Order by AN1917/D
(Motorola Order Number)
Rev. 0, 9/02

3-Phase PM Synchronous Motor Control with Quadrature Encoder Using DSP56F80x

Design of a Motor Control Application Based on Motorola Software Development Kit

Pavel Grasblum

Contents

1. Introduction of Application Benefit

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

PM Synchronous motors are very popular in a wide range of applications. Compared with DC motors, PM Synchronous motors are without a commutator, so they are more reliable than DC motors. Also, in comparison to AC induction motors, PM Synchronous motors have advantages. PM Synchronous motors generate the rotor magnetic flux with rotor magnets so that PM Synchronous motors are highly efficient. Therefore, PM Synchronous motors are used in high-end white goods (refrigerators, washing machines, dishwashers, etc.), high-end pumps, fans and in other appliances, which require high reliability and efficiency.

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

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

2. Motorola DSP Advantages and Features

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

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

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

Table 2-1. Memory Configuration

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

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

The PWM 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 1 to 16
- Individual software-controlled PWM outputs
- Mask and Swap of PWM outputs
- Programmable fault protection
- Polarity control
- 20mA current sink capability on the PWM pins
- Write-protectable registers

The PM Synchronous motor control utilizes the PWM block set in the complementary PWM mode, permitting generation of control signals for all switches of the power stage with inserted deadtime. The PWM block generates three sinewave outputs mutually shifted by 120 degrees.

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

- Each timer module consists of four 16-bit counters/timers
- Count up/down
- Counters are cascadable
- Programmable count modulo
- Max count rate equals peripheral clock/2 when counting external events
- Max count rate equals peripheral clock when using internal clocks
- Count once or repeatedly
- Counters are preloadable
- Counters can share available input pins
- Each counter has a separate prescaler
- Each counter has capture and compare capability

The PM Synchronous motor application utilizes one channel of the Quad Timer module counting in quadrature mode. It enables sensing of the rotor position using the Quadrature Encoder. The second channel of the Quad Timer module is set to generate a time base for a speed controller.

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

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

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

3. Target Motor Theory

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

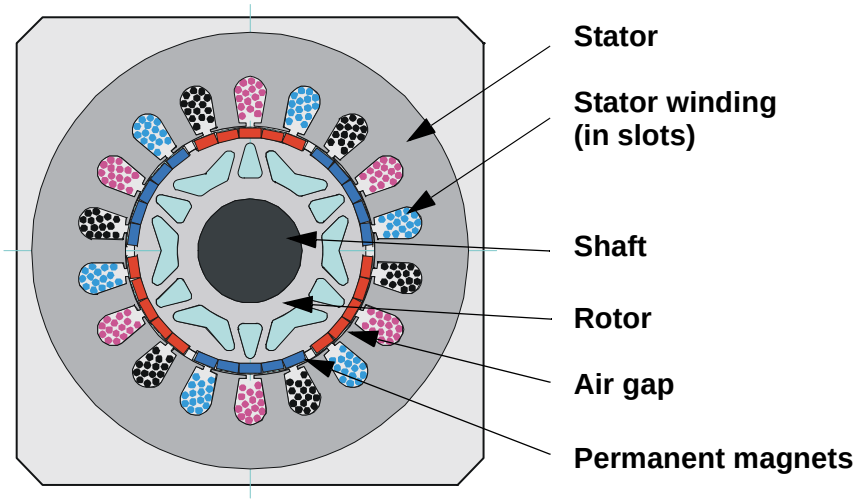

Figure 3-1. PM Synchronous Motor - Cross Section

In this respect, the PM Synchronous motor is equivalent to an induction motor where the air gap magnetic field is produced by a permanent magnet. It means that the rotor magnetic field is constant. PM Synchronous motors provide a set of advantages for designing modern motion control systems. The use of a permanent magnet to generate a substantial air gap magnetic flux makes it possible to design highly efficient PM motors.

The PM Synchronous motor is described by the following equations:

$$u_S = r_S \cdot i_S + \frac{d\psi_S}{dt} \quad \text{(EQ 3-1.)}$$

$$\psi_S = L_S \cdot i_S + \psi_M \quad \text{(EQ 3-2.)}$$

$$T_e = |i_S| \cdot |\psi_S| \cdot \sin(\angle i_S, \angle \psi_S) = |i_S| \cdot |\psi_M| \cdot \sin(\angle i_S, \angle \psi_M) \quad \text{(EQ 3-3.)}$$

where

u_S	is the space phasor of stator voltage
i_S	is the space phasor of stator current
r_S	is the stator phase resistance

Ψ_S	is the space phasor of stator magnetic flux
Ψ_M	is the space phasor of rotor magnetic flux evoked by the permanent magnet
T_e	is the electrical torque

As can be seen from equation **(EQ 3-3.)**, optimal torque is generated when the stator current vector is placed ± 90° relative to the rotor permanent magnet flux space vector. This situation is shown in **Figure 3-2**.

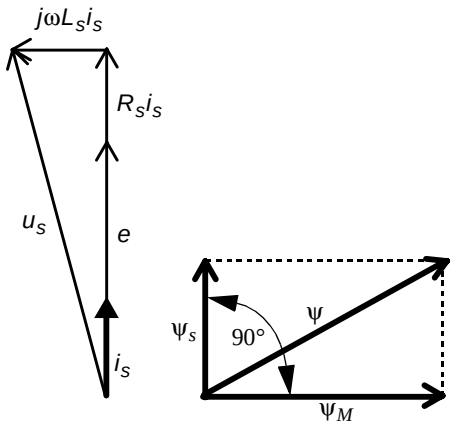

Figure 3-2. Torque Optimal Control of PM Synchronous Motor

where

R_S	is the stator resistance
L_S	is the stator inductance
e	is the Back-EMF voltage
Ψ	resultant magnetic flux

3.1 Digital Control of a PM Synchronous Motor

A PM Synchronous motor is driven by sinewave voltage coupled with the given rotor position. The generated stator flux together with the rotor flux, which is generated by a rotor magnet, defines the torque, and thus speed, of the motor. The sinevawe voltage output have to be applied to the 3-phase winding system in a way that angle between the stator flux and the rotor flux is kept close to 90° to get the maximum generated torque. To meet this criterion, the motor requires electronic control for proper operation.

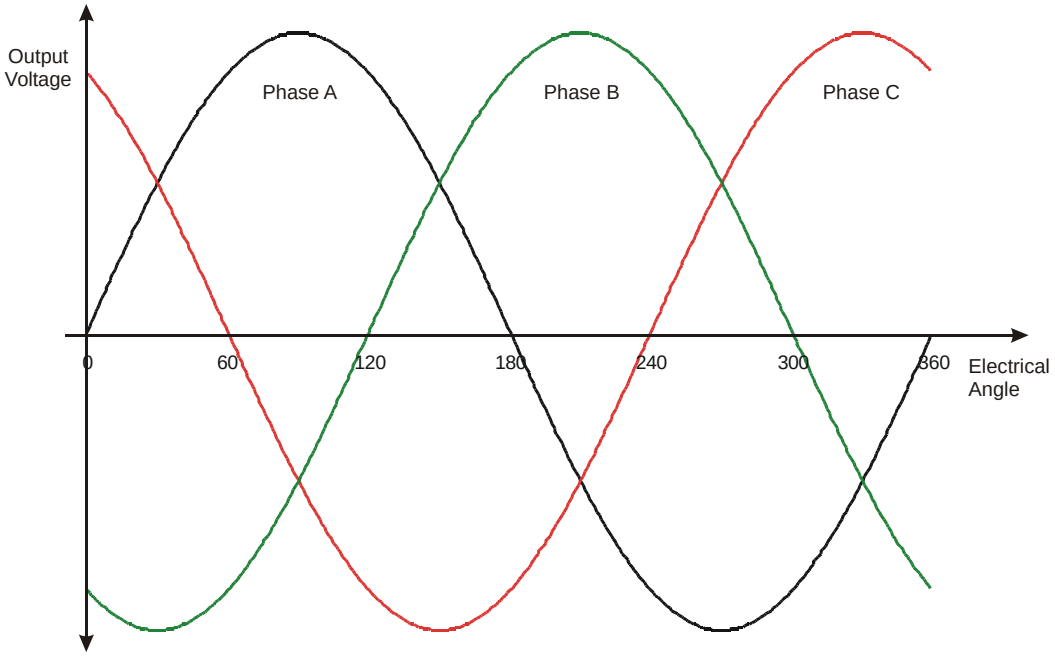

Figure 3-3. Sinewave Voltage Output Applied onto a PM Synchronous Motor

For a common 3-phase PM Synchronous motor, a standard 3-phase power stage is used. The same power stage is used for AC induction and BLDC motors. Such a power stage for 3-phase PM Synchronous motors is illustrated in **Figure 3-4**. The power stage utilizes six power transistors with independent switching. The power transistors are switched in the complementary mode. The sinewave output is generated using a PWM technique.

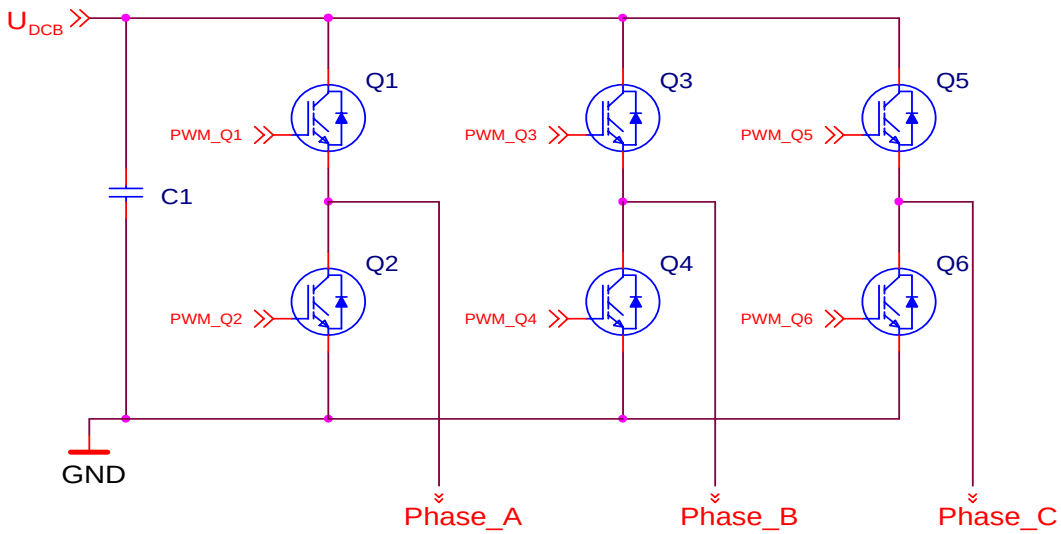

Figure 3-4. 3-Phase Power Stage

3.1.1 Control Technique

The presented control algorithm demonstrates the principle of PM Synchronous motor control and use of the DSP56F80x peripheral. It means that this algorithm can be used as starting point for more sophisticated algorithms.

As is well known, the PM Synchronous (Permanent Magnet) motor is very similar to a Brushless DC motor. The PM Synchronous motors differ in three respects:

- sinusoidal distribution of magnet flux in the air gap
- sinusoidal current waveforms
- sinusoidal distribution of stator conductors.

Using a six-step control technique we get six flux vectors. This technique is commonly used for BLDC motors (see **[14]**, **[15]**). In the case of sinusoidal voltage output, we are able to generate the stator flux in any position. The resultant flux is calculated as the sum of flux vectors of Phase A, Phase B and Phase C (see **Figure 3-5**). If the phase voltage changes sinusoidally over time, a fluent rotational flux field is generated. As a result, a PM Synchronous motor runs smother and quiter than a BLDC motor.

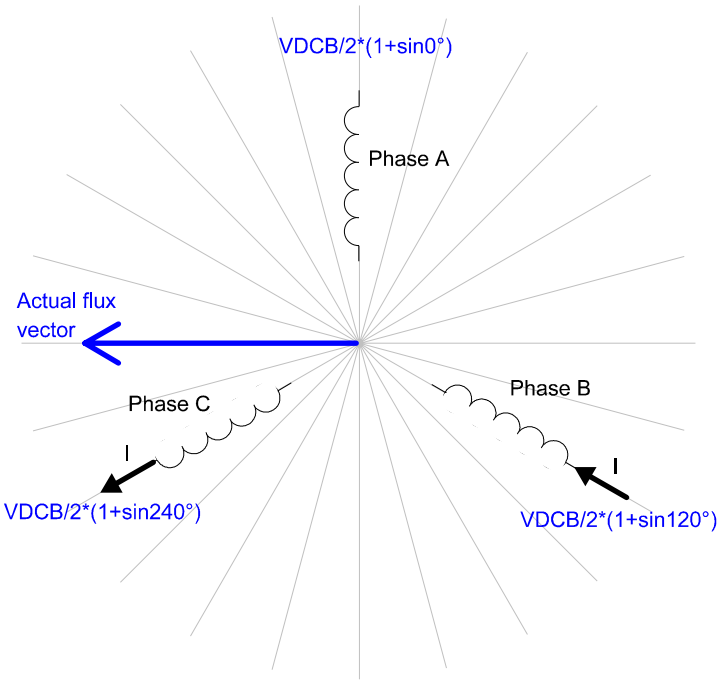

Figure 3-5. Stator Flux Generation

The motor runs with optimal torque generation when the angle between the stator and rotor flux is 90 electric degrees (see **Figure 3-6 a)**).

To ensure the angle between rotor and stator flux equals to 90 electric degrees it is necessary to know the position of the rotor and stator flux.

The position of the rotor flux is bound to the rotor position. Thus, by measuring rotor position we can get the exact position of the rotor flux.

The position of stator flux is bound to the vector of the stator current. To know the exact position of the stator flux, it requires measurement of the phase currents and the calculation of the stator current vector. Since the presented application does not measure current, there is no way to obtain the position of current vector.

To avoid the current measurement there is one of possible solution which aligns the vector of the applied voltage to be orthogonal to the rotor position (see **Figure 3-6 b)**). As can be seen, the angle between the stator and rotor flux is not exactly 90 electric degrees because the voltage drop on the stator inductance is not compensated. The real angle is lower than 90 electric degrees and depends on the load.

For low-cost applications, such a solution is fully sufficient. For high-end applications, the current measurement and the stator flux need to be evaluated. (see **[16]**)

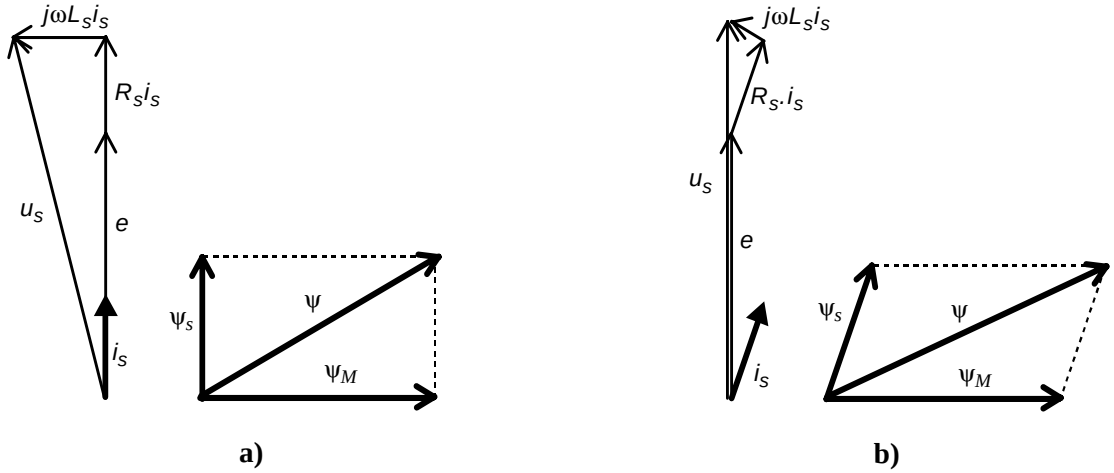

Figure 3-6. PM Synchronous Motor Phasor Diagram

3.1.2 Position Sensing

The rotor position is obtained from a Quadrature Encoder mounted on the rotor shaft. The encoder transfers the rotational movement into signal pulses corresponding to the position. The Quadrature Encoder output signals are connected to the on-chip Quadrature Decoder input. The signals go through a digital filter to the Quad Timer. The Quad Timer is set to count in quadrature mode. During alignment, the Quad Timer is preset to a value which represents a shift of the rotor position by 90 electrical degrees. Thus, the applied voltage is aligned with Back-EMF according to **Figure 3-6 b)**.

3.1.3 Position Alignment

Since the Quadrature Encoder doesn't give the absolute position, we need to know exactly the rotor position before the motor is started. One possible, and very easy implementable, method is the rotor alignment to a predefined position. The motor is powered by a selected static voltage pattern (usually the zero position in the sinewave table) and the rotor aligns to the predefined position. The alignment is done only once during first motor start. **Figure 3-7** shows the position of the aligned rotor. After alignment the position counter is set to 90 electric degrees from alignment position in order to preset the angle between stator and rotor flux.

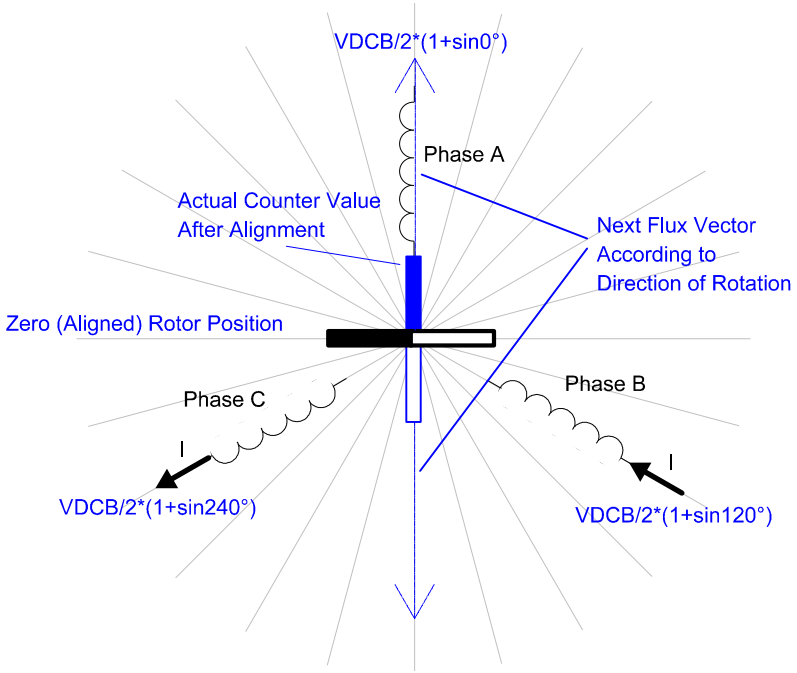

Figure 3-7. Alignment of Rotor Position

3.1.4 Speed Control

The correct shift between the rotor and stator flux ensures that the PM Synchronous motor generates a torque. The torque amplitude depends on the amplitude of the applied voltage. It means that the motor speed is also controlled by the amplitude of the applied voltage. The amplitude of the applied voltage is changed by the PWM technique. The required speed is controlled by a speed controller. The speed controller is implemented as a conventional PI controller. The PI controller compares the actual and required speeds. The difference between the actual and required speed is input to the PI controller and based on this difference. The PI controller calculates the duty cycle which corresponds to the voltage amplitude required to keep the required speed.

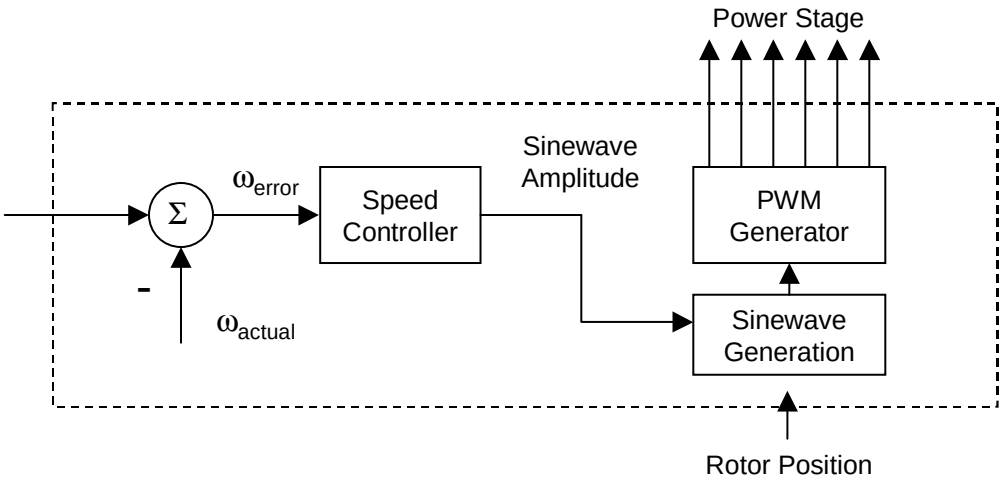

Figure 3-8. Speed Controller

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

$$u(t) = K_c\left[e(t) + \frac{1}{T_I}\int_0^{\tau} e(\tau)d\tau\right] \quad \text{(EQ 3-4.)}$$

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

$$u(k) = u_P(k) + u_I(k) \quad \text{(EQ 3-5.)}$$

$$u_P(k) = K_c \cdot e(k) \quad \text{(EQ 3-6.)}$$

$$u_I(k) = u_I(k-1) + K_c\frac{T}{T_I} \cdot e(k) \quad \text{(EQ 3-7.)}$$

where:

e(t), e(τ)	is the input error in time t, τ
e(k)	is the input error in step k
w(k)	is the desired value in step k
m(k)	is the measured value in step k
u(k)	is the controller output in step k
$u_p(k)$	is the proportional output portion in step k
$u_I(k)$	is the integral output portion in step k
$u_I(k-1)$	is the integral output portion in step k-1
T_I	is the integral time constant
T	is the sampling time
K_c	is the controller gain
t, τ	is the time
p	is the Laplace variable

4. System Concept

4.1 System Outline

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

- Voltage control of PM Synchronous motor using Quadrature Encoder
- Targeted for DSP56F803EVM, DSP56F805EVM, DSP56F807EVM
- Running on a 3-phase EVM Motor Board

- Control technique incorporates:
  - Voltage PM Synchronous motor control with speed-closed loop
  - Both directions of rotation
  - Motoring mode
  - Start from any motor position without rotor alignment
  - Minimum speed 50 RPM
  - Maximum speed 1000 RPM (limited by power supply)
- Manual interface (Start/Stop switch, Up/Down push button control, Led indication)
- PC master software control interface (motor start/stop, speed set-up)
- PC master software monitor
  - PC master software graphical Control Page (required speed, actual motor speed, start/stop status, DC-Bus voltage level, system status)
  - PC master software Speed Scope (observes actual & desired speeds)
- DC-Bus under-voltage fault protection

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

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

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

Note: The EMV Motor kit includes the 3-phase BLDC motor. Since the BLDC motor has very similar behaviors as PM Synchronous motor (see **Section 3.1**) the BLDC motor may be used for this application instead of the PM Synchronous motor.

4.2 Application Description

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

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

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

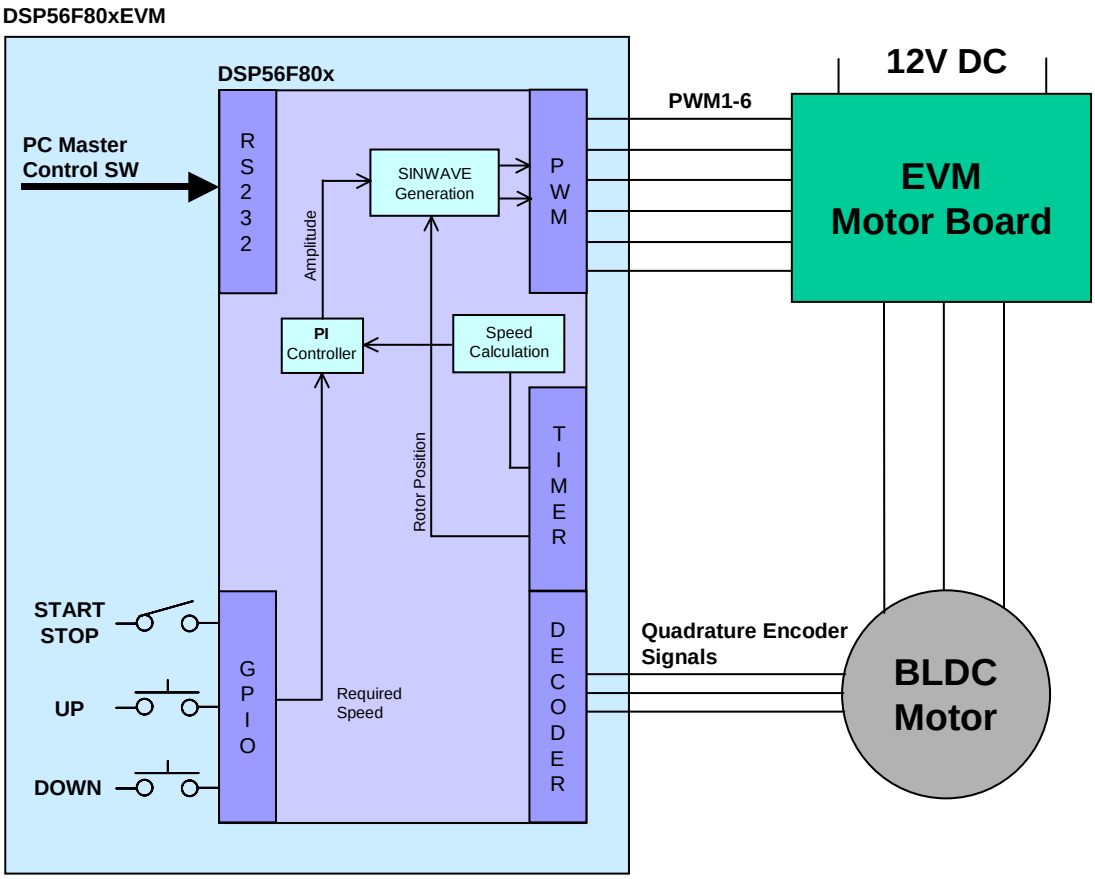

Figure 4-1. System Concept

The control process is as follows:

The state of the user interface is periodically scanned while the speed of the motor is measured on each new coming edge from the Quadrature Encoder (only one phase is used for speed measurement). According to the state of the control signals (Start/Stop switch, speed up/down buttons) the speed command is calculated. The comparison between the actual speed command and the measured speed generates a speed error. The speed error is brought to the speed PI controller that generates a new corrected amplitude of the sinewave output. The rotor position is also periodically scanned together with sinewave generation. The sinewave generation generates 3-phase sinewaves, shifted by 120 electrical degrees according to actual rotor position and the required amplitude. The output of sinewave generation defines directly the duty cycle of the PWM output signals for the power stage.

In the case of under-voltage, the PWM outputs are disabled and the fault state is displayed by an on-board LED.

4.3 Hardware Implementation

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

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

- DSP56F803
- DSP56F805
- DSP56F807

The application can run on an EVM motor board.

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

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

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

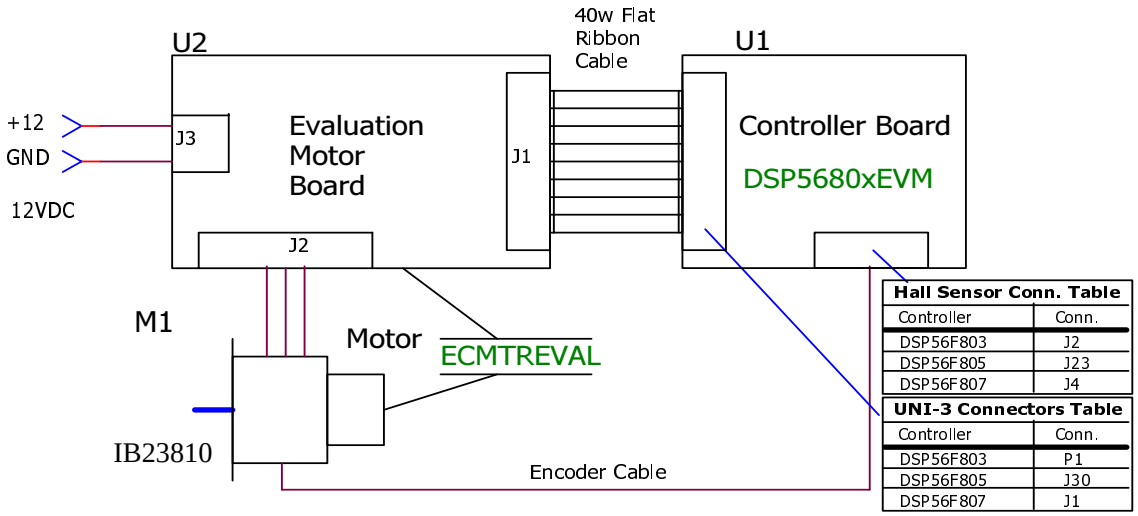

Figure 4-2. Low-Voltage Evaluation Motor HW System Configuration

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

- M1 - IB23810 Motor
  - supplied in kit ECMTREVAL - Evaluation Motor Board Kit
- U2 EVM Motor Board:
  - supplied in kit with IB23810 Motor: ECMTREVAL - Evaluation Motor Board Kit
  - described in: **Evaluation Motor Board User's Manual**
- U1 CONTROLLER BOARD for DSP56F803:
  - supplied as: DSP56803EVM
  - described in: **DSP Evaluation Module Hardware User's Manual**
- or U1 CONTROLLER BOARD for DSP56F805:
  - supplied as: DSP56805EVM
  - described in: **DSP Evaluation Module Hardware User's Manual**
- or U1 CONTROLLER BOARD for DSP56F807:
  - supplied as: DSP56807EVM
  - described in: **DSP Evaluation Module Hardware User's Manual**

5. Software Design

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

- Control Algorithm Data Flow
- Software Implementation

5.1 Data Flow

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

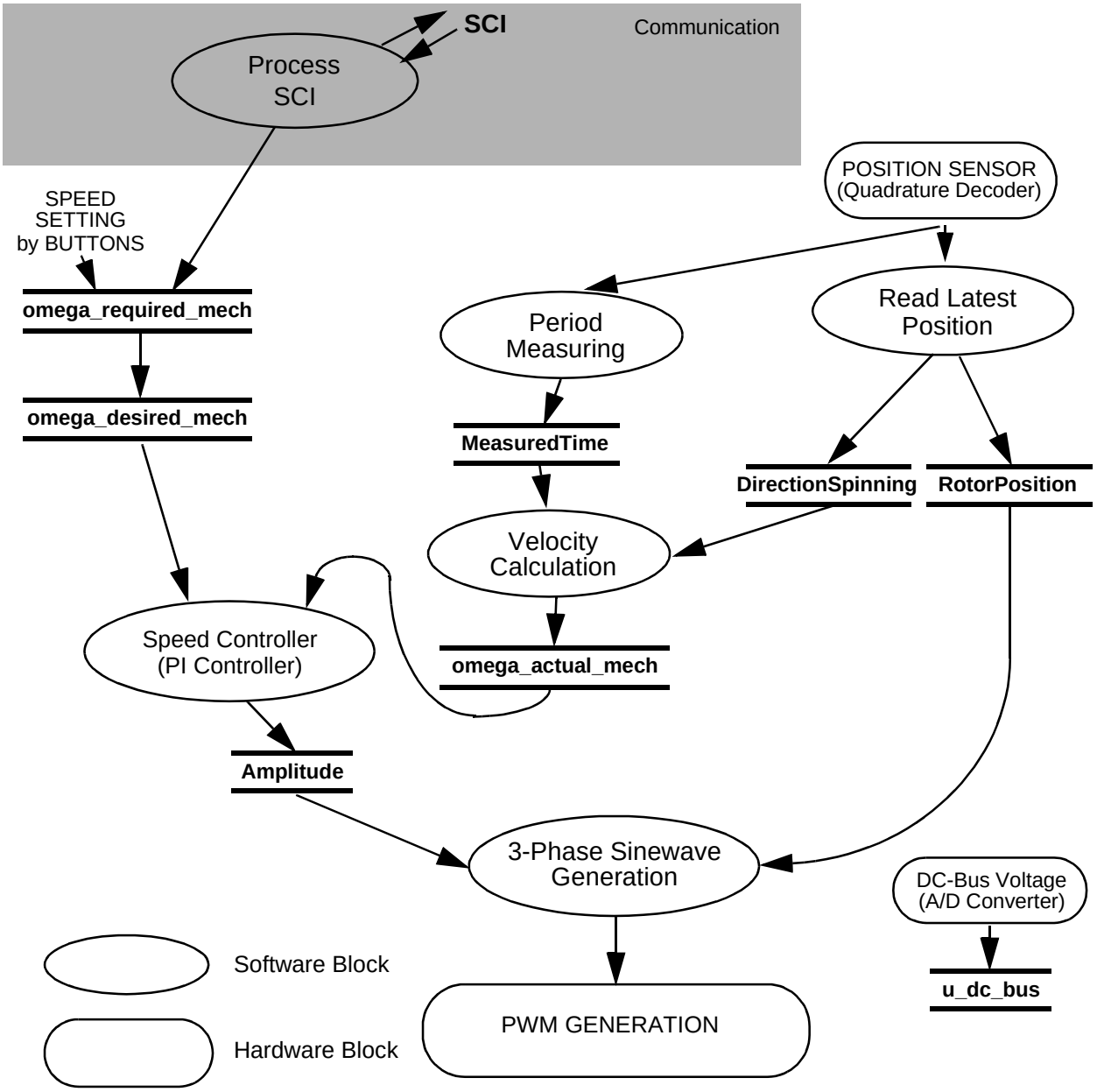

Figure 5-1. Main Data Flow

The main data flow can be divided to four parts:

- Speed control
- Velocity calculation
- 3-phase sinewave generation
- DC-Bus voltage measurement

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

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

For the rotor position scanning the Timer A0, set as a quadrature counter, is used. The Timer A0 gives the actual rotor position shifted by 90 electrical degrees after initialization. This rotor position `RotorPosition` is input to the 3-phase sinewave modulation together with required amplitude `Amplitude`. The result of the sinewave modulation is written directly to the PWM block. The rotor position scanning with sinewave modulation is performed by an interrupt routine, which is called each PWM reload (16 kHz). The next task, which is provided by an interrupt routine, is the calculation of the spin direction. The result, `DirectionSpinning`, is used for the speed calculation.

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

5.1.1 Read Latest Position

The process Read Latest Position is executed with each PWM Reload interrupt (16 kHz). The process reads the actual rotor position and calculates the direction of spinning.

5.1.2 Period Measuring and Velocity Calculation

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

5.1.3 Speed Controller

This process compares the required and actual speed and calculates the duty cycle of the PWM output signals. For detailed information see **Section 3.1.4, Speed Control**

5.1.4 3-phase Sinewave generation

The process 3-phase Sinewave Generation calculates the PWM output from the actual rotor position and the required sinewave amplitude. The output is written to the PWM module. This process is performed within the PWM Reload interrupt.

5.2 Software Implementation

The general software diagram shows the Main routine entered from Reset and the interrupt states (see **Figure 5-2**).

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

The following interrupt service routines are utilized:

- PWM Reload ISR- services signals generated by the Quadrature Encoder and generates the 3-phase sinewave output
- Input Capture ISR (TimerA1) - services period measurement for speed calculation
- Timer ISR - services the speed controller and LED diode blinking
- Push Button Up ISR and Push Button Down ISR - service the Up and Down push buttons
- SCI ISR - services communication with the PC master software

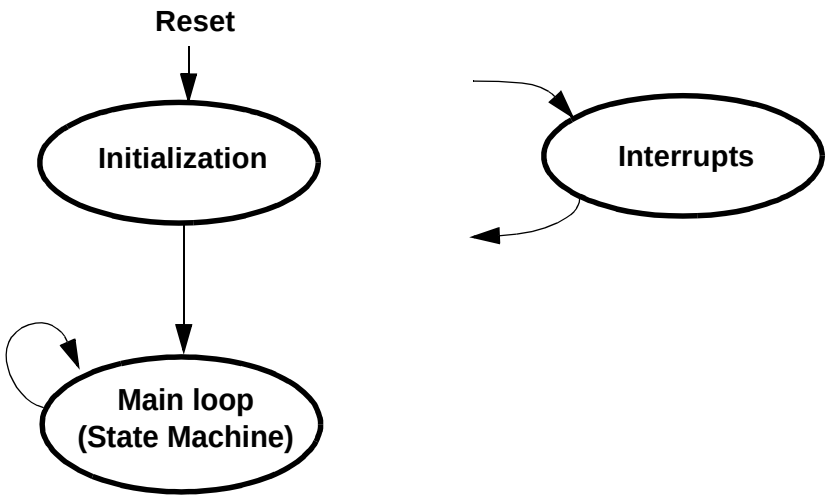

Figure 5-2. State Diagram - General Overview

5.2.1 Initialization

The Main Routine provides initialization of the DSP:

- Disables Interrupts
- Initializes DSP PLL
- Disables COP and LVI
- Initializes the POSIX Timer for time base reference 1 ms
- Initializes the LED
- Initializes the PWM module:
  - Center-aligned complementary PWM mode, positive polarity
  - PWM modulus - defines PWM frequency
  - PWM deadtime - defines PWM deadtime
  - Disable faults
- Initializes Quadrature Decoder
  - Sets on-chip digital filter of the Quadrature Decoder inputs
  - Connects Quadrature Decoder signals to QuadTimerA
- Initializes QuadTimerA - channel A0

  - set Count Mode to Quadrature Count
  - set Input Source to Input 0
  - set Input Polarity to Normal
  - set Secondary Input Source to Input 1
  - set Count Frequency to Repeatedly
  - set Count Length to Until Compare
  - set Count Direction to Down
  - disable Capture Mode
- Initializes QuadTimerA - channel A1
  - set Count Mode to Count
  - set Input Source to Bus Clock / 128
  - set Input Polarity to Normal
  - set Secondary Input Source to Input 1
  - set Count Frequency to Repeatedly
  - set Count Length to Past Compare
  - set Count Direction to Up
  - set Capture Mode = RisingEdges
  - associate Callback On Input Edge to CallbackOnNewEdge
  - associate CallbackOnOverflow to CallbackOnOverload
- Sets-up I/O ports (brake, switch, push buttons)
  - Brake, LED, switch on GPIO
  - Push buttons on interrupts IRQ0, IRQ1
- Initializes the Analog-to-Digital Converter
  - ADC set for sequential sampling, single conversion
  - Channel 0 = DC-Bus voltage
- Initializes control algorithm (speed controller, control algorithm parameters)
- Enables interrupts
- Starts ADC conversion

5.2.2 Interrupts

The interrupt handlers have the following functions:

- *PWM Reload* reads the actual rotor position, calculates the 3-phase sinewave output and spin direction and updates PWM Value Registers.
- *Input Capture Interrupt Handler (Timer A1)* reads the time between the two subsequent IC edges one phase of the Quadrature Encoder, which is used for speed calculation.
- *POSIX Timer Interrupt Handler* generates the time base 1ms. The routine, called within this time base, blinks the green LED diode, reads the result of the ADC conversion, calculates the speed and provides the speed controller.
- *Push Button Interrupt Handler* takes care of the push button service. The *UpButton Interrupt Handler* increments the desired speed, the *DownButton Interrupt Handler* decrements the desired speed.
- *PC and SCI Interrupt Handler*s provide SCI communication and service routines for the PC master software. These routines are fully independent of the motor control tasks.

5.2.3 Drive State Machine

The drive can be in any of the states shown in **Figure 5-3**, which shows the transition conditions between the drive states. The user is able to recognize the current state, by a blinking green LED diode. In the case of the *init* and *stop* state, the green LED diode blinks at a frequency of 2 Hz. In the *fault* state, the green LED diode blinks at a frequency of 8 Hz. During the *running* state, the green LED diode is continuously turned on.

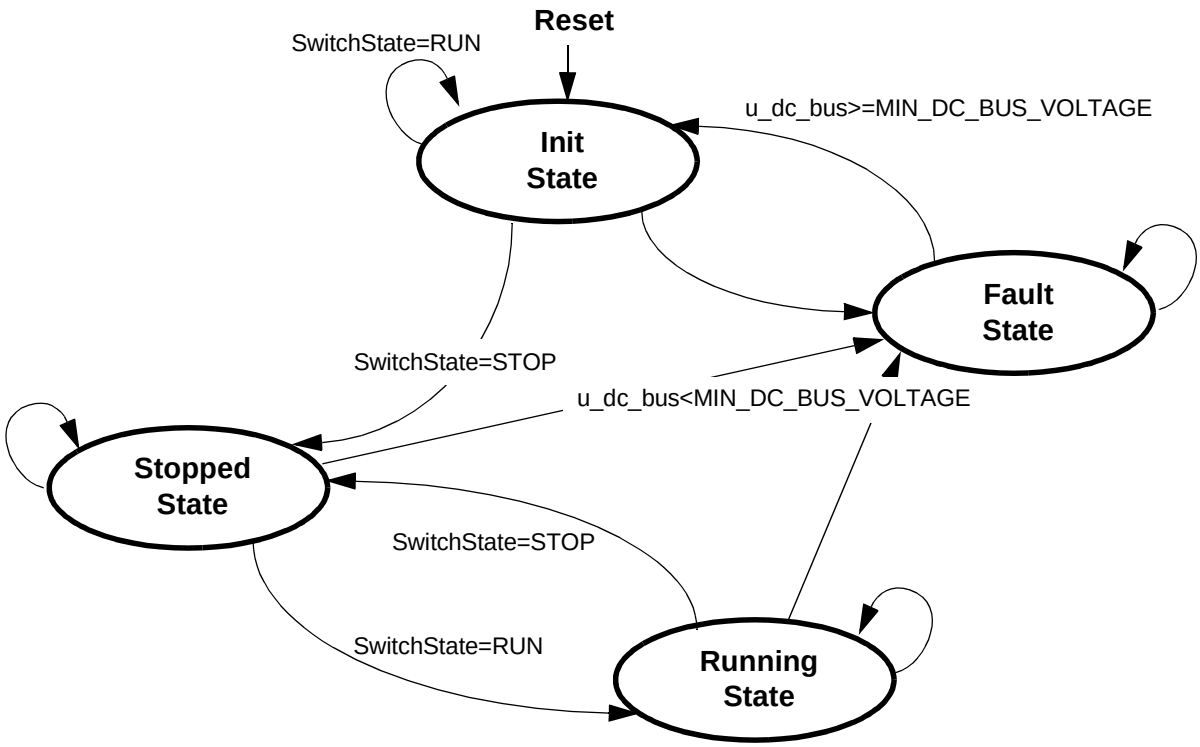

Figure 5-3. Drive State Machine Transitions

6. Implementation Notes

6.1 Scaling of Quantities

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

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

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

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

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

where:

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

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

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

6.1.1 DC-Bus Voltage Scaling

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

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

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

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

6.1.2 PI Controller Parameters

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

6.1.3 Velocity Calculation

The constant `OMEGA_ACTUAL_MECH_CONST` is defined by the following equations:

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

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

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

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

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

6.2 Motor Constant Calculation

The PM Synchronous motor control application uses the constants, which depend on a motor type (number of pole pairs) and on a Quadrature Encoder type (number of pulses per revolution). The depended constants are:

- PULSES_PER_REVOLUTION
- VOLTAGE_SHIFT
- SIN_TABLE_MULTIPLIER

The following paragraphs explain the constant calculations. The range for all constants is unsigned integer.

6.2.1 Constant PULSES_PER_REVOLUTION

The constant PULSES_PER_REVOLUTION defines the number of pulses of the Quadrature Encoder per electrical revolution. Since the Quadrature Encoder counts both rising and falling edges, the value is multiplied by four. The resultant value must be an integer.

$$\text{PULSES\_PER\_REVOLUTION} = \frac{4 \times \text{number of pulses per mech. revolution}}{\text{number of pole pairs}} - 1 \quad \text{(EQ 6-3.)}$$

In the case of presented application the constant is equal:

$$\text{PULSES\_PER\_REVOLUTION} = \frac{4 \times 500}{2} - 1 = 999$$

Note: In case that the constant is not an integer, it is necessary to set the constant PULSES_PER_REVOLUTION to a value which is equal to the number of pulses per mechanical revolution minus one. Then the actual rotor position has to be recalculated from the mechanical to electrical revolution.

6.2.2 Constant VOLTAGE_SHIFT

The constant VOLTAGE_SHIFT defines the shift of applied voltage by 90 el. degree and is calculated as:

$$\text{VOLTAGE\_SHIFT} = \frac{(\text{PULSES\_PER\_REVOLUTION} + 1)}{4} \quad \text{(EQ 6-4.)}$$

Then for the presented application the constant is equal to:

$$\text{VOLTAGE\_SHIFT} = \frac{(999 + 1)}{4} = 250$$

6.2.3 Constant SIN_TABLE_MULTIPLIER

The constant SIN_TABLE_MULTIPLIER rescales the rotor position, which is defined in pulses per electrical revolution, to the sinewave table, which is scaled from -1 to 1 <-π; π). Detailed information about sinewave generation can be found in the SDK documentation **[11]**. The constant is calculated as:

$$\text{SIN\_TABLE\_MUTIPLIER} = \frac{65535}{\text{PULSES\_PER\_REVOLUTION}} \times 256 \quad \text{(EQ 6-5.)}$$

For the presented calculation can be calculated:

$$\text{SIN\_TABLE\_MUTIPLIER} = \frac{65535}{999} \times 256 = 16794$$

7. SDK Implementation

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

7.1 Drivers and Library Functions

The PM Synchronous motor control application uses the following drivers:

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

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

The PM Synchronous motor control application uses the following library functions:

- mcgen3PhWaveSineIntp (3-phase sinewave generation; *mcfunc.lib* library)
- controllerPItype1 (standard PI controller; *mcfunc.lib* library)
- switchcontrol (switch control; *mcfunc.lib* library)

7.2 Appconfig.h File

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

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

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

7.3 Initialization of Drivers

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

The following steps are required to use the driver:

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

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

7.4 Interrupts

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

7.5 PC Master Software

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

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

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

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

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

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

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

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

- Start/stop control
- Set the motor speed

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

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

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

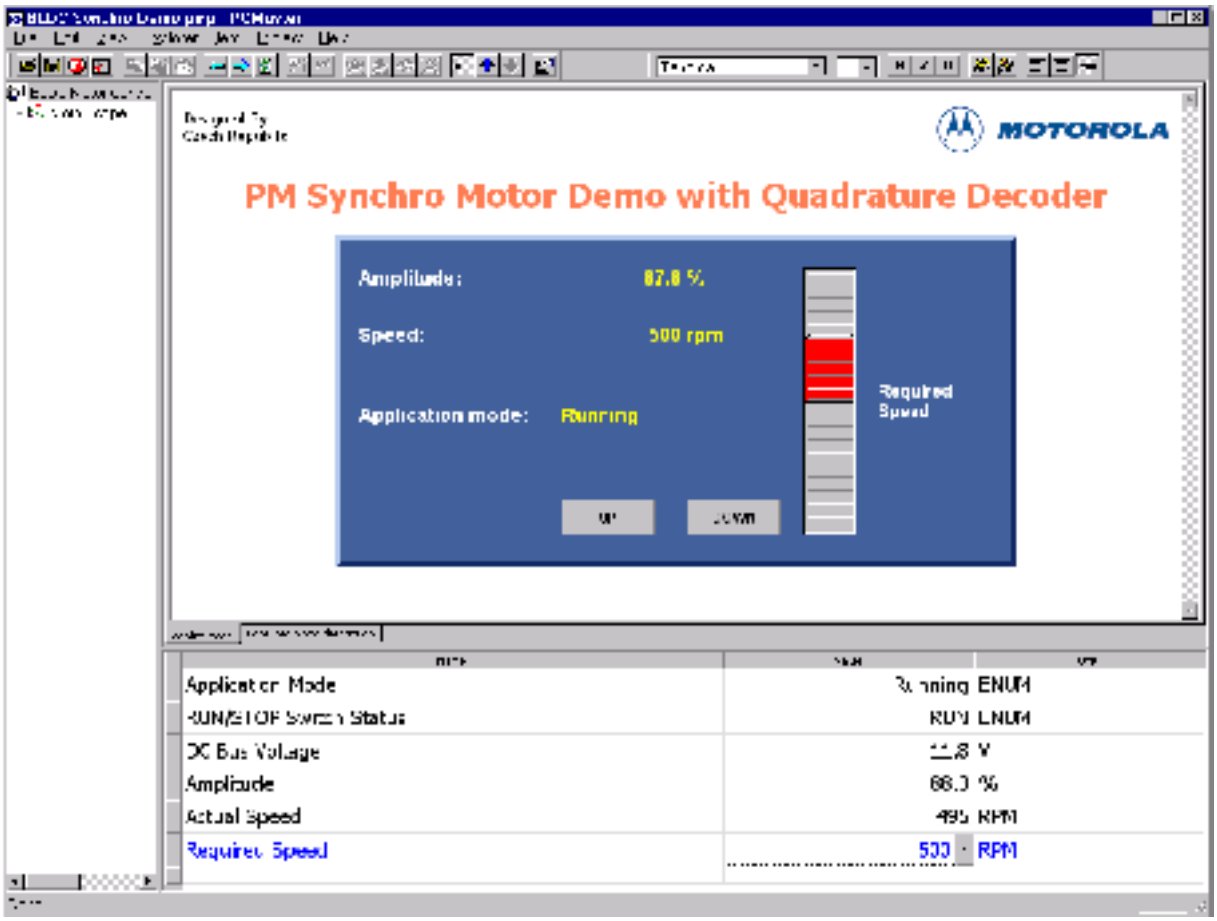

Figure 7-1. PC Control Window

8. DSP Usage

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

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

Memory (in 16 bit Words)	Available DSP56F803 DSP56F805	Available DSP56F807	Used Application	Used Application without PC master software, SCI
Program FLASH	32K	60K	8791	4926
Data FLASH	4k	8K	217	217
Program RAM	512	2K	101	101
Data RAM	2K	4K	903	568

9. References

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

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

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

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

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

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

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

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

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

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

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

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

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

[14] **3-Phase BLDC Motor Control with Hall Sensors Using DSP56F80x**, Pavel Grasblum, AN1916/D, Motorola 2001

[15] **3-Phase BLDC Motor Control with Quadrature Encoder Using DSP56F80x**, Pavel Grasblum, AN1915/D, Motorola 2001

[16] **3-Phase PM synchronous Motor Vector Control using DSP56F80x**, Libor Prokop and Pavel Grasblum, AN1931/D, Motorola 2001

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

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

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

Technical Information Center: 1–800–521–6274

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