

Sensorless PMSM Vector Control

Design Reference Manual

MCF51AC256
ColdFire

DRM109
Rev.0
04/2009

freescale.com



Chapter 1

Introduction

1.1	Introduction	5
1.1.1	Application Features and Components	5
1.1.2	Sinusoidal PM Synchronous Motors Applications Overview	5
1.2	Freescale Controller Advantages and Features	7
1.3	Bibliography	8
1.3.1	Acronyms and Abbreviations	9
1.3.2	Glossary of Terms	10
1.3.3	Glossary of Symbols	10

Chapter 2

Control Theory

2.1	3-Phase PM Synchronous Motor	13
2.2	Mathematical Description of PM Synchronous Motors	13
2.2.1	Space Vector Definitions	14
2.2.2	PM Synchronous Motor Model	15
2.3	Vector Control of PM Synchronous Motor	17
2.3.1	Fundamental Principle of Vector Control	17
2.3.2	Description of the Vector-Control Algorithm	18
2.3.3	Stator Voltage Decoupling	19
2.3.4	Space Vector Modulation	20
2.3.5	Motor Position Alignment	23

Chapter 3

System Concept

3.1	System Specification	25
3.2	Application Description	25
3.3	Control Process	26

Chapter 4

Hardware

4.1	Hardware Implementation	29
4.2	3-Phase Motor Control Drive Using MCF51AC256	31
4.3	Motor Specifications — Example	33

Chapter 5

Software Design

5.1	Introduction	35
5.2	Application Variables Scaling	35
5.2.1	Fractional Numbers Representation	35
5.2.2	Scaling of Analog Quantities	35
5.2.3	Scaling of Angles	36
5.2.4	Scaling of Parameters	36
5.3	Application Overview	38
5.3.1	ADC Conversion Timing and PWM Reload Interrupts	39
5.3.2	3-Phase Current Sampling	40
5.3.3	Position and Speed by Observer	40
5.4	Software Implementation	43
5.4.1	Software States	43
5.4.2	Initialization	46
5.4.3	Application Background Loop	47
5.4.4	Interrupts	48
5.4.5	FTM1 Overflow Interrupt	49
5.4.6	PI Controller Parameters	49
5.5	FreeMASTER Software	50
5.5.1	FreeMASTER Serial Communication Driver	50
5.5.2	FreeMASTER Recorder	51
5.5.3	FreeMASTER Control Page	51

Chapter 6

Application Setup

6.1	3-Phase Motor Control Drive Using MCF51AC256 Setup	53
6.2	Demo Hardware Setup	54

Chapter 7

Results and Measurements

7.1	System and Measurement Conditions	57
7.1.1	Hardware Setup	57
7.1.2	Software Setup	57
7.1.3	FreeMASTER	57
7.2	Measured Results	57
7.2.1	3-Phase Current Reconstruction	57
7.2.2	Speed Controller	58
7.2.3	Current Controller	59

Chapter 1

Introduction

1.1 Introduction

This document describes the design of a 3-phase PMSM vector control drive with 3-shunt current sensing without position sensor. The design is targeted for consumer and industrial applications. This cost-effective solution benefits from Freescale Semiconductor MCF51AC256 device dedicated for motor control.

1.1.1 Application Features and Components

The system is designed to drive a 3-phase PM synchronous motor. Application features:

- Sensorless 3-phase PMSM speed vector (FOC) control.
- Current sensing with three current sensors.
- Based on Freescale MCF51AC256 controller.
- Running on a 3-Phase Motor Control Drive (12 V) board with an MCF51AC256 daughter board.
- FreeMASTER software control interface and monitor.

Main application components available for customers are:

- Software — written in C-code using some library algorithms — available for the MCF51AC256.
- Hardware — based on Freescale universal motor control hardware modules.
- Documentation — this document.

1.1.2 Sinusoidal PM Synchronous Motors Applications Overview

Sinusoidal PM synchronous motors are more and more popular for new drives, replacing brushed DC, universal, and other motors in a wide application area. The reason is a better reliability (no brushes), better efficiency, lower acoustic noise, and also other benefits of electronic control. A disadvantage of PM synchronous motor drives might be the need for a more sophisticated electronic circuit. But nowadays, most applications need an electronic speed or torque regulation and other features with the necessity of electronic control. Once we use a system with electronic control, it is already only a small system cost increase to implement more advanced drives like the sinusoidal PM synchronous motor with digitally-controlled switching inverter and DC-bus circuit. It is only necessary to have a cost-effective controller with a good calculation performance. One of them is the Freescale MCF51AC256.

The PM synchronous motor also has advantages when compared to an AC induction motor. Because a PM synchronous motor achieves higher efficiency by generating the rotor magnetic flux with rotor magnets,

it is used in white goods (such as refrigerators, washing machines, dishwashers), pumps, fans, and in other appliances that require a high reliability and efficiency.

The 3-phase synchronous motors with permanent magnets come in two most popular variants. The sinusoidal PM synchronous motor and the trapezoidal BLDC motor. The sinusoidal PM synchronous motor is very similar to the trapezoidal BLDC (Electronically-Commuted) motor. There are two main differences:

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

Generally, we can say that sinusoidal PM synchronous motor performance is better due to constant torque, while the trapezoidal BLDC motor can be more easily controlled. The trapezoidal BLDC motors are used mainly for historical reasons. It was easier to create a six-step commutation and estimate the rotor position with simpler algorithms, since one phase is non conducting. The sinusoidal PM synchronous motors require more sophisticated control, but give us some benefits, such as smoother torque, lower acoustic noise, and so on. As shown in this document, the MCF51AC256 provides all the necessary functionality for 3-phase sinusoidal PM synchronous motor vector control. Using the MCF51AC256 we have a strong argument to replace the trapezoidal BLDC (Electronically-Commuted) motors with the 3-phase sinusoidal PM synchronous motors with almost no system cost increase.

The application described here is a speed vector control. The speed-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 bring overall improvements to drive performance over scalar control. Let's mention the higher efficiency, full torque control, decoupled control of flux and torque, improved dynamics, and so on.

For the PM synchronous motor control, it is necessary to know the exact rotor position. This application uses special algorithms to estimate the speed and position instead of using a physical position sensor.

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

1.2 Freescale Controller Advantages and Features

The Freescale MCF51AC256 is well-suited for digital motor control, combining the calculation capability with the MCU's controller features in a single chip. These controllers offer many dedicated peripherals such as pulse width modulation (FTM) modules, analog-to-digital converters (ADC), timers, communication peripherals (SCI, SPI, I²C), and on-board Flash and RAM.

The MCF51AC256 provides the following peripheral blocks:

- Two FTM modules with PWM outputs, fault input, fault-tolerant design with dead-time insertion, supporting both center-aligned and edge-aligned modes.
- 12-bit ADC; ADC and PWM modules can be synchronized.
- One two-input 16-bit general-purpose timer module.
- Two serial peripheral interfaces (SPI).
- Two serial communications interfaces (SCI) with LIN slave functionality.
- One inter-integrated circuit (I²C) port.
- On-board 5.0 V to 2.7 V voltage regulator for powering internal logic and memories.
- Integrated power-on reset and low-voltage interrupt module.
- All pins multiplexed with general-purpose input/output (GPIO) pins.
- Computer Operating Properly (COP) watchdog timer.
- Two Analog Comparators.
- One Real-Time Counter (RTC).
- One Programmable Interval Timer (PIT).
- External $\overline{\text{RESET}}$ input pin for hardware reset.
- msCAN Interface for Industrial Control.
- Low Voltage Detect (LVD), Low Voltage Warning (LVW).
- Multi-Clock Generator (MCG).
- IEC60730 class C safety features.
- On-chip ICE and BDM.
- Phase-locked loop (PLL) based frequency synthesizer for the controller core clock, with on-chip relaxation oscillator.

Table 1-1. Memory Configuration

Memory Type	MCF51AC256
Program Flash	256 KByte
Unified Data/Program RAM	32 KByte

The PMSM vector control benefits greatly from the FlexTimer (FTM) module and ADC.

The FTM offers flexibility in its configuration, enabling efficient 3-phase motor control. The FTM module is capable of generating asymmetric PWM duty cycles in a center-aligned configuration.

The FTM block has the following features:

- Three complementary PWM signal pairs, six independent PWM signals (or a combination).
- Complementary channel operation features.
- Dead time insertion.
- Separate top and bottom polarity control.
- Clock source up to two times the bus clock.
- Edge-aligned or center-aligned PWM reference signals.
- 16-bit resolution.
- Mask/swap capability.
- Programmable fault protection.
- Polarity control.
- Write-protectable registers.
- Trigger event generation for the ADC module (only in complementary mode when the channel three or five is free).

The ADC module has the following features:

- 12-bit resolution.
- 28 inputs.
- Hardware trigger.
- Input clock selectable from up to four sources.
- 2.5 μ s per sample conversion.
- Interrupt-generating capabilities when the sample is converted.

The application uses the ADC block synchronized to the PWM pulses. This configuration allows a simultaneous conversion of the required analog values for the inverter currents and DC-bus voltage within the required time.

1.3 Bibliography

1. *MCF51AC256 ColdFire® Integrated Microcontroller Reference Manual*, Freescale Semiconductor, 2007
2. *ColdFire® Family Programmer's Reference Manual*, Freescale Semiconductor, 2005
3. *CodeWarrior™ Development Studio 8/16-Bit IDE User's Guide*, Freescale Semiconductor, 2007
4. *3-Phase BLDC/PMSM Low-Voltage Motor Control Drive User Manual*, Freescale Semiconductor, 2009
5. *MCF51AC256 Daughter Board User Manual*, Freescale Semiconductor, 2009
6. *FreeMASTER Software Users Manual*, Freescale Semiconductor, 2004
7. *3-Phase PM Synchronous Motor Vector Control using DSP56F80x*, by Prokop L., Grasblum P., AN1931 Motorola, 2002

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

1.3.1 Acronyms and Abbreviations

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

Table 1-2. Acronyms and Abbreviated Terms

Term	Meaning
AC	Alternating current
ADC	Analog-to-digital converter
FTM	FlexTimer module
TPM	Timer module
BEMF	Back electromagnetic force = induced voltage
BLDC	Brushless direct current motor
COP	Computer operating properly (watchdog timer)
DC	Direct current
DSC	Digital signal controller
DT	Dead time: a short time that must be inserted between the turning off of one transistor in the inverter half bridge and turning on of the complementary transistor due to the limited switching speed of the transistors
FOC	Field-oriented control
GPIO	General-purpose input/output
I/O	Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal, and writes to an output to change the level of an external signal.
BDM	Background debug module
LED	Light-emitting diode
MCF51AC256	A Freescale ColdFireV1 32-bit controller
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.
PMSM	PM Synchronous Motor, permanent magnet synchronous motor
PWM	Pulse width modulation
RPM	Revolutions per minute
SCI	Serial communication interface module: a module that supports asynchronous communication.

1.3.2 Glossary of Terms

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

Table 1-3. Glossary

Term	Definition
brush	A component transferring electrical power from non-rotational terminals, mounted on the stator, to the rotor.
commutator	A mechanical device alternating DC current in a DC commutator motor and providing rotation of DC commutator motor.
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.
Quadrature Encoder	A position sensor giving two code tracks with sector positioned 90 degrees out of phase. The two output channels indicate both position and direction of motor rotation.
interrupt	A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.
PM Synchronous Motor	Permanent magnet synchronous motor.
PI controller	Proportional-integral controller.
reset	To force a device to a known condition.
software	Instructions and data that control the operation of a microcontroller.

1.3.3 Glossary of Symbols

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

Table 1-4. Glossary of Symbols

Term	Definition
d,q	Rotational orthogonal coordinate system.
g_{ω}	Adaptive speed scheme gain.
e_{α}, e_{β}	Alpha/Beta BEMF observer error.
i_{sa}, i_{sb}, i_{sc}	Stator currents of the a, b, and c phases.
$i_{sd,q}, i_{s(d,q)}$	Stator currents in the d, q coordinate system.
$i_{s(d,q)*}$	Stator currents in estimated d, q coordinate system.
$i_{s\alpha,\beta}, i_{s(\alpha,\beta)}$	Stator currents in α, β coordinate system.
\hat{i}_{sg}	Stator current space vector in general reference frame.
i_{sx}, i_{sy}	Stator current space vector components in general reference frame.
\hat{i}_{rg}	Rotor current space vector in general reference frame.
i_{rx}, i_{ry}	Rotor current space vector components in general reference frame.
J	Mechanical inertia.

Table 1-4. Glossary of Symbols (Continued)

Term	Definition
K_1, K_2	Angle-tracking observer coefficients.
K_M	Motor constant.
L_s	Stator-phase inductance.
L_{sd}	Stator-phase inductance d axis.
L_{sq}	Stator-phase inductance q axis.
p	Number of poles per phase.
R_s	Stator-phase resistance.
T_L	Load torque.
$u_{S\alpha,\beta}, u_{S(\alpha,\beta)}$	Stator voltages in α, β coordinate system.
$u_{Sd,q}, u_{S(d,q)}$	Stator voltages in d, q coordinate system.
a, b	Stator orthogonal coordinate system.
$\Psi_{S\alpha,\beta}$	Stator magnetic fluxes in α, β coordinate system.
$\Psi_{Sd,q}$	Stator magnetic fluxes in d, q coordinate system.
Ψ_M	Rotor magnetic flux.
θ_r	Rotor position angle in α, β coordinate system.
$\omega, \omega_{el}/\omega_F$	Electrical rotor angular speed / fields angular speed.



Chapter 2

Control Theory

2.1 3-Phase PM Synchronous Motor

The PM synchronous motor is a rotating electric machine with a classic 3-phase stator like that of an induction motor; the rotor has surface-mounted permanent magnets (see [Figure 2-1](#)).

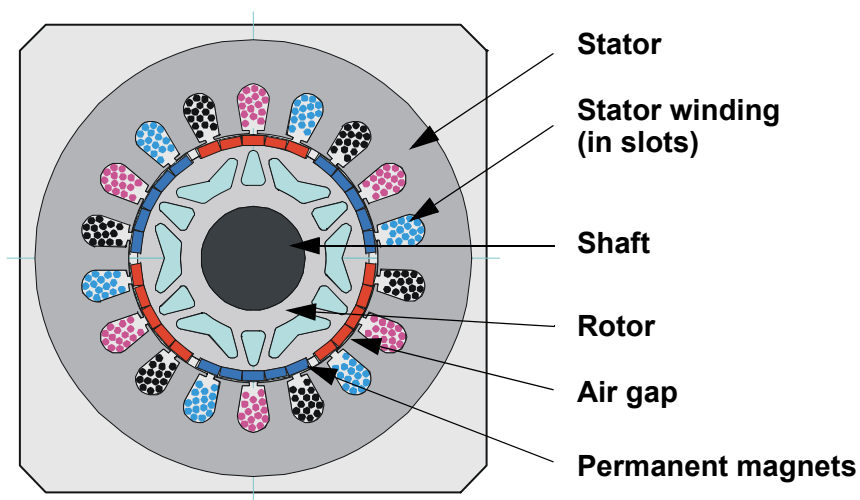


Figure 2-1. PM Synchronous Motor — Cross Section

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

2.2 Mathematical Description of PM Synchronous Motors

There are a number of PM synchronous motor models. The model used for vector-control design can be obtained by utilizing space-vector theory. The 3-phase motor quantities (such as voltages, currents, magnetic flux, and so on) 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 Definitions

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

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

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

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

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

The space vector defined by Equation 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:

$$\bar{i}_s = i_{s\alpha} + ji_{s\beta} \quad \text{Eqn. 2-3}$$

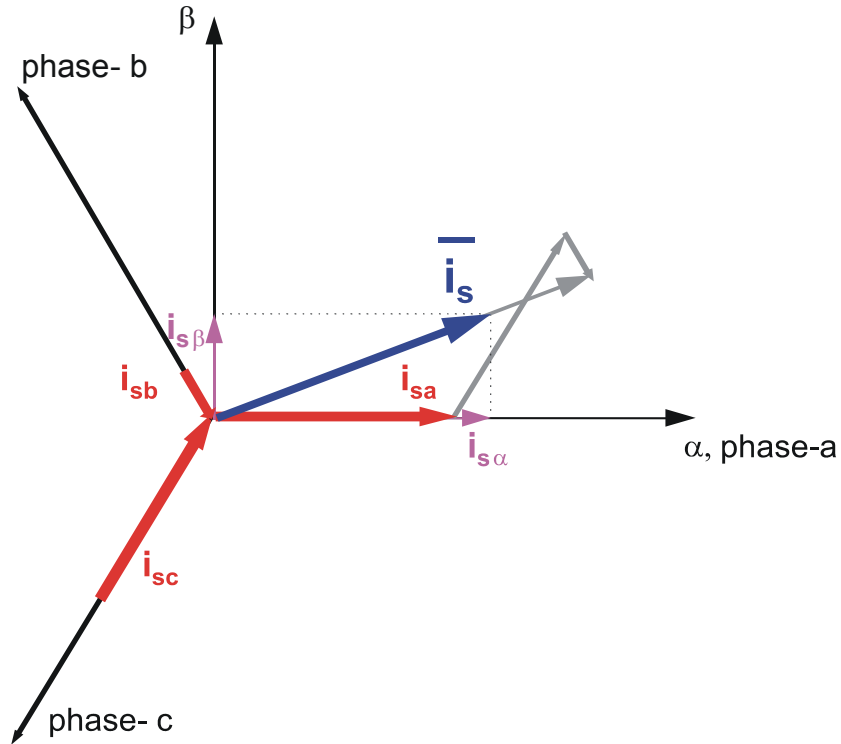


Figure 2-2. Stator-Current Space Vector and Its Projection

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

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

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

where $k=2/3$ is a transformation constant so that the final equation is:

$$i_{s\beta} = \frac{1}{\sqrt{3}} (i_{sb} - i_{sc}) \quad \text{Eqn. 2-6}$$

The space vectors of other motor quantities (voltages, currents, magnetic fluxes, and so on) can be defined in the same way as the stator-current space vector.

2.2.2 PM Synchronous Motor Model

For a description of the PM synchronous motor, the symmetrical 3-phase smooth-air-gap machine with sinusoidally-distributed windings is considered. The voltage equations of stator in the instantaneous form can then be expressed as:

$$u_{SA} = R_S i_{SA} + \frac{d}{dt} \psi_{SA} \quad \text{Eqn. 2-7}$$

$$u_{SB} = R_S i_{SB} + \frac{d}{dt} \psi_{SB} \quad \text{Eqn. 2-8}$$

$$u_{SC} = R_S i_{SC} + \frac{d}{dt} \psi_{SC} \quad \text{Eqn. 2-9}$$

where u_{SA} , u_{SB} , and u_{SC} are the instantaneous values of stator voltages, i_{SA} , i_{SB} , and i_{SC} are the instantaneous values of stator currents, and ψ_{SA} , ψ_{SB} , ψ_{SC} are instantaneous values of stator flux linkages, in phase SA, SB, and SC.

Due to the large number of equations in the instantaneous form, the equations [Equation 2-7](#), [Equation 2-8](#), and [Equation 2-9](#), it is more practical to rewrite the instantaneous equations using two-axis theory (Clarke transformation). The PM synchronous motor can be expressed as:

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

$$u_{S\beta} = R_S i_{S\beta} + \frac{d}{dt} \Psi_{S\beta} \quad \text{Eqn. 2-11}$$

$$\Psi_{S\alpha} = L_S i_{S\alpha} + \Psi_M \cos(\theta_r) \quad \text{Eqn. 2-12}$$

$$\Psi_{S\beta} = L_S i_{S\beta} + \Psi_M \sin(\theta_r) \quad \text{Eqn. 2-13}$$

$$\frac{d\omega}{dt} = \frac{p}{J} \left[\frac{3}{2} p (\Psi_{S\alpha} i_{S\beta} - \Psi_{S\beta} i_{S\alpha}) - T_L \right] \quad \text{Eqn. 2-14}$$

For glossary of symbols see [Section 1.3.3, “Glossary of Symbols,” Table 1-4](#).

The equations [Equation 2-10](#) through [Equation 2-14](#) represent the model of a PM synchronous motor in the stationary frames α , β , fixed to the stator.

Besides the stationary reference frame attached to the stator, motor model voltage space vector equations can be formulated in a general reference frame, which rotates at a general speed ω_g . If a general reference frame is used, with direct and quadrature axes x , y rotating at a general instantaneous speed $\omega_g = d\theta_g/dt$, as shown in Figure 2-3, where θ_g is the angle between the direct axis of the stationary reference frame (α) attached to the stator and the real axis (x) of the general reference frame, then Equation 2-15 defines the stator current space vector in general reference frame:

$$\bar{i}_{sg} = \bar{i}_s e^{-j\theta_g} = i_{sx} + j i_{sy} \quad \text{Eqn. 2-15}$$

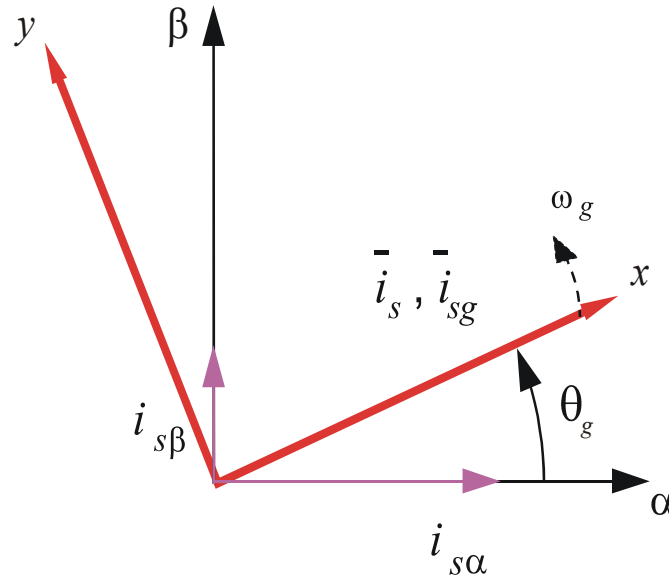


Figure 2-3. Application of the General Reference Frame

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

Similar considerations are held for the space vectors of the rotor voltages, currents, and flux linkages. The real axis (ra) of the reference frame attached to the rotor is displaced from the direct axis of the stator reference frame by the rotor angle q_r . Since 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 (ra) is $q_g - q_r$, in the general reference frame, the space vector of the rotor currents can be expressed as:

$$\bar{i}_{rg} = \bar{i}_r e^{-j(\theta_g - \theta_r)} = i_{rx} + j i_{ry} \quad \text{Eqn. 2-16}$$

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

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

The motor model voltage equations in the general reference frame can be expressed by utilizing introduced transformations of the motor quantities from one reference frame to the general reference frame. The PM synchronous motor model is often used in vector-control algorithms.

The aim of vector control is to implement control schemes that produce high dynamic performance and are similar to those used to control DC machines. To achieve this, the reference frames may be aligned with the stator flux-linkage space vector, the rotor flux-linkage space vector or the magnetizing space vector. The most popular reference frame is the reference frame attached to the rotor flux-linkage space vector, with direct axis (d) and quadrature axis (q).

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

$$u_{sd} = R_s i_{sd} + \frac{d}{dt} \Psi_{sd} - \omega_F \Psi_{sq} \quad \text{Eqn. 2-17}$$

$$u_{sq} = R_s i_{sq} + \frac{d}{dt} \Psi_{sq} + \omega_F \Psi_{sd} \quad \text{Eqn. 2-18}$$

$$\Psi_{sd} = L_s i_{sd} + \Psi_M \quad \text{Eqn. 2-19}$$

$$\Psi_{sq} = L_s i_{sq} \quad \text{Eqn. 2-20}$$

$$\frac{d\omega}{dt} = \frac{p}{J} \left[\frac{3}{2} p (\Psi_{sd} i_{sq} - \Psi_{sq} i_{sd}) - T_L \right] \quad \text{Eqn. 2-21}$$

By considering that below base speed $i_{sd}=0$, the Equation 2-21 can be reduced to the following form:

$$\frac{d\omega}{dt} = \frac{p}{J} \left[\frac{3}{2} p (\Psi_M i_{sq}) - T_L \right] \quad \text{Eqn. 2-22}$$

From the Equation 2-22, it can be seen that the torque is dependent and can be directly controlled by the current i_{sq} only.

2.3 Vector Control of PM Synchronous Motor

2.3.1 Fundamental Principle of Vector Control

High-performance motor control is characterized by smooth rotation over the entire speed range of the motor, full torque control at zero speed, fast accelerations and decelerations. To achieve such control, vector control techniques are used for 3-phase AC motors. The vector control techniques are usually also referred to as the field-oriented control (FOC). The basic idea of the FOC algorithm is to decompose a stator current into a magnetic field-generating part and a torque-generating part. Both components can be controlled separately after decomposition. The structure of the motor controller is then as simple as that for a separately excited DC motor.

Figure 2-4 shows the basic structure of the vector-control algorithm for the PM Synchronous motor.

To perform vector control, it is necessary to follow these steps:

- Measure the motor quantities (phase voltages and currents).
- Transform them into the 2-phase system (α, β) using Clarke transformation.
- Calculate the rotor flux space-vector magnitude and position angle.
- Transform stator currents into the d, q reference frame using 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 rotor position and velocity information to be sensed. Incremental encoders or resolvers attached to the rotor are naturally used as position transducers for vector-control drives. In some applications the use of speed/position sensors 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 that do not employ speed sensors are called “sensorless control”.

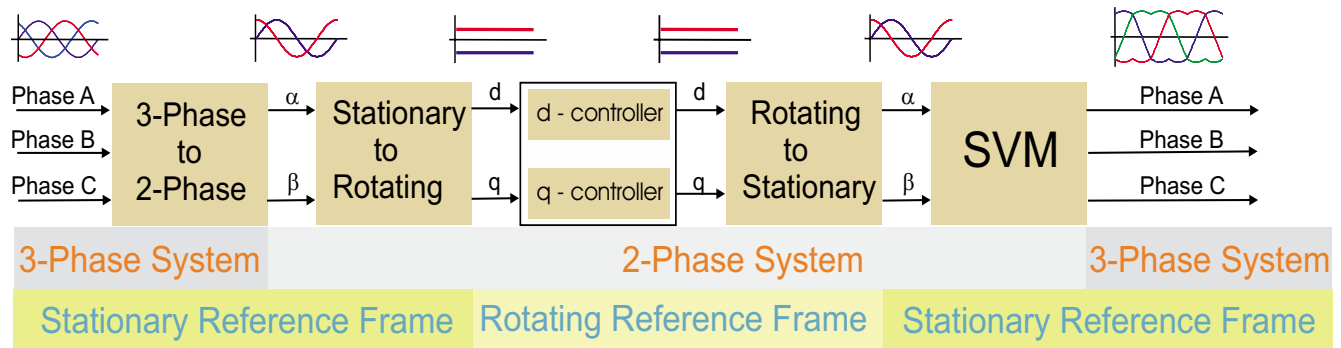


Figure 2-4. Vector Control Transformations

2.3.2 Description of the Vector-Control Algorithm

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

To achieve the goal of the PM synchronous motor control, the algorithm utilizes feedback signals. The essential feedback signals are: 3-phase stator current and the stator voltage. For the stator voltage the regulator output is used. For correct operation, the presented control structure requires position and speed information. This information is estimated by special algorithms because there is no position sensor in this application.

The fast control loop executes two independent current-control loops. They are the direct and quadrature-axis current (i_{sd}, i_{sq}) PI controllers. The direct-axis current (i_{sd}) is used to control the rotor magnetizing flux. The quadrature-axis current (i_{sq}) corresponds to the motor torque. The current PI controllers' outputs are summed with the corresponding d and q axis components of the decoupling stator voltage. Thus we obtain the desired space-vector for the stator voltage, which is applied to the motor.

The fast control loop executes all the necessary tasks to be able to achieve an independent control of the stator-current components. This includes:

- 3-Phase Current Reconstruction
- Forward Clarke Transformation
- Forward and Backward Park Transformations
- BEMF Observer
- Angle-Tracking Observer
- Stator Voltage Decoupling
- DC-Bus Voltage Ripple Elimination
- Space Vector Modulation (SVM)

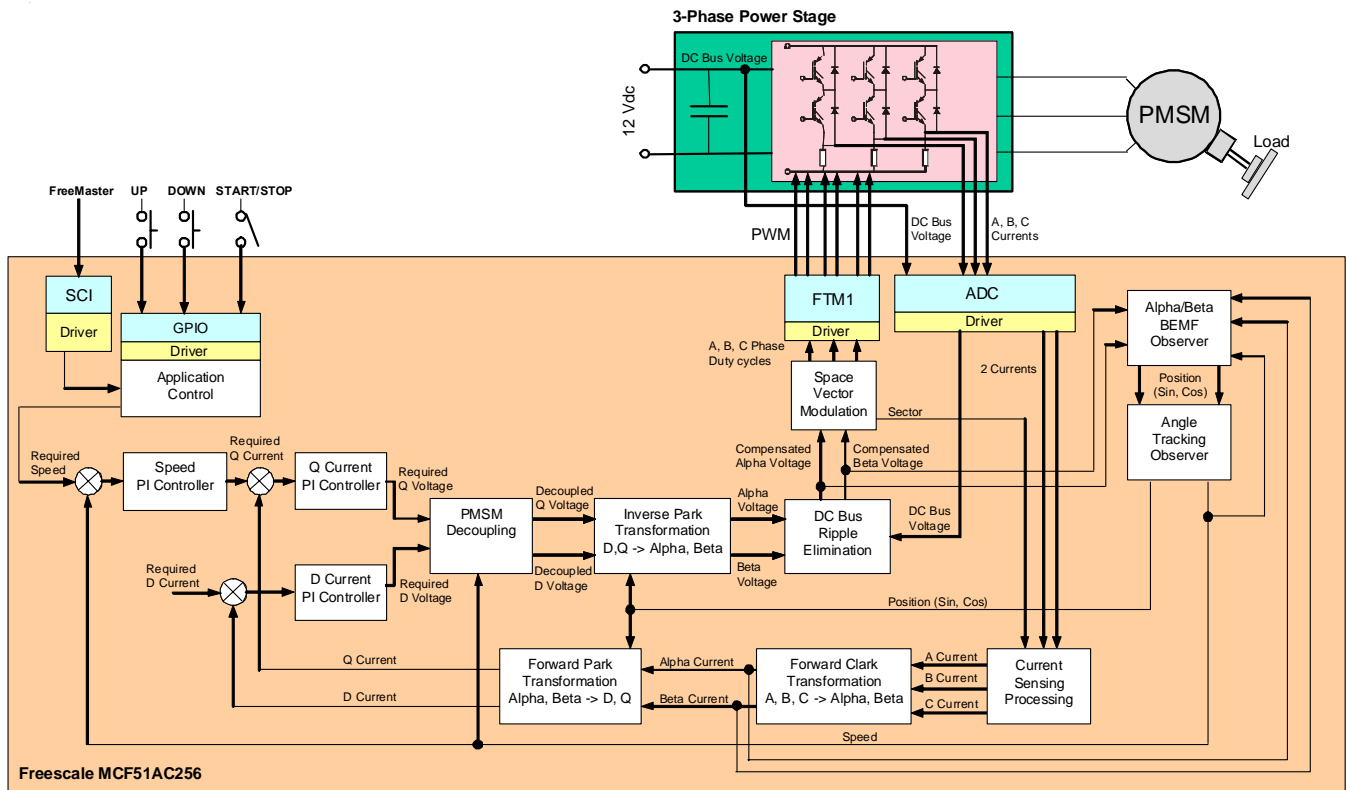


Figure 2-5. PMSM Vector-Control Algorithm Overview

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

2.3.3 Stator Voltage Decoupling

The direct-axis stator current i_{sd} (rotor field component) and the quadrature-axis stator current i_{sq} (torque-producing component) must be controlled independently. However, the equations of the stator-voltage components are coupled. The direct-axis component u_{sd} also depends on i_{sq} , and the quadrature-axis component u_{sq} also depends on i_{sd} .

The stator-voltage components u_{sd} and u_{sq} cannot be considered as decoupled control variables for the rotor flux and electromagnetic torque. The stator currents i_{sd} and i_{sq} can be independently controlled (decoupled control), only if the stator-voltage equations are decoupled, so these stator-current components are indirectly controlled by controlling the terminal voltages of the synchronous motor.

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

$$u_{sd} = u_{sd}^{lin} + u_{sd}^{decouple} \quad \text{Eqn. 2-23}$$

$$u_{sq} = u_{sq}^{lin} + u_{sq}^{decouple} \quad \text{Eqn. 2-24}$$

The linear components are defined:

$$u_{sd}^{lin} = R_s \times i_d + L_d \times \frac{di_d}{dt} \quad \text{Eqn. 2-25}$$

$$u_{sq}^{lin} = R_s \times i_q + L_q \times \frac{di_q}{dt} + \omega_{el} \times K_M \quad \text{Eqn. 2-26}$$

The decoupling components are defined:

$$u_{sd}^{decouple} = -L_q \times \omega_{el} \times i_q \quad \text{Eqn. 2-27}$$

$$u_{sq}^{decouple} = L_d \times \omega_{el} \times i_d \quad \text{Eqn. 2-28}$$

The decoupling components $u_{sd}^{decouple}, u_{sq}^{decouple}$ are evaluated from the stator-voltage equations [Equation 2-17](#) and [Equation 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 and the decoupling components are feedforwarded to the d and q voltages according to the following equations:

$$u_{sd} = u_{sd}^{lin} - L_q \times \omega_{el} \times i_q \quad \text{Eqn. 2-29}$$

$$u_{sq} = u_{sq}^{lin} + L_d \times \omega_{el} \times i_d \quad \text{Eqn. 2-30}$$

The above equations [Equation 2-29](#) and [Equation 2-30](#) are evaluated in the *Decoupling* block (see [Figure 2-5](#)).

2.3.4 Space Vector Modulation

Space Vector Modulation (SVM) can directly transform the stator-voltage vectors from the two-phase 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 2-6](#). Regarding the 3-phase power stage configuration, as shown in [Figure 2-6](#), eight possible switching states (vectors) are feasible.

They are given by combinations of the corresponding power switches. A graphical representation of all combinations is the hexagon shown in Figure 2-7. There are six non-zero vectors, U_0 , U_{60} , U_{120} , U_{180} , U_{240} , U_{300} , and two zero vectors, O_{000} and O_{111} , defined in α , β coordinates.

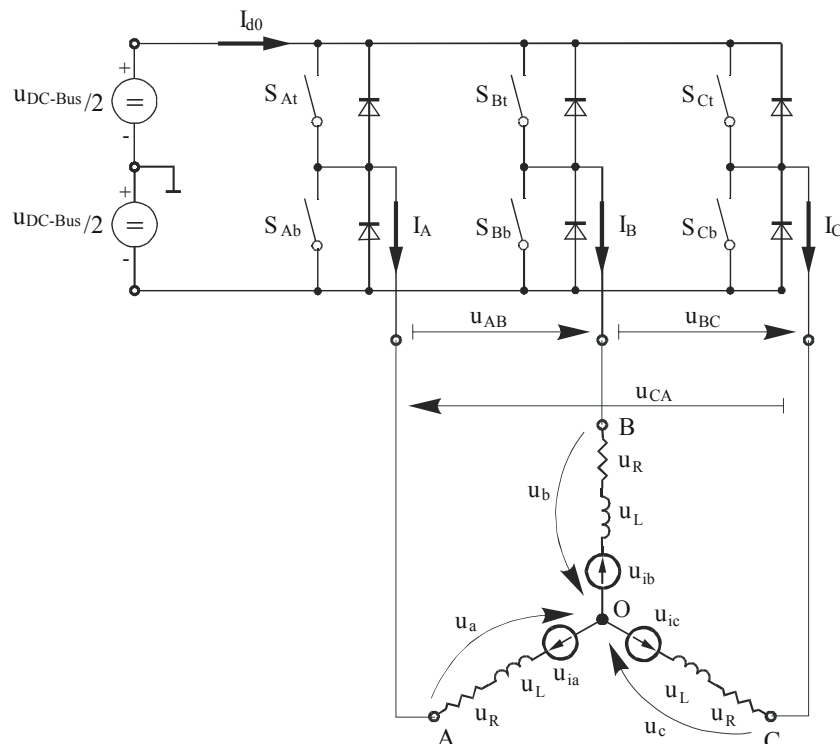


Figure 2-6. Power Stage Schematic Diagram

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

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	$2U_{DC-Bus}/3$	$-U_{DC-Bus}/3$	$-U_{DC-Bus}/3$	U_{DC-Bus}	0	$-U_{DC-Bus}$	U_0
1	1	0	$U_{DC-Bus}/3$	$U_{DC-Bus}/3$	$-2U_{DC-Bus}/3$	0	U_{DC-Bus}	$-U_{DC-Bus}$	U_{60}
0	1	0	$-U_{DC-Bus}/3$	$2U_{DC-Bus}/3$	$-U_{DC-Bus}/3$	$-U_{DC-Bus}$	U_{DC-Bus}	0	U_{120}
0	1	1	$-2U_{DC-Bus}/3$	$U_{DC-Bus}/3$	$U_{DC-Bus}/3$	$-U_{DC-Bus}$	0	U_{DC-Bus}	U_{240}
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}

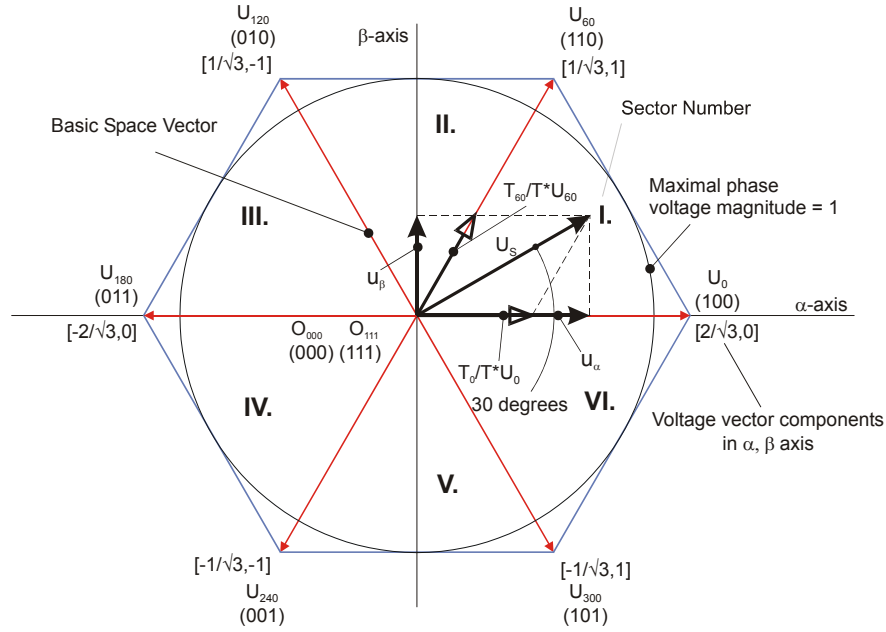


Figure 2-7. Basic Space Vectors and Voltage-Vector Projection

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

The SVM technique consists of several steps:

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

The principle of SVM is the application of the voltage vectors U_{xxx} and O_{xxx} for certain instances in such a way that the “mean vector” of the PWM period T_{PWM} is equal to the desired voltage vector.

This method gives the greatest variability in arranging the zero and non-zero vectors during the PWM period. One can arrange these vectors to lower switching losses; another might want to reach a different result, such as center-aligned PWM, edge-aligned PWM, minimal switching, and so on.

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:

$$T_{PWM} \times U_{S[\alpha, \beta]} = T_1 \times U_x + T_2 \times U_{x\pm60} + T_0 \times (O_{000} \vee O_{111}) \quad \text{Eqn. 2-31}$$

$$T_{PWM} = T_1 + T_2 + T_0 \quad \text{Eqn. 2-32}$$

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 2-31](#) splits into equations [Equation 2-33](#) and [Equation 2-34](#).

$$T_{PWM} \times U_{sx} = T_1 \times U_x \quad \text{Eqn. 2-33}$$

$$T_{PWM} \times U_{s(x \pm 60)} = T_2 \times U_{x \pm 60} \quad \text{Eqn. 2-34}$$

By solving this set of equations, we can calculate the necessary duration for the application of the sector base vectors U_x , $U_{x \pm 60}$ during the PWM period T_{PWM} to produce the right stator voltages.

$$T_1 = \frac{|U_{sx}|}{|U_x|} T_{PWM} \quad \text{for vector } U_x \quad \text{Eqn. 2-35}$$

$$T_2 = \frac{|U_{sx}|}{|U_{x \pm 60}|} T_{PWM} \quad \text{for vector } U_{x \pm 60} \quad \text{Eqn. 2-36}$$

$$T_0 = T_{PWM} - (T_1 + T_2) \quad \text{either for } O_{000} \text{ or } O_{111} \quad \text{Eqn. 2-37}$$

2.3.5 Motor Position Alignment

In the presented design, no position sensor is used. The BEMF observer gives us the position when the motor is spinning and therefore it is necessary to start the motor up in open-loop manner. We need to know the exact rotor position before the motor is started. One possible, and very easy implementable method is the rotor alignment to a predefined position. The motor is powered by a selected static voltage pattern (usually the zero position in the sinewave table) and the rotor aligns to “zero” position. The alignment is done only once during the motor start. Figure 2-8 shows the motor alignment. Before the constant current vector is applied to the stator, the rotor position is not known. After a stabilization period, the rotor flux should be aligned to the stator flux. In practice this is true, when the external load torque is low enough compared to the torque produced by the alignment vector.

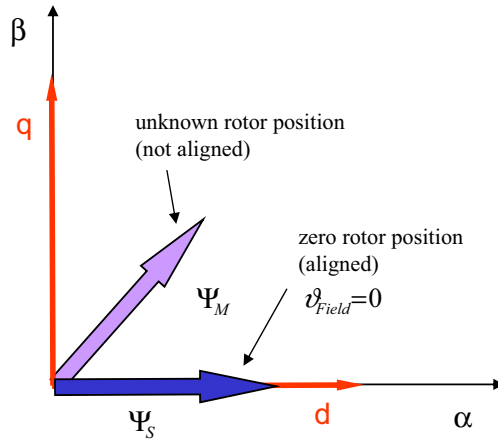


Figure 2-8. Rotor Alignment Stabilization — PMSM Starting Mode



Chapter 3

System Concept

3.1 System Specification

The system is designed to drive a 3-phase PM synchronous motor. The application meets the following performance specifications:

- It is targeted at the MCF51AC256 ColdFireV1 Controller.
- It is running on the 3-phase Motor Control Drive board with the MCF51AC256 Daughter Board.
- Control technique incorporates:
 - Vector control of 3-phase PM synchronous motor without position sensor.
 - Closed-loop speed control.
 - Both directions rotation.
 - Close-loop current control.
 - Flux and torque independent control.
 - Startup with alignment.
 - Field weakening is not implemented in this software version.
 - Reconstruction of 3-phase motor currents from two shunt resistors.
 - 125 μ s sampling period on MCF51AC256 with FreeMASTER recorder.
- FreeMASTER software control interface (motor start/stop, speed setup).
- FreeMASTER software monitor.
 - FreeMASTER software graphical control page (required speed, actual motor speed, start/stop status, DC-Bus voltage level, motor current, system status).
 - FreeMASTER software speed scope (observes actual and desired speeds, DC-Bus voltage, and motor current).
 - FreeMASTER software high-speed recorder (reconstructed motor currents, vector control algorithm quantities).
- DC-Bus over-voltage and under-voltage, over-current, overload, and startup fail 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 12 V, 5 A
- 3-Phase Motor Control Drive board
- 3-phase PM synchronous motor (default configuration for motor TG Drives TGT2-0032-30-24)
- MCF51AC256 daughter board

Sensorless PM Sinusoidal Motor Vector Control on MCF51AC256, Rev.0

The MCF51AC256, populated on the MCF51AC256 Daughter Board, executes the control algorithm. In response to the user interface and feedback signals, it generates PWM signals for the 3-Phase Motor Control Drive. Low-voltage waveforms, generated by the DC to AC inverter, are applied to the motor.

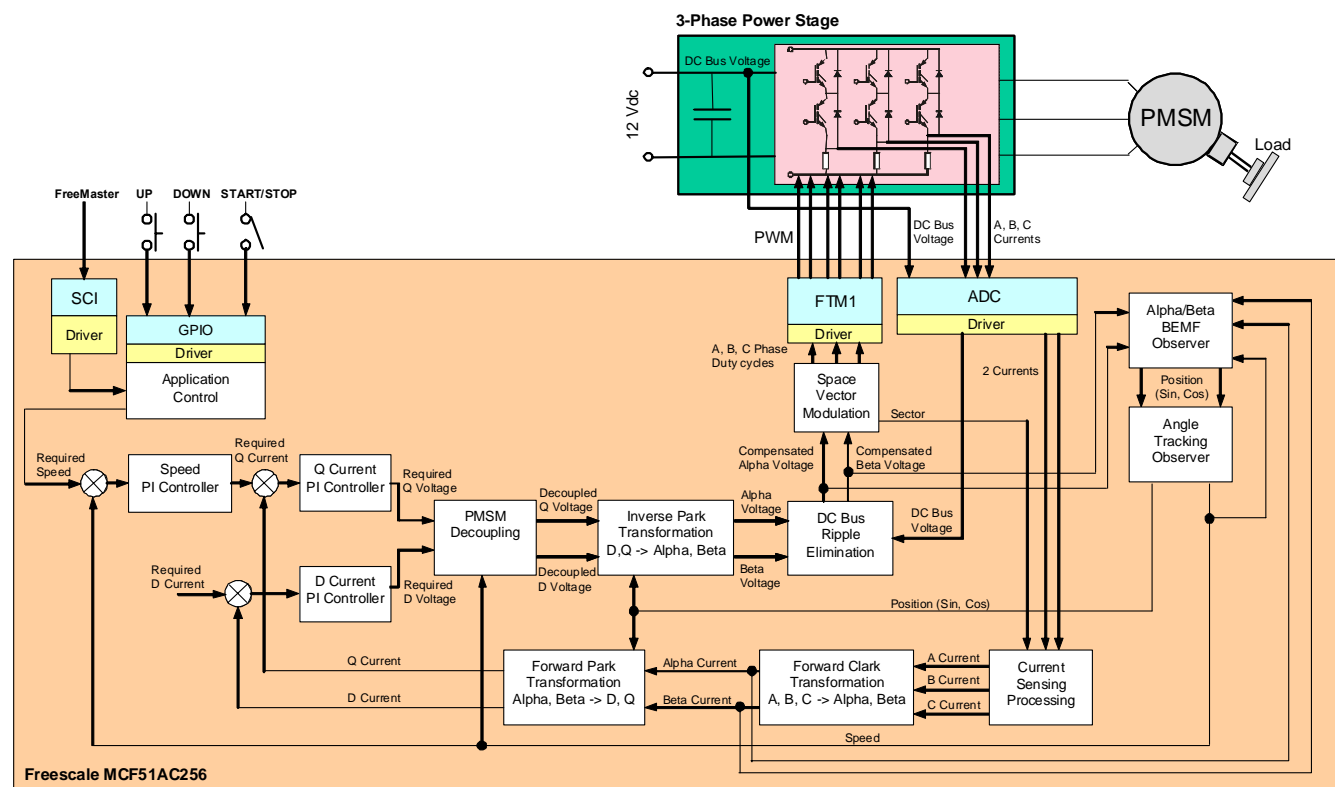



Figure 3-1. System Concept

3.3 Control Process

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

The DC-Bus current and voltage are sampled with ADC. The ADC sampling is triggered by FTM1 channel three and synchronized to the PWM signal. A digital filter is applied to the sampled values. The 3-phase motor current is reconstructed from two samples taken from the inverter's shunt resistors. The reconstructed 3-phase current is then transformed into space vectors and used by the FOC algorithm.

The rotor position and speed are given by the BEMF observer and the Angle-Tracking Observer. The BEMF observer calculates the sine and cosine of the angle using the alpha-beta voltage, alpha-beta current, and speed. The sine and cosine of the BEMF angle are supplied into the Angle-Tracking Observer, which returns the position and speed. Based on measured feedback signals, the FOC algorithm performs a vector-control technique, 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. The 3-phase stator voltage is generated by means of a 3-phase voltage source inverter and applied to the motor, which is connected to the power stage terminals.

The application can be controlled via a FreeMASTER control page from a host PC. The FreeMASTER communicates via serial RS232 protocol by means of a virtual COM port connected to PC via USB.

The application state machine handles the operating states of the drive. There are five states of the drive. The actual operating state is indicated by the FreeMASTER control page. In the case of over-voltage, under-voltage, or over-current, overload, and startup fail, 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 PM synchronous motor. It consists of the following modules:

- Host PC
- MCF51AC256 Daughter Board
- 3-Phase Motor Control Drive
- 3-phase PM synchronous motor

The application hardware system configuration is shown in [Figure 4-1](#).

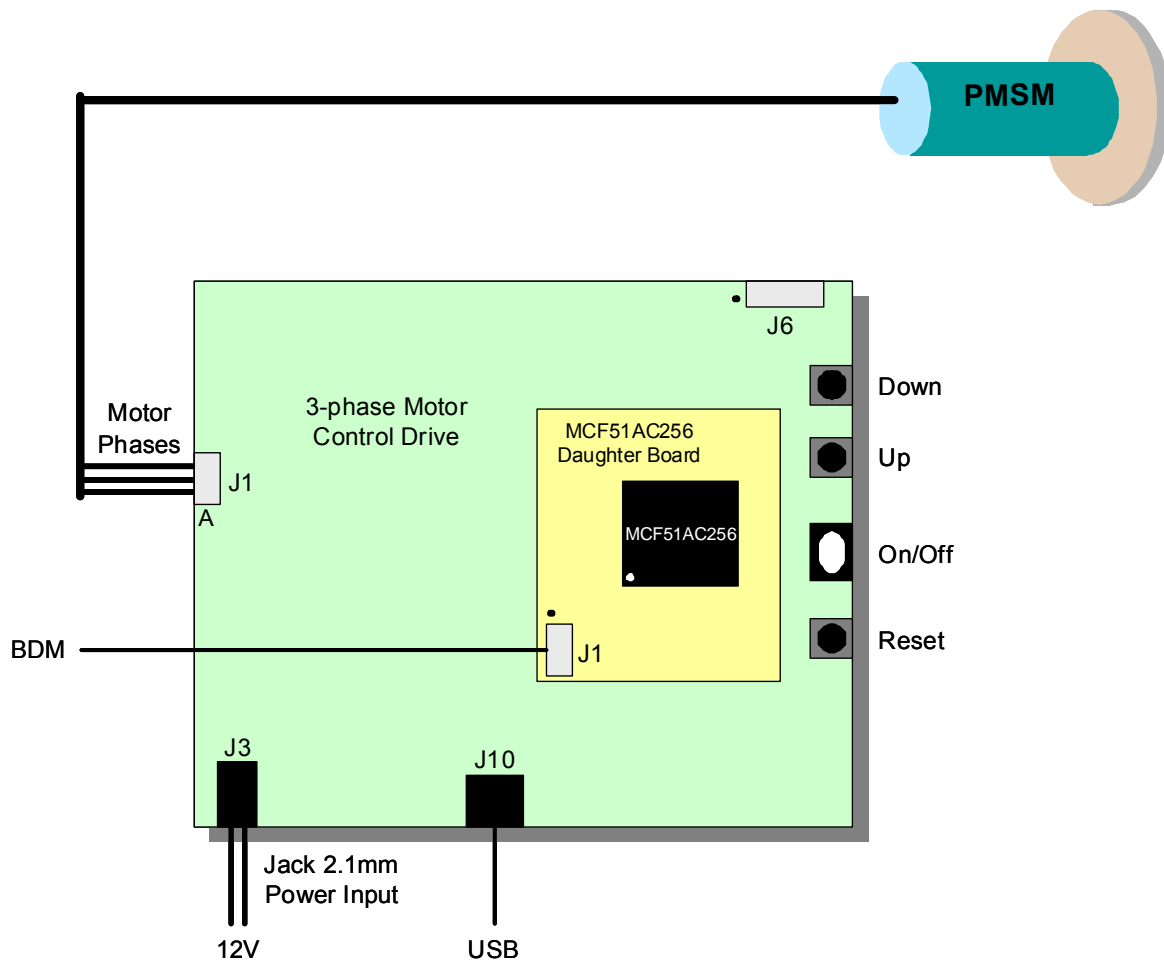


Figure 4-1. Hardware System Configuration

All system parts are supplied and documented in these references:

- MCF51AC256 Daughter Board:
 - Using Freescale's MCF51AC256 as the controller.
 - Supplied as an MCF51AC256 Daughter Board.
 - Described in the *MCF51AC256 Daughter Board User Manual*.
- 3-Phase BLDC/PMSM Low Voltage Motor Control Drive Board:
 - Low-voltage 3-phase power stage with an input of 12 V/5 A and variable voltage 3-phase MOSFET bridge output.
 - Described in the *3-phase BLDC/PMSM Low-Voltage Motor Control Drive User Manual*.

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

4.2 3-Phase Motor Control Drive Using MCF51AC256

The 3-Phase Motor Control Drive is based on an optimized PCB and power supply design. It demonstrates the abilities of the Freescale controllers and provides a hardware tool to help in the development of applications using several Freescale controllers targeted at motor-control applications.

The 3-Phase Motor Control Drive has an option of universal connection of Freescale controller like MCF51AC256. The MCF51AC256 Daughter Board is a small plug-in module that is populated with the MCF51AC256 chip. This board is plugged into the J7 and J8 connectors.

The 3-Phase Motor Control Drive with the MCF51AC256 Daughter Board is a controller and power stage set; it includes an MCF51AC256 part, encoder interface, communication options, digital and analog power supplies, and 3-phase MOSFET bridge output.

The 3-Phase Motor Control Drive using the MCF51AC256 Daughter Board setup is designed for the following purposes:

- To allow new users to become familiar with the features of the MCF51AC256 architecture.
- To serve as a platform for real-time motor control 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 BDM™ port. The breakpoint features of the BDM 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 BDM port simplifies the task of the developer considerably.
- To serve as a platform for hardware development. The hardware platform enables motor to be connected with a position/speed sensor. The BDM port's non-intrusive design means that 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 MCF51AC256, and can be used to develop real-time software and hardware products based on the MCF51AC256. It provides the features necessary to write and debug software, demonstrate the functionality of that software, and to interface with the customer's application specific device(s).

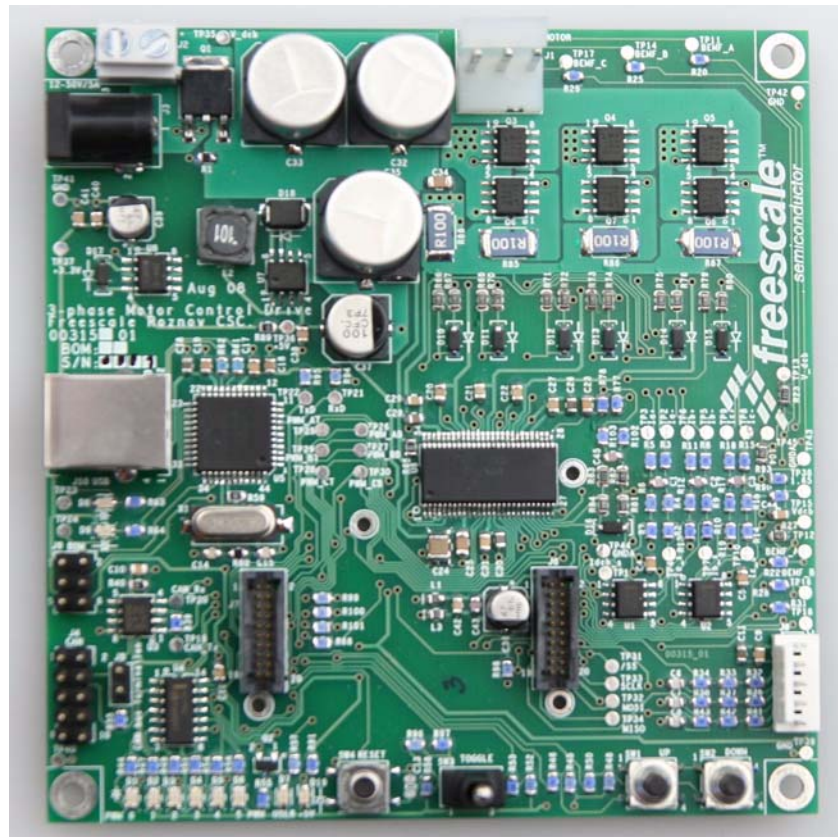


Figure 4-2. 3-Phase Motor Control Drive Top View



Figure 4-3. MCF51AC256 Daughter Board Top View

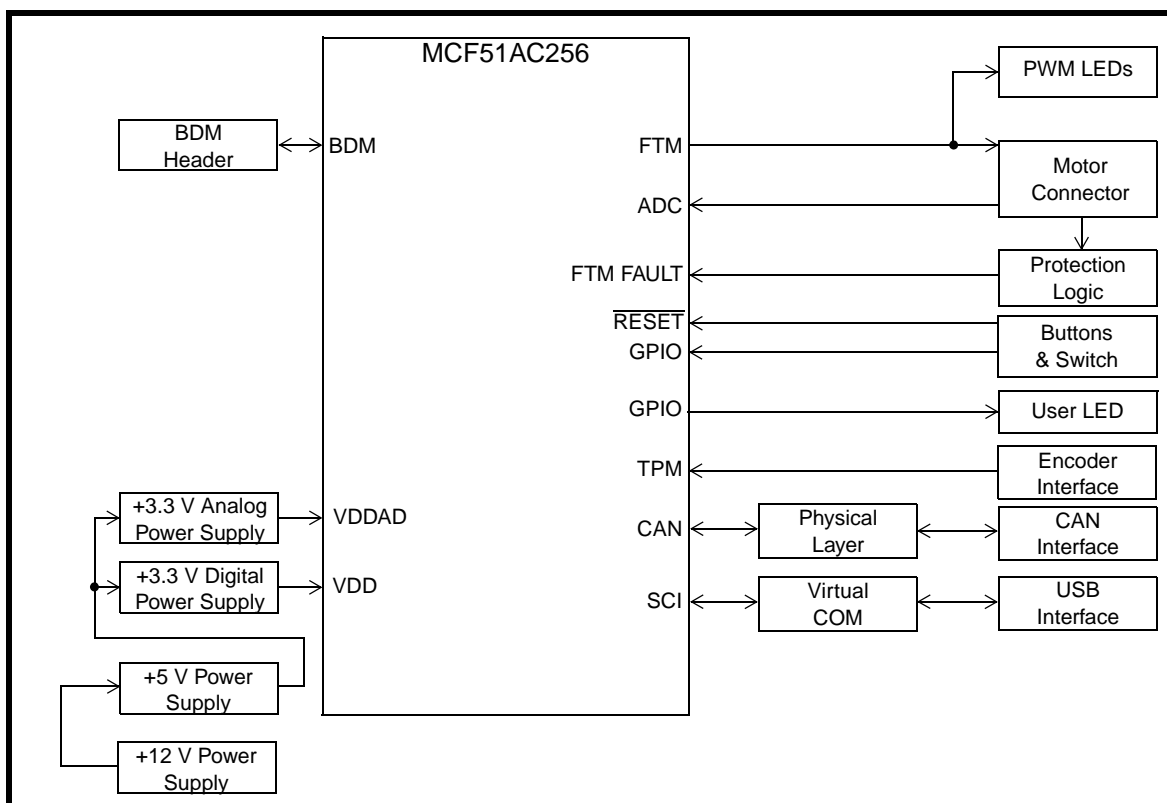


Figure 4-4. Block Diagram of the MCF51AC256 Peripherals and the Board

The 3-Phase Motor Control Drive with the MCF51AC256 Daughter Board are flexible enough to allow full exploitation of the MCF51AC256's features to optimize the performance of the user's motor control end product. See [Figure 4-2](#), [Figure 4-3](#), and [Figure 4-4](#).

4.3 Motor Specifications — Example

The motor used in this application is a standard production 3-phase PM synchronous motor with an incremental encoder mounted on the shaft. The motor and sensor have the following specifications:

Table 4-1. Specifications of the Motor and Incremental Sensor

Motor Specification:	Motor Type:	3-Phase PM Synchronous Motor TG Drives TGT2-0032-30-24/T0PS1KX
	Nominal Voltage (line-to-line)	30 V RMS
	Nominal Speed	3000 RPM [314 rad/s]
	Nominal Current (phase)	5.2 A RMS
	Nominal Torque	0.32 Nm
Motor Model Parameters	Stator Winding Resistance (line-to-line)	1.1 Ω
	Stator Winding Inductance d axis	390 μ H

Table 4-1. Specifications of the Motor and Incremental Sensor (Continued)

	Stator Winding Inductance q axis	470 μ H
	Number of Pole-Pairs	3
Position Sensor Specification:	Manufacturer:	INDUcoder
	Type:	ES 28-6-1024-05-D-R
	Line Count	1024
	Output	5 V \pm 10 % TTL



Chapter 5

Software Design

5.1 Introduction

This section describes the software design of the PMSM vector control application. First, the numerical scaling in fixed-point fractional arithmetic of the controller 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 PMSM vector control application uses a fractional representation for all real quantities, except time. Although MCF51AC256 does not have fractional arithmetics instructions support, the library algorithms and the application use this arithmetics. The N-bit signed fractional format is represented using 1.[N-1] format (one sign bit, N-1 fractional bits). Signed fractional numbers (SF) lie in the following range:

$$-1.0 \leq SF \leq +1.0 \cdot 2^{-(N-1)} \quad \text{Eqn. 5-1}$$

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 Analog Quantities

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

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

where:

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

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 a maximum measurable voltage range of the power stage. For the demo hardware board, the range is $V_{MAX} = 16.7$ V. Variable values in fractional format are defined by the following equation:

$$(\text{Frac16})\text{voltage_variable} = \frac{V_{MEASURED}}{V_{MAX}} \quad \text{Eqn. 5-3}$$

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

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

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

```
#define U_MAX          (9.642)
#define I_MAX          (3.52)
```

Please note, that I_MAX corresponds to a half range of the ADC converter input voltage (0–3.3 V). For motor phase-current sensing, the zero current level is shifted into the middle of this range (=1.65 V). Thus, the maximum positive and negative phase-current can only be half range of the ADC. The value sensed from the ADC is then shifted to eliminate the 1.65 V offset and multiplied by two to fit into the whole range.

5.2.3 Scaling of Angles

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

$$-\pi \approx 0x8000 \quad \text{Eqn. 5-5}$$

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

5.2.4 Scaling of Parameters

Real-value parameters, in equations such as the decoupling voltage and so on, 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 analog value, which forms the particular equation. The adjusted value is then split into a fractional range of $[-1, 1)$ and an N-bit scale. The scaling process can be explained by a simple example of Ohm's law equation.

$$V_{\text{real}} = R \times I_{\text{real}} \quad \text{Eqn. 5-7}$$

$$V_{\text{Frac16}} \times V_{MAX} = R \times I_{\text{Frac16}} \times I_{MAX} \quad \text{Eqn. 5-8}$$

$$V_{\text{Frac16}} = \left(R \times \frac{I_{MAX}}{V_{MAX}} \right) \times I_{\text{Frac16}} = R_{\text{adjusted}} \times I_{\text{Frac16}} \quad \text{Eqn. 5-9}$$

Let's substitute the following values:

$$R = 300 \, \Omega, I_{MAX} = 3.52 \, \text{A}, V_{MAX} = 16.7 \, \text{V}$$

The R_{adjusted} can be evaluated as follows:

$$R_{\text{adjusted}} = R \times \frac{I_{\text{MAX}}}{V_{\text{MAX}}} = 300 \times \frac{3.52}{16.7} = 63.2335 \quad \text{Eqn. 5-10}$$

The R_{adjusted} is out of range of the signed fractional number. We need to right shift the value by R_{Scale} -bits to fit into the desired range. Therefore, we introduce a scale (shift) part of the parameter. For this example we have to shift the result by $R_{\text{Scale}} = 6$ bits. The resistor value scaled to the signed fractional range is as follows:

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

$$R_{\text{Frac16}} = R \times \frac{I_{\text{MAX}}}{V_{\text{MAX}}} \times 2^{-R_{\text{Scale}}} = 300 \times \frac{3.52}{16.7} \times 2^{-6} = 0.9880 \quad \text{Eqn. 5-12}$$

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

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

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

All algorithms and motor parameters are scaled to their 16-bit, or possibly 32-bit fractional representation. For most parameters, there are two definitions. One evaluates the parameter fractional representation, and the other defines the required N-bit shift = scale.

The voltage-scaling factor is:

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

where:

- S_V is the scaling factor.
- V_{SYSRANGE} is the range of system representation voltage.
- V_{MAX} is the range of real voltage.

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

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

Then the real variable is:

$$V_{\text{real}} = \frac{(\text{Frac16})\text{voltage_variable}}{S_V} \quad \text{Eqn. 5-16}$$

The system parameters are then calculated as follows:

$$R_{\text{adjusted}} = R \times \frac{S_V}{S_I} = 300 \times \frac{1/16.7}{1/3.52} = 63.2335 \quad \text{Eqn. 5-17}$$

5.3 Application Overview

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

The **FTM1 Overflow** interrupt service routine is executed, when the FTM counter is at the maximum value, with a 62.5 μs period. It manages the ADC hardware trigger to be turned on every second PWM period.

The **ADC1** interrupt service routine is executed to read and start another sample conversion. It reads the DC-Bus voltage and two current samples. The current-control loop is performed at the end of the last sample conversion and the speed-control loop is performed in this interrupt every 40th time to reach a 5 ms period.

The **FTM1 Fault** interrupt service routine is executed on an over-current event to handle an over-current fault. It is executed only if the fault condition occurs.

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

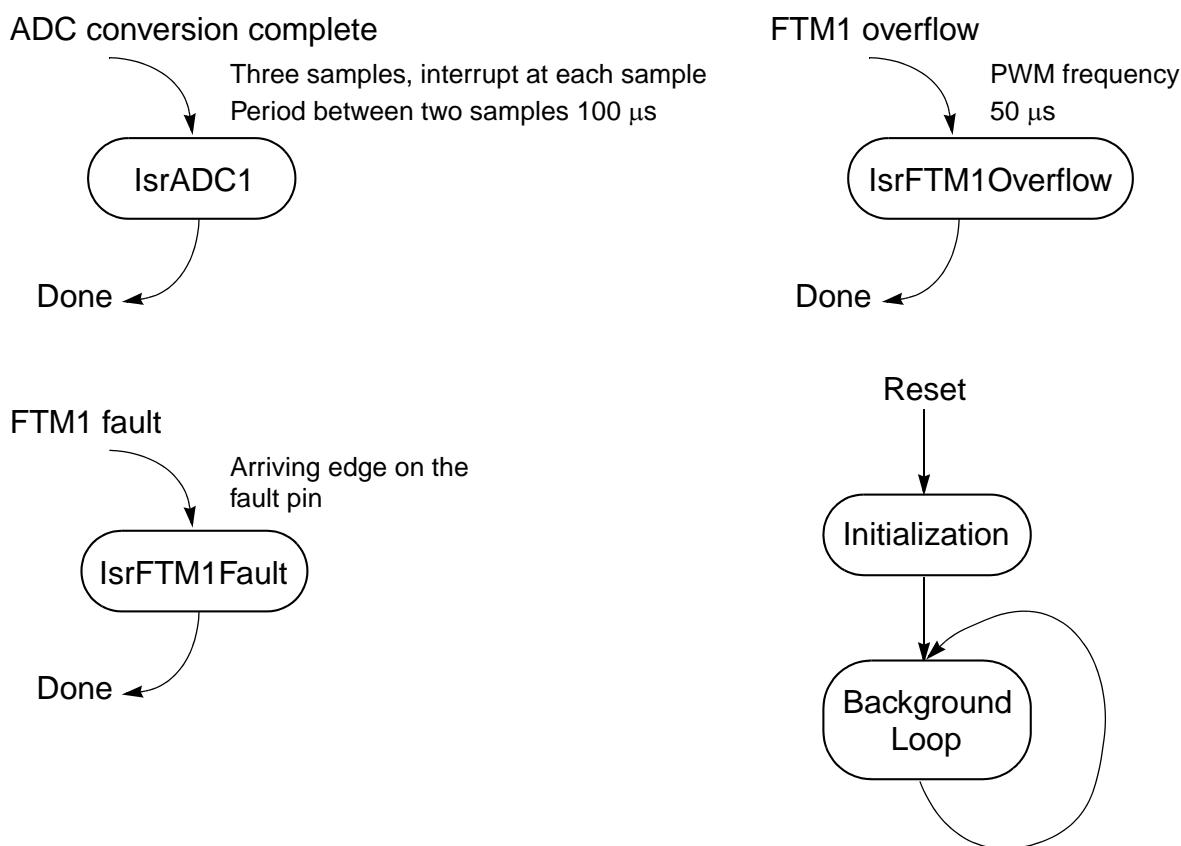


Figure 5-1. Main Data Flow

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

5.3.1 ADC Conversion Timing and PWM Reload Interrupts

When the FTM1 counter reaches its maximum value (PWM modulo) an interrupt is generated — `IsrFTM1Overflow`. In this interrupt, the ADC hardware trigger is enabled in such a way the trigger is generated just at the every second PWM period.

The PWM module is configured to run in center-aligned mode with counter modulo = 1500, which corresponds to a switching frequency of 16 kHz at a 24 MHz bus clock (PWM cycle period = 62.5 μ s). The trigger signal is generated when the FTM1 counter reaches the compare value of the channel three. The channel three value is set to 290 (evaluated to get the two current samples as close to the middle of the PWM on-pulse as possible) so as the trigger is generated when the counter counts down after the reload. When the counter reaches this value, the hardware trigger is generated and the DC-bus voltage conversion is initiated. When the sample is converted, the ADC interrupt is generated where the ADC is switched for the software triggering. The ADC is reconfigured for the first current measurement, and in the next ADC interrupt for the second current measurement. In this last ADC interrupt the current control (FOC) loop is calculated. This process is depicted in Figure 5-2.

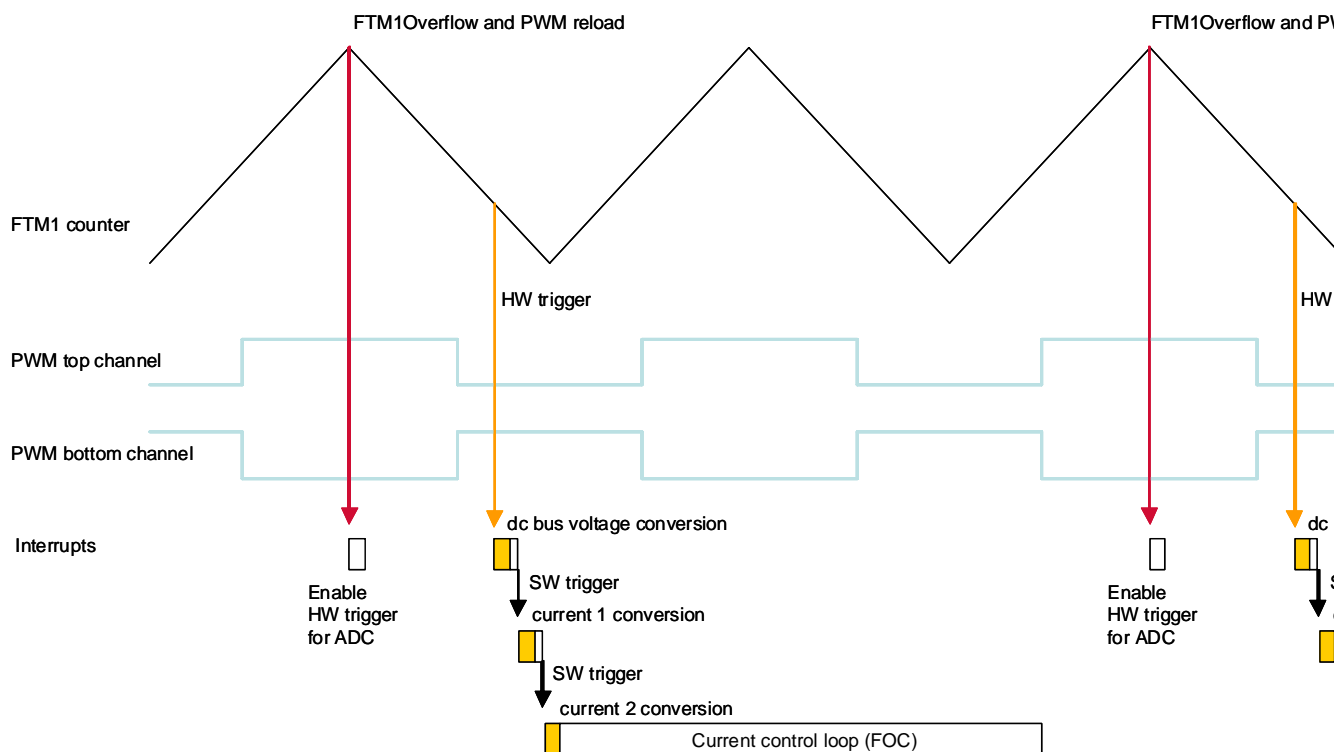


Figure 5-2. ADC and PWM Reload Interrupts

As can be seen in this figure, the FOC calculation does not fit into the area from the ADC sensing to the next period reload. That is why the loop is calculated at the half frequency of the PWM frequency.

5.3.2 3-Phase Current Sampling

The vector-control algorithm requires the sensing of the three motor phase currents. As the sum of the three currents in the motor windings is zero, only two currents are measured and the third one is calculated. Which phases are measured changes according to the actual vector, in other words the phases with the largest PWM on-pulse are selected to get the best current information.

The 3-phase stator currents are measured by means of current shunt sensor mounted in each phase between the lower switch source and the negative rail. The current pulses are sampled at exactly timed intervals. A voltage drop on the shunt resistors is amplified by operational amplifiers and shifted up by 1.65 V. The resultant voltage is converted by the ADC; see Figure 5-3.

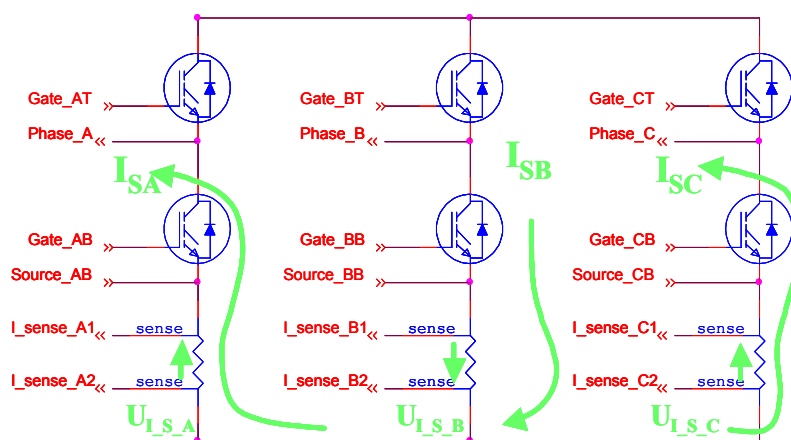


Figure 5-3. Current Shunt Resistors

5.3.3 Position and Speed by Observer

This application is sensorless, in other words it does not have any position or speed sensor mounted on the rotor shaft. Thus, the application has two special algorithms called observers to estimate the position and the speed of the rotor. These algorithms are regularly called in the main controlled loop after the currents are measured and transformed into the alpha-beta frame.

5.3.3.1 BEMF Observer

The BEMF observer (see Figure 5-4.) is an algorithm, which gives us the information about the sine and cosine of the angle of the electrical position. The observer function is called after the transformation of the measured current into the alpha-beta frame.

The observer has five inputs that are the alpha-beta voltage applied to the motor, alpha-beta current measured on the motor phases, and the rotor speed. The algorithm requires the electrical parameters of the motor to be correctly measured and properly scaled and set up.

It is obvious that the accuracy of the BEMF estimates is determined by the correctness of used motor parameters (R, L), by fidelity of the reference stator voltage, and by quality of compensator such as bandwidth, phase lag, and so on.

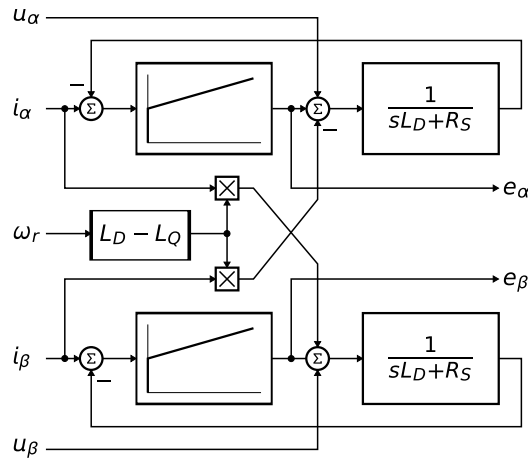


Figure 5-4. Block Diagram of Back-EMF Observer

This observer works correctly when the motor rotates. When the motor is stopped, it may give incorrect information. Therefore, this observer can be activated when the speed of the motor is at a certain level.

5.3.3.2 Angle-Tracking Observer

Another function that is called calculates the angle-tracking observer algorithm. It is called right after the BEMF observer calculation.

The observer requires two input arguments as sine and cosine samples. The angle-tracking observer compares values of the input signals $\sin(\theta)$ and $\cos(\theta)$ with their corresponding estimations $\sin(\hat{\theta})$ and $\cos(\hat{\theta})$. As in any common closed-loop systems, the intent is to minimize observer error towards zero value. The observer error is given by subtraction of the estimated rotor angle $\hat{\theta}$ from the actual rotor angle θ (see [Figure 5-5](#)).

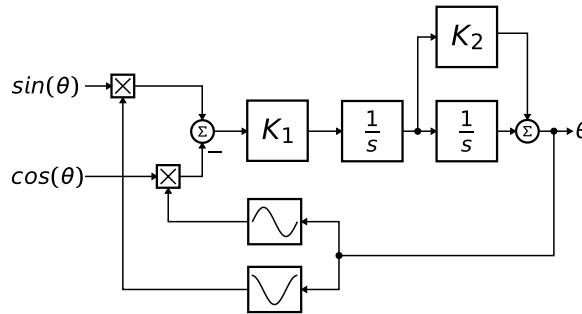


Figure 5-5. Block Scheme of the Angle-Tracking Observer

A secondary effect of this observer is the output angle — it is filtered — and the speed calculation. Therefore additional speed-calculation algorithm is not needed.

5.3.3.3 Motor Startup

As mentioned in [Section 5.3.3.1, “BEMF Observer,”](#) it is not possible to get valuable position and speed information when the motor is stopped or at low speed. Therefore it is necessary to somehow make the motor run before the BEMF observer output is used.

So, how to make the motor run with a certain speed without any position information? One possible way can be applying constant current to create constant torque. With constant torque the motor should accelerate with more or less constant acceleration rate (see [Figure 5-6](#)). Assuming this, we can program a ramp dependent on time with a constant incrementation — acceleration rate. Therefore the predicted speed will be a sum of the acceleration rate along time (see the green curve of [Figure 5-6](#)). This speed ramp can be adapted according to the real acceleration using the constant current. The predicted position will then be a simple sum of the speed along time (see blue curve of [Figure 5-6](#)). The code lines representation follows:

$$\text{Acceleration} = \text{const} \quad \text{Eqn. 5-18}$$

$$\text{PredictedSpeed} = \text{PredictedSpeed} + \text{Acceleration} \quad \text{Eqn. 5-19}$$

$$\text{PredictedPosition} = \text{PredictedPosition} + \text{PredictedSpeed} \quad \text{Eqn. 5-20}$$

The predicted speed, measured current, and applied voltage are supplied to the BEMF observer. At certain moment, when we are sure that the BEMF observer already calculates correct position, the application can be switched to use the estimated speed (see the red curve of [Figure 5-6](#)) and position values (see the cyan curve of [Figure 5-6](#)).

Because the estimated and predicted positions differ and immediate switching could be disturbing for the motor spinning, a trick was invented to make it smoother. This trick consists in adding portion of the difference between the estimated and predicted position to the predicted position. The portion is given by multiplication of the positions’ difference by a coefficient smaller than one. This coefficient is increased along time, till it reaches almost one. Such a position result then creates a smooth transition between the predicted and estimated position curve (see the purple curve of [Figure 5-6](#)). When the mentioned coefficient is almost one (in other words the position result almost catches the estimated position) the application is fully switched to use the estimated position. The code lines are the following:

$$\text{CatchUpAngle} = \text{PredictedPosition} + (\text{EstimatedAngle} - \text{PredictedPosition}) \times \text{CatchUpRatio} \quad \text{Eqn. 5-21}$$

$$\text{Speed} = \text{PredictedSpeed} + (\text{EstimatedSpeed} - \text{PredictedPositionSpeed}) \times \text{CatchUpRatio} \quad \text{Eqn. 5-22}$$

$$\text{CatchUpRatio} = \text{CatchUpRatio} + \text{const} \quad \text{Eqn. 5-23}$$

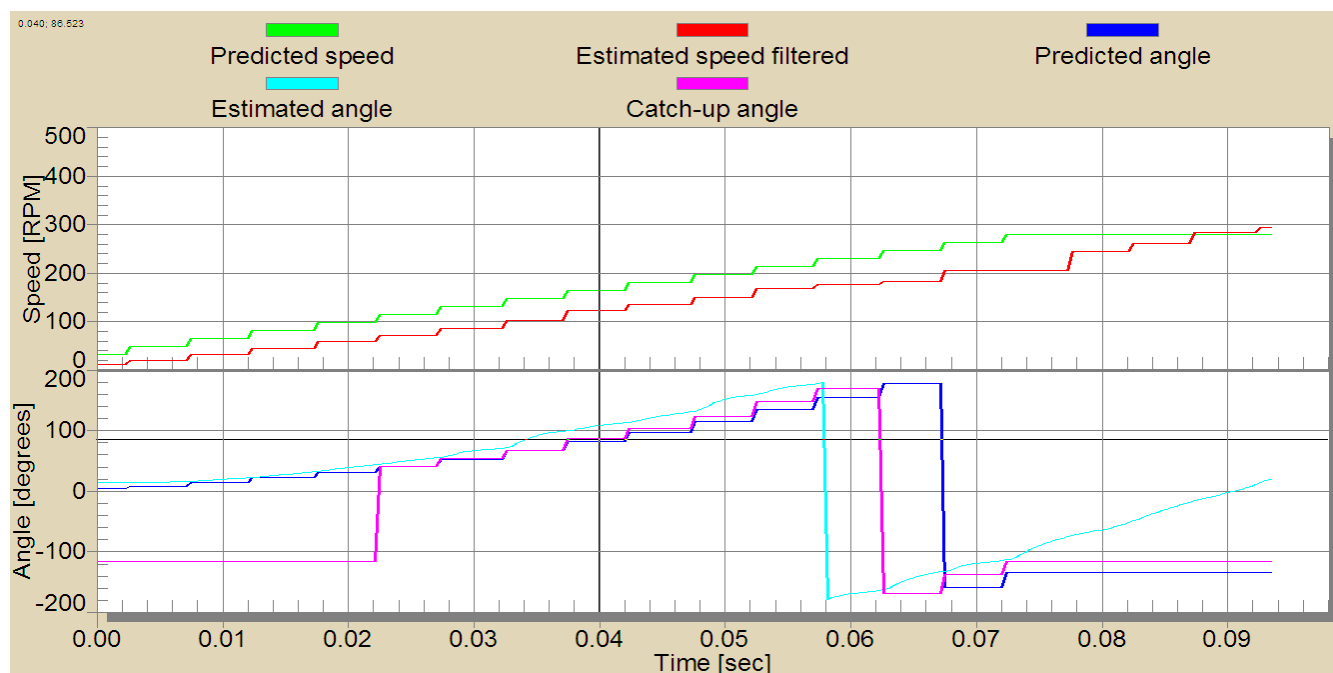


Figure 5-6. Motor Startup

As the startup is “blind”, in other words for some reason the acceleration of the motor might not be according to the predicted acceleration, the switching could create problems, if the actual position differed from the predicted position too much. For this reason a protection was implemented. If the estimated position differs too much from the predicted angle at the moment of switching, the application stops the motor and retries to start the motor again with realigning the rotor. This protection position difference is set to 55 electrical degrees in this application. The startup is retried five times, and if the motor startup is not successful within five times, the startup fail fault is generated.

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 controller and the application, then enters an infinite background loop. The background loop contains an application state machine.

5.4.1 Software States

5.4.1.1 Application Process States

The application process state is the highest level of the application state-machine. It has five application states: APP_FAULT, APP_READY, APP_TRANS_TO_RUN, APP_RUN, and APP_TRANS_TO_READY. Transition between the states is shown in [Figure 5-7](#) (black lines are regular transitions, red lines are transitions to fault).

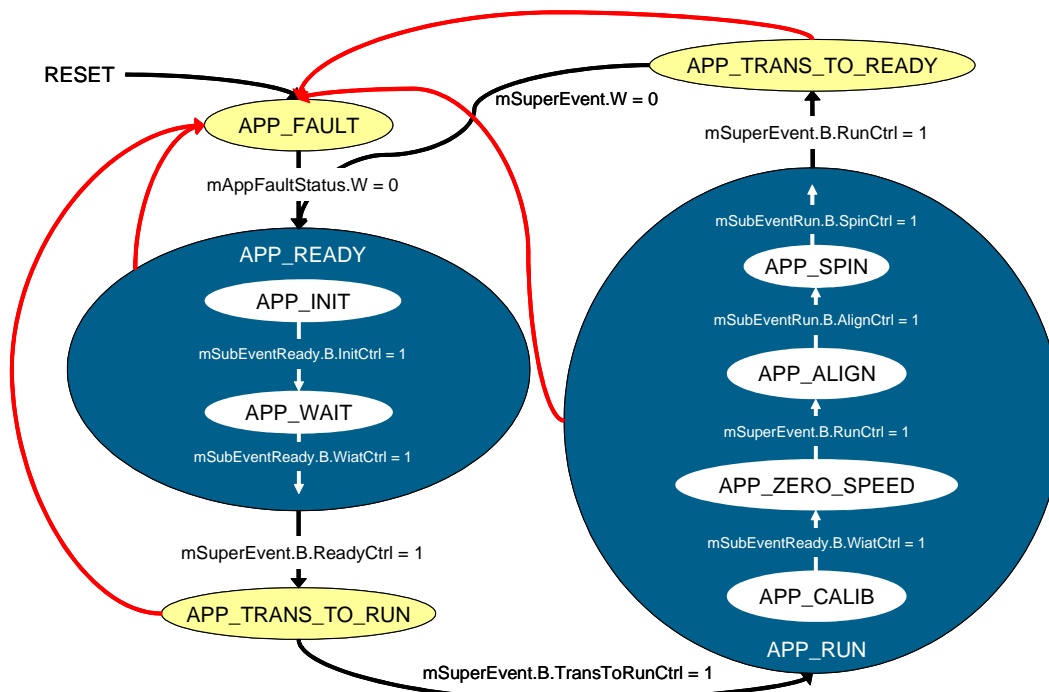


Figure 5-7. Application State Diagram — General Overview

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

The application process states with the sub-states are described in the following sections.

5.4.1.2 APP_FAULT

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

- Over-current
- Over-voltage
- Under-voltage
- Overload
- Startup fail

Next state (APP_READY) is set once all fault bits are cleared, that means no fault is detected (`mAppFaultStatus.W16 = 0`).

5.4.1.3 APP_READY

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

5.4.1.4 APP_TRANS_TO_RUN

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

5.4.1.5 APP_RUN

This most complex state consists of four sub-states. All necessary procedures for motor startup and rotation are processed in this state.

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

Next state is entered after *ReadyRunFlag* is cleared from FreeMASTER control page.

5.4.1.6 APP_TRANS_TO_READY

This transient state, between RUN and READY state, stops the motor, disables PWM pads, and clears state bits to provide the same application conditions like the state after RESET. Finally, READY state is entered.

5.4.1.7 Application Processes

Application processes are called from fast control loop IsrADC1 (125 μ s). They correspond to application states that control transitions between component processes. [Figure 5-8](#) shows a process state diagram.

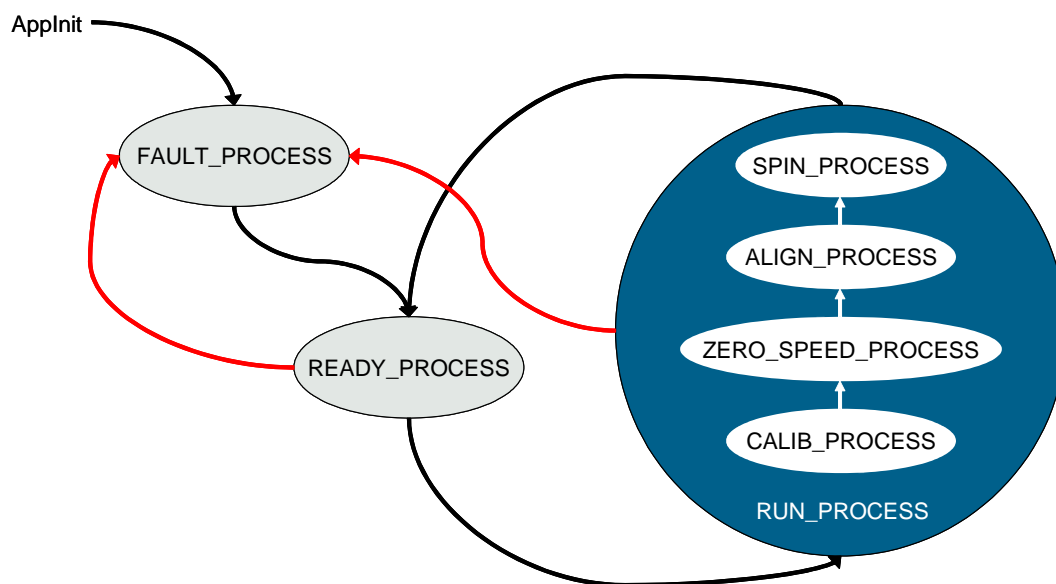


Figure 5-8. Process State Diagram

These processes perform routines and tasks required in each application state and sub-state such as ADC calibration, rotor alignment, and FOC calculations (rotor angle, DC-Bus ripple elimination, and so on) and transformations (Park, Clarke).

5.4.2 Initialization

Application 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 registers are initialized. This is the first function, which has to be called in the application main. The 3PP gate driver is configured via SPI as the next step and the FreeMASTER embedded driver is initialized according to settings in the *freemaster_cfg.h* configuration file.

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

- FreeMASTER scaling variables are defined.
- Analog-to-digital converter data structure is initialized using parameters from *PMSM_VC_app_setup.h* configuration file.
- *ADCInit* function is called.
- Speed controller and ramp parameters are set.
- Current controller parameters are set.
- Fifty-percent duty cycle is set and PWM pads are disabled.

5.4.2.1 FTM Configuration

To make the FTM generate PWM, it is necessary to set up these registers:

- FTM1 COMBINE
 - COMBINE bits 0, 1, 2 to 0 (combine mode disabled)
 - COMP bits 0, 1, 2 to 1 (complementary mode enabled)
 - SYNCEN bits 0, 1, 2 to 1 (synchronized reload enabled)
 - DTEN bits 0, 1, 2 to 1 (deadtime enabled)
- FTM1 POL
 - POL bits 0, 2, 4 to 1 (inverted polarity on top switches)
 - POL bits 1, 3, 5 to 0 (non-inverted polarity on bottom switches)
- FTM1 CxSC
 - MS bits 0, 1, 2, 3, 4, 5 = 0 (MSxB : MSxA = 0)
 - ELS bits 0, 1, 2, 3, 4, 5 = 1 (ELSxB : ELSxA = 1)
- FTM1 MODE
 - FTEN to 1 (FTM particular features enabled)
 - FAULTM = 2 (disables all channels with manual clearing)
 - FAULTIE to 1 (fault interrupt enabled)
- FTM1 OUTMASK = 0x3F (to disable PWM output)

- FTM1 MOD = 1500 (modulo 1500 to get 16 kHz from 48 MHz clock)
- FTM1 CNTIN = 0 (counter initialization to zero)
- FTM1 SC
 - SC = 0x8 (bus clock on FTM, divider 1)
 - CPWMS to 1
 - TOIE to 1 (overflow interrupt enabled)
- FTM1 DEADTIME
 - DTPS = 1 (deadtime clock divider 1)
 - DTVAL = 36 (deadtime 750 ns)
- FTM1 SYNC
 - CNTMIN to 0 (reload at low disabled)
 - CNTMAX to 1 (reload at high enabled)
- FTM1 ADCTRIG CH3TRIG to 1 (ADC trigger from the channel three)
- FTM1 FAULTFILTER FFVAL = 4 (fault filter = 4)

5.4.2.2 ADC Configuration

To make the ADC sample to be triggered from FTM1 it is necessary to set up these registers:

- SOPT2 ADHWT5 = 2 (ADC trigger source from FTM1)
- ADC1 SC1
 - AIEN to 1 (enable ADC interrupt)
 - ADCO to 0 (single conversion)
 - ADCH = 1 (channel one)
 - ADTRG to 1 (hardware trigger)
- ADC1 CFG
 - ADIV = 1 (input clock / 2)
 - MODE to = 1 (12-bit)
 - ADICLK = 0 (bus clock to input)
 - ADPC0 = 2 (ADC channel two)

5.4.3 Application Background Loop

This endless application background loop executes a simple application state-machine, which is shown in [Figure 5-7](#). Other process functions called from the loop are:

- FreeMASTER polling function *FMSTR_Poll()*
- Background part of the fault detection with *AppFaultDetection()*
- COP (watchdog) periodical servicing

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

5.4.4 Interrupts

There are several interrupt service routines executing the major motor control tasks. One is generated at the PWM reload, in other words when FTM1 overflows. Another is generated, when the ADC measures a sample, and then this routine calls the fast and slow control loops. Another interrupt is more or less asynchronous and it is the FTM1 fault.

The interrupt service routines and control tasks executed by each interrupt are described in the following subsections.

5.4.4.1 ADC1 Interrupt

The function `IsrADC1()` is assigned to this interrupt event. It is executed three times within every second PWM cycle. It is synchronized to FTM1, in such a way that the samples are taken in the middle of the PWM on-pulse. A more detailed description of the ADC1 interrupt timing can be found in [5.3.1, “ADC Conversion Timing and PWM Reload Interrupts.”](#) When the first ADC sample is measured, the ADC is switched to software mode and the next samples are triggered manually. When the last sample is measured, the function calls the FOC calculation and updates the PWM.

Tasks performed by the `IsrADC1()` function:

- It reads the measured value, subtracts the offset, and stores into the ADC result buffer.
- It increments the samples counter.
- It tests, whether the third conversion is finished.
- If not, it reconfigures and starts another ADC sample.
- If the last conversion — the rotor position is estimated, the FOC process function is called:
 - It stores reconstructed phase current values in the `mudtCurrentAbc` data structure.
 - It reads the sampled DC-Bus voltage from the ADC result buffer, and executes the digital filter on that samples.
 - It performs a transformation of the stator phase-currents from the 3-phase stationary reference frame into the 2-phase stationary reference frame (Forward Clarke Transformation).
 - It calculates the BEMF observer using the 2-phase stationary reference frame current and voltage, and estimated speed from the previous step.
 - It calculates the Angle-Tracking Observer using the BEMF observer's output with sine and cosine of the angle to get the estimated position and speed.
 - It performs a transformation of the stator phase-currents from the stationary into the rotational reference frame (Forward Park Transformation).
 - It executes the PI controllers of the d and q-axis components of the stator current.
 - It evaluates the d and q-components of the output voltage vector, by the summing controller outputs with the decoupling components of the stator voltage
 - It performs a transformation of the stator voltage space-vector d, q-components from the rotational reference frame into the α, β stationary reference frame.
 - It performs the DC-Bus ripple elimination algorithm on the output voltage.
 - It performs Space Vector Modulation on output voltage.

- It programs the PWM value registers and sets the LDOK bit.
- FreeMASTER recorder function is called, the speed loop divider is incremented, and when it reaches a certain value, the speed loop is calculated:
 - Calculate actual speed value.
 - Perform ramping of the motor speed.
 - Execute the motor-speed PI controller. Output of the speed controller sets the required torque-producing component of the stator current (q-axis).

5.4.5 FTM1 Overflow Interrupt

The **IsrFTM1Overflow()** is assigned to this interrupt event. It is executed every time the FTM1 counter overflows and starts counting down. At this instant this interrupt is generated. This function has a simple counter that is incremented and if it is two, it is reinitialized, and the ADC1 hardware trigger is turned on. The sense of this function is to enable the ADC trigger every second PWM period after reaching the maximum counter value.

Tasks performed by the **IsrFTM1Overflow()** function:

- It turns on the hardware trigger for ADC every second interrupt.
- It services the FTM1 Overflow flag.

5.4.5.1 FTM Fault Interrupt

The function **IsrFTM1Fault()** is assigned to this interrupt event. The interrupt is executed on a fault event. When a DC-Bus over-current is detected, an external comparator sets a signal on the fault pin to a high level. The FTM module sets all the PWM signals to the off state through wired logic. An interrupt request is generated.

Tasks performed by the **IsrFTM1Fault()** function:

- It disables the PWM output pads.
- It turns the motor off.
- It sets the application *OverDCBusVoltageFlag* or *OverDCBusCurrentFlag* flags.
- It services the corresponding interrupt request flag.

5.4.6 PI Controller Parameters

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

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

5.5 FreeMASTER Software

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

5.5.1 FreeMASTER Serial Communication Driver

The presented application includes the FreeMASTER Serial Communication Driver. The FreeMASTER Serial Communication Driver fully replaces the former PC Master driver. The new FreeMASTER driver remains fully compatible with the communication interface provided by the old PC Master drivers. It brings, however, many useful enhancements and optimizations.

The main advantage of the new driver is a unification across all supported Freescale processor products, as well as several new features that were added. One of the key features implemented in the new driver is Target-Side Addressing (TSA), which enables an embedded application to describe the memory objects it grants the host access to. By enabling the so-called “TSA-Safety” option, the application memory can be protected from illegal or invalid memory accesses.

To include the new FreeMASTER Serial Communication Driver in the application, the user has to manually include the driver files in the CodeWarrior project. For the presented application, the driver has already been included.

The FreeMASTER driver files are located in the following folders:

- *{Project}\Sources\freemaster\src_platforms\MCF51xx*, contains platform-dependent driver C-source and header files, including a master header file *freemaster.h*.
- *{Project}\Sources\freemaster\src_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 9600 Bd.

NOTE

Higher communication speeds than 9600 Bd might not be supported by the SCI/USB converters.

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

The driver is configured using the *freemaster_cfg.h* header file. The user has to modify this file to configure the FreeMASTER driver. The FreeMASTER driver C-source files include the configuration file, and use the macros defined there for conditional and parameter compilation.

A detailed description of the FreeMASTER Serial Communication Driver is provided in 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:

```
/* Freemaster recorder */  
FMSTR_Recorder();
```

In this application, the FreeMASTER recorder is called from the ADC interrupt, which creates a 125 μ s time-base for the recorder function. A detailed description of the FreeMASTER 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 PMSM vector control application. Start the FreeMASTER software window's project by clicking on the *PMSM_VC_Speed_CL_Sensorless_MCF51AC256_00315.pmp* file. [Figure 5-9](#) illustrates the FreeMASTER software control window after this project has been launched. To switch to the control page, click on the 'control page' tab.

A user is able to monitor all the important quantities of the motor. By clicking on the Speed gauge, the motor is started and the desired speed is set. The actual Motor Speed, Motor Currents, and Voltages are displayed on the control page gauges.

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

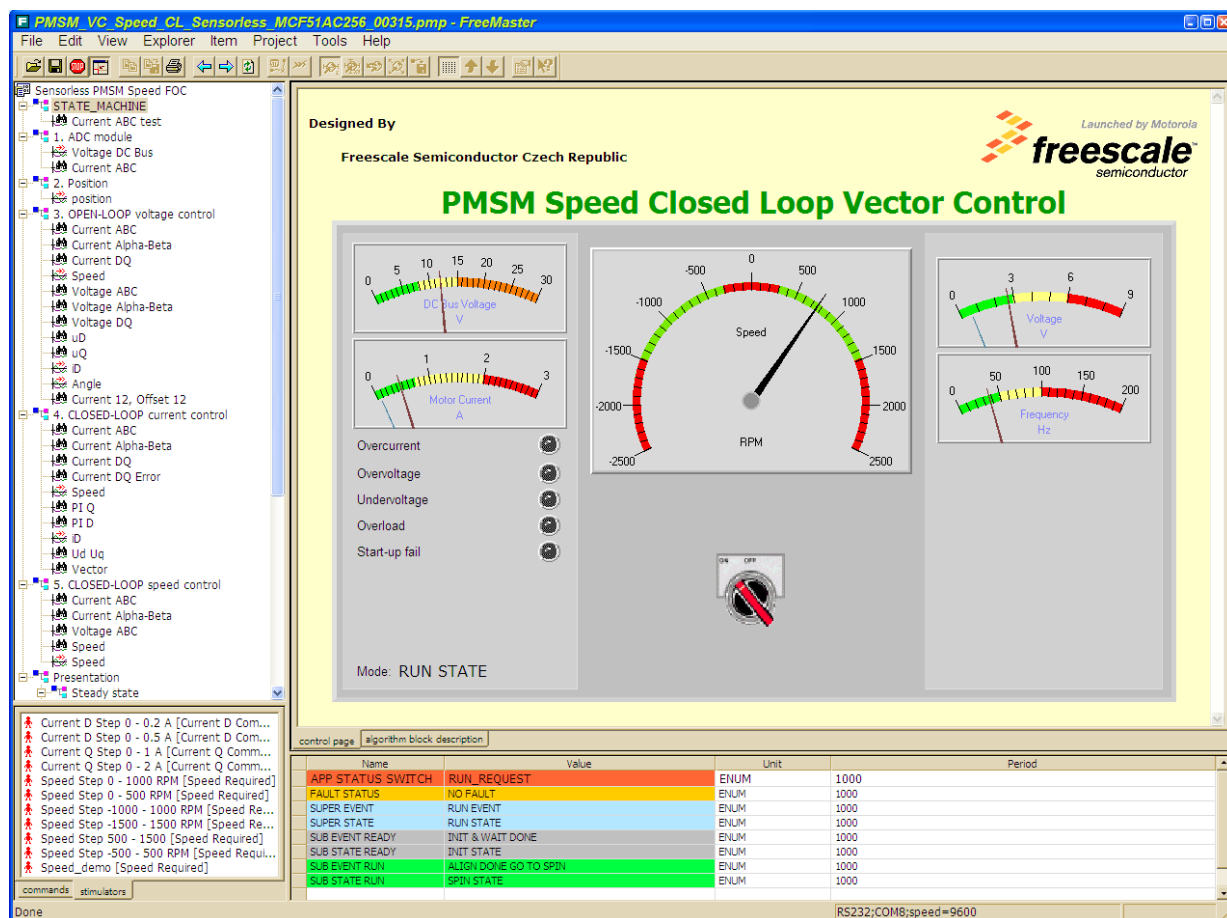


Figure 5-9. FreeMASTER Control Screen

The FreeMASTER software control page actions that are supported:

- Setting the Required Speed of the motor.

The FreeMASTER software control page displays:

- Speed
- DC-Bus voltage
- d, q axis currents and voltages
- Sine frequency
- Application fault status

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

- Demonstration Measurements.
- Control Regulators — for adjusting PI controller parameters.

Chapter 6

Application Setup

As described earlier, the PM synchronous motor vector control application is targeted at the MCF51AC256 device. The concept of the PM synchronous motor vector control drive incorporates the following hardware components:

- MCF51AC256 Daughter Board
- 3-Phase Motor Control Drive
- 3-phase PM synchronous motor (default configuration for motor TG Drives TGT2-0032-30-24/T0PS1KX)



Figure 6-1. Demo Application Setup

6.1 3-Phase Motor Control Drive Using MCF51AC256 Setup

The MCF51AC256 Daughter Board has to be plugged into the socket on the 3-Phase Motor Control Drive. Now the demo application code has to be programmed into the Flash memory. To do so, follow these steps:

1. Connect a +12 V power supply to the J3 power connector on the 3-Phase Motor Control Drive board.
2. Connect the BDM's USB cable to the host PC and to the J1 header on the MCF51AC256 Daughter Board.
3. Compile your project and program it into the device.

4. Disconnect the +12 V power supply from the J3 power connector on the 3-Phase Motor Control Drive board.
5. Unplug the BDM connector from the J1 header on the MCF51AC256 Daughter Board.

6.2 Demo Hardware Setup

The complete application setup is shown in [Figure 6-1](#) and [Figure 6-2](#). To build the demo application setup, follow these steps:

1. Connect a USB cable to an open USB port on the host PC and to the J10 connector on the 3-Phase Motor Control Drive board, for FreeMASTER remote control.
2. Connect the motor phases to the connector J1 on the 3-Phase Motor Control Drive board.
3. Connect a 12 V power supply to the connector J3 on the 3-Phase Motor Control Drive board.
4. Now it's necessary to select the virtual COM port number that was attached to the USB connection. Therefore, in Microsoft Windows, go to Start/Control Panel and click on the item System. In the System Properties dialog click on the Hardware tab and then on the Device Manager button. Expand the node of Ports (COM & LPT) controllers. Open FreeMASTER and push the STOP icon to stop the communication. Then go to the Project menu and click Options. In the Comm tab chose the option Direct RS232 connection and select the Port number that you have found in the Device Manager. Set Speed to 9600. Click OK and now uncheck the STOP icon to start the communication. FreeMASTER should now communicate with the application.

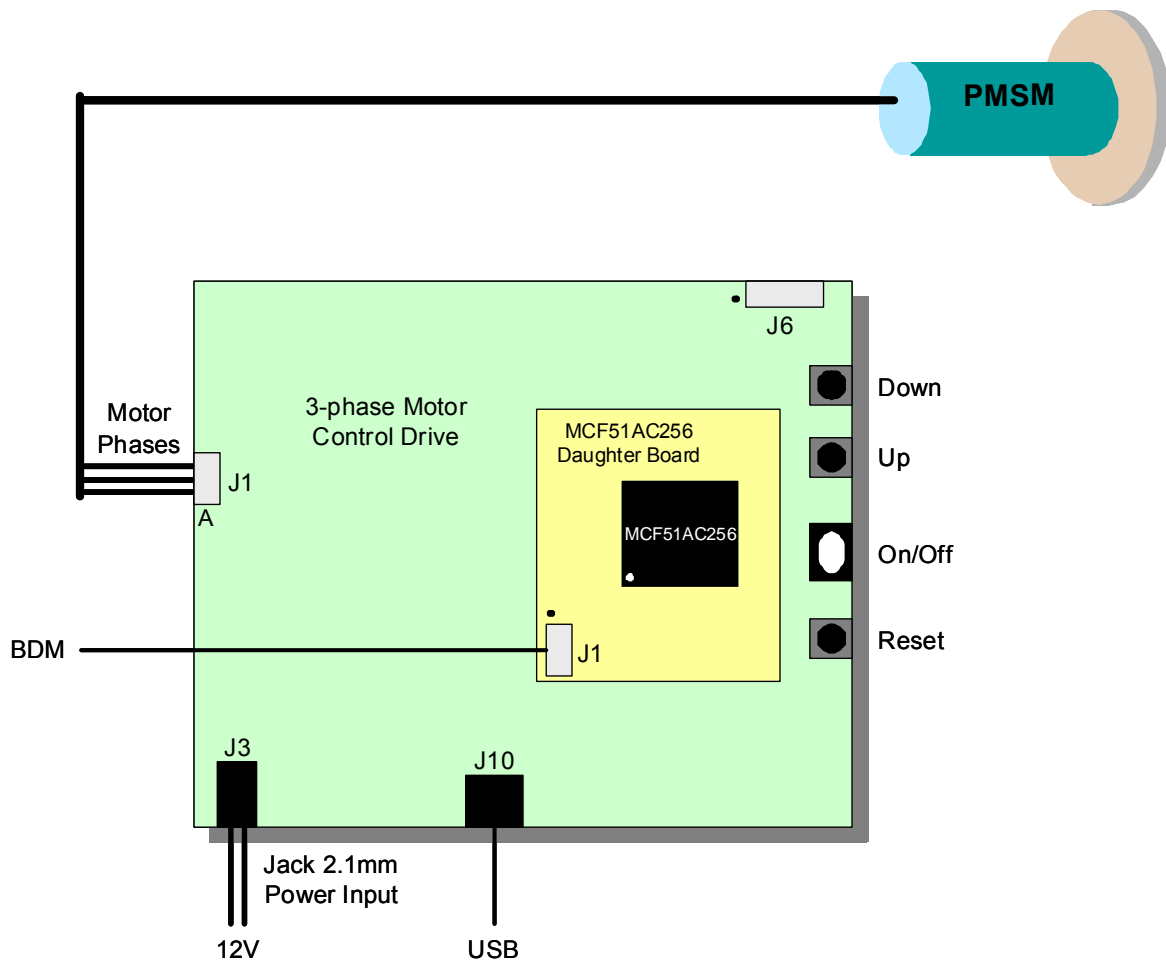


Figure 6-2. Demo Application Connection Overview



Chapter 7

Results and Measurements

This section describes measurements of the PMSM vector control application.

7.1 System and Measurement Conditions

7.1.1 Hardware Setup

The application measurements were provided using TGT2-0032-30-24 motor and the default hardware setup described in [Chapter 6, “Application Setup.”](#)

7.1.2 Software Setup

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

The measurements were provided using default code parameter settings, if not mentioned otherwise. The pre-processor constants *CLOSED_LOOP_CURRENT_CONTROL_ON* and *CLOSED_LOOP_SPEED_CONTROL_ON* are defined in *main.c* file. These enable compilation of the code for measurements.

7.1.3 FreeMASTER

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

The *PMSM_VC_Speed_CL_Sensorless_MCF51AC256_00315.pmp* file is used by the FreeMASTER software, which needs to be installed on the PC. The pmp file is included within the software structure. This incorporates the definition of the recorders, used for the measurements.

7.2 Measured Results

7.2.1 3-Phase Current Reconstruction

This section presents the measurements of 3-phase motor current samples, see [Figure 7-1](#).

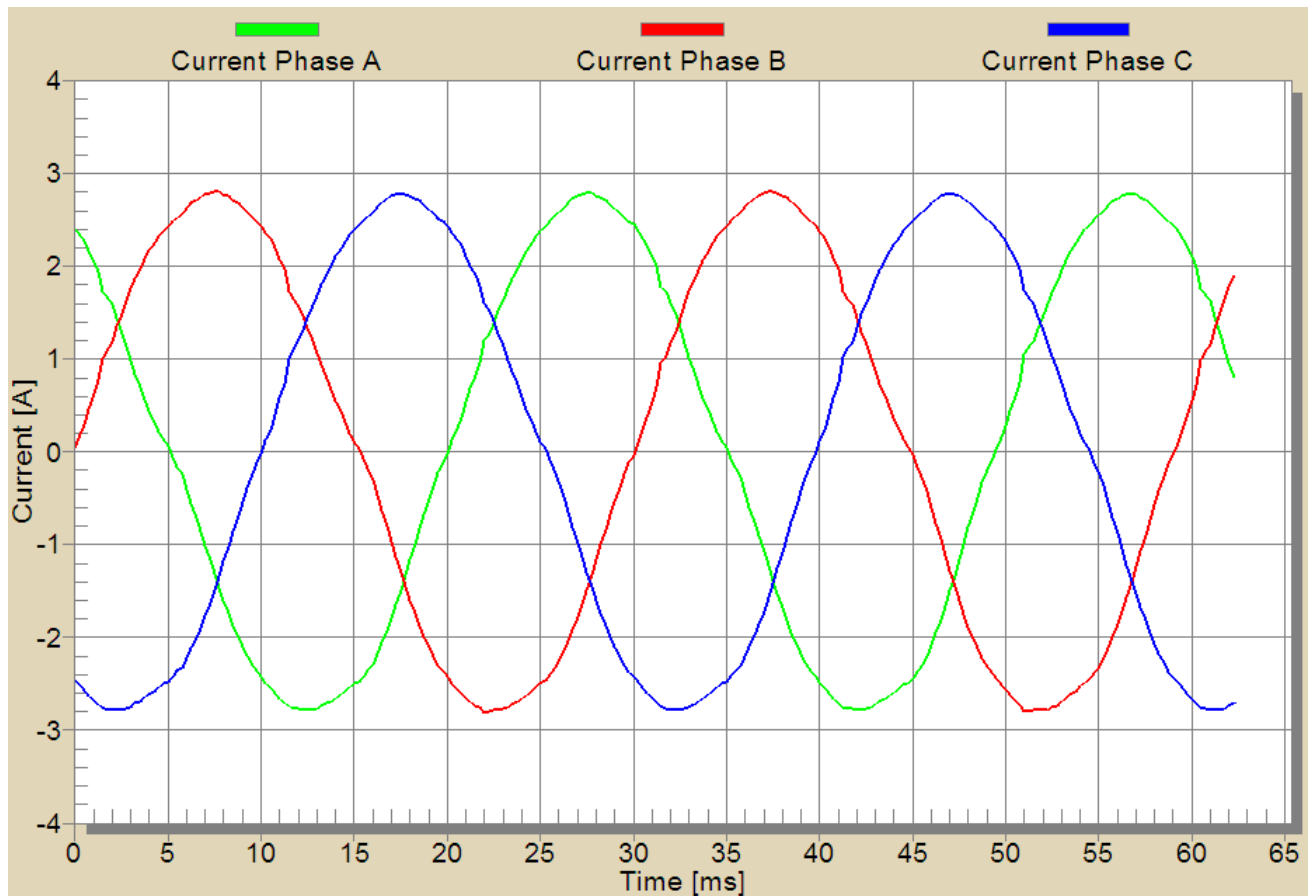


Figure 7-1. Motor Phase Currents

7.2.2 Speed Controller

The speed PI controller is calculated in the outer loop. Its required value is the set ramped required speed, and the actual value is the measured speed. The output of the controller is the required value of Q current for the current-control loop PI controller (`mudtCurrentDQCommand.f16Q`);, see [Figure 7-2](#).

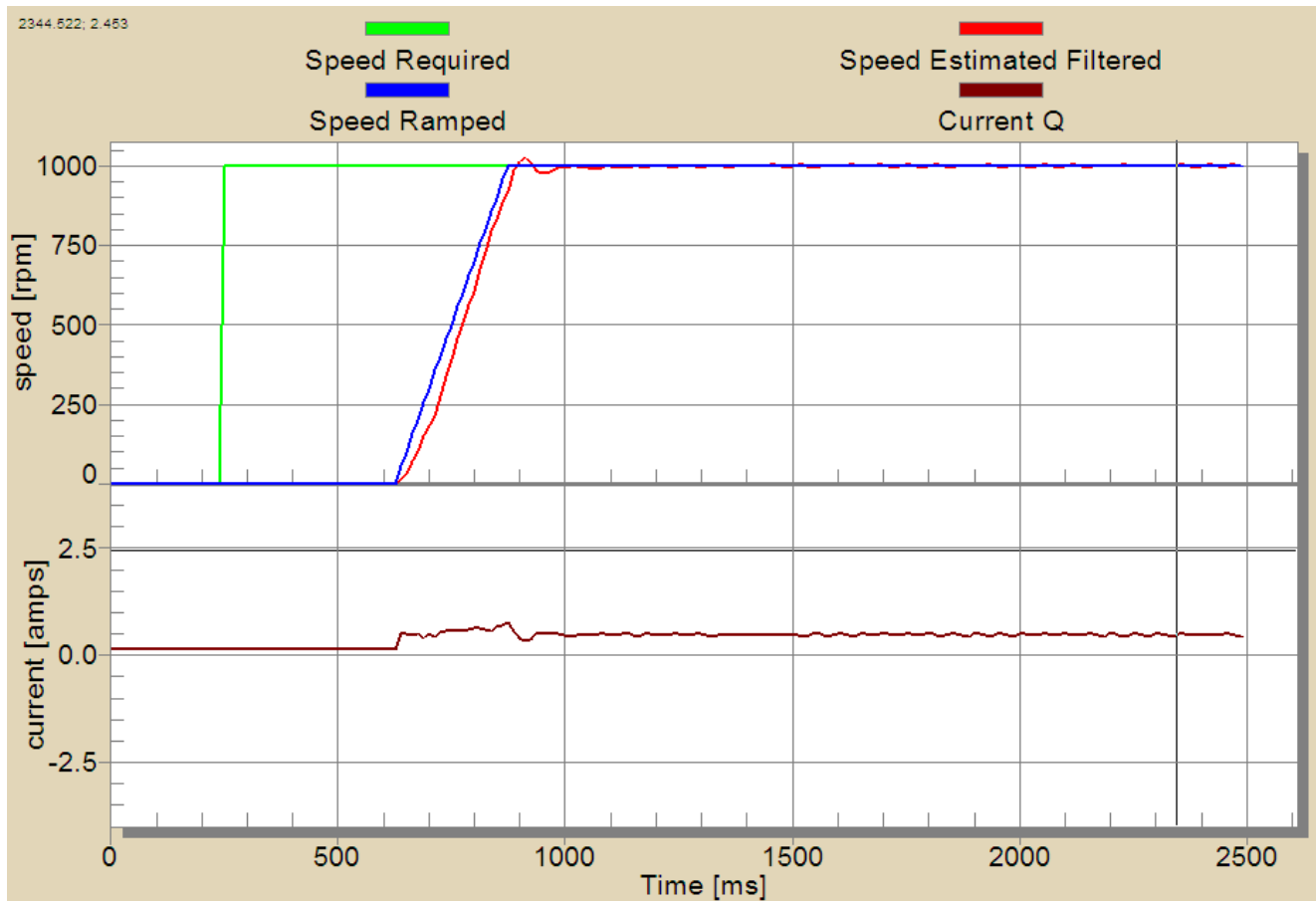


Figure 7-2. Speed Controller

7.2.3 Current Controller

The inner control loop processes d-axis and q-axis PI controllers. [Figure 7-3](#) shows step response of d-axis controller (q-axis is similar). Obtained by a transformation from motor-phase currents, d-axis command is set as a required value, and d-axis feedback as the actual value. The output of the controller is the d-axis voltage value, used for following transformation to a 3-phase system.

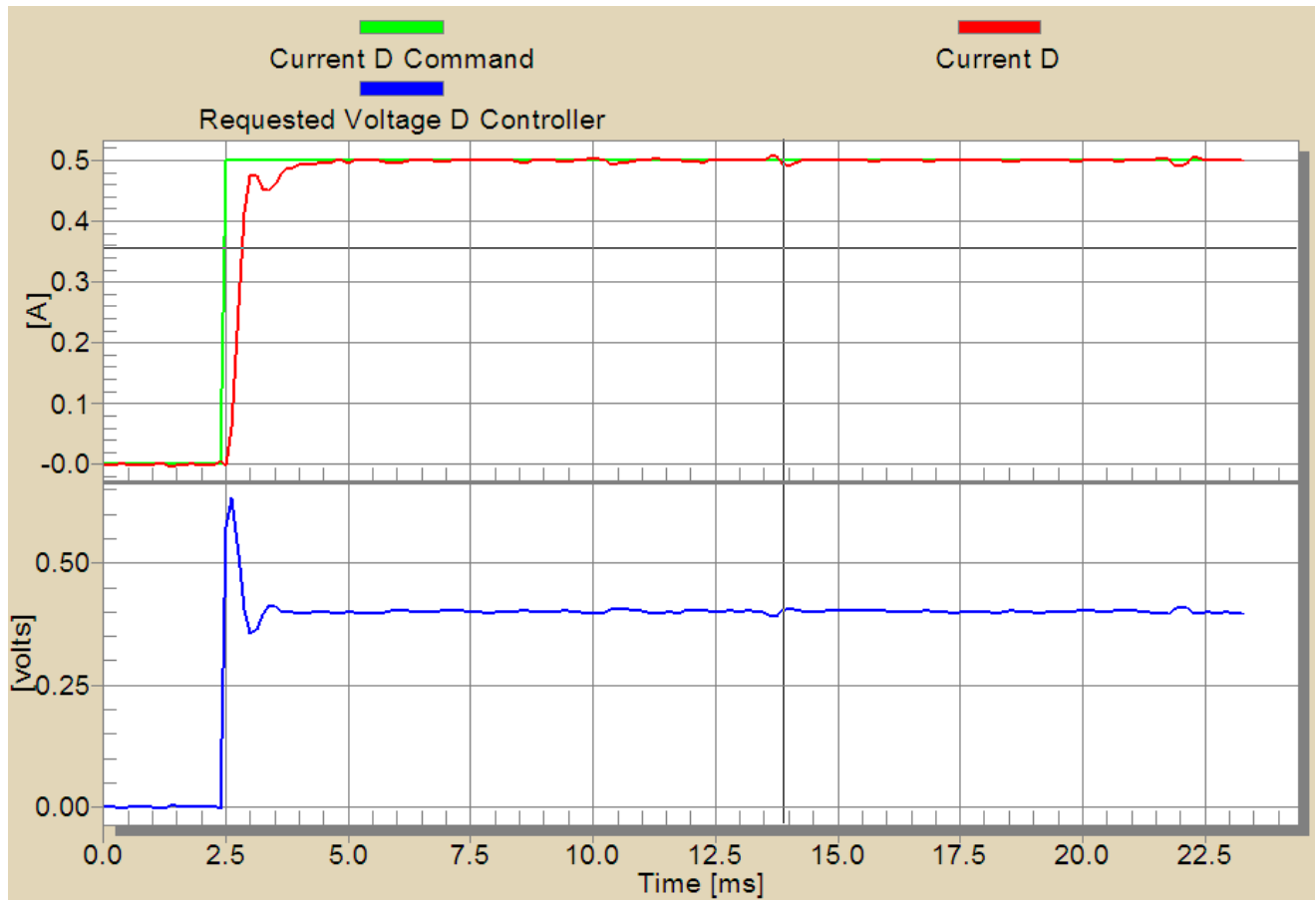


Figure 7-3. Current Controller

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
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

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2009. All rights reserved.