31.25 KHz. Any other frequency can be chosen by adjusting either clock frequency or the accuracy of the timer. It should be noted that PWM adds extra noise to the signals and changes the driven periods of the waveforms to look like blocks on the plots. See Figure 21 for an example of the signal with PWM present, (with 60% duty cycle), and Figure 23 for an example where PWM is not present. (PWM is at 100% duty cycle.)

2.3.4Sensorless driving

Sensorless driving relies on the BEMF which comes from the windings being influenced by a moving magnetic field. The BEMF opposes the main voltage applied to the windings according to Lenz's law. This law states that the EMF induced in a conductor moving perpendicular to a magnetic field, tends to oppose that motion. When the motor is running at full power the BEMF generated will equal the applied voltage minus the internal losses assuming no load. Under load, the difference between the BEMF and the applied voltage would be sufficient for the motor to draw enough current to deliver the required torque. This is described in [2].

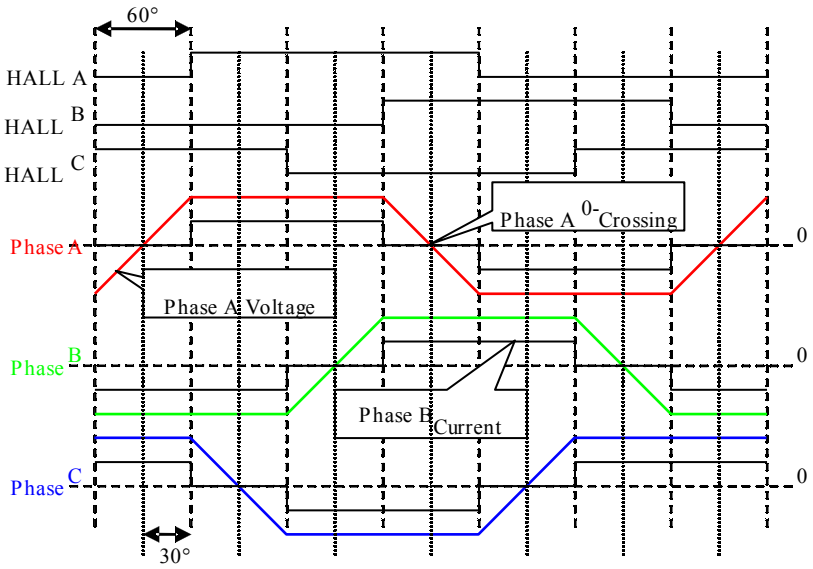


Figure 4 Traditional hall sensor outputs relative to the back-EMF

By sensing the BEMF one can deduce the rotor position and thus when it is time to commutate. Normally the hall sensors would show when it is time to commutate. In [Figure 4] the BEMF is shown as voltage on each phase relative to 0. 0-crossing then happens 30° advanced of each hall-sensors state change. This can be used to predict when one should commutate instead of trying to detect it directly. In [Figure 4] the current for each phase is also drawn in. The current would be synonymous with the applied power to that phase at the time. When applied current is 0, the phase is left floating, allowing the BEMF to be sensed. It is important to note that 0 in this regard is not necessarily 0 volts. Since the motor is run of a DC bus, 0 is V/2 for the controller.

2.3.5Areas of use

For many years brushless motors were used mainly in industry. Later they became the standard in many consumer devices requiring high rotational speeds, silent operation, low power consumption and/or accurate and stable speed control combined with long life. In later years brushless motors have been introduced to the consumer market in the hobby division. Now brushless motors and controllers are even sold in hobby-shops for use in model cars, boats and planes.

CD-ROM drives and other disc-reading devices such as hard-drives, DVD-ROM and floppy drives generally use sensor-based BLDC motors. A good reason for using them in hard-drives is the lack of brushes creating ferrite dust. Hard-drives need to keep their internals dust-free for reliable operation. Since brushless motors have long life and silent operation they are also used in low cost fan-applications. Here pulse-mode or switched reluctance motors are used. These are also sensor-based and generally low-powered. For higher power asynchronous motors running on varying number of phases are commonly used. Fuel pump systems can also benefit from a low maintenance, high efficiency motor where the possibility of sparks is eliminated.

To give some idea of how versatile a brushless motor can be some examples of weight, size and power is in order. To start small, a Feigao 1208430S weighs only 10.5 grams. It has a diameter of only 12mm and a length of 22mm. Yet, for model plane use this motor can handle 14 watts. A slightly larger motor, also intended for model planes is the AXI 2204/54. This motor weighs only 24.5 grams and measures 27mm in diameter. This motor can handle approximately 45 watts. Motors are available in rising weight/power combinations up past 1kw for model use. It should be mentioned that industrial motors have much lower power to weight ratio as their safety-margins are much higher. This is related to safety and reliability. Model usage pushes the motors close to their failing points while industrial use is much more conservative.

Figure 5 AXI 4120 motor from [11]

To compare model usage with industrial, the industrial brushless motor AM-750M/H from Adlee weighs 8000 grams. It can output 750 watts (rated) with a max output of 1500w and is designed to meet industrial standards. Comparing it to the AXI 4120/14 from Model Motors it becomes very clear that power of brushless motors can cover a wide range. The AXI 4120/14 [Figure 5] only weighs 320 grams and yet it handles 869 watts at 83% efficiency according to tests done by Model Motors. This is close to 1 horsepower output in a 320 gram motor. The max output of this motor is however around 800w showing specifications having smaller margins. Note that in addition to the manufacturers setting different limits and margins, the AXI-line of motors are out-runners. That is, they are arranged so that the rotor fits around the stator making up most of the outside of the motor. This increases the diameter of the working area of the motor. That is the area where the electromagnetic fields are performing work to turn the shaft. This larger diameter increases torque while reducing rpm. The general design of an out-runner increases power to weight by eliminating unnecessary mass in the motor. There is no housing and only a minimal foot attached to the stator to allow mounting. Another factor in the high power to weight relationship is the ability to cool the motor. The open design allows air to flow through the motor under operation. This may not be good in a dusty or otherwise contaminated environment.

2.4 Introduction to microcontrollers

A microcontroller is a small computer in a chip. It has an arithmetic logic unit (ALU) together with supporting hardware such as working memory, program-memory, I/O system and possibly other auxiliary systems. All parts are integrated into a single chip. So called "System on Chip".

Models can be chosen depending on the number of I/O pins and auxiliary systems required. However, physical dimensions rely mostly on packaging. Modern packages such as the micro lead frame (MLF) have reduced the devices footprints during the last years allowing MCUs to be used in new applications. Anywhere a simple function or appliance can be made more practical, faster, and more intelligent or in some other way benefit from a more sophisticated control than simply on/off you may expect to find a MCU. Toasters that monitor the change in color of the bread or electric shavers that tell you when they should be charged or blades changed based on your usage statistics have already made their way into many homes. Scales that calculate how you should adjust your diet based on previous measurements and your target weight is also available thanks to microcontrollers.

2.4.1 Atmel AVR family

The AVR family from Atmel is an 8-bit RISC based microcontroller family. The AVR architecture has approx 130 commands. There is a free compiler and editor package for C available called WinAVR. It includes the GNU GCC compiler for the AVR platform handling both C and C++. There is also a professional compiler from IAR. The IAR workbench is also available in a code-size limited educational version called IAR Kickstart. This version is used in this project.

In a general AVR device one can expect to find: (values for ATmega64 in parenthesis)

- A bank of program memory (64kbyte addressed as 32k x 2)
- A bank of working memory (4096 byte)
- A bank of EEPROM (2048 byte)
- A set of I/O ports (53 I/O lines total)
- At least one timer (Two 8-bit and two 16 bit timer/counters)
- Often a built-in oscillator (Internal oscillator as well as separate watchdog oscillator)
- Some standard interrupt vectors (35 interrupt vectors)
- Fuse bits to set operation parameters
- Lock bits for varying protection of the internal memory banks
- A loose collection of integrated features varying in presence and numbers such as
    - Analog to digital converters (ADC) (Present, 10 bits resolution)
    - Analog comparators (1 present)
    - Timers, possibly both 8 and 16 bit. (two of each)
    - PWM generators (Multiple. Timer 0,1 and 2 can all generate PWM)
    - General interrupt vectors
    - Feature related interrupt vectors

2.4.2Used in the project: Atmel ATmega64

The ATmega64 is used in the project mainly due to availability of the device and its mid-level pricing. Having multiple timers, a multiplexed input ADC and an integrated analog comparator it allows for testing theories on motor driving using a relatively low cost component. It is available at Farnell for 125-150 NOK depending on model with a 25% price reduction already at 10 units.

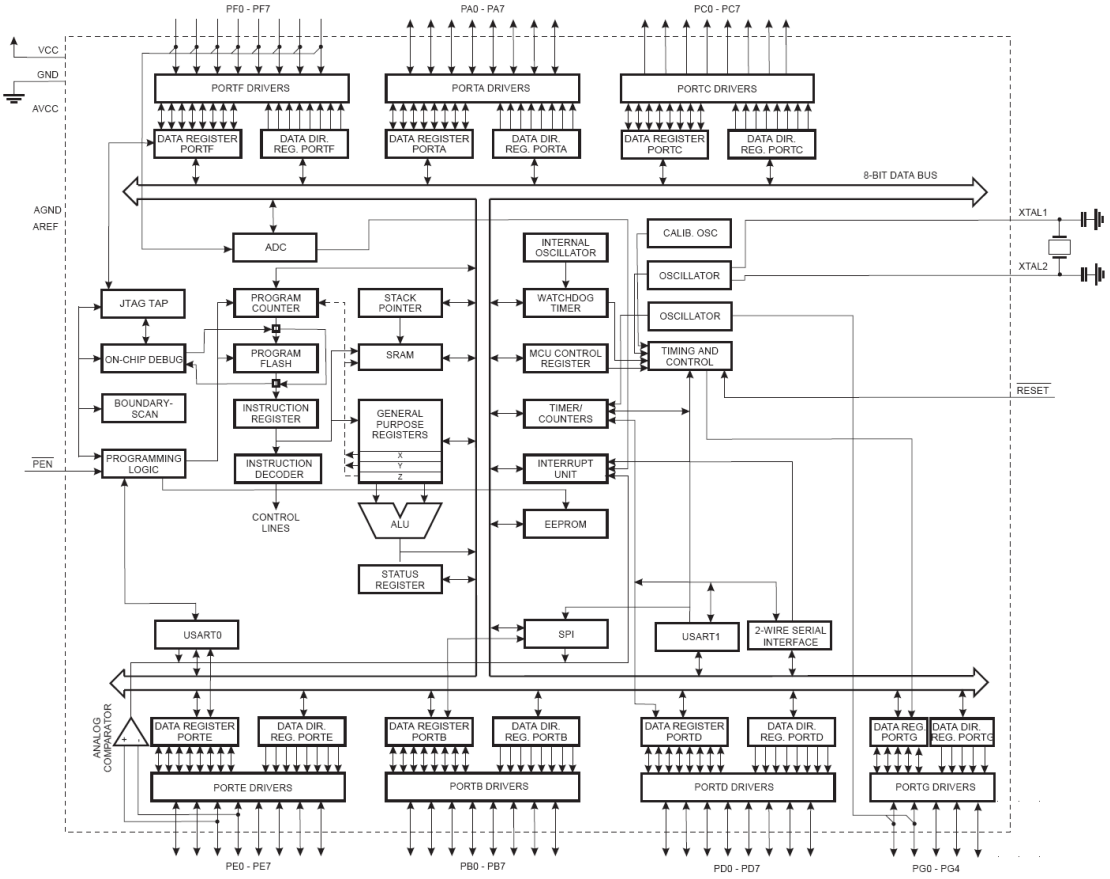


Figure 6 Block diagram of ATmega64

The ATmega64 has two 8-bit timers as well as two 16-bit timers. These timers can be linked to different clock sources. One option is to use the system clock directly. Alternatively it can be passed through a counter circuit that will reduce the clock-rate by a selected factor called a prescaler. Prescaler is not seen in block diagram on [Figure 6] but is part of each timer included in the timer/counters block. Of further interest are 53 programmable I/O lines and the ability to run at up to 16 MHz delivering up to 16 MIPS. To ease developing it supports JTAG (IEE 1149.1 compliant) interface with boundary-scan capabilities and extensive on-chip debugging support. The JTAG interface also allows reprogramming Flash and EEPROM memory, fuses (for device setup) and lock bits (for device memory protection). Another interesting feature of the ATmega64 is the availability of interrupts. Having 35 interrupt vectors handling different interrupts a lot of the code can be triggered by interrupts when needed. The analog comparator has an interrupt that can be used to trigger the timing code allowing fast response to any changes. The ADC can also run interrupt driven, allowing the MCU to perform other tasks while waiting for the ADC to complete a conversion.

The internal oscillator in the ATmega64 allows it to generate its own system clock. No external components are necessary. Applying power to a properly configured ATmega64 would allow it to start executing its program after a short startup delay. Applications that do not require very high timing accuracy can benefit from space requirements and lower cost when using the built in oscillator. According to figures in the datasheet the internal oscillator frequency varies both by supply voltage and temperature. There will also be manufacturing variations. To compensate for this, the oscillator can be calibrated both during design manufacture and testing. It can also be calibrated under normal use, for example if an accurate low-frequency clock source is available as input.

Hopefully the ATmega64 will prove to be overpowered for this control approach, opening up for the possibility of running the motor with an even lower priced MCU.

3 Previous works

There have been a lot of works done regarding brushless motor control in the last years. It is a field with increased focus and a fast growing market. The works on which this report is based on will be presented in brief.

3.1 The work of Jianwen Shao

"Direct Back EMF Detection Method for Sensorless Brushless DC (BLDC) Motor Drives" from September 2003, submitted to Virginia Polytechnic Institute and the State University. [1] ETD: etd-09152003-171904

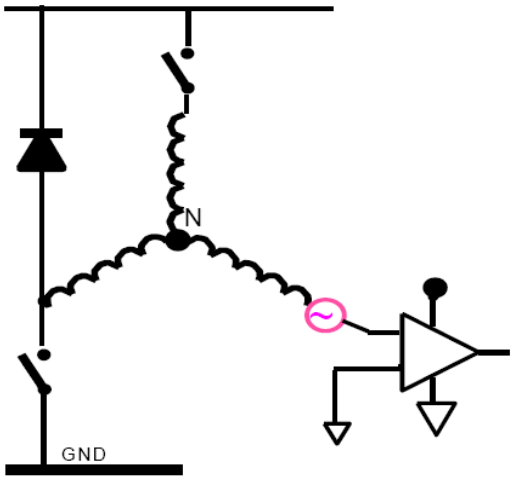


Figure 7 Jianwen Shao proposes this detection method.

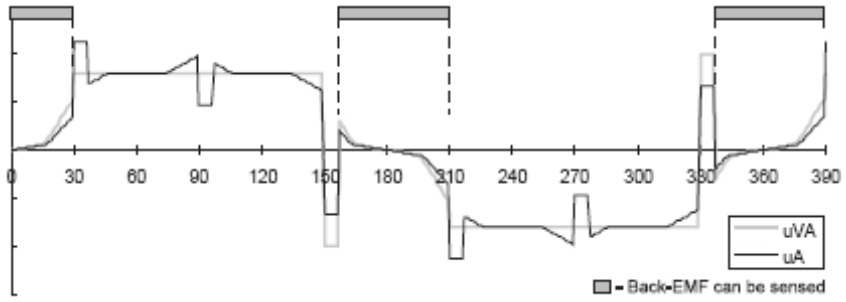
Describes the virtual neutral point which in essence is a composite BEMF, but uses a non-driven phase referenced to ground for BEMF sensing. 3 inputs are used for main EMF control together with 3 comparators. In Figure 7 only one phase and comparator is shown. The upper part of the figure shows the high supply rail. This supplies the positive drive voltage. GND as labeled is the ground reference and lower supply rail.

The idea is to disregard the comparators outputs during PWM-on period so as to detect 0-crossing only during PWM-off periods. To overcome this limitation Jianwen Shao also implements a way of referencing the comparators to V/2. That is half the DC bus voltage. He can thus detect the crossing during both high and low period but is still required to monitor 3 inputs to do so.

This work was the first inspiration towards the new proposed driving method.

3.2 AN1913 by Freescale using DSP 56F80x

Using DSP 56F80x series MCU Freescale implement a 3-phase sensorless motor control based on analog to digital conversion (ADC).



In [Figure 8] Freescale denotes what portion of a channels waveform can be used for BEMF sensing. The gray bars in the figure marks the portions of the wave-form where the phase is not driven.

Figure 8 Phase Voltage Waveform, figure 3-6 from [4]

The shaded areas in [Figure 8] show the validity of equation 3-6 from [4]. Equation 3-6 states that the branch voltage can be calculated as:

$$u_{VC} = \frac{3}{2}u_{ix} \qquad \text{(eq 1)}$$

Where:
$x = \{A, B, C\}$
Since the phases A, B and C are 120 degrees apart in rotation, one is always valid.

Without PWM the entire range depicted will indeed be valid for detection of the BEMF 0-crossing event. In reality the range is chopped up by the PWM drive. ADC acquisition is synchronized with the PWM to avoid voltage spikes from switching. This effectively defines the temporal resolution of which 0-crossing can occur. Freescale only samples once pr PWM cycle making the 0-crossing detection frequency equal to the PWM frequency.

This previous work shows the validity of the BEMF waveform.

3.3 AN970 by Microchip using PIC18F2431

This is another design using a purpose-built controller chip. It relies on ADC but opposed to [4] it also contains a digital filter. This allows it to attempt to filter out the high frequency PWM noise from the sampled BEMF signal. Microchip mentions the virtual neutral point and shows figures very similar to Takaos. However, they also use the virtual point relative to the phases. They chose the ADC method after considering the comparator method Takao describes.

The main reason for including this work was the good figures it contained regarding the motor waveforms.

4 “Composite BEMF” (CBEMF)

The novel Composite Back Electromagnetic Force (CBEMF) method presented here has not been found used in previous works. Most searches have been conducted online with multiple search engines.

4.1 Basic idea

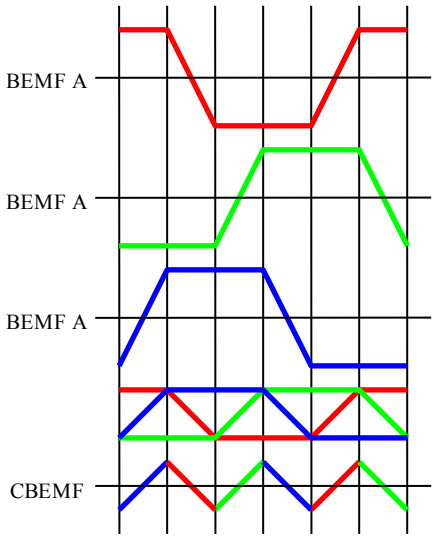


Figure 9 CBEMF signal origin.

When driving a BLDC motor two phases will, at any given time, be energized by the controller. One will be connected to the positive and one to the negative supply-rail. If one combined all 3 phases into a single signal, the two driven phases would cancel each other out leaving only the non-driven phase signal referenced to v/2. [Figure 9] illustrates the CBEMF principle. The angled lines in the figure contain the BEMF signals, covering the timeframes where the phase is not driven either high or low.

Since the CBEMF signal contains only the sections of the signal where eq is valid, it can be used continuously as the source of triggering information. See [3.2].

The combined signal will allow the detection of both peak and 0-crossing events. Peak events denote when a commutation should occur directly while a 0-crossing event will happen half-way between each commutation giving fore-warning. Triggering on a peak would require the detection of the peak and thus the knowledge that the signal has passed a peak. This can not be done without letting the optimal moment of commutation pass. Using prediction to commutate at the correct moment will unfortunately prevent the detection of the peak event, thus preventing the algorithm from adjusting the timing if it was too early. The 0-crossing will give ample warning and allow for a significant change in timing on a per-commutation basis. It will also be able to do accurate advance-timing where the switching is deliberately done ahead of the predicted peak.

When the signals are combined the resulting CBEMF signal will have triple the frequency. This is due to it containing the rising and falling flanks from all three phases interleaved, two flanks from each phase. Filtering should take this triple frequency into consideration. It is worth noting that I call the BEMF signal flanks not edges as I do not wish to confuse them with rising and falling flanks of square wave signals. Each flank of a BEMF signal will cover 1/6 of the cycle time and is a signal returned from the motor. It is not caused by a low current driver as with slow rising and falling edges.

4.2 Suggested benefits

Monitoring a single signal should mean less overhead in the microcontroller. If, in addition, this monitoring can be done using an integrated hardware unit the overhead would be reduced even more and response would improve. Result of low overhead and fast response should be the ability of a moderately clocked MCU to run a motor at high rpm with high commutation accuracy while still having available computation cycles to perform other tasks.

The common ADC method of BEMF detection requires a fast ADC and timing to avoid the PWM switches. Usually this results in a single sample being taken each PWM cycle, or at best two. One sample can be taken while PWM is active, and one while passive. Very fast ADCs may allow implementations with more samples but this is rare. As rpm increase, the number of PWM cycles and possible ADC samples pr commutation cycle will be reduced. When the number of samples is too low to accurately detect the 0-crossing and predict commutation, the controller must prevent the motor from accelerating further or it will loose synch with motor. The CBEMF method should be capable of keeping synch at very high speeds as the delay between the actual 0-crossing event and detection time is minimal. ADC based solutions have to wait for the ADC conversion to finish while the CBEMF method only has a small propagation delay through the analog comparator.
The short response time can increase the accuracy of the timing reducing current consumption over similar ADC based systems.

4.3 Algorithms

When the motor is running the MCU should be able to wait for the analog comparator to signal that a 0-crossing event has occurred. By measuring the time between either each 0-crossing or between the time of commutation and the next 0-crossing, the MCU can calculate the optimal delay and time for the next commutation to occur. This 0-crossing can be triggered using interrupts and would thus require very little of the MCU processing time, only being called upon when there has been a change in the analog comparator output. A flowchart of the interrupts is shown in chapter 5.2.3.1. The drive algorithm of waiting for 0-crossing and calculating a delay before commutation is a generally accepted standard that will be recognized in most works dealing with BEMF-based driving.

Due to switching noise the system would need to have a hold-off period after commutation where it ignores the signal from the comparator. This period would be fairly short but may be a limiting factor in the maximum rpm possible. If the 0-crossing occurs within the hold-off it can not be detected. Maximum electrical rpm would thus be less than:

$$\frac{60}{T_{holdoff} \times 2 \times 6} \qquad \text{(eq 2)}$$

Or simply:

$$\frac{5}{T_{holdoff}} \qquad \text{(eq 3)}$$

The formula is derived from the fact that if the crossing is detected immediately after the hold-off has ended, and the delay in the filter is not compensated for, the appropriate delay before commutation would be equal to the hold-off. Thus, the hold-off is multiplied by 2. Since there is 6 commutations pr cycle, it is also multiplied by 6. 60 converts from rps to rpm. A hold-off of 50µs would set an absolute maximum limit at 100.000 rpm. The required hold-off would be related to the motor characteristics and the filter implemented for the CBEMF circuit. As the filter in the CBEMF circuit delays the signal, it will also delay the initial switching-noise after each commutation. Softer switching would create more heat, but should allow for a shorter hold-off and greater maximum rpm. Softer switching can be implemented by reducing the drive current of the FET drivers.

5 Implementation

To test different algorithms and to get a good idea of the computing requirements for the CBEMF principle, implementation was done. Proof of concept was the first goal.

5.1 Toolkit used

The software package AVR Studio from Atmel allows for writing code in assembler and simulating this code in a virtual MCU. The suite also interfaces with the development platform STK500 allowing the user to set parameters for the board and program any target MCU mounted on the board. The development platform STK500 contains a programmable power supply and oscillator as well as a generic programmer for Atmels AVR line of microcontrollers. There are sockets for different form factor AVR chips as well as expansion sockets along the edges. The STK500 is depicted in Figure 10. STK500 complements the simulator by allowing code to be tested out in actual circuits. This development board has headers for all I/O signals allowing it to interface with application specific hardware as required.

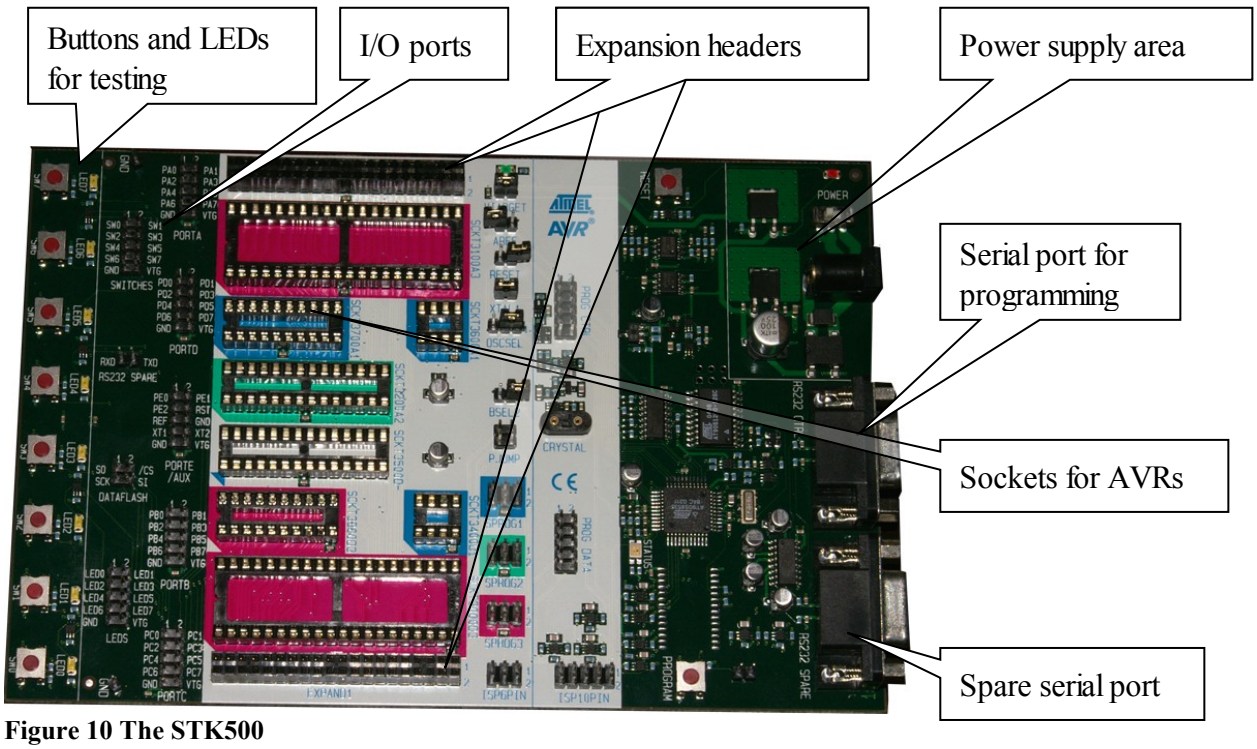


Figure 10 The STK500

To accommodate the ATmega64 an additional board (depicted in Figure 11), was needed called STK501. This board also features the JTAG header used for debugging. Together with the JTAG ICE MK-II, the AVR Studio software and STK500/501 setup allow full simulation/running and debugging of the ATmega64.

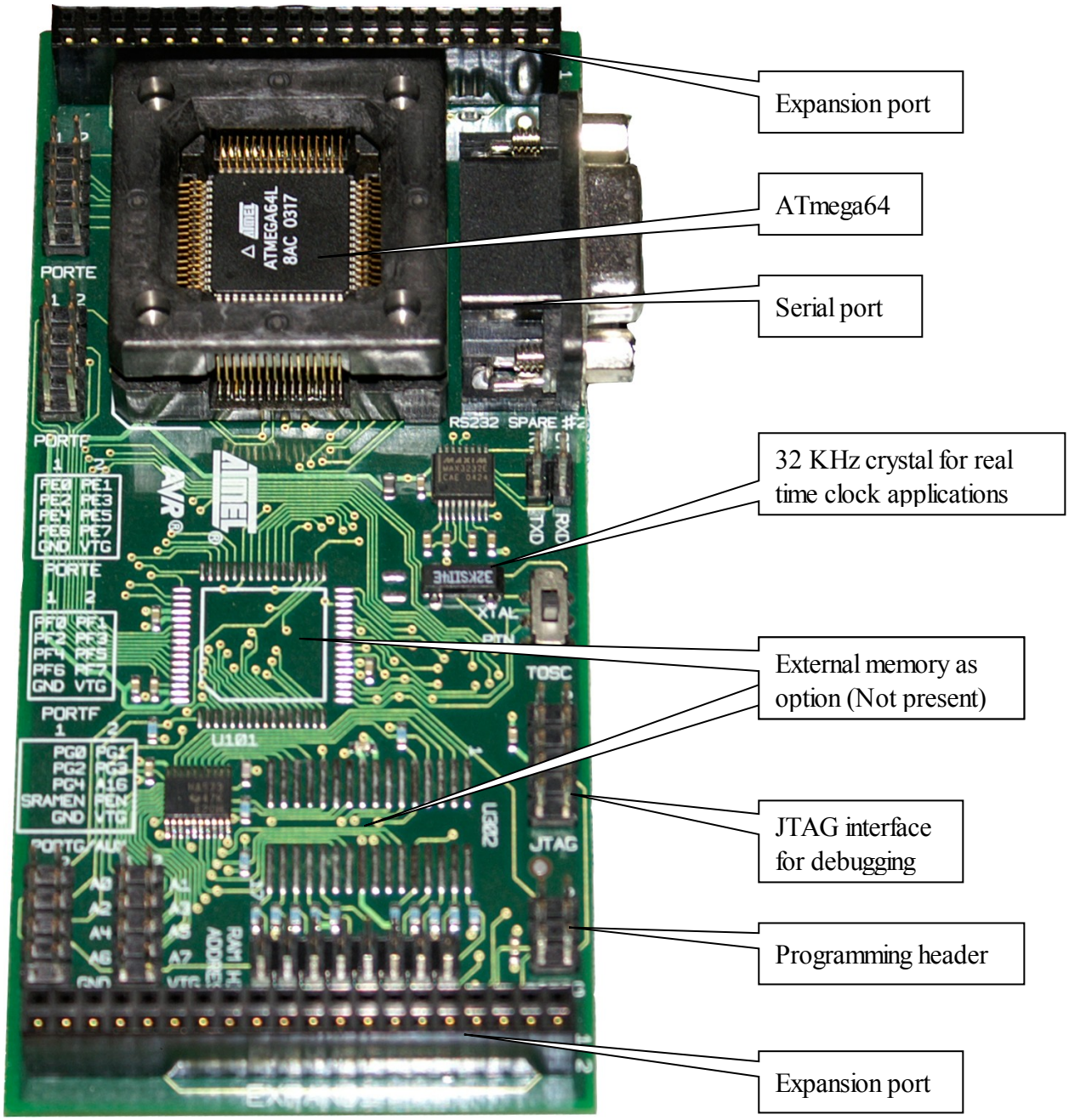


Figure 11 The STK500 add-on board

IAR supplies a Kickstart edition of their IAR C compiler. It has a code limitation of 4k but is otherwise fully functional and integrates very well with AVR Studio to provide a complete high-level development platform with powerful debugging capability. The IAR C compiler for AVR produces efficient and compact code without deviating from the ANSI standard more than necessary.

5.2 The Software

Software for the MCU was written entirely in C using the IAR Kickstart embedded workbench.

There were two driver-boards used in this work, the first being originally for hall-sensor driving. The second was designed to implement CBEMF specifically. The software flow as described here applies to both driver-boards. The only differences would be pin-assignments. The two driver-boards are covered in chapter 5.3 and 5.4 respectively.

5.2.1 General Flowchart

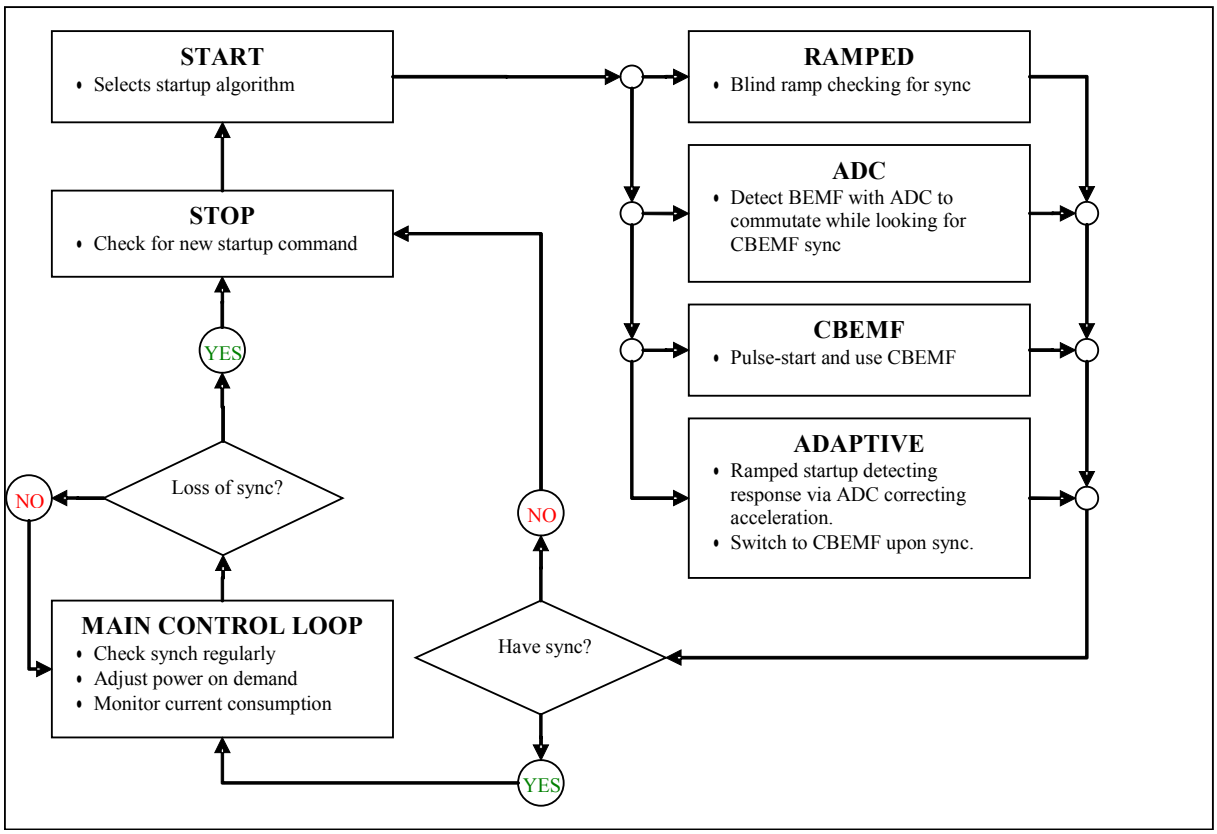


Figure 12 Flowchart covering main startup flow.

As can be seen in [Figure 12], the software flow has 3 main states where it is either stopped, starting or running in a closed-loop. The Start function selects the startup method to be tested. The main loop has to check for problems and abort running if any are detected. In the stopped state, it waits for a signal to indicate a start-up is requested.

5.2.2 Startup algorithms

Multiple algorithms were planned to compare ease and reliability of startup with different motors and loads.

5.2.2.1 Ramped startup

In a ramped startup the motors commutation frequency is ramped. The motor is simply commutated very slowly at first and accelerated up to a preset rpm. If the power applied is not varied, the motor will have most torque while rotating slowly making a startup that accelerates fast to start with and slows down as it closes on the motors ideal rpm relative to the applied power. Ideally one should be able to switch to CBEMF running as soon as the

BEMF from the motor is strong enough to give proper triggering of the analog comparator. While there are large differences between the motors ideal rpm and the driven rpm the signal is very noisy and does not necessarily contain the required information. The initial current drain will also be high unless it is reduced by applying PWM.

This algorithm would not handle different motors well. It would also have difficulties if the load varies between startups.

5.2.2.2ADC startup

ADC based startup utilizes the built in ADC in the MCU to analyze the motors responses during the startup phase. By timing the ADC samples in regards to the PWM switching one can get a fairly clean signal. Ideally one should be able to regulate the acceleration to keep the motor running close to its ideal acceleration curve.

If one can get reliable feedback from the motor this algorithm may handle different motors and loads fairly well. This start method will however require a lot of processing power to analyze the motors feedback.

5.2.2.3CBEMF / Pulsed startup

The idea behind the CBEMF / Pulsed startup is to give the motor a single pulse to start rotation and allow the CBEMF to pick up control directly. If the motor can gain enough momentum to deliver a valid CBEMF signal, the MCU can calculate the next commutation and enter a phase-locked running mode immediately. If this will cause erratic oscillations on startup should be interesting to find out.

If one can get the controller to lock in early this startup should handle most motors and load changes between startups without any adjustments to parameters.

Heavy cogging will cause problems with this method as it will prevent the motor from rotating without any power applied (freewheeling) while a CBEMF timing interval is being acquired. Cogging is described by Dr. Duane Hanselman [3 Chapter 4 and 9.2] as positions during rotation where the magnets in the rotor align with maximum amount of ferromagnetic material.

5.2.2.4Adaptive startup

Under previous attempts, a special characteristic of the CBEMF signal was observed. This is described under chapter 5.3.3.1. The idea was formed that using a ramped startup that adapts the ramp to this characteristic may allow for faster startup times.

Due to time limitations this method was not fully implemented nor tested. It does however seem like an interesting method.

5.2.3Running

When the motor is running, the algorithm for startup is no longer needed. Instead a simpler approach can be used to keep the motor running and to allow for higher speeds.

5.2.3.1Main drive loop

The main drive loop monitors statistics on running including current consumption and possible fault states. The actual commutation and timing is left to interrupts. Any non time-critical calculations can be done during the idle time, scheduled from the main loop. This could be calculations for assessment of actual rpm based on the commutation rate. There could also be calculations for target commutation rates based on wanted rpm before adjusting the PWM duty-cycle.
The main drive loop could even be running control and timing for additional systems in appliances such as a washing-machine or dryer.
Whenever the interrupt conditions for the 3 events **commutate**, **hold-off** and **emf** are true the main code will be halted and the required functions will be performed without much delay. See [Figure 13] for a visualization of the functions. Note that the decisions (represented by diamonds) are not software decisions, but happen based on interrupt flags in the microcontroller. Each state deactivates its own trigger, and activated the trigger for its succeeding function. The events thus occur promptly in the proper order with minimum processing needed. The driving of a sensorless motor was described in chapter 2.3.3.

For increased accuracy, all delays in system can be taken into account. As long as the delay is larger than the minimum measurable time in the system, it can be accounted for and subtracted from the delay used to time the next commutation. Using a timer clock of 1 MHz should allow for a wide RPM range from 150 to around 50,000 without too poor temporal resolution. There would be a 250 rpm change in commutation speed pr step at 50,000 rpm. To achieve higher accuracy as the motor gains rpm, one could switch to a higher frequency timer clock simply by adjusting the prescaler. 8 times faster clock is possible switching to a 1:1 prescaler and running the 8 MHz system clock as timesource. 8 MHz is the highest clock frequency available with the internal oscillator.

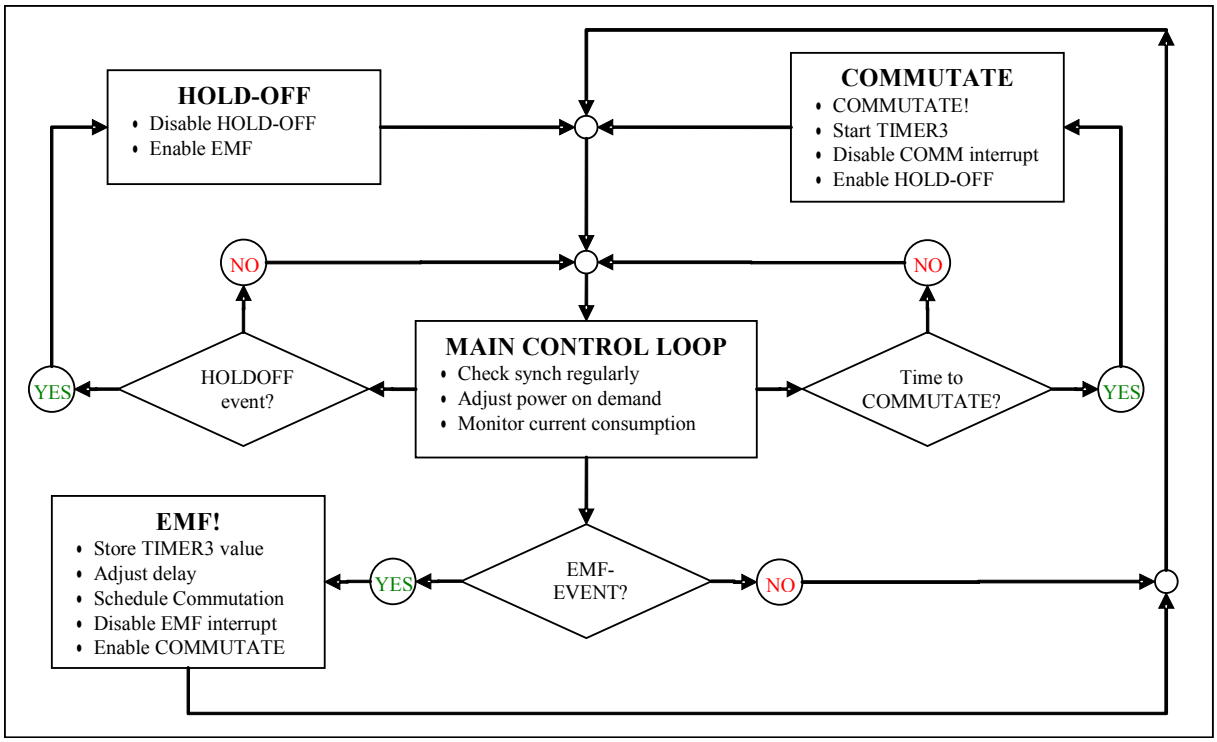


Figure 13 Flowchart of main-control loop with drive-interrupts.

5.3 First driver-stage

To drive the motor a driver-board was required. To test the basic idea a simple board was used. This board was provided by Atmel as a prototype ready made.

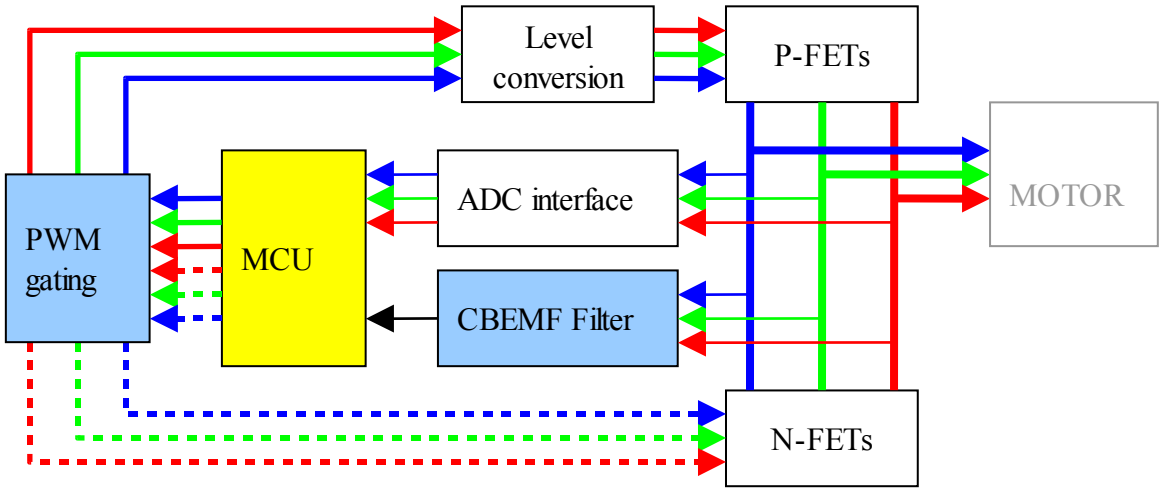


Figure 14 First driver-stage block diagram

The first driver-stage consisted of 3 half-bridges. Each high-side was driven by a small FET providing active pull-down while having a resistor provide passive pull-up. The board also carried an on-board shunt for measuring current draw. Additionally there was circuitry for connecting to hall sensors and connect the motor phases to the ADC. In Figure 14 the important interconnects are drawn. Blue shaded blocks were not on the board. They were implemented on prototyping board and connected to the circuit using pin headers. The MCU (marked in yellow) was on the STK501 board and was also connected via pin headers.

This allowed time to be saved initially only adding the required circuits to realize the CBEMF and PWM system. Then the software was written to allow testing.

5.3.1 CBEMF circuit

The CBEMF circuit was designed to combine the three motor phases into a single signal and to offset this signal to v/2 for the analog comparator. It also filters out some of the switching-noise. The filter also reduces the motor's BEMF signal and drive supply down to within the MCU's operating range. One must take care to remember that the motors BEMF will about equal the drive voltage when motor is running at full power (see chapter 2.3.4). The filter depicted in Figure 15 is set up to load the signal equally in respect to the ground and 5V line to allow 0-crossing to be detected in the middle. Together with a reference network it will deliver a signal to the comparator in the MCU. In the figure, the phases A, B and C are called U, V and W. This was later changed.

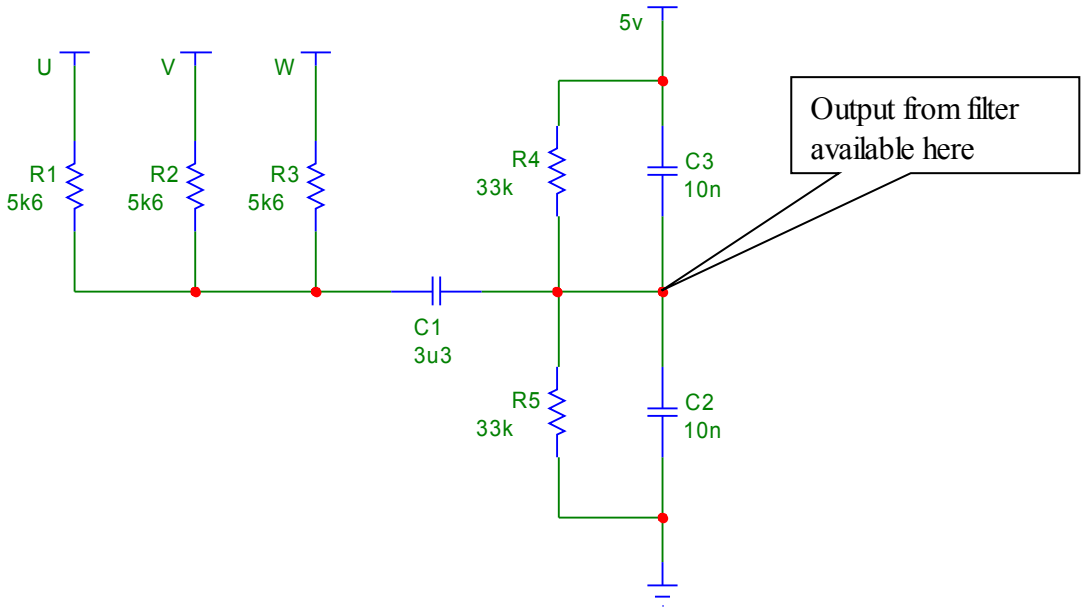


Figure 15 The CBEMF Circuit

The CBEMF circuit will introduce a delay and also limit the ability of the controller to detect the EMF 0-crossing event before the time to commutate. This only happens when the delay between 0-crossing and commutation is shorter than the delay caused in the filter and as such will only be relevant for very high rpm as already mentioned. Heavy filtering reduces noise in the CBEMF signal but increases the delay.

The circuit creates a simple band-pass filter that must be tuned to filter out as much as possible of the switching noise while at the same time introduce the smallest possible delay in the signal. Ideally a higher order filter should be used, but for simplicity and proof of concept this filter should do. C1 and R4/5 are part of the high-pass circuit while R1-3 and C2/3 are part of the low-pass circuit. With values depicted in [Figure 15], the filter would have a lower cutoff frequency of approx 3 Hz translating to a signal from 60 electrical rpm. (See chapter 2.3.3 on electrical rpm versus actual rpm.) The high threshold would be approximately 1421 Hz translating to about 28450 electrical rpm. This is using the classic R/C filter formula:

$$f_c = \frac{1}{2\pi RC} \qquad \textbf{(eq 4)}$$

This gives the cutoff-frequency where the output signal power is half the input. Also, remembering that R1-3 will be connected to signal, supply and ground respectively during use creating a voltage divider. The divider attenuates the signal by approximately 10 dB shifting the cutoff to -13dB and around 4.8 kHz. Values may have to be adjusted as performance of filter is tested. [Figure 16] shows the frequency response of the filter with two of the 3 inputs connected to ground/supply respectively.

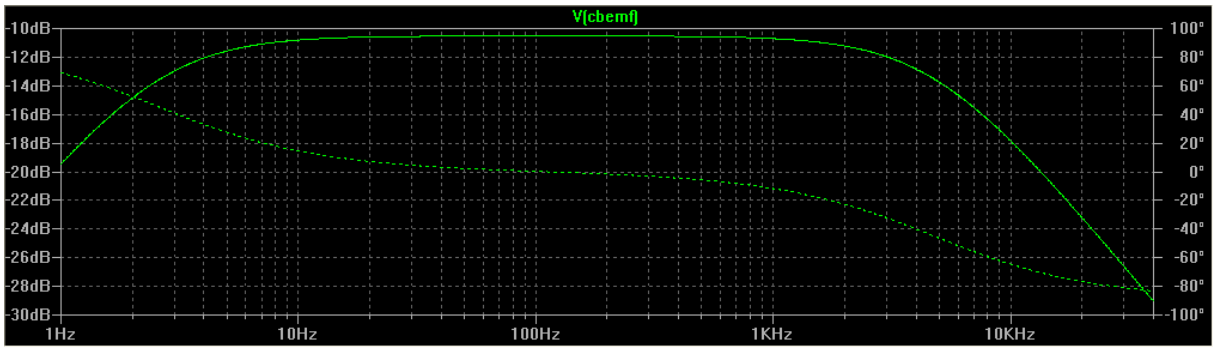


Figure 16 Frequency response of the CBEMF filter

5.3.2ADC–interface circuit

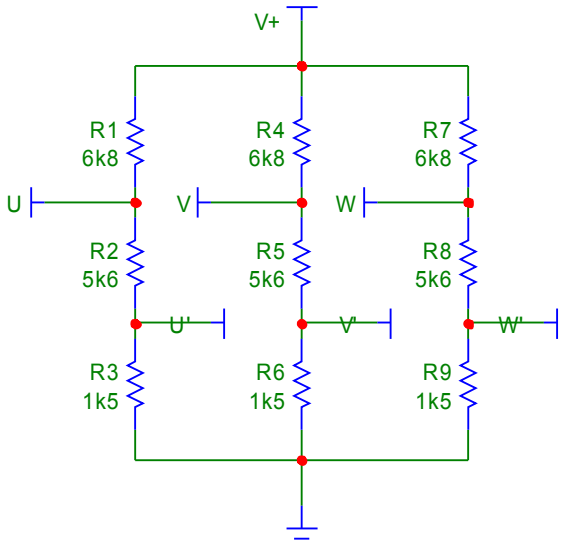


Figure 17 The ADC interface circuit

The ADC interface task is to reduce the motor voltages to within the operating range of the MCU. The built in ADC can be used with any external analog reference or an internal 2.56V reference. There is also the option of using the analog VCC supply.
For simplicity it is assumed that the MCUs internal protection is sufficient to handle any transients. The grid in Figure 17 is set up to cover input voltages up to around 23 volts if a 5v reference is being used. Values were chosen from E24 series. R1,4 and 7 serve to balance the load on the floating EMF signal so as to minimize offset towards ground.

5.3.3Testing

The analog comparator has a propagation delay of approximately 500ns. This was small enough to not warrant compensation in the algorithms.

The commutation interrupt needs 6 cycles after entering to save state of registers that will be used in the routine and similarly will need 6 cycles to restore these before returning to normal operation. Same timing applies for the comparator interrupt routine. The simpler hold-off routine on the other hand only requires 3 cycles to store and retrieve registers.
All three will also need 4 cycles to fetch the address of the interrupt routine and perform the jump to the actual routine. This covers 10 cycles for commutation and comparator, and 7 cycles for the hold-off. The return from an interrupt also takes 4 cycles giving same timing on return. If system clock is 8 MHz this means that commutation and comparator routines are delayed 1.25µs while hold-off is delayed 875ns. The hold-off delay being irrelevant as it only performs a small delay in turning on the CBEMF crossing detection and has no time-critical action to perform for the actual running of the motor.

None of the interrupt delays were large enough to warrant compensation at this stage. Even at 50.000 electrical rpm the delay would only approximate 1% of the delay between CBEMF crossing and the time for commutation.

Resource usage in the MCU was also low. The 3 main interrupts only take up 27+46+13 instructions for EMF, COMMUTATE and HOLD-OFF respectively. A total of 86 instructions. In addition there is an **init** routine that is called to set up timers and I/O at startup. This only takes 14 instructions of the possible 32k of available in program memory. There is also various other support functions as well as main loop. None of these take up

much space. With multiple startup alternatives the total code was still less than the 4k limit of IAR Kickstart. The actual code-size is however dependant on many factors and change according to optimizing settings. A lower priced controller with less program memory can easily be used to reduce cost of design.

No numbers on working memory usage was derived, however very few variables were used so the amount of memory used would be very low. The compiler also reuses memory that was only used for temporary calculations, lowering the required memory size even further.

5.3.3.1 Ramped start test

One of the first tests performed was the open-loop ramped startup. After adjusting the speed of the ramp-up and monitoring the CBEMF waveform on a deep capture oscilloscope, a novel feature of the signal came apparent. As shown in [Figure 18], the CBEMF changes as the motor and controller come in sync. At this point, the motor accelerates at same rate as the controller. The CBEMF will then be symmetrical, and match the predicted CBEMF as depicted in Figure 9

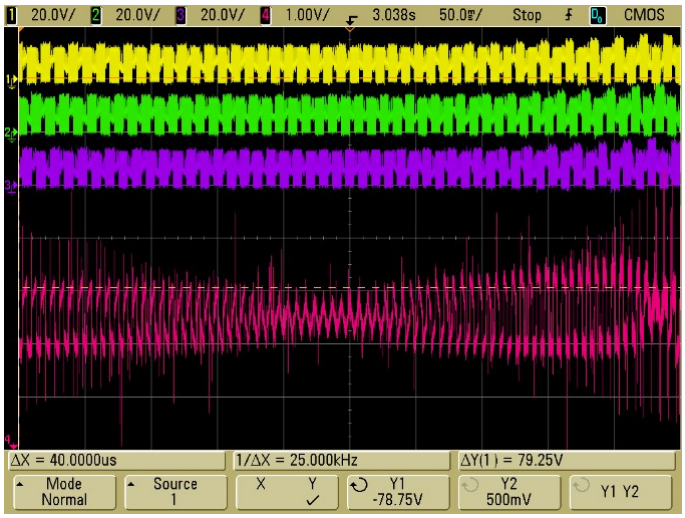


Figure 18 open-loop ramped startup reaching sync

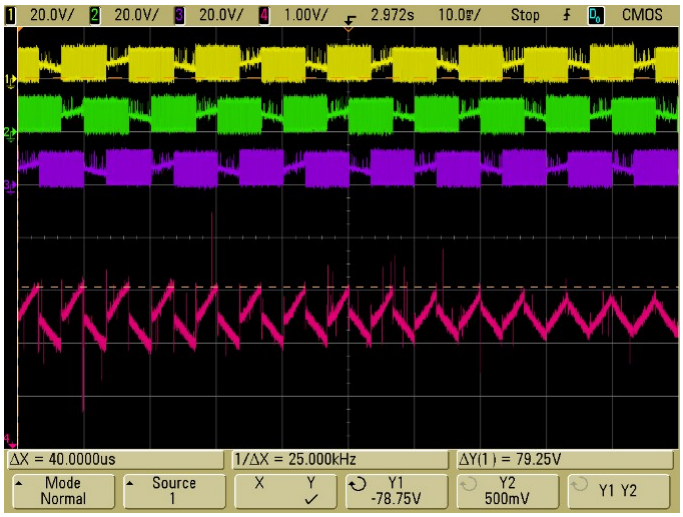


Figure 19 Closer view of left side of [Figure 18]

A closer view of the approaching left side of [Figure 18] is shown in [Figure 19]. One can clearly see how the CBEMF become a triangle-signal. This signal is composed of the rising and falling BEMF slopes interleaved.

If one looks at the CBEMF signal as a collection of slopes the next slope should start where the other finishes when the motor and controller are in sync. This is assuming the phases in the motor are symmetrical.
From the controllers view this can be looked at as a continuation of or breaking of a trend in the sampled data. If the first samples after a commutation deviate a lot in the opposite direction, compared to the current trend, the commutation happens too late. The CBEMF coming from the next phase was past its starting point when the commutation occurred, thus the CBEMF becomes discontinuous.

In [Figure 20] we see a closer view of the right side of [Figure 18]. As the motor is reaching its proper rpm its acceleration drops off while the controller keeps up the same rate. The result is the motor starts to fall behind the controller and we loose sync again.

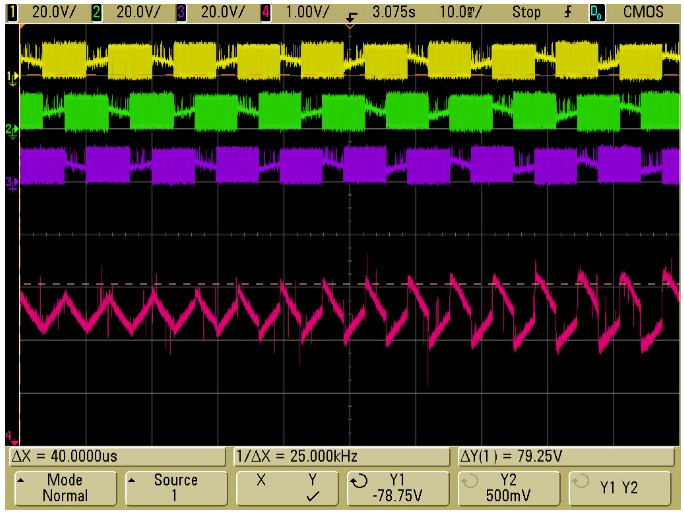


Figure 20 Closer view of the right side of [Figure 18]

This is very visible in the CBEMF signal. Looking at it from the controller's point of view the first samples after the commutation have exaggerated the trend and are now much higher or lower than the last samples before commutation. This is caused by the commutation happening too soon. The CBEMF coming from the new non-driven phase have not yet reached its peak.

This effect might be used to fine-tune the filter-delay and other parameters in the controller.

From the observers point of view it is important to note that the motors noise level changes as the controller and motor run in sync at close to optimal timing. The levels of noise become lower and the nature of the noise becomes smoother, with less sharp acoustics. Current consumption also drops as the motor and controller approach sync.

The actual starting using ramped start depended on a slow startup and enough power to be applied. If too little power was applied, the rotor would slip or skip and start to jitter or spin in opposite direction at a different rpm than expected. This comes from the motor deriving rotational force from a sub-harmonic, not the actual drive frequency. Initial power needed to start the motor also depends on the interaction between the motors permanent magnets and the ferrous stator. There will be some degree of binding at stable states in the rotation-cycle, referred to as cogging [3 Chapter 9.2]. This has to be overcome to start the shaft rotation.

This startup method also draws excessive current until motor and controller get in sync.
It did however start the motor and allowed focus to be put on the proof of concept test by implementing and testing the CBEMF system.

5.3.3.2ADC based start tests

The ADC based startup was implemented but did not prove to work better than the ramped startup. The algorithm attempted must be wrong as there are many examples that use this startup. The fast and successful implementations are protected however and insight into the algorithms is prevented. In [4] figure 5-10 a very nice figure of the startup sequence is shown. It appears to be an example only and not an actual acceleration as it does not have any PWM noise nor masking of the BEMF by the drive-signals as it is commutated. It does however show what an ideal startup looks like, accelerating the motor and acquiring the BEMF signal

in very few steps. With more work the ADC based method should still be able to handle large variations in motors size, response and load.

5.3.3.3CBEMF based start test

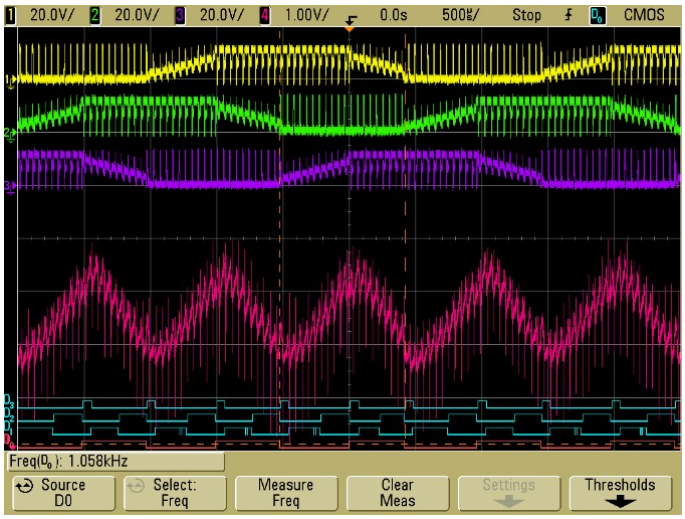
The CBEMF start tests proved not to be worth much investigation. The pulse required to start a cogging motor and make it freewheel for enough steps to acquire a BEMF signal lock is not practical when motor is under load. The pulse will also create a very high current spike. The method was abandoned in favor of more work on the CBEMF filter and reducing the noise in the CBEMF signal when running.

5.3.3.4Adaptive start test

Unfortunately the adaptive startup was never fully implemented or tested. Some attempts were made at using the offset information described in chapter 5.3.3.1 but were not successful. The problem may have been to identify when the signal contained valid information versus when it was noise.

5.3.3.5Comparator based running

The comparator-driven CBEMF based drive system was tested and found to be sensitive to PWM and other noise. At high rpm and closer to full PWM duty cycle it handled large changes in motor load without loosing sync with motor.



In [Figure 21] the three phases and CBEMF signal are shown while running at 60% on a 12v supply. The top 3 signals are the phase-voltages V_A, V_B, V_C and the bottom signal (red) is the CBEMF as fed to the comparator. The motor is a small-hard drive motor designed for high rpm. Commutation rate is approximately 2100 Hz making the electrical rpm around 21,000 rpm.

Figure 21 Drive waveforms at 60% duty

For a closer look at the drive waveform I include Figure 22. The field discharge after PWM turns outputs off is very apparent and lasts for approximately 7µs. The PWM noise on the CBEMF signal is also very apparent. Minimizing this noise should be a priority to make the system handle lower rpm using the CBEMF system. The noise has a peak-to-peak of 688mV.

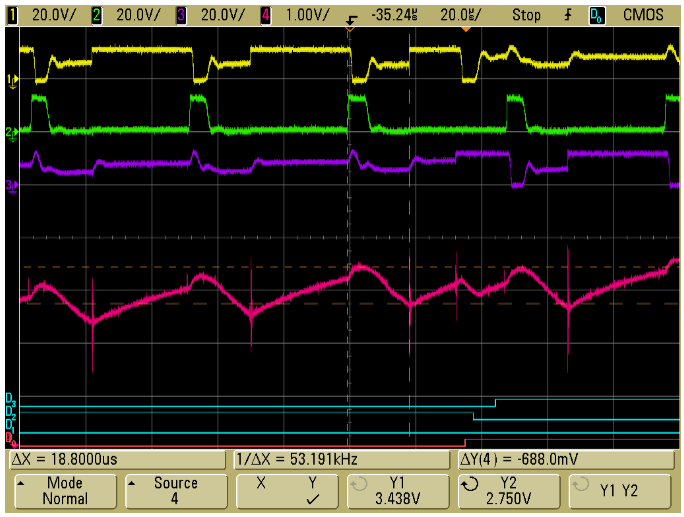


Figure 22 Close-up of drive-waveforms at 60% duty

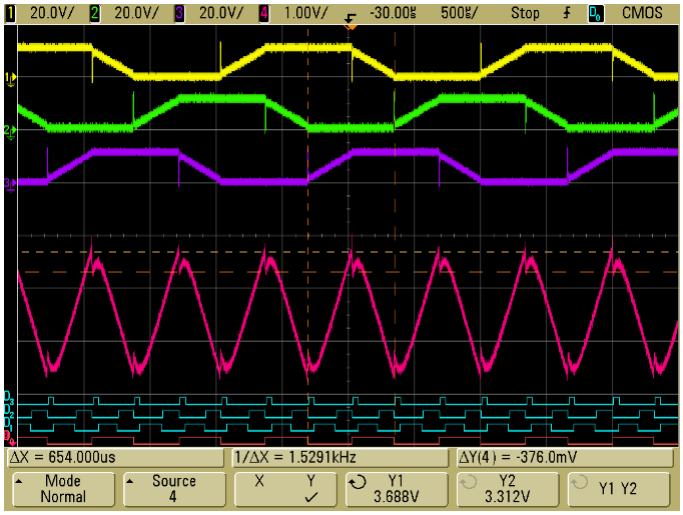


Figure 23 Hard drive motor at 30k rpm

To test the system at high speeds the same motor was run on 100% duty-cycle as shown in [Figure 23]. The motor then had an electrical rpm of around 30,300 meaning a commutation time of around 330µs. Studying the delay between commutations when motor load was not changed revealed a certain error (jitter). This jitter was very close to 7µs and translates to 1.26° phase variation at the current rpm. Considering it is normal to tune a motor for up to 15° either advanced or retarded, 1.26° is quite a small error.

The figure [Figure 23] is also free from PWM-noise, showing only a single spike where commutation occurs.

Looking close at the CBEMF we see a shift of 376mV upon commutation. With a full slope of approximately 2200mV this is around 17%. This can be caused by a combination of the spike during commutation and the filter constant in software. Since the error is opposite of direction of flank, the commutation could be happening too late. The filter constant in software (used to cancel out the delay in CBEMF filter), was 46µs and this might be a bit wrong, causing this permanent error/retarding in commutation. By using the ADC one might be able to adjust this factor while running by looking at the CBEMF right before, and right after a commutation. Increasing the filters delay factor from 46 to 56µs reduced the jump to 282mV. The jitter changed very little and read as 6.6µs. Increasing delay to 66µs reduced this jump even further, down to 240mV.

In light of the previously mentioned jitter, the motor was tested on almost 100% duty as shown in [Figure 24]. The jitter now became approx 56µs. This coincides very well with the PWM frequency of 20kHz which happens to have a cycle length of 50µs. Add this to the 6µs jitter in the system at 100% and we can be sure the PWM frequency is what disrupts the system at close to 100% duty. Looking at the CBEMF verifies that the noise related to switching can skew the trigger-point. It appears to delay it. Reducing switching noise as much as possible is thus necessary to use CBEMF accurately during PWM switching.

Figure 24 CBEMF noise at close to 100% duty.

Finally, the same motor was run with the driver-stage supplied with its maximum voltage of 20v. The motor now did 52,000 electrical rpm on 100% duty cycle. Measured jitter was

approx 6µs. To run faster would require a motor with a higher rpm pr volt constant (rpm/v) and/or a driver-stage capable of handling higher voltage.

An issue uncovered on this driver-board is the shift in signal levels due to changes in current. This shift is caused by the low-side shunt.

5.4 Second driver-stage

A new driver-board was designed to have better control with switching and have better matching on high/low-side switching timing. The reason for this was the noise generated on the CBEMF signal during PWM switching and an apparent difference in delay between high and low side turn-off. It was also designed to eliminate the current-dependant shift in the previous board. The board uses three integrated MOSFET-drivers. Each driver contains one high-side and one low-side FET driver capable of driving a half-bridge consisting of two N-type FETs. (See glossary for definition of FET and MOSFET.) Using N-type FETs allows better matching between high-side and low-side switching characteristics as well as improving the controller's current-handling without increasing the number of FETs needed. Scaling the driver for larger motors can be done by adding more FETs in parallel, or using larger FETs. The old board would not be easily modified to use a pure N-type driver-stage as it was designed with a transistor pull-down to handle the high side driving. This is in essence a single transistor and resistor pair doing the level conversion needed to pull the high-side P-FET gate signal down to GND (ground) with a 5V TTL signal from the MCU.

A block schematic of the new board is shown in Figure 25.

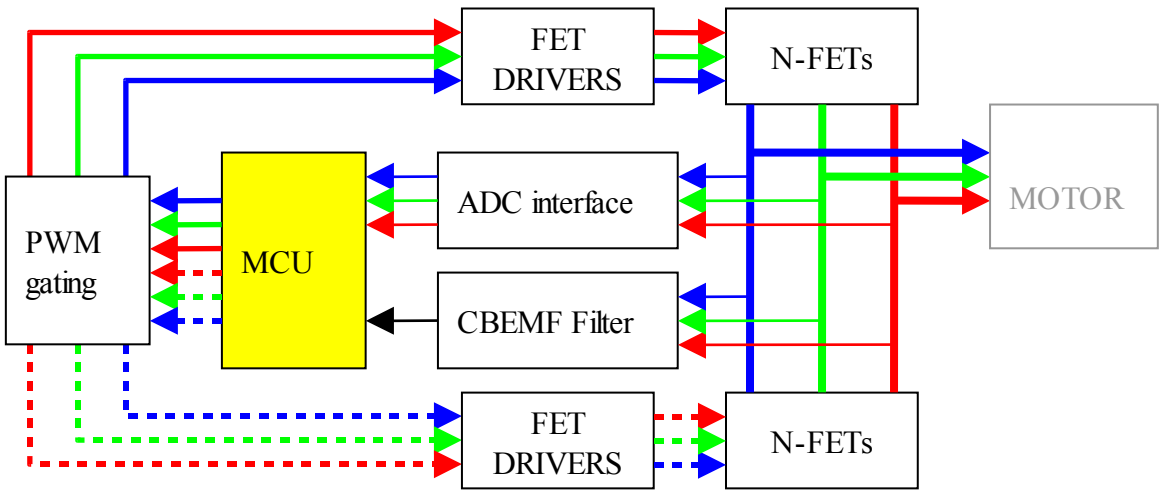


Figure 25 Second driver-stage block diagram

The block diagram in Figure 25 is arranged in upper and lower side drivers. This was done for readability only. In reality each of the 3 drivers handle one upper and one lower FET making up a single half-bridge.

The main reason for devising a new board was the theory that better matching on the symmetrical switching would reduce the switching noise on the CBEMF signal, improving accuracy to the point where detection would have higher resolution than the PWM signal.

5.4.1 Component selection

Due to good specifications and easy availability, the LM5101 was chosen to drive the N-type half bridges. It was more powerful than needed for the application, capable of drive-currents up to 3 ampere. The on/off delay-times are matched well and should work satisfactory in the application. Channel matching is specified to within 2-10ns. The benefits from such high current driving capability would become apparent if the design was scaled up to drive a higher current motor. Using multiple or larger FETs to allow for higher motor current, also requires more drive-current to keep switching losses down.

20N03HL DPAK N-channel MOSFETs from ON Semiconductor was available and had appropriate specifications and a low price. The board was designed to utilize these. Handling 20 ampere if adequately cooled at up to 30 volts combined with adequate switching characteristics and a low price made them a good baseline for possible later refinement. There are many MOSFETs with same footprint and pin assignment if there should be any reason to replace them.

5.4.2PWM gating

The sensing system requires the PWM switching in drive stage to be done on all 6 control-signals, and some of the 6 possible PWM outputs on the ATmega64 are shared with other functions that will be used in the design. The PWM signal was gated with the 6 drive signals externally using a hex inverter buffer with strobe. [Figure 26] depicts a functional diagram of this circuit. D1 to D6 is used for the 6 control-signals and PWM is applied to the INHIBIT pin. INHIBIT is active high, forcing output low whenever asserted. D1 through 6 are effectively inverted making the drive signals from the MCU active low and reset-safe with respect to internal pull-ups. This leaves the MCU free to use the hardware PWM generator and have it merge with the 6 regular drive channels. The PWM generator output OC0 is located on bit 4 of PORTB in the ATmega64. It is the PWM output of 8-bit Timer0 and is located on a pin without other shared functions.

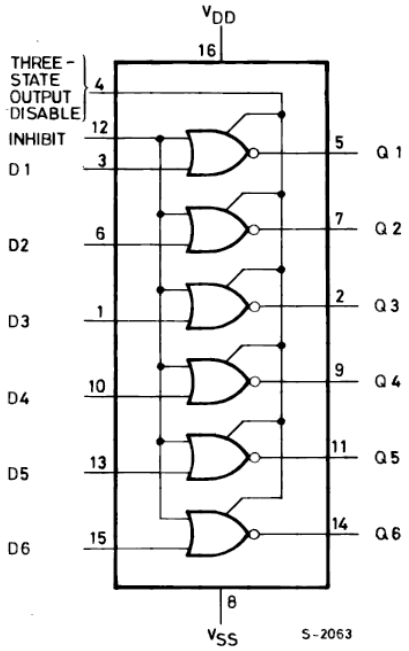


Figure 26 HEF4502B function

Optionally I could have used a software-based PWM triggering from an interrupt to update the drive signals but this would have required a lot of the MCU's processing-power to achieve. Not knowing how much time the drive algorithms would need, the choice fell on hardware gating.

There was a third option available that would require the use of all 6 PWM outputs and the enabling/disabling of the I/O ports output function to activate and deactivate the required signals. This option interferes with the built in soft pull-ups implemented in the MCU meaning these would have to be disabled. Further this alternative would block for using the internal analog comparator. In addition, this approach would take up both 16-bit timers. Blocking the comparator would mean using an external comparator, thus not saving any external components. Ideally, a MCU with a different pin-assignment and with a built-in gating-function would be ideal for this project. However, this would fall under the specialized controller category and not be a basic MCU.

5.4.3Layout

For layout the educational/freeware version of Eagle Layout Editor from CadSoft was used. EAGLE is an acronym for "Einfach Anzuwendender Grafischer Layout Editor" meaning "simple to use graphic layout editor".) EAGLE as an editor supports all features needed by this project. It has a quite extensive parts library and handles schematic input with forward and back annotation between layout and schematic. Adding new parts is quick while giving the necessary degree of control. There are also design rule checks (DRC) and electrical rule checks (ERC). For large designs it also has a configurable auto router which was not used for this design.

An important note is that EAGLE is available for both Windows and Linux.

The free (student/non profit-) version only handles 2 signal-layers and has a limitation on board size.

5.4.3.1Schematic

The schematic was done with basis on the datasheets of the LM5101 FET-drivers.

In [Figure 27] all sections of the schematic are shown. As is common in such schematics, nodes that are connected are named, but connections are not necessarily drawn to keep the schematic tidy. The version of the board shown in [Figure 27] is 1.45 while the last prototype made was 1.3 due to time limitations. The 1.3 prototype was modified to assimilate most of the changes shown in 1.45, mainly concerning decoupling capacitors for all integrated circuits. Initially a diode/resistor parallel circuit was used to link each driver's output to the FETs. This was removed as part of the efforts to prevent drivers from failing.
The exact cause of drivers failing is still not known, but the changes done to avoid it cover doubling the value of resistors RGxy where x={A,B,C} and y={H,L} to reduce maximum current supplied out of the drivers outputs. It also consisted of adding R1-R6 to limit the current supplied to input pins of drivers.

For larger representations of all sections from [Figure 27] refer to Chapter 9.

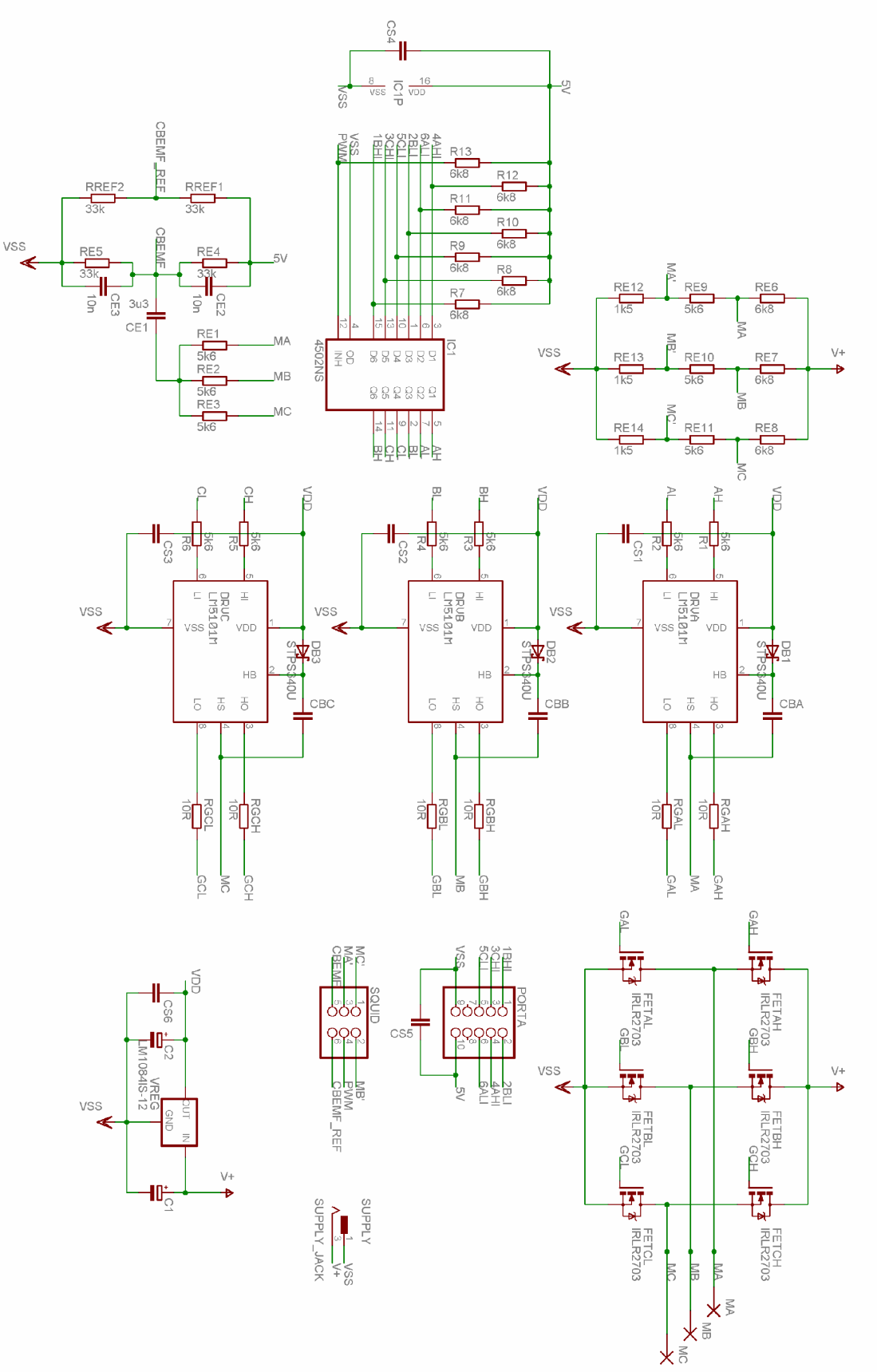


Figure 27 Overview of schematic layout for the driver-board.

5.4.3.2PCB

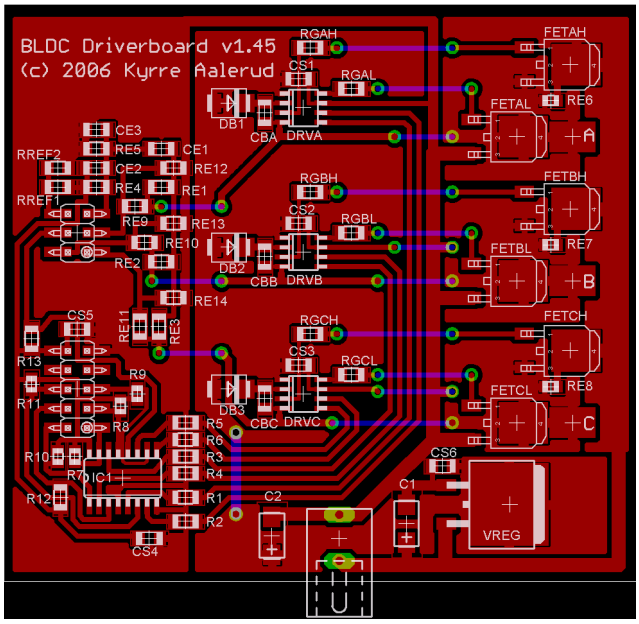


Figure 28 PCB layout for second driver-board

To avoid lengthy production delays or high expenses, the layout was done 2-sided. Only jumps were done on backside. This allowed prototypes to be etched and soldered up the same day.
A Photolithographic process was used to transfer the computer assisted design (CAD) -layout to the printed circuit board (PCB). Due to issues with the LM5101 many revisions were made of the circuit with corresponding changes to the layout. The final revision is shown in [Figure 28]. Ideally the design should be manufactured on a multilayer board with proper ground plane. This would ease later development. For a larger view of the PCB refer to chapter 10.

EAGLE project files are available on request. They are not suitable for inclusion in documentation.

5.4.4Testing

Due to problems with the chosen FET drivers the design and construction of this board took up more time than it should have. Drivers kept failing, shorting high-side output to VCC. Analysis did not give any definite answers as to why, but grounding issues were uncovered. Improving ground reference on multiple points, especially between FETs and drivers as well as reducing switching current to half the original, turned out to alleviate the problem.

If anyone are to base a design on this it is important to express that the layout of this board was for prototype purpose only. The design should be redone with multilayer board and proper ground-plane.

Further testing with non BEMF based driving (manually regulated) of a motor shows excessive noise still in system. Inspection of the CBEMF filter revealed it changes gain according to the PWM. This seems to cause extra noise.

6 Conclusions

Driving a motor with a low-cost MCU using CBEMF turned out to be possible and even practical. The accuracy was good and it has the potential to be even better. One would need a very fast ADC to be able to run at similar speeds without sacrificing some ability to handle dynamical changes in load.

Time was lost in the second driver-board which could have been used to analyze and improve startup algorithms. However, the main interest was on the CBEMF driving and the aim of the new board was to improve this.

Just designing the hardware for the project turned out to be harder than assumed. Switching currents at high rates causes noise and dealing with or avoiding this noise turned out to be hard. It was however very informative and interesting.

6.1 Benefits and downsides of CBEMF

The first benefit is being able to utilize a low-cost MCU.
Secondly the ability to use said low-cost MCU to run a motor at a high commutation rate with high degree of commutation-accuracy must be counted as a big benefit. With a better filter or method of composing the CBEMF signal the method has possibilities.

The most obvious downside of CBEMF is the lack of control. The controller does not know if the feedback triggering the commutation is coming from the correct phase at any given time. The system relies on the controller and motor staying in sync. Since the CBEMF control demonstrated abilities to handle large variations in load, mainly due to the per-commutation adjustments and low delay, it may keep sync in situations where ADC based driving may loose it. The lack of phase discrimination may or may not pose a problem, depending on the application. For applications requiring with absolute certainty that the motor never travels the wrong way, like hard drive use, CBEMF may not be useful. A viable area of use may be in controlling micro motors for model planes and other fan-applications. Basically high rpm motors sensitive to timing changes that do not have such special demands as hard drive motors do.

6.2 Possible improvements

The most apparent point of improvement seems to be filtering of the CBEMF waveform to remove the switching noise caused by the PWM control. Reducing the amount of noise generated would also help in this sense. Alternative methods might be needed to create the CBEMF signal, possibly decoupling it with the help of the PWM signal. This would be an alternative to digital filters which can be expensive if done in hardware, and would introduce delays if done in software.

Another point of improvement would be startup algorithms. Determining the position of the rotor before starting seems to be a key in starting a motor reliably. Detecting and reacting to the motors rate of acceleration would then allow for the fastest possible spin-up with available power.

7 Suggested further research

If further studies are to be held in this area it is recommended to supply some sort of modular hardware allowing filters and other possible feedback systems to be added by student, but providing the basic half-bridge control. This would allow more time to be spent on algorithms and feedback systems.

As CBEMF-drive is possible, a better method of obtaining the CBEMF signal would be beneficial. Ideally a method that is immune to PWM noise and general transients should be devised.

Another area that might merit further study is startup algorithms. The possibility of using evolution or learning algorithms may allow controllers to handle different motors without calibrating startup algorithms.

8 A - Glossary

ADC	-	Analog to Digital Converter Converts an analog voltage to a digital representation.
ALU	-	Arithmetic Logic Unit Unit to do arithmetical and logical calculations.
BEMF	-	Back Electromotive Force Induced force in a rotating motor opposing the applied field.
BLDC	-	Brushless Direct Current Used to refer to brushless direct current motors.
CAD	-	Computer Assisted Design Using computer to aid in designing. Used in many areas of engineering.
CBEMF	-	Compound back Electromotive Force A compound feedback signal from a brushless motor. Consists of the BEMF from all phases interleaved.
Commutation	-	The means of driving a brushless motor via different steps where each step supplies motor with a different part of the drive-waveform.
Commutator	-	In brushed motors the commutator is the interface-plate for the brushes. It distributes the connections to the windings to provide correct sequencing for the generated field allowing the motor to run. Usually a segmented plate or disc around the rotor shaft.
duty cycle	-	The ratio between on and off time in a repetitive signal.
FET	-	Field Effect Transistor A type of transistor that uses an electric field to control the conductivity in a semiconductor material.
jitter	-	Irregularity in timing or amplitude of a signal Define a range where a certain event occurs apparently at random.
MCU	-	Micro Controller Unit A chip containing all parts needed to run a small computational core. This includes working memory, program memory, long term storage memory and the ALU.
MOSFET	-	Metal-Oxide-Semiconductor Field Effect Transistor A type of FET that uses a metal-oxide in the design. Typically SiO_2.
PCB	-	Printed Circuit Board Sandwich of conductors and isolators with a number of layers.

PWM - Pulse Width Modulation
A means of power control where the drive signal is modulated by a fixed frequency signal with varying duty cycle according to power needed.

prescaler - Unit to reduce clock-rate supplied by either a set or selected factor to produce a slower clock.

9 B – Schematic layouts

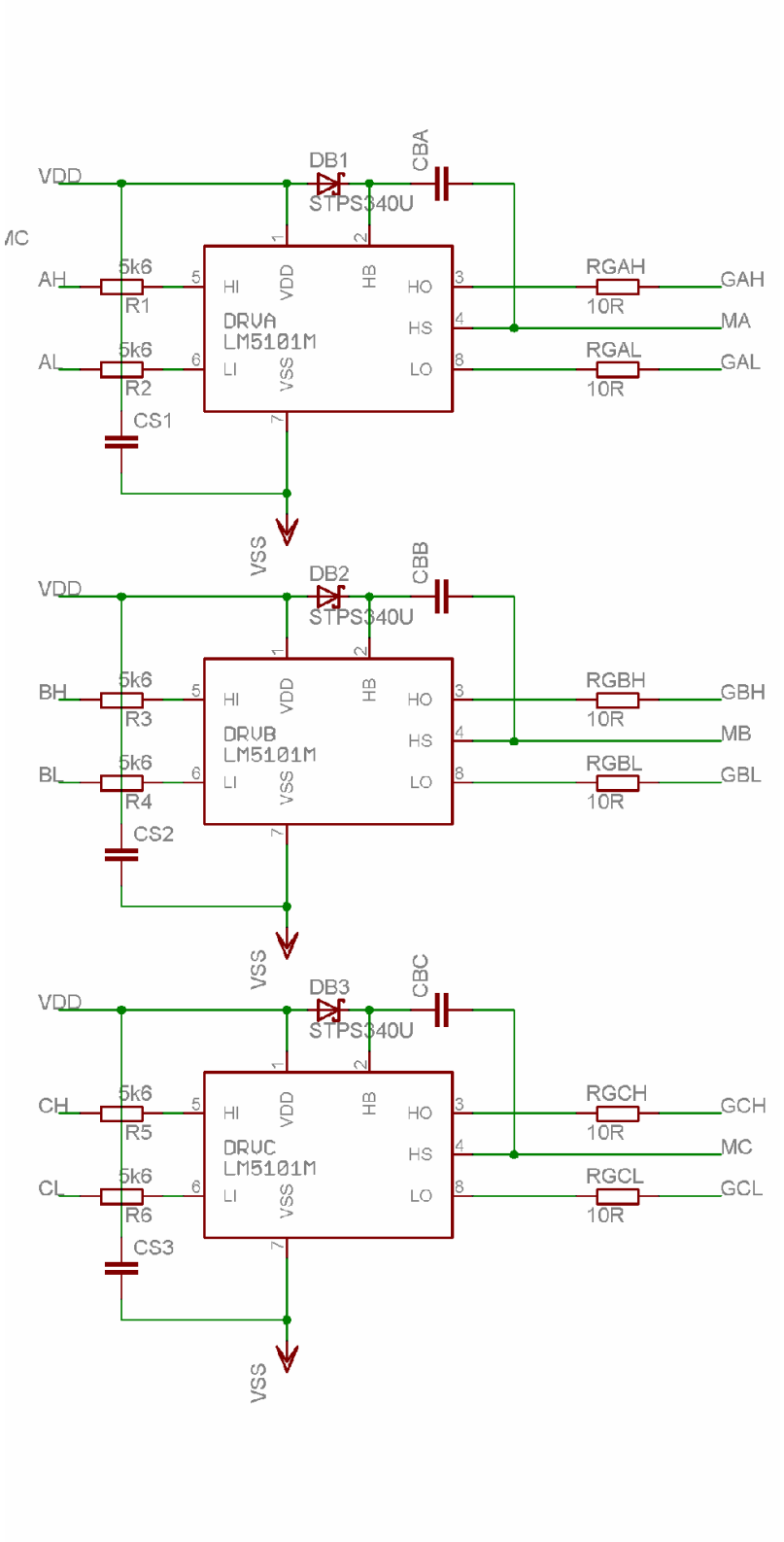


Figure 29 FET drivers section schematic layout

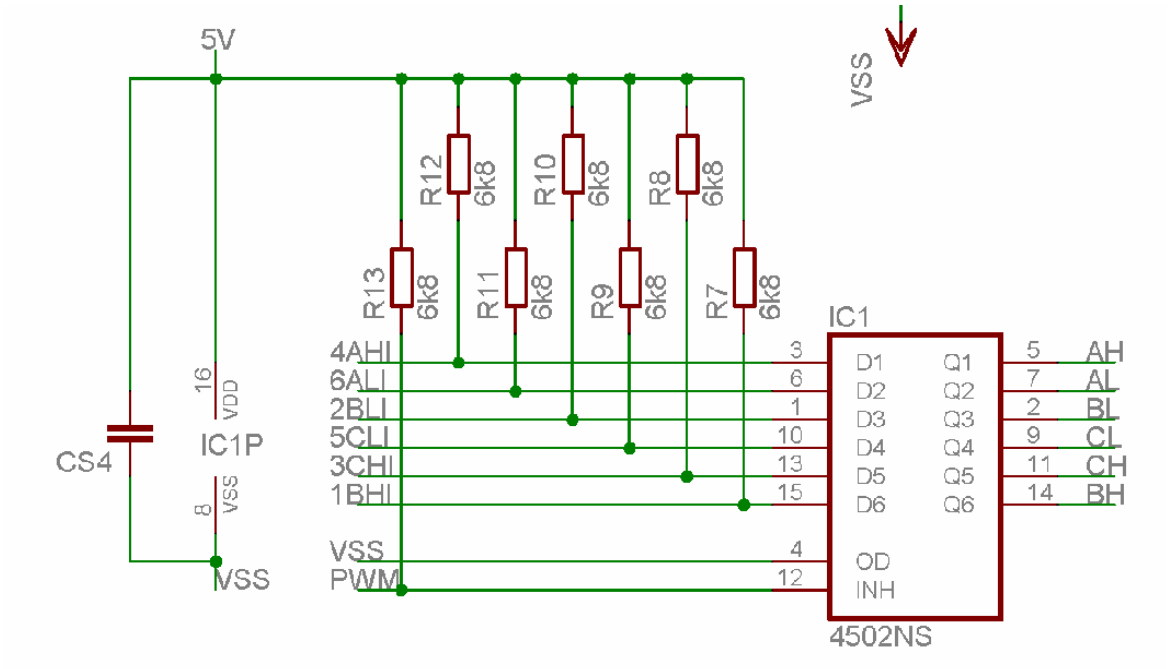


Figure 30 PWM gating circuit section schematic layout

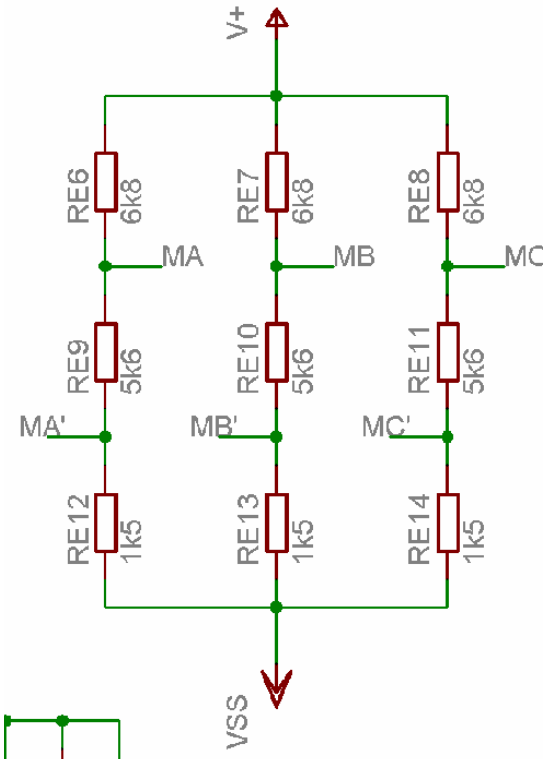


Figure 31 ADC interface schematic layout

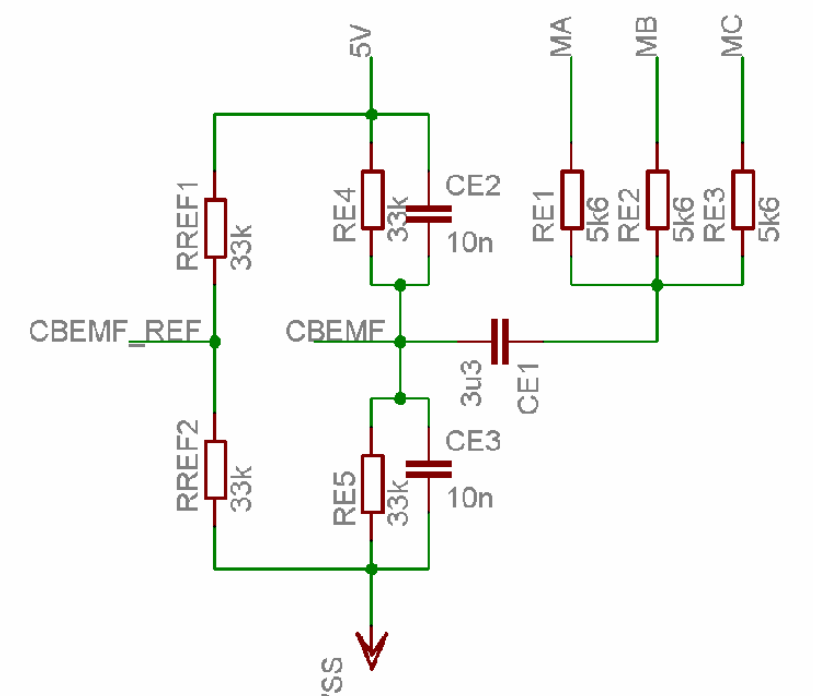


Figure 32 CBEMF circuit schematic layout

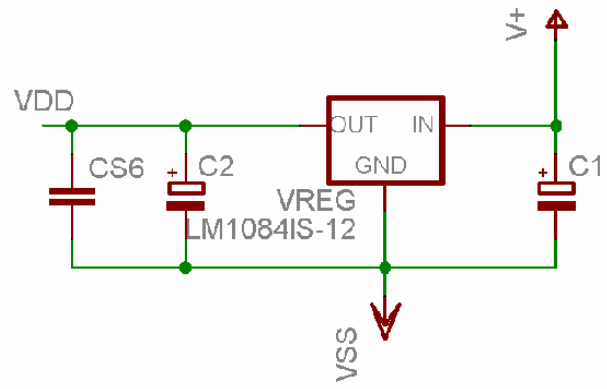


Figure 33 Voltage regulator and relevant components

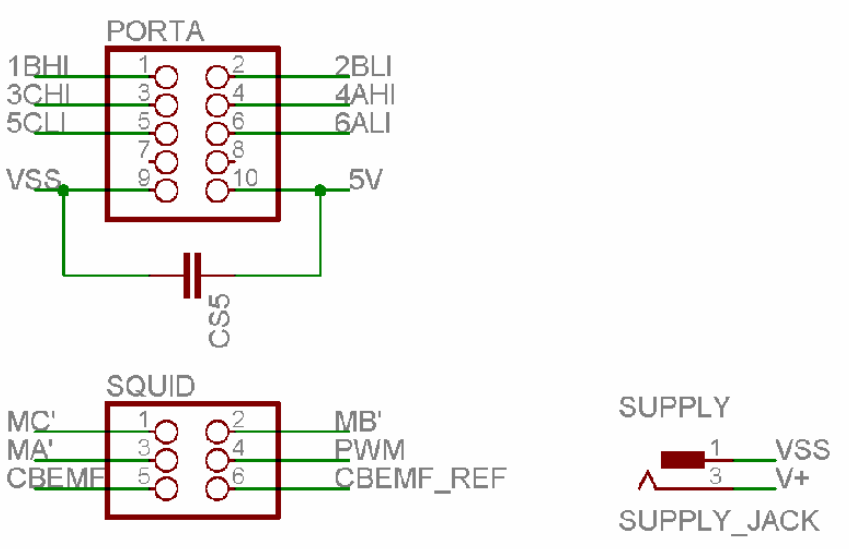


Figure 34 Connectors schematic layout

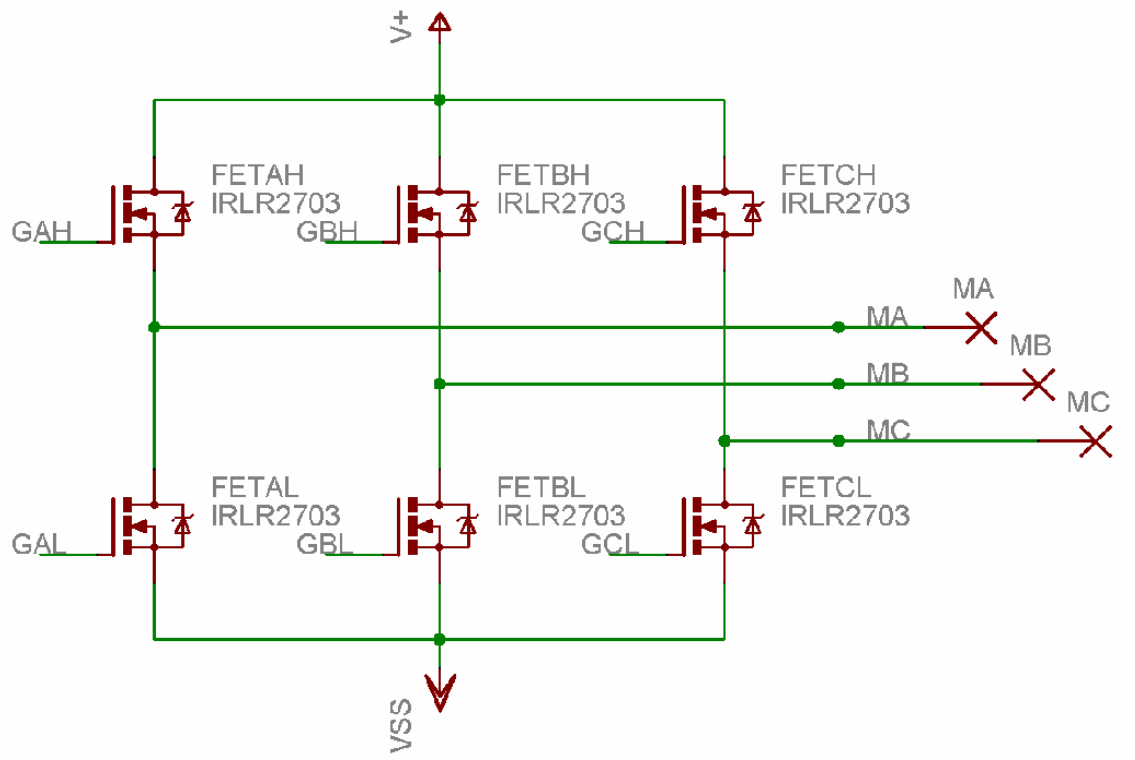


Figure 35 FET section schematic layout

10C – PCB layout

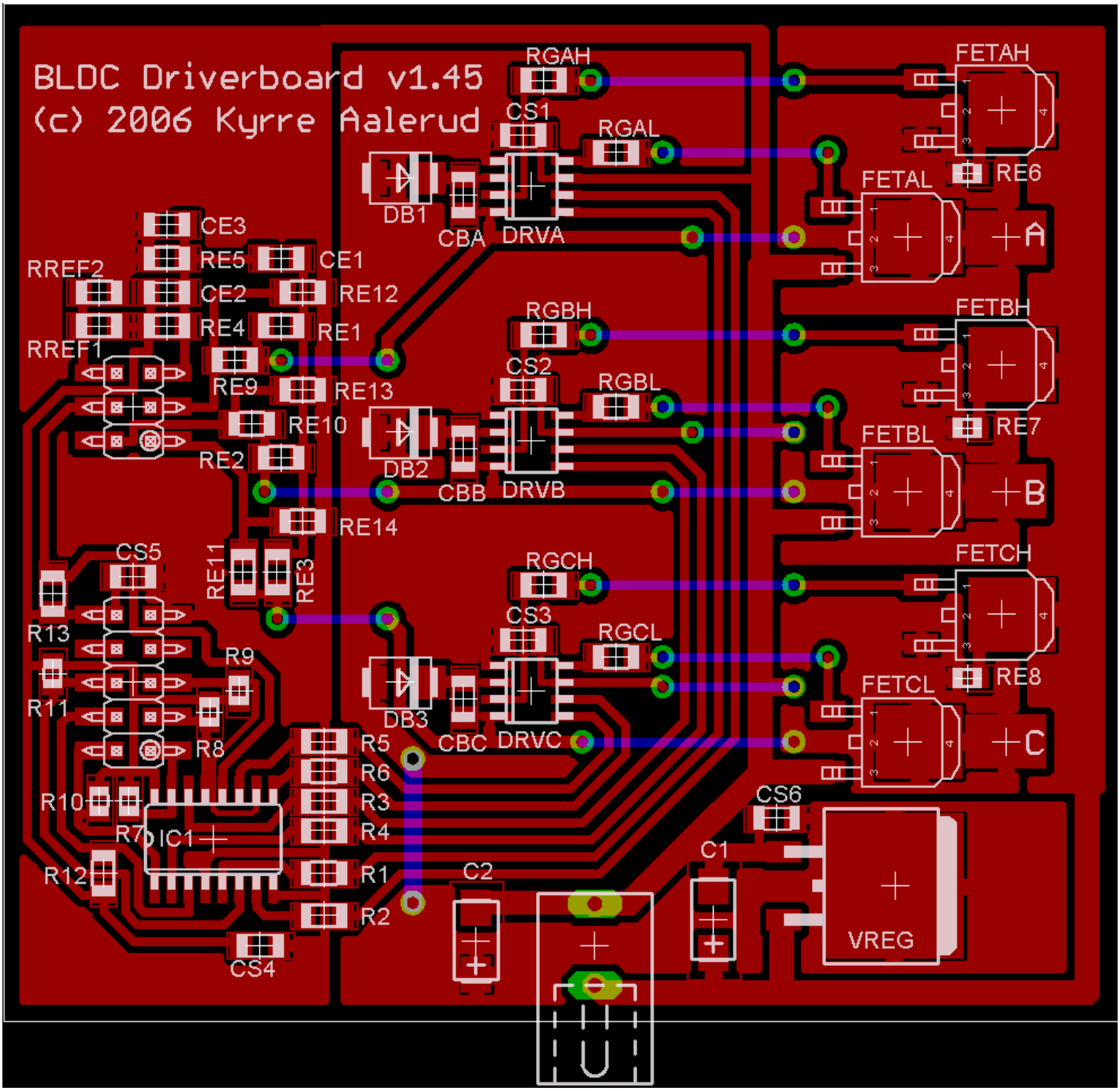


Figure 36 PCB version 1.45 layout

11 Bibliography

1 **Thesis: Direct Back EMF Detection Method for Sensorless Brushless DC (BLDC) Motor Drives**
http://scholar.lib.vt.edu/theses/available/etd-09152003-171904/unrestricted/T.pdf
Jianwen Shao, etd-09152003-171904 (Reference number in digital library.)
Thesis for Master of Science, September 2003, Blacksburg, Virginia
Virginia Polytechnic Institute

2 **Appnote: Brushless DC (BLDC) Motor Fundamentals**
http://ww1.microchip.com/downloads/en/AppNotes/00885a.pdf
Application note AN885
by Padmaraja Yedamale
Microchip Technology Inc.

3 **Book: Permanent Magnet Motor Design (Briefs)**
http://www.eece.maine.edu/motor/
ISBN: 1-881855-15-5
Dr. Duane Hanselman
June 2006

4 **Appnote: 3-phase BLDC Motor Control with Sensorless Back-EMF ADC Zero Crossing Detection using DSP 56F80x**
http://www.freescale.com/files/product/doc/AN1913.pdf
Application note AN1913, rev 3, 11/2005
Libor Prokop, Leos Chalupa
Freescale Semiconductor, Inc

5 **Overview: Permanent Magnet Synchronous Motor**
http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=02nQXGrrlPZL8l
Overview by freescale.

6 **Wikipedia on "Electric Motors" #AC Motors**
http://en.wikipedia.org/wiki/Electric_motor#AC_motors
Online resource.

7 **Appnote: Using the PIC18F2431 for Sensorless BLDC Motor Control**
http://ww1.microchip.com/downloads/en/AppNotes/00970A.pdf
Application note AN970, 2005.
by Padmaraja Yedamale
Microchip Technology Inc.

8 **The HEF5202B Datasheet**
ST.

9 **Datasheet: ATmega64(L)**
http://www.atmel.com/dyn/resources/prod_documents/doc2490.pdf
Datasheet, 4/2006.
Atmel Corporation

10 **Online resource: MMT Brushless DC motors**
http://www.movingmagnet.com/Brushless.htm
MMT is a brushless motor manufacturer.

11 **Online resource: Model Motors – AXI 4120/14**
http://www.modelmotors.cz
Model Motors is a brushless motor manufacturer.

12Figures

13Equations

eq 1 Equation 3-6 from [4].