Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Model Reference Adaptive Control Based Sensorless Speed Control of Induction Motor

By:

Workagegn Tatek

Thesis Submitted To Addis Ababa Institute of Technology in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical and Computer Engineering (Control Engineering)

Advisor:

Dr. Mengesha Mamo

January 2017

Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Thesis Submitted To Addis Ababa Institute of Technology in Partial Fulfillment of the Requirement for the Degree of Master of Science in Electrical and Computer Engineering (Control Engineering)

By:

Workagegn Tatek

Approval by Board of Examiners

Dr: ____________	____________	____________
Chairman Department of Graduate Committee	Signature	Date
Dr.Mengesha Mamo	____________	____________
Advisor	Signature	Date
____________	____________	____________
Internal Examiner	Signature	Date
____________	____________	____________
External Examiner	Signature	Date

Declaration

I, the undersigned, declared that this MSc thesis is my original work, has not been presented for fulfillment of a degree in this or any other University and all sources and materials used for the thesis is acknowledged.

Workagegn Tatek	________________________
Name	Signature
Addis Ababa	__________________________
Place	Date of Submission

This thesis work has been submitted for examination with my approval as a University Advisor.

Dr. Mengesha Mamo	________________________________
Advisor's Name	Signature

Acknowledgment

It is with sincere gratitude that I thank my advisor, Dr. Mengesha Mamo, for his critical comments, persistent encouragement and advice on every of the steps during my thesis.

Beside my advisor, I would like to thank Mr. Kibru Agza , Electrical Machine Laboratory Assistant in the School of Electrical and Computer Engineering and my friend Mr.Teshome Hambessa for their kind cooperation and encouragement in the final implementation of the thesis work.

Abstract

In this thesis, stator current based model reference adaptive system (MRAS) speed estimator is used for closed loop speed control of induction motor without mechanical speed sensor. Due to high sensitivity of motor parameters variation at low speed including zero, stability analysis of MRAS design is done to correct any mismatch parameters value in the MRAS is done to estimate the motor speed at these value. As result the induction motor sensorless control can operate over a wide range including zero speed. The performance of stator current based MRAS speed estimator was analyzed in terms of speed tracking capability, torque response quickness, low speed behavior, step response of drive with speed reversal, sensitivity to motor parameter uncertainty, and speed tracking ability in regenerative mode. The system gives good performance at no load and loaded condition. Hence, it can work with different load torque conditions and with parameters variation.

Stator current based MRAS estimator sensorless speed control technique can make the hardware simple and improve the reliability of the motor without introducing feedback sensor and it becomes more important in the modern AC induction motor. The sensorless vector control operation has been verified by simulation on Matlab and experimentally using Texas Instruments HVMTRPFCKIT with TMS320 F28035 DSP piccolo control card and 0.18kw AC induction motor. From the experimental work the actual speed of the motor with maximum steady state error of 0.00458pu has been achieved.

Keywords: Induction motor, Vector control, Stator current based MRAS, Sensorless, Piccolo™ TMS320F28035 Control Card

Tables of content

List of Tables

List of Figures

List of Abbreviation

DC	Direct current
AC	Alternating current
MRAS	Model reference adaptive system
DSP	Digital signal processing
PI	Proportional controller
IM	Induction motor
FOC	Field oriented control
IFOC	Indirect field oriented control
DFOC	Direct field oriented control
SVPWM	Space vector pulse width modulation
PWM	Pulse width modulation
Back_emf	Back electromagnetic force
i_{qs_ref}	Reference q axis synchronous current
i_{ds_ref}	Reference d axis synchronous current
i_{qs}	q -axis synchronous current
i_{ds}	d -axis synchronous current
$\hat{\varphi}_r$	Estimated rotor flux
$\hat{\imath}_{s\alpha}$	Estimated stationary α stator current
$\hat{\imath}_{s\beta}$	Estimated stationary β stator current
$\hat{\varphi}_{r\alpha}$	Estimated α axis rotor flux
$\hat{\varphi}_{r\beta}$	Estimated β axis rotor flux
$\widehat{\omega}_r$	Estimated rotor speed
ISR	Subroutine interrupts
HVMTRPFCKIT	High Voltage Motor Control and PFC Kit
ADC	Analog digital convertor
PFC	Power factor corrector
GPIO	General input output

Chapter One

Introduction

1.1 Background of the study

Induction machine receives its name from the fact that electrical current is induced in the rotor by magnetic fields generated by the stator windings. It is therefore required that the rotor of an induction machine contains an electrical conducting material. In fact, if the rotor is a solid cylinder of metal the machine could operate as an induction machine, though not a very good one. This conducting material may consists of two or three phase windings similar to those on the stator, this is known as a wound rotor induction machine, but many induction machine rotors have a squirrel cage design.

Induction machines were chosen for variable speed drives, due to primarily low material and manufacturing cost and also reliable, associated with the squirrel cage induction machine is proposed by [1],[2]. As a result, induction machines are typically used in low and medium cost drive applications which required moderate performance, such as conveyor belts, fans, pumps, etc. In [3]-[5] were proposed accurate speed identification is required for all high performance vectors controlled induction motor drives. The speed identification of IM can be performed by a shaft speed encoder. However, compared to speed estimation, shaft speed encoder has several disadvantages such as an increase in cost, size, complexity, maintenance requirements and a decrease in the reliability and robustness.

Speed encoders are expensive and introduce reliability concerns for vector controlled AC motor drives. The use of this encoder implies additional electronics, extra wiring, space and careful mounting which detracts from the inherent robustness of cage induction motors been proposed in [6]. Moreover at low powers (2kW to 5 kW) the cost of the sensor is about the same as the motor. Even at 50 kW, it can still be between 20 to 30% of the machine cost. Therefore it has been great interest in the research community in developing a high performance induction motor drive that does not require a speed or position encoder for its operation. The concept of sensorless vector control has been used for estimation techniques to estimate e the speed of the rotor from motor terminal voltage and current signals. For high performance drives, vector control based systems can be used. These methods include rotor field orientation, feed forward control of stator voltages, stator flux orientation, estimation of rotor flux, torque current model, model reference adaptive system etc. Adaptive approaches are useful when machine parameters are not fully known. In this thesis the rotor speed estimation method for an AC induction motor has been done by stator current based model reference adaptive system. Due to wearing, aging, breakdown and the changes in environment while the plant operates, the parameters associated with it may undergo a change. In such case the convectional speed estimation techniques cannot satisfy the performance specification stated in [2]. For such cases the stator current based model reference adaptive system estimator will be obtained a good choice. In the stator current based MRAS speed estimator the desired performance has been given in terms of a reference model and adaptive model. Each time an error is generated by comparing the reference and adaptive output. By considering this error, using suitable algorithm the gain expressions of the adaptive controller will be obtained. The stator current based MRAS speed estimator has been designed and test through various software tools. In this thesis design and implementation of an adaptive estimator using the Matlab simulation and implementation using Texas Instruments TMS320F28035 Control Card on general purpose AC induction motor has been done.

The thesis first introduces characteristics and modeling of induction motor, then stator current based MRAS speed estimation techniques and design procedure for speed and current controller using Matlab/Simulink, finally the simulation is implemented with experimental work.

1.2 Statement of the problem

In sensorless speed control of induction motor, the rotor speed has been estimated by various techniques. The simplest method is based on the angular velocity of rotor flux vector and slip calculation. The rotor flux vector and slip calculation method are quite popular and simple to implement, but the accuracy is not very good due to the great sensitivity to motor parameter variation. Other method is based on extended Kalman filter which is more robust to the induction motor parameter changes or identification errors but much more complicated in practical realization. The solution for rotor speed estimation is based on MRAS principle, in which an error vector is formed from the outputs of reference and adaptive models, models both dependent on different motor parameters. The error is driven to zero through adjustment of the parameter that influences one of the models. The MRAS approach has advantages like

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

simplicity, easy to implement and has direct physical interpretation. So stator current based model reference adaptive system solves the problem due to aging, un-model dynamics and parameter variation even at low speed.

1.3 Objectives of the study

The general objective of the thesis is to design and implement the stator current based model reference adaptive system for speed estimation of induction motor which overcomes the problem of parameter variation at low speed, load torque variation and Problems due to mechanical speed sensor.

The Specific objectives include:

- Study the characteristics and modeling of induction motor vector control drive system.
- Developing stator current based model reference adaptive system based on the nonlinear model of the induction motor drive system.
- Develop adaptive mechanism to estimate the rotor speed of the motor which result good error tracking performance and stable system.
- Simulating the speed control and error tracking of the induction motor using vector control realized by space vector pulse width modulation which has been done on Matlab/Simulink.
- Developed control strategies and implementation model for the stator current based MRAS for sensorless speed control of induction motor.

1.4 Methodology

The implementation of a sensorless speed control of induction motor based on stator current based MRAS speed estimator is shown in Figure 1.1. In this thesis work, indirect field oriented control based on the close loop stator current based MRAS speed estimator is used for the sensorless speed control of induction motor. Due to effect of parameters variation on sensorless speed control of induction motor the reference and adaptive model is design properly. The reference model is used the induction motor itself and the adaptive model is the current and flux estimator models together. The adaptive PI control parameters are calculated by finding the transfer function of the controller and then setting the appropriate gains from the root locus plot to achieve fast adaptive loop which is independent of the load torque variation. Its main emphasis is to gain insight into robust speed control of induction motor by mathematical modeling, simulation using Matlab and practical experiment using Texas Instrument

HVMotorCtrlPFC Kit with Piccolo TMS320 F28035 control card. Closed loop experiment is carried out to show part of the performance of the scheme and compare with the simulation part. The data required for the study was identified and collected by literature survey, Matlab simulation and experiment work.

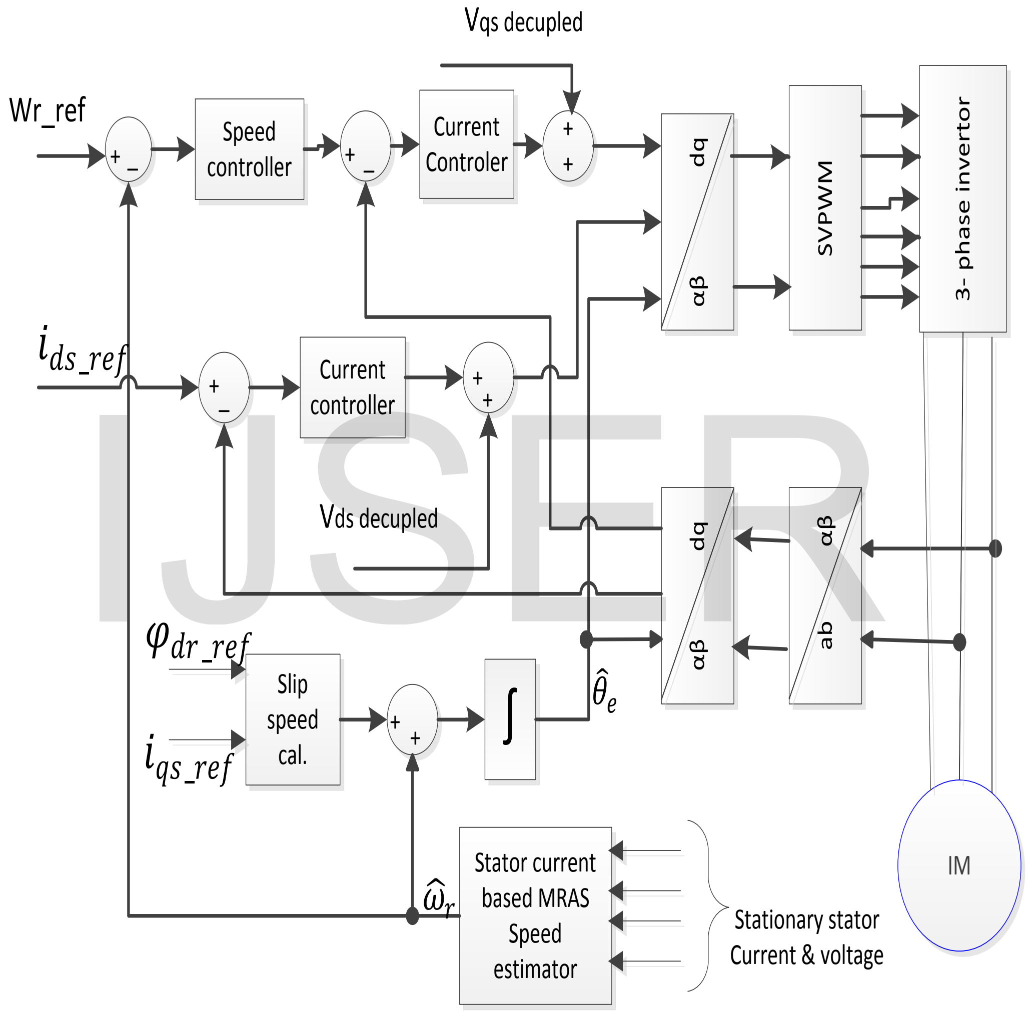


Figure 1.1 General block diagram of MRAS based sensorless speed control of induction motor

1.5 Thesis Organization

The thesis is organized into six chapters including this introduction. The rest of the thesis is organized as follows.

Chapter 2 describes the theory and operation principle of induction motor. Dynamical modeling of IM with indirect field oriented vector control based on coordinate transformation is described and sensorless vector control and its principle is presented. The speed and the current control are designed based on the robust speed analysis.

Chapter 3 in this chapter sensorless speed estimation of IM drive with indirect field oriented control based on stator current based model reference adaptive system is described. The details model and stability analysis of stator current based MRAS speed estimator is presented.

Chapter 4 this chapter discussed on simulation of the drive system on Matlab/Simulink including simulation result. Included in this chapter, simulation results based on load torque disturbance and motor parameter variation is discussed.

Chapter 5 this chapter discussed on the experimental implementation; both hardware and software requirements are discussed together with the development of control algorithms. Furthermore, closed loop experiment using Texas instrument TMS320F28035 control card on general purpose AC induction motor is presented and the output is discussed.

Chapter 6 draws the conclusion from the work done in this thesis and recommend further research possible in this area.

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

Chapter Two

Field Oriented Control of Induction Motor

Introduction

This chapter presents FOC of induction motor based on mathematical transformations of the standard three phase induction motor model into specific two phase d-q coordinate model. The three phase induction motor model is a complex matrix equation with sinusoidal functions of the rotor flux was proposed in [7]. This model has inherent control and computation disadvantages due to the ever changing rotor flux. This can be greatly improved with a simple transformation of the three phase model into a d-q model with the two coordinates rotate with respect to the rotor flux at the synchronous speed. The three phase stator current of the induction motor may be modeled as a complex space vector which can be transformed into two currents the d-coordinate stator current and the q-coordinate stator current. The d-coordinate stator current chosen to be in line with the d-axis rotor flux and the q-coordinate stator current chosen to be 90° lagging. This transformation also gives significant computational advantage in the field oriented control of induction motors. The following outlines are the sections in this chapter.

Section 2.1 presents the construction, principle of operation and three phases to two phase transformation of induction motor. The transformation is implemented in two steps. First, transform the three phase system into the two phase orthogonal coordinate system. This transformation is known as $\alpha-\beta$ transformation. Second, transform the $\alpha-\beta$ coordinates into the d-q coordinates, with d-coordinate chosen to be in line with the rotor flux and the q-coordinate chosen 90° lagging.

Section 2.2 presents vector control which provides an efficient control of the speed of induction motor which is done by controlling the motor speed and current.

Section 2.3 presents a space vector pulse width modulation (SVPWM) technique for the control of a three phase voltage source inverter (VSI) that drives the induction motor.

2.1 Induction motor

An induction or asynchronous motor is an AC electric motor in which the electric current in the rotor needed to produce torque is obtained by electromagnetic induction from the magnetic field of the stator winding. Induction refers to the field in the rotor is induced by the stator current and asynchronous refers to the fact that the rotor speed is not equal to the stator speed. No sliding contacts and permanent magnets are needed to make an induction motor work which makes it very simple and cheap to manufacture. As motors, they rugged and require very little maintenance. However, the speed is not as easily controlled as with DC motors and they draw large starting currents, and operate with a poor lagging factor when lightly loaded was proposed [8].

2.1.1 Construction of the Three Phase Induction Motor

Three phase AC induction motors are commonly used in industry applications. This type of motor has three main parts; rotor, stator and enclosure. The stator and the rotor do the work and the enclosure protects the stator and rotor. Most induction motors are rotary type with basically a stationary stator and a rotating rotor. The stator has a cylindrical magnetic core that is housed inside a metal frame. The stator magnetic core is formed by stacking thin electrical steel laminations with uniformly spaced slots stamped in the inner circumference to accommodate the three distributed stator windings. The stator windings are formed by connecting coils of copper or aluminum conductors that are insulated from the slot walls. The rotor consists of a cylindrical laminated iron core with uniformly spaced peripheral slots to accommodate the rotor windings.

There are two different types of induction motor rotor; Wound rotor induction motor has a three phase winding, similar to the stator winding. The rotor winding terminals are connected to three slip rings which turn with the rotor. The slip rings/brushes allow external resistors to be connected in series with the winding. The external resistors are mainly used during start-up under normal running conditions the windings short circuited externally. The second type of induction motor rotor is the squirrel cage rotor. It consists of series of conducting bars laid into slots carved in the face of rotor and shorted at either end by large shorting rings. In this thesis a squirrel cage rotor induction motor is used.

2.1.2 Principle of operation

The principle of operation of the induction motor is based on generating a rotating magnetic field. When three phase induction motor is connected to the appropriate AC voltage source it produces stator current and each of three coils for each phase is fed with an alternating current. The current induced in each phase generate magnetic field intensity and this cause a flow of magnetic flux. The rotating magnetic field interacts with a set of conductors arranged on the rotor and short circuited at the ends with two rings. This interaction between the magnetic field and the conductor induces a current in the bars. A current flows in the conductors of the rotor through the short circuiting rings at the end. This current in turn produces a magnetic field. There is a difference in between revolving field speed and rotor speed then the revolving field induces a voltage in the rotor winding. The difference between the rotor and the revolving field speeds is called the slip speed. The induced voltage results in a rotor current that generates a flux in the counter direction to the flux generated by the stator windings. The interaction between the rotor magnetic field and the squirrel cage bars induces torque and causes rotation.

2.1.3 Dynamic Model of Induction Motor

A dynamic model of the induction motor subjected to control must be known in order to understand and design the vector controlled drives and speed estimation technique. Such a model can be obtained by means of the two axis theory of electrical motors. Following are the assumptions made for the model:

- Each stator winding is distributed so as to produce a sinusoidal magnet motive force along air gap, i.e. space harmonics are negligible.
- The slotting in stator and rotor produces negligible variation in respective inductances.
- Mutual inductances are equal.
- The harmonics in voltages and currents are neglected.

In this thesis, a complex vector notation and some reference frame conversions are used. Since this is quite essential to the understanding of the rest of the theory, it was shortly described in the following subsection.

Three Phase Transformation

In this thesis generalized machine theory and mathematical transformations are used to decouple variables to facilitate the solution of difficult equation with time varying coefficients or to refer all variables in a common reference frame.

The most commonly used transformation is poly-phase to orthogonal two phase transformation. For the n-phase to two phase case, it can be expressed in the form of:

$$\left[f_{xy}\right] = \left[T_{(\Theta)}\right] \left[f_1 \,, f_2 \;\; \ldots n\right]^T \tag{2.1}$$

$$[T_{(\theta)}] = \sqrt{2/\mathrm{n}} \begin{bmatrix} \cos\frac{p}{2}\theta & \cos(\frac{p}{2}\theta - \alpha) & \dots \\ \sin\frac{p}{2}\theta & \sin(\frac{p}{2}\theta - \alpha) & \dots \end{bmatrix} \tag{2.2}$$

Where, α is the electrical angle between the two adjacent magnetic axes of a uniformly distributed n phase windings. The coefficient $\sqrt{2/\mathrm{n}}$ is introduced to make the transformation power invariant. Important subsets of the general n-phase to two phase transformation, though not necessarily power invariant, these are discussed in the following section.

Clark Transformation

The Clark transformation is basically employed to transform three phase to two phase quantities. The two phase variables in stationary reference frame are as shown in Figure 2.1. The α-axis coincides with the phase a-axis and the β-axis lags the α-axis by 90^0.

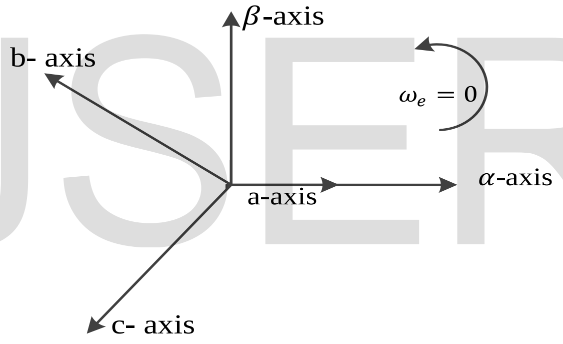


Figure 2.1 Two phase variables in stationary reference frame

$$[f_{\alpha\beta0}] = [T_{\alpha\beta0}][f_{\mathrm{abc}}] \tag{2.3}$$

The transformation matrix $[T_{\alpha\beta0}]$ is given by:

$$[\mathrm{T}\alpha\beta0] = \frac{2}{3}\begin{bmatrix} 1 & \frac{-1}{2} & -\frac{1}{2} \\ 0 & \sqrt{\frac{3}{2}} & -\sqrt{\frac{3}{2}} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.4}$$

The inverse transformation is given by:

$$[T\alpha\beta 0]^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \sqrt{\frac{3}{2}} & 1 \\ \frac{-1}{2} & -\sqrt{\frac{3}{2}} & 1 \end{bmatrix} \tag{2.5}$$

Park Transformation

The Park's transformation is a well-known transformation that converts the quantities into two phase synchronously rotating frame. The transformation is in the form of:

$$[Tdq0] = [Tdq0(\theta d)]\ [f_{abc}] \tag{2.6}$$

Where the dq0 transformation matrix is defined as:

$$\left[T_{dq0\ (\theta d)}\right] = \frac{2}{3}\begin{bmatrix} \cos\theta d & \cos(\theta d - \frac{2\pi}{3}) & \cos(\theta d + \frac{2\pi}{3}) \\ -sin\theta d & -\sin(\theta d - \frac{2\pi}{3}) & -\sin(\theta d + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.7}$$

And the inverse is given by:

$$[Tdq0]^{-1} = \frac{2}{3}\begin{bmatrix} \cos\theta d & -sin\theta d & 1 \\ \cos(\theta d - \frac{2\pi}{3}) & -\sin(\theta d - \frac{2\pi}{3}) & 1 \\ \cos(\theta d + \frac{2\pi}{3}) & -\sin(\theta d + \frac{2\pi}{3}) & 1 \end{bmatrix} \tag{2.8}$$

Where the θ_d is the transformation angle, the positive q-axis is defined as leading the positive d-axis by 90^0 in the original Park's transformation. Some authors define the q-axis as lagging the d-axis by 90^0. The transformation with q-axis lagging d-axis is given by:

$$\left[T_{dq0\ (\theta q)}\right] = \frac{2}{3}\begin{bmatrix} \cos\theta q & \cos(\theta q - \frac{2\pi}{3}) & \cos(\theta q + \frac{2\pi}{3}) \\ \sin\theta q & \sin(\theta q - \frac{2\pi}{3}) & \sin(\theta q + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.9}$$

The inverse is given by:

$$\left[Tdq0(\theta_q)\right]^{-1} = \begin{bmatrix} \cos\theta_q & \sin\theta_q & 1 \\ \cos(\theta_q - \frac{2\pi}{3}) & \sin(\theta_q - \frac{2\pi}{3}) & 1 \\ \cos(\theta_q + \frac{2\pi}{3}) & \sin(\theta_q + \frac{2\pi}{3}) & 1 \end{bmatrix} \tag{2.10}$$

Where $\theta_q$the transformation angle and q-axis is leading

The relationship between the θ_d and θ_q is:

$$\theta_q = \theta_d + \frac{\pi}{2} \tag{2.11}$$

One can show that [Tdq0] and [Tqd0], are basically the same, except for the ordering of the d and q variables.

2.1.4 Mathematical Modeling of Two Phase Induction Motor

Using transformation method in Section 2.2.3; mathematical modeling of two phase induction motor in different reference frame discuss as follows.

Stationary and Synchronous Reference Frames

There is some times a need to simulate an induction motor in the arbitrary rotating reference frame. But it is useful to convert one reference frame to other reference frame. The most commonly used reference frames are stationary reference frame and synchronously rotating frame. For transient studies of adjustable speed drives, it is usually more convenient to simulate an induction machine and its converter on a stationary reference frame. Moreover, calculation with stationary reference frame is less complex due to zero frame speed (some terms cancelled). For small signal stability analysis about some operating condition, a synchronously rotating frame which yields dc values of steady-state voltages and currents under balanced conditions is used.

When three phase induction motor is started it draws large currents, which produced voltage drips, oscillatory torques and even generate the harmonics in the power systems E. D. Mitronikas and A. N. Safacas were proposed in [10]. The d-q axis model is more reliable and accurate to investigate such problems. The dynamic model is derived by using two phase motor in direct and quadrature. This approach is useful because of the conceptual simplicity obtained with two set of windings one is on the stator and other is on the rotor, is described in [11],[12]. The three phase stationary reference frame (abc) into two phase reference frame ($dq0$) is carried out by Clark transformation equation.

The dynamic model of induction motor plays a vital role in the validation of design process of the motor drive systems, eliminating design mistakes and the resulting errors in the prototype constructions and testing was proposed by Prof. Himanshu K. Patel in [13]. Dynamic model includes three preferred speeds or reference frames as follows:

1. The stationary reference frame when the d-q axes do not rotate.

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

2. The rotor reference frame when the d-q axes rotate at rotor speed.
3. The synchronously rotating reference frame when the d-q axes rotate at synchronous speed.

The transient dynamic behavior of three phase squirrel cage induction motor can be analyzed by using any one of the mentioned three reference frames, proposed by P.C. Krause and C. H. Thomas in [14]. If the stator voltage is unbalanced or discontinuous and the rotor voltages are balanced, the stationary reference frames is useful. If the rotor and stator voltages are unbalanced or discontinuous, the rotor reference frames is used. And if stator and rotor all voltages are balanced and continuous, then synchronous reference frame is used.

2.2 Field Oriented Control of Induction Motor

Field Oriented Control can be used to vary the speed of an induction motor over a wide range. It was initially developed by Blaschke in [15]. It is commonly known as vector control because it controls both the magnitude and phase of the variables. In the vector control scheme, a complex current is synthesized from two current components, one of which is responsible for the flux level in the motor and another which controls the torque production in the motor. Vector control offers a number of benefits including speed control over a wide range, precise speed regulation, fast dynamic response and operation above base speed by Mircea Popescu in [16].

2.2.1 Principles of Vector Control

There are two approaches to obtain the flux vector, one direct measurement and other is indirect. According to the field oriented control can be classified as direct field orientation control (DFOC) and indirect field orientation control (IFOC).

Direct vector control Drive

In case of direct vector control the motor flux is measure by directly measure the air gap fluxes using Hall Effect sensors by Bimal K.Bose in [17]. However, the drift in the integrator with a search coil is problematic at very low frequencies and it tends to be temperature sensitive and fragile. An alternative approach is to measure the terminal voltage and phase currents of the machine and use these to estimate the flux.

In [19],[20], direct vector controlled drive, the control is dependent on stator resistance. Therefore, for perfect decoupling of flux and torque, estimation and compensation for these parameter variations is necessary. Also for certain speed sensorless drives, the estimation technique is dependent on plant parameters. In such cases, compensation for variation of these parameters is an absolute necessity. Not only for control or estimation in drives, non-hostile methods of winding temperature estimation using stator resistance identification has been found to be very reliable. In the direct field oriented control the stator frequency of the drive is not controlled. The motor is self-controlled by using the unit vector to help control the frequency and phase. There is no concern about instability because limiting within the safe limit automatically limits operation to the stable region. Transient response will be fast because torque control by i_{qs} does not affect flux. Vector control allows for speed control in all four quadrants since negative torque is directly taken care of in vector control.

Indirect field oriented Control

Indirect field oriented control; here the rotor flux angle is being measured indirectly, Instead of using air gap flux sensors. IFOC estimates the rotor flux by computing the slip speed (ω_{sl}). The stationary d and q axes are fixed on the stator and the rotor d and q axes are fixed on the rotor flux. The synchronous d and q-axes are rotating at synchronous speed and so there is a slip difference between the rotor speed and the synchronous speed given by:

$$\theta e=\int \omega_e dt = \int(\omega_{sl} + \omega_r)dt \tag{2.12}$$

In order to ensure decoupling between the rotor flux and the torque, the torque component of the current i_{qs} should be aligned with the synchronous q axis and the stator flux component of current i_{ds} should be aligned with the synchronous d-axis.

The stator and rotor voltage equations of induction motor in the field orientation synchronous reference frame can be written as in [21]:

$$V_{ds} = R_s i_{ds} - \omega_e \sigma i_{qs} + \frac{d}{dt}(\sigma i_{ds} L_s) - \frac{Lm}{Lr}\frac{d}{dt}\left[\omega_e\,\varphi_{qr} - \frac{d}{dt}(\varphi_{dr})\right] \tag{2.13}$$

$$V_{qs} = R_s i_{qs} + \omega_e \sigma i_{ds} + \frac{d}{dt}\left(\sigma i_{qs} L_s\right) + \frac{Lm}{Lr}\frac{d}{dt}\left[\omega_e\,\varphi_{dr} - \frac{d}{dt}\left(\varphi_{qr}\right)\right] \tag{2.14}$$

$$V_{dr} = R_r i_{dr} + \frac{d}{dt}(\varphi_{dr}) - [\omega_e - \omega_r]\varphi_{qr} = 0 \tag{2.15}$$

$$V_{qr} = R_r i_{qr} + \frac{d}{dt}\left(\varphi_{qr}\right) + [\omega_e - \omega_r]\varphi_{dr} = 0 \tag{2.16}$$

The rotor flux linkage equations can be written as:

$$\varphi_{dr} = L_r i_{dr} + L_m i_{ds}$$

$$\varphi_{qr} = L_r i_{qr} + L_m i_{qs}$$

These equations may be rewritten as:

$$i_{dr} = \frac{1}{L_r}\varphi_{dr} - \frac{L_m}{L_r} i_{ds}$$

$$i_{qr} = \frac{1}{L_r}\varphi_{qr} - \frac{L_m}{L_r} i_{qs}$$

Equating Equation (2.13) to Equation (2.16) and rotor flux linkage equations, the rotor flux and stator current written as follows:

$$\frac{d}{dt}\varphi_{dr} = \frac{L_m}{L_r} i_{ds} - \frac{1}{T_r}\varphi_{dr} - \omega_r\varphi_{qr} \qquad 2.17)$$

$$\frac{d}{dt}\varphi_{qr} = \frac{L_m}{L_r} i_{qs} - \frac{1}{T_r}\varphi_{qr} + \omega_r\varphi_{dr} \qquad (2.18)$$

$$\frac{d}{dt} i_{ds} = \frac{1}{\sigma L_s}\left(v_{ds} - R_s i_{ds} + \frac{L_m^2}{L_r T_r} i_{ds} - \frac{L_m}{L_r T_r}\varphi_{dr} - \frac{L_m}{L_r}\omega_r\varphi_{qr}\right) \qquad (2.19)$$

$$\frac{d}{dt} i_{qs} = \frac{1}{\sigma L_s}\left(v_{qs} - R_s i_{qs} - \frac{L_m^2}{L_r T_r} i_{qs} + \frac{L_m}{L_r T_r}\varphi_{qr} + \frac{L_m}{L_r} w_r\varphi_{dr}\right) \qquad (2.20)$$

Where, $\omega_{sl} = \omega_e - \omega_r$

For perfect decoupling control the total rotor flux needs to be aligned with the d-axis so that the q-axis rotor flux is set zero and d-axis rotor flux is maintained constant then equations written as: $\varphi_{qr} = 0$, this implies $\frac{d}{dt}\left(\varphi_{qr}\right) = 0$ and $\varphi_{dr=}\varphi_r$

Substitute into the previous equations to implement the indirect vector control strategy:

$$\theta_e = \theta_{sl} + \theta_r \qquad (2.20)$$

$$\varphi_r = L_m i_{ds} \qquad (2.21)$$

$$\frac{d}{dt}\left(\varphi_{qr}\right) + \frac{R_r}{L_r}\varphi_{dr} - \frac{L_m}{L_r} R_r i_{ds} - \omega_{sl}\varphi_{qr} = 0$$

$$\omega_{sl} = \frac{L_m R_r}{\varphi_{r\,L_r}}\, i_{qs}$$

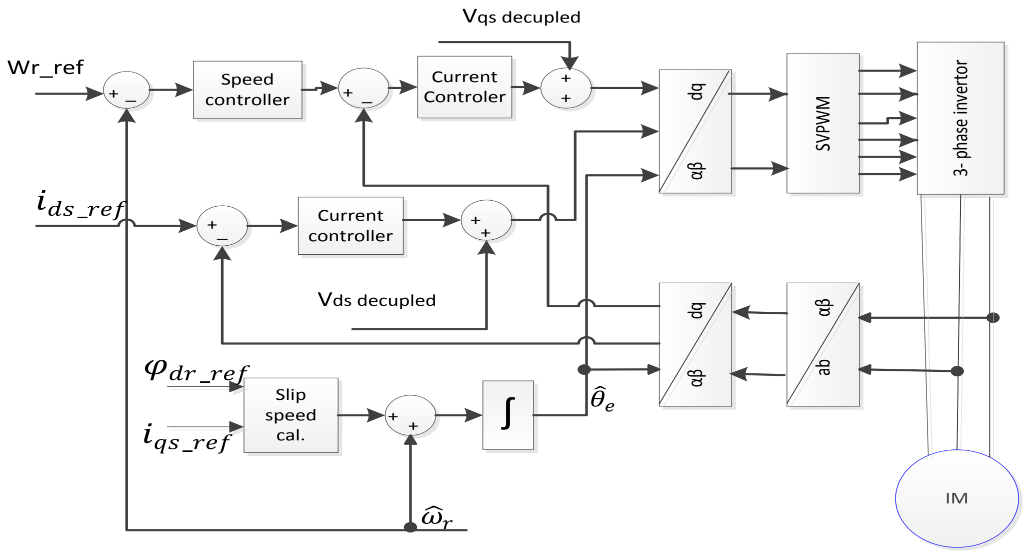


Figure 2.2 Sensorless indirect vector control of induction motor

2.2.2 Field Oriented Controller Design

The PI controllers are widely used in industries for high performance electric drives. The gains of the PI controllers can be designed by well-established method. Also, the PI controller is simple to operate and give zero steady state error. But improper selection of gains may affect the system performance making the system unstable. So, proper selection of gains is quite important. Different techniques can be employed for the design of gains of the PI controllers. The method used here is pole- zero cancellation technique. The higher order induction motor drive system is divided into two decoupled subsystems i.e. electrical and mechanical subsystem. The PI controllers are designed for d-q current and speed controller.

In the pole-zero cancellation technique, the zero of the PI controller is chosen so as to cancel one of the open loop poles of the system. Here, the ratio of the integral gain to the proportional gain is kept constant and a suitable value of the natural frequency of the first order system is to be chosen.

For rotor flux oriented control the rotor flux φ_{dr} is directed along the d-axis and is equal to φ_r and therefore φ_{qr}=0.Thus the Equation (2.13) to Equation (2.16) modifies to as shown in Equation (2.22) synchronous reference frame.

$$\frac{d}{dt}\begin{bmatrix} i_{ds} \\ i_{qs} \\ \varphi_{dr} \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sigma L_s}\left(R_s + \frac{L_m^2}{L_r T_r}\right) & \omega_e & \frac{1}{\sigma L_s}\frac{L_m}{L_r T_r} & 0 \\ -\omega_e & -\frac{1}{\sigma L_s}\left(R_s + \frac{L_m^2}{L_r T_r}\right) & \frac{-1}{\sigma L_s}\frac{L_m}{L_r}\omega_r & 0 \\ \frac{L_m}{T_r} & 0 & \frac{-1}{T_r} & 0 \\ 0 & \frac{L_m}{T_r} & -\omega_{sl} & 0 \end{bmatrix} \begin{bmatrix} i_{ds} \\ i_{qs} \\ \varphi_{dr} \\ 0 \end{bmatrix} + \frac{1}{\sigma L_s}\begin{bmatrix} v_{ds} \\ v_{qs} \\ 0 \\ 0 \end{bmatrix} \qquad (2.22)$$

From Equation (2.22) the transfer function for the d-q current controllers of the vector controlled induction motor drive is written as follows.

By applying the forward decoupling method from Equation (2.22) the d-q voltage are coupled by the following term:

$$v_{ds}\text{decoupling} = \omega_e i_{qs} + \frac{Lm}{\sigma L_s L_r T_r}\varphi_{dr}$$

$$v_{qs}\text{decoupling} = -(\omega_e i_{ds} + \frac{\omega r Lm}{\sigma L_s L_r}\varphi_{dr})$$

Current controller transfer function for d-axis and q-axis is given by:

$$\frac{i_{qs}}{v_{qs}} = \frac{i_{ds}}{v_{ds}} = \frac{\frac{1}{\sigma L_s}}{s + \frac{Rs}{\sigma L_s} + \frac{L_m^2}{\sigma L_s L_r T_r}} \qquad (2.23)$$

The electromagnetic torque equation of a rotor flux oriented IFOC is given as:

$$T_e = J\frac{d}{dt}w_r + f w_r + T_l \qquad (2.24)$$

$$T_e = \frac{3}{4}\frac{p\,L_m}{L_r}\varphi_{dr} i_{qs};\ \text{let}\quad k_t = \frac{3}{4}\frac{p\,L_m}{L_r}\varphi_{dr}$$

Where T_l, J, f and ω_r are the load torque, moment of inertia, coefficient of friction and rotor speed respectively. From Equation (2.24) the transfer function of the speed controller plant is given by:

$$\frac{\omega_r}{i_{qs}} = \left(\frac{\frac{1}{J}}{s+\frac{f}{J}}\right)\frac{1}{k_t} \qquad (2.25)$$

T_l, added as the disturbance when design the speed controller. It can be inferred from Equation (2.23) and (2.25) that the controller transfer functions are both first order system and therefore the design process for these speed and current controllers are the same.

Design PI Controllers in Discrete System

The transfer function of the PI d-q current controller is written as:

$C(s) = \frac{k_p\left(s+\frac{k_i}{k_p}\right)}{s}$ and the plant equation $G_i(s)$ is given by the Equation (2.23). The design procedure is conducted by calculate the open loop transfer function of the plant, derive the loop gain of the control system using pole zero cancellation method, and lastly obtain the controller parameters from the closed loop transfer function. For real time applications the system is controlled in discrete time domain and therefore the PI controller in Z domain the transfer function is $C_{(z)} = \frac{(k_p + k_i T)z - k_p}{z-1}$. The block diagram of the discrete control system for the current loop is shown in Figure 2.3.

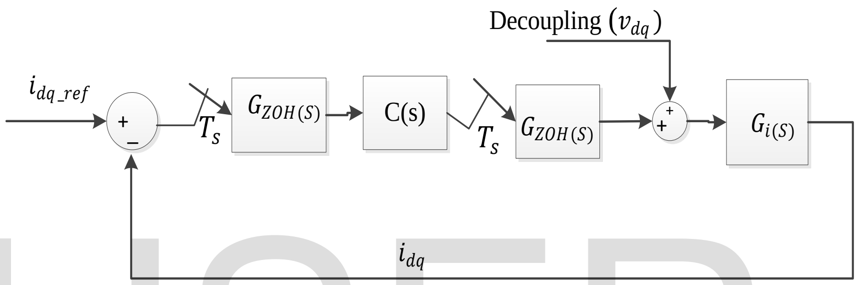


Figure 2.3 d-q Current controller in discrete time domain

The parameter k_p and k_i of the discrete controller are obtained by the following steps.

Step-1 calculates the open loop transfer function of the plant.

The open loop transfer function of the plant in Z-domain derived as:

$$G_{oiz} = \mathrm{C}(z)\ \mathrm{Z}[G_{ZOH}(s)\ G_i(s)] \tag{2.26}$$

Where $G_{ZOH}(s) = \frac{1-e^{-Ts}}{S}$ and the numerator $1 - e^{-Ts}$ in the Z domain can be written as: $1 - Z^{-1}$ similarly $C(z)$ can be written as $C(z) = k_p + k_i T \frac{Z - \frac{k_p}{k_p + k_i Ts}}{Z-1}$ and $G_i(s)$ is also given in Equation (2.23) and substitute in to Equation (2.26)

$$G_{oiz} = (k_p + k_i T_s \frac{Z - \frac{k_p}{k_p + k_i Ts}}{Z-1})(\frac{Z-1}{Z}\ Z\left[\frac{G_i(s)}{s}\right])$$

The Z transformation of $\frac{G_i(s)}{s}$ is given by $\frac{1}{\sigma L_s\, y}\left[\frac{1-e^{-yTs}}{Z-e^{-yTs}}\right]\frac{Z}{Z-1}$

where $y = \frac{R_s}{\sigma L_s} + \frac{\mathrm{L_m^2}}{\mathrm{\sigma L_s L_r T_r}}$

The open loop transfer function becomes:

$$G_{oiz} = (k_p + k_i T_s \frac{Z - \frac{k_p}{k_p + k_i T_s}}{Z-1}) \frac{1}{\sigma L_s\, y} \left[\frac{1-e^{-yT_s}}{Z-e^{-yT_s}}\right] \tag{2.27}$$

Step-2: deriving the loop gain of the transfer function using pole-zero cancellation method.

From Equation (2.27) using pole-zero cancellation it can be written:

$$\frac{k_p}{k_p + k_i T_s} = e^{-yT_s} \tag{2.28}$$

From Equation (2.28) $k_p = \frac{e^{-yT_s}}{1-e^{-yT_s}}\, k_i T_s$, then Equation (2.27) becomes:

$$G_{oiz} = k_p + k_i T_s \quad \frac{1}{\sigma L_s\, y} \left[\frac{1-e^{-yT_s}}{Z-1}\right] \tag{2.29}$$

Step-3: determining the controller parameters for a given bandwidth.

The closed loop transfer function of the system as shown in Figure 2.3 can be obtained as

$$G_{oiz} = \frac{k_p + k_i T_s\left(1-e^{-yT_s}\right)}{\sigma L_s\, y} \left(\frac{1}{Z-1+\frac{k_p + k_i Ts\left(1-e^{-yT_s}\right)}{\sigma L_s\, y}}\right) \tag{2.30}$$

In the discrete time domain the bandwidth of the standard closed loop first order system equation is shown by Equation (2.31):

$$\frac{1-e^{-B_w T_s}}{Z-e^{-B_w T_s}} \tag{2.31}$$

Where, B_w is the bandwidth, which was selected, based on the inverter switching frequency The inner (current) loop is faster than the outer (speed) loop, and the invertor switching time is faster than the current loop.

Equating Equation (2.30) and Equation (2.31)

$$k_p + k_i T_s = \frac{\sigma L_s\, y\left(1-e^{-B_w Ts}\right)}{1-e^{-yTs}} \tag{2.32}$$

Substitute the value of k_p from Equation (2.28) the proporational and integral gains of d-q current controller gains are obtainied as $k_i = \frac{\sigma L_s\, y\left(1-e^{-B_w Ts}\right)}{T_s}$ and $k_p = \frac{\sigma L_s\, e^{-yTs}\left(1-e^{-B_w Ts}\right)}{1-e^{-yTs}}$.

Similary the gain of the speed controllers is obtaioned as follows:

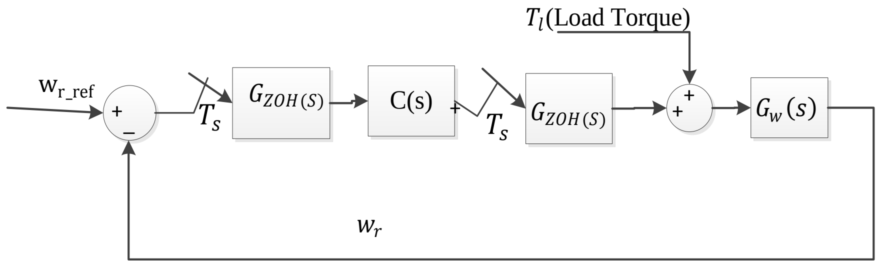


Figure 2.4 Rotor speed controller in discrete time domain

Step-1: Calculate the open loop transfer functions of the plant.

The open loop transfer function of the plant in Z domain can be derived as:

$$G_{owz} = \mathrm{C}(z)\ \mathrm{Z}[G_{ZOH}(s)\ G_w(s)] \tag{2.33}$$

Where, $G_{ZOH}(s) = \frac{1-e^{-Ts}}{s}$ and the numerator $1 - e^{-T_s}$ in the Z domain can be written as:

$1 - Z^{-1}$ similarly $C(z)$ can be written as $C(z) = k_p + k_i T_s \frac{Z - \frac{k_p}{k_p + k_i T s}}{Z-1}$ and $G_w(s)$ is also given in Equation (2.24). Substitute $C(z)$ in to Equation (2.33)

$$\frac{W_r(z)}{\Delta W_r(z)} = (k_p + k_i T_s \frac{Z - \frac{k_p}{k_p + k_i T s}}{Z-1})\left(\frac{Z-1}{Z}\ Z\left[\frac{G_w(s)}{s}\right]\right) \tag{2.34}$$

$$\frac{G_w(s)}{s} = \frac{1}{f}\left[\frac{1-e^{-xT}}{Z-e^{-xT}}\right] . \frac{Z}{Z-1}\frac{1}{k_t} \quad \text{where } x = \frac{f}{J}$$

The open loop transfer function becomes:

$$G_{owz} = (k_p + k_i T_s \frac{Z - \frac{k_p}{k_p + k_i T}}{Z-1})\left(\frac{1}{f}\left[\frac{1-e^{-xTs}}{Z-e^{-xTs}}\right]\frac{1}{k_t}\right) \tag{2.35}$$

Step-2: deriving the loop gain of the transfer function using pole-zero cancellation method.

From Equation (2.35) using pole-zero cancellation it can be written as:

$$\frac{k_p}{k_p + k_i T} = e^{-xTs} \tag{2.36}$$

Equation (2.36) can be written as $k_p = \frac{e^{-xTs}}{1-e^{-xT}}\ k_i T_s$ and substitute k_p in Equation (2.35):

$$G_{owz} = (k_p + k_i T_s\ )\left(\frac{1}{f}\left[\frac{1-e^{-xTs}}{Z-e^{-xTs}}\right]\right)\frac{1}{k_t} \tag{2.37}$$

Step-3: determining the controller parameters for a given bandwidth.

Closed loop transfer function of the system as shown in Figure 2.4 can be obtained as:

$$G_{cwz} = \frac{k_p + k_i Ts\left(1-e^{-xTs}\right)}{f} \frac{1}{k_t} \left(\frac{1}{Z-1+\frac{k_p+k_i Ts\left(1-e^{-xTs}\right)}{f}\frac{1}{k_t}} \right) \tag{2.38}$$

In discrete time domain the bandwidth of the standard first order closed loop system is given by Equation (2.39):

$$\frac{1-e^{-B_w Ts}}{Z-e^{-B_w Ts}} \tag{2.39}$$

Equating Equation (2.38) and (2.39):

$$k_p + k_i T_s = \frac{f\left(1-e^{-B_w Ts}\right)}{1-e^{-xTs}} k_t$$

Substitute the value of k_p from Equation (2.32) the proporational and integral gains of speed controller gains are obtainied as $k_i = \frac{f\left(1-e^{-B_w Ts}\right)}{T_s} k_t$ and $k_p = \frac{f e^{-xTs}\left(1-e^{-B_w Ts}\right)}{1-e^{-xTs}} k_t$.

The two phase voltage v_{ds} and v_{qs} in the stator reference frame are then transformed to three phase stator reference v_a, v_b and v_c which acts as modulating voltage for the modulator which uses the state space pulse width modulation (SVPWM) scheme. The modulator output which is in the form of pulses is used to drive the IGBT with anti-parallel diode acting as switches for the conventional two level voltage source inverter (VSI).

2.3 Space Vectors Pulse Width Modulation Inverter Fed Induction Motor

To control induction motor drives, PWM inverter is very popular. Using VSI possible to control both frequency and magnitude of the voltage and current applied to the induction motor drive. As a result, PWM inverter-fed IM drives are more variable, reliable and offer a wide range speed [22]. Using SVPWM techniques for three phase inverter switches and output of inverter is fed to speed control of IM drives. SVPWM is accomplished by rotating a reference vector around the state diagram, which is composed of six basic none zero vectors forming a hexagon shown in Figure 2.5. A circle can be inscribed inside the state map and corresponds to sinusoidal operation. The area inside the inscribed circle is called the linear modulation region or under-modulation region. As seen in Figure 2.5, the area between the inside circle and outside circle of the hexagon is called the nonlinear modulation region or over-modulation region. The concepts in the operation of linear and nonlinear modulation regions depend on the modulation index, which indirectly reflects on the inverter utilization capability.

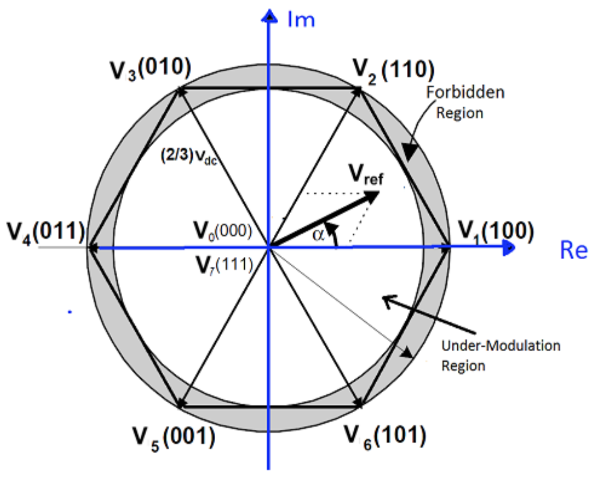


Figure 2.5 Under-modulation and over-modulation region in space vector representation [23]

In order to minimize the switching loss that is caused by the pure SVPWM, Max-Min offset is added to inject third harmonics and reduce the requirement of filter design. In this respect max-min SVPWM is used and the Simulink model is shown in appendix B.

Based on [18] the design of the offset is as follows:

Offset = $-(\frac{V_{\max}+V_{\min}}{2})$; $V_{\max} = max\{V_{An}, V_{Bn}, V_{cn}\}$; $V_{\min} = min\{V_{An}, V_{Bn}, V_{cn}\}$

The output voltage magnitude reaches the same value as that of the third harmonics injection PWM

2.3.1 Implementation Principle of Space Vector PWM

The principle of SVPWM requires the determination of a sector, calculation of vector segments, and it involves region identification based on the modulation index and calculation of switching time durations. The procedure for implementing a two level space vector PWM can be summarized as follows.

1. Calculate the angle θ and reference voltage vector $\vec{v}_{ref}$ based on the input voltage components.

2. Calculate the modulation index and determine if it is in the over-modulation region.

3. Find the sector in which $\vec{v}_{ref}$ lies, and the adjacent space vectors of $\vec{v}_k$ and $\vec{v}_{k+1}$ based on the sector angle θ.

4. Find the time intervals T_a and T_b and T_0 based on sampling time, and the angle.

5. Determine the modulation times for the different switching states.

Angle and Reference Voltage Vector

In the SVPWM, the three phase output voltage vector is represented by a reference vector that rotates at an angular speed of $\omega = 2pf$. The SVPWM uses the combinations of switching states to approximate the reference vector. A reference voltage vector ($\vec{v}_{ref}$) that rotates with angular speed ω in the α, β plane represents three sinusoidal waveforms with angular frequency w in the abc coordinate system.

The space vector with magnitude ($\vec{v}_{ref}$) rotates in a circular direction at an angular velocity of ω where the direction of rotation depends on the phase sequence of the voltages. If it has a positive phase sequence, then it rotates in the counter clockwise direction. Otherwise, it rotates in the clockwise direction with a negative phase sequence.

The three phase voltages could be described with only two components, α and β, in a two dimensional plane. The magnitude of each active vector is $2vdc/3$. The active vectors are 60^0 apart and describe a hexagon boundary. The locus of the circle projected by the space reference vector $\vec{v}_{ref}$ depends on $\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4, \vec{v}_5, \vec{v}_6, \vec{v}_7$

$$\vec{v}_{ref} = \frac{2}{3}[v_a + \propto v_b + \propto^2 v_c] \tag{2.40}$$

Where, $\propto = e^{j2\pi/3}$. The magnitude of the reference vector is given by:

$$|\vec{v}_{ref}| = \sqrt{v_\alpha{}^2 + v_\beta{}^2} \tag{2.41}$$

The phase angle is also express as $\theta = \tan^{-1}\left(\frac{v_\beta}{v_\alpha}\right)$. Where θ is an element of $[0,2\pi]$

Modulation Index of Linear Modulation

In the linear region, the rotating reference vector always remains within the hexagon. The largest output voltage magnitude is the radius of the largest circle that can be inscribed within the hexagon. This means that the linear region ends when the reference voltage is equal to the radius of the circle inscribed within the hexagon.

The ratio between the reference vector and the fundamental peak value of the square phase voltage wave $2\,V_{dc}/\pi$ is called the modulation index. The mode of operation is determined by the modulation index (MI). In this linear region, the MI can be express as:

$$\text{MI} = \frac{\vec{v}_{ref}}{V_{\text{max\_six step}}} \tag{2.42}$$

$$V_{\max\_\text{six step}} = \frac{4}{\pi}\left[\int_0^{\frac{\pi}{3}} \frac{v_{dc}}{3}\sin\theta d_\theta + \int_{\frac{\pi}{3}}^{\frac{\pi}{2}} \frac{2v_{dc}}{3}\sin\theta d_\theta\right]$$

$$V_{\max\_\text{six step}} = \frac{4v_{dc}}{3\pi}\left[\left(-\cos\frac{\pi}{3}+1\right)+\left(-2\cos\frac{\pi}{2}+2\cos\frac{\pi}{3}\right)\right]$$

$$V_{\max\_\text{six step}} = \frac{2v_{dc}}{\pi}$$

From the geometry of Figure 2.5 the maximum modulation index is obtained when $\vec{v}_{ref}$ the radius of the inscribed circle equals.

$$\vec{v}_{ref(max)} = \frac{2}{3}v_{dc}\cos\frac{\pi}{6}$$

Sector Determination

It is necessary to know in which sector the reference output lies in order to determine the switching time and sequence. The phase voltages correspond to eight switching states, six non-zero vectors and two zero vectors at the origin. Depending on the reference voltage $\vec{v}_\alpha$, and $\vec{v}_\beta$, the angle of the reference vector can be used to determine the sector as shown in Table 2.1.

Table 2.1 Sector identification

Sector	Degrees
1	$0 < \theta \leq 60°$
2	$60° < \theta \leq 120°$
3	$120° < \theta \leq 180°$
4	$180° < \theta \leq 240°$
5	$240° < \theta \leq 300°$
6	$300° < \theta \leq 360$

Duty Cycle Calculation

The duty cycle computation is done for each triangular sector formed by two state vectors. The magnitude of each switching state vector is $2v_{dc}/\sqrt{3}$ and the magnitude of a vector to the midpoint of the hexagon line from one vertex to another is $v_{dc}/\sqrt{3}$. The reference space vector rotates and moves through different sectors of the complex plane as time increases. In each PWM cycle, the reference vector $\vec{v}_{ref}$ is sampled at a fixed input sampling frequency(f_s). During this time, the sector is determined and the modulation vector $\vec{v}_{ref}$ is mapped onto two adjacent vectors. The non-zero vectors are written as:

$$\vec{v}_k = \frac{2}{3} v_{dc} e^{j(k-1)\frac{\pi}{3}} \tag{2.43}$$

Where k =1, 2, 3, 4, 5, 6

Therefore the non-zero vector for $\vec{v}_k$ and $\vec{v}_{k+1}$ become

$$\vec{v}_k \ = \frac{2}{3} v_{dc} \left[\cos(k-1)\frac{\pi}{3} + j\sin(k-1)\frac{\pi}{3}\right]$$

$$\vec{v}_{k+1} \ = \frac{2}{3} v_{dc} \left[\cos\left(k\frac{\pi}{3}\right) + j\sin\left(k\frac{\pi}{3}\right)\right]$$

Due to symmetry in the patterns in the six sectors, the integration can be carried out only on half pulse width modulation $T_S/2$. Zero voltages are applied during null state times.

$$\int_0^{\frac{T_S}{2}} \vec{v}_{ref}\ dt = \int_0^{\frac{T0}{4}} \overrightarrow{v_0}\, dt + \int_{\frac{T0}{4}}^{\frac{T0}{2}+Ta} \vec{v}_k\, dt + \int_{\frac{T0}{4}+Ta}^{\frac{T0}{2}+Ta+Tb} \vec{v}_{k+1}\ dt + \int_{\frac{T0}{4}+Ta+Tb}^{\frac{Ts}{2}} \vec{v}_7\, dt \tag{2.44}$$

$$\int_{\frac{T0}{4}+Ta+Tb}^{\frac{Ts}{2}} \overrightarrow{v_7}\, dt \ = 0,\ \int_0^{\frac{T0}{4}} \overrightarrow{v_0}\, dt = 0$$

Then the Equation (2.44) becomes

$$\int_0^{\frac{T_S}{2}} \vec{v}_{ref}\ dt = \int_{\frac{T0}{4}}^{\frac{T0}{2}+Ta} \vec{v}_k\, dt + \int_{\frac{T0}{4}+Ta}^{\frac{T0}{2}+Ta+Tb} \vec{v}_{k+1}\ dt$$

Thus the product of the reference voltage vector $\vec{v}_{ref}$ and $T_S/2$ are the sum of the voltage multiplied by the time interval of the chosen space vector. The reference voltage vector $\vec{v}_{ref}$ can be represented as function of $\vec{v}_k$and $\vec{v}_{k+1}$ as follows.

$$\vec{v}_{ref}\ \frac{T_s}{2} = \vec{v}_k T_a + \vec{v}_{k+1} T_b \tag{2.45}$$

$$\vec{v}_{ref} = \vec{v}_\alpha + \mathrm{j}\vec{v}_\beta$$

Where T_a and T_b denote the required on time of the active state vectors $\vec{v}_k$ and $\vec{v}_{k+1}$ during each sample period, and k is the sector number denoting the reference location. The calculated times T_a and T_b are applied to the switches to produce space vector PWM switching patterns based on each sector. The switching time is arranged according to the first half of the switching period while the half is a reflection forming asymmetrical pattern. If $\vec{v}_{ref}$ lies exactly in the middle between two vectors.

Assuming that the reference voltage and the voltage vector $\vec{v}_k$ and $\vec{v}_{k+1}$ are constant during each pulse width modulation period T_s and splitting the reference voltage $\vec{v}_{ref}$ into its real and imaginary components gives the following result.

$\begin{bmatrix} v_\alpha \\ v\beta \end{bmatrix} \frac{Ts}{2} = \frac{2v_{dc}}{3} \left(\begin{bmatrix} \cos\frac{(k-1)\pi}{3} & \cos\frac{(k\pi)}{3} \\ \sin\frac{(k-1)\pi}{3} & \sin\frac{(k\pi)}{3} \end{bmatrix} \begin{bmatrix} T_a \\ T_b \end{bmatrix} \right)$ from this equation T_a and T_b can be calculated as follows.

$$\begin{bmatrix} T_a \\ T_b \end{bmatrix} = \frac{\sqrt{3}Ts}{2v_{dc}} \begin{bmatrix} \sin\frac{(k\pi)}{3} & -\cos\frac{(k\pi)}{3} \\ -\sin\frac{(k-1)\pi}{3} & \cos\frac{(k-1)\pi}{3} \end{bmatrix} \begin{bmatrix} v_\alpha \\ v\beta \end{bmatrix} \tag{2.46}$$

Since the sum of $2T_a$ and $2T_b$ should be less than or equal to T_s, the inverter has to stay in a zero state for the rest of the period. The duration of the null vectors is the remaining time in the switching period.

T_s = T_0+2($T_a + T_b$), Then the time interval for the zero voltage vectors is

$$T_0 = T_s - 2(T_a + T_b) \tag{2.47}$$

The switching times are arranged symmetrical around the center of the switching period .The zero vector v_1 (1,1,1) is placed at the center of the switching period, and the zero vector v_0 (0,0,0) at the start and the end, and the total period for a zero vector is divided equally among the two zero vectors. In the under modulation region as the modulation index increases the reference voltage vector grows outward in magnitude. It reaches the inscribed circle of the hexagon and T_0 will reduce to zero whenever the tip of the reference voltage vector is on the hexagon. If the modulation index increases further T_0 becomes negative and meaningless.

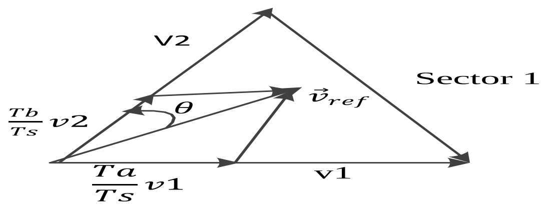


Figure 2.6 Sector identification

The calculated values of T_a and T_b in term of $T_s/v_{dc}$for all six sectors are listed in Table 2.2 the time durations of two adjacent non-zero vectors in each sector are calculated. Time Intervals Ta and Tb for each sector based on the magnitude and phase of the reference voltage. From Figure 2.6, a zero state vector is applied followed with two adjacent active vectors in half of the switching period. The next half of the switching period is symmetrical to the first half. To generate the signals that produce the rotating vector, an equation is required to determine the time intervals for each sector. Table 2.2 shows that the pulse patterns generated by space vector PWM in the given sector.

Table 2.2 Duty cycle calculation

Sector	θ	T_a	T_b
1	$0 < \theta \leq 60^0$	$\frac{3v_\alpha}{4} - \sqrt{3}\frac{v_\beta}{4}$	$\sqrt{3}\frac{v_\beta}{2}$
2	$60 < \theta \leq 120^0$	$\frac{3v_\alpha}{4} + \sqrt{3}\frac{v_\beta}{4}$	$\frac{-3v_\alpha}{4} + \sqrt{3}\frac{v_\beta}{4}$
3	$120 < \theta \leq 180^0$	$\sqrt{3}\frac{v_\beta}{2}$	$\frac{-3v_\alpha}{4} - \sqrt{3}\frac{v_\beta}{4}$
4	$180 < \theta \leq 240^0$	$\frac{-3v_\alpha}{4} + \sqrt{3}\frac{v_\beta}{4}$	$-\sqrt{3}\frac{v_\beta}{2}$
5	$240 < \theta \leq 300^0$	$\frac{-3v_\alpha}{4} - \sqrt{3}\frac{v_\beta}{4}$	$\frac{3v_\alpha}{4} - \sqrt{3}\frac{v_\beta}{4}$
6	$300 < \theta \leq 360^0$	$-\sqrt{3}\frac{v_\beta}{2}$	$\frac{3v_\alpha}{4} + \sqrt{3}\frac{v_\beta}{4}$

Determination of the Switching Times for Each Transistor Switch

It is necessary to arrange the switching sequence so that the switching frequency of each inverter leg is minimized. There are many switching patterns that can be used to implement SVPWM. To minimize the switching losses, only two adjacent active vectors and two zero vectors are used in a sector [24]-[26]. To meet this optimal condition, each switching period starts with one zero vectors and end with another zero vector during the sampling time. This rule applies normally to three phase inverters as a switching sequence. Therefore, the switching cycle of the output voltage is double the sampling time and the two output voltage waveforms become symmetrical.Table2.3 shows a symmetric switching sequence. Referring to this table, the binary representations of two adjacent basic vectors differ in only one bit, so that only one of the upper transistors switches is closed when the switching pattern moves from one vector to an adjacent one. The two vectors are time weighted in a sample period T_s to produce the desired output voltage.

Table 2.3 Seven Segment Switching Sequence.

Space vector	Switching state	On state switch	Off state switch	Vector definition
$\vec{v}_0$	000	S_4, S_6, S_2	S_1, S_3, S_5	$\vec{v}_0 = 0$
$\vec{v}_1$	100	S_1, S_6, S_2	S_4, S_3, S_5	$\vec{v}_1 = \frac{2}{3}V_{dc}$
$\vec{v}_2$	110	S_1, S_3, S_2	S_4, S_6, S_5	$\vec{v}_2 = \frac{2}{3}V_{dc}e^{j\frac{\pi}{3}}$
$\vec{v}_3$	010	S_4, S_3, S_2	S_1, S_6, S_5	$\vec{v}_3 = \frac{2}{3}V_{dc}e^{j2\frac{\pi}{3}}$
$\vec{v}_4$	011	$S_4, S_{3,}S_5$	S_1, S_6, S_2	$\vec{v}_4 = \frac{2}{3}V_{dc}e^{j\frac{3\pi}{3}}$
$\vec{v}_5$	001	S_4, S_6, S_5	S_1, S_3, S_2	$\vec{v}_5 = \frac{2}{3}V_{dc}e^{j\frac{4\pi}{3}}$
$\vec{v}_6$	101	S_1, S_6, S_5	$S_4, S_{3,}S_2$	$\vec{v}_6 = \frac{2}{3}V_{dc}e^{j\frac{5\pi}{3}}$
$\vec{v}_7$	111	S_1, S_3, S_5	S_4, S_6, S_2	$\vec{v}_7 = \frac{2}{3}V_{dc}$

Chapter Three

Model Reference Adaptive System Design

Introduction

The MRAS speed estimator is the most attractive approaches due to its design simplicity. It is based on the principle, in which the outputs of two models, one independent of the rotor speed and the other is dependent on the estimated rotor speed, have been used to form an error vector. The error vector is driven to zero by an adaptation mechanism which yields the estimated rotor speed. Depending on the choice of output quantities that form the error vector, several MRAS structures are possible. The most common MRAS structure is that based on the rotor flux error vector which provides the advantage of producing rotor flux angle estimate for the field orientation scheme. Other MRAS structures have also been proposed recently that use the back_ emf and the reactive power as the error vector estimator. In recently research area is concerned on stator current based MRAS and it is the intention of this thesis. Based on the theory of MRAS, estimation of rotor speed has been described and the detailed design of adaptation mechanism is presented in this Chapter. An adaptation technique is simple and needs a low computation power and has a high speed adaptation even at low speeds because it eliminates the produced error in the speed adaptation and it is more stable and robust. Root locus design is presented to design the PI controller parameters. This section covers the following outlines including this introduction.

Section 3.1 shortly presents the different speed estimation schemes of sensorless speed control induction motor drive.

Section 3.2 presents model reference adaptive system including the design of stator current based MRAS speed estimator.

Section 3.3 presents stability analysis of stator current based MRAS speed estimator technique for robust PI control parameters design.

3.1 Speed Estimation Schemes of Sensorless Speed control Induction Motor Drives

Various techniques are available for speed estimation of IM drives were proposed in [26]. These are mainly classified as; Rotor flux based, Frequency signal injection based, Model Reference Adaptive System (MRAS) based, Observer based, Artificial Intelligence (AI) based and Rotor slot harmonics based methods. Among all these methods, MRAS is simple, requires less computation time and has good stability was stated in [27], [28].

3.2 Model Reference Adaptive System

Model Reference Adaptive system (MRAS) is one of the famous speed estimation usually used for sensorless speed control of induction motor drive. It is one of many promising techniques employed in adaptive control. Among various types of adaptive system configuration, MRAS is important since it leads to relatively easy to implement systems with high speed of adaptation for a wide range of applications. One of the most noted advantage of this type of adaptive system is its high speed of adaptation. This is due to the fact that a measurement of the difference between the outputs of the reference model and adjustable model is obtained directly by the comparison of the states of the reference model with those of the adjustable system. The block reference model represents demanded dynamics of actual control loop. The block adjustable model has the same structure as the reference one, but with adjustable parameters instead of the unknown ones as shown in Figure 3.1. The MRAS speed estimation structure consists basically of *a* reference model, adjustable model and an adaptive mechanism. The reference model, which is independent of the rotor speed calculates the state variable ($i_{\alpha s}, i_{\beta s}$) from the induction motor model and the adjustable model, which is dependent on the rotor speed, estimates the state variables ($\hat{\imath}_{s\alpha}, \hat{\imath}_{s\beta}$) and ($\hat{\varphi}_{r\alpha}, \hat{\varphi}_{r\beta}$). The error between measured and estimated state variables is then used to drive an adaptation mechanism which generates the estimated speed, for the adjustable model as shown in the block diagram of Figure 3.1. It should be noted that, speed estimation methods using MRAS can be classified into various types according to the state variables. The most commonly used are the rotor flux based MRAS, reactive power MRAS, back emf based MRAS, and stator current based MRAS. In [30], [31] the rotor flux based MRAS the rotor flux is used as an output value for the model to estimate the rotor speed. In rotor flux based MRAS, the presence of an open integration in the stator leads to problems with initial conditions and drift. A low pass filter may be used instead of the pure integration; however, it has a degrading effect on speed estimation at low speeds and introduces time delay. The model reference adaptive approach based on back -emf rather than the rotor flux offers an alternative to avoid the problem of pure integration. The pure integration is avoided in this approach and there are no low pass filters that create a bandwidth limit.

A more severe source of inaccuracy is a possible mismatch of the reference model parameters; particularly of the stator resistance was proposed by M. Rashed and A.F. Stronach in [32]. In this thesis the stator current based MRAS estimator is based on the comparison between the measured stator current of the IM and the estimated current obtained from the stator current model, which is used to estimate the rotor speed.

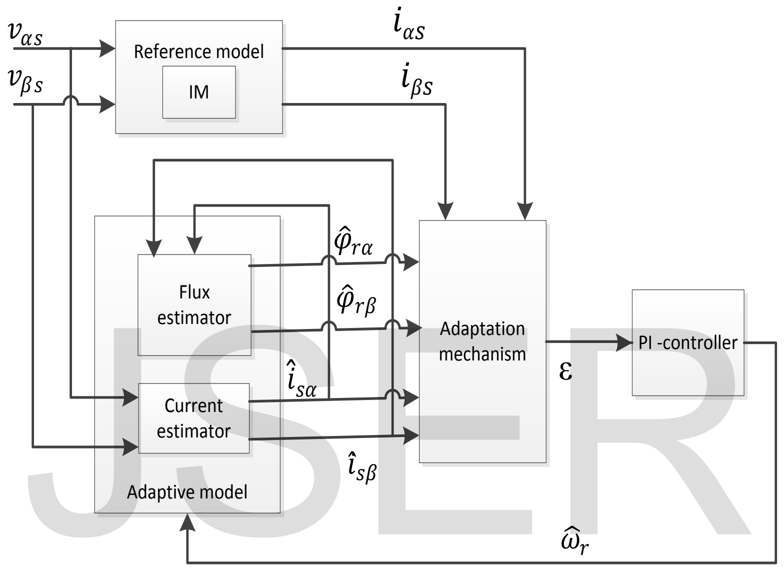


Figure 3.1 General block diagram of stator current based MRAS speed estimator.

As shown in Figure 3.1, the output of a reference model is compared to the output of an adaptive model until the error between the two models converges to zero. The adaptive model is based on stator current and rotor flux estimation and the reference model is the induction motor itself. The speed estimation technique (adaptation mechanism) and stability analysis of stator current based MRAS speed estimator scheme are discussed under Section 3.3 and 3.4.

3.3 Stator current based MRAS speed estimator design

The stator current based MRAS speed estimator is design based on the comparison between the measured stator current of the IM and the estimated current obtained from the flux current model is shown in Figure 3.1.

Using Equation (2.17) to Equation (2.20) the mathematical model of the rotor flux and the stator current are estimated in stationary reference frames as follows.

$$\frac{d}{dt}\hat{\varphi}_{r\alpha} = \frac{L_m}{L_r}\hat{\imath}_{s\alpha} - \frac{1}{T_r}\hat{\varphi}_{r\alpha} - \widehat{\omega}_r\hat{\varphi}_{r\beta} \tag{3.1}$$

$$\frac{d}{dt}\hat{\varphi}_{r\beta} = \frac{L_m}{L_r}\hat{\imath}_{s\beta} \ - \frac{1}{T_r}\hat{\varphi}_{r\beta} + \widehat{\omega}_r\hat{\varphi}_{r\alpha} \tag{3.2}$$

$$\frac{d}{dt}\hat{\imath}_{s\alpha} = \frac{1}{\sigma L_s}\begin{pmatrix} v_{\alpha s} - R_s\hat{\imath}_{s\alpha} + \frac{L_m^2}{L_rT_r}\hat{\imath}_{s\alpha} - \\ \frac{L_m}{L_rT_r}\hat{\varphi}_{r\alpha} - \frac{L_m}{L_r}\widehat{\omega}_r\hat{\varphi}_{r\beta} \end{pmatrix} \tag{3.3}$$

$$\frac{d}{dt}\hat{\imath}_{s\beta} = \frac{1}{\sigma L_s}\begin{pmatrix} v_{\beta s} - R_s\hat{\imath}_{s\beta} \ - \frac{L_m^2}{L_rT_r}\hat{\imath}_{s\beta} \quad + \\ \frac{L_m}{L_rT_r}\hat{\varphi}_{r\beta} + \frac{L_m}{L_r}\widehat{\omega}_r\hat{\varphi}_{r\alpha} \end{pmatrix} \tag{3.4}$$

Rearrange Equations (3.1) to Equations (3.4) in matrix form:

Let, $\dot{\boldsymbol{x}}$ = Ax + B$\vec{v}_s$ and $y = Cx$

Where, $\mathbf{x} = \begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \\ \varphi_{\alpha r} \\ \varphi_{\beta r} \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \end{bmatrix}$ A = the system matrix rearranged as follows:

$$A = \begin{bmatrix} -\frac{1}{\sigma L_s}\left(R_s + \frac{L_m^2}{L_rT_r}\right) & 0 & \frac{1}{\sigma L_s}\frac{L_m}{L_rT_r} & \frac{1}{\sigma L_s}\frac{L_m}{L_r}\widehat{\omega}_r \\ 0 & -\frac{1}{\sigma L_s}\left(R_s + \frac{L_m^2}{L_rT_r}\right) & -\frac{1}{\sigma L_s}\frac{L_m}{L_r}\widehat{\omega}_r & \frac{1}{\sigma L_s}\frac{L_m}{L_rT_r} \\ \frac{L_m}{T_r} & 0 & \frac{-1}{T_r} & -\widehat{\omega}_r \\ 0 & \frac{L_m}{T_r} & \widehat{\omega}_r & \frac{-1}{T_r} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \text{ and } B = \left[\frac{1}{\sigma L_s}\right] C^T$$

In the stator current based MRAS rotor speed estimator the adaptation algorithm is based on the error between estimated and measured stator current based on the Lyapunov function. The adaptation mechanism can be derived from the adaptive stator current and rotor flux estimator is constructed as follows.

$$\dot{\hat{\mathbf{x}}} = A\hat{\mathbf{x}} + Bv_s$$

$$\mathbf{y} = C\hat{\mathbf{x}}$$

Let,

$$\varepsilon_{i_{\alpha s}} = i_{\alpha s} - \hat{\imath}_{s\alpha}$$

$$\varepsilon_{i_\beta} = \ i_{\beta s} - \hat{\imath}_{s\beta}$$

$$\varepsilon_s = i_s - \hat{i}_s \quad \text{where, } i_s = \begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \end{bmatrix}$$

$$\varepsilon_\omega = \omega_r - \widehat{\omega}_r$$

Assuming the estimated and the measured rotor flux are the same.

$$\varphi_{\alpha r} = \hat{\varphi}_{r\alpha}$$

$$\varphi_{\beta r} = \hat{\varphi}_{r\beta}$$

Considering the Lyapunov function candidate [33]

$$v = v_1 + v_2$$

Let, $v_1 = \varepsilon^T \varepsilon$ and $v_2 = \frac{{\varepsilon_\omega}^2}{\gamma}$ where, $\varepsilon = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}}$

The derivative of this Lyaponov candidate function written as:

$$\frac{dv}{dt} = \begin{pmatrix} \varepsilon^T[(A)^T + A]\varepsilon - 2(w_r - \widehat{\omega}_r) \\ \left[k\left(\varepsilon_{i_\alpha}\hat{\varphi}_{r\beta} - \varepsilon_{i_\beta}\hat{\varphi}_{r\alpha}\right) - \frac{1}{\gamma}\frac{d}{dt}\widehat{\omega}_r\right] \end{pmatrix} \tag{3.5}$$

$$\varepsilon^T[(A)^T + (A)]\varepsilon < -Q \tag{3.6}$$

Where, Q=$\in In$ and $\in > 0$ and In is identity matrix

Using the Lyaponov stability of adaptive estimator has proved if two conditions are full fill. The eigenvalue of the estimator are selected to have negative real parts so that the states of the estimator will converge to the desired states of the estimated system. The term in factor of $(\omega_r - \widehat{\omega}_r)$ in the Equation (3.5) must be zero. The expression of the derivative of estimated speed becomes:

$$k\left(\varepsilon_{i_\alpha}\hat{\imath}_{s\beta} - \varepsilon_{i_\beta}\hat{\varphi}_{r\alpha}\right) - \frac{1}{\gamma}\frac{d}{dt}\widehat{\omega}_r = 0 \tag{3.7}$$

$$\frac{d}{dt}\widehat{\omega}_r = \gamma k\left(\varepsilon_{i_\alpha}\hat{\varphi}_{r\beta} - \varepsilon_{i_\beta}\hat{\varphi}_{r\alpha}\right) \tag{3.8}$$

$$\widehat{\omega}_r = K\int\left(\varepsilon_{i_\alpha}\hat{\varphi}_{r\beta} - \varepsilon_{i_\beta}\hat{\varphi}_{r\alpha}\right)dt \tag{3.9}$$

However this adaptive law of the speed has obtained by adjusted K (finite positive constant). For improved the dynamic of this estimator during the transitory phase of rotor speed, estimate the speed by large PI regulator; so it require a supplementary term proportional. Then

$$\widehat{\omega}_r = ki\int\left(\varepsilon_{i_\alpha}\hat{\varphi}_{\beta r} - \varepsilon_{i_\beta}\hat{\varphi}_{\alpha r}\right)dt + kp\left(\varepsilon_{i_\alpha}\hat{\varphi}_{\beta r} - \varepsilon_{i_\beta}\hat{\varphi}_{\alpha r}\right) \tag{3.10}$$

Where ki, Kp are adaptive gains for speed estimator

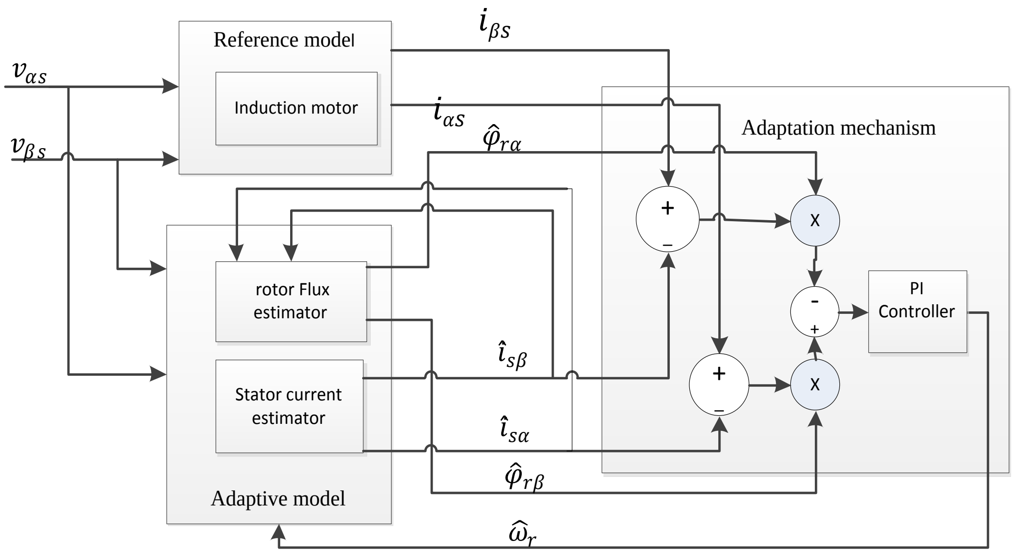


Figure 3.2 Adaptation mechanism for stator current based MRAS speed estimator.

3.4 Stability Analysis of Stator Current Based MRAS Speed Estimator

The stability analysis of the stator current based MRAS speed estimator is tested from the point of the IM and PI controller parameter changes, on the basis of the estimator transfer function. From the point of view the rotor speed estimation, the stator current based MRAS speed estimator can be analyzed as a system controlled by the signal ε which is the combination of the stator current and the rotor flux used in the adaptation loop. To find the value of the PI adaptive gains, analyze the closed loop transfer function of stator current based MRAS speed estimator.

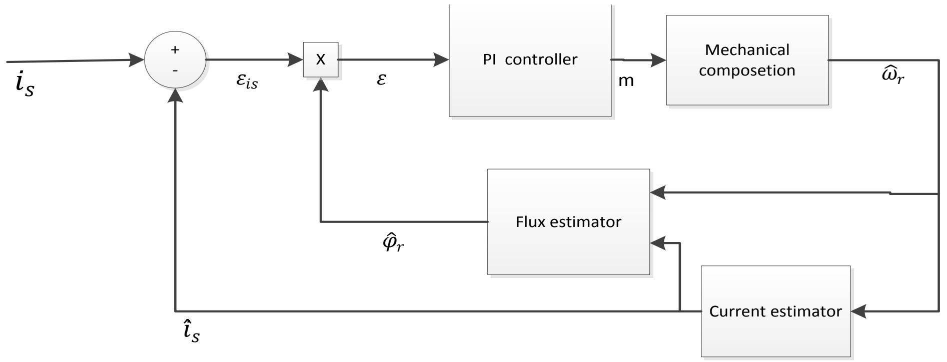


Figure3.3 Closed loop stator current based MRAS speed estimator with mechanical compensation.

The mechanical model of the system at stationary reference frame is obtained as follows.

$$\frac{d}{dt}\omega_r = \frac{3pL_m}{4JL_r}(\varphi_{\alpha r} i_{\beta s} - \varphi_{\beta r} i_{\alpha s}) - \frac{f}{J}\omega_r - \frac{T_l}{J} \tag{3.11}$$

Let $k_t = \frac{3pL_m}{4JL_r}$

Then the model in Figure 3. 4 simplified as:

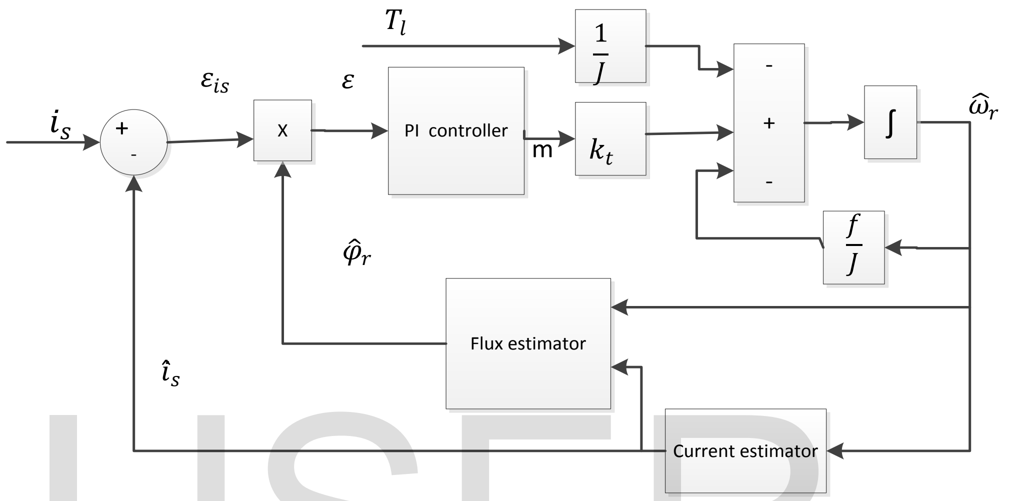


Figure 3.5 Simplified form of closed loop stator current based MRAS speed estimator.

In order to drive the PI controller parameter a linearized transfer functions between $m_{(s)}$ and $\varepsilon_{(s)}$ is obtained. Using small signal analysis under the assumption of field orientation, the difference between the measured and the estimated value; with x_0 is the operating point is written as:

$$\left.\begin{aligned} \Delta\dot{\boldsymbol{x}} &= \mathrm{A}\Delta\mathbf{x} + \Delta\mathrm{A}\boldsymbol{x_0} \\ \Delta\boldsymbol{y} &= C\Delta\mathbf{x} = C(sI - A)^{-1}\Delta\mathrm{A}\boldsymbol{x_0} \end{aligned}\right\} \tag{3.12}$$

Where, $\Delta\mathrm{x} = (\boldsymbol{x} - \widehat{\boldsymbol{x}})$ and

$$\boldsymbol{x_0} = \begin{bmatrix} i_{ds0} & i_{qs0} & \varphi_{dr0} & \varphi_{qr0} \end{bmatrix}^T$$

The system matrix ΔA is expressed as follows considering the rotor speed is as the only variable parameter.

$$\Delta A = \begin{bmatrix} 0 & 0 & 0 & a \\ 0 & 0 & -a & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Delta\omega_r \ \text{ where, } \ a = \frac{1}{\sigma L_s}\frac{L_m}{L_r}$$

$$\Delta y = \begin{bmatrix} i_{\alpha s} - \hat{\imath}_{s\alpha} \\ i_{\beta s} - \hat{\imath}_{s\beta} \end{bmatrix} = C(sI - A)^{-1} \begin{bmatrix} 0 & 0 & 0 & a \\ 0 & 0 & -a & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Delta\omega_r \, \boldsymbol{x_0} \tag{3.13}$$

Let the adjoint of $(sI - A)$:

$$adj(sI - A) = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \tag{3.14}$$

$$\begin{bmatrix} \frac{\Delta i_{\alpha s}}{\Delta\omega_r} \\ \frac{\Delta i_{\beta s}}{\Delta\omega_r} \end{bmatrix} = C(sI - A)^{-1} \begin{bmatrix} 0 & 0 & 0 & a \\ 0 & 0 & -a & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha s0} & i_{\beta s0} & \varphi_{\alpha r0} & \varphi_{\beta r0} \end{bmatrix}^T \tag{3.15}$$

Then substitute Equation (3.14) into the Equation (3.15) and solve $\Delta\omega_r$ from the mechanical model the transfer function of m(s) with $\varepsilon(s)$ is obtained.

From Equation (3.15) and the adjoint of $(sI - A)$

$$\frac{\Delta i_{\alpha s}}{\Delta\omega_r} = \frac{(c_{14} - a\, c_{12})}{|sI-A|}\varphi_{\beta r0}, \text{ and } \quad \frac{\Delta i_{\beta s}}{\Delta\omega_r} = \frac{(c_{24} - a\, c_{22})}{|sI-A|}\varphi_{\alpha r0}$$

$$\varepsilon = \left((i_{\alpha s(s)} - \hat{\imath}_{s\alpha(s)})\varphi_{\beta r(s)} - (i_{\beta s(s)} - \hat{\imath}_{s\beta(s)})\varphi_{\alpha r(s)}\right)$$

$$\varepsilon = \left(\Delta i_{\alpha s}\varphi_{\beta r0(s)} - \Delta i_{\beta s}\varphi_{\alpha r0(s)}\right)$$

$$\varepsilon = \left\{\frac{(c_{14} - a\, c_{12})}{|sI-A|}\varphi_{\beta r0}\varphi_{\beta r0} - \frac{(c_{24} - a\, c_{22})}{|sI-A|}{\varphi_{\alpha r0}}^2\right\} \ \Delta\omega_r$$

$$\varepsilon = -\frac{(c_{24} - a\, c_{22})}{|sI-A|}{\varphi_{\alpha r0}}^2 \ \Delta\omega_r \tag{3.16}$$

Then the transfer functions between $\varepsilon_{(s)}$ and $m_{(s)}$ is obtained as:

$$\frac{\varepsilon_{(s)}}{\mathrm{m(s)}} = \{-\frac{(c_{24} - a\, c_{22})}{|sI-A|}\frac{1}{\left(s+\frac{f}{J}\right)}\frac{P}{2J}\ {\varphi_{\alpha r0}}^2\} \tag{3.17}$$

Let $G_{(s)}$ is $\frac{\varepsilon_{(s)}}{\mathrm{m(s)}}$ then the simplified closed loop block diagram of stator current based MRAS speed estimator is shown in the following diagram.

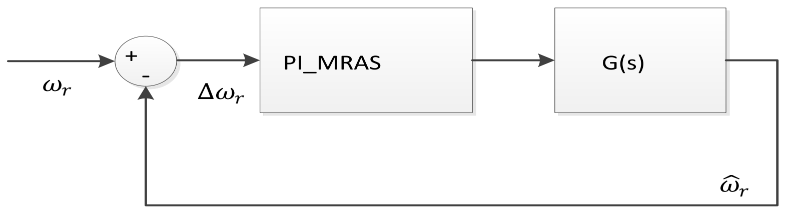


Figure 3.6 Closed loop block diagram of stator current based MRAS speed estimator

$$\frac{\omega_r}{\hat{\omega}_r} = \frac{G(s)\ \{kp_{mras} + \frac{ki_{mras}}{S}\}}{1 + G(s)\ \{kp_{mras} + \frac{ki_{mras}}{S}\}}$$

The design of K_p and K_i is introduce to ensure stability, error tracking and robust operation. The design criteria for this adaptive PI controller is done using Matlab /SISOTool, by using the transfer function of the plant and set the time domain constraint. The constraint considered in this design is percent of overshoot; settling time and rise time are less than five percent, less than two second and less than two second respectively.

As shown in Figure 3.7 the design of K_p and K_i is selected to ensure that all the poles and zeros are located in the left hand side of s-plane and this allows for the required fast and stable response. The location of closed loop transfer function poles characterizes the control system dynamics. Therefore, the location of the PI controller zero should be on the real axis to make sure that fast dynamics response.

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

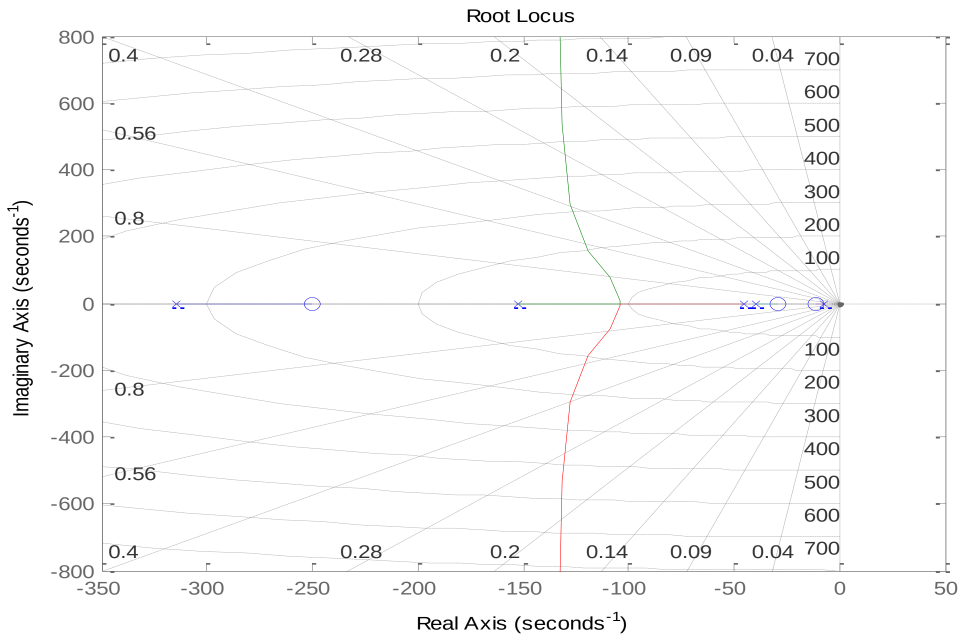


Figure 3.7 root locus of closed loop adaptive control system

Therefore, ki is taken of 89.173 and K_p/K_i should equal the 0.012. This allows for fast exponentially speed error decaying and acceptable overshoot or undershoots. The location of the PI controller zero K_p/K_i can be designed at any speed. All the poles of the closed loop transfer function lie in the left half of the s-plane then the system has been stable in the operating point.

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

Chapter Four

Simulation Results and Discussion

4.1. Introduction

The propose control system represented by Figure1.1 is designed for simulation by Matlab/Simulink model. Simulation results are presented and discussed to show the effectiveness of the proposed drive system based on stator current based MRAS speed estimator and indirect vector control at different operating conditions. For studying the performances of proposed system, a series of simulations and measurements have been carried out. In this respect, the dynamic response of the propose speed estimation algorithm is studied under different speed command.

4.2 Simulink Model of the MRAS Based Sensorless Speed Control of Induction Motor

A Simulink model of sensorless speed control of induction motor drive was developed using components from the Power System's Block set. This is a particularly useful add-on to Simulink that provides models for a wide range of power electronics devices and control structures. The inverter and asynchronous motor configuration is used from an existing Simulink file. The Indirect Field Oriented Control (IFOC) as well as the speed estimation structure was implemented using the theory outlined in Chapter 2 and Chapter 3 of this thesis and is shown in Figure 4.1.The induction motor used in this thesis is three phase with rate power of 0.18kw and rated speed 1300 rpm Squarely cage induction motor parameters are given in Table 4.1.

Table 4.1 Parameters used for simulation.

Stator resistance	Rotor Resistance	Stator Inductance	Rotor inductance	Mutual inductance	Moment of inertia	Viscous friction
R_S = 11.05	R_r = 6.11	L_S = 0.316423	L_r = 0.316423	L_m= 0.2939	J =0.009	f = 0.00061

Kp_{spd}	Ki_{spd}	Kp_{id}=kp_{iq}	Ki_{id}=ki_{iq}	Kpmras	kimras	Initial condition	Pwm Frequency
0.3949	0.263	0.0570	8.1105e+03	1.708	89.173	0	10kHz

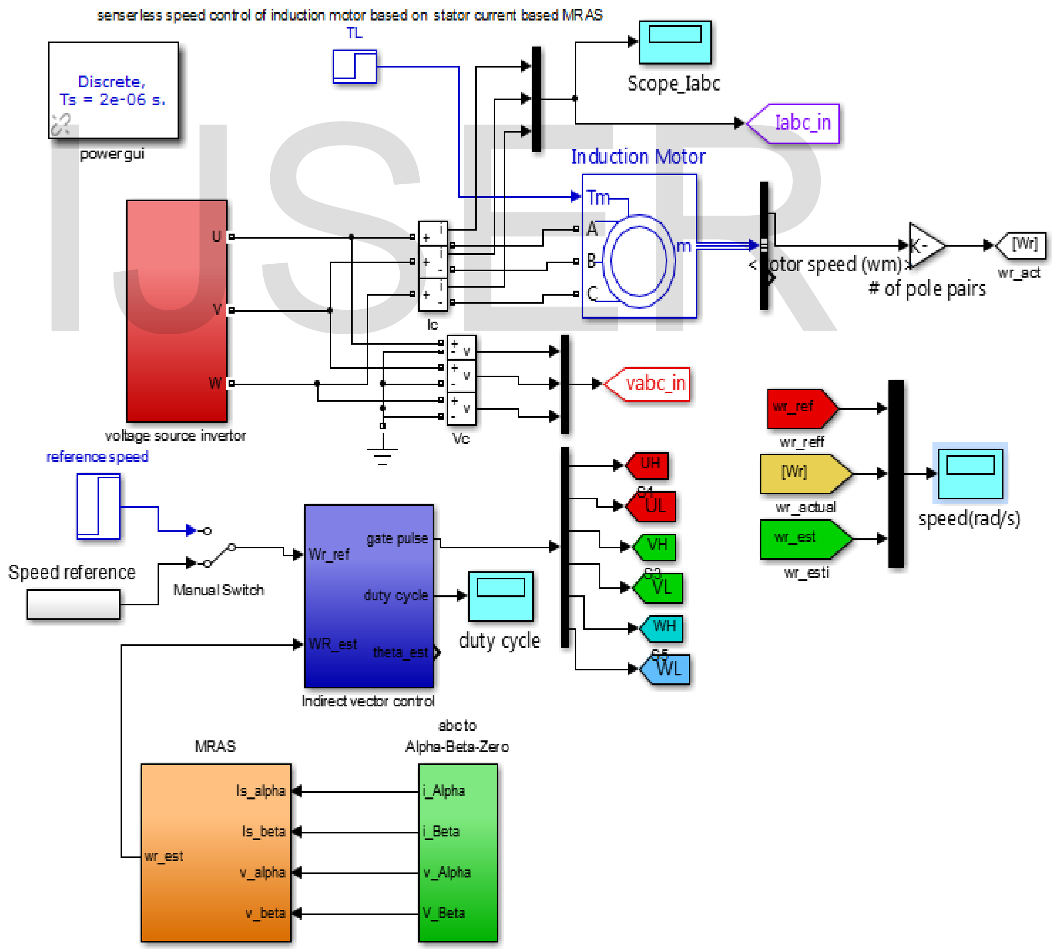


Figure 4.1 Sensorless Vector Control of IM using stator current based MRAS.

The proposed control scheme is simulated in discrete time. The induction motor as well as the inverter blocks are used in the design was already available in the standard Simulink library. The blocks that had to be constructed were the stator current based MRAS speed estimator and indirect vector control block seen in Figure 4.2 and Figure 4.3 respectively. The stator current based MRAS speed estimator calculates the rotor speed used for indirect field oriented control which control SVPWM, this used to drive the voltage source invertor.

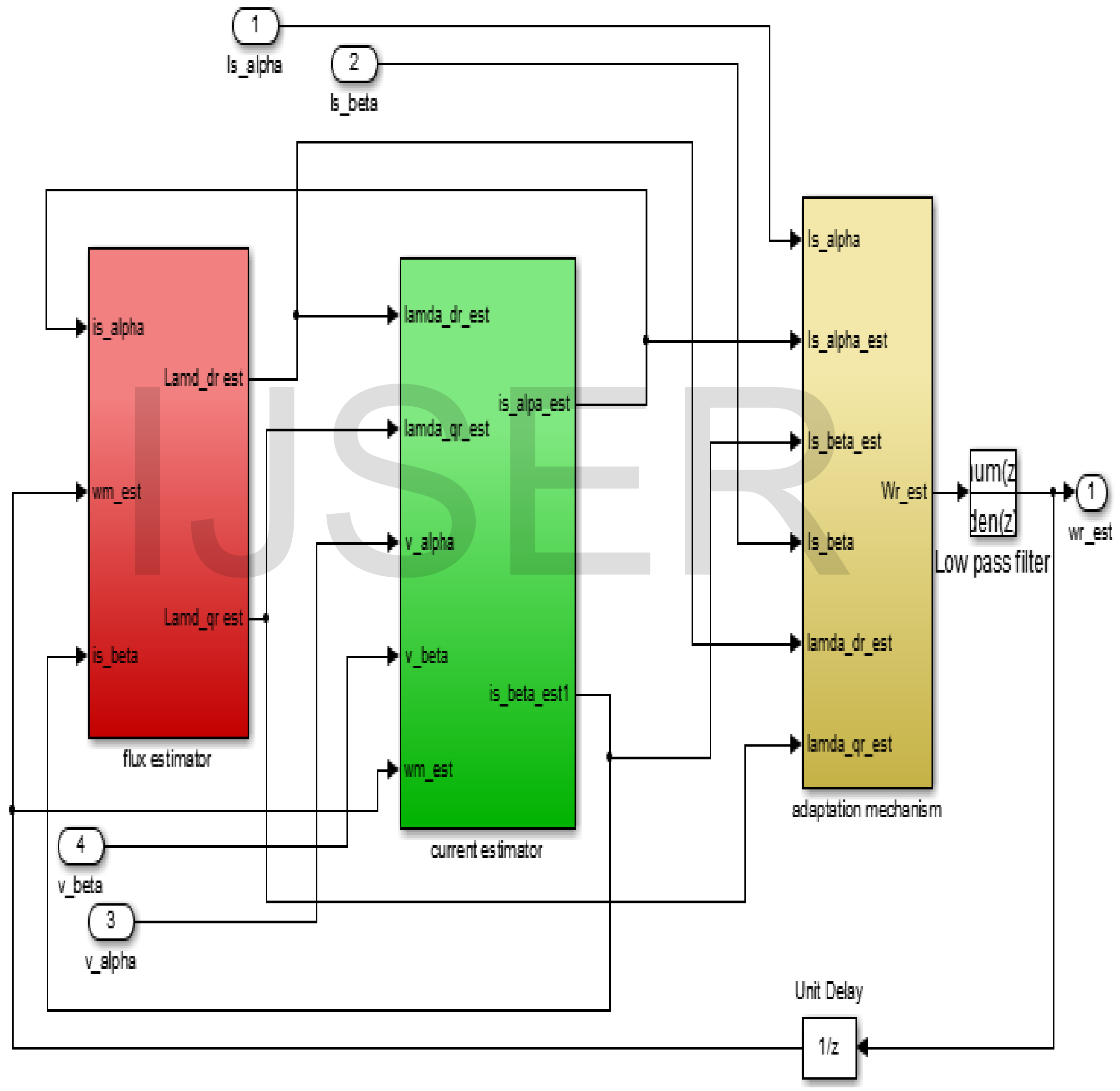


Figure 4. 2Stator current based MRAS speed estimator

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

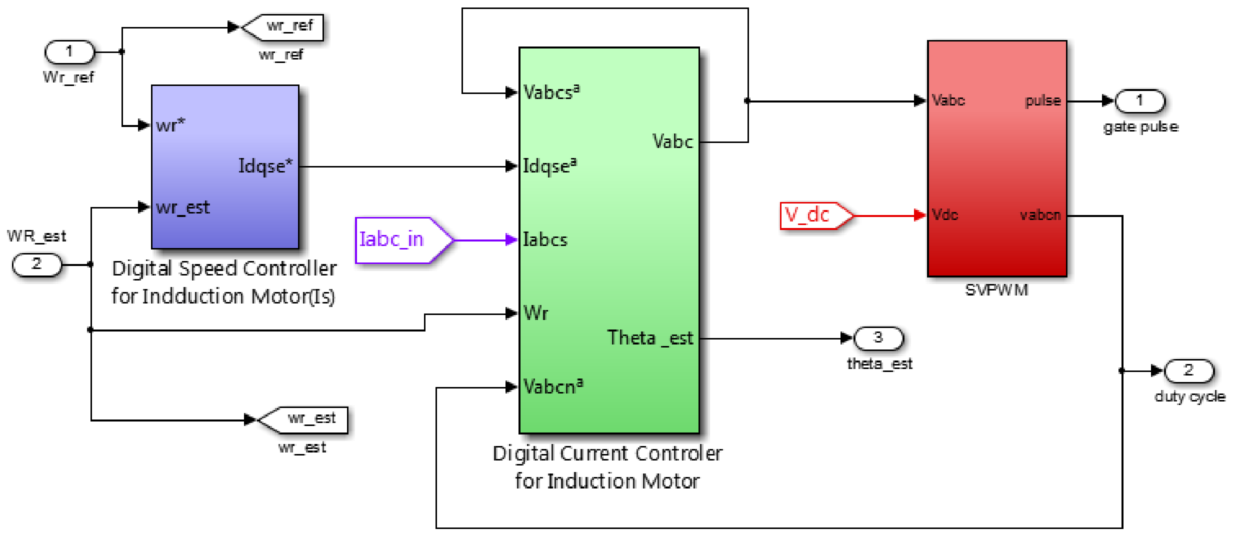


Figure 4.3 Indirect vector control of induction motor

4.3 Simulation Results

The simulation result of MRAS based sensorless speed control of induction motor drive was carried out to assess its performance. Knowledge of motor's parameter is important for this simulation since the estimator are highly parameter dependent and the effect of the parameter variation was tested based on different condition that are put on their effects on robustness of the speed control.

The first simulation result for MRAS based sensorless speed control of induction motor is the three phase stator current which is generated by the three phase voltage source inverter. This three phase voltage source inverter is controlled by SVPWM blocks for appropriate stator current generation. These three phases current should be equal magnitude and 120° phase shift with each other for appropriate rotating flux generation as shown in Figure 4.4.

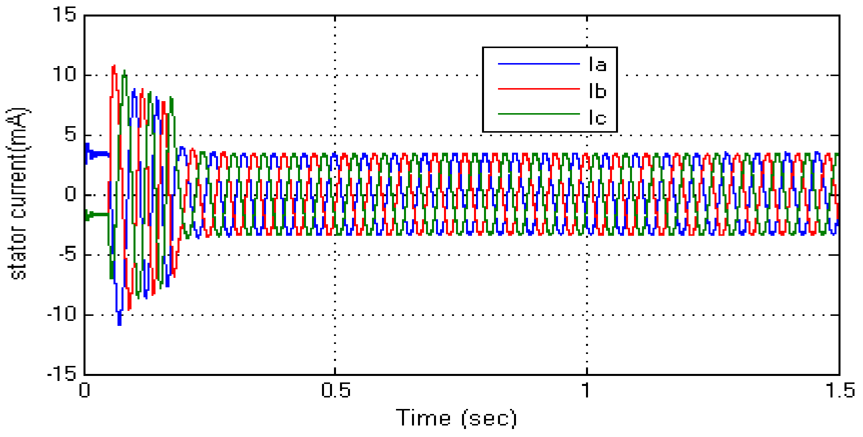


Figure 4.4 Three phase stator current at 100rad/s

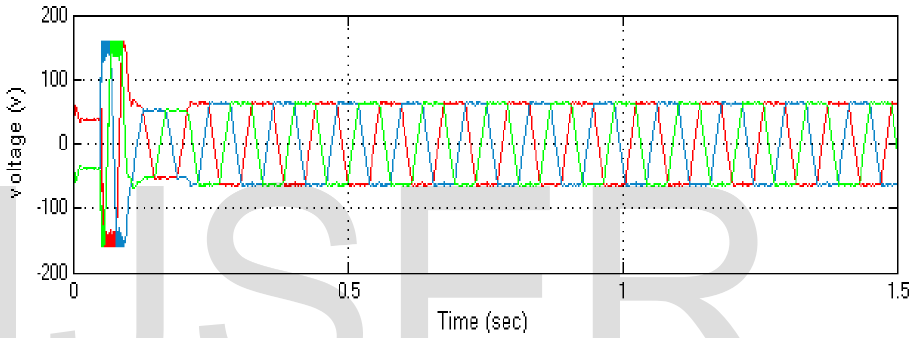


Figure 4.5 Duty cycle (Ta, Tb, Tc) at 50rad/s

As it has been seen from the Figure 4.4, the appropriate stator phase current is generated with good accuracy. Hence the system can feed the appropriate stator voltage to the motor. If the voltage applied to the motor is applied with appropriate magnitude and frequency, the speed of the motor is respected as set to the reference value.

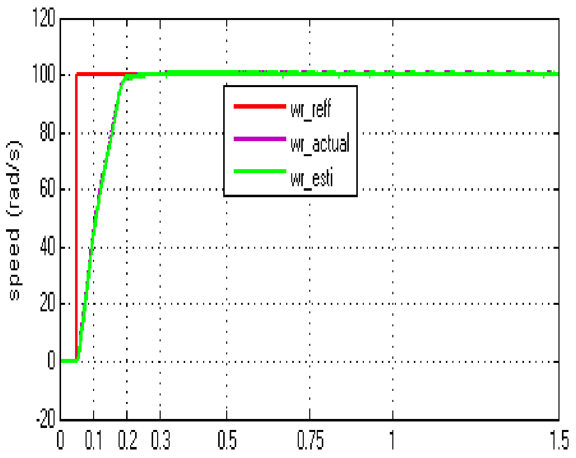
The simulation results of the proposed stator current based MRAS speed estimator for sensorless speed control of induction motor drive is discussed in terms of:

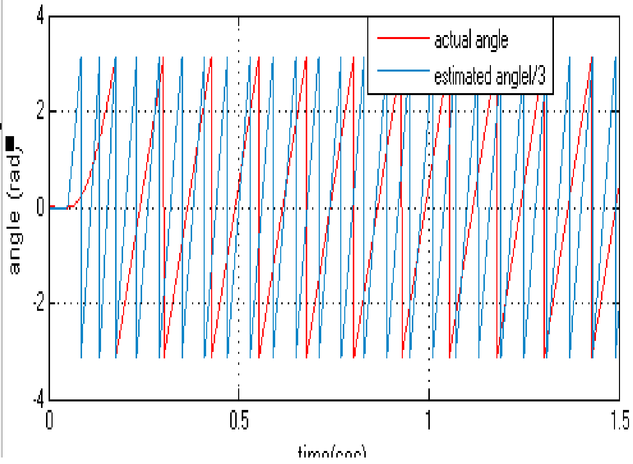
- Set point tracking capability.
- Torque response quickness.
- Low speed behavior.
- Step response of drive with speed reversal.
- Sensitivity to motor parameter uncertainty.
- Speed tracking ability in regenerative mode

Set point tracking capability

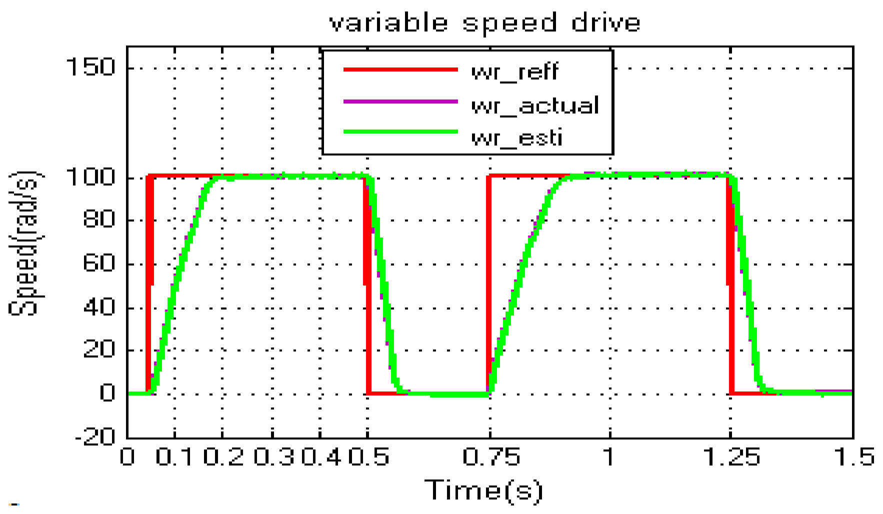
It is always crucial to know the performance of an estimator based on the ability of the estimated speed to converge the actual value, especially during the transient response. This criterion has been well accepted as a primary indicator when benchmarking the performance of a sensorless speed estimator. It shows the convergence of the estimated rotor speed to the actual speed. Using the same parameters in the induction motor and the stator current based MRAS speed estimator, the tracking performance of the estimator can be examined by changing the speed reference of the system. As shown in Figure 4.6 (a) and Figure 4.6 (c) the proposed estimator tracks both the step and square signals reference input. This shows the tracking performance of the estimator and actual speed to the reference speed can be examined by changing the reference of the system with the maximum steady state error of 0.0027% and good transient performance with rise time less than 0.1 second. As shown in Figure 4.6 (b) the estimated angle follows with the actual angle with the maximum error of 0.1rad.

In Figure 4.7 (a) variable speed driving of induction motor with step speed response, the actual and the estimated speed track the reference speed at every 0.13second. As shown in Figure 4.7(b) the actual speed and the estimated speed follows the reference with variable step input for 0.75 second and the input change to sine to check its tracking capability for different input. From this result, the propose system operates for different input with good accuracy of steady state and transient response.



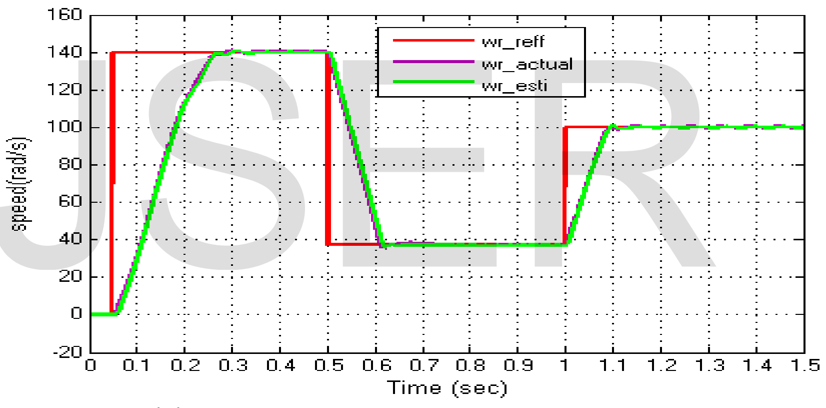


(a) step input speed response (100rad/s) with 2Nm load torque (b) Estimated and actual angle

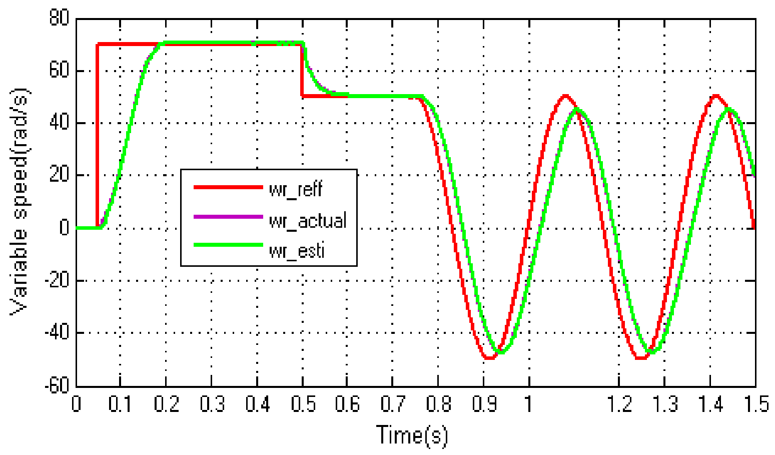


(c) Square input signal rotor speed response

Figure 4.6 Speed response for square and step reference speed in rad/s and the estimated angle.



(a) Variable step input rotor speed response



(b) Variable step input and sine signal rotor speed response

Figure 4.7 Variable Speed step response in rad/s with load torque of 2Nm

Torque response quickness

To find the torque response quickness the motor is stared with 1Nm load torque and this value is increased to 2Nm after 0.2 second this result a drop in motor speed. This happens because of the mismatch in the torques, i.e. the developed torque is less than the load torque. To compensate for this mismatch, the controller increases the developed torque then the motor speed increases and comes back to the set point as shown in Figure 4.8.

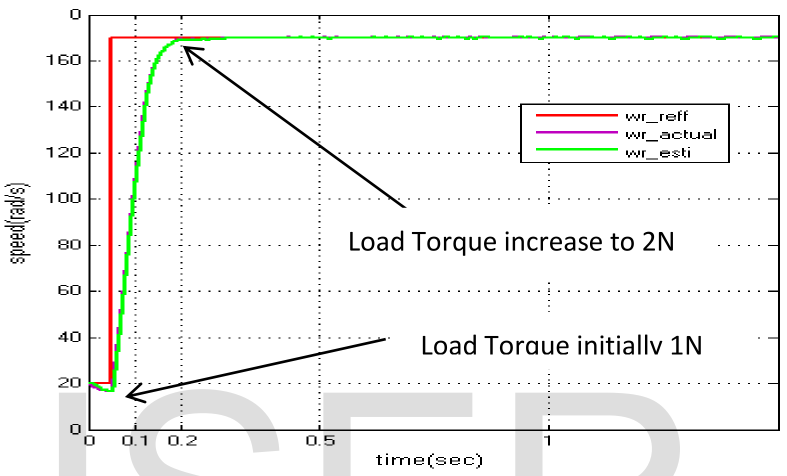
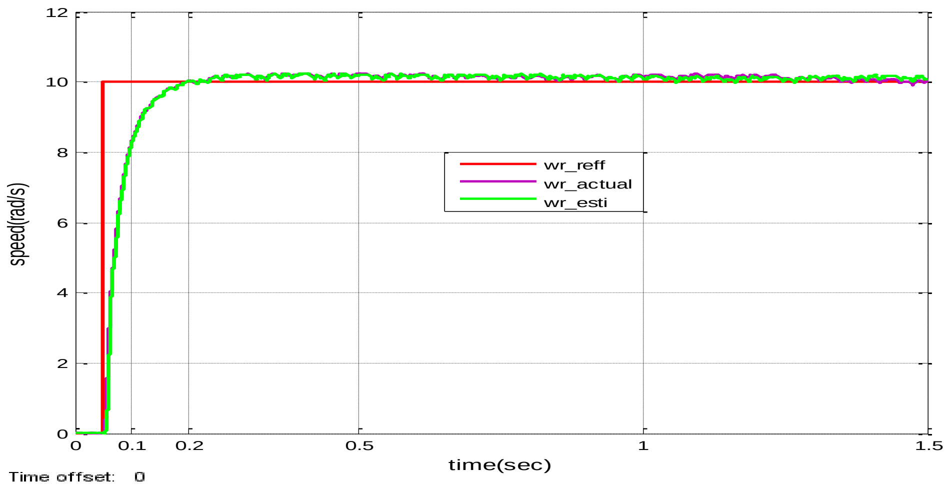


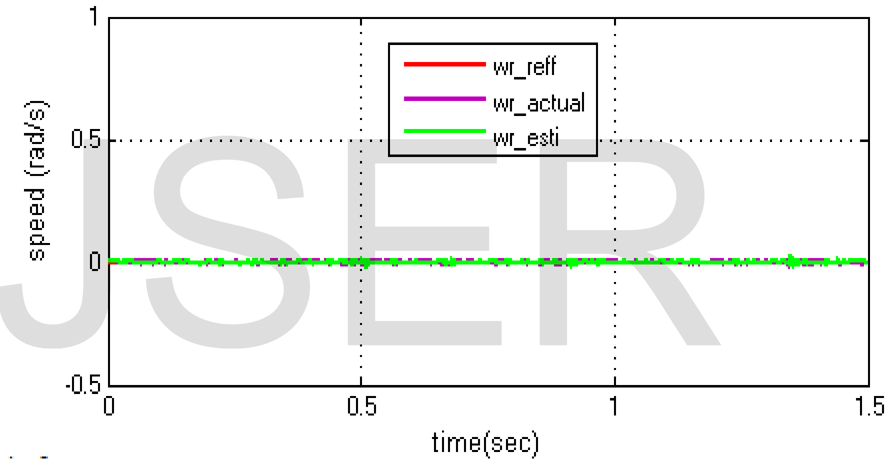
Figure 4.8 Load torque effect on stator current based MRAS.

Low and zero speed behavior

The aim of this test is to evaluate the performance of the stator current based MRAS speed estimator at low speed. Figure 4.9 (a) shows that the estimated speed follows the actual speed exactly and the reference speed very closely with steady state error of 0.024rad/s and good transient performance rise time less than 0.1 second. There is also good field orientation down to zero speed as shown in Figure 4.9(b). This means that the system is stable at zero speed and continuous operation is possible. There is short period the w_{r_act} and $\widehat{\omega}_r$ settle to their respective steady state values.



(a)The rotor speed response at 10rad/s step input signal



(b) The rotor speed response at 0.0rad/s step input signal

Figure 4.9 Step response for 10rad/s and zero rad/s speed response with no load torque

Step response of drive with speed reversal

Figure 4.10 shows that simulation result for speed reversal in step input. The motor reference speed is changed from 75rad/s to -35 rad/s at 0.5 second and again speed is set to 75 rad/s at 1 second. The result shows that the actual and estimated speed takes 0.09 second to follow the reference speed with good accuracy transient response. Reference speed and actual speeds are plotted in the same scale to observe the accuracy of stator current based MRAS based speed estimator.

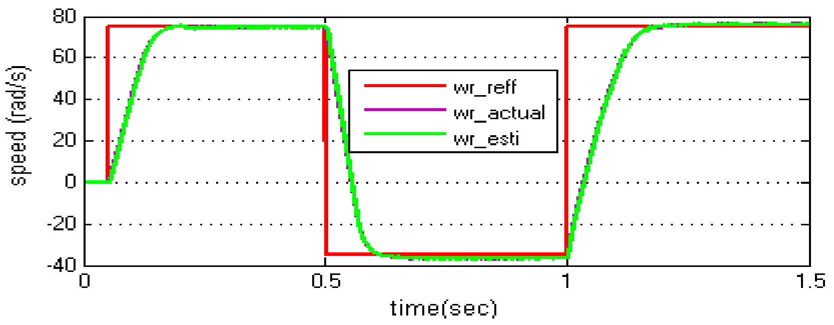


Figure 4.10 Step response of drive with speed reversal with load torque.

Parameter sensitivity

The sensitivity to the motor parameters change has been tested for three reference speeds, namely, 100%, 50% and 25% of the nominal speed; Parameters R_s, L_s, R_r, and L_r have been changed by $\pm$75% from their nominal value. As shown in Figure 4.11 the sensitivity to motor parameter changes of the sensorless field oriented control with stator current based MRAS speed estimator is less sensitive to motor parameter variation. As shown in Figure 4.12 when the motor parameters decrease by 75% of their nominal value the estimated speed exactly follow the actual speed and closely to the reference speed with good accuracy of transient with rise time less 0.1second with steady state error of less than 0.028rad/s. In Figure 4.11 the speed estimator is much less sensitive to the IM parameter changes at low speed and the sensitivity increases to some extent when the speed near to its nominal speed i.e.140 rad/s.

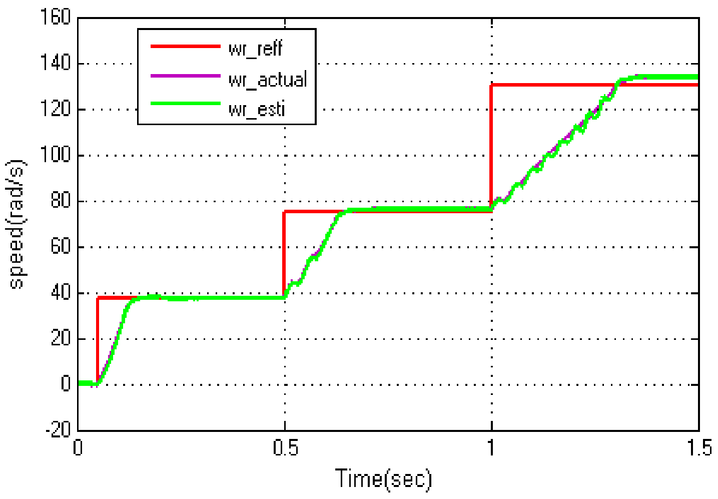


Figure 4.11 75% increasing of all parameters from their nominal value for different rotor speed

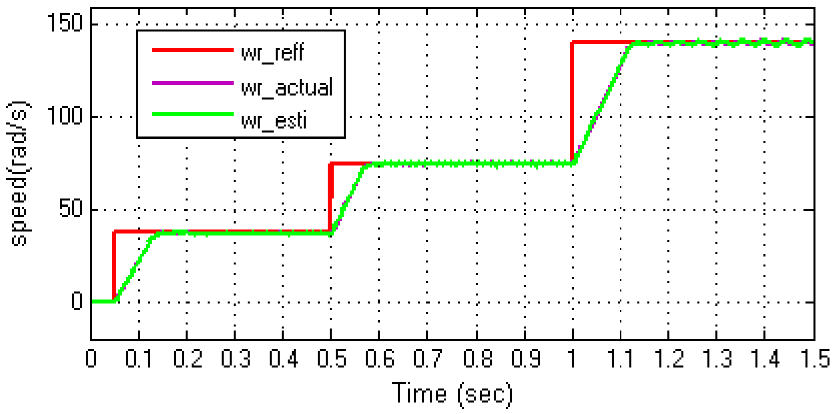


Figure 4.12 75% decreasing of all parameters from their nominal value for different rotor speed with load torque

The effect of stator and rotor resistance on stator current based MRAS speed estimator

Assume the rotor and stator resistance are vary increasing 50% and decreasing 50% with their nominal value and the other parameters are constant; with the speed 100%, 50% and 25% of the rated value. As shown in Figure 4.13 and Figure 4 15 when the stator resistance varies by $\pm$50% of its nominal value the estimated speed exactly follow the actual speed and closely to the reference speed with good accuracy of transient and steady state response. As shown in Figure 4.14 and Figure 4.16, when rotor resistance changes $\pm$50% from its nominal value it has good accuracy while the motor is running at low speed while the increasing the speed near to the nominal value settling time and rise time increases compare to the lower speed. From this result farther increasing of the rotor resistance may affect the motor speed while it runs near to its nominal speed.

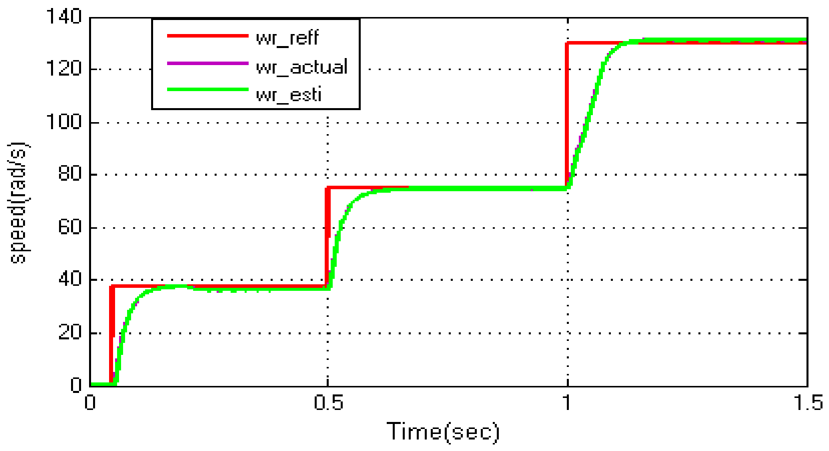


Figure 4.13 50% decreasing of Rs from its nominal value for different rotor speed

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

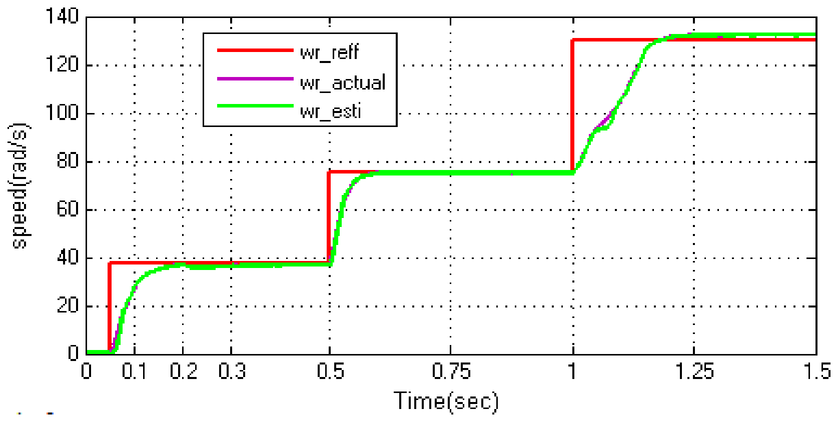


Figure 4. 14 50% decreasing of Rr from its nominal value for different rotor speed

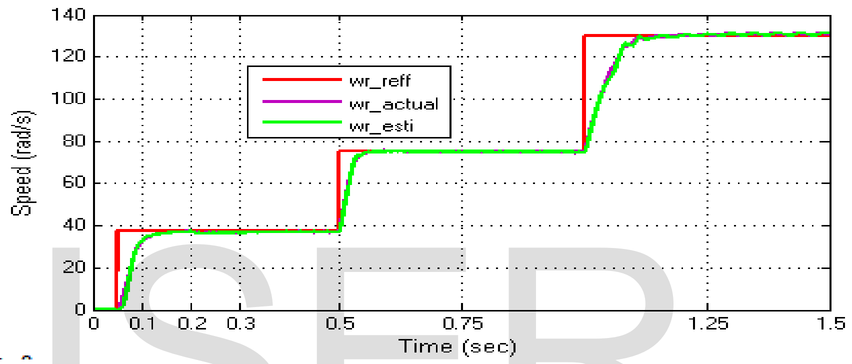


Figure 4.15 50% increasing of Rs from its nominal value for different rotor speed

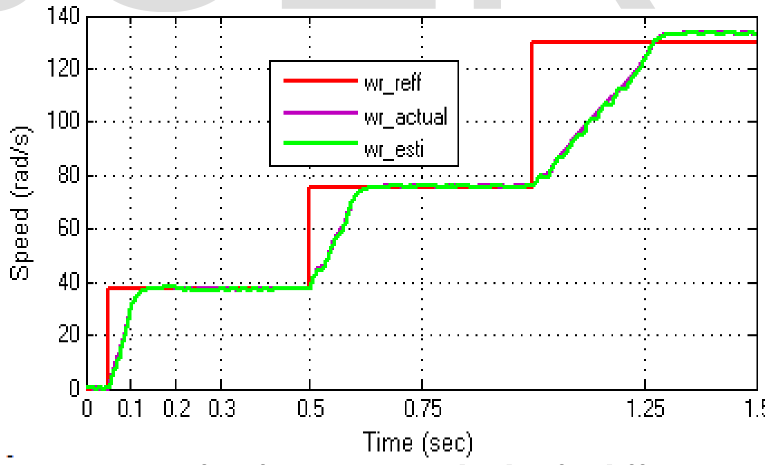


Figure 4.16 50% increasing of Rr from its nominal value for different rotor speed

Regenerative mode

Motor speed is changed from +50 rad/s to -50rad/s keeping the load torque constant at 2 Nm. The drive is operated in motoring mode up to 1second. Thereafter, it enters in the regenerating mode of operation for 0.5second and come back to motoring region after 0.5 second. As shown in Figure 4.17 the estimate and the actual speed follow the reference speed successfully.

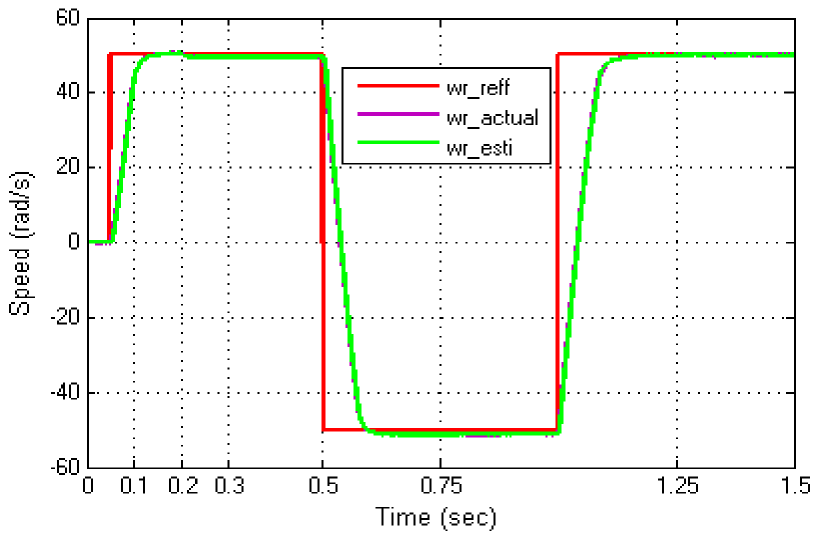
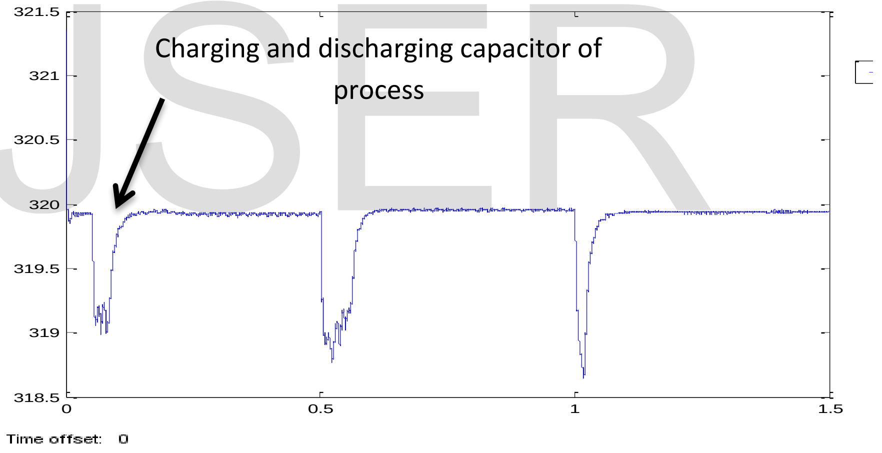
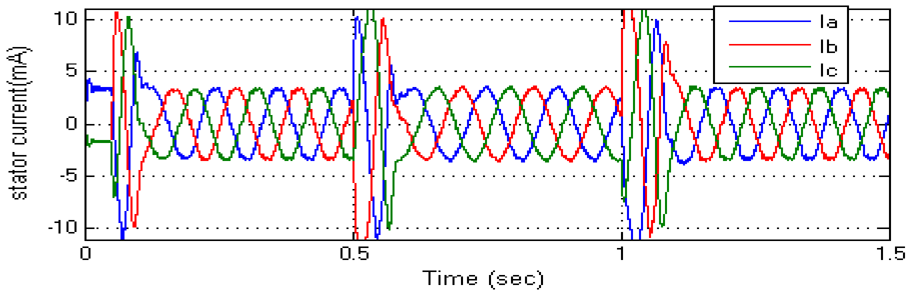


Figure 4.17 The actual and estimated speed performance in regenerative mode



(a) Dc bus voltage



(b)Three phase stator current at different rotor speed

Figure 4.18 DC link capacitor voltage and stator current at motoring and regenerating mode

From the Figure 4.18 the capacitor voltage decrease during the transient response due to the increase of stator current in that region ,i.e. for 0.05 second starting of the motoring region and at 0.5 second the operation enter to regenerating mode and finally it turn to back to motoring mode after one second. From this the capacitor DC link voltage drops and the stator current increases for transient response and turns to their rate value for the steady state response.

Staircase tracking waveform include low speed region

A reference speed of 5rad/s was initially applied at t = 0.05 seconds, increased to 25rad/s ,50rad/s and 100rad/s at t = 0.5 second, t=0.75 second and at t=1second respectively. As shown in Figure 4.19 the estimated and the actual speed follow the reference speed with good accuracy and it takes 0.09second to track the reference speed at different level of speed including low speed region.

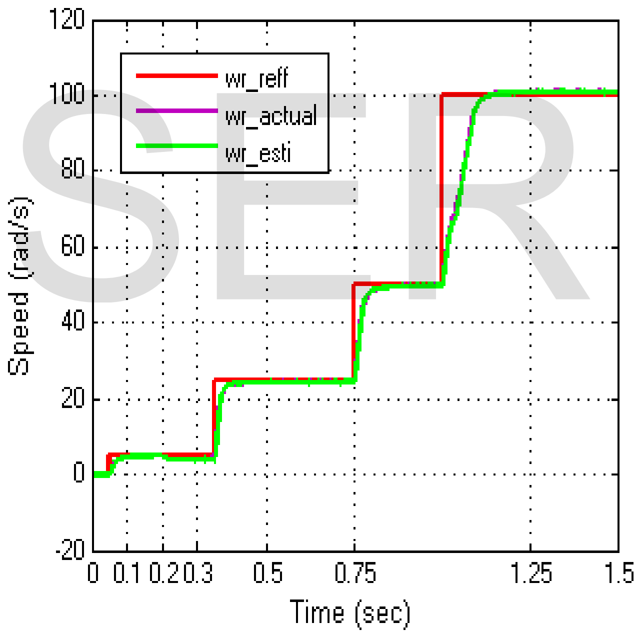


Figure 4.19 Speed response of staircase tracking waveform include low speed region

Chapter Five

Experimental Implementation

5.1 Introduction

To test the performance of the proposed scheme, experimental is carried out on the induction motor with the same parameters given in Table 4.1. Experimental setup of the induction motor control developed system is shown in Figure 5.6, which is based on a high voltage motor control developer's kit produced by Texas instruments Company. The PWM frequency for the experiment is 60 kHz and the ISR frequency 10 kHz is selected for best operation of the motor. Experimental data are captured for display and analysis via a graph tool in the code composer studio (CCS).

The overall experimental system contains both hardware and software. The hardware includes: High voltage motor control kit with TMS320F28035 DSP control card, three phase induction motor, PC with Code Composer Studio (CCSv6.0) installed, digital oscilloscope, digital multi-meter, JTAG probe, Rheostat and high voltage DC power supply. The software includes different coordinate transformation algorithm, controller's algorithm, stator current based MRAS speed estimator algorithm, space vector pulse width modulation algorithm and the kit interrupt software algorithm. The overall experimental block diagram contains mainly three modules: piccolo TMS320F28035 control card, High voltage motor control kit and induction motor as shown in Figure 5.1.

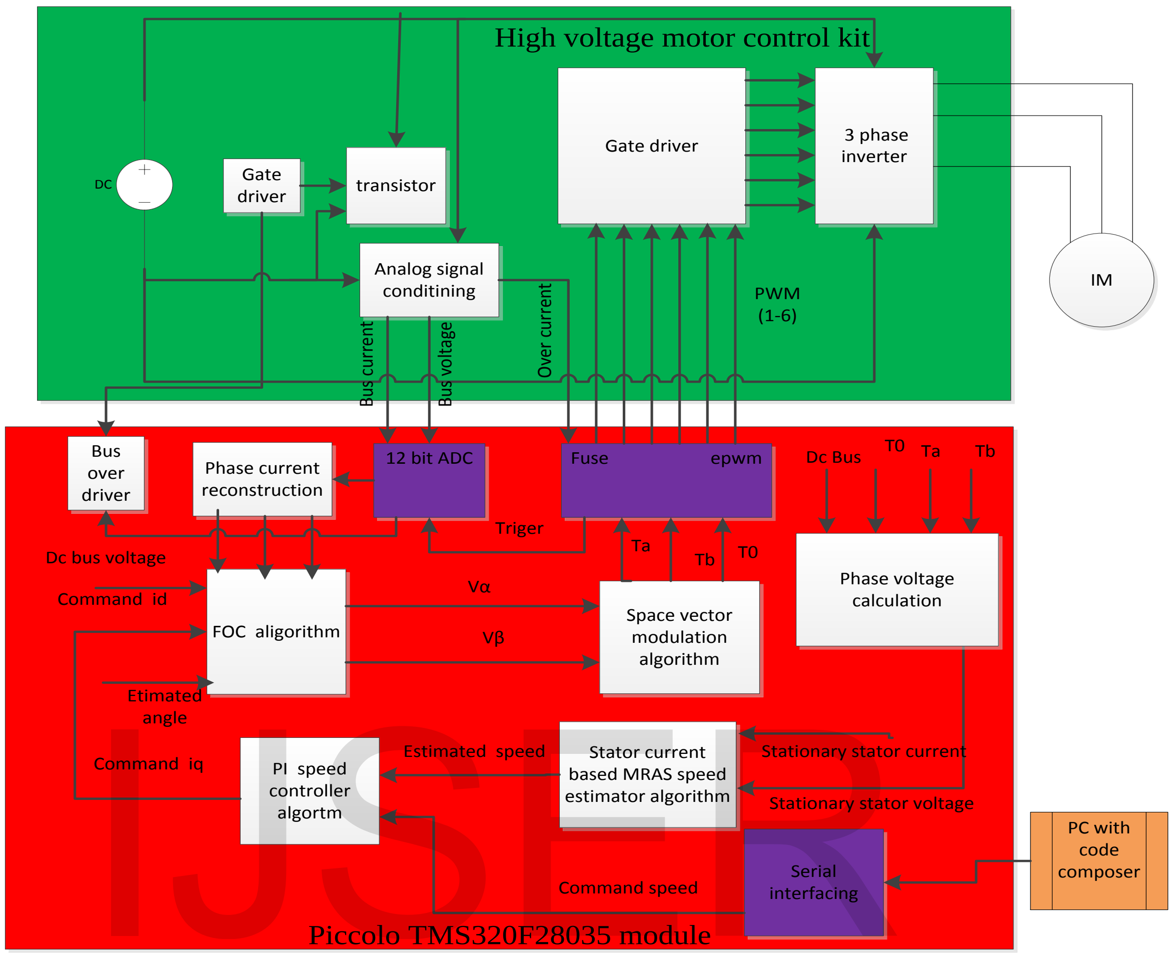

High voltage motor control kit
DC
Gate driver
transistor
Gate driver
3 phase inverter
IM
Analog signal conditining
PWM (1-6)
Bus current
Bus voltage
Over current
Bus over driver
Phase current reconstruction
12 bit ADC
Fuse
epwm
Dc Bus
T0
Ta
Tb
Triger
Dc bus voltage
Command id
FOC aligorithm
Vα
Vβ
Space vector modulation algorithm
Phase voltage calculation
Etimated angle
Command iq
PI speed controller algortm
Estimated speed
Stator current based MRAS speed estimator algorithm
Stationary stator current
Stationary stator voltage
Serial interfacing
PC with code composer
Command speed
Piccolo TMS320F28035 module

Figure 5.1 Experimental block diagram for MRAS based senserless speed control of IM

The digital motor control software library is the collection of digital motor control software modules. These modules allow users to quickly build or customize their own systems. The library supports the AC induction motor and comprises both peripheral dependent software drives and TMS320F28035 control card dependent modules.

As shown in Figure 5.1 the High voltage motor control kit module include: gate driver which control three phase voltage source inverter, analog signal conditioning circuit used to sense stator current and DC bus voltage, and three phase invertor to drive the motor. The second module is piccolo TMS320F28035 control card consist: control algorithms and hardware module. Control algorithms include like FOC, PI speed controller, phase current and voltage reconstruction, stator current based MRAS speed estimator and SVPWM algorithm. The hardware modules indicated by purple color in Figure 5.1.

5.2 Key Feature of HVMTRPFCKIT

The HVMTRPFC kit is standalone kit allowing evaluators to examine the TMS320F28035 control card to determine if it meets their application requirements. Furthermore, the module is an excellent platform to develop and run software from the TMS320F28035 processor. To simplify code development and shorten debugging time, a C2000 Tools Code Composer drive is provided. In addition, an on board JTAG connector provides interface to emulators, operating with other debuggers to provide C high level language debug.

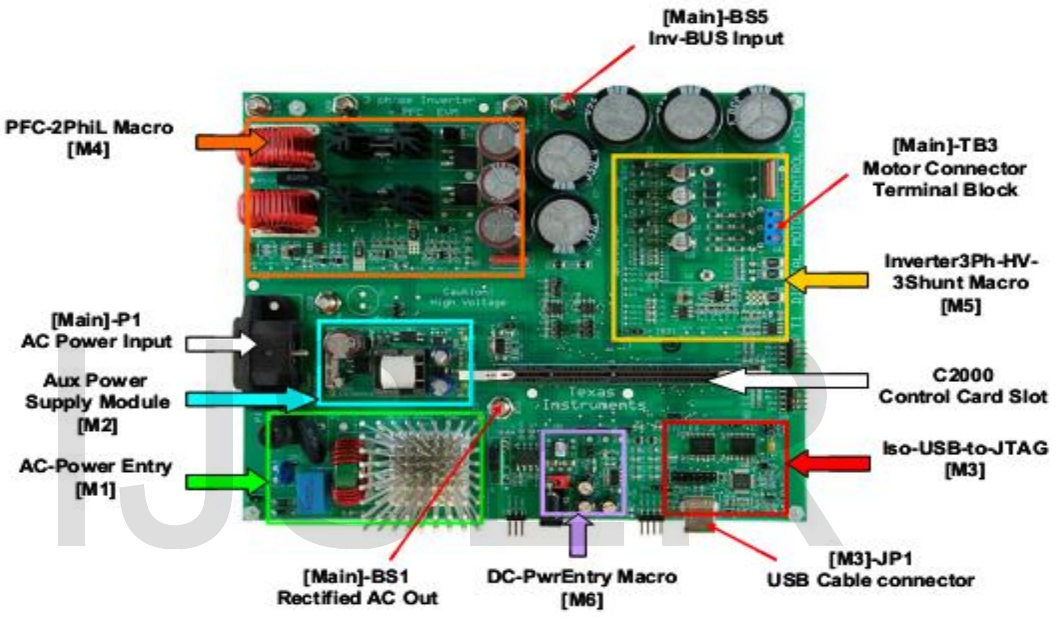


Figure 5.2 HVDMCMTRPFC kit board macros [34]

Key Features of the TMS320F28035 Piccolo™ Microcontrollers

In [34] the Piccolo TMS320F28035 ISO control card can be used as quick evaluation boards for the TMS320F28035, C2000 controller as well as a noise resistant plug-in card with isolate JTAG emulation build in to enable quick debug and bring up of boards requiring isolated emulator. All the key peripherals are brought out on the control card pins including the ADC, PWM, GPIO etc. Along with this an isolated emulator is on the control card itself, and can be connected to the host computer using a USB cable to connect to the debugger.

It has the following features:

- Small size – 90mm x 25mm (3.5 x 1)
- All GPIO, ADC and other key signals routed to gold connector fingers

- 5V input supply to the control card is needed for the controller. Extensive supply pin decoupling with L+C connected close to the device are provided on the control card
- Clamping diode protection at ADC input pins
- Anti-aliasing filter (noise filter) at ADC input pins
- Isolated JTAG Emulator though the USB connector
- Isolated Serial connection using USB Serial of the FTDI chip, though the USB connector
- The emulator drives its power from the USB connector and connection to the controller is isolated using digital isolators.

The major interfaces of TMS320F28035 are the JTAG (Joint Test Action Group) interface, analog input connector, I/O interface connector and expansion interface. JTAG connector uses to connect JTAG emulators to the DSP board. It comes in a variety of different ways to connect to the host PC. The block diagram shown in appendix C.

Hardware resource mapping[35]

The Figure 5.2 shows the various stage of the board in a block diagram format and illustrates the major connections and feedback values that are being mapped to the C2000 MCU and lists of these resources shown in Table 5.1.

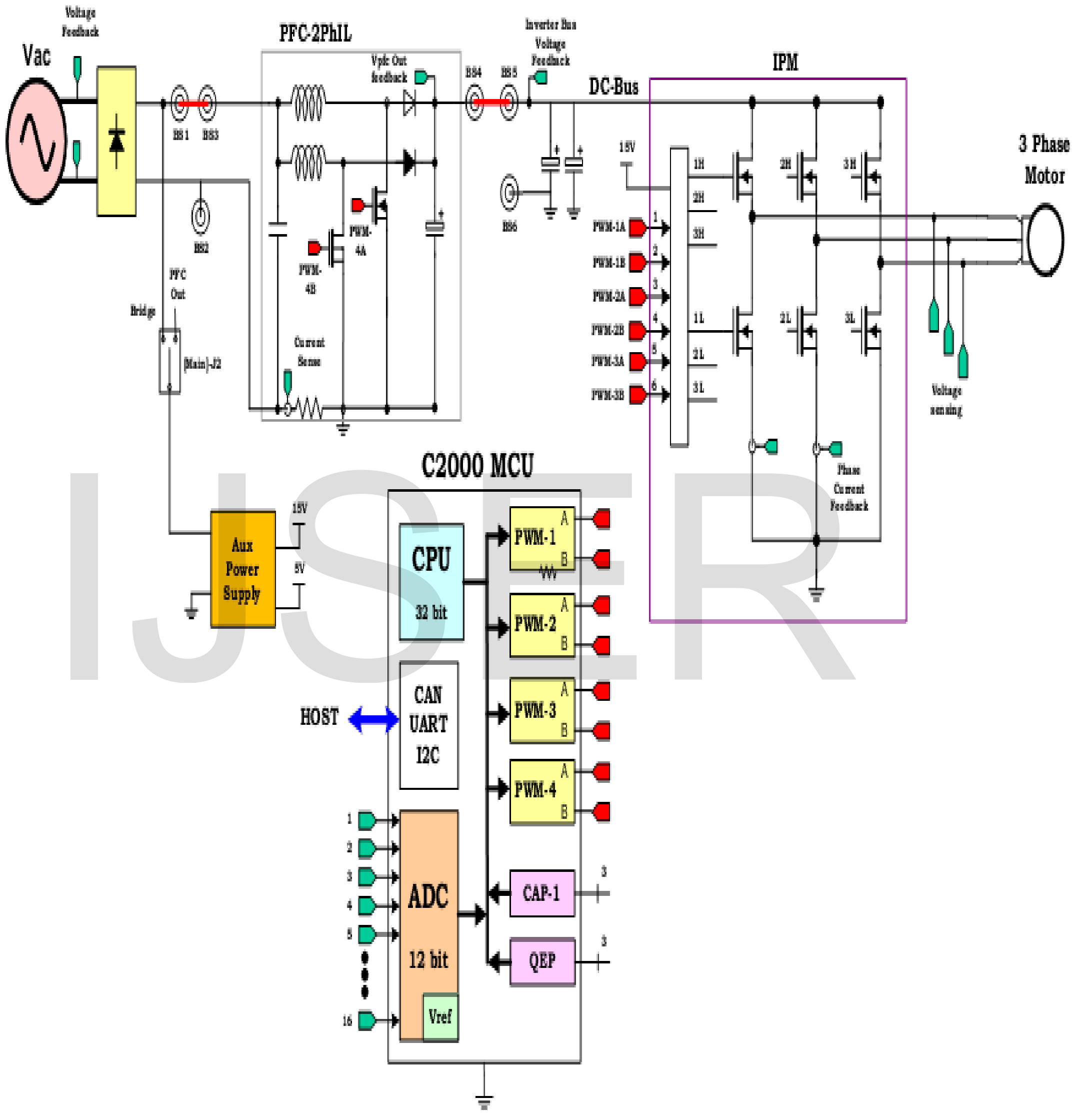


Figure 5.3 High Voltage DMC + PFC board diagram with C2000 MCU

Table 5.1 PWM and ADC resource allocation

Macro name	Signal name	PWM /ADC channel no mapping	Function
3-phase inverter	PWM -1H	PWM -1A	Inverter driver PWM
	PWM -1L	PWM -1B	Inverter driver PWM
	PWM -2H	PWM -2A	Inverter driver PWM
	PWM -2L	PWM -2B	Inverter driver PWM
	PWM -3H	PWM -3A	Inverter driver PWM
	PWM -3L	PWM -3B	Inverter driver PWM
	Ifb-U	ADC-B3, A1	Low side U-phase current sense
	Ifb-V	ADC-B5, B1	Low side V-phase current sense
	Ifb-W	ADC-A3, A5	Low side W-phase current sense
	Ifb-Sum	ADC-A2	DC Bus return current sense
	Vfb-Sum	ADC-A7	DC Bus Return voltage sense
	Vfb-U	ADC-B7	U- phase voltage sense
	Vfb-V	ADC-B6	V- phase voltage sense
	Vfb-W	ADC-B4	W- phase voltage sense
2 phase PFC	PWM -1	PWM -4A	PFC phase 1 drive PWM
	PWM -2	PWM -4B	PFC phase 2 drive PWM
	Ipfc	ADC-A6	Return current sense
	Vpfc	ADC-A4	PFC Output voltage sense
	L-fb	ADC-B2	Line voltage sense
	N-fb	ADC-B0	Neutral voltage sense
Main board	DAC -1	PWM -6A	Driving DAC signal
	DAC -2	PWM -6B	Driving DAC signal
	DAC-3	PWM -7A	Driving DAC signal
	DAC -4	PWM -7B	Driving DAC signal

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

5.3 System Algorithm for MRAS based sensorless speed control of IM

The overall system algorithm is based on two modules as shown in Figure 5.4. These are the initialization module and interrupt subroutine module.

Initialization module: After a DSP processor reset the initialization module performs the following task.

- DSP setup: core, clocks, ADC, GPIO, Event manager, etc.
- Variable initialization.
- Interrupt source selection and enable: it describes enabling the DSP modules to accept interrupt and from where the interrupt is originated.
- Waiting loop: It is a loop which waits for indefinite time length until the interrupt sub module sends the interrupt starting signal.

Interrupt subroutine module description: The interrupt module handles the whole field Oriented control algorithm. The goal of interrupt module is to update the stator voltage reference value and to ensure the regulation of stator currents and rotor mechanical speed to their reference value. It performs the following task.

- Reading ADC output current value and scaling up it: The stator current sensor output is connected to the ADC input of the digital signal processor. The DSP processor reads this current value from the ADC result register.
- Coordinate transformation: Different coordinate transformations for field oriented control are computed.
- Rotor speed estimation algorithm: It is a stator current based MRAS speed estimator algorithm.
- Speed and current control algorithm: It is Proportional Integral (PI) controller for controlling the speed and the current of the motor.
- SVPWM algorithm: it is an algorithm used to generate a nearly sinusoidal stator current by the inverter using q-axis and d-axis stator voltages.

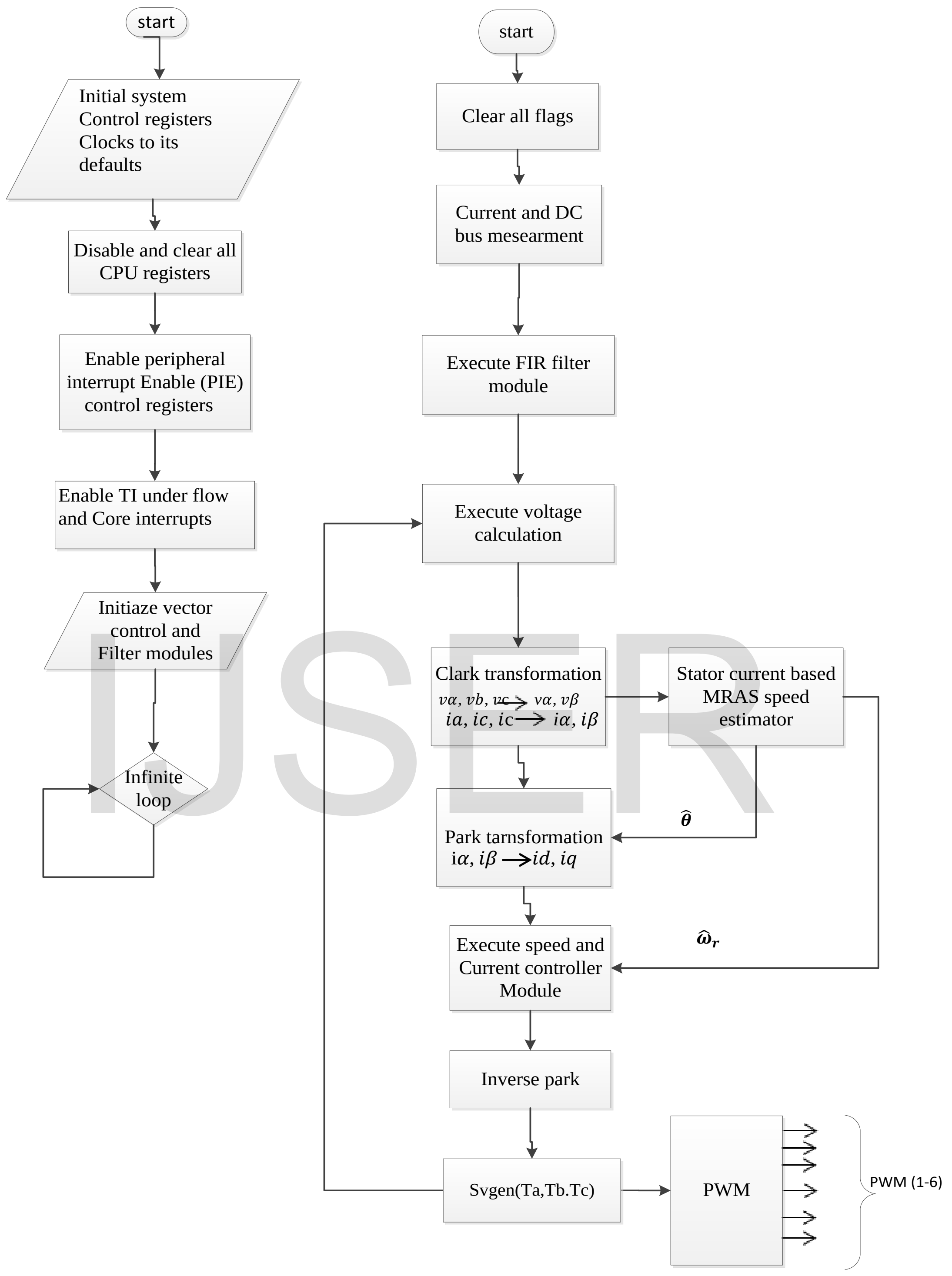

start
Initial system Control registers Clocks to its defaults
Disable and clear all CPU registers
Enable peripheral interrupt Enable (PIE) control registers
Enable TI under flow and Core interrupts
Initiaze vector control and Filter modules
Infinite loop
start
Clear all flags
Current and DC bus mesearment
Execute FIR filter module
Execute voltage calculation
Clark transformation
$v\alpha, vb, vc \rightarrow v\alpha, v\beta$
$i\alpha, ic, ic \rightarrow i\alpha, i\beta$
Stator current based MRAS speed estimator
$\hat{\theta}$
Park tarnsformation
$i\alpha, i\beta \rightarrow id, iq$
$\hat{\omega}_r$
Execute speed and Current controller Module
Inverse park
Svgen(Ta,Tb.Tc)
PWM
PWM (1-6)

Figure 5.4 Flowchart of the proposed system

Stator current based MRAS speed estimator algorithm has been written by using prediction correction method.

Predictor Corrector Method (PECEC)

Predictor Corrector method is sometimes called PECEC because it consists of five steps (Prediction – Evaluation – Correction – Evaluation – Correction).The process of computation consists of these steps:

1. Prediction – estimation of result by using Euler method.
2. Evaluation – computing of next correction member from previous result.
3. Correction – correction of the result from point 1.
4. Evaluation – computing of next correction member from previous result.
5. Correction – correction of the result from point 3.

The stator current based MRAS speed estimator algorithm is described mathematically as follows:

$$\left.\begin{aligned}
\frac{d}{dt}\hat{\varphi}_{r\alpha} &= \frac{\hat{\varphi}_{r\alpha(k+1)} - \hat{\varphi}_{r\alpha(k)}}{h} \\
\frac{d}{dt}\hat{\varphi}_{r\beta} &= \frac{\hat{\varphi}_{r\beta(k+1)} - \hat{\varphi}_{r\beta(k)}}{h} \\
\frac{d}{dt}\hat{\imath}_{s\alpha} &= \frac{\hat{\imath}_{s\alpha(k+1)} - \hat{\imath}_{s\alpha(k)}}{h} \\
\frac{d}{dt}\hat{\imath}_{s\beta} &= \frac{\hat{\imath}_{s\beta(k+1)} - \hat{\imath}_{s\beta(k)}}{h} \\
\frac{d}{dt}\hat{\omega}_{r} &= \frac{\hat{\omega}_{r(k+1)} - \hat{\omega}_{r(k)}}{h}
\end{aligned}\right\} \tag{5.1}$$

Where, h= step size sampling time

Predictor corrector methods apply for stator current and rotor flux estimation are as follows.

$T_r = \frac{L_r}{R_r}$

$\alpha = \frac{R_r}{L_r}$

$\gamma = \frac{(L_m{}^2 R_r + L_r{}^2 R_s)}{(\sigma L_s L_r{}^2)}$

$\sigma = 1 - \frac{L_m^2}{L_r L_s}$

$\beta = \frac{L_m}{\sigma L_r L_s}$

$K_1 = T_s \alpha$

$K_2 = T_s \omega_b$

$K_3 = T_s h L_m \left(\frac{I_b}{L_b}\right)$

$K_4 = T_s \alpha\beta\left(\frac{L_b}{I_b}\right)$

$K_5 = T_s \beta \frac{(L_b \omega_b)}{I_b}$

$K_6 = T_s\, \gamma$

$K_7 = T_s \left(\frac{1}{\sigma L_s}\right)\frac{V_b}{I_b}$

$K_8 = \frac{3}{2}(\frac{P}{2})(\frac{L_m}{L_r})(L_b\frac{I_b}{T_b})$ $\qquad K_{10} = T_s(\frac{P}{2})(\frac{1}{J})(\frac{I_b}{\omega_b})$

$K_9 = (T_s(\frac{\beta}{J}))$ $\qquad K_s = (\frac{1}{\omega_b T_r})$

Using corrector predictor method

First step:

$$\left.\begin{aligned}
\varphi_{r\alpha(k+1)} &= \hat{\varphi}_{r\alpha(k)} - K_1\hat{\varphi}_{r\alpha(k)} + K_2\hat{\omega}_r\hat{\varphi}_{r\beta(k)} + K_3\hat{I}_{\alpha(k)}\\
\varphi_{r\beta(k+1)} &= \hat{\varphi}_{r\beta(k)} - K_1\hat{\varphi}_{r\beta(k)} + K_2\hat{\omega}_r\hat{\varphi}_{r\alpha(k)} + K_3\hat{I}_{\beta(k)}\\
I_{\beta(k+1)} &= \hat{I}_{\beta(k)} + K_4\hat{\varphi}_{r\beta(k)} - K_5\hat{\omega}_r\hat{\varphi}_{r\alpha(k)} - K_6\hat{I}_{\beta} + K_7\hat{v}_{\beta}\\
I_{\alpha(k+1)} &= \hat{I}_{\alpha(k)} + K_4\hat{\varphi}_{r\alpha(k)} + K_5\hat{\omega}_r\hat{\varphi}_{r\beta(k)} - K_6\hat{I}_{\alpha(k)} + K_7\hat{v}_{\alpha}
\end{aligned}\right\} \tag{5.1}$$

Second:

$$\left.\begin{aligned}
\hat{\varphi}_{r\alpha(k+1)} &= - K_1\varphi_{r\alpha(k+1)} - K_2\hat{\omega}_r\varphi_{r\beta(k+1)} + K_3 I_{\alpha(k+1)}\\
\hat{\varphi}_{r\beta(k+1)} &= - K_1\varphi_{r\beta(k+1)} + K_2\hat{\omega}_r\varphi_{r\alpha(k+1)} + K_3 I_{\beta(k+1)}\\
\hat{I}_{\alpha(k+1)} &= K_4\varphi_{r\alpha(k+1)} + K_5\hat{\omega}_r\varphi_{r\beta(k+1)} - K_6 I_{\alpha(k+1)} + K_7\hat{v}_{\alpha}\\
\hat{I}_{\beta(k+1)} &= K_4\varphi_{r\beta(k+1)} - K_5\hat{\omega}_r\varphi_{r\alpha(k+1)} - K_6 I_{\beta(k+1)} + K_7\hat{v}_{\beta}
\end{aligned}\right\} \tag{5.2}$$

Take the change between predictor term and predictor estimated.

$$\left.\begin{aligned}
\Delta\varphi_{r\alpha(k)} &= \varphi_{r\alpha(k+1)} - \hat{\varphi}_{r\alpha(k)}\\
\Delta\varphi_{r\beta(k)} &= \varphi_{r\beta(k+1)} - \hat{\varphi}_{r\beta(k)}\\
\Delta I_{s\alpha(k)} &= I_{s\alpha(k+1)} - \hat{I}_{s\alpha(k)}\\
\Delta I_{s\beta(k)} &= I_{s\beta(k+1)} - \hat{I}_{s\beta(k)}
\end{aligned}\right\} \tag{5.3}$$

Third:

$$\left.\begin{aligned}
\hat{\varphi}_{r\alpha(k)} &= \hat{\varphi}_{r\alpha(k)} + \frac{(1+h)}{2}\hat{\varphi}_{r\alpha(k+1)} + \frac{(1-h)}{2}\Delta\varphi_{r\alpha(k)}\\
\hat{\varphi}_{r\beta(k)} &= \hat{\varphi}_{r\beta(k)} + \frac{(1+h)}{2}\hat{\varphi}_{r\beta(k+1)} + \frac{(1-h)}{2}\Delta\varphi_{r\beta(k)}\\
\hat{I}_{r\alpha(k)} &= \hat{I}_{r\alpha(k)} + \frac{(1+h)}{2}\hat{I}_{r\alpha(k+1)} + \frac{(1-h)}{2}\Delta I_{r\alpha(k)}\\
\hat{I}_{r\beta(k)} &= \hat{I}_{r\beta(k)} + \frac{(1+h)}{2}\hat{I}_{r\beta(k+1)} + \frac{(1-h)}{2}\Delta I_{r\beta(k)}
\end{aligned}\right\} \tag{5.4}$$

Electromagnetic torque:

$$T_e = K_8[\hat{\varphi}_{r\alpha(k)}\hat{I}_{\beta(k)} - \hat{\varphi}_{r\beta(k)}\hat{I}_{\alpha(k)}] \tag{5.5}$$

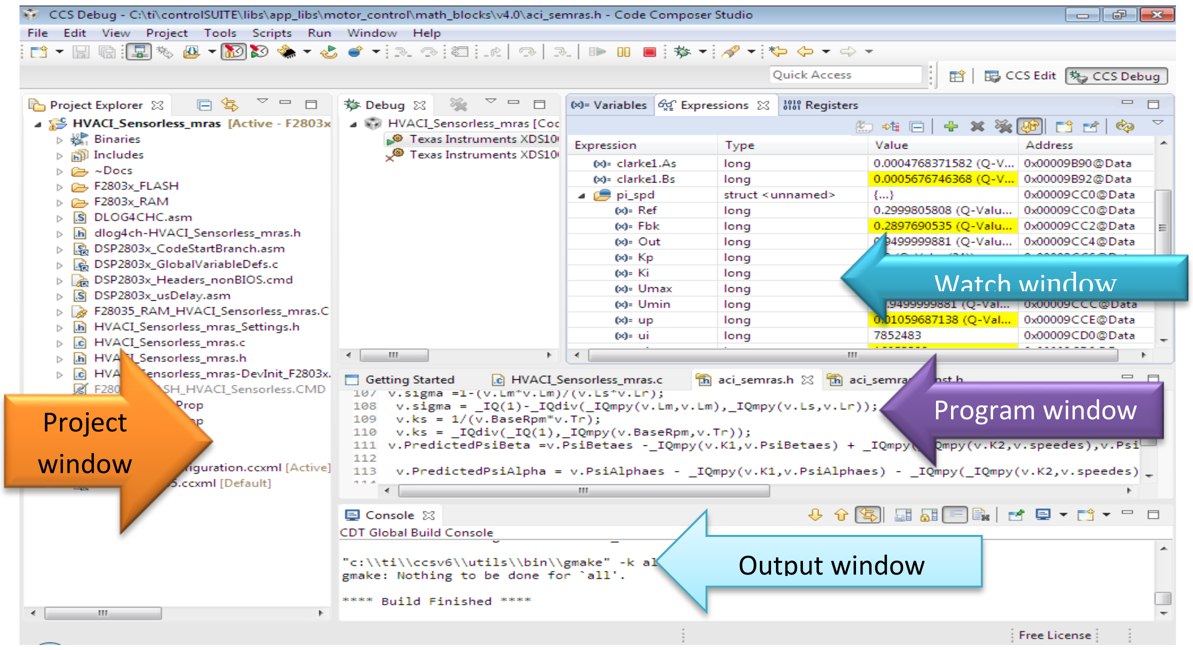
For rotor speed:

$$\left.\begin{aligned}
\omega_{r(k+1)} &= \omega_{r(k)} - K_9\omega_{r(k)} + K_{10}[T_e - T_l\,] \\
\widehat{\omega}_{r(k+1)} &= -K_9\omega_{r(k)} + K_{10}[T_e - T_l\,] \\
\Delta\omega_{r(k)} &= \omega_{r(k+1)} - \omega_{r(k)} \\
\widehat{\omega}_{r(k)} &= \widehat{\omega}_{r(k)} + \frac{(1+h)}{2}\widehat{\omega}_{r(k+1)} + \frac{(1-h)}{2}\Delta\omega_{r(k)}
\end{aligned}\right\} \qquad (5.6)$$

Using the Equation (5.1) to Equation (5.5) the stator current based MRAS speed estimator "C" code is written (Appendix A)

Code Composer Studio (CCS)

The code development is carried out on an application suite called code composer studio (CCS v6.0) and made by Texas instruments. It supports all their families and variations of DSP. The key feature of the CCS is to serve the whole development chain and supports C and Assembly programming language. In Figure 5.4 the outline of the programming interface is stated. CCS consists of four main parts: project window, program window, watch window, and output window. When choosing the compile command, potential errors will show up in the Output window. The Watch window enables tracking of variable values while running the program. The CCS communicates with DSP through the computer standard parallel port.


CCS Debug - C:\ti\controlSUITE\libs\app_libs\motor_control\math_blocks\v4.0\aci_semras.h - Code Composer Studio
File Edit View Project Tools Scripts Run Window Help
Quick Access
CCS Edit
CCS Debug
Project Explorer
HVACI_Sensorless_mras [Active - F2803x
Binaries
Includes
~Docs
F2803x_FLASH
F2803x_RAM
DLOG4CHC.asm
dlog4ch-HVACI_Sensorless_mras.h
DSP2803x_CodeStartBranch.asm
DSP2803x_GlobalVariableDefs.c
DSP2803x_Headers_nonBIOS.cmd
DSP2803x_usDelay.asm
F28035_RAM_HVACI_Sensorless_mras.C
HVACI_Sensorless_mras_Settings.h
HVACI_Sensorless_mras.c
Debug
HVACI_Sensorless_mras [Coc
Texas Instruments XDS10
Texas Instruments XDS10
Variables
Expressions
Registers
Getting Started
HVACI_Sensorless_mras.c
aci_semras.h
Console
CDT Global Build Console
Free License
Project window
Watch window
Program window
Output window

Expression	Type	Value	Address
clarke1.As	long	0.0004768371582 (Q-V...	0x00009B90@Data
clarke1.Bs	long	0.0005676746368 (Q-V...	0x00009B92@Data
pi_spd	struct <unnamed>	{...}	0x00009CC0@Data
Ref	long	0.2999805808 (Q-Valu...	0x00009CC0@Data
Fbk	long	0.2897690535 (Q-Valu...	0x00009CC2@Data
Out	long	9499999881 (Q-Valu...	0x00009CC4@Data
Kp	long		
Ki	long		
Umax	long		
Umin	long	.9499999881 (Q-Val...	0x00009CCC@Data
up	long	0.01059687138 (Q-Val...	0x00009CCE@Data
ui	long	7852483	0x00009CD0@Data

```
v.sigma =1-(v.Lm*v.Lm)/(v.Ls*v.Lr);
 v.sigma = _IQ(1)-_IQdiv(_IQmpy(v.Lm,v.Lm),_IQmpy(v.Ls,v.Lr));
 v.ks = 1/(v.BaseRpm*v.Tr);
 v.ks = _IQdiv(_IQ(1),_IQmpy(v.BaseRpm,v.Tr));
v.PredictedPsiBeta =v.PsiBetaes -_IQmpy(v.K1,v.PsiBetaes) + _IQmpy(  Qmpy(v.K2,v.speedes),v.Psi

 v.PredictedPsiAlpha = v.PsiAlphaes - _IQmpy(v.K1,v.PsiAlphaes) - _IQmpy(_IQmpy(v.K2,v.speedes)
```

```
"c:\\ti\\ccsv6\\utils\\bin\\gmake" -k al
gmake: Nothing to be done for `all'.

**** Build Finished ****
```

Figure 5.5 Snapshot of CCS programming interface while the motor is running at 0.3pu

5.4 Experimental setup

The experimental setup of MRAS based sensorless speed control of induction motor is shown in Figure5.6.

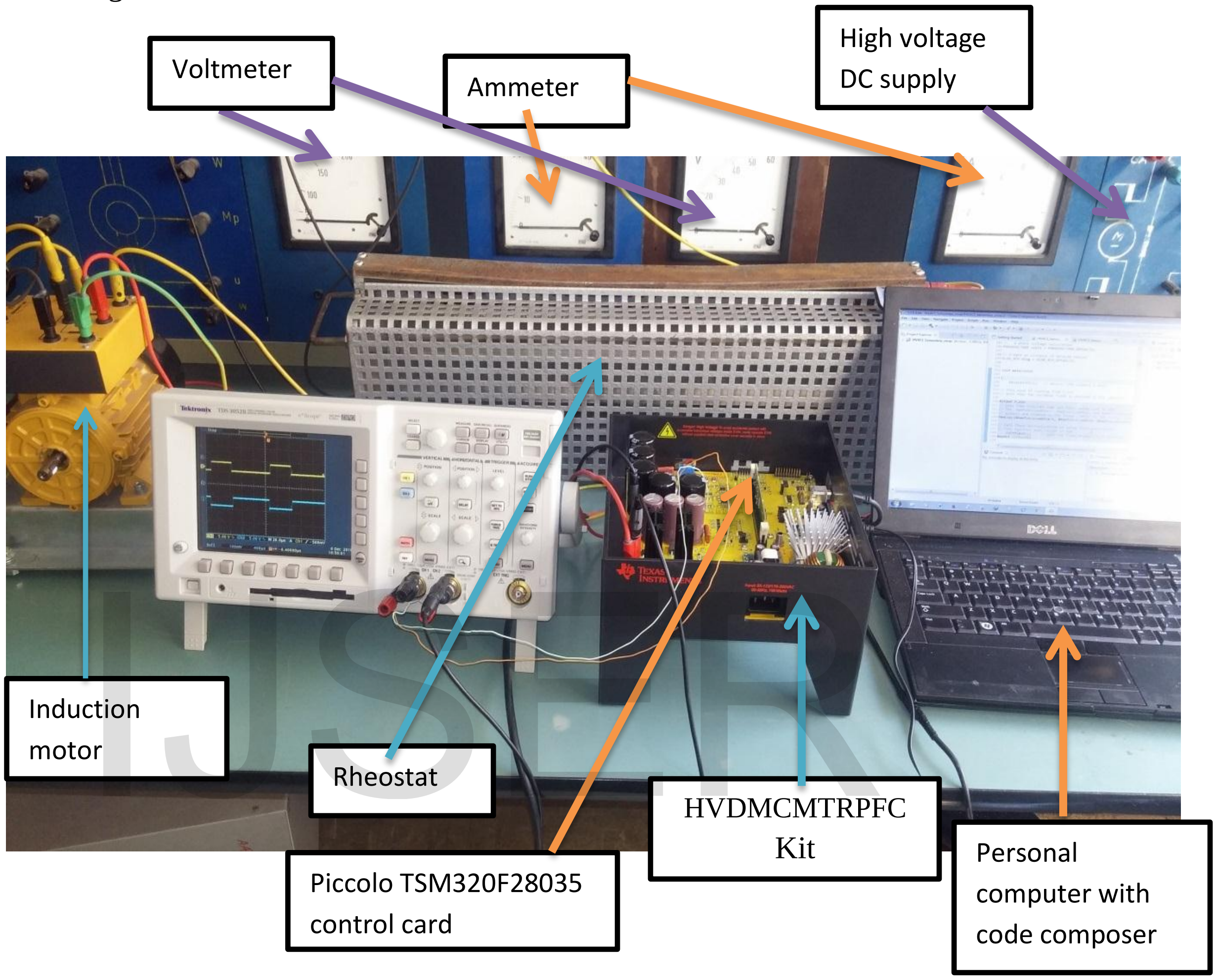


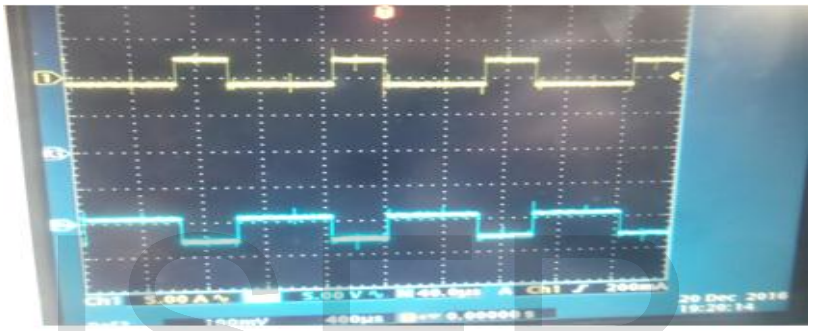
Figure 5.6 Experimental setup of the proposed system

The experimental setup includes the host computer which is used to program the system algorithm through code composer studio v6.0.The HVDMCMTRPFC kit is connected with the host computer by USB cable through the FTDI driver. The source code on the host computer is debugged and loaded to the memory of the piccolo TMS320F28035 control card through this cable. Controller Power comprises of the 15V, 5Vand 3.3V that the board uses to power the control card and the logic and sensing circuit present on the board. This power can be sourced from Auxiliary Power supply module [M2]. Auxiliary Power supply module [M2] can generate 15V and 5V DC from rectified AC.

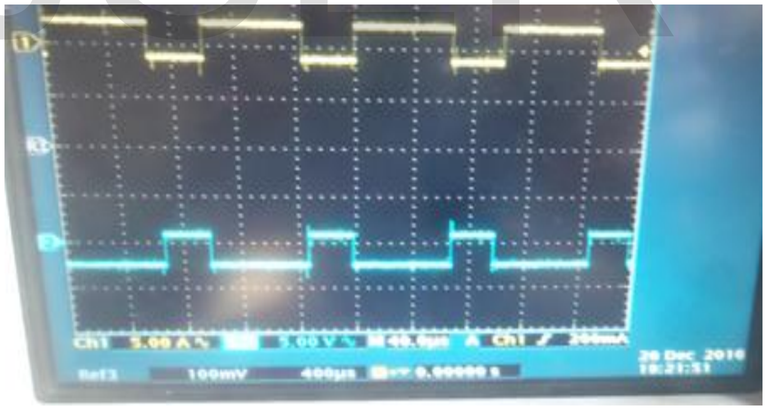
DC bus power is the high voltage line that provides voltage for the inverter stage to generate three phase AC voltage to control the motor. [Main]-BS5 and [Main]-BS6 are the power and ground connector for this inverter bus respectively.

5.5 Experimental Result

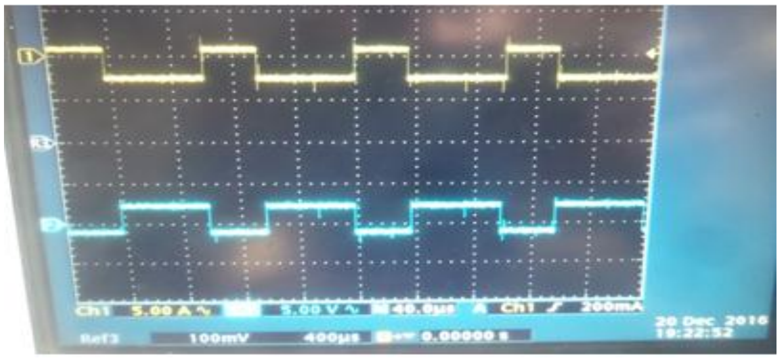
The main target in this experimental investigation is to control the motor at variable speed by generating the PWM signal which generates the appropriate sinusoidal current on the stator of the motor through DSP. This makes the motor to rotate at the required speed and control the motor at different speed including zero speed. The six PWM output signals are shown in Figure 5.7 while the motor is running 0.2pu.



(a) Pwm1_H and Pwm1_L



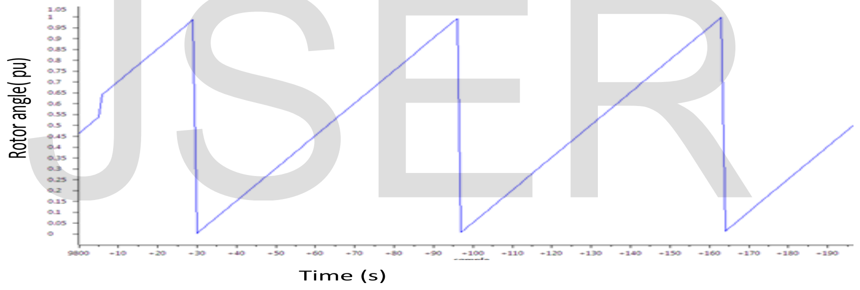
(b) Pwm2_H and Pwm2_L



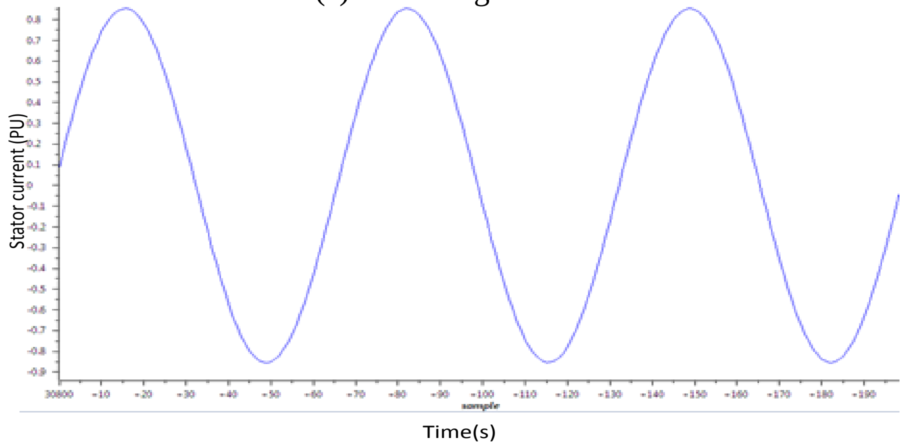
(c) Pwm3_H and Pwm3_L

Figure 5.7 the six PWM output signal from the DSP while the motor is running 0.2pu

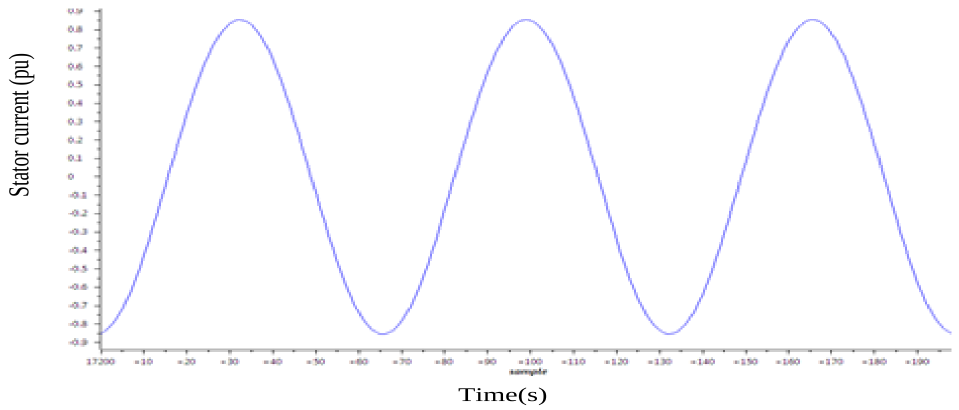
As shown in Figure 5.8 the estimated angle and the stator current have been seen while the motor is running at 0.2pu. From this the estimated angle and the stator current has been good accuracy and almost similar with the simulation result shown in Figure 4.4 of the stator current and Figure 4.6(c) of the actual angle.



(a) Rotor angle



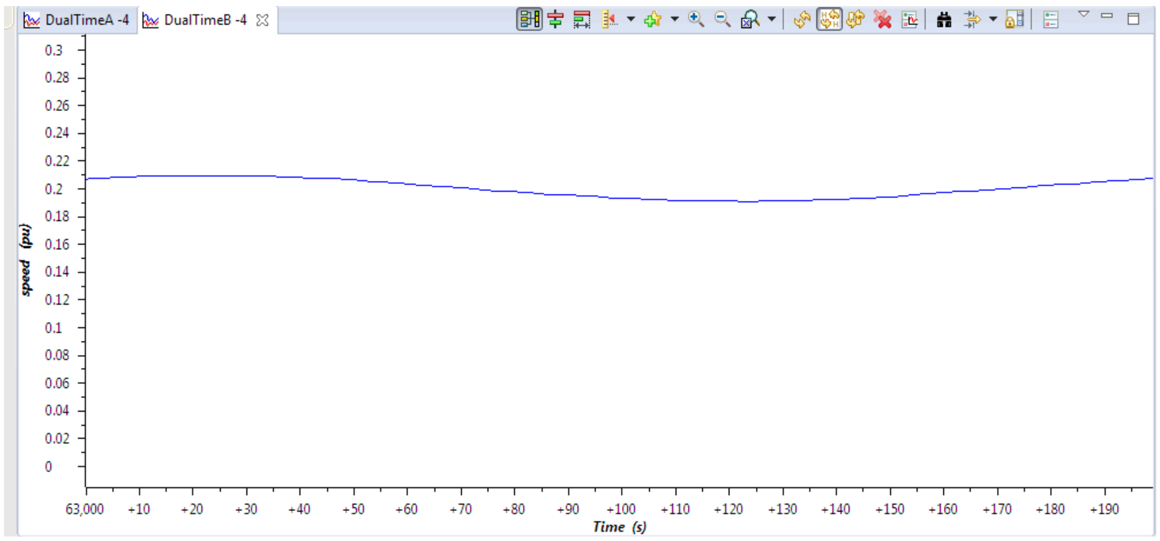
(b) α_axis stator current



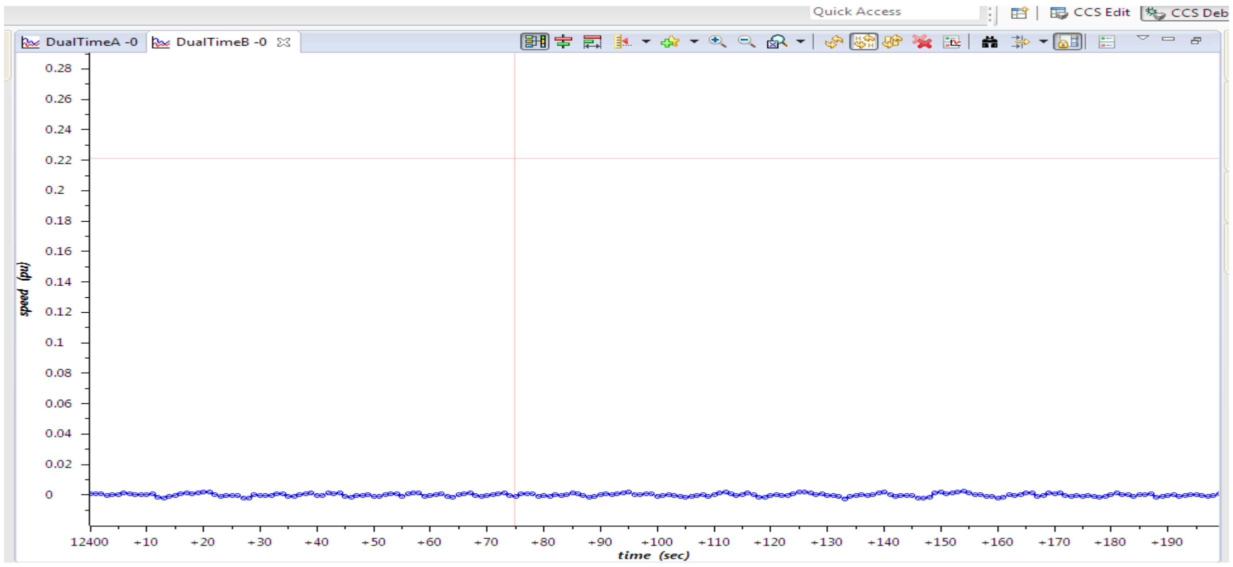
($\boldsymbol{c}$) β_axis stator current

Figure 5.8 Estimated rotor angle and stator current while the motor is running 0.2pu in closed loop

It can be seen in Figure 5.9 and Figure 5.10; that the induction motor tracks the desired speeds well in wide ranges, which demonstrates the effectiveness of the proposed scheme. From these the rotor speed has been good accuracy and almost similar with that of the simulation result shown in Figure 4.9. The implementation result shows that stator current based MRAS speed estimator can estimate the rotor speed with 0.00458pu of steady state error and good transient performance has been achieved. The implementation result shows stator current based MRAS speed estimator can estimate the rotor speed with good performance of steady state and transient response for sensorless speed control of induction motor.

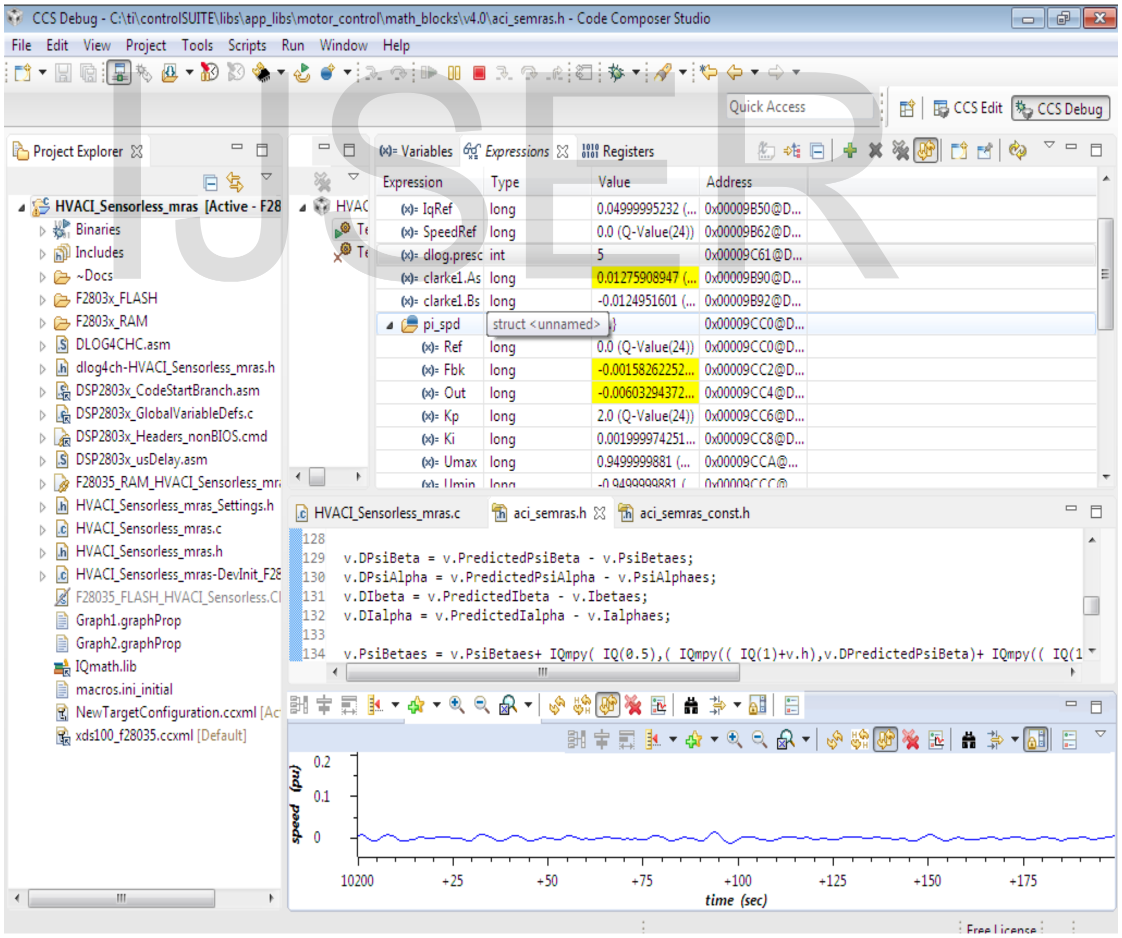


(a) Estimated rotor speed at 0.2pu



(b) Estimated rotor speed at 0.0pu

Figure 5.9 rotor speed while the motor is running in closed loop at 0.2 and 0.0pu

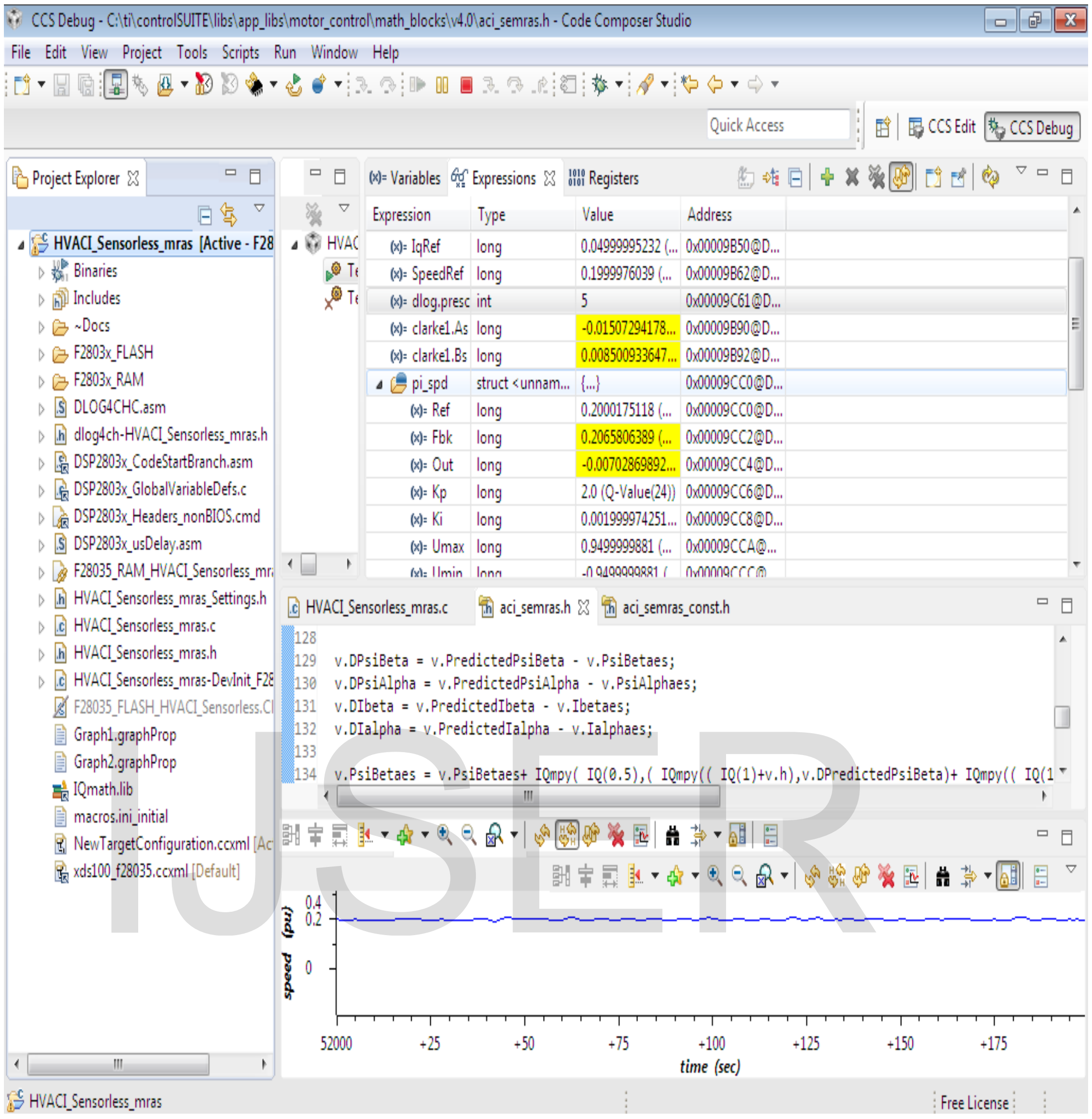


```
HVACI_Sensorless_mras [Active - F28
  Binaries
  Includes
  ~Docs
  F2803x_FLASH
  F2803x_RAM
  DLOG4CHC.asm
  dlog4ch-HVACI_Sensorless_mras.h
  DSP2803x_CodeStartBranch.asm
  DSP2803x_GlobalVariableDefs.c
  DSP2803x_Headers_nonBIOS.cmd
  DSP2803x_usDelay.asm
  F28035_RAM_HVACI_Sensorless_mr
  HVACI_Sensorless_mras_Settings.h
  HVACI_Sensorless_mras.c
  HVACI_Sensorless_mras.h
  HVACI_Sensorless_mras-DevInit_F28
  F28035_FLASH_HVACI_Sensorless.Cl
  Graph1.graphProp
  Graph2.graphProp
  IQmath.lib
  macros.ini_initial
  NewTargetConfiguration.ccxml [Ac
  xds100_f28035.ccxml [Default]
```

Expression	Type	Value	Address
IqRef	long	0.04999995232 (...	0x00009B50@D...
SpeedRef	long	0.0 (Q-Value(24))	0x00009B62@D...
dlog.presc	int	5	0x00009C61@D...
clarke1.As	long	0.01275908947 (...	0x00009B90@D...
clarke1.Bs	long	-0.0124951601 (...	0x00009B92@D...
pi_spd	struct <unnamed>	}	0x00009CC0@D...
Ref	long	0.0 (Q-Value(24))	0x00009CC0@D...
Fbk	long	-0.00158262252...	0x00009CC2@D...
Out	long	-0.00603294372...	0x00009CC4@D...
Kp	long	2.0 (Q-Value(24))	0x00009CC6@D...
Ki	long	0.001999974251...	0x00009CC8@D...
Umax	long	0.9499999881 (...	0x00009CCA@...
Umin	long	-0.9499999881 (	0x00009CCC@

```
v.DPsiBeta = v.PredictedPsiBeta - v.PsiBetaes;
v.DPsiAlpha = v.PredictedPsiAlpha - v.PsiAlphaes;
v.DIbeta = v.PredictedIbeta - v.Ibetaes;
v.DIalpha = v.PredictedIalpha - v.Ialphaes;

v.PsiBetaes = v.PsiBetaes+ IQmpy( IQ(0.5),( IQmpy(( IQ(1)+v.h),v.DPredictedPsiBeta)+ IQmpy(( IQ(1
```

(a) Snapshot of CCS programming interface while the motor is running at 0.0pu


CCS Debug - C:\ti\controlSUITE\libs\app_libs\motor_control\math_blocks\v4.0\aci_semras.h - Code Composer Studio
File Edit View Project Tools Scripts Run Window Help
Quick Access
CCS Edit
CCS Debug
Project Explorer
HVACI_Sensorless_mras [Active - F28
Binaries
Includes
~Docs
F2803x_FLASH
F2803x_RAM
DLOG4CHC.asm
dlog4ch-HVACI_Sensorless_mras.h
DSP2803x_CodeStartBranch.asm
DSP2803x_GlobalVariableDefs.c
DSP2803x_Headers_nonBIOS.cmd
DSP2803x_usDelay.asm
F28035_RAM_HVACI_Sensorless_mr
HVACI_Sensorless_mras_Settings.h
HVACI_Sensorless_mras.c
HVACI_Sensorless_mras.h
HVACI_Sensorless_mras-DevInit_F28
F28035_FLASH_HVACI_Sensorless.C
Graph1.graphProp
Graph2.graphProp
IQmath.lib
macros.ini_initial
NewTargetConfiguration.ccxml [Ac
xds100_f28035.ccxml [Default]
Variables
Expressions
Registers
HVACI_Sensorless_mras.c
aci_semras.h
aci_semras_const.h
0.4
0.2
0
speed (pu)
52000
+25
+50
+75
+100
+125
+150
+175
time (sec)
HVACI_Sensorless_mras
Free License

Expression	Type	Value	Address
IqRef	long	0.04999995232 (...	0x00009B50@D...
SpeedRef	long	0.1999976039 (...	0x00009B62@D...
dlog.presc	int	5	0x00009C61@D...
clarke1.As	long	-0.01507294178...	0x00009B90@D...
clarke1.Bs	long	0.008500933647...	0x00009B92@D...
pi_spd	struct <unnam...	{...}	0x00009CC0@D...
Ref	long	0.2000175118 (...	0x00009CC0@D...
Fbk	long	0.2065806389 (...	0x00009CC2@D...
Out	long	-0.00702869892...	0x00009CC4@D...
Kp	long	2.0 (Q-Value(24))	0x00009CC6@D...
Ki	long	0.001999974251...	0x00009CC8@D...
Umax	long	0.9499999881 (...	0x00009CCA@...
Umin	long	-0.9499999881 (	0x00009CCC@

```
v.DPsiBeta = v.PredictedPsiBeta - v.PsiBetaes;
v.DPsiAlpha = v.PredictedPsiAlpha - v.PsiAlphaes;
v.DIbeta = v.PredictedIbeta - v.Ibetaes;
v.DIalpha = v.PredictedIalpha - v.Ialphaes;

v.PsiBetaes = v.PsiBetaes+ IQmpy( IQ(0.5),( IQmpy(( IQ(1)+v.h),v.DPredictedPsiBeta)+ IQmpy(( IQ(1
```

(b) Snapshot of CCS programming interface while the motor is running at 0.2pu

Figure 5.10 Snapshot of CCS programming interface while the motor is running at 0.2pu and 0.0pu

Chapter Six

Conclusion and Recommendation

6.1 Conclusion

In this thesis, design and implementation of stator current based MRAS speed estimator overcome the problem due to mechanical speed sensor and parameter sensitivity for speed control of induction motor has been investigated. The computational techniques used to simplify the stator current based MRAS speed estimator design and its implementation on Texas instrument TMS320F28035 control card is discussed. The stator current based MRAS speed estimator has been found to be well suited to the rotor speed estimation of induction motor. The simulation results show stator current based MRAS can estimate the rotor speed with good performance speed and position error of 0.024rad/s and 0.1rad for speed control of induction motor has been achieved. As it is shown from Figure 4.6, stator current based MRAS speed estimator estimate the rotor speed of induction motor with a steady state error of 0.028% and good transient response with rise time of less than 0.1 second and settling time 0.15second has been achieved.

The performance of stator current based MRAS speed estimator was analyzed in terms of speed tracking capability, torque response quickness, low speed behavior, step response of drive with speed reversal, sensitivity to motor parameter uncertainty, and speed tracking ability in regenerative mode. The system gives good performance at no load and loaded condition. Hence, it can work with different load torque conditions and with parameters variation.

The closed loop experimental investigation is implemented using Texas Instrument, Piccolo TMS320F28035 control card. Demonstration results smooth speed control and with maximum steady state error of 0.00458pu and good transient response has been achieved.

6.2 Recommendation

Future Research Suggestions is in the following direction.

The adaptive PI controller of stator current based MRAS speed estimator which is used in this thesis is based on the linearization of adaptive mechanism (as shown in the Figure 3.2). However, it needs to be converted to linear system which actually makes the mathematical analysis of the system complex. But, it is possible to make this analysis simple by using artificial non-linear controller instead.

References

[1] P. Vas, “Sensorless Vector and Direct Torque Control”. New York: Oxford Univ. Press, 1998

[2] Seung-ki-su. “control of electrical machine drives (bookos.org)” Published by John Wiley & Sons, Inc., Hoboken, New Jersey. Page 316-345.

[3] M. S. Zaky, M Khater, H. Yasin and S. S. Shokralla, “Speed sensorless control of induction motor drives (review paper)”, ACTA Electrothechnica, Vol. 49, No. 3, pp. 221-228, 2008.

[4] J. W. Finch and D. Giaouris, “Controlled AC electrical drives”, IEEE Trans. on Industrial Electronic, Vol. 55, No. 1, pp. 1-11, 2008.

[5] S. Sedghizadeh, C. Lucas and H. Ghafoori Fard, “Sensorless speed control of switched reluctance motor drive using the binary observer with online flux-linkage estimation”, Iranian Journal of Electrical & Electronic Engineering, Vol. 5, No. 2, pp. 143-150, 2009.

[6] .Blasco Giménez, Ramón High performance sensorless vector control of induction motor drives. PhD thesis, University of Nottingham.h, (1995)

[7] R. Di Gabriele, F. Parasiliti, M. Tursini, "Digital Field Oriented Control for induction motors: implementation and experimental results". Universities Power Engineering Conference (UPEC97).

[8] Bimal K.Bose ,”Modern Power Electronics And AC Drives” ,Condra Chair Of Excellence In Power Electronics ,The University Of Tennesse Knoxville.

[9] Ong, Chee-Mun., “Dynamic Simulation of Electric Machinery”, N.J, Printice Hall, 1998

[10] E. D. Mitronikas and A. N. Safacas, “An improved sensorless vector control method for an induction motor drive”, IEEE Transactions on Industrial Electronic, Vol. 52, No. 6, pp. 1660-1668 Dec. 2005.

[11] R. Krishan, “Electric motor drives: Modelling, analysis and control,” Prentice Hall Company, New Jersey, 2001.

[12] B.K.Bose, “Modern Power Electronics and AC Drives”. New Delhi, India: Prentice-Hall, 2002, Ch. 8, pp. 333–435.

[13] Prof. Himanshu K. Patel, “Steady State and Transient Performance Analysis of Three Phase Induction Machine using MATLAB Simulations”, International Journal of Recent Trends in Engineering, Vol 1, No. 3, May 2009

[14] P.C. Krause and C. H. Thomas,“ Simulation of Symmetrical Induction Machinery”, IEEE Transaction on Power Apparatus and Systems, Vol. 84, November 1965, pp. 1038-1053.

[15] F.Blashke, “The principle of field orientation as applied to the new trans vector closed loop control system for rotating field machines,” siemens Review, vol.34,PP.217-220 May 1972.

[16] Mircea Popescu, “Induction Motor Modeling for Vector Control Purposes”, Helsinki University of Technology Department of Electrical and Communications Engineering Laboratory of Electromechanics, 2000.

[17] Bimal K.Bose, “Modern Power Electronics and AC Drives.”; The Universtiy of Tennesse. Knoxville

[18] Haitham Abu-Rub, Atif Iqbal, Jaroslaw Guzinski and “High Performance Control of Ac Drives With Matlab/Simulink Models , published 2012.

[19] L. Umanand and S. Bhat, “Online estimation of stator resistance of an induction motor for speed control applications,” IEE Proc. Electr. Power Appl., vol. 142, pp. 97–103, Mar. 1995.

[20] P.Zhang, Bin Lu, and T.G. Habetler; “A Remote and Sensorless Stator Winding Resistance Estimation Method for Thermal Protection of Soft-Starter-Connected Induction Machines”; IEEE Transactions on Industrial Electronics, Vol. 55, No. 10, Pp. 3611-3618, October 2008

[21] Dal Y. Ohm Drivetech, “Dynamic Model of Induction Motors For Vector Control” Inc., Blacksburg, Virginia

[22] Mr. Sandeep N Panchal, Mr. Vishal S Sheth, Mr. Akshay A Pandya. “Simulation Analysis of SVPWM Inverter Fed Induction Motor Drives”

[23] J. Holtz, W. Lotzkat, and A.M. Khambadkone, “On Continuous Control of PWM Inverters in the Over-modulation Range Including the Six-Step Mode”, IEEE Transactions on Power Electronics, Vol. 8, No. 4, pp. 546-553, October 1993.

[24] E. Hendawi, F. Khater, and A. Shaltout, "Analysis, Simulation and Implementation of Space Vector Pulse Width Modulation Inverter" International Conference on Application of Electrical Engineering, pp. 124-131, 2010.

[25] B. Wu. "High-Power Converters and AC Drives". IEEE Press, John Wiley and Sons, Inc., 2006.

[26] D.G. Holmes and T.A. Lipo. "Pulse Width Modulation for Power Converters". John Wiley & Sons, Inc. 2003.

[27] Ma, Chaozheng, "Speed sensorless control of 3-phase induction motor using MRAS speed estimator" (2003). Theses and dissertations.

[28] Joachim Holtz, "Sensorless Control of Induction Motor Drives". IEEE Proc., Paper 19 90 (8) (2002), pp. 1359-1394.

[29] J. W. Finch and D. Giaouris, "Controlled AC electrical drives", IEEE Trans. on Industrial Electronic, Vol. 55, No. 1, pp. 1-11, 2008.

[30] M. N. Marwali and A. Keyhani, "A comparative study of rotor flux based MRAS and back EMF based MRAS speed estimators for speed sensorless vector control of induction machines". Proc. of the IEEE-IAS Annual Meeting, 1997

[31] F.Z. Peng, T. Fukao, "Robust speed identification for speed sensorless vector control of induction motors". IEEE Trans. Ind. Applic., 30 (5), 1994, pp. 1234–1240.

[32] M. Rashed and A.F. Stronach, "A stable back-EMF MRAS-based sensorless low speed induction motor drive insensitive to stator resistance variation". IEE Proc. Electr. Power Applic., 151 (6) (2004), pp. 685-693.

[33] Wafa Bourbia, Farid Berrezzek and Bachir Bensaker , "Circle-criterion Based Nonlinear Observer Design for Sensorless Induction Motor Control", International Journal of Automation and Computing, December 2014..

[34] TMS320C28x DSP. CPU and Instruction Set Reference Guide, Texas Instruments, Inc. Dallas, TX, [Online]. Available: http://www.ti.com.

[35] Bilal Akin, Manish Bhardwaj , "Sensorless Field Oriented Control of 3-Phase Induction Motors" Texas Instruments, Inc.

Appendix

Appendix A

"C" code for stator current based MRAS speed estimator

```
Description: Primary system file for the Real Implementation of  stator current
based MRAS Sensorless Field Orientation Control for Induction Motors
/*==========================================================
File name:      stator current based MRAS speed estimator (_MRAS_CONST.H)
==========================================================*/
#ifndef __ACI_MRAS_CONST_H__
#define __ACI_MRAS_CONST_H__
typedef struct  {
            float32  Rs;      // Input: Stator resistance (ohm)
            float32  Rr;      // Input: Rotor resistance (ohm)
            float32  Ls;      // Input: Stator inductance (H)
            float32  Lr;      // Input: Rotor inductance (H)
            float32  Lm;   // Input: Magnetizing inductance (H)
            float32  B;    //viscus friction
            float32  J;      //momuntem of inertia
            float32  p;      //number of pole
            float32  Tb;     // input base torque
            float32  Ib;     // Input: Base phase current (amp)
            float32  Lb;  // base flux
            float32  Vb;  // Input: Base phase voltage (volt)
            float32  Wb;  // base speed
            float32  Ts;  // Input: Sampling period in sec
            float32  Tr;  // Parameter: Rotor time constant
            float32  K1; // Output: constant using in rotor flux calculation
            float32  K2; // Output: constant using in rotor flux calculation
            float32  K3; // Output: constant using in rotor flux calculation
            float32  K4;// Output: constant using in stator current calcultion
            float32  K5;// Output: constant using in stator current calculation
            float32  K6;// Output: constant using in stator current calculation
            float32  K7;// Output: constant using in stator current calculation
            float32  K8;// Output: constant using in torque calculation
            float32  K9;   // Output: constant using in speed calculation
            float32  K10;  // constant
            float32  Ks;      // slip speed calculation constant
            float32  sigma; // constant
            float32  h;      //time step of computation
            float32  beta;  // constant
            float32 gamma;  // constant
} ACIMRAS_CONST;
/*-----------------------------------------------------------
   Default initializer for the ACI MRAS_CONST object.
---------------------------------------------*/
#define ACIMRAS_CONST_DEFAULTS
```

```
                    {0,0,0,0,0,0,0,0,\
                    0,0,0,0,0,0,0,0, 0,0,\
                    0,0,0,0,0,0,0,0,0,0,0,\
                         }

/*-------------------------------------------------------
   ACIMRAS_CONST MACRO Definition
---------------------------------------------------*/
#define ACIMRAS_CONST_MACRO(v)      \
  v.Tr = v.Lr/v.Rr;        \
  v.sigma = 1 - (v.Lm*v.Lm)/(v.Ls*v.Lr); \
  v.gamma = (v.Lm*v.Lm*v.Rr + v.Lr*v.Lr*v.Rs)/(v.sigma*v.Ls*v.Lr*v.Lr);  \
             v.h =  v.Rr/v.Lr;            \
             v.beta = v.Lm/(v.sigma*v.Ls*v.Lr);    \
             v.K1 = v.Ts*v.h;        \
             v.K2 = v.Ts*v.Wb;             \
              v.K3 = v.Ts*v.h*v.Lm*(v.Ib/v.Lb);    \
              v.K4 = v.Ts*v.h*v.beta*(v.Lb/v.Ib);    \
               v.K5 = v.Ts*v.beta*(v.Lb*v.Wb/v.Ib);    \
               v.K6 = v.Ts*v.gamma;    \
               v.K7 = v.Ts*(1/(v.sigma*v.Ls))*(v.Vb/v.Ib);  \
               v.K8 = 1.5*(v.p/2)*(v.Lm/v.Lr)*(v.Lb*v.Ib/v.Tb); \
               v.K9 = v.Ts*(v.B/v.J);        \
               v.K10 = v.Ts*(v.p/2)*(1/v.J)*(v.Tb/v.Wb);\
               v.Ks = 1/(v.Wb*v.Tr);              \

              #endif

/* ============================================================

File name:      stator current based speed  mras estimator (MRAS.H)
============================================================*/

              #ifndef __ACI_MRAS_H__
              #define __ACI_MRAS_H__
             typedef struct {_iq  Ualpha;  // Input: alpha-axis phase voltage at k
             _iq    Ubeta;                 // Input: beta-axis phase voltage at k
             _iq    IAlpha;                  // Input: stationary d-axis stator variable
             _iq    IBeta;                  // Input: stationary q-axis stator variable
             _iq    Ialphaes;            // Output: alpha-axis phase current at k   estimated
             _iq    Ibetaes;              // Output: beta-axis phase current at k  estimated
             _iq    PsiAlphaes;            // Output: alpha-axis rotor flux at k    estimated
             _iq    PsiBetaes;             // Output: beta-axis rotor flux at k  estimated
             _iq    K1;            // Parameter: constant using in rotor flux calculation
             _iq    K2;            // Parameter: constant using in rotor flux calculation
             _iq    ks;             //v.Ks = 1/(v.Wb*v.Tr); slip constant
             _iq    K3;           // Parameter: constant using in rotor flux calculation
             _iq    K4;         // Parameter: constant using in stator current calculation
             _iq    K5;         // Parameter: constant using in stator current calculation
             _iq    K6;         // Parameter: constant using in stator current calculation
_iq    K7;             // Parameter:constantusing in stator current calculation
_iq    K8;              // Parameter: constant using in Torque calculation
_iq    K9;              // Parameter: constant using in rotor speed calculation
       _iq    K10;             // Parameter: constant using in rotor speed calculation
       _iq    h;               // Parameter: step time constant
       int32   speedes;          // speed estimation
       _iq    Ki;                // Parameter: PI integral  adptive gain speed
       _iq    Kp;                // Parameter: PI proportionnal  adaptive gain speed
       _iq    speedesRpm;          // Output: Estimated speed mras in rpm
       _iq    PredictedPsiBeta;
       _iq    PredictedPsiAlpha;
       _iq    PredictedIbeta;
       _iq    PredictedIalpha;
       _iq    DPredictedPsiBeta;
       _iq    DPredictedPsiAlpha;
       _iq    DPredictedIbeta;
       _iq    DPredictedIalpha;
       _iq    DPsiBeta;
       _iq    DPsiAlpha;
       _iq    DIbeta;
       _iq    DIalpha;
       _iq    Torquem;
       _iq    PredictedWrm;
       _iq    Wrm;
       _iq    LoadTorquem;
       _iq    DPredictedWrm;
       _iq    DWrm;
       _iq    WrmRpm;
       _iq   SquaredPsi;
       _iq   WSlip;   //slip speed
       float32    Tr;  // rotor time constant
       float32    Rs; // Input: Stator resistance (ohm)
       float32    Rr; // Input: Rotor resistance (ohm)
       float32    Ls; // Input: Stator inductance (H)
       float32    Lr; // Input: Rotor inductance (H)
       float32    Lm; // Input: Magnetizing inductance (H)
       float32    Ts; // Input: Sampling period in sec
       float32    sigma;
       Uint32     BaseRpm; // Parameter: base motor speed in rpm
       _iq    v1;
       _iq    Umax;
       _iq    Umin;
       _iq    i1;
       _iq    up;
       _iq    ui;
       _iq    w1;
       _iq   Out;
       _iq   ThetaFlux;
       float32  Wb;
       _iq    PsiDrS;
       _iq   IQsS;
        _iq  PsiQrS;
       _iq   IDsS;
       _iq WSyn;
       _iq   WSlp;
       _iq  WSli;
        } ACIMRAS;
       /*-----------------------------------------------------
           Default initializer for the ACI MRAS object.
       -------------------------------------------------------*/
       #define ACIMRAS_DEFAULTS {0,0,0,0,0,0,0,0,0,0,_IQ(0.1),0,              \
                  0,0,0,0,0,0,0,0,0,_IQ(1.5166),_IQ(0.678),0,0, \
                  0,0,0,0,0,0,0,0,0,0,0,0,                   \
                  0,0,0,0,0,0,0,0,0,0,0,0,                   \
                 0,0,0,0,3600,_IQ(1.0),_IQ(-1.0),_IQ(-1.0),    \
                _IQ(0.0), _IQ(0.0), _IQ(0.0),_IQ(0.0),0,150,0,0,0,0,0,0,0,
                           }
        /*------------------------------------------------------
        Stator current based mras speed Estimator( MRAS  MACRO Definition)
       -------------------------------------------------------*/
       #define ACIMRAS_MACRO(v)
        /* Rotor flux/Stator current calculation */
       /* Predictor */
         v.Tr=v.Lr/v.Rr;
        v.Tr=_IQdiv(v.Lr,v.Rr);
        v.sigma =1-(v.Lm*v.Lm)/(v.Ls*v.Lr);
        v.sigma = _IQ(1)-_IQdiv(_IQmpy(v.Lm,v.Lm),_IQmpy(v.Ls,v.Lr));
        v.ks = 1/(v.Wb*v.Tr);
       v.ks = _IQdiv(_IQ(1),_IQmpy(v.Wb,v.Tr));

        v.PredictedPsiBeta=v.PsiBetaes-_IQmpy(v.K1,v.PsiBetaes)+
       _IQmpy(_IQmpy(v.K2,v.speedes),v.PsiAlphaes) + _IQmpy(v.K3,v.Ibetaes);

         v.PredictedPsiAlpha = v.PsiAlphaes - _IQmpy(v.K1,v.PsiAlphaes) -
       _IQmpy(_IQmpy(v.K2,v.speedes),v.PsiBetaes) + _IQmpy(v.K3,v.Ialphaes);

         v.PredictedIbeta = v.Ibetaes + _IQmpy(v.K4,v.PsiBetaes) -
       _IQmpy(_IQmpy(v.K5,v.speedes),v.PsiAlphaes) -_IQmpy(v.K6,v.Ibetaes)+
       _IQmpy(v.K7,v.Ubeta);

          PredictedIalpha = v.Ialphaes + _IQmpy(v.K4,v.PsiAlphaes) +
       _IQmpy(_IQmpy(v.K5,v.speedes),v.PsiBetaes) -_IQmpy(v.K6,v.Ialphaes) +
       _IQmpy(v.K7,v.Ualpha);
               Corrector */
       v.DPredictedPsiBeta = - _IQmpy(v.K1,v.PredictedPsiBeta)
       +_IQmpy(_IQmpy(v.K2,v.speedes),v.PredictedPsiAlpha)
       +_IQmpy(v.K3,v.PredictedIbeta);

       v.DPredictedPsiAlpha = -_IQmpy(v.K1,v.PredictedPsiAlpha) -
```

```
_IQmpy(_IQmpy(v.K2,v.speedes),v.PredictedPsiBeta) +
_IQmpy(v.K3,v.PredictedIalpha);

  PredictedIbeta = _IQmpy(v.K4,v.PredictedPsiBeta) -
_IQmpy(_IQmpy(v.K5,v.speedes),v.PredictedPsiAlpha) -
_IQmpy(v.K6,v.PredictedIbeta) + _IQmpy(v.K7,v.Ubeta);

   v.DPredictedIalpha = _IQmpy(v.K4,v.PredictedPsiAlpha) +
_IQmpy(_IQmpy(v.K5,v.speedes),v.PredictedPsiBeta) -
_IQmpy(v.K6,v.PredictedIalpha) + _IQmpy(v.K7,v.Ualpha);

    v.DPsiBeta =  v.PredictedPsiBeta - v.PsiBetaes;

v.DPsiAlpha = v.PredictedPsiAlpha - v.PsiAlphaes;

v.DIbeta = v.PredictedIbeta - v.Ibetaes;

  v.DIalpha = v.PredictedIalpha - v.Ialphaes;

v.PsiBetaes=v.PsiBetaes+_IQmpy(_IQ(0.5),(_IQmpy((_IQ(1)+v.h),v.DPredictedPsi
Beta)+_IQmpy((_IQ(1)-v.h),v.DPsiBeta)));

v.PsiAlphaes=v.PsiAlphaes+
_IQmpy(_IQ(0.5),(_IQmpy((_IQ(1)+v.h),v.DPredictedPsiAlpha) +_IQmpy((_IQ(1)-
v.h),v.DPsiAlpha)));

  v.Ibetaes = v.Ibetaes+
_IQmpy(_IQ(0.5),(_IQmpy((_IQ(1)+v.h),v.DPredictedIbeta) +_IQmpy((_IQ(1)-
v.h),v.DIbeta)));

v.Ialphaes =
v.Ialphaes+_IQmpy(_IQ(0.5),(_IQmpy((_IQ(1)+v.h),v.DPredictedIalpha)
+_IQmpy((_IQ(1)-v.h),v.DIalpha)));

 /* Electromagnetic Torque calculation*/
 v.Torquem = _IQmpy(v.K8,(_IQmpy(v.PsiAlphaes,v.Ibetaes) -
_IQmpy(v.PsiBetaes,v.Ialphaes)));

 /* Rotor speed calculation */
  v.PredictedWrm = v.Wrm - _IQmpy(v.K9,v.Wrm) + _IQmpy(v.K10,(v.Torquem -
v.LoadTorquem));

  v.DPredictedWrm = - _IQmpy(v.K9,v.PredictedWrm) +
_IQmpy(v.K10,(v.Torquem - v.LoadTorquem));

  v.DWrm = v.PredictedWrm - v.Wrm;
 /*corrector; */

  v.Wrm = v.Wrm + _IQmpy(_IQ(0.5),(_IQmpy((_IQ(1)+v.h),v.DPredictedWrm) +
_IQmpy((_IQ(1)-v.h),v.DWrm)));
```

```
                v.WrmRpm = _IQmpy(v.Wrm,v.BaseRpm);

                 /* adaptive PI controller section speed estimation */

                v.up = _IQmpy ((v.IAlpha - v.Ialphaes), v.PsiBetaes) - _IQmpy ((v.IBeta -
                v.Ibetaes), v.PsiAlphaes);
                 /* integral term */
                 v.ui = (v.Out == v.v1)?(_IQmpy(v.Ki, v.up)+ v.i1) : v.i1;
                v.i1 = v.ui;
                           /* control output */
                v.v1 = _IQmpy(v.Kp, (v.up + v.ui));
                 v.Out= _IQsat(v.v1, v.Umax, v.Umin);
                v.speedes=v.Out;
                 /* slip speed calcalculation*/
                 v.SquaredPsi =
                IQmpy(v.PsiAlphaes,v.PsiAlphaes)+_IQmpy(v.PsiBetaes,v.PsiBetaes);
                 v.WSlp= _IQmpy(v.ks,(_IQmpy(v.PsiAlphaes,v.Ibetaes) -
                _IQmpy(v.PsiBetaes,v.Ialphaes)));
                 v.WSli= _IQdiv(v.WSlip,v.SquaredPsi);
                v.WSlip=_IQtoIQ21(v.WSli);
                  /* Compute the rotor flux angle*/
                v.WSyn +=_IQmpy(v.Ts,(v.WSlip + v.speedesRpm));
                 v.ThetaFlux=v.WSyn;

        #endif // __ACI_MRAS_H_
//----------------------------------------------
  // main code
//------------------------------------------------------------*/
System Name:       HVACI_Sensorless_mras

File Name:         HVACI_Sensorless_mras.C
// Include header files used in the main function

      #include "PeripheralHeaderIncludes.h"
      #define   MATH_TYPE      IQ_MATH
      #include "IQmathLib.h"
      #include "HVACI_Sensorless_mras.h"
      #include "HVACI_Sensorless_mras_Settings.h"
      #include <math.h>
      #ifdef FLASH
      #pragma CODE_SECTION(MainISR,"ramfuncs");
      #pragma CODE_SECTION(OffsetISR,"ramfuncs");
      #endif

      // Prototype statements for functions found within this file.
      interrupt void MainISR(void);
      interrupt void OffsetISR(void);
      void DeviceInit();
      void MemCopy();
      void InitFlash();
      void HVDMC_Protection(void);
```

```
// State Machine function prototypes
//------------------------------------
// Alpha states
void A0(void);      //state A0
void B0(void);      //state B0
void C0(void);      //state C0

// A branch states
void A1(void);      //state A1
void A2(void);      //state A2
void A3(void);      //state A3

// B branch states
void B1(void);      //state B1
void B2(void);      //state B2
void B3(void);      //state B3

// C branch states
void C1(void);      //state C1
void C2(void);      //state C2
void C3(void);      //state C3

// Variable declarations
void (*Alpha_State_Ptr)(void);   // Base States pointer
void (*A_Task_Ptr)(void);        // State pointer A branch
void (*B_Task_Ptr)(void);        // State pointer B branch
void (*C_Task_Ptr)(void);        // State pointer C branch

// Used for running BackGround in flash, and ISR in RAM
extern Uint16 *RamfuncsLoadStart, *RamfuncsLoadEnd, *RamfuncsRunStart;


int16  VTimer0[4];  // Virtual Timers slaved off CPU Timer 0 (A events)
int16  VTimer1[4];  // Virtual Timers slaved off CPU Timer 1 (B events)
int16  VTimer2[4];  // Virtual Timers slaved off CPU Timer 2 (C events)
int16  SerialCommsTimer;

// Global variables used in this system

_iq VdTesting = _IQ(0.2);             // Vd reference (pu)
_iq VqTesting = _IQ(0.2);             // Vq reference (pu)
_iq IdRef      = _IQ(0.1);            // Id reference (pu)
_iq IqRef      = _IQ(0.05);                  // Iq reference (pu)
_iq SpeedRef   = _IQ(0.3);            // Speed reference (pu)

float32 T = 0.001/ISR_FREQUENCY;    // Samping period (sec)

Uint16 OffsetFlag = 0;
_iq offsetA = 0;
_iq offsetB = 0;
_iq offsetC = 0;
_iq K1 = _IQ(0.998); //Offset filter coefficient K1: 0.05/(T);
_iq K2 = _IQ(0.001999); //Offset filter coefficient K1: 0.05/(T+0.05);
```

```
extern _iq IQsinTable[];
extern _iq IQcosTable[];

Uint32 IsrTicker  = 0;
Uint16 BackTicker = 0;
Uint16 lsw = 0;                  //Loop switch
Uint16 TripFlagDMC = 0;   //PWM trip status

int16 DlogCh1 = 0;
int16 DlogCh2 = 0;
int16 DlogCh3 = 0;
int16 DlogCh4 = 0;

// Default ADC initialization
int ChSel[16]   = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int TrigSel[16] = {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5};
int ACQPS[16]   = {8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8};

volatile Uint16 EnableFlag = FALSE;
Uint16 SpeedLoopPrescaler = 10;  // Speed loop prescaler
Uint16 SpeedLoopCount = 1;       // Speed loop counter
//  rotor speed estimations
ACIMRAS semras = ACIMRAS_DEFAULTS; //semras=speed estimation based on MRAS
//  constant calculations for speed estimations
ACIMRAS_CONST semras_const = ACIMRAS_CONST_DEFAULTS ;
//(ICLARKE is added in SVGEN module)
CLARKE clarke1 = CLARKE_DEFAULTS;
PARK park1 = PARK_DEFAULTS;
IPARK ipark1 = IPARK_DEFAULTS;

//  PI regulators to regulate the d and q  axis currents, and speed
PI_CONTROLLER pi_spd = PI_CONTROLLER_DEFAULTS;
PI_CONTROLLER pi_id  = PI_CONTROLLER_DEFAULTS;
PI_CONTROLLER pi_iq  = PI_CONTROLLER_DEFAULTS;

//  a PWM driver instance
PWMGEN pwm1 = PWMGEN_DEFAULTS;
//  a PWM DAC driver instance
PWMDAC pwmdac1 = PWMDAC_DEFAULTS;

SVGEN svgen1 = SVGEN_DEFAULTS;

RMPCNTL rc1 = RMPCNTL_DEFAULTS;
//Instance a ramp(sawtooth) generator to simulate an Anglele
RAMPGEN rg1 = RAMPGEN_DEFAULTS;

//a phase voltage calculation
PHASEVOLTAGE volt1 = PHASEVOLTAGE_DEFAULTS;

// Create an instance of DATALOG Module
DLOG_4CH dlog = DLOG_4CH_DEFAULTS;

void main(void)

 {
        DeviceInit();// Device Life support & GPIO

       #ifdef FL ASH
        MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
        InitFlash(); // Call the flash wrapper init function
        #endif //(FLASH)

    // Waiting for enable flag set
    while (EnableFlag==FALSE)
     {
       BackTicker++;
     }
// Timer period definitions found in device specific PeripheralHeaderIncludes.h
        CpuTimer0Regs.PRD.all =  mSec1;        // A tasks
        CpuTimer1Regs.PRD.all =  mSec50;       // B tasks
        CpuTimer2Regs.PRD.all =  mSec500;      // C tasks

// Tasks State-machine init
              Alpha_State_Ptr = &A0;
              A_Task_Ptr = &A1;
              B_Task_Ptr = &B1;
              C_Task_Ptr = &C1;
       // Initialize PWM module
            pwm1.PeriodMax =SYSTEM_FREQUENCY*1000000*T/2;
             pwm1.HalfPerMax=pwm1.PeriodMax/2;
            pwm1.Deadband = 2.0*SYSTEM_FREQUENCY;
            PWM_INIT_MACRO(1,2,3,pwm1)
       // Initialize PWMDAC module
           pwmdac1.PeriodMax=500;
       // @60Mhz, 1500->20kHz, 1000-> 30kHz, 500->60kHz
           pwmdac1.HalfPerMax=pwmdac1.PeriodMax/2;
           PWMDAC_INIT_MACRO(6,pwmdac1)   // PWM 6A,6B
           PWMDAC_INIT_MACRO(7,pwmdac1)   // PWM 7A,7B

       // Initialize DATALOG module
            dlog.iptr1 = &DlogCh1;
            dlog.iptr2 = &DlogCh2;
            dlog.iptr3 = &DlogCh3;
            dlog.iptr4 = &DlogCh4;
            dlog.trig_value = 0x1;
            dlog.size = 0x0C8;
            dlog.prescalar = 5;
            dlog.init(&dlog);

// Initialize ADC for DMC Kit Rev 1.1
              ChSel[0]=1;
              ChSel[1]=1;
              ChSel[2]=9;
              ChSel[3]=3;
              ChSel[4]=15;
              ChSel[5]=14;
              ChSel[6]=12;
              ChSel[7]=7;

              ADC_MACRO_INIT(ChSel,TrigSel,ACQPS)
        // Initialize the RAMPGEN module
                  rg1.StepAngleMax = _IQ(BASE_FREQ*T);
// Initialize the speed estimator (stator current based MRAS) constants module
                  semras_const.Rs = RS;  //semras=senserless MRAS
                  semras_const.Rr = RR;
                  semras_const.Ls = LS;
                  semras_const.Lr = LR;
                  semras_const.Lm = LM;
                  semras_const.p =  POLES;
                  semras_const.J = 0.009;
                  semras_const.B = 0.0006;
                  semras_const.Wb=314;
                  semras_const.Ib = 10;
                  semras_const.Vb = BASE_VOLTAGE;
                  semras_const.Ts = T;
                  semras_const.Lb = 0.98; base flux
                  semras_const.Tb= 1.7803;
                  ACIMRAS_CONST_MACRO(semras_const)

             // Initialize the speed estimator (stator current based MRAS) module
                semras.K1 = _IQ(semras_const.K1);
                semras.K2 = _IQ(semras_const.K2);
                semras.K3 = _IQ(semras_const.K3);
                semras.K4 = _IQ(semras_const.K4);
                semras.K5 = _IQ(semras_const.K5);
                semras.K6 = _IQ(semras_const.K6);
                semras.K7 = _IQ(semras_const.K7);
                semras.K8 = _IQ(semras_const.K8);
                semras.K9 = _IQ(semras_const.K9);
                semras.K10 = _IQ(semras_const.K10);
                semras.Kp = _IQ(1.5166);   // adaptive  kP controller
                semras.Ki = _IQ(0.678);     // adaptive ki controller
                semras.Rs = semras_const.Rs;
                semras.Rr = semras_const.Rr;
                semras.Ls = semras_const.Ls;
                semras.Lr = semras_const.Lr;
                semras.Lm = semras_const.Lm;
                semras.Ts = T;
                semras.BaseRpm=120*BASE_FREQ/POLES;
           // Initialize the PI module for Id
               pi_spd.Kp=_IQ(0.0570);
              pi_spd.Ki=_IQ(T/0.004);
              pi_spd.Umax =_IQ(0.95);
              pi_spd.Umin =_IQ(-0.95);
        // Initialize the PI module for Iq
              pi_id.Kp=_IQ(0.0570);
              pi_id.Ki=_IQ(T/0.004);
              pi_id.Umax =_IQ(0.3);
              pi_id.Umin =_IQ(-0.3);
        // Initialize the PI module for speed
              pi_iq.Kp=_IQ(0.3949);
              pi_iq.Ki=_IQ(0.263);
              pi_iq.Umax =_IQ(0.8);
              pi_iq.Umin =_IQ(-0.8);

        //Call HVDMC Protection function
              HVDMC_Protection();

              EALLOW;// This is needed to write to EALLOW protected registers

              PieVectTable.ADCINT1 = &OffsetISR;
        // Enable PIE group 1 interrupt 1 for ADC1_INT
              PieCtrlRegs.PIEIER1.bit.INTx1 = 1;
        // Enable EOC interrupt (after the 4th conversion)
              AdcRegs.ADCINTOVFCLR.bit.ADCINT1=1;
              AdcRegs.ADCINTFLGCLR.bit.ADCINT1=1;
              AdcRegs.INTSEL1N2.bit.INT1CONT=1;
              AdcRegs.INTSEL1N2.bit.INT1SEL=4;
              AdcRegs.INTSEL1N2.bit.INT1E=1;
        // Enable CPU INT1 for ADC1_INT:
              IER |= M_INT1;
        // Enable global Interrupts and higher priority real-time debug events:
              EINT;   // Enable Global interrupt INTM
              ERTM;  // Enable Global real time interrupt DBGM

              EDIS;
        // IDLE loop. Just sit and loop forever:
              for(;;)  //infinite loop
              {
                      // State machine entry & exit point

        (*Alpha_State_Ptr)();     // jump to an Alpha state (A0,B0,...)
        }

        } //END MAIN CODE

        {
        // loop rate synchronizer for A-tasks
        if(CpuTimer0Regs.TCR.bit.TIF == 1)
        {
              CpuTimer0Regs.TCR.bit.TIF = 1;   // clear flag

              //-----------------------------------------------------------
              (*A_Task_Ptr)();          // jump to an A Task (A1,A2,A3,...)

              VTimer0[0]++;             // virtual timer 0, instance 0 (spare)
              SerialCommsTimer++;
                }
        Alpha_State_Ptr = &B0;          // Comment out to allow only A tasks
         }
        void B0(void)
         {
        // loop rate synchronizer for B-tasks
        if(CpuTimer1Regs.TCR.bit.TIF == 1)
        {
              CpuTimer1Regs.TCR.bit.TIF = 1;                      // clear flag
              (*B_Task_Ptr)();          // jump to a B Task (B1,B2,B3,...)
              //-----------------------------------------------------------
```

```
               VTimer1[0]++;              // virtual timer 1, instance 0 (spare)
                    }
            Alpha_State_Ptr = &C0;        // Allow C state tasks
                    }
             void C0(void)
                    {
        // loop rate synchronizer for C-tasks
        if(CpuTimer2Regs.TCR.bit.TIF == 1)
                  {
               CpuTimer2Regs.TCR.bit.TIF = 1;          // clear flag

               (*C_Task_Ptr)();           // jump to a C Task (C1,C2,C3,...)
               //-----------------------------------------------------------
               VTimer2[0]++;              //virtual timer 2, instance 0 (spare)
                 }
        Alpha_State_Ptr = &A0;     // Back to State A0
               }

//     A - TASKS (executed in every 1 msec)
       void A1(void) // SPARE (not used)

                  {
               if(EPwm1Regs.TZFLG.bit.OST==0x1)
             TripFlagDMC=1;      // Trip on DMC (halt and IPM fault trip )

        //the next time CpuTimer0 'counter' reaches Period value go to A2
          A_Task_Ptr = &A2;
             }

 void A2(void) // SPARE (not used)
          {
        A_Task_Ptr = &A3;
        //--------------------
      }
void A3(void) // SPARE (not used)
        {
        //the next time CpuTimer0 'counter' reaches Period value go to A1
        A_Task_Ptr = &A1;
             }
void B1(void) // Toggle GPIO-00
               {
        B_Task_Ptr = &B2;
        }
  void B2(void) //  SPARE
         {
        B_Task_Ptr = &B3;             }

  void B3(void) //  SPARE
        {
        B_Task_Ptr = &B1;       }

           void C1(void)
              {
        if(EPwm1Regs.TZFLG.bit.OST==0x1)
          { TripFlagDMC=1;
          GpioDataRegs.GPBTOGGLE.bit.GPIO42 = 1;
          }
         if(GpioDataRegs.GPADAT.bit.GPIO15 == 1)
          { TripFlagDMC=1;
          GpioDataRegs.GPBTOGGLE.bit.GPIO44 = 1;
                 }
               GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;

               C_Task_Ptr = &C2;
                  }
        void C2(void) //  SPARE

                     {
               C_Task_Ptr = &C3;

                     }
        void C3(void) //  SPARE
             {
               C_Task_Ptr = &C1;
               }

        //MainISR
        interrupt void MainISR(void)
        {
        // Verifying the ISR
             IsrTicker++;
             rc1.TargetValue = SpeedRef;
             RC_MACRO(rc1)
             rg1.Freq = rc1.SetpointValue;
             RG_MACRO(rg1)

        clarke1.As = _IQmpy2(_IQ12toIQ(AdcResult.ADCRESULT1)-offsetA); // Phase A
        clarke1.Bs = _IQmpy2(_IQ12toIQ(AdcResult.ADCRESULT2)-offsetB); // Phase B

               CLARKE_MACRO(clarke1)

                 park1.Alpha = clarke1.Alpha;
                 park1.Beta  = clarke1.Beta;
                 if(lsw==0) park1.Angle = rg1.Out;
                 else park1.Angle = semras.ThetaFlux;
                 park1.Sine = _IQsinPU(park1.Angle);
                 park1.Cosine = _IQcosPU(park1.Angle);
                 PARK_MACRO(park1)

        //  Connect inputs of the PI module and call the PI SPD controller macro
              if (SpeedLoopCount==SpeedLoopPrescaler)
                 {
                  pi_spd.Ref = rc1.SetpointValue;
              pi_spd.Fbk = semras.speedes;
                 PI_MACRO(pi_spd)
              SpeedLoopCount=1;
             }
               else SpeedLoopCount++;

               if(lsw==0)   {pi_spd.ui=0; pi_spd.i1=0;}
```

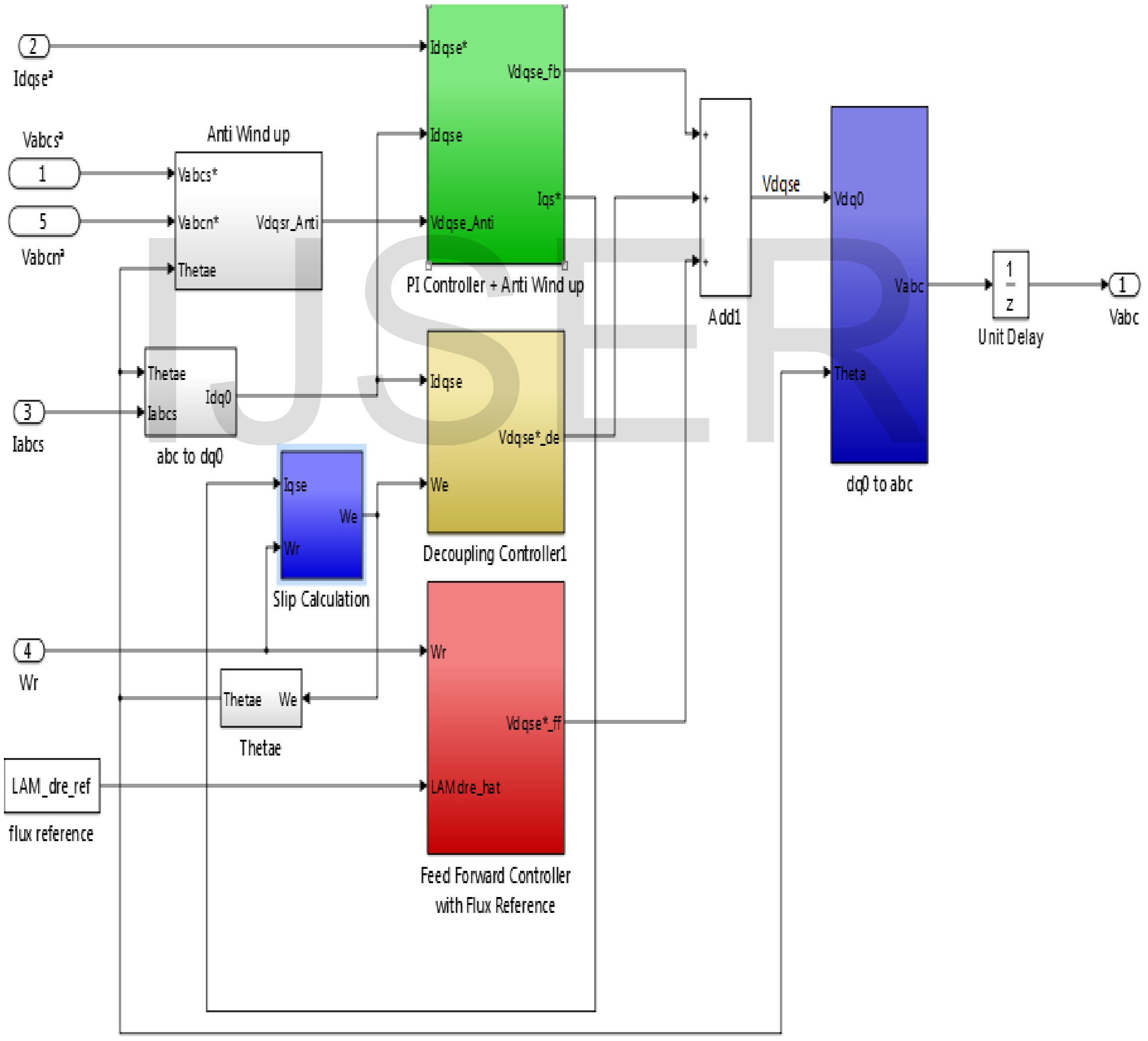
```
//  Connect inputs of the PI module and call the PI ID controller macro
    if(lsw==0) pi_iq.Ref = IqRef;
    else pi_iq.Ref =  pi_spd.Out;
      pi_iq.Fbk = park1.Qs;
      PI_MACRO(pi_iq)

//  Connect inputs of the PI module and call the PI ID controller macro
      pi_id.Ref = IdRef;
      pi_id.Fbk = park1.Ds;
      PI_MACRO(pi_id)

//Connect inputs of the INV_PARK module and call the inverse park trans.
      ipark1.Ds = pi_id.Out;
      ipark1.Qs = pi_iq.Out ;
      ipark1.Sine   = park1.Sine;
      ipark1.Cosine = park1.Cosine;
      IPARK_MACRO(ipark1)

//  Connect inputs of the VOLT_CALC module and call the phase voltage
        volt1.DcBusVolt =_IQ12toIQ(AdcResult.ADCRESULT7);// DC Bus v
          volt1.MfuncV1 = svgen1.Ta;
          volt1.MfuncV2 = svgen1.Tb;
          volt1.MfuncV3 = svgen1.Tc;
          PHASEVOLT_MACRO(volt1)


//Connect inputs of the ACI module and call the speed estimation macro
          semras.Ualpha = volt1.Valpha;
          semras.Ubeta = volt1.Vbeta;
          semras.IAlpha = clarke1.Alpha;
          semras.IBeta = clarke1.Beta;
          ACIMRAS_MACRO(semras)

//  Connect inputs of the SVGEN_DQ module and call the space-vector gen. m
             svgen1.Ualpha = ipark1.Alpha;
             svgen1.Ubeta  = ipark1.Beta;
             SVGENDQ_MACRO(svgen1)

//  Connect inputs of the PWM_DRV module and call the PWM signal gen.
          pwm1.MfuncC1 = svgen1.Ta;
          pwm1.MfuncC2 = svgen1.Tb;
          pwm1.MfuncC3 = svgen1.Tc;
          PWM_MACRO(1,2,3,pwm1)
                                                //
//    Connect inputs of the PWMDAC module
           pwmdac1.MfuncC1 = volt1.Valpha;
          pwmdac1.MfuncC2 = clarke1.Beta;
           PWMDAC_MACRO(6,pwmdac1)

           pwmdac1.MfuncC1 =semras.ThetaFlux;
           pwmdac1.MfuncC1 = semras.speedes;
           PWMDAC_MACRO(7,pwmdac1)
//    Connect inputs of the DATALOG module
            DlogCh1 = (int16)_IQtoIQ15(ipark1.Alpha);
```

```
DlogCh2 = (int16)_IQtoIQ15(ipark1.Beta);
DlogCh3 = (int16)_IQtoIQ15(semras.ThetaFlux);
DlogCh4 = (int16)_IQtoIQ15(semras.speedes);
//    Call the DATALOG update function.
dlog.update(&dlog);
// Enable more interrupts from this timer
AdcRegs.ADCINTFLG.bit.ADCINT1=1;
// Acknowledge interrupt to recieve more interrupts from PIE group 3
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;

}// MainISR Ends Here

interrupt void OffsetISR(void)
{
// Verifying the ISR
IsrTicker++;
// DC offset measurement for ADC
if (IsrTicker>=5000)
{
offsetA=_IQmpy(K1,offsetA)+_IQmpy(K2,_IQ12toIQ(AdcResult.ADCRESULT1
));          //Phase A offset
offsetB=_IQmpy(K1,offsetB)+_IQmpy(K2,_IQ12toIQ(AdcResult.ADCRESULT2
));               //Phase B offset
offsetC=_IQmpy(K1,offsetC)+_IQmpy(K2,_IQ12toIQ(AdcResult.ADCRESULT3
));               //Phase C offset

}

if (IsrTicker > 20000)
{
EALLOW;
PieVectTable.ADCINT1=&MainISR;
EDIS;
}
// Enable more interrupts from this timer
AdcRegs.ADCINTFLG.bit.ADCINT1=1;

// Acknowledge interrupt to recieve more interrupts from PIE group
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;

}
void HVDMC_Protection(void)
{
EALLOW;
// CPU Halt Trip
EPwm1Regs.TZSEL.bit.CBC6=0x1;
EPwm2Regs.TZSEL.bit.CBC6=0x1;
EPwm3Regs.TZSEL.bit.CBC6=0x1;

EPwm1Regs.TZSEL.bit.OSHT1   = 1;  //enable TZ1 for OSHT
EPwm2Regs.TZSEL.bit.OSHT1   = 1;  //enable TZ1 for OSHT
EPwm3Regs.TZSEL.bit.OSHT1   = 1;  //enable TZ1 for OSHT

EPwm1Regs.TZCTL.bit.TZA = TZ_FORCE_LO; // EPWMxA will go low
EPwm1Regs.TZCTL.bit.TZB = TZ_FORCE_LO; // EPWMxB will go low
```

```
   EPwm2Regs.TZCTL.bit.TZA = TZ_FORCE_LO; // EPWMxA will go low
   EPwm2Regs.TZCTL.bit.TZB = TZ_FORCE_LO; // EPWMxB will go low
   EPwm3Regs.TZCTL.bit.TZA = TZ_FORCE_LO; // EPWMxA will go low
   EPwm3Regs.TZCTL.bit.TZB = TZ_FORCE_LO; // EPWMxB will go low
   EDIS;
// Clear any spurious OV trip
  EPwm1Regs.TZCLR.bit.OST = 1;
  EPwm2Regs.TZCLR.bit.OST = 1;
  EPwm3Regs.TZCLR.bit.OST = 1;
                             }
```

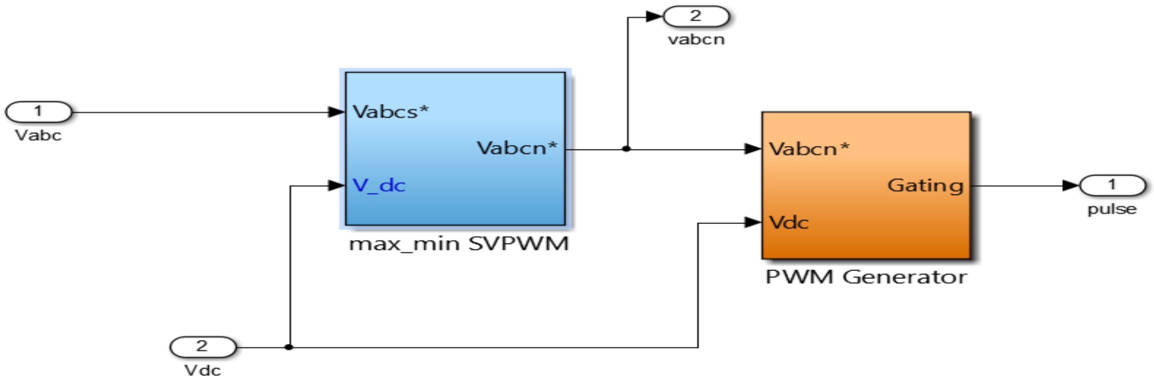
Appendix B

Matlab Simulink of the indirect vector control

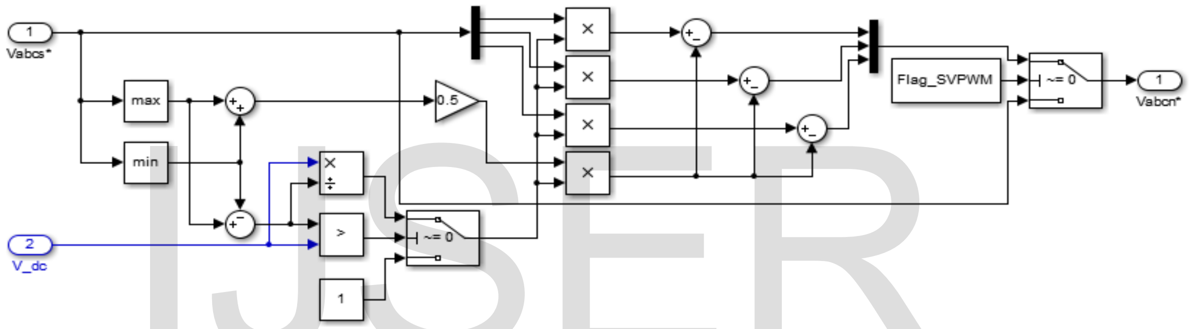


IJSER

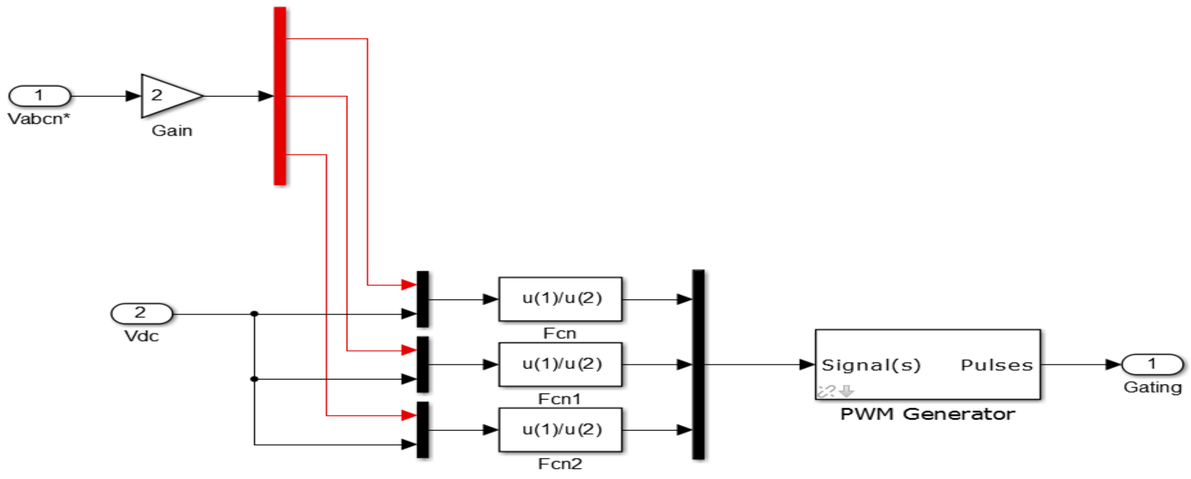
Matlab Simulink model for SVPWM



(a) General block diagram of Max min svpwm



(b) Max min SVPWM detailed design



(c) Detailed design of PWM generator

International Journal of Scientific & Engineering Research Volume 8, Issue 6, June-2017
ISSN 2229-5518

Functional block diagram of TMS320 F28035 Control card

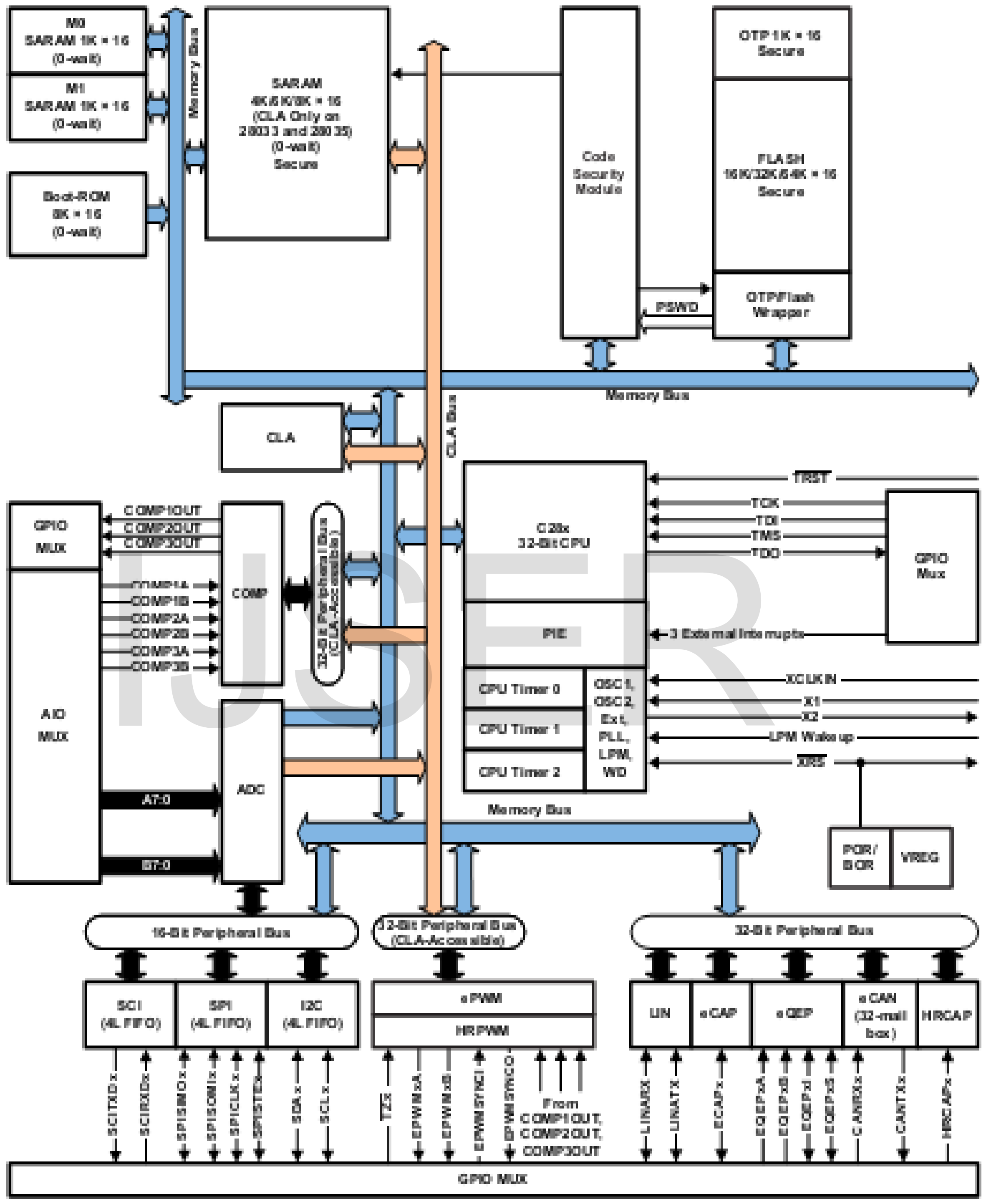


Figure A. Function block diagram of TMS320 F28035 control card

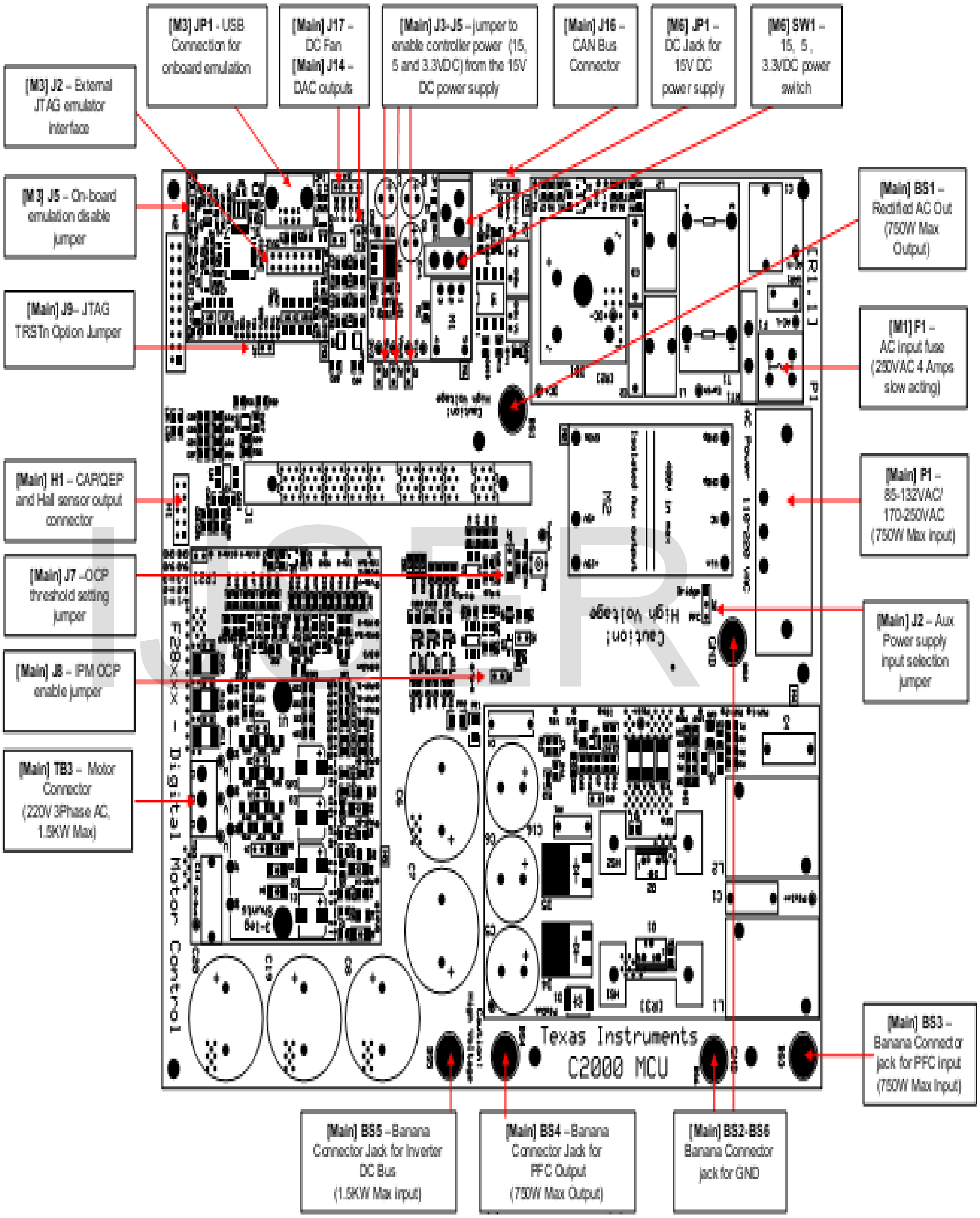

[M3] JP1 - USB Connection for onboard emulation
[Main] J17 – DC Fan
[Main] J14 – DAC outputs
[Main] J3-J5 – jumper to enable controller power (15, 5 and 3.3VDC) from the 15V DC power supply
[Main] J16 – CAN Bus Connector
[M6] JP1 – DC Jack for 15V DC power supply
[M6] SW1 – 15, 5, 3.3VDC power switch
[M3] J2 – External JTAG emulator interface
[M3] J5 – On-board emulation disable jumper
[Main] J9- JTAG TRSTn Option Jumper
[Main] BS1 – Rectified AC Out (750W Max Output)
[M1] F1 – AC input fuse (250VAC 4 Amps slow acting)
[Main] H1 – CAP/QEP and Hall sensor output connector
[Main] P1 – 85-132VAC/ 170-250VAC (750W Max input)
[Main] J7 – OCP threshold setting jumper
[Main] J2 – Aux Power supply input selection jumper
[Main] J8 – IPM OCP enable jumper
[Main] TB3 – Motor Connector (220V 3Phase AC, 1.5KW Max)
[Main] BS3 – Banana Connector jack for PFC input (750W Max Input)
[Main] BS5 – Banana Connector Jack for Inverter DC Bus (1.5KW Max input)
[Main] BS4 – Banana Connector Jack for PFC Output (750W Max Output)
[Main] BS2-BS6 Banana Connector jack for GND
F28xxx - Digital Motor Control
Texas Instruments
C2000 MCU
Caution! High Voltage
Isolated Aux output
AC Power 110-220 VAC

Figure B. HVMotorCtrl+PFC Kit Jumpers and Connectors Diagram