Design and Implementation of Four-quadrant Operation in Single-Switch Based Switched Reluctance Motor Drive System

By:
Sung Yeul Park

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

July 6, 2004
Blacksburg, Virginia

Keywords: Single Switch based Switched Reluctance Motor, Four-quadrant operation, Self-starting, Speed Control

Design and Implementation of Four-quadrant Operation in Single-Switch Based Switched Reluctance Motor Drive System

By:

Sung Yeul Park

(Abstract)

In step with development of advanced, cost effective semiconductors and electrical motor drive components, the Switched Reluctance Machine (SRM) has become the center of public attention. Interest in a single-phase SRM has arisen in many places, especially because of its low cost applications. However, some drawbacks have plagued single-phase SRM: the lack of self-starting capability and restricted operation conditions.

This thesis presents a four-quadrant operation SRM drive system with a single controllable switch for two phase configuration. The SRM's configuration has four main stator poles, four rotor poles, and four auxiliary stator poles. Because of this special arrangement, a four-quadrant operation with a given power converter topology and proposed control algorithms has been realized. The focus of the paper is to realize a four-quadrant operation with a single-switch converter based SRM. In addition, this research resulted in a new self-starting scheme without adding permanent magnets. Simulation results and experimental results utilizing the control algorithm verify the performance of the system.

Acknowledgements

There are numerous people I must thank who have helped me through the course of my graduate studies. I would like to express my deepest gratitude to my advisor, Dr. Krishnan Ramu. Without his support, this research would never have come to fruition. Also, I would like to thank Dr. Douglas K. Lindner and Dr. Daniel J. Stilwell for taking the time to be part of my examination committee.

Most of all, I thank God, who provided me with the opportunity and wisdom for graduate study.

I would also like to thank my family and my fellow students. Their consistent encouragements, prayer, and help made me to reach where I am now. In particular, I would like to thank Keun-Soo Ha for his assistance with the motor drive design and analysis problems. I would also like to thank Amanda Martin Staley for her permission to use data and information about machine characteristics from her M.S. thesis. I would like to thank those whom I worked with, my fellow graduate students in the Center for Rapid Transit Systems, who are Ajit Bhanot, Christopher Hudson, Nimal Savio Lobo, Corey Michael Barnes, and Hong-Sun Lim.

My parents, Chul Ju Park and Soo Nam Cho, my wife Song Suk, and my son Youngjin Daniel deserve special mention for their continuous support.

The converter, control algorithm and machine topology are the intellectual property of Panaphase Technologies, LLC. Permission to present this research is acknowledged.

Table of Contents

List of Figures

List of Tables

Chapter1. Introduction

1.1 SPSRM Features

A Switched reluctance machine (SRM) consist of doubly salient poles in stator and rotor. Windings are present on the stator, and the rotor has no magnet or winding coil and is made up of steel lamination. The principle of SRM to produce torque is: when wound coils on the stator are energized, the rotor moves into a position of minimum reluctance. To energize the stator pole phase, the conventional SRM drives usually use power electronic switches and diodes. Consequently, as the number of phases increases, more switches and diodes are required. This increases the cost of motor drives.

To realize mass producible, cost effective, and at the same time speed variable SRM, the configuration of a single phase SRM (SPSRM) has come to the forefront and has been spotlighted by current industry and home appliance applications. With fewer switches and diodes, and a lower cost of a machine, the SPSRM has various advantages. However, the three drawbacks of an SPSRM should be considered. First, lacks self-starting ability, which is caused by a lack of torque production at the point of the maximum inductance positions (aligned positions) and the minimum inductance positions (unaligned positions). Second, it has lower power utilization since it utilizes only 50% of the coil winding. Third, it has only a two-quadrant operation: one of which is the motoring operation and the other is the braking operation.

1.2 Solution for the SPSRM drawback

For overcoming the self-starting of the SPSRM, most papers suggested attaching a permanent magnet on its stator sides.[3, 4, 5]. Recently, two different approaches to coping with the self-starting emerged: one is to use auxiliary windings [5], and the other is to use saturation effects with the stepped rotor configured machine [7].

1.3 Proposal from this Thesis

In this research, to solve the problem of ensuring self-starting, the 4:4:4 machine configuration [5] was chosen. While most of the research papers [8, 9, 10] presented the four-quadrant operation with the multi-phase SRM, this thesis tells how to realize the four-quadrant operation with single switch based SRM (SSSRM), which is presented in [13].

1.4 Organization of Thesis

Rest of this thesis is organized into chapters in the following order.

- Chapter 2 - presents the proposed motor dimensions and characteristics, the self-starting scheme, and the principle of the four-quadrant operation with a single controllable switch.
- Chapter 3 – presents the nonlinear SSSRM modeling and how the dynamic simulation is designed.
- Chapter 4 - presents implementation including hardware interfaces with each module and software algorithm realization.
- Chapter 5 - presents the verification of its drive system performance with simulation results and experimental results for the controller design.
- Chapter 6 - summarizes the key results of its study and provides the conclusions of this research.

Chpater2. The Principle of the SSSRM

In this chapter, the machine description, converter topology description, self-starting scheme, and the principle of the four-quadrant operation will be presented.

2.1 Machine Description

2.1.1 Machine Dimensions

The machine configuration used in this thesis is from [5]. It is a 4:4:4 SRM, which means it has four main stator poles, four rotor poles, and four auxiliary stator poles. The machine was designed to operate at a rectified dc voltage from 120V AC with up to 8 Amps, 2 Amps in main stator winding and auxiliary winding, respectively. Its maximum rated speed is 10,000rpm. The mechanical dimensions of the rotor and the stator are shown in figure 2.1.

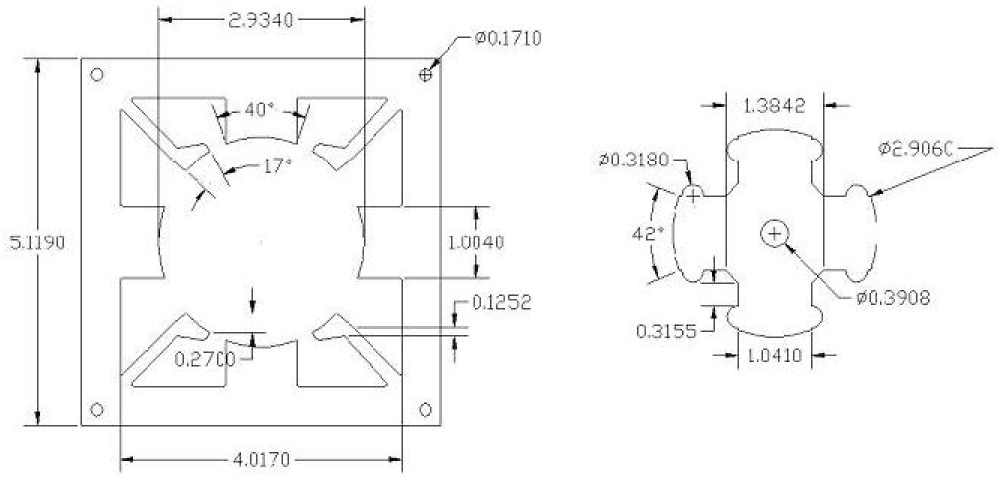


Figure 2.1 4:4:4 SSSRM rotor and stator dimensions

2.1.2 Machine Winding

Figure 2.2 shows the main stator poles winding and the auxiliary stator poles winding. All of the main windings are connected in series. The winding are connected to the converter circuit to produce main torque. The auxiliary stator poles are connected in series of two sets with only one set connected to the converter circuit to produce the auxiliary torque for self-starting.

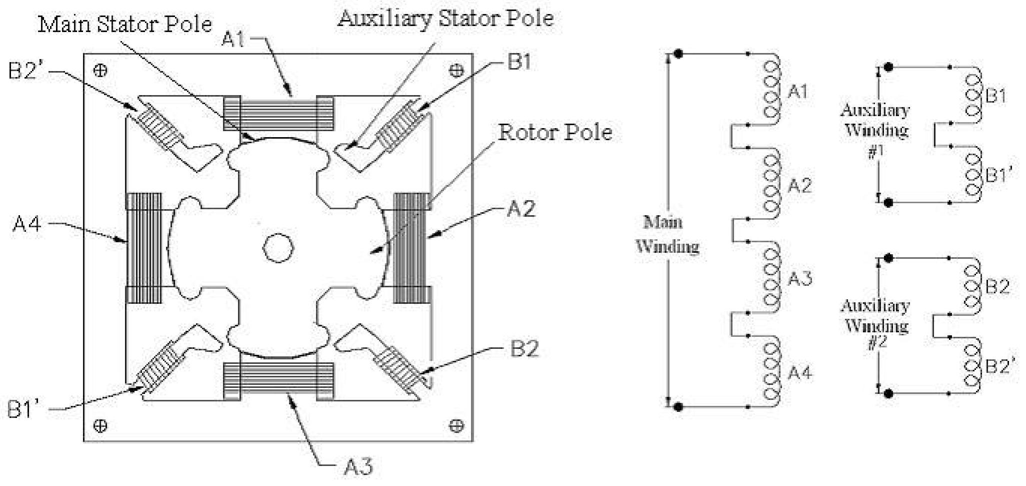


Figure 2.2 Winding configurations

As can be seen, this machine is a two phase SRM construction. In the converter configuration, this machine is controlled by only one controllable switch and two diodes. This machine and converter configuration is called a single switch based switched reluctance machine system (SSSRM).

2.1.3 Finite Element Analysis Results

The machine electromagnetic torque and flux linkages characteristics can be derived by using finite element analysis (FEA) software. FEA analysis was performed for a machine that had the same dimensions as the experimental prototype. These characteristics were used for dynamics simulation.

Air gap torques and inductance of the main winding and those of the auxiliary winding were extracted from the FEA results and are shown in figure 2.3, figure 2.4.

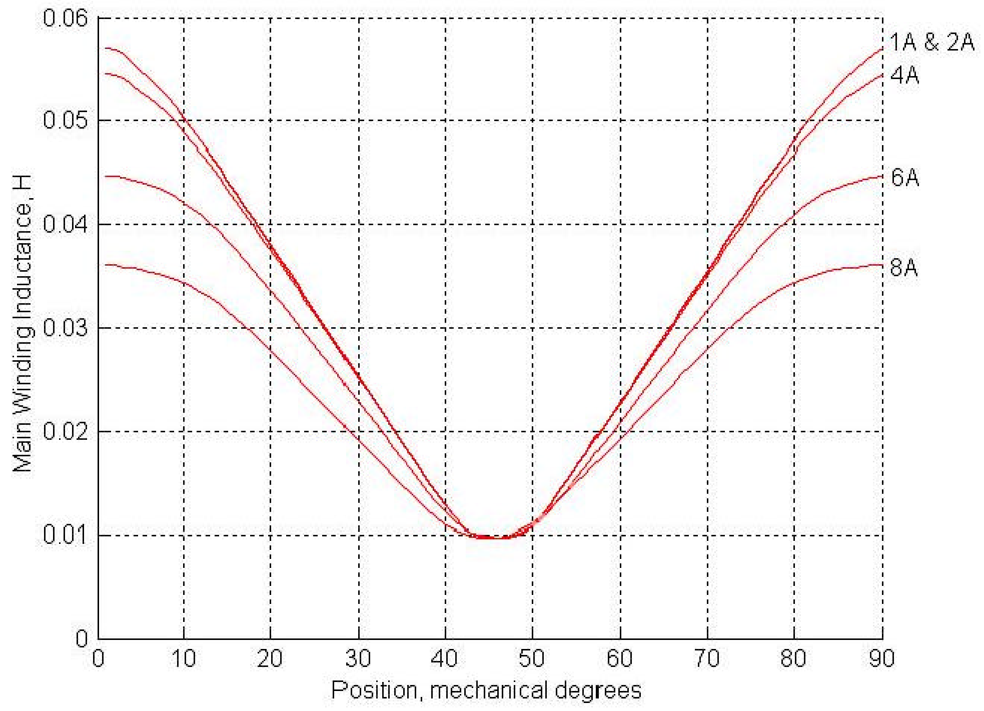


Figure 2.3 Inductance profile of the main winding

From the figure 2.3, the aligned position is 0° and the unaligned position is 45°. The region of the negative inductance slope is between the aligned position and the unaligned position. Otherwise, the region of the positive inductance slope is between the unaligned position and aligned position. The polarity of the inductance slope affects the polarity of the torque produced by the main winding current.

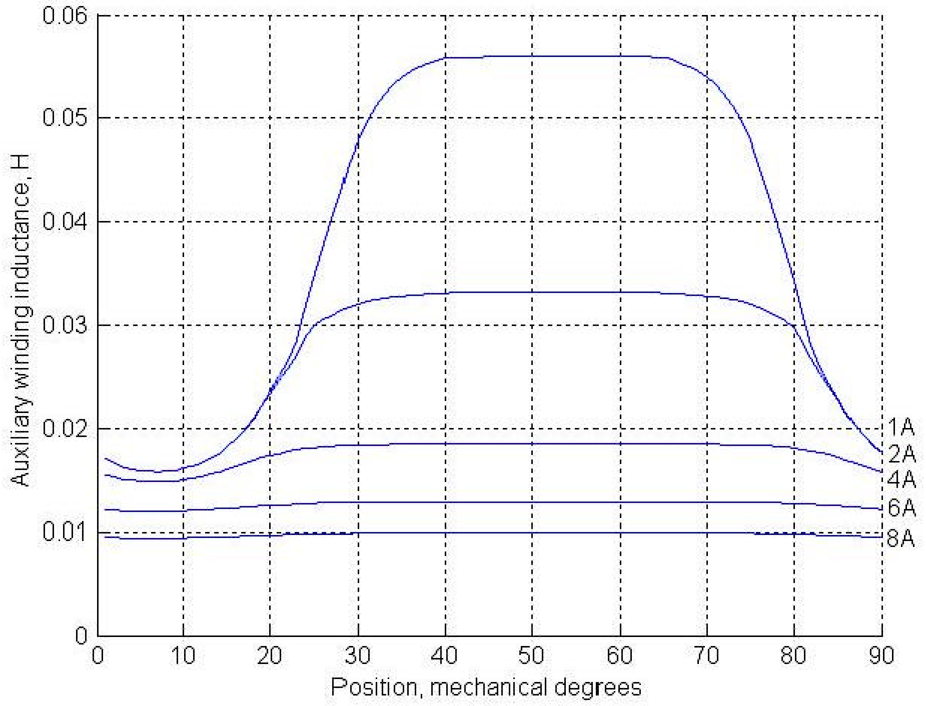


Figure 2.4 Inductance profile of the auxiliary winding

From the figure 2.4, the aligned position of the auxiliary stator pole is 8° and the unaligned position is 49°.

2.2 Converter Description

2.2.1 Converter Topology

To achieve self-starting and realize a cost effective variable speed SSSRM drive, an appropriate power converter topology is required with proper algorithms to control speed. The power converter is shown in figure 2.5. Described here is one of the motor drive systems was invented by Prof. Krishnan Ramu[6] which utilizes the one-controllable-switch.

As stated before, the majority of the torque is produced by the main winding set in the two phase SRM.

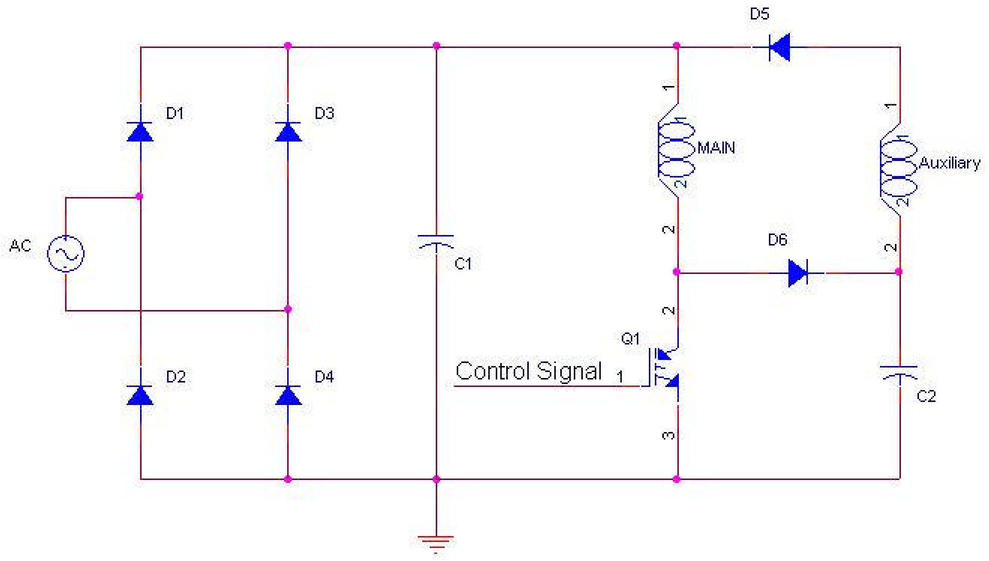


Figure 2.5 Schematic of the power converter

The power converter obtains its dc link either from a single phase (as shown in figure 2.5) or from a three phase ac line through appropriate rectifier and an electrolytic capacitor. The drive consists of a controllable switch Q_1, two diodes D_5, D_6 and a capacitor C_2. D_5 can be optional.

2.2.2 Converter Operation

The converter operation consists of two modes with respect to switch on and off. In the first mode, the switch is on, shown in figure 2.7. The current from dc link voltage, V_{dc} , flows to the main winding and switch. The main winding is controlled with the controllable switch directly.

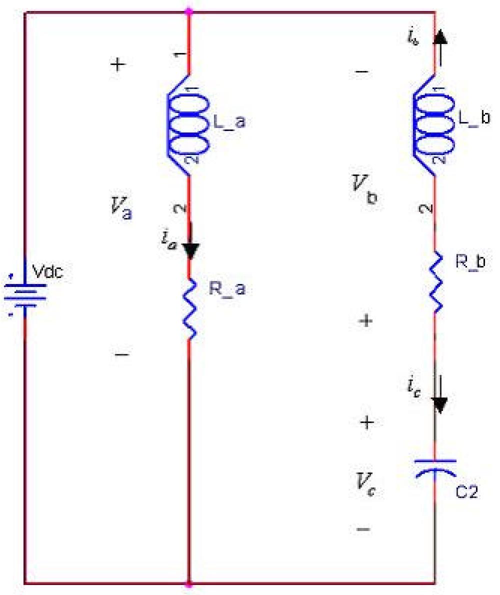


Figure 2. 6 Converter operation: switch on

As seen in figure2.6, using Kirchhoff's laws, we can derive the following:

$$V_a = V_{dc} = R_a i_a + \frac{dl_a}{dt} \tag{2.1}$$

where, V_a is the voltage of the main winding, V_{dc} is the dc link voltage, R_a is the resistance of the main winding, i_a is the current of the main winding, l_a is the flux linkage of the main winding given by:

$$l_a = L_a(q, i_a) \cdot i_a \tag{2.2}$$

where, $L_a(q,i_a)$ is the inductance dependent on the rotor position and the current of the main winding, i_a. The voltage equation of the main winding, then, is

$$V_a = R_a i_a + L_a(q,i_a)\frac{di_a}{dt} + i\frac{dq}{dt}\frac{dL_a(q,i_a)}{dq} \tag{2.3}$$

Using the Kirchhoff's laws, the voltage equation of the auxiliary winding is obtained as following:

$$V_b = V_c - V_{dc} = R_b i_b + \frac{d l_b}{dt} \tag{2.4}$$

where V_b is the voltage of the auxiliary winding, V_c is the voltage of the capacitor, C2, R_b is the resistance of the auxiliary winding, l_b is the flux linkage of the auxiliary winding given by:

$$l_b = L_b(q,i_b)\cdot i_b \tag{2.5}$$

where, $L_b(q,i_b)$ is the inductance dependent on the rotor position and the current of the auxiliary winding, i_b. The voltage equation of the auxiliary winding, then, is

$$V_b = R_b i_b + L_b(q,i_b)\frac{di_b}{dt} + i_b\frac{dq}{dt}\frac{dL_b(q,i_b)}{dq} \tag{2.6}$$

The voltage of the capacitor, C2, is obtained as following:

$$V_c = \frac{1}{c}\int i_c + v_o(t_{off}) \tag{2.7}$$

where, i_c is the current of the capacitor, C_2, v_o is the initial voltage of the capacitor, C_2, t_{off} is the last time at turning the switch off.

The currents through the main and auxiliary winding should also satisfy the following condition as for the converter topology:

$$i_a \geq 0,\ i_b \geq 0 \tag{2.8}$$

$$i_c = -i_b \tag{2.9}$$

In the second mode, as shown in figure 2.7, the state shows that the switch was turned off. The currents are allowed to free wheel.

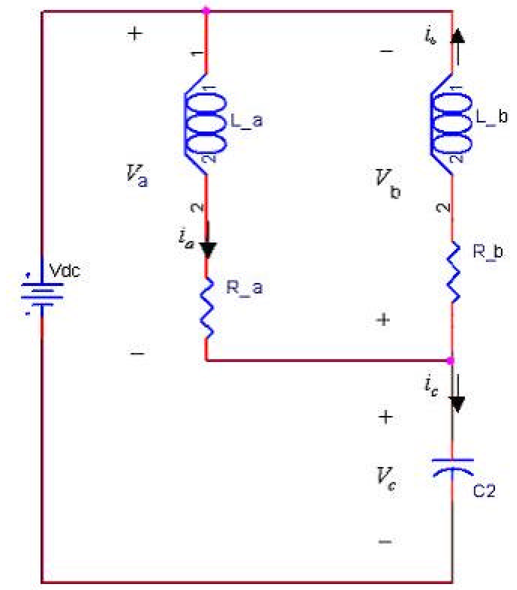


Figure 2.7 Converter operation: switch off

Using the Kirchhoff's laws again, we derive the following for the second mode:

$$V_a = V_{dc} - V_c = R_a i_a + \frac{d\,l_a}{dt} \tag{2.10}$$

The voltage equation of the auxiliary winding is obtained as following:

$$V_a = V_c - V_{dc} = R_a i_a + \frac{d\,l_a}{dt} \tag{2.11}$$

The voltage of the capacitor, C_2, is obtained as following:

$$V_c = \frac{1}{c}\int i_c + v_o(t_{on}) \tag{2.12}$$

where, i_c is the current of the capacitor, C_2, v_o is the initial voltage of the capacitor, C_2, t_{on} is the last time at turning the switch on

.

For flowing the current through the auxiliary winding, the condition should be satisfied as following:

$$i_a \geq 0\,,\ \ i_b \geq 0 \tag{2.13}$$

$$i_c = i_a - i_b \tag{2.14}$$

When the switch is turned off, the capacitor C_2 is charged. Because the capacitor C_2 is connected the inductance of the main winding, the voltage of the capacitor oscillates.

In both operations with the switch on and off, the main winding is involved. From this, it is seen that a controllable switch also controls the current in the auxiliary winding indirectly. It results in one controllable switch controlling the current in the main and auxiliary phases. Alternately one can perceive D_6 and C_2 as a snubber circuit for transferring the energy away from the main winding. The snubber circuit can commutate current during switch turn off and also help drive current through the auxiliary winding.

The mechanical equation is obtained as following:

$$J\frac{dw_m}{dt} + Bw_m = T_e - T_l \tag{2.15}$$

$$T_e = T_a - T_b \tag{2.16}$$

where, J is the rotor and load inertia, B is the friction coefficient of the motor and the load, T_e is the electromagnetic torque, T_l is the load torque, T_a is the torque produced from the main winding, and T_b is the toque produced from the auxiliary winding .

2.3 Self-starting scheme

During aligned rotor position corresponding to main stator poles, main stator poles do not produce any torque. Therefore, ways to start the machine at all rotor positions have to be developed.

The idea of self-starting is invented by Prof. Krishnan Ramu. With proper machine configuration and power converter, a suitable control algorithm is very important to realize the self-starting with SSSRM. The idea is to produce torque by driving current through the auxiliary winding, whatever the rotor was in the aligned position, unaligned position, or in between. The aligned position is the position of maximum inductance. The unaligned position is the position of minimum inductance.

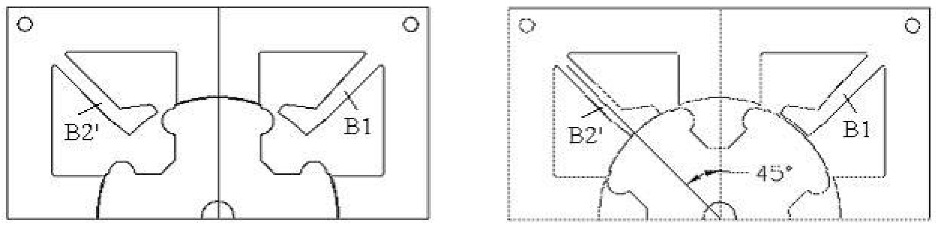
When the rotor is placed between the aligned position and the unaligned position, applying a pulse signal can produce starting torque. However, it still has the starting problem if the rotor aligns with the stator pole exactly. A single start pulse can produce torque from the auxiliary winding, when the energy in the main winding is transferred to the auxiliary winding, but usually it is not enough to move the rotor completely out of the aligned position.

The torque produced from the phase current with respect to the rotor position can be derived as below :

$$T_e = \frac{1}{2} i^2 \frac{dL(\theta, i)}{d\theta} \tag{2.17}$$

The torque from the main stator pole cannot be produced at the aligned or unaligned position, because the term of the derivative inductance is 0.

For the SSSRM, the aligned position of the main stator pole are 0°, 90°, 180°, and 270°. And the unaligned positions are 45°, 135°, 225°, and 315°. The number of rotor poles is 4. So the aligned / unaligned position are repeated every 0° and 45° shown in figure 2.8. However, the aligned position of the auxiliary stator pole is 8° and the unaligned position of the auxiliary stator pole is 49°. Recall that only one set of auxiliary winding is used (B1). The auxiliary pole(B2') do not contribute to the self-starting algorithm.



(a) Aligned rotor position (b) Unaligned rotor position

Figure 2.8 Aligned and unaligned rotor positions

Using the SSSRM converter topology, the auxiliary winding can be energized with a pulsating signal to the main winding. After this energy transfers the current of the auxiliary winding produces torque to move the rotor. Even if the rotor is placed in the aligned or unaligned position with respect to the main stator pole. Thus, by applying a pulse to the SSSRM, the aligned equilibrium point can be broken. By applying another pulse, the SSSRM can produce torque by the current of the main winding. To realize self-starting with SSSRM, the On-time and Off-time is inserted. Detail algorithm will be explained in chapter 4.

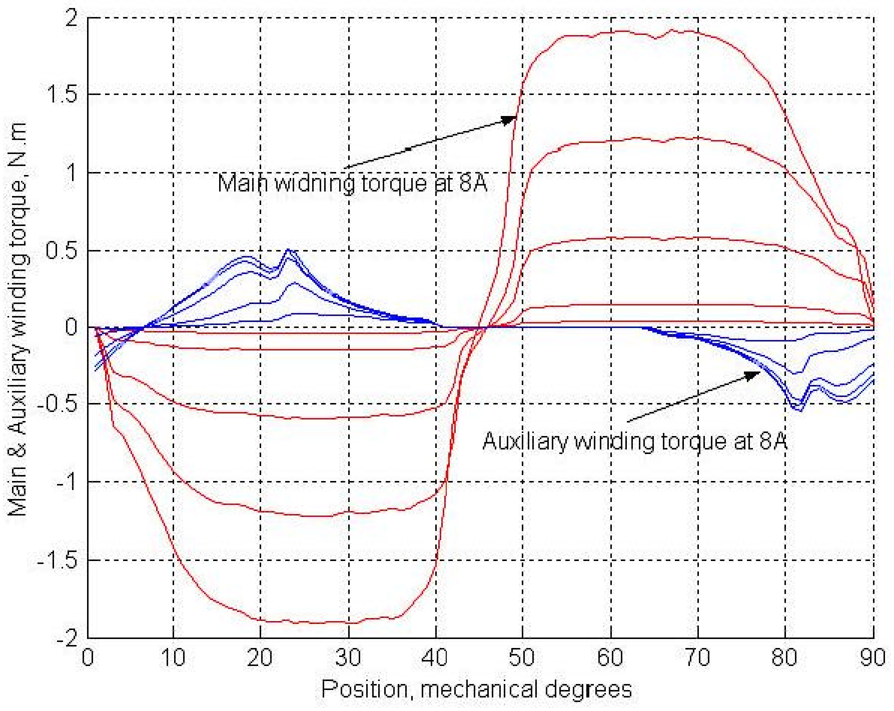


Figure 2.9 Torque profile of the main winding

From the torque profile of the main winding and the auxiliary winding, at 0° position, the torque from the main winding is 0, but the torque of the auxiliary winding can be produced negative torque. At 45°, figure 2.9 shows the torque of the main winding and the auxiliary winding looks like zero, but if the figure was zoomed in, torque of the auxiliary winding is small value.

The torque, produced by the current of the auxiliary winding is much smaller than torque produced by the current of the main winding. Therefore, torque from the auxiliary winding can only make the rotor move a little bit, but it is enough to break the equilibrium point for the aligned position with the main stator pole and the rotor pole.

2.4 The Principle of the Four-Quadrant Operation with SSSRM

There are four quadrants that exist with respect to speed polarity vs. torque polarity. From the table 2.1, the each quadrant can be defined with depending on the speed polarity and torque polarity.

Function	Quadrant	Speed	Torque
Forward Motoring (CW Motoring)	I	+	+
Forward Regenerating (CW Braking)	IV	+	–
Reverse Motoring (CCW motoring)	III	–	–
Reverse Regenerating (CCW Braking)	II	–	+

Table 2.1 Four-quadrant operation table [2]

For convenience, forward motoring is called CW motoring, reverse motoring is called CCW motoring, and regenerating can be called braking.

Figure 2.10, shown inductance profile and torque polarity with speed polarity, present a waveform, derived table2.1.

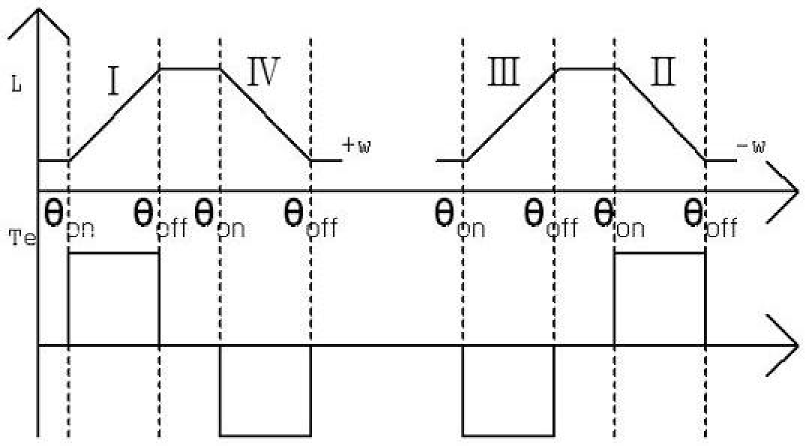


Figure 2.10 Four-quadrant operation with firing angle [1]

2.4.1 Clockwise (CW) Direction Motoring

In order to achieve CW motoring, the stator winding should be excited when the rotor is moving from unaligned to the aligned position. Because the region between the unaligned(45°) to aligned position(90°) is the region of the positive inductance slope. Positive torque is produced by firing the PWM gate signal at the positive inductance slope region shown in figure 2.10 region I. Assuming that the rotor poles pass unaligned position (almost in alignment with respect to the auxiliary stator poles) of the main phase winding, the main phase winding is energized when such a position is detected. When the rotor poles have nearly reached unaligned position with the main poles, the current in the main phase is turned off. The machine spins in the clockwise (CW) direction during this time.

2.4.2 Clockwise (CW) Direction Regenerating

The CW braking, on the contrary, is achieved by excitation of the stator windings when the rotor moves from the aligned position towards the unaligned position. Negative torque will be produced by firing the PWM gate signal in the negative inductance slope region shown in figure 2.10 region IV. During this time, the kinetic energy in the machine is transferred to the dc link source via the auxiliary winding. Note that the machine is still rotating in the CW direction, but its speed rapidly decreases toward zero.

2.4.3 Change Rotor Direction From CW to CCW

During speed reversal, the controller goes into the CW braking mode as explained in the paragraph above. That brings the rotor to standstill position. Instead of waiting for the absolute standstill position, continuously energize the main winding during the aligned to unaligned rotor region. This not only slows the rotor to standstill fast but also provides an opportunity to revers e the direction. The rotor poles come to a stop between the main and auxiliary pole. We apply one pulse to rotate in the reverse direction.To apply a pulse of self-starting, while changing the rotation direction, the delay time should be considered. By the output signal of the position sensor, the controller checks the rotor position, and determines the direction. Multiple pulses are applied, but the direction of the rotor can not be assured. Therefore, it is necessary to determine the instant the rotor of the machine is ideally positioned for reversal, and apply a pulse signal to change from CW regeneration to CCW motoring. Detail algorithm is explained in chapter 4.

2.4.4 Counter Clockwise (CCW) Direction Motoring and Braking

Once the rotor is moving in CCW direction, the CCW motoring is realizedwith the PWM firing scheme as the CW braking sequence. It makes rotor to rotate CCW shown in figure 2.10 region III. The CCW braking sequence is same as the CW motoring sequence shown in figure 2.10 region II.

Chapter 3 SSSRM modeling for the dynamic simulation

3.1 SSSRM modeling

The SRM is a nonlinear system as there is a term in the voltage equation containing the product of the rotor speed and the phase current. Inductance and torque vary depending on the phase current and the rotor position. Because of this, the linear control system theory cannot be applied easily to the controller design.

SSSRM nonlinear modeling should consider the nonlinearities of the inductance and torque. The machine magnetic characteristics were obtained from the finite element analysis tool (Maxwell 2D software) and used in simulation.

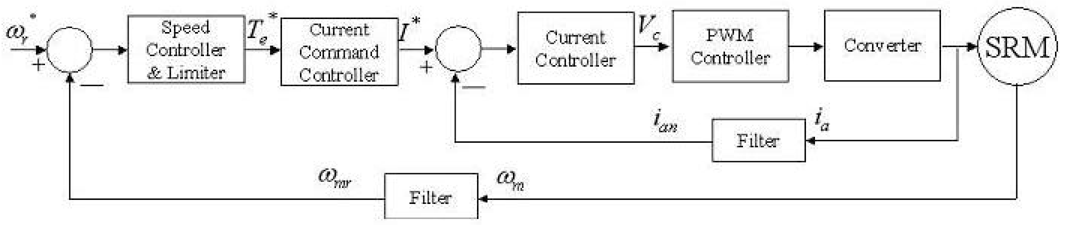


Figure 3.1 Closed-loop, speed-controlled SRM drive system

The state variables are defined from the block diagram of the speed controlled SSSRM drive system, shown in figure 3.1, and explained using a flow chart how the model can be simulated. [1]

3.1.1 The motor equations

The SSSRM voltage equations and mechanical equations, derived in chapter 2, are given below with turning the switch on and off.

Switch On :

$$V_a = V_{dc} = R_a i_a + L_a(q, i_a)\frac{di_a}{dt} + i_a w_m \frac{dL_a(q, i_a)}{dq} \tag{3.1}$$

$$V_b = V_c - V_{dc} = R_b i_b + L_b(q, i_b)\frac{di_b}{dt} + i_b w_m \frac{dL_b(q, i_b)}{dq} \tag{3.2}$$

$$V_c = \frac{1}{c}\int i_c dt + V_c(t_{off}) \tag{3.3}$$

Switch Off :

$$V_a = V_{dc} - V_c = R_a i_a + L_a(q, i_a)\frac{di_a}{dt} + i_a w_m \frac{dL_a(q, i_a)}{dq} \tag{3.4}$$

$$V_b = V_c - V_{dc} = R_b i_b + L_b(q, i_b)\frac{di_b}{dt} + i_b w_m \frac{dL_b(q, i_b)}{dq} \tag{3.5}$$

$$V_c = \frac{1}{c}\int i_c dt + V_c(t_{on}) \tag{3.6}$$

$$J\frac{dw_m}{dt} + Bw_m = T_e - T_l \tag{3.7}$$

where,

$$w_m = \frac{dq}{dt} \tag{3.8}$$

The state variables are defined as

$$x1 = q \tag{3.9}$$

$$x2 = w_m \tag{3.10}$$

$$x5 = i_a \tag{3.11}$$

$$x8 = i_b \tag{3.12}$$

The motor equations in terms of the state variables are

$$\dot{x}1 = \frac{dq}{dt} = w_m = x2 \tag{3.13}$$

$$\dot{x}2 = \frac{dw_m}{dt} = \frac{T_e}{J} - \frac{T_l}{J} - \frac{Bw_m}{J} = \frac{T_e}{J} - \frac{T_l}{J} - \frac{Bx2}{J} \tag{3.14}$$

$$\dot{x}5 = \frac{di_a}{dt} = \frac{V_a}{L_a} - \frac{R_a i_a}{L_a} - \frac{i_a w_m}{L_a} = \frac{V_a}{L_a} - \frac{R_a x5}{L_a} - \frac{x5x2}{L_a} \tag{3.15}$$

$$\dot{x}8 = \frac{di_b}{dt} = \frac{V_b}{L_b} - \frac{R_b i_b}{L_b} - \frac{i_b w_m}{L_b} = \frac{V_b}{L_b} - \frac{R_b x8}{L_b} - \frac{x8x2}{L_b} \tag{3.16}$$

3.1.2 Speed Feedback

The feedback transducer and filter can be presented as shown in figure 3.2.

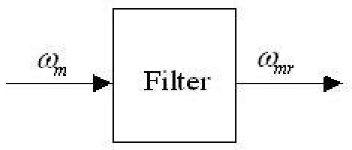


Figure 3.2 Speed-feedback filter

The transfer function of the speed feedback filter is

$$G_w(s) = \frac{w_{mr}(s)}{w_m(s)} = \frac{H_w}{1 + sTw} \tag{3.17}$$

In time domain, equation (3.17) can be rearranged by letting

$$x3 = w_{mr} \tag{3.18}$$

and then

$$\dot{x}3 = \frac{1}{T_w}(H_w x2 - x3) \tag{3.19}$$

3.1.3 Speed Controller and Current Command Controller

The speed controller block diagram is shown in figure 3.3.

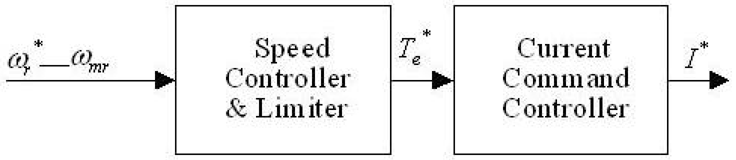

$\omega_r^* - \omega_{mr}$
Speed Controller & Limiter
T_e^*
Current Command Controller
I^*

Figure 3.3 Block diagram of the speed controller with the current command controller

The transfer function of the proportional-plus-integral controller is given as

$$G_s(s) = K_{ps} + \frac{K_{is}}{s} \tag{3.20}$$

where, K_{is} is the integral gain of the speed controller, K_{ps} is the proportional gain of the speed controller. Letting

$$\dot{x}4 = w_r^* - w_{mr} = w_r^* - x3 \tag{3.21}$$

The torque command signal is derived as

$$T_e^* = K_{ps}(w_r^* - x3) + K_{is}x4 \tag{3.22}$$

For the safe operation of the drive system, the torque command should be limited to allowable limits determined by the converter and motor peak capabilities. Letting the maximum allowable torque be $T_{\max}$. The torque command limit is integrated into the simulation as

$$-T_{\max} \le T_e^* \le T_{\max} \tag{3.23}$$

The current command is derived by

$$i^* = \frac{T_e^*}{T_{\max}} \times i_{\max} \tag{3.24}$$

3.1.4 Current Feedback

In the SSSRM drive system, controller directly controls the current of the main winding between the current of the main winding and the current of the auxiliary winding. Because of this, the feedback current is the current of the main winding, i_a.

The current of the main winding was defined from the equation (3.11).

$$\dot{x5} = \frac{V_a}{L_a} - \frac{R_a x5}{L_a} - \frac{x5x2}{L_a} \tag{3.25}$$

The feedback transducer of the current feedback can be presented as shown in figure 3.4.

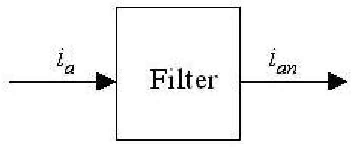


Figure 3.4 Current feedback filter

The transfer function of the current feedback filter is

$$G_c(s) = \frac{i_{an}(s)}{i_a(s)} = \frac{H_c}{1+sT_c} \tag{3.26}$$

In time domain, equation (3.26) can be rearranged by letting

$$x6 = i_{an} \tag{3.27}$$

and then,

$$\dot{x6} = \frac{1}{T_c}(H_c x5 - x6) \tag{3.28}$$

3.1.5 Current Controller

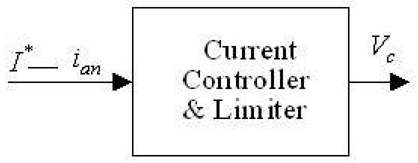


Figure 3.5 Block diagram of the current controller

The transfer function of the proportional-plus-integral controller is given as

$$G_c(s) = K_{pc} + \frac{K_{ic}}{s} \tag{3.29}$$

Letting

$$\dot{x}7 = I^* - i_{an} = I^* - x6 \tag{3.30}$$

The control voltage, V_c, is derived as

$$V_c = K_{pc}(I^* - x6) + K_{ic}x7 = K_{pc}\,\dot{x}7 + K_{ic}x7 \tag{3.31}$$

where, K_{ic} is the integral gain of the current controller, K_{pc} is the proportional gain of the current controller.

3.1.6 PWM controller

The control voltage, V_c, has to be limited to a value of maximum control voltage, V_{cm} corresponding to the maximum current of the main winding. Hence, the limiter is prescribed as

$$0 \le V_c \le V_{cm} \tag{3.32}$$

The PWM signal is generated by combining the ramp signal and the control voltage signal, V_c. PWM on-time signal is produced if the control voltage is greater than the ramp(carrier) signal; PWM off-time signal is generated when the control signal is less than the ramp signal. Then the voltage of the main winding and the voltage of the auxiliary winding can be determined with respect to the PWM on- and off- times, discussed for power converter topology in chapter 2.

The voltage of the capacitor, C2, is defined as

$$x9 = V_{cap} \tag{3.33}$$

$$\dot{x}9 = \frac{dV_{cap}}{dt} = \frac{i_c}{C2} \tag{3.34}$$

3.2 Dynamic Simulation of the SSSRM

In this session, SSSRM dynamic simulation is presented using states equations and the motor parameters.

3.2.1 State equations and Numerical Solution

Equations (3.1) through (3.34) constitute the state equations as below:

$$\dot{x}1 = x2 \tag{3.35}$$

$$\dot{x}2 = \frac{T_e}{J} - \frac{T_l}{J} - \frac{Bx2}{J} \tag{3.36}$$

$$\dot{x}3 = \frac{1}{T_w}(H_w x2 - x3) \tag{3.37}$$

$$\dot{x}4 = w_r^{\ *} - x3 \tag{3.38}$$

$$\dot{x}5 = \frac{V_a}{L_a} - \frac{R_a x5}{L_a} - \frac{x5x2}{L_a} \tag{3.39}$$

$$\dot{x}6 = \frac{1}{T_c}(H_c x5 - x6) \tag{3.40}$$

$$\dot{x}7 = I^* - x6 \tag{3.41}$$

$$\dot{x}8 = \frac{V_b}{L_b} - \frac{R_b x8}{L_b} - \frac{x8x2}{L_b} \tag{3.42}$$

$$\dot{x}9 = \frac{i_c}{C2} \tag{3.43}$$

3.2.2 Motor Parameters

For the SSSRM dynamic simulation, the motor parameters were defined as below:

- DC link voltage, $V_{dc} = 120 \times \sqrt{2} \cong 170V$

- The resistance of the main winding, $R_a = 6.2\Omega$

- The resistance of the auxiliary winding, $R_b = 9.2\Omega$

- Capacitance of the capacitor, $C2 = 4.7uF$

- Maximum current, $I_{\max} = 8A$

- Friction Coefficient, $B = 0.000023 N.m.s/rad$

- Moment of Inertia, $J = 0.0004 kgm^2$

- Maximum Speed voltage, $w_{\max\_volt} = 3.3V$

- Maximum speed, $w_{\max} = 12000 rpm$

- Maximum control voltage, $V_{cm} = 3.3V$

- Speed transducer gain, $H_w = \frac{w_{\max\_volt}}{w_{\max\_rps}} = \frac{3.3}{1256.6} = 0.002626$

- Current transducer gain, $H_c = \frac{V_{cm}}{I_{\max}} = \frac{3.3}{8} = 0.4125$

- PWM carrier frequency, $f_c = 10kHz$

3.2.3 Dynamic Simulation Procedure

The dynamic simulation of the state equations is achieved by numerical integration. A flowchart for the dynamic simulation is given in figure 3.6.

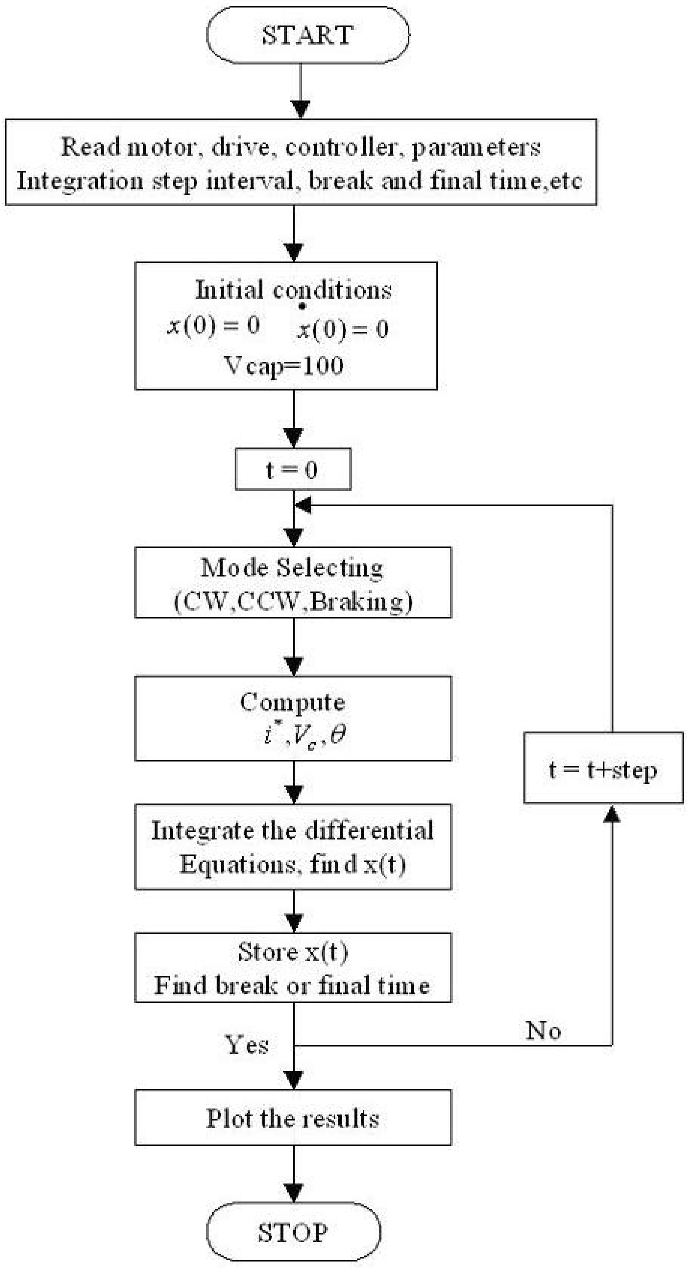


Figure 3.6 Flowchart for the dynamic simulation

The dynamic simulation procedure is below:

Step 1: starts to read parameters and load FEA data, and then sets up the initial conditions. V_{cap}, the voltage of the capacitor C2, is set to initial value when the switch turned off. The voltage across this capacitor should be considered for applying the self-starting algorithm to the dynamic simulation. This initial condition causes to produce torque at aligned rotor position.

At standstill, the back emf terms should be excluded for the computation of the main winding voltage and the auxiliary winding voltage. The controller gains were selected by nominal va lues from trial and error. Advance firing angle and advance commutation angle should be considered for removing the negative torque during high speed motoring operation.

Step 2 : compute current command, control voltage, and theta from the current control loop and the speed control loop.

Step 3 : integrate the differential equations and store data

Step 4 : check the break time or final time to set up the motor control mode. The mode may be changed with following sequence: CW motoring -> CW braking -> CCW motoring -> CCW braking. And go to step 2.

Step 5 : when the time become simulation final time, plot the stored data.

The dynamic simulation is achieved using matlab simulation software. The source code was added to the appendix E.

3.2.4 Dynamic Simulation Results

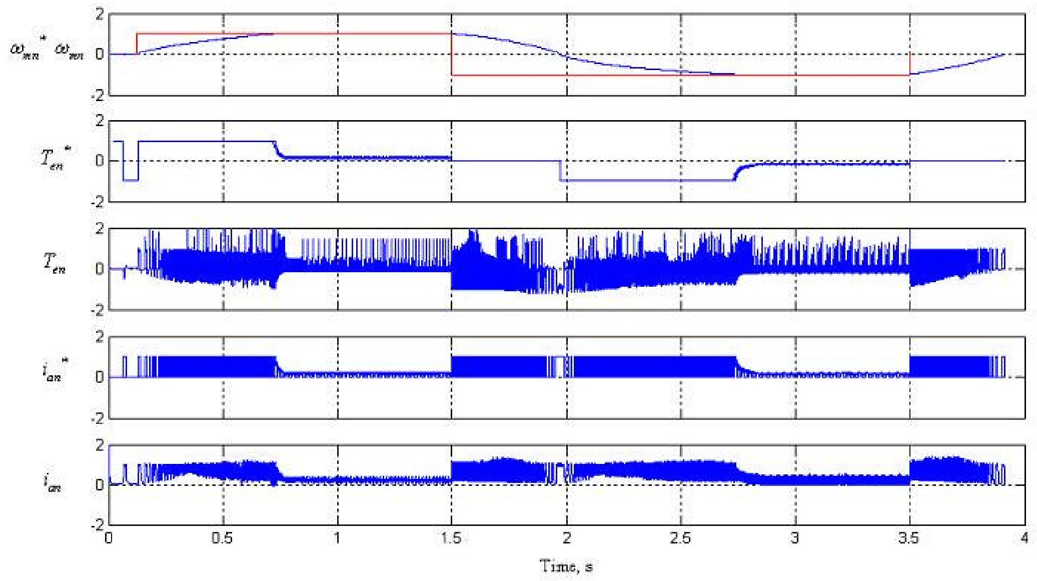


Figure 3.7 Dynamic simulation of a four-quadrant SSSRM drive for a step command in bipolar speed reference, in normalized units

Figure 3.7 shows the performance of a four-quadrant drive system. The reference speed is step commanded. The four-quadrant operation has done with following the reference speed : CW motoring -> CW braking -> CCW motoring - > CCW braking. During the braking mode, the speed command is not considered because the drive system does not control the rotor speed. Only the current limitation was considered. The quadrants of operation are also plotted in the figure 3.7, to appreciate the correlation of speed, torque, current and voltage in the motor drive system.

Chapter 4. Hardware Implementation

In this chapter, the hardware implementations are presented: Power Converter description, the DSP Controller (TI DSP) hardware and the software algorithms descriptions, and the signal interface modules description. The block diagram of the SSSRM Drive System, seen in figure 4.1, shows the overall system architecture.

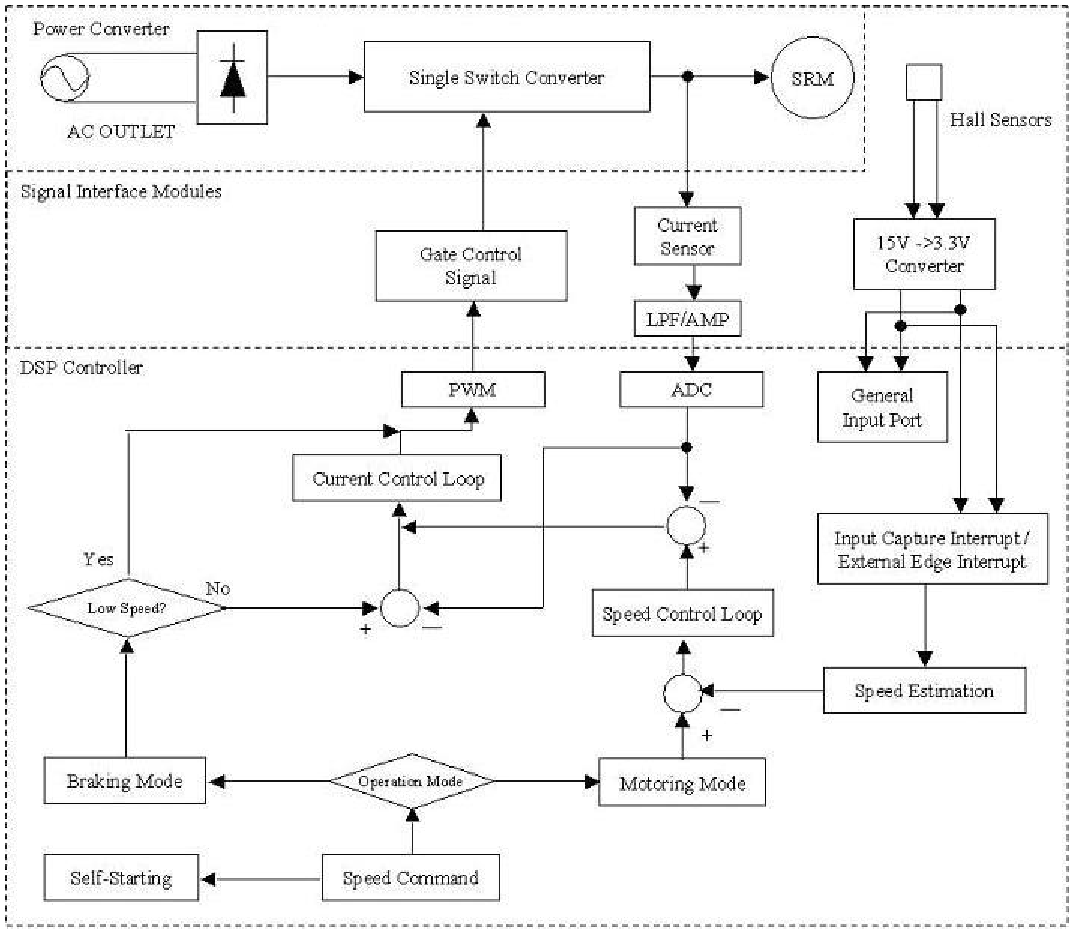


Figure 4.1 Block diagram of the SSSRM drive system

The whole hardware system is shown in figure 4.2. Each module will be explained in this section.

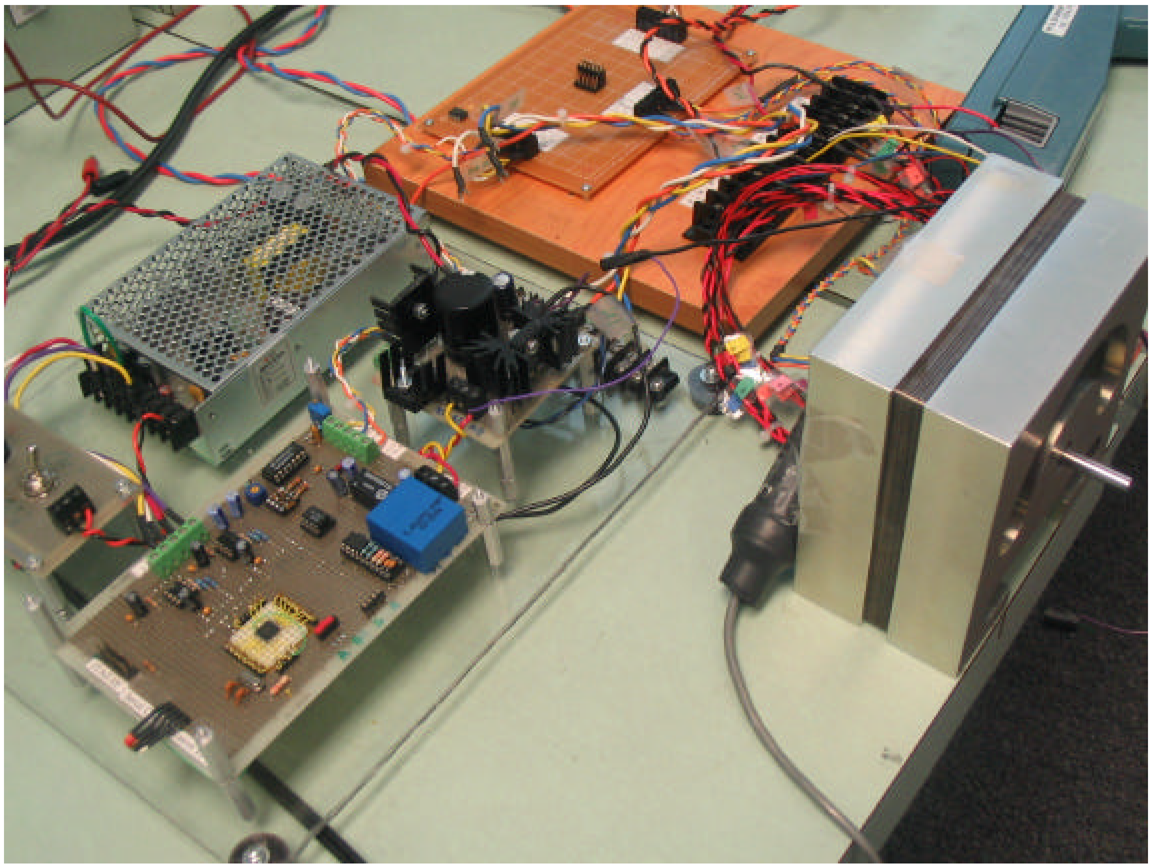


Figure 4.2 Prototype board for the SSSRM drive system

4.1 Power Converter

A full wave bridge rectifier was used to supply the DC Power,. As shown in figure 4.3, the rest of the power converter consists of one switch, two diodes, and one capacitor. The detail description of the power converter topology was presented in chapter 2.

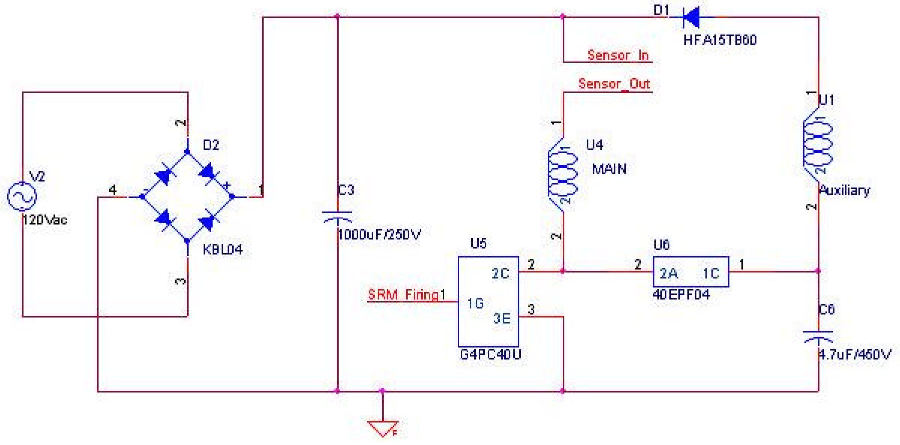


Figure 4.3 Schematic of the power converter

4.1.1 Power Electronic Components Selection

From the inductance profile characteristics of SRM, the switch can be turned on only for a portion of the rotor rotation. Therefore, with 50% utilization, the average current for the bridge rectifier is 4A. The KBL04 series was chosen for the rectifier diode, which has a maximum dc current of 8A and an one- cycle surge current of 210A.

The capacitor for the dc link was selected according to the following procedure:

Supplied energy to the circuit in one cycle $= V_{DC} i_{av} t = \frac{1}{2} C_{DCLink} V_{DC}^{2}$ (4.1) where, t =dT is turn on time when the current is flowing and i_{av} is the average current at the winding of one period.

$$C_{DCLink} = \frac{2i_{av}dT}{V_{DC}} = \frac{8T}{\sqrt{2 \cdot 120}} = 0.04714\,T \tag{4.2}$$

Since the number of rotor poles is 4, then T = time to complete $90°$ of a revolution with assuming a base speed of 5000 rpm.

$$0.25rev = 5000\,(rev/\min) \cdot T \;\; => \;\; T = 3ms \tag{4.3}$$

$C_{DCLink} = 0.04714 \cdot 0.003 = 141.4uF$ => For a conservative design, 560uF, 250V

For the capacitor value for the snubber circuit, 4.7uF / 250V was selected. [11]

For switching current at 8A, G4PC40, with a rated switching frequency at 8 – 40kHz and 20A current, was selected. For the fast recovery rectifier, 40EPF04 was selected.

4.1.2 Power Supply Configuration

The configuration of the power supply is shown in figure 4.4. For the prototype of the SPSRM, the commercial SMPS was used to supply the power for the controller's power supply and signal interface modules.

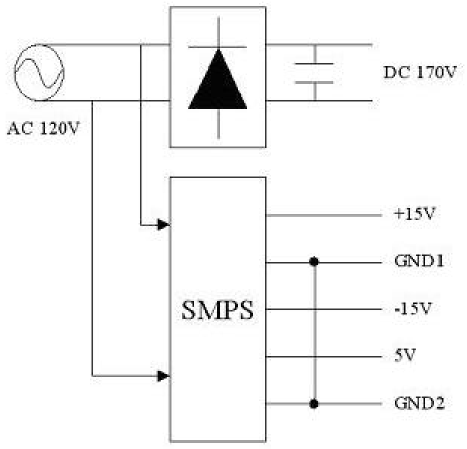


Figure 4.4 Block diagram of the power supply

+15V, -15V were supplied to the LA-25N current sensor, LM2907, and the power source of the hall sensor. 5V was needed for the JTAG interface module and the power source for programming the flash memory of TI DSP.

For 3.3V operation of the TI DSP, the 3.3V regulator (TPS7301), shown in figure 4.5, was used. It also supplied the power to the op-amp for filtering and amplifying the current feedback signal.

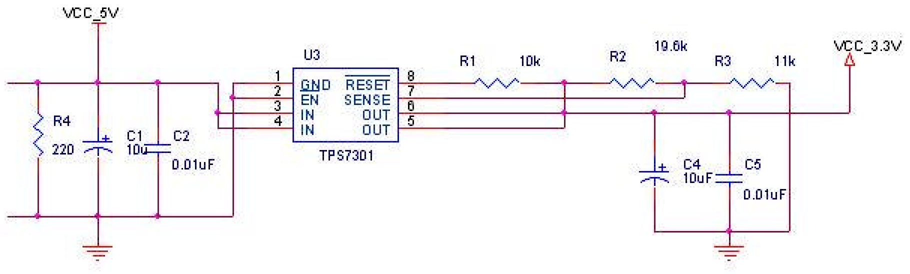


Figure 4.5 Schematic of the 3.3V power supply

4.2 TI DSP(TMS320LF2401) Hardware

4.2.1 TI DSP Overview

To accomplish the proposed four-quadrant operation with SSSRM, TI DSP (TMS320LF2401) was used. It played an important role as a digital controller. The functions of TI DSP are shown in figure 4.6.

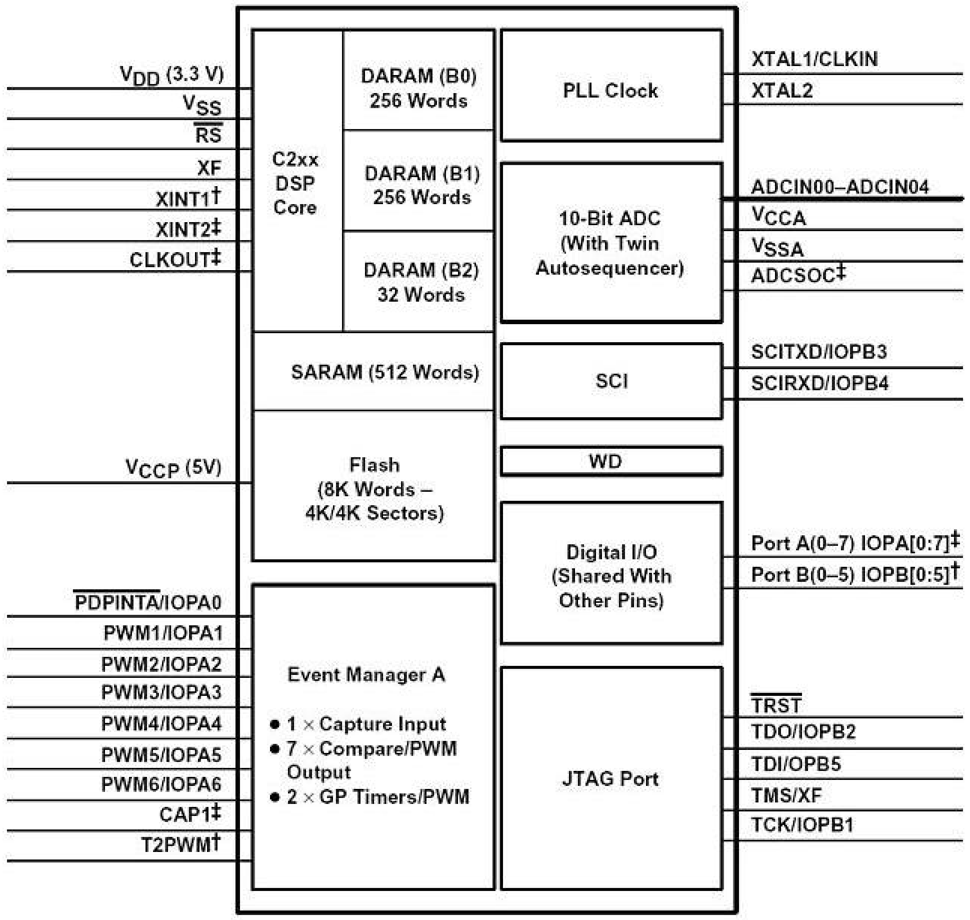


Figure 4.6 Block diagram of the TMS320LF2401 functions

4.2.2 TI DSP Development

For TI DSP development, a Code composer was used for the C compiler providing by TI DSP Corporation. For the basic testing of the function of TI DSP, eZdsp LF2401 board, which was manufactured by Spectrum Digital Corporation, was used. As a download tool, the XDS510PP JTAG Emulator Pod was used.

From the various functions, several were chosen:

- Timer1: setting the PWM period and triggering the ADC starting signal with operating at20kHz of Period and 16bit resolutions.
- ADC: converting the current feedback analog signal to a digital signal with 10bit resolutions and operating at 20kHz.
- JTAG port: downloading the code from a user computer to a TI DSP flash memory
- SCI: communicating using a serial port with an user computer with 19200 baud rate.
- Input Capture: detecting rotor speed whenever the hall sensor produces a rising edge.
- PWM: controlling the switch duty cycle with operating at 20kHz and 16bit resolutions.
- Timer2: counting the timer counter for the rotor speed check with operating a 50us timer counter
- External Interrupt: detecting Hall sensor rising edge and falling edge.

4.2.3 TI DSP Operation Circuit

As shown in figure 4.7, the TI DSP basically operates with only external crystal operating at 10MHz and 3.3V power supply. 5V is provided to VCCP terminal for programming the flash memory.

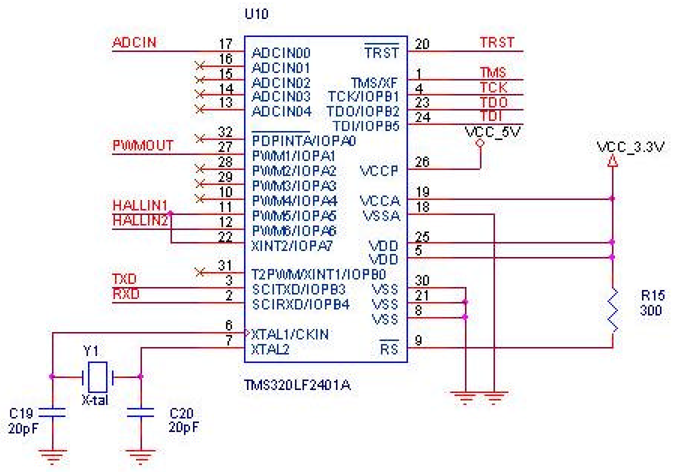


Figure 4.7 TI DSP board circuit schematic

Except VCC and GND terminals, all other terminals were connected with the signal interface modules. Therefore, detailed descriptions about each terminal can be found in the following sections.

4.3 Software Algorithm

All of the controller designs were carried out using digital control algorithms with TI DSP. In this session, four control algorithms are presented: the self-starting algorithm , the current control algorithm, the speed control algorithm, and the four-quadrant control algorithm.

4.3.1 Self-starting Algorithm

The flow chart for the self-starting with multiple pulses is shown in figure 4.8.

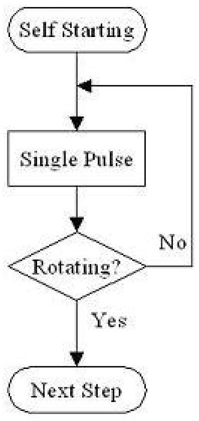


Figure 4.8 Flow chart of the multiple pulse generation

As explained in chapter 2, through pulsing the gate of the switch, excited current generates torque with the main winding or the auxiliary winding in any position. During this time, the PWM function is in standby until rotor starts to rotate. The self-starting algorithm incorporates a while-loop for generating multiple pulses.

In this system, the rotation of the rotor can be determined from the edge signals of the Hall sensors as shown in table 4.1

4.3.2 Hall Sensor Configuration

Two Hall sensors shown in figure 4.9 were placed in two positions not only to obtain the rotor position, but also to obtain the rotor speed and the rotating direction. They are located on the machine outer case 45° apart. Four small permanent magnets were attached on the top of the pole, which is connected to the rotor shaft. The output of the Hall sensors is detected and used for the speed calculating and for determining the rotation direction.

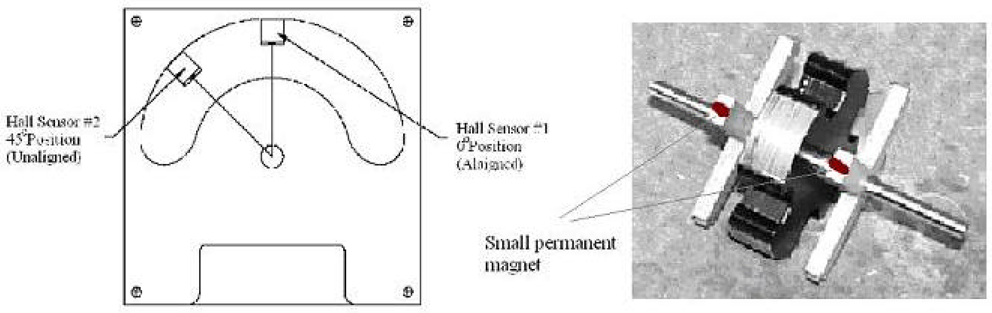


Figure 4.9 Hall sensor configuration

Sensor #	CW Direction		CCW Direction	
Hall Sensor #1	H (↑)	L (↓)	H (↑)	L (↓)
Hall Sensor #2	L	H	H	L

Table 4.1 Truth table for the rotor direction

where, H(↑) is high level rising edge, L (↓) is low level falling edge

4.3.3 Current Control Algorithm

For the current control, the PI control algorithm was used. The current control algorithm is shown in figure 4.10.

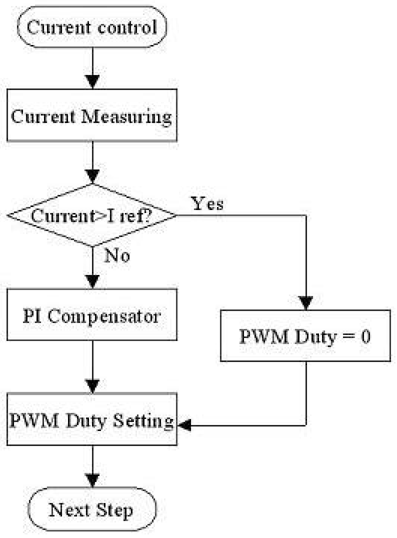


Figure 4.10 Flow chart of the current control algorithm

For the current control, the controller needs a feedback signal. Measuring the main winding current and finding the current error is the basis for the current control. The PWM duty is dependent on the current error. If the current error is less than 0, the PWM duty is determined 0%. If the current error is greater than the maximum current value, the PWM duty is determined with the maximum PWM duty.

There are several methods to measure the current of the main winding: to use a shunt resistor, to use a Hall-effect current sensor, or to use a current transformer. In this research, a Hall-effect type current sensor was used to obtain the current value of the main winding.

4.3.4 Current Feedback with a Shunt Resistor

Following the ohm's law, the current can be calculated from the voltage across the shunt resistor. The configuration of the shunt resistor is shown in figure 4.11. The shunt resistor can be located anywhere, but for convenient filtering and signal measurement, placing the shunt resistor at the emitter of the switch is preferable.

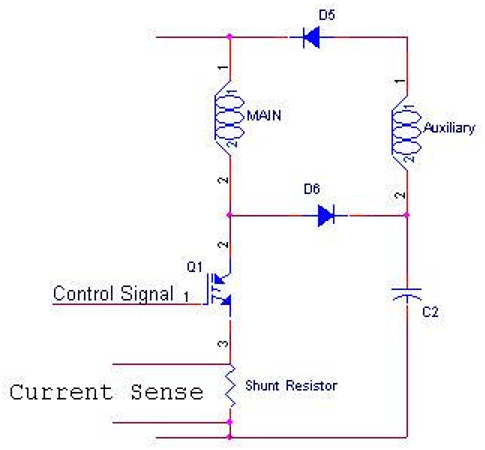


Figure 4.11 Schematic of the current feedback with a shunt resistor

This method has several advantages and disadvantages. The advantages are that it can be realized cost effectively and the current can be calculated easily. However, the disadvantage is that the ground path of the power converter board is the same with the ground path of the DSP controller board. So the measurement may be affected by the noise from the power converter board. The accuracy of measurement with the shunt resistor is much less than that of the measurement with the Hall-effect type current sensor.

4.3.5 Current Feedback with a Hall-Effect Type (HET) Current Sensor (LEM)

Instead of using a shunt resistor, the current of the main winding can be measured using a HET current sensor (LEM). It is usually used for high accuracy measurement applications. The noise from the main power board can be minimized as there is isolation between the ground path of the power converter board and that of the controller board. The placement of the current sensor is shown in figure 4.12.

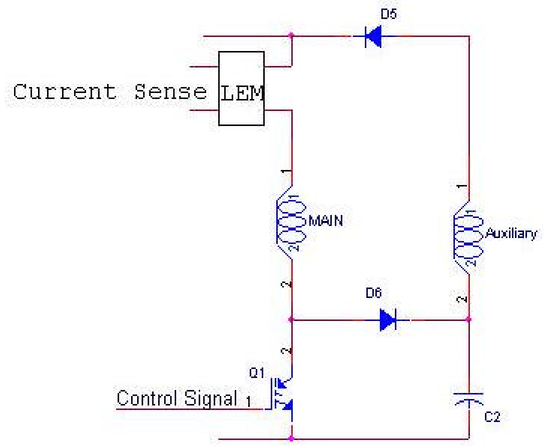


Figure 4.12 Schematic of the current feedback with a current sensor

After obtaining the current signal, either by a shunt resistor or a HET current sensor, a filter and an amplifier are added to reduce the noise that is generated from the switching device and the SSSRM. After filtering and amplifying the current signal, it is fed into the analog to digital converter channel of TI DSP.

4.3.6 Speed Control Algorithm

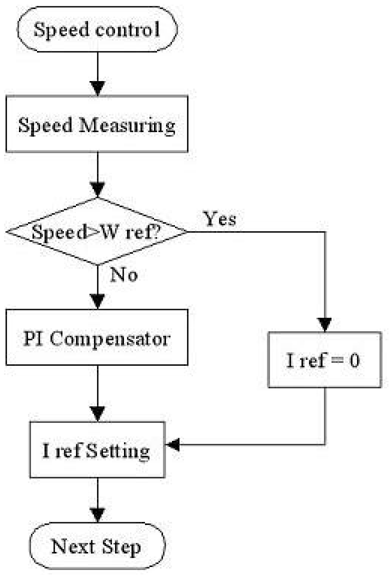


Figure 4.13 Flow chart of the speed control algorithm

The time between the edges of the two Hall sensors is inversely proportional to the speed. Whenever the edge signal from the Hall sensor was detected, the timer counts up and the number of counts is used to calculate the speed. If the error is positive, at CW/CCW motoring operation, the current command will be increased for more torque in motoring region. If the speed error is negative, the PWM function should be activated in braking region to decrease the rotor speed. In the CW motoring operation and the CCW motoring operation, the speed control algorithm activates the PWM function either in the motoring region or in the braking region in order to increase or decrease the rotor speed. If the CW braking operation and the CCW braking operation are being used, the speed control algorithm should be used. The purpose of the braking operation is to change direction and to limit current.

4.3.7 Quadrant Control Algorithm

The purpose of the quadrant controller is to set up the PWM sequence with respect to the quadrant command. As described in section 2.4.5, the controller also serves to generate the reversal-start pulse. After the rotor speed reaches the desired speed, the reversal-starting signal will be applied to the power converter. Figure 4.14 shows the flow chart for the four-quadrant control algorithm.

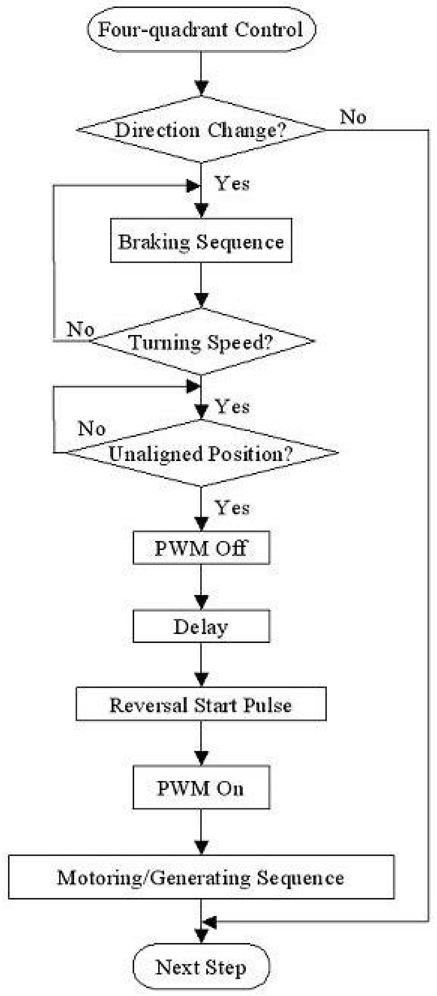


Figure 4.14 Flow chart of the four-quadrant control algorithm

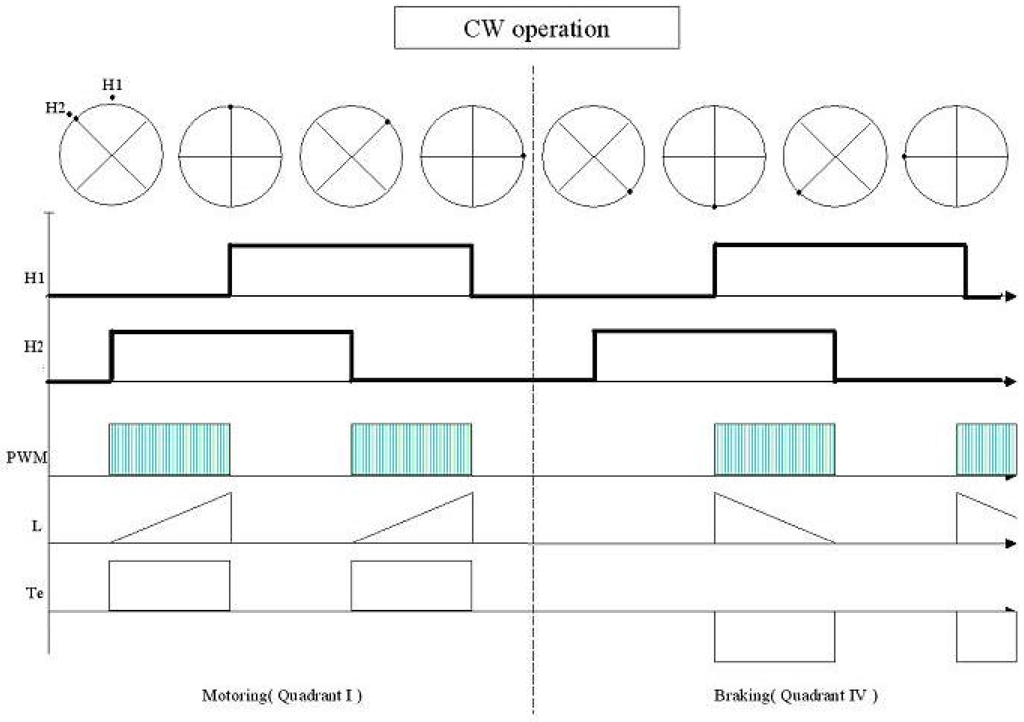


Figure 4.15 Motoring and braking operation at the CW direction

The four-quadrant operation can be explained with figure 4.15 and figure 4.16. Figure 4.15 shows CW motoring and braking operation. SSSRM drive system use two Hall sensors to obtain the rotor position information. Hall sensor #1 and #2 are located at the aligned position, and the unaligned pos ition, respectively. The polarity of the inductance slope is positive from the unaligned region to the aligned region, otherwise, the polarity of the inductance slope is negative region. For the motoring operation in the CW direction, the controller activa tes the PWM function in the positive inductance slope region, because of the polarity of the torque is determined by the polarity of the inductance. On the other hand, in the braking operation in the CW direction, the controller activates with the PWM func tion in the negative inductance slope region.

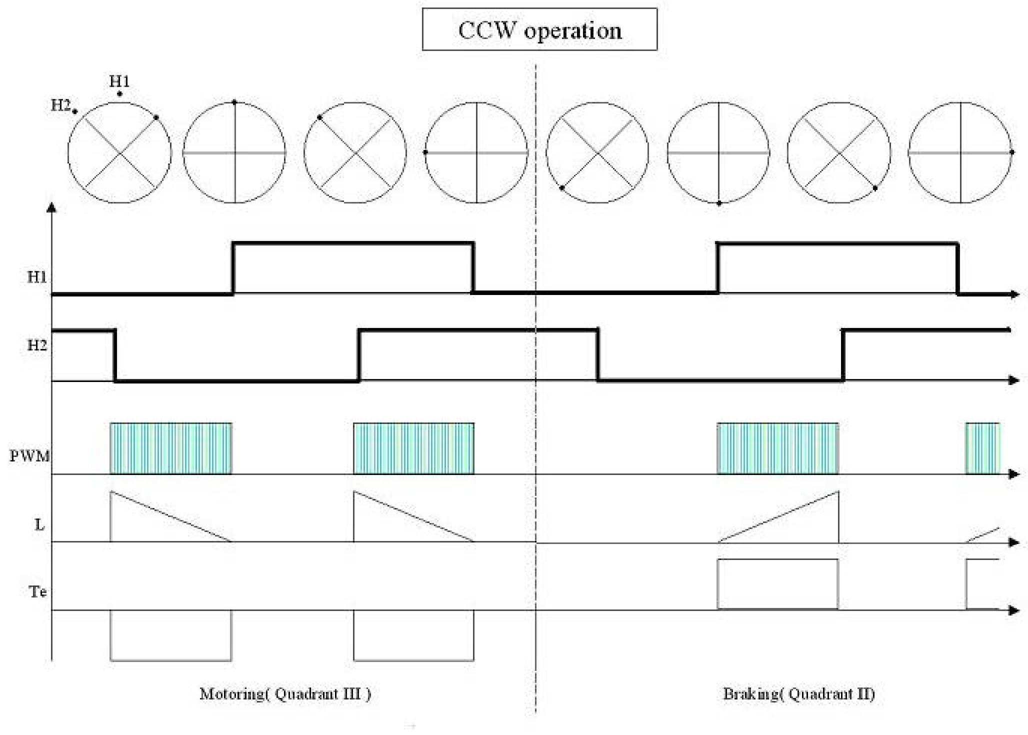


Figure 4.16 Motoring and braking operation at the CCW direction

The motoring and braking operation of the CCW direction is opposite to the sequence as described for the CW direction. The motoring operation in the CCW direction, PWM signal should be activated in the negative inductance slope region, otherwise, it activates the PWM function in the positive inductance slope region to run the braking operation.

4.4 External Module Interface

In this section, signal interface modules are presented with schematic and short descriptions.

4.4.1 Hall Sensor Interface

For obtaining the position data, the hall position sensors, shown in figure 4.17, were used. They are open-collector output type. Therefore, for interfacing with the other circuit, a pull up resistor should be added between the output terminal and V_{cc} terminal.

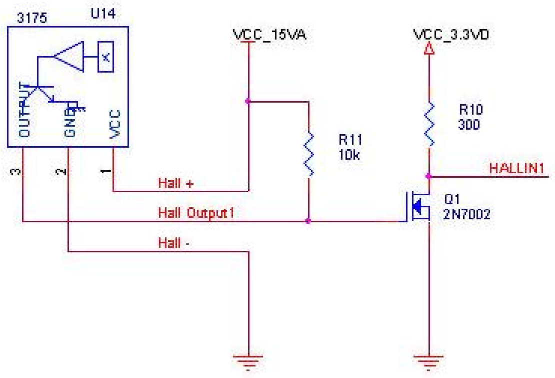


Figure 4.17 Schematic of the hall sensor interface module

For the 3.3V operation of TI DSP, the 2n7002, which is a 3.3V MOSFET, was connected to the hall sensor output terminal. Two output signals from the hall sensor interface module were connected to the Pin 11 and Pin 12 of the TI DSP.

4.4.2 Optocoupler Interface (IGBT Gate Drive Interface)

The TI DSP can control the IGBT with the duty cycle of the PWM output signal. The PWM output signal is 3.3V level. However, the IGBT gate signal level should be 15V. For switching of the IGBT gate drive, isolated 15V is required. For that, NMD0505, an isolated DC-DC converter, is used to provide 15V to the IGBT gate driver optocoupler, shown in figure 4.18, with an isolated ground.

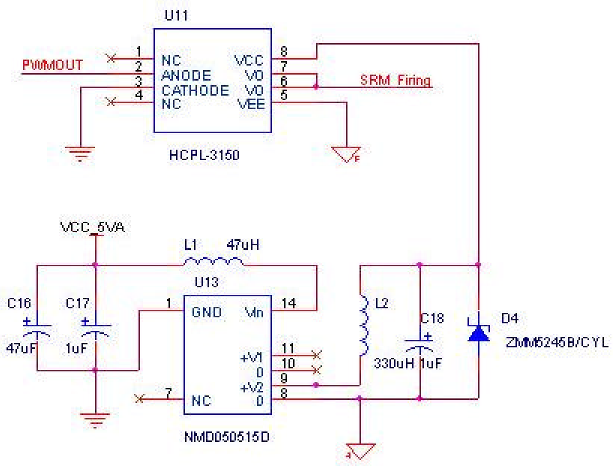


Figure 4.18 Schematic of the gate drive interface

The LED, built in the HCPL-3150, is optically coupled to an integrated circuit. It transmits the PWM signal to the output terminal. The ground, which is connected with isolated ground of NMD0505, is connected with the ground of the power converter.

4.4.3 JTAG Interface

JTAG Technology is one of the boundary-scan technologies and services for testing the printed circuit boards and systems, and for programming the flash memory. JTAG interface module, shown in figure 4.19, is connected with TI DSP JTAG terminals and XDS510PP, which is connected to the user PC with parallel port. XDS510PP transfers data from PC to TI DSP or from TI DSP to PC.

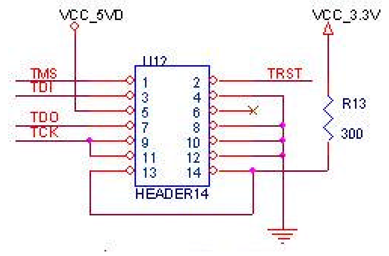


Figure 4.19 Schematic of the JTAG interface module

After compiling the source code, and then using JTAG Interface module, the executable machine code was downloaded into the flash memory of the TI DSP. For operating the JTAG emulator, 5V was supplied.

4.4.4 Speed Measurement Interface

To measure and display the rotor speed, the serial communication interface (SCI) and the frequency voltage converter (f/v converter) is used. The serial communication interface is used for low-speed measurement and the frequency voltage converter is used for high-speed measurement. Figure 4.20 shows SCI module for sending the speed data from DSP controller to the user computer.

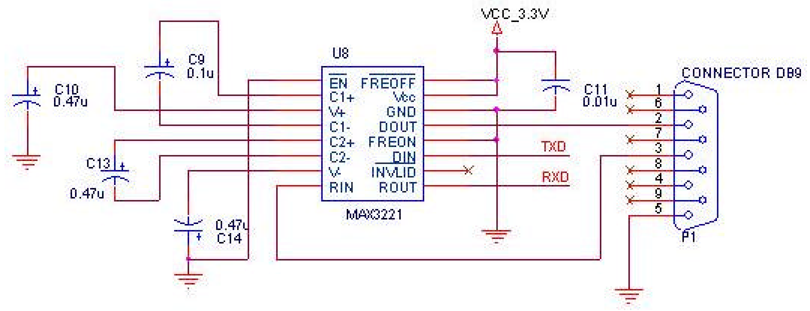


Figure 4.20 Schematic of the serial communication interface module

Since the TI DSP is 3.3V operation, the output and input signal should be adjusted to 3.3V level. The MAX3221 serves as the interface module as a 3V to 5.5V single channel RS-232 line driver/receiver.

Figure 4.21 shows f/v converter for displaying the current rotor speed. After receiving data from the controller through SCI, the data needs to be converted to speed unit and is displayed with another software (i.e. excel or matlab).

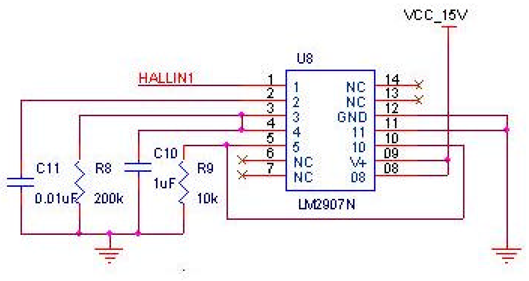


Figure 4.21 Schematic of the f/v converter module

Alternatively, f/v converter directly shows the rotor speed as a voltage level at the oscilloscope. The output voltage can be calculated as below:

$$V_o = V_{cc} \times f_{in} \times C_{11} \times R_8 \qquad (4.4)$$

For example, if the frequency is 400Hz (12000rpm), and desire output voltage is 12V, then the resitator R8 can be determined from the equation as following:

$$R_8 = \frac{V_o}{V_{cc} \times f_{in} \times C_{11}} = \frac{12}{15 \times 400 \times 0.01u} = 200k\Omega \qquad (4.5)$$

4.4.5 Current Feedback Module

As presented in chapter 3, the current flowing on the main winding can be measured with the current feedback module, shown in figure 4.22.

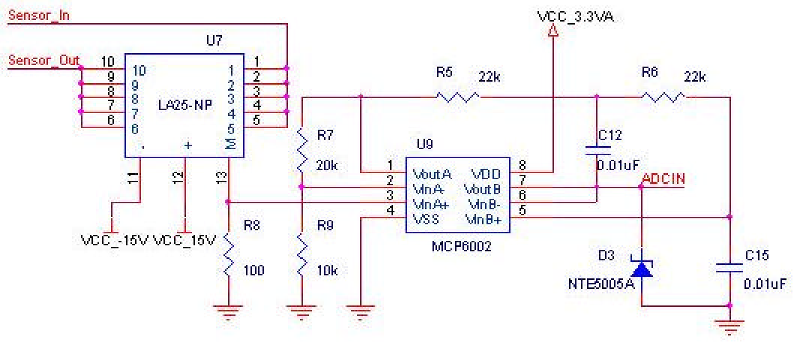


Figure 4.22 Schematic of the current feedback module

Current sensor module consists of a HET current sensor, which is manufactured by LEM, a low pass filter, and an amplifier with an OP-amp. The current sensor, providing conversion ratio of 1000 : 1, was used with measuring resistance. For example, if the current value is 1A, and the measuring resistance 100 ohm, then the output voltage is 0.1V, and $I_{sense} = 1mA$, $R_{measure} = 100\Omega$

By applying to the ohm's law,

$$V_{sense} = I_{sense} \times R_{measure} = 0.001 \times 100 = 0.1V \qquad (4.6)$$

To match signal level and to pre-amplify the signal, the non-inverting gain amplifier, shown in figure 4.23, was used. Pre-amplifying has the effect of increasing the distance between the original signal and the noise floor.

$$Gain = (1+\frac{R2}{R1}) = (1+\frac{30k}{10k}) = 4 \qquad (4.7)$$

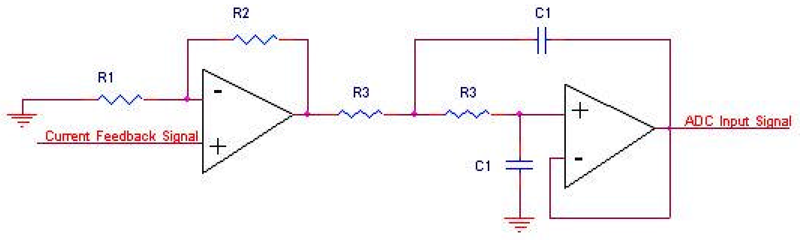


Figure 4.23 Schematic of the low pass filter and amplifier

The filter is a second order low-pass KRC filter designed for unity gain. Its cutoff frequency is

$$f_{cutoff} = \frac{1}{2p \cdot R3 \cdot C1} = \frac{1}{2p \cdot 22k \cdot 0.01u} \cong 723Hz \qquad (4.8)$$

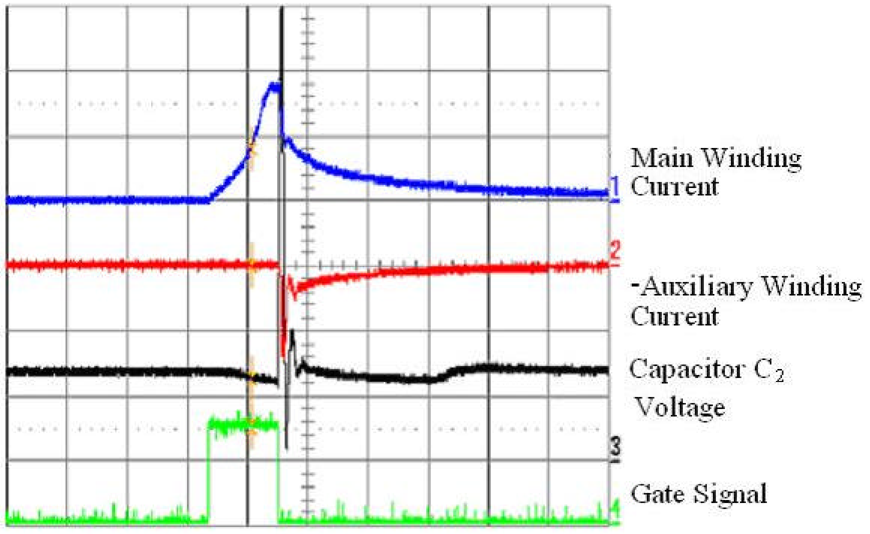
From the current feedback module, the reason to set the gain at 3 is not to exceed the acceptable voltage range. Since the current rate of the SPSRM is 8A and the measuring resistor is 100 ohms, the output voltage value at 8A from the current sensor is 0.8V, but the output of the feedback module has to be less than 3.3V, because TI DSP operates at 3.3V. So, with the current design, the output voltage from the current feedback module connected to the ADC input port of the TI DSP is 3.2V at the rated current of 8A ($0.8A \times 4\Omega = 3.2V$).

Chapter 5. Simulation and Experimental Results

Following the implementation of the converter and controller designs, the entire system was tested to verify the stated performance objectives. Verification covers the self-starting algorithm, the current controller design, and the speed controller design.

5.1 Self-starting Verification

To simulate a self-starting at aligned position, the initial position of the rotor is zero, the initial speed is also zero, and the initial conditions of the main winding current, the auxiliary winding current, and the capacitor, C2, voltage were considered from the experimental results shown in figure 5.1.



x-axis 2ms/div

y-axis 1 & 2 : 10A/div, 3 : 100v/div, 4 : 1v/div

Figure 5.1 Waveform of the initial conditions when one pulse was applied

Figure 5.1 shows the main winding current, the auxiliary winding current, capacitor C_2 voltage with being applied one pulse signal.

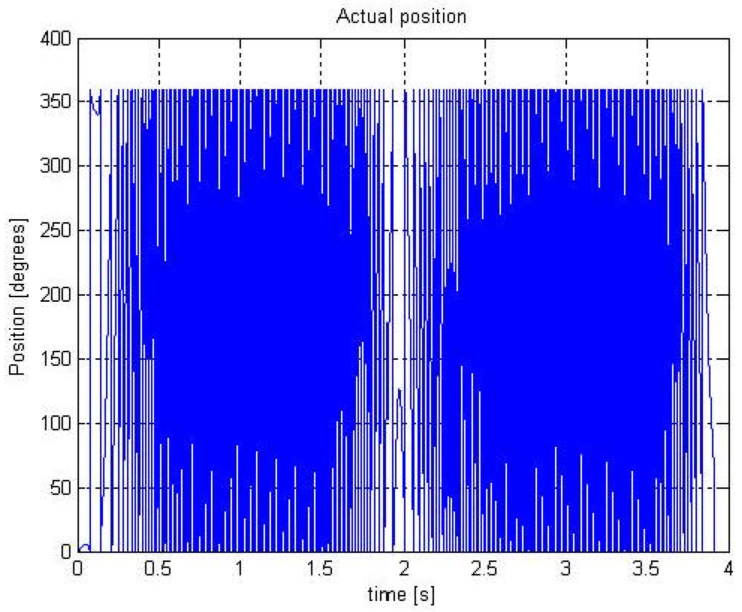
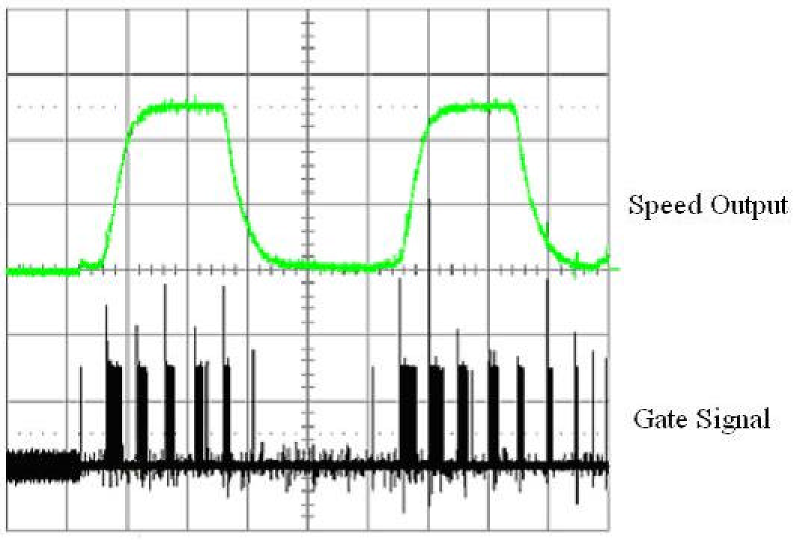


Figure 5.2 Simulation result of self-starting algorithm

Figure 5.2 shows the simulation result of the variation of the rotor position. The rotor position rolls over with every 360°. The rate in which the rotor position changes indicates the rotor speed. Using same initial conditions, the rotor begins to move a little bit. The controller controls the gate signal with respect to the speed command after the controller sensed moving the rotor.

The experimental result for self-starting is achieved by applying one pulse to the main winding coil as shown in figure 5.3.

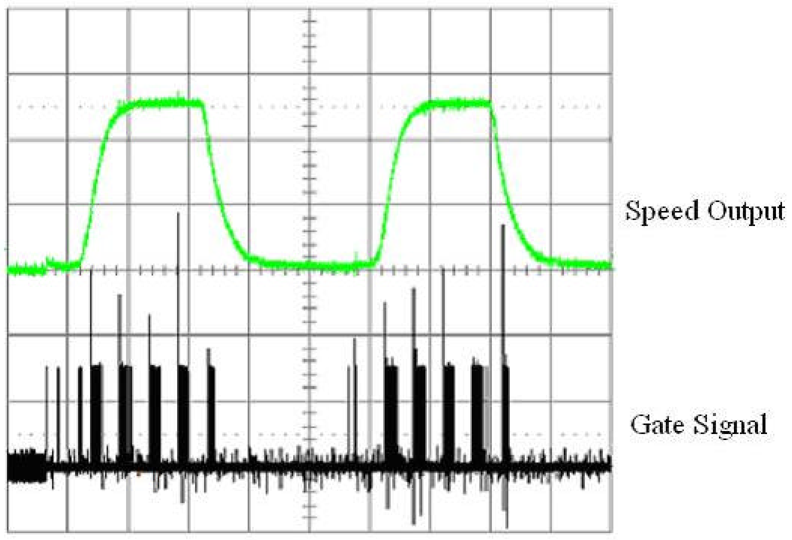


x-axis 5s/div
y-axis 1 : 2000rpm/div, 2 : 1v/div,

Figure 5.3 Waveform of the Self-starting at between aligned and unaligned position

When a pulse was applied to the main winding, the rotor begins to move with a certain speed. The rotor speed decreases, because the PWM function will not be occurred unless the controller detects the Hall sensor output signal. If the controller detects the Hall sensor output signal, it determines the rotation direction, and activates appropriate the PWM function. Consequently, when the rotor is in the motoring region, then the PWM signal is applied to the gate terminal of the converter. The SSSRM will be driven in a motoring mode. The self-starting pulse consists of an on-time and off-time. Usually, the on-time is larger than PWM period for generating starting torque. The off-time should be long enough to detect the rotation of the rotor from the Hall sensor, which has a 90° resolution, because the controller checks only one Hall sensor signal in algorithm. The on-time for the start pulse is 2ms and the off-time is 1second. From the figure 5.3, it can be realized that the SSSRM is rotated by applying the one starting pulse. It took 2 second between the time the pulse was applied to when the gate signal PWM was generated.

When the rotor is placed at the aligned position exactly, torque is not produced from the main winding current. However, if the rotor moves a little bit from torque generated from auxiliary winding current, we can apply another pulse to the main winding, and the rotor will start to rotate as shown in figure 5.4.



x-axis 5s/div
y-axis 1 : 2000rpm/div, 2 : 1v/div

Figure 5.4 Waveform of self-starting at aligned position

After one pulse was applied to the main winding at aligned position, the equilibrium point was broken. The self-starting is achieved by applying another pulse to the main winding. Finally, after two pulses were applied to the SSSRM, the rotor started to rotate from aligned position. The pulses have the same period of on-time and off-time, because the controller needs the same time to detect the Hall sensor output signal.

The second pulse was applied 1 second after applying the first pulse. It then took 2 seconds for the PWM gate signal to start after the second pulse was applied to the SSSRM. From the simulation results and the experimental results, the self-starting scheme was performed and verified.

5.2 Speed Controller Results

Figure 5.5 is a four-quadrant simulation of the speed loop given a positive/negative step speed command of + / - 5000 rpm.

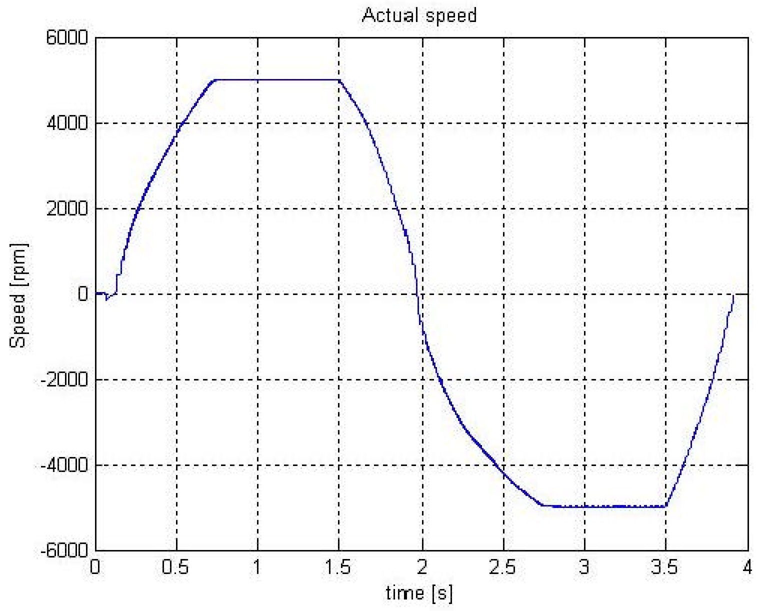
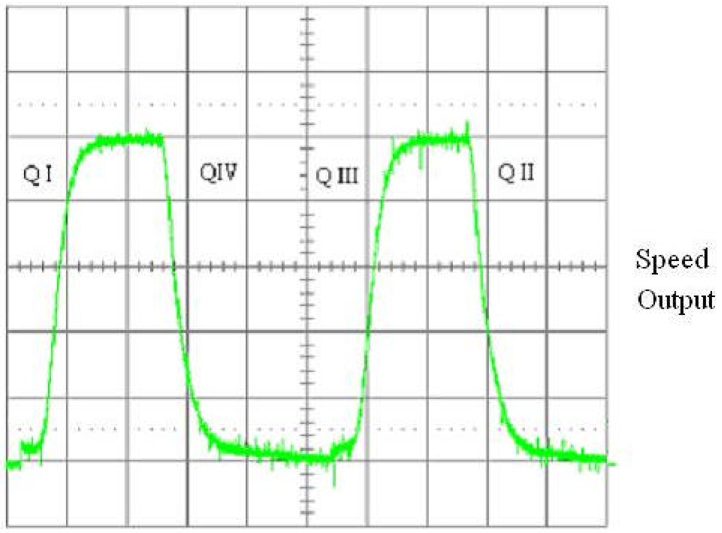


Figure 5.5 Simulation result of the four-quadrant speed operation

In this simulation, the speed loop controller algorithm was applied after detecting the variation of the rotor position of at least three degrees. The rise time of the speed at CW motoring mode is 0.6s. The SSSRM is driven in CW motoring for 0.2 seconds to 1.5 seconds, and then the speed reference is changed from +5000 rpm to –5000 rpm. Therefore, the CW braking mode was applied to the SSSRM for 1.5second. In braking mode, the speed was decreased dramatically. In simulation, after the speed of the rotor reached to the zero speed, the controller started to control for the CCW motoring. The CCW motoring and CCW braking is driven with the same algorithm with CW operation. It should be noted, the rise time of the speed output in the CCW motoring mode is 7.5s. This is longer than that of the speed output in the CW motoring mode, because the torque produced from the auxiliary winding is asymmetric to the SSSRM.

The speed control loop is implemented by the same architecture with the digital PI controller as mentioned in chapter 3 and appendix A. Figure 5.6 shows experimental result of the four-quadrant operation.



x-axis : 5s/div, y-axis : 1000rpm/div

Figure 5.6 Waveform of the four-quadrant speed operation

The SSSRM operates in the following sequence: CW motoring -> CW braking -> CCW motoring -> CCW braking. Therefore, the speed command was 5000rpm at first. At 1.5 second, one pulse signal was applied to the main winding, and then the rotor starts to move. At this point, the speed slowly went down. After the controller detected the Hall sensor signal, the controller activated the PWM. At 12.5 second, the speed command was changed to the CCW direction, and then the motor was driven into the braking operation. In braking mode, the speed control is not considered and only the current controller is realized. Thus, the rotor speed will be decreased dramatically while following the current reference. In this experiment, the current reference in the braking mode is 1.2A.

To change rotation direction, a reversal pulse should be applied at specific speed. After reaching a certain low speed (i.e. 250rpm) in braking mode, the PWM function is cleared and then the rotor speed was allowed to decrease slowly. When the controller detects the minimum speed (i.e. 100rpm), the controller gives a reversal pulse at the rotor angle that produces the highest negative torque. At 25.5 seconds, the reversal pulse was applied to the main winding, and then the rotor starts to rotate to the opposite direction. After achieving the CCW rotation, the controller drives the SSSRM in the CCW motoring operation. For the CCW braking operation, the same algorithm applies as in the CW braking operation.

From the simulation results and the experimental results, the four-quadrant speed control loop was performed and verified.

5.3 Current Controller Results

The current control loop is inside of the speed control loop. From the speed control loop, the current command is determined The current controller performance can be evaluated from the current waveform of the main winding.

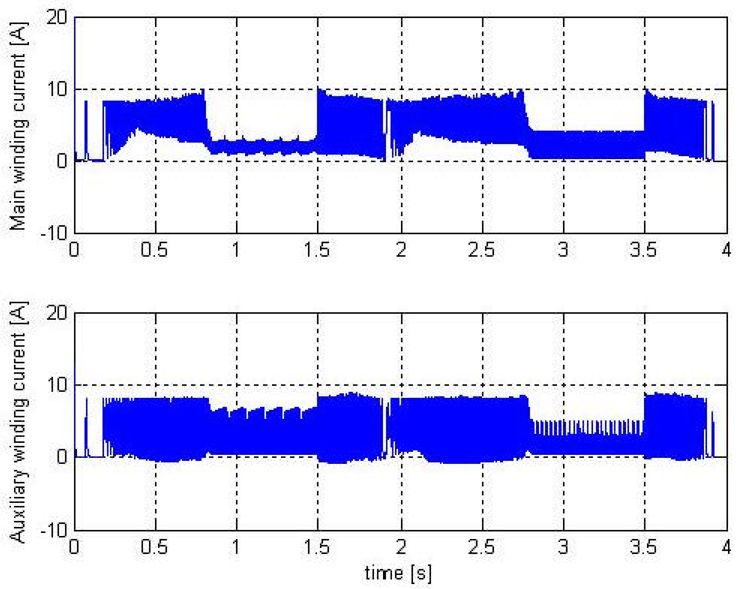
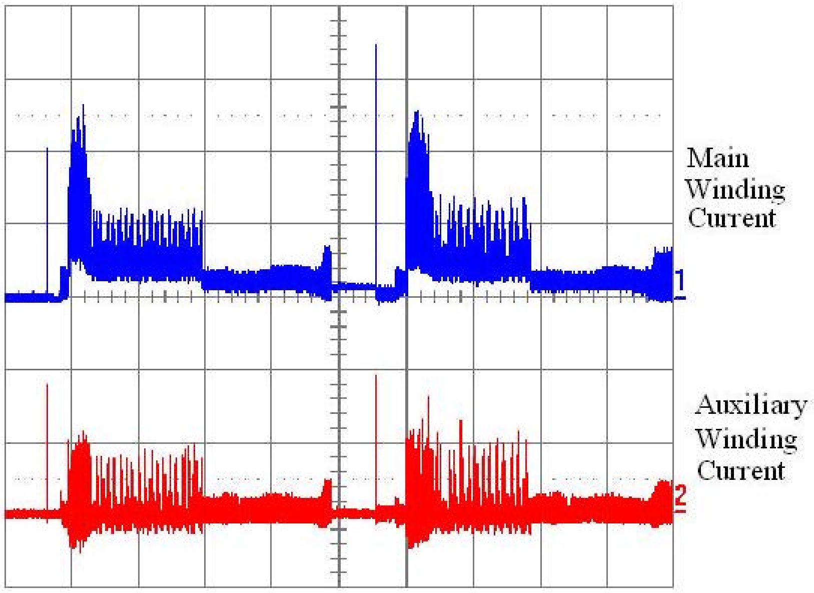


Figure 5.7 Simulation results of the main winding current and the auxiliary winding current

Following the outer speed control loop in simulation, the main winding current reflects the speed error. At first the speed error is big, the current command will also be at it's maximum limit. When the real speed reaches the speed command, the current command is decreased. At 1.5seconds, the controller enters the braking mode. The current value is set to 8A in order to decrease the rotor speed quickly. Figure 5.7 shows for controlling the auxiliary winding current indirectily.



x-axis : 5s/div, y-axis : 5A/div

Figure 5.8 Waveform of the main w inding current and the auxiliary winding current

Figure 5.8 shows the experimental result waveform of the main winding and the auxiliary winding current, respectively. The main winding current is controlled by speed control loop directly. For the motoring mode, the current is increased while catching up with the speed reference. After reaching the steady state speed, the main winding current is reduced around 2.5A. In braking mode, only current control algorithm is applied to the SSSRM, because the speed of the rotor is required to be dramatically reduced and finally stopped relatively quickly. The auxiliary winding current is controlled by the switch indirectly. Its waveform is very similar with the main winding current. However, the auxiliary winding current may be negative or positive with respect to the current flowing through the capacitor, C_2.

From the simulation results and the experimental results, the current control loop was performed and verified.

Chapter 6. Conclusions and Discussions

6.1 Conclusions

This thesis describes a single switch based SRM drives a self-starting algorithm, and a four-quadrant speed control algorithm. The conclusions of this research are:

1. A self-starting algorithm for the single switch based SRM was verified. Through experimental and simulation results, it is proved that multiple pulses can start to rotate machine even at aligned position.
2. The current and speed controllers were developed and implemented.
3. Four-quadrant operation algorithm was verified. The SSSRM architecture provides four-quadrant operation. Thus making a low cost four-quadrant feasible for the first time in published literature.

6.2 Recommendations for future study

The following recommendations are made for future study.

1. Derivation of control design methodology, which provides accurate speed and current control of the SSSRM.
2. Modification of the SSSRM architecture to accommodate more complex control and higher speed ranges.
3. Accurate modeling methods of the SSSRM for controller design.
4. Modification of the SSSRM architecture without position sensors for cost applications.

References

[1] R. Krishnan, "Switched Reluctance Motor Drives", CRC Press, 2001.

[2] R. Krishnan, "Electric Motor Drives-Modeling, Analysis, and Control", Prentice-Hall, 2001.

[3] G.C.Jenkinson and J.M.Stephenson, "STARTING OF A SINGLE-PHASE RELUCTANCE MOTOR", IEE 1999, 9TH Internatial Conference on Electrical Machines and Drives, PP391 - 395

[4] J.Y.Lim, Y.C.Jung, S.Y.Kim, and J.C.Kim, "Single Phase Switched Reluctance Motor for Vacuum Cleaner", IEEE-ISIE, 2001, pp 1393 - 1400

[5] R. Krishnan, A.M. Staley, and K. Sitapati, "A Novel Single-Phase Switched Reluctance Motor Drive System", IEEE-IECON, 2001, pp. 1488-1493.

[6] R. Krishnan, "Single switch based power converter topologies", disclosures to VTIP, 2002, patent pending.

[7] T. Higuchi, J.Fiedler, and R.W. Doncker, "On the Design of a Single-Phase Switched Reluctance Motor", IEEE, 2003, pp. 561-567

[8] Syed Hossain, Iqbal Husain, Harald Klode, Bruno Lequesne, and Avoki Omekanda, " Four-Quadrant Control of a Switched Reluctance Motor for a Highly Dynamic Actuator Load, APEC, 2002, pp. 41-47.

[9] C E B Green and J M Stephenson, "A Sensorless Switched Reluctance Drive", Electric Machines and Drives, 1997, pp. 64-68.

[10] B. Fahimi and Raymond B Sepe Jr., "Development of 4-Quadrant Sensorless Control of SRM Drives Over the Entire Speed Range", IEEE-IAS, 2002, pp. 1625-1632.

[11] G. Suresh, B. Fahimi, K.M.Rahman, M.Eshani, and I.Panahi, "Four-quadrant

Sensorless SRM Drive with High Accuracy at All Speeds", APEC, 1999, pp.1226-1231

[12] Analog Devices Application Note., Implementing PI controllers with the ADMC401, Jan, 2000

[13] R. Krishnan, Sung-Yeul Park, Keunsoo Ha, "Theory and Operation of a Four Quadrant Switched Reluctance Motor Drive with a Single Controllable Switch-The Lowest Cost Four Quadrant Brushless Motor Drive", IEEE-IAS, 2004.

Appendices

Appendix A - Implementing PI controller[12]

1. The continuous time domain

PI controllers are in most cases analysed and tuned in the continuous time domain. The corresponding transfer function is given as

$$U(s) = (K_p + \frac{K_i}{s}) \cdot I(s) \qquad (1)$$

where I and U denote the input error signal and the controller's output, respectively, s is the Laplace variable and K_p and K_i are the two parameters of the PI controller associated with the proportional(P) and integral(I) part. However, tuning a PI controller is not very intuitive when specifying these two parameters. The equation above may be rewritten as follows:

$$U(s) = K_p \cdot \left(1 + \frac{K_i}{K_p}\frac{1}{s}\right) \cdot I(s) = K_p \cdot w_{pi} \cdot \left(\frac{s/w_{pi} + 1}{s}\right) \cdot I(s) \qquad (2)$$

where $w_{pi} = \frac{K_i}{K_p}$ (expressed in [rad/s]). Evidently, this transfer function presents a pole in the origin and a zero located at w_{pi}. The gain at high frequencies is given by K_p itself.

2. Discrete time domain – Zero order hold

The transition from the continuous to the discrete time domain entails that the integral operation has to be approximated by a discrete summation.

With this approach, the signal is sampled at the instant k and held constant until the next sampling instant k+1. The following illustrates this.

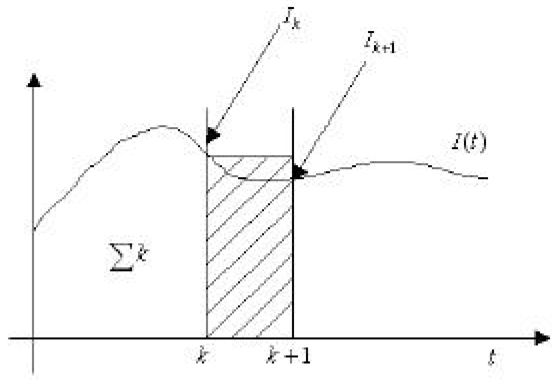


Figure 1. Integral with ZOH approximation

The integral operation is approximated by accumulating the rectangular areas. Denoting the sum at instant k with ∑k, the signal at instant k with I_k and the sample time with T_{sample}, the "integration" is achieved by :

$$\sum k+1 = \sum k + I_k \cdot T_{sample} \tag{3}$$

As known, the same equation may be expressed in the z-domain by

$$z \cdot \sum(z) = \sum(z) + I(z) \cdot T_{sample} \tag{4}$$

which leads to

$$\sum(z) = \frac{T_{sample}}{z-1} I(z) \tag{5}$$

Therefore the continuous integration $\frac{1}{s}$ in equation (1) is replaced by the 1st factor in the equation above. The transfer function in the discrete domain is obtained from (1) and (5) as:

$$U(z) = K_p \cdot \left(1 + w_{pi} \cdot \frac{T_{sample}}{z-1}\right) I(z) = \frac{K_p z + K_p (w_{pi} \cdot T_{sample} - 1)}{z-1} I(z) \tag{6}$$

This corresponds to the following difference equation:

$$U_{k+1} = K_p \cdot I_{k+1} + K_p (w_{pi} \cdot T_{sample} - 1) I_k + U_k \tag{7}$$

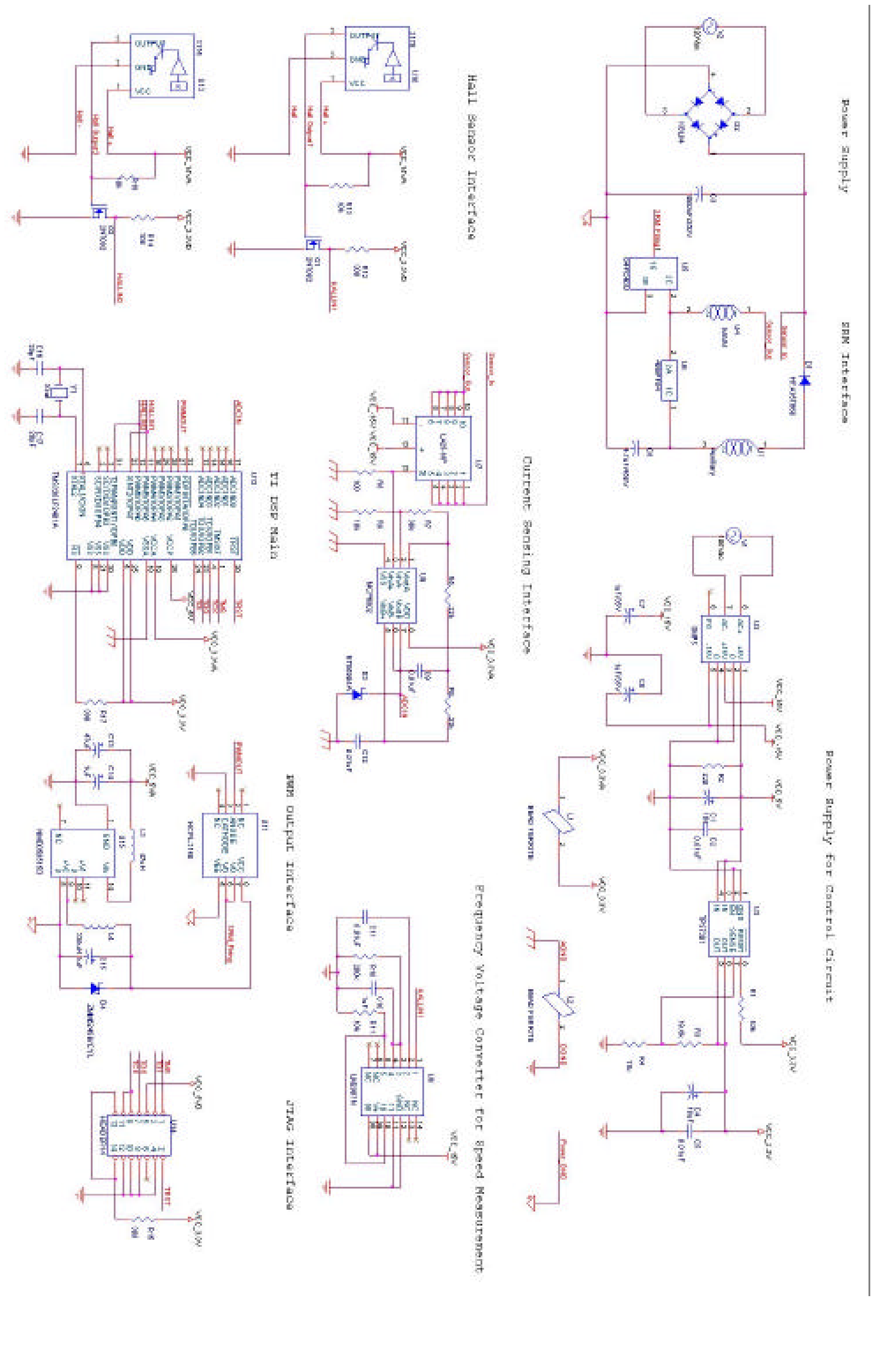
where, U_{k+1} is the new output, U_k is the old output, I_{k+1} is the new error input, and I_k is the old error input.

Appendix B – Bill of material

Number	Part Number	Description	Part Reference	Quan-tity
1	10u	Monolithic Capacitor	C1, C4,	2
2	0.01uF	Monolithic Capacitor	C2, C5, C9, C11, C12	5
3	1000uF/250V	Aluminum Electrolytic Capacitor	C3	1
4	4.7uF/450V	Aluminum Electrolytic Capacitor	C6	1
5	1uF/35V	Aluminum Electrolytic Capacitor	C7, C8, C10, C14, C15	5
6	47uF	Aluminum Electrolytic Capacitor	C13	1
7	20pF	Monolithic Capacitor	C16, C17	2
8	HFA15TB60	Ultrafast recovery Diode	D1	1
9	KBL04	Bridge Rectifier	D2	1
10	NTE5005A	3.3V Zener Didoe	D3	1
11	ZMM5245B/CYL	3.3V Zener Didoe	D4	1
12	BEAD FERRITE	Bead	L1, L2	2
13	47uH	Inductor	L3	1
14	330uH	Inductor	L4	1
15	2N7002	MOSFET	Q1, Q2	2
16	10k	Resistor	R1, R11	1
17	220	Resistor	R2	1
18	19.6k	Resistor	R3	1
19	11k	Resistor	R4	1
20	22k	Resistor	R5, R6	2
21	30k	Resistor	R7	1
22	100	Resistor	R8	1
23	10k	Resistor	R9, R13, R16	3
24	200k	Resistor	R10	1
25	300	Resistor	R12, R14, R15, R17	4
26	SMPS	Commercial Power Supply	U2	1
27	TPS7301	Voltage Regulator	U3	1
28	G4PC40U	IGBT	U5	1
29	40EPF04	Fast recovery Diode	U6	1
30	LA25-NP	Current Sensor	U7	1
31	LM2907N	F/V converter	U8	1

Number	Part Number	Description	Part Reference	Quantity
32	MCP6002	OP-AMP	U9	1
33	3175	Hall Latch Sensor	U10, U13	2
34	HCPL-3150	Opto-coupler	U11	1
35	TMS320LF2401A	TI DSP Controller	U12	1
36	HEADER14	Header	U14	1
37	NMD050515D	DC/DC converter	U15	1
38	10MHz X-tal	Crystal	Y1	1

Appendix C – Schematic of the SSSRM Drive System



Power Supply
SRM Interface
Power Supply for Control Circuit
Hall Sensor Interface
Current Sensing Interface
Frequency Voltage Converter for Speed Measurement
TI DSP Main
PWM Output Interface
JTAG Interface

Appendix D – TI DSP Source Code

```
/*----------------------------------------------------------*/
/*    Four-quadrant Operation in a SSSRM        */
/*   file name : step_final.c                    */
/*----------------------------------------------------------*/

#include  "testez2401.h"
#include "tms3 20lf2401.h"
#include "stdlib.h"

#define SetBit(x, y) (x |= (y))
#define ClrBit(x, y) (x &= (~y))
#define ChkBit(x, y) (x & (y))

#define CW 1
#define CCW 0

#define On 1
#define Off 0

#define Positive 1
#define  Negative  0

#define High 1
#define  Low    0

#define T1UFINT_FLAG 0x0200
#define CAP1INT_FLAG 0x0001
#define T2OFINT_FLAG 0x0008
#define  T2PINT_FLAG    0x0001

/* SCI module define part -----------------------------------------------------  */

#define TXRDY 0x0080
#define RXRDY 0x0040
#define SW_RESET 0x0020

struct{
                unsigned int desire_direction : 1;
                unsigned int real_direction : 1;
                unsigned int adc : 1;
                unsigned int capture : 1;
                unsigned int speed : 1;
                unsigned int polarity : 1;
                unsigned int external : 1;
                unsigned int PWM : 1;
} flag;


/*;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;PWM Constant Define
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

#define SwitchingFrequency 999 //50us period, 20kHz
/* CLK / PWM Period -1 = 20MHz / 20KHz -1 = 1000 -  1  =  999   */

#define  Timer2Frequency  399  //10us  period,  100kHz
/* CLK / PWM Period -1 = 40MHz / 100KHz -1 = 400 -  1  =  399   */
/*;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Constant Definitions
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
#define Kpi 1
#define Kii 500
#define  Kpw  20//30   //0.3
#define  Kiw  100   //0.1
#define Tc 0.0001
#define Tw 0.0005
#define Wpi Kiw / Kpw * Tw
#define Aw Kpw * (Kiw / Kpw * Tw - 1)
#define Ai Kpi * (Kii / Kpi * Tc - 1)


float Current_Speed, Output_New_Speed, Output_Old_Speed, Error_New_Speed, Error_Old_Speed;
float Output_New_Current, Output_Old_Current, Error_New_Current,  Error_Old_Current;
float  IRef,  WRef;
unsigned int Timer_edge,Timer_clock, Timer_adc, Timer_speed, Timer_Counter_ms, Mode;
unsigned int Adv_Counter, Adv_angle, ADC_result, Speed_Counter, Speed_Counter1;
unsigned long baud;
```

```
unsigned int temp, temp1, temp2;

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Variable and Sections Definitions
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

#define  PWM_max              999
#define  Duty_max            700//70%  duty  max
#define  IRef_max           600  //  4.8A  current

/* SCI module define part ---------------------------------------------------  */

void PositionCheck(void);
void Current_Control(void);
void StartCheck(void);
void Speed_Control(void);
void SCIPrintDec(int data);
void SCISend( char data);
void Reversal_Control(void);

void Reversal_Control(void)
{
        for(;;) // Opposite Direction Shot Routine
        {

                if(Speed_Counter1 > 15000) //origin : 30000
                {
                        if(Speed_Counter < 2)
                        {
                                *ACTRA = 0x0000; // PWM Off
                                Timer_clock = 0;
                                Timer_Counter_ms   =  0;
                                for(;;)if(Timer_clock >= 50)break;             //PWM   clear   and   delay
origin 50

                                Timer_clock = 0;
                                Timer_Counter_ms = 0;
                                *ACTRA = 0x0003;
                                for(;;)if(Timer_Counter_ms >= 3)break; //Reversal Pulse Signal High

                                *ACTRA = 0x0000;

                                for(;;)if(Timer_Counter_ms >= 1500)break; //Reversal Pulse Signal Low

                                Timer_Counter_ms = 0;

                                if(flag.real_direction == CCW)flag.desire_direction = CCW;
                                else flag.desire_direction = CW;

                                *CMPR1 = 100;

                                for ( temp1 = 0; temp1 < 10000; temp1++ )
                                {
                                        temp2 = 20;

                                        while ( temp2 )
                                        {
                                        temp2 --;
                                                PositionCheck();
                                        }
                                }

                                if(flag.desire_direction == CCW)Mode = 3;
                                else Mode = 1;
                                break;

                        }
                }

        }
}

void SCISend( char data )
{
        while( !(*SCICTL2 & TXRDY) ) ;

        *SCITXBUF = data;
}

void SCIPrintDec(int data)
{
        int temp1, temp2, temp3, temp4;
```

```
        temp1 = (int)data/1000;
        temp2 = (int)(data - temp1*1000)/100;
        temp3 = (int)(data - temp1*1000 - temp2*100)/10;
        temp4 = (int)(data - temp1*1000 - temp2*100 - temp3*10);
        SCISend(temp1+48);
        SCISend(temp2+48);
        SCISend(temp3+48);
        SCISend(temp4+48);
        SCISend(13);
}

void StartCheck(void)
{
        for(;;) // Opposite Direction Shot Routine
        {
                Timer_Counter_ms = 0;
                Timer_clock = 0;
                *ACTRA = 0x0003;
                for(;;)if(Timer_Counter_ms == 2)break;

                *ACTRA = 0x0000;
                Timer_Counter_ms = 0;
                Timer_clock = 0;
                flag.capture = Off;
                for(;;)if(Timer_Counter_ms == 1000)break;

                if(flag.capture == On)break;
        }

        //Pin 5 rising trigger condition
        if(flag.real_direction == CCW)flag.desire_direction = CCW;
        else flag.desire_direction = CW;

        *CMPR1 = 100;

        for ( temp1 = 0; temp1 < 10000; temp1++ )
        {
                  temp2 = 20;

                while ( temp2 )
                {
                temp2--;
                        PositionCheck();
                }
        }

        if(flag.real_direction == CCW)
        {
                flag.desire_direction = CCW;
                Mode = 3;
        }
        else
        {
                flag.desire_direction = CW;
                Mode = 1;
        }

        Timer_clock = 0;
        Timer_Counter_ms = 0;
}

void Speed_Control(void)
{

//PI control from ADMC application Note
        Current_Speed = (128000 / (Speed_Counter1));

        Error_New_Speed = WRef - Current_Speed;

        Output_New_Speed = Kpw * Error_New_Speed + Aw*Error_Old_Speed + Output_Old_Speed;


        if(Output_New_Speed < 0)
        {
                flag.polarity = Negative;
                Output_New_Speed = abs(Output_New_Speed);
        }
        else flag.polarity = Positive;

        if(Output_New_Speed > IRef_max) Output_New_Speed = IRef_max;
        IRef = Output_New_Speed;
```

```
		if(flag.polarity == Negative) Output_Old_Speed = - Output_New_Speed;
		else Output_Old_Speed = Output_New_Speed;
		Error_Old_Speed = Error_New_Speed;

		Timer_speed = 0;
		flag.speed = Off;

}

void Current_Control(void)
{

		Error_New_Current = (IRef - ADC_result);
		Output_New_Current = Kpi * Error_New_Current + Ai * Error_Old_Current + Output_Old_Current;

		if(Output_New_Current < 0) Output_New_Current = 0;
		if(Output_New_Current > Duty_max) Output_New_Current = Duty_max;
		*CMPR1 = Output_New_Current;

		Output_Old_Current = Output_New_Current;
		Error_Old_Current = Error_New_Current;
		flag.adc = Off;
}

void PositionCheck(void)
{
		unsigned int i;

		i = *PADATDIR;
		i &= 0x0060;

		if(flag.desire_direction == CW)
		{
				if((i == 0)||(i == 96))*ACTRA = 0x0000; // bit6-0, bit5-0		Swtich	Off	force	low

				else if((i == 32)||(i == 64))*ACTRA = 0x0001; // bit6-0, bit5-1 Switch On active low
		}
		else
		{
				if((i == 0)||(i == 96))*ACTRA = 0x0001; // bit6-0, bit5-0		Swtich Off force low
				else if((i == 32)||(i == 64))*ACTRA = 0x0000; // bit6-0, bit5-1 Switch On active low

		}
}

void main(void)
{
//Disable the watchdog

		B0_Enable;

		*WDCR = 0x00E8;
/*
* bit 7		1:	clear WD flag
* bit 6		1:	disable the dog
* bit 5-3	101:	must be written as 101
* bit 2-0	000:	WDCLK divider = 1
*/
//Setup the system control registers

		*SCSR1 = 0x00CD; //40MHz Operation
/* bit 15		0:	reserved
* bit 14		0:	CLKOUT = CPUCLK
* bit 13-12	00:	IDLE1 selected for low-power mode
* bit 11-9	000:	PLL x4 mode
* bit 8		0:	reserved
* bit 7		1:	1 = enable ADC module clock
* bit 6		1:	1 = enable SCI module clock
* bit 5		1:	1 = enable SPI module clock
* bit 4		1:	1 = enable CAN module clock
* bit 3		1:	1 = enable EVB module clock
* bit 2		1:	1 = enable EVA module clock
* bit 1		0:	reserved
* bit 0		1:	clear the ILLADR bit
*/
//		*SCSR2 = (*SCSR2 | 0x000B) & 0x000F;
/*
* bit 15-6	0's:	reserved
* bit 5		0:	DO NOT clear the WD OVERRIDE bit
* bit 4		0:	XMIF_HI-Z, 0=normal mode, 1=Hi-Z'd
* bit 3		1:	1 = disable the BOOT ROM
* bit 2         1:      MP/MC*, 1 = Flash addresses mapped external
* bit 1-0       11:     11 = SARAM mapped to prog and data
*/

//Setup the ADC

                *ADCTRL1 = 0x4000;
/*
* bit 14        1:      1 = reset ADC module*/

                NOP;

                *MAXCONV = 0x0000;
/*
* bit 15-7      0's:    reserved
* bit 6-4       000:    MAX_CONV2 value
* bit 3-0       0000:   MAX_CONV1 value (0 means 1 conversion)
*/
                *CHSELSEQ1 = 0x0000;

/*
* bit 15-12     0000:   CONV03 channel
* bit 11-8      0000:   CONV02 channel
* bit 7-4       0000:   CONV01 channel
* bit 3-0       0000:   CONV00 channel (only active conversion)
*/
                *ADCTRL1 = 0x2F10; //origin : 0x2F90;
/*
* bit 15        0:      reserved
* bit 14        0:      RESET, 0=no action, 1=reset ADC
* bit 13-12     10:     SOFT and FREE, 10=stop after current conversion
* bit 11-8      1111:   ACQ_Prescaler, 1111 = 32 x Tclk
* bit 7         0:      CPS, 0: Fclk=CPUCLK/1, 1: Fclk=CPUCLK/2
* bit 6         0:      CONT_RUN, 0=start/stop mode, 1=continuous run
* bit 5         0:      0=hi priority int, 1=low priority int
* bit 4         1:      0=dual sequencer, 1=cascaded sequencer
* bit 3         0:      0=calibration mode disabled
* bit 2         0:      BRG_ENA, used in calibration mode only
* bit 1         0:      HI/LO, no effect in normal operation mode
* bit 0         0:      0=self-test mode disabled
*/
                *ADCTRL2 = 0x4600; //origin : 0x4700; for soc start from EVA
/* bit 15       0:      EVB_SOC_SEQ, 0=no action
* bit 14        1:      RST_SEQ1/STRT_CAL, 0=no action
* bit 13        0:      SOC_SEQ1, 0=clear any pending SOCs
* bit 12        0:      SEQ1_BSY, read-only
* bit 11-10     01:     INT_ENA_SEQ1, 01=int on every SEQ1 conv
* bit 9         1:      INT_FLAG_SEQ1, write 1 to clear
* bit 8         0:      EVA_SOC_SEQ1, 1=SEQ1 start from EVA
* bit 7         0:      EXT_SOC_SEQ1, 1=SEQ1 start from ADCSOC pin
* bit 6         0:      RST_SEQ2, 0=no action
* bit 5         0:      SOC_SEQ2, no effect in cascaded mode
* bit 4         0:      SEQ2_BSY, read-only
* bit 3-2       00:     INT_ENA_SEQ2, 00=int disabled
* bit 1         0:      INT_FLAG_SEQ2, write 1 to clear
* bit 0         0:      EVB_SOC_SEQ2, 1=SEQ2 started by EVB
*/

//Setup the Timer1 for the PWM1

                *MCRA = 0x1802; /* group A pins */
/*
bit 15 0: reserved
bit 14 0: reserved
bit 13 0: 0=IOPB5, 1=IOPB5
bit 12 1: 0=IOPB4, 1=SCIRXD
bit 11 1: 0=IOPB3, 1=SCITXD
bit 10 0: 0=IOPB2, 1=IOPB2
bit 9 0: 0=IOPB1, 1=IOPB1
bit 8 0: 0=XINT1/IOPB0, 1=T2PWM

bit 7 0: 0=XINT2/ADCSOC/CAP1/IOPA7, 1=CLKOUT
bit 6 0: 0=IOPA6, 1=PWM6
bit 5 0: 0=IOPA5, 1=PWM5
bit 4 0: 0=IOPA4, 1=PWM4
bit 3 0: 0=IOPA3, 1=PWM3
bit 2 0: 0=IOPA2, 1=PWM2
bit 1 1: 0=IOPA1, 1=PWM1
bit 0 0: 0=IOPA0, 1=PDPINTA
*/

                *SCICCR = 0x07; //Protocol Setting
                // |stop bits|Parity Odd/even|Parity enable|Loop back enable|
                //   0000b => 1 stop bit, odd parity, parity disable, loop back disable
                // |IDLE/ADDR|char2|char1|char0|
                //   0111b => idle mode, 8 bit data
                //   0110b => idle mode, 7 bit data

                *SCICTL1 = 0x03; //TX, RX activating
                // |reserved|RX ERR INT|SW reset|reserved|
                // 0000b => disabe RX error interrupt1 and software reset
                // |TXWAKE|SLEEP|TX ENA|RX ENA|
                // 0011b => TX, RX enable

                //Baudrate Setting ; baud = 38400 at 40MHz
                *SCIHBAUD = (unsigned int)((0x0000>>16) & 0x0000FFFF);
                *SCILBAUD = (unsigned int)(0x0081 & 0x0000FFFF);

                //SCIHBAUD = 0x00;
                //SCILBAUD = BAUD_SELECT(BAUD_RATE);
                //BRR = BAUD_SELECT(BAUD_RATE);

                *SCICTL1 = 0x23; //Software reset

/*** Setup timers 1 and the PWM configuration ***/
                *T1CON = 0x0000; /* disable timer 1 */
                *T2CON = 0x0000; /* disable timer 2 */

                *GPTCONA = 0x0080; /* configure GPTCONA */
/*
bit 15 0: reserved
bit 14 0: T2STAT, read.only
bit 13 0: T1STAT, read.only
bit 12.11 00: reserved
bit 10.9 00: T2TOADC, 00 = no timer2 event starts ADC
bit 8.7 01: T1TOADC, 01 = underflow interrupt timer1 event starts ADC
bit 6 0: TCOMPOE, 0 = Hi.z all timer compare outputs
bit 5.4 00: reserved
bit 3.2 00: T2PIN, 00 = forced low
bit 1.0 00: T1PIN, 00 = forced low
*/


/* Timer 1: configure to clock the PWM on PWM1 pin */
/* Symmetric PWM, 20KHz carrier frequency, 25% duty cycle */
                *T1CNT = 0x0000; /* clear timer counter */
                *T1PR = SwitchingFrequency; /* set timer period */
                *DBTCONA = 0x0000; /* deadband units off */
                *CMPR1 = 0; //pwm_duty; /* set PWM1 duty cycle */
                *T1CON = 0x1240; //840; /* configure T1CON register */
/*
bit 15.14  00: stop immediately on emulator suspend
bit 13      0: reserved
bit 12.11  10:     = continuous up count mode
             (01: 01 = continous.up/down count mode)
bit 10.8   010: = x/4 prescaler
bit 7       0: reserved in T1CON
bit 6       1: TENABLE, 1 = enable timer
bit 5.4    00: 00 = CPUCLK is clock source
bit 3.2    00: 00 = reload compare reg on underflow
bit 1       0: 0 = disable timer compare
bit 0       0: reserved in T1CON
*/

                *ACTRA = 0x0000; /* PWM1 pin set active high */
/*
it 15 0: space vector dir is CCW (doní »t care)
bit 14.12 000: basic space vector is 000 (dontí » care)
bit 11.10 10: PWM6/IOPA6 pin forced low
bit 9.8 10: PWM5/IOPA5 pin forced low
bit 7.6 10: PWM4/IOPA4 pin forced low
bit 5.4 01: PWM3/IOPA3 pin active high
bit 3.2 00: PWM2/IOPA2 pin forced low
bit 1.0 01: PWM1/IOPA1 pin active high
*/

/* Timer 2: configure to clock for Input Capture Unit */
// Timer2 is a timer for calculating speed : 10us timer
                *T2CNT = 0x0000; /* clear timer counter */
                *T2PR = Timer2Frequency; /* set timer period */

                *T2CON = 0x1040; /* configure T2CON register */
/*
bit 15.14  00: stop immediately on emulator suspend
bit 13        0: reserved
bit 12.11   10:      = continuous up count mode
                 (01: 01 = continous.up/down count mode)
bit 10.8    000: 000 = x/1 prescaler
bit 7         0: reserved in T2CON
bit 6        1: TENABLE, 1 = enable timer2
bit 5.4     00: 00 = CPUCLK is clock source
bit 3.2     00: 00 = reload compare reg on underflow
bit 1        0: 0 = disable timer compare
bit 0         0: reserved in T2CON
*/

              *COMCONA = 0x8200; /* configure COMCON register */
/*
bit 15 1: 1 = enable compare operation
bit 14.13 00: 00 = reload CMPRx regs on timer 1 underflow
bit 12 0: 0 = space vector disabled
bit 11.10 00: 00 = reload ACTR on timer 1 underflow
bit 9 1: 1 = enable PWM pins
bit 8.0 0í »s: reserved
*/

              *CAPCONA = 0x0000; // capture1 reset
              *CAPCONA = 0x22C0; // rising Edge detect by based on Timer2
/*
bit 15           0: Clear all values
bit 14.13  01: Enalbe Capture Units 1 and 2
bit 12.10   000: reserved
bit 9        0: Select GP Timer1 or 2, 0:Timer2 1:Timer1
bit 8          0: reserved
bit 7.6      11: CAP1Edge
                              00=No detection,    01=Detect rising edge
                              10=Detect falling, 11=Detect both edge
bit 5.0      000000: reserved
*/

//External Interrupt Initialize
              *XINT1CR |= 0x0001;
/*
* bit15 XINTx Flag 1; 0 = No detected, 1 = Transition detected
* bit 2 XINTx Polarity 0; 0 = Interrupt generated on a falling edge, 1 = rising edge
* bit 1 XINTx Priority 0; 0 = High priority, 1 = Low priority
* bit 0 XINTx Enable   1; 0 = Disable Interrupt, 1 = Enable Interrupt
*/

//Setup the core interrupts
              *IMR = 0x0000;
              *IFR = 0x003F;
              *IMR = 0x000F;
/*
* bit 15-6          Reserved    0
* bit 5 INT6FLAG / INT6MASK
* bit 4 INT5FLAG / INT5MASK
* bit 3 INT4FLAG / INT4MASK
* bit 2 INT3FLAG / INT3MASK
* bit 1 INT2FLAG / INT2MASK
* bit 0 INT1FLAG / INT1MASK
*/

/*** Setup the event manager interrupts ***/
              *EVAIFRA = 0xFFFF; /* clear all EVA group A interrupts */
              *EVAIFRB = 0xFFFF; /* clear all EVA group B interrupts */
              *EVAIFRC = 0xFFFF; /* clear all EVA group C interrupts */
              *EVAIMRA = 0x0200; /* enable desired EVA group A interrupts */
              *EVAIMRB = 0x0001; /* enable desired EVA group B interrupts */
              *EVAIMRC = 0x0001; /* enable desired EVA group C interrupts */

/*** Configure IOPB3 pin as an output ***/

     *ACTRA = 0x0000; //active low

flag.capture = Off;
flag.speed = On;
flag.polarity = Positive;
flag.external = High;

Error_New_Current = Error_Old_Current = Output_New_Current = Output_Old_Current = 0;
Output_New_Speed = Output_Old_Speed = Error_Old_Speed = Error_New_Speed = Current_Speed = 0;
ADC_result = 0;

Timer_adc = Timer_speed = Timer_Counter_ms = Timer_clock = Speed_Counter = 0;
Speed_Counter1 = 65535;
```

```
Adv_angle  =  0;
WRef  =  427;  //5000rpm

EI;

StartCheck();

        while(1)
        {
                switch(Mode)
                {
                        case 1 ://CW mode
                                if(flag.speed == On)Speed_Control();
                                if(flag.adc  ==  On)Current_Control();
                                if(Current_Speed > 256 ) Adv_angle = 2;


                                if(Timer_Counter_ms > 10000) // 8 second waiting
                                {
                                        Mode = 2;
                                        IRef = 170;

                                        Timer_Counter_ms = 0;

                                        flag.desire_direction = CCW;
                                }


                        break;

                        case 2 : //braking mode
                                PositionCheck();
                                if(flag.adc  ==  On)Current_Control();

                                if(Speed_Counter1 > 6000)Reversal_Control(); //250rpm



                        break;

                        case 3 : //CCW mode
                                if(flag.speed  ==  On)Speed_Control();
                                if(flag.adc  ==  On)Current_Control();

                                if(Current_Speed > 128 ) Adv_angle = 2;
                                if(Current_Speed > 256 )  Adv_angle = 3;

                                if(Timer_Counter_ms > 10000) // 8 second waiting
                                {
                                        Mode = 2;
                                        IRef = 150; //origin : 130

                                        Timer_Counter_ms = 0;
                                                        flag.desire_direction = CW;

                                }


                        break;
                }
        }

}

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Timer 1 Underflow Interrupt Service Routine
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
interrupt void timer1_isr(void)
{
        DI;

        *EVAIFRA  |=  T1UFINT_FLAG;

        EI;
}

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Timer 2 Underflow Interrupt Se rvice Routine
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
interrupt void timer2_isr(void) //10us timer counter
{
        DI;
```

```
            *EVAIFRB |= T2PINT_FLAG;    //overflow flag clear

            if(++Timer_clock >= 100) //1ms timer
            {
                        ++Timer_Counter_ms ;
                        Timer_clock = 0;
            }

            if(++Timer_speed >= 50)flag.speed = On; //2kHz cycle

            if(++Timer_adc >= 10)
            {
                        *ADCTRL2 |= 0x2000; //Set Bit13
                        Timer_adc = 0;
            }

            ++Speed_Counter;

            if((Mode == 1) || (Mode == 3))
            {
                        if(Adv_Counter > ++Timer_edge)
                        {
                                    if(flag.PWM == On)*ACTRA = 0x0001;
                                    else *ACTRA = 0x0000;
                        }
                        else
                        {
                                    if(flag.PWM == On)*ACTRA = 0x0000;
                                    else *ACTRA = 0x0001;
                        }
            }

            EI;
}

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Capture Input Interrupt using Timer 2
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
interrupt void capture_isr(void)
{
            DI;

            *EVAIFRC |= CAP1INT_FLAG;   // clear from EVA flag

            Speed_Counter1 = Speed_Counter;
            Speed_Counter = 0;
            Adv_Counter = (int)(Timer_edge / 45 *(45 - Adv_angle));
            Timer_edge = 0;

            temp = *PADATDIR;
            temp &= 0x0060;

            if((temp == 0)||(temp == 96))flag.real_direction = CCW;
            else if((temp == 32)||(temp == 64))flag.real_direction = CW;

            if(flag.polarity == Positive)flag.PWM = On;
            else flag.PWM = Off;

            flag.capture = On;

            EI;
}

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;External Interrupt Service Routine
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

interrupt void int1_isr(void)
{

            DI;

            if((*PIVR - 0x0001) == 0)//external interrupt
            {
                        *XINT1CR |= 0x8000;
                        if(flag.polarity == Positive)flag.PWM = Off;
                        else flag.PWM = On;
                        Adv_Counter = (int)(Timer_edge / 45 *(45 -  Adv_angle));
                        Timer_edge = 0;

                        if(flag.external == Low)
```

```
                {
                        flag.external = High;
                        *XINT1CR |= 0x0004; //rising edge setting
                }
                else
                {
                        flag.external = Low;
                        *XINT1CR &= ~0x0004; //falling edge setting
                }
        }

        if((*PIVR - 0x0004) == 0)//ADC interrupt
        {
                ADC_result = (*RESULT0 >> 6); //to save adc result

                if(ADC_result <= 30) ADC_result = 0;

                flag.adc = On;

                *ADCTRL2 |= 0x4200;
        }
        EI;
}
```

Appendix E – Matlab Simulation Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1 PHASES SR MNACHINE (4 STATORS , 4 ROTORS)
% SELF STARTING OF SR MACHINE
% LOAD TORQUE = 0% of THE MAXIMUM TORQUE
%
% Kps = 200, Kis = 2, Kpi = 100, Kii = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LOAD MACHINE DATA

load machine_data;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ASSIGN VARIABLES
count = 0;
steps = 1.5e6;
break_steps_1 = 1.5e6;     % breaking SRM in forward rotation
break_steps_2 = 3.5e6;     % breaking SRM in reverse rotation
break_steps_3 = 1e6;
break_step_4 = 1.3e6;

t_step = 1e-6;                              % simulation step -> 1us
t_break_1 = t_step*break_steps_1;   % breaking at 150ms
t_break_2 = t_step*break_steps_2;   % breaking at 350ms
t_end   = t_step*steps;                    % end of simulation --> 500ms

% SR macine parameters
v_dc = 170;                                % DC voltage, applied to windings [Vdc]
main_R = 6.2;                             % main winding resistance [Ohm]
aux_R = 9.2;                               % auxiliary winding resistance [Ohm]
Cap = 4.7e-6;                             % capacitance
T_e_max = 1.9134;                      % max Torque command at 8A
i_max = 8;                                   % max current
B = 2.3e-5;                                 % friction constant [Nm s / rad]
J = 4e-4;                                     % inertia [kg m^2]
Load = 0*T_e_max;                              % load torque [Nm] = 0% of maximum torque

omega_max_volt = 3.3;                                    % voltage corresponding to the max. speed
Nr_max = 12000;                                               % max speed in RPM
omega_max = 2*pi*Nr_max/60;
v_c_max = 3.3;                                % limiting voltage

% speed controller, PI
K_ps = 200;
K_is = 2;

% current controller, PI + limiter
K_pi = 50; %origin 100
K_ii = 1;

% speed feedback filter, gain + delay time
H_w = omega_max_volt/omega_max;      % gain [Vs/rad]
% T_w = 2e-3;                                  % delay time [s]

% current feedback filter
H_i = v_c_max/i_max;                       % gain [Ohm]
T_i = 5e-6;                                   % delay time [s]

% PWM controller
```

```
f_c = 20e3;                          % carrier frequency, [Hz]
T_c = 1/f_c;                         % carrier period [s]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INITIALIZATION
t=0; t_spd=0; c=0;
c_f=0; c_r=0;
c_f_m=0; c_f_r=0;                  % flags for motoring and braking in forward operation mode
c_r_m=0; c_r_r=0;                   % flags for motoring and braking in reverse operation mode

direction_flag = 0;                % 0;Forward roation , 1;Reverse rotation

% Startup position of the machine
theta=0*(pi/180);
theta_old=0*(pi/180);

% Advanced angle and commutation angle
% 1500rpm - 2500rpm
Adv_angle_1= 1*pi/180;                          % Advanced angle = 1 degrees;
Comm_angle_1 = 1*pi/180;                        % Commutation angle = 1 degrees;

% 2500rpm - 3500rpm
Adv_angle_2 = 5*pi/180;                         % Advanced angle = 1 degrees;
Comm_angle_2 = 5*pi/180;                        % Commutation angle = 1 degrees;

% 3500rpm - 4200rpm
Adv_angle_3 = 7*pi/180;                         % Advanced angle = 3 degrees;
Comm_angle_3 = 7*pi/180;                         % Commutation angle= 3 degrees;

% 4200rpm - 4900rpm
Adv_angle_4 = 11*pi/180;                         % Advanced angle = 9 degrees;
Comm_angle_4 = 11*pi/180;                        % Commutation angle = 9 degrees;

% 4900rpm - 6000rpm
Adv_angle_5 = 13*pi/180;                         % Advanced angle = 9 degrees;
Comm_angle_5 = 13*pi/180;                        % Commutation angle = 9 degrees;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% -1500rpm ~ -2500rpm
Adv_angle_1_neg = 3*pi/180;                         % Advanced angle = 1 degrees;
Comm_angle_1_neg = 3*pi/180;                       % Commutation angle = 1 degrees;

% -2500rpm ~ -3500rpm
Adv_angle_2_neg = 7*pi/180;                        % Advanced angle = 3 degrees;
Comm_angle_2_neg  = 7*pi/180;                      % Commutation angle = 3 degrees;

% -3500rpm ~ -4200rpm
Adv_angle_3_neg = 9*pi/180;                        % Advanced angle = 7 degrees;
Comm_angle_3_neg = 9*pi/180;                        % Commutation angle = 7 degrees;

% -4200rpm ~ -4900rpm
Adv_angle_4_neg = 11*pi/180;                        % Advanced angle = 11 degrees;
Comm_angle_4_neg = 11*pi/180;                      % Commutation angle = 11 degrees;

% -4900rpm ~ -6000rpm
Adv_angle_5_neg = 14*pi/180;                        % Advanced angle = 11 degrees;
Comm_angle_5_neg = 14*pi/180;                      % Commutation angle = 11 degrees;

% Reference speed
ref_rpm = 5000;
ref_rps = 2*pi*ref_rpm/60;
omega_r_star = (ref_rps/omega_max)*omega_max_volt;            % Speed -> Volt.
omega_r_star_neg = -omega_r_star;

% Initial conditions at selfstartup
```

```
start_flag = 0;

vv_a_on = v_dc;    ia_on = v_dc/main_R;
vv_c_on = 180;                    % initial capacitor charge voltage must be greater than v_dc to flow currents into the auxiliary winding
vv_b_on=vv_c_on - v_dc;
ib_on = vv_b_on/aux_R;  ic_on = ib_on;
omega_m_on=0;

delta_La = 0;  delta_Lb = 0;  delta_theta = 0;

deg_for_spd_con_start = 3;           % degrees for starting speed control after self-starting --> 3 degrees

init_spd_rpm = 0;
init_spd_rps = 2*pi*init_spd_rpm/60;

xx(1) = theta;                          % theta=0
xx(2) = init_spd_rps;              % speed before filtering
xx(3) = xx(2)*H_w;                 % speed after filtering
xx(4) = 0;                               % dot_xx(4)=omega_r_star - omega_mr=0
xx(5) = ia_on;                        % main winding current before filtering
xx(6) = xx(5)*H_i;                  % current after filtering
xx(7) = 0;                                % dot_xx(7)=i_star_a*H_i - xx(6)=0
xx(8) = ib_on;                          % auxiliary current
xx(9) = vv_c_on;                     % capacitor voltage

time_size=1:steps;
spd_size=1:steps/100;

      all_t = zeros(size(time_size));                                % time vector
      all_t_spd = zeros(size(spd_size));                          % speed controller size
     all_theta = zeros(size(time_size));                          % real position in radian
     all_theta_deg = zeros(size(time_size));                  % real position in degrees
      all_omega_m_rpm = zeros(size(time_size));          % real speed in rpm
     all_omega_err = zeros(size(time_size));                  % speed error in rpm
     all_T_e_star = zeros(size(spd_size));                      % torque command in Nm
     all_La = zeros(size(time_size));                               % main winding real inductance in H
     all_Lb = zeros(size(time_size));                               % auxiliary winding real inductance in H
     all_ia = zeros(size(time_size));                                % main winding current in Amp
     all_ib = zeros(size(time_size));                                % auxiliary winding current in Amp
     all_ic = zeros(size(time_size));                                % capacitor current in Amp
     all_i_star_a = zeros(size(time_size));                     % main winding current command in Amp
     all_vv_a = zeros(size(time_size));                            % main winding voltage in Volt
     all_vv_b = zeros(size(time_size));                            % auxiliary winding voltage in Volt
      all_vv_c = zeros(size(time_size));                            % capacitor voltage in Volt
     all_T_e = zeros(size(time_size));                              % real torque in Nm
     all_back_emf_a = zeros(size(time_size));                 % main winding back-emf in Volt
     all_back_emf_b = zeros(size(time_size));                 % auxiliary winding back-emf in Volt
     all_PWM = zeros(size(time_size));                            % PWM duty cycle

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MAIN PROGRAM
while t <= t_end
   % ticking away, for the moments decay...
    t = t+t_step;
    c = c+1;
    all_t(c) = t;

     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % GETTING REAL POSITION AND SPEED
     % computation of magnitudes
    theta = mod(xx(1),2*pi);
    theta_deg = theta*180/pi;
    omega_m_rpm = xx(2)*60/(2*pi);                    % rad/s --> rpm
   if direction_flag == 1 & t >= t_break_2 & omega_m_rpm > 0
          break;
```

```
 end

all_theta(c) = theta;
 all_theta_deg(c) = theta_deg;                % rad --> deg
all_omega_m_rpm(c) = omega_m_rpm;

% GETTING REAL CURRENT AND INDUCTANCE
La = lookup(main_induct,theta,xx(5));     % look up inductance in table, main winding
Lb = lookup(aux_induct,theta,xx(8));      % look up inductance in table, auxiliary winding

all_La(c) = La;
all_Lb(c) = Lb;
all_ia(c) = xx(5);
all_ib(c) = xx(8);

if theta_deg > deg_for_spd_con_start
    start_flag = start_flag + 1;
 end

if start_flag >= 1                               % if theta_deg > 3 degrees, then start the speed control
    if omega_m_rpm > 0                                   % if speed is more than 0 rpm, then forward rotation starts
         direction_flag = 0;
        delta_omega = omega_r_star - xx(3);
    elseif omega_m_rpm < 0                              % if speed is lower than 0 rpm, then reverse rotation starts
         direction_flag = 1;
        delta_omega = omega_r_star_neg - xx(3);
     end
    all_omega_err(c) =  (delta_omega/H_w)*60/(2*pi);    % Volt --> rad/s --> rpm

     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % FORWARD OPERATION OF SR MOTOR
    if direction_flag == 0
         c_f = c_f + 1;
        if t < t_break_1
            c_f_m = c_f_m + 1;
             c_f_r = 0;
             c_r_m = 0;
             c_r_r = 0;
             mode = 0;             % MOTORING MODE
              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % SPEED LOOP CONTROLLER - 10kHz
            if mod(c_f_m,100) == 1
                  t_spd = t_spd + 1;
                 all_t_spd(t_spd) = t;
                int_omega = xx(4);   %delta_omega + int_omega;
               Spd_out = K_ps*delta_omega + K_is*int_omega; % unit [V]

                 % Torque Controller
                T_e_star_out = (T_e_max/omega_max_volt) * Spd_out; % Volt --> Torque
                 T_e_star = limit(T_e_star_out,-T_e_max,T_e_max);

                 all_T_e_star(t_spd) = T_e_star;

                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              % CURRENT COMMAND generator according to the position and command torque
                i_star_out = (T_e_star/T_e_max) * i_max;     % Torque --> Current

                theta_mod = mod(theta, pi/2);
               if omega_m_rpm > 0 & omega_m_rpm <= 1500
                      if i_star_out > 0                                    % if T_e_star_out > 0
                         if (theta_mod >= 0) & (theta_mod < pi/4)
                              i_star_a = 0;
                        elseif (theta_mod >= pi/4) & (theta_mod < pi/2)
                              i_star_a = abs(i_star_out);
                             i_star_a = limit(i_star_a,0,i_max);          % Phase A - positive slope inductance
                end
        elseif i_star_out < 0                                       % if T_e_star_out < 0
            if (theta_mod >= 0) & (theta_mod < pi/4)
                i_star_a = abs(i_star_out);
                i_star_a = limit(i_star_a,0,i_max);             % Phase A  - negative slope inductance
            elseif (theta_mod >= pi/4) & (theta_mod < pi/2)
                i_star_a = 0;
            end
        end
    elseif omega_m_rpm > 1500 & omega_m_rpm <= 2500
        if i_star_out > 0
            if (theta_mod >= 0  & theta_mod < pi/4  - Adv_angle_1) | (theta_mod >= pi/2 - Comm_angle_1 & theta_mod < pi/2)
                i_star_a = 0;
            elseif (theta_mod >= pi/4  - Adv_angle_1) & (theta_mod < pi/2 - Comm_angle_1)
                i_star_a = abs(i_star_out);
                i_star_a = limit(i_star_a,0,i_max);             % Phase A  - positive slope inductance
            end
        elseif i_star_out < 0
            if (theta_mod >= 0) & (theta_mod < pi/4)
                i_star_a = abs(i_star_out);
                i_star_a = limit(i_star_a,0,i_max);             % Phase A  - negative slope inductance
            elseif (theta_mod >= pi/4) & (theta_mod < pi/2)
                i_star_a = 0;
            end
        end
    elseif omega_m_rpm > 2500 & omega_m_rpm <= 3500
        if i_star_out > 0
            if (theta_mod >= 0  & theta_mod < pi/4  - Adv_angle_2) | (theta_mod >= pi/2 - Comm_angle_2 & theta_mod < pi/2)
                i_star_a = 0;
            elseif (theta_mod >= pi/4  - Adv_angle_2) & (theta_mod < pi/2 - Comm_angle_2)
                i_star_a = abs(i_star_out);
                i_star_a = limit(i_star_a,0,i_max);             % Phase A  - positive slope inductance
            end
        elseif i_star_out < 0
            if (theta_mod >= 0) & (theta_mod < pi/4)
                i_star_a = abs(i_star_out);
                i_star_a = limit(i_star_a,0,i_max);             % Phase A - negative slope inductance
            elseif (theta_mod >= pi/4) & (theta_mod < pi/2)
                i_star_a = 0;
            end
        end
    elseif omega_m_rpm > 3500 & omega_m_rpm <= 4200
        if i_star_out > 0
            if (theta_mod >= 0  & theta_mod < pi/4  - Adv_angle_3) | (theta_mod >= pi/2 - Comm_angle_3 & theta_mod < pi/2)
                i_star_a = 0;
            elseif (theta_mod >= pi/4  - Adv_angle_3) & (theta_mod < pi/2 - Comm_angle_3)
                i_star_a = abs(i_star_out);
                i_star_a = limit(i_star_a,0,i_max);             % Phase A  - positive slope inductance
            end
        elseif i_star_out < 0
            if (theta_mod >= 0) & (thet a_mod < pi/4)
                i_star_a = abs(i_star_out);
                i_star_a = limit(i_star_a,0,i_max);             % Phase A  - negative slope inductance
            elseif  (theta_mod >= pi/4) & (theta_mod < pi/2)
                i_star_a = 0;
            end
        end
    elseif omega_m_rpm > 4200 & omega_m_rpm <= 4900
        if i_star_out > 0
            if (theta_mod >= 0  & theta_mod < pi/4  - Adv_angle_4) | (theta_mod >= pi/2 - Comm_angle_4 & theta_mod < pi/2)
                i_star_a = 0;
            elseif (theta_mod >= pi/4  - Adv_angle_4) & (theta_mod < pi/2 - Comm_angle_4)
                i_star_a = abs(i_star_out);
                i_star_a = limit(i_star_a,0,i_max);             % Phase A  - positive slope inductance
            end
```

```
                    elseif i_star_out < 0
                        if (theta_mod >= 0) & (theta_mod < pi/4)
                              i_star_a = abs(i_star_out);
                             i_star_a = limit(i_star_a,0,i_max);          % Phase  A  - negative slope inductance
                        elseif (theta_mod >= pi/4) & (theta_mod < pi/2)
                              i_star_a = 0;
                          end
                      end
              elseif omega_m_rpm > 4900 & omega_m_rpm <= 6000
                   if i_star_out > 0
                        if (theta_mod >= 0  & theta_mod < pi/4  - Adv_angle_5) | (theta_mod >= pi/2 - Comm_angle_5 & theta_mod < pi/2)
                              i_star_a = 0;
                       elseif (theta_mod >= pi/4  - Adv_angle_5) & (theta_mod < pi/2  - Comm_angle_5)
                              i_star_a = abs(i_star_out);
                            i_star_a = limit(i_star_a,0,i_max);           % Phase  A  - positive slope inductance
                          end
                    elseif i_star_out < 0
                        if (theta_mod >= 0) & (theta_mod < pi/4)
                              i_star_a = abs(i_star_out);
                            i_star_a = limit(i_star_a,0,i_max);           % Phase  A  - negative slope inductance
                        elseif (theta_mod >= pi/4) & (theta_mod < pi/2)
                               i_star_a = 0;
                          end
                      end
                  end
              end
      elseif t >= t_break_1
            c_f_m = 0;
           c_f_r = c_f_r + 1;
            c_r_m = 0;
            c_r_r = 0;
            mode = 1;                % REGENERATION MODE

           if mod(c_f_r,100) == 1
                t_spd = t_spd + 1;
                 all_t_spd(t_spd) = t;
                 all_T_e_star(t_spd) = 0;
             end

            theta_mod = mod(theta, pi/2);
           if (theta_mod >= 0) & (theta_mod < pi/4)
                  i_star_a = i_max;                                    % Phase  A  - negaitive slope inductance
          elseif (theta_mod >= pi/4) & (theta_mod < pi/2)
                   i_star_a  = 0;                                        % Phase  A  - zero current
             end
       end
      all_i_star_a(c) = i_star_a;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % REVERSE OPERATION OF SR MOTOR
elseif direction_flag == 1
     c_r = c_r + 1;
      if t < t_break_2
            c_f_m = 0;
            c_f_r = 0;
           c_r_m= c_r_m + 1;            % speed controller loop counter  in reverse operation
            c_r_r = 0;
           mode = 0;            % MOTORING AND SPEED CONTROL MODE
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
           % SPEED LOOP CONTROLLER  - 10kHz
             if mod(c_r_m,100) == 1
                t_spd = t_spd + 1;
                 all_t_spd(t_spd) = t;

                int_omega = xx(4);    %delta_omega + int_omega;
  Spd_out = K_ps*delta_omega + K_is*int_omega; % unit [V]

   % Torque Controller
 T_e_star_out = (T_e_max/omega_max_volt) * Spd_out; % Volt -> Torque
  T_e_star = limit(T_e_star_out,-T_e_max,T_e_max);

  all_T_e_star(t_spd) = T_e_star;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CRRENT COMMAND generator according to the position and command torque
  i_star_out = (T_e_star/T_e_max) * i_max;        % Toruqe -> Current

  theta_mod = mod(theta, pi/2);

  if omega_m_rpm >= -1500 & omega_m_rpm < 0
       if i_star_out < 0
         if theta_mod >= 0 & theta_mod < pi/4
                i_star_a = abs(i_star_out);                            % Reverse Motoring
                i_star_a = limit(i_star_a,0,i_max);                   % Phase A - positive slope inductance
          elseif theta_mod >= pi/4 & theta_mod < pi/2
                i_star_a = 0;
            end
      elseif i_star_out > 0
           if theta_mod >= 0 & theta_mod < pi/4
                i_star_a = 0;
          elseif theta_mod > pi/4 & theta_mod < pi/2
                i_star_a = abs(i_star_out);                            % Reverse Braking
               i_star_a = limit(i_star_a,0,i_max);                    % Phase A - negative slope inductance
            end
        end
  elseif omega_m_rpm >= -2500 & omega_m_rpm < -1500
      if i_star_out < 0
        if (theta_mod >= 0 + Comm_angle_1_neg) & (theta_mod < pi/4 + Adv_angle_1_neg)
                i_star_a = abs(i_star_out);                            % Reverse Motoring
               i_star_a = limit(i_star_a,0,i_max);                    % Phase A - positive slope inductance
       elseif (theta_mod >= 0 & theta_mod <  Comm_angle_1_neg) | (theta_mod >= pi/4 + Adv_angle_1_neg & theta_mod < pi/2)
                i_star_a = 0;
            end
      elseif i_star_out > 0
         if theta_mod >= 0 & theta_mod < pi/4
                i_star_a = 0;
            elseif theta_mod > pi/4 & theta_mod < pi/2
                i_star_a = abs(i_star_out);                            % Reverse Braking
               i_star_a = limit(i_star_a,0,i_max);                    % Phase A - negative slope inductance
            end
        end
  elseif omega_m_rpm >= -3500 & omega_m_rpm < -2500
      if i_star_out < 0
         if (theta_mod >= 0 + Comm_angle_2_neg) & (theta_mod < pi/4 + Adv_angle_2_neg)
                i_star_a = abs(i_star_out);                            % Reverse Motoring
               i_star_a = limit(i_star_a,0,i_max);                    % Phase A - positive slope inductance
         elseif (theta_mod >= 0 & theta_mod <  Comm_angle_2_neg) | (theta_mod >= pi/4 + Adv_angle_2_neg & theta_mod < pi/2)
                i_star_a = 0;
            end
      elseif i_star_out > 0
         if theta_mod >= 0 & theta_mod < pi/4
                i_star_a = 0;
          elseif theta_mod > pi/4 & theta_mod < pi/2
                 i_star_a = abs(i_star_out);                           % Reverse Braking
               i_star_a = limit(i_star_a,0,i_max);                    % Phase A - negative slope inductance
            end
        end
  elseif omega_m_rpm >= -4200 & omega_m_rpm < -3500
      if i_star_out < 0
        if (theta_mod >= 0 + Comm_angle_3_neg) & (theta_mod < pi/4 + Adv_angle_3_neg)
```

```
                    i_star_a = abs(i_star_out);                    % Reverse Motoring
                   i_star_a = limit(i_star_a,0,i_max);           % Phase A - positive slope inductance
              elseif (theta_mod >= 0 & theta_mod <  Comm_angle_3_neg) | (theta_mod >= pi/4 + Adv_angle_3_neg & theta_mod <pi/2)
                    i_star_a = 0;
                 end
           elseif i_star_out > 0
              if theta_mod >= 0 & theta_mod < pi/4
                    i_star_a = 0;
               elseif theta_mod > pi/4 & theta_mod < pi/2
                    i_star_a = abs(i_star_out);                    % Reverse Braking
                   i_star_a = limit(i_star_a,0,i_max);           % Phase A - negative slope inductance
                 end
              end
        elseif omega_m_rpm >= -4900 & omega_m_rpm < -4200
           if i_star_out < 0
              if (theta_mod >= 0 + Comm_angle_4_neg) & (theta_mod < pi/4 + Adv_angle_4_neg)
                    i_star_a = abs(i_star_out);                    % Reverse Motoring
                   i_star_a = limit(i_star_a,0,i_max);           % Phase A - positive slope inductance
             elseif (theta_mod >= 0 & theta_mod <  Comm_angle_4_neg) | (theta_mod >= pi/4 + Adv_angle_4_neg & theta_mod < pi/2)
                    i_star_a = 0;
                 end
              elseif i_star_out>0
              if theta_mod >= 0 & theta_mod < pi/4
                    i_star_a = 0;
               elseif theta_mod > pi/4 & theta_mod < pi/2
                    i_star_a = abs(i_star_out);                    % Reverse Braking
                   i_star_a = limit(i_star_a,0,i_max);           % Phase A - negative slope inductance
                 end
              end
        elseif omega_m_rpm >= -6000 & omega_m_rpm < -4900
           if i_star_out < 0
             if (theta_mod >= 0 + Comm_angle_5_neg) & (theta_mod < pi/4 + Adv_angle_5_neg)
                    i_star_a = abs(i_star_out);                    %Reverse Motoring
                   i_star_a = limit(i_star_a,0,i_max);           % Phase A - positive slope inductance
              elseif (theta_mod >= 0 & theta_mod <  Comm_angle_5_neg) | (theta_mod >= pi/4 + Adv_angle_5_neg & theta_mod < pi/2)
                    i_star_a = 0;
                 end
           elseif i_star_out > 0
              if theta_mod >= 0 & theta_mod < pi/4
                    i_star_a = 0;
               elseif theta_mod > pi/4 & theta_mod < pi/2
                    i_star_a = abs(i_star_out);                    % Reverse Braking
                    i_star_a = limit(i_star_a,0,i_max);           % PhaseA - negative slope inductance
                 end
              end
           end
     end
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif t >= t_break_2           % BRAKING MODE
     c_f_m = 0;
     c_f_r = 0;
     c_r_m = 0;
     c_r_r= c_r_r + 1;
     mode = 1;
    if mod(c_r_r,100) == 1
         t_spd = t_spd + 1;
          all_t_spd(t_spd) = t;
          all_T_e_star(t_spd) = 0;
     end

    theta_mod = mod(theta, pi/2);
   if (theta_mod >= 0) & (theta_mod < pi/4)
          i_star_a = 0;                                          % Phase A - zero current
   elseif (theta_mod >= pi/4) & (theta_mod < pi/2)
         i_star_a = i_max;                                       % Phase A - negative slope inductance
```

```
                end
            end
            all_i_star_a(c) = i_star_a;
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       % CURRENT CONTROLLER -20kHz
     if mod(c_f_m,50) == 1 | mod(c_f_r, 50) == 1 | mod(c_r_m , 50) == 1 | mod(c_r_r, 50) == 1
            % Ia current controller
            delta_ia = i_star_a*H_i - xx(6);
             int_ia= xx(7);
           v_ca = K_pi*delta_ia + K_ii*int_ia;        % PI controller
            v_ca = limit(v_ca,0,v_c_max);                          % limiter

            v_pwm_a = v_ca/v_c_max;                      % PWM duty cycle

            if v_pwm_a > 0.95
                  v_pwm_a = 0.95;
             end

            all_PWM(c) = v_pwm_a;
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       % PWM CONTROLLER
      tri = triangle(f_ct);   % get triangle signal

      if v_pwm_a > tri
            vv_a = v_dc;
           vv_b = xx(9) - v_dc;
            ic = -xx(8);
      elseif v_pwm_a < tri
           vv_a = v_dc - xx(9);
           vv_b = xx(9) - v_dc;
           ic = xx(5) - xx(8);
        end
       all_vv_a(c) = vv_a;
      all_vv_b(c) = vv_b;
       all_vv_c(c) = xx(9);
       all_ic(c) = ic;
   else
       all_i_star_a(c) = 0;

      vv_a = v_dc - xx(9);
      vv_b = xx(9) - v_dc;
      ic = xx(5) - xx(8);

       all_vv_a(c) = vv_a;
      all_vv_b(c) = vv_b;
       all_vv_c(c) = xx(9);
       all_ic(c) = ic;
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % TORQUE GENERATION
  T_e_a = lookup(main_tor,theta,xx(5));
  T_e_b = lookup(aux_tor,theta,xx(8));
 T_e = T_e_a - T_e_b;
  all_T_e(c) = T_e;

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % LOAD TORQUE
  T_l = Load;

 if t > break_steps_3
      Load = 0.1 * T_e_max;
```

```
    elseif t > break_step_4
            Load = 0;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % DIFFERENTIAL  EQUATIONS FOR SR MACHINE
    if start_flag < 1 & delta_theta == 0
        delta_theta = theta - theta_old;
        delta_omega = 0;
        i_star_a = 0;

        xx(1) = xx(1) + xx(2)*t_step;                                    % d_theta/dt
        xx(2) = xx(2) + (1/J)*(T_e -B*xx(2) - T_l)*t_step;               % d_omega_m/dt
        xx(3) = H_w*xx(2);                                               % omega_mr =
H_w*omega_m
        xx(4) = xx(4) + delta_omega*t_step;                              % xx(4) = command speed  -
omega_mr
        xx(5) = xx(5) + (1/La)*(vv_a - main_R*xx(5))*t_step;   % d_ia/dt
        xx(6) = xx(6) + (1/T_i)*(H_i*xx(5) - xx(6))*t_step;              % d_ima/dt
        xx(7) = 0;    %xx(7) + (i_star_a*H_i  - xx(6))*t_step;           % xx(9) = command
current - ima
        xx(8) = xx(8) + (1/Lb)*(vv_b - aux_R*xx(8))*t_step;    % d_ib/dt
        xx(9) = xx(9) + (ic/Cap)*t_step;                                 % capacitor voltage

        if xx(9) < 100
            xx(9) = 100;
        elseif xx(9) > 300
            xx(9) = 300;
        end

        % Store Back-EMF
        all_back_emf_a(c) = 0;   %xx(2)*xx(5)*(delta_La/delta_theta);    % Back EMF of main winding
        all_back_emf_b(c) = 0;   %xx(2)*xx(8)*(delta_Lb/delta_theta);    % Back EMF of auxiliary
winding

        La_old = La;
        Lb_old = Lb;
        theta_old = theta;
    elseif (start_flag < 1 & delta_theta ~= 0) | start_flag >= 1
        delta_La = La - La_old;
        delta_Lb = Lb - Lb_old;
        delta_theta = theta - theta_old;

        xx(1) = xx(1) + xx(2)*t_step;                                    % d_theta/dt
        xx(2) = xx(2) + (1/J)*(T_e -B*xx(2) - T_l)*t_step;               % d_omega_m/dt
        xx(3) = H_w*xx(2);                                               % omega_mr =
H_w*omega_m
        xx(4) = xx(4) + delta_omega*t_step;                              % xx(4) = command speed  -
omega_mr
        xx(5) = xx(5) + (1/La)*(vv_a -main_R*xx(5) - xx(2)*xx(5)*(delta_La/delta_theta))*t_step;   % d_ia/dt
        xx(6) = xx(6) + (1/T_i)*(H_i*xx(5) - xx(6))*t_step;              % d_ima/dt
        xx(7) = xx(7) + (i_star_a*H_i - xx(6))*t_step;                   % xx(9) = command current - ima
        xx(8) = xx(8) + (1/Lb)*(vv_b -aux_R*xx(8) - xx(2)*xx(8)*(delta_Lb/delta_theta))*t_step;   % d_ib/dt
        xx(9) = xx(9) + (ic/Cap)*t_step;                                 % capacitor voltage

        if xx(9) < 100
            xx(9) = 100;
        elseif xx(9) > 300
            xx(9) = 300;
        end

        % Store Back-EMF
        all_back_emf_a(c) = xx(2)*xx(5)*(delta_La/delta_theta);          % Back EMF of main winding
        all_back_emf_b(c) = xx(2)*xx(8)*(delta_Lb/delta_theta);          % Back EMF of auxiliary winding
        La_old = La;
        Lb_old = Lb;
        theta_old = theta;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % DISPLAY THE ELAPSED SIMULATION TIME
   count = count + 1;
    percent_new = (count/steps)*100;
   if mod(percent_new,1) == 0
        clc;
         percent_new
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLOTTING

figure(1)
subplot(5,1,1), plot(all_t(1:c), all_omega_m_rpm(1:c)/5000,'k'), axis([0 4 -2 2]), grid on

subplot(5,1,2), plot(all_t_spd(1:t_spd), all_T_e_star(1:t_spd)/2,'k'), axis([0 4 -2 2]),   grid on

subplot(5,1,3), plot(all_t(1:c), all_T_e(1:c)/2,'k'), axis([0 4 -2 2]), grid on

subplot(5,1,4), plot(all_t(1:c), all_i_star_a(1:c)/8,'k'), axis([0 4 -2 2]), grid on

subplot(5,1,5), plot(all_t(1:c),all_ia(1:c)/8,'k'), axis([0 4 -2 2]), grid on
```

VITA

Sung Yeul Park was born February 1, 1973 in Seoul Korea. He graduated Hoseo University in February of 1998 with a Bachelor of Science in Control and Instrument Engineering. He joined the M.S program at Virginia Polytechnic Institute and State University in fall, 2002. After the completion of his master's degree, he will pursue his Ph.D. His research interests include: Switched reluctance motor drives, Power Electronics, and Micro-controller Application Systems.