ENERGY EFFICIENT PUMPSET SYSTEM USING A VARIABLE SPEED DRIVE

A PROJECT REPORT

Submitted by

KAVITHA.V
PRITHIVI.K
SWATHIKA.V

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRICAL AND ELECTRONICS ENGINNEERING

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE: 641006

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2009

BONAFIDE CERTIFICATE

BONAFIDE CERTIFICATE

Certified that this project report entitled **"Energy Efficient Pump Set System Using A Variable Speed Drive"** is the bonafide work of

Miss.Kavitha.V -Register No. 71205105018
Miss. Prithivi.K -Register No. 71205105033
Miss.Swathika.V -Register No. 71205105050

who carried out the project work under my supervision.

Signature of the Head of the Department
(Prof.K.Regupathy Subramanian)

20/4/09
Signature of the Guide
(Mrs.K.Malarvizhi)

Certified that the candidate with university Register No. 71205105018 71205105033 71205105050 **was examined in project viva voce Examination held on** 24.4.09

Internal Examiner

24/4/09
External Examiner

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING,
KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE 641 006.

INDUSTRY CERTIFICATE

HENDRA
PUMPSETS

MAHENDRA

Mahendra Submersible Pumps (P) Ltd
423/2, Kalapatty Road, Coimbatore - 641 014.
Phone : 2628067, 2627097 Fax : 0422 - 2631017
E-mail : mahendrasub@mahendrapumps.in Web Site : www.mahendrapumps.in

Date : 2.4.2009

TO WHOMEVER IT MAY CONCERN

This is to certify that the following final year B.E.(Electrical & Electronics) students of KUMARAGURU COLLEGE OF TECHNOLOGY,COIMBATORE. have completed the project ***ENERGY EFFICIENT PUMPSET SYSTEM USING A VARIABLE SPEED DRIVE*** in our concern for a period from Aug 2008 to Mar 2009.

KAVITHA.V	71205105018
PRITHIVI.K	71205105033
SWATHIKA.V	71205105050

During this period, they have shown interest in designing and conducting the tests required and has done a good work towards completion of the project.

Their performance was good. We at Mahendra, wish them a very successful future.

Mahendra Submersible Pumps (P) Ltd

SYNOPSIS

SYNOPSIS

Pump sets used for domestic and commercial purposes run at a constant speed. They consume the same unit of energy for different levels of usage. If they are run according to the requirements, the energy conserved would be significant.

A Variable Speed Drive (VSD) for the constant speed centrifugal pumps is proposed in this project. The system implemented converts a single phase input of 230V to DC by using a single phase diode bridge rectifier and a capacitive filter which assists in stabilizing the output DC voltage produced. After that, the system provides the gate drive signal to a three phase Pulse Width Modulated (PWM) inverter driving a single phase squirrel cage induction motor (coupled to the pump).The PWM signal (gate drive signal) is generated by PIC16F72 microcontroller .The program for the same is written. The three phase IGBT inverter uses the DC voltage supplied from the diode bridge and the gate drive signals to produce a single phase sinusoidal output which drives the induction motor at a speed according to the user's settings.

The implementation of VSD in a pump proves to be energy efficient. Also, flow control in industries become easy with the use of a VSD.

ACKNOWLEDGEMENT

ACKNOWLEDGMENT

We wish to express our deep sense of gratitude and indebtedness to our project guide **Mrs.K.Malarvizhi**, Assistant Professor, Department of Electrical and Electronics Engineering, Kumaraguru College of Technology, Coimbatore for her valuable guidance and constructive suggestions, the keen and constant inspiration and interest given by her at all stages throughout our project.

We would also express our thanks to our project coordinator **Mrs.R.Mahalakshmi**, Assistant Professor, Department of Electrical and Electronics Engineering, Kumaraguru College of Technology, Coimbatore, for her guidance to our project.

We would like to thank **Mr. S.Ramakrishnan**, Assistant Manager-Design, Mahendra Submersible Pumps (P) Ltd, Coimbatore, for his valuable guidance and technical support throughout our project.

We are also thankful to all the faculty members including supporting staff of our department for their cooperation, advice and assistance during the course of our project.

And, finally we thank the Almighty for his blessings, to bring out our project a great success.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 NEED FOR THE PROJECT

Pumps are the single largest user of electricity in agricultural and industrial sectors of our country, accounting to millions of tonnes of carbon dioxide (CO_2) emissions annually.

It is a well-known fact that in all the available commercial pump sets the speed is always constant irrespective of the water level. This also accounts to the wear and tear of the pump sets thereby increasing its maintenance. As commercial pumps are coupled to induction motors their speed control is difficult owing to the non linear and complex structure of the induction motors. By implementing a Variable Speed Drive (VSD), the speed can be varied according to the user's settings which according to the affinity laws lead to reduced consumption of energy.

Over the past decade, the field of power electronics has passed through rapid development due to the advancement of many modern technologies such as; Microcontrollers, DSP Cards and Data Acquisition Cards. The aforementioned technologies made power electronics projects and applications easier to implement and more accessible. In this project, a Variable Speed Drive to control the speed of a single-phase induction motor (coupled to the pump) using PIC16F72 microcontroller is designed and implemented.

The complete system will consist of two sections; a Power Circuit and a Control Circuit. The power circuit consists of the single-phase diode bridge rectifier, C Filter and Three Phase PWM Inverter. The control circuit consists of the PIC16F72 microcontroller and Gate Drivers.

1.2 OBJECTIVE OF THE PROJECT

The main objective of the project is to design and implement a variable speed drive to control the speed of commercial pumps, thus reducing the energy consumption. The drive circuit designed also provides operation of the pump above its rated speed. Flow control in industries can be made efficient with the use of VSD.

1.3 SCOPE OF THE PROJECT

The project can be used in domestic, industrial and agricultural sectors where the static head on which the pump has to operate varies with time. Also, as the pump is run as per requirement the wear and tear of the machine can be reduced.

1.4 ADVANTAGES OF THE PROJECT

a) Energy conservation.

b) Use power efficiently to produce the necessary torque at a given speed.

c) Low maintenance and repair costs.

d) Linking the pump system to other automation systems is easy.

1.5 ORGANISATION OF THE REPORT

Chapter 1 discusses about the need, objective, scope of the project and advantages of using VSD in speed control of centrifugal pumps.

Chapter 2 deals with the brief description of the centrifugal pumps with its vital terms and the Affinity Laws, governing the world of pumps.

Chapter 3 explains about the basics of single phase induction motor with appropriate diagrams and the speed control adopted in the project.

Chapter 4 gives the hardware description of the Diode Bridge Rectifier, PIC16F72, Inverter with concepts of PWM and the overall circuit of the project.

Chapter 5 details the software program of the PIC16F72 to generate the PWM pulses that control the IGBTs.

Chapter 6 explains about the energy conservation with relevant mathematical calculations and comparisons.

Chapter 7 gives the conclusion of the project.

CHAPTER 2

CENTRIFUGAL PUMP

2. CENTRIFUGAL PUMP

2.1 BASIC FUNCTION

The purpose of pumps is to transfer liquids from a source to a destination. The centrifugal pump is a mechanical device for increasing the pressure of liquid. In passing through the pump, the liquid is accelerated in the impeller, discharging into the casing at high velocity. This energy used is converted into pressure of the liquid as effectively as possible. Centrifugal pumps account to 80% of the industrial pumps.

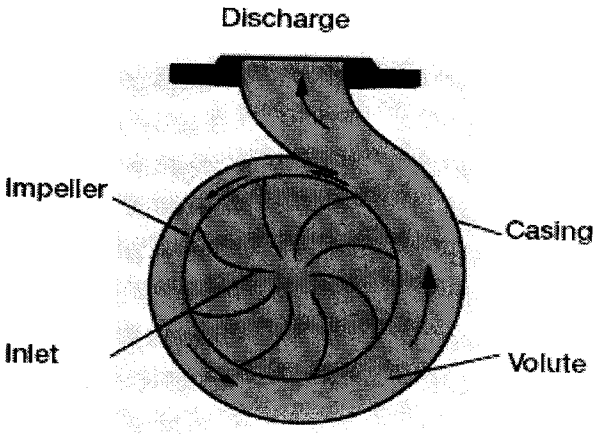

Fig.1. Centrifugal Pump

2.2 TERMINOLOGY ASSOCIATED TO PUMPS

Head - The net work done on a unit weight of water by the pump impeller. It is the amount of energy added to the water between the suction and discharge sides of the pump. Pump head is measured as pressure difference between the discharge and suction sides of the pump.

Static head - The vertical distance from the water level at the source to the highest point where the water must be delivered. It is the sum of static lift and static discharge. Static head is independent of the system discharge and is constant for all values of discharge. However, it is possible that the static head may vary over time due to the changes in the system.

Operating point - A centrifugal pump can operate at a combination of head and discharge points given by its pump curve. The particular combination of head and

discharge at which a pump is operating is called the pump's operating point. Once this point is determined, brake power, efficiency, and net positive suction head required for the pump can be obtained from the set of pump curves.

Pump curves-The pump curves show the technical performance of the pump. The horizontal axis shows the flow rate and the vertical axis shows the head and power generated.

A system curve, normally plotted together with the pump curve describes the static head and resistance of the pipeline. The operating point of the pump is at the intersection of the system curve and the pump curve.

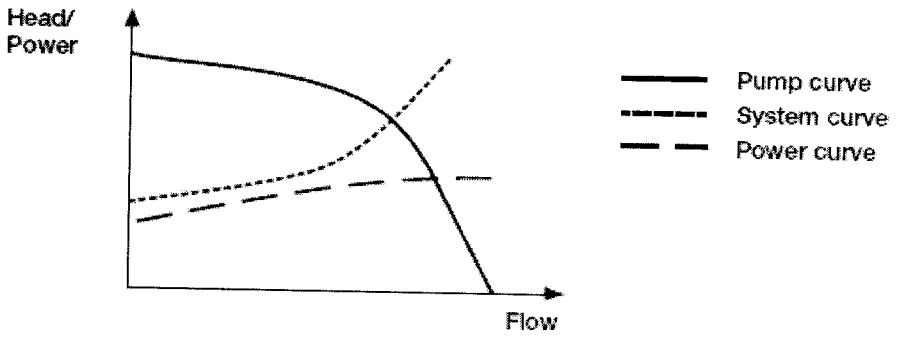

Fig.2. Pump Curves

2.3 AFFINITY LAWS

The affinity laws below describe the relation between the rotational speeds of the pump (n), flow rate (Q), head generated (H) and power absorbed (P).

Speed and flow are directly proportional - Flow

$$\frac{Q_1}{Q_2} = \frac{n_1}{n_2}$$

Head is proportional to the square of the speed - Head

$$\frac{H_1}{H_2} = \left(\frac{n_1}{n_2}\right)^2$$

Power is proportional to the speed or flow cubed - Power

$$\frac{P_1}{P_2} = \left(\frac{n_1}{n_2}\right)^3$$

CHAPTER 3

INDUCTION MOTOR

3. INDUCTION MOTOR

3.1 INTRODUCTION

Single-phase AC induction motor is least expensive and low maintenance type motor. This type of motor has only one stator winding (main winding) and operates with a single-phase power supply. In all single-phase induction motors, the rotor is the squirrel cage type.

The cage induction motor has electrical circuits on both sides i.e., on stator and rotor but it is singly fed machine in the sense that all external electrical input into the stator circuit only. There is no external electrical input into the rotor circuit. According to electromagnetic induction principle, the rotor currents are induced by the rotating magnetic field of the stator. This is a great advantage of the machine, that there are no rubbing contacts needed to make electrical connections from stationary environment to the rotating motor.

The single-phase induction motor is not self-starting. When the motor is connected to a single-phase power supply, the main winding carries an alternating current. This current produces a pulsating magnetic field. Due to induction, the rotor is energized. As the main magnetic field is pulsating, the torque necessary for the motor rotation is not generated. This will cause the rotor to vibrate, but not to rotate. Hence, the single-phase induction motor is required to have a starting mechanism that can provide the starting kick for the motor to rotate.

3.2 OPERATING PRINCIPLE

When the stator is energized from a single-phase supply, a rotating magnetic field is created in the air gap of the motor. The rotating magnetic flux will induce voltages in both the stator coils and the rotor coils, in the same way as the alternating flux in a transformer core induces voltages in both the primary and the secondary coils. If the rotor is held fast and prevented from rotating, the set-up is basically a three-phase induction motor transformer with the secondary windings short-circuited. Therefore there will be induced voltages in the rotor phase coils and the resulting currents. The electromagnetic forces resulting from the interaction of the current in the rotor conductors and the air gap flux will result in a torque, which will cause the rotor to rotate. The basic cause of the

creation of the induced voltage, the current and the torque is the relative motion between the air-gap flux and the rotor conductors. Therefore the direction of the rotation will be such that to minimize this relative velocity. In other words, the rotor will rotate in the same direction as the rotating magnetic field. But the rotor speed increases, the relative speed between the rotor conductors and field becomes less and less. Therefore the induced e.m.f., the current and the resulting torque also becomes less. There will be no induced e.m.f., if the rotor spins at the synchronous speed, because there will be no relative motion between the field and the rotor conductors.

The starting mechanism of the single-phase induction motor is mainly an additional stator winding (start/auxiliary winding). The start winding can have a series capacitor and/or a centrifugal switch. When the supply voltage is applied, current in the main winding lags the supply voltage due to the main winding impedance. At the same time, current in the start winding leads/lags the supply voltage depending on the starting mechanism impedance. Interaction between magnetic fields generated by the main winding and the starting mechanism generates a resultant magnetic field rotating in one direction. The motor starts rotating in the direction of the resultant magnetic field. Once the motor reaches about 75% of its rated speed, a centrifugal switch disconnects the start winding. From this point on, the single-phase motor can maintain sufficient torque to operate on its own.

3.3 PERMANENT SPLIT CAPACITOR (PSC) MOTOR

PSC motors are unidirectional, which means they are designed to rotate in one direction. By adding either extra windings, and external relays and switches, or by using gear mechanisms, the direction of rotation can be changed.

A permanent split capacitor (PSC) motor has a run type capacitor permanently connected in series with the start winding. This makes the start winding an auxiliary winding once the motor reaches the running speed. Since the run capacitor must be designed for continuous use, it cannot provide the starting boost of a starting capacitor. The typical starting torque of the PSC motor is low, from 30% to 150% of the rated torque. PSC motors have low starting current, usually less than 200% of the rated current, making them excellent for applications with high on/off cycle rates.

The PSC motors have several advantages. The motor design can easily be altered for use with speed controllers. They can also be designed for optimum efficiency and High-Power Factor (PF) at the rated load. They're considered to be the most reliable of the single-phase motors, mainly because no centrifugal starting switch is required.

Permanent split-capacitor motors have a wide variety of applications depending on the design. These include fans, blowers with low starting torque needs and intermittent cycling uses, such as adjusting mechanisms, gate operators and garage door openers.

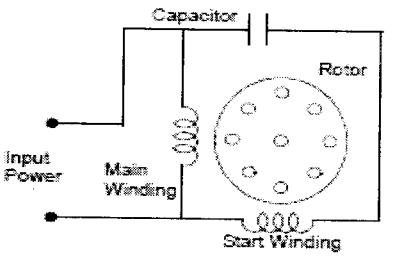

Fig.3. A PSC Motor

3.4 OPEN-LOOP V/f METHOD OF SPEED CONTROL

For low cost, low performance drive, open-loop constant V/f control is normally employed. With open-loop speed control, the rotor speed will be less than the synchronous speed by slip speed. In other words, the desired speed, ω^*, will differ from the actual speed by slip speed. The slip speed on the other hand, depends on load. To improve the performance or the speed regulation, slip speed can be estimated and added to the reference speed – slip compensation technique.

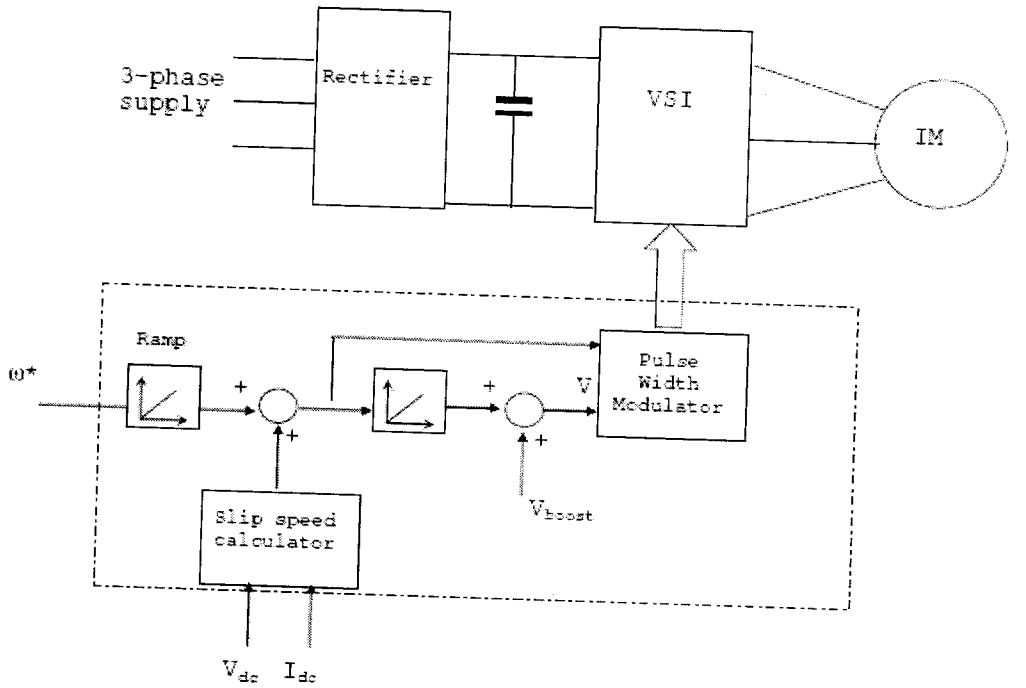

Fig.4.Open Loop V/f Control

Constant V/f

Approximates constant air-gap flux when E_{ag} is large

$E_{ag} = k\ f\ \Phi_{ag}$

Φ_{ag}= constant = $E_{ag}/f \simeq v/f$

Where E_{ag} is the emf of air gap and $\Phi_{ag\ is}$ the air gap flux.

Speed is adjusted by varying f - maintaining V/f constant to avoid flux saturation.

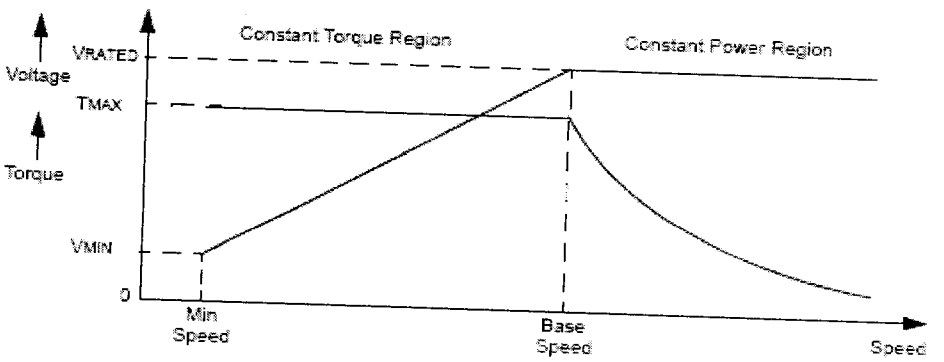

Fig.5. V/f Curve

The voltage and the frequency are varied at a constant ratio up to the base speed. The flux and the torque remain almost constant up to the base speed. Beyond the base speed, the supply voltage cannot be increased. Increasing the frequency beyond the base speed results in the field weakening and the torque reduces. Above the base speed, the torque governing factors become more nonlinear as the friction and windage losses increase significantly. Due to this, the torque curve becomes nonlinear. Based on the motor type, the field weakening can go up to twice the base speed. This control is the most popular in industries and is popularly known as the constant V/f control. By selecting the proper V/f ratio for a motor, the starting current can be kept well under control. This avoids any sag in the supply line, as well as heating of the motor.

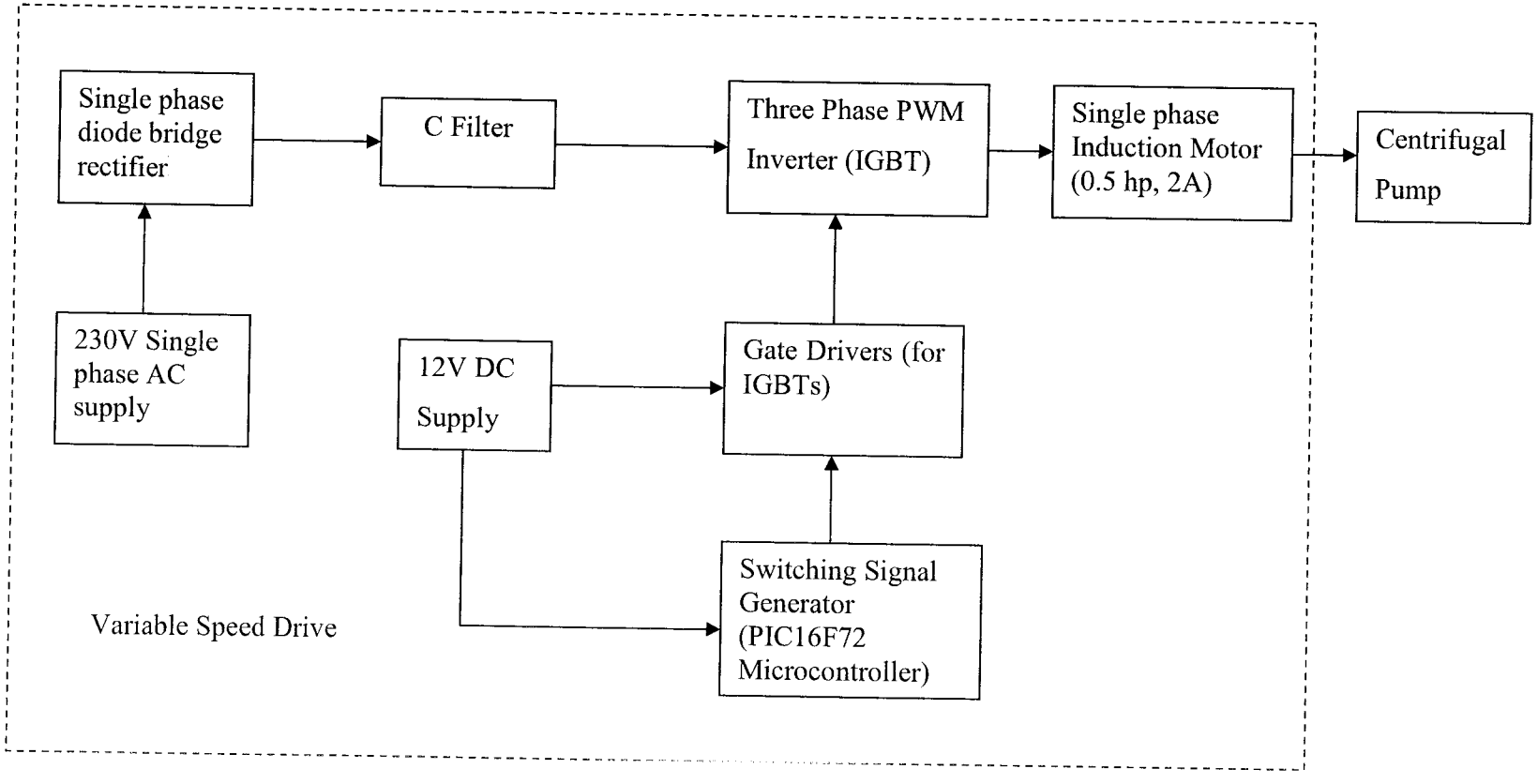

Fig.6. Variable Speed Drive (VSD) Coupled To A Pump -Block Diagram

CHAPTER 4

HARDWARE IMPLEMENTATION

4. HARDWARE IMPLEMENTATION

4.1 POWER CIRCUIT OF VSD

4.1.1 DIODE BRIDGE RECTIFIERS

A diode bridge rectifier is an arrangement of diodes in a bridge configuration that provides same polarity of output voltage for either polarity of input voltage. A bridge rectifier provides full wave rectification from a 2 wire AC input resulting in low cost and weight. The essential feature of the diode bridge rectifier is that the polarity of the output is always the same regardless of the polarity of the input.

Basic Function

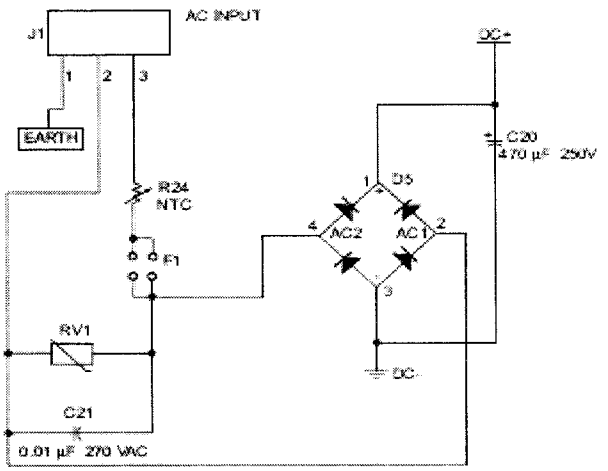

Fig.7. Single Phase Diode Bridge Rectifier

When the input to the circuit is applied to the diagonally opposite corners of the network, and the output is taken from the remaining two corners. Let us assume that there is appositive potential at the point A and negative potential at the point B. The positive potential at point A will forward bias D3 and reverse bias D4.The negative potential at point B will forward bias D1 and reverse bias D2.At this time D1 and D3 are forward biased and will allow current to pass through them;D4 and D2 are reverse biased and will block current flow.

One half cycle later, the polarity of the input reverse forward biasing D2 and D4, reverse biasing D1 and D3.Since the current flows through the load during both half

cycles of the applied voltage, this bridge rectifier is a full wave rectifier. One advantage of bridge rectifier over a conventional full wave rectifier is that it produces a voltage output that is nearly twice that of the conventional full wave circuit.

Output Smoothing

Output of the bridge rectifier is stabilized by means of a reservoir capacitor or smoothing capacitor. The capacitor is used to lessen the variation in rectified ac output voltage waveform from the bridge. The capacitor provides low impedance path to the ac component of the output reducing the ac voltage across the ac current through the load.

4.1.2THREE PHASE PWM INVERTER

An inverter is a device that converts DC power into AC power at desired output voltage level. The inverter output can be single-phase or poly phase and can have square wave, sine wave, PWM wave, stepped wave at the output.

Role Of Inverters

In the dc machines the field and armature are fixed in space by the commutator. But in the case of induction machines no additional components exist to separate the field(to produce the flux) from the armature(to produce the torque) channels to the optimum space angle of 90 electrical degrees between them. In the place of commutator, the induction machine acquires the functionality of the commutator with an inverter. The inverter controls both the magnitude of the current and its phase allowing the machine's flux and torque channels to be decoupled by controlling precisely and injecting the flux and torque-producing currents in the induction machine to match the required rotor flux linkages and electromagnetic torque.

Insulated Gate Bipolar Transistors (IGBT)

Insulated gate bipolar transistor (IGBT) is development in power MOS technology. These devices retain the high input impedance of the MOSFET and have a low on-state voltage drop, which is comparable to that of a bipolar transistor and which varies only moderately with junction temperature. However the turn-off time of an IGBT can be significantly greater than that of a power MOSFET. An IGBT is a voltage controlled device and it has lower switching and conducting loss. An IGBT is a voltage controlled device and it has lower switching and conducting loss. An IGBT is inherently faster than BJT.

Three Level Inverter Using IGBT

Three phase inverters are commonly used to supply three phase loads. It is possible to supply a three phase load by means of separate single phase inverters, where each inverter produces an output displaced by 120 electrical degrees with respect to each other. The most frequently used three phase inverter circuit consists of three legs, one for each phase. The advantage of this circuit is, output transformer is not necessary and also requires less number of IGBTs.

4.1.3 PULSE WIDTH MODULATION TECHNIQUES

Since the inverter contains electronic switches, it is possible to control the output voltage as well as optimize the harmonics by performing multiple switching within the inverter with the constant dc input voltage V_{dc}

CLASSIFICATION OF PWM TECHNIQUES

The classifications of PWM techniques can be given as follows:

- Sinusoidal PWM (SPWM)
- Selected harmonic elimination (SHE) PWM
- Minimum ripple current PWM
- Space-vector PWM (SVM)
- Random PWM
- Hysteresis band current control PWM
- Sinusoidal PWM with instantaneous current control
- Delta modulation
- Sigma-delta modulation

4.1.4 BASIC SCHEMATIC OF THE PROJECT

The circuit below shows the control of the single-phase permanent split capacitor (PSC) motor with a three-phase inverter bridge. One end of the main winding and start windings of the IM are connected to one half bridges each. The other ends are tied together and

connected to third half bridge. The 90 degree phase shift between the main and start winding of an IM is achieved in the software configuration itself. This eliminates the use of capacitor in a PSC motor.

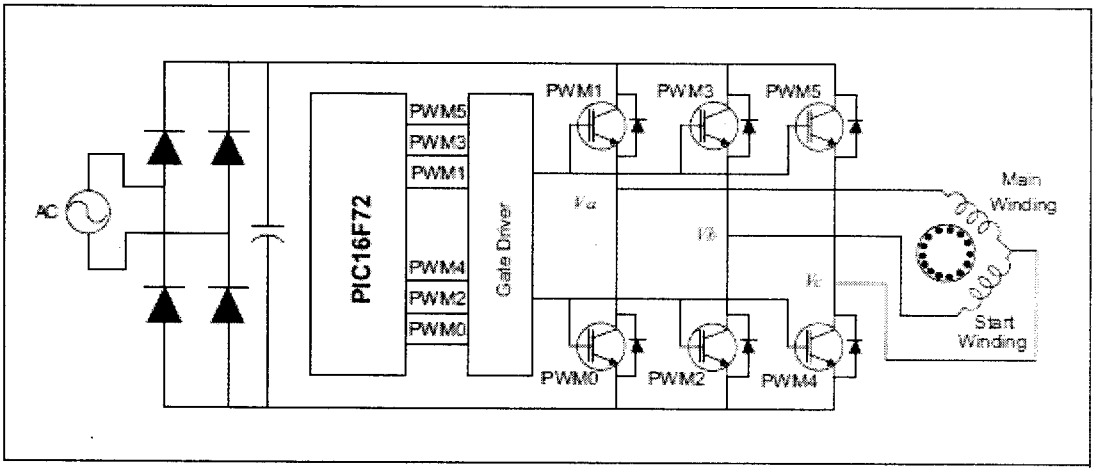

Fig.8. Basic Schematic

Explanation

The voltages V_a, Vb and V_c should be controlled to achieve the phase difference between the effective voltages across the main and start windings to have a 90 degree phase shift to each other. The voltages across phase a and phase bare out of phase with each other and the phase difference between phase a and c is θ degrees.

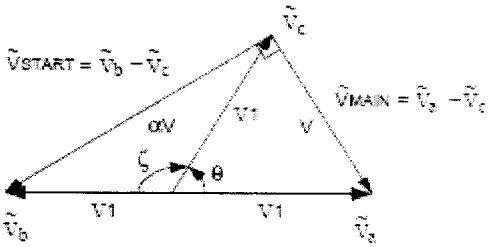

Fig.9. Voltage Phasor Diagram.

By applying basic trigonometry, θ can be calculated by

Angle $\theta = 180^{\circ} - 2\tan^{-1}(\alpha)$

By applying the Pythagorean Theorem, the voltage vector V1 can be calculated as

$V1 = V \times \frac{(1+\alpha^2)}{2}$

The turns ratio of the start winding to the main winding is defined by

$$\alpha = \frac{V_{START}}{V_{MAIN}}$$

where V_{MAIN} and V_{START} are the effective voltage across the main winding and start winding respectively.

In order to have equal voltage stress on all devices, thus improving the device utilization and provide the maximum possible output voltage for a given DC bus voltage, all three inverter phase voltages are kept at the same amplitude as follows:

$$|V_a| = |V_b| = |V_c|$$

The effective voltage across the main and start winding is given as:

$$V_{MAIN} = V_a - V_c$$

$$V_{START} = V_b - V_c$$

The voltages are shown in the phasor diagram in figure: 9.

Because the turn ratio remains constant for a given motor, α can be a compile time option. With this, θ and V1 can be pre-computed for a given motor. This simplifies the run time calculation. Based on the phase angle, phase voltages V_a, V_b and V_c can be calculated as follows:

$$V_a = V1 \times \cos(\omega t) + \frac{V_{dc}}{2}$$

$$V_b = -V1 \times \cos(\omega t) + \frac{V_{dc}}{2}$$

$$V_c = V1 \times \cos(\omega t \pm \theta) + \frac{V_{dc}}{2}$$

V_{dc} is the DC bus voltage, and ωt is the angular velocity of the electrical cycle. The direction of rotation can be easily controlled by adding or subtracting θ in the V_c calculation.

4.2 CONTROL CIRCUIT OF VSD

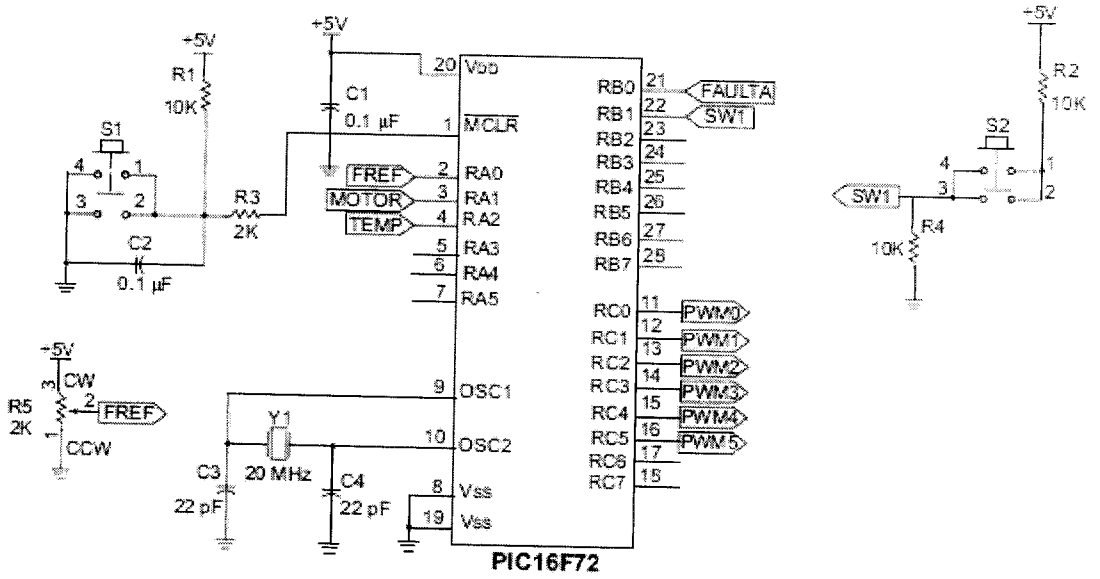

Fig.10. PIC16F72 Interface

4.2.1 PIC16F72

FEATURES

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- 8-bit, 5-channel analog-to-digital converter
- Capture, compare, PWM (CCP) module
  - -Capture is 16-bit, max. resolution is 12.5ns
  - -Compare is 16-bit, max. resolution is 200ns
  - -PWM max. resolution is 10-bit
- 1,000 erase/write cycle FLASH program memory typical
- Programmable code protection
- Processor read access to program memory

PERFORMANCE AND TECHNOLOGY

- Only 35 single word instructions to learn
- All single cycle instructions except for program branches, which are two-cycle

- Operating speed: DC-200MHZ clock input
  DC-200ns instruction cycle
- 2K x 14 words of Program Memory
- 128 x 8 bytes of Data Memory (RAM)
- Interrupt capability
- Eight-level deep hardware stack
- Direct, Indirect and Relative Addressing modes
- Low power consumption, high speed CMOS FLASH technology
- Fully static design
- Wide operating voltage range: 2.0V to 5.5V
- Industrial temperature range

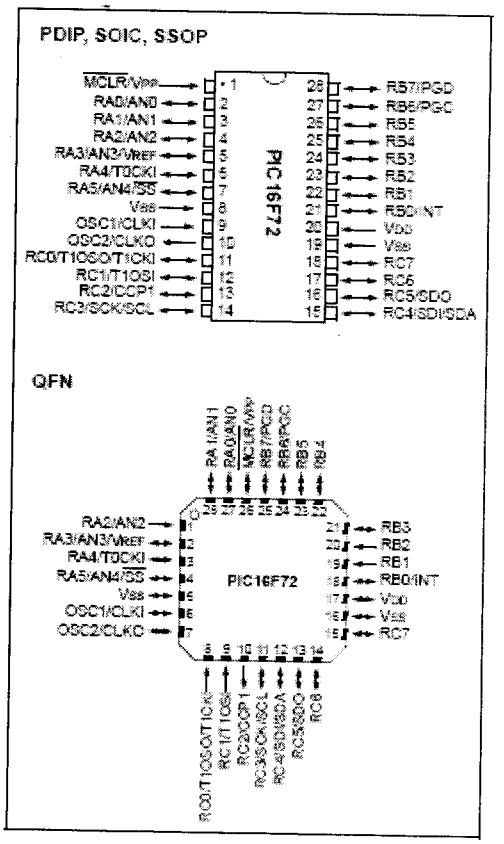

Fig.11. Pin Diagram

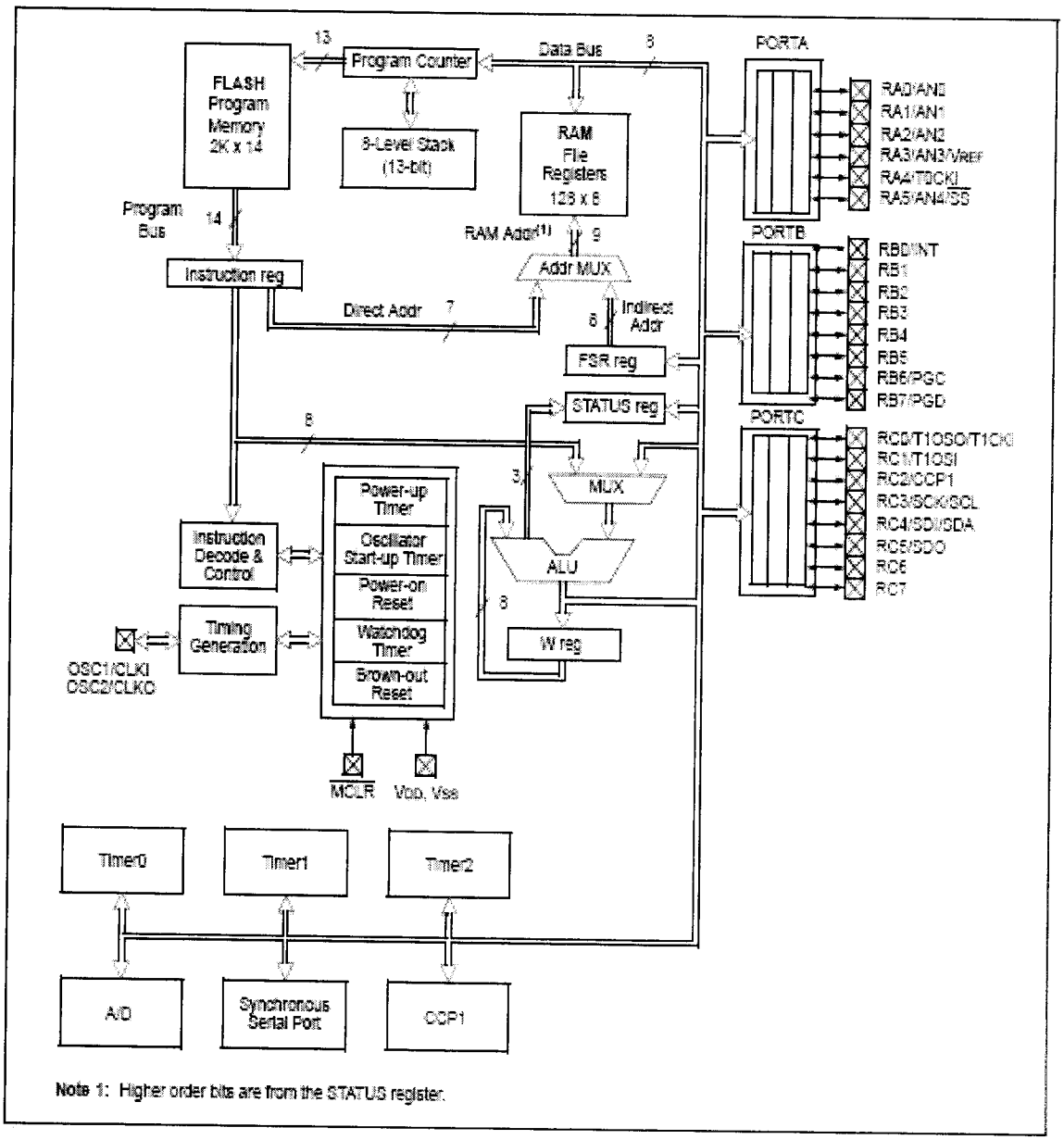

Fig.12. Architecture Of PIC16F72

The program memory contains 2K words, which translate to 2048 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 128 bytes. There are 22 I/O pins that are user configurable on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupts
- Timer0 clock input
- Timer1 clock/oscillator
- Capture/Compare/PWM
- A/D converter
- SPI/I2C

MEMORY ORGANISATION

There are two memory blocks in the PIC16F72 device. They are the program memory and the data memory. Each block has separate buses so that concurrent access can occur.

a. Program Memory Organization

PIC16F72 devices have a 13-bit program counter capable of addressing a 8K x 14 program memory space. The address range for this program memory is 0000h - 07FFh. Accessing a location above the physically implemented address will cause a wraparound. The RESET Vector is at 0000h and the Interrupt Vector
is at 0004h.

b. Data Memory Organization

The Data Memory is partitioned into multiple banks that contain the General Purpose Registers and the Special Function Registers. Bits RP1 (STATUS<6>) and RP0 (STATUS<5>) are the bank select bits. Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain SFRs. Some "high use" SFRs from one bank may be mirrored in another bank, for code reduction and quicker access (e.g., the STATUS register is in Banks 0 - 3).

General Purpose Register File

The register file can be accessed either directly, or indirectly, through the File Select Register FSR.

Special Function Registers

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. The Special Function Registers can be classified into two sets: core (CPU) and peripheral.

I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

TIMER0 MODULE

The TIMER0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

TIMER1 MODULE

The TIMER1 module timer/counter has the following features:

- 16-bit timer/counter

(Two 8-bit registers; TMR1H and TMR1L)

- Readable and writable (both registers)
- Internal or external clock select
- Interrupt on overflow from FFFFh to 0000h
- RESET from CCP module trigger

TIMER2 MODULE

The Timer2 module timer has the following features:

- 8-bit timer (TMR2 register)
- 8-bit period register (PR2)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR2 match of PR2
- SSP module optional use of TMR2 output to generate clock shift

READING PROGRAM MEMORY

The FLASH Program Memory is readable during normal operation over the entire VDD range. It is indirectly addressed through Special Function Registers (SFR). Up to 14-bit wide numbers can be stored in memory for use as calibration parameters, serial numbers, packed 7-bit ASCII, etc. Executing a program memory location containing data that forms an invalid instruction results in a NOP.

There are five SFRs used to read the program and memory:

- PMCON1
- PMDATL
- PMDATH
- PMADRL
- PMADRH

The program memory allows word reads. Program memory access allows for checksum calculation and reading calibration tables.

When interfacing to the program memory block, the PMDATH:PMDATL registers form a two-byte word, which holds the 14-bit data for reads. The PMADRH: PMADRL registers form a two-byte word, which holds the 13-bit address of the FLASH location

being accessed. This device has up to 2K words of program FLASH, with an address range from 0h to 07FFh. The unused upper bits PMDATH<7:6> and PMADRH<7:5> are not implemented and read as zeros.

CAPTURE/COMPARE/PWM (CCP) MODULE

The CCP (Capture/Compare/PWM) module contains a 16-bit register that can operate as a:

- 16-bit capture register
- 16-bit compare register
- PWM master/slave duty cycle register.

Capture/Compare/PWM Register1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. All are readable and writable.

Capture Mode

In Capture mode, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 register when an event occurs on pin RC2/CCP1. An event is defined as:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

An event is selected by control bits CCP1M3:CCP1M0 (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit CCP1IF (PIR1<2>) is set. It must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value is overwritten by the new captured value.

Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against the TMR1 register pair value. When a match occurs, the RC2/CCP1 pin is:

- Driven High
- Driven Low
- Remains Unchanged

The action on the pin is based on the value of control bits CCP1M3:CCP1M0 (CCP1CON<3:0>). At the same time, interrupt flag bit CCP1IF is set. The output may become inverted when the mode of the module is changed from Compare/Clear on Match (CCPxM<3:0> = '1001') to Compare/Set on Match (CCPxM<3:0> = '1000'). This may occur as a result of any operation that selectively clears bit CCPxM0, such as a BCF instruction.

When this condition occurs, the output becomes inverted when the instruction is executed. It will remain inverted for all following Compare operations, until the module is reset.

PWM Mode

In Pulse Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Set-Up For Pwm Operation

The following steps should be taken when configuring
the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the TRISC<2> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

SYNCHRONOUS SERIAL PORT (SSP) MODULE

The Synchronous Serial Port (SSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be Serial EEPROMs, shift registers, display drivers, A/D converters, etc. The SSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I2C)

ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The analog-to-digital (A/D) converter module has five inputs for the PIC16F72. The A/D allows conversion of an analog input signal to a corresponding 8-bit digital number. The output of the sample and hold is the input into the converter, which generates the result via successive approximation. The analog reference voltage is software selectable to either the device's positive supply voltage (VDD) or the voltage level on the RA3/AN3/VREF pin.

The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode. To operate in SLEEP, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

The A/D module has three registers:

- A/D Result Register ADRES
- A/D Control Register 0 ADCON0
- A/D Control Register 1 ADCON1

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off and any conversion is aborted. The ADCON0 register, shown in Register 10-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 10-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be a voltage reference) or a digital I/O.

Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 9.0 TAD per 8-bit conversion. The source of the A/D conversion clock is software selectable. The four possible options for TAD are:

- 2TOSC
- 8TOSC
- 32TOSC
- Internal RC oscillator (2 - 6 μs)

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time as small as possible, but no less than 1.6 μs and not greater than 6.4 μs.

A/D Operation during SLEEP

The A/D module can operate during SLEEP mode. This requires that the A/D clock source be set to RC (ADCS1:ADCS0 = 11). When the RC clock source is selected, the A/D module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed, which eliminates all digital switching noise from the conversion. When the conversion is completed, the GO/DONE bit will be cleared, and the result loaded into the ADRES register. If the A/D interrupt is enabled, the device will wake-up from SLEEP. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set. When the A/D clock source is another clock option (not RC), a SLEEP instruction will cause the present conversion to be aborted and the A/D module to be turned off, though the ADON bit will remain set. Turning off the A/D places the A/D module in its lowest current consumption state.

Effects of a RESET

A device RESET forces all registers to their RESET state. The A/D module is disabled and any conversion in progress is aborted. All A/D input pins are configured as analog inputs. The ADRES register will contain unknown data after a Power-on Reset.

INTERRUPTS

The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits (GIE). Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit or GIE bit.

4.3 VSD CIRCUIT CONFIGURATION

The circuit consists of the PIC16F72 microcontroller, driver IC, IGBT Inverter Bridge and the single phase induction motor coupled to pump. The driver IC converts the PWM signals generated by the microcontroller to a level with which the IGBTs can operate with. The circuit also consists of the current and temperature sensors which periodically monitor the inverter circuit for thermal and over-current protection. The main advantage of this circuit is that it can also be used for 3 phase induction motor by simply changing the program on the firmware.

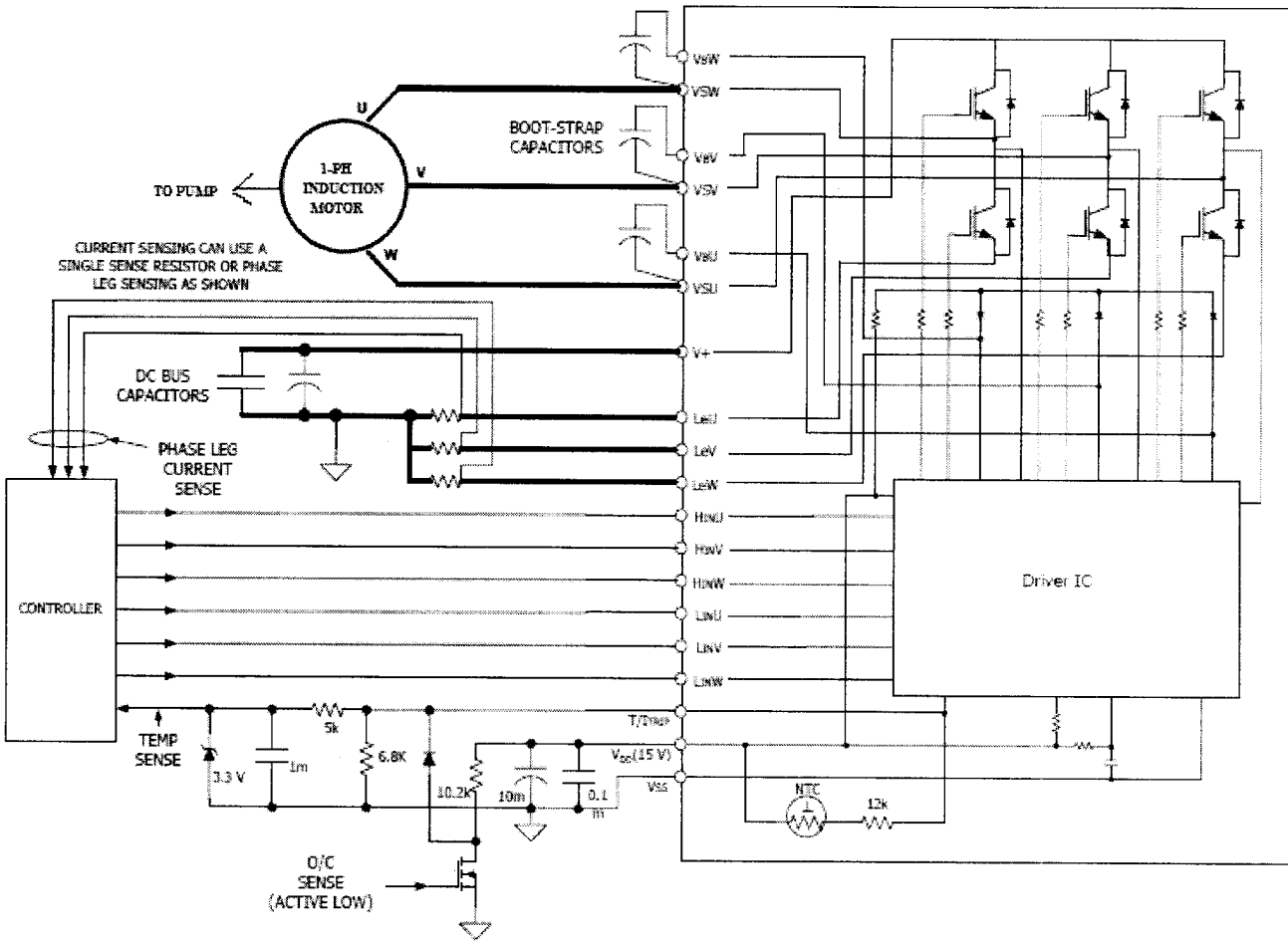

Fig.13. Complete Circuit Diagram of VSD

CHAPTER 5

SOFTWARE

5. SOFTWARE

```
;This program is used to control the speed of a Permanent Series Capacitor(PSC) motor
;3 phase control of PSC motor without the starting capacitor
;Microcontroller used: PIC16F72
;-------------------------------------------------------------------------------------
;Operation:
;1) Connect motor terminals to J2 with following order
;Connect J2/1(M1) to Main winding on the motor
;Connect J2/2(M2) to Auxiliary winding on the motor
;Connect J2/3(M3) to Common point of Main and auxiliary winding on the motor
;2) Connect AC power input power to J1
;3) Power up the board
;4) Press switch S2,
;5) Turn potentiometer R5 to vary motor speed.
;6) Set potentiometer R21 to about half way. Vary this for different motor ratings.
;(On final production board, replace with appropriate resistor)
;7)Code monitors the over current and if it exceeds for more than
;'HARDWARE_OC_COUNT' times, fault is generated and motor will stop. Reset the
;system and start from #3
;8) Software over current and heat sink over temperatures will set flag, if occurred.
;-------------------------------------------------------------------------------------
#define PIC16F72
#define PSC_MOTOR_CONTROL
;#define THREE_PHASE_MOTOR_CONTROL
;-------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------
        INCLUDE             <MC_16F72.INC>
;-------------------------------------------------------------------------------------
;Configuration bits definition
;Oscillator          : HS
;Watchdog timer      : off
```

```
;Power up timer        : on
;Brown out detect      : on
;Code protect  :off
;         __CONFIG 0x2007, 0x5A
;-------------------------------------------------------------------------------------------
;Macro
MULT   MACRO  BIT      ;MACRO FOR  UNSIGNED MULTIPLICATION
       btfsc  NO_1_LSB,BIT
       addwf  RESULT_MSB,F
       RRF    RESULT_MSB,F
       RRF    RESULT_LSB,F
       ENDM               ;END OF MACRO FOR MULTIPLICATION
;-------------------------------------------------------------------------------------------
STARTUP    CODE 0X00  ;RESET VECTOR ADDRESS
       goto   START

       CODE 0X04          ;INTERRUPT VECTOR LOCATION
       goto   ISR_INT     ;goto INTERRUPT SERVICE ROUTINE
;*************************************************************
PROG CODE
START
;*************************************************************
;INITIALIZATION OF THE PORTS AND TIMERS
       bsf            STATUS,RP0
       movlw 0X03
       movwf          TRISB          ;RB0-1 AND CONFIGURED AS INPUT
       movlw 0X00
       movwf TRISC          ;RC0-RC7 CONFIGURED AS OUTPUT
       bcf            STATUS,RP0
       movlw b'00101010'   ;Turn off PWM1,3,5 PWMs(active low) at the  beginning
       of the cycle
```

```
        movwf PWM_PORT  ;& turn on PWM0,2,4
        clrf    PWM_PR_CH1_Buff
        clrf    PWM_PR_CH2_Buff
        clrf    PWM_PR_CH3_Buff
        clrf    PWM1_DS_Buff
        clrf    PWM2_DS_Buff
        clrf    PWM3_DS_Buff
        clrf    PWM4_DS_Buff
        clrf    FLAGS                   ;CLEAR ALL FLAGS
        clrf    FLAGS2                  ;CLEAR ALL FLAGS
                call    STOP_MOTOR      ;STOP MOTOR
        call    COPY_TABLE_TO_RAM  ;COPY SINE TABLE FROM PROGRAM MEMORY TO RAM FOR FASTER ACCESS
;*****************************************************************
;INITIALIZE ADC REGISTERS
;*****************************************************************
        bcf             STATUS,RP0
        movlw 0X81
        movwf ADCON0            ;CONFIGURE FOR 32TOSC AND CHANNEL FOR CONVERSION - RA0 (FREQUNENCY)
        bsf             STATUS,RP0
        movlw 0x02              ;CONFIGURE RA0-RA4 AS ANALOG INPUT
        movwf ADCON1
        movlw 0X07              ;RA0-RA3 INPUT and RA4-RA7 OUTPUT
        movwf TRISA
        bcf STATUS,RP0
;*****************************************************************
;TIMER1 INITIALIZATION WITH PRESCALER - USED FOR SETTING THE PWM TIMING
;*****************************************************************
        movlw b'00100001'   ;LOAD THE T1CON WITH CONTROL WORD
```

```
        movwf T1CON         ;FOR TMR1 ON AND PRESCALAR IS 1:4, INTERNAL
        CLOCK
;*****************************************************************
;TIMER0    INITIALIZATION    WITH    PRESCALER    -    USED    FOR
ACCELERATION,DECELERATION AND ADC TRIGGER FOR FREQ. CONV.
;*****************************************************************
        bsf             STATUS,RP0
        movlw b'10000110'   ;prescale 1:128
        movwf OPTION_REG
        bcf             STATUS,RP0
        clrf    INTCON      ;DISABLE    ALL    INTERRUPTS    AND    FLAGS
ASSOCIATED WITH
        clrf    PIR1        ;DISABLE ALL INTERRUPT FLAGS
        bsf     STATUS,RP0
        clrf    PIE1        ;DISABLE ALL INTERRUPTS
        movlw 0X03          ;SET #POR AND #BOR FLAGS
        movwf PCON
        bcf     STATUS,RP0
;---------
WAIT_HERE
        btfss   KEY_PORT,FWD_REV_KEY
        goto    WAIT_HERE
;*****************************************************************
        call    BIG_DELAY
;Turning on this bit will turn on DC bus to the IRAMS
        bsf     PORT_RELAY,RELAY_BIT        ;       PORTB,7
        call    BIG_DELAY
;*****************************************************************
        call    RUN_MOTOR_AGAIN
        movlw 0x30
        movwf NEW_FREQ
```

```
        call    UPDATE_PWM_DUTYCYCLES           ;YES, UPDATE THE PWM
        DUTY CYCLE WITH NEW VALUE
        call    PRIORATIZE_PWMS
        call    UPDATE_TABLE_OFFSET             ;UPDATE 3 OFFSETS
        call    CALCULATE_NEW_SPEED
        bcf     STATUS,RP0
        bsf     INTCON,INTE                     ;ENABLE RB PORT
        CHANGE INTERRUPT FOR RB4
        bsf     INTCON,PEIE                     ;PERIPHERAL
        INTERRUPTS ENABLE
        bsf     INTCON,GIE                      ;GLOBAL INTERRUPT
        ENABLE

;*****************************************************************
;MAIN LOOP, THE PROGRAM WILL BE LOOPING
;*****************************************************************
MAIN_LOOP
        btfss   FLAGS,MOTOR_FREQ_COUNTER
        goto    BYPASS
        bcf     FLAGS,MOTOR_FREQ_COUNTER        ;CLEAR TMR1 OV FLAG
        call    UPDATE_PWM_DUTYCYCLES   ;YES, UPDATE THE PWM DUTY
        CYCLE WITH NEW VALUE
        call    PRIORATIZE_PWMS
        call    UPDATE_TABLE_OFFSET     ;UPDATE 3 OFFSETS
        call    CALCULATE_NEW_SPEED

        btfss   ADCON0, GO              ;If AD Conversion is complete
        bsf             ADCON0, GO      ;then start a new conversion
BYPASS
        btfsc   PIR1,ADIF               ;ADC CONVERSION COMPLETE
        INTERRUPT?
```

```
        call    AD_CONV_COMPLETE    ;YES - CONVERSON RESULT 'F -
        MOTOR FREQUENCY'
        call    KEY_CHECK           ;Check keys change
        call    PROCESS_KEY_PRESSED
        goto    MAIN_LOOP           ;GO BACK TO MAIN LOOP
;*****************************************************************
;INTERRUPT SERVICE ROUTINE
;*****************************************************************
ISR_INT
        movwf   W_TEMP              ;COPY W TO A TEMPORARY
        REGISTER
        swapf   STATUS,W            ;SWAP STATUS NIBBLES AND PLACE
        INTO W REGISTER
        movwf   STATUS_TEMP         ;SAVE STATUS TO A TEMPORARY
        REGISTER IN BANK0
        bcf     STATUS,RP0
        btfsc   PIR1,TMR1IF         ;TIMER1 OVERFLOW INTERRUPT?
        goto    TIMER1_OVERFLOW     ;YES-INTERRUPT FOR UPDATING
TMR1 REGISTERS BASED ON POT SETTING
        btfsc   INTCON,T0IF         ;TIMER0 OVERFLOW INTERRUPT?
        goto    TIMER0_OVERFLOW     ;YES - INTERRUPT FOR UPDATING
TMR1 REGISTERS BASED ON POT SETTING
        btfsc   INTCON,INTF         ;RB INTERRUPT?
        goto    CHECK_FAULT         ;YES - OVER CURRENT FAULT
POPUP                               ;RETRIEVE SAVED WREG AND STAUS
REGISTER VALUES
        swapf STATUS_TEMP,W         ;SWAP ORIGINAL STATUS REGISTER
VALUE INTO W (RESTORES ORIGINAL BANK)
        movwf STATUS                ;RESTORE STATUS REGISTER FROM
W REGISTER
```

```
        swapf W_TEMP,F                  ;SWAP    W_TEMP    NIBBLES    AND
RETURN VALUE TO W_TEMP
        swapf W_TEMP,W                  ;SWAP W_TEMP TO W TO RESTORE
ORIGINAL W VALUE WITHOUT AFFECTING STATUS
        RETFIE                          ;RETURN FROM INTERRUPT
;*****************************************************************
TIMER1_OVERFLOW                         ;TMR1 OVERFLOW ISR
        bcf             PIR1,TMR1IF     ;CLEAR TMR1IF
;---
        btfss   FLAGS1,PWM_FIRST
        goto    TAKE_CARE_PWM_SECOND
        btfsc   PWM_PR_CH1_Buff,0
        bsf             PWM_PORT,PWM1_PIN   ;PWM1 is active low
        btfsc   PWM_PR_CH1_Buff,1
        bsf             PWM_PORT,PWM3_PIN   ;PWM3 is active low
        btfsc   PWM_PR_CH1_Buff,2
        bsf             PWM_PORT,PWM5_PIN   ;PWM5 is active low
        btfsc   PWM_PR_CH1_Buff,0
        bcf             PWM_PORT,PWM0_PIN   ;PWM0 is active low
        btfsc   PWM_PR_CH1_Buff,1
        bcf             PWM_PORT,PWM2_PIN   ;PWM2 is active low
        btfsc   PWM_PR_CH1_Buff,2
        bcf             PWM_PORT,PWM4_PIN   ;PWM4 is active low
        bcf             FLAGS1,PWM_FIRST
        movlw 0xFF
        movwf TMR1H
        bsf             FLAGS1,PWM_SECOND
        comf    PWM2_DS_Buff,W
        movwf TMR1L
        btfss   STATUS,Z
        goto    POPUP
```

```
;---
TAKE_CARE_PWM_SECOND
        btfss   FLAGS1,PWM_SECOND
        goto    TAKE_CARE_PWM_THIRD

        btfsc   PWM_PR_CH2_Buff,0
        bsf             PWM_PORT,PWM1_PIN    ;PWM1 is active low
        btfsc   PWM_PR_CH2_Buff,1
        bsf             PWM_PORT,PWM3_PIN    ;PWM3 is active low
        btfsc   PWM_PR_CH2_Buff,2
        bsf             PWM_PORT,PWM5_PIN    ;PWM5 is active low
        btfsc   PWM_PR_CH2_Buff,0
        bcf             PWM_PORT,PWM0_PIN    ;PWM0 is active low
        btfsc   PWM_PR_CH2_Buff,1
        bcf             PWM_PORT,PWM2_PIN    ;PWM2 is active low
        btfsc   PWM_PR_CH2_Buff,2
        bcf             PWM_PORT,PWM4_PIN    ;PWM4 is active low
        bcf             FLAGS1,PWM_SECOND
        movlw 0xFF
        movwf TMR1H
        bsf             FLAGS1,PWM_THIRD
        comf    PWM3_DS_Buff,W
        movwf TMR1L
        btfss   STATUS,Z
        goto    POPUP
;---
TAKE_CARE_PWM_THIRD
        btfss   FLAGS1,PWM_THIRD
        goto    TAKE_CARE_FOR_NEXT_CYCLE
        btfsc   PWM_PR_CH3_Buff,0
        bsf             PWM_PORT,PWM1_PIN    ;PWM1 is active low
```

```
        btfsc   PWM_PR_CH3_Buff,1
        bsf             PWM_PORT,PWM3_PIN    ;PWM3 is active low
        btfsc   PWM_PR_CH3_Buff,2
        bsf             PWM_PORT,PWM5_PIN    ;PWM5 is active low
        btfsc   PWM_PR_CH3_Buff,0
        bcf             PWM_PORT,PWM0_PIN    ;PWM0 is active low
        btfsc   PWM_PR_CH3_Buff,1
        bcf             PWM_PORT,PWM2_PIN    ;PWM2 is active low
        btfsc   PWM_PR_CH3_Buff,2
        bcf             PWM_PORT,PWM4_PIN    ;PWM4 is active low
        bcf             FLAGS1,PWM_THIRD
        movlw 0xFF
        movwf TMR1H
        bsf             FLAGS1,PWM_NEW_CYCLE
        comf    PWM4_DS_Buff,W
        movwf TMR1L
        btfss   STATUS,Z
        goto    POPUP
;---
TAKE_CARE_FOR_NEXT_CYCLE
        movlw b'00111111'           ;Turn off all PWMs(active low)
        movwf PWM_PORT
        comf    PWM1_DS_Buff,W
        movwf TMR1L
        movlw 0xFF
        movwf TMR1H
        bcf     FLAGS1,PWM_NEW_CYCLE
        bsf     FLAGS1,PWM_FIRST
        movlw b'00010101'           ;Turn on PWM1,3,5 PWMs(active low) at the
beginning of the cycle
        movwf PWM_PORT
```

```
        goto    POPUP
;*****************************************************************
TIMER0_OVERFLOW                                         ;TMR0 overflow ISR
        movf    MOTOR_FREQUENCY,W       ;Load the Lower byte of
SpeedCommand to TMR0L
        movwf TMR0
        bsf     FLAGS,MOTOR_FREQ_COUNTER
        bcf     INTCON,T0IF                     ;Clear T0IF
        movlw HARDWARE_OC_COUNT         ;Over current count from the
comparator
        movwf OC_COUNT
        goto    POPUP
;*****************************************************************
CHECK_FAULT
        bcf     INTCON,INTF
        decfsz  OC_COUNT,F
        goto    POPUP
        call    STOP_MOTOR                      ;STOP MOTOR
        goto    POPUP
;*****************************************************************
AD_CONV_COMPLETE                                ;ADC INTERRUPT
        bcf             PIR1,ADIF               ;ADIF FLAG IS CLEARED FOR
NEXT INTERRUPT
        btfsc   FLAGS2,FREQ_REF                 ;IS THE CONVERSION RESULT
is FREQUENCY REF?
        goto    CONV_IS_FREQ
        btfsc   FLAGS2,MOTOR_CUR                ;IS THE CONVERSION RESULT
is Motor current?
        goto    CONV_IS_MOTOR_CURRENT   ;YES - READ RESULT AS
MOTOR_CURRENT
```

```
        btfsc   FLAGS2,HEATSINK_TEMP        ;IS THE CONVERSION RESULT
is Motor current?
        goto    CONV_IS_HS_TEMP             ;YES - READ RESULT AS
MOTOR_CURRENT
        bsf             FLAGS2,FREQ_REF
        bcf             ADCON0,3
        bcf             ADCON0,4
        return
CONV_IS_FREQ
        bcf             FLAGS2,FREQ_REF
        bsf             FLAGS2,MOTOR_CUR
        bsf             ADCON0,3            ;Channel1 is selected for next
conversion
        bcf             ADCON0,4
        movf    ADRES,W                     ;READ AD CONVERSION RESULT
        movwfNEW_FREQ                       ;CHECK FOR LOWER AND UPPER
ALLOWED LIMIT OF FREQ.
        sublw   0X30                        ;MINIMUM FREQUENCY SET TO 5HZ
(SCALING FACTOR X4)
        btfss   STATUS,C                    ;IS POT SETTING FOR FREQ, MORE
THAT LOWER SET LIMIT?
        goto    CHECK_UPPER_LIMIT_FREQUENCY ;YES - NOW CHECK
UPPER LIMIT
        movlw 0X30                          ;NO - SET FREQ. TO LOWER ALLOWED
LIMIT - 5HZ (X4)
        movwfNEW_FREQ
        return
CHECK_UPPER_LIMIT_FREQUENCY
        movlw 0XD0
        subwf  NEW_FREQ,W
```

```
        btfss   STATUS,C                ;IS POT SETTING MORE THAN
ALLOWED UPPER LIMIT OF FREQ?
        return                          ;NO - RETURN FROM
INTERRUPT
        movlw 0XD0                      ;YES - SET FREQ TO UPPER ALLOWED
LIMIT - 60HZ (X4)
        movwf NEW_FREQ
        return
CONV_IS_MOTOR_CURRENT
        bsf     FLAGS2,HEATSINK_TEMP
        bcf     FLAGS2,MOTOR_CUR
        bcf     ADCON0,3        ;Channle2 is selected for next conversion
        bsf     ADCON0,4
        movf    ADRES,W         ;READ AD CONVERSION RESULT
        sublw   MAX_MOTOR_CURRENT ;Max current limit
        btfss   STATUS,C        ;IS current > SET LIMIT?
        bsf     FLAGS2,OVER_CURRENT ;Yes, Take action
        return
CONV_IS_HS_TEMP
        bsf     FLAGS2,FREQ_REF
        bcf     FLAGS2,HEATSINK_TEMP
        bcf     ADCON0,3        ;Channle0 is selected for next conversion
        bcf     ADCON0,4
        movf    ADRES,W         ;READ AD CONVERSION RESULT
        sublw   MAX_HEATSINK_TEMP ;Max temp limit
        btfsc   STATUS,C        ;IS temp > SET LIMIT?
        return
        bsf     FLAGS,OVER_TEMPARATURE  ;Yes, Take action
;       call    STOP_MOTOR
        return
;*****************************************************************
```

```
;THIS ROUTINE WILL UPDATE THE PWM DUTY CYCLE ACCORDING TO THE
;OFFSET TO THE TABLE
;THIS ROUTINE SCALES THE PWM VALUE FROM THE TABLE BASED ON THE
FREQUENCY TO KEEP V/F
;CONSTANT AND LOADS THEM IN APPROPRIATE REGISTER DEPENDING ON
SETTING
;*****************************************************************
UPDATE_PWM_DUTYCYCLES
        movlw LOW  SINE_TABLE_RAM
        movwf FSR                       ;BASE ADDRESS OF SINE TABLE IN
RAM IS LOADED TO FSR
        movf  TABLE_OFFSET1,W           ;TABLE_OFFSET1 IS COPIED TO WREG
        addwf FSR,F                     ;ADRESS TO BE READ=SINE TABLE
BASE ADRESS + TABLE_OFFSET1
        BANKISEL  SINE_TABLE_RAM
        movf  INDF,W                    ;COPY SINE TABLE VALUE, POINTED
BY FSR, TO WREG
        btfsc  STATUS,Z                 ;CHECK IS VALUE READ ZERO?
        goto   PWM1_IS_0                ;YES, goto PWM1_IS_0
        movwf NO_1_LSB                  ;NO, SINE TABEL VALUE X SET_FREQ
TO SCALE TABLE VALUE BASED ON FREQUENCY SETTING
        call   MUL_8X8                  ;CALL ROUTINE FOR UNSIGNED 8x8
BIT MULTIPLICATION
        movf  RESULT_MSB,W              ;8 MSB OF 16 BIT RESULT IS STORED
        movwf TEMP_LOC                  ;AT TEMP_LOC - THIS REPRESENT
PWM DUTY CYCLE VALUE FOR PHASE 1
        goto   UPDATE_PWM2              ;GO FOR UPDATING PWM DUTY
CYCLE FOR 2ND PHASE
PWM1_IS_0
        clrf   TEMP_LOC                 ;CLEAR PWM DUTY CYCLE VALUE
FOR PHASE 1
```

```
UPDATE_PWM2
        movlw LOW  (SINE_TABLE_RAM)
        movwf FSR                       ;BASE ADDRESS OF SINE TABLE IN
RAM IS LOADED TO FSR
        movf  TABLE_OFFSET2,W           ;TABLE_OFFSET2 IS COPIED TO WREG
        addwf FSR,F                     ;ADRESS TO BE READ=SINE TABLE
BASE ADRESS + TABLE_OFFSET2
        BANKISEL   SINE_TABLE_RAM
        movf  INDF,W                    ;COPY SINE TABLE VALUE, POINTED
BY FSR, TO WREG
        btfsc  STATUS,Z                 ;CHECK IS VALUE READ ZERO?
        goto   PWM2_IS_0                ;YES, goto PWM2_IS_0
        movwf NO_1_LSB                  ;NO, SINE TABEL VALUE X SET_FREQ
TO SCALE TABLE VALUE BASED ON FREQUENCY SETTING
        call   MUL_8X8                  ;CALL ROUTINE FOR UNSIGNED 8x8
BIT MULTIPLICATION
        movf  RESULT_MSB,W              ;8 MSB OF 16 BIT RESULT IS STORED
        movwf TEMP_LOC_1                ;AT TEMP_LOC_1 - THIS REPRESENT
PWM DUTY CYCLE VALUE FOR PHASE 2
        goto   UPDATE_PWM3              ;GO FOR UPDATING PWM DUTY
CYCLE FOR 3RD PHASE
PWM2_IS_0
        clrf   TEMP_LOC_1               ;CLEAR PWM DUTY CYCLE VALUE
FOR PHASE 2
UPDATE_PWM3
        movlw LOW  SINE_TABLE_RAM
        movwf FSR                       ;BASE ADDRESS OF SINE TABLE IN
RAM IS LOADED TO FSR
        BANKSEL    TABLE_OFFSET3
        movf  TABLE_OFFSET3,W           ;TABLE_OFFSET3 IS COPIED TO WREG
```

```
        addwf FSR,F                     ;ADRESS TO BE READ=SINE TABLE
BASE ADRESS + TABLE_OFFSET3
        BANKISEL   SINE_TABLE_RAM
        movf   INDF,W                   ;COPY SINE TABLE VALUE, POINTED
BY FSR, TO WREG
        btfsc   STATUS,Z                ;CHECK IS VALUE READ ZERO?
        goto    PWM3_IS_0               ;YES, goto PWM3_IS_0
        movwf NO_1_LSB                  ;NO, SINE TABEL VALUE X SET_FREQ
TO SCALE TABLE VALUE BASED ON FREQUENCY SETTING
        call    MUL_8X8                 ;CALL ROUTINE FOR UNSIGNED 8x8
BIT MULTIPLICATION
        movf   RESULT_MSB,W             ;8 MSB OF 16 BIT RESULT IS STORED
        movwf TEMP_LOC_2                ;AT TEMP_LOC_2 - THIS REPRESENT
PWM DUTY CYCLE VALUE FOR PHASE 3
        goto    SET_PWM12               ;GO FOR CHECKING DIRECTION OF
MOTOR ROTATION REEQUIRED
PWM3_IS_0
        clrf    TEMP_LOC_2              ;CLEAR PWM DUTY CYCLE VALUE
FOR PHASE 3
SET_PWM12
        movf   TEMP_LOC,W
        movwf PWM1_DS_Buff
        movf   TEMP_LOC_1,W             ;CCPR1L AND CCPR2L RESPECTIVELY
        movwf PWM2_DS_Buff
        movf   TEMP_LOC_2,W
        movwf PWM3_DS_Buff
        RETURN

;*****************************************************************;
THIS ROUTINE UPDATES THE OFFSET POINTERS TO THE TABLE AFTER
EVERY ACCESS
```

```
;*****************************************************************
UPDATE_TABLE_OFFSET
        bcf             STATUS,RP0
        btfss   FLAGS,OFFSET1_FLAG          ;IF SET INCR. ON TABLE
        goto    DECREMENT_OFFSET1
        movlw (SINE_TABLE_ENTRIES-1)        ;CHECK FOR THE LAST VALUE
ON THE TABLE
        SUBWF       TABLE_OFFSET1,W
        btfsc   STATUS,C
        goto    CLEAR_OFFSET1_FLAG
        INCF    TABLE_OFFSET1,F             ;INCREMENT OFFSET1
        goto    UPDATE_OFFSET2
CLEAR_OFFSET1_FLAG
        bcf     FLAGS,OFFSET1_FLAG
DECREMENT_OFFSET1
        DECFSZ      TABLE_OFFSET1,F         ;DECREMENT OFFSET1
        goto    UPDATE_OFFSET2
        bsf             FLAGS,OFFSET1_FLAG
UPDATE_OFFSET2
        btfss   FLAGS,OFFSET2_FLAG          ;IF SET INCR. ON TABLE
        goto    DECREMENT_OFFSET2
        movlw (SINE_TABLE_ENTRIES-1)        ;CHECK FOR THE LAST VALUE
ON THE TABLE
        SUBWF       TABLE_OFFSET2,W
        btfsc   STATUS,C
        goto    CLEAR_OFFSET2_FLAG
        INCF    TABLE_OFFSET2,F             ;INCREMENT OFFSET2
        goto    UPDATE_OFFSET3
CLEAR_OFFSET2_FLAG
        bcf     FLAGS,OFFSET2_FLAG
DECREMENT_OFFSET2
```

```
        DECFSZ      TABLE_OFFSET2,F         ;DECREMENT OFFSET2
        goto    UPDATE_OFFSET3
        bsf     FLAGS,OFFSET2_FLAG
UPDATE_OFFSET3
        btfss   FLAGS,OFFSET3_FLAG          ;IF SET INCR. ON TABLE
        goto    DECREMENT_OFFSET3
        movlw (SINE_TABLE_ENTRIES-1)        ;CHECK FOR THE LAST VALUE
ON THE TABLE
        SUBWF       TABLE_OFFSET3,W
        btfsc   STATUS,C
        goto    CLEAR_OFFSET3_FLAG
        INCF    TABLE_OFFSET3,F             ;INCREMENT OFFSET3
        RETURN
CLEAR_OFFSET3_FLAG
        bcf     FLAGS,OFFSET3_FLAG
DECREMENT_OFFSET3
        DECFSZ      TABLE_OFFSET3,F         ;DECREMENT OFFSET3
        RETURN
        bsf     FLAGS,OFFSET3_FLAG
        RETURN
;*****************************************************************************
CALCULATE_NEW_SPEED
        movf    NEW_FREQ,W
        movwf MOTOR_FREQUENCY
        movlw 0x22
        addwf   MOTOR_FREQUENCY,F
        bcf             STATUS,C
        rrf             MOTOR_FREQUENCY,F
        bcf             STATUS,C
        rrf             MOTOR_FREQUENCY,F
        movlw 0xB5
```

```
        addwf  MOTOR_FREQUENCY,F
        return
;*****************************************************************************
;PWMs are arranged from low duty cycle to high duty cycle
;Original duty cycles are input to routine on PWMx_DS
;Output on PWM_PRx and PWM_PR_CHx
PRIORATIZE_PWMS
        movlw 0x1
        movwf PWM_PR_CH1_Buff
        movlw 0x2
        movwf PWM_PR_CH2_Buff
        movlw 0x4
        movwf PWM_PR_CH3_Buff

        movf   PWM1_DS_Buff,W
        subwf  PWM2_DS_Buff,W
        btfss  STATUS,C
        goto   INTCHG_1_2
        goto   CHECK_2_3
INTCHG_1_2
;interchange duty cycles in 1 and 2
        movf   PWM1_DS_Buff,W
        movwf TEMP_LOC
        movf   PWM2_DS_Buff,W
        movwf PWM1_DS_Buff
        movf   TEMP_LOC,W
        movwf PWM2_DS_Buff
;interchange the PWM outputs
        movf   PWM_PR_CH1_Buff,W
        movwf TEMP_LOC
        movf   PWM_PR_CH2_Buff,W
```

```
        movwf PWM_PR_CH1_Buff
        movf  TEMP_LOC,W
        movwf PWM_PR_CH2_Buff
CHECK_2_3
;interchange duty cycles in 2 and 3
        movf  PWM2_DS_Buff,W
        subwf PWM3_DS_Buff,W
        btfss STATUS,C
        goto  INTCHG_2_3
        goto  CHECK_1_2_AGN
INTCHG_2_3
        movf  PWM2_DS_Buff,W
        movwf TEMP_LOC
        movf  PWM3_DS_Buff,W
        movwf PWM2_DS_Buff
        movf  TEMP_LOC,W
        movwf PWM3_DS_Buff
;interchange the PWM outputs pins
        movf  PWM_PR_CH2_Buff,W
        movwf TEMP_LOC
        movf  PWM_PR_CH3_Buff,W
        movwf PWM_PR_CH2_Buff
        movf  TEMP_LOC,W
        movwf PWM_PR_CH3_Buff
CHECK_1_2_AGN
;interchange duty cycles in 1 and 2
        movf  PWM1_DS_Buff,W
        subwf PWM2_DS_Buff,W
        btfss STATUS,C
        goto  INTCHG_1_2_AGN
        goto  CALCULATE_TIMER_RELOAD
```

```
INTCHG_1_2_AGN
        movf   PWM1_DS_Buff,W
        movwf TEMP_LOC
        movf   PWM2_DS_Buff,W
        movwf PWM1_DS_Buff
        movf   TEMP_LOC,W
        movwf PWM2_DS_Buff
;interchange the PWM outputs pins
        movf   PWM_PR_CH1_Buff,W
        movwf TEMP_LOC
        movf   PWM_PR_CH2_Buff,W
        movwf PWM_PR_CH1_Buff
        movf   TEMP_LOC,W
        movwf PWM_PR_CH2_Buff
CALCULATE_TIMER_RELOAD
        movlw 0x9C                ;Corresponding to 8KHz count with Timer1
prescale 1:4
        movwf PWM4_DS_Buff
        movf   PWM3_DS_Buff,W
        subwf  PWM4_DS_Buff,F
        movf   PWM2_DS_Buff,W
        subwf  PWM3_DS_Buff,F
        movf   PWM1_DS_Buff,W
        subwf  PWM2_DS_Buff,F
        RETURN


;*****************************************************************************
;This routine checks for the keys status. 2 keys are checked, Run/Stop and
;Forward(FWD)/Reverse(REV)
;*****************************************************************************
```

```
KEY_CHECK
        btfss   KEY_PORT,RUN_STOP_KEY                   ;Is     key     pressed
"RUN/STOP"?
        goto    CHECK_FWD_REV_KEY
        btfsc   FLAGS1,DEBOUNCE
        return
        call    KEY_DEBOUNCE
        btfss   FLAGS1,DEBOUNCE
        return
        bsf             FLAGS1,KEY_RS
        return
CHECK_FWD_REV_KEY
        btfss   KEY_PORT,FWD_REV_KEY                    ;Is     key     pressed
"RUN/STOP"?
        goto    SET_KEYS
        btfsc   FLAGS1,DEBOUNCE
        return
        call    KEY_DEBOUNCE
        btfss   FLAGS1,DEBOUNCE
        return
        bsf             FLAGS1,KEY_FR
        return
SET_KEYS
        btfss   FLAGS1,DEBOUNCE
        return
        bcf             FLAGS1,DEBOUNCE
        bsf             FLAGS1,KEY_PRESSED
        btfss   FLAGS1,KEY_RS
        goto    ITS_FWD_REV
        btfss   FLAGS1,RUN_STOP                 ;Toggle RUN_STOP bit
        goto    $+3
```

```
        bcf             FLAGS1,RUN_STOP
        return
        bsf             FLAGS1,RUN_STOP
        return
ITS_FWD_REV
        btfss   FLAGS1,FWD_REV                  ;Toggle RUN_STOP bit
        goto    $+3
        bcf             FLAGS1,FWD_REV
        return
        bsf             FLAGS1,FWD_REV
        return

;*****************************************************************************
KEY_DEBOUNCE
        decfsz  DEBOUNCE_COUNTER,F              ;Key debounce time checked
        return
        bsf             FLAGS1,DEBOUNCE
        movlw DEBOUNCE_COUNT
        movwf DEBOUNCE_COUNTER
        return
;*****************************************************************************
;This routine takes action for the keys pressed
;If SW1(RUN/STOP) Key is pressed, the state toggles between RUN and STOP
;If SW2(FWD/REV) Key is pressed, the state toggles between FORWARD and
REVERSE
PROCESS_KEY_PRESSED
        btfss   FLAGS1,KEY_PRESSED              ;Is there a key press waiting?
        return
        btfss   FLAGS1,KEY_RS                   ;Is it RUN/STOP?
        goto    CHECK_FWD_REV
```

```
        btfss   FLAGS1,RUN_STOP                 ;Yes,Was the previous state a
Stop?
        goto    STOP_MOTOR_NOW
        call    RUN_MOTOR_AGAIN                 ;Yes, Then RUN the motor
        bcf             FLAGS1,KEY_PRESSED      ;Clear the Flag
        bcf             FLAGS1,KEY_RS
;       bsf             LED_PORT,RUN_STOP_LED   ;Turn on LED to indicate
motor running
        return

STOP_MOTOR_NOW
        call    STOP_MOTOR                      ;Was the previous state a
RUN?, Then stop the motor
        bcf             FLAGS1,KEY_PRESSED
        bcf             FLAGS1,KEY_RS
;       bcf             LED_PORT,RUN_STOP_LED   ;Clear Flags and indicate
motor stopped on LED
        return
CHECK_FWD_REV
        btfss   FLAGS1,KEY_FR                   ;Is the Key pressed =
FWD/REV?
        return
;       btg             LED_PORT,FWD_REV_LED    ;Yes,
;       bcf             LED_PORT,RUN_STOP_LED
        call    STOP_MOTOR                      ;Stop the motor before
reversing
        call    BIG_DELAY                       ;Delay between reversing the
direction
        call    BIG_DELAY
        call    RUN_MOTOR_AGAIN                 ;Run the motor in Reverse
direction
```

```
        bcf             FLAGS1,KEY_PRESSED          ;Clear Flags
        bcf             FLAGS1,KEY_FR
;       bsf             LED_PORT,RUN_STOP_LED
        return
;*****************************************************************;THIS ROUTINE STOPS THE MOTOR BY DRIVING THE PWMS TO 0% DUTY CYCLE. ALSO DISABLE
;SELECT INTERRUPT TO MAINTAIN STOP CONDITION OF MOTOR
;*****************************************************************
STOP_MOTOR
        movlw b'00111111'   ;Turn off PWM1,3,5 PWMs(active low) at the beginning of the cycle
        movwf PWM_PORT  ;& turn on PWM0,2,4
        bsf             STATUS,RP0
        bcf             PIE1,TMR1IE
        bcf             PIE1,ADIE
        bcf             STATUS,RP0
        bcf             INTCON,T0IE
        bcf             FLAGS,MOTOR_FREQ_COUNTER
        clrf    TABLE_OFFSET1
        clrf    TABLE_OFFSET2
        clrf    TABLE_OFFSET3
;       bcf             PORT_RELAY,RELAY_BIT
        RETURN
;*****************************************************************
;THIS ROUTINE STARTS MOTOR FROM PREVIOUS STOP WITH MOTOR PARAMETERS INITIALIZED
;*****************************************************************
RUN_MOTOR_AGAIN
        call    INIT_MOTOR_PARAMETERS
        bsf             STATUS,RP0
```

```
        bsf             PIE1,TMR1IE
        bcf             STATUS,RP0
        bsf             INTCON,T0IE
        RETURN
;*****************************************************************
;THIS ROUTINE INITIALIZES THE PARAMETERS REQUIRED FOR MOTOR INITIALIZATION.
;*****************************************************************
INIT_MOTOR_PARAMETERS
#ifdef PSC_MOTOR_CONTROL
        movlw 0X012                     ;INITIALIZE THE TABLE OFFSET TO 3 REGISTERS
        movwf TABLE_OFFSET1             ;TO FORM 0-120-240 DEGREES
        movlw 0X00
        movwf TABLE_OFFSET2
        movlw 0X08
        movwf TABLE_OFFSET3

        bcf             FLAGS,OFFSET1_FLAG      ;OFFSET FLAGS INITIALIZATION
        bsf             FLAGS,OFFSET2_FLAG
        btfsc   FLAGS1,FWD_REV
        goto    INIT_MOT_REV
        bsf             FLAGS,OFFSET3_FLAG      ;Offset3 initialized
        goto    INIT_MOT_FREQ
INIT_MOT_REV
        bcf             FLAGS,OFFSET3_FLAG
#endif
#ifdef THREE_PHASE_MOTOR_CONTROL
        movlw 0x09              ;Initialize the table offset to 3 registers
        movwf TABLE_OFFSET1             ;to form 0-120-240 degrees
```

```
        bsf     FLAGS,OFFSET1_FLAG    ;Offset flags initialization

        btfsc   FLAGS1,FWD_REV
        goto    INIT_MOT_REV

        movlw 0x03
        movwf TABLE_OFFSET2
        bcf     FLAGS,OFFSET2_FLAG

        movlw 0x0F
        movwf TABLE_OFFSET3
        bcf     FLAGS,OFFSET3_FLAG
        goto    INIT_MOT_FREQ
INIT_MOT_REV
        movlw 0x0F
        movwf TABLE_OFFSET2
        bcf     FLAGS,OFFSET2_FLAG

        movlw 0x03
        movwf TABLE_OFFSET3
        bcf     FLAGS,OFFSET3_FLAG
#endif
INIT_MOT_FREQ
        movlw 0XB1
        movwf TMR0
        bsf             FLAGS,MOTOR_FREQ_COUNTER
        RETURN
;*****************************************************************
;ROUTINE FOR 8*8 BIT MULTIPLICAION
;*****************************************************************
MUL_8X8
```

```
        clrf   RESULT_MSB
        clrf   RESULT_LSB
        movf   NEW_FREQ,W              ;MOVE THE MULTIPLICAND TO W
REG.
        bcf    STATUS,C                ;CLEAR THE CARRY BIT IN THE
STATUS REG.
        MULT   0
        MULT   1
        MULT   2
        MULT   3
        MULT   4
        MULT   5
        MULT   6
        MULT   7
        RETLW  0
;*************************************************************************
;UPON INITIALIZATION THE SINE TABLE CONTENTS ARE COPIED TO THE
;RAM FROM  PROGRAM MEMORY
;*************************************************************************
COPY_TABLE_TO_RAM
        BANKSEL    SINE_TABLE_RAM
        BANKISEL   SINE_TABLE_RAM
        movlw LOW SINE_TABLE_RAM
        movwf FSR
        movlw 0X13
        movwf TEMP_LOC
        clrf     TEMP_LOC_1
COPY_AGAIN
        movlw HIGH  SINE_TABLE
        movwf PCLATH
        movf   TEMP_LOC_1,W
```

```
        call    SINE_TABLE
        movwf INDF
        INCF  TEMP_LOC_1,F
        INCF  FSR,F
        DECFSZ      TEMP_LOC,F
        goto   COPY_AGAIN
        movlw LOW  SINE_TABLE_RAM          ;FSR POINTS TO THE STARTING
OF THE TABLE
        movwf FSR
        RETURN
;*********************************************************************
;GENERAL PURPOSE DELAY ROUTINE - PRESENT DELAY TIME ~38msec
;*********************************************************************
DELAY
        movlw DELAY_COUNT1
        movwf TEMP_LOC
DEC_TEMP_LOC
        movlw DELAY_COUNT2
        movwf TEMP_LOC_1
DEC_TEMP_LOC_1
        DECFSZ      TEMP_LOC_1,F
        goto   DEC_TEMP_LOC_1
        DECFSZ      TEMP_LOC,F
        goto   DEC_TEMP_LOC
        RETURN
B_DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
```

```
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        return
BIG_DELAY
        call    B_DELAY
        call    B_DELAY
        call    B_DELAY
        call    B_DELAY
        return
;*********************************************************************
;EQUATION USED FOR CALCULATION OF SINE TABLE ENTRIES =
;(SIN(ANGLE)+1)*FF/2
;Sine table has value corresponding to every 10 electrical degrees
;*********************************************************************
TABLE       CODE 0X0005
SINE_TABLE
        addwf PCL,F
        RETLW       .2
        RETLW       .4
        RETLW       .8
        RETLW       .16
        RETLW       .24
        RETLW       .34
        RETLW       .50
        RETLW       .64
        RETLW       .80
        RETLW       .96
```

```
RETLW    .112
RETLW    .128
RETLW    .144
RETLW    .156
RETLW    .168
RETLW    .178
RETLW    .184
RETLW    .188
RETLW    .190

END
```

CHAPTER 6

ENERGY CONSERVATION

6. ENERGY CONSERVATION

The energy conservation achieved by the use of VSD for a pump set is illustrated by means of a C++ program. The results generated by the program for the following inputs demonstrate energy efficiency of the VSD.

INPUT

Pump efficiency: 50%

Rated power output: 0.5 hp

Nominal head of the pump: 100 feet

Number of operating heads: 4

First operating head: 25 feet

Second operating head: 50 feet

Third operating head: 75 feet

Fourth operating head: 100 feet

Rated speed of the pump: 2880 rpm

Annual pump hours: 730 hours

RESULTS

The program produces the results for 4 different speeds assuming 182.5hrs for each speed.

Operating Head	**Pump Without VSD**		**Pump With VSD**	
in feet	**Speed in rpm**	**Energy consumption in kWhr**	**Speed in rpm**	**Energy consumption in kWhr**
25	2880	136.145	1440	17.018
50	2880	136.145	2040	48.384
75	2880	136.145	2500	89.061
100	2880	136.145	2880	136.145

Table.1. Energy Consumption Comparison

Annual energy consumption without VSD =544.580017kWhr

Annual energy consumption with VSD =290.600708kWhr

Energy conserved annually =253.979309 kWhr

Percentage energy conserved = 46.64%

MODEL CALCULATION

By Affinity Laws,

$(Head_2/Head_1) = (Speed_2/Speed_1)^2$ -------- (1)

$(Power_2/Power_1)=(Speed_2/Speed_1)^3$ -------- (2)

$Head_1$=100 feet, $Head_2$= 25 feet

$Speed_1$ =2880 rpm

$Power_1$ =0.5 hp = 373 watts

From (1),

$(25/100) = (Speed_2/2880)^2$

$Speed_2$ = 1440 rpm

From (2),

$(Power_2/0.373) = (1440/2880)^3$

$Power_2$=93.25 watts

Energy consumed for 182.5 hrs = 93.25 x 182.5

=17.018 kWhr

A similar calculation is made for all the 4 operating heads.

Annual energy consumption with VSD = 17.018+48.384+89.061+136.145

= 290.600708 kWhr

Energy conserved annually = 544.580017- 290.600708

= 253.979309 kWhr

Percentage energy conserved = (253.979039 x 100) / 544.580017

= 46.64%

CHAPTER 7

CONCLUSIONS

7. CONCLUSION

Microcontroller-based control for a PSC motor makes the system easy to implement. Implementing the algorithm using a 3-phase inverter bridge gives flexibility and efficiency of control.

Variable speed, depending upon the load requirement, provides significant energy saving. A reduction of 20% in the operating speed of the motor from its rated speed will result in an almost 50% reduction in the input power to the motor. This is not possible in a system where the motor is directly connected to the supply line. In many flow control applications, a mechanical throttling device is used to limit the flow. Although this is an effective means of control, it wastes energy because of the high losses and reduces the life of the motor valve due to generated heat.

Also when the motor is operated at a minimum load (i.e., open shaft), the current drawn by the motor is primarily the magnetizing current and is almost purely inductive. As a result, the power factor is very low, typically as low as 0.1. When the load is increased, the working current begins to rise. The magnetizing current remains almost constant over the entire operating range, from no load to full load. Hence, with the increase in the load, the PF will improve. When the motor operates at a PF less than unity, the current drawn by the motor is not sinusoidal in nature. This condition degrades the power quality of the supply line and may affect performances of other utility equipment connected on the same line. All the above mentioned problems are overcome by use of a VSD.

This project can be extended for the purpose of flow control, pressure control, level control, temperature control. Automation of pumping units can also be effected by the extension of VSD.

CHAPTER 8

REFERENCES

8. REFERENCES

[1] N.Mohan, T.M.Undeland and W.P. Robbins, "Power Electronics: converter, applications and design", Media Enhanced third edition, 2002

[2] A.Merello, A.Rugginenti and M.Grasso, "Using monolithic high voltage gate drivers", Reference: DT04-4revA, www.irf.com

[3] J.Adams, "Bootstrap component selection for control ICs", Reference: DT98-2a, www.irf.com

[4] Gopal.K.Dubey , "Fundamentals Of Electrical Drives", Narosa Publications,2002

[5] PIC16F72 Data Sheet : 28-Pin, 8-Bit CMOS FLASH Microcontroller with A/D Converter, www.microchip.com

[6] ABB drives : Using Variable Speed Drives (VSDs) in pump applications, www.abb.com

[7] Bidirectional V/f control of Single and Three Phase Induction Motors using PIC16F72, Reference: AN967, www.microchip.com

[8] AC Induction Motor Fundamentals, Reference: AN887, www.microchip.com

[9] International Rectifier's IRAMS10UP60A, Integrated Power Module, Reference: PD94640, www.irf.com

CHAPTER 9

APPENDIX

9. APPENDIX

9.1 PROGRAM FOR DETERMINING ENERGY EFFICIENCY FOR A GIVEN PUMP WITH VSD

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void speed();
void power();
void save();
int i,n,j,hp,a[10],nhf;
float pe,x[10],pi,po,oh[10],ohf[10],nh,rs,y[10],wvsd,vsd=0.0,ph;
double b;
void main()
{
clrscr();
cout<<"Enter the details:";
cout<<"\nEnter pump efficiency:";
cin>>pe;
cout<<"\nEnter the rated power output(hp):";
cin>>hp;
cout<<"\nEnter nominal head of the pump(in ft):";
cin>>nhf;
nh=nhf*12*0.025;
cout<<"\nEnter no.of operating heads:";
cin>>n;
for(i=1;i<=n;i++)
{
cout<<"\nEnter operating head(ft):";
cin>>ohf[i];
oh[i]=ohf[i]*12*0.025;
}
```

```
cout<<"\nEnter the rated speed of the pump(rpm):";
cin>>rs;
cout<<"\nEnter annual pump hours";
cin>>ph;
speed();
power();
save();
getch();
}
void speed()
{
po=hp*746.0/1000;
pi=po/pe;
cout<<"\nResult:";
cout<<"\nPower input(kW):"<<pi<<"\nPower output(kW):"<<po<<endl;
cout<<"\nFeet-metre conversion";
cout<<"\nNominal head in m:"<<nh;
for(i=1;i<=n;i++)
cout<<"\nOperating head in m for "<<ohf[i]<<" feet is:"<<oh[i];
cout<<"\nSpeed calculations for various heads:"<<endl;
for(i=1;i<=n;i++)
{
b=(oh[i]/nh)*rs*rs;
x[i]=sqrt(b);
//cout<<"\nSpeed:"<<x[i];
a[i]=ceil(x[i]);
if(a[i]%10==0)
{
cout<<"\nSpeed for "<<oh[i]<<"m head "<<" is "<<a[i]<<" rpm";
}
else
```

```
{
for(j=1;(a[i]%10)!=0;j++)
a[i]++;
cout<<"\nSpeed for "<<oh[i]<<"m head "<<" is "<<a[i]<<" rpm";
}
}
}
void power()
{
cout<<"\nOutput calculations for calculated speeds:"<<endl;
for(i=1;i<=n;i++)
{
y[i]=(a[i]/rs)*(a[i]/rs)*(a[i]/rs)*po;
cout<<"\nOutput for "<<a[i]<<" rpm "<<" is "<<y[i]<<" kilowatts";
}
}
void save()
{
wvsd=ph*po/pe;
//output/efficiency=input
cout<<"\nAnnual Energy Consumption without VSD:"<<wvsd<<"kwhr";
for(i=1;i<=n;i++)
vsd+=y[i];
vsd=vsd*ph/(n*pe);
cout<<"\nAnnual Energy Consumption with VSD:"<<vsd<<"kwhr";
cout<<"\nSavings in energy:"<<(wvsd-vsd)<<" kwhr";
}
```

CHAPTER 10

PHOTOGRAPHS

10. PHOTOGRAPHS

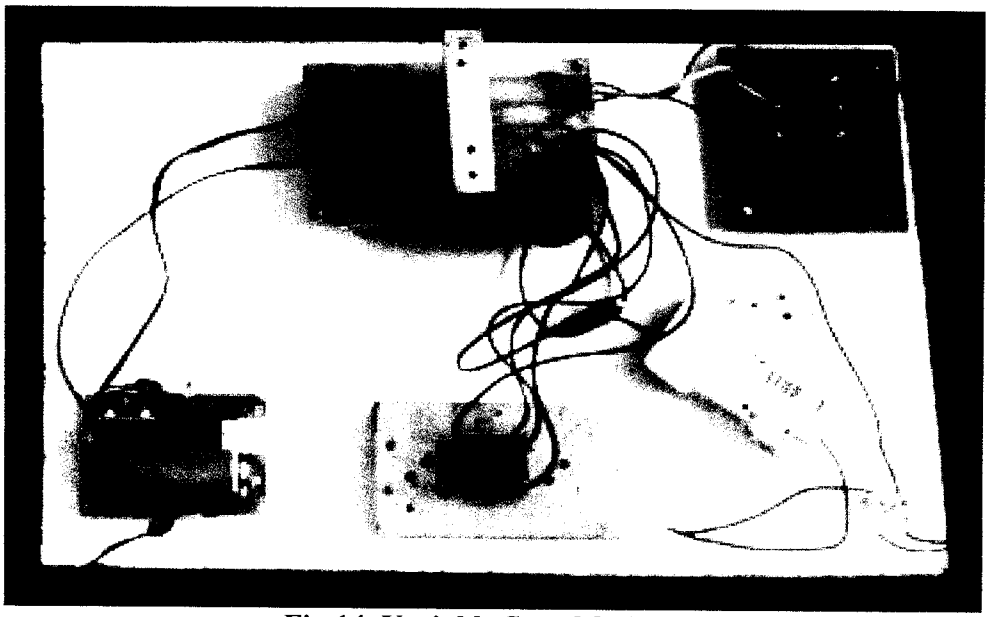

Fig.14. Variable Speed Drive Circuit

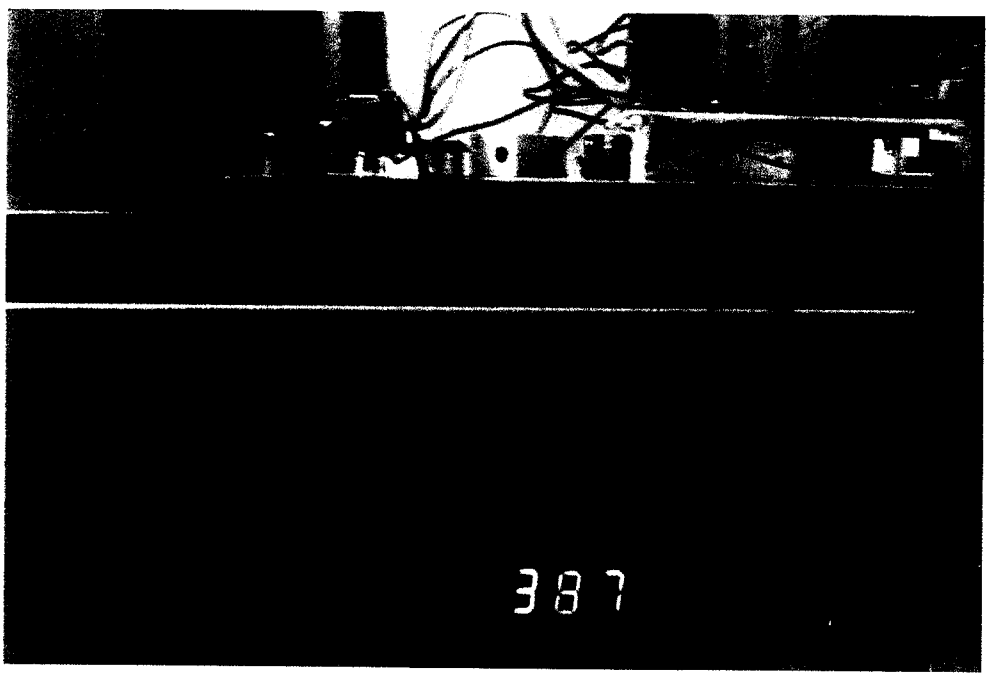

Fig.15. Variable Speed Drive Circuit with Frequency Meter