SENSORLESS SPEED ESTIMATION FOR LONG TERM FLYWHEEL ENERGY STORAGE SYSTEM IN STANDBY MODE

by

Rongqiang Liu

B. Eng., Ryerson University, Toronto, ON, CA, 2011

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the program of

Electrical Engineering

Toronto,Ontario, Canada 2014

© Rongqiang Liu 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for thepurpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

Rongqiang Liu

Sensorless Speed Estimation for Long Term Flywheel Energy Storage System in Standby Mode

Electrical and Computer Engineering

Master of Applied Science

Ryerson University

Toronto 2014

A novel technique for sensorless speed estimation is presented in this thesis for squirrel cage induction machine (SCIM) drived long-term flywheel energy storage system (FESS) in standby mode.

The SCIM model for long-term large-capacity FESS is presented. Based on dynamic model , a hybrid rotor flux observer and speed observer are derived. The hybrid rotor flux observer takes advantages of both the current model and voltage model flux observers by seamlessly incorporating these two models together for a better flux estimation performance even at low speed range. The fundamental speed observer is derived from the dynamic model for speed estimation with a fast response time for a tradeoff of the adaptive capabilities.

In order to observe the speed in standby mode, a modified field-oriented control (FOC) scheme is presented. The hybrid flux observer and speed observer are tested in association with the modified FOC. The proposed control technique adopts approaches in an effort to minimize the impact generated by the excitation and speed estimation process to the FESS.

Simulation and experiments are conducted to verify the feasibility of the proposed speed estimation at the standby mode. It is also observed that a step change of excitation current has a significant impact to the existing FESS. A ramp control for excitation current is added to avoid the possible oscillation of the estimated speed and the disturbance to the FESS. The speed estimation settling time is optimized based on the experiment and simulation.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. David Xu for giving me this opportunity to work on this thesis. Without his patient guidance, encouragement and generous support, this thesis would not be achieved. His insights to Power Electronic designs, passion for research sets an example for me. It has been an honor and pleasure to have him as my advisor during my study at Ryerson University.

Also, I'd like to thank Dr. Srinivas Bhaskar Karanki for sharing his technical knowledge and spending his valuable time during the crucial parts of this research. His technical advice and continuous support during my studies are highly appreciated.

I would also like to express my deep appreciation to Dr. Xiaoqiang Guo for proof reading, to Mr. Bing Gong and Hongxi Li for sharing their programming skills and resources on DSP.

Financial supports from the Office of Center for Urban Energy (CUE) and the Department of Electrical and Computer Engineering are greatly acknowledged.

Table of Contents

Preliminary

List of Figures

List of Tables

List of Symbols

K_E	Total kinetic energy
J	Moment of inertia
ω	Angular velocity
r	Radius
α	Length of the cylinder
m	Mass
ρ	Density
Δt	Time interval
ΔEc	Energy change
e_{dqs}	Back emf stator dq axis component
i_{dqs}	Stator current in dq axis component
i_{ex}	Excitation current
L_m	Magnetizing inductance
L_{ls}	Stator leakage inductance
L_{lr}	Rotor leakage inductance
L_s, L_r	Stator and rotor self inductance
L_m	Mutual inductance
R_s	Stator resistance
R_r'	Rotor resistance referred to stator side
T_{em}	Electromechanical torque

i_m	Magnetizing current
$i_{ds}, i_{qs}, i_{dr}, i_{qr}$	d, q axis stator and rotor current vector
$v_{ds}, v_{qs}, v_{dr}, v_{qr}$	d, q axis stator and rotor voltage vector
$\lambda_{ds,}\lambda_{qs}$	Stator and rotor flux vector
P	Number of pole pairs
p	Derivative operator with respect to time d/dt
$\sigma = 1 - \frac{L_m^2}{L_s L_r}$	Leakage coefficient
$\tau_r = \frac{L_r}{R_r}$	Rotor speed constant
V_{dc}	DC link voltage
ω_r	Rotor electrical speed
ω_e	Synchronous speed
ω_{sl}	Slip speed
λ_r	Rotor flux aligned with synchronous frame
θ_s	Stator current angle
θ_f	Rotor flux angle
θ_r	Rotor angle (rotor position)
θ_{sl}	Slip angle

Acronyms and Abbreviations

AC/DC: Alternating Current/Direct Current

AGC: Automatic Generation Control

CM/VM: Current Model/Voltage Model

DFIM/DFIM: Doubly-Fed Induction Machine/Motor

DSP: Digital Signal Processor

EMF: ElectroMotive Force

FESS: Flywheel Energy Storage System

FOC: Field-Oriented Control

VSI: Voltage Source Inverter

VSC: Voltage Source Converter

UPS: Uninterruptible Power Supply

WRIM: Wound Rotor Induction Machine(Machine/Motor)

SCIM: Squirrel-Cage Induction Machine(Machine/Motor)

PMSM: Permanent Magnet Synchronous Machine(Machine/Motor)

PWM: Pulse Width Modulation

ISO: Independent System Operator

IM/ IG: Induction Motor/Induction machine

SEIG: Self-Excited Induction Machine

VFDs: Variable-Frequency Drives

MRAS: Model Reference Adaptive System

EKF: Extended Kalman Filtering

UKF: Unscented Kalman Filtering

MMF: Magneto Motive Force

SG: Synchronous Machine

SVPWM: Space Vector Pulse Width Modulation

PIC: PI-controller

PSO: Particle Swarm Optimization

PLL: Phase Locked Loop

VCO: Voltage Controlled Oscillator

PD: Phase Detector

LPF: Low Pass Filter

Chapter 1: Introduction

1.1 Background

1.1.1 Evolution of flywheel

The word ‘flywheel’ appeared first during the start of industrial revolution. During that period, flywheels were used mostly for smoothing torque pulses in steam engines. They store kinetic energy, also known as potential energy, within a rapidly spinning wheel and transfer it for use when needed. The primitive flywheels were built with heavier rims towards increasing mass for higher energy storage rather than increasing speed. The higher the mass, the higher the weight and size of the flywheel is. Therefore, it was initially held back by prohibitive weight and unwanted precession forces.

A flywheel with a low mass but a high limit for the mechanical tension gives an optimized weight. It is more effective to increase the speed than the mass to achieve higher density energy storage. The primary limiting factor to flywheel energy storage potential is due to the maximum allowable material stress that materials can achieve. Advances in the field of high-strength composite materials make the composite flywheels operate viable higher speeds, and therefore can store more energy for a given mass than a conventional steel flywheel. New composite material such as carbon fiber and glass fiber with a compound construction has greatly improved the specific tensile with the strength 5 times higher than steel for flywheel rotors and makes the high speed flywheel possible. In fact, the stronger the flywheel’s material is, the higher the

rotational speed and the energy it can store. The specific strength of typical flywheel materials is listed in Table 1-1.

Table 1-1 Specific strengths of flywheel materials [1]

Flywheel material	Specific strength (kJ/kg)
Cast iron	19
Carbon steel	44
Alloy steel	100
Wood (beech)	130
Kevlar	1700
S-glass	1900
Graphite	8900

The advancements in power electronics, high speed machines, vacuum and magnetic bearing and system integration technology have led to the development of light, high-speed flywheel systems. It has only been in the last two decades that flywheel technology has been seriously considered for use in mobile applications, such as hybrid vehicle systems and traction applications that uses FESS instead of an electric battery to accumulate and then release—when needed for acceleration—regenerative braking energy.

1.1.2 Stored Energy of FESS

The total kinetic energy stored in a flywheel is calculated by the equation: [2]

$$K_E = \frac{1}{2} J\omega^2 \tag{1-1}$$

where J is the moment of inertia of the flywheel and ω is the angular velocity. Eqn.(1-1) illustrates that the stored kinetic energy of flywheel is linearly proportional to the flywheel mass and the square of the rotational speed.

The moment of inertia is a physical quantity, which depends on the mass and shape of the flywheel. A solid cylinder will have a moment of inertia equal to

$$J = \frac{1}{2} mr^2 = \frac{1}{2} r^4 a\rho \qquad (1\text{-}2)$$

Where r is the radius, α is the length of the cylinder, m is the mass, and ρ is the density of the wheel material.

Tensile strength of different material determines its maximum rotating speed, and therefore, the difference in unit energy storage capability. A comparison of typical flywheel energy storage capabilities is listed in Table 1-2. A light-weighted material such as Kevlar 60% fibre can store much higher energy content than maraging steel at the same unit weight.

Table 1-2 Comparison of typical flywheel energy storage capabilities [3]

Material	Density 10^3 kg/m^3	Useful Energy 10^3 J/kg	Mass of the flywheel 10^3 kg
Wood birch	0.55	21.0	1720
Mild steel	7.80	29.5	1220
S-Glass	1.90	70.5	509
Maraging steel	8.00	86.4	417
Carbon 60% fibre	1.55	185.4	194
Kevlar 60% fibre	1.4	274.3	131

For a given time interval Δt, the flywheel can deliver (or absorb) a quantity ΔEc of energy with an average power given by:

$$P_{avg} = \frac{\Delta Ec}{\Delta t} = \frac{1}{2} J \frac{(\omega_2^2 - \omega_1^2)}{\Delta t} \quad (1\text{-}3)$$

The FESS is a kind of clean energy storage system and has lots of advantages:

1) High power and energy density with high-speed of charge/discharge performance.

2) Environment friendly with no direct emission, and contains no acids or other hazardous materials.

3) Long service life. The flywheel bearing systems and rotating parts require no maintenance. Flywheels are designed and built for 20 years of virtually maintenance-free operation.

4) Wide operating temperature range. It is relatively insensitive to temperature fluctuation and environmental conditions and can operate between -20℃ to +40℃.

5) No limit on repeat charge/discharge counts. The flywheel is not affected by repeated deep discharges cycles. Almost all other forms of energy storage have very limited cyclic capabilities.

6) Guaranteed energy content. The amount of energy stored in a flywheel is readily determined from its rotational speed. The speed is continuously monitored by the electronics so if ever the state of charge of the flywheel deteriorates, the electronics can immediately provide a warning signal.

Modern flywheels use a motor to spin the rotor and convert the kinetic energy to electrical energy. Most modern high-speed flywheel energy storage systems consist of a massive rotating cylinder (a rim attached to a shaft) that is supported on a stator by magnetically levitated bearings. To maintain efficiency, the flywheel system is operated in a vacuum chamber to reduce

drag. The flywheel is connected to a motor-machine that interacts with the utility grid through advanced power electronics.

1.2 FESS applications

Typically flywheels are considered to be pulse power devices which compete well with ultra-capacitors or high power lithium-ion batteries. The very first modern flywheel systems were designed to provide uninterruptible power supply (UPS) and very large pulses of electricity for scientific or industrial use. Long-term energy storage flywheels have also been developed. There are already used for commercial applications and other potential applications identified such as space applications, Uninterruptible Power Supply (UPS), grid quality enhancement, hybrid/electric vehicles, peak power shaving in grid for integration of emerging renewable energy sources, etc.

A successful application of Beacon's 20-megawatt grid frequency regulation facility in Stephentown, NY, has been delivering frequency regulation services since early 2011 [4]. The 20-megawatt Stephentown facility in New York "boasts 200 flywheels operating in parallel to provide 20 megawatts of up-regulation and 20 megawatts of down-regulation in immediate response to the ISO's AGC [Automatic Generation Control] signal." The plant has been in commercial operation for two years and has provided more than 250,000 megawatt-hours of frequency regulation service.

FESS is also replacing the industrial batteries used by mission-critical data centers, hospitals and semiconductor fabrication facilities. A 30 MW flywheel for semiconductor fabrication

facility in Germany can absorb 5 MW for 5 seconds which allows the plant to switch from one power source to the backup power supply. [5]

Importantly, the addition of a FESS in power network lowers the peak power requirements which save energy during idle periods for the present power networks. Fuel consumption went down with significant reductions in NO_2 and particulate matter (PM) emissions while the power generation capacity increased for peak periods. This benefit can also extend to the renewable energy generation. The stability of electricity generation is at risk due to the intermittent characteristic of the renewable energy source such as wind or solar energy for its unpredictability. Large scale long term energy storage system such as FESS is required to achieve higher penetration ratio when renewable sources are integrated into the conventional power network [6].

1.3 Requirement of FESS in power systems

Power quality issues have been a top concern for modern power generation and distribution systems. Seasonal loads and electrical distortions make the power system unstable and generate the power quality problems: the instantaneous interruption, voltage sag, flickers, short-term power-off [7], lack of reactive power or active power [8], and etc. In order to solve these problems, variety of energy storage systems are applied with different terminologies. Conventional energy storage systems include pumped storage hydropower, thermal energy storage, hydrogen, modern fossil fuel based power plants (and especially natural gas combined cycles) and rechargeable flow batteries, etc. Among conventional energy storage devices for power conditioning, batteries are too expensive and don't last long enough; pumped hydro is

cheap but not feasible for most locations; thermal storage is promising but still too expensive or hard to scale. Natural gas is the major fuel for electricity production in the frequency regulation service, and natural gas power plants have a very high efficiency (above 60% for the best available technology), a very high flexibility and low CO2 emissions (reduces the CO2 emissions per kWh up to 80% if compare with an old coal fired power plant. Depending on the location at which electricity is produced, transported, consumed and held in reserve (back-up), storage can be large-scale (GW), medium-sized (MW) or micro, local systems (kW).

The advancement of related technologies makes flywheel a promising candidate for power conditioning. The power rating for flywheel energy storage systems ranges from kW to MW [9]. Technical requirements of FESS in power systems include:

a) Fast response of energy storage system is a key requirement in power conditioning

Surveys have shown that the majority of voltage sags and momentary interruptions in power systems last less than 3 seconds, 80% of power line disturbances last for less than a second; therefore, fast response energy storage systems are required to effectively provide a short burst of power to stabilize the power system. FESS can be used as energy buffers storing or retrieving energy to supply the load during the periods of low or no power output in its supported power network. Fast response FESS can compensate and make the power supply stability greatly improved. Due to technological advancements in material science and emergence of power electronics, the high speed FESS has made it a viable solution to power quality problems.

b) Sustainable green energy provides an enviro-friendly solution to renewable energy storage.

Sustainability is a philosophy and set of practices that have gained considerable acceptance among global organizations in recent years. The concept of sustainability addresses the economic, social and environmental impacts of operations. The flywheel-based energy storage systems, unlike fossil-fuel power plants or batteries, are sustainable "green" technology solutions that consume no fossil fuel, nor produce CO_2 or other emissions during operation. UPS system is often used to improve the reliability of power supply. However, conventional lead-acid battery based UPS system in power conditioning system has some drawbacks of low efficiency, gas pollution, short life time, low-speed of charge/discharge, huge size and weight. FESS is also highly scalable, and have fewer environmental impacts compared to batteries. In addition, the application of FESS also allows a more efficient way of storing energy for the expansion of renewable energy generating systems such as wind and solar.

c) Economic consideration from energy storage system.

Flywheels have a product life of more than 20 years and incur low annual operating and maintenance costs. Industrial batteries are less expensive initially than a flywheel, but when frequent battery replacement is factored; a flywheel-based solution can be considerably less expensive. For the International Space Station in particular, nickel-hydrogen batteries are currently being used with a rating of 38,000 cycles and 6.5 year lifetime. A flywheel can be used with the same space and weight, but the flywheel would last 3-10 times longer. The cost savings is estimated by NASA to be $200 million. The flywheel would be 93.7% efficient.

d) Large energy storage system with long service life.

Flywheel energy storage system can be interconnected in parallel, or matrix, to provide energy storage for certain utility applications to achieve high power ratings. With the advancement of the FESS technology, flywheel systems can be a promising energy storage alternative to provide consistent, dependable energy for a variety of important applications through the use of a high speed electric motor/generator. Flywheels can deliver large amounts of energy in a short span of time, and require little to no maintenance over their lifetime. A primary advantage of flywheels is the long lifetime of 15 to 20 years. This is evidenced by the fact that a composite flywheel has 90,000 cycles at high speeds of 48,000 rpm with no evident degradation.

1.4 Motivation and Objective

FESS is a feasible candidate for high power long-term energy storage. It has a unique advantage over batteries or super capacitors since its stored energy can be exactly determined by monitoring its angular velocity for a given flywheel. Usually, a speed sensor is required to be mounted on the shaft to measure the motor speed. However, a speed sensor increases the cost and requires a connection line between the control system and the motor. Speed sensor failures are frequent, and are mainly due to the extremely harsh operating conditions. Therefore, the reliability of the high speed sensor is low when the whole flywheel (including motor) is floating by magnetic bearing in a vacuum tank. To avoid the problems of the speed sensors, sensorless control schemes have been developed with the several implementation methodologies, including the back-EMF method [10], slip frequency calculation method; [11] [12]; speed estimation using state equation [13]; estimation based on slot space harmonic voltages [14]; Kalman filtering

techniques [15]; neural network based or Fuzzy-logic based sensorless control [16]; MRAS observer [17]; high frequency injection technique [18] and etc. The main objective of these sensorless speed estimation schemes is to obtain the flywheel speed and position information without the use of an extra sensor. Therefore, the dimensions and cost of the drive system can be further reduced. However, all these schemes are based on on-line estimation strategies when the flywheel is charging/discharging the energy. However, during the standby mode when all gating signals are disabled from its converter, the speed information is lost.

Generally, most FESS are classified as short-term high power density energy storage devices but with low energy content which can only hold energy for seconds to minutes to provide ride-through power and voltage stabilization for power quality and power recycling applications. Therefore, FESS are best used for high power, low energy applications such as UPS or voltage sag [19] compensation that require many cycles. The long-term FESS aims to significantly increase its storage time (12 hours at 80% load efficiency) [20]. In order to achieve this goal, the iron loss and friction loss of the FESS drive should be avoided. Moreover, the impact caused by this proposed sensorless speed estimation process to the long-term FESS should be minimized in standby mode. All the design techniques applied in this research are based on this principle.

IM are relatively rugged, reliable and inexpensive machines. They are increasingly used with variable-frequency drives (VFDs) in variable-speed service offering especially important energy savings opportunities. As the result, IMs are widely used in industries, and almost 95% is of squirrel cage type [21]. SCIM has many advantages: First of all, it is less expensive. Next, it has a very compact structure and insensitive to environment. Furthermore, it does not require

periodic maintenance for brush replacement like DC motors. Last but not least, SCIM offers the opportunity for FESS to work in high speed range with zero iron loss in standby mode, which offers the best possibility to develop high-speed long-term FESS to provide higher energy content and power density.

This thesis presents the modeling, simulation and experimental analysis of sensorless speed estimation scheme for long-term FESS at standby mode. In standby mode, if the motor is kept energized, FESS will slow down gradually due to its iron loss for continuous online speed monitoring. In order to reduce the power loss, the FESS is disconnected from its power converter by shutting down the gating signals when the flywheel is not in its charge/discharge mode. In this way, the iron loss of the IM is zero. This proposed speed estimation control scheme focuses on the standby mode of FESS, but it also works well in charging and discharging modes with minor modification to the conventional field-oriented control. Since the speed of a FESS will not change dramatically at standby mode due to its large inertia, the speed can be monitored periodically on a regular basis without any deterioration of its performance. The machine of FESS will be energized in very short duration when speed and position information are required. In order to obtain the speed information during standby mode, modifications of the conventional field-oriented control are required to make the algorithms suitable for standby mode.

Unlike other sensorless speed estimation techniques working on on-line conditions in which the speed can be continuously monitored, this proposed sensorless speed estimation technique focuses on the standby mode of FESS in which the speed is estimated periodically when needed. Therefore, approaches must be taken in order to achieve a quick response time

and settling time as well as the minimization of power loss due to the intermittent pattern of estimation process.

1.5 Thesis Outline, Overview of the chapters

Chapter 1 provides an overall review of the flywheel background, applications and the requirements for power conditioning of FESS. In chapter 2, the selection of the machine for long-term FESS and the requirement of the proposed sensorless speed estimation are presented. In this chapter, the sensorless speed estimation topologies are reviewed for the derivation for this standby mode speed monitoring. In chapter 3, the SCIM circuit, hybrid rotor flux observer, sensorless speed estimation observer and the modified field-oriented control (FOC) are derived in an effort to achieve the sensorless speed estimation at standby mode with the minimized disturbance to the FESS. The hybrid rotor flux observer for field-oriented control is presented to work in current mode at low speed and voltage mode at high speed. In chapter 4, the setup of the simulation and experiment hardware are discussed. The simulation and experimental results of the proposed estimated speed at the standby mode are verified in this chapter. In addition, the relationship of the settling time with the excitation current ramp control is also discussed; in chapter 5, the future work will include a detailed analysis of a FESS for validating the simulation model with a laboratory flywheel interfaced to an analog model power system will be covered.

Chapter 2:Motor Drive Systems for Flywheel

2.1 Configuration of FESS

A flywheel energy storage system is a typical IM drive system, which contains three main components: an IM, a power converter and a controller.

FESS itself is a kinetic or mechanical battery, spinning at very high speeds to store energy that is instantly available when needed. The flywheel system comprises a flywheel, an IM and an AC/DC converter (rectifier/inverter), which controls the speed of the flywheel and therefore, the exchanged power. The IM is used for energy conversion. The FESS requires the IM working both as motors and machines. The power electronic interface consists of two voltage sourced converters (VSC) connected through a common DC link. A typical FESS application is shown in Fig. 2-1 .

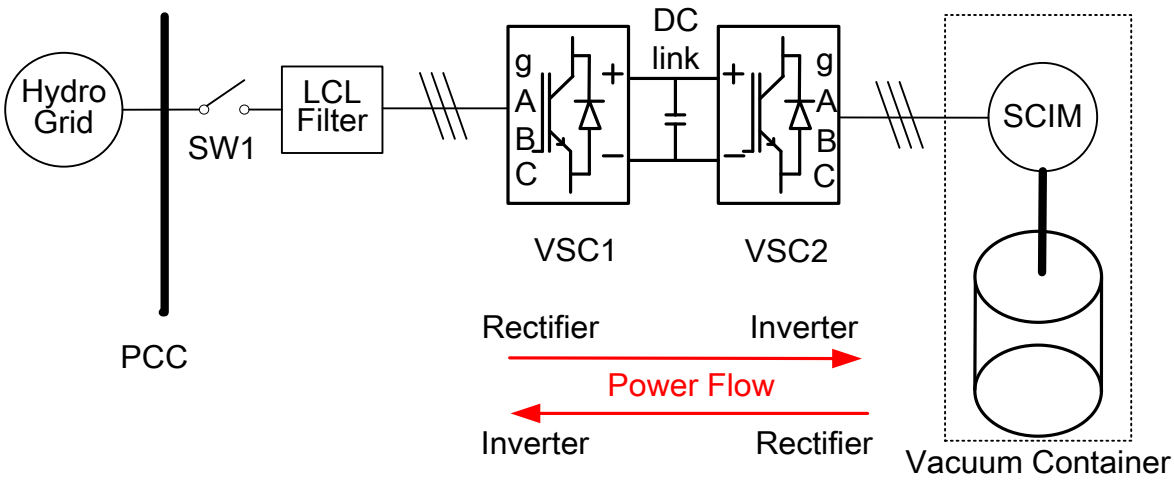


Fig. 2-1 Circuit Diagram of FESS setup

2.1.1 Definition of the working mode of FESS

A FESS works in three modes: charge, discharge and standby mode. The FESS is coupled into the grid via a back-to-back converter and LCL filter. The voltage source converter 1 (VSC1) is connected to grid through the filter to obtain a high performance for the grid side PQ regulation. VSC2 is used to control the operation of the SCIM as a motor/machine depending on the system conditions. VSC1 and VSC2 are coupled through DC link. In the charge, discharge and standby mode, the SW1 remains closed.

The DC link exists between the back-to-back converters. On one hand, the utility connection is rectified into a high voltage DC. On the other hand, that DC is switched to generate a new AC power waveform. It's a link because it connects the input and output stages. The term "DC link" is also used to describe the decoupling capacitor in the DC link. The power stored in the DC-link can also works as a power source for the short excitation during the sensorless speed estimation period. By regulating the DC-link voltage from VSC2, a relative constant DC voltage is achieved on the DC link capacitor.

In the charging mode, power grid injects energy into the flywheel through the bi-directional converter which results in a higher DC link voltage than the set value. By regulating this DC link voltage Vdc to its set value, the power is transferred to the FESS.

In discharging mode, the grid extracts energy from DC link and the DC link voltage starts to drop. Then FESS provides energy to the DC link from FESS by regulating the DC link voltage to its set value.

In standby mode, When DC link voltage closes to its rated, and there is no charging or discharging command from grid, the flywheel will enter the standby mode after certain time

with SW1 remain closed. For this proposed standby mode speed estimation approach, the FESS enters into standby mode with all gating signals disabled for VSC2 in an effort to minimize the power loss during the standby mode. By open the SW1, the FESS goes to offline which is disconnected from the power grid.

2.1.2 Function of the LCL filter

The use of power converters is very important in maximizing the power transfer from FESS to the utility grid. A LCL filter is often used to interconnect an inverter to the utility grid in order to filter the harmonics produced by the inverter.

- Reduction of the high switching frequency ripple impact on the grid
- Elimination of harmonics from PWM signals at the switching frequency
- Elimination of high level of dv/dt ratio fast-switching transistors created.
- Increased machine insulation lifetime for small machines
- Reduced cabling cost for large machines
- Increased heat dissipation capabilities for machines

2.1.3 Structure of the back-to-back converter

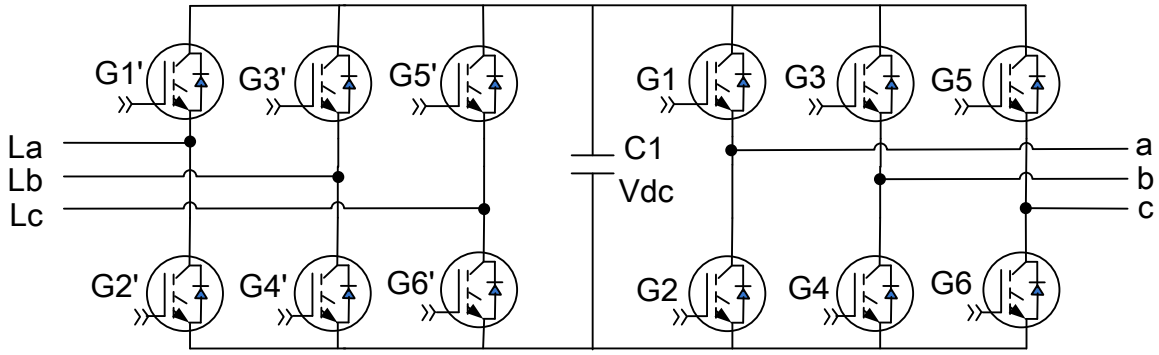


Fig. 2-2 3-Phase Back-to-Back Power Converter

The back-to-back converter (inverter/rectifier) consists of three half-bridge units where the upper and lower switches are controlled complementarily; meaning when the upper one is turned on, the lower one must be turned off, and vice versa. Because the power device's turn-off time is longer than its turn-on time, the dead time must be inserted between turning off one transistor of the half-bridge, and turning on its complementary device. The output voltage is mostly created by a Pulse Width Modulation (PWM) technique. The 3-phase voltage waves are shifted 120^{o} to one another, thus a 3-phase motor can be supplied. The three phase voltage source inverter is implemented for both the simulation and experiment in this research.

2.1.4 Design requirement of proposed scheme:

In order to design and evaluate the proposed scheme, the evaluation standard is setup:

1) The excitation duration should be as less as possible so that the disturbance to the existing FESS system can be minimized.

2) The speed estimation algorithm should have a quick response time and a settling time with acceptable estimated speed accuracy.

3) The generated torque due to the excitation process should be minimized.

2.2 Selection of excitation for the SCIM

The stator voltages and currents in a healthy 3-phase IM contain speed and position information. By properly processing samples of these quantities, the speed and position information can be extracted. In order to achieve the standby mode speed estimation from the terminal parameters of FESS, the SCIM needs to be magnetized for a short duration so that voltage and current will be induced at its stator terminals. Except for permanent magnet

machines, a machine produces output voltage proportional to the magnetic field, which is proportional to the excitation current; if there is no excitation current, there is zero voltage. Generally the excitation comes in the following ways:

1) The residual flux. The residual flux is originated from some degree of the magnetic field that was applied previously to the induction machine. The rotor iron retains a residual magnetism when the machine is turned off. If the machine is started with no load connected; the initial weak field creates a weak voltage in the stator coils, which in turn increases the field current, until the machine "builds up" to full voltage. Typically, in healthy rotors, little residual flux will remain after the motor is de-energized [22]. Therefore, this approach will not be capable for proper operation of the proposed speed estimation process.

If the machine does not have enough residual magnetism to build up to full voltage, usually provision is made to inject current into the rotor from another source. This may be a battery, capacitor, or rectified current from a source of alternating current power.

2) Self-Excitation. A major characteristic of IM is the need for excitation of the magnetic field via the supply terminals. The magnetization reactance demands reactive current from the grid. Without the supply of reactive power, the IM cannot produce any active power. When the flywheel is rotating at the standby mode, an excitation capacitor can be connected across the stator terminals to form a Self-excited induction generator (SEIG). The IM will self-excite, using the external capacitor, only if the rotor has an adequate remnant magnetic field. The induced electromotive force (EMF) and current in the stator windings will continue to rise until the steady-state condition is reached. This induced voltage and current will be used to compute the

rotor speed for the FESS at standby mode. A typical diagram of self-excited induction machine is shown in Fig. 2-3.

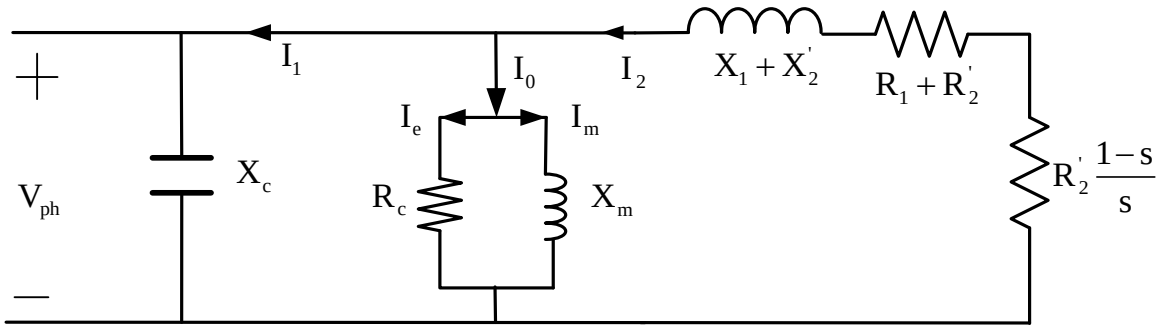


Fig. 2-3 Self-excited induction machine with external capacitor

The drawback of this method is the requirement for the residual flux and the extra mechanism for the external capacitor to be connected to the induction machine and the proper capacitance setup.

3) Excitation from ac power converter. In standby mode, the SW1 in Fig. 2-1 remains closed. Under the variable-speed operation, a bi-directional power converter, generally a back-to-back converter is required to interface the induction machine and the grid. In order to generate the required magnetization filed for speed estimation, the stator coil works as a magnetization coil for the pulse excitation during the speed estimation period. The power is supplied from the DC link in standby mode. As the field-oriented control provides the decoupled control of excitation and torque, this approach is applied in this research with the excitation level fully controlled.

2.3 Power Source for excitation

The structure of the popular back-to-back converters with DC voltage link shows improved reliability and allows a greater output voltage. The capacitor in the dc link provides decoupling function so that the two converters can be driven independently according to usual PWM techniques providing excellent input and output performances. Therefore the DC-link also works as the energy buffer, and stores the amount of energy needed to keep the DC voltage ripple below a suitable limit. For most proposed FESS, the constant dc voltage is regulated by the flywheel side converter [23] [24]. When the DC-Link voltage Vdc decreases, the induction machine is controlled to operate as a generator, transforming the inertial energy stored in the flywheel into electrical energy supplied to the capacitors. When Vdc increases, the induction machine is working as a motor, transferring energy from the capacitors to the flywheel. Therefore for most proposed FESS control schemes, the aim of the DC-link is to control the voltage which is regulated by the FESS side control against fluctuations and maintain relatively constant energy storage in the DC-link. For this proposed sensorless speed estimation, since the DC-link voltage is regulated by FESS, and the excitation current is supplied from the DC-link during the short-duration estimation period, a small amount of energy is withdrawn from the DC-link capacitor for the excitation of SCIM in standby mode. If the DC-link voltage drops below its rated value, the energy can be drawn from the flywheel to the DC-link to maintain a relatively constant voltage and energy content.

2.4 Selection of the machine for long-term FESS

Flywheel energy storage systems can utilize all types of AC three-phase machines. The choice of the machine type is determined by the energy storage application and particularly by expected duration of energy storage. Flywheels are operated at high and variable speed. Each has specific requirements leading to a variety of electric machine topologies. The working condition requires the FESS to work as both a motor and machine during its operation time. Motor/machines are really one device that can run in two opposite modes. Most commonly used electric machines are presented in Fig. 2-4. Depending on the construction and operating principle, Electric machines are divided in two main groups: Asynchronous (Induction machines (IM) and Synchronous machines (SG) [25].

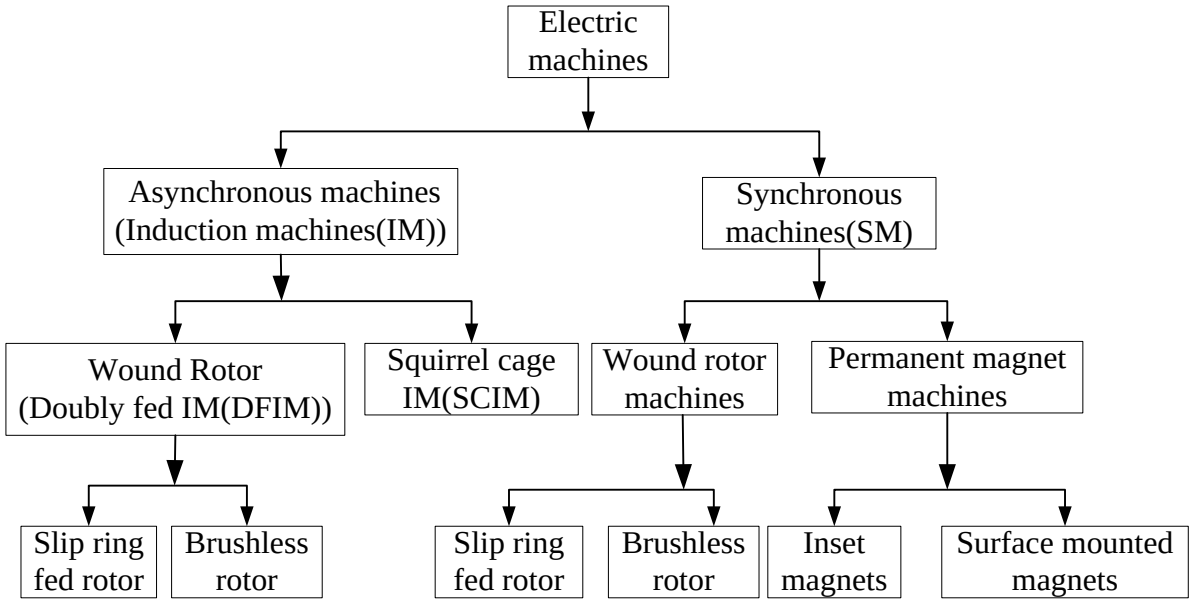


Fig. 2-4 Classification of commonly used electric machines

There are three most common types of electrical machines employed in FESS, such as wound rotor induction machine (WRIM), squirrel-cage (SCIM) and permanent magnet synchronous machine (PMSM) [2]. Each of machines has its unique operation characteristics. Hence, the

design of power electronic and control scheme varies with respect to the specific type of machine.

The IM is the most widely used electrical motor. Almost 80% of the mechanical power used by industries is provided by three phase induction motors because of its simple and rugged construction, low cost, good operating characteristics, etc. IMs are asynchronous machines which require no mechanical commutation, separate-excitation or self-excitation as in universal, DC and large synchronous motors. IMs are generally classified into two categories, squirrel cage and asynchronous wound rotor (DFIM). Under the variable-speed operation, a bi-direction power converter, which is commonly a back-to-back converter, is required to interface the induction machine and the grid. The major disadvantage of IM is the need for excitation of the magnetic field via the supply terminals.

2.4.1 Wound rotor induction machine (DFIM)

Wound rotor induction machine, is also known as Doubly-Fed Induction Machine (DFIM). The FESS DFIM can be regarded as a doubly-fed induction machine without prime mover or a doubly-fed induction motor without mechanical load, so the control method and application technology of DFIM proposed in literature is also suitable for FESS DFIM [26] [27]. The speed control of the DFIM makes it possible to achieve stable variable-speed operation. Adjusting the rotor speed makes the machine/motor either release the kinetic energy to the power system or absorb it from the power system. Thus, the machine-motor has the capability of achieving, not only reactive power control, but also active power control based on a flywheel effect of the rotor. The ac excitation on the basis of a rotor-position feedback loop makes it possible to achieve stable variable-speed operation. The stator is connected directly to the lines; the rotor is

connected through back-to-back VSI to the same line. When the rotor frequency is higher than the stator frequency ($f_L = 50Hz$), the DFIM is working as a machine; conversely, the DFIM is working as a motor. The typical configuration of a DFIM flywheel drive is shown in Fig. 2-5 [25]

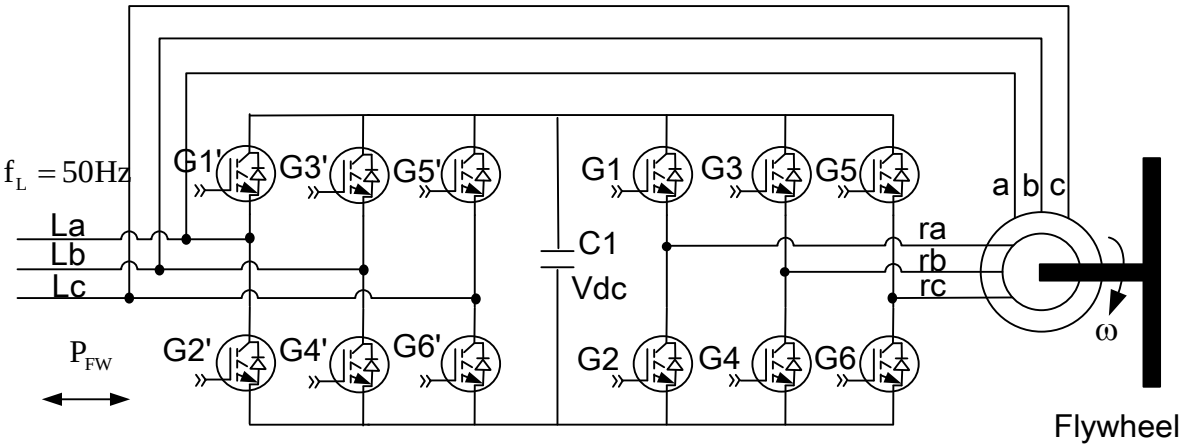


Fig. 2-5 Doubly-fed slip ring induction machine flywheel drive

2.4.2 Squirrel Cage Induction Machine (SCIM)

The SCIM has internally shorted rotor circuits (rotor bars) and therefore there is no external connection to the rotor. This construction makes the SCIM rugged, reliable and economical than DFIM. The simple way to use it as an autonomous machine is to connect its stator windings to a capacitor bank in parallel to the load. As discussed above, the remaining magnetic flux, added to the magnetizing current through the capacitor bank yields the built up of the electromotive force and its increase to a useful value. Another way to achieve an autonomous operating is to connect the stator windings to a rectifier/inverter. In this case, the device has to be controlled in order to maintain the DC voltage from the FESS. The rotor flux oriented control is used to maintain the terminal voltage constant. [2]

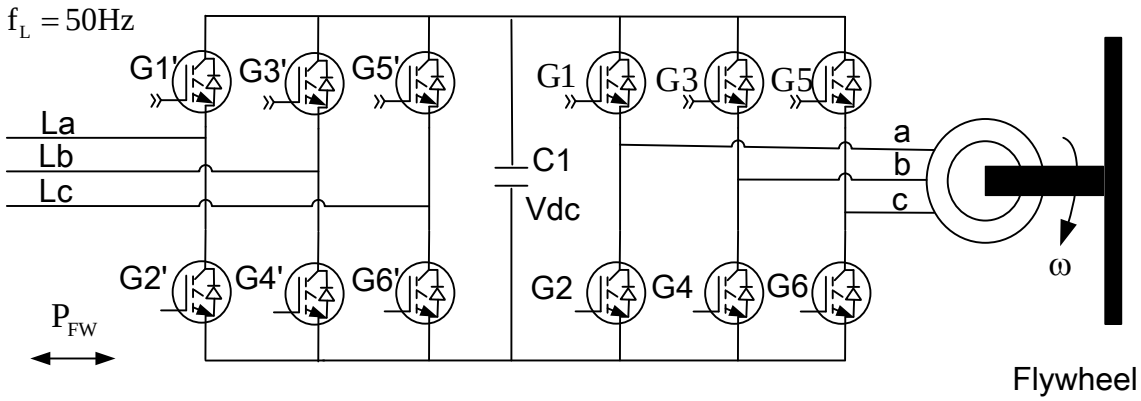


Fig. 2-6 Squirrel cage induction machine flywheel drive

2.4.3 Permanent Magnet Synchronous Machine (PMSM)

The power circuit of the PMSM is the same as for the squirrel cage induction machine. Most PMSMs utilize permanent magnets which are mounted on the surface of the rotor, thus eliminating electrical losses in the rotor circuit. Moreover, the PMSM can achieve high speed operation due to its mechanical robust rotor construction, resulting in higher energy density. This makes the motor appear magnetically "round", and the motor torque is the result of the reactive force between the magnets on the rotor and the electromagnets of the stator. The PMSM based FESS, will induce continuous terminal voltage on the stator to form a closed circuit through the clamp diode on the converter, which results in continuous power loss even if all gating signals are disabled for the converter.

2.4.4 Comparison of the flywheel drives

Most common and promising types of machines use in energy storage systems discussed in this chapter are presented in Table 2-1. In energy storage systems with expected long duration of energy storage idle losses should be radically limited. Such systems can utilize asynchronous induction machines or synchronous machines. During energy charging or discharging a small

amount of energy is needed for the machine excitation (power losses in the field winding resistance in a synchronous machine or losses due to the magnetizing (reactive) component in an induction machine). In energy storage systems intended for relatively short duration storage, permanent magnet machines (synchronous or brushless) can be used. In long-term flywheel energy storage systems with a high rotational speed and, consequently, high frequency of the fundamental component of the machine voltage, the difficulty lies in correct shaping of sinusoidal current waveform obtained by means of PWM modulation. In such a case a correct power supply of a brushless DC machine can be more easily achieved. Permanent magnet machines require no additional energy for excitation but certain small losses occur in them due to currents induced in conducting parts by variable magnetic field of rotating magnets. These losses can be reduced by employing brushless coreless machines. Such machines have very small winding inductance and in order to achieve a continuous current and they require additional external reactors when supplied from PWM modulated inverters.

Table 2-1 Electric machines in Energy Storage applications

Type		Speed	Flux Weakening	Characteristics
Induction	SCIM	High	Yes	- Robust and no idle losses. - Lower efficiency than other topologies. - Low cost & Low maintenance - need for excitation
	DFIM	Low	No	Work in limited range around the synchronous speed
Permanent Magnet		High	Possible	- High efficiency and power density - Sensitive to temperature - High material price

2.4.5 Conclusion

Both induction (IM) and permanent-magnet (PM) machines are appropriate for flywheel storage and power smoothing applications. However, for large-capacity long-term FESS, the operational power loss is an important factor that should be considered. There are three types of losses occur in three phase induction motor, including iron or core losses, mechanical losses and brush friction losses. The permanent magnet machine will generate constant flux with continuous iron loss, and slip ring DFIM will produce brush friction losses for FESS. Therefore, this will result in the speed drop of the FESS gradually, especially for long term FESS.

In general, the FESS working in high speed range is preferred for high power content. Under the field oriented control, the machine is capable to work above its rated speed, thus flux-weakening is preferred for high speed FESS. Squirrel cage induction machine (SCIM) is a good candidate for long-term large capacity FESS due to its advantages. Moreover, the iron loss of the SCIM is zero which is a very essential advantage for long term storage in power system at standby mode. For this reason, the squirrel cage induction machine is selected for this study, and the IM is connected as SCIM for both simulation and experiment to validate the proposed scheme.

2.5 Sensorless Speed estimation topologies

In order to produce robust structures to parameter variations, many sensorless topologies are proposed for rotor speed estimation with different induction machine models and control methods over years. Several sensorless speed estimation techniques are mostly commonly used and investigated: including the back-EMF method, Model Reference Adaptive System (MRAS),

Kalman Filtering techniques (EKF and UKF), high frequency injection technique, etc. [10]-[18], have gained much attention. By applying these techniques, precise flux and speed information can be obtained. However, the algorithmic complexity and calculation intensity look higher when compared with simple current model (CM) and voltage model (VM) observers in real-time applications. They also require a strong mathematical computation power to deal with. They are proved to be good alternatives for high performance ac drives.

Sensorless techniques can be divided into two classes, those using the fundamental properties or model of the machine and those exploiting subsidiary features often called anisotropies based model.

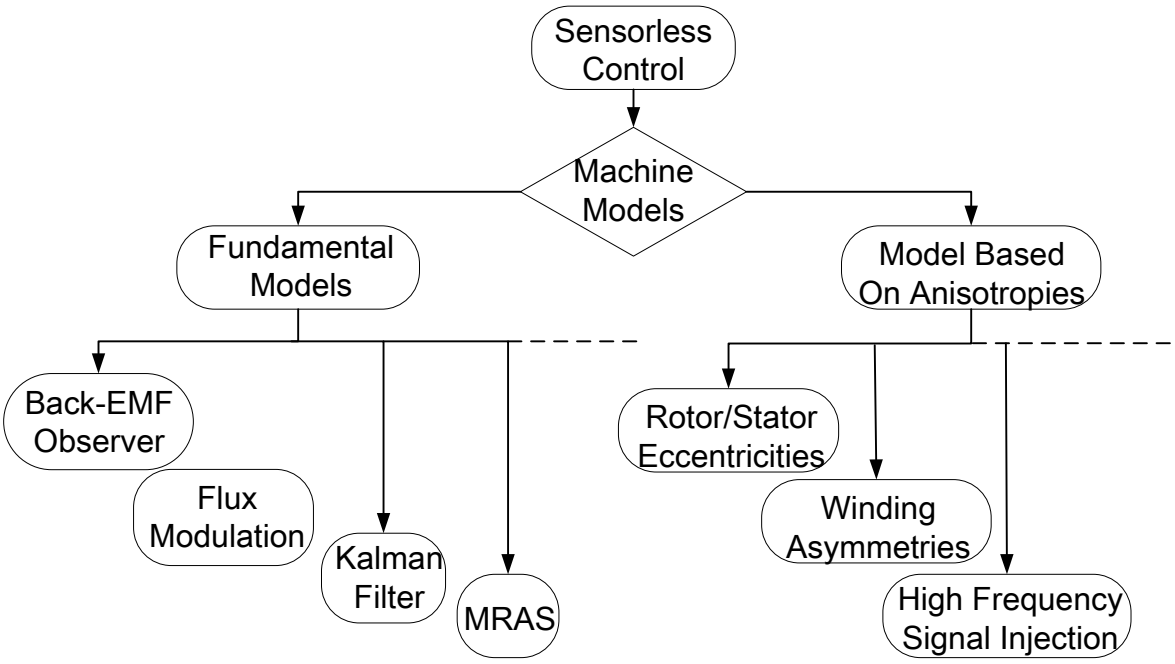


Fig. 2-7 Categories of sensorless control techniques

2.5.1 Fundamental Models

2.5.1.1 The fundamental back-EMF method

The simplest fundamental model methods are based on rotor flux position estimation by integrating the back-EMF. Such an approach is very simple and effective, but fails at low and zero speed. Low-speed operation is quite critical, since the motor back EMF is too low and rotor-flux-estimation results are very sensitive to stator resistance variations, or simply measurement noises. A large amount of research effort has been spent in order to develop reliable, low-cost sensorless control strategies, generally estimate the rotor position by processing electrical motor variables, such as phase currents or stator voltages. For all the back-EMF approaches, the method used for the rotor position and speed estimation is the main difference to define various approaches. Other back-EMF based Sensorless schemes adopt sophisticated identification procedures (such as flux-linkage observers, Kalman filter, fuzzy logic, the model reference approach (MRAS) etc.) to allow the low speed operation under some limitations, but can be too complex and expensive to be used in practical systems.

In addition, the back-EMF based rotor position estimation method can be highly sensitive to stator inductances and resistances. However, since the back-EMF is proportional to the rotor speed, it is very small at low speed and does not exist at zero speed. Therefore, the back-EMF-based sensorless methods may not start and fail at low speeds. In general, low speed operation is critical for such fundamental based methods, since the estimated rotor flux can be very sensitive to stator resistance variations, and measurement noises.

2.5.1.2 Model Reference Approach System (MRAS)

To avoid the drawbacks of the fundamental back-EMF method, the rotor-flux MRAS is introduced. These topologies adopt feedback correction along with the machine model to improve the estimation accuracy. Model Reference Adaptive Systems (MRAS) exploit the redundancy of two different models computing the same motor variables to estimate the speed. A comparison is made between the outputs of two estimators. For sensorless speed control algorithms, the quantity differing with the reference model from the adjustable model is the rotor speed. Since the voltage model estimator does not contain the rotor speed quantity, therefore, it can be considered as the reference model of the induction machine. The current model, however, containing the rotor speed as an input quantity, is considered as an adjustable model. The error between these two estimators is used as an input to an adaptation mechanism. The estimated rotor speed in the adjustable model is changed in such a way that the difference between these two estimators converges to zero gradually, and the estimated rotor speed will finally be equal to actual rotor speed. Other MRAS algorithms also suggests the electromotive force, rotor flux rather than the rotor speed as reference quantity for speed estimation. [28]

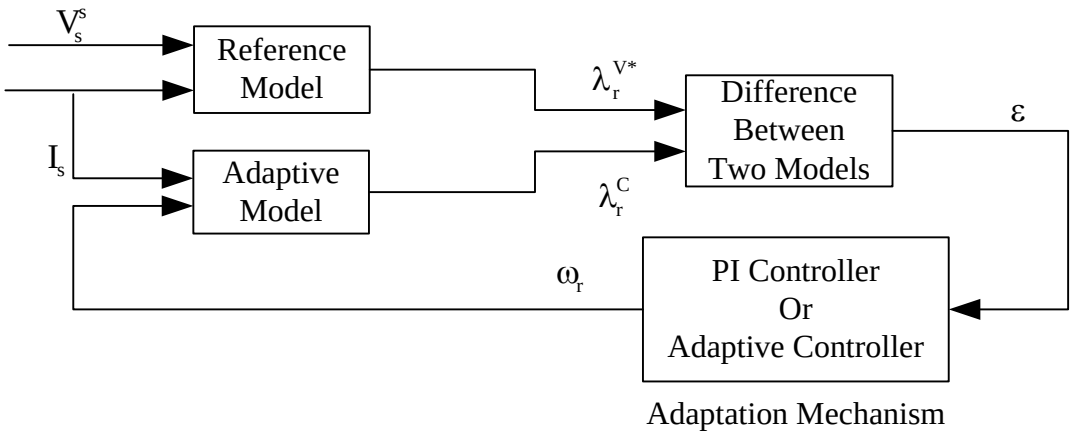


Fig. 2-8 MRAS-based speed estimator scheme

2.5.1.3 The Adaptive Approach

The performance of rotor flux MRAS speed estimator depends on the design of adaptation mechanism control techniques. The conventional constant gain PI-controller (PIC) has a simple structure and offers satisfactory performance over a wide range of ideal operation, but, may not give satisfactory performance under sudden system change. Moreover, it requires precise mathematical model, continuous tuning and accurate gain values of proportional (Kp) and integral (Ki) to achieve high performance drive. It is quite difficult to gain high performance of a sensorless drive using fixed gain values of PI controller, because of continuous variation in the machine parameters and nonlinearities presents in the voltage source inverter [17]. Therefore, to overcome with the above drawbacks, the adaptive control based adaptation mechanisms are desirable to tolerate machine parameter uncertainties and achieve high performance sensorless drive, such as online based gains tuning, sliding mode, fuzzy logic, neural network, particle swarm optimization (PSO) and etc. However, these algorithms may suffer from the computational intensity.

2.5.1.4 Kalman filter (KF)

Kalman Filter is another method employed to identify the speed and rotor-flux of an induction machine based on the measured quantities such as stator current and voltage. Kalman filter approach is based on the system model and a mathematical model describing the induction motor dynamics. KF itself works for linear systems, so for non-linear induction motor model, extended Kalman filter (EKF) is used. The EKF-based applications take a stochastic approach for the solution of the problem at zero and very low speed. With this method, it is possible to make the online estimation of states while simultaneously performing

identification of parameters in a relatively short time interval, also taking system/process errors and measurement noises directly into account. The EKF improves transient performance significantly for its high convergence rate. These advantages of the EKF make it a wide application in sensorless estimation in spite of its computational complexity. However, KF approach is computationally intensive and depends on the accuracy of the model of the motor.

2.5.2 Model Based on anisotropies

The distribution of stator winding and variation of air gap reluctance due to stator and rotor slots are main causes of air gap flux harmonics. The induction motor performance is affected by the harmonics in the time variation of the impressed voltage. The air gap flux is not purely sinusoidal as it contains odd harmonics (5th, 7th, 11th etc.). The harmonics caused due to variation of air gap reluctance are called tooth or slot harmonics [29]. Estimation techniques can be based on stator phase voltage third harmonics or spatial phenomena inherent within the machine, such as slot harmonics, rotor/stator eccentricities, and winding asymmetries. However, all these approaches can fail at zero speed for lack of useful information. Signal injection based sensorless method is developed to estimate the rotor position at low speed or zero speed for machines which exhibit rotor magnetic saliency. Such methods have the capability to provide accurate position and speed estimation without requiring information about the motor parameters. This is due to the presence of saliency/anisotropy in the machine, which gives information about the rotor speed and position. The injected signal is usually of high frequency, and the rotor position is estimated from the high frequency voltage equation. The saliency/anisotropy is the inductance difference between direct-inductance Ld and quadrature-inductance Lq, and it is due to either the asymmetric structure of the machine or the

flux induced magnetic saturation due to the fundamental excitation. The interaction between the injected high frequency signal and the spatial variations in the machine inductances produces an amplitude modulated signal containing useful information attached to the carrier frequency. The signal injection method suffers from computational complexity and requirement of external hardware for signal injection.

Conventionally, to extract the useful information from the spectrum of the resultant signal, Phase-Locked Loop structure based demodulation schemes have been used. PLL is a feedback loop where a voltage controlled oscillator (VCO) can be automatically synchronized to a periodic input signal. The basic PLL has three components connected in a feedback loop as shown in the diagram of Fig. 2-9 with a VCO, a Phase Detector (PD) and a Low Pass Filter(LPF).

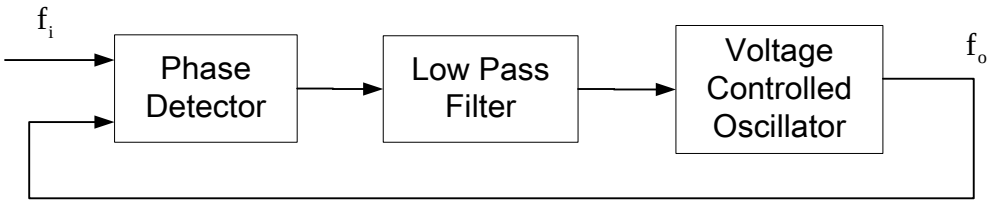


Fig. 2-9 Block diagram of the PLL

However, the PLL dynamics can be affected by the variation of inductances, PI parameters, and filter characteristics. In addition the error between the actual and estimated angle has to be small at the beginning of the estimation process. These restrictions are essential to maintain stability and to track the rotor speed and position accurately. However, such conditions are not always satisfied and are dependent on the operating conditions of the machine.

2.6 Proposed speed estimation

For a Flywheel Energy Storage System, since the stored energy increases only linearly with its moment of inertia but increases as the square of its rotational speed, it is preferred to increase the rotational speed of the flywheel instead of its rotational mass. In order to monitor the amount of stored energy of the FESS, its angular speed needs to be measured regularly.

For induction machines, an AC power source provides a rotating flux in the stator of the induction machine as the magnetization coil. Induced voltage and current in the rotor will produce a rotor flux field, to interact with the rotating stator flux field. This interaction will generate torque which drives the rotor to rotate. The rated speed is determined by the supplied stator flux frequency for a fixed motor, but will never reach the synchronous speed. This proposed speed estimation scheme for FESS is using the same concept as an induction machine in a reverse way. By creating a magnetic field in the stator side, the rotor is placed within the magnetic field. Since the rotor is spinning by its large inertia, the rotor rotates more rapidly than the magnetic field. The rotating shaft begins dragging the magnetic field forward, sending electricity following into the machine's coils.

At the stand-by mode, the FESS is not absorbing energy or supplying energy from/to its supporting power system. In order to further reduce its power loss, the inverter is turned off by disabling all its gating signals at standby mode. When the flywheel speed information is required, an excitation current is supplied from the FOC to the stator side of the induction machine through the inverter. A magnetic field is created by this excitation current. By cutting this magnetic field, the revolving rotor will generate rotating flux field at the same frequency as the rotor speed. The rotating magnetic flux from the rotor induces three phase current reversely in the stator at the

same frequency as the rotor speed. It should be noted that there is no frequency difference between the rotor speed and the induced stator current. Since the stator is constructed stationary, the generated torque will act on the rotor to lower the rotor speed during the excitation period. Various techniques are introduced and applied to the proposed control scheme in this study in an effort to minimize the impact to the FESS due to this generated torque from speed estimation. By measuring the stator voltage and current, the rotor speed information can be extracted. In order to improve the accuracy of the estimation at low speed, a hybrid flux observer that takes advantage of both the current model and voltage model is derived and presented.

Chapter 3:Sensorless Speed Estimation for IM Driven Flywheels

3.1 Squirrel Cage Induction Machine Modeling

IM are relatively rugged, reliable and inexpensive machines and are traditionally used in fixed-speed service. However, they are increasingly used with variable-frequency drives (VFDs) in variable-speed service. VFDs offer especially important energy savings opportunities for IMs. As the result, IMs are widely used in industries, and almost 95% is of squirrel cage type. Squirrel cage IM has many advantages: First of all, it is less expensive. Next, it has very compact structure and insensitive to environment. Furthermore, it does not require periodic maintenance for brush replacement like DC motors. However, because of its highly non-linear and coupled dynamic structure, a squirrel cage IM requires more complex control schemes than DC motors.

3.1.1 Single phase equivalent circuit for parameter testing

The steady-state performance of a poly-phase IM can be obtained using per phase equivalent circuit:

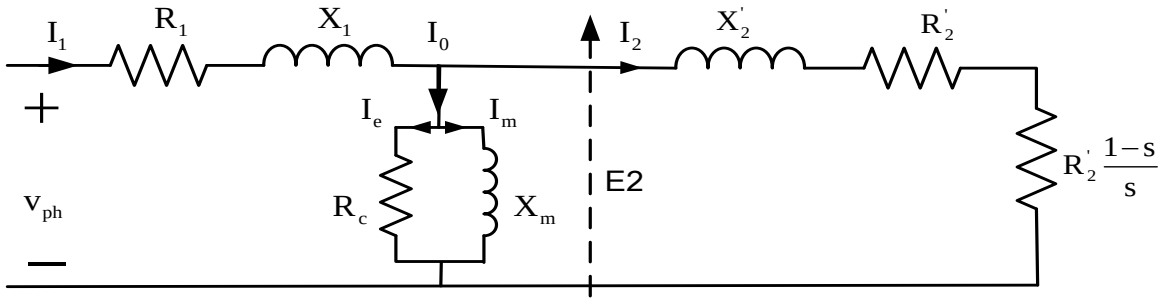


Fig. 3-1 per phase equivalent circuit of a poly phase IM

The parameters of the induction machine could be determined by 'no load' and 'blocked-rotor' tests, the former determines R_c and X_m while the latter yields R_1, R'_2, X_1, X'_2.

In order to validate the proposed sensorless speed estimation approach for the FESS at standby mode, simulation and experiment are conducted with a SCIM setup. Based on the no load and blocked rotor test, the parameters of the SCIM used in this project are extracted and listed in Table 3-1

Table 3-1 Tested Induction Machine(IM1) Parameters

Rated	Voltage(line to line)		220/440V	
	Power		1/2 hp	
	Frequency		60 Hz	
	Speed		1750 rpm	
	Number of poles		4	
Tested	Rs	0.6405Ω	Ls	0.0452H
	Rr	0.3625Ω	Lr	0.0452H
	Lm	0.0416		

3.1.2 Three-phase transformations (Clarke, Park, IPark)

In a balanced IM system, the values on the natural three phase a-b-c system would always balance each other. This is one of the core values for the direct quadrature (d-q) transformation; The application of d-q transformation reduces the three phase a-b-c AC quantities to 2 quantities in d-q axis. The d-q transformation is a mathematical transformation used to simplify the analysis of three phase circuit. The transformed arbitrary d-q reference frame is dynamic equivalent circuit as a-b-c system with reduced number of relevant variables in the system. The transformation between the a-b-c and d-q system is illustrated in Fig. 3-2

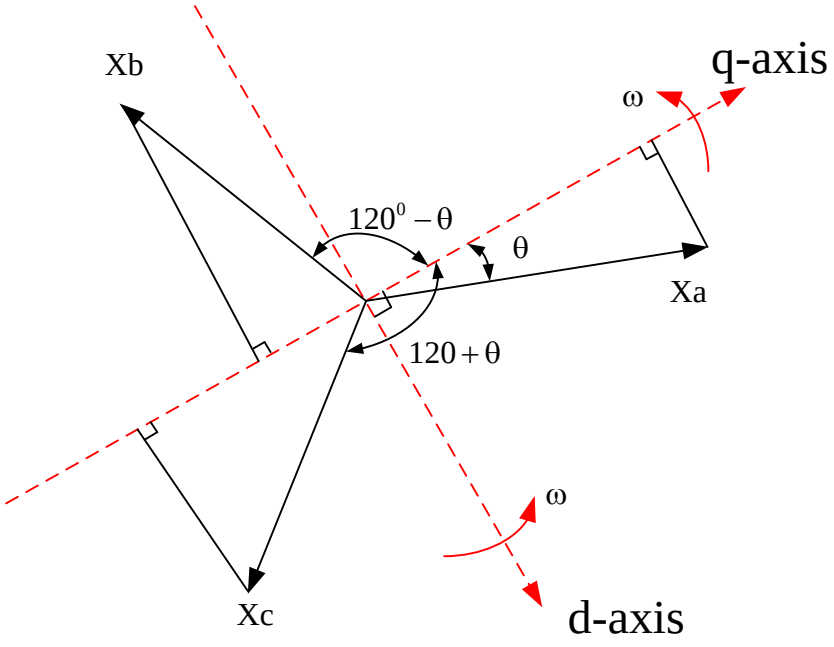


Fig. 3-2 abc-d-q transformation

In a balanced three-phase system, the a-b-c components can be projected onto the d and q axes or d-q axes. In making these projections, the expressions for the components in phase with the d and q axes can be obtained respectively. Although the speed of these axes can be any speed that is convenient for analysis, generally it will specify to be synchronous speed, ω_s, rotor speed ω_r and arbitrary speed ω.

For AC systems, the stationary reference frame will produce a rotating vector and difficult to analyze for the controller design. It is preferred to synchronize the reference frame with the rotating speed of the system. The rotating reference frame has the unique property of eliminating all time varying inductances from equations of three-phase AC machines due to the rotor spinning. As the result, the transformation of machine equations in synchronous reference frame forces all sinusoidal varying inductances in the a-b-c frame to become constant in the d-q reference frame. Simplified calculations can then be carried out on these imaginary quantities

before performing the inverse transformation to recover the actual three phase AC results. This technique is widely applied for the analysis of AC systems.

In field-oriented control, the control inputs can be specified in two phase arbitrary rotating d-q frame as i_{ds} and i_{qs} such that i_{ds} being aligned with the d-axis or the flux vector. These two-phase synchronous control inputs are converted into two-phase stationary quantities and then to three-phase stationary control inputs.

The relationship of the abc-d-q transformation with different reference frame is shown in Fig. 3-3. The Alpha-Beta-Zero to dq0 block performs a transformation of αβ0 Clarke components in a fixed reference frame to dq0 Park components in a rotating reference frame. The dq0 to Alpha-Beta-Zero block performs a transformation of dq0 Park components in a rotating reference frame to αβ0 Clarke components in a fixed reference frame.

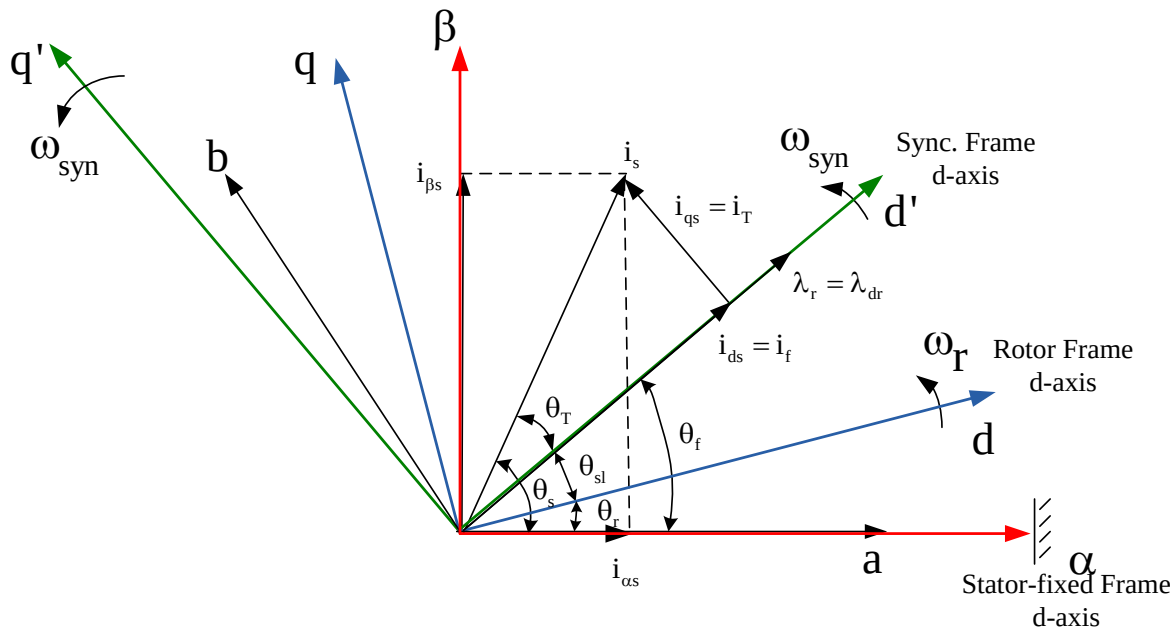


Fig. 3-3 Rotor flux field oriented control

Where: φ_r - rotor flux aligned with synchronous frame

θ_f - Rotor flux angle

i_s - Stator current (stator fixed αβ frame)

θ_s - Stator current angle

θ_r - Rotor angle (rotor position)

θ_{sl} - Slip angle

$\theta_f = \theta_r + \theta_s$

i_{ds}, i_q - D-q axis stator currents

In this project, the applied transformation includes Clarke, Park and Inverse Park transformation.

Clarke's transformation is to transfer the three-phase stationary parameters from a-b-c system to the two-phase stationary α-β reference frame. The α-axis and β-axis are orthogonal.

$$\left[f_{\alpha\beta}\right] = T_{\alpha\beta}\left[f_{abc}\right] \tag{3-1}$$

Where the transformation matrix $T_{\alpha\beta 0}$:

$$T_{\alpha\beta} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \tag{3-2}$$

Park's transformation is to transfer the three-phase stationary parameters from a-b-c system to the two-phase arbitrary rotating d-q reference frame. It is a combination of the Clarke transformation with a rotation with an angular velocity ω, where $\theta = \int \omega dt$

$$[f_{qd}] = T_{qd}(\theta)[f_{abc}] \tag{3-3}$$

Where $T_{qd0}(\theta)$ is the transformation matrix, and θ is the angular displacement of Park's reference frame.

$$T_{qd}(\theta) = \frac{2}{3}\begin{bmatrix} \cos\theta & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta - \frac{4\pi}{3}) \\ \sin\theta & \sin(\theta - \frac{2\pi}{3}) & \sin(\theta - \frac{4\pi}{3}) \end{bmatrix} \tag{3-4}$$

Based on the Clarke's transform, the Park's transform can be simplified.

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} \tag{3-5}$$

Inverse Park's transformation: the Inverse Park's transformation is to transfer the two-phase arbitrary parameters from d-q system to the three-phase a-b-c reference frame. The frame of reference may rotate at any constant, varying angular velocity, or it may remain stationary.

$$\left[f_{abc}\right] = T_{dq}(\theta)^{-1}\left[f_{dq}\right] \tag{3-6}$$

Where the inverse of Park's transformation matrix is given by

$$T_{dq}(\theta)^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \cos(\theta - \frac{2\pi}{3}) & -\sin(\theta - \frac{2\pi}{3}) \\ \cos(\theta + \frac{2\pi}{3}) & -\sin(\theta + \frac{2\pi}{3}) \end{bmatrix} \tag{3-7}$$

3.1.3 Machine Model in arbitrary d-q reference frame

Although the behavior of a symmetrical IM may be described in any reference frame, there are four that are commonly employed: The arbitrary reference frame (denoted as ω), the

stationary reference frame ($\omega = 0$), the rotor reference frame ($\omega = \omega_r$) and the synchronous reference frame ($\omega = \omega_e$). In synchronous reference frame, the rotating speed of the synchronous reference frame is given by $\omega_e = 2\pi f_s$ where f_s is the power supplying frequency. The voltage equations for each of these reference frames may be obtained from the voltage equations in the arbitrary reference frame by assigning the appropriate speed to ω.

For AC machines, the stationary reference frame denotes variables and transformations associated with circuits that are stationary in 'real life' as opposed to rotor circuits that are free to rotate, while the synchronous reference frame is the electrical angular velocity of the air gap rotating magnetic field established by stator currents of fundamental frequency. Generally, the conditions of operation will determine the most convenient reference frame for analysis and/or simulation purposes.

The d-q-axis model of the IM can be obtained by decomposing the space vectors into their corresponding d- and q- axis components, that is,

$$\begin{cases} \overrightarrow{v_s} = v_{ds} + jv_{qs}; \overrightarrow{i_s} = i_{ds} + ji_{qs}; \overrightarrow{\lambda_s} = \lambda_{ds} + j\lambda_{qs} \\ \overrightarrow{v_r} = v_{dr} + jv_{qr}; \overrightarrow{i_r} = i_{dr} + ji_{qr}; \overrightarrow{\lambda_r} = \lambda_{dr} + j\lambda_{qr} \end{cases} \tag{3-8}$$

The d-q transformed three-phase IM in arbitrary reference frame is shown in Fig. 3-4

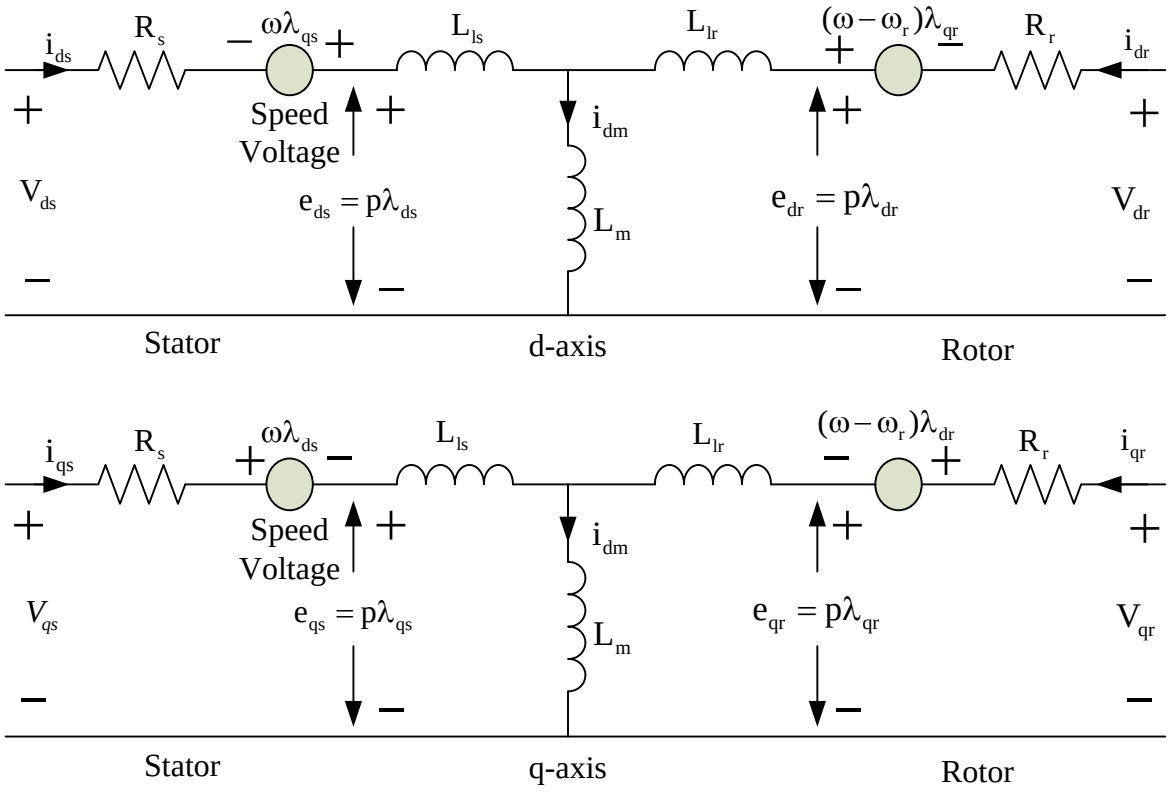


Fig. 3-4 Mathematical Model of IMs in Arbitrary Reference Frame

This mathematical model of IM is a fundamental task for this proposed speed estimation scheme. Stationary reference frame is used for the SCIM model derivation. The two phase d-q model at the synchronous speed will help to simplify and carry out this decoupled control concept to the IM.

The following assumptions are made in the derivation to simplify the process:

• Losses caused by eddy currents, hysteresis and saturation are neglected.

• The back EMF is sinusoidal;

• Parameter Variations due to temperature change are not considered.

• Machine inductances and resistance remain constant.

Based on Fig. 3-4, the stator properties of the motor in continuous time are completely described by the following equations in arbitrary d-q reference frame:

Stator voltage equations:

$$v_{dqs} = R_s i_{dqs} + j\omega\lambda_{dqs} + p\lambda_{dqs} \tag{3-9}$$

For squirrel cage IMs, the rotor is constructed with cylindrical laminated core with parallel slots for carrying the rotor conductors. These conductors are made from thick, heavy bars of copper or aluminum or its alloys. There are end rings which are welded or electrically braced or even bolted at both ends of the rotor, thus maintaining electrical continuity. These end rings are short-circuited, which gives a look similar to a squirrel cage thus the name. The end rings and the rotor conducting bars are permanently short-circuited. Therefore, $V_{dr} = 0$, $V_{qr} = 0$.

The derived rotor side voltage equation of SCIM in arbitrary d-q reference frame is shown in Eqn. (3-10)

Rotor voltage equation:

$$0 = R_r i_{dqr} + j(\omega - \omega_r)\lambda_{dqr} + p\lambda_{dqr} \tag{3-10}$$

From Fig. 3-4, the stator flux linkage, rotor flux linkage and rotor side back EMF equations can be derived.

Stator flux linkage equation:

$$\lambda_{dqs} = L_s i_{dqs} + L_m i_{dqr} \tag{3-11}$$

Rotor flux linkage equation:

$$\lambda_{dqr} = L_r i_{dqr} + L_m i_{dqs} \tag{3-12}$$

Stator flux vector equation:

$$\lambda_{dqs} = \int (v_{dqs} - i_{dqs} R_s) dt \tag{3-13}$$

Rotor flux vector equation:

$$\lambda_{dqr} = \frac{L_r}{L_m}\left(\lambda_{dqs} - \sigma L_s i_{dqs}\right) \tag{3-14}$$

Electromagnetic torque:

$$T_e = \frac{3PL_m}{2L_r}\left(\lambda_{dr} i_{qs} - \lambda_{qr} i_{ds}\right) \tag{3-15}$$

Rotor speed equation:

$$\omega_r = \frac{T_e - T_L}{JP + B} \tag{3-16}$$

Where d: direct axis;

q: quadrature axis;

s: stator side;

r: rotor side

Reorganize the above Eqn.(3-9) ~ Eqn.(3-14), the squirrel cage IM model is derived as a function of $\lambda_{qr}, \lambda_{dr}, i_{qs}, i_{ds}$.

Substitute i_{dqr} from Eqn. (3-11) in the stationary reference frame ($\omega = 0$) into Eqn.(3-12)

$$\lambda_{dqr} = \left(\frac{L_m R_r}{L_r} i_{dqs} + j\omega_r \lambda_{dqr}\right) \frac{1}{s + \frac{R_r}{L_r}} \tag{3-17}$$

Substitute i_{dqr} from Eqn. (3-11) in the stationary reference frame into Eqn.(3-9),

$$i_{dqs} = \left(v_{dqs} - j\frac{L_m}{L_r}\omega_r \lambda_{dqr} + \frac{L_m R_r}{L_r^2}\lambda_{dqr}\right) \frac{1}{\left(L_s - \frac{L_m^2}{L_r}\right)s + \left(R_s + \frac{L_m^2 R_r}{L_r^2}\right)} \tag{3-18}$$

Therefore, the derived IM model is list in Fig. 3-5 Squirrel cage IM and load model.

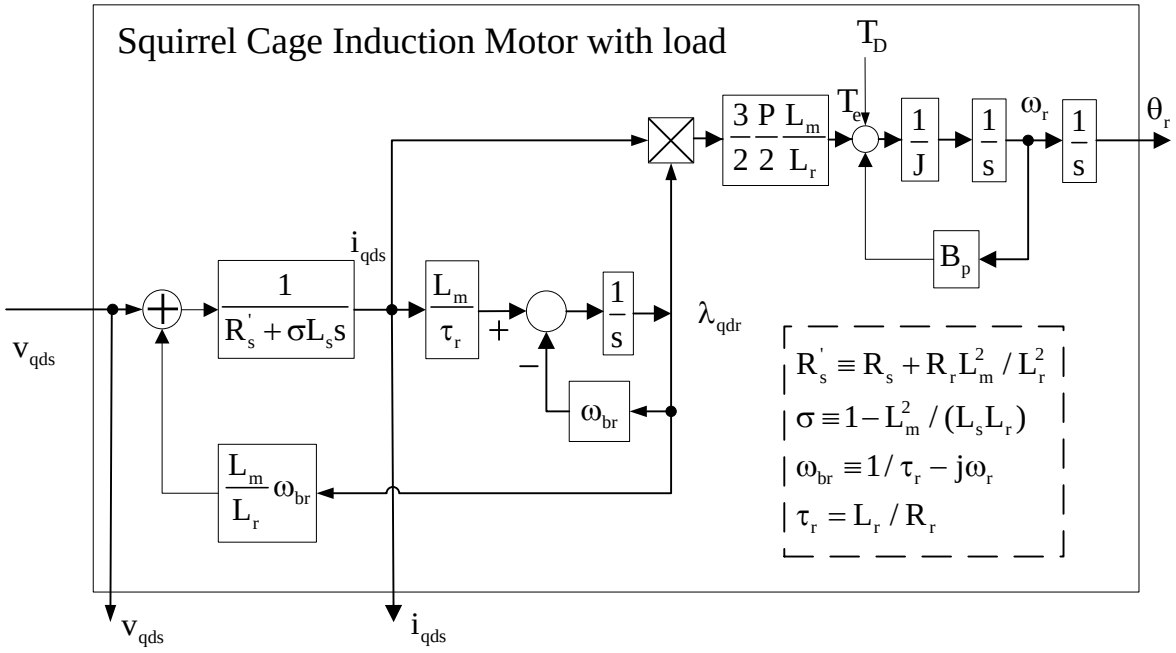


Fig. 3-5 Squirrel cage IM and load model

3.2 Derivation of the Control Scheme: modified Field-Oriented Control (vector control)

Traditional open-loop scalar control (V/Hz) of the IM with variable frequency may provide a satisfactory solution under limited conditions. However, when high performance dynamic operation is required, these methods may not be sufficient. Therefore, more sophisticated control methods are required to make the performance of the IM comparable with DC motors. The advancement in drive control techniques, fast semiconductor power switches, powerful and cheap microcontrollers made IMs alternatives to DC motors in industries. The most popular IM drive method has been the field oriented control (FOC) in the past two decades for its decoupled control technique, and there is an increasing trend in FOC towards the use of sensorless techniques that avoid the use of speed sensor and flux sensor. The hardware sensors of the drive are replaced with state observers to minimize the cost and increase the system reliability.

Field-oriented control (FOC) is a variable frequency drive (VFD) control method which controls three-phase AC electric motor output. The FOC is performed in a two-phase d-q reference frame. All the variables from three-phase a-b-c system need to be transformed into the two-phase stationary reference frame and then retransform these variables from the stationary reference frame to a rotary reference frame with arbitrary angular velocity of ω. It employs torque control concepts in the IM which are patterned from DC machine. In a DC machine, the commutator is holding a fixed, orthogonal spatial angle with the armature magneto motive force(MMF). This action is emulated in FOC by orienting the stator current with respect to the rotor flux in order to attain independently controlled flux and torque in IM. It implies that the instantaneous torque is controlled by stator current vector that is orthogonal to the rotor flux vector. Such controllers are called field-oriented controllers, also referred to as vector controllers. "Field orientation" and "vector control" are used virtually interchangeably.

3.2.1 Decoupling Control Technique of FOC

In FOC, the stator current components supplied to the machine are oriented in phase and in quadrature to the rotor flux vector λ_{qdr}. This is accomplished by locking the phase of the reference system such that the rotor flux is entirely in the d-axis (flux axis), resulting in the mathematical constraint $\lambda_{qr} = 0$. [25]

Therefore,

$$\lambda_r = \sqrt{\lambda_{dr}^2 + \lambda_{qr}^2} = \lambda_{dr} \tag{3-19}$$

Substitute the mathematical constrain $\lambda_{qr} = 0$ into Eqn. (3-12) in d-q axis result in

$$\lambda_{qr} = L_m i_{qs} + L_r i_{qr} = 0 \tag{3-20}$$

And $$\lambda_{dr} = L_m i_{ds} + L_r i_{dr} \tag{3-21}$$

With the constraint substituted into Eqn. (3-15)

$$T_m = \frac{3}{2}\frac{P}{2}\frac{L_m}{L_r}\lambda_{dr} i_{qs} \tag{3-22}$$

The torque equation shows a proportional relation between the desired torque and the torque-producing current component i_{qs}. Therefore, a linear torque control is achieved and only proportional to torque-producing current i_{qs} when rotor flux is fixed.

Substitute the constrain $\lambda_{qr} = 0$ into Eqn. (3-10) in d-q axis

$$0 = R_r i_{qr} - (\omega - \omega_r)\lambda_{dr} \tag{3-23}$$

$$0 = R_r i_{dr} - p\lambda_{dr} \tag{3-24}$$

In steady-state, $\lambda_{dr} = \lambda_r = \text{const}$. Therefore, $p\lambda_{dr}$=0. Consequently from Eqn.(3-24), the rotor current component $i_{dr} = 0$. However, during flux changes, $i_{dr} = 0$ is not zero from Eqn. (3-21)

$$i_{dr} = \frac{\lambda_{dr} - L_m i_{ds}}{L_r} \tag{3-25}$$

The rotor current i_{qr} and the torque-producing current i_{qs} follow immediately from Eqn.(3-20)

$$i_{qr} = -\frac{L_m}{L_r} i_{qs} \tag{3-26}$$

Combining Eqn.(3-24) and Eqn.(3-26) to eliminate i_{dr} yields the equation relating i_{ds} and $\lambda_{dr} = \lambda_r$.

$$(R_r + L_r p)\lambda_{dr} = R_r L_m i_{ds} \tag{3-27}$$

The relation between the flux-producing current i_{ds} and the rotor flux λ_{dr}, is a first-order linear transfer function with a time constant τ_r, where $\tau_r = L_r / R_r$

From the derivations above, Eqn.(3-22) shows that the electrical torque is only proportional to i_{qs}* when rotor flux is fixed without an interaction with i_{ds}*. Eqn.(3-27) shows a linear relationship between the rotor flux and the i_{ds}* without an interaction with i_{qs}*. Therefore, a decoupled control for torque and rotor flux from the stator current in d-q axis is achieved.

3.2.2 Direct Field Orientation (DFO)

The key issues associated with the rotor flux oriented control are to accurately determine the rotor flux angle θ_e for field orientation. Since it is not possible to directly sense rotor flux, a rotor flux-oriented system must employ some computation to obtain the desired information from a directly sensed signal. The angle θ_e can be found either by Indirect Field Orientation control(INFOC) or by Direct Field Orientation control(DFOC). DFOC is most often used for sensorless control and estimates the rotor flux from the terminal quantities (stator voltages and currents). This method does not require a speed sensor for the measurement of rotor speed. Therefore, it is employed for sensorless speed estimation in this project.

A variety of flux observers can be employed to obtain improved response and less sensitivity to machine parameters. A major problem with most direct orientation schemes is their inherent problems at very low speeds where the machine IR drops are dominant and/or the required integration of signals becomes problematic [23]. For this project, a hybrid rotor flux observer is used to calculate the instantaneous rotor flux at every sampling period in an effort to avoid these problems.

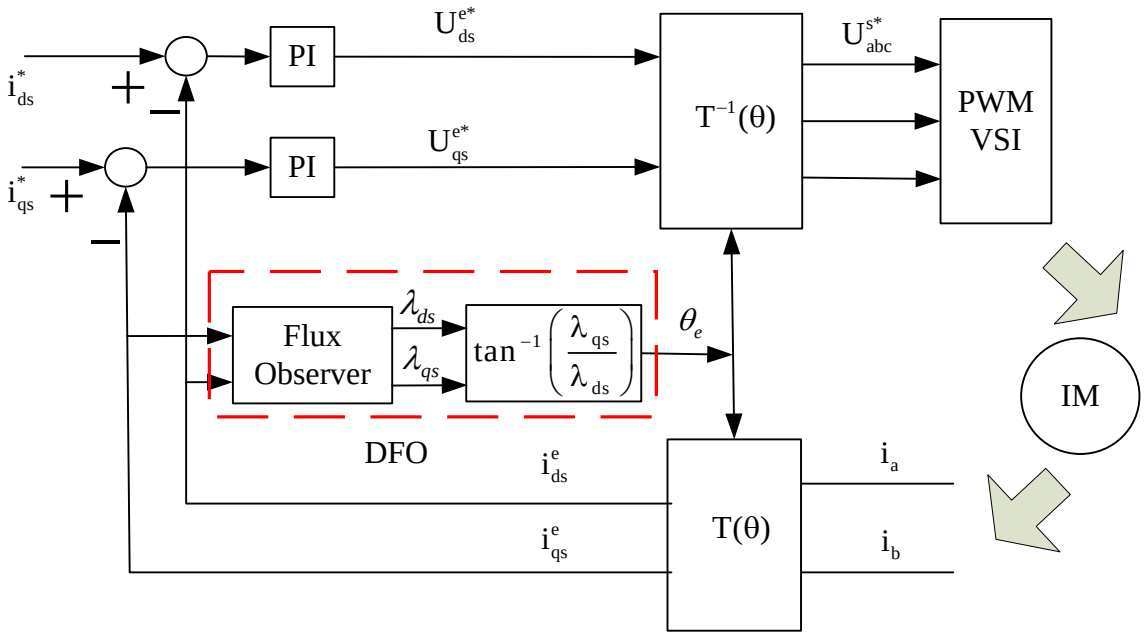


Fig. 3-6 Schematic of Field-Oriented Control with DFO

3.2.3 Flux Observers Introduction

There are two basic approaches for speed and position estimation in induction motors. The first approach uses the fundamental machine model to design model reference adaptive systems, nonlinear observers, extended Kalman filters, or adaptive observers. It has long been recognized that the challenging part in this approach is keeping a load stationary at (or near) zero flux frequency. The second approach uses secondary phenomena or the parasitic effects of the machine to develop methods that will be effective at low frequency.

Examination of the literature on the first approach shows the following drawbacks:

• Analysis is limited to local linear models; it is rare to find analysis that takes into consideration the nonlinearities of the system;

• Model uncertainty is usually ignored in the analysis, even though the presence of such uncertainty (e.g., changes of resistances with temperature) could change the conclusions in a fundamental way;

• No analysis of the overall closed-loop control system. For example, it is typical in methods based on rotor or flux position sensing that the analysis be limited to the position estimation problem itself with no analysis of the impact of the estimation error on the performance of the closed-loop system. None of these papers provide closed-loop analysis in the presence of parameter uncertainty that is not compensated for by adaptation. More importantly, all the proposed controllers are more involved than the traditional field oriented proportional–integral (PI) controllers and the closed-loop analysis provided in these papers does not provide a theoretical understanding of such simple controllers.

The main idea of this study is to use intrinsic motor electro-mechanical properties to estimate the rotor speed or position. The control of the flywheel induction motor requires precise speed information. Therefore, a rotational transducer, such as a resolver or an encoder, is usually mounted to the shaft of the motor to measure the motor speed. However, a rotational transducer cannot be mounted in some cases, such as motor drives in a hostile environment, high-speed motor drives, etc. A speed sensor increases the cost and requires a connection line between the control system and the motor, thereby preventing the stable operation of the control system due to interference from the signal line.

For today's AC drive systems, the ability to obtain accurate flux data is crucial to implement since the control performance is seriously influenced by the estimated flux accuracy. A pure integration of the stator-induced voltage is the most convenient approach to obtaining stator flux

in the AC drive. Although pure integration is simple in concept, it has the following significant challenges in providing adequate real-time performance. 1) A small drift or dc offset inherently present in current measurement channels can cause an integrator to saturate. 2) Variations in the stator resistance result in a magnitude and phase error of the estimated flux at low speeds. 3) An initial condition error produces a constant output dc offset in the estimated flux. As a result, only a few published works deal with pure integration for flux estimation. Although excellent experimental results have been reported at low speeds, including a zero stator frequency, the complexity and computational burden have hindered their widespread applications. The inadequacy of pure-integrating treatments has motivated a diverse literature on the technique of using low-pass filters with a fixed or variable cutoff frequency. The replacement can largely alleviate the drift and the initial value problem while the magnitude and phase angle error are significantly introduced when the stator frequency reaches down to a few hertz. Based on the derivation method, flux observers can be grouped into open-loop observers and closed-loop observers.

3.2.3.1 Open-loop flux observers

Open loop flux observers have two fundamental models: the current model (CM) and the voltage model (VM). These two observers in general, are the two most commonly used ways to estimate the flux using the terminal quantities, and it describes the IM with differential equations in the form of different state variables.

Voltage model based observers use the measured stator voltage and current as inputs. The voltage model flux observer is dominated by the voltage drop on stator resistance at low speed. It requires a pure integration that is difficult to implement for low excitation frequencies due to the

offset and initial condition problems. It is a simple way to implement as only terminal parameters are needed. The VM works well at high rotor speed. However, at low speed, the stator voltage becomes a relatively small magnitude, and is dominated by the voltage drop on stator resistance. As the VM adopts integral to calculate the state variables of the system, it is difficult both to decide on the initial value, and prevent the drift of the output of a pure integrator. Therefore, it can be easily distorted by integral error and measurement inaccuracy [30]. At very low stator frequency, the estimation accuracy is particularly sensitive to the inaccurate stator resistance value. This inaccuracy causes estimation errors both in the amplitude and the estimated angle of the stator flux vector, which will further add to the distortion to the speed estimation. The situation will be improved above a few hertz stator frequency. [31]

Current model based observers, on the other hand, use the measured stator currents and rotor velocity to estimate the rotor flux. The velocity dependency of the current model is an important factor to be considered for sensorless estimation since this means that although using the estimated flux eliminates the flux sensor, the rotor velocity and position must be obtained either from a speed sensor or a speed observer. For both sensorless flux and sensorless speed observers, these two observers may overlap and cannot be tuned independently due to the fact that the overlap of the flux observer and speed observer will accumulate errors and make the system unstable. In addition to being less suitable for speed-sensorless operation, the CM has also a fairly high parameter sensitivity which may results in inaccurate field orientation, which is not necessarily critical for stability, but degrades the dynamic properties. The flux transient affects the torque in particular, which results in the change in rotor speed. Due to the parameter sensitivity, the CM should be used at low speeds only.

3.2.3.2 Closed loop flux observers

In order to produce more robust structures to parameter variations, many closed-loop topologies based on the CM and VM are proposed. In closed loop observers, feedback correction is used along with the machine model to improve the estimation accuracy. Several sensorless control techniques are mostly commonly used and investigated. Among these, Model reference adaptive system (MRAS), Kalman Filtering techniques (EKF and UKF), fuzzy logic, sliding mode, etc. gained much attention. By applying these techniques, precise flux and speed information can be obtained, however, the algorithmic complexity and calculation intensity looks higher when compared with open-loop observers in real-time applications. They also require a strong mathematical computation power to deal with. They are proved to be good alternatives for high performance on-line AC drives.

For this study, in order to estimate the rotor speed at standby mode as quickly as possible, highly computational and complex flux observers which requires on-line computation are not applied for this sensorless speed estimation scheme, even though these observers may have more robust control and accurate result. For this purpose, the flux observer is derived from the stator terminal voltage and current. The rotor flux observer employs both the voltage and current models of the IM to construct a hybrid open-loop observer structure to seamlessly operate in current mode at low frequency and voltage mode in high frequency.

3.2.4 Derivation of the Hybrid Rotor Flux observers

3.2.4.1 Voltage Model based rotor flux observer

The voltage model rotor flux observer is based on the detection of stator voltage and stator current as inputs to compute the rotor flux.

From Eqn. (3-11) and Eqn.(3-12)

$$\lambda_{dqr} = \frac{L_r}{L_m}(\lambda_{dqs} - \sigma L_s i_{dqs}) \tag{3-28}$$

The issue for the VM regarding the integration process inherited from the IM dynamics is a complicate process to calculate the state variables of the system. The equation describing the relationship between the stator voltage, current and flux linkages in stationary reference frame is given in

$$v_{dqs} = i_{dqs}R_s + p\lambda_{dqs} \tag{3-29}$$

The stator flux can then be estimated by integrating the stator voltage on the leakage and mutual inductance

$$\lambda_{dqs} = \int (v_{dqs} - i_{dqs}R_s)dt \tag{3-30}$$

$$\lambda_{dqr}^{v} = \frac{L_r}{L_m}\left(\int (v_{dqs} - R_s i_{dqs})dt - \sigma L_s i_{dqs}\right) \tag{3-31}$$

Since the voltage mode flux observer adopts the integral, it causes DC bias and the initial values problems. Practically, to overcome this problem a high-pass first order filter replaces the integrator. In the actual implementation, a high pass filter is used instead of pure integration to avoid low frequency integration problems.

$$\lambda_{dqr}^{v} = \frac{L_r}{L_m}\left(\frac{T_c}{1+T_c}(v_{dqs} - R_s i_{dqs} - \sigma L_s p i_{dqs}\right) \tag{3-32}$$

3.2.4.2 Current Model based rotor flux observer

The current model rotor flux observer is based on the detection of stator current and rotor velocity. The current model flux observer works well at low speed, however, the accuracy of the estimated rotor flux can be easily affected by the rotor parameters, and these parameters will vary with rotor temperature.

From Eqn. (3-11)

$$i_{dqr} = \frac{1}{L_r}(\lambda_{dqr} - L_m i_{dqs}) \tag{3-33}$$

Substitute i_{dqr} from Eqn. (3-11) into Eqn. (3-12)

$$\lambda_{dqr} = \frac{1}{\tau_r p + 1}(L_m i_{dqs} - \tau_r \omega_r \lambda_{qdr}) \tag{3-34}$$

3.2.4.3 Hybrid rotor flux observers

When the voltage model is used to calculate the rotor flux, the integral operation will be inevitable. Such problem as initial value setting, error accumulation due to parameter variation of the motor will result in bad flux estimation and system instability. To overcome those problems, a flux observer is designed by modifying voltage model on the basis of current model to estimate rotor flux.

The current model performs best at zero and low speed, and the other performs best at high speeds. Both voltage model and current model have its own pros and cons. Therefore, it is better to use voltage model at high speed, and current model at low speed. A seamless transition from the CM to the less parameter sensitive “voltage model” is preferred at nominal speeds, and this leads to the hybrid rotor flux observer.

$$\lambda_{dqr} = \left(\frac{1+T_c p}{1+T_c p}\right)\lambda_{dqr} = \left(\frac{T_c}{1+T_c p}\right)e_{dqr} + \left(\frac{1}{1+T_c p}\right)\lambda_{dqr} \tag{3-35}$$

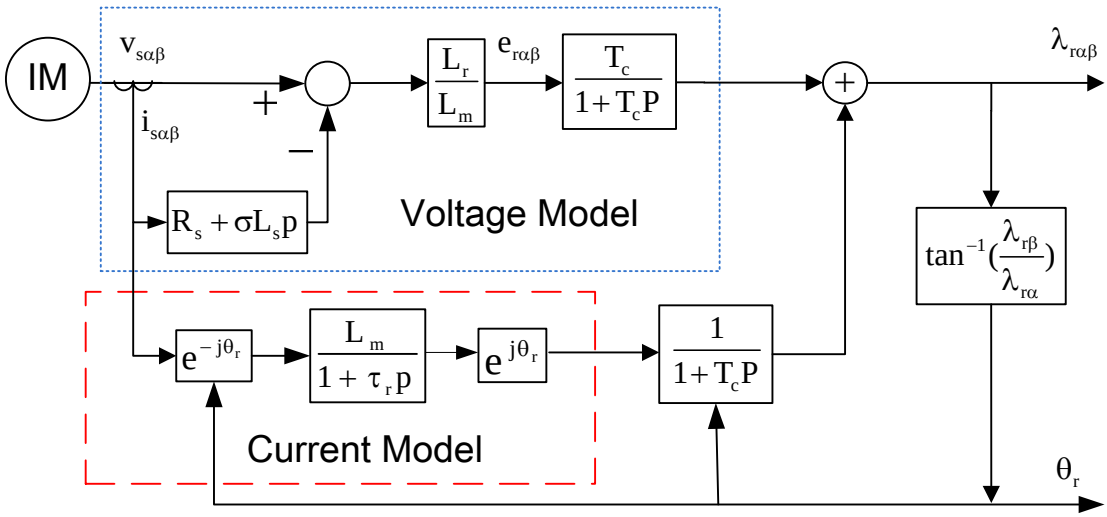


Fig. 3-7 Hybrid Flux Observer (CM+VM)

The first part of Eqn. (3-35) is the observed flux value from voltage model, and the second part is the error between the real value and the observed flux value which is compensated by current model. The selection of filter time constant Tc should make the error as small as possible which means a large Tc value is desired. However, a large Tc will lead to a slow dynamic response of the flux observer. Therefore, the magnitude of the Tc value is selected based on the full consideration of requirements to reach its best performance.

The formulation of a hybrid flux observer is shown in Fig. 3-7. It allows better performance to be achieved by seamlessly incorporating two models for high-performance flux regulation as well as flux orientation. The combined approach compromises the advantages of both models in performance and parameter insensitivity. The transition between models is governed by the bandwidth of the compensated observer which is selected by Tc. Typical bandwidths are in the range of 1 to 10Hz. For this simulation, Tc=0.2, which gives a cut-off frequency at 5Hz for current model rotor flux observer.

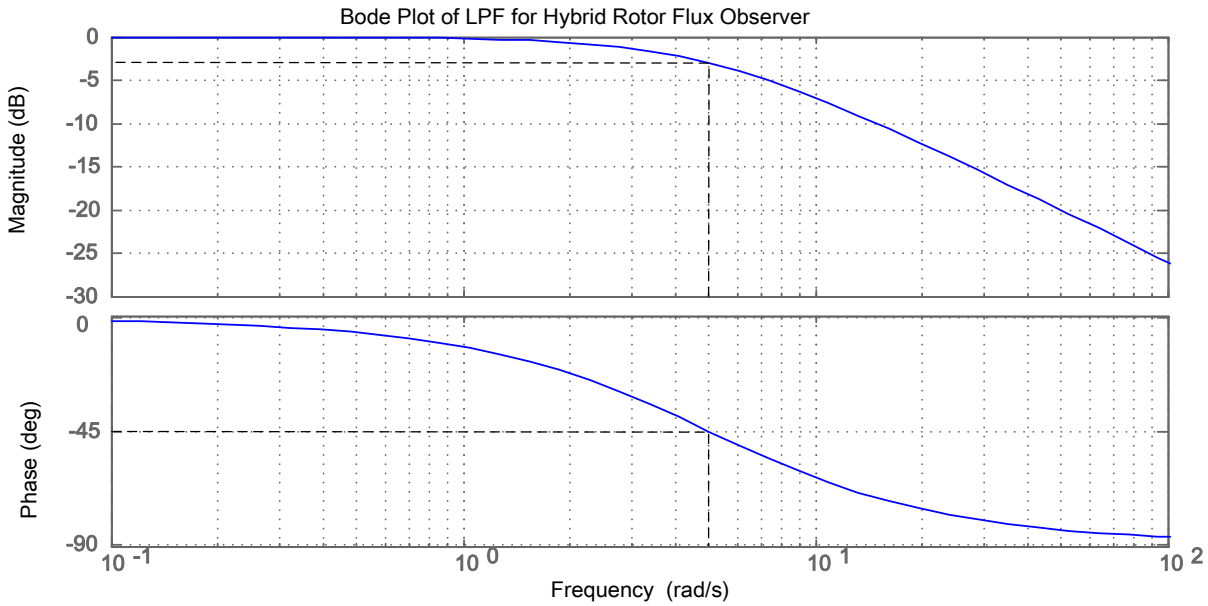


Fig. 3-8 Bode Plot of cut-off frequency of CM

Therefore, only with the measurement of the stator voltage and current, the rotor flux angle can be determined, and there is no speed sensor needed.

3.2.5 Derivation of Speed Observer

3.2.5.1 Introduction of speed observers

Given that the rotor speed of the IM is measured by a mechanical shaft sensor, then flux estimation is a fairly easy task. Therefore, a speed sensor is usually required to be mounted on the shaft to measure the motor speed. However, a speed sensor increases the cost and requires a connection line between the control system and the motor. In addition, the reliability of the high speed sensor is generally low especially when the whole flywheel (including motor) is floating on magnetic bearing in a vacuum tank. To avoid the problems, there has been an increasing interest in electric drives without mechanical sensors (e.g., tacho machines, optical encoders, resolvers, etc.). Such drives are attractive because of their low cost and high performance. These sensorless methodologies includes slip frequency calculation method; speed estimation using

state equation; estimation based on slot space harmonic voltages; Kalman filtering techniques; neural network based or Fuzzy-logic based sensorless control and etc. [10]- [18]. However, all these approaches are based on on-line estimation strategies when the flywheel is continuously energized.

The main objective of these schemes is that they do not need an extra sensor to obtain the flywheel speed and position information. The speed and position information can be extracted by estimating the motor's back electromotive force (EMF); therefore, the dimensions and cost of the drive system can be further reduced. This approach can be easily applied to any IM, and this design is independent of the feedback controller from the control system. Therefore, in order to validate our speed estimation scheme at standby mode for flywheel energy storage system, our algorithm in this study is also based on this approach with the field-oriented control (FOC) to estimate rotor speed and flux.

Sensorless speed estimation schemes proposed in the past few years can be achieved over a fairly large speed range with very good dynamic performance. In FESS applications, the speed range of the motor usually is limited since the very low speed of flywheel should be avoided. Based on Eqn.(1-1) if the flywheel speed drops to 50% of its rated speed, the stored energy will only have 25% left of its full capacity.

3.2.5.2 Derivation of the proposed speed observer

A satisfactory speed regulation is extremely important not only to produce desired torque performance from the IM but also to guarantee the decoupling between control of torque and flux. This speed estimation scheme is derived from the SCIM from the stator terminal currents and the estimated rotor flux in stationary reference frame. The instantaneous speed information is

updated for each sampling period. A general block diagram for this sensorless field-oriented control with speed and rotor flux observer is in shown in Fig. 3-9 .

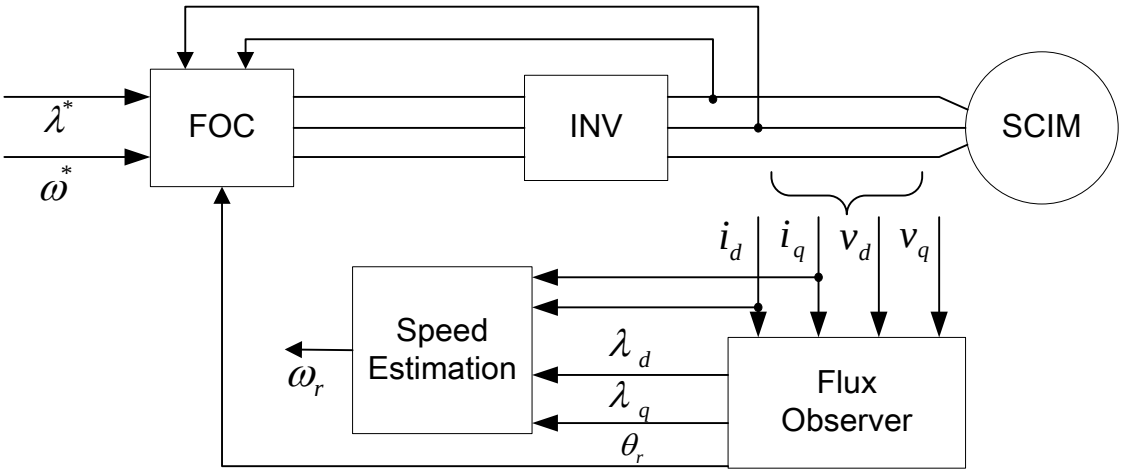


Fig. 3-9 Speed Estimation Block Diagram

From Eqn. (3-11) and Eqn.(3-13), eliminating R_r

$$i'_{dqr} = \frac{1}{L_m p}(v_{dqs} - R_s i_{dqs}) - \frac{L_s}{L_m} i_{dqs} \tag{3-36}$$

Substitute i'_{dqr} into Eqn. (3-10)

$$\omega_r = \frac{p\lambda_{qr} i_{dr} - p\lambda_{dr} i_{qr}}{\lambda_{dr} i_{dr} + \lambda_{qr} i_{qr}} \tag{3-37}$$

Since i_{dr}, i_{qr} are not measurable, replace it with i_{ds}, i_{qs} from Eqn. (3-11)

$$\omega_r = \frac{(\lambda_{ds} - L_s i_{ds})p\lambda_{qr} - (\lambda_{qs} - L_s i_{qs})p\lambda_{dr}}{(\lambda_{ds} - L_s i_{ds})\lambda_{dr} + (\lambda_{qs} - L_s i_{qs})\lambda_{qr}} \tag{3-38}$$

In most cases, the rotor speed ω_r can also be computed as the difference between the synchronous speed ω_e and the slip speed ω_{sl}. The synchronous speed can be derived as:

$$\omega_e = \frac{(p\lambda_{qs})\lambda_{ds} - (p\lambda_{ds})\lambda_{qs}}{\lambda_{ds}^2 + \lambda_{qs}^2} \tag{3-39}$$

And the slip speed, using quantities in stationary reference frame, as:

$$\omega_{sl} = \frac{L_m}{\tau_r} \frac{(\lambda_{ds} i_{qs} - \lambda_{qs} i_{ds})}{\lambda_{ds}^2 + \lambda_{qs}^2} \quad (3\text{-}40)$$

Where λ_{ds}, λ_{qs}: the d and q-axis components of the stator flux

$p\lambda_{dr}$, $p\lambda_{qr}$: the derivative of the d and q-axis components of rotor flux

i_{ds}, i_{qs}: the d and q-axis components of the stator current

L_m: Lm: the magnetizing inductance

τ_r: Rotor time constant.

Therefore,

$$\omega_r = \omega_e - \omega_{sl} \quad (3\text{-}41)$$

3.2.6 Derived Field-Oriented Control for Sensorless Speed Estimation

The most popular induction motor drive control method has been the field oriented control (FOC) in the past two decades. The sensorless control problem may be stated as follows:

(a) Achieve vector control of the induction motor for speed and /or torque tracking;

(b) Be robust to parameter uncertainty;

(c) Handle full rated load-torque on the motor at start-up;

(d) Achieve rated speed of the motor in steady-state.

These conditions are typically required of a field-oriented control system with a sensor. Furthermore, the recent trend in FOC is towards the use of sensorless techniques that avoid the use of speed sensor and flux sensor. Field Oriented Control (FOC) represents the method by which one of the fluxes (rotor, stator or air gap) is considered as a basis for creating a reference frame for one of the other fluxes with the purpose of decoupling the torque and flux-producing components of the stator current. The decoupling assures the ease of control for complex three-phase motors in the same manner as DC motors with separate excitation. This means the

armature current is responsible for the torque generation, and the excitation current is responsible for the flux generation. A typical FOC based IM drive for FESS is presented in [32].

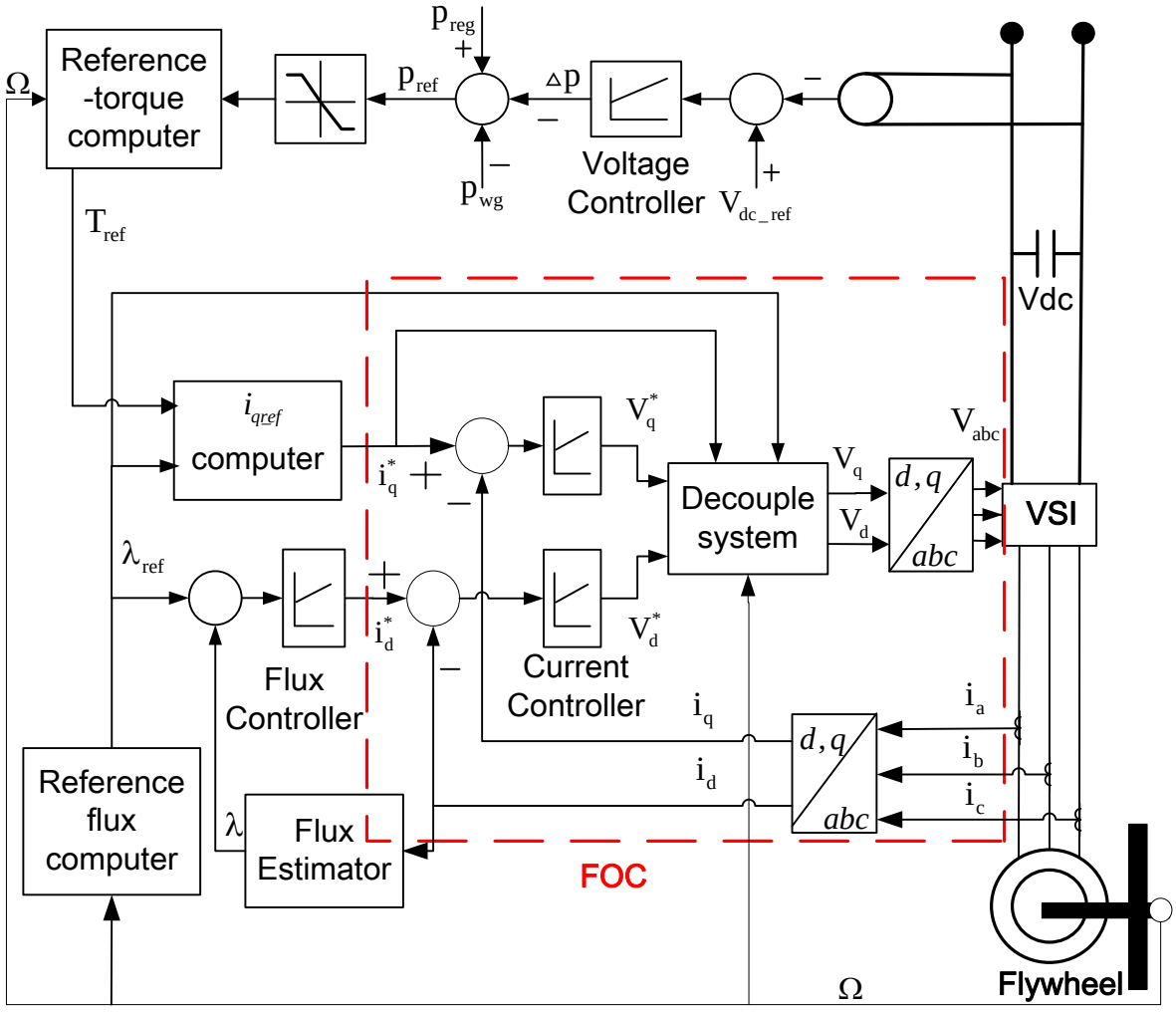


Fig. 3-10 FESS control scheme using sensored FOC for the IM

In the FOC, the d–q reference frame is locked to the rotor-flux vector. Hence, the flux and torque can be separately controlled by the stator direct-axis current id and the quadrature-axis current iq, respectively. In general the flywheel will run at high speed and field weakening may be appropriate; it is noted however that at higher speeds, a given amount of power can be extracted with less change in speed and hence less torque. The proposed techniques are thus

equally valid for field weakening operation. The torque current reference iq* is derived from the Vdc controller, while the flux level is regulated by the id*. Therefore, the FOC accomplishes the following control tasks:

1) to regulate the dc-link voltage;

2) to regulate the power flow on the grid or on an isolated load.

Based on previous analysis, the modified FOC scheme is presented in Fig. 3-11. This modified FOC is implemented in both MATLAB/SIMULINK model and the DSP based converter for simulation and experiment. In this application note, the rotor flux is considered as a reference frame for the stator and air gap flux. This scheme adopts decoupled control of the torque and flux. The flux-producing current component iex is used to excite the IM during speed measurement period. In order to minimize the measuring impact to the speed of FESS, the torque-producing current component i*qs is set to zero to produce zero torque due to the linear relationship under the field-oriented control scheme [25]. Since the speed of flywheel will not change dramatically within a short period during the speed measurement at standby mode, a zero order holder is used to maintain the speed information during the interval when the excitation current is removed.

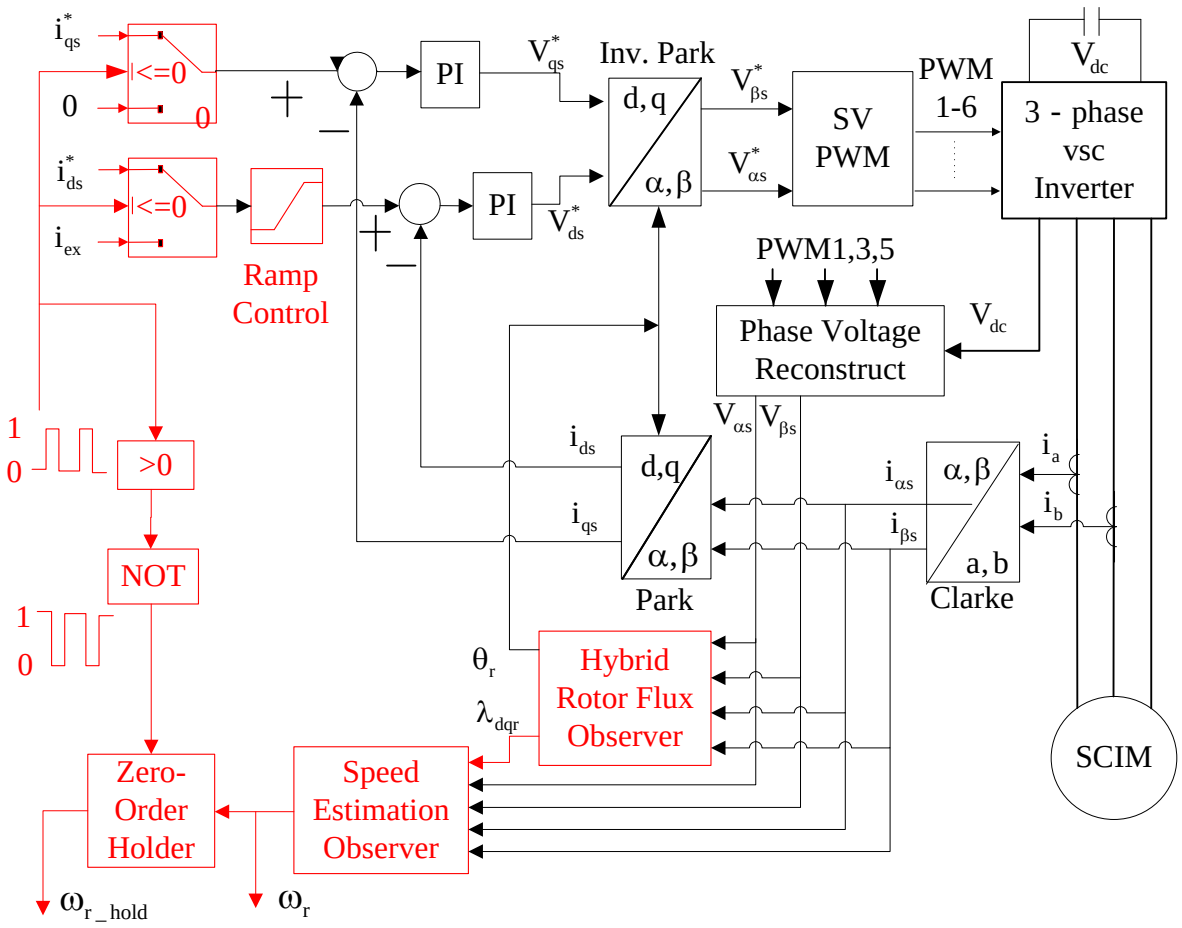


Fig. 3-11 Modified FOC for Sensorless Speed Estimation at standby mode

When the excitation current i_{ex} is applied to the FOC, due to the change in di_{ex}/dt, a large magnitude of transient torque will be generated. This torque will have a significant impact to the FESS in standby mode and create large power loss due to the excitation. In order to minimize this effect, the excitation current i_{ex} is applied to the control through a ramp control block. The ramp control plays an important role for the suppression of the transient torque, which results in least disturbance to the FESS in its standby mode.

Chapter 4:Simulation and Experiment Results

4.1 Simulation and experiment setup

4.1.1 Machine Specification for simulation and experiment

In order to validate the proposed speed estimation scheme, two induction machine models are used. The induction machine parameters are listed in .

Table 4-1.

Table 4-1: Induction Machine Parameters (IM1 & IM2)

Parameters	IM1	IM2
Rated Power (hp)	2	200
Rated Voltage (V)	220/440	575
Frequency (Hz)	60	400
Rated Speed (rpm)	1750	11900
Rs (Ω)	0.6405	0.02475
Ls (H)	0.0452	0.014534
Rr (Ω)	0.3625	0.0133
Lr (H)	0.0452	0.014534
Lm (H)	0.0418	0.01425
# of poles pairs	2	2

IM1 is an actual induction machine used for both simulation purpose and experiment. IM1 is relatively a low speed induction machine which its rated frequency is 60Hz. The parameters of

this machine are extracted from the blocked rotor test and no load test as discussed in chapter 2. In the experiment setup, the FESS is simulated by IM1 which is connected as a SCIM and coupled with a DC motor as a prime mover. The DC motor drives the SCIM to high speed to simulate a standby mode high speed FESS while the SCIM is not energized. The rated parameters of the DC motor are listed in Table 4-2.

Table 4-2 Rated parameters of the DC motor

Items	Values
Rated armature voltage, V_{rated}	125V
Rated armature current, I_{rated}	8A
Rated rotor speed, n_{rated}	1750rpm

Based on the rated parameters of the DC motor, the rated rotor angular velocity is

$$\omega_{r,rated} = 2\pi \frac{n_{rated}}{60} = 183.2 \text{rad/s}$$

IM2 is an asynchronous high speed machine model from MATLAB/SIMULINK library which its rated frequency works at 400Hz. IM2 is used to simulate a flywheel system that works and tested at high speed range.

The experiment is conducted to verify the proposed speed estimation method discussed in previous chapters. The schematic setup for both the experiment and simulation is illustrated in Fig. 4-1. Since the DC motor and the SCIM are mechanically coupled, hence, by varying the supplied voltage to the DC motor, the SCIM is driven by the DC motor to rotate at the same speed as DC motor. The prototype of the FESS is verified under various speed conditions. The schematic setup of the experiment is shown in Fig. 4-1.

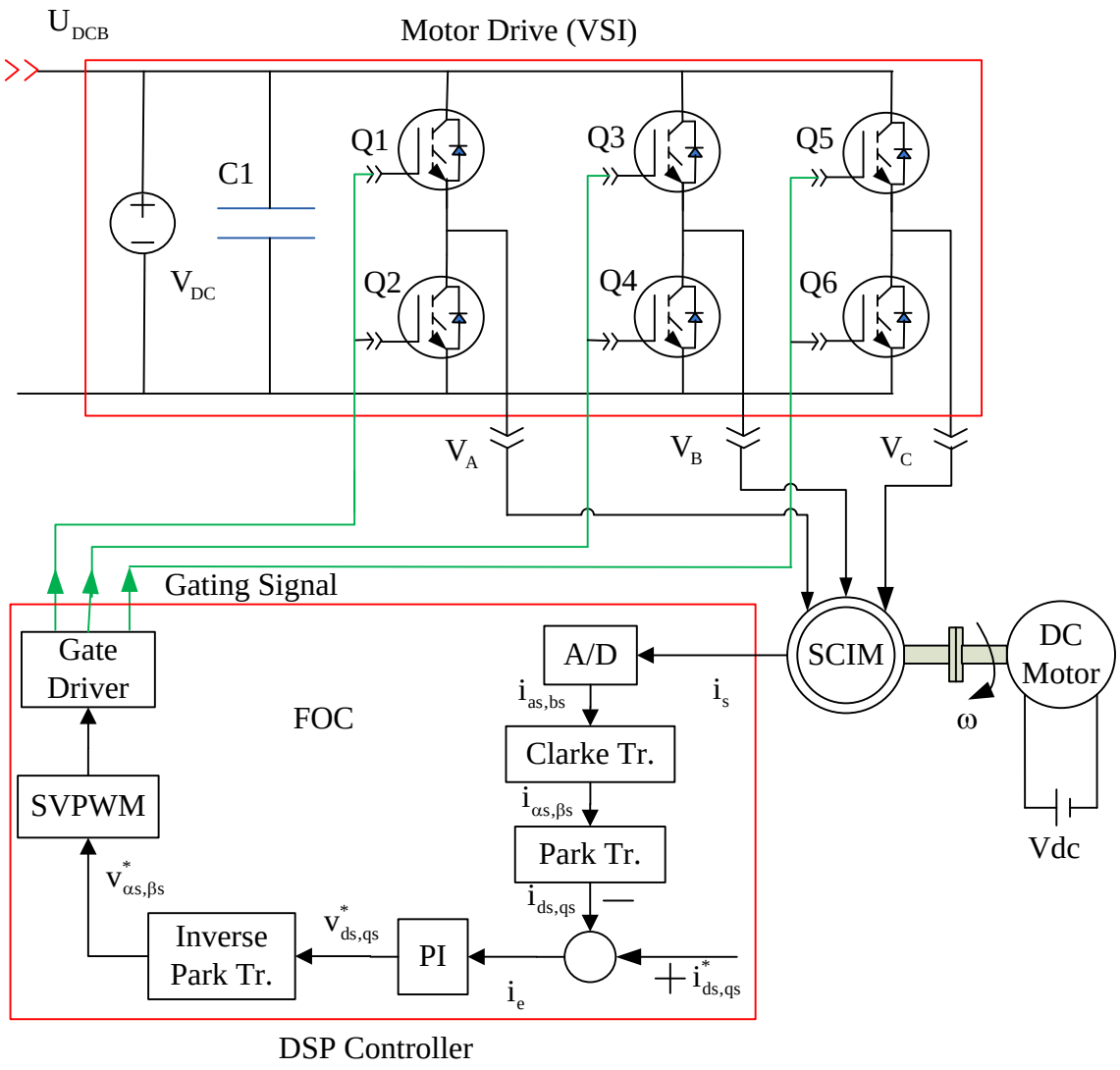


Fig. 4-1 Schematic setup of the FESS experiment

4.1.2 The DSP based motor drive

With modern power electronics and advanced microprocessor technology, AC Motor Drives are able to efficiently control motor speed, improve machine automation and save energy. Modern AC Drives can accurately control speed and torque, smoothly handle an increased load, and provide numerous custom control and configuration operating modes. It provides a full range

of motor control technologies and is used throughout a wide range of industries, to enhance and improve machine automation.

For this project, the motor drive consists of a back-to-back converter and the control part which is implemented on the eZdsp^{TM} F2812 DSP board. The eZdsp^{TM} F2812 from Texas Instrument belongs to Texas Instruments' TMS320 family, and is a stand-alone card--allowing evaluator. This module is an excellent platform to develop and run software for the back-to-back converter control.

Texas Instruments' TMS320 family consists of fixed-point, floating-point, multiprocessor digital signal processors (DSPs). These DSPs have an architecture designed specifically for real-time signal processing. The F2812 is a member of the 'C2000 DSP platform, and is optimized for control applications. The eZdsp^{TM} F2812 is used for real time digital control systems. It contains the TMS320F2812 Digital Signal Processor with 150 MIPS operating speed, 16 PWM outputs, up to 128x16k flash, on-chip RAM and off-chip SRAM memory. In addition to these, it also contains 16 channels of 12-Bit ADC, up to 56 General Purpose I/O (GPIO) pins, timers and counters, on board IEEE 1149.1 JTAG emulation connector for real-time sampling and communications, etc. This scheme is very useful for electronic control systems such as artificial intelligent electromechanical control valves, electrical vehicles (EV) and hybrid vehicles which controls nonlinear and event driven process or systems in motor drives applied in automotive, aerospace and power plants. The 'F28x series of DSP controllers combine this real-time processing capability with controller peripherals to create a suitable solution for vast majority of control system applications.

The event manager function of the F2812 is application-optimized peripheral unit, coupled with the high-performance DSP core, enables the use of advanced control techniques for high-precision and high-efficiency full variable-speed control of motors. Included in the event manager are special pulse-width modulation (PWM) generation functions, such as a programmable dead-band function and a space vector PWM state machine for 3-phase motors that provides quite a high efficiency in the switching of power transistors. Three independent up/down timers, each with its own compare register, support the generation of asymmetric (non-centered) as well as symmetric (centered) PWM waveforms. The eZDSP F2812 board offers 12 PWM outputs which can be used to control the back-to-back converter. For this experiment, one group of PWM output from eZDSP F2812 is used to interface with electronic commutation driver to control the excitation level and torque. The proposed FOC scheme, including the PI controller and the Space Vector Pulse Width Modulation (SVPWM) is implemented on the eZdspTM F2812 based converter to control the voltage source inverter. The SVPWM pulse machine model is also modeled in MATLAB to generate firing pulses for the flywheel converter and interfaced to the converter.

4.2 Simulation Results

4.2.1 Simulation Results from IM1

In order to simulate the working conditions of the FESS at standby mode, the FESS is represented by an SCIM model rotating at high speed. The running time of IM1 is divided into 3 segments:

0s - 0.1s: The SCIM model is set to mechanical speed input at its rated condition to simulate a rotating high-speed flywheel. No excitation current is supplied to the SCIM during this period. This segment simulates that the flywheel is working at the standby mode when no flywheel speed information is required.

0.1s - 0.4s: The gating signals of the VSC2 are enabled (Fig. 4-2A). The flux-producing current i_{ex} is supplied through the modified FOC to the SCIM to excite the machine for speed measurement. The flux change from the hybrid rotor flux observer is shown in Fig. 4-2B. Due to the flux change during this period, a transient torque is generated as shown in Fig. 4-2C . The terminal parameters are sampled by A/D converter on DSP, and the speed information is extracted from these induced terminal parameters and updated at every sampling period (Fig. 4-2D).

0.4s - 0.5s: The speed measurement process is completed, and the excitation current i_{ex} is removed by setting it to zero. The FESS enters into standby mode again.

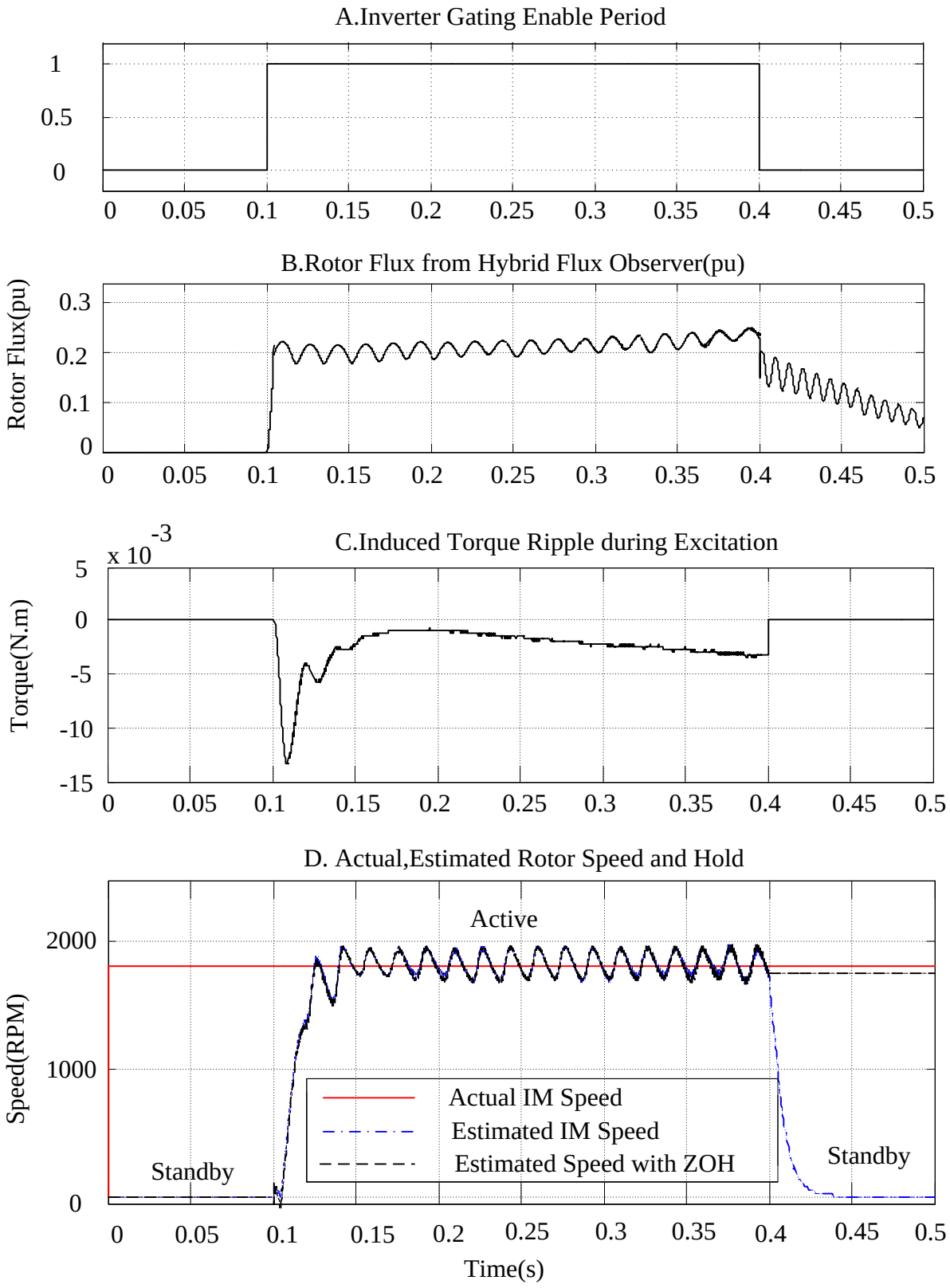


Fig. 4-2 Simulation Results from IM1

The excitation level i_{ex} is selected based on the consideration of the impact of the measurement to the exiting FESS. A higher excitation level will produce a more accurate result for the speed estimation. However, the higher the excitation is, the larger the transient torque will be generated, therefore, more disturbances to the FESS. For this simulation, the excitation current reference is set to 0.2 pu.

In order to minimize the impact from the process of the speed estimation to the FESS, the torque from the modified FOC scheme is maintained to zero by setting the reference i*qs=0 along with mathematical constrain $\lambda_{qr} = 0$ from the modified FOC. However, due to the sudden change of the excitation current i*ds, a torque ripple is generated as shown in (Fig. 4-2C) during the transient measurement period. This generated short-period torque is very small in magnitude and can be negligible, especially when the measurement duration is short enough.

After the excitation is removed from the induction motor, the speed estimation is disabled and the result from last estimation is maintained by a Zero-Order Holder(ZOH) to next excitation (Fig. 4-2D). In order to minimize the torque impact caused by the flux change due to the excitation, the excitation period should be as short as possible. However, the duration of the excitation time needs to be larger enough than the settling time of the estimation. The settling time of the estimated speed takes about 60 ms to reach steady state. This value can be further improved by adjusting the cut-off frequency of the speed filter in the speed estimation observer for a trade-off of higher percentage of overshoot caused by the flux change. The trade-off of

settling time of the estimated speed and accuracy should be carefully investigated for different machines.

During the estimation cycle, the calculated d,q axis current reference and the measured current component are shown in Fig. 4-3. The d-axis current is increasing since the measurement is taken at the excitation transient process. For q-axis current component, the average of the q-axis current component remains zero in order to achieve zero torque if the ripple is not considered.

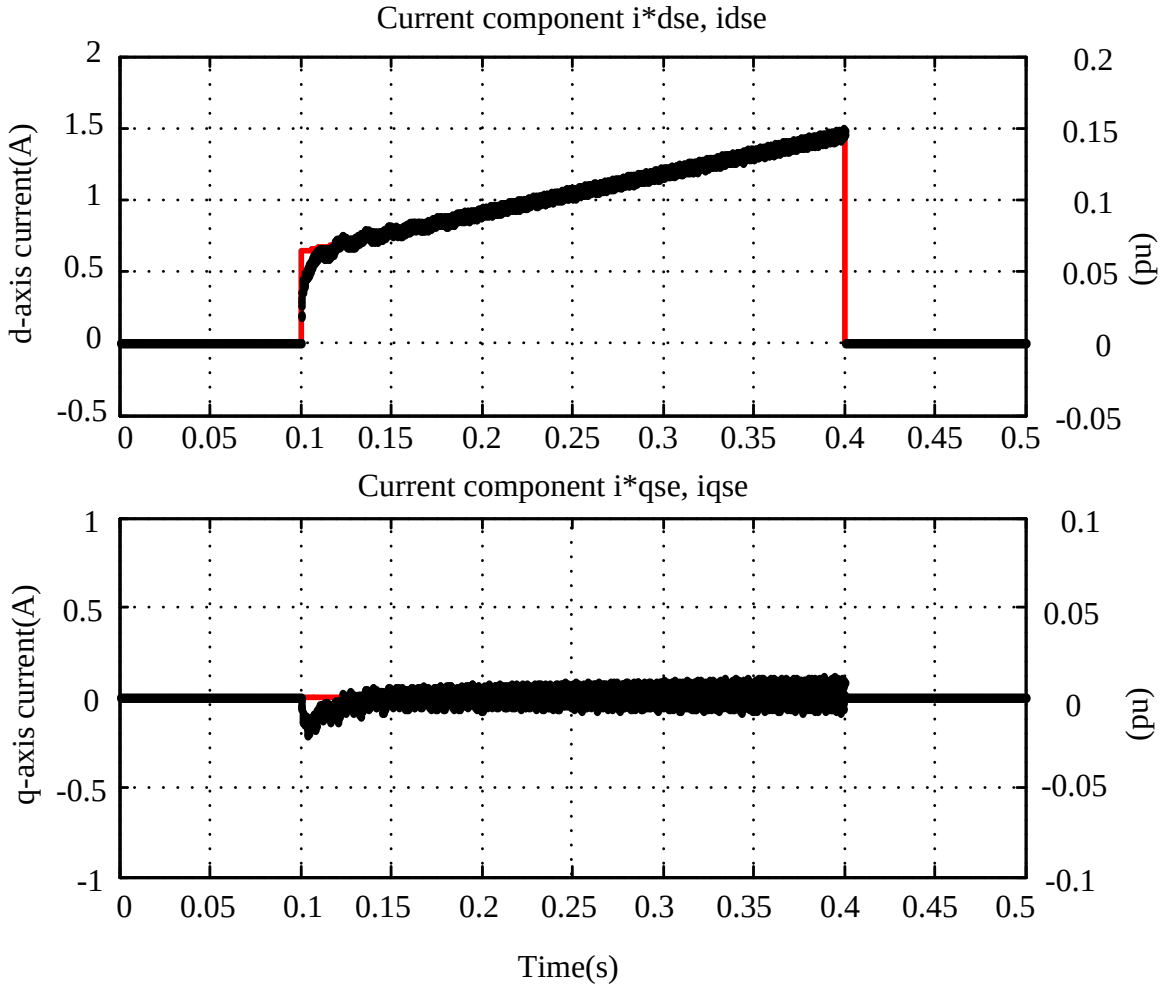


Fig. 4-3 d, q-axis reference and actual current during excitation

4.2.2 Simulation results from IM2

In order to evaluate the proposed approach for a high speed FESS at standby mode, a simulation is conducted on IM2. The running time of the simulation is divided into 4 segments:

0s - 0.1s: The SCIM model is set to mechanical speed input at its rated condition to simulate a rotating high-speed flywheel. No excitation current is supplied to the SCIM during this period. This segment simulates that the flywheel is working at the standby mode with no flywheel speed information required.

0.1s - 0.4s: A flux-producing current iex is supplied through the modified FOC to the IM2 to excite the machine for speed measurement. The speed information is extracted from induced terminal parameters.

0.4s - 0.5s: The speed measurement process is completed, and the excitation current is removed. The FESS enters into standby mode again.

0.5s - 1s: This period shows another speed estimation cycle to simulate the repeating pattern of speed estimation process.

For this simulation, the excitation current level is set to 0.4 pu. In order to minimize the impact of the speed estimation to the FESS, the torque from the modified FOC scheme is maintained to zero by setting i*qs =0. The generated torque ripple is shown in (Fig. 4-4C). This generated short-period torque is very small and can be negligible.

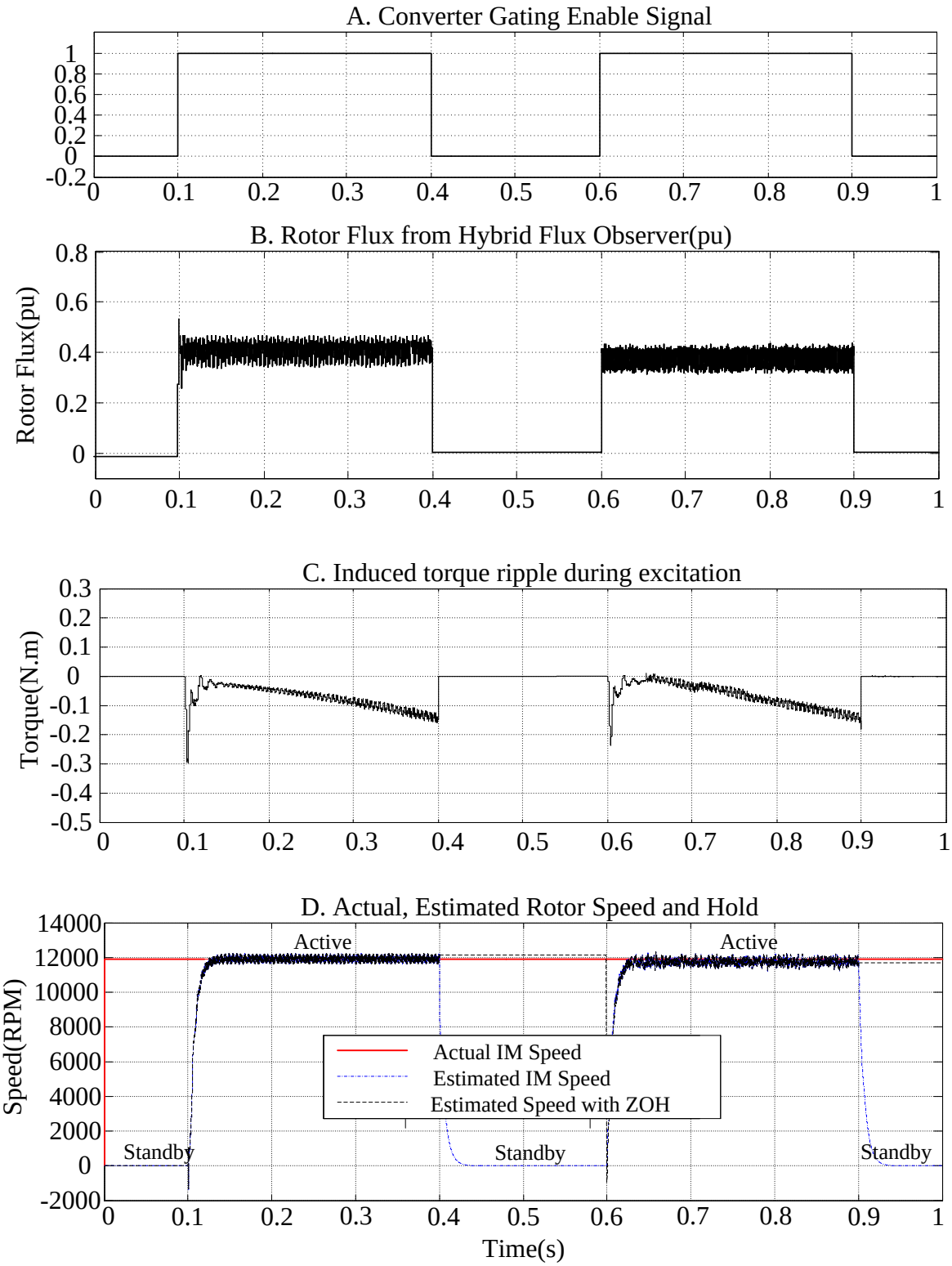


Fig. 4-4 Simulation results from IM2

After the excitation is removed from the induction motor, the speed estimation is disabled and the result from last estimation is kept by the zero-order holder until next excitation. The duration of the excitation is limited to a few milliseconds, which is larger than the setting time of estimation output. Of course, the excitation should be as small as possible to further reduce the iron loss. But the tradeoff of setting durations of excitation and accuracy should be carefully investigated.

4.3 Experimental setup and results

The experiment is conducted with the induction motor(IM1), connected as a SCIM and coupled with a DC motor as prime mover to bring the SCIM to high speed. The SCIM is fed through a 300 V, 30A, MOSFET inverter with Pulse Width Modulation (PWM). The DC link voltage of the inverter is set to 100V for a common DC panel connection. Field oriented Control with pulse width modulation, is implemented on a Digital Signal Processor (DSP) using the Texas Instrument (TI) F2812 microprocessor based converter. To avoid sampling distortion, the sampling frequency is set to 10 kHz, which is much higher than the rated frequency of the stator voltage and current at 60Hz. For the experiment, the SCIM stator voltage and current are measured from the Analog/Digital converter on the DSP.

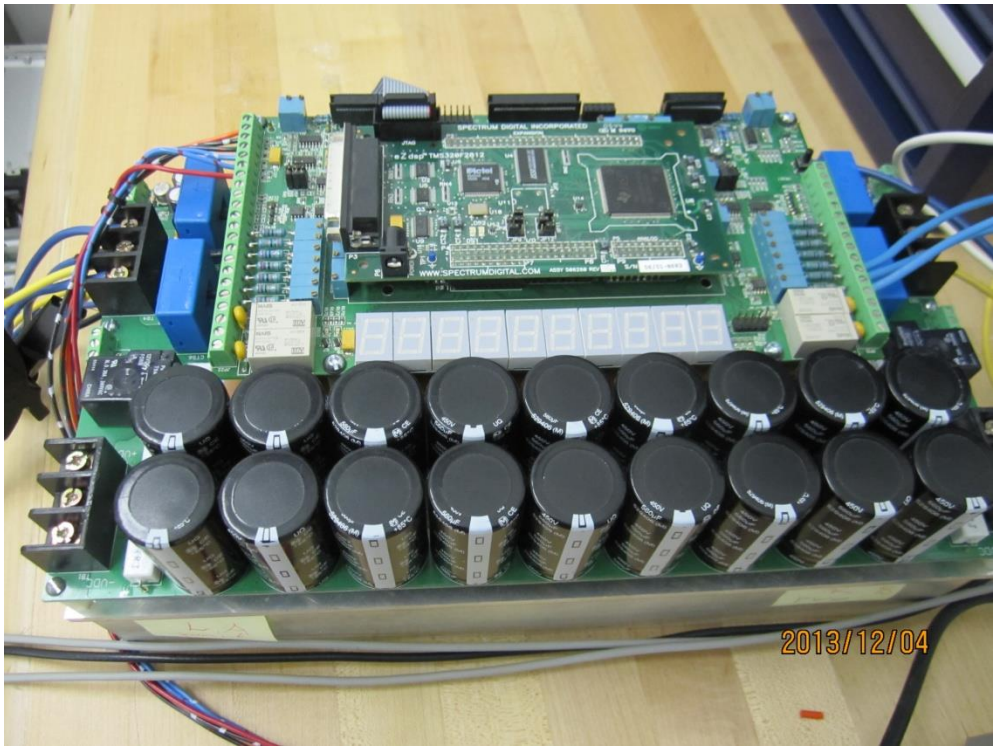


Fig. 4-5 Back-to-back Converter

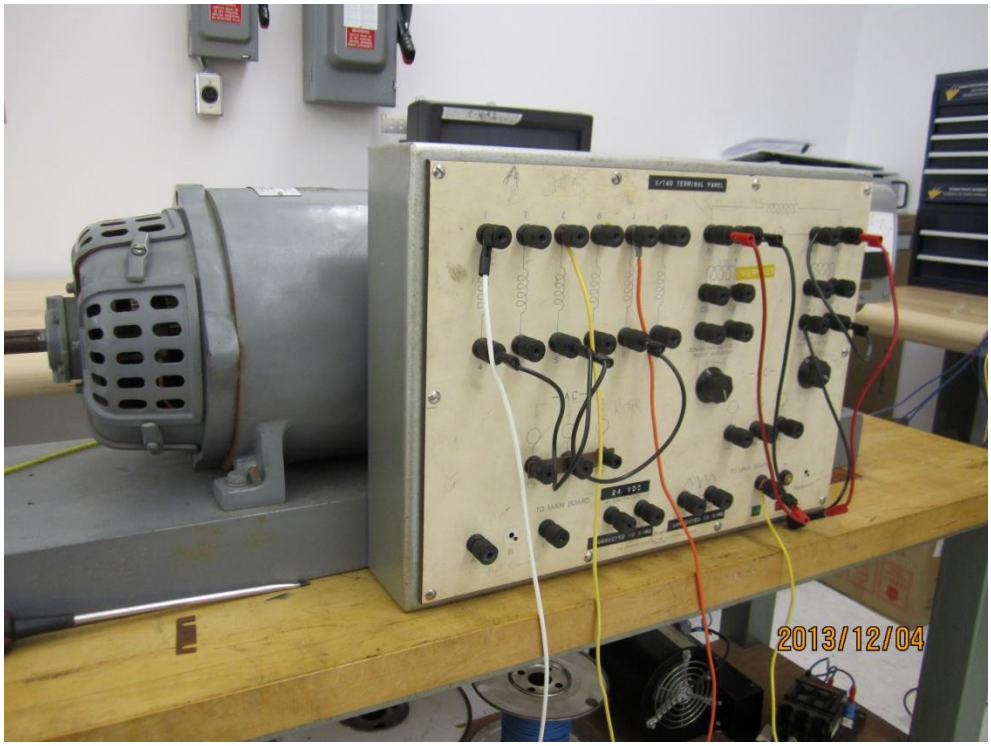


Fig. 4-6 Squirrel cage induction machine

For this experiment, the actual measured speed from a tachometer reading is about 832 rpm. To minimize the torque ripple with the least settling time, the practical excitation level is selected between 0.2 - 0.45 pu. For this experiment, the excitation current reference is set to 0.4pu.

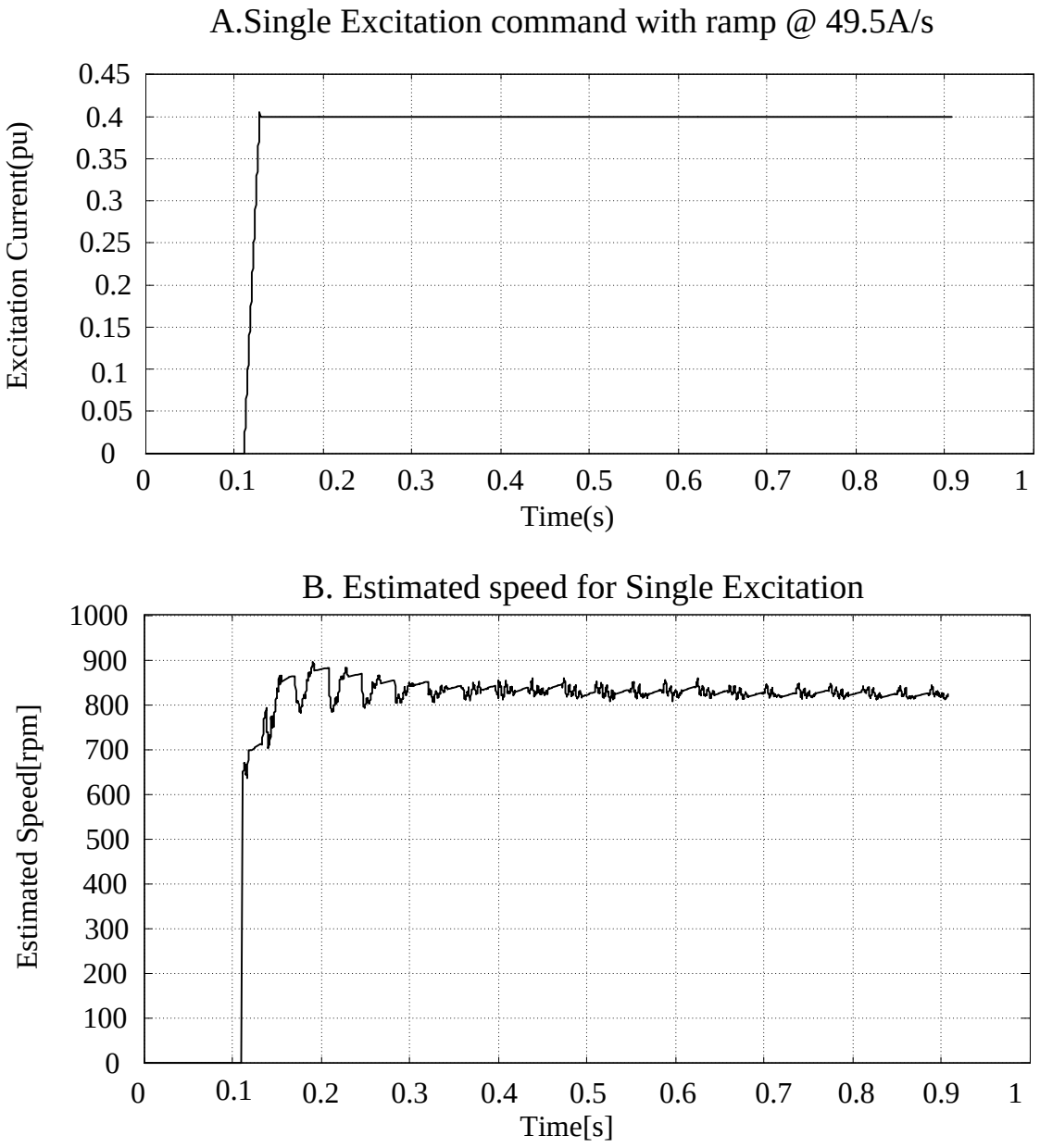
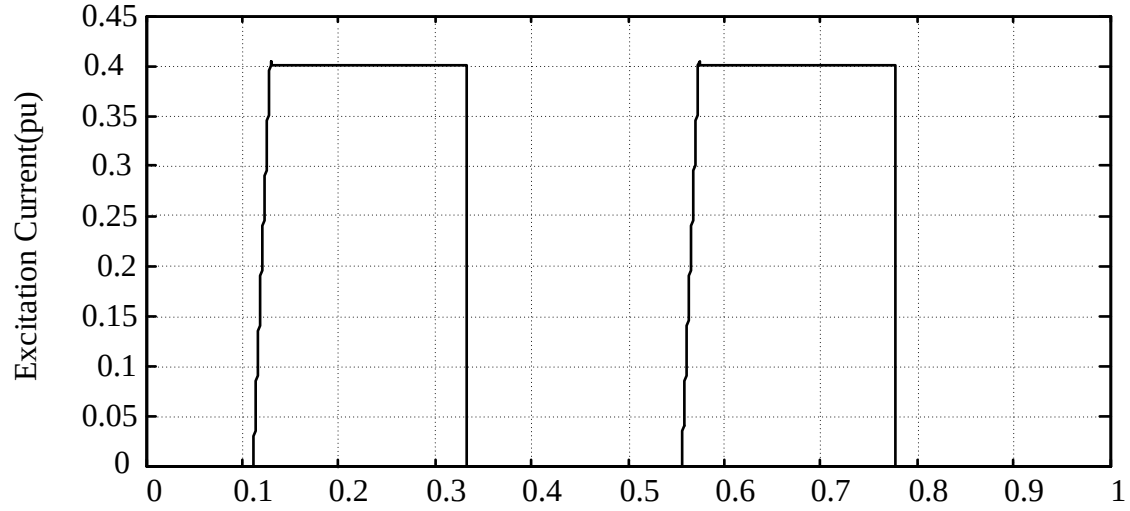


Fig. 4-7 Single Excitation with ramp @ 49.5A/s and Estimated Speed

From Fig. 4-7A, a single excitation current command that goes through the ramp control is supplied to the stator side of the SCIM while it is mechanically driven by a DC motor. The resulted estimated speed is shown in Fig. 4-7B, with a speed magnitude ranging from 800 to 850rpm, which is consistent with the measured speed at 832 rpm.

A. Repeating Excitation command with ramp @ 49.5A/s



B. Estimated speed for repeating excitation

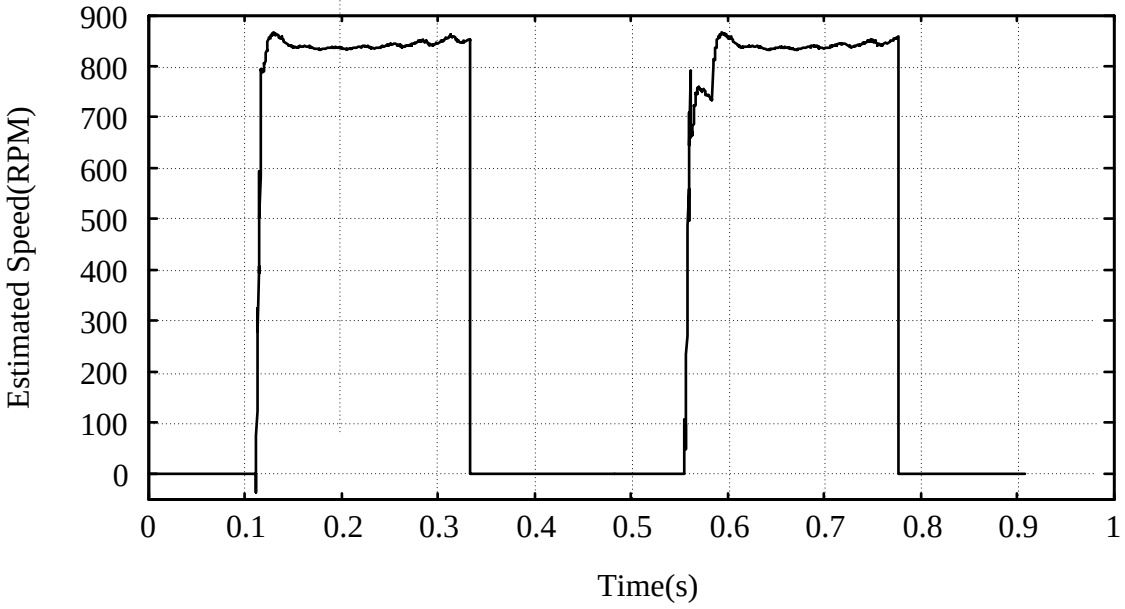


Fig. 4-8 Repeating Excitation with ramp @ 49.5A/s and Estimated Speed

In order to further test the repeating measurement of the estimated speed, two excitation pulses are supplied to the SCIM in Fig. 4-8A. The estimated speed is shown in Fig. 4-8B.

4.3.1 Investigation on settling time vs. ramp rate control

For a lower excitation level, the accuracy of the estimated speed is distorted by the small magnitude of the induced voltage and current; while a large excitation level will cause a large induced current at the moment when a flux excitation command is given, which will trigger the over current protection on the converter in the experiment, or large overshoot and oscillation problems which will increase the settling time of the estimated speed. In order to overcome the sudden change of the applied flux excitation current, the excitation current is applied with a ramp control (Fig. 3-11). The relationship between the setting times vs. ramp rate is investigated in Fig. 4-9. Based on the Fig. 4-9, the practical current excitation level is selected between 0.3 - 0.45pu. For this experiment, the selected i_{ex} ramp rate is 0.005(0.011A/T sampling or 49.5A/s), to minimize the torque ripple with least settling time.

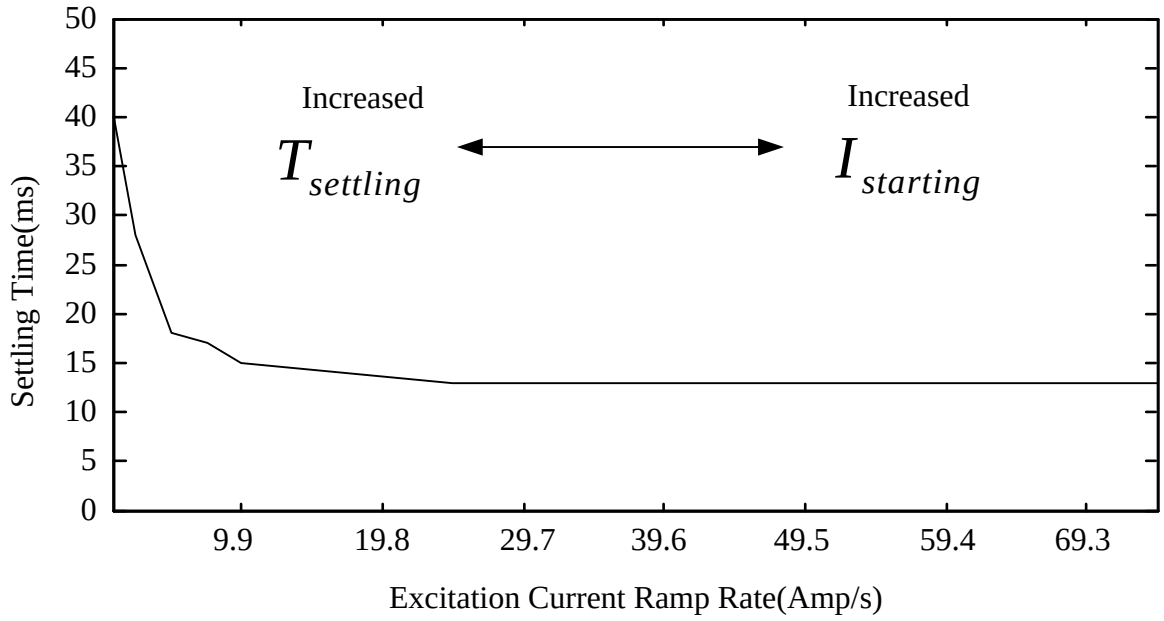


Fig. 4-9 Settling time of estimated speed at various excitation level

4.3.2 Investigation on generated torque vs. ramp rate control

A torque ripple will be generated due to the excitation process which will cause impact to the FESS. The ramp control technique will not only improve the settling time, but also reduce the generated torque ripple from the excitation process. A comparison of the generated torque at two different conditions is shown in Fig. 4-10. It demonstrates that the ramp control technique can effectively reduce the torque ripple at the beginning of the excitation process, which results in minimized impact to the FESS. The higher the ramp rate, the larger the generated torque ripples.

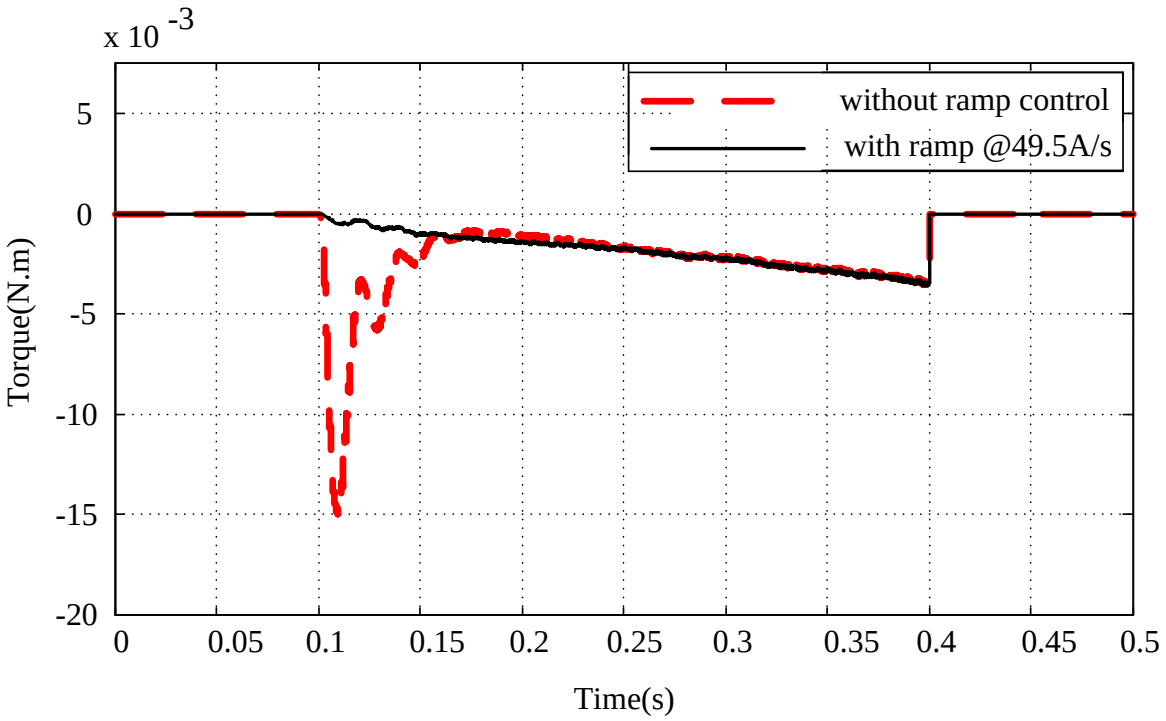


Fig. 4-10 Comparison of the generated torque with/without ramp control

Chapter 5:Conclusions and future works

5.1 Conclusions

In this thesis, the induction motor driven flywheels and sensorless speed estimation techniques are reviewed. Based on previous researches, a back-EMF based sensorless speed estimation approach for FESS at standby mode is proposed. Various techniques have been applied to the proposed control scheme to achieve the minimal time cost and least disturbance to the existing FESS. A hybrid flux observer seamlessly incorporates the fundamental current model and voltage model flux observer in an effort to take advantage of the two models, and is used to achieve the instantaneous rotor flux estimation during the speed estimation period. The speed information is extracted from terminal parameters and updated in every sampling period.

Since the flywheel speed will not change dramatically in standby mode, the speed information during the intervals between two measurements is held to and updated by next estimation cycle. Moreover, since the FESS does not need to be continuously energized for speed estimation, the power loss of the FESS can be further reduced while the speed information is maintained.

The optimal excitation level is investigated in this paper to minimize the impact to the energy stored in the flywheel. The simulation and experimental results validate the proposed algorithm for sensorless speed estimation at standby mode.

In this study, the speed sensor for FESS is eliminated to reduce the cost with improved reliability of FESS. This also provides the possibility for the integrated design of the long-term flywheel energy storage system.

5.2 Major Contributions

This study proposes a novel approach for the sensorless speed estimation technique at standby mode for FESS. Simulation and experimental results have demonstrated the effectiveness of this technique. With previous researches, this thesis extends the speed monitoring capabilities from the charge/discharge mode into the full operation period of the FESS.

The main contribution of the thesis is as following:

1. Proposed the novel standby mode sensorless speed estimation approach for the long term flywheel energy storage system. Previous researches have proposed several sensorless speed estimation methods for on-line charge/discharge mode. These sensorless estimation approaches focus on the adaptive approach in an effort to achieve higher estimation accuracy in the cost of estimation time and can be computationally intensive. In addition, the speed estimation for standby mode is not covered and the power loss during the estimation period is not considered. This proposed sensorless speed estimation approach provides a solution for the standby mode with the minimum disturbance to the FESS, and acceptable accuracy of the rotor speed with fast settling time. Therefore, the speed estimation can cover the full operating cycle of FESS: charge/discharge and standby mode.
2. Based on the induction machine and conventional field-oriented control, a modified FOC scheme is derived to achieve the control of the FESS speed monitoring at standby mode. This modified FOC can be easily integrated into the conventional FOC to achieve the control and speed estimation for the FESS in full operation period. After

the speed information is obtained, a zero-order holder is used so that the speed information can be maintained between two estimation intervals, and it will updated by next estimation cycle in an effort to reduce the estimation frequency.

3. In order to minimize the disturbance caused by the speed monitoring process, a ramp control technique is applied to the excitation process for a balanced control between the settling time of the estimated speed with the generated transient torque. The relationship between the settling time of the estimated speed with the applied ramp control rate is investigated. A comparison of the generated torque with ramp control and no ramp control is also presented to emphasize the importance of this technique in an effort to minimize the power loss due to the excitation process. The ramp rate control also has an effect on the settling time of the estimated speed. The increased ramp rate will cause a quick response time of the estimated speed but with an increasing tendency of overshoot and oscillation of the estimated speed which increases the settling time of the estimated speed , while the slow ramp rate will also cause a slow response and increased settling time of the estimated speed. A try and error method is used to find the best estimation result for the simulation and experiment.
4. The various excitation levels are simulated. A conclusion is drawn that the higher the excitation level, the higher the accuracy of the estimated speed but with higher power loss due to the higher generated transient torque. An excitation level ranging from 0.2pu to 0.4 pu is tested for simulation and experiment with acceptable accuracy of the estimated speed.

5. By providing the excitation to the FESS, the machine is magnetized for a short duration so that the speed information can be sensorlessly extracted from the induced terminal parameters. Experiment and simulation are conducted, and the results validate the proposed standby mode sensorless speed estimation approach.

5.3 Future Research Work

The results presented here have demonstrated the effectiveness of the standby-mode sensorless speed estimation approach, The models developed in this thesis could be used as a starting point to investigate the following:

• The SCIM based FESS can be controlled to operate above its rated speed in an effort to achieve high energy content by using flux weakening control with the field-oriented control technique. This proposed speed estimation technique will be further tested at the rated speed range above.

• The proposed technique achieves the sensorless speed estimation at standby mode for FESS. Accuracy should be further improved to obtain close tracking of the flywheel speed. In addition, the robustness of the algorithm should be studied and improved for an adaptation of machine parameter variations and speed variations.

• More detailed analysis of the speed estimation algorithm should be performed. The laboratory results will be compared with the simulation results and the flywheel energy storage system performance can be analyzed.

Reference

[1] G. Genta, "Kinetic Energy Storage," *University Press, Cambridge,* 1985.

[2] K. Veszprémi and I. Schmidt, "Different flywheel energy storage drives for renewables — limits and optimization," *Power Electronics and Motion Control Conference (EPE/PEMC),* pp. T12-131-T12-137., 2010.

[3] A. Ter-Gazarian, "Energy Storage Systems For Power Systems," *IEE Power,* 1994.

[4] Beacon Power Corp, "Frequency Regulation and Flywheels fact sheet," 11 July 2011..

[5] R. Hebner, J. Beno and A. Walls, "IEEE spectrum," 2002. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=993788. [Accessed 10 2013].

[6] N. Hamsic, A. Schmelter, A. Mohd, E. Ortjohann, E. Schultze, A. Tuckey and J. Zimmermann, "Increasing Renewable Energy Penetration in Isolated Grids Using a Flywheel Energy Storage System," *Power Engineering, Energy and Electrical Drives, 2007. POWERENG 2007. International Conference on,* pp. pp.195,200, 12-14 April 2007.

[7] Z. a. Z. Long, "Modeling and control of a flywheel energy storage system for uninterruptible power supply," *Sustainable Power Generation and Supply,* pp. 1-6, 2009.

[8] J. W. Park, K. Lee and H. J. Lee, "Control of active power in a doubly-fed induction generator taking into account the rotor side apparent power," *IEEE 35th Power Electronics Specialists Conference, vol.3,,* p. 2060–2064, 2004.

[9] R.Cardenas, R.Pena, G.M.Asher, J.Clare and R.Blasco-Gimenez, "Industrial Electronics, IEEE Transactions on," *Control strategies for power smoothing using a flywheel driven by*

a sensorless vector-controlled induction machine operating in a wide speed range, pp. 603-614, 2004.

[10] M. Huikuri, N. Nevaranta, M. Niemela and J. Pyrhonen, "Sensorless positioning of a non-salient permanent magnet linear motor by combining open-loop current angle rotation method and back-EMF estimator," *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, pp. 3142,3148, Nov. 2013.

[11] Y. Fan, W. Qu, H. Lu, X. Cheng, X. Zhang, L. Wu and S. Jiang, "A slip frequency correction method applied to induction machine," *Electrical Machines and Systems ICEMS 2008. International Conference on*, p. 1122–1125, 2008.

[12] T.-W. Kim and A. Kawamura, "Slip frequency estimation for sensorless low speed control of induction motor," *Advanced Motion Control, 1996. AMC '96-MIE. Proceedings., 1996 4th International Workshop on, vol. 1*, p. 156–161, 1996.

[13] T. Nag, A. Sen and D. Chatterjee, "An observer based on-line rotor speed estimation technique for a vector controlled induction machine," *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, p. 630–633, 2008.

[14] A. Ferrah, P. Hogben-Laing, K. Bradley, G. Asher and M. Woolfson, "The effect of rotor design on sensorless speed estimation using rotor slot harmonics identified by adaptive digital filtering using the maximum likelihood approach," *Thirty-Second IAS Annual Meeting, IAS '97.,Conference Record of the 1997 IEEE, vol. 1,* p. 128–135, 1997.

[15] H.-W. Kim and S.-K. Sul, "A new motor speed estimator using kalman filter in low-speed range," *Industrial Electronics, IEEE Transactions on, vol. 43, no. 4*, p. 498–504, 1996.

[16] Y. Xiaoting, Z. Qingchun and Z. Tao, "Speed estimation of induction motor based on neural network," *Intelligent Control and Information Processing (ICICIP), 2011 2nd International Conference on, vol. 2,* p. 619–623., 2011.

[17] R. Prakash1 and R. Anita, "Neuro- PI controller based model reference adaptive control for nonlinear systems," *International Journal of Engineering, Science and Technology Vol. 3, No. 6,* pp. 44-60, 2011.

[18] K. Ohyama, G. Asher and M. Sumner, "Low speed and regenerating operation of sensorless vector control system of induction motor using observer gain tuning and high frequency injection technique," *Power Electronics and Applications, 2009. EPE '09. 13th European Conference on* , pp. 1,8, 8-10 , Sept. 2009.

[19] S. Samineni, B. Johnson, H. Hess and J. Law, "Modeling and analysis of a flywheel energy storage system for voltage sag correction," *Electric Machines and Drives Conference, 2003. IEMDC'03. IEEE International , vol.3, no.,* pp. pp.1813,1818, June 2003.

[20] A. Schulz, "TU WIEN," Institut für Mechanik und Mechatronik, 25 January 2013. [Online]. Available: http://www.mec.tuwien.ac.at/messtechnik_und_aktorik/messtechnik_und_aktorik/forschung/projekte/flywheel/.

[21] s. balu, "Bright hub engineering," 2 June 2011. [Online]. Available: http://www.brighthubengineering.com/diy-electronics-devices/43723-how-are-squirrel-cage-induction-motors-constructed/. [Accessed 2 10 2013].

[22] D. L. McKinnon and H. A. Smolleck, "Influence of rotor residual flux on the measurement

of inductance and its possible use as an impending fault indicator," *2004 PdMA Corporation,* 2004.

[23] R.Cardenas, R.Pena, G.Asher and J.Clare, "Control strategies for energy recovery from a flywheel using a vector controlled induction machine," *Power Electronics Specialists Conference, 2000. PESC 00. 2000 IEEE 31st Annual , vol.1, no.,,* pp. 454,459 vol.1, 2000.

[24] H. Akagi and H. Sato, "Control and performance of a doubly-fed induction machine intended for a flywheel energy storage system," *Power Electronics, IEEE Transactions on , vol.17, no.1,,* pp. pp.109,116, Jan 2002.

[25] B. Wu, Y. Lang, N. Zargari and S. Kouro, Power Conversion and Control of Wind Energy Systems, Toronto c, 2010.

[26] A. Tapia, G. Tapia, J. X. Ostolaza and J. R. Sanenz, "Modeling and control of a wind turbine driven doubly fed induction generator," *IEEE Trans. Energy Convers,* pp. 194-204, 2003.

[27] X. L. T. Yifan, "Vector control and fuzzy logic control of doubly fed variable speed drives with DSP implementation," *IEEE Trans. Energy Convers -vol.10, no.4,* pp. 661-668, 1995.

[28] K.-K. Shyu, H.-J. Shieh and S.-S. Fu, "Model reference adaptive speed control for induction motor drive using neural networks," *Industrial Electronics, IEEE Transactions on* , pp. 180,182, Febrary 1998.

[29] U.A.Bakshi and M.V.Bakshi, Electrical Machine II, Technical Publications Pune, 2009.

[30] K. Zhao and X. You, "Speed estimation of induction motor using modified voltage model flux estimation," *Power Electronics and Motion Control Conference, 2009. IPEMC '09.*

IEEE 6th International, p. 1979–1982, 2009.

[31] J. Holtz, "Proceedings of the IEEE," *Sensorless control of induction motor drives,* pp. 1359-1394, 2002.

[32] G. Cimuca, S. Breban, M. Radulescu, C. Saudemont and B. Robyns, "Design and Control Strategies of an Induction-Machine-Based Flywheel Energy Storage System Associated to a Variable-Speed Wind Generator," *Energy Conversion, IEEE Transactions on , vol.25, no.2,* pp. pp.526,534, June 2010.

Appendices

5.4 Appendix A: Matlab/Simulink Blocks Model

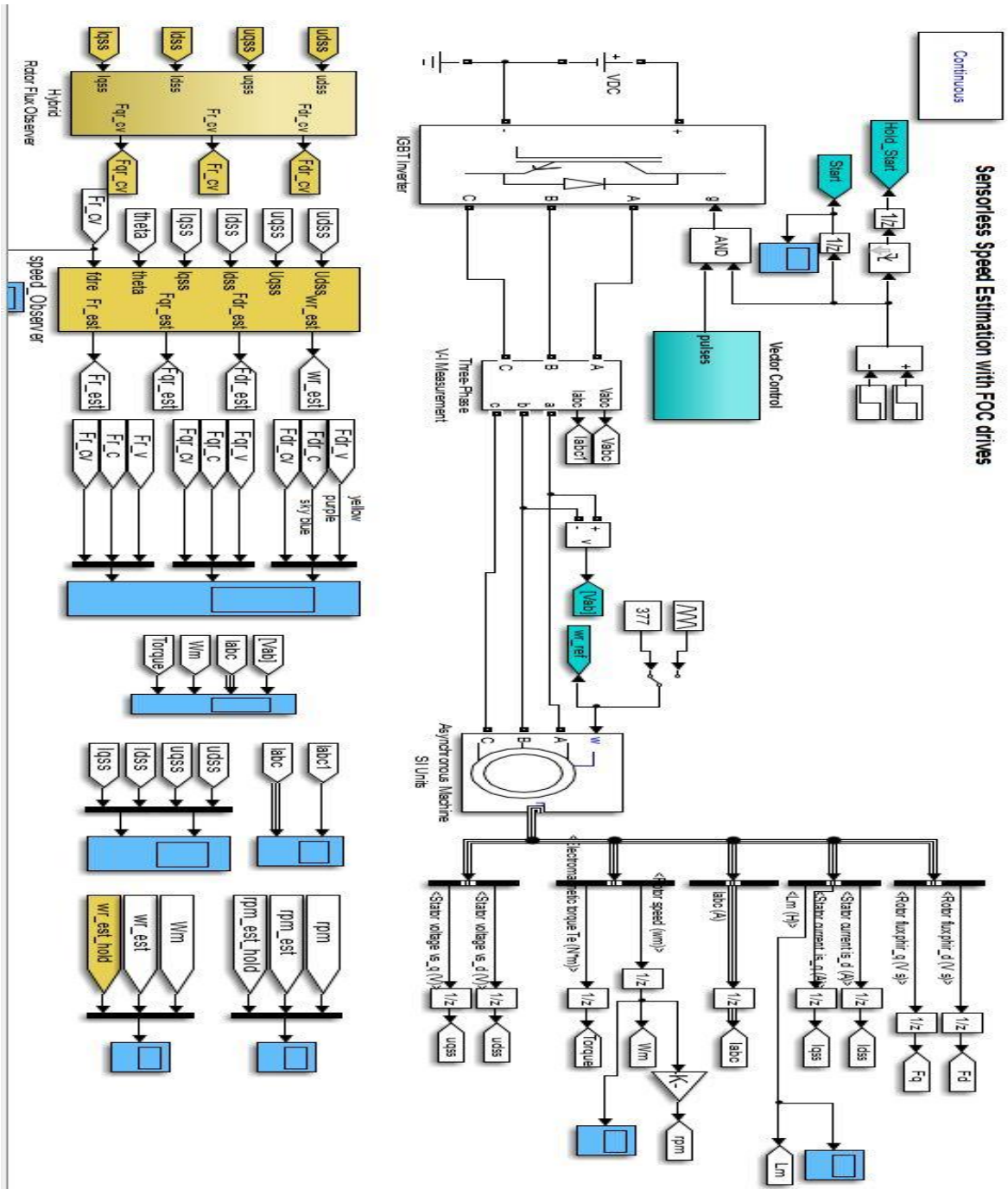

Sensorless Speed Estimation with FOC drives
Continuous
Hold_Start
Start
AND
VDC
IGBT Inverter
pulses
Vector Control
Three-Phase
V-I Measurement
Vabc
Iabc
Vabc1
Iabc1
Asynchronous Machine
SI Units
[Vab]
wr_ref
377
<Rotor flux phir d (V.s)>
<Rotor flux phir q (V.s)>
Fd
Fq
<Stator current is_d (A)>
<Stator current is_q (A)>
Idss
Iqss
<Lm (H)>
Lm
Iabc (A)
Iabc
<Rotor speed (wm)>
Wm
rpm
<Electromagnetic torque Te (N*m)>
Torque
<Stator voltage us_d (V)>
<Stator voltage us_q (V)>
Udss
Uqss
Hybrid
Rotor Flux Observer
speed_Observer
Fdr_cv
Fr_cv
Fqr_cv
Udss_wr_est
fdfre Fr_est
theta
Fdr_est
Fqr_est
Fr_est
wr_est
Fdr_cv
Fdr_c
Fdr_v
Fqr_cv
Fqr_c
Fqr_v
Fr_cv
Fr_c
Fr_v
yellow
purple
sky blue
[Vab]
Torque
Wm
Iabc
rpm
rpm_est
rpm_est_hold
wr_est
wr_est_hold

Fig. A-0-1 Sensorless Speed Estimation with FOC

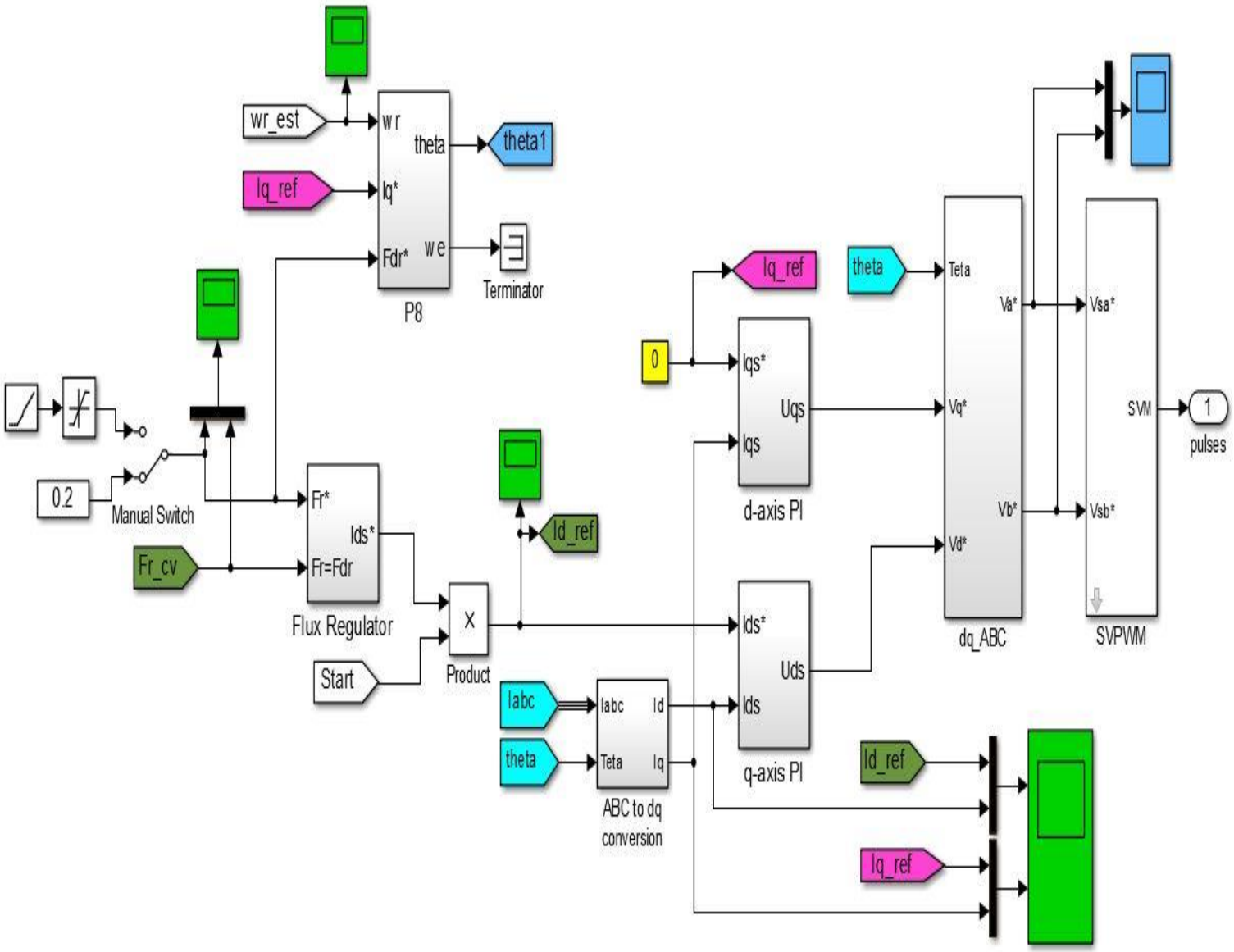

wr_est
wr
theta
theta1
Iq_ref
Iq*
Fdr*
we
Terminator
P8
0.2
Manual Switch
Fr_cv
Fr*
Ids*
Fr=Fdr
Flux Regulator
Id_ref
X
Product
Start
Iabc
theta
Iabc
Id
Teta
Iq
ABC to dq conversion
0
Iq_ref
Iqs*
Uqs
Iqs
d-axis PI
Ids*
Uds
Ids
q-axis PI
theta
Teta
Va*
Vq*
Vb*
Vd*
dq_ABC
Vsa*
Vsb*
SVM
SVPWM
1
pulses
Id_ref
Iq_ref

Fig. A-0-2 Modified Field-Oriented Control

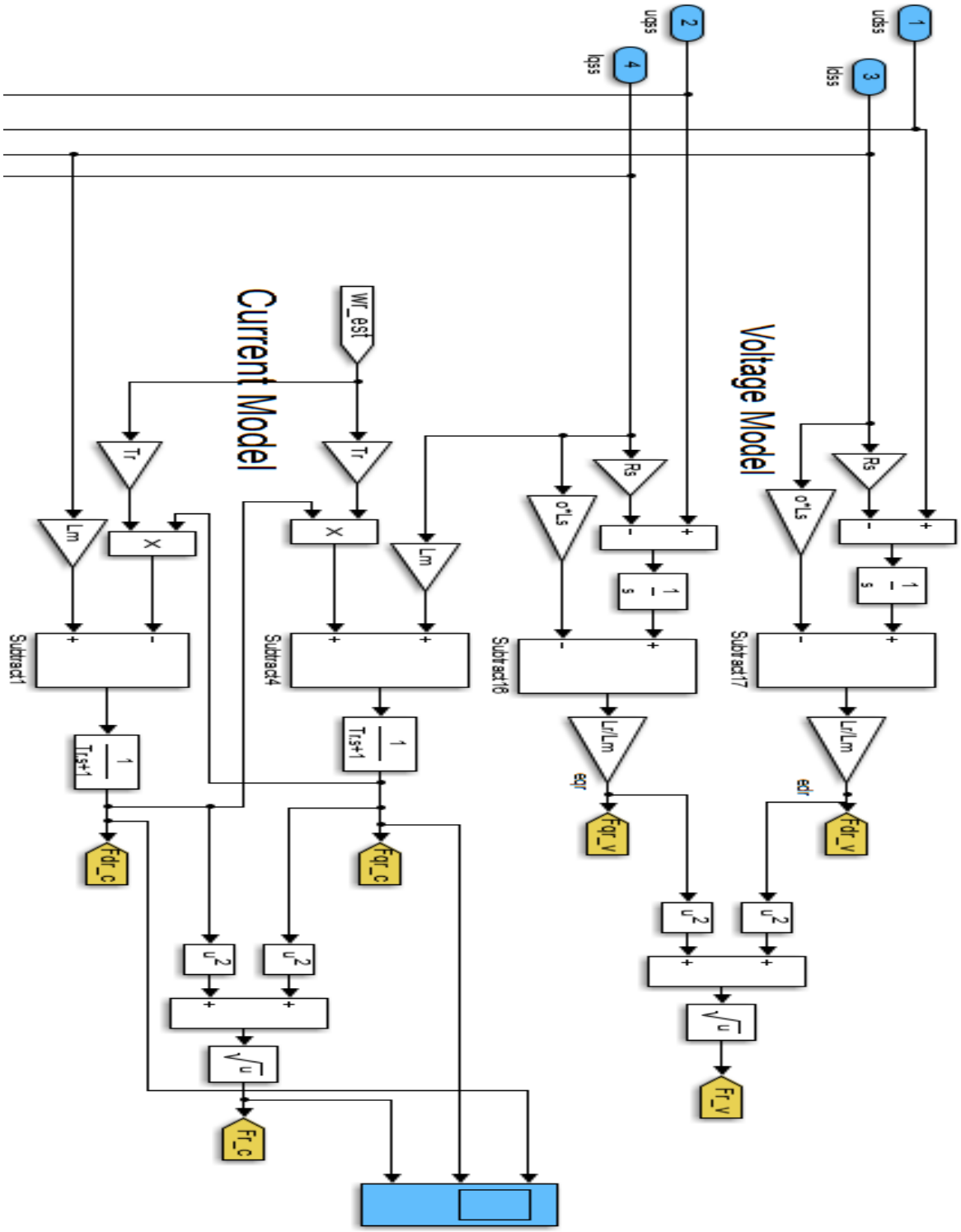
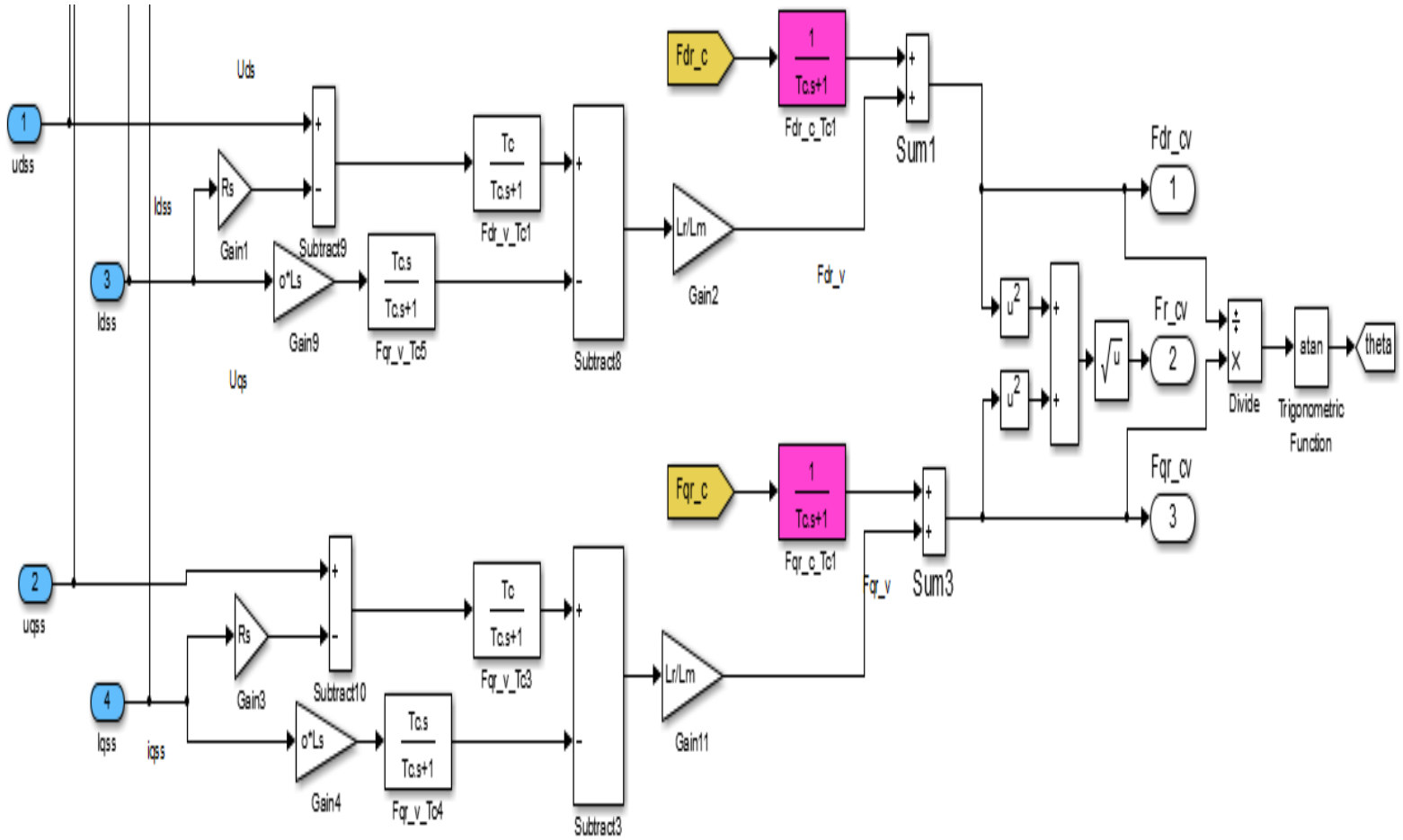

Voltage Model
Current Model
udss
uqss
idss
iqss
Rs
σ*Ls
Lr/Lm
Subtract17
Subtract16
Subtract4
Subtract1
Lm
Tr
Wr_est
Fdr_v
Fqr_v
Fdr_c
Fqr_c
Fr_v
Fr_c

Fig. A-0-3 Current Model(CM)+Voltage Model(VM)


1
uds
2
uqs
3
ids
4
iqs
Uds
Uqs
ids
iqs
Rs
Gain1
Rs
Gain3
Subtract9
Subtract10
σ*Ls
Gain9
σ*Ls
Gain4
Tc.s
Tc.s+1
Fqr_v_Tc5
Fqr_v_Tc4
Tc
Tc.s+1
Fdr_v_Tc1
Fqr_v_Tc3
Subtract8
Subtract3
Fdr_c
Fqr_c
1
Tc.s+1
Fdr_c_Tc1
Fqr_c_Tc1
Lr/Lm
Gain2
Lr/Lm
Gain11
Fdr_v
Fqr_v
Sum1
Sum3
u²
u²
√u
Fdr_cv
Fr_cv
Fqr_cv
1
2
3
÷
X
Divide
atan
Trigonometric Function
theta

observer

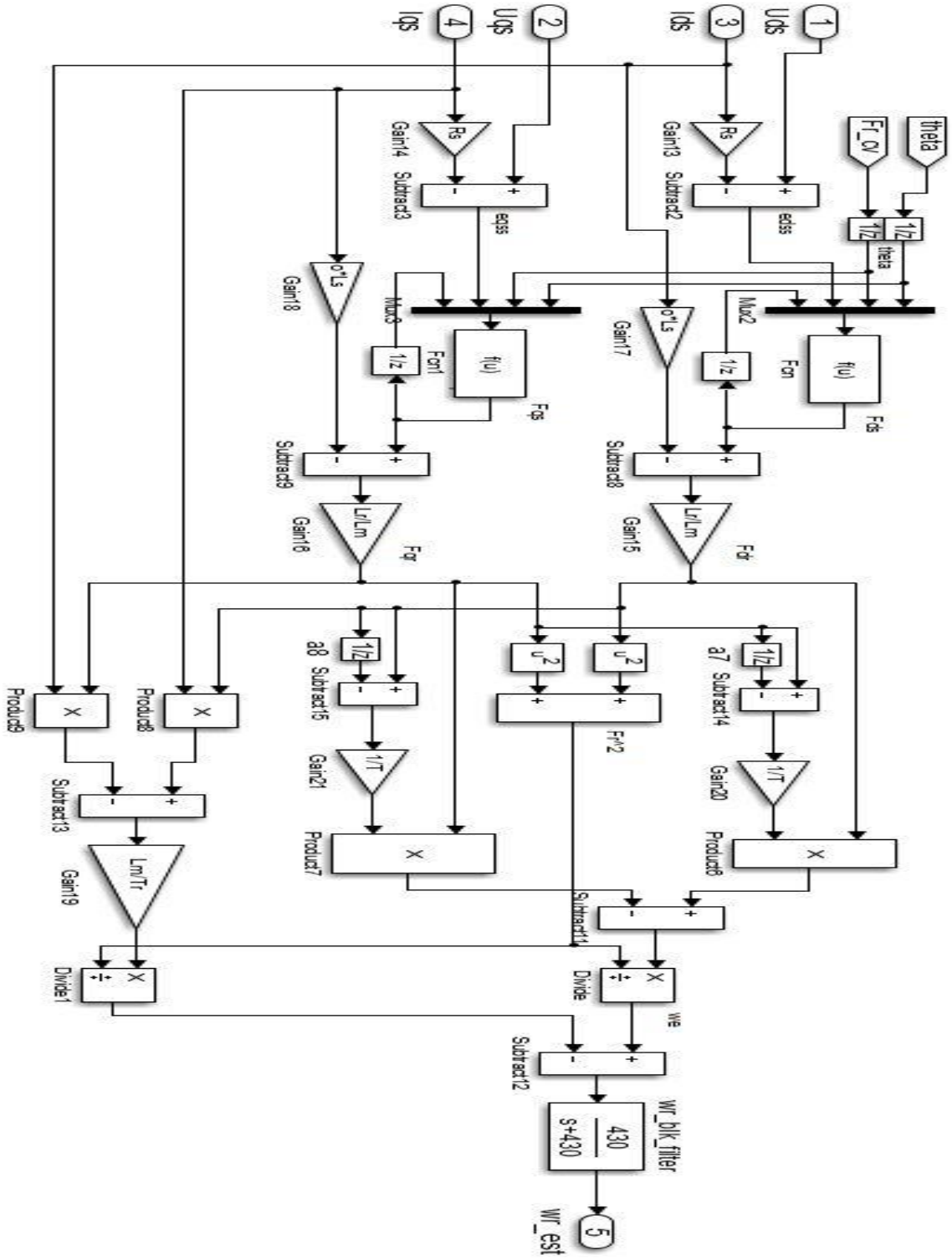

1
Uds
2
Uqs
3
Ids
4
Iqs
Fr_cv
theta
1/z
theta
Rs
Gain13
Subtract2
eqss
Rs
Gain14
Subtract3
eqss
Mux2
Mux3
f(u)
Fcn
Fds
Fcn1
1/z
Lo*Ls
Gain17
Lo*Ls
Gain18
Subtract8
Subtract9
Lr/Lm
Gain15
Lr/Lm
Gain16
Fdr
Fqr
u^2
u^2
Fr^2
a7
Subtract14
Gain20
1/T
Product6
a8
Subtract15
Gain21
1/T
Product7
Product8
Product9
Subtract13
Lm/Tr
Gain19
Subtract11
Divide
Divide1
we
Subtract12
wr_blk_filter
430
s+430
5
Wr_est

Fig. A-0-5 Sensorless Speed Observer

5.5 Appendix B: DSP program code

```
/* ======================*/
#include "DSP281x_Device.h"
#include "DSP281x_Examples.h"
#include "IQmathLib.h"
#include "ACIM_FOC.h"
#include "parameter_60Hz.h"
#include "build.h"
#include <math.h>

#include "FPGA_csr.h"
#include "Display.h"

void Delay_mili_second (int delay);
void init_fpga(void);
void Enable_Inv(void);
void Disable_Inv(void);
void Close_Relay(void);
void Open_Relay(void);
void Delay_mili_second (int delay);
void Delay_macro_second (int delay);
void InitGpio(void);
void Protection_PLL (void);
void InitXintfClocks(void);
void InitPeripherals(void);
void InitEVA(void);
void InitEVB(void);
void Init_variable(void);
void AD_bias16(void);
void get_adc_V(void);
void get_theta(void);

// Prototype statements for functions found within this file.
interrupt void inv_isr(void);
interrupt void adc_isr(void);
interrupt void pdpinta_isr(void);

// Global variables used in this system
float32 VdTesting = 0.4;      // Vd testing (pu) 0.25
float32 VqTesting = 0.35;     // Vq testing (pu) 0.35
float32 IdRef = 0.4;          // Id reference (pu) 0.2
0.5
float32 IqRef = 0.2;          // Iq reference (pu)
0.05
float32 SpeedRef = 0.45;      // Speed reference
(pu) 0.5

Uint16 Tsw=16672;//PWM Switching period in
counter format; T1PR;up/down==>switching freq
fpwm=150e6/((16672)*2*1*1)=4498.56Hz
#define Sw_freq 8997.1 //sampling frequency
150e6/((16672>>4)*2*1*1)=8997.7*8
float32 Tsampling = 2.2229E-4; //T=Tsampling
period in seconds;0.001/ISR_FREQ=50us;
isr_freq=20

Uint16 IsrTicker = 0;
Uint16 BackTicker = 0;

_iq Ualpha_max=_IQ(0);
_iq Ialpha_max=_IQ(0);
_iq Ibeta_max=_IQ(0);
_iq rg1max=_IQ(0);
int16 Isr_num=0;
```

```
int16 CurClosed=1;    //0: open loop;    1: closed
loop
int16 Graph_update=1;
int16 PwmDacCh1 = 0;
int16 PwmDacCh2 = 0;
int16 PwmDacCh3 = 0;
int16 lsw = 0;    // 0: rg1.Out; 1:fe1.ThetaFlux; 2:
cm1.Theta;
int16 flag_Kp=0,flag_Ki=0;
volatile Uint16 EnableFlag = FALSE;
//***************************************
/*Control flags*/
Uint16 FUNCTION, //1:Turbine simulator
2:Inertia test    3:Performance test 4:Speed control
        CONTACTOR, /*0-open, 1-close*/
        INTERRUPT, /*0-t1ufint stopped,
1-Rectifier running, 2-Inverter*/
//      FAULT,      /*0-no fault 1- fault
detected*/
        PLL,        /*0-PLL not locked
1-locked*/
        Fault_info_A, /*Result of fault detection
from left (A) side*/
        Fault_info_B, /*Result of fault detection
from left (B) side*/
        SCREEN;     /*0-system status screen0
1-status screen 1 */
Uint16 test_number=0;
/*PLL */
Uint16 Theta_con, Theta_err;    //phase and error
info in counter format
Uint16 nmax;
Uint32 Theta_pu; //line voltage angle in per unit
value
_iq28 Theta_line;       //line voltage angle in radian
value (0-2pi)
_iq28 Theta_phase; //phase voltage angle in radian
value
_iq28 cos_theta, sin_theta;
// Flags
Uint16 SpeedLoopPrescaler = 10;         // Speed
loop prescaler
Uint16 SpeedLoopCount = 1;              // Speed
loop counter
Uint16 FAULT=0;
// ADC
#define AVE_SAMPLE      8
//AVE_SAMPLE=2^AVE_SHFT
#define AVE_SHFT        3
#define AVE_SAMPLE_DC 16
//AVE_SAMPLE_DC=2^AVE_SHFT_DC
#define AVE_SHFT_DC     4
#define ADC_usDELAY     8000
#define ADC_usDELAY2    40
Uint16 Sample_ch0[AVE_SAMPLE],
Sample_ch1[AVE_SAMPLE],
        Sample_ch2[AVE_SAMPLE],
Sample_ch3[AVE_SAMPLE],
        Sample_ch4[AVE_SAMPLE],
Sample_ch5[AVE_SAMPLE_DC],
        Sample_ch6[AVE_SAMPLE_DC],
Sample_ch7[AVE_SAMPLE],
        Sample_ch8[AVE_SAMPLE_DC],
Sample_ch9[AVE_SAMPLE],
```

```
        Sample_ch10[AVE_SAMPLE],
Sample_ch11[AVE_SAMPLE];
    // Sample_ch12[AVE_SAMPLE],
Sample_ch13[AVE_SAMPLE],
    // Sample_ch14[AVE_SAMPLE],
Sample_ch15[AVE_SAMPLE];

Uint16 ConversionCount=0;
Uint16 ConversionCount1=0;
int32 AD_BIAS0, AD_BIAS1, AD_BIAS2,
AD_BIAS3, AD_BIAS4,
        AD_BIAS5, AD_BIAS6, AD_BIAS7,
AD_BIAS8, AD_BIAS9,
        AD_BIAS10, AD_BIAS11, AD_BIAS12,
AD_BIAS13, AD_BIAS14,
        AD_BIAS15;

/*Outer loop DC voltage control*/
int32 Vab, Vbc, Vca; //(_iq18 format)
int32 Vdc_P,Vdc_N; //(_iq18 format)
_iq18 Vdc_ref,Vdc,Vdc0,Vdc_L,Vdc_L0,
Vdc_err,Vdc_err1;
int32 Euv, Evw, Ewu; //(_iq18 format)
int32 Ia_g, Ib_g, Ic_g; //(_iq18 format)
int32 Ia, Ib, Ic; //(_iq18 format)
int32 Ia1, Ic1, Ia2, Ic2; //(_iq18 format)
//int32 Id_max=0;

/*Low-pass filter*/
_iq28
LPFa,LPFb,LPFa_DC,LPFb_DC,LPFa_speed,LPF
b_speed,LPFa_torque,LPFb_torque;

/*Inverter*/
//_iq18 D_inv,D_inv1;
_iq18 Iu,Iv,Io,Iu_ref, Iu_err, Iu_err1,I_test;
_iq18 Iw,Iw0,Iw_L,Iw_L0,Iu,Iu0,Iu_L,Iu_L0;
//_iq28 kp_Iu, ki_Iu;    //PI controller

#pragma
DATA_SECTION(graph_data,"graph_data");
#pragma
DATA_SECTION(graph_data1,"graph_data1");
//#pragma
DATA_SECTION(graph_data2,"graph_data2");
//#pragma
DATA_SECTION(graph_data3,"graph_data3");

/*Graph of waveform*/
int32    graph_data[4096];    //
int32    graph_data1[4096]; //
//int32    graph_data2[2048]; //
//int32    graph_data3[2048]; //
Uint16 graph_index=0, lsw_prestate=0;
Uint16 i=0;
Uint16 idCount=0, iqCount=0;
//*****************************************
// Init a ramp controller
RMPCNTL rc1 = RMPCNTL_DEFAULTS;
// Init a ramp machine to simulate an Angle
RAMPGEN rg1 = RAMPGEN_DEFAULTS;

// Init transform objects
CLARKE clarke1 = CLARKE_DEFAULTS;
PARK park1 = PARK_DEFAULTS;
IPARK ipark1 = IPARK_DEFAULTS;
```

```
// Init d,q,speed PID regulators
PI_CONTROLLER pid1_id =
PI_CONTROLLER_DEFAULTS;
PI_CONTROLLER pid1_iq =
PI_CONTROLLER_DEFAULTS;
PI_CONTROLLER pid1_spd =
PI_CONTROLLER_DEFAULTS;

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

// Init a Space Vector PWM modulator
SVGENDQ svgen_dq1 =
SVGENDQ_DEFAULTS;

// Init    current/dc-bus voltage measurement driver
ILEG2DCBUSMEAS ilg2_vdc1 =
ILEG2DCBUSMEAS_DEFAULTS;

// Init model objects
CURMOD cm1 = CURMOD_DEFAULTS;
// Init a current model constant object
CURMOD_CONST cm1_const =
CURMOD_CONST_DEFAULTS;
// Init a Voltage phase, alpha and beta interface
PHASEVOLTAGE volt1 =
PHASEVOLTAGE_DEFAULTS;

// Instance a induction model constant object
ACI_CONST aci1_const =
ACI_CONST_DEFAULTS;
ACIFE_CONST fe1_const =
ACIFE_CONST_DEFAULTS;
ACISE_CONST se1_const =
ACISE_CONST_DEFAULTS;

// Instance a induction model object,rotor flux and
speed estimations
ACI aci1 = ACI_DEFAULTS;
ACIFE fe1 = ACIFE_DEFAULTS;
ACISE se1 = ACISE_DEFAULTS;

// Init a speed calculator based on capture
SPEED_MEAS_CAP speed1 =
SPEED_MEAS_CAP_DEFAULTS;

void main(void)
{
// Initialize PLL, WatchDog, Clocks to
default;DSP281x_SysCtrl.c.
    InitSysCtrl();

// Step 2. Initialize the Xintf clocks
    InitXintfClocks();
// Step 3. Initalize GPIO:
    InitGpio();

// Step 4. Clear all interrupts and initialize PIE
vector table:
    DINT; // Disable CPU interrupts
    InitPieCtrl(); // Initialize the PIE control
registers to their default state.
```

```
                        // The default state is all PIE
interrupts disabled and flags
                        // are cleared.
    IER = 0x0000; // Disable CPU interrupts
    IFR = 0x0000; // Clear all CPU interrupt flags

// HISPCP prescale set to default values
      EALLOW;
      SysCtrlRegs.HISPCP.all = 0x0000;        //
SYSCLKOUT/1
      EDIS;

//PIE vector table points to
DSP281x_DefaultIsr.c:DSP281x_PieVect.c.
      InitPieVectTable();

// Enable Underflow interrupt bits for GP timer 1
      EvaRegs.EVAIMRA.bit.T1UFINT = 1;
      EvaRegs.EVAIFRA.bit.T1UFINT = 1;
// Reassign T1UFINT to a different ISR then the
shell routine in DSP281x_DefaultIsr.c.
      EALLOW;
      PieVectTable.PDPINTA = &pdpinta_isr;
      PieVectTable.T1UFINT = &inv_isr;
//T1PR=Tsw
      PieVectTable.ADCINT = &adc_isr;
//T4PR=Tsw>>4; T4TOADC=1;T4CON start with
T3
      EDIS;

// Step 5. Initialize all the Device Peripherals:
    InitPeripherals();

//      PieCtrlRegs.PIEIER4.bit.INTx5 = 1; //
Enable t3cint in the PIE: Group 4 interrupt 5
      PieCtrlRegs.PIEIER2.bit.INTx6 = 1; //enable
T1UFINT
      PieCtrlRegs.PIEIER1.bit.INTx1 = 1; //enable
PDPINTA
      PieCtrlRegs.PIEIER1.bit.INTx6 = 1; //Enable
ADCINT

      IER |= M_INT1;
      IER |= M_INT2;

// Initialize PWM module;
// t1con=0x8840==>free run, up/down,Perscaler*1,
count period=2*T1PR;
      pwm1.PeriodMax =
SYS_FREQ*1000000*Tsampling/2;    // ISR
period=Tsampling*1
      pwm1.init(&pwm1);

// Initialize RAMPGEN module
      rg1.StepAngleMax =
_IQ(BASE_FREQ*Tsampling);

// Initialize the ACI constant module
      aci1_const.Rs = RS;
      aci1_const.Rr = RR;
      aci1_const.Ls = LS;
      aci1_const.Lr = LR;
      aci1_const.Lm = LM;
      aci1_const.p = P;
      aci1_const.B = BB;
```

```
        aci1_const.J = JJ;
        aci1_const.Ib = BASE_CURRENT;
        aci1_const.Vb = BASE_VOLTAGE;
        aci1_const.Wb = 2*PI*BASE_FREQ;
        aci1_const.Tb = BASE_TORQUE;
        aci1_const.Lb = BASE_FLUX;
        aci1_const.Ts = Tsampling;
        aci1_const.calc(&aci1_const);

    // Initialize the ACI module
        aci1.K1 = _IQ(aci1_const.K1);
        aci1.K2 = _IQ(aci1_const.K2);
        aci1.K3 = _IQ(aci1_const.K3);
        aci1.K4 = _IQ(aci1_const.K4);
        aci1.K5 = _IQ(aci1_const.K5);
        aci1.K6 = _IQ(aci1_const.K6);
        aci1.K7 = _IQ(aci1_const.K7);
        aci1.K8 = _IQ(aci1_const.K8);
        aci1.K9 = _IQ(aci1_const.K9);
        aci1.K10 = _IQ(aci1_const.K10);
        aci1.BaseRpm = 120*BASE_FREQ/P;
        aci1.LoadTorque =
    _IQ(TL/BASE_TORQUE);

    // Initialize the ACI_FE constant module
        fe1_const.Rs = RS;
        fe1_const.Rr = RR;
        fe1_const.Ls = LS;
        fe1_const.Lr = LR;
        fe1_const.Lm = LM;
        fe1_const.Ib = BASE_CURRENT;
        fe1_const.Vb = BASE_VOLTAGE;
        fe1_const.Ts = Tsampling;
        ACIFE_CONST_MACRO(fe1_const);

    // Initialize the ACI_FE module
        fe1.K1 = _IQ(fe1_const.K1);
        fe1.K2 = _IQ(fe1_const.K2);
        fe1.K3 = _IQ(fe1_const.K3);
        fe1.K4 = _IQ(fe1_const.K4);
        fe1.K5 = _IQ(fe1_const.K5);
        fe1.K6 = _IQ(fe1_const.K6);
        fe1.K7 = _IQ(fe1_const.K7);
        fe1.K8 = _IQ(fe1_const.K8);
        fe1.Kp = _IQ(0.055);    //2.8
        fe1.Ki = _IQ(0.002);    //Tsampling/0.45

    // Initialize the ACI_SE constant module
        se1_const.Rr = RR;
        se1_const.Lr = LR;
        se1_const.fb = BASE_FREQ;
        se1_const.fc = 200;
        se1_const.Ts = Tsampling;
        ACISE_CONST_MACRO(se1_const);

    // Initialize the ACI_SE module
        se1.K1 = _IQ(se1_const.K1);
        se1.K2 = _IQ21(se1_const.K2);
        se1.K3 = _IQ(se1_const.K3);
        se1.K4 = _IQ(se1_const.K4);
        se1.BaseRpm = 1800;
    //12000;//120*BASE_FREQ/P;

    // Initialize the PID module for Id
        pid1_id.Kp = _IQ(0.75);    //1;0.75
        pid1_id.Ki = _IQ(0.0018);     //0.0018
```

```
        pid1_id.Umax = _IQ(0.95); //0.95
        pid1_id.Umin = _IQ(-0.95); //-0.95

// Initialize the PID module for Iq
        pid1_iq.Kp = _IQ(0.75); //1
        pid1_iq.Ki = _IQ(0.0018);     //Tsampling/0.04
        pid1_iq.Umax = _IQ(0.95);    //0.95
        pid1_iq.Umin = _IQ(-0.95);    //-0.95

// Initialize the PID_REG3 module for speed
        pid1_spd.Kp = _IQ(0.2);        //0.02
        pid1_spd.Ki = _IQ(0.1);    //0.1
        pid1_spd.Umax = _IQ(0.95);    //1
        pid1_spd.Umin = _IQ(-0.95);      //-1

//      for (i=0; i<2048; i++)
//      {graph_data[i]=0x0;
//        graph_data1[i]=0x0;
//        graph_data2[i]=0x0;
//        graph_data3[i]=0x0;
//      }

  if (FAULT ==0)

      {     AD_bias16();    //call AD_bias(#) to
compute the average bias for each channel

     AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1 = 1;
//enable SEQ1 interrupt (every EOS)
            Init_variable();
            InitGpio();
            init_fpga();
            if (BUILDLEVEL!=LEVEL1)
//virtual machine test
            {Close_Relay();}

            Enable_Inv();

            EINT;
            ERTM;
      }
// IDLE loop. Just sit and loop forever:
      for(;;) BackTicker++;
}
/****************************************
/*    NAME:            inv_isr()          */
/*    RETURNS:         void               */
/*    DESCRIPTION: Interrupt for inverter side
calculation */
/****************************************/
interrupt void inv_isr(void)
{ // Verifying the ISR
      IsrTicker++;

// ****** LEVEL1: virtual machine closed
Current_Loop + Speed_Loop test
**********working
// set pid1_spd.Kp = _IQ(0.02);       //0.02
#if (BUILDLEVEL==LEVEL1)     //

//    PARK module and call the park transformation
calculation function.
      park1.Alpha = aci1.Ialpha;    //in pu
      park1.Beta = aci1.Ibeta;
```

```
        park1.Angle = fe1.ThetaFlux;
        park1.Sine = _IQsinPU(park1.Angle);
        park1.Cosine = _IQcosPU(park1.Angle);
        PARK_MACRO(park1);

//    Connect inputs of the PID module and call the
PID speed controller calculation function.
        pid1_spd.Ref = _IQ(SpeedRef);
        pid1_spd.Fbk = se1.WrHat;
        PI_MACRO(pid1_spd);

//    PID_REG3 module and call the PID ID
controller calculation function.
//        if (idCount<=256)    //16384
//              {idCount++;
//         pid1_id.Ref = 0;}
//      else
//         {pid1_id.Ref = _IQ(IdRef);}    // _IQ(IqRef)
         pid1_id.Ref = _IQ(IdRef);;
         pid1_id.Fbk = park1.Ds;
         PI_MACRO(pid1_id);    //==>pid1_id.Out


//    PID_REG3 module and call the PID IQ
controller calculation function.
//         if (iqCount<=512)    //16384
//              {iqCount++;
//          pid1_iq.Ref = 0;}
//      else
//          {pid1_iq.Ref = _IQ(IqRef);}    //
_IQ(IqRef)

          pid1_iq.Ref = pid1_spd.Out;//_IQ(IqRef);
          pid1_iq.Fbk = park1.Qs;
          PI_MACRO(pid1_iq);    //==>pid1_id.Out

// INV_PARK module and call the inverse park
transformation calculation function.
        ipark1.Ds = pid1_id.Out;
        ipark1.Qs = pid1_iq.Out;
        ipark1.Angle = fe1.ThetaFlux;
        ipark1.Sine = _IQsinPU(ipark1.Angle);
        ipark1.Cosine = _IQcosPU(ipark1.Angle);
        IPARK_MACRO(ipark1);    //==>ipark1.Ds;
ipark1.Qs

// Connect inputs of the ACI module and call the
induction motor model calculation function.
        aci1.Ualpha = ipark1.Alpha;
        aci1.Ubeta = ipark1.Beta;
        aci1.calc(&aci1);

// ACI Flux angle estimator interface
        fe1.UDsS = aci1.Ualpha;
        fe1.UQsS = aci1.Ubeta;
        fe1.IDsS = aci1.Ialpha;
        fe1.IQsS = aci1.Ibeta;
        ACIFE_MACRO(fe1);

//    ACI Speed Estimator interface to generate
Estimated Speed
        se1.IDsS = aci1.Ialpha;
        se1.IQsS = aci1.Ibeta;
        se1.PsiDrS = fe1.PsiDrS;
        se1.PsiQrS = fe1.PsiQrS;
        se1.ThetaFlux = fe1.ThetaFlux;
```

```
        ACISE_MACRO(se1);
        //      speed_pu1 =se1.WrHat;    //speed in pu
        //      speed_rpm1=se1.WrHatRpm;    //speed in
rpm

//    SVGEN_DQ module and call the space-vector
gen.calculation function.
        svgen_dq1.Ualpha = ipark1.Alpha;    //id
        svgen_dq1.Ubeta = ipark1.Beta;    //iq
        SVGEN_MACRO(svgen_dq1);        //==>Ta,
Tb, Tc

//    PWM_DRV module and call the PWM signal
generation update function.
        pwm1.MfuncC1 =
(int16)_IQtoIQ15(svgen_dq1.Ta); // MfuncC1 is in
Q15
        pwm1.MfuncC2 =
(int16)_IQtoIQ15(svgen_dq1.Tb); // MfuncC2 is in
Q15
        pwm1.MfuncC3 =
(int16)_IQtoIQ15(svgen_dq1.Tc); // MfuncC3 is in
Q15
        pwm1.update(&pwm1);

//    Volt Macro to reconstruct Vphase_ABC and
Valpha, Vbeta;
        volt1.MfuncV1 = svgen_dq1.Ta;    //iq format
        volt1.MfuncV2 = svgen_dq1.Tb;
        volt1.MfuncV3 = svgen_dq1.Tc;
        volt1.DcBusVolt =
_IQdiv(_IQ18toIQ((int32)Vdc),_IQ(BASE_VOLT
AGE));
        VOLT_MACRO(volt1);

//    Waveforms display
        graph_data[graph_index] = fe1.ThetaFlux;
//pid1_id.Ref;//volt1.Valpha;//ipark1.Alpha;//clarke
1.As;//pid1_id.up;//park1.Ds;//
        graph_data1[graph_index]
=se1.WrHat;//pid1_id.Fbk;//volt1.Vbeta;//ipark1.Be
ta;//clarke1.Bs;//pid1_iq.up;//park1.Qs;//
//      graph_data2[graph_index] =
pid1_id.Fbk;//ipark1.Ds;//clarke1.Alpha;
//pid1_id.Out;//clarke1.As;//
//      graph_data3[graph_index] =
pid1_iq.Fbk;//ipark1.Qs;//clarke1.Beta;
//pid1_iq.Out;//clarke1.Bs;//

        if (graph_index==4096 &&
Graph_update==1)
                {graph_index=0;
        idCount=0;iqCount=0;}
            else if(graph_index<4096)
                {graph_index++;}
#endif // (BUILDLEVEL==LEVEL1
Current_Loop+Speed_Loop closed test)
// *****LEVEL2: Actual IM closed Current_Loop
+ Speed_Loop test *****working
// set pid1_spd.Kp = _IQ(0.2);    Vdc=150V lsw=1
after debug-->real time(red stick)
first-->run-->continuously refresh
//speed and current loop all working;
speedRef*1800=estimated=measured
        #if (BUILDLEVEL==LEVEL2)    //
        if (lsw==1)
```

```
        {if
(EvbRegs.COMCONB.bit.FCOMPOE==0)
{Enable_Inv();}

        //  RMP module and call the Ramp
control calculation function.
        rc1.TargetValue = _IQ(SpeedRef);
//rc1.calc() modified
        RC_MACRO(rc1);  //rc1:var name; calc:
field name, call rmp_cntl_calc()
        //  RAMP GEN module and call the
Ramp machine calculation function.
        rg1.Freq = rc1.SetpointValue;
//SetpointValue add 0.00005 each step
        RG_MACRO(rg1);
//v->StepAngleMax*v->Freq;  StepAngleMax =
BASE_FREQ*Tsampling;
        }
        else  {rc1.SetpointValue=0;  rg1.Out=
0; Disable_Inv();}

//   Call the ILEG2_VDC read function.
ilg2_vdc1.read(&ilg2_vdc1);
//   get_theta();
     get_adc_V();
     Vdc=_IQ18(150);
//  CLARKE module and call the clarke
transformation calculation function.  -1~+1
     clarke1.As =
_IQdiv(_IQ18toIQ((int32)Iu),_IQ(BASE_CURRE
NT));  //Iu _iq18 format BASE_CURRENT =
7.778
     clarke1.Bs =
_IQdiv(_IQ18toIQ((int32)Iw),_IQ(BASE_CURRE
NT));  //Iw _iq18 format change to _iq20 format
     CLARKE_MACRO(clarke1);  //Iu_L, Iw_L
are real values

//  PARK module and call the park transformation
calculation function.
   park1.Alpha = clarke1.Alpha;
   park1.Beta = clarke1.Beta;

   if (lsw == 0) {park1.Angle = rg1.Out;}
     else if (lsw == 1) {park1.Angle =
fe1.ThetaFlux;}

   park1.Sine = _IQsinPU(park1.Angle);
   park1.Cosine = _IQcosPU(park1.Angle);
   PARK_MACRO(park1);

//  Connect inputs of the PID module and call the
PID speed controller calculation function.
   if (SpeedLoopCount==SpeedLoopPrescaler)
    {
      pid1_spd.Ref = rc1.SetpointValue;
      pid1_spd.Fbk = se1.WrHat;
      PI_MACRO(pid1_spd);
      SpeedLoopCount=1;    }
   else SpeedLoopCount++;

    if(lsw==0)    {pid1_spd.ui=0;
pid1_spd.i1=0;}
```

```
   //    PID_REG3 module and call the PID ID
controller calculation function.
     pid1_id.Ref = _IQ(IdRef);;
     pid1_id.Fbk = park1.Ds;
     PI_MACRO(pid1_id);    //==>pid1_id.Out


//    PID_REG3 module and call the PID IQ
controller calculation function.
      if (lsw == 0)    {pid1_iq.Ref = _IQ(IqRef);}
     else if (lsw == 1) {pid1_iq.Ref = pid1_spd.Out;}
//;connect to aci_se; speed closed loop

     pid1_iq.Fbk = park1.Qs;
     PI_MACRO(pid1_iq);    //==>pid1_id.Out

// INV_PARK module and call the inverse park
transformation calculation function.
     ipark1.Ds = pid1_id.Out;
     ipark1.Qs = pid1_iq.Out;
     ipark1.Sine = park1.Sine;
     ipark1.Cosine = park1.Cosine;
     IPARK_MACRO(ipark1);    //==>ipark1.Ds;
ipark1.Qs

// Connect inputs of the ACI module and call the
induction motor model calculation function.
//          aci1.Ualpha = ipark1.Alpha;
//          aci1.Ubeta = ipark1.Beta;
//          aci1.calc(&aci1);
//    SVGEN_DQ module and call the space-vector
gen.calculation function.
     svgen_dq1.Ualpha = ipark1.Alpha;    //id
     svgen_dq1.Ubeta = ipark1.Beta;    //iq
     SVGEN_MACRO(svgen_dq1);          //==>Ta,
Tb, Tc

//    PWM_DRV module and call the PWM signal
generation update function.
     pwm1.MfuncC1 =
(int16)_IQtoIQ15(svgen_dq1.Ta); // MfuncC1 is in
Q15
     pwm1.MfuncC2 =
(int16)_IQtoIQ15(svgen_dq1.Tb); // MfuncC2 is in
Q15
     pwm1.MfuncC3 =
(int16)_IQtoIQ15(svgen_dq1.Tc); // MfuncC3 is in
Q15
     pwm1.update(&pwm1);

//      Volt Macro to reconstruct Vphase_ABC and
Valpha, Vbeta;
     volt1.MfuncV1 = svgen_dq1.Ta;
     volt1.MfuncV2 = svgen_dq1.Tb;
     volt1.MfuncV3 = svgen_dq1.Tc;
     volt1.DcBusVolt =
_IQdiv(_IQ18toIQ((int32)Vdc),_IQ(BASE_VOLT
AGE));
     VOLT_MACRO(volt1);

//COMPARE CM1.THETA AND
FE1.FLUXTHETA CURRENT MODE NEED
SPEED
//ACI3-3 HAS SPEED CAPTURE, ACI3-4 USE
VIRTUAL MACHINE
```

```
//PAUSE VERIFY VOLTAGE AND CURRENT PU VALUE; CHECK Rotor speed

// ACI Flux angle estimator interface
    fe1.IDsS = clarke1.Alpha;    //output from CLARKE in pu
    fe1.IQsS = clarke1.Beta;
    fe1.UDsS = volt1.Valpha;    //output from VOLT in pu
    fe1.UQsS = volt1.Vbeta;
    ACIFE_MACRO(fe1);

//    ACI Speed Estimator interface to generate Estimated Speed
    se1.IDsS = clarke1.Alpha;    // in pu
    se1.IQsS = clarke1.Beta;      // in pu
    se1.ThetaFlux = fe1.ThetaFlux;
    se1.PsiDrS = fe1.PsiDrS;
    se1.PsiQrS = fe1.PsiQrS;
    ACISE_MACRO(se1);
    //       speed_pu1 =se1.WrHat;    //speed in pu
    //       speed_rpm1=se1.WrHatRpm;    //speed in rpm

//    Waveforms display
       graph_data[graph_index] = se1.WrHat;//fe1.ThetaFlux;
//pid1_id.Ref;//volt1.Valpha;//ipark1.Alpha;//clarke1.As;//pid1_id.up;//park1.Ds;//
       graph_data1[graph_index] =se1.WrHatRpm;//pid1_id.Fbk;//volt1.Vbeta;//ipark1.Beta;//clarke1.Bs;//pid1_iq.up;//park1.Qs;//
//      if (Isr_num<1)    {Isr_num++;}    else {Isr_num=0;}
//    if (lsw ==1) {Graph_update=1;lsw_prestate=1;} else {Graph_update=0;}
  //    if (lsw_prestate==1 && lsw==0) {graph_index=0; lsw_prestate=0;}

       graph_data2[graph_index] = volt1.Valpha;//_IQmpy(_IQ(volt1.Valpha),_IQ(BASE_VOLTAGE));
       graph_data3[graph_index] = volt1.VphaseA; //_IQmpy(_IQ(volt1.VphaseA),_IQ(BASE_VOLTAGE));

       if (graph_index==4096 && Graph_update==1)
                    {graph_index=0;
       idCount=0;iqCount=0;}
            else if ((graph_index<4096)&& (Isr_num==0))
                    {graph_index++;}

#endif // (BUILDLEVEL==LEVEL2 Current Loop+Speed loop )

// ****** LEVEL3: AC IM coupled with DC motor; Constant IdRef=0.4;*********
//Actual machine closed Current_Loop + Speed_Loop test working (level 2)
//pid1_iq.Ref = pid1_spd.Out;==>pid1_iq.Ref =0; lsw=0 to run;
       #if (BUILDLEVEL==LEVEL3)    //
            if (lsw==1)
```

```
            {if
(EvbRegs.COMCONB.bit.FCOMPOE==0)
{Enable_Inv();}

            //  RMP module and call the Ramp
control calculation function.
            rc1.TargetValue = _IQ(SpeedRef);
//rc1.calc() modified
            RC_MACRO(rc1);  //rc1:var name;
calc: field name, call rmp_cntl_calc()
            //  RAMP GEN module and call the
Ramp machine calculation function.
            rg1.Freq = rc1.SetpointValue;
//SetpointValue add 0.00005 each step
            RG_MACRO(rg1);
//v->StepAngleMax*v->Freq;  StepAngleMax =
BASE_FREQ*Tsampling;
            }
            else  {rc1.SetpointValue=0;
rg1.Out= 0; }//Disable_Inv();

    //   Call the ILEG2_VDC read function.
ilg2_vdc1.read(&ilg2_vdc1);
    //   get_theta();
         get_adc_V();
         Vdc=_IQ18(100);
    //  CLARKE module and call the clarke
transformation calculation function.  -1~+1
         clarke1.As =
_IQdiv(_IQ18toIQ((int32)Iu),_IQ(BASE_CURRE
NT));  //Iu _iq18 format BASE_CURRENT =
7.778
         clarke1.Bs =
_IQdiv(_IQ18toIQ((int32)Iw),_IQ(BASE_CURRE
NT));  //Iw _iq18 format change to _iq20 format
         CLARKE_MACRO(clarke1);  //Iu_L,
Iw_L are real values

    //  PARK module and call the park
transformation calculation function.
         park1.Alpha = clarke1.Alpha;
         park1.Beta = clarke1.Beta;

         if (lsw == 0) {park1.Angle = rg1.Out;}
//lsw=0==>rg1.out=0
          else if (lsw == 1) {park1.Angle =
fe1.ThetaFlux;}

         park1.Sine = _IQsinPU(park1.Angle);
         park1.Cosine = _IQcosPU(park1.Angle);
         PARK_MACRO(park1);

    //  Connect inputs of the PID module and call
the PID speed controller calculation function.
         if
(SpeedLoopCount==SpeedLoopPrescaler)
          {
           pid1_spd.Ref = rc1.SetpointValue;
           pid1_spd.Fbk = se1.WrHat;
           PI_MACRO(pid1_spd);
           SpeedLoopCount=1;    }
         else SpeedLoopCount++;

          if(lsw==0)    {pid1_spd.ui=0;
pid1_spd.i1=0;}
```

```
        //  PID_REG3 module and call the PID ID controller calculation function.
            pid1_id.Ref = _IQ(IdRef);
            pid1_id.Fbk = park1.Ds;
            PI_MACRO(pid1_id);    //==>pid1_id.Out


        //  PID_REG3 module and call the PID IQ controller calculation function.
            if (lsw == 0)    {pid1_iq.Ref = 0;}//_IQ(IqRef);
            else if (lsw == 1) {pid1_iq.Ref = pid1_spd.Out;} //connect to aci_se; speed closed loop

            pid1_iq.Fbk = park1.Qs;
            PI_MACRO(pid1_iq);    //==>pid1_id.Out

        // INV_PARK module and call the inverse park transformation calculation function.
            ipark1.Ds = pid1_id.Out;
            ipark1.Qs = pid1_iq.Out;
            ipark1.Sine = park1.Sine;
            ipark1.Cosine = park1.Cosine;
            IPARK_MACRO(ipark1);
//==>ipark1.Ds; ipark1.Qs

        // Connect inputs of the ACI module and call the induction motor model calculation function.
        //          aci1.Ualpha = ipark1.Alpha;
        //          aci1.Ubeta = ipark1.Beta;
        //          aci1.calc(&aci1);

        //  SVGEN_DQ module and call the space-vector gen.calculation function.
            svgen_dq1.Ualpha = ipark1.Alpha;    //id
            svgen_dq1.Ubeta = ipark1.Beta;    //iq
            SVGEN_MACRO(svgen_dq1);
//==>Ta, Tb, Tc


        //  PWM_DRV module and call the PWM signal generation update function.
            pwm1.MfuncC1 = (int16)_IQtoIQ15(svgen_dq1.Ta); // MfuncC1 is in Q15
            pwm1.MfuncC2 = (int16)_IQtoIQ15(svgen_dq1.Tb); // MfuncC2 is in Q15
            pwm1.MfuncC3 = (int16)_IQtoIQ15(svgen_dq1.Tc); // MfuncC3 is in Q15
            pwm1.update(&pwm1);

        //      Volt Macro to reconstruct Vphase_ABC and Valpha, Vbeta;
            volt1.MfuncV1 = svgen_dq1.Ta;
            volt1.MfuncV2 = svgen_dq1.Tb;
            volt1.MfuncV3 = svgen_dq1.Tc;
            volt1.DcBusVolt = _IQdiv(_IQ18toIQ((int32)Vdc),_IQ(BASE_VOLTAGE));
            VOLT_MACRO(volt1);
        // ACI Flux angle estimator interface
            fe1.IDsS = clarke1.Alpha;    //output from CLARKE in pu
            fe1.IQsS = clarke1.Beta;
```

```
        fe1.UDsS = volt1.Valpha;    //output from
VOLT in pu
        fe1.UQsS = volt1.Vbeta;
        ACIFE_MACRO(fe1);

    //  ACI Speed Estimator interface to generate
Estimated Speed
        se1.IDsS = clarke1.Alpha;    // in pu
        se1.IQsS = clarke1.Beta;     // in pu
        se1.ThetaFlux = fe1.ThetaFlux;
        se1.PsiDrS = fe1.PsiDrS;
        se1.PsiQrS = fe1.PsiQrS;
        ACISE_MACRO(se1);
        //      speed_pu1 =se1.WrHat;    //speed in
pu
        //      speed_rpm1=se1.WrHatRpm;
//speed in rpm

    //  Waveforms display
          graph_data[graph_index] =
pid1_id.Ref;//fe1.ThetaFlux;
//pid1_id.Ref;//volt1.Valpha;//ipark1.Alpha;//clarke
1.As;//pid1_id.up;//park1.Ds;//
          graph_data1[graph_index]
=se1.WrHatRpm;//pid1_id.Fbk;//volt1.Vbeta;//ipar
k1.Beta;//clarke1.Bs;//pid1_iq.up;//park1.Qs;//

    //    if (Isr_num<1)    {Isr_num++;}    else
{Isr_num=0;}
    //    if (lsw ==1)
{Graph_update=1;lsw_prestate=1;}    else
{Graph_update=0;}
    //    if (lsw_prestate==1 && lsw==0)
{graph_index=0; lsw_prestate=0;}

//          graph_data2[graph_index] =
volt1.Valpha;//_IQmpy(_IQ(volt1.Valpha),_IQ(BA
SE_VOLTAGE));
//          graph_data3[graph_index] =
volt1.VphaseA;
//_IQmpy(_IQ(volt1.VphaseA),_IQ(BASE_VOLT
AGE));

          if (graph_index==4096 &&
Graph_update==1)
                {graph_index=0;
    idCount=0;iqCount=0;}
              else if ((graph_index<4096)&&
(Isr_num==0))
                {graph_index++;}

    #endif // (BUILDLEVEL==LEVEL3
Current_Loop+Speed_Loop closed test)

// ****** LEVEL4: AC IM coupled with DC motor;
Pulse IdRef=0.4;*********
        //Actual machine closed Current_Loop +
Speed_Loop test working (level 2)
        //pid1_iq.Ref =
pid1_spd.Out;==>pid1_iq.Ref =0; lsw=0 to run;
        //      if (idCount<=500 || (idCount>=2000
&& idCount<=2500))    {idCount++; pid1_id.Ref =
0;}
        //              else
```

```
        //                  {idCount++; pid1_id.Ref =
_IQ(IdRef);}    // _IQ(IqRef)

                        #if  (BUILDLEVEL==LEVEL4)
//
                    if (lsw==1)
                        {if
(EvbRegs.COMCONB.bit.FCOMPOE==0)
{Enable_Inv();}

                        //    RMP module and call the
Ramp control calculation function.
                        rc1.TargetValue =
_IQ(SpeedRef);    //rc1.calc() modified
                        RC_MACRO(rc1);    //rc1:var
name; calc: field name, call rmp_cntl_calc()
                        //    RAMP GEN module and
call the Ramp machine calculation function.
                        rg1.Freq = rc1.SetpointValue;
//SetpointValue add 0.00005 each step
                        RG_MACRO(rg1);
//v->StepAngleMax*v->Freq;    StepAngleMax =
BASE_FREQ*Tsampling;
                        }
                        else    {rc1.SetpointValue=0;
rg1.Out= 0; }//Disable_Inv();

            //      Call the ILEG2_VDC read function.
ilg2_vdc1.read(&ilg2_vdc1);
            //      get_theta();
                    get_adc_V();
                    Vdc=_IQ18(100);

                    //    CLARKE module and call the clarke
transformation calculation function.    -1~+1
                        clarke1.As =
_IQdiv(_IQ18toIQ((int32)Iu),_IQ(BASE_CURRE
NT));    //Iu _iq18 format BASE_CURRENT =
7.778
                        clarke1.Bs =
_IQdiv(_IQ18toIQ((int32)Iw),_IQ(BASE_CURRE
NT));    //Iw _iq18 format change to _iq20 format
                        CLARKE_MACRO(clarke1);
//Iu_L, Iw_L are real values

                    //    PARK module and call the park
transformation calculation function.
                    park1.Alpha = clarke1.Alpha;
                    park1.Beta = clarke1.Beta;

                    if (lsw == 0) {park1.Angle =
rg1.Out;}
                      else if (lsw == 1) {park1.Angle =
fe1.ThetaFlux;}

                    park1.Sine = _IQsinPU(park1.Angle);
                    park1.Cosine =
_IQcosPU(park1.Angle);
                    PARK_MACRO(park1);

                    //    Connect inputs of the PID module
and call the PID speed controller calculation
function.
                    if
(SpeedLoopCount==SpeedLoopPrescaler)
                      {
```

```
                pid1_spd.Ref = rc1.SetpointValue;
                pid1_spd.Fbk = se1.WrHat;
                PI_MACRO(pid1_spd);
                SpeedLoopCount=1;     }
              else SpeedLoopCount++;

               if(lsw==0)      {pid1_spd.ui=0;
pid1_spd.i1=0;}

            //    PID_REG3 module and call the
PID ID controller calculation function.
               if (idCount<=500)    {idCount++;
pid1_id.Ref = 0;}
                                 else

//{idCount++;pid1_id.Ref=_IQ(IdRef);}
                                    {idCount++;
                                   if
(pid1_id.Ref<_IQ(IdRef))
                                   {pid1_id.Ref =
pid1_id.Ref+_IQ(0.005);}
                                   else
{pid1_id.Ref=_IQ(IdRef);}}// _IQ(IqRef)
     //          pid1_id.Ref = _IQ(IdRef);
              pid1_id.Fbk = park1.Ds;
              PI_MACRO(pid1_id);
//==>pid1_id.Out



          //    PID_REG3 module and call the PID
IQ controller calculation function.
               if (lsw == 0)    {pid1_iq.Ref =
0;}//_IQ(IqRef);
               else if (lsw == 1) {pid1_iq.Ref =
pid1_spd.Out;} //connect to aci_se; speed closed
loop

               pid1_iq.Fbk = park1.Qs;
               PI_MACRO(pid1_iq);
//==>pid1_id.Out

          // INV_PARK module and call the
inverse park transformation calculation function.
               ipark1.Ds = pid1_id.Out;
               ipark1.Qs = pid1_iq.Out;
               ipark1.Sine = park1.Sine;
               ipark1.Cosine = park1.Cosine;
               IPARK_MACRO(ipark1);
//==>ipark1.Ds; ipark1.Qs

          // Connect inputs of the ACI module and
call the induction motor model calculation function.
          //        aci1.Ualpha = ipark1.Alpha;
          //        aci1.Ubeta = ipark1.Beta;
          //        aci1.calc(&aci1);
          //    SVGEN_DQ module and call the
space-vector gen.calculation function.
               svgen_dq1.Ualpha = ipark1.Alpha;
//id
               svgen_dq1.Ubeta = ipark1.Beta;    //iq
               SVGEN_MACRO(svgen_dq1);
//==>Ta, Tb, Tc

          //    PWM_DRV module and call the
PWM signal generation update function.
```

```
            pwm1.MfuncC1 =
(int16)_IQtoIQ15(svgen_dq1.Ta); // MfuncC1 is in
Q15
            pwm1.MfuncC2 =
(int16)_IQtoIQ15(svgen_dq1.Tb); // MfuncC2 is in
Q15
            pwm1.MfuncC3 =
(int16)_IQtoIQ15(svgen_dq1.Tc); // MfuncC3 is in
Q15
            pwm1.update(&pwm1);


        //      Volt Macro to reconstruct
Vphase_ABC and Valpha, Vbeta;
            volt1.MfuncV1 = svgen_dq1.Ta;
            volt1.MfuncV2 = svgen_dq1.Tb;
            volt1.MfuncV3 = svgen_dq1.Tc;
            volt1.DcBusVolt =
_IQdiv(_IQ18toIQ((int32)Vdc),_IQ(BASE_VOLT
AGE));
            VOLT_MACRO(volt1);


        //COMPARE CM1.THETA AND
FE1.FLUXTHETA CURRENT MODE NEED
SPEED
        //ACI3-3 HAS SPEED CAPTURE,
ACI3-4 USE VIRTUAL MACHINE
        //PAUSE VERIFY VOLTAGE AND
CURRENT PU VALUE; CHECK Rotor speed


        // ACI Flux angle estimator interface
            fe1.IDsS = clarke1.Alpha;    //output
from CLARKE in pu
            fe1.IQsS = clarke1.Beta;
            fe1.UDsS = volt1.Valpha;    //output
from VOLT in pu
            fe1.UQsS = volt1.Vbeta;
            ACIFE_MACRO(fe1);


        //    ACI Speed Estimator interface to
generate Estimated Speed
            se1.IDsS = clarke1.Alpha;    // in pu
            se1.IQsS = clarke1.Beta;     // in pu
            se1.ThetaFlux = fe1.ThetaFlux;
            se1.PsiDrS = fe1.PsiDrS;
            se1.PsiQrS = fe1.PsiQrS;


            ACISE_MACRO(se1);
          if (pid1_id.Ref==0)
{se1.WrHatRpm=0;}


            //      speed_pu1 =se1.WrHat;
//speed in pu
            //      speed_rpm1=se1.WrHatRpm;
//speed in rpm


        //    Waveforms display
              graph_data[graph_index] =
pid1_id.Ref;//fe1.ThetaFlux;
//pid1_id.Ref;//volt1.Valpha;//ipark1.Alpha;//clarke
1.As;//pid1_id.up;//park1.Ds;//
              graph_data1[graph_index]
=se1.WrHatRpm;//pid1_id.Fbk;//volt1.Vbeta;//ipar
k1.Beta;//clarke1.Bs;//pid1_iq.up;//park1.Qs;//


        //    if (Isr_num<1)    {Isr_num++;}
else    {Isr_num=0;}
```

```
        //      if (lsw ==1)
{Graph_update=1;lsw_prestate=1;}    else
{Graph_update=0;}
            //    if (lsw_prestate==1 && lsw==0)
{graph_index=0; lsw_prestate=0;}

    //          graph_data2[graph_index] =
volt1.Valpha;//_IQmpy(_IQ(volt1.Valpha),_IQ(BA
SE_VOLTAGE));
    //          graph_data3[graph_index] =
volt1.VphaseA;
//_IQmpy(_IQ(volt1.VphaseA),_IQ(BASE_VOLT
AGE));

                if (graph_index==4096 &&
Graph_update==1)
                        {graph_index=0;
    idCount=0;iqCount=0;Graph_update=1;}
                    else if ((graph_index<4096)&&
(Isr_num==0))
                        {graph_index++;}

                    #endif //
(BUILDLEVEL==LEVEL4
Current_Loop+Speed_Loop closed test)
/*************************************
/
//      Call the PWMDAC update function.
//    pwmdac1.update(&pwmdac1);
//      Call the DATALOG update function.
//      dlog.update(&dlog);
// Enable more interrupts from this timer
    EvaRegs.EVAIMRA.bit.T1UFINT = 1;
// Note: To be safe, use a mask value to write to the
entire
    // EVAIFRA register.    Writing to one bit will
cause a read-modify-write
    // operation that may have the result of writing
1's to clear
    // bits other then those intended.
    EvaRegs.EVAIFRA.all = BIT9;

// Acknowledge interrupt to recieve more interrupts
from PIE group 2
    PieCtrlRegs.PIEACK.all =
PIEACK_GROUP2;
}


/*************************************/
/*  NAME:    adc_isr()                */
/*  RETURNS:        void              */
/*  DESCRIPTION: Interrupt for ADC results
revieval      */
/*************************************
/
interrupt void    adc_isr(void)
{       Sample_ch0[ConversionCount]=((AdcRegs.
ADCRESULT0>>4) );
Sample_ch1[ConversionCount]=((AdcRegs.ADCR
ESULT1>>4) );
Sample_ch2[ConversionCount]=((AdcRegs.ADCR
ESULT2>>4) );
Sample_ch3[ConversionCount]=((AdcRegs.ADCR
ESULT3>>4) );
```

```
Sample_ch4[ConversionCount]=((AdcRegs.ADCR
ESULT4>>4) );
Sample_ch7[ConversionCount]=((AdcRegs.ADCR
ESULT7>>4) );
Sample_ch8[ConversionCount1]=((AdcRegs.ADC
RESULT8>>4) );//Iw
Sample_ch9[ConversionCount]=((AdcRegs.ADCR
ESULT9>>4) );
Sample_ch10[ConversionCount]=((AdcRegs.ADC
RESULT10>>4) );
Sample_ch11[ConversionCount]=((AdcRegs.ADC
RESULT11>>4) );
Sample_ch12[ConversionCount]=((AdcRegs.ADC
RESULT12>>4) );
Sample_ch13[ConversionCount]=((AdcRegs.ADC
RESULT13>>4) );
Sample_ch14[ConversionCount]=((AdcRegs.ADC
RESULT14>>4) );
Sample_ch15[ConversionCount]=((AdcRegs.ADC
RESULT15>>4) );
        if  (ConversionCount==AVE_SAMPLE-1)
ConversionCount=0;
        else ConversionCount++;
Sample_ch5[ConversionCount1]=((AdcRegs.ADC
RESULT5>>4) );//Udc+
Sample_ch6[ConversionCount1]=((AdcRegs.ADC
RESULT6>>4) );//Udc-
        if
(ConversionCount1==AVE_SAMPLE_DC-1)
ConversionCount1=0;
        else ConversionCount1++;
   AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;
// Reset SEQ1
   AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
// Clear INT SEQ1 bit
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
// Acknowledge interrupt to PIE
} /*End of Adc_isr*/

/***********************************/
/*   NAME:     InitAdc()          */
/*   RETURNS:        void       */
/*   DESCRIPTION: ADC configuration, use
Event Manager A to start ADC */
/***********************************/
void InitAdc(void)
{//0:ADINA0:Temperature sensor from left(A) side
//1:ADINA1:Phase A current ia from left(A) side,
-50A/0x7ff
//2:ADINA2:Phase C current ic from left(A) side,
-50A/0x7ff
//3:ADINA3:Phase A or line AB voltage from
left(A) side, 500V/0x7ff
//4:ADINA4:Phase B or line BC voltage from
left(A) side, 500V/0x7ff
//5:ADINA5:Phase C or line CA voltage from
left(A) side, 500V/0x7ff
//6:ADINA6:Positive side DC link voltage sensor
from left(A) side, -500V/0x7ff
//7:ADINA7:Negative side DC link voltage sensor
from left(A) side, -500V/0x7ff
//8:ADINB0:Temperature sensor from right (B)
side
```

```
//9:ADINB1:Phase W or line WU voltage sensor from right (B) side, 500V/0x7ff
//A:ADINB2:Negative side DC link voltage sensor from right (B) side, -500V/0x7ff
//B:ADINB3:Positive side DC link voltage sensor from right (B) side, -500V/0x7ff
//C:ADINB4:Phase U current sensor from right (B) side, -50A/0x7ff
//D:ADINB5:Phase W current sensor from right (B) side, -50A/0x7ff
//E:ADINB6:Phase U or line UV voltage sensor from rihgt(B) side, 500V/0x7ff
//F:ADINB7:Phase V or line VW voltage sensor from right(B) side, 500V/0x7ff
      /*Start ADC*/
      AdcRegs.ADCTRL3.bit.ADCBGRFDN = 0x3;
      // Power up bandgap/reference circuitry
      Delay_micro_second (ADC_usDELAY);
// Delay before powering up rest of ADC
      AdcRegs.ADCTRL3.bit.ADCPWDN = 1;
      // Power up rest of ADC
      Delay_micro_second (ADC_usDELAY2);

   /*Configure ADC*/
    AdcRegs.ADCTRL1.bit.ACQ_PS = 0x1;//S/H width in ADC module periods = 2 ADC clocks
    AdcRegs.ADCTRL3.bit.ADCCLKPS = 0x5; //ADC module clock = HSPCLK/5 = 25MHz
    AdcRegs.ADCTRL1.bit.SEQ_CASC = 1; // 1    Cascaded mode
    AdcRegs.ADCTRL1.bit.CONT_RUN = 0; //not Continous running
    AdcRegs.ADCTRL1.bit.SEQ_OVRD = 1; //****      // Enable Sequencer override feature (bit 5)
    AdcRegs.ADCMAXCONV.all = 0x000B; // Setup 16 conv's on SEQ1
AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x1; //AdcResult##(CONV##)<--ADINA##
AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x2;
AdcRegs.ADCCHSELSEQ1.bit.CONV02 = 0x3;
AdcRegs.ADCCHSELSEQ1.bit.CONV03 = 0x4;
AdcRegs.ADCCHSELSEQ2.bit.CONV04 = 0x5;
AdcRegs.ADCCHSELSEQ2.bit.CONV05 = 0x6;
    AdcRegs.ADCCHSELSEQ2.bit.CONV06 = 0x7;
AdcRegs.ADCCHSELSEQ2.bit.CONV07 = 0xC;
AdcRegs.ADCCHSELSEQ3.bit.CONV08 = 0xD;
AdcRegs.ADCCHSELSEQ3.bit.CONV09 = 0xE;
AdcRegs.ADCCHSELSEQ3.bit.CONV10 = 0xF;
AdcRegs.ADCCHSELSEQ3.bit.CONV11 = 0x9;
//AdcRegs.ADCCHSELSEQ4.bit.CONV12 = 0x8;
//AdcRegs.ADCCHSELSEQ4.bit.CONV13 = 0xE;
//AdcRegs.ADCCHSELSEQ4.bit.CONV14 = 0xF;
//AdcRegs.ADCCHSELSEQ4.bit.CONV15 = 0x9;
//AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1 = 1; // enable SEQ1 interrupt (every EOS)
AdcRegs.ADCTRL2.bit.EVB_SOC_SEQ = 1; // enable EVBSOC to start SEQ1
}   /*End of InitAdc*/
/*************************************/
/*   NAME: get_adc_V()              */
/*   RETURNS:      void             */
/*   DESCRIPTION: Line voltage and current conditioning with multi-sampling*/
```

```
/*************************************/
void get_adc_V(void)
{
    Uint16 i;

    Uint32 Sum_ch0=0;
    Uint32 Sum_ch1=0;
    Uint32 Sum_ch2=0;
    Uint32 Sum_ch3=0;
    Uint32 Sum_ch4=0;
    Uint32 Sum_ch5=0;
    Uint32 Sum_ch6=0;
    Uint32 Sum_ch7=0;
    Uint32 Sum_ch8=0;
    Uint32 Sum_ch9=0;
    Uint32 Sum_ch10=0;
    Uint32 Sum_ch11=0;
    //Uint32 Sum_ch12=0;
    //Uint32 Sum_ch13=0;
    //Uint32 Sum_ch14=0;
    //Uint32 Sum_ch15=0;

    for(i = 0; i<AVE_SAMPLE; i++)
        {
          Sum_ch0+=Sample_ch0[i];
          Sum_ch1+=Sample_ch1[i];
          Sum_ch2+=Sample_ch2[i];
          Sum_ch3+=Sample_ch3[i];
          Sum_ch4+=Sample_ch4[i];
          Sum_ch7+=Sample_ch7[i];
         Sum_ch8+=Sample_ch8[i];
          Sum_ch9+=Sample_ch9[i];
        Sum_ch10+=Sample_ch10[i];
          Sum_ch11+=Sample_ch11[i];
          //Sum_ch12+=Sample_ch12[i];
          //Sum_ch13+=Sample_ch13[i];
          //Sum_ch14+=Sample_ch14[i];
          //Sum_ch15+=Sample_ch15[i];
          }
      for(i = 0; i<AVE_SAMPLE_DC; i++)
          {
            Sum_ch5+=Sample_ch5[i];
            Sum_ch6+=Sample_ch6[i];
          }
    /*************************************/
          Ia2=Ia1;
          Ia1=Ia;
          Ic2=Ic1;
          Ic1=Ic;
       /*Chanel 1*/
          Ia=Sum_ch0 -
    (AD_BIAS0<<(AVE_SHFT));
          Ia=(Ia<<(18-AVE_SHFT));        //(_iq18
    format)
          Ia=Ia/2044; /*2023 for Iag,2035 for Ia*/
          Ia=-Ia*50; /*current sensor 5:1 */
        /*Chanel 2*/
          Ic=Sum_ch1 -
    (AD_BIAS1<<(AVE_SHFT));
          Ic=(Ic<<(18-AVE_SHFT)); //(_iq18 format)
          Ic=Ic/2048; /*2002 for Icg,2050 for Ic*/
          Ic=-Ic*50; /*current sensor 5:1 */

        /*Chanel 3*/
          Vab=Sum_ch2 -
    (AD_BIAS2<<AVE_SHFT);
```

```
        Vab=(Vab<<(18-AVE_SHFT));  //(_iq18
format)
        Vab=Vab/2100; /*DSP ADC trim: 5V=3159
-5V=1039*/
        Vab=Vab*300; /*Voltage sensor 30:1 */

    /*Chanel 4*/
        Vbc=Sum_ch3 -
(AD_BIAS3<<AVE_SHFT);
        Vbc=(Vbc<<(18-AVE_SHFT)); // (_iq18
format)
        Vbc=Vbc/2098;/*DSP ADC trim: 5V=3156
-5V=1042*/
        Vbc=Vbc*300; /*Voltage sensor 30:1 */

    /*Chanel 5*/
        Vca=Sum_ch4 -
(AD_BIAS4<<AVE_SHFT);
        Vca=(Vca<<(18-AVE_SHFT)); // (_iq18
format)
        Vca=Vca/2132;    /*DSP ADC trim:
5V=3170 -5V=1026*/
        Vca=Vca*300; /*Voltage sensor 30:1*/

    /*Chanel 6*/
        Vdc_P = Sum_ch5 -
(AD_BIAS5<<(AVE_SHFT_DC));
        Vdc_P = (Vdc_P<<(18-AVE_SHFT_DC));
//      (_iq18 format)
        Vdc_P=Vdc_P/1920;                /*DSP
ADC trim: 5V=1041 -5V=3158*/
        Vdc_P=-Vdc_P*300;         ///* sensor
30:1*/*/
    /*Chanel 7*/
        Vdc_N = Sum_ch6 -
(AD_BIAS6<<(AVE_SHFT_DC));
        Vdc_N = (Vdc_N<<(18-AVE_SHFT_DC));
//(_iq18 format)
        Vdc_N=Vdc_N/2036;                /*DSP
ADC trim: 5V=1015 -5V=3130*/
        Vdc_N=-Vdc_N*300;         ///* sensor
30:1*/*/
        Vdc0=Vdc;
        Vdc=Vdc_P+Vdc_N;

    /*Chanel 12*/
        Iu=Sum_ch7 -
(AD_BIAS7<<(AVE_SHFT));
        Iu=(Iu<<(18-AVE_SHFT));        //(_iq18
format)
        Iu=Iu/2035;    //2050
        Iu=Iu*50; /*current sensor 5:1 */

    /*Chanel 13*/
        Iw=Sum_ch8 -
(AD_BIAS8<<(AVE_SHFT));
        Iw=(Iw<<(18-AVE_SHFT)); //(_iq18
format)
        Iw=Iw/2060; //2050
        Iw=Iw*50; /*current sensor 5:1 */
//      Iw+=_IQ18(0.14);

        Iv=-Iw-Iu;
    /*Chanel 14*/
```

```
        Euv=Sum_ch9 -
(AD_BIAS9<<AVE_SHFT);
        Euv=(Euv<<(18-AVE_SHFT));  //(_iq18
format)
        Euv=Euv/2100; /*DSP ADC trim: 5V=3159
-5V=1039*/
        Euv=Euv*300; /*Voltage sensor 30:1 */

      /*Chanel 15*/
        Evw=Sum_ch10 -
(AD_BIAS10<<AVE_SHFT);
        Evw=(Evw<<(18-AVE_SHFT)); // (_iq18
format)
        Evw=Evw/2100;/*DSP ADC trim:
5V=3156 -5V=1042*/
        Evw=Evw*300; /*Voltage sensor 30:1 */

      /*Chanel 9*/
        Ewu=Sum_ch11 -
(AD_BIAS11<<AVE_SHFT);
        Ewu=(Ewu<<(18-AVE_SHFT));      //
(_iq18 format)
        Ewu=Ewu/2100;    /*DSP ADC trim:
5V=3170 -5V=1026*/
        Ewu=Ewu*300; /*Voltage sensor 30:1*/

   /*Low pass filtering for Vdc*/
        Vdc_L0=Vdc_L;
        Vdc_L= _IQ18mpyIQX(LPFa_DC,28,
(Vdc0+Vdc),18);
        Vdc_L= _IQ18mpyIQX(LPFb_DC,28,
Vdc_L0,18)+Vdc_L;

    /*Low pass filtering for Iu*/
        Iu_L0=Iu_L;
        Iu_L= _IQ18mpyIQX(LPFa_DC,28,
(Iu0+Iu),18);
        Iu_L+= _IQ18mpyIQX(LPFb_DC,28,
Iu_L0,18);      //DC means count 4 times
        Iu0=Iu;

   /*Low pass filtering for Iw*/
        Iw_L0=Iw_L;
        Iw_L= _IQ18mpyIQX(LPFa_DC,28,
(Iw0+Iw),18);
        Iw_L+= _IQ18mpyIQX(LPFb_DC,28,
Iw_L0,18);
        Iw0=Iw;
   /*Electrical torque of DC motor*/
//          torque_motor=_IQ18mpy((Iu-Io),
_IQ18(Kf));

} /*End of Get_adc_V*/

/***********************************/
/*   NAME:            AD_bias()     */
/*   RETURNS:         void          */
/*   DESCRIPTION: ADC dc bias caliberation
*/
/***********************************/
 int AD_bias (unsigned int channel)
{
      int32 bias_data=0;
      Uint32 i;
      bias_data=0;
      for (i=0;i<1024;i++){
```

```
        while
(AdcRegs.ADCST.bit.INT_SEQ1== 0){}
        AdcRegs.ADCST.bit.INT_SEQ1_CLR =
1;        // Clear INT SEQ1 bit
        switch (channel) {
            case 0:
    bias_data+=AdcRegs.ADCRESULT0>>4;
                break;
            case 1:
    bias_data+=AdcRegs.ADCRESULT1>>4;
                break;
            case 2:
    bias_data+=AdcRegs.ADCRESULT2>>4;
                break;
            case 3:
    bias_data+=AdcRegs.ADCRESULT3>>4;
                break;
            case 4:
    bias_data+=AdcRegs.ADCRESULT4>>4;
                break;
            case 5:
    bias_data+=AdcRegs.ADCRESULT5>>4;
                break;
            case 6:
    bias_data+=AdcRegs.ADCRESULT6>>4;
                break;
            case 7:

    bias_data+=AdcRegs.ADCRESULT7>>4;
                break;
            case 8:
    bias_data+=AdcRegs.ADCRESULT8>>4;
                break;
            case 9:
    bias_data+=AdcRegs.ADCRESULT9>>4;
                break;
            case 10:
    bias_data+=AdcRegs.ADCRESULT10>>4;
                break;
            case 11:
    bias_data+=AdcRegs.ADCRESULT11>>4;
                break;
            case 12:
    bias_data+=AdcRegs.ADCRESULT12>>4;
                break;
            case 13:
    bias_data+=AdcRegs.ADCRESULT13>>4;
                break;
            case 14:
    bias_data+=AdcRegs.ADCRESULT14>>4;
                break;
            case 15:
            default:
    bias_data+=AdcRegs.ADCRESULT15>>4;
                break;
          }
        }
        bias_data=bias_data>>10;
        return bias_data;
    } /*End of AD_bias*/
    /***************************/
    /*  NAME:   AD_bias16()          */
    /*  RETURNS:      void           */
    /*  DESCRIPTION: LCD display and ADC dc
    bias caliberation */
    /*****************************/
```

```
void AD_bias16(void)
{     //Text_LCD_clear_display();
      //Text_LCD_printxy(0,1,"    Test the AD
Bias");
      //Text_LCD_printxy(0,2,"    Calculating CH
0");
      AD_BIAS0=AD_bias(0);//2059
      //Text_LCD_printxy(0,2,"    Calculating CH
1");
      AD_BIAS1=AD_bias(1);//2066
      //Text_LCD_printxy(0,2,"    Calculating CH
2");
      AD_BIAS2=AD_bias(2);//2053
      //Text_LCD_printxy(0,2,"    Calculating CH
3");
      AD_BIAS3=AD_bias(3);//2059
      //Text_LCD_printxy(0,2,"    Calculating CH
4");
      AD_BIAS4=AD_bias(4);//2059
      //Text_LCD_printxy(0,2,"    Calculating CH
5");
      AD_BIAS5=AD_bias(5);
      //Text_LCD_printxy(0,2,"    Calculating CH
6");
      AD_BIAS6=AD_bias(6);
      //Text_LCD_printxy(0,2,"    Calculating CH
7");
      AD_BIAS7=AD_bias(7);//2047
      //Text_LCD_printxy(0,2,"    Calculating CH
8");
      AD_BIAS8=AD_bias(8);//2050
      //Text_LCD_printxy(0,2,"    Calculating CH
9");
      AD_BIAS9=AD_bias(9);//2055
      //Text_LCD_printxy(0,2,"    Calculating CH
10");
      AD_BIAS10=AD_bias(10);//2051
      //Text_LCD_printxy(0,2,"    Calculating CH
11");
      AD_BIAS11=AD_bias(11);//2047
      /*//Text_LCD_printxy(0,2,"    Calculating CH
12");
      AD_BIAS12=AD_bias(12);
      //Text_LCD_printxy(0,2,"    Calculating CH
13");
      AD_BIAS13=AD_bias(13);
      //Text_LCD_printxy(0,2,"    Calculating CH
14");
      AD_BIAS14=AD_bias(14);
      //Text_LCD_printxy(0,2,"    Calculating CH
15");
      AD_BIAS15=AD_bias(15);*/
      //Text_LCD_printxy(0,2,"    Calculation
DONE ");
      Delay_mili_second(100);
} /*End of AD_bias16*/

void InitEVA(void)
{
    // Configure EVA, EVA Clock is already
enabled in InitSysCtrl();

EvaRegs.EVAIMRA.bit.PDPINTA=1;//PDPINTA
protection enable
  // Enable compare for PWM1-PWM6
   EvaRegs.CMPR1 = 0x0000;
```

```
   EvaRegs.CMPR2 = 0x0000;
   EvaRegs.CMPR3 = 0x0000;
   // Compare action control.    Action that takes
place
   EvaRegs.ACTRA.all = 0x0999;// active low
   EvaRegs.DBTCONA.all = 0x0FF4; // Enable
deadband 3.4uS
   EvaRegs.COMCONA.all = 0xA000;//update
twise

/* Initialize QEP circuits*/
   EvaRegs.CAPCONA.all = 0x9004;//disable
capture 1,2;select timer2;detect rising edge for
capture 3.

/* Initalize EVA Timer1*/
    EvaRegs.T1PR = Tsw;//    Timer1 period (for
motor side switching frequency)
    EvaRegs.T1CNT = 0x0000;          // Timer1
counter
    // Timer enable
    EvaRegs.T1CON.all = 0x0800;    //
(updowncount mode) //input clock(HSPCLK)/1
    //EvaRegs.EVAIMRA.bit.T1UFINT=1;
//Enable Timer1 underflow interrupt

 /* Initalize EVA Timer2*/
    EvaRegs.T2CNT = 0x0000;
    EvaRegs.T2PR =39999; // period register (for
3600*4 pules every cycle of speed sensor)
    EvaRegs.T2CON.all = 0x1870;
//(directional-up/downcount mode)input clock from
QEP
EvaRegs.T1CON.bit.TENABLE=1;//Timer enable
}
void InitEVB(void)
{
    // Configure EVA
    // EVA Clock is already enabled in
InitSysCtrl();

EvbRegs.EVBIMRA.bit.PDPINTB=1;//PDPINTA
protection enable
 // Enable compare for PWM7-PWM12

   EvbRegs.CMPR4 = 0x0000;
   EvbRegs.CMPR5 = 0x0000;
   EvbRegs.CMPR6 = 0x0000;
   // Compare action control.    Action that takes
place on a cmpare event
   EvbRegs.ACTRB.all = 0x0666;//active high
   EvbRegs.DBTCONB.all = 0x0FF4; // Enable
deadband 3.4uS
   EvbRegs.COMCONB.all =
0xA000;//0xA000;//seven segments and update
twice

/* Initalize the timers*/
/* Initalize EVB Timer3*/

    EvbRegs.T3PR = Tsw;// Timer3 PWM
period(for computing fre of 8997Hz which is twice
of rectifier switching frequency)
    EvbRegs.T3CMPR = Tsw>>1;         // Timer3
compare
```

```
    EvbRegs.T3CNT = 0x0000;            // Timer3
counter
    // TMODE = continuous up/down
    // Timer enable
    EvbRegs.T3CON.all = 0x0802;    //
(updowncount mode) //input
clock(HSPCLK)/1//timer compare enable
    EvbRegs.EVBIMRA.bit.T3CINT=1; //Enable
Timer3 compare interrupt

  /* Initalize EVB Timer4*/
    EvbRegs.T4CNT = 0x0000;
    EvbRegs.T4PR = Tsw>>4;      //4    Setup
period register (for sampling feq of 8997Hz*8)
    EvbRegs.GPTCONB.bit.T4TOADC = 1;
// Enable EVASOC in EVB (underflow starts ADC)
    EvbRegs.T4CON.all = 0x0880;            //
(updowncount mode) //input
clock(HSPCLK)/1//start with T3
    EvbRegs.T3CON.bit.TENABLE=1;//Timer
enable

    /* Initalize EVB Capture4&5*/
    EvbRegs.CAPCONB.all=0xA6F0;
    EvbRegs.EVBIMRC.all=0x0003;
    EvbRegs.EVBIFRC.all=0x0007;
}
/************************************/
/*    NAME:       pdpinta_isr()    */
/*    RETURNS:          void          */
/*    DESCRIPTION: Protection interrupt generated
from FPGA*/
/************************************/
interrupt void pdpinta_isr(void)
{
      Open_Relay ();
      Disable_Inv();
      FAULT = 1;
      PieCtrlRegs.PIEACK.all =
PIEACK_GROUP1;// Acknowledge interrupt to
PIE
      /*No clear of the interrupt flag*/
} /*End of Pdpinta_isr*/

/************************************/
/*    NAME:
Enable_Rec(),Disable_Rec(),Enable_Inv(),Disable_
Inv()    */
/*    DESCRIPTION: Enable or disable PWM
output                */
/************************************
/
  void Enable_Inv(void)
{     EvbRegs.COMCONB.bit.FCOMPOE=1;
  }
  void Disable_Inv(void)
{     EvbRegs.COMCONB.bit.FCOMPOE=0;
  }
/************************************/
/* NAME: Close_Relay(), Open_Relay */
/*    DESCRIPTION: Open and close relay for
protection    */
/************************************/
void Close_Relay(void)
{
   Uint16 i;
```

```
GpioDataRegs.GPBCLEAR.bit.GPIOB12 =
1;//grid side charging resistor by-pass relay open
   GpioDataRegs.GPACLEAR.bit.GPIOA7 =
1;//motor side charging resistor by-pass relay open
   Delay_mili_second(200);
   GpioDataRegs.GPBDAT.bit.GPIOB7 = 1;//grid
side power relay close
      for (i=0; i<10; i++)
      {
      Delay_mili_second(100);
//     LED_display();
      }
GpioDataRegs.GPBDAT.bit.GPIOB12 = 1;//grid
side charging resistor by-pass relay close
   GpioDataRegs.GPADAT.bit.GPIOA7 =
1;//motor side charging resistor by-pass relay close
//    CONTACTOR = 1;
}

void Open_Relay(void)
{    GpioDataRegs.GPBCLEAR.bit.GPIOB7 =
1;//grid side power relay open
   //CONTACTOR = 0;
}

void Delay_mili_second (int delay)
{
      Uint16 i,j;
      i=delay*30;//150M Hz DSP clock
      for (j=0; j<1000; j++)
      {
        while (i){i--;};
        i=delay*30;
      }
}

void Delay_micro_second (int delay)
{
      unsigned int i;
      i=delay*30; //150M Hz DSP clock
      while (i){i--;};
}
  //---------------------------------------------
// The End.
void InitGpio(void)
{
        // sets GPIO Muxs as I/Os
        EALLOW;
      GpioMuxRegs.GPAMUX.all= 0x077F;     //
Configure MUXs as digital I/Os or
      GpioMuxRegs.GPBMUX.all= 0x077F;     //
peripheral I/Os
      GpioMuxRegs.GPDMUX.all=0x0000;
GpioMuxRegs.GPEMUX.all=0x0000;
GpioMuxRegs.GPFMUX.all=0x0030;
GpioMuxRegs.GPGMUX.all=0x0030;
GpioMuxRegs.GPADIR.all=0xF8FF;     //
GpioMuxRegs.GPBDIR.all=0xF8FF;     // GPIO
DIR select GPIOs as output or input
GpioMuxRegs.GPDDIR.all=0x0000;
GpioMuxRegs.GPEDIR.all=0x0000;
GpioMuxRegs.GPFDIR.all=0x3FDF;
      GpioMuxRegs.GPGDIR.all= 0x0010;
GpioMuxRegs.GPAQUAL.all=0x0000;    // Set
GPIO input qualifier values
GpioMuxRegs.GPBQUAL.all=0x0000;
```

```
GpioMuxRegs.GPDQUAL.all=0x000F
GpioMuxRegs.GPEQUAL.all=0x000F

      EDIS;
} /*End of Gpio_select*/
/*****************************************
/
/*       FUNCTION DECLARATION    */
/*****************************************
/
/*    NAME:      Protection_PLL()       */
/*    RETURNS:         void                */
/*    DESCRIPTION: Protection action when PLL
is lost       */
/*********************************/
void Protection_PLL (void)
{
      Open_Relay ();
//       Disable_Rec();
      Disable_Inv();
      INTERRUPT = 0;
      FAULT = 1;
      CONTACTOR = 0;
      Fault_info_A=*FAULT_DETECT_A;
      Fault_info_B=*FAULT_DETECT_B;
//       Display_Fault ();
} /*End of Protection_PLL*/
/*****************************************
/
/*    NAME:    init_fpga()               */
/*    RETURNS:         void              */
/*    DESCRIPTION: Initialization of FPGA
registers          */
/*************************************/
 void init_fpga(void)
 {        /*Hardware Protection*/
       *FAULT_CLEAR_A = 0x0;
       *FAULT_ENABLE_A = 0x1ff;
       *FAULT_CLEAR_B = 0x0;
       *FAULT_ENABLE_B = 0x1ff;
       /*Thermal Protection (disabled)*/
       *FAULT_THERMO_ENABLE = 0x0;
       *FAN_PERIOD = 14999;
       *FAN_COUNT = 0x0;
       *FAN_COMPARE_A = 0x0;
       *FAN_COMPARE_B = 12180;
       //*FAN_CONFIG = 0x1F;
       /*Fault signal distribution*/
       *FAULT_DISTRIBUTE = 0xffff;
       /*Zero-crossing detection (Zero-crossing
starts automatically)*/
       *PHASE_SHIFT_A = 0x4DDC;       //
0x4c2c ///*shift input to 90degree delay*/
                                                  // 0xb090
/*shift input to 180degree delay*/
       //*PHASE_SHIfT_CONFIG_A = 0x0118;///*
divider=24+1,DPLL freq=Fclock/(divider+1); input
Vbc*/
       *PHASE_SHIFT_CONFIG_A =
0x0318;//0x0318:Vu or Vuw 0x0118:Vb or Vbc;
0x0418 Vv or Vvw
       /*IO pin configuration*/
       *GPIOA6_CONFIG = 0x0A; /*A side PLL
output*/
       *GPIOB6_CONFIG = 0x0B; /*A side SYN
signal*/
```

```
        /*XINT interrupt configuration*/
        *XINT1_CONFIG = 0x0; /*Disable XINT1,
2*/
        *XINT2_CONFIG = 0x0;
        /*Key input Text LCD displace interface*/
        *KEY_IN = 0xf;//clear all the registered key
input
        *SWITCH_IN = 0x00;
          /*FPGA LED*/
        *FPGA_LED = 0x0;
  } /*End of Init_fpga*/
/*****************************/
/*    NAME:
InitXintfClocks(),InitPeripherals()      */
/*    RETURNS:          void
*/
/*    DESCRIPTION: System    configuration
*/
/*****************************/
void InitXintfClocks(void)
{XintfRegs.XTIMING0.all=0x000358AC;
XintfRegs.XTIMING6.all=0x0003E746;//3D4A5;
XintfRegs.XINTCNF2.all=0x00000007;
} /*End of InitXintfClocks*/
void InitPeripherals(void)
{
    //Text_LCD_initiate();
    InitAdc();
    InitEVA();
    InitEVB();
    init_fpga();
//     Init_variable();
}
 void Init_variable(void)
 {
   /*ADC*/
   ConversionCount=0;
   /*Low pass filter*/
//    LPFa=_IQ28(1-2.0/(2.0+w_AD/Sw_freq));
//    LPFb=_IQ28(1)-(LPFa<<1);
//    LPFa_DC=_IQ28(1-2.0/(2.0+w_DC/Sw_freq));
//    LPFb_DC=_IQ28(1)-(LPFa_DC<<1);
//
LPFa_speed=_IQ28(1-2.0/(2.0+w_speed/Sw_freq))
;
//    LPFb_speed=_IQ28(1)-(LPFa_speed<<1);
//
LPFa_torque=_IQ28(1-2.0/(2.0+w_torque/Sw_freq
));
//    LPFb_torque=_IQ28(1)-(LPFa_torque<<1);

   } /*End of Init_variable*/

/*// Initialize PWMDAC module;
t1con=0x8840;TPS=/1; 150e6/(2*2500*1*1)=30k
       pwmdac1.PeriodMax =
(SYS_FREQ*200/(30*2))*5; //2500
       pwmdac1.PwmDacInPointer0 =
&PwmDacCh1;
       pwmdac1.PwmDacInPointer1 =
&PwmDacCh2;
       pwmdac1.PwmDacInPointer2 =
&PwmDacCh3;
       pwmdac1.init(&pwmdac1);
// Initialize DATALOG module
       dlog.iptr1 = &DlogCh1;
```

```
        dlog.iptr2 = &DlogCh2;
        dlog.iptr3 = &DlogCh3;
        dlog.iptr4 = &DlogCh4;
        dlog.trig_value = 0x0;
        dlog.size = 0x100;
        dlog.prescalar = 1;
        dlog.init(&dlog);
// Initialize capture module
//      cap1.init(&cap1);
// Initialize enable drive module (FOR DMC1500
ONLY)
//      drv1.init(&drv1);
// Initialize ADC module
//      ilg2_vdc1.init(&ilg2_vdc1);
// Initialize the SPEED_PR module x128-T2,
150MHz, 1000-teeth sprocket
//      speed1.InputSelect = 0;
//      speed1.BaseRpm = 120*BASE_FREQ/P;
//rpm=120f/P
//      speed1.SpeedScaler =
60*(SYS_FREQ*1000000/1000)*1/(128*speed1.B
aseRpm);
*/
```