3-Phase BLDC Motor Sensorless Control using MC56F8013

Designer Reference Manual

56F801x
Hybrid Controller

DRM070
Rev. 0
5/2005

freescale.com

freescale™
semiconductor

3-Phase BLDC Motor Sensorless Control using MC56F8013

Designer Reference Manual

by: Ozan Sadi Sihmanoglu
Freescale Czech Systems Laboratories
Roznov pod Radhostem, Czech Republic

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

Contents

Chapter 6
Application Setup

Chapter 7
Tuning

Appendix A.
References

Appendix B.
Glossary

Chapter 1
Introduction

1.1 Introduction

This reference design describes the design of a 3-phase brushless direct current (BLDC) drive based on Freescale's DSP56F8013 dedicated motor control device.

BLDC motors are very popular in many application areas. The BLDC motor does not have a commutator and is, consequently, more reliable than the DC motor. The BLDC motor has other advantages over the AC induction motor. Because BLDC motors achieve higher efficiency by generating the rotor magnetic flux with rotor magnets, they are used in high-end white goods (such as refrigerators, washing machines, and dishwashers), high-end pumps, fans, and other appliances that require high reliability and efficiency.

The concept of the current application is of a closed-loop speed-controlled, trapezoidal BLDC drive, using a back-EMF sensing control technique. It provides an example of BLDC motor control design using a Freescale digital signal processor (DSP).

This reference design includes basic motor theory, system design concept, hardware implementation, software design, and tuning considerations, and includes the FreeMaster software visualization tool.

1.1.1 Pump, Fan and Compressor Applications

BLDC motors are widely used in pump, fan and compressor applications because of their robust structure. A common feature of these applications is that they do not require position information; only speed information is required, and only to perform control. BLDC motors can be used without complex control algorithms. This reference manual presents an ideal solution using the BLDC motor, for such applications.

1.1.2 Why Sensorless Control?

In the BLDC motor, the rotor position must be known in order to energize the phase pair and to control the phase voltage. If sensors are used to detect rotor position, then sensed information must be transferred to a control unit (see Figure 1-1).

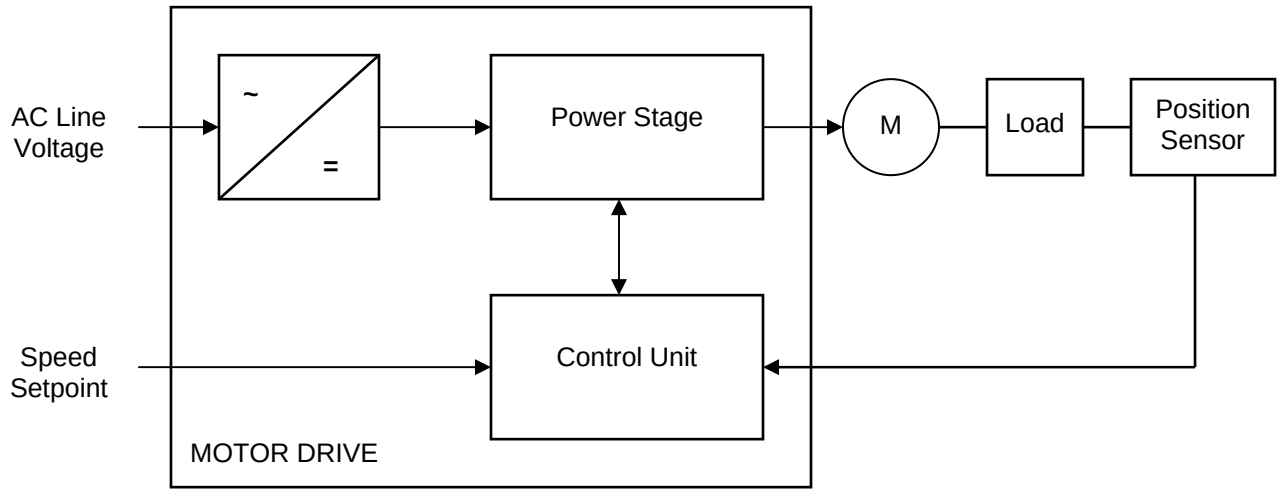


Figure 1-1. Classical System

This requires additional connections to the motor, which may not be acceptable in some applications. It may not be physically possible to make the required connections to the position sensors. Also, the additional cost of the position sensors and the wiring may be unacceptable. The physical connection problem could be solved by incorporating the driver in the motor body; however, a significant number of applications do require a sensorless solution.

1.2 Freescale Hybrid Controller Advantages and Features

The Freescale MC56F801x family is well suited to digital motor control, combining the DSP's calculation capability with the MCU's controller features on a single chip. These hybrid controllers offer many dedicated peripherals, such as pulse width modulation (PWM) module(s), an analog-to-digital converter (ADC), timers, communication peripherals (SCI, SPI, I2C), on-board Flash and RAM.

The MC56F801x family members provides the following peripheral blocks:

- One pulse width modulator module (although limited pinout on MC56F8014 and MC56F8011) with PWM outputs, fault inputs, fault tolerant design with dead-time insertion; supports both center-aligned and edge-aligned modes
- 12-bit ADCs, which support two simultaneous conversions; ADC and PWM modules can be synchronized
- One dedicated 16-bit general purpose quad timer module
- One serial peripheral interface (SPI)
- One serial communications interface (SCI) with LIN slave functionality
- One inter-integrated circuit (I2C) port
- On-board 3.3 V to 2.5 V voltage regulator for powering internal logic and memories
- Integrated power-on-reset and low voltage interrupt module
- All pins multiplexed with GPIO (general purpose input/output)
- Computer operating properly (COP) watchdog timer
- External reset input pin for hardware reset
- JTAG/On-Chip Emulation (OnCE™) module for unobtrusive, processor-speed-independent debugging

- Phase locked loop based frequency synthesizer for the hybrid controller core clock, with on-chip relaxation oscillator

Table 1-1. Memory Configuration for the MC56F801x family

	MC56F8011	MC56F8012	MC56F8013	MC56F8014
Program Flash	12K x 16-bit	12K x 16-bit	16K x 16-bit	16K x 16-bit
Program RAM	2K x 16-bit	2K x 16-bit	4K x 16-bit	4K x 16-bit
Data RAM	2K x 16-bit	2K x 16-bit	4K x 16-bit	4K x 16-bit

BLDC motor control benefits greatly from the flexible PWM module, fast ADC, and quadrature timer module.

The PWM offers flexibility in its configuration, enabling efficient control of the BLDC motor. The PWM block has the following features:

- Three complementary PWM signal pairs, six independent PWM signals, or a mixture thereof
- Complementary channel operation features
- Independent top and bottom deadtime insertion (MC56F8013 and later devices)
- Separate top and bottom pulse width correction via current status inputs or software
- Separate top and bottom polarity control
- Edge-aligned or center-aligned PWM reference signals
- 15-bit resolution
- Half-cycle reload capability
- Integral reload rates from one to sixteen period
- Mask/swap capability
- Individual, software-controlled PWM output
- Programmable fault protection
- Polarity control
- 10/16mA current sink capability on PWM pins
- Write-protectable registers

The PWM module is capable of providing six PWM signals with bipolar switching (the diagonal power switches are driven by the same signal). In addition, the PWM provides six-step BLDC commutation control (where one motor phase is left unpowered so that the back-EMF can be detected). The PWM duty cycle can be set asynchronously to the commutation of the motor phases event using the channel swap feature.

The ADC module has the following features:

- 12-bit resolution
- Dual ADCs per module; three input channels per ADC
- Maximum ADC clock frequency is 5.33 MHz with 187 ns period
- Sampling rate up to 1.78 million samples per second
- Single conversion time of 8.5 ADC clock cycles (8.5 x 187 ns = 1.59 ms)
- Additional conversion time of six ADC clock cycles (6 x 187 ns = 1.125 ms)
- Eight conversions in 26.5 ADC clock cycles (26.5 x 187 ns = 4.97 ms) using parallel mode
- Can be synchronized to the PWM via the SYNC input signal provided the integration permits the PWM to trigger a timer channel connected to that input

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

The ADC is used to measure DC-bus voltage, DC-bus current, and back-EMF phase voltages. The ADC's high/low level detection capability provides automatic detection of the DC over-voltage and DC under-voltage, serviced in the associated interrupt service routine (ISR).

The application uses the ADC block in simultaneous mode and sequential scan. It is synchronized with PWM pulses. This configuration allows the simultaneous conversion, within the required time, of required analog values of all phase currents, voltage and temperature.

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

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

The application uses four channels of the quadrature timer for PWM to ADC synchronization and commutation timing. Another channel of the quadrature timer module is set to generate a timebase for a speed and alignment controller.

Chapter 2
Control Theory

2.1 The Brushless DC Motor

The BLDC 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. It is also referred to as an electronically commuted motor. There are no brushes on the rotor, and the commutation is performed electronically at certain rotor positions. The stator is usually made from magnetic steel sheets. A typical cross-section of a BLDC motor is shown in Figure 2-1. The stator phase windings are inserted in the slots (distributed winding) or can be wound as one coil on the magnetic pole. Because the air gap magnetic field is produced by permanent magnets, the rotor magnetic field is constant. The magnetization of the permanent magnets and their displacement on the rotor is chosen so that the shape of the back-EMF (the voltage induced in the stator winding due to rotor movement) is trapezoidal. This allows a DC voltage, with a rectangular shape (see Figure 2-2), to be used to create a rotational field with low torque ripples.

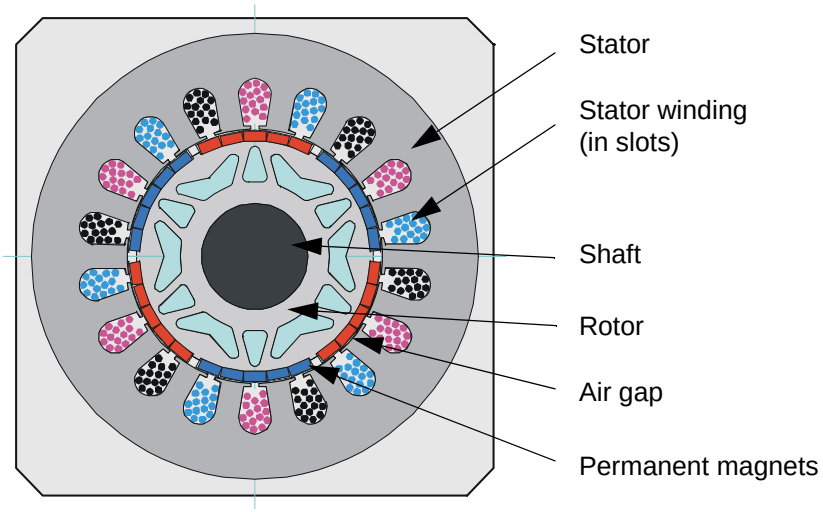


Figure 2-1. BLDC Motor Cross-section

The motor can have more then one pole-pair per phase. The number of pole-pairs per phase defines the ratio between the electrical revolution and the mechanical revolution. For example, the BLDC motor shown has three pole-pairs per phase; which represents three electrical revolutions per mechanical revolution.

The rectangular, easy to create, shape of the applied voltage makes controlling and driving the motor simple. However, the rotor position must be known at certain angles, to be able to align the applied voltage with the back-EMF. Alignment of the back-EMF with commutation events is very important; when this is achieved, the motor behaves as a DC motor and runs at maximum efficiency. Thus, simplicity of control and performance makes the BLDC motor the best choice for low-cost and high-efficiency applications.

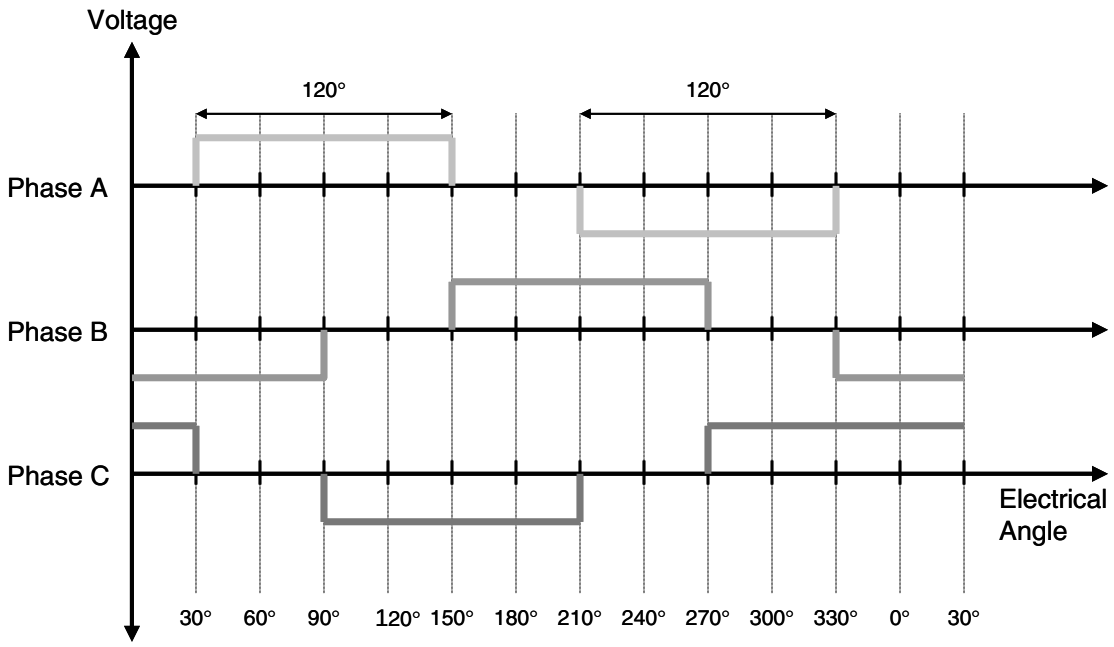


Figure 2-2. Three Phase Voltage System for a BLDC Motor

Figure 2-3 shows the number of waveforms, the magnetic flux linkage, the phase back-EMF voltage, and the phase-to-phase back-EMF voltage. The magnetic flux linkage was measured by calculating the integration phase back-EMF voltage, which was measured on the non-fed motor terminals of the BLDC motor. As can be seen, the shape of the back-EMF is approximately trapezoidal and the amplitude is a function of the actual speed. During speed reversal, the amplitude, sign and phase sequence are all changed.

The filled areas in the tops of the phase back-EMF voltage waveforms indicate the intervals where the particular phase power stage commutations are conducted. As can be seen, the power switches are cyclically commutated through the six steps. The crossing points of the phase back-EMF voltages represent the natural commutation points. During normal operation, commutation is performed here. Some control techniques lead the commutation by a defined angle to control the drive above the PWM voltage control.

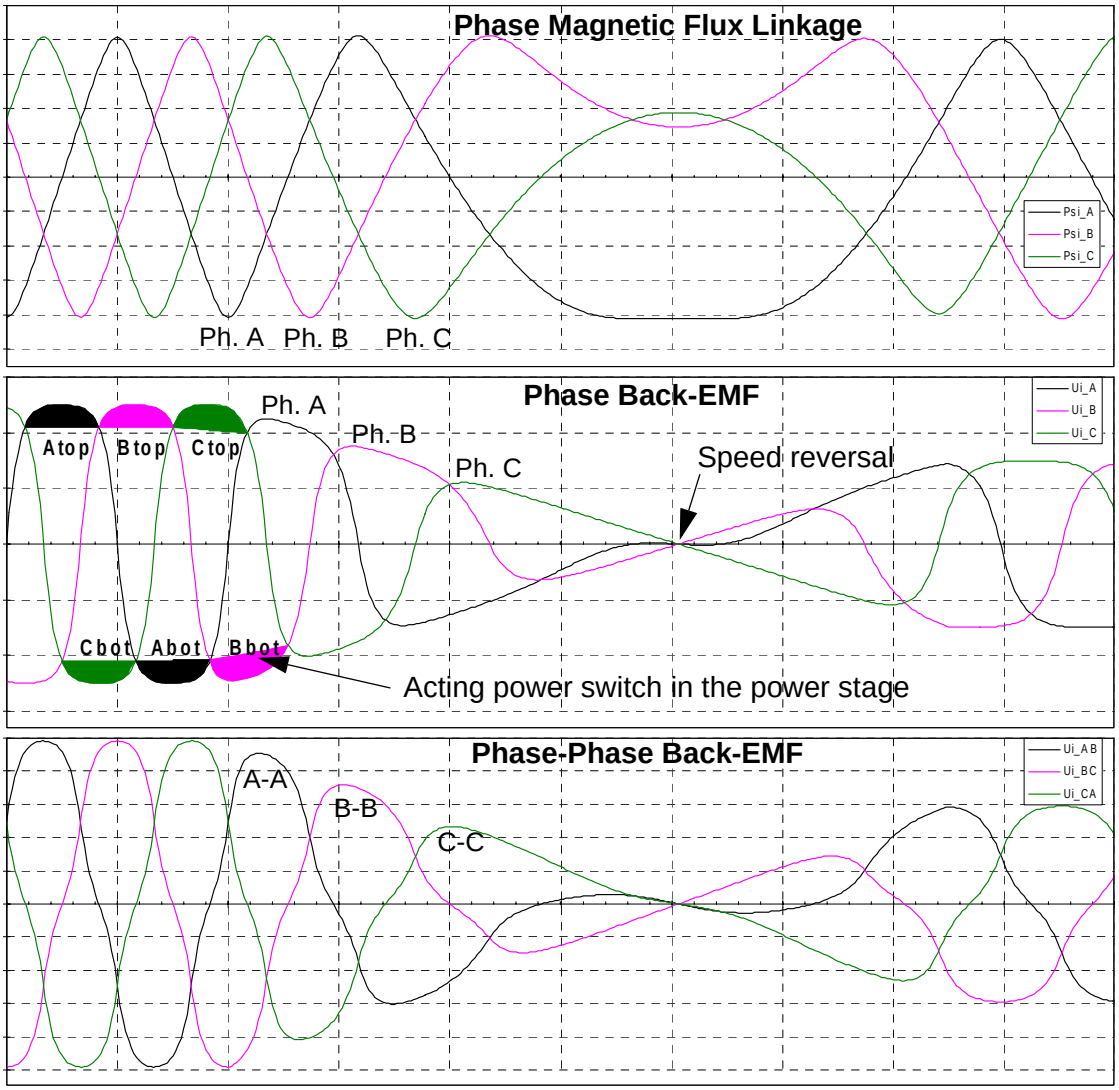


Figure 2-3. BLDC Motor/Back-EMF and Magnetic Flux

2.2 Power Stage

The voltage for the 3-phase BLDC motor is supplied from a typical 3-phase power stage. The 3-phase power stage is controlled by the hybrid controller's on-chip PWM module; which creates the desired switch control signals. A hybrid controller with a BLDC motor and a power stage is shown in Figure 2-4.

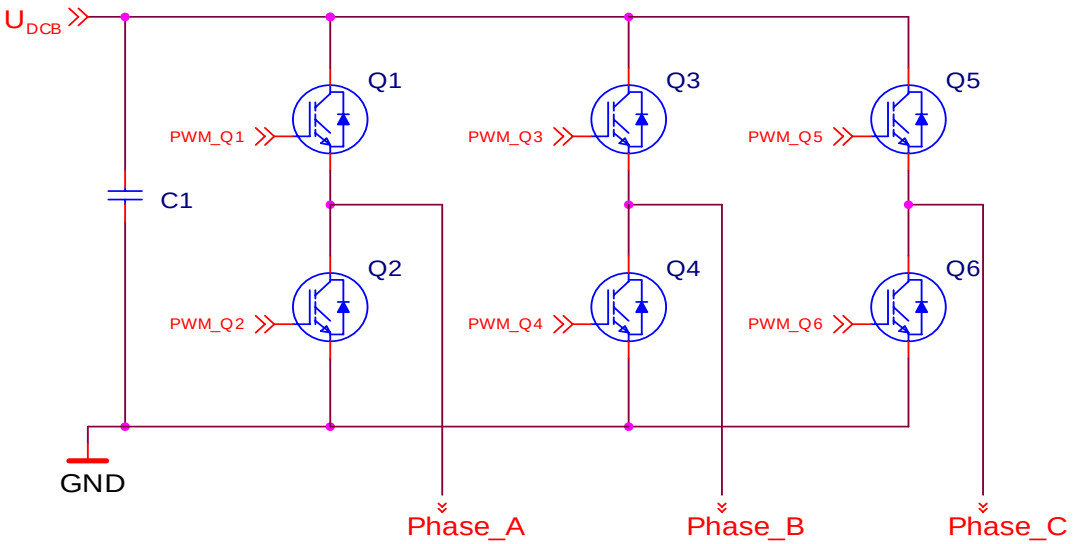


Figure 2-4. 3 Phase BLDC Motor Power Stage

2.3 Mathematical Description of the BLDC Motor

2.3.1 Power Stage — Motor System Model

To explain and simulate the idea of back-EMF sensing techniques, a simplified mathematical model, based on the basic circuit topology (see Figure 2-5), is provided.

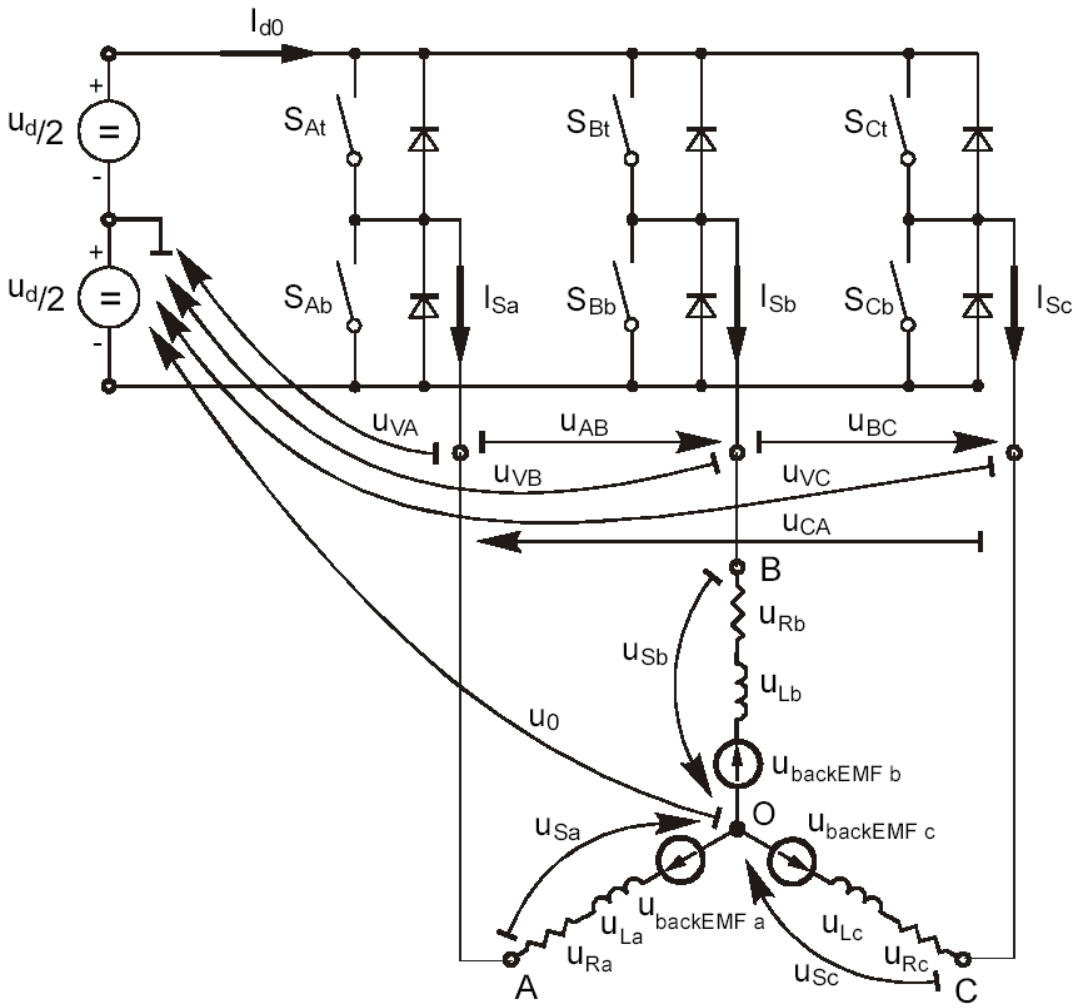


Figure 2-5. Power Stage and Motor Topology

The goal of the model is to find how the motor characteristics depend on the switching angle. The switching angle is the angular difference between a real switching event and an ideal one (at the point where the phase-to-phase back-EMF crosses zero).

The motor drive model consists of a 3-phase power stage and a BLDC motor. The power for the system is provided by a voltage source (U_d). Six semiconductor switches ($S_{A/B/C\ t/b}$), controlled elsewhere, allow the rectangular voltage waveforms (see Figure 2-2) to be applied. The semiconductor switches and diodes are simulated as ideal devices. The natural voltage level of the whole model is applied at one half of the DC-bus voltage. This simplifies the mathematical expressions.

2.3.2 Mathematical Model

The BLDC motor is usually very symmetrical. All phase resistances, phase and mutual inductances, and flux linkages can be thought of as equal to, or as a function of the position ∅ with a 120× displacement. The electrical BLDC motor model then consists of the set of stator voltage equations given by Equation 2-1.

$$\begin{bmatrix} u_{Sa} \\ u_{Sb} \\ u_{Sc} \end{bmatrix} = R_S \begin{bmatrix} i_{Sa} \\ i_{Sb} \\ i_{Sc} \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \psi_{Sa} \\ \psi_{Sb} \\ \psi_{Sc} \end{bmatrix} \tag{2-1}$$

The following set of equations is valid for the presented topology:

$$\begin{aligned} u_{Sa} &= u_{VA} - u_0 \\ u_{Sb} &= u_{VB} - u_0 \\ u_{Sc} &= u_{VC} - u_0 \end{aligned} \tag{2-2}$$

$$u_0 = \frac{1}{3}\left[\sum_{x=A}^{C} u_{Vx} - \sum_{x=a}^{c} u_{backEMFx}\right] \tag{2-3}$$

$$u_{Sa} = \frac{1}{3}\left[2u_{VA} - u_{VB} - u_{VC} + \sum_{x=a}^{c} u_{backEMFx}\right] \tag{2-4}$$

$$u_{Sb} = \frac{1}{3}\left[2u_{VB} - u_{VA} - u_{VC} + \sum_{x=a}^{c} u_{backEMFx}\right]$$

$$u_{Sc} = \frac{1}{3}\left[2u_{VC} - u_{VA} - u_{VB} + \sum_{x=a}^{c} u_{backEMFx}\right]$$

where;

$u_{VA} \cdots u_{VC}$	branch voltages between one power stage output and its natural zero.
$u_{Sa} \cdots u_{Sc}$	motor phase winding voltages.
$u_{backEMFa} \cdots u_{backEMFc}$	phase back-EMF induced in the stator winding.
u_O	differential voltage between the central point of the star connection of motor winding and the power stage natural zero
$i_{Sa} \cdots i_{Sc}$	phase currents
R_S	phase resistances

The equations Equation 2-4 can be rewritten taking into account the motor phase resistance and the inductance. The mutual inductance between the two motor phase windings can be neglected because it is very small and has no significant effect for our abstraction level.

$$\begin{aligned} u_{VA} - u_{backEMFa} - \frac{1}{3}\left[\sum_{x=A}^{C} u_{Vx} - \sum_{x=a}^{c} u_{backEMFx}\right] &= R_s i_{Sa} + L_s \frac{di_{Sa}}{dt} \\ u_{VB} - u_{backEMFb} - \frac{1}{3}\left[\sum_{x=A}^{C} u_{Vx} - \sum_{x=a}^{c} u_{backEMFx}\right] &= R_s i_{Sb} + L_s \frac{di_{Sb}}{dt} \\ u_{VC} - u_{backEMFc} - \frac{1}{3}\left[\sum_{x=A}^{C} u_{Vx} - \sum_{x=a}^{c} u_{backEMFx}\right] &= R_s i_{Sc} + L_s \frac{di_{Sc}}{dt} \end{aligned} \quad (2\text{-}5)$$

where;

R_s, L_s phase resistances and inductances

The internal torque of the motor itself is defined as:

$$T_i = \frac{1}{\omega}\sum_{x=a}^{c} u_{backEMFx} \cdot i_{Sx} = \sum_{x=a}^{c} \frac{d\psi_{Sx}}{d\theta} \cdot i_{Sx} \quad (2\text{-}6)$$

where;

T_i internal motor torque (no mechanical losses)

ω, θ rotor speed and rotor position

ψ_{Sx} magnetic flux of phase winding x

It is important to understand how the back-EMF can be sensed and how the motor behavior depends on the alignment of the Back-EMF to commutation events. This is explained in the next sections.

2.3.3 Back-EMF Sensing

The back-EMF sensing technique is based on the fact that only two phases of a BLDC motor are energized at a time (see Figure 2-2). The third phase is a non-fed phase that can be used to sense the back-EMF voltage.

Consider the situation when phases A and B are powered and phase C is non-fed. No current passes through phase C. Assume the following conditions are met:

$$\begin{aligned} &S_{Ab}, S_{Bt} \leftarrow \text{PWMswitching} \\ &u_{VA} = \mp\frac{1}{2}u_d, u_{VB} = \pm\frac{1}{2}u_d \\ &i_{Sa} = -i_{Sb}, i_{Sc} = 0, di_{Sc} = 0 \\ &u_{backEMFa} + u_{backEMFb} + u_{backEMFc} = 0 \end{aligned} \quad (2\text{-}7)$$

The branch voltage u_{VC} can be calculated when considering the above conditions:

$$u_{VC} = \frac{3}{2}u_{backEMFC} \quad (2\text{-}8)$$

Figure 2-6 shows that the branch voltage of phase C, between the power stage output C and the natural voltage level, can be sensed. Thus the back-EMF voltage is obtained, and zero crossing can be recognized.

The general expression can be found by:

$$u_{Vx} = \frac{3}{2}u_{backEMFx} \tag{2-9}$$

where:

$$x = A, B, C$$

There are two conditions that must be met:

1. Top and bottom switch (in diagonal) must be driven with the same PWM signal.
2. No current must go through the non-fed phase used to sense the back-EMF.

Figure 2-6 shows branch and motor phase winding voltages during a 0–360°electrical interval. Shaded rectangles designate the validity of Equation 2-9. In other words, the back-EMF voltage can be sensed during designated intervals.

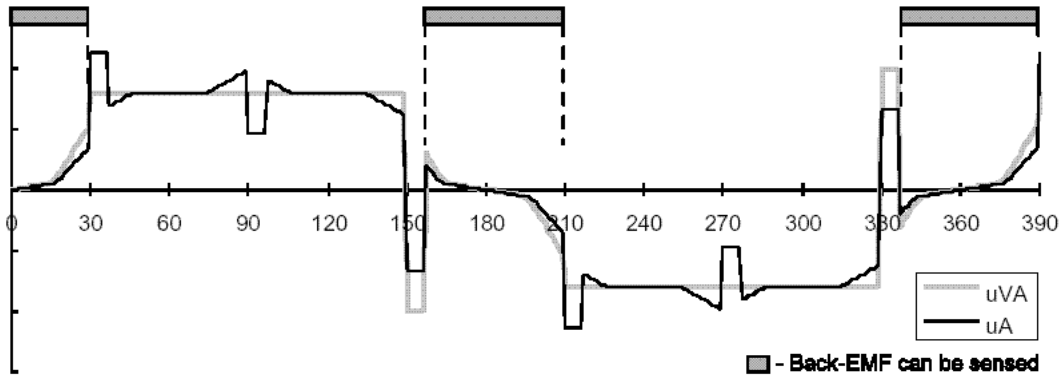


Figure 2-6. Phase Voltage Waveforms

However simple this solution looks, in reality it is more difficult, because the sensed branch voltage also contains some ripples.

2.3.3.1 Effect of Mutual Inductance

In BLDC motor's back-EMF wave shape, the mutual inductances play an important role. The difference in the mutual inductances between the coils that carry the phase current, and the coil used for back-EMF sensing, causes the PWM pulses to be superimposed onto the detected back-EMF voltage. In fact, it is produced by the high rate of change of phase current, transferred to the free phase through the coupling of the mutual inductances.

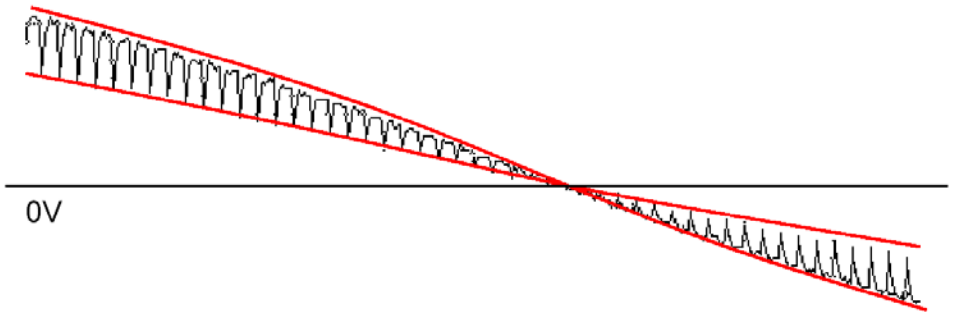


Figure 2-7. Mutual Inductance Effect

Figure 2-7 shows the real measured branch voltage. The red curves highlight the effect of the difference in the mutual inductances. This difference is not constant.

Due to the construction of the BLDC motor, both mutual inductances vary. They are equal at the position that corresponds to the back-EMF zero crossing detection.

The branch waveform detail is shown in Figure 2-8. Channel 1 shows the disturbed branch voltage. The superimposed ripples clearly match the width of the PWM pulses, and thus prove the conclusions drawn from the theoretical analysis.

The effect of the mutual inductance corresponds well in observations carried out on five different BLDC motors. These observations were made during the development of the sensorless technique.

NOTE

A BLDC motor with stator windings distributed in the slots has technically higher mutual inductances than other types. Therefore, this effect is more significant. On the other hand, the BLDC motor with windings wound on separate poles shows minor presence of the effect of mutual inductance.

CAUTION

However noticeable this effect, it does not degrade the back-EMF zero crossing detection, because it is cancelled at the zero crossing point. The addition of simple filtering helps to reduce the ripples further.

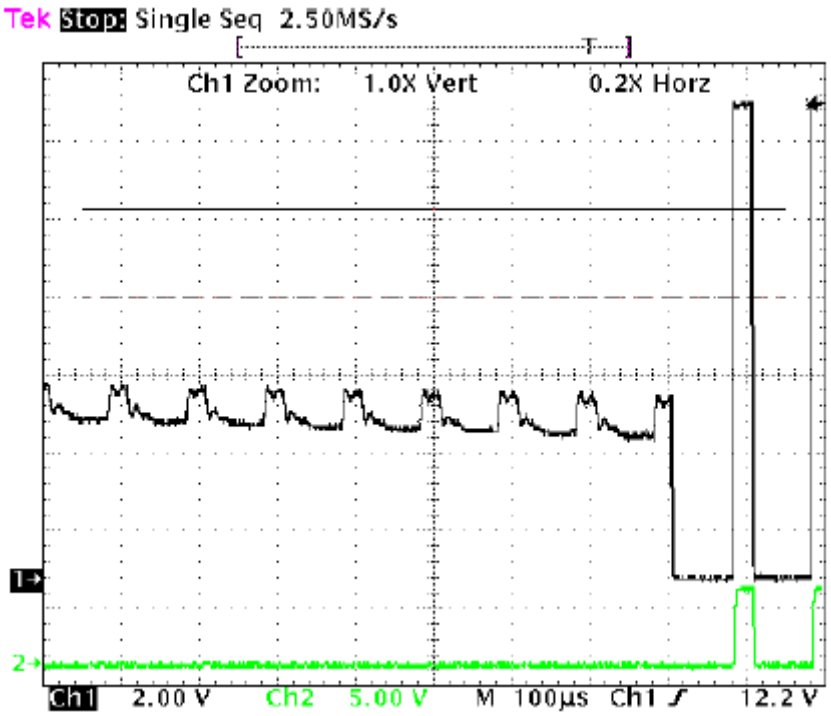


Figure 2-8. Detail of Mutual Inductance Effect

2.3.3.2 Effect of Mutual Phase Capacitance

The negative effect of mutual inductance is not the only one to disturb back-EMF sensing. So far, the mutual capacitance of the motor phase windings has been neglected in the motor model, since it affects neither the phase currents nor the generated torque. Usually, the mutual capacitance is very small. Its influence is significant only during PWM switching, when the system experiences very high *du/dt*.

The effect of the mutual capacitance can be studied using the model shown in Figure 2-9

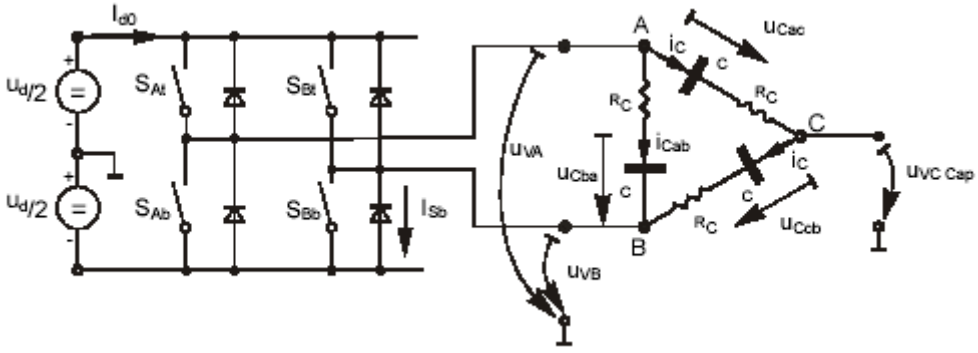


Figure 2-9. Mutual Capacitance Model

Consider the situation when motor phase A is switched from negative DC-bus rail to positive, and phase B is switched from positive to negative. This is described by the conditions defined by the following equations.

$$S_{Ab}, S_{Bt} \leftarrow PWM \tag{2-10}$$
$$u_{VA} = -\frac{1}{2}u_d \rightarrow \frac{1}{2}u_d, u_{VB} = \frac{1}{2}u_d \rightarrow -\frac{1}{2}u_d$$
$$i_{Cac} = i_{Ccb} = i_c$$

The voltage that disturbs the back-EMF sensing, using the free (not powered) motor phase C, can be calculated as follows.

$$u_{VCCap} = \frac{1}{2}(u_{Ccb} + u_{Cac} + 2R_C) - (u_{Ccb} + R_C) = \frac{1}{2}(u_{Cac} - u_{Ccb}) \tag{2-11}$$

The final expression for disturbing voltage can be found as follows.

$$u_{VCCap} = \frac{1}{2}\left(\frac{1}{C_{ac}} - \frac{1}{C_{cb}}\right)\int i_c dt = \frac{1}{2}\left(\frac{C_{cb} - C_{ac}}{C_{cb} \cdot C_{ac}}\right)\int i_c dt \tag{2-12}$$

NOTE

Equation 2-12 expresses the fact that only the unbalance of the mutual capacitance (not the capacitance itself) disturbs back-EMF sensing. When the capacitances are equal (balanced), the disturbances disappear. This is demonstrated in Figure 2-10 and Figure 2-11.

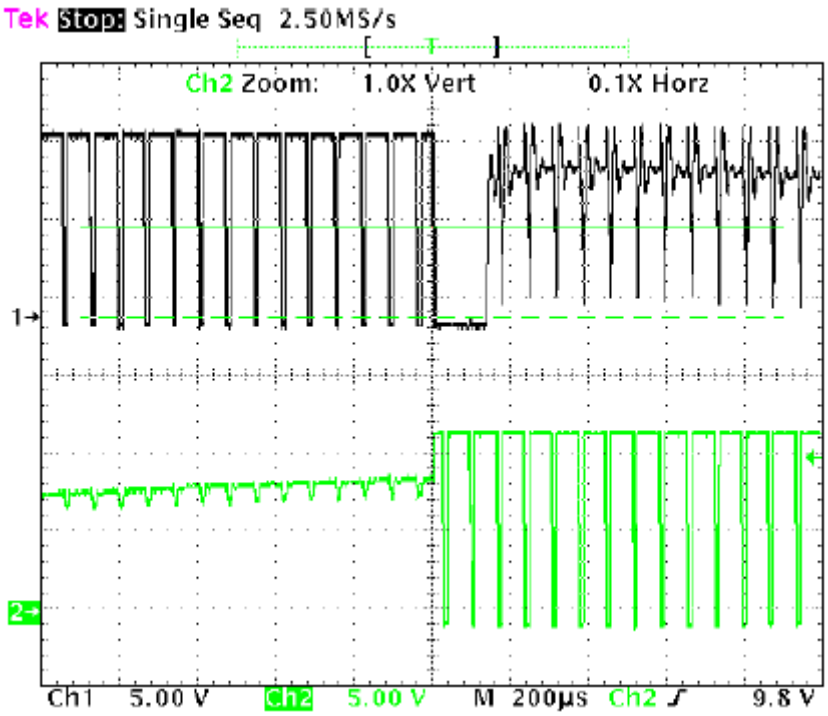


Figure 2-10. Distributed Back-EMF by Unbalanced Capacity Coupling

Channel 1 in Figure 2-11 shows the disturbed branch voltage, while the other phase (channel 2) is not affected because it faces balanced mutual capacitance. The imbalance was purposely made by adding a small capacitor on the motor terminals, to demonstrate the effect more clearly. After the imbalance is removed the branch voltage is clean, with no spikes.

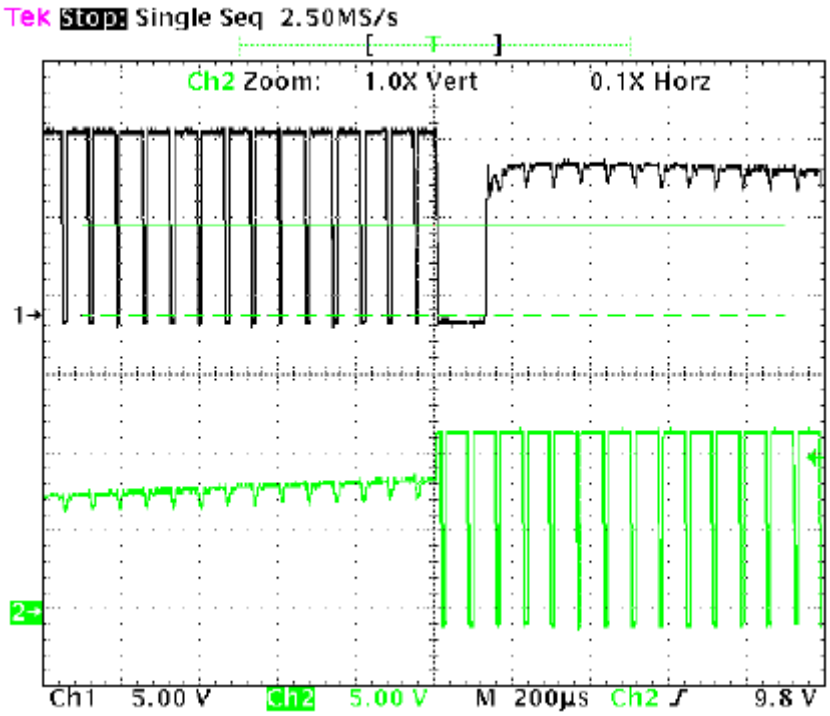


Figure 2-11. Balanced Capacity Coupling

NOTE

The configuration of the phase winding end turns has a significant impact; it must be managed properly to preserve the balance in the mutual capacity. This is important, especially for prototype motors, which are usually hand-wound.

NOTE

Failure to maintain balance of the mutual capacitance can easily disqualify such a motor from using sensorless techniques based on the back-EMF sensing. Usually, BLDC motors with windings wound on separate poles show minor presence of the mutual capacitance. Thus, the disturbance is also insignificant.

Chapter 3
System Concept

3.1 System Specification

The motor control system is designed to drive a 3-phase BLDC motor in a speed-controlled closed-loop system. The application meets the following performance specifications:

- Sensorless BLDC motor control by back-EMF sensing
- Targeted at MC56F8013 platform
- Running on a low voltage (12 VDC) 3-phase BLDC motor control development platform
- Control technique incorporates:
  - Sensorless control with closed-loop speed control in run mode and closed-loop current control in alignment mode
  - ADC zero crossing detection for sensorless control
  - Rotation in both directions
  - Motoring mode
  - Start from any motor position with rotor alignment
  - Automatic calibration of phase back-EMF measurements
  - Manual interface (RUN/STOP switch, UP/DOWN push button control)
  - FreeMaster software control interface (motor run/stop, speed setup, alarm indicators)
  - FreeMaster software remote monitor
  - DC over-voltage, DC under-voltage and DC over-current alarms
  - Adjustable DC-bus current limitation with software

3.2 Sensorless Drive Concept

The concept of the chosen system is shown in Figure 3-1.

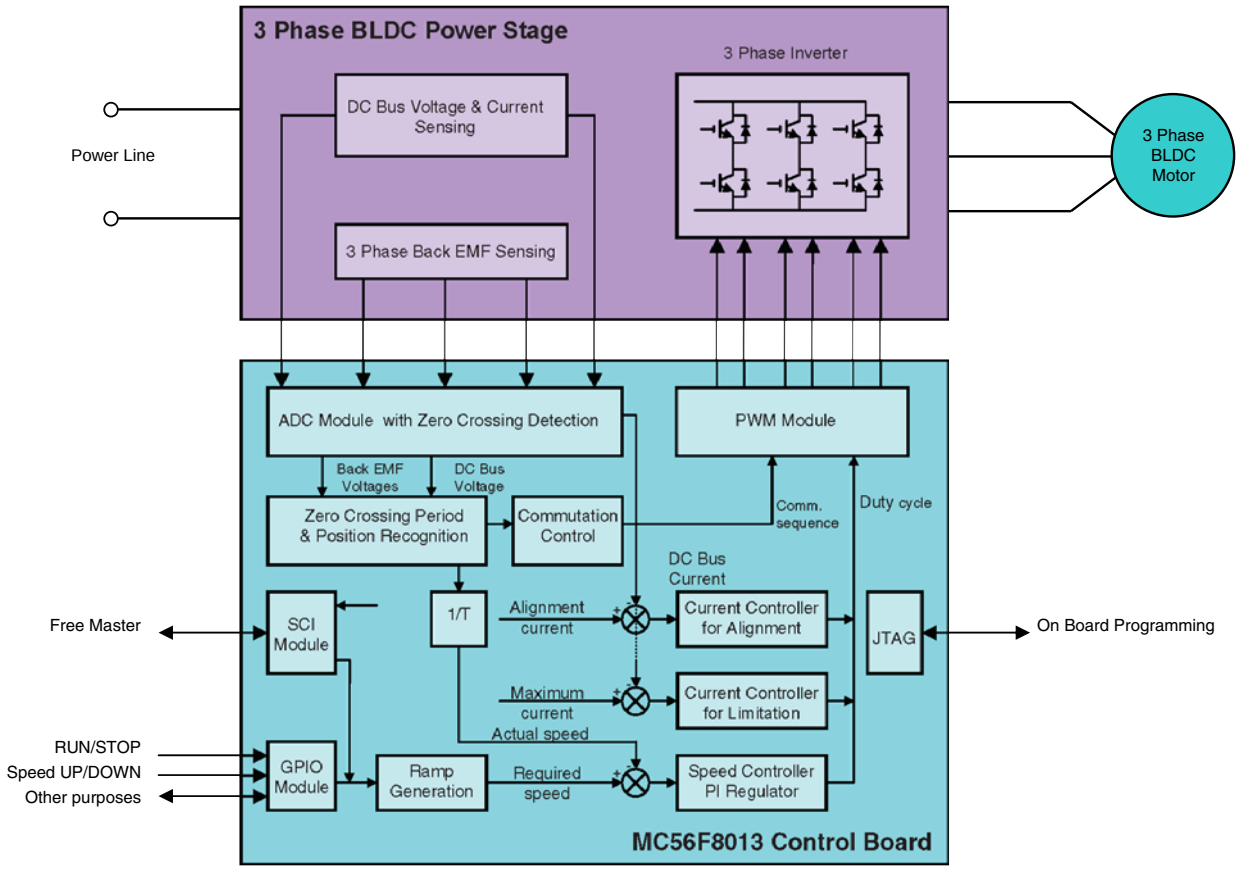


Figure 3-1. Drive Concept

The basic operation theory of the system is as follows.

At any time, the application can be in one of three modes: run, stop, and fault. Similarly, the commutation process can be in one of four modes: stop, alignment, starting, and running. These modes are described below.

The RUN/STOP switch and UP/DOWN buttons are monitored continuously over the GPIO interface. If the RUN/STOP switch is in the RUN position, and if there is no fault condition in the system, the application switches to run mode and waits for the speed setup. After the speed setup is changed from zero to any permitted value (using the UP/DOWN push buttons or FreeMaster interface), the commutation sequence starts. For this, first, the application performs the alignment, to adjust the rotor position for the rest of commutation. During alignment, the DC-bus current is kept constant. this is achieved by the current controller for alignment, where the DC-bus current is measured and fed into the Current Controller for alignment. After alignment, the application performs a calibration sequence, to compensate for the differences between back-EMF sensing networks. When alignment is finished, the application switches to the starting and running commutation states.

During starting and running, back-EMF signals are sensed by the ADC. The zero crossing period and the motor position are calculated from this information, and the zero crossing period is fed to the speed

controller. The set speed, which is fed to the speed controller in the form of a ramp, is compared with the actual speed and a speed error is generated. As with the current controller for alignment, the output of the speed controller is applied to the PWM module.

The current controller for limitation operates in parallel with the speed controller and the current controller for alignment. The aim of the current controller for limitation is to limit the DC-bus current to a predefined level. To achieve this, a filtered DC-bus current measurement is fed to this controller, where the limit level is defined by a software setpoint. The PWM duty cycle is decreased by this controller, only if the DC-bus current level is higher that the set level.

In the background, the DC-bus voltage is continuously measured for alarm recognition and for updating of the back-EMF detection offset.

DC over-voltage and DC under-voltage faults are detected by the ADC; DC over-current faults are detected by the PWM Fault input.

When a fault condition occurs, the application switches to the Fault state, stops the motor and waits for the fault to be reset via the FreeMaster interface or via the RUN/STOP switch. In addition, the FreeMaster interface allows monitoring of the system and adjustment of all system variables.

3.3 System Blocks Concept

3.3.1 PWM Voltage Generation for BLDC Motors

A 3-phase voltage system as described in 2.1 The Brushless DC Motor must be created to run the BLDC motor. It is provided by a 3-phase power stage with six power switches (IGBTs or MOSFETs) controlled by the hybrid controller's on-chip PWM module.

When generating PWM signals for this BLDC motor control application, the bottom and top power switches of the non-fed phase must be switched off (see Figure 3-2 and Figure 3-3).

For the application, PWM signals can be created in two ways: independent PWM mode and complementary PWM mode. Thanks to the MC56F8013 PWM module and its Mask/Swap feature, both modes can be used with minor software overhead.

Complementary PWM Mode

In complementary PWM mode, the top and bottom switch of a phase operate in a complementary way. This mode enables regenerative braking, where the DC-bus voltage may increase during deceleration or stop. In this mode, Swap and Mask features of PWM module are used together.

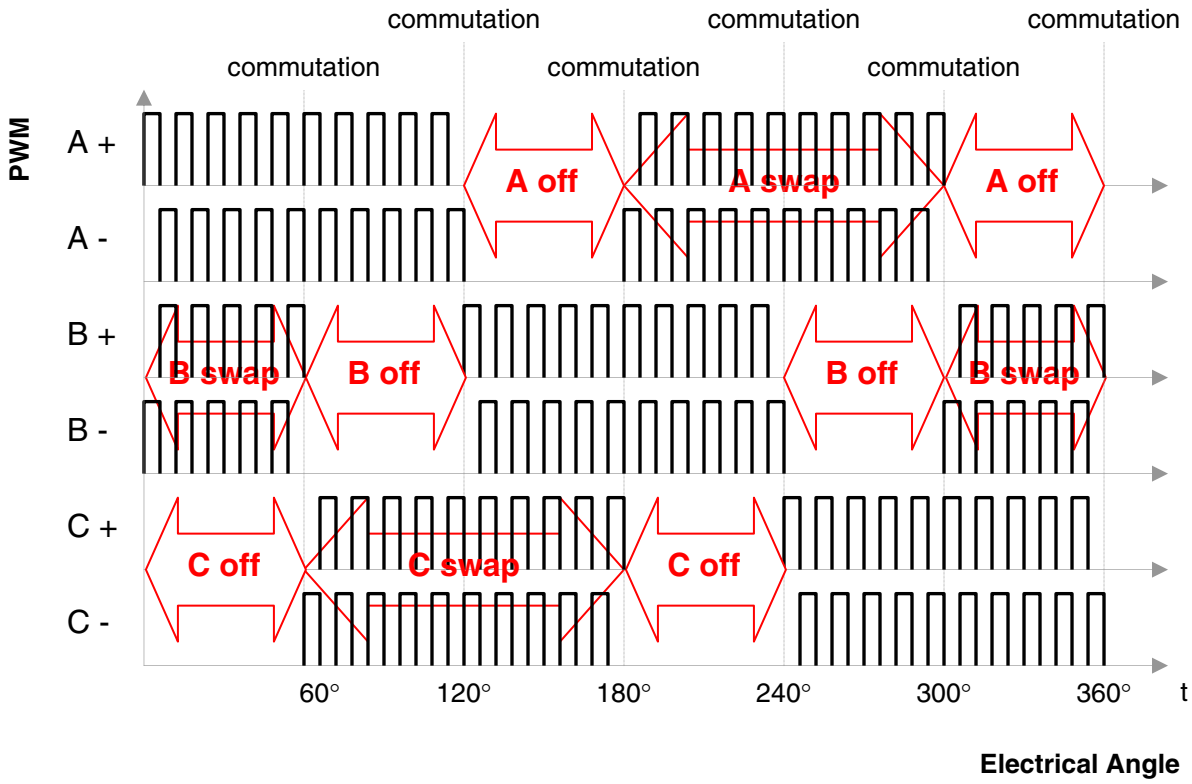


Figure 3-2. Complementary PWM Mode Patterns for BLDC Motor

Independent PWM Mode

In independent PWM mode, the top and bottom switch of a phase operate independent over a commutation period, if the top switch is performing PWM, the bottom switch is off, and vice versa. In this mode, regenerative braking is not enabled. Only the Mask feature of the PWM module is required.

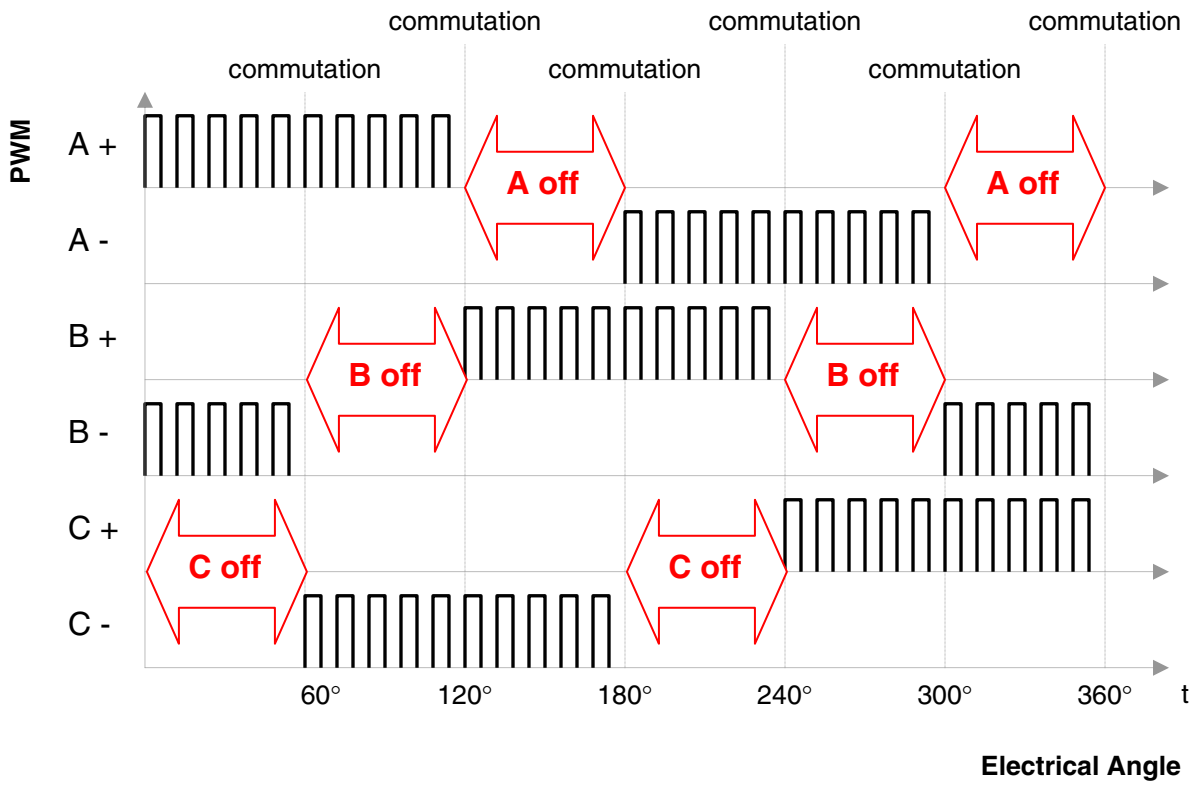


Figure 3-3. Independent PWM Mode Patterns for BLDC Motor

3.3.2 ADC Sampling Mechanism

PWM switching of the power stage causes high voltage spikes on the phase voltages. These voltage spikes are generated on the non-fed phase because of mutual inductances and mutual capacitor couplings between the motor windings. The non-fed phase branch voltage is then disturbed by PWM switching. Figure 3-4 shows the effect of the mutual inductance on the non-fed phase.

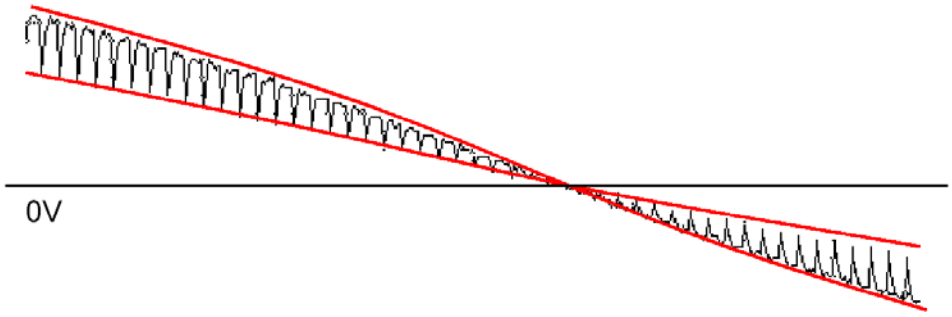


Figure 3-4. Effect of mutual inductance on Back-EMF

The non-fed phase branch voltage is disturbed at the PWM switching edges. Therefore, the application synchronizes the back-EMF zero crossing detection with the PWM. The analog to digital conversion of phase branch voltages is triggered in the middle of the PWM pulse. Then the voltage for the back-EMF is sensed at moments when the non-fed phase branch voltage is already stabilized.

To set the exact moment of sampling, the MC56F801x family offers the ability to synchronize ADC and PWM modules via the SYNC signal. The PWM outputs a synchronization pulse, which is connected as an input to the synchronization module T3 (Quad Timer 3) via the system integration module. A high-true pulse occurs for each reload of the PWM, regardless of the state of the LDOK bit. The intended purpose of T3 is to provide a user-selectable delay between the PWM SYNC signal and the updating of the ADC values. A conversion process can be initiated by the SYNC input, which is an output of TC3. The delay from the PWM SYNC signal to the start of the ADC measurement is updated after every calculation of the PWM duty cycle, to ensure that the samples are taken in the middle of the PWM pulses. The time diagram of the automatic synchronization between PWM and ADC is shown in Figure 3-5.

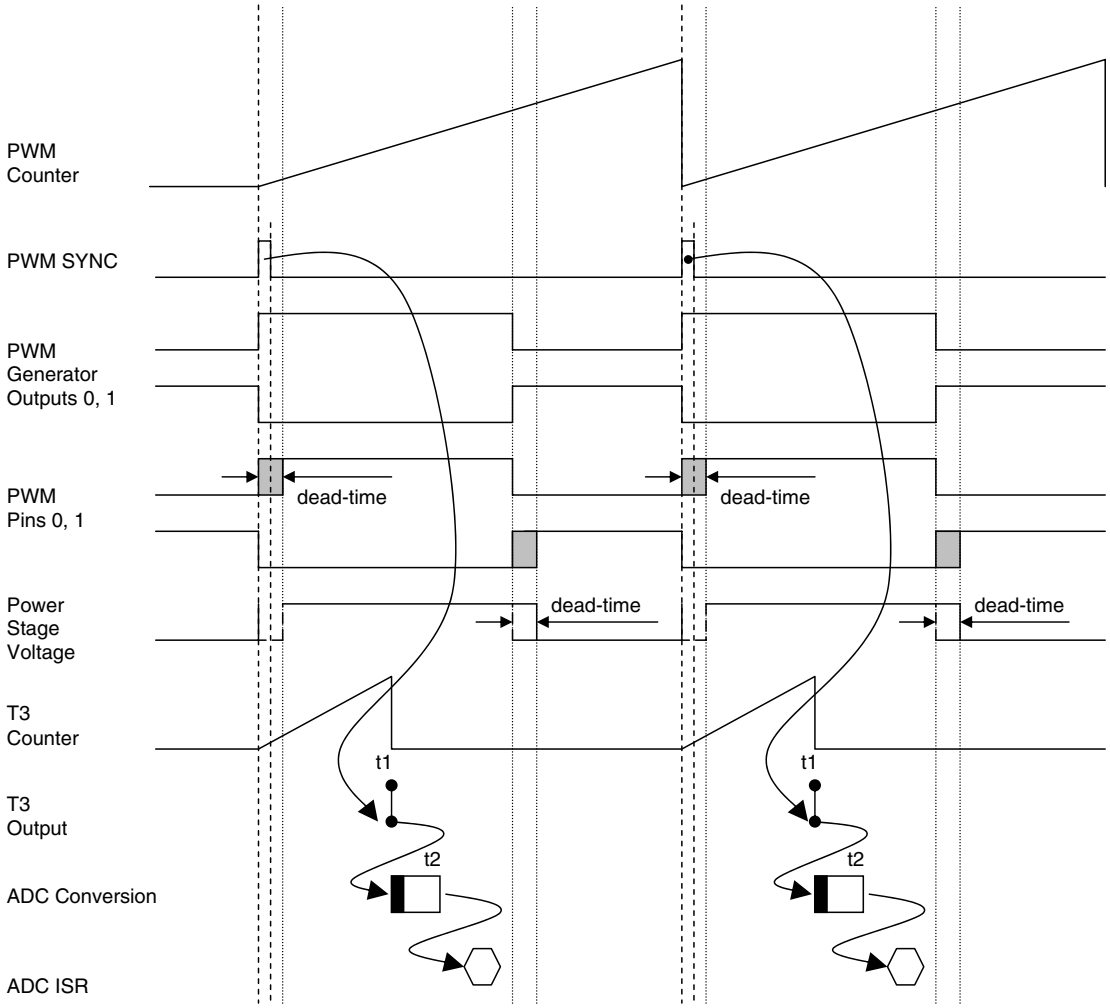


Figure 3-5. Time Diagram of PWM and ADC Synchronization

3.3.3 Back-EMF Zero Crossing Detection

Back-EMF zero crossing is detected by sensing the motor's non-fed phase branch voltage (u_{vi} in 2.3.3 Back-EMF Sensing) and DC-bus voltage u_d using the ADC. (See Chapter 2 Control Theory).

The Freescale MC56F801x family offers an excellent on-chip ADC module. Its unique feature set provides automatic detection of the signal crossing the value contained in the ADC offset register.

Back-EMF zero crossing detection can be split into two main tasks:

- ADC zero crossing checking
- ADC zero crossing offset setting to follow the variation of the DC-bus voltage

3.3.3.1 ADC Zero Crossing Checking

Zero crossing for position estimation is detected using the ADC.

As stated, the ADC has individual ADC offset registers for each ADC channel. The value in the offset register can be subtracted from the ADC output. The final result of the ADC conversion is then two's compliment data. The other feature associated to the offset registers is the zero crossing interrupt. The zero crossing interrupt is asserted whenever the ADC conversion result changes the sign compared to the previous conversion result. Refer to the manual for detailed information. Therefore, this application uses an ADC zero crossing Interrupt to get the back-EMF zero crossing event.

3.3.3.2 ADC Zero Crossing Offset Setting

Actually, the zero crossing is detected, if the phase voltage of the non-fed phase changes it sign, when the offset value is set to half of the DC-bus. Therefore, the ADC offset register must be set to one half of the DC-bus voltage value. This update must be continuously performed, to reflect the DC-bus voltage variation caused by the ripple of DC-bus voltage. This is valid at the following conditions:

- Motor phases are symmetrical (all 3-phases have same parameters)
- All hardware dividers for the ADC of the DC-bus voltage and all 3-phase voltages, have equal ratio

In this application, to compensate the differences between hardware dividers of phase voltage measurements, a calibration sequence is followed. This calibration is performed only once, just after the first alignment of the motor. During calibration, alignments are performed for each phase and the third phase is measured to obtain the offset value. Ideally, the value read from the third phase must be equal to half the DC-bus voltage, but because of component tolerances, if differs from 50%. When the commutation is running, these measured calibration values are multiplied by the measured DC-bus voltage and the ADC offset registers are updated.

NOTE

*In the application software, use the **#define** directives in bldcadczcconfig.h file to enable or disable the calibration feature.*

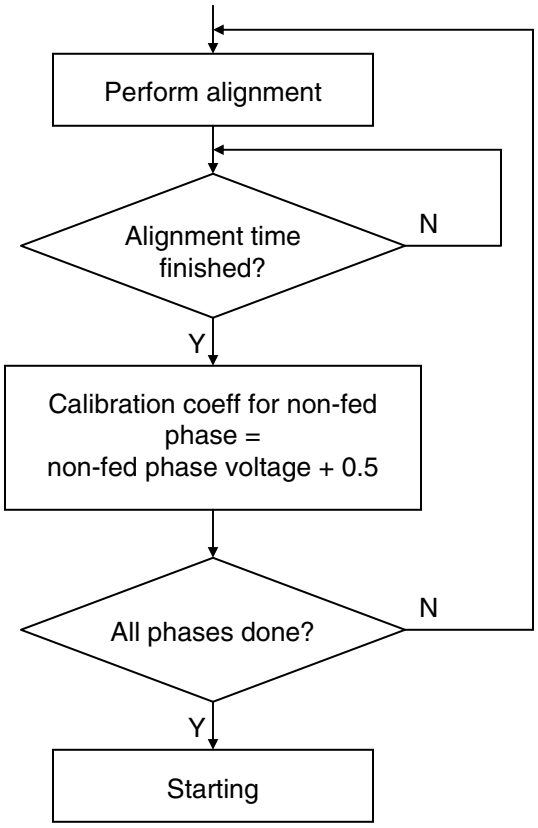


Figure 3-6. Calibration

3.3.4 Sensorless Commutation Control

This section describes sensorless BLDC motor commutation using the back-EMF zero crossing technique. To start and run the BLDC motor, the control algorithm must go through the following states:

- Alignment
- Starting (Back-EMF Acquisition)
- Running

Figure 3-7 shows the transitions between these states. First, the rotor is aligned to a known position without position feedback. When the rotor moves, a back-EMF is induced on the non-fed phase and is acquired by the ADC. As a result, the position is known and can be used to calculate the speed and process the commutation in the running state.

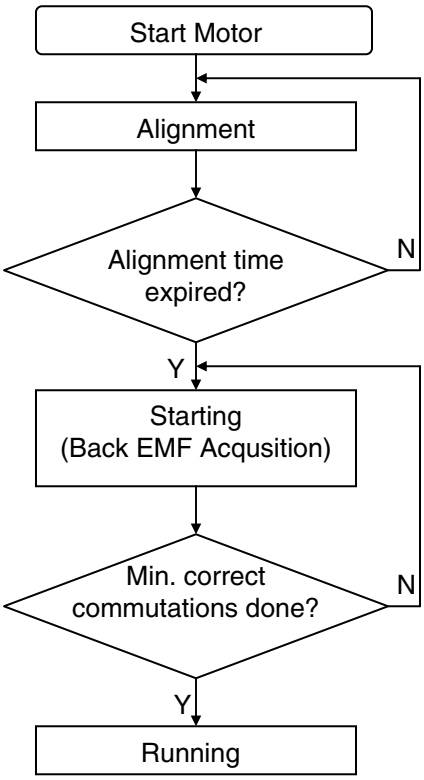


Figure 3-7. Commutation Control States

3.3.4.1 Alignment

Before the motor starts, there is a short time (which depends on the motor's electrical and mechanical time constant) when the rotor position is aligned to a known position by applying PWM signals to only two motor phases (no commutation). The alignment current controller keeps the current within predefined limits. This state is required to create a high startup torque and to recognize the rotor position. When the preset timeout expires, this state is finished.

The alignment current controller subroutine (with PI regulator) is called to control the DC-bus current. The subroutine sets the correct PWM ratio for the required current. The current is sampled and the current controller is calculated during every PWM cycle.

The BLDC motor rotor position (with flux vectors during alignment) is shown in Figure 3-8.

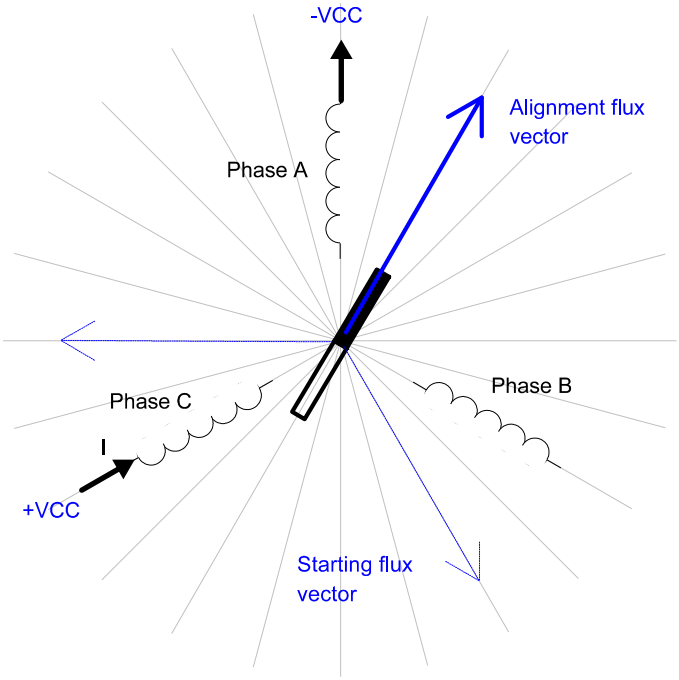


Figure 3-8. Alignment

3.3.4.2 Starting (Back-EMF Acquisition)

The back-EMF sensing technique enables sensorless detection of the rotor position; however, the drive must first be started without this feedback. This is necessary, because the amplitude of the induced voltage is proportional to the motor speed; hence, the back-EMF cannot be sensed at very low speeds, and a special startup algorithm must be used.

To start the BLDC motor, adequate torque must be generated. The motor torque is proportional to the stator magnetic flux, the rotor magnetic flux, and the sine of angle between both magnetic fluxes.

This implies (for BLDC motors) the following:

- The level of phase current must be high enough.
- The angle between the stator and rotor magnetic fields must be in the range 90° ± 30°.

The first condition is satisfied during the alignment state by keeping the DC-bus current at a level sufficient to start the motor. In the starting state (back-EMF acquisition), the same value of PWM duty cycle is used as the one that stabilized the DC-bus current during the alignment state.

The second condition is more difficult to fulfil without having position feedback information. After the alignment state, both the stator and the rotor magnetic fields are aligned (0° angle). Therefore, two fast commutations (faster then the rotor can follow) must be applied to create an angular difference between the magnetic fields (see Figure 3-9).

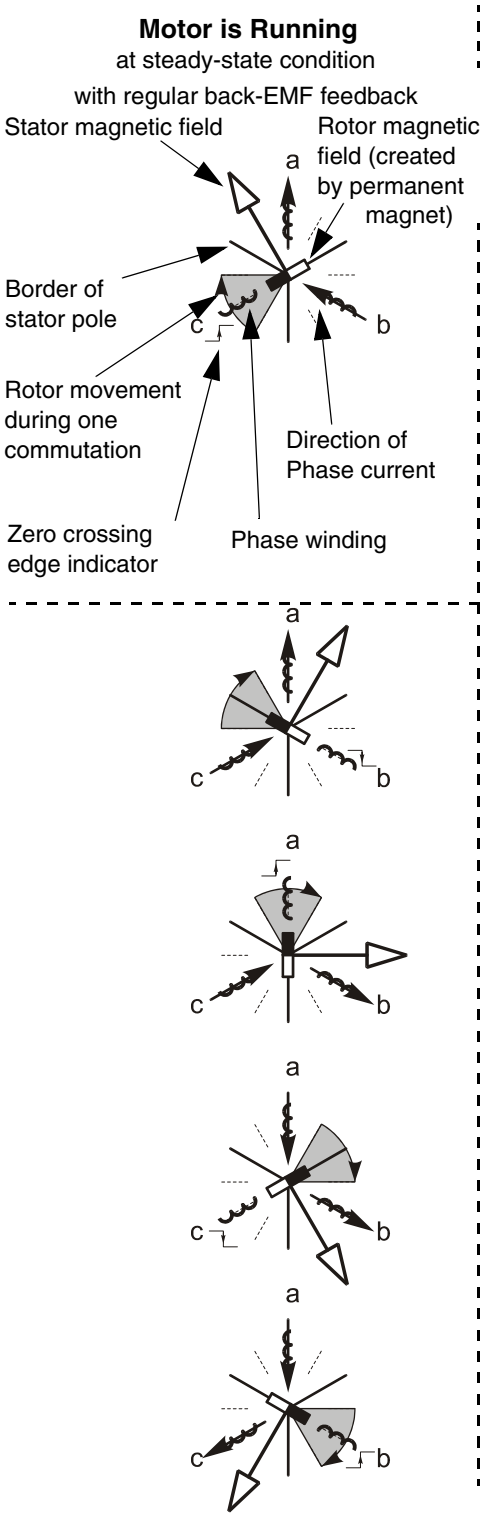
The commutation time is defined by the start commutation period (**Per_CmtStart**).

This allows the motor to be started with a required torque and a minimum speed to be achieved. This minimum speed is defined as the speed at which back-EMF zero crossings can be detected. Until back-EMF zero crossing detection is locked to the commutation process (defined as the run state, explained in 3.3.4.3 Running), the starting process ensures that commutations are done manually, with predefined commutation times.

After several successive back-EMF zero crossings, the exact commutation times can be calculated. The commutation process is adjusted. The control flow continues to the running state. The BLDC motor is then running with regular feedback and the speed controller can be used to control the motor speed by changing the PWM duty cycle value.

For starting, there are three possible zero crossing detection scenarios:

- Normal operation — Zero crossing is detected between two commutation period. This is the ideal operation.
- No zero crossing detected — No zero crossing is detected between two commutation periods.
- Zero crossing missed — In the application, after every commutation period, zero crossing detection is disabled for a time Toff, which is proportional to the commutation period. After time Toff has expired, the application checks the polarity of the back-EMF signal. If the polarity is not as expected, it means that the back-EMF shape had a zero crossing when the application was waiting during time Toff. In this case, the application decides that a zero crossing was missed.



Alignment State

The rotor position is stabilized by applying PWM signals to only two motor phases

Starting (back-EMF Acquisition)

The two fast (faster then the rotor can move) commutation are applied to create an angular difference of the stator magnetic field and rotor magnetic field.

The back-EMF feedback is tested. When the back-EMF zero crossing is recognized the time of new commutation is evaluated. Until at least two successive back-EMF zero crossings are received the exact commutation time can not be calculated. Therefore the commutation is done in advance to ensure that successive back-EMF zero crossing event would not be missed.

Running

After several back-EMF zero crossing events the exact commutation time is calculated. The commutation process is adjusted.
Motor is running with regular back-EMF feedback.

Figure 3-9. Vectors of Magnetic Fields

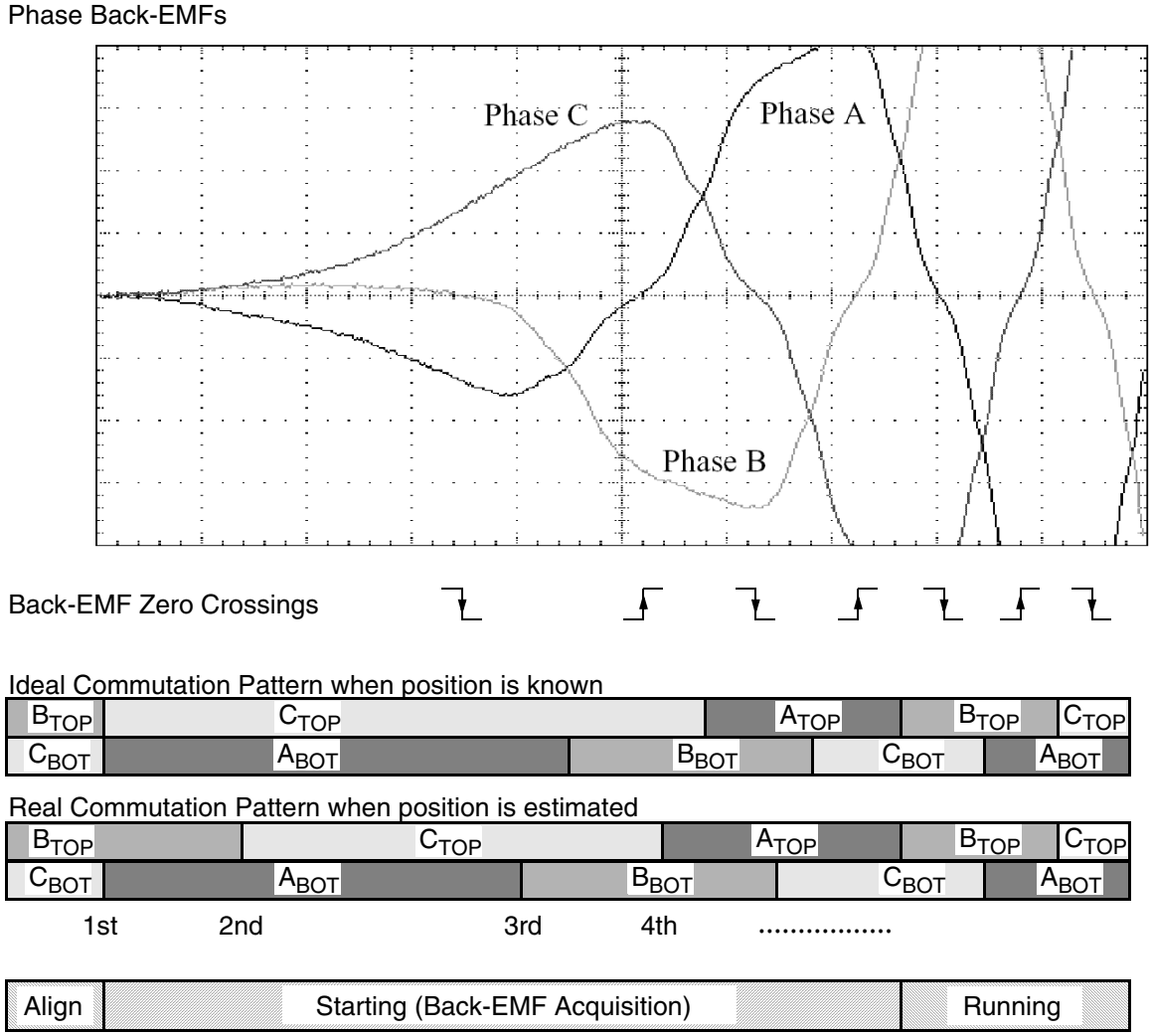


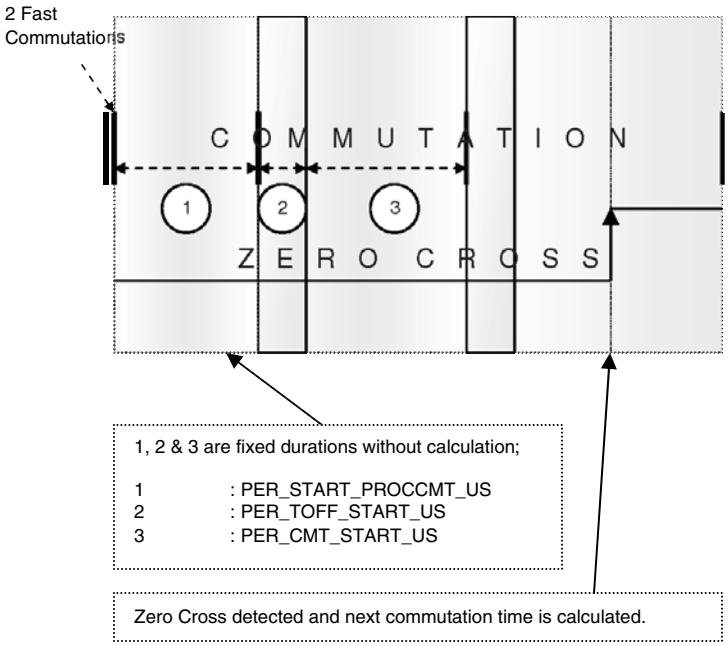
Figure 3-10. Back-EMF at Startup

Figure 3-10 demonstrates the back-EMF during startup. The amplitude of the back-EMF varies according to the rotor speed. During the starting state (back-EMF acquisition), the commutation is done in advance. In the running state, the commutation is done at the right times.

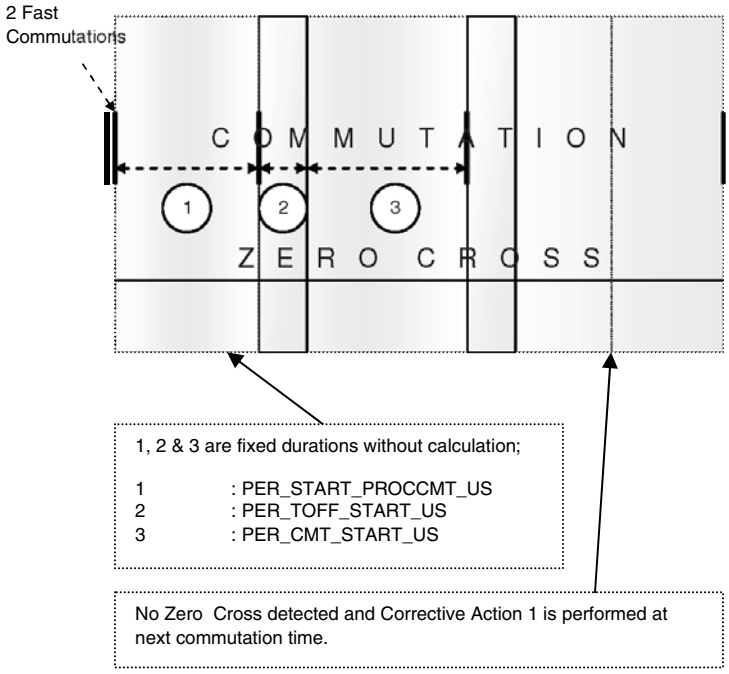
Starting — Calculation of Commutation Times

During starting, next commutation time is calculated in three different way, according to zero crossing detection, after regular operations during start.

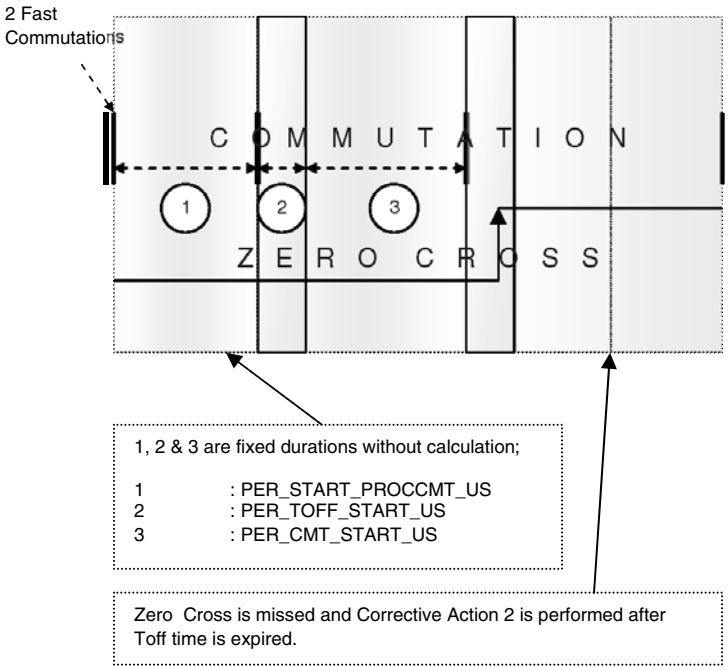
- Normal operation — During start, zero crossing is detected and the next commutation time is calculated.



- No zero crossing detected — During startup, no zero crossing is detected and corrective action 1 is performed.



- Zero crossing missed — Zero crossing is missed and corrective action 2 is performed.



3.3.4.3 Running

The commutation process is the series of states which is assured when back-EMF zero crossing is successfully captured. The new commutation time is calculated after back-EMF zero crossing is captured and the commutation is performed. The following processes must be provided:

- BLDC motor commutation service
- Back-EMF zero crossing moment capture service
- Computation of commutation times
- Handler for interaction between these commutation processes

Algorithms BLDC Motor Commutation with Zero Crossing Detection

The diagrams above aid in explaining how the commutation works. After commuting the motor phases, a time interval (Per_Toff[n]) is set, which allows the shape of the back-EMF to be stabilized. Stabilization is required because the electromagnetic interference and flyback current in antibody diode can generate glitches that may add to the back-EMF signal. This can cause a misinterpretation of back-EMF zero crossing. Then the new commutation time (T2[n]) is preset and performed at this time if the back-EMF zero crossing is not captured. If the back-EMF zero crossing is captured before the preset commutation time expires, then the exact calculation of the commutation time (T2*[n]) is made based on the captured zero crossing time (T_ZCros[n]). The new commutation is updated at this new time.

Similar to starting, for running there are three possible zero crossing detection scenarios.

- Normal operation — Zero crossing is detected between two commutation period. This is the ideal operation.
- No zero crossing detected — No zero crossing is detected between two commutation periods.

- Zero crossing missed — In the application, after every commutation period, zero crossing detection is disabled for a time Toff, which is proportional to the commutation period. After time Toff has expired, the application checks the polarity of the back-EMF signal. If the polarity is not as expected, it means that the back-EMF shape had a zero crossing when the application was waiting during time Toff. In this case, the application decides that a zero crossing was missed.

If the back-EMF feedback is lost, for any reason, within one commutation period, corrective action is taken to return to the regular states.

The flow chart explaining the principle of BLDC Commutation control with back-EMF zero crossing detection is shown in Figure 3-11.

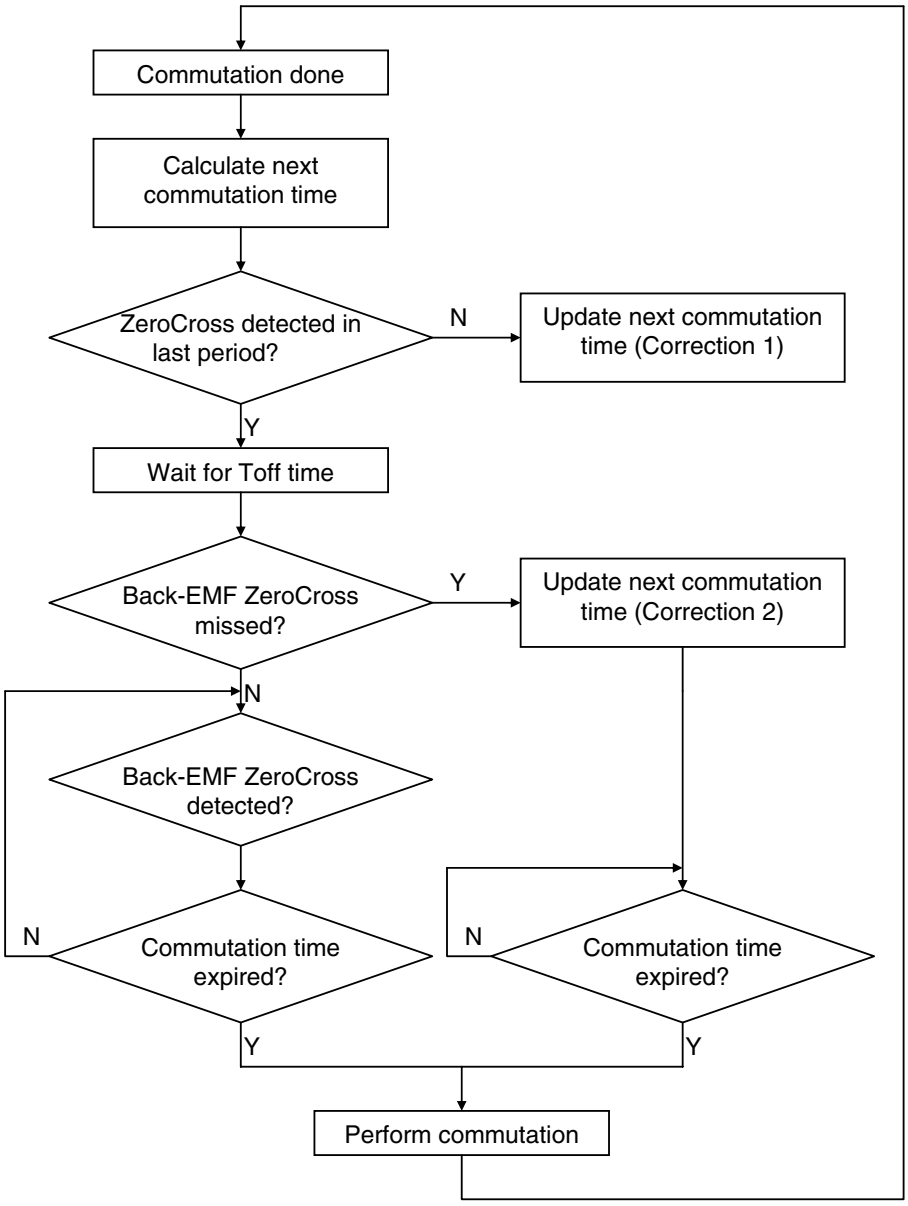


Figure 3-11. Flow Chart of BLDC Commutation with Back-EMF Zero Crossing Detection

Running — Calculation of Commutation Times

Similar to start, during running, the next commutation time is calculated in three different ways, according to zero crossing detection.

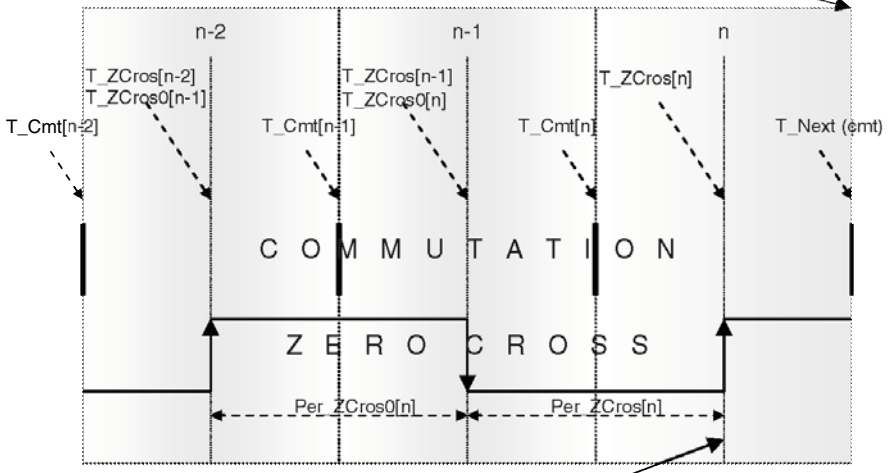
- Normal operation — Zero crossing is properly detected and all calculations are performed as expected. After zero crossing detection, the preset time for commutation is updated.

Commutation performed and next commutation time is predicted as following :

T_Next[n](cmt) = T_Cmt0[n] + Per_CmtPreset[n]
= T_Cmt0[n] + Coef_CmtPrecomp * Per_ZCrosFlt[n-1]

If Coef_CmtPrecomp * Per_ZCrosFlt > Max_PerCmt =>

Result is limited at Max_PerCmt



Zero Cross detected and next commutation time is calculated as following :

Per_ZCros[n] = T_ZCros[n] - T_ZCros[n-1] = T_ZCros[n] - T_ZCros0

Per_ZCrosFlt[n] = (1/2 * Per_ZCros[n] + 1/2 * Per_ZCros0)

HlfCmt[n] = 1/2 * Per_ZCrosFlt[n] - Advance_angle
= 1/2 * Per_ZCrosFlt[n] - Coef_HlfCmt * Per_ZCrosFlt[n]

T_Next[n](cmt) = T_ZCros[n] + HlfCmt[n]

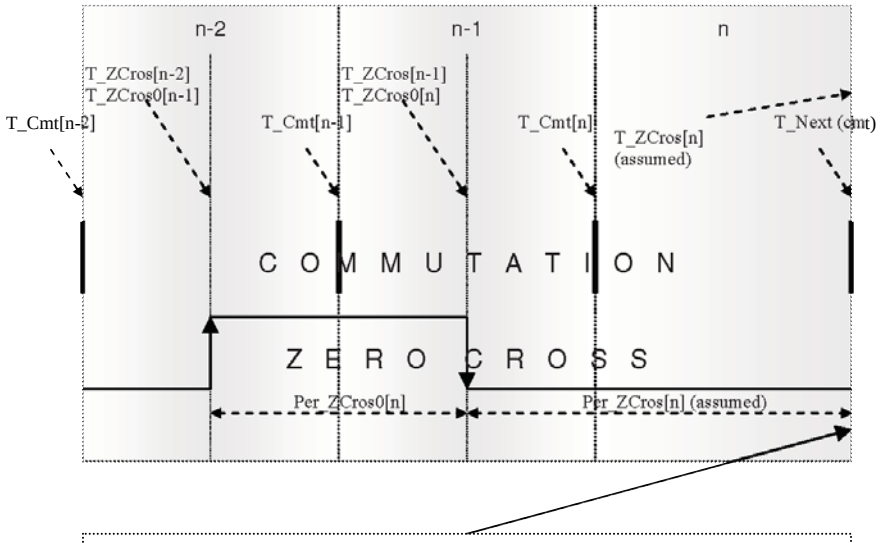
The best commutation was get with Advance_angle: 60Deg*1/8 = 7.5Deg which means Coef_HlfCmt = 0.375 at Running state!

Per_Toff[n+1] = minimum(Per_ZCrosFlt * Coef_Toff, Max_PerCmtProc)
(Coef_Toff = 0.35 at Running state, Max_PerCmtProc = 100!)

Per_ZCros0 <= Per_ZCros[n]

T_ZCros0 <= T_ZCros[n]

- No zero crossing detected: In this case, there is no zero crossing detected between two commutations. Therefore, commutation is performed at the preset time and corrective action 1 is performed.



No Zero Cross detected in Commutation period.
Corrective Action 1 is performed as following :

T_ZCros[n] <= T_Cmt[n+1]

Per_ZCros[n] = T_ZCros[n] - T_ZCros[n-1] = T_ZCros[n] - T_ZCros0

Per_ZCrosFlt[n] = (1/2 * Per_ZCros[n] + 1/2 * Per_ZCros0)

HlfCmt[n] = 1/2 * Per_ZCrosFlt[n] - Advance_angle
= 1/2 * Per_ZCrosFlt[n] - Coef_HlfCmt * Per_ZCrosFlt[n]

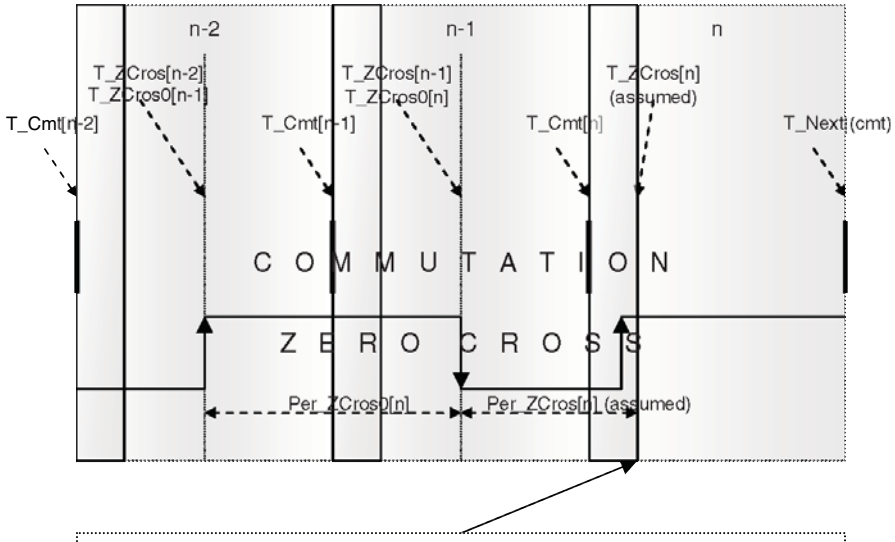
The best commutation was get with Advance_angle: 60Deg*1/8 = 7.5Deg which means Coef_HlfCmt = 0.375 at Running state!

Per_Toff[n+1] = minimum (Per_ZCrosFlt * Coef_Toff, Max_PerCmtProc)

Per_ZCros0 <= Per_ZCros[n]

T_ZCros0 <= T_ZCros[n]

- Zero crossing missed — In this case, it has been detected that zero crossing was missed. This detection is performed by monitoring the polarity of the back-EMF voltage of the non-fed phase, after time Toff has expired. corrective action 2 is performed.



Zero Cross is missed, because it was during Toff time.
Corrective Action 2 is performed as following :

T_ZCros[n] <= CmtT[n]+Toff[n]

Per_ZCros[n] = T_ZCros[n] - T_ZCros[n-1] = T_ZCros[n] - T_ZCros0

Per_ZCrosFlt[n] = (1/2 * T_ZCros[n] + 1/2 * T_ZCros0)

HlfCmt[n] = 1/2 * Per_ZCrosFlt[n] - Advance_angle
= 1/2 * Per_ZCrosFlt[n] - Coef_HlfCmt * Per_ZCrosFlt[n]

The best commutation was get with Advance_angle: 60Deg*1/8 = 7.5Deg which means Coef_HlfCmt = 0.375 at Running state!

Per_ZCros0 <= Per_ZCros[n]

T_ZCros0 <= T_ZCros[n]

Where;

T_Cmt[n]:	Time for commutation for step n.
T_Next[n] (cmt):	Time of the next commutation for step n
T_ZCros[n]:	Time of the zero crossing for step n
T_zCros0[n]:	Time of the previous zero crossing for step n
Per_Toff[n+1]:	Period of the Toff duration to be applied at step n+1
Per_CmtPreset:	Preset commutation period from commutation to next commutation if no zero crossing was captured
Per_ZCros[n]:	Period between last two zero crossings for step n
Per_ZCros0[n]:	Previous period between zero crossings for step n
Per_ZCrosFlt:	Estimated period of commutation filtered for step n
Per_HlfCmt:	Period from zero crossing to commutation (half commutation)

The required commutation timing is provided by setting of commutation constants: **Coef_CmtPrecompFrac, Coef_CmtPrecompLShft, Coef_HlfCmt, Coef_Toff,** in structure **RunComputInit.**

Chapter 4
Hardware

4.1 Hardware Implementation

The BLDC sensorless application runs on Freescale's MC56F8013 evaluation board (EVB) and on many power stage and motor configurations. For simplification, it can be better to split power stages and motors, because both motors can be used with both power stages.

The application can run on the following power stages:

- EVM33395 evaluation motor board
- Micro power board

The application can run with following motors:

- IB23811 motor (produced by MCG)
- N2311 low voltage motor (produced by Pittman)

One of the power stages and one of the motors can be selected to create an application setup and can run using the MC56F8013 EVB.

More information on running the application with different power stages and motors is given in Chapter 7 Tuning.

NOTE

*In the application software, use the **#define** directives in the bldcadczcconfig.h file to select the desired power stage board and motor.*

NOTE

Although this reference design demonstrates the application using low voltage power stages, it can be adapted to high voltage power stages and applications as well.

4.1.1 Hardware Implementation Using EVM33395 Evaluation Motor Board

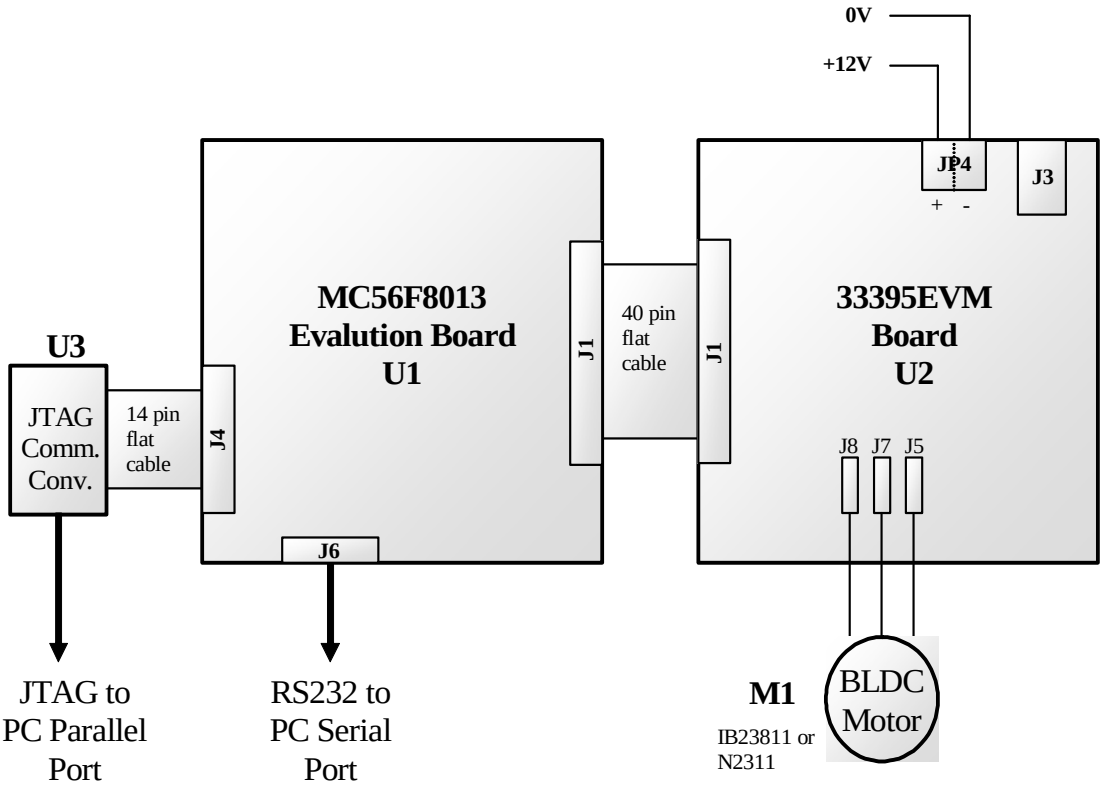


Figure 4-1. Hardware Diagram of System Using EVM33395 Board

4.1.2 Hardware Implementation with Micro Power Board

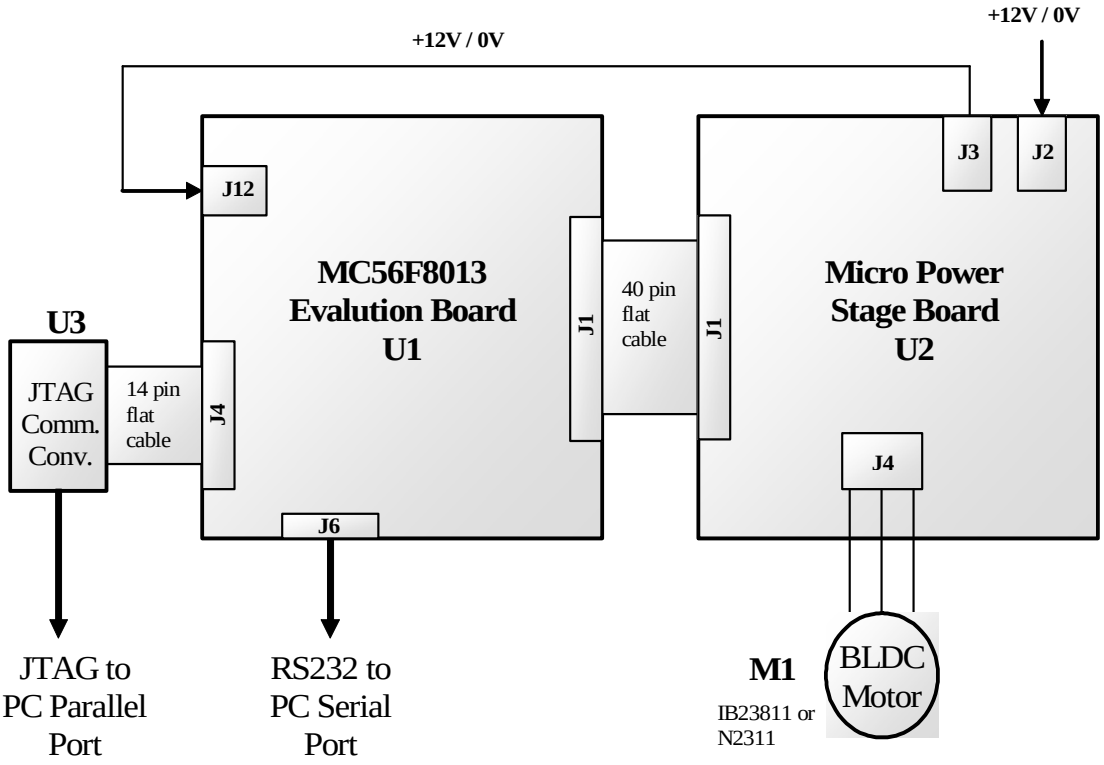


Figure 4-2. Hardware Diagram with Micro Power Board

4.2 Component Descriptions

4.2.1 MC56F8013 EVB

The MC56F8013 EVB is used to demonstrate the abilities of the MC56F8013 and to provide a hardware tool, for use in the development of applications that use the MC56F8013.

The MC56F8013 EVB is an evaluation board that includes a MC56F8013 part, peripheral expansion connectors, and some peripheral expansions. The expansion connectors are for signal monitoring and user feature expandability.

The EVB has the following features:

- MC56F8013 16-bit +3.3 V hybrid controller operating at 32 MHz
- Internal oscillator
- Joint Test Action Group (JTAG) port interface connector for an external debug host target interface
- RS-232 interface
  - Galvanic isolation
  - Single wire/bi-wire communication option
- SPI connector
- 64 kilobit serial EEPROM
- UNI-3 motor interface
- Encoder/Hall effect interface

- Configurable ADC pin connections
- DC-bus over-voltage sensing and over-current sensing
- Phase back-EMF sensing
- Zero crossing detection
- Tacho-dynamo interface for digital/analog sensing
- Push-buttons (Reset, Up, Down)
- RUN/STOP toggle switch
- Indication LEDs (Power ON, Fault, User, PWM)
- On-board power regulation from an external 12 VDC supplied power input
- All GPIOA, GPIOB, GPIOC pins available to user via header pins

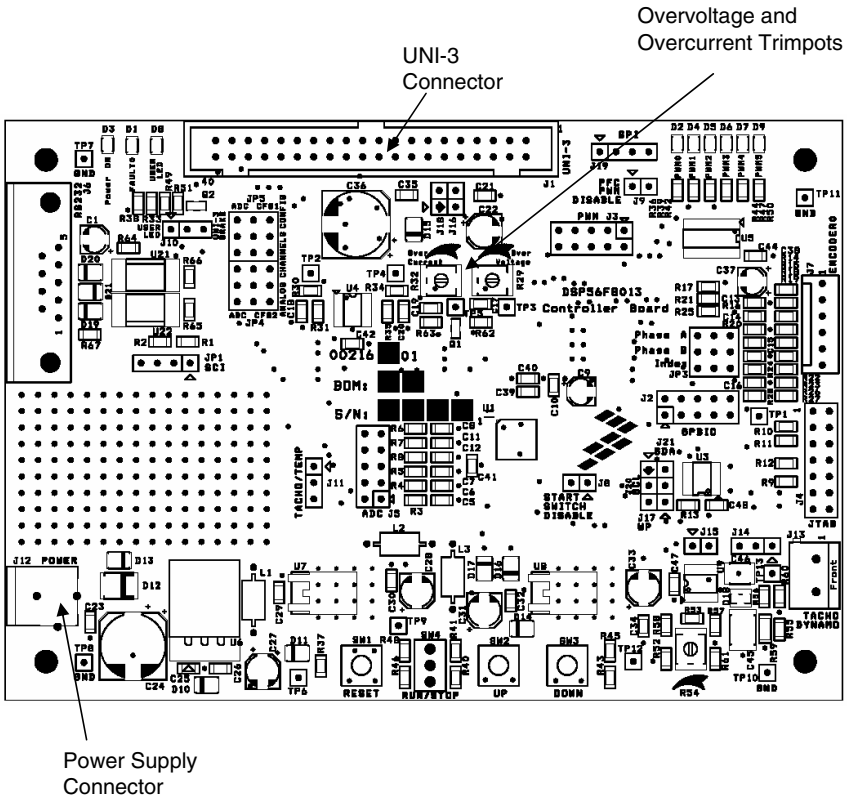


Figure 4-3. MC56F8013 EVB

4.2.2 Power Stage Boards

The BLDC sensorless application can run on two low voltage (12 VDC) power stages. These power stages are described below.

4.2.2.1 EVM33395 Evaluation Motor Board

Freescale's embedded motion control series SMOS evaluation motor board is a 12-volt, 8-amp, surface-mount power stage with an analog SMOS driver. In combination with one of the embedded motion control series control boards, it provides a development platform that allows algorithms to be written and tested without the need to design and build a power stage. It supports algorithms that use Hall sensors, encoder feedback, and back-EMF signals for sensorless control.

The SMOS evaluation motor board has an over-current protection that is independent of the control board, yet some care in its setup and use is required for board or motor protection. Current-measuring circuitry is set up for 8 amps full scale, according to trimmer position. A 25°C ambient temperature operation with output current up to 10 A RMS continuous is within the board's thermal limits. Note that there is no thermal protection provided on the board.

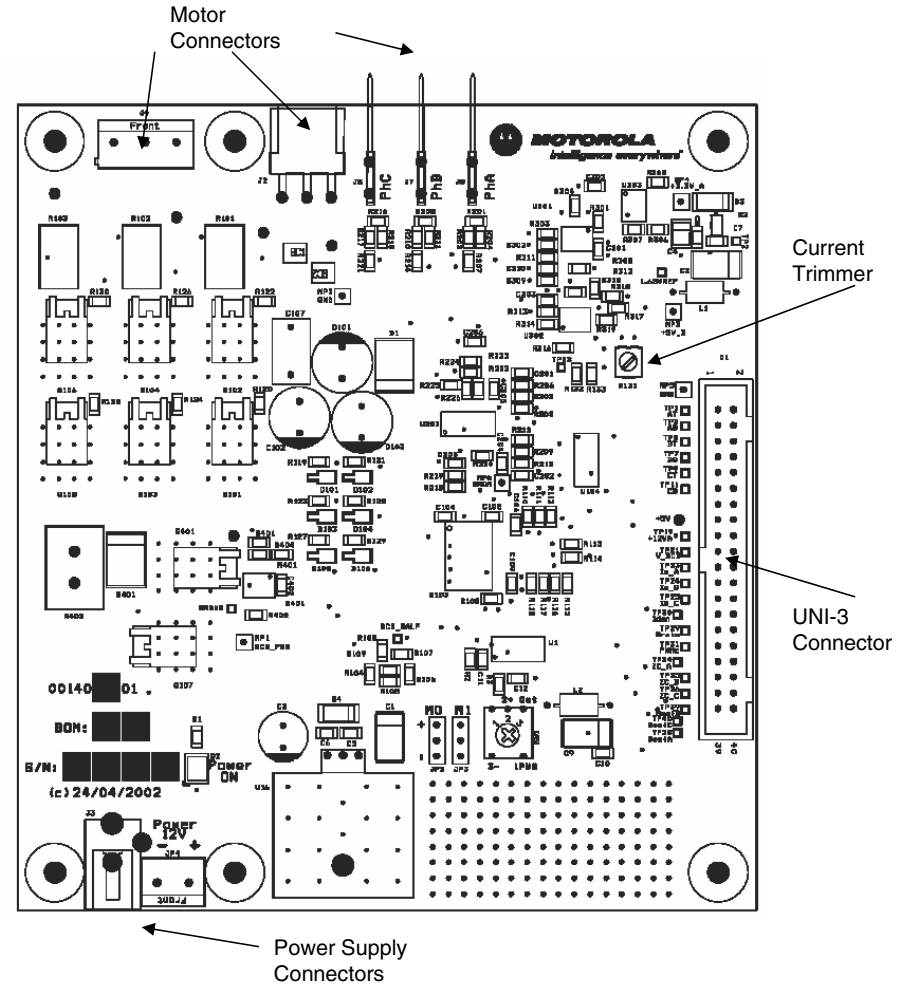


Figure 4-4. EVM33395 Evaluation Motor Board View

Input connections are made via a 40-pin ribbon cable connector J1. Power connections to the motor are made on one of the output connectors J2 or J4 or FASTON type (J5 J7 J8). Phase A (J8), phase B (J7), and phase C (J5) are labeled on the board, the phase pin order for all three connector types is identical. Power requirements are met by a single external 12 VDC power supply. Two connectors, labeled J3 and JP4, are provided for the 12 VDC power supply; they are located on the front edge of the board. Power is supplied to one or the other, but not both.

Table 4-1. Electrical Characteristics of EVM33395 Board

Characteristic	Symbol	Min	Typ	Max	Units
Power supply voltage	Vdc	10.2	12	16	V
Quiescent current	I_{CC}	—	70	—	mA
High state logic 1 input voltage	V_{IH}	2.4	3.3 or 5	7	V
Low state logic 0 input voltage	V_{IL}	—	< 0.4	0.8	V
Input resistance	R_{In}	—	10	—	kΩ
Analog output range	V_{Out}	0	—	3.3	V
Phase current sense voltage	I_{Sense}	—	172	—	mV/A
Bus voltage sense voltage	V_{Bus}	—	206	—	mV/V
Power MOSFET On resistance	$R_{DS(On)}$	—	10	16	mΩ
RMS output current	I_M	—	—	10	A
Total power dissipation	P_{diss}	—	—	18	W

4.2.2.2 Micro Power Board

Freescale's embedded motion control series three-phase micro power stage is a 12 V, 3 A, surface-mount power stage. In combination with one of the embedded motion control series control boards, it provides a software development platform that allows algorithms to be written and tested without the need to design and build a power stage. It supports algorithms that use Hall sensors and back-EMF signals for sensorless control.

The three-phase micro power stage does not have any over-current protection that is independent of the control board, so some care in its setup and use is required if a lower impedance motor is used. With the motor that is supplied in the kit, the power output stage will withstand a full stall condition without the need for over-current protection. Current measuring circuitry is set up for 8.25 A, full scale. In a 25°C ambient operation at up to 3 A continuous RMS output current is within the board's thermal limits.

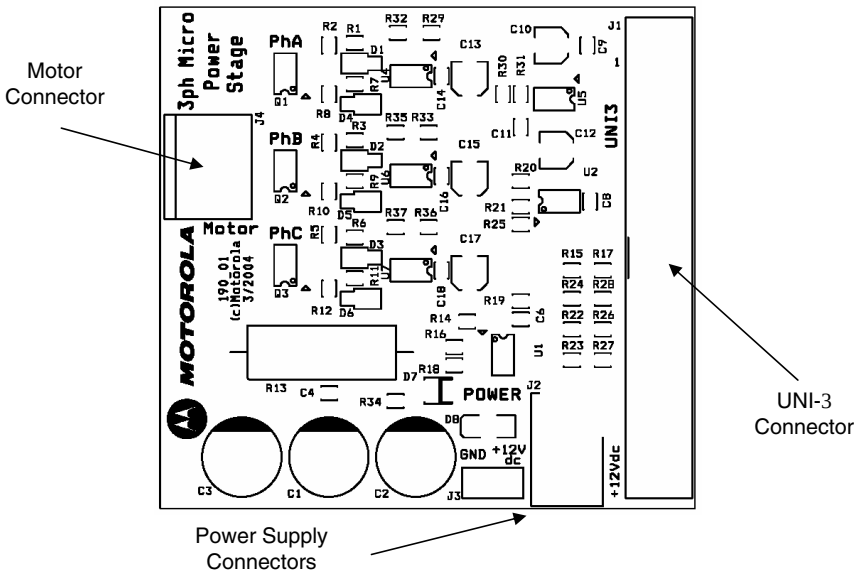


Figure 4-5. Micro Power Stage Board

Input connections are made via 40-pin ribbon cable connector J1. Power connections to the motor are made on output connector J4. Phase A, B, and C are labeled on the board. Power requirements are met with a single external 12 VDC, 4 A power supply. Two connectors, labeled J2 and J3, are provided for the 12 VDC power supply. While the former is used for 12 VDC input, the latter is used for 12 VDC output for powering the controller board. Both are located in a corner of the board.

Table 4-2. Electrical Characteristics of the Micro Power Stage Board

Characteristic	Symbol	Min	Typ	Max	Units
Power supply voltage	Vdc	10	12	16	V
Quiescent current	I_{CC}	—	18	—	mA
Minimum logic 1 input voltage	V_{IH}	2.4	—	—	V
Maximum logic 0 input voltage	V_{IL}	—	—	0.8	V
Input resistance	R_{In}	—	10	—	kΩ
Analog output range	V_{Out}	0	—	3.3	V
Bus current sense voltage	I_{Sense}	—	200	—	mV/A
Bus voltage sense voltage	V_{Bus}	—	202	—	mV/V
Power MOSFET On resistance (top switch)	$R_{DS(On)}$	—	60	70	mΩ
Power MOSFET On resistance (bottom switch)	$R_{DS(On)}$	—	170	200	mΩ
Continuous output current	I_D	—	—	3	A
Pulsed output current	I_{DM}	—	—	15	A
Total power dissipation	P_{diss}	—	—	2	W
Deadtime	t_{off}	400	—	—	ns

4.2.3 Motors

The motors specified below sections are used in the application. Both motors can be used with both power stages described above. Other motors can also be adapted for the application, by defining and changing the motor related parameters (see Chapter 7 Tuning). Detailed motor specifications are shown below.

4.2.3.1 IB23811 (produced by MCG)

Table 4-3. Electrical Characteristics of MCG IB23811 Motor

Characteristic	Symbol	Min	Typ	Max	Units
Reference winding voltage	V_t	—	—	170	V
Speed @ V_t		—	—	6000	RPM
Torque constant	K_t	—	0.0840	—	Nm/A
		—	11.90	—	oz-in/A
Voltage constant	K_e	—	8.8	—	V/kRPM
Terminal resistance	R_t	0.13	—	0.18	Ω
Winding inductance	L	—	6.8	—	mH
Continuous current	I_{cs}	—	—	1.8	A
Number of pole-pairs	J_m	—	2	—	—
Temperature rating		–10	—	80	°C
		14	—	176	°F

4.2.3.2 N2311 (produced by Pittman)

Table 4-4. Electrical Characteristics of Pittman N2311 Motor

Characteristic	Symbol	Min	Typ	Max	Units
Reference winding voltage	V_t	—	—	9.6	V
Speed @ V_t		—	—	12000	RPM
Torque constant	K_t	—	0.007	—	Nm/A
		—	1.082	—	oz-in/A
Voltage constant	K_e	—	0.8	—	V/kRPM
Terminal resistance	R_t	0.13	—	0.18	Ω
Winding inductance	L	—	2.9	—	mH
Continuous current	I_{cs}	—	—	9.96	A
No load current @ V_t	I_{ps}	—	1.20	—	A
Number of pole-pairs	J_m	—	4	—	—
Temperature rating		–10	—	80	°C
		14	—	176	°F

Chapter 5
Software Design

5.1 Introduction

This section describes the design of the drive's software blocks. The software description comprises the following two topics.

- Main Software Flow Chart
- Data Flow

5.2 Main Software Flow Chart

The main software flow chart incorporates the main routine entered from the reset and interrupt states. The Main routine includes the initialization of the hybrid controller and the main loop. There are two initialization routines, one belonging to Processor Expert, and the other to the application program. Inside the application initialization, also used power stage and motor parameters are loaded.

The main loop incorporates the application state machine — the highest software level which provides settings for other software levels (BLDC motor commutation control, zero crossing offset control, speed control, alignment current control, current limitation control). The inputs of the application state machine are the RUN/STOP switch state, the required speed omega, and the drive fault status. The required mechanical speed can be set from the FreeMaster or manually by the UP and DOWN push-buttons.

The RUN/STOP switch is checked to provide an input for the application state machine (application run mode or stop mode).

The main software flow chart is given in Figure 5-1.

Interrupt service subroutines perform the main commutation tasks:

- Commutation timing — **IsrCommutation**
  - Uses quad timer 0
  - Calling period depends on motor speed. At 2000 RPM for a 2-pole-pair motor, the commutation period is 2500 µs.
  - Performs commutation and commutation related actions, if required. It cooperates with drivers inside **BldcZC.c**.
- Speed and alignment current control — **IsrSpeedCurrentControl**

  Uses quad timer 2

  Calling period is 200 µs.

  Includes all PI controllers required for speed and alignment current regulations. To limit the DC-bus current, it runs a current limiting PI controller in parallel with these PI controllers. It also provides alignment timing.
- ADC sampling — **IsrADCEndofScan**

  The calling period is defined by the PWM frequency and the PWM reload prescaler. In this application, the calling period is 50 µs.

Because this interrupt has the highest frequency, FreeMaster recorder is also put inside. It also creates a timebase for button processing.

- ADC limit and zero crossing detection — **IsrADCLimit**

  The calling period depends on the motor speed. At 2000 RPM for a 2-pole-pair motor, the zero crossing period is 2500 µs.

  The zero crossing detection part detects zero crossings and performs actions related with zero crossing detection. It cooperates with drivers inside BldcZC.c.

- **IsrPWMReload**

  The calling period is defined by the PWM frequency and the PWM reload prescaler. In this application, the calling period is 50 µs.

  It reloads the PWM duty cycle, calculated by the speed PI controller. This interrupt is included for future expansion.

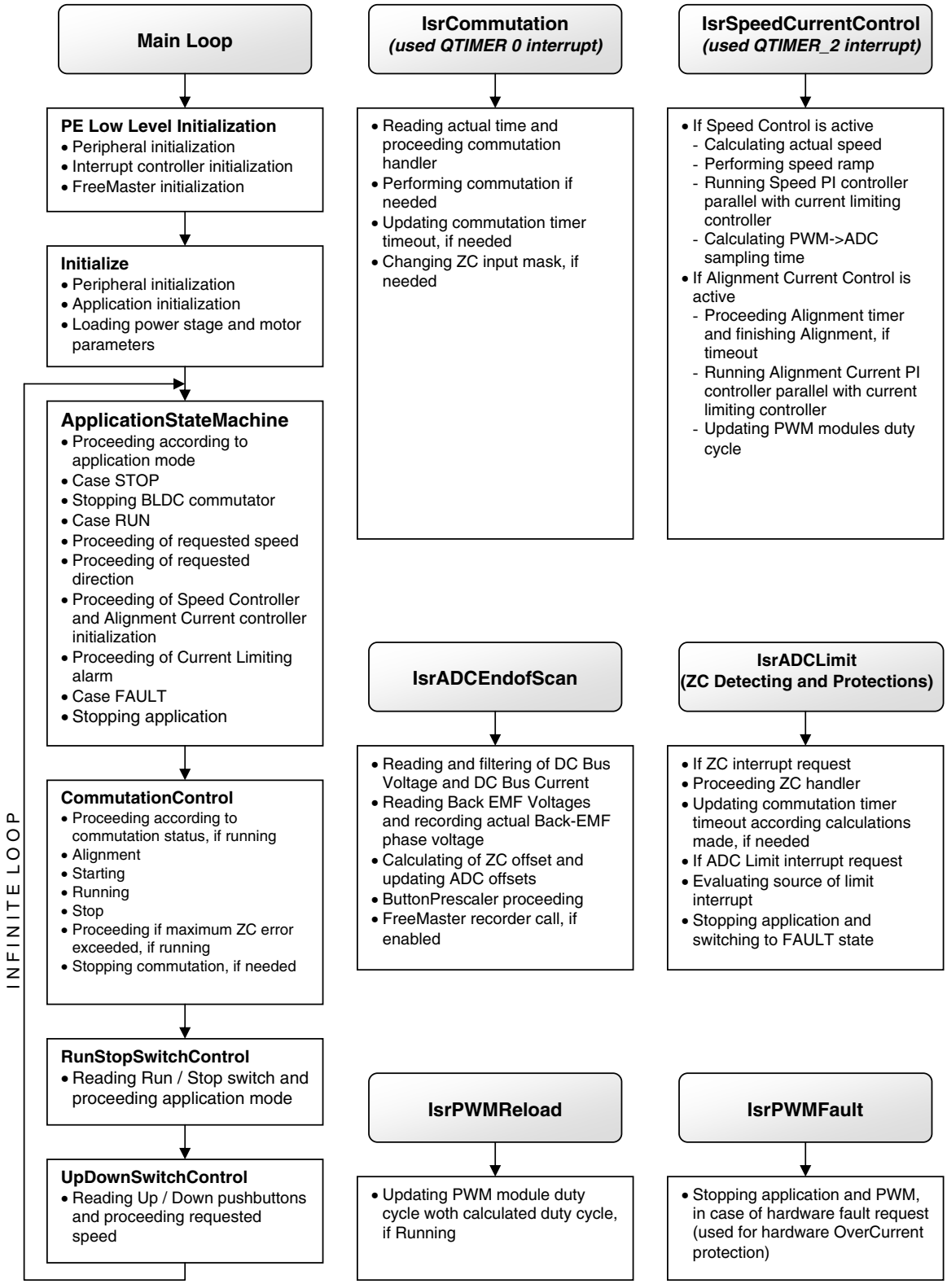


Figure 5-1. Main Software Flow Chart

5.3 Data Flow

The control algorithm for the BLDC motor sensorless control drive is described in the data flow charts shown in Figure 5-2. (The variables and constants described should be clear from their names.)

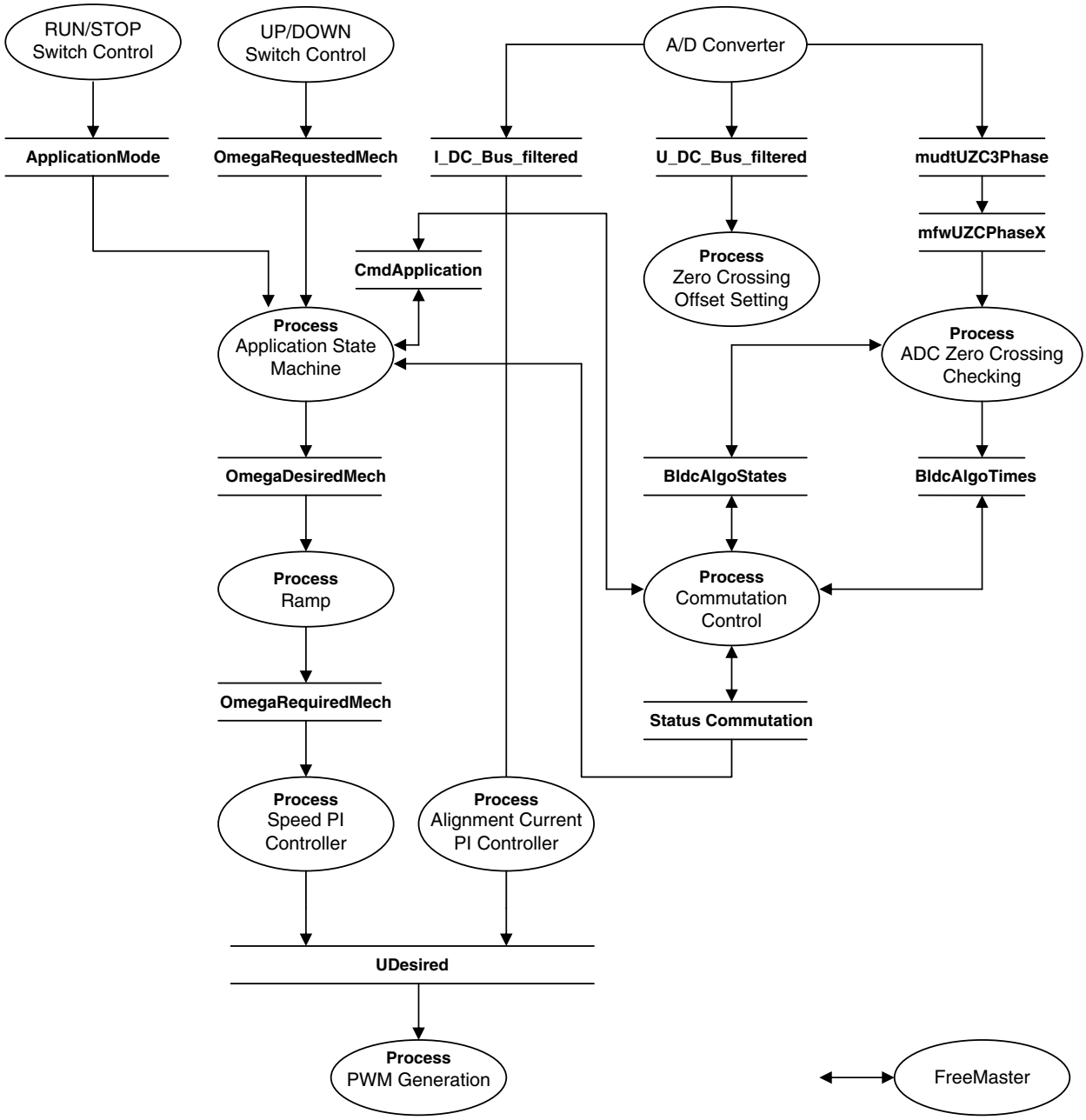


Figure 5-2. Data Flow Chart

5.3.1 Application State Machine Process

This process controls the application states by status and command words.

- The run and stop states are handled. Based on the state of the application, **Cmd_Application** command (**Cmd**) request (**Rq**) flags are processed.
- The speed requested by the user, via the UP/DOWN buttons or via FreeMaster, is achieved with minimum control and sign processing.
- Alignment commutation status is initiated and finished, if required.
- PWM duty cycle values and PWM outputs are modified to a no PWM state, if the application is in the Stop state or the requested speed is below a minimum value.

5.3.2 Commutation Control Process

This process controls the sensorless BLDC motor commutations, as explained in 5.1 Introduction. The process communicates mainly over the **BldcAlgoStates** and **BldcAlgoTimes** structures.

- All commutation states (stop, alignment, starting, run) are handled separately.
- Commutation and other commutation timing activities are performed. (**IsrCommutation**)

5.3.3 ADC Zero Crossing Checking Process

This process is based on the ADC zero crossing feature. When the non fed phase branch voltage changes sign when compared with the previous conversion result, a zero crossing interrupt is initiated.

- The instant of zero crossing interrupt is saved through the BldcAlgoTimes structure.

5.3.4 Zero Crossing Offset Setting Process

To ensure correct behavior of the ADC zero crossing checking, the ADC zero crossing offset registers must be set such that the zero back-EMF voltage is converted to a zero ADC value. Because phase voltages vary between 0 V and the DC-bus voltage value, ideally, the zero crossing offset registers must be set to half the DC-bus voltage value.

This process is performed periodically inside **IsrSpeedCurrentControl**.

- The ADC offset registers for all free phase voltages are set to **U_Dc_Bus_Half**.

5.3.5 Speed PI Controller Process

The general principle of the speed PI control loop is illustrated in Figure 5-3.

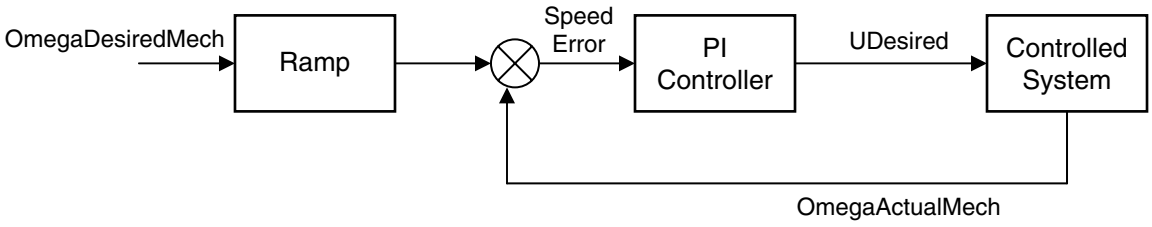


Figure 5-3. Closed Loop Control System

The closed-loop speed control is characterized by the feedback of the actual motor speed. This information is compared with the reference set point and an error signal is generated. The magnitude and

polarity of the error signal corresponds to the difference between the actual and desired speed. Based on the speed error, the PI controller generates the corrected motor voltage to compensate for the error.

- The speed PI controller works with a constant execution (sampling) period. This is achieved by calling from a periodic interrupt. The constant **PER_SPEED_SAMPLE_S** is defining the calling period of the speed controller
- The speed PI controller proportional and integral constants were set experimentally.
- The speed PI controller's output is limited by another controller, the current limitation PI controller.

5.3.6 Alignment Current PI Controller Process

The process is similar to the speed controller. The **I_Dc_Bus** current is controlled based on the **U_Dc_Bus_Desired** reference current. The current controller is processed only during the alignment state, to keep the stator flux constant.

- The alignment current PI controller works with the constant execution (sampling) period of the **IsrSpeedCurrentControl** interrupt service routine.
- The alignment current PI controller proportional and integral constants were set experimentally.
- The alignment current PI controller's output is limited by another controller, current limitation PI controller.

5.3.7 Current Limitation PI Controller Process

This PI controller is used to limit DC-bus current by software. During the alignment state the output of the alignment current PI controller and during the running state the output of the speed PI controller is limited by this PI controller.

- The current limitation PI controller works with the same constant execution (sampling) period of the speed PI controller or the alignment current PI controller, depending on the commutation state.
- The current limitation PI controller proportional and integral constants were set experimentally.

5.3.8 PWM Generation Process

The PWM generation process creates the following:

- BLDC motor commutation pattern as described in 2.1 The Brushless DC Motor
- Required duty cycle

5.3.9 Fault Control Process

The fault control process is used for drive protection. **DriveFaultStatus** is passed to the PWM generation process and to the application state machine process to disable the PWMs and to control the application accordingly.

Chapter 6
Application Setup

6.1 Application Description

As described earlier, the BLDC sensorless application, which is targeted at the MC56F8013, can run on two power stages and two motors. For setting up the application, only the combination with one power stage and one motor is given. All other setup combinations are similar. See Chapter 7 Tuning.

The concept of the BLDC sensorless drive incorporates the following hardware components:

- MC56F8013 EVB
- Micro power stage board
- IB23811 motor set

Table 6-1. Application Specifications

Motor Characteristics	
Motor type	4-pole, 3-phase, star connected, BLDC motor
Speed range	6000 RPM (at 170 VDC)
Maximum electrical power	150 W
Maximum phase voltage	3 * 170 V
Maximum phase current	1.8 A
Drive Characteristics	
Speed range	< 2000 RPM
Input voltage	12 VDC
Maximum DC-bus voltage	16 VDC
Maximum DC-bus current	4 A
Control algorithm	Closed-loop speed control
Switching frequency	20 kHz

The following quantities are measured by the application:

- DC-bus voltage
- DC-bus current
- Phase voltages (back-EMF voltages)
- Rotor speed (measured by back-EMF sensing, without sensor)

The following faults and alarms are used to protect the drive:

- DC over-voltage (by ADC high limit detection; stops drive)
- DC under-voltage (by ADC low limit detection; stops drive)
- DC over-current (by external comparator; stops drive)
- DC current limiting (by DC-bus current measurement)

6.1.1 Control Process

The RUN/STOP switch and UP/DOWN buttons are continuously monitored over the GPIO interface. If the RUN/STOP switch is in the RUN position, and if there is no fault condition in the system, the application switches to run mode and waits for speed setup. After the speed setup is changed from zero to any permitted value (using the UP/DOWN push buttons or the FreeMaster interface), the commutation sequence starts. For this, first, the application performs an alignment, to adjust the rotor position for the rest of commutation. During alignment, the DC-bus current is kept constant. This is achieved by the current controller for alignment, where the DC-bus current is measured and fed into the current controller for alignment. After alignment, the application performs a calibration, to compensate for the differences between back-EMF sensing networks. After alignment is finished, the application switches to the starting and running commutation states.

A detailed view of MC56F8013 EVB for this control process is given in Figure 6-1.

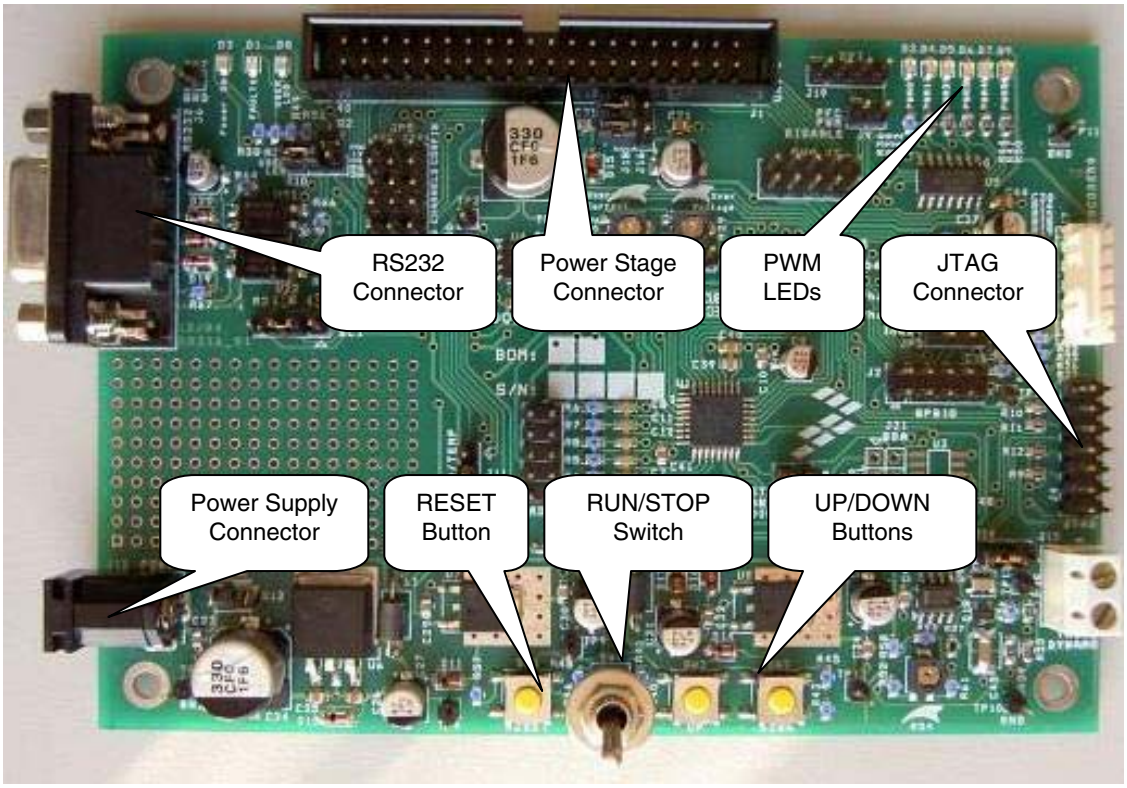


Figure 6-1. MC56F8013 Evaluation Board

6.1.2 Drive Protection

The DC-bus voltage is measured during the control process. It is used to protect the drive from DC over-voltage and DC under-voltage, with the help of the ADC. The ADC generates an interrupt if limits are exceeded, in which case, PWM outputs are disabled and the drive is stopped. Similarly, the DC-bus current is also measured for current limitation. If the DC-bus current is higher that the desired current, the PWM duty cycles are limited, to limit the DC-bus current. This is a software limitation only. Beyond this, the DC-bus current is compared on an external comparator and the output of the comparator is fed to the Fault input of the PWM module. If the DC-bus current exceeds the value set by the external trimpot, a

fault signal is generated to the hybrid controller and the drive is stopped. All fault and alarm messages are displayed on the FreeMaster screen.

After the drive has entered the Fault state, it waits until the fault is reset. The fault can be reset in two ways:

- by clicking the RESET button on the FreeMaster screen
- by switching the RUN/STOP switch on MC56F8013 board to the OFF position

6.1.3 FreeMaster Interface

The FreeMaster software was designed to provide a debugging, diagnostic, and demonstration tool for the development of algorithms and applications. Moreover, is 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. It consists of a software component running on a personal computer (PC) and another running on the target hybrid controller, connected by an RS-232 serial port. A small program resident in the hybrid controller communicates with FreeMaster software to parse commands, return status information to the PC, and process control information from the PC. FreeMaster software executing on the PC uses Microsoft Internet Explorer as the user interface.

To enable the FreeMaster software operation on the hybrid controller target board application, add PC_Master bean to the application. PC_Master bean is located under CPU External Devices > Display in the bean selector of Processor Expert.

The FreeMaster bean automatically includes the SCI driver and installs all necessary services. There is no need to install the SCI driver, because PC_Master bean encapsulates its own SCI driver.

The default communication speed for the SCI is 9600 baud; this is set automatically by the FreeMaster software driver.

Included in the FreeMaster software is a recorder that can sample the application variables at a specified sample rate. Samples are stored to a buffer, and read by the PC via an RS-232 serial port. The sampled data can be displayed in a graph or stored. The recorder behaves like a simple on-chip oscilloscope with trigger and pre-trigger capabilities. The size of the recorder buffer and the FreeMaster recorder timebase can be defined in the PC_Master bean.

The recorder routine must be called periodically in the loop in which samples are required. The following line must be added to the loop code.

```
pcmasterdrvRecorder(); /* FreeMaster recorder routine call */
```

In this application, the FreeMaster recorder is called from the **ADCEndofScan** interrupt, because it is the most frequently, periodically called routine in the application.

A detailed description of the FreeMaster software is provided in the FreeMaster Software User Manual.

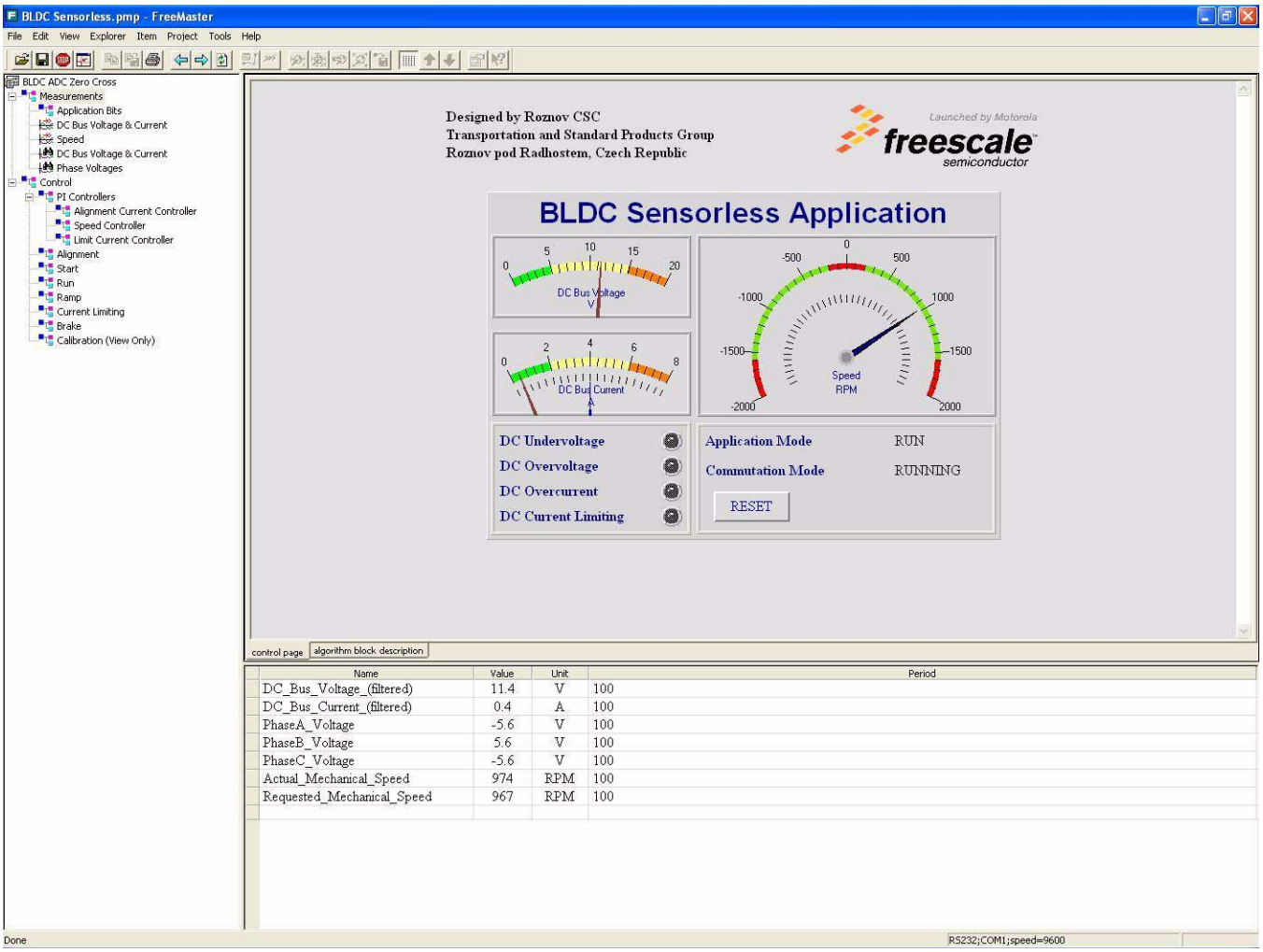


Figure 6-2. FreeMaster Screen

The FreeMaster software supports the following control actions.

- Sets the required speed of the motor
- Sets the maximum DC-bus current (current limitation level)
- Adjusts all PI controllers, motor, commutation parameters
- Resets fault status
- Performs speed stimulus/profile

The FreeMaster software displays the following information:

- Required speed of the motor
- Actual speed of the motor
- Application status — stop/run/fault
- Commutation status — stop/alignment/starting/running
- DC-bus voltage and DC-bus current (filtered)
- Phase voltages/back-EMF voltages
- Fault status — No fault/DC over-voltage/DC under-voltage/DC over-current
- Alarm status — current limitation
- Scopes for DC-bus voltage, DC-bus current and speed
- Recorders for DC-bus voltage, DC-bus current, and back-EMF voltages

Start the FreeMaster software window's application, **BLDC Sensorless.pmp**. Figure 6-2 illustrates the FreeMaster software control window after this project has been launched.

NOTE

If the FreeMaster software project (.pmp file) is unable to control the application, it is possible that the wrong load map (.elf file) has been selected. FreeMaster software uses the load map to determine addresses for global variables being monitored. After the FreeMaster software project has been launched, this option may be selected in the FreeMaster software window under Project/Select Other Map FileReload.

6.2 Application Setup

Figure 6-3 shows the hardware setup for the 3-phase BLDC sensorless motor control application. The MC56F8013 EVB, Micro Power stage and N2311 motor, JTAG command converter, and power supply are shown.

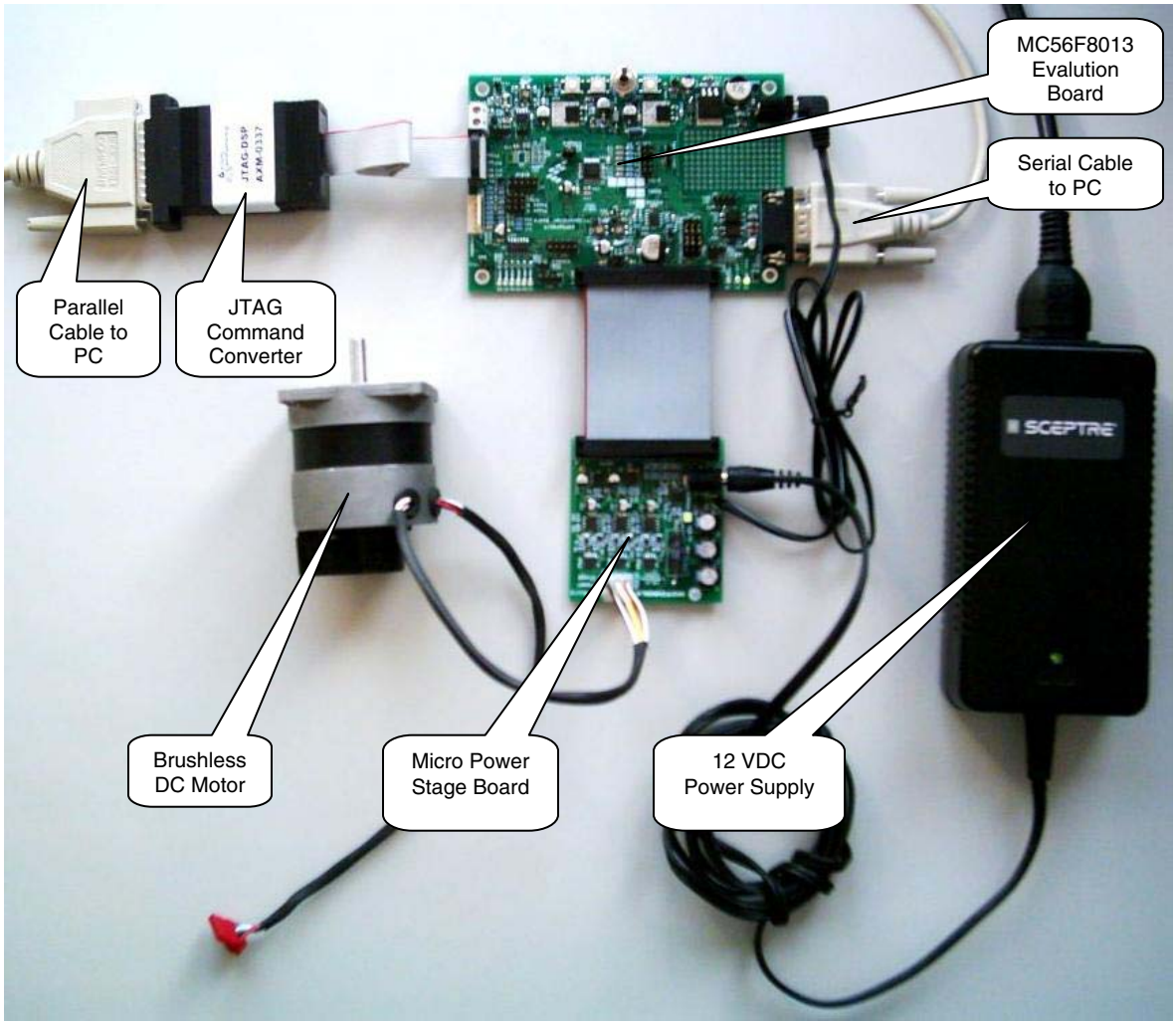


Figure 6-3. Setup of the 3-phase BLDC Sensorless Control Application

The system consists of the following components.

- MC56F8013 EVB
- JTAG command converter — converts PC parallel port command to JTAG port
- 3-phase, low-voltage (12 VDC) Micro Power stage board
- BLDC motor type IB23811 MCG
- Power supply — 12 VDC, 4 A
- Parallel cable — required for the Metrowerks CodeWarrior debugging and software loading.
- Serial cable — required for the FreeMaster software debugging tool only

6.2.1 MC56F8013 EVB Setup

6.2.1.1 Jumper settings

To execute the 3-phase BLDC motor sensorless control application, the MC56F8013 EVB requires the jumper settings to be set as shown in Figure 6-4 and Table 6-2.

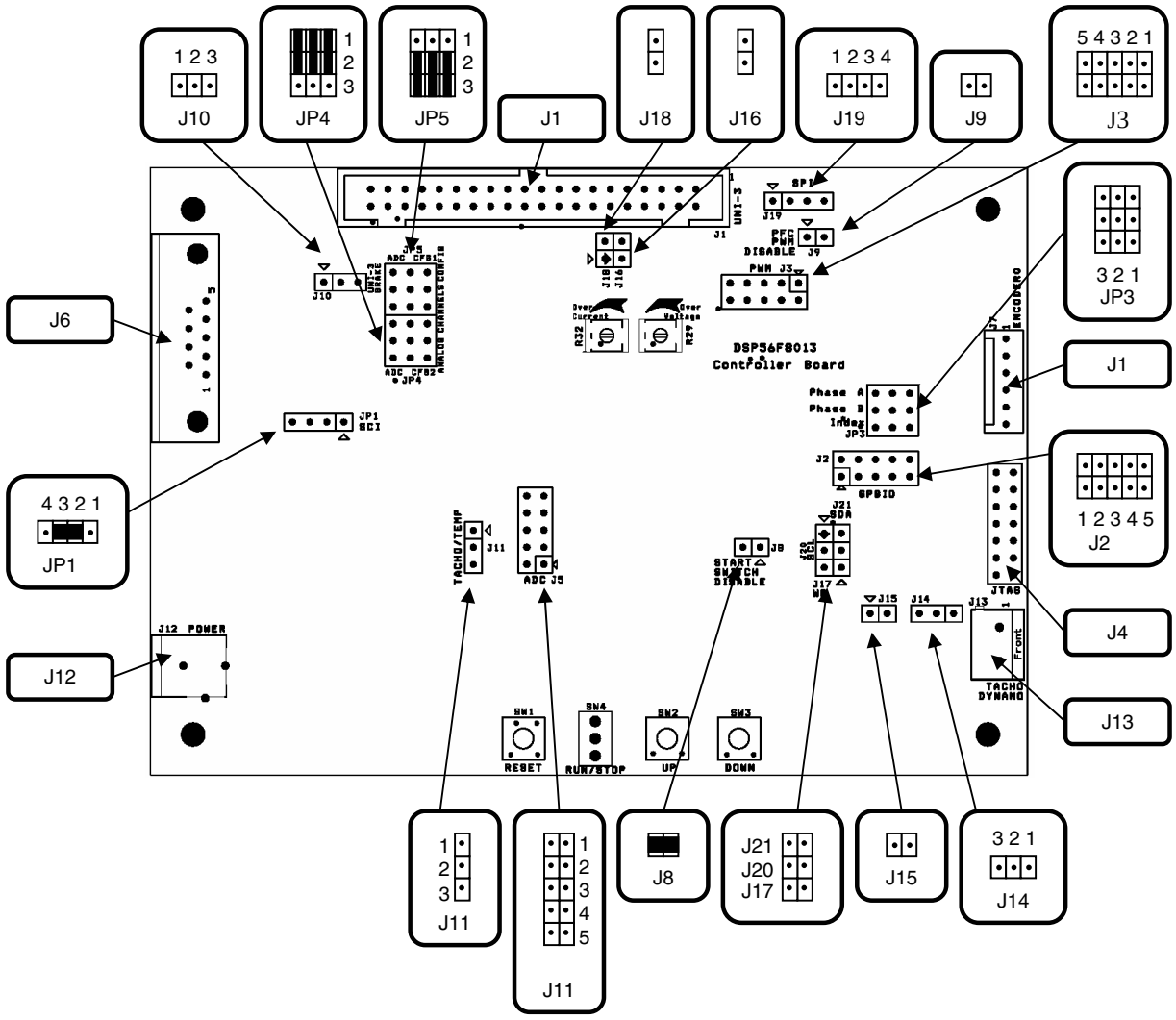


Figure 6-4. MC56F8013 EVB Connector and Jumper Reference

Table 6-2. MC56F8013 EVB Connectors and Jumpers Settings

Jumper Group	Comment	Connections
JP1	RxD of hybrid controller to RxD of RS-232 driver interface	2–3
JP3	Back-EMF zero crossing or encoder signals selection	NC
JP4	ADC B configuration/DC-bus voltage and DC-bus current signals to ADC B connection	1–2, 4–5, 7–8
JP5	ADC A configuration/back-EMF signals to ADC A connection	2–3, 5–6, 8–9
J1	UNI-3 connector	N/A
J2	GPIO B expansion	N/A
J3	PWM expansion	N/A
J4	JTAG expansion	N/A
J5	ADC expansion	N/A
J6	RS-232 connector	N/A
J7	Encoder connector	N/A
J8	RUN/STOP switch enable	1–2
J9	PFC PWM disable	NC
J11	Tacho/temp selection	NC
J12	Power jack	N/A
J13	Tacho dynamo connection	NC
J14	Tacho processing selection	NC
J15	Tacho comparator output connection to GPIOB4	NC
J16	+5 V from UNI-3 to on-board +5 V connection	NC
J17	External EEPROM write protect selection	NC
J18	+15 V from UNI-3 to on-board +15 V connection	NC
J19	SPI expansion	N/A
J20	External EEPROM SCL connection	NC
J21	External EEPROM SDA connection	NC

NC — Not connected
N/A — Not applicable

NOTE

If it is intended to run the application with the EVM33395 evaluation motor board, jumpers J16 and J18 must be shorted. This allows the MC56F8013 EVB to be fed from the EVM33395 evaluation motor board.

6.2.1.2 Trimpot Settings

The MC56F8013 EVB contains three trimpots.

- R29 — DC over-voltage hardware fault setpoint

  In this application, the DC over-voltage hardware fault is not used. Therefore, on the micro power stage board, this trimpot (TP3) should be adjusted to 3.3 V to avoid a DC over-voltage fault.
- R32 — DC over-current hardware fault setpoint

  On the micro power stage board, the DC-bus current is sampled at 200 mV/A. This trimpot (TP5) should be adjusted to 2.85 V to have a DC over-current fault at peak 6 A. (1.65 V offset).
- R54 — tacho-dynamo sense threshold setpoint

In this application, the tacho-dynamo section of the MC56F8013 EVB is not used. Therefore, the setting of this trimpot can be ignored.

6.3 Projects Files

The application is created using CodeWarrior 7.1 and Processor Expert 3.95.

The BLDC sensorless application is composed of the following files.

- **...\ bldc_zc_8013 \ bldc_zc_8013.mcp**, application project file
- **...\ bldc_zc_8013 \ bldczcdefines.c**, header file containing setup information
- **...\ bldc_zc_8013 \ bldc_zc_8013.mcp**, application project file
- **...\ bldc_zc_8013 \ code \ bldc_zc_8013.c**, main program
- **...\ bldc_zc_8013 \ code \ smm_pROM_xRAM.cmd**, linker command file
- **...\ bldc_zc_8013 \ bin \ sdm_pROM_xRAM.elf**, executable application file
- **...\ bldc_zc_8013 \ bin \ sdm_pROM_xRAM.elf.xMAP**, MAP file
- **...\ bldc_zc_8013 \ freemaster \ BLDC Sensorless.pmp**, FreeMaster software file

6.4 Application Build and Execute

When building the application, the user can create an application that runs from internal Flash only, because MC56F8013 does not support external RAM.

From the Metrowerks CodeWarrior IDE, the project may be built by executing the Project/Make command, as shown in Figure 6-5. This will build and link the BLDC motor sensorless control application and all required Metrowerks and Processor Expert libraries.

To execute the application, select Project/Debug in the CodeWarrior IDE, followed by the run command. For more help with these commands, refer to the CodeWarrior tutorial documentation in the following file located in the CodeWarrior installation folder:

<...>\CodeWarrior Documentation\PDF\Targeting_DSP56800.pdf

If the Flash target is selected, CodeWarrior will automatically program the internal Flash of the hybrid controller with the executable generated during Build. After Flash has been programmed with the executable, the evaluation module target system may be run in a stand-alone mode from Flash.

When the application is running, move the RUN/STOP switch to the RUN position and set the required speed using the UP/DOWN push buttons. Pressing the UP/DOWN buttons should incrementally increase the motor speed until it reaches maximum speed. If this is successful, the motor will be spinning.

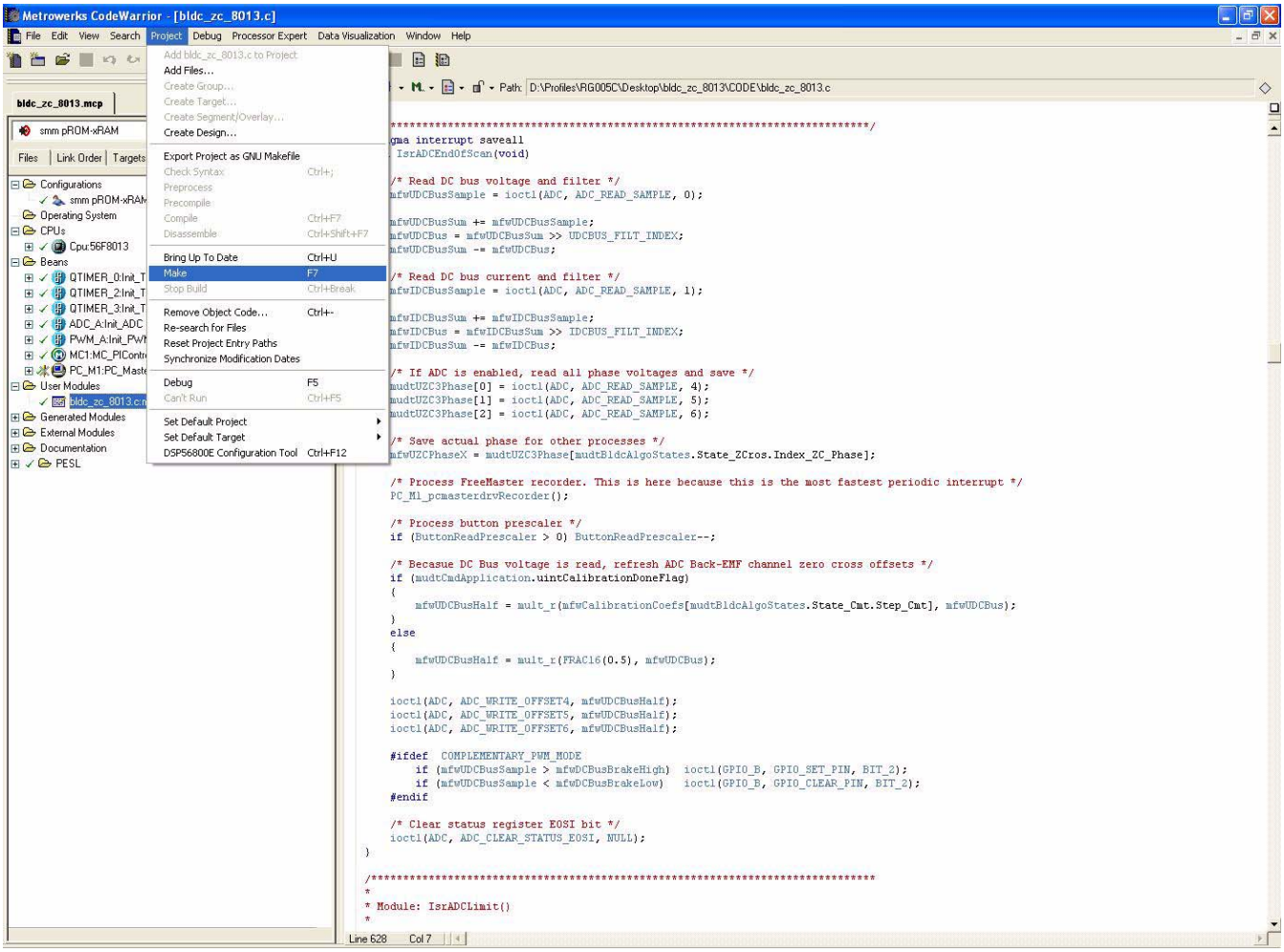


Figure 6-5. Executing Make Command

6.5 Hybrid Controller Usage

Table 6-3 shows how much memory is used to run the BLDC sensorless motor control application. Stack size is set to 512 words. Heap size is set to 256 words.

Table 6-3. RAM and Flash Memory Usage for Processor Expert 3.58 and CodeWarrior 7.1(1)

MC56F8013 Memory (16-bit words)		Used Application + Stack with FreeMaster(2)	Used Application + Stack without FreeMaster
Program Flash	8K	8109	4106
Program RAM	2K	78	64
Data RAM	2K	1904	984

NOTES:
1. With optimization level 2.
2. FreeMaster data, recorder, and command buffers occupy 809 words.

Chapter 7
Tuning

7.1 Introduction

The BLDC sensorless application is designed to cover a wide range of BLDC motor sensorless control concepts. Thanks to its parametric structure, it is possible to change the behavior of the application; it can be adjusted and tuned for different processors, power stages, and motors. In addition, the FreeMaster interface makes this tuning process easier.

7.2 Classification

The software is tuned for two power stages (EVM33395, Micro Power) and two motors (MCG IB2381, Pittman N2311) as described in application note. It can of course be used for other motors, but the software parameters must be set accordingly.

The software parameters are divided three main groups:

- Group 1 — Application Related Parameters

  These parameters depend on the hybrid controller used, and the application's performance. They include the calling frequency of the speed controller, and the DC-bus voltage reading filtering constants, and other similar parameters. These parameters are mainly defined in **bldczcdefines.h**.

  There are also some mask definitions related to the hardware used. These masks are defined mainly in **bldcdrv.h** and used in **bldcdrv.c**.
- Group 2 — Power Stage Related Parameters

  Parameters included in this group have no relation to the motor. When these parameters are set properly, several motors with the same power stage can be driven, by tuning only parameters in group 1.

  These parameters are defined mainly in **bldczcdefines.h**.

  In this application, these parameters are called **EVM33395_xxx** and **MP_xxx**, where **xxx** is the parameter name.

  To use the desired set of parameters, in the main software there are two functions defined.
  - **EVM33395BoardSettings()**
  - **MPBoardSettings()**
- Group 3 — Motor Related Parameters

  These parameters are not related to the power stage. When these parameters are set properly, other power stages can be used to drive the same motor, only by tuning parameters in Group 1.

  These parameters are defined mainly in **bldczcdefines.h**.

  In this application, these parameters are called **MOTORA_xxx** and **MOTORB_xxx**, where **xxx** is the parameter name.

To use the desired set of parameters, in the main software there are two functions defined:
– **MOTORASettings()**, IB23811 motor
– **MOTORBSettings()**, N2311 Motor

In the parameters tables given below, all parameters range and default values are filled according to the application setup using the Micro Power Stage Board and MotorA (IB23811 MCG), which is described in Chapter 6 Application Setup.

7.3 Application Related Parameters

7.3.1 Parameters Defined in bldczcdefines.h

7.3.1.1 Parameter Summary

Some parameters in Group 1 are listed and explained in Table 7-1.

Table 7-1. Application Related Parameter Summary

Name	Description	Units	Range	Default	FreeMaster Location
UDCBUS_FILT_INDEX	Filtering constant of ADC samples for DC-bus voltage.	N/A	0–8(1)	4	N/A
IDCBUS_FILT_INDEX	Filtering constant of ADC samples for DC-bus current.	N/A	4–8(1)	6	N/A
NO_INIT_DC_BUS_SAMPLES	Number of samples to measure and average DC-bus voltage, after ADC is first initialized.	N/A	1–64(1)	16	N/A
PER_SPEED_SAMPLE_S	Execution (sampling) period of Speed Controller.	Seconds	0.0002–0.0020	0.001	Control > PI Controllers > Speed Controller
OMEGA_INCREMENT_SYSU	Increment/decrement step of desired speed.	Fraction	1–8192(1)	2048	N/A
RAMP_INCREMENT_UP	Increment step for acceleration ramp of speed setting.	Fraction	1–32767	32	Control > Ramp
RAMP_INCREMENT_DOWN	Increment step for deceleration ramp of speed setting.	Fraction	1–32767	32	Control > Ramp

NOTES:
1. Higher values are theoretically possible, but are not recommended.

7.3.1.2 UDCBUS_FILT_INDEX

Filtering constant of ADC samples for DC-bus voltage.

DC-bus voltage is sampled using the ADC. Because samples received from the ADC may include noise due to the switching mode application, these samples are applied to an exponential filter as follows (inside the **ADCEndOfScan** interrupt).

```
mfwUDCBusSum += mfwUDCBusSample;
mfwUDCBus = mfwUDCBusSum >> UDCBUS_FILT_INDEX;
mfwUDCBusSum -= mfwUDCBus;
```

In this filtering method, the output of the filter approximates to the input filter, exponentially, which eliminates noise very effectively. As can be seen, higher **UDCBUS_FILT_INDEX** values will perform better filtering, but will also cause delay and phase shift.

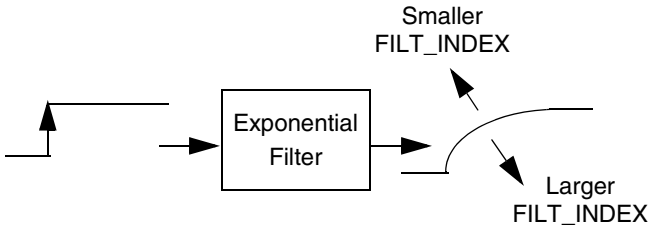


Figure 7-1. Step Response of Exponential Filter

7.3.1.3 IDCBUS_FILT_INDEX

Filtering constant of ADC samples for DC-bus current.

The DC-bus current is sampled using the ADC. Because samples received from the ADC may include noise due to the switching mode application, these samples are applied to an exponential filter as follows (inside the **ADCEndOfScan** interrupt)

```
mfwIDCBusSum += mfwIDCBusSample;
mfwIDCBus = mfwIDCBusSum >> IDCBUS_FILT_INDEX;
mfwIDCBusSum -= mfwIDCBus;
```

In this filtering method, the output of the filter approximates to the input filter exponentially, which eliminates noise very effectively. As can be seen, higher **IDCBUS_FILT_INDEX** values will perform better filtering, but will also cause delay and phase shift.

7.3.1.4 NO_INIT_DC_BUS_SAMPLES

Number of samples to measure and average DC-bus voltage, after ADC is first initialized.

After ADC is initialized, this number of samples is taken, summed, and then divided by this value. The filtered DC-bus voltage value is used to set the ADC offset register for the first time.

See the following code (in **InitADCMeasurements**).

```
for (wwLoopCtr = 1; wwLoopCtr <= NO_INIT_DC_BUS_SAMPLES; wwLoopCtr++)
{
       /* Start ADC Conversion */
              ioctl(ADC, ADC_START, NULL);

       /* Wait until ADC Conversion ends */
              while (!ioctl(ADC, ADC_GET_STATUS_EOSI, NULL));

       /* Read result and add to cumulative sum */
              mfwUDCBusSum += ioctl(ADC, ADC_READ_SAMPLE, 0);
}

mfwUDCBus = mfwUDCBusSum / NO_INIT_DC_BUS_SAMPLES;
```

7.3.1.5 PER_SPEED_SAMPLE_S

Execution (sampling) period of the speed PI controller.

The speed PI controller is called from a periodic interrupt. Inside this periodic interrupt, there is a prescaler assigned for calling speed PI controller. Every time this prescaler gives a timeout signal, the speed PI controller is called.

7.3.1.6 OMEGA_INCREMENT_SYSU

Increment/decrement step of the desired speed.

This value is used as increment/decrement step of the desired speed, when UP/DOWN switches are used to set the speed. It has no effect when setting the speed from FreeMaster.

Note that this value is in fractional format, not absolute revolutions per minute.

7.3.1.7 RAMP_INCREMENT_UP

Increment step for the acceleration ramp of speed setting.

In the BLDC sensorless application a ramp is implemented as follows.

The implemented ramp performs a linear ramp generation determined by input parameters as shown in Figure 7-2

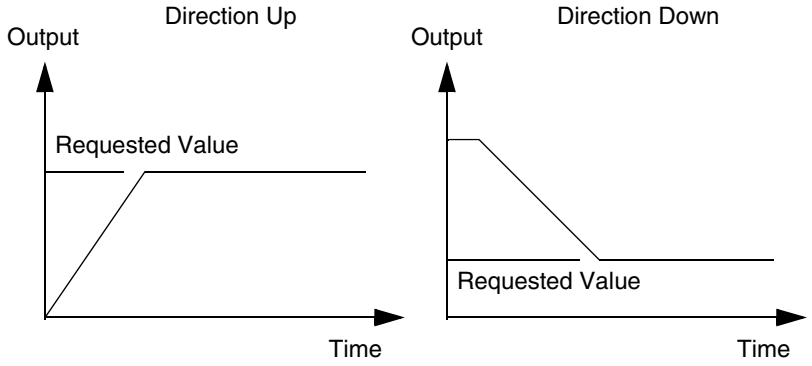


Figure 7-2. Ramp

If **requestedValue** is greater than **actualValue**, the **rampGetValue** function returns **actualValue** + **incrementUp** until the maximum is reached (maximum is **requestedValue**), at which point it will return **requestedValue**.

If **requestedValue** is less than **actualValue**, the **rampGetValue** function returns **actualValue** – **incrementDown** until the minimum is reached, (minimum is **requestedValue**), at which point it will return **requestedValue**.

Note that this value is in fractional format, not absolute revolutions per minute.

7.3.1.8 RAMP_INCREMENT_DOWN

Decrement step for the deceleration ramp of the speed setting.

See 7.3.1.7 RAMP_INCREMENT_UP.

7.3.2 Parameters Defined in bldcdrv.h (Masks)

These mask definitions are related to the hybrid controller and the hardware used. They include mainly ADC masks for zero crossing checking and PWM masks for hard and soft switching patterns. The masks are defined in **bldcdrv.h** are used in arrays in **bldcdrv.c**.

These parameters are not available for changing within FreeMaster.

7.3.2.1 Parameter Summary

Name	Description
ADCZC_PHASE_x_ANY **ADCZC_PHASE_x_POS_NEG** **ADCZC_PHASE_x_NEG_POS**	Mask for ADC zero crossing detection. Here, x corresponds to A, B or C. This mask defines the ADC masks used to mask zero crossing input. Therefore, for correct operation, all masks must be defined according to the ADC zero crossing control register. For example, if ADC channel 6 is used for phase A back-EMF sensing, these masks must be defined as following: ADCZC_PHASE_A_ANY.....0x3000 ADCZC_PHASE_A_POS_NEG...0x1000 ADCZC_PHASE_A_NEG_POS...0x2000
INDEX_XC_PHASE_x	Index for phase selection for phase x.

7.3.3 PWM Mode Selection Definition

The BLDC sensorless application enables one of two PWM modes: complementary mode and independent mode. Each mode is described in Figure 7-3 and Figure 7-4.

These parameters are not available for changing within FreeMaster.

In the application software, some properties are only implemented for complementary PWM mode, where complementary PWM mode allows energy recuperations.

NOTE

*In the application software, use the **#define** directives in the **bldcadczcconfig.h** file to select the PWM operation mode.*

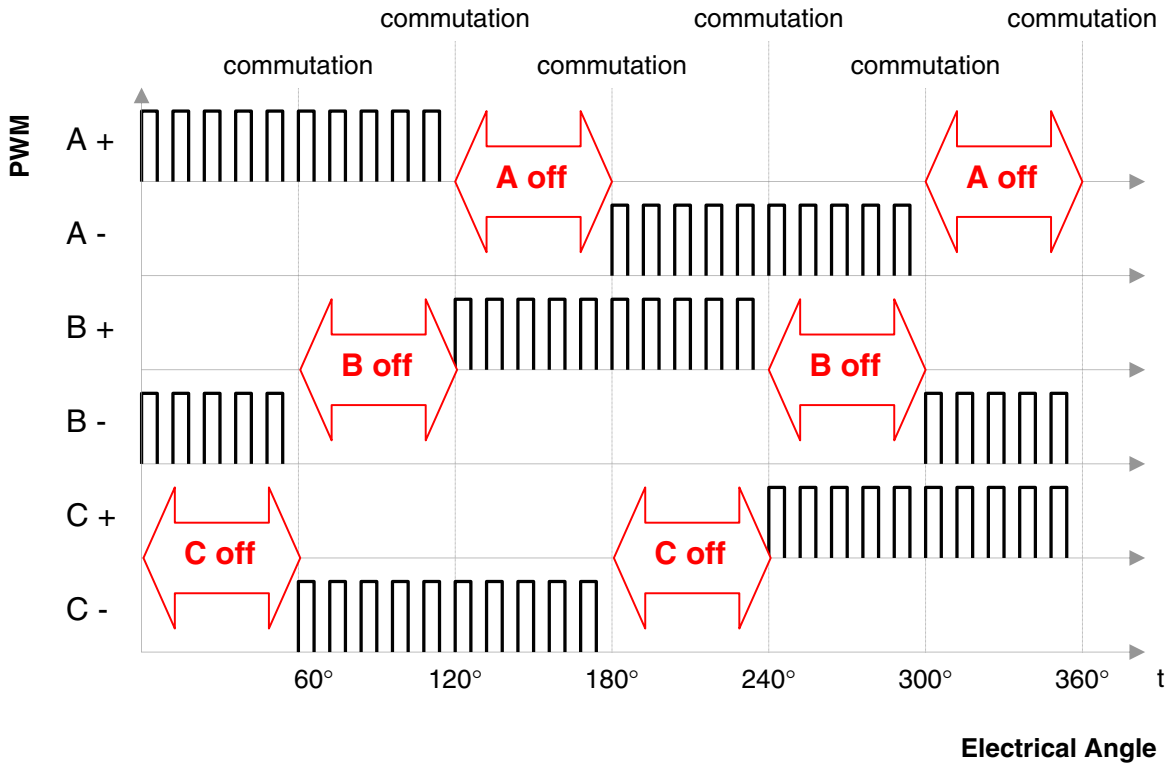


Figure 7-3. Independent PWM Mode

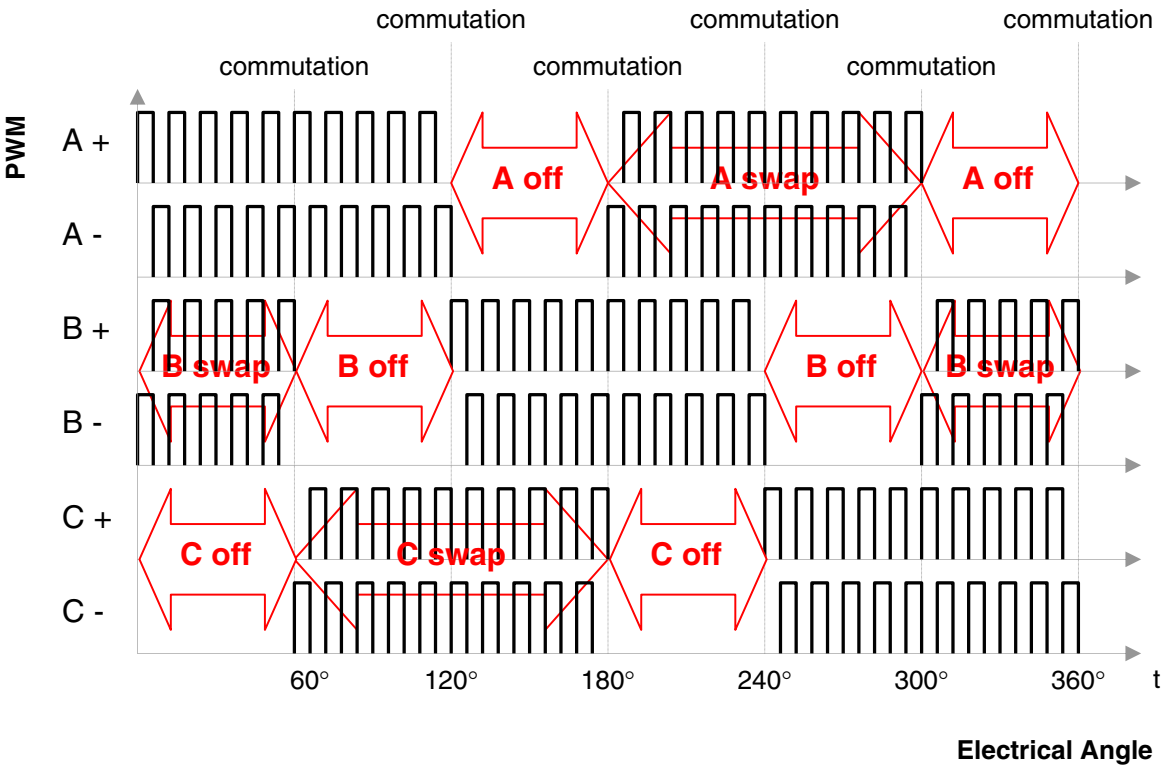


Figure 7-4. Complementary PWM Mode

7.4 Power Stage Related Parameters

Parameters for the desired power stage can be loaded by calling one of the following functions.

- **EVM33395BoardSettings()**
- **MPBoardSettings()**

Unused functions will be dead-stripped by the compiler, so they will not occupy program Flash.

To select the desired power stage board, use **#define** directives in the **bldczcconfig.h** header file.

7.4.1 Parameter Summary

The parameters in Group 2 are listed and explained in Table 7-2.

Table 7-2. Power Stage Related Parameter Summary

Name	Description	Units	Range	Default	FreeMaster Location
APP_VOLT_MAX	DC-bus voltage value, then maximum output is received from ADC.	V	Defined by board components. No range.	16.3	N/A
APP_CUR_MAX	DC-bus current value, then maximum output is received from ADC.	A	Defined by board components. No range.	8.25	N/A
DC_UNDERVOLTAGE	DC under-voltage level of application.	V	0.0–8.0	5.0	N/A
DC_OVERVOLTAGE	DC over-voltage level of application.	V	13.0–16.2	15.0	N/A
DC_OVERCURRENT	DC over-current level of application.	A	0.0–8.25	3.0	N/A

7.4.2 APP_VOLT_MAX

The DC-bus voltage value, when the maximum output is received from the ADC. This value corresponds to the maximum value of the ADC DC-bus voltage reading range.

For example, if the ADC high reference is set to 3.3 V, the ADC converter output will be at a maximum when 3.3 V is received from its input. If the external analog sampling circuit provides 3.3 V to the ADC, when the DC-bus voltage is 16.0 V this parameter should be 16.0.

This parameter should be defined with the decimal point to enable the pre-compiler to perform floating point operations.

7.4.3 APP_CUR_MAX

The DC-bus current value, when the maximum output is received from the ADC. This value corresponds to the maximum value of the ADC DC-bus current reading range.

For example, if the ADC high reference is set to 3.3 V, the ADC output will be at a maximum when 3.3 V is received from its input. If the external analog sampling circuit provides 3.3 V to the ADC, when the DC-bus current is 4.0 A this parameter should be 4.0.

This parameter should be defined with the decimal point to enable the pre-compiler to perform floating point operations.

7.4.4 DC_UNDERVOLTAGE

The DC under-voltage level of the application.

A **DC_Undervoltage** fault is triggered by the ADC, when the DC-bus voltage drops below this value, and the application switches to the Fault state. Also, an indication LED is enabled on the FreeMaster page.

7.4.5 DC_OVERVOLTAGE

The DC over-voltage level of application.

A **DC_Overvoltage** fault is triggered by the ADC, when the DC-bus voltage exceeds this value, and the application switches to the Fault state. Also, an indication LED is enabled on the FreeMaster page.

7.4.6 DC_OVERCURRENT

The DC over-current level of the application.

A **DC_Overcurrent** fault is triggered from the ADC, when the DC-bus current exceeds this value, and the application switches to the Fault state. Also, an indication LED is enabled on the FreeMaster page.

7.5 Motor Related Parameters

The parameters in Group 3 can be divided into sub group.

- Alignment parameters
- Start parameters
- Run parameters
- General motor and zero crossing parameters
- Controller parameters
  - Speed PI controller
  - Alignment current PI controller
  - Current limitation PI controller

As described above, parameters for desired motor can be loaded by calling one of the following functions:

- **MotorA()**, IB23811 Motor
- **MotorB()**, N2311 Motor

Unused functions will be dead-stripped by the compiler, so they will not occupy program Flash.

To select the desired motor, use **#define** directives in the **bldczcconfig.h** header file.

7.5.1 Alignment Parameters

7.5.1.1 Parameter Summary

Parameters for alignment are listed and explained in Table 7-3.

Table 7-3. Alignment Parameter Summary

Name	Description	Unit	Range	Default	FreeMaster Location
PER_ALIGNMENT_S	Duration of the alignment commutation state.	s (seconds)	0.1–12.0	0.5	Control > Alignment
CUR_ALIGN_DESIRED_A	Alignment current.	A	0.1–3.0	1.5	Control > Alignment
ALIGNMENT_STEP_CMP_ABC	PWM pattern applied to the motor during alignment, when the desired speed is positive.	N/A	0 [A+ B-]–5 [A+ C-]	5 [A+ C-]	Control > Alignment
ALIGNMENT_STEP_CMP_ACB	PWM pattern applied to the motor during alignment, when the desired speed is negative.	N/A	0 [A+ B-]–5 [A+ C-]	4 [B+ C-]	Control > Alignment

7.5.1.2 PER_ALIGNMENT_S

The duration of the alignment state of commutation, in seconds.

The duration of alignment must be long enough to stabilize the motor, before it starts.

For tuning, it is recommended to keep the duration of the alignment state sufficiently long to ensure that alignment is achieved. When it has been seen that the motor is aligned properly, this duration can be shortened, as long as it enables proper alignment.

7.5.1.3 CUR_ALIGN_DESIRED_A

Value of the alignment current.

The alignment current PI controller tries to keep the DC-bus current at this value, during the alignment commutation state.

The alignment current can be as high as the nominal current of the motor.

7.5.1.4 ALIGNMENT_STEP_CMP_ABC

The PWM pattern applied to the motor during the alignment state, when the desired speed is positive.

This parameter can be a value from 0 to 5, where every value corresponds to a PWM pattern, as follows.

0: A top, B bottom
1: B bottom, C top
2: C top, A bottom
3: A bottom, B top
4: B top, C bottom
5: C bottom, A top

These patterns are defined by the **BldcZC_Cmt_StepTable** array in **bldcdrv.c**.

7.5.1.5 ALIGNMENT_STEP_CMP_ACB

The PWM pattern applied to the motor in the alignment state, when the desired speed is negative.

This parameter can be a value from 0 to 5, where every value corresponds to a PWM pattern, as follows.

0: A top, B bottom
1: B bottom, C top
2: C top, A bottom
3: A bottom, B top
4: B top, C bottom
5: C bottom, A top

These patterns are defined by the **BldcZC_Cmt_StepTable** array in **bldcdrv.c**.

7.5.2 Start Parameters

7.5.2.1 Parameter Summary

Parameters for start are listed and explained in Table 7-4.

Table 7-4. Start Parameter Summary

Name	Description	Units	Range	Default	FreeMaster Location
PER_START_PROCCMT_US	Minimum value for period Toff.	µs	50.0–30000.0	170.0	Control > Start
PER_CMTSTART_US	Commutation period used to compute the first (start) commutation period	µs	50.0–30000.0	3600.0	Control > Start
PER_TOFFSTART_US	First (start) interval Toff after commutation (where back-EMF zero crossing is not detected).	µs	50.0–30000.0	7200.0	Control > Start
COEF_START_CMT_PRECOMP_FRAC	Coefficient for calculating preset commutation period	Fraction	FRAC16(0.1) – FRaC16(0.9)	FRAC16(0.5)	Control > Start
COEF_START_CMT_PRECOMP_LSHFT	Coefficient for calculating preset commutation period	N/A	1–4	2	Control > Start
COEF_START_HLFCMT	Coefficient for calculating commutation period (where back-EMF zero crossing is not detected).	Fraction	FRAC16(0.1) – FRAC16(0.9)	FRAC16(0.125)	Control > Start
COEF_START_TOFF	Coefficient for calculating period Toff (where back-EMF zero crossing is not detected during start).	Fraction	FRAC16(0.1) – FRAC16(0.5)	FRAC16(0.250)	Control > Start
MIN_ZCROSOK_START	Minimum number of zero crossing OK commutation to finish the BLDC starting phase.	N/A	2–20	2	Control > Start

Changing these parameters in the FreeMaster environment will not affect the application until the motor is stopped and restarted.

7.5.2.2 PER_START_PROCCMT_US

The minimum value for the period Toff.

This value is used to limit the period Toff to a minimum. Therefore, the period Toff applied after every commutation can have a minimum value of **PER_START_PROCCMT_US**.

7.5.2.3 PER_CMTSTART_US

The commutation period used to compute the first (start) commutation period.

During the start commutation state, no zero crossing has been detected before. Therefore, the commutation algorithm must predict a value for the commutation period and start the process. This value is used as the first predicted commutation period.

To select this value, see Table 7-5.

7.5.2.4 PER_TOFFSTART_US

The first (start) interval Toff after commutation (where back-EMF zero crossing is not detected).

This value is the first predicted period duration Toff for the start commutation state.

Setting these constants is an experimental process. It is difficult to obtain a precise formula because many factors involved are difficult to quantify for real drives (motor and load mechanical inertia, motor electromechanical constants, and sometimes also the motor load, for example). Therefore, constants must be set for a specific motor. Table 7-5 aids in the setting of constants.

Table 7-5. Recommended Start Periods

Motor size	x_PER_CMTSTART_US [µs]	x_PER_TOFFSTART_US [µs]	First commutation period [ms]
Slow motor/ high load motor mechanical inertia	>5000	>10000	>10
Fast motor/ high load motor mechanical inertia	<5000	<10000	<10

Slowing down the speed regulator may help if startup problems occur when using the settings in Table 7-5.

7.5.2.5 COEF_START_CMT_PRECOMP_FRAC

Coefficient for calculating the commutation preset period.

This coefficient is used with **COEF_START_CMT_PRECOMP_LSHFT** to build **COEF_START_CMT_PRECOMP** as follows

$$\text{CoefStartCmtPrecomp} = \text{CoefStartCmtPrecpFrac} \ll \text{CoefStartCmtPreCompLshft}$$

With the help of this shift operation, it is possible to make the commutation period value greater than FRAC16(1.0).

This constant has the same function as **COEF_RUN_CMT_PRECOMP_FRAC,** where one is for the run commutation stage and one is for the start commutation state.

For a detailed explanation, see 7.5.3.3 COEF_RUN_CMT_PRECOMP_FRAC.

7.5.2.6 COEF_START_CMT_PRECOMP_LSHFT

Coefficient for calculating the commutation preset period.

This coefficient is used with **COEF_START_CMT_PRECOMP_FRAC** to build **COEF_START_CMT_PRECOMP**. See 7.5.2.5 COEF_START_CMT_PRECOMP_FRAC

This constant has the same function as **COEF_RUN_CMT_PRECOMP_LSHFT**, where one is for the run commutation stage and one is for the start commutation state.

For a detailed explanation, see 7.5.3.3 COEF_RUN_CMT_PRECOMP_FRAC.

7.5.2.7 COEF_START_HLFCMT

Coefficient for calculating the commutation period, after zero crossing is detected.

This constant has the same function as **COEF_RUN_HLFCMT**, where one is for the run commutation state and one is for the start commutation state.

For a detailed explanation, see 7.5.3.5 COEF_RUN_HLFCMT.

7.5.2.8 COEF_START_TOFF

Coefficient for calculating the period Toff (where back-EMF zero crossing is not detected during start).

This constant has the same function as **COEF_RUN_TOFF**, where one is for the run commutation state and one is for the start commutation state.

For a detailed explanation, see 7.5.3.6 COEF_RUN_TOFF.

7.5.2.9 MIN_ZCROSOK_START

Minimum number of zero crossing OK commutation to finish the BLDC starting phase.

After this number of zero crossings is detected, the start commutation state ends and the run commutation state begins.

7.5.3 Run Parameters

7.5.3.1 Parameter Summary

Parameters for run are listed and explained in Table 7-6.

Table 7-6. Run Parameters Summary

Name	Description	Units	Range	Default	FreeMaster Location
PER_RUN_PROCCMT_US	Minimum value for period Toff.	µs	50.0–30000.0	170.0	Control > Run
COEF_RUN_CMT_PRECOMP_FRAC	Coefficient for calculating preset commutation period	Fraction	FRAC16(0.1) – FRAC16(0.9)	FRAC16(0.5)	Control > Run
COEF_RUN_CMT_PRECOMP_LSHFT	Coefficient for calculating preset commutation period	N/A	1–4	2	Control > Run
COEF_RUN_HLFCMT	Coefficient for calculating commutation period, after zero crossing is detected	Fraction	FRAC16(0.1) – FRAC16(0.9)	FRAC16(0.5)	Control > Run
COEF_RUN_TOFF	Coefficient for calculating period Toff	Fraction	FRAC16(0.1) – FRAC16(0.5)	FRAC16(0.250)	Control > Run
MAX_ZCROSERR	Maximum commutation period counts with no zero crossing detection, to stop the commutation	N/A	2–30	4	Control > Run

Changing these parameters in the FreeMaster environment does not affect the application until the motor is stopped and restarted.

7.5.3.2 PER_RUN_PROCCMT_US

Minimum value for the period Toff.

This value is used for limit the period Toff to a minimum. Therefore, the period Toff applied after every commutation can be a minimum of **PER_RUN_PROCCMT_US.**

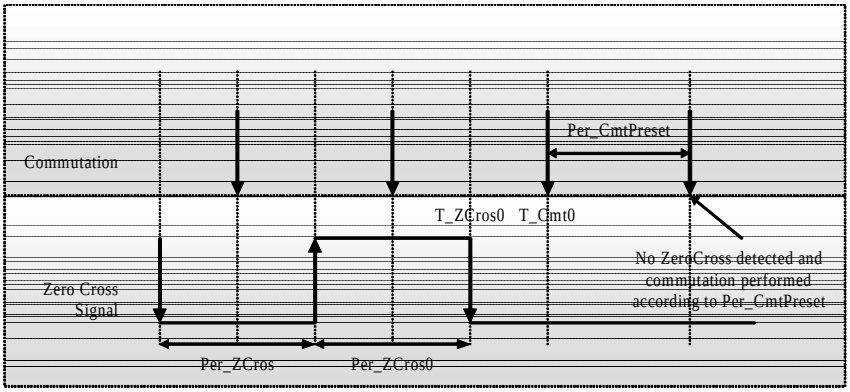
7.5.3.3 COEF_RUN_CMT_PRECOMP_FRAC

Coefficient for calculating the commutation preset period.

This coefficient is used with **COEF_RUN_CMT_PRECOMP_LSHFT** to build **COEF_RUN_CMT_PRECOMP** as follows.

$$\text{CoefRunCmtPrecomp} = \text{CoefRunCmtPrecpFrac} \ll \text{CoefRunCmtPreCompLshft}$$

The preset period is calculated after every commutation, as a default commutation period in case no zero crossing is detected in the next commutation period. If no zero crossing is detected before the preset commutation period expires, commutation is performed. If zero crossing is detected before the preset commutation time expires, the next commutation time is updated and the preset commutation time is not used.



Inside *SetCalculation*

Per_ZCros0

Per_ZCrosFlt = (Per_ZCros + PerZCros0) / 2

Inside *PresetCalculation*

if (Per_ZCrosFlt * **COEF_RUN_CMT_PRECOMP_FRAC** * 2 ^ **COEF_RUN_CMT_PRECOMP_LSHFT**) < Max_PerCmt

Per_CmtPreset = Per_ZCrosFlt * **COEF_RUN_CMT_PRECOMP_FRAC** * 2 ^ **COEF_RUN_CMT_PRECOMP_LSHFT**

else

Per_CmtPreset = Max_PerCmt

In the next incoming commutation (*CmtTimeOut*)

Next cmt time = T_Cmt0 + Per_CmtPreset

Figure 7-5. Using COEF_RUN_CMT_PRECOMP

With the help of this shift operation, it is possible to make the commutation period value greater than FRAC16(1.0).

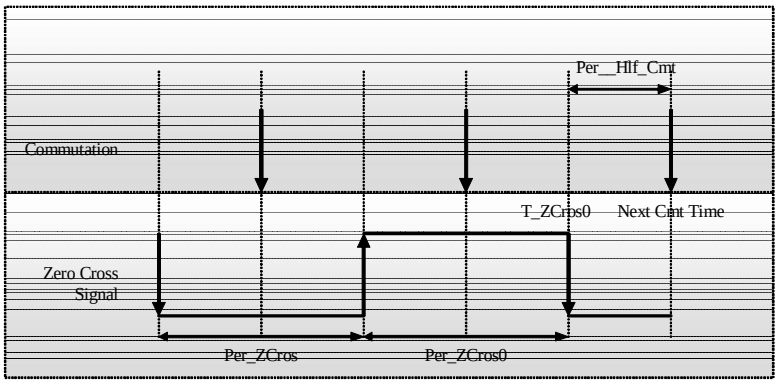
7.5.3.4 COEF_RUN_CMT_PRECOMP_LSHFT

Coefficient for calculating the commutation preset period.

This coefficient is used with **COEF_RUN_CMT_PRECOMP_FRAC** to build **COEF_RUN_CMT_PRECOMP**. See 7.5.3.3 COEF_RUN_CMT_PRECOMP_FRAC.

7.5.3.5 COEF_RUN_HLFCMT

Coefficient for calculating the commutation period after zero crossing is detected, as described in Figure 7-6.



Inside *SetCalculation*

Per_ZCrosFlt = (Per_ZCros + PerZCros0) / 2
Per_HlfCmt = Per_ZCrosFlt * **COEF_RUN_HLFCMT**

After determining ZeroCross time (*bldczcZCrosEdgeIntAlg* or *ZCToffTimeout*)

Next Cmt Time = T_ZCros + Per_Hlf_Cmt

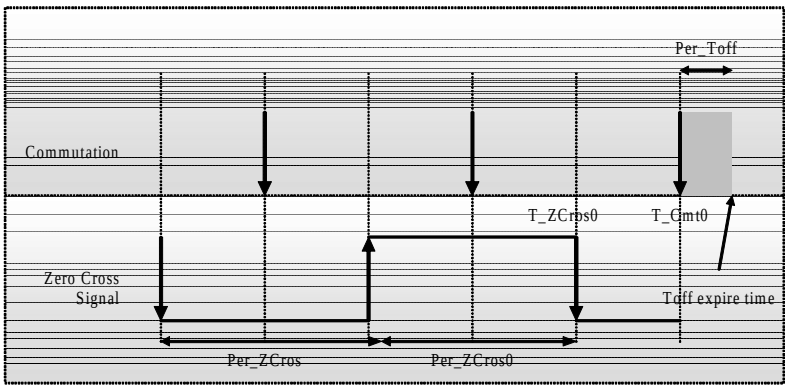
Figure 7-6. COEF_RUN_HLFCMT Usage

7.5.3.6 COEF_RUN_TOFF

Coefficient for calculating the period Toff.

After every commutation, zero crossing detection is disabled for a period Toff. This period is calculated using **COEF_RUN_TOFF**, as described in Figure 7-7.

The period Toff applied after commutation has a minimum limit.



Inside *SetCalculation*

Per_ZCrosFlt = (Per_ZCros + PerZCros0) / 2

if (Per_ZCrosFlt * **COEF_RUN_TOFF**) > Const_PerProcCmt

Per_Toff = Per_ZCrosFlt * **COEF_RUN_TOFF**

else

Per_Toff = Const_PerProcCmt

In the next incoming commutation (*CmtTimeOut*)

Toff expire time = T_Cmt0 + Per_Toff

Figure 7-7. COEF_RUN_TOFF Usage

```
Where:

T_Cmt[n]          : Time for commutation for step n
T_Next[n] (cmt)   : Time of the next commutation for step n
T_ZCtos[n]        : Time of the zero crossing for step n
T_zCtos0[n]       : Time of the previous zero crossing for step n
Per_Toff[n+1]     : Period of the Toff duration to be applied at step n+1
Per_CmtPreset     : Preset Commutation Period from commutation to next commutation
                    if no Zero Crossing was captured
Per_ZCtos[n]      : Period between last 2 Zero Crossings for step n
Per_ZCtos0[n]     : Previous period between Zero Crossings for step n
Per_ZCtosFlt      : Estimated period of commutation filtered for step n
Per_HlfCmt        : Period from Zero Crossing to commutation (half commutation)
```

7.5.3.7 MAX_ZCROSERR

Maximum number of commutation period counts with no zero crossing detection, to stop the commutation.

This constant is used to control commutation problems. If no zero crossing is detected for **MAX_ZCROSERR** commutation cycles, the application switches to the Stop commutation state and tries for a new alignment, if allowed by the run/stop status.

When tuning the software for other motors, this constant can be increased temporarily.

7.5.4 General Parameters

7.5.4.1 Parameter Summary

These parameter are generally related witch the motor. They are listed and explained in Table 7-7.

Table 7-7. General Parameters Summary

Name	Description	Units	Range	Default	FreeMaster Location
SPEED_ROTOR_MAX_RPM	Maximum mechanical rotor speed.	RPM	Defined by motor design, no range	2000	N/A
MOTOR_COMMUTATION_PREV	Number of commutations per revolution.	N/A	Defined by motor design, no range	12	N/A
OMEGA_MIN_SYSU	Minimum speed.	Fraction	0–6553	3276	N/A
CURR_LIMIT_DESIRED_A	Desired current limitation value for DC-bus.	A	0.1–4.0	4.0	Control > Current Limiting

7.5.4.2 SPEED_ROTOR_MAX_RPM

Maximum mechanical rotor speed.

7.5.4.3 MOTOR_COMMUTATION_PREV

Number of commutations per revolution.

This value must be set according to the number of motor poles.

7.5.4.4 OMEGA_MIN_SYSU

If the adjusted speed or actual speed of motor is below this value, the motor is stopped.

7.5.4.5 CURR_LIMIT_DESIRED_A

The DC-bus current will be limited above this value. If the DC-bus current is greater than this value, the PWM duty cycle will be limited until the DC-bus current drops again below this value.

The current limitation is active the both alignment and running commutation states. An alarm signal is also generated, if current limiting is acting on the output.

This parameter can also be set from the DC-bus current gauge on the FreeMaster page.

7.5.5 Controller Parameters

the BLDC sensorless application contains three PI controllers.

- Speed controller: to keep the motor speed at the desired value; active only during run mode.
- Alignment current controller: to keep the alignment current at the desired value; active only during alignment.
- Current limitation controller: to keep the DC-bus current below a desired value; active during alignment and running.

All three PI controllers have same structure. As with all PI controllers, their proportional gain and integral gain can be adjusted.

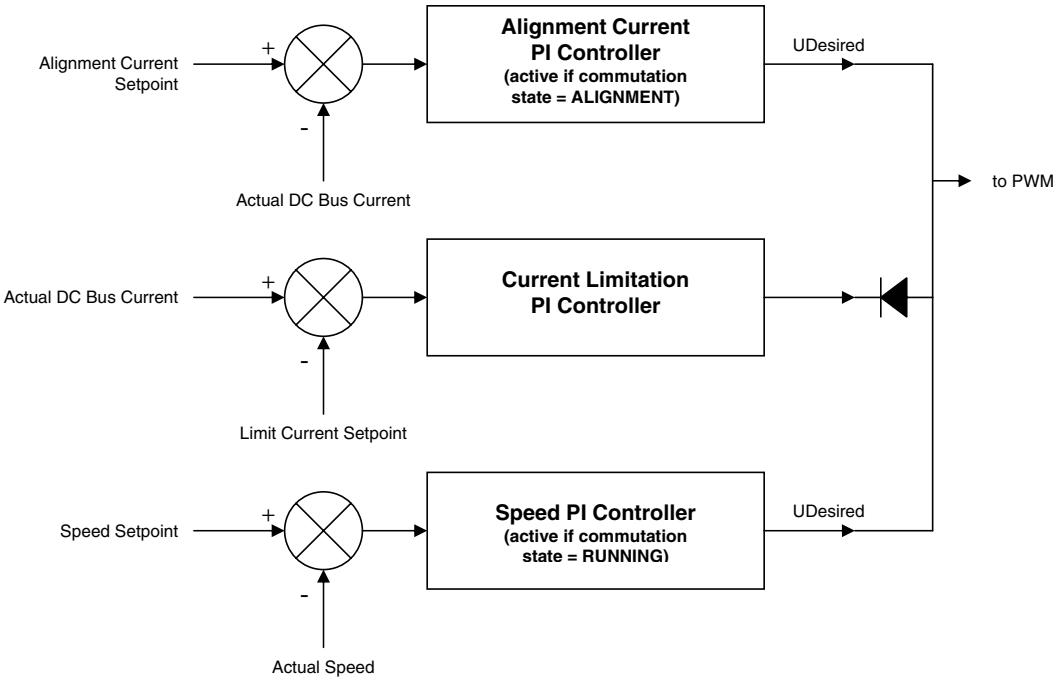


Figure 7-8. PI Controller Structure

In this application, all PI controllers proportional and integral constants are set experimentally.

7.5.5.1 Parameter Summary

Parameters for the PI controllers are listed and explained in Table 7-8.

Table 7-8. Controller Parameters Summary

Name	Description	Units	Range	FreeMaster Location
PI_PROPORTIONAL_GAIN	Portion value of proportional gain.	Fraction	FRAC16(0.5) – FRAC16(1.0)	Control > PI Controllers
PI_PROPORTIONAL_GAIN_SCALE	Scale value of proportional gain.	N/A	(-13)–(+13)	Control > PI Controllers
PI_INTEGRAL_GAIN	Portion value of integral gain.	Fraction	FRAC16(0.5) – FRAC16(1.0)	Control > PI Controllers
PI_INTEGRAL_GAIN_SCALE	Scale value of integral gain.	N/A	(-13)–(+13)	Control > PI Controllers

7.5.5.2 PI_PROPORTIONAL_GAIN

Portion value of proportional gain.

This value is used with **PROPORTIONAL_GAIN_SCALE** to build the proportional gain of the application. See 7.5.5.6 PI Algorithm for detailed explanations and parameter ranges.

7.5.5.3 PI_PROPORTIONAL_GAIN_SCALE

Scale value of proportional gain.

This value is used with **PROPORTIONAL_GAIN** to build the proportional gain of the application. See 7.5.5.6 PI Algorithm for detailed explanations and parameter ranges.

7.5.5.4 PI_INTEGRAL_GAIN

Portion value of integral gain.

This value is used with **INTEGRAL_GAIN_SCALE** to build the integral gain of the application. See 7.5.5.6 PI Algorithm for detailed explanations and parameter ranges.

7.5.5.5 PI_INTEGRAL_GAIN_SCALE

Scale value of proportional gain.

This value is used with **INTEGRAL_GAIN** to build the integral gain of the application. See 7.5.5.6 PI Algorithm for detailed explanations and parameter ranges.

7.5.5.6 PI Algorithm

The PI algorithm in continuous time domain:

$$u(t)= K\left[e(t) + \frac{1}{T_I}\int_0^{\tau} e(\tau)d\tau\right] \quad (7\text{-}1)$$

$$e(t) = w(t) - y(t) \quad (7\text{-}2)$$

The transfer function is shown below:

$$F(p)= K\left[1 + \frac{1}{T_I}\cdot\frac{1}{p}\right] = \frac{u(p)}{e(p)} \quad (7\text{-}3)$$

The PI algorithm in discrete time domain in fractional arithmetic:

$$u_f(k)= K_{sc}\cdot e_f(k) + u_{If}(k-1) + K_{Isc}\cdot\frac{T}{T_I}\cdot e_f(k) \quad (7\text{-}4)$$

$$e(k) = w(k) - y(k) \quad (7\text{-}5)$$

where

$$u_f(k) = u(k)/u_{max} \quad (7\text{-}6)$$

$$w_f(k) = w(k)/w_{max} \quad (7\text{-}7)$$

$$y_f(k) = y(k)/y_{max} \quad (7\text{-}8)$$

$$e_f(k) = e(k)/e_{max} \quad (7\text{-}9)$$

$$K_{sc} = K\cdot\frac{e_{max}}{u_{max}} \quad (7\text{-}10)$$

$$K_{Isc} = K\cdot\frac{T}{T_I}\cdot\frac{e_{max}}{u_{max}} \quad (7\text{-}11)$$

The integral is approximated by the Backward Euler method, also known as Backward Rectangular or right-hand approximation. For this method, $1/p$ is approximated by $u_I(k) = u_I(k-1) + T \cdot e(k)$.

$$K_{sc} = ProportionalGain \cdot 2^{-ProportionalGainScale} \quad (7\text{-}12)$$

$$K_{Isc} = IntegralGain \cdot 2^{-IntegralGainScale} \quad (7\text{-}13)$$

$$0{,}5 < ProportionalGain < 1 \quad (7\text{-}14)$$

$$0{,}5 < IntegralGain < 1 \quad (7\text{-}15)$$

$$-14 < ProportionalGainScale < 14 \quad (7\text{-}16)$$

$$-14 < IntegralGainScale < 14 \quad (7\text{-}17)$$

Calculation of the scaling coefficients **ProportionalGainScale** and **IntegralGainScale** can be done using the following equations.

$$\frac{\log(0.5) - \log(ProportionalGain)}{\log 2} < ProportionalGainScale \quad (7\text{-}18)$$

$$\frac{\log 1 - \log(ProportionalGain)}{\log 2} > ProportionalGainScale \quad (7\text{-}19)$$

$$\frac{\log(0.5) - \log(IntegralGain)}{\log 2} < IntegralGainScale \quad (7\text{-}20)$$

$$\frac{\log 1 - \log(IntegralGain)}{\log 2} > IntegralGainScale \quad (7\text{-}21)$$

For example, if $K = 0.05$ then the scaling factor and scaling gain are given by the following equations.

$$\frac{\log(0.5) - \log(0.05)}{\log 2} < Scale < \frac{\log 1 - \log(0.05)}{\log 2}$$

$$3.3219 < Scale < 4.3219$$

From these equations, $Scale = 4$, and $Gain = K \cdot 2^{Scale} = 0.05 \cdot 2^4 = 0.8$

The controller implements the following limitations.

- The integral portion limitation (anti wind-up effect):

$$NegativePILimit \le u_{If}(k) \le PositivePILimit \quad (7\text{-}22)$$

- The controller output limitation:

$$NegativePILimit \le u_f(k) \le PositivePILimit \quad (7\text{-}23)$$

where:

$e_f(k)$	=	Input error in step k - discrete time domain fractional
$w_f(k)$	=	Desired value in step k - discrete time domain fractional
$y_f(k)$	=	Measured value in step k - discrete time domain fractional
$u_f(k)$	=	Controller output in step k - discrete time fractional
$u_{If}(k-1)$	=	Integral output portion in step k-1 - discrete time fractional

T_I = Integral time constant

T = Sampling time

t = Time

K = Controller gain

p = Laplace variable

Ranges of the PI parameters are as follows

$$0.5 < \mathrm{ProportionalGain} < 1$$

$$0.5 < \mathrm{IntegralGain} < 1$$

$$-14 < \mathrm{IntegralProportionalGainScale} < 14$$

$$0 - 14 < \mathrm{ProportionalGainScale} < 14$$

$$2^{-15} < K < 2^{15}$$

$$2^{-15} < K \cdot \frac{T}{T_I} < 2^{15}$$

Appendix A.
References

1. *Design of Brushless Permanent-magnet Motors*, J.R. Hendershot JR and T.J.E. Miller, Magna Physics Publishing and Clarendon Press, 1994
2. *Brushless DC Motor Control using the MC68HC708MC4*, AN1702/D, John Deatherage and Jeff Hunsinger, Freescale Semiconductor, Inc.
3. *DSP56F800 Family Manual*, DSP56F800FM/D, Freescale Semiconductor, Inc.
4. *Elektricke pohony*, Caha, Z.; Cerny, M. (1990), SNTL, ISBN 80-03-00417-7, Praha.
5. Freescale Semiconductor, Inc. web page: http://www.freescale.com
6. Pittman web page: http://www.pennmotion.com
7. MCG web page: http://mcg-net.com
8. *CodeWarrior for Motorola DSP56800 Embedded Systems*, CWDSP56800, Metrowerks 2001
9. *DSP56F8013 Evaluation Module Hardware Schematics*, Freescale Semiconductor, Inc., 2001
10. *Freescale 33395 Evaluation Motor Board Users Manual, DRM33395*, Freescale Semiconductor, Inc., 2004
11. *Freescale Micro Power Stage Board Users Manual*, Freescale Semiconductor, Inc.
12. *FreeMaster Software Users Manual*, Freescale Semiconductor, Inc., 2004
13. *Processor Experts Users Manual*
14. *Motor Control Algorithms Description*, MCSL 2002

Appendix B.
Glossary

AC — alternating current

ADC — analogue-to-digital converter

back-EMF — back electromotive force (in this document, it means the voltage induced in motor winding due to the movement of the rotor)

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

BLDC motor — brushless direct current motor

commutation — a process providing the creation of a rotation field by switching of power transistor (electronic replacement of brush and commutator)

commutator — a mechanical device that alternates the DC current in a DC commutator motor, causing rotation of the DC commutator motor

COP — Computer operating properly watchdog timer

DC — direct current

DC-bus — power supply bus carrying direct current

DC motor — direct current motor; unless stated otherwise, it means a motor with brushes

DSP — digital signal processor

dead-time (DT) — short time that must be inserted between turning off one transistor in an inverter half bridge and turning on the complementary transistor, because of the limited switching speed of the transistors

duty cycle — the ratio of the time a signal is on to the time it is off; duty cycle is usually stated as a percentage.

GPIO — general purpose input/output

Hall sensor — a position sensor. In this application, sensors are arranged to give six defined events (each 60 electrical degrees) per electrical revolution (for a 3-phase motor)

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

ISR — interrupt service routine

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 on an external signal

JTAG — a Joint Test Action Group interface allowing on-chip emulation and programming

LIN — local interconnect network

logic 1 — a voltage level approximately equal to the input power voltage (V_{DD})

logic 0 — a voltage level approximately equal to the ground voltage (V_{SS})

MCU — microcontroller unit

Processor Expert — integrated development environment for Freescale MC56F8xxx series hybrid controllers

PI controller — proportional-integral controller

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

PWM — pulse width modulation

quadrature decoder — a module providing decoding of position from a quadrature encoder mounted on a motor shaft

quad timer — a module with four 16-bit timers

reset — to force a device to a known condition

RMS — root mean square

RPM — revolutions per minute

serial communications interface module (SCI) — a module that supports asynchronous communication

serial peripheral interface module (SPI) — a module that supports synchronous communication

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

software interrupt (SWI) — an instruction that causes an interrupt and its associated vector fetch

timer — a module used to relate events in a system to a point in time

zero crossing point — the instant when the back-EMF voltage of a phase crosses half of the DC-bus voltage value

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

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

DRM070
Rev. 0
5/2005