# Sensorless PMSM Vector Control with a Sliding Mode Observer for Compressors Using MC56F8013

**Devices Supported:**

**MC56F8013**
**MC56F8023**

Document Number: DRM099
Rev. 0
09/2008

## *How to Reach Us:*

**Home Page:**
www.freescale.com

**E-mail:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 26668334
support.asia@freescale.com

**For Literature Requests Only:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

DRM099
Rev. 0
09/2008

# Chapter 1
# Introduction

# Chapter 5
# Software Design

# Chapter 1
# Introduction

## 1.1 Introduction

This document describes the design of a three-phase sensorless PMSM vector control drive with a sliding mode observer (SMO). The design is targeted mainly at compressor control and other consumer and industrial applications. This cost-effective solution benefits from Freescale Semiconductor MC56F8013 device dedicated for motor control and the MC56F8346.

### 1.1.1 Application Features and Components

The system is designed to drive a three-phase PM synchronous motor. Application features:

- Three-phase sensorless PMSM speed vector (FOC) control
- Sliding mode observer with adaptive velocity estimation
- Based on Freescale MC56F8013 (resp. 56F8346) controller
- Running on a three-phase high voltage (230/115V) power stage
- FreeMASTER software control interface and monitor

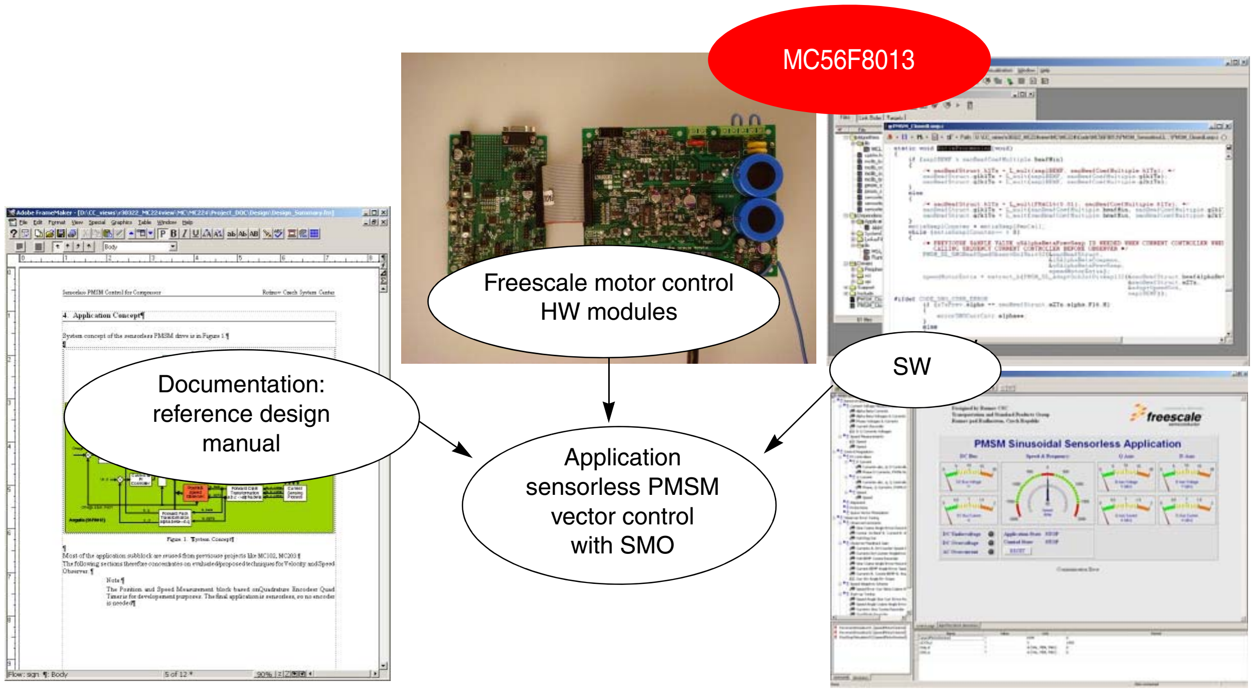

MC56F8013
Freescale motor control HW modules
SW
Documentation: reference design manual
Application sensorless PMSM vector control with SMO

**Figure 1-1. Application Sensorless PM Synchronous Motor Vector Control using MC56F8013**

Main application components available for customers are:

- S/W — written in C-code using a few library algorithms available for the MC56F8013 and MC56F8346
- H/W — based on Freescale universal motor control h/w modules
- Documentation — this document

### 1.1.2 Sinusoidal PM Synchronous Motors Applications Overview

Sinusoidal PM synchronous motors have become more popular for new drives. They replace brushed DC, universal, and other motors in a wide application area. It has better reliability (no brushes), better efficiency, lower acoustic noise, and other benefits for electronic control. A disadvantage of PM synchronous motor drives may be the need for a more sophisticated electronic circuit. But today, applications need electronic speed, torque regulation, and other features that need electronic control. One example is, a variable speed compressor for a refrigerator or air-conditioning applications. After a system with electronic control is used, it has a small system cost increase to implement more advanced drives like the sinusoidal PM synchronous motor with digitally controlled switching inverter and a DC-bus circuit. It is necessary only to have a cost effective controller with a good calculation performance. One of them is the Freescale MC56F8013, or similar devices like the MC56F802x, MC56F803x, or MC56F8346.

The PM synchronous motor also has advantages if compared to an AC induction motor. Because a PM synchronous motor achieves higher efficiency by generating the rotor magnetic flux with rotor magnets, it is used in refrigerators, washing machines, dishwashers, pumps, fans, and in other appliances that require high reliability and efficiency.

Three phase synchronous motors with permanent magnets come in two popular variants. The sinusoidal PM synchronous motor and the trapezoidal BLDC motor. The sinusoidal PM synchronous motor is similar to the trapezoidal BLDC Electronically Commuted motor. There are two main differences:

- Motor construction
  - The shape of the BEMF inducted voltage — sinusoidal PM synchronous motor versus trapezoidal BLDC motor
- Control — shape of the control voltage
  - Three-phase sinusoidal, all three-phases connected at one time, versus rectangular six-step commutation, one phase is non-conducting at any time.

Generally, it is said that the sinusoidal PM synchronous motor performance is better, due to constant torque and the trapezoidal BLDC motor can easily be controlled. The trapezoidal BLDC motors are mainly used for historical reasons. It is easier to create a six-step commutation and estimate the rotor position with simpler algorithms because one phase is non-conducting. The sinusoidal PM synchronous motors require more control and has some benefits, such as smoother torque and lower acoustic noise. As shown in this document, the MC56F8013 provides all the necessary functionalities for sensorless three-phase sinusoidal PM synchronous motor vector control. If using the MC56F8013, it can replace the trapezoidal BLDC electronically commuted motors with the three-phase sinusoidal PM synchronous motors with almost no system cost increase.

Described here is speed vector control. The speed control algorithms can be sorted into two general groups. The first group is referred to as scalar control. The constant volt per hertz control is a popular technique that represents scalar control. The other group is called vector or field oriented control (FOC). The vector oriented techniques bring overall improvements to drive performance over scalar control. For higher efficiency; full torque control, decoupled control of flux and torque, improved dynamics.

For the PM synchronous motor control, it is necessary to estimate the rotor position. A good error filtering, low angle error, and dynamic performance can be obtained with feedback observers. The sliding mode observer (SMO) is one of the observers with a system model suitable for PM synchronous motors. There are a few techniques on how to estimate the rotor angular speed. The technique with an SMO and an adaptive speed scheme uses a system model where the system coefficients are calculated from the estimated speed. A benefit of such solution is low error ripple with no necessity for low pass filtering.

The reference design manual describes the basic motor theory, the system design concept, hardware implementation, and the software design, including the FreeMASTER software visualization tool.

## 1.2 Freescale Controller Advantages and Features

The Freescale MC56F80xx family is well suited for digital motor control. It combines the DSP’s calculation capability with the MCU’s controller features on a single chip. These hybrid controllers offer many dedicated peripherals such as pulse width modulation (PWM) modules, analogue-to-digital converters (ADC), timers, communication peripherals (SCI, SPI, IIC), and on-board flash and RAM.

The MC56F80xx family members provide the following peripheral blocks:

- One PWM module with PWM outputs, fault inputs, fault-tolerant design with dead time insertion, supporting both centre-aligned, and edge-aligned modes
- 12-bit ADC, supporting two simultaneous conversions. Both ADC and PWM modules can be synchronized.
- One dedicated 16-bit general purpose quad timer module
- One serial peripheral interface (SPI)
- One serial communications interface (SCI) with LIN slave functionality
- One inter-integrated circuit (IIC) port
- On-board 3.3 V to 2.5 V voltage regulator for powering internal logic and memories
- Integrated power-on reset and low voltage interrupt module
- All pins multiplexed with general purpose input/output (GPIO) pins
- Computer operating properly (COP) watchdog timer
- External reset input pin for hardware reset
- JTAG/On-Chip Emulation (OnCE™) module for unobtrusive processor-speed-independent debugging
- Phase-locked loop (PLL) based frequency synthesizer for the hybrid controller core clock with on-chip relaxation oscillator

**Table 1-1. Memory Configuration**

| Memory Type | MC56F8013 |
|---|---|
| Program flash | 16 KByte |
| Unified data/program RAM | 4 KByte |

The sensorless PMSM vector control with a sliding mode observer benefits greatly from the flexible PWM module, fast ADC, and quad timer module.

The PWM offers flexibility in its configuration, enabling efficient three-phase motor control. The PWM module is capable of generating asymmetric PWM duty cycles in centre-aligned configuration. A benefit from this feature is to achieve a reconstruction of three-phase currents in critical switching patterns. The PWM reload SYNC signal is generated to provide synchronization with other modules (Quadtimers, ADC).

The PWM block has the following features:

- Three complementary PWM signal pairs, six independent PWM signals, or a combination
- Complementary channel operation features
- Independent top and bottom dead time insertion
- Separate top and bottom pulse width correction via current status inputs or software
- Separate top and bottom polarity control
- Edge-aligned or centre-aligned PWM reference signals
- 15-bit resolution
- Half-cycle reload capability
- Integral reload rates from one to sixteen periods
- Mask/swap capability
- Individual software-controlled PWM output
- Programmable fault protection
- Polarity control
- 10 mA or 16 mA current sink capability on PWM pins
- Write-protectable registers

The ADC module has the following features:

- 12-bit resolution
- Dual ADCs per module and three input channels per ADC
- Maximum ADC clock frequency of 5.33 MHz with a 187 ns period
- Sampling rate of up to 1.78 million samples per second
- Single conversion time of 8.5 ADC clock cycles (8.5 x 187 ns = 1.59 ms)
- Additional conversion time of six ADC clock cycles (6 x 187 ns = 1.125 ms)
- Eight conversions in 26.5 ADC clock cycles (26.5 x 187 ns = 4.97 ms) using parallel mode

- Ability to use the SYNC input signal to synchronize with the PWM, provided the integration allows the PWM to trigger a timer channel connected to the SYNC input
- Ability to sequentially scan and store up to eight measurements
- Ability to scan and store up to four measurements on each of two ADCs operating simultaneously and in parallel
- Ability to scan and store up to four measurements on each of two ADCs operating asynchronously to each other in parallel
- Interrupt generating capabilities at the end of a scan when an out-of-range limit is exceeded and on a zero crossing
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single-ended or differential inputs
- PWM outputs with hysteresis for three of the analogue inputs

The application uses the ADC block in simultaneous mode scan. It is synchronized to the PWM pulses. This configuration allows the simultaneous conversion of the required analogue values for the DC-bus current and voltage within the required time.

The quad timer is an extremely flexible module providing all required services relating to time events. It has the following features:

- Four 16-bit counters/timers
- Count up/down
- Counters are cascadable
- Programmable count modulus
- Maximum count rate equal to the peripheral clock/2 when counting external events
- Maximum count rate equal to the peripheral clock/1 when using internal clocks
- Count once or repeatedly
- Counters are preloadable
- Counters can share available input pins
- Each counter has a separate prescaler
- Each counter has capture and compare capability

The application uses four channels of the quad timer for:

- One channel for PWM-to-ADC synchronization
- Two channels for reading quadrature encoder signals
- One channel for system base of slow control loop (1 ms period)

## 1.3 Front Matter

### 1.3.1 Preface

### 1.3.2 Bibliography

1. *56F8013 Data Sheet, MC56F8013,* Freescale Semiconductor, 2006
2. *56F8346 Data Sheet, MC56F8023,* Freescale Semiconductor, 2006
3. *56F802X and 56F803X Peripheral Reference Manual,* MC56F80XXRM, Freescale Semiconductor, 2006
4. *56F801X Peripheral Reference Manual, MC56F8000RM,* Freescale Semiconductor, 2006
5. *DSP56800E Reference Manual, DSP56800ERM,* Freescale Semiconductor, 2005
6. *CodeWarrior™ Development Studio for Freescale™ 56800/E Digital Signal Controllers,* Freescale Semiconductor, 2006
7. *MC56F8013/23 Controller Board Users Manual,* Freescale Semiconductor, 2007
8. *MC56F8346 Controller Board Hardware User's Manual,* Freescale Semiconductor
9. *3 ph AC/BLDC High-Voltage Power Stage User Manual,* Freescale Semiconductor, 2007
10. *Free Master Software Users Manual,* Freescale Semiconductor, 2004
11. *New Adaptive Sliding Observers for Position and Velocity-Sensorless Controls of Brushless DC Motors,* by Zhiqian Chen, Matuwo Tomita, Shinji Doki, Shigero Okuma., IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 47, NO. 3, JUNE 2000
12. *Sensorless Vector and Direct Torque Control,* by Vas P., Oxford University Press, ISBN 0-19-856465-1, New York, 1998
13. *3-Phase PM Synchronous Motor Vector Control using DSP56F80x,* by Prokop L., Grasblum P., *AN1931— 3-Phase PM Synchronous Motor Vector Control* Motorola, 2002

For a current list of documentation, go to www.freescale.com.

### 1.3.3 Acronyms and Abbreviations

Table 1-2 contains sample acronyms and abbreviations used in a document.

**Table 1-2. Acronyms and Abbreviated Terms**

| Term | Meaning |
|---|---|
| AC | Alternating current |
| ADC | Analog-to-digital converter |
| BEMF | Back electromagnetic force = induced voltage |
| BLDC | Brushless direct current motor |
| COP | Computer operating properly (watchdog timer) |
| DC | Direct current |

Table 1-2. Acronyms and Abbreviated Terms (continued)

| Term | Meaning |
|---|---|
| DSC | Digital signal controller |
| DT | Dead time: a short time that must be inserted between the turning off of one transistor in the inverter half bridge and turning on of the complementary transistor due to the limited switching speed of the transistors. |
| FOC | Field oriented control |
| GPIO | General purpose input/output |
| I/O | Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level of an external signal. |
| JTAG | Joint test action group: acronym commonly used to refer to an interface allowing on-chip emulation and programming. |
| LED | Light emitting diode |
| MC56F80x | A Freescale family of 16-bit DSPs dedicated to motor control. |
| PI controller | Proportional-integral controller |
| PLL | Phase-locked loop: a clock generator circuit that a voltage controlled oscillator produces an oscillation synchronized to a reference signal. |
| PMSM | PM synchronous motor, permanent magnet synchronous motor |
| PWM | Pulse width modulation |
| RPM | Revolutions per minute |
| SCI | Serial communication interface module: a module that supports asynchronous communication. |
| SMO | Sliding mode observer |

## 1.3.4 Glossary of Terms

Table 1-3 shows a glossary of terms used in this document.

Table 1-3. Glossary

| Term | Definition |
|---|---|
| Brush | A component transferring electrical power, from non-rotational terminals mounted on the stator to the rotor. |
| Commutator | A mechanical device alternating DC current in a DC commutator motor and providing rotation of DC commutator motor. |
| DC/DC Inverter | Power electronics module that converts DC voltage level to a different DC voltage level. |
| Duty cycle | The ratio of the amount of time the signal is on to the time it is off. Duty cycle is usually quoted as a percentage. |
| Hall sensor | A position sensor giving six defined events (each 60 electrical degrees) per electrical revolution (for a three-phase motor). |
| Interrupt | A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine. |

Table 1-3. Glossary (continued)

| Term | Definition |
|---|---|
| PM synchronous motor | Permanent magnet synchronous motor |
| PI controller | Proportional-integral controller |
| Quick start | SW set of algorithms and drivers including graphical configuration tool for DSC/CPU initialization |
| Reset | To force a device to a known condition |
| Software | Instructions and data that control the operation of a microcontroller |

## 1.3.5 Glossary of Symbols

Table 1-4 shows a glossary of symbols used in this document.

Table 1-4. Glossary of Symbols

| Term | Definition |
|---|---|
| d,q | Rotational orthogonal coordinate system |
| d,q*, (d,q)* | Rotational orthogonal coordinate system estimated coordinates |
| $e_{Sd,q}, e_{S(d,q)}$ | BEMF (induced voltage) in d,q coordinate system |
| $e_{S\alpha,\beta}, e_{S(\alpha,\beta)}$ | BEMF (induced voltage) in a,b coordinate system |
| $e_{S0}$ | BEMF at a stand point |
| $\hat{\mathbf{e}}_{Sd,q}, \hat{\mathbf{e}}_{S(d,q)}$ | Estimated BEMF (induced voltage) in d,q coordinate system |
| $\hat{\mathbf{e}}_{S\alpha,\beta}, \hat{\mathbf{e}}_{S(\alpha,\beta)}$ | Estimated BEMF (induced voltage) in a,b coordinate system |
| $E \quad \lVert\hat{e}\rVert$ | Estimated BEMF amplitude |
| $g_1$ | SMO feedback gain real component |
| $g_1, g_2$ | To force a device to a known condition |
| $g_\omega$ | Adaptive speed scheme gain |
| $i_{Sd,q}, i_{S(d,q)}$ | Stator currents in d,q coordinate system |
| $i_{S(d,q)*}$ | Stator currents in estimated d,q coordinate system |
| $i_{S\alpha,\beta}, i_{S(\alpha,\beta)}$ | Stator currents in a,b coordinate system |
| $\hat{i}_{S\alpha,\beta}, \hat{i}_{S(\alpha,\beta)}$ | Estimated stator currents in a,b coordinate system |
| $\hat{i}_{Sd,q}, \hat{i}_{S(d,q)}$ | Estimated stator currents in d,q coordinate system |
| $\hat{i}_{S(d,q)*}$ | Estimated stator currents in estimated d,q coordinate system |
| $J$ | Mechanical inertia |
| $k_1$ | SMO switching gain |
| $K_M$ | Motor constant |

Table 1-4. Glossary of Symbols (continued)

| Term | Definition |
|---|---|
| $L_s$ | Stator phase inductance |
| $\Delta L_s$ | Stator phase inductance error |
| $L_{sd}$ | Stator phase inductance d axis |
| $L_{sq}$ | Stator phase inductance q axis |
| $p_p$ | Number of poles per phase |
| $R_s$ | Stator phase resistance |
| $\mathrm{t_e}$ | Electromagnetic torque |
| $T_L$ | Load torque |
| $u_{S\alpha,\beta}, u_{S(\alpha,\beta)}$ | Stator voltages in a,b coordinate system |
| $u_{Sd,q}$, $u_{S(d,q)}$ | Stator voltages in d,q coordinate system |
| $\alpha,\beta$ | Stator orthogonal coordinate system |
| $\varepsilon_{1(\alpha,\beta)}$ | Current estimation error in a,b coordinate system |
| $\varepsilon_{2(\alpha,\beta)}$ | BEMF estimation error error in a,b coordinate system |
| $\varepsilon_{1(d,q)}$ | Current estimation errorin d,q coordinate system |
| $\varepsilon_{2(d,q)}$ | BEMF estimation error error in d,q coordinate system |
| $\varepsilon_{2d}$ | BEMF estimation error error d component of d,q coordinate system |
| $\Psi_{S\alpha,\beta}$ | Stator magnetic fluxes in a,b coordinate system |
| $\Psi_{Sd,q}$ | Stator magnetic fluxes in d,q coordinate system |
| $\Psi_M$ | Rotor magnetic flux |
| $\Delta\theta$ | Angle estimation error |
| $\Delta\theta_{max}$ | Maximal wanted angle estimation error |
| $\theta_r$ | Rotor position angle in a,b coordinate system |
| $\omega_{el0}$ | Electrical rotor angular speed at a stand point |
| $\omega$, $\omega_{el}/\omega_F$ | Electrical rotor angular speed / fields angular speed |

# Chapter 2
Control Theory

## 2.1 Three-Phase PM Synchronous Motor

The PM synchronous motor is a rotating electric machine with a classic three-phase stator like that of an induction motor. The rotor has surface-mounted permanent magnets. See Figure 2-1.

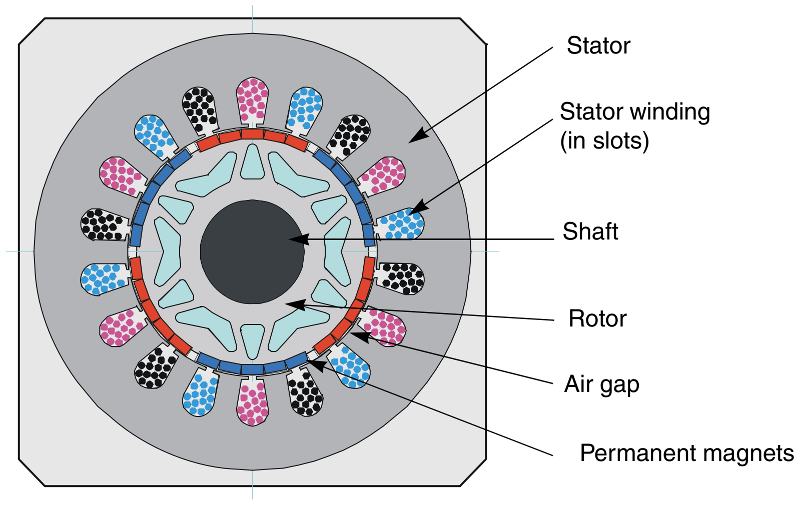


**Figure 2-1. PM Synchronous Motor — Cross Section**

The PM synchronous motor is equivalent to an induction motor. The air gap magnetic field is produced by a permanent magnet, therefore the rotor magnetic field is constant. PM synchronous motors offer a number of advantages in designing modern motion control system. The use of a permanent magnet to generate substantial air gap magnetic flux makes it possible to design highly efficient PM motors.

## 2.2 Mathematical Description of PM Synchronous Motors

There are a number of PM synchronous motor models. The model used for vector control design is obtained by using the space-vector theory. The three-phase motor quantities such as voltages, currents, and magnetic flux are expressed in terms of complex space vectors. The space vector model is valid for any instantaneous variation of voltage and current. It adequately describes the performance of the machine under both steady-state and transient operations. Complex space vectors can be described using only two orthogonal axes. The motor can be seen as a two-phase machine. Using the two-phase motor model reduces the number of equations and simplifies the control design.

### 2.2.1 Space Vector Definitions

In this example, $i_{sa}$, $i_{sb}$ , and $i_{sc}$ are the instantaneous balanced three-phase stator currents:

$$i_{sa} + i_{sb} + i_{sc} = 0 \qquad \textbf{Eqn. 2-1}$$

The stator current space vector can then be defined as follows:

$$\bar{i}_s = k(i_{sa} + a i_{sb} + a^2 i_{sc}) \qquad \textbf{Eqn. 2-2}$$

$a$ and $a^2$ are the spatial operators $a = e^{(j2\pi)/3}$ $a^2 = e^{(j-2\pi)/3}$ and $k$ are the transformation constant and is chosen $k=2/3$. Figure 2-2 shows the stator current space vector projection.

The space vector defined by Figure 2-2 can be expressed using the two-axis theory. The real part of the space vector is equal to the instantaneous value of the direct-axis stator current component, $i_{s\alpha}$ and whose imaginary part is equal to the quadrature-axis stator current component, $i_{s\beta}$. Therefore, the stator current space vector in the stationary reference frame attached to the stator can be expressed as:

$$i_s = i_{s\alpha} + j i_{s\beta} \qquad \textbf{Eqn. 2-3}$$

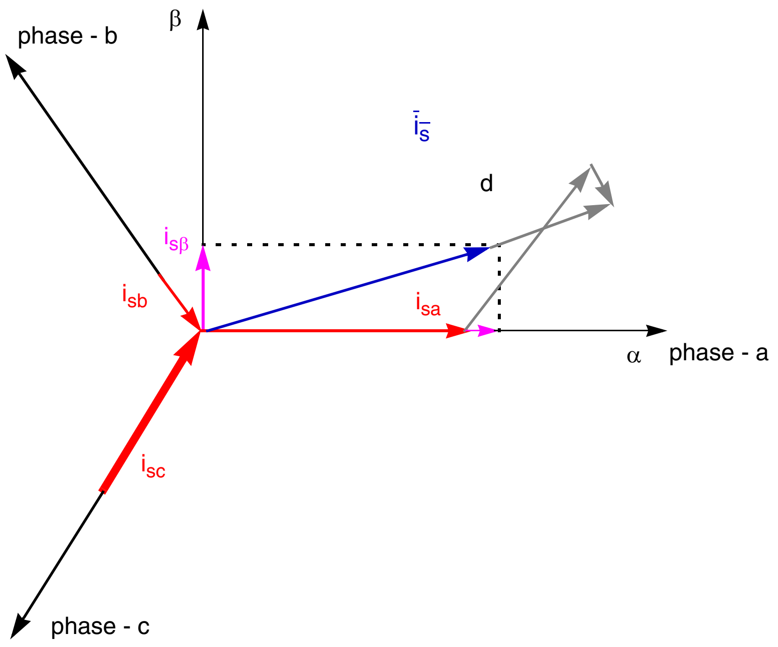


**Figure 2-2. Stator Current Space-Vector and Its Projection**

In symmetrical three-phase machines, the direct and quadrature axis stator currents $i_{s\alpha}$, and $i_{s\beta}$ are a fictitious quadrature-phase (two-phase) current components that are related to the actual three-phase stator currents as follows:

$$i_{s\alpha} = k\left(i_{sa} - \frac{1}{2}i_{sb} - \frac{1}{2}i_{sc}\right) \qquad \text{Eqn. 2-4}$$

$$i_{s\beta} = k\frac{\sqrt{3}}{2}(i_{sb} - i_{sc}) \qquad \text{Eqn. 2-5}$$

where $k=2/3$ is a transformation constant.

The space vectors of other motor quantities for example voltages, currents, and magnetic fluxes can be defined in the same way as the stator current space vector.

## 2.2.2 PM Synchronous Motor Model

For a description of the PM synchronous motor, the symmetrical three-phase smooth-air-gap machine with sinusoidally-distributed windings is elaborated. The voltage equations of stator in the instantaneous form can then be expressed as:

$$u_{SA} = R_S i_{SA} + \frac{d}{dt}\Psi_{SA} \qquad \text{Eqn. 2-6}$$

$$u_{SB} = R_S i_{SB} + \frac{d}{dt}\Psi_{SB} \qquad \text{Eqn. 2-7}$$

$$u_{SC} = R_S i_{SC} + \frac{d}{dt}\Psi_{SC} \qquad \text{Eqn. 2-8}$$

The $u_{SA}$, $u_{SB}$ and $u_{SC}$ are the instantaneous values of stator voltages. $i_{SA}$, $i_{SB,}$ and $i_{SC}$ are the instantaneous values of stator currents. $\psi_{SA}$, $\psi_{SB}$, and $\psi_{SC}$ are instantaneous values of stator flux linkages in phase SA, SB, and SC.

Due to the large number of equations in the instantaneous form Equation 2-6, Equation 2-7, and Equation 2-8, is more practical to re-write the instantaneous equations using two axis theory (Clarke transformation). The PM synchronous motor can be expressed as:

$$u_{S\alpha} = R_S i_{S\alpha} + \frac{d}{dt}\Psi_{S\alpha} \qquad \text{Eqn. 2-9}$$

$$u_{S\beta} = R_S i_{S\beta} + \frac{d}{dt}\Psi_{S\beta} \qquad \text{Eqn. 2-10}$$

$$\Psi_{S\alpha} = L_S i_{S\alpha} + \Psi_M \cos\theta r \qquad \text{Eqn. 2-11}$$

$$\Psi_{S\beta} = L_S i_{S\beta} + \Psi_M \sin\theta r \qquad \text{Eqn. 2-12}$$

$$\frac{d\omega}{dt} = \frac{p}{J}\left[\frac{3}{2}p(\Psi_{S\alpha} i_{S\beta} - \Psi_{S\beta} i_{S\alpha}) - T_L\right] \qquad \text{Eqn. 2-13}$$

For the glossary of symbols, see Section 1.3.5, “Glossary of Symbols”, Table 1-4.

Equation 2-9 through Equation 2-13 represent the model of a PM synchronous motor in the stationary frame α, β fixed to the stator.

Besides the stationary reference frame attached to the stator, motor model voltage space vector equations can be formulated in a general reference frame that rotates at a general speed $\omega_g$. If a general reference frame is used with direct and quadrature axes *x*,*y* rotating at a general instantaneous speed $\omega_g = d\theta_g/dt$ as shown in Figure 2-3. Here $\theta_g$ is the angle between the direct axis of the stationary reference frame (α) attached to the stator and the real axis (*x*) of the general reference frame. Equation 2-14 then defines the stator current space vector in the general reference frame:

$$\overline{i_{sg}} = \bar{i}_s e^{-j\theta g} = i_{sx} + j i_{sy} \qquad \textbf{Eqn. 2-14}$$

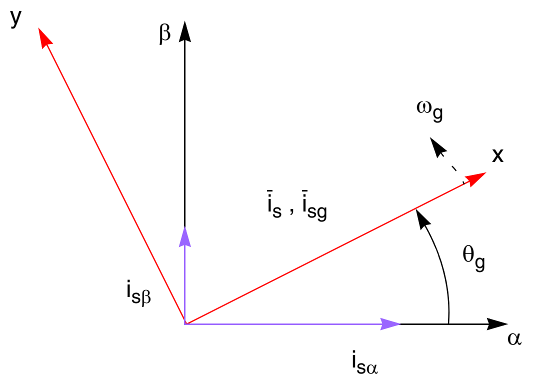


**Figure 2-3. General Reference Frame Application**

The stator voltage and flux-linkage space vectors can be similarly obtained in the general reference frame.

Similar considerations hold for the space vectors of the rotor voltages, currents, and flux linkages. The real axis (ra) of the reference frame attached to the rotor is displaced from the direct axis of the stator reference frame by the rotor angle qr. Notice that the angle between the real axis (x) of the general reference frame and the real axis of the reference frame rotating with the rotor (ra) is qg-qr in the general reference frame. The space vector of the rotor currents can be expressed as:

$$\overline{i_{rg}} = \bar{i}_r e^{-j(\theta g - \theta r)} = (i_{rx} + j i_{ry}) \qquad \textbf{Eqn. 2-15}$$

$\bar{i}_r$ is the space vector of the rotor current in the rotor reference frame.

The space vectors of the rotor voltages and rotor flux linkages in the general reference frame can be similarly expressed.

The motor model voltage equations in the general reference frame are expressed by using introduced transformations of the motor quantities from one reference frame to the general reference frame. The PM synchronous motor model is often used in vector control algorithms. The aim of vector control is to implement control schemes that produce high dynamic performance and are similar to those used to control DC machines. To achieve this, the reference frames may be aligned with the stator flux-linkage space vector, the rotor flux-linkage space vector, or the magnetizing space vector. The most popular reference frame is the reference frame attached to the rotor flux linkage space vector with direct axis (d) and quadrature axis (q).

After transformation into d, q coordinates. The motor model is as follows:

$$u_{Sd} = R_S i_{Sd} + \frac{d}{dt}\Psi_{Sd} - \omega_F \Psi_{Sq} \quad \textbf{\textit{Eqn. 2-16}}$$

$$u_{Sq} = R_S i_{Sq} + \frac{d}{dt}\Psi_{Sq} + \omega_F \Psi_{Sd} \quad \textbf{\textit{Eqn. 2-17}}$$

$$\Psi_{Sd} = L_S i_{Sd} + \Psi_M \quad \textbf{\textit{Eqn. 2-18}}$$

$$\Psi_{Sq} = L_S i_{Sq} \quad \textbf{\textit{Eqn. 2-19}}$$

$$\frac{d\omega}{dt} = \frac{p}{J}\left[\frac{3}{2}p(\Psi_{Sd} i_{Sq} - \Psi_{Sq} i_{Sd}) - T_L\right] \quad \textbf{\textit{Eqn. 2-20}}$$

By considering the below base speed $i_{sd}=0$, the Equation 2-20 can be reduced to the following form:

$$\frac{d\omega}{dt} = \frac{p}{j}\left[\frac{3}{2}p(\Psi_M i_{Sq}) - T_L\right] \quad \textbf{\textit{Eqn. 2-21}}$$

Equation 2-21, shows that the torque is dependent and can only be directly controlled by the current $i_{sq}$.

## 2.3 Vector Control of PM Synchronous Motor

### 2.3.1 Fundamental Principle of Vector Control

High-performance motor control is characterized by a smooth rotation over the entire speed range of the motor, full torque control at zero speed, fast accelerations, and decelerations. To achieve this control, vector control techniques are used for three-phase AC motors. The vector control techniques are also referred to as field-oriented control (FOC). The basic idea of the FOC algorithm is to decompose a stator current into a magnetic field-generating part and a torque-generating part. Both components can be controlled separately after decomposition. The structure of the motor controller is then as simple as for a separate DC motor.

Figure 2-4 shows the basic structure of the vector control algorithm for the PM synchronous motor. To perform vector control, it is necessary to follow these steps:

1. Measure the motor quantities, phase voltages, and currents
2. Transform them into the two-phase system (α,β) using a Clarke transformation
3. Calculate the rotor flux space-vector magnitude and position angle
4. Transform stator currents into the d, q reference frame using a Park transformation
5. The stator current torque ($i_{sq}$) and flux ($i_{sd}$) producing components are separately controlled
6. The output stator voltage space vector is calculated using the decoupling block
7. The stator voltage space vector is transformed by an inverse Park transformation back from the d, q reference frame into the two-phase system fixed with the stator
8. Using space vector modulation, the output three-phase voltage is generated

To decompose currents into torque and flux producing components ($i_{sd}$,$i_{sq}$), the position of the motor magnetizing flux is needed. This requires accurate rotor position and velocity information to be sensed. Incremental encoders or resolvers attached to the rotor are naturally used as position transducers for vector control drives. In some applications the use of speed/position sensors are not desirable. The aim is to not measure the speed/position directly, but to employ indirect techniques to estimate the rotor position instead. Algorithms that do not employ speed sensors are called sensorless control.

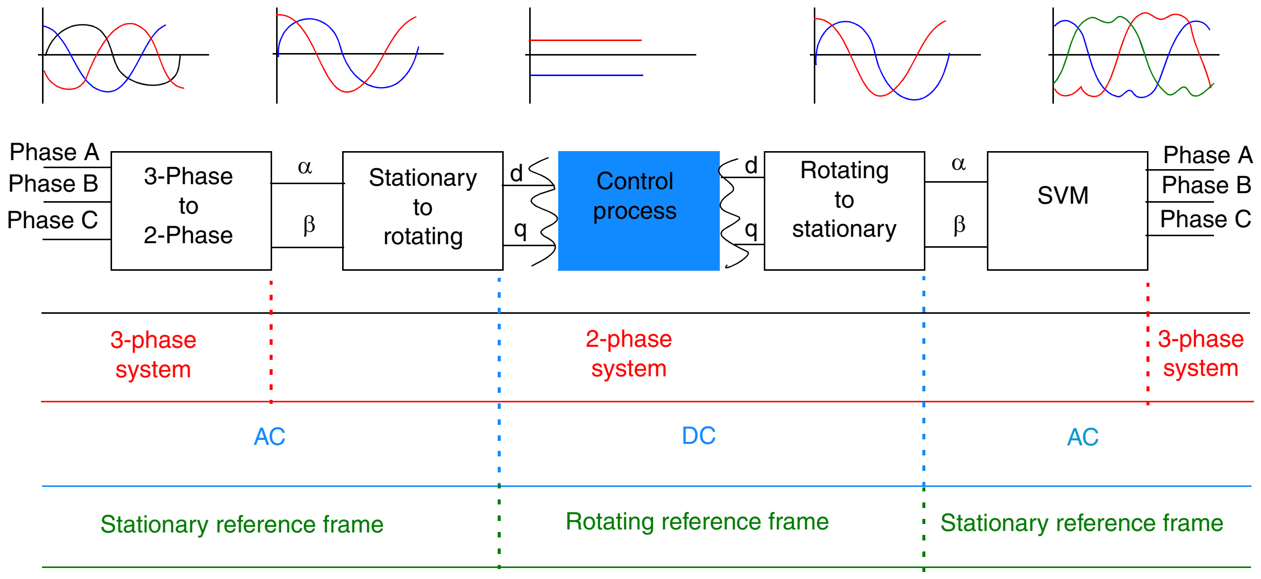


**Figure 2-4. Vector Control Transformations**

### 2.3.2 Description of the Vector Control Algorithm

The overview block diagram of the implemented control algorithm is illustrated in Figure 2-5. Similarly, as with other vector control oriented techniques, it is able to control the field and torque of the motor separately. The aim of control is to regulate the motor speed. The speed command value is set by high level control. The algorithm is executed in two control loops. The fast inner control loop is executed with a 125 µs period. The slow outer control loop is executed with a period of one millisecond.

To achieve the goal of the PM synchronous motor control, the algorithm use feedback signals. The essential feedback signals are: three-phase stator current and the stator voltage. For the stator voltage the regulator output is used. For correct operation, the control structure presented requires a rotor position and speed information. In the case of the algorithms presented a sensorless technique using a sliding mode observer with an adaptive speed scheme is used.

The fast control loop executes two independent current control loops. They are the direct and quadrature-axis current ($i_{sd}$, and $i_{sq}$) PI controllers. The direct-axis current ($i_{sd}$) is used to control the rotor magnetizing flux. The quadrature-axis current ($i_{sq}$) corresponds to the motor torque. The current PI controllers' outputs are added to the corresponding d and q axis components of the decoupling stator voltage, obtaining the desired space-vector for the stator voltage applied to the motor. The fast control loop executes all the necessary tasks to be able to achieve an independent control of the stator current components. This includes:

- Forward Clark transformation
- Forward and backward park transformations
- Sliding mode observer and rotor position reconstruction
- DC-bus voltage ripple elimination
- Space vector modulation (SVM)

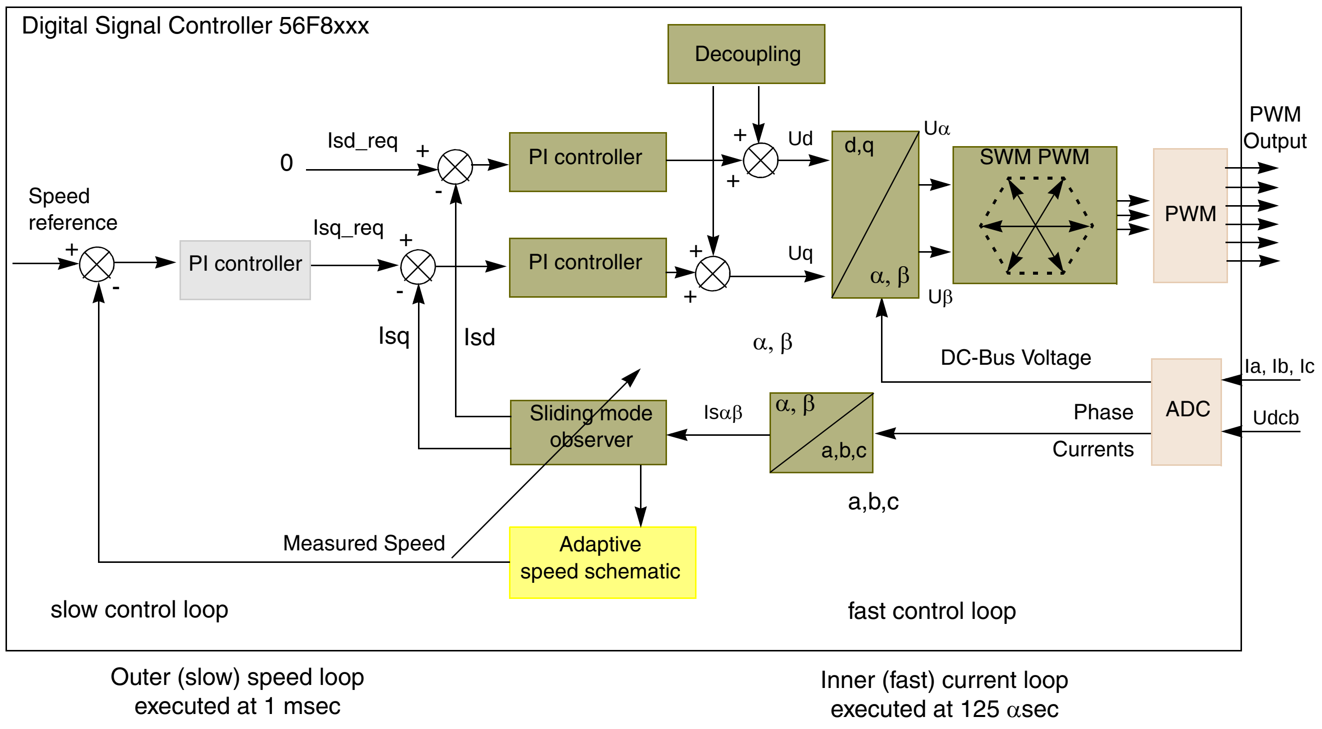


**Figure 2-5. PMSM Vector Control Algorithm Overview**

The slow control loop executes the speed controller and lower priority control tasks. The PI speed controller output sets a reference for the torque producing a quadrature axis component of the stator current ($i_{sq}$).

## 2.3.3 Stator Voltage Decoupling

For purposes of rotor magnetizing-flux oriented vector control, the direct-axis stator current $i_{sd}$ (rotor field component) and the quadrature-axis stator current $i_{sq}$ (torque-producing component) must be controlled independently. However, the equations of the stator voltage components are coupled. The direct axis component $u_{sd}$ depends on $i_{sq}$, and the quadrature axis component $u_{sq}$ depends on $i_{sd}$. The stator voltage components $u_{sd}$ and $u_{sq}$ cannot be considered as decoupled control variables for the rotor flux and electromagnetic torque. The stator currents $i_{sd}$ and $i_{sq}$ can only be independently controlled (decoupled control) if the stator voltage equations are decoupled, therefore the stator current components are indirectly controlled by controlling the terminal voltages of the synchronous motor.

The equations of the stator voltage components in the d, q reference frame can be reformulated and separated into two components: linear components $u_{sd}^{lin}, u_{sq}^{lin}$ and decoupling components $\mathrm{u}_{\mathrm{sd}}^{\mathrm{decouple}}, \mathrm{u}_{\mathrm{sq}}^{\mathrm{decouple}}$. The equations are decoupled as follows:

$$u_{sd} = u_{sd}^{lin} + u_{sd}^{decouple} \qquad \textbf{Eqn. 2-22}$$

$$u_{sq} = u_{sq}^{lin} + u_{sq}^{decouple} \qquad \textbf{Eqn. 2-23}$$

The decoupling components $u_{sd}^{decouple}, u_{sq}^{decouple}$ are evaluated from the stator voltage equations Equation 2-16 and Equation 2-17. They eliminate cross-coupling for current control loops at a given motor operating point. Linear components $u_{sd}^{lin}, u_{sq}^{lin}$ are set by the outputs of the current controllers. The voltage decoupling components are evaluated according to the following equations:

$$u_{sd}^{decouple} = R_s i_{sd} - p_p \omega (L_{s\sigma} + L_{r\sigma}) i_{sq} \qquad \textbf{Eqn. 2-24}$$

$$u_{sq}^{decouple} = R_s i_{sq} + p_p \omega (L_{s\sigma} + L_{r\sigma}) i_{sd} + p_p \omega L_m i_{mr} \qquad \textbf{Eqn. 2-25}$$

The above equations Equation 2-24 and Equation 2-25 are evaluated in the decoupling block. See Figure 2-5.

### 2.3.4 Space Vector Modulation

Space vector modulation (SVM) can directly transform the stator voltage vectors from the two-phase $\alpha,\beta$ coordinate system into pulse width modulation (PWM) signals duty cycle values.

The standard technique of output voltage generation uses an inverse Clarke transformation to obtain three-phase values. Using the phase voltage values, the duty cycles needed to control the power stage switches are then calculated. Although this technique gives good results, space vector modulation is more straightforward, valid only for transformation from the $\alpha,\beta$-coordinate system.

The basic principle of the standard space vector modulation technique can be explained with the help of the power stage schematic diagram depicted in Figure 2-6. Regarding the three-phase power stage configuration as shown in Figure 2-6, eight possible switching states (vectors) are feasible. They are given by combinations of the corresponding power switches. The hexagon shown in Figure 2-7 is a graphical representation of all the combinations. There are six non-zero vectors, $U_0$, $U_{60}$, $U_{120}$, $U_{180}$, $U_{240}$, $U_{300}$, and two zero vectors, $O_{000}$ and $O_{111}$, are defined in $\alpha,\beta$ coordinates.

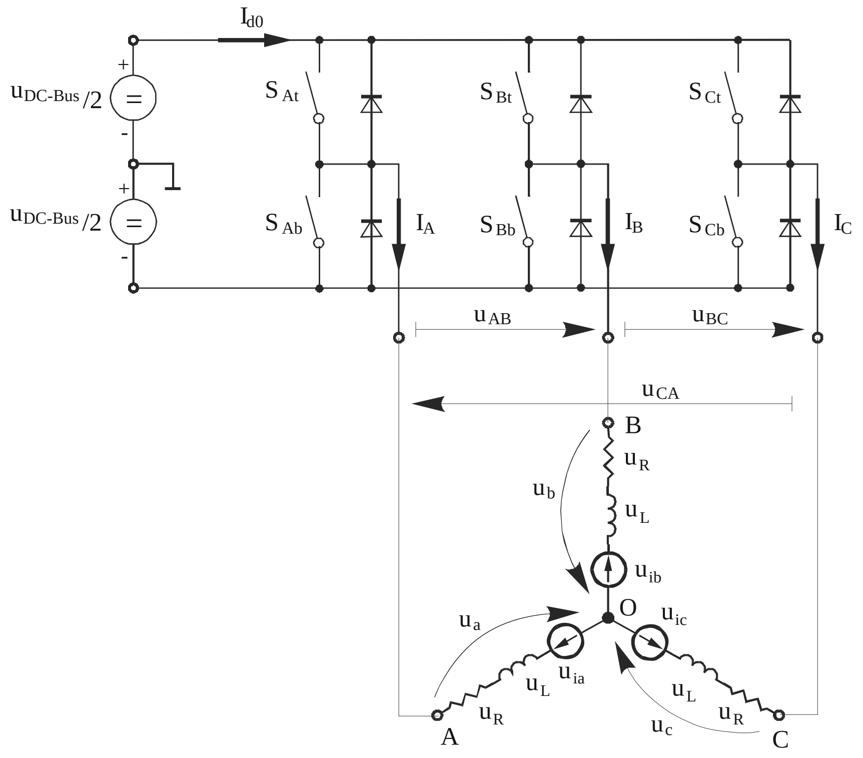


**Figure 2-6. Power Stage Schematic Diagram**

In Figure 2-7, the combination of on/off states for each voltage vector is coded by the three-digit number in parenthesis. Each digit represents one phase. For each phase, a value of one means that the upper switch is on and the bottom switch is off. A value of zero means that the upper switch is off and the bottom switch is on. These states together with the resulting instantaneous output line-to-line voltages, phase voltages, and voltage vectors are listed in Table 2-1.

**Table 2-1. Switching Patterns and Resulting Instantaneous**

| a | b | c | $U_a$ | $U_b$ | $U_c$ | $U_{AB}$ | $U_{BC}$ | $U_{CA}$ | Vector |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $0_{000}$ |
| 1 | 0 | 0 | $2U_{DC\text{-}Bus}/3$ | $-U_{DC\text{-}Bus}/3$ | $-U_{DC\text{-}Bus}/3$ | $U_{DC\text{-}Bus}$ | 0 | $-U_{DC\text{-}Bus}$ | $U_0$ |
| 1 | 1 | 0 | $U_{DC\text{-}Bus}/3$ | $U_{DC\text{-}Bus}/3$ | $-2U_{DC\text{-}Bus}/3$ | 0 | $U_{DC\text{-}Bus}$ | $-U_{DC\text{-}Bus}$ | $U_{60}$ |
| 0 | 1 | 0 | $-U_{DC\text{-}Bus}/3$ | $2U_{DC\text{-}Bus}/3$ | $-U_{DC\text{-}Bus}/3$ | $-U_{DC\text{-}Bus}$ | $U_{DC\text{-}Bus}$ | 0 | $U_{120}$ |
| 0 | 1 | 1 | $-2U_{DC\text{-}Bus}/3$ | $U_{DC\text{-}Bus}/3$ | $U_{DC\text{-}Bus}/3$ | $-U_{DC\text{-}Bus}$ | 0 | $U_{DC\text{-}Bus}$ | $U_{240}$ |
| 0 | 0 | 1 | $-U_{DC\text{-}Bus}/3$ | $-U_{DC\text{-}Bus}/3$ | $2U_{DC\text{-}Bus}/3$ | 0 | $-U_{DC\text{-}Bus}$ | $U_{DC\text{-}Bus}$ | $U_{300}$ |
| 1 | 0 | 1 | $U_{DC\text{-}Bus}/3$ | $-2U_{DC\text{-}Bus}/3$ | $U_{DC\text{-}Bus}/3$ | $U_{DC\text{-}Bus}$ | $-U_{DC\text{-}Bus}$ | 0 | $U_{360}$ |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | $0_{111}$ |

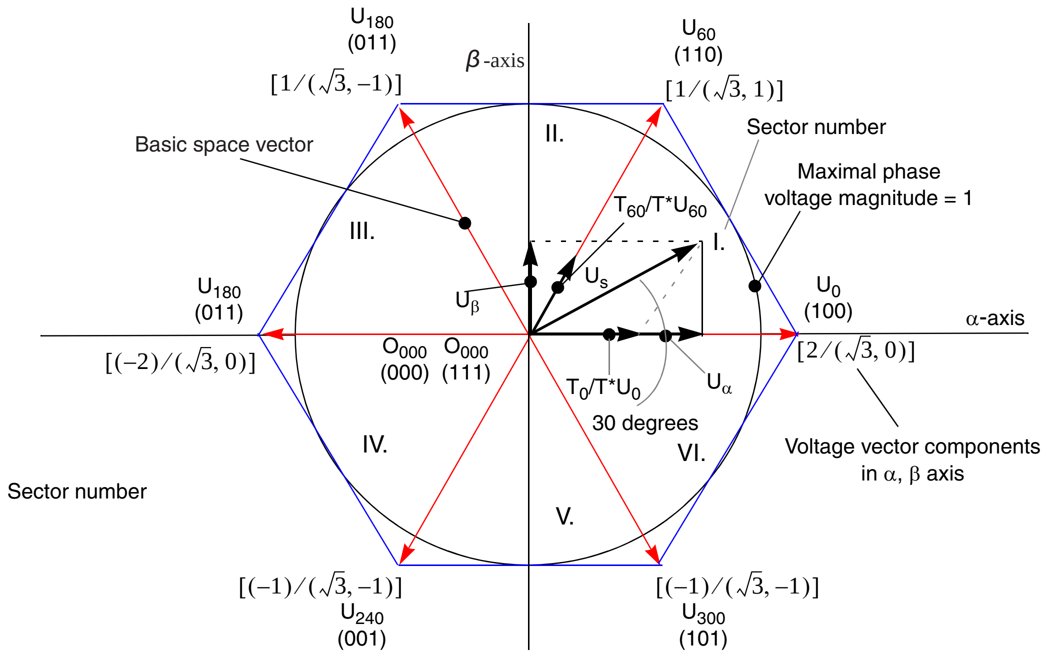


**Figure 2-7. Basic Space Vectors and Voltage Vector Projection**

SVM is a technique used as a direct bridge between vector control, voltage space vector, and PWM.

The SVM technique consists of several steps:

- Sector identification
- Space voltage vector decomposition into directions of sector base vectors $U_x$, $U_{x\pm60}$
- PWM duty cycle calculation

The principle of SVM is the application of the voltage vectors UXXX and OXXX and in certain instances the main vector of the PWM period TPWM is equal to the desired voltage vector.

This method gives the greatest variability in arranging the zero and non-zero vectors during the PWM period. Using the principle has the possibility to arrange these vectors as required. In some applications to lower switching losses or gets different results, for example centre-aligned PWM, edge-aligned PWM, or minimal switching.

For the chosen SVM the following rule is defined:

- The desired space voltage vector is created only by applying the sector base vectors, the non-zero vectors on the sector side ($U_x$, $U_{x\pm60}$), and the zero vectors ($O_{000}$ or $O_{111}$).

The following equations define the principle of the SVM:

$$T_{PWM} \cdot U_{S[\alpha,\beta]} = T_1 \cdot U_X + T_2 \cdot U_{X\pm60} + T_0 \cdot (O_{000} \vee O_{111}) \qquad \text{Eqn. 2-26}$$

$$T_{PWM} = T_1 + T_2 + T_0 \qquad \text{Eqn. 2-27}$$

To solve the time periods $T_0$, $T_1$ and $T_2$, it is necessary to decompose the space voltage vector $U_{S[\alpha,\beta]}$ into directions of the sector base vectors $U_x$, and $U_{x\pm60}$. The Equation 2-26 is split into equations Equation 2-28 and Equation 2-29.

$$T_{PWM} \cdot U_{SX} = T_1 \cdot U_X \qquad \textbf{Eqn. 2-28}$$

$$T_{PWM} \cdot U_{S(X\pm60)} = T_2 \cdot U_{X\pm60} \qquad \textbf{Eqn. 2-29}$$

By solving these set of equations, the necessary duration can be calculated for the application of the sector base vectors $U_x$, and $U_{x\pm60}$ during the PWM period $T_{PWM}$ to produce the correct stator voltages.

$$T_1 = \frac{|U_{SX}|}{|U_X|}T_{PWM} \quad \text{for vector } U_x \qquad \textbf{Eqn. 2-30}$$

$$T_2 = \frac{|U_{SX}|}{|U_{X\pm60}|}T_{PWM} \quad \text{for vector } U_{x\pm60} \qquad \textbf{Eqn. 2-31}$$

$$T_0 = T_{PWM} - (T_1 + T_2) \quad \text{either for } O_{000} \text{ or } O_{111} \qquad \textbf{Eqn. 2-32}$$

## 2.4 Sensorless Control of PM Synchronous Motor

### 2.4.1 Sensorless PM Synchronous Motor Technique Classification

As described in previous sections, the rotor position angle $\theta r$ and angular speed $\omega$ need to be measured or estimated to implement the speed vector control. There are a few techniques for the estimation.

According to the motor model and the angular speed range the method can be split into:

- Section 2.4.1.1, “Estimation Based on Motor Inductance Saliency”
- Section 2.4.1.2, “Estimation Based on BEMF Induced Voltage or Flux Reconstruction”

#### 2.4.1.1 Estimation Based on Motor Inductance Saliency

The sensorless estimation techniques of this group are based on a measurement of motor inductance saliency. The motor inductance saliency is a rotor position angle function. Some motors have a strong inductance saliency effect. These techniques are able to estimate the rotor position from a motor stop and a speed near to zero. At the medium to high speed range the saliency measurement is usually impossible due to BEMF.

#### 2.4.1.2 Estimation Based on BEMF Induced Voltage or Flux Reconstruction

The sensorless estimation techniques of this group is based on estimation of the motor BEMF induced voltage or electromagnetic flux. The rotor position and speed is estimated from reconstructed BEMF voltage or electromagnetic flux quantities.

At zero speed, the BEMF induced voltage is equal to zero. The BEMF becomes significant from a certain rotor angular speed. Therefore, sensorless estimation techniques perform with a perfect precision in the medium to high speed range. Usually it can be used from 5% of nominal speed or lower.

The sensorless estimation techniques that use any kind of feedback observer with a system model are the most popular. These techniques are based on a deterministic system model. Usually motor identification is provided before setting the controller parameters and the controller uses a system model that estimates the motor BEMF or flux with good enough precision.

All the following sections concentrate on the BEMF feedback observer with a system model.

#### 2.4.1.3 BEMF or Flux Observer with a System Model and $\alpha, \beta$ Versus d, q Coordinates

The motor model is usually based on an invariant stator coil inductance (insignificant variance). Many PM synchronous motors have an inductance variance negligible over rotor angle especially if compared to the BEMF influence. Then the observer calculated in α,β coordinates can be used.

Some PM synchronous motors have significant inductance saliency that prevents the use of the α, β model. In this case most of the PM synchronous motors have an inductance almost constant in the rotor related d, q coordinates (lateral induction $L_q$ is constant but different from $L_d$, which is also constant). Then the rotor related d, q coordinate model is used.

This manual therefore describes two possible observers and SW versions:

- Section 2.4.2.2, "Sliding Mode Observer with a,b Coordinate Model"
- Section 2.4.2.8, "Sliding Mode Observer with the d, q Coordinates Model"

The Section 2.4.2.11, "SMO in d, q versus a,b Coordinates Suitability for a Motor" advises which SW version is suitable for a specific motor.

### 2.4.2 Sliding Mode BEMF Observer with Adaptive Velocity Estimation

The purpose of the sliding mode observer (SMO) is shown in Figure 2-8. The position of the rotor with its permanent magnet flux is not certain. It can be estimated from the BEMF:

$$\hat{\theta} = tan^{-1}\left(-\frac{\hat{e}_{S\alpha}}{\hat{e}_{S\beta}}\right) \qquad \text{Eqn. 2-33}$$

The BEMF can not be directly measured. An estimation from an observer is needed. The feedback observer estimates the BEMF using a system model with voltage and current vector inputs.

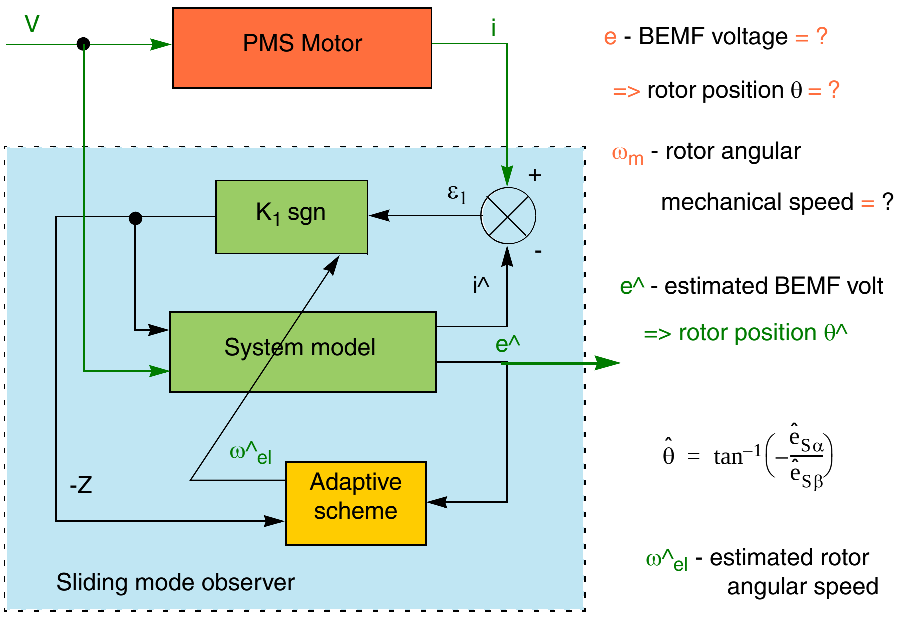


**Figure 2-8. Sliding Mode Observer with Adaptive Velocity Estimation**

The estimated BEMF $\hat{e}$ are system variables in the SMO. These system variables are used for position sine and cosine evaluation according to equations Equation 2-63, and Equation 2-64. Sliding mode observer is a type of feedback observer. Feedback is provided through the sgn function. The SMO is suitable for the PM synchronous motor due to the BEMF model.

The version of the SMO used in this application uses a model where the coefficients are calculated according to angular speed $\hat{\omega}_{el}$. The adaptive scheme for velocity estimation is therefore necessary.

### 2.4.2.1 Continuous Time System Model $\alpha,\beta$ Coordinates

Assumptions:

1. As mentioned earlier, the α, β model assumes invariant inductance L(θ) = const., L(i) = const.
2. Angular speed ω is assumed as constant because the electrical system constants are much lower compared to mechanical acceleration of the rotor.
3. The motor BEMF is sinusoidal. This more or less applies for the PM synchronous motors. Due to assumption 2, the BEMF amplitude is also constant.

From Equation 2-9, Equation 2-10, Equation 2-11, and Equation 2-12, the following equation can be derived for a continuous time motor model:

$$\frac{d}{dt}\begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \\ e_{s\alpha} \\ e_{s\beta} \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} \\ 0 & 0 & 0 & -\omega_{el} \\ 0 & 0 & \omega_{el} & 0 \end{bmatrix} \bullet \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \\ e_{s\alpha} \\ e_{s\beta} \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} u_{s\alpha} \\ u_{s\beta} \end{bmatrix}$$

**Eqn. 2-34**

Equation 2-34 can be used for most motors where the inductance salience variance is insignificant especially compared to the BEMF effect on the system equation in the operating speed range.

Equation 2-34 can be written using submatrix:

$$\frac{d}{dt}\begin{bmatrix} \hat{i}_{s(\alpha,\beta)} \\ \hat{e}_{s(\alpha,\beta)} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \bullet \begin{bmatrix} \hat{i}_{s(\alpha,\beta)} \\ \hat{e}_{s(\alpha,\beta)} \end{bmatrix} + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} \bullet \begin{bmatrix} u_{s(\alpha,\beta)} \end{bmatrix}$$

**Eqn. 2-35**

Where:

$$A_{11} = a_{11} \cdot I = -\frac{R_S}{L_S} \cdot I$$

**Eqn. 2-36**

$$A_{12} = a_{12} \cdot I = -\frac{1}{L_S} \cdot I$$

**Eqn. 2-37**

$$A_{22} = a_{22} \cdot J = \omega_{el} \cdot J$$

**Eqn. 2-38**

$$B_1 = b_1 \cdot I = -a_{12} \cdot I = \frac{1}{L_S} \cdot I$$

**Eqn. 2-39**

### 2.4.2.2 Sliding Mode Observer with $\alpha,\beta$ Coordinate Model

The sliding mode observer uses the system model with the sgn feedback function. The continuous time version of the SMO is described by Equation 2-40. This is the equation calculated by the controller to get the estimated values $\hat{\mathbf{e}}_{S(\alpha,\beta)}$ for the rotor position.

$$\frac{d}{dt}\begin{bmatrix}\hat{i}_{S(\alpha,\beta)} \\ \hat{e}_{S(\alpha,\beta)}\end{bmatrix}(t) = \begin{bmatrix}A_{11} & A_{12} \\ 0 & A_{22}\end{bmatrix} \bullet \begin{bmatrix}\hat{i}_{S(\alpha,\beta)} \\ \hat{e}_{S(\alpha,\beta)}\end{bmatrix}(t) + \begin{bmatrix}B_1 \\ 0\end{bmatrix} \bullet \begin{bmatrix}\hat{u}_{S(\alpha,\beta)}\end{bmatrix}(t) + K_1 \bullet \begin{bmatrix}I \\ G\end{bmatrix} \bullet SGN(\hat{i}_{S(\alpha,\beta)}(t) - i_{S(\alpha,\beta)}(t))$$

**Eqn. 2-40**

**NOTE**

The submatrix system from Equation 2-40 incorporates parameter A11 rotor angular speed $\hat{\omega}_{el}$ . An adaptive speed scheme described in 2.4.2.6, "Adaptive Speed Scheme" is used for convergence of the estimated speed. The whole estimator is the SMO from Equation 2-40 together with the adaptive speed scheme Equation 2-68.

Where:

$$I = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}$$

**Eqn. 2-41**

$$J = \begin{bmatrix}0 & -1 \\ 1 & 0\end{bmatrix}$$

**Eqn. 2-42**

Sgn vector function:

$$sgn\left(\begin{bmatrix}i_\alpha \\ i_\beta\end{bmatrix}\right) = \begin{bmatrix}sgn_\alpha = 1 \Leftrightarrow i_\alpha > 0, sgn_\alpha = 0 \Leftrightarrow i_\alpha = 0, sgn_\alpha = -1 \Leftrightarrow i_\alpha < 0 \\ sgn_\beta = 1 \Leftrightarrow i_\beta > 0, sgn_\beta = 0 \Leftrightarrow i_\beta = 0, sgn_\beta = -1 \Leftrightarrow i_\beta < 0\end{bmatrix}$$

**Eqn. 2-43**

Switching Gain:

$$K_1 = k_1 \cdot I$$

**Eqn. 2-44**

Feedback Gain:

$$G = g_1 \cdot I + g_2 \cdot J$$

**Eqn. 2-45**

The system model coefficients are in Equation 2-36 to Equation 2-39.

The time discrete version of Equation 2-40 (necessary for digital control) is Equation 2-56.

The functionality and the error convergence is described in the following section.

### 2.4.2.3 Functionality and the Estimation Error of the SMO with the a,b Coordinate Model

Defining of the error vectors.

Current error vector:

$$\varepsilon_{1(\alpha,\beta)} = \begin{bmatrix} \hat{i}_{S\alpha} \\ \hat{i}_{S\beta} \end{bmatrix} - \begin{bmatrix} i_{S\alpha} \\ i_{S\beta} \end{bmatrix} \qquad \textbf{Eqn. 2-46}$$

BEMF error vector:

$$\varepsilon_{2(\alpha,\beta)} = \begin{bmatrix} \hat{e}_{S\alpha} \\ \hat{e}_{S\beta} \end{bmatrix} - \begin{bmatrix} e_{S\alpha} \\ e_{S\beta} \end{bmatrix} \qquad \textbf{Eqn. 2-47}$$

The sliding mode hyperlane is selected as:

$$\varepsilon_{1(\alpha,\beta)} = \lfloor i_{S(\alpha,\beta)} - \hat{i}_{S(\alpha,\beta)} \rfloor = 0 \qquad \textbf{Eqn. 2-48}$$

The sliding mode occurs when the sliding mode condition is met:

$$\varepsilon_{1(\alpha,\beta)}T \bullet \frac{d}{dt}(\varepsilon_{1(\alpha,\beta)}) < 0 \qquad \textbf{Eqn. 2-49}$$

If $k_1$ has a high enough negative value, Equation 2-49 is satisfied and the sliding mode occurs.

$$[i_{S(\alpha,\beta)} - \hat{i}_{S(\alpha,\beta)}] = \varepsilon_{1(\alpha,\beta)} = \frac{d}{dt}(\varepsilon_{1(\alpha,\beta)}) = 0 \qquad \textbf{Eqn. 2-50}$$

From Equation 2-40, Equation 2-46, and Equation 2-47:

$$\frac{d}{dt}[\varepsilon_{1(\alpha,\beta)}] = \begin{bmatrix} A_{11} & A_{12} \end{bmatrix} \bullet \begin{bmatrix} \varepsilon_{1(\alpha,\beta)} \\ \varepsilon_{2(\alpha,\beta)} \end{bmatrix} + k_1 \cdot I \bullet sgn(\varepsilon_{1(\alpha,\beta)}) \qquad \textbf{Eqn. 2-51}$$

Defining the switching signal vector function $Z_{(\alpha,\beta)}$:

$$-[Z_{(\alpha,\beta)}] = k_1 \cdot I \bullet sgn(\hat{i}_{(\alpha,\beta)} - i_{(\alpha,\beta)}) \qquad \textbf{Eqn. 2-52}$$

$Z_{(\alpha,\beta)}$ signal compensates the BEMF error $\varepsilon_2$, from Equation 2-50 and Equation 2-51:

$$\lfloor Z_{(\alpha,\beta)} \rfloor = A_{12} \bullet \varepsilon_{2(\alpha,\beta)} \qquad \textbf{Eqn. 2-53}$$

From the SMO Equation 2-40:

$$\frac{d}{dt}\hat{e}_{S(\alpha,\beta)} = A_{22} \bullet \hat{e}_{S(\alpha,\beta)} + K_1 \bullet G\,sgn(\varepsilon_{1(\alpha,\beta)}) \qquad \textbf{Eqn. 2-54}$$

$Z_{(\alpha,\beta)}$ is used as a feedback. The BEMF error function in the SMO is:

$$\frac{d}{dt}(\varepsilon_{2(\alpha,\beta)}) = (A_{22} - G \bullet A_{12}) \bullet \varepsilon_{2(\alpha,\beta)} \qquad \textbf{Eqn. 2-55}$$

### 2.4.2.4 Time Discrete Sliding Mode Observer with the $\alpha,\beta$ Coordinate Model

In a digital control application, a time discrete equation of the SMO is needed. The Euler method is the appropriate way to transform to a time discrete observer. This is due to fast calculation and implementation of the sgn function by the SMO. The iteration precision of the Euler method is sufficient enough and the time discrete SMO is:

$$\begin{bmatrix}\hat{i}_{S(\alpha,\beta)}\\ \hat{e}_{S(\alpha,\beta)}\end{bmatrix}(k+1) = \left(I + \begin{bmatrix}A_{11} & A_{12}\\ 0 & A_{22}\end{bmatrix}\cdot T_S\right)\bullet\begin{bmatrix}\hat{i}_{S(\alpha,\beta)}\\ \hat{e}_{S(\alpha,\beta)}\end{bmatrix}(k) + \begin{bmatrix}\mathbf{B}_1\\ 0\end{bmatrix}\cdot T_S\bullet[u_{S(\alpha,\beta)}](k) + K1\bullet\begin{bmatrix}I\\ G\end{bmatrix}\cdot T_S\bullet sgn(\hat{i}_{S(\alpha,\beta)}(k) - i_{S(\alpha,\beta)}(k))$$

*Eqn. 2-56*

**NOTE**

The time discrete model in Equation 2-56 is the SMO observation calculated by the DSC to get the estimated BEMF $\hat{\mathbf{e}}_{S(\alpha,\beta)}$ . This equation together with the adaptive speed scheme using Equation 2-69 $\hat{\omega}_{el}$ gives the angular speed and rotor position angle sine and cosine estimation using Equation 2-63, and Equation 2-64.

Time discrete system matrix α, β coordinates:

$$\begin{bmatrix}A_{11} & A_{12}\\ 0 & A_{22}\end{bmatrix}\cdot T_S = \begin{bmatrix}-\frac{R_S}{L_S}T_S & 0 & -\frac{1}{L_S}T_S & 0\\ 0 & -\frac{R_S}{L_S}T_S & 0 & -\frac{1}{L_S}T_S\\ 0 & 0 & 0 & -\omega_{el}T_S\\ 0 & 0 & \omega_{el}T_S & 0\end{bmatrix}$$

*Eqn. 2-57*

$$B_1\cdot T_S = \begin{bmatrix}\frac{1}{L_S}T_S & 0\\ 0 & \frac{1}{L_S}T_S\end{bmatrix}$$

*Eqn. 2-58*

Then:

$$A_{11}T_S = a_{11}T_S \cdot I = -\frac{R_S}{L_S}T_S \cdot I \qquad \text{Eqn. 2-59}$$

$$A_{12}T_S = a_{12}T_S \cdot I = -\frac{1}{L_S}T_S \cdot I \qquad \text{Eqn. 2-60}$$

$$A_{22}T_S = \omega_{el}T_S \cdot J \qquad \text{Eqn. 2-61}$$

$$B_1T_S = b_1T_S \cdot I = -a_{11}T_S \cdot I = \frac{1}{L_S}T_S \cdot I \qquad \text{Eqn. 2-62}$$

The unit I and imaginary matrix J are according to Equation 2-40 and Equation 2-41.

### 2.4.2.5 Rotor Position

Evaluation of the rotor position sine and cosine is necessary for the vector control transformations:

$$cos(\theta) = \frac{e_{S\alpha}}{\sqrt{(e_{S\beta}^2 + e_{S\alpha}^2)}} \cdot sgn(\omega_{el}) \qquad \text{Eqn. 2-63}$$

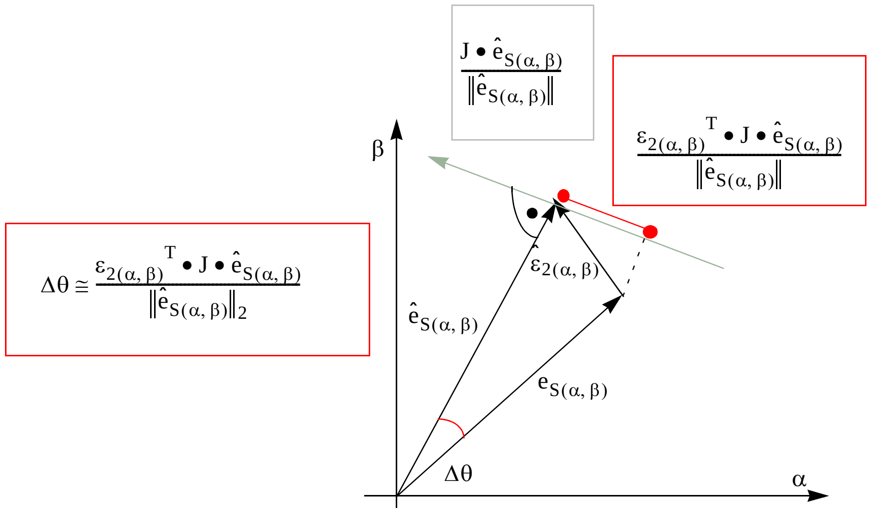
$$sin(\theta) = \frac{-e_{S\beta}}{\sqrt{(e_{S\beta}^2 + e_{S\alpha}^2)}} \cdot sgn(\omega_{el}) \qquad \text{Eqn. 2-64}$$

The sine and cosine of the rotor position is sufficient information for the vector control calculations. See Section 2.3.4, “Space Vector Modulation”. The angle itself does not need to be quantified, but it may be needed for testing purposes.

### 2.4.2.6 Adaptive Speed Scheme

The sliding mode observer described in this document uses a model of the fourth order system matrix. The angular speed is used for calculation of the submatrix $A_{22}$ coefficients. Therefore, it needs an adaptive angular speed estimation included into the feedback system.

Graphical representation of the angle error feedback used for the adaptive speed scheme is in Figure 2-9.



$\frac{J \bullet \hat{e}_{S(\alpha,\beta)}}{\|\hat{e}_{S(\alpha,\beta)}\|}$
$\frac{\varepsilon_{2(\alpha,\beta)}^T \bullet J \bullet \hat{e}_{S(\alpha,\beta)}}{\|\hat{e}_{S(\alpha,\beta)}\|}$
$\Delta\theta \cong \frac{\varepsilon_{2(\alpha,\beta)}^T \bullet J \bullet \hat{e}_{S(\alpha,\beta)}}{\|\hat{e}_{S(\alpha,\beta)}\|_2}$
β
$\varepsilon_{2(\alpha,\beta)}$
$\hat{e}_{S(\alpha,\beta)}$
$e_{S(\alpha,\beta)}$
Δθ
α

Figure 2-9. Adaptive Speed Scheme

The following relationship between the BEMF vector error e2, estimated BEMF vector é and SMO switching function z can be deduced from the SMO equations:

$$Z_{(\alpha,\beta)}^T \bullet J \bullet \hat{e}_{S(\alpha,\beta)} = (A_{12} \bullet \varepsilon_{2(\alpha,\beta)})^T \bullet J \bullet \hat{e}_{S(\alpha,\beta)} = a_{12}\|\hat{e}_{S(\alpha,\beta)}\|_2 \frac{\varepsilon_{2(\alpha,\beta)}^T \bullet J \bullet \hat{e}_{S(\alpha,\beta)}}{\|\hat{e}_{S(\alpha,\beta)}\|_2} \qquad \textbf{Eqn. 2-65}$$

Where

$$\|\hat{e}_{(\alpha,\beta)}\|_2 = \hat{e}_{S\alpha}^2 + \hat{e}_{S\alpha}^2 \qquad \textbf{Eqn. 2-66}$$

As can be seen in Figure 2-9. There is a dependence between the angle error and the BEMF vector error:

$$\Delta\theta \cong \frac{\varepsilon_{2(\alpha,\beta)}^T \bullet J \bullet \hat{e}_{S(\alpha,\beta)}}{\|\hat{e}_{(\alpha,\beta)}\|_2} \qquad \textbf{Eqn. 2-67}$$

The scalar result also gives the angle error signature and can be used for the adaptive speed convergency.

The following equation can be used as adaptive speed scheme:

$$\frac{d}{dt}\hat{\omega}_{el} = g_{\omega} \bullet (Z_{(\alpha,\beta)}{}^{T} \bullet J \bullet \hat{e}_{S(\alpha,\beta)}) \qquad \textbf{Eqn. 2-68}$$

The estimated speed $\hat{\omega}_{el}$ for Equation 2-56 is finally iterated using the time discrete form of the adaptive scheme Equation 2-68:

$$\hat{\omega}_{el}(k+1) = g_{\omega} \bullet (Z_{(\alpha,\beta)}(k)^{T} \bullet J \bullet \hat{e}_{S(\alpha,\beta)}(k+1))T_S + \hat{\omega}_{el}(k) \qquad \textbf{Eqn. 2-69}$$

The stability of an SMO with an adaptive scheme is described in reference 11. See Section 1.3.2, “Bibliography”.

### 2.4.2.7 Continuous Time System Model d, q Coordinates

Assumptions:

1. As mentioned earlier the d, q model assumes unequal inductances $Ld \neq Lq$ and invariant inductance, Ld = const, Lq = const., and L(i) = const.

**NOTE**

This is the main difference from the $\alpha,\beta$ observer. Many PM synchronous motors have a stator inductance saliency variant over the rotor angle, due to magnetic gaps according to the PM rotor position. Related to the rotor coordinates, most of the motors can be described with a constant but unequal Ld and Lq inductions.

2. Angular speed ω is assumed as constant. This is because the electrical system constants are much lower compared to the mechanical acceleration of the rotor.
3. The motor BEMF is sinusoidal. This more or less applies for PM synchronous motors. Then, due to the assumption above the BEMF amplitude is also constant.

From Equation 2-16, Equation 2-17, Equation 2-18, and Equation 2-19, the following formula can be derived for a continuous time motor model:

$$\frac{d}{dt}\begin{bmatrix} i_{Sd} \\ i_{Sq} \\ e_{Sd} \\ e_{Sq} \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_{sd}} & \omega_{el}\frac{L_{sq}}{L_{sd}} & -\frac{1}{L_{sd}} & 0 \\ \omega_{el}\frac{L_{sd}}{L_{sq}} & -\frac{R_s}{L_{sq}} & 0 & -\frac{1}{L_{sq}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} i_{Sd} \\ i_{Sq} \\ e_{Sd} \\ e_{Sq} \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{sd}} & 0 \\ 0 & \frac{1}{L_{sq}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} u_{Sd} \\ u_{Sq} \end{bmatrix} \qquad \textbf{Eqn. 2-70}$$

Equation 2-70 can be used for most motors where the inductance salience variance is significant. Rotor related Ld, and Lq inductances are almost constant, particularly if compared to the BEMF effect on the system equation in the operating speed range. According to the assumptions 2 and 3 the BEMF amplitude is expected as a constant. In this case, a very slow change compared to the current and voltage dynamics, therefore all the coefficients are constant. The BEMF vector is not rotated on the d, q coordinates because it is fixed to the rotor angle.

Equation 2-70 can also be written as:

$$\frac{d}{dt}\begin{bmatrix} i_{Sd} \\ i_{Sq} \\ e_{Sd} \\ e_{Sq} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & 0 & a_{24} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} i_{Sd} \\ i_{Sq} \\ e_{Sd} \\ e_{Sq} \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ 0 & b_{22} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} u_{Sd} \\ u_{Sq} \end{bmatrix} \qquad \textbf{Eqn. 2-71}$$

Where:

$$a_{11} = -\frac{R_s}{L_{sd}} \qquad \textbf{Eqn. 2-72}$$

$$a_{22} = -\frac{R_s}{L_{sq}} \qquad \textbf{Eqn. 2-73}$$

$$b_{11} = a_{13} = \frac{1}{L_{sd}} \qquad \textbf{Eqn. 2-74}$$

$$b_{11}T_s = -a_{13}t_s = \frac{1}{L_{sd}}T_s \qquad \textbf{Eqn. 2-75}$$

$$b_{22} = -a_{24} = \frac{1}{L_{sq}} \qquad \textbf{Eqn. 2-76}$$

$$a_{12} = \omega_{el}\frac{L_{sq}}{L_{sd}} \qquad \textbf{Eqn. 2-77}$$

$$a_{21} = -\omega_{el}\frac{L_{sd}}{L_{sq}} \qquad \textbf{Eqn. 2-78}$$

### 2.4.2.8 Sliding Mode Observer with the d, q Coordinates Model

The sliding mode observer in d, q coordinates is similar to the SMO with α,β coordinates.

$$\frac{d}{dt}\begin{bmatrix}\hat{i}_{S(d,q)}\\ \hat{e}_{S(d,q)}\end{bmatrix}(t) = \begin{bmatrix}a_{11} & a_{12} & a_{13} & 0\\ a_{21} & a_{22} & 0 & a_{24}\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix} \bullet \begin{bmatrix}\hat{i}_{S(d,q)}\\ \hat{e}_{S(d,q)}\end{bmatrix}(t) + \begin{bmatrix}b_{11} & 0\\ 0 & b_{22}\\ 0 & 0\\ 0 & 0\end{bmatrix} \bullet [u_{S(d,q)}](t) + K_1 \bullet \begin{bmatrix}I\\ G\end{bmatrix} \bullet sgn(\hat{i}_{S(d,q)}(t) - i_{S(d,q)}(t))$$

**Eqn. 2-79**

Where:

The unit matrix and gain coefficients are described in Equation 2-41 to Equation 2-45. The system coefficients are according to Equation 2-72 to Equation 2-78.

**NOTE**

The submatrix system from Equation 2-79 incorporates parameter a12, and a21 with rotor angular speed $\hat{\omega}_{el}$ . The adaptive speed scheme, described in Section 2.4.2.10, “Adaptive Speed Scheme for the d, q Observer” is used for convergence of the estimated speed. Therefore, the whole estimator is the SMO from Equation 2-71 together with the adaptive speed scheme Equation 2-96.

Equation 2-79 can also be written as:

$$\frac{d}{dt}\begin{bmatrix}\hat{i}_{S(d,q)}\\ \hat{e}_{S(d,q)}\end{bmatrix}(t) = \begin{bmatrix}A_{11} & A_{12}\\ 0 & 0\end{bmatrix} \bullet \begin{bmatrix}\hat{i}_{S(d,q)}\\ \hat{e}_{S(d,q)}\end{bmatrix}(t) + \begin{bmatrix}B_1\\ 0\end{bmatrix} \bullet [u_{S(d,q)}](t) + K_1 \bullet \begin{bmatrix}I\\ G\end{bmatrix} \bullet sgn(\hat{i}_{S(d,q)}(t) - i_{S(d,q)}(t))$$

**Eqn. 2-80**

Where:

$$A_{11} = \begin{bmatrix}-\frac{R_s}{L_{sd}} & \omega_{el}\frac{L_{sq}}{L_{sd}}\\ -\omega_{el}\frac{L_{sd}}{L_{sq}} & -\frac{R_s}{L_{sq}}\end{bmatrix}$$

**Eqn. 2-81**

$$A_{12} = \begin{bmatrix}-\frac{1}{L_{sd}} & 0\\ 0 & -\frac{1}{L_{sq}}\end{bmatrix}$$

**Eqn. 2-82**

$$B_1 = \begin{bmatrix} \frac{1}{L_{sd}} & 0 \\ 0 & \frac{1}{L_{sq}} \end{bmatrix}$$ **Eqn. 2-83**

The Sliding mode observer functionality, switching function, and error are similar to that described in Section 2.4.2.2, “Sliding Mode Observer with a,b Coordinate Model”. The main difference is that all the variables must be transformed and calculated in the d, q coordinate system. The functionality and other differences are described in Section 2.4.2.9, “Time Discrete Sliding Mode Observer with the d, q Coordinate Model”.

### 2.4.2.9 Time Discrete Sliding Mode Observer with the d, q Coordinate Model

The time discrete equation of the SMO in d, q coordinates are:

$$\begin{bmatrix} \hat{\mathbf{i}}_{S(d,q)} \\ \hat{\mathbf{e}}_{S(d,q)} \end{bmatrix}(k+1) = \left( \mathbf{I} + \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & 0 & a_{24} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot T_s \right)_s \bullet \begin{bmatrix} \hat{\mathbf{i}}_{S(d,q)} \\ \hat{\mathbf{e}}_{S(d,q)} \end{bmatrix}(k) + \begin{bmatrix} b_{11} & 0 \\ 0 & b_{22} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot T_s \bullet [\mathbf{u}_{S(d,q)}](k) + \mathbf{K_1} \bullet \begin{bmatrix} I \\ G \end{bmatrix} \cdot T_s \bullet \mathbf{sgn}(\hat{\mathbf{i}}_{S(d,q)}(k) - \mathbf{i}_{S(d,q)}(k))$$

**Eqn. 2-84**

This is the equation calculated by the controller to get the estimated values $\hat{\mathbf{e}}_{S(d,q)}$ for the rotor position.

Where:

$$a_{11} t_s = -\frac{R_s}{L_{sd}} T_s$$ **Eqn. 2-85**

$$a_{22} T_s = -\frac{R_s}{L_{sq}} T_s$$ **Eqn. 2-86**

$$b_{11} T_s = -a_{13} T_s = \frac{1}{L_{sd}} T_s$$ **Eqn. 2-87**

$$b_{22} = -a_{24} T_s = \frac{1}{L_{sq}} T_s$$ **Eqn. 2-88**

$$a_{12} T_s = \omega_{el} \frac{L_{sq}}{L_{sd}} T_s$$ **Eqn. 2-89**

$$a_{21} T_s = -\omega_{el} \frac{L_{sd}}{L_{sq}} T_s$$ **Eqn. 2-90**

**NOTE**

Time discrete model in the Equation 2-84 is the SMO observer calculated finally by the DSC to get the estimated BEMF $\hat{\mathbf{e}}_{S(\alpha,\beta)}$. This equation together with adaptive speed scheme for $\hat{\omega}_{el}$ gives the angular speed and rotor position angle sine and cosine estimation (using Equation 2-91 to Equation 2-93).

The main task of the controller using the SMO calculating in d, q coordinates is to correctly provide the transformation of the dedicated quantities from the stator related α,β system into the rotor related d, q coordinates. One iteration sampling step of this observer is shown in Figure 2-10. The current is transformed to rotor related d, q coordinates. The controller calculates the SMO according to Equation 2-84. The result of the calculation is corrected BEMF $e_{Scor}$ (d component might be unequal to 0!). The controller converts the corrected BEMF into α,β coordinates, then provides a rotation $e^{J\omega T}$ according to the estimated speed. These steps are necessary to calculate the new step values of the rotor angle sine(k+1) and cosine(k+1). The angle sine and cosine values are calculated according to Equation 2-91 to Equation 2-93.

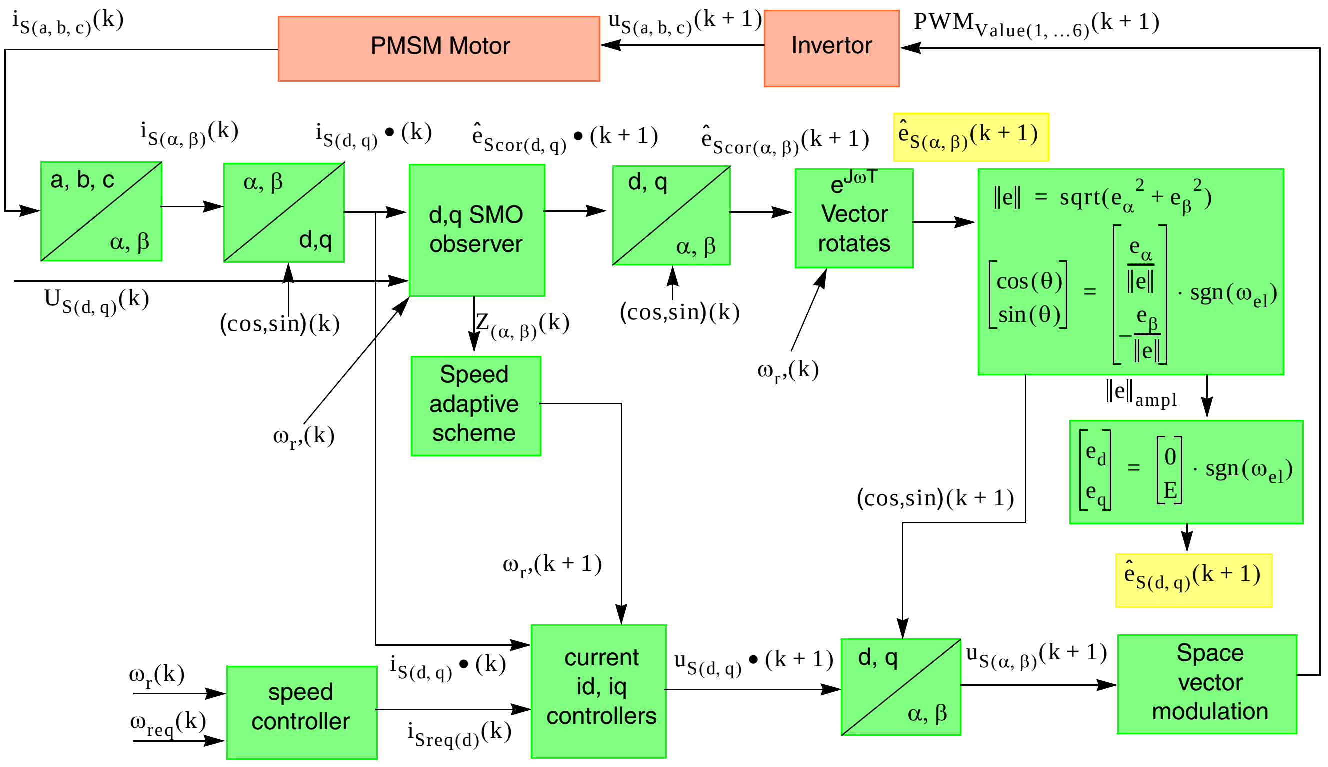

$i_{S(a,b,c)}(k)$
PMSM Motor
$u_{S(a,b,c)}(k+1)$
Invertor
$PWM_{Value(1,\ldots,6)}(k+1)$
$i_{S(\alpha,\beta)}(k)$
$i_{S(d,q)} \bullet (k)$
$\hat{e}_{Scor(d,q)} \bullet (k+1)$
$\hat{e}_{Scor(\alpha,\beta)}(k+1)$
$\hat{e}_{S(\alpha,\beta)}(k+1)$
a, b, c / α, β
α, β / d,q
d,q SMO observer
d, q / α, β
$e^{J\omega T}$ Vector rotates
$\|e\| = \mathrm{sqrt}(e_\alpha^2 + e_\beta^2)$
$\begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix} = \begin{bmatrix}\frac{e_\alpha}{\|e\|}\\ -\frac{e_\beta}{\|e\|}\end{bmatrix} \cdot \mathrm{sgn}(\omega_{el})$
$U_{S(d,q)}(k)$
(cos,sin)(k)
$Z_{(\alpha,\beta)}(k)$
(cos,sin)(k)
$\omega_{r'}(k)$
Speed adaptive scheme
$\omega_{r'}(k)$
$\|e\|_{ampl}$
$\begin{bmatrix}e_d\\ e_q\end{bmatrix} = \begin{bmatrix}0\\ E\end{bmatrix} \cdot \mathrm{sgn}(\omega_{el})$
(cos,sin)(k + 1)
$\hat{e}_{S(d,q)}(k+1)$
$\omega_{r'}(k+1)$
$\omega_r(k)$
$\omega_{req}(k)$
speed controller
$i_{S(d,q)} \bullet (k)$
$i_{Sreq(d)}(k)$
current id, iq controllers
$u_{S(d,q)} \bullet (k+1)$
d, q / α, β
$u_{S(\alpha,\beta)}(k+1)$
Space vector modulation

**Figure 2-10. PMSM Controller with Sliding Mode Observer in d, q Coordinates**

The BEMF amplitude:

$$\|e_s\| = \sqrt{(e_{s\beta}^2 + e_{s\alpha}^2)} \qquad \textbf{Eqn. 2-91}$$

Used for the rotor position calculation, therefore:

$$cos(\theta) = \frac{e_{s\beta}}{\|e_s\|} \cdot sgn(\omega_{el}) \qquad \textbf{Eqn. 2-92}$$

$$sin(\theta) = \frac{-e_{s\beta}}{\|e_s\|} \cdot sgn(\omega_{el}) \qquad \textbf{Eqn. 2-93}$$

Finally, the predicted BEMF in d, q coordinates, $e_{(d,q)}(k+1)$ needs to be aligned with the predicted $e_{(\alpha,\beta)}(k+1)$ in stator coordinates and (sine(θ)(,cosine(θ))(k+1). The d component of the BEMF in d, q coordinates is 0, therefore:

$$\hat{e}_{s(d,q)}(k+1) = (\|e\|(k+1), 0) \cdot sgn(\omega_{el}) \qquad \textbf{Eqn. 2-94}$$

### 2.4.2.10 Adaptive Speed Scheme for the d, q Observer

The sliding mode observer described in this document uses a model from the 4th order system matrix. The angular speed is used for calculation for the submatrix $A_{22}$ coefficients. Therefore, there is a need for adaptive angular speed estimation included into the feedback system.

Graphical representation of the angle error feedback used for the adaptive speed scheme is in Figure 2-11.

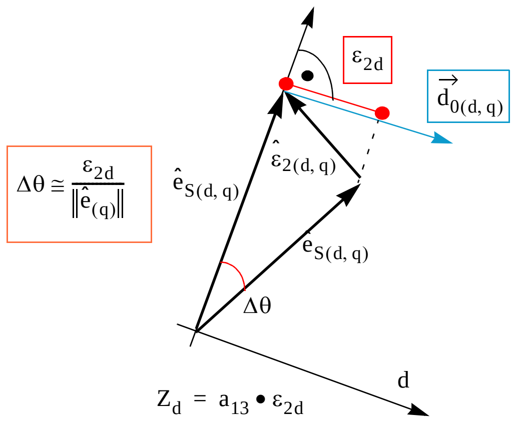

$\varepsilon_{2d}$
$\vec{d}_{0(d,q)}$
$\Delta\theta \cong \frac{\varepsilon_{2d}}{\|\hat{e}_{(q)}\|}$
$\hat{e}_{S(d,q)}$
$\hat{\varepsilon}_{2(d,q)}$
$e_{S(d,q)}$
$\Delta\theta$
d
$Z_d = a_{13} \bullet \varepsilon_{2d}$

**Figure 2-11. Adaptive Speed Scheme for the d,q Observer**

From the SMO equations, the relationship between the BEMF vector estimation error $\varepsilon_2$, and the SMO switching function z can be deduced:

$$Z_d = a_{13} \bullet \varepsilon_{2d} \qquad \textbf{Eqn. 2-95}$$

The following equation can be used as the adaptive speed scheme:

$$\frac{d}{dt}\hat{\omega}_{el} = g_{\omega} \cdot Z_{(d)} \qquad \textbf{Eqn. 2-96}$$

The adaptive scheme for the d, q observer is simpler than the α,β observer.

The estimated speed $\hat{\omega}_{el}$ for Equation 2-84 is finally iterated using the time discrete form of the adaptive scheme Equation 2-96:

$$\hat{\omega}_{el}(k+1) = g_{\omega} \cdot Z_{(d)}(k)T_s \qquad \textbf{Eqn. 2-97}$$

### 2.4.2.11 SMO in d, q versus $\alpha,\beta$ Coordinates Suitability for a Motor

The SMO version depends on the actual PM synchronous motor.

Many PM synchronous motors have the winding inductance variance negligible over rotor angle if compared to BEMF influence. In many application cases, the α,β coordinate model must be sufficient enough taking advantage of its stability and simplicity.

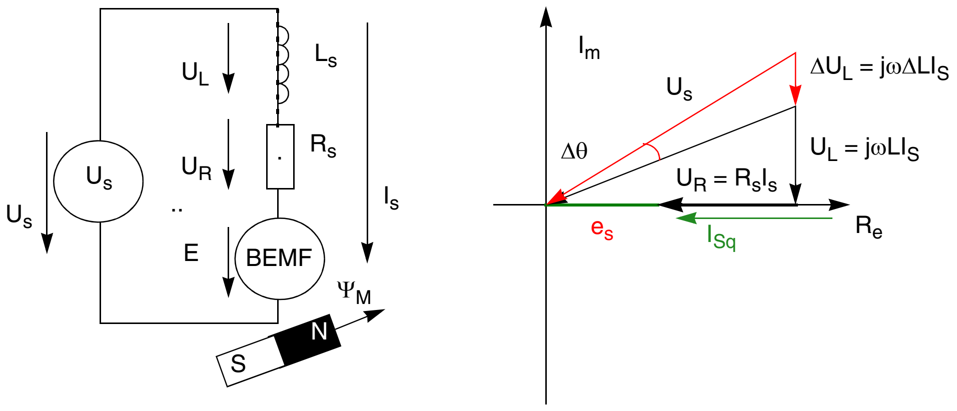


**Figure 2-12. Position Error Versus Inductance Variation**

When the motor has salient inductance characteristics judge if an α,β or d, q coordinate observer is appropriate for the actual PMSM. Evaluate the dependence of the position error $\Delta\theta$ on the inductance variation $\Delta L_s$ .

Here are some assumptions for simplification:

- High power factor, low angle shift between current and voltage vectors applied on the motor.
- The current vector with the d axis is 0. No field weakening.

Figure 2-12 can help understand the equation obtained for roughly estimating the stator inductance error versus the estimated angle error:

$$\Delta\theta = \frac{\omega_{el}\Delta L_s I_{Sq}}{e_S + R_S I_{Sq}} = \frac{\omega_{el}\Delta L_S I_{Sq}}{K_M\omega_{el} + R_s I_{Sq}} = \frac{\omega_{el}\Delta L_S I_{Sq}}{\frac{e_{S0}}{\omega_{el}}\omega_{el} + R_s I_{Sq}} \qquad \textbf{Eqn. 2-98}$$

In a very high speed range, $\omega_{el} >> 0$ and Equation 2-98 can be simplified as:

$$\Delta\theta \cong \frac{\Delta L_S I_{Sq}}{K_M} = \frac{\Delta L_S I_{Sq}}{\frac{e_{S0}}{\omega_{el0}}}$$ **Eqn. 2-99**

At low speed range $\omega_{el} >> 0$ and Equation 2-98 can be simplified as:

$$\Delta\theta \cong \frac{\omega_{el}\Delta L_S}{R_S I_S}$$ **Eqn. 2-100**

According to experience with the SW after practical measurements using the a,ß observer is suggested if the dependence of the angle error on the inductance variation is $\Delta\theta_{\max} \geq 2..3$ degree. Then the position error in the highest speed range can be evaluated according to:

$$\Delta\theta_{max}[deg] > \frac{180}{\Pi}\frac{\Delta L_S I_{Sq}}{K_M} = \frac{180}{\Pi}\frac{\Delta L_S I_{Sq}}{\frac{e_{S0}}{\omega_{el0}}}$$ **Eqn. 2-101**

And the lowest speed range:

$$\Delta\theta_{max}[deg] > \frac{180}{\Pi}\frac{\omega_{el}\Delta L_S}{R_S I_S}$$ **Eqn. 2-102**

If one of the equations exceeds the maximal error (2..3 degree), use the d, q coordinate observer because of its benefits. Otherwise, use the $\alpha$, $\beta$ observer due to better stability and simplicity. The above equations may help to judge if the motor inductance saliency requires the use of the d, q coordinate observer.

Some motors may have a high induction invariancy even when transferred to rotor related coordinates. This means the $L_d$, and $L_q$ values fluctuate with the current vector angle. More or less, a rotor position error even when using the d, q observer is obtained.

### 2.4.3 Open-Loop Start-up

The sensorless techniques described in this document are based on BEMF. This means the position can not be estimated at a zero or low speed. Therefore the motor start-up is provided with a vector of a constant current amplitude.

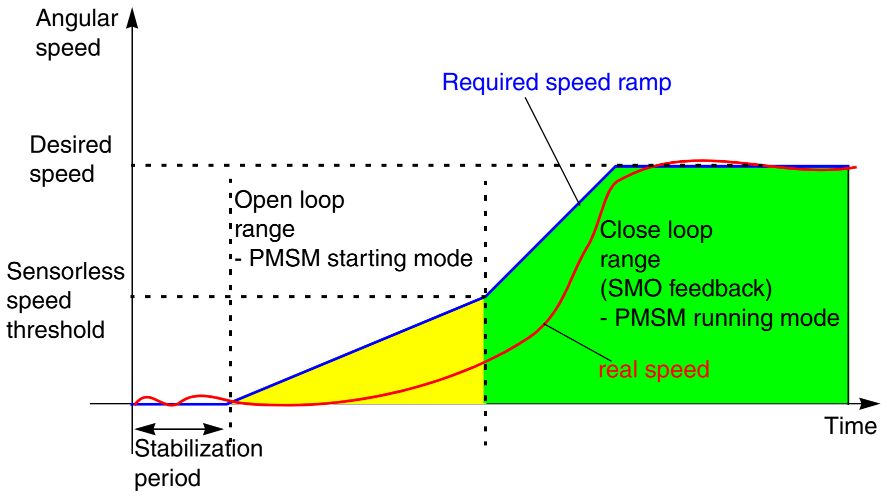


**Figure 2-13. PMSM Start-up**

A simplified motor start-up is shown in Figure 2-13. The start-up can begin with a constant vector. This is applied for a defined stabilization period. After a current vector of a constant amplitude forces the rotor rotation according to an acceleration ramp. This is the open loop range in Figure 2-13. This state is also called the PMSM starting mode. In this open-loop mode the stator and rotor flux are not aligned well, the motor is therefore controlled with lower efficiency. When the ramp reaches the sensorless speed threshold the close-loop mode with position and speed feedback from the SMO is entered. This state is also called the PMSM run mode. In this mode the rotor position and flux are being estimated and the PMSM motor is controlled according to stator field orthogonal to rotor field theory (FOC).

Stator and rotor flux vectors at the PMSM start-up states are displayed in Figure 2-14, Figure 2-15, and Figure 2-16.

- Figure 2-14 shows the motor alignment.
  Before the constant current vector is applied to the stator, the rotor position is not known. After a stabilization period the rotor flux must be aligned to the stator flux. This is true if the external load torque is low enough, compared to the torque produced by the alignment vector.
- Figure 2-15 shows the assumption of the rotor position at the start with the open-loop.
  The current is applied to the rotor d axis. At low external torque the rotor must be aligned to the d axis. Normally, due to acceleration and external torque, the rotor position is not that certain or even different. But the assumption of the starting algorithms is that the rotor flux is aligned with the d axis.
- Figure 2-16 shows the running mode.
  The stator flux is orthogonal to the rotor flux.

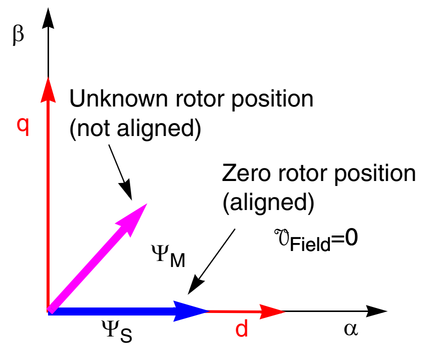


**Figure 2-14. Rotor Alignment Stabilization — PMSM Starting Mode**

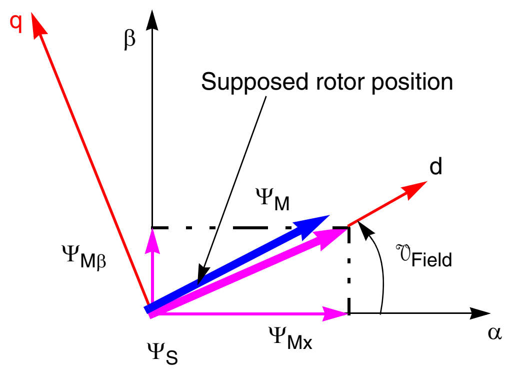


**Figure 2-15. Rotor and Stator Field – Open–Loop – PMSM Starting Mode**

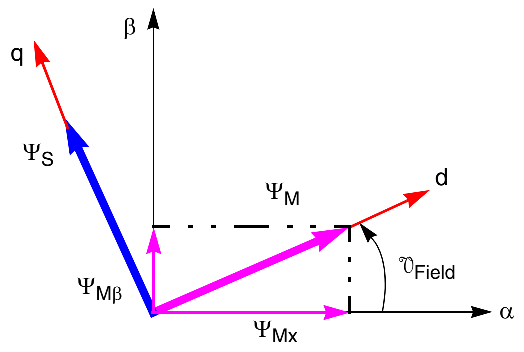


**Figure 2-16. Rotor and Stator Field – Close–Loop – PMSM Running Mode**

# Chapter 3
System Concept

## 3.1 System Specification

The system drives a three-phase PM synchronous motor. This application meets the following performance specifications:

- Targeted at the MC56F8013 (resp. 56F8346) digital signal controller
- Running on the MC56F8013 (resp. 56F8346) controller board and three-phase high voltage power stage
- Control technique incorporating:
  - — Sensorless vector control of three-phase PM synchronous motor with sliding mode observer
  - — Closed-loop speed control
  - — Bi-directional rotation
  - — Both motor and generator modes
  - — Close-loop current control
  - — Flux and torque independent control
  - — Starting up with alignment, open-loop start-up
  - — Field weakening is not implemented in this SW version
  - — Possible 62 μs sampling period (current and SMO loop) in the MC56F8013 without FreeMASTER
  - — 125 μ sampling period in MC56F8013 with FreeMASTER recorder
- Two SW versions according to the SMO implementation
  - — SMO calculated in d, q coordinates
  - — SMO calculated in $\alpha$,$\beta$ coordinates
- FreeMASTER software control interface, motor start/stop and speed setup
- FreeMASTER software monitor
  - — FreeMASTER software graphical control page, required speed, actual motor speed, start/stop status, DC-bus voltage level, motor current, and system status
  - — FreeMASTER software speed scope, observes actual and desired speeds, DC-bus voltage, and motor current
  - — FreeMASTER software high-speed recorder, reconstructed motor currents, and vector control algorithm quantities
- DC-bus over-voltage, under-voltage, and over-current protection

## 3.2 Application Description

A standard system concept is chosen for the drive. See Figure 3-1. The system incorporates the following hardware boards:

- Power supply 90 V - 260 V AC RMS, 5 A
- Three-phase high voltage power stage
- Three-phase PM synchronous motor and default configuration for motor TGdrives TGT3
- MC56F8013/MC56F8023 controller board

The MC56F8013 after populated on the controller board executes the control algorithm. In response to the user interface and feedback signals, it generates PWM signals for the three-phase high voltage power stage. High-voltage waveforms generated by the DC to AC inverter are applied to the motor.

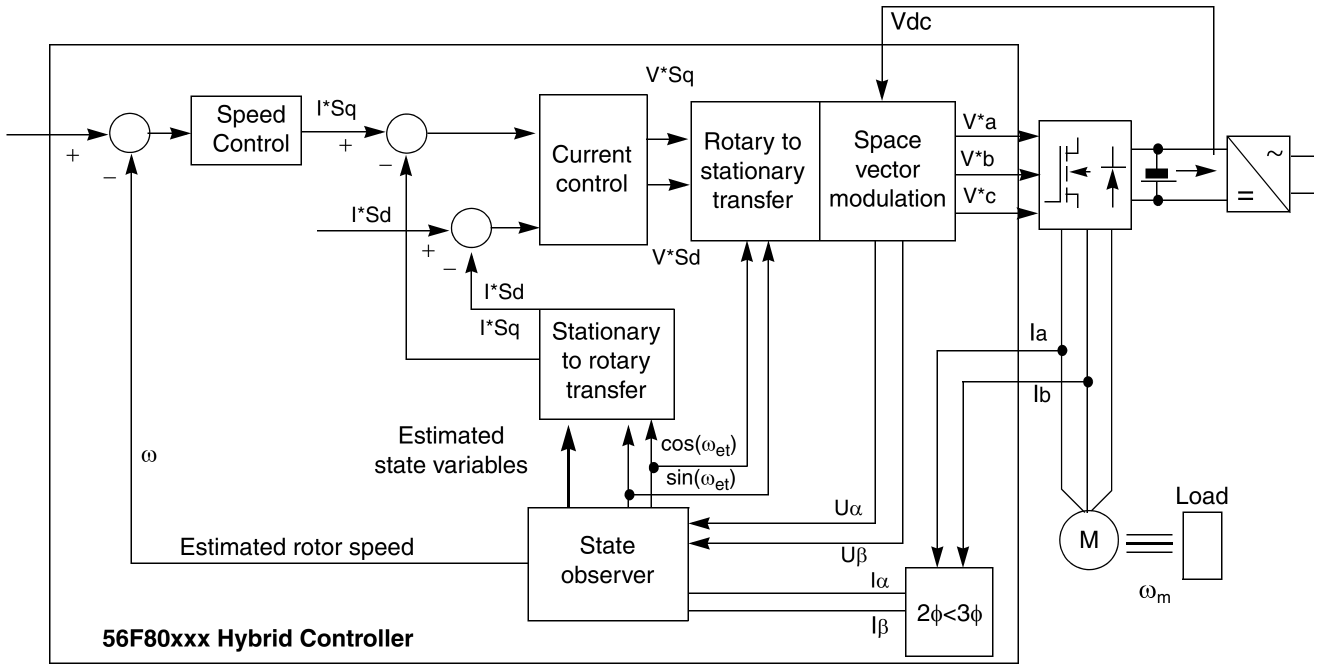


**Figure 3-1. System Concept**

## 3.3 Control Process

The user interface state is scanned periodically while the actual speed of the motor, DC-bus voltage, and phase currents are sampled. The speed command is calculated according to the state of the control signals, start/stop, and the required speed from FreeMASTER. The speed command is then processed by means of the speed ramp algorithm. The comparison between the actual speed command obtained from the ramp algorithm output and the measured speed generates a speed error. The speed error is input to the speed PI controller, generating a new desired level of reference for the torque producing component of the stator current.

The DC-bus current and voltage is sampled by the ADC. The ADC sampling is triggered by QuadTimer channel 3 and synchronized to the PWM signal. The motor currents are sampled from shunt resistors. The current is then used transformed into space vectors and used by the SMO for the position estimation and by the FOC algorithm.

The SMO evaluates the rotor position and outputs the rotor position sine, cosine and speed. Based on measured and estimated feedback signals, the FOC algorithm performs a vector control technique oriented to the rotor magnetizing flux space-vector as described in 2.3.2, “Description of the Vector Control Algorithm”. Two independent current PI control loops are executed to achieve the desired behavior of the motor. Output from the FOC is a stator voltage space-vector transformed by means of space-vector modulation into PWM signals. The three-phase stator voltage is generated by means of a three phase voltage source inverter and applied to the motor connected to the power stage terminals.

The application can be controlled via a FreeMASTER control page from a host PC. The FreeMASTER communicates via serial RS232 protocol. The RS232 is opto-isolated to achieve safety isolation from high voltage.

The drive application state machine manages the operating states of the drive. The drive has four states:

1. APP_RUN
2. APP_STOP
3. APP_FAULT
4. APP_INIT

The actual operating state is indicated by the FreeMASTER control page.

In the case of over-voltage, under-voltage, or over-current, the signals for the three-phase inverter are disabled and the fault state is displayed.

# Chapter 4 Hardware

## 4.1 Hardware Implementation

This application drives a three-phase PM synchronous motor. It consists of the following modules:

- Host PC
- MC56F8013/23 controller board
- Three-phase AC/BLDC high voltage power stage
- Three-phase PM synchronous motor

The application hardware system configuration is shown in Figure 4-1.

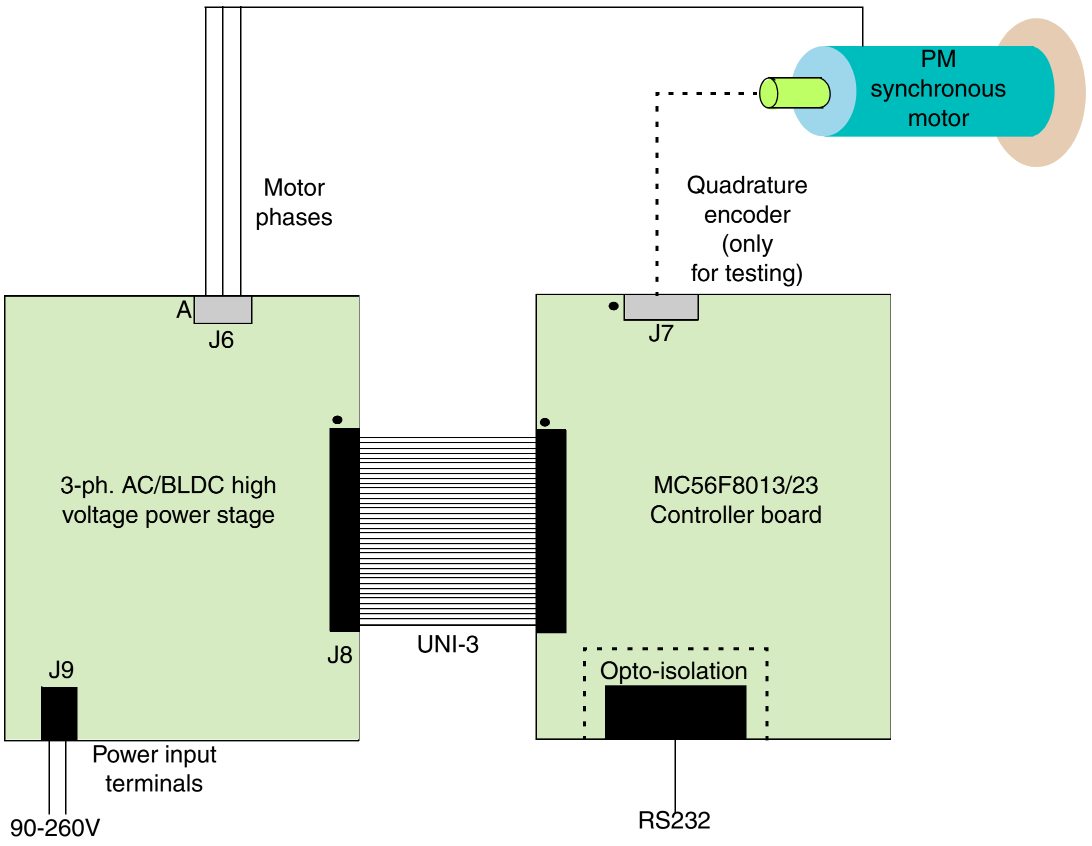

PM synchronous motor
Motor phases
Quadrature encoder (only for testing)
A
J6
J7
3-ph. AC/BLDC high voltage power stage
MC56F8013/23 Controller board
J8
UNI-3
J9
Opto-isolation
Power input terminals
90-260V
RS232

**Figure 4-1. Hardware System Configuration**

All system parts are supplied and documented in these references:

- MC56F8013/23 controller board:
  - Using Freescale's MC56F8013 or MC56F8023 as the controller
  - Supplied as an MC56F8013 or MC56F8023 controller board
  - Described in the MC56F8013 or MC56F8023 *Controller Board User's Manual*
- Three-phase AC/BLDC high voltage power stage:
  - High-voltage three-phase power stage with single-phase input 115/230 Volt AC and 750 VoltAmp variable voltage three-phase IGBT bridge output.
  - Described in the three-phase AC/BLDC *High Voltage Power Stage User's Manual*

A detailed description of each individual board can be found in the appropriate user manual, or on the Freescale web site www.freescale.com. The user manuals include a schematic of the board, a description of individual function blocks, and a bill of materials (parts list).

## 4.2 MC56F8013 or MC56F8023 Controller Board

The MC56F8013 or MC56F8023 controller board is based on an optimized PCB and power supply design. It demonstrates the abilities of the MC56F8013 or MC56F8023 and provides a hardware tool to help in the development of applications using the MC56F8013 or MC56F8023 targeted at motor control applications.

The MC56F8013 or MC56F8023 controller board can be populated either by the MC56F8013 or MC56F8023 parts. PCBs marked with the numbers 00216A01 and 00216A02 are populated by an MC56F8013 device. PCBs marked with the numbers 00216B02 are populated by an MC56F8023 device.

The controller board is an evaluation module type of board. It includes an MC56F8013 or MC56F8023 part, encoder interface, tacho-generator interface, communication options, digital and analogue power supplies, and peripheral expansion connectors. The expansion connectors are for signal monitoring and user feature expandability. Test pads are provided for monitoring critical signals and voltage levels.

The MC56F8013 or MC56F8023 controller board is designed for the following purposes:

- To allow new users to become familiar with the features of the MC56F801x or MC56F802x architecture.
- To serve as a platform for real-time software development. The tool suite allows to develop and simulate routines, download the software to on-chip memory, run the software, and debug it using a debugger via the JTAG/OnCE™ port. The breakpoint features of the OnCE port specify complex break conditions easily and execute software at full-speed until the break conditions are satisfied. The ability to examine and modify all user accessible registers, memory, and peripherals through the OnCE port simplifies considerably the task of the developer.
- To serve as a platform for hardware development. The hardware platform enables external hardware modules to be connected. The OnCE port's unobtrusive design means all of the memory on the digital signal controller chip is available to the user.

The board facilitates the evaluation of various features present in the MC56F8013 or MC56F8023 and can be used to develop real-time software and hardware products based on the MC56F8013 or the MC56F8023. It provides the features necessary to write and debug software, demonstrate the functionality of that software, and to interface with the customer's application specific device(s).

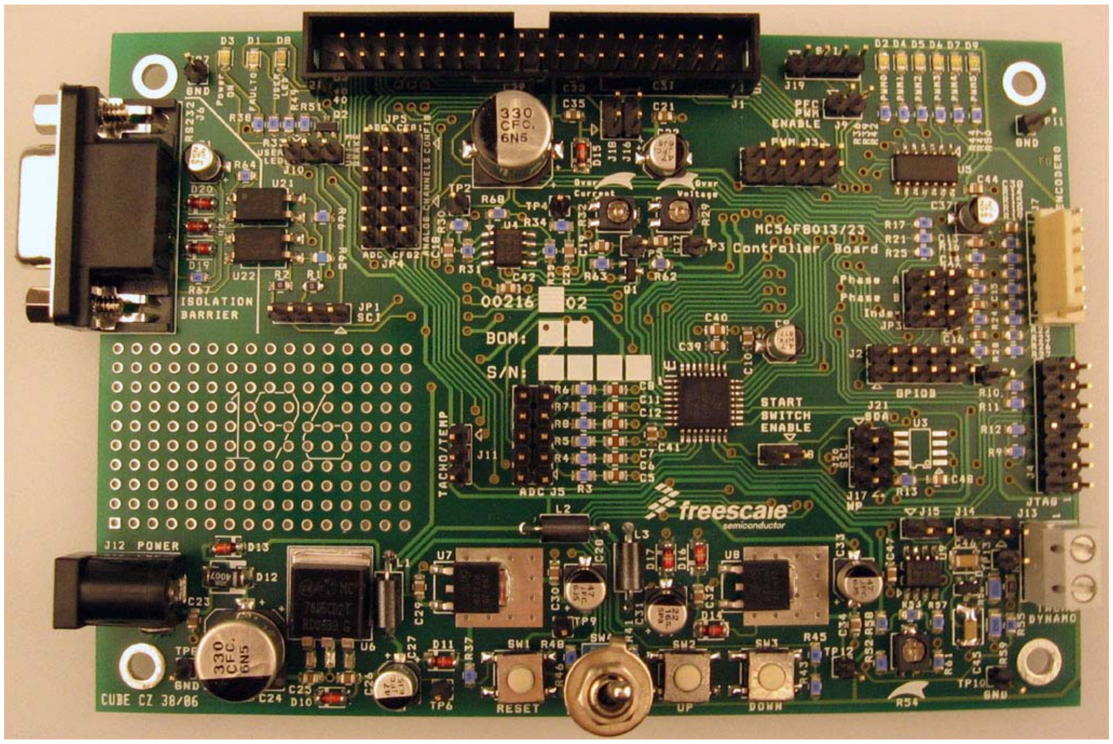


**Figure 4-2. MC56F8013/23 Controller Board Top View**

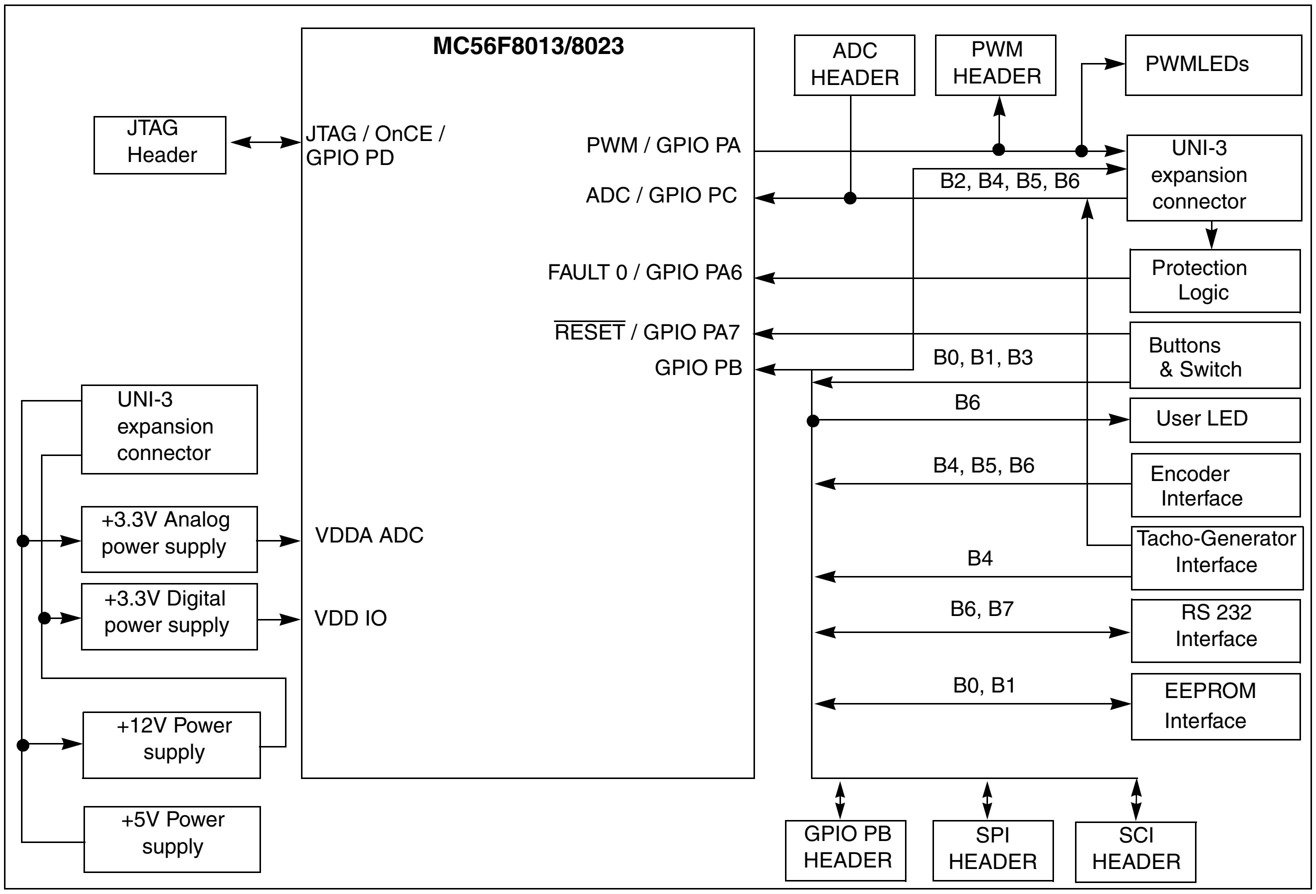


**Figure 4-3. Block Diagram of the MC56F8013/23 Controller Board**

The MC56F8013/23 controller board is flexible enough to allow full exploitation of the MC56F8013/8023's features to optimize the performance of the user’s end product. See Figure 4-2 and Figure 4-3.

## 4.3 Three-Phase AC/BLDC High Voltage Power Stage

Freescale’s three-phase high-voltage (HV) AC power stage is a 750-voltamps (one horsepower), three-phase power stage that operates off DC input voltages from 140 to 325 volts, and AC line voltages from 100 to 240 volts. In combination with one of the controller boards, it provides a software development platform that allows algorithms to be written and tested without the need to design and build a power stage. It supports a wide variety of algorithms for both AC induction and brushless DC (BLDC) motors.

The high-voltage AC power stage has a printed circuit board. The printed circuit board contains an input rectifier, brake IGBT and diode, bridge IGBTs, IGBT gate drive circuits, analogue signal conditioning, low-voltage power supplies, and some large passive power components. All of the power devices that need to dissipate heat and a temperature sensor are mounted on a heatsink situated below the printed circuit board. See Figure 4-4.

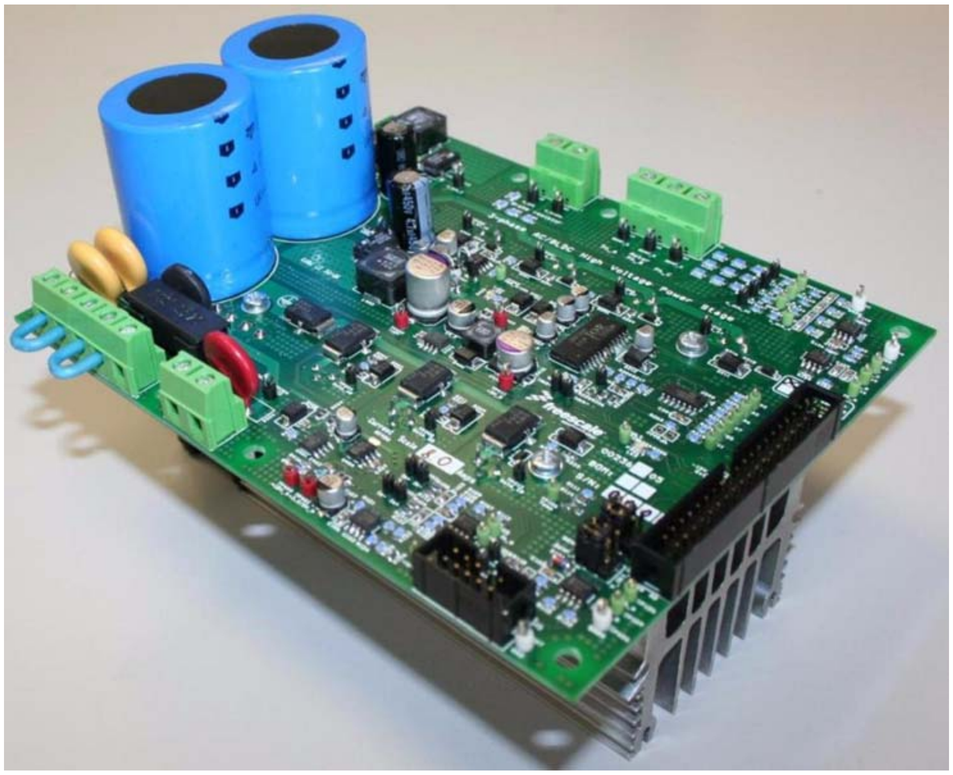


Figure 4-4. Three-Phase AC/BLDC High Voltage Power Stage

Figure 4-5 shows a block diagram. Input connections are made via the 40-pin ribbon cable connector J8. Power connections to the motor are made on output connector J6. Phase A, phase B, and phase C are labeled Ph_A, Ph_B, and Ph_C on the board. Power requirements are met by a single external 140 to 325 volt DC power supply or an AC line voltage. Either input is supplied through connector J9. An external brake resistor can be connected via connector J4. The power stage can be extended by an external PFC board. The PFC board connection is made via power connector J5 and signal connector J2.

Current measuring circuitry can be set up for four or eight amps full scale. Both bus and phase leg currents are measured. An over-current trip point is set at 10 amps.

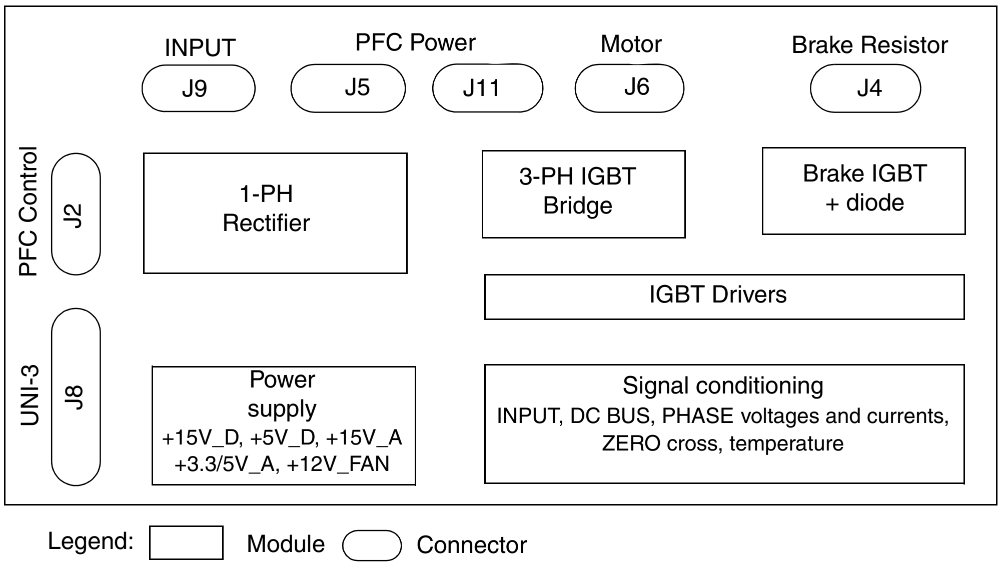


**Figure 4-5. Three-Phase AC/BLDC Power Stage Block Diagram**

**Table 4-1. Electrical Characteristics of Three-Phase AC/BLDC Power Stage**

| Characteristic | Symbol | Min | Type | Max | Units |
|---|---|---|---|---|---|
| DC input voltage | $V_{dc}$ | 140 | — | 325 | V |
| AC input voltage | $V_{ac}$ | 100 | — | 240 | V |
| Logic 1 input voltage | $V_{IH}$ | 1.5 | — | 1.7 | V |
| Logic 0 input voltage | $V_{IL}$ | 0.9 | — | 1 | V |
| Input resistance | $R_{In}$ | | 10 | — | kΩ |
| Analogue output range* | $V_{Out}$ | 0 | | 3.3 | V |
| Bus current sense voltage | $I_{Sense}$ | — | 206.25 | — | mV/A |
| Bus current sense offset | $I_{offset}$ | — | $+V_{REF}$ | — | V |
| Bus voltage sense voltage | $V_{Bus}$ | — | 8.09 | — | mV/V |
| Bus voltage sense offset | $V_{offset}$ | — | 0 | v | V |
| Continuous output current ** | $I_C$ | — | | 10 | A |
| Deadtime (built in IR2133) | $t_{off}$ | — | 250 | — | ns |

* Range set according +3.3V_A/+5V_A Power Supply.
** The values are measured at 25×C, for other temperatures the values may be different.

## 4.4 Motor Specifications — Example

The motor used in this application is a standard production three-phase PM synchronous motor with an incremental encoder mounted on the shaft. The motor is start connected. The motor and sensor have the following specifications:

**Table 4-2. Specifications of the Motor and Incremental Sensor**

| | | |
|---|---|---|
| **Motor Specification:** | **Motor Type:** | **Three-Phase PM Synchronous Motor TG drives TGT3-0065-30-320/ToPS2X** |
| | Nominal voltage (line-to-line) | 380 V RMS |
| | Nominal speed | 3000 RPM |
| | Nominal current (phase) | 1.04A RMS |
| | Nominal torque | 0.65 Nm |
| **Motor Model Parameters** | Stator winding resistance | 18.5 Ohm |
| | Stator winding inductance d axis | 20.5 mH |
| | Stator winding inductance q axis | 17.5 mH |
| | Number of pole-pairs | 3 |
| **Position Sensor Specification:** | Manufacturer: | INDUcoder |
| | Type: | ES 28-6-1024-05-D-R |
| | Line Count | 1024 |
| | Output | 5V ± 10% TTL |

**NOTE**

The sensor is not needed for this sensorless application, but it is used for demonstration and fine-tuning.

# Chapter 5
Software Design

## 5.1 Introduction

This section describes the software design of the sensorless PMSM vector control application. First, the numerical scaling in fixed-point fractional arithmetic of the DSC is discussed. Then, issues such as speed and current sensing are explained. Finally, the control software implementation is described. This chapter is to help understand the software designed.

## 5.2 Application Variables Scaling

### 5.2.1 Fractional Numbers Representation

The sensorless PMSM vector control application uses a fractional representation for all real quantities, except time. The N-bit signed fractional format is represented using 1. [N-1] format (1 sign bit, N-1 fractional bits). Signed fractional numbers (SF) are in the following range:

$$-1.0 \leq SF \leq +1.0 - 2^{-[N-1]} \qquad \textbf{Eqn. 5-1}$$

For words and long-word signed fractions, the most negative number that can be represented is –1.0 and the internal representation is 0x8000 and 0x80000000. The most positive word is 0x7FFF or $1.0 - 2^{-15}$ and most positive long-word is 0x7FFFFFFF or $1.0 - 2^{-31}$.

### 5.2.2 Scaling of Analogue Quantities

Analogue quantities such as voltage, current, and frequency are scaled to the maximum measurable range dependent on the hardware. The following equation shows the relationship between a real and a fractional representation:

$$\mathit{Fractional\ Value} = \frac{\mathit{Real\ Value}}{\mathit{Real\ Quantity\ Range}} \qquad \textbf{Eqn. 5-2}$$

Where:

- Fractional value is a fractional representation of the real value [Frac16]
- Real value is the real value of the quantity [V, A, rpm]
- Real quantity range is the maximum range of the quantity, defined in the application [V, A, rpm]

The above scaling can be demonstrated on a DC-bus voltage and motor phase voltage as an example. All variables representing voltage are scaled the same in the application. They are scaled to maximum measurable voltage range of the power stage. For the demo hardware board the range is $V_{MAX}$= 407 V. Variable values in fractional format are defined by the following equation:

$$(Frac16)voltage\_variable = \frac{V_{MEASURED}}{V_{MAX}} \qquad \textbf{Eqn. 5-3}$$

The fractional variables are internally stored as signed 16-bit integer values and can be evaluated as follows:

$$(Int16)voltage\_variable \; = \; (Frac16)voltage\_variable \cdot 2^{15} \qquad \textbf{Eqn. 5-4}$$

The maximum range of analogue quantities used by the application is defined by #define statements in the application configuration files. The default scaling ranges for the reference design hardware set-up are as follows:

```
#define HW_APP_VOLT_MAX_V          407.0                      /* Volts */
#define HW_APP_CURR_RANGE_MAX_A    2*4.0                      /* Amps */
#define MOTOR_APP_SPEED_MAX_RPM    6000.0                     /* RPM */
```

MOTOR_APP_SPEED_MAX_RPM corresponds to a full range of the ADC converter input voltage (0-3.3 V). For motor phase-current sensing, the zero current level is shifted into the middle of this range (=1.65 V). The maximum positive and negative phase-current that can be sensed is HW_APP_CURR_MAX_A = CURRENT_SCALE/2. If the current sensing range of the power stage is from –4.0 Amps to +4.0 Amps, the value of CURRENT_SCALE is set to the value of 8.0 Amps. If the current sensing range of the power stage is from –8.0 Amps to +8.0 Amps, the value of MOTOR_APP_SPEED_MAX_RPM is set to the value of 16.0 Amps.

## 5.2.3 Scaling Angles

The angles such as rotor flux position, are represented as a 16-bit signed fractional value in the range [–1,1] that corresponds to the angle in the range [–pi, pi). In a 16-bit signed integer value, the angle is represented as follows:

$$-pi \approx 0x8000 \qquad \textbf{Eqn. 5-5}$$

$$pi \cdot (1.0 - 2^{-15}) \approx 0x7FFF \qquad \textbf{Eqn. 5-6}$$

## 5.2.4 Scaling Parameters

Real-value parameters in equations such as the SM observer and decoupling voltage are represented as 16-bit signed fractional values in the range [–1,1). The real parameter value (Ohms, Henry) has to be adjusted to correspond to the scaling range of the analogue value that forms the particular equation. The adjusted value is then split into a fractional range of [–1,1] and an N-bit scale. The scaling process can be explained by an example of the Ohms law equation.

$$V_{real} \; = \; R \cdot I_{real} \qquad \textbf{Eqn. 5-7}$$

$$V_{Frac16} \cdot V\_MAX \; = \; R \cdot I_{Frac16} \cdot I\_MAX \qquad \textbf{Eqn. 5-8}$$

$$V_{Frac16} = \left(R \cdot \frac{I\_MAX}{V\_MAX}\right) \cdot I_{Frac16} = R_{adjusted} \cdot I_{Frac16} \quad \textbf{Eqn. 5-9}$$

Substitute the following values:

R = 300 Ohms, I_MAX = 8 Amps, V_MAX = 407 Volts

The $R_{adjusted}$ can be evaluated:

$$R_{adjusted} = R \cdot \frac{I\_MAX}{V\_MAX} = 300 \cdot \frac{8}{407} = 5.8968 \quad \textbf{Eqn. 5-10}$$

The $R_{adjusted}$ is out of range of the signed fractional number. Right shift the value by $R_{Scale}$ bits to fit into the wanted range. Therefore, a scale (shift) part of the parameter is introduced. For this example shift the result by $R_{Scale}$=3 bits. The resistor value scaled to the signed fractional range is as follows:

$$R_{Frac16} \cdot 2^{R_{Scale}} = R_{adjusted} \quad \textbf{Eqn. 5-11}$$

$$R_{Frac16} = R \cdot \frac{I\_MAX}{V\_MAX} \cdot 2^{-R_{Scale}} = 300 \cdot \frac{8}{407} \cdot 2^{-3} = 0.7371 \quad \textbf{Eqn. 5-12}$$

The Ohms law equation scaled into signed fractional arithmetic is evaluated as follows:

$$V_{Frac16} = \left(\left(R \cdot \frac{I\_MAX}{V\_MAX} \cdot 2^{-R_{Scale}}\right)(I_{Frac16})\right) \cdot 2^{R_{Scale}} \quad \textbf{Eqn. 5-13}$$

Notice that the final multiplication result has to be left-shifted back by $R_{Scale}$ bits to stay within the proper range of the $V_{Frac16}$ variable.

All algorithm and motor parameters are scaled to their 16-bit or possibly 32-bits fractional representation. For most parameters there are two definitions. One evaluates the parameter fractional representation and the other defines the required N-bit shift = scale. Below are the parameters A11 of the sliding mode observer as an example:

```
#define MOTOR_EST_A11TS_FRAC-0.52   /* a11*Ts=-(Rs/Ls*Ts) SMO model coefficient */
#define MOTOR_EST_A11TS_SC -3       /* a11*Ts=MOTORA_EST_A11TS_F16*2^MOTORA_EST_A11TS_SC */
```

The N-bit shift constant can be defined as both negative and positive. If negative, a left shift is applied to the scaled variable.

It is recommended using the Excel spreadsheet included in the SW files to evaluate the fractional and shift constants of the parameters.

In some constant calculations (Excel spreadsheet included in the SW) a scaling factor is introduced. The reason is universality of some equations for possible reuse with system quantities represented differently from the fractional.

The voltage scaling factor is:

$$S_V = \frac{V_{SYSRANGE}}{V_{MAX}} \quad \textbf{Eqn. 5-14}$$

Where:

- SV is the scaling factor
- VSYSRANGE is the range of system representation voltage
- VMAX is the range of real voltage

In this application, the system representation is a fractional number of the range as in Equation 5-1. Therefore, the scaling coefficient is usually:

$$S_V = \frac{1}{V_{MAX}} \tag{Eqn. 5-15}$$

Then the real variable is:

$$V_{real} = \frac{(Frac16)voltage\_variable}{S_V} \tag{Eqn. 5-16}$$

The system parameters are then calculated accordingly:

$$R_{adjusted} = R \cdot \frac{S_V}{S_I} = 300 \cdot \frac{1/407}{1/8} = 5 \cdot 8968 \tag{Eqn. 5-17}$$

## 5.3 Application Overview

### 5.3.1 SW Versions

The application described in this DRM is prepared for two versions of the sliding mode observer:

- Sliding mode observer with the α,β coordinate model – PMSM_SensorlessCL_AlpBetNDHRel
- Sliding mode observer with the d, q coordinate model – PMSM_SensorlessCL_DQHRel

The SW version suitable for an actual motor is in Section 2.4.2.11, “SMO in d, q versus a,b Coordinates Suitability for a Motor”. For most cases, the α,β coordinate model is sufficient.

Besides the MC56F8013, it is possible to use of the MC56F8346 processor with a dedicated MC56F8346 controller board, therefore there are two options for both SMO’s.

- SW for MC56F8013
- SW for MC56F8346

In the following sections the SW version implementation using the MC56F8013 is described. This is the main targeting device for the application. The SW for the MC56F8346 differs mainly in an allocation of processor peripherals.

#### 5.3.1.1 Overview

The application software is interrupt driven running in real time. There are three periodic interrupt service routines executing the major motor control tasks. See Figure 5-1.

The QuadTimer (TMR) channel 1 interrupt service routine is executed on a compare every 1 ms. It performs a speed control loop.

The PWM reload interrupt service routine is executed only once on the first PWM reload after application initialization, PWM start. It performs a fast current control loop. After, the reload is disabled and the ADC end of scan is used for the fast current control loop.

The ADC end of scan interrupt service routine is executed at the end of an ADC scan. The ADC scan is executed according to the SW setting (PWM_RELOAD_FREQ) every second or on each PWM half reload. As default, the SW setting is a 125 μs period. It performs a fast control loop with sliding mode observer calculation and current control.

The PWM fault interrupt service routine is executed on an over-current event to an over-current fault or a DC-bus over-voltage condition. It is executed only if the fault condition occurs.

The background loop is executed in the main application. It s non-critical timing tasks such as the application state machine and FreeMASTER communication polling.

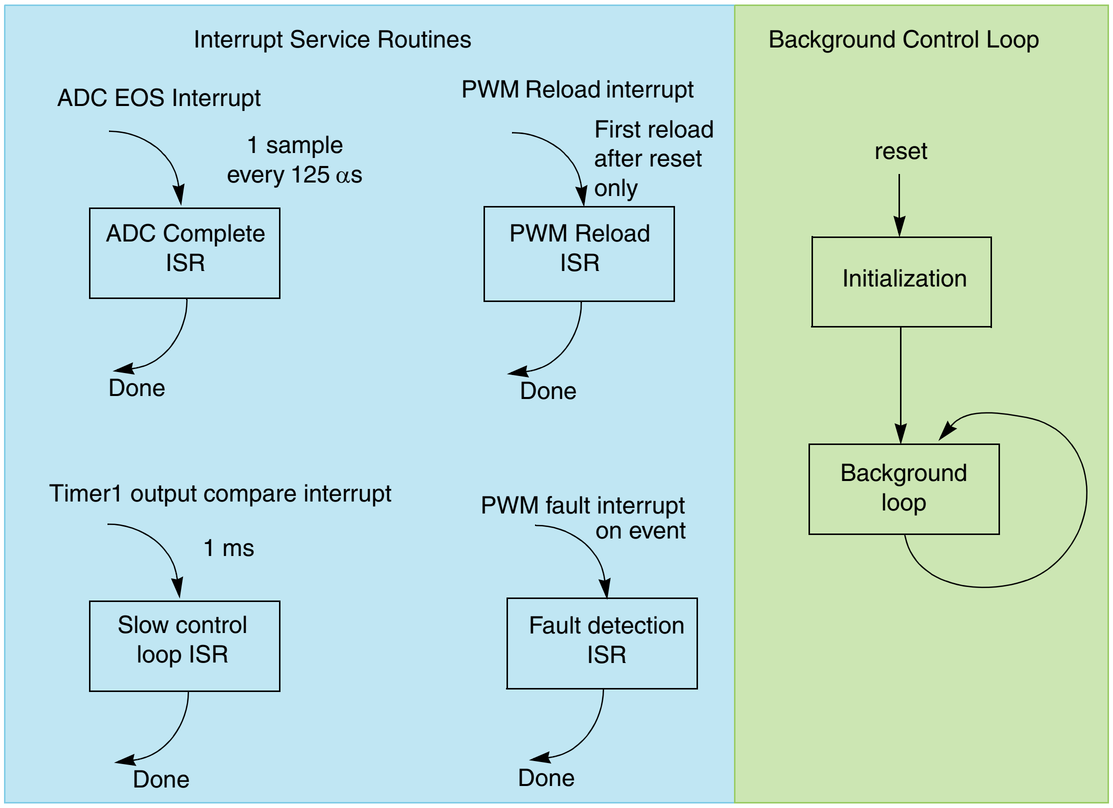


**Figure 5-1. Main Data Flow**

The individual processes of the control routines are described in the following sections.

## 5.3.2 ADC End of Scan Timing and PWM Reload Interrupts

The fast current control loop is executed in the ADC CompleteISR and the ADC scan is synchronized to the PWM_reload_sync signal. After ADC sampling has finished, the ADC CompleteISR is executed.

The PWM module is configured to run in centre-aligned mode with a switching frequency of 16 kHz at a 32 MHz bus clock (PWM cycle period = 62.5 μs). The PWM_reload_sync signal is generated every second PWM cycle with a 125 μs period. The PWM_reload_sync is connected to a secondary input pin 3. This is the signal of the TMR module. An output of the TMR channel 3 is connected to the SYNC0 signal used to trigger the ADC in simultaneous mode. The TMR channel 3 is configured in triggered count mode. A connection link between the PWM module, TMR module, and the ADC module enables defining the exact multiple time instants of ADC sampling that are synchronized to the generated PWM signal. An overview of module interconnections is shown in Figure 5-2.

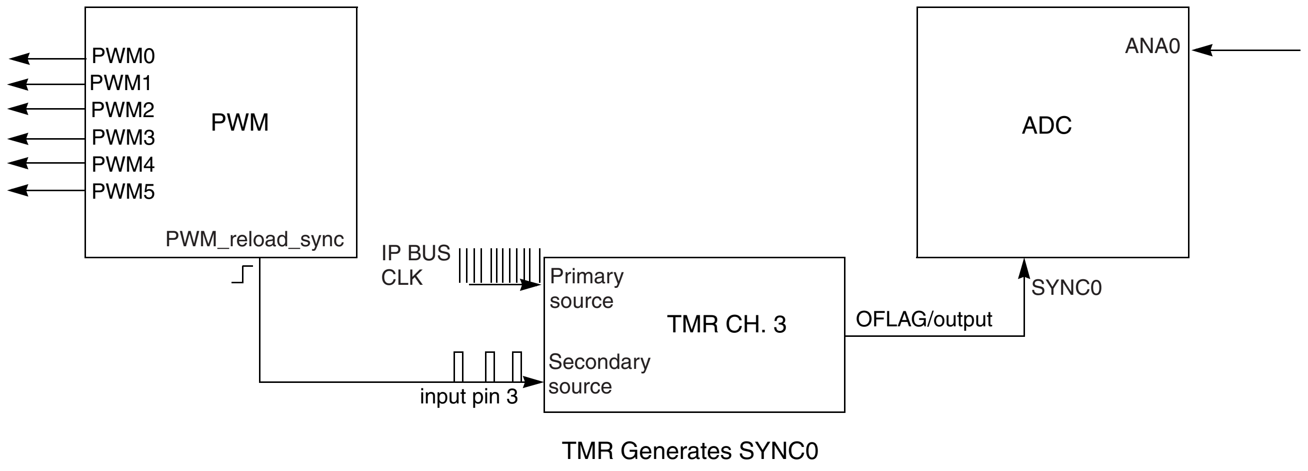


**Figure 5-2. ADC Triggering**

## 5.3.3 Current Sensing

Phase currents are measured by a shunt resistor in each phase. A voltage drop in the shunt resistor is amplified by an operational amplifier, and shifted up by 1.65 V. The resultant voltage is converted by an A/D converter. See Figure 5-3.

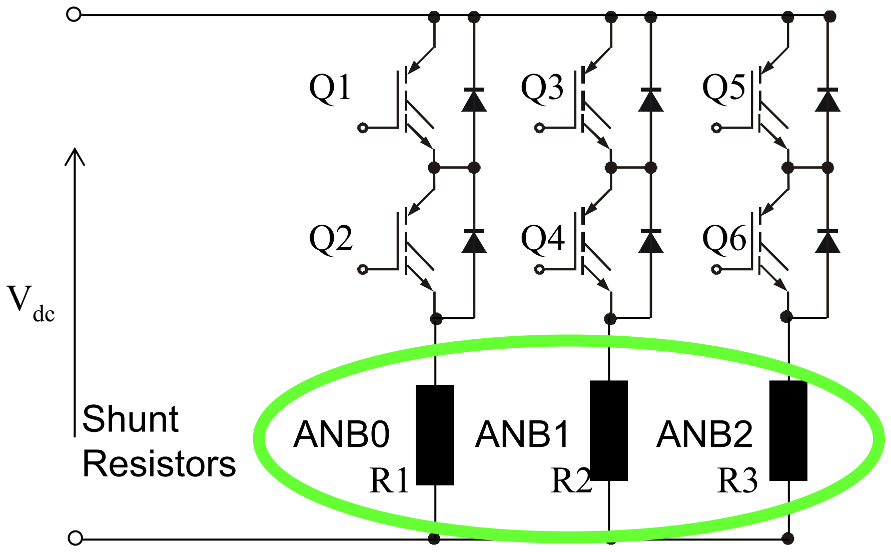


**Figure 5-3. Current Shunt Resistors**

As shown in Figure 5-3, the currents cannot be measured at each instant. For example, the current flows through Phase A (and shunt resistor R1) only if transistor Q2 is switched on. Likewise, the current in Phase B can be measured if transistor Q4 is switched on, and the current in Phase C can be measured if transistor Q6 is switched on. In order to get an actual instant of current sensing, voltage shape analysis must be performed.

The voltage shapes of two different PWM periods are shown in Figure 5-6. The voltage shapes correspond to center-aligned PWM sinewave modulation. The best instant of current sampling is in the middle of the PWM period where all bottom transistors are switched on.

To set the exact instant of sampling, the DSP56F80x family offers the ability to synchronize ADC and PWM modules via the SYNC signal. This exceptional hardware feature patented by Freescale is used for current sensing. The PWM outputs a synchronization pulse connected as an input to the synchronization module T3 (Quad timer channel 3). A high-true pulse occurs for each reload of the PWM regardless of the LDOK bit state. The intended purpose of T3 is to provide a user-selectable delay between the PWM SYNC signal and the updating ADC values. A conversion process can be initiated by the SYNC input an output of T3.

The timing diagram in Figure 5-4 shows how the ADC sampling to the PWM update is performed. The events are executed in the following steps:

1. After PWM (system) restart, the PWM is set for every opportunity reload with both full and half cycle reload. Therefore, when the PWM counter reaches zero a PWM reload occurs and PWM_reload_sync is generated.
2. The PWM reload ISR is first time entered.
3. Second PWM reload occurs and PWM_reload_sync is generated at PWM counter modulo value (half reload). TMR channel 3 count is triggered by the PWM_reload_sync signal connected to its secondary source input. The timer starts counting up from zero.
4. The PWM reload ISR is second time entered. The PWM reload ISR sets the PWM module for every fourth opportunity reload. This trick gives the functionality of the PWM_reload_sync being generated each a second half cycle reload.
5. A compare on T3 register occurs. A rising edge on the SYNC0 input of the ADC module starts an ADC conversion.
6. The ADC conversion is finished. The ADC end of scan 1 flag (EOSI1) is set.
7. The ADC end of scan ISR is entered. The ADC end of scan ISR – part1 calculates the current regulator. The new values are stored in the PWM value registers (VAL0-5) at the end of the ADC end of scan ISR – part1.
8. The PWM reload occurs and the PWM is loaded with the buffered values from the PWM value registers (VAL0-5). TMR channel 3 count is triggered by the PWM_reload_sync signal connected to its secondary source input. The timer then starts counting up from zero.
9. The ADC end of scan ISR – part 2 calculates SMO for position and speed estimation.
10. Repeat steps.

The timing scheme where the current sensing alternates with the PWM reload is used to get the best and dynamic sampling performance for the PMSM current control. The ADC end of scan ISR – part 1 duration is low enough to calculate current regulator with PWM update between two half cycle reloads. The ADC end of scan ISR – part 1 and part 2 duration is low enough to go between two PWM cycles.

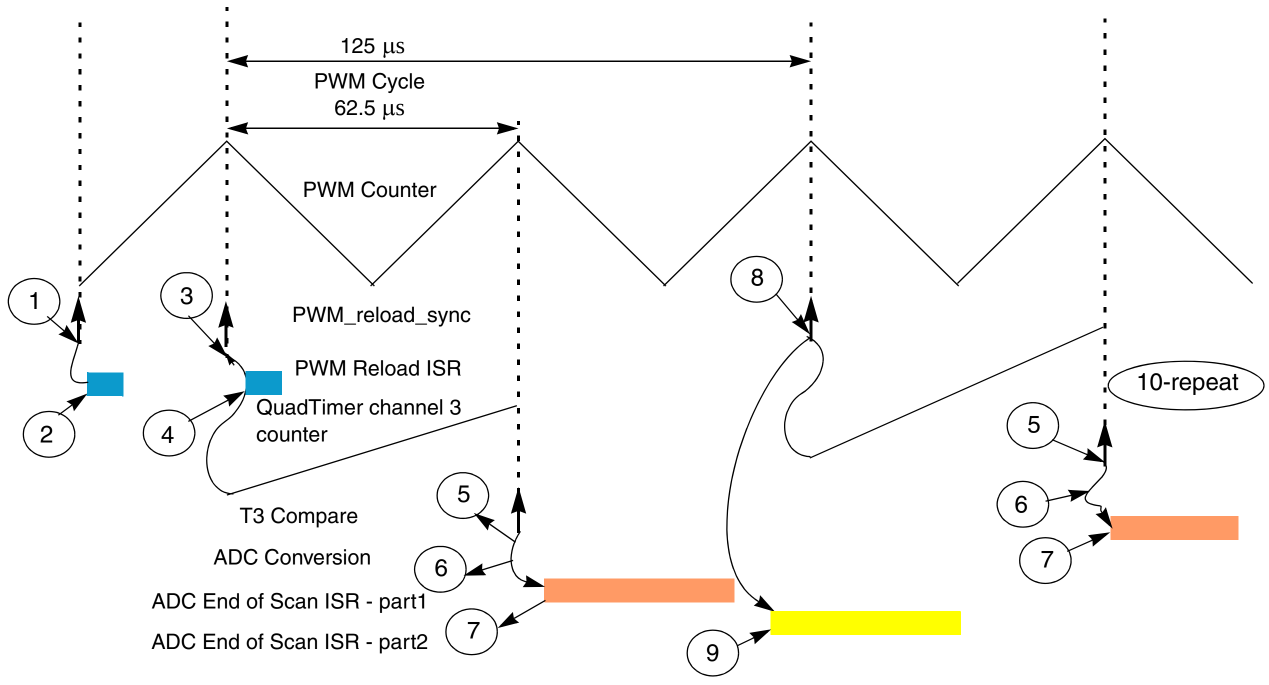


**Figure 5-4. Time Diagram of PWM and ADC Synchronization and ADC End Of Scan Interrupts**

All three currents cannot be measured at an arbitrary voltage shape. The PWM period II in Figure 5-6 shows an instant when the bottom transistor of phase A is switched on for a very short time. The current can not be accurately measured if the on-time is shorter than a critical time. The critical time is given by hardware configuration, transistor commutation times and response delays of the processing electronics. Therefore, only two currents are measured and a third current is calculated from the following equation:

$$0 = i_A + i_B + i_C \qquad \textbf{Eqn. 5-18}$$

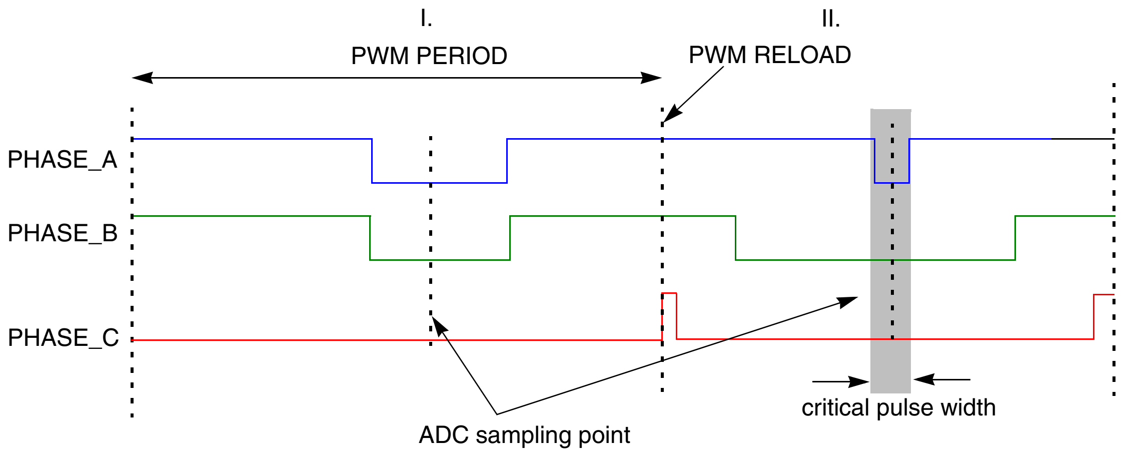


**Figure 5-5. Voltage Shapes of Two Different PWM Periods**

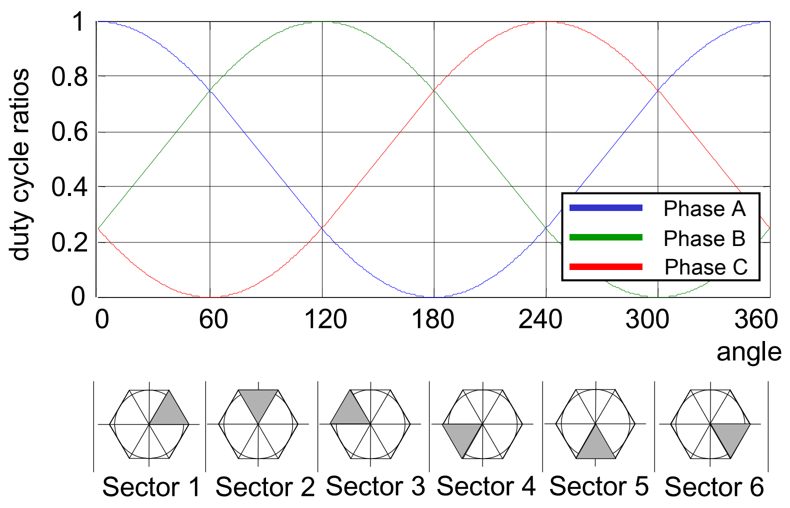


**Figure 5-6. 3—Phase Sinewave Voltages and Corresponding Sector Value**

A decision must be made about which phase current is to be calculated. The simplest technique is to calculate the current of the most positive voltage phase. For example, phase A generates the most positive voltage within section 0 – 60 °, phase B within section 60 ° – 120 °, and so on. See Figure 5-6.

In this case, the output voltages are divided into six sectors, as shown in Figure 5-6. The current calculation is then made according to the actual sector value.

Sectors 1 and 6:

$$i_A = -i_B - i_C$$ ***Eqn. 5-19***

Sectors 2 and 3:

$$i_B = -i_A - i_C$$ ***Eqn. 5-20***

Sectors 4 and 5:

$$i_C = -i_B - i_A$$ ***Eqn. 5-21***

***Eqn. 5-22***

**NOTE**

The sector value is used for only the current calculation and has no other meaning in the sinewave modulation. If any type of space vector modulation is used, the sector value as part of space vector calculation is obtained.

### 5.3.4 Sliding Mode Observer with Adaptive Velocity Estimation Implementation

The SMO for position and speed is implemented in the *EstimProcessing()* function.

#### 5.3.4.1 $\alpha,\beta$ Observer Implementation

The code version of the observer with α,β stator related coordinates uses the *PMSM_SL_SMOBemfSpedObservSclUniv32()* function for the SMO feedback. The function realizes the sliding mode observer according to Equation 2-56 with the system coefficients according to Equation 2-57, and Equation 2-58. The *EstimProcessing*() uses the *smoBemfStruct* structure variable for state variables and coefficients. The estimated current $\hat{\mathbf{i}}_{(\alpha,\beta)}$ vector state variable is the component called *curr*. The estimated bemf $\hat{\mathbf{e}}_{(\alpha,\beta)}$ is the component called *bemfAlphaBeta*. These system variables are implemented as a 32-bit fractional representation. This 32-bit resolution is necessary to guarantee a good enough iteration precision at low amplitudes. The SMO feedback gain coefficients are part of the *smoBemfStruct* structure. The observer feedbacks are not constant, but the coefficients $g_1$, and $g_2$ are multiplied by the BEMF amplitude:

$$g_1 k_1 T_s = \|\hat{e}_S\| \cdot (g_1 k_1 T)_{multiple} \qquad \text{Eqn. 5-23}$$

$$g_2 k_1 T_s = \|\hat{e}_S\| \cdot (g_2 k_1 T)_{multiple} \qquad \text{Eqn. 5-24}$$

The structure *smoBemfStruct* parameter markings compared to the Chapter 2, “Control Theory” are:

**Table 5-1. smoBemfStruct Parameters**

| Element | Definition | Symbol |
|---|---|---|
| bemfAlphaBeta | BEMF vector | $\hat{\mathbf{e}}_{S(\alpha,\beta)}$ |
| curr | current vector | $\hat{\mathbf{e}}_{S(\alpha,\beta)}$ |
| mZTs | minus switching vector, time discrete | $-\mathbf{z}_{(\alpha,\beta)} \cdot T_s$ |
| k1Ts | switching gain, time discrete | $k_1 T_s$ |
| g1k1Ts | real gain, time discrete | $g_1 k_1 T_s$ |
| g2k1Ts | imaginary gain, time discrete | $g_1 k_1 T_s$ |

Other important parameters (not included in the smoBemfStruct):

**Table 5-2. Other Parameters**

| Element | Definition | Symbol |
|---|---|---|
| smoBemfCoefMultiple.g1k1Ts | BEMF vector | $(g_1 k_1 T_s)_{multiple}$ |
| smoBemfCoefMultiple.g2k1Ts | current vector | $(g_2 k_1 T_s)_{multiple}$ |

Some gain coefficients such as $g_1$, and $k_1$ were implemented as one combined coefficient $g_1k_1$. This implementation is suitable for the estimator quantities scaling and estimation process execution.

The adaptive speed scheme is implemented using the *PMSM_SL_AdaptSchIntDivAmpl32()* function. The Equation 2-69 is modified by division of the BEMF amplitude amplBEMF:

**Eqn. 5-25**

$$\omega_{el}(k+1) = \frac{g_{\omega} \bullet (Z_{(\alpha,\beta)}(k)^T \bullet J \bullet \hat{e}_{S(\alpha,\beta)}(k+1))}{\|\hat{e}_{S(\alpha,\beta)}(k+1)\|} T + \hat{\omega}_{el}(k)$$

This gives a better dynamic behavior and system stability over the whole speed range.

The speed scheme parameters are located in the structure *adaptSpeedSch* and the variables involved in the speed adaptive scheme are:

**Table 5-3. Adaptive Scheme Parameters**

| Element | Definition | Symbol |
|---|---|---|
| adaptSpeedSch.integGain | gain time discrete | $g_{\omega} = \text{integGain} \cdot 2^{\text{integGainScale}}$ |
| adaptSpeedSch.ntegGainScale | gain scale (see Section 5.2.4, "Scaling Parameters") | $g_{\omega} = \text{integGain} \cdot 2^{\text{integGainScale}}$ |
| speedMotorEstim | estimated speed | $\hat{\omega}_{el}$ |

Finally, the position information given by the sine and cosine is quantified from the estimated BEMF α,β representation *bemfAlphaBeta*. The Equation 2-63 and Equation 2-64 are used. For those calculations, it is necessary to have the signature of the speed. See Equation 2-63, and Equation 2-64. The ramp speed variable *speedMotorRampOpenLF32HL* is used for the sgn evaluation instead of the estimated speed *speedMotorEstim*. This is due to stability reasons. The SW parameters for the position calculation are then:

**Table 5-4. smoBemfStruct Parameters**

| Element | Definition | Symbol |
|---|---|---|
| bemfAlphaBeta.alpha | BEMF alpha | $e_{S\alpha}$ |
| bemfAlphaBeta.beta | BEMF beta | $e_{S\beta}$ |
| sinCosEst.cos | estimated sine | $\sin(\theta)$ |
| sinCosEst.sin | estimated cosine | $\cos(\theta)$ |
| speedMotorRampOpenLF32HL | speed at the ramp end | $\omega_{el}$ |
| amplBEMF | BEMF amplitude | $\|\hat{e}_s\|$ |

The position determined by the sine and cosine of the rotor angle is:

$$amplBEMF = \|\hat{e}_S\| = \sqrt{(bemfAlphaBeta \cdot alpha^2 + bemfAlphaBeta \cdot beta^2)} \qquad \textbf{Eqn. 5-26}$$

$$sinCosEst \cdot cos = \frac{bemfAlphaBeta \cdot alpha}{\sqrt{(bemfAlphaBeta \cdot alpha^2 + bemfAlphaBeta \cdot beta^2)}} \cdot sgn(speedMotorRampOpenLF32HL) \qquad \textbf{Eqn. 5-27}$$

$$sinCosEst \cdot sin = \frac{-bemfAlphaBeta \cdot beta}{\sqrt{(bemfAlphaBeta \cdot alpha^2 + bemfAlphaBeta \cdot beta^2)}} \cdot sgn(speedMotorRampOpenLF32HL) \qquad \textbf{Eqn. 5-28}$$

### 5.3.4.2 d, q Observer Implementation

The code version of the observer with d, q rotor related coordinates uses the *PMSM_SL_DQSMOBemfSpedObservSclUniv32()* function for the SMO feedback.

The function realizes the sliding mode observer according to Equation 2-84 with the system coefficients according to Equation 2-85 to Equation 2-90. The *EstimProcessing*() function uses the *smoBemfStruct* structure variable for state variables and coefficients. The estimated current $\hat{\mathbf{i}}_{(\alpha,\beta)}$ vector state variable is the component called *curr*. The estimated bemf $\hat{\mathbf{e}}_{(\alpha,\beta)}$ is the component called *bemfAlphaBeta*. These system variables are implemented as a 32-bit fractional representation. This 32-bit resolution is necessary to guarantee a good enough iteration precision at low amplitudes. The SMO feedback gain coefficients are part of the *smoBemfStruct* structure. The observer feedbacks are not constant, but the coefficients $g_1$, $g_2$ are multiplied by the BEMF amplitude. This is the same as Equation 5-23 and Equation 5-24.The adaptive speed scheme is implemented using the *PMSM_SL_IntegInlInpF32* function. It realizes the Euler time discrete iteration as Equation 2-97. The structure *smoBemfStruct* parameter markings compared to Chapter 2, “Control Theory” are:

**Table 5-5. smoBemfStruct Parameters**

| Element | Definition | Symbol |
|---|---|---|
| bemfAlphaBeta | BEMF α, β coordinate vector | $\hat{\mathbf{e}}_{S(\alpha,\beta)}$ |
| bemf | BEMF d, q coordinate vector | $\hat{\mathbf{e}}_{S(d,q)}$ |
| curr | current vector | $\hat{\mathbf{e}}_{S(\alpha,\beta)}$ |
| mZTs | minus switching vector, time discrete | $-\mathbf{z}_{(\alpha,\beta)} \cdot T_s$ |
| k1Ts | switching gain, time discrete | $k_1 T_s$ |
| g1k1Ts | real gain, time discrete | $g_1 k_1 T_s$ |
| g2k1Ts | imaginary gain, time discrete | $g_2 k_1 T_s$ |

Some gain coefficients such as $g_1$, and $k_1$ were implemented as one combined coefficient $g_1k_1$. This implementation is suitable for the estimator quantities scaling and estimation process execution.

The adaptive speed scheme parameters *adaptSpeedSch* are the same as with the α,β observer.

Finally, the position information given by the sine and cosine is quantified from the estimated BEMF α,β representation *bemfAlphaBeta*. This is provided the same way in Section 5.3.4.1, “a,b Observer Implementation”, the α,β observer with equations Equation 5-26, and Equation 5-28.

### 5.3.5 Auxiliary Position and Speed Sensing Using Encoder

The sensorless PM synchronous vector control application from the principle does not need an encoder. The position and speed are calculated by the sliding mode observer. However, for tuning, testing, demonstration, and debugging, it may be useful to use an encoder for position and speed sensing. Therefore, the code incorporates subroutines for encoder position and speed. By default SW setting, the speed and position from the encoder are not used for control but are used for FreeMASTER measurements. The compile of the encoder code can be suppressed by clearing the CODE_WITH_ENCODER preprocessor definitions.

The incremental encoder is mounted on the motor shaft. This generates two quadrature encoded signals (phases A and B). See Figure 5-7.

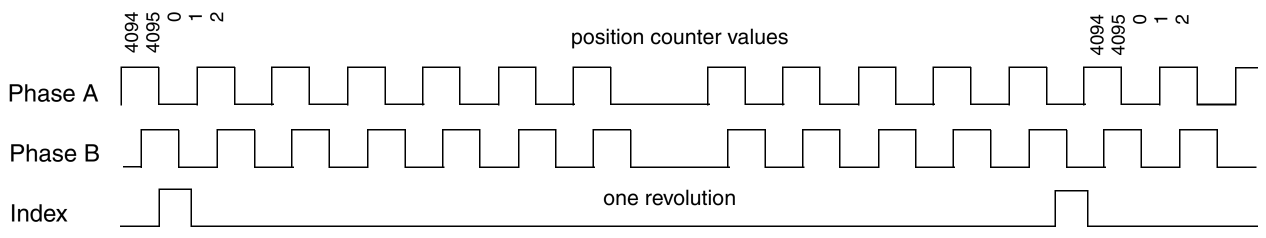


**Figure 5-7. Quadrature Encoder Signals**

The quadrature encoded signals from the position sensor are connected to the QuadTimer module input pins (pin 0 and pin 1). The QuadTimer channel 0 is configured to decode these encoded signals. The timer counter increments/decrements to provide a position information on the incremental sensor. At the same time, the QuadTimer channel 0 is configured to capture a counter value on both edges on secondary input pin 1. The captured counter value corresponds to the rotor position.

For speed calculation, an additional QuadTimer channel is required to generate a time-base reference. QuadTimer channel 1 is configured to generate a 1 ms time-base reference. Similar to channel 0, channel 1 is configured to capture a counter value on both edges on secondary input pin 1. The captured counter value corresponds to the exact time period of the captured rotor positions. Configuration of the QuadTimer channels are shown in Figure 5-8.

QuadTimer channel 1 is configured to perform a compare with a 1 ms period. It counts IP bus clock pulses on the primary source with prescaler set to 2. The counter is configured to count roll-over. On every compare event, an interrupt is generated and an interrupt service routine is called. The interrupt service routine calls a function to evaluate the motor speed and services the timer channel for the next compare interrupt.

There are two common ways to measure speed. The first method measures the time between two following edges of the quadrature encoder and used at low speed. The second method measures the position difference per constant period. At higher speeds when the measured period is short, the speed calculation algorithm switches to the second method.

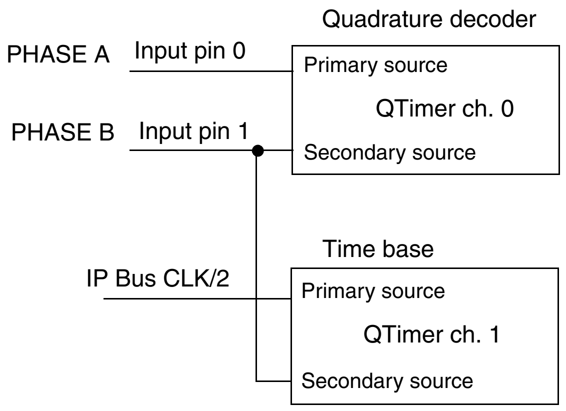


**Figure 5-8. QuadTimer Channels Configuration**

The proposed algorithm combines both of the mentioned methods. The algorithm simultaneously measures the number of quadrature encoder pulses per constant period and their accurate time period. The speed can then be expressed as:

$$speed = \frac{k_1 \cdot N}{T} = \frac{k_1 \cdot N}{T_{elkT2} N_{elkT2}} = \frac{k \cdot N}{N_{elkT2}} \qquad \textbf{\textit{Eqn. 5-29}}$$

Where:

| | | |
|---|---|---|
| *speed* | calculated speed | [–] |
| $k$ | scaling constant | [–] |
| $k_1$ | scaling constant | [s] |
| $N$ | number of counted pulses per constant period | [–] |
| $T$ | accurate period of *N* pulses | [s] |
| $T_{clkT2}$ | period of input clock to time-base timer (TMR1) | [s] |
| $N_{clkT2}$ | number of pulses counted by quadrature timer (TMR0) | [–] |

The time base is provided by QuadTimer channel 1. This is set to call a slow control loop every 1 ms where the speed measurement is calculated. To evaluate motor speed, the function *ENC_GetMotorSpeedEl(&enc)* is called in the 1 ms interrupt service routine. The speed processing algorithm works as follows:

1. The newly captured values of both timers are read from the capture registers. The difference in the number of pulses (TMR0) and their accurate period (TMR1) are calculated from the actual and previous values.
2. The new values are saved for the next period and the capture register is enabled. From this time, the first edge on input pin 1 signals the capture of values of both timers (TMR0 and TMR1) and the capture register is disabled.
3. The speed is calculated according to Equation 5-29.
4. This process is repeated with each call of the speed processing algorithm. See Figure 5-9.

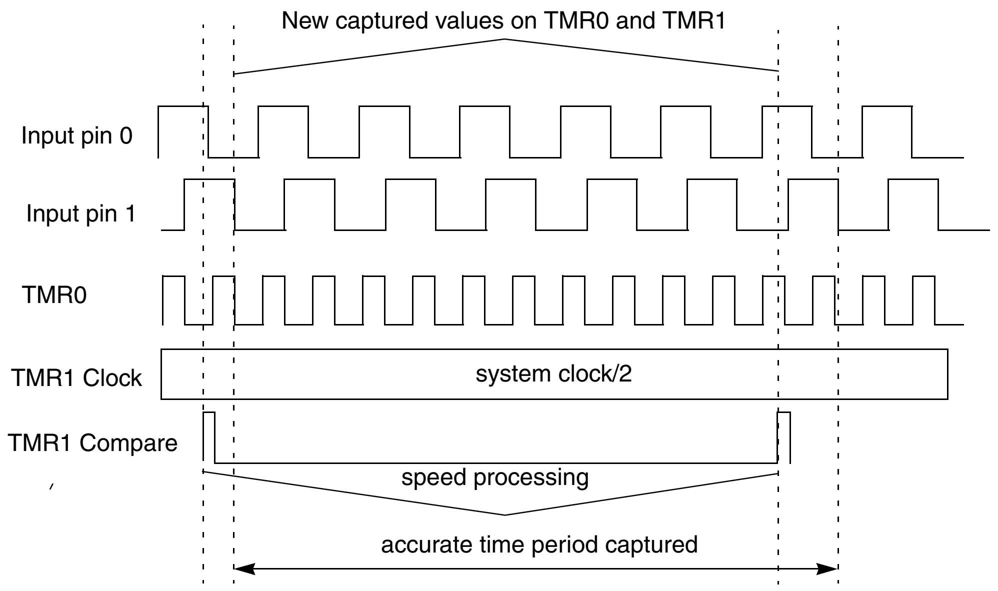


**Figure 5-9. Speed Processing**

## 5.4 Software Implementation

The general software diagram incorporates the main routine (Main) entered from a reset and the interrupt states. See Figure 5-1.

This main routine initializes the DSC and the application then enters an infinite background loop. The background loop contains an application state machine.

### 5.4.1 SW States

#### 5.4.1.1 Application Process States

The application process state is the highest level of the application state-machine. It has three application states: APP_INIT, APP_STOP, APP_RUN, and APP_FAULT. Transition between the states is shown in Figure 5-10.

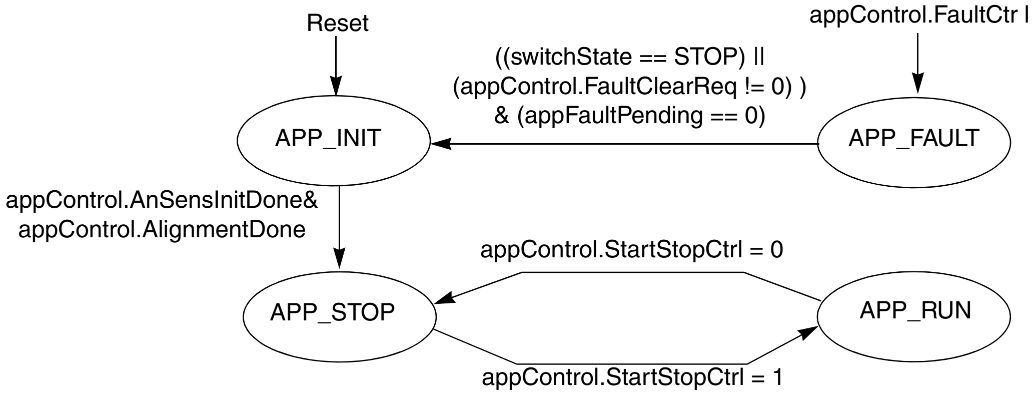

Reset
appControl.FaultCtr l
((switchState == STOP) ||
(appControl.FaultClearReq != 0) )
& (appFaultPending == 0)
APP_INIT
APP_FAULT
appControl.AnSensInitDone&
appControl.AlignmentDone
appControl.StartStopCtrl = 0
APP_STOP
APP_RUN
appControl.StartStopCtrl = 1

**Figure 5-10. Application State Diagram — General Overview**

Each state has dedicated transients of the subprocess PMSM control. These are displayed in Section 5.4.1.2, “Application States Details”.

The PMSM control states are as follows:

- Calibration ADC

  ADC calibration state is the state where the PWM module is switched off. There must be no current flowing through the motor. The ADC running measurement is to provide zero calibration. At the end of the state the ADC offsets are set to the average of the measured values.
- PMSM alignment

  A current vector with a constant speed and current is applied to the motor. This is used for the encoder position reset. The PMSM control state ALIGNMENT is used only when the encoder code compilation is defined (CODE_WITH_ENCODER). The following figures display the case with the encoder.
- PMSM stop mode

  In the stop mode, the PWM module is switched off. There is no current flowing through the motor.
- PMSM starting mode
  - — Stabilization
  - — Ramp no estimation
  - — Ramp estimation

The PMSM starting mode has three substates. Stabilization provides a vector with constant current amplitude and an almost constant angle. The rotor is stabilized for a defined period. After the ramp, no estimation is executed. This provides the open-loop start up although the sliding mode observer is not running. After the starting ramp reaches a defined threshold, the ramp estimation mode is entered. This remains the open-loop start-up, but the estimator is already running and estimation starts approaching real quantities.

- PMSM run mode

In the PMSM run mode, the application is running a closed-loop with a feedback from the SMO. If the estimated speed goes below a defined threshold, the run mode remains with PMSM starting mode.

In the following section the transients between states are provided according to the variables:

| | |
|---|---|
| `appControl.FaultCtrl` | `any fault detected flag` |
| `appControl.FaultClearReq` | `fault clear request` |
| `appControl.AlignmentDone` | `application Alignment Done flag` |
| `appControl.AnSensInitDone` | `application current Sensing Initialization Done flag` |
| `appControl.StartStopCtrl` | `start stop flag` |
| `appFaultPending` | `application faults pending` |
| `speedMotorAct` | `actual motor speed` |
| `speedMotorSMOonThreshold` | `speed threshold to start observer` |
| `speedMotorSpEstOnThreshold` | `speed threshold to start adaptive speed estimation\` |
| `speedMotorRampOpenLF32HL` | `open-loop ramp speed` |
| `speedMotorThresholdStartMode` | `speed threshold to return to starting mode` |
| `speedMotorThresholdSless` | `speed threshold to start the regular PMSM_RUM_MODE with a position and speed feedback from SMO` |
| `startStateDurationCntr` | `stabilisation state duration counter` |
| `startStateDurationThreshold` | `stabilisation state duration threshold - determines the state length` |
| `switchState` | `mechanical switch state` |

### 5.4.1.2 Application States Details

The application process states. The substates are displayed in the following figures.

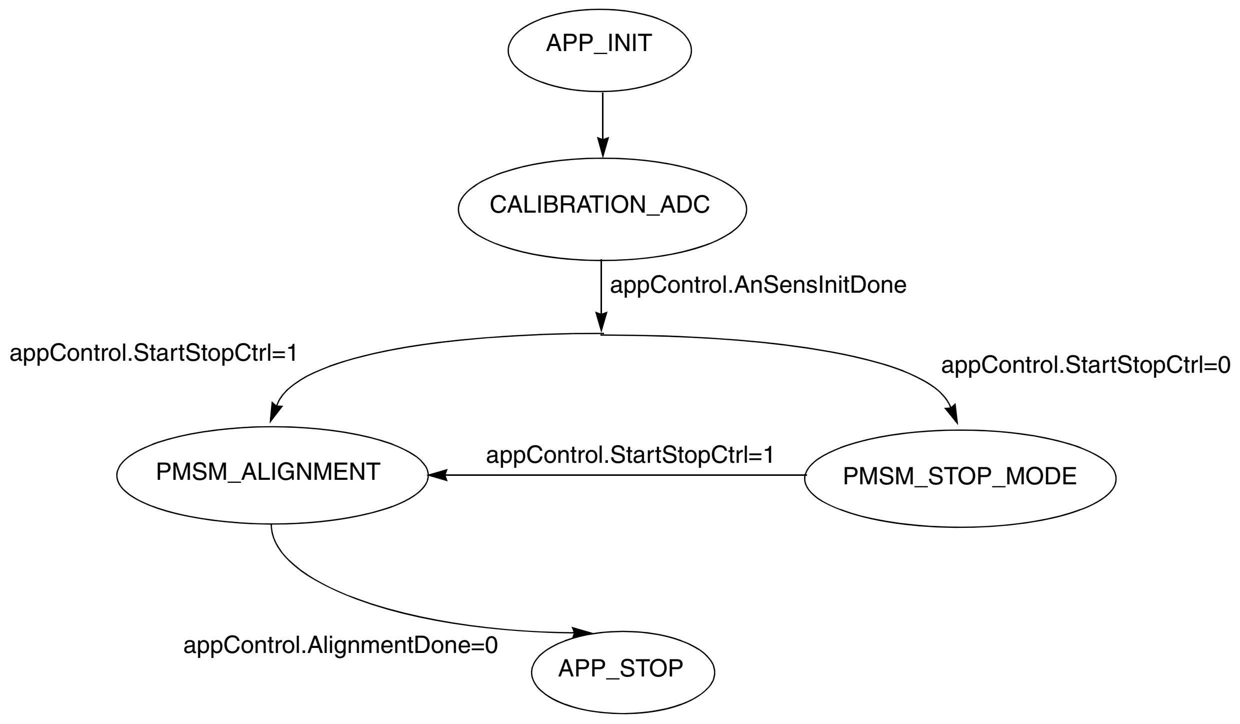

APP_INIT
CALIBRATION_ADC
appControl.AnSensInitDone
appControl.StartStopCtrl=1
appControl.StartStopCtrl=0
PMSM_ALIGNMENT
appControl.StartStopCtrl=1
PMSM_STOP_MODE
appControl.AlignmentDone=0
APP_STOP

**Figure 5-11. APP_INIT with PMSM Control Substates**

Application initial state starts with ADC calibration/initialization. Then (code with encoder) the current alignment vector is provided. Finally, the APP_INIT state goes into APP_STOP.

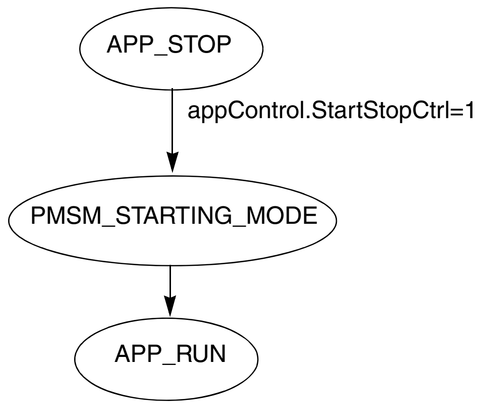


**Figure 5-12. APP_STOP State with PMSM Control Substates**

Application APP_STOP state waits for the start control and then initiates the APP_RUN state with the PMSM_STARTING_MODE substate.

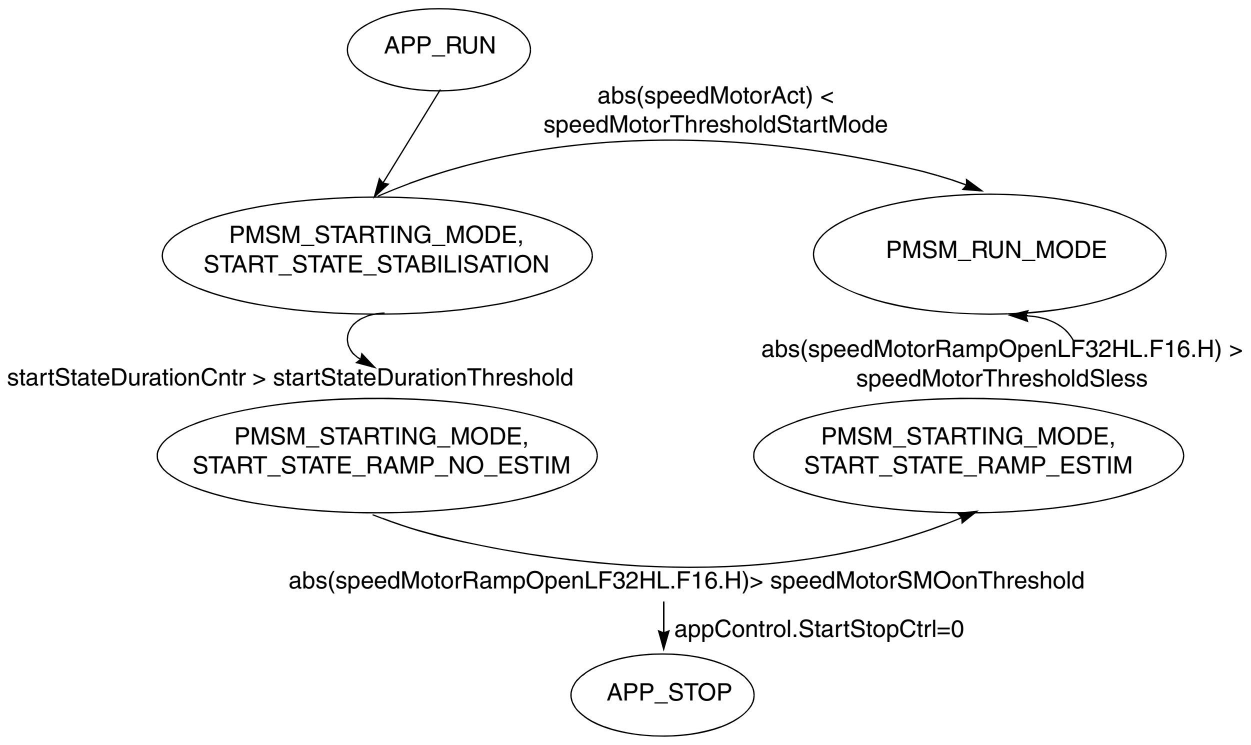

APP_RUN
abs(speedMotorAct) < speedMotorThresholdStartMode
PMSM_STARTING_MODE, START_STATE_STABILISATION
PMSM_RUN_MODE
startStateDurationCntr > startStateDurationThreshold
abs(speedMotorRampOpenLF32HL.F16.H) > speedMotorThresholdSless
PMSM_STARTING_MODE, START_STATE_RAMP_NO_ESTIM
PMSM_STARTING_MODE, START_STATE_RAMP_ESTIM
abs(speedMotorRampOpenLF32HL.F16.H)> speedMotorSMOonThreshold
appControl.StartStopCtrl=0
APP_STOP

**Figure 5-13. APP_RUN state with PMSM Control Substates**

The application APP_RUN state is entered to rotate the motor. Because rotation starts the open-loop, there are stabilization and starting modes for the PMSM control subprocess. The stabilization state remains after a defined period. The starting mode is without estimation. The SMO is not calculated due to low BEMF at low rotor speeds. If the open-loop ramp speed reaches the speedMotorSMOOnThreshold, the open-loop start-up runs and the SMO is calculated. After the ramp exceeds the speedMotorSpEstOnThreshold, the speed estimation starts to be calculated. After speedMotorThresholdSless, the regular run mode with feedback from the SMO is entered. If the estimated speed goes below the *speedMotorThresholdStartMode*, the speed is too low for the observer estimation and the PMM control process goes back to the starting mode.

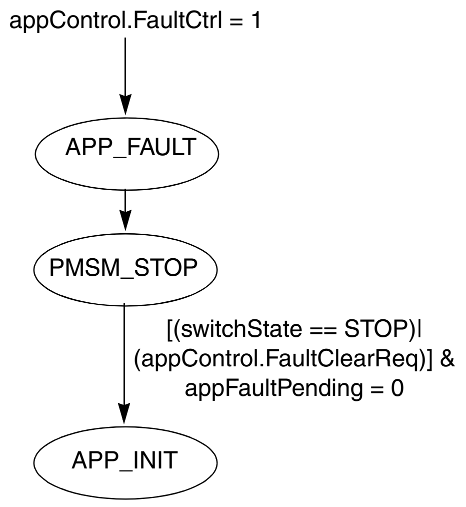


**Figure 5-14. APP_FAULT with PMSM Control Substates**

If any fault is detected, the appControl.FaultCtrl is set in the background loop. It enters the APP_FAULT state. The fault state can be exited with a mechanical switch set to stop or appControl.FaultClearReq by FreeMASTER if there is no fault pending.

### 5.4.2 Initialization

Initialization is entered after a reset. When the application main is entered, a low-level initialization is called. According to the peripheral and CPU, the DSC registers are initialized. This is the first function that has to be called in the application main. Next, the FreeMASTER embedded driver is initialized according to settings in the freemaster_cfg.h configuration file. Finally, the application initialization function *ApplicationResetInit()* is called. Tasks performed in *ApplicationResetInit()* are as follows:

- Application init state with the PMSM stop state set
- The application registers are set according to HW parameters from the *PMSM_ClosedLoop.h*
- Scaling range variables are initialized for the FreeMASTER control page
- The application registers are set according to motor parameters from the *PMSM_ClosedLoop.h*
  - Data structures of the SMO are initialized
  - Data structures of all PI controllers are initialized (proportional and integral gains, and output limits)
  - Current decoupling structures are initialized
  - Speed ramp acceleration/deceleration variables are initialized
- DSP interrupt and clock modules (OCCS, SIM, INTC) are set
- FreeMASTER with the SCI0 module is initialized
- QuadTimer channel 3 is initialized to perform PWM module and ADC synchronization if MEASURE_CODE_EXE_TIME is defined and the QTIMER_2 for the code execution measurement is initialized

- Encoder processing data structure (`enc`) is initialized
- Encoder initialization function is called with the initialized enc structure *ENC_Init(&enc)* this function configures QuadTimer channels 0 and 1 for quadrature encoder processing.
- Analogue sensing is initialized starting with the calibration state. ADC converter offset registers are initialized.
- Arithmetic saturation is set to off in the background code for the FreeMASTER, however the interrupt functions need saturation and is set at the beginning of the dedicated ISR.
- Finally, interrupts are enabled
- The watchdog is disabled and the application enters an endless background loop.

### 5.4.3 Application Background Loop

The endless application background loop executes a simple application state-machine, shown in Figure 5-10. Other process functions called from the loop are:

- FreeMASTER polling function *FMSTR_Poll()*
- Brake control for maintaining DC-bus over-voltage switch *BrakeControl()*
- Background part of the fault detection with *FaultDetection()*

The main application control tasks are executed in interrupt service routines that interrupt the background loop.

### 5.4.4 Interrupts

There are two periodic interrupt service routines executing the major motor control tasks, the PWM reload interrupt called after reset to set the correct ADC sampling and fault interrupt in the application. Control tasks are split into fast and slow control loops. Each loop has a corresponding priority. The interrupt service routines and control tasks executed by each interrupt are described in the following subsections.

#### 5.4.4.1 ADC End Of Scan Interrupt

The function *ADCCompleteISR()* is assigned to this interrupt event. The interrupt is configured to be executed with a priority level 2. It is executed in synchronization with the PWM cycle. Synchronization with the PWM reload signal is triggered by a TMR3 output. A more detailed description of the ADC End Of Scan interrupt timing can be found in Section 5.3.2, “ADC End of Scan Timing and PWM Reload Interrupts”.

The function executes a fast control loop. The most time-critical tasks of the vector control algorithm are performed here, including the current control loop and sliding mode observer. According to the PMSM control state (variable *pmsmControlState*), the *ADCCompleteISR()* performs the following tasks:

- Analogue signals sensing:
  - — Read the sampled phase currents voltage from the ADC
  - — Read the sampled DC-bus voltage from the ADC and execute the digital filter on that sample
- Forward Clark transformation from phase-currents (a, b, c) into the two-phase stationary reference frame $\alpha,\beta$ coordinates

- Forward Park transformation of the stator phase currents from stator relative α,β coordinates (*iSAlphaBetaCompens*) into the rotational d, q reference frame (`iSdq`)
- Execute the PI controllers for the d and q-axis components of the stator current
- Perform a transformation of the stator voltage space-vector d, q components from the rotational reference frame into the α,β stationary reference frame
- Perform the DC-bus ripple elimination algorithm on the output voltage
- Perform space vector modulation on the output voltage
- Program the PWM value registers and set the LDOK bit
- Configure the ADC channels for the next phase current sampling
- In the case of a starting stage, it preforms the required current vector rotation
- If encoder code compilation is defined (CODE_WITH_ENCODER), the encoder position sine and cosine are calculated
- The PMSM control state is evaluated and possible transitions made, for example starting/running
- FreeMASTER recorder is executed. The sampling must be set according to the *ADCCompleteISR* call period.
- Service the ADC complete interrupt flag

#### 5.4.4.2 PWM Reload Interrupt

The function *PWM_ReloadISR()* is assigned to this interrupt event. The priority of the interrupt is set to level 2. The *PWMReloadISR()* provides trick/initialization for the half cycle reload setting and ADC sampling. The PWMReloadISR is called only twice after application reset. At application reset, the PWM half cycle reload and the reload opportunity for each PWM must be set, then pwmReloadCounter = 2. The second call of the PWMReloadISR is executed during a half cycle reload and the PWM_RELOAD_FREQUENCY is changed to an odd number PWM_RELOAD_OPPORTUNITY. This gives the functionality that all PWM reloads and ADC sampling start at a PWM half cycle reload. According to the PWM_RELOAD_OPPORTUNITY preprocessor constant, each half reload and each second half reload is set.

#### 5.4.4.3 QuadTimer Channel 1 Compare Interrupt

The function *SlowControlLoopISR()* is assigned to this interrupt event. The interrupt is executed periodically with a 1 ms period. The priority of the interrupt is set to level 1.

The *SlowControlLoopISR()* function executes a slow control loop. The least time-critical tasks of the vector control algorithm are performed here including the speed control loop.

Tasks performed by the *SlowControlLoopISR()* function:

- If encoder code compilation is defined (CODE_WITH_ENCODER) execute the *ENC_GetMotorSpeedEl(&enc)* function to evaluate the actual rotor speed
- Perform ramping of the motor speed. The ramp depends on the PMSM_Starting or PMSM_Run mode
- Execute the motor speed PI controller. Output of the speed controller sets the required torque-producing component of the stator current (q-axis).
- Service the corresponding interrupt request flag

#### 5.4.4.4 PWM Fault Interrupt

The function *PWM_FaultISR()* is assigned to this interrupt event. The interrupt is executed on an event. When a DC-bus over-current is detected an external comparator sets a signal on the PA6/FAULT0 pin to a high level. The PWM module sets all the PWM signals to the off state through wired logic. An interrupt request is then generated and the priority of the interrupt is set to level 2.

Tasks performed by the *PWM_FaultISR()* function:

- Disable the PWM output pads
- Turn off the motor
- Set the application *appFaultStatus.OverVoltageFlag* and *appFaultPending. OverVoltageFlag* or *appFaultStatus.OverCurrentFlag* and *appFaultPending.OverCurrentFlag* flags
- Clear dedicated DSC modules faults

### 5.4.5 PI Controller Parameters

The PI controller parameters consist of the gain and gain scale parameters of the proportional and integral constants. The proportional or integral gain parameter lies in the fractional number 0 to 1, (representing 0 to 32767). If the gain scale is positive the parameter shifts the particular gain to the right or to the left if negative. The gain scale number represents the number of shifts.

**NOTE**

The scale parameters of the controllers (proportional, integral gain) are negative compared to others such as the SMO scale parameters described in Section 5.2.4, “Scaling Parameters”. This is due to reuse of the controller libraries with a different scale philosophy. Therefore, postfix _NEG is used in the definition, MOTOR_RPM_REG_PGAIN_SCALE_NEG.

The limit parameters represent the minimum and maximum outputs from the PI controller. The output is within these limits.

## 5.5 FreeMASTER Software

The FreeMASTER software is designed to provide a debugging, diagnostic, and demonstration tool for the development of algorithms and applications. It is also useful for tuning the application for different power stages and motors. Almost all the application parameters can be changed via the FreeMASTER interface. This consists of a component running on a PC and another part running on the target DSC connected via an RS-232 serial port. A small program is resident in the DSC that communicates with the FreeMASTER software to parse commands, return status information to the PC, and processes control information from the PC. FreeMASTER software executed on the PC uses Microsoft Internet Explorer as the user interface.

### 5.5.1 FreeMASTER Serial Communication Driver

This application includes the FreeMASTER Serial Communication Driver. The FreeMASTER serial communication driver fully replaces the former PC Master driver. The new FreeMASTER driver remains fully compatible with the communication interface provided by the old PC Master drivers bringing many useful enhancements and optimizations.

The primary advantage of the new driver is unification across all supported Freescale processor products and several new features that were added. One of the key features implemented in the new driver is target-side addressing (TSA) that enables an embedded application to describe the memory objects it grants the host access to. By enabling the TSA-Safe option, the application memory can be protected from illegal or invalid memory accesses.

To include the new FreeMASTER serial communication driver in the application the user has to manually include the driver files in the CodeWarrior project. For this application, the driver has already been included.

The FreeMASTER driver files are located in following folders:

- {Project}support\freemaster\56F8xxx — contains platform-dependent driver C-source and header files including a master header file freemaster.h.
- {Project}support\freemaster\common — contains common driver source files shared by the driver for all supported platforms.

All C files included in the FreeMASTER folders are added to the project for compilation and linking. See the support group in the project. The master header file freemaster.h declares the common data types, macros, and prototypes of the FreeMASTER driver API functions. This must be included in the application by using #include directive wherever calling any of the FreeMASTER driver API functions is needed.

The FreeMASTER driver does not perform any initialization or configuration of the SCI module it uses to communicate. It is the user’s responsibility to configure the communication module before the FreeMASTER driver is initialized by the FMSTR_Init() call. The default baud rate of the SCI communication is set to 9600 Bd.

**NOTE**

Higher communication speeds than 9600 Bd is not supported by the MC56F8013/23 controller board due to the limited speed of opto-couplers.

FreeMASTER uses a poll-driven communication mode. It does not require the setting of interrupts for the SCI. Both communication and protocol decoding are d in the application background loop. The polling-mode requires a periodic call of the `FMSTR_Poll()` function in the application main.

The driver is configured using the appconfig.h header file. Changes to the file are preferably made through the provided quick start graphical configuration tool in CodeWarrior toolbar project/configuration tool. The user has to modify this file to configure the FreeMASTER driver. The FreeMASTER driver C-source files include the configuration file and use of the macros defined there for conditional and parameter compilation.

A detailed description of the FreeMASTER serial communication driver is provided in the *FreeMASTER Serial Communication Driver User's Manual.*

## 5.5.2 FreeMASTER Recorder

Part of the FreeMASTER software is a recorder able to sample the application variables at a specified sample rate. The samples are stored in a buffer and read by the PC via an RS232 serial port. The sampled data can be displayed in a graph or the data can be stored. The recorder behaves as a simple on-chip oscilloscope with trigger / pre-trigger capabilities. The size of the recorder buffer and the FreeMASTER recorder time base can be defined in the appconfig.h configuration.

The recorder routine must be called periodically from the loop to take samples. The following line must be added to the loop code:

```
FMSTR_Recorder(); /* FreeMASTER recorder routine call */
```

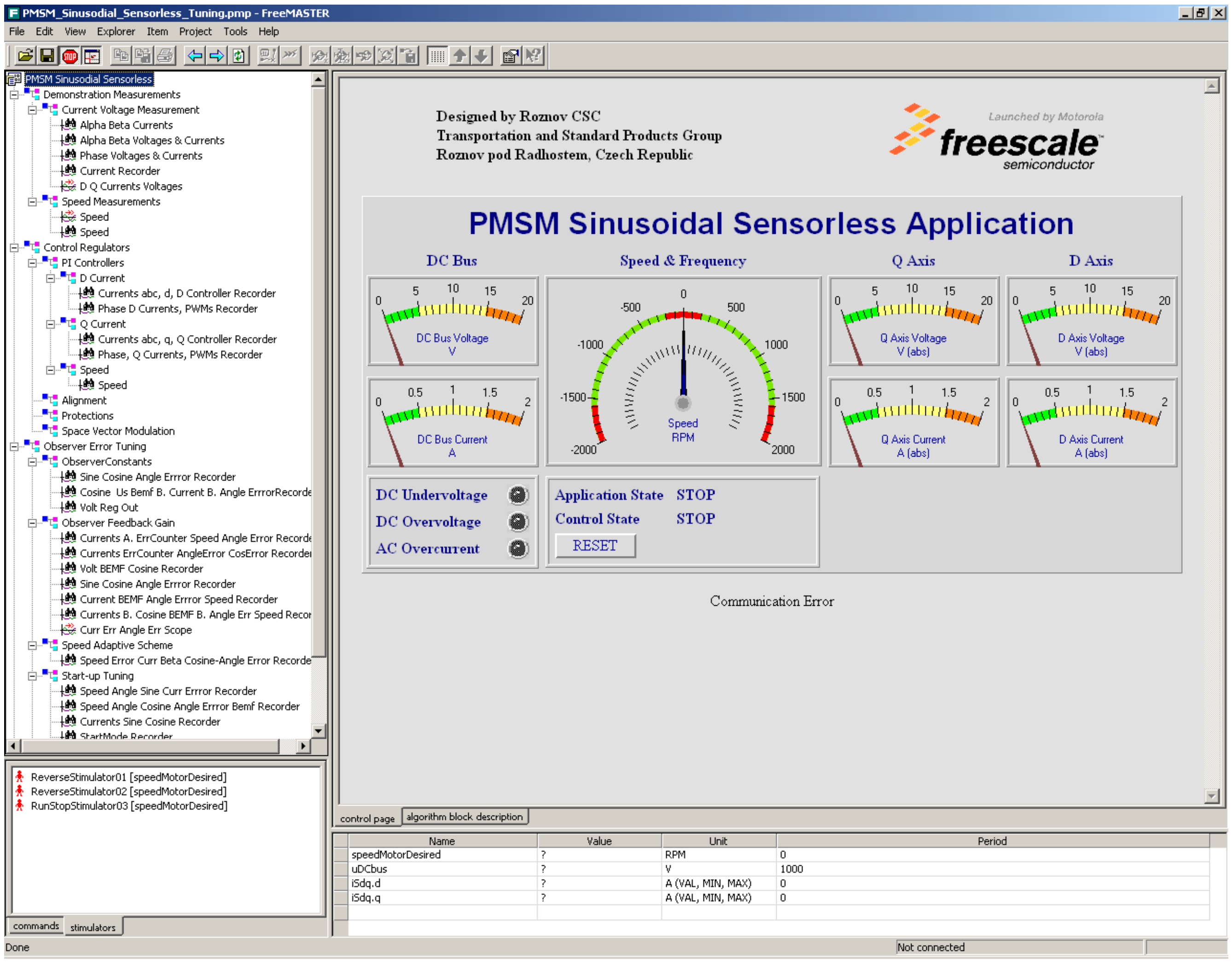
In this application, the FreeMaster recorder is called from the ADC Complete EOS interrupt that creates a 125 μs time-base for the recorder function. A detailed description of the Free Master software is provided in the *FreeMASTER Software User Manual.*

## 5.5.3 FreeMASTER Control Page

The FreeMASTER control page creates a graphical user interface (GUI) for the sensorless PMSM vector control application. Start the FreeMASTER software window’s project by clicking on the PMSM_Sinusodial_Sensorless_Tuning.pmp file. Figure 5-15 illustrates the FreeMASTER software control window after this project has been launched. To switch to the control page, click on the control page tag.

The user is able to monitor all the important qualities of the motor. By clicking the speed gauge, the motor is started and the desired speed is set. The actual motor speed, motor currents, and voltages are displayed on the control page gauges.

The application status is displayed. A status fault LED indicates the occurrence of an application fault.


PMSM_Sinusoidal_Sensorless_Tuning.pmp - FreeMASTER
File Edit View Explorer Item Project Tools Help
PMSM Sinusodial Sensorless
Demonstration Measurements
Current Voltage Measurement
Alpha Beta Currents
Alpha Beta Voltages & Currents
Phase Voltages & Currents
Current Recorder
D Q Currents Voltages
Speed Measurements
Speed
Speed
Control Regulators
PI Controllers
D Current
Currents abc, d, D Controller Recorder
Phase D Currents, PWMs Recorder
Q Current
Currents abc, q, Q Controller Recorder
Phase, Q Currents, PWMs Recorder
Speed
Speed
Alignment
Protections
Space Vector Modulation
Observer Error Tuning
ObserverConstants
Sine Cosine Angle Errror Recorder
Cosine Us Bemf B. Current B. Angle ErrrorRecorde
Volt Reg Out
Observer Feedback Gain
Currents A. ErrCounter Speed Angle Error Recorde
Currents ErrCounter AngleError CosError Recorder
Volt BEMF Cosine Recorder
Sine Cosine Angle Errror Recorder
Current BEMF Angle Errror Speed Recorder
Currents B. Cosine BEMF B. Angle Err Speed Recor
Curr Err Angle Err Scope
Speed Adaptive Scheme
Speed Error Curr Beta Cosine-Angle Error Recorde
Start-up Tuning
Speed Angle Sine Curr Errror Recorder
Speed Angle Cosine Angle Errror Bemf Recorder
Currents Sine Cosine Recorder
StartMode Recorder
ReverseStimulator01 [speedMotorDesired]
ReverseStimulator02 [speedMotorDesired]
RunStopStimulator03 [speedMotorDesired]
commands
stimulators
Designed by Roznov CSC
Transportation and Standard Products Group
Roznov pod Radhostem, Czech Republic
Launched by Motorola
freescale semiconductor
PMSM Sinusoidal Sensorless Application
DC Bus
Speed & Frequency
Q Axis
D Axis
DC Bus Voltage V
DC Bus Current A
Speed RPM
Q Axis Voltage V (abs)
Q Axis Current A (abs)
D Axis Voltage V (abs)
D Axis Current A (abs)
DC Undervoltage
DC Overvoltage
AC Overcurrent
Application State STOP
Control State STOP
RESET
Communication Error
control page
algorithm block description
Done
Not connected

| Name | Value | Unit | Period |
|---|---|---|---|
| speedMotorDesired | ? | RPM | 0 |
| uDCbus | ? | V | 1000 |
| iSdq.d | ? | A (VAL, MIN, MAX) | 0 |
| iSdq.q | ? | A (VAL, MIN, MAX) | 0 |

**Figure 5-15. FreeMASTER Control Screen**

The FreeMASTER software control page. These are actions that are supported:

- Setting the required speed of the motor
- Reset the application fault status

The FreeMASTER software control page displays:

- DC Bus current and voltage
- d, q axis currents and voltages
- Application fault status

For SW tuning and demonstration, the following sub-blocks are prepared:

- Demonstration measurements
- Control regulators — for adjusting PI controller parameters
- Observer tuning
  - Adjusting the SMO parameters
  - Adjusting the SMO feedback
  - Adjusting start-up parameters
- Other tuning — code execution measurement, DC bus brake

## 5.6 SW Parameters Setting to a Specific Motor

The default SW parameter settings have been tuned for a default HW set-up with the motor TGT3-0065-30-320. The set-up is described in Chapter 6, “Application Setup”. If other motor and HW is used, the SW setting needs to be changed according to their specific parameters. The detailed parameters settings and measurements are not described in this document. There is a possibility to use dedicated xls files that are included within each SW version. These files can calculate the constants for the PMSM_ClosedLoop.h according to the motor and HW parameters. The xls files that are included within the code directories:

- The Sliding Mode Observer with the α,β coordinate model uses the file SWSettingPMSM_SensorlessCL_AlpBetHRe.xls included within PMSM_SensorlessCL_AlpBetNDHRel directory.
- The Sliding Mode Observer with the d, q coordinate model uses the file SWSettingPMSM_SensorlessCL_DQHRe.xls included within PMSM_SensorlessCL_DQHRel directory.

A fine tuning might then be provided using the FreeMASTER tool.

# Chapter 6
Application Setup

As described earlier, the sensorless PM synchronous motor vector control application is targeted at the MC56F8013 device. The concept of the sensorless PM synchronous motor vector control drive incorporates the following hardware components:

- The MC56F8013/23 Controller Board and possibly MC56F8346 Controller Board.
- A 3ph AC/BLDC high voltage power stage board.
- In-line optoisolation board.
- Three-Phase PM Synchronous Motor, default configuration for motor TG drives TGT3-0065-30-320.

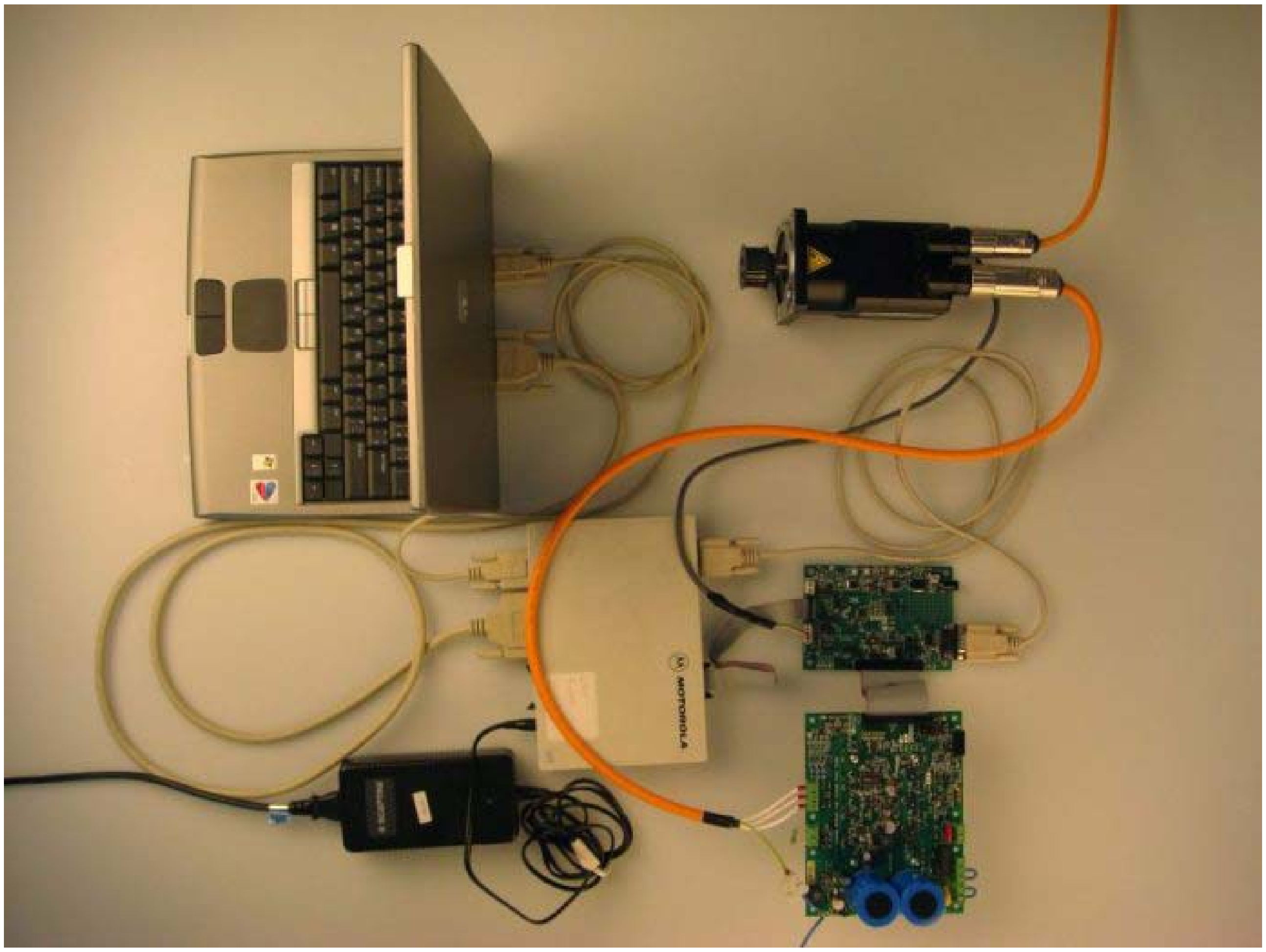


**Figure 6-1. Demo Application Setup**

## 6.1 MC56F8013 and MC56F8023 Controller Board Setup

Prior to the MC56F8013/23 controller board being connected to the power-stage, it needs to be configured for the correct operation. The demo application code also has to be programmed first into the Flash memory. For configuring the MC56F8013/23 controller board follow these steps:

Set the jumper configuration on the MC56F8013/23 controller board as shown in the Table 6-1:

1. Connect a +12 V power supply to the J12 power connector on the MC56F8013/23 controller board.
2. Set the over-current/over-voltage thresholds.

   Trimpot R29 sets the over-voltage threshold. Use a voltmeter to measure the threshold level at test-point TP3. Turn the R29 trimpot to set the threshold level. The voltage level on TP3 must be >3.2 V.

   Trimpot R32 sets the over-current threshold. Use a voltmeter to measure the threshold level at test-point TP5. Turn the R32 trimpot to set the threshold level. The voltage level on TP3 must be >3.2V.
3. Connect the parallel JTAG command converter or the USB-TAP to the host PC and to the J4 header on the MC56F8013/23 controller board.
4. Compile your project and program it into the device.
5. Disconnect the +12 V power supply from the J12 power connector on the MC56F8013/23 controller board.
6. Unplug the JTAG command converter or the USB-TAP from the J4 header on the MC56F8013/23 controller board.

**Table 6-1. MC56F8013/23 Controller Board Jumper Setting**

| Jumper | Setting | Description |
|---|---|---|
| JP1 | 2–3 | SCI Bi-wire configuration |
| JP4 | 1–2, 4–5, 7–8 | ADC CFG2 configuration (motor phase-current sensing) |
| JP5 | 1–2, 4–5, 7–8 | ADC CFG1 configuration (motor phase-current sensing) |
| JP3 | 4–5, 7–8 | Incremental encoder configuration |
| J16 | 1–2 | +5 V power supply from UNI-3 connector |
| JP18 | 1–2 | +15 V power supply from UNI-3 connector |

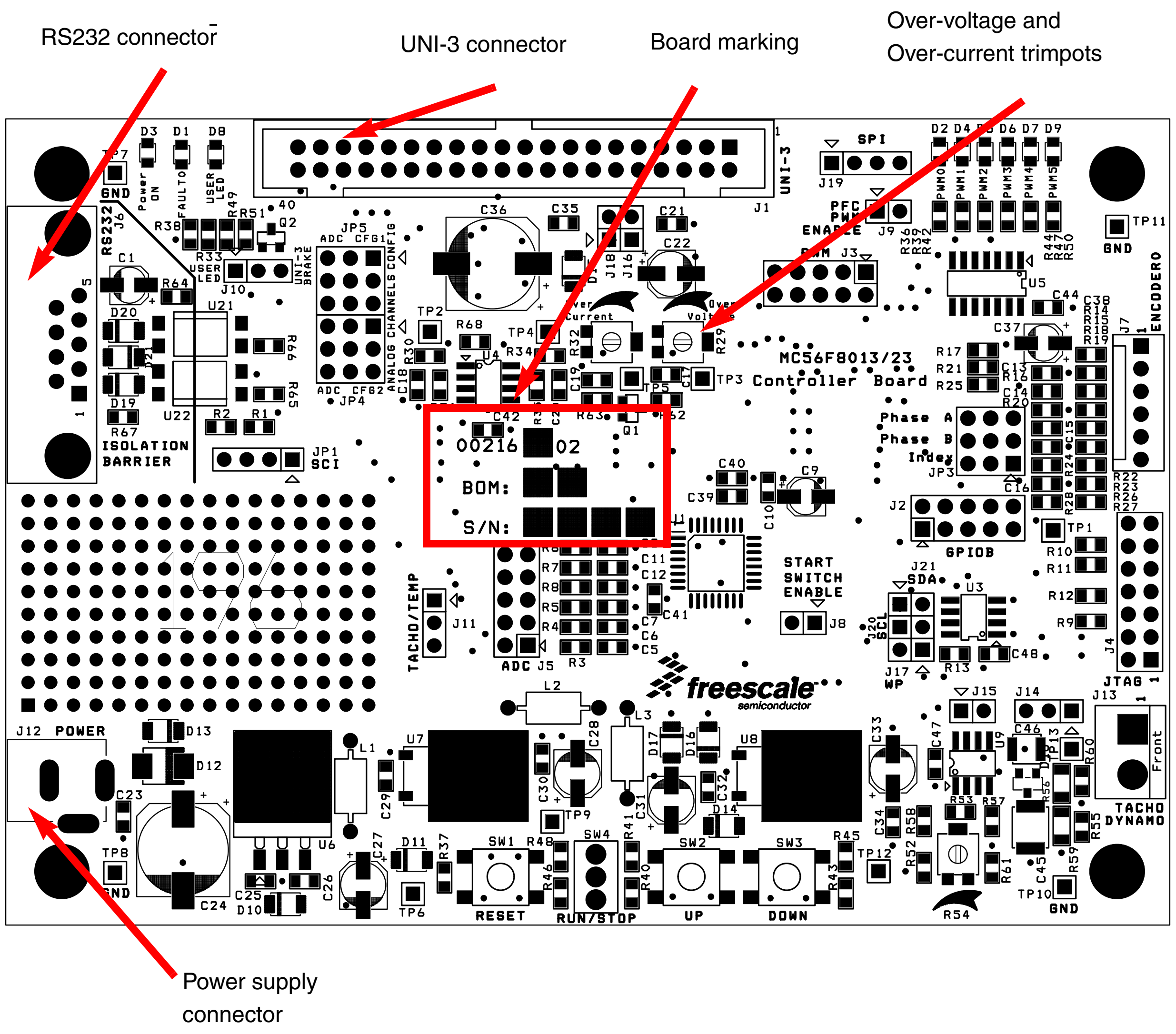


**Figure 6-2. MC56F8013/23 Controller Board View**

## 6.2 MC56F8346 Controller Board Setup

Another possibility is using the MC56F8346 device with a dedicated SW version. The applications are almost the same. The difference is using the SW for the MC56F8346 (differs with peripherals utilization) and a different 56F8346 controller board. The MC56F8346 controller board setting is described in a manual for the board.

## 6.3 Demo Hardware Setup

When the MC56F8013/23 controller board is configured it can be connected to the power stage and the whole demo hardware set-up can be built. The complete application set-up, is shown in Figure 6-1 and Figure 6-2. To build the demo application set-up, follow these steps:

1. Connect the UNI-3 connector (J1) on the MC56F8013/23 controller board to the UNI-3 counterpart connector on the 3 ph AC/BLDC high voltage power stage board via a 40 pin ribbon cable.
2. Connect a serial cable to an open COM port on the host PC and to the J6 DB-9 connector on the MC56F8013/23 controller board for the FreeMASTER remote control.
3. Connect an incremental encoder cable to the J7 connector on the MC56F8013/23 controller board.
4. Connect the motor phases to terminals J6 on the 3 ph AC/BLDC high voltage power stage board.
5. Connect a 115/230V AC power supply or a 100 – 320 V DC power supply to terminals J6 on the 3 ph AC/BLDC high voltage power stage board.
6. Switch-on the high-voltage power supply.
7. Start the FreeMASTER project and switch to the control page panel.

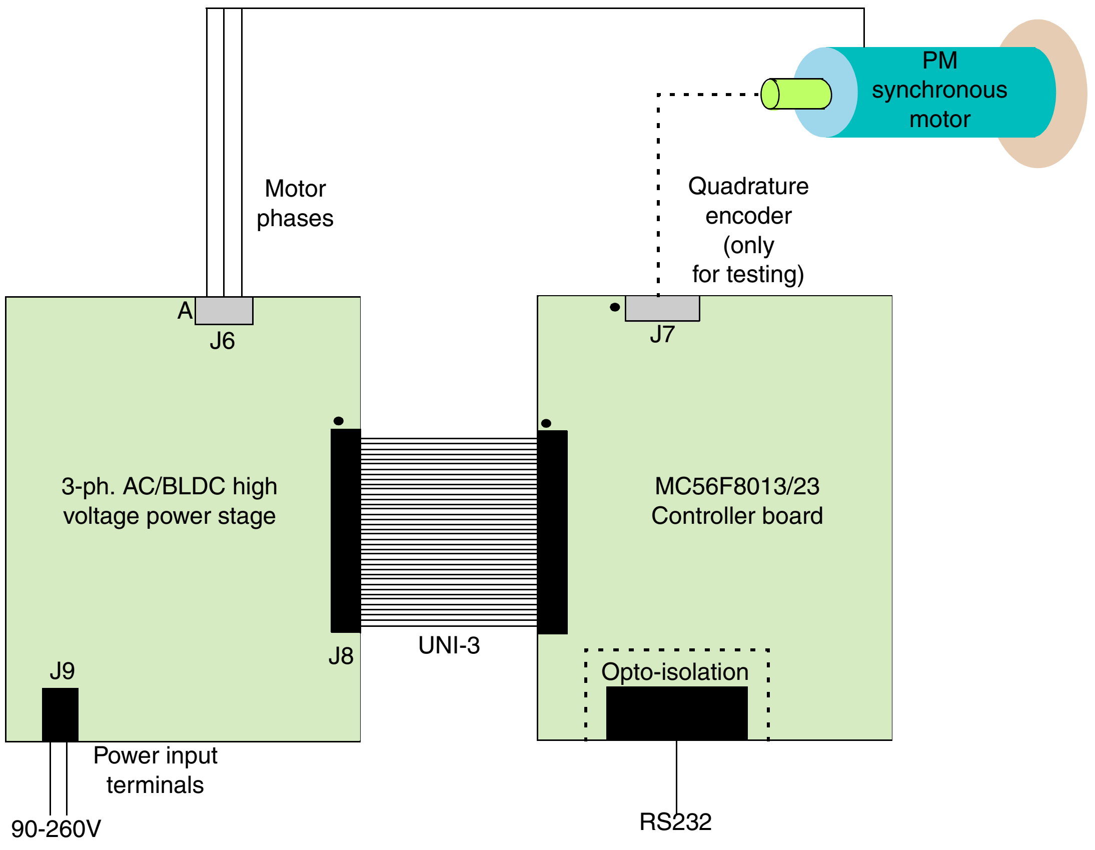


Figure 6-3. Demo Application Connection Overview

**CAUTION**

There is a risk of electric shock. The MC56F8013/23 controller board and the 3 ph AC/BLDC high voltage power stage board are connected to high voltage. The start/stop toggle switch and the up/down buttons on the MC56F8013/23 controller board are disabled in this application, avoid touching them. The only safe mode of control is by remotely controlling the application using the host PC running the FreeMASTER application. The RS232 port is the only interface that provides galvanic isolation.

**CAUTION**

The JTAG connector does not provide galvanic isolation. If a JTAG connection is required the demo hardware set-up has to be disconnected from high-voltage. To debug the application or to download code to the device flash memory using JTAG, use a low-voltage +12 V power supply connected to the J12 connector on the MC56F8013/23 controller board.

# Chapter 7
# Results and Measurements

This section describes measurements of the sensorless PMSM vector control application with an SMO. One of the most important parts of this sensorless PMSM controller is the sliding mode observer. Most of the measurements are concentrated on it.

## 7.1 System and Measurement Conditions

### 7.1.1 HW Setup

The application measurements were provided using the TGT3-0065-30-320 motor and the default HW setup described in Chapter 6, "Application Setup". The motor load is provided by the Vibro meter SA 1PB43.

### 7.1.2 SW Setup

The SW file PMSM_SensorlessCL_DQHRel is used for the a,b coordinate observer and the PMSM_SensorlessCL_AlpBetNDHRel file is used for the d, q coordinate observer measurements.

The measurements were provided using default code parameter settings, if not mentioned otherwise. The pre-processor constants CODE_MEASURE_ERROR_POSITION_SPEED, CODE_WITH_ENCODER, CODE_SMO_CURR_ERROR, and MEASURE_CODE_EXE_TIME are defined in the PMSM_ClosedLoop.h file. These enable compilation of the code for measurements.

### 7.1.3 FreeMASTER

The measurements were obtained with the FreeMASTER control/communication tool using the recorder feature. The recorder reads the defined variables, with sampling defined by the execution frequency of the function FMSTR_Recorder().

The PMSM_Sinusodial_Sensorless_Tuning.pmp file is used by the FreeMASTER SW and needs to be installed on the PC. The pmp file is included within the SW structure (PMSM_SensorlessCL_DQHRel or PMSM_SensorlessCL_AlpBetNDHRel). This incorporates the definition of the recorders used for the measurements.

### 7.1.4 Rotor Angle Estimation Error

The speed estimation and rotor position estimation precision measurements are obtained using an encoder. Therefore, the set-up with encoder connection and code (CODE_WITH_ENCODER defined) are used.

The sine and cosine errors are calculated according to:

$$\Delta sin = \hat{sin}(\theta) - sin(\theta) \qquad \textbf{Eqn. 7-1}$$

$$\Delta cos = \hat{cos}(\theta) - cos(\theta) \qquad \textbf{Eqn. 7-2}$$

The application does not quantify the rotor angle because the cosine and sine are sufficient information for all transformations. But for measurement reasons a function that approximates the angle error is implemented.

$$scalarErr = \sqrt{\Delta sin^2 + \Delta cos^2} \cdot sgn(\Delta\theta) \qquad \textbf{Eqn. 7-3}$$

Where:

$$sgn(\Delta\theta) = sgn(sin(\theta) \cdot \hat{cos}(\theta) - cos(\theta) \cdot \hat{sin}(\theta)) \qquad \textbf{Eqn. 7-4}$$

In the range of small errors, below 30 degrees. The error value is close to the angle estimation error:

$$\Delta\theta \cong scalarErr = \sqrt{\Delta sin^2 + \Delta cos^2} \cdot sgn(\Delta\theta) \qquad \textbf{Eqn. 7-5}$$

Equation 7-3 and Equation 7-4 can be calculated easier than two calculations of arcsine estimated and real value. Therefore the scalarErr is used for the measurement as an approximation of Δθ.

The code is included when the preprocessor constant CODE_MEASURE_ERROR_POSITION_SPEED is defined.

## 7.2 Sensorless Estimation with the d, q Coordinate SMO

The SMO estimation precision and dynamics have the highest impact on the sensorless vector control application's overall performance. This section describes the measurements and the measurement results of the system's static and dynamic responses. Each measurement includes some graphs with concentration on the estimation angle and estimation speed error. The measurements were provided using the default SW parameter settings.

**NOTE**

The default SW parameters settings are a trade-off between constant speed and dynamic performance. The SMO feedback, regulators and speed ramps can be set for lower errors at constant speed/torque or at dynamic transients.

### 7.2.1 Required Speed Step Transient – d, q Coordinate SMO

This section presents measurements with a required speed step transient. The speed step is defined in the required speed column of Table 7-1.

Table 7-1. Required Speed Transient

| Required Speed | Load Torque | Sampling Period | Figure | Position Error | Speed Error |
|---|---|---|---|---|---|
| rpm | Nm | ms | | Degree | rpm |
| 500 to 3000 | 0 | 2.625 | Figure 7-1 | –2.0 to 0.5 | –30 to 20 |
| 500 to 3000 | 0.4 | 2.625 | — | –2.0 to 1.0 | –50 to 25 |
| 3000 to 500 | 0 | 2.625 | — | –2.0 to 2.5 | –50 to 25 |
| 3000 to 500 | 0.4 | 2.625 | Figure 7-2 | –1.5 to 3.0 | –30 to 30 |

The recorder pictures are sensed using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/performance measurement/speed error cosine-angle error recorder. The variable transients are displayed in Figure 7-1 and Figure 7-2. The results are displayed in Table 7-1.

The variable meanings:

| | |
|---|---|
| speedMotorSens | rotor speed sensed by the encoder |
| speedMotorEstim | rotor speed estimated by the SMO |
| speedMotorErr | difference between speedMotorEstim and speedMotorSens |
| sinCosEst.cos | estimated rotor position cosine |
| sinCosSens.cos | sensed rotor position cosine |
| sinCosElScalarErr | rotor electrical angle estimation error according to equation Equation 7-5 |

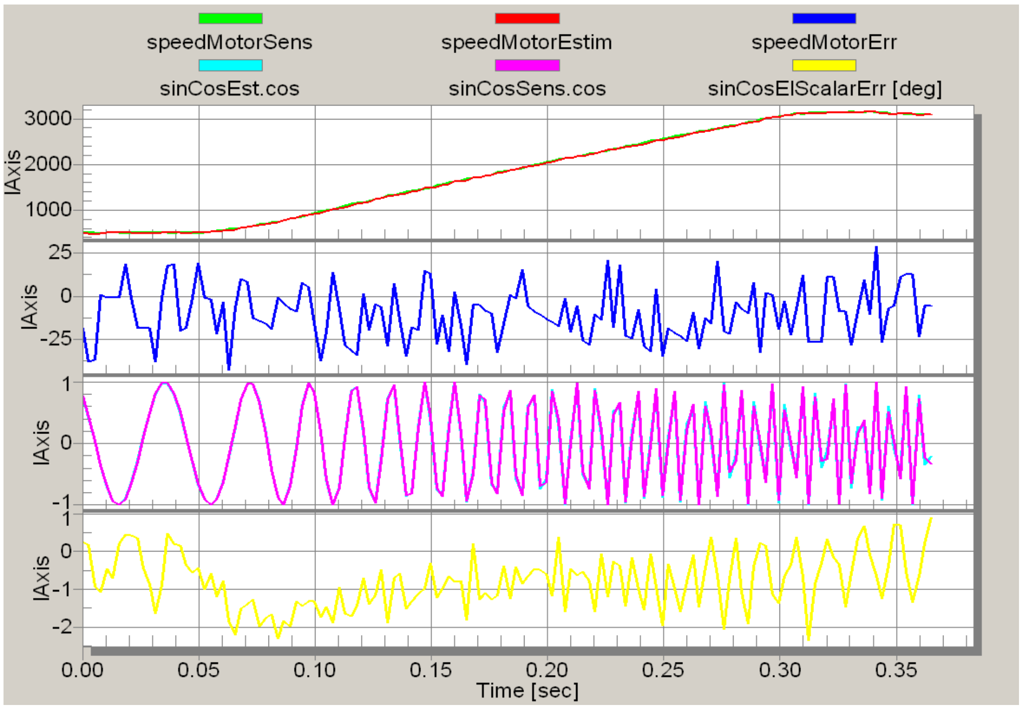


**Figure 7-1. Acceleration Measurement, Speed 500 to 3000 rpm, Load 0 Nm**

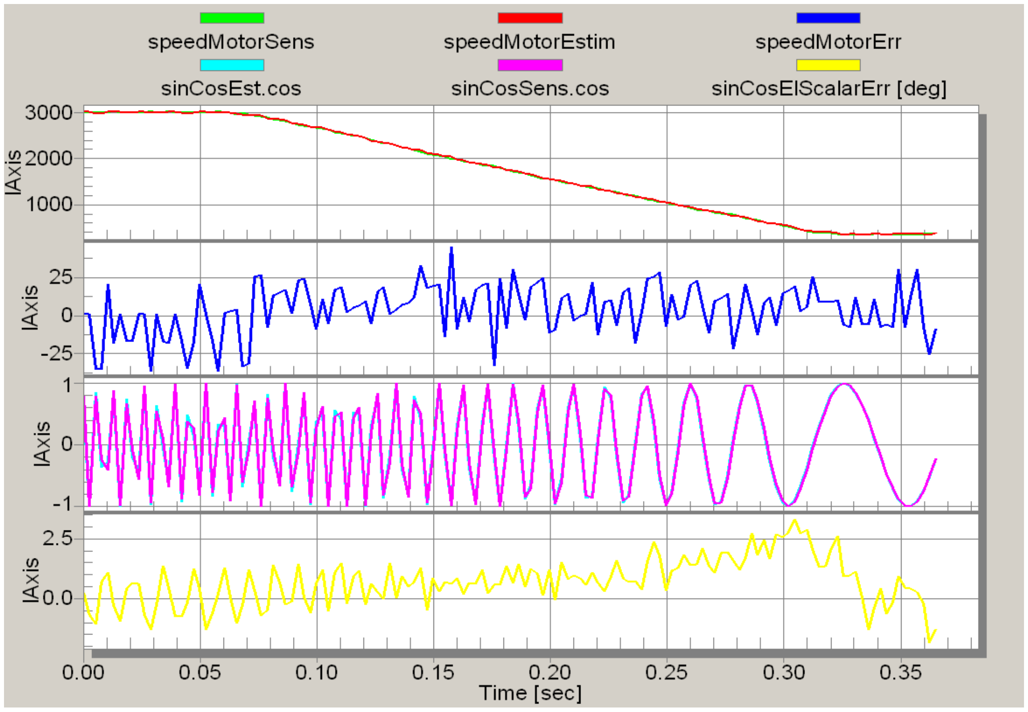


**Figure 7-2. Deceleration Measurement, Speed 3000 to 500 rpm, Load 0.4 Nm**

### 7.2.2 Load Torque Step Transient – d, q Coordinate SMO

This section presents measurements with a constant required speed and load torque step transient at a defined speed range. The load torque step is defined in the table column load torque.

The recorder pictures are sensed using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/performance measurement/speed error cosine-angle error recorder. The variable transients are displayed in Figure 7-2 to Figure 7-4. The results are displayed in Table 7-2. The variable meanings are described in Section 7.2.1, “Required Speed Step Transient – d, q Coordinate SMO”.

Table 7-2. Load Torque Transient

| Required Speed | Load Torque | Recorder Name | Sampling Period | Figure | Position Error | Speed Error |
|---|---|---|---|---|---|---|
| rpm | Nm | | ms | | Degree | rpm |
| 3000 | 0 to 0.4 | — | 2.625 | — | –1.5 to 1.5 | ±25 |
| 3000 | 0.4 to 0 | — | 2.625 | Figure 7-3 | –2.0 to 2.0 | ±30 |
| 1000 | 0 to 0.4 | — | 2.625 | Figure 7-4 | –1.0 to 2.0 | ±30 |

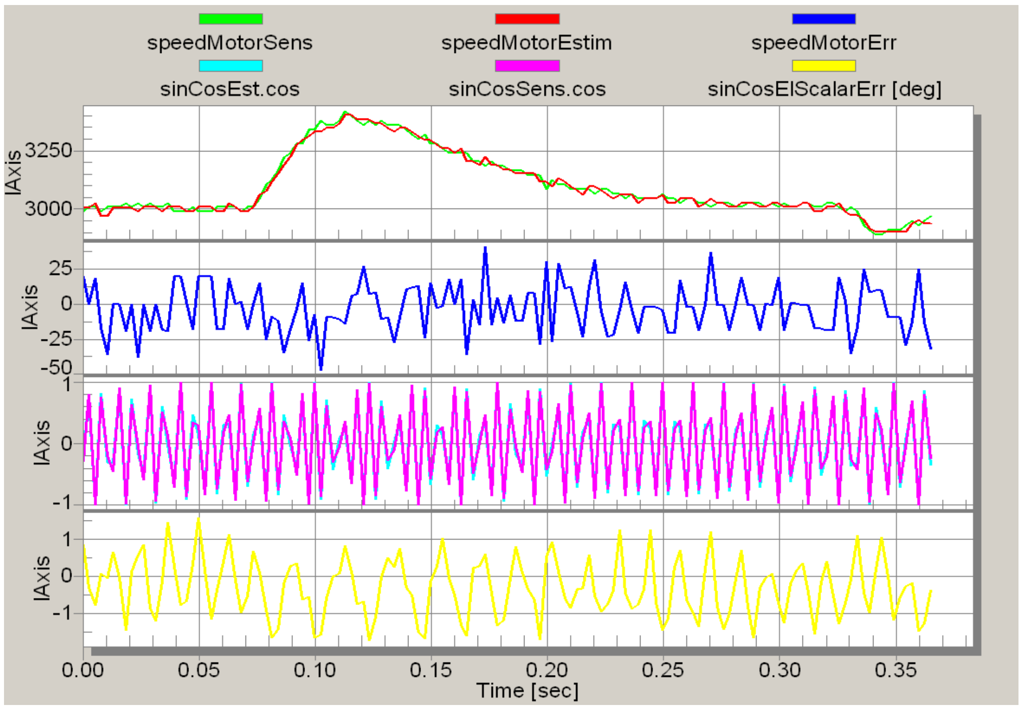


Figure 7-3. Load Transient Measurement, Speed 3000 rpm, Load 0.4 to 0 Nm

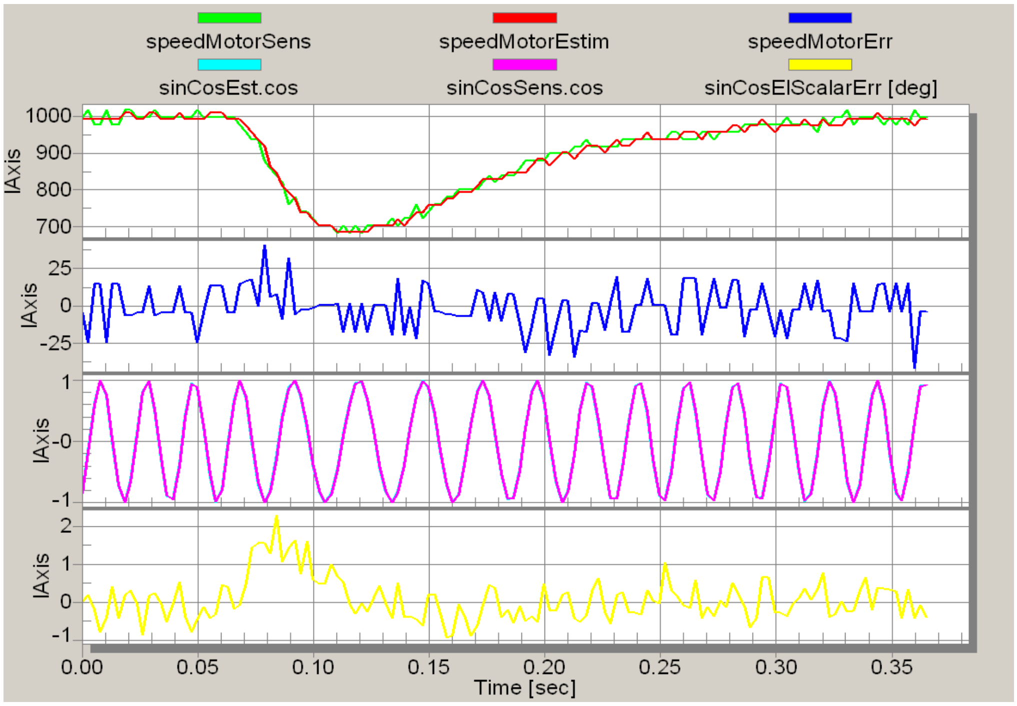


Figure 7-4. Load Transient Measurement, Speed 1000 rpm, Load 0 to 0.4 Nm

## 7.2.3 Constant Speed Measurements – d, q Coordinate SMO

This section presents measurements with a constant required speed and a constant load torque at a defined speed range.

Table 7-3. Constant Speed Measurements

| Required Speed | Load Torque | Sampling Period | Figure | Position Error | Speed Error |
|---|---|---|---|---|---|
| rpm | Nm | ms | | Degree | rpm |
| 400 | 0 | 2.625 | — | –1.5 to 1.5 | ±25 |
| 1000 | 0 | 2.625 | — | –1.0 to 1.0 | ±25 |
| 2000 | 0 | 2.625 | — | –1.0 to 1.0 | ±25 |
| 3000 | 0 | 2.625 | Figure 7-5 | –1.5 to 1.0 | ±30 |
| 400 | 0.4 | 2.625 | Figure 7-6 | –1.5 to 1.5 | ±25 |
| 1000 | 0.4 | 2.625 | — | –0.5 to 1.0 | ±25 |
| 2000 | 0.4 | 2.625 | — | –1.0 to 1.0 | ±25 |
| 3000 | 0.4 | 2.625 | — | –1.25 to 1.25 | ±30 |

The recorder pictures are sensed using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/performance measurement/speed error cosine-angle error recorder. The variable transients are displayed in Figure 7-5 and Figure 7-6. The results are displayed in Table 7-3. The variables are described in Section 7.2.1, “Required Speed Step Transient – d, q Coordinate SMO”.

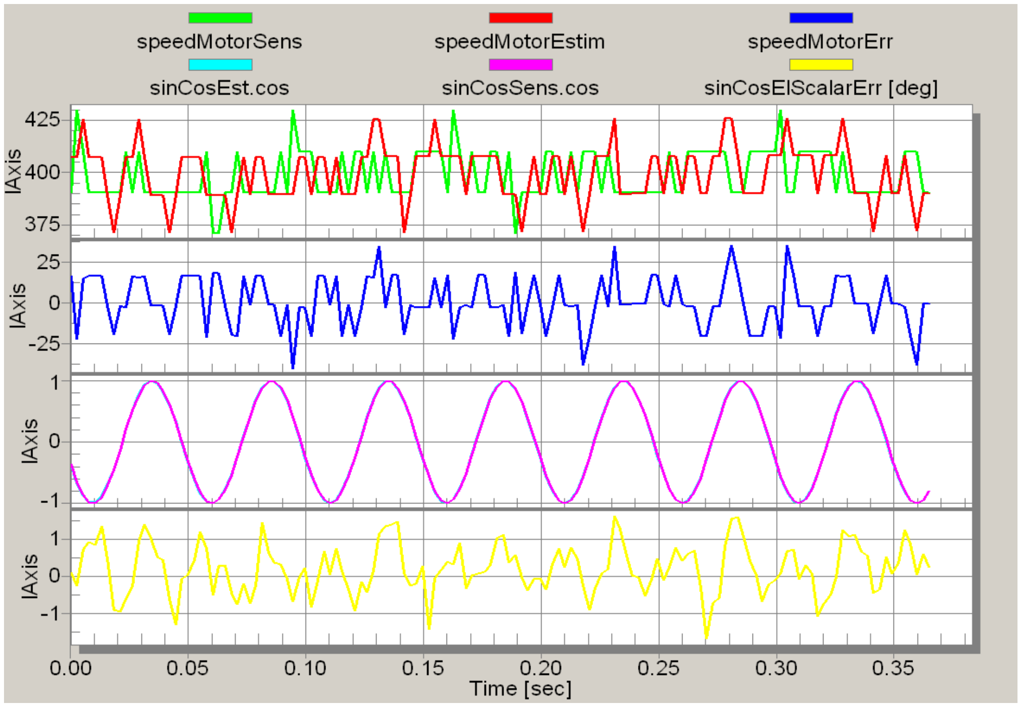


**Figure 7-5. Constant Speed Measurement, Speed 400 rpm, Load 0.4 Nm**

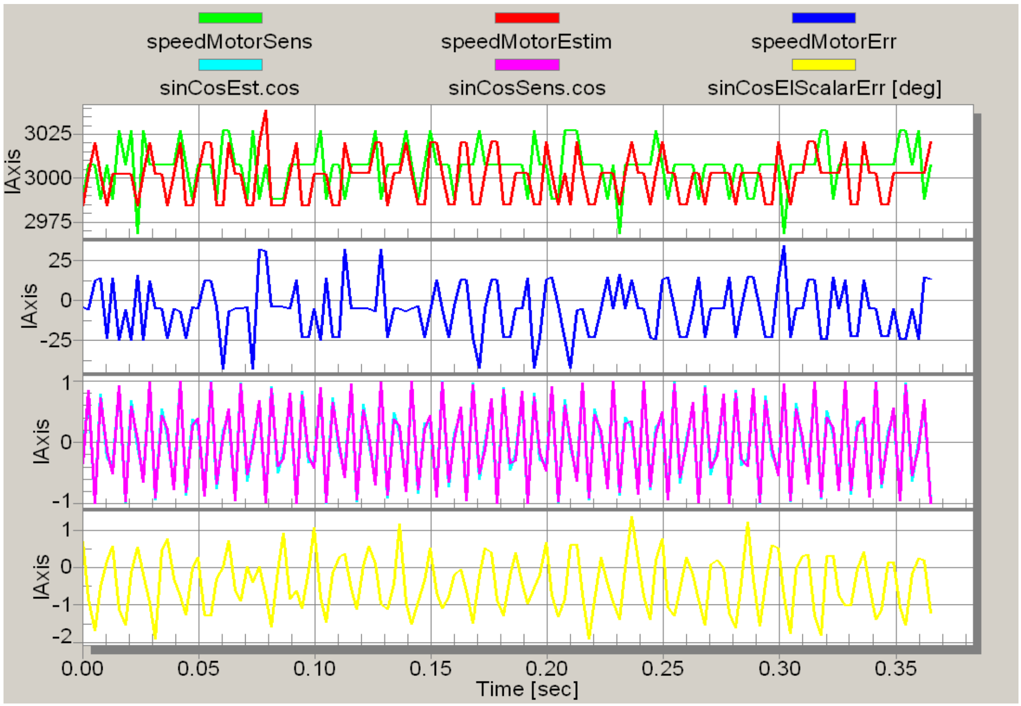


**Figure 7-6. Constant Speed Measurement, Speed 3000 rpm, Load 0 Nm**

## 7.2.4 Start-up – d, q Coordinate SMO

The start-up measurement incorporates the motor operations from 0 speed to a defined required speed. The control SW state machine starts with rotor alignment/stabilization continuing with an open-loop and regular running states.

**Table 7-4. Start-up Measurements**

| Required Speed | Load Torque | Sampling Period | Figure |
|---|---|---|---|
| rpm | Nm | ms | |
| 0 to 1000 | 0 | 5.375 | Figure 7-7 |
| 0 to 1000 | 0.4 | 5.375 | Figure 7-8 |

The recorder pictures are sensed using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/start-up tuning/angle cosine curr error recorder. The variable transients are displayed in Figure 7-7 to Figure 7-8. The results are displayed in Table 7-4.

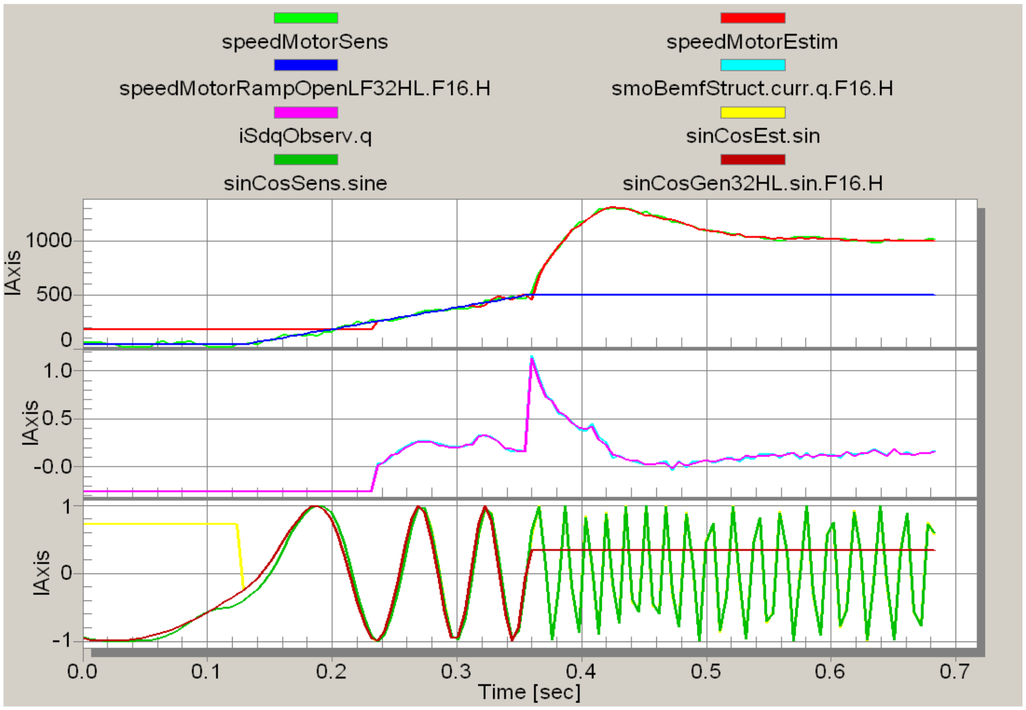


**Figure 7-7. Start-up Measurement, Speed 0 to 1000 rpm, Load 0 Nm**

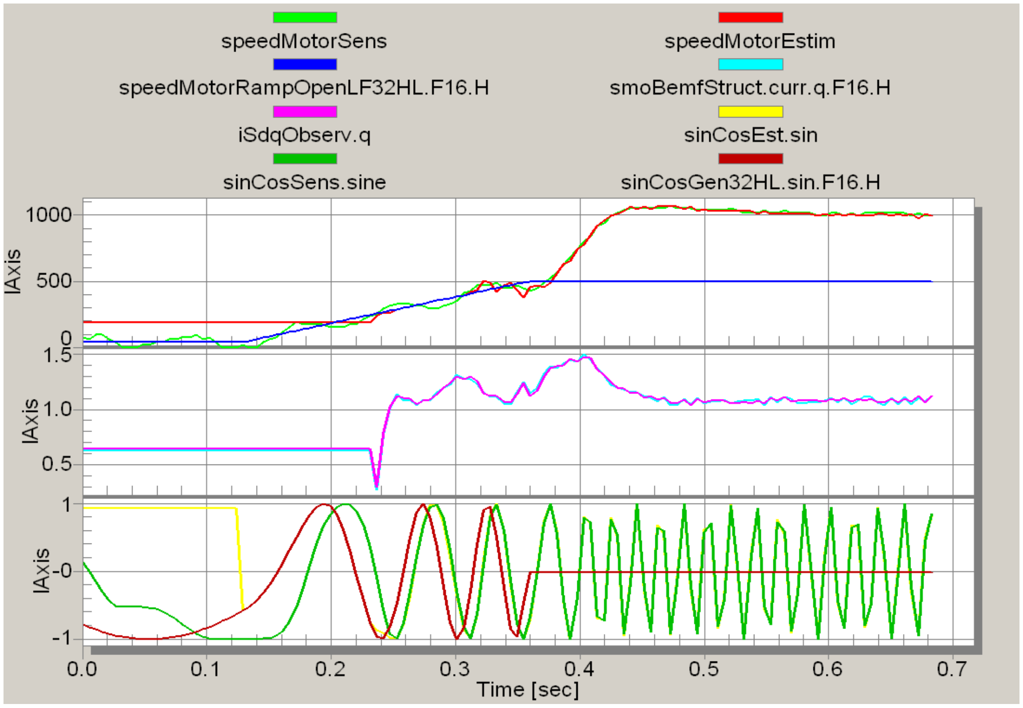


**Figure 7-8. Start-up Measurement, Speed 0 to 1000 rpm, Load 0.4 Nm**

## 7.3 Sensorless Estimation with the α,β Coordinate SMO

The SMO estimation precision and dynamics have the highest impact on the sensorless vector control application's overall performance. This section describes the measurements and the measurement results of the system's static and dynamic responses. Each measurement includes some graphs with concentration on the estimation angle and estimation speed error. The measurements were provided using the default SW parameter settings.

**NOTE**

The default SW parameters settings are a trade-off between constant speed and dynamic performance. The SMO feedback, regulators, and speed ramps can be set for lower errors at constant speed/torque or at dynamic transients.

### 7.3.1 Required Speed Step Transient – α,β Coordinate SMO

This section presents measurements with a required speed step transient. The speed step is defined in the required speed column of Table 7-5.

The recorder pictures are sensed using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/performance measurement/speed error curr beta cosine-angle error recorder. The variable transients are displayed in Figure 7-9 and Figure 7-10. The results are displayed in Table 7-5. The variables are described after Figure 7-10.

Table 7-5. Required Speed Transient

| Required Speed | Load Torque | Sampling Period | Figure | Position Error | Speed Error |
|---|---|---|---|---|---|
| rpm | Nm | ms | | Degree | rpm |
| 500 to 3000 | 0 | 2.625 | Figure 7-9 | –4.0 to 1.0 | ±45 |
| 500 to 3000 | 0.4 | 2.625 | — | –4.0 to 1.0 | ±45 |
| 3000 to 500 | 0 | 2.625 | — | –3.0 to 3.0 | ±45 |
| 3000 to 500 | 0.4 | 2.625 | Figure 7-10 | –3.0 to 3.0 | ±50 |

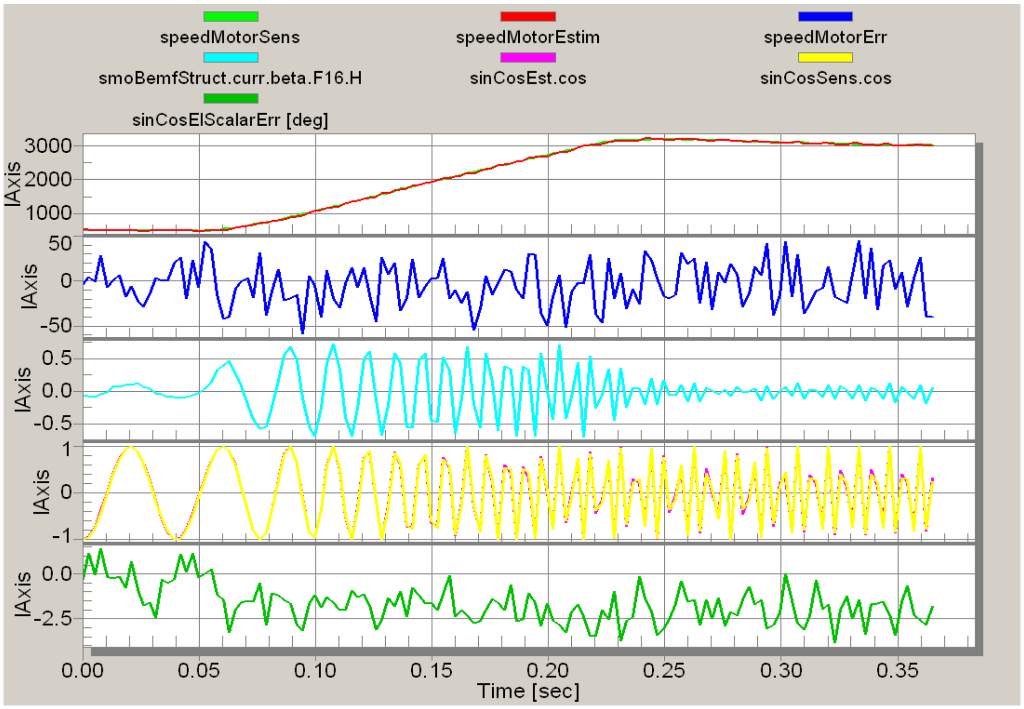


**Figure 7-9. Acceleration Measurement, Speed 500 to 3000 rpm, Load 0 Nm**

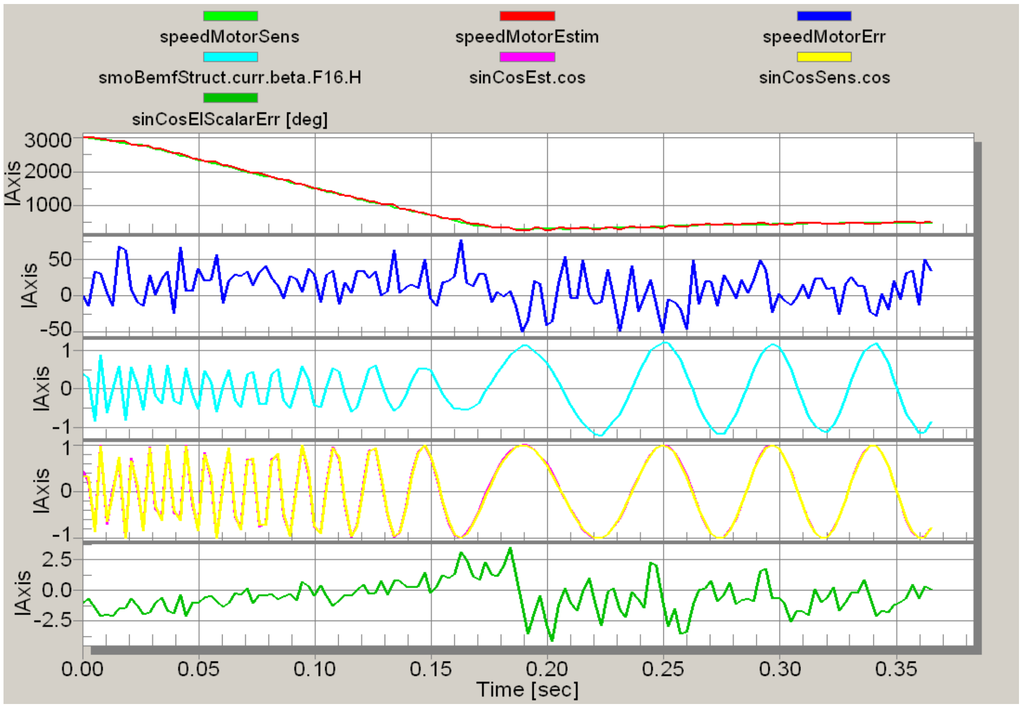


**Figure 7-10. Deceleration Measurement, Speed 3000 to 500 rpm, Load 0.4 Nm**

The figures display the following variables:

| | |
|---|---|
| speedMotorSens | rotor speed sensed by the encoder |
| speedMotorEstim | rotor speed estimated by the SMO |
| speedMotorErr | difference between estimated and measured rotor speed speedMotorEstim – speedMotorSens |
| smoBemfStruct.curr.beta.F16.H | estimated current – beta component |
| sinCosEst.cos | estimated rotor position cosine |
| sinCosSens.cos | sensed rotor position cosine |
| sinCosElScalarErr | rotor electrical angle estimation error according to equation Equation 7-5 |

## 7.3.2 Load Torque Step Transient – $\alpha,\beta$ Coordinate SMO

This section presents measurements with a constant required speed and load torque step transient at a defined speed range. The load torque step is defined in the table column load torque.

**Table 7-6. Load Torque Transient**

| Required Speed | Load Torque | Recorder Name | Sampling Period | Figure | Position Error | Speed Error |
|---|---|---|---|---|---|---|
| rpm | Nm | | ms | | Degree | rpm |
| 3000 | 0 to 0.4 | — | 2.625 | — | –0.5 to –2.5 | ±50 |
| 3000 | 0.4 to 0 | — | 2.625 | Figure 7-11 | –1 to –3.5 | ±40 |
| 1000 | 0 to 0.4 | — | 2.625 | Figure 7-12 | 0.5 to –1.5 | ±50 |

The recorder pictures are sensed using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/performance measurement/speed error curr beta cosine-angle error recorder. The variable transients are displayed in Figure 7-11 and Figure 7-12. The results are displayed in Table 7-6. The variables are described in Section 7.3.1, “Required Speed Step Transient – a,b Coordinate SMO”.

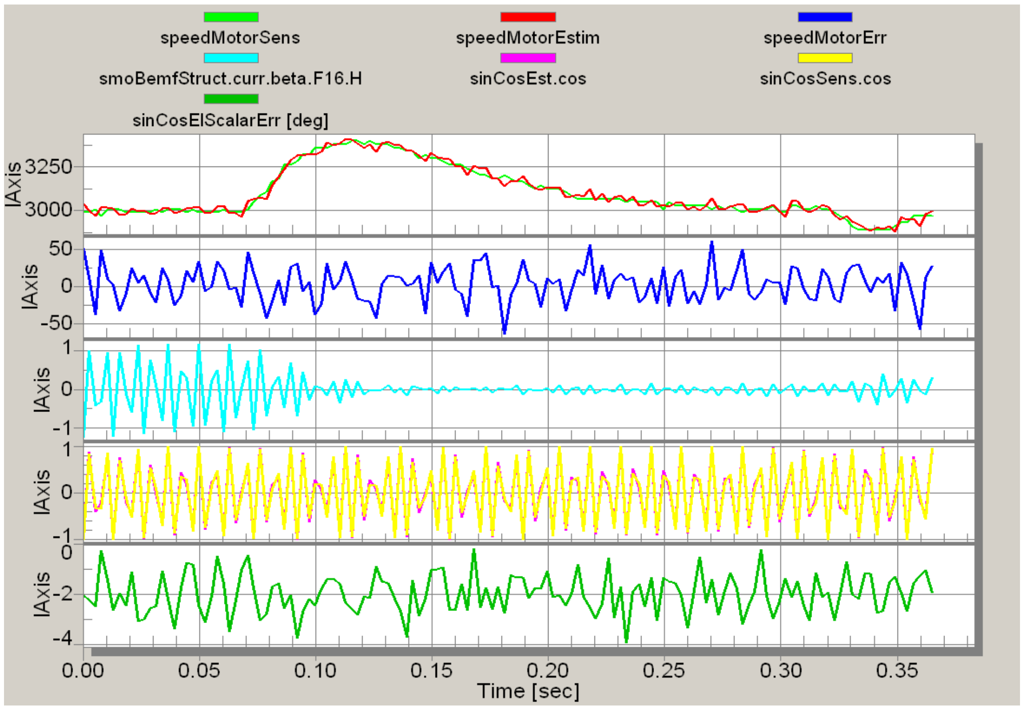


Figure 7-11. Load Transient Measurement, Speed 3000 rpm, Load 0.4 to 0 Nm

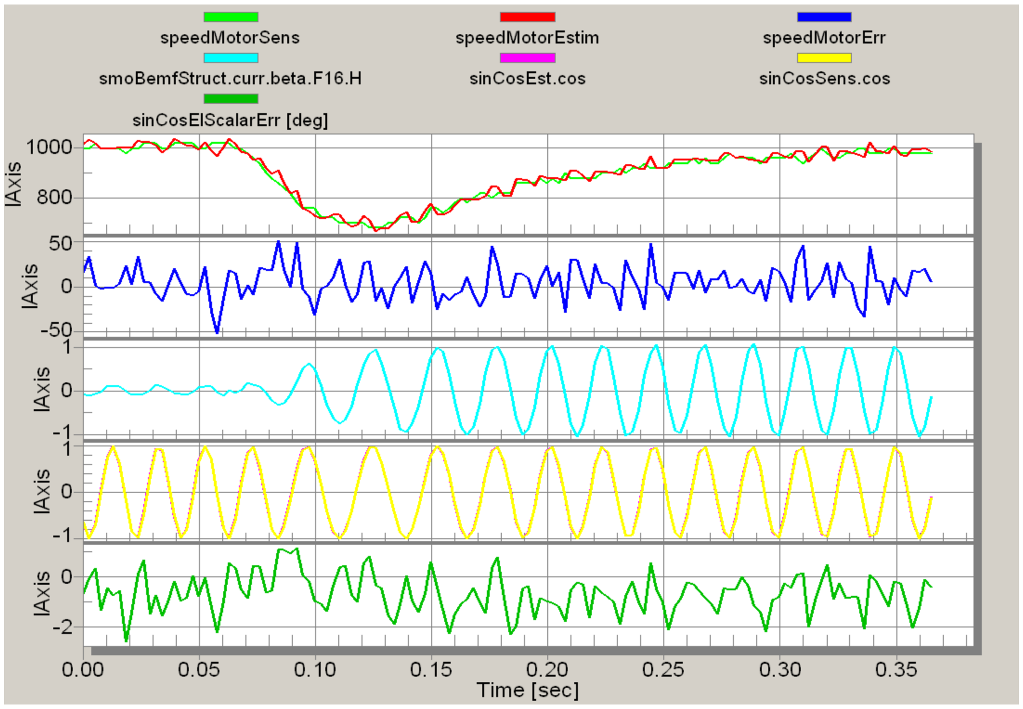


Figure 7-12. Load Transient Measurement, Speed 1000 rpm, Load 0 to 0.4 Nm

### 7.3.3 Constant Speed Measurements – α,β Coordinate SMO

This section presents measurements with a constant required speed and a constant load torque at a defined speed range. The recordings in Table 7-7 are obtained using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/start-up tuning/speed angle sine curr error recorder.

The recorder pictures are sensed using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/performance measurement/speed error curr beta cosine-angle error recorder. The variable transients are displayed in Figure 7-13 and Figure 7-14. The results are displayed in Table 7-7. The variables are described in Section 7.3.1, "Required Speed Step Transient – a,b Coordinate SMO".

Table 7-7. Constant Speed Measurements

| Required Speed | Load Torque | Sampling Period | Figure | Position Error | Speed Error |
|---|---|---|---|---|---|
| rpm | Nm | µs | | Degree | rpm |
| 400 | 0 | 875 | — | 2.0 to –2.0 | ±25 |
| 1000 | 0 | 875 | — | 1.0 to –1.5 | ±25 |
| 2000 | 0 | 875 | — | 0.0 to –2.0 | ±30 |
| 3000 | 0 | 875 | Figure 7-13 | 0 to –2.5 | ±30 |
| 400 | 0.4 | 875 | Figure 7-14 | –2.5 to 2.0 | ±30 |
| 1000 | 0.4 | 875 | — | 1.0 to –1.5 | ±30 |
| 2000 | 0.4 | 875 | — | 0 to –2.5 | ±30 |
| 3000 | 0.4 | 875 | — | 0 to –2.5 | ±30 |

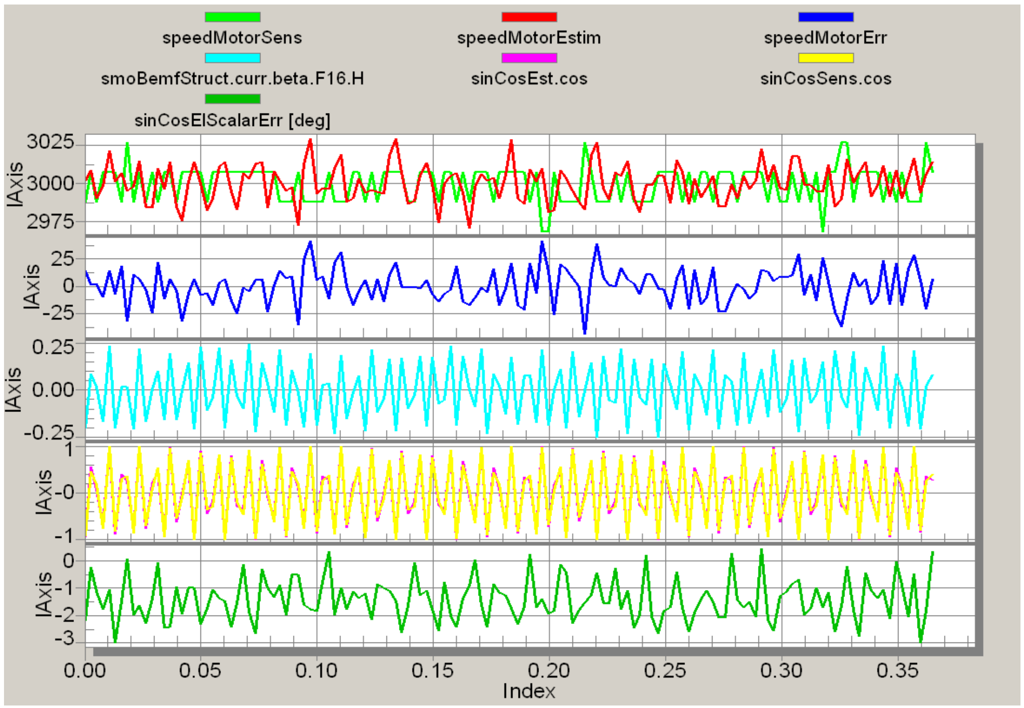


**Figure 7-13. Constant Speed Measurement, Speed 3000 rpm, Load 0 Nm**

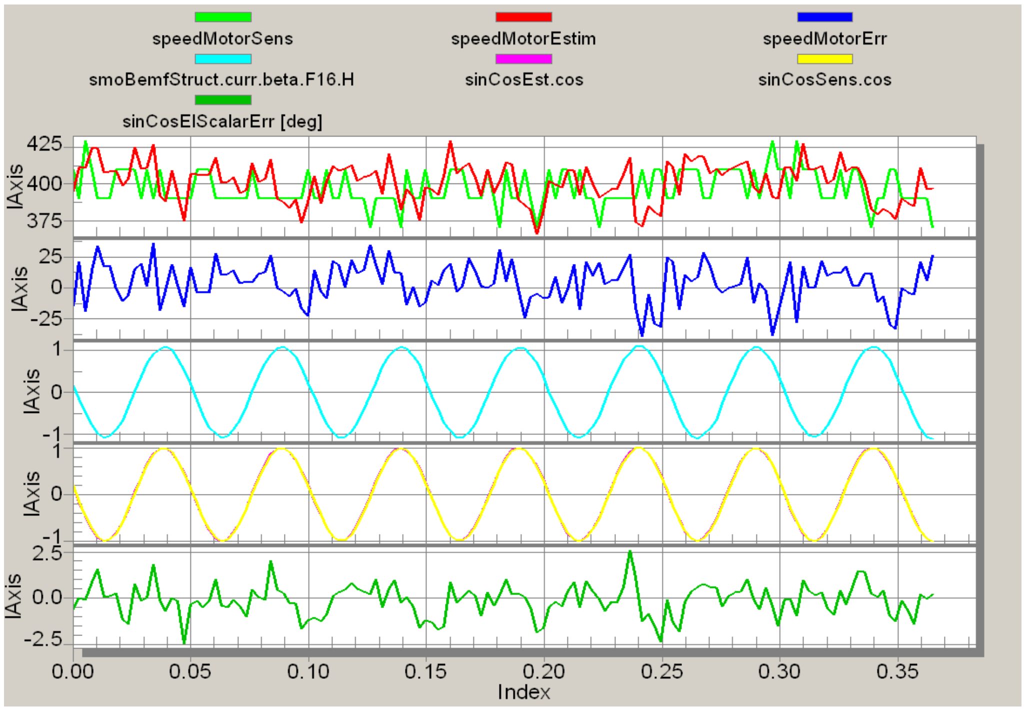


**Figure 7-14. Constant Speed Measurement, Speed 400 rpm, Load 0.4 Nm**

## 7.3.4 Start-up – $\alpha,\beta$ coordinate SMO

The start-up measurement discusses the motor operations starting from 0 speed to a defined required speed. The control SW state machine starts with rotor alignment/stabilization continuing with an open-loop and regular running states.

**Figure 7-15. Start-up Measurements**

| Required Speed | Load Torque | Sampling Period | Figure |
|---|---|---|---|
| rpm | Nm | ms | |
| 0 to 1000 | 0 | 5.375 | Figure 7-16 |
| 0 to 1000 | 0.4 | 5.375 | Figure 7-17 |

The recorder pictures are sensed using the FreeMASTER project PMSM_Sinusodial_Sensorless_Tuning.pmp with its recorder observer error tuning/start-up tuning/speed angle sine curr error recorder. The variable transients are displayed in Figure 7-16 and Figure 7-17. The results are displayed in Table 7-15.

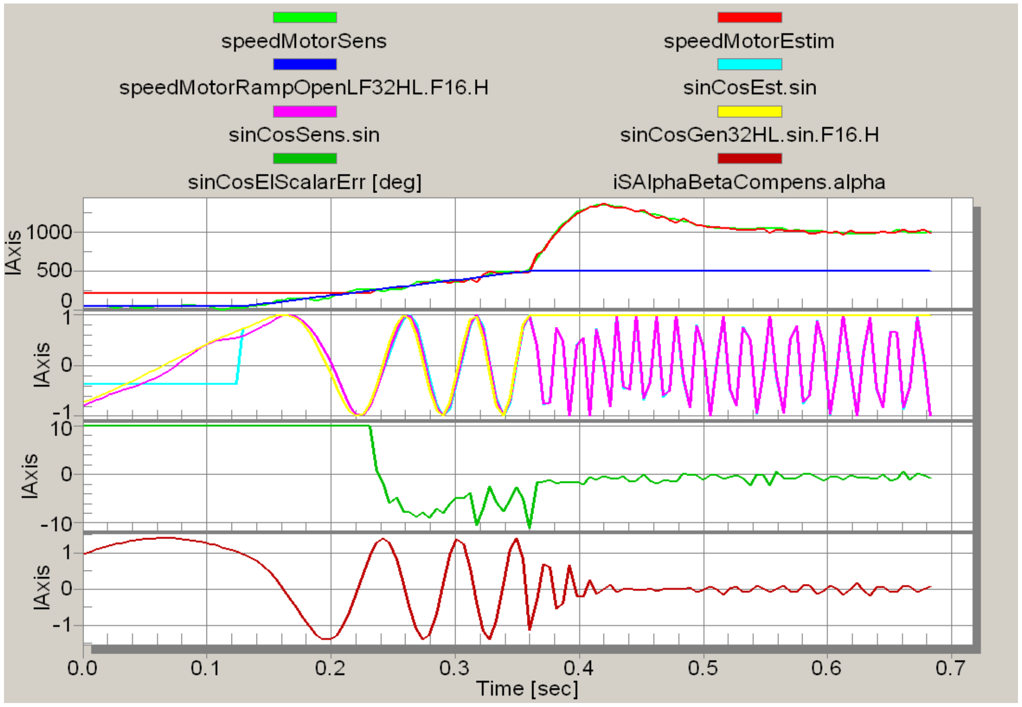


**Figure 7-16. Start-up Measurement, Speed 0 to 1000 rpm, Load 0 Nm**

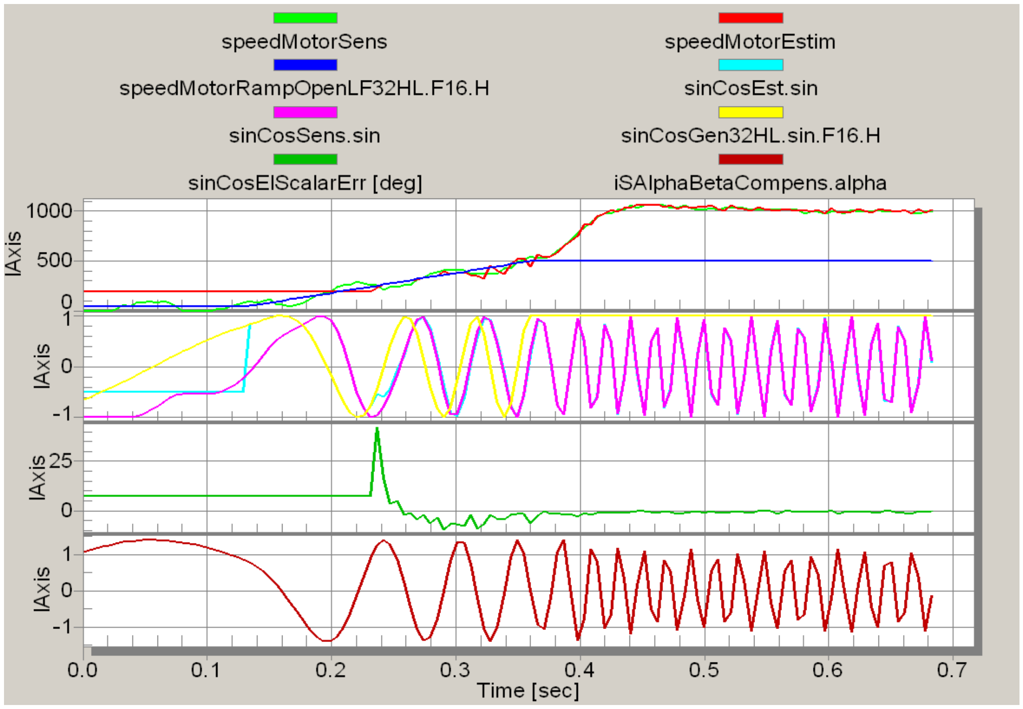


**Figure 7-17. Start-up Measurement, Speed 0 to 1000 rpm, Load 0.4 Nm**

## 7.4 Conclusion

The measurement results show that both SMOs give stable results over the defined speed and load torque range. As mentioned above, one of the most important features of any position and speed observer is the correct functionality over the required speed and load range. Special attention needs to be given to the dynamic response of the estimation.

The application with the d, q coordinates model gives slightly better results than the α, β model. However, the difference is not that significant because the BLDC motor TGT3-0065-30-320 has constant parameters over the rotor angle. The benefit of the α,β is a better stability and starting due to the simplified model. For motors like the TGT3-0065-30-320, it is recommended to use the application with the α, β model.