Freescale Semiconductor
Application Note

AN1914
Rev. 1, 11/2005

3-Phase BLDC Motor Control with Sensorless Back EMF Zero Crossing Detection Using 56F80x

Design of 3-Phase BLDC Motor Control Application Based on the Software Development Kit

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. 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 Back-EMF Zero Crossing technique for position detection. It serves as an example of a sensorless BLDC motor control system using a Digital Signal Controller (DSC) and 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 features, basic BLDC motor theory, system design concept, hardware implementation and software design including the PC master software visualization tool.

Today more and more variable speed drives are designed into appliance products to increase product performance and system efficiency. The low dynamic drive, whereby the load or speed is changed quite slowly in comparison with the system mechanical time constant, is a solution for many common appliance applications because simple algorithms can perform the control tasks. Moreover, the necessary computing power can be minimized by using dedicated on chip peripheral modules (such as A/D converter, dedicated PWM outputs, input capture and output compare functions).

Three phase Brushless DC (BLDC) motors are good candidates because of their high efficiency capability and easy to drive features. The disadvantage of this kind of motor is the fact that commutation of motor phases relies on its rotor position. Although the rotor position is usually sensed by sensors, there are applications that require sensorless control. Benefits of the sensorless solution are elimination of the position sensor and its connections between the control unit and the motor.

The sensorless rotor position technique detects the zero crossing points of Back-EMF induced in the motor windings. The phase Back-EMF Zero Crossing points are sensed while one of the three phase windings is not powered. The obtained information is processed in order to commutate energized phase pair and control the phase voltage, using Pulse Width Modulation.

This application note provides a fundamental mathematical method for modelling, torque calculation and control concept of the presented drive. The drive was developed in order to address simple applications (e.g. pumps, compressors, fans...) within certain ranges of speed and load. Results from simulation show the drive behavior at different working conditions and better explain the drive strategy.

2. DSC Advantages and Features

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

The 56F805 device provides the following peripheral blocks:

- Two Pulse Width Modulator modules (PWMA & PWMB), each with six PWM outputs, three Current Status inputs, and four Fault inputs, fault tolerant design with deadtime insertion, supports both Center- and Edge- aligned modes
- Two twelve-bit, Analog-to-Digital Convertors (ADCs) that support simultaneous conversions with dual 4-pin multiplexed inputs. ADC can be synchronized by PWM modules
- Two Quadrature Decoders (Quad Dec0 & Quad Dec1), each with four inputs, or, alternatively, two additional Quad Timers (A & B)
- Two dedicated General Purpose Quad Timers totalling 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 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
- External reset output pin for system reset
- JTAG/On-Chip Emulation (OnCE)
- 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 x 16-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, or six independent PWM signals
- Features of 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 one to 16
- Individual software-controlled PWM output
- Programmable fault protection
- Polarity control
- 20-mA current sink capability on each PWM pin
- 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) and 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 using the channel swap feature.

The Quadrature Timer feature set is as follows:

- Four channels, independently programmable as input capture or output compare
- Each channel has its own timebase source
- Each of four channels can use any of four timer inputs
- Rising edge, falling edge, or both edges input capture trigger
- Set, clear, or toggle output capture action
- Pulse Width Modulator (PWM) signal generation
- Programmable clock sources and frequencies, including external clock

The Quadrature Timer provides the capability to precisely control the key sensorless BLDC events by providing the time base for zero crossing events and the output compare function for scheduling the commutation events.

Dual Analog-to-Digital Converter (ADC) modules—four inputs on each has the following feature set:

- Eight total analog inputs
- 12-bit range
- Monotonic over entire range with no missing codes
- First channel on each ADC can be swapped with the alternate ADC
- Can perform two simultaneous analog-to-digital conversions
- Conversion time = 1.25 us
- Contains programmable zero offset register
- Generates interrupt on completion of conversion
- Optional conversion interrupt is asserted when the analog voltage level exceeds, or
- falls below, the value contained in the zero offset register
- Output is in two's complement or unsigned formats

The Analog-to-Digital Converter is utilized to measure DC-bus voltage, DC-Bus current and the power module temperature. Its 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 Motor Targeted by This Application

The Brushless DC motor (BLDC motor) 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 magnetic circuit is usually made from magnetic steel sheets. The stator phase windings are inserted in the slots (distributed winding) as shown in **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 are chosen such a way that the Back-EMF (the voltage induced into the stator winding due to rotor movement) shape is trapezoidal. This allows the three phase voltage system (see **Figure 3-2**), with a rectangular shape, to be used to create a rotational field with low torque ripples.

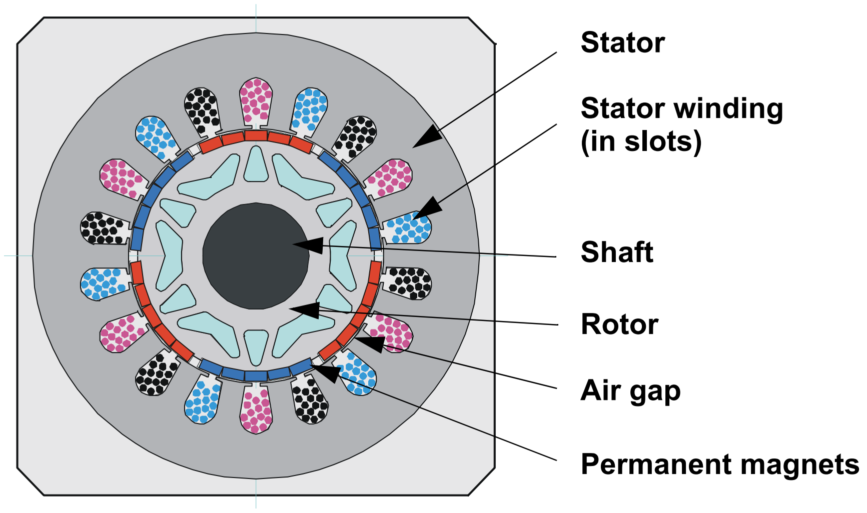


Figure 3-1. BLDC Motor - Cross Section

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

The rectangular, easy to create, shape of applied voltage ensures the simplicity of control and drive. But 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. In this condition the motor behaves as a DC motor and runs at the best working point. Thus simplicity of control and good performance make this motor a natural choice for low-cost and high-efficiency applications.

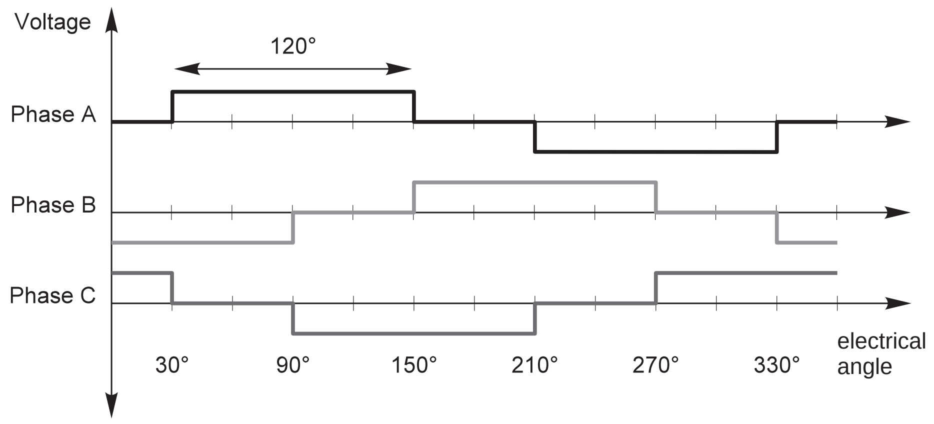


Figure 3-2. Three Phase Voltage System

Figure 3-3 shows number of waveforms: the magnetic flux linkage, the phase Back-EMF voltage and the phase-to-phase Back-EMF voltage. The magnetic flux linkage can be measured; however in this case it was calculated by integrating the 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 amplitude is changed 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 occur. 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. In normal operation the commutation is performed here. Some control techniques advance the commutation by a defined angle in order to control the drive above the PWM voltage control.

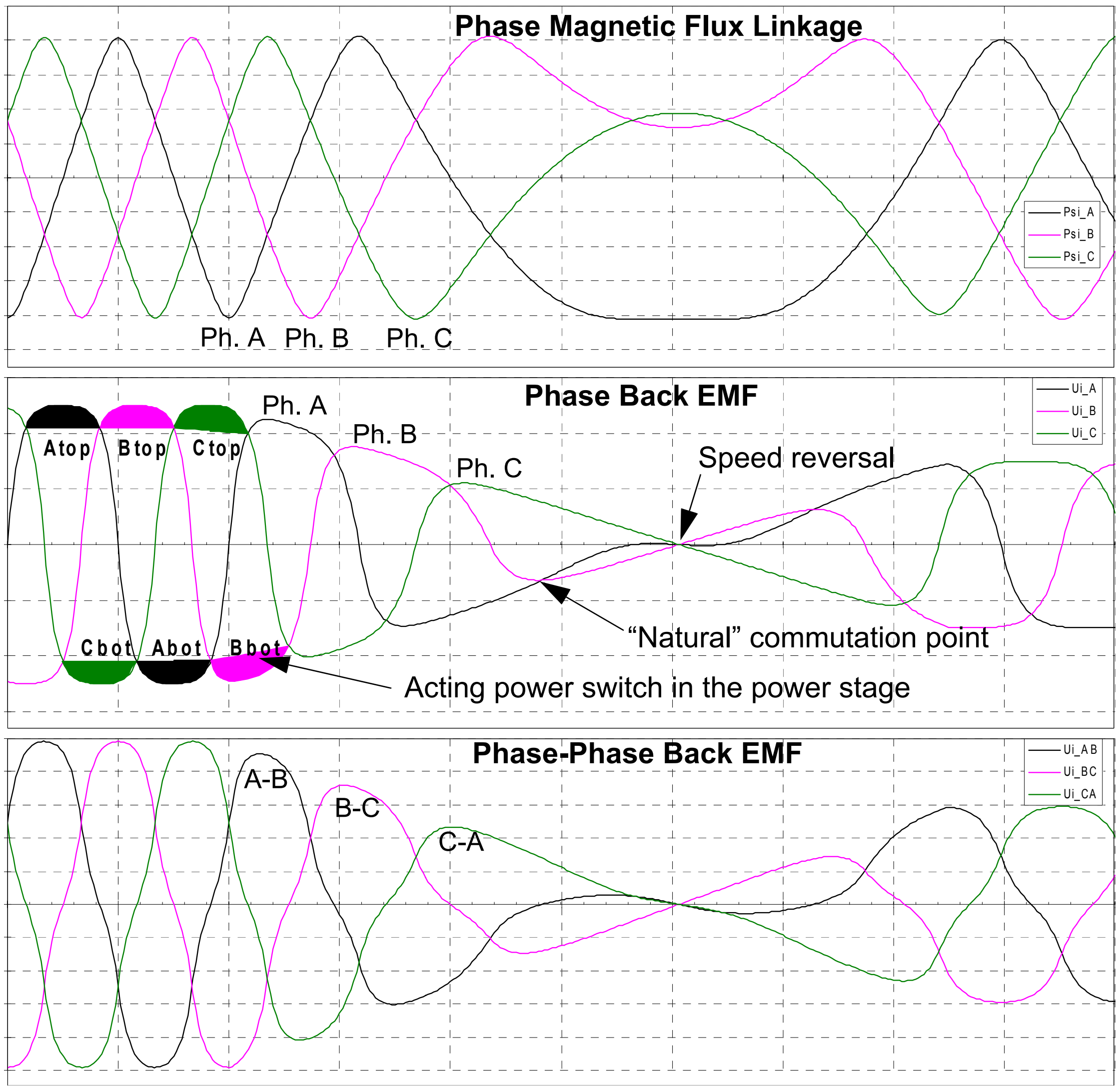


Figure 3-3. BLDC Motor - Back EMF and Magnetic Flux

3.2 3-Phase BLDC Power Stage

The voltage for 3-phase BLDC motor is provided by a 3-phase power stage controlled by a DSC. The PWM module is usually implemented on a DSC to create desired control signals.

A device with BLDC motor and power stage is shown in **Figure 3-3**.

3.3 Why Sensorless Control?

As explained in the previous section, the rotor position must be known in order to drive a Brushless DC motor. If any sensors are used to detect rotor position, then sensed information must be transferred to a control unit (see **Figure 3-4**). Therefore additional connections to the motor are necessary. This may not be acceptable for some applications. There are at least two reasons why you might want to eliminate the position sensors:

- Inability to make additional connections between position sensors and the control unit
- Cost of the position sensors and wiring

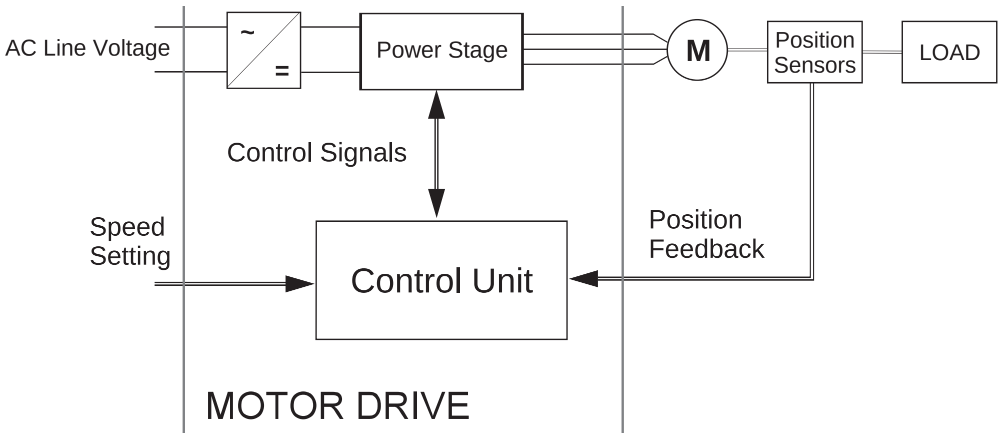


Figure 3-4. Classical System

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**) has been created.

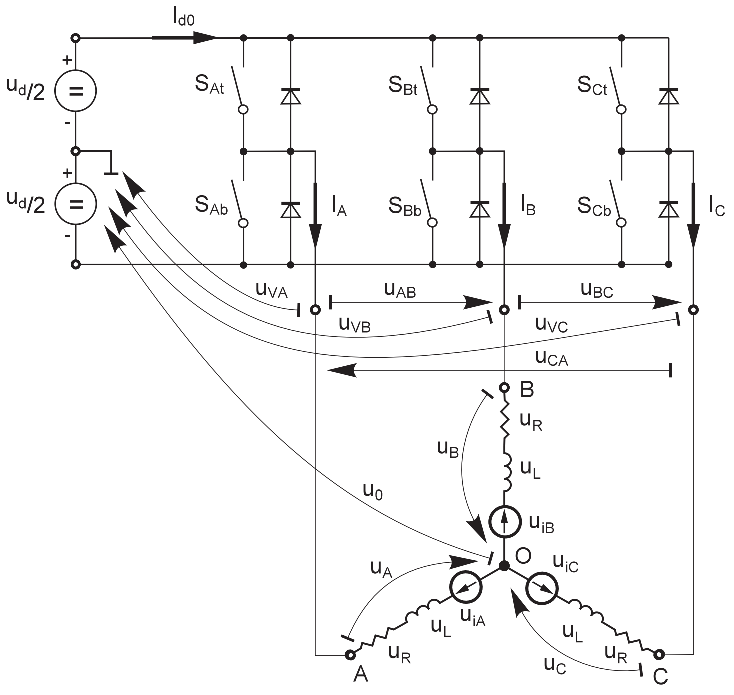


Figure 3-5. Power Stage - Motor Topology

The second 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 normal three 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 put 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; the voltages between one power stage output and its virtual zero.
$u_A\ldots u_C$	are motor phase winding voltages.
$u_{iA}\ldots u_{iC}$	are phase Back-EMF voltages induced in the stator winding.
u_C	is the voltage between the central point of the star of motor winding and the power stage natural zero
$i_A\ldots i_C$	are phase currents

The equations (EQ 3-1.) can be written 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 DC Brushless motor are connected at a time (see **Figure 3-2**), so the third phase 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 is going through this phase. This is described by the following conditions:

$$\begin{gathered} S_{Ab}, S_{Bt} \leftarrow are\ energized \\ u_{VA} = \mp\frac{1}{2}u_d,\ u_{VB} = \pm\frac{1}{2}u_d \\ i_A = -i_B,\ i_C = 0,\ di_C = 0 \\ u_{iA} + u_{iB} + u_{iC} = 0 \end{gathered} \qquad \text{(EQ 3-1.)}$$

The branch voltage C can be calculated when considering the above conditions:

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

As shown in **Figure 3-5**, the branch voltage of phase C can be sensed between the power stage output C and the zero voltage level. Thus the Back-EMF voltage is obtained and the zero crossing can be recognized.

The general expressions can also be found:

$$u_{Vx} = \frac{3}{2}u_{ix} where\ x = A,B,C \qquad \text{(EQ 3-3.)}$$

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 3-3.). In other words, the Back-EMF voltage can be sensed during designated intervals.

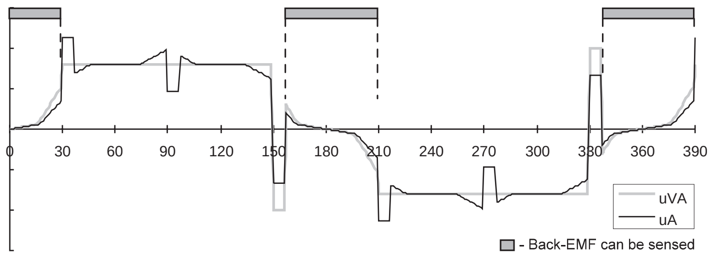


Figure 3-6. Phase Voltage Waveform

3.6 Back-EMF Sensing Circuit

An example of the possible implementation of the Back-EMF sensing circuit is shown in **Figure 3-7**.

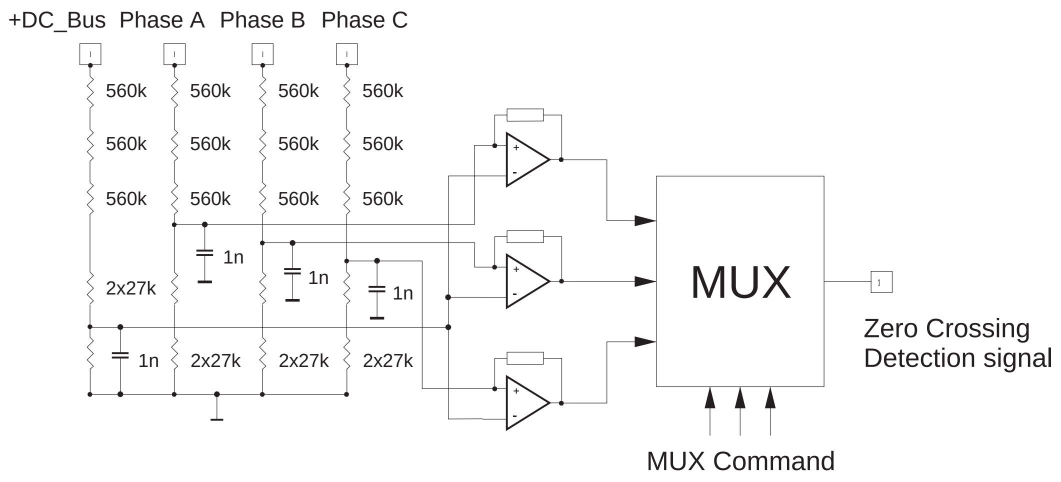


Figure 3-7. Back-EMF Sensing Circuit Diagram

As explained in the theoretical part of this application note, the phase zero crossing event can be detected at the moment when the branch voltage (of a free phase) crosses the half DC-bus voltage level. The resistor network is used to divide sensed voltages down to a 0-15V voltage level. The comparators sense the zero voltage difference of the input signal. The multiple resistors reduce the voltage across each resistor component to an acceptable level. A simple RC filter prevents the comparators from being disturbed by high voltage spikes produced by IGBT switching. The MUX selects the phase comparator output, which corresponds to the current commutation stage. This Zero Crossing Detection signal is transferred to the timer input pin.

The comparator control and zero crossing signals plus the voltage waveforms are shown in **Figure 3-8**.

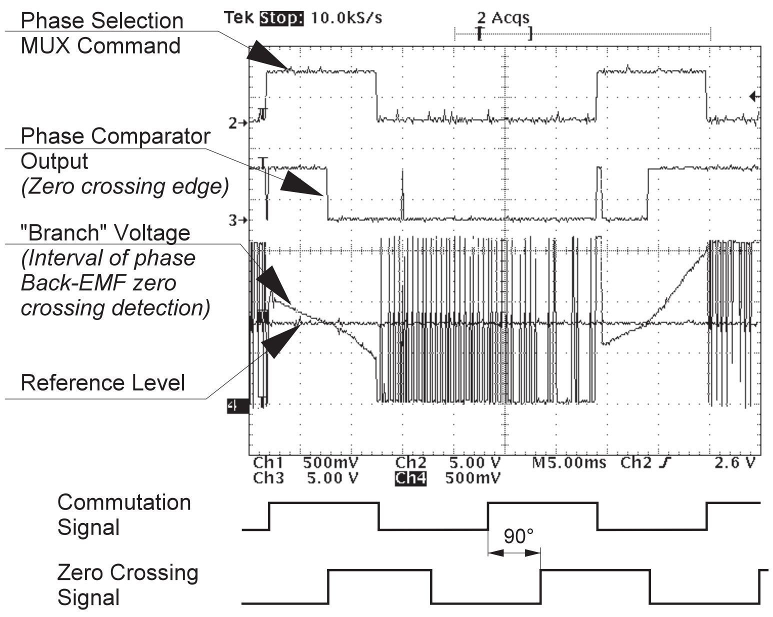


Figure 3-8. The Zero Crossing Detection

The voltage drop resistor is used to measure the DC-bus current which is chopped by the PWM. The obtained signal is rectified and amplified (0-3.3V with 1.65V offset). The internal controller's A/D converter and Zero Crossing detection are 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 switch. PWM technique is used to the control motor phase voltage.

4. System Design Concept

4.1 System Specification

The system was designed to meet the following performance specifications:

- Control technique incorporates
  - sensorless BEMF Zero Crossing commutation control
  - closed loop without current loop
  - bi-directional rotation
  - motoring mode
- Targeted for 56F803/805EVM 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 BEMF Zero Crossing 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 concept below was chosen. The sensorless rotor position technique developed detects the zero crossing points of Back-EMF induced in the motor windings. The phase Back-EMF Zero Crossing points are sensed while one of the three phase windings is not powered. The obtained information is processed in order to commutate the energized phase pair and control the phase voltage, using Pulse Width Modulation.

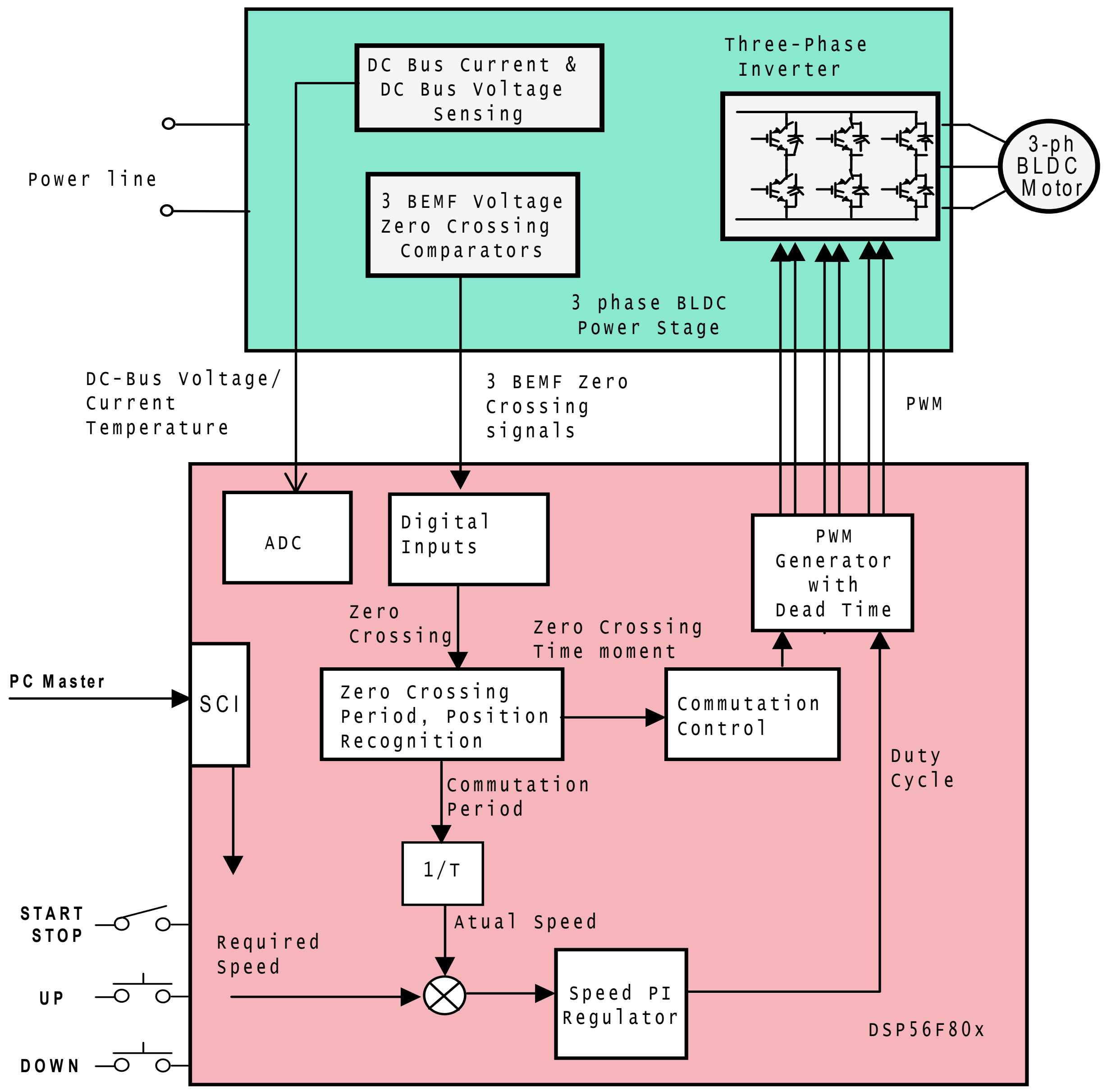

DC Bus Current & DC Bus Voltage Sensing
Three-Phase Inverter
3-ph BLDC Motor
Power line
3 BEMF Voltage Zero Crossing Comparators
3 phase BLDC Power Stage
DC-Bus Voltage/ Current Temperature
3 BEMF Zero Crossing signals
PWM
ADC
Digital Inputs
PWM Generator with Dead Time
Zero Crossing
Zero Crossing Time moment
PC Master
SCI
Zero Crossing Period, Position Recognition
Commutation Control
Duty Cycle
Commutation Period
1/T
START STOP
Atual Speed
Required Speed
UP
Speed PI Regulator
DSP56F80x
DOWN

Figure 4-1. System Concept

The Back-EMF zero crossing detection enables position recognition. 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 SW in order to filter high voltage spikes produced by the switching of the IGBTs (MOSFETs). This signal is transferred to the device's Encoder Input which is also used as a digital filter. The SW selects one of the phase comparator outputs which corresponds to the current commutation step.

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
- Sensorless Commutation Control
- Speed Control

The implementation of the control technique with all the SW processes is shown in Flow Chart, State diagrams and Data Flow (see **Figure 7-2** through **Figure 7-8**).

5.2 PWM voltage Generation for BLDC

As was already explained, the three phase voltage system shown in **Figure 3-2** needs to be created to run the BLDC motor. It is provided by 3-phase power stage with 6 IGBTs (MOSFET) controlled by the on-chip PWM module (see **Figure 5-1**). The PWM signals with state currents are shown in **Figure 5-2** and **Figure 5-3**.

Figure 5-2 shows that both Bottom and Top power switches of the “free“ phase must be switched off. This is needed for any effective control of Brushless DC motor with trapezoidal BEMF.

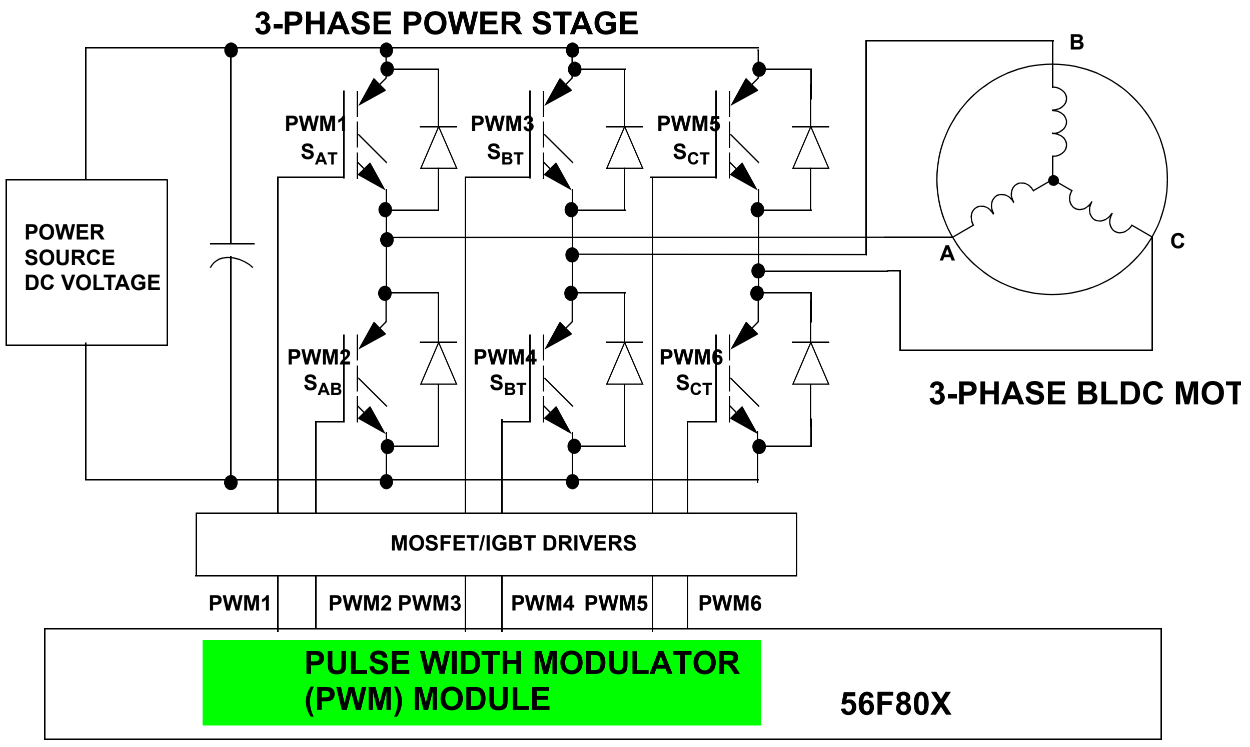


Figure 5-1. PWM with BLDC Power Stage

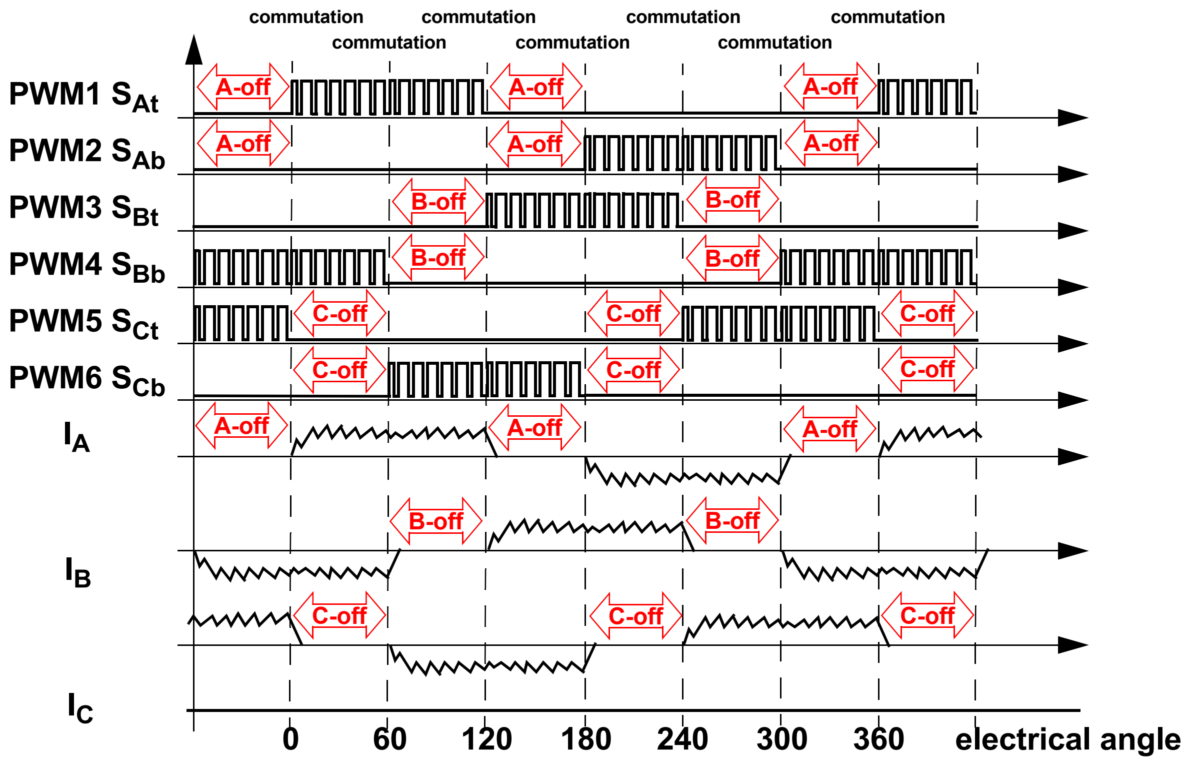


Figure 5-2. 3-phase BLDC Motor Commutation PWM Signal

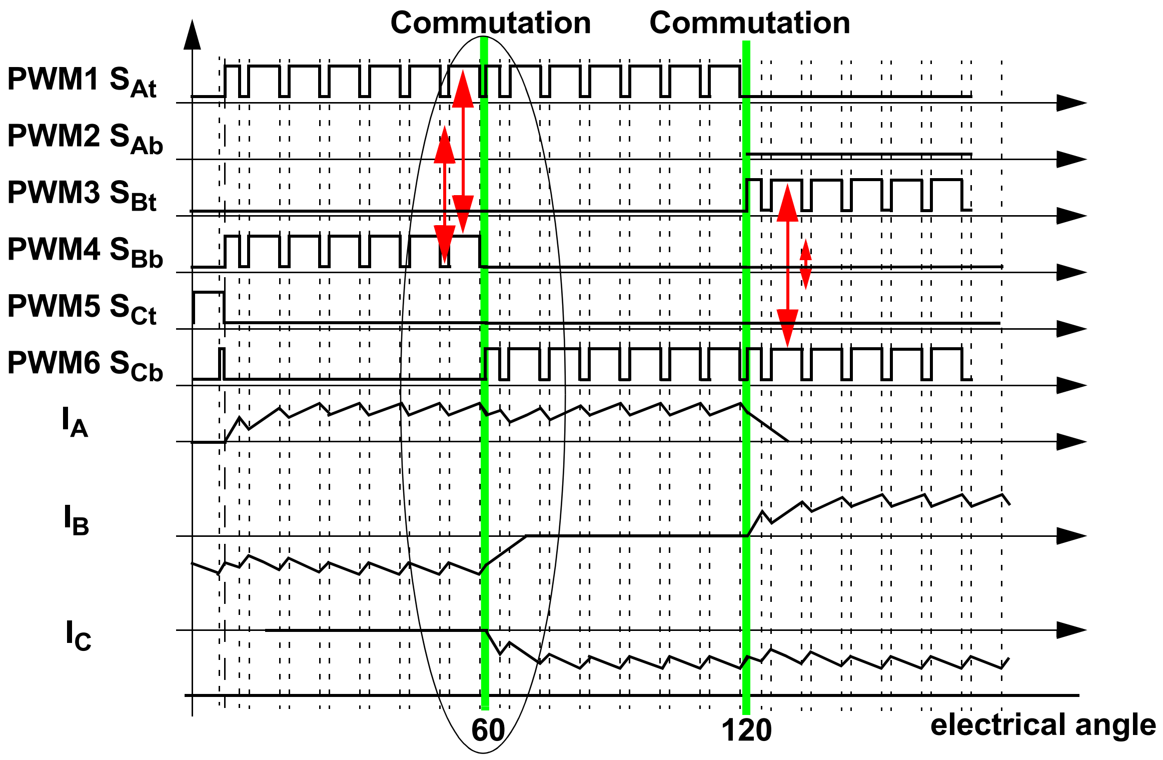


Figure 5-3. BLDC Commutation with Bipolar (Hard) Switching

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 connected coils is always ±DC bus voltage whenever there is a current flowing through these coils. Thus the condition for successful BEMF Zero Crossing sensing is fulfilled as described in Section 3.

5.3 BEMF Zero Crossing Sensing

5.3.1 BEMF Zero Crossing Checking

The BEMF Zero Crossing of the 3 phases is checked using hardware comparators as described in Section 3. The outputs of the comparators are led to Quadrature Decoder Inputs. Where the digital filtration block is used to filter the spike on the Zero Crossing signals.

The software selects the “free” phase at each commutation step and reads the filtered signal to detect the BEMF Zero Crossing event.

5.3.2 BEMF Zero Crossing Synchronization with PWM

The power stage PWM switching causes the high voltage transient of the phase voltages. This transient is passed to “free” phase due to mutual capacitor between the motor windings coupling. Figure 5-4 shows that free phase “branch” voltage U_{va} is disturbed by PWM voltage shown on phase “branch” voltage U_{vb}.

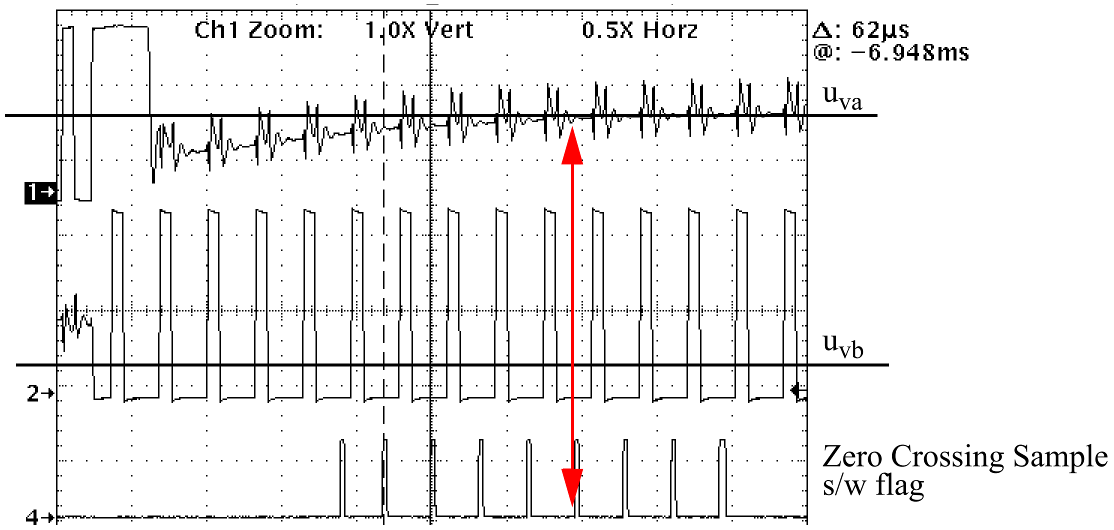


Figure 5-4. BEMF Zero Crossing Synchronization with PWM

The non-fed phase “branch” voltage U_{va} is disturbed at the PWM edges. Therefore the presented BLDC Motor Control application synchronizes the BEMF Zero Crossing detection with PWM.

5.4 Sensorless Commutation Control

This chapter concentrates on sensorless BLDC motor commutation with BEMF 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; then the rotation is started without the position feedback. When the rotor moves, the Back-EMF is acquired so the position is known and can be used to calculate the speed and processing of the commutation in the Running state.

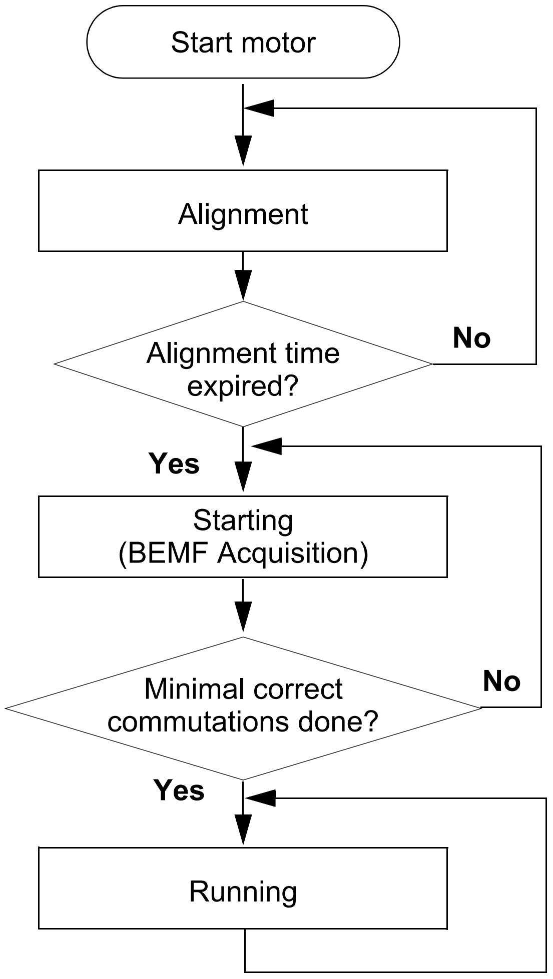

Start motor
Alignment
Alignment time expired?
No
Yes
Starting (BEMF Acquisition)
Minimal correct commutations done?
No
Yes
Running

Figure 5-5. Commutation Control Stages

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 stabilized 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. When the preset time-out expires then this state is finished.

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

The current PI controller works with constant execution (sampling) period determined by PWM frequency: Current Controller period = 1/PWM frequency.

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

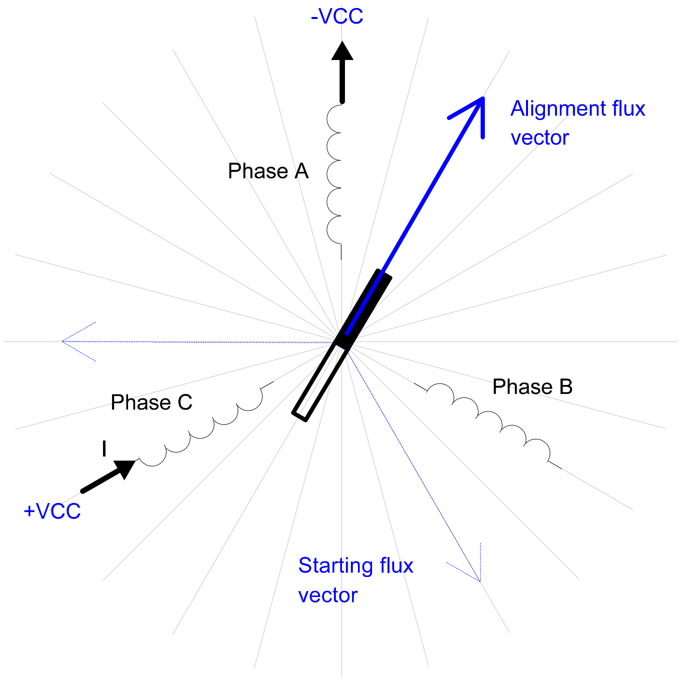


Figure 5-6. Alignment

5.4.2 Running

The commutation process is the series of states which assure that the Back-EMF zero crossing is successfully captured, the new commutation time is calculated and, finally, the commutation is performed. The following processes needs 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 **Motor Control.pdf, chapter BLDC Motor Commutation with Zero Crossing Sensing** (see [12.1]).

From pictures an overview of how the commutation works can be understood. After commuting the motor phases there is a time interval (Per_Toff[n]) when the shape of Back-EMF must stabilized (after the commutation the fly-back diodes are conducting the decaying phase current, therefore sensing of the Back-EMF is not possible). Then the new commutation time (T2[n]) is preset. The new commutation will be 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 actions are taken in order to return to the regular states.

The flow chart explaining the principle of BLDC CommutationControl with BEMF Zero Crossing Sensing is shown in **Figure 5-7**.

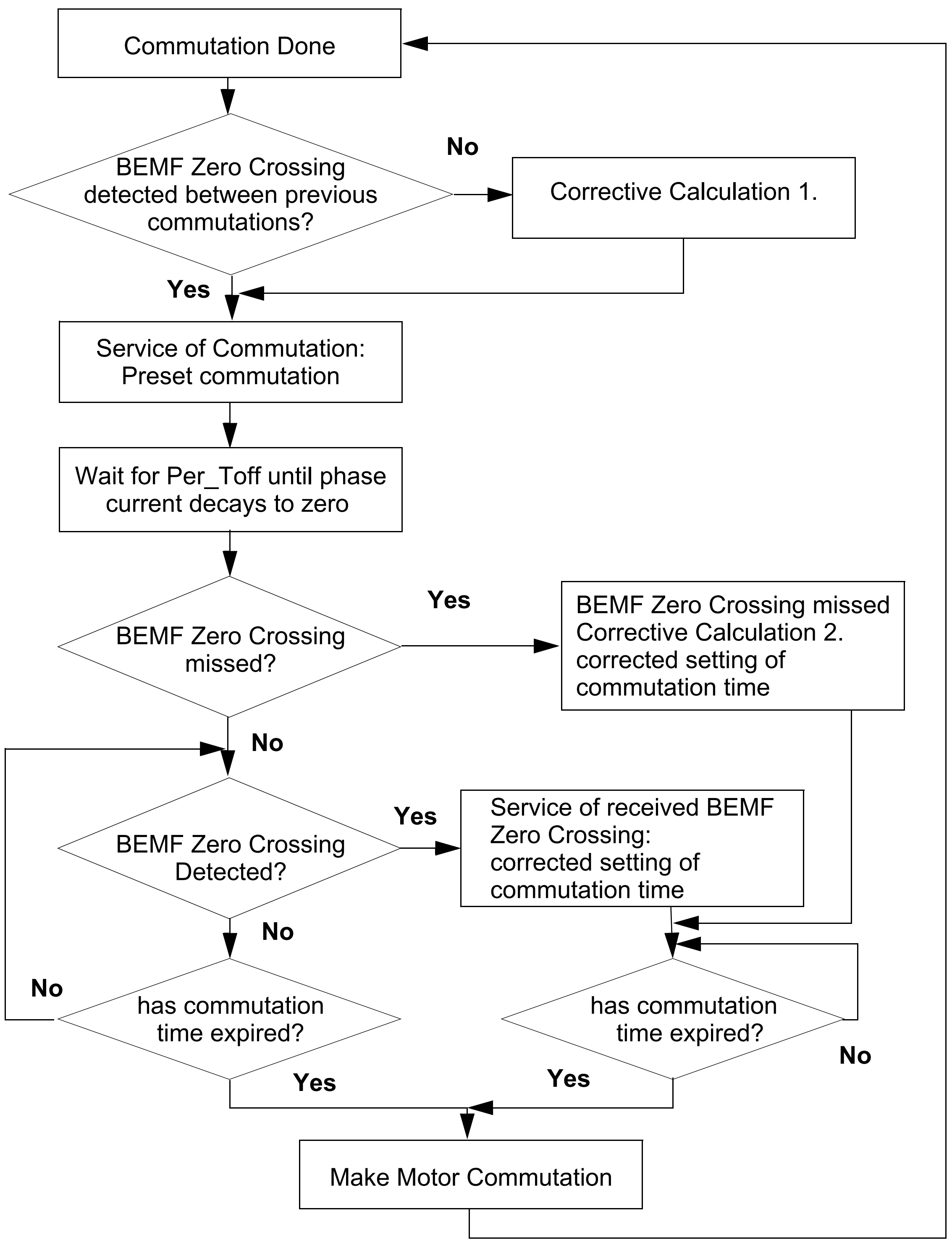
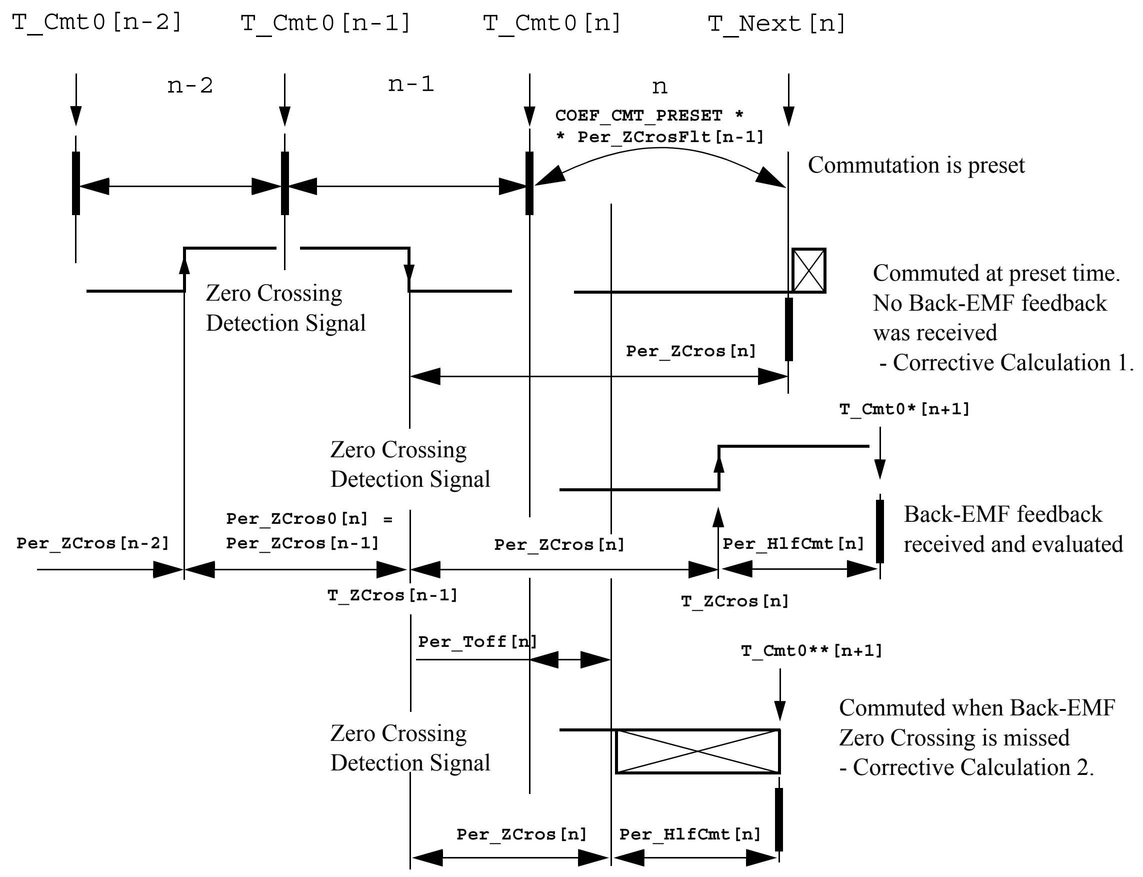

Commutation Done
BEMF Zero Crossing detected between previous commutations?
No
Corrective Calculation 1.
Yes
Service of Commutation: Preset commutation
Wait for Per_Toff until phase current decays to zero
BEMF Zero Crossing missed?
Yes
BEMF Zero Crossing missed Corrective Calculation 2. corrected setting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

5.4.2.2 Running - Commutation Times Calculation

Commutation time calculation is provided by algorithm **bldcZCComput** described in **Motor Control.pdf, chapter BLDC Motor Commutation with Zero Crossing Sensing** (see [12.1]).



T_Cmt0[n-2]
T_Cmt0[n-1]
T_Cmt0[n]
T_Next[n]
n-2
n-1
n
COEF_CMT_PRESET *
* Per_ZCrosFlt[n-1]
Commutation is preset
Zero Crossing Detection Signal
Commuted at preset time. No Back-EMF feedback was received - Corrective Calculation 1.
Per_ZCros[n]
T_Cmt0*[n+1]
Zero Crossing Detection Signal
Back-EMF feedback received and evaluated
Per_ZCros[n-2]
Per_ZCros0[n] = Per_ZCros[n-1]
Per_ZCros[n]
Per_HlfCmt[n]
T_ZCros[n-1]
T_ZCros[n]
Per_Toff[n]
T_Cmt0**[n+1]
Zero Crossing Detection Signal
Commuted when Back-EMF Zero Crossing is missed - Corrective Calculation 2.
Per_ZCros[n]
Per_HlfCmt[n]

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 Stage**:

- **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 Stage!
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 Stage!
Per_Toff[n+1] = Per_ZCrosFlt*Coef_Toff and Max_PerCmtProc minimum
          Coef_Toff = 0.35 at Running Stage, 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 Stage!
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 Stage!
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 these 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 90deg±30deg.

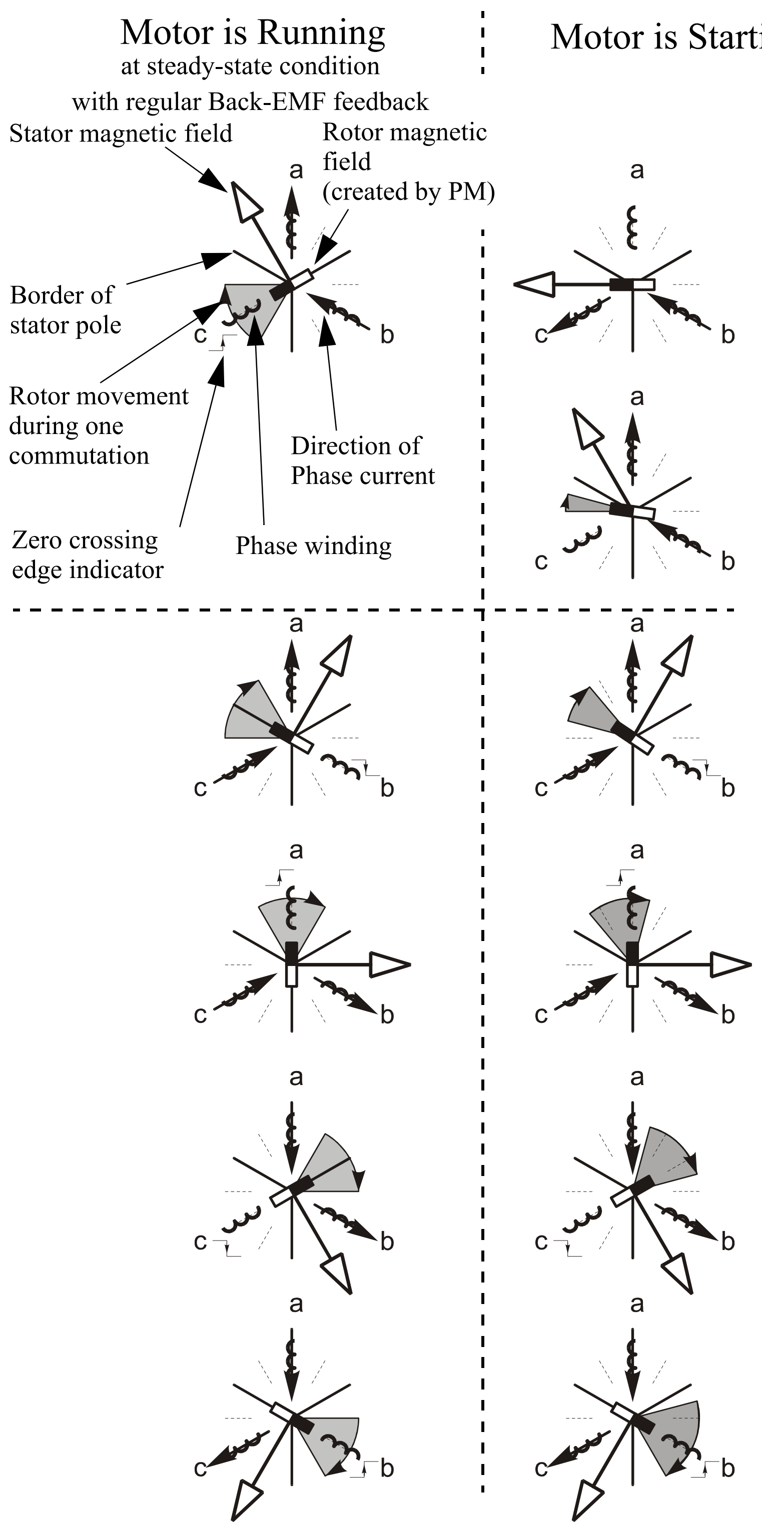
The first condition is satisfied during the Alignment state by keeping the DC Bus current on the level which is sufficient to start the motor. In the Starting (Back-EMF Acquisition) state the same value of PWM duty cycle is used as the one which has stabilized the DC-Bus current during the Align state.

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

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

This allows starting the motor such that minimal speed (defined by state when Back-EMF can be sensed) is achieved during several commutations while producing the required torque. Until the Back-EMF feedback is locked 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 and 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 Stage

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

Starting (BEMF Acquisition)

The two fast (faster then the rotor can move) commutations 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 events 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

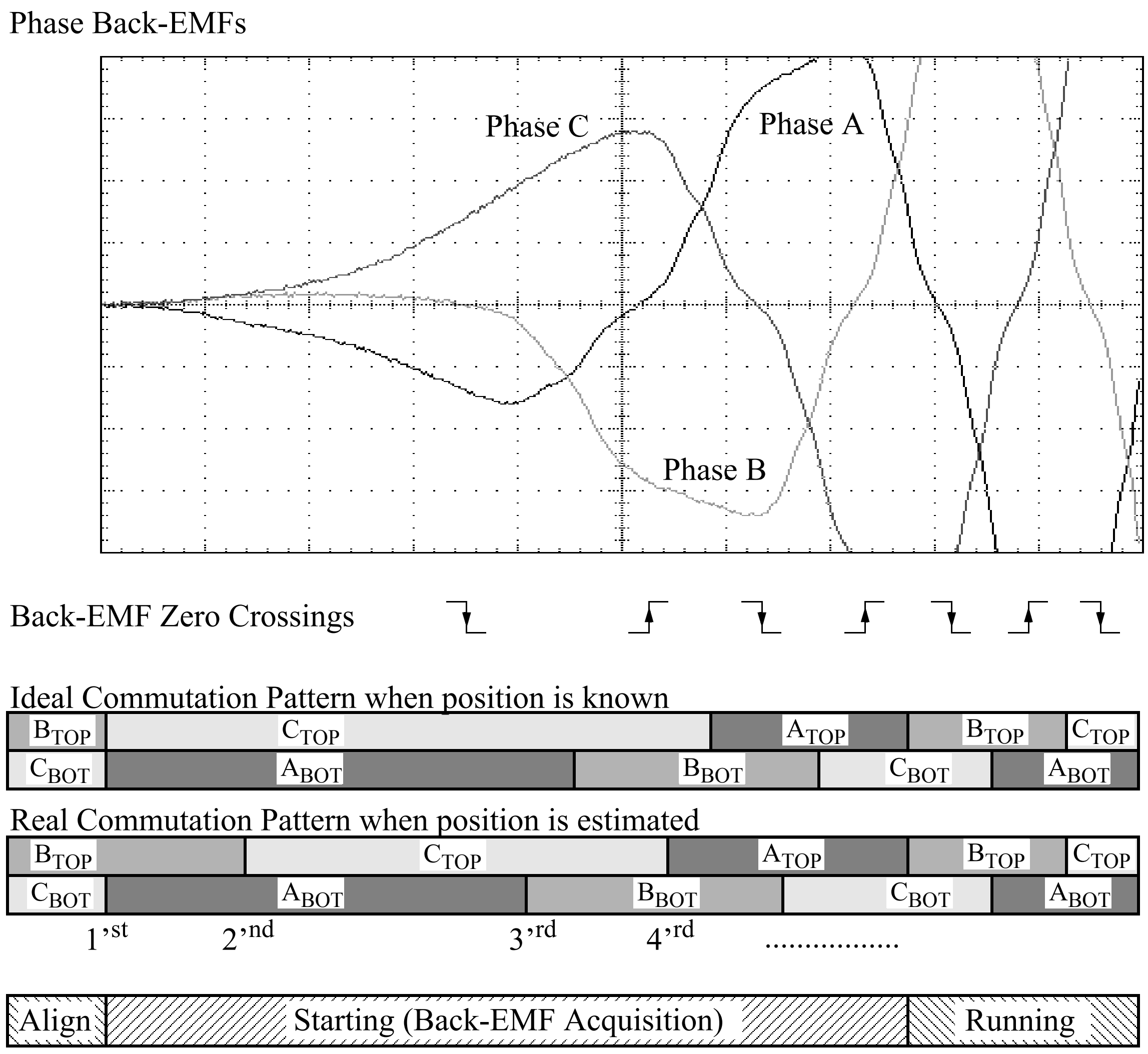


Figure 5-10. Back-EMF at Start-Up

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) Stage. The commutation times T2[1] and T2[2] are calculated without any influence of Back-EMF feedback.

.

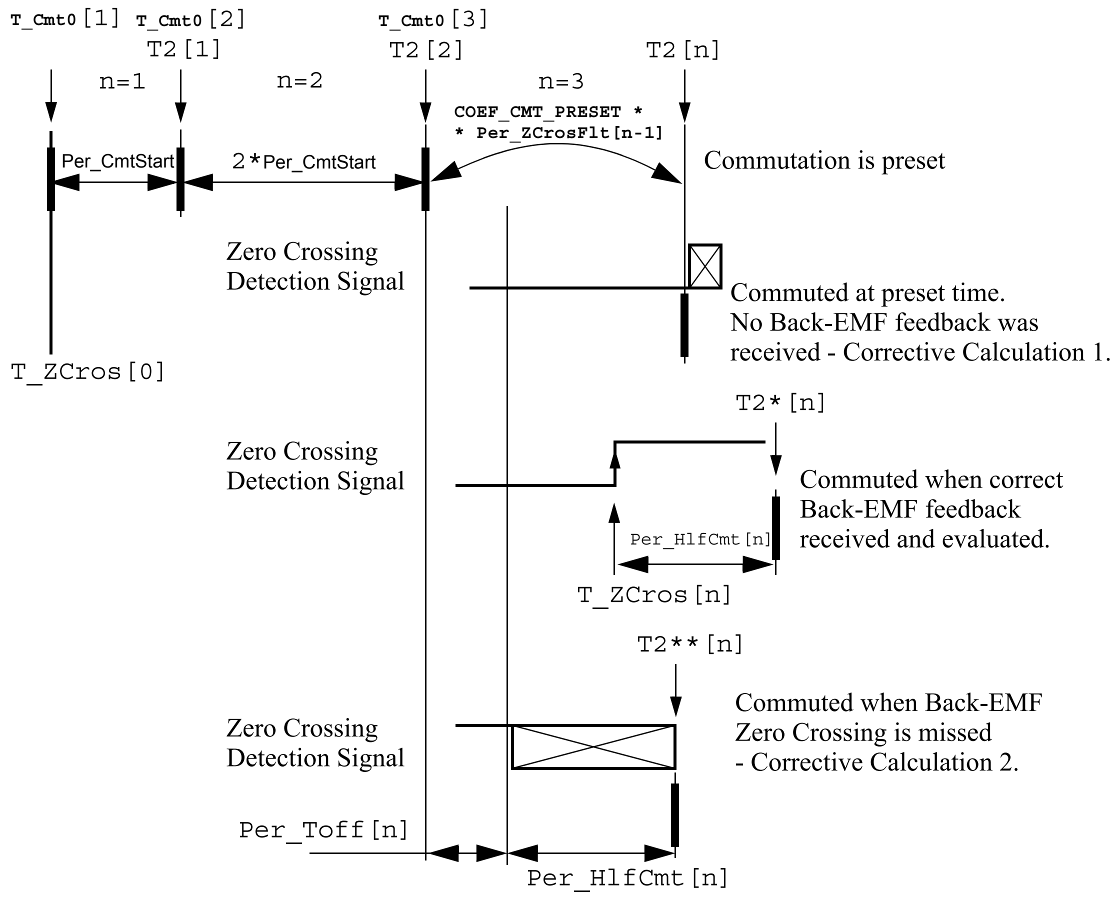

T_Cmt0 [1]
T_Cmt0 [2]
T_Cmt0 [3]
T2 [1]
T2 [2]
T2 [n]
n=1
n=2
n=3
COEF_CMT_PRESET *
* Per_ZCrosFlt [n-1]
Per_CmtStart
2 * Per_CmtStart
Commutation is preset
Zero Crossing Detection Signal
Commuted at preset time. No Back-EMF feedback was received - Corrective Calculation 1.
T_ZCros [0]
T2* [n]
Zero Crossing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

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

5.4.3.1 Starting - Commutation Times Calculation

The calculations made during Starting (Back-EMF Acquisition) Stage can be seen in **Motor Control.pdf, chapter BLDC Motor Commutation with Zero Crossing Sensing** (see Section 12.2).

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

```
coefficient Coef_CmtPrecomp = 2 at Starting Stage!
coefficient Coef_HlfCmt = 0.125 with advanced angle Advance_angle: 60Deg*3/8 = 22.5Deg
        at Starting Stage!
Coef_Toff = 0.5 at Running Stage, Max_PerCmtProc = 100!
```

5.5 Speed Control

The speed close loop control is provided by a well known PI regulator as shown in **Section 7.2.4**. The actual speed (Omega_Actual) is computed from average of two BEMF Zero Crossing periods (time intervals) received from the sensorless commutation control block.

The speed controller works with 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.

There are more software versions targeted for a specific device and Evaluation Module:

- 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 software program runs on each of three hardware and motor platforms without any change of parameters:

- 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 information can also be found in **Section 12.1**.

Notes: The detailed description of individual boards can be found in comprehensive user’s manuals belonging to each board. The user’s 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**.

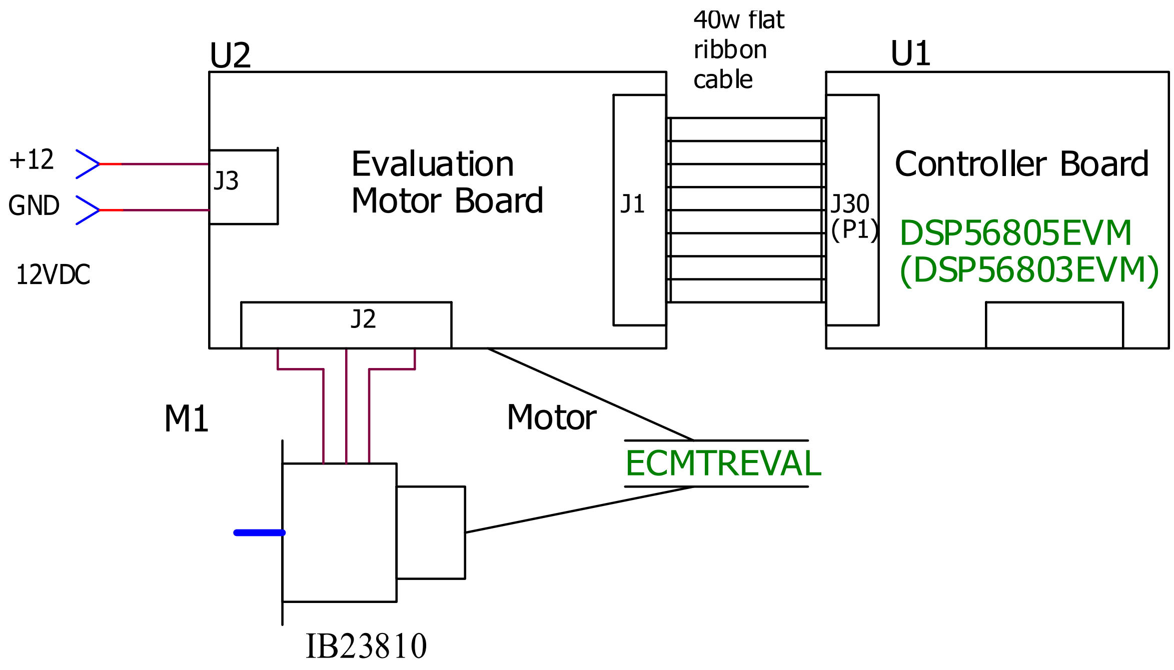


Figure 6-1. Low Voltage Evaluation Motor Hardware System Configuration

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: **Evaluation Motor Board User's Manual**
- U1 CONTROLLER BOARD for 56F805:
  - supplied as: 56F805EVM
  - described in: **56F805 Evaluation Module Hardware User's Manual**
- or U1 CONTROLLER BOARD for 56F803:
  - supplied as: 56F803EVM
  - described in: **56F803 Evaluation Module Hardware User's Manual**

Information of all above mentioned boards and documents can be found on:
http://mot-sps.com/motor/devtools/index.html

6.3 Low Voltage Hardware Set

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

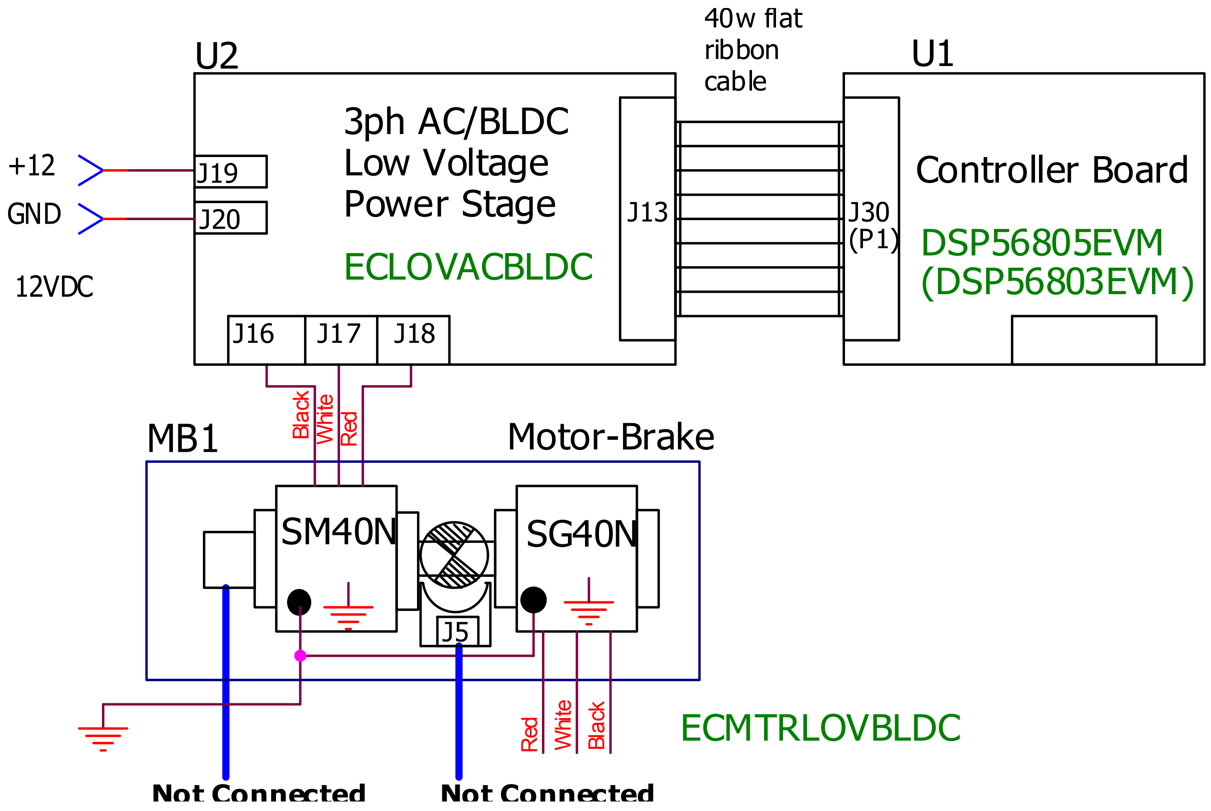


Figure 6-2. Low Voltage Hardware System Configuration

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

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

Information of all above mentioned boards and documents can be found on:
http://mot-sps.com/motor/devtools/index.html

6.4 High Voltage Hardware Set

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

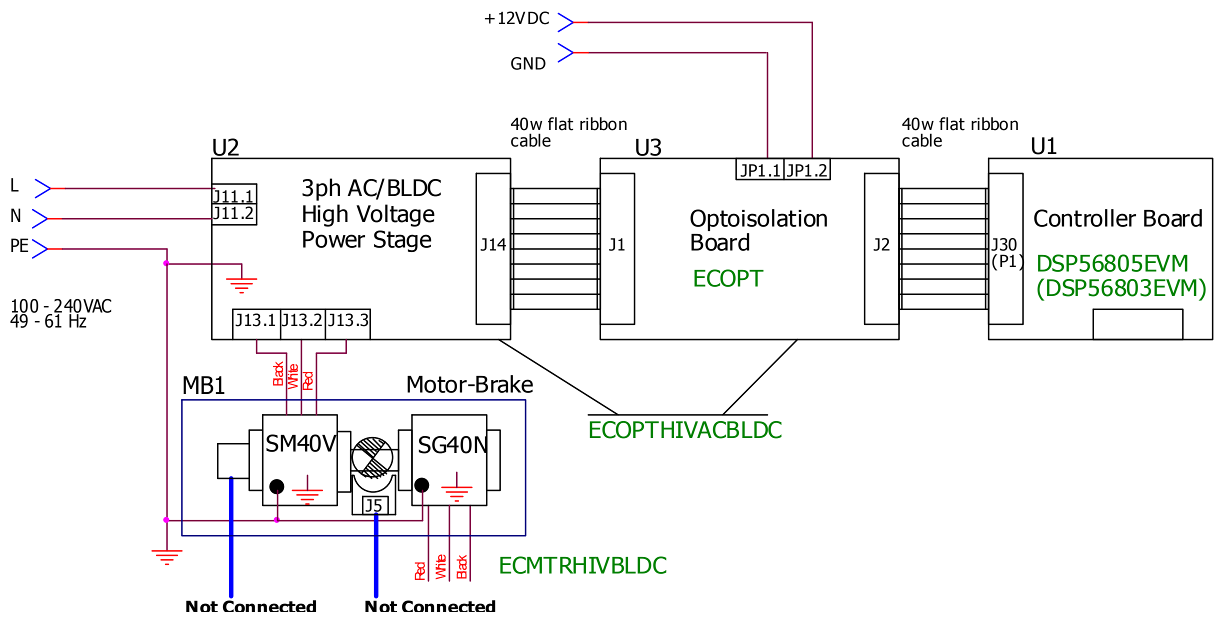


Figure 6-3. High Voltage Hardware System Configuration

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

- U1 - Controller Board for 56F805:
  - supplied as: 56F805EVM
  - described in: **Evaluation Module Hardware User's Manual**
- or U1 - Controller Board for 56F803:
  - supplied as: 56F803EVM
  - described in: **56F803 Evaluation Module Hardware User's Manual**
- U2 - 3 ph AC/BLDC High Voltage Power Stage
  - supplied in kit with Optoisolation Board as: ECOPTHIVACBLDC
  - described in: **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: **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 for all boards and documents can be found at:
www.freescale.com

7. SW Design

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

- Main Software Flow Chart
- Data Flow
- State Diagram

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

7.1 Main SW 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 SW level which precedes settings for other software levels, BLDC motor Commutation Control, Speed Control, Alignment Current Control, etc. The inputs of Application State Machine are Run/Stop Switch state, Required Speed Omega and Drive Fault Status. Required Mechanical Speed can be set from PC master software or manually with Up/Down buttons.

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

The Speed Control is detailed description is in sections **Section 7.2.3** and **Section 7.3.5**. Alignment Current Control is described in **Section 7.2.4** and **Section 7.3.6**.

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, overcurrent and overvoltage PWM fault handler.

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

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

The PWM Reload ISR is used to evaluate BEMF Zero Crossing, start ADC conversion and memorize Zero Crossing sampling time T_ZCSample.

The ADC Complete ISR is used to read voltages, current and temperature samples from the ADC. It also sets Current control and when the Current Control setting is enabled.

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

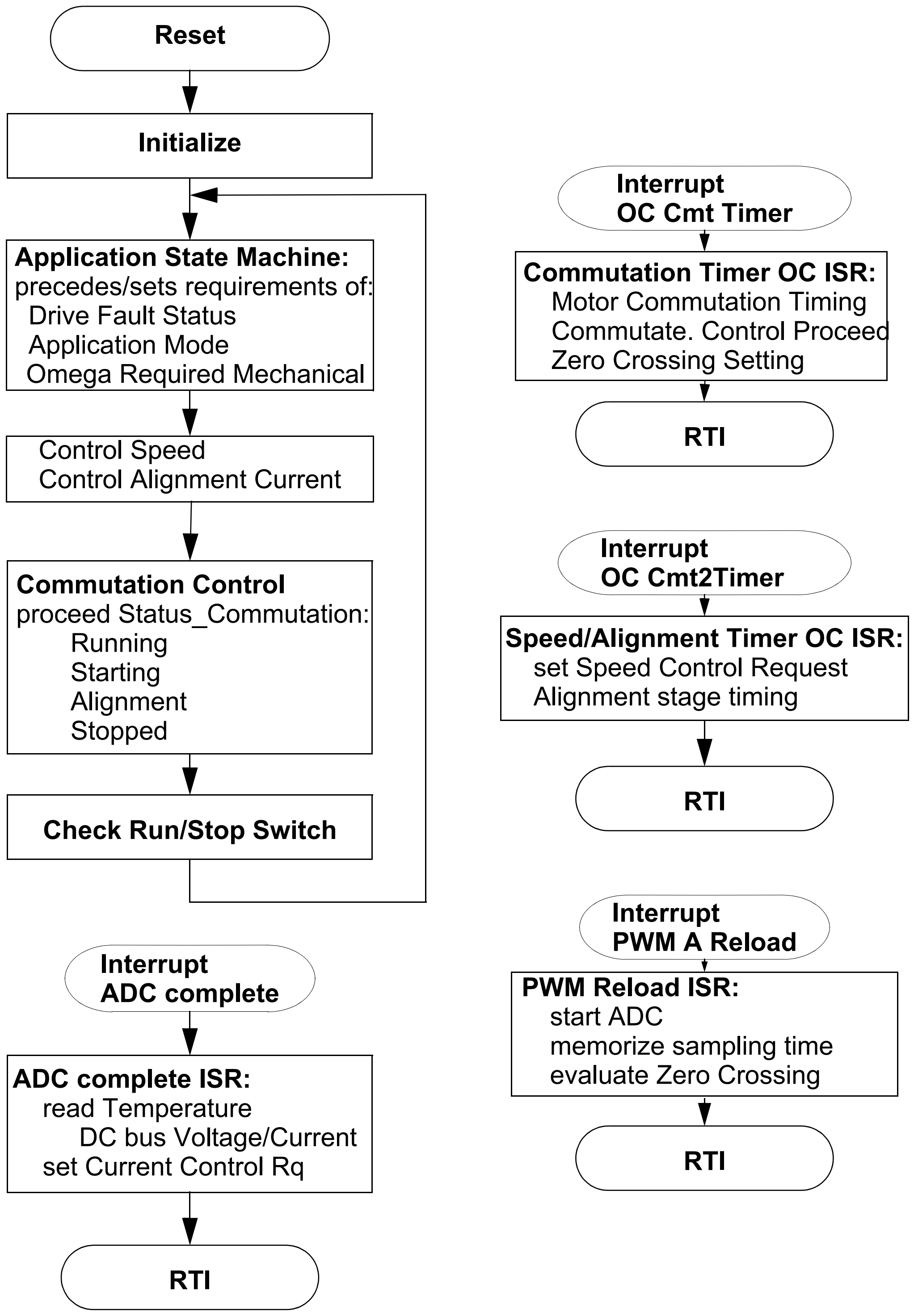


Figure 7-1. Main Software Flow Chart - Part 1

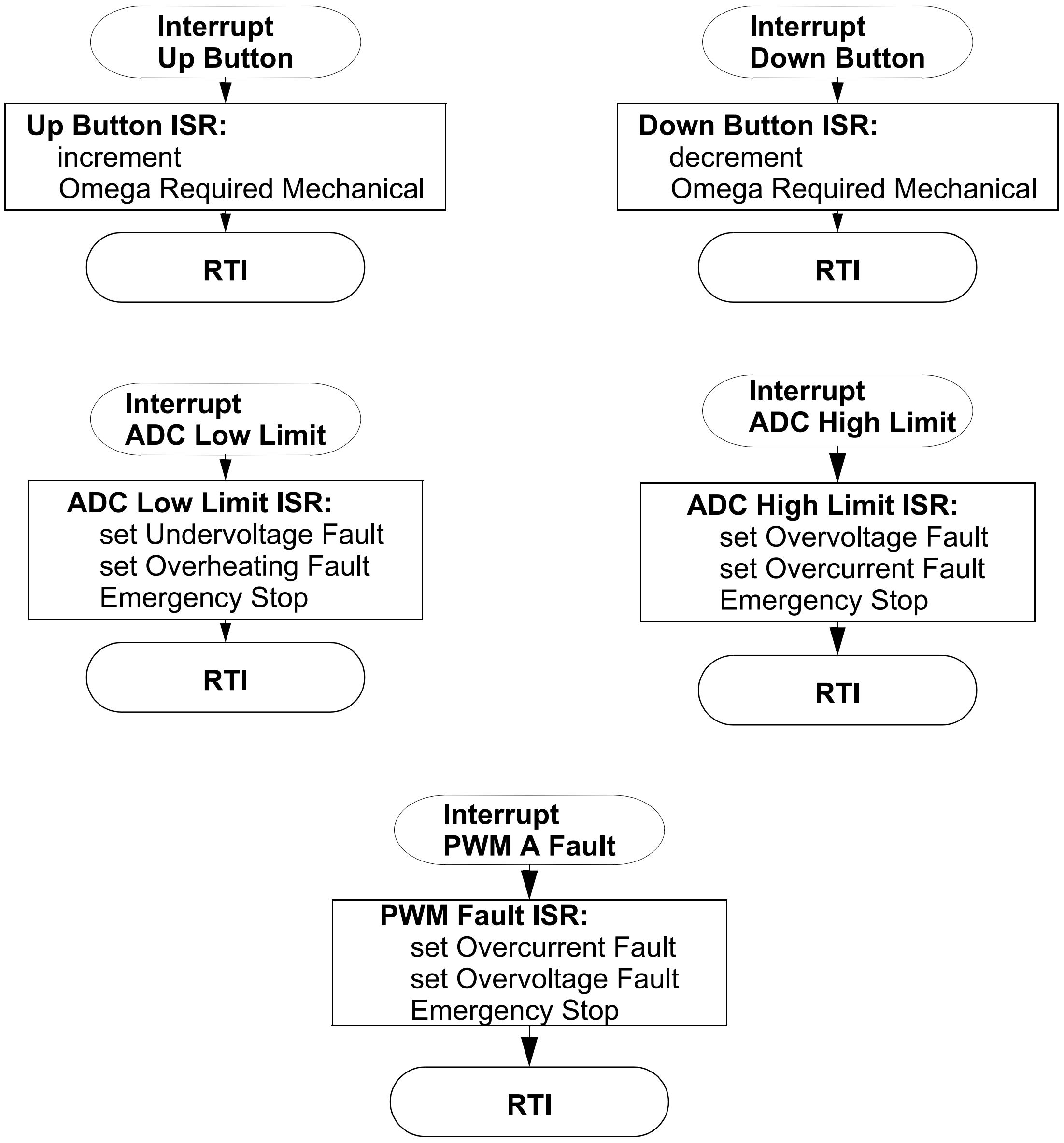


Figure 7-2. Main Software Flow Chart - Part 2

7.2 Data Flow

The control algorithm obtains values from the user interface and sensors, processes them and generates 3-phase PWM signals for motor control as can be seen on the data flow analysis shown in Figure 7-3.

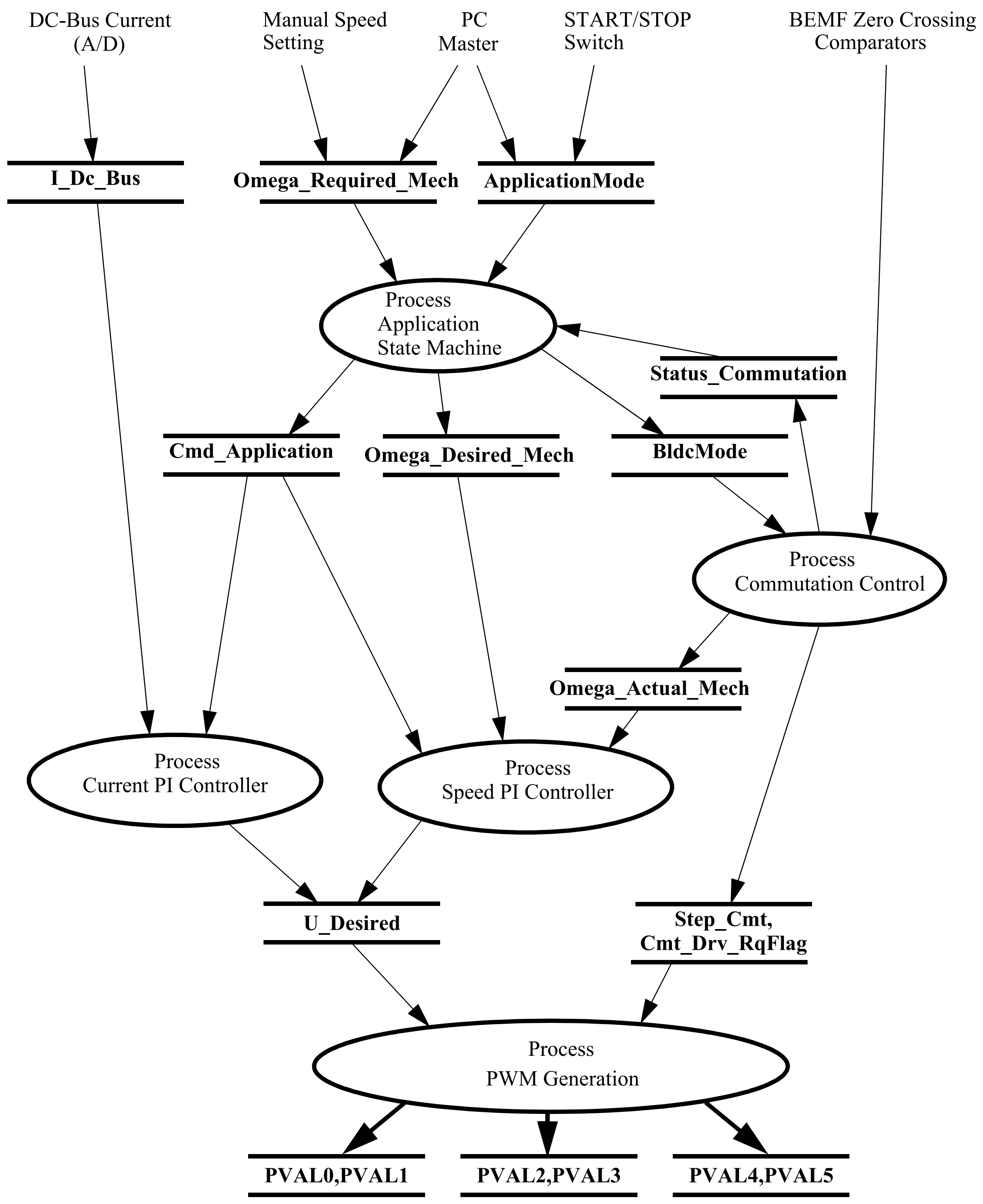


Figure 7-3. Data Flow - Part 1

Protection processes are shown in **Figure 7-4** and described in the following sub-sections.

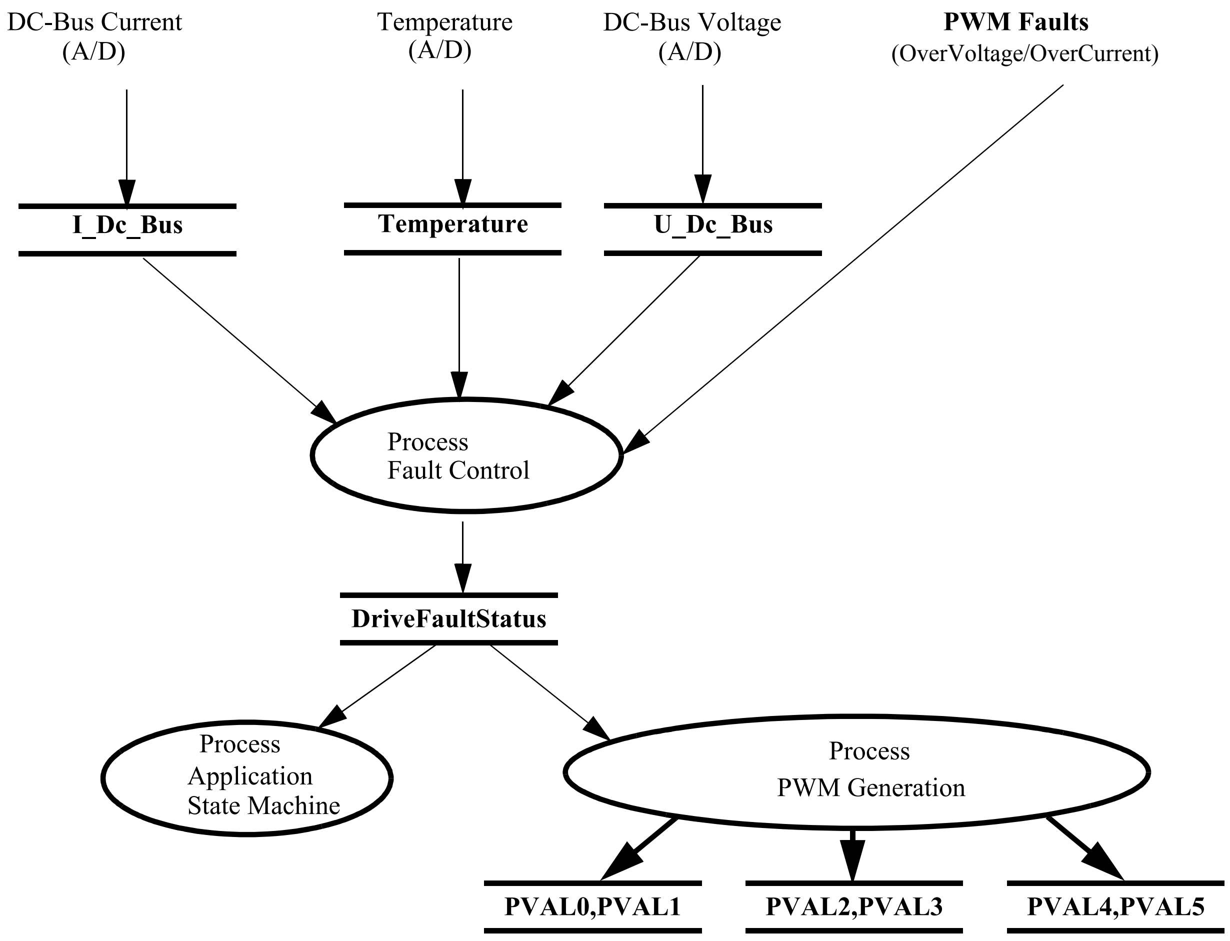


Figure 7-4. Data Flow - part2

7.2.1 Process Application State Machine

This process controls the application subprocesses by status and command words as can be seen in **Figure 7-3.**

Based on the status of the **Status_Commutation** (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**. Its outputs, **Step_Cmt** and **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 Speed PI Controller

The general principle of the speed PI control loop is illustrated in **Figure 7-5.**

.

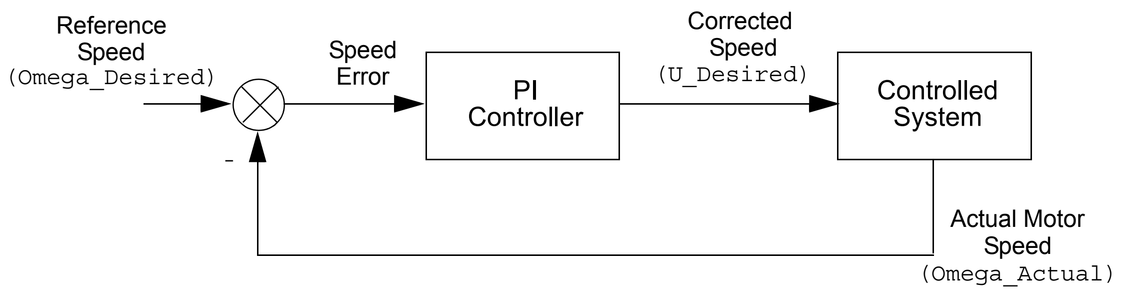


Figure 7-5. Closed Loop Control System

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 the timer interrupt with the constant **PER_SPEED_SAMPLE_S**. The PI controller is proportional and integral constants were set experimentally.

7.2.4 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 Alignment stage.

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

Current Controller period = 1/pwm frequency.

The PI controller is proportional and integral constants were set experimentally.

7.2.5 Process PWM Generation

The Process PWM Generation creates:

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

7.2.6 Process Fault Control

The Process Fault Control is used for drive protection. It can be understood from **Figure 7-4**. 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

The state diagrams of the whole SW are described below.

7.3.1 Main SW States - General Overview

The SW can be split into following processes:

- Process Application State Machine
- Process Commutation Control
- Process Speed PI Controller
- Process Current PI Controller
- Process PWM Generation
- Process PWM Generation

as shown in 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

In Main SW initialization provides following actions:

- **CmdApplication** = 0
- **DriveFaultStatus** = NO_FAULT
- PCB Motor Set Identification
  - **boardId** function is used to detect one of 3 possible hardware sets. According to used hardware one of three control constant sets are loaded (functions **EVM_Motor_Settings, LV_Motor_Settings, HV_Motor_Settings**)
- ADC Initialization
- Led diodes initialization
- Switch (Start/Stop) initialization
- Push Buttons (Speed up/down) initialization
- Commutation control initialization
- PWM initialization
- PWM fault interrupts initialization
- Zero Crossing inputs = Quadrature decoder filter initialization
- Output Compare Timers initialization

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

7.3.3 State Diagram - Process Application State Machine

Process Application State Machine state diagram is displayed in Figure 7-6. Application State Machine controls the main application functionality.

The application can be controlled:

- manually
- from PC master software

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

In PC master software control mode the Start/Stop is controlled manually and the Required Speed is set via 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 SW processes are controlled according this Application State Machine status.

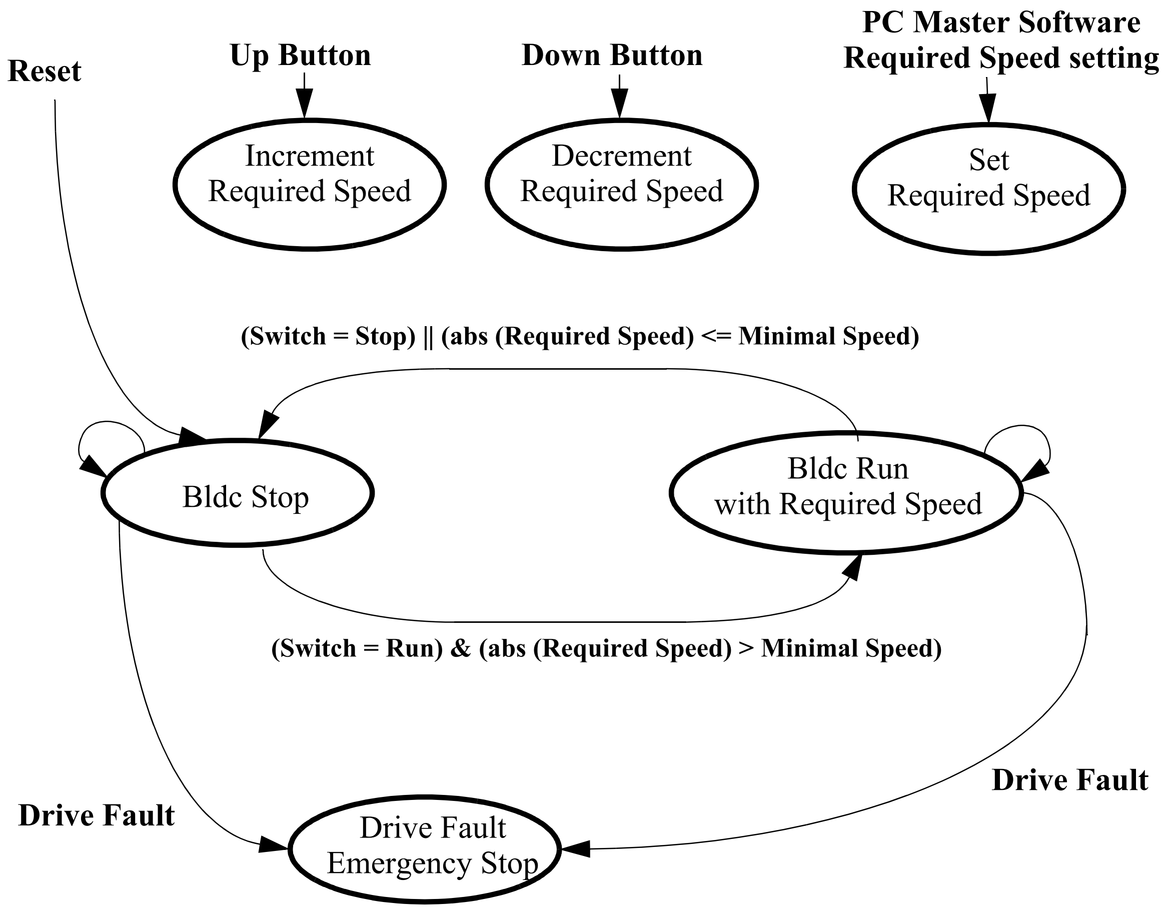


Figure 7-6. State Diagram - Process Application State Machine

7.3.4 State Diagram - Process Commutation Control

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

- Alignment state
  - motor is powered with current through 2 phases - no commutations provided.

- Starting (Back-EMF Acquisition) State
  - motor is started with making first 2 commutations, then it is running as at 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.

Drive starts by setting the Alignment stage where the Alignment commutation step is set and Alignment stage is timed. After the time-out the Starting stage is entered with initialization of BEMF Zero Crossing algorithms. After the required number of successive commutations with correct Zero Crossing are done, the Running stage is entered. If the number of commutations with wrong Zero Crossing exceeds a pre-determined Maximal number, the Running and Starting stages are exited to the Stop stage. The commutation control is determined by the variable **StatusCommutation**.

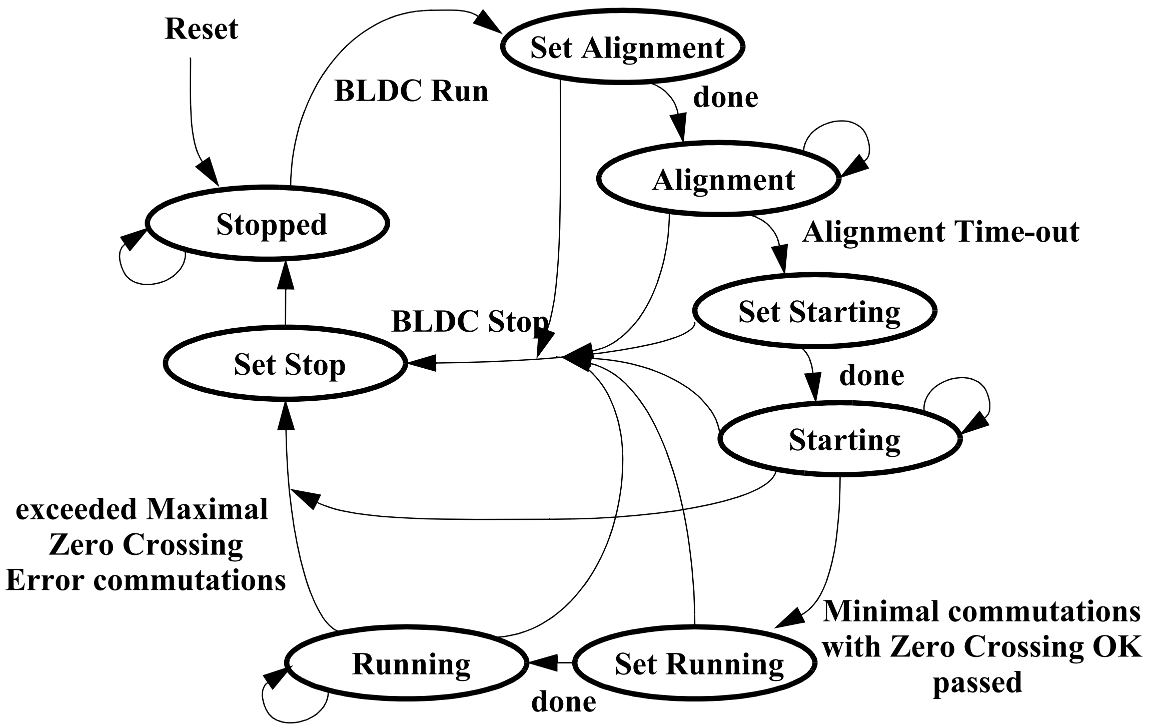


Figure 7-7. State Diagram - Process Commutation Control

7.3.4.1 Commutation Control - Running State

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

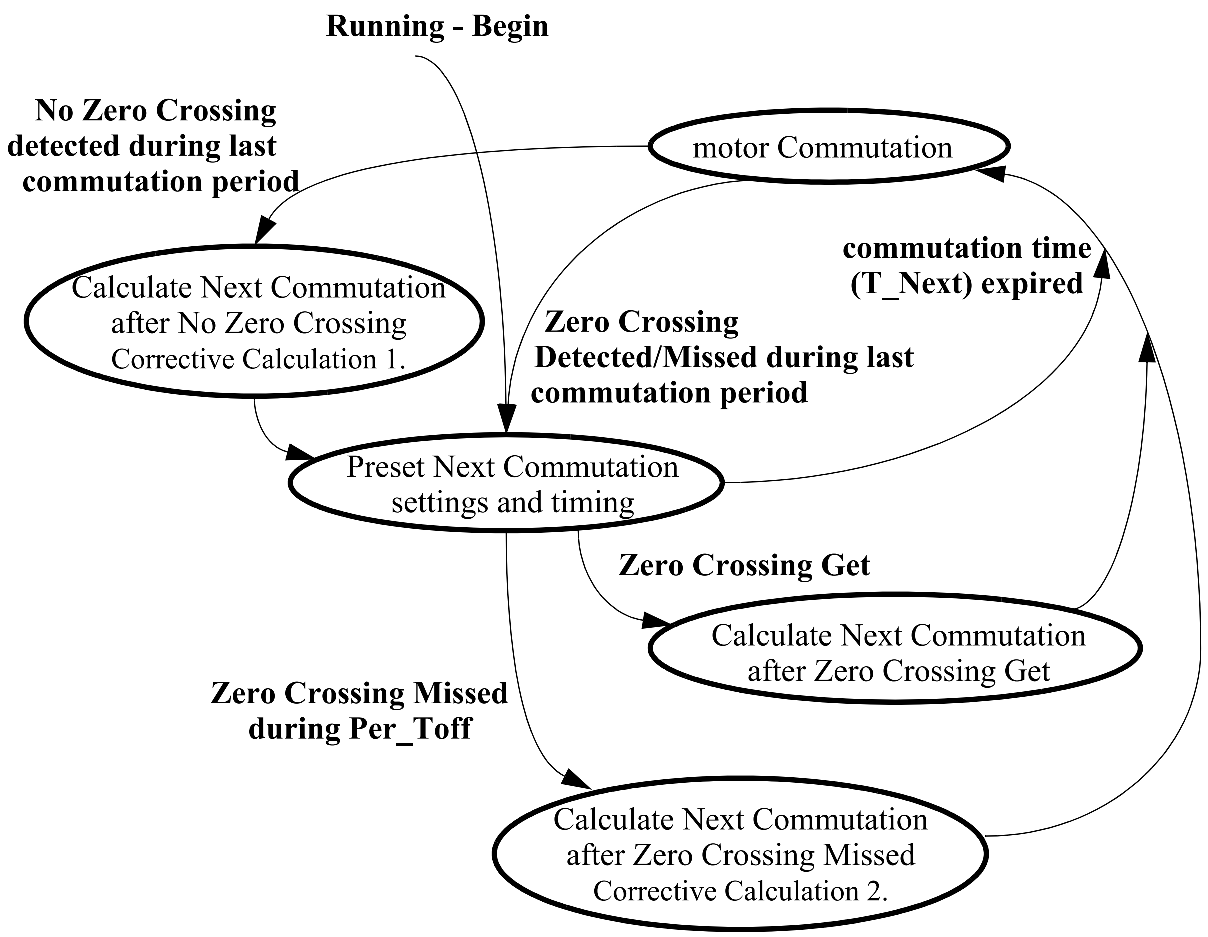

Running - Begin
No Zero Crossing detected during last commutation period
motor Commutation
Calculate Next Commutation after No Zero Crossing Corrective Calculation 1.
commutation time (T_Next) expired
Zero Crossing Detected/Missed during last commutation period
Preset Next Commutation settings and timing
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 7-8. Substates - Running

This state is almost wholly serviced by the BLDC Zero Crossing algorithms which are documented in Motor Control.pdf, chapter BLDC Motor Commutation with Zero Crossing Sensing (see **Section 12.1**). First the **bldczcHndlr** is called with actual time from 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 main loop and in Cmt (commutation) Timer Interrupt.

7.3.4.2 Commutation Control - Starting state

The starting state is the Running state as described in **Figure 7-8**.

7.3.4.3 Commutation Control - Set Running

This state services the transition from Starting (Back-EMF Acquisition) state to Running state by the BLDC Zero Crossing algorithms (see Section 12.1) according to the following actions:

- T_Actual = Cmt Timer Counter
- setting new commutation parameters and initialized commutation with **bldczcHndlrInit** algorithm
- initialization of computation with **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
- 2 additional motor commutations are prepared (in order to create 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

In this state:

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

7.3.4.6 Commutation Control - Set Alignment

In this state BLDC motor is set to Alignment state, where voltage is put across 2 motor phases and current is controlled to be at required value. The following actions are provided in Set Alignment state:

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

7.3.5 State Diagram - Process Speed PI Controller

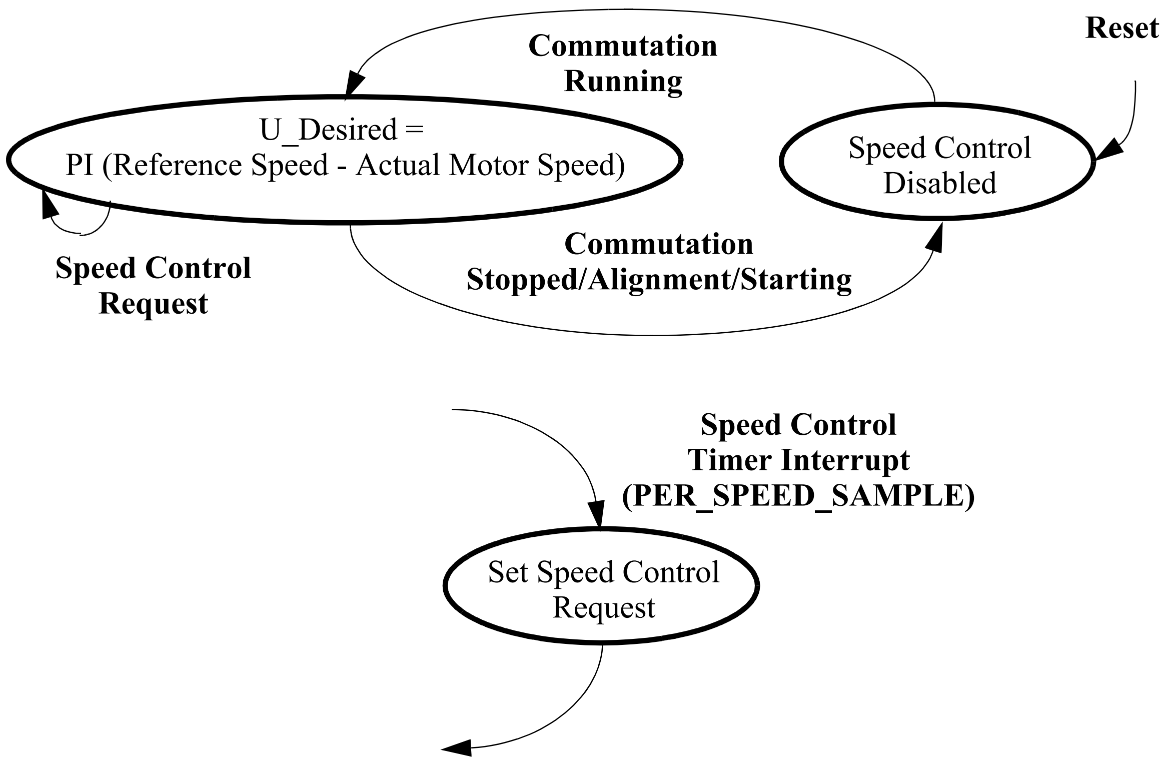

Commutation Running
Reset
U_Desired = PI (Reference Speed - Actual Motor Speed)
Speed Control Disabled
Speed Control Request
Commutation Stopped/Alignment/Starting
Speed Control Timer Interrupt (PER_SPEED_SAMPLE)
Set Speed Control Request

Figure 7-9. 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.

7.3.6 State Diagram - Process Current PI Controller

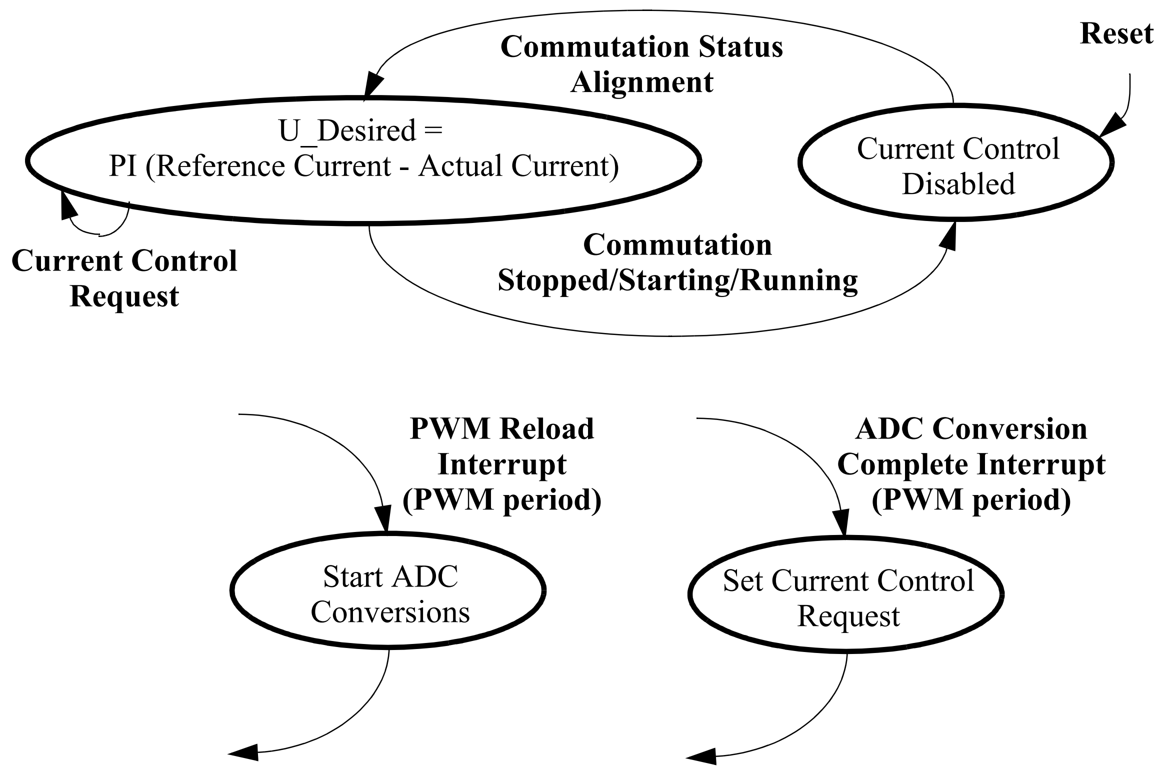


Figure 7-10. State Diagram - Process Speed PI Controller

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

7.3.7 State Diagram - Process Fault Control

The process Fault State is described by Interrupt subroutines which provide its functionality.

7.3.7.1 PWM Fault A Interrupt Subroutine

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

In this interrupt subroutine following faults from 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**

7.3.7.2 ADC Low Limit Interrupt Subroutine

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

In this interrupt subroutine 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**

7.3.7.3 ADC High Limit Interrupt Subroutine

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

In this interrupt subroutine 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. SDK Implementation

The Embedded SDK is a collection of APIs, libraries, services, rules and guidelines. This software infrastructure is designed to let 56F8xx software developers create high-level, efficient, portable code. This chapter describes how the BLDC motor control application with BEMF Zero Crossing is written under SDK.

8.1 Drivers and Library Function

The BLDC motor control application with BEMF Zero Crossing uses the following drivers:

- ADC driver
- Quadrature Timer driver
- Quadrature encoder
- PWM driver
- Led driver
- Switch driver
- Button driver

The all driver except Timer driver are included in BSP.LIB library. The Timer driver is included in SYS.LIB library.

The BLDC motor control application with BEMF Zero Crossing uses the following library functions:

- *bldczcHndlrInit* (handler initialization for BLDC commutation control with BEMF Zero Crossing; bldc.lib library)
- *bldczcHndlr* (handler for BLDC commutation control with BEMF Zero Crossing; bldc.lib library)
- *bldczcTimeoutIntAlg* (time-out interrupt algorithm for BLDC commutation control with BEMF Zero Crossing; bldc.lib library)

- *bldczcHndlrStop* (stop handler for BLDC commutation control with BEMF Zero Crossing; BLDC.LIB library)
- *bldczcComputInit* (computation initialization for BLDC commutation control with BEMF Zero Crossing; BLDC.LIB library)
- *bldczcComput* (computation for BLDC commutation control with BEMF Zero Crossing; bldc.lib library)
- *bldczcCmtInit* (commutation initialization for BLDC commutation control with BEMF Zero Crossing; BLDC.LIB library)
- *bldczcCmtServ* (commutation serve for BLDC commutation control with BEMF Zero Crossing; BLDC.LIB library)
- *bldczcZCrosInit* (zero crossing initialization for BLDC commutation control with BEMF Zero Crossing; BLDC.LIB library)
- *bldczcZCrosIntAlg* (zero crossing interrupt algorithm for BLDC commutation control with BEMF Zero Crossing; BLDC.LIB library)
- *bldczcZCrosServ* (zero crossing service for BLDC commutation control with BEMF Zero Crossing; BLDC.LIB library)
- *controllerPItype1* (calculation of PI controller; MCFUNC.LIB library)
- *boardId* (hardware board identification; BSP.LIB library)

8.2 *Appconfig.h* File

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

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

The *appconfig.h* file is divided to two sections. The first section defines which components of SDK are included in the application. The second part of the *appconfig.h* file overrides standard settings of components during their initialization.

8.3 Driver Initialization

Each peripheral on the chip or on the EVM board is accessible through its driver. The driver initialization of all used peripherals is described in this chapter. For detailed description of drivers see document **Embedded SDK (Software Development Kit) Targeting 5680X Platform w**here X means the target device (56F803, 56F805, 56F807).

To open driver the following step must be done:

- fill configuration structure if necessary (this depends on the type of driver)
- write the configuration items to *appconfig.h* if necessary (this depends on the type of driver)
- call the *open* (create) function

The access to peripheral functions driver is provided by the *ioctl* function.

8.4 Interrupts

The SDK serves the calling of interrupt routines and automatically clears interrupt flags. The user defines the callback functions which are called during interrupts. The callback functions are assigned during the driver's opening. The callback function assignment is defined as one item of the initialization structure which is used as input parameter of open function. Some drivers define the callback function in the *appconfig.h* file.

9. PC Master Software

PC master software was designed to provide the debugging, diagnostic and demonstration tool for development of algorithms and application. It consists of components running on PC and parts 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
```

This can be seen in the Software Design chapter of the SDK. It automatically includes the SCI driver and installs all necessary services.

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

10. Controller Usage

Figure 10-1 shows how much memory is used to run the BLDC motor drive with BEMF Zero Crossing in a speed closed loop. A part of the device's memory is still available for other tasks.

Table 10-1. 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	32K	13094	8915
Data RAM	2K	1425+352	1057+352

11. Setting of SW parameters for other motor kits

The SW 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.

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

...\dsp5680xevm\nos\applications\bldc_zerocross\bldcadczcdefines.h

and *config* files:

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

or in the file (Flash version):

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

The motor control drive usually needs setting/tuning of:

- dynamic parameters
- current/voltage parameters

The SW selects valid parameters (one of the 3 parameter sets) based in the identified hardware. **Table 11-1** shows the starting string of the SW constants used for each hardware.

Table 11-1. SW 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 will be replaced by x. The sections is sorted in order recommended to follow, when one is tuning/changing parameters.

Notes: Most important constants for reliable motor start-up are described in **Section 11.2.2** and in **Section 11.1.2**.

11.1 Current and Voltage Settings

11.1.1 DC Bus Voltage, Maximal and Minimal Voltage and Current Limits Setting

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 SW 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)

11.1.2 Alignment Current and Current Regulator Setting

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

The current during Alignment stage (before motor starts) is recommended to be set to nominal motor current value.

#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 documentation.)

The current controller works with constant execution (sampling) period determined by 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 Alignment stage, 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

11.2 Commutation Control Settings

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

11.2.1 Alignment Period

The time duration of alignment stage must be long enough to stabilize the rotor before it starts.

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

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

Notes: For first tuning it is recommended to set this period high enough (e.g. 5s). Then, if the motor works well it can be significantly lowered (e.g. 0.1s).

11.2.2 Start-up Periods

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

All this section settings 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] */
```

The unit of these constants is 1 µs.

`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 BEMF Zero Crossing is not sensed.

The older versions of the software (SDK 2.2) used the constants with system units:

```
#define x_PER_CMTSTART        0x0c00       /* Start Commutation Period * [1.7777us] */
#define x_PER_TOFFSTART       0x1800       /* Start Zero Crossing
                                              Toff Period * [1.7777us] */
```

The unit of these constant is 1.777us. These constants are automatically calculated in newer SDK software versions.

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

Then the first motor commutation period = `x_PER_CMTSTART_US * 2`

The Back-EMF Zero Crossing is not sensed during whole first period, because it is very small and hence the Zero Crossing information is not reliable during this period.

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

Table 11-2 helps with setting of this constant.

Table 11-2. 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 11.3.1**) helps if a problem with start up is encountered using the above stated setting .

11.2.3 Minimal Zero Commutation of Starting (Back-EMF Acquisition) Stage

```
#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 the Zero Crossing OK commutation to finish the BLDC starting phase.

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

11.2.4 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 again, whenever **x_MAX_ZCROSERR** successive commutations with problematical Zero Crossing appears.

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

11.2.5 Commutation Proceeding Period

Commutation preceeding period is the constant time after motor commutation, when BEMF Zero Crossing is not measured (until the phase current decays to zero).

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

The unit of this constant is 1 µs.

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

The older versions of the software (SDK 2.2) used the constant with system units:

```
#define x_CONST_PERPROCCMT   100  /* Period of Commutation proceeding [1.7777us]*/
```

The unit of this constant is 1.777us. This constant is automatically calculated in newer SDK software versions.

11.2.6 Commutation Timing Setting

Notes: Normally this structure should not necessarily be changed. If the constants described in this section need to be changed a detailed study of the control principle needs to be studied in **Section 5** and SDK document describing algorithms BLDC motor commutation with Zero Crossing sensing (*MotorControl.pdf*).

If it is required to change the motor commutation advancing (retardation) the coefficients in starting and running structures need 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 stage (see **Section 5.4.3**).

The **x_RunComputInit** structure is used by the application software during Running stage (see **Section 5.4.2**).

```
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 BEMF Zero Crossing is captured. The application SW multiplies fractional Coef_CmtPrecomp with commutation period.

```
Coef_HlfCmt
```

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

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

```
Coef_Toff
```

determines the interval between BEMF Zero Crossing and motor commutation

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

11.3 Speed Setting

11.3.1 Maximal and Minimal Speed and Speed Regulator Setting

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

In order to compute the speed setting, it is important to set the 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 have 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 integralgain 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.

The coefficients x_SPEED_PI_PROPORTIONAL_GAIN_REAL (resp. x_SPEED_PI_INTEGRAL_TI_REAL) are not directly used for regulator setting, but can be used to calculate `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.

12. References

12.1 Software Development Kit, SDK Rev.2.2

- **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*

12.2 User's Manuals and Application Notes

- **Low Cost High Efficiency Sensorless Drive for Brushless DC Motor using MC68HC(7)05MC4**, AN1627, Freescale Semiconductor, Inc.
- **56F800 16-bit Digital Signal Processor Family Manual**, DSP56F800FM, Freescale Semiconductor, Inc.
- **56F803 Evaluation Module Hardware User's Manual,** DSP56F803EVMUM, Freescale Semiconductor, Inc.
- **56F805 Evaluation Module Hardware User's Manual**, DSP56F805EVMUM, Freescale Semiconductor, Inc.
- **56F80x 16-bit Digital Signal Processor User's Manual**, DSP56F801-7UM, Freescale Semiconductor, Inc.
- **Evaluation Motor Board User's Manual,** MEMCEVMBUM, Freescale Semiconductor, Inc.
- **Optoisolation Board User's Manual,** Freescale Semiconductor, Inc.
- **PC Master User Manual,** 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

AN1914
Rev. 1
11/2005