High Frequency MOSFET Based Three Phase Inverter For Drive Application Using TI's DSC

A Project Report

submitted by

JAGDISH SINGH

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

May 2016

THESIS CERTIFICATE

This is to certify that the thesis titled **High Frequency MOSFET Based Three Phase Inverter For Drive Application Using TI's DSC**, submitted by **Jagdish singh**, to the Indian Institute of Technology, Madras, for the award of the degree of **MASTER OF TECHNOLOGY**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. N Lakshminarasamma
(Project Guide)
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 24th May 2016

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my guide Assistant Professor Dr. N Lakshminarasamma, for her prolific encouragement and guidance. I am grateful to her for her timely advice and constructive criticism which made the course of my project an excellent learning experience.
I would like to thank faculty in Power Electronics group Dr. Krishna Vasudevan, Dr. Kamalesh Hatua, and Dr. Srirama Srinivas for their support during my course work through out my stay in IIT.
I also appreciate the help from Mrs. Jayasudha with the lab facilities and equipment. My appreciation to my fellow students in the Power Electronics laboratory, especially Kedarnath, Nithin, Subhransu, Nirija and Vijaya for spending their invaluable time with me in discussing about the project, goes well beyond words.
I thank my parents for bringing me up with the best education and moral values and for enabling me to pursue my dreams. I would like to extend my thanks to my friends RajKamal, Shubham, Vivek and Madhav for their companionship and encouragement which made my stay in the IIT Madras campus, a memorable one.

ABSTRACT

KEYWORDS: 3 Phase Inverter, BLDC motor Driver Circuitry.

BLDC motors are one of the motors rapidly gaining popularity. It has several advantages over brushed DC motor and Induction motor. To drive a three phase BLDC motor three phase inverter is needed. This project mainly concentrated on design and hardware implementation of three phase inverter with its driver circuitry and protection circuitry. Texas Instruments's TMS320f28335 Digital Signal Controller(DSC) is used to implement algorithm for electronic commutation in BLDC motor which is characterized by trapezoidal back-EMF. Programming of DSC s done in C language with Code Composer Studio (CCS) as its IDE.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

BLDC	Brushless Direct Current Motor
PMSM	Permanent Magnet Synchronous Motor
EMI	Electromagnetic Interference
DSC	Digital Signal Controller
PWM	Pulse width Modulation
SPWM	Sinusoidal Pulse width Modulation
ePWM	Enhanced Pulse Width Modulation
ADC	Analogue To Digital Converter
GPIO	General Purpose Input-Output
BEMF	Back Electromotive Force
MOSFET	Metal Oxide Field Effect Transistor
IGBT	Insulated Gate Bipolar Transistor
IC	Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
KVL	Kirchoff's Voltage Law
PCB	Printed Circuit Board
EAGLE	Easily Applicable Graphical Layout Editor
CSS	Code Composer Studio
IDE	Integrated Design Environment
RAM	Random Access memory
USB	Universal Serial Bus
I2C	Inter Integrated Circuit
CAN	Control Area Network
TBCLK	Time Base Clock
SYSCLKOUT	System Clock
RPM	Revolution Per Minute

CHAPTER 1

INTRODUCTION

1.1 Introduction

The motor or an electrical motor is a device that has brought about one of the biggest advancements in the fields of engineering and technology ever since the invention of electricity. An Electric motor is an Electric machine which converts electrical energy into mechanical energy. The reverse of this would be the conversion of mechanical energy into electrical energy and is done by an electric generator. The very basic principal of functioning of an electrical motor lies on the fact that force is experienced in the direction perpendicular to magnetic field and the current, when field and current are made to interact with each other.

1.2 Classification of Motor

There are different types of motor have been developed for different specific purposes but electric motors are broadly classified into two categories on basis of their power source type:
1)DC motors(driven by DC supply)
2)AC motors(driven by AC supply)

AC motors are one which use sinusoidal voltage to drive them where as stiff or constant voltage is require for DC motor. Further AC motor is divided into two parts Asynchronous motor and Synchronous motor, synchronous motors are further divided into BLDC and PMSM. Both BLDC and PMSM use permanent magnet at rotor instead of field winding. BLDC motor is not part of DC machine it require DC power source but finally it require AC wave for operation which is obtained by inverter circuitry. DC motor is divide into two parts commutator based motor and homopolar motors. In this report we will concentrate on BLDC motors only. **Brushed DC** electric motor generates torque directly from DC power supplied to the motor by using internal commutation, stationary magnets (permanent or electromagnets), and rotating electrical magnets where as **Brush-less** DC motors use a rotating permanent magnet in the rotor, and stationary electrical current/coil magnets on the motor housing for the stator. A motor controller converts DC to AC. This design is mechanically simpler than that of brushed motors because it eliminates the complication of transferring power from outside the motor to the spinning rotor. The motor controller can sense the rotor's position via Hall effect sensors or similar and precisely control the timing, phase of the current in the rotor coils.

The BLDC motor has several advantages over brushed motor. Table 1.1 summarize the advantages of the BLDC motor when compared against DC motor.

Table 1.1: Comparison between BLDC motor and DC motor

$Feature$	$BLDCMotor$	$BrushedDCMotor$	$ActualAdvantage$
Commutation	Electronic commutation based on rotor position information	Mechanical brushes and commutator	Electronic switches replace the mechanical devices
Efficiency	High	Moderate	Voltage drop on electronic device is smaller than that on brushes.
Maintenance	Little/None	Periodic	No brushes/commutator maintenance.
Output Power/Frame Size (Ratio)	High	Moderate/Low	Modern permanent magnet and no rotor losses.
Speed/Torque Characteristics	Flat	Moderately flat	No brush friction to reduce useful torque.
Dynamic Response	Fast	Slow	Lower rotor inertia because of permanent magnets.
Speed Range	High	Low	No mechanical limitation imposed by brushes or commutator.
Electric Noise	Low	High	No arcs from brushes to generate noise, causing EMI problems.
Lifetime	Long	Short	No brushes and commutator.

1.3 Organization of Project Report

Chapter 2 deals with comparison between BLDC and PMSM, construction and operation of 3 phase BLDC motor, mathematical model and type of switching used for power transistor and commutation table for clock wise and anti-clock wise rotation.
Chapter 3 deals with hardware implementation for driver, opto-coupler, voltage sensor and current sensor with their PCB schematic and board layout.
Chapter 4 presents brief introduction of TMS320f28335, its basic program flow chart and various modules used in software implementation like GPIO, ePWM, ADC and Interrupt. Also results are presented with resistive load and motor load .

CHAPTER 2

CONSTRUCTIN AND OPERATION OF BLDC MOTOR

The BLDC motor, figure2.1, is an AC synchronous motor with permanent magnets on the rotor (moving part) and windings on the stator (fixed part). Permanent magnets create the rotor flux and the energized stator windings create electromagnet poles. The rotor (equivalent to a bar magnet) is attracted by the energized stator phase. By using the appropriate sequence to supply the stator phases, a rotating field on the stator is created and maintained. This action of the rotor chasing after the electromagnet poles on the stator is the fundamental action used in synchronous permanent magnet motors. The lead between the rotor and the rotating field must be controlled to produce torque and this synchronization implies knowledge of the rotor position.

On the stator side, three phase motors are the most common. These offer a good com-

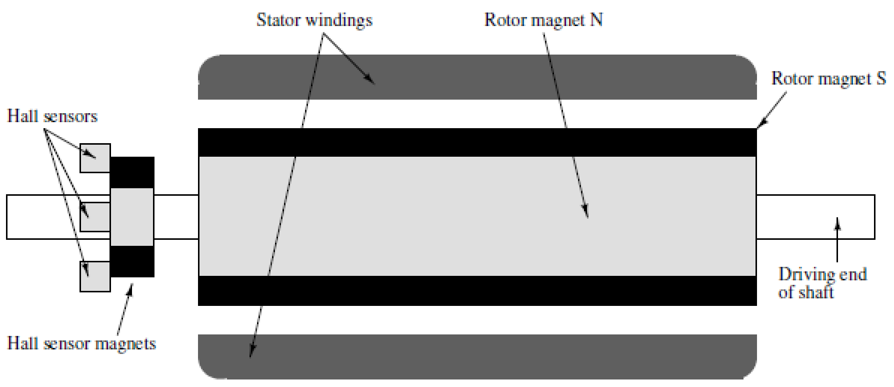


Figure 2.1: BLDC motor transverse section

promise between precise control and the number of power electronic devices required to control the stator currents. For the rotor, a greater number of poles usually create a greater torque for the same level of current. On the other hand, by adding more magnets, a point is reached where, because of the space needed between magnets, the torque no longer increases. The manufacturing cost also increases with the number of poles. As a consequence, the number of poles is a compromise between cost, torque and volume.

2.1 Construction of BLDC motor

There are three classifications of the BLDC motor: single-phase, two-phase and three-phase. This discussion assumes that the stator for each type has the same number of windings. The single-phase and three-phase motors are the most widely used. Figure 2.2 shows the simplified cross section of a three-phase BLDC motor.

A **Rotor** consists of a shaft and a hub with permanent magnets arranged to form be-

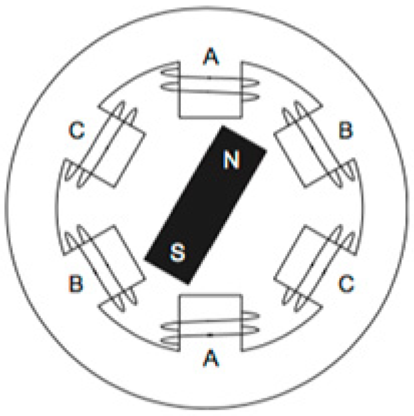


Figure 2.2: Three Phase BLDC motor.

tween two to eight pole pairs that alternate between north and south poles. Figure 2.3 shows cross sections of three kinds of magnets arrangements in a rotor. There are multiple magnet materials, such as ferrous mixtures and rare-earth alloys. Ferrite magnets are traditional and relatively inexpensive, though rare-earth alloy magnets are becoming increasingly popular because of their high magnetic density. The higher density helps to shrink rotors while maintaining high relative torque when compared to similar ferrite magnets.

The position of the rotor is determined by **Hall sensors** which are normally mounted on the stationary part. Hall sensors are placed 120° apart and six commutation sequences are achieved using possible combinations of three hall sensors. The hall sensor requires DC power source of 5V. The main drawback of using hall sensors is reduced immunity to electromagnetic interference and noises.

There are two types of **Stator** windings: trapezoidal and sinusoidal, which refers to the shape of the back electromotive force (BEMF) signal. The shape of the BEMF is determined by different coil interconnections and the distance of the air gap. As their names indicate, the trapezoidal motor gives a back EMF in trapezoidal fashion and the sinusoidal motor′s back EMF is sinusoidal, as shown in Figure 2.4 and Figure 2.5. In

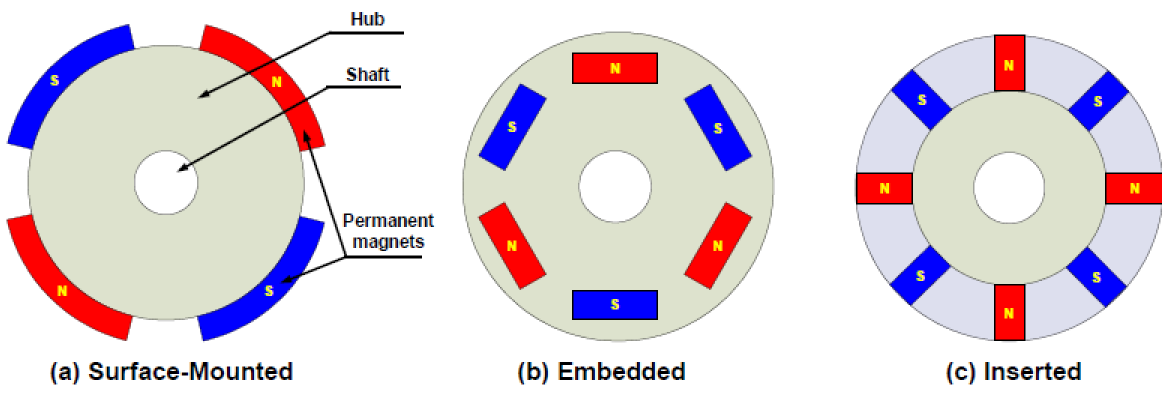


Figure 2.3: Rotor Magnets Cross-Sections

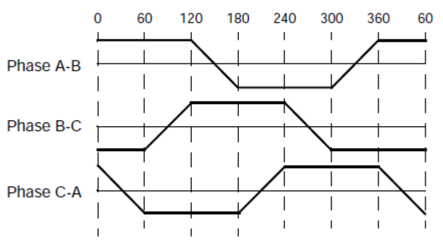


Figure 2.4: Trapezoidal Back EMF

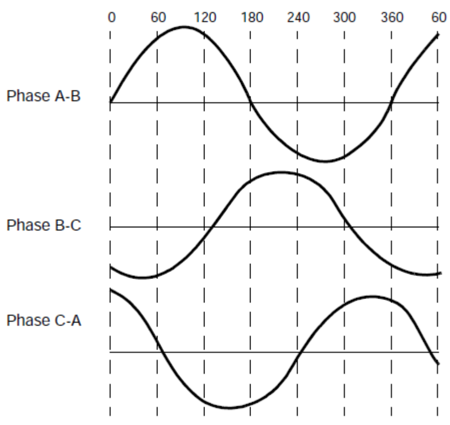


Figure 2.5: Sinusoidal Back EMF

addition to the back EMF, the phase current also has trapezoidal and sinusoidal variations in the respective types of motor. This makes the torque output by a sinusoidal motor smoother than that of a trapezoidal motor. However, this comes with an extra cost, as the sinusoidal motors take extra winding interconnections because of the coils distribution on the stator periphery, thereby increasing the copper intake by the stator windings.

Permanent magnet synchronous motors can be classified in Brush-less Direct Current Motor (BLDC) and Permanent Magnet Synchronous Motor (PMSM) depending on back-emf profiles. Both BLDC and PMSM motors have permanent magnets on the rotor but differ in the flux distributions and back-emf profiles. To get the best performance out of the synchronous motor, it is important to identify the type of motor in order to apply the most appropriate type of control as described in the next chapters. Table 2.1 compare BLDC with PMSM on various aspects.

Table 2.1: Comparison of BLDC and PMSM motors

$BLDC$	$PMSM$
Synchronous machine	Synchronous machine.
Fed with direct currents	Fed with sinusoidal currents.
Trapezoidal Bemf	Sinusoidal Bemf.
Stator Flux position commutation each 60°	Continuous stator flux position variation.
Only two phases ON at the same time	Possible to have three phases ON at the same time.
Torque ripple at commutations	No torque ripple at commutations.
Low order current harmonics in the audible range	Less harmonics due to sinusoidal excitation.
Higher core losses due to harmonic content	Lower core loss.
Less switching losses	Higher switching losses at the same switching freq.
Control algorithms are relatively simple	Control algorithms are mathematically intensive.

2.2 Opertaion of 3 Phase BLDC Motor

The Three phase BLDC motor is operated in two-phase mode, i.e two phases that produce the highest torque are energized while the third phase if off. The voltage must be properly applied to the two phases of the three-phase winding system so that the angle between the stator flux and the rotor flux is kept close to $90°$, to get the maximum generated torque. Due to this fact, the motor requires electronic control based on actual rotor angle position for proper operation. The signal from the position sensor(Hall sensor) produce a three digit number that changes every 60° (electrical degrees) as shown in figure 2.6, also shows ideal current and back-emf waveform.

A standard 3 phase power stage is used for the common 3 phase BLDC motor, as illustrated in Figure 2.7. The power stage utilizes six power transistors with switching in either the independent mode or complementary, which are discussed in the following sections.

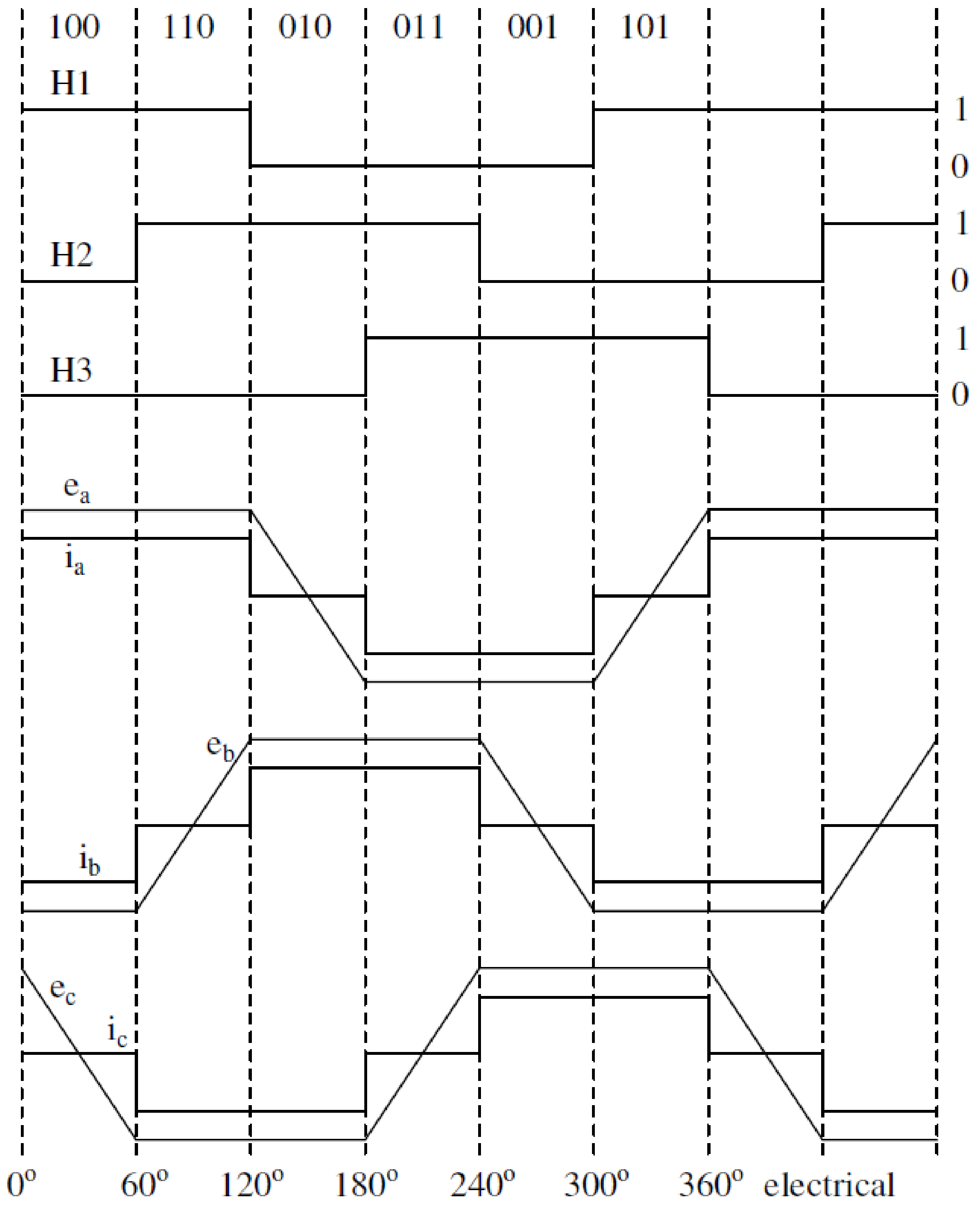


Figure 2.6: Three Phase BLDC motor.Ideal back-emf′s, phase currents and hall signals.

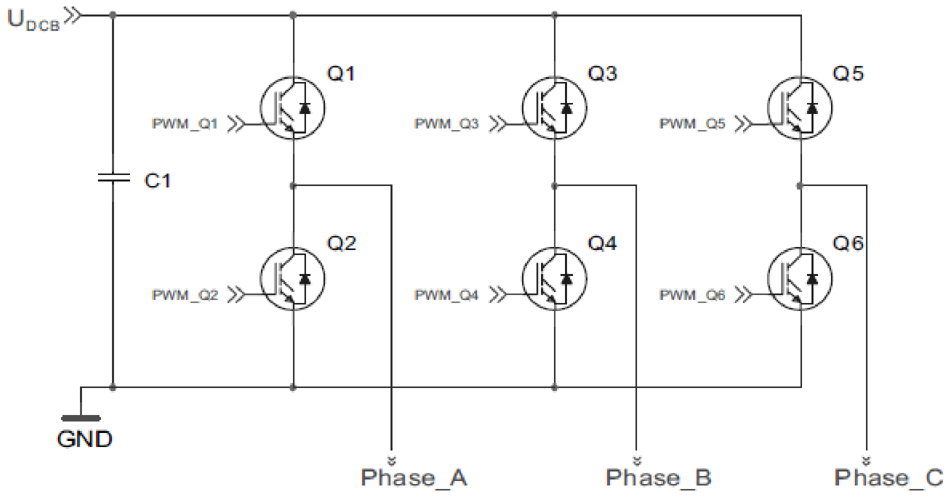


Figure 2.7: Three Phase BLDC motor.Ideal back-emf′s, phase currents and hall signals.

2.2.1 Independent Switching of Power Transistors

During independent switching, only two transistors are switched on when current is conducted from the power supply to the phase of the BLDC motor. In one phase, the top transistor is switched on; in the second phase, the bottom transistor is switched on and the third phase is not powered. During freewheeling, all transistors are switched off. Figure 2.8 depicts independent switching.

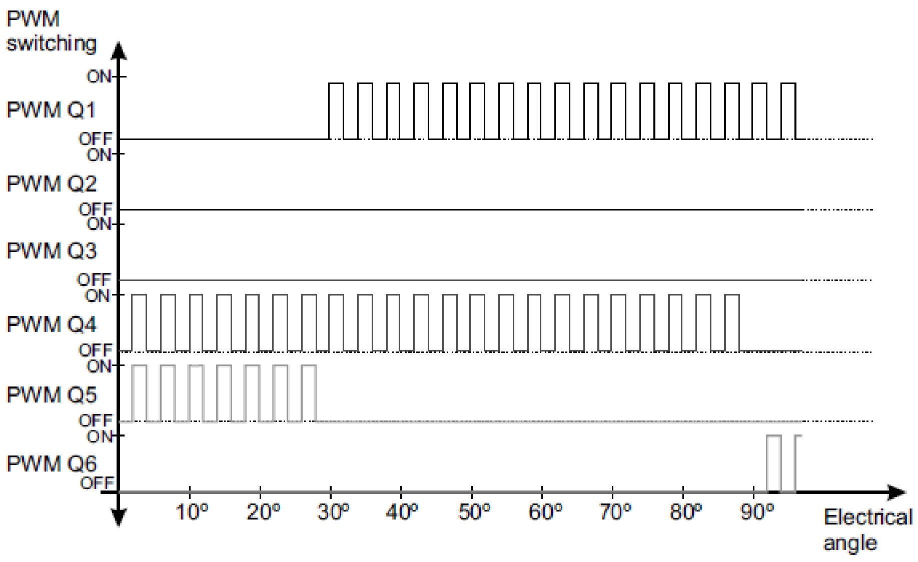


Figure 2.8: Independent Switching of Power Transistors.

2.2.2 Complementary Switching of Power Transistors

During complementary switching, two transistors are switched on when the phase of the BLDC motor is connected to the power supply. One primary difference occurs during freewheeling. During independent switching, all transistors are switched off. The current continues to flow in the same direction through freewheeling diodes until it falls to zero. In complementary switching, the complementary transistors are switched on during freewheeling, so the current may be able to flow in the opposite direction. Figure 2.9 depicts complementary switching.

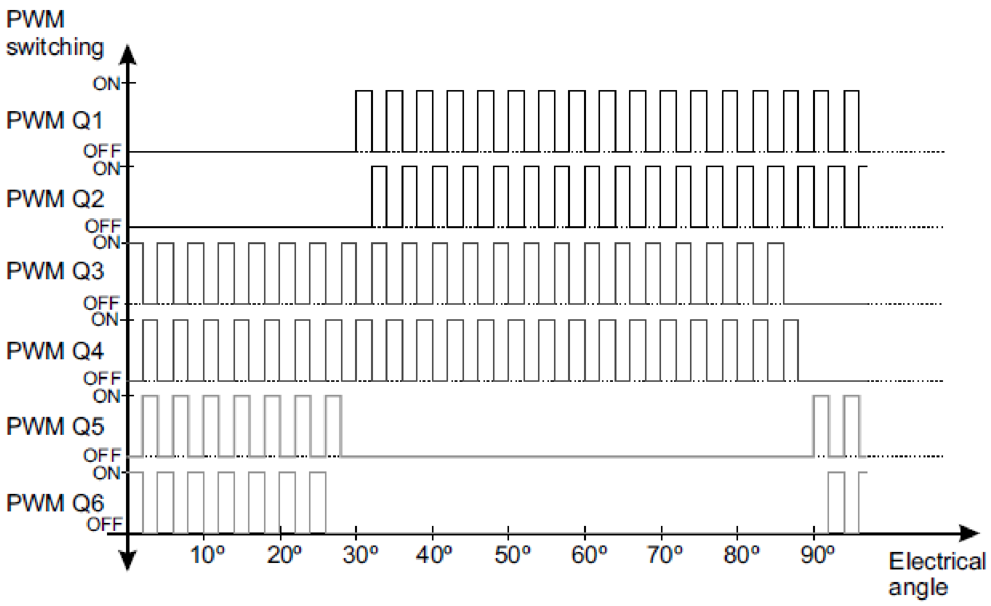


Figure 2.9: Complementary Switching of Power Transistors.

2.3 Commutation

Commutation provides the creation of a rotation field. It is necessary to keep the angle between stator and rotor flux close to 90° for a BLDC motor to operate properly. Six-step control creates a total of six possible stator flux vectors. The stator flux vector must be changed at a certain rotor position. The rotor position is usually sensed by Hall sensors. The Hall sensors generate three signals that also comprise six states. Each of Hall sensors states corresponds to a certain stator flux vector. All Hall sensor states with corresponding voltages are illustrated in table 2.2 for clockwise rotation and table 2.3.

Table 2.2: Commutation Sequence for Clockwise Rotation.

$Rotorposition$	H_a	H_b	H_c	$PhaseA$	$PhaseB$	$PhaseC$
0° -60°	1	0	0	-V_{DC}	+V_{DC}	NC
60° -120°	1	0	1	NC	+V_{DC}	-V_{DC}
120°-180°	0	0	1	+V_{DC}	NC	-V_{DC}
180°-240°	0	1	1	+V_{DC}	-V_{DC}	NC
240°-300°	0	1	0	NC	-V_{DC}	+V_{DC}
300°-360°	1	1	0	-V_{DC}	NC	+V_{DC}

Table 2.3: Commutation Sequence for Anticlockwise Rotation.

$Rotorposition$	H_a	H_b	H_c	$PhaseA$	$PhaseB$	$PhaseC$
0° - 60°	1	0	0	+V_{DC}	-V_{DC}	NC
60°-120°	1	0	1	+V_{DC}	NC	-V_{DC}
120°-180°	0	0	1	NC	+V_{DC}	-V_{DC}
180°-240°	0	1	1	-V_{DC}	+V_{DC}	NC
240°-300°	0	1	0	-V_{DC}	NC	+V_{DC}
300°-360°	1	1	0	NC	-V_{DC}	+V_{DC}

Commutation allows the movement of the motor where as speed at which the actual rotor is forced to the next position is determined by the strength of this magnetic force, and this is determined by the voltage applied to the stator windings. By using PWMs at a higher frequency than the commutations, the amount of voltage applied to the stator can be easily controlled, therefore the speed of the motor can be controlled.

2.4 Mathematical Model of BLDC motor

Three phase star connected BLDC motor can be described by following four equations:

$$v_a = R(i_a - i_b) + L\frac{d}{dt}(i_a - i_b) + e_a - e_b \tag{2.1}$$

$$v_b = R(i_b - i_c) + L\frac{d}{dt}(i_b - i_c) + e_b - e_c \tag{2.2}$$

$$v_c = R(i_c - i_a) + L\frac{d}{dt}(i_c - i_a) + e_c - e_a \tag{2.3}$$

$$T_e = kfw_m + J\frac{dw_m}{dt} + T_L \tag{2.4}$$

The symbols v, i and e denote the phase-to-phase voltages, phase currents and back-emf′s respectively, in the three phase a, b and c. The resistance R and the inductance L are per phase values and T_e and T_L are the electrical and load torque. J is the rotor inertia, k_f is a friction constant and w_m is the rotor speed. The back-emf′s and the electrical torque can be expressed as following:

$$e_a = \frac{k_e}{2} w_m F(\theta_e) \tag{2.5}$$

$$e_b = \frac{k_e}{2} w_m F(\theta_e - \frac{2\pi}{3}) \tag{2.6}$$

$$e_c = \frac{k_e}{2} w_m F(\theta_e - \frac{4\pi}{3}) \tag{2.7}$$

$$T_e = \frac{k_t}{2}[F(\theta_e)i_a + F(\theta_e - \frac{2\pi}{3})i_b + F(\theta_e - \frac{4\pi}{3})i_c] \tag{2.8}$$

where k_e and k_t are the back-emf constant and the torque constant. The electrical angle θ_e is equal to the rotor angle times the number of pole pairs $\theta_e \frac{p}{2}\theta_m$. The function $F(\cdot)$ gives the trapezoidal waveform of the back-emf. One period of this function can be written as

$$F(\theta_e) = \begin{cases} 1 & \text{if } 0 \le \theta_e < \frac{2\Pi}{3} \\ 1 - \frac{6}{\pi}(\theta_e - \frac{2\pi}{3}) & \text{if } \frac{2\pi}{3} \le \theta_e < \Pi \\ -1 & \text{if } \pi \le \theta_e < \frac{5\Pi}{3} \\ 1 + \frac{6}{\pi}(\theta_e - \frac{5\pi}{3}) & \text{if } \frac{5\pi}{3} \le \theta_e < 2\Pi \end{cases}$$

since each voltage equation is linear combination of other two voltage equation so only two equation is needed. By using current equation

$$i_a + i_b + i_c = 0 \tag{2.9}$$

the voltage equations becomes

$$v_{ab} = R(i_a - i_b) + L\frac{d}{dt}(i_a - i_b) + e_a - e_b \tag{2.10}$$

$$v_{bc} = R(i_a + 2i_b) + L\frac{d}{dt}(i_a + 2i_b) + e_b - e_c \tag{2.11}$$

and then complete model become

$$\begin{pmatrix} i_a^{'} \\ i_b^{'} \\ w_m^{'} \\ \theta_m^{'} \end{pmatrix} = \begin{pmatrix} \frac{-R}{L} & 0 & 0 & 0 \\ 0 & \frac{-R}{L} & 0 & 0 \\ 0 & 0 & \frac{-R}{L} & 0 \\ 0 & 0 & 0 & \frac{-R}{L} \end{pmatrix} \begin{pmatrix} i_a \\ i_b \\ w_m \\ \theta_m \end{pmatrix} + \begin{pmatrix} \frac{2}{3L} & \frac{1}{3L} & 0 \\ \frac{-1}{3L} & \frac{1}{3L} & 0 \\ 0 & 0 & \frac{1}{J} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_{ab} - e_{ab} \\ v_{bc} - e_{bc} \\ T_e - T_L \end{pmatrix}$$

CHAPTER 3

HARDWARE ORGANIZATION

3.1 Introduction

Figure 3.1 depict hardware implementation of hall sensored three phase bldc motor with Texas instrument TMS320f28335 as it DSC and voltage source inverter. Whole hardware circuitry is divided into two PCB board, first PCB contain driver circuitry for mosfet, power input port for IC powering, isolation circuitry. Second board contain three phase inverter with MOSFET as switching device, current sensor for current sensing and over current protection signal, Digital AND gate to merge all fault signal to a single fault signal. Two different PCB is used to separate low power component from high power component. To carry signal from Driver PCB board to power PCB board or other way ribbon cable is used. Hardware contain both surface mount and dip package for IC′s placement. Dip package is used for MOSFET driver, Digital AND gate and opto-coupler. for current sensor and 3 phase inverter surface mount package is used. Main electric power to power hardware is obtained from DC power supply. Each component and its application here is explained in brief. Table 3.1 shows input design specification which is used to select MOSFET module and other circuitry.

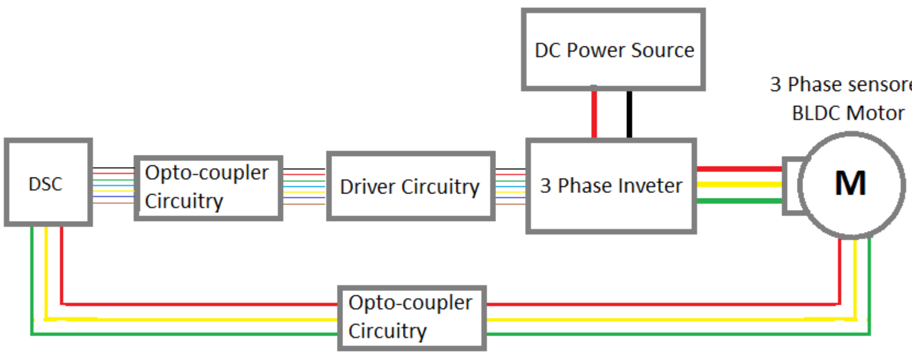


Figure 3.1: Block diagram of hardware implementation

Table 3.1: Design Input specification

$Parameter$	$Value$
Motor$'$s power rating	80 Watt
Motor$'$s voltage rating	24 V
Current rating	4.43 A
DC bus ripple	3 %
PWM switching frequency	100Khz

3.2 Three phase inverter

Three phase inverter module is main part of hardware which handle power to drive motor. For our project purpose 6-IN-1 MOSFET (Metal Oxide Semiconductor Field Effect Transistor) based module, GMM3x60-015X2 figure 3.2 from IXYS, is used as switching device instead of IGBT, because they have low Rds on value and higher switching frequency. Mosfet based module, which is surface mount package, is selected instead of 6 single MOSFET because of ease in implementation and space management on PCB board. This type of MOSFET module is suitable in automobile industry for electric power steering, starter generator or for low power electric vehicles.

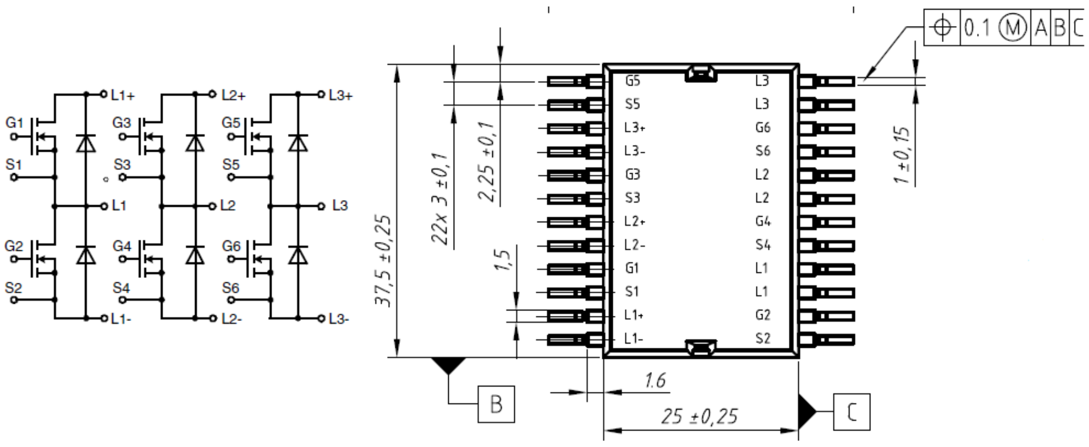


Figure 3.2: Three phase full Bridge Mosfet Module

Table 3.2 shows electrical specification for MOSFET module. $V_{GS(th)}$ is threshold voltage, the minimum gate-to-source voltage differential that is needed to create a conducting path between the source and drain terminals. R_{DCon} is on-state resistance of a power MOSFET. Turn-on delay, T_r, is the time taken to charge the input capacitance of the device before drain current conduction can start.

Table 3.2: Electrical ratings for MOSFET Module(GMM3x60-015X2)

$Parameter$	$Value$	$Units$
V_{DSS}	150	V
I_D	25	A
R_{DCon}	19	$m\Omega$
V_{GS}	15	V
$V_{GS(th)}$	4.5	V
Q_g	97	nC
Q_{gs}	29	nC
Q_{gd}	30	nC
T_r	50	ns
T_f	25	ns
E_{on}	0.25	mJ
E_{off}	0.05	mJ
Weight	25	g

Similarly, turn-off delay,T_f, is the time taken to discharge the capacitance after the after is switched off. Q_g represent gate charge require to on-off MOSFET which is provided by MOSFET driver.

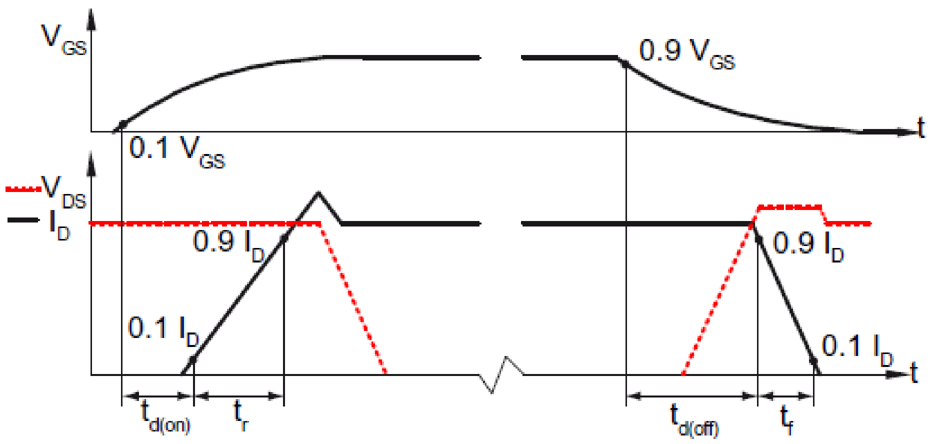


Figure 3.3: Definition of switching times

3.2.1 Mosfet gate driver circuitry

In three phase full bridge circuit we have both high side and low side MOSFET to drive such a system we require high-low MOSFET driver IC and for MOSFET **??** with above specification, we choose IR2110 high-low side driver which gives max current of 2 Amp to charge gate of MOSFET. It comes with both surface mount and Dip package for this project we are using Dip package. Its logic input is compatible with both standard CMOS or LSTTL output, down to 3.3v logic. IC require two different power supply VCC(15V) and VDD(5V), first one is termed as Logic supply(MOSFET V_{GS} dependent) and other is used to power driver itself. External electrolytic capacitor is used between VCC and VSS(gnd) as decoupling capacitor to filter out noise in power supply which may affect operation of driver IC.

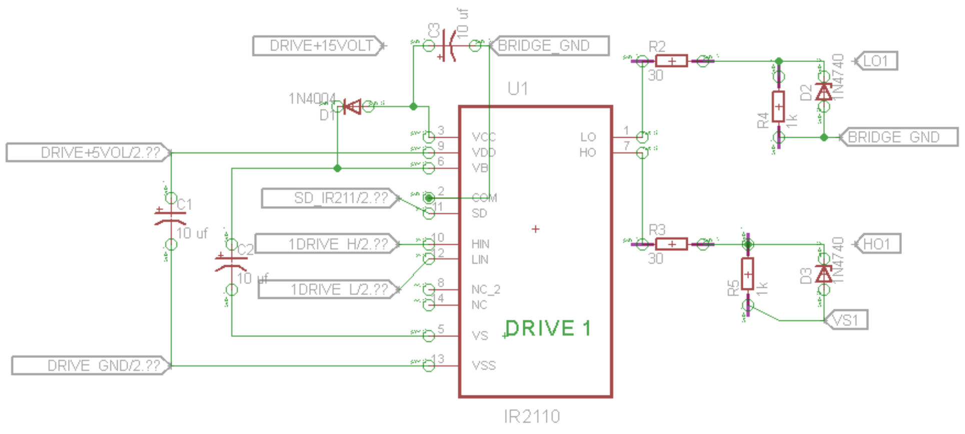


Figure 3.4: Typical connection diagram

IR2110 require logic input voltage of greater than VSS(gnd) but less than VDD. SD pin(Input) is used to shutdown driver output in case of fault by giving it high signal equals to VDD but in this project we are direct shutdown of PWM signal from DSC itself. D1, C2 (electrolytic capacitor) are part of boost-trap circuitry. A large enough capacitance must be chosen for C2 so that it can supply the charge required to keep MOSFET on for all the time. C2 must also not be too large that charging is too slow and the voltage level does not rise sufficiently to keep the MOSFET on. The higher the on time, the higher the required capacitance. Thus, the lower the frequency, the higher the required capacitance for C2. Resistor R2 and R3 are gate current-limiting resistor. D2 and D3 are zener diodes used between gate and source of high and low MOSFET it used in reverse mode to limit differential voltage between gate and source, values of these zener diode depend on V_{GS} rating of MOSFET, table 3.2 i.e 15 volt. Resistor R4 and R5 is used give a path to discharge gate charge when MOSFET is turned off. Table shows lead definitions of IR2110

Table 3.3: Lead definitions of IR2110

Symbol	*Discription*
VDD	Logic supply
HIN	Logic input for high side gate driver output (HO), in phase
SD	Logic input for shutdown
LIN	Logic input for low side gate driver output (LO), in phase
VSS	Logic ground
VB	High side floating supply
HO	High side gate drive output
VS	High side floating supply return
VCC	Low side supply
LO	Low side gate drive output
COM	Low side return

3.2.2 Opto-coupler

Opto-coupler or optical isolator, is a component that transfers electrical signals between two isolated circuits by using light. Opto-isolators prevent high voltages from affecting the system receiving the signal. In this project total of nine opto-coupler is used to isolate DSC, i.e TMS320f28335, from power circuitry. Out of these nine, six opto-couplers, shown in figure 3.5, are used to carry gating signal from DSC to driver′s input pin, with opto-isolation and other three opto-couplers, shown in figure3.6, are used carry to hall signals from motor hall sensors to DSC with opto-isolation, which is used to generate PWM to make proper commutation.

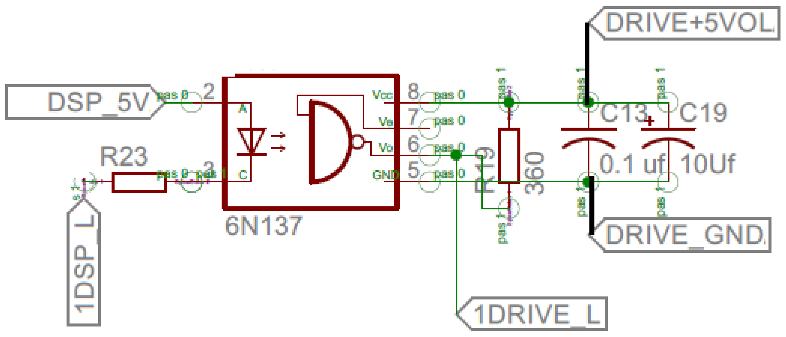


Figure 3.5: Opto-coupler between DSC and IR2110

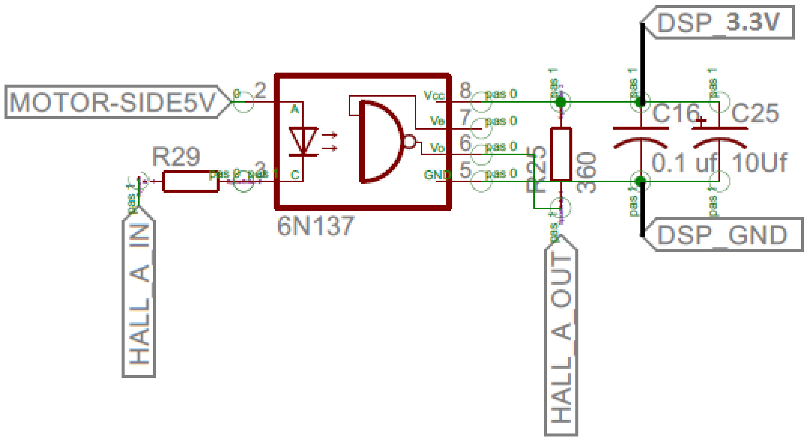


Figure 3.6: Opto-coupler between DSC and motor hall sensor

For above description we choose single channel high speed logic gate Opto-coupler, 6N137 from FAIRCHILD semiconductors. According to truth table 3.4 of Opto-coupler when enable in not connect output is negation to input, i.e when input is high output is low and vice-versa.

In this project we did some modification to get same output as input in this configu-

Table 3.4: Actual Opto-coupler′s truth table(PIN 3 = gnd)

$Input(PIN2)$	$Enable$	$Output$
H	H	L
L	H	H
H	L	H
L	L	H
H	NC	L
L	NC	H

ration enable pin is in NC mode and a fix 5V voltage is given to pin 2 of Opto-coupler and input which is to isolate is given to pin 3, truth table of modified circuit is shown in table 3.5.

Table 3.5: Modified Opto-coupler′s truth table(PIN 2= 5V or 3.3V)

$Input(PIN3)$	$Enable$	$Output$
H	H	H
L	H	L
H	L	H
L	L	H
H	NC	H
L	NC	L

Resistors R29 and R23 is used limit current through emitter of opto-coupler to 8 mA.
for driver-dsc opto-coupler:

$$3.3V = 8 * 10^{-3} * R$$

$$R = 412.5\Omega$$

for Motor hall-dsc opto-coupler:

$$5V = 8 * 10^{-3} * R$$

$$R = 625\Omega$$

On output side of Opto-coupler decoupling capacitor of value 10μF and 0.1μF are to be between used VCC and Gnd to make stiff voltage . In case of drive-dsc opto-coupler VCC is 5V, because it require 5V for driver IC to sense as high input but for Motor hall-dsc opto-coupler VCC is 3.3 V, because 3.3 V is max input voltage limit for DSC.

3.2.3 Current sensing

For current sensing $ACS709$ from $Allegro^{TM}$ is used, figure 3.7, small surface mount package is used to mount on PCB. It consists of precision linear hall sensor integrated circuit with copper conduction path, applied current flows through this copper conduction path, and the analog output voltage from the Hall sensor IC linearly tracks the magnetic field generated by the applied current. Resistance for this copper track is less than $1\text{m}\Omega$ so power loss is very low. The voltage on the Over-current Input V_{OC} pinis used to define an over-current fault threshold for the device. When current flowing through the copper conduction path (between the IP+ and IP- pins) exceeds this threshold, fault pin, Pin 19, will transit to logic low state. According to device datasheet to enable fault sensing $Fault_{EN}$ Pin must be greater than $VCC \times 0.8 = 3.2V$.

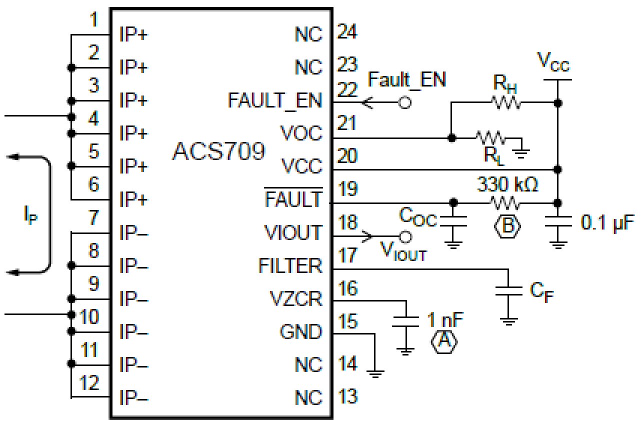


Figure 3.7: The $Allegro^{TM}$ ACS709 current sensor

Resistor R_H and R_L is used to set voltage on V_{OC} pin, which is to set over-current fault threshold for device. Value for can be calculated as follow:

$$V_{OC} = Sens \times I_{OC}$$

where V_{OC} is in mV, $Sens$ in mV/A i.e is 56mV/A, and I_{OC} over-curren tfault switch-point in A, i.e is 10A.

$$V_{OC} = 0.560V$$

further:

$$V_{OC} = VCC \times \frac{R_L}{R_L + R_H}$$

wher VCC is supply voltage, i.e is 5V.

$$\frac{R_L}{R_L + R_H} = \frac{56}{500}$$

so we choose $R_L = 56k\Omega$ and $R_H = 444k\Omega$.

3.2.4 Voltage sensing

For voltage sensing $Avago's\ ACPL - C870$ voltage sensor is used, figure 3.8, output of which is optically isolated. It has input voltage sensing range of 2V and high $1G\Omega$ input impedance, which makes it well suitable for isolated voltage sensing. to sense voltage above 2V, a resistive voltage divider is needed to scale sensing voltage in given range.

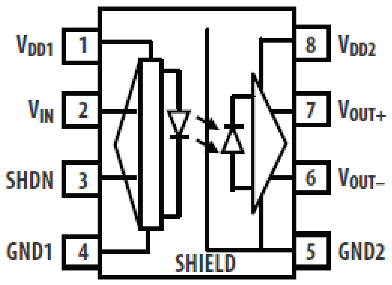


Figure 3.8: $ACPLC870$ voltage sensor$'s$ functional diagram

Voltage gain lies in rang of 0.97 to 1.03. When shut down is enable, i.e $V_{SD} = 5V$, V_{out+} and V_{out-} becomes 0V and 2.46V respectively. Table 3.6 describe pin assignment:

Table 3.6: ACPLC870 voltage sensor′s pin description.

$PinNo.$	$Symbol$	$Description$
1	VDD1	Supply voltage for input side(4.5 V to 5.5 V), relative to GND1
2	VIN	Voltage input
3	SHDN	Shutdown pin (Active High)
4	GND1	Input side ground
5	GND2	Output side ground
6	VOUT-	Negative output
7	VOUT+	Positive output
8	VDD2	Supply voltage for output side(3 V to 5.5 V), referenced to GND2

3.2.5 DC Link Capacitor design

DC-link capacitor is used to obtain a steady bus voltage, electrolytic capacitor is used to filter out low frequency voltage ripple and ceramic capacitor is used to filter out high frequency voltage ripple. Following expression is used to calculate minimum value of electrolytic capacitor:

$$C_{min} = \frac{I_{motorpeak}}{\Delta V f_s} \tag{3.1}$$

where $I_{motorpeak}$, is the peak motor current i.e 4.43Amp, $\Delta V$is the maximum allowed voltage ripple i.e 0.7V and f_s is the switching frequency i.e 100Khz.

$$C_{min} = 63.2\mu F$$

3.2.6 PCB Layout

As mention in section 3.1, whole hardware circuitry is divided into two PCB board for low power and high power components. For schematics and board layout Easily Applicable Graphical Layout Editor commonly known as $EAGLE$ from CadSoft is used. Trace width for signal trace and power trace for PCB for a given current is calculated using formulas from IPC-2221(formely IPC-D-275) which is as follow:
First, the Area is calculated:

$$Area[mils^2] = (Current[Amps]/(k * (TempRise[deg.C])^b))^(1/c) \tag{3.2}$$

Then, the Width is calculated:

$$Width[mils] = Area[mils^2]/(Thickness[oz] * 1.378[mils/oz]) \tag{3.3}$$

Where k, b, and c are constants resulting from curve fitting to the IPC-2221 curves. For internal layers: k = 0.024, b = 0.44, c = 0.725 and external layers: k = 0.048, b = 0.44, c = 0.725.
For this project only external layers is used for traces. Current value is 10Amps and thickness is 3 oz/ft^2. Ambient temperature is taken as $25°C$ and temperature rise is as $10°C$. By using above formulas and specification require results are as shown in table 3.7 :

Table 3.7: Results for External layer(for 1cm trace length)

Quantity	*Value*	*Unit*
Required Trace Width	2.40	mm
Resistance	0.000701	Ohms
Voltage Drop	0.00701	Volts
Power Loss	0.0701	Watts

Actual schematics and board layout for driver board and power board is presented in Appendix A and Appendix B respectively.

CHAPTER 4

SOFTWARE IMPLEMENTATION AND RESULTS

4.1 Introduction

The Texas Instrument′s TMS320f28335 is the DSC used for software implementation of motor control,it is member of $C2000^{TM}$ generation. TMS320F28335 has Harvard architecture,i.e separate memory space and buses for code and data. Figure 4.1 shows block diagram with various peripherals and hardware architecture:

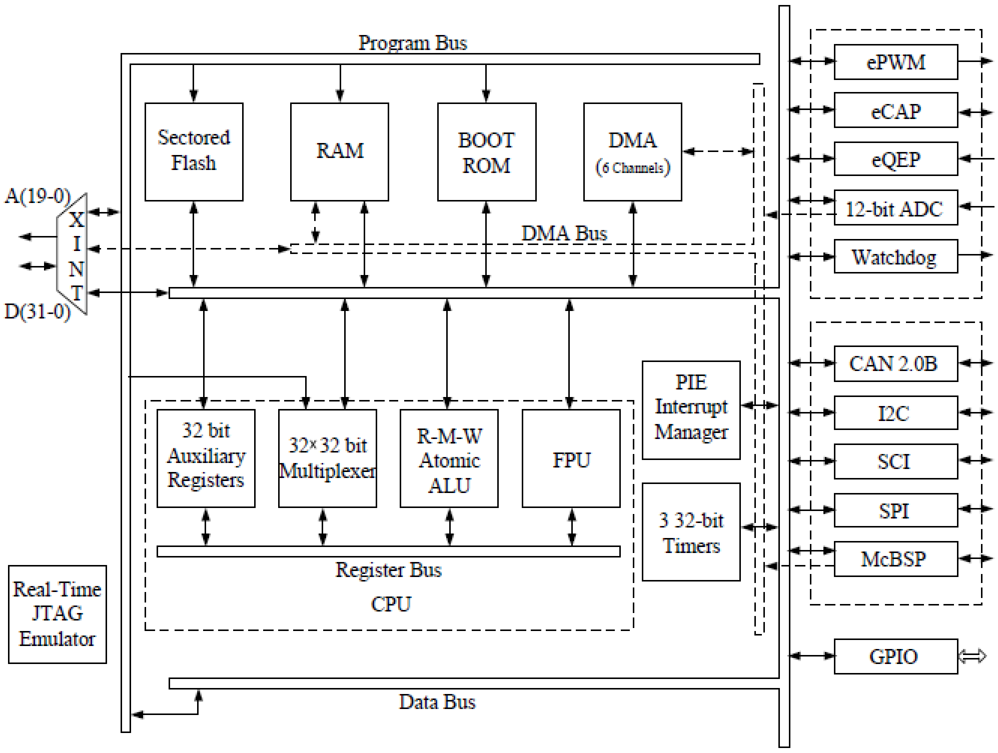


Figure 4.1: Block diagram of TMS320F28335.

For Software implementation *CodeComposerStudio* (CSS) which is Integrated Design Environment (IDE) from Texas Instrument, version 6.0.1, is used with several libraies from $ControlSUITE^{TM}$ for $C2000^{TM}$ is used.

Their are two 2 choice to software implementation, one is C language and other is Assembly language. In this project C Language is used to implement motor control. DSC is a 32-bit floating processor with max clock rate of 150MHz. DSC has both internal flash (512KB)and on-board RAM (68KB) for standalone control. There are also 88 General Purpose Input/Output (GPIO) pins where some of them have special features such as PWM and Encoder support, for example it has 18 PWM channels, 2 are Quadrature encoder and 6 Event captures. It also have 16 12-bit ADC channels with conversion time of $80ns$. It also have communication ports like USB, McBSP, I2C, UART/SCI, SPI, CAN.

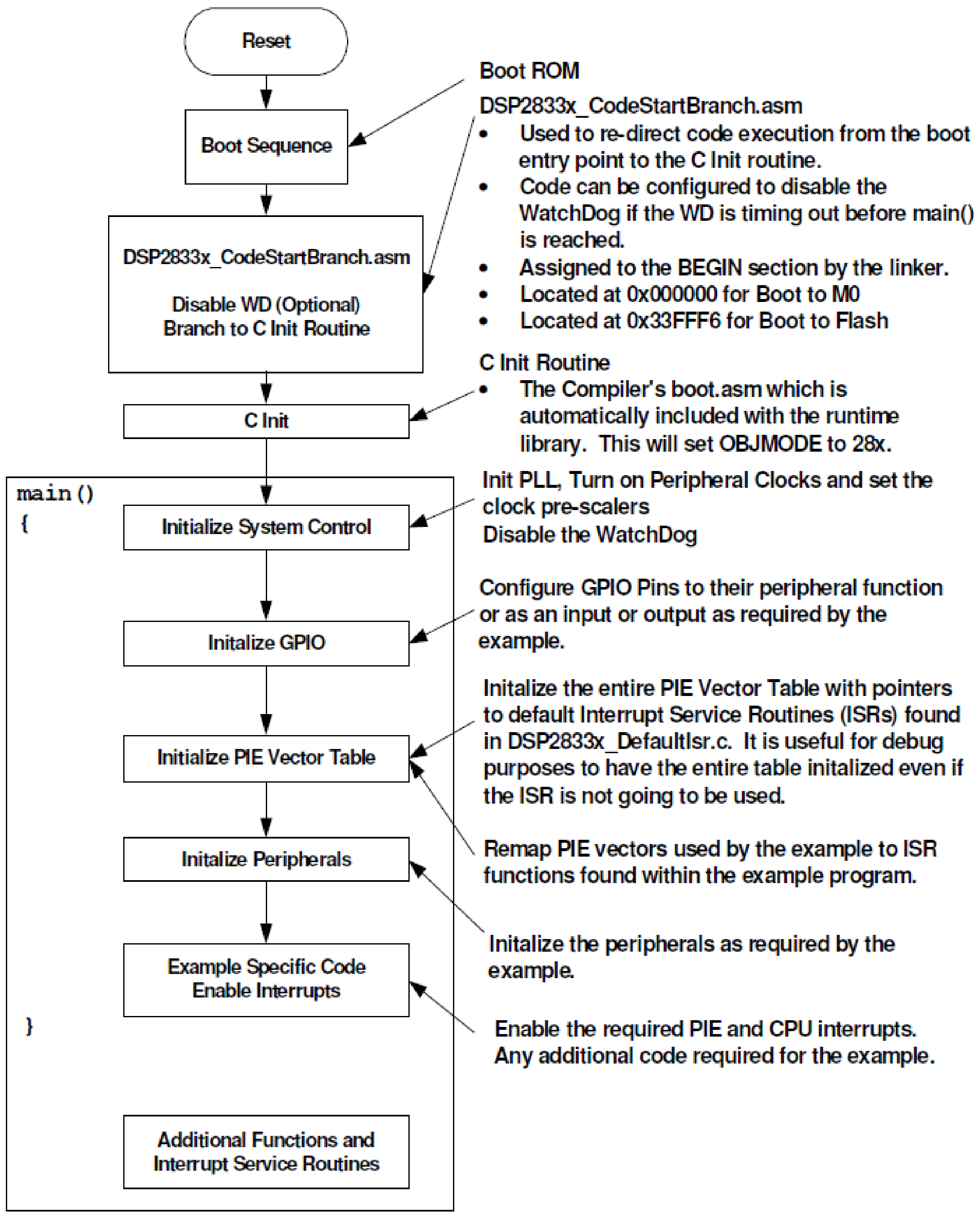
To make our work ease we will take advantage of some useful files, which have been created and provided by Texas Instruments so called as Header files, Source file and linker file. Few of them are listed below in table 4.1 with their function:

Table 4.1: Header files and Source code with their functions

Filename	*Function*
DSP2833x-Device.h	This header file is required to use the header files. This file includes all of the required peripheral specific header files and includes device specific macros and typedef statements.
DSP2833x-Adc.h	ADC register structure and bit-field definitions.
DSP2833x-Adc.c	ADC specific functions and macros.
DSP2833x-CpuTimers.h	CPU-Timer register structure and bit-field definitions.
DSP2833x-CpuTimers.c	CPU-Timer specific functions and macros.
DSP2833x-SysCtrl.h	System register definitions. Includes Watchdog, PLL, CSM, Flash/OTP, Clock registers.
DSP2833x-SysCtrl.c	System control (watchdog, clock, PLL etc) specific functions and macros.
DSP2833x-Gpio.h	General Purpose I/O (GPIO) register structures and bit-field definitions.
DSP2833x-Gpio.c	General-purpose IO (GPIO) specific functions and macros.
DSP2833x-GlobalVariableDefs.c	This file defines all global variable names to access memory mapped peripheral registers.
DSP2833x-DefaultIsr.h	Function prototype statements for the ISRs in DSP2833x-DefaultIsr.c.
DSP2833x-DefaultIsr.c	Shell interrupt service routines (ISRs) for the entire PIE vector table.
28335-RAM-lnk.cmd	28335/28235 memory linker command file. Includes all of the internal SARAM blocks on a 28335/28235 device
DSP2833x-Headers-nonBIOS.cmd	Assigns the peripheral register structure data sections to the proper memory location.

4.2 Basic Program Flow Chart

Figure 4.2 shows basic flow for setting up TMS320f28335,it include both hardware and software initialization.



Reset
Boot Sequence
DSP2833x_CodeStartBranch.asm
Disable WD (Optional)
Branch to C Init Routine
C Init
main ()
{
Initialize System Control
Initalize GPIO
Initialize PIE Vector Table
Initalize Peripherals
Example Specific Code
Enable Interrupts
}
Additional Functions and Interrupt Service Routines
Boot ROM
DSP2833x_CodeStartBranch.asm
• Used to re-direct code execution from the boot entry point to the C Init routine.
• Code can be configured to disable the WatchDog if the WD is timing out before main() is reached.
• Assigned to the BEGIN section by the linker.
• Located at 0x000000 for Boot to M0
• Located at 0x33FFF6 for Boot to Flash
C Init Routine
• The Compiler's boot.asm which is automatically included with the runtime library. This will set OBJMODE to 28x.
Init PLL, Turn on Peripheral Clocks and set the clock pre-scalers
Disable the WatchDog
Configure GPIO Pins to their peripheral function or as an input or output as required by the example.
Initalize the entire PIE Vector Table with pointers to default Interrupt Service Routines (ISRs) found in DSP2833x_DefaultIsr.c. It is useful for debug purposes to have the entire table initalized even if the ISR is not going to be used.
Remap PIE vectors used by the example to ISR functions found within the example program.
Initalize the peripherals as required by the example.
Enable the required PIE and CPU interrupts.
Any additional code required for the example.

Figure 4.2: Basic program flow chart for setting up TMS320f28335

In hardware initialization first processor reset itself then initialize various function like stack pointers, registers, clocks and watchdog. After hardware initialization software initialization start and setting are loaded to RAM, settings like PWM frequency, GPIO configuration, ADC channels, interrupt service routine etc comes under software initialization.

4.3 GPIO Initialization

F28335 has total 88 GPIO pin numbered from GPIO0 to GPIO87 and each single physical pin can be used for upto 4 different function by multiplexing technique. Their are several register group that we have to define to get required functioning of GPIO pin like register group GPxDIR defines the direction of pin, i.e Input or Output. Clearing a bit position to zero configures the line as an input, setting the bit position to 1 configures the line as an output. Registers GPxDAT is used to read data from input line where as register GPxSET, GPxCLEAR and GPxTOGGLE are used to set high, clear and toggle GPIO respectively. For our projects purpose, few of these GPIO pin are used functioning of them is shown in table 4.2 and figure 4.3 shows GPIO module of DSC.

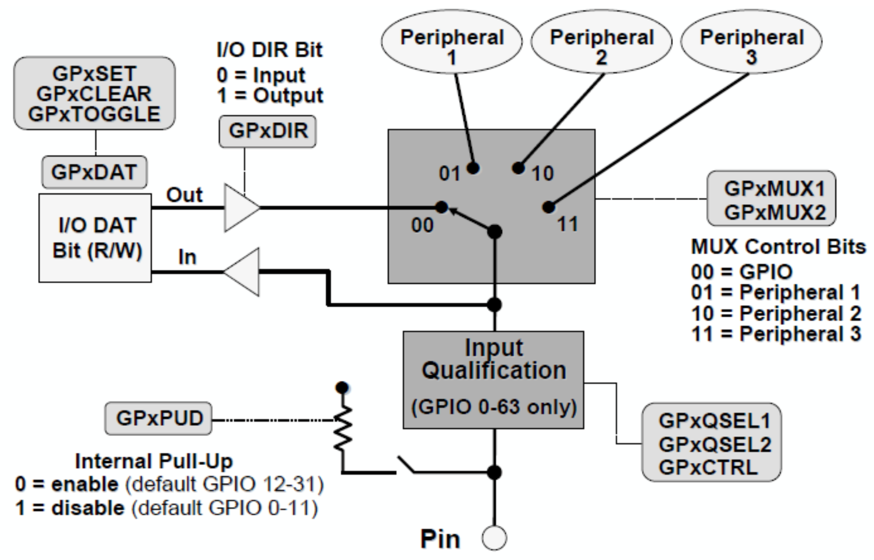


Figure 4.3: General purpose Input/Output module for TMS320f28335

Table 4.2: GPIO Pin with their description

Pin	$Input/output$	$Discription$
GPIO0	Output	ePWM-1A
GPIO1	Output	ePWM-1B
GPIO2	Output	ePWM-2A
GPIO3	Output	ePWM-2B
GPIO4	Output	ePWM-3A
GPIO5	Output	ePWM-3B
GPIO6	Input	HALL A
GPIO8	Input	HALL B
GPIO10	Input	HALL C
GPIO17	Input	Fault singal
GPIO24	Input	eCAPTURE

4.4 ePWM Initialization

F28335 has 6 six independent ePWM modules. In F28335 has two basic operating modes for ePWM system (1) standard ePWM 16-bit mode and (2) 24-bit High Resolution PWM mode (HRPWM). For this project standard ePWM bit mode is used. Each ePWM has two output channel named as EPWMxA and EPWMxB where x represent no from 1 to 6. Each ePWM module ,figure4.4, consists of 7 seven sub-module named as Time-base (TB) module, Counter-compare (CC) module, Action-qualifier (AQ) module, Dead-band (DB) module, PWM-chopper (PC) module, Event-trigger (ET) module and Trip-zone (TZ) module of all 7 seven sub-module is listed in table 4.3. Various registers like TBCTL, CMPA, TBPRD, DBCTL, AQCTLA etc are to be configured to get desired output. To set value for TBPRD following formula is to be used:

$$TBPRD = \frac{1}{2}(\frac{f_{SYSCLKOUT}}{f_{PWM} * CLKDIV * HSPCLKDIV}) \tag{4.1}$$

$TBPRD$ is period register a 16 bit register so it can handle a maximum value of $(2^{16} - 1)$ or 65535, factor of half is used only for "up/dowm" mode for "up" or "down" mode factor of half is to be removed. Where $f_{SYSCLKOUT}$ is system clock out frequency 150 MHz, f_{PWM} is required frequency of PWM signal, $CLKDIV$ is Clock division and $HSPCLKDIV$ is High speed clock division. For example to get PWM signal of 10 KHz following value is used, $CTRMODE = 2$, i.e "up/down" mode, $CLKDIV = 0$, i.e $f_{SYSCLKOUT}$ is divide by 1., $HSPCLKDIV = 1$, i.e $f_{SYSCLKOUT}$ is divide by 2, $f_{PWM} = 100KHz$ and $TBPRD = 375$ and for $f_{PWM} = 50KHz$ and $TBPRD = 750$.

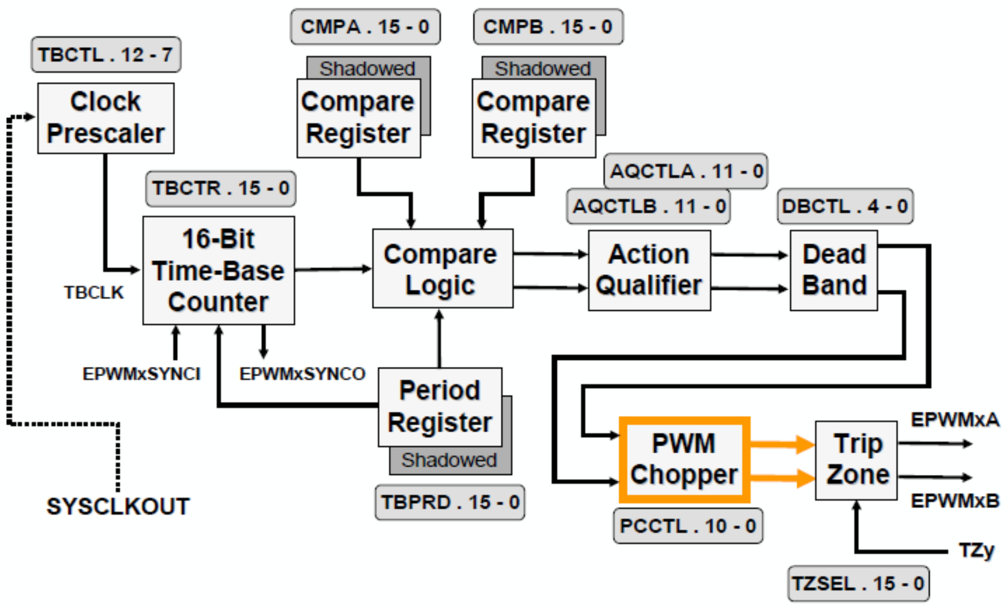


Figure 4.4: Flow diagram of ePWM module

Table 4.3: Configuration Parameter for different ePWM module

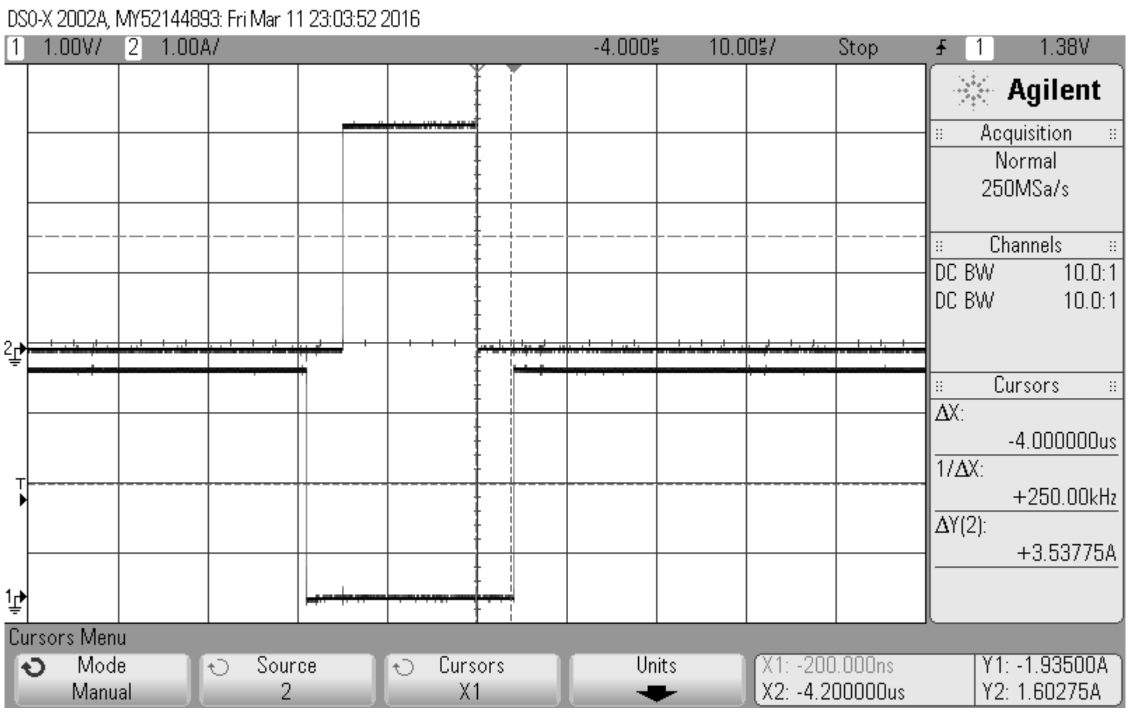
Module Name	*Configuration Parameters*
Time-base module	•Use to scale time-base clock (TBCLK) relative to the system clock (SYSCLKOUT). •PWM time-base counter (TBCTR) frequency or period. •Set the mode for the time-base counter, i.e count-up, count-down and count-up-down. • Set phase difference between two ePWM module.
Counter-compare module	•Set PWM duty cycle for output EPWMxA and/or output EPWMxB. • Specify the time at which switching events occur on the EPWMxA or EPWMxB output.
Action-qualifier module	•Specify the type of action taken when a time-base or counter-compare submodule event occurs, eg switch high, switch low, toggle or no action taken. •Force the PWM output state through software control.
Dead-band module	•Use to set rising edge delay or falling edge delay. • Use to make traditional complementary dead band relationship between upper and lower switches.
PWM-chopper module	• Use for high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band sub modules. • Set Programmable chopping (carrier) frequency .
Event-trigger module	• Specify the tripping action taken when a fault occurs, i.e Force EPWMxA and/or EPWMxB to high, low, ignore trip condition or to high impedance state. • Enable the trip-zone to initiate an interrupt.
Trip-zone module	•Enable the ePWM events that will trigger an interrupt. •Enable ePWM events that will trigger an ADC start-of-conversion event.

In Dead-band module ePWMxA is used as base signal to get Active High Complementary mode desired for pair of power switches in one phase. Value of DBRED and DBFED is adjusted to set rising edge delay and falling edge delay for ePWMxA and ePWMxB respectively. Value of DBRED/DBFED is calculate as follow:

$$DBRED = \frac{RequireRisingEdgeDelay}{TBCLK} \tag{4.2}$$

$$DBFED = \frac{RequireFallingEdgeDelay}{TBCLK} \tag{4.3}$$

for this project value of Dead-band is $4\mu s$ figure 4.5 shows actual dead-band of $4\mu s$ with Active High Complementary mode.


DSO-X 2002A, MY52144893: Fri Mar 11 23:03:52 2016

1 1.00V/ 2 1.00A/ -4.000µs 10.00µs/ Stop 1 1.38V

Acquisition: Normal 250MSa/s

Channels: DC BW 10.0:1, DC BW 10.0:1

Cursors: ΔX: -4.000000us; 1/ΔX: +250.00kHz; ΔY(2): +3.53775A

Mode Manual; Source 2; Cursors X1

X1: -200.000ns X2: -4.200000us Y1: -1.93500A Y2: 1.60275A

Figure 4.5: ePWM1A and ePWM1B in Active High Complementary mode with Dead-band of $4\mu s$.

ePWM trip zone module shown in figure 4.6 is used to sense over current fault signal generated by $Allegro^{TM}$ $ACS709$ current sensor.Their are 6 trip-zone pin available that are sourced from GPIO MUX, But for this project single pin is used. Actually we have 4 current sensors and fault signal from all these four is Digitally ANDed by AND gate IC $SN74AS21$ and output put of this AND gate is sent to DSC via trip-zone pin TZ6. In case of fault condition, which is ONE shot fault mode in this case, both eP-

WMxA and ePWMxB are forced to low state, where x= 1, 2 and 3 representing each phase of three phase inverter bridge.

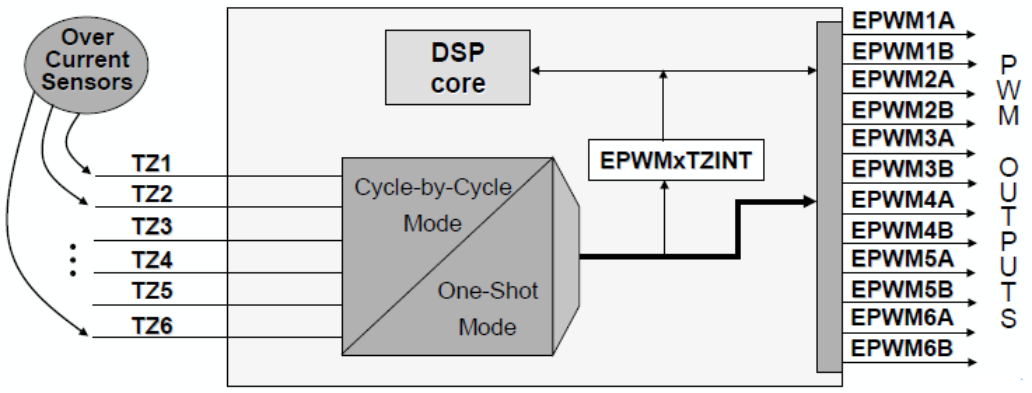


Figure 4.6: Trip-zone module

4.5 Interrupt and ADC Initialization

F28335 has a total of 16 Interrupt line out of which two are "Non-Maskable" and rest 14 are "Maskable", which mean these are programmable. Their are 96 possible source of interrupt which are grouped into 12 PIE-lines and 8 sources per line. To enable /disable individual source we have to program several group resister which comes under peripheral interrupt expansion(PIE). For this project we used 3 interrupts, 1^{st} is CPU timer interrupt used for hall sensor sensing and implementing commutation table, 2^{nd} is ADC interrupt used to sense potentio-meter reading which is used to adjust duty cycle of switching signal and 3^{rd} is eCAPTURE interrupt used to capture time stamp of hall sensor on and off state, which is further used to measure speed of motor.

Cpu timer interrupt occur every $100\mu s$ which mean 10000 execution per sec. Max speed of motor is 1500 RPM i.e is 25 rotation per sec and their are six combination of hall sensor in a single cycle which mean hall sensor change 150 times per cycle which is very less in comparison to Cpu-timer interrupt execution, other wise it will create problem in execution of commutation.

Analogue to digital converter is used to convert analogue vale to digital value, F28335 is equipped with 16 dedicated input pins to measure analogue voltage in range of 0 to 3 volt and convert into digital number which can be calculated as follow:

$$V_{in} = \frac{D * (V_{REF+} - V_{REF-})}{2^n - 1} + V_{REF-} \tag{4.4}$$

$$V_{in} = \frac{D * 3}{4095}$$

where V_{in} represent input analogue voltage, V_{REF+} and V_{REF-} represent reference voltage used to limit analogue voltage range i.e is $3.0V$ and $0V$ for F28335, D represent Digital number for given input analogue voltage and n represent resolution of ADC converter i.e 12. Analogue to Digital Converter (ADC) module, figure 4.7, in F28335 has resolution of 12 bits with maximum conversion rate of $80ns$. ADC can be initialize either by external trigger, software or by ePWM module. ADC can be configured in several operating mode which are combination of three basic mode named as sequencer mode, sampling mode and start mode. Sequencer mode is divided into two types cascaded sequencer mode (16 states) and dual sequencer mode (8 states) similarly sampling mode is divided into two ways sequential sampling (1 channel at a time)and simultaneous sampling (2 channels at a time) and start mode describe about type of sequence of mode used i.e single sequence mode (stop at end of sequence)or continuous sequence mode (wrap at end of sequence). For this project ADC is configured in cascade sequencer mode with sequential sampling in continuous mode of operation with $ePWM2A\_SOC$ as triggering mechanism.

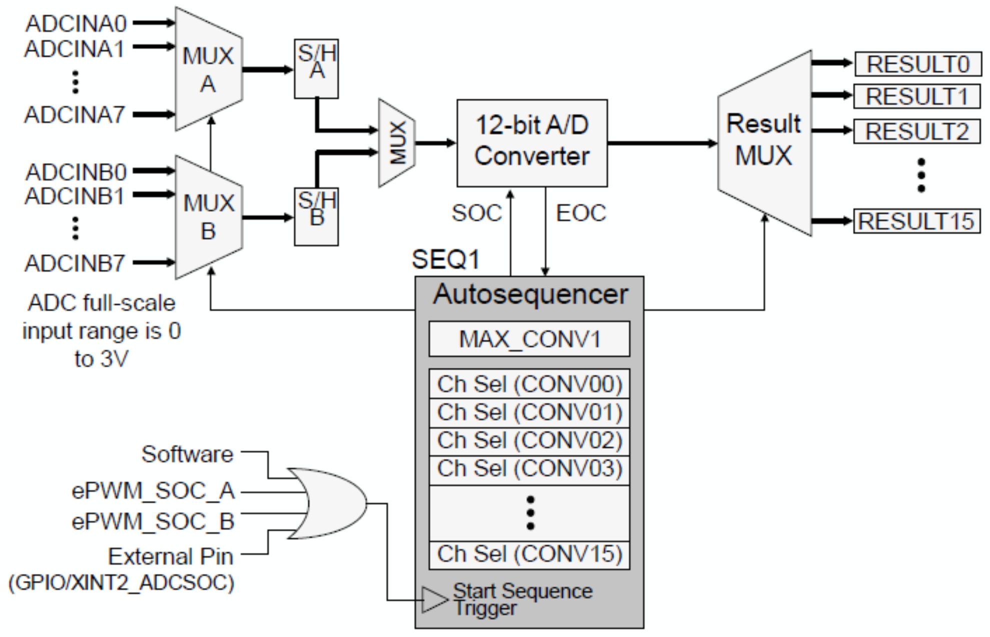


Figure 4.7: ADC module in cascade sequencer mode.

4.6 Experimental Results With Resistive Load

In this section we will discuss about operation of three phase voltage source inverter with resistive load, typically in star connection. three phase voltage source inverter can be operated into two mode 1) $180°$ conduction mode and 2) $120°$ conduction mode. Main difference between these operation mode is of operating time in $180°$ conduction each MOSFET conduct for $180°$ where as in $120°$ conduction mode each MOSFET conducts for $120°$. In $180°$ conduction mode three devices are active at a time where as in $120°$ conduction mode only two are active at time. Gating pulses for $180°$ conduction mode is given in following order 123, 234, 345, 456, 561 and 612 where as for $120°$ conduction mode it is 12, 23, 34, 45, 56 and 61 for converter whose switches whose switches are numbered as shown in figure 4.8:

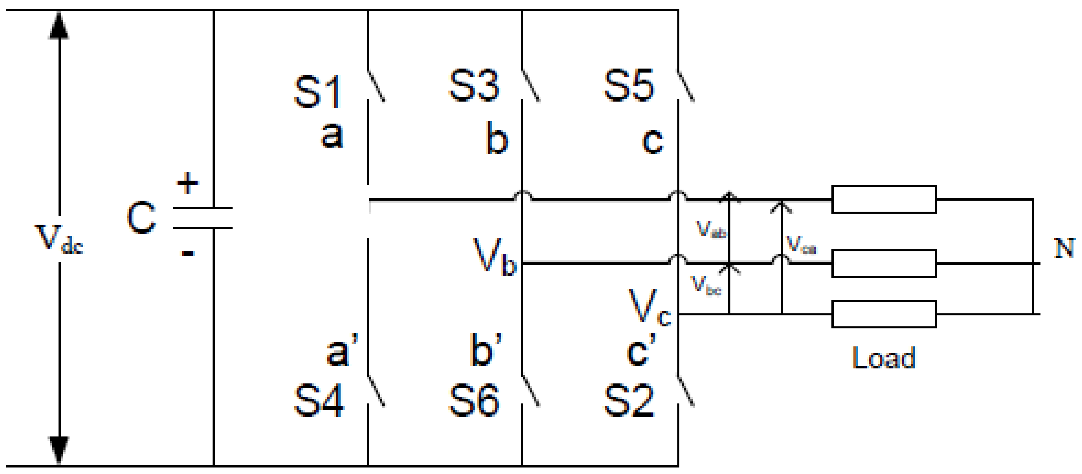


Figure 4.8: Three phase inverter.

Inverter board is powered with DC power supply from $TDK-Lambda's\ GEN80-42$ model. DSC is powered by laptop's USB port. Driver Board require three different type of voltage supply 3.3V with respect to DSC ground, 5V and 15V with respect to inverter ground all these three power supply is taken from another DC source from $Aplab$ with proper grounding connection. Voltage is measured by differential probe from $AgilentTechnologies$ and current is measured by $Tektronics's$ current probe amplifier TCP300 and Digital storage oscilloscope (DSO) from $AgilentTechnologies$ is used to view waveforms.

Figure 4.9 shows experimental waveform of line voltage V_{ab} and phase V_{an} obtain when three phase inverter is operated in 180° conduction mode with switching frequency of $50KHz$, and figure 4.10 shows waveform with switching frequency of $100KHz$.

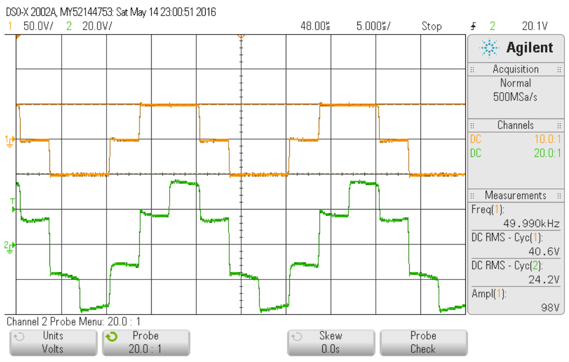


Figure 4.9: Experimental line voltage V_{ab} and phase V_{an} in 180° conduction mode with switching frequency of $50KHz$.

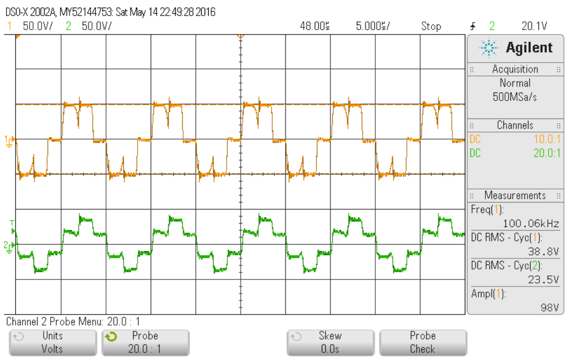


Figure 4.10: Experimental line voltage V_{ab} and phase V_{an} in 180° conduction mode with switching frequency of $100KHz$.

Where as figure 4.11 shows ideal wave form of line voltage V_{ab} and phase V_{an} obtain when three phase inverter is operated in 180° conduction mode.

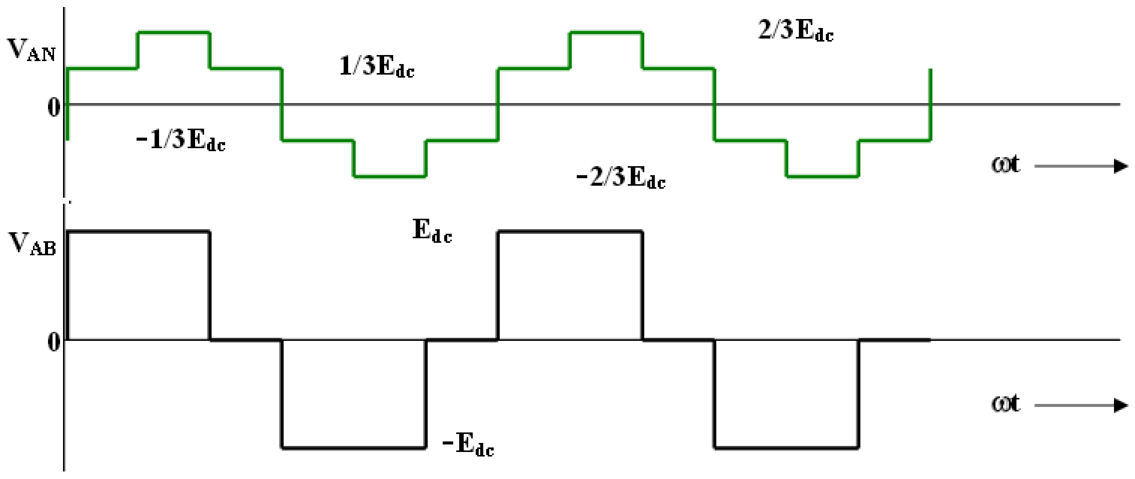


Figure 4.11: Ideal line voltage V_{ab} and phase V_{an} in 180° conduction mode.

Figure 4.12 show experimental waveform line voltage V_{ab} and line current i_a in 180° conduction mode with switching frequency of $50KHz$ and figure 4.13 shows waveform with switching frequency of $100KHz$.

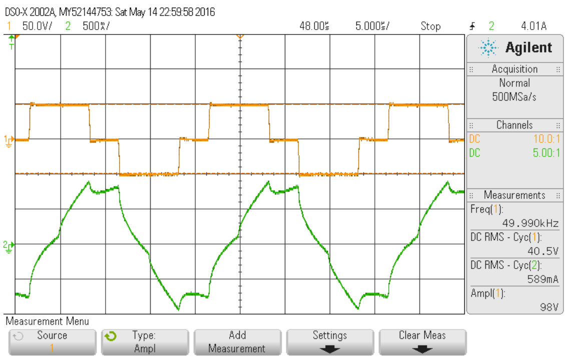


Figure 4.12: experimental waveform line voltage V_{ab} and line current i_a in 180° conduction mode with switching frequency of $50KHz$

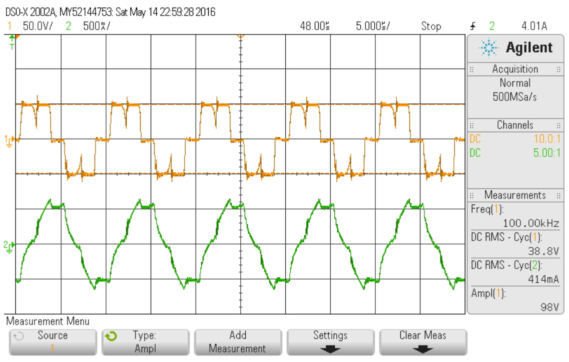


Figure 4.13: Experimental waveform line voltage V_{ab} and line current i_a in 180° conduction mode with switching frequency of $100KHz$

Where as figure 4.14 shows ideal wave form of line voltage V_{ab} and line current i_a obtain when three phase inverter is operated in 180° conduction mode.

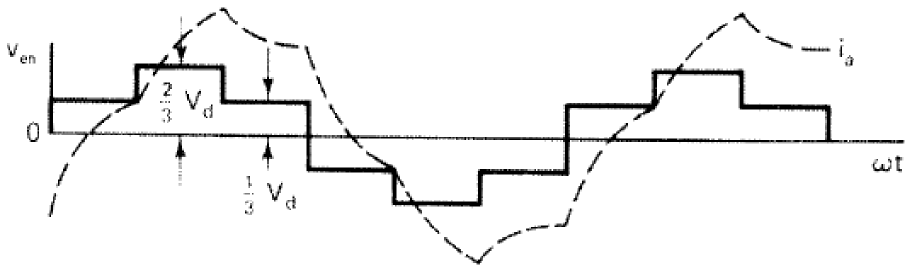


Figure 4.14: Ideal line voltage V_{ab} and line current i_a in 180° conduction mode.

Figure 4.15 and 4.16shows experimental line voltage V_{ab} and V_{bc} in 180° conduction mode with switching frequency of $50KHz$ and $50KHz$ respectively, with 120° phase shift.

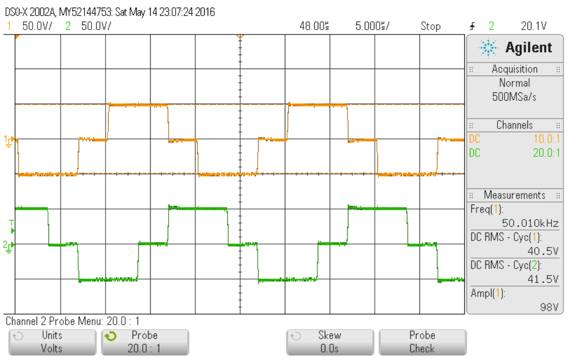


Figure 4.15: Experimental waveform line voltage V_{ab} and V_{bc} in 180° conduction mode with switching frequency of $50KHz$.

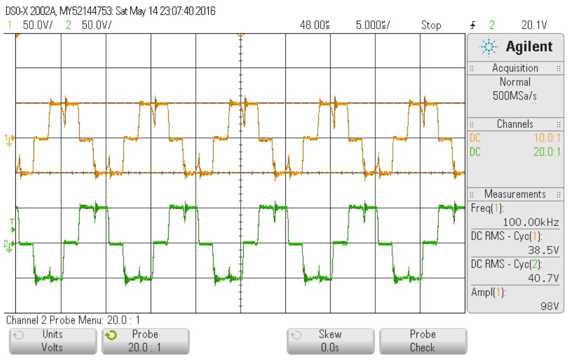


Figure 4.16: Experimental waveform line voltage V_{ab} and V_{bc} in 180° conduction mode with switching frequency of $100KHz$.

To get variable AC voltage and variable frequency pules width modulation (PWM) is used. There are different kind of PWM techniques, but Sine Pulse Width Modulation (SPWM) is mostly preferred. With proper switching it is possible to control the magnitude of lower order harmonics. 3 phase inverter is programmed to get sinusoidal current wave for with $50Hz$ frequency. Figure 4.17 experimental waveform for line voltage V_{ab} and phase current i_a and figure 4.18 show experimental waveform for Line voltage V_{ab} and phase voltage V_{an}

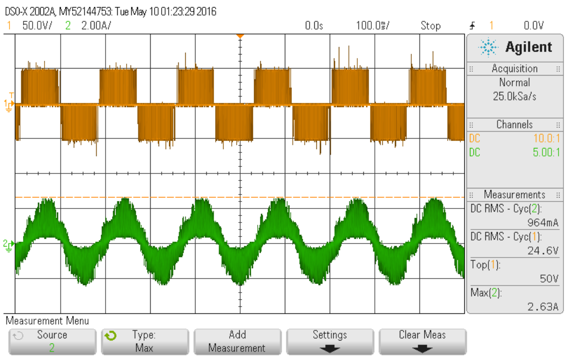


Figure 4.17: Line voltage V_{ab} and phase current i_a with SPWM technique.

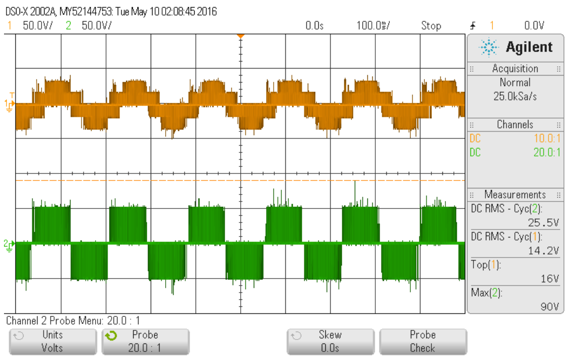


Figure 4.18: Line voltage voltage V_{ab} and Phase voltage V_{an}

4.7 Experimental Results For Motor Load

In this section we will discuss about results obtain by MATLAB simulation with actual hardware result. For three phase sensored BLDC motor used with rating of 24 $Volts$, 80 $Watts$ and 1500 RPM.

Figure 4.19 show 120 degree flat region of back EMf-A and back EMF-B when motor is driven by mechanical input, time period of wave form is of $155ms$ where as top flat region contribute $55ms$ which is 2/3 of period and both wave form has a phase difference of 120 which is same as simulation result which shown in figure 4.20.

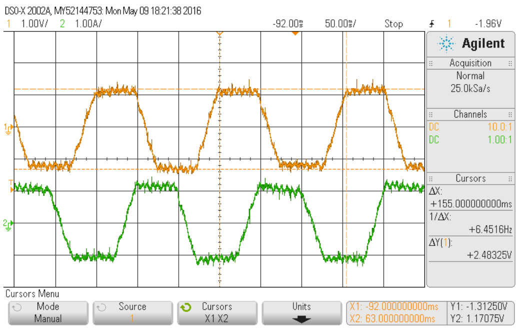


Figure 4.19: Trapezoidal Back EMF for A phase and B phase, hardware results.

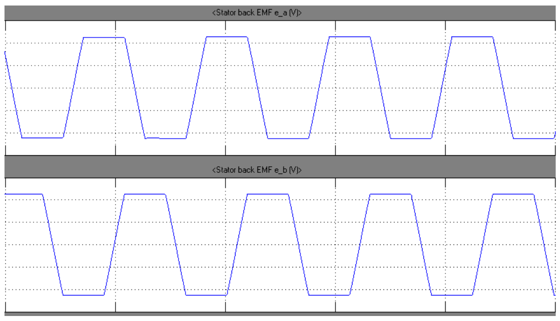


Figure 4.20: Trapezoidal Back EMF for A phase and B phase, simulation result

Figure 4.21 show Back EMF-A versus hall-A which are is same as simulation result shown in figure 4.22. Hall sensor is used to sense position of rotor and appropriate phase is excited to get constant power output from motor. Hall sensor give high signal when top flat region of start and remain hign till starting point of lower flat region.

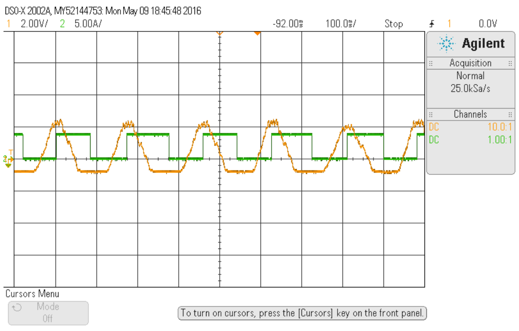


Figure 4.21: Back EMF for A phase v/s HAll sensor A, hardware results.

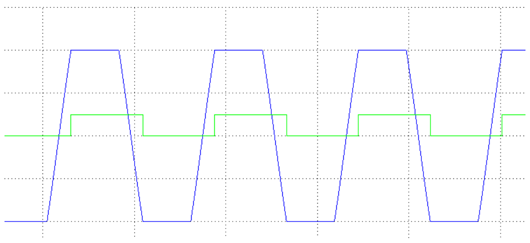


Figure 4.22: Back EMF for A phase v/s HAll sensor A, simulation result

Figure 4.23 shows Hall-A and HAll-B when motor is running and figure 4.24 shows simulation result. Hall-A and Hall-B are $120°$ phase shifted with each hall senor high for 180° and low for another 180°

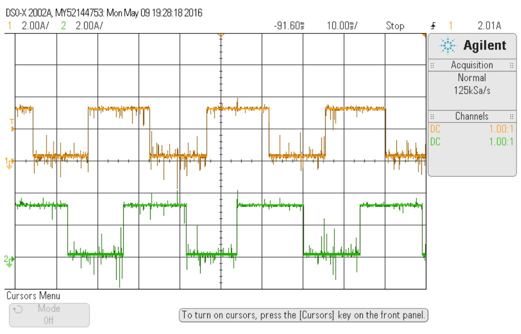


Figure 4.23: Hall-A sensor versus Hall-B sensor, hardware results.

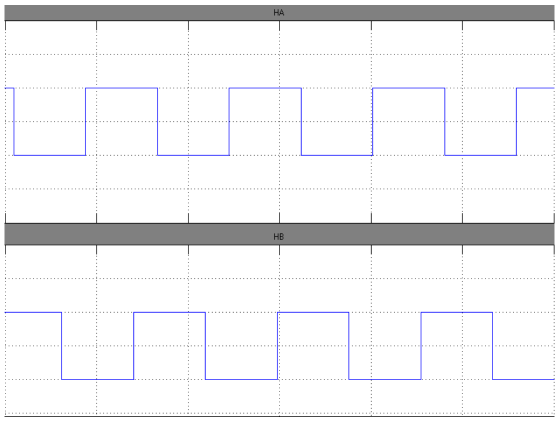


Figure 4.24: Hall-A sensor versus Hall-B sensor, simulation result

Figure 4.25 shows switching of upper MOSFET of phase 1,2 and 3. It is observer that at a time only 2 phases are activated and lower MOSFET of these legs are in Active High Complementay mode as show in figure 4.5.

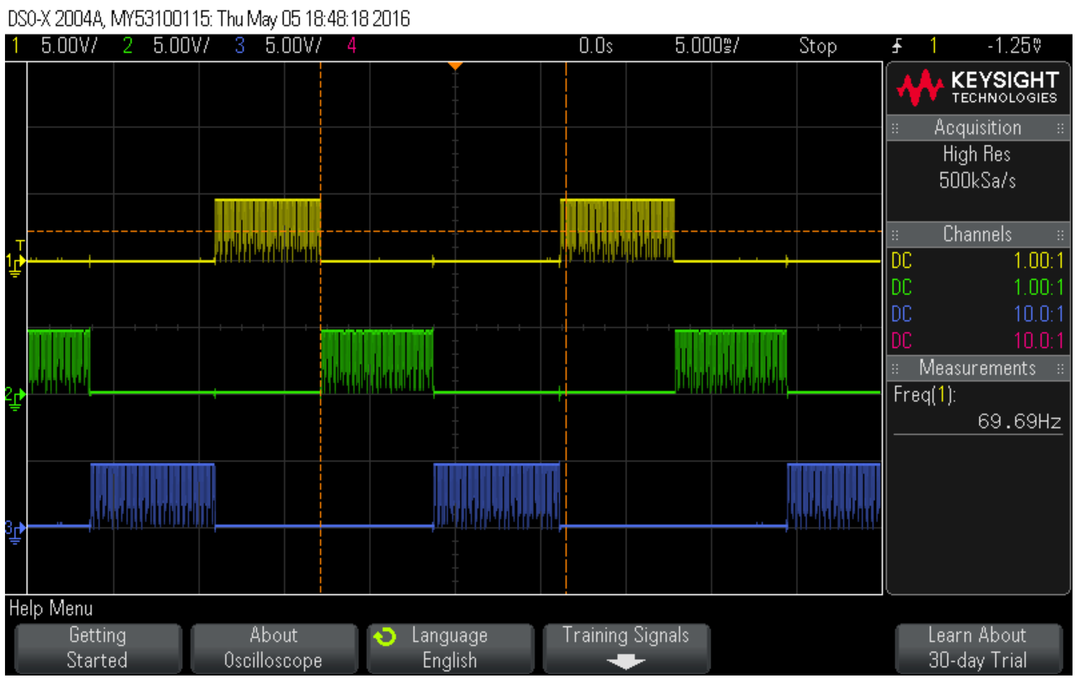


Figure 4.25: Switching of upper MOSFET for phase 1, 2 and 3.

Figure 4.26 shows relation between line voltage V_{ab} and V_{bc} which are 120° phase shifted with respect to each other and figure 4.27 shows hall-A sensor signal with line voltage V_{ab} .

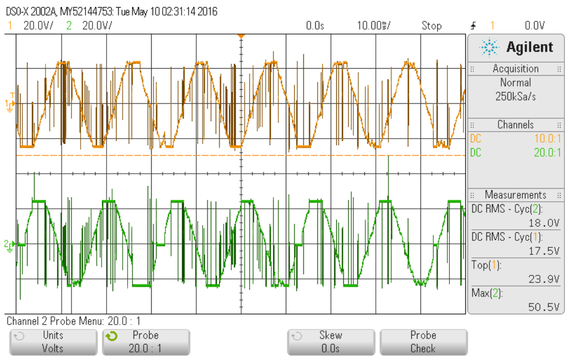


Figure 4.26: Line Voltage V_{ab} versus line voltage V_{bc}

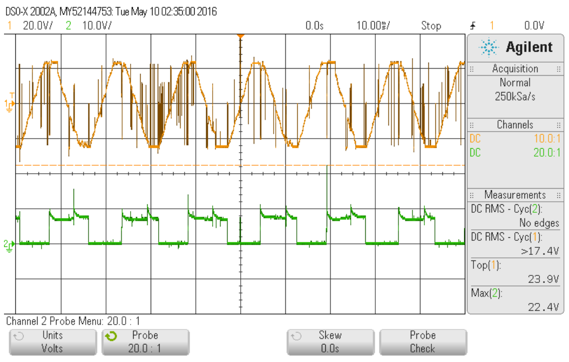


Figure 4.27: Hall-a signal versus line voltage V_{ab}

Figure 4.28 shows relationship between phase current for A-Phase and line voltage V_{ab}.

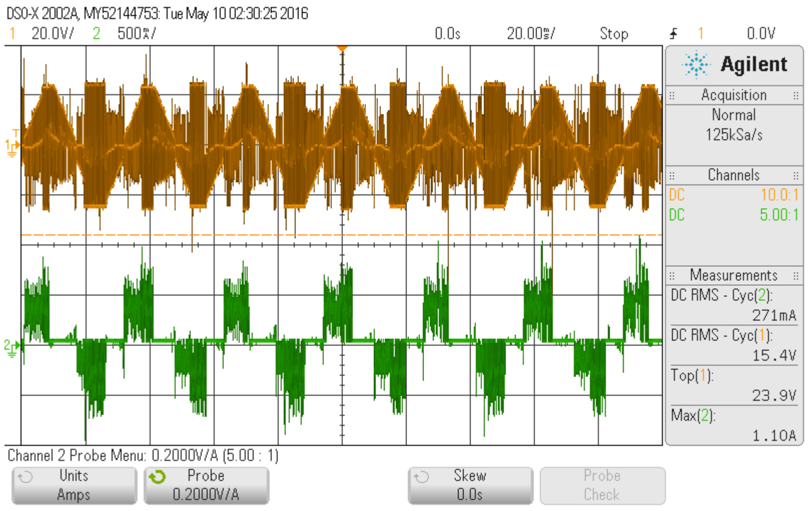


Figure 4.28: Phase A current versus line voltage V_{ab}.

CHAPTER 5

CONCLUSION

The thesis purpose was to design and implementation of three phase inverter for BLDC motor with sensored operation. A three phase inverter has been simulated, design, built and evaluated. Texas Instrument's TMS320f28335 DSC is used to implement algorithm for sensored operating mode, external potentiometer (which act as throttle) is used to change RPM of motor. Analogue voltage from potentiometer is sensed by ADC and according to that duty cyle of PWM is changed which in turn input voltage to BLDC motor.

Star connected resistive load is used to load the inverter in $180°$ conduction mode with switching frequency of $50KHz$ and $100KHz$and experimental result is compared with ideal results. SPWM technique with frequency of $50Hz$ is also used to get sinusoidal output current and modulation index is varied by same potentiometer which is used to control speed of motor.

APPENDIX A

Driver Board Layout And Schematics

Figure A.1 board layout where as figure A.2 and figure A.3 shows schematic diagram for Driver board.

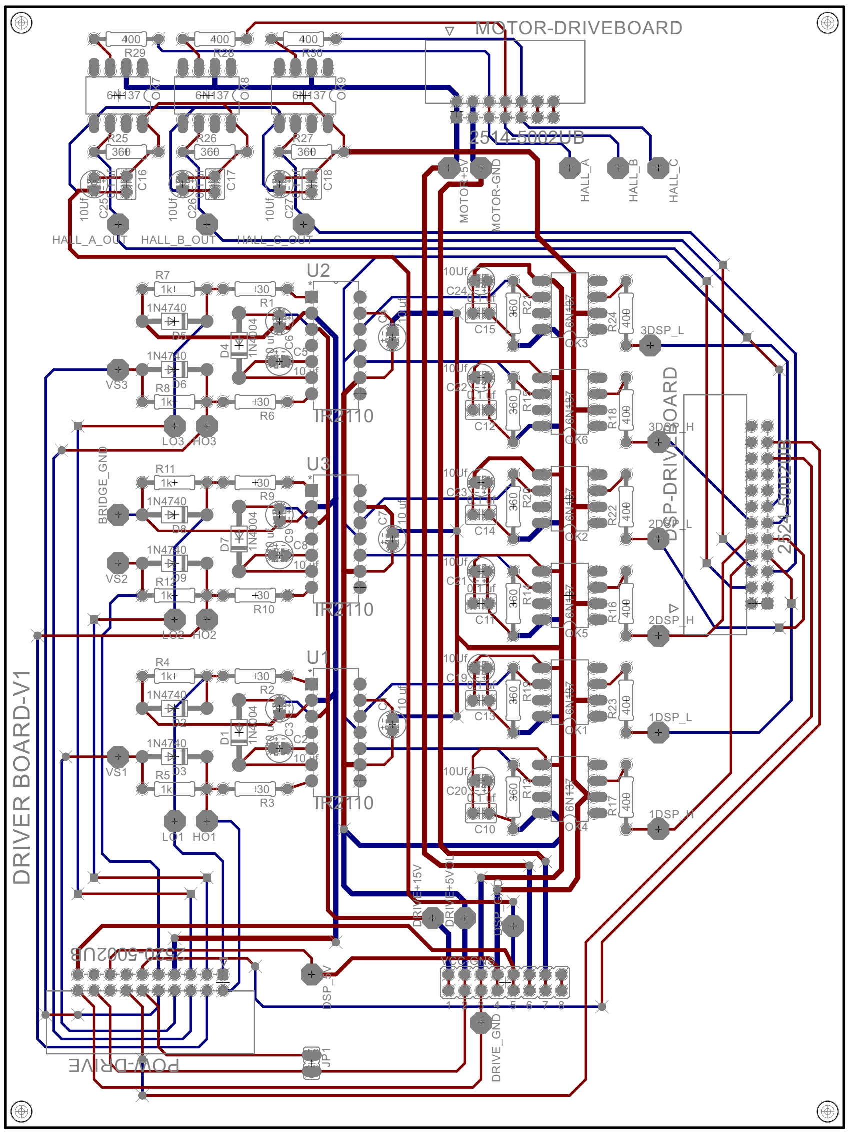


Figure A.1: Board layout for driver board in EAGLE Cad.

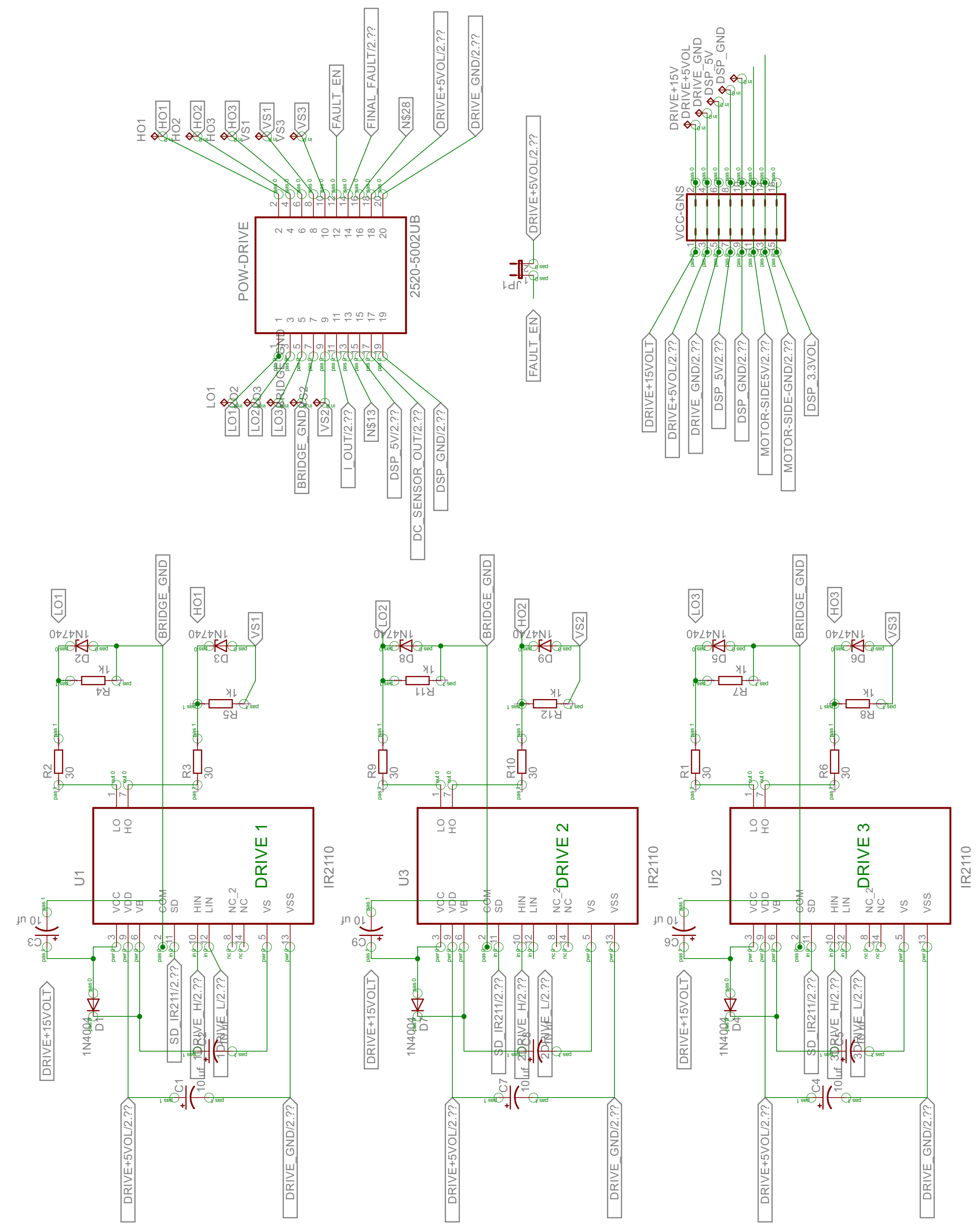


Figure A.2: Schematics for driver board in EAGLE Cad part1.

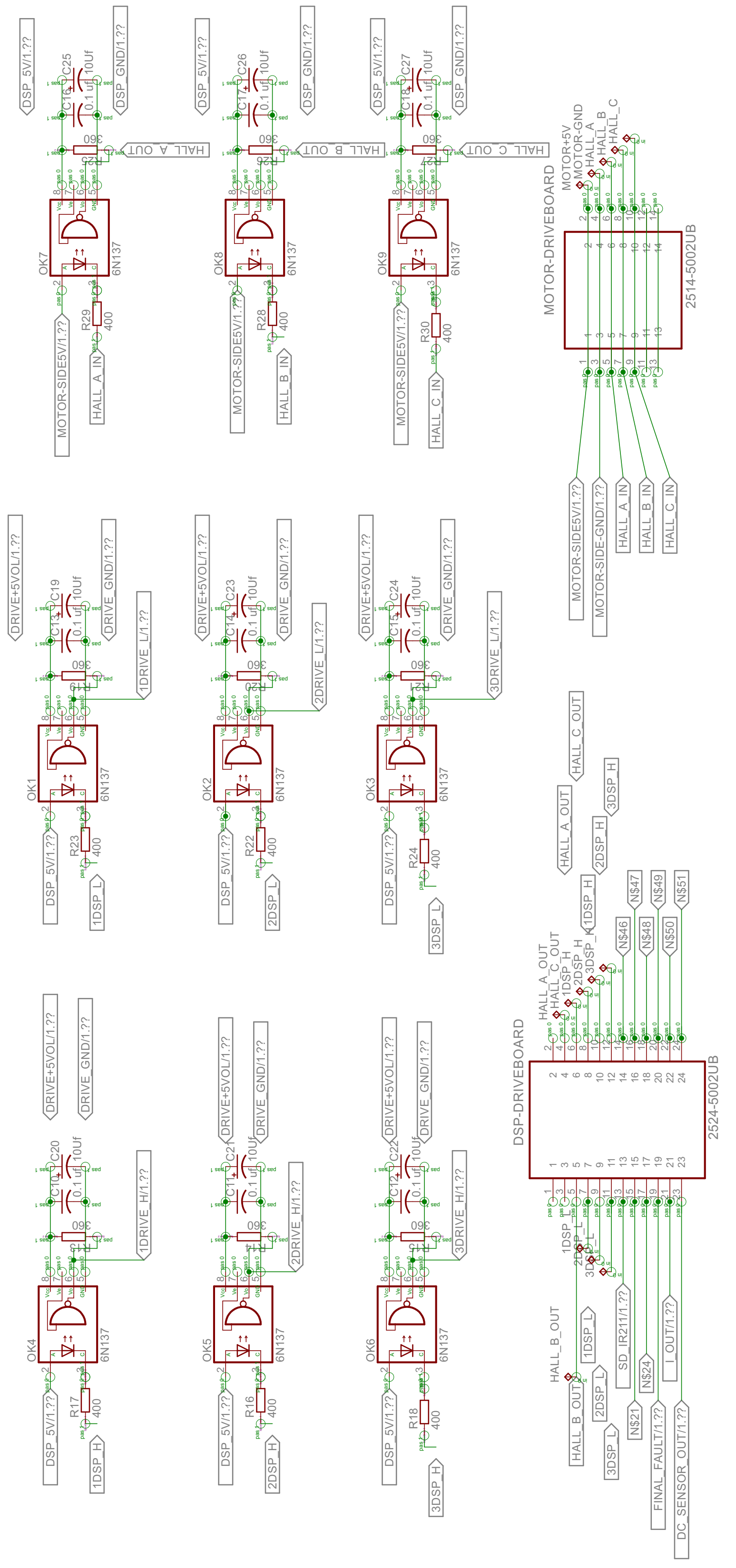


Figure A.3: Schematics for driver board in EAGLE Cad part2.

APPENDIX B

Power Board Layout And Schematics

Figure B.1 shows board layout and figure B.2 shows schematic diagram for Power board.

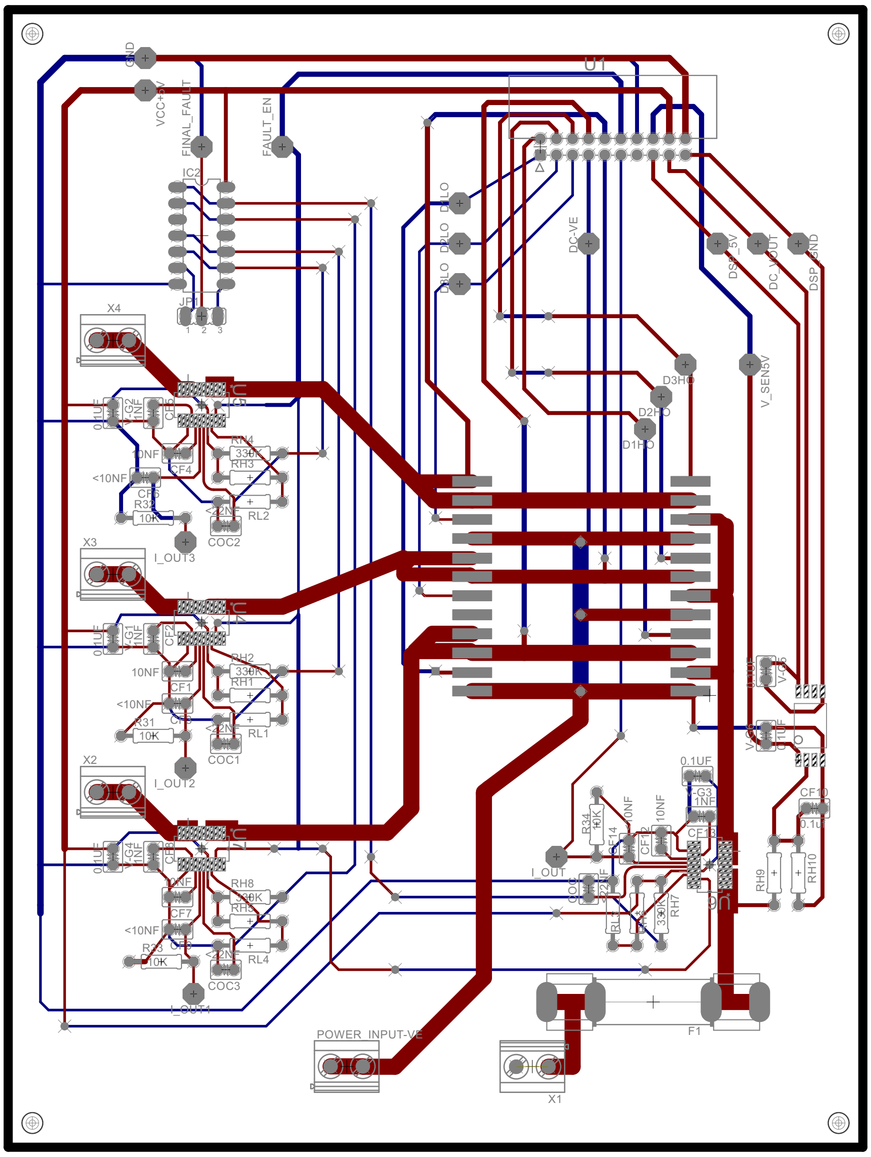


Figure B.1: Board layout for power board in EAGLE Cad.

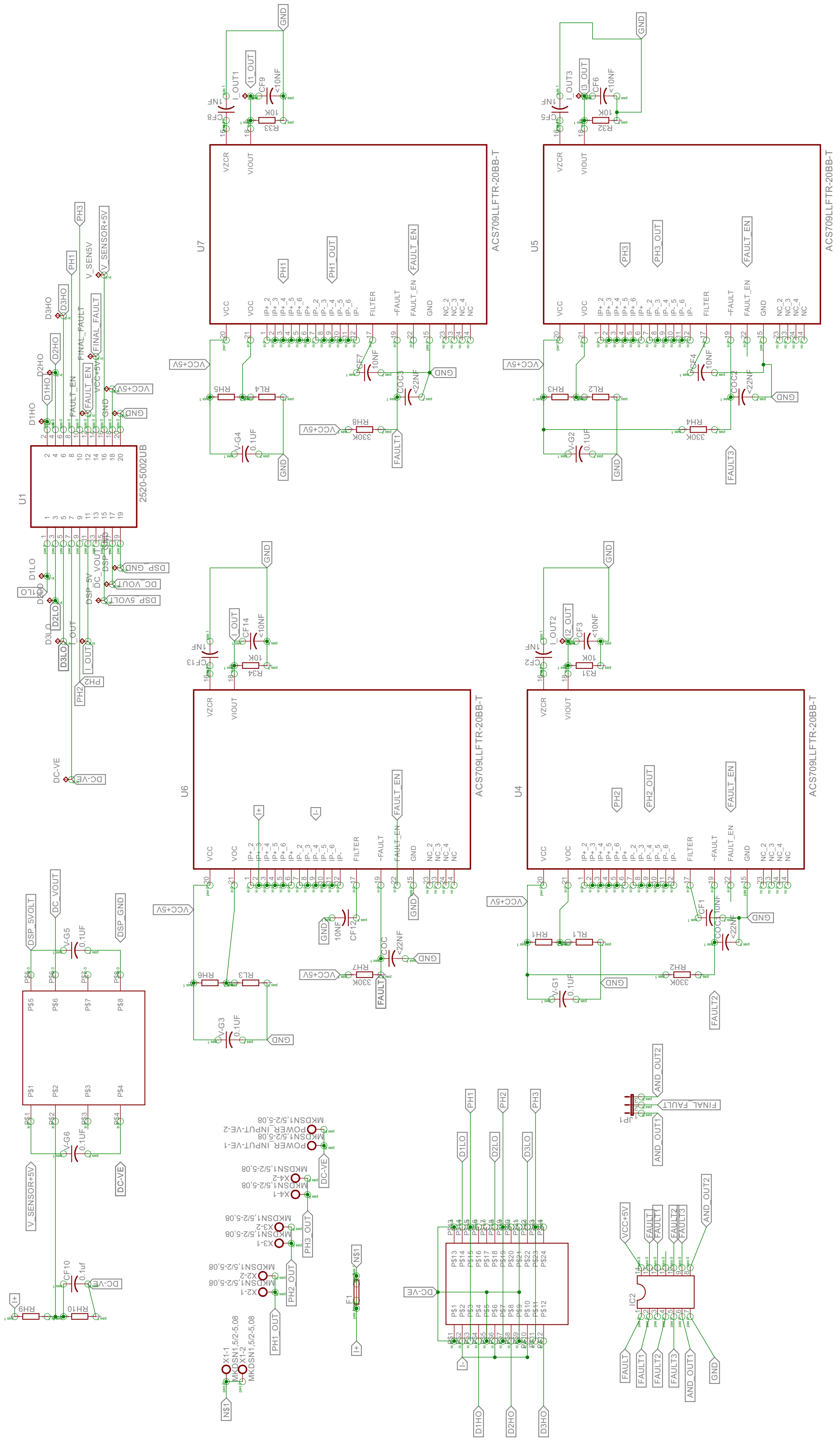


Figure B.2: Schematics for power board in EAGLE Cad.

APPENDIX C

C Code For Commutation Sequence For BLDC Motor

```c
#include "DSP2833x_Device.h"
#include "math.h"
// external function prototypes
extern void InitSysCtrl(void);
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);
extern void InitCpuTimers(void);
extern void ConfigCpuTimer(struct CPUTIMER_VARS *, float, float);
extern void InitAdc(void);
// Prototype statements for functions found within this file.
void Setup_adc(void);
void call_dog1(void);
void call_dog2(void);
void Gpio_select(void);
void Setup_ePWM(void);
void Setup_eCAP(void);
interrupt void cpu_timer0_isr(void);
interrupt void adc_isr(void); // ADC  End of Sequence ISR
interrupt void eCAP1_isr(void);
//---------------------------------------------------------------------------
#define deadband 300;// dead band of 4usec for both rise and fall
unsigned int duty;
int speed;
int Omega=0;
int Ha=0; //hall A 4Apwm
int Hb=0; //hall B 5Apwm
int Hc=0; //hall C 6Apwm
int value=0; //for switch case
//---------Main-Code--------
void main(void)
{
  int counter=0;  // binary countesr for digital output
  InitSysCtrl();  // Basic Core Init from DSP2833x_SysCtrl.c
  EALLOW;
    SysCtrlRegs.WDCR= 0x00AF; // Re-enable the watchdog
    EDIS;      // 0x00AF  to NOT disable the Watchdog, Prescaler = 64
  DINT;        // Disable all interrupts
  Gpio_select();    // GPIO9, GPIO11, GPIO34 and GPIO49 as output
  Setup_ePWM();   // init of ePWM 1,2 and 3
  Setup_eCAP();   // init  eCAP1
  InitPieCtrl();    // basic setup of PIE table; from DSP2833x_PieCtrl.c
  InitPieVectTable(); // default ISR's in PIE
  InitAdc();      // ADC insitalize
```

```
Setup_adc();      // Adc setup
EALLOW;
PieVectTable.TINT0 = &cpu_timer0_isr;
PieVectTable.ADCINT = &adc_isr;
PieVectTable.ECAP1_INT= &eCAP1_isr;
EDIS;
InitCpuTimers();  // basic setup CPU Timer0, 1 and 2
ConfigCpuTimer(&CpuTimer0,150,100);
PieCtrlRegs.PIEIER1.bit.INTx7 = 1;  // cpu timer in PIE group 1
PieCtrlRegs.PIEIER1.bit.INTx6 = 1;  // ADC in PIE group 1
PieCtrlRegs.PIEIER4.bit.INTx1 = 1;  // Enable ECAP1_INT in PIE group 4
IER |=9;  // Enable INT4 and INT1
EINT;
ERTM;
CpuTimer0Regs.TCR.bit.TSS = 0;  // start timer0
while(1)
{
      while(CpuTimer0.InterruptCount == 0);
    CpuTimer0.InterruptCount = 0;
    EALLOW;
    SysCtrlRegs.WDKEY = 0x55; // service WD #1
    EDIS;
    counter = counter + 1;
    if(counter&1) GpioDataRegs.GPASET.bit.GPIO9 = 1;
      else GpioDataRegs.GPACLEAR.bit.GPIO9 = 1;
    if(counter&2) GpioDataRegs.GPASET.bit.GPIO11 = 1;
      else GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
    if(counter&4) GpioDataRegs.GPBSET.bit.GPIO34 = 1;
      else GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;
    if(counter&8) GpioDataRegs.GPBSET.bit.GPIO49 = 1;
      else GpioDataRegs.GPBCLEAR.bit.GPIO49 = 1;
}
}
void call_dog1(void)
{
EALLOW;
SysCtrlRegs.WDKEY = 0x55; // service WD #1
EDIS;
}
void call_dog2(void)
{
EALLOW;
SysCtrlRegs.WDKEY = 0xAA; // service WD #1
EDIS;
}
void Gpio_select(void)
{
EALLOW;
GpioCtrlRegs.GPAMUX1.all = 0;   // GPIO15 ... GPIO0 = General Puropse I/O
//------------------------------EPWM activation start-----------------------------
GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1; // ePWM1A active
```

```
  GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1; // ePWM1B active
  GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1; // ePWM2A active
  GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 1; // ePWM2B active
  GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1; // ePWM3A active
  GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 1; // ePWM3B active
//------------------------------EPWM ACTIVATION END-------------------------------
  GpioCtrlRegs.GPAMUX2.all = 0;    // GPIO31 ... GPIO16 = General Purpose I/O
//------------------------------tripzone------------------------------------------
  GpioCtrlRegs.GPAMUX2.bit.GPIO17 = 3; // GPIO17 as /TZ6 trip zone
//----------------------------tripzone end-----------------------------------------
//------------------------------HAll ACTIVATION START-----------------------------
  GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 0;    // hall A EPWM4A
  GpioCtrlRegs.GPAMUX1.bit.GPIO8 = 0;    // hall B EPWM5A
  GpioCtrlRegs.GPAMUX1.bit.GPIO10 = 0;   // hall C EPWM6A
  GpioCtrlRegs.GPAMUX2.bit.GPIO24= 1; // eCAP1 active using to measure speed of motor
//-------------------------------HALL ACTIVATION END------------------------------
  GpioCtrlRegs.GPBMUX1.all = 0;    // GPIO47 ... GPIO32 = General Purpose I/O
  GpioCtrlRegs.GPBMUX2.all = 0;    // GPIO63 ... GPIO48 = General Purpose I/O
  GpioCtrlRegs.GPCMUX1.all = 0;    // GPIO79 ... GPIO64 = General Purpose I/O
  GpioCtrlRegs.GPCMUX2.all = 0;    // GPIO87 ... GPIO80 = General Purpose I/O
//-------------------------------Led activation start-----------------------------
  GpioCtrlRegs.GPADIR.all = 0;
  GpioCtrlRegs.GPADIR.bit.GPIO9 = 1;  // peripheral explorer: LED LD1 at GPIO9
  GpioCtrlRegs.GPADIR.bit.GPIO11 = 1; // peripheral explorer: LED LD2 at GPIO11
  GpioCtrlRegs.GPADIR.bit.GPIO6 = 0;     // hall A EPWM4A INPUT MODE
  GpioCtrlRegs.GPADIR.bit.GPIO8 = 0;     // hall B EPWM5A INPUT MODE
  GpioCtrlRegs.GPADIR.bit.GPIO10 = 0;    // hall C EPWM6A INPUT MODE
  GpioCtrlRegs.GPBDIR.all = 0;     // GPIO63-32 as inputs
  GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1; // peripheral explorer: LED LD3 at GPIO34
  GpioCtrlRegs.GPBDIR.bit.GPIO49 = 1; // peripheral explorer: LED LD4 at GPIO49
  GpioCtrlRegs.GPCDIR.all = 0;     // GPIO87-64 as inputs
//-------------------------------Led activation end------------------------------
  EDIS;
}
void Setup_ePWM(void)
{
  //notes:
  // 1A and 1B for 1st leg up and dowm respectively.
  // 2A and 2B for 2nd leg up and dowm respectively.
  // 3A and 3B for 3rd leg up and dowm respectively.
//----------------------------PWM1 SETTING START---------------------------------
  EPwm1Regs.TBCTL.bit.CLKDIV =  0;  // CLKDIV = 1
  EPwm1Regs.TBCTL.bit.HSPCLKDIV = 1;  // HSPCLKDIV = 2
  EPwm1Regs.TBCTL.bit.CTRMODE = 2;  // up - down mode
  EPwm1Regs.AQCTLA.all = 0x0090;    // set ePWM1A on CMPA up AND clear ePWM1A on CMPA
      down
  EPwm1Regs.AQCTLB.all = 0;
  EPwm1Regs.TBPRD = 3750;        // 10KHz - PWM signal
  EPwm1Regs.CMPA.half.CMPA  = 375;//(Uint16)SIN;    //duty cycle first
  EPwm1Regs.DBCTL.all= 11;       // use to get 1A and its complementry 1B (AHC mode)
  EPwm1Regs.DBFED= deadband;         // DEAD BAND FOR FALLING EDGE 10uSEC= TBCLK*DBFED
```

```
EPwm1Regs.DBRED= deadband;         // DEAD BAND FOR RISING EDGE 10uSEC= TBCLK*DBFED
//-----------------------------PWM1 SETTING END--------------------------------------
//-----------------------------PWM2 SETTING START------------------------------------
EPwm2Regs.TBCTL.bit.CLKDIV =  0;  // CLKDIV = 1
EPwm2Regs.TBCTL.bit.HSPCLKDIV = 1;  // HSPCLKDIV = 2
EPwm2Regs.TBCTL.bit.CTRMODE = 2;  // up - down mode
EPwm2Regs.AQCTLA.all = 0x0090;    // set ePWM2A on CMPA up AND clear ePWM2A on CMPA down
EPwm2Regs.AQCTLB.all = 0;
EPwm2Regs.TBPRD = 3750;       // 10KHz - PWM signal
EPwm2Regs.CMPA.half.CMPA  = 375;//(Uint16)SIN;    // duty cycle first
EPwm2Regs.DBCTL.all= 11;      // use to get 2A and its complementry 2B (AHC mode)
EPwm2Regs.DBFED= deadband;         // DEAD BAND FOR FALLING EDGE 10uSEC= TBCLK*DBFED
EPwm2Regs.DBRED= deadband;         // DEAD BAND FOR RISING EDGE 10uSEC= TBCLK*DBFED
//-----------------------------PWM2 SETTING END--------------------------------------
//-----------------------------PWM3 SETTING START------------------------------------
EPwm3Regs.TBCTL.bit.CLKDIV =  0;  // CLKDIV = 1
EPwm3Regs.TBCTL.bit.HSPCLKDIV = 1;  // HSPCLKDIV = 2
EPwm3Regs.TBCTL.bit.CTRMODE = 2;  // up - down mode
EPwm3Regs.AQCTLA.all = 0x0090;    // set ePWM3A on CMPA up AND clear ePWM3A on CMPA down
EPwm3Regs.AQCTLB.all = 0;
EPwm3Regs.TBPRD = 3750;       // 10KHz - PWM signal
EPwm3Regs.CMPA.half.CMPA  = 375;//(Uint16)SIN;    // duty cycle first
EPwm3Regs.DBCTL.all= 11;      // use to get 3A and its complementry 3B (AHC mode)
EPwm3Regs.DBFED= deadband;         // DEAD BAND FOR FALLING EDGE 10uSEC= TBCLK*DBFED
EPwm3Regs.DBRED= deadband;         // DEAD BAND FOR RISING EDGE 10uSEC= TBCLK*DBFED
//-----------------------------PWM3 SETTING END--------------------------------------
//-----------------------------Phase shifting----------------------------------------
EPwm1Regs.TBCTL.bit.SYNCOSEL = 1;  // generate a syncout if CTR = 0
EPwm2Regs.TBCTL.bit.PHSEN = 1;     //disable // enable phase shift for ePWM2
EPwm2Regs.TBCTL.bit.SYNCOSEL = 0;  // syncin = syncout
EPwm2Regs.TBPHS.half.TBPHS = 0;   // control by hall sensor
EPwm3Regs.TBCTL.bit.PHSEN = 1;      //disable// enable phase shift for ePWM3
//EPwm3Regs.TBCTL.bit.SYNCOSEL = 0;   // syncin = syncout
EPwm3Regs.TBPHS.half.TBPHS = 0;
//----------------------------phase shift end-----------------------------------------
//----------------------------trip-zone start(protected)------------------------------
EALLOW;
EPwm1Regs.TZCTL.bit.TZA = 2;    // force ePWM1A to zero
EPwm1Regs.TZCTL.bit.TZB = 2;    // force ePWM1B to zero
EPwm1Regs.TZSEL.bit.OSHT6 = 1;  // select TZ6 as one shot over current source
EPwm2Regs.TZCTL.bit.TZA = 2;    // force ePWM1A to zero
EPwm2Regs.TZCTL.bit.TZB = 2;    // force ePWM1B to zero
EPwm2Regs.TZSEL.bit.OSHT6 = 1;  // select TZ6 as one shot over current source
EPwm3Regs.TZCTL.bit.TZA = 2;    // force ePWM1A to zero
EPwm3Regs.TZCTL.bit.TZB = 2;    // force ePWM1B to zero
EPwm3Regs.TZSEL.bit.OSHT6 = 1;  // select TZ6 as one shot over current source
EDIS;
//----------------------------trip-zone end(protected)--------------------------------
//----------------------------for adc intrupt----------------------------------------
```

```
EPwm2Regs.ETPS.all = 0x0100;      // Configure ADC start by ePWM2
/*
 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
*/
EPwm2Regs.ETSEL.all = 0x0A00;     // Enable SOCA to ADC
/*
 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
*/
//-----------------------------------------------------------------------------------
}
void Setup_adc(void)
{
    AdcRegs.ADCTRL1.all = 0;      // Reset ADC reg
    AdcRegs.ADCTRL1.bit.ACQ_PS = 7;    // 7 = 8 x ADCCLK
    AdcRegs.ADCTRL1.bit.SEQ_CASC =1;   // 1=cascaded sequencer
    AdcRegs.ADCTRL1.bit.CPS = 0;     // divide by 1
    AdcRegs.ADCTRL1.bit.CONT_RUN = 0; // stops after reaching end of sequence
    AdcRegs.ADCTRL2.all = 0;      // Reset ADC reg
    AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1 = 1;// 1=enable SEQ1 interrupt
    AdcRegs.ADCTRL2.bit.EPWM_SOCA_SEQ1 =1;// 1=SEQ1 start from ePWM_SOCA trigger
    AdcRegs.ADCTRL2.bit.INT_MOD_SEQ1 = 0;// 0= interrupt after every end of sequence
    AdcRegs.ADCTRL3.bit.SMODE_SEL= 0;    // SEQUENTIAL
    AdcRegs.ADCTRL3.bit.ADCCLKPS = 3; // ADC clock: FCLK = HSPCLK / 2 * ADCCLKPS,
        HSPCLK = 75MHz (see DSP2833x_SysCtrl.c)and FCLK = 12.5 MHz
    AdcRegs.ADCMAXCONV.all = 0;       // max no of conversion( 1 here )
    AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0;  // Setup ADCINA0 as 1st SEQ1 conv.  //
        frequency  (5-70)
    AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1;   // Reset sequencer
}
void Setup_eCAP(void)
{
//----------------------------------------------------------------------
//--- Configure eCAP unit for capture
//----------------------------------------------------------------------
ECap1Regs.ECEINT.all = 0;          // Disable all eCAP interrupts
ECap1Regs.ECCTL1.bit.CAPLDEN = 0;      // Disabled loading of capture results
ECap1Regs.ECCTL2.bit.TSCTRSTOP = 0;      // Stop the counter
ECap1Regs.TSCTR = 0;              // Clear the counter
ECap1Regs.CTRPHS = 0;             // Clear the counter phase register
```

```
ECap1Regs.ECCTL1.all = 0x01EE;        // ECAP control register 1
ECap1Regs.ECCTL2.all = 0x0096;        // ECAP control register 2
ECap1Regs.ECEINT.all = 0x0006;        // Enable desired eCAP interrupts
}
//-------------------------------ISR Start--------------------------------------------

interrupt void cpu_timer0_isr(void)
{
    CpuTimer0.InterruptCount++;
    speed= 0.9157509*duty;  // =3750*duty/4095directly prop to out voltage
    Ha=GpioDataRegs.GPADAT.bit.GPIO6;
    Hb=GpioDataRegs.GPADAT.bit.GPIO8;
    Hc=GpioDataRegs.GPADAT.bit.GPIO10;
    value = (4*(Ha)+2*(Hb)+1*(Hc));  // hall signal converted to case no HA-1, Hb-2 ,
        HC-no tape
    if (1)  //value==0)
    {
      switch(value)
      {
      case 0:
        EPwm1Regs.AQCTLA.all = 0x0000;    //normal case drive upper mosfet
        EPwm2Regs.AQCTLA.all = 0x0000;    //normal case drive upper mosfet
        EPwm3Regs.AQCTLA.all = 0x0000;    //normal case drive upper mosfet
        EPwm1Regs.DBCTL.all= 3;
        EPwm2Regs.DBCTL.all=3;
        EPwm3Regs.DBCTL.all= 3;
        break;
      case 1:
        EPwm1Regs.CMPA.half.CMPA  = 0;//(Uint16)speed;  //(Uint16)SIN;    //duty cycle
             first
        EPwm2Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    // duty cycle
            first
        EPwm3Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    // duty cycle
            first
        EPwm1Regs.AQCTLA.all = 0x0090;    //xxxxxnormal case drive upper mosfet
        EPwm2Regs.AQCTLA.all = 0x0060;    //normal case drive upper mosfet
        EPwm3Regs.AQCTLA.all = 0x0090;    // normal case drive upper mosfet
        EPwm1Regs.DBCTL.all= 3;//0;//3;
        EPwm2Regs.DBCTL.all=11;
        EPwm3Regs.DBCTL.all=11;
        break;
      case 2:
        EPwm1Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    //duty cycle
            first
        EPwm2Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    // duty cycle
            first
        EPwm3Regs.CMPA.half.CMPA  = 0;//(Uint16)speed;  //(Uint16)SIN;    // duty
            cycle first
        EPwm1Regs.AQCTLA.all = 0x0060;    //normal case drive upper mosfet
        EPwm2Regs.AQCTLA.all = 0x0090;    // normal case drive upper mosfet
        EPwm3Regs.AQCTLA.all = 0x0090;    //xxxxxnormal case drive upper mosfet
        EPwm1Regs.DBCTL.all= 11;
        EPwm2Regs.DBCTL.all= 11;
        EPwm3Regs.DBCTL.all= 3;//0;//3;
        break;
      case 3:
        EPwm1Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    //duty cycle
            first
        EPwm2Regs.CMPA.half.CMPA  =0;//(Uint16)speed; //(Uint16)SIN;    // duty cycle
            first
        EPwm3Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    // duty cycle
            first
        EPwm1Regs.AQCTLA.all = 0x0060;    //normal case drive upper mosfet
        EPwm2Regs.AQCTLA.all = 0x0090;    //xxxxxnormal case drive upper mosfet
        EPwm3Regs.AQCTLA.all = 0x0090;    //normal case drive upper mosfet
        EPwm1Regs.DBCTL.all= 11;
        EPwm2Regs.DBCTL.all= 3;//0;//3;
        EPwm3Regs.DBCTL.all= 11;
        break;
      case 4:
        EPwm1Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    //duty cycle
            first
        EPwm2Regs.CMPA.half.CMPA  =0;//(Uint16)speed; //(Uint16)SIN;    // duty cycle
            first
        EPwm3Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    // duty cycle
            first
        EPwm1Regs.AQCTLA.all = 0x0090;    //normal case drive upper mosfet
        EPwm2Regs.AQCTLA.all = 0x0090;    //xxxxxnormal case drive upper mosfet
        EPwm3Regs.AQCTLA.all = 0x0060;    //normal case drive upper mosfet
        EPwm1Regs.DBCTL.all= 11;
        EPwm2Regs.DBCTL.all= 3;//0;//3;
        EPwm3Regs.DBCTL.all= 11;
        break;
      case 5:
        EPwm1Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    //duty cycle
            first
        EPwm2Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    // duty cycle
            first
        EPwm3Regs.CMPA.half.CMPA  =0;// (Uint16)speed;  //(Uint16)SIN;    // duty
            cycle first
        EPwm1Regs.AQCTLA.all = 0x0090;    //normal case drive upper mosfet
        EPwm2Regs.AQCTLA.all = 0x0060;    //normal case drive upper mosfet
        EPwm3Regs.AQCTLA.all = 0x0090;    //xxxxxnormal case drive upper mosfet
        EPwm1Regs.DBCTL.all= 11;
        EPwm2Regs.DBCTL.all= 11;
        EPwm3Regs.DBCTL.all= 3;//0;//3;
        break;
      case 6:
        EPwm1Regs.CMPA.half.CMPA  = 0;//(Uint16)speed;  //(Uint16)SIN;    //duty cycle
             first
        EPwm2Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    // duty cycle
            first
```

```
    EPwm3Regs.CMPA.half.CMPA  = (Uint16)speed;  //(Uint16)SIN;    // duty cycle first
    EPwm1Regs.AQCTLA.all = 0x0090;    //xxxxxnormal case drive upper mosfet
    EPwm2Regs.AQCTLA.all = 0x0090;    //normal case drive upper mosfet
    EPwm3Regs.AQCTLA.all = 0x0060;    //normal case drive upper mosfet
    EPwm1Regs.DBCTL.all= 3;//0;//3;
    EPwm2Regs.DBCTL.all= 11;
    EPwm3Regs.DBCTL.all= 11;
    break;
   case 7:
    EPwm1Regs.AQCTLA.all = 0x0000;    //do nothing
    EPwm2Regs.AQCTLA.all = 0x0000;    //do nothing
    EPwm3Regs.AQCTLA.all = 0x0000;    //do nothing
    EPwm1Regs.DBCTL.all= 3;
    EPwm2Regs.DBCTL.all= 3;
    EPwm3Regs.DBCTL.all= 3;
    break;
   }
   call_dog2();
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
  }
  else
  {
   EPwm1Regs.CMPA.half.CMPA  = 1850;//(Uint16)SIN;   //duty cycle first
   EPwm2Regs.CMPA.half.CMPA  = 1850;//(Uint16)SIN;   // duty cycle first
   EPwm3Regs.CMPA.half.CMPA  = 1850;//(Uint16)SIN;   // duty cycle first
   call_dog2();
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
  }
}
interrupt void  adc_isr(void)
{
  duty = AdcMirror.ADCRESULT0;  // store results global
   AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;       // Reset SEQ1 for next adc sequence
   AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;   // Clear INT SEQ1 bit
   call_dog2();
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE
}
interrupt void eCAP1_isr(void)
{
 ECap1Regs.ECCLR.bit.INT = 1;        // Clear the ECAP1 interrupt flag
 ECap1Regs.ECCLR.bit.CEVT1 = 1;        // Clear the CEVT1 flag
 ECap1Regs.ECCLR.bit.CEVT2 = 1;        // Clear the CEVT2 flag
 ECap1Regs.ECCLR.bit.CEVT3 = 1;        // Clear the CEVT3 flag
 ECap1Regs.ECCLR.bit.CEVT4 = 1;        // Clear the CEVT4 flag
 // speed of motor using ecap time capture method
 Omega = (20/((int32)ECap1Regs.CAP2 + (int32)ECap1Regs.CAP4));
 PieCtrlRegs.PIEACK.all = PIEACK_GROUP4; // Must acknowledge the PIE group 4
}
```

LIST OF PAPERS BASED ON THESIS

1. T.J.E.Miller, "*Brushless Permanent-Magnet And Reluctance Motor Driver,*" *Oxford University Press, New york, 1989.*

2. Bimal K.Bose, "*Modern power electronics and AC drives*", Prentice Hall, Upper Saddle River, New Jersey, 2005.

3. Jian Zhao/Yangwei Yu, "*Brushless DC Motor Fundamentals Application Note*", AN047, Monolithic Power, July 2011.

4. Yedale Padmaraja., "*Brushless DC Motor Fundamentals*", AN885, Microchip Technology Inc, 2003.

5. K. Safala and T.Teja Sreenu, "*A Novel DSP Controller Based Position And Speed Control of BLDC Motor on MATLAB and SIMULINK*", Advances in Electrical and Computer Engineering ISSN: 1582-7445.

6. Vandana Govindan T.K, Anish Gopinath and S.Thomas George, "*DSP based Speed Control of Permanent Magnet Brushless DC Motor*", IJCA Special Issue on Computational Science - New Dimensions and Perspectives, NCCSE, 2011.

7. Petr Staszko, "*Three-Phase BLDC Sensorless Motor Control Application*", DRM144, Freescale Semiconductor, Feb 2014.

8. Bilal Akin and Manish Bhardwaj, "*Trapezoidal Control of BLDC Motors Using Hall Effect Sensors*", SPRABQ6, Texas Instruments, July 2013.

9. Texas Instruments, "*TMS320F28335, F28334, F28332 Digital Signal Controllers (DSCs) Data Manual*", SPRS439M, June 2007 Revised August 2012.

10. Texas Instruments, "*TMS320x28xx, 28xxx Peripheral Reference Guide*", SPRU566L, June 2003 Revised June 2015.

11. Texas Instruments, "*TMS320x2833x, 2823x System Control and Interrupts*",SPRUFB0D September 2007 Revised March 2010.

12. Texas Instruments, "*TMS320x2833x, 2823x Enhanced Pulse Width Modulator (ePWM) Module*", SPRUG04A, October 2008 Revised July 2009.

13. Texas Instruments, "*TMS320x2833x, 2823x Enhanced Capture (eCAP) Modulee*", SPRUFG4A, August 2008 Revised June 2009.

14. Texas Instruments, "*TMS320x2833x Analog-to-Digital Converter (ADC) Module*", SPRU812A, September 2007 Revised October 2007.