3-Phase AC Induction Vector Control Drive with Single Shunt Current Sensing

Designer Reference Manual

568000
Digital Signal Controller

DRM092
Rev. 0
7/2007

freescale.com

3-Phase AC Indudction Vector Control Drive with Single Shunt Current Sensing

Designer Reference Manual

by: Petr Stekl
Freescale Czech Systems Laboratories
Roznov pod Radhostem, Czech Republic

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://www.freescale.com

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Revision History

Date	Revision Level	Description	Page Number(s)
July, 2007	0	Initial release	74

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction

This paper describes the design of a direct vector control algorithm for a three-phase AC induction drive based on Freescale's MC56F8013/MC56F8023 dedicated motor control devices.

Three-phase AC induction motors (ACIM) are popular in industry for a number of reasons. Their construction is extremely optimized, since they have been produced for years. They are very simple and manufacturing costs are favorable. They have no brushes and require minimum maintenance. The robustness of the motor is another strong advantage. We would find induction motors mostly in applications such as water pumps, compressors, fans and air-conditioning systems.

In order to achieve variable speed operations in a three-phase AC induction motor, a variable voltage and variable frequency needs to be supplied to the motor. Modern three-phase variable speed drives (VSD) are supplied with digitally controlled switching inverters.

The control algorithms can be sorted into two general groups. The first group is referred to scalar control. The constant Volt per Hertz control is a very popular technique representing scalar control. The other group is called vector or field oriented control (FOC). The vector oriented techniques brings overall improvements in drive performance over scalar control. Let's mention the higher efficiency, full torque control, decoupled control of flux and torque, improved dynamics, etc.

The reference design deals with the design of a vectors control algorithm for a three-phase AC induction motor implemented on Freescale Semiconductor digital signal controllers MC56F8013/MC56F8023. The presented design is targeted mainly at consumer and industrial applications. This cost-effective solution and high reliability were two key requirements considered. Minimizing system cost the algorithm implements a single-shunt current sensing eliminating three current sensors to one. The high range of motor operating speeds up to 18000RPM is another advantage of the presented design. An adaptive closed loop rotor flux estimator enhances control performance and increases the overall robustness of the system. Sensitivity of the parameter drift can be considerably minimized in this way.

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

1.2 Freescale Controller Advantages and Features

The Freescale MC56F80xx family is well suited 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, analogue-to-digital converters (ADC), timers, communication peripherals (SCI, SPI, I^2C), and on-board Flash and RAM.

The MC56F80xx family members provide the following peripheral blocks:

- One PWM module with PWM outputs, fault inputs, fault-tolerant design with dead time insertion, supporting both centre-aligned and edge-aligned modes
- 12-bit ADC, supporting two simultaneous conversions; 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
- On-board 3.3V to 2.5V voltage regulator for powering internal logic and memories
- Integrated power-on reset and low voltage interrupt module
- All pins multiplexed with general purpose input/output (GPIO) pins
- Computer operating properly (COP) watchdog timer
- External reset input pin for hardware reset
- JTAG/On-Chip Emulation (OnCE™) module for unobtrusive, processor-speed-independent debugging
- Phase-locked loop (PLL) based frequency synthesizer for the hybrid controller core clock, with on-chip relaxation oscillator

Table 1-1 Memory Configuration

Memory Type	MC56F8013	MC56F8023
Program Flash	16 KByte	32 KByte
Unified Data/Program RAM	4 KByte	8 KByte

The three-phase ACIM vector control a with single shunt sensor benefits greatly from the flexible PWM module, fast ADC, and quad timer module.

The PWM offers flexibility in its configuration, enabling efficient three-phase motor control. The PWM module is capable of generating asymmetric PWM duty cycles in centre-aligned configuration. 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 (Quadimers, ADC).

The PWM block has the following features:

- Three complementary PWM signal pairs, six independent PWM signals (or a combination)
- Complementary channel operation features
- Independent top and bottom dead time insertion
- Separate top and bottom pulse width correction via current status inputs or software
- Separate top and bottom polarity control
- Edge-aligned or centre-aligned PWM reference signals
- 15-bit resolution
- Half-cycle reload capability
- Integral reload rates from one to sixteen periods

- Mask/swap capability
- Individual, software-controlled PWM output
- Programmable fault protection
- Polarity control
- 10mA or 16mA 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 5.33MHz with a 187ns period
- Sampling rate of up to 1.78 million samples per second
- Single conversion time of 8.5 ADC clock cycles (8.5 x 187ns = 1.59µs)
- Additional conversion time of six ADC clock cycles (6 x 187ns = 1.125µs)
- Eight conversions in 26.5 ADC clock cycles (26.5 x 187ns = 4.97µs) using parallel mode
- Ability to use the SYNC input signal to synchronize with the PWM (provided the integration allows the PWM to trigger a timer channel connected to the SYNC input)
- Ability to sequentially scan and store up to eight measurements
- Ability to scan and store up to four measurements on each of two ADCs operating simultaneously and in parallel
- Ability to scan and store up to four measurements on each of two ADCs operating asynchronously to each other in parallel
- Interrupt generating capabilities at the end of a scan when an out-of-range limit is exceeded and on a zero crossing
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single-ended or differential inputs
- PWM outputs with hysteresis for three of the analogue inputs

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

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

- Four 16-bit counters/timers
- Count up/down
- Counters are cascadable
- Programmable count modulus
- Maximum count rate equal to the peripheral clock/2, when counting external events
- Maximum count rate equal to the peripheral clock/1, when using internal clocks
- Count once or repeatedly
- Counters are preloadable

- Counters can share available input pins
- Each counter has a separate prescaler
- Each counter has capture and compare capability

The application uses four channels of the quad timer for:

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

Chapter 2
Control Theory

2.1 Three-Phase AC Induction Motor

The AC induction motor is a rotating electric machine designed to operate from a 3-phase source of alternating voltage. For variable speed drives, the source is normally an inverter that uses power switches to produce approximately sinusoidal voltages and currents of controllable magnitude and frequency.

A cross-section of a two-pole induction motor is shown in Figure 2-1. Slots in the inner periphery of the stator accommodate 3-phase winding a,b,c. The turns in each winding are distributed so that a current in a stator winding produces an approximately sinusoidally-distributed flux density around the periphery of the air gap. When three currents that are sinusoidally varying in time, but displaced in phase by 120° from each other, flow through the three symmetrically-placed windings, a radially-directed air gap flux density is produced that is also sinusoidally distributed around the gap and rotates at an angular velocity equal to the angular frequency of the stator currents.

The most common type of induction motor has a squirrel cage rotor in which aluminum conductors or bars are cast into slots in the outer periphery of the rotor. These conductors or bars are shorted together at both ends of the rotor by cast aluminum end rings, which also can be shaped to act as fans. In larger induction motors, copper or copper-alloy bars are used to fabricate the rotor cage winding.

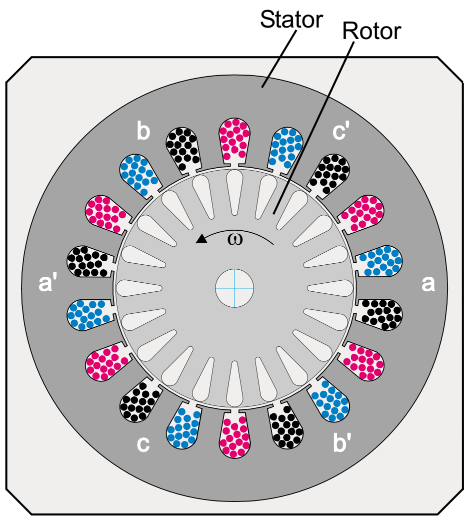


Figure 2-1 Three-Phase AC Induction Motor — Cross Section

As the sinusoidally-distributed flux density wave produced by the stator magnetizing currents sweeps past the rotor conductors, it generates a voltage in them. The result is a sinusoidally-distributed set of currents in the short-circuited rotor bars. Because of the low resistance of these shorted bars, only a small relative angular velocity between the angular velocity of the flux wave (ω_s)and the mechanical angular velocity (ω)

of the two-pole rotor is required to produce the necessary rotor current. The relative angular velocity is called the slip velocity (ω_{slip}). The interaction of the sinusoidally-distributed air gap flux density and induced rotor currents produces a torque on the rotor. The typical induction motor speed-torque characteristic is shown in Figure 2-2.

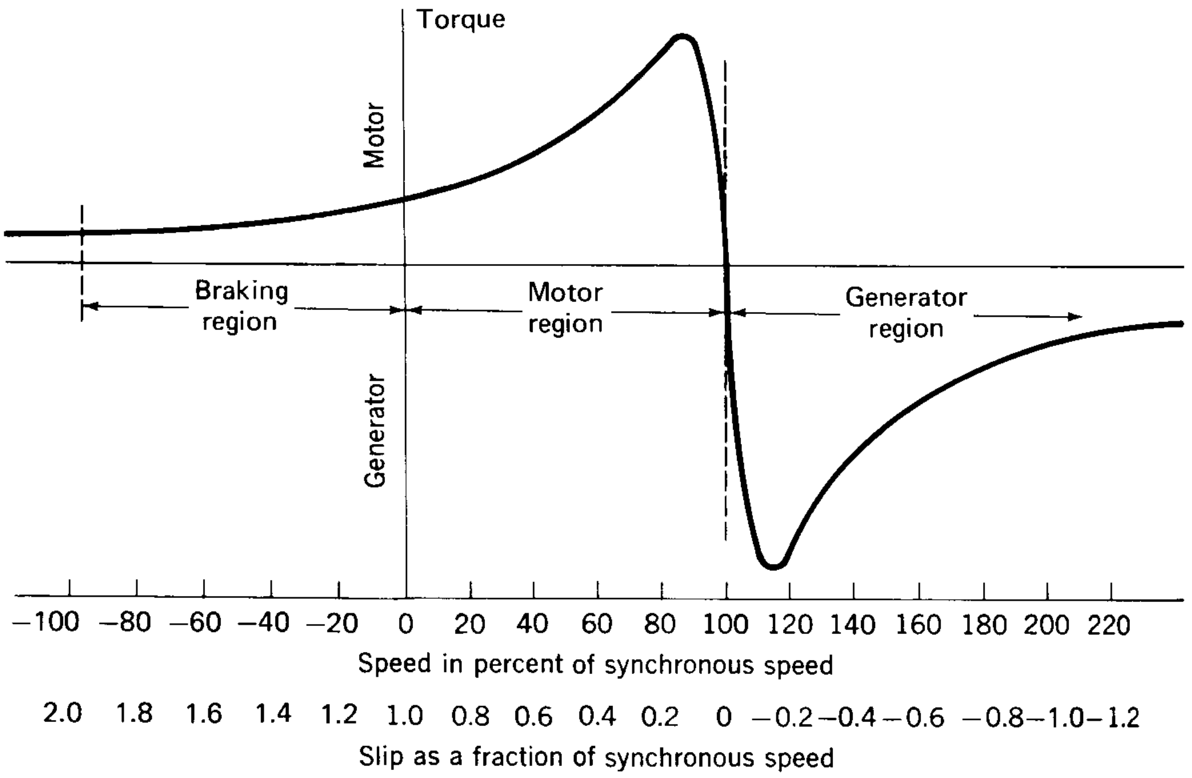


Figure 2-2 AC Induction Motor Speed-Torque Characteristic

Squirrel-cage AC induction motors are popular for their simple construction, low cost per horsepower and low maintenance (they contain no brushes, as do DC motors). They are available in a wide range of power ratings. With field-oriented vector control methods, AC induction motors can fully replace standard DC motors, even in high-performance applications.

2.2 Mathematical Description of AC Induction Motors

There are a number of AC induction motor models. The model used for vector control design can be obtained by utilizing space-vector theory. The 3-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 2-phase machine. Utilizing of the 2-phase motor model reduces the number of equations and simplifies the control design.

2.2.1 Space Vector Definition

Let’s assume i_{sa}, i_{sb}, and i_{sc} are the instantaneous balanced 3-phase stator currents:

$$i_{sa} + i_{sb} + i_{sc} = 0 \tag{2-1}$$

Then we can define the stator current space vector as follows:

$$\bar{i}_s = k(i_{sa} + ai_{sb} + a^2 i_{sc}) \tag{2-2}$$

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 is chosen $k=2/3$. Figure 2-3 shows the stator current space vector projection.

The space vector defined by (2-2) can be expressed utilizing 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:

$$\bar{i}_s = i_{s\alpha} + ji_{s\beta} \tag{2-3}$$

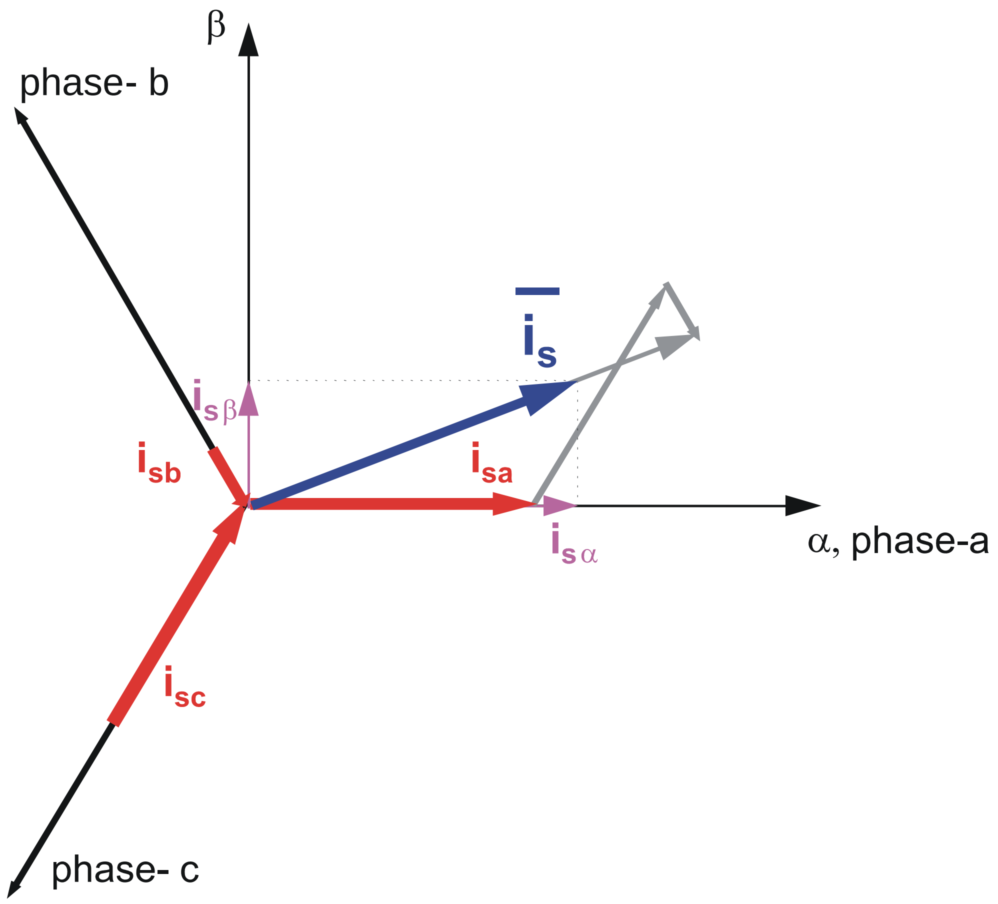


Figure 2-3 Stator Current Space-Vector and Its Projection

In symmetrical 3-phase machines, the direct and quadrature axis stator currents $i_{s\alpha}$, $i_{s\beta}$ are fictitious quadrature-phase (2-phase) current components, which are related to the actual 3-phase stator currents as follows:

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

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

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 AC Induction Motor Model

The AC induction motor model is given by the space vector form of the voltage equations. The system model defined in the stationary α,β-coordinate system attached to the stator is expressed by the following equations. The motor model is supposed to be ideally symmetrical with a linear magnetic circuit characteristic.

a. The stator voltage differential equations:

$$u_{s\alpha} = R_s i_{s\alpha} + \frac{d}{dt}\Psi_{s\alpha} \tag{2-6}$$

$$u_{s\beta} = R_s i_{s\beta} + \frac{d}{dt}\Psi_{s\beta} \tag{2-7}$$

b. The rotor voltage differential equations:

$$u_{r\alpha} = 0 = R_r i_{r\alpha} + \frac{d}{dt}\Psi_{r\alpha} + \omega\Psi_{r\beta} \tag{2-8}$$

$$u_{r\beta} = 0 = R_r i_{r\beta} + \frac{d}{dt}\Psi_{r\beta} - \omega\Psi_{r\alpha} \tag{2-9}$$

c. The stator and rotor flux linkages expressed in terms of the stator and rotor current space vectors:

$$\Psi_{s\alpha} = L_s i_{s\alpha} + L_m i_{r\alpha} \tag{2-10}$$

$$\Psi_{s\beta} = L_s i_{s\beta} + L_m i_{r\beta} \tag{2-11}$$

$$\Psi_{r\alpha} = L_r i_{r\alpha} + L_m i_{s\alpha} \tag{2-12}$$

$$\Psi_{r\beta} = L_r i_{r\beta} + L_m i_{s\beta} \tag{2-13}$$

d. Electromagnetic torque expressed by utilizing space vector quantities:

$$t_e = \frac{3}{2}p_p(\Psi_{s\alpha}i_{s\beta} - \Psi_{s\beta}i_{s\alpha}) \tag{2-14}$$

For glossary of symbols see Appendix B Glossary of Symbols.

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, with direct and quadrature axes *x,y* rotating at a general instantaneous speed $\omega_g = d\theta_g/dt$ is used, as shown in Figure 2-4, 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 the following equation defines the stator current space vector in general reference frame:

$$\overline{i_{sg}} = \bar{i}_s e^{-j\theta_g} = i_{sx} + j i_{sy} \tag{2-15}$$

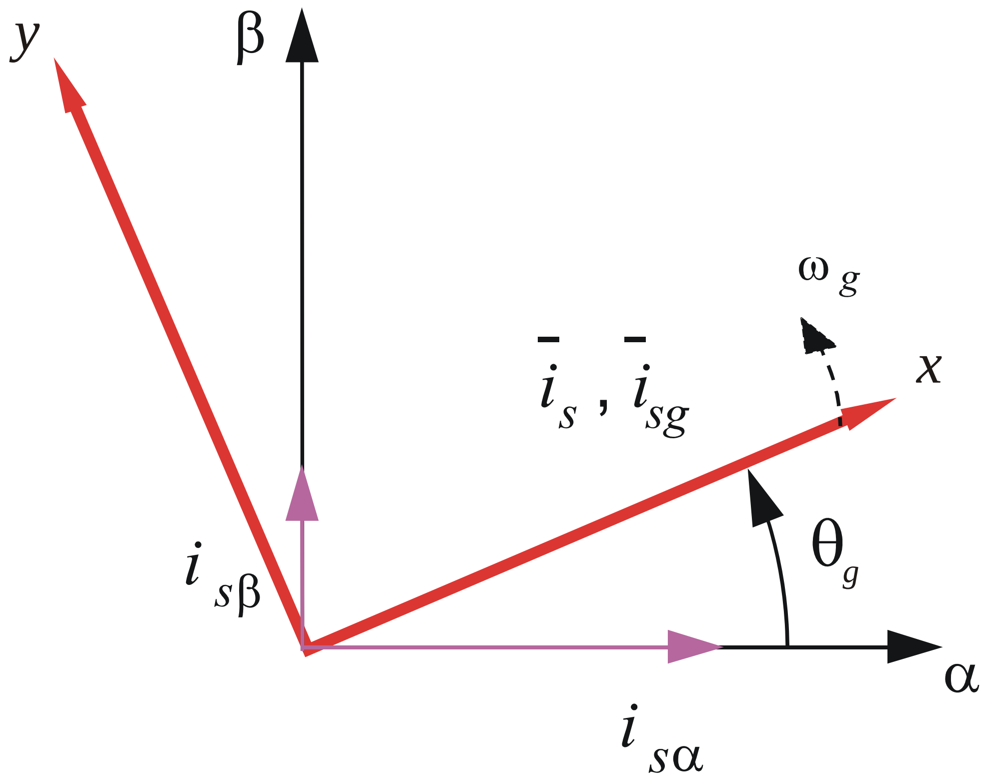


Figure 2-4 Application of the General Reference Frame

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 ($r\alpha$) of the reference frame attached to the rotor is displaced from the direct axis of the stator reference frame by the rotor angle θ_r. 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 ($r\alpha$) is θ_g-θ_r. In the general reference frame, the space vector of the rotor currents can be expressed as:

$$\overline{i_{rg}} = \bar{i}_r e^{-j(\theta_g - \theta_r)} = i_{rx} + j i_{ry} \tag{2-16}$$

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

Similarly, the space vectors of the rotor voltages and rotor flux linkages 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 motor model voltage equations in the general reference frame can be expressed. The AC induction motor model is often used in vector control algorithms. The aim of vector control is to implement control schemes which 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 d-q reference frame. The rotor flux linkage space vector is attached to the direct axis (*d*) of the coordinate system. The angular velocity of the d-q reference frame is equal to the sychronous speed of the motor. In transformation, we put $\omega_g = \omega_s$.

After transformation into d-q reference frame, the motor model is the following:

$$u_{sd} = R_s i_{sd} + \frac{d}{dt}\Psi_{sd} - \omega_s \Psi_{sq} \qquad (2\text{-}17)$$

$$u_{sq} = R_s i_{sq} + \frac{d}{dt}\Psi_{sq} + \omega_s \Psi_{sd} \qquad (2\text{-}18)$$

$$u_{rd} = 0 = R_r i_{rd} + \frac{d}{dt}\Psi_{rd} - (\omega_s - \omega)\Psi_{rq} \qquad (2\text{-}19)$$

$$u_{rq} = 0 = R_r i_{rq} + \frac{d}{dt}\Psi_{rq} + (\omega_s - \omega)\Psi_{rd} \qquad (2\text{-}20)$$

$$\Psi_{sd} = L_s i_{sd} + L_m i_{rd} \qquad (2\text{-}21)$$

$$\Psi_{sq} = L_s i_{sq} + L_m i_{rq} \qquad (2\text{-}22)$$

$$\Psi_{rd} = L_r i_{rd} + L_m i_{sd} \qquad (2\text{-}23)$$

$$\Psi_{rq} = L_r i_{rq} + L_m i_{sq} \qquad (2\text{-}24)$$

$$t_e = \frac{3}{2} p_p (\Psi_{sd} i_{sq} - \Psi_{sq} i_{sd}) \qquad (2\text{-}25)$$

The diagram in Figure 2-5 displays d-q reference frame and the relationships between the stator voltage (U_s), stator current (I_s), and the rotor, stator and magnetizing flux (Ψ_r,Ψ_s,Ψ_m). The rotor magnetizing flux space-vector is aligned to the d-axis of the d-q reference frame.

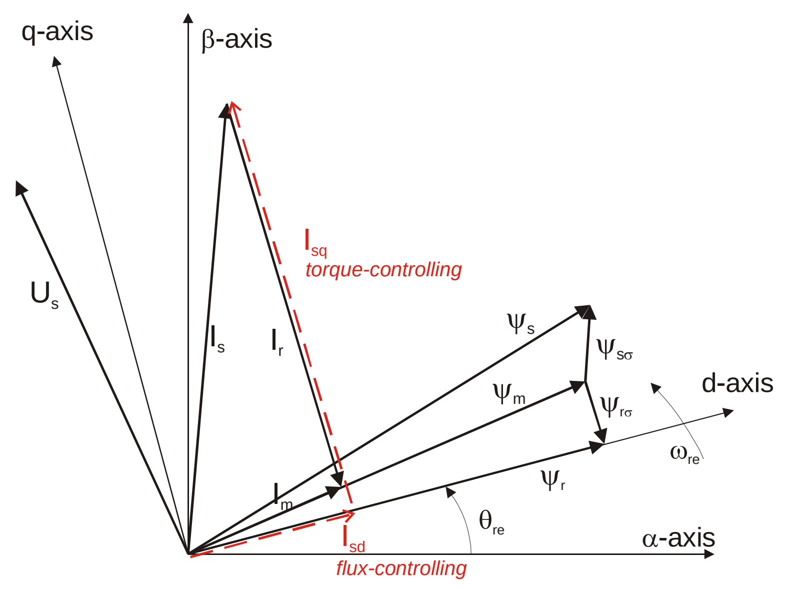


Figure 2-5 Induction Motor Space-Vector Diagram

In the steady state the AC induction motor model can be represented with the help of an equivalent steady state circuit, which is shown in Figure 2-6.

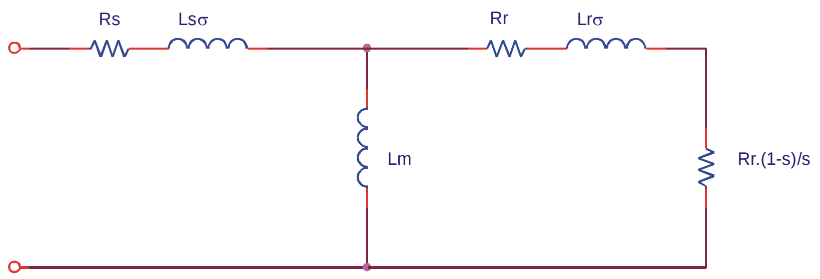


Figure 2-6 Induction Motor Equivalent Circuit

2.3 Vector Control of AC Induction Motor

2.3.1 Fundamental Principal 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, fast accelerations and decelerations. To achieve such control, vector control techniques are used for three-phase AC motors. The vector control techniques are usually also referred to field-oriented control (FOC). The basic idea of the FOC algorithm is to decompose a stator current into flux and torque producing components. 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 2-7 shows the basic structure of the vector control algorithm for the AC induction motor. To perform vector control, it is necessary to follow these steps:

- Measure the motor quantities (phase voltages and currents)

- Transform them into the 2-phase system (α,β) using a Clarke transformation
- Calculate the rotor flux space-vector magnitude and position angle
- Transform stator currents into the d-q reference frame using a Park transformation
- 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 2-phase system fixed with the stator
- Using space vector modulation, the output 3-phase voltage is generated

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 velocity information sensed by a speed or position sensor attached to the rotor. Incremental encoders or resolvers are naturally used as position transducers for vector control drives. In cost sensitive applications like washing machines, tachogenerators are widely used. In some applications the use of speed/position sensors is not desirable either. Then, the aim is not to measure the speed/position directly, but to employ some indirect techniques to estimate the rotor position instead. Algorithms which do not employ speed sensors, are called “sensorless control”.

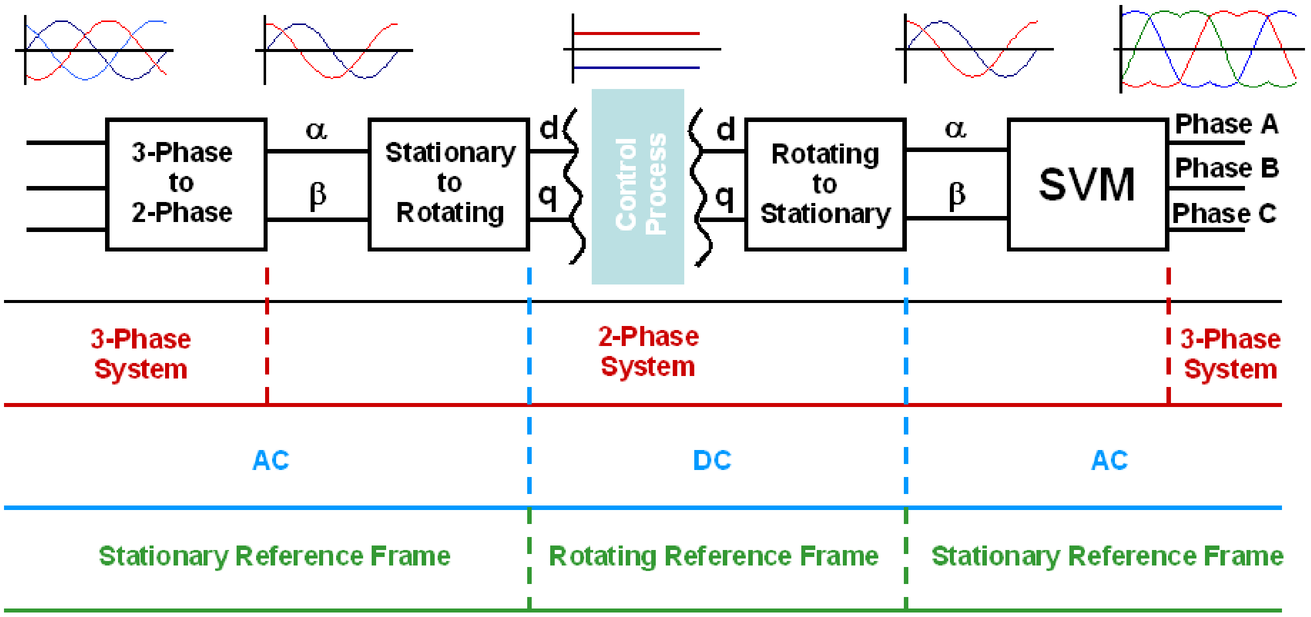


Figure 2-7 Vector Control Transformations

2.3.2 Description of the Vector Control Algorithm

The overview block diagram of the implemented control algorithm is illustrated in Figure 2-8. Similarly, as with other vector control oriented techniques, the it is able to control the excitation and torque of the induction motor separately. The aim of control is to regulate of 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 125µs period. The slow outer control loop is executed with a period of one millisecond.

To achieve the goal of the induction motor control, the algorithm utilizes a set of feedback signals. The essential feedback signals are as follows: DC-bus voltage, three-phase stator current reconstructed from the DC-bus current, and motor speed. 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 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. This includes:

- 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)

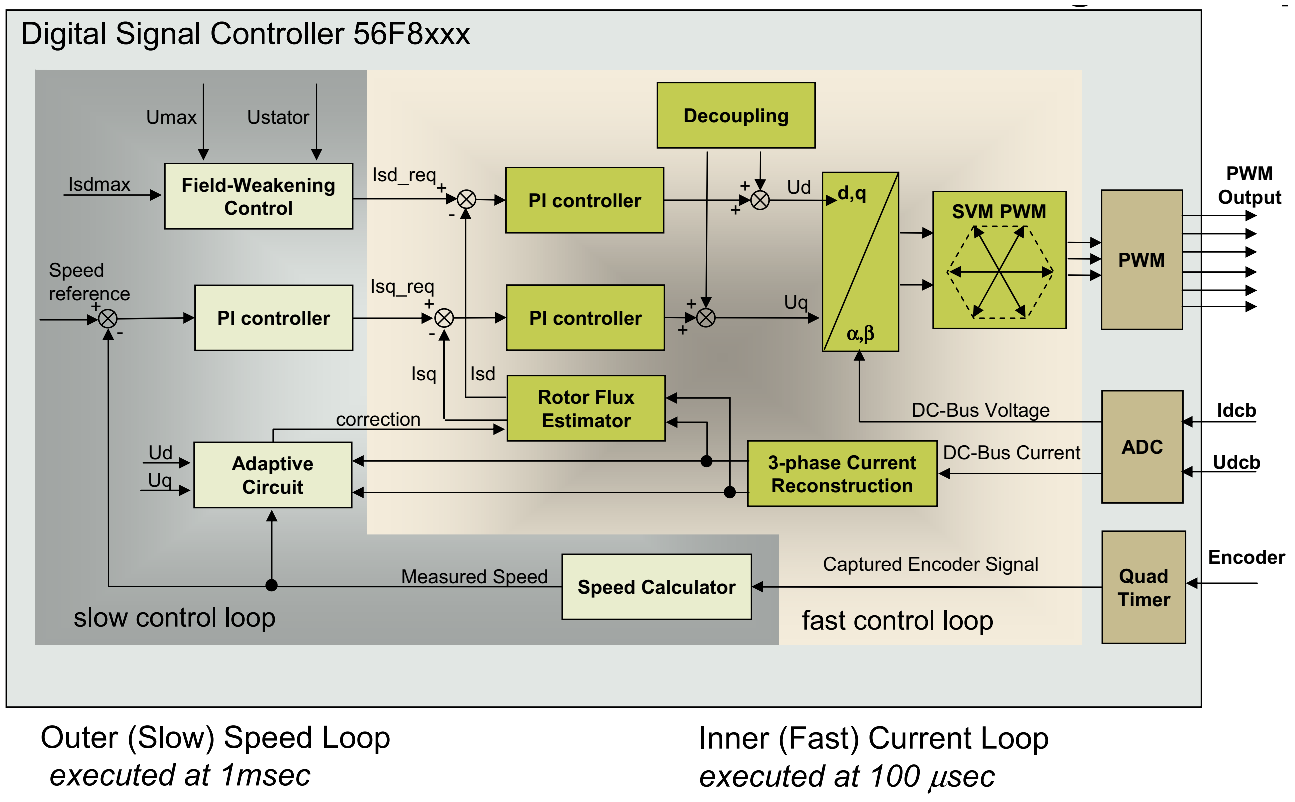


Figure 2-8 Vector Control Algorithm Overview

The slow control loop executes speed and field-weakening controllers 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}). The reference for flux producing direct axis component of the stator current (i_{sd}) is set by the Field-Weakening controller. The Adaptive Circuit performs correction on the rotor time constant to minimize the error of the rotor flux position estimation.

2.3.3 Rotor Flux Estimator

Knowledge of the rotor-flux space-vector position is vital for AC induction motor vector control. With the rotor magnetic-flux space-vector, the rotational (d-q) reference frame can be established. There are a number of methods for obtaining the rotor magnetic flux space-vector. In selecting of the most suitable algorithm we had to consider key drive requirements. In the case of the presented drive, the critical requirement we have to consider is the wide range in operating speeds (0 -18000 RPM). It is, therefore, preferable to evaluate the rotor-flux model equations in the time invariant d,q reference frame instead of the time variant α,β reference frame.

The rotor-flux components in the rotational d,q reference frame can be expressed with the help of the rotor model equations (2-19) and (2-20). These equations, however, are dependent on the rotor current currents (i_{rd},i_{rq}), which we cannot sense directly. The rotor current components can be expressed from equations (2-23) and (2-24) as follows:

$$i_{rd} = \frac{\Psi_{rd}}{L_r} - \frac{L_m}{L_r} i_{sd} \tag{2-26}$$

$$i_{rq} = \frac{\Psi_{rq}}{L_r} - \frac{L_m}{L_r} i_{sq} \tag{2-27}$$

Substituting the above equations to the rotor model equations we get:

$$\frac{d}{dt}\Psi_{rd} = \frac{L_m}{\tau_r} i_{sd} - \frac{1}{\tau_r}\Psi_{rd} + (\omega_s - \omega)\Psi_{rq} \tag{2-28}$$

$$\frac{d}{dt}\Psi_{rq} = \frac{L_m}{\tau_r} i_{sd} - \frac{1}{\tau_r}\Psi_{rq} - (\omega_s - \omega)\Psi_{rd} \tag{2-29}$$

The τ_r is a rotor time constant, which is defined as:

$$\tau_r = \frac{L_r}{R_r} \tag{2-30}$$

In the vector control algorithm we assume the rotor flux space-vector is aligned to the d-axis of the rotating reference frame. We can express this assumption as follows:

$$\Psi_{rd} = |\overline{\Psi_r}| \quad \text{and} \quad \Psi_{rq} = 0 \tag{2-31}$$

If substituting into the equations (2-28) and (2-29), we will get a single differential equation for the rotor magnetizing flux:

$$\frac{d}{dt}\Psi_{rd} = \frac{L_m}{\tau_r} i_{sd} - \frac{1}{\tau_r}\Psi_{rd} \tag{2-32}$$

So, equation (2-32) can be written for a rotor magnetizing current as well. This current is defined as:

$$\overline{i_{mr}} = \frac{\overline{\Psi_r}}{L_m} \tag{2-33}$$

The equation (2-32) written for rotor magnetizing current is:

$$\frac{d}{dt}i_{mr} = \frac{L_m}{\tau_r}i_{sd} - \frac{L_m}{\tau_r}i_{mr} = \frac{L_m}{\tau_r}(i_{sd} - i_{mr}) \tag{2-34}$$

The above equation can be easily computed by a microcontroller. We have chosen this equation for its discreteness.

To be able to evaluate the direct axis component of the stator current (i_{sd}) we need to have the position of the rotor flux space-vector. This position can be evaluated by integrating of the sum of the rotor and slip frequencies.

$$\theta_\Psi = \int_0^t (\omega_r + \omega_{slip})dt \tag{2-35}$$

The slip frequency ω_{slip} can be evaluated from actual rotor magnetizing current and quadrature-axis component of the stator current. The equation for slip frequency evaluation can be obtained combining equations (2-20), (2-24), (2-25) and (2-33). The resulting formula we get after simplification is:

$$\omega_{slip} = \frac{1}{\tau_r} \cdot \frac{i_{sq}}{i_{mr}} \tag{2-36}$$

We can substitute (2-36) into (2-35).

$$\theta_\Psi = \int_0^t \left(\omega_r + \frac{1}{\tau_r} \cdot \frac{i_{sq}}{i_{mr}}\right)dt \tag{2-37}$$

Equations (2-34) and (2-37) describe fully the rotor magnetizing flux model of the induction motor in the rotating (d,q) reference frame. The advantage of this model is the fact that it is evaluated in a time-invariant frame. The variables, which are a subjects of the integration, are represented as DC values. The convergence of the model is not influenced by the motor frequency and very simple Euler integral method can be used for the numerical evaluation. On the contrary, the derived model is heavily dependent on the rotor time-constant. The rotor time-constant varies with the motor temperature largely. To ensure a correct algorithm operation we need to use a corrective algorithm. The rotor time-constant correction algorithm is explained in a the next section.

To be able to evaluate the motor magnetizing flux model on DSC equations (2-34) and (2-37) have to be made discrete. For this, we use the Euler backward method. Let's assume that the sampling period is T_{sample}. The algorithm for numerical integration of the rotor flux equations is as follows:

$$i_{mr}^k = i_{mr}^{k-1} + T_{sample}\frac{L_m}{\tau_r}(i_{sd}^k - i_{mr}^{k-1}) \tag{2-38}$$

$$\theta_\Psi^k = \theta_\Psi^{k-1} + T_{sample}\left(\omega_r^k + \frac{1}{\tau_r} \cdot \frac{i_{sq}^k}{i_{mr}^k}\right) \tag{2-39}$$

The upper indexes k and k-1 represent corresponding variables sampled in steps k and k-1 respectively.

2.3.4 Rotor Time-Constant Correction

The rotor flux model expressed in 2.3.3 Rotor Flux Estimator features a strong dependency on the rotor time-constant. An inaccurate value of τ_r can lead to an unwanted coupling between the d and q axes. This inaccuracy may deteriorate the dynamic performance of the drive with unwanted instabilities. The problem can be avoided by the on-line adaption of the rotor time-constant.

There are a number of correction techniques described in literature. The aim of these correction techniques is to evaluate some of the motor state variables from measured quantities and to compare those variables with estimated quantities. The differences between the evaluated and estimated quantities are used to adopt the model parameters. We call these estimators closed-loop estimators. Compared to open loop estimators, the accuracy of the system can be increased.

The correction algorithm, that was designed for purpose of the presented application, is based on evaluation of the back e.m.f. components of the stator voltage. For the algorithm design we have considered the requirement of the wide speed range in the motor operation. As in case of the rotor magnetizing flux estimator, we have chosen the time-invariant d,q reference frame for evaluating the correction algorithm.

The back e.m.f. components of the stator voltage can be evaluated from the stator voltage equations (2-17) and (2-18). It is desirable to make to do the correction in the rotor time-constant within the low bandwidth control loop. The stator voltage equations can be simplified for a steady-state operation of the motor as follows:

$$u_{sd} = R_s i_{sd} - \omega_s \Psi_{sq} \tag{2-40}$$

$$u_{sq} = R_s i_{sq} + \omega_s \Psi_{sd} \tag{2-41}$$

The above equations are a function of the stator magnetizing flux Ψ_S. The control algorithm is oriented on the rotor magnetizing flux Ψ_r. The relationship between rotor and stator flux can be derived from equations (2-21)—(2-24). After simplification we get:

$$\Psi_{sd} = \left(\frac{L_s L_r - L_m^2}{L_r}\right) i_{sd} + \frac{L_m}{L_r} \Psi_{rd} \tag{2-42}$$

$$\Psi_{sq} = \left(\frac{L_s L_r - L_m^2}{L_r}\right) i_{sq} + \frac{L_m}{L_r} \Psi_{rq} \tag{2-43}$$

Substituting the above equations into (2-40) and (2-41), we get the stator voltage equations as a function of the stator current and rotor magnetizing flux.

$$u_{sd} = R_s i_{sd} - \omega_s \left(\frac{L_s L_r - L_m^2}{L_r}\right) i_{sq} - \omega_s \frac{L_m}{L_r} \Psi_{rq} \tag{2-44}$$

$$u_{sq} = R_s i_{sq} + \omega_s \left(\frac{L_s L_r - L_m^2}{L_r}\right) i_{sd} + \omega_s \frac{L_m}{L_r} \Psi_{rd} \tag{2-45}$$

For the rotor magnetizing flux oriented control we assume conditions (2-31). Then, the resulting stator voltage equations are as follows:

$$u_{sd} = R_s i_{sd} - \omega_s \left(\frac{L_s L_r - L_m^2}{L_r}\right) i_{sq} \tag{2-46}$$

$$u_{sq} = R_s i_{sq} + \omega_s \left(\frac{L_s L_r - L_m^2}{L_r}\right) i_{sd} + \omega_s \frac{L_m}{L_r} \Psi_{rd} \tag{2-47}$$

The direct axes voltage equation (2-46) is a function of stator current components (i_{sd}, i_{sq}), stator winding resistance (R_s) and motor inductances (L_s, L_r, L_m). It is not a function of the rotor time-constant. This condition (2-46) can be used for adapting the rotor time-constant. We can evaluate the error signal according to (2-48).

$$u_{sd} - \left(R_s i_{sd} - \omega_s \left(\frac{L_s L_r - L_m^2}{L_r}\right) i_{sq}\right) = \text{error} \tag{2-48}$$

The error signal is an input to the PI controller. The PI controller keeps the error signal at zero, adjusting the rotor time-constant in the rotor magnetizing-flux position estimator equation (2-39). The complete algorithm for the rotor flux estimation, including rotor time constant correction, is shown in Figure 2-9.

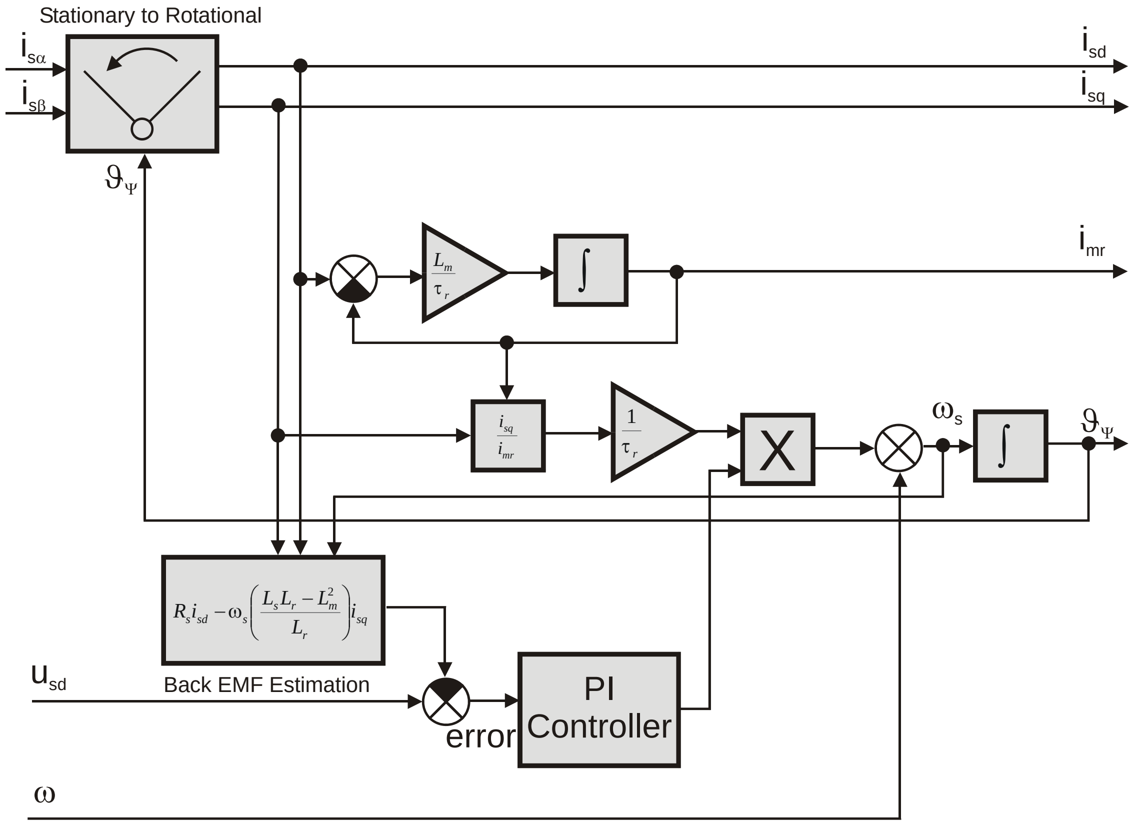


Figure 2-9 Rotor Mag. Flux Estimator

The input variables into the estimator are the stator current components in the stationary α,β reference frame (i_α,i_β), direct axis of the stator voltage in the rotating reference frame (u_d) and the actual rotor speed (ω). Output of the algorithm is the magnetizing rotor current (i_{mr}) and the rotor magnetizing flux position (ϑ_Ψ). The direct and quadrature axis components of the stator current (i_{sd}, i_{sq}), which are obtained after transformation into the rotating reference frame, are used as feedback signals for the corresponding PI controllers, see Figure 2-8.

2.3.5 Stator Voltage Decoupling

For purposes of rotor magnetizing-flux oriented vector control, the direct-axis stator current i_{sd} (rotor flux-producing 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 induction 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 as follows:

$$u_{sd} = u_{sd}^{lin} + u_{sd}^{decouple} \tag{2-49}$$

$$u_{sq} = u_{sq}^{lin} + u_{sq}^{decouple} \tag{2-50}$$

The decoupling components $u_{sd}^{decouple}, u_{sq}^{decouple}$ are evaluated from the stator voltage equations (2-17) and (2-18). They eliminate cross-coupling for current control loops at a given motor operating point. Linear components $u_{sd}^{lin}, u_{sq}^{lin}$ are set by the outputs of the current controllers. The voltage decoupling components are evaluated according to the following equations:

$$u_{sd}^{decouple} = R_s i_{sd} - p_p \omega (L_{s\sigma} + L_{r\sigma}) i_{sq} \tag{2-51}$$

$$u_{sq}^{decouple} = R_s i_{sq} + p_p \omega (L_{s\sigma} + L_{r\sigma}) i_{sd} + p_p \omega L_m i_{mr} \tag{2-52}$$

The above equations (2-51) and (2-52) are evaluated in the *Decoupling* block (see Figure 2-8).

2.3.6 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 3-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 depicted in Figure 2-10.Regarding the 3-phase power stage configuration, as shown in Figure 2-10, 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 2-11. 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.

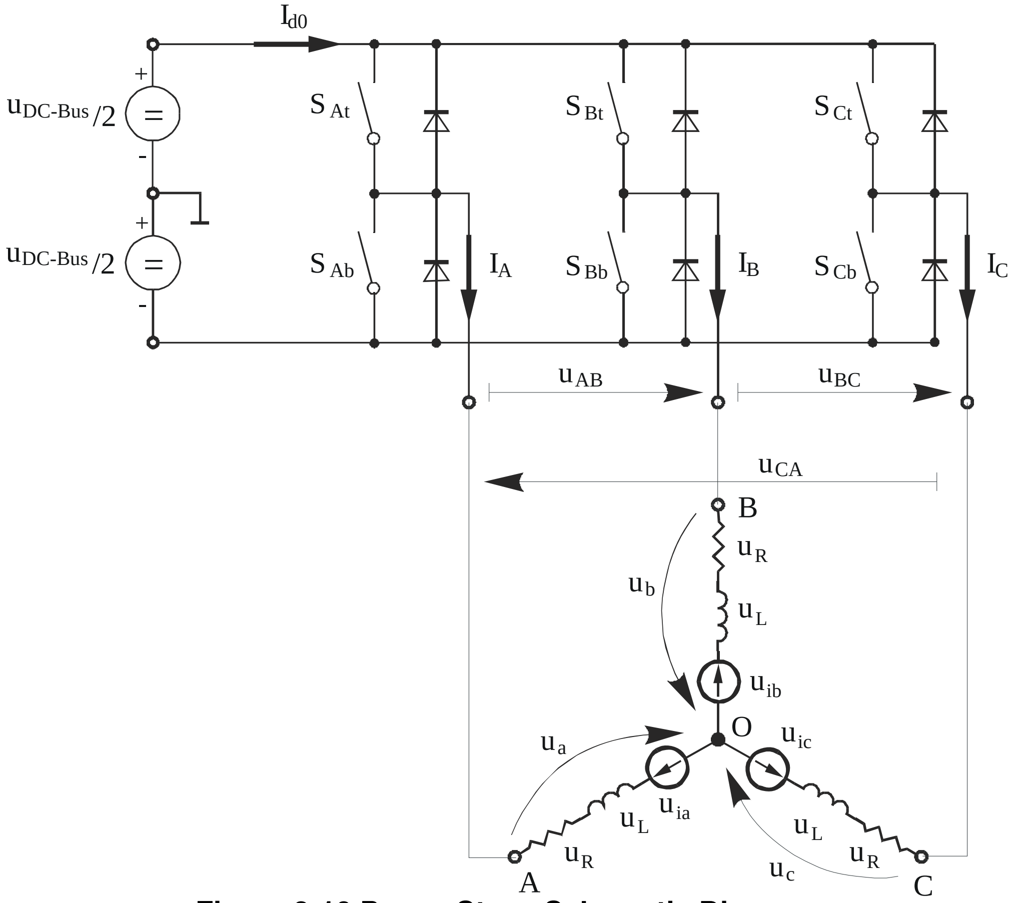


Figure 2-10 Power Stage Schematic Diagram

The combination of ON/OFF states in the power stage switches for each voltage vector is coded in Figure 2-11 by the three-digit number in parenthesis. Each digit represents one phase. For each phase, a value of one means that the upper switch is ON and the bottom switch is OFF. A value of zero means that the upper switch is OFF and the bottom switch is ON. These states, together with the resulting instantaneous output line-to-line voltages, phase voltages and voltage vectors, are listed in Table 2-1.

Table 2-1 Switching Patterns and Resulting Instantaneous Line-to-Line and Phase Voltages

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}

Table 2-1 Switching Patterns and Resulting Instantaneous Line-to-Line and Phase Voltages (Continued)

a	b	c	U_a	U_b	U_c	U_{AB}	U_{BC}	U_{CA}	Vector
0	0	1	$-U_{DC-Bus}/3$	$-U_{DC-Bus}/3$	$2U_{DC-Bus}/3$	0	$-U_{DC-Bus}$	U_{DC-Bus}	U_{300}
1	0	1	$U_{DC-Bus}/3$	$-2U_{DC-Bus}/3$	$U_{DC-Bus}/3$	U_{DC-Bus}	$-U_{DC-Bus}$	0	U_{360}
1	1	1	0	0	0	0	0	0	O_{111}

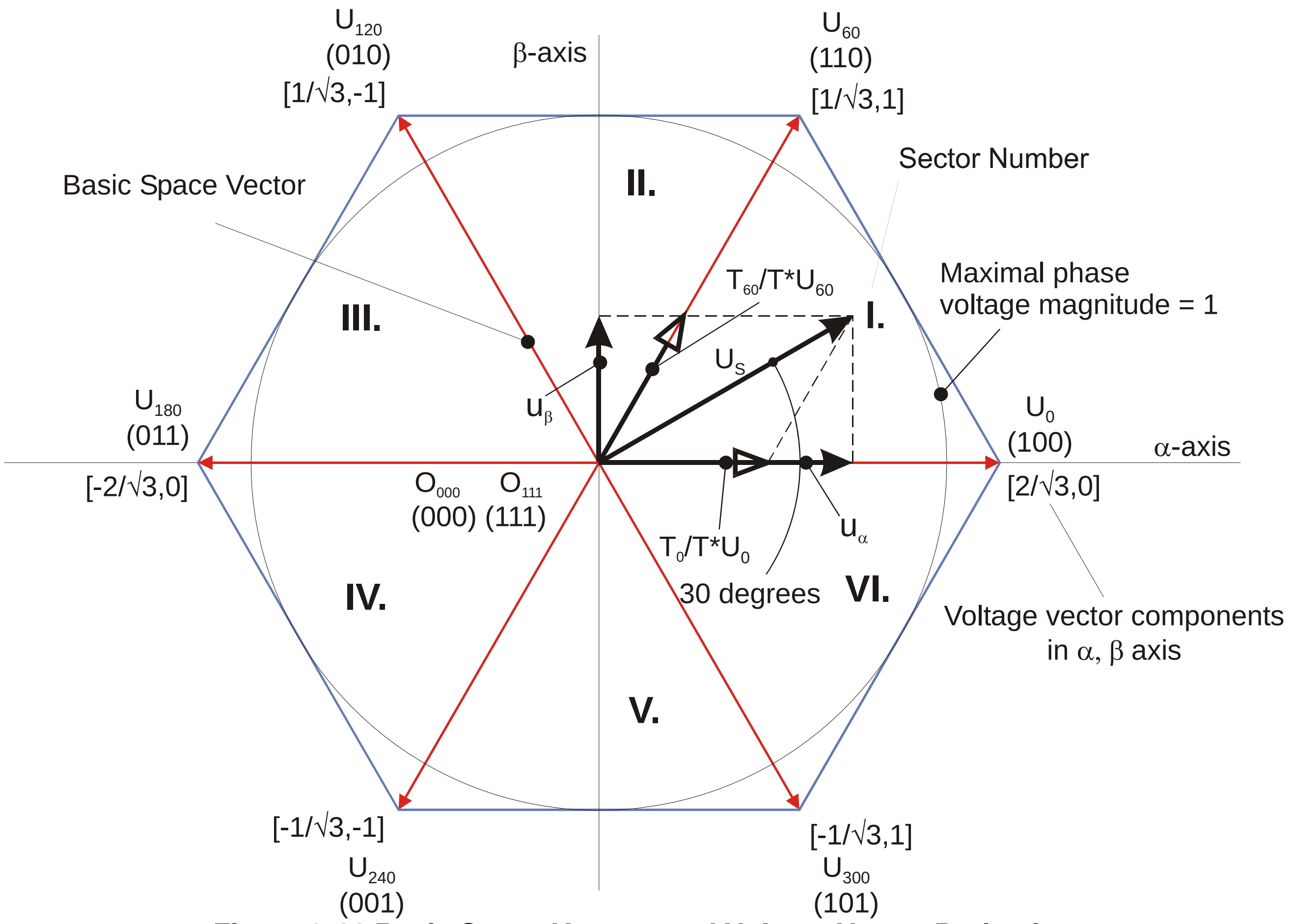


Figure 2-11 Basic Space Vectors and Voltage Vector Projection

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 centre-aligned PWM, edge-aligned PWM, minimal switching, etc.

For the chosen SVM, we define the following 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:

$$T_{PWM} \cdot U_{S[\alpha,\beta]} = T_1 \cdot U_X + T_2 \cdot U_{X\pm60} + T_0 \cdot (O_{000} \vee O_{111}) \tag{2-53}$$

$$T_{PWM} = T_1 + T_2 + T_0 \tag{2-54}$$

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 (2-53) splits into equations (2-55) and (2-56).

$$T_{PWM} \cdot U_{SX} = T_1 \cdot U_X \tag{2-55}$$

$$T_{PWM} \cdot U_{S(X\pm60)} = T_2 \cdot U_{X\pm60} \tag{2-56}$$

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.

$$T_1 = \frac{|U_{SX}|}{|U_X|} T_{PWM} \quad \text{for vector } U_x \tag{2-57}$$

$$T_2 = \frac{|U_{SX}|}{|U_{X\pm60}|} T_{PWM} \quad \text{for vector } U_{x\pm60} \tag{2-58}$$

$$T_0 = T_{PWM} - (T_1 + T_2) \quad \text{either for } O_{000} \text{ or } O_{111} \tag{2-59}$$

Chapter 3
System Concept

3.1 System Specification

The system is designed to drive a three-phase AC induction motor. The application meets the following performance specification:

- Targeted at the MC56F8013/23 Controller Board
- Running on a 3-phase High Voltage Power Stage
- Control technique incorporating:
  - Vector control of three-phase AC induction motor with position encoder
  - Closed-loop speed control
  - Both directions of rotation
  - Both motor and generator modes
  - Reconstruction of three-phase motor currents from DC-Bus shunt resistor
  - Closed loop current control
  - Flux and torque independent control
  - Adaptive rotor flux space-vector estimator
  - Field-weakening for high speeds
  - High speed range, max speed – 18000 RPM (2-pole motor)
- 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 overvoltage and undervoltage, overcurrent protection

3.2 Application Description

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

- Power supply 90V - 260V AC RMS, 5A
- 3-phase High Voltage Power Stage
- Three-Phase AC Induction Motor (default configuration for motor Elektrim SKh 71 - 4A2)
- MC56F8013/23 Controller Board

The MC56F8013/23 Controller Board executes the control algorithm. In response to the user interface and feedback signals, it generates PWM signals for the 3-phase High Voltage Power Stage. High-voltage waveforms generated by the DC to AC inverter are applied to the motor.

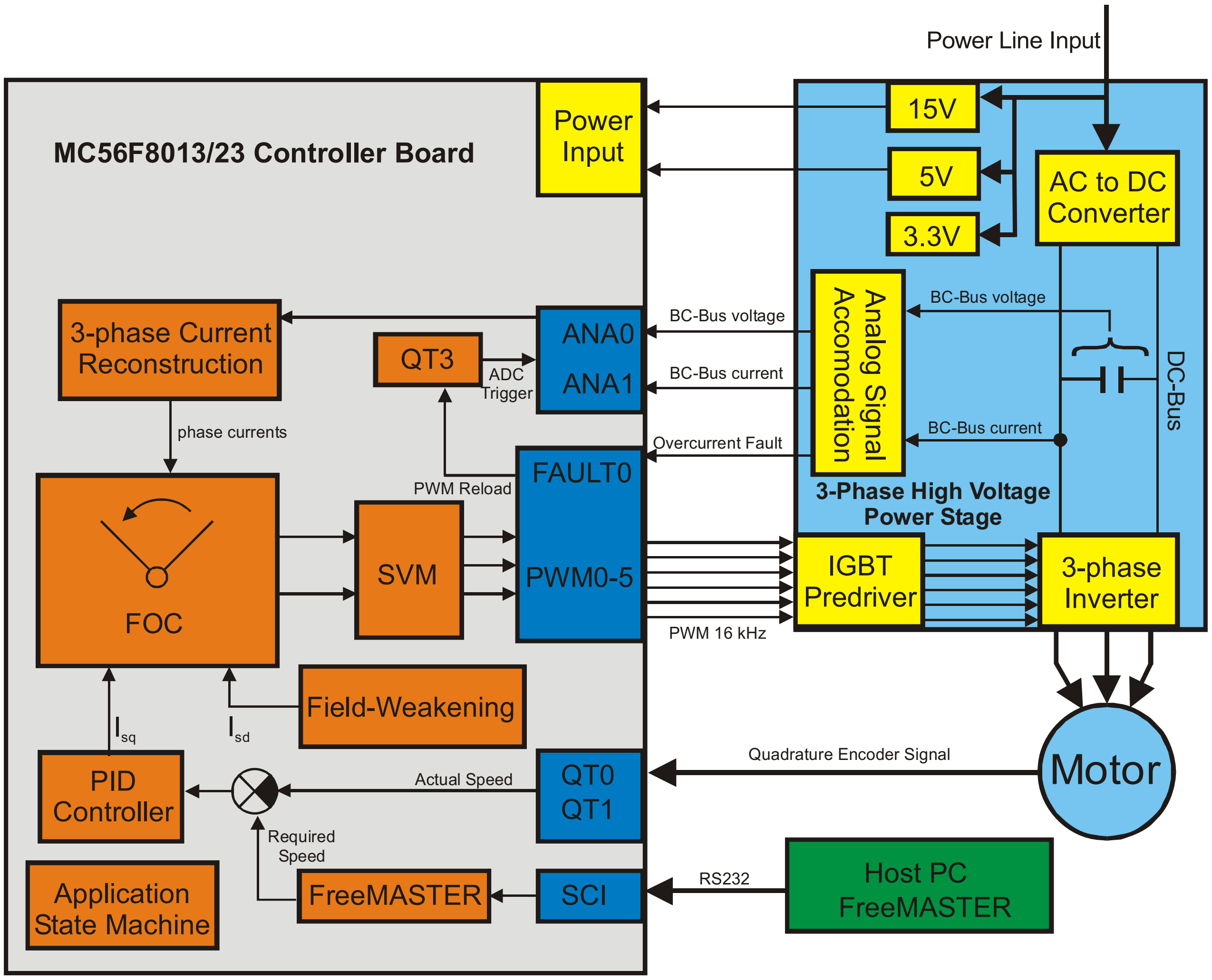


Figure 3-1 System Concept

3.3 Control Process

The state of the user interface is scanned periodically, while the actual speed of the motor, DC-Bus voltage and current 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 reference for the rotor magnetizing-flux producing component of the stator current is determined by the field-weakening algorithm.

The DC-Bus current and voltage is 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-Bus shunt resistor. The reconstructed three-phase current is then transformed into space vectors and used by the FOC algorithm.

Based on feedback signals, the FOC algorithm performs a vector control technique oriented to the rotor magnetizing flux space-vector as described in 2.3.2 Description of the Vector Control Algorithm. Two independent current PI control loops are executed to achieve the desired behavior of the motor. Output from the FOC is a stator voltage space-vector, 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 powerstage terminals.

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

The state machine of the drive handles the operating states of the drive. There are four states of the drive: RUN, STOP, FAULT and INIT. The actual operating state is indicated by the FreeMASTER control page.

In the case of overvoltage, undervoltage or overcurrent, the signals for the 3-phase inverter are disabled and the fault state is displayed.

Chapter 4
Hardware

4.1 Hardware Implementation

The application is designed to drive a 3-phase AC motor. It consists of the following modules:

- Host PC
- MC56F8013/23 Controller Board
- 3-phase AC/BLDC High Voltage Power Stage
- 3-phase AC Induction Motor

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

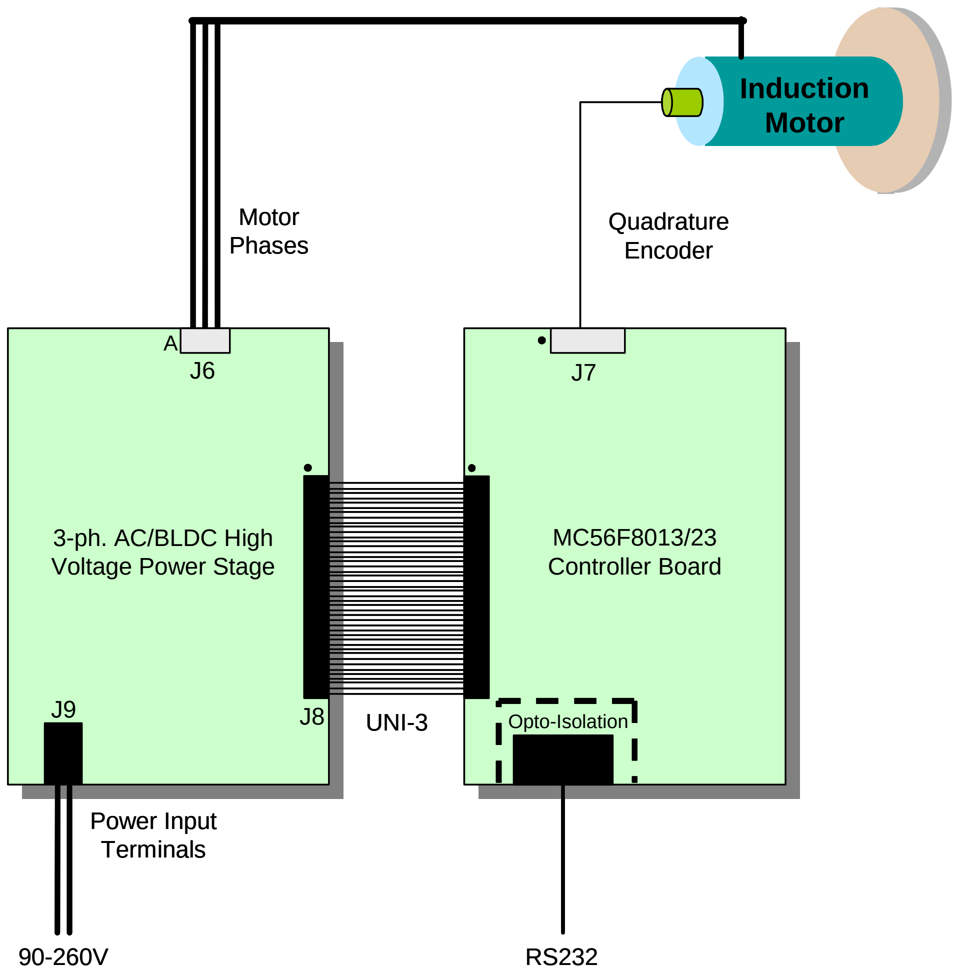


Figure 4-1 Hardware System Configuration

All system parts are supplied and documented in these references:

- MC56F8013/23 Controller Board:
  - Using Freescale's MC56F8013 or MC56F8023 as the controller
  - Supplied as an MC56F8013/23 Controller Board
  - Described in the *MC56F8013/23 Controller Board User's Manual*
- 3-Phase AC/BLDC High Voltage Power Stage:
  - High-voltage three-phase power stage with single-phase input 115/230 Volt AC and 750 VoltAmp variable voltage three-phase IGBT bridge output.
  - Described in the *3-Phase AC/BLDC High Voltage Power Stage User's Manual*

A detailed description of each individual board can be found in the appropriate user manual, or on the Freescale web site http://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 using the MC56F8013/23 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 analogue power supplies, and peripheral expansion connectors. The expansion connectors are for signal monitoring and user feature expandability. Test pads are provided for monitoring critical signals and voltage levels.

The MC56F8013/23 controller board is designed for the following purposes:

- To allow new users to become familiar with the features of the MC56F801x/802x architecture.
- To 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 simplifies considerably the task of the developer.
- To serve as a platform for hardware development. The hardware platform enables external hardware modules to be connected. The OnCE port's unobtrusive design means all of the memory on the digital signal controller chip is available to the user.

The board facilitates the evaluation of various features present in the MC56F8013/8023,and can be used to develop real-time software and hardware products based on the MC56F8013 or the MC56F8023. It provides the features necessary to write and debug software, demonstrate the functionality of that software, and to interface with the customer's application specific device(s). 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 .

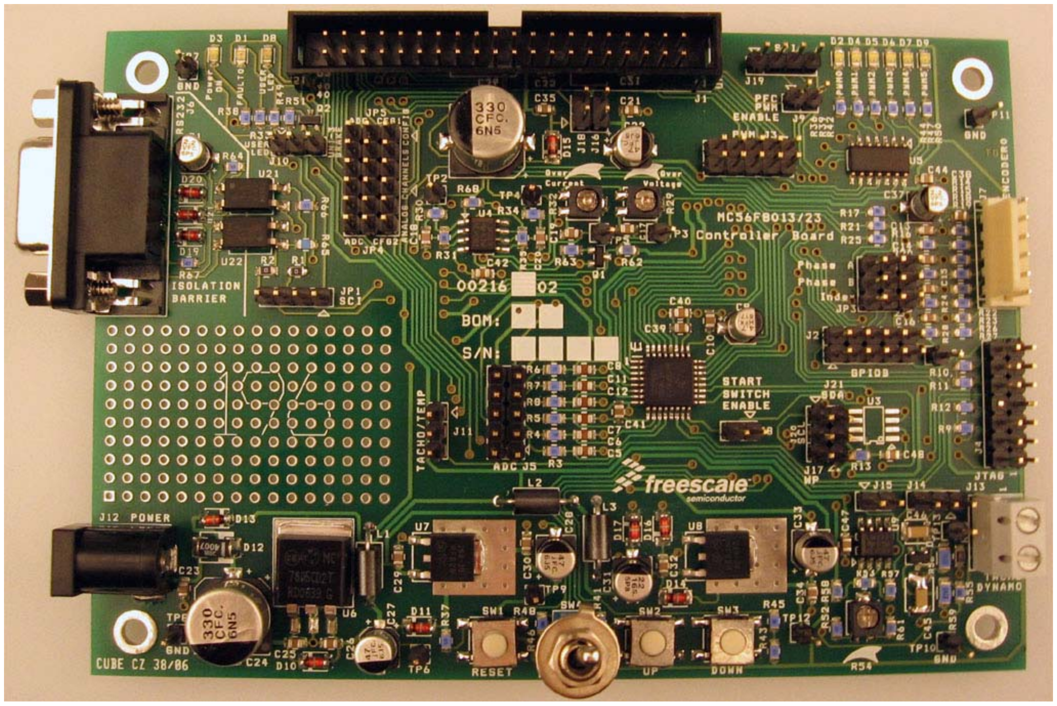


Figure 4-2 MC56F8013/23 Controller Board Top View

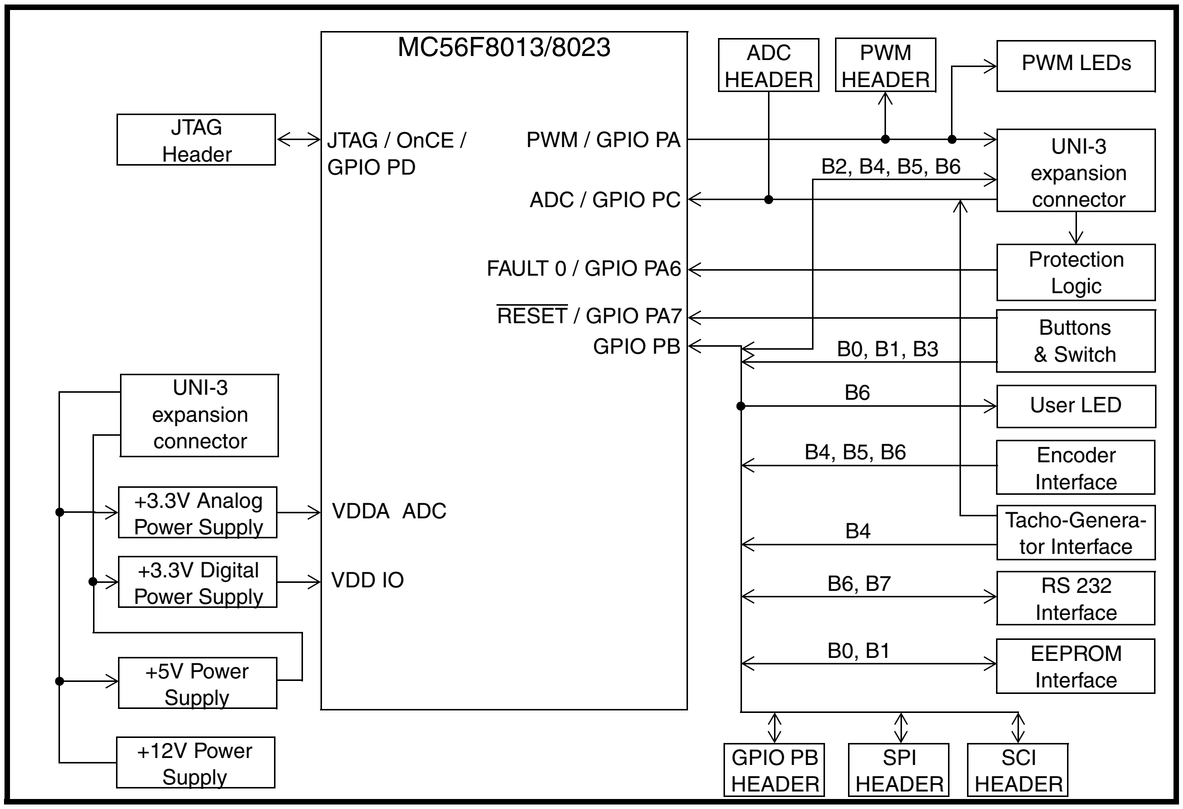


Figure 4-3 Block Diagram of the MC56F8013/23 Controller Board

4.3 3-Phase AC/BLDC High Voltage Power Stage

Freescale’s three-phase high-voltage (HV) AC power stage is a 750-voltamps (one horsepower), 3-phase power stage that will operate off DC input voltages from 140 to 325 volts, and AC line voltages from 100 to 240 volts. In combination with one of the controller boards, it provides a software development platform that allows algorithms to be written and tested without the need to design and build a power stage. It supports a wide variety of algorithms for both AC induction and brushless DC (BLDC) motors.

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

Figure 4-5 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 overcurrent trip point is set at 10 amps.

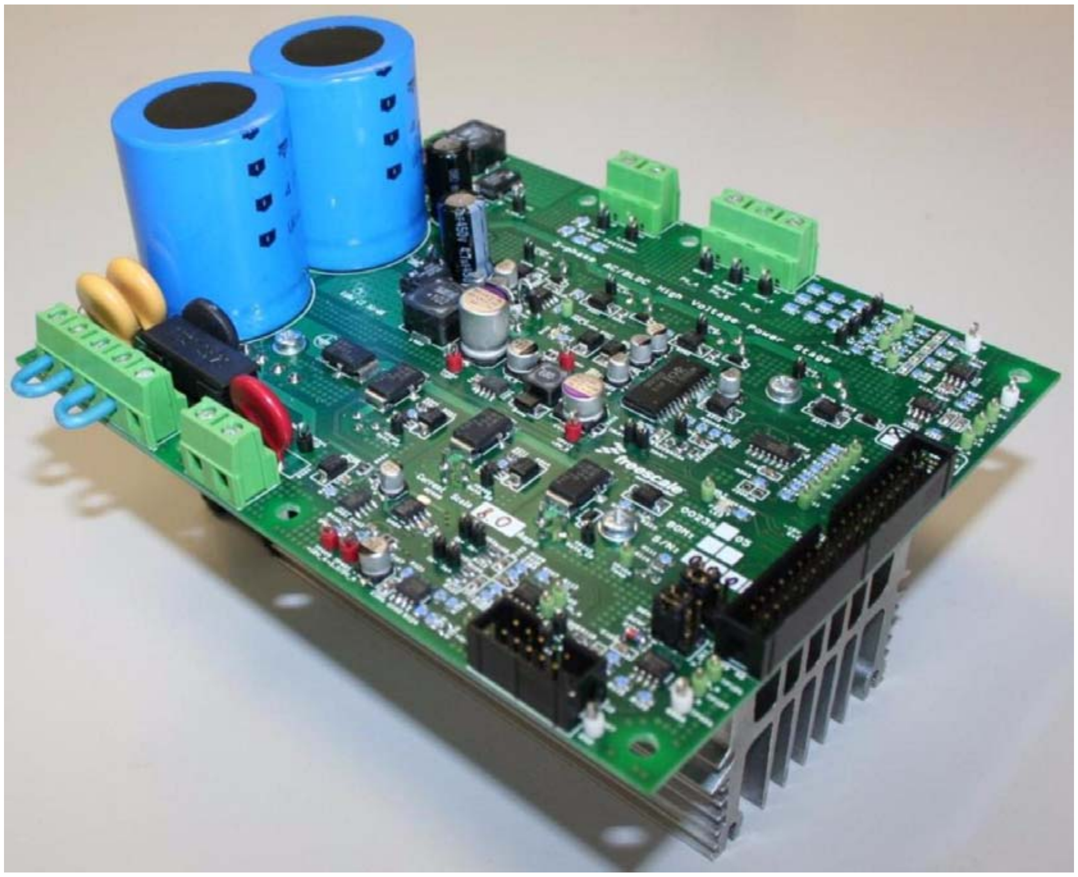


Figure 4-4 3-Phase AC/BLDC High Voltage Power Stage

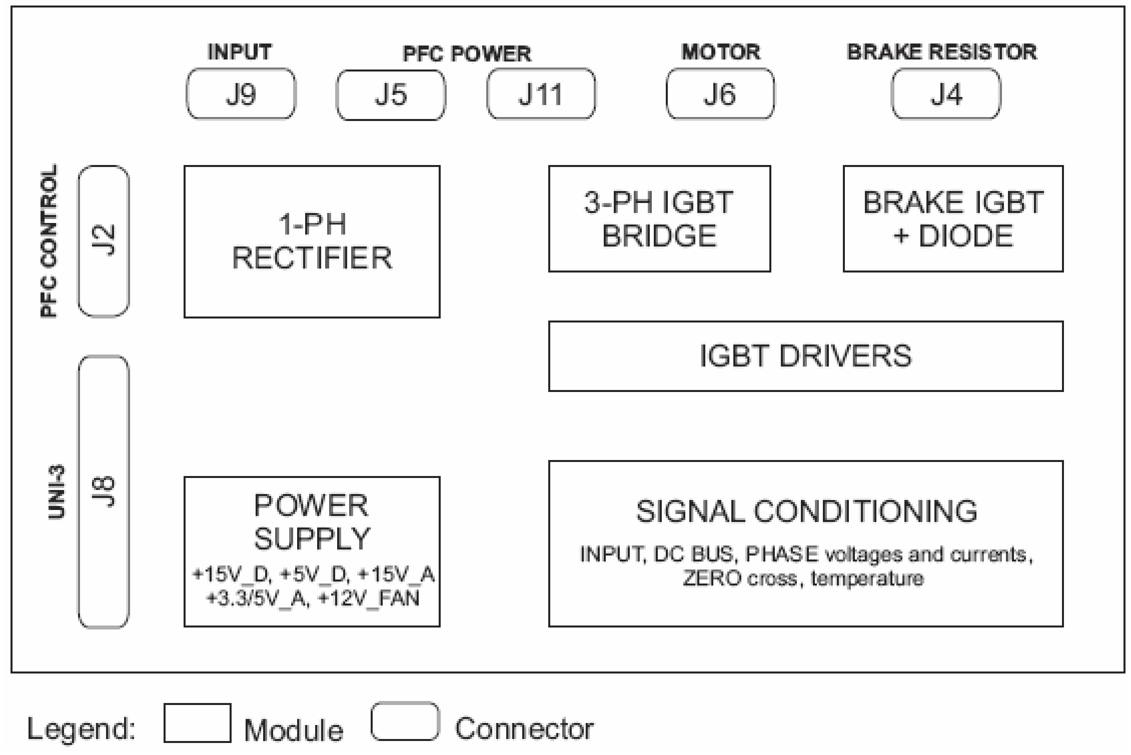


Figure 4-5 3-Phase AC/BLDC Power Stage Block Diagram

Table 4-1 Electrical Characteristics of 3-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
Analogue Output Range*	V_{Out}	0	—	3.3	V
Bus Current Sense Voltage	I_{Sense}	—	206.25	—	mV/A
Bus Current Sense Offset	I_{offset}		$+V_{REF}$		V
Bus Voltage Sense Voltage	V_{Bus}	—	8.09	—	mV/V
Bus Voltage Sense Offset	V_{offset}		0		V
Continuous Output Current **)	I_C	—	—	10	A
Deadtime (built in IR2133)	t_{off}	—	250	—	ns

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

4.4 Motor Specifications — Example

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

Table 4-2 Specifications of the Motor and Incremental Sensor

	Motor Type:	**3-Phase AC Induction Motor ELEKTRIM SKh 71 - 4A2**
Motor Specification:	Nominal Voltage (line-to-line)	380V RMS
	Nominal Speed	1380 RPM
	Nominal Current (phase)	0.85A RMS
	Nominal Power	0.25kW/0.33HP
	Nominal cos ϕ	0.68
Motor Model Parameters	Stator Winding Resistance	30.6 Ohm
	Rotor Winding Resistance	29.6 Ohm
	Stator Winding Leakage Inductance	61.4 mH
	Rotor Winding Leakage Inductance	143.3 mH
	Motor Magnetizing Inductance	1090 mH
	Number of Pole-Pairs	2
Position Sensor Specification:	Manufacturer:	HEIDENHAIN
	Type:	ROD 426-3600-03R
	Line Count	3600
	Output	5V ± 10% TTL

Chapter 5
Software Design

5.1 Introduction

This section describes the software design of the AC induction vector control drive 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 in understanding of the designed software.

5.2 Application Variables Scaling

5.2.1 Fractional Numbers Representation

The AC induction motor 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:

$$-1.0 \le SF \le +1.0 - 2^{-[N-1]} \quad (5\text{-}1)$$

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

5.2.2 Scaling of Analogue Quantities

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

$$\text{Fractional Value} = \frac{\text{Real Value}}{\text{Real Quantity Range}} \quad (5\text{-}2)$$

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.]

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

$$(\text{Frac16})\text{voltage\_variable} = \frac{V_{MEASURED}}{V_{MAX}} \quad (5\text{-}3)$$

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

$$(\mathrm{Int16})\mathrm{voltage\_variable} = (\mathrm{Frac16})\mathrm{voltage\_variable} \cdot 2^{15} \tag{5-4}$$

The maximum range of analogue quantities used by the application is defined by #define statements in the application configuration files. The application configuration files are listed below:

```
boardparams.h
motorparams.h
```

The default scaling ranges for the reference design hardware set-up are as follows:

```
#define VOLTAGE_SCALE          407.0      /* Volts */
#define CURRENT_SCALE          8.0        /* Amps */
#define FREQ_STATOR_MAX        300.0      /* Hz */
#define MOTOR_REVOLUTIONS_MAX  3000.0     /* RPM */
```

Please note, that CURRENT_SCALE 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 (=1.65V). Thus, the maximum positive and negative phase-current which can be sensed is CURRENT_SCALE/2.In other words, if the current sensing range of the power stage is from -4.0 Amps to

+ 4.0 Amps, the value of CURRENT_SCALE is set to the value 8.0 Amps. And if the current sensing range of the power stage is from -8.0 Amps to + 8.0 Amps, the value of CURRENT_SCALE is set to the value 16.0 Amps

5.2.3 Scaling of Angles

The angles, such as rotor flux position, are represented as a 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:

$$-\mathrm{pi} \approx 0\mathrm{x}8000 \tag{5-5}$$

$$\mathrm{pi} \cdot (1.0 - 2^{-15}) \approx 0\mathrm{x}7\mathrm{FFF} \tag{5-6}$$

5.2.4 Scaling of Parameters

Real-value parameters in equations such as the rotor flux estimator, decoupling voltage, etc. are represented as 16-bit signed fractional values in the range [-1,1). The real parameter value (Ohms, Henry) has to be adjusted to correspond to the scaling range of the analogue values, which forms the particular equation. The adjusted value is then scaled using an N-bit shift to fit the fractional range of [-1,1). The scaling process can be explained in a simple example of Ohms law equation.

$$V^{\mathrm{real}} = R \cdot I^{\mathrm{real}} \tag{5-7}$$

$$V^{\mathrm{Frac16}} \cdot \mathrm{V\_MAX} = R \cdot I^{\mathrm{Frac16}} \cdot \mathrm{I\_MAX} \tag{5-8}$$

$$V^{\mathrm{Frac16}} = \left(R \cdot \frac{\mathrm{I\_MAX}}{\mathrm{V\_MAX}}\right) \cdot I^{\mathrm{Frac16}} = R^{\mathrm{adjusted}} \cdot I^{\mathrm{Frac16}} \tag{5-9}$$

Let’s substitute the following values:

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

The R^{adjusted} can be evaluated:

$$R^{\mathrm{adjusted}} = R \cdot \frac{\mathrm{I\_MAX}}{\mathrm{V\_MAX}} = 300 \cdot \frac{8}{407} = 5.8968 \tag{5-10}$$

The $R^{adjusted}$ is out of range of the signed fractional number. We need to right shift the value by N-bits to fit into the desired range. For this example we have to shift the result by N = 3 bits. The resistor value scaled to signed fract range is as follows:

$$R^{Frac16} = R \cdot \frac{I\_MAX}{V\_MAX} \cdot 2^{-N} = 300 \cdot \frac{8}{407} \cdot 2^{-3} = 0.7371 \qquad (5\text{-}11)$$

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

$$V^{Frac16} = \left(\left(R \cdot \frac{I\_MAX}{V\_MAX} \cdot 2^{-N}\right) I^{Frac16}\right) \cdot 2^{N} \qquad (5\text{-}12)$$

Please note, that the final multiplication result has to be left-shifted back by N bits to stay within the proper range of the V^{Frac16} variable.

All algorithm and motor parameters are scaled to their 16-bit fractional representation in the `motorparams.h` header file. For most parameters there are two definitions. One evaluates the parameter fractional representation, and the other defines the required N-bit shift. Let's show the scaling constants for the stator voltage decoupling equations (2-51) and (2-52) from `motorparams.h` as an example.

```
/* N-bit right shift constants */
#define DECOUPLE_LLEAK_SCALE 3
#define DECOUPLE_LM_SCALE 6

/* Fractional representation of constants for decoupling equations*/
#define DECOUPLE_RS   FRAC16(R_S*CURRENT_SCALE/VOLTAGE_SCALE)
#define DECOUPLE_LLEAK FRAC16(2*PI*(L_SL+L_RL)*CURRENT_SCALE*
                             \FREQ_STATOR_MAX/(VOLTAGE_SCALE*(1<<DECOUPLE_LLEAK_SCALE)))
#define DECOUPLE_LM FRAC16(2*PI*L_M*CURRENT_SCALE*FREQ_STATOR_MAX/(VOLTAGE_SCALE*(1<<DECOUPLE_LM_SCALE)))
```

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

It is recommended to use the **ScalingTool.XLS** spreadsheet to evaluate N-shift scales. This scaling tool is located in the **{Project}ScalingTool** folder.

5.3 Application Overview

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

The **QuadTimer (TMR) channel 2** 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 125μs period. 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-Bus current samples.

The **PWM Fault** interrupt service routine is executed on an overcurrent event to handle an overcurrent fault condition. It is executed only if the fault condition occurs.

The **background** loop is executed in the application main. It handles time non-critical tasks, such as the application state machine and FreeMASTER communication polling.

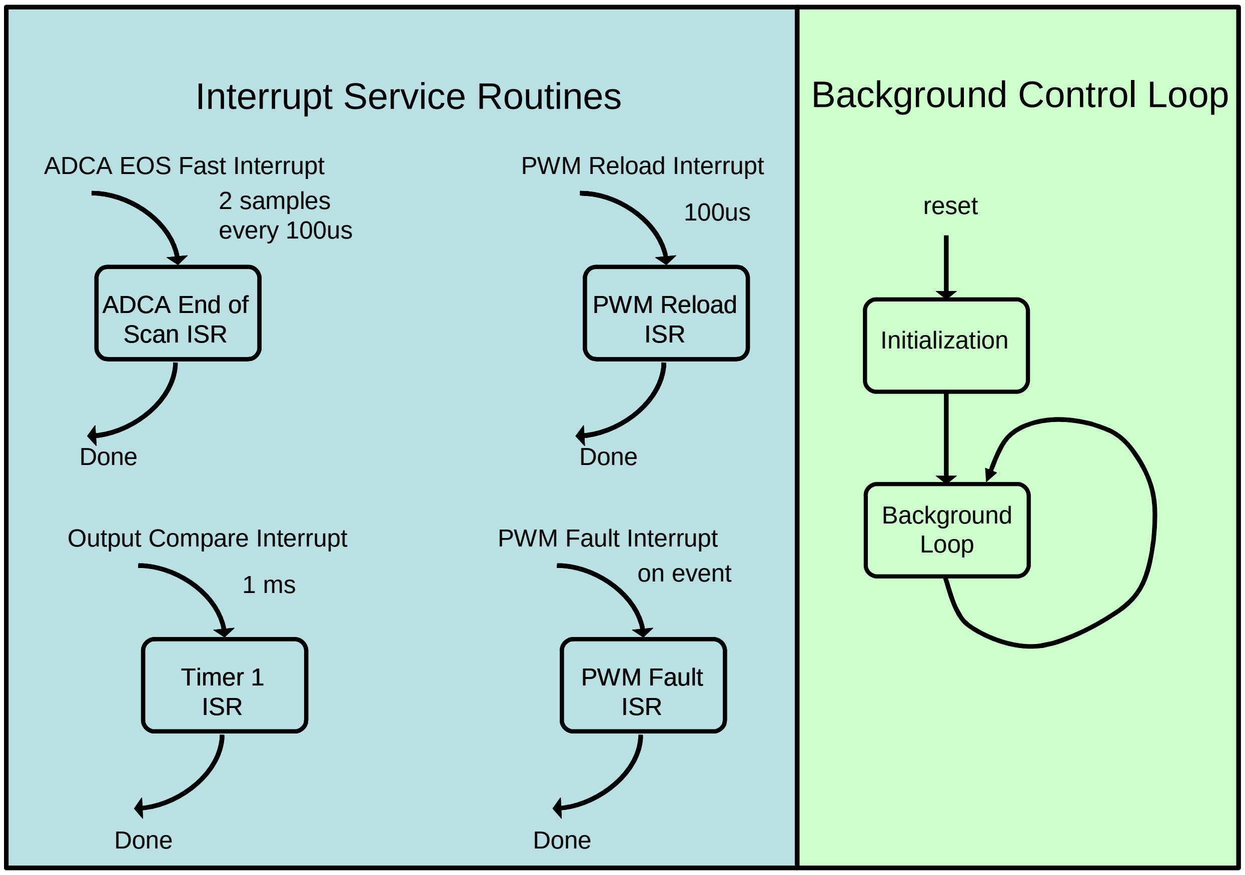


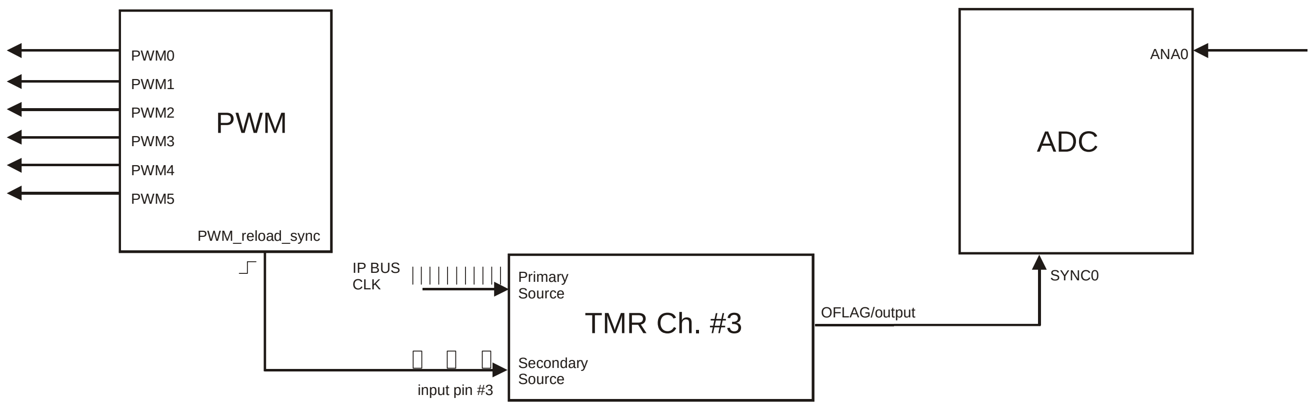
Figure 5-1 Main Data Flow

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

5.3.1 ADC End of Scan and PWM Reload Interrupts Timing

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-Bus 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 centre-aligned mode with counter modulo CMOD = 800, which corresponding to a switching frequency of 16kHz at a 32 MHz bus clock (PWM cycle period = 62.5μs). The PWM_reload_sync signal is generated every second PWM cycle with a 125μs period. The PWM_reload_sync is connected to a secondary input pin#3, 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 the 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.



TMR generates SYNC0

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:

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. A 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 ch. 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 ch. 3. An output (OFLAG) of the TMR ch. 3 is forced to zero. EOSI1 flag is cleared.
7. The second compare on Compare 1 (COMP1) occurs. Sequence of events 3 to 6 is repeated. Before ADC EOS ISR is exited, the TMR ch. 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 ch. 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 at point #1.

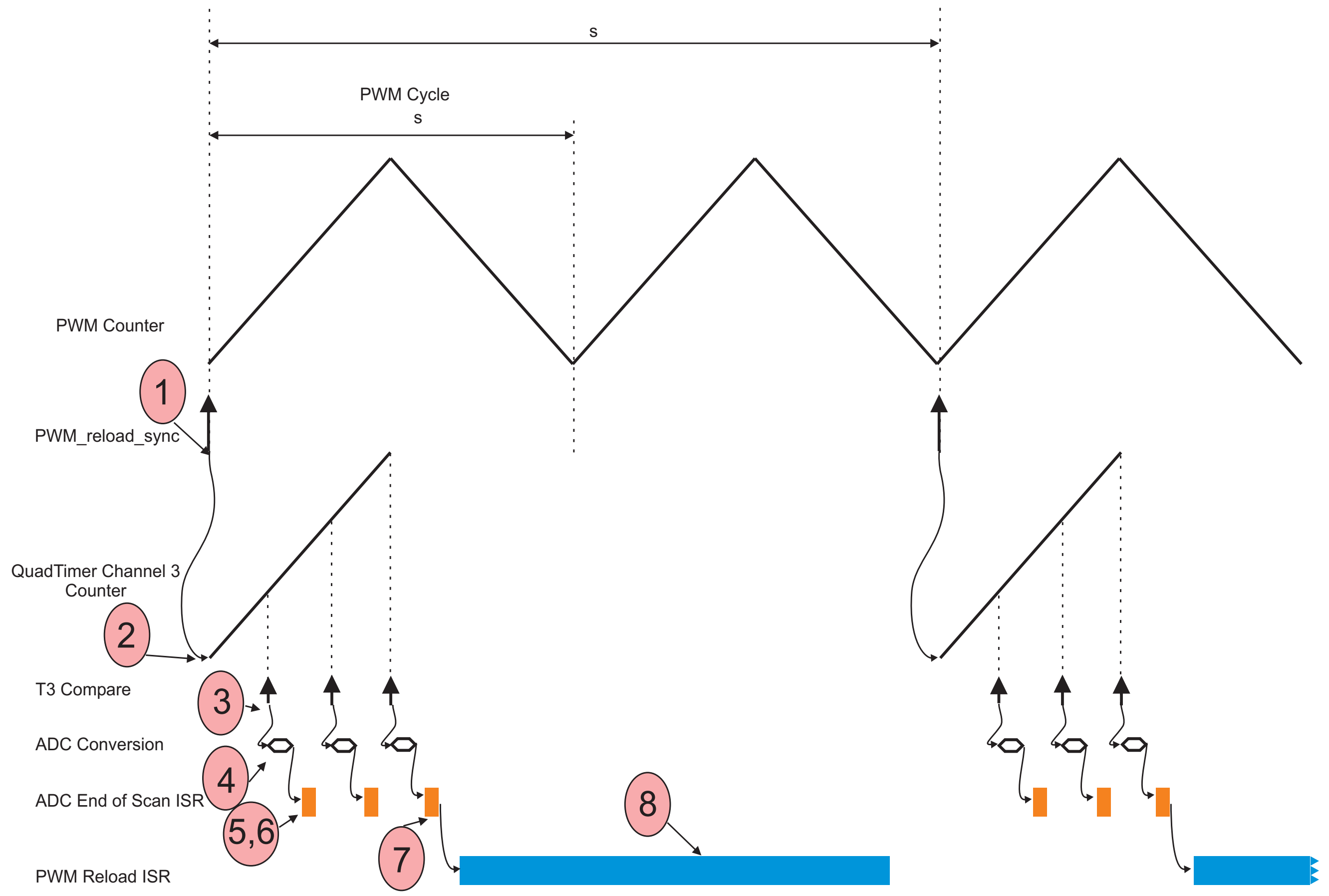


Figure 5-3 ADC End Of Scan and PWM Reload Interrupts

The triple triggered ADC sampling is used to perform a reconstruction of the three-phase motor current from the single DC-Bus 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 3-phase driver and shifted up by 1.65V. The resultant voltage is converted by the ADC; see Figure 5-5.

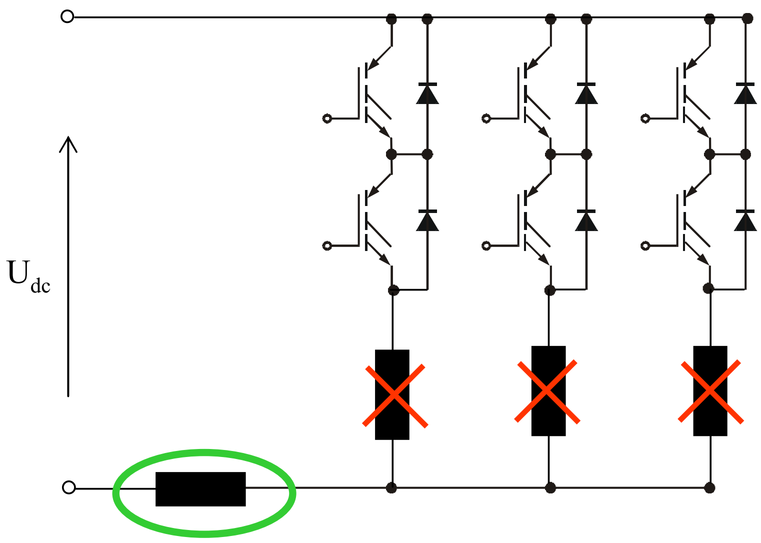


Figure 5-4 DC-Link Current Shunt

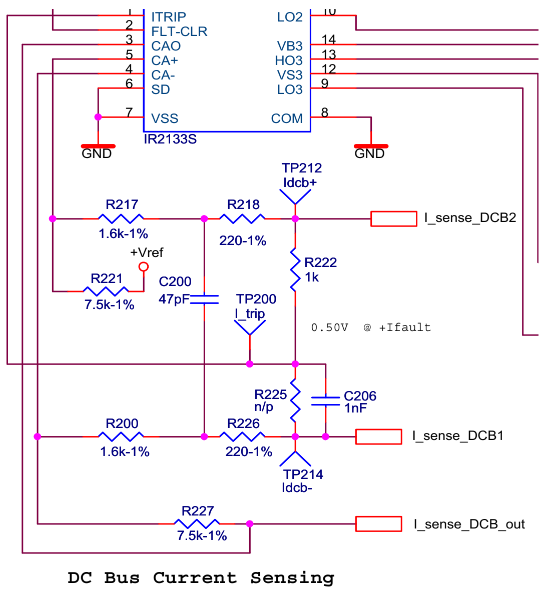


Figure 5-5 Current Amplifier for DC-Link Current

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 V_1 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 V_2 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

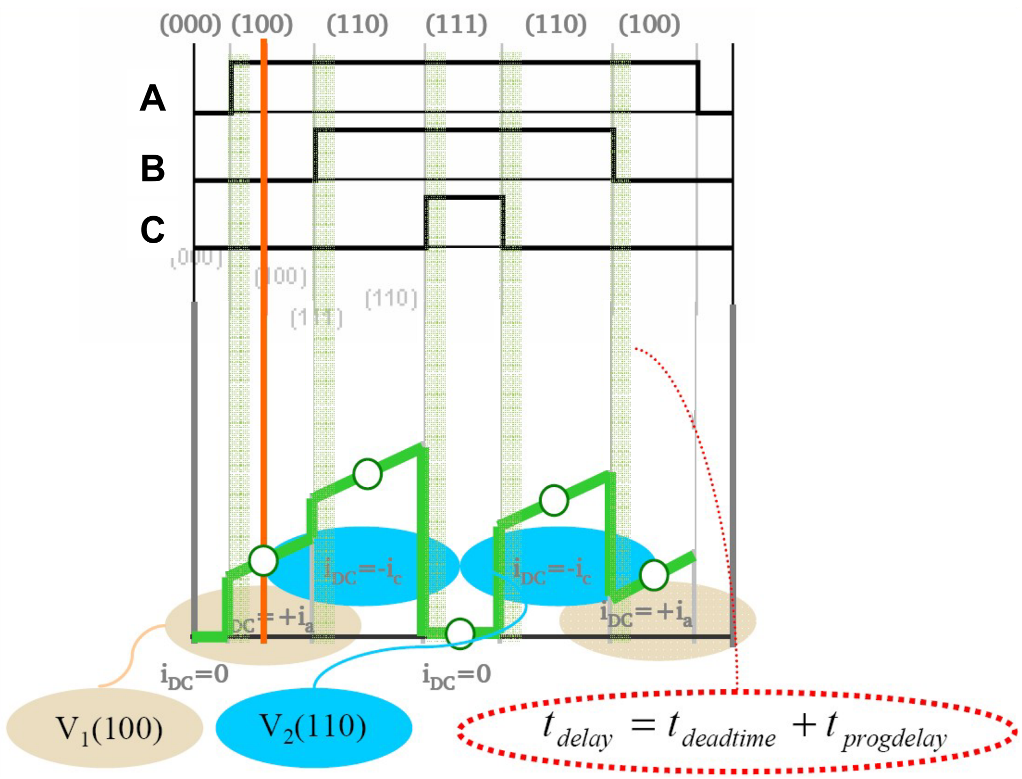


Figure 5-6 Current Sampling Timing

The ADC converter is triggered three times on every second PWM period. This 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 IGBT's 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 - sampling interval is too short and none of the current samples can be taken; see Figure 5-8.

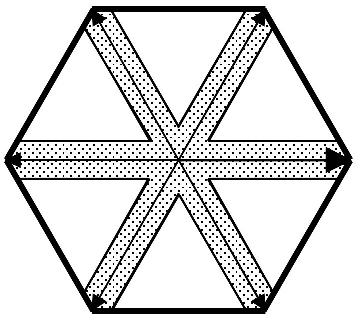


Figure 5-7 Passing Active Vector

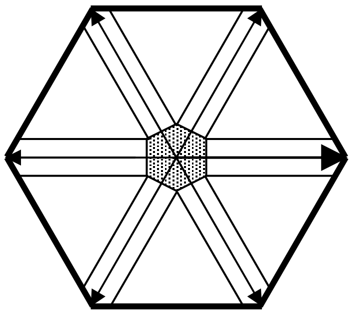


Figure 5-8 Low Modulation Index

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

The solution for asymmetrical PWM's use can be applied in both cases. In the first case, the voltage vector crosses a sector border, the centre 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 centre edge remains frozen as well, and both side edges are moved in opposite direction; see Figure 5-10.

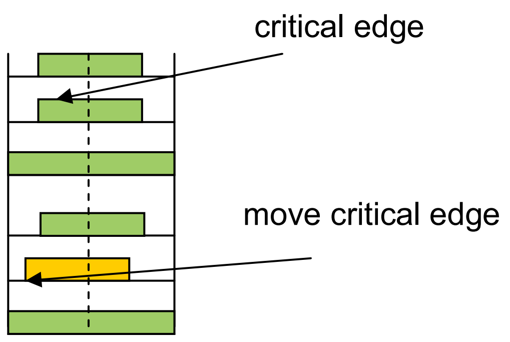


Figure 5-9 Edge Moving when Passing Active Vector

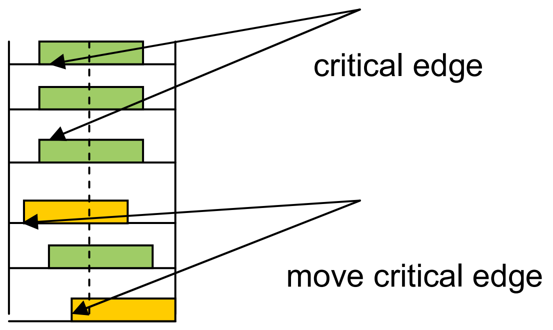


Figure 5-10 Edge Moving when Low Modulation Indexes

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 2.5µs (hardware dependent).
- The maximum voltage vector amplitude is limited by a minimum required PWM edges shift (2.5us), i.e. it cannot reach a 100% duty-cycle. If a 100% duty cycle is being applied, asymmetrical PWM's cannot be performed.
- The 3-ph current reconstruction algorithm requires both extensive calculations and a bit manipulation. The execution time on the 56F8013 is cca 20µs., The current reconstruction algorithm is performed in the PWM Reload Interrupt routine together with other calculations. This execution time is limited by a half of the PWM reload period
- The PWM Reload interrupt routine performs the fast inner the current loop together with current reconstruction algorithm. The three-phase currents are transformed into alpha, beta components of the space-vector in the stationary reference frame. Having these alpha, beta components, the actual vector size (amplitude) is evaluated and used as a feedback signal for the PI controller.The execution time on 56F8013 is 56µs while PWM period is 62.5µs (16kHz). Due to a shortage of time after processing the other algorithms, the DC-Link current measurement and PWM reload interrupt is process every second PWM period in 125µs loop; see Figure 5-3.

5.3.3 Speed Sensing

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

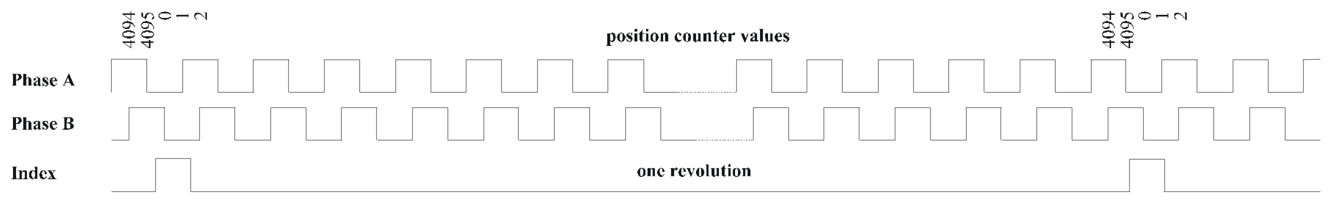


Figure 5-11 Quadrature Encoder Signals

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 a position information on the incremental sensor. At the same time, the QuadTimer channel 0 is configured to capture a counter value on both edges on secondary input pin #1. The captured counter value corresponds to the rotor position.

For speed calculation an additional QuadTimer channel is required to generate a time-base reference. QuadTimer channel 1 is configured to generate a 1ms 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 1ms period. It counts IP Bus Clock pulses on the primary source with a prescaler set to 2. The counter is configured to count roll-over. On every compare event an interrupt is generated and 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.

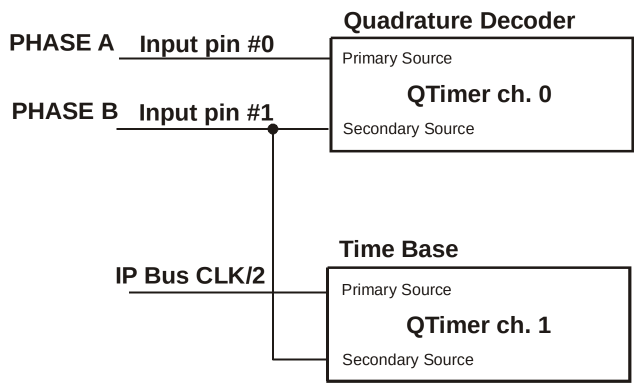


Figure 5-12 QuadTimer Channels Configuration

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

$$\text{speed} = \frac{k_1 \cdot N}{T} = \frac{k_1 \cdot N}{T_{clkT2} N_{clkT2}} = \frac{k \cdot N}{N_{clkT2}} \tag{5-13}$$

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 1ms where the speed measurement is calculated. To evaluate motor speed, the function `ENC_GetMotorSpeedEl(&enc)` is called in the 1ms 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 (5-13)
4. This process is repeated with each call of the speed processing algorithm; see Figure 5-13

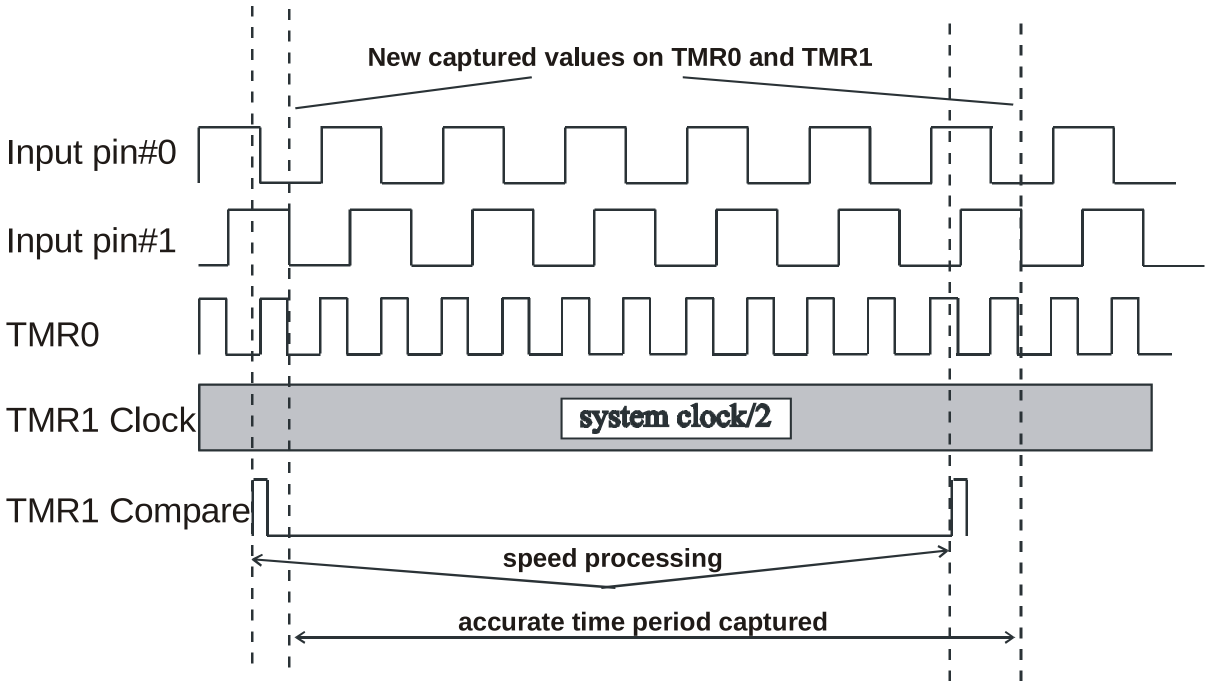


Figure 5-13 Speed Processing

5.4 Software Implementation

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

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

5.4.1 Initialization

Initialization is entered after a reset. When the application main is entered, a low-level initialization is called. According to the peripheral and CPU, the DSC registers are initialized. This is the first function 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 `AppInit()` is called. Tasks performed in `AppInit()`are as follows:

- scaling range variables are initialized for the FreeMASTER control page
- speed ramp acceleration/deceleration variables are initialized
- encoder processing data structure (`enc`) is initialized
- encoder initialization function is called with initialized `enc` structure `ENC_Init(&enc)`, and this function configures QuadTimer channels 0 and 1 for quadrature encoder processing.
- sine/cosine function data structures are initialized
- data structures of all PI controllers are initialized (proportional and integral gains, output limits)
- decoupling and induced voltage functions data structures are initialized
- shadow registers for ADC End of Scan fast interrupt are initialized
- QuadTimer ch#3 is initialized to perform ADC triggering

- ADC converter offset registers are initialized

The ADC End of Scan interrupt flag is cleared and the DSC global interrupts are enabled. The watchdog is enabled. The application enters an endless background loop.

5.4.2 Application Background Loop

The endless application background loop executes a simple application state-machine, which is shown in Figure 5-14. The state-machine has three application states: STOP, RUN and FAULT. Transition between the states is defined based on the bits in the application status and control word `appControl`.

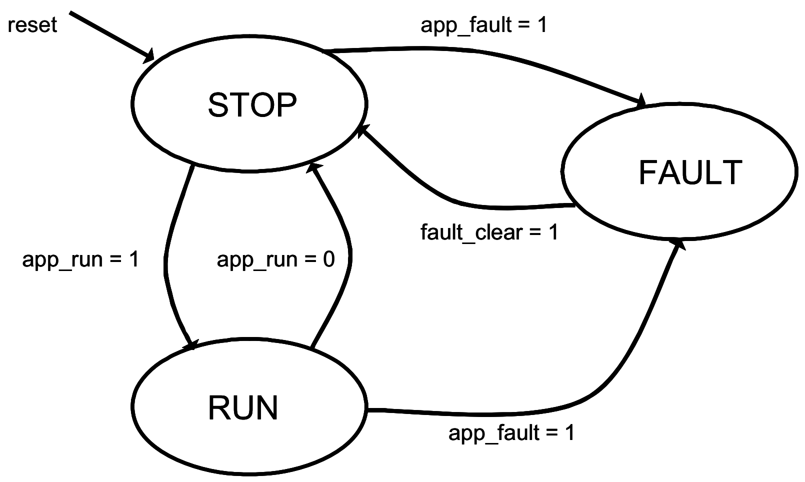


Figure 5-14 State Diagram — General Overview

The FreeMASTER polling function `FMSTR_Poll()` and the watchdog clearing function are called in the background loop as well.

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

5.4.3 Interrupts

There are three periodic interrupt service routines executing the major motor control tasks, and an overcurrent 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

5.4.3.1 ADC End Of Scan Interrupt

The function `ADA_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 twice within one 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 and PWM Reload Interrupts Timing. The first ADC trigger is set at the beginning of the PWM cycle, when all bottom transistors are off.

Samples taken at this point are used to calibrate the offset of the phase current channels. The second ADC trigger is set in the middle of the PWM cycle, when all bottom transistors are on. The currents of all three phases are sampled here.

Tasks performed by the `ADA_EndOfScanISR()`function:

- service interrupt flag
- store values from the result registers for samples 4 - 6 into the ADC result buffer
- decrement the N register counter (counts the number of subsequent triggered conversions; in the presented application the number subsequent conversions is set to two)
- service the TMR3 output OFLAG
- if the last conversion, TMR3 is stopped and the PWM reload interrupt is enabled

5.4.3.2 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 and rotor flux integration.

Tasks performed by the `PWM_ReloadISR()`function:storimg them 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 samples
- Test over-voltage fault limits
- 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
- Perform an integration of the rotor magnetizing current and an estimation of the motor slip frequency
- Evaluate the stator synchronous frequency and integrate the position of the rotor magnetizing-flux space-vector, calculate the sine and cosine components of the rotor-flux position angle
- Execute the PI controllers of the d and q-axis components of the stator current
- Evaluate the d,q-components of the output voltage vector, by the summing controller outputs with the decoupling components of the stator voltage
- Limit the output voltage vector to a unity circle for the Space-Vector Modulation algorithm
- execute digital filters on the d,q-components of the output voltage vector. These filtered values are used for correction of the rotor time-constant τ_r, which is performed in a slow control loop.
- 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 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

5.4.3.3 QuadTimer channel #1 Compare interrupt

The function `QT1_CompareISR()`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 `QT1_CompareISR()` 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 `QT1_CompareISR()`function:

- Execute the `ENC_GetMotorSpeedEl(&enc)` function to evaluate the actual rotor speed
- Controls motor acceleration/deceleration to protect the DC-Bus over-voltage fault when decelerating
- Perform ramping of the motor speed command
- Evaluate limits for the maximum torque-producing component of the stator current, to limit the maximum motor slip and the maximum stator current amplitude
- Execute the motor speed PI controller. Output of the speed controller sets the required torque-producing component of the stator current (q-axis)
- Perform the decoupling algorithm for the stator voltage
- Perform the field-weakening algorithm. Amplitude of the stator voltage is controlled by means of the PI controller. The PI controller sets required rotor-flux producing component of the stator current (d-axis).
- Execute the rotor time-constant correction algorithm
- Services corresponding the interrupt request flag

5.4.3.4 PWM Fault Interrupt

The function `PWM_FaultISR()`is assigned to this interrupt event. The interrupt is executed on an event. When a DC-Bus over-current is detected an external comparator sets 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 off the motor
- Set the application FAULT flag and OVER_CURRENT FLAG
- Services corresponding to the interrupt request flag

5.4.4 PI Controller Parameters

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

The FreeMASTER driver does NOT perform any initialization or configuration of the SCI module it uses to communicate. This 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 9600Bd. Higher communication speed is not supported by the MC56F8013/23 Controller Board due to limited speed of opto-couplers.

The FreeMASTER uses a poll-driven communication mode. It does not require the setting of interrupts for SCI. Both communication and protocol decoding is 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 a single header file, named **freemaster_cfg.h** located in project root. The user has to modify this file to configure FreeMASTER driver. The FreeMASTER driver C-source files include the **freemaster_cfg.h** file and use the macros defined there for conditional and parameter compilation.

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

5.5.2 FreeMASTER Recorder

Part of the FreeMASTER software is 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 RS232 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 **freemaster_cfg.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 QuadTimer channel 1 interrupt, which creates a 50µs time-base for the recorder function. The FreeMASTER recorder is the only routine called in this interrupt.

A detailed description of the Free Master software is provided in the **FreeMASTER Software User Manual**.

5.5.3 FreeMASTER Control Page

The FreeMASTER control page creates a Graphical User Interface (GUI) for the AC induction motor application. Start the FreeMASTER software window's project by clicking on the *acim_vc_example.pmp* file*.* Figure 5-15 illustrates the FreeMASTER software control window after this project has been launched.To switch to the control page click on the '**control page**' tag.

A user is able to monitor all the important quantities of the motor. By clicking the ON/OFF button, the motor is started. By clicking the Speed gauge, the user can set the desired speed. The actual Motor Speed, Motor Current and Voltage are displayed on the control page gauges.

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

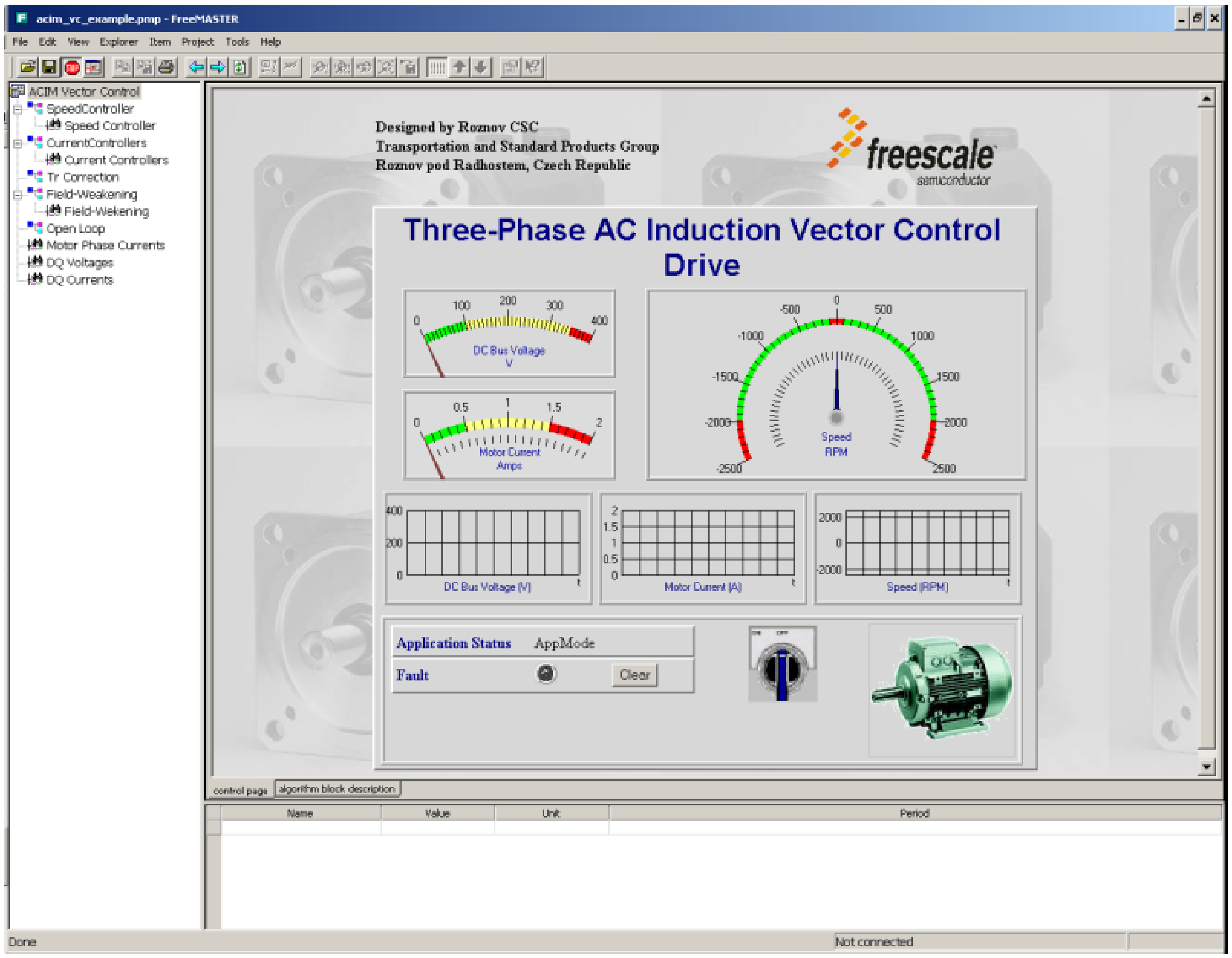


Figure 5-15 FreeMASTER Control Screen

The FreeMASTER software control actions that are supported:

- Setting the Required Speed of the motor
- Adjusting all the PI controller parameters,
- Adjusting the motor parameters
- Clearing the application Fault status

The FreeMASTER software displays the following information:

- Required Speed of the motor
- Actual Speed of the motor
- Application status - Stop / Run / Fault
- Dc-bus voltage and motor current gauges
- Scopes for DC-bus voltage & motor current and speed
- Recorders for the quantities utilized for the vector control algorithm

Chapter 6
Application Setup

As described earlier, the three-phase AC induction motor vector control application is targeted at the MC56F8013 and MC56F8023 devices. The concept of the induction motor vector control drive incorporates the following hardware components:

- MC56F8013/23 Controller Board
- 3ph AC/BLDC High Voltage Power Stage Board
- Three-Phase AC Induction Motor (default configuration for motor Elektrim SKh 71 - 4A2)

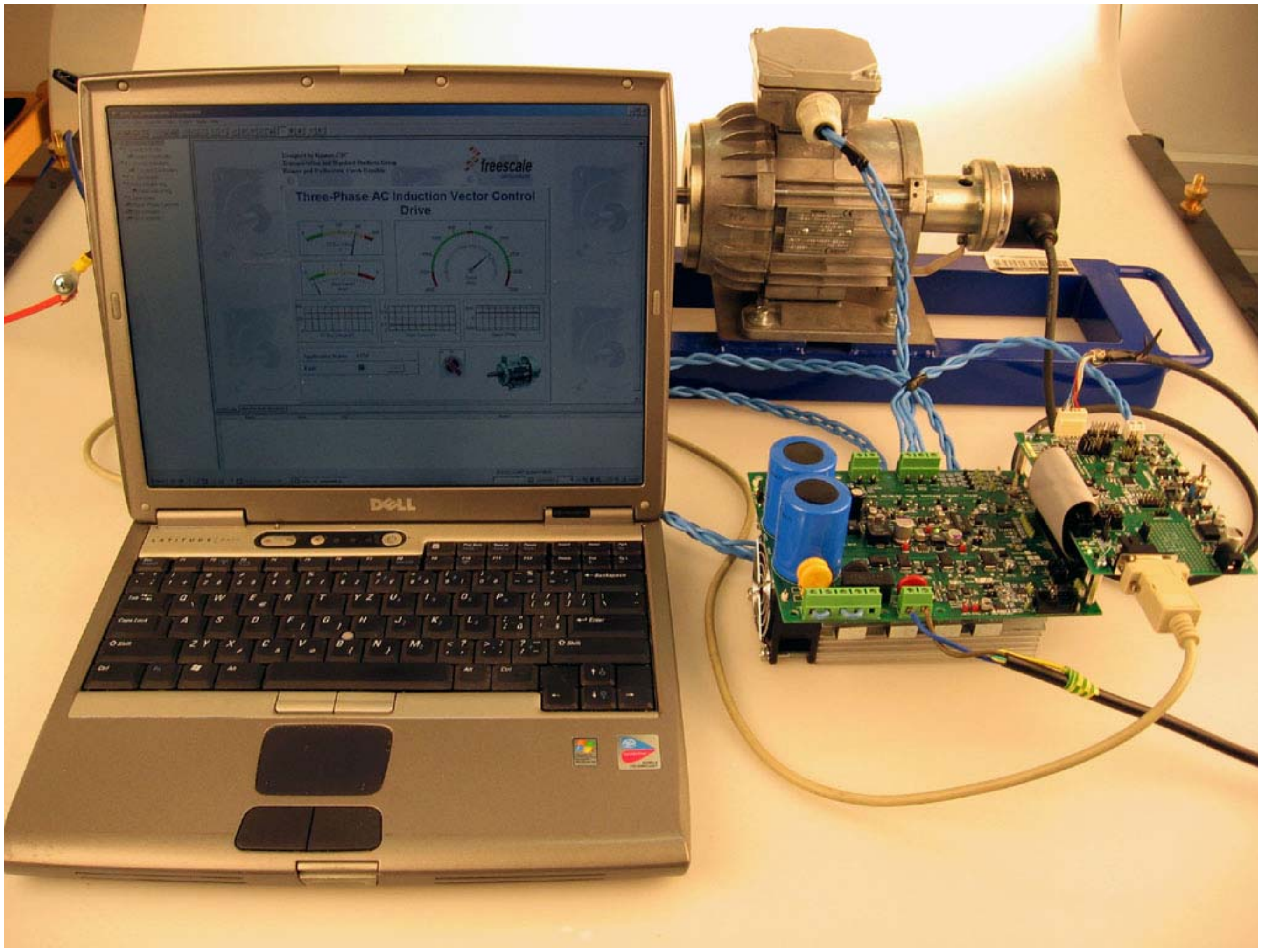


Figure 6-1 Demo Application Setup

6.1 MC56F8013 Controller Board Setup

Prior to the MC56F8013/23 Controller Board being connected to the power-stage, it needs to be configured for the correct operation. Also, the demo application code has to be programmed into the Flash memory first. For the MC56F8013/23 Controller Board configuration follow 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
JP4	1-2, 4-5, 7-8	ADC CFG2 Configuration (motor phase-current sensing)
JP5	1-2, 4-5, 7-8	ADC CFG1 Configuration (motor phase-current sensing)
JP3	4-5, 7-8	Incremental Encoder Configuration
J16	1-2	+5V Power Supply from UNI-3 connector
J18	1-2	+15V Power Supply from UNI-3 connector
Note: Other Jumpers are set to OPEN		

2. Connect a +12V power supply to the J12 power connector on the MC56F8013/23 Controller Board.
3. Set the Over-Current/Over-Voltage Thresholds.

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

   Trimpot R32 sets the over-current threshold. Use a voltmeter to measure the threshold level at test-point TP5. Turn the R32 trimpot to set the threshold level. Voltage level on TP3 should be >3.2V.
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 +12V 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.

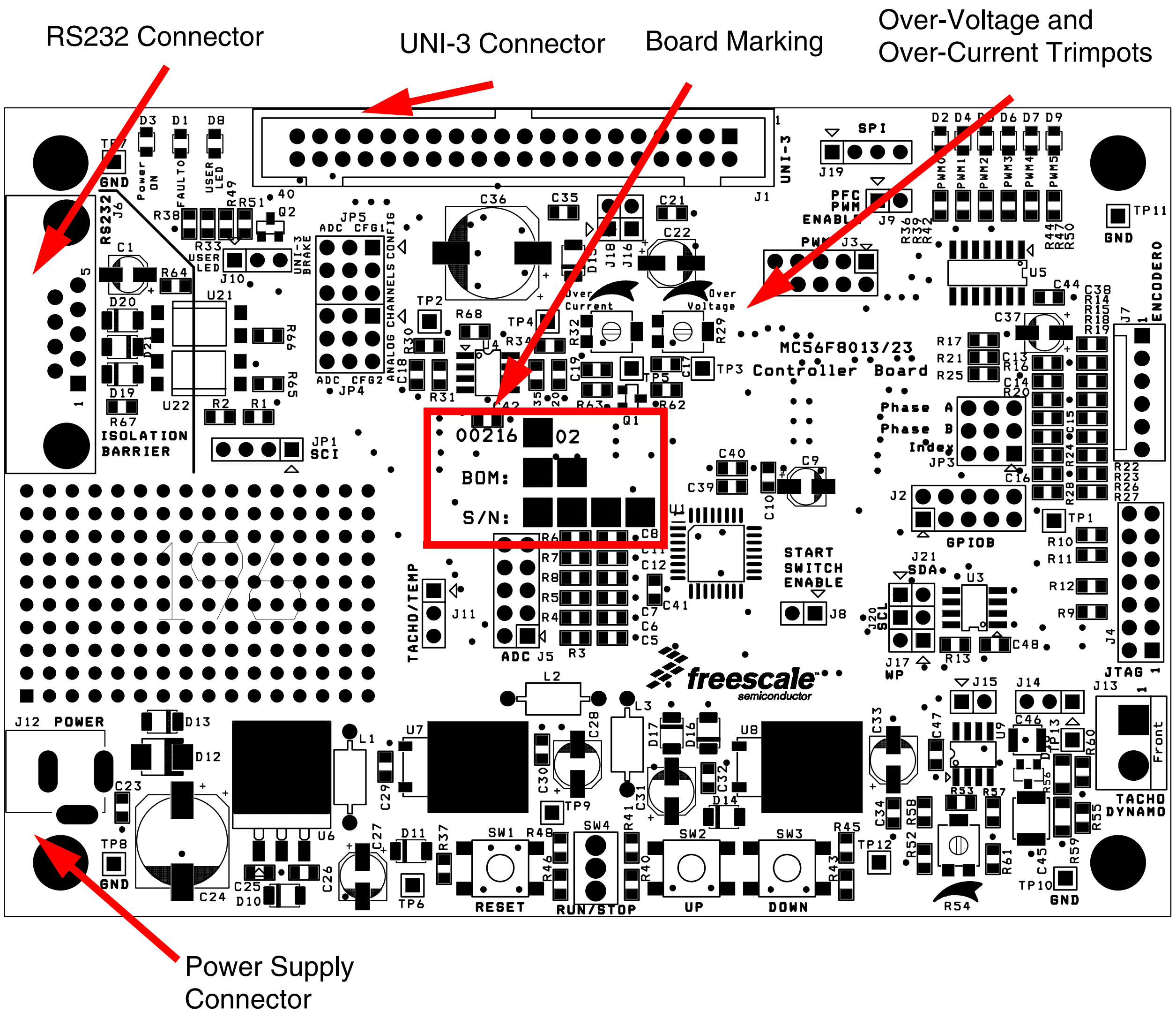


Figure 6-2 MC56F8013/23 Controller Board View

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 set-up can be built. The complete application set-up, is shown in Figure 6-1 and Figure 6-3. To build the demo application set-up follow these steps:

1. Connect the UNI-3 connector (J1) on the MC56F8013/23 Controller Board to the UNI-3 counterpart connector on the 3ph 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 terminals J6 on the 3ph AC/BLDC High Voltage Power Stage Board.
5. Connect a 115/230V AC power supply or a 100 - 320 V DC power supply to terminals J6 on the 3ph AC/BLDC High Voltage Power Stage Board.

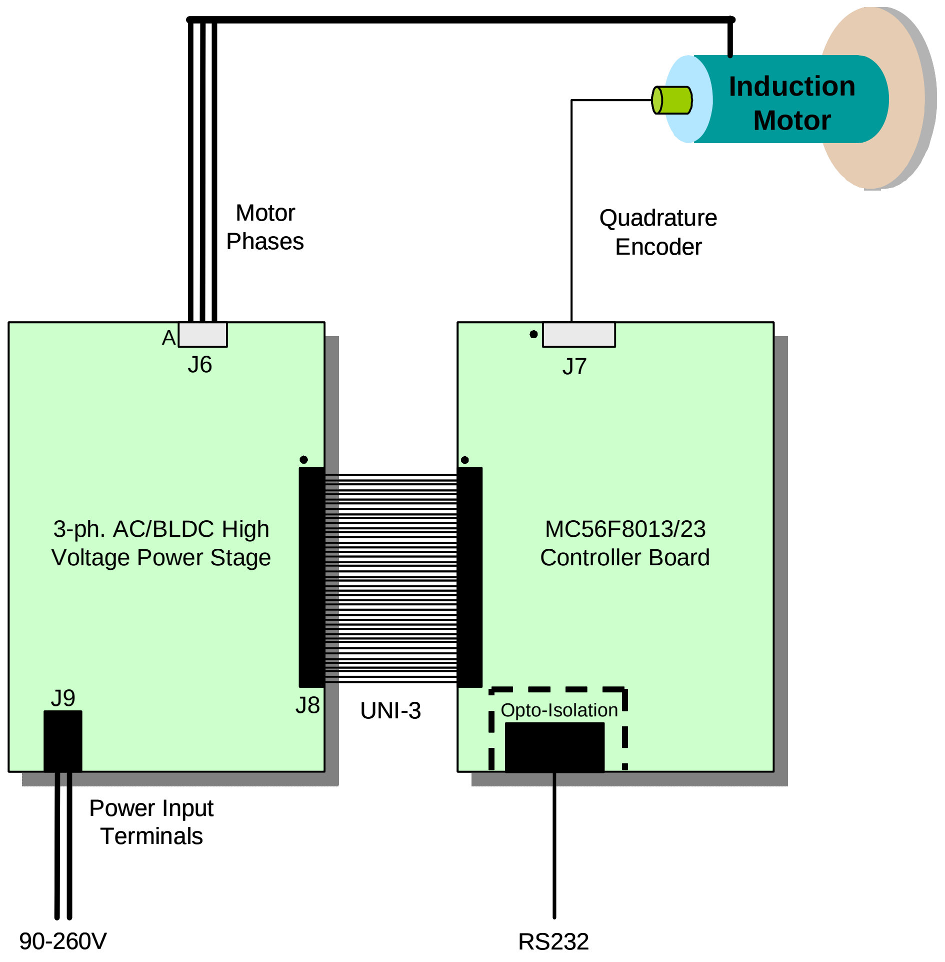


Figure 6-3 Demo Application Connection Overview

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

CAUTION

There is a risk of electric shock! Both the MC56F8013/23 Controller Board and the 3ph 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 should avoid touching them. The only safe mode of control is by remotely controlling the application using the host PC running the FreeMASTER application. The RS232 port is the only interface which provides galvanic isolation.

CAUTION

The JTAG connector does not provide galvanic isolation. The demo hardware set-up has to 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 +12V power supply connected to the J12 connector on the MC56F8013/23 Controller Board!

Appendix A References

The following documents can be found on the Freescale web site: http://www.freescale.com.

1. 56F8013 Data Sheet, MC56F8013, Freescale Semiconductor, 2006
2. *56F8023 Data Sheet,* MC56F8023, Freescale Semiconductor, 2006
3. *56F802X and 56F803X Peripheral Reference Manual*, MC56F80XXRM, Freescale Semiconductor, 2006
4. *56F801X Peripheral Reference Manual*, MC56F8000RM, Freescale Semiconductor, 2006
5. *DSP56800E Reference Manual*, DSP56800ERM, Freescale Semiconductor, 2005
6. *CodeWarrior™ Development Studio for Freescale™ 56800/E Digital Signal Controllers*, Freescale Semiconductor, 2006
7. *MC56F8013/23 Controller Board Users Manual*, Freescale Semiconductor, 2007
8. *3ph AC/BLDC High-Voltage Power Stage User Manual*, Freescale Semiconductor, 2007
9. *Free Master Software Users Manual*, Freescale Semiconductor, 2004
10. Bose, K. B., *Power Electronics and Variable Frequency Drives*, IEEE Press, ISBN 0-7803-1061-6, New York, 1997
11. Vas P.*, Sensorless Vector and Direct Torque Control*, Oxford University Press, ISBN 0-19-856465-1, New York, 1998
12. Zeman K., Peroutka Z., Janda M., *Automaticka regulace pohonu s asynchronnimi motory*, University of West Bohemia, Plzen, 2004

Appendix B
Glossary of Symbols

α,β	— Stator orthogonal coordinate system
d,q	— Rotational orthogonal coordinate system
$u_{s\alpha,\beta}$	— Stator voltages in α,β coordinate system
$u_{sd,q}$	— Stator voltages in d,q coordinate system
$i_{s\alpha,\beta}$	— Stator currents in α,β coordinate system
$i_{sd,q}$	— Stator currents in d,q coordinate system
$u_{r\alpha,\beta}$	— Rotor voltages in α,β coordinate system
$u_{rd,q}$	— Rotor voltages in d,q coordinate system
$i_{r\alpha,\beta}$	— Rotor currents in α,β coordinate system
$i_{rd,q}$	— Rotor currents in d,q coordinate system
$\mathrm{i_{mr}}$	— Rotor magnetizing current
$\Psi_{s\alpha,\beta}$	— Stator magnetic fluxes in α,β coordinate system
$\Psi_{sd,q}$	— Stator magnetic fluxes in d,q coordinate system
$\Psi_{r\alpha,\beta}$	— Rotor magnetic fluxes in α,β coordinate system
$\Psi_{rd,q}$	— Rotor magnetic fluxes in d,q coordinate system
θ_{ψ}	— Rotor magnetizing flux
R_s	— Stator phase resistance
R_r	— Rotor phase resistance
L_s	— Stator phase inductance
$L_{s\sigma}$	— Stator phase leakage inductance
$L_{r\sigma}$	— Rotor phase leakage inductance
L_r	— Rotor phase inductance
L_m	— Mutual (stator to rotor) inductance
ω / ω_s	— Electrical rotor angular speed / synchronous angular speed
$\mathrm{f_s}$	— Electrical stator synchronous frequency
$\mathrm{f_{slip}}$	— Electrical rotor slip frequency
p_p	— Number of poles per phase
$\mathrm{t_e}$	— Electromagnetic torque
τ_r	— Rotor time constant

Appendix C
Glossary of Abbreviations

AC — alternating current

ADC — analog-to-digital converter

brush — a component transferring electrical power, from non-rotational terminals mounted on the stator, to the rotor

BLDC — brushless direct current motor

commutator — A mechanical device alternating DC current in a DC commutator motor and providing rotation of DC commutator motor

COP — computer operating properly (watchdog timer

DC — direct current

DC/DC Inverter— power electronics module that converts DC voltage level to a different DC voltage level

DSC — digital signal controller

MC56F80x — a Freescale family of 16-bit DSPs dedicated to motor control

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

duty cycle — the ratio of the amount of time the signal is on to the time it is off. Duty cycle is usually quoted as a percentage

GPIO — general purpose input/output

Hall sensor — a position sensor giving six defined events (each 60 electrical degrees) per electrical revolution (for a 3-phase motor)

interrupt — a temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine

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

PI controller — proportional-integral controller

PLL — phase-locked loop: a clock generator circuit in which a voltage controlled oscillator produces an oscillation that is synchronized to a reference signal

PWM — pulse width modulation

Quad timer — a module with four 16-bit timers

reset — to force a device to a known condition

RPM — revolutions per minute

SCI — serial communication interface module: a module that supports asynchronous communication

software — instructions and data that control the operation of a microcontroller

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 2666 8080
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

DRM092
Rev. 0, 7/2007