Order by AN1915/D
(Motorola Order Number)
Rev. 0, 9/02

3-Phase BLDC Motor Control with Quadrature Encoder using DSP56F80x

Design of Motor Control Application Based on Motorola Software Development Kit

Pavel Grasblum

Contents

1. Introduction of Application Benefit

This Application Note describes the design of a 3-phase BLDC (Brushless DC) motor drive based on Motorola's DSP56F80x dedicated motor control device. The software design takes advantage of the SDK (Software Development Kit) developed by Motorola.

BLDC motors are very popular in a wide application area. Compared with a DC motor, the BLDC motor loads a commutator and so it is more reliable than the DC motor. Also in comparison to an AC induction motor, the BLDC motor has advantages. BLDC motors generate the rotor achieve magnetic flux with rotor magnets so that BLDC motors higher efficiency. Therefore BLDC motors are used in high-end white goods (refrigerators, washing machines, dishwashers, etc.), high-end pumps, fans and in other appliances, which require high reliability and efficiency.

The concept of the application is a speed-closed loop BLDC drive using a Quadrature Encoder. It serves as an example of a BLDC motor control system design using a Motorola DSP with SDK support. It also illustrates the usage of dedicated motor control libraries that are included in the SDK.

This Application Note includes the basic motor theory, system design concept, hardware implementation and software design, including the PC Master visualization tool.

2. Motorola DSP Advantages and Features

The Motorola DSP56F80x family is well suited for digital motor control, combining the DSP's calculation capability with the MCU's controller features on a single chip. These DSPs offer many dedicated peripherals, such as Pulse Width Modulation (PWM) module, Analog-to-Digital Converter (ADC), Timers, communication peripherals (SCI, SPI, CAN), on-board Flash and RAM. Generally, all family members are well-suited for various motor controls.

A typical member of the family, the DSP56F805, provides the following peripheral blocks:

- Two Pulse Width Modulator modules (PWMA & PWMB), each with six PWM outputs, three Current Sense inputs, and four Fault inputs, fault tolerant design with deadtime insertion, supporting both center- and edge- aligned modes
- Twelve-bit Analog-to-Digital Converters (ADCs), supporting two simultaneous conversions with dual 4-pin multiplexed inputs; the ADC can be synchronized by PWM modules
- Two Quadrature Decoders (Quad Dec0 & Quad Dec1), each with four inputs, or two additional Quad Timers A & B
- Two dedicated General Purpose Quad Timers totaling 6 pins: Timer C with 2 pins and Timer D with 4 pins
- CAN 2.0 A/B Module with 2-pin ports used to transmit and receive
- Two Serial Communication Interfaces (SCI0 & SCI1), each with two pins, or four additional GPIO lines
- Serial Peripheral Interface (SPI), with a configurable 4-pin port, or four additional GPIO lines
- Computer Operating Properly (COP) timer
- Two dedicated external interrupt pins
- Fourteen dedicated General Purpose I/O (GPIO) pins, 18 multiplexed GPIO pins
- External reset pin for hardware reset
- JTAG/On-Chip Emulation (OnCE)
- Software-programmable, Phase Lock Loop-based frequency synthesizer for the DSP core clock

Table 2-1. Memory Configuration

	DSP56F801	DSP56F803	DSP56F805	DSP56F807
Program Flash	8188 x 16-bit	32252 x 16-bit	32252 x 16-bit	61436 x 16-bit
Data Flash	2K x 16-bit	4K x 16-bit	4K x 16-bit	8K x 16-bit
Program RAM	1K x 16-bit	512 x 16-bit	512 x 16-bit	2K x 16-bit
Data RAM	1K x 16-bit	2K x 16-bit	2K x 16-bit	4K x 16-bit
Boot Flash	2K x 16-bit	2K x 16-bit	2K x16-bit	2K x 16-bit

In addition to the fast Analog-to-Digital converter and the 16-bit Quad Timers, the most interesting peripheral from the BLDC motor control point of view is the Pulse Width Modulation (PWM) module. The PWM module offers a high degree of freedom in its configuration, permitting efficient control of the BLDC motor.

The PWM has the following features:

- Three complementary PWM signal pairs, or six independent PWM signals

- Complementary channel operation
- Deadtime insertion
- Separate top and bottom pulse width correction via current status inputs or software
- Separate top and bottom polarity control
- Edge-aligned or center-aligned PWM signals
- 15 bits of resolution
- Half-cycle reload capability
- Integral reload rates from 1 to 16
- Individual software-controlled PWM outputs
- Mask and Swap of PWM outputs
- Programmable fault protection
- Polarity control
- 20-mA current sink capability on PWM pins
- Write-protectable registers

The BLDC motor control utilizes the PWM block set in the complementary PWM mode (for control in complementary mode - see chapter 3.1.2 Complementary Switching of Power Transistors), permitting the generation of control signals for all switches of the power stage, with inserted deadtime. The PWM outputs can be controlled separately by software, where setting the control signal to logical 0 or 1 enables/disables control signal. The next key feature, which is very useful in BLDC motor control, is the channel swap function. The swap function allows the immediate change of top and bottom transistors in the phase. These functions allow the rotor commutation and speed control to be split into two independent program parts. The state of the control signals can be changed immediately when required by the motor position (phase commutation) without changing the content of the PWM value registers. These changes can be accomplished asynchronously to the PWM duty cycle update.

The Quad Timer is an extremely flexible module, providing all of the required services related to time events. It has the following features:

- Each timer module consists of four 16-bit counters/timers
- Count up/down
- Counters are cascadable
- Programmable count modulo
- Max count rate equals peripheral clock/2 when counting external events
- Max count rate equals peripheral clock 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 BLDC motor application utilizes one channel of the Quad Timer module counting in quadrature mode. It enables rotor position sensing using the Quadrature Encoder. The second channel of the Quad Timer module is set to generate a time base for the speed controller.

The Quadrature Decoder is a module that provides decoding of position signals from a Quadrature Encoder mounted on a motor shaft. It has the following features:

- Includes logic to decode quadrature signals
- Configurable digital filter for inputs
- 32-bit position counter
- 16-bit position difference counter
- Maximum count frequency equals the peripheral clock rate
- Position counter can be initialized by SW or external events
- Preloadable 16-bit revolution counter
- Inputs can be connected to a general purpose timer to aid low speed velocity.

The BLDC motor application utilizes the Quadrature Decoder connected to Quad Timer module A. It uses the Decoder's digital input filter, to filter the Encoder's signals, but does not make use of its decoding functions. So the decoder's digital processing capabilities are free to be used by another application.

3. Target Motor Theory

A brushless DC (BLDC) motor is a rotating electric machine, where the stator is a classic three-phase stator like that of an induction motor, and the rotor has surface-mounted permanent magnets (see **Figure 3-1**).

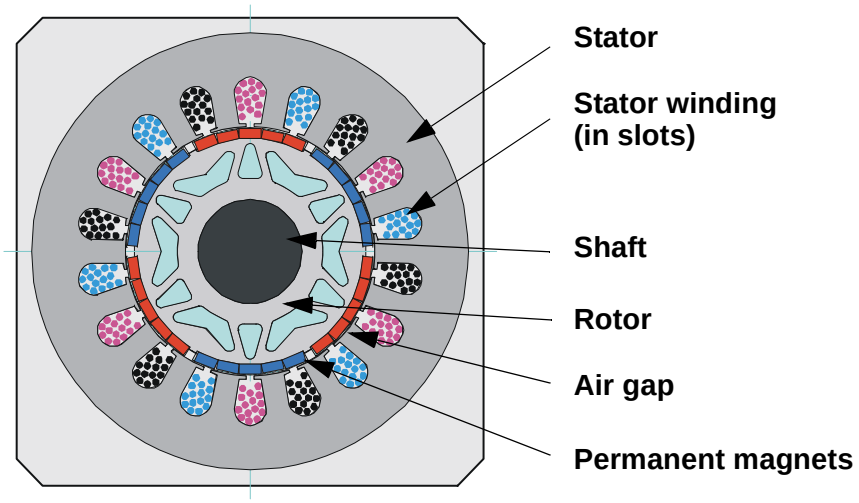

Figure 3-1. BLDC Motor - Cross Section

In this respect, the BLDC motor is equivalent to an inverted DC commutator motor, in which the magnet rotates while the conductors remain stationary. In the DC commutator motor, the current polarity is reversed by the commutator and brushes. However, in the brushless DC motor, the polarity reversal is performed by power transistors switched in synchronization with the rotor position. Therefore, BLDC motors often incorporate either internal or external position sensors to sense the actual rotor position, or the position can be detected without sensors.

3.1 Digital Control of a BLDC Motor

The BLDC motor is driven by rectangular voltage strokes coupled with the given rotor position (see **Figure 3-2**). The generated stator flux, together with the rotor flux, which is generated by a rotor magnet, defines the torque and thus the speed of the motor. To get the maximum generated torque, the voltage strokes have to be applied to the three-phase winding system,

so that the angle between the stator flux and the rotor flux is kept close to 90°. To meet this criteria, the motor requires electronic control for proper operation.

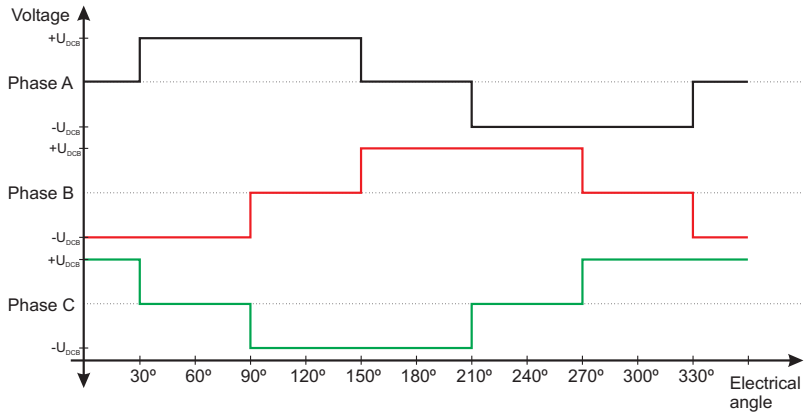

Figure 3-2. Voltage Strokes Applied onto the 3-ph BLDC Motor

For the common 3-phase BLDC motor a standard 3-phase power stage is used. Such a power stage for 3-phase BLDC motors is illustrated in **Figure 3-3**. The power stage utilizes six power transistors with independent switching. The power transistors may be switched to independent or complementary mode.

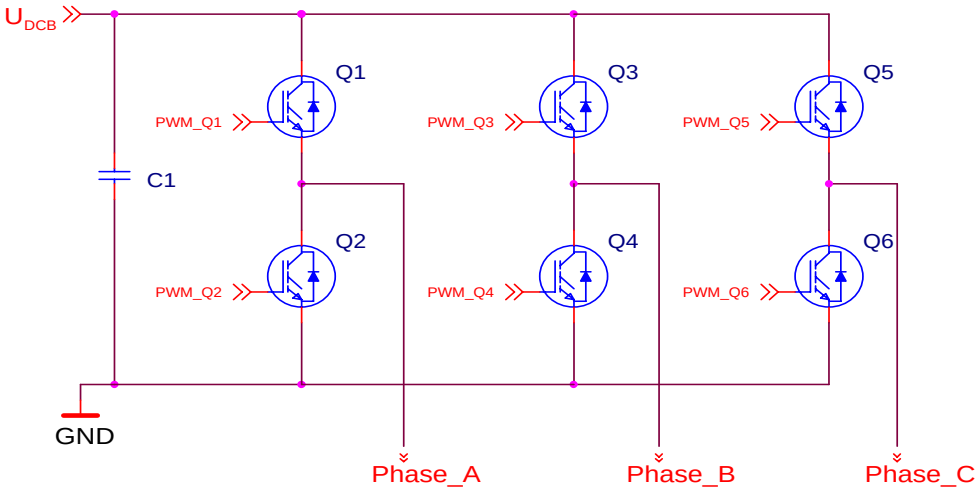

Figure 3-3. 3-phase BLDC Power Stage

In both modes the 3-phase power stage supplies two motor phases concurrently. The third phase is not powered (see **Figure 3-2**). Thus, we get six possible voltage vectors that are applied to the BLDC motor. **Figure 3-2** shows the maximum voltage amplitude applied to the BLDC motor, which is equal to the DC Bus Voltage. The lower voltage is generated using a PWM technique (see **Figure 3-4** and **Figure 3-5**). There are two basic types of power transistor switching: independent switching and complementary switching.

3.1.1 Independent Switching of Power Transistors

With independent switching, only two transistors are switched on when the current is conducted from the power supply to the phase of the BLDC motor. In one phase, the top transistor is switched on, in the second phase the bottom transistor is switched on and the third phase is not powered. During freewheeling, all transistors are switched off (see **Figure 3-4**).

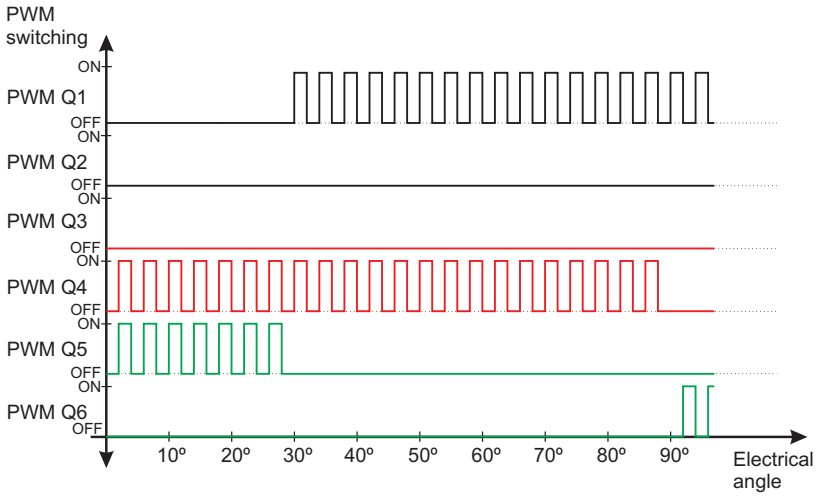

Figure 3-4. Independent Switching of Power Transistors

3.1.2 Complementary Switching of Power Transistors

With complementary switching, two transistors are switched on when the phase of the BLDC motor is connected to the power supply. But there is a difference during freewheeling. With independent switching all the transistors are switched off, the current continues to flow in the same direction through freewheeling diodes, and falls to zero. With complementary switching, the opposite occurs - transistors are switched on during freewheeling. Thus, the current is able to flow in the opposite direction. **Figure 3-5** depicts complementary switching.

Note: Both described switching modes are able to work in bipolar or unipolar mode. For detailed information about bipolar and unipolar modes see the Application Note DSP56F80x MC PWM Module in Motor Control Applications **[2]**. **Figure 3-4** and **Figure 3-5** illustrate the bipolar switching mode. The presented application utilizes the complementary unipolar PWM mode.

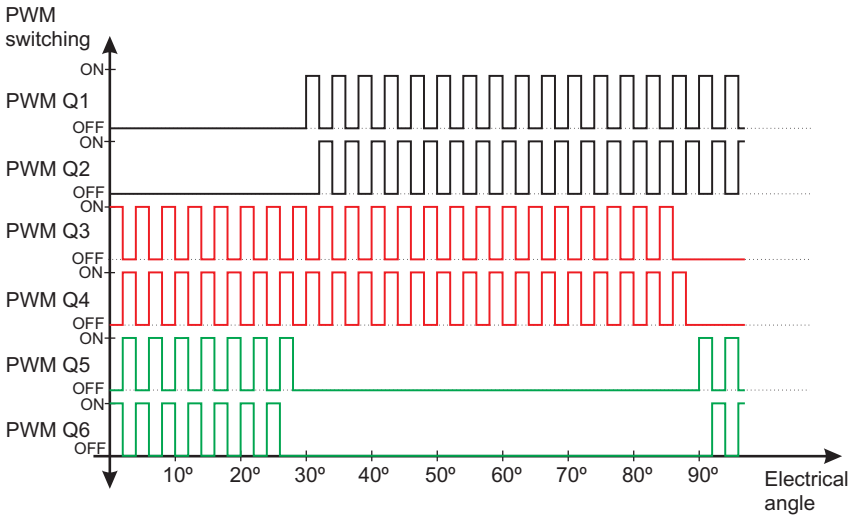

Figure 3-5. Complementary Switching of Power Transistors

3.1.3 Commutation

Commutation creates a rotation field. As was explained, for the proper operation of a BLDC motor, it is necessary to keep the angle between stator and rotor flux close to 90°. With six-step control we get a total of six possible stator flux vectors. The stator flux vector must be changed at a certain rotor position.

The rotor position is usually sensed by Hall Sensors. The Hall Sensors directly detect the commutation moment. The presented application uses the Quadrature Encoder to sense rotor position. The rotor position is usually sensed by Hall Sensors, which directly detected the commutation moment. The presented application uses the Quadrature Encoder to sense rotor position. Therefore the rotor position must be translated to determine the commutation moment.

The electrical revolution can be divided into six sectors. Each sector corresponds to a certain stator flux vector as illustrated in **Figure 3-6**. The commutation sequence is illustrated in tables **Table 3-1** and **Table 3-2**.

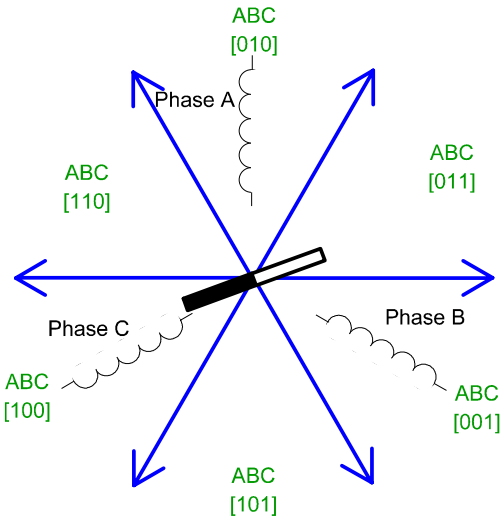

Figure 3-6. Stator Flux Vectors at Six-Step Control

The next two figures depict the commutation process. The actual rotor position in **Figure 3-7** corresponds to the sector ABC[110] (see **Figure 3-6**). The actual voltage pattern can be derived from the **Table 3-2**. Phase A is connected to the positive DC-bus voltage by the transistor Q1, Phase C is connected to the ground by transistor Q6 and Phase B is unpowered.

As soon as the rotor reaches a certain position (see **Figure 3-7**) the sector is changed from ABC[110] to ABC[100]. From **Table 3-2** a new voltage pattern is selected and applied to the BLDC motor.

As can be seen, using a six-step control technique there is no possibility of keeping the angle between the rotor flux and the stator flux precisely at 90°. The actual angle varies from 60° to 120°.

The commutation is repeated per each 60 electrical degrees. The commutation event is critical for its angular (time) accuracy. Any deviation causes the torque ripples and hence speed variation.

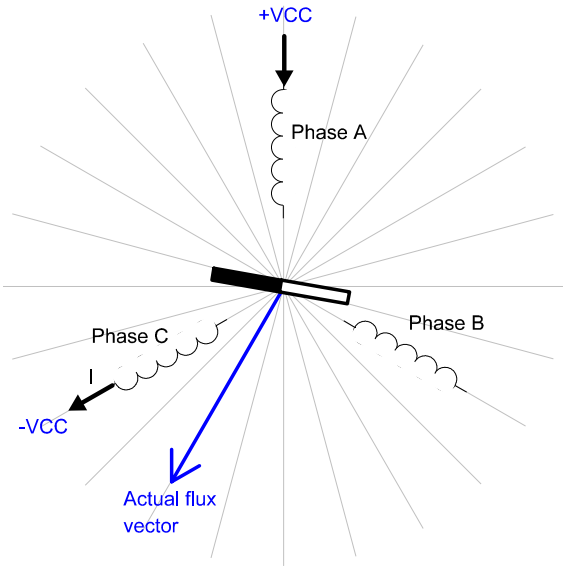

Figure 3-7. Situation Prior to Commutation

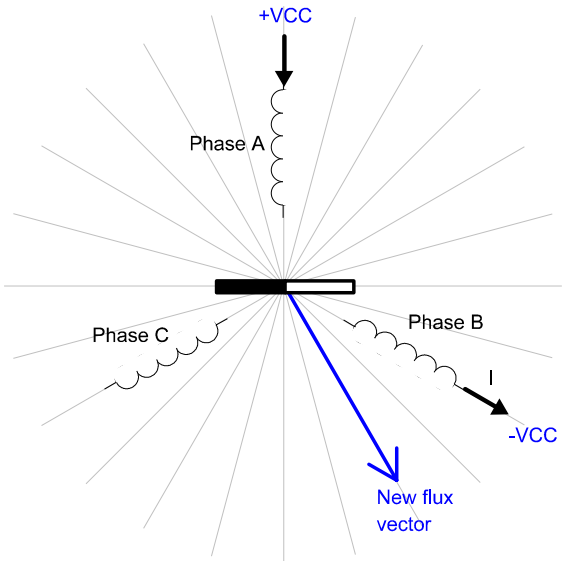

Figure 3-8. Situation Following Commutation

Table 3-1. Commutation Sequence for Clockwise Rotation

Control word [ABC]			Phase A	Phase B	Phase C
1	0	0	$-V_{DCB}$	$+V_{DCB}$	NC
1	0	1	NC	$+V_{DCB}$	$-V_{DCB}$
0	0	1	$+V_{DCB}$	NC	$-V_{DCB}$
0	1	1	$+V_{DCB}$	$-V_{DCB}$	NC
0	1	0	NC	$-V_{DCB}$	$+V_{DCB}$
1	1	0	$-V_{DCB}$	NC	$+V_{DCB}$

Table 3-2. Commutation Sequence for Counter Clockwise Rotation

Control word [ABC]			Phase A	Phase B	Phase C
1	0	0	$+V_{DCB}$	$-V_{DCB}$	NC
1	1	0	$+V_{DCB}$	NC	$-V_{DCB}$
0	1	0	NC	$+V_{DCB}$	$-V_{DCB}$
0	1	1	$-V_{DCB}$	$+V_{DCB}$	NC
0	0	1	$-V_{DCB}$	NC	$+V_{DCB}$
1	0	1	NC	$-V_{DCB}$	$+V_{DCB}$

3.1.3.1 Quadrature Encoder versus Hall Sensors

The BLDC motor application uses the Quadrature Encoder for rotor position sensing. The Quadrature Encoder output consists of three signals. Two phases, A and B, represent the rotor position, and an Index pulse defines the zero position. All Quadrature Encoder signals are depicted in **Figure 3-9**. Compared with Hall Sensors, there are some differences, which affect the control algorithm. The main differences are that the Quadrature Encoder does not give commutation moment and absolute position, as do the Hall Sensors.

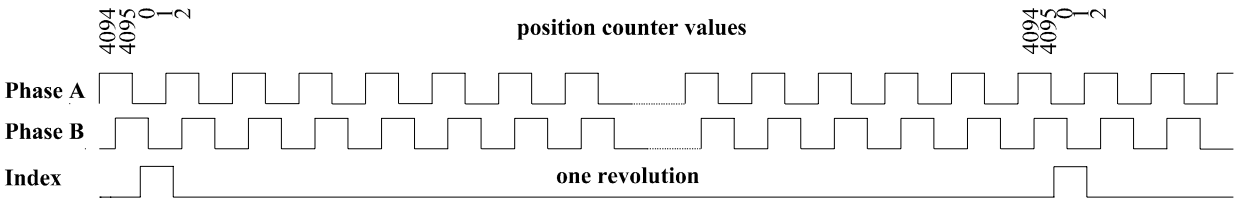

Figure 3-9. Quadrature Encoder output signals

The differences between the Quadrature Encoder and Hall Sensors are summarized in **Table 3-3**.

Table 3-3. Differences between Quadrature Encoder and Hall Sensors

Quadrature Encoder	Hall Sensors
three outputs	three outputs
does not give absolute position	gives absolute position
give precise position	gives 6 events per electrical revolution

The two control algorithms are described later on with the Quadrature Encoder. The first solution is commonly used and periodically scans the Quadrature Encoder. The second solution uses specific advantages of DSP's peripherals and translates the Quadrature Encoder outputs directly into Hall Sensors signals. These internal signals are used as input for the commutation algorithm. This algorithm is implemented in the presented application.

3.1.3.2 Commutation with Periodical Scanning of Quadrature Encoder

The commutation of the BLDC motor is performed in the six defined moments. Since the Quadrature Encoder gives the precise position, the one electrical revolution is divided into six sectors (see **Figure 3-10**). To recognize the commutation moment, it is necessary to scan Quadrature Encoder

position very quickly. The frequency of scanning depends on the maximal rotor speed, the number of pole pairs and on the required precision of commutation moment detection. The same scan frequency as used in the PWM (16kHz) is satisfactory for the common applications.

In this case, the rotor position can be scanned in the moment of a PWM reload interrupt. The algorithm translates the actual position into the one of the six sectors. If a change of sector is detected, the commutation is performed.

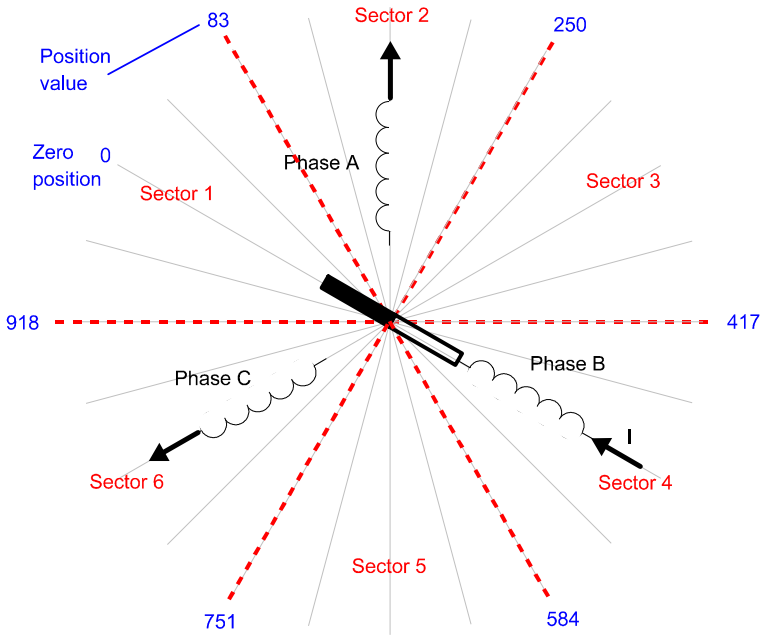

Figure 3-10. Separation of One Electrical Revolution into Six Commutation Sectors

Note: The **Figure 3-10** considers 500 pulses per mech. revolution, both rising and falling edges counting, two pole pairs (500 x 4 / 2 = 1000 pulses per el. revolution)

3.1.3.3 Direct conversion of Quadrature Encoder signals to a commutation sector

A different method is the direct conversion of Encoder output to commutation sectors. The conversion is provided completely automatically by hardware. The advantage of this method is that an interrupt rate depends on the actual motor speed, while in the previous method, there is a constantly high interrupt rate in order to scan the actual motor position. The direct conversion operates in following way:

The electrical revolution is divided into six sectors, as in **Figure 3-10**. The rotor position is scanned by the quadrature counter, which has its inputs connected to the Quadrature Decoder. After rotor alignment to known position, both compare registers are set to values that correspond to sector borders and the counter is set to zero, see **Figure 3-11**. When the motor starts to move, the counter starts to count the pulses of the Quadrature Encoder. Nothing happens until the counter reaches one of the compare values.

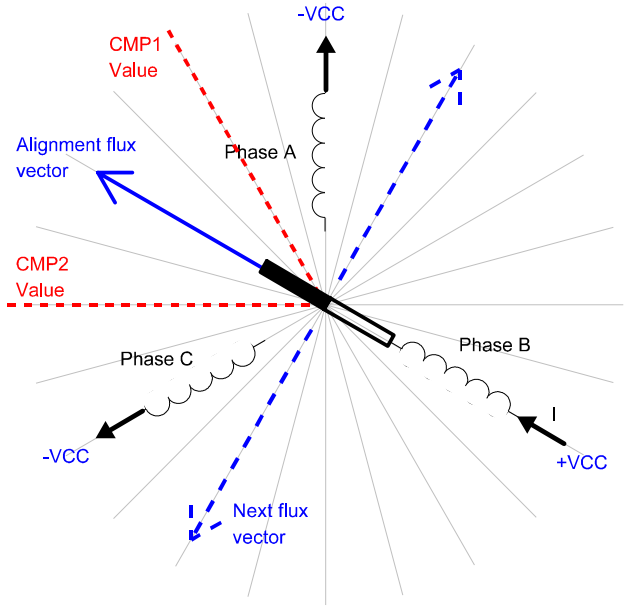

Figure 3-11. Situation Prior to First Compare Following Alignment

As soon as the counter reaches one of the compare values, the commutation interrupt is called. The commutation interrupt recognizes the spin direction and sets new values into compare registers. The new values corresponds to the next commutation sector in clockwise or counter clockwise direction according to actual spin direction. Then the control word, which saves the actual commutation sector, is updated and the new voltage pattern is applied to the motor. The corresponding voltage pattern is taken from **Table 3-2** and **Table 3-2**. **Figure 3-12** depicts the new situation after commutation.

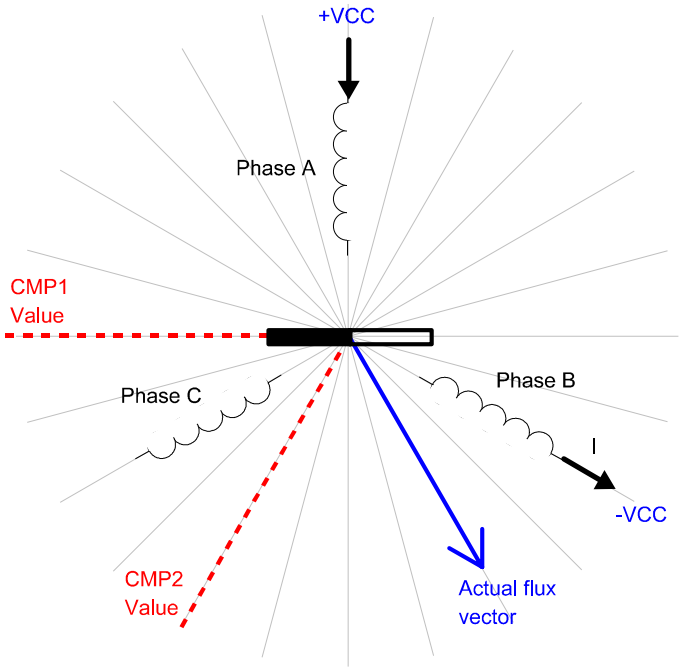

Figure 3-12. Situation Following Commutation

We can see that the commutation interrupt is called six times per electrical revolution. Note that the commutation interrupt is called in the same moment as when we use Hall Sensors. Thus we can use the same control routines, which are included in the SDK.

3.1.4 Position Alignment

Since the Quadrature Encoder doesn't give the absolute position, we need to know exactly the rotor position before the motor is started. One possible, and very easily implemented, method is the rotor alignment to a predefined position. The motor is powered by a defined static voltage pattern and the rotor aligns to predefined position. This alignment is only done once, during the initial motor start up. The **Figure 3-13** shows the position of the aligned rotor. After alignment, the compare registers of the position counter are set to +/- 30 electric degrees from the alignment position, in order to preset the commutation sector. (according to **3.1.3.3**) The next voltage vector is set to be orthogonal to the alignment position. The resultant direction of the voltage vector is given by polarity of applied voltage.

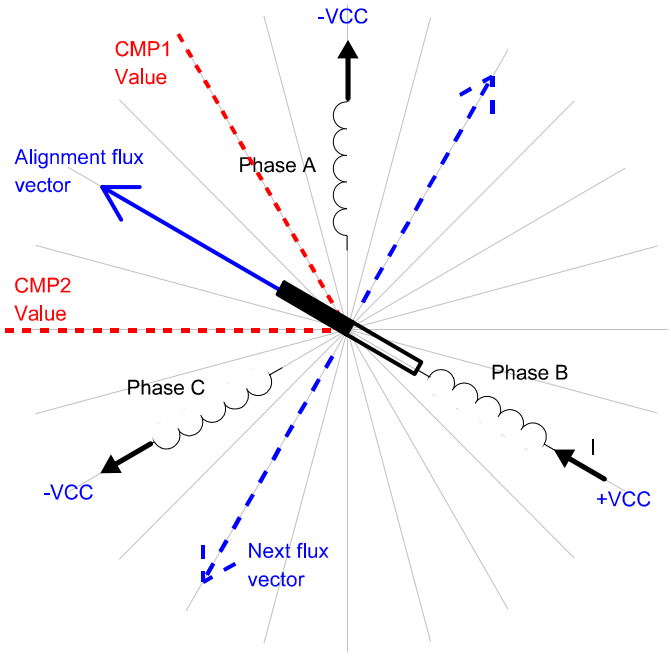

Figure 3-13. Alignment Rotor Position

3.1.5 Speed Control

Commutation ensures the proper rotor rotation of the BLDC motor, while the motor speed only depends on the amplitude of the applied voltage. This amplitude is changed by the PWM technique. The required speed is controlled by a speed controller, implemented as a conventional PI controller. The PI controller compares the actual speed to the required speed and, using the difference, calculates duty cycle with the voltage amplitude required to correct the discrepancy.

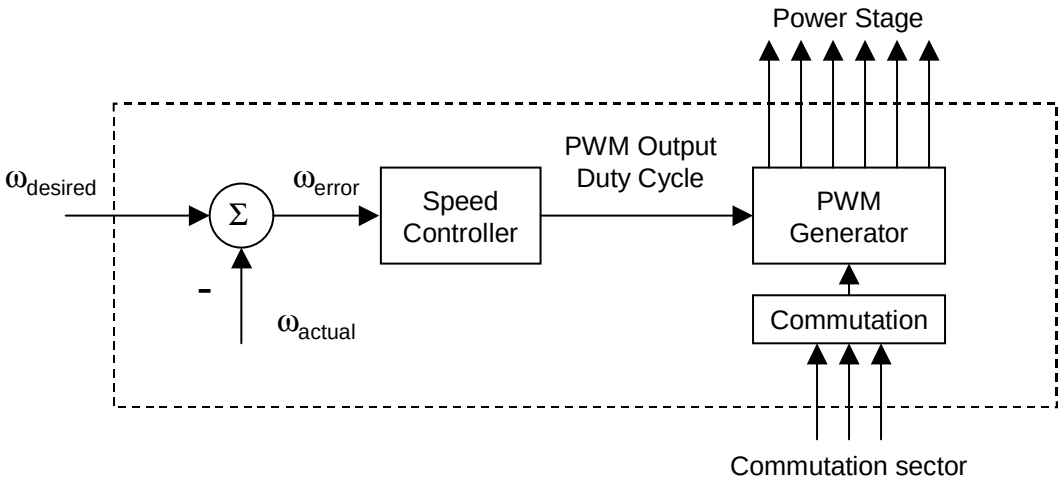

Figure 3-14. Speed Controller

The speed controller calculates a Proportional-Integral (PI) algorithm according to equations below:

$$u(t) = K_c\left[e(t) + \frac{1}{T_I}\int_0^{\tau} e(\tau)d\tau\right] \quad \text{(EQ 3-1.)}$$

After transformation to a discrete time domain using an integral approximation by a Backward Euler method, we get the following equations for the numerical PI controller calculation:

$$u(k) = u_P(k) + u_I(k) \quad \text{(EQ 3-2.)}$$

$$u_P(k) = K_c \cdot e(k) \quad \text{(EQ 3-3.)}$$

$$u_I(k) = u_I(k-1) + K_c\frac{T}{T_I} \cdot e(k) \quad \text{(EQ 3-4.)}$$

where:

e(t), e(τ)	=	Input error in time t, τ
e(k)	=	Input error in step k
w(k)	=	Desired value in step k
m(k)	=	Measured value in step k
u(k)	=	Controller output in step k
u_p(k)	=	Proportional output portion in step k
u_I(k)	=	Integral output portion in step k
u_I(k-1)	=	Integral output portion in step k-1
T_I	=	Integral time constant
T	=	Sampling time
K_c	=	Controller gain
t, τ	=	Time
p	=	Laplace variable

4. System Concept

4.1 System Outline

The system is designed to drive a 3-phase BLDC motor. The application meets the following performance specification:

- Voltage control of BLDC motor using Quadrature Encoder
- Targeted for DSP56F803EVM, DSP56F805EVM, DSP56F807EVM
- Running on 3-phase EVM Motor Board
- Control technique incorporates:
  - Voltage BLDC motor control with speed-closed loop
  - Both direction of rotation
  - Motoring mode
  - Start from any motor position without rotor alignment
  - Minimal speed 50 RPM
  - Maximal speed 1000 RPM (limited by power supply)
- Manual interface (Start/Stop switch, Up/Down push button control, Led indication)
- PC master software control interface (motor start/stop, speed set-up)
- PC master software monitor
  - PC master software graphical Control Page (required speed, actual motor speed, start/stop status, DC-Bus voltage level, system status)
  - PC master software Speed Scope (observes actual & desired speeds)
- DC-Bus under-voltage fault protection

The introduced BLDC drive is designed to power a low-voltage BLDC motor equipped with Quadrature Encoder, which is supplied with the EVM Motor Board. The motor has the following specifications:

Table 4-1. Specifications of the 3-ph BLDC Motor

Motor Specification:	Motor Type:	3-Phase BLDC Motor 4Poles
	Speed Range:	< 5000 RPM
	Line Voltage:	60V
	Phase Current:	2A
Position Sensor Specification:	Sensor 1 Type:	3-Phase Hall Sensors
	Sensor 2 Type:	Quadrature Encoder 500 Pulses Per Revolution

4.2 Application Description

A standard system concept is chosen for the drive (see **Figure 4-1**). The system incorporates the following hardware boards:

- Power Supply 12V DC, 4Amps
- EVM Motor Board
- BLDC Motor IB23810 with Quadrature Encoder
- Evaluation Board DSP56F803, DSP56F805 or DSP56F807

The DSP runs the main control algorithm. According to the user interface and feedback signals, it generates 3-phase PWM output signals for the BLDC inverter.

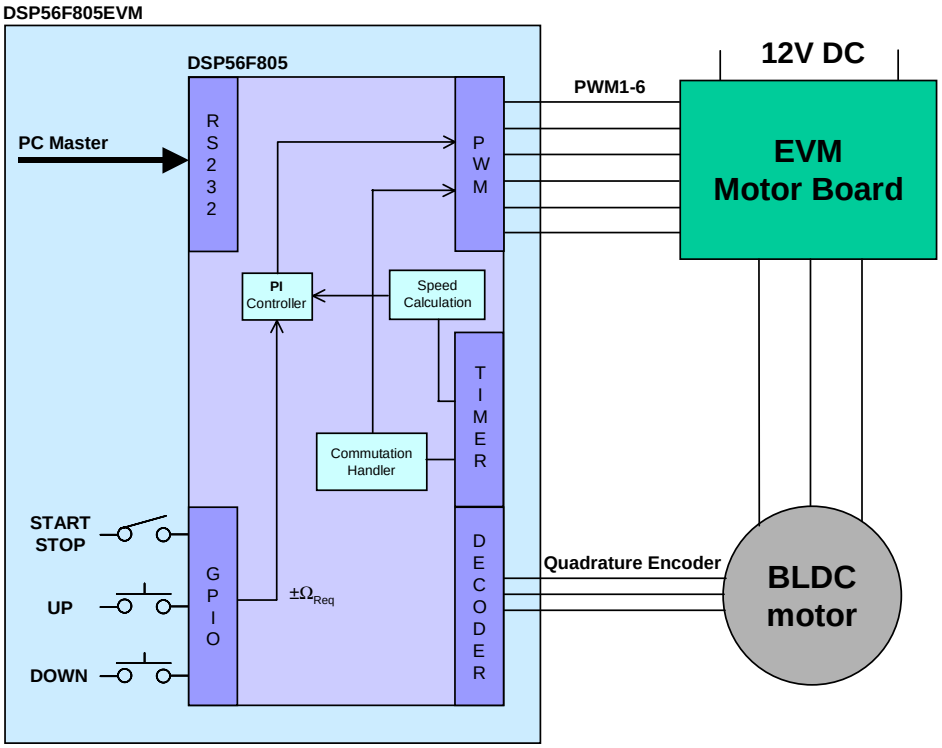

Figure 4-1. System Concept

The control process is as follows:

The state of the user interface is periodically scanned, while the speed of the motor is measured on each rising edge from the Quadrature Encoder (only one phase is used for speed measurement). According to the state of the control signals (Start/Stop switch, speed up/down buttons) the speed command is calculated. The comparison between the actual speed command and the measured speed generates a speed error. The speed error is brought to the speed PI controller that generates a new corrected duty cycle. The duty cycle value together with commutation algorithm creates the PWM output signals for the BLDC power stage.

The Quadrature Encoder signals are converted to six interrupts per electrical revolution. This interrupt provides the commutation algorithm.

In the case of under-voltage, the PWM outputs are disabled and the fault state is displayed.

4.3 Hardware Implementation

As already stated, the application runs on Motorola motor control DSPs using the DSP EVM Boards and a dedicated 3-phase BLDC platform.

The application can be controlled by the following Motorola motor control DSPs:

- DSP56F803
- DSP56F805
- DSP56F807

The application can run on an EVM motor board.

The application HW setup is shown in **Figure 4-2**. The system hardware setup for a particular DSP varies only by the EVM board used. The application software is identical for all DSPs. The EVM and the chip differences are handled by the SDK off-chip drivers for the particular DSP EVM board.

Detailed application HW setup can be found in the document **Targeting_DSP5680x_Platform** that is part of the SDK documentation.

Dedicated User's Manuals describe the individual boards in detail. The User's Manual incorporates a schematic of the board, a description of individual function blocks and a bill of materials for the board. Individual boards can be ordered from Motorola as standard products. The following chapter illustrates the configuration of an EVM motor board, together with references to the documentation. Descriptions of all the mentioned boards and documents can be found at: *http://www.motorola.com/*.

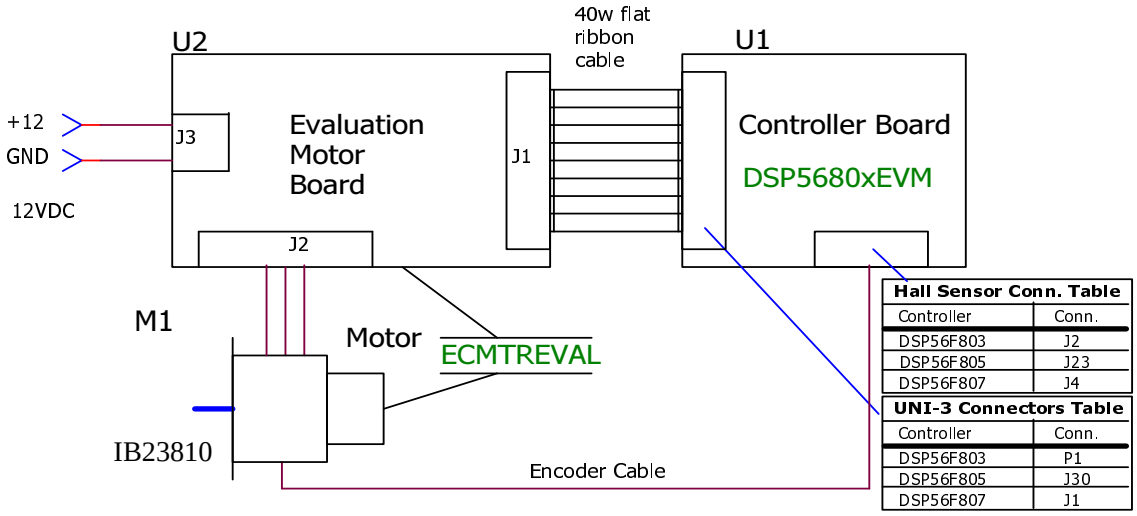

Figure 4-2. Low-Voltage Evaluation Motor HW System Configuration

All the system parts are supplied and documented according the following references:

- M1 - IB23810 Motor
  - — supplied in kit ECMTREVAL - Evaluation Motor Board Kit
- U2 EVM Motor Board:
  - — supplied in kit with IB23810 Motor: ECMTREVAL - Evaluation Motor Board Kit
  - — described in: **Evaluation Motor Board User's Manual**
- U1 CONTROLLER BOARD for DSP56F803:
  - — supplied as: DSP56803EVM
  - — described in: **DSP Evaluation Module Hardware User's Manual**

- or U1 CONTROLLER BOARD for DSP56F805:
  - supplied as: DSP56805EVM
  - described in: **DSP Evaluation Module Hardware User's Manual**
- or U1 CONTROLLER BOARD for DSP56F807:
  - supplied as: DSP56807EVM
  - described in: **DSP Evaluation Module Hardware User's Manual**

5. Software Design

This section describes the design of the software blocks for the drive. The software will be described in terms of:

- Control Algorithm Data Flow
- Software Implementation

5.1 Data Flow

The control algorithm of a close loop BLDC drive is described in **Figure 5-1**. The individual processes are described in the following sections.

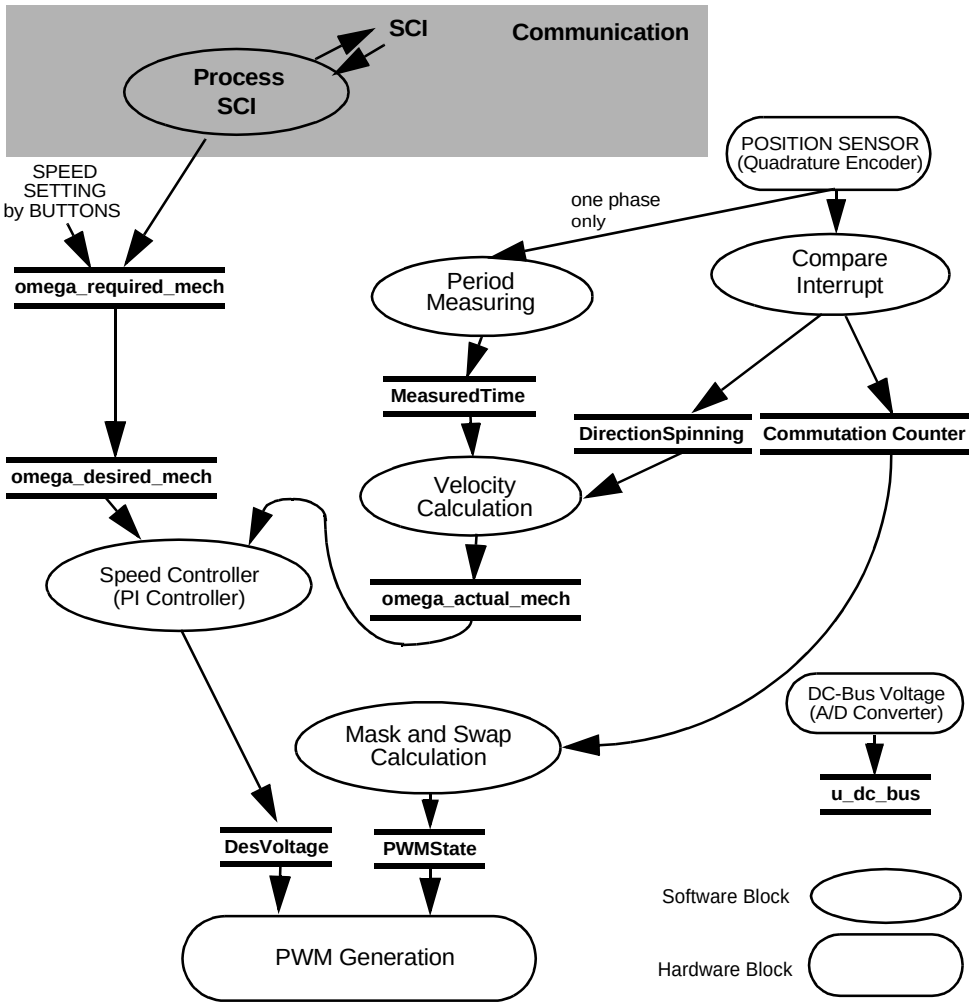

Figure 5-1. Main Data Flow

The main data flow can be divided to four parts:

- Speed control
- Velocity calculation
- Rotor commutation
- DC-Bus voltage measurement

Speed control starts with the required speed `omega_required_mech`. This variable is set by user buttons or remotely by the PC, within allowed limits. The variable `omega_required_mech` is copied to `omega_desired_mech` at a defined moment. This variable is used as a shadow variable to avoid change of the required speed from the PC at any time. The variable `omega_desired_mech` is input to the speed PI controller as a reference value.

`MeasuredTime` incorporates a time period of one phase of Quadrature Encoder. The time period is used for a speed calculation. Calculated speed, `omega_actual_mech`, is input to the speed PI controller as a secondary input. The PI controller output determines the duty cycle of the generated PWM output signals.

For the commutation, the Timer A0, set as a quadrature counter, is used. As soon as the timer A0 reaches any of compare values, a compare interrupt is called. The interrupt routine saves the actual commutation sector to `CommutationCounter`. The `CommutationCounter` variable is input to the mask and swap calculation, which calculates the final shape of the output voltage. The output variable `PWMState` is written directly to the PWM block. The next task, which is provided by the interrupt routine, is calculation of the spin direction. The result, `DirectionSpinning`, is used for the speed calculation. Finally, the routine updates both the compare registers of Timer A0.

A variable `u_dc_bus` contains the actual DC-Bus voltage. The value is used for an under-voltage detection.

5.1.1 Compare Interrupt

The process Compare Interrupt is executed when the timer A0 reaches any of compare values. The process identifies the spin direction. According to the spin direction, the process updates the compare values to define the limit for the new commutation sector. The number of pulses for one commutation sector is given by constant `NORMAL_COMMUTATION_INTERVAL`. Where the number of pulses is not an integer there is a constant, `SHORT_COMMUTATION_INTERVAL`, which is applied once per electrical revolution. This constant ensures that the number of pulses counted through one electrical revolution is exactly the same as the number of pulses of the Quadrature Encoder. Where the number of pulses for one commutation sector is an integer the constant `SHORT_COMMUTATION_INTERVAL` is not implemented.

5.1.2 Period Measuring and Velocity Calculation

The processes, Period Measuring and Velocity Calculation, read the time between the adjacent edges of one phase of the Quadrature Encoder, and calculate the actual motor speed `omega_actual_mech`.

5.1.3 Speed Controller

This process compares the required and actual speeds and calculates the duty cycle of the PWM output signals. For detailed information see **Section 3.1.5, Speed Control**.

5.1.4 Mask and Swap Calculation

This process performs a rotor commutation. As already mentioned, only two phases are powered by a six-step control. The proper PWM output can be generated by changing the PWM value (duty cycle) registers only. This has two disadvantages. Firstly the speed controller, which changes the duty cycle, affects the commutation algorithm (as a result of changing the duty cycle). Secondly, change in the duty cycle is synchronized with PWM reload, which may give rise to a delay between a proper commutation moment and the PWM reload. This is especially pronounced at high speeds, when the commutation period is very short.

The DSP56F80x family has two features dedicated to BLDC motor control: The ability to swap odd and even PWM generator outputs, and the ability to mask (disable) any PWM generator outputs. Thus the same PWM pulse can be applied on the selected upper and lower switch of the inverter. These two features allow the creation of a rotational field without changing the contents of the PWM value registers (duty cycle). The influence of masking and swapping on the PWM generator outputs is illustrated in **Figure 5-2**. Detailed information about PWM settings for BLDC motors can be found in the Application Note DSP56F80x MC PWM Module in Motor Control Applications **[2]**.

The commutation algorithm `bldchsCommHandlerComp` calculates the output `PWMState`, based on the actual control word (`CommutationCounter`). The structure of `PWMState` consists of two parts (variables). The first part (`PWMState.Swap`) defines the swapping of phases, and the second part (`PWMState.Mask`) defines the phase masking. These values are written directly to the PWM generator. The swapping value is written to the PWM Channel Control Register and the masking value to the PWM Output Control Register. The following paragraph describes in detail the commutation transition from the one to the next commutation sector as is depicted on **Figure 3-7**, **Figure 3-8** and **Figure 5-2**.

For example, the even PWM value registers are set to 75% duty cycle. The odd PWM value registers are set to complementary value 100% - 75% = 25% duty cycle. The rotor in **Figure 3-7** is situated in sector ABC[110]. The state of masking and swapping is as follow (see **Table 3-2**):

- Phase A

  not masked, not swapped (the Phase A is connected to positive DC-Bus voltage)
- Phase B

  masked/disabled
- Phase C

  not masked, swapped (the Phase C is connected to negative DC-Bus voltage. The swap of Phase B provides the identical pulses on the upper switch of Phase A and the lower switch of Phase C and conversely on the lower switch of Phase A and the upper switch Phase C)

As soon as the rotor passes the sector border (**Table 3-8**) the commutation is done in two steps:

- swap Phase B (unconnected phase)
- mask/disable Phase C; unmask/enable Phase B

After commutation the state of masking and swapping is (see **Table 3-2**):

- Phase A

  not masked, not swapped (the Phase A is connected to positive DC-Bus voltage)
- Phase B

  not masked, swapped (the Phase B is connected to negative DC-Bus voltage.)
- Phase C

  masked/disabled

This commutation process is repeated six times per electrical revolution.

Note: In complementary switching mode, it is necessary to use the software control feature for masking. For independent mode, it is possible to use the masking feature in the PWM Channel Control Register. This feature works properly in independent mode only.

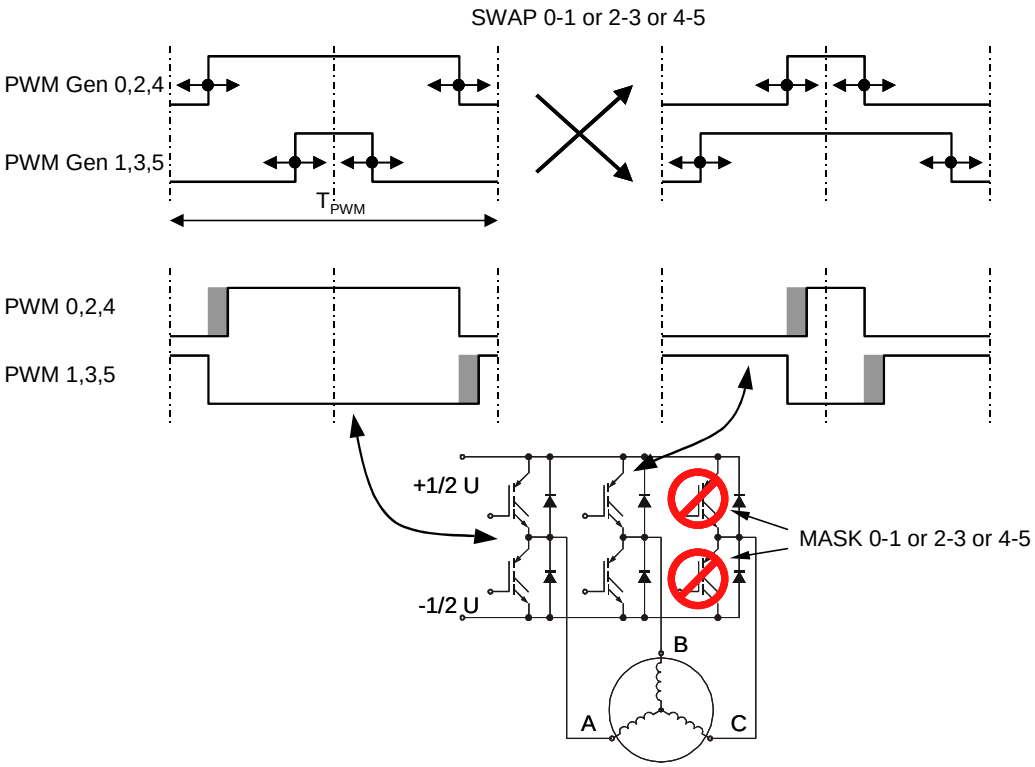

Figure 5-2. PWM Swapping and Masking for BLDC Motor

5.2 Software Implementation

The general software diagram shows the Main routine entered from Reset and the interrupt states (see **Figure 5-3**).

The Main routine initializes both the DSP and the application, then goes into an infinite background loop. This loop contains an application State Machine.

The following interrupt service routines are utilized:

- Compare (TimerA0) - services signals generated by the Quadrature Encoder
- Input Capture ISR (TimerA1) - services period measurement for speed calculation
- Timer ISR - services the speed controller and LED diode blinking
- Push Button Up ISR and Push Button Down ISR - services the Up and Down push buttons
- SCI ISR - services communication with the PC master software

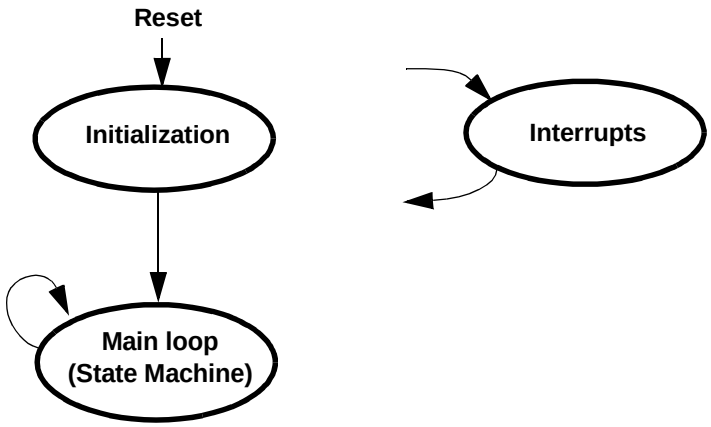

Figure 5-3. State Diagram - General Overview

5.2.1 Initialization

The Main Routine provides initialization of the DSP:

- Disables Interrupts
- Initializes DSP PLL
- Disables COP and LVI
- Initializes the POSIX Timer for a time base reference of 1ms
- Initializes LED
- Initializes the PWM module:
  - Center-aligned complementary PWM mode, positive polarity
  - PWM modulus - defines PWM frequency
  - PWM deadtime - defines PWM deadtime
  - Disable faults
- Initializes Quadrature Decoder
  - Sets the on-chip digital filter of the Quadrature Decoder inputs
  - Connects Quadrature Decoder signals to Quad TimerA
- Initializes Quad TimerA - channels A0
  - set Count Mode to Quadrature Count
  - set Input Source to Input 1
  - set Input Polarity to Normal
  - set Secondary Input Source to Input 0
  - set CountFrequency to Repeatedly
  - set Count Length to Past Compare
  - set Count Direction to Down
  - disable Capture Mode
  - associate Callback On Input Edge to ISRQTimer

- Initializes Quad TimerA - channel A1
  - set Count Mode to Count
  - set Input Source to Bus Clock / 128
  - set Input Polarity to Normal
  - set Secondary Input Source to Input 2
  - set Count Frequency to Repeatedly
  - set Count Length to Past Compare
  - set Count Direction to Up
  - set Capture Mode = BothEdges
  - associate Callback On Input Edge to CallbackOnNewEdge
  - associate CallbackOnOverflow to CallbackOnOverload
- Sets-up I/O ports (brake, switch, push buttons)
  - Brake, LED, switch on GPIO
  - Push buttons on interrupts IRQ0, IRQ1
- Initializes the Analog-to-Digital Converter
  - ADC set for sequential sampling, single conversion
  - Channel 0 = DC-Bus voltage
- Initializes control algorithm (speed controller, control algorithm parameters)
- Enables interrupts
- Starts ADC conversion

5.2.2 Interrupts

The interrupt handlers have the following functions:

- *Compare Interrupt Handler (Timer A0)* calculates the actual commutation sector. That value is input to the commutation algorithm. The description of the commutation is in **Section 3.1.3** and **Section 5.1.4**.
- *Input Capture Interrupt Handler (Timer A1)* reads the time between the two subsequent IC edges in one phase of the Quadrature Encoder, which is used for speed calculation.
- *POSIX Timer Interrupt Handler* generates the time base 1ms. The routine, called within this time base, blinks the green LED diode, reads the result of the ADC conversion, calculates the speed and provides the speed controller.
- *Push Button Interrupt Handler* takes care of the push button service. The *UpButton Interrupt Handler* increments the desired speed, the *DownButton Interrupt Handler* decrements the desired speed.
- *PC and SCI Interrupt Handler*s provide SCI communication and service routines for the PC master software. These routines are fully independent of the motor control tasks.

5.2.3 Drive State Machine

The drive can be in any of the states shown in **Figure 5-4**, which shows the transition conditions among the drive states. The user is able to recognize the current state, by a blinking green LED diode. In case of the *init* and *stop* state, the green LED diode blinks with a frequency of 2 Hz. In the *fault* state, the green LED diode blinks with a frequency of 8 Hz. During the *running* state the green LED diode is continuously lit.

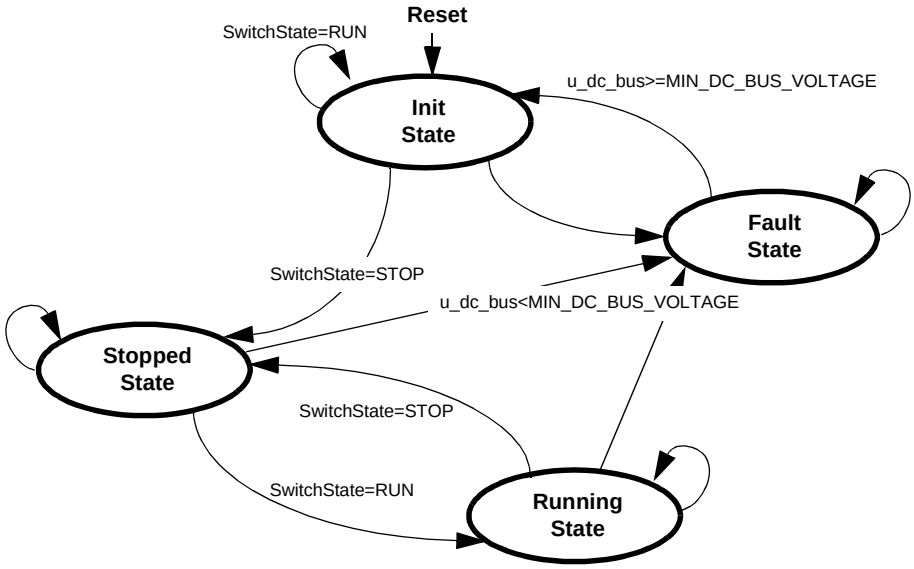

Figure 5-4. Drive State Machine Transitions

6. Implementation Notes

6.1 Scaling of Quantities

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

$$-1.0 \leq SF \leq +1.0 - 2^{-[N-1]} \qquad \text{(EQ 6-1.)}$$

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

The following equation shows the relationship between real and fractional representations:

$$\text{Fractional Value} = \frac{\text{Real Value}}{\text{Real Quantity Range}} \qquad \text{(EQ 6-2.)}$$

where:

Fractional Value is a fractional representation of the real value [Frac16]

Real Value is the real value of the quantity [V, A, RPM, etc.]

Real Quantity Range is the maximal range of the quantity, defined in the application [V, A, RPM, etc.]

6.1.1 DC-Bus Voltage Scaling

The DC-Bus voltage sense is defined by following equation:

$$u\_dc\_bus = \frac{V_{DC\_BUS}}{V_{MAX}} \cdot 32767$$

Where: u_dc_bus = variable of DC-Bus voltage, V_{DC_BUS} = measured DC-Bus voltage, V_{MAX} = max. measurable DC-Bus voltage.

V_{MAX} = 16V for the EVM Motor Board

6.1.2 PI Controller Parameters

The constant P was chosen as 0.2 ($26214 * 2^{-17}$), and the constant I was chosen as 0.3 ($31457 * 2^{-20}$) or 0.12 ($31457 * 2^{-18}$). A better response to speed error is achieved when the constant I is changed according to the actual speed. The constant I is equal to 0.3 from 50 to 200 RPM. Over 200 RPM the constant I is equal to 0.12. These controller parameters were experimentally tuned.

6.1.3 Velocity Calculation

The constant OMEGA_ACTUAL_MECH_CONST is defined by the following equations:

position difference = 1/500 rev (given by each rising edge of one phase of the Quadrature Encoder and two pole pairs motor)

max. period time = 0.008 s (chosen according to required min. speed)

v_{min} = 60*(position difference)/(max. period time) = 15 RPM

$v_{max} = 100 * v_{min}$ = 1500 RPM (chosen according to required max. speed)

OMEGA_ACTUAL_MECH_CONST = $32767 * v_{min}/v_{max}$ = 327

7. SDK Implementation

The Motorola Embedded SDK is a collection of APIs, libraries, services, rules and guidelines. This software infrastructure is designed to let DSP5680x software developers create high-level, efficient, and portable code. The application code is available in the SDK. This chapter describes how the BLDC motor control application is written under the SDK.

7.1 Drivers and Library Functions

The BLDC motor control application uses the following drivers:

- ADC driver
- Timer driver
- Quad Timer driver
- Quadrature Decoder driver
- PWM driver
- LED driver
- Switch driver
- Button driver

All drivers except the Timer driver are included in the *bsp.lib* library. The Timer driver is included in the *sys.lib* library.

The BLDC motor control application uses the following library functions:

- bldchsCommHandlerComp (BLDC motor commutation algorithm; *bldc.lib* library)
- controllerPItype1 (standard PI controller; *mcfunc.lib* library)
- switchcontrol (switch control; *mcfunc.lib* library)

7.2 Appconfig.h File

The purpose of the *appconfig.h* file is to provide a mechanism for overwriting the default configuration settings which are defined in the *config.h* file.

There are two *appconfig.h* files. The first *appconfig.h* file is dedicated to External RAM (..*ConfigExtRam* directory) and the second one is dedicated to FLASH memory (..*ConfigFlash* directory). In the case of the BLDC motor control application, both files are identical.

The *appconfig.h* file can be divided into two sections. The first section defines which components of the SDK libraries are included in the application; the second part overwrites the standard settings of the components during their initialization.

7.3 Initialization of Drivers

Each peripheral on the DSP chip or on the EVM board is accessible through a driver. The driver initialization of each peripheral used is described in this chapter. For a detailed description of drivers see the document **Embedded SDK Targeting Motorola DSP5680x Platform**.

The following steps are required to use the driver:

- Include driver support in the *appconfig.h* file
- Fill the configuration structure in the application code for specific drivers (depends on driver type)
- Initialize the configuration setting in *appconfig.h* for specific drivers (depends on driver type)
- Call the *open* (create) function

Access to individual driver functions is provided by the *ioctl* function call.

7.4 Interrupts

The SDK serves the interrupt routine calls and automatically clears interrupt flags. The user defines the callback functions called during interrupts. The callback functions are assigned during driver initialization - `open()`. Callback function assignment is defined as one item of the initialization structure which is used as a parameter of the function `open()`. Some drivers define the callback function in the *appconfig.h* file.

7.5 PC Master Software

PC master software was designed to provide an application debugging, diagnostic and demonstration tool for the development of algorithms and applications. It runs on a PC connected to the DSP EVM via an RS232 serial cable. A small program resident in the DSP communicates with the PC master software to parse commands, return status information to the PC and process control information from the PC. PC master software, executing on a PC, uses part of Microsoft Internet Explorer as the user interface.

PC master software is part of the Motorola Embedded SDK and may be selectively installed during SDK installation.

To enable PC master software operation on the DSP target board application, the following lines must be added to the *appconfig.h* file:

```
#define INCLUDE_SCI          /* SCI support */
#define INCLUDE_PCMASTER     /* PC master software support */
```

This automatically includes the SCI driver and installs all necessary services.

The default baud rate of the SCI communication is 9600Bd. It is set automatically by the PC master software driver and can be changed if needed.

A detailed description of PC master software is provided by the dedicated User's Manual **[12]**.

The 3-phase BLDC motor control application utilizes PC master software for remote control from the PC. It enables the user to:

- Start/stop control
- Set the motor speed

Variables read by the PC master software and displayed to the user are:

- Required and actual motor speed
- Application operational mode
- Start/stop status
- DC-bus voltage

The PC master software Control Page is illustrated in **Figure 7-1**. The profiles of the required and actual speeds can be seen in the Speed Scope window.

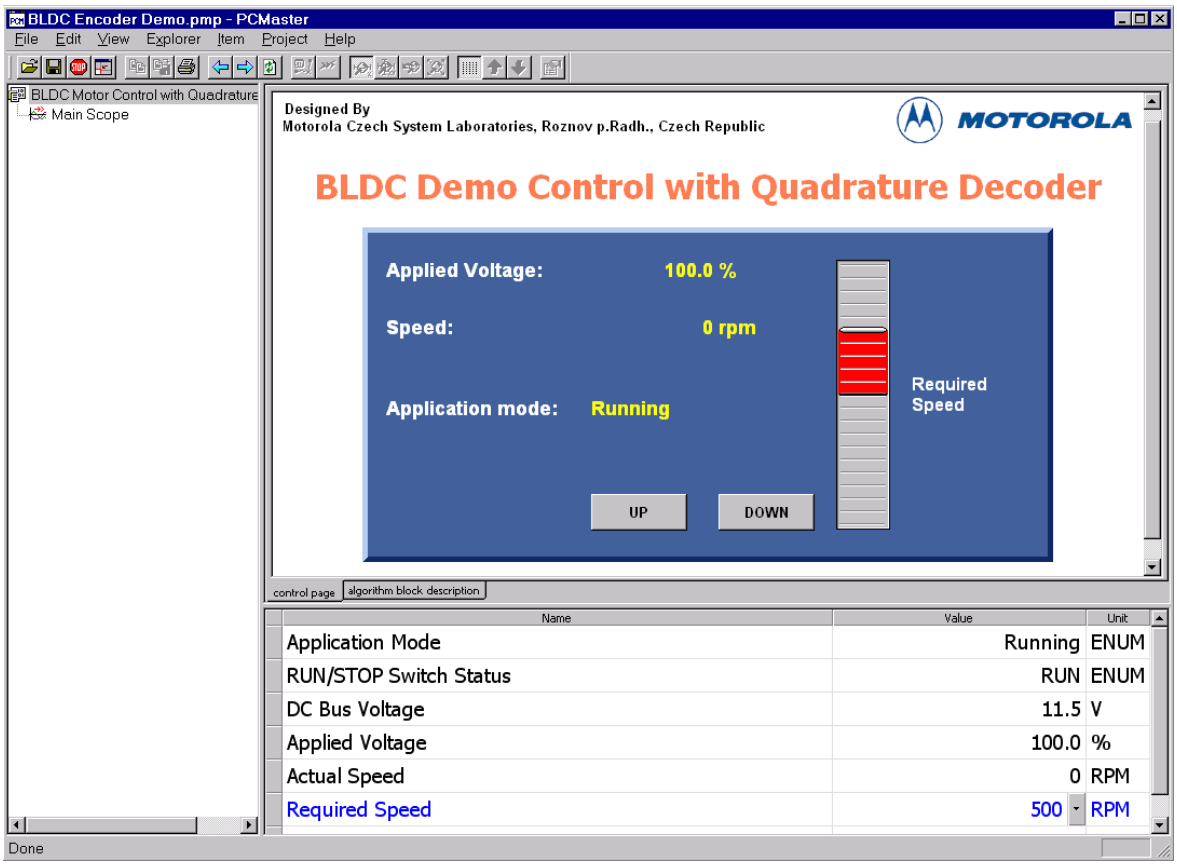

Figure 7-1. PC Control Window

8. DSP Usage

Table 8-1 shows how much memory is needed to run the 3-phase BLDC drive in a speed-closed loop using Quadrature Encoder. A majority of the DSP's memory is still available for other tasks.

Table 8-1. RAM and FLASH Memory Usage for SDK2.4 and CW5.01

Memory (in 16 bit Words)	Available DSP56F803 DSP56F805	Available DSP56F807	Used Application	Used Application without PC master software, SCI
Program FLASH	32K	60K	8905	5040
Data FLASH	4k	8K	258	258
Program RAM	512	2K	101	101
Data RAM	2K	4K	673	338

9. References

[1] **Brushless DC Motor Control using the MC68HC708MC4**, John Deatherage and Jeff Hunsinger, AN1702/D, Motorola

[2] **DSP56F80x MC PWM Module in Motor Control Applications**, Leos Chalupa, AN1927/D, Motorola

[3] **Design of Brushless Permanent-magnet Motors**, J.R. Hendershot JR and T.J.E. Miller, Magna Physics Publishing and Clarendon Press, 1994

[4] **CodeWarrior for Motorola DSP56800 Embedded Systems**, CWDSP56800, Metrowerks 2001

[5] **DSP56F800 16-bit Digital Signal Processor, Family Manual**, DSP56F800FM/D, Motorola 2001

[6] **DSP56F80x 16-bit Digital Signal Processor, User's Manual**, DSP56F801-7UM/D, Motorola 2001

[7] **DSP56F803 Evaluation Module Hardware User's Manual**, DSP56F803EVMUM/D, Motorola 2001

[8] **DSP56F805 Evaluation Module Hardware User's Manual**, DSP56F805EVMUM/D, Motorola 2001

[9] **DSP56F807 Evaluation Module Hardware User's Manual**, DSP56F807EVMUM/D, Motorola 2001

[10] **Evaluation Motor Board User's Manual,** MEMCEVMBUM/D, Motorola

[11] **Embedded Software Development Kit for 56800/56800E**, MSW3SDK000AA, available on Motorola SPS web page, Motorola 2001

[12] **User Manual** for PC master software, included in the SDK documentation, Motorola 2001

Motorola SPS web page: *http://www.motorola.com/*

[13] **3-Phase BLDC Motor Control with Hall Sensors Using DSP56F80x**, Pavel Grasblum, AN1916/D, Motorola 2001

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors/

MOTOROLA

AN1915/D