Freescale Semiconductor
Application Note

AN1913
Rev. 3, 11/2005

3-phase BLDC Motor Control with Sensorless Back-EMF ADC Zero Crossing Detection using 56F80x

Design of Motor Control Application Based on the Software Development Kit SDK

Libor Prokop,
Leos Chalupa

Contents

1. Introduction

This Application Note describes the design of a 3-phase sensorless BLDC motor drive with Back-EMF Zero Crossing using an AD converter. It is based on Freescale's 56F80x family dedicated for motor control applications.

The concept of the application is that of a speed-closed loop drive using an AD converter for Back-EMF Zero Crossing technique position detection. It serves as an example of a sensorless BLDC motor control system using Freescale's Digital Signal Processor (DSC) with SDK support. It also illustrates the usage of dedicated motor control on chip peripherals, software drivers and software libraries that are included in the SDK.

This Application Note includes a description of the controller's features, basic BLDC motor theory, system design concept, hardware implementation and software design including the PC master software visualization tool inclusion.

2. DSC Advantages and Features

Freescale's 56F80x family are well suited for digital motor control, combining the DSP's calculation capability with the MCU's controller features on a single chip. These devices offer many dedicated peripherals like Pulse Width Modulation (PWM) module, Analog-to-Digital Converter (ADC), Multi-function Quadrature Decoder, Timers, communication peripherals (SCI, SPI, CAN), and on-chip Flash and RAM. Generally, all family members are well suited for motor control applications.

The 56F805 device provides the following peripheral blocks:

- Two Pulse Width Modulator modules each with six PWM outputs, three Current Sense inputs, and four Fault inputs, fault tolerant design with dead-time insertion; supports both center- and edge-aligned modes
- Two 12-bit Analog-to-Digital Converters (ADC) which support two simultaneous conversions; ADC and PWM modules can be synchronized
- Two Quadrature Decoders each with four inputs or two additional Quad Timers
- Two dedicated General Purpose Quad Timers totaling six pins: Timer C with two pins and Timer D with four pins
- CAN 2.0 B Module with 2-pin port for transmit and receive
- Two Serial Communication Interfaces, each with two pins (or four additional GPIO lines)
- Serial Peripheral Interface (SPI) with configurable four-pin port (or four additional GPIO lines)
- 14 dedicated General Purpose I/O (GPIO) pins, 18 multiplexed GPIO pins
- Computer Operating Properly (COP) watchdog timer
- Two dedicated external interrupt pins
- External reset input pin for hardware reset
- External reset output pin for system reset
- JTAG/On-Chip Emulation (OnCE™) module for unobtrusive, processor speed-independent debugging
- Software-programmable, Phase Lock Loop-based frequency synthesizer for the core clock

Table 2-1. Memory Configuration

	56F801	56F803	56F805	56F807
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

The BLDC motor control greatly benefits 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
- Deadtime insertion
- Deadtime distortion correction using 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 16 period
- Individual, software-controlled PWM output
- Programmable fault protection
- 20-mA current sink capability on PWM pins
- Write-protectable registers

The PWM module is capable of providing the six PWM signals with bipolar switching (the diagonal power switches are driven by the same signal). In addition, the PWM provides the six-step BLDC commutation control (where one motor phase is left unpowered so 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.

An Analog-to-Digital Converter (ADC) module has the following features:

- Dual ADCs per module
- Eight input channels per module
- 12-bit resolution
- Monotonic over entire range with no missing codes
- Simultaneous conversion mode
- Single conversion time in 1.7 us; eight conversions in 5.3 us (using simultaneous mode)
- Interrupt generating capabilities at: end-of-scan, Zero Crossing, and high/low limit check
- Two output formats: two’s compliment and unsigned

The Analog-to-Digital Converter is utilized to measure DC-Bus voltage, DC-Bus current and the power module temperature. The ADC’s Hi/Lo level detection capability provides automatic detection of the over/under-voltage, over-current and over temperature protection (serviced in associated ISR).

3. Target Motor Theory

3.1 BLDC MotorTargeted by This Application

The Brushless DC motor (BLDC) 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. The stator phase windings are inserted in the slots (distributed winding) as shown on **Figure 3-1** or it can be wound as one coil on the magnetic pole. The magnetization of the permanent magnets and their displacement on the rotor is chosen so that the Back-EMF (the voltage induced into the stator winding due to rotor movement) shape is trapezoidal. This allows the DC voltage (see **Figure 3-2**), with a rectangular shape, to be used to create a rotational field with low torque ripples.

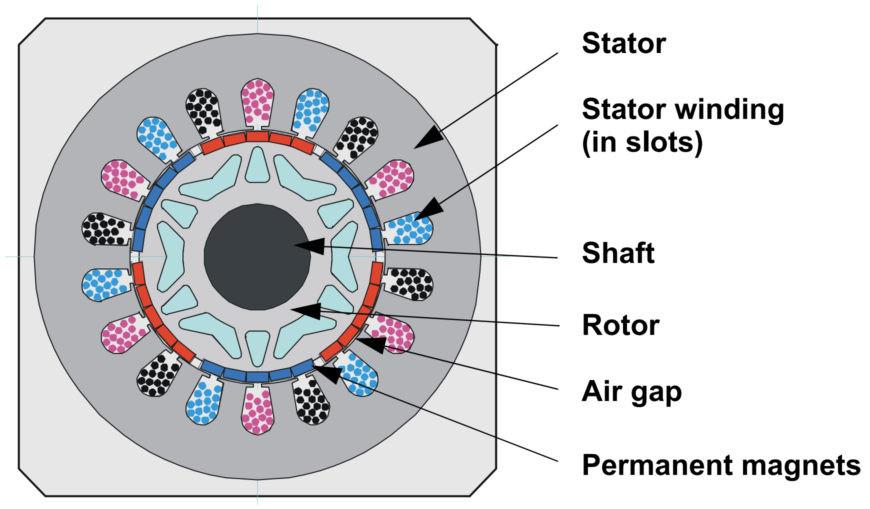


Figure 3-1. BLDC Motor - Cross Section

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

The rectangular, easy to create, shape of applied voltage ensures the simplicity of control and drive. However, the rotor position must be known at certain angles in order to align the applied voltage with the Back-EMF. The alignment between Back-EMF and commutation events is very important. At this condition the motor behaves as a DC motor and runs at the best working point. Thus, simplicity of control and performance makes the BLDC motor the best choice for low-cost and high-efficiency applications.

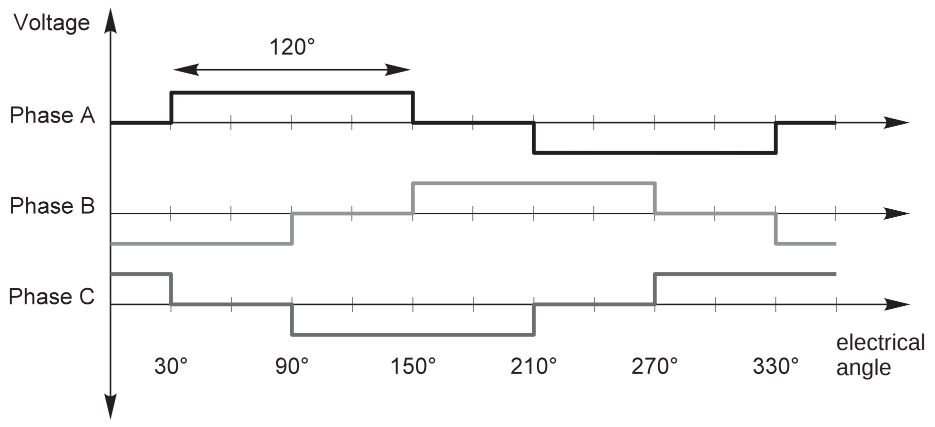


Figure 3-2. Three Phase Voltage System

Figure 3-3 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 the speed reversal the amplitudeis changed and its sign and the phase sequence change too.

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. At the normal operation, the commutation is performed here. Some control techniques lead the commutation by a defined angle in order to control the drive above the PWM voltage control.

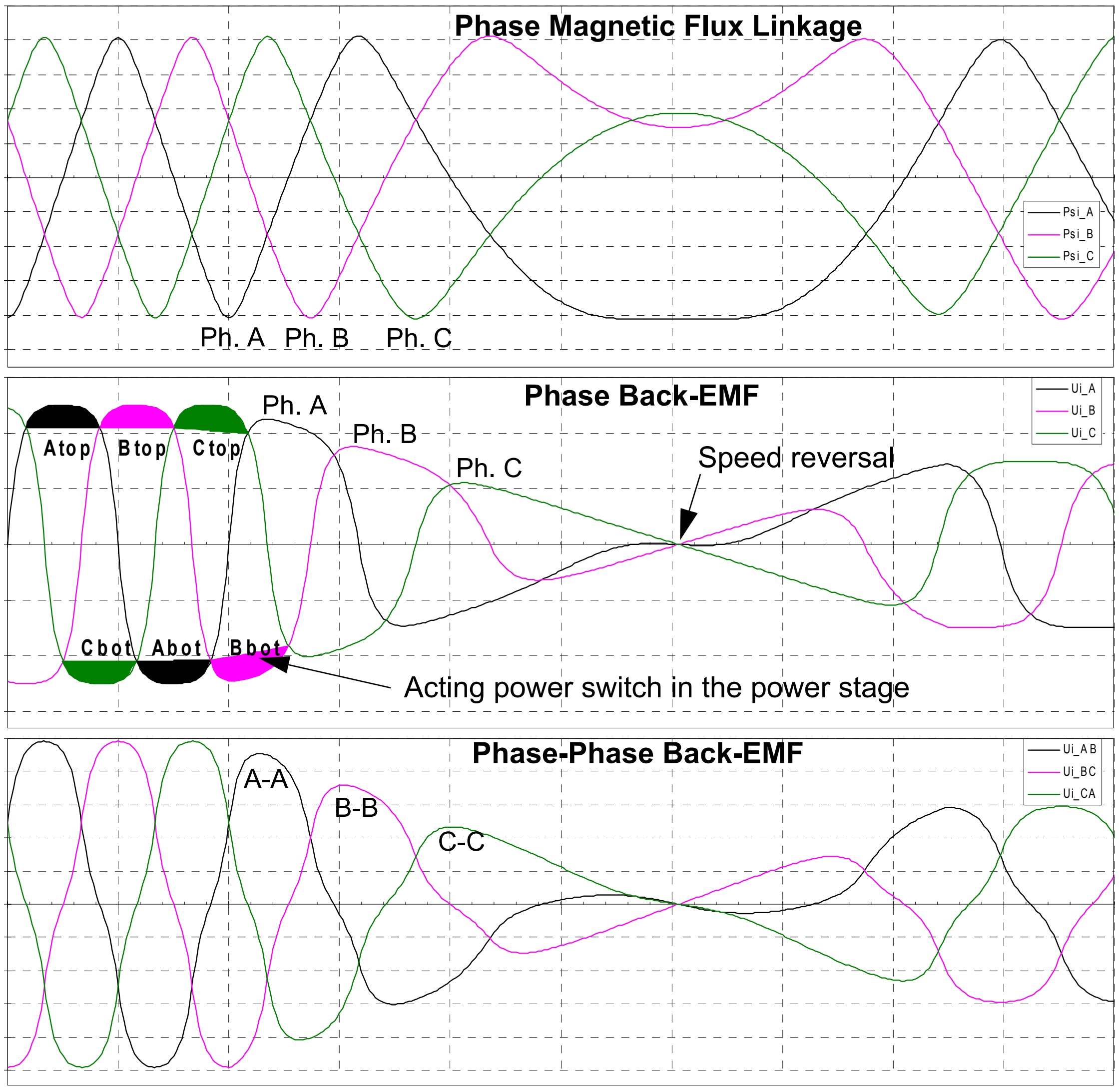


Figure 3-3. BLDC Motor - Back-EMF and Magnetic Flux

3.2 3-Phase BLDC Power Stage

The voltage for a 3-phase BLDC motor is provided by a 3-phase power stage. The 3-phase power stage is controlled by the device on-chip PWM module; which creates the desired switch control signals. A controller with a BLDC motor and power stage is shown in **Figure 5-1**.

3.3 Why Sensorless Control?

As explained in the previous section, the rotor position must be known in order to energize the phase pair and control phase voltage of a BLDC motor. If any sensors are used to detect rotor position, then sensed information must be transferred to a control unit (see **Figure 3-4**).

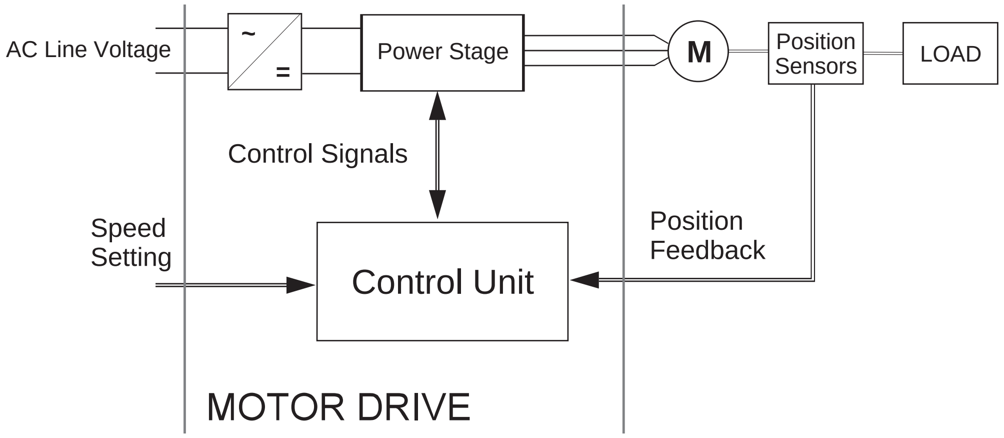


Figure 3-4. Classical System

Therefore, additional connections to the motor are necessary. This may not be acceptable for some kind of applications. There are at least two reasons why you might want to eliminate the position sensors: impossibility to make additional connections between position sensors and the control unit cost of the position sensors and wiring.

The first reason might be solved by integration of the driver within the motor body. However, a significant number of applications requiring a sensorless solution still remain.

For additional BLDC control information, refer to **Section 5.** and AN1627 (**Section 11.2**).

3.4 Power Stage - Motor System Model

In order to explain and simulate the idea of Back-EMF sensing techniques, a simplified mathematical model based on the basic circuit topology (see **Figure 3-5**), is provided.

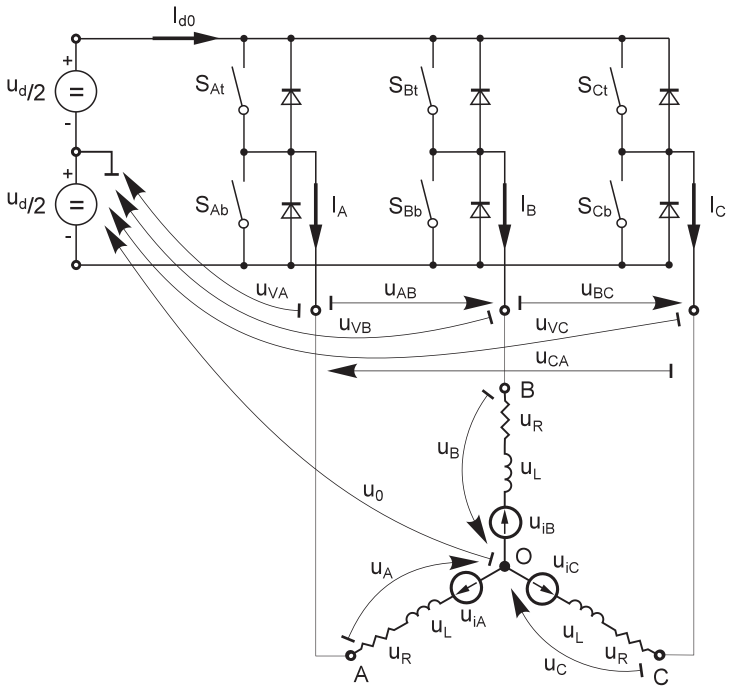


Figure 3-5. Power Stage - Motor Topology

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 plus a Brushless DC 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 3-2**) 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.

3.4.1 Mathematical Model

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

$$\begin{aligned}
u_A &= \frac{1}{3}\left(2u_{VA} - u_{VB} - u_{VC} + \sum_{x=A}^{C} u_{ix}\right) \\
u_B &= \frac{1}{3}\left(2u_{VB} - u_{VC} - u_{VA} + \sum_{x=A}^{C} u_{ix}\right) \\
u_C &= \frac{1}{3}\left(2u_{VC} - u_{VA} - u_{VB} + \sum_{x=A}^{C} u_{ix}\right) \\
u_O &= \frac{1}{3}\left(\sum_{x=A}^{C} u_{Vx} - \sum_{x=A}^{C} u_{ix}\right) \\
0 &= i_A + i_B + i_C
\end{aligned} \qquad \text{(EQ 3-1.)}$$

where:

$u_{VA} \ldots u_{VC}$	are “branch” voltages between one power stage output and its natural zero.
$u_A \ldots u_C$	are motor phase winding voltages.
$u_{iA} \ldots u_{iC}$	are phase Back-EMF induced in the stator winding.
u_O	is the differential voltage between the central point of the star connection of motor winding and the power stage natural zero
$i_A \ldots i_C$	are phase currents

The equations (EQ 3-1.) 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.

$$\begin{aligned}
u_{VA} - u_{iA} - \frac{1}{3}\left(\sum_{x=A}^{C} u_{Vx} - \sum_{x=A}^{C} u_{ix}\right) &= R \cdot i_A + L\frac{\mathrm{d}i_A}{\mathrm{d}t} \\
u_{VB} - u_{iB} - \frac{1}{3}\left(\sum_{x=A}^{C} u_{Vx} - \sum_{x=A}^{C} u_{ix}\right) &= R \cdot i_B + L\frac{\mathrm{d}i_B}{\mathrm{d}t} \\
u_{VC} - u_{iC} - \frac{1}{3}\left(\sum_{x=A}^{C} u_{Vx} - \sum_{x=A}^{C} u_{ix}\right) &= R \cdot i_C + L\frac{\mathrm{d}i_C}{\mathrm{d}t}
\end{aligned} \qquad \text{(EQ 3-2.)}$$

where:

R,L - motor phase resistance, inductance

The internal torque of the motor itself is defined as:

$$T_i = \frac{1}{\omega}\sum_{x=A}^{C} u_{ix} \cdot i_x = \sum_{x=A}^{C} \frac{d\Psi_x}{d\theta} \cdot i_x \qquad \text{(EQ 3-3.)}$$

where:

T_i	-	internal motor torque (no mechanical losses)
ω,θ	-	rotor speed, rotor position
x	-	phase index, it stands for A,B,C
Ψ_x	-	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.

3.5 Back-EMF Sensing

The Back-EMF sensing technique is based on the fact that only two phases of a Brushless DC motor are energized at a time (see **Figure 3-2**). The third phase is a non-fed phase that can be used to sense the Back-EMF voltage.

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

$$\begin{aligned} & S_{Ab}, S_{Bt} performing PWM switching \\ & u_{VA} = \mp\frac{1}{2}u_d, u_{VB} = \pm\frac{1}{2}u_d \\ & i_A = -i_B, i_C = 0, di_C = 0 \\ & u_{iA} + u_{iB} + u_{iC} = 0 \end{aligned} \qquad \text{(EQ 3-4.)}$$

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

$$u_{VC} = \frac{3}{2}u_{iC} \qquad \text{(EQ 3-5.)}$$

Figure 3-5 illustrates 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 the Zero Crossing can be recognized.

The general expression can be found by:

$$u_{Vx} = \frac{3}{2}u_{ix} \qquad \text{(EQ 3-6.)}$$

where:

$$x = A, B, C \qquad \text{(EQ 3-7.)}$$

There are two necessary conditions which must be met:

- Top and bottom switch (in diagonal) have to be driven with the same PWM signal
- No current is going through the non-fed phase used to sense the Back-EMF

Figure 3-6 shows branch and motor phase winding voltages during a 0-360°electrical interval. Shaded rectangles designate the validity of the equation (EQ 3-6.). In other words, the Back-EMF voltage can be sensed during designated intervals.

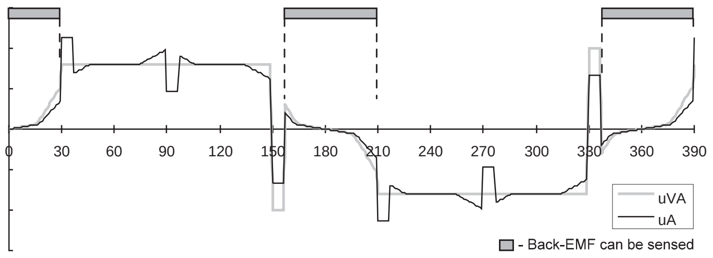


Figure 3-6. Phase Voltage Waveforms

4. System Design Concept

4.1 System Specification

- Control technique incorporates
  - using AD converter for sensorless Back-EMF Zero Crossing commutation control
  - motoring mode
  - single speed feedback loop
  - both direction of the rotation
- Targeted for 56F80xEVM platforms
- Running on one of three optional board and motor hardware sets
  - Low Voltage Evaluation Motor hardware set
  - Low Voltage hardware set
  - High Voltage hardware set at variable line voltage 115 - 230V AC
- Overvoltage, Undervoltage, Overcurrent, and Temperature Fault protection
- Manual Interface (Start/Stop switch, Up/Down push button control, LED indication)
- PCMaster Interface
- Power Stage Identification with control parameters set according to used hardware set

The introduced BLDC motor control drive with Back-EMF Zero Crossing using AD converter is designed as a system that meets the following general performance requirements:

Table 4-1. Low Voltage Evaluation Hardware Set Specifications

Motor Characteristics:	Motor Type	4 poles, three phase, star connected, BLDC motor
	Speed Range:	< 5000 rpm (at 60V)
	Maximal line voltage:	60V
	Phase Current	2A
	Output Torque	0.140Nm (at 2A)
Drive Characteristics:	Speed Range	< 2000 rpm
	Input Voltage:	12V DC
	Max DC-Bus Voltage	15.8 V
	Control Algorithm	Speed Closed Loop Control
Load Characteristic:	Type	Varying

Table 4-2. Low Voltage Hardware Set Specifications

Motor Characteristics:	Motor Type	6 poles, three phase, star connected, BLDC motor
	Speed Range:	3000 rpm (at 12V)
	Max. Electrical Power:	150 W
	Phase Voltage:	3*6.5V
	Phase Current	17A
Drive Characteristics:	Speed Range	< 3000 rpm
	Input Voltage:	12V DC
	Max DC-Bus Voltage	15.8 V
	Control Algorithm	Speed Closed Loop Control
Load Characteristic:	Type	Varying

Table 4-3. High Voltage Evaluation Hardware Set Specifications

Motor Characteristics:	Motor Type	6 poles, three phase, star connected, BLDC motor
	Speed Range:	2500 rpm (at 310V)
	Max. Electrical Power:	150 W
	Phase Voltage:	3*220V
	Phase Current	0.55A
Drive Characteristics:	Speed Range	< 2500 rpm
	Input Voltage:	310V DC
	Max DC-Bus Voltage	380 V
	Control Algorithm	Speed Closed Loop Control
	Optoisolation	Required
Load Characteristic:	Type	Varying

4.2 Sensorless Drive Concept

The chosen system concept is shown below. The sensorless rotor position detector detects the Zero Crossing points of Back-EMF induced in non-fed motor windings. The obtained information is processed in order to commutate energized phase pair and control the phase voltage, using Pulse-Width-Modulation.

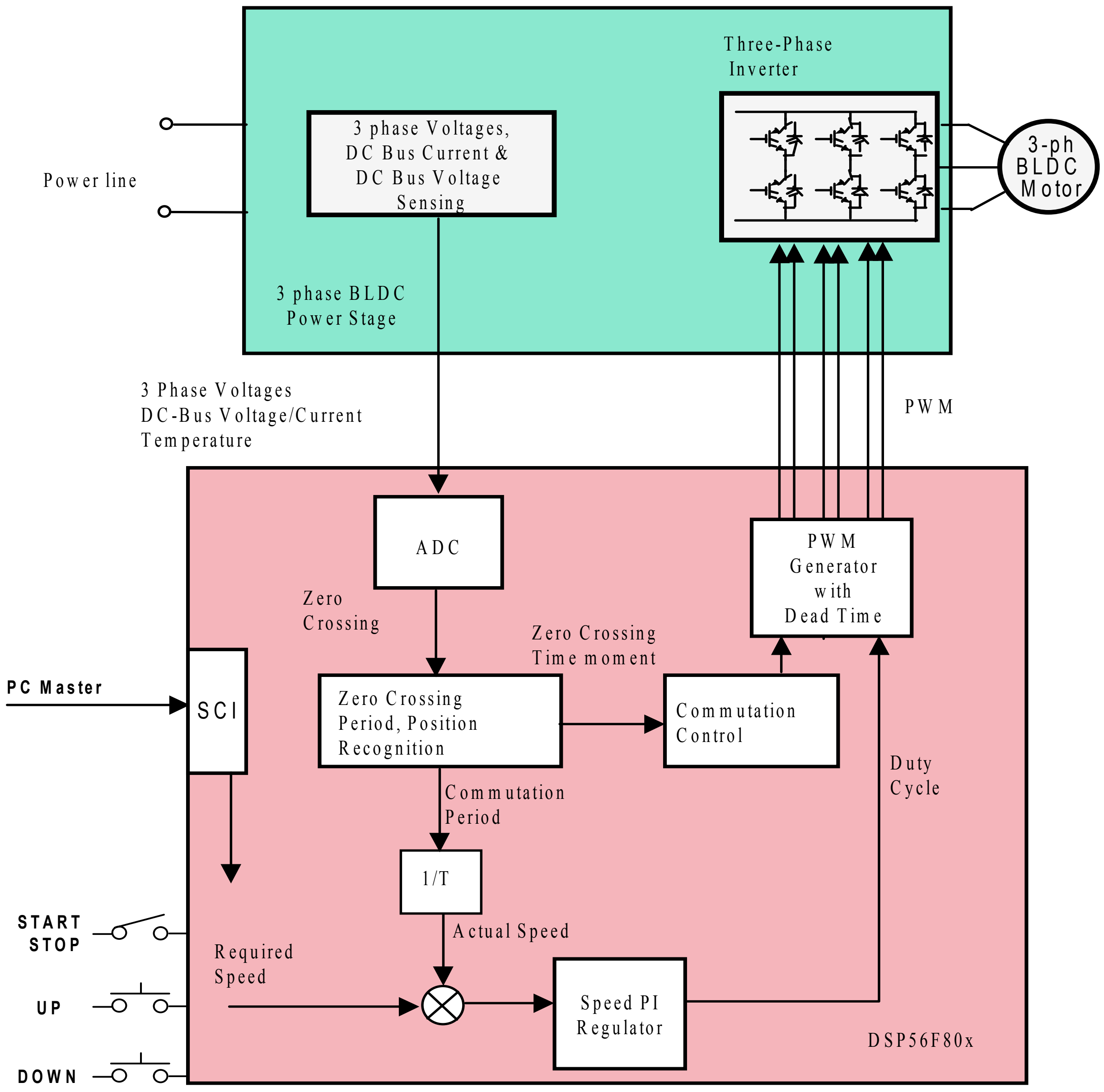


Figure 4-1. System Concept

The resistor network is used to divide sensed voltages down to a 0-3.3V voltage level. Zero Crossing detection is synchronized with the center of center aligned PWM signal by the software in order to filter high voltage spikes produced by the switching of the IGBTs (MOSFETs).

The divided phase voltages are connected to the AD converter module on the controller and are processed in order to get the Back-EMF Zero Crossing signal. The Back-EMF Zero Crossing detection enables position recognition, as explained in previous sections. The software selects one of the phases which corresponds to the present commutation step.

A current shunt is used to measure the DC-Bus current. The obtained signal is rectified and amplified (0-3.3V with 1.65V offset). The controller's A/D converter as well as Zero Crossing detection is synchronized with the PWM signal. This synchronization avoids spikes when the IGBTs (or MOSFETs) are switching and simplifies the electric circuit.

The A/D converter is also used to sense the DC-Bus Voltage and drive Temperature. The DC-Bus voltage is divided down to a 3.3V signal level by a resistor network.

The six IGBTs (copack with built-in fly back diode) or MOSFETs and gate drivers create a compact power stage. The drivers provide the level shifting that is required to drive high side bridge circuits commonly used in motor drives. The PWM technique is applied to the control motor phase voltage.

5. Control Technique

5.1 Control Technique - General Overview

The general overview of used control technique is shown in **Figure 4-1**. It will be described in following subsections:

- PWM voltage generation for BLDC
- Back-EMF Zero Crossing sensing
- Sensorless Commutation Control
- Speed Control

The implementation of the control technique with all the software processes are shown in Flow Chart, State diagrams and Data Flow. Refer to **Section 7.1**, **Section 7.2**, and **Section 7.3**.

5.2 PWM Voltage Generation for BLDC

A 3-phase voltage system (see **Figure 3-2**.) needs to be created to run the BLDC motor. It is provided by 3-phase power stage with 6 IGBTs (MOSFET) power switches controlled by the device on-chip PWM module (see **Figure 5-1**).

The PWM signals, with their current state, are shown in **Figure 5-2** and **Figure 5-3**.

Figure 5-2 shows that both the Bottom and Top power switches of the non-fed phase must be switched off.

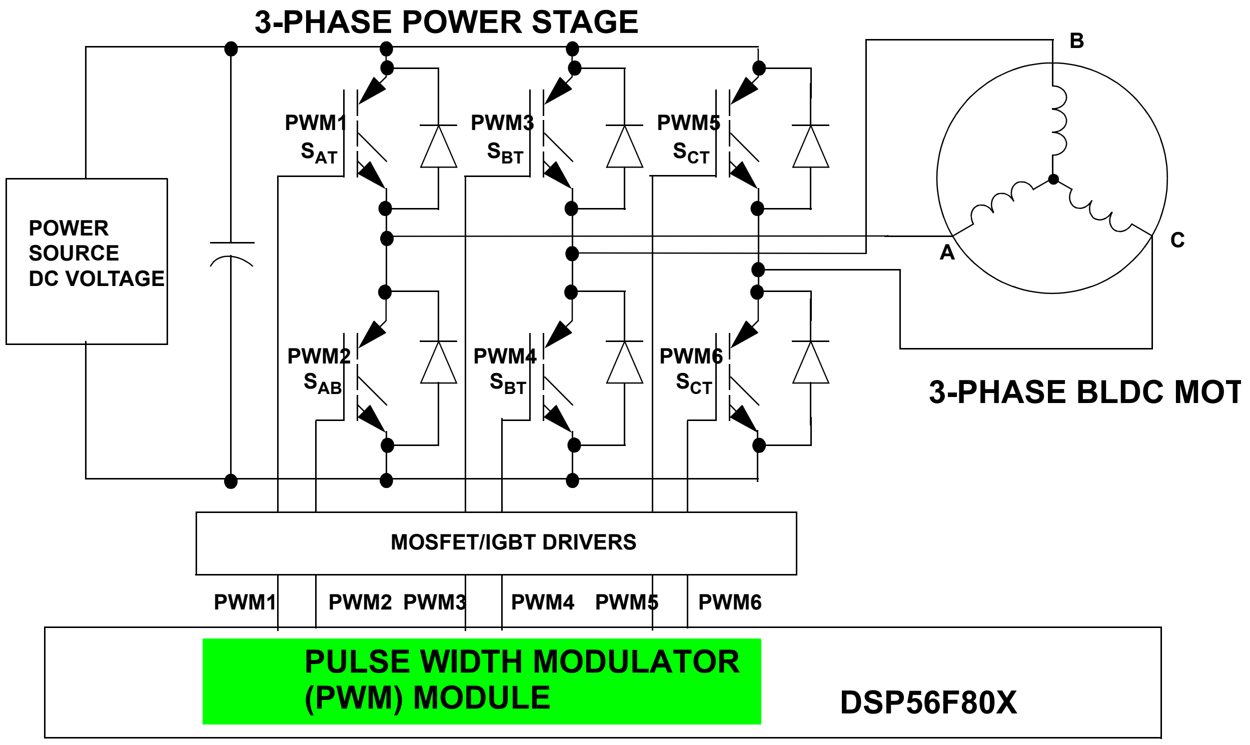


Figure 5-1. PWM with BLDC Power Stage

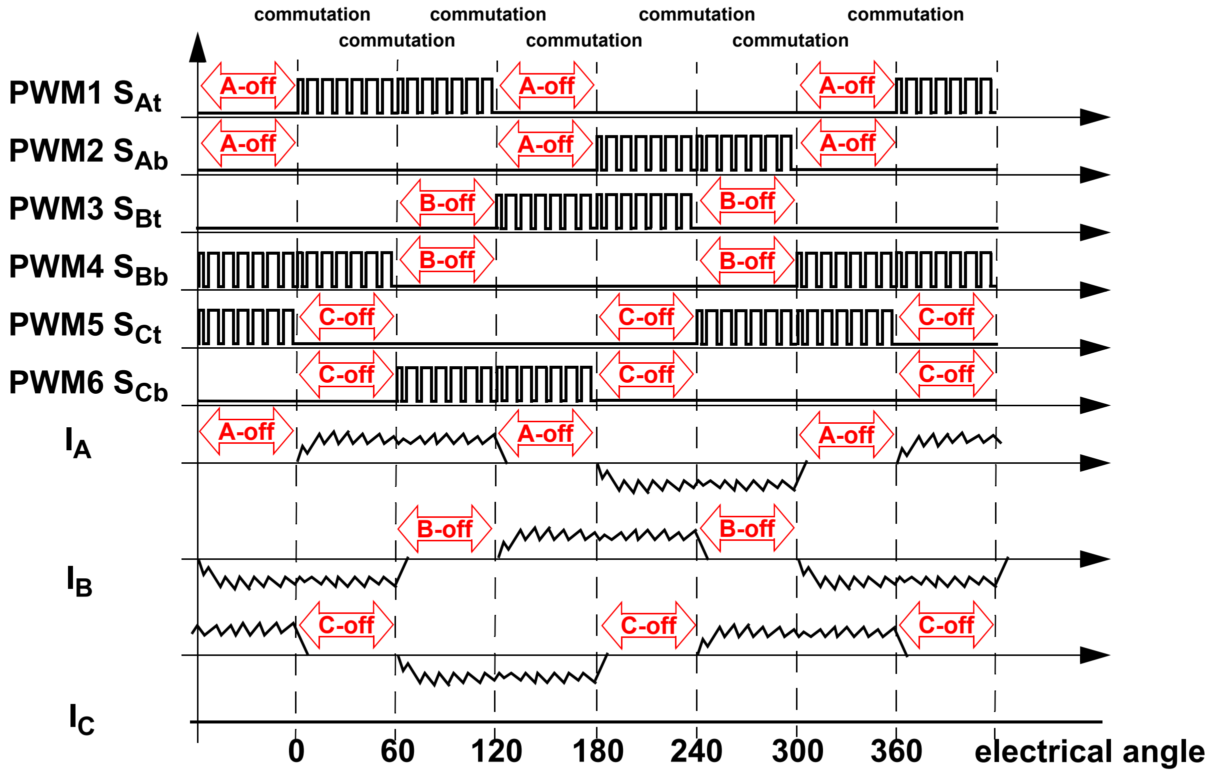


Figure 5-2. 3-phase BLDC Motor Commutation PWM Signal

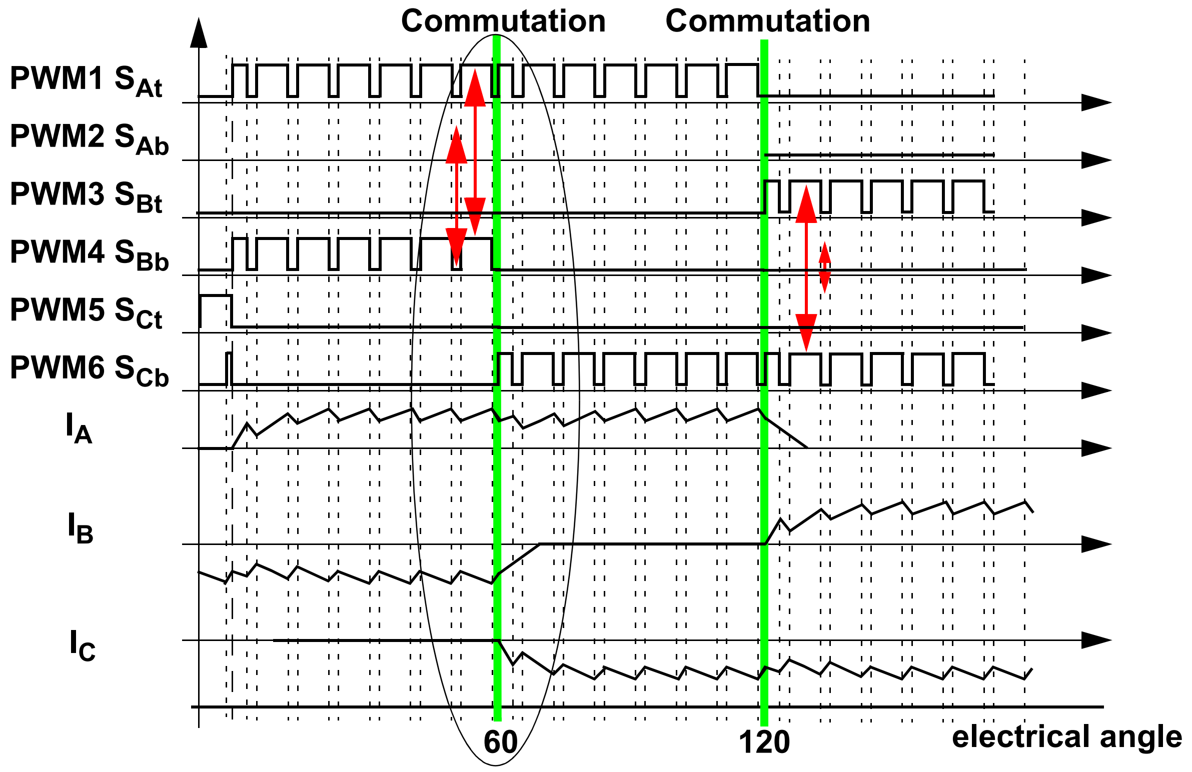


Figure 5-3. BLDC Commutation with Bipolar (Hard) Switching

Figure 5-3 shows that the diagonal power switches are driven by the same PWM signal as shown with arrow lines. This technique is called bipolar (hard) switching. The voltage across the two energized coils is always ±DC-Bus voltage whenever there is a current flowing through these coils.

5.3 Back-EMF Zero Crossing sensing

5.3.1 ADC Converter Used for Back-EMF Zero Crossing

The Back-EMF Zero Crossing is detected by sensing the motor non-fed phase “branch“ voltage (u_{vi} in Section 3.5) and DC-bus voltage u_d utilizing the ADC. (Refer to Section 3.).

The 56F80x family offers an excellent on-chip Analog-to-Digital converter. Its unique feature set provides an automatic detection of the signal crossing the value contained in the ADC offset register.

Then the Back-EMF Zero Crossing 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

5.3.1.1 ADC Zero Crossing Checking

The Zero Crossing for position estimation is sensed using the AD converter.

As stated, the AD converter has individual ADC Offset Registers for each ADC channels. The value in the Offset Register can be subtracted from the AD conversion output. The final result of the AD 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.

This application utilizes ADC Zero Crossing Interrupt to get the Back-EMF Zero Crossing event.

5.3.1.2 ADC Zero Crossing Offset Setting

As explained in the previous section, the ADC Offset Register is set to one half of the DC-Bus value. This is valid at the following conditions:

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

The ADC Offset Register needs to be continuously updated, to reflect the DC-Bus voltage variation caused by the ripple of DC-Bus voltage.

The above mentioned conditions are not 100% fulfilled in real drive due to the unbalance in real sensing circuitry and the motor phases. Therefore, the real application must compensate such unbalance.

The presented application first sets the ADC Offset Registers[0..3] of all 3-phases to:

- ADC Offset Register[0..3] = Calibration Phase Voltage Coefficient * DC-Bus

Where the Calibration Phase Voltage Coefficient is set to 0.5. Later during the Alignment state the Calibration Phase Voltage Coefficient is further corrected. The DC-Bus and non-fed phase branch voltage are measured and the correction is calculated according to the following formula:

Calibration Phase Voltage Coefficient = (ADC Offset Register + Free Phase Branch Voltage)/DC-Bus voltage

5.3.2 Back-EMF Zero Crossing Synchronization with PWM

The power stage PWM switching causes high voltage spike on the phase voltages. This voltage spike is passed to the non-fed phase due to mutual capacitor coupling between the motor windings (see **Figure 5-4**). Non-fed phase "branch" voltage U_{va} is then disturbed by PWM switching shown on phase branch voltage U_{vb}.

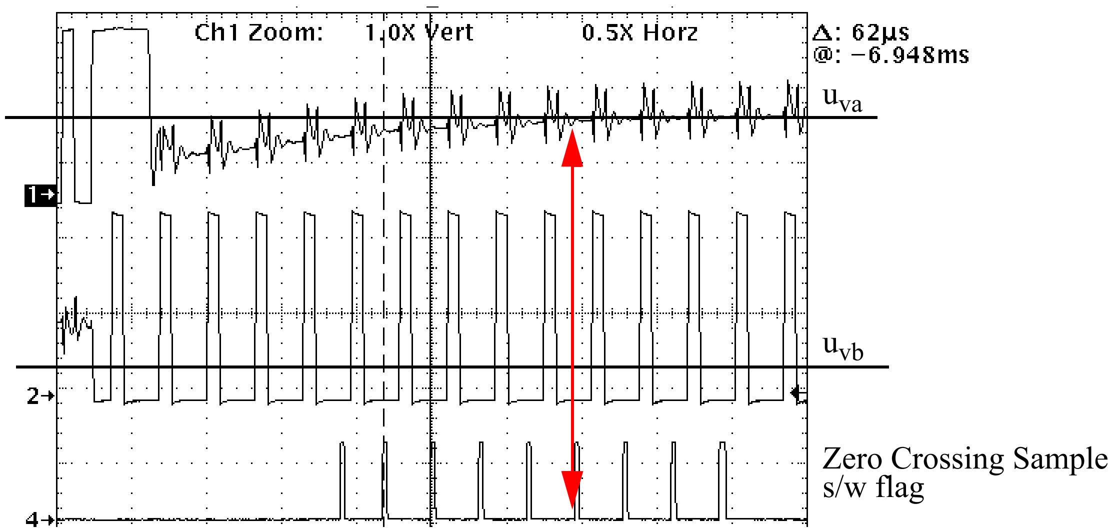


Figure 5-4. Back-EMF Zero Crossing Synchronization with PWM

The non-fed phase “branch” voltage U_{va} is disturbed at the PWM switching edges. Therefore the presented BLDC Motor Control application synchronizes the Back-EMF Zero Crossing detection with PWM. The AD conversion of phase branch voltages is triggered in the middle of PWM pulse. Then the voltage for Back-EMF is sensed at the time moments because the non-fed phase branch voltage is already stabilized.

5.4 Sensorless Commutation Control

This section presents sensorless BLDC motor commutation with the Back-EMF Zero Crossing technique.

In order to start and run the BLDC motor, the control algorithm has to go through the following states:

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

Figure 5-5 shows the transitions between the states. First, the rotor is aligned to a known position without the position feedback. When the rotor moves, the Back-EMF is induced on the non-fed phase and 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.

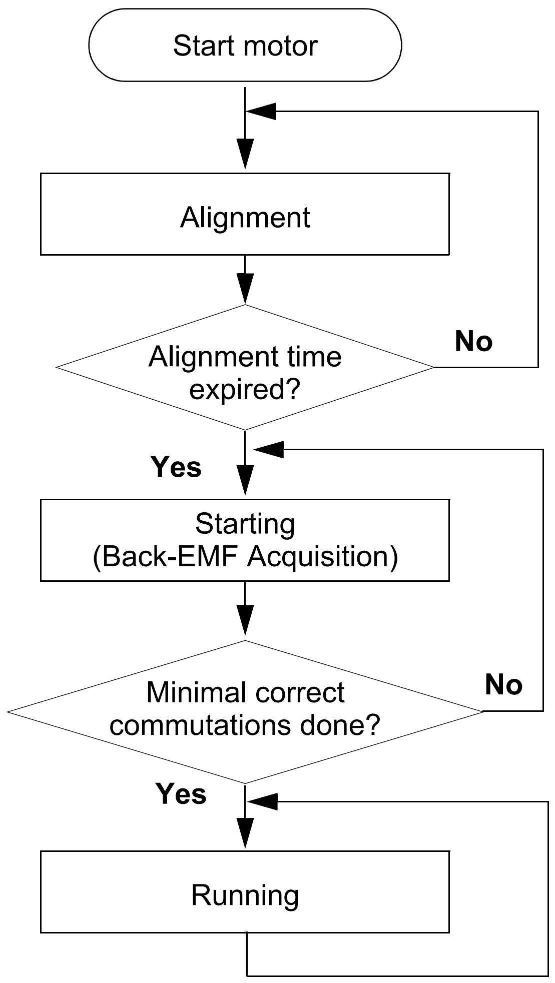

Start motor
Alignment
Alignment time expired?
No
Yes
Starting (Back-EMF Acquisition)
Minimal correct commutations done?
No
Yes
Running

Figure 5-5. Commutation Control States

5.4.1 Alignment

Before the motor starts, there is a short time (which depends on the motor’s electrical time constant) when the rotor position is aligned to a known position by applying PWM signals to only two motor phases (no commutation). The Current Controller keeps the current within predefined limits. This state is necessary in order to create a high start-up torque and recognized the rotor position. When the preset time-out expires, this state is finished.

- The Current Controller subroutine (with PI regulator) is called to control the DC-Bus current. The subroutine sets the right PWM ratio for the required current.

The current is sampled and the Current Controller is calculated in every PWM cycle.

The BLDC motor rotor position (with flux vectors during alignment) is shown in Figure 5-6.

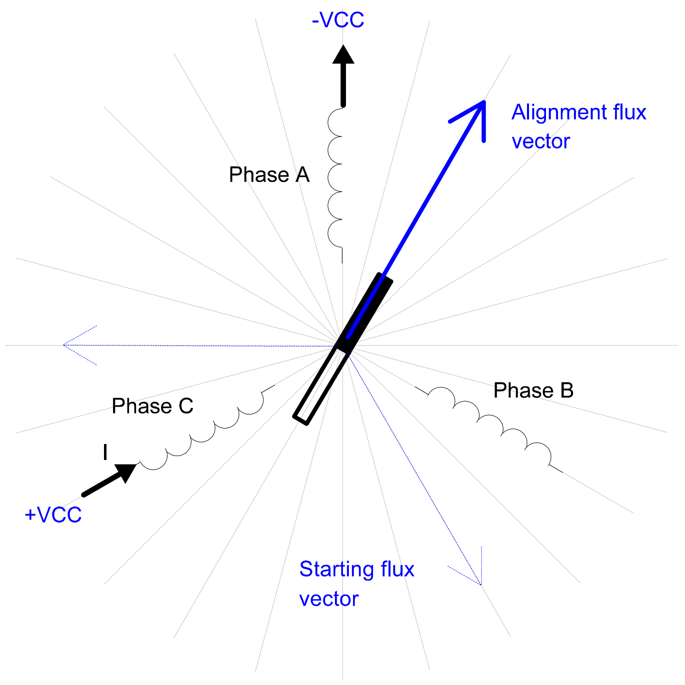


Figure 5-6. Alignment

5.4.2 Running

The commutation process is the series of states which is assured when the 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 need to be provided:

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

5.4.2.1 Algorithms BLDC Motor Commutation with Zero Crossing Sensing

All these processes are provided by new algorithms which were designed for these type of applications within SDK. They are described in **Section 11.1**.

Diagrams aid in explaining how the commutation works. After commuting the motor phases, a time interval (Per_Toff[n]) is set that allows the shape of the Back-EMF to be stabilized. Stabilization is required because the electro-magnetic interference and fly-back 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 performed at this new time.

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

The flow chart explaining the principle of BLDC Commutation control with Back-EMF Zero Crossing Sensing is shown in **Figure 5-7**.

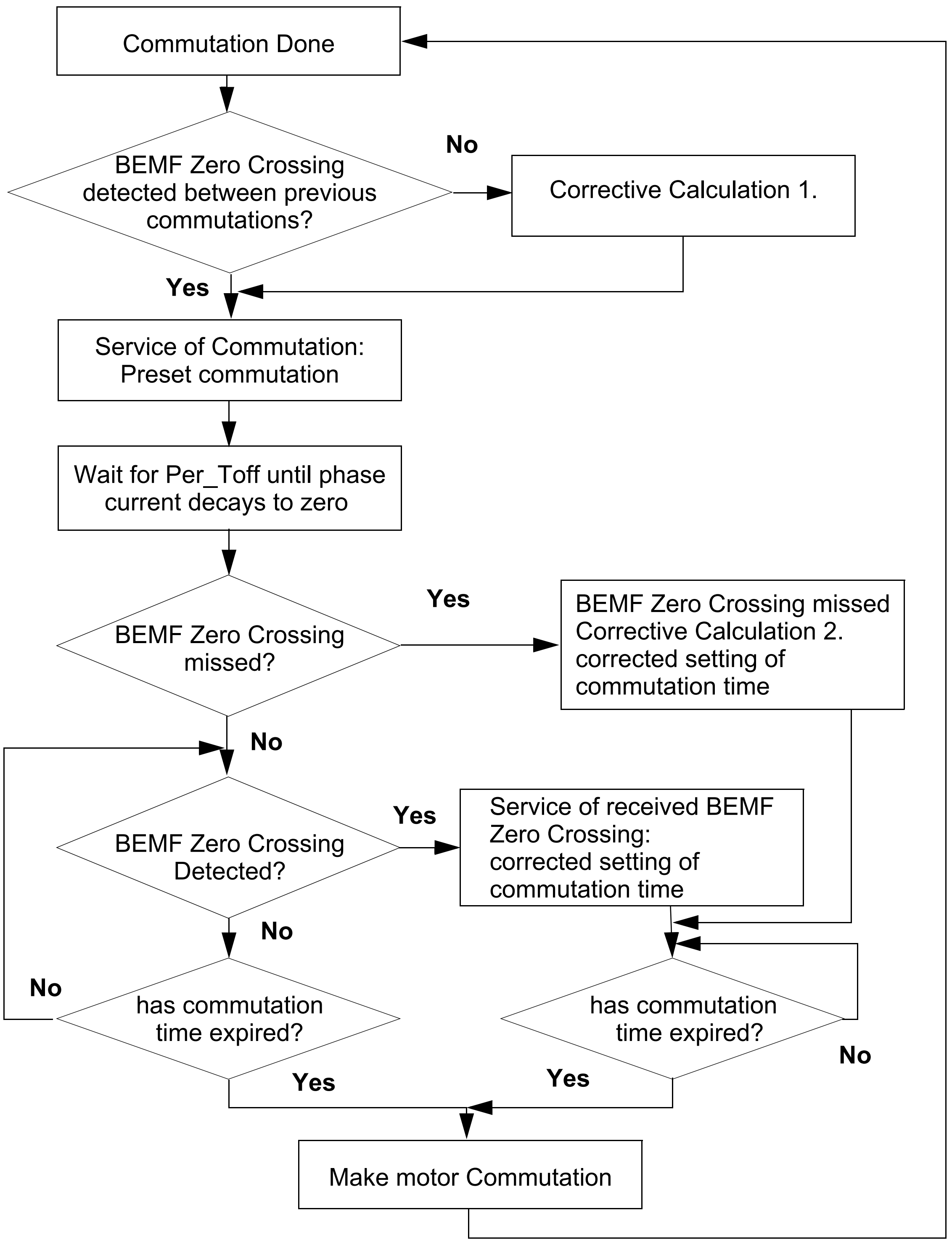

Commutation Done
BEMF Zero Crossing detected between previous commutations?
No
Corrective Calculation 1.
Yes
Service of Commutation: Preset commutation
Wait for Per_Toff until phase current decays to zero
BEMF Zero Crossing missed?
Yes
BEMF Zero Crossing missed Corrective Calculation 2. corrected setting of commutation time
No
BEMF Zero Crossing Detected?
Yes
Service of received BEMF Zero Crossing: corrected setting of commutation time
No
No
has commutation time expired?
has commutation time expired?
No
Yes
Yes
Make motor Commutation

Figure 5-7. Flow Chart - BLDC Commutation with Back-EMF Zero Crossing Sensing

5.4.2.2 Running - Commutation Times Calculation

Commutation times calculation is provided by algorithm **bldcZCComput** described in **Section 11.1**.

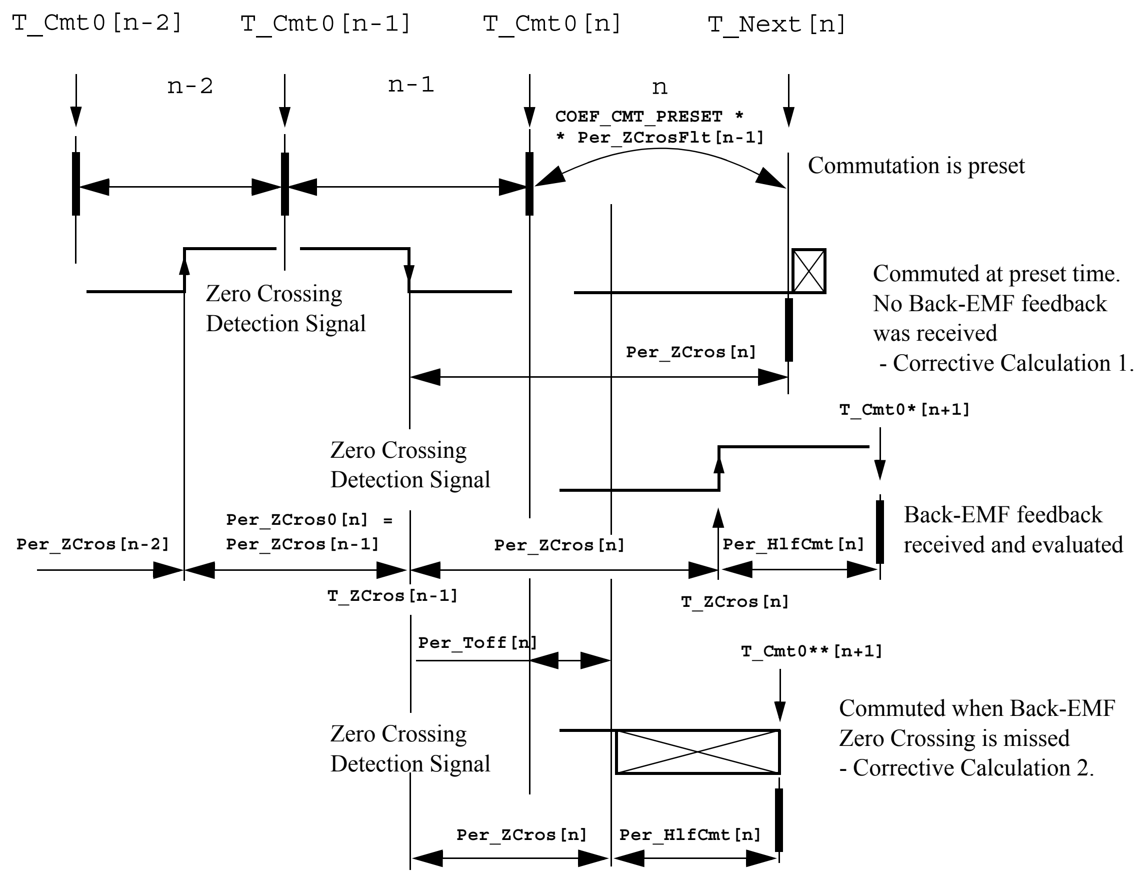


Figure 5-8. BLDC Commutation Times with Zero Crossing Sensing

The following calculations are made to calculate the commutation times (T_Next[n]) during the **Running state**:

- **Service of Commutation** - The commutation time (T_Next[n]) is predicted:

```
T_Next[n] = T_Cmt0[n] + Per_CmtPreset[n] =
          = T_Cmt0[n] + Coef_CmtPrecomp*Per_ZCrosFlt[n-1]
          coefficient Coef_CmtPrecomp = 2 at Running state!
If Coef_CmtPrecomp*Per_ZCrosFlt>Max_PerCmt
          then result is limited at Max_PerCmt
```

- **Service of received Back-EMF Zero Crossing** - The commutation time (T_Next*[n]) is evaluated from the captured Back-EMF Zero Crossing time (T_ZCros[n]):

```
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]- C_CMT_ADVANCE*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] = Per_ZCrosFlt*Coef_Toff and Max_PerCmtProc minimum
          Coef_Toff = 0.35 at Running state, Max_PerCmtProc = 100!
Per_ZCros0 <-- Per_ZCros[n]
T_ZCros0 <-- T_ZCros[n]
T_Next*[n] = T_ZCros[n] + HlfCmt[n]
```

- If no Back-EMF Zero Crossing was captured during preset commutation period (`Per_CmtPreset`[n]) then **Corrective Calculation 1.** is made:

```
T_ZCros[n] <-- CmtT[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 = 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] = Per_ZCrosFlt*Coef_Toff and Max_PerCmtProc minimum
Per_ZCros0 <-- Per_ZCros[n]
T_ZCros0 <-- T_ZCros[n]
```

- If Back-EMF Zero Crossing is missed then **Corrective Calculation 2.** is made:

```
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 = 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_Cnt0 = time of the last commutation
T_Next = Time of the Next Time event (for Timer Setting)
T_zCros = Time of the last Zero Crossing
T_zCros0 = Time of the previous Zero Crossing
Per_Toff = Period of the Zero Crossing off
Per_CmtPreset = Preset Commutation Periof from commutation to next commutation if no
          Zero Crossing was captured
Per_ZCros = Period between Zero Crossings (estimates required commutation period)
Per_ZCros0 = Pervious period between Zero Crossings
Per_ZCrosFlt = Estimated period of commutation filtered
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.**

5.4.3 Starting (Back-EMF Acquisition)

The Back-EMF sensing technique enables a sensorless detection of the rotor position, however the drive must be first started without this feedback. It is caused by the fact that the amplitude of the induced voltage is proportional to the motor speed. Hence, the Back-EMF cannot be sensed at a very low speed and a special start-up algorithm must be performed.

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

It implies (for BLDC motors) the following:

1. The level of phase current must be high enough.
2. The angle between the stator and rotor magnetic fields must be in 90deg±30deg.

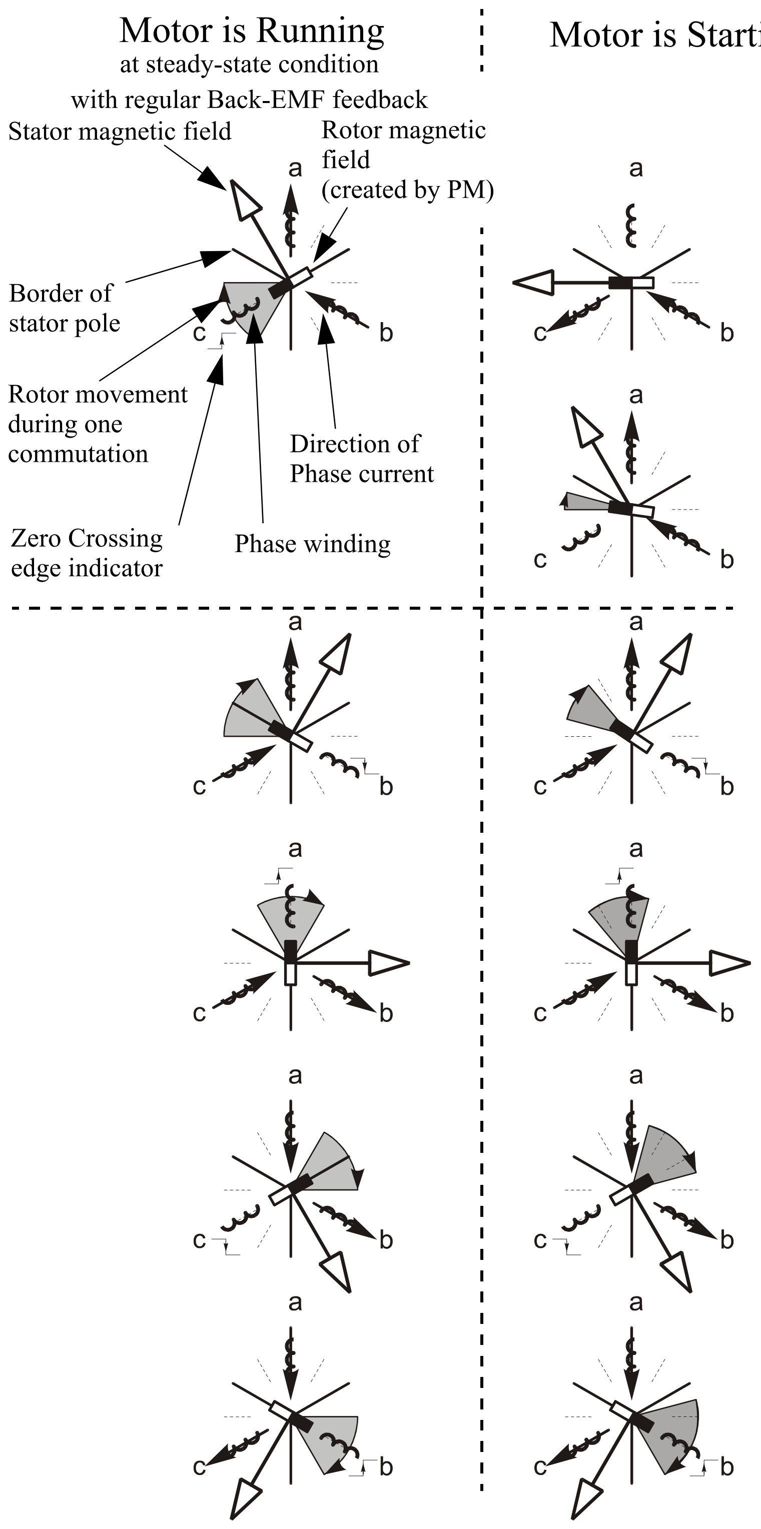
The first condition is satisfied during the Alignment state by keeping the DC-Bus current on the level which is sufficient to start the motor. In the Starting (Back-EMF Acquisition) state the same value of PWM duty cycle is used as the one which has stabilized the DC-Bus current during the Align state.

The second condition is more difficult to fulfill without any position feedback information. After the Alignment state, both the stator and the rotor magnetic fields are aligned (0deg angle). Therefore, the two fast (faster then the rotor can follow) commutation must be applied to create an angular difference of the magnetic fields (see **Figure 5-9**).

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

This allows to start the motor the way that minimal speed (defined by state when Back-EMF can be sensed) is achieved during several commutation while producing the required torque. Until the Back-EMF feedback is locked into the Commutation Process (explained in **Section 5.4.2**) assures that commutations are done in advance, so that successive Back-EMF Zero Crossing events are not missed.

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.


Motor is Running
at steady-state condition
with regular Back-EMF feedback
Motor is Starting
Stator magnetic field
Rotor magnetic field (created by PM)
Border of stator pole
Rotor movement during one commutation
Direction of Phase current
Zero Crossing edge indicator
Phase winding
a
b
c

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 in order to assure 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 5-9. Vectors of Magnetic Fields

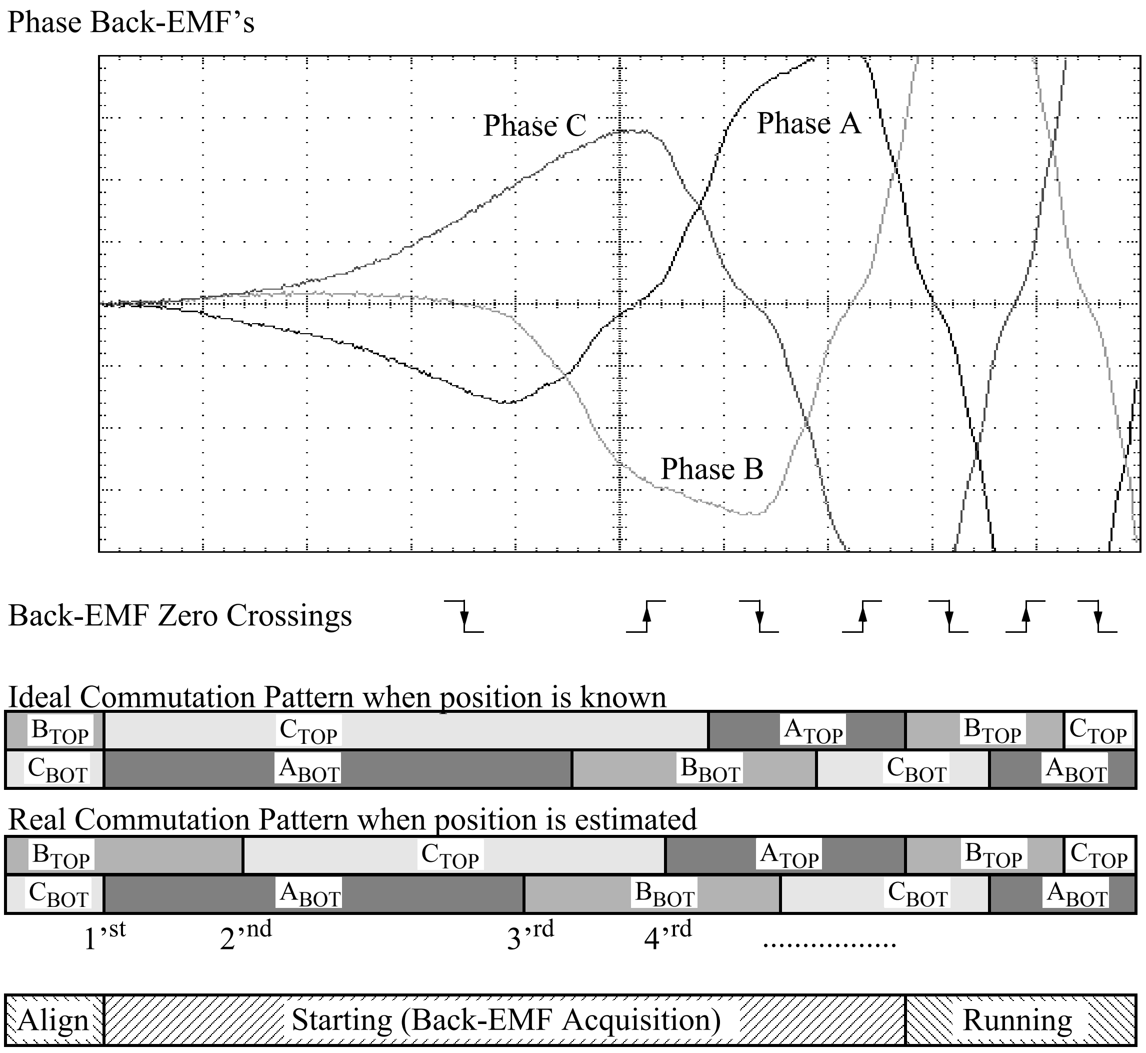


Figure 5-10. Back-EMF at Start-Up

Figure 5-10 demonstrates the Back-EMF during the start-up. The amplitude of the Back-EMF varies according to the rotor speed. During the Starting (Back-EMF Acquisition) state the commutation is done in advance. In the Running state the commutation is done at the right moments.

Figure 5-11 illustrates the sequence of the commutations during the Starting (Back-EMF Acquisition) state.

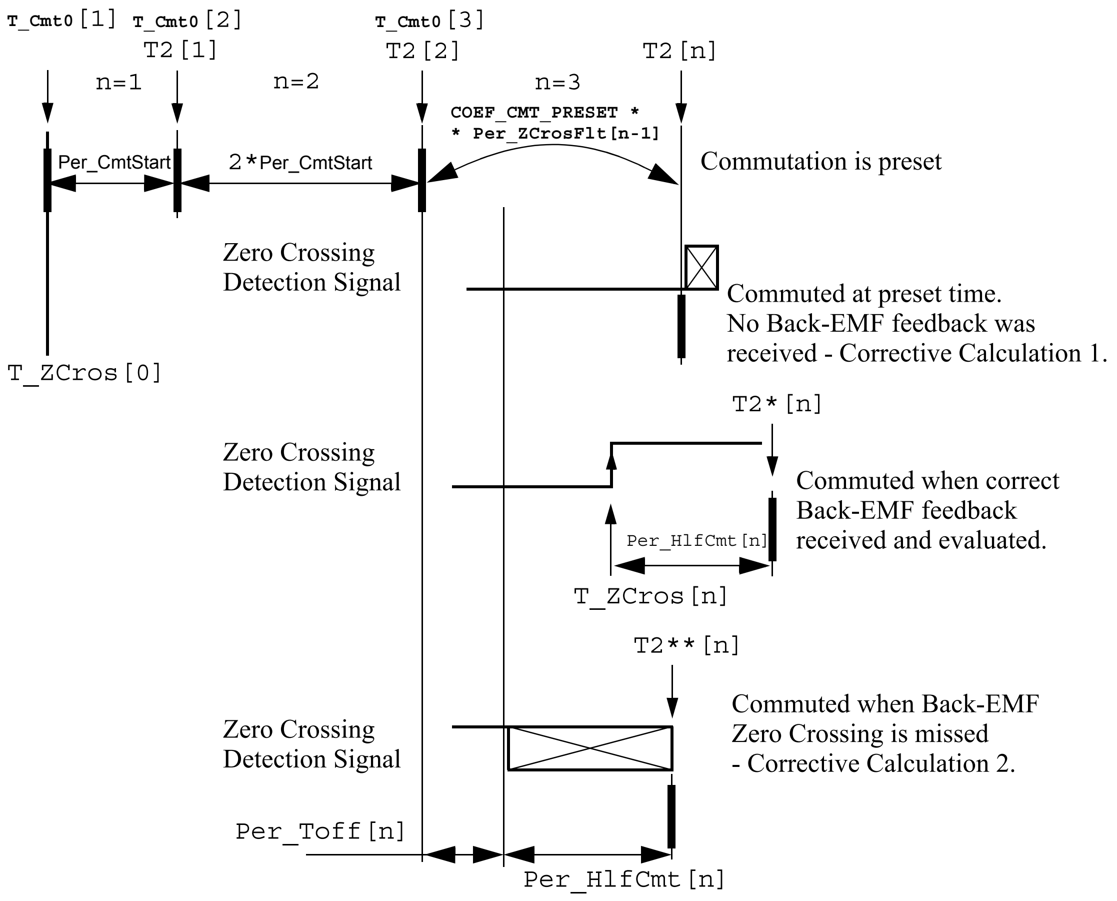

T_Cmt0[1]
T_Cmt0[2]
T2[1]
T_Cmt0[3]
T2[2]
T2[n]
n=1
n=2
n=3
COEF_CMT_PRESET *
* Per_ZCrosFlt[n-1]
Per_CmtStart
2*Per_CmtStart
Commutation is preset
Zero Crossing Detection Signal
Commuted at preset time. No Back-EMF feedback was received - Corrective Calculation 1.
T_ZCros[0]
T2*[n]
Zero Crossing Detection Signal
Commuted when correct Back-EMF feedback received and evaluated.
Per_HlfCmt[n]
T_ZCros[n]
T2**[n]
Zero Crossing Detection Signal
Commuted when Back-EMF Zero Crossing is missed - Corrective Calculation 2.
Per_Toff[n]
Per_HlfCmt[n]

Figure 5-11. Calculation of the Commutation Times during the Starting (Back-EMF Acquisition) State

As can be seen in **Figure 5-11** the commutation times T2[1] and T2[2] are calculated without any influence of Back-EMF feedback

5.4.3.1 Starting - Commutation Times Calculation

The calculations made during Starting (Back-EMF Acquisition) state can be seen in **Section 11.1**.

Even the commutation process sub-states of the Starting (Back-EMF Acquisition) state remain the same as during the Running state. The required commutation timing depends on the MCS state (Starting state, Running state). It is provided by different settings of the commutation constants **Coef_CmtPrecompFrac, Coef_CmtPrecompLShft, Coef_HlfCmt, Coef_Toff,** in structure **StartComputInit** (differs from RunComputInit). The commutation times calculation is the same as described in Section 5.4.2.2; however, the following computation coefficients are different:

```
coefficient Coef_CmtPrecomp = 2 at Starting state!
coefficient Coef_HlfCmt = 0.125 with advanced angle Advance_angle: 60Deg*3/8 = 22.5Deg
        at Starting state!

Coef_Toff = 0.5 at Running state, Max_PerCmtProc = 100!
```

5.5 Speed Control

The speed-closed loop control is provided by a PI regulator as described in Section 7.2.5. The actual speed (Omega_Actual) is computed from the average of two Back-EMF Zero Crossing periods (time intervals) gained from sensorless commutation control block.

The speed controller works with a constant execution (sampling) period **PER_SPEED_SAMPLE_S** (request from timer interrupt).

6. Hardware

6.1 System Outline

The motor control system is designed to drive the 3-phase BLDC motor in a speed-closed loop.

The application can run on Freescale's motor control devices using the EVM Board:

- 56F803
- 56F805
- 56F807

The hardware setup of the system for a particular device varies only by the EVM Board used. The application software is identical for all devices, the EVM and chip differences are handled by SDK drivers for the particular EVM board.

Automatic board identification allows one program to run on each of three hardware and motor platforms without any parameter change:

- Low Voltage Evaluation Motor Hardware Set
- Low Voltage hardware set
- High Voltage Hardware Set

The hardware setup is shown in Figure 6-1, Figure 6-2 and Figure 6-3. More info can also be found in Section 11.1.

Notes: The detailed description of individual boards can be found in the User's Manuals for each board. The user manual incorporates the schematic of the board, description of individual function blocks and bill of materials. The individual boards can be ordered from Freescale as a standard product.

6.2 Low Voltage Evaluation Motor Hardware Set

The system configuration is shown in Figure 6-1.

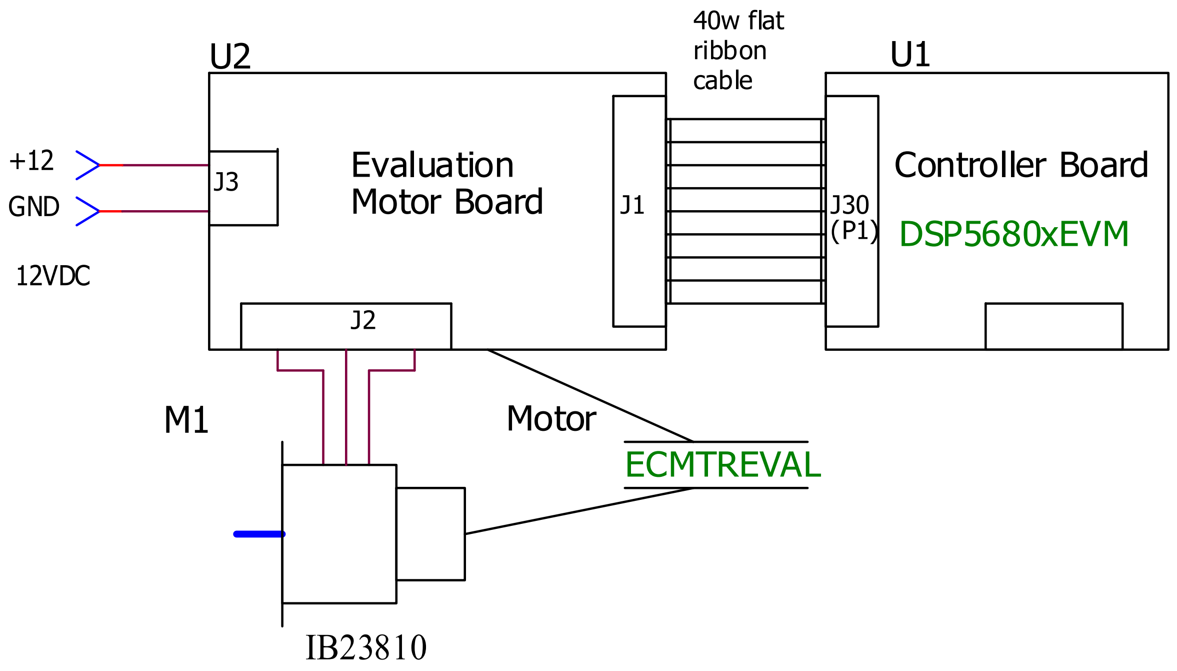


Figure 6-1. Low Voltage Evaluation Motor Hardware System Configuration

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

- M1 - IB23810 Motor
  - — supplied in kit with IB23810 Motor as: ECMTREVAL - Evaluation Motor Board Kit
- U2 3 ph AC/BLDC Low Voltage POWER STAGE:
  - — supplied in kit with IB23810 Motor as: ECMTREVAL - Evaluation Motor Board Kit
  - — described in: MEMCEVMBUM Evaluation Motor Board User’s Manual
- U1 CONTROLLER BOARD for 56F805:
  - — supplied as: 56F805EVM
  - — described in: 56F805EVMUM Evaluation Module Hardware User’s Manual
- or U1 CONTROLLER BOARD for 56F803:
  - — supplied as: 56F803EVM
  - — described in: 56F803EVMUM Evaluation Module Hardware User’s Manual

Information about boards and documents can be found at:
www.freescale.com

6.3 Low Voltage hardware set

The system configuration is shown in **Figure 6-2**.

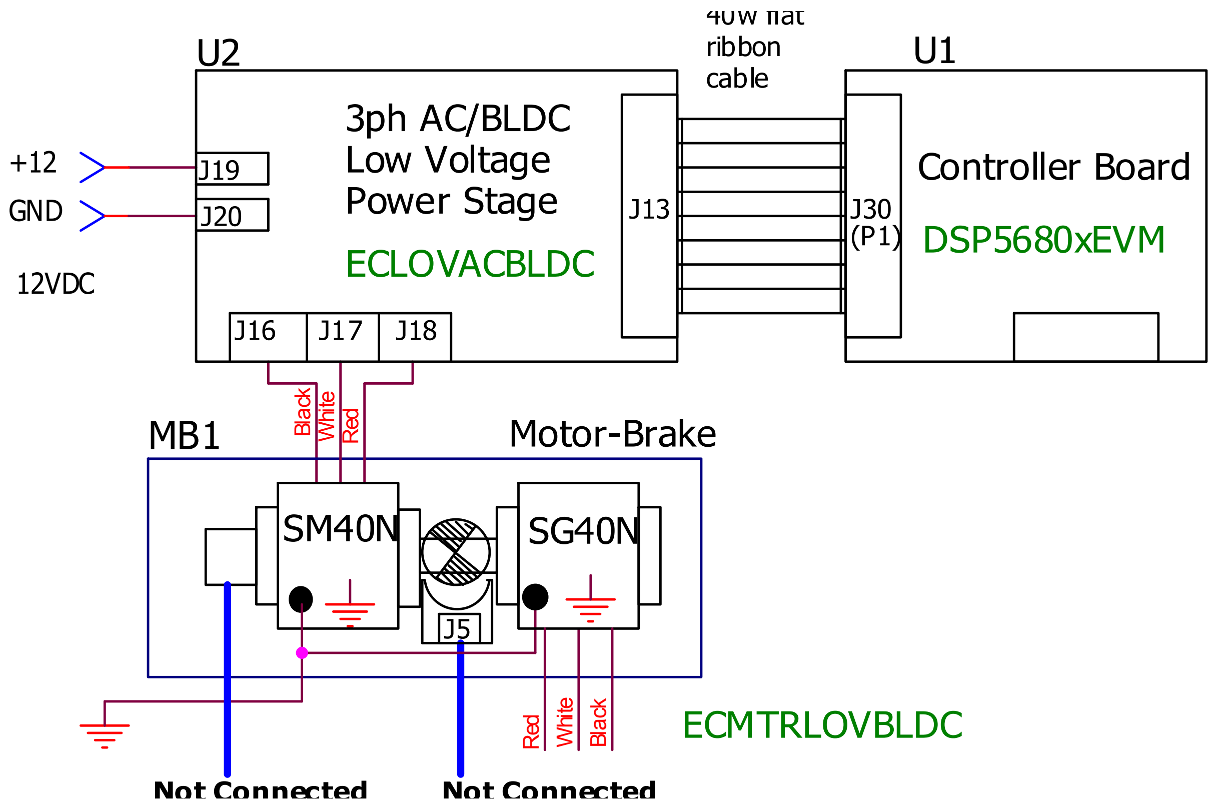


Figure 6-2. Low Voltage Hardware System Configuration

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

- U1 Controller Board for 56F805:
  - supplied as: 56F805EVM
  - described in: **56F805EVMUM Evaluation Module Hardware User's Manual**
- or U1 Controller Board for 56F803:
  - supplied as: 56F803EVM
  - described in: **56F803EVMUM Evaluation Module Hardware User's Manual**
- U2 - 3 ph AC/BLDC Low Voltage Power Stage
  - supplied as: ECLOVACBLDC
  - described in: **MEMC3PBLDCLVUM/D 3-phase Brushless DC Low Voltage Power Stage**
- MB1 - Motor-Brake SM40N + SG40N
  - supplied as: ECMTRLOVBLDC

Information about boards and documents can be found T:
www.freescale.com

6.4 High Voltage Hardware Set

The system configuration is shown in **Figure 6-3**

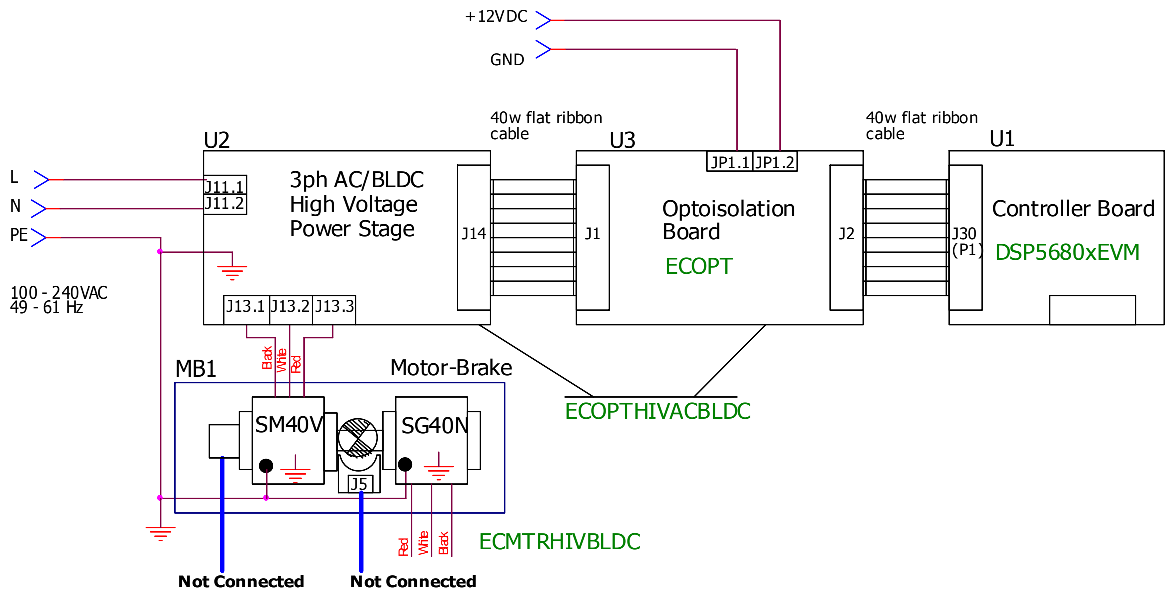


Figure 6-3. High Voltage Hardware System Configuration

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

- U1 - Controller Board for 56F805:
  - supplied as: 56F805EVM
  - described in: **56F805EVMUM Evaluation Module Hardware User's Manual**
- or U1 - Controller Board for 56F803:
  - supplied as: 56F803EVM
  - described in: **56F803EVMUM Evaluation Module Hardware User's Manual**
- U2 - 3-phase AC/BLDC High Voltage Power Stage
  - supplied in kit with Optoisolation Board as: ECOPTHIVACBLDC
  - described in: **MEMC3BLDCPSUM/D - 3-phase Brushless DC High Voltage Power Stage**
- U3 - Optoisolation Board
  - supplied with 3 ph AC/BLDC High Voltage Power Stage as: ECOPTHIVACBLDC
  - or supplied alone as: ECOPT - ECOPT optoisolation board
  - described in: **MEMCOBUM/D Optoisolation board User's Manual**

Warning: It is strongly recommended to use opto-isolation (optocouplers and optoisolation amplifiers) during the development time to avoid any damage to the development equipment.

- MB1 Motor-Brake SM40V + SG40N
  - supplied as: ECMTRHIVBLDC

Information about boards and documents can be found at:
www.freescale.com

7. Software Design

This section describes the design of the software blocks of the drive. The software is described in the following terms:

- Main Software Flow chart
- Data Flow
- State Diagram

For more information of the used control technique refer to **Section 5.**

7.1 Main Software Flow Chart

The main software flow chart incorporates the Main routine entered from Reset and interrupt states. The Main routine includes the initialization of the device and the main loop. (It is shown in **Figure 7-1**, and **Figure 7-2**.)

The main loop incorporates Application State Machine - the highest software level which proceeds settings for other software levels (BLDC motor Commutation Control, Zero Crossing Offset Control, Speed Control, Alignment Current Control). The inputs of Application State Machine is Run/Stop Switch state and Required Speed Omega and Drive Fault Status. Required Mechanical Speed can be set from the PC master software or manually by Up/Down buttons.

Commutation Control proceeds BLDC motor commutation with the states described in **Section 5.** and **Section 7.3.4**.

The Speed Control detailed description is in **Section 7.2.5** and **Section 7.3.7**. Alignment Current Control is described in **Section 7.2.6** and **Section 7.3.8**.

The Run/Stop switch is checked to provide an input for Application State Machine (ApplicationMode Start or Stop).

The interrupt subroutines provide commutation Timer services, ADC starting in the PWM reload interrupt, ADC service, ADC Zero Crossing checking, Limit analog values handling, and overcurrent and overvoltage PWM fault handling.

The Commutation Timer ISR is used for Commutation Timing and Commutation Control and Zero Crossing Checking.

The Speed/Alignment Timer ISR is used for Speed regulator time base and for Alignment state duration timing.

The PWM Reload ISR is used to start ADC conversion for ADC Zero Crossing and other channels and memorize the sampling time T_ZCSample.

The ADC Zero Crossing ISR is used to evaluate Back-EMF Zero Crossing.

The ADC completion ISR is used to read voltages, current and temperature samples from the ADC converter. It also sets Current control and Zero Crossing Offset Request flags when the Current Control or Zero Crossing Offset setting are enabled.

The other interrupts (**Figure 7-2**) are used for System Fault handling and setting of Required Mechanical Speed input for Application State Machine (ApplicationMode Start or Stop).

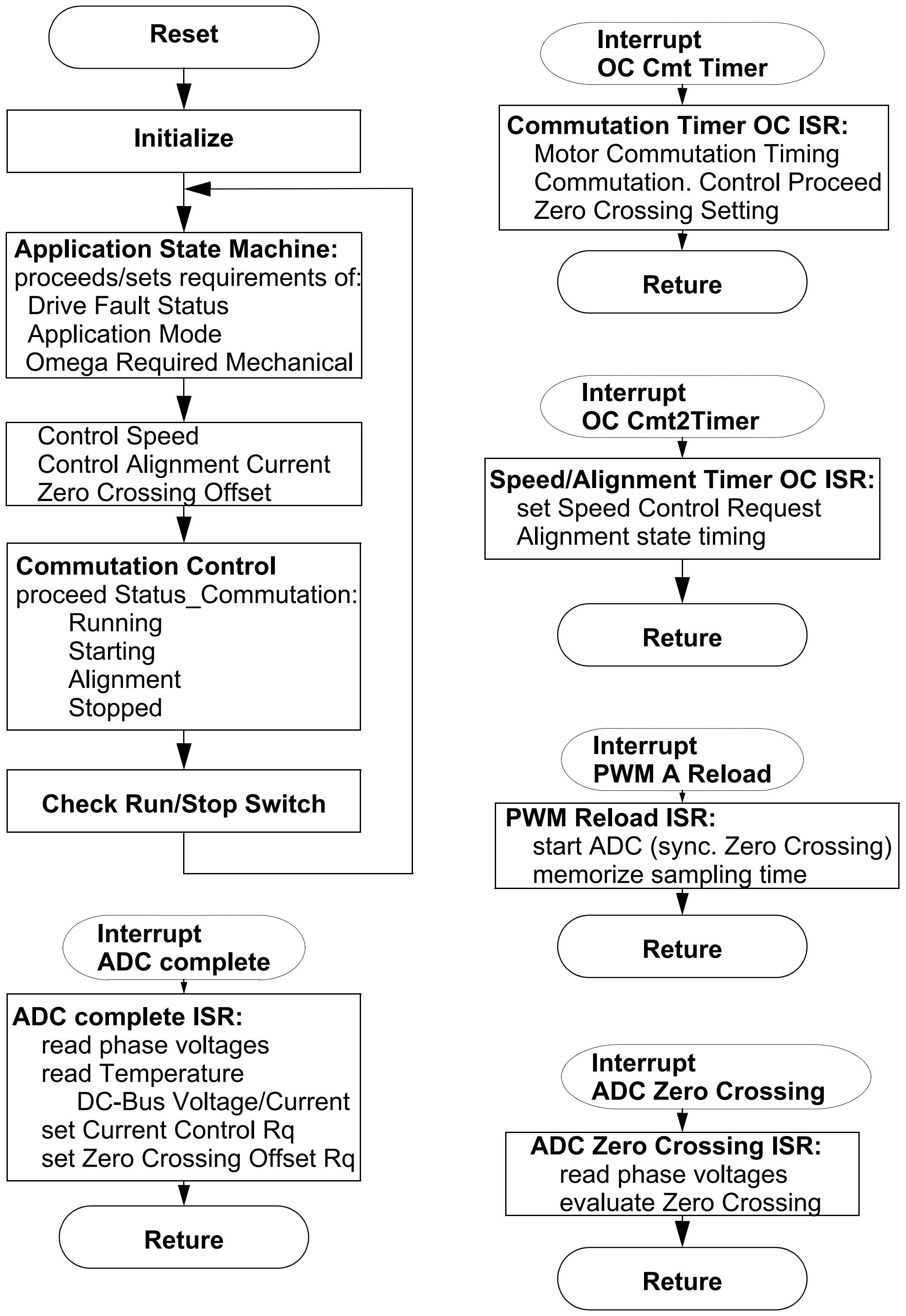


Figure 7-1. Main Software Flow Chart - Part 1

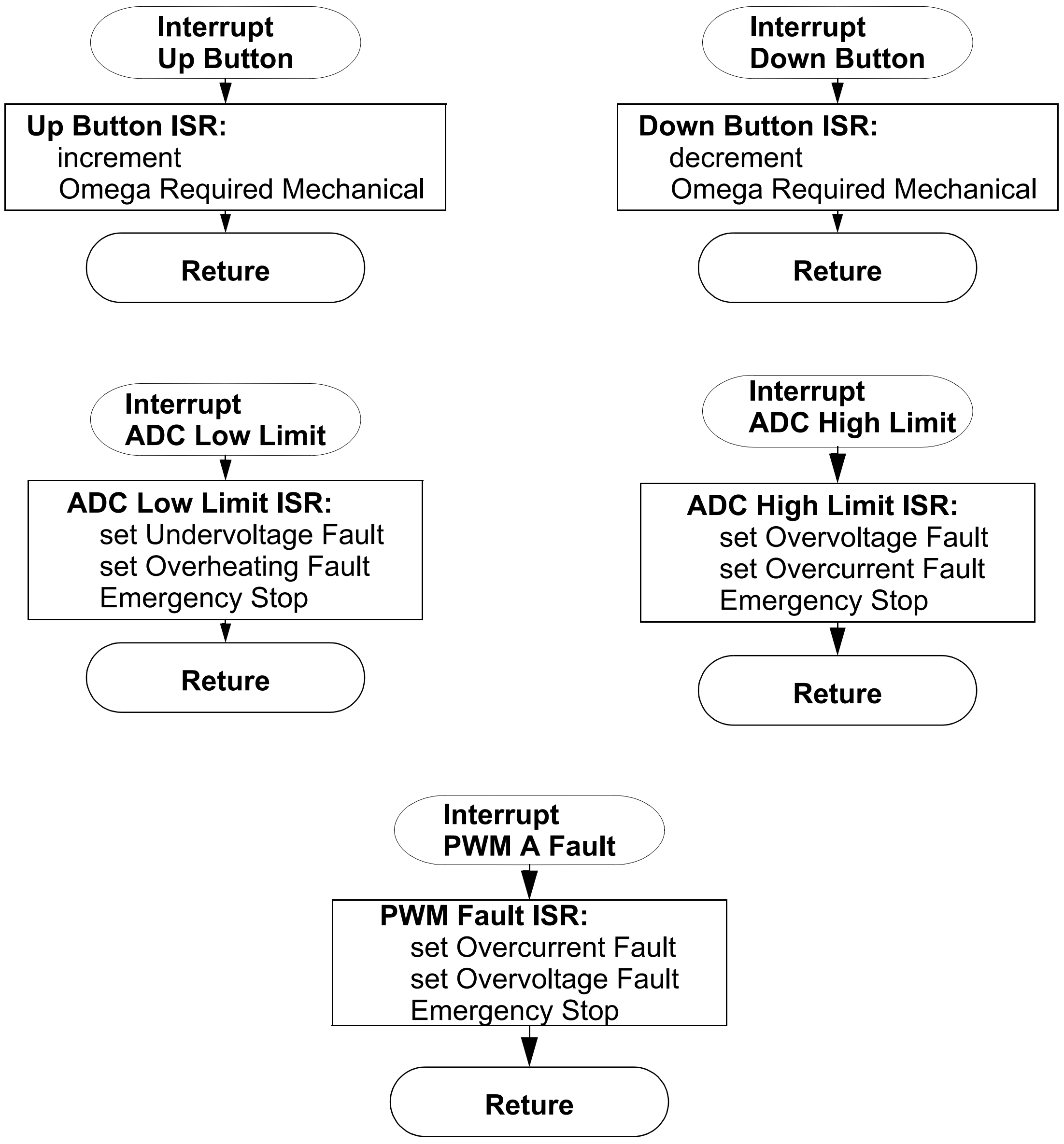


Figure 7-2. Main Software Flow Chart - Part 2

7.2 Data Flow

The control algorithm process values obtained from the user interface and sensors, generates 3-phase PWM signals for motor control (as can be seen on the data flow analysis).

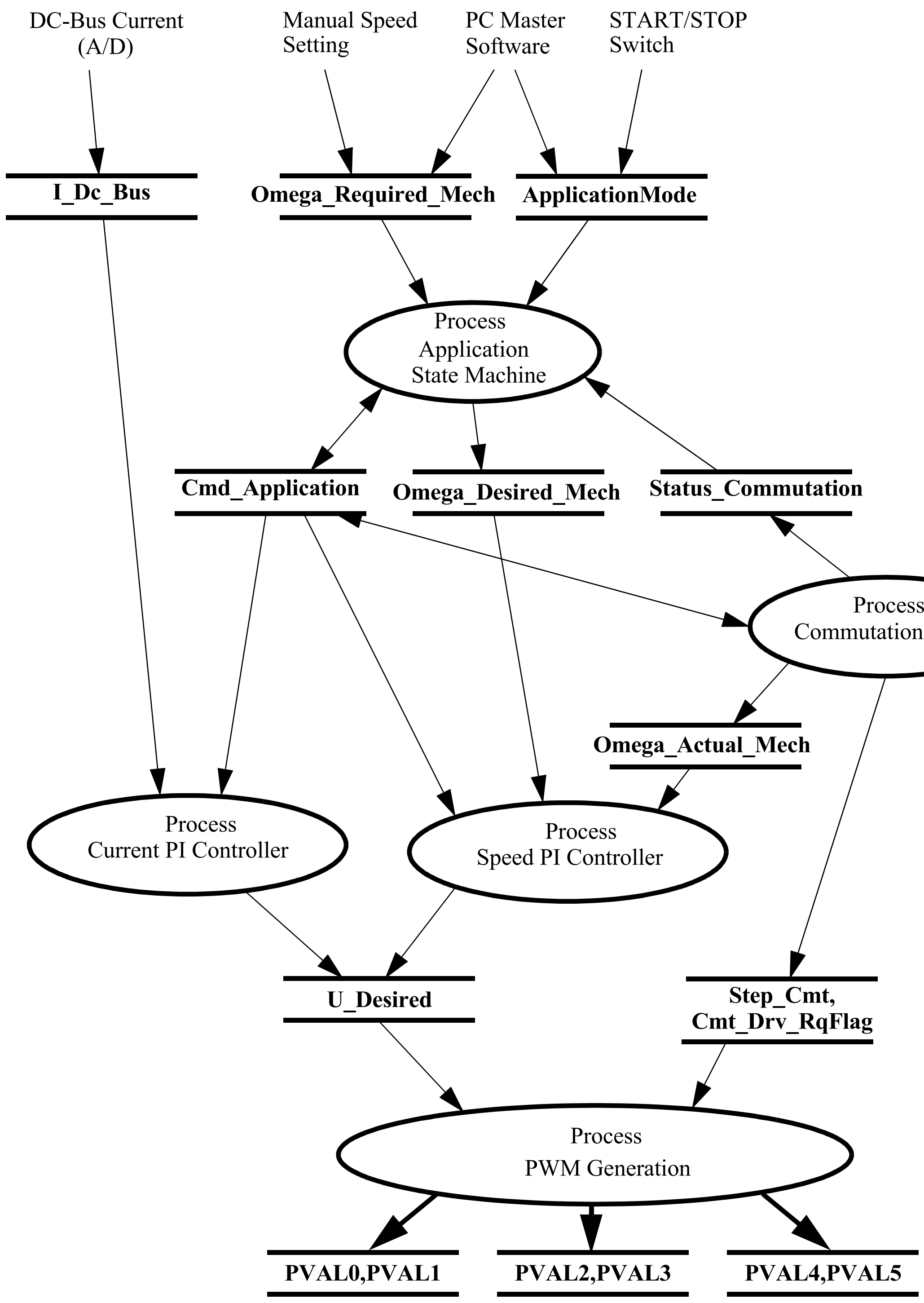


Figure 7-3. Data Flow - Part 1

The control algorithm of the BLDC motor drive with Back-EMF Zero Crossing using AD converter, is described in Figure 7-3 and Figure 7-4.

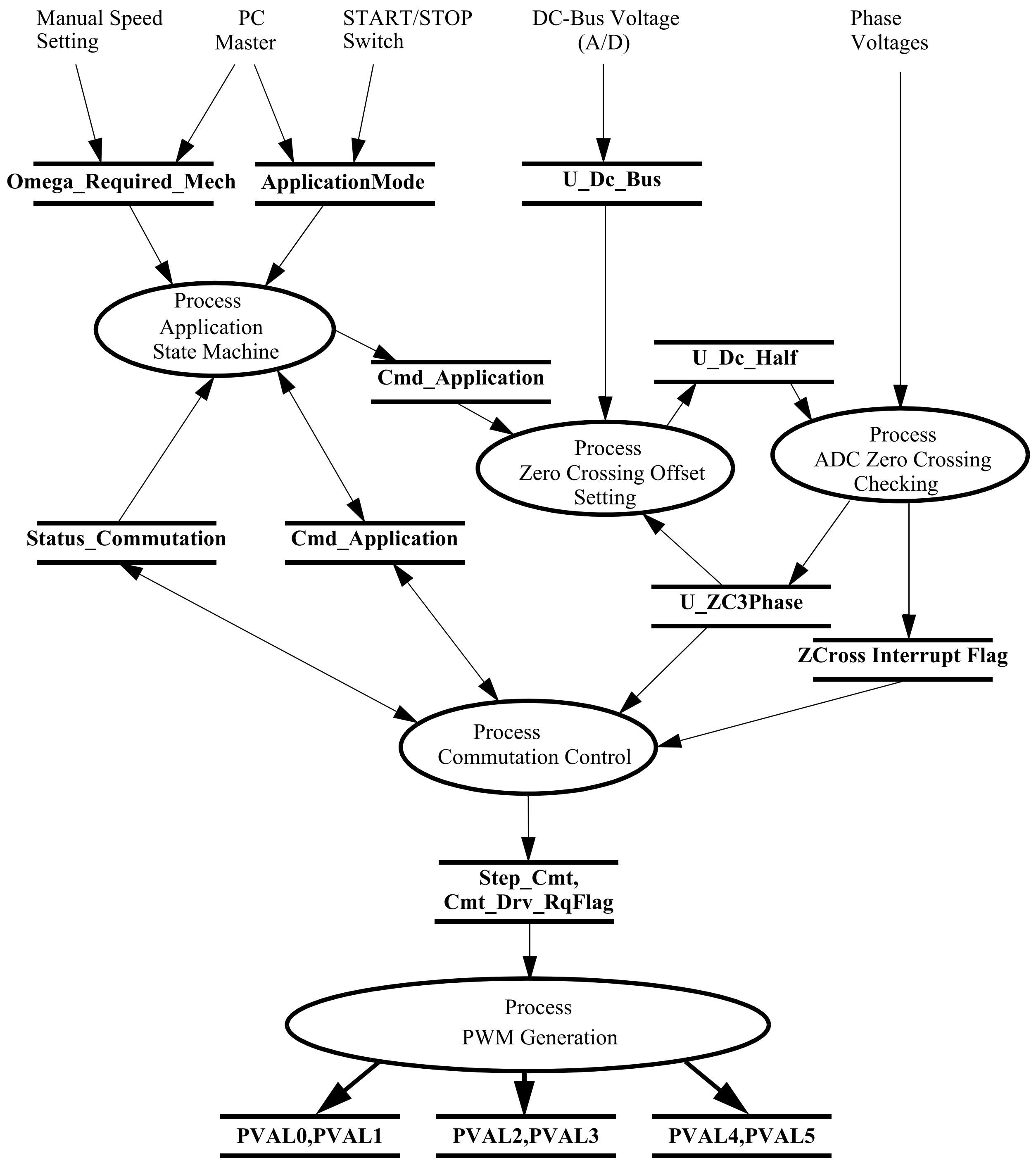


Figure 7-4. Data Flow - Part 2

Protection processes are shown in **Figure 7-5**. It consists of processes described in the following sub-sections.

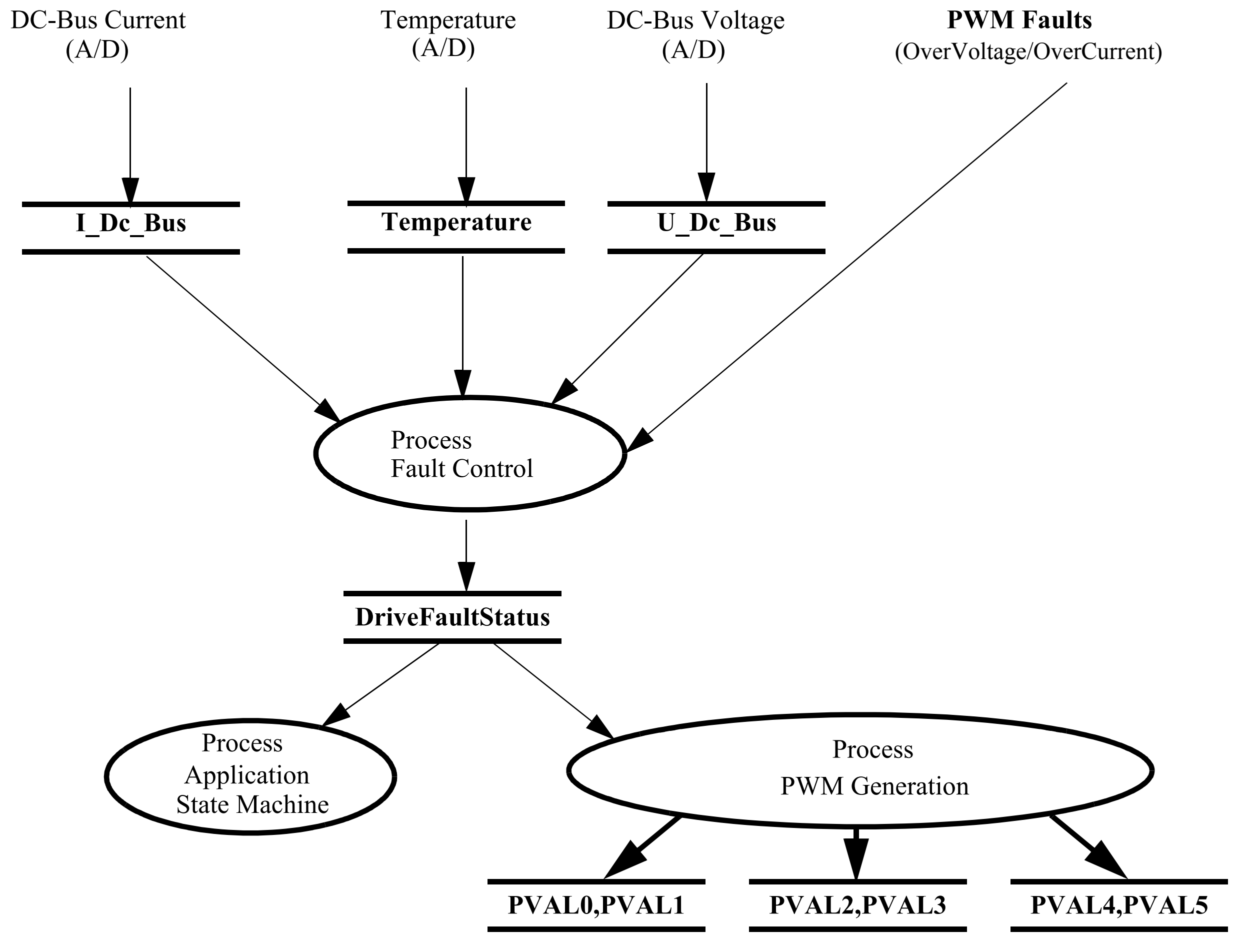


Figure 7-5. Data Flow - Part 3

7.2.1 Process Application State Machine

This process controls the application subprocesses by status and command words (**Figure 7-3** and **Figure 7-4**).

Based on the status of the **Cmd_Application** Cmd flags (set by the Commutation Control process) the **Cmd_Application** Rq flags are set to request calculation of the Current PI Controller (Alignment state) or Speed PI Controller (Running state) and to control the angular speed setting (reflects the status of the START/STOP Switch and the Run/Stop commands).

7.2.2 Process Commutation Control

This process controls sensorless BLDC motor commutations as explained in **Section 5.**. The process outputs, the **Step_Cmt** and the **Cmt_Drv_RqFlag,** are used to set the PWM Generation process. The output **Omega_Actual_Mech** is used for the Speed Controller process.

7.2.3 Process ADC Zero Crossing Checking

This process is based on the ADC Zero Crossing feature. When the free (not energized) phase branch voltage changes the sign comparing previous conversion results, the Zero Crossing interrupt is initiated. Then the BLDC motor commutation control is performed in the Zero Crossing ISR.

7.2.4 Process Zero Crossing Offset Setting

In order to assure proper behavior of the ADC Zero Crossing Checking, the ADC (Zero Crossing) Offset Registers are initialized the way that the zero Back-EMF voltage is converted to zero ADC value. The initialization is provided in two steps:

- Calibration of Phase Voltage Coefficients
- Setting of the ADC Offset Registers (**U_Dc_Bus_Half**) according to measured DC-Bus voltage

The ADC Offset Registers for all free phase voltages are set to **U_Dc_Bus_Half**.

The phase Zero Crossing calibration coefficient is obtained during the Alignment state (to reflect the unbalance of the sensing circuitry) from non-fed phase branch voltage measurements (average value) and DC-Bus voltage measurements (average value).

- **Coef_Calibr_U_Phx** = (**U_Dc_Bus_Half** + **U_Phx**)/**U_Dc_Bus**

During motor running (and starting) the **U_Dc_Bus_Half** is continuously updated based on the following formula:

- **U_Dc_Bus_Half** = **Coef_Calibr_U_Phx** * **U_Dc_Bus**

7.2.5 Process Speed PI Controller

The general principle of the speed PI control loop is illustrated in Figure 7-6.

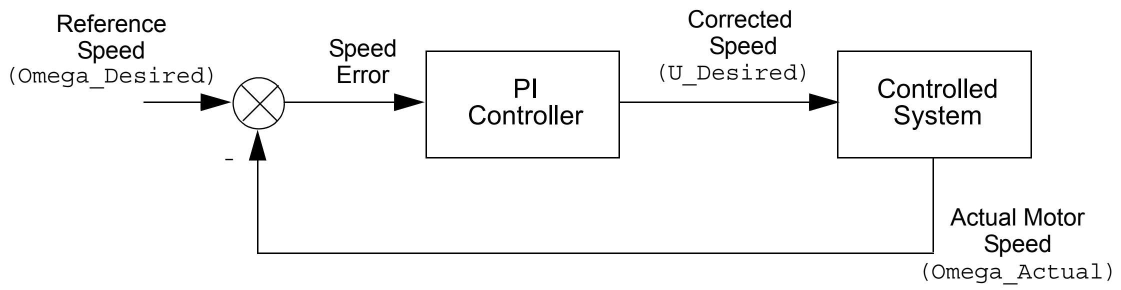


Figure 7-6. Closed Loop Control System

The speed closed loop control is characterized by the feedback of the actual motor speed. This information is compared with the reference set point and the 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 in order to compensate for the error.

The speed controller works with a constant execution (sampling) period. The request is driven from a timer interrupt with the constant **PER_SPEED_SAMPLE_S**. The PI controller proportional and integral constants were set experimentally.

7.2.6 Process Current PI Controller

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.

The current controller works with a constant execution (sampling) period which is determined by the following PWM frequency:

Current Controller period = 1/pwm frequency.

The PI controller proportional and integral constants were set experimentally.

7.2.7 Process PWM Generation

The Process PWM Generation creates the following:

- BLDC motor commutation pattern as described in **Section 1.**
- Required duty cycle

7.2.8 Process Fault Control

The Process Fault Control is used for drive protection (see **Figure 7-5**). The **DriveFaultStatus** is passed to the PWM Generation process and to the Application State Machine process in order to disable the PWMs and to control the application accordingly.

7.3 State Diagram

All the software state diagrams are described below.

7.3.1 Main Software States - General Overview

The software can be split into the following processes:

- Process Application State Machine
- Process Commutation Control
- Process ADC Zero Crossing Checking
- Process Zero Crossing Offset Setting
- Process Speed PI Controller
- Process Current PI Controller
- Process PWM Generation
- Process Fault Control

For detailed information, refer to **Section 7.2**. The general overview of the software states is in the State Diagram - Process Application State Machine; which is the highest level. (Only the process Fault Control is on the same level because of the motor emergency stop.)

The status of all the processes after reset is defined in **Section 7.3.2**.

7.3.2 Initialize

The Main software initialization provides the following actions:

- **CmdApplication** = 0
- **DriveFaultStatus** = NO_FAULT
- PCB Motor Set Identification
  - **boardId** function is used to detect one of three possible hardware sets. According to the used hardware, one of three control constant sets are loaded (functions **EVM_Motor_Settings, LV_Motor_Settings, HV_Motor_Settings**)
- ADC Initialization with Zero Crossing initialization
- LED diodes initialization
- Switch (Start/Stop) initialization
- Push Buttons (Speed up/down) initialization
- Commutation control initialization
- PWM initialization
- PWM fault interrupts initialization
- Output Compare Timers initialization

Notes: The EVM board can be connected to a different number of power stage boards. In order to assure the right hardware is connected, board identification is performed. When inappropriate hardware is detected, the **DriveFaultStatus**|=**WRONG_HARDWARE** is set and the motor remains stopped.

7.3.3 State Diagram - Process Application State Machine

The Process Application State Machine state diagram is displayed in **Figure 7-7**. The Application State Machine controls the main application functionality.

The application can be controlled as follows:

- Manually
- From PC master software

In the manual control, the application is controlled by the Start/Stop switch and the Up/Down push buttons to set Required Speed.

In the PC master software control mode the Start/Stop is controlled manually and the Required Speed is set using the PC master software.

The motor is stopped whenever the absolute value of Required speed is lower then Minimal Speed or switch set to stop or if there is a system failure - Drive Fault (Emergency Stop) state is entered. All the software processes are controlled according this Application State Machine status.

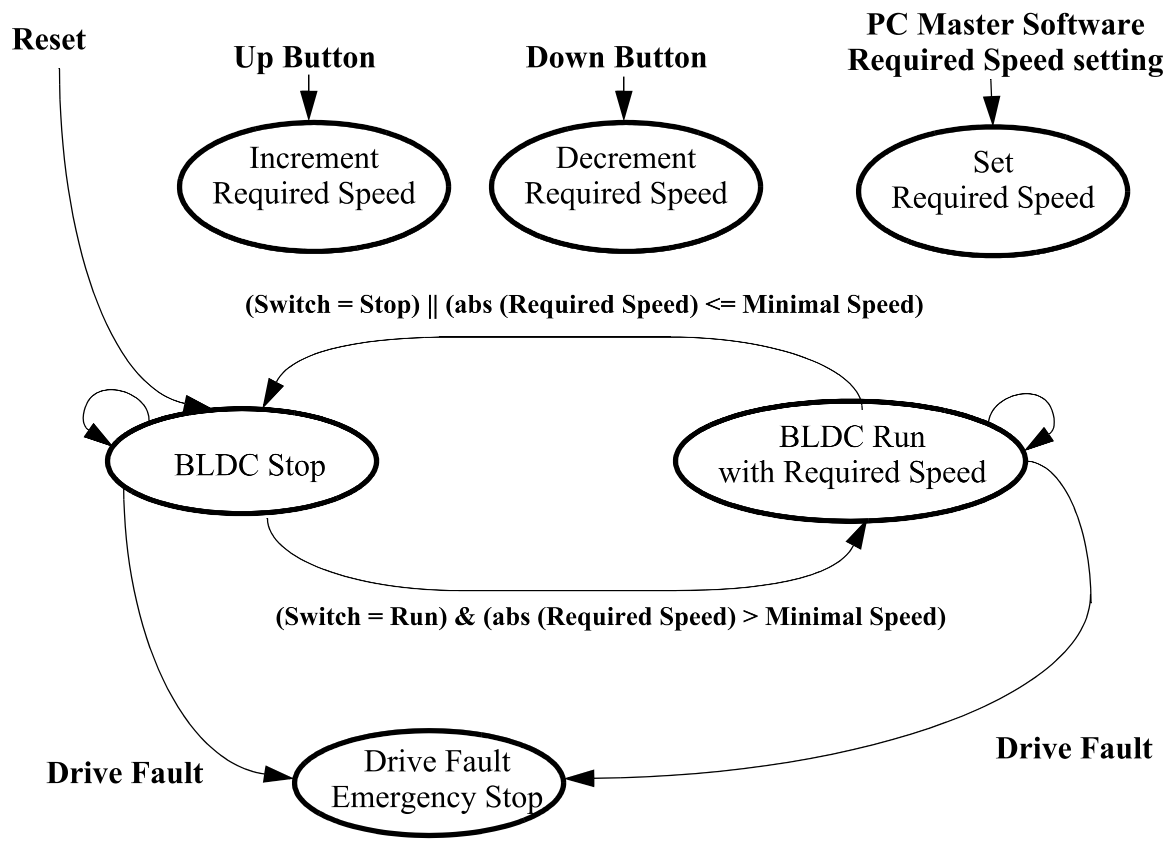

Reset
Up Button
Down Button
PC Master Software Required Speed setting
Increment Required Speed
Decrement Required Speed
Set Required Speed
(Switch = Stop) || (abs (Required Speed) <= Minimal Speed)
BLDC Stop
BLDC Run with Required Speed
(Switch = Run) & (abs (Required Speed) > Minimal Speed)
Drive Fault
Drive Fault Emergency Stop
Drive Fault

Figure 7-7. State Diagram - Process Application State Machine

7.3.4 State Diagram - Process Commutation Control

A State Diagram of the process Commutation Control is illustrated in **Figure 7-8**. The Commutation Control process takes care of the sensorless BLDC motor commutation. The requirement to run the BLDC motor is determined by the upper software level Application State Machine. When the Application State Machine is in the BLDC Stopped state, Commutation Control status is stopped. If it is in the BLDC Stopped state, the Commutation Control goes through the states described in **Section Section 5.**. The following states are possible:

- Alignment state
  - — Motor is powered with current through two phases - no commutations provided
- Starting (Back-EMF Acquisition) state
  - — Motor starts with first two commutations, then it is running (as in the Running state) using Start parameters for commutation calculation **StartComputInit** (so the commutation advance angle and the Per_Toff time are different)
- Running state
  - — Motor is running with Run parameters for commutation calculation **RunComputInit**
- Stopped state
  - — Motor is stopped with no power going to motor phases

The drive starts by setting the Alignment state where the Alignment commutation step is set and Alignment state is timed. After the time-out the Starting state is entered with initialization of Back-EMF Zero Crossing algorithms for the Starting state. After the required number of successive commutations with correct Zero Crossing, the Running state is entered. The Running and Starting states are exited to the Stopped state, if the number of commutations with wrong Zero Crossing exceeds the Maximal number. The commutation control is determined by the variable **StatusCommutation**.

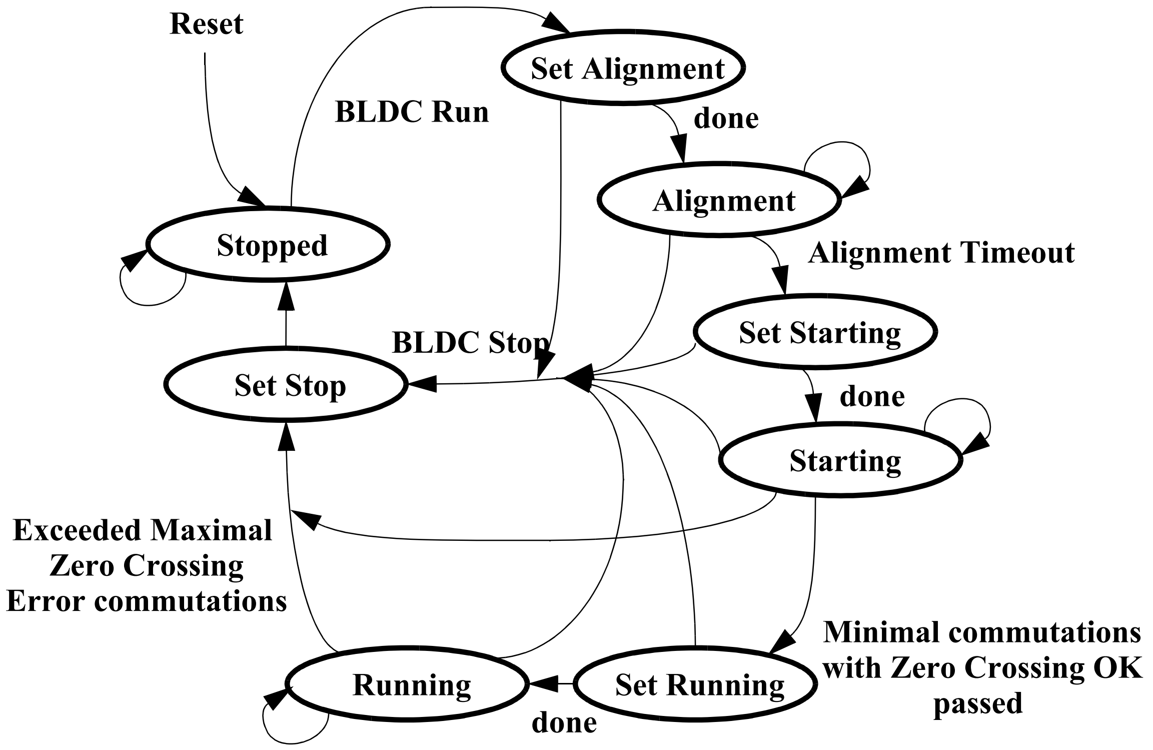


Figure 7-8. State Diagram - Process Commutation Control

7.3.4.1 Commutation Control - Running state

The State diagram of the Commutation Control Running state is shown in **Figure 7-9** and is explained in **Section 5.**. The selection of the state after the motor commutation depends on the detection of the Back-EMF Zero Crossing during the previous commutation period. If no Back-EMF Zero Crossing was detected, the commutation period is corrected (Corrective Calculation 1). Next, the Commutation time and commutation registers are preset. If Zero Crossing happens during the Per_Toff time period, the commutation period is corrected using the Corrective Calculation 2. When the commutation time expires, a new commutation is performed.

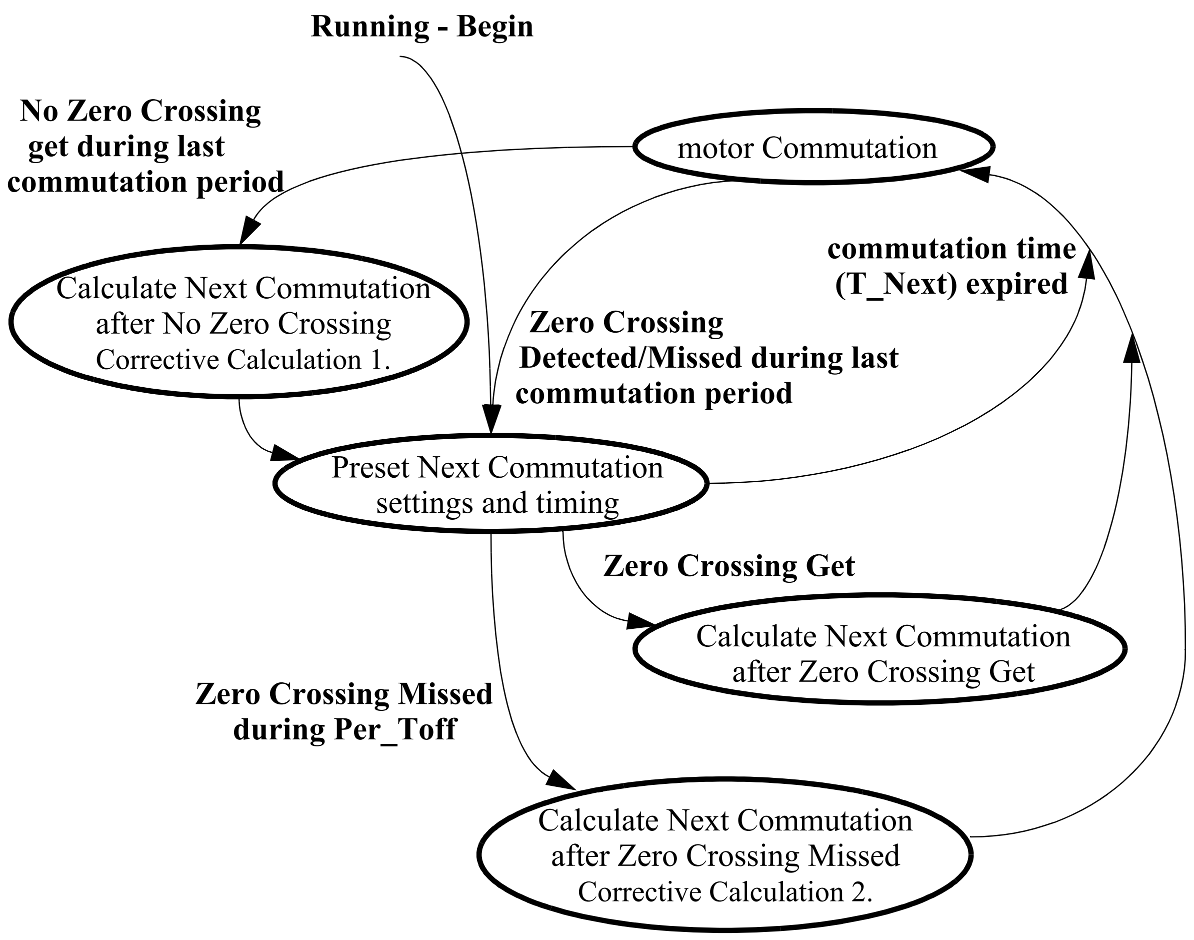

Running - Begin
No Zero Crossing get during last commutation period
motor Commutation
commutation time (T_Next) expired
Calculate Next Commutation after No Zero Crossing Corrective Calculation 1.
Zero Crossing Detected/Missed during last commutation period
Preset Next Commutation settings and timing
Zero Crossing Get
Calculate Next Commutation after Zero Crossing Get
Zero Crossing Missed during Per_Toff
Calculate Next Commutation after Zero Crossing Missed Corrective Calculation 2.

Figure 7-9. Substates - Running

This state is almost entirely serviced by the BLDC Zero Crossing algorithms which are documented in **Motor Control.pdf, Chapter 5, BLDC Motor Commutation with Zero Crossing Sensing)**. First, the **bldczcHndlr** is called with actual time from the Cmt Timer Counter to control requests and commutation control registers. Other BLDC Zero Crossing algorithms are called, according to the request flags. The state services are located in the main loop and in the Cmt (commutation) Timer Interrupt.

7.3.4.2 Commutation Control - Starting State

The Starting state is as the Running state, see **Figure 7-9**.

7.3.4.3 Commutation Control - Set Running

This state serves the transition from the Starting (Back-EMF Acquisition) state to Running state by the BLDC Zero Crossing algorithms (Section 11.1) according to the following actions:

- T_Actual = Cmt Timer Counter
- Setting new commutation parameters and initialized commutation with **bldczcHndlrInit** algorithm
- Initialization of computation with the **bldczcComputInit** algorithm

7.3.4.4 Commutation Control - Set Starting

This state is used to set the start of the motor commutation.

The following actions are performed in this state:

- Commutation initialized to start commutation step and required direction
- Two additional motor commutations are prepared (in order to create the starting torque)
- setting commutation parameters and commutation handler initialization by **bldczcHndlrInit** algorithm
- first action from **bldczcHndlrInit** algorithm (for commutations algorithms) is timed by Output Compare Timer for Commutation timing control (OC Cmt)
- PWM is set according the above prepared motor commutation steps
- Zero Crossing is initialized by bldcZCrosInit
- Zero Crossing computation is initialized by bldczcComputInit
- Zero Crossing is Enabled

7.3.4.5 Commutation Control - Set Stop

The following actions are performed in this state:

- **bldczcHndlrStop** algorithm is called
- PWM output is disabled in order to stop motor rotation and switch off the motor power supply.

7.3.4.6 Commutation Control - SetAlignment

In this state, the BLDC motor is set to the Alignment state, where voltage is put across two motor phases and a current is set to be at the required value. The following actions are provided in the Set Alignment state:

- PWM set according to **Align_Step_Cmt** variable status
- Current controller is initialized
- PWM output is enabled
- Aligned Time is timed by Output Compare Timer for Speed and Alignment

7.3.5 State Diagram - Process ADC Zero Crossing Checking

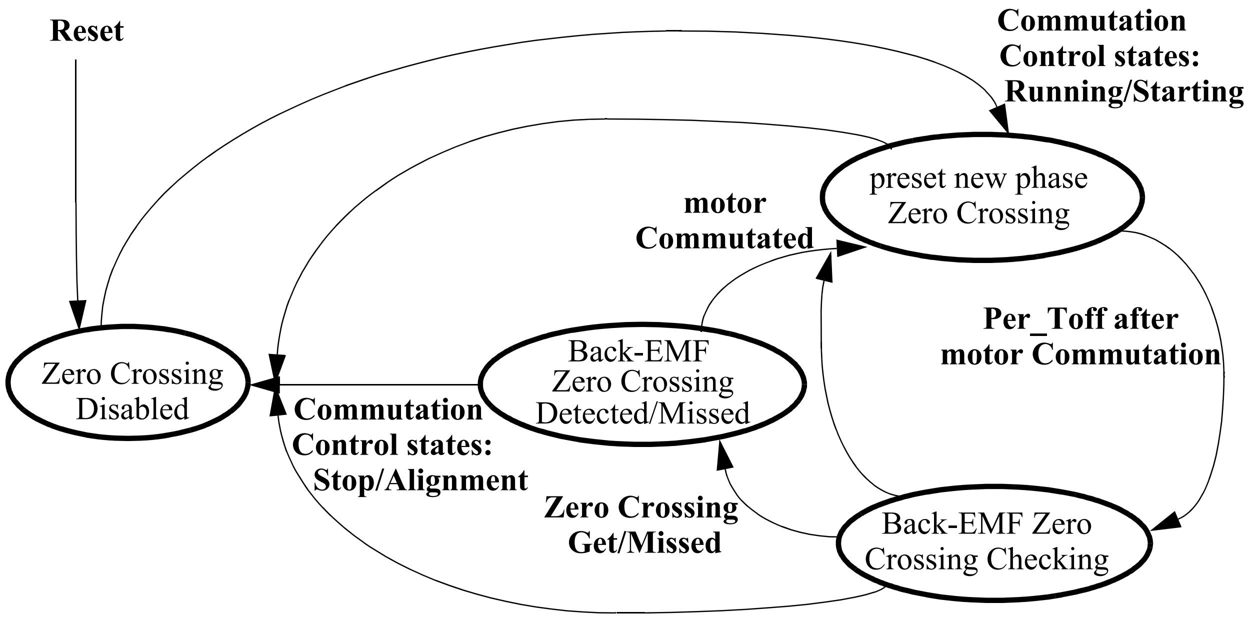

Reset
Commutation Control states: Running/Starting
preset new phase Zero Crossing
motor Commutated
Per_Toff after motor Commutation
Zero Crossing Disabled
Back-EMF Zero Crossing Detected/Missed
Commutation Control states: Stop/Alignment
Zero Crossing Get/Missed
Back-EMF Zero Crossing Checking

Figure 7-10. State Diagram - Process ADC Zero Crossing Checking

The status of the ADC Zero Crossing checking depends on Commutation Control states. It is enabled only during the Running and Starting Commutation Control states. When the BLDC motor is commuted, the new Zero Crossing phase is preset. When, after motor commutation, the time interval (Per_Toff) is passed, the Back-EMF Zero Crossing is checked. When the Back-EMF Zero Crossing is Detected or Missed, the Commutation control process is informed that a new commutation time needs to be computed.

This process is almost entirely provided by the BLDC Zero Crossing algorithms. The **bldczcHndlr** is called with actual time from the Cmt Timer Counter. This algorithm is assigned to control requests and set some commutation control registers. According to the request flags set by this algorithm, other BLDC Zero Crossing algorithms are called.

This process, together with the Commutation Control Running state, are the most important software processes for sensorless BLDC motor commutation. These process services are located in the main loop and in the ADC Interrupt subroutine.

7.3.6 State Diagram - Process ADC Zero Crossing Offset Setting

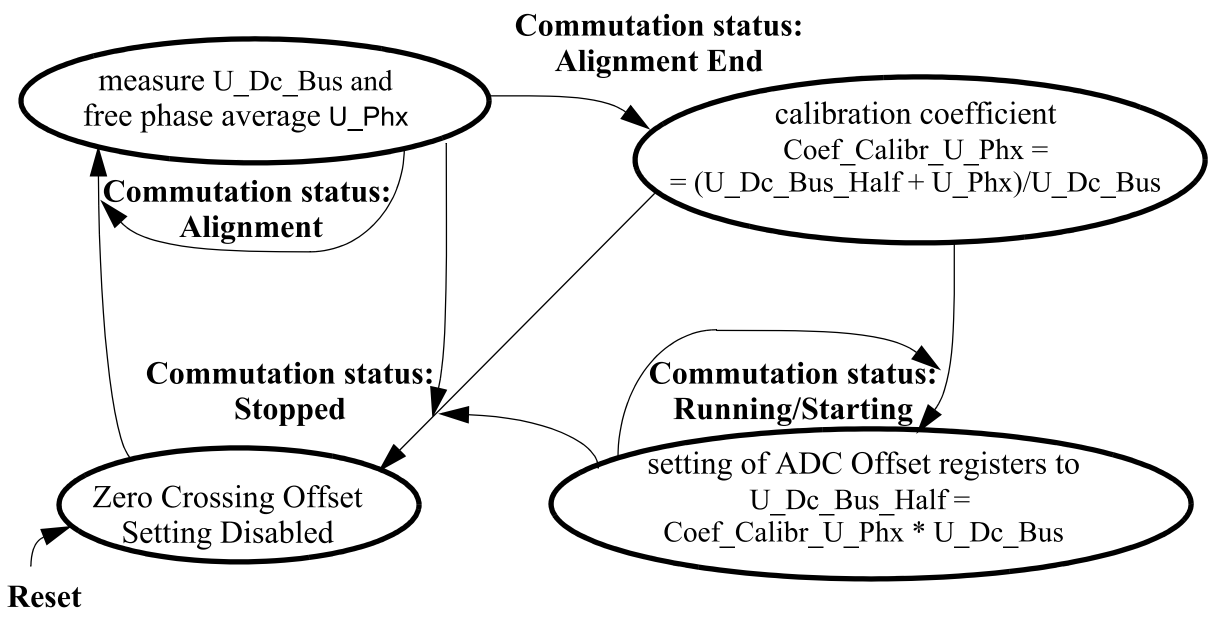


Figure 7-11. State Diagram - Process ADC Zero Crossing Offset Setting

The **Figure 7-11** state diagram describes the tuning process of the ADC (Zero Crossing) Offset Registers described in **Section 5.** and **Section 7.2.4**.

7.3.7 State Diagram - Process Speed PI Controller

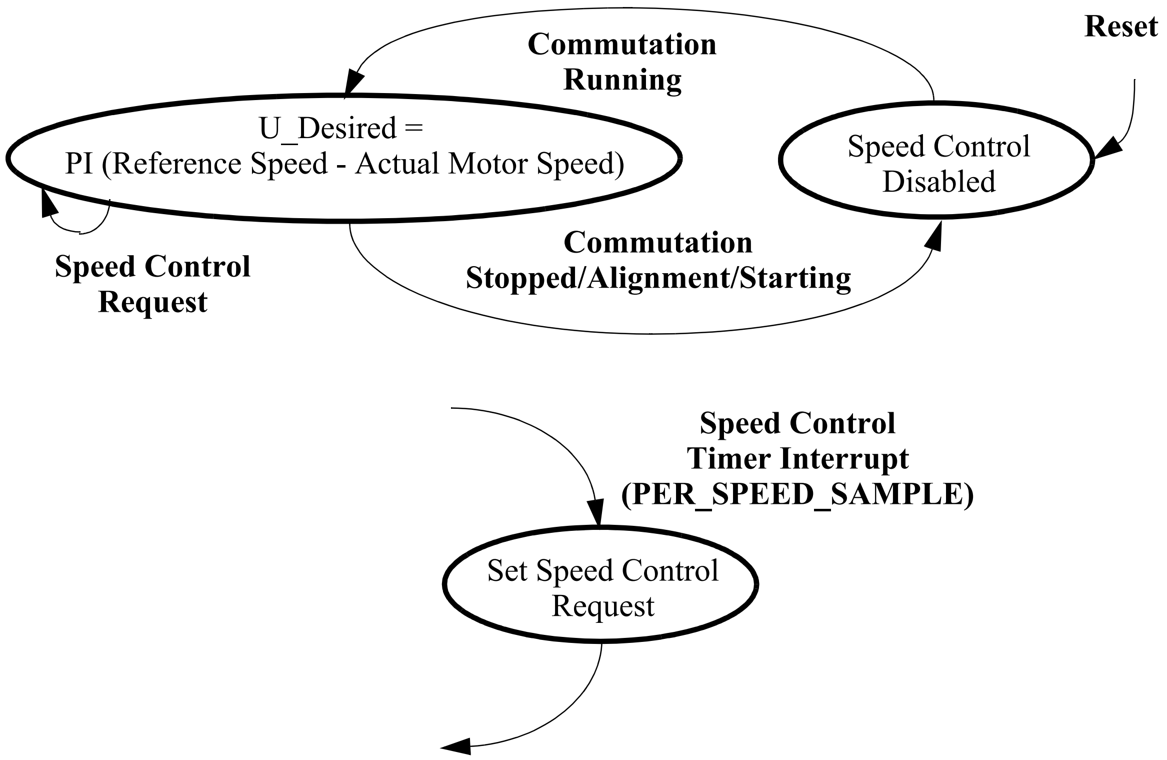

Commutation Running
Reset
U_Desired = PI (Reference Speed - Actual Motor Speed)
Speed Control Disabled
Speed Control Request
Commutation Stopped/Alignment/Starting
Speed Control Timer Interrupt (PER_SPEED_SAMPLE)
Set Speed Control Request

Figure 7-12. State Diagram - Process Speed PI Controller

The Speed PI controller algorithm controllerPItype1 is described in the SDK documentation. The controller execution (sampling) period is PER_SPEED_SAMPLE, period of Speed Control Timer Interrupt.

7.3.8 State Diagram - Process Current PI Controller

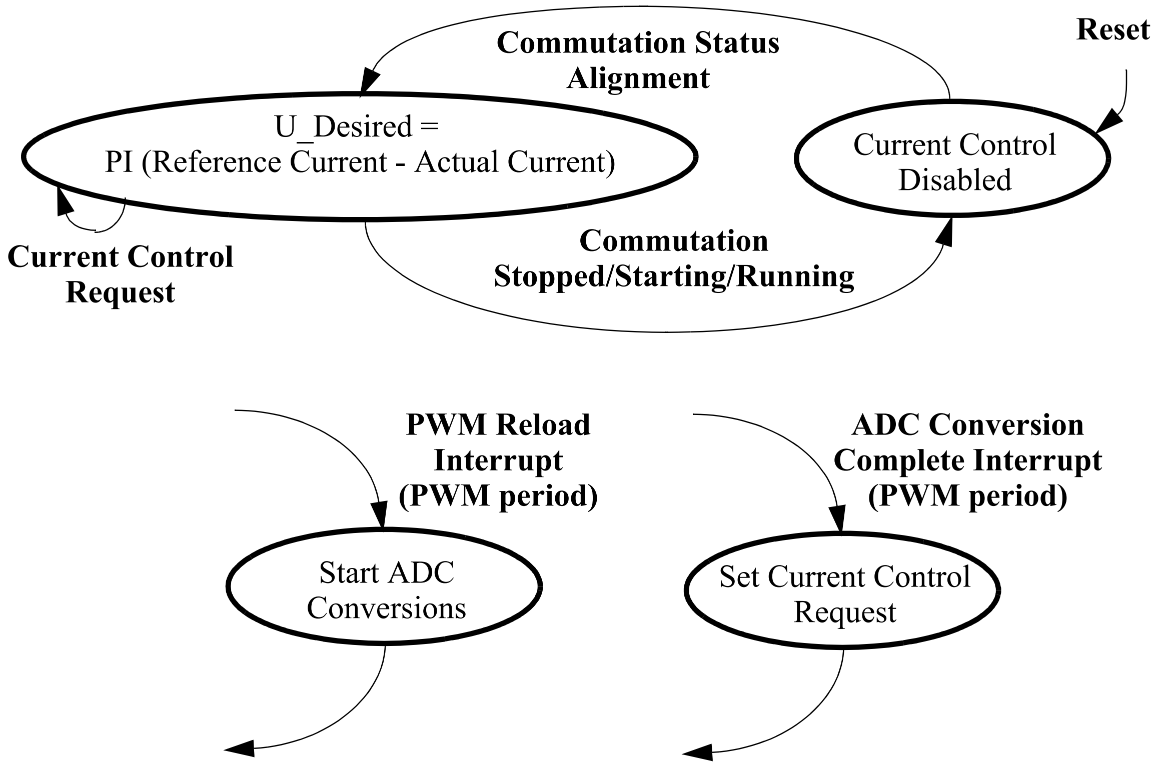


Figure 7-13. State Diagram - Process Speed PI Controller

The Current PI controller algorithm, **controllerPItype1**, is described in the SDK documentation. The controller execution (sampling) period is determined by the PWM module period, since the ADC conversion is started each PWM reload (once per PWM period). The Current Control Request is set in ADC Conversion Complete Interrupt.

7.3.9 State Diagram - Process Fault Control

The process Fault State is described in Interrupt subroutines.

7.3.9.1 PWM Fault A Interrupt Subroutine

This subroutine is called at PWM A (or PWM in case of 56F803) Fault Interrupt.

In this interrupt subroutine, the following faults from the PWM Fault pins are processed:

- When Overvoltage occurs (the Overvoltage fault pin set)
  - **DriveFaultStatus** |= **OVERVOLTAGE**
- When Overcurrent occurs (the Overcurrent fault pin set)
  - **DriveFaultStatus** |= **OVERCURRENT**

7.3.9.2 ADC Low Limit Interrupt Subroutine

This subroutine is called when at least one ADC low limit is detected.

In this interrupt subroutine, the following low limit exceeds are processed:

- The undervoltage of the DC-Bus voltage
  - DriveFaultStatus |= UNDERVOLTAGE_ADC_DCB,
- The over temperature (detected here because of the sensor reverse temperature characteristic)
  - DriveFaultStatus |= OVERHEATING.

7.3.9.3 ADC High Limit Interrupt Subroutine

This subroutine is called when at least one ADC high limit is exceeded.

In this interrupt subroutine, the following high limit exceeds are processed:

- The overvoltage of the DC-Bus voltage
  - **DriveFaultStatus** |= **OVERVOLTAGE_ADC_DCB**
- The overcurrent of the DC-Bus current input
  - **DriveFaultStatus** |= **OVERCURRENT_ADC_DCB**

8. PC Master Software

PC master software was designed to provide the debugging, diagnostic and demonstration tool for development of algorithms and applications. It consists of components running on a PC and, partially, running on the target development board.

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

The baud rate of the SCI communication is 9600Bd. It is set automatically by the PC master software driver.

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

```
#define SCI_DRIVER
#define INCLUDE_PCMASTER
```

Refer to **Section 7.** PC master software automatically includes the SCI driver and installs all necessary services.

The detailed PC master software description is provided by the **PC Master User Manual**.

9. DSC Usage

Figure 9-1 shows how much memory is needed to run the BLDC motor drive with Back-EMF Zero Crossing using ADC in a speed-closed loop. A part of the device's memory is still available for other tasks.

Table 9-1. RAM and FLASH Memory Usage for SDK2.2

Memory (in 16 bit Words)	**Available** 56F803 56F805	**Used** Application + Stack	**Used** Application without PC master software, SCI
Program FLASH	31.5K	14288	10568
Data RAM	2K	1481+352	1145+352

10. Setting of Software Parameters for Other Motors

The software was tuned for three hardware and motor kits (EVM, LV, HV) as described in Section 6. and Section 4.1. It can of course be used for other motors, but the software parameters need to be set accordingly.

Most of the parameters are located in the file (External RAM version):

...\dsp5680xevm\nos\applications\bldc_adc_zerocross\bldcadczcdefines.h

and config files:

...\dsp5680xevm\nos\applications\bldc_adc_zerocross\configextram\appconfig.h

or in the file (Flash version):

...\dsp5680xevm\nos\applications\bldc_adc_zerocross\configFlash\appconfig.h

The motor control drive usually needs setting/tuning of the following:

- Dynamic parameters
- Current/voltage parameters

The software selects valid parameters (one of the three parameter sets) based in the identified hardware. The table below shows the starting string of the software constants used for each hardware set.

Table 10-1. Software Parameters Marking

Hardware Set	**Software Parameters Marking**
Low Voltage Evaluation Motor Hardware Set	EVM_yyy
Low Voltage hardware set	LV_yyy
High Voltage Hardware Set	HV_yyy

In the following text the EVM, LV, HV are replaced with x. The sections are sorted in recommended order (when one is tuning/changing parameters).

Notes: The most important constants for reliable motor start-up are described in Section 10.2.2 and in Section 10.1.2.

10.1 Current and Voltage Settings

10.1.1 DC-Bus Voltage, Maximal and Minimal Voltage and Current Limit Settings

For the right regulator settings, it is required to set the expected DC-Bus voltage in *bldcadczcdefines.h:*

```
#define x_VOLT_DC_BUS          12.0      /* DC-Bus expected voltage */
```

The current voltage limits for software protection are:

```
#define x_DCB_UNDERVOLTAGE      3.0      /* Undervoltage limit [V] */
#define x_DCB_OVERVOLTAGE      15.8      /* Overvoltage limit [V] */
#define x_DCB_OVERCURRENT      48.0      /* Overcurrent limit [A] */
```

Notes: Note the hardware protection with setting of pots R116, R71 for 56F805EVM or R40, R45 for 56F803EVM (see EVM manuals for details)

10.1.2 Alignment Current and Current Regulator Setting

All settings in this section are in *bldcadczcdefines.h.*

The current during the Alignment state (before the motor starts) is recommended to be set to the current nominal motor value:a

#define x_CURR_ALIGN_DESIRED_A 17.0 /* Alignment Current Desired [A] */

Usually it is necessary to set the PI regulator constants. (The PI regulator is described in algorithm **controllerPItype1** description in SDK doc).

The current controller works with a constant execution (sampling) period determined by the PWM frequency:

Current Controller period = 1/pwm frequency.

Both proportional and integral gain have two coefficients: gain portion and scale.

Current Proportional gain:

```
#define x_CURR_PI_PROPORTIONAL_GAIN 30000      /* proportional gain portion */
#define x_CURR_PI_PROPORTIONAL_GAIN_SCALE 24   /* proportional gain scale*/
```

Current Integral gain:

```
#define x_CURR_PI_INTEGRAL_GAIN  19000        /* integral gain portion */
#define x_CURR_PI_INTEGRAL_GAIN_SCALE 23      /* integral gain gain scale */
```

The PI controller proportional and integral constants can be set experimentally.

Notes: If the overcurrent fault is experienced during the Alignment state, then it is recommended to slow down the regulator. If the yy_GAIN_SCALE is increased, the gain is decreased.

Notes: The coefficients x_CURR_PI_PROPORTIONAL_GAIN_REAL (resp. x_CURR_PI_INTEGRAL_TI_REAL) are not directly used for regulator setting, but can be used to calculate the `x_CURR_PI_PROPORTIONAL_GAIN, x_CURR_PI_PROPORTIONAL_GAIN_SCALE` (resp. `x_CURR_PI_INTEGRAL_GAIN, x_CURR_PI_INTEGRAL_GAIN_SCALE`) using the formulae in the comments.

10.2 Commutation Control Settings

In order to get the motor reliably started, the commutation control constants must be properly set.

10.2.1 Alignment Period

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

This duration is set in seconds in *bldcadczcdefines.h.*

```
#define x_PER_ALIGNMENT_S          0.5    /* Alignment period [s] */
```

Notes: For the first tuning, it is recommended to set this duration long enough (for example, 5 seconds). Then, if the motor works well, it can be significantly shortened (for example, 0.1 seconds).

10.2.2 Start-up Periods

The constants defining the start up need to be changed according to the drive dynamic.

All settings in this section are in *bldcadczcdefines.h:*

```
#define x_PER_CMTSTART_US        7200.0      /* Start Commutation Period [micros] */
#define x_PER_TOFFSTART_US      14400.0      /* Start Zero Crossing
                                               Toff Period [micros] */
```

`x_PER_CMTSTART_US` is the commutation period used to compute the first (start) commutation period.

`x_PER_TOFFSTART_US` is the first (start) Toff interval after commutation (where Back-EMF Zero Crossing is not sensed).

The unit of this constant is 1 µs.

Notes: It is recommended to set `x_PER_TOFFSTART_US = 2*x_PER_CMTSTART_US`.

Next, set the first motor commutation period = `x_PER_CMTSTART_US * 2`

The Back-EMF Zero Crossing is not sensed during the first duration since it is very short. Hence, the Zero Crossing information is not reliable during this period.

Notes: Setting these constants is an experimental process. It is difficult to use a precise formula because there are many factors involved which are difficult to obtain in case of real drive (motor and load mechanical inertia, motor electromechanical constants, and sometimes also the motor load). Therefore, constants need to be set for a specific motor.

Table 10-2 aids in the setting of constants.

Table 10-2. Start-up Periods

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

Notes: Slowing down the speed regulator (see **Section 10.3.1**) helps if a problem with start up (using the above stated setting) is encountered.

10.2.3 Minimal Zero Commutation of Starting (Back-EMF Acquisition) State

```
#define x_MIN_ZCROSOK_START    0x02       /* minimal Zero Crossing OK commutation
                                              to finish BLDC starting phase */
```

This constant **x_MIN_ZCROSOK_START** determines the minimal number of Zero Crossing OK commutation to finish the BLDC starting phase.

Notes: It is recommended to use only the value 0x02 or 0x03. If this constant is set too high, the motor control does not enter the Running state fast enough.

10.2.4 Wrong Zero Crossing

```
#define x_MAX_ZCROSERR 0x04 /*Maximal Zero Crossing Errors (to stop commutations) */
```

The constant **x_MAX_ZCROSERR** is used for control of commuting problems. The application software stops and starts the motor whenever successive **x_MAX_ZCROSERR** commutations (with problematical Zero Crossings) appear.

Notes: During tuning of the software for other motors, this constant can temporarily be enlarged.

10.2.5 Commutation Proceeding Period

The Commutation proceeding period is the constant time after motor commutation; when Back-EMF Zero Crossing is not measured (commutation current decay period).

```
#define x_CONST_PERPROCCMT_US 170.0  /* Period of Commutation proceeding [micros]*/
```

The unit of this constants is 1 µs.

Notes: This constant needs to be lower then 1/3 of the (minimal) commutation period at motor maximal speed.

10.2.6 Commutation Timing Setting

Notes: Normally these structures do not require change. If the constants described in this section need to be changed, a detailed study of the control principle needs to be studied (see **Section 5.** and **MotorControl.pdf**).

If it is required to change the motor commutation advancing (retardation) the coefficients in starting and running structures needs to be changed:

```
x_StartComputInit
x_RunComputInit
```

Both structures are in bldcadczcdefines.h.

The **x_StartComputInit** structure is used by the application software during Starting state (see **Section 5.4.3**).

The **x_RunComputInit** structure is used by the application software during Running state see **Section 5.4.3.1**).

```
Coef_CmtPrecompLShft
Coef_CmtPrecompFrac
```

fractional and scaling part of **Coef_CmtPrecomp**

final Coef_CmtPrecomp = **Coef_CmtPrecompFrac** << **Coef_CmtPrecompLShft**

this final **Coef_CmtPrecomp** determines the interval between motor commutations when no Back-EMF Zero Crossing is captured. The application software multiplies fractional Coef_CmtPrecomp **with commutation period.**

```
Coef_HlfCmt
```

determines Commutation advancing (retardation) - the interval between Back-EMF Zero Crossing and motor commutation

The application software multiplies fractional **Coef_HlfCmt** with commutation period.

```
Coef_Toff
```

determines the interval between Back-EMF Zero Crossing and motor commutation

The application software multiplies fractional **Coef_Toff** with commutation period

10.3 Speed setting

10.3.1 Maximal and Minimal Speed and Speed Regulator Setting

All these section settings are in *bldcadczcdefines.h*.

In order to compute the speed setting, it is important to set number of BLDC motor commutations per motor mechanical revolution:

```
#define x_MOTOR_COMMUTATION_PREV   18    /* Motor Commutations Per Revolution */
```

Maximal required speed in rpm is set by:

```
#define x_SPEED_ROTOR_MAX_RPM    3000    /* maximal rotor speed [rpm] */
```

If you also request to change the minimal motor speed, then you need to set minimal angular speed

```
#define x_OMEGA_MIN_SYSU         4096  /* angular frequency minimal [system unit] */
```

Notes: Remember that minimal angular speed is not in radians, but in system units! Where 32768 is the maximal speed done by `x_SPEED_ROTOR_MAX_RPM`

The speed PI regulator constants can be tuned as described below. All settings can be found in *bldcadczcdefines.h.*

The execution period of the speed controller is set by:

```
#define PER_SPEED_SAMPLE_S 0.001   /* Sampling Period of the Speed Controller [s] */
```

Both proportional and integral gain has two coefficients: portion and scale!

Speed Proportional gain:

```
#define x_SPEED_PI_PROPORTIONAL_GAIN      22000 /* speed proportional gain portion*/
#define x_SPEED_PI_PROPORTIONAL_GAIN_SCALE  19 /* speed proportional gain scale*/
```

Speed Integral gain:

```
#define x_SPEED_PI_INTEGRAL_GAIN          27500 /* speed integral gain portion */
#define x_SPEED_PI_INTEGRAL_GAIN_SCALE       23 /* speed integral gain gain scale */
```

The PI controller proportional and integral constants can be set experimentally.

Notes: If the motor has problems when requested speed is changed, then it is recommended to slow down the regulator. If the yy_GAIN_SCALE is increased, the gain is decreased!

There are also the coefficients x_SPEED_PI_PROPORTIONAL_GAIN_REAL (resp. x_SPEED_PI_INTEGRAL_TI_REAL), which are not directly used for regulator setting, but can be used to count `x_SPEED_PI_PROPORTIONAL_GAIN`, `x_SPEED_PI_PROPORTIONAL_GAIN_SCALE` (resp. `x_SPEED_PI_INTEGRAL_GAIN`, `x_SPEED_PI_INTEGRAL_GAIN_SCALE`) using the formulae in the comments!

11. References

11.1 Software Development Kit, SDK rev.2.3

- Targetting_DSP56805_Platform.pdf
  - located at: *Embedded SDK\help\docs\sdk\targets\Targetting_DSP56805_Platform\content*
- Targetting_DSP56803_Platform.pdf
  - located at: *Embedded SDK\help\docs\sdk\targets\Targetting_DSP56803_Platform\content*
- Motor Control.pdf, chapter BLDC motor commutation with Zero Crossing sensing
  - *located at: Embedded SDK\help\docs\sdk\libraries\motorcontrol\content*

11.2 User’s Manuals and Application Notes

- **AN1627, Low Cost High Efficiency Sensorless Drive for brushless DC Motor using MC68HC(7)05MC4**
- **56F800 16-bit Digital Signal Processor, Family Manual, DSP56F800FM, Freescale Semiconductor, Inc.**
- **56F80x 16-bit Digital Signal Processor, User’s Manual, 56F801-7UM, Freescale Semiconductor, Inc.**
- *web page: www.freescale.com*

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

AN1913
Rev. 3
11/2005