PMSM Vector Control with Single-Shunt Current-Sensing Using MC56F8013/23

Design Reference Manual

Devices Supported:

MC56F8023
MC56F8013

Document Number: DRM102
Rev. 0
04/2008

freescale™
semiconductor

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

DRM102
Rev. 0
04/2008

Chapter 1
Introduction

Chapter 2
Control Theory

Chapter 3
System Concept

Chapter 4
Hardware

Chapter 5
Software Design

Chapter 6
Application Setup

Chapter 7
Results and Measurements

Chapter 1
Introduction

1.1 Introduction

This paper describes the design of a three-phase PMSM vector control drive with single shunt current sensing, using a quadrature encoder as a position and speed sensor. The design is targeted for consumer and industrial applications. This cost-effective solution benefits from the dedicated motor control features in the Freescale Semiconductor MC56F8023 device.

1.1.1 Application Features and Components

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

- Three-phase PMSM speed vector (FOC) control using quadrature encoder
- Current sensing using only a current sensor
- Freescale MC56F8023 controller
- Three-phase high voltage (230/115 V) power stage
- FreeMASTER software control interface and monitor

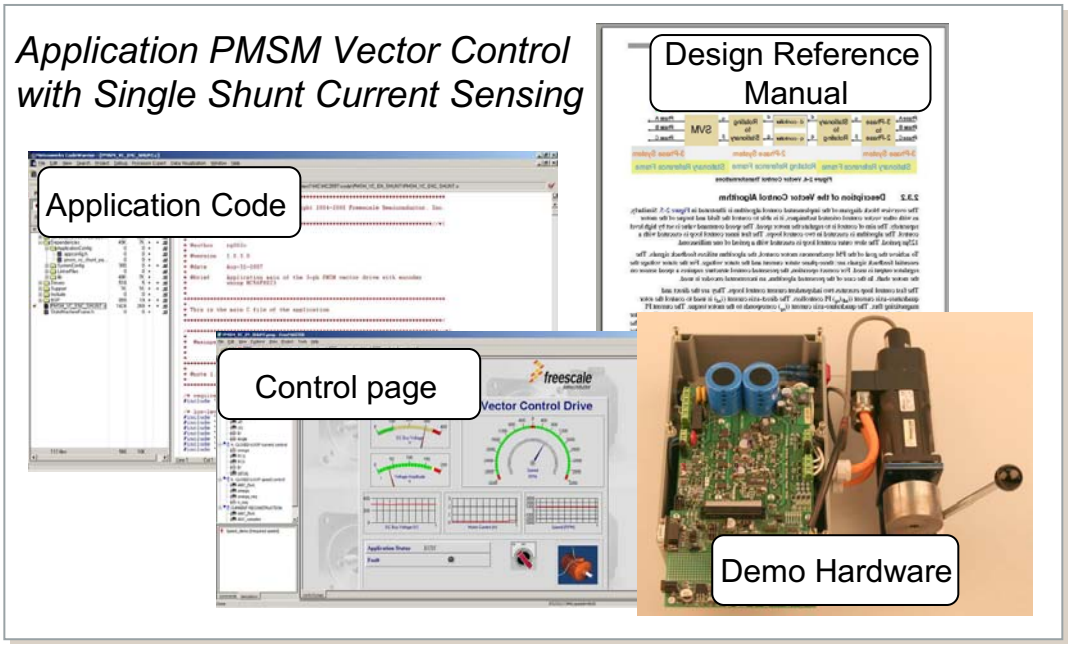

Figure 1-1. PM Synchronous Motor Vector Control Application Using MC56F8023

The main application components available for customers are:

- Software — written in C-code using some library algorithms available for the MC56F8013/23
- Hardware — based on Freescale universal motor control hardware modules
- Documentation — this document

1.1.2 Overview of Sinusoidal PM Synchronous Motor Application

Sinusoidal PM synchronous motors are more and more popular for new drives, replacing brushed DC, universal, and other motors in a wide variety of applications. The reasons are better reliability (no brushes), better efficiency, lower acoustic noise, and other benefits of electronic control. A disadvantage of PM synchronous motor drives might be the need for a more sophisticated electronic circuit. But today, most applications need electronic speed regulation, torque regulation, or other features that require electronic control. If we use a system with electronic control, there is only a small increase in system cost if we implement more advanced drives, like the sinusoidal PM synchronous motor with digitally controlled switching inverter and DC-bus circuit. It is necessary only to have a cost-effective controller with good performance when doing calculations. One of them is the Freescale MC56F8023, or similar devices like the MC56F803x or MC56F8346.

The PM synchronous motor also has advantages when 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 major household appliances (such as refrigerators, washing machines, and 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 very similar to the trapezoidal BLDC (electronically commuted) motor. There are two main differences:

- Motor construction — Shape of 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, we can say that sinusoidal PM synchronous motor performance has the advantage of constant torque, but the trapezoidal BLDC motor is easier to control. The trapezoidal BLDC motors are used mainly for historical reasons. It was 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 sophisticated control, but have some benefits, such as smoother torque, lower acoustic noise, etc. As shown in this document, the MC56F8023 provides all the necessary functionality for three-phase sinusoidal PM synchronous motor vector control. Using the MC56F8023, we have a strong argument for replacing the trapezoidal BLDC (electronically commuted) motor with a three-phase sinusoidal PM synchronous motor with almost no increase in system cost.

The application described here is a speed vector control. Speed control algorithms can be sorted into two general groups. The first group is scalar control algorithms. The constant-volt-per-hertz method is a very popular technique for scalar control. The other group is called vector-oriented or field-oriented control (FOC). Compared to scalar control, vector-oriented techniques bring overall improvement to drive

performance. Some advantages are higher efficiency, full torque control, decoupled control of flux and torque, improved dynamics, etc.

For PM synchronous motor control, it is necessary to know the exact rotor position. The quadrature encoder is used as a speed and position sensor in this application.

Two key requirements of the described design are that it be cost-effective and highly reliable. To minimize system cost, the algorithm implements single-shunt current sensing, which replaces three current sensors with one.

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

1.2 Freescale Controller Advantages and Features

The Freescale MC56F80xx family is well-suited to digital motor control, combining 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, analog-to-digital converters (ADC), timers, communications peripherals (SCI, SPI, I2C), and onboard flash and RAM.

The MC56F80xx family members provide the following features:

- One PWM module with PWM outputs, fault inputs, fault-tolerant design with dead time insertion, support of both center-aligned and edge-aligned modes
- 12-bit ADC, supporting two simultaneous conversions; 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 (I^2C) port
- Onboard 3.3 V to 2.5 V voltage regulator for powering internal logic and memories
- Integrated power-on reset and low voltage interrupt module
- Multiplexing of all pins with general-purpose input/output (GPIO) pins
- Computer operating properly (COP) watchdog timer
- Two internal 12-bit digital-to-analog converters (DACs)
- Two analog comparators
- One programmable interval timer (PIT)
- 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	MC56F8023
Program Flash	32 KByte
Unified Data/Program RAM	4 KByte

The PMSM vector control with single-shunt current sensing 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. It is capable of generating asymmetric PWM duty cycles in center-aligned configuration. We can benefit from this feature 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 some combination of these options)
- 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 center-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 — 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 μs)
- Additional conversion time of six ADC clock cycles (6 x 187 ns ≈ 1.125 μs)
- Eight conversions in 26.5 ADC clock cycles (26.5 x 187 ns ≈ 4.97 μs) 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
- Capability to generate interrupt 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 analog inputs

This 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 analog values for the DC-link current and voltage, within the required time.

The QuadTimer 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 will cascade
- 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 can be preloaded
- 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:

- 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 Bibliography

1. *56F8023 Data Sheet*, MC56F8023, Freescale Semiconductor, 2006
2. *56F802X and 56F803X Peripheral Reference Manual*, MC56F80XXRM, Freescale Semiconductor, 2006
3. *56F801X Peripheral Reference Manual*, MC56F8000RM, Freescale Semiconductor, 2006
4. *DSP56800E Reference Manual*, DSP56800ERM, Freescale Semiconductor, 2005
5. *CodeWarrior Development Studio for Freescale 56800/E Digital Signal Controllers*, Freescale Semiconductor, 2006
6. *MC56F8013/23 Controller Board Users Manual*, Freescale Semiconductor, 2007
7. *3-Phase AC/BLDC High Voltage Power Stage Board Users Guide*, Freescale Semiconductor, 2007
8. *AN2395 — PC Master Software Usage*, Freescale Semiconductor, 2002
9. *3-Phase PM Synchronous Motor Vector Control*, AN1931, by Prokop L., Grasblum P., 2005

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

1.3.1 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
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

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

Term	Meaning
PLL	Phase-locked loop: a clock generator circuit in which a voltage controlled oscillator produces an oscillation that is 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

1.3.2 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 that 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 given as a percentage
Quadrature encoder	A position sensor giving two code tracks with sector positioned 90 degrees out of phase — the two output channels indicate both position and direction of motor rotation
Interrupt	A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine
PM synchronous motor	Permanent magnet synchronous motor
PI Controller	Proportional-integral controller
Quick Start	Software 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

1.3.3 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

Table 1-4. Glossary (Continued)of Symbols

Term	Definition
i_{Sa}, i_{Sb}, i_{Sc}	Stator currents in three-phase coordinate system
$i_{Sd,q}$, $i_{S(d,q)}$)	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 α,β coordinate system
$\hat{i}_{S\alpha,\beta}$, $\hat{i}_{S(\alpha,\beta)}$	Estimated stator currents in α,β coordinate system
$\hat{i}_{Sd,q}$, $i_{S(d,q)}$	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
L_s	Stator phase inductance
ΔL_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
t_e	Electromagnetic torque
T_L	Load torque
u_{SA}, u_{SB}, u_{SC}	Stator voltages in three-phase coordinate system
$u_{S\alpha,\beta}$, $u_{S(\alpha,\beta)}$	Stator voltages in α,β coordinate system
$u_{Sd,q}$, $u_{S(d,q)}$	Stator voltages in d,q coordinate system
α,β	Stator orthogonal coordinate system
$\Psi_{S\alpha,\beta}$	Stator magnetic fluxes in α,β coordinate system
$\Psi_{Sd,q}$	Stator magnetic fluxes in d,q coordinate system
Ψ_M	Rotor magnetic flux
θr	Rotor position angle in α,β coordinate system
ω_{el0}	Electrical rotor angular speed at a stand point
ω, ω_{el} / ω_F	Electrical rotor angular speed / field 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).

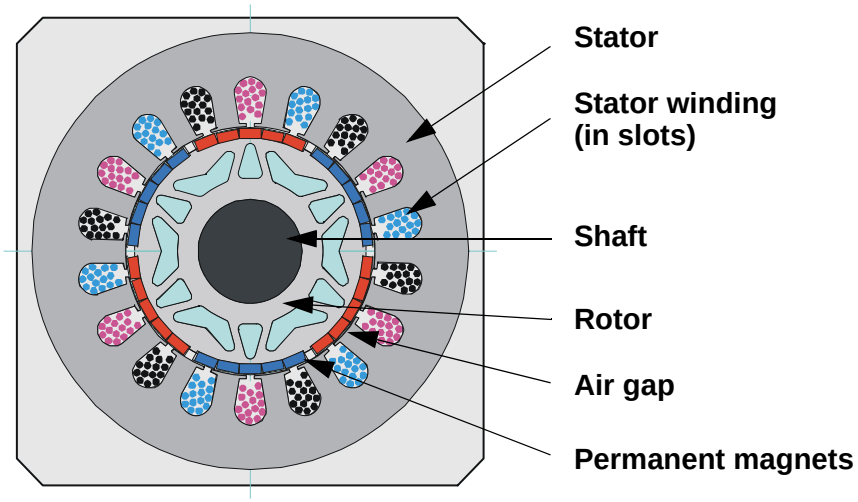

Figure 2-1. PM Synchronous Motor — Cross Section

In this respect, the PM synchronous motor is equivalent to an induction motor, where the air gap magnetic field is produced by a permanent magnet, so the rotor magnetic field is constant. PM synchronous motors offer a number of advantages when used in modern motion control systems. 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 can be obtained by using space vector theory. The three-phase motor quantities (such as voltages, currents, magnetic flux, etc.) are expressed in terms of complex space vectors. Such a model is valid for any instantaneous variation of voltage and current, and 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. We can look at the motor 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

Let's assume i_{sa} , i_{sb} , and i_{sc} are the instantaneous balanced three-phase stator currents:

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

Then we can define the stator current space vector as:

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

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

The space vector defined by Equation 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, $i_{s\beta}$. Thus, the stator current space vector in the stationary reference frame attached to the stator can be expressed as:

$$\bar{i}_s = i_{s\alpha} + j i_{s\beta} \qquad \textit{Eqn. 2-3}$$

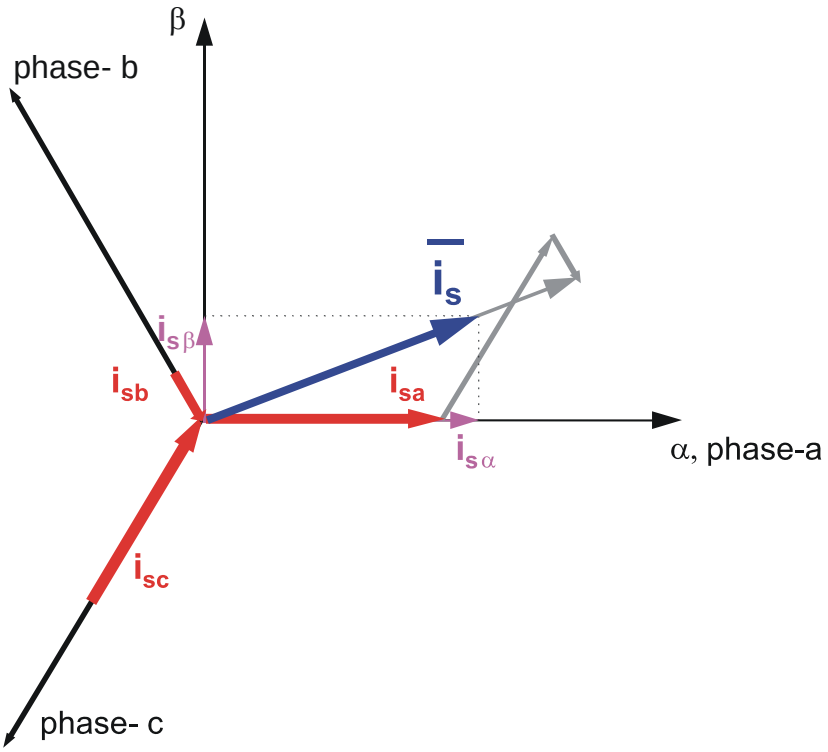

Figure 2-2. Stator Current Space Vector and Its Projection

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

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

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

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

The space vectors of other motor quantities (voltages, currents, magnetic fluxes, etc.) 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 considered. The voltage equations of stator in the instantaneous form can then be expressed as:

$$u_{SA} = R_S i_{SA} + \frac{d}{dt}\psi_{SA}$$ **Eqn. 2-6**

$$u_{SB} = R_S i_{SB} + \frac{d}{dt}\psi_{SB}$$ **Eqn. 2-7**

$$u_{SC} = R_S i_{SC} + \frac{d}{dt}\psi_{SC}$$ **Eqn. 2-8**

where 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, and ψ_{SA}, ψ_{SB}, ψ_{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 — the equations Equation 2-6, Equation 2-7, and Equation 2-8 — it is more practical to rewrite the instantaneous equations using two-axis theory (Clarke transformation). The PM synchronous motor can be expressed as:

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

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

$$\Psi_{S\alpha} = L_S i_{S\alpha} + \Psi_M \cos(\theta_r)$$ **Eqn. 2-11**

$$\Psi_{S\beta} = L_S i_{S\beta} + \Psi_M \sin(\theta_r)$$ **Eqn. 2-12**

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

For a glossary of symbols see Section 1.3.3, “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, which rotates at a general speed ω_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, where θ_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, then Equation 2-14 defines the stator current space vector in a general reference frame:

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

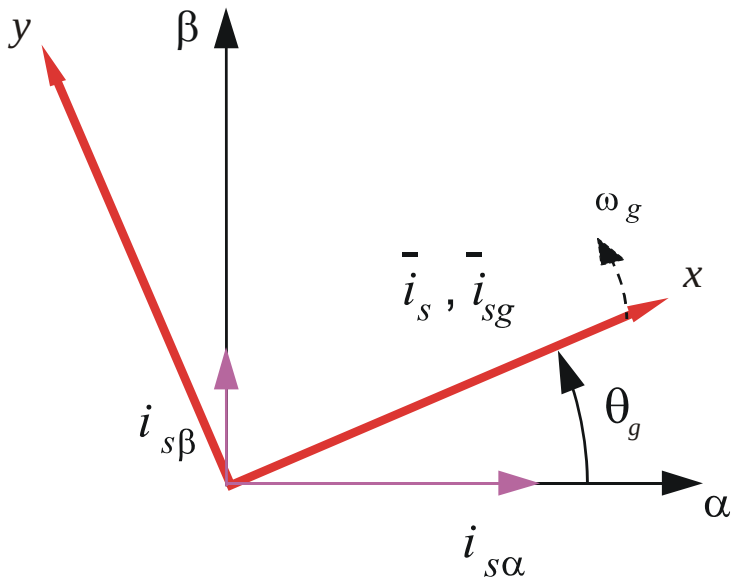

Figure 2-3. Application of the General Reference Frame

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 (rα) of the reference frame attached to the rotor is displaced from the direct axis of the stator reference frame by the rotor angle θr. Because it can be seen 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 (rα) is θg–θr, in the general reference frame, the space vector of the rotor currents can be expressed as:

$$\overline{i_{rg}} = \bar{i}_r e^{-j(\theta_g - \theta_r)} = i_{rx} + j i_{ry} \qquad \textit{Eqn. 2-15}$$

where $\bar{i}_r$ 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 can be 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 a direct axis (d) and a quadrature axis (q).

After transformation into d, q coordinates, the motor model is:

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

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

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

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

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

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

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

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

2.3 Vector Control of PM Synchronous Motor

2.3.1 Fundamental Principle of Vector Control

High-performance motor control is characterized by smooth rotation over the entire speed range of the motor, full torque control at zero speed, and fast acceleration/deceleration. To achieve such control, vector control techniques are used for three-phase AC motors. The vector control techniques are usually 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 that for a separately excited 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 perform 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.

Also keep these points in mind:

- The stator current torque (i_{sq}) and flux (i_{sd}) producing components are separately controlled.
- The output stator voltage space vector is calculated using the decoupling block.
- 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.
- The output three-phase voltage is generated using space vector modulation.

To be able to decompose currents into torque and flux producing components (i_{sd}, i_{sq}), we need to know the position of the motor-magnetizing flux. This requires accurate sensing of the rotor position and

velocity. 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 is not desirable. In these applications the aim is not to measure the speed/position directly, but to employ some indirect techniques to estimate the rotor position instead. Algorithms that do not employ speed sensors are called “sensorless control.”

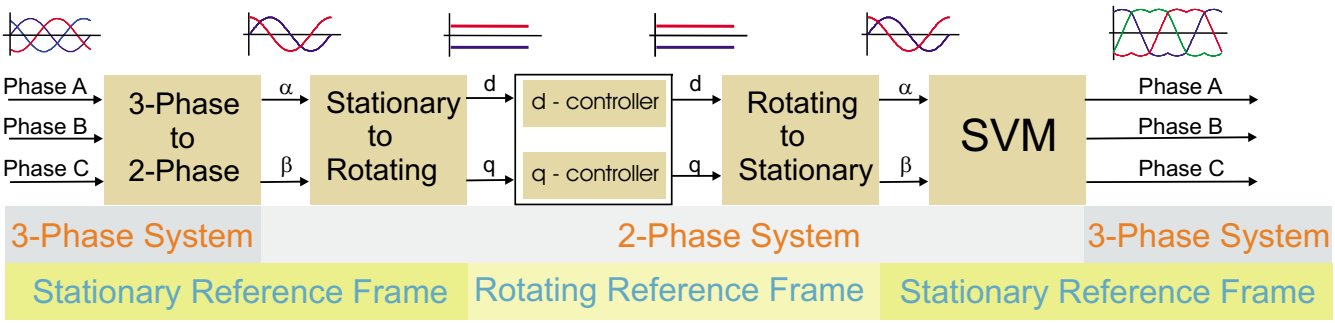

Figure 2-4. Vector Control Transformations

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 possible 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 1 ms.

To achieve the goal of the PM synchronous motor control, the algorithm uses feedback signals. The essential feedback signals are three-phase stator current and stator voltage. For the stator voltage, the regulator output is used. For correct operation, the presented control structure requires a speed sensor on the motor shaft. In the case of the presented algorithm, an incremental encoder is used.

The fast control loop executes two independent current control loops. They are the direct and quadrature-axis current (i_{sd},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 summed with the corresponding d and q axis components of the decoupling stator voltage. Thus we obtain the desired space vector for the stator voltage, which is 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. These include:

- Three-phase current reconstruction
- Forward Clark transformation
- Forward and backward Park transformations
- Rotor magnetizing flux position evaluation
- DC-bus voltage ripple elimination
- Space vector modulation (SVM)

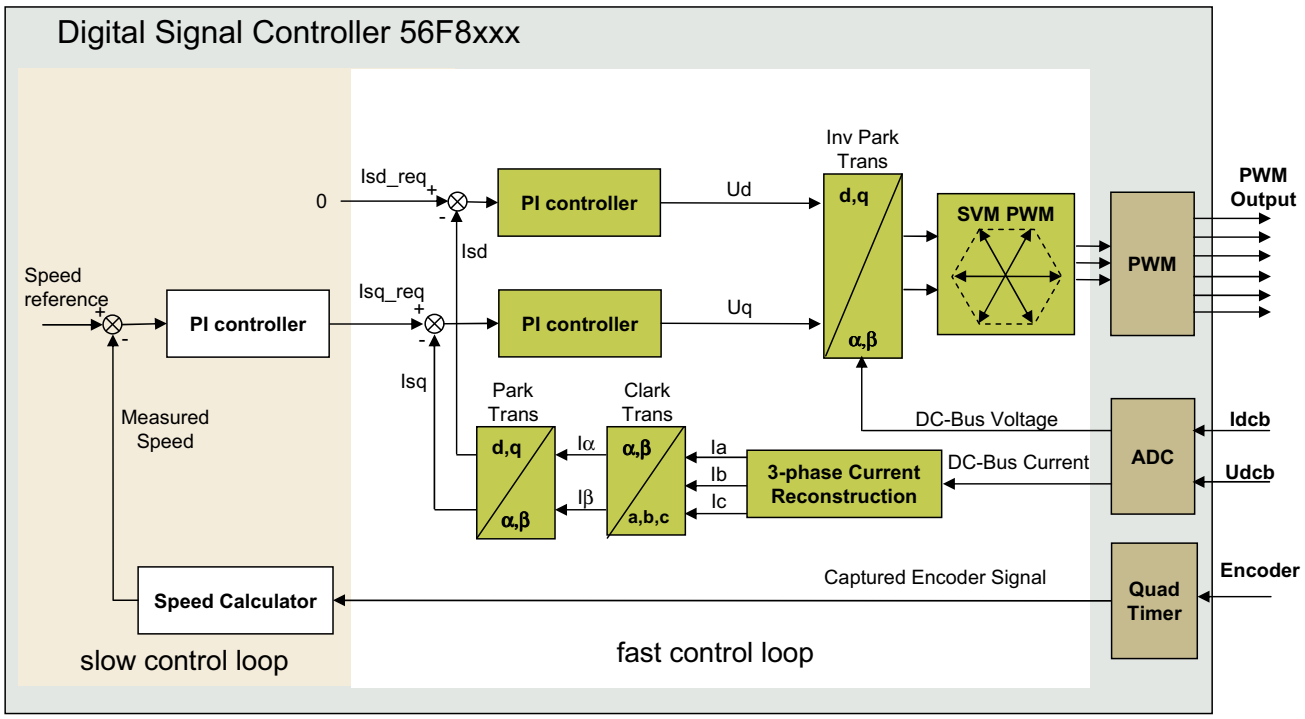

Figure 2-5. PMSM Vector Control Algorithm Overview

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 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} also depends on i_{sq}, and the quadrature axis component u_{sq} also 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, so these 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 $u_{sd}^{decouple}, u_{sq}^{decouple}$. The equations are decoupled in this way:

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

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

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 these equations:

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

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

2.3.4 Space Vector Modulation

Space vector modulation (SVM) can directly transform the stator voltage vectors from the two-phase α,β-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 α,β-coordinate system).

The basic principle of the standard space vector modulation technique can be explained with the help of the power stage schematic diagram given 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.A graphical representation of all combinations is the hexagon shown in Figure 2-7. 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}, defined in α,β coordinates.

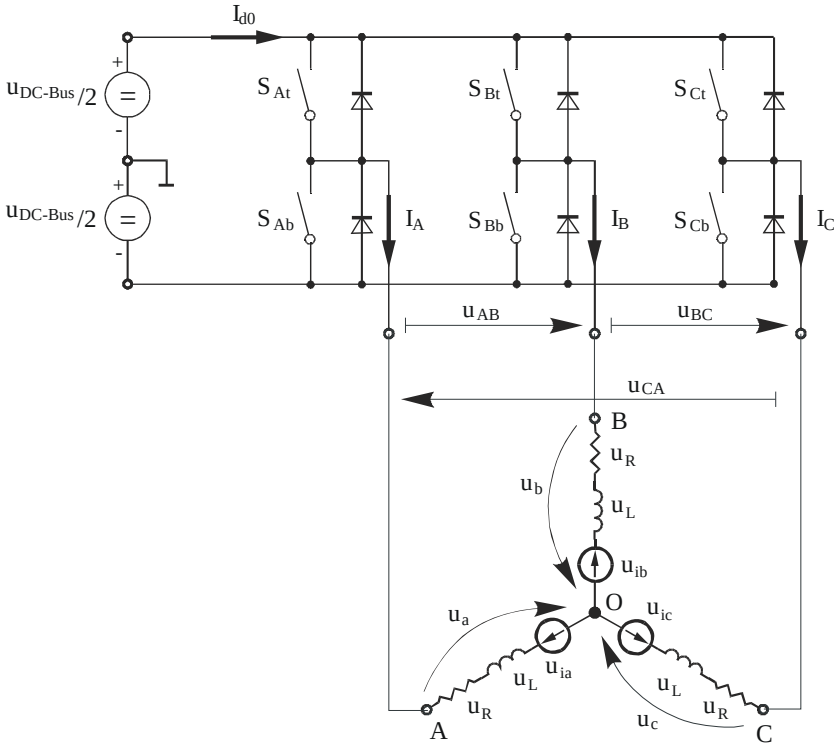

Figure 2-6. Power Stage Schematic Diagram

The combination of ON/OFF states in the power stage switches for each voltage vector is coded in Figure 2-7 by the three-digit number in parentheses. 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	O_{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	O_{111}

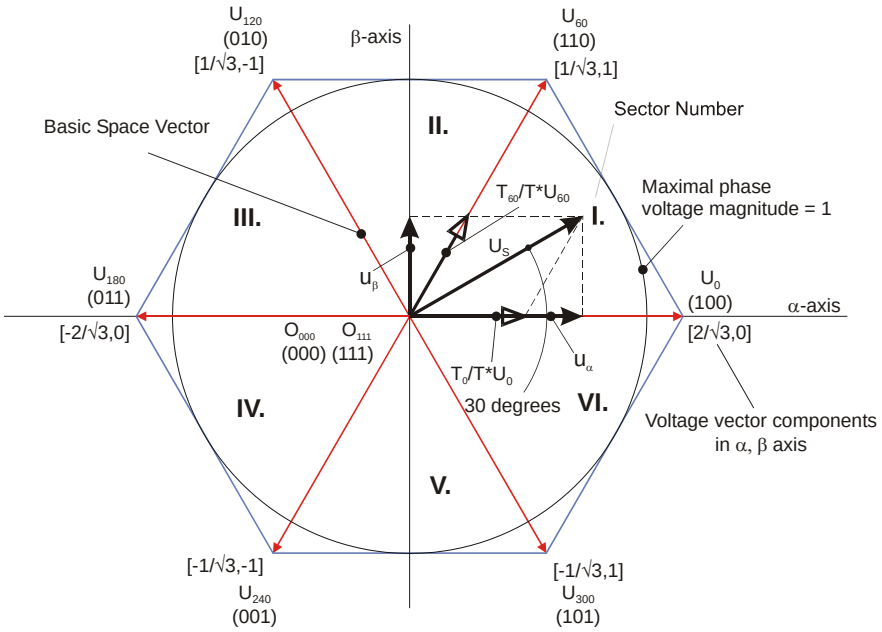

Figure 2-7. Basic Space Vectors and Voltage Vector Projection

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

The SVM technique consists of several steps:

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

The principle of SVM is the application of the voltage vectors U_{XXX} and O_{XXX} for certain instances in such a way that the mean vector of the PWM period t_{PWM} 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. One can arrange these vectors to lower switching losses; another might want to reach a different result, such as center-aligned PWM, edge-aligned PWM, minimal switching, etc.

For the chosen SVM, we define this rule:

- 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 expressions define the principle of the SVM:

$$t_{PWM} \times U_{S[\alpha,\beta]} = t_1 \times U_x + t_2 \times (U_{x\pm60} + t_0 \times (O_{000} \vee O_{111})) \quad \textit{Eqn. 2-26}$$

$$t_{PWM} = t_1 + t_2 + t_0 \quad \textit{Eqn. 2-27}$$

In order 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, $U_{x\pm60}$. The Equation 2-26 splits into equations Equation 2-28 and Equation 2-29.

$$t_{PWM} \times U_{SX} = t_1 \times U_x \quad \textit{Eqn. 2-28}$$

$$t_{PWM} \times U_{S(x\pm60)} = t_2 \times U_{x\pm60} \quad \textit{Eqn. 2-29}$$

By solving this set of equations, we can calculate the necessary duration for the application of the sector base vectors U_x, $U_{x\pm60}$ during the PWM period T_{PWM} to produce the right stator voltages.

$$t_1 = \frac{|U_{SX}|}{|U_x|} t_{PWM} \quad \textbf{for vector } U_x \quad \textit{Eqn. 2-30}$$

$$t_2 = \frac{|U_{SX}|}{|U_{x\pm60}|} t_{PWM} \quad \textbf{for vector } U_{x\pm60} \quad \textit{Eqn. 2-31}$$

$$t_0 = t_{PWM} - (t_1 + t_2) \quad \textbf{either for } O_{000} \textbf{ or } O_{111} \quad \textit{Eqn. 2-32}$$

2.3.5 Motor Position Alignment

In this design, the quadrature encoder is used as a sensor for motor position and speed. Because the quadrature encoder doesn't give the absolute position, before the motor is started we need to know the rotor position exactly. One possible, and very easily implemented, method is to align the rotor to a predefined position. The motor is powered by a selected static voltage pattern (usually the zero position in the sine wave table) and the rotor aligns to the predefined position. The alignment is done only once during first motor start. Figure 2-8 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. In practice, this is true when the external load torque is low enough compared to the torque produced by the alignment vector.

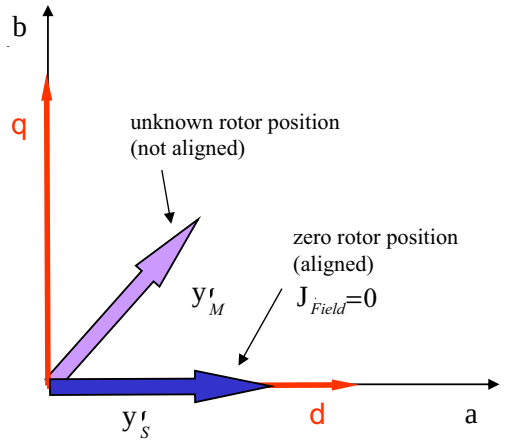

Figure 2-8. Rotor Alignment Stabilization — PMSM Starting Mode

Chapter 3
System Concept

3.1 System Specifications

The system is designed to drive a three-phase PM synchronous motor. The application meets these performance specifications:

- Targeted at the MC56F8013/23 digital signal controller
- Running on the MC56F8013/23 controller board and three-phase high voltage power stage
- Control technique incorporating:
  - Vector control of three-phase PM synchronous motor with quadrature encoder
  - Closed-loop speed control
  - Bidirectional rotation
  - Both motor and generator modes
  - Closed-loop current control
  - Independent control of flux and of torque
  - Startup with alignment
  - Reconstruction of three-phase motor currents from DC-link shunt resistor
  - 125 µs sampling period on MC56F8023 with FreeMASTER recorder
- FreeMASTER software control interface (motor start/stop, speed setup)
- FreeMASTER software monitor
  - FreeMASTER software graphical control page (required speed, actual motor speed, start/stop status, DC-bus voltage level, motor current, 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, vector control algorithm quantities)
- DC-bus over-voltage and under-voltage, over-current protection

3.2 Application Description

A standard system concept is chosen for the drive (see Figure 3-1). The system incorporates these hardware boards:

- Power supply 90 V–260 V AC RMS, 5 A
- Three-phase, high-voltage power stage
- Three-phase PM synchronous motor (default configuration for TG Drives 3-phase PMSM motor)

- MC56F8013/23 controller board

The MC56F8023 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.

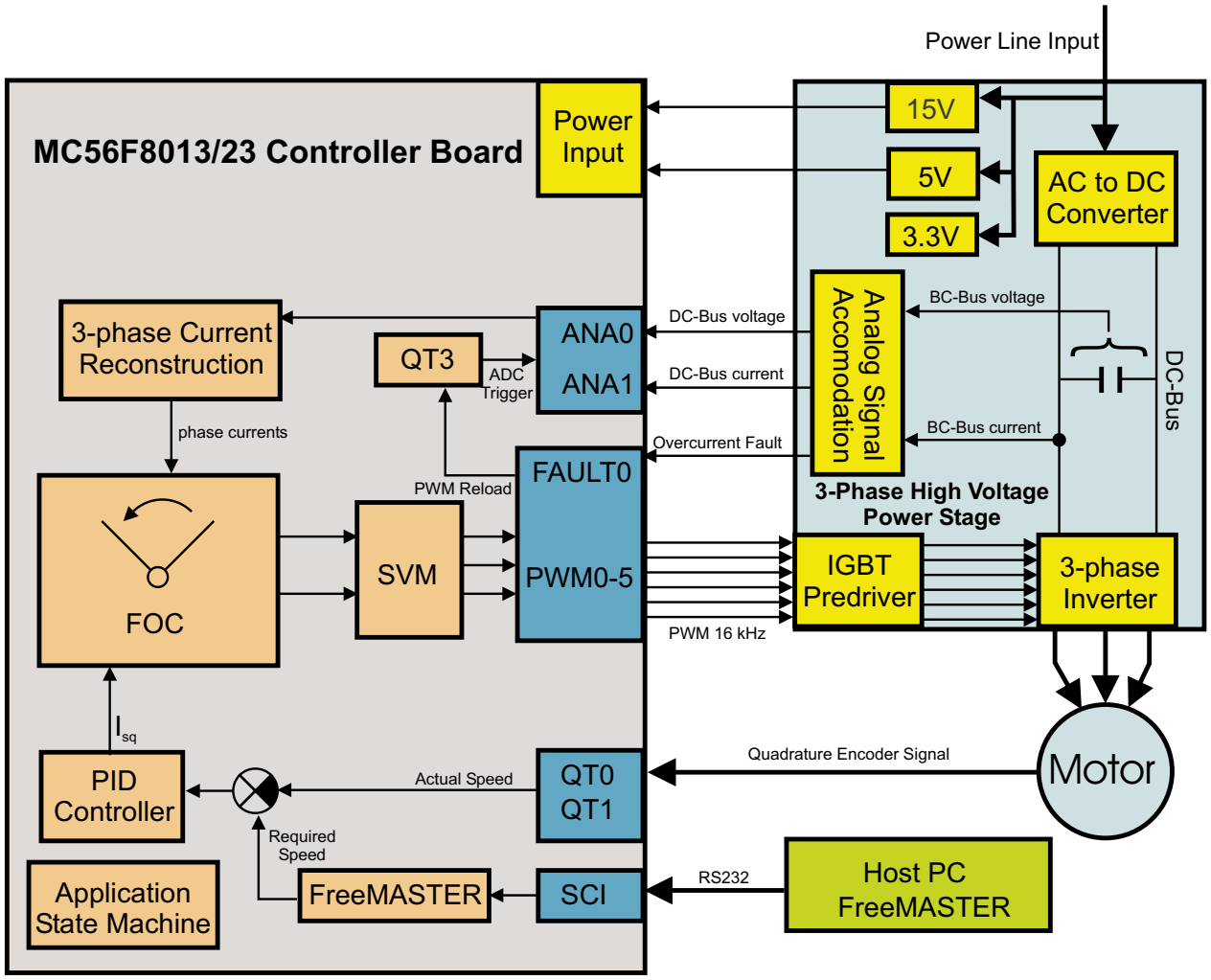

Figure 3-1. System Concept

3.3 Control Process

The state of the user interface is scanned periodically, while the actual speed of the motor, DC-bus voltage, and the phase currents are sampled. The speed command is calculated according to the state of the control signals (Start/Stop, Required Speed from FreeMASTER). Then the speed command is 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 are sampled with ADC. The ADC sampling is triggered by QuadTimer channel 3 and synchronized to the PWM signal. A digital filter is applied to the sampled values. The three-phase motor current is reconstructed from samples taken from the DC-link shunt resistor. The reconstructed three-phase current is then transformed into space vectors and used by the FOC algorithm.

The rotor position and speed are sensed by the quadrature encoder. Based on measured feedback signals, the FOC algorithm performs a vector control technique oriented to the rotor-magnetizing flux space vector, as described in Section 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, which is transformed by means of space vector modulation into PWM signals. Three-phase stator voltage is generated by means of a three-phase voltage source inverter and applied to the motor, which is 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 RS-232 protocol. The RS-232 is opto-isolated to achieve safety isolation from high voltage.

The application state machine of the drive manages the operating states of the drive. There are five states of the drive. 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 is designed to drive a three-phase PM synchronous motor. It consists of these 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.

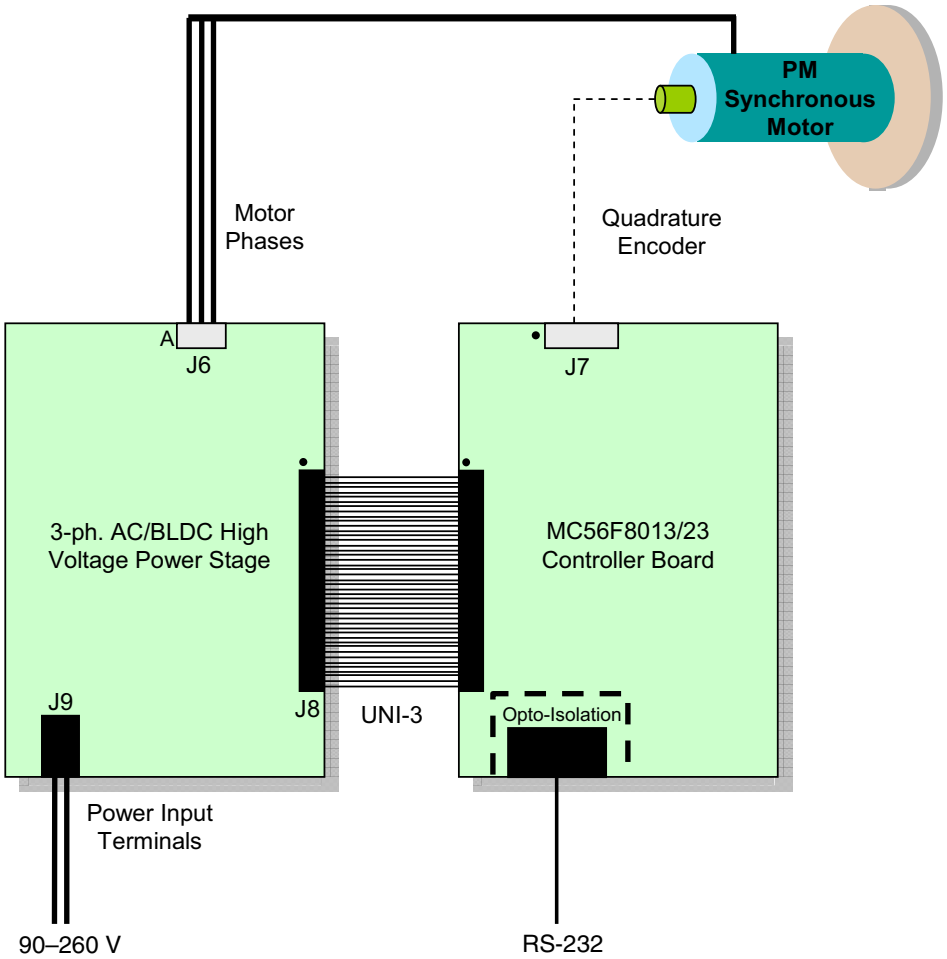

Figure 4-1. Hardware System Configuration

All system parts are supplied and documented:

- MC56F8013/23 controller board:
  - Uses Freescale's MC56F8013 or MC56F8023 as the controller
  - Described in the *MC56F8013 Controller Board Hardware User's Manual*
- Three-phase AC/BLDC high-voltage power stage:
  - High-voltage, three-phase power stage with single-phase input 115/230 V AC and 750 VA variable voltage three-phase IGBT bridge output
  - Described in the *3-Phase AC/BLDC High Voltage Power Stage Board Users Guide*

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 MC56F8013/23 Controller Board

The MC56F8013/23 controller board is based on an optimized PCB and power supply design. It demonstrates the abilities of the MC56F8013/23 and provides a hardware tool to help in the development of applications that use the MC56F8013/23 and are targeted at motor control applications.

The MC56F8013/23 controller board can be populated either by 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 analog power supplies, and peripheral expansion connectors. The expansion connectors are for signal monitoring and to allow expandability of user features. Test pads are provided for monitoring critical signals and voltage levels.

The MC56F8013/23 controller board is designed for these purposes:

- New users can become familiar with the features of the MC56F801x/802x architecture.
- The board can serve as a platform for real-time software development.
  The tool suite allows you 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 let you specify complex break conditions easily and execute your 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 considerably simplifies 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 MC56F8013/8023, and can be used to develop real-time software and hardware products based on either device. 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).

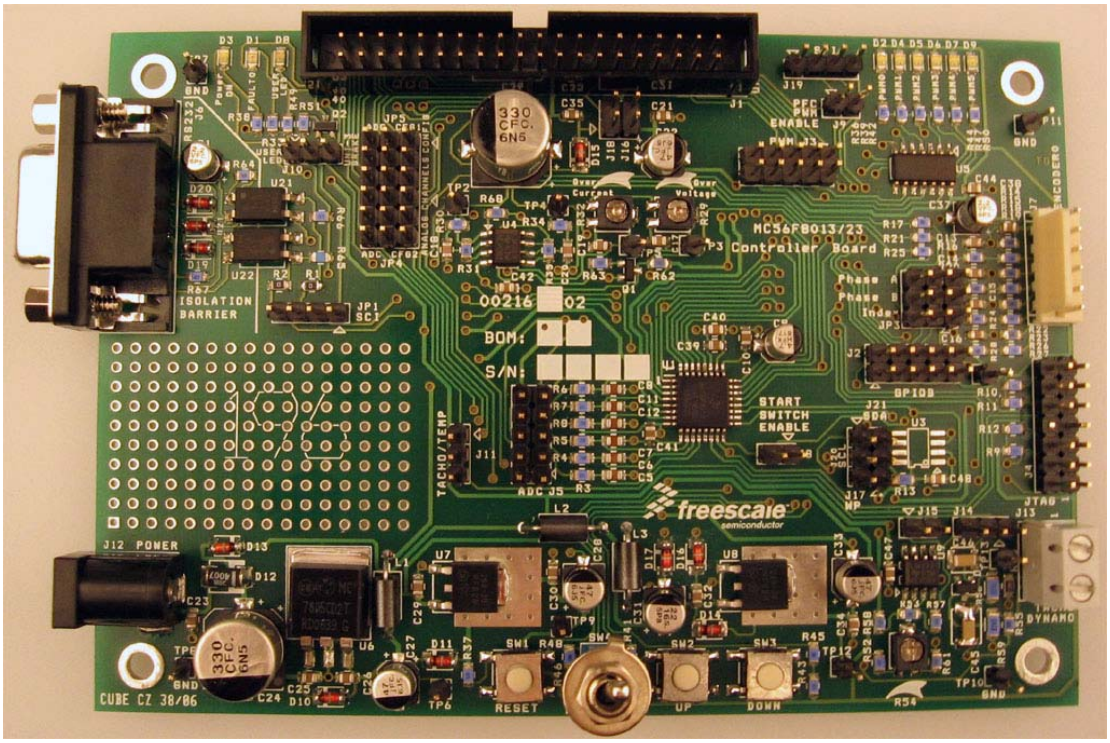

Figure 4-2. MC56F8013/23 Controller Board Top View

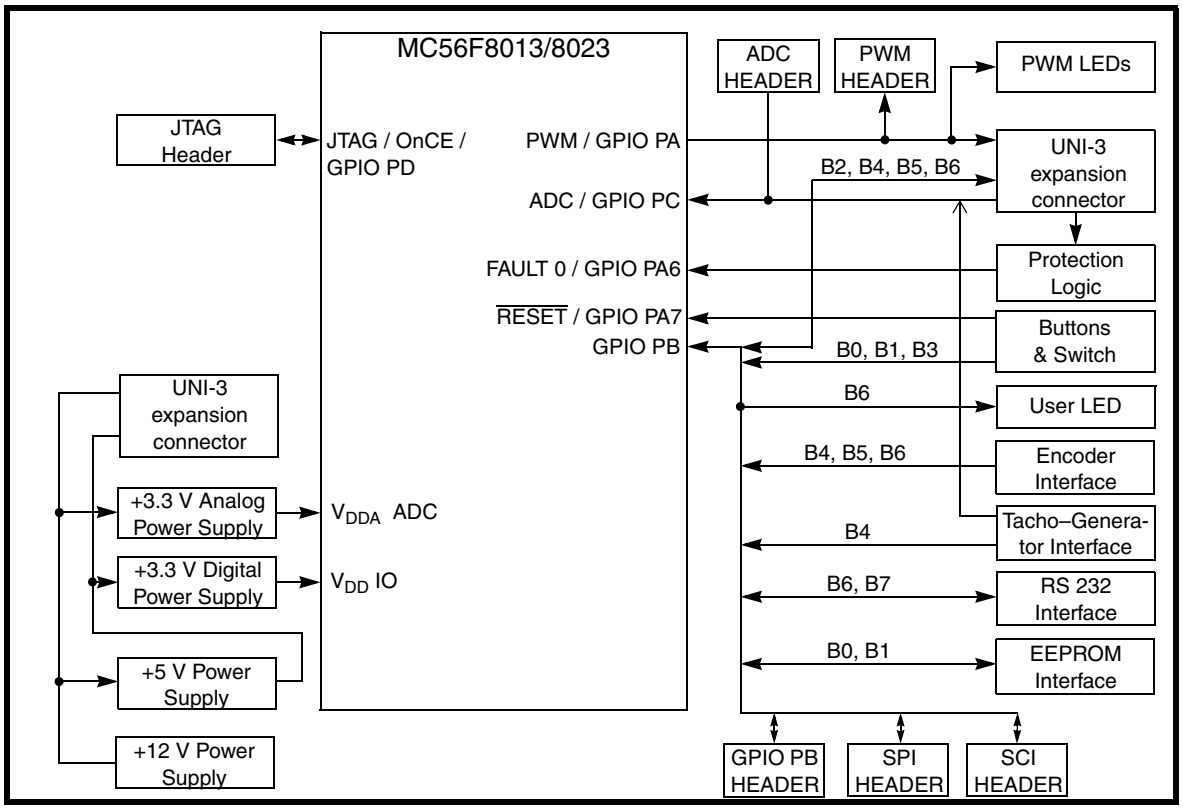

Figure 4-3. Block Diagram of the MC56F8013/23 Controller Board

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 will operate off DC input voltages from 140 to 325 volts, and AC line voltages from 100 to 240 V. 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, analog 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 underneath the printed circuit board (see Figure 4-4).

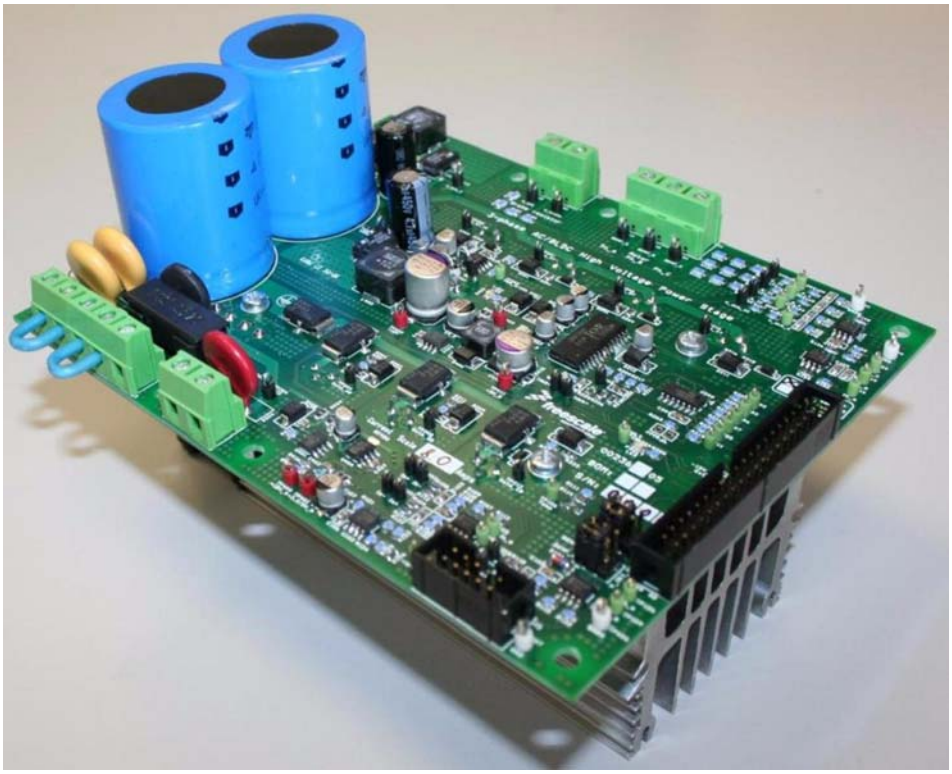

Figure 4-4. Three-Phase AC/BLDC High Voltage Power Stage

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 labelled 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 4 or 8 amps full scale. Both bus and phase leg currents are measured. An over-current trip point is set at 10 amps.

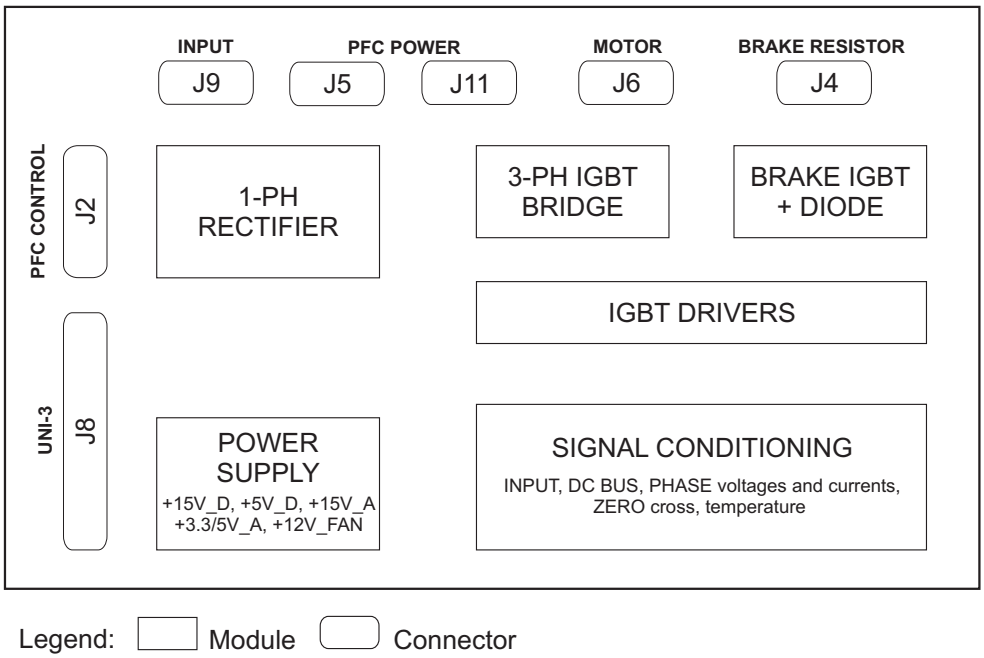

Figure 4-5. Three-Phase AC/BLDC Power Stage Block Diagram

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

Characteristic	Symbol	Min	Typ	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
Analog 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
Continuous Output Current**	I_C	—	—	10	A
Deadtime (built in IR2133)	t_{Off}	—	250	—	ns

* Range set according +3.3 VA / +5 VA 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 these specifications:

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

	Motor Type	**Three-Phase PM Synchronous Motor TG drives TGT3-0130-30-320/T0KX-1.0M**
Motor Specification	Nominal Voltage (line-to-line)	340 V RMS
	Nominal Speed	3000 RPM
	Nominal Current (phase)	1.58 A RMS
	Nominal Torque	1.15 Nm
Motor Model Parameters	Stator Winding Resistance	6.25 Ohm
	Stator Winding Inductance d axis	11.1 mH
	Stator Winding Inductance q axis	12.5 mH
	Number of Pole-Pairs	3
Position Sensor Specification	Manufacturer	INDUcoder
	Type	ES 28-6-1024-05-D-R
	Line Count	1024
	Output	5 V ± 10% TTL

Chapter 5
Software Design

5.1 Introduction

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

5.2 Scaling of Application Variables

5.2.1 Fractional Numbers Representation

The 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) lie in the following range:

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

For words and long-word signed fractions, the greatest negative number that can be represented is –1.0, whose internal representation is $8000 and $80000000, respectively. The greatest positive word is $7FFF or $1.0 - 2^{-15}$, and the greatest positive long word is $7FFFFFFF or $1.0 - 2^{-31}$.

5.2.2 Scaling of Analog Quantities

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

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

where:

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

As an example, the above scaling can be demonstrated on DC-bus voltage and motor phase voltage. All variables representing voltage are scaled the same in the application. They are scaled to the 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:

$$(\text{Frac16})\text{voltage\_variable} = \frac{V_{Measured}}{V_{max}} \quad \textit{Eqn. 5-3}$$

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

$$(\text{Int16})\text{voltage\_variable} = (\text{Frac16})(\text{voltage\_variable} \times 2^{15}) \quad \textit{Eqn. 5-4}$$

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

```
#define XLS_PMSM_U_MAX              235                     /* Volts */
#define XLS_PMSM_I_MAX              2*4.0                   /* Amps */
```

Please note that XLS_PMSM_I_MAX corresponds to a full range of the ADC converter input voltage (0 – 3.3V). For motor phase-current sensing, the zero current level is shifted into the middle of this range (= 1.65 V). Thus, the maximum positive and negative phase current which can be sensed is Max_Current = XLS_PMSM_I_MAX/2. In other words, if the current sensing range of the power stage is from –4.0 A to + 4.0 A, the value of XLS_PMSM_I_MAX is set to the value 8.0 A. And if the current sensing range of the power stage is from –8.0 A to + 8.0 A, the value of XLS_PMSM_I_MAX is set to the value 16.0 A.

5.2.3 Scaling of Angles

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

$$-\text{pi} \approx 0\text{x}8000 \quad \textit{Eqn. 5-5}$$

$$\text{pi} \times (1.0 - 2^{-15}) \approx 0\text{x}7\text{FFF} \quad \textit{Eqn. 5-6}$$

5.2.4 Scaling of Parameters

Real-value parameters, in equations such as the SM observer, decoupling voltage, etc., are represented as 16-bit signed fractional values in the range [–1,1). The real parameter value (ohms, henries) has to be adjusted to correspond to the scaling range of the analog value, which 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 a simple example of the Ohm's law equation.

$$V_{Real} = R \times I_{Real} \quad \textit{Eqn. 5-7}$$

$$V_{Frac16} \times \text{V\_MAX} = R \times I_{Frac16} \times \text{I\_MAX} \quad \textit{Eqn. 5-8}$$

$$V_{Frac16} = \left(R \times \frac{\text{I\_MAX}}{\text{V\_MAX}}\right) \times I_{Frac16} = R_{Adjusted} \times I_{Frac16} \quad \textit{Eqn. 5-9}$$

Let's substitute the following values:

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

The $R_{Adjusted}$ can be evaluated:

$$R_{Adjusted} = R \times \frac{I\_MAX}{V\_MAX} = 300 \times \frac{8}{407} = 5.8968 \quad \textit{Eqn. 5-10}$$

The $R_{Adjusted}$ is out of range of the signed fractional number. We need to right shift the value by R_{Scale}-bits to fit into the desired range. Therefore, we introduce a scale (shift) part of the parameter. For this example we have to shift the result by R_{Scale} = 3 bits. The resistor value scaled to the signed fractional range is as follows:

$$R_{Frac16} \times 2^{R_{Scale}} = R_{Adjusted} \quad \textit{Eqn. 5-11}$$

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

The Ohm's law equation scaled into signed fractional arithmetic is evaluated as follows:

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

Please note, 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-bit 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. Let's show the encoder position scale parameter as an example:

```
#define ENC_POSITION_SCALE_SHIFT    -6
#define ENC_POSITION_SCALE          FRAC16(0.75)
```

The N-bit shift constant can be defined as both negative and positive. If negative, a left shift is applied to the scaled variable. Then the real value of the encoder position scale is:

$$cPositionScaleReal = \frac{EncPositioScale}{2^{EncPositionScaleShift}} = \frac{0.75}{2^{-6}} = 48 \quad \textit{Eqn. 5-14}$$

To evaluate the fractional and shift constants of the parameters, use the Excel spreadsheet included in the software files.

We also introduce several scaling factors for some of the complicated constant calculations (in the Excel spreadsheet included with the software). Their purpose is to transform integer constants into the defined fractional range [–1, 1) and vice versa. With these scaling factors, some of the equations can be made universal, and can thus potentially be reused for system quantities that are not represented by the fractional range [–1,1).

The voltage scaling factor is:

$$S_V = \frac{V_{sysrange}}{V_{max}} \quad \textit{Eqn. 5-15}$$

where:

- S_V is the scaling factor

- $V_{sysrange}$ is the range of system representation voltage
- V_{max} is the range of real voltage

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

$$S_V = \frac{1}{V_{max}} \quad \textit{Eqn. 5-16}$$

Then the real variable is:

$$V_{Real} = \frac{(Frac16)Voltage\_Variable}{S_V} \quad \textit{Eqn. 5-17}$$

The system parameters are then calculated accordingly:

$$R_{Adjusted} = R \times \frac{S_V}{S_I} = 300 \times \frac{1/407}{1/8} = 5.8968 \quad \textit{Eqn. 5-18}$$

5.3 Application Overview

The application software is interrupt-driven when 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 every second PWM reload, with a 125 µs period (TS_FAST). It performs a fast current control loop.

The ADC End of Scan interrupt service routine is executed for three consequential sample readings within one PWM cycle. It reads the DC-link current samples.

The PWM Fault interrupt service routine is executed on an over-current event to handle 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 application main(). It handles non-critical timing tasks, such as the application state machine and FreeMASTER communication polling.

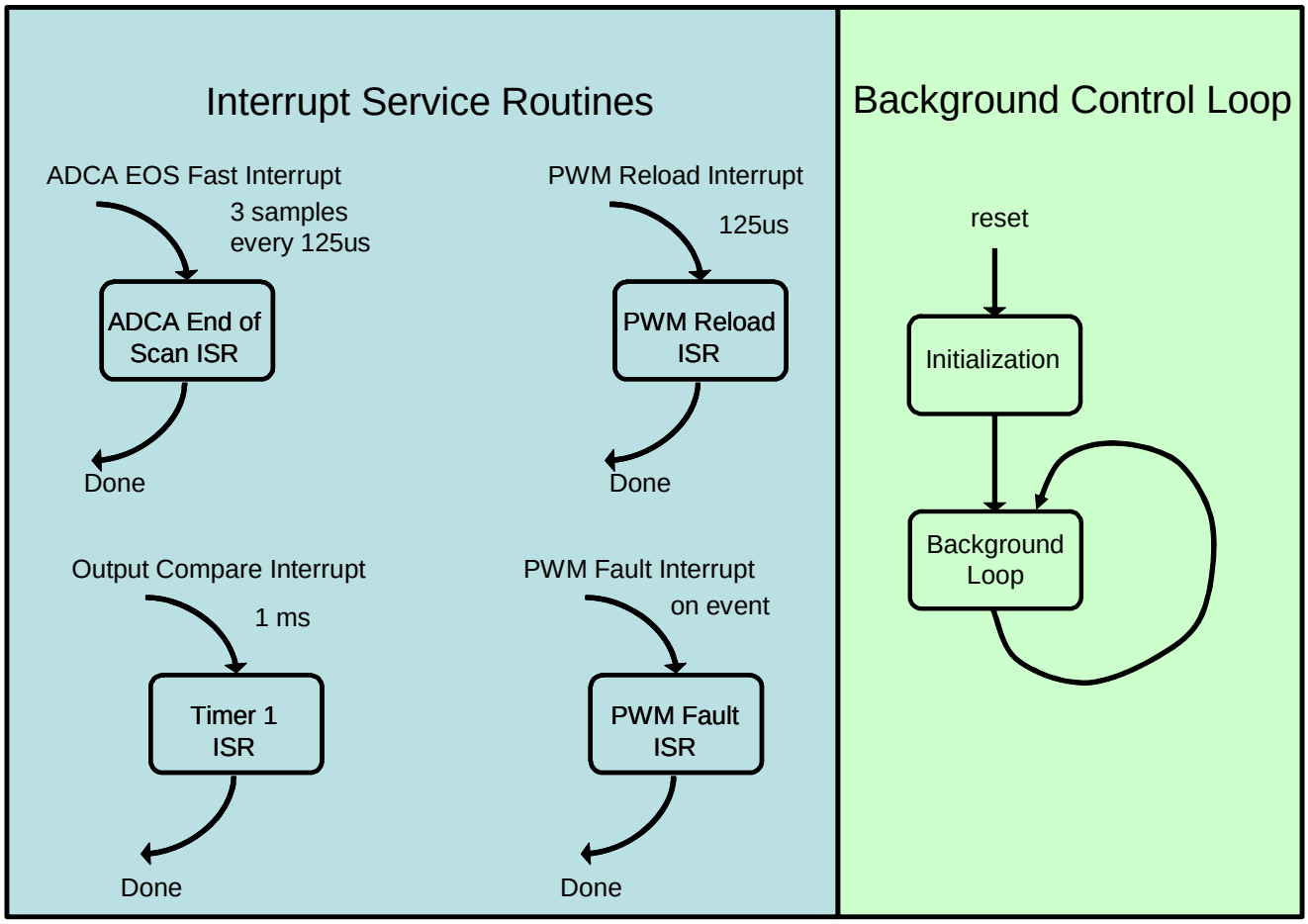

Figure 5-1. Main Data Flow

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

5.3.1 ADC End-of-Scan Timing and PWM Reload Interrupts

The fast current control loop is executed in the PWM reload ISR, which is synchronized to the PWM_reload_sync signal. Prior to the PWM reload ISR being executed, three ADC samples of DC-link current are taken and processed by the ADC end of scan ISR. After ADC sampling is finished the PWM reload ISR is enabled and executed.

The PWM module is configured to run in center-aligned mode with counter modulo CMOD = 800, which corresponds to 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, which carries the signal of the TMR module. An output of the TMR channel 3 is connected to the SYNC0 signal, which is used to trigger the ADCA in simultaneous mode. The TMR channel 3 is configured in triggered count mode. A connection link between the PWM module, TMR module, and ADC module enables defining the exact multiple time instants of ADC sampling, which are synchronized to the generated PWM signal. An overview of module interconnections is shown in Figure 5-2.

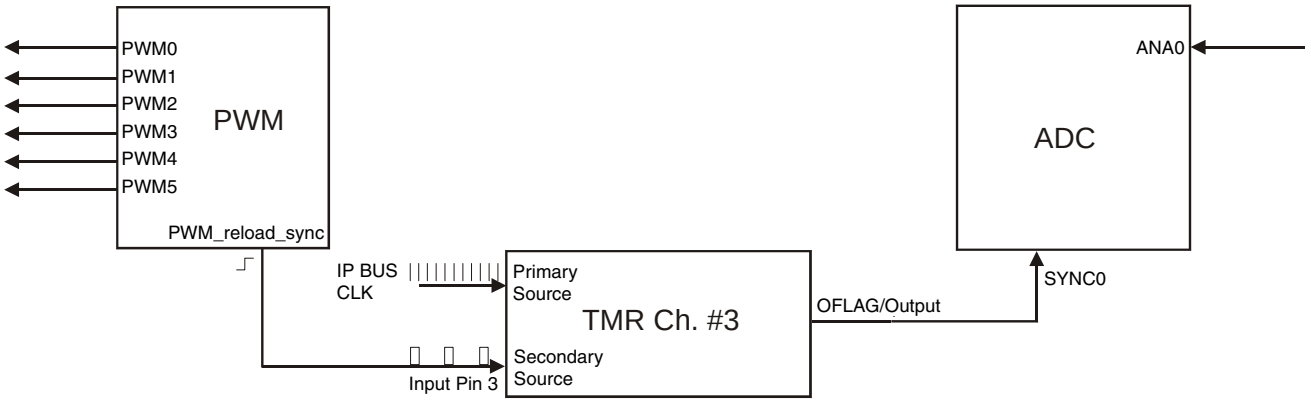

Figure 5-2. ADC Triggering

The timing diagram in Figure 5-3 shows how a triple-triggered ADC conversion is performed. The events are executed in the following steps, which are numbered in the figure.

1. A PWM counter reaches zero. A PWM reload occurs. Values stored in the PWM value registers (VAL0–5) are applied to the PWM outputs. PWM reload flag (PWMF) is set to one and pending. The PWM reload interrupt is disabled at the beginning of every PWM reload cycle. The signal PWM_reload_sync is generated by the PWM module.
2. TMR channel 3 count is triggered by the PWM_reload_sync signal, which is connected to its secondary source input. The timer starts counting up from zero.
3. A compare on Compare 1 register (COMP1) occurs. An output (OFLAG) of the TMR channel 3 is set to one. A value from Comparator Load 1 register (CMPLD1) is loaded into the COMP1 register.
4. A rising edge on the SYNC0 input of the ADC module starts an ADC conversion.
5. The ADC conversion is finished. The ADC end of scan 1 flag (EOSI1) is set.
6. The ADC end of scan ISR is entered. The interrupt is processed as a fast interrupt with a priority level 2. A value from Result 0 (RSLT0) register is stored in a buffer. A new value stored in a timing table is loaded into the CMPLD1 register of TMR channel 3. An output (OFLAG) of the TMR channel 3 is forced to zero. EOSI1 flag is cleared.
7. The second compare on Compare 1 (COMP1) occurs. Steps 3 through 6 are repeated. Before ADC EOS ISR is exited, the TMR channel 3 is stopped, the ADC EOS interrupt is disabled, and the PWM reload interrupt is enabled.
8. Once PWM reload interrupt is enabled, the pending PWMF flag generates an interrupt and the PWM reload ISR is entered. The PWM reload ISR performs routines for the fast current control loop. When finished, the new values are stored in the PWM value registers (VAL0–5). The TMR channel 3 registers COMP1 and CMPLD1 are loaded with these new values and the counter is reset. The PWM Reload interrupt is disabled.

When a new PWM reload event occurs, the sequence starts again at step one.

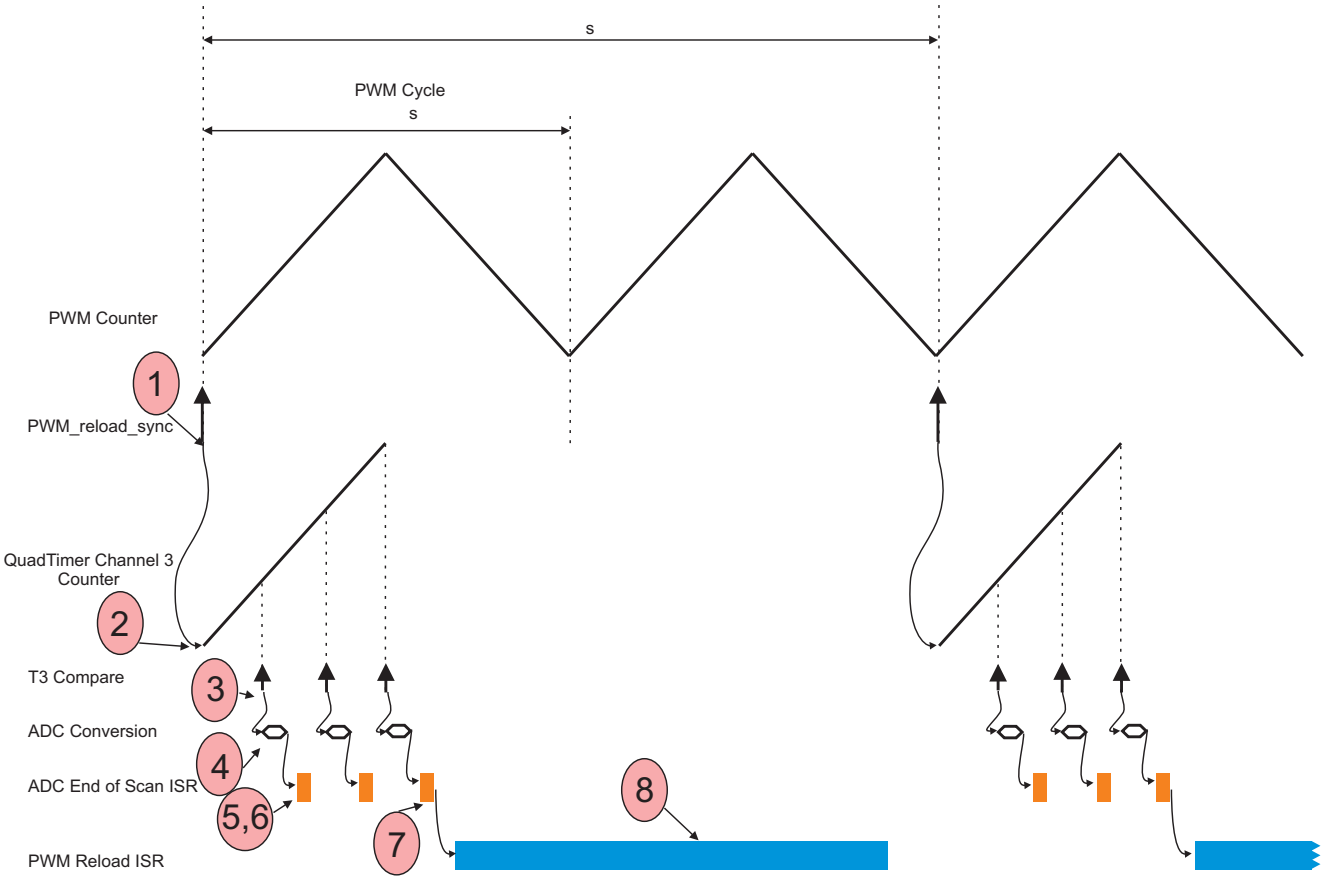

Figure 5-3. ADC End Of Scan and PWM Reload Interrupts

The triple-triggered ADC sampling is used to perform a reconstruction of the three-phase motor current from the single DC-link shunt resistor. The reconstruction algorithm is described in the following section.

5.3.2 Three-Phase Current Reconstruction

The vector control algorithm requires the sensing of the three motor phase currents. A standard approach is to sense the phase currents directly through current transformers, or Hall effect sensors, directly coupled to the motor phase lines that carry the current between the switches and the motor. To reduce the number of current sensors and overall cost of the design, the three-phase stator currents are measured by means of a single DC-link current shunt sensor; see Figure 5-4. The DC-link current pulses are sampled at exactly timed intervals. A voltage drop on the shunt resistor is amplified by an operational amplifier inside the three-phase driver and shifted up by 1.65 V. The resultant voltage is converted by the ADC; see Figure 5-5.

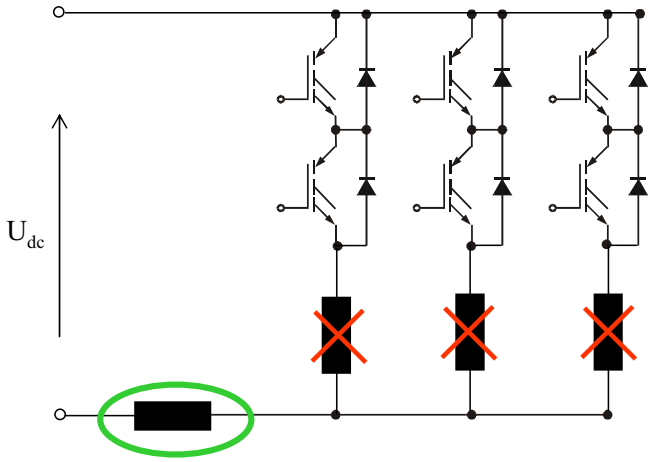

Figure 5-4. DC-Link Current Shunt

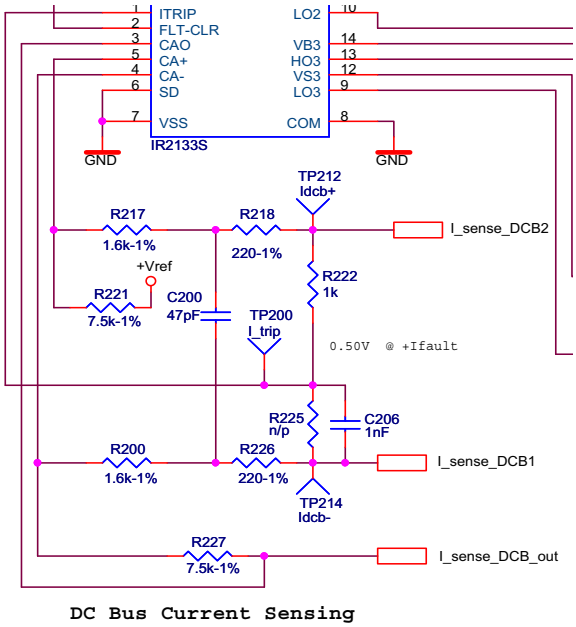

Figure 5-5. Current Amplifier for DC-Link Current

Based on the actual combination of switches, the three-phase currents of the stator are reconstructed. The AD converter measures the DC-link current during the active vectors of the PWM cycle. When the voltage vector V1 is applied, current flows from the positive rail into the phase A winding and returns to the negative rail through the B and C phase windings. When the voltage vector V2 is applied, the DC link current returning to the negative rail equals the T phase current. Therefore, in each sector, two phase current measurements are available; see Figure 5-6. The calculation of the third phase current value is possible because the three winding currents sum to zero. The voltage vector combination and corresponding reconstructed motor phase currents are shown in Table 5-1.

Table 5-1. Measured Currents

Voltage Vector	DC-Link Current
$V_1(100)$	$+i_a$
$V_2(110)$	$-i_c$
$V_3(010)$	$+i_b$
$V_4(011)$	$-i_a$
$V_5(001)$	$+i_c$
$V_6(101)$	$-i_b$
$V_7(111)$	0
$V_0(000)$	0

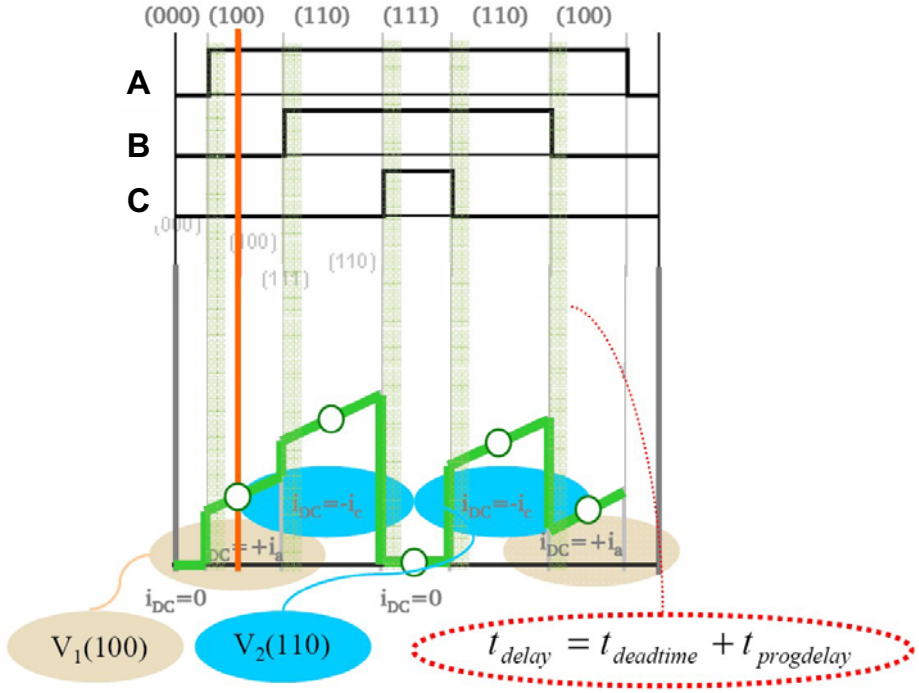

Figure 5-6. Current Sampling Timing

The ADC converter is triggered three times on every second PWM period. The first two triggers sample the DC-link current corresponding to the two phase currents. They are set to the middle of the switching vector. The third trigger is set to the middle of the PWM period. All bottom IGBTs are switched off — current is not flowing through the shunt resistor. Samples taken at this point refer to a zero current. These are used for channel offset calibration.

However, the DC-link current cannot be measured in two cases:

- When the voltage vector is crossing a sector border — in this case, only one sample can be taken; see Figure 5-7
- When the modulation index is low — the sampling interval is too short and none of the current samples can be taken; see Figure 5-8

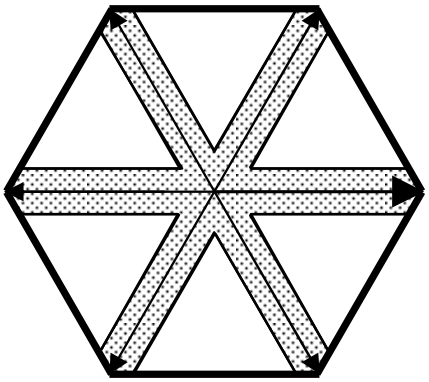

Figure 5-7. Passing Active Index

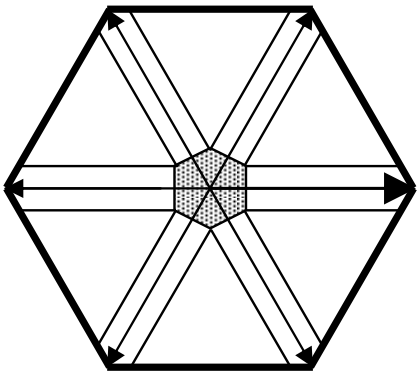

Figure 5-8. Low Modulation Vector

This current measurement limitation can be partly solved using asymmetrical PWM pulses. Two PWM pulses are shifted to obtain enough time for current sampling. Nevertheless, duty cycles for all the PWM pulses have to be preserved.

The solution used for asymmetrical PWMs can be applied in both cases. In the first case, the voltage vector crosses a sector border, the center edge of the PWM period is frozen, and one critical edge is moved; see Figure 5-9. In the second case, when the modulation index is low, the center edge remains frozen as well, and both side edges are moved in opposite direction; see figure Figure 5-10.

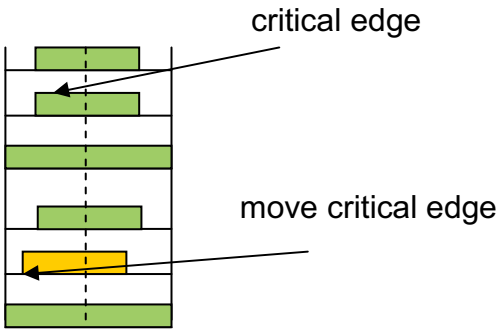

Figure 5-9. Edge Moving When Passing an Active Vector

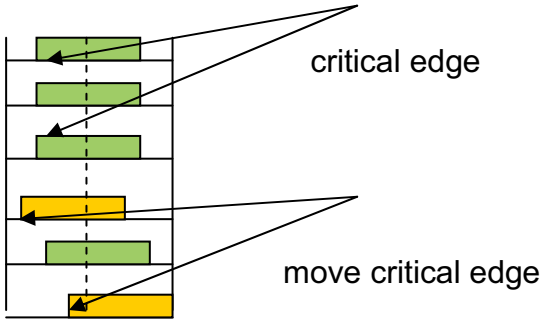

Figure 5-10. Edge Moving When Modulation Index is Low

Limits for current reconstruction:

- On every second PWM cycle, three triggered samples have to be taken. The conversion time + ADC ISR execution time limits the minimum time for two consequent samples to 3 µs.
- The minimum duration of a voltage vector to be able to sample a current signal is approximately 2.5 µs (hardware dependent).
- The maximum voltage vector amplitude is limited by a minimum required PWM edge shift (2.5 µs). In other words it cannot reach a 100% duty-cycle. If a 100% duty cycle is being applied, asymmetrical PWMs cannot be performed.
- The three-phase current reconstruction algorithm requires both extensive calculations and bit manipulation. The execution time on the MC56F8023 is about 20 µs. The current reconstruction algorithm is performed in the PWM reload interrupt routine together with other calculations. This execution time is limited to one-half of the PWM reload period.
- The PWM reload interrupt routine performs the fast inner current loop together with the current reconstruction algorithm. The three-phase currents are transformed into alpha and beta components of the space vector in the stationary reference frame. Having these alpha and beta components, the actual vector size (amplitude) is evaluated and used as a feedback signal for the PI controller. The execution time on the MC56F8023 is 56 µs while the PWM period is 62.5 µs (16 kHz). Due to a shortage of time after processing the other algorithms, the DC-link current measurement and PWM reload interrupt is processed every second PWM period in a 125 µs loop; see Figure 5-3.

5.3.3 Position and Speed Sensing Using the Encoder

The position and speed of the rotor are sensed by an incremental encoder mounted on the motor shaft. This generates two quadrature-encoded signals (phases A and B); see Figure 5-11.

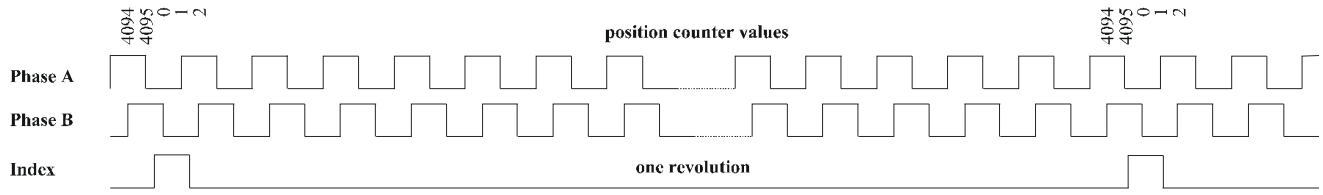

Figure 5-11. Quadrature Encoder Signals

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 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. Similarly, as with 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 is shown in Figure 5-12.

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 a prescaler set to 2. The counter is configured to count rollover. 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; the second method measures the position difference per constant period. The first method is used at low speed. At higher speeds, when the measured period is very short, the speed calculation algorithm switches to the second method.

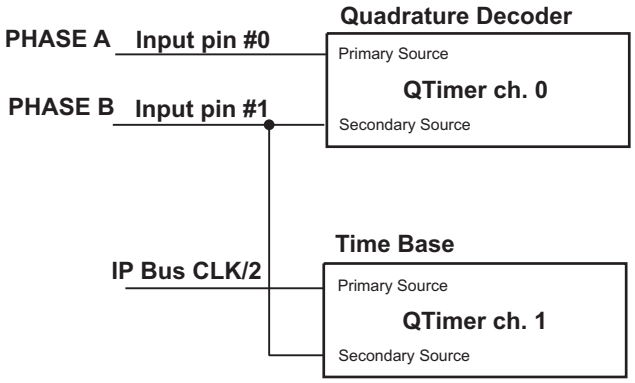

Figure 5-12. QuadTimer Channel Configuration

The proposed algorithm combines both of the aforementioned 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:

$$Speed = \frac{k_1 \times N}{t} = \frac{k_1 \times N}{T_{clkt2} N_{clkt2}} = \frac{k \times N}{N_{clkt2}} \qquad \textit{Eqn. 5-19}$$

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, which 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, TMR1) and the capture register is disabled.
3. The speed is calculated according to Equation 5-19.
4. This process is repeated with each call of the speed processing algorithm; see Figure 5-13.

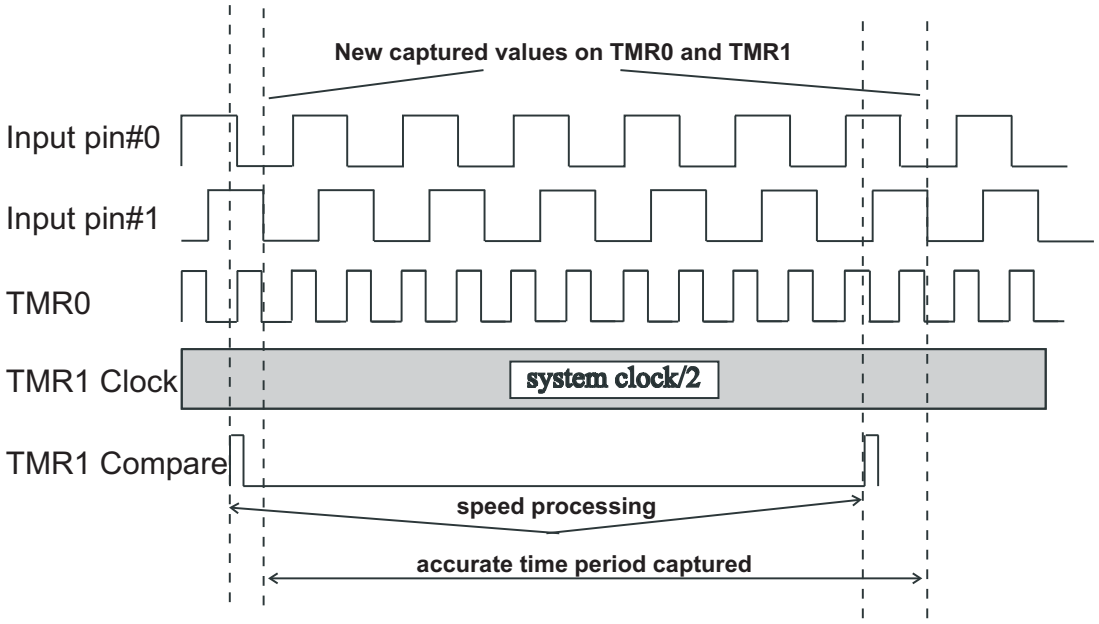

Figure 5-13. Speed Processing

5.4 Software Implementation

The general software diagram incorporates the main() routine 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 Software States

5.4.1.1 Application Process States

The application process state is the highest level of the application state machine. It has five application states: APP_FAULT, APP_READY, APP_TRANS_TO_RUN, APP_RUN, and APP_TRANS_TO_READY. Transition between the states is shown in Figure 5-14.

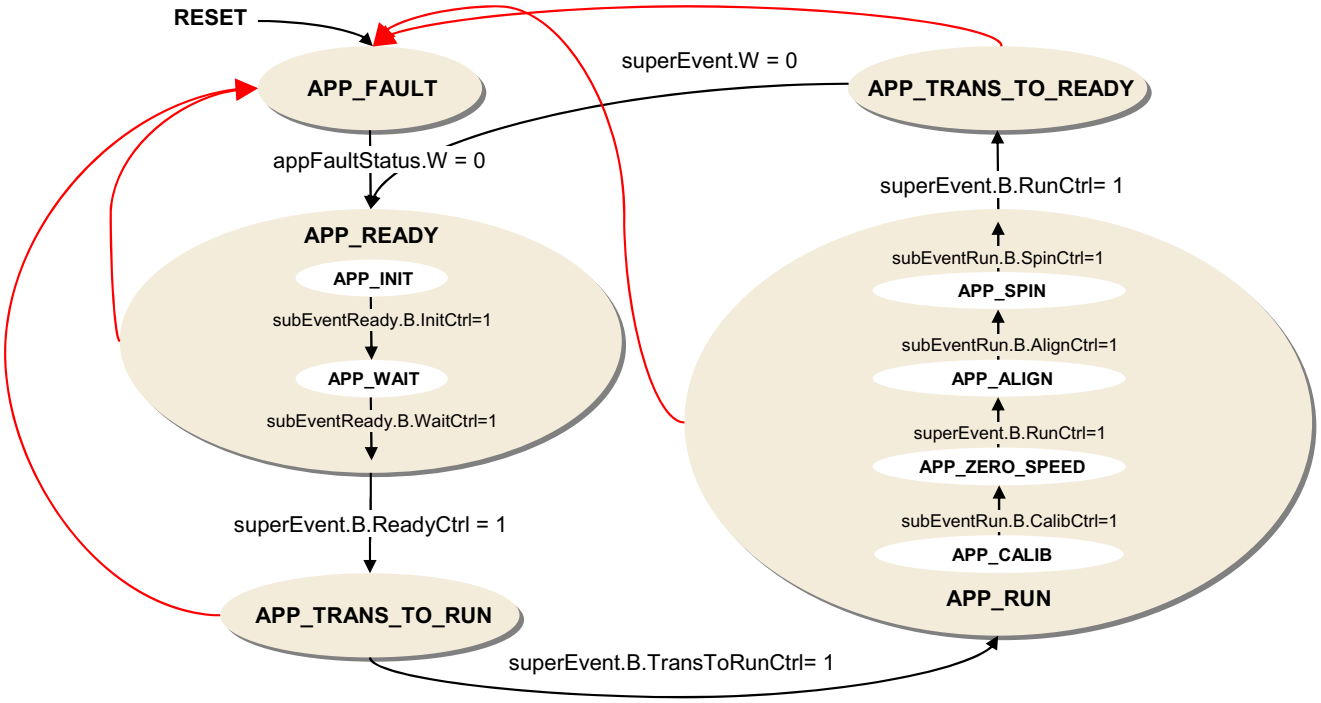

Figure 5-14. Application State Diagram — General Overview

After reset, the application is set to the fault state where over-current, over-voltage, under-voltage, and over-temperature fault bit states are tested. If there is no fault detected, the application goes to the next state. Once any of them are set, the fault state is immediately entered.

The application process states, with their substates, are described in the following sections.

5.4.1.1.1 APP_FAULT

The application goes to this state immediately after reset or when a fault is detected. The system allows all the states to pass on to the APP_FAULT state. Faults detected in the application are:

- Over current
- Over voltage
- Under voltage
- Over temperature

The next state (APP_READY) is set once all fault bits are cleared. This means that no fault has been detected (`appFaultStatus.w=0`).

5.4.1.1.2 APP_READY

This state is entered from either APP_FAULT or APP_TRANS_TO_READY state. The application variable initialization is performed (APP_INIT substate) and the system waits until the *ReadyRunFlag* is set from the FreeMASTER control software. Then the next state is entered.

5.4.1.1.3 APP_TRANS_TO_RUN

This transient state, between READY and RUN states, enables the PWM pads and sets the fifty-percent duty cycle.

5.4.1.1.4 APP_RUN

This is the most complex state, and it consists of four substates. All necessary procedures for motor startup and rotation are processed in this state.

- APP_CALIB — waiting until the ADC channel is properly calibrated and channel offsets are stored in ADC offset registers
- APP_ZERO_SPEED — current measurement is started and fifty-percent duty cycle (zero speed) is initiated
- APP_ALIGN — the rotor is aligned to the known position before the motor is started
- APP_SPIN — the motor is started according to the required speed

The next state is entered after the ReadyRunFlag is cleared from the FreeMASTER control page.

5.4.1.1.5 APP_TRANS_TO_READY

This transient state, between the RUN state and the READY state, stops the motor, disables the PWM pads, and clears the state bits. This puts the application conditions into the same status as they would be after reset. Finally, the READY state is entered.

5.4.1.2 Application Processes

Application processes are called from fast control loop PWM_Reload (125 µs). They correspond to application states which control transitions between component processes. Figure 5-15 shows the process state diagram.

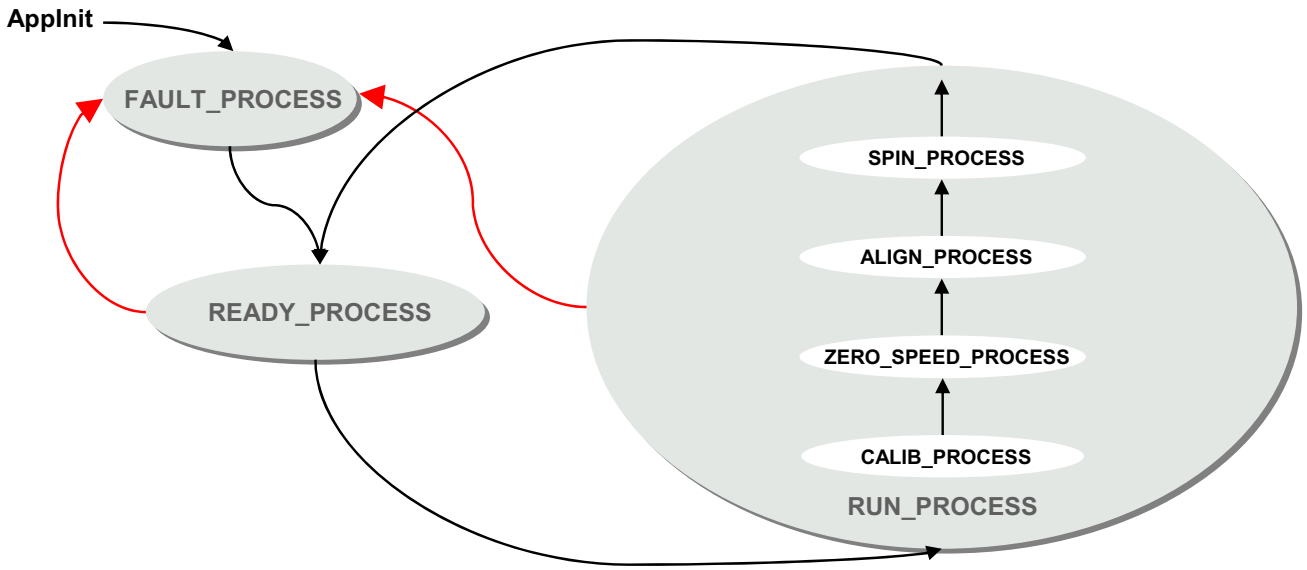

Figure 5-15. Process State Diagram

These processes perform routines and tasks required in each application state and substate, such as ADC calibration, rotor alignment, and FOC calculation (rotor angle, DC-bus ripple elimination, etc.) and transformations (Park, Clark).

5.4.2 Initialization

Application initialization is entered after a reset. When the application main() is entered, a low-level initialization is called. The DSC registers are initialized according to the peripherals and the CPU. This is the first function which has to be called in the application main(). As the next step, the FreeMASTER embedded driver is initialized according to settings in the freemaster_cfg.h configuration file.

Finally, the application initialization function AppSubStateReadyInit() is called when the APP_READY state is entered. Tasks performed in AppSubStateReadyInit() are as follows:

- FreeMASTER scaling variables are defined
- The analog-to-digital convertor data structure is initialized using parameters from the PMSM_VC_app_setup.h configuration file
- The ADC_Init function is called
- The encoder module data structure is initialized using parameters from the PMSM_VC_app_setup.h configuration file
- The ENC_Init function is called and the actual omega value filter is initialized
- The zero-cancellation filters and d,q current controller parameters are initialized using the configuration file
- Speed controller and ramp parameters are set
- The fifty-percent duty cycle is set and the PWM pads are disabled

5.4.3 Application Background Loop

The endless application background loop executes a simple application state machine, which is shown in Figure 5-14. Other process functions called from the loop are:

- FreeMASTER polling function FMSTR_Poll()
- The background part of the fault detection with AppFaultDetection()
- Watchdog periodic feeding

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

5.4.4 Interrupts

There are three periodic interrupt service routines executing the major motor control tasks, and then the PWM reload interrupt, which is 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 ADC_EndOfScanISR() is assigned to this interrupt event. The interrupt is configured to be executed as a fast interrupt with a priority level 2. It is executed three times within every second PWM cycle. It is synchronized to the PWM reload signal and triggered by a TMR3 output. A more detailed description of the ADC end of scan interrupt timing can be found in 5.3.1, “ADC End-of-Scan Timing and PWM Reload Interrupts.” The third ADC trigger is set to the middle of the PWM cycle, when all bottom transistors are off and no current is flowing via the shunt resistor.

Samples taken at this point are used to calibrate the offset of the DC-link current sample. The first two samples are set to the middle of the switching vector. Current sampled at these points is employed for the motor phase current reconstruction.

Tasks performed by the ADC_EndOfScanISR()function:

- Clear the end of scan interrupt flag.
- Store the value from result register 0 (sample 4) into the ADC result buffer.
- Decrement the N register counter.
- Test whether this is the third conversion finished.
- If not, store the midpoint to Timer 3 compare preload register (offset sampling).
- Clear the TMR3 output flag (OFLAG).
- If this is the last conversion, TMR3 is stopped, the PWM reload interrupt is enabled, and the EOS interrupt is disabled.

5.4.4.2 PWM Reload Interrupt

The function PWM_ReloadISR() is assigned to this interrupt event. It is executed every time the series of two subsequent ADC samples has finished. When the PWM Reload occurs at the beginning of a PWM cycle the interrupt request flag is set. The interrupt service routine is disabled ate this moment to enable the first intake of ADC samples. When ADC sampling is finished, the ADC end of scan interrupt routine enables execution of the PWM Reload interrupt service routine. The priority of the interrupt is set to level 2.

The PWM_ReloadISR() function executes a fast control loop. The most time-critical tasks of the vector control algorithm are performed here, including the current-control loop.

Tasks performed by the PWM_ReloadISR()function:

- Start ADC for DC-bus voltage sampling.
- Store reconstructed phase current values in the i_abc_recons data structure.
- Read the sampled DC-bus voltage from the ADC result register 0, and execute the digital filter on that sample.
- Perform a transformation of the stator phase-currents from the three-phase stationary reference frame into the two-phase stationary reference frame (forward Clarke transformation).
- Perform a transformation of the stator phase currents from the stationary into the rotational reference frame (forward Park transformation).
- Execute digital filters on the d,q components of the stator current to remove sampling spikes.
- Execute the PI controllers of the d-axis and q-axis components of the stator current.
- Evaluate the d,q components of the output voltage vector, by summing the controller outputs with the decoupling components of the stator voltage.
- 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.
- Initialize the shadow registers and configure TMR3 for the next ADC trigger.
- Service the PWM reload interrupt flag.
- Disable the PWM reload interrupt.

5.4.4.3 QuadTimer Channel #1 Compare Interrupt

The function QTA1_compare_ISR() 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 QTA1_compare_ISR() function executes a slow control loop. The less time-critical tasks of the vector control algorithm are performed here, including the speed control loop.

Tasks performed by the QTA1_compare_ISR() function:

- When encoder code compilation is defined (ENCODER_MODULE_ON), execute the ENC_NewTimeBase(&enc1024) function to set a new compare value for the 1 ms loop.
- Calculate actual speed value utilizing the ENC_AngularVelocity2(&enc1024, thetaRotorEl_Enc) function.
- Perform ramping of the motor speed.
- 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_FAULT_ISR() 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 generated. The priority of the interrupt is set to level 2.

Tasks performed by the PWM_FaultISR() function:

- Disable the PWM output pads.
- Turn the motor off.
- Set the application OverDCBusVoltageFlag or OverDCBusCurrentFlag flags.
- Service the corresponding interrupt request flag.

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, is a fractional number in the range from 0 to 1 (representing 0 to 32767), and the gain scale parameter shifts the particular gain to the left if positive. The gain scale number represents the number of shifts.

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

5.5 FreeMASTER Software

FreeMASTER software was designed to provide a debugging, diagnostic, and demonstration tool for the development of algorithms and applications. Moreover, it’s very useful for tuning the application for different power stages and motors, because 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 process control information from the PC. FreeMASTER software executing on the PC uses Microsoft Internet Explorer as the user interface.

5.5.1 FreeMASTER Serial Communication Driver

The presented 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. It brings, however, many useful enhancements and optimizations.

The main advantage of the new driver is a unification across all supported Freescale processor products, as well as several new features that were added. One of the key features implemented in the new driver is target-side addressing (TSA), which enables an embedded application to describe the memory objects it grants the host access to. By enabling the so-called TSA-Safety 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 the presented application, the driver has already been included.

The FreeMASTER driver files are located in the 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 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 should be included in your application (using #include directive), wherever you need to call any of the FreeMASTER driver API functions.

Note that the FreeMASTER driver does not perform any initialization or configuration of the SCI module it uses to communicate. Therefore 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 baud.

NOTE

Higher communication speeds than 9600 baud are 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 SCI. Both communication and protocol decoding are handled 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 the macros defined there for conditional and parameter compilation.

A detailed description of the FreeMASTER Serial Communication Driver is provided in AN2471, "PC Master Software Communication Protocol Specification."

5.5.2 FreeMASTER Recorder

Part of the FreeMASTER software is also a recorder, which is 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 RS-232 serial port. The sampled data can be displayed in a graph or the data can be stored. The recorder behaves like a simple on-chip oscilloscope with trigger / pretrigger 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 in which you want to take the 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, which creates a 125 ms time base for the recorder function. A detailed description of the FreeMASTER software is provided in AN2395, “PC Master Software Usage.”

5.5.3 FreeMASTER Control Page

The FreeMASTER control page creates a graphical user interface (GUI) for the PMSM vector control application. Start the FreeMASTER software window’s project by clicking on the PMSM_VC_EN_SHUNT.pmp file. Figure 5-16 illustrates the FreeMASTER software control window after this project has been launched.To switch to the control page, click on the “control page” tag.

A user is able to monitor all the important quantities 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.

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

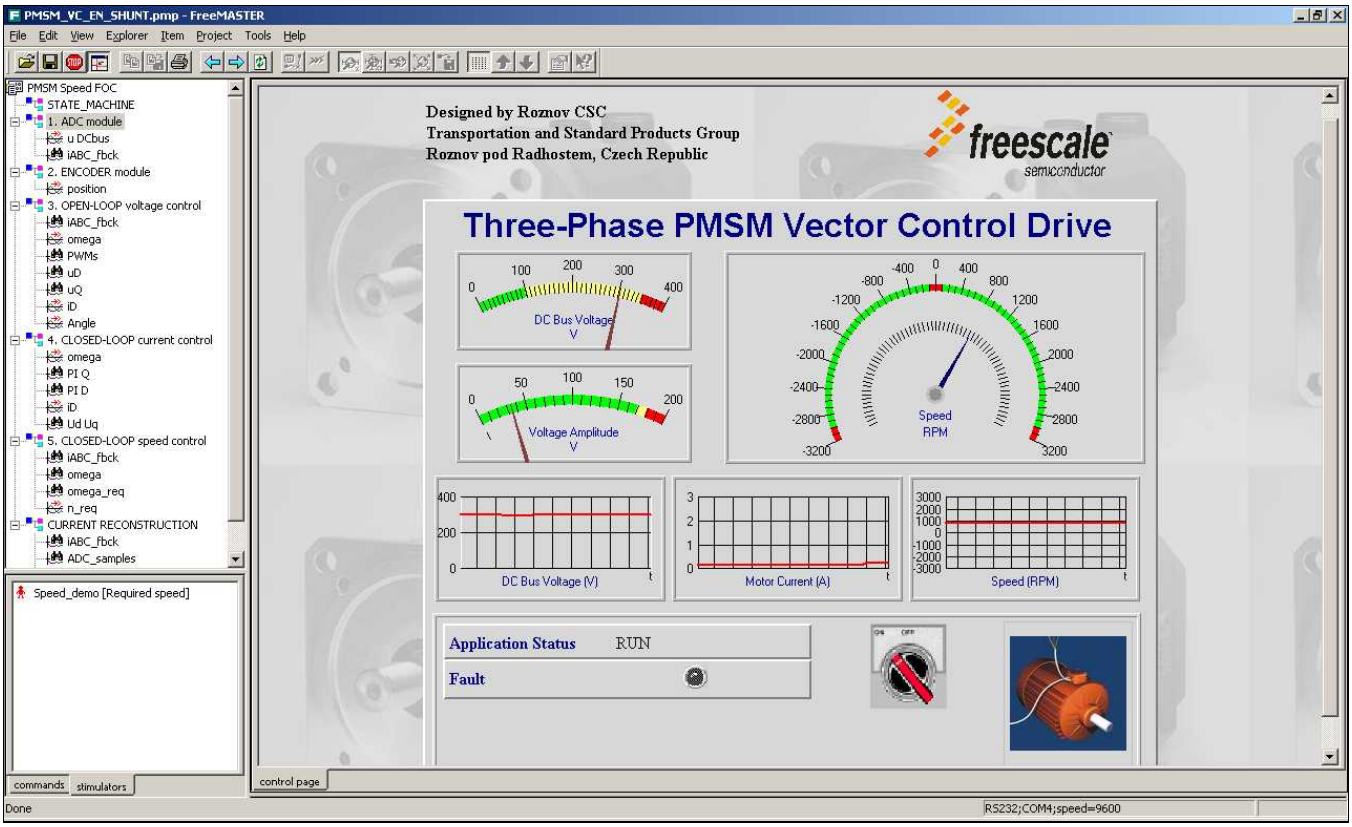

Figure 5-16. FreeMASTER Control Screen

The following FreeMASTER software control page actions are supported:

- Setting the required speed of the motor
- Switch running motor on/off

The FreeMASTER software control page displays:

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

5.6 Setting the Software Parameters for a Specific Motor

The default software parameter settings have been tuned for a default hardware setup with the motor TGT3-0065-30-320. The setup is described in Chapter 6, “Application Setup.” When another motor and hardware are used, the software settings need to be changed according to their specific parameters. Detailed parameter setting and measurement is not described in this document. But one possibility is to use a dedicated setup tool xls file, which is included within this software version. This PMSM_VC_SHUNT_EN.xls file can calculate the constants and generate them for PMSM_VC_app_setup.h, according to the motor and hardware parameters. The xls file is included within the code directories (ApplicationConfig\plugin_excel.gct\) and is opened from GCT; see Figure 5-17.

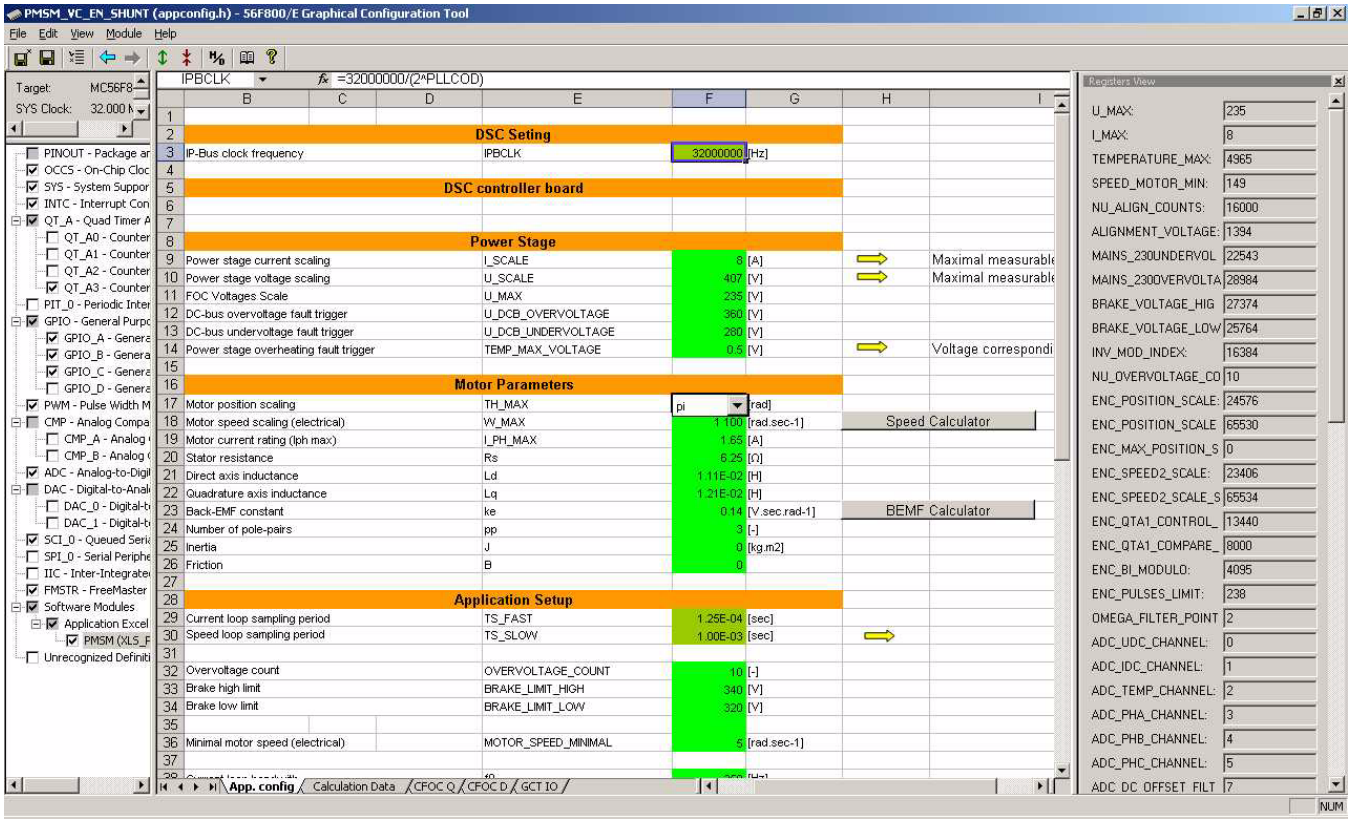

Figure 5-17. Setup Tool Screen

Chapter 6
Application Setup

As described earlier, the PM synchronous motor vector control application is targeted at the MC56F8013/23 device. The design of the PM synchronous motor vector control drive incorporates these hardware components:

- MC56F8013/23 controller board
- Three-phase AC/BLDC high-voltage power stage board
- Three-phase PM synchronous motor (default configuration for motor TG drives TGT3-0130-30-320/T0KX-1.0M)

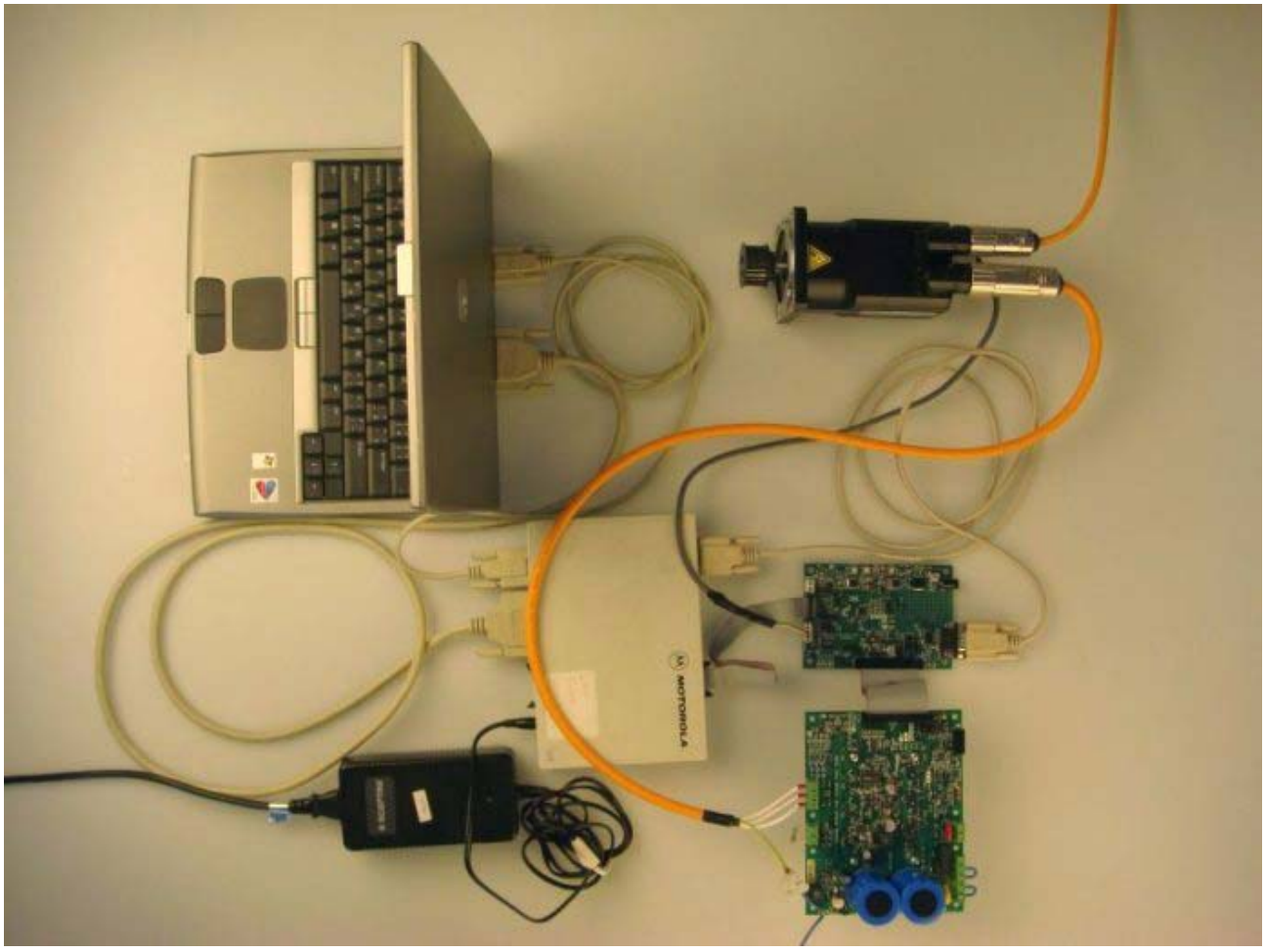

Figure 6-1. Demo Application Setup

6.1 MC56F8013/23 Controller Board Setup

Prior to the MC56F8013/23 controller board being connected to the power stage, it needs to be configured for correct operation. Also, the demo application code first has to be programmed into the flash memory. For the MC56F8013/23 controller board configuration, obey these steps:

1. Set the jumper configuration on the MC56F8013/23 controller board as shown in the table:

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

Jumper	Setting	Description
JP1	2–3	SCI bi-wire configuration
JP3	4–5, 7–8	Incremental encoder 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)
JP16	1–2	+5 V power supply from UNI-3 connector
JP18	1–2	+15 V power supply from UNI-3 connector
Note: Other jumpers are set open		

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

   Trim potentiometer R29 sets the over-voltage threshold. Use a voltmeter to measure the threshold level at test point TP3. Turn the R29 trim potentiometer to set the threshold level. Voltage level on TP3 should be > 3.2 V.

   Trim potentiometer R32 sets the over-current threshold. Use a voltmeter to measure the threshold level at test point TP5. Turn the R32 trim potentiometer to set the threshold level. Voltage level on TP3 should be > 3.2 V.
4. 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.
5. Compile your project and program it into the device.
6. Disconnect the +12 V power supply from the J12 power connector on the MC56F8013/23 controller board.
7. Unplug the JTAG command converter (or the USB-TAP) from the J4 header on the MC56F8013/23 controller board.

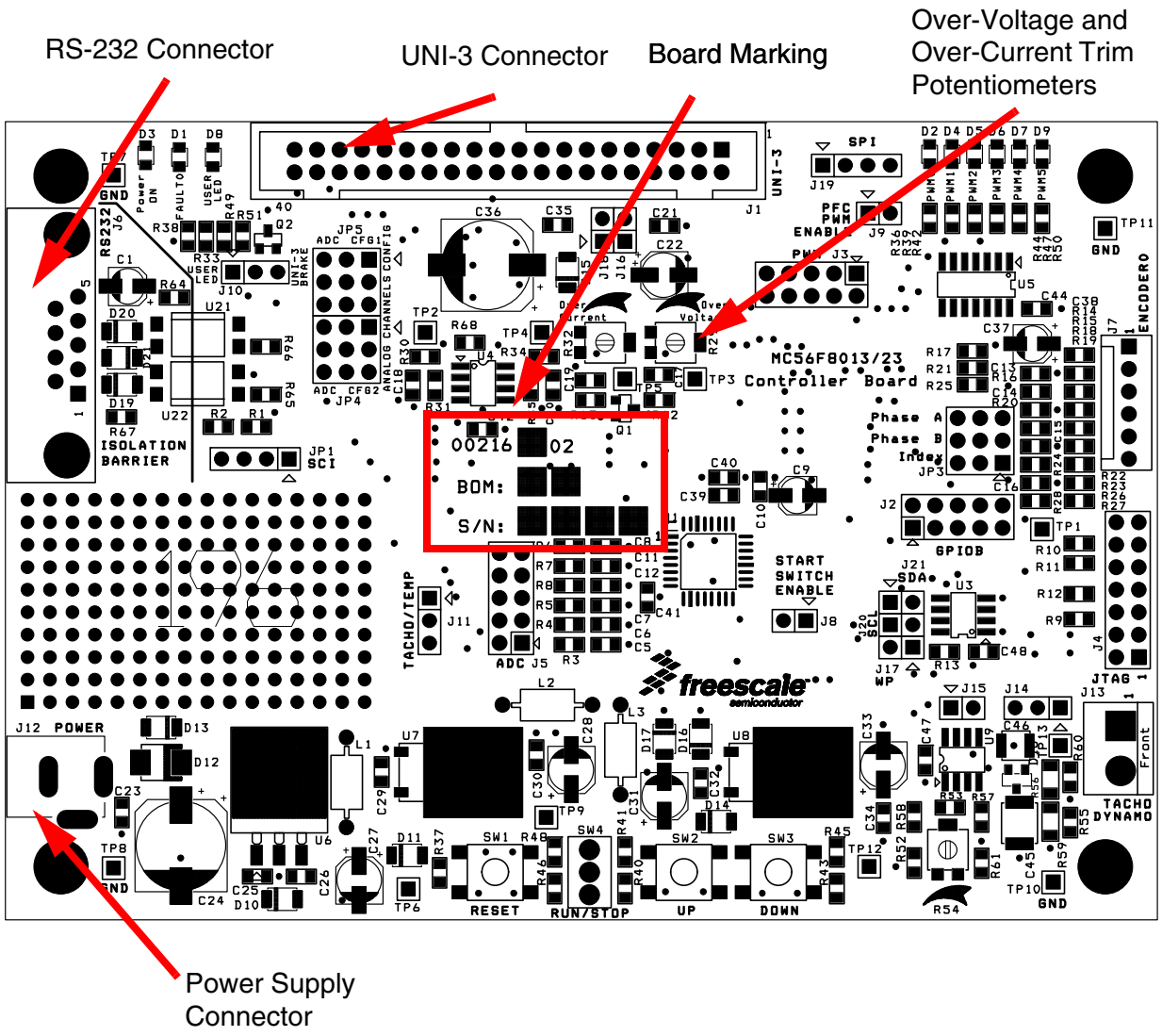

Figure 6-2. MC56F8013/23 Controller Board View

6.2 Demo Hardware Setup

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

1. Connect the UNI-3 connector (J1) on the MC56F8013/23 Controller Board to the UNI-3 counterpart connector on the three-phase 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 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 terminal J6 on the three-phase AC/BLDC high voltage power stage board.
5. Connect a 115/230 V AC power supply or a 100–320 V DC power supply to terminal J6 on the three-phase AC/BLDC high voltage power stage board.

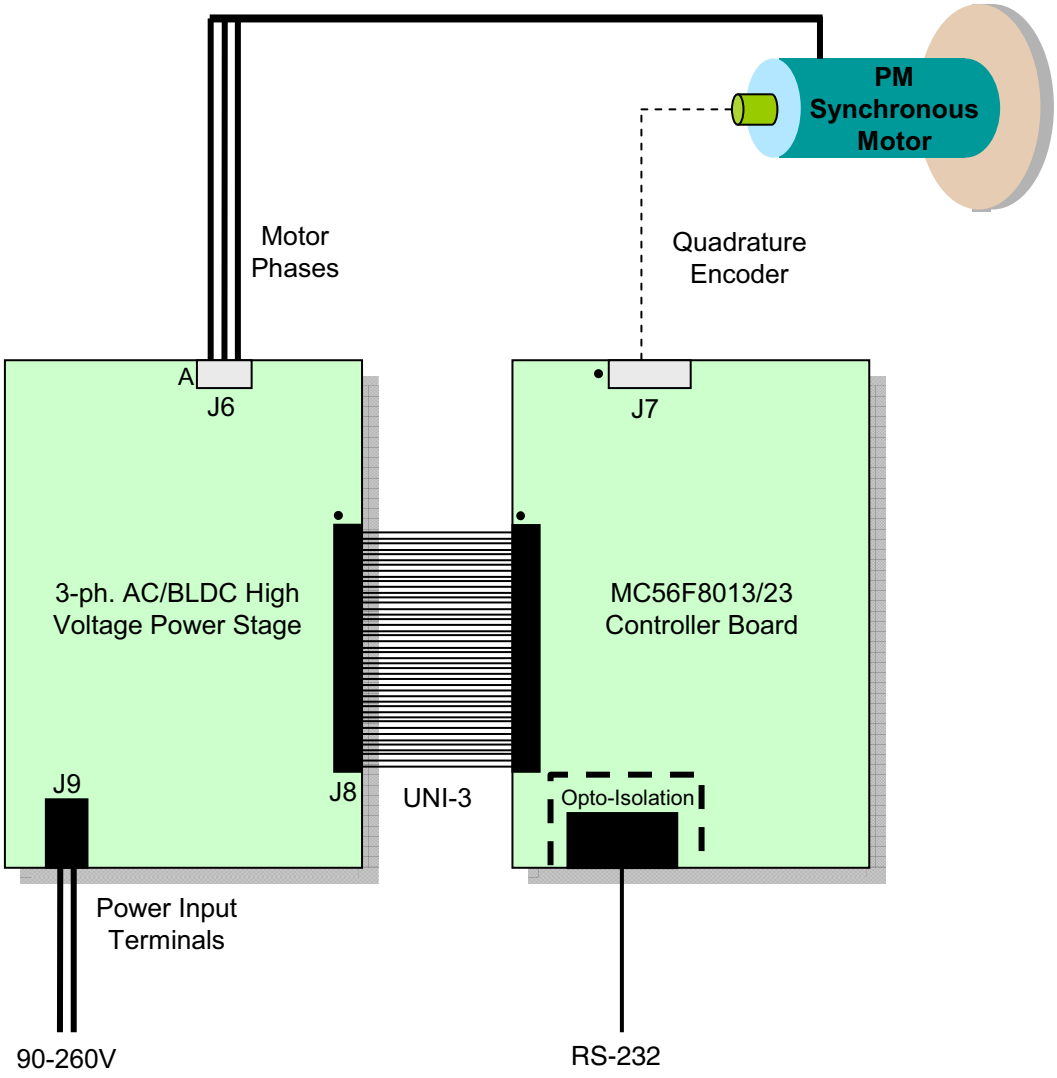

Figure 6-3. Demo Application Connection Overview

6. Switch on the high-voltage power supply.
7. Start the FreeMASTER project and switch to the “control page” pane.

WARNING

There is a risk of electric shock. Both the MC56F8013/23 controller board and the three-phase 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. The user must avoid touching them. The only safe mode of control is by remotely controlling the application using the host PC running the FreeMASTER application. The RS-232 port is the only interface that provides galvanic isolation.

WARNING

The JTAG connector does not provide galvanic isolation. The demo hardware setup must be disconnected from high voltage if a JTAG connection is required. 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 PMSM vector control application with single shunt current measurement. One of the most important parts of this PMSM vector drive is three-phase current reconstruction from DC-link current samples. Most of the measurements are concentrated on it.

7.1 System and Measurement Conditions

7.1.1 Hardware Setup

The application measurements were provided using a TGT3-0130-30-320 motor and the default hardware setup described in Chapter 6, “Application Setup.”

7.1.2 Software Setup

The software file PMSM_VC_ENC_SHUNT.c contains the main() function together with the state machine and corresponding process functions.

The measurements were provided using default code parameter settings, if not mentioned otherwise. The preprocessor constants CLOSED_LOOP_CURRENT_CONTROL_ON and CLOSED_LOOP_SPEED_CONTROL_ON are defined in the PMSM_VC_ENC_SHUNT.cfile. 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_VC_EN_SHUNT.pmp file is used by the FreeMASTER software, which needs to be installed on the PC. The pmp file is included within the software structure. This incorporates the definition of the recorder used for the measurements.

7.2 Measured Results

7.2.1 Three-Phase Current Reconstruction

This section presents the measurements of the three-phase motor current reconstructed using DC-link samples. Ripples are caused by PWM signal shifting during sector crossing; see Figure 7-1.

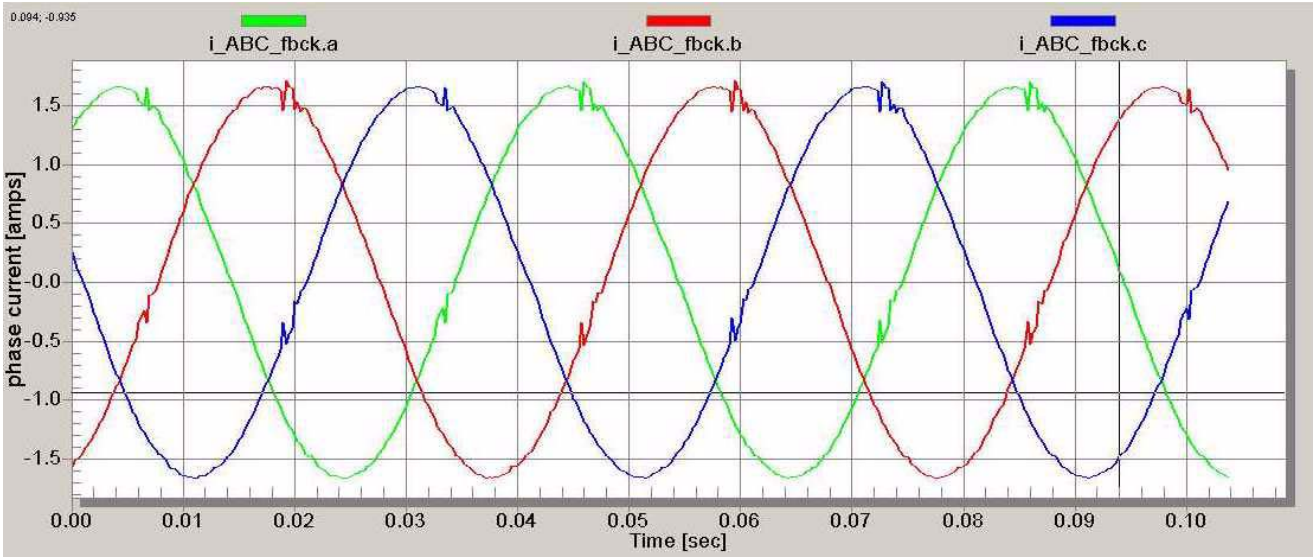

Figure 7-1. Motor Phase Reconstructed Currents

7.2.2 DC-Link Sample Timing

Sample times are set according to actual phase-duty cycle. The samp_scaled[2] signal is set as a constant to value 1080 for DC-link offset measuring. While sector crossing, the PWM signal shifting is applied. Thus, the svm_sector signal oscillates; see Figure 7-2.

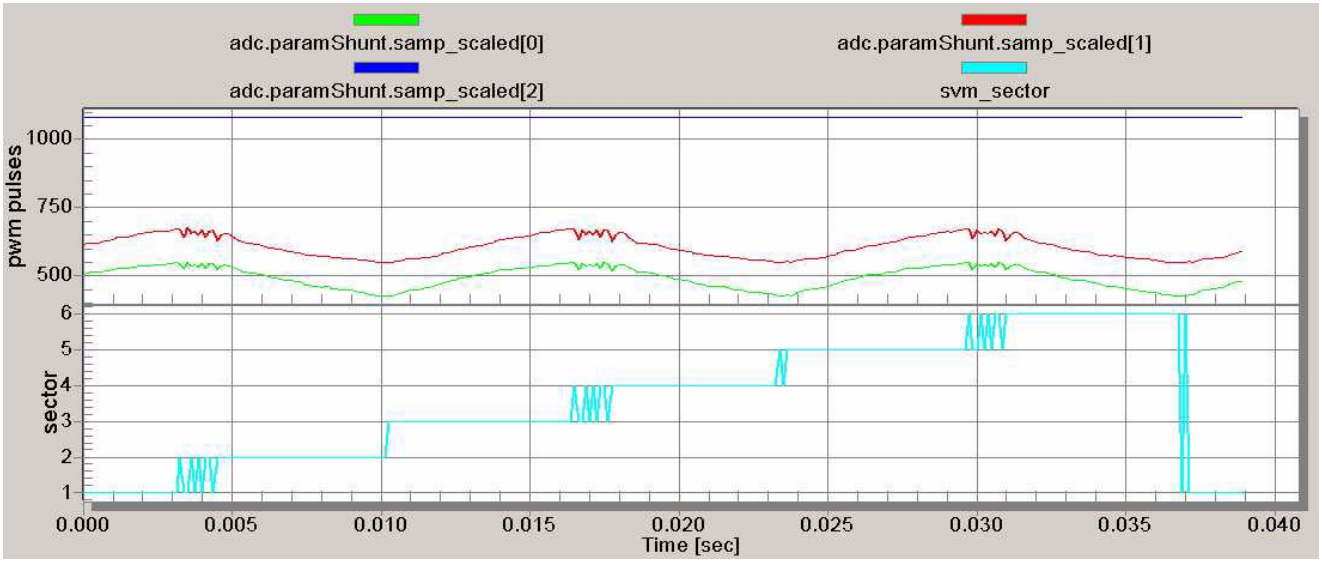

Figure 7-2. Sample Timing

7.2.3 PWM Signal Shifting

Actual values of motor phase currents need to be measured instantaneously. Because phase PWM signal overlapping at the time of sector crossing does not allow measuring them, signal shifting is applied. Figure 7-3 shows the duty cycle of phase A (turquoise signal, the line that runs up and down through the

middle of the diagram) and the shifted signals of the high switch (brown signal—the lines at the very top and very bottom of the figure) and the low switch (gray signal—the lines that mirror the brown signal).

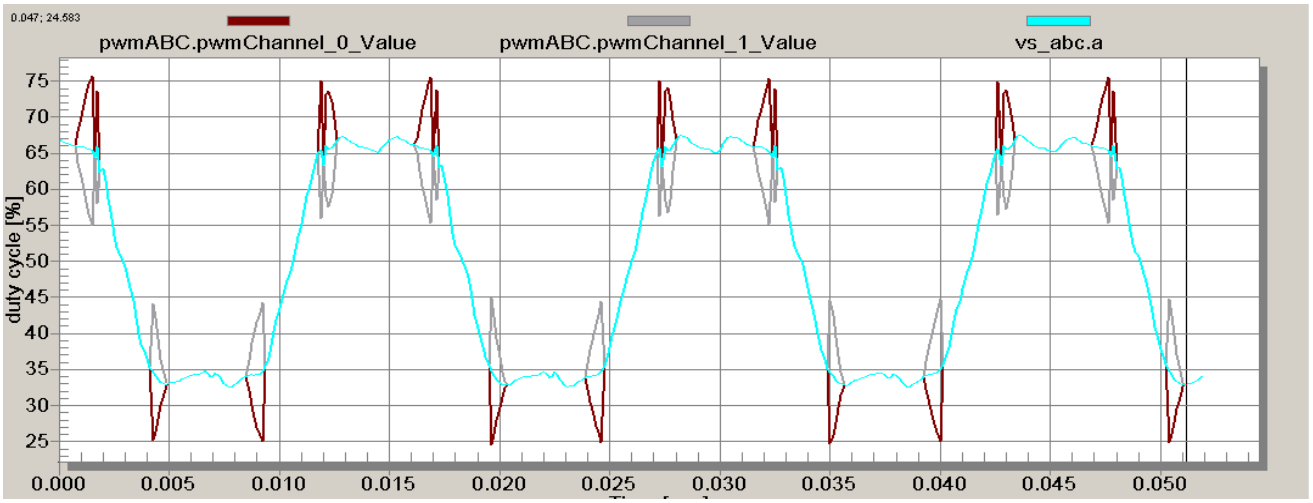

Figure 7-3. PWM Signal Shifting

7.2.4 Speed Controller

The outer control loop processes the speed PI controller. The required input values for this controller are the ramped required speed and the measured motor speed, which can be taken from the quadrature decoder. The output of the controller is the required value of the q-axis inner control loop PI controller (i_DQ_Command.q); see Figure 7-4.

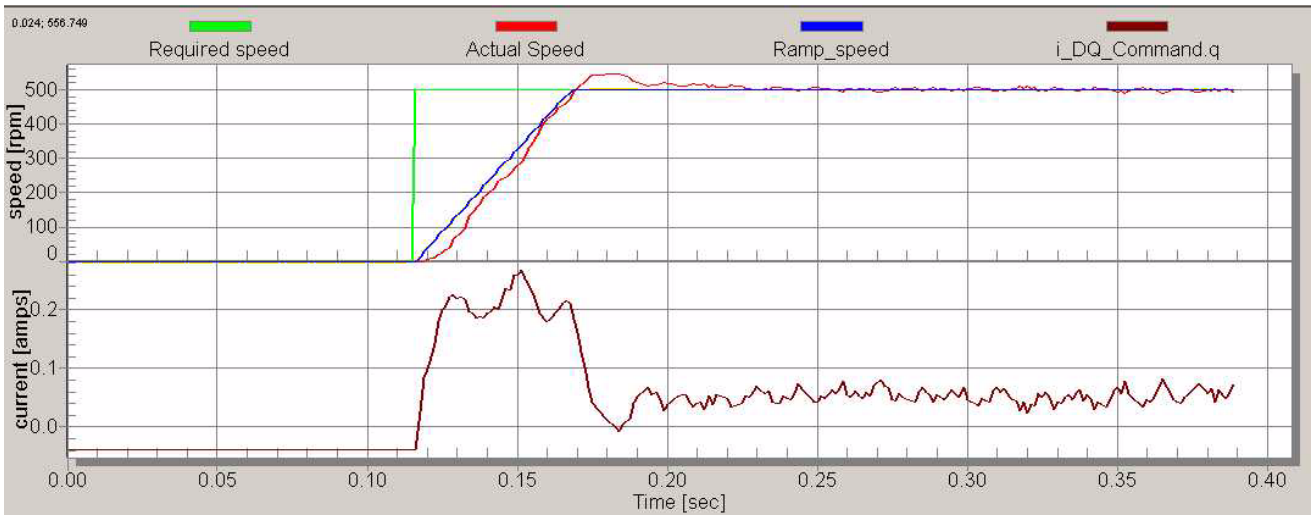

Figure 7-4. Speed Controller

7.2.5 Current Controller

The inner control loop processes the d-axis and q-axis PI controllers. The constant CLOSED_LOOP_SPEED_CONTROL_ON must be undefined for this measurement to not use the outer speed loop. Figure 7-5 shows the step response of the d-axis controller (the q-axis controller is similar).

The d-axis command is set as a required value and the d-axis feedback value (obtained by a transformation from motor phase currents) as the actual value. The output of the controller is the d-axis voltage value used for the following transformation to a three-phase system. (u_DQ_Controller.q).

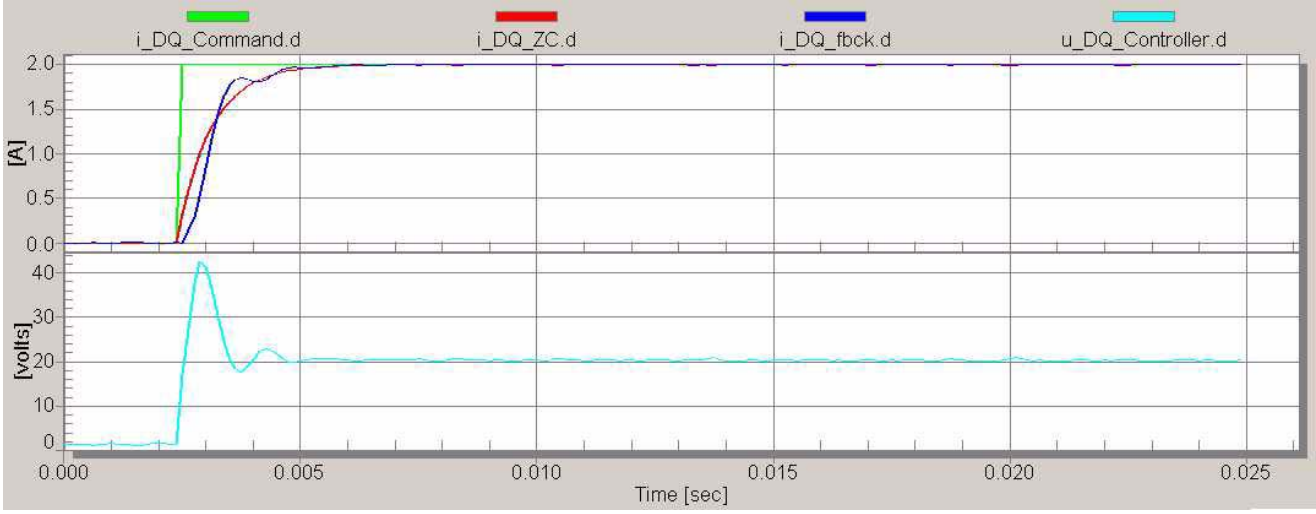

Figure 7-5. Speed Controller

7.3 Conclusion

The measurement results show the real behavior of the application used here. However, the single-shunt current-sensing produces some inaccuracy to the motor control process. This solution is sufficient for general motor control applications where FOC control technique use is required while preserving a cost-effective solution.