NTNU – Trondheim
Norwegian University of
Science and Technology

Design of a Switch-Mode Power Electronic Converter for Teaching Laboratory

Ragnhild Solheim

Master of Energy and Environmental Engineering
Submission date: June 2012
Supervisor: Lars Einar Norum, ELKRAFT

Norwegian University of Science and Technology
Department of Electric Power Engineering

Problem Description

This work should focus on creating a switch-mode power electronic converter for use in practical teaching within the disciplines of power electronics, electric drive systems and digital control. To broaden the expediency, it should be made suitable for a wide range of applications. As it will be used as a teaching material, a safe user interface must be implemented. A digital controller should be used for the converter control, including the generation of PWM signals and an analog-to-digital conversion of suitable measurements. A case study showing how the converter can be used in a DC motor drive should be presented.

Assignment given February 1st 2012

Professor Lars E. Norum

PREFACE

This report shows the work of the last semester of my master degree in Energy and Environmental engineering at Norwegian University of Science and Technology (NTNU). It has been a very interesting project, with a wide range of challenging tasks, most of them unfamiliar for me when starting. Therefore I have learned a lot; how to design a printed circuit board, the theory behind control of motor drives and how to implement this in real-time programming and how to measure motor speed by a microcontroller, to mention the most specific. I have also gained a lot more experience with lab work and real-time programming in general.

Finishing this work, there are a number of people I would like to address my thanks to, who I could never have done this without. First of all, I would like to thank my supervisor, Professor Lars E. Norum, who defined the problem of my thesis and who seems to know exactly what will interest me, before I know it myself. I also want to say how much I appreciate him being optimistic according to my work. Then I will thank my co-supervisors, Fritz Schimpf and Frederick Ishengoma for always having time for my questions about respectively PCB design and the microcontroller coding. I would also like to thank Bård Almås and Vladimir Klubicka at the institute's service lab, for helping me with all the practical work and for lending me a desk in their very pleasant working environment.

All the time spent working on this thesis would not have been the same without my fellow students. Thank you for always being relaxed and helping me clear my mind during the breaks. I will also like to express my gratitude to you, Ragnhild, Maren, Kristin and Granpa for proofreading my report.

Finally, I would like to thank my parents for all their support.

Ragnhild Solheim

Trondheim, June 20th 2012

SUMMARY

A power electronic switch-mode three-leg converter is a flexible converter and hence very useful for practical teaching of several disciplines within electric power engineering. It can be used as a half-bridge, full-bridge and three-phase converter, to mention a few, and enables the user to study many different power electronic topologies. The converter, controlled by a microcontroller, can also be used for teaching digital control of power electronics. Its output can be varied in frequency, magnitude and waveforms, and can also be measured by the microcontroller. The flexibility of the converter makes it possible to utilize it in drive circuits for a wide range of loads and can therefore also be used for teaching electric drive systems.

This thesis shows a solution of how to design the switch-mode three-leg power electronic converter. The converter is designed and implemented on a printed circuit board (PCB) together with other necessary components. To meet the safety requirements of the problem description, the power rating is low, 12 A * 50 V, and the power circuit is isolated from the microcontroller on the PCB. The microcontroller chosen is the Texas Instruments Piccolo™ ControlCARD and pulse-width modulation and analog-to-digital conversion is implemented with real-time programming.

This system developed is verified, except for the MOSFET drivers and measurement circuits. As time was limited, the laboratory work had to be ended in favor of writing the report. Unfortunately, this made it impossible to test the full system setup. A full description of the changes to be implemented for the whole system to be functioning and further tested is provided.

A system for using the converter designed in a DC motor drive, by utilizing two of the bridge-legs as a full-bridge converter, is studied. The programming code is tailored for the specific purpose and speed measurements and control algorithms were added. Due to the converter not functioning, the testing of the DC motor drive could not be performed. However, full planning and controller implementation was done.

SAMMENDRAG

En ”switch-mode” kraftelektronikkomformer med tre brolegger er en fleksibel omformer som er svært nyttig for bruk i praktisk læringssammenheng, innenfor flere fagområder innen elkraftteknikken. Den kan bli brukt som en halvbro, fullbro eller trefase omformer, for å nevne noen, noe som gjør det mulig å studere mange kraftelektroniske topologier. Omformeren, som er kontrollert av en mikrokontroller, kan også bli brukt til å lære digitalkontroll av kraftelektronikk. Utgangsspenningen kan varieres i frekvens, størrelse og kurveform, i tillegg til at den også kan måles av mikrokontrolleren. Det at omformeren er så fleksibel gjør at den kan brukes i driverkretser for et vidt spekter av forskjellige laster, og kan dermed også brukes til praktisk læring av elektriske driversystemer.

Denne masteroppgaven viser en utforming av en ”switch-mode” kraftelektronikkomformer med tre brolegger. Omformeren er laget på et printet kretskort (PCB) sammen med andre nødvendig komponenter. For å imøtekomme sikkerhetskravene i oppgaveteksten er det valgt en lav merkeeffekt på 12 A*50 V, i tillegg til at kraftkretsen er isolert fra kontrollkretsen på kortet. Mikrokontrolleren brukt er Texas Instruments Piccolo™ ControlCARD og pulsbreddemodulasjon og analog-til-digital konvertering er implementert ved bruk av sanntidsprogrammering.

Hele systemet er verifisert, bortsett fra MOSFET driverne og målekretsene. Ettersom tiden var begrenset, måtte laboratoriearbeidet avsluttes til fordel for rapportskrivingen. Dessverre førte dette til at det ikke var mulig å sjekke systemet som en helhet. En fullstendig beskrivelse av endringer som må utføres for å få systemet til å virke og videre testet, er imidlertid tilgjengelig.

Et system for å bruke omformeren i en DC motordriver, ved å nyttiggjøre to av broleggene som en fullbro-omformer, er studert. Programmeringskoden er skreddersydd til dette formålet og hastighetsmåling og regulatoralgoritmer er inkludert. Hele systemet er planlagt og regulatoren implementert, men da

omformeren på nåværende tidspunkt ikke virker, har ikke systemet blitt verifisert som helhet.

TABLE OF CONTENT

LIST OF TABLES

List of Figures

1 Introduction

1.1 Creating a Switch-Mode Three-Leg Power electronic Converter

To answer the problem description and make a flexible power electronic switch-mode converter suitable for general laboratory usage, the three-leg converter topology is chosen. According to Mohan et. al. (2003), it can operate as a half-bridge, full-bridge or three-phase converter, in four quadrants, leading to a wide range of usage: DC-DC converter, DC-AC single-phase or three-phase inverter, in a DC or AC motor drive, to mention some. It can hence be used to teach a broad coverage of the disciplines of power electronics and electric drive systems.

The converter is being controlled by a microcontroller, which can vary the output voltage in magnitude, frequency and waveform. A real-time digital control algorithm will be used to control the microcontroller. This will also make the system suitable for practical teaching of digital control theory.

The circuit diagram of the converter with three legs is shown below. Each leg includes two switches and its output is the point between the two, marked A, B or C. The switches are controlled by a pulse-width-modulation (PWM) signal. The output is determined by how the switches in a leg are controlled, how many of the three legs that are used and how they are controlled compared to each other. The input voltage is usually DC.

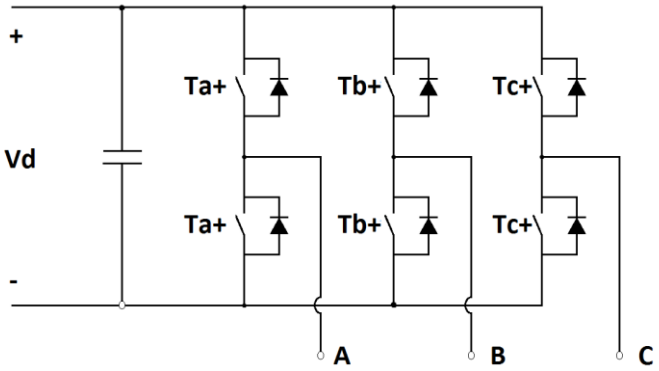


FIGURE 1.1: CONVERTER WITH THREE BRIDGE-LEGS (MOHAN, ET AL., 2003)

The converter topology is implemented on a printed circuit board (PCB) together with other necessary equipment to run the converter. The planning of this is done in CadSoft Eagle™ PCB Design Software. The microcontroller used is the Texas Instruments Piccolo™ ControlSTICK. It controls the converter by generating the switches' PWM signal. The control algorithm is implemented with real-time programming in Code Composer Studio™.

To see how the converter behaves for different switching schemes, the currents and voltages will be measured, and analog-conversion circuits will be implemented to be able to convert the measurements to digital by the microcontroller.

To overcome the safety requirements, the control circuit including the microcontroller will be isolated from the power circuit and the power ratings is limited to a maximum of 50 V and 12 A.

As required, the second part of this report describes and implements how the converter can be utilized as a full-bridge converter in a DC motor drive. A feedback speed and torque controller is implemented within the microcontroller's code. It uses the output current and the speed which is measured by a quadrature encoder. A simple block diagram of the system is presented under.

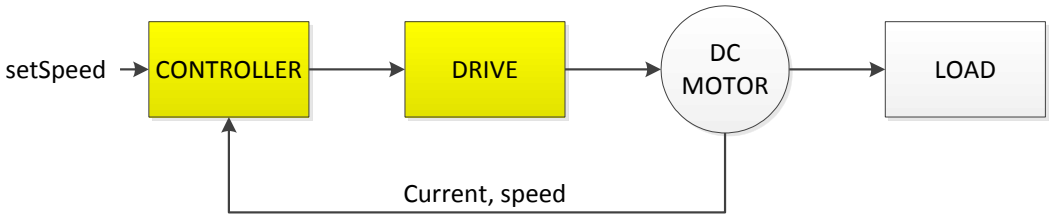


FIGURE 1.2: BLOCK DIAGRAM OF MOTOR CONTROL

1.2 SCOPE AND LIMITATIONS

The two main areas which have been focused on during this study are marked yellow in the figure above. These are the converter, both hardware and software, and how to use it in a DC motor drive.

As a first draft, it is focused on getting the converter to work and to use it with a simple control algorithm to control a DC motor. Therefore an optimizing of the design is not considered. This applies to the schematics and board layout, choice of components and programming. Apart from that, the safety aspect mentioned in the problem description is considered by isolating the control circuit from the power circuit and having a low power rating.

While writing this report, it has been tried to describe what is done and the thoughts behind it in a way which makes it as easy as possible to understand for the student who is continuing the work. However, it is assumed the reader has some knowledge about real-time programming, Code Composer Studio™ and CadSoft Eagle™ PCB Design Software.

1.3 ORGANIZATION OF THE REPORT

This first chapter gives an overview of the solution of the problem presented. Further, it outlines the scope and limitations of the work.

The second chapter describes the converter design. First, it presents the theory about one, two and three-leg converters followed by the description

of the planning and implementation of the converter on a printed circuit board. Then, a programming code doing the general tasks of the microcontroller is explained.

The third chapter describes how the converter can be used as a DC motor drive. First, some theory about the DC motor, operation and control theory are given, and then the system as a whole is described. Finally, the modifying of the general code together with the additional code for motor control is explained.

The mounting and verification of the, hardware, software and the system as a whole is given in chapter 4.

Chapter 5 provides the improvements for later versions.

The conclusion and recommendations for further work is outlined in chapter 6.

2 CREATION OF THE CONVERTER

2.1 CONVERTER THEORY

This section is based on the theory about switch-mode power electronic converters given in Mohan, et al. (2003) and Mohan (2003).

2.1.1 ONE BRIDGE-LEG

To explain the theory behind the converter, the one-leg operation is first considered, since the basic operation theory of each leg is the same regardless of how many legs connected. This topology is also called a half-bridge converter, and is shown in Figure 2.1.

The voltage applied across the bridge-leg is V_d. The two switches are switched asynchronously; one is on when the other is off. The output voltage is measured between point A and N (=ground) is hence set to either V_d or ground, depending on which switch is on. Its average value depends on how long each of the switches is turned on compared to the other one.

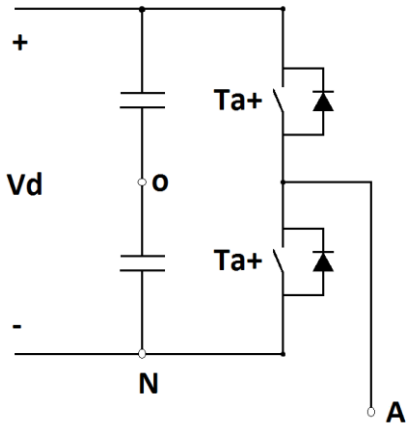


FIGURE 2.1: HALF-BRIDGE CONVERTER

To express the output voltage mathematically, a few terms must first be presented. The switching period T_s is defined as the time it takes for both switches to be turned on and off once. The upper switch is on during $T_{on}$and off during T_{off}, and the opposite is true for the lower switch. Thus, $T_s = T_{on} + T_{off}$. The switching frequency f_s is one over the switching period. A duty ratio $D$is defined as the on-time over the switching period. The duty ratio for the upper switch and the bridge-leg's output voltage $V_{AN}$is given by $D_+ = T_{on}/T_s$, and for the lower switch as $D_- = T_{off}/T_s = 1 - D_+$. By these terms, the average output voltage V_{AN} can be written as

$$V_{AN} = \frac{V_d T_{on} + 0 \cdot T_{off}}{T_s} = \frac{V_d T_{on}}{T_s} = V_d D \qquad 2.1$$

(Mohan, et al., 2003)

P W M

The switches are controlled by a pulse-width-modulation (PWM) signal, which is a square wave pulse alternating between high and zero. It turns the switch *on* during the high pulse and *off* when it is zero. Hence, the frequency of the PWM signal determines the switching frequency. The PWM signals for the two switches are opposite, to make the switching asynchronous.

The two PWM signals for the two switches in a bridge-leg are both made of one triangular waveform v_{tri} and one control voltage $v_{control}$. The triangular waveform is alternating between $+\hat{V}_{tri}$ and $-\hat{V}_{tri}$. The upper switch's PWM signal is high when the control voltage is higher than the triangular and low otherwise. The opposite is true for the lower switch's signal. The waveforms are shown in Figure 2.2, where q_A is the PWM signal of the upper switch.

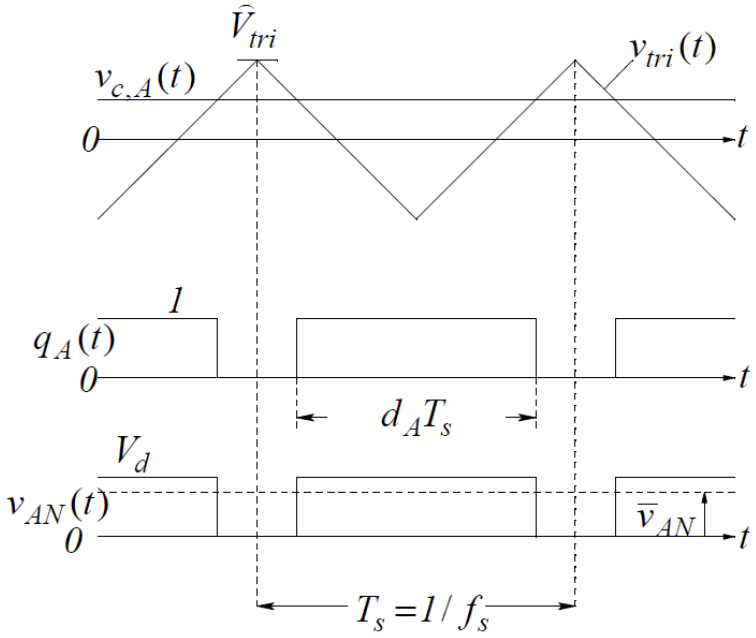


FIGURE 2.2: PWM SIGNAL AND OUTPUT VOLTAGE WAVEFORM FOR ONE BIDGE-LEG (MOHAN, 2003)

A little blanking time interval where both switches are off is implemented for safety reasons to avoid short circuiting of the input voltage, if both MOSFETs occasionally are on simultaneously. It is important that this interval is not too long, since the output current has to be continuous for the linearity of equation 2.1 to apply.

To get a smooth output voltage waveform at the average value over one switching period, a filter consisting of a capacitor and inductor is needed. This is explained later in sub-section 2.2.3.

The duty ratio D of the upper switch' PWM, signal can also be given as a function of the control voltage $v_{control}$ versus the amplitude of the triangular signal $\hat{V}_{tri}$ as shown in equation 2.2.

$$D = \frac{T_{on}}{T_S} = \frac{1}{2}\left(1 + \frac{v_{control}}{\hat{V}_{tri}}\right) \qquad 2.2$$

(Mohan, et al., 2003)

Hence, $v_{control}$ can be used to control the output voltage. It can be held constant for DC – DC converter operation or it can vary sinusoidaly to get a sinusoidal output voltage, for instance. The frequency of the output voltage will be the same as the frequency of the control voltage.

DC AND AC OUTPUT VOLTAGE

For DC-DC converter operation of the half-bridge converter, the output voltage is measured between point A and ground N. The polarity of the voltage cannot be changed. The switches used are MOSFET, and because of their anti-parallel diodes, the current can flow in both directions through the switch when it is on. Therefore, the one-leg converter can operate in two quadrants of the current-voltage plane: +V, +I and +V, -I. The voltage magnitude is controlled by controlling $v_{control}$. The waveforms of this type of operation are shown in Figure 2.2.

For DC-AC operation, two capacitors in series are connected at the DC input, as shown in Figure 2.1. The output voltage is measured between point A and their junction, o. It is important that they are sufficiently large such that the voltage at the point o stays constant at $V_d/2$. The output voltage is thus alternating between $+V_d/2$ and $-V_d/2$.

The right term to use is now *inverter*. To get a sinusoidal output, the control voltage has to vary sinusoidal. This is shown in the figure under. The frequency of the output voltage is given by the frequency of the control voltage, f_1.

$$v_{control} = \hat{V}_{control} \sin(\omega_1 t) \qquad 2.3$$

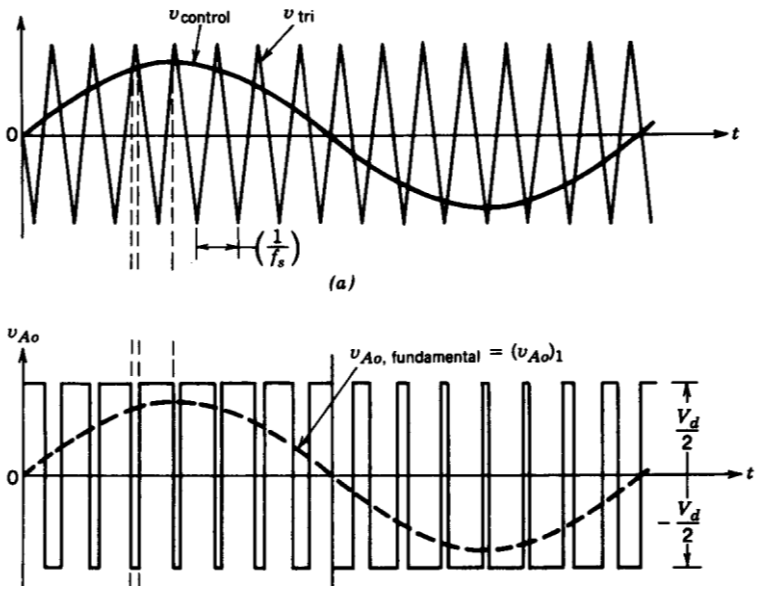


FIGURE 2.3: AC OPERATION OF THE HALF-BRIDGE CONVERTER(MOHAN, ET AL., 2003)

For a sinusoidal $v_{control}$, the ratio between the control signal's amplitude and the triangular signal's amplitude is called amplitude modulation m_a.

$$m_a = \frac{\hat{V}_{control}}{\hat{V}_{tri}} \qquad 2.4$$

(Mohan, et al., 2003)

The frequency modulation is the ratio between the control signal's frequency f_1 and the triangular signal's frequency f_s

$$m_f = \frac{f_s}{f_1} \qquad 2.5$$

(Mohan, et al., 2003)

To get a sinusoidal output voltage, $m_a < 1$. If $1 < m_a < 4/\pi$, the converter operates in overmodulation mode. When $m_a > 4/\pi$, the output voltage is square wave form with duty ratio equal to 0.5. This is because the control voltage now is larger than the triangular waveform amplitude half of its

period, and lower than it the other half, no matter where in the cycle the triangular waveform is.

2.1.2 Two Bridge-Legs

For the two-leg converter, also called full-bridge converter, the load is connected between the outputs of the two legs. The load voltage can thus vary between $+V_d$ and $-V_d$. Therefore, when the one-leg converter operates in two quadrants of the current-voltage plane, the full-bridge converter can operate in all four.

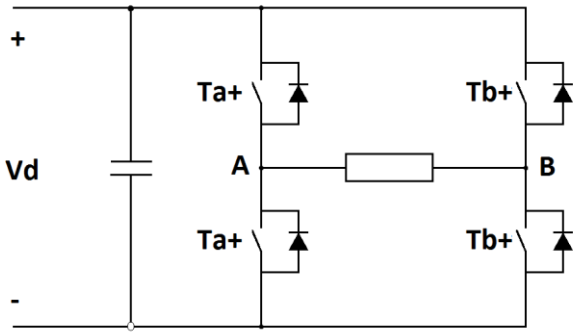


Figure 2.4: Full-bridge converter (Mohan, et al., 2003)

The switching theory described for one bridge-leg apply for the full-bridge converter, but there are two types of switching schemes for how to switch the two bridge-legs relative to each other; bipolar and unipolar voltage switching.

Bipolar switching

Bipolar voltage switching indicates that the opposite switches (Ta+ and Tb-, Ta- and Tb+) are treated as switch pairs and switched simultaneously. The switches in each pair are controlled by the same PWM signal. The possible current direction and voltage polarities for both switch states are shown in the figure under. The current direction depends on if the load is generating or consuming power.

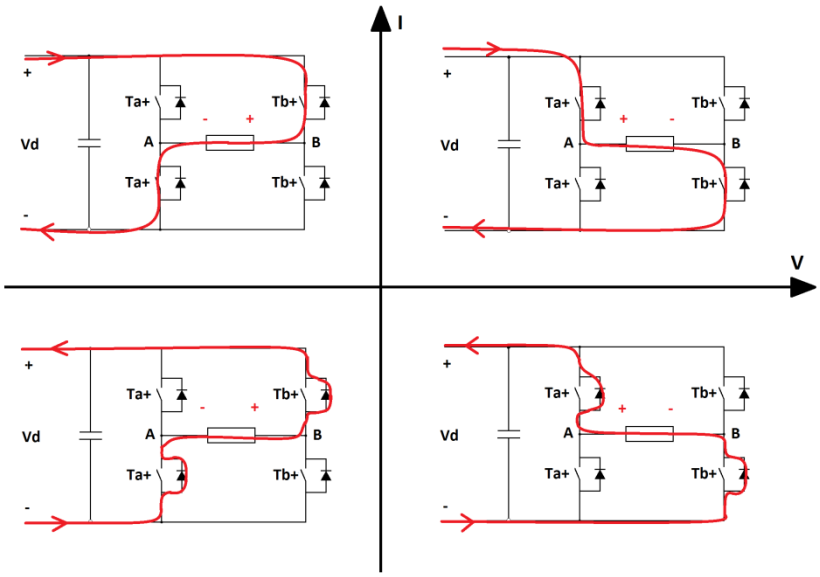


FIGURE 2.5: FOUR-QUADRANT OPERATION OF A FULL-BRIDGE CONVERTER

The output voltage of bridge-leg A with respect to ground N is given as

$$V_{AN} = D_A V_d \tag{2.6}$$

While the output voltage of bridge-leg B with respect to ground N is given as

$$V_{BN} = (1 - D_A)V_d \tag{2.7}$$

Combining equation 2.2, 2.6 and 2.7, the average load voltages is

$$V_0 = V_{AN} - V_{BN} = D_1 V_d - D_2 V_d = (2D_1 - 1)V_d = \frac{V_d}{\hat{V}_{tri}} v_{control} = k v_{control} \tag{2.8}$$

(Mohan, et al., 2003)

As shown in equation 2.8, the average load voltage V_0 is proportional to the control voltage $v_{control}$ which can be varied in both magnitude and polarity.

The voltage and signal waveforms for bipolar voltage switching are given in the figure below.

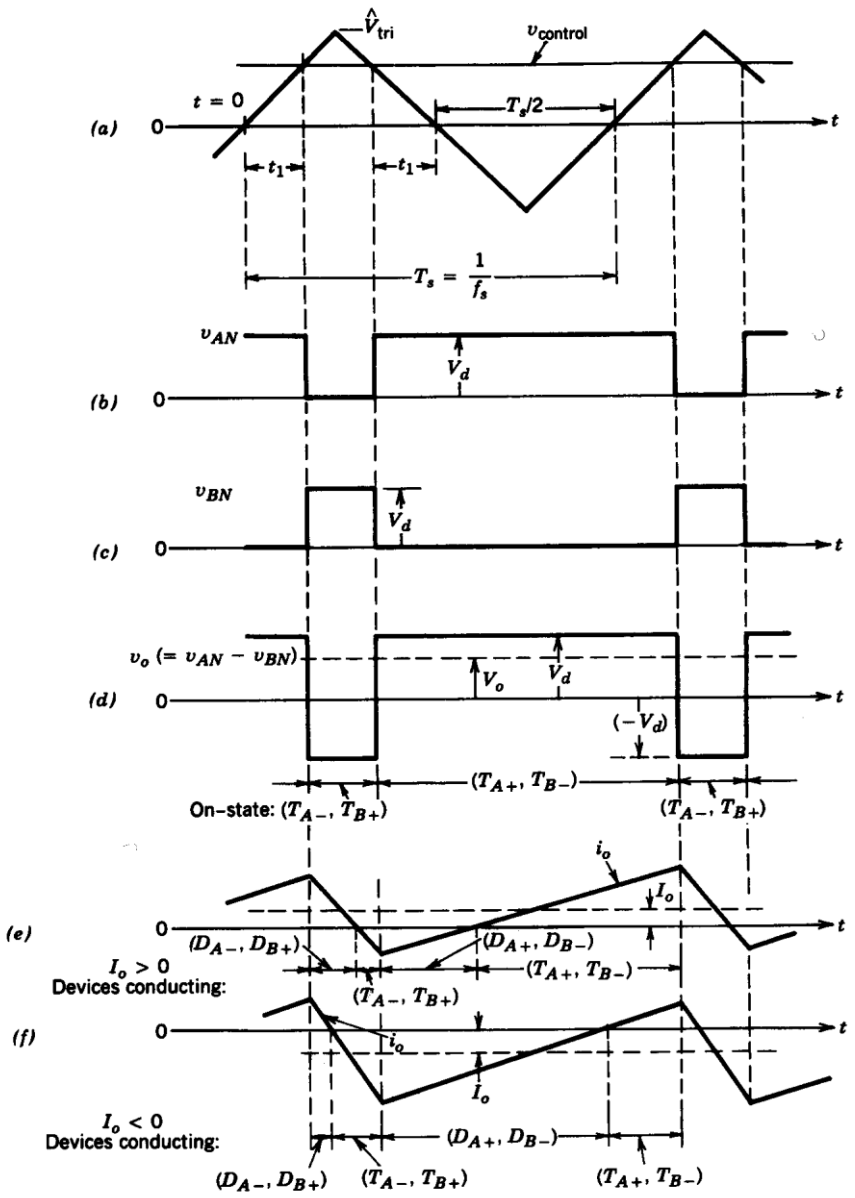


FIGURE 2.6: PWM FOR A FULL-BRIDGE CONVERTER WITH BIPOLAR VOLTAGE SWITCHING (MOHAN, ET AL., 2003)

As for one bridge-leg, the control voltage is held constant for DC-DC operation, as shown in Figure 2.6. For sinusoidal DC-AC operation, the magnitude of the control voltage is varied sinusoidal, as shown in the figure below.

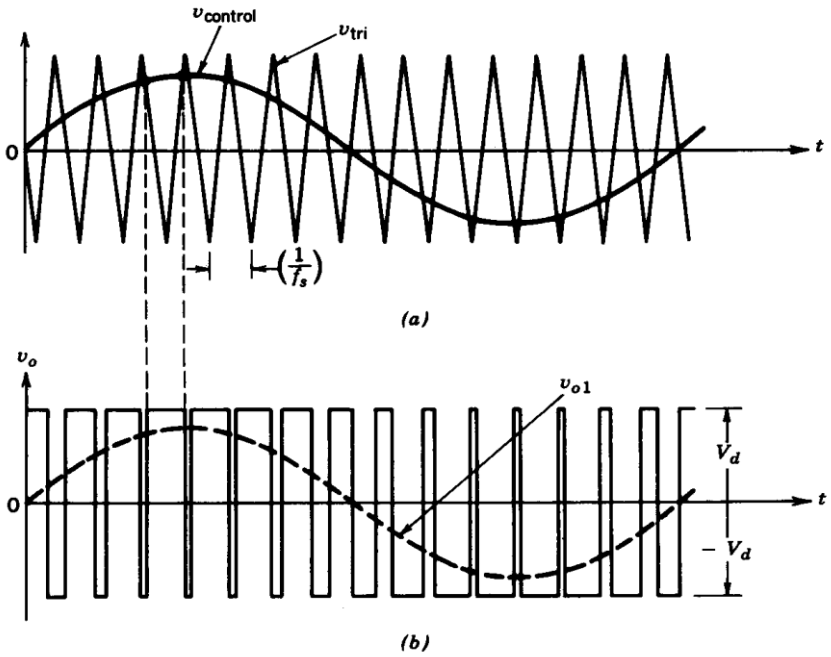


Figure 8-12 PWM with bipolar voltage switching.

FIGURE 2.7: SINUSOIDAL PWM FOR A FULL-BRIDGE CONVERTER WITH BIPOLAR VOLTAGE SWITCHING (MOHAN, ET AL., 2003)

UNIPOLAR VOLTAGE SWITCHING

With unipolar voltage switching, the two bridge-legs are controlled by their own PWM signals made of the two control voltages $+v_{control}$ and $-v_{control}$. The waveforms are given in Figure 2.8.

It can be proven that the average output voltage is proportional to $v_{control}$ and given by equation 2.8, as for bipolar voltage switching. However, within a switching period, the output voltage V_0 will now oscillate between V_d and 0 or between 0 and $-V_d$. The voltage ripple will have a higher frequency, but lower magnitude, as seen in Figure 2.8. The RMS value equals V_d, and is independent of the duty ratio.

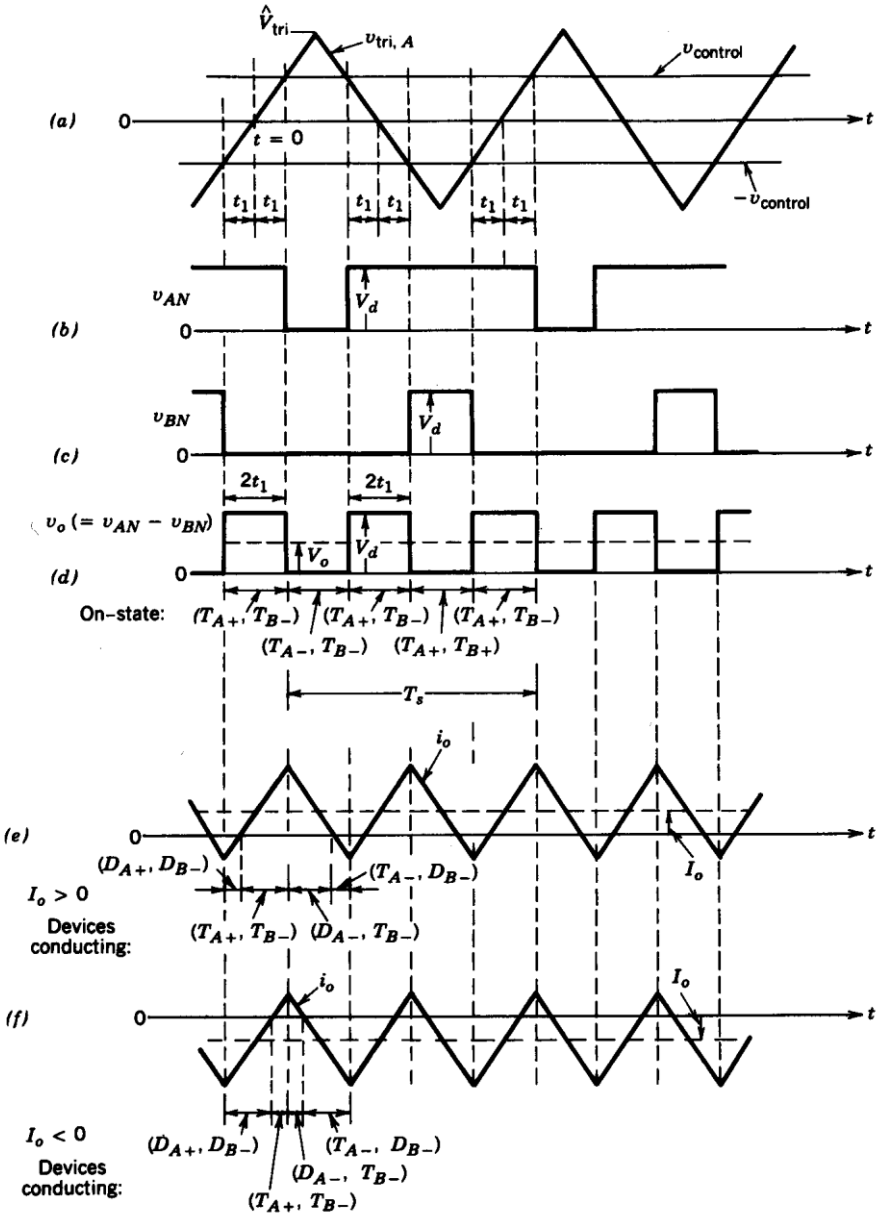


Figure 7-29 PWM with unipolar voltage switching.

FIGURE 2.8: PWM FOR FULL-BRIDGE CONVERTER WITH UNIPOLAR VOLTAGE SWITCHING (MOHAN, ET AL., 2003)

For DC-AC operation with unipolar switching, the waveforms look like in the following figure. The two control signals are of the same magnitude, with opposite polarity. Unlike bipolar voltage switching at AC operation, the output voltage is for half of the cycle alternating between V_d and $0\ V$, while for the other half cycle between $0\ V$ and $-V_d$. The ripple is therefore lower in magnitude.

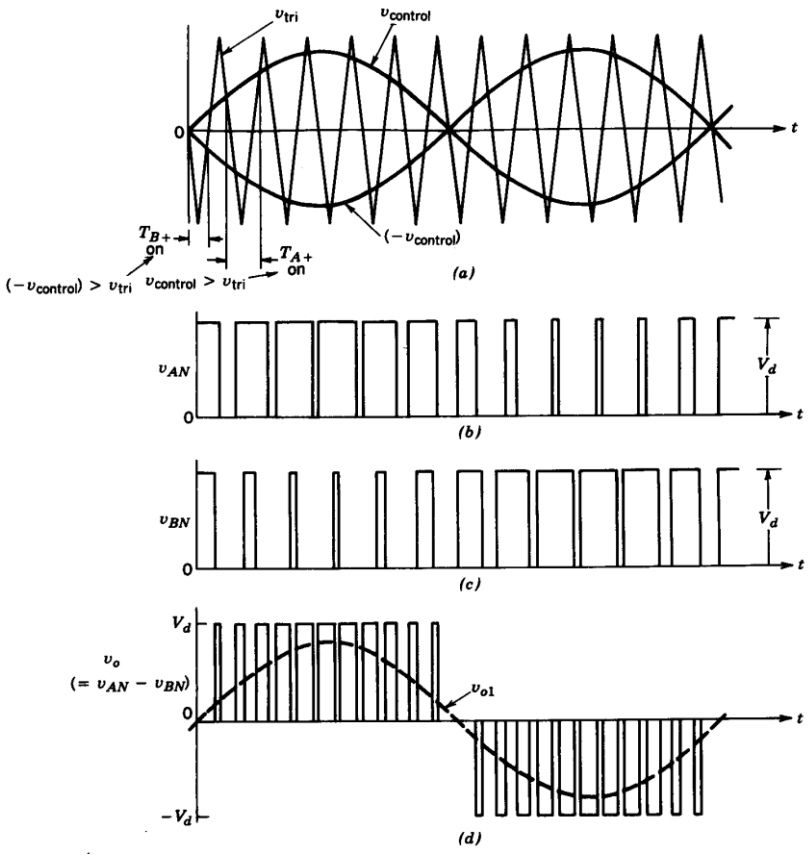


FIGURE 2.9: SINUSOIDAL PWM FOR FULL-BRIDGE CONVERTER WITH UNIPOLAR VOLTAGE SWITCHING (MOHAN, ET AL., 2003)

2.1.3 THREE BRIDGE-LEGS

When all three bridge-legs are used, the topology is a three-phase DC-AC inverter. The PWM signal for three bridge-legs are made of one triangular waveform with three different control voltages varying sinusoidal, 120° shifted from each other. The PWM signals will hence look as in the figure below.

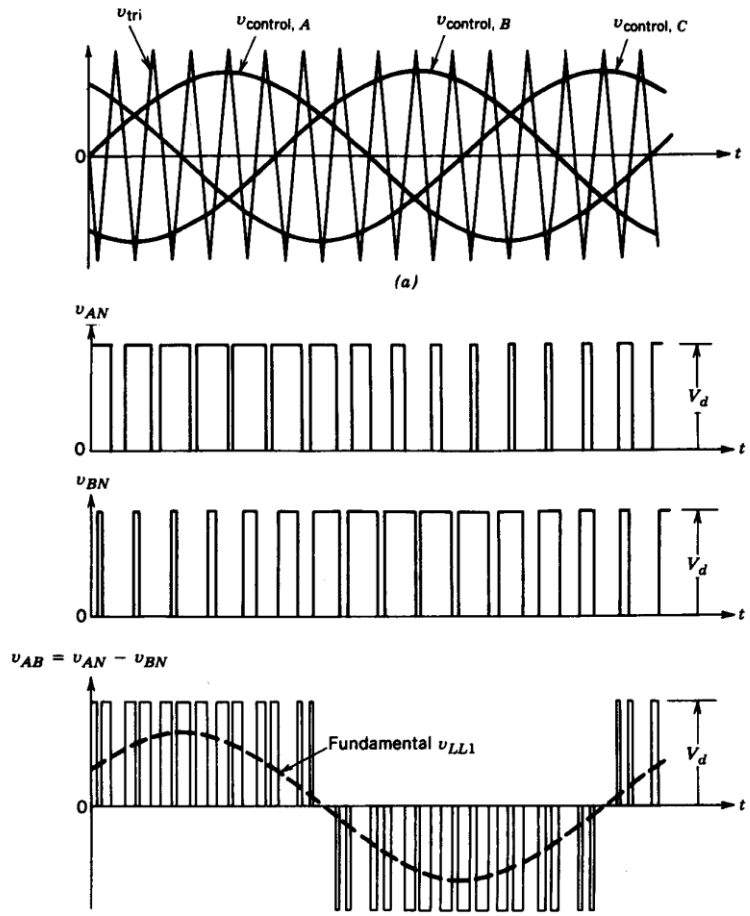


FIGURE 2.10: PWM FOR THREE PHASE DC-AC INVERTER

As explained for one or two bridge-leg operation, for sufficiently large values of m_a, the three-leg converter will also operate in square wave mode. Each bridge-leg will have a duty ratio of 50 %, and the output signals of each are shifted 120° from the others.

2.2 Converter Design and Implementation

The converter is implemented on a printed circuit board (PCB) together with other necessary equipment. This includes the microcontroller connections, MOSFETs driver circuits, output filters, voltage and current measurement circuits, and voltage supplies for the different components used. The different parts and equipments are divided into two groups: the power electronic group including the main power flow, and the control group with the components connected directly to the microcontroller.

To meet the safety requirements in the problem description, the control and the power electronic parts are isolated from each other letting the system have two ground potentials; GND1 for the power electronic part and GND2 for the control part. The power electronic part includes the three-leg converter with filters and voltage supplies. Some of the measurements are also done on this ground potential. The control part includes the microcontroller, some voltage supplies, and some parts of the measurement circuits. Signals crossing the isolation are transferred with isolated components. These are the PWM and the current and voltage measurement signals. There are two sets of voltage supplies, one for each ground potential.

The PCB is supplied by one external power supply. It supplies the converter as well as the voltage supplies which are supplying the different components. Its maximum voltage V_d and current I_d is respectively 50 V and 12 A.

A block diagram of the circuit is shown in the figure under. Signal scaling refers to the scaling of the measurement signals to fit the microcontroller input. This will be explained later.

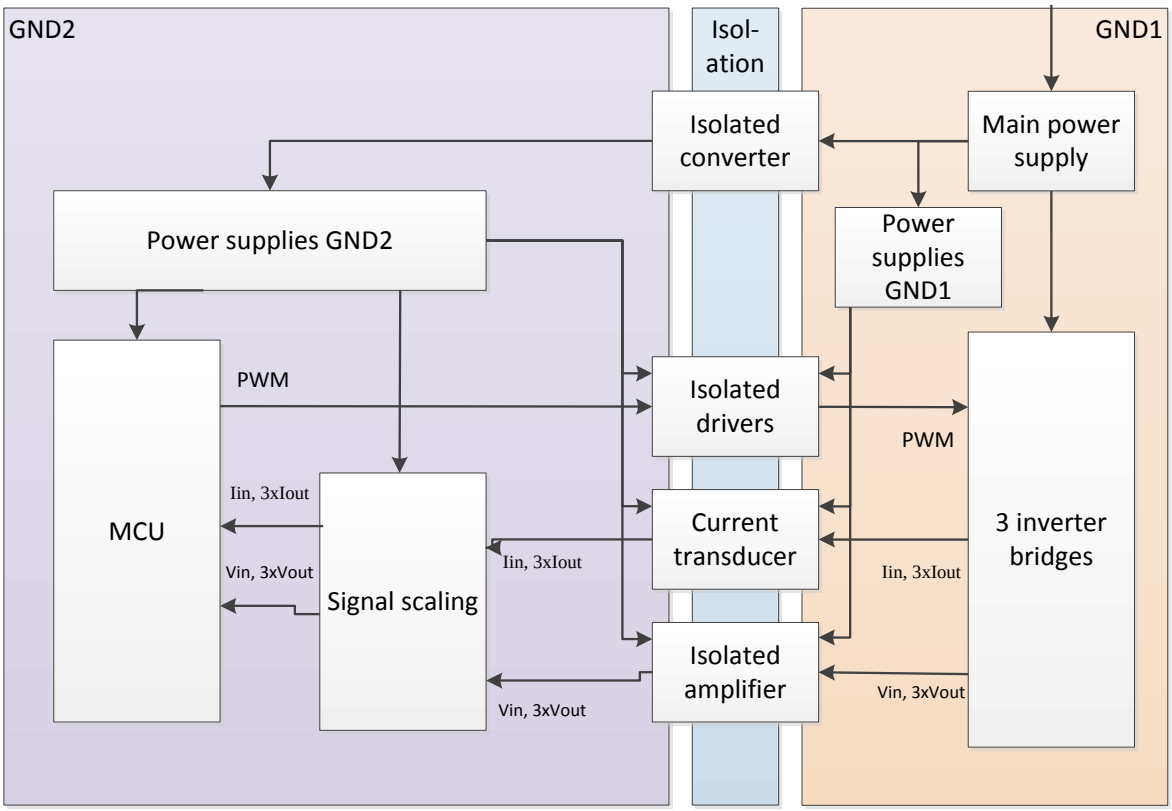


FIGURE 2.11: BLOCK DIAGRAM OF THE PRINTED CIRCUIT BOARD

The design of the schematics and board layout is done in CadSoft Eagle™ PCB Design Software and based on the work of Ishengoma, et al. (2011). The file is called master. The 5 sheets of the schematics are given in appendix A in figure A-1 to A-5. Sheet 1 and 2 shows the power supplies and the test-pin outputs. In sheet 3 is the main power flow including the converter's bridge-legs with MOSFETs and drivers, the current and voltage measurements, the filter and the three output connections. In sheet 4 one can find the signal scaling circuits for the current and voltage measurements to fit the voltage levels of the microcontroller. The last sheet, number 5, shows the connections to the microcontroller.

The different parts, components and layout, of the circuit will now be explained in more detail. The components used and their ratings are listed in table A-1. The first page of each datasheet is given in appendix F. All values for the different components are gotten from the datasheets.

2.2.1 MICROCONTROLLER

For the converter control, the Texas Instruments TMS320F28069™ Piccolo™ controlSTICK microcontroller unit (MCU) is chosen (Texas Instruments D, 2012). It is generating PWM signals for the switches and does an analog-to-digital (ADC) conversion of the input and output current and voltage measurements.

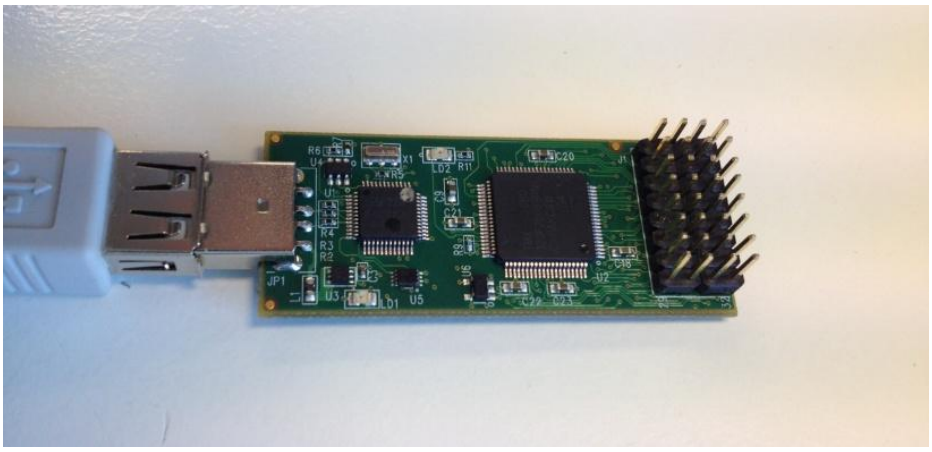


FIGURE 2.12: TEXAS INSTRUMENTS PICCOLO™ CONTROLSTICK

This microcontroller is first of all chosen because it is used in other laboratory work at NTNU and was therefore readily available. According to Texas Instruments (2012 C) it is suitable for this purpose because of its high efficiency, low cost, real-time control, high frequency of 80 MHz, fast interrupt response and processing. It is a 32-bit microcontroller, which is adequate enough for this system. It is made for high precision and efficiency for systems such as solar inverters, white goods appliances, hybrid automotive batteries, power line communications (PLC) and LED lightning (Texas Instruments C, 2012).

Depending on which pins that should be available, one can either choose the controlSTICK or controlCARD layout. The controlSTICK is suitable for controlling the converter since both the PWM output pins and the ADC pins are available.

The code controlling the microcontroller is implemented in Code Composer Studio™ (CCStudio). This is an integrated development environment (IDE) for developing and debugging embedded applications controlled by Texas Instruments embedded processors (Texas Instruments B, 2012). Among some, it comprises compilers, source code editor, project build environment, debugger, profiler, simulators and real-time operating system (Texas Instruments B, 2012).

The MCU needs a voltage supply of 5V, which is usually gotten from the computer (Texas Instruments D, 2012). All other connections are between 0 and 3.3 V (Texas Instruments A, 2011). This indicates that the PWM signal alters between 0 and 3.3 V and that the signals to be converted must be within this range. The analog-to-digital conversion (ADC) ratio of the controller is given by equation 2.9.

$$Digital\ value = 0, when\ input\ < 0\ V$$

$$Digital\ value = \frac{4096}{3.3} * input, when\ 0 \leq input\ \leq 3.3\ V \quad (2.9)$$

$$Digital\ value = 4095, when\ input\ \geq 3.3\ V$$

(Texas Instruments A, 2011)

2.2.2 The Three Converter Legs

The switches in the bridge legs are MOSFETs controlled by isolated drivers, one per MOSFET. The drivers are marked pink and the MOSFETs are marked yellow in sheet 3 of the circuit schematics below (full-page figure in Figure A-3). The MOSFETs and its driver are numbered from the upper left corner, such that the first bridge-leg consists of MOSFET 1 and 2 and so on.

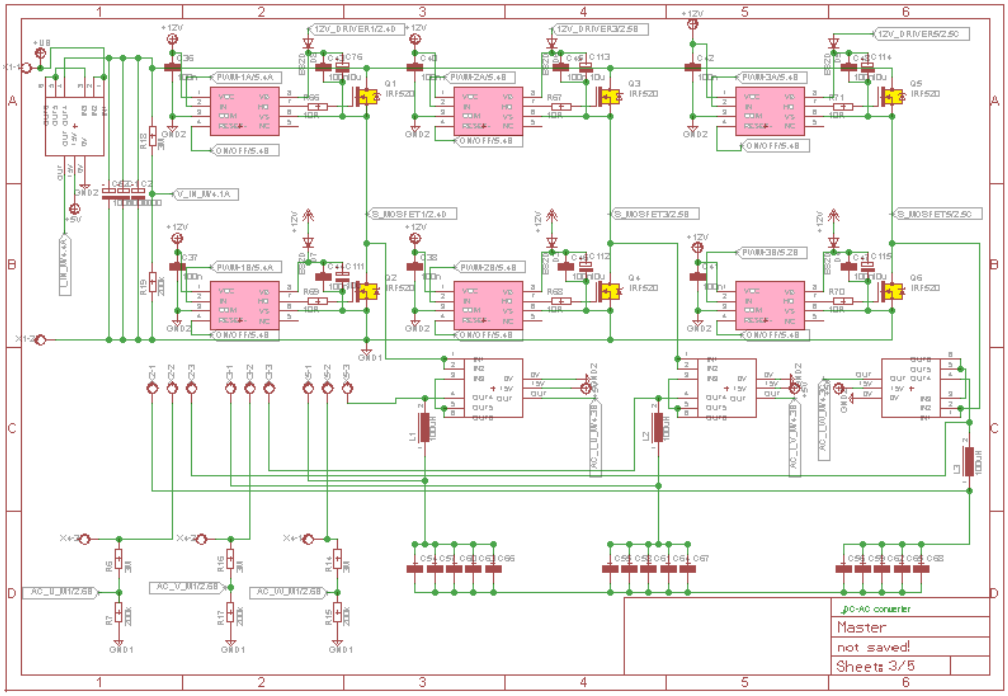


FIGURE 2.13: CLIP OF SHEET 3 WITH THE DRIVERS AND MOSFETS MARKED

The MOSFET chosen is IRFB4110PbF (International Rectifiers, 2011), because of its high switching speed and suitable voltage range. The turn-on time is less than 0.1 µs, while the turn-off time is about 0.15 µs at voltage and current conditions larger than needed in this study. This indicates that a switching frequency of 50 kHz, as intended in this work, with a period of 20 µs is suitable for most duty ratios. With a duty ratio of (0.1µs+0.15µs)/20µs=0.0125=1.25%, the MOSFET just reaches to turn on and off once.

The MOSFET needs a gate-source voltage of about 10V to turn completely on. The drivers steps the PWM signal up from 3.3 V and isolate it while transferring it from the microcontroller to the MOSFETs and power electronic circuit.

A suitable driver found is IRS2123S (International Rectifiers, A, 2009). Although the driver is not galvanic isolated, the silicon which it is made of isolates the high-side from the level shifters, this makes it suitable for this purpose (International Rectifiers, 2012).

A closer figure of the circuit layout of one bridge-leg is shown below. The driver needs a voltage supply of around 12 V on both microcontroller and MOSFET side. The capacitances are there to reduce noise and to keep the voltage stable at its connections. The driver's reset pin, RESET-, needs to be set high (3.3 V) for the driver to turn on. This is connected to and controlled by the microcontroller.

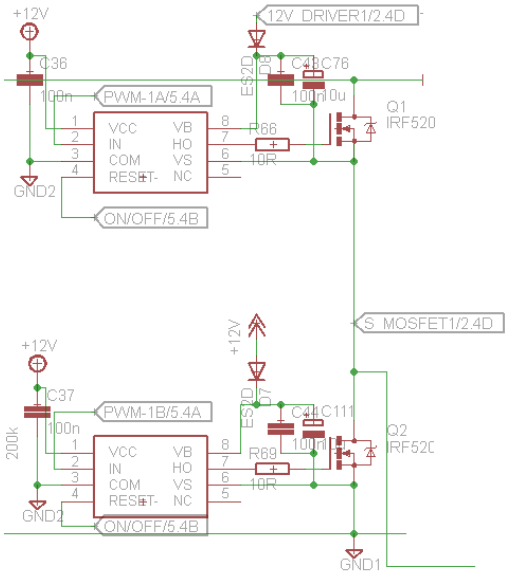


FIGURE 2.14: ONE BRIDGE-LEG (CLIP OF SHEET 3)

2.2.3 OUTPUT FILTER

As described in section 2.1, the converter output voltage will have a square-wave form. To smooth this, a LC filter is included. It is marked blue on Figure 2.15. A simplified circuit diagram of the filter is given in Figure 2.16.

The filter must be customized for the specific load used, since all components include some inductance and capacitances. Therefore, at first eyesight it may look as if there are several different capacitors in the filter, but only one should be used. Doing it this way gives the ability to try different capacitors and see which works best for the final system. The coils are made by a core named "Amidon T106-26 stacked" and can be ordered

from www.reichelt.de. The number of windings determines the inductor value.

For motor control, for instance, the filter is not needed because the motor filters the voltage itself. To choose whether the filter should be used or not, switches are placed such that the filter can be included or bypassed. The switches connections are marked pink in Figure 2.15. They consist of three pins per bridge-leg output. The middle one is the output, and one can either connect it to the right to bypass the filter or to the left to include the filter.

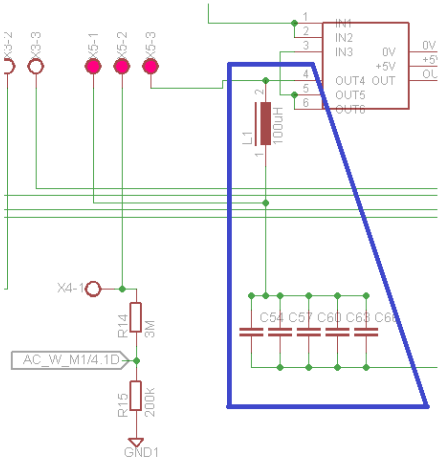


FIGURE 2.15: THE OUTPUT VOLTAGE FILTERS (CLIP OF SHEET 3)

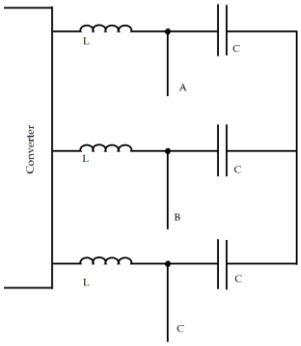


FIGURE 2.16: A SIMPLIFIED DIAGRAM OF THE FILTERS

2.2.4 The Current and Voltage Measurements

The currents and voltages are both measured at the input and output of the converter for all three bridge-legs. The measurement devices are shown in sheet 3 of the circuit diagrams in appendix A, figure A-3. The sheet is reproduced in a smaller version here (Figure 2.17) where the current measurement devices are marked pink and the voltage measurements marked yellow. The measurement signals out of the devices are first converted by an analog converter circuit to be within the accepted voltage range of the microcontroller of 0 - 3.3 V. They are then converted to digital by the microcontroller. They also have to be isolated somewhere on the way from the power electronic part to the microcontroller. The current and voltage measurement circuits will be explained separately in the following text.

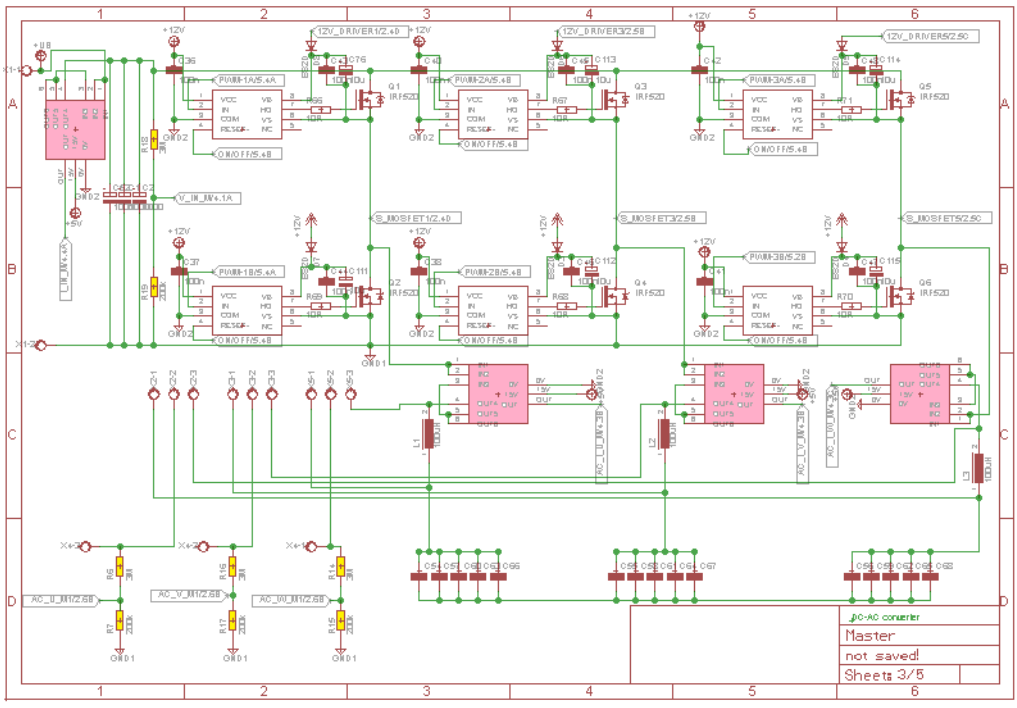


Figure 2.17: Clip of sheet 3 with the current transducers (pink) and voltage dividers (yellow) marked

CURRENT MEASUREMENTS

A current transducer is chosen for the current measurements, because of its directly galvanic isolation of the measurement signal. They are marked pink on Figure 2.17. The main current flows through three coils in the transducer which induces an output voltage, called the current measurement signal, V_{I_M}. This signal is proportional to and isolated from the power current.

The transducer used is a LEM Current Transducer LTS 25-NP (LEM, u.d.), chosen because of its suitable voltage and current levels. The transducer's measurement-current range is determined by how the three coils are interconnected. To get a +/-12 A input current range, the first two coils (pin 1,6 and 2,5) must be parallel connected, while the last coil (pin 3,4) is connected in series with these two. This is shown in Figure 2.17.

The transducers need a voltage supply of +5 V on the GND2 side. The output signal range is between 0.5 and 4.5 V and it is given as a function of the current I in 2.10.

$$V_{I\_M} = \frac{I + 15}{6} \tag{2.10}$$

To get these signals to the range of the microcontroller from 0 to 3.3 V, an analog-signal-scaling circuit is needed. It mainly consists of an operational amplifier which is connected as a differential amplifier. The operational amplifier chosen is a TLC274INE4 (Texas Instruments, 1987). It includes all four amplifiers needed, one for each current signal, and need a supply voltage of +/-15 V.

The main feature about difference amplifiers is that its output is the difference between its +pin and –pin input signal, amplified with the ratio of R_2 over R_1. R_1 is the resistance between the two input signals and the amplifier, while R_2 is the resistance in the feedback loop between the output and the –pin and between the +pin and ground. A circuit diagram of this is shown in Figure 2.18.

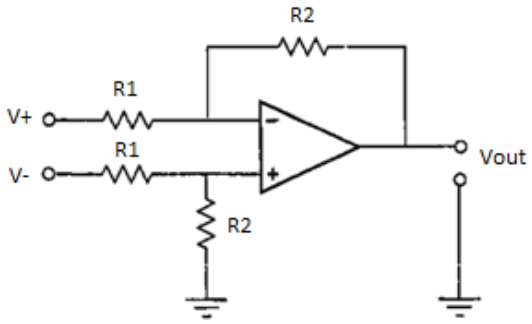


FIGURE 2.18: DIFFERENTIAL AMPLIFIER TOPOLOGY

The difference amplifier transfer function is given as

$$V_{out} = \frac{R_2}{R_1}(V_+ - V_-) \quad 2.11$$

To get the current measurement signal V_{I_M} from 0.5-4.5 V to 0-3.3 V, it is connected to the +pin and 0.5 V is connected to the –pin. R_1 = 100 kΩ and R_2= 75 kΩ, hence a gain of 0.75. The analog signal scaling circuit is shown in Figure 2.19. The output signal $V_{I\_M\_DIG}$ is given as

$$V_{I\_M\_DIG} = \frac{75k\Omega}{100k\Omega}(V_{I\_M} - 0.5V) \quad 2.12$$

$V_{I\_M\_DIG}$ is connected directly to the microcontroller and V_{I_M} is the current measurement signal from the transducer connected at the opamp input.

Combining equation 2.10 and 2.12, one gets the current value into the microcontroller as a function of the measured current to be

$$V_{I\_M\_DIG} = \frac{75k\Omega}{100k\Omega}\left(\frac{I + 15}{6} - 0.5V\right) \quad 2.13$$

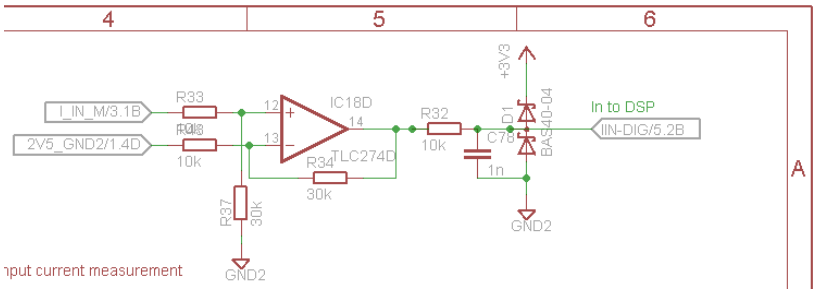


FIGURE 2.19: THE CURRENT ANALOG CONVERTER CIRCUIT (CLIP OF SHEET 4)

The last part of the analog signal converter is a RC low-pass filter consisting of a resistor and a capacitor with a cutoff frequency of 16 kHz, which suites a switching frequency of 50 kHz (Ishengoma, 2011).

2.14

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi * 10k\Omega * 1nF} = 16kHz$$

(Balchen, et al., 1999)

(Ishengoma, et al., 2011)

The two zener diodes make sure the voltage into the microcontroller stays between 0 and 3.3 V at any time.

VOLTAGE MEASUREMENTS

For the voltage measurements, voltage dividers made of large precision resistances (+/- 1 %) are used get the voltage measurement signal into the range of the microcontroller of 0-3.3 V. The resistances need to be large to minimize their affection on the power circuit. The voltage dividers are marked yellow in Figure 2.17. They are placed after the filter such that the smooth output waveform is measured if the filter is included.

The voltage range for both the input and output voltages is between 0 and V_d = 50 V. The resistances chosen are 3 MΩ and 200 kΩ making the voltage measurement signal V_{V_M} equal to 3.125 V for a maximum input voltage of

50 V. The voltage divider affects the power circuit by drawing a current of 50V/3.2 MΩ = 15.6 µA, which is negligible.

Each voltage measurement signal is then isolated by an isolation amplifier, with gain equal to 1. Hence the voltage measurement signal to the microcontroller $V_{V\_M\_DIG}$ equals V_{V_M}. These are given as a function of the measured voltage by equation 2.15.

$$V_{V\_M\_DIG} = V_{V\_M} = \frac{200\ k\Omega}{200\ k\Omega + 3000\ k\Omega} * V = 0.0625 * V \qquad 2.15$$

Table 2.1 shows the maximum error with the 1 % precision resistances used. The error is small and hence negligible.

TABLE 2.1: MAXIMUM ERROR OF THE MEASUREMENT SIGNAL WITH 1 % PRESISION RESISTANCES

R1	**R2**	V_{V_M}	$V_{V\_M}(V = 50V)$	$V(V_{V\_M}) = \frac{V_{V\_M}}{0.0625}$	**% ERROR**
200kΩ	3 MΩ	0.0625*V	3.125 V	50 V	0
200kΩ+1 % = 202 kΩ	3 MΩ - 1 % = 2.97 MΩ	0.0637*V	3.184 V	50.944 V	1.888 %
200 kΩ - 1 % = 198 kΩ	M mΩ + 1 % = 3.03 MΩ	0.0613*V	3.065 V	49.04 V	- 1.92 %

The isolated amplifier chosen is ISO124D (Burr-Brown Products from Texas Instruments, 1997) because of its simplicity, suitable voltage range and low power consumption of maximum 7 mA. It needs a supply voltage of +/- 15 V on both GND1 and GND2 sides.

Before the signal is connected to the microcontroller, it passes through a low-pass filter and the safety diodes, the same as for the current measurements. The isolation amplifier, filter and diodes are shown in Figure 2.20.

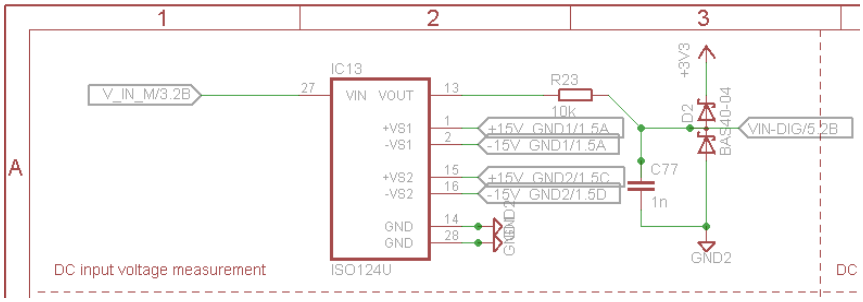


FIGURE 2.20: ISOLATION AMPLIFIERS FOR THE VOLTAGE MEASUREMENT SIGNAL (CLIP OF SHEET 4)

2.2.5 VOLTAGE SUPPLIES

The components presented above need voltage supplies of different voltage levels. These are made of IC (integrated circuits) DC-DC converters. A block diagram of the voltage supplies and the components they are supplying are shown in Figure 2.21. There, the name of the load or voltage supply, its manufacturer's part number and input voltage rating is written in the boxes. The number on the arrow into and out of the box is respectively the input and output current rating. The green box is the main power supply which also supplies the converter. The details of the supplies which are not self-explained by the figure, are further described below.

The names of the power supplies are given by the output voltage level and the ground potential. +12V_GND2 is +12V with respect to GND2. +12V_DRIVE1 is the +12V supplying the MOSFET side of driver 1.

The circuit diagrams of the power supplies are shown in the schematics in sheet 1 and 2 and a complete list of power supplies is given in Table A-1.

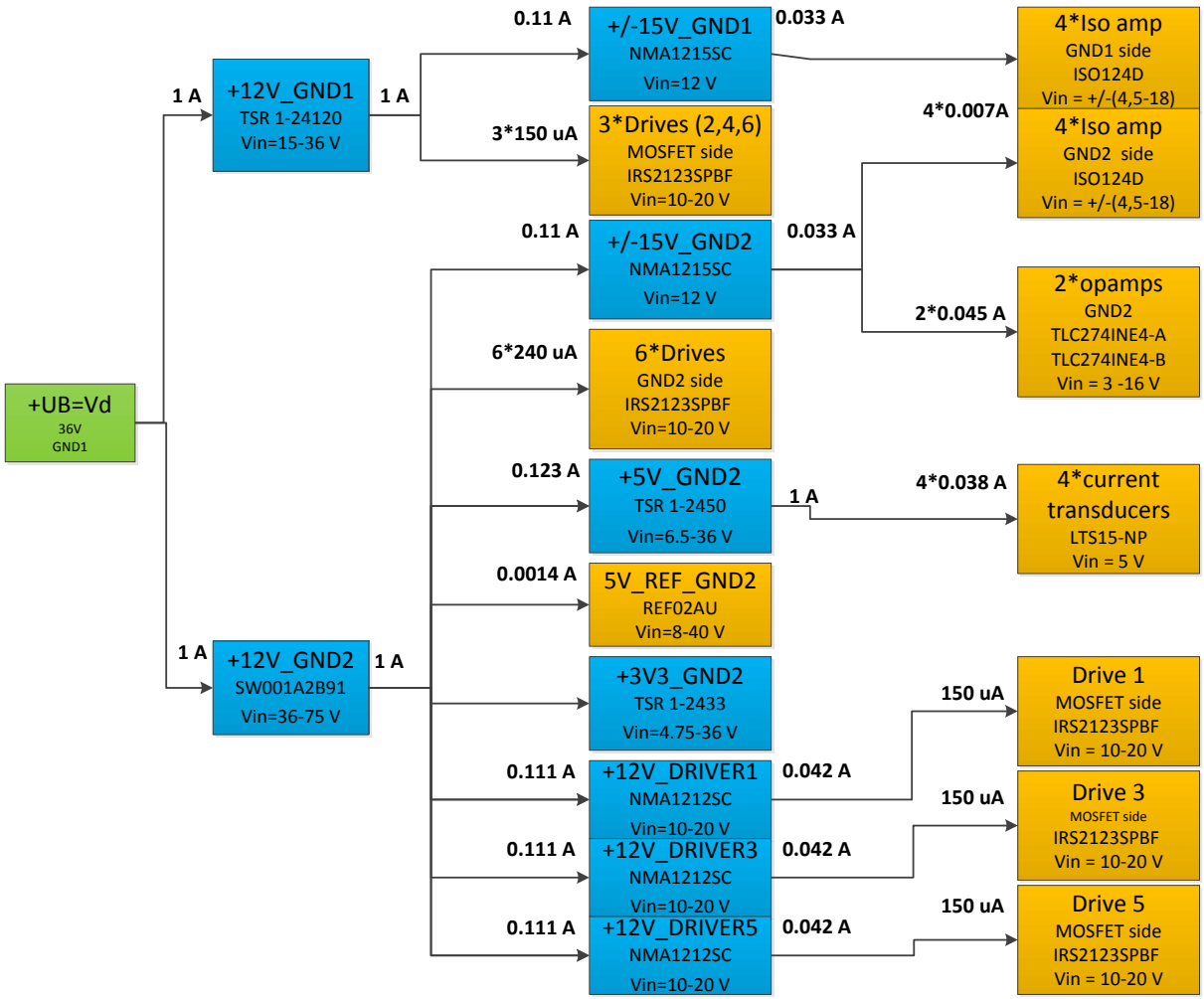


FIGURE 2.21: AN OVERVIEW OF THE POWER SUPPLIES ON THE PCB

(Burr-Brown Products from Texas Instruments, 1993)(Burr-Brown Products from Texas Instruments, 1997)(International Rectifiers, 2011)(International Rectifiers, B, 2009)(International Rectifiers, A, 2009)(LEM, u.d.)(Lineage Power, 2009)(Murata Power Solutions, Inc., 2012)(National Semiconductor, 2006)(Texas Instruments, 1987)(Traco Power, 2009)(Würth Elektronik, 2005).

The main input voltage V_d is directly supplying +12V_GND1 and +12V_GND2 which further supplies all components within each ground group. The main supply is referred to GND1 and hence the +12V_GND2 DC-DC converter must be isolated. Both of them should have an output current of up to 1A to manage the supply of the other devices with a good margin. For +12V_GND1 the TSR 1-24120 (Traco Power, 2009) is chosen

and for +12V_GND2, the SW001A2B91 (Lineage Power, 2009) is chosen. They are shown at the coordinates A1-A2 and B1-B2 in sheet 1 of the schematics in Figure A-1.

The +12V_GND1 converter requires an input voltage range of 15-36 V (Traco Power, 2009) while the +12V_GND2 one requires an input voltage range of 36-75 V (Lineage Power, 2009). Since they are supplied by the same supply V_d, clearly, this is not the best solution and must be considered for the next version. In the mean time, the input voltage V_d has to be fairly constant at 36 V, and cannot be used to control the input voltage of the converter. Therefore, the control of the inverter bridge-legs has to stand for the control of the output voltage.

The MOSFETs' drivers are supplied by +12V_GND2 on the microcontroller side. On the MOSFET side, the drivers must be supplied by +12V with respect to the MOSFET's source's potential. For the three lower drivers (2, 4, 6), source is connected to GND1, and they thus supplied by +12V_GND1. The three upper drivers (1, 3, 5) have their source potential at the bridge-leg's output, which is alternating between 0 and V_d. They are therefore supplied by a voltage source supplying +12V with respect to that potential. Hence, +12V_DRIVER1, +12V_DRIVER3 and +12V_DRIVER5 which are shown in sheet 2 of the schematic (Figure A-2).

5V_REF_GND2 (REF02AU (Burr-Brown Products from Texas Instruments, 1993) is the 5V reference voltage which the 0.5 V voltage reference is made of. This is done by an operational amplifier (TLC274INE4 (Texas Instruments, 1987)) and precision resistances (+/- 1%), as shown in coordinates B3-D4 of sheet 1, Figure A-1. The operational amplifier is the same type as for the current measurement scaling circuit and also this one is also supplied by the +15V_GND2 supply.

2.3 Printed Circuit Board (PCB) Layout

Next, the layout of the printed circuit board (PCB) will be explained. In the process of placing the components, there were several considerations to take, some even conflicting each other.

First of all, the two ground potentials should be physically separated from each other, and all components placed on their respective areas. The components bringing signals across from one potential to the other should be placed right between the two potentials.

To reduce electromagnetic interference issues, the components in the power electronic circuit should be placed as close to each other as possible, and the wires should be as short as possible. In this part of the board, there should also be room for heat sinks on the 6 MOSFETs. All power supplies should be placed fairly close to the object they are supplying, as far it is possible. The noise reducing capacitors must be placed as close to the components they are protecting, this is very important. In addition, signals and voltages should travel as short as possible.

The printed-circuit-board layout is shown as an illustration along the text in Figure 2.22 and as a full-page picture in the figure A-6. The board has two conducting layers, one on top and one on the bottom. The current paths on the top side are marked red and on the bottom side blue. The current paths of the power circuit are made wide in order to conduct the large current of up to12 A. The two ground potentials are placed as areas on the bottom side, shown by the two large blue areas. The surface mounted (SMD) components should be soldered on the white pads, while the through-hole components soldered onto the green pads. The component’s areas are marked white.

The components were placed in steps. As a starting point, the two ground potential areas were roughly set. Then, all components were divided into

three groups: GND1, GND2 and isolation components. The third step was the layout of the GND1 area. It was tried to get the wires as short as possible. But the space required by the components and heat sinks on the MOSFETs sat a limit of how close they could be placed. The MOSFETs should be placed close to the driver circuits placed in the isolated area. The filters were placed furthest away from the isolated area and GND2 since no connection between these are required.

Some compromises had to be made. The current transducers had to be placed between the converter bridge-legs output, and the filter to shorten the power circuit wires, even if they should be placed at the isolation area. Since the drivers were placed at the isolating area close to the MOSFETs, there was no room for the current transducers there anyways. In order not to let the wires with the signals from and supply to the current transducers, split any of the ground potential areas, they are drawn around, which made them very long.

The next step was the layout of the isolation components. It was tried to get the drivers as close to the MOSFETs as possible. The +12V_DRIVE1/3/5 power supplies could just as well be placed near the drivers, since those are their only load. To place the voltage-measurement circuits near the output connections and at both ground potentials, the GND2 area had to be expanded along the right edge of the board.

The next step was to place the GND2 components which are the power supplies, current measurement signal scaling circuit, and the MCU connections. The two +12V power supplies were placed near the input voltage. Wires go from them along the isolation area to their loads. The +/-15V_GND1 were placed by its only load, the isolation amplifiers. The current measurement circuit was placed near the 3V3_GND2 and +/-15V_GND2 supplies in the bottom right corner and the +5V_GND2 was placed near those.

The last step was the wiring. The wires which lead a large current are made wide, and the others are thinner to save space.

The placing of components could have been made easier and the wires shorter with a board containing more than two conducting layers. The drawback with this is that the wires cannot be accessed and hence not cut or redirected if any faults should be found later.

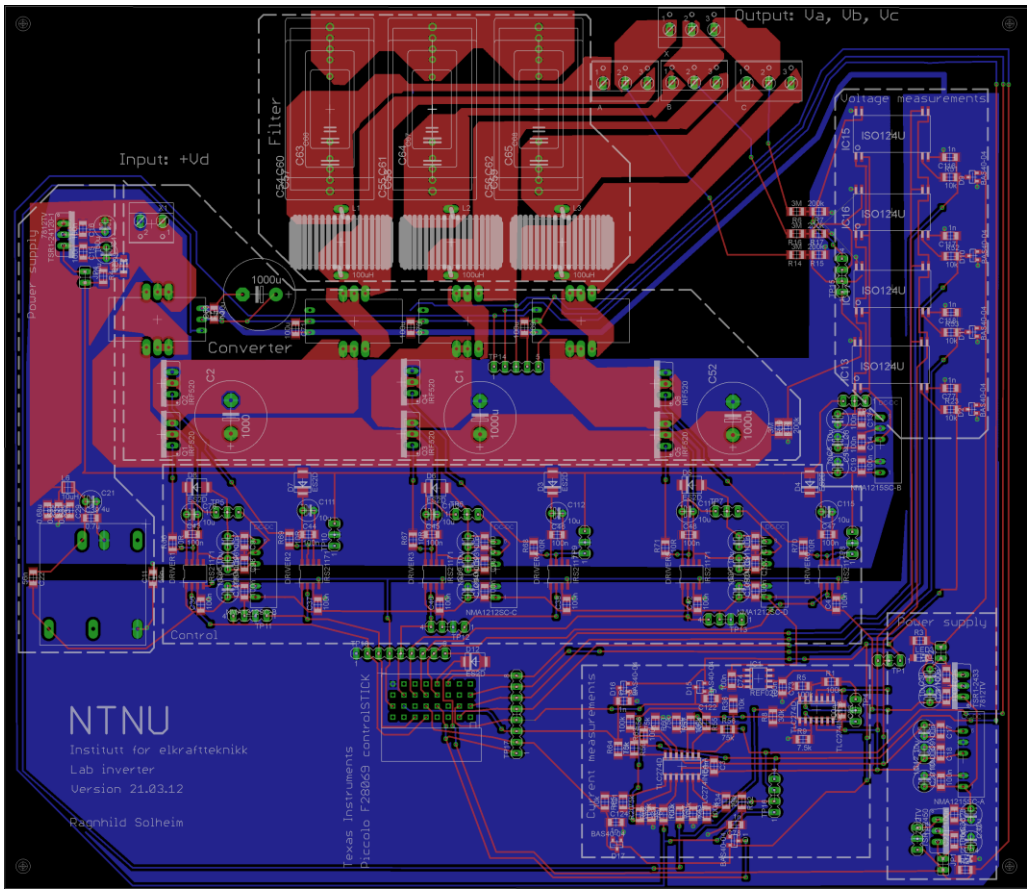


FIGURE 2.22: PRINTED CIRCUIT BOARD LAYOUT

Now, a more detailed description of the layout chosen is given to show where exactly the different components are placed and how they look on the board print.

The output filters are placed in the top centre of the board, a clip of the filter from circuit board is given below. The capacitors are the squares while the inductances are the white striped areas. Here one can see that it is possible

to connect different capacitors to see which works best for the final system, as explained in sub-section 2.2.3.

The output connections, A, B, C, are placed to the right of the filters. The three switches used to bypass or use the filters are placed right under the output. They are as explained in sub-section 2.2.3.

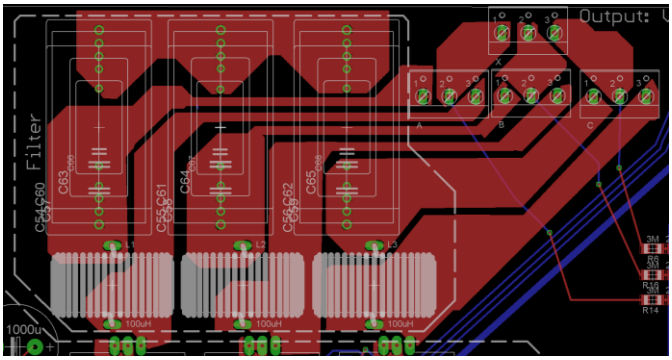


FIGURE 2.23: THE FILTER

In the area called ‘Converter’ one can find the MOSFETs, current transducers and capacitors. The current transducer placed separately from the others is measuring the input current. The capacitors are placed close to each bridge-leg to reduce noise.

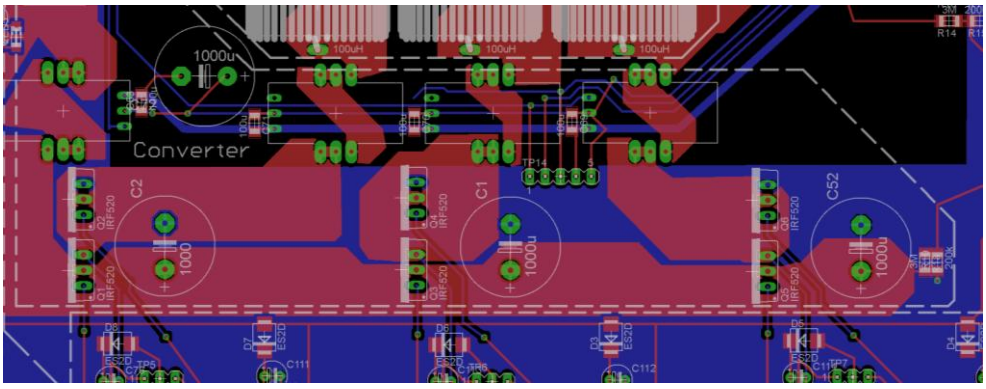


FIGURE 2.24: THE CONVERTER BRIDGE-LEGS

The control part of the MOSFETs is placed under its respective converter bridge-legs, in the area marked “Control”, see the clip under. Here, one can see the two ground potentials marked blue with the neutral area marked

black between them. The GND1 is on the upper side. The components transferring signals from one side to the other are partly on each potential.

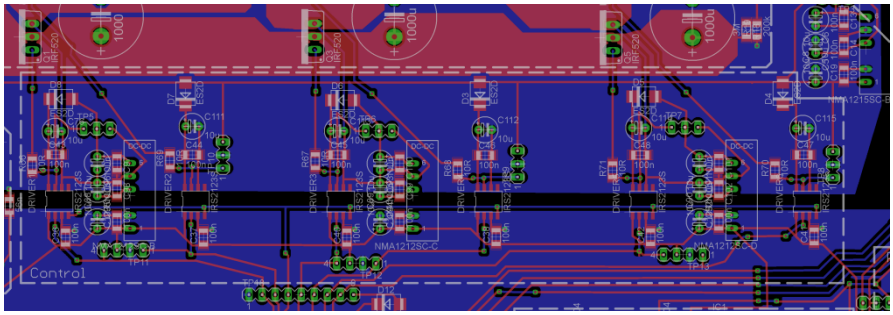


FIGURE 2.25: THE MOSFET DRIVER CIRCUITS

Figure 2.26 gives a closer look at the driver circuit components for one bridge leg. One can see the two drivers, one on each side and the 12V DC-DC converter supplying the upper MOSFET is in the middle getting its voltage supply from 12V_GND2.

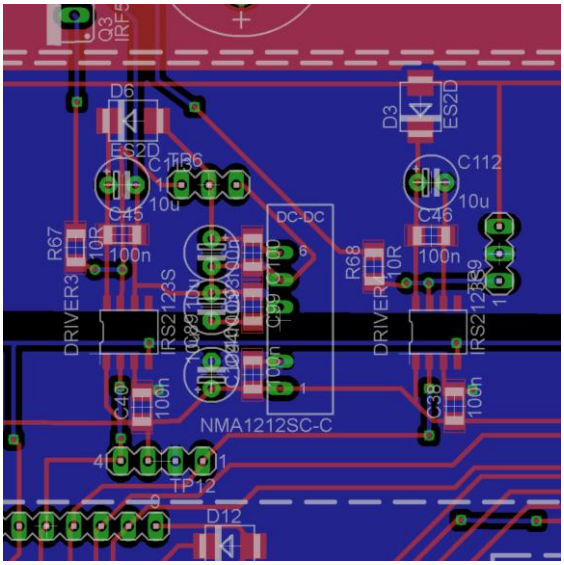


FIGURE 2.26: THE CONTROL CIRCUIT FOR ONE CONVERTER BRIDGE-LEG

A clip of the voltage measurement circuit is given in Figure 2.27. The large squares are the isolation amplifiers and in the bottom is the +/-15V DC-DC converter supplying the GND1 side of the amplifiers. The voltage dividing resistors are placed between the outputs and the isolation amplifiers.

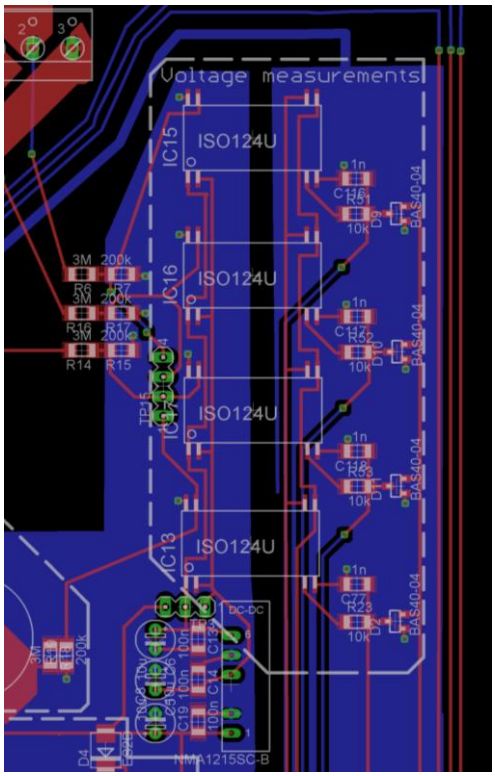


FIGURE 2.27: VOLTAGE-MEASUREMENT-SIGNAL ISOLATION

The main power input is in the upper left corner. Both the GND1 and GND2 12V voltage supply are placed near the input within the area called “Power Supply”. A clip of this part is shown under.

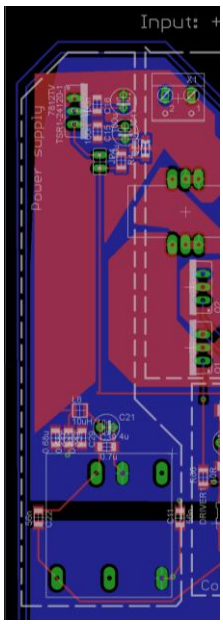


FIGURE 2.28: THE TWO +12V CONVERTERS

The current measurement circuits are placed in the lower middle of the board, a clip is shown in Figure 2.29. In the upper right corner are the reference voltages and in the middle is the IC with four operational amplifiers doing the analog signal scaling of the current measurements before they are connected to the microcontroller.

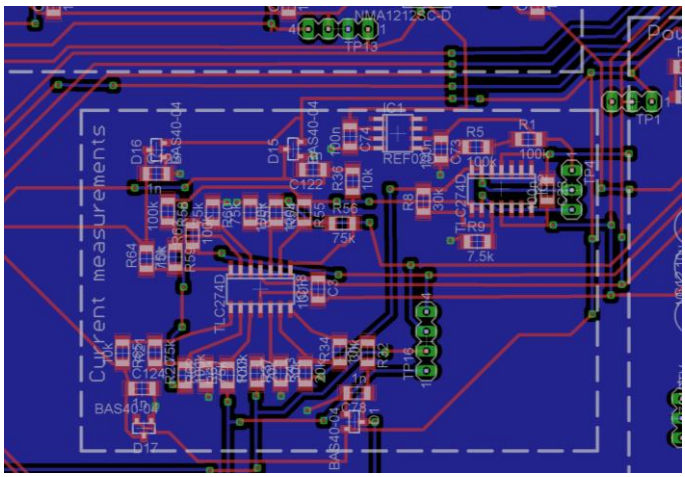


FIGURE 2.29: THE CURRENT MEASUREMENT SIGNAL SCALING

The rest of the power supplies are placed in the bottom right corner of figure Figure 2.22, a clip given in Figure 2.30. These are the +5V_GND2, +3V3_GND2 and +/-15V_GND2.

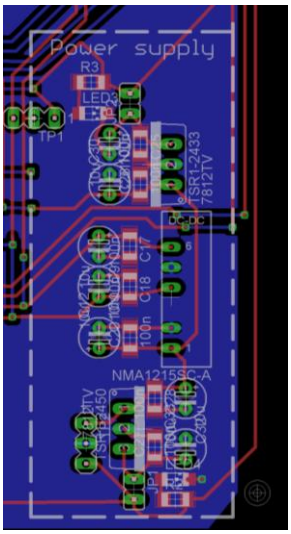


FIGURE 2.30: THE +5V_GND2, +/-15V_GND2 AND +3V3_GND2 POWER SUPPLIES

2.4 SOFTWARE FOR DRIVING THE CONVERTER

To summarize what is explained earlier, the microcontroller has two main tasks: Generation of 3 independent pairs of PWM signals and do an analog-to-digital conversion (ADC) of the voltage and current measurements. This is implemented with real-time programming with the C programming language in Code Compose Studio™ v4 Core Edition. The project folder is called “converter” and the code is given in appendix C. When the exact purpose of the converter is decided, this code must be tailored for that, and any additional code added.

The theory behind the software design is obtained from the TMS320x2806x Piccolo Technical Reference Manual (Texas Instruments A, 2011) and the lectures in ELK-21 Electronic for Control of Power taught at NTNU by Frederick Ishengoma fall 2011 (Ishengoma, 2011). The setup of the ADC, PWM and main file is based on the course material in that course.

Both the PWM signals and the ADC are controlled by the same clock; the PWM time-base counter. It counts up and down, and the PWM channels use it as their triangular waveform. Hence, the switching frequency is set by this clock. At every zero point, the lowest point, the start-of-conversion (SOC) of the signals connected to the ADC pins is triggered. This is also called the sampling of the signals. The sampling frequency is thus the same as the switching frequency and set by this clock. This is shown in the figure under.

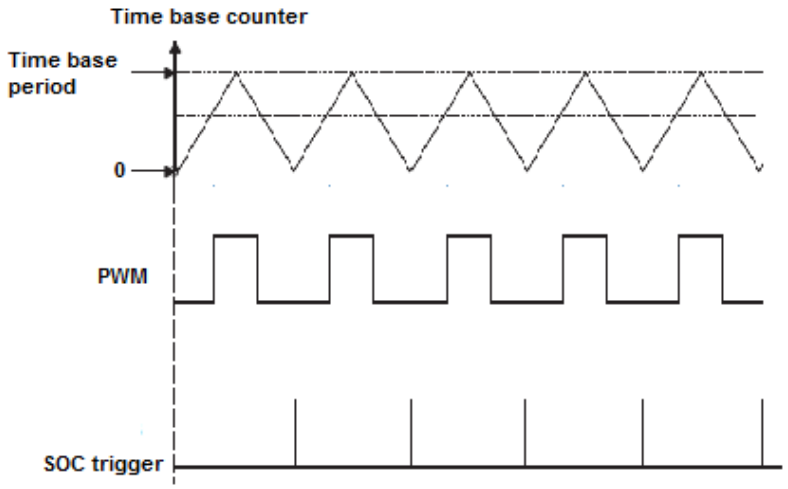


FIGURE 2.31: THE PRINCIPLE BEHIND THE PWM AND ADC IN THE CODE

It is important that the ADC is triggered at the zero point of the clock, since the sampling then will happen in the middle of the switching period, where the current and voltage ripple is equal to its average value over the switching period. This is illustrated in Figure 2.6. There is hence no need for additional sampling within one switch period to find the average value.

The end of conversion (EOC) of the last converted channel triggers the ADC interrupt. The interrupt service routine (ISR) reads and stores the conversion results. An interrupt causes a pause in the program to run its interrupt service routine, and goes back to the program when finished.

To summarize, a block diagram of this is given in Figure 2.32. The TBCTR and TBPRD are the bit names for respectively the time-base counter and the time-base period. The latter is the bit where the frequency is set. CMPA is the $v_{control}$ signal.

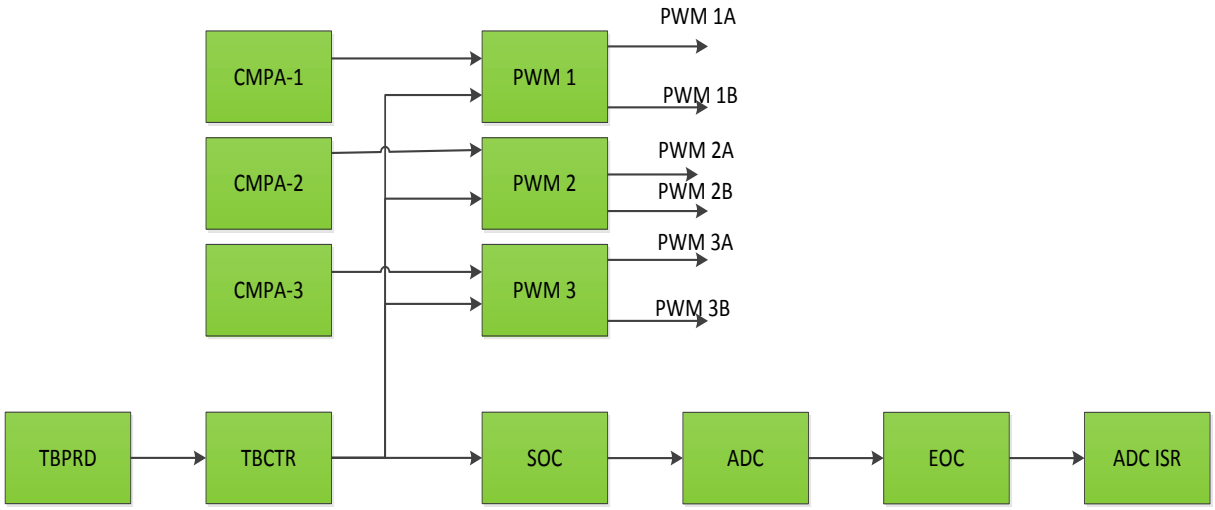


FIGURE 2.32: OVERVIEW OF THE PWM AND ADCSIGNALS

The way this is done, the sampling frequency and switching frequency are equal. However, the *usage* of the conversion results does not have to be done at the same frequency, i.e. in motor control for instance.

The program is organized from the main file and the main function; hence the more detailed description of the code is explained with the basic of the main file. Excerpts of some of the code are given along with the explanations, but all is given in appendix C.

2.4.1 MAIN FILE

The main file (converter_main-F2806x_1.c) of the code is organized as shown in the block diagram below. First, the parameters and functions are initialized. Then, the main function is called. It starts by calling the functions DeviceInit(), InitPWMs(), InitADC() which respectively initializes the different devices used and the PWM and ADC modules, by their registers. These functions' headers are written in separate files, to make the code tidier, and are explained in more detail later. The next step is the enabling of the interrupts. Finally, an endless loop is called to make the code run as long as wanted, and do as many interrupts as desired. The code of the main function is written in the text box, Figure 2.34.

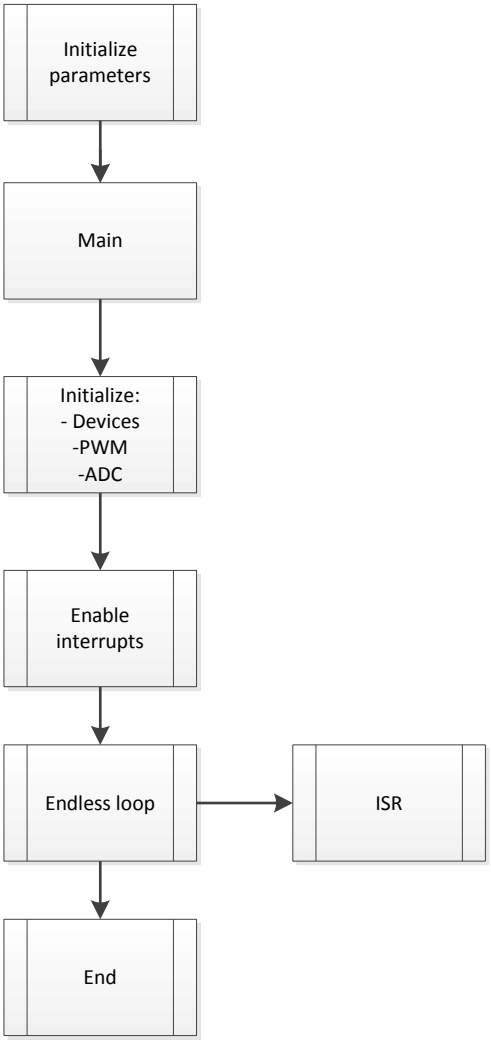


FIGURE 2.33: BLOCK DIAGRAM OF MAIN FUNCTION

```
void main(void)
{

// Initialize System Control
// Enable Peripheral Clocks, GPIO
// Initialize the PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
 DeviceInit();
 InitEPWMs();//Initialize PWM modules
 InitADC();         // Initialize ADC


 //Enable the interrupt
 EALLOW;
       PieVectTable.ADCINT1=&ADCINT1_ISR;
 EDIS;

 PieCtrlRegs.PIEIER1.bit.INTx1=1;
 IER |= M_INT1;
 EINT;

  for(;;)  //infinite loop
  {
    asm(" NOP");
    if (EndlessLoopCounter++ >= 4294967295)
    {
        EndlessLoopCounter=0;
    }
  }

}// end of main
```

FIGURE 2.34: THE MAIN FUNCTION

The following sub-sections will describe the initialization of the devices, PWM signals and ADC modules and the interrupt-service-routine.

2.4.2 DEVICE INITIALIZATION

The device initialization (DeviceInit()) function initializes the devices specific for this application. Its header is given in the file converter_DevInit_F2806x.c. The function includes enabling the peripheral clocks for the ADC and the first three ePWM modules and the system time-base clock. The clocks are explained under their respective sub-sections. It configures the 6 eEPWM pins. The configuration commands are given in the text box and the pins chosen for PWM and ADC are given in the table below. The GPIO-18 pin is set to give a high output signal constantly. This is to supply the RESET- pin of the MOSFET driver. It can later be turned

off as wanted either within the interrupt or manually in the watch window of Code Composer Studio™. The statements used are shown in Figure 2.35. The pins chosen and their function is shown in Table 2.2.

```
SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;          // ADC
SysCtrlRegs.PCLKCR1.bit.EPWM1ENCLK = 1;        // ePWM1
SysCtrlRegs.PCLKCR1.bit.EPWM2ENCLK = 1;        // ePWM2
SysCtrlRegs.PCLKCR1.bit.EPWM3ENCLK = 1;        // ePWM3
SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;         // Enable TBCLK
GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;            // 1=EPWM1A
GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1;            // 1=EPWM1B,
GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1;            // 1=EPWM2A
GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 1;            // 1=EPWM2B
GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1;            // 1=EPWM3A
GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 1;            // 1=EPWM3B
GpioCtrlRegs.GPAMUX2.bit.GPIO18 = 0;           // 0=GPIO
GpioDataRegs.GPASET.bit.GPIO18 = 1;            // Set High initially
```

FIGURE 2.35: EXCERPTS OF THE DEVICE INITIALIZATION CODE FOR PIN AND CLOCK CONFIGURATION

TABLE 2.2: PIN CONFIGURATION

SIGNAL	PIN
PWM-1A	GIPO-00
PWM-1B	GIPO-01
PWM-2A	GIPO-02
PWM-2B	GIPO-03
PWM-3A	GIPO-04
PWM-3B	GIPO-05
Input current	ADC-A0
Output current A	ADC-A1
Output current B	ADC-A2
Output current C	ADC-A3
Input voltage	ADC-B0
Output voltage A	ADC-B1
Output voltage B	ADC-B2
Output voltage C	ADC-B3
RESET-	GPIO-18 (configured to be set high initially)

If additional functions are implemented in the code later, one must remember to specify the pins and enable the module clocks within this function.

2.4.3 PWM Signal Initialization

The PWM modules are initialized in the InitEPWMs() function given in converter_PWM_Setup.c file. Three ePWM channels are used; ePWM1, ePWM2 and ePWM3, controlling each converter bridge. This file initializes the registers for all three. Each channel has two ePWM outputs, A and B, which are controlling the two MOSFETs in one bridge-leg. When one turns on, the other one turns off and vice versa. All three channels are using an equal triangular waveform signal, a PWM time base counter (TBCTR) and one control signal (CMPA) each to make the PWM signal. Hence, the three bridge-legs are independent of each other while having the same frequency. The PWM-A signal is set to low (AQ_CLEAR) when the time-base counter equals CMPA on down-count (CAD) and set to high (AQ_SET) when they are equal on the time base counter's up-count (CAU). The opposite is set for the B signal. The textbox under shows the implementation of this in the code.

```
// PWM1A low (AQ_CLEAR) when CMPA = TBCNT on down count (CAD)
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
// PWM1A high (AQ_SET) when CMPA = TBCNT on up count (CAU)
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
// PWM1B high (AQ_SET) when CMPA = TBCNT on down count (CAD)
EPwm1Regs.AQCTLB.bit.CAD = AQ_SET;
// PWM1B low (AQ_CLEAR) when CMPA = TBCNT on up count (CAU)
EPwm1Regs.AQCTLB.bit.CAU = AQ_CLEAR;
```

Figure 2.36: Excerpts of the PWM setup code of how to configure the PWM signals

The frequency f_{PWM} of the time-base counter, the switching frequency, is set by the time-base-period bit, EPwmXRegs.TBPRD, and the system clock frequency of the microcontroller, f_{TBCLK}, of 80 MHz.

$$f_{PWM} = \frac{f_{TBCLK}}{2 * TBPRD} \quad (2.16)$$

(Texas Instruments A, 2011)

For a PWM frequency, f_{PWM}, of 50 kHz, the TBPRD becomes 800. The switching frequency can easily be varied by selecting another value for TBPRD.

The duty ratio is calculated from the EPwmXRegs.CMPA.half.CMPA value by the equation

$$D = 1 - \frac{CMPA}{TBPRD} \quad (2.17)$$

(Texas Instruments A, 2011)

The initial CMPA value is set to TBPRD/2 within this file which gives a duty ratio of 50 % for all PWM signals. While the program is running, the CMPA value can be changed within the interrupt to fit the desired output values.

The blanking-time is set by the EPwmXRegs.DBFED (forward edge delay) and EPwmXRegs.DCRED (reverse edge delay) bits to 200 time base periods for both rising and falling edge of the signal. One time base period is one period of the system clock of 80 MHz. The blanking time is thus 200/80 MHz = 2.5 µs. Recalling the MOSFET's turn on and off time of 0.1 and 0.15 µs, this is high enough. It should be adjusted when the system is up and running.

ePWM1, ADCSOCA will be used to trigger the analog to digital conversion (ADC) once every period of the triangular waveform, 50 kHz, at its zero crossing by the statement EPwmXRegs.ETSEL.bit.SOCASEL = ET_CTR_ZERO.

To summarize, Figure 2.37 shows the different waveforms, values and signals for one ePWM channel. The three main parameters of the PWM modules are

- The switching frequency, which is determined by TBPRD
- The duty ratio, set by CMPA, sets on the desired value of the converter's output.
- The blanking time, set by DBFED and DBRED, will depend on the MOSFETs and must be fit for the ones chosen.

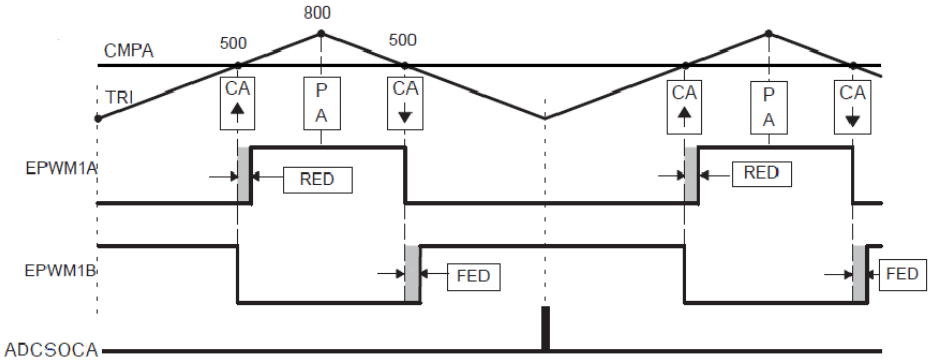


FIGURE 2.37: TBCTR, CMPA, PWM, ADCSOCA, BLANKING TIME(TEXAS INSTRUMENTS A, 2011)

2.4.4 ADC INITIALIZATION

The initialization of the ADC setup is done in InitADC() which header is given in the converer_Adc.c file. The eight channels of conversions are specified in this function. They are all set to be triggered by ePWM1, ADCSOCA, by the statement AdcRegs.ADCSOCYCTL.bit.TRIGSEL = 5. Which pin each channel is going to convert is set by the AdcRegs.ADCSOCYCTL.bit.CHSEL bit.

The ADC ISR is triggered by the end-of-conversion (EOC) of channel 7, which is the last one. It is determined that the flag bit must be cleared in

order to start another ISR. The flag bit is cleared within the ADC interrupt service routine.

The conversion results are given as digital numbers of 12 bit. Equation 2.9 gives the ratio of the input signal.

2.4.5 ADC Interrupt Service Routine

As explained above, the ADC interrupt service routine (ISR) is what is done after the signals of the 8 channels have been converted. A block diagram of the ADC ISR is given in Figure 2.38. The conversion results are stored in a circular array with room for 50 values of each conversion. The term circular array referrers to the overwriting of the oldest values as the array becomes full. Equations given in sub-section 2.2.4 and equation 2.9 are used to get the real current and voltage values from the digital numbers. This is not calculated within the ISR, to save time. The values can be processed later in Matlab for instance.

The microcontroller has a limit of storing around 2000 values, making 50 per channel OK, and allowing a maximum of 250. This can be rearranged in order to get more values for one parameter.

In the end, the PIE group is acknowledged and the ADC int flag bit cleared to enable further interrupts. A counter counts the number of ISRs. This is done for debugging such that it easily can be seen if the ISR is done.

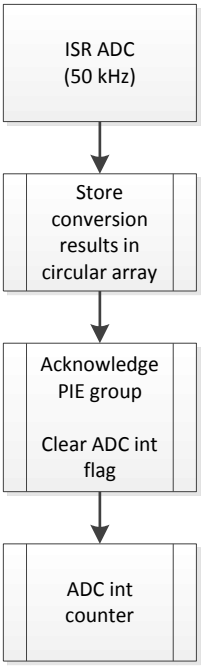


FIGURE 2.38: BLOCK DIAGRAM OF THE ADC INTERRUPT SERVICE ROUTINE

3 A CASE STUDY: DC MOTOR DRIVE

As written earlier, the switch-mode power electronic converter has a wide range of usage. These kinds of converters are often utilized in power processing units because of their high energy efficiency and low cost, size, and weight (Mohan, 2003). The full-bridge converter and three-phase inverter are the two most common. These can operate in all four quadrants of the current-voltage plane and can hence be used as a dc-to-ac inverter or ac-to-dc rectifier or just dc-dc operation (Mohan, et al., 2003).

The full-bridge converter is mostly used in dc motor drives, dc-ac conversion in single-phase uninterruptible ac power supplies, or dc-ac high intermediate frequency conversion in switch-mode transformer-isolated dc power supplies (Mohan, et al., 2003). The three-phase dc-ac inverters are generally used in ac motor drives and uninterruptible ac power supplies (Mohan, et al., 2003). When used in such power processing units (PPUs), the converter's efficiency can exceed 95 %, and even exceed 98 % for large power ratings (Mohan, 2003).

This second part of this study, explains the utilizing of the converter in a DC motor drive for speed control. First, some basic theory about the DC motor, DC motor control and speed measurement is given. It is explained what parameters that needs to be controlled in order to vary the speed of the DC motor, how the control is done and structured and how to measure the actual rotational speed of the DC motor. Then the system as a whole is described, i.e. how to connect the different parts together to control the speed of the motor. Finally, the software tailored for this operation is explained.

3.1 THE DC MOTOR

The theory behind this section is obtained from (Chapman, 2005) and (Mohan, 2003).

A fundamental figure of the DC machine is displayed in Figure 3.1. The stator generates a magnetic field in which the rotor rotates. The rotor consists of one or more current coils, generating a magnetic field when voltage is applied. The magnetic field from the rotor coils opposes the magnetic field from the stator and causes the rotor to rotate. When the fields are pointing in the same direction, the rotor current direction is altered. Such will the stator field continue to cause the rotor to rotate. The voltage source is connected to the rotor through brushes and conductors while it rotates. The connection between the stationary brushes on the supply side and the conductors at the rotor side alters every half rotation to change the rotor current direction.

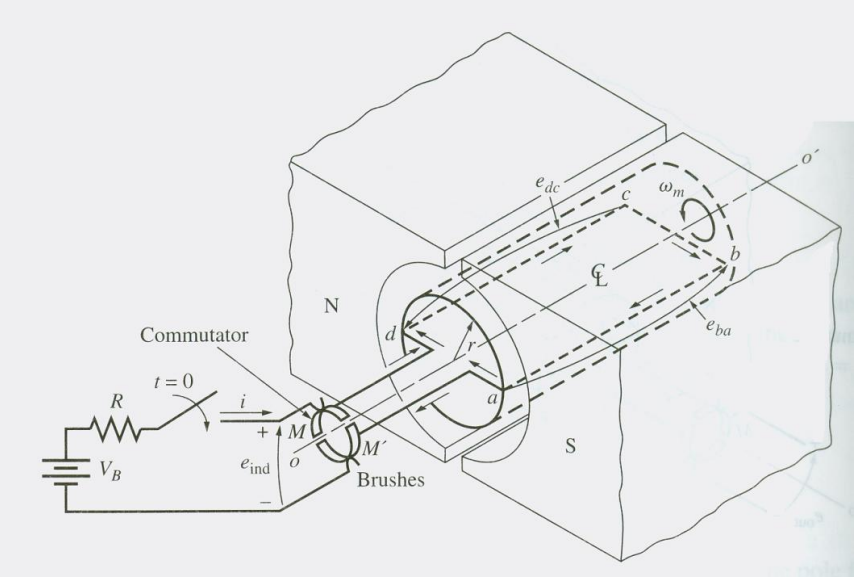


FIGURE 3.1: FUNDAMENTALS OF THE DC MACHINE(CHAPMAN, 2005)

The equivalent circuit of such a separately exited DC machine is shown in the Figure 3.2 below. The left hand side circuit represents the field circuit which generates the magnetic field in the stator of the machine. This requires a separate power source, hence the name separately excited. R_F

represents the resistance and L_F represents the inductance in those windings. For a permanent magnet (PM) DC machine, the stator field is made by permanent magnets, and is thus constant.

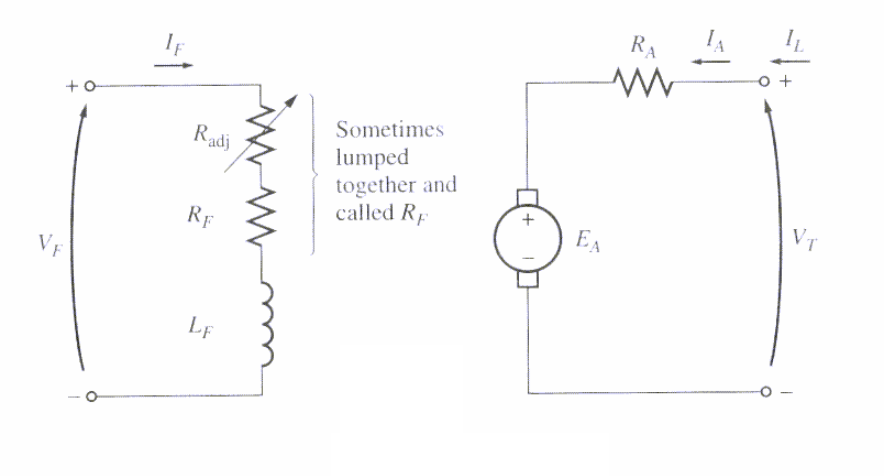


FIGURE 3.2: CIRCUIT DIAGRAM OF SEPARATELY EXITED DC MOTOR(CHAPMAN, 2005)

The right hand side of the circuit diagram represents the rotor. $V_T$is the applied rotor voltage and V_A is the resistance in the windings. The back emf, E_A, is the induced voltage in the rotor from the stator field when the rotor is rotating. The applied voltage V_A is given as

$$V_T = E_A + I_A R_A \tag{3.1}$$

Where E_A is given as

$$E_A = K_E \phi \omega \tag{3.2}$$

K_E is a constant depending on the machine's characteristics, ϕ is the flux from the stator field, and ω is the rotational speed given in radians per second.

The DC machine can operate in all four quadrants of the current-voltage plane. The applied voltage polarity determines the direction of the rotation and the current direction of the determines whether it is operating as a motor

or generator. In this report, the motor operation is considered and the current is thus only flowing *into* the machine.

The torque of the motor is given as

$$\tau_{ind} = K_T \varphi I_A \tag{3.3}$$

By inserting equation 3.2 and 3.3 into 3.1, one can see that the rotational speed ω is given as

$$\omega = \frac{V_T}{K\varphi} - \frac{R_A}{K_T K_E \varphi^2} \tau_{ind} \tag{3.4}$$

The speed is controlled by varying the applied voltage V_T.

The magnetic field is supplied by an external voltage source which in this study is considered a constant power source or a permanent magnet which cannot be varied.

The induced torque depends on the load and the change in speed, according to equation 3.5.

This shows that during constant speed, the induced torque must be equal to the load torque. While during an acceleration or deceleration of the speed, the induced torque must be respectively higher or lower than the load torque. The machine will thus draw the needed current from the DC source to either rotate the load during steady state, or to accelerate or decelerate the load during speed increase or decrease.

$$\frac{d\omega}{dt} = \frac{1}{J_{eq}} (\tau_{ind} - \tau_L) \tag{3.5}$$

To avoid over-current condition in the process of controlling the speed by the applied voltage V_T, it is required to control the current drawn by the

motor. The following sections will describe how one can control both the input voltage and current to the motor by means of an electric drive.

The motor constants K_E and K_T are given as

$$K_E = K_T = \frac{ZP}{2\pi a} \quad (3.6)$$

(Chapman, 2005)

Where Z is the total number of conductors in the rotor, a is the number of current paths and P is the number of pole pairs.

3.2 THE CONTROL THEORY OF THE DC MACHINE

The theory described here is obtained from (Mohan, 2003) and (Wescott, 2000).

3.2.1 PI CONTROLLERS

For the control of a DC motor, the proportional-integral (PI) controller is sufficient (Mohan, 2003). A block diagram of a general PI controller controlling a plant is shown in the figure below. The plant in this concern includes the motor and drive.

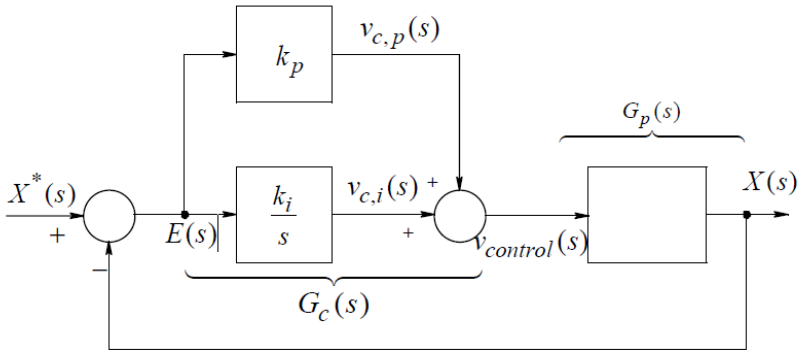


FIGURE 3.3: PI CONTROLLER (MOHAN, 2003)

This is a feedback system, where the controller's input $E(s)$ is the error between the system's input and output.

The proportional controller's output is the error multiplied by the proportional gain k_p. This is the simplest way to control a motor. For low gains, the system slowly reaches the desired point of operation. For higher gains the system does so faster, but by increasing the gain there will be more and more oscillations before the desired point is reached. For too high gains the system may get unstable and oscillate around the desired output. For systems with too much delay, the proportional controller will cause

oscillations even for low gains. With too low gain, this system will not reach the desired output with only the proportional controller, and a long term error will occur.

The integral controller summarizes all errors and multiplies the sum to the integral gain, k_i. The integral controller hence "remembers" what it has gone through before and will therefore help to cancel long term errors. However, the integral controller alone is too slow and hence the system will easily get unstable with only the integral controller implemented. Since the integral controller summarizes the error over time, the sum can get too high and drive the system far away from the target. To prevent this, a maximum and minimum value of the integrator must be set. This is called anti-windup.

The sum of the proportional and integral controller's output is the input to the drive of the motor. This will regulate the control signal to the motor such that the error between the desired and actual value eventually becomes zero.

The ratio between the input and output to a system is defined as the system's transfer function. The general transfer function of the PI controller is in the s plane written as

$$\frac{V_c(s)}{E(s)} = k_p + \frac{k_i}{s} = k_p\left(1 + \frac{k_i}{sk_p}\right) = k_p\left(1 + \frac{1}{sT_i}\right) \quad 3.7$$

In time domain it is given as

$$v_c(t) = k_p(e(t) + \frac{1}{T_i}\int_{-\infty}^{t} e(\tau)d\tau) \quad 3.8$$

(Balchen, et al., 1999)

To implement the controller digitally, to be able to use the microcontroller for the control, the transfer function must be made discrete, as by equation 3.9. *k* is the sample number k=0,1,2… and T_s is the sampling period.

$$v_c(k) = k_p\left(e(k) + \frac{1}{T_i}\sum e(k)T_s\right) \text{ (Balchen, et al., 1999)} \quad 3.9$$

To find the proportional and integral gains, k_p and k_i, of the controller the transfer function for the whole system must be considered.

The open-loop transfer function $G_{OL}(s)$ of a system is disregarded the feedback loop. For the system shown in Figure 3.3 above it is given as

$$G_{OL}(s) = G_c(s)G_p(s) \tag{3.10}$$

(Mohan, 2003)

$G_c(s)$ is the controller's transfer function and $G_p(s)$ is the plant's transfer function. The closed-loop transfer function $G_{CL}(s)$ for a unity feedback system is given as

$$G_{CL}(s) = \frac{G_{OL}(s)}{1 + G_{OL}(s)} \tag{3.11}$$

(Mohan, 2003)

The open-loop frequency-response is shown in the figures under. Here, the cross-over frequency ω_c and the phase margin are marked. These are two terms that categorize the system which can be used to find the gain constants of the controllers.

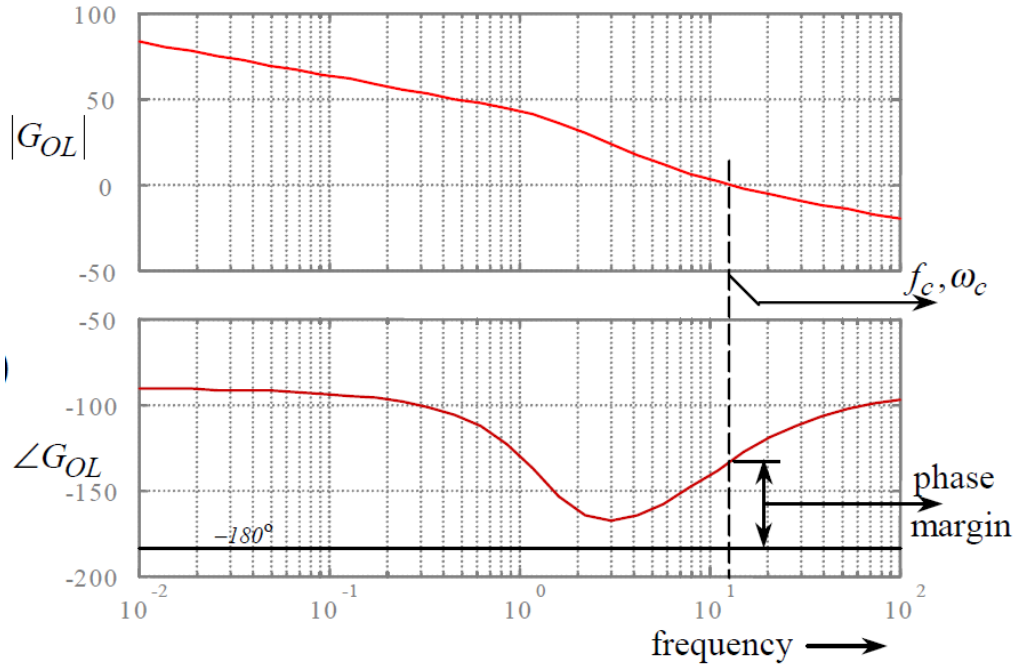


FIGURE 3.4: FREQUENCY RESPONSE OF THE OPEN-LOOP TRANSFER-FUNCTION (MOHAN, 2003)

The cross-over frequency ω_c is defined as the frequency where the gain (magnitude) equals unity, i. e. $|G_{OL}(s)| = 0dB$. In most systems, the cross-over frequency is equal to the closed-loop bandwidth which is proportional to the response time.

At the cross-over frequency the phase delay of the open-loop transfer-function must be less than 180° for the closed-loop feedback-system to be stable. The phase margin is defined as the phase angle of the open-loop transfer-function, measured with respect to -180°.

$$Phase\ margin = \varphi_{OL}|_{f_c} - (-180°) \qquad 3.12$$

(Mohan, 2003)

For a stable system, the phase margin $\varphi_{pm,\Omega}$ should be greater than 45°, close to 60°.

3.2.2 THE CASCADE CONTROL STRUCTURE

For full DC motor control, three different controllers are needed: The torque (current), speed and position controller. These are organized into the cascade control structure which is of common usage for the DC motor control because of the high flexibility.

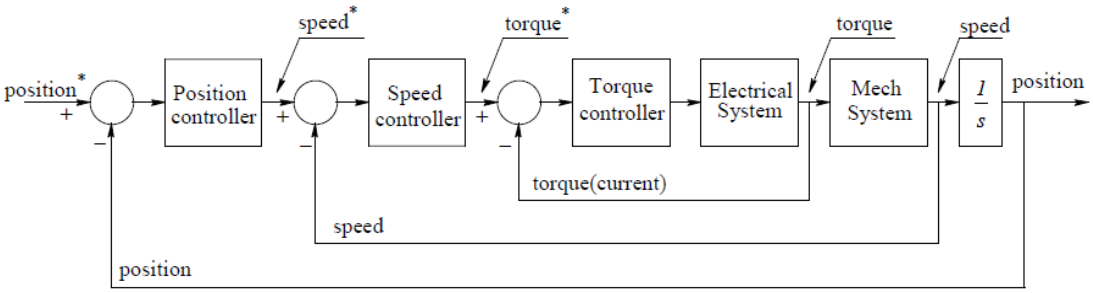


FIGURE 3.5: CASCADE CONTROL STRUCTURE FOR DC MOTOR CONTROL(MOHAN, 2003)

The innermost loop represents the torque control and the electrical system, here consisting of a DC machine and the drive which is its power processing unit. Since the current is proportional to the torque, this loop controls the current too. The second loop is the speed-control loop together with a representation of the mechanical system. The outermost loop is the position control loop which is not considered in this report.

Each loop is having a bandwidth (crossover frequency) of one order of magnitude lower than the loop within. The response time is therefore shortest for the torque loop and longest for the position loop.

The first step in the design is to assume operation around the steady-state point where the system is assumed to be linear. Then, if it is not, the controller must be adjusted as appropriate.

When finding the gain constants of each control loop, one has to start with the torque/current control loop. Its crossover frequency should be one or two orders of magnitude smaller than the switching frequency of the motor control (Mohan, 2003).

The block diagram of the torque loop is given in the figure under. The first block is the PI controller with the current error as its input. The drive is represented by a proportional controller, k_{PWM}. The rest of the system represents the DC motor, from equation 3.1, 3.2 and 3.3.

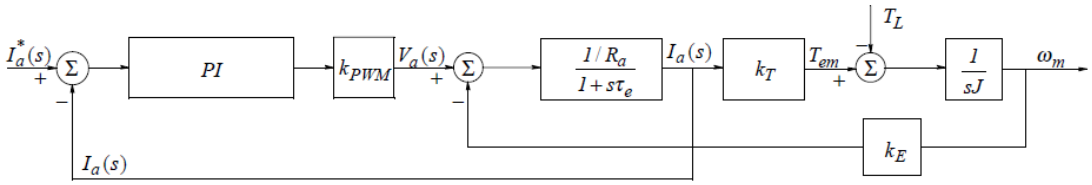


FIGURE 3.6: TORQUE CONTROL LOOP(MOHAN, 2003)

To find its transfer function, the block diagram should first be simplified. Noticing the current and torque being proportional to each other according to equation 3.3, and lumping together the motor blocks into one, the block diagram now looks like:

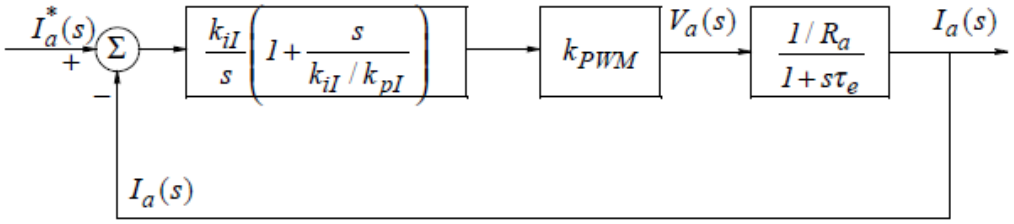


FIGURE 3.7: SIMPLIFIED TORQUE CONTROL LOOP(MOHAN, 2003)

The subscript *i* indicates current/torque control. The open-loop transfer function is thus

$$G_{I,OL} = \frac{k_{iI}}{s}\left[1+\frac{s}{\frac{k_{iI}}{k_{pI}}}\right] k_{PWM} \frac{1/R_a}{1+\frac{s}{1/\tau_e}} \qquad 3.13$$

The gains are selected by setting the zero (k_{iI}/k_{pI}) of the PI controller to cancel the motor pole at $(1/\tau_{e)}$.

$$k_{pI} = \tau_e k_{iI} \tag{3.14}$$

(Mohan, 2003)

Under these conditions, the transfer function is simplified into

$$G_{I,OL}(s) = \frac{k_{iI} k_{PMW}}{s R_a} \tag{3.15}$$

(Mohan, 2003)

At the selected cross over frequency,

$$k_{iI} = \frac{\omega_{cl} R_a}{k_{PWM}} \tag{3.16}$$

(Mohan, 2003)

When considering the speed loop, the crossover frequency and bandwidth is set to one order of magnitude smaller than that of the torque loop. The torque-control loop can therefore be set to 1, while considering the speed loop. This part of the cascade structure is shown in the figure under. The last block represents the mechanical system.

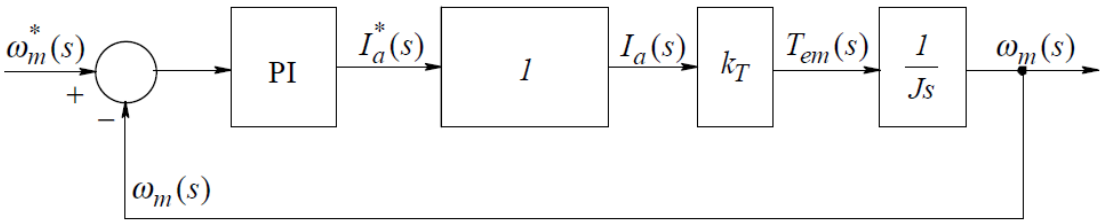


FIGURE 3.8: SPEED CONTROL LOOP(MOHAN, 2003)

This gives an open loop transfer function of

$$G_{\Omega,OL}(s) = \frac{k_{i\Omega}}{s}\left[1 + \frac{s}{\frac{k_{i\Omega}}{k_{p\Omega}}}\right]\frac{k_T}{sJ_{eq}} = \frac{k_{i\Omega} k_T}{J_{eq}}\frac{[1+s/(k_{i\Omega}/k_{p\Omega})]}{s^2} \tag{3.17}$$

The subscript Ω stands for speed and J_{eq} is the total effective inertia for the motor and load.

This transfer function has a double pole at the origin. At the crossover frequency, the magnitude of the transfer function equals 1 and the angle is the phase margin subtracted by 180°.

$$\frac{k_{i\Omega}k_T}{J_{eq}}\frac{\left[1+s/(k_{i\Omega}/k_{p\Omega})\right]}{s^2}=1 \tag{3.18}$$

$$\angle\frac{k_{i\Omega}k_T}{J_{eq}}\frac{\left[1+s/(k_{i\Omega}/k_{p\Omega})\right]}{s^2}=-180°+\varphi_{pm,\Omega} \tag{3.19}$$

The gain constants of the speed controller $k_{i\Omega}$ and $k_{p\Omega}$ can now be found by solving 3.18 and 3.19.

An easier way to determine the gain constants is by trying and failing systematically, as explained by Wescott (2000). First, both gains are set to zero. Then the proportional gain start value is chosen between 1 and 100. If the system is oscillating, decrease the gain with a factor of 10, of it is not oscillating, increase it with the same factor. The closer one gets to the desired value, the smaller rate of change for the gain constant should be used. Stop when the system responds well to the change with a fast response with not too much overshoot. Then the integral gain should be tuned. Start with a value between 0.0001 and 0.01. Change it until the response is fairly fast without too much oscillation.

SAMPLING TIME

The sampling time should initially be set to between 1/100th and 1/10th of the desired settling time (Wescott, 2000). It can be beneficial to keep the sampling frequency as a variable and have the opportunity to increase it. In the code, the sampling time should hence be made to a controllable variable.

Since this system does not have a differential controller, the sampling frequency can be decreased a bit. The sampling frequency should never be set lower than 5 times the desired settling time.

3.3 SPEED MEASUREMENT

The instantaneous rotational speed ω of the motor is measured by a quadrature encoder of the type HEDS-5540. It is mounted directly on the motor shaft and sends pulses to the microcontroller which it can use to calculate the speed. The encoder is explained here by use of the information from the QE's datasheet (Hewlett Packard, u.d.) and Texas Instruments A (2011).

It is made of a disk with 1024 slots placed evenly around the edge, Θ degrees apart. A light source is set on one side, and two photo sensors, A and B, placed $\Theta/4$ degrees apart on the other side are sensing every time a slot pass and use this to generate the quadrature encoder pulse (QEP) signals, QEP-A and QEP-B. They are shown in Figure 3.9, together with the disk, light and photo sensors.

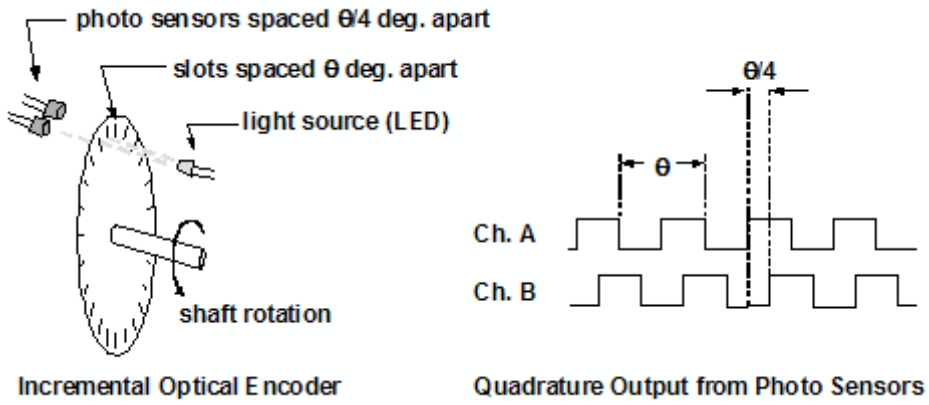


FIGURE 3.9: THE PRINCIPLE BEHIND THE QUADRATURE ENCODER (TEXAS INSTRUMENTS, 2010)

High rotational speeds are calculated by counting the number of pulses passed during a fixed time interval. For low rotational speeds the time it takes for one pulse to pass is measured. Hence, the encoder can measure a wide range of speeds.

One of the QEP signals will lead the other by 90°, and this is used to find the rotational direction.

According to its datasheet (appendix F), the quadrature encoder needs a voltage suppliy of 0-7 V. It is supplied by the +3V3_GND2 on the PCB, hence its output signals are within the microcontroller's range of 0-3.3 V. Since it is not connected to the power circuit, only the axis of the motor, its output signals can directly be connected to the microcontroller unit without isolation.

3.4 Utilizing the Converter in the DC Motor Drive

To use the converter in a DC motor drive only two of the bridge-legs are needed. The motor is connected between the outputs of the two, which configures a full-bridge converter. The input voltage to the motor can now be varied and hence its rotational speed and input current controlled. The circuit diagram of the connections is shown in the figure under.

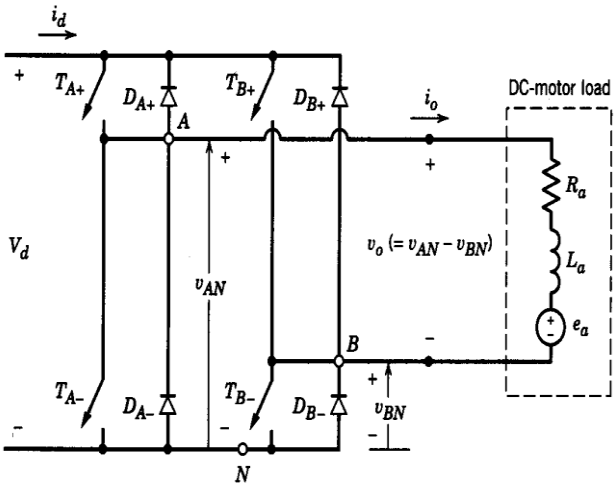


Figure 3.10: A full-bridge DC-DC converter controlling a DC motor(Mohan, et al., 2003)

With bipolar voltage switching, the full-bridge converter can be used to drive the motor in all 4 quadrants. However, only the two quadrants with positive current flow into the machine will be used in this study. The polarity of the motor input voltage determines the rotational direction.

The PI controllers must be implemented within the microcontroller's code. The microcontroller gets the set-speed input and calculates the actual motor speed by the QEP signals from the encoder. The error between these two is the input to the speed PI controller. Its output is the set-current. The set-current minus the measured current, is the input to the current PI controller which controls the PWM duty ratio. The PWM signals are fed into the drive

which controls the motor. The PWM duty ratio is adjusted in such a way that the actual speed will meet the set speed, at the same time as undesirably high current transients are avoided.

The switching frequency is initially set to 50 kHz. The current-loop crossover-frequency has to be one order of magnitude lower, 5 kHz. The speed-loop crossover-frequency is set to one order of magnitude lower than the current loop's, at 100 Hz. The controller gains are initially set to 0 and the value should be found, as described above, when the system is up and running.

The system block diagram is given in Figure 3.11.

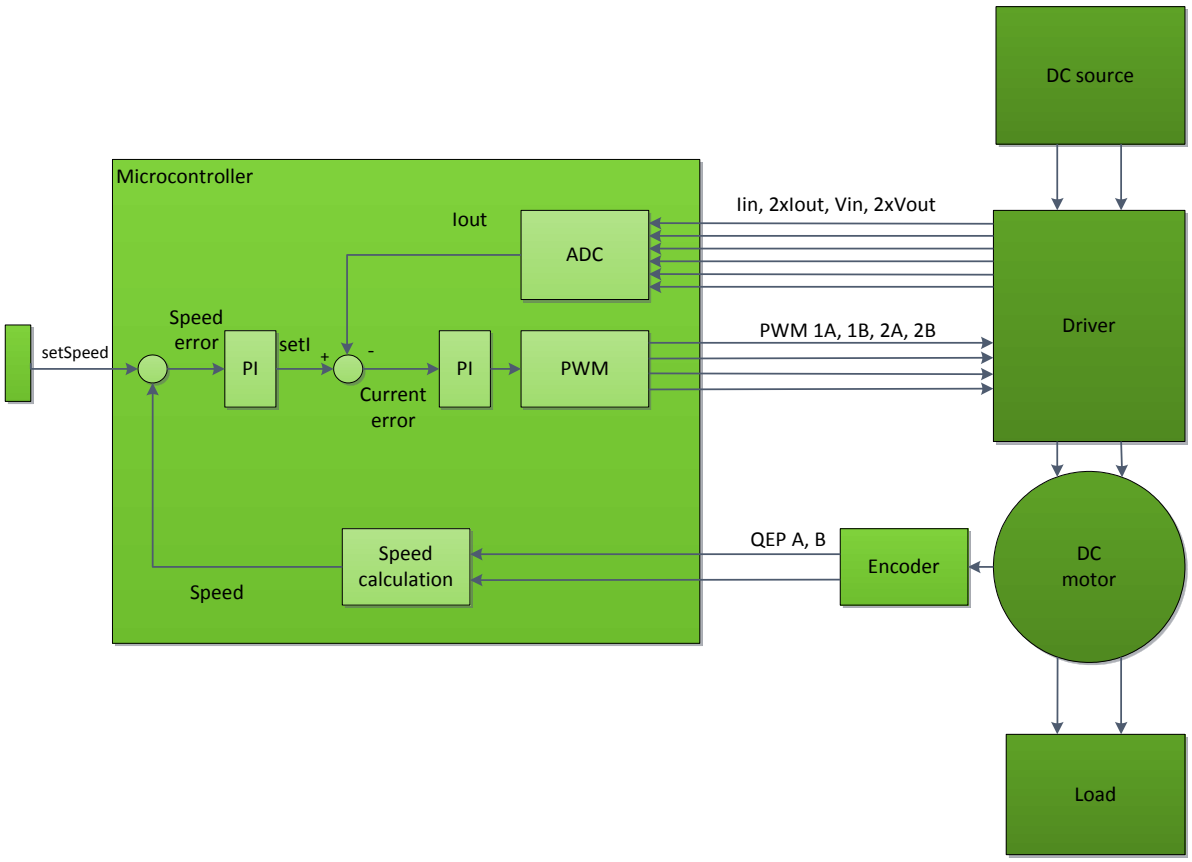


FIGURE 3.11: BLOCK DIAGRAM OF THE SYSTEM

3.5 SOFTWARE IMPLEMENTATION FOR DC MOTOR DRIVE

For specific task of DC motor control, the code presented in section 2.4, with some modifications, is used as a foundation and additional code is added to do the speed measurement and control. The controller algorithm is based on the examples given by Wescott (2000). The code will be explained in detail as in section 2.4, but first comes a short description to give an overview.

The switching frequency, the frequency of the PWM signals, is initially set to 50 kHz. The code is written such that this can be varied by only one parameter, TBCTR. The current loop is implemented within the ADC ISR. Because the ADC ISR is called at the switching frequency, the current loop is placed within an if-loop which is only called every 10^{th} ADC ISR. This makes the frequency of the current loop one order of magnitude lower than the switching frequency, i.e. 5 kHz for a switching frequency of 50 kHz.

A setup for the QEP input is made and the speed is measured by the quadrature encoder pulse (QEP) and calculated within the QEP ISR. The speed loop is also implemented in the QEP ISR which is triggered at 100 Hz, one order of magnitude lower than the current loop.

What is added to the existing code is the settings for the QEP and QEP ISR, as well as some changes in the DeviceInit() and main(). The PWM-1 and PWM-2 modules are set to bipolar voltage switching, and the PWM-3 module is disabled. The code will now be explained in more detail. The most important parameters are the switching frequency, the current-loop and speed-loop bandwidth, the controller's gains and the maximum values for the current controller.

The code is included in appendix D.

3.5.1 CURRENT LOOP

The current loop, implemented within the ADC ISR, is shown in the textbox below. The code is based on Wescott (2000) and the equations presented in section 0. The sampling period is given as indexB times EPwm1Regs.TBPRD, such that the code will follow any changes done to the sampling/switching frequency automatically. When indexB is 10, the execution of the current loop will be one order of magnitude lower than the switching frequency.

The rest of the code is self-explained from the comments and equations in section 3.2.

```
//Current/torque loop
      if (indexB==10)
//To make the current loop calculate for every 10th
interrupt, making the sampling time for the current loop on
order of magnitude lower than the switching frequency
      {
      //calculates the output of the current control loop
            currentError=setCurrent-AdcResult.ADCRESULT2;
      //calculates the current error
            pTermI=kpI*currentError;
            //proportional term
            iStateI+=currentError;
            //sums up the current errors
            if(iStateI>iStateIMax) iStateI=iStateIMax;
      //anti-windup
            if(iStateI<iStateIMin) iStateI=iStateIMin;
            iTermI=kiI*iStateI*EPwm1Regs.TBPRD/4000000;
      //integral term: sampling period =
      //10*EPwm1Regs.TBPRD*2/80MHz
            currentController=pTermI+iTermI;
      //current controller output


      //calculate the new duty ratio
            EPwm1Regs.CMPA.half.CMPA -= currentController;

      EPwm2Regs.CMPA.half.CMPA=EPwm1Regs.CMPA.half.CMPA;

            indexB=1;
      }
```

FIGURE 3.12: CURRENT/TORQUE CONTROL LOOP

3.5.2 QEP INITIALIZATION AND SPEED CALCULATION

The initialization of the QEP peripherals, is done in the InitEQEP() function called by the main() function. Its function header is shown in the textbox below and given in the DC_md_QEP_Setup.c file. The basics of this function are taken from the eqep_freqcal example in the TI's ControlSUITE folder.

```
void InitEQEP()
{

EQep1Regs.QUPRD=800000;             // Unit Timer =
100Hz=80 MHz/800000.
EQep1Regs.QDECCTL.bit.QSRC=00;       // QEP input is
quadrature count mode

EQep1Regs.QEPCTL.bit.FREE_SOFT=2;   // Position counter is
unaffected by emulation suspend
EQep1Regs.QEPCTL.bit.PCRM=11;        // PCRM=11 mode:
//QPOSCNT is latched into QPOSLAT and reset on unit time
//out event

EQep1Regs.QEPCTL.bit.UTE=1;          // Unit Timer Enable

EQep1Regs.QPOSMAX=4294967295;       // Max value of
POSCNT, = max value of 32 bit int.
EQep1Regs.QEPCTL.bit.QPEN=1;         // QEP enable

EQep1Regs.QCAPCTL.bit.UPPS=2;        // 1/4 for unit
position
EQep1Regs.QCAPCTL.bit.CCPS=6;        // 1/64 for CAP clock
EQep1Regs.QCAPCTL.bit.CEN=1;         // QEP Capture Enable
EQep1Regs.QEPCTL.bit.QCLM=1;         // Latch on unit time
out

EQep1Regs.QEINT.bit.UTO=1;          // Enable unit time
out interrupt

EQep1Regs.QFLG.bit.UTO=1;                  // Unit time out
interrupt flag: Set by eQEP unit timer period match

} // end InitEQEP()
```

FIGURE 3.13: THE INITIALIZATION OF THE QEP MODULE

The quadrature unit timer (QUTMR) is enabled by the UTE bit and set to a frequency of 100 Hz by setting its period (QUPRD) to 800000, i.e. equation 3.20. It is a saw-tooth waveform which counts to 799999 before it is resets

to 0. The zero reaching is called the unit time event and it will trigger the QEP interrupt service routine.

$$f_{QUTMR} = \frac{f_{Sys\ clock}}{QUPRD} \tag{3.20}$$

The position counter (QPOSCNT) is set to count every rising and falling edge of both QEP-A and QEP-B signals, i.e. 4 counts per pulse period. The value is latched into the QPOSLAT bit and then reset on every unit time event, i.e. once every period of the unit timer.

These parameters are used for the high speed measurements. The QPOSLAT value over 4 says how many pulses that have passed, and the unit timer period is the time it took for them to pass. The high speed given in periods per seconds, is thus QPOSLAT over 4 divided by the switching period, given in equation 3.20. To get the speed in revolutions per minute, it is multiplied by 60 s/min over 1024 pulses/rotation. The high speed equation is given below.

$$\begin{aligned} v_{CW,high\ speed}(k) &= \frac{x(k)}{T} = \frac{\frac{QPOSLAT}{4}}{\frac{QUPRD}{80MHz}} * \frac{60}{1024} \\ &= \frac{QPOSLAT}{QUPRD} * 1171875 \end{aligned} \tag{3.21}$$

These parameters are shown in Figure 3.14. Notice the change in direction after two QEP pulses. The QDIR bit is 1 for forward, and 0 for backward rotation. For the illustration of it, QPOSMAX, the maximum value of the position counter, is set to 10 in the figure. As seen, the position counter is reset to the maximum value on the unit time even when rotation counterclockwise. In the code, the maximum value is set to 2^32-1=4294967295. The value of the position counter is reset if it exceeds this value. According to equation 3.21 this gives a possible maximum measureable speed of $6.29*10^9$ rpm, which is more than high enough.

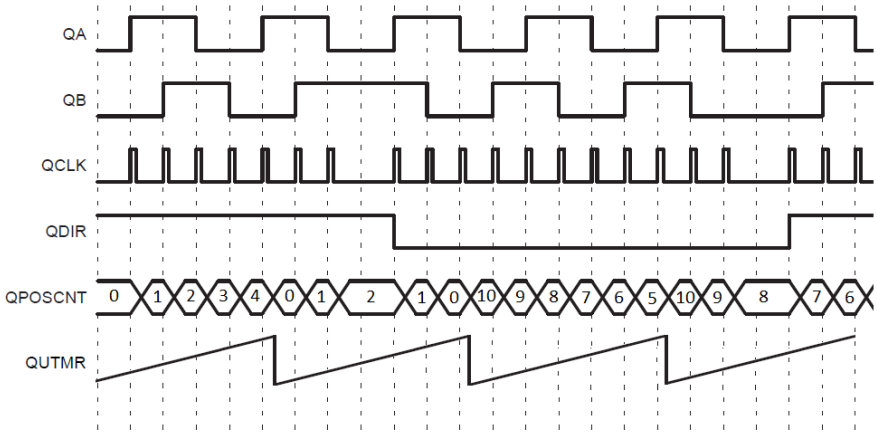


FIGURE 3.14: THE TWO QUADRATURE ENCODER PULSES, QUADRATURE UNIT TIMER, DIRECTION, AND POSITION COUNTER (TEXAS INSTRUMENTS A, 2011)

For counterclockwise (CCW) rotation, the QPOSCNT counts backwards, starting with its maximum value. The high-speed calculation is thus done by subtracting the QPOSLAT (QPOSCNT) value from QPOSMAX value and putting a negative sign in front of the answer. The rest is done as in equation 3.21.

$$\begin{aligned} v_{CCW,high\ speed}(k) &= \frac{x(k)}{T} \\ &= \frac{(QPOSMAX - QPOSLAT)/4}{TBPRD} * \frac{1}{1024} \\ &* 60 \\ &= -\frac{QPOSMAX - QPOSLAT}{QUPRD} \\ &* 1171875 \end{aligned} \qquad 3.22$$

For low speed measurements the QEP edge capture unit is used. It is measuring the time for the QEP pulses to pass. The QEP capture timer (QCTMR) is pre-scaled from the system clock of 80 MHz by the CCPS bit which is set to 1/64. It (QCTMR) is latched into the capture period register (QCPRDLAT) on every unit position event (UPEVNT). Hence, QCPRDLAT/CCPS is the time it takes for one unit position event to occur. The UPEVNT is pre-scaled by the UPPS bit to happen every 4th count of

the position counter (QPOSCNT). Recalling that the position counter counts four per pulse of the QEP signal, the unit position event will happen once every period of the QEP signal. This is summarized in the low-speed equation below. The low speed calculation will be the same regardless of the rotational direction. But for counterclockwise rotation, a negative sign is used.

$$v_{\frac{CW}{CCW},low\ speed}(k) = \frac{x}{t(k)} = \frac{4/4}{\frac{QCPRDLAT}{(CCPS * 80MHz)}} * \frac{60}{1024} = \frac{1}{QCPRDLAT} * \frac{75000000}{1024} \quad 3.23$$

The waveforms of the capture signals are shown in Figure 3.15. In the figure, notice that the speed decreases after a couple of QEP pulses, and that the unit position event (UPEVNT) happens every other, not fourth, position counter (QPOSCNT) in the illustration.

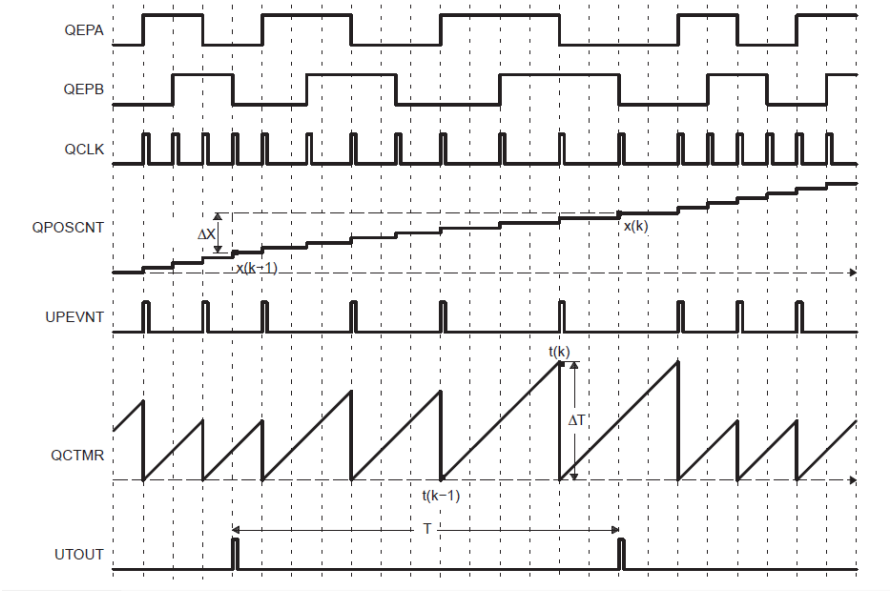


FIGURE 3.15: THE EDGE-CAPTURE PULSES(TEXAS INSTRUMENTS A, 2011)

EQep1Regs.QCPRD is a 32 bit paramenter, which can hold a value up to 2^32-1=4294967295. If the time measured for one slot to pass is this long,

the motor should be considered at stand-still. This will give a speed of $2.84*10^{-7}$ rpm. Therefore, this calculation will work for unrealistic low speeds.

3.5.3 QEP Interrupt Service Routine

The speed control is done within the QEP interrupt service routine (ISR). The main purpose with this file is to calculate the speed and get a new setCurrent value as an input to the current loop.

First, the speed is calculated. The EQep1Regs.QEPSTS.bit.QDF bit indicates if the motor is turning clockwise (=1) or counterclockwise (=0). The register EQep1Regs.QPOSLAT indicates the position counted since last interrupt. If this value is high, the equation 3.21 for high speeds should be used, if the value is low, equation 3.23 for low speeds should be used. The value on the border of high and low, 'border' must be set after testing the calculation. This is done in section 4.2.3.

```
        int border = 70;      //indicates the QPOSLAT value at the border
between the high and low speed measurements

        //Speed calculation
        if (EQep1Regs.QEPSTS.bit.QDF==1) // If forward motor rotation
        {
                //if high speed, equation 3.19 (rpm):
                if (EQep1Regs.QPOSLAT>border) actualSpeed =
1171875*EQep1Regs.QPOSLAT/EQep1Regs.QUPRD;
                // if low speed, equation 3.21 (rpm):
                else if(EQep1Regs.QPOSLAT<=border) actualSpeed =
75000000/EQep1Regs.QCPRDLAT/1024;
        }
        else if (EQep1Regs.QEPSTS.bit.QDF==0)// If reverse motor direction
        {
                Uint32 POSLAT = EQep1Regs.QPOSMAX - EQep1Regs.QPOSLAT;
                //if high speed, equation 3.20 (rpm):
                if(POSLAT >border) actualSpeed = -
(1171875*POSLAT/EQep1Regs.QUPRD);
                //if low speed, equation 3.21 (rpm):
                else if (POSLAT<=border) actualSpeed = -
75000000/EQep1Regs.QCPRDLAT/1024;
        }
        //Store the results in a circular array
        speedArray[index]=actualSpeed;
```

Figure 3.16: Speed calculation

Then, the error between this measured speed value and the desired speed is calculated and this will be the speed PI controllers input.

The speed control loop is given in the text box under. It is explained by the comments and equations in section 3.2.

```
//Speed PI controller:
        speedError = setSpeed - actualSpeed; //Calculate the error
        pTermS  =  kpS*speedError;
        // proportional term
        iStateS += speedError;
        // the sum of the speed errors
        if (iStateS>iStateSMax)iStateS=iStateSMax;   // anti-windup
        if (iStateS<iStateSMin)iStateS=iStateSMin;
        iTermS =  kiS*iStateS*EQep1Regs.QUPRD/80000000; // integral term.
Sampling period=EQep1Regs.QUPRD/80MHz
        speedController=pTermS+iTermS;
        // speed controller's output

        setCurrent+=speedController;  // update setCurrent
        if(setCurrent >4096)setCurrent = 4096;// avoid setcurrent to
become too high.
// This is the maximum value of the input current.
```

FIGURE 3.17: SPEED CONTROL LOOP

To summarize, the most important variables for the speed loop and speed calculation are:

- QUPRD which controls the speed sampling and speed controller frequency. This bit is set in the InitEQEP() function.
- The value of QPOSLAT which decides the border of the high and low speed equation. This value is set by the border variable before the speed-calculation loop in the QEP ISR. This value must be verified for the specific motor chosen, it is done in sub-section 4.2.3.
- The proportional and integral gain, kpS and kiS. These are set at their initialization in the beginning of the main file. These are found as described under section 0 and equation 3.9.

3.5.4 MODIFICATIONS OF THE EXISTING CODE

In the PWM initialization, the PWM-2A and –2B is set to go high and low opposite of the PWM-1A and -1B to get bipolar voltage switching with both control signals being equal. This makes it easier to control the CMPA in the current-control loop. How to do this was described in sub-section 2.4.3. PWM-1A and PWM-2B are now equal, and PWM-1B and PWM-2A are equal.

```
// Settings for the PWM1A and PWM1B
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLB.bit.CAD = AQ_SET;
EPwm1Regs.AQCTLB.bit.CAU = AQ_CLEAR;
// Settings for the PWM2A and PWM2B
EPwm2Regs.AQCTLA.bit.CAD = AQ_SET;
EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm2Regs.AQCTLB.bit.CAD = AQ_CLEAR;
EPwm2Regs.AQCTLB.bit.CAU = AQ_SET;
```

FIGURE 3.18: PWM SETUP

The deviceinit function must also enable the QEP clocks and configure the QEP pins. The QEP-A and QEP-B signals are connected to respectively the GPIO-20 and GPIO-21 pins. Since only PWM channel 1 and 2 are used, the PWM channel 3 is disabled.

```
SysCtrlRegs.PCLKCR1.bit.EQEP1ENCLK = 1;     // eQEP1
SysCtrlRegs.PCLKCR1.bit.EQEP2ENCLK = 1;     // eQEP2
SysCtrlRegs.PCLKCR1.bit.EPWM3ENCLK = 1;     // ePWM3
// 0=GPIO,  1=EQEP1A,  2=MDXA,  3=COMP1OUT
GpioCtrlRegs.GPAMUX2.bit.GPIO20 = 1;
// 0=GPIO,  1=EQEP1B,  2=MDRA,  3=COMP2OUT
GpioCtrlRegs.GPAMUX2.bit.GPIO21 = 1;
```

FIGURE 3.19: INITIALIZATION OF QEP MODULE CLOCK AND PINS

The declaration of the new ISRs and functions, variables and interrupts must be done before the main function. Within the main function, the initialization function InitADC() must be called as well as the enabling of the new interrupts.

The setCurrent is initially set to 0, since the current loop will run around 5000 times before the speed loop is called. The set-speed and set-current are also set to zero initially, to avoid any start up transients. The gains are also set to zero initially. The iStateMax and iStateMin values are set to +/-100. This value, together with the gains must be specified after the system is up and running.

4 Verification of the Design

4.1 Hardware

4.1.1 Mounting of Components on PCB

First, all components were soldered onto the PCB. The surface mounted (SM) components were the first ones out, since these are the most difficult ones, and plenty of space around the component is required during the soldering.

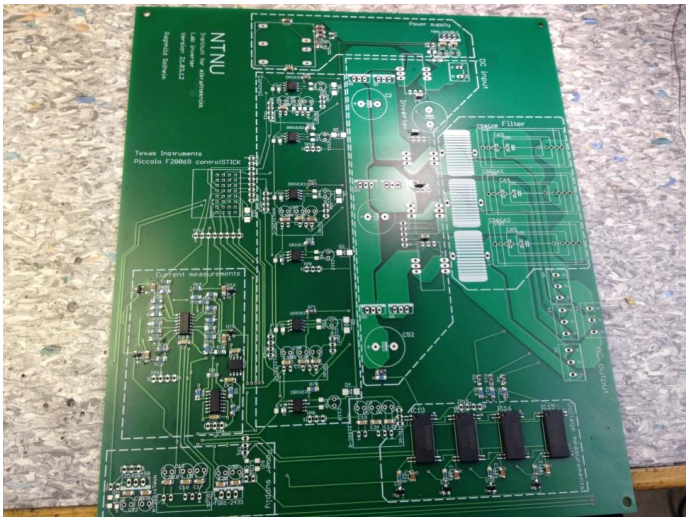


Figure 4.1: The PCB with the surface mounted components

Then the other components were soldered.

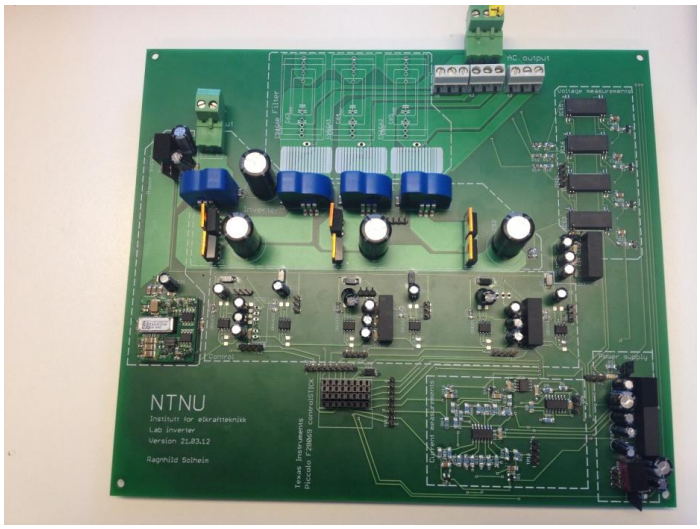


FIGURE 4.2: THE PCB WITH ALL COMPONENTS SOLDERED ONTO IT

The inductor in the +12V_GND2 circuit was a little larger than first assumed, and cupper tape was used to conduct it to its SMD pad as seen in Figure 4.3.

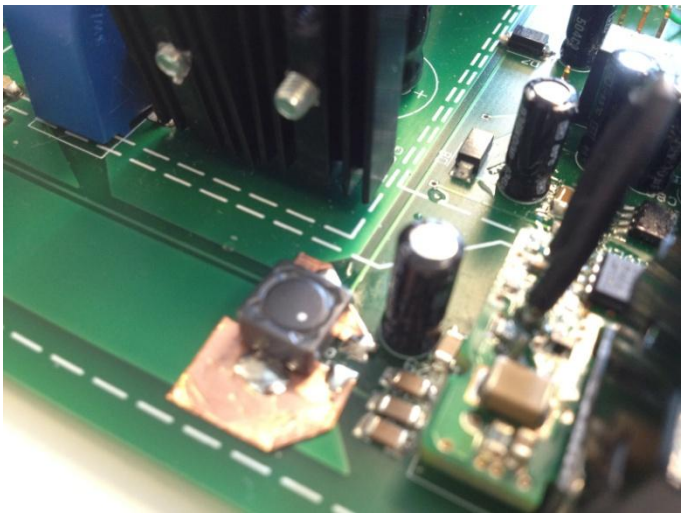


FIGURE 4.3: INDUCTOR IN +12V_GND2 CIRCUIT MOUNTED WITH CUPPER TAPE

The holes for the current transducers were too small. A drill was used to expand the holes. Doing this, there were no longer any connection at the hole's wall, and they had to be soldered on the upper side to be properly connected.

The three TSR DC-DC converters were larger than first assumed; therefore they had to be placed on top of the small capacitors. This works for now, but it needs to be fixed for the next version. A picture of the +12V_GND1 DC-DC converter is shown under as well as clip from the circuit board layout, where the space taken by the DC-DC converters are marked yellow.

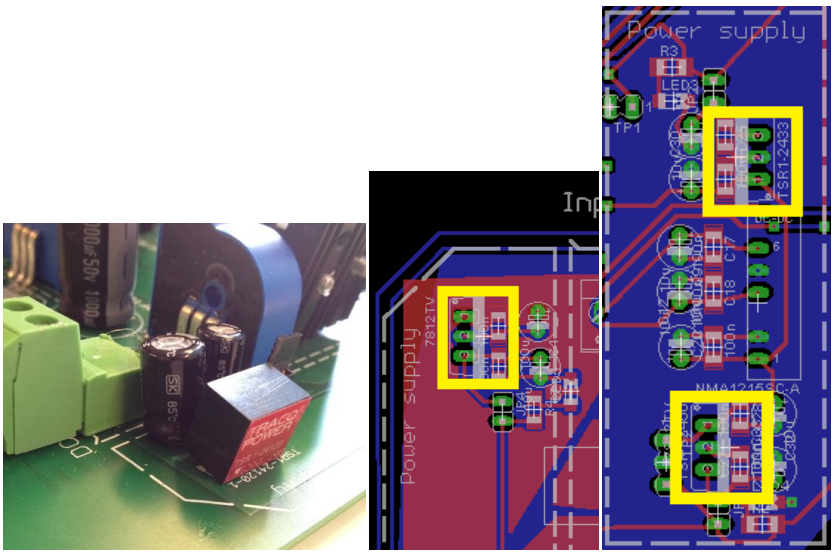


FIGURE 4.4: TSR DC-DC CONVERTER TOO BIG AND PLACED ON TOP OF THE CAPACITORS

The finished board with all components and connection wires is shown in Figure 4.5. The bypassing of the filters, as explained in sub-section 2.2.3, is done as shown with the orange wires in the upper right corner. The heat sinks were mounted with isolation between them and the MOSFETs. Notice that the microcontroller also is mounted.

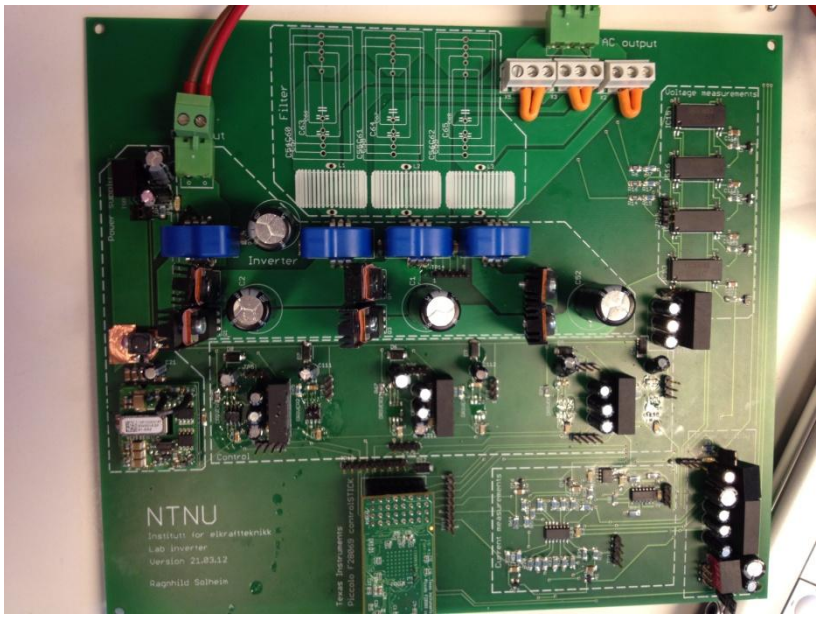


FIGURE 4.5: THE PCB READY FOR USE

4.1.2 VERIFICATION OF PCB

When the mounting was finished, and it turned out the converter did not work, it was difficult to localize the faults on the PCB since all components were connected. A better way to do it would have been to solder and verify part by part. But by debugging the circuit and cutting some current paths, the faults were found eventually. The first one was that the +/-15V_GND2 DC-DC converter was overloaded. It supplied the GND2 side of the four isolated amplifiers as well the two operational amplifiers in the current measurement part. The four isolated amplifiers draw 7 mA each (Burr-Brown Products from Texas Instruments, 1997) and the two operational amplifiers draw 45mA each (Texas Instruments, 1987), while the one converter is able to supply only 33mA (Murata Power Solutions, Inc., 2012). As a temporary solution, the two operational amplifiers were disconnected from the voltage supply. A better solution of this for future versions is presented in chapter 5.

The 5V_GND2 DC-DC converter supplying the current transducers was also overloaded. The reason for this problem is not as clear, the current transducers should draw maximum 18 mA each (LEM, u.d.), and the

+5V_GND2 voltage supply should be able to supply up to 1 A (Traco Power, 2009). Since the operational amplifiers for the current measurements were disconnected, the voltage supply to the current transducers could just as well be disconnected too. The reason of the problem is not yet found, when it was focused on getting the converter part to work first of all.

As explained in sub-section 2.2.5, the two +12V DC-DC converters require conflicted operating voltage ranges. Despite this, both work at an input voltage of around 36 V. However, a better solution should be found for the next version. This is described in chapter 5.

The rest of the power supplies work as they should.

The problem description was modified after the PCB was ordered to include DC motor control. Hence, the speed measurements were not implemented in the PCB. Nor did the TI Piccolo™ controlSTICK have the QEP input pins available. A new microcontroller was ordered, the Texas Instruments Piccolo™ Experimenter's kit: A Piccolo™ controlCARD with a docking station. This was wire-connected to the PCB, since it did not fit the original microcontroller connections on the PCB.

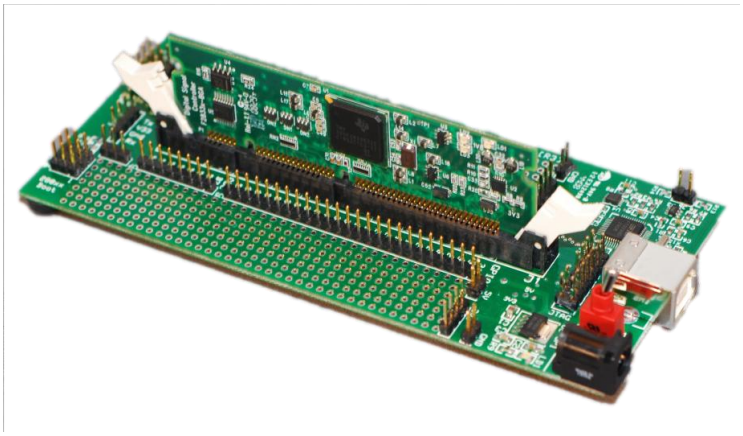


FIGURE 4.6: TI PICCOLO EXPERIMENTER'S KIT: CONTROLCARD WITH DOCKING STATION(TEXAS INSTRUMENTS D, 2012)

The signals from the encoder and its voltage supply are connected to the microcontroller and the PCB via a separate circuit board. The picture under shows the encoder (the black box mounted on the motor shaft to the left),

the separate circuit board and wires. These are the wires for the QEP-A, QEP-B, QEP-1 signals and 3.3V and GND2. The signals are connected to the QEP pins of the microcontroller while the voltage supply is connected to the PCB.

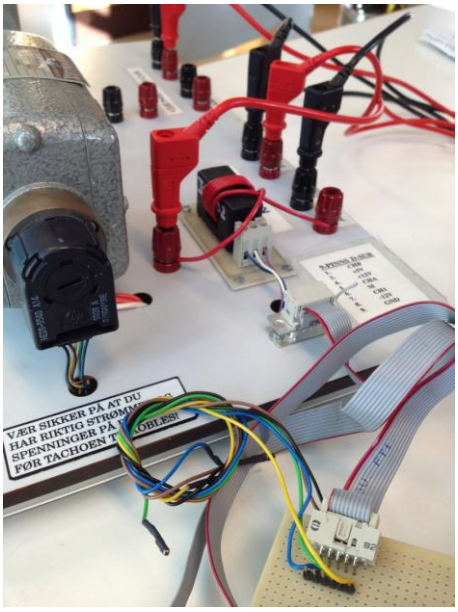


FIGURE 4.7: THE QUADRATURE ENCODER AND ITS CONNECTION WIRES

The Mosfet drivers were supposed to drive the MOSFETs from the PWM signal of the microcontroller. Even if all pins on the driver had the correct voltage level and signal, the driver did not give any output signal. By verifying the drivers on a separate circuit board, it turned out that the high level of the PWM signal must be around 9 V for the drivers to notice it. It is actually written in the datasheet that the high part must be 0.7 times the supply voltage, and that the supply voltage must be in the range of 10-20V (International Rectifiers, A, 2009). Supplying the drivers by a 12 V source would need a PWM high signal of at least 0.7*12 V = 8.4 V. The signal from the microcontroller is only 3.3 V and therefore, the driver did not see it.

A signal generator was then connected to the PCB's PWM input, and the drivers worked properly for that signal, which means that it is only the signal level which prevents the drivers from working. Since the signal generator only had one output channel, both MOSFETs could not be

switched at the same time and the whole bridge leg could hence not be verified by this method.

But the power circuit could be verified even if the MOSFETs could not be controlled by their drivers. First, all MOSFETs were turned off manually, by connecting the gate to the source for a short time, to de-charge the stray capacitances in the MOSFET. This was very important in order to prevent any short circuiting of the main power supply. The voltage between the upper MOSFET's drain, which is the input potential, and the lower MOSET's source, which is GND1 potential, was equal to the applied voltage V_d. This implies that the current transducers and the rest of the circuit from the input connections to the MOSFETs work.

Then, the upper MOSFETs were turned on by connecting the gate to its respective +12V_DRIVE voltage for a short time. The input voltage was now measured at the outputs A, B and C of the PCB. Hence, the power circuit works.

As a solution to the driver problem, an optocoupler (ACNV4506 (Avago Technologies, 2011)) was used to lift the PWM signal from the microcontroller to the driver. Since the optocoupler prevents a better isolation than the driver itself, the drivers were reconnected to as shown in their datasheet with only one voltage supply, +12V_GND1 (International Rectifiers, A, 2009). Because it is a high side driver, it did not need a different supply or arrangement to driver the upper MOSFET in a bridge when connected this way. Since the optocoupler inverts the signal, an inverting buffer (HEF4049B (Fairchild Semiconductor(tm), 1987)) was connected between it and the microcontroller. Otherwise, the blanking time of the PWM signal would have been turned into an "on-time" where both MOSFETs were turned on at the same time, hence short circuiting the input voltage source. The inverting buffer also prevents too much current being drawn from the microcontroller by the octocoupler.

The components were connected as recommended in their datasheets, with values of resistances and capacitances as given there (Fairchild

Semiconductor(tm), 1987)(Avago Technologies, 2011). A schematic of this solution is shown in the figure below.

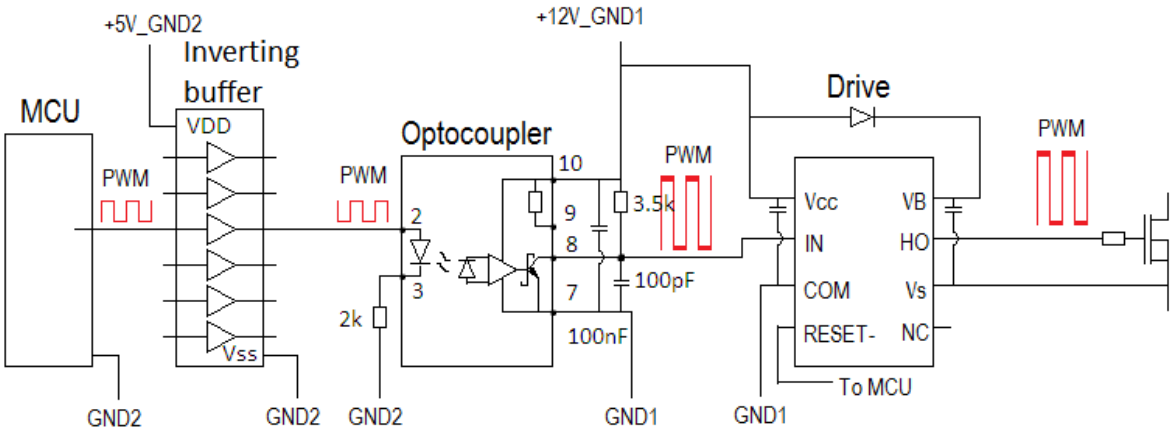


FIGURE 4.8: CIRCUIT DIAGRAM OF OPTOCOUPLER CONNECTION FOR PWM SIGNAL

The inverting buffer is supplied by +5V_GND2, it draws only 4 µA and the voltage supply can supply up to 1A (Fairchild Semiconductor(tm), 1987). The inverting buffer draws a current of +/-0.3 µA from the PWM signal, so there is no problem with too much current being drawn from the microcontroller either (Fairchild Semiconductor(tm), 1987). The optocoupler is supplied on the GND1 side by +12V_GND1. This can supply a total current of up to 1A, and an additional current of 20 mA to the optocoupler is negligible (Avago Technologies, 2011).

Before soldering this part it was verified with a board as shown in the figure below. The power supplies were now gotten from separate power sources and the PWM signal from a signal generator.

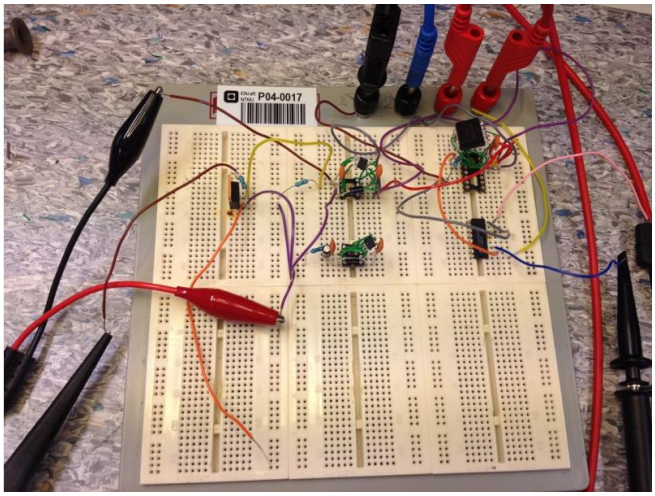


FIGURE 4.9: VERIFICATION OF OPTOCOUPLER CONNECTION

The optocoupler's response time was too long to make it work properly for high frequencies. The resistances connected between its pin 8 and 10 were varied until the best response was reached with 3.5 kΩ. But even then, the output signal was not a perfect square wave and the response time longer than specified in the datasheet (Avago Technologies, 2011). This is most likely due to the long wires connecting it from the microcontroller to the PCB. The capacitors were also tried varied, but without any noticeable change.

The inverter and optocoupler with connections were soldered onto a separate circuit board and connected to the drivers and voltage supply on the PCB. The drivers were disconnected from the SMD pads and reconnected by wires to the new potentials. The signal got all the way to the MOSFETs, turning them on and off. However, because of the shape of the PWM signal from the optocoupler, was not a perfect square wave, the duty ratio of the PWM signal out of the drivers was a bit lower than the signal out of the microcontroller.

After this was done, another driver was found (IRS21171 (International Rectifiers, B, 2009)). This one has the same connections as the previous

one, but it can see an input signal of 3.3 V. Hence, with this driver, everything can be connected as it was planned from the beginning. The octocoupler solution was therefore discarded, because of the complicity of it compared to using the new drivers. Unfortunately, time ran out and the new drivers were not verified.

4.2 Software Verification

4.2.1 PWM Signals

To verify the PWM signals, the microcontroller's PWM pins were connected to an oscilloscope. Figure 4.10 shows the PWM signals from one channel with blanking time of 200 system clock cycles and a duty ratio of 50 %. Figure 4.12 and Figure 4.13 show the same signal, only a little closer and with cursors measuring the forward and rising edge delay (FED/RED). To the right it is seen that the time between the cursors are 2.5 µs which equals a blanking time of 200 cycles with the system time-base clock of 80 MHz, as written in the code. The frequency of the signal is measured to be 50.17 kHz, which is close enough to the desired value of 50 kHz.

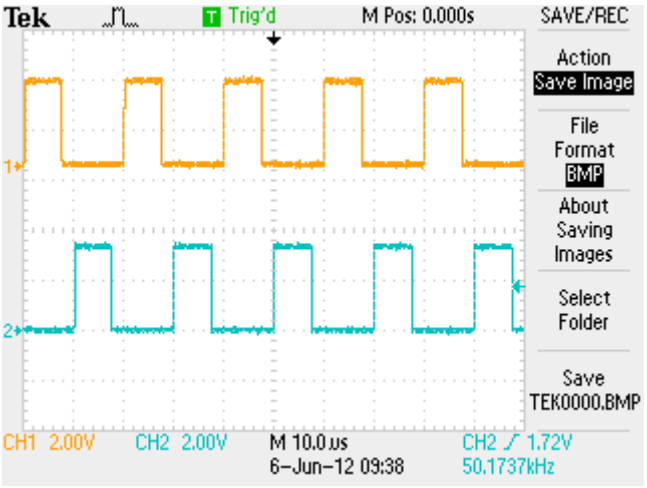


Figure 4.10: PWM1A (yellow) and PWM2A (blue) with a duty ratio of 50 % and a Blanking time of 200 time-base-clock cycles

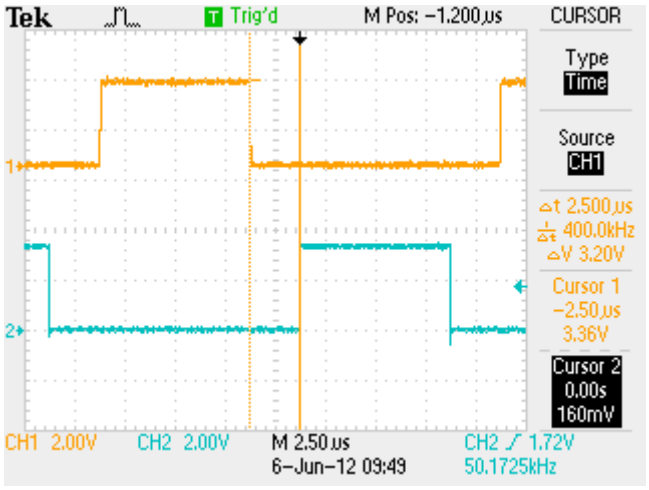


FIGURE 4.11: ONE CYCLE OF PWM1A (YELLOW) AND PWM1B (BLUE) WITH A DUTY RATIO OF 50 %. CURSORS ARE SHOWING FORWARD-EDGE DELAY (FED) AS 2500 µS = 200 TIME-BASE-CLOCK CYCLES

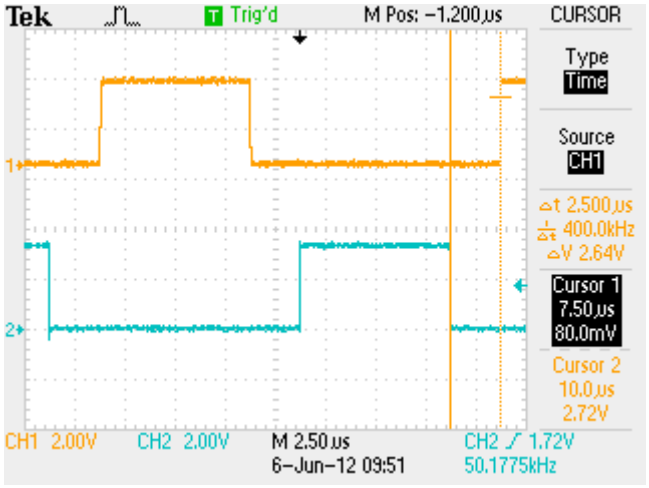


FIGURE 4.12: ONE CYCLE OF PWM1A (YELLOW) AND PWM1B (BLUE) WITH A DUTY RATIO OF 50%. CURSORS ARE SHOWING THE RISING EDGE DELAY (RED) AS 2500 µS = 200 TIME-BASE-CLOCK CYCLES

Figure 4.13 shows the two PWM signals with a CMPA value of 600, which equals a duty ratio of 600/800 = 75 %.

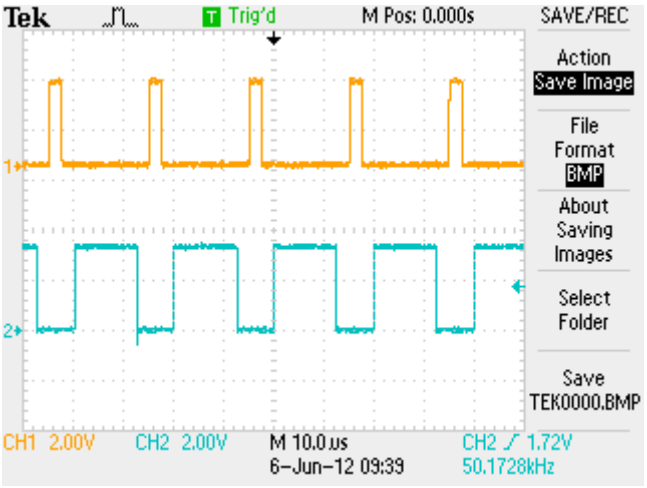


FIGURE 4.13: PWM1A (YELLOW) AND PWM1B (BLUE) WITH A DUTY RATIO OF 75 % AND A BLANKING TIME OF 200 TIME-BASE-CLOCK CYCLES

The next two figures show that the PWM1A and PWM2B are equal and PWM1B and PWM2A are equal as for bipolar voltage switching. The duty ratio is still 75 %.

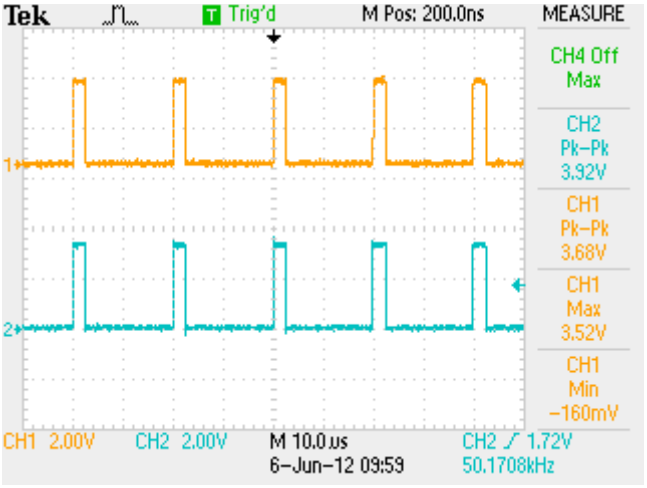


FIGURE 4.14: PWM1A (YELLOW) AND PWM2B (BLUE) FOR DUTY RATIO 75 % WITH BIPOLAR-VOLTAGE-SWITCHING SETUP

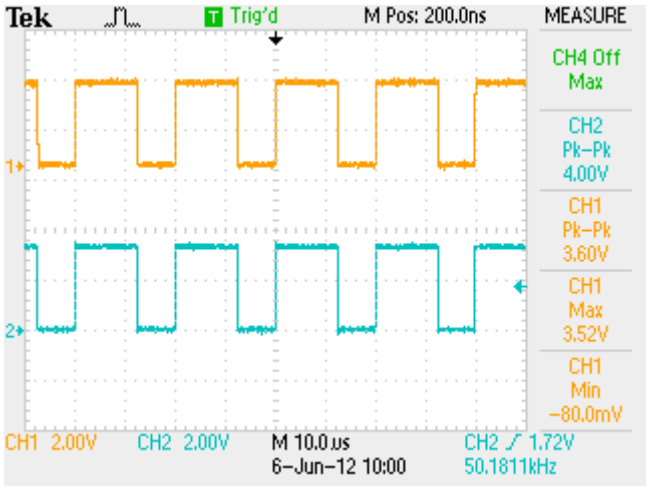


FIGURE 4.15: PWM1B (BLUE) AND PWM2A (YELLOW) WITH DUTY RATIO OF 75 % WITH BIPOLA- VOLTAGE-SWITCING SETUP

4.2.2 ADC

To verify the analog-to-digital conversion ratio, a potmeter was connected with its input pins to the microcontrollers 3.3 V and GND pins and the output to the ADC-A0 pin. The voltage value at the ADC-A0 pin is measured by a multimeter and compared to the digital value gotten from the ADC results. The results are shown in the table under. The expected digital value is calculated from equation 2.9.

TABLE 4.1: VERIFICATION OF ADC

VADC-A0 (V)	Expexcted digital value	ADC result (digital)	Ratio between ADC result and VADC-A0
0.001	1	0	0
0.197	245	288	1462
0.401	498	587	1464
0.600	745	879	1465
0.798	990	1175	1472
1.007	1250	1480	1470
1.208	1499	1780	1474
1.400	1738	2061	1472
1.606	1993	2365	1473
1.802	2237	2655	1473
1.996	2477	2940	1473
2.205	2737	3250	1474
2.400	2979	3535	1473
2.604	3232	3837	1474
2.747	3410	4049	1474
2.797	3472	4095	1464
3.200	3972	4095	1280
3.314	4095	4095	1236

The right column shows the ratio between the digital value gotten from the ADC and the value measured by the multimeter. This ratio should have been equal to 4096/3.3=1241.2 as given in equation 2.9. Even if it far from this value, the ratio stays fairly constant at 1474 for input signals between 1.2 and 2.75 V which seems to be the highest value it can convert. For signals lower than 1.2V, the 1475 ratio could be used without much fault level.

This makes the conversion range 0-2.7 V instead of 0-3.3 V. The current and voltage measurement signals should be scaled for this new range for a new version of the PCB. For now, the maximum current and voltage value which can be measured is given by respectively equation 2.13 and 2.15.

$$I_{max} = \left(2.7 * \frac{100}{75} + 0.5\right) * 6 - 15 = 9.6\,A$$

$$V_{max} = 2.7 * \frac{3000\ k\Omega + +200\ k\Omega}{200\ k\Omega} = 43.2\ V$$

The current maximum value of 9 A is OK. Since the +12V_GND2 DC-DC converter requires an input voltage of 36, and the +12V_GND1 requires a voltage of less than 36 V, the maximum measureable voltage of 43 V is OK as well. However, this might be considered when making a new version of the PCB.

For low voltage and current values, the ADC is not accurate enough. A solution of this problem can be to lift the 0 value a bit, as the current transducers already do. The 0 point of the measured value could give a measurement signal of 0.5 V, for instance. However, this will give a poorer resolution of the results.

It should also be possible to do something about the conversion ratio by calibrating the microcontroller as explained in sub-section 8.1.9 in (Texas Instruments A, 2011).

To summarize, the ADC works, but it should be calibrated or the analog conversion circuit adjusted.

4.2.3 Speed Calculation and Speed-Control Loop

This sub-section shows the verification of the high and low speed calculation and the PI controller in the speed loop.

The verification is done by generating two PWM signals, one shifted 90° from the other, to emulate the QEP-A and QEP-B signals from the encoder. They are generated on the PWM4A and PWM4B pins and connected to the QEP-A and QEP-B pins. These are initialized in a separate function, InitEPWM4() called from the main function. The function header is given in a separate file, DC_md_EPwm4Setup.c. The signals are shown in Figure 4.16 under. To emulate a varying speed, the signals' period (EPwm4Regs.TBPRD) can be varied in the watch window. The CMPA

value is set in the QEP ISR to be half the period, such that the duty ratio always is 50 %.

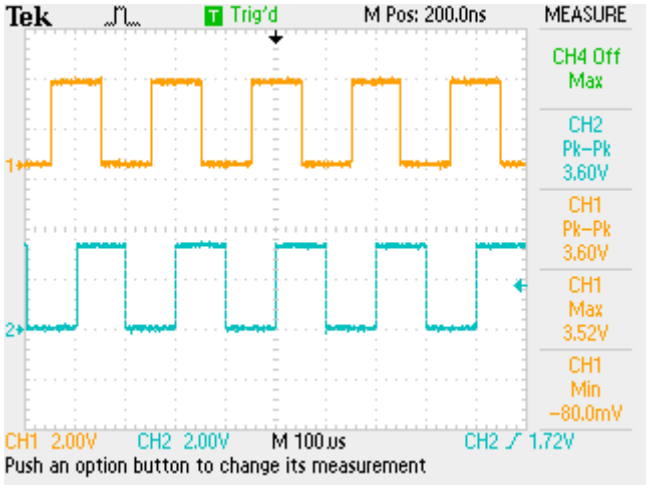


FIGURE 4.16: THE QEP SIGNALS MADE BY THE PWM4A AND PWM4B PINS

First, the high and low speed calculation are verified, to see if they are correct and in which range they work best. This is to find the border parameter described in sub-section 3.5.3. The results are shown in

Table 4.2 under. It shows the TBPRD value of the PWM4 signal, the frequency this will represent calculated by 80MHz/TBPRD and the speed calculated from the frequency by

$$Speed = \frac{f * 60\ s/min}{1024\ slots/rot} \quad 4.1$$

The last two columns show the high and low speed measurements from the code, based on respectively equation 3.21 and 3.22. If the resulting value were alternating between two values, the two are both presented in the table separated by a comma.

TABLE 4.2: VERIFICATION OF THE HIGH AND LOW SPEED CALCULATIONS

TBPRD	Calculated frequency (Hz)	Speed calculated by the frequency	High speed (rpm)	Low speed (rpm)
10	4000000	234375	32767	0, 7706
50	800000	46875	32767	7706, -28915
100	400000	23438	23437	18310, 24414
200	200000	11719	11718	10463, 12207
400	100000	5859	5859	5634, 6103
600	66667	3906	3906	3854
800	50000	2930	2929	2929
1000	40000	2344	2343	2362
2000	20000	1172	1171	1162
3000	13333	781	780-782	779, 787
4000	10000	586	485	485
5000	8000	469	468	469
10000	4000	234	234	234
20000	2000	117	117	117
30000	1333	78	77-79	78
40000	1000	59	58	58
50000	800	47	46	46
60000	667	39	38-39	39
65535	610	36	36.35	36

For the lower speeds, from 36 to 781 rpm (TBPRD = 65535 to 3000), both high and low speed measurements are correct.

It was not possible to emulate lower speeds than 36 rpm, since the TBPRD bit is a 32 bit signed integer which maximum value is 65535. But the low speed measurements should work for these speeds according to sub-section 0.

For speeds higher than 781 rpm, only the high-speed measurements are correct.

Despite what is written in sub-section 0, that the high speed function works for all “unrealistic high speeds”, one can see that the maximum value is reached for TBPRD 50 and 10, of 32767 rpm. However, even if there is a limit, it is high enough.

Therefore, for speeds between 0 and, say, 100 rpm, the low speed measurements should be used. The high speed measurements should be used for speeds between 100 rpm up to as far as it can measure, 32767 rpm.

100 rpm gives a QPOSLAT value of around 70, using equation 3.21. This is now implemented in the “border” parameter in the speed measurements in QEP ISR in DC_md_main-F2806x_1.c.

The backward speed is verified by switching the signal wires such that PWM4A is connected to QEPB and PWM-4B is connected to QEPA. The measurements give the same results as for the forward rotation for all values, only with a minus sign in front.

It is then verified that the PI speed-control loop works. Instead of controlling the set current, the code is rewritten to control the EPwm4Regs.TBPRD value directly as shown in the textbox under. The code will hence control the frequency of the signal until it emulates the speed given in the setSpeed variable. There is no dynamics included within the system, the response is thus immediate and so the gain must be set when the real system is up and running. A proportional gain of 10 and an integral gain of 0.1 are set for this test.

```
//Speed PI controller:
        speedError = setSpeed - actualSpeed; //Calculate the error
        pTermS  =  kpS*speedError;           // proportional term
        iStateS += speedError;        // the sum of the speed errors
        if (iStateS>iStateSMax)iStateS=iStateSMax;  // anti-windup
        if (iStateS<iStateSMin)iStateS=iStateSMin;
        // integral term. Sampling period=EQep1Regs.QUPRD/80MHz
        iTermS =  kiS*iStateS*EQep1Regs.QUPRD/80000000;
        speedController=pTermS+iTermS;       // speed controller's output

        EPwm4Regs.TBPRD-=speedController;
```

FIGURE 4.17: SPEED CONTROLLER CONTROLLING PWM-4

With no dynamics included, the system responds fast at a setSpeed change from 300 to 400 rpm, seen in Figure 4.18.

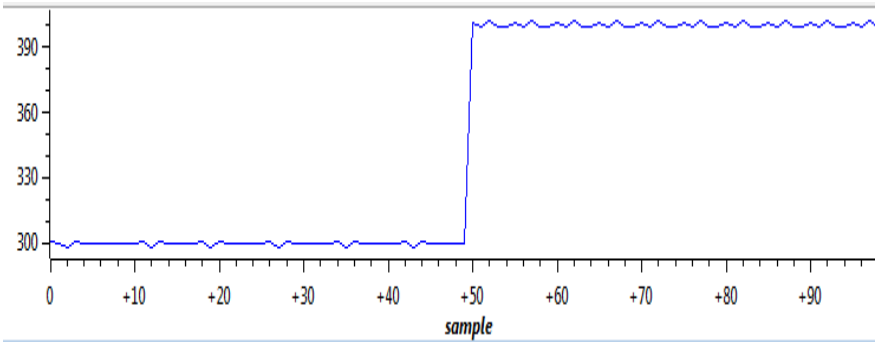


FIGURE 4.18: THE SPEED RESPONSE FOR A CHANGE IN SETSPEED FROM 400 TO 300 RPM.

The current-control loop cannot be verified by this method, but for now, if the speed loop works, it is likely to believe that the current loop does so too.

4.2.4 THE SYSTEM PUT TOGETHER

The project compiles and the pins are used in the parts above, which implies that the device initialization function works.

The different parameters are checked and changed manually in the watch window and the desired response is received, indicating that all parts of the code work and are executed. The correct values are calculated.

4.3 System Verification

The verification of the whole system cannot be done at this stage since the board is still not working. First of all, the drivers are not working, which implies that the MOSFETs do not turn on. Second, the current measurement circuits are not working, such that the control part would not get its required input.

Anyway, to see how it should have been, the system is mounted as shown in the block diagram of Figure 3.11. This includes the main power supply, a motor with its field supply, quadrature encoder and load. The load is a generator with a load resistance of 10 Ω. The setup is shown in Figure 4.19.

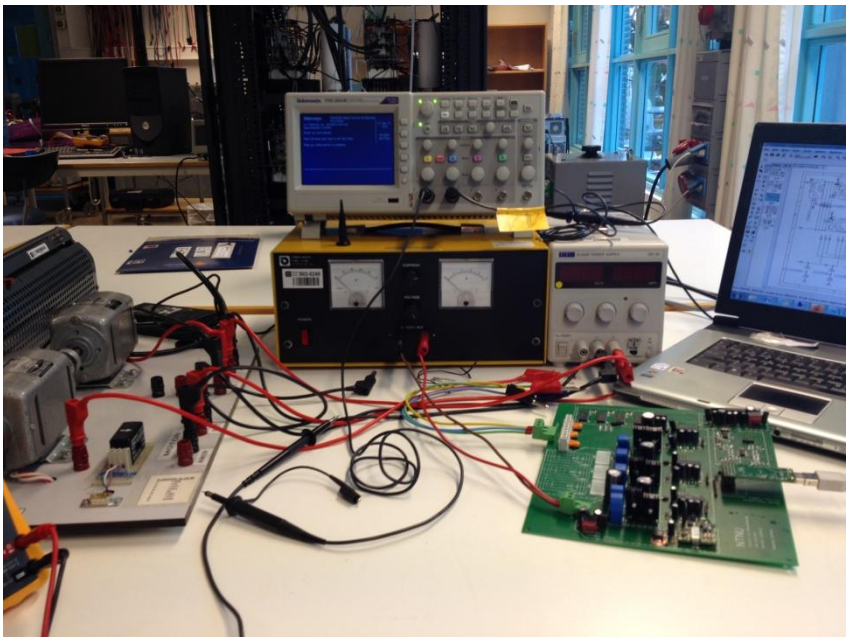


Figure 4.19: System setup for DC motor control

The main power supply (the yellow one in the middle) supplies the circuit board. A separate power supply (the grey beside of the yellow) supplies the field current of the motor and load-generator. The motor and generator are mounted on the board shown to the left.

Only the two bridge-legs used are connected to the microcontroller. PWM-1A, PWM-1B, PWM-2A and PWM-2B are used to control respectively driver 1 to 4.

The current and voltage measurement signals are connected to each respective ADC pin (Table 2.2).

The computer and microcontroller must be at the same potential as GND2, which is floating with respect to the power points and GND1. The computer thus had to be connected to a separation transformer.

The output of the QEP module was connected via the separate circuit board to the QEP inputs. Its power supply wires are connected to the PCB.

To prevent connection transients, the system was connected before the code was started. The code was also the first that was turned off.

4.3.1 Speed Measurement Verification

For these verifications, the motor was connected directly to a variable power supply and a to a field current supply together with the generator which was set to $V_f = 14.5V$ and $I_f = 1.35$ A . The generator load was a 8.2 Ω resistor. The PCB was connected to its power supply. The quadrature encoder was connected to the 3.3 V voltage supply and GND2 at the PCB. The microcontroller was also connected to the PCB.

First, the QEP-A and QEP-B signals are connected to the oscilloscope to see the waveforms for different speeds. The field supply was constant, while the input voltage was varied, to emulate the variable voltage out of the converter. Two pictures recorded from the oscilloscope are shown below, for an input voltage of respectively 11.26 V and 18.99 V, both within the range of the converter, which makes this realistic.

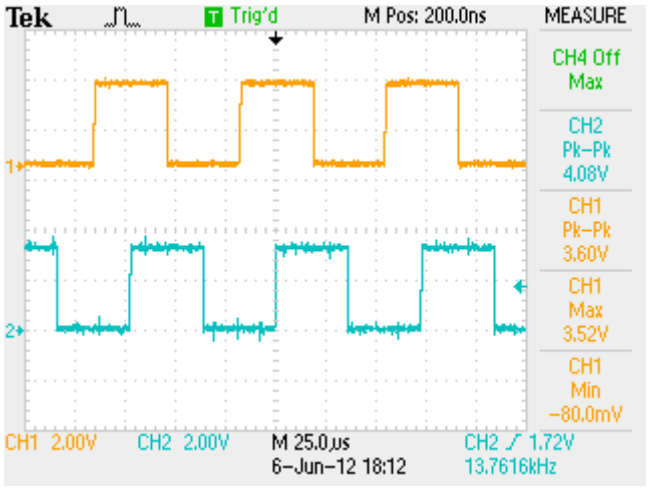


FIGURE 4.20: QEP SIGNALS WITH INPUT VOLTAGE 11.26 V AND INPUT CURRENT OF 1.391 A.

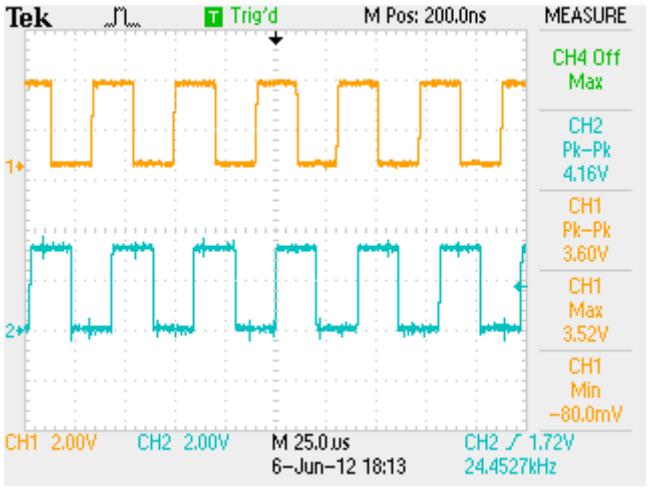


FIGURE 4.21: QEP SIGNALS WITH INPUT VOLTAGE OF 18.99 V AND INPUT CURRENT OF 2.179 A

The speed can be calculated by the frequency by equation 4.1, which is written at the lower right corner of the figures, respectively 13.7616 and 24.4527 kHz.

Speed1=13.7616 kHz/1024 slots/rotation*60 sec/min = 806.32 rpm

Speed2=24.4527 kHz/1024 slots/rotation*60 sec/min= 1432 rpm

Then, the speeds for several input voltages are measured by both the code and oscilloscope. The results are shown in

Table 4.3. The first two columns show the voltage and current input to the motor. Both high and low speed measurements from the code are shown, but the red ones are the ones that should not be used according to sub-section 4.2.3. The QEP frequency is measured by the oscilloscope, and the 6th column shows the speed calculated by the frequency with equation 4.1. The fault level is shown in the rightmost column.

The code is rewritten a little in order to get both high and low speed values such that these can be compared.

TABLE 4.3: REULTS OF SPEED-MEASUREMENT

DC motor input voltage	DC motor input current	High speed calculation (equation 3.21)	Low speed calculation (equation 3.23)	QEP frequency	Speed calculated by the frequency and equation over	Fault level: coloumn 3 or 4 minus 6
23.07	2.58	1757	1786-1843	30.13	1765	-8
22.15	2.4	1721	1720	29.50	1729	-8
21.08	2.3	1640	1641	28.05	1644	-4
20.16	2.21	1561	1592	26.78	1569	-8
19.09	2.10	1483	1494	25.33	1484	-1
18.18	2.01	1406	1408-1436	24.08	1411	-5
17.02	1.89	1313	1300-1321	22.50	1318	-5
16.03	1.79	1233	1210-1240	21.10	1236	-3
14.96	1.7	1149	1148	19.67	1153	-4
14.07	1.62	1070	1072	18.40	1078	-8
13.08	1.53	993	989	17.01	997	-8
1.99	1.42	906	904	15.48	907	-3
11.05	1.32	829	828	14.18	831	-3
10.00	1.22	747	747	12.80	750	-3
8.99	1.13	666	665	11.37	666	-1
7.98	1.03	584	581	9.99	585	-4
6.9	0.93	498	498	8.53	500	-2
6.17	0.85	439	438	7.54	442	-4
5.60	0.73	348	345	5.96	349	-4
3.97	0.63	266	267	4.58	268	-1
2.97	0.53	188	187	3.22	189	-2
2.09	0.47	116	117	1.97	115	-2
1.01	0.35	27-33	29-38	0.56	33	-1

The difference between the value calculated by the code in column 3 or 4 and the value calculated from the frequency measured by the oscilloscope in column 5, is small. This is most likely due to inaccurate readings; all values were alternating a bit while reading them.

5 IMPROVEMENTS ON PCB VERSION 2

Most of the faults found and described in the earlier parts of the report, are fixed in a new version of the PCB, version II. These are not yet verified, but the ratings or placing of the new components is based on the faults found in version I. Version II is saved in the new Eagle file master v2.

The sheets have been reorganized a little. The first sheet now includes the power supplies on GND1 and the three upper drivers. The second sheet includes all power supplies for GND2. The third and fourth sheets are unchanged. Sheet three still includes the power circuit, bridge-legs, MOSFETs, drivers, filters, current transducers and voltage dividers. Sheet four still includes the signal conversion circuits for the current and voltage measurements. The fifth and last sheet includes the test pins, the microcontroller connections, the quadrature connections and some capacitances for the +5V source.

First of all, new operational amplifiers for the current measurement circuits are found (LMC660CM (National Semiconductor, 2006)). These are supplied by their own power supplies, two NMA1212SC. According to the datasheets, the new operational amplifiers need a maximum current supply of 18 mA and an input voltage between 5 and 15.5 V(National Semiconductor, 2006). The new power supplies can supply up to 42 mA at 12 V (Murata Power Solutions, Inc., 2012).

The +12V_GND2 voltage supply is changed to SC001A2B91Z, to suit the voltage level of the +12V_GND1 voltage converter better. Both of them are now requiring a voltage level between 18 and 36 V(Lineage Power, 2009).

The drivers are changed to IRS21171 because they suite the output PWM signal from the MCU (International Rectifiers, B, 2009).

During the verification and mounting of the components on the PCB, more test pins were needed. The new components also require their own test pins. Therefore, the test pins are reorganized a little and some new pins are added.

The microcontroller connection is new. Since it is already mounted on a docking station which has a PC connection and LEDs, this is considered the best solution for now. The pins are therefore connected to the PCB by wires.

Also included in the new PCB is the connections for the qudrature encoder.

Larger SMD pads for the inductor in +12V_GND2 circuit are made. Larger space for the TSR DC-DC converters (+12V_GND1, +3V3_GND2, +5V_GND2) is also given. The diameter of the holes for the current transducers is increased.

An overview of the new power supplies are given in the block diagram under and a new list of components are added in the appendix B table B-1.

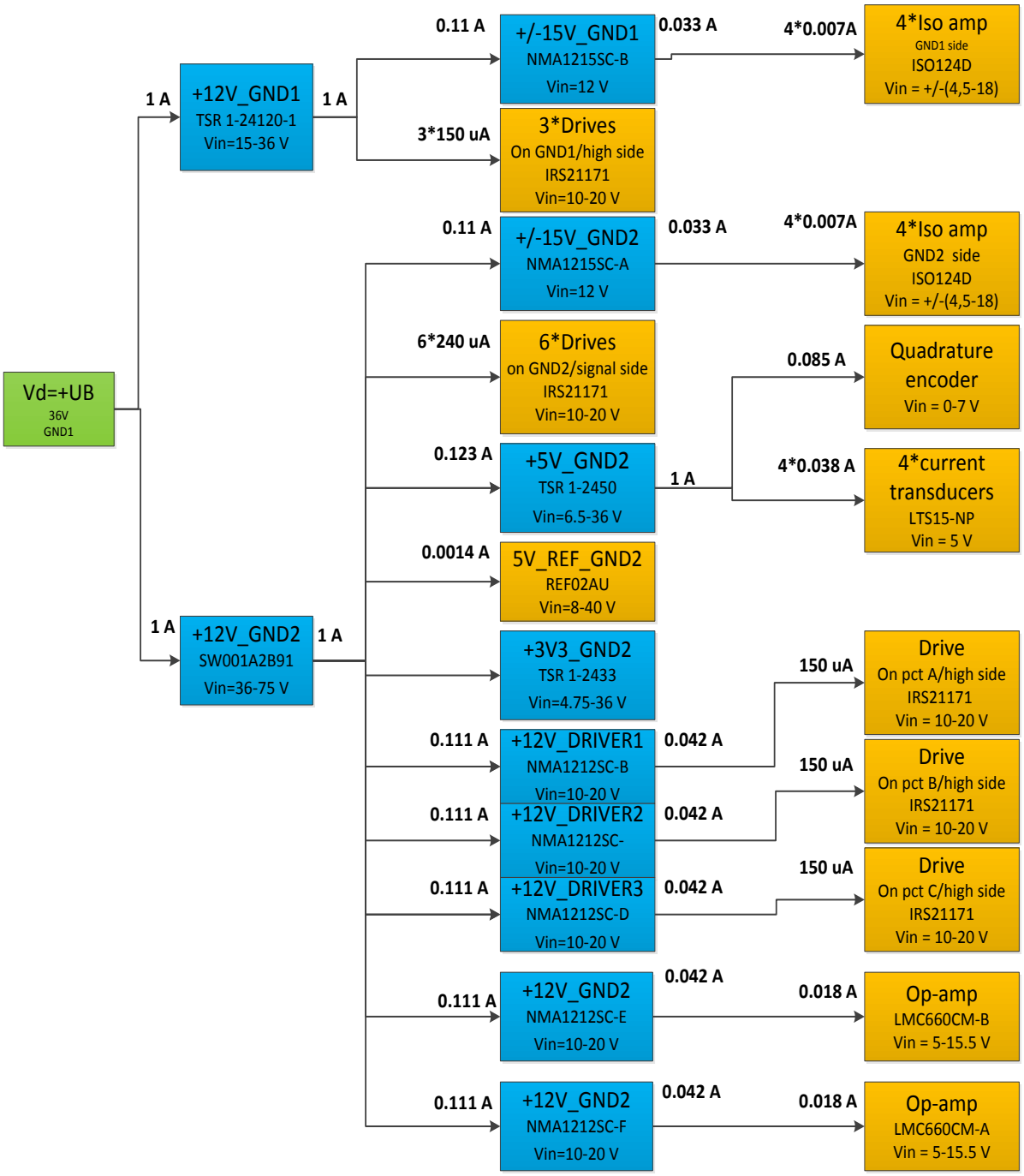


FIGURE 5.1: OVERVIEW OF THE POWER SUPPLIES ON PCB AFTER IMPROVEMENTS

(Burr-Brown Products from Texas Instruments, 1993)(International Rectifiers, 2011)(International Rectifiers, B, 2009)(LEM, u.d.)(Lineage Power, 2009)(Murata Power Solutions, Inc., 2012)(National Semiconductor, 2006)(Traco Power, 2009)

6 Conclusion and Further Work

6.1 What is Done

The design of a three-leg converter topology is presented in this thesis. The converter was implemented on a PCB together with other necessary equipment such as power supplies, filters, current and voltage measurement circuits, and microcontroller connections. The schematic and layout of the PCB was planned in CadSoft Eagle PCB Design Software. The circuit board was mounted and verified and improvements to the PCB implemented on a second version of the Eagle files. The main parts that remain are to get the MOSFET drivers and the measurement circuits to work.

A TI Piccolo™ ControlCARD microcontroller controls the circuit board and the code implemented is written in Code Composer Studio™. The code is meant to be used as a general code, which must be tailored for the specific purpose of the converter. The code is verified through testing, except for the parts of that needed to be verified using the PCB.

A system for DC motor control is planned and implemented. A DC motor, a quadrature encoder and power supply were connected to the PCB, the general code was tailored to be able to control the speed and input current of the motor. As much as possible of the system were verified, but since the PCB was not functioning at the end of the laboratory work, the system as a whole could not be demonstrated or verified.

6.2 Present state of system

At the present state, all voltage supplies and the power circuit part of the PCB works. The drivers get all their correct inputs, but the drivers themselves are not suitable for the purpose and do not give any PWM output. New drives are found, but not verified. The voltage-measurement-

signal-isolation circuit is expected to work, just waiting for a variable voltage output voltage to measure.

In the control code the device initialization, the main function and file, and the setup for the PWM, QEP and ADC work. The speed measurements and speed controller also function. What remains is to calibrate the ADC to get the correct conversion ratio, and to verify the current controller code and find the proper gain constants, anti-windup constants and maximum current output values. Also, the control-system must be verified as a whole.

The quadrature encoder is connected to the microcontroller and it works as it should.

6.3 Further Work

1. Verify the new drivers on the PCB
2. When the drivers work, verify the PCB and the general code by measuring the output voltage while changing the CMPA value
3. Connect the DC motor at the converter's output and try to vary the speed manually by varying CMPA. Connect the quadrature encoder and measure the speed digitally.
4. Verify voltage measurement circuit
5. Calibrate the ADC ratio of the microcontroller in the code
6. Find out why +5V_GND2 was overloaded with current transducers connected
7. Implement the new improvements and order the PCB version II. Solder and verify part by part, especially the current measurement circuit
8. When all mentioned above works, the control part of the code must be verified. Start with the current loop, anti-windup and the maximum current output value. Finally find the controller's gain constants.
9. Optimize the existing system,
    - Less losses

- More accurate conversions
- Shorter current paths
- Place microcontroller on the PCB
- Make a faster code with less calculations within the interrupt

Or use the converter in a three-phase AC motor drive. Much of the control theory for the DC motor can be used, but it will be more complicated and challenging.

10. Remember to set aside enough time for laboratory work and do not assume that it will work initially.

7 REFERENCES

Avago Technologies, 2011. *ACNV5406.* s.l.:Avago Technologies.

Balchen, J. G., Andresen, T. & Foss, B. A., 1999. *Reguleringsteknikk.* 1. red. Trondheim: Tapir.

Burr-Brown Products from Texas Instruments, 1993. *Datasheet +5V Precision Voltage Reference.* s.l.:Burr-Brown Products from Texas Instruments.

Burr-Brown Products from Texas Instruments, 1997. *Datasheet Precision Lowest-Cost Isolation Amplifier ISO124.* s.l.:Burr-Brown Products from Texas Instruments.

Chapman, S. J., 2005. *Electric Machinery Fundamentals.* Singapore: McGraw-Hill.

Fairchild Semiconductor(tm), 1987. *CD4049UBC Hex Inverting Buffer.* s.l.:Fairchild Semiconductor Corporation.

Hewlett Packard, u.d. *Quick Assembly Two and Three Channel Optical Encoders, Technical Data, HEDS-5540.* s.l.:Hewlett Packard.

International Rectifiers, A, 2009. *Datasheet IRS2123S/IRS2124S.* s.l.:International Rectifiers.

International Rectifiers, B, 2009. *Datasheet IRS21171 Single Channel Driver.* s.l.:International Rectifiers.

International Rectifiers, 2011. *Datasheet IRFB4110PbF HEXFET Power MOSFET.* s.l.:International Rectifiers.

International Rectifiers, 2012. *Application Note AN-978: HV Floating MOS-Gate Driver ICs.* [Internett]
Available at: http://www.irf.com/technical-info/appnotes/an-978.pdf
[Funnet Mai 2012].

Ishengoma, F., 2011. *Lecturen notes in ELK-21 Electronics for Control of Power at NTNU, fall 2011*. s.l.:s.n.

Ishengoma, F., Schimpf, F. & Norum, L., 2011. *DSP-controlled Photovoltaic Inverter for Universal Application in Reasearch and Education.* Trondheim: s.n.

LEM, u.d. *Current Transducer LTS 15-NP*. s.l.:LEM.

Lineage Power, 2009. *Datasheet SW/SC001/003 Series DC-DC Converter Power Modules.* s.l.:World Wide Headquarters Lineage Power Corporation.

Mohan, N., 2003. *Electric Drives - an integrative approach.* Minneapolis: MNPERE.

Mohan, N., Undeland, T. & Robbins, W. P., 2003. *Power Electronics.* s.l.:John Wiley & Sons, Inc.

Murata Power Solutions, Inc., 2012. *Datasheet NMA 5V, 12V & 15V Series.* s.l.:Murata Power Solutions, Inc..

National Semiconductor, 2006. *Datasheet LMC660 CMOS Quad Operational Amplifier*. s.l.:National Semiconductor.

Rahman, F., 2008. *Electric Drive Systems.* Sydney: s.n.

Savitch, W., 2008. *Apsolute C++*. Boston: Pearson Education, Inc.

Texas Instruments A, 2011. *TMS320x2806x Piccolo Technical Reference Manual*, s.l.: s.n.

Texas Instruments B, 2012. *Texas Instruments - Digital Signal Processors and ARM Microprocessors.* [Internett]
Available at:
http://www.ti.com/lsds/ti/dsp/support/dev_tool/ccs_overview.page
[Funnet June 2012].

Texas Instruments C, 2012. *Texas Instruments Microcontrollers.* [Internett]
Available at:

http://www.ti.com/mcu/docs/mcuproductcontentnp.tsp?sectionId=95&familyId=919&tabId=2883&DCMP=Piccolo&HQS=piccolo
[Funnet June 2012].

Texas Instruments D, 2012. *Texas Instruments.* [Internett]
Available at: http://www.ti.com/tool/tmdsdock28335
[Funnet Mai 2012].

Texas Instruments, 1987. *Datasheet TLC274 LinCMOS(TM) Precision Quad Operational Amplifiers.* s.l.:Texas Instruments.

Texas Instruments, 2010. *C2000 Piccolo One-Day Workshop Module 3.* [Internett]
Available at:
http://processors.wiki.ti.com/index.php/File:Piccolo1DayWorkshopMod3_36_477.png
[Funnet Mai 2012].

Traco Power, 2009. *Datasheet DC/DC Converters TRS 1 Series, 1A.* s.l.:Traco Electronic AG.

Wescott, T., 2000. *www.eetimes.com.* [Internett]
Available at:
http://www.eetimes.com/ContentEETimes/Documents/Embedded.com/2000/f-wescot.pdf
[Funnet April 2012].

Würth Elektronik, 2005. *Datasheet 744877100 Power-Choke WE-DD.* s.l.:Würth Elektronik elSos GmbH & Co.KG.

APPENDIX A: PCB VERSION I

A-1: LIST OF COMPONENTS

TABLE A-1: LIST OF COMPONENTS VERSION I

	COMPONENT	MANUFACTURER PART. NO.	ORDER CODE AT FARNELL	QUANTITY	REFERENCE
1	DC-DC converter	TSR 1-24120	1672130	1	(Traco Power, 2009)
2	DC-DC converter	NMA1215SC	1021441	2	(Murata Power Solutions, Inc., 2012)
3	DC-DC converter	SW001A2B91	2076930	1	(Lineage Power, 2009)
4	DC-DC converter	TSR 1-2433	1696319	1	(Traco Power, 2009)
5	DC-DC converter	TSR 1-2450	1696320	1	(Traco Power, 2009)
6	DC-DC converter	NMA1212SC	1021439	3	(Traco Power, 2009)
7	Inductor	744877100	1869664	1	(Würth Elektronik, 2005)
8	Voltage reference	REF02AU	1234885	1	(Burr-Brown Products from Texas Instruments, 1993)
9	Opamp	TLC274INE4	1234855	2	(Texas Instruments, 1987)
10	Current transducer	LTS15-NP	1617409	4	(LEM, u.d.)
11	Driver	IRS2123SPBF	1925150	6	(International Rectifiers, A, 2009)
12	MOSFET	IRFB4110PBF	1436955	6	(International Rectifiers, 2011)
13	Isolated opamp	ISO124D	1544012	4	(Burr-Brown Products from Texas Instruments, 1997)

A-2: SCHEMATICS

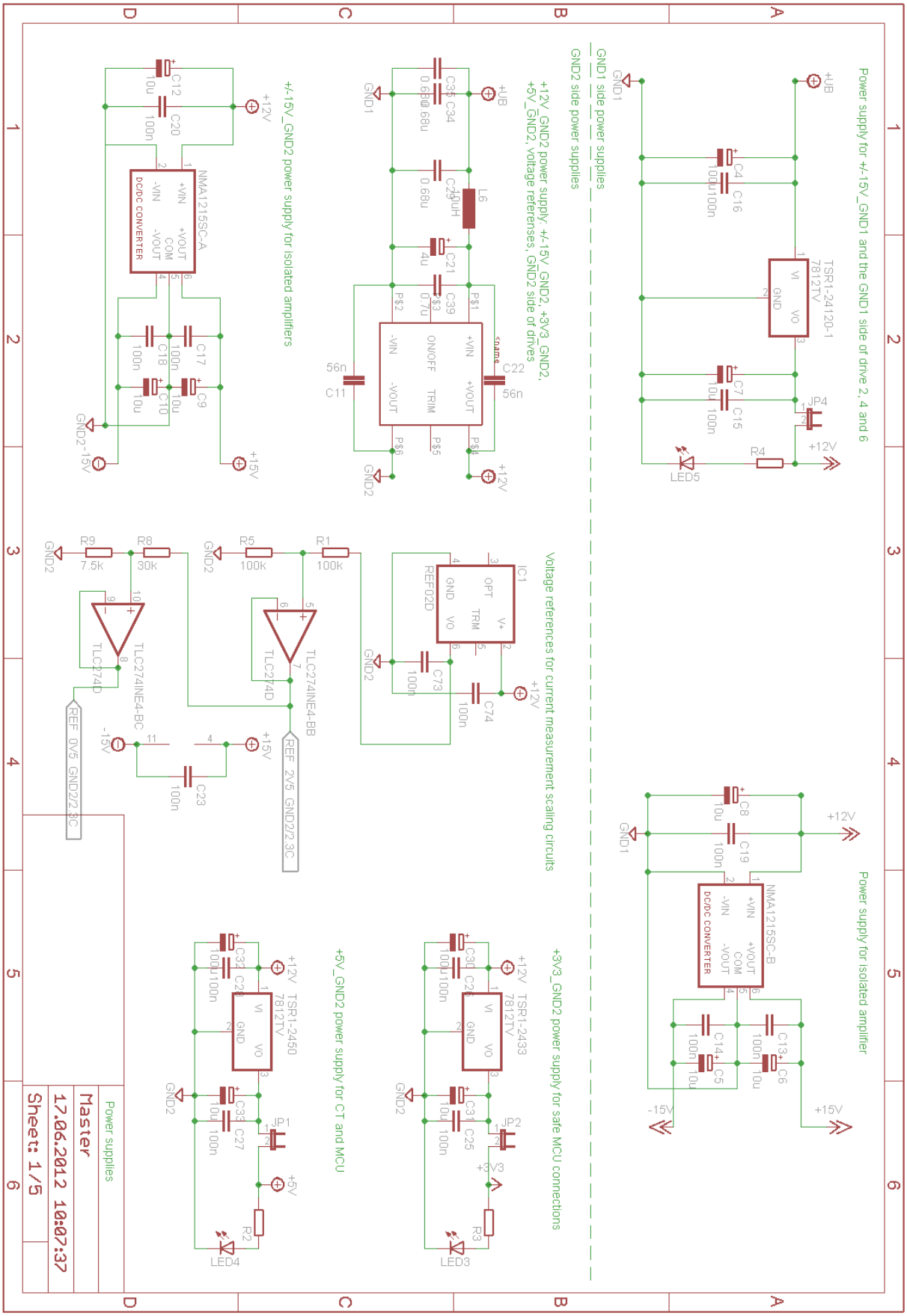


FIGURE A-1: SHEET 1

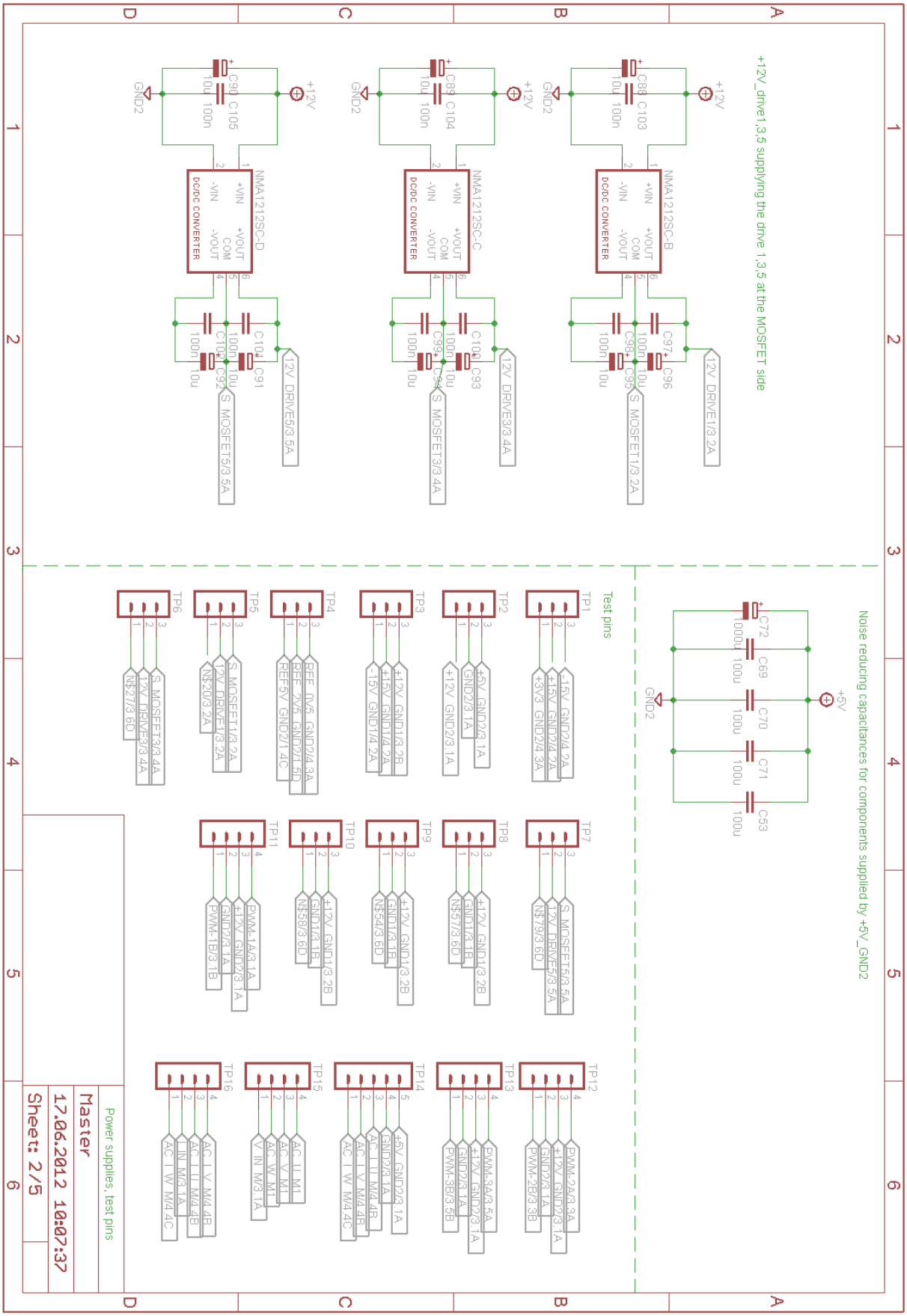


FIGURE A-2: SHEET 2

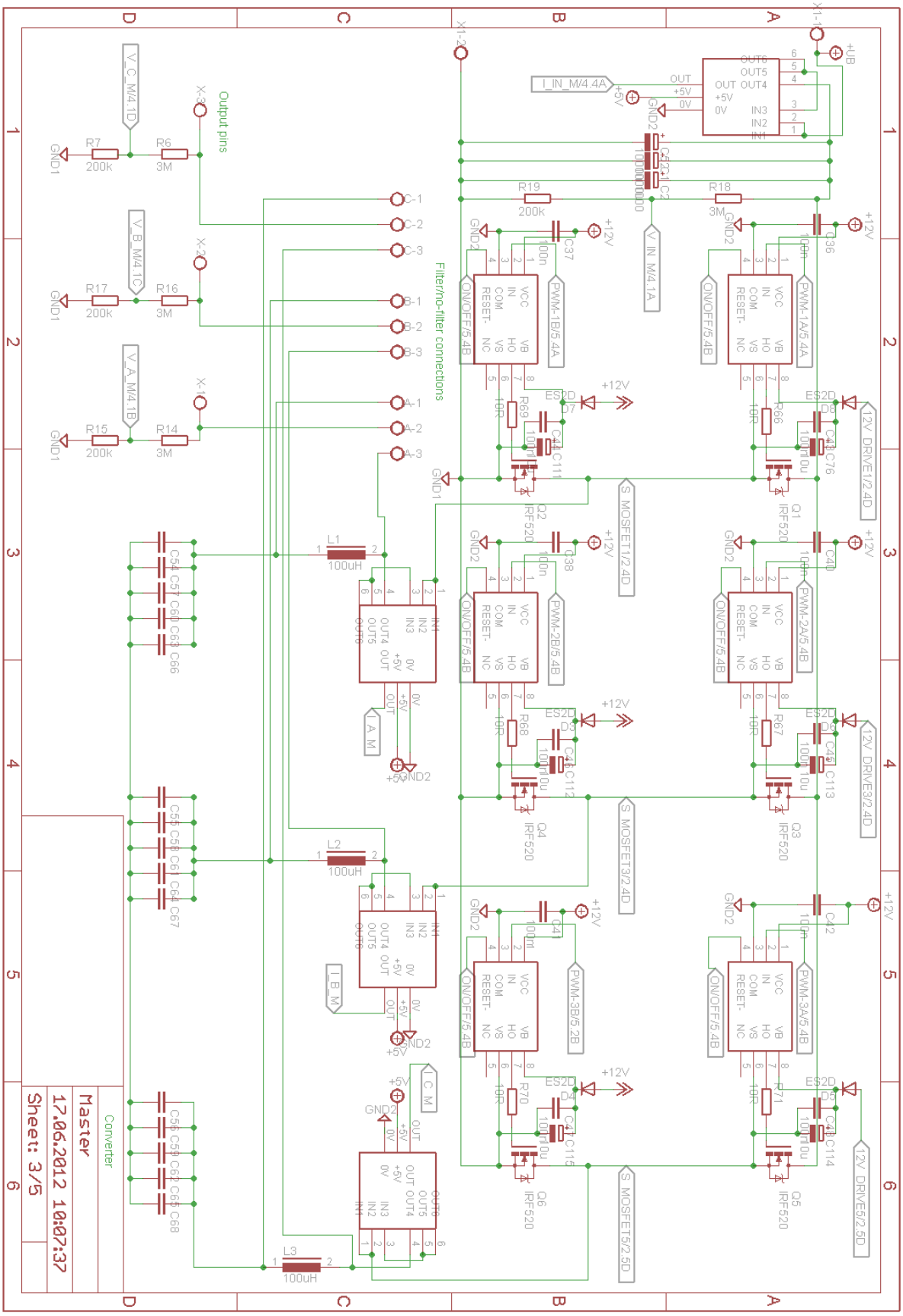


FIGURE A-3: SHEET 3

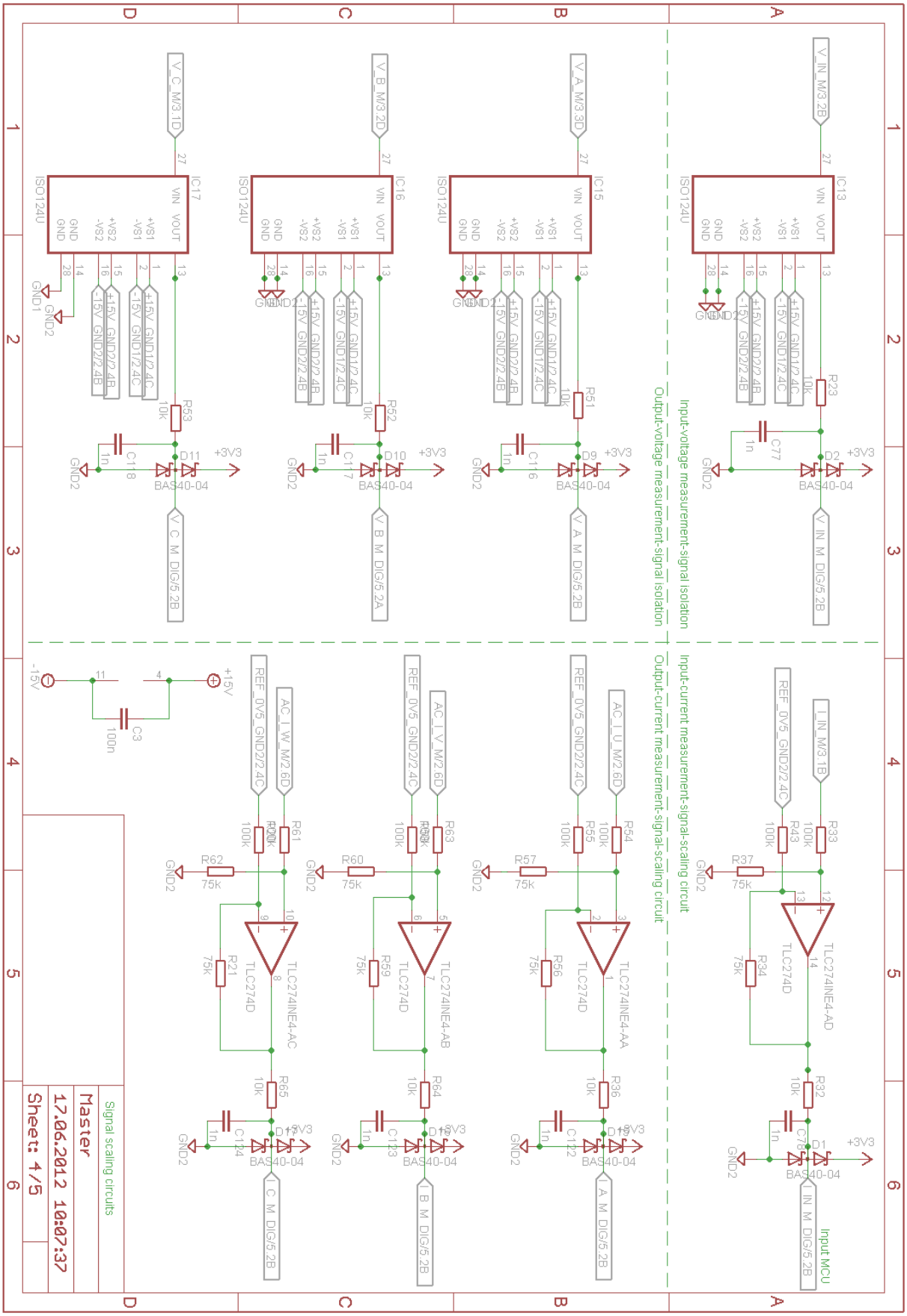


FIGURE A-4: SHEET 4

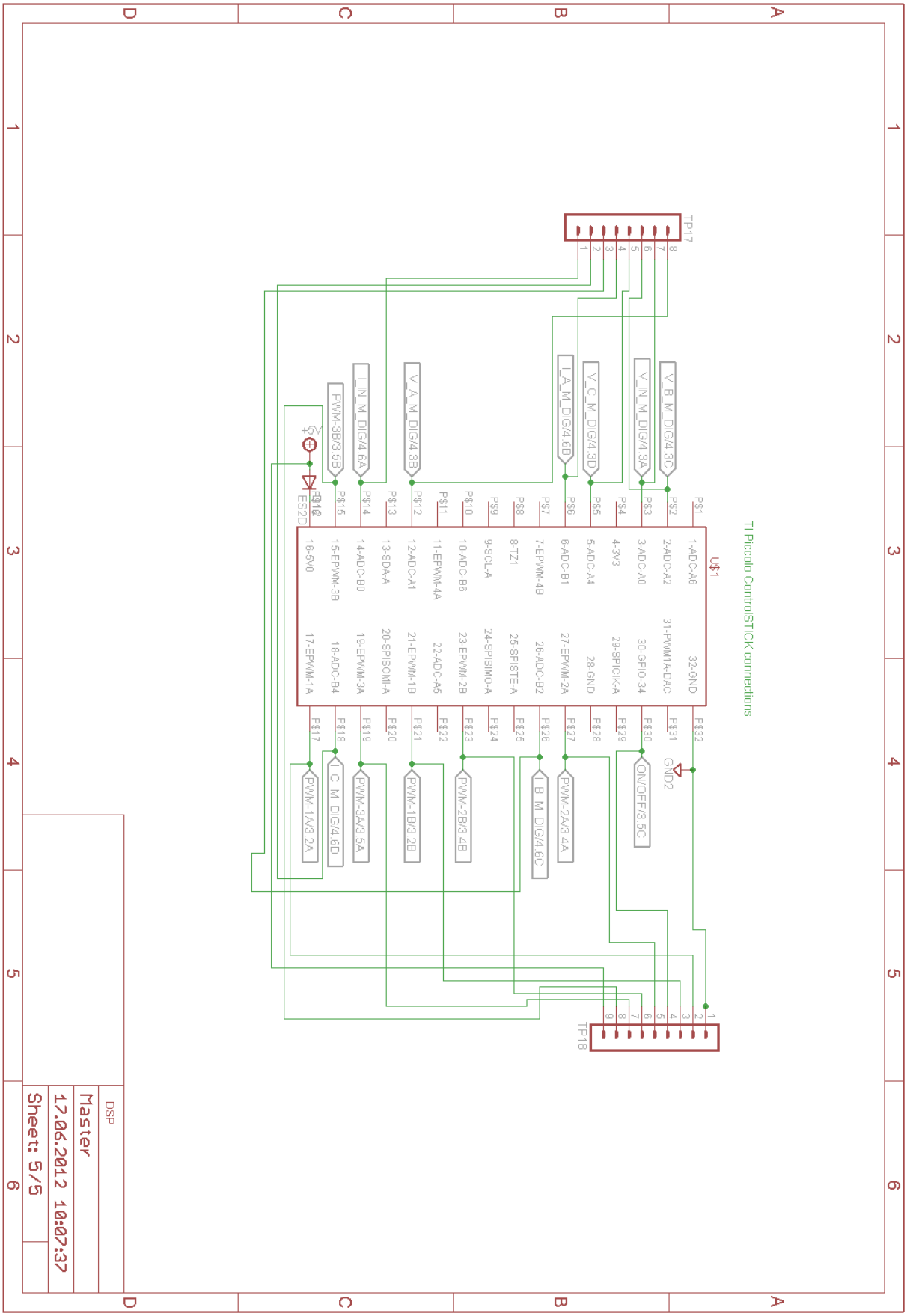


FIGURE A-5: SHEET 5

A-3: Layout

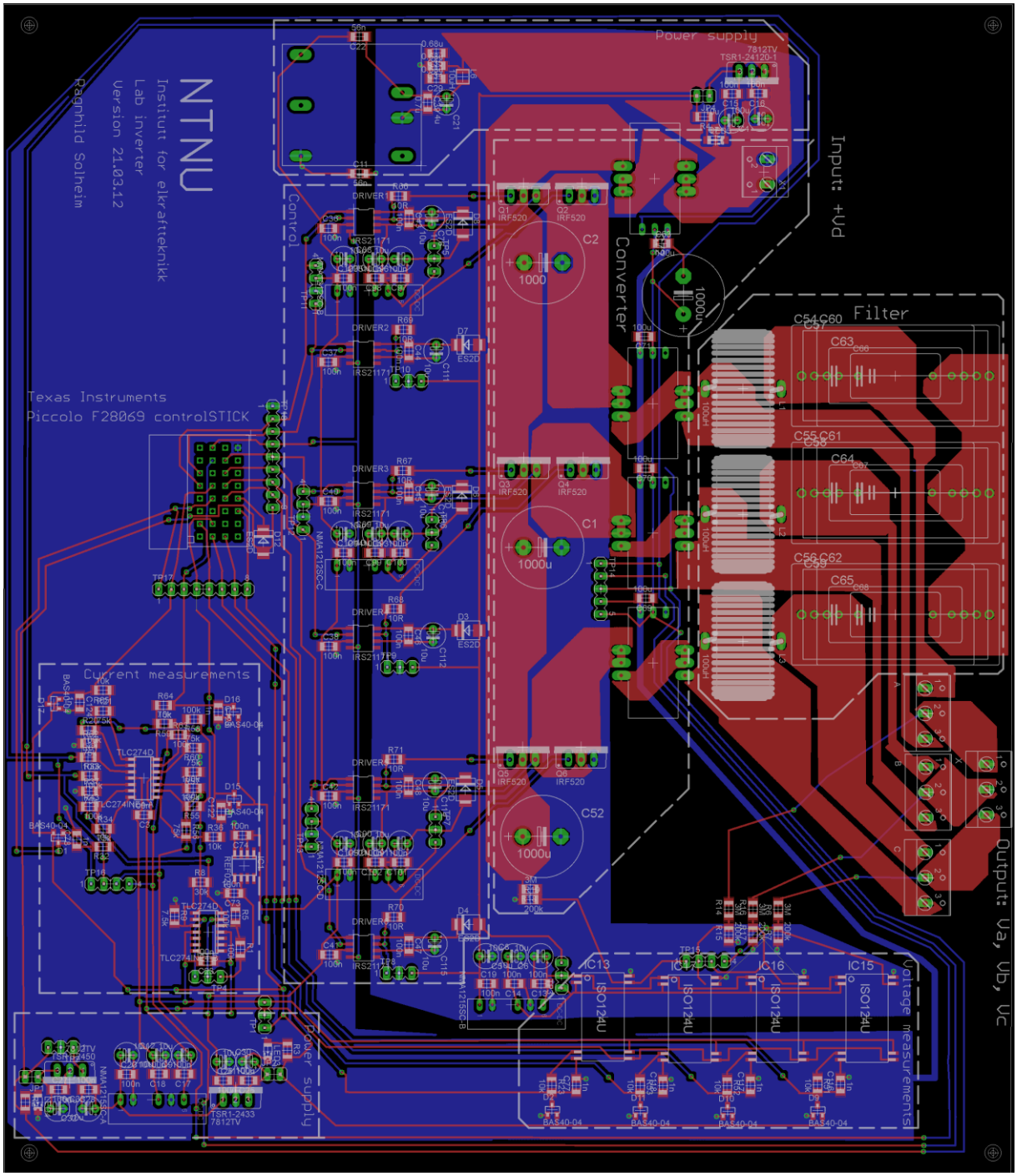


Figure A-6: Board Layout

APPENDIX B: PCB VERSION II:

B-1: LIST OF COMPONENTS

TABLE B-1: LIST OF COMPONENTS VERSION II

	COMPONENT	MANUFACTURER PART. NO.	ORDER CODE AT FARNELL	QUANTITY	REFERENCE
1	DC-DC converter	TSR 1-24120	1672130	1	(Traco Power, 2009)
2	DC-DC converter	NMA1215SC	1021441	2	(Murata Power Solutions, Inc., 2012)
3	DC-DC converter	NMA1212SC	1021439	5	(Murata Power Solutions, Inc., 2012)
6	DC-DC converter	SC001A2B91Z	2076930	1	(Lineage Power, 2009)
10	DC-DC converter	TSR 1-2433	1696319	1	(Traco Power, 2009)
11	DC-DC converter	TSR 1-2450	1696320	1	(Traco Power, 2009)
12	Inductor	744877100	1869664	1	(Würth Elektronik, 2005)
13	Voltage reference	REF02AU	1234885	1	(Burr-Brown Products from Texas Instruments, 1993)
14	Opamp	LMC660CM	9487115	2	(National Semiconductor, 2006)
15	Current transducer	LTS15-NP	1617409	4	(LEM, u.d.)
16	Driver 1	IRS21171	1925153	6	(International Rectifiers, B, 2009)
22	MOSFET	IRFB4110PBF	1436955	6	(International Rectifiers, 2011)
23	Isolated opamp	ISO124D	1544012	4	(Burr-Brown Products from Texas Instruments, 1997)

TABLE B-2: TEST PINS VERSION II

Test pin	Connection	Test pin	Connection
TP1-1	GND1	TP14-1	PWM-3B
TP1-2	+UB	TP14-2	GND2
TP1-3	+12V_GND1	TP14-3	+12V_GND2
TP2-1	GND1	TP14-4	PWM-3A
TP2-2	+15V_GND1	TP15-1	PWM_MOSFET6
TP2-3	-15V_GND1	TP15-2	GND1
TP2-4	+12V_GND1	TP15-3	+12V_GND1
TP3-1	PWM_MOSFET1	TP16-1	PWM_MOSFET4
TP3-2	12V_DRIVE1	TP16-2	GND1
TP3-3	GND1	TP16-3	+12V_GND1
TP3-4	S_MOSFET1	TP17-1	PWM_MOSFET2
TP4-1	PWM_MOSFET3	TP17-2	GND1
TP4-2	12V_DRIVE3	TP17-3	+12V_GND1
TP4-3	GND1	TP18-1	V_C_M
TP4-4	S_MOSFET3	TP18-2	GND1
TP5-1	PWM_MOSFET5	TP18-3	V_B_M
TP5-2	12V_DRIVE5	TP18-4	V_IN_M
TP5-3	GND1	TP18-5	V_A_M
TP5-4	S_MOSFET5	TP19-1	GND2
TP6-1	GND2	TP19-2	V_C_M_DIG
TP6-2	GND2	TP19-3	V_B_M_DIG
TP6-3	+12V_GND2	TP19-4	V_A_M_DIG
TP7-1	+12V_GND2	TP19-5	V_IN_M_DIG
TP7-2	-15V_GND2	TP21-1	GND2
TP7-3	+15V_GND2	TP21-2	I_A_M
TP7-4	GND2	TP21-3	I_IN_M
TP8-1	GND2	TP22-1	QEP-B
TP8-2	+12V_GND2	TP22-2	+3V3_GND2
TP8-3	+5V_GND2	TP22-3	QEP-A
TP9-1	GND2	TP22-4	QEP-1
TP9-2	+12V_GND2	TP22-5	GND2
TP9-3	+3V3_GND2	TP23-1	-VCC_SIGNALSCALING_OPAMP
TP10-1	GND2	TP23-2	+VCC_SIGNALSCALING_OPAMP
TP10-2	REF_2V5_GND2	TP23-3	GND2
TP10-3	REF_0V5_GND2	TP24-1	I_B_M
TP11-1	GND2	TP24-2	GND2
TP11-2	-VSS_VOLTREF_OPAMP	TP24-3	I_C_M
TP11-3	+VSS_VOLTREF_OPAMP		
TP11-4	+12V_GND2		
TP12-1	PWM-1B		
TP12-2	GND2		

TP12-3	+12V_GND2		
TP12-4	PWM-1A		
TP13-1	PWM-2B		
TP13-2	GND2		
TP13-3	+12V_GND2		
TP13-4	PWM-2A		

B-2: SCHEMATICS

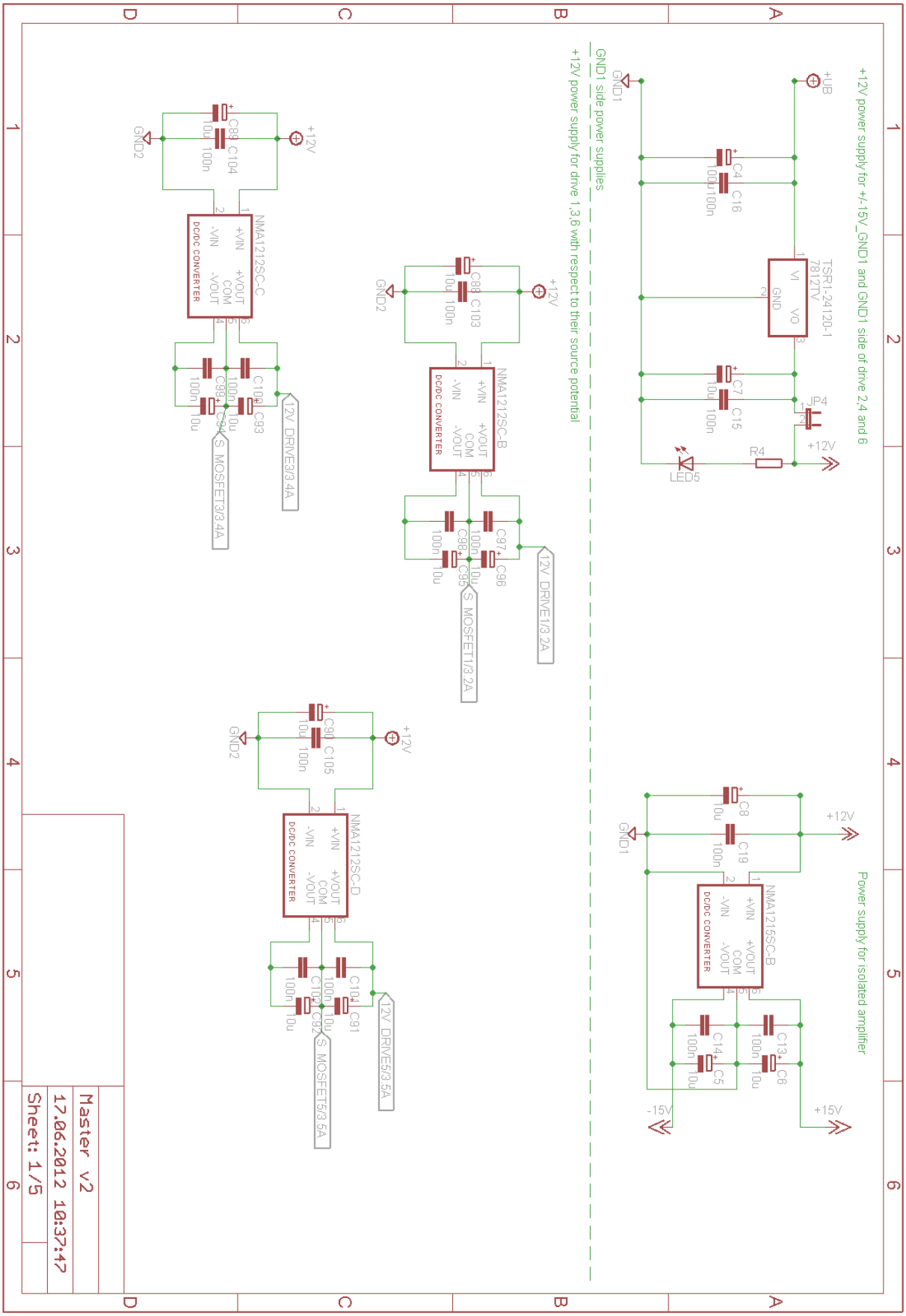


FIGURE B-1: SHEET 1

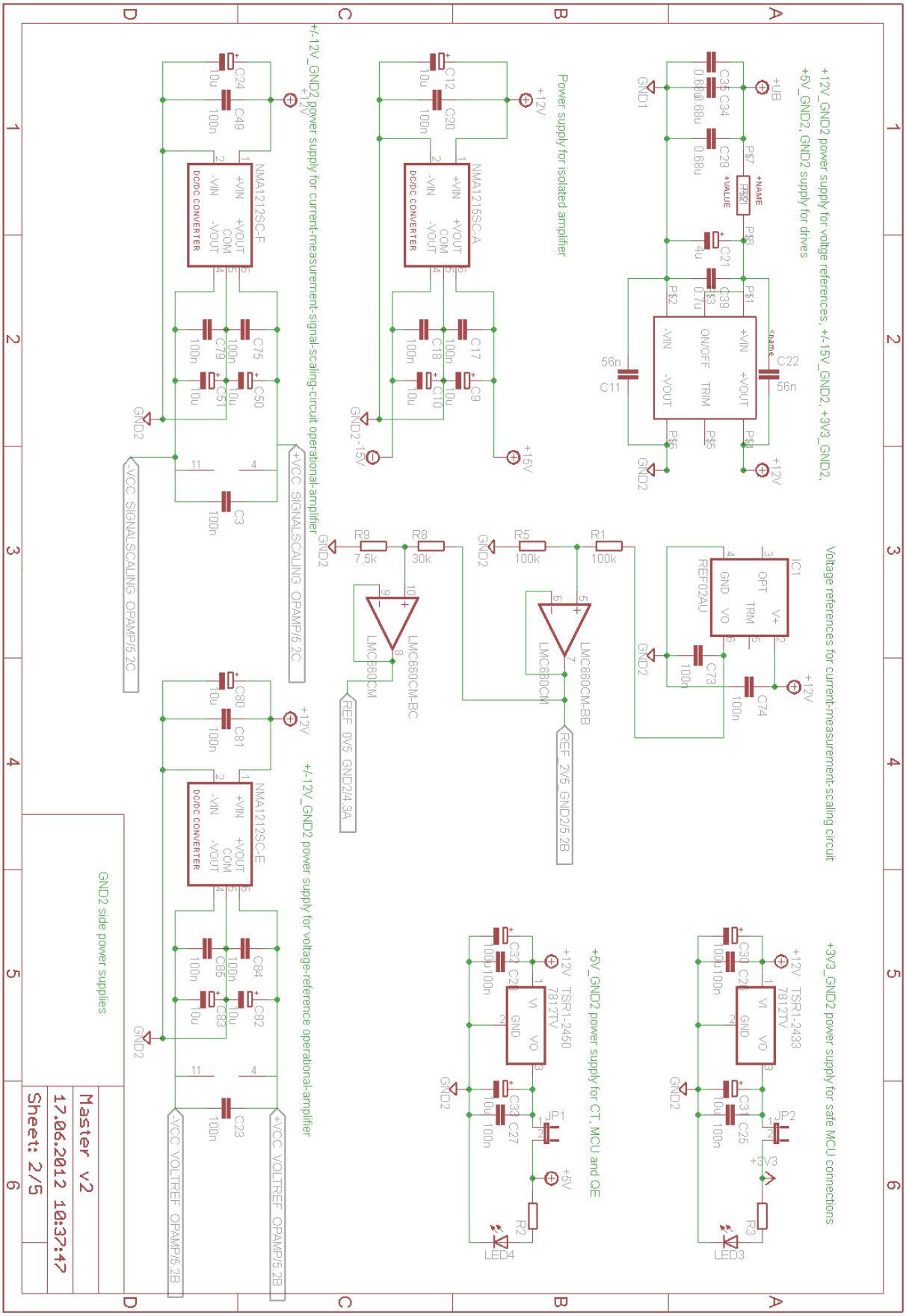


FIGURE B-2: SHEET 2

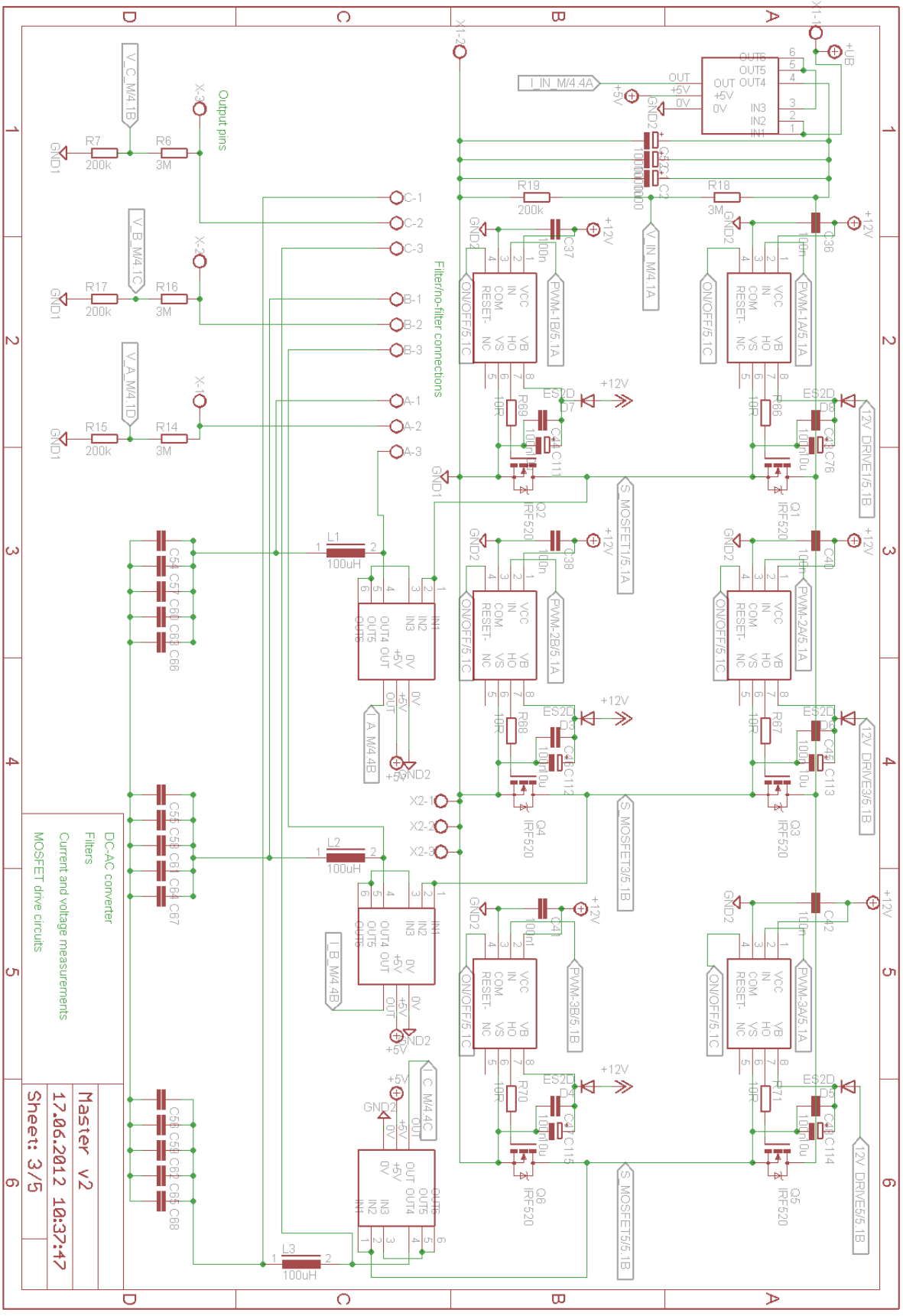


FIGURE B-3: SHEET 3

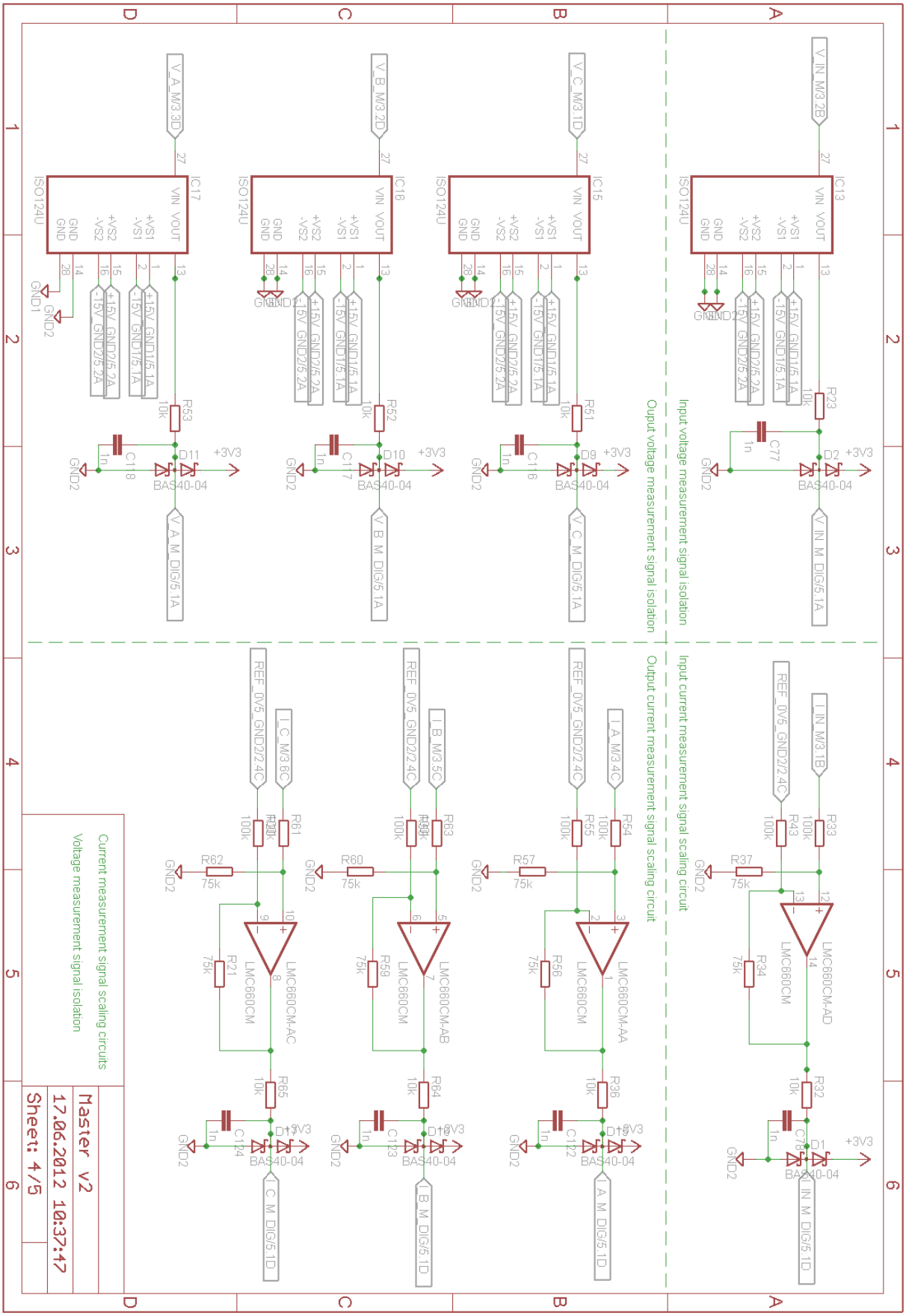


FIGURE B-4: SHEET 4

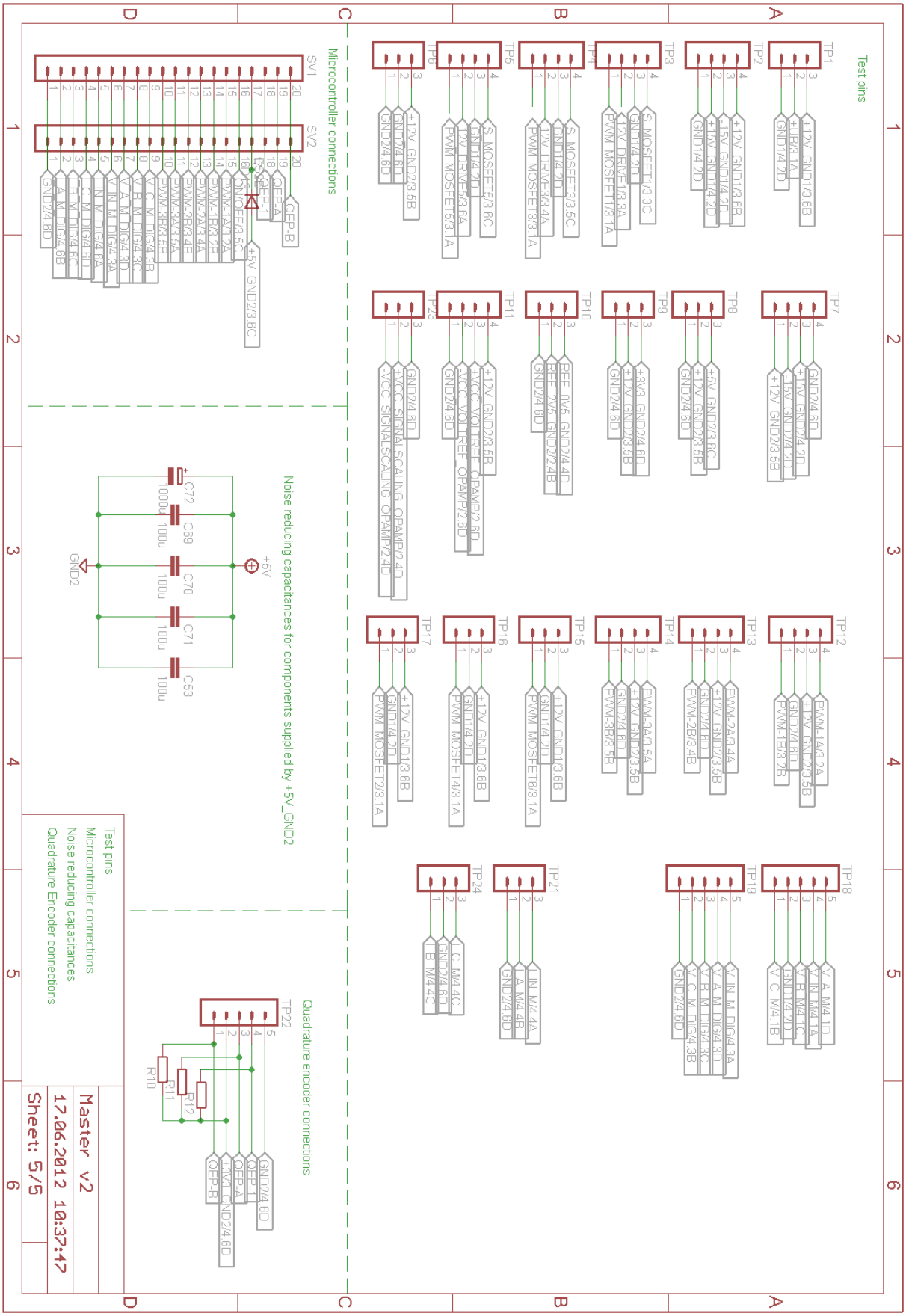


FIGURE B-5: SHEET 5

B-3: Layout

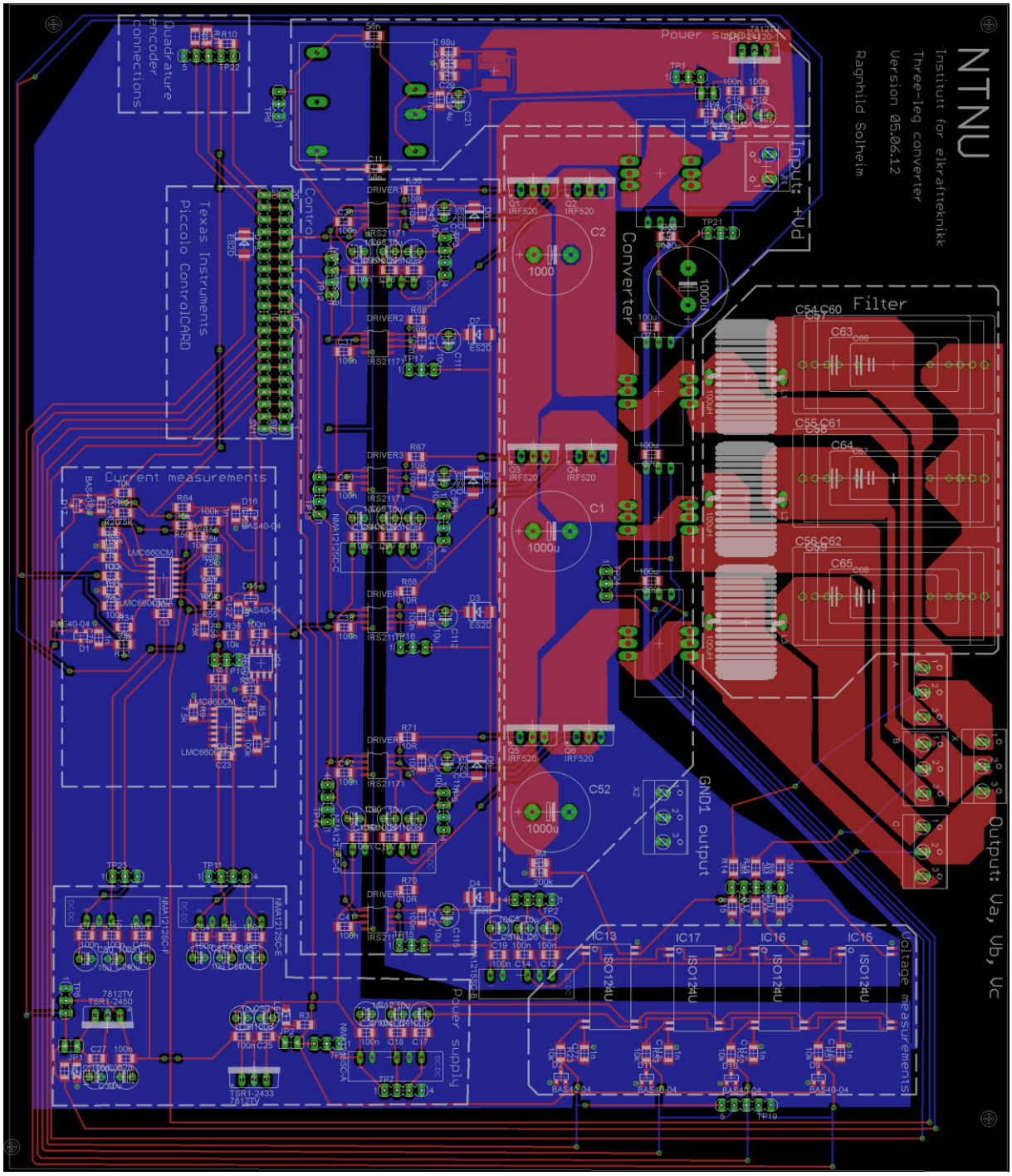


Figure B-6: Board Layout

APPENDIX C: GENERAL CONVERTER MICROCONTROLLER CODE

C-1: MAIN FILE

```
/**********************************
File name: converter_main-F2806x_1.c
Purpose: Control a DC motor drive
***********************************************/

#include "F2806x_Device.h"


void DeviceInit(void);  // defined in DC_md_DevInit_F2806x.c
void InitEPWMs(void);   // defined in DC_md_PWM_Setup.c
void InitADC(void);           // defined in DC_md_Adc.c

//define interrupt service routines
interrupt void ADCINT1_ISR(void);

Uint32 EndlessLoopCounter = 0;
int AdcIntCounter = 0;

//ADC ISR parameters
Uint16 AdcBuf[50][8]={0};




void main(void)
{

// Initialize System Control
// Enable Peripheral Clocks, GPIO
// Initialize the PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
 DeviceInit();
 InitEPWMs();  //Initialize PWM modules
 InitADC();           // Initialize ADC


 //Enable the interrupt
 EALLOW;
       PieVectTable.ADCINT1=&ADCINT1_ISR;
 EDIS;

 PieCtrlRegs.PIEIER1.bit.INTx1=1;
 IER |= M_INT1;
 EINT;

  for(;;)  //infinite loop
  {
    asm(" NOP");
    if (EndlessLoopCounter++ >= 4294967295)
    {
       EndlessLoopCounter=0;
    }
  }

}// end of main
```

```
interrupt void ADCINT1_ISR(void)
{
        static int index = 0;

        //Store the converted values in a circular array:
        AdcBuf[index][0]=AdcResult.ADCRESULT0;
        AdcBuf[index][1]=AdcResult.ADCRESULT1;
        AdcBuf[index][2]=AdcResult.ADCRESULT2;
        AdcBuf[index][3]=AdcResult.ADCRESULT3;
        AdcBuf[index][4]=AdcResult.ADCRESULT4;
        AdcBuf[index][5]=AdcResult.ADCRESULT5;
        AdcBuf[index][6]=AdcResult.ADCRESULT6;
        AdcBuf[index][7]=AdcResult.ADCRESULT7;

        index ++;
        if (index == 50) index =0;

        static int indexB = 1;
        indexB ++;



        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;             // Must acknowledge the PIE group
        AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;               // Clear ADCINT1 flag to enable
further interrupts

        //ADC interrupt counter
          AdcIntCounter++;
      if (AdcIntCounter >=200)
        AdcIntCounter=0;

} //end of ADCINT1_ISR
```

C-2: DEVICE INITIALIZATION

```
//---------------------------------------------------------------------------
//      FILE:                   converter_DevInit_F2806x.c
//      Description:   Device initialization specific to an Application
//      Version:                1.0
//  Target:             TMS320F2806x family
//      Type:                   Device Dependent
//  Copyright       ELK21 course 2011
//  Date:           June 2012
//---------------------------------------------------------------------------

#include "F2806x_Device.h"   // DSP2860x Headerfile Include File

// Functions that will be run from RAM need to be assigned to
// a different section.  This section will then be mapped to a load and
// run address using the linker cmd file.

#pragma CODE_SECTION(InitFlash, "ramfuncs");
#define Device_cal (void   (*)(void))0x3D7C80

void DeviceInit(void);       // peripheral clock enables and GPIO setups
void PieCntlInit(void);      // initializes the PIE control registers to a known state.
void PieVectTableInit(void); // initialize vector Table
void WDogDisable(void);      // disable watchdog timer
void PLLset(Uint16);         // set the CLKIN to CPU
void ISR_ILLEGAL(void);

//---------------------------------------------------------------------
//  Configure Device for target Application Here
//---------------------------------------------------------------------

void DeviceInit(void)
{
       WDogDisable();        // Disable the watchdog initially
       DINT;                 // Global Disable all Interrupts
       IER = 0x0000;  // Disable CPU interrupts
       IFR = 0x0000;  // Clear all CPU interrupt flags


// Select Internal Oscillator 1 as Clock Source (default),
// and turn off all unused clocks to conserve power.
// Refer to IntOsc1Sel function in F2806x_SysCtrl.c

       EALLOW;
       SysCtrlRegs.CLKCTL.bit.INTOSC1OFF = 0;   // Internal Oscillator 1 Off Bit. 0=ON
8default)
    SysCtrlRegs.CLKCTL.bit.OSCCLKSRCSEL=0;  // Clk Src = INTOSC1
       SysCtrlRegs.CLKCTL.bit.XCLKINOFF=1;     // Turn off XCLKIN
       SysCtrlRegs.CLKCTL.bit.XTALOSCOFF=1;    // Turn off XTALOSC
       SysCtrlRegs.CLKCTL.bit.INTOSC2OFF=1;    // Turn off INTOSC2
    EDIS;


// SYSTEM CLOCK speed based on internal oscillator = 10 MHz
// Refer F2806x Technical reference manual table 1-24 page 84
// Table 1-24 PLL setings

// 0x10=  80    MHz           (16)
// 0xF =  75    MHz           (15)
// 0xE =  70    MHz           (14)
// 0xD =  65    MHz           (13)
// 0xC =  60   MHz            (12)
// 0xB =  55   MHz            (11)
// 0xA =  50   MHz            (10)
// 0x9 =  45   MHz            (9)
// 0x8 =  40   MHz            (8)
// 0x7 =  35   MHz            (7)
// 0x6 =  30   MHz            (6)
// 0x5 =  25   MHz            (5)
// 0x4 =  20   MHz            (4)
// 0x3 =  15   MHz            (3)
// 0x2 =  10   MHz            (2)
```

```
// CLKIN=OSCCLK*PLLCR[DIV]*PLLSTS[DIVSEL]= 10MHz * 16)/2 = 80 MHz

        PLLset( 0x10 );       // the value of 0x10 makes CLKIN to CPU = 80 MHz
                              // PLLSet is within this file
                              // It is a modification of InitPll(DSP28_PLLCR,DSP28_DIVSEL)
                              // defined in F2806x_SysCtrl.c

// Initialise interrupt controller and Vector Table
// to defaults for now. Application ISR mapping done later.

    PieCntlInit();
    PieVectTableInit();

   EALLOW; // below registers are "protected", allow access.

// LOSPCP - LOW SPEED CLOCKS prescale register settings
// Refer F2806x Technical Reference manual table 1-19 page 75

// LOSPCP prescale register settings, normally it will be set to default values
// reset default is SYSCLKOUT/4

   SysCtrlRegs.LOSPCP.all = 0x0002;         // Sysclk / 4 (20 MHz)

// XCLKOUT to SYSCLKOUT ratio.  By default XCLKOUT = 1/4 SYSCLKOUT
// Refer F2806x Technical Reference manual Figure 1-35 page 95
   SysCtrlRegs.XCLK.bit.XCLKOUTDIV=2;   // XCLKOUT=SYSCLKOUT

   // PERIPHERAL CLOCK ENABLES
//---------------------------------------------------
// If you are not using a peripheral you may want to switch
// the clock off to save power, i.e. set to = 0.
// Value of 1 means the clock is ON
//
// Note: not all peripherals are available on all 280x derivates.
// Refer to the datasheet for your particular device.

//*** Refer to  BlinkingLED-DevInit_F2806x.c from
// C:\TI\controlSUITE\development_kits\F28069 controlSTICK\Timer - BlinkingLED
// also refer to F2806x TRM pages 70-74
// for the clocks refer F2806x TRM figure 1-13 page 69
// From TRM, Read the default clock setting (enabled or disabled)
// For example, CPUTIMERxENCLK (x=1-3) are enabled at reset. Refer figue 1-17 page 74

   SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;    // ADC
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR3.bit.COMP1ENCLK = 0;  // COMP1
   SysCtrlRegs.PCLKCR3.bit.COMP2ENCLK = 0;  // COMP2
   SysCtrlRegs.PCLKCR3.bit.COMP3ENCLK = 0;  // COMP3
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR0.bit.I2CAENCLK = 0;   // I2C
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR0.bit.SPIAENCLK = 0;   // SPI-A
   SysCtrlRegs.PCLKCR0.bit.SPIBENCLK = 0;   //SPI-B
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR0.bit.MCBSPAENCLK = 0; //McBSP-A
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR0.bit.SCIAENCLK = 0;   // SCI-A
   SysCtrlRegs.PCLKCR0.bit.SCIBENCLK = 0;   //SCI-B
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR1.bit.EQEP1ENCLK = 1;    // eQEP1
   SysCtrlRegs.PCLKCR1.bit.EQEP2ENCLK = 1;    // eQEP2
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR1.bit.ECAP1ENCLK = 0;  //eCAP1
   SysCtrlRegs.PCLKCR1.bit.ECAP2ENCLK = 0;  // eCAP2
   SysCtrlRegs.PCLKCR1.bit.ECAP3ENCLK = 0;  // eCAP3
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR1.bit.EPWM1ENCLK = 1;  // ePWM1
   SysCtrlRegs.PCLKCR1.bit.EPWM2ENCLK = 1;  // ePWM2
   SysCtrlRegs.PCLKCR1.bit.EPWM3ENCLK = 1;  // ePWM3
   SysCtrlRegs.PCLKCR1.bit.EPWM4ENCLK = 1;  // ePWM4
   SysCtrlRegs.PCLKCR1.bit.EPWM5ENCLK = 0;  // ePWM5
   SysCtrlRegs.PCLKCR1.bit.EPWM6ENCLK = 0;  // ePWM6
   SysCtrlRegs.PCLKCR1.bit.EPWM7ENCLK = 0;  // ePWM7
   SysCtrlRegs.PCLKCR1.bit.EPWM8ENCLK = 0;  // ePWM8
   //--------------------------------------------------
   SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;   // Enable TBCLK
   //--------------------------------------------------
```

```
   SysCtrlRegs.PCLKCR3.bit.DMAENCLK = 0;     // DMA
   //-----------------------------------------------
   SysCtrlRegs.PCLKCR3.bit.CLA1ENCLK = 0;    // CLA
   //-----------------------------------------------

  //-------------------------------------------------------------------------------------------
// GPIO (GENERAL PURPOSE I/O) CONFIG
//-------------------------------------------------------------------------------------------
//------------------------
// QUICK NOTES on USAGE:
//------------------------
// If GpioCtrlRegs.GP?MUX?bit.GPIO?= 1, 2 or 3 (i.e. Non GPIO func), then leave
//     rest of lines commented
// If GpioCtrlRegs.GP?MUX?bit.GPIO?= 0 (i.e. GPIO func), then:
//     1) uncomment GpioCtrlRegs.GP?DIR.bit.GPIO? = ? and choose pin to be IN or OUT
//     2) If IN, can leave next to lines commented
//     3) If OUT, uncomment line with ..GPACLEAR.. to force pin LOW or
//                      uncomment line with ..GPASET.. to force pin HIGH or
//-------------------------------------------------------------------------------------------
// The following has included all GPIO for F28069 80 PINS (controlSTICK USB)
// for pins names refer F2806x datasheet Figure 3.8 page 26
// Also F2806x datasheet table 3-6 page 31
//-------------------------------------------------------------------------------------------
//  GPIO-00 - PIN FUNCTION = --Spare-- PIN 69
       GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;         // 0=GPIO, 1=EPWM1A, 2=Resv, 3=Resv
       //GpioCtrlRegs.GPADIR.bit.GPIO0 = 0;        // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO0 = 1; // uncomment if --> Set Low initially
       //GpioDataRegs.GPASET.bit.GPIO0 = 1;        // uncomment if --> Set High initially
//-------------------------------------------------------------------------------------------
//  GPIO-01 - PIN FUNCTION = --Spare--PIN 68
       GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1;         // 0=GPIO, 1=EPWM1B, 2=rsvd, 3=COMP1OUT
//     GpioCtrlRegs.GPADIR.bit.GPIO1 = 0;          // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO1 = 1; // uncomment if --> Set Low initially
//     GpioDataRegs.GPASET.bit.GPIO1 = 1;          // uncomment if --> Set High initially
//-------------------------------------------------------------------------------------------
//  GPIO-02 - PIN FUNCTION = --Spare-- PIN 67
       GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1;         // 0=GPIO, 1=EPWM2A, 2=Resv, 3=Resv
       //GpioCtrlRegs.GPADIR.bit.GPIO2 = 0;        // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO2 = 1; // uncomment if --> Set Low initially
//     GpioDataRegs.GPASET.bit.GPIO2 = 1;          // uncomment if --> Set High initially
//-------------------------------------------------------------------------------------------
//  GPIO-03 - PIN FUNCTION = --Spare-- PIN 66
       GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 1;         // 0=GPIO, 1=EPWM2B, 2=SPISOMIA,
3=COMP2OUT
//     GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;          // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO3 = 1; // uncomment if --> Set Low initially
//     GpioDataRegs.GPASET.bit.GPIO3 = 1;          // uncomment if --> Set High initially
//-------------------------------------------------------------------------------------------
//  GPIO-04 - PIN FUNCTION = --Spare-- PIN 07
       GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1;         // 0=GPIO, 1=EPWM3A, 2=Resv, 3=Resv
       GpioCtrlRegs.GPADIR.bit.GPIO4 = 0;          // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO4 = 1; // uncomment if --> Set Low initially
//     GpioDataRegs.GPASET.bit.GPIO4 = 1;          // uncomment if --> Set High initially
//-------------------------------------------------------------------------------------------
//  GPIO-05 - PIN FUNCTION = --Spare-- PIN 08
       GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 1;         // 0=GPIO, 1=EPWM3B, 2=SPISIMOA, 3=ECAP1
//     GpioCtrlRegs.GPADIR.bit.GPIO5 = 0;          // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO5 = 1; // uncomment if --> Set Low initially
//     GpioDataRegs.GPASET.bit.GPIO5 = 1;          // uncomment if --> Set High initially
//-------------------------------------------------------------------------------------------
//  GPIO-06 - PIN FUNCTION = --Spare-- PIN 46
       GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 1;         // 0=GPIO, 1=EPWM4A, 2=EPWMSYNCI,
3=EPWMSYNCO
       GpioCtrlRegs.GPADIR.bit.GPIO6 = 0;          // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO6 = 1; // uncomment if --> Set Low initially
//     GpioDataRegs.GPASET.bit.GPIO6 = 1;          // uncomment if --> Set High initially
//-------------------------------------------------------------------------------------------
//  GPIO-07 - PIN FUNCTION = --Spare-- PIN 45
       GpioCtrlRegs.GPAMUX1.bit.GPIO7 = 1;         // 0=GPIO, 1=EPWM4B, 2=SCIRXDA, 3=ECAP2
       GpioCtrlRegs.GPADIR.bit.GPIO7 = 0;          // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO7 = 1; // uncomment if --> Set Low initially
//     GpioDataRegs.GPASET.bit.GPIO7 = 1;          // uncomment if --> Set High initially
//-------------------------------------------------------------------------------------------
//  GPIO-08 - PIN FUNCTION = --PIN 43      // 0=GPIO,  1=EPWM5A,  2=Resv,  3=ADCSOCA0
       GpioCtrlRegs.GPAMUX1.bit.GPIO8 = 0;
       GpioCtrlRegs.GPADIR.bit.GPIO8 = 0;          // 1=OUTput,  0=INput
//     GpioDataRegs.GPACLEAR.bit.GPIO8 = 1; // uncomment if --> Set Low initially
```

```
//      GpioDataRegs.GPASET.bit.GPIO8 = 1;          // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-09 - PIN FUNCTION = --PIN 49      // 0=GPIO9, 1=EPWM5B, 2=SCITXDB, 3=ECAP3
        GpioCtrlRegs.GPAMUX1.bit.GPIO9 = 0;
        GpioCtrlRegs.GPADIR.bit.GPIO9 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO9 = 1; // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO9 = 1;          // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-10 - PIN FUNCTION = --PIN 60      // 0=GPIO,  1=EPWM6A,  2=Resv,  3=ADCSOCB0
  GpioCtrlRegs.GPAMUX1.bit.GPIO10 = 0;
  GpioCtrlRegs.GPADIR.bit.GPIO10 = 0;               // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO10 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-11 - PIN FUNCTION = --PIN 59     // 0=GPIO11, 1=EPWM6B, 2=SCIRXDB, 3=ECAP1
   GpioCtrlRegs.GPAMUX1.bit.GPIO11 = 0;
   GpioCtrlRegs.GPADIR.bit.GPIO11 = 0;              // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO11 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-12 - PIN FUNCTION = --Spare--
        GpioCtrlRegs.GPAMUX1.bit.GPIO12 = 0; // 0=GPIO, 1=TZ1n, 2=SCITXDA, 3=SPISIMOB
        GpioCtrlRegs.GPADIR.bit.GPIO12 = 0;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO12 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO12 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-13 - PIN FUNCTION = --Spare-- PIN 75  // 0=GPIO,  1=TZ2,  2=Resv,  3=SPISOMIB
   GpioCtrlRegs.GPAMUX1.bit.GPIO13 = 0;
        GpioCtrlRegs.GPADIR.bit.GPIO13 = 0;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO13 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO13 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//  GPIO-14 - PIN FUNCTION = --Spare-- PIN 76  // 0=GPIO,  1=TZ3,  2=SCITXDB,  3=SPICLKB
        GpioCtrlRegs.GPAMUX1.bit.GPIO14 = 0;
        GpioCtrlRegs.GPADIR.bit.GPIO14 = 0;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO14 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO14 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//  GPIO-15 - PIN FUNCTION = --Spare-- PIN 70  // 0=GPIO,  1=ECAP2,  2=SCIRXDB,  3=SPISTEB
    GpioCtrlRegs.GPAMUX1.bit.GPIO15 = 0;
        GpioCtrlRegs.GPADIR.bit.GPIO15 = 0;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO15 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO15 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-16 - PIN FUNCTION = --Spare-- PIN 44
        GpioCtrlRegs.GPAMUX2.bit.GPIO16 = 0; // 0=GPIO, 1=SPISIMOA, 2=Resv  3=TZ2n
        GpioCtrlRegs.GPADIR.bit.GPIO16 = 0;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO16 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO16 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-17 - PIN FUNCTION = --Spare-- PIN 42
        GpioCtrlRegs.GPAMUX2.bit.GPIO17 = 0; // 0=GPIO, 1=SPISOMIA, 2=Resv  3=TZ3n
        GpioCtrlRegs.GPADIR.bit.GPIO17 = 0;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO17 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO17 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-18 - PIN FUNCTION = --Spare-- PIN 41
        GpioCtrlRegs.GPAMUX2.bit.GPIO18 = 0; // 0=GPIO, 1=SPICLKA, 2=SCITXDB, 3=XCLKOUT
        GpioCtrlRegs.GPADIR.bit.GPIO18 = 1;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;       // uncomment if --> Set Low initially
        GpioDataRegs.GPASET.bit.GPIO18 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-19 - PIN FUNCTION = --Spare-- PIN 52
        GpioCtrlRegs.GPAMUX2.bit.GPIO19 = 0; // 0=GPIO, 1=SPISTEA, 2=SCIRXDB, 3=ECAP1
        GpioCtrlRegs.GPADIR.bit.GPIO19 = 0;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO19 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-20 - PIN FUNCTION = --PIN 05
   GpioCtrlRegs.GPAMUX2.bit.GPIO20 = 1;         // 0=GPIO,  1=EQEP1A,  2=MDXA,  3=COMP1OUT
        GpioCtrlRegs.GPADIR.bit.GPIO20 = 0;         // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO20 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO20 = 1;         // uncomment if --> Set High initially
//--------------------------------------------------------------------------------
//  GPIO-21 - PIN FUNCTION = --PIN 06
```

```
   GpioCtrlRegs.GPAMUX2.bit.GPIO21 = 1;     // 0=GPIO,  1=EQEP1B,  2=MDRA,  3=COMP2OUT
   GpioCtrlRegs.GPADIR.bit.GPIO21 = 0;              // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO21 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO21 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
//  GPIO-22 - PIN FUNCTION = --PIN 78
    GpioCtrlRegs.GPAMUX2.bit.GPIO22 = 1;     // 0=GPIO,  1=EQEP1S,  2=MCLKXA,  3=SCITXDB
       GpioCtrlRegs.GPADIR.bit.GPIO22 = 0;          // 1=OUTput,  0=INput
//GpioDataRegs.GPACLEAR.bit.GPIO22 = 1;     // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO22 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------

//  GPIO-23 - PIN FUNCTION = --PIN 01
  GpioCtrlRegs.GPAMUX2.bit.GPIO23 = 1;       // 0=GPIO,  1=EQEP1I,  2=MFSXA,  3=SCIRXDB
  GpioCtrlRegs.GPADIR.bit.GPIO23 = 0;               // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO23 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO23 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
//  GPIO-24 - PIN FUNCTION = --PIN 77   // ** special for PN device
    GpioCtrlRegs.GPAMUX2.bit.GPIO24 = 0;     // 0=GPIO,  1=ECAP1,  2=rsvd,  3=SPISIMOB
       GpioCtrlRegs.GPADIR.bit.GPIO24 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO24 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO24 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
   // GPIO-25 - PIN FUNCTION  = PIN 31      // ** special for PN device
       GpioCtrlRegs.GPAMUX2.bit.GPIO25 = 0; // 0=GPIO,  1=ECAP2,  2=rsvd,  3=SPISOMIB
       GpioCtrlRegs.GPADIR.bit.GPIO25 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO25 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO25 = 1;         // uncomment if --> Set High initially
//====================
//  GPIO-26 - PIN FUNCTION = --PIN 62       // ** special for PN device
    GpioCtrlRegs.GPAMUX2.bit.GPIO26 = 0;     // 0=GPIO,  1=ECAP3,  2=rsvd,  3=SPICLKB
       GpioCtrlRegs.GPADIR.bit.GPIO26 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO26 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO26 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
//  GPIO-27 - PIN FUNCTION = --PIN 61  //   ** special for PN device
   GpioCtrlRegs.GPAMUX2.bit.GPIO27 = 0;      // 0=GPIO,  1=HRCAP2,  2=rsvd,  3=SPISTEB
       GpioCtrlRegs.GPADIR.bit.GPIO27 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO27 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO27 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
//  GPIO-28 - PIN FUNCTION = --PIN 40  //
       GpioCtrlRegs.GPAMUX2.bit.GPIO28 = 0; // 0=GPIO,  1=SCIRXDA,  2=SDAA,  3=TZ2
       GpioCtrlRegs.GPADIR.bit.GPIO28 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO28 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO28 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
  //GPIO-29 - PIN FUNCTION = --PIN 34
       GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 0; // 0=GPIO,  1=SCITXDA,  2=SCLA,  3=TZ3
    GpioCtrlRegs.GPADIR.bit.GPIO29 = 0;              // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO29 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO29 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
//  GPIO-30 - PIN FUNCTION = --PIN 33        ** special for PN device
    GpioCtrlRegs.GPAMUX2.bit.GPIO30 = 0;     // 0=GPIO,  1=CANRXA,  2=rscvd,  3=EPWM7A
       GpioCtrlRegs.GPADIR.bit.GPIO30 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO30 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPASET.bit.GPIO30 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------
 //  GPIO-31 - PIN FUNCTION = --PIN 32        ** special for PN device
    GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 0;     // 0=GPIO,  1=CANTXA,  2=rsvd,  3=EPWM8A
       GpioCtrlRegs.GPADIR.bit.GPIO31 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPACLEAR.bit.GPIO31 = 1;       // uncomment if --> Set Low initially
       GpioDataRegs.GPASET.bit.GPIO31 = 0;          // uncomment if --> Set High initially
       //-------------------------------------------------------------------------------
//  GPIO-32 - PIN FUNCTION = --Spare-- PIN 79
       GpioCtrlRegs.GPBMUX1.bit.GPIO32 = 0; // 0=GPIO,  1=I2C-SDA,  2=EPWMSYNCI,  3=ADCSOCA0
       GpioCtrlRegs.GPBDIR.bit.GPIO32 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPBCLEAR.bit.GPIO32 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPBSET.bit.GPIO32 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
//  GPIO-33 - PIN FUNCTION = --Spare-- PIN 80
       GpioCtrlRegs.GPBMUX1.bit.GPIO33 = 0; // 0=GPIO,  1=I2C-SCLA,  2=EPWMSYNCO,  3=ADCSOCB0
       GpioCtrlRegs.GPBDIR.bit.GPIO33 = 0;          // 1=OUTput,  0=INput
//      GpioDataRegs.GPBCLEAR.bit.GPIO33 = 1;       // uncomment if --> Set Low initially
//      GpioDataRegs.GPBSET.bit.GPIO33 = 1;         // uncomment if --> Set High initially
```

```
//---------------------------------------------------------------------------------------
//  GPIO-34 - PIN FUNCTION = PIN 55 = LED for F2806x controlSTICK
       GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0; // 0=GPIO,  1=COMP2OUT,  2=Resv,  3=COMP3OUT
       GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;         // 1=OUTput,  0=INput
//     GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;       // uncomment if --> Set Low initially
       GpioDataRegs.GPBSET.bit.GPIO34 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
// GPIO 35-38 are defaulted to JTAG usage, and are not shown here to enforce JTAG debug
// usage.
//---------------------------------------------------------------------------------------
//  GPIO-39 - PIN FUNCTION = --PIN 53       // GPIO39
    GpioCtrlRegs.GPBMUX1.bit.GPIO39 = 0;    // 0=GPIO,  1=Resv,  2=Resv,  3=Resv
       GpioCtrlRegs.GPBDIR.bit.GPIO39 = 0;         // 1=OUTput,  0=INput
//     GpioDataRegs.GPBCLEAR.bit.GPIO39 = 1;       // uncomment if --> Set Low initially
//     GpioDataRegs.GPBSET.bit.GPIO39 = 1;         // uncomment if --> Set High initially
//---------------------------------------------------------------------------------------
//  GPIO-40 - GPIO-58 does not exist for 80 pin device
       EDIS;   // Disable register access
}

//===========================================================================
// NOTE:
// IN MOST APPLICATIONS THE FUNCTIONS AFTER THIS POINT CAN BE LEFT UNCHANGED
// THE USER NEED NOT REALLY UNDERSTAND THE BELOW CODE TO SUCCESSFULLY RUN THIS
// APPLICATION.
//===========================================================================

// disable watchdog - refer F2806x TRM Figure 35 and table 40 page 62
void WDogDisable(void)
{
    EALLOW;
    SysCtrlRegs.WDCR= 0x0068; // Bits 0-2 WDPS - Watchdog pre-scale = 000 => WDCLK =
OSCCLK/512/1 (default)
    EDIS;                     // Bits 3-5 - 101 -
                              // Bit WDDIS - Watchdog disable - 1 is disable watchdog
}

// This function initializes the PLLCR register.
//void InitPll(Uint16 val, Uint16 clkindiv)
void PLLset(Uint16 val)
{
   volatile Uint16 iVol;

   // Make sure the PLL is not running in limp mode
   if (SysCtrlRegs.PLLSTS.bit.MCLKSTS != 0)
   {
         EALLOW;
      // OSCCLKSRC1 failure detected. PLL running in limp mode.
      // Re-enable missing clock logic.
      SysCtrlRegs.PLLSTS.bit.MCLKCLR = 1;
      EDIS;
      // Replace this line with a call to an appropriate
      // SystemShutdown(); function.
      asm("        ESTOP0");     // Uncomment for debugging purposes
   }

   // DIVSEL MUST be 0 before PLLCR can be changed from
   // 0x0000. It is set to 0 by an external reset XRSn
   // This puts us in 1/4
   if (SysCtrlRegs.PLLSTS.bit.DIVSEL != 0)
   {
       EALLOW;
       SysCtrlRegs.PLLSTS.bit.DIVSEL = 0;
       EDIS;
   }

   // Change the PLLCR
   if (SysCtrlRegs.PLLCR.bit.DIV != val)
   {

      EALLOW;
      // Before setting PLLCR turn off missing clock detect logic
      SysCtrlRegs.PLLSTS.bit.MCLKOFF = 1;
      SysCtrlRegs.PLLCR.bit.DIV = val;
      EDIS;

      // Optional: Wait for PLL to lock.
```

```
       // During this time the CPU will switch to OSCCLK/2 until
       // the PLL is stable.  Once the PLL is stable the CPU will
       // switch to the new PLL value.
       //
       // This time-to-lock is monitored by a PLL lock counter.
       //
       // Code is not required to sit and wait for the PLL to lock.
       // However, if the code does anything that is timing critical,
       // and requires the correct clock be locked, then it is best to
       // wait until this switching has completed.

       // Wait for the PLL lock bit to be set.
       // The watchdog should be disabled before this loop, or fed within
       // the loop via ServiceDog().

          // Uncomment to disable the watchdog
       WDogDisable();

          while(SysCtrlRegs.PLLSTS.bit.PLLLOCKS != 1) {}

       EALLOW;
       SysCtrlRegs.PLLSTS.bit.MCLKOFF = 0;
          EDIS;
        }

          //divide down SysClk by 2 to increase stability
        EALLOW;
        SysCtrlRegs.PLLSTS.bit.DIVSEL = 2;
        EDIS;
}

// This function initializes the PIE control registers to a known state.
//
void PieCntlInit(void)
{
    // Disable Interrupts at the CPU level:
    DINT;

    // Disable the PIE
    PieCtrlRegs.PIECTRL.bit.ENPIE = 0;

        // Clear all PIEIER registers:
        // PIE Interrupt Enable Registers
        PieCtrlRegs.PIEIER1.all = 0;
        PieCtrlRegs.PIEIER2.all = 0;
        PieCtrlRegs.PIEIER3.all = 0;
        PieCtrlRegs.PIEIER4.all = 0;
        PieCtrlRegs.PIEIER5.all = 0;
        PieCtrlRegs.PIEIER6.all = 0;
        PieCtrlRegs.PIEIER7.all = 0;
        PieCtrlRegs.PIEIER8.all = 0;
        PieCtrlRegs.PIEIER9.all = 0;
        PieCtrlRegs.PIEIER10.all = 0;
        PieCtrlRegs.PIEIER11.all = 0;
        PieCtrlRegs.PIEIER12.all = 0;

        // Clear all PIEIFR registers:
        // PIE Interrupt Flag Registers
        PieCtrlRegs.PIEIFR1.all = 0;
        PieCtrlRegs.PIEIFR2.all = 0;
        PieCtrlRegs.PIEIFR3.all = 0;
        PieCtrlRegs.PIEIFR4.all = 0;
        PieCtrlRegs.PIEIFR5.all = 0;
        PieCtrlRegs.PIEIFR6.all = 0;
        PieCtrlRegs.PIEIFR7.all = 0;
        PieCtrlRegs.PIEIFR8.all = 0;
        PieCtrlRegs.PIEIFR9.all = 0;
        PieCtrlRegs.PIEIFR10.all = 0;
        PieCtrlRegs.PIEIFR11.all = 0;
        PieCtrlRegs.PIEIFR12.all = 0;
}

void PieVectTableInit(void)
{
       int16 i;
       PINT *Dest = &PieVectTable.TINT1;
```

```
        EALLOW;
        for(i=0; i < 115; i++)
    *Dest++ = &ISR_ILLEGAL;
        EDIS;

        // Enable the PIE Vector Table
        PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
}

interrupt void ISR_ILLEGAL(void)   // Illegal operation TRAP
{
  // Insert ISR Code here

  // Next two lines for debug only to halt the processor here
  // Remove after inserting ISR Code
  asm("          ESTOP0");
  for(;;);

}

// This function initializes the Flash Control registers

//                   CAUTION
// This function MUST be executed out of RAM. Executing it
// out of OTP/Flash will yield unpredictable results

void InitFlash(void)
{
   EALLOW;
   //Enable Flash Pipeline mode to improve performance
   //of code executed from Flash.
   FlashRegs.FOPT.bit.ENPIPE = 1;

   //                CAUTION
   //Minimum waitstates required for the flash operating
   //at a given CPU rate must be characterized by TI.
   //Refer to the datasheet for the latest information.

   //Set the Paged Waitstate for the Flash
   FlashRegs.FBANKWAIT.bit.PAGEWAIT = 3;

   //Set the Random Waitstate for the Flash
   FlashRegs.FBANKWAIT.bit.RANDWAIT = 3;

   //Set the Waitstate for the OTP
   FlashRegs.FOTPWAIT.bit.OTPWAIT = 5;

   //                CAUTION
   //ONLY THE DEFAULT VALUE FOR THESE 2 REGISTERS SHOULD BE USED
   FlashRegs.FSTDBYWAIT.bit.STDBYWAIT = 0x01FF;
   FlashRegs.FACTIVEWAIT.bit.ACTIVEWAIT = 0x01FF;
   EDIS;

   //Force a pipeline flush to ensure that the write to
   //the last register configured occurs before returning.

   asm(" RPT #7 || NOP");
}


// This function will copy the specified memory contents from
// one location to another.
//
//      Uint16 *SourceAddr        Pointer to the first word to be moved
//                            SourceAddr < SourceEndAddr
//      Uint16* SourceEndAddr     Pointer to the last word to be moved
//      Uint16* DestAddr          Pointer to the first destination word
//
// No checks are made for invalid memory locations or that the
// end address is > then the first start address.

void MemCopy(Uint16 *SourceAddr, Uint16* SourceEndAddr, Uint16* DestAddr)
{
    while(SourceAddr < SourceEndAddr)
    {
       *DestAddr++ = *SourceAddr++;
    }
```

```
    return;
}

//============================================================================
// End of file.
//============================================================================
```

C-3: PWM INITIALIZATION

```
//###########################################################################
//      FILE:                   converter_PWM_Setup.c
//      Description:    PWM setup for full-bridge DC-DC converter
//      Version:                1.0
//  Target:             TMS320F2806x family
//      Type:                   Device Dependent
//  Copyright       ELK21 course 2011
//  Date:                       June 2012
//###########################################################################

#include "F2806x_Device.h"          // DSP2860x Headerfile Include File
#include "F2806x_EPWM_defines.h"     // Defines specifics to EPWM

#include "DSP28x_Project.h"                                             // Device Headerfile
and Examples Include File

#define TBCTLVAL  0x200E           // up-down count, timebase=SYSCLKOUT

void InitEPWMs(void)
{

 //--------------------------------------------------------------------
 //--- Must disable the clock to the ePWM modules if you
 //--- want all ePMW modules synchronized.
 //--------------------------------------------------------------------
       asm(" EALLOW");                                                  // Enable EALLOW protected
register access
       SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
       asm(" EDIS");                                          // Disable EALLOW protected
register access

//--------------------------------------------------------------------
//--- Configure ePWM1 for 50 kHz symmetric PWM on EPWM1A and EPWM1B pin
//    Also configure ePWM1 to trigger the ADC
//--------------------------------------------------------------------

       // Setup counter mode

       EPwm1Regs.TBCTL.bit.CTRMODE = TB_FREEZE;                    // Stop-freeze counter
operation until all bits are set
       EPwm1Regs.TBCTL.bit.FREE_SOFT=0x3;                                  //1x - Free run
       EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;                     // Disable phase
loading - for MASTER mode
       EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;                      //  reload PRD on
counter=0
       EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;             // sync-out disabled
       EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
       EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;

       EPwm1Regs.TBPRD = 800;                                                  // Set timer
period for up-and-down-count
       EPwm1Regs.TBCTR = 0x0000;                                               // Clear
counter

       EPwm1Regs.TBPHS.half.TBPHS = 0x0000;                        // Set Phase register to
zero
       EPwm1Regs.CMPA.half.CMPA=EPwm1Regs.TBPRD/2;                 // Initial duty ratio = 50 %

       // Setup shadowing
       EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;                 // shadow mode
       EPwm1Regs.CMPCTL.bit.LOADAMODE=CC_CTR_ZERO;                 // Load on Zero
       // Set actions- up-count
       EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;                        // PWM1A low (AQ_CLEAR) when
CMPA=TBCNT on down count (CAD)
       EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;                          // PWM1A high (AQ_SET) when
CMPA =TBCNT on up count (CAU)

       EPwm1Regs.AQCTLB.bit.CAD = AQ_SET;                          // PWM1B high (AQ_SET) when
CMPA = TBCNT on down count (CAD)
       EPwm1Regs.AQCTLB.bit.CAU = AQ_CLEAR;                        // PWM1B low (AQ_CLEAR) when
CMPA = TBCNT on up count (CAU)
```

```
        // set deadband, Trip-zone and PWM_chopper units
        EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;            // enable Dead-band module
        EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;                 // Active Hi complementary
        EPwm1Regs.DBFED = 200;                                                  // FED = 200
TBCLKs = 2.4us
        EPwm1Regs.DBRED = 200;                                                  // RED = 200
TBCLKs
        EPwm1Regs.PCCTL.bit.CHPEN = 0;                                          // PWM
chopper unit disabled
        EPwm1Regs.TZDCSEL.all = 0x0000;                                         // All trip
zone and DC compare actions disabled

    // Configure Event-Trigger Prescale Register (ETPS)
        // Refer table 3-64 TRM page 358
        EPwm1Regs.ETPS.all = 0x0000;                                    // Configure SOCA
        EPwm1Regs.ETPS.bit.SOCAPRD=ET_1ST;                        // 01 = generate SOCA on
first event

        // bit 15-14     00:     EPWMxSOCB, read-only
        // bit 13-12     00:     SOCBPRD, don't care
        // bit 11-10     00:     EPWMxSOCA, read-only
        // bit 9-8       01:     SOCAPRD, 01 = generate SOCA on first event
        // bit 7-4       0000:   reserved
        // bit 3-2       00:     INTCNT, don't care
        // bit 1-0       00:     INTPRD, don't care


        // Configure Event-Trigger Selection Register (ETSEL)
        //// Refer table 3-63 TRM page 357

        EPwm1Regs.ETSEL.all = 0x0000;
        EPwm1Regs.ETSEL.bit.SOCAEN=1;                                   // Start of
Conversion A Enable
        EPwm1Regs.ETSEL.bit.SOCASEL= ET_CTR_ZERO;                 // 010=SOCA when TBCTR =
ZERO

        // bit 15        0:      SOCBEN, 0 = disable SOCB
        // bit 14-12     000:    SOCBSEL, don't care
        // bit 11        1:      SOCAEN, 1 = enable SOCA
        // bit 10-8      010:    SOCASEL, 010 = SOCA on PRD event
        // bit 7-4       0000:   reserved
        // bit 3         0:      INTEN, 0 = disable interrupt
        // bit 2-0       000:    INTSEL, don't care



    EPwm1Regs.TBCTL.bit.CTRMODE=TB_COUNT_UPDOWN;          //Enable the time-base in count up
down mode

//-----------------------------------------------------------------------
//--- Configure ePWM2 for 50 kHz symmetric PWM on EPWM2A and EPWM2B pin
//-----------------------------------------------------------------------

        // Setup counter mode
        EPwm2Regs.TBCTL.bit.CTRMODE = TB_FREEZE;                  // Stop-freeze counter
operation
        EPwm2Regs.TBCTL.bit.FREE_SOFT=0x3;                                //1x - Free run
        EPwm2Regs.TBCTL.bit.PHSEN = TB_DISABLE;                           // Disable phase
loading - for MASTER mode
        EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;                            //  reload PRD on
counter=0
        EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;           // sync-out disabled
        EPwm2Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
        EPwm2Regs.TBCTL.bit.CLKDIV = TB_DIV1;

        EPwm2Regs.TBPRD = 800;                                                  // Set timer
period for up-count
        EPwm2Regs.TBCTR = 0x0000;                                               // Clear
counter

        EPwm2Regs.TBPHS.half.TBPHS = 0x0000;                      // Set Phase register to
zero

        EPwm2Regs.CMPA.half.CMPA=EPwm2Regs.TBPRD/2;               //Duty ratio = 50 %

        // Setup shadowing
```

```
        EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;               // shadow mode
        EPwm2Regs.CMPCTL.bit.LOADAMODE=CC_CTR_ZERO;               // Load on Zero

        // Set actions- up-count
        EPwm2Regs.AQCTLA.bit.CAD = AQ_SET;                              // PWM2A high when
CMPA = TBCNT on down count (CAD)
        EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;                      // PWM2A low when CMPA =
TBCNT on up count (CAU)

        EPwm2Regs.AQCTLB.bit.CAD = AQ_CLEAR;                      // PWM2B low when CMPA =
TBCNT on down count (CAD)
        EPwm2Regs.AQCTLB.bit.CAU = AQ_SET;                              // PWM2B high when
CMPA = TBCNT on up count (CAU)

    // set deadband, Trip-zone and PWM_chopper units
        EPwm2Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;            // enable Dead-band module
        EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;                 // Active Hi complementary
        EPwm2Regs.DBFED = 200;                                                          // FED
= 200 TBCLKs
        EPwm2Regs.DBRED = 200;

        EPwm2Regs.PCCTL.bit.CHPEN = 0;                                          // PWM
chopper unit disabled
        EPwm2Regs.TZDCSEL.all = 0x0000;                                         // All trip
zone and DC compare actions disabled

    EPwm2Regs.TBCTL.bit.CTRMODE=TB_COUNT_UPDOWN;          //Enable the time-base in count up
down mode


//-----------------------------------------------------------------------
//--- Configure ePWM3 for 50 kHz symmetric PWM on EPWM3A and EPWM3B pin
//-----------------------------------------------------------------------



        // Setup counter mode

        EPwm3Regs.TBCTL.bit.CTRMODE = TB_FREEZE     ;                   // Stop-freeze
counter operation
        EPwm3Regs.TBCTL.bit.FREE_SOFT=0x3;                              // 1x - Free run
        EPwm3Regs.TBCTL.bit.PHSEN = TB_DISABLE;                         // Disable phase
loading - for MASTER mode
        EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;                          // reload PRD on
counter=0
        EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;     // sync-out disabled
        EPwm3Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
        EPwm3Regs.TBCTL.bit.CLKDIV = TB_DIV1;

        EPwm3Regs.TBPRD = 800;                                                  // Set timer
period for up-count
        EPwm3Regs.TBCTR = 0x0000;                                               // Clear
counter

        EPwm3Regs.TBPHS.half.TBPHS = 0x0000;                      // Set Phase register to
zero

        EPwm3Regs.CMPA.half.CMPA=EPwm3Regs.TBPRD;                 //Duty ratio: Can this
become a sinusoidal signal? Set to an initial value to 50% and change it in the interrupt.

        // Setup shadowing
        EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;               // shadow mode
        EPwm3Regs.CMPCTL.bit.LOADAMODE=CC_CTR_ZERO;               // Load on Zero

        // Set actions- up-count
        EPwm3Regs.AQCTLA.bit.CAD = AQ_CLEAR;
        EPwm3Regs.AQCTLA.bit.CAU = AQ_SET;                              // Clear PWM1A on
CMPA=counter, up count

        EPwm3Regs.AQCTLB.bit.CAD = AQ_SET;
        EPwm3Regs.AQCTLB.bit.CAU = AQ_CLEAR;


        // set deadband, Trip-zone and PWM_chopper units

        EPwm3Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;            // enable Dead-band module
        EPwm3Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;                 // Active Hi complementary
```

```
        EPwm3Regs.DBFED = 200;                                                  // FED = 200
system clocks
        EPwm3Regs.DBRED = 200;
        EPwm3Regs.PCCTL.bit.CHPEN = 0;                                          // PWM
chopper unit disabled
        EPwm3Regs.TZDCSEL.all = 0x0000;                                         // All trip
zone and DC compare actions disabled


    EPwm3Regs.TBCTL.bit.CTRMODE=TB_COUNT_UPDOWN;           //Enable the time-base in count up
down mode




        //-----------------------------------------------------------------------
        // To start the ePWM Time-base clock (TBCLK) within the ePWM modules,
        // the TBCLKSYNC bit in PCLKCR0 must also be set.
        //--- Enable the clocks to the ePWM module.
        //--- Note: this should be done after all ePWM modules are configured
        //--- to ensure synchronization between the ePWM modules.
        //-----------------------------------------------------------------------

        asm(" EALLOW");
        // Enable EALLOW protected register access
        SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;                              // TBCLK to ePWM
modules enabled
        asm(" EDIS");                                                                   //
Disable EALLOW protected register access

}// end InitEPWMs()
```

C-3: ADC INITIALIZATION

```
//---------------------------------------------------------------------------
//      FILE:                   converter_Adc.c
//      Description:    ADC setup for measurements
//      Version:                1.0
//  Target:             TMS320F2806x family
//      Type:                   Device Dependent
//  Copyright       ELK21 course 2011
//  Date:           June 2012
//---------------------------------------------------------------------------
// TRM = F2806x Technical reference manual

#include "F2806x_Device.h"                          // F2806x header file peripheral address
definitions

#define CPU_RATE   12.500L                  // for a 80MHz CPU clock speed (SYSCLKOUT)
                                            // 1/80 MHz = 12.5 ns
#define ADC_usDELAY  1000L                          // used for delay of 1ms = 1000 us

// DO NOT MODIFY THIS LINE.
//DSP28x_usDelay defined in F2806x_usDelay.asm
#define DELAY_US(A)  DSP28x_usDelay(((((long double) A * 1000.0L) / (long double)CPU_RATE) -
9.0L) / 5.0L)
extern void DSP28x_usDelay(Uint32 Count);


/*************************************************************
** Description: Initializes the ADC on the F2806x
*************************************************************/
void InitADC(void)
{

       EALLOW;                                                  // Enable EALLOW protected
register access

// enable ADC clock in case user forgot to enable it in DeviceInit()
SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;       // Enable ADC clock

//--- Reset the ADC module
// Note: The ADC is already reset after a DSP reset, but this example is just showing
// good coding practice to reset the peripheral before configuring it as you never
// know why the DSP has started the code over again from the beginning).
//ADC Control Register 1 (ADCCTL1) - refer TRM Table 8-3 page 491

       AdcRegs.ADCCTL1.bit.RESET = 1;               // ADC module software reset - Bit 15


// Must wait 2 ADCCLK periods for the reset to take effect.
       asm(" NOP");
       asm(" NOP");

//--- Power-up and configure the ADC
// ADC Control Register 1 (ADCCTL1)
// Refer TRM table 8-3 page 491 also power-up sequence page 487

    AdcRegs.ADCCTL1.bit.ADCPWDN      = 1;    // ADC power down, 0=powered down, 1=powered up
    AdcRegs.ADCCTL1.bit.ADCBGPWD     = 1;    // ADC bandgap power down, 0=powered down,
1=powered up
    AdcRegs.ADCCTL1.bit.ADCREFPWD    = 1;    // ADC reference power down, 0=powered down,
1=powered up
    AdcRegs.ADCCTL1.bit.ADCREFSEL    = 0;    // ADC reference select, 0=internal, 1=external
    AdcRegs.ADCCTL1.bit.ADCENABLE    = 1;    // 0=disable, 1=Enable ADC - bit 14

    DELAY_US(ADC_usDELAY);                                  // Delay - Delay - Must be there
for F2806x devices



    AdcRegs.ADCCTL1.bit.INTPULSEPOS  = 1;    // create int pulses 1 cycle prior to output latch
                                             // INT pulse generation, 0=start of conversion,
1=end of conversion
       // For other values, use default ones
```

```
//-----ADC Control Register 2 (ADCCTL2
// Refer TRM table 8-4 page 493

       //AdcRegs.ADCCTL2.all = 0x0001;                     // ADC clock configuration

       AdcRegs.ADCCTL2.bit.CLKDIV4EN=0;            //ADC clock divider.  0=no effect,
1=CPUCLK/4 if CLKDIV2EN=1 (else no effect)
       AdcRegs.ADCCTL2.bit.ADCNONOVERLAP=0;        // 0=overlap sample and conversion, 1=no
overlap
       AdcRegs.ADCCTL2.bit.CLKDIV2EN=1;            //ADC clock divider.  0=CPUCLK, 1=CPUCLK/2



//--- SOC0 configuration
// ADC Sample Mode Register - ADCSAMPLEMODE - table 8-11 TRM page 499

       AdcRegs.ADCSAMPLEMODE.bit.SIMULEN0 = 0;     // 0=Single sample mode set for SOC0 and
SOC1
                                                                                    //
1=Simultaneous sample for SOC0 and SOC1

    //ADC SOC0 - SOC15 Control Registers (ADCSOCxCTL)
    //TRM table 8-18 page 503

    //bit TRIGSEL - SOCx Trigger Source Select.
       AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 5;         // Trigger using ePWM1SOCA
       AdcRegs.ADCSOC1CTL.bit.TRIGSEL = 5;         // Trigger using ePWM1SOCA
       AdcRegs.ADCSOC2CTL.bit.TRIGSEL = 5;         // Trigger using ePWM1SOCA
       AdcRegs.ADCSOC3CTL.bit.TRIGSEL = 5;         // Trigger using ePWM1SOCA
       AdcRegs.ADCSOC4CTL.bit.TRIGSEL = 5;         // Trigger using ePWM1SOCA
       AdcRegs.ADCSOC5CTL.bit.TRIGSEL = 5;         // Trigger using ePWM1SOCA
       AdcRegs.ADCSOC6CTL.bit.TRIGSEL = 5;         // Trigger using ePWM1SOCA
       AdcRegs.ADCSOC7CTL.bit.TRIGSEL = 5;         // Trigger using ePWM1SOCA
       // bit CHSEL - SOCx Channel Select. Selects the channel to be converted when SOCx is
received by the ADC.
       AdcRegs.ADCSOC0CTL.bit.CHSEL = 0;           // Convert channel ADCINA0 (ch0)
       AdcRegs.ADCSOC1CTL.bit.CHSEL = 1;           // Convert channel ADCINA1 (ch0)
       AdcRegs.ADCSOC2CTL.bit.CHSEL = 2;           // Convert channel ADCINA2 (ch0)
       AdcRegs.ADCSOC3CTL.bit.CHSEL = 3;           // Convert channel ADCINA3 (ch0)
       AdcRegs.ADCSOC4CTL.bit.CHSEL = 8;           // Convert channel ADCINB0 (ch0)
       AdcRegs.ADCSOC5CTL.bit.CHSEL = 9;           // Convert channel ADCINB1 (ch0)
       AdcRegs.ADCSOC6CTL.bit.CHSEL = 10;          // Convert channel ADCINB2 (ch0)
       AdcRegs.ADCSOC7CTL.bit.CHSEL = 11;          // Convert channel ADCINB3 (ch0)
       // bit ACQPS -SOCx Acquisition Prescale. Controls the sample and hold window for SOCx.
        // Minimum value allowed is 6.
       AdcRegs.ADCSOC0CTL.bit.ACQPS = 6;           // Acquisition window set to (6+1)=7
cycles
       AdcRegs.ADCSOC1CTL.bit.ACQPS = 6;           // Acquisition window set to (6+1)=7
cycles
       AdcRegs.ADCSOC2CTL.bit.ACQPS = 6;           // Acquisition window set to (6+1)=7
cycles
       AdcRegs.ADCSOC3CTL.bit.ACQPS = 6;           // Acquisition window set to (6+1)=7
cycles
       AdcRegs.ADCSOC4CTL.bit.ACQPS = 6;           // Acquisition window set to (6+1)=7
cycles
       AdcRegs.ADCSOC5CTL.bit.ACQPS = 6;           // Acquisition window set to (6+1)=7
cycles
       AdcRegs.ADCSOC6CTL.bit.ACQPS = 6;           // Acquisition window set to (6+1)=7
cycles
       AdcRegs.ADCSOC7CTL.bit.ACQPS = 6;           // Acquisition window set to (6+1)=7
cycles

       //for calibrating:
       ///AdcRegs.ADCOFFTRIM.bit.OFFTRIM=350;

  // ADC Interrupt Trigger SOC Select 1 Register (ADCINTSOCSEL1)
  // TRM table 8-12, page 500
  AdcRegs.ADCINTSOCSEL1.bit.SOC0 = 0;       // No ADCINT triggers SOC0.  TRIGSEL field
determines trigger.
//ADC Start of Conversion Priority Control Register (SOCPRICTL)
// TRM table 8-12, page 500
//bit SOCPRIORITY -
       AdcRegs.SOCPRICTL.bit.SOCPRIORITY = 0;      // All SOCs handled in round-robin mode
for all channels

//--- ADCINT1 configuration
// Interrupt Select 1 And 2 Register (INTSEL1N2)
```

```
// TRM table 8-9 page 496
 // bit INT1CONT - ADCINTy Continuous Mode Enable
       AdcRegs.INTSEL1N2.bit.INT1CONT = 0;
       // 1= ADCINTy pulses are generated whenever an EOC pulse is generated irrespective if
the flag bit is
       // cleared or not
       // 0=No further ADCINTx pulses are generated until ADCINTx flag (in ADCINTFLG register)
is cleared by user.

    // ADCINTy Interrupt Enable
    AdcRegs.INTSEL1N2.bit.INT1E = 1;                // 0=ADCINT1 is disabled, 1=ADCINT1 is
enabled.
    //INTySEL - ADCINTy EOC Source Select
       AdcRegs.INTSEL1N2.bit.INT1SEL = 7;                 // EOC7 triggers ADCINT1
                                                       // here you specify which EOC should
trigger an interrupt

//--- Enable the ADC interrupt - This is done in the main program

       EDIS;                                                           // Disable EALLOW
protected register access

} // end InitAdc()
//--- end of file -----------------------------------------------------
```

APPENDIX D: DC MOTOR DRIVE MICROCONTROLLER CODE

D-1: MAIN FILE

```
/**********************************
File name: DC_md_main-F2806x_1.c
Purpose: Control a DC motor drive
***********************************************/

#include "F2806x_Device.h"


void DeviceInit(void);  // defined in DC_md_DevInit_F2806x.c
void InitEPWMs(void);   // defined in DC_md_PWM_Setup.c
void InitEQEP(void);    // defined in DC_md_EQep_Setup.c
void InitADC(void);           // defined in DC_md_Adc.c
void InitEPWM4(void); // defined in DC_md_Adc.c    (used for verification)

//define interrupt service routines
interrupt void EQEP1_ISR(void);
interrupt void ADCINT1_ISR(void);

Uint32 EndlessLoopCounter = 0;
int QepIntCounter = 0;
int AdcIntCounter = 0;

// Speed controller parameters
int speedArray[50];
int actualSpeed = 0;
int setSpeed = 0;
int speedError=0;
double kpS=0;
double kiS=0;
double pTermS=0;
double iStateS=0;
int iStateSMax=100;
int iStateSMin=-100;
double iTermS=0;
int speedController=0;



// Current controller parameters
int setCurrent=0;
int currentError=0;
double kpI=0;
double kiI=0;
double pTermI=0;
double iStateI=0;
int iStateIMax=100;
int iStateIMin=-100;
double iTermI=0;
int currentController=0;


//ADC ISR buffer array
Uint16 AdcBuf[50][8]={0};




void main(void)
{
```

```
// Initialize System Control
// Enable Peripheral Clocks, GPIO
// Initialize the PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
 DeviceInit();
 InitEPWMs();  //Initialize PWM modules
 InitADC();           // Initialize ADC
 InitEQEP();   // Initialize QEP
 InitEPWM4();  // Initialize EPWM4

 //Enable the interrupt
 EALLOW;
       PieVectTable.EQEP1_INT=&EQEP1_ISR;
       PieVectTable.ADCINT1=&ADCINT1_ISR;
 EDIS;

 PieCtrlRegs.PIEIER1.bit.INTx1=1;
 PieCtrlRegs.PIEIER5.bit.INTx1=1;
 IER |= M_INT1;
 IER |=M_INT5;
 EINT;

  for(;;)  //infinite loop
  {
    asm(" NOP");
    if (EndlessLoopCounter++ >= 4294967295)
    {
       EndlessLoopCounter=0;
    }
  }

}// end of main



//EQP ISR:
interrupt void EQEP1_ISR(void)
{
       static int index=0;    //Initialize an index which counts every interrupt

       int border = 70;       //indicates the QPOSLAT value at the border between the high and
low speed measurements

       //Speed calculation
       if (EQep1Regs.QEPSTS.bit.QDF==1)             // If forward motor rotation
       {
              //if high speed, equation 3.19 (rpm):
              if (EQep1Regs.QPOSLAT>border) actualSpeed =
1171875*EQep1Regs.QPOSLAT/EQep1Regs.QUPRD;
              // if low speed, equation 3.21 (rpm):
              else if(EQep1Regs.QPOSLAT<=border) actualSpeed =
75000000/EQep1Regs.QCPRDLAT/1024;
       }
       else if (EQep1Regs.QEPSTS.bit.QDF==0)        // If reverse motor direction
       {
              Uint32 POSLAT = EQep1Regs.QPOSMAX - EQep1Regs.QPOSLAT;
              //if high speed, equation 3.20 (rpm):
              if(POSLAT >border) actualSpeed = - (1171875*POSLAT/EQep1Regs.QUPRD);
              //if low speed, equation 3.21 (rpm):
              else if (POSLAT<=border) actualSpeed = - 75000000/EQep1Regs.QCPRDLAT/1024;
       }
       //Store the results in a circular array
       speedArray[index]=actualSpeed;

       //Speed PI controller:
       speedError = setSpeed - actualSpeed;               //Calculate the error
       pTermS  =  kpS*speedError;                                      // proportional term
       iStateS += speedError;                                          // the sum of the
speed errors
       if (iStateS>iStateSMax)iStateS=iStateSMax;          // anti-windup
       if (iStateS<iStateSMin)iStateS=iStateSMin;
       iTermS =  kiS*iStateS*EQep1Regs.QUPRD/80000000; // integral term. Sampling
period=EQep1Regs.QUPRD/80MHz
       speedController=pTermS+iTermS;                                  // speed
controller's output
```

```
        setCurrent+=speedController;                                    // update setCurrent
        if(setCurrent >4096)setCurrent = 4096;                          // avoid setcurrent to
become too high.

        // This is the maximum value of the input current.

        index++;
        if (index==50) index=0;


        PieCtrlRegs.PIEACK.all = PIEACK_GROUP5;     // Acknowledge this interrupt to receive
more interrupts
        EQep1Regs.QCLR.bit.UTO=1;                                       // Clears unit time out
interrupt flag
        EQep1Regs.QCLR.bit.INT=1;                                       // Clears the interrupt flag
and enables further interrupts
                                                                                        // to
be generated if an event flags is set to 1
        EQep1Regs.QEPSTS.bit.UPEVNT=1;                                  // Clear unit position event
flag to allow for more latches
                                                                                        // of
the QCTMR value into the QCPRD register


        //QEP interrupt counter
        QepIntCounter++;
        if (QepIntCounter==200)       QepIntCounter=0;


}//end of ISR QEP




interrupt void ADCINT1_ISR(void)
{
        static int index = 0;

        //Store the converted values in a circular array:
        AdcBuf[index][0]=AdcResult.ADCRESULT0;
        AdcBuf[index][1]=AdcResult.ADCRESULT1;
        AdcBuf[index][2]=AdcResult.ADCRESULT2;
        AdcBuf[index][3]=AdcResult.ADCRESULT3;
        AdcBuf[index][4]=AdcResult.ADCRESULT4;
        AdcBuf[index][5]=AdcResult.ADCRESULT5;
        AdcBuf[index][6]=AdcResult.ADCRESULT6;
        AdcBuf[index][7]=AdcResult.ADCRESULT7;

        index ++;
        if (index == 50) index =0;

        static int indexB = 1;
        indexB ++;

        //Current/torque loop
        if (indexB==10) //To make the current loop calculate for every 10th interrupt, making
the sampling time for the current loop on order of magnitude lower than the switching
frequency
        {
                //calculates the output of the current control loop
                currentError=setCurrent-AdcResult.ADCRESULT2;       //calculates the current
error
                pTermI=kpI*currentError;
        //proportional term
                iStateI+=currentError;                                                  //sums up the
current errors
                if(iStateI>iStateIMax) iStateI=iStateIMax;          //anti-windup
                if(iStateI<iStateIMin) iStateI=iStateIMin;
                iTermI=kiI*iStateI*EPwm1Regs.TBPRD/4000000;         //integral term: sampling
period=10*EPwm1Regs.TBPRD*2/80MHz
                currentController=pTermI+iTermI;                                //current controller
output


                //calculate the new duty ratio
```

```
                EPwm1Regs.CMPA.half.CMPA -= currentController;
                EPwm2Regs.CMPA.half.CMPA=EPwm1Regs.CMPA.half.CMPA;

                indexB=1;
        }


        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;             // Must acknowledge the PIE group
        AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;               // Clear ADCINT1 flag to enable
further interrupts

        //ADC interrupt counter
          AdcIntCounter++;
       if (AdcIntCounter >=200)
         AdcIntCounter=0;

} //end of ADCINT1_ISR
```

D-2: Device Initialzation

See C-2

D-3: PWM Initialization

See C-3

D-4: ADC Initialization

See C-3.

D-5: QEP INITIALIZATION

```
//--------------------------------------------------------------------------------
//      FILE:                   DC_md_QEP_Setup.c
//      Description:    QEP setup for speed measurements and speed controller
//      Version:                1.0
//  Target:             TMS320F2806x family
//      Type:                   Device Dependent
//  Copyright       Texas Instruments
//  Date:                       April 2012
//--------------------------------------------------------------------------------

#include "F2806x_Device.h"                          // DSP2860x Headerfile Include File


void InitEQEP()
{

                EQep1Regs.QUPRD=800000;              // Unit Timer = 100Hz=80 MHz/800000.
                EQep1Regs.QDECCTL.bit.QSRC=00;      // QEP quadrature count mode

                EQep1Regs.QEPCTL.bit.FREE_SOFT=2;    // Position counter is unaffected by
emulation suspend
                EQep1Regs.QEPCTL.bit.PCRM=11;       // PCRM=11 mode: on unit time out event:
                                                                                // QPOSCNT is
latched into QPOSLAT and reset
                EQep1Regs.QEPCTL.bit.UTE=1;         // Unit Timer Enable

                EQep1Regs.QPOSMAX=4294967295;        // Max value of POSCNT, = max value of 32
bit int.
                EQep1Regs.QEPCTL.bit.QPEN=1;        // QEP enable

                EQep1Regs.QCAPCTL.bit.UPPS=2;       // 1/4 for unit position
        EQep1Regs.QCAPCTL.bit.CCPS=6;       // 1/64 for CAP clock
                EQep1Regs.QCAPCTL.bit.CEN=1;        // QEP Capture Enable
                EQep1Regs.QEPCTL.bit.QCLM=1;        // Latch on unit time out

                EQep1Regs.QEINT.bit.UTO=1;           // Enable unit time out interrupt

                EQep1Regs.QFLG.bit.UTO=1;                   // Unit time out interrupt flag:
Set by eQEP unit timer period match

} // end InitEQEP()
//--- end of file -----------------------------------------------------
```

D-6: PWM4 INITIALIZATION

```
//###########################################################################
//
// FILE:       Example_EpwmSetup.c
//
// TITLE:      Pos speed measurement using EQEP peripheral
//
// DESCRIPTION: This is for the verification of the speed measurements and speed controller
algorithm
//
// This file contains source for the ePWM initialization for the
// pos/speed module
//
//###########################################################################
// $TI Release: F2806x C/C++ Header Files and Peripheral Examples V130 $
// $Release Date: November 30, 2011 $
//###########################################################################

#include "DSP28x_Project.h"     // Device Headerfile and Examples Include File


#define CPU_CLK   80e6

#define PWM_CLK   5e3              // 5kHz (300rpm) EPWM1 frequency. Freq. can be changed here
#define SP        CPU_CLK/(2*PWM_CLK)
#define TBCTLVAL  0x200E           // up-down count, timebase=SYSCLKOUT

void InitEPWM4()
{
        EPwm4Regs.TBSTS.all=0;
        EPwm4Regs.TBPHS.half.TBPHS =0;
        EPwm4Regs.TBCTR=0;

        EPwm4Regs.CMPCTL.all=0x50;     // immediate mode for CMPA and CMPB
        EPwm4Regs.CMPA.half.CMPA=SP/2;
        EPwm4Regs.CMPB=0;

        EPwm4Regs.AQCTLA.all=0x60;     // CTR=CMPA when inc->EPWM1A=1, when dec->EPWM1A=0
        EPwm4Regs.AQCTLB.all=0x09;     // CTR=PRD ->EPWM1B=1, CTR=0 ->EPWM1B=0
        EPwm4Regs.AQSFRC.all=0;
        EPwm4Regs.AQCSFRC.all=0;

        EPwm4Regs.TZSEL.all=0;
        EPwm4Regs.TZCTL.all=0;
        EPwm4Regs.TZEINT.all=0;
        EPwm4Regs.TZFLG.all=0;
        EPwm4Regs.TZCLR.all=0;
        EPwm4Regs.TZFRC.all=0;

        //EPwm4Regs.ETSEL.all=0x0A;      // Interrupt on PRD
        //EPwm4Regs.ETPS.all=1;
        //EPwm4Regs.ETFLG.all=0;
        //EPwm4Regs.ETCLR.all=0;
        //EPwm4Regs.ETFRC.all=0;

        EPwm4Regs.PCCTL.all=0;

        EPwm4Regs.TBCTL.all=0x0010+TBCTLVAL; // Enable Timer
        EPwm4Regs.TBPRD=SP;

}
```

Appendix E: List of the Available Digital Content

E-1: Eagle Files

- Version I schematics: master.sch
- Version I layout: master.brd
- Version II schematics: master v2.sch
- Version II layout: master v2.brd

E-2: Code Composer Studio Projects

- General converter code project: Converter
- DC motor drive code project: DC_motor_drive

APPENDIX F: FIRST PAGE OF DATASHEETS

LEM

Current Transducer LTS 15-NP

I_{PN} = 15 At

For the electronic measurement of currents : DC, AC, pulsed, mixed, with a galvanic isolation between the primary circuit (high power) and the secondary circuit (electronic circuit).

16058

Electrical data

I_{PN}	Primary nominal current rms		15	At
I_{PM}	Primary current, measuring range		0 .. ± 48	At
I_P	Overload capability		250	At
V_{OUT}	Analog output voltage @ I_P		2.5 ± (0.625·I_P/I_{PN})	V
	I_P = 0		2.5 [1]	V
G	Sensitivity		41.6	mV/A
N_S	Number of secondary turns (± 0.1 %)		2000	
R_L	Load resistance		≥ 2	kΩ
R_{IM}	Internal measuring resistance (± 0.5 %)		83.33	Ω
TCR_{IM}	Temperature coefficient of R_{IM}		< 50	ppm/K
V_C	Supply voltage (± 5 %)		5	V
I_C	Current consumption @ V_C = 5 V	Typ	28+I_S[2]+(V_{OUT}/R_L)	mA

Accuracy - Dynamic performance data

X	Accuracy @ I_{PN}, T_A = 25°C		± 0.2		%
	Accuracy with R_{IM} @ I_{PN}, T_A = 25°C		± 0.7		%
ε_L	Linearity error		< 0.1		%
			Typ	Maxi	
TCV_{OUT}	Temperature coefficient of V_{OUT} @ I_P = 0	- 10°C .. + 85°C	65	120	ppm/K
		- 40°C .. - 10°C		170	ppm/K
TCG	Temperature coefficient of G	- 40°C .. + 85°C		50 [3]	ppm/K
V_{OM}	Magnetic offset voltage @ I_P = 0, after an overload of	3 x I_{PN}		± 0.5	mV
		5 x I_{PN}		± 2.0	mV
		10 x I_{PN}		± 2.0	mV
t_{ra}	Reaction time @ 10 % of I_{PN}		< 100		ns
t_r	Response time to 90 % of I_{PN} step		< 400		ns
di/dt	di/dt accurately followed		> 35		A/µs
BW	Frequency bandwidth (0 .. - 0.5 dB)		DC .. 100		kHz
	(- 0.5 .. 1 dB)		DC .. 200		kHz

General data

T_A	Ambient operating temperature	- 40 .. + 85	°C
T_S	Ambient storage temperature	- 40 .. + 100	°C
m	Mass	10	g
	Standards	EN 50178: 1997	
		IEC 60950-1: 2001	

Notes: [1] Absolute value @ T_A = 25°C, 2.475 < V_{OUT} < 2.525
[2] I_S = I_P/N_S
[3] Only due to TCR_{IM}.

Features

- Closed loop (compensated) multi-range current transducer using the Hall effect
- Unipolar voltage supply
- Compact design for PCB mounting
- Isolated plastic case recognized according to UL 94-V0
- Incorporated measuring resistance
- Extended measuring range.

Advantages

- Excellent accuracy
- Very good linearity
- Very low temperature drift
- Optimized response time
- Wide frequency bandwidth
- No insertion losses
- High immunity to external interference
- Current overload capability.

Applications

- AC variable speed drives and servo motor drives
- Static converters for DC motor drives
- Battery supplied applications
- Uninterruptible Power Supplies (UPS)
- Switched Mode Power Supplies (SMPS)
- Power supplies for welding applications.

Application domain

- Industrial.

070129/18

HEWLETT PACKARD

Quick Assembly Two and Three Channel Optical Encoders

Technical Data

HEDS-5500/5540
HEDS-5600/5640
New **HEDM-5500/5600**

Features

- **Two Channel Quadrature Output with Optional Index Pulse**
- **Quick and Easy Assembly**
- **No Signal Adjustment Required**
- **External Mounting Ears Available**
- **Low Cost**
- **Resolutions Up to 1024 Counts Per Revolution**
- **Small Size**
- **-40°C to 100°C Operating Temperature**
- **TTL Compatible**
- **Single 5 V Supply**

Description

The HEDS-5500/5540, HEDS-5600/5640, and HEDM-5500/5600 are high performance, low cost, two and three channel optical incremental encoders. These encoders emphasize high reliability, high resolution, and easy assembly.

Each encoder contains a lensed LED source, an integrated circuit with detectors and output circuitry, and a codewheel which rotates between the emitter and detector IC. The outputs of the HEDS-5500/5600 and HEDM-5500/5600 are two square waves in quadrature. The HEDS-5540 and 5640 also have a third channel index output in addition to the two channel quadrature. This index output is a 90 electrical degree, high true index pulse which is generated once for each full rotation of the codewheel.

The HEDS series utilizes metal codewheels, while the HEDM series utilizes a film codewheel allowing for resolutions to 1024 CPR. The HEDM series is not available with a third channel index.

These encoders may be quickly and easily mounted to a motor. For larger diameter motors, the HEDM-5600, and HEDS-5600/5640 feature external mounting ears.

The quadrature signals and the index pulse are accessed through five 0.025 inch square pins located on 0.1 inch centers.

Standard resolutions between 96 and 1024 counts per revolution are presently available. Consult local Hewlett-Packard sales representatives for other resolutions.

Applications

The HEDS-5500, 5540, 5600, 5640, and the HEDM-5500, 5600 provide motion detection at a low cost, making them ideal for high volume applications. Typical applications include printers, plotters, tape drives, positioning tables, and automatic handlers.

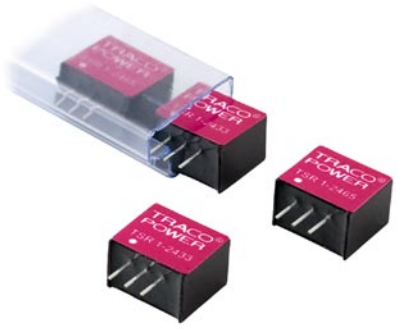
ESD WARNING: NORMAL HANDLING PRECAUTIONS SHOULD BE TAKEN TO AVOID STATIC DISCHARGE.

DC/DC Converters

TSR-1 Series, 1 A

Features

- Up to 96 % efficiency
  – No heat-sink required
- Pin compatible with LMxx linear regulators
- SIP-package fits existing TO-220 footprint
- Built in filter capacitors
- Operation temp. range –40°C to +85°C
- Short circuit protection
- Wide input operating range
- Excellent line / load regulation
- Low standby current
- 3-year product warranty



The new TSR-1 series step-down switching regulators are drop-in replacement for inefficient 78xx linear regulators. A high efficiency up to 96 % allows full load operation up to +60 °C ambient temperature without the need of any heat-sink or forced cooling.
The TSR-1 switching regulators provide other significant features over linear regulators, i.e. better output accuracy (±2 %), lower standby current of 2 mA and no requirement of external capacitors. The high efficiency and low standby power consumption makes these regulators an ideal solution for many battery powered applications.

Models

Order code	Input voltage range	Output voltage	Output current max.	Efficiency typ. @ Vin min.	Efficiency typ. @ Vin max.
TSR 1-2412	**4.6 – 36 VDC***	1.2 VDC		74 %	62 %
TSR 1-2415	**4.6 – 36 VDC***	1.5 VDC		78 %	65 %
TSR 1-2418	**4.6 – 36 VDC***	1.8 VDC		82 %	69 %
TSR 1-2425	**4.6 – 36 VDC***	2.5 VDC		87 %	75 %
TSR 1-2433	**4.75 – 36 VDC***	3.3 VDC	**1.0 A**	91 %	78 %
TSR 1-2450	**6.5 – 36 VDC***	5.0 VDC		94 %	84 %
TSR 1-2465	**9.0 – 36 VDC***	6.5 VDC		93 %	87 %
TSR 1-2490	**12 – 36 VDC***	9.0 VDC		95 %	90 %
TSR 1-24120	**15 – 36 VDC***	12 VDC		95 %	92 %
TSR 1-24150	**18 – 36 VDC***	15 VDC		96 %	94 %

* For input voltage higher than 32 VDC an input capcitor 22 µF / 50 V is required. See application notes (page 3)

International
IOR Rectifier

February 18, 2009

IRS211(7,71,8)(S)
SINGLE CHANNEL DRIVER

IC Features

- Floating channel designed for bootstrap operation
- Fully operational to +600V
- Tolerant to negative transient voltage, dV/dt immune
- Gate drive supply range from 10 V to 20V
- Undervoltage lockout
- CMOS Schmitt-triggered inputs with pull-down
- Output in phase with input
- RoHS compliant
- IRS2117 and IRS2118 available in PDIP8

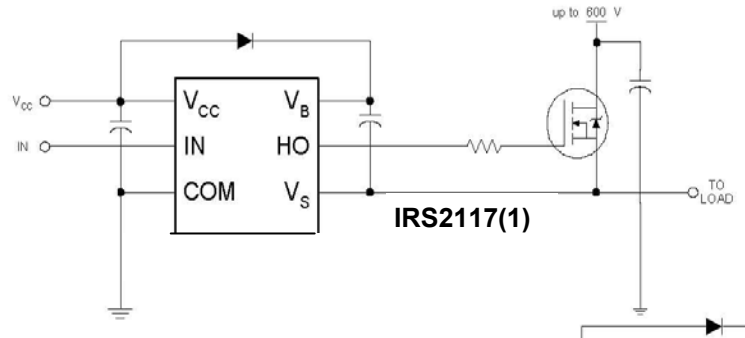
Product Summary

Topology		Single High Side
V_{OFFSET}		600 V
V_{OUT}		10V-20 V
I_{O+} & I_{O-} (typical)		290 mA & 600 mA
IN voltage threshold	IRS211(7,8)	9.5 V & 6 V
	IRS21171	2.5 V & 0.8 V

Package Type

SOIC8

PDIP8



(Refer to Lead Assignments for correct pin configuration). These diagrams show electrical connections only. Please refer to our Application Notes and Design Tips for proper circuit board layout.

Spezifikation für Freigabe / specification for release

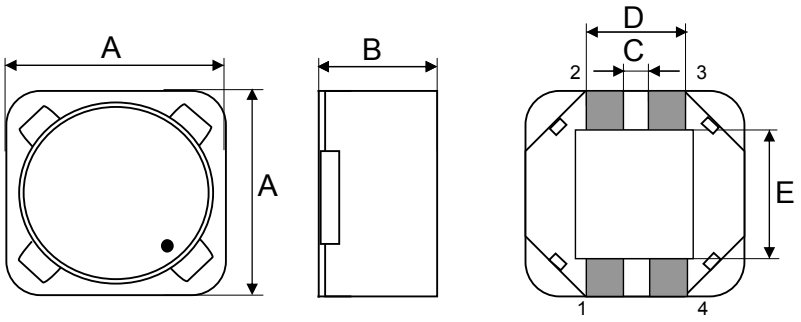
Kunde / customer :
Artikelnummer / part number : **744877100** LF

Bezeichnung : **DOPPELDROSSEL WE-DD**
description : **POWER-CHOKE WE-DD**

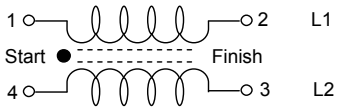
RoHS compliant

DATUM / DATE : 2005-05-23

A Mechanische Abmessungen / dimensions:



	Typ M	
A	**7,3 ± 0,4**	mm
B	**4,8 max.**	mm
C	**1,0 ± 0,1**	mm
D	**2,7 ± 0,1**	mm
E	**4,0 ± 0,2**	mm
F		mm

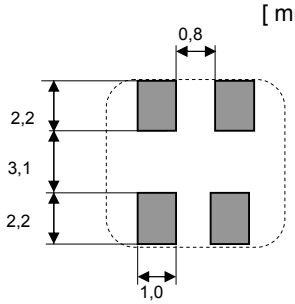


B Elektrische Eigenschaften / electrical properties:

Eigenschaften / properties	Testbedingungen / test conditions		Wert / value	Einheit / unit	tol.
Induktivität (je Wicklg.) / inductance (each wdg.)	**1 kHz / 0,25V**	L_1, L_2	**10,0**	µH	**±20%**
DC-Widerstand (je Wicklg.) / DC-resistance (each wdg)		$R_{DC1,2}$	**0,110**	Ω	**typ.**
DC-Widerstand (je Wicklg.) / DC-resistance (each wdg)		$R_{DC1,2}$	**0,120**	Ω	**max.**
Nennstrom (je Wicklg.)/ rated Current (each wdg.)	**ΔT = 40 K**	I_{N1}, I_{N2}	**1,10**	A	**max.**
Sättigungsstrom (je Wicklg.) / saturation current (each wdg.)	**ΔL/L = -10%**	I_{sat}	**2,80**	A	**typ.**
Eigenres.-Frequenz / self-res.-fequency		SRF	**27**	MHz	**typ.**
Nennspannung / rated voltage		U_{DC}	**80**	V	**max.**

C Lötpad / soldering spec.:

[mm]



D Prüfgeräte / test equipment:

HP 4274 A für/for L und/and Q

HP 34401 A für/for I_{DC} und/and R_{DC}

E Testbedingungen / test conditions:

Luftfeuchtigkeit / humidity: 33%

Umgebungstemperatur / temperature: +20°C

F Werkstoffe & Zulassungen / material & approvals:

Basismaterial / base material: Ferrit/ ferrite

Draht / wire: 2 SFBW; 155°C

G Eigenschaften / general specifications:

Betriebstemperatur / operating temperature: -40°C - + 125°C

Umgebungstemp. / ambient temperature: -40°C - + 85°C

It is recommended that the temperature of the part does not exceed 125°C under worst case operating conditions.

Freigabe erteilt / general release:

Kunde / customer		Würth Elektronik	
Datum / date	Unterschrift / signature	Geprüft / checked	Kontrolliert / approved

Name	Änderung / modification	Datum / date
SST	Version 8	05-05-23
SST	Version 7	05-03-29
MST	Version 6	04-10-11
SST	Version 5	04-09-15
MST	Version 4	04-04-27
SST	Version 3	04-03-12
AG	Version 2	03-06-11
AG	Version 1	02-10-30

Würth Elektronik eiSos GmbH & Co.KG

D-74638 Waldenburg · Max-Eyth-Strasse 1 - 3 · Germany · Telefon (+49) (0) 7942 - 945 - 0 · Telefax (+49) (0) 7942 - 945 - 400
http://www.we-online.com

FAIRCHILD SEMICONDUCTOR™

October 1987
Revised April 2002

CD4049UBC • CD4050BC
Hex Inverting Buffer • Hex Non-Inverting Buffer

General Description

The CD4049UBC and CD4050BC hex buffers are monolithic complementary MOS (CMOS) integrated circuits constructed with N- and P-channel enhancement mode transistors. These devices feature logic level conversion using only one supply voltage (V_{DD}). The input signal high level (V_{IH}) can exceed the V_{DD} supply voltage when these devices are used for logic level conversions. These devices are intended for use as hex buffers, CMOS to DTL/TTL converters, or as CMOS current drivers, and at V_{DD} = 5.0V, they can drive directly two DTL/TTL loads over the full operating temperature range.

Features

- Wide supply voltage range: 3.0V to 15V
- Direct drive to 2 TTL loads at 5.0V over full temperature range
- High source and sink current capability
- Special input protection permits input voltages greater than V_{DD}

Applications

- CMOS hex inverter/buffer
- CMOS to DTL/TTL hex converter
- CMOS current “sink” or “source” driver
- CMOS HIGH-to-LOW logic level converter

Ordering Code:

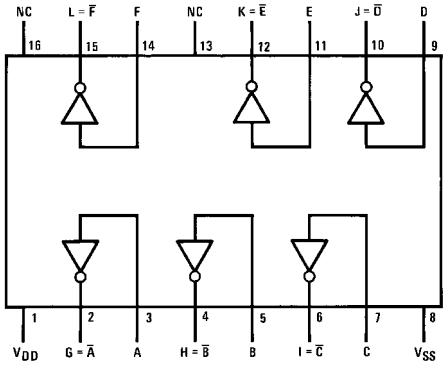
Order Number	Package Number	Package Description
CD4049UBCM	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
CD4049UBCN	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
CD4050BCM	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
CD4050BCN	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter “X” to the ordering code.

Connection Diagrams

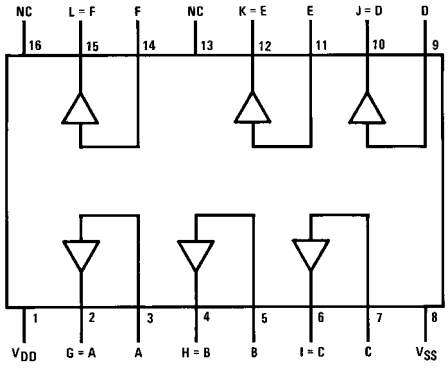
Pin Assignments for DIP

CD4049UBC



Top View

CD4050BC



Top View

Data Sheet
October 5, 2009

SW/SC001/003 Series DC-DC Converter Power Modules: 18-36V & 36-75Vdc Input; 3.3V-15Vdc Output; 1-3.5A Output Current

RoHS Compliant

Applications

- Wireless Networks
- Distributed power architectures
- Optical and Access Network Equipment
- Enterprise Networks
- Latest generation IC's (DSP, FPGA, ASIC) and Microprocessor powered applications

Options

- Remote On/Off logic (positive or negative), pin optional for TH version (Suffix 1 or 4)
- Output voltage adjustment-Trim, pin optional for TH version (Suffix 9)
- Surface Mount/Tape and Reel (-SR Suffix)

Features

- Compliant to RoHS EU Directive 2002/95/EC (-Z versions)
- Compliant to ROHS EU Directive 2002/95/EC with lead solder exemption (non-Z versions)
- Delivers up to 3.5A Output current

  15V (1A), 12V (1.25A), 5.0V (3A) and 3.3V (3.5A)
- High efficiency – 86% at 5.0V full load (V_{IN}=54 Vdc)
- Low output ripple and noise
- Small Size and low profile

  27.94mm x 24.38mm x 8.5mm

  (1.10 x 0.96 x 0.335 in)
- Industry Standard pin-out:

  TH version is LW series compatible
- Surface mount (SMT) or Through hole (TH)
- Remote On/Off (optional pin on TH version)
- Output overcurrent/voltage protection
- Single Tightly regulated output
- Output voltage adjustment trim ±10%
- Wide operating temperature range (-40°C to 85°C)
- Meets the voltage insulation requirements for ETSI 300-132-2 and complies with and is Licensed for Basic Insulation rating per EN 60950
- CE mark meets the 2006/95/EC directive§
- *UL** 60950-1Recognized, *CSA*† C22.2 No. 60950-1-03 Certified, and *VDE*‡ 0805: (IEC60950, 3rd Edition) Licensed
- ISO** 9001 and ISO 14001 certified manufacturing facilities
- Approved for Basic Insulation

Description

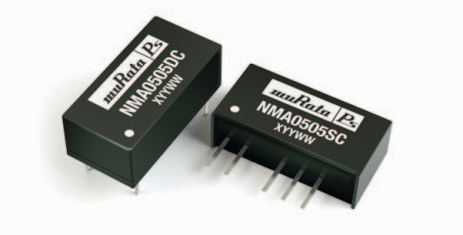
The SW/SC series power modules are isolated dc-dc converters that operate over a wide range of input voltage (V_{IN} = 18 - 36Vdc for SC modules and V_{IN} = 36 – 75Vdc for SW modules) and provide a single precisely regulated output. This series is a low cost, smaller size alternative to the existing LW/LAW/LC with enhanced performance parameters. The output is fully isolated from the input, allowing versatile polarity configurations and grounding connections. The modules exhibit high efficiency, typical efficiency of 86% for 5.0V/3A. Built-in filtering for both input and output minimizes the need for external filtering.

§ This product is intended for integration into end-use equipment. All of the required procedures of end-use equipment should be followed.
* *UL* is a registered trademark of Underwriters Laboratories, Inc.
† *CSA* is a registered trademark of Canadian Standards Association.
‡ *VDE* is a trademark of Verband Deutscher Elektrotechniker e.V.
** ISO is a registered trademark of the International Organization of Standards

Murata Power Solutions

NMA 5V, 12V & 15V Series

Isolated 1W Dual Output DC/DC Converters



FEATURES

- RoHS compliant
- Efficiency up to 80%
- Power density up to 0.85W/cm^3
- Wide temperature performance at full 1 Watt load, –40°C to 85°C
- Dual output from a single input rail
- UL 94V-0 package material
- No heatsink required
- Footprint from 1.17cm^2
- Industry standard pinout
- Power sharing on output
- 1kVDC isolation
- 5V, 12V, & 15V input
- 5V, 9V, 12V and 15V output
- Internal SMD construction
- Fully encapsulated with toroidal magnetics
- No external components required
- MTTF up to 3.1 million hours
- No electrolytic or tantalum capacitors

DESCRIPTION

The NMA series of industrial temperature range DC/DC converters are the standard building blocks for on-board distributed power systems. They are ideally suited for providing dual rail supplies on primarily digital boards with the added benefit of galvanic isolation to reduce switching noise. All of the rated power may be drawn from a single pin provided the total load does not exceed 1 watt.

RoHS COMPLIANT

For full details go to
www.murata-ps.com/rohs

SELECTION GUIDE

Order Code	Nominal Input Voltage	Output Voltage	Output Current	Input Current at Rated Load	Efficiency	Isolation Capacitance	MTTF[1]	Package Style
	V	V	mA	mA	%	pF	kHrs	
NMA0505DC	5	±5	±100	289	69	28	3103	DIP
NMA0509DC	5	±9	±55	267	75	32	2257	
NMA0512DC	5	±12	±42	260	77	34	1579	
NMA0515DC	5	±15	±33	256	78	36	1065	
NMA0505SC	5	±5	±100	289	69	28	3103	SIP
NMA0509SC	5	±9	±55	267	75	32	2257	
NMA0512SC	5	±12	±42	260	77	34	1579	
NMA0515SC	5	±15	±33	256	78	36	1065	
NMA1205DC	12	±5	±100	120	69	33	2193	DIP
NMA1209DC	12	±9	±55	113	74	46	1734	
NMA1212DC	12	±12	±42	111	75	55	1303	
NMA1215DC	12	±15	±33	110	76	54	932	
NMA1205SC	12	±5	±100	120	69	33	2193	SIP
NMA1209SC	12	±9	±55	113	74	46	1734	
NMA1212SC	12	±12	±42	111	75	55	1303	
NMA1215SC	12	±15	±33	110	76	54	932	
NMA1505DC	15	±5	±100	91	71	39	1941	DIP
NMA1512DC	15	±12	±42	87	78	68	790	
NMA1515DC	15	±15	±33	84	80	84	523	
NMA1505SC	15	±5	±100	91	71	39	1941	SIP
NMA1512SC	15	±12	±42	87	78	68	790	
NMA1515SC	15	±15	±33	84	80	84	523	

INPUT CHARACTERISTICS

Parameter	Conditions	Min.	Typ.	Max.	Units
Voltage range	Continuous operation, 5V input types	4.5	5	5.5	V
	Continuous operation, 12V input types	10.8	12	13.2	
	Continuous operation, 15V input types	13.5	15	16.5	
Reflected ripple current			20	40	mA p-p

ABSOLUTE MAXIMUM RATINGS

Lead temperature 1.5mm from case for 10 seconds	300°C
Internal power dissipation	450mW
Input voltage V_{IN}, NMA05 types	7V
Input voltage V_{IN}, NMA12 types	15V
Input voltage V_{IN}, NMA15 types	18V

1. Calculated using MIL-HDBK-217FN2 calculation model with nominal input voltage at full load.

All specifications typical at T_A=25°C, nominal input voltage and rated output current unless otherwise specified.

National Semiconductor

June 2006

LMC660
CMOS Quad Operational Amplifier

General Description

The LMC660 CMOS Quad operational amplifier is ideal for operation from a single supply. It operates from +5V to +15.5V and features rail-to-rail output swing in addition to an input common-mode range that includes ground. Performance limitations that have plagued CMOS amplifiers in the past are not a problem with this design. Input V_{OS}, drift, and broadband noise as well as voltage gain into realistic loads (2 kΩ and 600Ω) are all equal to or better than widely accepted bipolar equivalents.

This chip is built with National's advanced Double-Poly Silicon-Gate CMOS process.

See the LMC662 datasheet for a dual CMOS operational amplifier with these same features.

Features

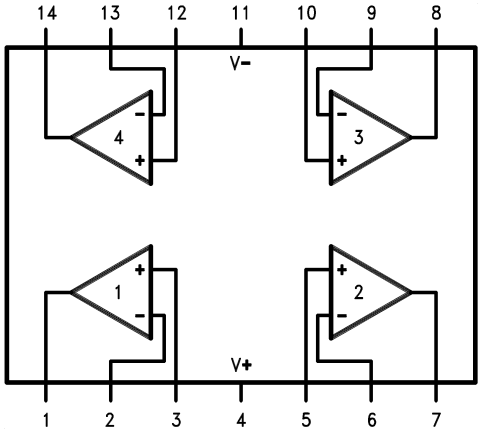
- Rail-to-rail output swing
- Specified for 2 kΩ and 600Ω loads
- High voltage gain: 126 dB
- Low input offset voltage: 3 mV
- Low offset voltage drift: 1.3 µV/˚C
- Ultra low input bias current: 2 fA
- Input common-mode range includes V^-
- Operating range from +5V to +15.5V supply
- I_{SS} = 375 µA/amplifier; independent of V^+
- Low distortion: 0.01% at 10 kHz
- Slew rate: 1.1 V/µs

Applications

- High-impedance buffer or preamplifier
- Precision current-to-voltage converter
- Long-term integrator
- Sample-and-Hold circuit
- Peak detector
- Medical instrumentation
- Industrial controls
- Automotive sensors

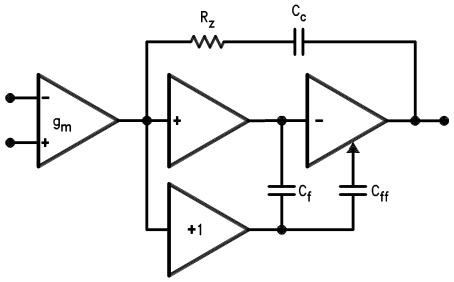
Connection Diagram

14-Pin SOIC/MDIP



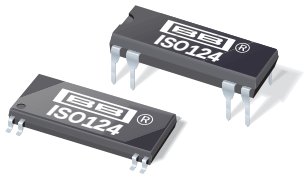
00876701

LMC660 Circuit Topology (Each Amplifier)



00876704

ISO124



SBOS074C – SEPTEMBER 1997 – REVISED SEPTEMBER 2005

Precision Lowest-Cost ISOLATION AMPLIFIER

FEATURES

- **100% TESTED FOR HIGH-VOLTAGE BREAKDOWN**
- **RATED 1500Vrms**
- **HIGH IMR: 140dB at 60Hz**
- **0.010% max NONLINEARITY**
- **BIPOLAR OPERATION: V_O = ±10V**
- **DIP-16 AND SO-28**
- **EASE OF USE: Fixed Unity Gain Configuration**
- **±4.5V to ±18V SUPPLY RANGE**

APPLICATIONS

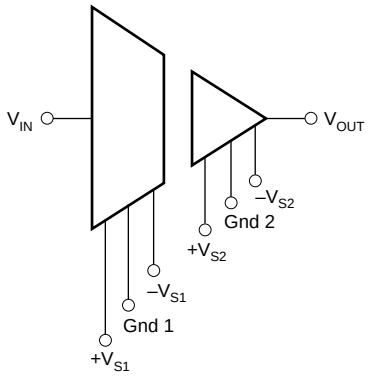
- **INDUSTRIAL PROCESS CONTROL: Transducer Isolator, Isolator for Thermocouples, RTDs, Pressure Bridges, and Flow Meters, 4-20mA Loop Isolation**
- **GROUND LOOP ELIMINATION**
- **MOTOR AND SCR CONTROL**
- **POWER MONITORING**
- **PC-BASED DATA ACQUISITION**
- **TEST EQUIPMENT**

DESCRIPTION

The ISO124 is a precision isolation amplifier incorporating a novel duty cycle modulation-demodulation technique. The signal is transmitted digitally across a 2pF differential capacitive barrier. With digital modulation, the barrier characteristics do not affect signal integrity, resulting in excellent reliability and good high-frequency transient immunity across the barrier. Both barrier capacitors are imbedded in the plastic body of the package.

The ISO124 is easy to use. No external components are required for operation. The key specifications are 0.010% max nonlinearity, 50kHz signal bandwidth, and 200µV/°C V_{OS} drift. A power supply range of ±4.5V to ±18V and quiescent currents of ±5.0mA on V_{S1} and ±5.5mA on V_{S2} make these amplifiers ideal for a wide range of applications.

The ISO124 is available in DIP-16 and SO-28 plastic surface mount packages.



International
IOR Rectifier

July 1, 2009

IRS2123S/IRS2124S

HIGH SIDE DRIVER IC

Features

- Floating channel designed for bootstrap operation
- Fully operational to +600 V
- Tolerant to negative transient voltage – dV/dt immune
- Gate drive supply range from 10 V to 20 V
- Undervoltage lockout
- CMOS Schmitt-triggered inputs with pull-down
- Output in phase with input (IRS2123) or out of Phase with input (IRS2124)
- Leadfree, RoHS compliant

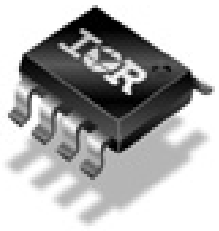
Typical Applications

- General purpose single highside inverters

Product Summary

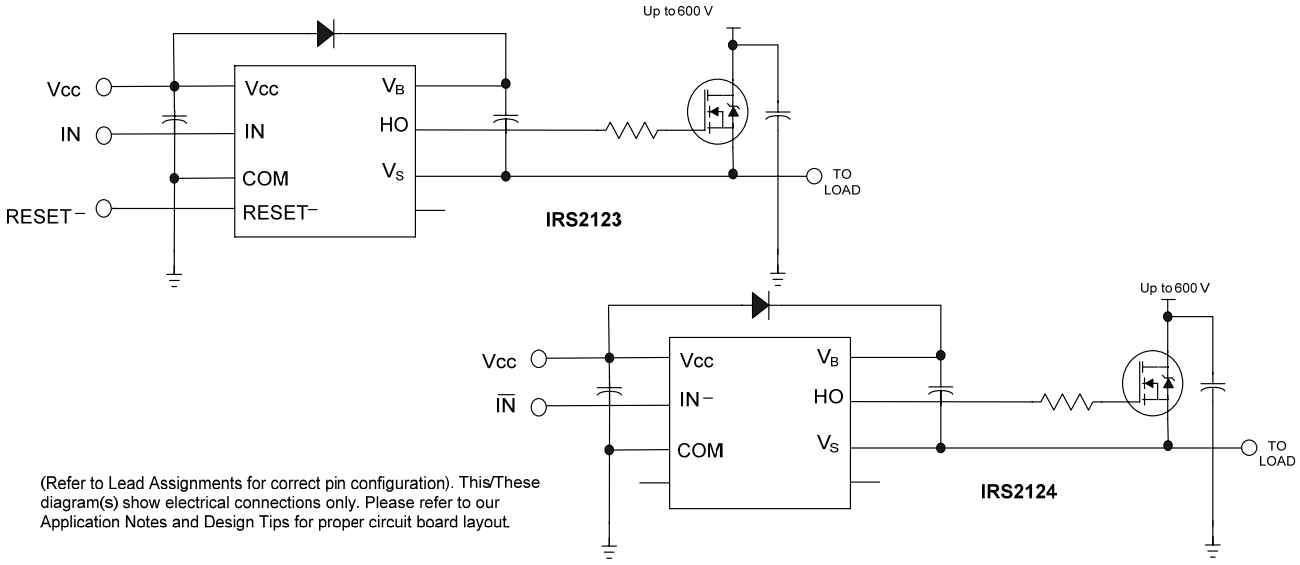
Topology	Single highside
V_{OFFSET}	≤ 600 V
V_{OUT}	10 V – 20 V
I_{o+} & I_{o-} (typical)	500 mA
t_{ON} & t_{OFF} (typical)	140 ns & 140 ns

Package Options



8-Lead SOIC

Typical Connection Diagram



(Refer to Lead Assignments for correct pin configuration). This/These diagram(s) show electrical connections only. Please refer to our Application Notes and Design Tips for proper circuit board layout.

International
IOR Rectifier

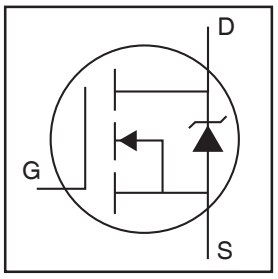
IRFB4110PbF

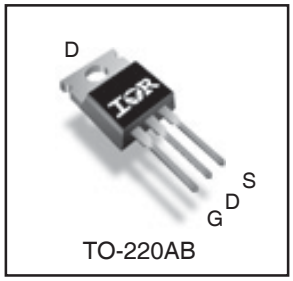
HEXFET® Power MOSFET

Applications
- High Efficiency Synchronous Rectification in SMPS
- Uninterruptible Power Supply
- High Speed Power Switching
- Hard Switched and High Frequency Circuits

V_{DSS}	100V
$R_{DS(on)}$ typ.	3.7mΩ
max.	4.5mΩ
I_D (Silicon Limited)	180A ①
I_D (Package Limited)	120A

Benefits
- Improved Gate, Avalanche and Dynamic dv/dt Ruggedness
- Fully Characterized Capacitance and Avalanche SOA
- Enhanced body diode dV/dt and dI/dt Capability





TO-220AB

G	D	S
Gate	Drain	Source

Absolute Maximum Ratings

Symbol	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, VGS @ 10V (Silicon Limited)	180①	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V (Silicon Limited)	130①	
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V (Wire Bond Limited)	120	
I_{DM}	Pulsed Drain Current ②	670	
P_D @T_C = 25°C	Maximum Power Dissipation	370	W
	Linear Derating Factor	2.5	W/°C
V_{GS}	Gate-to-Source Voltage	± 20	V
dv/dt	Peak Diode Recovery ④	5.3	V/ns
T_J	Operating Junction and	-55 to + 175	°C
T_{STG}	Storage Temperature Range		
	Soldering Temperature, for 10 seconds	300	
	(1.6mm from case)		
	Mounting torque, 6-32 or M3 screw	10lb·in (1.1N·m)	

Avalanche Characteristics

$E_{AS\ (Thermally\ limited)}$	Single Pulse Avalanche Energy ③	190	mJ
I_{AR}	Avalanche Current ②	See Fig. 14, 15, 22a, 22b	A
E_{AR}	Repetitive Avalanche Energy ⑤		mJ

Thermal Resistance

Symbol	Parameter	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case ⑨	—	0.402	°C/W
$R_{\theta CS}$	Case-to-Sink, Flat Greased Surface	0.50	—	
$R_{\theta JA}$	Junction-to-Ambient ⑧	—	62	

National Semiconductor

June 2006

LMC660
CMOS Quad Operational Amplifier

General Description

The LMC660 CMOS Quad operational amplifier is ideal for operation from a single supply. It operates from +5V to +15.5V and features rail-to-rail output swing in addition to an input common-mode range that includes ground. Performance limitations that have plagued CMOS amplifiers in the past are not a problem with this design. Input V_{OS}, drift, and broadband noise as well as voltage gain into realistic loads (2 kΩ and 600Ω) are all equal to or better than widely accepted bipolar equivalents.

This chip is built with National's advanced Double-Poly Silicon-Gate CMOS process.

See the LMC662 datasheet for a dual CMOS operational amplifier with these same features.

Features

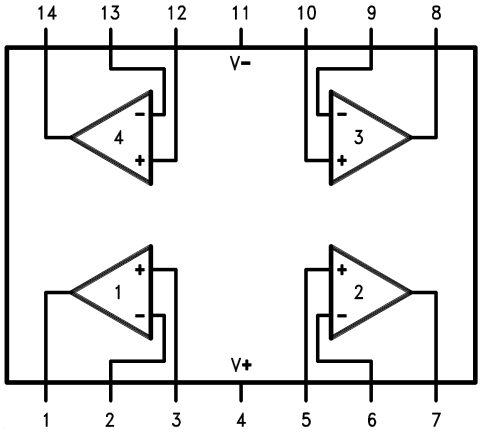
- Rail-to-rail output swing
- Specified for 2 kΩ and 600Ω loads
- High voltage gain: 126 dB
- Low input offset voltage: 3 mV
- Low offset voltage drift: 1.3 µV/˚C
- Ultra low input bias current: 2 fA
- Input common-mode range includes V^-
- Operating range from +5V to +15.5V supply
- I_{SS} = 375 µA/amplifier; independent of V^+
- Low distortion: 0.01% at 10 kHz
- Slew rate: 1.1 V/µs

Applications

- High-impedance buffer or preamplifier
- Precision current-to-voltage converter
- Long-term integrator
- Sample-and-Hold circuit
- Peak detector
- Medical instrumentation
- Industrial controls
- Automotive sensors

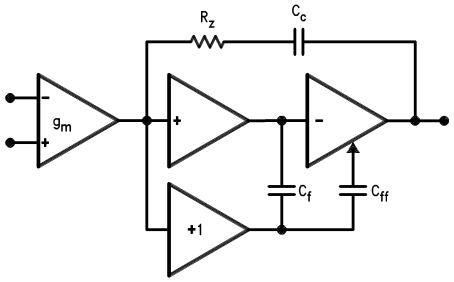
Connection Diagram

14-Pin SOIC/MDIP



00876701

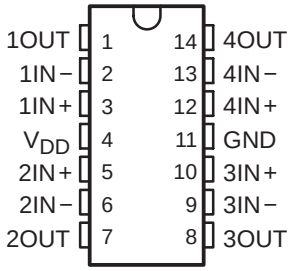
LMC660 Circuit Topology (Each Amplifier)



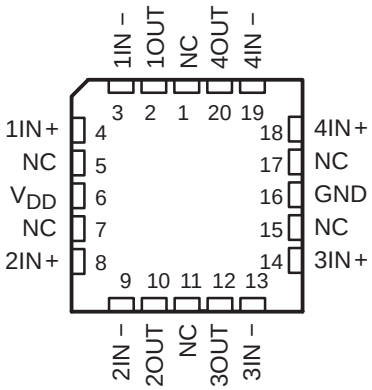
00876704

- **Trimmed Offset Voltage:**
  TLC279 . . . 900 μV Max at 25°C,
  V_{DD} = 5 V
- **Input Offset Voltage Drift . . . Typically 0.1 μV/Month, Including the First 30 Days**
- **Wide Range of Supply Voltages Over Specified Temperature Range:**
  0°C to 70°C . . . 3 V to 16 V
  −40°C to 85°C . . . 4 V to 16 V
  −55°C to 125°C . . . 4 V to 16 V
- **Single-Supply Operation**
- **Common-Mode Input Voltage Range Extends Below the Negative Rail (C-Suffix and I-Suffix Versions)**
- **Low Noise . . . Typically 25 nV/$\sqrt{Hz}$ at f = 1 kHz**
- **Output Voltage Range Includes Negative Rail**
- **High Input Impedance . . . 10^{12} Ω Typ**
- **ESD-Protection Circuitry**
- **Small-Outline Package Option Also Available in Tape and Reel**
- **Designed-In Latch-Up Immunity**

D, J, N, OR PW PACKAGE
(TOP VIEW)



FK PACKAGE
(TOP VIEW)



NC − No internal connection

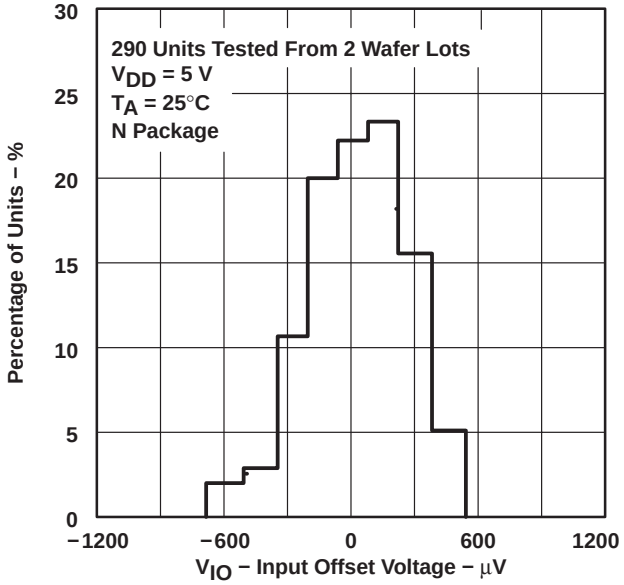
description

The TLC274 and TLC279 quad operational amplifiers combine a wide range of input offset voltage grades with low offset voltage drift, high input impedance, low noise, and speeds approaching that of general-purpose BiFET devices.

These devices use Texas Instruments silicon-gate LinCMOS™ technology, which provides offset voltage stability far exceeding the stability available with conventional metal-gate processes.

The extremely high input impedance, low bias currents, and high slew rates make these cost-effective devices ideal for applications which have previously been reserved for BiFET and NFET products. Four offset voltage grades are available (C-suffix and I-suffix types), ranging from the low-cost TLC274 (10 mV) to the high-precision TLC279 (900 μV). These advantages, in combination with good common-mode rejection and supply voltage rejection, make these devices a good choice for new state-of-the-art designs as well as for upgrading existing designs.

DISTRIBUTION OF TLC279
INPUT OFFSET VOLTAGE



LinCMOS is a trademark of Texas Instruments.

ACNV4506

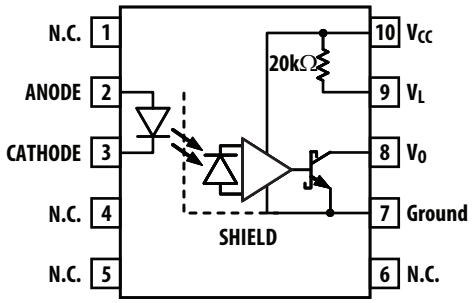
Intelligent Power Module and Gate Drive Interface Optocouplers

Data Sheet

Description

The ACNV4506 device contains a GaAsP LED optically coupled to an integrated high gain photo detector. Minimized propagation delay difference between devices makes these optocouplers excellent solutions for improving inverter efficiency through reduced switching dead time. Specifications and performance plots are given for typical IPM applications.

Functional Diagram



Note:
A 0.1 μF bypass capacitor must be connected between pins 7 and 10.

Truth Table

LED	V_O
ON	LOW
OFF	HIGH

Features

- Performance Specified for Common IPM Applications Over Industrial Temperature Range.
- Short Maximum Propagation Delays
- Minimized Pulse Width Distortion (PWD)
- Very High Common Mode Rejection (CMR)
- High CTR.
- Available in Widebody DIP10 and GulWing packages with 13.0 mm creepage and clearance.
- Safety Approval (pending):
  - UL Recognized with 7500 V_{rms} for 1 minute per UL1577.
  - CSA Approved.
  - IEC/EN/DIN EN 60747-5-2 Approved with V_{IORM} = 2262V_{peak}.

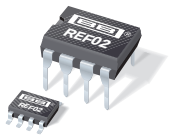
Specifications

- Wide operating temperature range: –40°C to 105°C.
- Typical propagation delay t_{PHL} = 200 ns, t_{PLH} = 350 ns
- Typical Pulse Width Distortion (PWD) = 150 ns.
- 30 kV/μs minimum common mode rejection (CMR) at V_{CM} = 1500 V.
- CTR = 90%(typ) at I_F = 10mA

Applications

- IPM Isolation
- Isolated IGBT/MOSFET Gate Drive
- AC and Brushless DC Motor Drives
- Industrial Inverters

CAUTION: *It is advised that normal static precautions be taken in handling and assembly of this component to prevent damage and/or degradation which may be induced by ESD.*



SBVS003B – JANUARY 1993 – REVISED JANUARY 2005

+5V Precision VOLTAGE REFERENCE

FEATURES

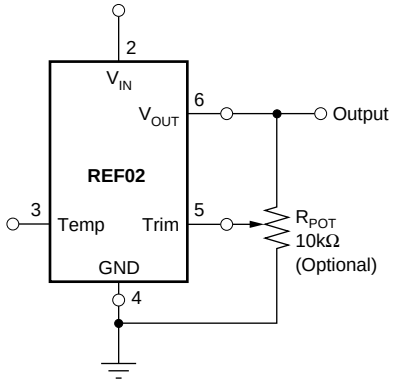
- **OUTPUT VOLTAGE: +5V ±0.2% max**
- **EXCELLENT TEMPERATURE STABILITY: 10ppm/°C max (–40°C to +85°C)**
- **LOW NOISE: 10μV_{PP} max (0.1Hz to 10Hz)**
- **EXCELLENT LINE REGULATION: 0.01%/V max**
- **EXCELLENT LOAD REGULATION: 0.008%/mA max**
- **LOW SUPPLY CURRENT: 1.4mA max**
- **SHORT-CIRCUIT PROTECTED**
- **WIDE SUPPLY RANGE: 8V to 40V**
- **INDUSTRIAL TEMPERATURE RANGE: –40°C to +85°C**
- **PACKAGE OPTIONS: DIP-8, SO-8**

APPLICATIONS

- **PRECISION REGULATORS**
- **CONSTANT CURRENT SOURCE/SINK**
- **DIGITAL VOLTMETERS**
- **V/F CONVERTERS**
- **A/D AND D/A CONVERTERS**
- **PRECISION CALIBRATION STANDARD**
- **TEST EQUIPMENT**

DESCRIPTION

The REF02 is a precision 5V voltage reference. The drift is laser trimmed to 10ppm/°C max over the extended industrial and military temperature range. The REF02 provides a stable 5V output that can be externally adjusted over a ±6% range with minimal effect on temperature stability. The REF02 operates from a single supply with an input range of 8V to 40V with a very low current drain of 1mA, and excellent temperature stability due to an improved design. Excellent line and load regulation, low noise, low power, and low cost make the REF02 the best choice whenever a 5V voltage reference is required. Available package options are DIP-8 and SO-8. The REF02 is an ideal choice for portable instrumentation, temperature transducers, Analog-to-Digital (A/D) and Digital-to-Analog (D/A) converters, and digital voltmeters.



+5V Reference with Trimmed Output